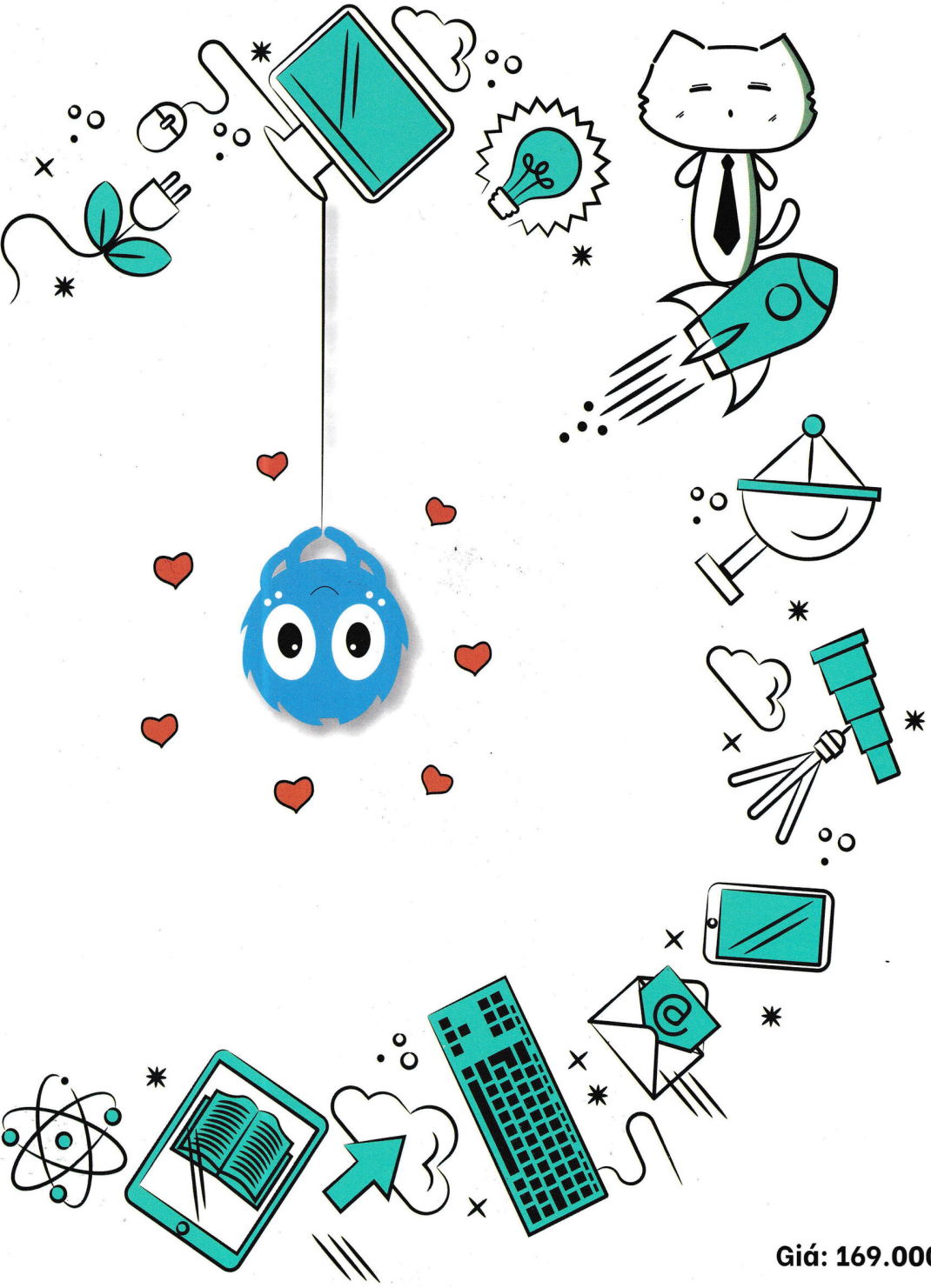
Nhóm tác giả Spiderum & TopCV

Người trong muôn nghề

NGÀNH IT CÓ GÌ?

NHÀ XUẤT BẢN THẾ GIỚI
THẾ GIỚI
spiderum
topcv
Follow your career



Giá: 169.000đ

ISBN 978604345064-4

Một ấn phẩm của spiderum và topcv

Sách **"Người Trong Muôn Nghề - Ngành IT có gì?"** là một ấn phẩm thuộc dự án hợp tác xuất bản từ Spiderum và TopCV.

Người trong muôn nghề

NGÀNH IT CÓ GÌ?

(Đã bán 10.000 bản)

NHÀ XUẤT BẢN THẾ GIỚI

LỜI TỰA

- *Học công nghệ thông tin rồi đi lắp máy tính với cài win dạo à?*
- *Sao lại lao đầu vào cái nghề toàn số má với mấy dòng lệnh khó hiểu thế?*
- *Dân làm máy tính vừa "đụt" vừa khô khan, ai yêu nổi, yêu nổi ai?*
- *Lập trình khó lắm, chắc phải siêu sao toán mới học được.*
- *Làm công nghệ lương nghìn đô như bỡn!*

Đó chắc hẳn là những câu hội thoại, những lời đánh giá mà các bạn đang học/làm Công nghệ Thông tin (CNTT) gặp phải ít nhất 1 lần trong đời, còn những người không theo học/làm CNTT từng nghĩ đến ít nhất 1 lần trong đời (trung thực với bản thân và thừa nhận đi!).

Trong quá trình thực hiện bộ sách hướng nghiệp cho các bạn trẻ, chúng tôi cũng không ngừng trăn trở: Ngành CNTT có thật sự như vậy không? Làm sao để giúp các bạn trẻ có được một cái nhìn toàn diện và khách quan nhất về lĩnh vực rộng lớn này? Những cuốn sách về CNTT hiện có trên thị trường thường là tài liệu nặng tính kỹ thuật dành cho dân chuyên ngành, hoặc những bài viết ngắn với thông tin khá chung chung, hoặc chỉ có góc nhìn từ một cá nhân nên thiếu tính đa chiều. Vậy là nhóm độc giả nằm "lưng chừng" như những bạn học sinh cấp 3, sinh viên Đại học và các bậc phụ huynh đang quan tâm, muốn tìm hiểu về ngành CNTT bị bỏ mặc "bơ vơ", biết đọc gì đây?

Đó là lý do cuốn sách bạn đang cầm trên tay ra đời.

Nằm trong series sách hướng nghiệp của Spiderum và TopCV, **"Người trong muôn nghề: Ngành IT có gì?"** là:

- Cuốn sách đầu tiên trên thị trường đem đến bức tranh toàn cảnh về ngành CNTT cũng như lộ trình phát triển của các vị trí nghề nghiệp phổ biến trong lĩnh vực này.
- Cuốn sách đầu tiên giúp bạn hiểu ngành CNTT không chỉ dừng ở chiếc máy tính hay những dòng code khô khan mà còn rất nhiều những câu chuyện thú vị khác: thiết kế, làm sản phẩm, làm game, làm ứng dụng di động, làm web, gặp gỡ và giao tiếp khách hàng,...
- Cuốn sách đầu tiên "phá vỡ" các định kiến như: *Làm CNTT chỉ toàn những kẻ nhút nhát, "đầu to mắt cận" hay Nghề này không dành cho con gái.* Bạn sẽ thấy, dân IT cũng bạo dạn đi Tây đi Tàu "chinh chiến" thật oách, thấy ngành này chẳng hề khô khan chút nào qua góc nhìn của một người vợ có chồng là lập trình viên, và thấy các bạn nữ cũng không thua kém cánh đàn ông về khoản công nghệ nếu thật sự yêu thích.

Cuốn sách là tập hợp **20** bài viết chứa đựng những chia sẻ "móc hết gan ruột" của các tác giả - những người trực tiếp hoạt động trong lĩnh vực công nghệ. Họ ở đủ mọi độ tuổi, vị trí công việc, địa lý, giới tính: Từ bạn trẻ đang là sinh viên, những người mới vào nghề vài năm tới các đàn anh cỡ 20 năm trong nghề; Từ người học tập và sinh sống tại nước ngoài cho đến bạn xuất thân bình dị từ làng quê nghèo; Từ bậc tiền bối đặt nền móng cho nền CNTT Việt Nam trong lịch sử tới các doanh nhân thời hiện đại gây dựng những công ty công nghệ trị giá hàng triệu, hàng tỷ USD,...

Nội dung sách chia làm 3 chương:

CHƯƠNG 1: TỔNG QUAN NGÀNH IT

Cung cấp thông tin cơ bản về các vị trí nghề nghiệp, mức thu nhập trung bình và những môi trường làm việc đặc thù trong ngành. Ngoài ra, bạn còn được đến với những câu chuyện về lịch sử phát triển của CNTT tại Việt Nam, cũng như nắm bắt các diễn biến trong hiện tại và xu hướng tương lai qua góc nhìn của các Tech Founder.

CHƯƠNG 2: MUÔN NẺO ĐƯỜNG NGHỀ

Đi sâu vào từng vị trí việc làm cụ thể thông qua trải nghiệm thực tế của những người trong ngành, như: Đặc điểm, vai trò của từng loại công việc; Các kiến thức, kỹ năng các bạn cần chuẩn bị và những bài học, kỷ niệm vui, buồn trong nghề.

CHƯƠNG 3: HÀNH TRANG VÀO NGÀNH

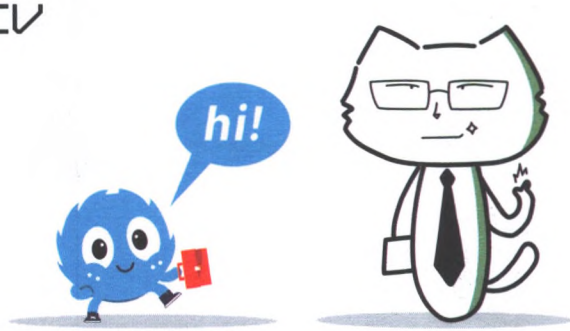
Những "trang bị" cần thiết cả về mặt thái độ, tâm lý lẫn kiến thức để bạn có thể phát huy tối đa khả năng trong hành trình chinh phục thế giới công nghệ.

Là sản phẩm của hơn 4 tháng lao động miệt mài, chúng tôi tin cuốn sách sẽ giúp bạn gạt bớt những hoang mang, rối ren khi chọn ngành, chọn nghề thông qua việc nắm bắt tổng quan về lĩnh vực IT cũng như thấu hiểu chính bản thân.

Nếu bạn đang vướng mắc với câu hỏi: "Liệu mình có nên chọn ngành CNTT?", đây là cuốn sách dành cho bạn.

Đừng chần chừ, hãy đến với những trang sách tiếp theo để khám phá thế giới công nghệ đầy màu sắc. Một hành trình tuyệt vời đang chờ bạn phía trước!

Spiderum & TopCV



CÁC TÁC GIẢ VÀ KHÁCH MỜI

Sách **"Người trong muôn nghề: Ngành IT có gì"** - Spiderum & TopCV trân trọng gửi lời cảm ơn tới:

Nguyễn Chí Công
Giám đốc Bảo tàng CNTT

Nguyễn Lê Thành
Chief Security Officer VNG

Lê Hồng Minh
Chủ tịch VNG

Nguyễn Doãn Minh Giang
Founder & CEO OWS Việt Nam

Thi Măng Cụt
Founder icetea.io

Lợi Lưu
CEO & Co-founder Kyber Network

Curly Rae Braces
Software Engineer

Scarlet
Sinh viên trường UBC, Canada

Đỗ Tuấn Anh
CEO Appota

Nguyễn Nhật Hoàng
Engineering Manager ShopBack

Tống Tùng Giang
Gameplay Programmer

Nguyễn Quang Uy
AI Research Scientist, GRDAI, VNG

Husky
Product Owner

Hoàng Nguyễn
Co-founder & Design Coach GEEK Up

Doãn Thiện
Co-founder lovelock.one

Giang Lê
Operation Manager Tek Experts

Thu Phạm
Vợ anh dev Spiderum

Hien Nguyen
Software Engineer

Phạm Bình
Full-stack Developer

ĐỘI NGŨ THỰC HIỆN

Series sách: **"Người trong muôn nghề"** thuộc Dự án xuất bản của spiderum & topcv

ĐIỀU PHỐI DỰ ÁN

Trần Việt Anh
Sáng lập Spiderum

Trần Trung Hiếu
Sáng lập TopCV

THIẾT KẾ & MINH HOẠ

Dung Nguyễn
Trưởng nhóm thiết kế

Khang
Minh họa

NỘI DUNG

Nga Levi

Dũng Ez

TRUYỀN THÔNG

Vũ Khuê

Isa Quan

KỸ THUẬT

Nguyễn Tuấn Hưng

Nguyễn Mạnh Tuấn

MỤC LỤC

I: Tổng quan ngành IT

III: Hành trang vào ngành

Phụ lục

II: Muôn nẻo đường nghề

1

TỔNG QUAN NGÀNH IT

& Ngành IT
những câu hỏi thường gặp

Biên soạn bài viết: **Ban biên tập**

Đứng trước việc lựa chọn hướng đi phát triển sự nghiệp trong lĩnh vực Công nghệ thông tin (CNTT), các bạn trẻ luôn đặt ra nhiều câu hỏi: "Ngành này có những lĩnh vực nào?", "Học CNTT xong đi làm nghề gì?", "Mức lương cho các vị trí công việc ra sao?",... Trong bài viết này, Spiderum (Nền tảng chia sẻ kiến thức và thảo luận cho người trẻ) và TopCV (Hệ sinh thái hỗ trợ tuyển dụng toàn diện dành cho ứng viên và doanh nghiệp) sẽ cung cấp những thông tin tổng quan với hy vọng giúp bạn hiểu rõ hơn về ngành CNTT để đưa ra lựa chọn nghề nghiệp phù hợp.

Làm CNTT là làm gì?

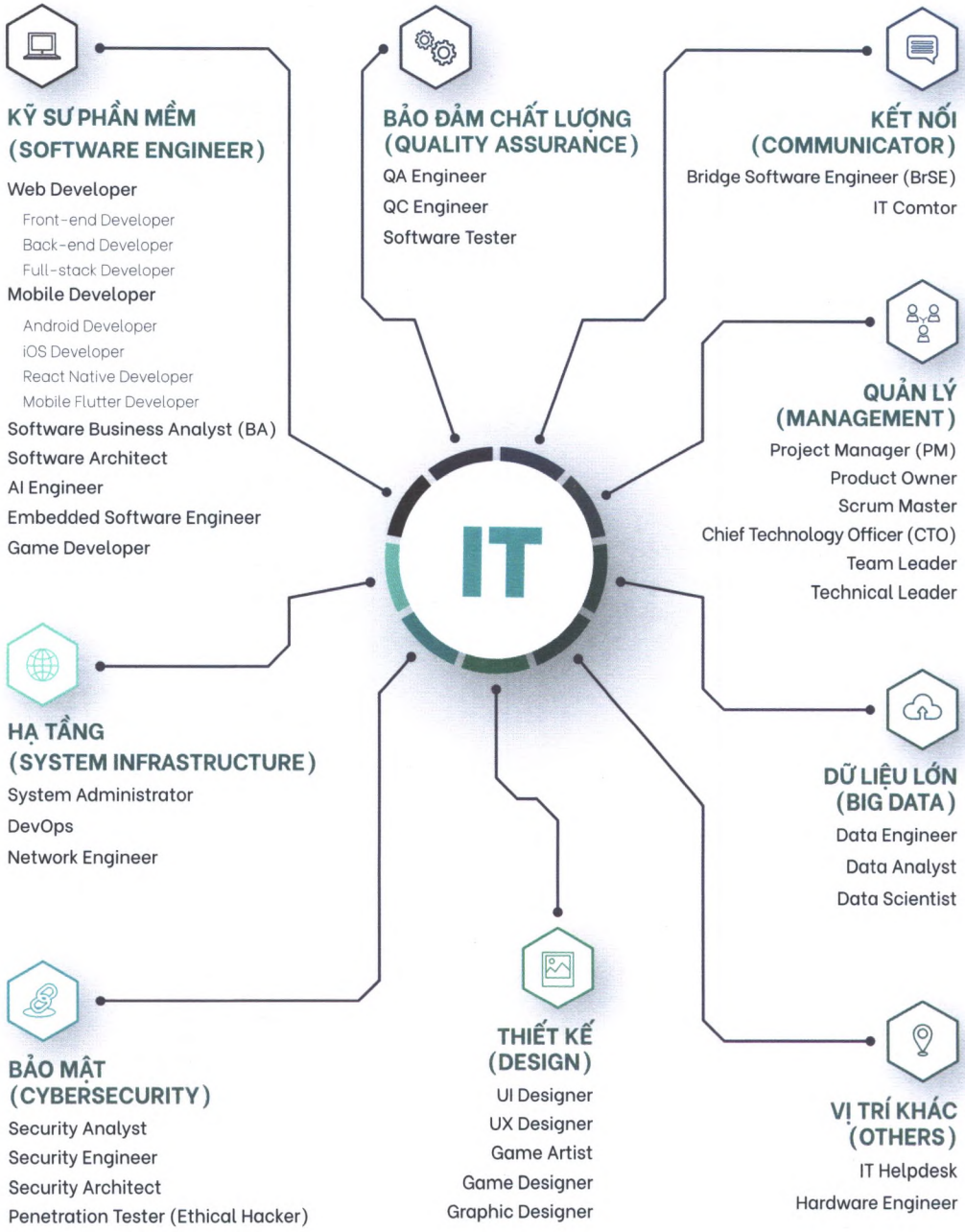
Theo trang WordNet của Đại học Princeton, Công nghệ thông tin (tiếng Anh là IT: Information Technology) là một nhánh ngành kỹ thuật sử dụng máy tính và viễn thông để thu nhận, lưu trữ và truyền tải thông tin. Thuật ngữ này xuất hiện lần đầu năm 1958 tại bài viết của 2 tác giả Harold J. Leavitt và Thomas L. Whisler trên tạp chí Harvard Business Review. Trong đó, họ viết: "Công nghệ mới này vẫn chưa có tên riêng. Chúng tôi sẽ gọi nó là công nghệ thông tin"[1].

Ở Việt Nam, khái niệm CNTT được hiểu và định nghĩa trong nghị quyết 49/CP ký ngày 04/08/1993 về phát triển công nghệ thông tin của Chính phủ: "Công nghệ thông tin là tập hợp các phương pháp khoa học, các phương tiện và công cụ kỹ thuật hiện đại - chủ yếu là kỹ thuật máy tính và viễn thông - nhằm tổ chức khai thác và sử dụng có hiệu quả các nguồn tài nguyên thông tin rất phong phú và tiềm năng trong mọi lĩnh vực hoạt động của con người và xã hội."

"Vậy làm công nghệ thông tin là làm gì?"

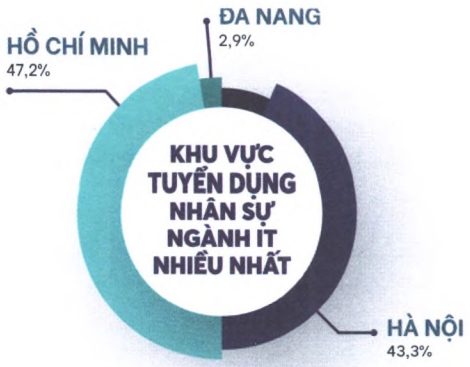
[1] *https://hbr.org/1958/11/management-in-the-1980s*

Đây là thắc mắc đội ngũ biên tập chúng tôi nhận được rất nhiều từ các bạn trẻ. Khó có thể trả lời câu hỏi này một cách trọn vẹn bởi CNTT là lĩnh vực vô cùng đa dạng các loại hình công việc, chưa kể, phụ thuộc vào từng mô hình công ty mà những vị trí trong thực tế có sự chồng chéo lẫn nhau. Dưới đây là sơ đồ phân chia một cách cơ bản những công việc điển hình trong ngành.

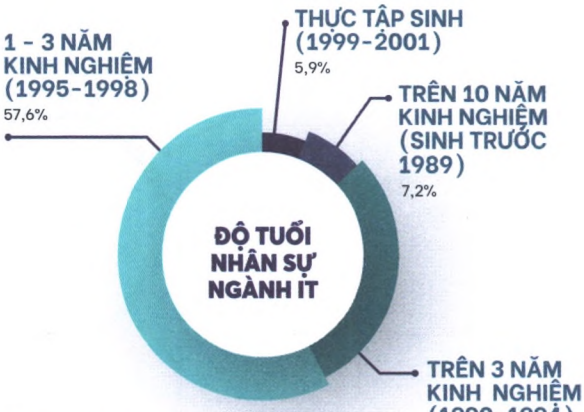


Những vị trí công việc điển hình trong ngành IT

Sơ đồ trên có thể đem lại cho các bạn một cái nhìn tổng quan nhất về những lĩnh vực công việc nằm trong nhóm ngành CNTT. Có thể thấy rằng, công việc trong ngành IT không bó hẹp trong mảng lập trình như nhiều bạn vẫn lầm tưởng mà còn có nhiều vị trí thú vị khác như: Làm sản phẩm (các vị trí liên quan tới product), Thiết kế (Design), Là cầu nối giữa khách hàng với đội ngũ phát triển (BrSE, IT Comtor), Đảm bảo chất lượng (QA, QC) hay Hỗ trợ hệ thống (IT Helpdesk),... Như vậy ngành IT không hề khô khan như nhiều người lầm tưởng mà cũng có rất nhiều công việc cần đến sự sáng tạo, kỹ năng giao tiếp tốt, và khả năng thấu hiểu vấn đề của người dùng/khách hàng. CNTT không nhất thiết chỉ dành cho những người "đầu to mắt cận" hoặc chỉ phù hợp với các bạn nam như một số định kiến từ trước tới nay.



Thời điểm ghi nhận: 5/2020
(Tổng hợp dựa trên tổng số tin đăng tuyển trên các kênh tuyển dụng IT lớn nhất tại Việt Nam: ITViec, Vietnamworks, CareerBuilder, TopDev, TopCV)



Dữ liệu tính từ 2017 - 5/2020
(Theo thống kê của TopCV)

Nhân lực ngành CNTT có những điểm nào đáng chú ý?

Bước vào thời đại số hóa, công nghệ và khoa học kỹ thuật được xem là động lực lớn thúc đẩy phát triển kinh tế, xã hội.

Bao gồm nhóm lập trình viên, tester, QA, design, BA, nhân sự làm trong ngành IT nói chung,..
(Theo thống kê của TopCV)

Từ khá lâu, CNTT đã trở thành ngành học được chú trọng trong hệ thống đào tạo của các trường Đại học. Những năm gần đây, ngành này cũng là lựa chọn được nhiều bạn trẻ quan tâm bởi cơ hội việc làm cao và mức thu nhập được đánh giá là tốt hơn so với trung bình xã hội. Theo báo cáo thị trường IT Việt Nam năm 2020, top 3 lý do khiến một người làm trong ngành CNTT cảm thấy hài lòng về công việc là: Được đào tạo bài bản & phát triển kỹ năng chuyên sâu; Lộ trình phát triển sự nghiệp rõ ràng; Mức lương, thưởng, đãi ngộ xứng đáng.

Nhu cầu tuyển dụng trong ngành CNTT năm 2019 tăng gần 2,5 lần so với năm 2018, dự báo năm 2020 sẽ tiếp tục tăng (số lượng nhân sự tuyển dụng cao gấp 1,6 lần năm 2019). Theo các số liệu về thị trường CNTT, năm 2020, Việt Nam cần hơn 400.000 nhân lực và dự kiến năm 2021 là 500.000. Hiện tại, mỗi năm các cơ sở đào tạo ra khoảng 50.000 nhân sự cho ngành, nhưng chỉ 30% số đó đáp ứng đủ kỳ vọng của doanh nghiệp. Sinh viên ra trường chưa đạt yêu cầu làm việc chuyên môn, hơn nữa, các kỹ năng mềm (teamwork, giao tiếp, quản lý thời gian,...) còn yếu. Nguyên nhân của vấn đề này xuất phát từ nhiều phía: do chương trình đào tạo không đúng với nhu cầu doanh nghiệp, tính đổi mới của giáo trình không theo kịp tốc độ thay đổi của công nghệ, sinh viên chưa chủ động trong việc tự học và đi làm,... Do đó, thị trường nhân lực trong ngành này rơi vào tình trạng cung không đủ cầu. Điều đó cũng đồng nghĩa với việc những bạn trẻ có khả năng tìm hiểu thông tin, am hiểu xu hướng việc làm trong ngành, biết trau dồi cho mình những kỹ năng phù hợp sẽ có rất nhiều cơ hội rộng mở. Cuốn sách này của Spiderum và TopCV hi vọng sẽ là một trong các nguồn thông tin tham khảo hữu ích giúp bạn làm được điều đó.

Mức lương của các vị trí phổ biến trong ngành CNTT (dựa theo kỹ năng/ngôn ngữ lập trình) thường được xem là cao hơn so với mặt bằng chung của các ngành nghề khác. Tuy nhiên cần lưu ý rằng mức lương sẽ có sự biến động rất lớn tuỳ thuộc vào kinh nghiệm, năng lực chuyên môn và môi trường làm việc. Không nên ảo tưởng rằng làm ngành IT thì ra trường đương nhiên phải có lương cao. Bảng so sánh các mức lương đưa ra dưới đây được phân thành các mức trung bình thấp, trung bình cao, và trung bình chung của các vị trí. Bạn nên nhớ rằng những con số ở đây chỉ có tính chất tham khảo tương đối, có thể mức lương sẽ rất khác tuỳ thuộc vào bản thân bạn.

	MỨC LƯƠNG TB THẤP NHẤT $	MỨC LƯƠNG TB CAO NHẤT $$$	MỨC LƯƠNG TB $$
mobile	11	22,2	17,5
ios	11,2	23,1	18,2
android	11,3	23	17,8
flutter	11,5	22,7	18,5
front-end	10,4	20	16
back-end	14	25	20,7
fullstack	14,5	28,2	22,3
devops	11,2	22	17,4
brse	29,2	49,2	43,5
business analyst	10,2	19,5	15,2
designer	10,5	17,7	14,6
ui/ux	11,8	21,6	17,5
game	10,4	20,2	16,2
tester	9	16	12,9
php	10,1	19,2	15,2
javascript	11,9	23,2	18,6
.net	10,9	20,8	16,5
java	12,2	24,6	19,7
c/c++	10,9	21	16,4
c#	11,2	22,3	17,4
nodejs	12,3	24,3	19,2
python	13,8	26	21
ruby	14,6	30,2	23,9
laravel	10,9	23,3	18,5
react native	10,9	21,8	17,7
angularjs	13,4	24,8	20,7
vuejs	13,6	26	21
unity	11,7	20,6	16,9
wordpress	9,1	15,4	12,5
magento	10,4	20,9	16,2
odoo	13,3	22,9	18,9
abap	10,5	12,3	12,2

Mức lương (triệu VNĐ) phân bổ theo các kỹ năng/ngôn ngữ lập trình, dành cho nhóm từ 1 - 3 năm kinh nghiệm
(Theo thống kê của TopCV)

Ngành CNTT và những câu hỏi thường gặp?

Dựa trên những câu hỏi khảo sát bạn đọc từ mọi miền đất nước của Spiderum cùng kinh nghiệm trong mảng tuyển dụng, đào tạo nhân lực từ TopCV, chúng tôi đã biên soạn một bộ Q&A chi tiết nhằm giải đáp thắc mắc xoay quanh câu chuyện hướng nghiệp của các bạn trẻ.

Học CNTT sau này đi làm nghề sửa máy tính có phải không?	Không phải, CNTT có rất nhiều nhánh khác nhau và hầu hết không làm nghề sửa máy tính.
Học CNTT có cần giỏi Toán không?	Câu trả lời là: "Có" và "Không", tuỳ thuộc vào yêu cầu công việc. Có công việc bạn chỉ sử dụng đến toán học lớp 5 nhưng cũng có công việc cần sử dụng tới đại số tuyến tính. Tuy nhiên, không thể phủ nhận nếu học Toán tốt bạn sẽ có lợi thế khi học lập trình, và nếu muốn làm những công việc đang "hot" như AI Engineer (Trí tuệ nhân tạo) thì Toán là yêu cầu bắt buộc nhé.
Học sinh cấp 3 cần những gì cho ngành CNTT?	Trong thời gian học THPT bạn có thể thử sức với một số môn liên quan tới ngành CNTT như lập trình Pascal, Cơ sở dữ liệu trên Microsoft Access. Ngoài ra bạn có thể tự mày mò, học thêm trên Internet để hiểu rõ hơn về ngành.
Nên học lập trình ngôn ngữ gì để kiếm được nhiều tiền?	Không có ngôn ngữ lập trình nào đảm bảo có thể kiếm tiền được nhiều hơn các ngôn ngữ lập trình khác. Mục đích của lập trình là giải quyết vấn đề bằng máy tính, bạn giải quyết được vấn đề càng khó thì càng ra tiền. Vậy để kiếm được nhiều tiền, bạn cần kỹ năng lập trình tốt và kinh nghiệm, còn tùy vào công việc bạn muốn làm sẽ có các ngôn ngữ tương ứng để bạn lựa chọn.
Nên học lập trình ở đâu? Ở trung tâm hay ở trường Đại học?	Nếu có thể bạn nên học ở trường Đại học vì bạn sẽ được đào tạo đầy đủ nhất các kiến thức nền tảng. Ngoài ra, bạn cũng có thể kết hợp học thêm ở trung tâm và tự học thêm ở nhà.

Có nên học các khóa đào tạo CNTT ngắn hạn?	Nếu có điều kiện, bạn có thể đăng ký học thêm các khóa đào tạo CNTT nếu nó giúp bạn tiết kiệm thời gian, nhưng không nên phụ thuộc vào các khóa học đó hoặc đi học quá nhiều khoá học nhỏ lẻ. Phần lớn các bạn học lập trình tốt ở trên trường ĐH và có kỹ năng tự học tốt không cần tham gia các khóa đào tạo ngắn hạn.
Học CNTT thì nên học trường Đại học nào?	Bạn có thể học bất kỳ trường Đại học nào cũng được vì chương trình đào tạo của các trường không quá khác biệt. Việc học giỏi hay không nằm ở thái độ học tập của bạn, ngôi trường bạn học không đảm bảo được điều đó. Vậy nên, bạn có thể lựa chọn một trường mà bạn yêu thích và vừa sức để thi đỗ.
Đi làm có cần bằng Đại học không?	Bằng Đại học không phải yêu cầu bắt buộc để đi làm. Doanh nghiệp cần kiến thức và kỹ năng của bạn chứ không phải tấm bằng. Tất nhiên, nếu không học Đại học, bạn cũng phải xây dựng cho bản thân một kế hoạch học tập khác, vì không cần bằng chứ không phải không cần học nhé.
Bằng Đại học loại giỏi có quan trọng không?	"Có" và "Không". Phần lớn các công ty tuyển dụng không yêu cầu bạn phải có bằng giỏi nhưng nếu bạn đạt bằng giỏi, bạn sẽ có những lợi thế cạnh tranh nhất định so với các bạn khác. Đặc biệt khi mới ra trường, nếu bạn không có kinh nghiệm thực tế thì bằng Đại học và bảng điểm chính là cái nhìn đánh giá đầu tiên. Ngoài ra, nếu tiếp tục học lên cao hơn, bằng giỏi có thể giúp bạn thêm điểm cộng khi xin học bổng.
Có nên đi làm sớm trong quá trình còn ngồi trên ghế nhà trường hay không?	Câu trả lời là: "Có". Bạn nên đi làm để có thêm kinh nghiệm thực tế. Tất nhiên, bạn phải cân đối để không ảnh hưởng tới việc học tập và nên chọn công việc đúng với chuyên ngành học của mình.
Sinh viên học trái ngành nhưng muốn chuyển sang làm CNTT được không?	Đầu tiên, bạn phải tìm hiểu rõ hơn về ngành CNTT để tránh bị "trái ngành" thêm một lần nữa. Sau khi đã chắc chắn muốn chuyển về CNTT, bạn cần quyết tâm đủ lớn để làm lại từ đầu. Nếu bạn muốn có sức cạnh tranh tương đương những bạn học đúng chuyên ngành, bạn hãy đầu tư thời gian và công sức tương đương để bù đắp lại các kiến thức bị thiếu. Ngoài ra, cũng có những vị trí không yêu cầu kiến thức kỹ thuật chuyên sâu nên có thể không cần học đúng ngành ngay từ đầu, như Hỗ trợ/Chăm sóc khách hàng thậm chí là BA, Product,...

Mức độ quan trọng của thuật toán với người học lập trình?	Giải thuật là công việc bắt buộc trong lập trình. Tuy nhiên, thuật toán không phải là cái gì đó quá cao siêu như các bài thi Olympic tin học. Ví dụ, bạn đưa ra một cách thức tính tổng các số chẵn trong một dãy số cũng là giải thuật toán rồi. Còn nếu bạn mong muốn ứng tuyển vào các công ty công nghệ lớn, khả năng giải thuật của bạn phải "siêu" thì mới có cơ hội qua được bài kiểm tra đầu vào nhé.
Mức độ quan trọng của ngoại ngữ (Tiếng Anh) với người học lập trình?	Với công việc lập trình, tiếng Anh là kỹ năng bắt buộc. Ở mức độ tối thiểu lập trình viên phải có khả năng đọc hiểu tài liệu tiếng Anh để phục vụ cho chính việc học lập trình của bản thân và tìm kiếm các giải pháp trong quá trình làm việc. Nói vui: "Khi bạn tìm được một bản dịch tiếng Việt thì thế giới đã chuyển sang sử dụng công nghệ mới rồi!"
Nên chọn một công nghệ hot để đáp ứng nhu cầu thị trường hay tập trung trau dồi kiến thức nền tảng?	Nếu không có kiến thức nền tảng thì chưa chắc bạn có thể hiểu và sử dụng được một công nghệ "hot". Lời khuyên là bạn nên học tốt kiến thức nền tảng ngay từ đầu để sau đó không gặp trở ngại khi tiếp cận các công nghệ mới.
Xin việc ngành CNTT có khó không?	Xin việc ngành công nghệ thông tin không khó miễn là bạn đủ năng lực, bởi lẽ CNTT đang là một trong những ngành dẫn đầu về số lượng cơ hội việc làm cũng như mức thu nhập hấp dẫn. Tuy nhiên, đừng hiểu nhầm là "cứ học CNTT ra trường sẽ có việc" vì dù bạn có học CNTT hay bất kỳ ngành nào khác, doanh nghiệp cũng sẽ có những yêu cầu tối thiểu về năng lực.
Cơ hội thăng tiến nghề nghiệp trong ngành CNTT thế nào?	Cơ hội thăng tiến trong ngành CNTT rất rộng mở, tất cả phụ thuộc vào năng lực của bạn. Từ vị trí bắt đầu như thực tập sinh lập trình đến vị trí cao cấp như Project Manager (PM) hay Giám đốc công nghệ (CTO) cũng có nhu cầu tuyển dụng rất lớn.
Xu hướng tương lai của ngành CNTT? (Nhánh nào phát triển/bão hòa?)	Hiện tại nhu cầu tuyển dụng của ngành CNTT vẫn rất lớn nên khi chọn ngành này, bạn không phải quá bận tâm về vấn đề nhánh nào bão hoà, nhánh nào phát triển. Nếu bạn có kiến thức nền tảng vững chắc, việc chuyển dịch theo nhu cầu thị trường sẽ không ảnh hưởng quá nhiều. Trong 5 năm tới, khó có thể dự đoán chính xác ngành CNTT thay đổi nhanh thế nào, tuy nhiên, xu hướng cho thấy rõ sự lên ngôi của các công nghệ Big Data, AI, Blockchain, IoT.

Lược sử ngành CNTT Việt Nam

CHUYỆN CŨ CÙNG NHỮNG BÀI HỌC CHƯA BAO GIỜ CŨ

Tác giả: **Nguyễn Chí Công**
Giám đốc Bảo tàng CNTT
Nguyên Giám đốc Trung tâm Hệ thống Thông tin ISC,
Viện nghiên cứu công nghệ quốc gia

Thế hệ trẻ lớn lên trong thời đại số có lẽ đã quen thuộc với những món đồ công nghệ nhỏ gọn, tiện lợi và thời thượng, được bày bán khắp nơi. Ngược lại lịch sử, có lẽ ít người trong chúng ta nghĩ đến những thiết bị cồng kềnh, khó sử dụng và rất khan hiếm. Lịch sử ngành CNTT đã từng bước qua thật nhiều nốt thăng trầm để đạt tới những tiến bộ như ngày nay. Hãy cùng đi qua những dấu mốc lịch sử đáng nhớ của ngành CNTT Việt Nam cùng chú Nguyễn Chí Công - người tham gia chế tạo máy tính đầu tiên của Việt Nam và cũng là một trong những thành viên đầu tiên sáng lập Tập đoàn FPT.

Chặng đường dài cùng vô số những kỷ niệm...

Mùa đông năm 2007, tôi may mắn mời được ông thầy cũ Alain Teissonnière cùng sang thăm vài thành phố Trung Quốc và về sống tại gia đình tôi đến hết Tết Mậu Tý. Thầy tặng tôi mấy cuốn sách mới in, nhân tiện lướt xem bìa và hỏi về nội dung của những tác phẩm vừa xuất bản tại Việt Nam đang có trong nhà tôi. Thầy chưa từng nhắc, nhưng tôi không thể nào quên được công lao thầy đã móc nối các giáo sư tại Ecole Estienne Paris (Trường Cao đẳng Mỹ thuật và Công nghiệp Đồ hoạ Paris) giúp đỡ chúng tôi thực hiện các lớp đào tạo ngắn ngày ở Hà Nội trong thập niên 1990, mà nhờ đó, có nhiều học viên đồ hoạ máy tính và chế bản điện tử về sau rất thành đạt.

Nhân dịp Tết, tôi dẫn thầy đến gặp một số người quen cũ ở Viện CNTT và các cơ quan khác. Thầy rất vui vì cuộc sống riêng của họ đã no đủ hơn thầy. Đó là những anh em còn liên lạc được trong số hàng trăm người xưa kia ít nhiều đã được thầy và giáo sư công huân[1] Henri Van Regemorter, Chủ tịch CCSTV (Ủy ban vì sự Hợp tác Khoa học và Kỹ thuật với Việt Nam), tìm cho nơi nhận thực tập sinh hoặc nghiên cứu sinh ở Pháp. Thầy đã giúp đỡ hết mực tận tình trong những ngày chúng tôi xa quê và sẵn sàng trả lời mỗi bức thư chúng tôi gửi đến cả khi chúng tôi đã trở về nước.

Lâu lắm mới có dịp, các cuộc trò chuyện thường kéo dài và chúng tôi nhớ lại những lần cùng nhau làm việc tại các cơ quan dân sự của Pháp và Việt Nam theo một thoả thuận hợp tác song phương được ký bởi lãnh đạo hai bên sau 1975, dù nước ta đang bị đa số quốc gia phương Tây cấm vận. Thời ấy, chúng tôi thường phải mất gần 1 năm kiên trì trao đổi thư từ chỉ để sắp xếp được một đợt công tác dù ngắn ngủi hay dài hạn. So sánh ngày tháng ghi trong các bức thư và in trên tem dán ngoài phong bì sẽ thấy nếu đạp xe đạp, ta có thể đi và về giữa hai thủ đô còn nhanh hơn bưu điện.

Thầy bảo loài người trên thế giới ngày nay đã gần nhau hơn rất nhiều nhờ có mạng Internet và các ứng dụng CNTT. Chỉ cần lấy ví dụ về việc chuẩn bị chuyến đi của thầy lần này: hai bên gửi vài e-mail trong vòng 3 tuần lễ là giải quyết suôn sẻ mọi thủ tục lớn nhỏ, từ thị thực đến vé máy bay.

Thầy nhớ trước thập niên 1990, các nhà khoa học nước ngoài phải di chuyển bằng xe của cơ quan đối tác vì Hà Nội chưa có taxi. Tôi trả lời thầy rằng đời sống đô thị cực kỳ khó khăn vào thời "bao cấp" ấy, các nhu yếu phẩm như gạo, thịt, thuốc lá, vải, xà phòng,... chỉ được cung cấp theo tem phiếu với định lượng tối thiểu. Từ 1976 đến 1986 tại miền Bắc, cả nông nghiệp và thương nghiệp đều do các hợp tác xã và nhà nước nắm giữ. Tôi nhắc đến chuyến tôi cùng vợ đưa thầy về quê ngoại thăm những ngôi chùa cổ tuyệt đẹp xứ Kinh Bắc đang xập xệ mà không có tiền sửa chữa, làng xóm xác xơ chỉ còn nghe những tiếng cười vô tư của trẻ em...

Thầy kể lại lần từ Paris phải vòng vèo qua nước Nga, sau một ngày mệt rũ mới đến được sân bay Hà Nội vẫn còn đầy hố bom. Thầy ngạc nhiên vô cùng vì chính trong tháng cuối cùng của năm 1976 đáng nhớ ấy, những chàng trai và cô gái Việt đã nhanh chóng nắm bắt được công nghệ thông tin

[1] *công lao lớn*

tiên tiến nhất để làm ra chiếc máy vi tính đầu tiên của Việt Nam ngay trước thềm Tết Đinh Tỵ 1977.

Những kỷ niệm về một thời đói rét nhưng đáng nhớ như vậy phải hàng nghìn trang nữa mới có thể ghi hết, bởi trong lứa thanh xuân đó không chỉ có chúng tôi mà còn biết bao người khác.

Thầy Alain đã mất vì ung thư năm 2009, lớp người chúng tôi cũng sắp theo thầy. Nhưng tôi hoàn toàn tin tưởng các thế hệ tiếp nối sẽ làm được những kỳ tích lớn hơn bởi ngoài truyền thống dũng cảm của cha anh để lại, các bạn còn có những công cụ và tri thức của thời đại mới. Để kịp thời nắm lấy những cơ hội sẽ xuất hiện, trước hết ta cần rút ra các kinh nghiệm sâu sắc từ những bước đã đi qua trên con đường tiến bộ của CNTT.

Ngành CNTT Việt Nam - bao con người ghi dấu những cột mốc

Quãng đường gần nửa thế kỷ từ khi tốt nghiệp, đi làm rồi về hưu nhưng vẫn tiếp tục hoạt động đúng chuyên môn cho đến nay đã để lại trong tôi rất nhiều cảm xúc khó mà kể hết. Trải qua bao gian nan, vượt qua thất bại để thành công, tôi xin chia sẻ chút kinh nghiệm rời rạc nhưng đều liên quan đến CNTT và có thể có ích đối với các bạn trẻ.

Mạo hiểm chấp nhận rủi ro để hướng tới đổi mới

Ngành CNTT ra đời tại Việt Nam vào thập niên 1960 với những con người đầu tiên biết vận hành, nghiên cứu, sử dụng máy tính điện tử nhập khẩu từ 2 siêu cường Mỹ và Liên Xô (cũ) trong cuộc chiến tàn khốc nhất thế kỷ XX mà dân tộc ta phải trải qua. Trong giai đoạn đất nước còn chiến tranh, đầu tư vào CNTT là một bước đi có thể coi là rủi ro. Nhưng đôi khi để đạt được thành tựu thì rủi ro là điều ta luôn phải chấp nhận và phải mạnh dạn. Điều này đúng ở cả bình diện đất nước lẫn cá nhân tôi. Tôi cho rằng, việc sớm chọn đúng nghề có tầm quan trọng như khởi nghiệp trong cuộc đời lao động và trưởng thành của mỗi cá nhân.

Đầu năm 1966, chiến tranh diễn ra khắp đất nước, tôi chuẩn bị vào Đại học, cha tôi nói rằng kỹ thuật hiện đại là thứ rất cần khi hoà bình trở lại và sẽ phải tiến hành công nghiệp hoá ở Việt Nam. Cuối năm ấy, tôi du học xa quê hương thì cha mất. Nỗi đau lớn đã làm tôi nhớ lại lời khuyên quý báu của ông và tìm hiểu các ngành học mới ra đời. Vốn sớm biết ngoại ngữ và có nhiều thông tin từ nước ngoài, đồng thời học khá đều các môn khoa học tự nhiên và xã hội, tôi thấy mình có thể xin học CNTT, lúc đó là một trong những ngành chưa phổ biến nên có rất ít sinh viên ghi tên. Ngược lại, đối với tôi, những gì hiếm và lạ đều hấp dẫn. Tôi được nhận vào trường CVUT (Đại học Kỹ thuật Séc) tại thủ đô Praha. Sau 2 năm học các môn đại cương, tôi chọn chuyên khoa Máy tính vì thấy khoa thành lập chưa lâu, phù hợp với mong ước của tuổi trẻ thích mạo hiểm và tìm tòi những điều mới lạ. Năm 1972, tôi tốt nghiệp kỹ sư máy tính cùng hơn 20 người nước ngoài và đi thực tập tại Xí nghiệp sản xuất máy tính ARITMA.

Chỉ 6 năm sau chiến thắng Điện Biên, giáo sư Tạ Quang Bửu (1910-1986) đã lập dự án xây dựng một trung tâm tính toán ở Uỷ ban Khoa học và Kỹ thuật nhà nước. Trở về Hà Nội cuối tháng 4 -1973, tôi được nhận vào làm việc ở Phòng Máy Tính tại Ủy ban này. Nơi đây đặt Minsk-22, dàn máy tính của

Liên Xô (cũ), được vận hành từ 1968 bởi những nhà nghiên cứu, kỹ sư, công nhân tài giỏi với vị thủ trưởng đáng mến là Tiến sĩ Khoa học Phan Đình Diệu (1936 - 2018). Chính đội ngũ đó và đồng nghiệp từ các cơ quan khác đã giúp đào tạo hầu hết những người học CNTT trong nước trên chiếc máy tính Minsk-22, trước khi miền Bắc Việt Nam nhập thêm các máy khác. Đội ngũ đã xử lý được rất nhiều bài toán ứng dụng dân sự và quân sự đương thời, như tổng hợp và phân tích số liệu từ cuộc điều tra dân số đầu tiên của nước Việt Nam thống nhất (1976), ứng dụng kỹ thuật vi xử lý trong điều khiển công nghiệp, hỗ trợ việc đào tạo và trang bị các máy VT81 cho hàng chục cơ quan nhà nước (1979). Một số ứng dụng tiêu biểu khác có thể kể tới như hệ thống tin học phục vụ thăm dò biển ở Tổng cục dầu khí, hệ đo và truyền dẫn dữ liệu điện năng trên đường dây cao thế, hệ theo dõi tham số môi trường tại lăng Chủ tịch Hồ Chí Minh, hệ bảo vệ kho Công trường Thuỷ điện Hoà Bình, hệ quản lý vật tư ở Nhà máy Cơ khí Hà Nội và Xí nghiệp Điện tử Tân Bình. Đặc biệt, Phòng Kỹ thuật số kết hợp với Phòng Lập trình đã thành công trong chuyến đi năm 1981 vào TP. Hồ Chí Minh để làm phần mềm quản lý vật tư cho Xí nghiệp Máy may Sinco.

Quãng thời gian đất nước đi từ chiến tranh rồi tới ngày hoà bình lập lại đó, hầu hết mọi người chỉ nghĩ làm sao cho cuộc sống đỡ vất vả hơn, làm sao có đủ ăn? CNTT là một thứ gì đó thật xa vời và hoang đường, vậy nhưng tôi cảm thấy mình thật may mắn vì đã là một trong những người đầu tiên ứng dụng thành tựu của CNTT vào giải quyết các vấn đề trong đời sống. Lúc đó chúng tôi không dám mơ là ngành này sẽ đem tới sự giàu có, mà chỉ biết rằng mình đã chọn đúng con đường của bản thân vì được nghiên cứu, tìm tòi, sáng tạo ra những thứ mới mẻ và hữu ích.

Ngày 27 - 12 - 1976, tiền thân Viện CNTT là Viện Khoa học Tính toán và Điều khiển (KHTT&ĐK) ra đời sau khi Ủy ban KH và Kỹ thuật Nhà nước sáp nhập Phòng Máy tính với Ban Điều khiển học. Ngay tháng 1 - 1977, tại chân một gò đất gọi là Đồi Thông, Hà Nội, Phòng Kỹ thuật số của tôi đã tiếp thu nhanh chóng các bài giảng của thầy Alain Teissonnière và chế tạo được máy vi tính VT80 dựa trên chip vi xử lý Intel 8080A nối với các bộ phận điện tử khác bằng cách quấn dây. Thành tích vang dội này lập tức thu hút sự chú ý vượt ra ngoài giới tin học.

Cũng trong năm 1977, theo kế hoạch của cơ quan, tôi đang học một ngoại ngữ khác để chuẩn bị làm nghiên cứu sinh khoa học, bỗng nhiên được thủ trưởng thông báo có suất đi thực tập kỹ thuật vi xử lý ở CH Pháp trong 9 tháng. Tôi đã đồng ý thay đổi kế hoạch dù biết đi ngắn như thế thì không thể có học vị và những lợi ích cá nhân khác (do được sống ở nước ngoài tới 4 năm trong thời kỳ Việt Nam lâm vào khủng hoảng kinh tế). Tôi dự đoán vi xử lý sẽ nhanh chóng làm thay đổi các ngành công nghiệp bằng các ứng dụng linh hoạt với chi phí nhỏ hơn nhiều so với trước kia. Dự báo ấy thực ra vẫn còn chưa đủ tầm nhìn xa, bởi chỉ 5 năm sau, mọi mặt của xã hội đều có thể bị máy vi tính xâm nhập. Rất may, tôi đã nắm bắt sớm và không để cơ hội tuột tay.

Thời gian đó, tôi may mắn được chọn làm thực tập sinh tin học Việt Nam đầu tiên tại một trung tâm nghiên cứu của EDF (công ty Điện lực Pháp) và đã thiết kế máy vi tính VT81 dựa trên

Cảm giác chấp nhận những rủi ro và là người tiên phong tiếp thu kiến thức mới, tạo ra cũng như chứng kiến những ứng dụng công nghệ trực tiếp gây ảnh hưởng tích cực đến đất nước là niềm hạnh phúc khó có thể diễn tả bằng lời.

chip vi xử lý Intel 8085 và chế tạo bằng các bảng mạch in. Lần lượt hàng chục đồng nghiệp từ Phòng Kỹ thuật số đã sang Pháp thực tập và tiếp tục theo hướng phát triển các giao diện kết nối VT81 với một số thiết bị ngoại vi như đĩa mềm, đĩa cứng,... Phòng Kỹ thuật số đã thiết kế chế tạo máy vi tính VT83 dựa trên chip vi xử lý Zilog Z80 (đầu thập niên 1980), đồng thời nghiên cứu thực hiện những giao diện quan trọng khác như kết nối bàn phím, màn hình, máy in phục vụ nhập xuất thông tin chữ Việt. Một vài thành công khác như những ứng dụng ban đầu tại Nhà máy Xi măng Hoàng Thạch và Ban Cơ yếu Trung ương (1981); ứng dụng hệ thống tin học văn phòng vào phục vụ Hội đồng Bộ trưởng (1984). Để dọn đường cho bước ngoặt đó, chúng tôi soạn các giáo trình, đi khắp nơi làm công tác đào tạo nhân lực bằng các lớp học ngắn hạn và thường xuyên tổ chức xêmina phổ biến những kiến thức mới.

Đến năm 1982, cả nước có 33 dàn máy tính đa năng cồng kềnh dùng trong tin học truyền thống và 49 máy chuyên dụng của Tổng cục Thống kê. Tuy nhiên, sự xuất hiện của các máy vi tính vừa gọn nhẹ vừa rẻ hơn nhiều lần với các ứng dụng linh hoạt bội phần so với tin học cổ điển đã báo hiệu sự lên ngôi của vi tin học.

Việc vượt qua rất nhiều khó khăn, thiếu thốn để đạt được thành tựu trong lĩnh vực công nghệ là điều mà thế hệ chúng tôi cảm thấy vô cùng tự hào. Tôi cho rằng, cảm giác chấp nhận những rủi ro và là người tiên phong tiếp thu kiến thức mới, tạo ra cũng như chứng kiến những ứng dụng công nghệ trực tiếp gây ảnh hưởng tích cực đến đất nước là niềm hạnh phúc khó có thể diễn tả bằng lời.

Những thành quả đạt được thời mở cửa

Giai đoạn 10 năm (1975 - 1985) cho thấy các kế hoạch công nghiệp hoá bằng sản xuất đồ điện tử dân dụng và lắp ráp máy vi tính đều đã sụp đổ vì chưa có nền kinh tế thị trường đòi hỏi cạnh tranh lành mạnh, hiệu quả với chất lượng cao ở Việt Nam. Các ngành nông nghiệp, công nghiệp nhẹ và công nghiệp khác như khai khoáng, luyện kim, cơ khí càng khủng hoảng hơn, dẫn đến cuộc Đổi mới đường lối kinh tế của nước ta được Chủ tịch Trường Chinh khởi xướng năm 1986.

Từ sau đổi mới, Việt Nam đã có những biến chuyển sâu rộng trong mọi mặt đời sống kinh tế, xã hội, và bộ mặt ngành CNTT cũng có nhiều thay đổi.

Năm 1988, Hội Tin học TP. Hồ Chí Minh và Hội Tin học Việt Nam được thành lập. Đồng thời xuất hiện một số doanh nghiệp phi quốc doanh, trong đó có Công ty Công nghệ Thực phẩm FPT do Viện nghiên cứu Công nghệ Quốc gia và Viện Cơ học liên kết đỡ đầu, với TS. Trương Gia Bình làm Tổng giám đốc và tôi phụ trách Trung tâm Dịch vụ Tin học ISC. Khi ISC bắt đầu nghiên cứu hệ điều hành Unix thì chứng kiến sự kiện chấn động toàn cầu: khối Đông Âu XHCN và Liên Xô tan rã hoàn toàn không tiếng súng (cuối những năm 80, đầu những năm 90). Tạp chí Tin học & Đời sống ra đời năm 1990, và FPT đã có hợp đồng tin học đầu tiên: Xây dựng mạng cục bộ và phần mềm giữ chỗ đặt vé máy bay cho Hàng không Việt Nam.

Năm 1991, tôi chính thức về Viện nghiên cứu Công nghệ Quốc gia làm giám đốc Trung tâm Hệ thống Thông tin ISC, nơi sau này đã thành công trong nghiên cứu, đào tạo, triển khai mạng cao tốc, chế bản, thiết kế đồ hoạ và vẽ font chữ Việt. Năm 1992, Viện Tin học đã truyền email bằng quay số điện thoại qua cổng ở Australia được nối với Internet.

Tôi tham gia JTC1/VN (Ban kỹ thuật tiêu chuẩn CNTT VN) năm 1993, xây dựng được 2 tiêu chuẩn về bộ mã Quốc ngữ và chữ Nôm. Năm đó, Chính phủ lần đầu tiên ban hành Nghị quyết 49/CP về phát triển CNTT trong thập niên 1990.

Năm 1994 đánh dấu nhiều sự kiện: Mỹ bỏ cấm vận Việt Nam, ra đời Tạp chí PC World Viet Nam, thành lập Ban chỉ đạo Chương trình Quốc gia về CNTT và ISC xây dựng mạng truyền báo diện rộng cho VDC (Công ty Điện toán và Truyền số liệu). Năm 1995 tiếp tục xây dựng 2 tiêu chuẩn: bàn phím Quốc ngữ và kho chữ Nôm; thành lập IFI - trường Tin học khối Pháp ngữ tại Hà Nội và Thủ tướng ban hành Quyết định 211/TTg phê duyệt Kế hoạch tổng thể về ứng dụng và phát triển CNTT đến năm 2000.

Năm 1996, "Trí tuệ Việt Nam" - mạng xã hội chạy qua điện thoại của FPT - ra đời ngay khi công ty VDC mở dịch vụ VN Mail. Năm 1997, ISC xây dựng thành công mạng diện rộng nối Văn phòng Chính phủ với các Bộ, ngành và UBND các Tỉnh/thành phố theo Quyết định 280/TTg, Việt Nam chính thức mở cổng Internet và cho tạp chí trực tuyến Quê Hương hoạt động trên đó. Báo điện tử Nhân Dân và VietnamNet ra đời, công ty FPT mở dịch vụ Internet (1998), báo điện tử Lao Động ra mắt, Chính phủ cho giải thể Ban chỉ đạo Chương trình QG về CNTT (1999). Chủ trương "Đổi mới" đi vào thực tiễn ở nước ta với việc nhà nước cho phép phát triển nền kinh tế thị trường định hướng XHCN. Nhiều người trẻ trong số những nhà nghiên cứu và giảng dạy đã tập hợp nhau lại để thành lập các công ty ứng dụng công nghệ cao.

Sự sáng tạo của một người làm CNTT có thể đến từ những quan sát vô cùng nhỏ, từ những "nguyên liệu" vô cùng đơn giản.

Năm 2000 kết thúc vụ hoang tưởng tin học toàn cầu "Sự cố Y2K"; toàn bộ quyền quản lý tên miền .VN được chuyển từ Australia về Việt Nam và Chính phủ đã ban hành Nghị quyết 07/2000/NQ-CP về phát triển công nghiệp phần mềm giai đoạn 2000 - 2005.

Từ năm 2000 đến nay, chắc hẳn các bạn đã biết sự phát triển CNTT nhanh như thế nào rồi. 20 năm đó có lẽ phải mất nhiều trang giấy mới có thể mô tả hết, dù chỉ vắn tắt.

Hầm máy tính Odra 1304 tại Đồi Thông tháng 12.1976

Lớp học kĩ thuật vi xử lý tại Đồi Thông Liễu Giai tháng 12.1976

Các anh chị em phòng KT tính toán tại Đồi Thông 1977

Nguyễn Chí Công và Phan Minh Tân làm việc trên máy Vi tính VT81 tại Đồi Thông hè 1979

Nguyễn Chí Công và chiếc máy vi tính VT81 tại Đồi Thông 1979

Nguyễn Chí Công, GS Vũ Đình Cự - Viện trưởng nghiên cứu công nghệ quốc gia, Thủ Tướng Phạm Văn Đồng và TS Trần Đình Anh

GS Phan Đình Diệu và Nguyễn Chí Công bàn bạc với Ủy ban Pháp về hợp tác khoa học và kỹ thuật với Việt Nam

Với rất nhiều sự kiện như tôi vừa nêu trên, hẳn bạn đọc cũng hình dung được một chặng đường dài với nhiều biến động của ngành CNTT nước nhà. Câu hỏi đặt ra là chúng ta học được gì từ sau những vấp ngã, trưởng thành đó?

Đầu tiên, rủi ro luôn đi kèm với những cơ hội. Nếu bạn sợ hãi tiếp cận với những thứ mới thì sẽ không bao giờ tạo ra những thành tựu mới cả. Có những thiết bị công nghệ ở thời chúng tôi là xa xỉ (máy tính cá nhân, điện thoại thông minh,...) nhưng bây giờ chúng đã trở nên cực kỳ phổ biến, nhà nhà đều có. Thứ hai, bạn không nhất thiết phải giàu có hoặc được đầu tư nhiều tiền thì mới có thể sáng tạo trong ngành này. Sự sáng tạo của một người làm CNTT có thể đến từ những quan sát vô cùng nhỏ, từ những "nguyên liệu" vô cùng đơn giản.

Theo tôi, Chính phủ đóng một vai trò quan trọng trong việc tạo ra cơ chế và đòn bẩy cho ngành CNTT phát triển sáng tạo. Thật mừng vì thời điểm hiện tại CNTT đang là lĩnh vực nhận được sự quan tâm lớn, tuy nhiên, tôi cho rằng những hoạt động xúc tiến thương mại trong lĩnh vực này nên được chú trọng thêm, và Nhà nước cần tạo một sân chơi lành mạnh, công bằng, tin tưởng hơn để các doanh nghiệp CNTT vừa và nhỏ có cơ hội sáng tạo và đổi mới hết khả năng. Hiện tại vẫn là mạnh ai nấy làm, sự hợp tác và hỗ trợ lẫn nhau ở cấp độ doanh nghiệp, thậm chí là Hiệp hội, còn nhiều hạn chế.

Về phía các bạn trẻ, tôi không hi vọng các bạn ghi nhớ hết những sự kiện lịch sử của ngành CNTT nước nhà, chỉ mong các bạn sẽ trân trọng những nỗ lực của các thế hệ ông cha đã đặt nền móng cho những viên gạch đầu tiên của ngành. CNTT vẫn là một lĩnh vực non trẻ và nhiều triển vọng. Đừng vội thoả mãn với những gì mình đang có, bởi khoảng cách về trình độ giữa Việt Nam và thế giới vẫn còn rất lớn.

Liệu Việt Nam có thể vươn lên trở thành cường quốc CNTT?

Tôi nghĩ đáp án cho câu hỏi này hoàn toàn phụ thuộc vào chính các bạn và con cháu. Số liệu thống kê cho phép tin tưởng rằng thế hệ nay mai có thể hoàn thành ước nguyện đó:

1. Lực lượng lao động vẫn chiếm tỷ lệ cao so với dân số (quý I năm 2019 ước tính là 55 triệu người, giảm 207 nghìn người so với quý trước và tăng 332 nghìn người so với quý I năm 2018). Năng suất lao động kể từ 2008 đã tăng 22,5%, trung bình hàng năm tăng 4,7%. Lực lượng lao động này có lợi thế trong việc tiếp cận, học hỏi, nắm bắt những công nghệ mới. Mặt khác, bản thân họ cũng trở thành những người tiêu dùng thường xuyên trên nền tảng số.

2. Tốc độ phát triển GDP tốt (đặc biệt tốt ở lĩnh vực CNTT nói riêng), từ năm 2011 đến 2017, GDP đã tăng hơn gấp đôi từ 105 tỷ lên 220 tỷ USD. Năm 2019, báo cáo "Nền kinh tế số Đông Nam Á 2019" cho biết nền kinh tế số Việt nam trị giá 12 tỷ USD (chiếm 5% GDP cả nước) và dự báo sẽ chạm tới 43 tỷ USD năm 2025, cụ thể trong các lĩnh vực: du lịch trực tuyến, thương mại điện tử, truyền thông trực tuyến và đặt xe công nghệ. Việt Nam cùng Indonesia đang có tốc độ tăng trưởng kinh tế số đứng đầu Đông Nam Á (38%/năm so với 33%/năm của cả khu vực). Hơn nữa, các công ty công

nghệ tại Việt Nam đang bắt kịp những công nghệ mới trên thế giới ở hầu hết các lĩnh vực: Dữ liệu lớn (Big Data), Trí tuệ nhân tạo/Máy học (AI/ML), Blockchain,...

3. Theo "Báo cáo Tăng trưởng thông minh hơn: Học tập và Phát triển công bằng ở Đông Á - Thái Bình Dương" do Ngân hàng Thế giới (WB) công bố tháng 3/2018: "7 trong số 10 hệ thống giáo dục hàng đầu của thế giới nằm ở khu vực Đông Á - Thái Bình Dương, trong đó sự phát triển thực sự ấn tượng là ở Trung Quốc và Việt Nam".

Đó là những cơ sở để chúng ta tin tưởng vào tiềm năng của một thế hệ trẻ được đào tạo chỉn chu và vững chắc về kiến thức, cùng với một thị trường đông đảo những người dùng biết sử dụng công nghệ.

Tuy nhiên, cũng cần phải học hỏi kinh nghiệm quốc tế và tìm hiểu xem Việt Nam có những khó khăn gì để vượt qua bằng cách điều chỉnh các chính sách đối phó cho phù hợp với điều kiện riêng ở từng thời kỳ.

Trong Chỉ thị số 01/CT-TTg về thúc đẩy phát triển doanh nghiệp công nghệ số Việt Nam, chính phủ đã đề quyết tâm cao cho việc hiện thực hóa quá trình chuyển đổi số thông qua các giải pháp, tổ chức thực hiện cụ thể trong cả doanh nghiệp và các Bộ, ban, ngành. Thủ tướng Nguyễn Xuân Phúc đã nhấn mạnh: "Khát vọng về một Việt Nam hùng cường, thị trường gần 100 triệu người và các bài toán đặc thù của Việt Nam trong các lĩnh vực nông nghiệp, giao thông, y tế, giáo dục, tài chính, tài nguyên, môi trường,... chính là tiền đề thuận lợi cho các doanh nghiệp công nghệ số Việt Nam lớn mạnh và vươn ra thế giới".

Trước những điều kiện cả về khách quan lẫn chủ quan đều khá thuận lợi như vậy, trong tương lai, các bạn trẻ sẽ có những cơ hội như thế nào về mặt sự nghiệp lẫn cơ hội nghề nghiệp trong ngành CNTT?

Có lẽ câu trả lời đã khá rõ: các nước G7 đang già đi và phụ thuộc vào nhập khẩu lao động. Riêng các bạn trong ngạch CNTT càng có nhiều cơ hội ở cả trong và ngoài nước vì hiện nay những điều kiện để học tập và làm việc trên Internet cũng như cung ứng cho thị trường toàn cầu hoá đều khá dễ dàng. Nếu các bạn can đảm, tự tin vì biết ngoại ngữ và đọc nhiều để có đủ thông tin thì mới đánh giá được cơ hội và chủ động đi đến đích thắng lợi. Thực tế cho thấy cơ hội luôn luôn xuất hiện nhưng không kéo dài và dễ vuột mất nếu ta không chuẩn bị tốt. Khi có cơ hội lớn, ta còn phải biết làm việc nhóm, chia sẻ cộng đồng và do đó cần có kỹ năng mềm.

Kết

Trải qua vô số những thăng trầm và biến động của thời cuộc, tôi mong bạn đọc hãy nhìn nhận lịch sử không chỉ ở góc độ kể chuyện quá khứ, mà thông qua đó để hiểu và trân trọng hiện tại, rút ra những bài học hướng tới xây dựng tương lai. Các bạn có tuổi trẻ, có sức bật, hãy nuôi những giấc mơ và khát vọng lớn, phấn đấu trở thành thế hệ kiến tạo những thay đổi to lớn cho đất nước. Ngành CNTT còn cả một bầu trời rộng lớn phía trước đang chờ đón các bạn.

Chúc các bạn luyện tập mạnh khoẻ cả thể chất và tinh thần, sẵn sàng tâm - trí - lực để đón nhận những dịp may.

Ngành IT Việt Nam
Hiện tại & Tương lai

Biên soạn bài viết: **Ban biên tập**

Kế thừa những dấu mốc mang tính bước ngoặt trong lịch sử, hiện tại ngành CNTT ở Việt Nam đang diễn biến ra sao? Tiềm năng phát triển của ngành trong tương lai như thế nào? Và những tấm gương thành công trong giới công nghệ có câu chuyện, kinh nghiệm gì muốn gửi gắm đến các bạn trẻ? Để có thể lựa chọn con đường nghề đúng đắn, đừng quên trang bị cho bản thân những kiến thức tổng quan nhất về diễn biến ngành CNTT thông qua bài viết này nhé.

Thị trường CNTT Việt Nam và những điểm nhấn

Xuất phát là một nền kinh tế nông nghiệp lạc hậu, đến nay, chúng ta có thể nhận ra những bước chuyển mình rõ rệt của Việt Nam trong hành trình thu hẹp khoảng cách về công nghệ với thế giới. Những cụm từ như "Cách mạng công nghiệp 4.0", "Internet of Things" (Internet vạn vật), "Big Data" (Dữ liệu lớn), "Trí tuệ nhân tạo", "Điện toán đám mây",... không chỉ là những thuật ngữ được truyền thông đề cập mà thực sự đang dần len lỏi vào mọi hoạt động của từng cá nhân, doanh nghiệp.

Theo Bảng xếp hạng Chỉ số đổi mới toàn cầu 2019 (GII), Việt Nam đứng thứ 42/129 các nền kinh tế toàn cầu. Diễn đàn kinh tế thế giới 2019 xếp chỉ số ICT Adoption (chỉ số ứng dụng Viễn thông và Công nghệ thông tin) của Việt Nam tăng từ hạng 95 lên hạng 41. Ngoài ra, Liên minh Viễn thông Quốc tế (ITU) đánh giá Việt Nam có những bước tiến đáng ghi nhận về xếp hạng an toàn, an ninh mạng toàn cầu, tăng 50 bậc so với năm 2017.

Số liệu thống kê trong ngành Thông tin & Truyền thông (TT&TT) cũng cho thấy sự tăng trưởng toàn diện trên các lĩnh vực: Tổng doanh thu toàn Ngành tăng 8,8% so với năm 2018, đạt 3.100.000 tỷ đồng (gần 135 tỷ USD); Doanh thu công nghiệp phần mềm là 5 tỷ USD, tăng 500 triệu USD.

Bộ trưởng Bộ TT&TT Nguyễn Mạnh Hùng xác định 2020 là năm chuyển đổi số quốc gia, khởi động tiến tới một Việt Nam số. Bộ TT&TT sẽ thực hiện những nhiệm vụ, giải pháp để cải thiện môi trường kinh doanh, nâng cao năng lực cạnh tranh quốc gia, với một số mục tiêu: Nâng xếp hạng nhóm chỉ số Ứng dụng công nghệ thông tin lên từ 2 - 3 bậc; Nâng xếp hạng nhóm chỉ số công nghệ thông tin và sáng tạo trong mô hình kinh doanh lên ít nhất 3 - 5 bậc; Nâng xếp hạng các chỉ số thuộc nhóm Sáng tạo trực tuyến lên 2 - 5 bậc.

Việt Nam cũng được dự báo chuẩn bị đón nhận làn sóng đầu tư nước ngoài mới. Từ năm 2019, để tránh bị ảnh hưởng từ cuộc chiến thương mại Mỹ - Trung, các tập đoàn công nghệ đã lên kế hoạch rời Trung Quốc và tiến trình này đang được đẩy nhanh do dịch Covid-19. Sở hữu một nền chính trị ổn định, các kết quả tăng trưởng kinh tế khả quan, cộng thêm với việc nổi lên là hình mẫu phòng chống dịch Covid-19 thành công trên toàn cầu, Việt Nam trở thành điểm đầu tư thế giới quan tâm.

Theo tin của Nikkei Asian Review, tập đoàn Apple chọn Việt Nam làm điểm đến và sẽ sản xuất khoảng 3 - 4 triệu đơn vị tai nghe AirPods (trước đây, phần lớn các tai nghe AirPods được sản xuất tại Trung Quốc). Việt Nam cũng đang đón nhận việc dịch chuyển các dây chuyền sản xuất laptop, điện thoại của các "đại gia công nghệ" như Google, Microsoft và một phần hoạt động sản xuất máy chơi game Switch Lite của hãng Nintendo. Samsung cho biết thời gian tới cần 4.000 nhân lực cho lĩnh vực công nghệ, hay tập đoàn Hindustan Computers Limited, Ấn Độ - 1 trong 3 công ty công nghệ lớn nhất Ấn Độ, top 5 công ty outsource trên thế giới - đã đầu tư trung tâm công nghệ thông tin ở Tp. HCM, có thể tuyển dụng 10.000 kỹ sư trong 5 năm tới. Foxconn - nhà cung ứng linh kiện lớn cho Apple - cũng đã làm nhà máy tại Bắc Giang,...

Ngoài ra, trước làn sóng khởi nghiệp và khao khát sáng tạo các sản phẩm công nghệ "Make in Vietnam", nhiều doanh nghiệp Việt đã định hướng làm chủ công nghệ, sản xuất sản phẩm ngang tầm thế giới. Có thể kể đến điện thoại Bphone của Tập đoàn Công nghệ Bkav (do người Việt nghiên cứu, chế tạo, thiết kế bản mạch, viết phần mềm), Tập đoàn Vingroup với các thương hiệu nhận được nhiều sự quan tâm như ô tô VinFast hay điện thoại Vsmart (đầu tư và sản xuất theo mô hình của các hãng công nghệ lớn trên thế giới),...

Theo báo cáo thị trường IT năm 2020, Việt Nam có 12 dịch vụ IT nổi bật:

Thương mại điện tử (E-Commerce)
Sendo, Lazada, Shopee, Tiki, thegioididong

Công nghệ tài chính (Fintech)
Momo, Moca, ViettelPay, AirPay, MoneyLover, OnePAY, VNPAY

Gọi xe/thức ăn
Grab (GrabFood), GO-VIET (GoFood), Be, Now, BAEMIN

Du lịch trực tuyến
Vntrip.vn, Ivivu.com, Chudu24.com, Mytour.vn, Tripi.vn, Gotadi.com

Business Process Outsourcing (BPO)
FPT, Tek Experts, FSI, Expertrans, Mắt Bão

Hightech (AI/ML, IoT, Blockchain,...)
Infinity Blockchain Labs, Kyber Network, Umbala Network, Lina, Cinnamon, Asilla, Arimo

Business Service (SaaS,...)
Base, Haravan, Sapo, Misa, KiotViet, TopCV

Edtech
GotIt!, Elsa, Monkey Junior, Unica, Kyna, Hocmai

Vận chuyển hàng hóa
Giaohangnhanh, Giaohangtietkiem, Grab, AhaMove, Go-Viet, Ecotruck

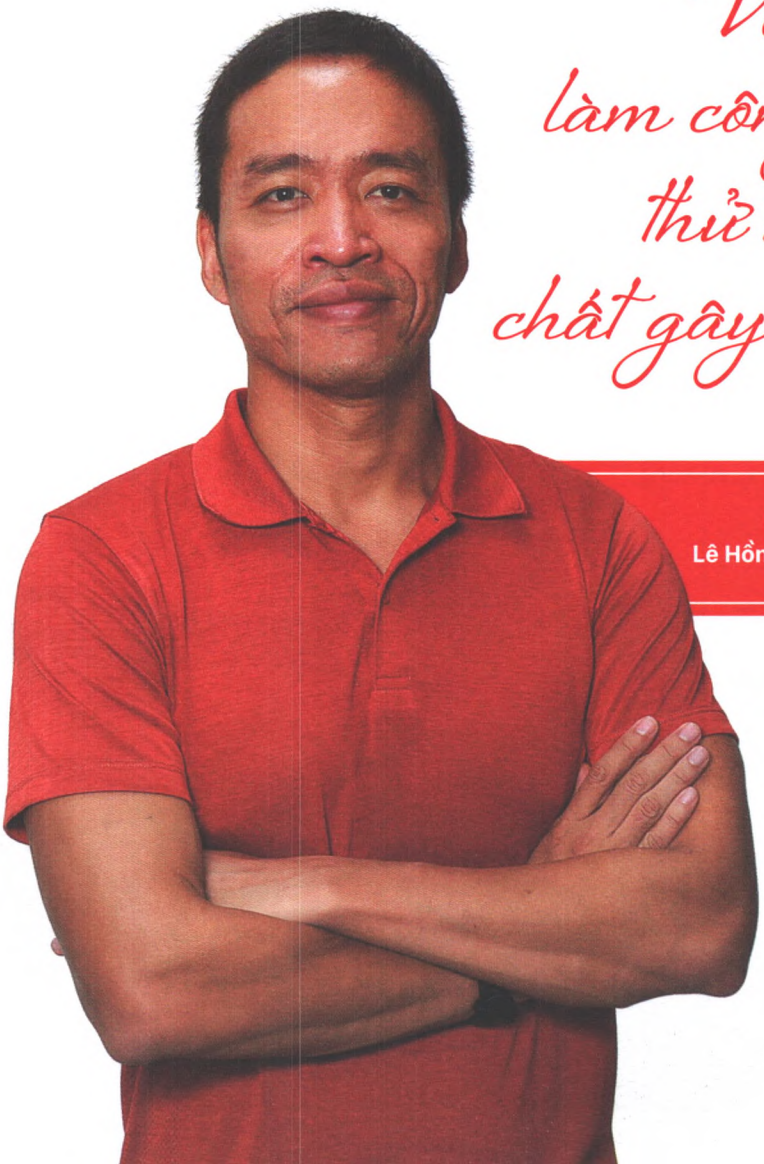
Xuất bản phần mềm
Fsoft, Tinh Vân, KMS, TPS, Sun*

Truyền thông trực tuyến, Nội dung số
VCCorp, VTC, VNG, FPT Online, Nexttech

Healthcare
eDoctor, Vinmec

Các sự kiện, tổ chức và hoạt động nổi bật về công nghệ để các bạn tham khảo: Tech Summit, Mobile Day, Vietnam Web Summit, Techfest Vietnam, DEVDAY.org, Tech Expo, VietAI, Grokking TechTalks, GDG Vietnam, MEETUP.vn

Các Tech Founder nói gì với thế hệ trẻ yêu công nghệ?
Không chỉ dừng ở những số liệu thống kê hay các tin tức vĩ mô, Spiderum đã có buổi trò chuyện với các cá nhân thành công trong lĩnh vực công nghệ và lắng nghe những chia sẻ chân tình của họ dành cho lớp trẻ. Đứng trước thế giới công nghệ đầy biến động, các bạn trẻ muốn lựa chọn ngành CNTT cần lưu ý những gì?



"Với người làm công nghệ, thử thách là chất gây nghiện"

Khách mời phỏng vấn:
Lê Hồng Minh - Chủ tịch VNG

Anh có nhận định gì về thị trường công nghệ tại Việt Nam trong vài năm tới?

Thế giới đang có sự chuyển dịch từ không gian trực tuyến sang không gian thực và Việt Nam không phải ngoại lệ. Trừ Amazon, các công ty như Google, Yahoo!,... đều có kênh online, và kinh doanh trên nền tảng quảng cáo trước. Amazon sẽ không thể thành công ty trị giá hàng tỷ đô la Mỹ nếu chỉ kinh doanh trực tuyến. 5 năm trở lại đây, nhiều công ty trên thị trường như Grab, Airbnb đều chuyển từ trực tuyến sang thực tế (online sang offline).

Trong vòng 3 - 5 năm tới, xu hướng chuyển từ trực tuyến sang thực tế sẽ là dòng chảy chủ đạo ở Việt Nam. Công nghệ sẽ chạm đến mọi khía cạnh của cuộc sống, chúng ta còn nhiều cơ hội ở các dịch vụ như tài chính, chăm sóc sức khỏe,... Hiện tại, bên cạnh các ứng dụng giải trí và nền tảng kết nối, VNG đang đầu tư vào thị trường thanh toán điện tử và các dịch vụ điện toán đám mây, triển khai nhiều dự án công nghệ thú vị và có giá trị cho người dùng.

Gần đây, tại những buổi nói chuyện với sinh viên hay một số sự kiện về công nghệ, anh chia sẻ nhiều về câu chuyện phát triển con người thay vì các sản phẩm, xu hướng công nghệ hay khởi nghiệp như trước đây. Vì sao vậy?

Thật ra VNG luôn quan tâm yếu tố con người từ xưa đến nay. Khi mình trưởng thành, đây càng là mục tiêu phải ưu tiên hàng đầu.

Tôi cảm nhận những điều sau có thể trở thành giới hạn của VNG sau 15 năm liên tục tăng trưởng: Nâng tầm một số mảng kinh doanh đạt đến tầm vóc thị trường lớn hơn hiện tại rất nhiều; Tiến vào những mảng kinh doanh mới, sản phẩm mới mà trước giờ chưa làm. Mình muốn đánh trận này mà không

rõ liệu có đủ nguồn lực đáp ứng được những mục tiêu đó hay không thì trước sau cũng thất bại. Nguồn lực không đơn giản là trả thật nhiều tiền thuê người về, mà cần cả quá trình kiên trì, liên tục đầu tư vào những nhân viên đang làm việc và cả những người mới tham gia công ty, giúp họ phát triển chứ không phải cứ tuyển vào không làm được thì nghỉ. Cá nhân tôi cảm nhận đây là vấn đề quan trọng nhất, thậm chí còn vượt trên chuyện xây dựng sản phẩm.

Một bạn trẻ lựa chọn xây dựng sự nghiệp trong lĩnh vực công nghệ cần phải chuẩn bị những gì để thành công?

Trước tiên, các bạn định nghĩa thành công là gì?

Khi học cấp 3 hoặc Đại học, nhiều bạn chưa từng suy nghĩ đến khái niệm của thành công. Sẽ rất nguy hiểm nếu bạn chỉ nghe tiêu chuẩn về thành công từ những người khác mà không suy nghĩ đủ sâu về nó. Thành công với rất nhiều người là sở hữu mức lương cao hoặc kiếm được nhiều tiền, điều này không sai, nhưng liệu nó có đúng với bản thân bạn không? Tôi có nhiều người bạn đạt cực kỳ nhiều thành tựu trong kinh doanh, dư dả về tài sản và có danh tiếng, nhưng họ không có được cảm giác thành công hay hạnh phúc, và vẫn tiếp tục chạy theo cái vòng xoáy đó. Đến một giai đoạn họ tự hỏi: "Ủa, tôi muốn làm gì đây?". Cuối cùng, năm 40 - 50 tuổi, họ định nghĩa thành công với họ là mở 1 tiệm cà phê, mỗi ngày pha những cốc cà phê ngon cho 10 người khách là cảm thấy thỏa mãn rồi. Nên trước hết, các bạn cần định nghĩa thật rõ thành công đối với bản thân mình là gì? Câu trả lời có thể không có được ngay bởi nó là một quá trình tích lũy: nghe người khác nói chuyện, đọc sách, trải nghiệm thực tế cuộc sống,... Các bạn đừng đơn giản hóa thành công bằng câu chuyện kiếm được bao nhiêu tiền. Như vậy, tôi nghĩ các bạn sẽ đi rất xa và hài lòng với quyết định của mình.

Vậy xin hỏi anh một câu cụ thể hơn: Học công nghệ có thể kiếm nhiều tiền không?

Tôi từng nhận được thắc mắc: Làm sao để có được 1 triệu đô la từ công nghệ?

Đối với ngành CNTT, để có được 1 triệu đô la, các bạn phải có kỹ năng xuất chúng để trở thành kỹ sư giỏi tạo ra những sản phẩm chủ đạo tại các công ty công nghệ. Qua quá trình làm việc, bạn sẽ được chia sẻ lợi ích như cổ phần. Khi giá trị công ty đi lên, lượng cổ phần của bạn có thể trị giá 1 triệu đô. Đây là con đường mà rất nhiều kỹ sư tại VNG đã làm được. Điều tôi muốn chia sẻ với các bạn trẻ là đừng bị suy nghĩ chỉ có làm kinh doanh hay làm trong các lĩnh vực như tài chính, bất động sản,... mới có thể kiếm 1 triệu đô. Dĩ nhiên, nhiều người đã làm như vậy, và nếu thật sự bạn muốn công việc đó thì hãy làm, lúc đó có lẽ CNTT không phải lựa chọn phù hợp với bạn. Còn nếu yêu thích công nghệ, hãy trở thành người xuất sắc trong lĩnh vực đấy, đó là cơ hội để bạn chạm tay vào con số 1 triệu đô và tiếp tục tạo được nhiều giá trị khác trong tương lai. Tôi hy vọng các bạn có niềm tin là khi trở thành kỹ sư CNTT xuất chúng, bạn hoàn toàn có thể đạt được con số trên.

Làm sao để trở thành người xuất chúng trong lĩnh vực công nghệ? Đặc biệt là lĩnh vực này chúng ta còn phải cạnh tranh với các kỹ sư từ khắp nơi trên thế giới?

"Người xuất chúng" là khái niệm rất rộng và chung chung. Từ trải nghiệm cá nhân, tôi cho rằng một sinh viên công nghệ cần có 3 yếu tố:

1. Kỹ năng và kiến thức nền tảng tốt: Đôi khi, bạn không cần học quá nhiều, quá rộng, nhưng phải nắm thật chắc về những thứ cơ bản nhất như toán học, thuật toán, lập trình. Vì rất nhiều lời giải phức tạp đến từ chuyện mình hiểu những điều cơ bản một cách sâu sắc. Tôi ví dụ, các bạn sinh viên làm bài tập có thể lên Internet, tìm kiếm rồi copy, chỉnh sửa lại một chút là cũng làm được bài. Bởi code bây giờ có sẵn ở rất nhiều các thư viện, bạn tải về cũng không cần hiểu quá sâu nó hoạt động như nào, miễn sao chạy ra kết quả mong muốn là ổn. Tuy nhiên, dù có thể giải được các bài toán trước mắt, nhưng về sau, bạn sẽ không đủ năng lực để đi lâu dài vì không hiểu bản chất vấn đề. Việc nắm tốt các kiến thức cơ bản nền tảng giúp bạn tổng quát hóa các lời giải của mình.

2. Hiểu biết bên ngoài lĩnh vực chuyên môn: Các bạn đừng giới hạn mình ở chuyện chỉ lập trình tốt mà phải hiểu tương đối rộng hơn về thế giới công nghệ nói chung, cũng như có kỹ năng làm việc, giao tiếp, quản lý công việc, sắp xếp thời gian,...

3. Năng lực làm việc đội nhóm: Dù xuất chúng đến đâu bạn cũng không thể làm một mình. Tôi hay nói với mọi người trong VNG những chuyện mình làm không quá cao siêu, phức tạp, thậm chí đơn giản thôi nhưng phải thực sự tập trung làm nó đến chất lượng đẳng cấp thế giới (world class). VNG có thể có 1 - 2 kỹ sư tầm world class, nhưng để phát triển được một sản phẩm, một mảng kinh doanh mới cần tới vài trăm kỹ sư. Như vậy, chúng ta không chỉ cần năng lực cá nhân mà phải có cả năng lực làm việc cùng nhau. Đó lại là thách thức hoàn toàn khác.

Điều gì khiến anh cảm thấy tự hào khi là người hoạt động trong lĩnh vực công nghệ?

Tôi luôn quan niệm hãy sống một cuộc sống thật trọn vẹn và tạo ra ảnh hưởng đến đời sống của càng nhiều người càng tốt. Ở vị trí CEO hiện tại, tất nhiên có nhiều thứ khiến tôi đau đầu. Nhưng cảm giác hạnh phúc khi đạt được một thành quả nào đó đủ mạnh để khỏa lấp tất cả những điều khó chịu. Thành quả ấy, cũng như ở bất kỳ công ty công nghệ nào khác, chính là tầm ảnh hưởng với xã hội thông qua những sản phẩm mà mình tung ra thị trường, những sản phẩm của kỹ sư Việt được hàng triệu người hay tuyệt vời hơn là hàng tỷ người đón nhận, sử dụng. Tôi thấy tự hào vì VNG đã góp phần tạo nên cuộc sống tốt đẹp cho nhiều người.

Yếu tố thứ hai, thông qua việc bước tới, tôi luôn có cơ hội được làm những điều mới và từ đó, học những thứ mới. Thế giới công nghệ vẫn tiếp tục thay đổi nhanh chóng và khó đoán định. Bài học quan trọng trong 3 - 5 năm qua là dù đang thành công và bận rộn với các sản phẩm hiện tại, đội ngũ VNG vẫn rất quyết liệt và không ngần

ngại thử nghiệm những sản phẩm và hướng đi mới. Xây dựng một tương lai "vô định" là một trải nghiệm vừa rất đáng sợ, vừa rất hứng khởi. Với nhiều người, thử thách có thể tạo ra sự mệt mỏi. Nhưng với tôi và những người làm công việc liên quan đến công nghệ, thử thách là chất gây nghiện.

Thứ ba, trong những cái mới, tôi thấy mình có thể thay đổi, cải tiến môi trường xung quanh, và đập bỏ đi những thứ không còn phù hợp.

Thứ tư là cảm giác thoải mái khi được làm việc với những con người mà mình yêu mến, tin cậy. Tôi luôn tìm thấy niềm vui khi được giúp các bạn trẻ phát triển. Vì chính tôi cũng từng là một người trẻ, từng vui vẻ, hồ hởi khi được học những điều mới hay khi chinh phục những cột mốc đầu tiên. Khi một tập thể gồm nhiều người cùng nhìn về một hướng, thứ cảm xúc và sức mạnh cộng hưởng ấy thật sự rất tuyệt vời.

Anh có điều gì muốn gửi gắm tới các bạn trẻ đang muốn đặt chân vào thế giới công nghệ?

Có lẽ ai chơi game cũng biết một câu kinh điển: "Everything is a game. Don't take it serious". Tôi thuộc loại chơi game cho vui, và cả trong kinh doanh cũng có tư tưởng thua thì chơi lại. Đôi khi tôi cũng tự hỏi: "Cái tính ấy có tốt hay không? Cạnh tranh thị trường ngày một khốc liệt, mình có nên chuyển sang máu ăn thua một chút chăng?". Nhưng bù lại, chính việc xem mọi thứ như một trò chơi (người ta vẫn nói Life is a big game mà), tôi không xem ai như kẻ thù cả. Có thể trên thương trường đấu nhau sát ván, nhưng ra ngoài, bỏ công việc sang một bên vẫn có thể bắt tay, uống cùng nhau ly bia, ăn bữa cơm, chứ đâu có gì phải căng thẳng. Vì sau tất cả, chúng ta luôn cần đối thủ. Không có đối thủ, làm sao chúng ta duy trì được sự tập trung và tinh thần tiến lên?

Với tinh thần như vậy, tôi mong các bạn trẻ đừng lo lắng sẽ thất bại. Bạn chưa làm mà sợ thất bại thì lấy gì để thành công? Ngành công nghệ có thể không hào nhoáng lung linh như nhiều bạn nghĩ, mà có vô vàn những thời điểm khó khăn, những thách thức ghê gớm khiến bạn muốn từ bỏ. Tuy nhiên, nếu thấy khó khăn mà bỏ cuộc, bạn sẽ không bao giờ nhìn thấy được thành công phía trước.

Khi bạn làm điều gì đó vì đam mê, mà không quá ám ảnh với áp lực phải thành công ngay lập tức, phải kiếm được nhiều tiền, rất có thể một điều gì đó khác thường, đặc biệt sẽ đến với bạn.

Blockchain là hiện thực, không phải "cơn ảo mộng" của công nghệ

Khách mời phỏng vấn:
Lợi Lưu - CEO & Co-founder Kyber Network

Được biết Kyber Network là một trong những doanh nghiệp đi đầu tại Việt Nam trong việc ứng dụng công nghệ Blockchain. Anh có thể giải thích ngắn gọn cho 1 người không có chuyên môn hiểu công nghệ Blockchain là gì?

Hiểu theo nghĩa đơn giản nhất, Blockchain là công nghệ lưu trữ và truyền tải dữ liệu một cách an toàn và bảo mật dựa trên mật mã học. Một hệ thống Blockchain bao gồm rất nhiều nút độc lập có khả năng xác thực dữ liệu dựa trên sự đồng thuận, thay vì sử dụng một bên trung gian làm việc đó.

Trong quá trình hoạt động, Blockchain liên tục cập nhật cơ sở dữ liệu mới, lưu trữ lại thành từng khối (block) và được đẩy liên tục vào cơ sở dữ liệu có sẵn. Các khối được lưu trữ với nhau, khối trước nối với khối sau, dựa theo công nghệ mã hóa sao cho sự thay đổi của một khối trong quá khứ sẽ làm thay đổi các khối sau đó. Nó giống như một dây xích (chain), các mắt xích

là các block được nối lại với nhau. Nếu một mắt xích bị vỡ hoặc thay đổi, sợi dây xích sẽ không thể liền mạch. Chính vì vậy, dữ liệu trên Blockchain gần như được lưu trữ vĩnh viễn và rất khó bị thay đổi.

Ngành IT rất rộng và nhiều hướng phát triển, vì sao anh lại lựa chọn phát triển về Blockchain?

Khi còn ngồi trên ghế nhà trường, tôi đã quan tâm đến lập trình, đặc biệt về thuật toán, cấu trúc dữ liệu và quyết định theo học Khoa học Máy tính tại Đại học Quốc gia Hà Nội. Tôi bắt đầu chú ý đến Blockchain từ năm 2014, trước khi Crypto và Ethereum trở thành hiện tượng trên toàn thế giới. Tôi và bạn học, trong đó có những người đã cùng tôi sáng lập Kyber Network, đều choáng ngợp bởi tính cách mạng mà công nghệ này mang lại. Chúng tôi thấy Blockchain mở ra viễn cảnh hệ thống giao dịch và thanh toán toàn cầu có khả năng vận hành mà không cần phụ thuộc bất kỳ nhân tố tập trung nào trong tương lai. Điều này như đưa cả thế giới bước vào kỷ nguyên mới, do đó nó thôi thúc tôi tham gia vào lĩnh vực này.

Blockchain là một công nghệ mới đối với Việt Nam. Anh đánh giá thế nào về tiềm năng phát triển của Blockchain tại Việt Nam?

Công nghệ Blockchain xuất hiện trên thế giới khoảng 10 năm nay, tuy có nhiều ứng dụng thiết thực nhưng tôi cho rằng công nghệ này vẫn đang trong giai đoạn nghiên cứu và phát triển ban đầu. Do đó, đây cũng chính là cơ hội cho Việt Nam đuổi kịp các quốc gia khác trong việc tìm hiểu, khai thác và làm ra những ứng dụng thực tiễn, tạo đà nhảy vọt cho cuộc cách mạng công nghiệp 4.0 mà chúng ta đang triển khai.

Hiện nay tại Việt Nam, công nghệ Blockchain được áp dụng chủ yếu trong các lĩnh vực tài chính, chuỗi cung ứng, dịch vụ công cộng và phần nào nhen nhóm trong lĩnh vực bảo hiểm, bất động sản, y tế, tiêu dùng và quản lý tổ chức. Trong đó, hơn 80% là ứng dụng trong lĩnh vực tài chính, cụ thể như: thanh toán quốc tế, thanh toán bù trừ giao dịch liên ngân hàng, bảo lãnh, cho vay, thu thuế và kiểm toán,... Bên cạnh đó, Việt Nam là đất nước có dân số trẻ, tỷ lệ người dùng Internet cao và thương mại điện tử phát triển nhanh. Những yếu tố kể trên giúp Việt Nam được đánh giá là quốc gia có tiềm năng lớn trong việc phát triển và ứng dụng công nghệ Blockchain.

Tuy vậy, việc ứng dụng công nghệ này ở Việt Nam đang trong giai đoạn đầu, còn gặp nhiều khó khăn và thử thách. Đó là những vấn đề về cơ sở pháp lý, về giảm chi phí giao dịch, các tiêu chuẩn và quy chuẩn kỹ thuật, thiếu nguồn nhân lực chất lượng cao,... Nếu nắm bắt và theo kịp công nghệ này, đồng thời xây dựng được các nền tảng cơ bản để hệ sinh thái Blockchain phát triển mạnh mẽ, nó có thể tạo ra một cuộc cách mạng mới trong kinh tế, xã hội. Việt Nam sẽ có cơ hội rút ngắn khoảng cách với thế giới trong tương lai không xa.

Có ý kiến lo ngại rằng: "Blockchain là công nghệ quá mới, không biết sẽ tồn tại được hay không? Liệu chúng ta có ảo tưởng và đánh giá nó quá cao?". Anh nghĩ sao về quan điểm này?

Bất kỳ công nghệ nào khi mới ra đời cũng không tránh khỏi những nghi ngờ, hoài nghi. Nhìn lại lịch sử phát triển khoa học công nghệ của nhân loại, khi chiếc điện thoại đầu tiên được phát minh vào năm 1876, máy tính cá nhân (PC) lần đầu được giới thiệu vào năm 1949, và hệ thống thương mại điện tử đời đầu (teleshopping) được ra mắt mắt vào năm 1979, không ai nghĩ cần chúng trong cuộc sống. Tuy nhiên, đến nay ta đã thấy những công nghệ đó tác động và thay đổi toàn nhân loại ra sao: điện thoại và PC là thiết bị thiết yếu với mỗi cá nhân và thương mại điện tử là ngành có doanh thu hàng nghìn tỷ USD mỗi năm. Do đó, tôi tin một công nghệ mang tính đột phá, có tính ứng dụng cao như Blockchain cũng không nằm ngoài xu thế phát triển đó.

Hiện tại, nền kinh tế thế giới đang dần chuyển từ mô hình truyền thống của các tổ chức tập trung sang mô hình phân quyền và tự chủ hơn. Sự thay đổi này đánh dấu việc ra đời của một thế hệ mới các tổ chức, doanh nghiệp "phi vật chất hóa": các tổ chức không yêu cầu văn phòng hay các tài sản vật lý khác, kể cả nhân viên. Với tiến trình đó, công nghệ Blockchain ra đời đang là cơn sốt thu hút nhiều tập đoàn lớn nghiên cứu, ứng dụng. Một số cái tên tiêu biểu như: Google đang nghiên cứu công nghệ Blockchain nhằm hỗ trợ mảng kinh doanh điện toán đám mây của tập đoàn này; Microsoft đã bày tỏ sự quan tâm của mình tới công nghệ Blockchain khi hợp tác với ConsenSys công bố phần mềm EBaaS (Ethereum Blockchain as a Service) trên nền tảng điện toán đám mây Azure; Tập đoàn IBM xây dựng một trung tâm nghiên cứu và phát triển

(R&D) Blockchain để triển khai công nghệ này trong nhiều lĩnh vực, sử dụng nền tảng Blockchain của chính công ty mình là Hyperledger. Nhiều ngân hàng đã chủ động xây dựng các trung tâm thí nghiệm R&D, và hợp tác với các nhà phát triển Blockchain để nghiên cứu tiềm năng của cuộc cách mạng công nghệ này. Ngoài ra, các tổ chức tài chính, học viện, công ty tư vấn hay chính phủ một số quốc gia cũng đã đi sâu nghiên cứu về Blockchain. Trong khi đó, các nhà phát triển phần mềm lẫn doanh nghiệp vẫn đang tìm con đường mới để sử dụng Bitcoin, Ethereum hay thậm chí xây dựng một Blockchain mới hoàn toàn.

Do đó, chúng ta có thể thấy công nghệ Blockchain sẽ ảnh hưởng sâu sắc đến cách thức các cá nhân, tổ chức hoạt động và cộng tác trên thế giới trong tương lai.

Một bạn trẻ muốn đặt chân vào Blockchain có những hướng phát triển nào?

Hiện nay, ngành công nghiệp Blockchain đang thu hút rất nhiều sự quan tâm của các chính phủ và doanh nghiệp. Theo một nghiên cứu của Tạp chí Forbes, Việt Nam có thể trở thành trung tâm ứng dụng công nghệ Blockchain mới của khu vực và trên thế giới. Tuy nhiên, nhân sự có kinh nghiệm trong lĩnh vực này chưa có nhiều. Các doanh nghiệp sẵn sàng tuyển các kỹ sư công nghệ thông tin nói chung có hứng thú làm việc với Blockchain, kết hợp với khả năng tự học hỏi nghiên cứu và quá trình đào tạo thực tế tại công ty, từ đó tạo cơ hội thực hiện phát triển sản phẩm Blockchain. Nếu các bạn trẻ đang suy nghĩ đến việc phát triển sự nghiệp tại lĩnh vực này thì câu trả lời là hoàn toàn có thể. Công nghệ này tuy mới và cần nhiều sáng tạo nhưng tương lai sẽ rất tươi sáng.

Về những ngành nghề trong Blockchain, các bạn có thể xem xét 2 định hướng sau:

1. Các vị trí kỹ thuật: Lập trình viên Blockchain đóng vai trò quan trọng hàng đầu trong một dự án Blockchain, tập trung phát triển phần cốt lõi của dự án dựa trên nền kiến thức rất chuyên sâu về công nghệ (smart contract, node, consensus,...). Ngoài ra, tương tự các công ty IT khác, một công ty Blockchain cũng có các vị trí lập trình viên phát triển các phần ít liên quan trực tiếp hơn tới Blockchain, các kiểm thử viên (QA, tester,...).
2. Các vị trí phi kỹ thuật: Ví dụ như phát triển sản phẩm, thiết kế, marketing, cộng đồng và khách hàng, nhân sự,... Tuy không yêu cầu chuyên môn kỹ thuật về Blockchain nhưng tất cả vị trí này đều cần bạn nghiên cứu nền tảng để nắm vững được kiến thức nhất định về công nghệ.

Dù làm việc tại vị trí nào, lĩnh vực Blockchain cũng đều rất thú vị. Tôi có thể liệt kê các lý do chính sau đây:

- Bạn đang tham dự vào một trong những ngành công nghệ mang tính đột phá và mới mẻ nhất tại Việt Nam cũng như trên toàn cầu. Ngoài ra, bạn có cơ hội nâng cao kiến thức về công nghệ Blockchain và học hỏi on-job-training[1], nâng cao trình độ ngoại ngữ, làm việc trong những dự án mang tầm cỡ quốc tế với các đồng nghiệp và các thành viên hỗ trợ từ khắp nơi trên thế giới,...

[1] *Đào tạo tại chỗ: chương trình đào tạo nội bộ được thiết kế giúp nhân viên có được kiến thức thực hành ngay tại nơi làm việc.*

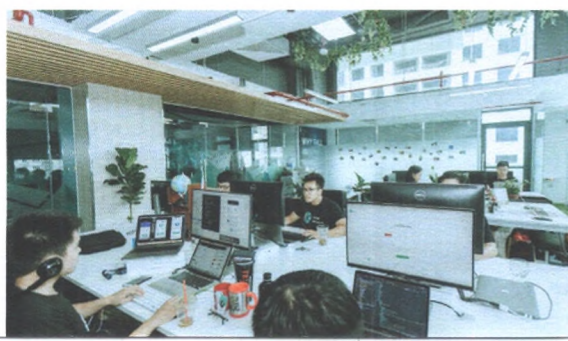
- Bạn sẽ song hành cùng với sự phát triển của dự án và công ty: Hiện nay, đa phần các dự án về Blockchain đều là startup. Do vậy, khi gia nhập vào một công ty Blockchain, bạn thường bắt đầu làm việc với một team nhỏ, sản phẩm còn sơ khai. Dần dần, dự án phát triển và mở rộng quy mô, bạn sẽ làm việc với team lớn hơn, sản phẩm ổn định và người dùng tăng trưởng lên,... Cứ như vậy, mỗi giai đoạn phát triển của dự án lại yêu cầu bạn những kỹ năng công việc khác nhau. Điều đó khuyến khích bạn luôn tự học hỏi, phát triển bản thân để thích nghi và đáp ứng yêu cầu công việc trong từng giai đoạn. Trải nghiệm này khó có được nếu bạn làm trong các doanh nghiệp lâu năm đã đi vào hoạt động ổn định.

- Chế độ lương thưởng khá hấp dẫn: Trung bình thu nhập của các vị trí Blockchain khá tốt so với mặt bằng thị trường lao động ngành IT nói chung. Ngoài ra, chỉ riêng ở các công ty Blockchain, nhân viên có thể nhận phần thưởng bằng chính các loại tiền mã hóa do công ty phát triển. Đây được coi là hình thức giúp nhân viên gắn bó và phát triển cùng công ty.

Các bạn trẻ muốn học về Blockchain ở Việt Nam có thể học ở trường Đại học, cơ sở đào tạo hay phải tự học, tự nghiên cứu? Và nếu tự học thì bắt đầu từ đâu?

Công nghệ Blockchain có liên quan mật thiết đến ngành khoa học máy tính và mật mã. Nếu bạn muốn học để trở thành một lập trình viên, các bạn có thể theo học ngành công nghệ thông tin ở các trường Đại học hiện nay. Bạn sẽ có được những nền tảng ban đầu về kiến trúc dữ liệu, giải thuật, mật mã học,... Ngoài ra, tại Việt Nam cũng có một số công ty công nghệ có những khóa học chuyên sâu về Blockchain mà các bạn có thể tham gia.

Nếu đã có nền tảng về CNTT, bạn có thể tự nghiên cứu về công nghệ này. Khác với cách đây vài năm, hiện nay các tài liệu nghiên cứu có sẵn rất nhiều trên mạng và dễ dàng tiếp cận. Có một số trang web ví dụ như: cryptozombies.io giúp bạn học ngôn ngữ Solidity (ngôn ngữ lập trình của Ethereum) miễn phí. Các dự án Blockchain hầu như đều có mã nguồn mở nên ai cũng có thể tiếp cận và nghiên cứu. Bạn còn có thể tham gia các buổi hội thảo, tọa đàm về công nghệ này. Hiện nay Kyber Network cũng thường xuyên hợp tác với các đối tác để tổ chức hội thảo chia sẻ và trao đổi các vấn đề xoay quanh thế giới Blockchain.



Streaming: Công nghệ đưa nhân loại đến gần nhau hơn

Khách mời phỏng vấn:
Nguyễn Doãn Minh Giang
Founder & CEO OWS Việt Nam
Kỹ sư CNTT Đại học Keio, Nhật Bản

Hiện nay, khái niệm "streaming" khá phổ biến và quen thuộc với cộng đồng người sử dụng công nghệ. Vậy công nghệ streaming là gì?

Có thể ví streaming với hình tượng đơn giản là dòng nước chảy liên tục. Đó là kỹ thuật giúp gửi một lượng dữ liệu liên tục và ổn định tới các đích đến, cho phép người dùng tiếp nhận xem/nghe gần như ngay lập tức. Thông thường, streaming[1] sẽ truyền các dữ liệu dạng audio hoặc video, nhưng đang dần có các hình thức khác xuất hiện.

Vì sao anh lựa chọn công nghệ Streaming để đến với ngành IT? Anh có thể chia sẻ một chút về câu chuyện của OWS?

"Ngành IT" là khái niệm tôi nghĩ hơi mơ hồ trong thời điểm hiện nay của loài người, không biết có nên gọi là "ngành IT" nữa không? Chúng ta đang đi vào thời đại số, công nghệ số thì chính xác hơn.

[1] *Trong nội dung cuộc nói chuyện này, streaming được nhắc đến là Media Streaming, trong đó các dữ liệu âm thanh và hình ảnh được gửi từ điểm thu đến điểm phát để thoả mãn các yêu cầu đặt ra như: chất lượng cao nhất hoặc tốc độ nhanh nhất.*

Từ trước khi bắt đầu, chúng tôi - đội ngũ những nhà sáng lập công ty - luôn khát khao đưa công nghệ vào giúp đời sống chúng ta tốt hơn, tích cực hơn. Chúng tôi cũng ước mơ xây dựng một công ty, tổ chức về công nghệ và đã dành rất nhiều thời gian tìm hiểu về các tập đoàn trên thế giới. Thực sự, họ đều có điểm gốc rễ là sở hữu những "công nghệ lõi". Các công ty công nghệ lớn trên thế giới đầu tư rất mạnh và giữ bản quyền cả công bố lẫn không công bố về những công nghệ này (đó mới thực sự là công ty công nghệ), như: Microsoft có hệ điều hành, nền tảng Office,... Google có công nghệ gốc "Search" (tìm kiếm),... Apple có hệ điều hành. Họ sở hữu và làm chủ các công nghệ hàng đầu thế giới. Vậy Việt Nam ta có gì? Và hàng ngàn doanh nghiệp công nghệ hiện nay thực sự đã mang tính "công nghệ" chưa? Nếu chúng ta không thể làm được những công nghệ gốc, mà chỉ làm công nghệ ứng dụng, ta sẽ mãi là nước gia công cho thế giới mà không có khả năng cho thuê chuyên gia với công nghệ mình sở hữu.

Streaming là một trong những công nghệ lõi và là bài toán đầu tiên chúng tôi muốn giải. Làm sao để chất lượng đường truyền tốt nhất? Làm sao kết nối mọi người dễ dàng hơn? Làm sao có thể giúp bác sĩ kết nối với người bệnh để khám chữa từ xa với yêu cầu độ chính xác cao, độ trễ của tín hiệu phải đạt dưới hàng trăm mi-li-giây? Làm sao để học viên và giáo viên cùng nhau học tập mà không phải đến trường (như trong đợt bệnh dịch vừa qua)? Làm sao ta có thể thỏa sức sáng tạo cùng đồng nghiệp làm việc từ xa, cùng bạn bè giải trí, xem phim cùng nhau? Sứ mệnh đầu tiên của chúng tôi là kết nối con người gần nhau hơn, thế giới phẳng hơn, cùng nhau học tập, làm việc mà không bị giới hạn bởi không gian, thời gian. Nghe streaming có vẻ đơn giản, nhưng khoảng thời gian chúng tôi bắt đầu chính thức (cũng như không chính thức) cũng ngót nghét gần chục năm để có thể làm chủ được công nghệ này.

Anh đánh giá thế nào về tiềm năng phát triển của lĩnh vực streaming tại Việt Nam? Những ứng dụng của công nghệ này là gì?

Công nghệ streaming thực sự là công nghệ gốc, công nghệ cốt lõi của nhiều giải pháp ứng dụng khác nhau. Do đó, thị trường này rất tiềm năng trên cả thế giới, không chỉ riêng Việt Nam.

Dịch Covid-19 vừa qua là minh chứng khi các nhà cung ứng dịch vụ streaming trên toàn thế giới đều quá tải, "cháy hàng". Giá cổ phiếu những công ty công nghệ streaming không ngừng tăng trong khi toàn thị trường đảo chiều.

Ngoài ra, những nguy cơ và thách thức của vấn đề an toàn thông tin trong các chương trình hội nghị quốc gia cũng đòi hỏi các doanh nghiệp trong nước phải làm chủ công nghệ streaming để đảm bảo an ninh. Các doanh nghiệp nội địa cần cung cấp sản phẩm streaming tốt, đáp ứng nhiều người dùng, an toàn, phù hợp với địa lý địa hình, hạ tầng Việt Nam hiện nay.

Streaming là công nghệ kết nối, do vậy ở đâu có nhu cầu kết nối, ở đó có thể áp dụng streaming. Mọi người thường gặp các ứng dụng của công nghệ này trong các dịch vụ truyền hình online, học/họp trực tuyến, gọi điện nhóm,... Ở đó, công nghệ này đóng vai trò gửi và nhận các tín hiệu âm thanh, hình ảnh giữa các người dùng.

Anh có nhận định gì về cơ hội việc làm liên quan tới streaming tại Việt Nam?

Nhu cầu ứng dụng streaming vào cuộc sống rất lớn, như các bài toán về Computer Vision (Thị giác máy tính), ứng dụng AI tìm kiếm nội dung trong ứng dụng streaming, các thiết bị IoT và con người tương tác gần nhau hơn, các sản phẩm liên quan tới VR[1], AR[2], video streaming 3D, các trò chơi tương tác thời gian thực,... Thị trường hiện nay có nhiều việc cần các kỹ sư phát triển các ứng dụng trong kỷ nguyên số. Và quan trọng hơn cả, quốc gia trong thời đại mới là quốc gia thịnh vượng có sở hữu những công nghệ và các công nghệ gốc - cốt lõi. Vì thế, cơ hội việc làm luôn rộng mở chào đón các bạn học các lĩnh vực liên quan tới công nghệ như: CNTT, Điện tử, Toán tin ứng dụng,...

Theo anh, các bạn trẻ muốn làm lĩnh vực streaming phải chuẩn bị những gì?

Tùy từng mức độ bài toán mà các bạn chuẩn bị các kiến thức và kĩ năng khác nhau.

Ở mức độ đơn giản, mọi người chỉ cần kỹ năng về xây dựng ứng dụng Mobile, Web hoặc PC là có thể xây dựng ứng dụng streaming (thông qua sử dụng các dịch vụ, mã nguồn mở có sẵn).

Sâu hơn nữa, nếu muốn chỉnh sửa, can thiệp vào các nhân streaming có sẵn, hay tự xây dựng, bạn cần các kiến thức về lập trình mạng, lập trình đa luồng. Về ngôn ngữ lập trình, bạn sẽ thuận lợi hơn nếu biết các ngôn ngữ có khả năng can thiệp sâu vào hệ thống, quản lý bộ nhớ tốt, ví dụ như C/C++.

Ở những tầng chuyên môn cao hơn, các bạn sẽ tự tìm hiểu các công nghệ để phù hợp với bài toán cần giải quyết, như các hệ thống phân tán, thuật toán nén, hạ tầng,...

Một bạn trẻ muốn đặt chân vào mảng này có những hướng phát triển nào? Anh có lời khuyên gì dành cho các bạn?

Như mình nói ở trên, tuỳ vào khả năng và mong muốn của mỗi người, các bạn có thể đảm nhiệm các vị trí khác nhau như: xây dựng ứng dụng Mobile, PC, Web nếu bạn có khả năng và yêu thích về mảng front-end; nếu đam mê về lập trình hệ thống, lập trình mạng, bạn có thể làm các công việc liên quan đến back-end; nếu thích thuật toán, bạn có thể tham gia các công việc liên quan đến thuật toán nén.

Mình tin vào sự thông minh của người Việt Nam, đặc biệt ở thế hệ trẻ được tiếp cận công nghệ từ khá sớm. Hơn nữa, điều quan trọng khi bắt đầu là bạn muốn làm và có quyết tâm. Hy vọng chúng ta có thể khẳng định với thế giới về một quốc gia sở hữu cũng như làm chủ nhiều công nghệ số nhất.

[1] *Virtual Reality - Thực tế ảo*
[2] *Augmented Reality - Thực tế tăng cường*

Chọn công ty – Nghĩ kỹ đi, rồi... QUYẾT BỪA!

Có lẽ ai trong chúng ta đều từng rơi vào trạng thái hoang mang, khó nghĩ khi đứng trước vô số các ngã rẽ giữa chọn nghề, chọn việc, chọn công ty,... Vậy ngành CNTT có những lựa chọn công việc ra sao? Mỗi môi trường công ty có điểm gì đặc thù? Và đâu là những "vũ khí" mà một lập trình viên cần trang bị cho "cuộc chiến" công nghệ này?

Tác giả: **Thi Măng Cụt**
Founder icetea.io

Dấn thân vào nghề

Vợ tôi học CNTT tại Đại học Quốc gia Hà Nội. Sở dĩ cô ấy chọn CNTT vì cấp 3 lực học tốt nên muốn học ngành "xứng tầm", khi đó công nghệ lại đang hot. Đa phần mọi người ở thời điểm ấy đều chọn trường Đại học theo cách như vậy. Sau gần 10 năm theo nghề, ban đầu là lập trình viên, rồi chuyển sang kiểm thử cho "nhàn và hợp với phụ nữ", cuối cùng, cô ấy cũng nghỉ việc và chuyển sang kinh doanh.

Tôi may mắn hơn chút. Do được tiếp xúc với máy tính từ cấp 3, tôi sớm phát hiện mình rất mê lập trình. Vì thế, chọn nghề CNTT với tôi dường như là hiển nhiên, được làm công việc yêu thích và thu nhập đủ sống là niềm hạnh phúc lớn.

Thế nên, nếu bạn định theo nghiệp lập trình viên, trước tiên, hãy thử lập trình cái gì đó trên máy tính: một trò chơi nhỏ hoặc một trang web cá nhân chẳng hạn. Nếu chưa biết làm, cứ tìm hướng dẫn trên Internet, có đủ cả.

Làm xong, hãy đánh giá xem:

- Nếu bạn thấy ham thích thì thật tuyệt vời. Bạn không cần chọn nghề nữa, nghề đã chọn bạn rồi. Chỉ cần nhớ, sau này trong công việc bạn còn gặp rất nhiều thứ "không thích", hãy kiên trì.
- Nếu bạn không thấy đặc biệt yêu thích, nhưng khả năng làm được và có chút hứng thú tìm hiểu thêm, đây vẫn là một lựa chọn nên cân nhắc. Phần lớn chúng ta không có đam mê nào rõ ràng, nên cứ chọn một công việc mà mình có thể làm tốt.
- Nếu bạn không thấy hứng thú gì hoặc nản chí, đây không phải ngành nghề phù hợp. Lập trình là nghề rất khó "cố" lâu dài.

Và khi quyết định theo nghề lập trình, bạn sẽ còn tiếp tục vấp phải thật nhiều những câu hỏi, đại loại như:

Nên tập trung học hay đi làm thêm khi còn trên ghế nhà trường?

Đi làm tập đoàn lớn hay startup? Gia công phần mềm hay làm sản phẩm?

Làm "ông chủ" hay "làm thuê"?

17 năm trong nghề với đủ mọi thăng trầm buồn vui, tôi xin chia sẻ vài kinh nghiệm tôi trải qua và đúc kết được về đặc điểm của từng kiểu công ty, các vị trí làm việc. Tuy nhiên, những gì bạn đọc mới dừng ở lý thuyết, là trải nghiệm của riêng tôi, bạn phải thực sự trải nghiệm mới ăn thua. Có thể bạn không tin, nhưng đa số các bạn nữ khi được hỏi tiêu chuẩn về người chồng tương lai họ nói một kiểu, đến khi lấy chồng lại ra người kiểu khác, chả liên quan.

Ngành CNTT: Môi trường nào phù hợp với bạn?

 # Làm việc ở công ty nước ngoài

Nơi đầu tiên tôi làm việc là một chi nhánh công ty Mỹ lai Thụy Sỹ tại Việt Nam. Tôi vẫn nhớ ngày đầu đi làm là 1/10/2003. Lý do chọn đơn giản là mác công ty nước ngoài nghe khá oai, lương khởi điểm 200 USD (khá cao thời đó), sau 5 tháng còn được tăng gấp rưỡi. Công ty khoảng 20 người, giờ giấc thoải mái, công việc không vất vả, sếp hay cho đi uống bia và du lịch miễn phí.

Điều quý giá với tôi khi làm việc ở đây là được đọc một số cuốn sách rất hay. Đáng kể nhất có cuốn *Code Complete* của một bác ở Microsoft viết, và cuốn *Design Patterns* của "Gang of Four", cuốn sách làm toát lên

vẻ đẹp của lập trình hướng đối tượng. Mỗi tội sau đó tôi bị bệnh "lạm dụng design patterns" đến khổ, cứ tưởng thế là hay, code trở nên trừu tượng và tốn công sức khi bảo trì. Sau này, tôi nhận ra code càng đơn giản, càng ít thì càng tốt. Các bạn trẻ nên tự hào vì code của mình đơn giản, thay vì xấu hổ và cố nhồi nhét kiến trúc cao siêu.

Một điều đáng nói nữa, tôi nghỉ việc ở công ty này không có lý do gì chính đáng, thậm chí chẳng thèm tìm chỗ làm mới. Có lẽ đó là cách nghỉ việc của tuổi trẻ khi có nhiều năng lượng và cảm xúc mà bản thân không hiểu được. Giống như con sóng cứ dao động và tiến lên, không cần hiểu lý do. Có chăng là cảm giác "kích thích" khi bắn mail nghỉ việc, thấy oai oai.

 # Làm việc ở tập đoàn lớn

Thời gian làm việc lâu nhất của tôi là tại FPT Software (FSOFT), công ty gia công phần mềm lớn có quy mô nhân sự vượt trội. Nếu bạn tự lập một công ty gia công phần mềm, bạn sẽ hiểu vượt quy mô 100 người khó như thế nào, để thấy FSOFT là một hiện tượng rất khó lặp lại ở ngành gia công phần mềm Việt Nam.

Ngày đó, FPT có danh tiếng rất tốt trong xã hội, kiểu hình ảnh một công ty nhiều người tài năng đang "làm thứ gì đó khang khác". Vì thế, đa số sinh viên đều mong muốn được thử sức. Ban đầu, tôi nộp đơn vào 2 bộ phận khác nhau, một đơn xin làm lập trình viên, một đơn xin làm tester. Cả 2 đều... tạch.

Sau này, tôi được người khác giới thiệu nên chạy qua trà đá chém gió với anh leader rồi được nhận luôn, chả cần phỏng vấn. Tôi còn được xếp hạng "level 3", tầng team lead. Thế mới thấy sức mạnh của quan hệ lớn thế nào, chứ nếu đường đường chính chính nộp đơn chắc tôi lại... tạch tiếp.

Ban đầu, tôi làm lập trình viên thông thường, công việc chỉ bận hơn công ty cũ một chút. Nhưng khi bắt đầu chuyển sang phụ trách kỹ thuật rồi quản trị dự án, nơi đây trở thành chốn "chôn vùi tuổi thanh xuân" của tôi. Các dự án làm ngoài giờ liên miên, cả năm trời có khi không ngẩng đầu lên, không biết chuyện gì đang xảy ra ngoài xã hội, nhiều đêm ăn ngủ tại công ty.

Sau này, kể cả ra khởi nghiệp, tôi cũng chưa bao giờ "cống hiến" đến mức như vậy. Nhưng phải thừa nhận, những sức ép như vậy mới khiến mình trưởng thành. Từ khả năng quản lý stress, quản lý thời gian, giao việc, tạo động lực cho nhân viên, giám sát và đốc thúc, báo cáo và thảo luận với khách hàng cho tới tiếp khách,... Chẳng thế mà rất nhiều người khởi nghiệp thành công từng từ "lò" này mà ra.

Ở FSOFT, tôi không học được nhiều về kỹ thuật và nghiệp vụ. Thay vào đó, tôi học được kỹ năng mềm, cách làm việc chuyên nghiệp, và tinh thần "đếch biết gì cũng tiến". Lý thuyết sách vở thường khuyên ta chỉ nhận việc trong khả năng đảm bảo được để giữ chất lượng, tiến độ và phát triển bền vững. Ở FSOFT không cần như thế, cứ nhận bừa, ôm hết, dù chưa có kinh nghiệm và thiếu nhân lực đi nữa cũng cố tìm cách mà xoay, rồi sẽ OK hết. Có dự án thành công, có cái "chưa thành công lắm", nhưng công ty càng lúc càng phình to, dự án càng lúc càng nhiều, nên chắc làm vậy là đúng. Chỗ nào chưa biết cứ ném đá dò đường mà đi, tinh thần xung phong rất được khuyến khích.

Một ưu điểm nữa của FSOFT là rất "sợ khách hàng". Người Việt thường sợ sếp chưa đủ, nhất là FSOFT không có văn hóa kỷ luật như Viettel hay VinGroup. Việc "sợ khách hàng" tạo thành sức ép chuyển dịch qua các tầng quản lý, vươn đến từng ngõ ngách công ty, thúc đẩy mọi người cố gắng.

Tham vọng lớn, cái sự "đếch biết gì cũng tiến" và văn hoá sợ khách hàng có lẽ là nền tảng tạo ra FSOFT ngày nay.

Điều dễ nhận thấy nhất ở những công ty lớn và vẫn tăng trưởng nhanh như FSOFT là chất lượng nhân sự không đồng đều. FSOFT đủ lớn để nhận người đủ loại trình độ, người hay kẻ dở tùm lum. Kể cả trình độ hơi "đuối", bạn vẫn có thể được nhận vì ở đây có nhiều công việc đa dạng. Và người ta cố gắng khắc phục điều đó bằng quy trình để đảm bảo làm ra sản phẩm có chất lượng ổn định ở mức chấp nhận được (nhiều bạn sẽ than phiền làm tài liệu quá nhiều, nhiều "quy trình" mang tính đối phó). Dẫu vậy, các dự án "cháy", các vấn đề vẫn ở khắp nơi. Tuy nhiên, một công ty tăng trưởng tốt giống như một dòng sông cuộn chảy, sẽ cuốn đi hầu hết các rác thải.

Nếu giỏi, (tất nhiên) bạn có rất nhiều cơ hội. Nếu máu (đặc biệt máu làm quản lý) và có năng lực, bạn có thể leo lên vị trí quản lý cả trăm người rất nhanh, vì công ty mở rộng liên tục. Cơ hội đi nước ngoài cũng nhiều. Đây là điểm khác biệt so với khi bạn làm ở một công ty nước ngoài tại Việt Nam, tuy lương ban đầu cao hơn và sức ép công việc ít hơn, nhưng làm mãi vẫn vậy. Công ty có bấy nhiêu người, ít có vị trí "hổng" thì thăng tiến vào đâu?

Ở FSOFT, nếu làm tốt, chỉ vài năm bạn sẽ phải đối diện với lựa chọn: đi theo con đường làm kỹ thuật hay quản lý. Nếu có thiên hướng quản lý, bạn chả phải đắn đo. Nhưng nếu bạn là dân kỹ thuật, đây lại là điều cần cân nhắc.

Thực tế, hầu hết mọi người chọn con đường quản lý. Bởi đối với một công ty tăng trưởng dựa trên quy mô nhân sự, quản lý nói chung sẽ cần thiết và được trọng dụng hơn. Bạn trở thành "sếp", oai hơn, có quyền hơn, và thường kiếm được nhiều tiền hơn. Tôi cũng đã chọn con đường đó. Cho đến lúc đối diện với ngã rẽ: tiếp tục leo tiếp lên cao hay dừng lại. Làm quản lý cấp cao đòi hỏi một số tố chất mà dân kỹ thuật tài tử như tôi bị thiếu: sự nghiêm khắc, sẵn sàng làm việc "ác", tính kỷ luật, nghiêm túc, khả năng tập hợp chiến hữu,... Bù lại, tôi có vài ưu điểm khác nên nếu cố gắng cũng gần được tròn vai. Thế nhưng thời điểm đó, tôi chọn rời đi vì công việc đã lâu không còn đem lại niềm vui.

 # Làm việc ở Startup

Tôi tham gia Kyber Network khi công ty còn rất non trẻ này vừa gọi thành công số vốn rất lớn. Lúc đó, Kyber vẫn là một nhóm nhỏ ngồi trong căn phòng chật chội tại UP Coworking Space. Đây cũng là lúc công ty bắt đầu phình lên và muốn tìm thêm người có kinh nghiệm giúp quản trị dự án và hỗ trợ các vấn đề chung. Tôi được bạn giới thiệu (lại là sức mạnh của quan hệ), nhân khi đang rảnh, thấy offer có vẻ ổn và cũng muốn tìm hiểu xem Blockchain là gì mà dân tình có vẻ sục sôi như vậy, tôi đồng ý tham gia.

Vì mới gọi được vốn nên công ty cũng khá "rủng rỉnh" so với các startup khác. Làm việc ở đây khá thoải mái, văn phòng đẹp, có khu để tiêu khiển, lãnh đạo có năng lực, nhiều bạn trẻ tài năng.

Kyber bắt rất đúng thời điểm khi làn sóng công nghệ Blockchain đang "lên đồng". Giữa rừng các dự án lừa đảo, Kyber là dự án nghiêm túc và thành công. Điều này đòi hỏi năng lực kỹ thuật để nắm bắt được bản chất công nghệ mới, năng lực thực thi tốt, và một chút may mắn. Cứ tầm 10 năm lại có 1 làn sóng "gold rush"[1] như vậy. Đa số đứng ngoài quan sát, số khác tham gia và lãnh những kinh nghiệm thương đau, không có nhiều người nắm bắt và phát triển một cách chân chính và thành công.

Tôi học được ở đây những kiến thức và kĩ năng lập trình về Blockchain, điều mà chỉ đọc sách sẽ rất trừu tượng. Từ kinh nghiệm tích lũy khi làm cho Kyber Network, tôi đã lập ra icetea.io, startup về Blockchain sau này. Nếu bạn muốn học về công nghệ mới (như AI, Blockchain, thực tế ảo,...), tốt hơn hết hãy tham gia các công ty làm thực về lĩnh vực này.

Đối với lập trình viên, làm ở startup được sử dụng công nghệ mới hơn, không phải tạo các tài liệu "vô bổ" và tuân thủ quy trình mang tính đối phó như tập đoàn lớn. Ngoài ra, khi vào các công ty startup nhỏ, bạn thường được gặp và trao đổi với Founder, nghe câu chuyện của họ, bạn sẽ cảm nhận nguồn năng lượng tích cực và nhiệt huyết của người chủ.

 # Làm freelance

Nếu làm ở công ty, sếp giao gì bạn làm ấy, có dự án nào làm dự án đó. Còn freelance, bạn có thể chọn dự án và đề xuất công nghệ. Bạn nào tính cách tự do, khó chịu với sự "áp đặt" từ sếp, lựa chọn loại hình này rất phù hợp.

Freelance có nhiều kiểu, từ chuyên nghiệp toàn thời gian cho đến làm thêm ngoài giờ, từ làm trong nước cho đến làm trên các nền tảng freelance nước ngoài như Upwork.

Điểm hấp dẫn nhất của freelance là bạn tự do về thời gian và địa điểm. Cách làm này phù hợp với những bạn thích tự mày mò, làm việc độc lập, thích một mình. Tôi có một số người bạn làm freelance thành công trên các nền tảng nước ngoài. Họ trông khá "phong lưu", luôn sẵn sàng cà phê cà pháo. Nhưng để được như vậy, họ phải dày công xây dựng profile trên nền tảng họ tham gia. Làm freelance yêu cầu năng lực lập trình và giải quyết vấn đề của khách hàng tốt. Làm tốt và kiên nhẫn thì uy tín sẽ tăng và có thu nhập tốt và ổn định dần.

Điểm trừ khi làm freelance cho khách Tây là lệch múi giờ, bạn phải thức khuya để trao đổi. Các bạn trẻ quen đêm cày ngày thức thì OK, nhưng với người có gia đình, có tuổi, sẽ khá bất tiện. Công việc này cũng không ổn định, lúc có lúc không và biến động theo mùa vụ.

Nhiều bạn đang làm chính ở công ty rồi nhận thêm việc ngoài. Đây cũng là cách tốt để kiếm thêm và nâng cao tay nghề. Tuy nhiên, bạn sẽ rất mệt nếu tham lam. Ngoài ra, không nên làm việc ngoài trong giờ ở công ty, bạn sẽ gây ấn tượng rất xấu.

Tôi từng nhận 1 dự án làm thêm khá lớn, thấy tiền cũng được nên tham. Lúc nhận dự án đó công việc ở công ty đang khá nhàn, ai dè nhận xong sếp giao phụ trách 1 dự án khó. Ban ngày làm đã mệt, tối đêm lại ngồi cày, loáng cái đã 3, 4 giờ sáng, kéo dài trong nhiều ngày, đầu óc mê mụ. Mà nào đã hết, do không thảo luận kỹ các điều kiện

[1] *cuộc đổ xô đi tìm "vàng", ở đây hiểu là các cơ hội lớn*

về bảo hành và bảo trì, sau cứ có việc gì khách hàng lại gọi "làm phiền", đeo bám mãi mấy năm mới thôi.

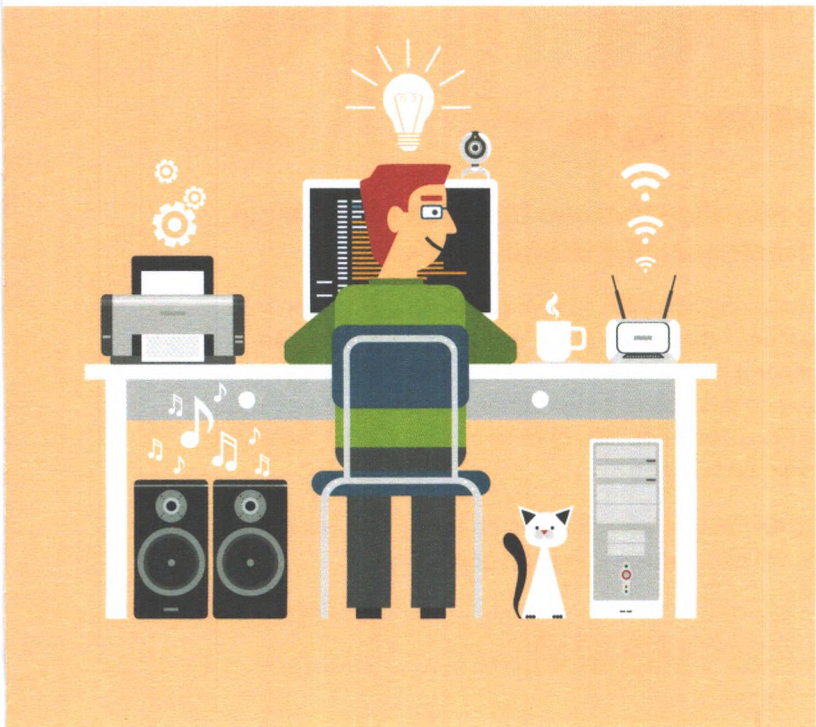
Số lượng freelancer trên thế giới ngày càng tăng nhanh dẫn đến cạnh tranh cao. Bạn cần kiên nhẫn chứ không thể ngày một ngày hai được. Ở lĩnh vực nào cũng thế, có một số ít rất thành công và được đưa ra làm hình mẫu, đa số còn lại chỉ đạt kết quả làng nhàng. Những bạn thành công, ngoài lý do năng lực kỹ thuật, họ còn biết đặt mình vào vị trí khách hàng để hiểu, trao đổi sớm, có nguyên tắc, trách nhiệm, biết nói không, không hứa hẹn quá khả năng và kiên nhẫn xây dựng quan hệ cũng như uy tín.

Nghề freelance toàn thời gian nói chung hợp với số ít có năng lực và tính cách phù hợp: có thể độc lập giải quyết vấn đề và thích độc lập tự do. Ngoài ra để làm với nước ngoài, bạn cần khả năng giao tiếp ngoại ngữ. Nó không dành cho số đông.

 # Khởi nghiệp

Khởi nghiệp có sức quyến rũ rất lớn. Các bạn trẻ nhìn thấy ở đó ánh hào quang: Trở thành "Founder" hay "CEO" này nọ, nghe thật oách. Mọi người thấy bạn "giỏi giang".

Khi còn làm cho Kyber Network, tôi nhận ra Blockchain là công nghệ khá hay ho dù còn non trẻ và cần thêm rất nhiều thời gian để hoàn thiện hơn. Điều đáng nói là công nghệ này chỉ ứng dụng loanh quanh làm sàn giao dịch coin, ứng dụng bài bạc, game nuôi thú ảo, và hơn nữa chỉ dành cho những người trong "giới crypto". Chúng quá phức tạp, phiền nhiễu, và rủi ro để dùng cho "người bình thường". Trong khi đó, Blockchain có được sử dụng rộng rãi thì chúng ta mới nhận ra các điểm yếu của nó để khắc phục, và đó là cách tốt nhất để công nghệ này trưởng thành. Đó là lý do tôi lập ra icetea.io, một startup nhỏ với mục tiêu đưa Blockchain từ tủ kính ra thế giới thực, đến với người dùng thông thường.



Ai bắt đầu khởi nghiệp cũng cần một lý do như vậy, để "hợp lý hoá" cái máu khởi nghiệp trong mình. Sau mấy tháng hì hụi làm ra MVP[1] chạy được, tôi bắt đầu đi gọi vốn. Quá trình gọi vốn dài đằng đẵng, hao tâm tổn lực, và khiến tôi không còn tập trung

[1] *Minimum Viable Product: sản phẩm khả dụng tối thiểu*

phát triển sản phẩm được nữa. May mắn lắm thì gọi được một ít để duy trì doanh nghiệp thêm một thời gian, còn không, chẳng mấy chốc nguồn vốn cạn kiệt, sản phẩm chỉ đủ để "show" chứ làm gì đã kiếm được tiền. Và cái ngày trả lương cho nhân viên hằng tháng bỗng chốc trở nên đáng sợ.

Tôi thuộc thế hệ "già" mới làm khởi nghiệp, nên chúng tôi "lấy ngắn nuôi dài", làm thêm nhiều việc lặt vặt để kiếm tiền, không phụ thuộc vào nguồn vốn nhà đầu tư. Thời kỳ "bong bóng startup" cũng đã qua, nên nếu các bạn muốn bắt đầu, có lẽ câu hỏi quan trọng nhất là làm sao duy trì startup trong 1 thời gian dài để tìm ra "mạch nước", trước khi nghĩ đến chuyện gọi vốn.

Vậy, rốt cục nên chọn làm cho tập đoàn lớn, startup, freelancer hay tự khởi nghiệp?

Câu trả lời tuỳ thuộc vào năng lực và tính cách của mỗi người. Một số bạn có "năng khiếu" đưa ra lựa chọn khá dễ dàng và tự nhiên, cơ hội đến là theo, không nghĩ nhiều; một số khác thường đắn đo. Cần khẳng định, không có lựa chọn nào là tốt hơn cả, chúng đều có mặt tốt mặt xấu, đều là sự đánh đổi. Nếu rõ ràng có một thứ tốt hơn, bạn đã chẳng phân vân, phải không?

Hy vọng những gì tôi mô tả ở trên giúp bạn có chút ý niệm loại công việc nào phù hợp với năng lực và tính cách bản thân.

Nhưng dù làm ở đâu, bạn cũng cần chú ý đến...

 # Khả năng tự học

Đối với ngành lập trình, các kiến thức học trong trường chỉ sơ đẳng và tương đối cũ. Trong khi công nghệ thay đổi nhanh chóng như màu tóc hot girl. Vì thế, năng lực tự học và học qua công việc là tối quan trọng nếu bạn muốn tiến xa trên con đường chuyên môn.

Nếu cần thước đo chuẩn xác nhất về năng lực chuyên môn, đó là: khả năng học (learn) khi bạn còn trẻ, và khả năng "gạt đi cái đã học" (unlearn) khi bạn là lập trình viên có thâm niên.

Hồi sinh viên, tôi mua 1 cuốn sách lập trình ở hiệu sách cũ trên đường Láng, cứ thế mày mò. Sau này có Internet thì tha hồ tra cứu. Bạn có thể tìm hiểu cơ bản trước, sau đó làm theo các bài chỉ dẫn. Đọc nhiều tiếng Anh tự khắc lúc nào đó trình độ tiếng Anh lên hẳn level mới.

 # Đi làm từ khi sinh viên

Ngày sinh viên, tôi không có nhiều cơ hội đi làm, chưa kể tôi lại là người hướng nội và tự ti. Khi đó, tôi ngồi mày mò code hết ứng dụng này đến ứng dụng khác, kiểu chơi cờ caro, xem bói tử vi,... Mạng mẽo chả có, chỉ có vài đứa bạn đến chơi thử (giá mà có mấy cái chợ như Apple AppStore hay Google Play để đưa lên như bây giờ).

Tôi thuộc phe ủng hộ làm thêm khi còn sinh viên. Và nếu làm từ xa trong 1 nhóm có cả những người từ nước ngoài càng tốt, chẳng hạn tham gia vào các dự án freelance, dự án nguồn mở. Các dự án nguồn mở đa phần có tiêu chuẩn và cách làm chuẩn mực hơn dự án đóng (mở ra cả thế giới nhìn thấy nên cũng phải ngon nghẻ chút).

Khi làm thêm, áp lực và trách nhiệm khiến bạn học được rất nhiều, khác xa với các bài tập trên lớp. Bạn sẽ gặp vô số những "lần đầu tiên": Lần đầu tiên đối diện với việc hứa hẹn quá khả năng rồi cày như trâu để giữ lời; Lần đầu tiên thất hứa; Lần đầu tiên phải đặt bản thân vào góc nhìn của khách hàng, chấp nhận bỏ đi cái mình vỗ ngực cho là "hay ho" nhưng không phải ý khách hàng muốn,...

 # Công nghệ và thời trang

Giống thời trang, vào bất kỳ thời điểm nào công nghệ cũng có vài thứ trở nên hot. Các công nghệ mới có sức hút rất mạnh, nhất là với các bạn trẻ. Lúc có tuổi hơn, bạn bắt đầu hờ hững dần, thấy dùng công nghệ cũ, ổn định, và đã được chứng minh chắc ăn hơn lại đỡ mệt.

Có nhiều công nghệ rất "trending", ngỡ đâu ngon lắm, rồi ít lâu lặn mất tiêu, chẳng ăn thua. Hồi tôi mới ra trường, đi đâu cũng thấy nói về XML, như thể nó làm thay đổi thế giới. Bẵng đi một thời gian thấy chìm dần, hóa ra chả có vẹo gì.

Một số bạn hay hỏi: "Nhiều công nghệ quá, cái này cái kia giờ có vẻ thời thượng, có nên học không?". Thật ra, như thế nào mà chả được, cứ tìm hiểu công nghệ mới nếu bạn đủ hào hứng và thời gian cho phép, còn không thì thôi, dùng cái cũ đã. Nếu làm cho khách hàng, nên chọn công nghệ mà bạn thấy tự tin thoải mái một chút để đảm bảo tiến độ và chất lượng, còn tự làm theo sở thích thì cứ... xoã.

 # Thái độ khi xin việc và nghỉ việc

Người ta hay so sánh giữa "thái độ" và "trình độ". Trình độ khó thay đổi ngày một ngày hai. Còn thái độ, nếu muốn

ngon, cứ tập trung vào thái độ khi xin việc và nghỉ việc trước. Làm tốt cái này bạn sẽ có tiếng tốt, đời ấm no.

Nhớ lại lần đầu tiên đi xin việc, dù chưa có kinh nghiệm thực tế, tôi ghi trong CV rất nhiều công nghệ cao siêu. Đi phỏng vấn cứ lên gân như thể mình giỏi lắm, rốt cục chả ma nào nhận. Qua vài bận như thế, tôi bắt đầu chùng xuống, sửa CV cho khiêm tốn và đặc biệt chỉnh sửa thái độ cầu thị, lắng nghe và tôn trọng người phỏng vấn, y như rằng được nhận rào rào. Có chỗ trước đã từ chối vì chê thái độ có vẻ bất cần, tôi bèn viết email "bày tỏ lòng thành", nào ngờ họ đổi ý nhận luôn. Xin việc giống như đi cưa gái ấy, cứ chân thành bạn đã thắng hơn nửa.

Sau này, khi tôi là người phỏng vấn, gặp ứng viên có thái độ tốt, kể cả trình độ không phù hợp với vị trí cần tuyển, tôi đều cố tìm xem có vị trí nào khác phù hợp với bạn này không. Không được nữa thì giới thiệu sang nơi khác, rồi ghi nhận lại hẹn có vị trí phù hợp sẽ gọi,...

Khi nghỉ việc cũng vậy, bạn cần trao đổi sớm để công ty có thời gian xoay xở và thực hiện bàn giao đầy đủ. Hãy suy nghĩ làm sao cho việc mình nghỉ ít ảnh hưởng đến công ty nhất. Kể cả bạn có chán ghét công ty, đó cũng là điều nên làm vì lợi ích của chính bạn. Trước tôi có làm cùng một cô mà tôi không thích lắm. Nhưng khi nghỉ, cô ấy làm tài liệu rất chi tiết và tâm huyết để những người còn lại có thể nắm được công việc. Đây là điểm cộng lớn đến mức sau này cứ nói chuyện liên quan đến chủ đề này, tôi lại lôi cô ấy ra làm mẫu mực.

Hãy nói về điểm tốt khi nhắc về công ty cũ, công ty nào chả có vài điểm tốt.

Sau cùng, cứ... quyết bừa

Kinh qua kha khá các loại công ty, rất nhiều lần tôi đứng trước các lựa chọn. Thế nhưng, tôi chưa bao giờ nuối tiếc về bất kỳ quyết định nào. Điều tiếc nuối, nếu có, là nhiều lần đã cân nhắc quá lâu, chứ không phải chuyện sai hay đúng. Cá nhân tôi nhận thấy, phân vân lâu chưa bao giờ giúp mình lựa chọn chính xác hơn. Ấy là chưa kể, việc phân vân không thể ra quyết định giống như bạn cứ bơm hơi mãi vào quả bóng bay mà không cho nó xì, lâu ngày dễ gây stress. Mệt!

"Chọn con tim hay nghe lý trí?" là câu hỏi kinh điển, có lẽ chỉ thua mỗi câu "Tồn tại hay không tồn tại" của hoàng tử Hamlet.

Khi phân vân ta hay hỏi ý kiến người khác. Nhiều người còn tìm đến thầy bói làm chỗ dựa tinh thần. Để ý xem, chúng ta tuy làm như mèo mửa nhưng khuyên bảo người khác lại rất hay.

Kiểu mấy ông bà ế chỏng chơ lại như "chuyên gia tâm lý" khi gỡ rối chuyện tình cảm người khác. Này, tôi mà làm được theo những điều tôi khuyên người khác, cuộc đời tôi đã ngon nghẻ khác hẳn rồi.

Thế nên, khi lâm vào tình thế khó, tôi thường tự hỏi: "Nếu tôi là thằng khác, tôi sẽ khuyên chính tôi làm gì lúc này?", rồi cứ thế làm theo. Đảm bảo hiệu nghiệm hơn câu "Hỏi ý kiến vợ xong làm ngược lại" - câu này vớ vẩn, vợ nói chuẩn lắm, đừng đùa.

Khi khuyên người khác ta dùng lý trí, còn khi tự thân hành động cảm xúc mới là thứ chi phối. Như vậy, về lý thuyết, dùng lý trí tốt hơn. Vậy mà, nhìn lại, hết thảy mọi quyết định lớn trong quá khứ của tôi, về bản chất, đều cảm tính, nôm na gọi là "chọn bừa". Nhưng sau đó, tôi luôn tìm được lý do mang tính lý trí để biện giải, hợp lý hoá. Hoá ra lý trí tuy "khoa học" đấy, nhưng lại rất mềm dẻo, "điều chỉnh" uốn nắn khá dễ. Kết quả là quyết định cảm tính của ta được "chống lưng", "phê chuẩn" bởi lý trí. Điều này khác hoàn toàn với việc quyết định trong cơn "cảm xúc", như giận dữ, say xỉn, hoặc lòng tham nổi lên - một điều chắc không cần phải nói ai cũng biết là nên tránh.

Khi quan tâm đến cơ hội nào, tôi thường đến tận nơi gặp, trao đổi, cảm nhận không khí và văn hoá của công ty, các công việc mà công ty đang làm, những điều mà công ty hướng đến, lương và quyền lợi có ổn không,... Sau đó về nhà ngủ vài hôm để mọi thứ lắng lắng xuống. Rồi quyết bừa!

Vì vậy, bạn cứ thoải mái... quyết định bừa, chỉ cần tìm được lý do phù hợp để biện hộ và cảm thấy an lòng, đừng vi phạm đạo đức là được.

Vì vậy, bạn cứ thoải mái... quyết định bừa, chỉ cần tìm được lý do phù hợp để biện hộ và cảm thấy an lòng, đừng vi phạm đạo đức là được.

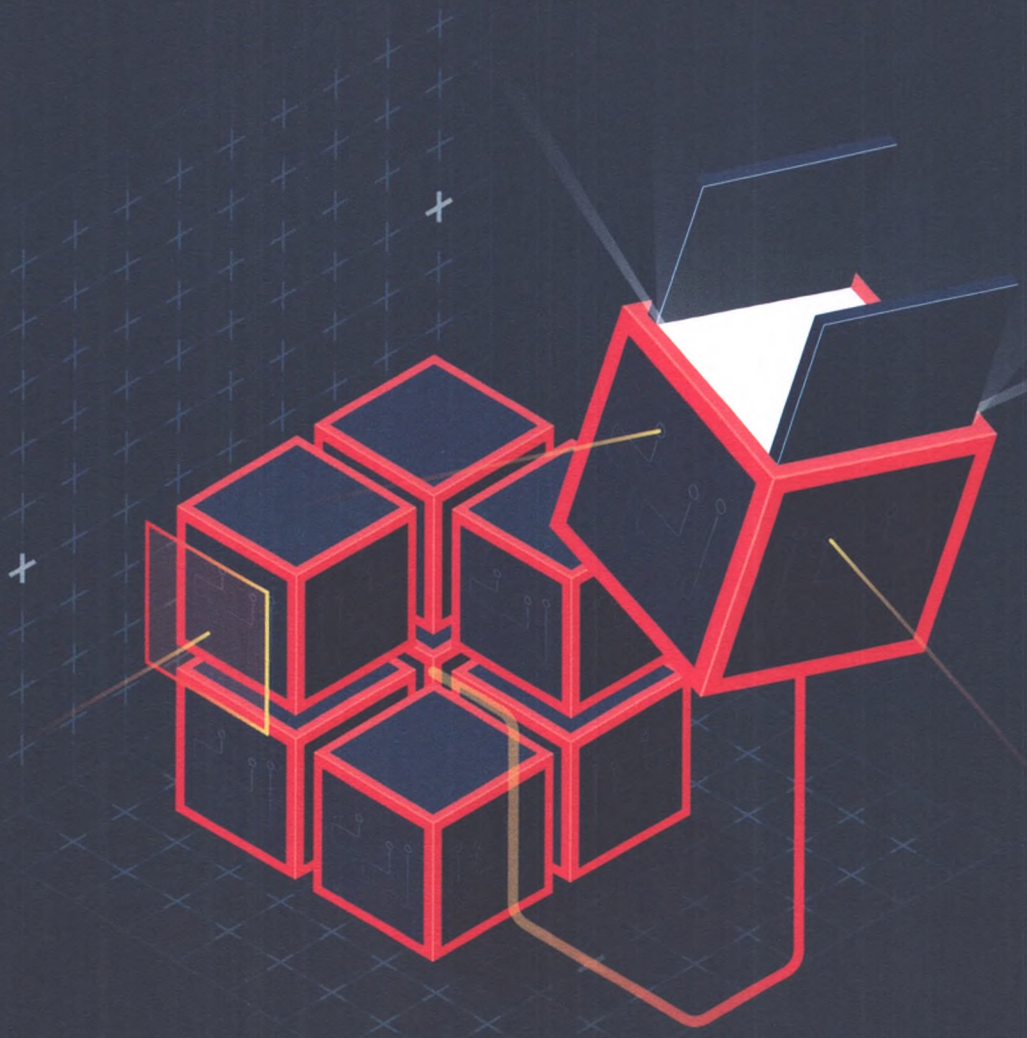
Bạn hãy có lòng can đảm ra quyết định, can đảm dừng lại khi nhận ra mình sai, và thành thật với cảm xúc của bản thân. Thử hình dung cuộc sống sẽ còn gì thú vị nếu tất cả lựa chọn đều đúng và hoàn hảo?

Vậy là chúng mình đã kết thúc phần đầu của cuốn sách. Hy vọng bạn đã nắm bắt được các thông tin hữu ích về ngành IT, cũng như các vị trí công việc, môi trường công ty để định hướng rõ ràng hơn cho bản thân nếu có ý định lựa chọn ngành này.

Góc bật mí:

Để khám phá xem bản thân mình phù hợp với môi trường nào, bạn có thể đọc những bài chia sẻ kinh nghiệm tại app Spiderum, hoặc làm các bài trắc test trắc nghiệm tính cách thông qua app của TopCV nhé:

Quét để tải ứng dụng



2

MUÔN NẺO ĐƯỜNG NGHỀ

Muốn học lập trình nên bắt đầu từ đâu?

Tác giả: **Scarlet**
Sinh viên năm nhất ngành Computer Science
tại University of British Columbia, Canada

Xin chào, mình tên là Scarlet, hiện đang học năm nhất ở trường University of British Columbia, Canada. Hiện tại là năm thứ 6 mình sống trên đất nước lá phong này.
Mình bắt đầu lập trình từ 2 năm trước và từ một đứa không biết gì, mình đã thành công thu được 3 kỳ thực tập ở những công ty lớn và thắng nhiều cuộc thi công nghệ trong thành phố. Cho đến giờ, cảm giác khó khăn khi tìm tòi về lập trình vẫn còn rất mới với bản thân mình và mình hiểu làm cách nào để vượt qua. Mong các bạn newbie sẽ thiết lập được một điểm bắt đầu sau khi đọc bài :)

"Muốn học lập trình nên bắt đầu từ đâu?" là câu hỏi rất phổ biến với những ai còn tò mò và đang chập chững tiếp cận với thế giới lập trình nhưng bị chết trôi bởi quá nhiều nguồn tràn lan trên mạng. Vì cũng nhận được một lượng lớn tin nhắn mọi người hỏi về vấn đề này, mình nghĩ đây là một dịp tốt để chia sẻ về road-map của bản thân (những gì mình ước đã biết ở thời gian đầu) và làm thế nào để phá băng vào ngành một cách hiệu quả nhất.

Lần đầu tiên mình được tiếp xúc với thế giới lập trình là khi qua thành phố Seattle (Mỹ) và thăm anh chị họ đang học Computer Science (Khoa học Máy tính) tại trường Đại học Washington. Lúc nghe mình đang khám phá nhiều loại ngành nghề, hai người lập tức lôi mình ra một góc cùng cái laptop để vừa làm bài tập trên lớp vừa say sưa giảng những thứ căn bản như biến (variable), điều

kiện (condition),... và gợi ý mình lên trang web Codecademy làm thử vài bài tập đơn giản. Trước khi về, anh chị xoa đầu mình, chúc may mắn rồi dúi vào tay mình cái USB chứa bản PDF của cuốn sách giáo khoa hơn 1,000 trang mà hai người đang học trong khóa lập trình cơ bản trên đại học, bảo mình thích thì đọc cho vui.

Hồi đấy, mình ngó cuốn sách thấy dày quá nên về nhà bỏ hẳn qua một bên, háo hức google "*Làm thế nào để học lập trình*" rồi ngồi xem một núi tutorials trên YouTube, đọc linh tinh rất nhiều thứ khác và cứ cắm đầu vào chỉ dẫn trên Codecademy vì tin tưởng trang này sẽ đào tạo mình thành một lập trình viên sừng sỏ.

1 tuần sau, mình bỏ cuộc.

Lúc đấy mình chẳng hiểu sao cứ đọc mãi nhưng chẳng hiểu gì cả, càng học càng rối, càng đau đầu. Như đa số những người ngoại đạo khác, mình bị nhấn chìm trong nguồn tài liệu trên mạng và cảm giác bất lực khi bản thân chưa đủ kinh nghiệm để tự lọc ra thông tin cho việc học.

Bẵng đi một thời gian cùng với rất nhiều lần thử sau đó, cuối cùng mình cũng lần mò được các bước cần thiết để đột nhập thành công vào thế giới này lẫn tìm được hứng thú nhằm trụ lại lâu dài. Nếu biết cách học đúng, lập trình sẽ là ngành cực kỳ thú vị và muôn màu đủ sắc khiến bất kỳ ai cũng có thể mê mẩn mãi không thôi. Vì thế mình cũng muốn đưa ra những chia sẻ và kinh nghiệm cá nhân này với hy vọng các bạn bắt đầu có ý định tìm hiểu sẽ đỡ tốn thời gian hơn, và tất nhiên sẽ có cơ hội tận hưởng cái đẹp của ngành này một cách toàn diện hơn nữa :)

Khởi động

Phần này rất đơn giản. Trước khi phấn khích nhảy vào cách học lập trình, hãy đặt những câu hỏi này cho bản thân và tự trả lời bằng cách viết xuống giấy/sticky note:

1. Vì sao bạn học lập trình? Mục đích và mục tiêu của bạn là gì?

2. Bạn sẵn sàng bỏ ra bao lâu để "phá băng" vào ngành này?

3. Cách bạn tự học thường như thế nào?

Và dán lên đâu đó dễ thấy, đại loại nó sẽ như một bảng chỉ đường soi sáng cuộc đời khi bạn lỡ lạc lối sau này.

Đồng thời ở câu 1:

- **Nếu bạn học vì muốn trở thành tỷ phú như Bill Gates hay nổi tiếng qua một đêm như Nguyễn Hà Đông với Flappy Bird?**

→ Hãy tìm một mục tiêu gần và thực tế hơn.

- **Nếu bạn học vì bố mẹ bảo?**

→ Xin lỗi, bạn sẽ không đua được với mấy đứa đam mê thật sự như tụi mình đâu. Hãy suy nghĩ kỹ nếu bạn đang học vì ai đó thay vì cho chính bản thân bạn.

- **Nếu bạn học vì đam mê?**

→ Hãy đặt một mục tiêu cụ thể hơn, vì đam mê xuất phát từ cảm xúc và nó sẽ không ổn định, vì thế bạn cần lý do vững hơn để trụ lại. Ví dụ: Phát triển cộng đồng công nghệ ở Việt Nam, phát triển ứng dụng giúp sinh viên nghèo kết nối với tri thức, có một công việc ổn định và được làm những gì mình thích,...

I. Chọn ngôn ngữ

Câu hỏi thần thánh *"Tôi nên học ngôn ngữ nào đầu tiên?"* luôn xếp đầu danh sách thắc mắc của các bạn newbie và có vô vàn video ngoài kia nói về chủ đề này. Đối với mình, ngôn ngữ lập trình được coi là công cụ và bạn nên quan tâm đến việc *"Học ngôn ngữ này sẽ giúp ích gì cho vấn đề mình đang cố giải quyết?"* hơn là *"Ngôn ngữ nào là tốt nhất để học?"*. Đơn giản vì học bất kỳ ngôn ngữ nào cũng giúp bạn hiểu những khái niệm căn bản trong lập trình. Không có ngôn ngữ nào là tốt nhất cả, nhưng chọn một ngôn ngữ phù hợp với nhu cầu của bản thân để bắt đầu sẽ giúp bạn tối ưu hoá thời gian đạt được mục tiêu mà bạn đã đặt ra ở phần khởi động phía trên.

Dưới đây là top 3 ngôn ngữ mà mình nghĩ là thích hợp với một người chưa biết gì về lập trình[1]:

- *Python (dễ nhất)*: Rất nhiều trường Đại học lớn trên thế giới đã chọn ngôn ngữ này để dạy những lớp lập trình căn bản cho sinh viên vì nó đơn giản, dễ học nhưng cũng rất đa năng. Bạn có thể dùng Python để phát triển back-end cho một website (Django, Flask), phát triển những ứng dụng về trí tuệ nhân tạo với TensorFlow, hoặc chuyên sâu về ngành Data Science (Phân tích Dữ liệu) cũng sẽ cần tới ngôn ngữ này.
- *Javascript (tầm trung)*: Đây là ngôn ngữ rất phổ biến với cộng đồng hỗ trợ cực mạnh. Với ngôn ngữ này, bạn có thể phát triển từ web (full-stack) cho đến app điện thoại. Javascript là ngôn ngữ phải học cho những ai đang hướng tới con đường lập trình full-stack web/mobile. Có rất nhiều nguồn học miễn phí và công cụ đã xây dựng sẵn để bạn sử dụng. Javascript có cả sự đơn giản của Python lẫn phức tạp của Java nên với những ai không thích cảm giác khó khăn khi chuyển từ ngôn ngữ đầu tiên sang ngôn ngữ thứ hai trong tương lai, Javascript là lựa chọn phù hợp.
- *Java (khó hơn)*: Dù tên như vậy nhưng ngôn ngữ này không có dây mơ rễ má với Javascript và hai cái hoàn toàn khác nhau. Đây là ngôn ngữ lâu đời có độ phức tạp cao hơn 2 ngôn ngữ trên và được dùng phổ biến nhất trong phát triển Android app, back-end cho một website hoặc những phần mềm hệ thống trong doanh nghiệp. Mình khuyên không đụng tới ngôn ngữ này đầu tiên TRỪ KHI bạn đang nhắm tới một nền tảng vững chắc trong ngành Computer Science như cấu trúc dữ liệu, thuật toán và khái niệm lập trình hướng đối tượng (Object-Oriented Programming) nhằm nâng cao tư duy lập trình trong thời gian sớm nhất. Trong trường hợp đấy, Java sẽ là một công cụ hoàn hảo để phát triển sớm tư duy logic và có hệ thống của bạn.

---

[1] **Một vài khái niệm cơ bản:**

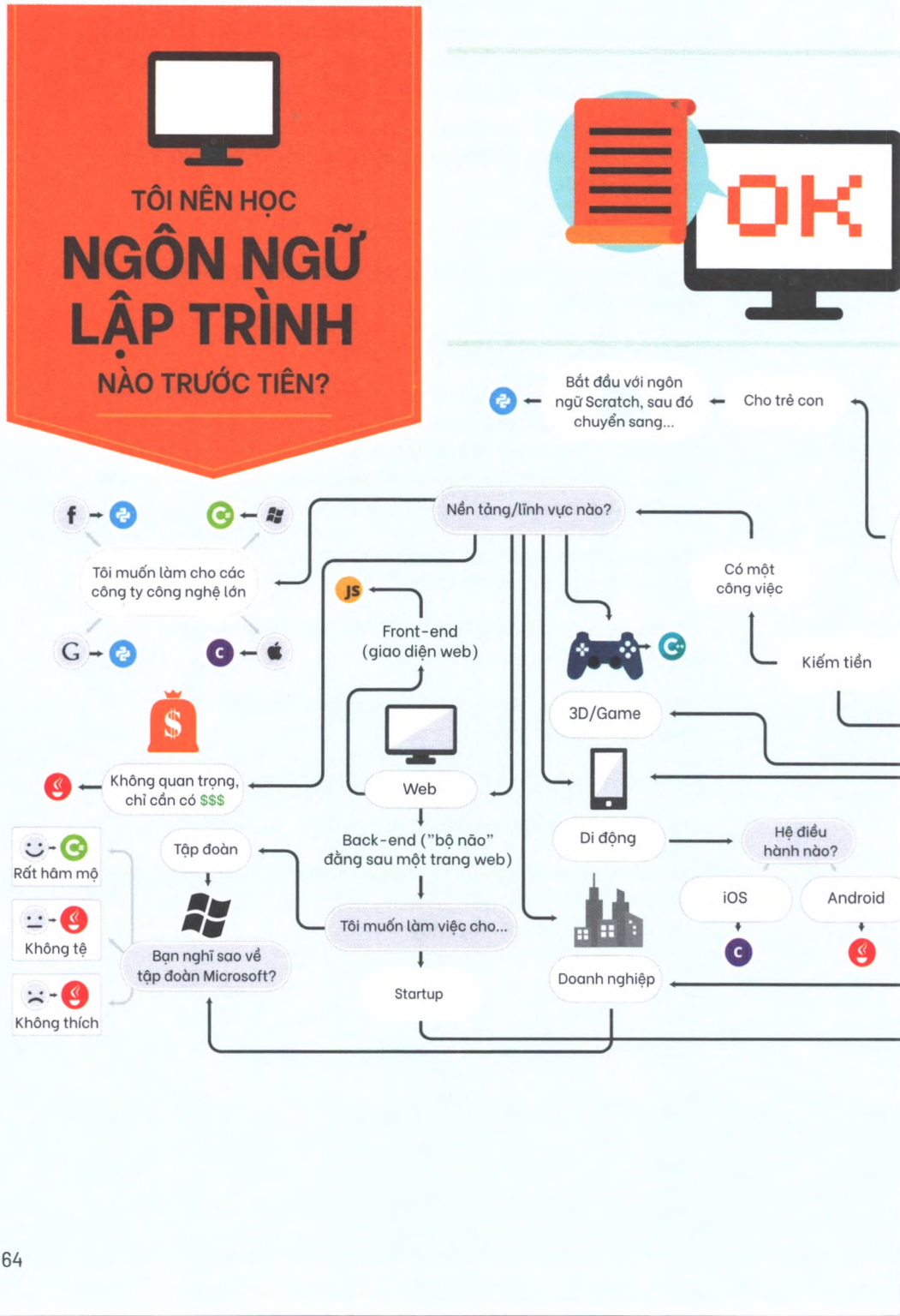
- **Front-end:** là giao diện mà người dùng thấy và tương tác, tất cả những gì bạn có thể thấy trên một trang web đều thuộc front-end, từ UI/UX đến việc xử lý và hiển thị dữ liệu từ back-end truyền tới.
- **Back-end:** là nơi lấy dữ liệu từ database (cơ sở dữ liệu) và trả về front-end, đây là phần thường được nằm ở máy chủ và không thể tương tác trực tiếp với người dùng.
- **Full-stack:** chỉ những lập trình viên phát triển cả front-end, back-end và cơ sở dữ liệu - có khả năng "bao" một trang web từ đầu đến cuối.
- **Frameworks:** những công cụ và môi trường được phát triển sẵn giúp bạn tiết kiệm thời gian trong việc đạt được một mục đích gì đó mà không phải xây lại từ đầu.

Hãy chọn 1 trong 3 ngôn ngữ này theo hướng bạn hứng thú (phát triển web, phân tích khoa học dữ liệu, học máy, khoa học máy tính,...) hoặc đơn giản hơn là theo... độ lì của bạn. Ví dụ, nếu bạn biết bản thân là người dễ chán và bỏ cuộc thì chọn anh chàng Python thân thiện sẽ phù hợp hơn cô nàng Java đỏng đảnh.

Còn nếu bạn đã biết rõ bản thân cần chuyên sâu theo hướng nào thì đây là danh sách những ngôn ngữ phổ biến được chia theo từng mảng chính:

- *Phát triển front-end web: Javascript, HTML & CSS*
- *Phát triển back-end web: Go, Scala, Python, Javascript, Ruby, hoặc PHP*
- *Thiết kế cơ sở dữ liệu (Database): SQL*
- *Phát triển ứng dụng điện thoại (Mobile development): Swift (iOS), Java (Android), Javascript (React Native), hoặc Dart (Flutter) - 2 cái sau đều là cross-platform framework, có nghĩa là chỉ phát triển một codebase nhưng dùng được cho cả iOS và Android.*
- *Ứng dụng trên hệ điều hành Windows (Windows development): C#*
- *Ứng dụng trên hệ điều hành iOS (iOS development): Swift hoặc Objective-C*
- *Lập trình nhúng (Operating/Embedded systems development) : C/C++*
- *Phát triển game (Game development): C++, Unity và C#*
- *Khoa học dữ liệu (Data Science): Python, R*

Đối với những bạn chưa biết bản thân muốn gì, cũng không thích 3 lựa chọn mình gợi ý ở trên và thích có hình minh hoạ hơn thì bạn có thể tham khảo sơ đồ dưới đây:

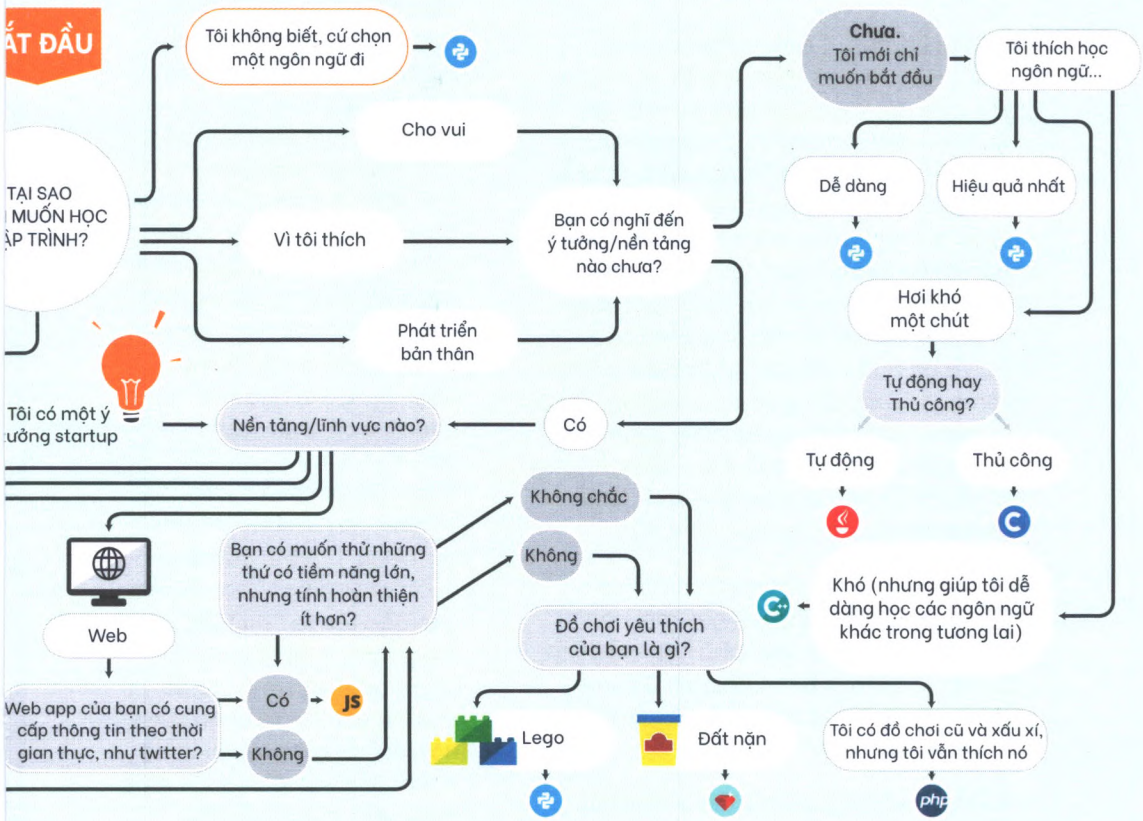

TÔI NÊN HỌC
NGÔN NGỮ LẬP TRÌNH
NÀO TRƯỚC TIÊN?
OK
Bắt đầu với ngôn ngữ Scratch, sau đó chuyển sang...
Cho trẻ con
Nền tảng/lĩnh vực nào?
Tôi muốn làm cho các công ty công nghệ lớn
Front-end (giao diện web)
Có một công việc
Kiếm tiền
3D/Game
Không quan trọng, chỉ cần có $$$
Web
Back-end ("bộ não" đằng sau một trang web)
Di động
Hệ điều hành nào?
iOS
Android
Rất hâm mộ
Không tệ
Không thích
Tập đoàn
Bạn nghĩ sao về tập đoàn Microsoft?
Tôi muốn làm việc cho...
Startup
Doanh nghiệp

Nguồn: http://carlcheo.com/startcoding – Việt hóa bởi **Spiderum**

CÁC NGÔN NGỮ LẬP TRÌNH

PYTHON · JAVA · C · PHP · C++

JAVASCRIPT · C# · RUBY · OBJECTIVE-C



Hoặc tham khảo những ngôn ngữ đang được yêu cầu cao nhất theo một thống kê năm 2019 ở trang web:
hired.com/state-of-software-engineers

Hoặc tham gia bài quiz của trang web này nếu bạn là người thích tương tác để hiểu bản thân hơn: *www.bestprogramminglanguagefor.me*

Đối với các bạn mới toanh trong ngành lập trình, để chọn được ngôn ngữ phù hợp nhất, các bạn đừng đọc xong những thứ này rồi lao vào tutorials ngay, mà hãy:

1. Chắt lọc ra 2 - 4 ngôn ngữ mà bạn hứng thú muốn thử từ những nguồn có sẵn trên mạng hoặc như mình đã gợi ý ở trên.

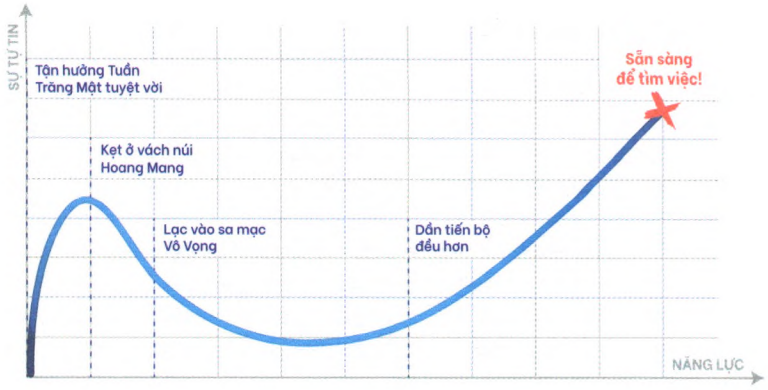
2. Dành ra khoảng 1 tuần để thoả sức nghiên cứu và chơi đùa với tất cả những ngôn ngữ đó. Việc này giúp bạn có cái nhìn toàn diện và phù hợp với bản thân hơn. Hơn nữa, khi tập trung vào học một ngôn ngữ, bạn sẽ giảm được tình trạng "đứng núi này trông núi nọ", thèm thuồng muốn nhảy sang ngôn ngữ khác trong tương lai. *Cốt lõi của lập trình là tư duy giải quyết vấn đề chứ không phải ngôn ngữ, và cả quá trình chỉ có thể được hoàn thiện qua hành động và kinh nghiệm theo thời gian.* Việc nhảy lung tung qua nhiều ngôn ngữ trong khi chưa nắm được những khái niệm cơ bản sẽ làm bạn mất thời gian và đôi khi cả hoang mang nữa.

3. Chọn ra ngôn ngữ mà bạn cảm thấy *tự nhiên nhất để bắt đầu.* Cái này mỗi người mỗi khác, ví dụ ngôn ngữ phù hợp với mình đầu tiên lại là Java do mình cảm thấy nó dễ hiểu và hệ thống chặt chẽ nên việc học cũng thoải mái hơn rất nhiều.

Lưu ý là trong thời gian thử nhiều thứ khác nhau thì bạn làm gì cũng được, nhưng một khi đã quyết định chọn một ngôn ngữ để tập trung học trong vòng 6 tháng - 1 năm tới thì hãy toàn tâm toàn ý vào em nó, đừng có đi tán tỉnh lung tung, ok?

II. Học chắc lý thuyết và làm các bài tập nhỏ

Phần này đòi hỏi sự kiên nhẫn cao vì 3 - 6 tháng đầu là khoảng thời gian khó khăn nhất. Không phải do lập trình căn bản khó, mà vì đây là lúc tâm lý người mới học muốn bỏ cuộc nhất. Có rất nhiều khái niệm mới, trừu tượng, nên đa số mọi người cảm thấy khó thở vì nó không đơn giản như tưởng tượng chút nào. Cộng thêm việc học theo hướng dẫn miễn phí trên mạng hay rơi vào cái bẫy phổ biến: Bạn làm theo như một cái máy, có thể mọi thứ hoạt động nhưng bạn không nắm rõ chuyện gì đang xảy ra. Bởi vậy, nhiều người khoe với mình đã làm được cả chục đồ án sau khi tự học một thời gian, nhưng tới khi mình hỏi đồ án này hoạt động thế nào hay dòng code kia có chức năng ra sao, họ không giải thích được. Để tránh lỗi "xây nhà từ nóc" này, mình đã cặm cụi trong 3 tháng đầu đọc cuốn sách giáo khoa về Java căn bản tên là *"Building Java Programs: A Back to Basics Approach"* mà anh chị đã cho. Người viết ra cuốn sách này là một giáo sư nổi tiếng ở Stanford với lối viết rất mộc mạc, gần gũi và dễ hiểu.



Những giai đoạn mà bất kỳ ai tự học lập trình đều phải trải qua

Từ đó, mỗi ngày sau khi học xong ở trường, mình lại hối hả chạy thẳng lên thư viện gần nhà dành ra tầm 6 - 7 tiếng học tiếp, vừa đọc vừa ghi chép, xong một chương thì làm bài tập trong sách rồi tự chấm điểm. Cách này chậm hơn một chút so với cách thông thường nhưng những kiến thức căn bản mình nắm được về ngôn ngữ lập trình nói chung và lập trình hướng đối tượng nói riêng thì vững và nhớ lâu hơn hẳn so với khi xem YouTube. Đến những chương lên cao, mình được học thêm về nhiều thuật toán và cấu trúc dữ liệu khác nhau cực kỳ thú vị. Thật bất ngờ khi càng học mình càng mê mẩn bởi cái hay của ngành lập trình, như những ứng dụng thực tế trong đời sống hằng ngày, tư duy logic để giải quyết vấn đề, cách áp dụng sự sáng tạo vào những thứ khô khan, hay đơn giản là ngồi cả ngày chơi trò trốn tìm với dấu chấm phẩy (phần này không vui mấy...)

Sau khi đã có nền tảng chắc chắn rồi, mình bắt đầu bung ra học những khoá MOOCS (Massive Open Online Course) nổi tiếng như Harvard CS50x - học về khoa học máy tính căn bản dùng C và Python, hay Columbia CSMM 101 - tìm hiểu về AI (trí tuệ nhân tạo) và Machine Learning (máy học). Trong thời gian rảnh, mình tìm đọc những cuốn sách không cần kiến thức chuyên môn giải thích về cách hoạt động của Google, những thuật toán nổi tiếng, dữ liệu lớn,... hoặc tùy hứng tham gia các hội thảo về phát triển game hay mobile app.

Có lẽ đáng nhớ nhất là khi mình hoàn thành chứng chỉ Software Development Micromaster của UBC trong vòng 2 tuần của kỳ nghỉ xuân. Động lực chính lúc đó chẳng qua do... thất tình. Mối tình đầu bất ngờ nói lời chia tay trong một ngày nắng đẹp làm cả thế giới nội tâm của mình như sụp đổ. Thế là 2 tuần sau đó, nếu đi qua thư viện thành phố, bạn sẽ chứng kiến cảnh tượng kinh dị có thật: một con bé nước mắt nước mũi chảy tèm lem, vừa sụt sùi khóc vừa cắm mặt vào laptop chép bài, trên mũi còn cắm nguyên hai cục khăn giấy to tướng chống nhỏ nước lên mặt giấy, đều đặn 12 - 15 tiếng mỗi ngày, 7 ngày mỗi tuần.

Qua trải nghiệm tự học và nhiều lần quan sát sau này, mình rút ra một vài bài học quan trọng mà quan sát thấy nhiều người mới bắt đầu mắc lỗi:

Lỗi #1 - Hổng kiến thức nền:

Rất nhiều người bỏ qua bước này và lao thẳng vào phát triển đồ án mà quên đi một thứ quan trọng là cần tập trung phát triển tư duy lập trình trong giai đoạn đầu này. Hãy kiên nhẫn học lý thuyết trước, kết quả hoàn toàn xứng đáng vì bạn sẽ tiến xa hơn rất nhiều với một nền tảng chắc chắn. Trong 3 - 6 tháng đầu, bạn cần nắm vững và hiểu rõ:

1) Syntax cơ bản và đặc tính của ngôn ngữ bạn chọn (Ngôn ngữ là OOP hay functional programming? Ưu và khuyết điểm là gì?,...)

2) Khái niệm lập trình cơ bản như: vòng lặp (loops), các loại biến (variable types), điều kiện (if-else, switch), các loại dữ liệu (int, string, double, float,...), đệ quy (recursion),...

3) Các cấu trúc dữ liệu cơ bản như Array, List, Map, Set, và Tree. Hiểu những đặc tính cơ bản của từng thứ và biết cách áp dụng cái nào trong những hoàn cảnh khác nhau.

4) Các thuật toán cơ bản như sorting và searching (có rất nhiều thuật toán khác nhau cho việc này), đồng thời nắm chắc những khái niệm như độ phức tạp không gian và thời gian (space and time complexity) và cả Big O notation.

Và vì kiến thức nền quan trọng, bạn nên tìm một nguồn có uy tín cao và đừng ngại bỏ ra một khoản tiền nhằm đảm bảo chất lượng kiến thức. Vì thế, mình xếp thứ tự ưu tiên các nguồn học như sau:

1. *Sách*: Bạn có thể mua trên Amazon hoặc tải PDF trên Github, khuyến khích nên đọc tiếng Anh từ đầu vì trong tiếng Anh, những khái niệm lập trình được giải thích dễ hiểu và tự nhiên hơn rất nhiều. Cái này Google phát là ra, ví dụ bạn cứ search từ khoá "*Best Books <ngôn ngữ lập trình muốn học>*", rồi sau khi đọc reviews các thứ từ Amazon, hãy search "*<ngôn ngữ lập trình> pdf github download*" (hay đại loại vậy) trước khi quyết định mua trên Amazon.

   Với Java thì mình thấy cuốn "*Building Java Programs: A Back to Basics Approach*" cực kỳ thích hợp với người mới. Với Javascript thì cuốn "*Eloquent JavaScript*" mình nghĩ khá ổn, có điều hơi khô khan nên các bạn cần tự tìm hiểu trước khi quyết định.

   Lưu ý: hãy né những cuốn có tựa đề như: "Học abc trong 24h", "Trở thành xyz chuyên nghiệp trong vòng 1 tuần",... vì đơn giản đó là cú lừa. Chẳng có đường tắt nào để học lập trình thành công cả, và con đường duy nhất bạn có thể đi là học, làm, thất bại, và lặp đi lặp lại để tăng kinh nghiệm theo thời gian. Việc học nên hướng tới về lâu dài theo năm chứ đừng mang tư duy học trong 1 ngày, vài tuần là được.

2. *MOOCs của các trường đại học lớn (EdX, Coursera)*

3. *Các nền tảng code trực tuyến (Codecademy, FreeCodeCamp)*. Mình đã thử mua vài khoá trên Udemy nhưng thấy nó không hữu dụng lắm nên hơi hối hận, nhưng bạn có thể thử vì biết đâu lại hợp.

4. *Blogs (Medium, FreeCodeCamp)*

5. *Tutorial videos (YouTube)*: Cá nhân mình nghĩ không nên bắt đầu từ đây vì YouTube là nơi tưởng chừng như chất lượng nhưng cũng rất bát nháo, dễ bị sao lãng và tiếp thu nhiều thông tin trái chiều khi học trên đây - điều tối kỵ đối với những người mới bắt đầu.

Lỗi #2 - Quá hấp tấp:

Theo kinh nghiệm của mình, những bạn càng vội thì càng dễ bỏ cuộc ở thời gian này. Với những người mới bắt đầu, hãy nhắm tới một kế hoạch học lâu dài và bền bỉ bằng cách chậm chậm tăng đà, không nên hành động ngẫu hứng. Ví dụ khi mới đọc sách lập trình, thay vì quyết tâm dành 3 tiếng liền trong ngày đầu tiên, bạn hãy chia ra một ngày dành ra tầm 5/10/20/30 phút để đọc và cứ đều như thế, tới khi tự tin hoặc muốn đọc nhanh hơn thì tăng thời gian.

Hãy bắt đầu nhỏ thôi.

Tương tự với những dự án, bạn nên làm các bài tập nhỏ thường được cho trong sách và từ từ tăng dần độ khó thay vì ảo tưởng "Tôi sẽ xây dựng một đế chế Facebook thứ hai!". Đôi lúc, nếu cảm thấy muốn bỏ cuộc, hãy tìm bạn học chung để tăng động lực. Đồng thời, việc trao đổi/giải thích kiến thức với bạn bè sẽ giúp bạn hiểu sâu hơn lý thuyết đã học. Hoặc khi bị mắc kẹt ở một khâu nào đó, bạn hãy lặp lại câu thần chú theo mình: *"Google là thầy, Stack Overflow là bạn"* và bình tĩnh giải quyết, vì 99% vấn đề bạn gặp phải đã có ai đó gặp phải trước đó và đưa đáp án lên mạng rồi. Kỹ năng search Google cho vấn đề bạn gặp phải là một kỹ năng sống còn trong ngành lập trình, và bạn có thể search *"<ngôn ngữ lập trình> <lỗi gặp phải> stackoverflow"* để thiết lập điểm bắt đầu. Nhưng hãy cố gắng hiểu trước đoạn code mà bạn sắp copy vô tội vạ để nắm sâu hơn về vấn đề bạn đang giải quyết.

Lỗi #3 - Lười:

Rất nhiều người chỉ thụ động ngồi xem và đọc tài liệu rồi nghĩ hiểu rõ khái niệm/kiến thức rồi. Điều này rất dễ xảy ra khi một khái niệm trong lập trình có vẻ dễ hiểu trên lý thuyết nhưng lại rất khó để thành thạo trong thực tế. Cách tốt nhất để tránh trường hợp này là bạn tích cực làm, làm và làm thật nhiều bài tập được đưa ra, luôn suy nghĩ làm cách nào để viết code tốt hơn. Như Khổng Tử đã nói: *"I hear and I forget. I see and I remember. I do and I understand."*

Cũng có nhiều người cho rằng xong một nguồn học rồi thì mình đã học xong mọi thứ và không cần phải học gì nữa. Sai nhé! Có người trong ngành này 5 - 10 năm rồi nhưng luôn luôn có cái mới để học. Bạn đừng rơi vào bẫy tự mãn này và nên giữ cái tâm tò mò muốn học hỏi điều mới. Đây là điểm khác biệt lớn giữa một một người lập trình viên tốt và một người lập trình viên tồi về lâu dài.

III. Tăng kinh nghiệm bằng các dự án thực tế

Tự học một thời gian, mình dần lấy được sự tự tin từ nền tảng kiến thức vững chắc qua sách vở và những khoá học trên mạng, từ đó lấy động lực tham gia đủ cuộc thi lớn bé trong thành phố.

Hackathon là tên gọi chung những cuộc thi kéo dài 24/36/48 giờ liên tục dành cho phát triển ý tưởng công nghệ. Cơ bản là bạn sẽ phải mang theo laptop, gối, chăn, bàn chải đánh răng và những vật dụng cá nhân cần thiết để cắm sân qua đêm. Thể lệ cuộc thi thường khá đơn giản: bạn lập một đội từ 3 - 5 người, phát triển từ một ý tưởng sơ khai (nghiêm cấm bất kỳ hình thức chuẩn bị nào trước) thành một sản phẩm hoàn thiện, và cuối cuộc thi thuyết trình với ban giám khảo.

Có 2 loại hackathon:

- Kỹ thuật (technical hackathon): thường dành cho lập trình viên và dân thiết kế. Người dự thi có thể phát triển bất kỳ thứ gì liên quan đến công nghệ. Phần mềm thì có app, web, game,... Phần cứng thì có robots, xe mini,... Giới hạn là bất cứ thứ gì điên rồ mà bạn có thể nghĩ ra.

- Không kỹ thuật (non-technical hackathon/case competition): dành cho tất cả ngành nghề. Cuộc thi sẽ đưa ra một đề tài hoặc vấn đề bất kỳ trên toàn cầu, trong thời gian cho phép bạn phải phát triển một mô hình và kế hoạch kinh doanh lẫn thiết kế hoàn chỉnh cho giải pháp của bạn.

Trong 2 năm mình đã cố gắng đi cả hai vì mỗi loại có cái khó và hay riêng của nó. Nhớ lần đầu tiên tham dự một technical hackathon, mình lo lắng đến mức lạc cả giọng, dù trước đó đã cố gắng chuẩn bị tâm lý. Lóng ngóng bước vào căn phòng toàn người lớn mà ai nhìn cũng giỏi giang làm mình cảm thấy thật nhỏ bé và vô dụng, dường như mớ kiến thực học được bốc hơi hết. Một khắc trong đầu còn lóe qua suy nghĩ: "hay là trốn về nhỉ?", nhưng thật may là mình đã tự tát bản thân một phát cho tỉnh, lấy hết can đảm tiến về phía một nhóm người lạ làm quen. Khi tới lượt mình giới thiệu, mọi người đều ngỡ ngàng kiểu: "Trời, mới lớp 11 dám lấy gì mà thi thố?!", rồi 10 phút sau cả phòng quay quanh mình hỏi đủ thứ chuyện. Cả tối hôm đó rất vui khi lần đầu tiên mình được trải nghiệm cảm giác thức trắng đêm xây dựng sản phẩm cùng mọi người, tranh luận xôn xao một góc phòng dù những đội khác đã mệt quá và lăn ra ngủ gần hết. Thừa thắng xông lên, kể từ đó đến nay mình đã tham dự 19 cái hackathon, thắng thua có đủ. Nhưng quan trọng hơn, mình được gặp gỡ rất nhiều người có kinh nghiệm trong ngành chịu khó ngồi xuống và kiên nhẫn giảng giải cho mình nghe những thứ họ biết và trải qua. Vì được mở mang tầm mắt mỗi lần đi như thế, kỹ năng lập trình của mình đã tiến bộ hơn rõ rệt trong một thời gian ngắn.

Giải nhất đầu tiên :)

Một trong những hackathon để lại kỷ niệm sâu sắc nhất với mình là cuộc thi Hack The North vào tháng 9 vừa qua. Đây là hackathon lớn nhất Châu Mỹ với hơn 1,000 người dự thi đến từ khắp nơi trên thế giới được tổ chức ở tận bên bờ Đông của Canada. Sau khi mất cả tháng để thuyết phục gia đình, mình cuối cùng được tự thân xách ba lô bay qua Waterloo dự thi. Lần đầu tiên đi du lịch một mình, lần đầu tiên đến thành phố lạ, hay lần đầu tiên được tiếp xúc với những sinh viên tài năng đến từ Stanford, Harvard, Oxford đã làm mình choáng ngợp bởi những trải nghiệm quá sức mới mẻ và thú vị. Đây là một hackathon hoành tráng nên rất nhiều công ty công nghệ lớn đổ về như Google, Deloitte, Facebook,... Vì thế, mình được dịp thoả thích hỏi họ những điều vẫn luôn tò mò. Ấn tượng nhất là cuộc thi cung cấp mọi thứ rất đầy đủ: đồ ăn 3 bữa mỗi ngày, mỗi bữa có những món từ nhiều văn hoá khác nhau và siêu chất lượng, ngoài ra có đồ ăn phụ như há cảo, khoai tây chiên, trà sữa giữa những buổi ăn chính, và còn cung cấp luôn phòng nghỉ cùng túi ngủ ấm áp. Tại đây, đủ loại hoạt động được diễn ra giúp thí sinh xả stress: từ bắn cung, chiến súng nước, thi xếp ly, đến những thứ nhẹ nhàng hơn như karaoke, chơi với chó. Cuộc thi cũng tổ chức rất nhiều buổi hội thảo cho những ai muốn học thêm kiến thức mới và kỹ năng cần thiết cho đồ án của họ, còn có cả mentors từ nhiều công ty khác nhau để giúp đỡ 1 - 1 nếu đội nào cần.

Hoạt động ăn chơi thì nhiều nhưng đây lại là lần mình tập trung nhất - lần đầu tiên tự phát triển một phần mềm khá hoàn chỉnh mà không cần đến sự trợ giúp bên ngoài. Đội ban đầu chỉ có 3 người, mình và một cặp đôi đang học cử nhân ngành Computer Science đến từ Bỉ. Ý tưởng dự thi của mình là một ứng dụng web nơi người dùng có thể tô màu những bức tranh vẽ sống động bằng cách hát - tiếng hát càng trầm thì màu càng đậm, giọng lên cao thì màu nhạt dần.

Mình hứng khởi lao vào phát triển đồ án và khi bắt đầu thì một đống vấn đề nảy sinh. Vì ý tưởng này chưa từng được nghĩ ra trước đó nên không có đồ án liên quan nào trên mạng có thể đem ra nghiên cứu, kết quả là mình phải tự lực cánh sinh, viết phần mềm lại từ đầu. Một khó khăn nữa mình phải đối mặt là hai anh chị vì bị lệch múi giờ nên đã bỏ cuộc giữa chừng, vì thế mình phải gánh hết vai trò cho đến lúc hoàn thành sản phẩm. Ngồi hơn 12 tiếng để tính toán những dữ liệu cần thiết cho mối quan hệ giữa ánh sáng và âm thanh từ tài liệu những năm 90s, mày mò tự học công nghệ mới để giúp hoạt động những thuật toán mình tự thiết kế, đụng phải lỗi, sửa lỗi, rồi vừa làm vừa học trong 36 giờ không ngủ. Cảnh tượng khá buồn cười khi giữa một căn phòng

ngổn ngang máy tính và tiếng ngáy, có một đứa nhóc đang loay hoay hát thử vào cái microphone vào 4 giờ sáng. Chủ nhật hôm sau, hạn nộp bài là 9:00AM, mình mắt nhắm mắt mở nộp lúc 8:59 rồi gục xuống bàn bất tỉnh luôn (đùa đấy, chỉ là mệt quá nên ngủ thôi).

Qua chuyến đi này, mình học được cực kỳ nhiều điều mới và kết được rất nhiều người bạn dễ thương xuyên lục địa. Và có thể nói, đây là một trong những project mình tự hào nhất đến nay, và cũng nhận ra được chân lý quý giá: giai đoạn lập đội rất quan trọng.

Nhưng...!

Sau khi đi quá nhiều hackathon, mình nhận ra bài học lớn: chỉ nên đi dưới 10 cái để trải nghiệm và mở rộng quan hệ chứ không nên đi nhiều hơn. Hackathon là nơi các công ty thường đến để thuê tuyển học sinh/sinh viên có tiềm năng, song đồng thời cũng là nơi họ "cướp" những ý tưởng hay từ các dự án đứng nhất để đem về phát triển ở công ty họ, nên nếu bạn có ý tưởng startup nào hấp dẫn, hãy nung nấu và tự làm chứ đừng hớ hênh đem khoe ở hackathon. Thêm một điểm nữa là những dự án phát triển ở hackathon, dù giúp bạn học được nhiều thứ trong thời gian ngắn và phát triển kỹ năng làm việc nhóm, đều thuộc những dự án ngắn hạn và thường bị ngưng lại một khi hackathon kết thúc.

Khi đã trải nghiệm đủ, mình nghĩ đây là thời gian thích hợp để bạn tách ra và bắt đầu phát triển dự án cá nhân hoặc với một nhóm bạn lập trình thân thiết. Có thể bạn có ý tưởng triệu đô và thành lập startup, có thể đó là một vấn đề bạn đang gặp phải và muốn tự giải quyết, có thể bạn muốn clone lại game ưa thích hồi bé của bạn,... Có vô vàn ý tưởng ngoài kia để bạn bắt đầu làm. Đây cũng là giai đoạn thích hợp để bạn bắt đầu tìm hiểu về những phương pháp thiết kế và phát triển phần mềm phổ biến ngoài kia như Scrum/Agile Development, Waterfall development, Test-Driven Development, Extreme Programming,... để phục vụ và chọn xem hướng nào phù hợp với tính cách của bạn và đặc trưng dự án.

Ngoài ra có một quyển mà mình lúc nào cũng giới thiệu mọi người đọc là *Clean Code* của Uncle Bob, huyền thoại vàng trong làng lập trình. Đây là cuốn sách đảm bảo sẽ khai sáng tư duy của bạn về lập trình với văn phong đơn giản, dễ đọc, nhưng cực kỳ chất lượng, nói về cách viết code sao cho sạch, đẹp và dễ quản lý.

Về phần mình, sau khi tham dự hackathon, mình cùng một nhóm bạn xung phong lên nói chuyện với hiệu trưởng để giúp trường phát triển một phần mềm phục vụ thầy cô và nhân viên sắp xếp lịch dạy tự động bằng Python. Ngoài ra, mình cũng giúp trường phát triển một hệ thống song song dành cho giáo viên và học sinh tự quản lý thời gian biểu và nhận thông báo từ nhà trường mà mình tự làm trong 2 tuần của kỳ nghỉ xuân. Những hệ thống mà tụi mình phát triển tất nhiên là không hoàn hảo và còn nhiều thiếu sót, nhưng quan trọng là mình đã đẩy bản thân vào thế khó và từ đó học được rất nhiều từ những dự án thành công lẫn thất bại.

Kết

Tổng hợp lại, có 4 giai đoạn bạn sẽ đi qua khi tự học lập trình:

0. Xác định rõ mục đích và mục tiêu học

1. Chọn ngôn ngữ phù hợp với mục đích/nhu cầu/định hướng

2. Lấy kiến thức nền từ lý thuyết

3. Lấy kinh nghiệm thực tế từ chiến trường (hackathon & dự án)

Mình hy vọng những chia sẻ từ trải nghiệm cá nhân sẽ giúp ích được phần nào cho các bạn mới bắt đầu. Mình đã từng ở giai đoạn đó, mình hiểu những gì đã viết ở trên đều mang nặng nhân sinh quan của mình. Mỗi người sẽ có con đường rất khác nhau. Vì thế, bạn nên là người tự chắt lọc những thứ phù hợp nhất và áp dụng vào bản thân.

Và cuối cùng, như một món quà chúc may mắn, mình xin chia sẻ vài nguồn học chất lượng được đóng góp bởi các tiền bối trên Spiderum và đã được mình tổng hợp lại ở dưới.

Chúc may mắn!

Sách/Khóa học online:

- **Michael 0x2a Github** - Tổng hợp rất nhiều nguồn cho nhiều ngôn ngữ lập trình khác nhau
- **tiago-atha's Github** - Tổng hợp một kho sách miễn phí cho lập trình
- **FreeCodeCamp** - Nhiều khoá học uy tín và có một cộng đồng lớn để luôn hỗ trợ khi bạn cần. Trên đây cũng có khá nhiều bài viết hướng dẫn rất hữu ích cho người mới.
- **Codecademy** - Giao diện đẹp để học và tương tác với code nhưng tập trung khá nhiều về mảng lý thuyết
- **Teach Yourself CS** - Tổng hợp và gợi ý nguồn sách cho những khái niệm cốt lõi của Computer Science
- **Open Source Society University** - Tổng hợp nhiều nguồn học trong một giáo trình đại học online mà bạn có thể tự nghiên cứu ở nhà
- **Berkeley CS61A & CS61B** - 2 khoá Computer Science căn bản đến từ trường đại học nổi tiếng Berkeley. Cả 2 trang web của khoá học được thiết kế rất rõ ràng và dễ theo dõi.
- **GeeksForGeeks** - trang này nổi từ đời đầu 2010 ~ 2013, trước đây UI không đẹp nhưng mới đây đã nâng cấp phần giao diện trông ổn hơn rất nhiều. Điểm cộng: cách dàn content phân chia level rõ ràng thành Easy → Medium → Hard.
- **TutorialsPoint** - cũng vừa được tút lại giao diện. Một trong các nguồn có lộ trình phân chia rạch ròi vẫn còn tồn tại, và đồng thời đã cải tiến khá nhiều phần Playground để vọc vạch thực hành.
- **Learn-Anything** - sử dụng logic của Alfred Workflow để hiện gợi ý cho từ khoá người dùng nhập. Góc phải của kết quả tìm kiếm có dấu (-) và (+) là rate review của người dùng để đánh giá kết quả nguồn sau khi học giúp người sau cân nhắc, đỡ tốn thời gian hơn.

Cộng đồng để hỏi khi bí

Trong quá trình học, bạn nên tích cực tra khảo trên mạng và hỏi những người đi trước trên những cộng đồng lớn.

- Stack OverFlow
- Reddit/Quora

Chuẩn bị cho phỏng vấn xin việc

- HackerRank
- LeetCode
- TopCoder

Lập Trình Web

"sân chơi" còn nhiều "đất diễn"

Tác giả: **Nguyễn Nhật Hoàng - Codeaholicguy**
Engineering Manager ShopBack

Theo xu hướng hiện nay, nhiều bạn lập trình viên chọn con đường Web Development để phát triển sự nghiệp. Trong thời đại công nghệ, đại đa số các doanh nghiệp đều có mong muốn phát triển Web, dẫn đến nhu cầu lập trình viên Web luôn luôn nằm ở mức cao trong nhiều năm vừa rồi. Qua bài viết này, mình hy vọng các bạn sẽ có cái nhìn rõ ràng về ngành lập trình Web, cũng như chuẩn bị những hành trang cơ bản để phát triển bản thân trong lĩnh vực thú vị này.

Tổng quan về Web Development

Cùng sự bùng nổ của Internet, Web trở thành "gã khổng lồ" trong thế giới phần mềm. Dù bạn có muốn trở thành một Web Developer hay không, ít nhất bạn cũng nên biết một chút về phát triển Web, Web hoạt động thế nào, bởi "gã khổng lồ" này sẽ đi theo bạn trong suốt quãng đường nghề nghiệp liên quan đến phần mềm. Tuy cách thức phát triển đã thay đổi rất nhiều từ quá khứ đến hiện tại, nhưng mình có thể tóm tắt Web Development trong một câu: "Web Development là quá trình phát triển ứng dụng Web chạy trên nền tảng các trình duyệt Web".

Thời điểm hiện tại, Web Development là thế giới rộng lớn mà nếu muốn chinh phục nó, bạn sẽ phải có kiến thức về Web Design (Thiết kế Web), Networking (Mạng máy tính), Security (Bảo mật), Database (Cơ sở dữ liệu), Distributed System (Hệ thống phân tán), Cloud Computing (Điện toán đám mây) và hàng tá thứ khác nữa.

Khi trở thành một Web Developer, bạn sẽ phải giao tiếp với bộ phận chuyên phụ trách về sản phẩm để hiểu rõ yêu cầu và tìm ra cách tốt nhất giúp ứng dụng Web thu hút người dùng, cũng như giao tiếp với khách hàng để thỏa mãn yêu cầu của họ. Bên cạnh đó, bạn cũng sẽ làm việc với người quản lý dự án, người thiết kế, để phát triển sản phẩm tốt nhất với ngân sách và quỹ thời gian được phân bổ.

Ngoài ra, khi trang Web được đưa ra thị trường, bạn sẽ phải bảo trì nó cùng với dòng đời của sản phẩm, đồng thời đưa ra những chức năng mới để thu hút khách hàng và theo kịp những thay đổi về công nghệ.

Vì thế, để trở thành Web Developer, ngoài những kỹ năng về công nghệ, bạn cần có khả năng làm việc nhóm tốt, bởi làm việc tốt với những người khác là yếu tố quan trọng giúp cho dự án thành công. Khả năng giải quyết vấn đề cũng như đưa ra các giải pháp sáng tạo cũng là một yếu tố mà bạn cần có để phát triển tốt hơn trong sự nghiệp của mình.

Những vị trí trong ngành Web Development

Câu hỏi "Web Developer làm những công việc gì?" thực sự rất khó để trả lời. Cùng với sự phát triển của Web Development, mỗi Web Developer sẽ tập trung vào một khía cạnh khác nhau trong việc tạo ra một ứng dụng Web. Trong thế giới phát triển Web, sẽ có 3 hướng chính bạn có thể tập trung, đó là Front-end, Back-end, và Full-Stack.

Front-end Developer

Front-end Developer là người phát triển bộ mặt của ứng dụng Web: chuyển những thiết kế nhận được từ khách hàng hay nhóm thiết kế thành ứng dụng Web, viết code để hiện thực sản phẩm từ thiết kế. Một Front-end Developer sẽ phải thuần thục ít nhất 3 ngôn ngữ lập trình: HTML, CSS và JavaScript.

Đối với Front-end Developer, họ chịu trách nhiệm đảm bảo tất cả nội dung cần thiết cho trang Web được hiển thị rõ ràng và nằm ở đúng vị trí, có màu sắc phù hợp, các liên kết đi đến đúng nơi và thao tác của người dùng phải hoạt động chính xác.

Back-end Developer

Trong khi Front-end phát triển bộ mặt của ứng dụng Web thì Back-end chính là hệ thống xương sống giúp cho ứng dụng Web có thể vận hành trơn tru.

Back-end Developer sẽ phải viết code để phía máy chủ xử lý các yêu cầu của người dùng, thao tác với cơ sở dữ liệu, và tương tác với những hệ thống khác. Điều quan trọng nhất đối với Back-end Developer là tạo ra những đoạn code hiệu quả, an toàn, và xử lý nhanh bởi tốc độ của trang Web là một yếu tố quan trọng giúp tối ưu trải nghiệm của người dùng và cả các công cụ tìm kiếm như Google Search.

Có nhiều ngôn ngữ lập trình phổ biến thường được sử dụng ở Back-end, có thể kể đến như Java, C#, PHP, Python, Ruby, JavaScript và nhiều ngôn ngữ khác nữa mà mình không thể liệt kê hết. Mỗi Back-end Developer sẽ phải hiểu biết rõ ràng và sâu sắc về ngôn ngữ mà họ sử dụng. Điều này cực kì quan trọng, vì nó giúp họ đưa ra những giải pháp hiệu quả nhất giúp ứng dụng Web dễ dàng bảo trì và mở rộng.

Full-stack Developer

Hướng phát triển Full-stack là hướng phát triển mất nhiều thời gian của các bạn nhất. Nói một cách đơn giản, Full-stack Developer là người có thể làm được cả Front-end và Back-end. Những gì Full-stack Developer làm cũng là câu trả lời gần nhất cho câu hỏi "Web Developer làm những công việc gì?".

Full-stack Developer có thể xây dựng một ứng dụng Web hoàn chỉnh, điều này cực kì phù hợp với những công ty nhỏ, bị giới hạn về mặt chi phí. Họ cũng là người có nhiều kinh nghiệm, hiểu biết sâu sắc về cả Front-end lẫn Back-end để đưa ra giải pháp phù hợp.

Hầu hết Full-stack Developer thường bắt đầu ở một mảng Back-end hay Front-end, khi đã đủ hiểu biết về mảng họ đang phát triển, họ sẽ cố gắng lấn sang mảng còn lại. Một Full-stack Developer có thể là chuyên gia trong một số thành phần nào đó, nhưng để là chuyên gia trong tất cả thì chắc chắn bạn phải có rất nhiều kinh nghiệm. Việc trở thành Full-stack Developer giúp bạn hiểu rõ hơn về quy trình phát triển ứng dụng Web, cũng như mang đến nhiều cơ hội việc làm hơn.

Làm thế nào để trở thành Web Developer?

Một sự nhầm lẫn thường thấy là khi suy nghĩ về việc trở thành Web Developer, bạn dễ bị nhầm lẫn với định hướng phát triển về Web Designer. Mặc dù việc thiết kế và phát triển Web có thể là cùng một người nhưng nếu định hướng phát triển là Web Developer, bạn sẽ phải tập trung nhiều hơn vào việc viết code, trong khi Web

Designer là người tập trung nhiều về khía cạnh thiết kế, bố cục, màu sắc, thông thường họ không cần phải biết/viết code.

Nếu nghiêm túc về việc phát triển sự nghiệp Web Developer, bạn cần chắc chắn rằng mình đã đặt mục tiêu rõ ràng cho bản thân. Hãy bắt đầu bằng việc lựa chọn định hướng phát triển là Front-end Developer hay Back-end Developer.

Tại sao không lựa chọn trở thành Full-stack Developer? Như đã nói bên trên, Full-stack cần nhiều kinh nghiệm và bạn phải có một mảng thế mạnh của mình trước khi bắt đầu sự nghiệp theo hướng này.

Sau khi lựa chọn được định hướng phát triển, bạn sẽ tìm được lộ trình học tương ứng.

Nếu định hướng theo Front-end, bạn chắc chắn phải học HTML, CSS và JavaScript, bởi đây là những ngôn ngữ bắt buộc phải biết. Sau khi có kiến thức tốt về những ngôn ngữ trên, bạn có thể thoải mái lựa chọn những library hoặc framework để học như AngularJS, ReactJS hay Vue.js.

Nếu định hướng theo Back-end, đầu tiên bạn phải chọn một trong số những ngôn ngữ phổ biến mình đã đề cập ở trên như: Java, C#, PHP, JavaScript, Python, Ruby,... Ngoài ra, trong quá trình học, bạn cũng nên bổ sung kiến thức về cơ sở dữ liệu, học SQL để biết cách thao tác với chúng. Sau khi có kiến thức tốt về cơ sở dữ liệu và ngôn ngữ bạn chọn, hãy tìm framework nổi trội ở ngôn ngữ đó để tiếp tục học. Mỗi ngôn ngữ sẽ có một framework nổi trội nhất, ví dụ Java có Spring, C# có ASP.Net, PHP có Laravel, Ruby có Ruby on Rails,...

Quá trình này chắc chắn bạn mất rất nhiều thời gian, cũng như bỏ công sức cho việc học tập. Nhưng một khi đã hiểu được phát triển Web là gì và có tầm nhìn rõ ràng về những thứ bạn muốn đạt được, mọi việc sẽ trở nên dễ dàng hơn nhiều.

Dưới đây là một số ngôn ngữ lập trình phổ biến để phát triển Web:

- *HTML*: ngôn ngữ giúp tạo nên bộ khung của trang Web, cũng là ngôn ngữ mà bất cứ lập trình viên Front-end nào cũng cần phải biết.
- *CSS*: Nếu HTML là bộ khung của trang Web thì CSS chính là phần da thịt đẹp đẽ được đắp lên bộ khung đó.
- *JavaScript*: ngôn ngữ được sử dụng để hiện thực những tương tác trên trang Web, ví dụ khi người dùng ấn vào một nút trên Web thì JavaScript sẽ giúp Front-end Developer xử lý các tương tác đó.
- *PHP*: ngôn ngữ dùng ở Back-end, hơn 80% các trang Web đang tồn tại đều được phát triển bằng PHP. Mặc dù đang giảm dần về tính phổ biến, PHP vẫn là ngôn ngữ tốt ở phía Back-end mà bạn nên cân nhắc học.
- *Java*: ngôn ngữ dùng ở Back-end, phần lớn các Back-end Developer đều bắt đầu sự nghiệp của mình với Java. Trong nhiều năm liên tiếp, Java cũng là ngôn ngữ phổ biến nhất thế giới.
- *SQL*: ngôn ngữ dùng ở Back-end, giúp bạn truy vấn cơ sở dữ liệu, đây là ngôn ngữ mà bất cứ Back-end Developer nào cũng cần phải biết.

Ngoài ra, còn rất nhiều ngôn ngữ khác, mình chỉ đề cập những ngôn ngữ mà mình cho là nổi trội nhất trong ngành Web Development.

Hành trình trở thành một Full-stack Development

Đối với cá nhân mình, lập trình là niềm đam mê từ thuở còn đi học. Từ những năm trung học, mình được tiếp xúc nhiều với lập trình thông qua ngôn ngữ Pascal. Khi nhận ra những ứng dụng phức tạp đều được tạo ra từ những dòng lệnh đơn giản, mình bắt đầu yêu thích công việc này.

Thời điểm bắt đầu yêu thích lập trình, mình làm phát triển game. Mình mày mò và tìm tòi thông qua các diễn đàn trên mạng rồi tự làm game. Năm lớp 11, mình đưa ra cộng đồng sản phẩm game nho nhỏ của mình và nhận được không ít sự ủng hộ.

Mình học kỹ sư phần mềm tại Đại học FPT. Trong quá trình đi học, mình nhận ra việc phát triển ứng dụng Web sẽ giúp ích được cho nhiều người hơn làm Game. Những ứng dụng Web của các công ty lớn như Google, Facebook,... phục vụ cho hàng triệu, hàng tỷ người dùng mỗi ngày. Vì thế, mình quyết định chuyển sang lập trình Web. Quãng thời gian sinh viên, mình có đi làm part-time cho một công ty quảng cáo (agency) do được một người anh giới thiệu, công việc lúc đó xoay quanh việc cắt HTML, rồi làm CSS sao cho đúng là được. Nhờ vậy mà mình thành thạo HTML/CSS hơn, vì kiến thức học trong trường chỉ có Core Java, C#, .NET và nhiều lý thuyết.

Mình bắt đầu công việc toàn thời gian đầu tiên với vị trí Back-end Developer. Lúc ấy, môi trường làm việc thực tế khiến mình bị choáng ngợp bởi quá nhiều thứ để học mà trong trường không hề dạy. Mình phải tự học khá nhiều thứ như Spring, Hibernate, RESTful API,... Nhờ may mắn, mình gặp được những người đồng nghiệp tốt, họ chỉ dẫn cũng như giúp đỡ mình rất nhiều.

Một dấu mốc đáng nhớ là năm 2016, mình tham gia Facebook Hackathon[1] Vietnam. Trong vòng 24h, mình cùng team phát triển một ứng dụng Web, sử dụng nền tảng của Facebook để giải quyết việc gửi đánh giá cho người dùng. Team mình thắng giải sản phẩm tốt nhất trong hạng mục Marketing, giải thưởng là một gói hỗ trợ FbStart của Facebook trị giá 80.000 USD.

Sau hơn 3 năm làm Back-end Developer, mình học được rất nhiều kiến thức về Back-end, System Architecture (Kiến trúc hệ thống), Cloud Computing (Điện toán đám mây). Mình cũng hiểu về quy trình sản xuất phần mềm dưới cương vị là Back-end Developer như thế nào. Lúc này, mình quyết định phát triển bản thân theo hướng Full-stack, bắt đầu tự học thêm về Back-end sử dụng NodeJS, rồi học Front-end với ReactJS và chuyển sang vị trí mới tại một công ty phát triển sản phẩm. Trong suốt quá trình làm việc ở vị trí Full-stack, mình học được rất nhiều về sự đánh đổi giữa công nghệ và sự tiện lợi cho người dùng, cũng như cách thức làm việc giữa các thành phần của hệ thống với nhau để tạo nên một sản phẩm tốt. Đó cũng là những kiến thức giúp mình phát triển thành một Quản lý (Engineering Manager) ở thời điểm hiện tại.

Việc tập trung làm Back-end ở thời điểm mới bắt đầu đi làm giúp cho mình có được nền tảng vững chắc để phát triển bản thân theo hướng Full-stack. Thay vì ngay lập tức lao vào học nhiều thứ để trở thành Full-stack Developer, việc tập trung một mảng nhất định sẽ giúp bạn có nền tảng vững chắc để thực hiện những bước tiếp theo trong quá trình phát triển sự nghiệp.

Lập trình Web không phải chuyện một sớm một chiều

Trong quá trình học trở thành Web Developer, chắc hẳn bạn sẽ gặp thách thức về những "khái niệm dễ dàng gây nhầm lẫn", "không biết nên học ngôn ngữ gì vì có quá nhiều thứ để học", hay "quá nhiều công cụ, công nghệ thay đổi liên tục, học xong rồi cũng dễ quên".

Đây đều là vấn đề chung ai cũng sẽ gặp phải. Từ trải nghiệm bản thân, mình nghĩ câu hỏi lớn nhất bạn thường hay gặp chính là: "Nên học framework nào?", "Nên học ReactJS hay AngularJS?", hay "Nên học Java hay C#". Việc tự đặt ra những câu hỏi này ngay thời điểm bắt đầu sẽ là rào cản lớn nhất ngăn trở bạn bắt đầu việc học, đừng để quyết định trong tương lai cản trở việc học tập của bạn ở thời điểm hiện tại. Một trong những cách để thoát khỏi sự do dự là bám sát lộ trình mà bản thân bạn muốn trở thành.

[1] *Hackathon: tên gọi chung những cuộc thi kéo dài 24/36/48 giờ liên tục dành cho phát triển ý tưởng công nghệ*

Ví dụ, như đã chia sẻ ở trên, nếu bạn muốn trở thành Front-end Developer, yếu tố tiên quyết là phải học HTML, CSS, rồi sau đó tiếp tục học đến JavaScript. Sau khi bạn đã có đủ kiến thức về những ngôn ngữ đó rồi, việc sử dụng framework nào sẽ không còn là vấn đề lớn nữa.

Việc tạo ra một lộ trình đồng nghĩa với chia nhỏ quá trình thành từng bước một, điều này giúp bạn thực hiện từng bước dễ dàng và tập trung hơn. Hơn nữa, bạn sẽ không phải lo lắng cho những thứ nằm gần cuối lộ trình ngay khi bắt đầu. Ví dụ, nếu bạn học ReactJS, hãy bắt đầu bằng việc viết những ứng dụng hiển thị một danh sách đơn giản. Sau đó tiếp tục phát triển nó lên, như biến nó thành một ứng dụng nhắc việc (Todolist), cố gắng thực hiện đầy đủ các chức năng như thêm, xóa, sửa. Sau khi đã có ứng dụng hoàn chỉnh, hãy kết nối với máy chủ để lưu trữ dữ liệu của ứng dụng, liên kết với mạng xã hội, hay chức năng tùy ý mà bạn có thể nghĩ ra. Kế tiếp, bạn có thể làm thêm những hiệu ứng chuyển động (animation) để ứng dụng thêm bắt mắt. Bằng việc chia nhỏ quá trình như vậy, từ một ứng dụng đơn giản, đến cuối cùng bạn sẽ có một sản phẩm tương đối phức tạp. Ngoài ra, trong suốt quá trình, với mỗi bước bạn đều phải học thêm cái mới và ứng dụng lại những thứ mình đã học.

Cũng như việc học tất cả mọi thứ khác, học Web Development không thể kết thúc trong một sớm một chiều. Việc cố gắng hiểu qua loa một số khái niệm có thể phá hoại cả quá trình của bạn. Vì thế, khi học một khái niệm mới, nên chắc chắn bạn đã thử áp dụng nó, chơi với nó, cảm thấy thoải mái với nó, và thậm chí hãy mang nó kết hợp với các khái niệm khác nếu có thể. Quá trình này thoạt đầu có vẻ mất nhiều thời gian hơn việc đọc và hiểu thật nhanh, nhưng thật sự bạn sẽ tiết kiệm đáng kể thời gian vì không phải quay lại để gợi nhớ, và kiến thức cũng sẽ vững chắc hơn rất nhiều.

Hãy thoải mái bản thân khi học, nếu bạn coi những khái niệm mới là một món đồ chơi mới, bạn sẽ dành thời gian chơi với nó, thử ứng dụng nó với những thứ xung quanh, bạn sẽ thấy thú vị và dành thời gian cho nó nhiều hơn. Đừng gò ép bản thân phải code vài tiếng mỗi ngày, thay vào đó, chỉ cần bạn tạo ra một thời gian biểu phù hợp, hằng ngày dành một chút thời gian để code, suy nghĩ về những thứ đã học, bạn sẽ dễ cảm thấy thích thú và nhớ lâu hơn.

Kết

Phát triển Web là kỹ năng quan trọng mà bất cứ lập trình viên nào cũng nên biết. Nhu cầu về Web Developer cho các doanh nghiệp trên thị trường hiện tại là không nhỏ. Ngoài khả năng gia nhập các tập đoàn hay công ty công nghệ lớn, các bạn có rất nhiều cơ hội tham gia những công ty khởi nghiệp. Cùng với phong trào khởi nghiệp đang phát triển mạnh mẽ, mọi công ty đều cần phát triển Web, vì Web chính là phương thức giúp đưa sản phẩm ra thị trường nhanh nhất và tiết kiệm chi phí nhất.

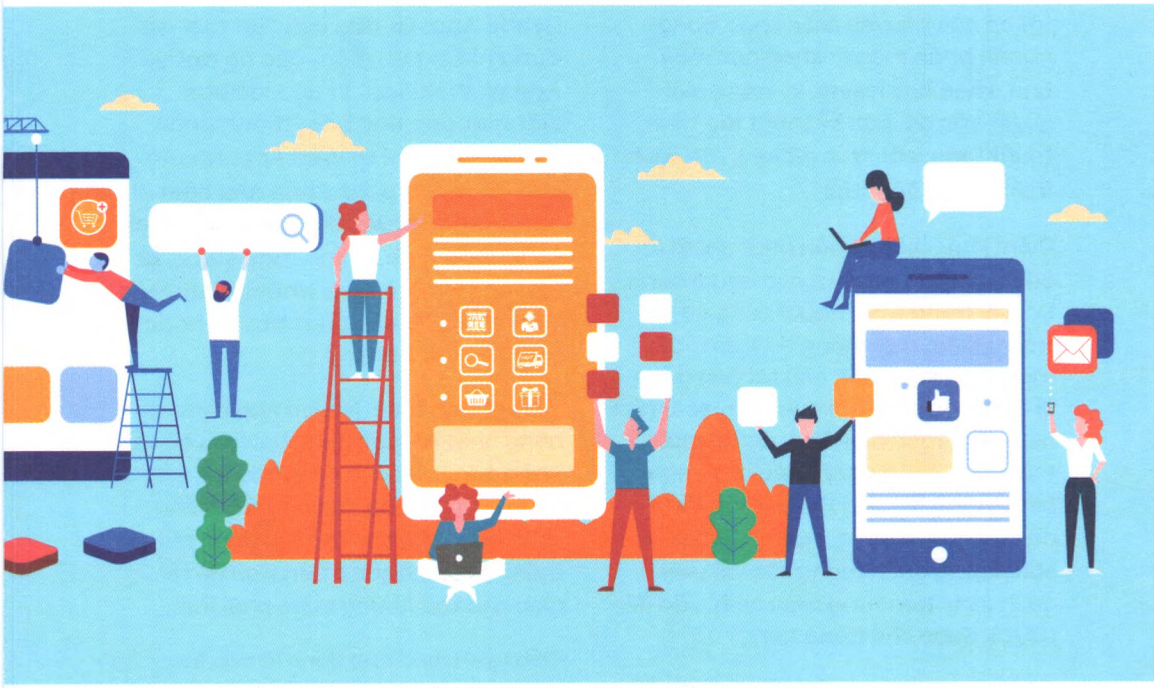
Hiện nay, theo những nghiên cứu từ các công ty tuyển dụng, mức lương hiện tại của một lập trình viên Web có thể từ 12 triệu cho đến 40 triệu tùy vào trình độ. Mức thu nhập không tồi, nhu cầu tuyển dụng cao, cùng với khả năng làm việc từ xa, Web Developer đang trở thành một ngành hot mà bất cứ bạn trẻ nào có đam mê với ngành phần mềm đều không muốn bỏ qua.

Đừng gò ép bản thân phải code vài tiếng mỗi ngày, thay vào đó, chỉ cần bạn tạo ra một thời gian biểu phù hợp, hằng ngày dành một chút thời gian để code, suy nghĩ về những thứ đã học, bạn sẽ dễ cảm thấy thích thú và nhớ lâu hơn.

LẬP TRÌNH MOBILE - CUỘC CÁCH MẠNG THAY ĐỔI CUỘC SỐNG

Khách mời phỏng vấn: **Đỗ Tuấn Anh - CEO Appota**
Biên soạn bài viết: **Ban biên tập**

Nói tới ngành công nghệ thông tin là nói tới rất nhiều hướng đi khác nhau, nhưng nếu có một lĩnh vực nào đó thực sự tạo nên cuộc cách mạng trong vòng hơn một thập kỷ trở lại đây, đó chính là những thiết bị di động. Ngày nay, ⅔ dân số thế giới được kết nối với ít nhất một thiết bị di động. Các ứng dụng trên mobile cho phép bạn có thể làm mọi thứ từ mua sắm, học tập, hẹn hò cho đến chơi game. Tiềm năng và cơ hội của mảng lập trình mobile là rất lớn, đồng thời "bài toán mobile" cũng đem tới những thử thách vô cùng thú vị cho các nhà phát triển. Thông qua bài viết, tôi muốn chia sẻ với các bạn về hành trình khai phá mảnh đất đầy tiềm năng này của Appota, biết đâu bạn sẽ tìm thấy niềm cảm hứng cho mình?



Tổng quan về lập trình mobile

Là người tốt nghiệp khoa Sử - Đại học Khoa học Xã hội và Nhân văn, Đại học Quốc gia Hà Nội, tôi từng muốn trở thành nhà báo nhưng bản thân lại có đam mê lớn với tin học từ cấp 3. Thời điểm năm 2009 điện thoại di động "cục gạch" rất phổ biến với người Việt Nam. Hầu như ai cũng sở hữu loại điện thoại mà bạn phải nhắn tin không dấu, dùng teencode để bấm phím thật nhanh. Đó là thời điểm Internet và GPRS ra đời, điện thoại dựa trên Java trở thành mặt hàng được các công ty công nghệ tại Việt Nam lựa chọn để khai thác và mở rộng dịch vụ. Thế nhưng trong dòng chảy mạnh mẽ đó, tôi có một suy nghĩ khác: điện thoại thông minh (smartphone) mới là công nghệ của tương lai. Tôi bắt đầu nhen nhóm tham vọng của mình đối với mảng này.

Quay lại câu hỏi: Lập trình di động (mobile) là gì? Hiểu một cách đơn giản, lập trình mobile là sử dụng các ngôn ngữ lập trình để tạo ra và phát triển các ứng dụng gia tăng tiện ích cho thiết bị di động. Hiện nay chúng ta không còn lạ lẫm với việc các ứng dụng di động dành cho smartphone len lỏi vào từng góc cạnh của đời sống hiện đại. Đứng đằng sau những ứng dụng đó là các lập trình viên di động.

Rất nhiều bạn trẻ khi tìm hiểu về lĩnh vực này sẽ thắc mắc: Vậy lập trình mobile và lập trình web có gì giống và khác nhau? Mảng nào "hot" hơn mảng nào?

Theo nhiều chuyên gia đánh giá, không có sự khác nhau rõ rệt nào về độ khó, mức thu nhập, cũng như khả năng phát triển đường dài khi so sánh giữa 2 mảng lập trình Web và Mobile. Về chuyên môn, chúng đều yêu cầu một tập hợp các kiến thức và kỹ năng công nghệ giống nhau (và giống với

tất cả các chuyên môn khác trong ngành phần mềm): khoa học máy tính, khoa học mạng, kỹ năng giải quyết vấn đề, các kỹ thuật lập trình: thuật toán, cấu trúc dữ liệu, mã sạch, thiết kế, tái cấu trúc,...

Điểm khác biệt cơ bản nhất chính là các bước trải nghiệm của người dùng. Đối với trang web, người dùng có thể trải nghiệm mọi thông tin trên một màn hình duy nhất, còn tại ứng dụng di động, họ phải trải qua rất nhiều màn hình như: màn hình khởi động app, màn hình đăng nhập (nếu có), màn hình thông tin,... Vì thế, khi lập trình cho thiết bị di động, bạn phải hiểu biết sâu hơn về cách tương tác với user, cách xử lý trên từng màn hình, vấn đề UX/UI[1] được chú trọng hơn.

Lập trình mobile có 3 hướng phát triển: Native App, Web App và Hybrid App. Cụ thể:

Native App: Lập trình viên code hoàn toàn bằng một ngôn ngữ lập trình dành riêng cho hệ điều hành của thiết bị đó (Android code bằng Java, iOS sử dụng Swift). Với Native App, các bạn sẽ thực hiện được rất nhiều các tinh chỉnh vì có thể dễ dàng tận dụng đầy đủ các tính năng của thiết bị: sử dụng camera, gọi đến bộ nhớ truy xuất các tính năng cảm biến chuyển động,...

Web App: Là ứng dụng chạy trên nền Web (một dạng nâng cấp các phiên bản Website). Các Web App có khả năng chạy tương tự giống các ứng dụng độc lập, như khả năng chạy offline, full màn hình, tương tác với phần cứng của điện thoại, nhưng sẽ bị hạn chế ở một số tính năng nâng cao đòi hỏi liên kết sâu vào hệ thống.

Hybrid App: Là đứa con "lai tạo" sử dụng những ưu điểm của cả Native App và Web App. Ví dụ, các bạn viết một app đọc báo, thành phần đọc text có thể là Web App, nhưng một số thành phần khác như hiện notification nhắc nhở bạn đọc, chỉnh sáng màn hình, chỉnh màu sắc,... là sử dụng Native App. Những bạn code Hybrid phải biết 1 chút Native hoặc tải các SDK[2] hỗ trợ.

Về mặt kiến thức, Native App hiện vẫn nặng nhất bởi các bạn phải hiểu cả về phần cứng và cách nó tương tác với code như thế nào. Hybrid thì bớt khó khăn hơn vì bạn có thể tận dụng những thành phần có sẵn ở bên web để tối giản kha khá những bước phải làm.

Hiện tại, Hybrid App đang là trào lưu. Nếu cách đây 2 - 3 năm, một dự án thường yêu cầu có 1 bạn code Web, 1 bạn code Android, 1 bạn code iOS, bây giờ có thể giảm xuống chỉ còn 1 người nhờ sự xuất hiện những ngôn ngữ lập trình có khả năng code xuyên nền tảng (cross platform). Các ngôn ngữ như Kotlin hay Typescript (được xây dựng dựa trên nền tảng của Javascript) có thể code trên 1 nền tảng nhưng xuất ra file cho cả Web App, PC, Android, iOS (khác với những ngôn ngữ lập trình trước đây có phân biệt rõ ràng giữa lập trình di động và lập trình web, như lập trình web chỉ quan tâm đến HTML, PHP, lập trình Android dùng Java, iOS dùng Swift,...).

Việc sử dụng những ngôn ngữ lập trình xuyên nền tảng cũng giúp các công ty giảm tải được số nhân sự. Tất nhiên, để làm được tốt bạn vẫn cần phải học thêm các định nghĩa cơ bản cho từng nền tảng (như từ Mobile App sang Web App cần hiểu định nghĩa về Header,

[1] *trải nghiệm/giao diện người dùng*

[2] *Software development kit: Bộ công cụ phát triển phần mềm*

Footer, SEO,... còn từ Web App sang Mobile App phải làm quen lại các khái niệm như nền tảng tương thích với ngôn ngữ lập trình, các loại màn hình và UI tương tác,...).

Đa phần các doanh nghiệp trẻ bây giờ đang theo hướng cross platform. Tuy nhiên, có những tập đoàn lớn vẫn duy trì riêng team iOS và Android bởi để thay đổi cả một hệ thống lớn là không đơn giản. Vì thế, không thể nói các bạn cứ chọn ngôn ngữ đa nền tảng thì xin việc chỗ nào cũng dễ, mà còn tùy vào nhu cầu của từng doanh nghiệp.

Thuận lợi và khó khăn của lập trình mobile

Có 2 điểm thuận lợi của lập trình mobile. Đầu tiên là hiện nay số lượng ngôn ngữ lập trình hỗ trợ mobile tương đối nhiều. Thứ hai, lập trình viên có rất nhiều template để tham khảo và chỉnh sửa.

Khó khăn (và cũng là thách thức đầy thú vị) của lập trình mobile là khả năng xử lý tính "mượt mà" của ứng dụng với từng thiết bị. Kể cả khi sử dụng các ngôn ngữ lập trình cross platform, bạn vẫn phải điều chỉnh, tối ưu và kiểm tra lại, mất khá nhiều thời gian. Khi lập trình web, ta chỉ cần quan tâm web có chạy tốt trên trình duyệt hay không? Tốc độ trang web nhanh hay chậm? Còn trên mobile, bạn cần quan tâm nhiều thứ khác. Ví dụ, nếu chạy ứng dụng mà thiết bị quá nóng, vậy là phần cứng gặp vấn đề. Tiếp theo là kích thước màn hình, mỗi thiết bị đều có các kích thước màn hình cố định nên chúng ta bắt buộc phải tối ưu chứ không thể kéo giãn ra như trang web được. Hay vấn đề về CPU, hệ điều hành iOS chỉ có 1 dòng CPU nhưng Android lại có nhiều hãng cung cấp CPU khác nhau. Tối ưu cho từng thành phần như thế nào là câu hỏi không đơn giản.

Kiến thức và tư duy cần có của một lập trình viên mobile

Về mặt kiến thức, bạn cần nắm ít nhất 1 trong số các ngôn ngữ lập trình phổ biến như Java, Javascript, C/C++, C#, Python, Swift, Objective-C, HTML5, React/React Native, Kotlin. Một khi các bạn đã xác định được mình sẽ học về ngôn ngữ nào thì cố gắng nghiên cứu, tìm kiếm nhiều hơn về nó. Hãy tự làm, tự lập trình các ứng dụng demo bởi nhà tuyển dụng sẽ đánh giá các bạn dựa trên những sản phẩm thực tế nhiều hơn là số lượng ngôn ngữ lập trình các bạn biết.

Về tư duy, tôi muốn mở rộng hơn những kỹ năng mà bạn vẫn thường nghe đến như tư duy logic, tư duy giải quyết vấn đề,... Bên cạnh tư duy kỹ thuật, bạn cần có *tư duy làm sao để ứng dụng làm ra được người dùng đón nhận.*

Trên thị trường hiện nay có 2 xu hướng làm sản phẩm ứng dụng di động:

Một là các bạn clone[1] các game/app của chính mình rồi up lên store (sản xuất ồ ạt). Các bạn có thể kiếm được nhiều user (user tải app và vào quảng cáo là bạn có tiền), từ 1 app tạo ra 10 "con nhân bản", mỗi "con" lại đến với một thị trường khác nhau. Việc này phụ thuộc khả năng tư duy xác định được người dùng mục tiêu của bạn như thế nào. Tuy nhiên, nhìn ở góc độ người dùng, họ sẽ bị loạn, ngợp, không biết ứng dụng nào nên dùng, game nào nên chơi.

Xu hướng thứ hai là phát triển một app/game trong thời gian dài. Ở một thị trường cạnh tranh như hiện tại, làm sao để app/game của bạn được người dùng biết đến và quan tâm? Đầu tiên, các bạn phải nghiên cứu để biết người dùng cần một ứng dụng thế nào? Thị trường đang thiếu cái gì? Sau đó, xem trong phân khúc đấy người dùng đang sử dụng ứng dụng nào (nghiên cứu đối thủ cạnh tranh để biết họ có gì, thiếu gì, mình cần làm tốt cái gì). Hãy tải 10 ứng dụng nằm trong top 10 của danh mục đó, sử dụng chúng rồi so sánh với nhau để tìm ra điểm ưu, nhược điểm. Khi đó, bạn sẽ có cái nhìn tổng quan về thị trường để bắt đầu xây dựng giá trị cốt lõi cho sản phẩm, chia chúng ra thành từng khâu nhỏ rồi bắt đầu phát triển. Những giá trị mà người dùng thụ hưởng cần được tưởng tượng ra trước, phác thảo rõ ràng chứ không phải cứ lập trình sản phẩm mình thích rồi mới tính cách để người dùng sử dụng. Trong giai đoạn tưởng tượng sản phẩm mới, bạn cần đến "sự điên rồ", nó sẽ khiến sản phẩm của bạn khác biệt và vượt trội.

Khi xây app, bên cạnh câu chuyện tối ưu sao cho ứng dụng của mình có trải nghiệm tốt, các bạn cần lắng nghe phản hồi của người dùng để không ngừng cải thiện. Đặc biệt với các app mới, từng comment của khách hàng đều rất đáng quý.

Ngoài ra, để app/game của bạn trở thành trào lưu, đừng bỏ quên gamification[2]. Nó giúp tạo ra cộng đồng vì người dùng sẽ tương tác lẫn nhau (ví dụ như cho phép người chơi game chia sẻ và so sánh điểm với người khác). Tôi nhấn mạnh vào yếu tố tạo cộng đồng vì nó mở ra cơ hội rất lớn để sản phẩm của bạn được nhiều người biết đến.

Hãy làm ứng dụng một cách có tâm nhất. Nếu chỉ làm với tâm lý kiếm tiền, đôi khi bạn sẽ gây ra ảnh hưởng tiêu cực đến sản phẩm.

Hành trình 10 năm xây dựng hệ sinh thái dựa trên mobile

Nếu bạn đã hình dung được những thông tin cơ bản về lập trình di động trong phần trên, đến đây, tôi muốn chia sẻ những trải nghiệm, câu chuyện của Appota, từ những ngày đầu xây dựng một vài sản phẩm trên mobile đến việc gây dựng cả hệ sinh thái như thế nào.

Dù không xuất thân là một lập trình viên nhưng tôi có niềm đam mê rất lớn với công nghệ, đặc biệt là mảng mobile. Tôi bắt đầu tham vọng thống trị thị trường phân phối nội dung số smartphone với việc thành lập diễn đàn GSM (một trong số các diễn đàn công nghệ điện thoại thông minh đầu tiên tại Việt Nam). 1 năm sau,

[1] *bản sao chính xác của một đối tượng có cùng đặc điểm và tính chất*

[2] *ứng dụng các thành phần của game (kỹ thuật, cách thức, luật chơi và những yếu tố khác,...) vào một hoạt động bất kỳ với mục đích tạo động lực và hứng thú cho người dùng, hay còn gọi là "game hoá"*

Appstore.vn (kho lưu trữ nội dung số bên thứ ba) ra đời và sản phẩm này là nền tảng, thúc đẩy quyết định theo đuổi thị trường smartphone của tôi. Appstore.vn có hàng chục nghìn lượt đăng ký thành viên mới mỗi ngày, giúp người dùng smartphone ở Việt Nam trong giai đoạn đầu dễ dàng tải trò chơi và mua ứng dụng, bởi thời điểm năm 2010 việc dùng ứng dụng trên các store quốc tế rất khó khăn.

Năm tiếp theo là bước ngoặt lớn nhất khi tôi tham gia chương trình của Founder Institute và quyết định cùng 12 anh em ra mắt Appota - công ty khởi nghiệp đầu tiên tại Việt Nam xây dựng nền tảng phân phối nội dung cho smartphone, bao gồm bộ công cụ SDK giúp các nhà phát triển ứng dụng dễ dàng phân phối tới hàng triệu người dùng tại Việt Nam. Các nhà phát triển chỉ cần tập trung phát triển sản phẩm, khâu phát hành, đóng gói và phân phối là việc của Appota.

Đứng trước một giấc mơ lớn, công việc của tôi và các cộng sự không hề dễ dàng. Những ngày đầu thành lập, chúng tôi ngồi trong căn phòng rộng chưa đến 20m2 tại đường Nguyễn Du, Hà Nội. Mỗi người làm cùng lúc nhiều nhiệm vụ khác nhau. Chúng tôi không chỉ là những anh chàng làm công nghệ trẻ tuổi, chỉ biết cắm mặt vào máy tính điện thoại để làm sản phẩm mà còn kiêm nhiệm nhiều nhiệm vụ khác như marketing, chăm sóc khách hàng,...

Lúc này, công ty đã có tác động mạnh mẽ đến thị trường phân phối nội dung tại Việt Nam và Đông Nam Á. Trong thời gian ngắn, tôi và các anh em đã hợp tác với hơn 10.000 nhà phát triển và quảng cáo trên toàn thế giới, bao gồm NHN, KaKao, Line, Metaps, VNG, VTC Online và FPT Online để cung cấp nội dung phù hợp văn hoá Việt cho người dùng.

Cuối năm 2012, chúng tôi tham gia vào thị trường game đầy rủi ro và cạnh tranh. Đó là một lĩnh vực mới mà không mấy ai trong đội ngũ có kinh nghiệm, ngoại trừ vai trò là những người chơi game từ lúc ngồi trên ghế nhà trường. Thế nhưng các thành viên vẫn kiên trì nghiên cứu và phát triển, thành quả cho đến hiện tại là Gamota, công ty phát hành game thuộc Appota Group, trở thành 1 trong 3 nhà phát hành game di động lớn nhất tại Việt Nam.

Một trong những chiến lược kinh doanh quan trọng của Appota là xây dựng cộng đồng người dùng mạnh mẽ và luôn mang lại trải nghiệm mới mẻ. Do đó, vào tháng 9 năm 2013, onClan, mạng xã hội đầu tiên dành cho game thủ, được chúng tôi thành lập tại TechCrunch Disrupt 2013 ở San Francisco. Đến cuối năm 2014, Appota đã đạt 22 triệu người dùng trên khắp Việt Nam và Đông Nam Á.

Đầu năm 2018, Appota đã hợp tác với Facebook để trở thành đối tác chính thức của Facebook tại Việt Nam và nhanh chóng triển khai chương trình Game Streamer đầu tiên và duy nhất tại Việt Nam. Nếu bạn yêu thích các streamer như ViruSs, MisThy, Chim Sẻ Đi Nắng, đây là những creator tiêu biểu đang hợp tác với Appota cùng hơn 300 creator nổi tiếng khác.

Bên cạnh đó, nhận thấy thị trường mobile đang có những bước chuyển biến lớn gắn liền với công nghệ IoT, Appota tiếp tục dấn thân phát triển các sản phẩm vật lý, hoạt động thông qua điện thoại thông minh thuộc dự án AppotaHome. Hiện tại có hàng ngàn gia đình đang sử dụng khoá bảo mật thông minh được vận hành bởi ứng dụng AChin do Appota phát triển. Ứng dụng này cũng được chúng tôi phát

triển đồng thời với giải pháp quản lý tài nguyên doanh nghiệp, bước đầu cho hành trình "chuyển đổi số".

Thị trường mobile hay rộng hơn là thị trường công nghệ luôn trong tình trạng hỗn loạn. Bạn có thể gặt hái thành công to lớn vào lúc này, nhưng cũng lập tức "hết thời" trong nháy mắt. Do đó, tôi và khoảng 500 thành viên ở Appota liên tục cải tiến, sáng tạo và phải sẵn sàng thay đổi, giới thiệu sản phẩm mới, đổi mới sản phẩm cũ, nâng cao trải nghiệm của người dùng để bắt kịp và dẫn đầu xu thế mobile và công nghệ. Các ứng dụng của Appota đến nay phủ rộng ở nhiều hình thức giải trí khác nhau như đọc sách, nghe nhạc, phim, xem tin tức, truyện tranh, Lịch Như Ý,... Chúng tôi cũng hướng tới những thử thách xây dựng các hệ sinh thái như Esports và mạng quảng cáo đa kênh (MCN), phát triển hệ thống thanh toán cho các cửa hàng ăn uống và địa điểm giải trí.

Hãy suy nghĩ thấu đáo, quyết định nghiêm túc và nhiều khát vọng

Hiện nay, lập trình mobile hiện được đánh giá là ngành hot nhất trong tất cả các ngành lập trình. Trên các trang tuyển dụng, tỷ lệ các job tuyển lập trình mobile cũng chiếm tới 70 - 80%. Tuy nhiên, đa số các bạn đều chưa đáp ứng được yêu cầu công việc nên các doanh nghiệp đang trong tình trạng khát nhân lực.

Vậy các bạn trẻ cần chuẩn bị gì để dấn thân vào lĩnh vực này?

Điều kiện tiên quyết là tiếng Anh, bởi chỉ có nó mới giúp bạn nghiên cứu mở rộng tri thức và không trở thành người "tối cổ" trong nghề.

Ngoài ra, bạn cũng cần chuẩn bị tâm thế rằng công việc của bạn là cuộc chơi lớn với các đồng nghiệp trên toàn cầu. Bạn không phải người làm trên một thửa ruộng, một khu vườn, mà sản phẩm bạn tạo ra có thể thay đổi trải nghiệm và cuộc sống của hàng tỉ người trên thế giới.

Thông thường, các bạn mới bắt đầu tham gia lập trình mobile sẽ phụ trách code cho một thành phần nhỏ của dự án. Sau khi tích lũy đủ kinh nghiệm, bạn có thể thăng tiến lên làm tech lead. Lúc này, bên cạnh chuyên môn lập trình tốt, bạn còn là người quyết định phong cách code – yếu tố rất quan trọng và định hướng toàn bộ dự án. Người tech lead sẽ định hình rõ cách code cho dự án, từ những việc đơn giản như sử dụng từng gạch đầu dòng thế nào, chỉ thị ra sao,... Nhờ vậy, khi người cũ rời đi và người mới tham gia, họ dễ dàng tiếp nhận mà không bị tình trạng "đổ vỏ" cho nhau. Sau vị trí tech lead, bạn có thể lên đến cấp độ product: Lúc này bạn có tư duy về mặt UX/UI, tư duy về người dùng, tư duy về kinh doanh. Đây là cấp độ mà bạn có thể nghĩ ra sản phẩm, bạn biết sản phẩm cần gì, code cần gì,... Người làm lập trình mobile giỏi có xu hướng phát triển thành Full-stack Developer, khi đó bạn sẽ là "super hero" được các công ty và startup luôn tìm kiếm.

Thật khó để nói trước những cuộc cách mạng tiếp theo trong ngành công nghệ, nhưng đối với lập trình mobile, chỉ khi có một thiết bị khác thay thế thì ngành này mới có sự biến đổi. Và trước khi điều đó đến, lập trình mobile vẫn tiếp tục là một trong những ngành top đầu về nhu cầu tuyển dụng trên thế giới.

Tất nhiên, không phải cứ đi đúng đường ngay từ đầu mới làm được lập trình mobile. Tôi không chọn nghề này trên ghế nhà trường, bản thân không có sự chuẩn bị một cách bài bản cũng như không có khả năng tiếp cận thông tin sớm và rộng mở như thế hệ trẻ hiện tại. Ngành này đến với tôi như một cái duyên, và tôi phải nỗ lực học hỏi mọi thứ với 200% khả năng để chớp lấy cơ hội. Thật sự không ai nghĩ người tốt nghiệp ngành Sử lại làm CNTT, nhưng tôi chưa bao giờ thấy kiến thức về xã hội nhân văn bị lãng phí. Bởi ngành CNTT cần nhiều thứ hơn chỉ biết code. Suy cho cùng, đích đến của mọi sản phẩm công nghệ là phục vụ cuộc sống con người, vậy nên người làm sản phẩm cần hiểu người dùng cần gì, muốn gì, sử dụng như thế nào và góc nhìn xã hội nhân văn đã giúp tôi nhanh chóng nắm bắt những yếu tố đó, giúp sản phẩm hoàn thiện và có đầu ra tốt.

Kết hợp giữa kiến thức CNTT và những kiến thức nền tảng sẵn có, trải nghiệm nhiều sản phẩm, tôi tiếp tục tìm đến những chuyên gia, tham khảo và học hỏi họ ở nhiều mảng khác nhau để biến ý tưởng của tôi thành hiện thực. Từ khi còn làm nhân viên hay cho đến lúc giữ vị trí CEO, tôi luôn kết hợp những yếu tố này để hoàn thiện các sản phẩm. Công thức ở đây chính là:

Kiến thức nền tảng sẵn có
+ Kiến thức CNTT
+ Trải nghiệm liên tục các sản phẩm công nghệ
=> Tưởng tượng sản phẩm mới có giá trị cho người dùng
=> Tìm đến chuyên gia để cùng họ biến ý tưởng thành hiện thực

Rất nhiều bạn nói về chuyện lập trình mobile có thể trở thành triệu phú sau 1 đêm, như trường hợp Flappy Bird của Nguyễn Hà Đông. Tôi nghĩ điều này hoàn toàn khả thi. Ở Việt Nam có nhiều nhà lập trình rất nổi tiếng nhưng họ ẩn mình nên không có nhiều người biết đến. Sản phẩm của họ có cả triệu, chục triệu người dùng. Thực tế cho thấy, không chỉ ngành lập trình mobile mà ngành nào cũng vậy, nếu muốn thành công, bạn phải hội tụ rất nhiều yếu tố, năng lực, thậm chí là may mắn. Khi chuẩn bị kỹ càng về học thức, kinh nghiệm, kỹ năng phản ứng với các thử thách,... cuộc sống sẽ mở ra cho bạn nhiều cơ hội chạm đến thành công hơn.

Lời khuyên của tôi dành cho các bạn trẻ là hãy hỏi nhiều, trải nghiệm nhiều sản phẩm, "nghịch" nhiều thứ, mạnh dạn tưởng tượng. Dù sẽ có lúc người khác bảo bạn "không thể được", bạn hãy mạnh dạn tìm đến những người có thể giúp bạn "make it happen" trong mọi lĩnh vực, từ kỹ thuật cho đến kinh doanh, marketing,...

Nhiều người nói "không" hoặc "để tôi nghĩ đã" như một phản xạ có điều kiện, dù bạn ở lứa tuổi nào đi nữa cũng nên thay đổi cách phản ứng này. Hãy nói "Được, tôi sẽ cân nhắc" và nghĩ cách biến thành hiện thực! Tôi nghĩ đó là bước đầu tiên cho mọi thành công!

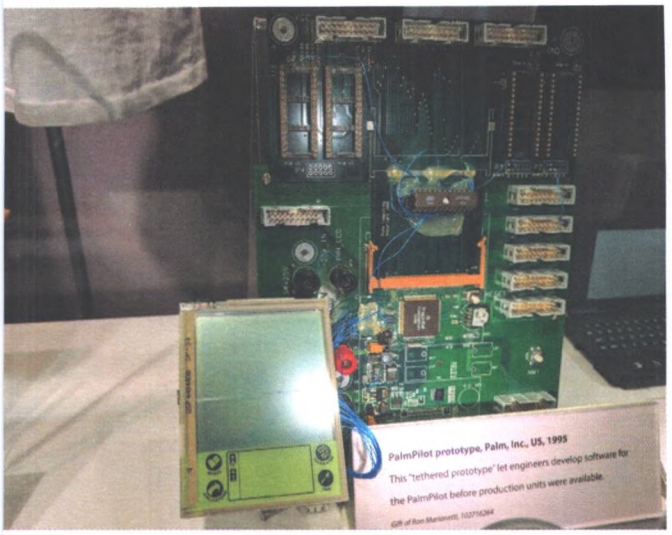
Lập Trình Nhúng TẠI SAO VÀ CÓ GÌ VUI?

Tác giả: **Curly Rae Braces**
(Cảm ơn L.N.)
Kỹ sư phần mềm

Máy tính ngày nay xuất hiện ở mọi nơi nếu chúng ta để ý. Chúng không chỉ là chiếc máy bạn dùng để lướt Facebook, hay chiếc điện thoại thông minh bạn dùng hằng ngày, mà nó nằm cả trong chiếc điều khiển từ xa, chiếc lò vi sóng, chiếc đồng hồ báo thức, quạt điện, tủ lạnh bạn dùng. Bạn có thể tìm được máy tính đúng nghĩa trong những vật bạn không hề nghĩ tới: thẻ ATM, SIM điện thoại, thẻ nhớ máy ảnh, hay chiếc thẻ tên bạn đeo trước ngực. Giờ đây, máy tính trở nên phổ biến đến mức người ta gắn chúng vào từng cuốn sách trong thư viện dưới cái tên thẻ RFID[1].

Sự hiện diện ở từng ngõ ngách của những chiếc máy tính nhỏ nhúng trong các thiết bị tiêu dùng hàng ngày khai sinh ra một ngành mới: Lập trình nhúng.

[1] *Radio Frequency Identification: Công nghệ nhận dạng đối tượng bằng sóng vô tuyến*



Board mạch "prototype" (bản phát triển) của máy tính cầm tay Palm (năm 1995). Hai viên pin tiểu AA có thể nuôi sống máy một tháng, lợi thế điển hình của một thiết bị nhúng.

(Ảnh do tác giả chụp từ Computer History Museum)

Lập trình nhúng là làm gì, thiết bị nhúng là gì?

Lập trình nhúng (embedded programming) là lập trình trên những thiết bị phần cứng đặc biệt, ít tài nguyên và làm việc chuyên dụng. Ngay cả một chiếc laptop hiện tại cũng có rất nhiều lõi CPU, rất nhiều GB RAM và đĩa cứng. Ngược lại, một chip nhúng có thể chậm hơn cả trăm lần, có vài KB RAM và không có đĩa cứng. Điều này làm cho bạn phải hiểu chuyên sâu hơn về phần cứng và những gì bộ xử lý có thể làm để sử dụng tài nguyên bạn có một cách hiệu quả nhất.

Nếu máy tính cá nhân hay điện thoại thông minh chỉ có một vài kiểu, chạy một vài hệ điều hành thì các thiết bị nhúng lại muôn hình vạn trạng. Nói ai đó làm lập trình nhúng cũng tương tự nói ai đó là nhà nghiên cứu sinh học, đó là một công việc rất phổ quát. Các máy tính hay chip nhúng là cái gì, chạy cái gì là một câu hỏi có rất nhiều câu trả lời. Các thiết bị như WiFi router, bóng đèn thông minh và một số thiết bị IoT chạy trên nhân hệ điều Linux và Busybox (là bộ công cụ nhỏ, nhúng bao quanh nhân Linux) thì tương đối giống với máy tính. Có nhiều thiết bị không chạy hệ điều hành gì cả, hoặc chạy một hệ điều hành không đa nhiệm, hoặc chỉ chạy một môi trường máy ảo nào đó như Java. Có nhiều thiết bị khác như modem hay thẻ SIM chạy các chương trình được ai đó viết tay bằng ngôn ngữ Assembly. Nhiều con chip điều khiển trong các thiết bị như lò vi sóng, thiết bị thu phát WiFi, NFC[1], hay con chip làm mã hóa và giải mã của máy trò chơi điện tử thậm chí còn có kiến trúc tập lệnh (instruction set) của riêng nó. Ta có thể so sánh con người rất thông minh, cũng giống chiếc máy tính cá nhân hay chiếc điện thoại; các thiết bị nhúng thì như tất cả các loài sinh vật khác trên trái đất này. Con người đa dạng nhưng các loài sinh vật trên trái đất còn đa dạng hơn: Khỉ hay cá heo rất thông minh và làm được nhiều việc giống người, chó thì kém thông minh hơn một chút, rắn hay cây

[1] *NFC (Near-Field Communications): công nghệ kết nối không dây tầm ngắn sử dụng cảm ứng từ trường để kết nối giữa các thiết bị (smartphone, tablet, loa, tai nghe,...) khi có sự tiếp xúc trực tiếp*

cối lại càng kém thông minh, nhưng loài sinh vật nào cũng hữu dụng nếu chúng ta biết cách sử dụng chúng.

Lập trình nhúng cần bạn suy nghĩ khác ở những điểm nào?

Vậy viết "app" trên điện thoại có được gọi là lập trình nhúng không? Trước kia, lập trình app trên điện thoại cũng có thể coi là lập trình nhúng. Nhưng đa phần điện thoại, máy tính bảng bây giờ tài nguyên nhiều và chạy hệ điều hành bình thường, không khác gì máy tính cả. Vì thế, lối suy nghĩ khi ta viết "app" giống với lập trình thông thường hơn là lập trình nhúng.

Vậy tại sao mọi thứ cần rắc rối như vậy? Tại sao ta không dùng máy tính cá nhân hay điện thoại di động để làm mọi thứ? Có nhiều lý do, nhưng một trong những lý do là sự hạn chế mang lại nhiều điều có ích. Bạn không thể gắn điện thoại lên chân chim để theo dõi hành trình di trú của nó được, bạn cần một con chip và bảng mạch nhúng rất nhỏ để làm việc này. Hơn nữa, không phải thiết bị gì làm được nhiều việc cũng siêu việt hơn thiết bị đơn giản. Một chiếc máy trợ tim, hay máy đáp tàu vũ trụ, hay bộ phận chống trượt trên ô tô thường là các thiết bị phải đánh cược vào việc ngay cả trong tình huống xấu nhất, chúng cũng có thể có đưa ra quyết định trong một thời gian ngắn. Cơ chế làm việc của một hệ điều hành đa nhiệm thông thường như Windows hay Linux không thể đảm bảo một chương trình có được thực thi vào đúng lúc không (đó cũng là lý do vì sao khi máy tính bạn bận làm gì thì các chương trình trên đó trở thành "đơ" - "Not Responding"). Ngược lại, một thiết bị nhúng tuy có CPU chậm nhưng khi chạy đơn nhiệm lại có thể đảm bảo chuẩn xác cao độ về thời gian mà phần lớn các máy tính cá nhân (nhanh hơn thậm chí cả trăm ngàn lần) không đảm bảo được.

Ta có thể ví lập trình cho máy tính cá nhân, web hay điện thoại di động cũng như đi rèn luyện, đào tạo nghiên cứu con người, làm được nhiều việc nhưng có nhiều vấn đề, hay mất tập trung. Còn lập trình nhúng như đi điều khiển chó chăn cừu và đánh hơi bom, voi làm xiếc, hoặc chăn ong lấy mật. Niềm vui của lập trình nhúng giống như niềm vui khi bạn dạy được con thú vậy. Bạn sẽ được đền đáp cho hy sinh của mình bằng những con chip hay cỗ máy làm được những việc mà máy tính hay điện thoại không làm được, mang cho bạn tốc độ phản hồi nhanh hơn, tốn ít tài nguyên hơn, rẻ hơn.

Bạn cần những công cụ gì?

Về công cụ hay ngôn ngữ, làm việc với các thiết bị trên, bạn sẽ đến với thế giới kỳ diệu và đa dạng những kỹ năng, thiết bị, tập lệnh, ngôn ngữ bạn cần. Nói chung, lập trình nhúng thường bị giới hạn bởi những bộ vi xử lý chậm và đơn giản, nên thường yêu cầu bạn hiểu và sử dụng những ngôn ngữ cấp thấp hơn (như C hoặc Assembly), nhưng không phải không có ngoại lệ.

Giả sử bạn muốn học để sau này làm việc được với những thiết bị như thế, bạn nên bắt đầu từ đâu, bằng cách đặt câu hỏi gì? Tôi nghĩ một điều quan trọng cần nhận ra, đó là công việc thực tế khác với ngành học ở chỗ công việc thì cụ thể, còn ngành học lại tổng quát. Khi chúng ta đang dùng máy trò chơi điện tử, chắc chắn có ai ở một nơi nào đó trên thế giới đi thiết kế, lập

trình cho con chip mã hóa trên đó. Nhưng vào trường Đại học, không có ngành nào là ngành thiết kế chip mã hóa cho máy trò chơi điện tử cả. Thậm chí, sách viết về vấn đề này cũng rất hiếm. Như vậy, chúng ta chỉ cần cách nghĩ và kiến thức về điện tử và hệ thống máy tính nói chung chứ không nên đánh cược vào bất cứ một ngôn ngữ hay công cụ đặc biệt gì.

Tôi nghĩ một cách hiệu quả là bắt đầu từ vấn đề hoặc một dự án (vấn đề hay dự án của người khác cũng tốt). Khi bắt tay vào việc, ta cần dùng đủ mọi công cụ để xây dựng giải pháp mới. Những gì tôi muốn chia sẻ với bạn là quá trình bắt nguồn ý tưởng, quá trình đặt câu hỏi tại sao, thay vì câu hỏi như thế nào. Vì một khi bạn biết mình thực sự muốn làm gì, vì sao mình làm vậy, mình có mục đích gì, dần dần sau nhiều lần thử đi thử lại, bạn sẽ tìm được câu trả lời cho việc làm như thế nào một cách hợp lý nhất. Việc lập trình cho những thiết bị đó là công việc không đơn giản, nhưng chắc chắn sẽ có người thích (cũng như có người thích làm thầy cô giáo dạy trẻ con, có người thích huấn luyện chó đi làm xiếc).

Những cơ duyên không rủ cũng tới

Bản thân tôi làm lập trình nhúng trong khi được đào tạo ngành Khoa học Máy tính phổ quát (và tôi tin một kỹ sư máy tính hay điện tử đều có thể làm lập trình nhúng được). Tôi tìm thấy niềm vui trong việc làm với các thiết bị nhỏ để chúng thực hiện công việc tôi cần. Đó là khi tôi làm việc "gãi đúng chỗ ngứa" của chính mình (và của nhiều người). Xây dựng các giải pháp như vậy mang đến cho tôi nhiều cơ duyên bất ngờ.

Việc đầu tiên tôi được "gãi ngứa" cho mình và người khác là điều khiển máy chiếu trong phòng học. Thời đó, tôi là sinh viên Đại học làm hỗ trợ kỹ thuật trong bộ phận IT của trường. Lúc đó, một việc tôi thường xuyên phải làm là đi tới từng lớp học để hỗ trợ kỹ thuật. Vì mỗi phòng học có một máy chiếu khác nhau, các giáo sư lớn tuổi nhiều khi không biết bấm nút nào trên điều khiển để bật tắt máy chiếu và chọn đúng nguồn laptop hay đầu DVD trong phòng. Khi họ loay hoay bật tắt được (hoặc gọi tôi tới giúp) thì đã mất toi 1/4 thời gian tiết học. Thử nhân lên mỗi tiết học bao nhiêu lớp trong trường, mỗi lớp có bao nhiêu sinh viên và giáo viên, và mỗi người tốn bao nhiêu phút chờ đợi để thiết lập bài giảng, ta sẽ thấy thời gian tiêu tốn rất nhiều. Giá mà có một thiết bị giúp chuẩn hóa các thao tác, làm cho việc điều khiển máy chiếu dễ dàng thì hay biết mấy!

Để giải quyết vấn đề này, trường tôi có mua thiết bị điều khiển lắp vào bục giảng của một công ty làm điện tử. Thiết bị này rất đẹp nhưng trang bị nó trong một phòng học rất tốn kém, từ 500 - 1000 đô la cho một bảng mạch nhỏ chỉ làm mỗi việc bật tắt và điều khiển cơ bản máy chiếu. Vì thế, trường giao cho tôi nhiệm vụ tìm phương án rẻ hơn, để tiền đầu tư của nhà nước được hiệu quả.

Tôi nghĩ giá như có thiết bị nào như máy Raspberry Pi[1] thì tốt. Nhưng lúc đó thiết bị điện tử không nhiều như bây giờ, ngay cả Raspberry Pi cũng mới ra đời nên chỉ nhà phát triển phần mềm nào vinh hạnh lắm mới mua được một cái để thử. Một hôm nửa đêm đọc trang diễn đàn về Raspberry Pi,

[1] *Máy tính nhỏ gọn và rẻ tiền chạy nhân Linux, tích hợp nhiều phương tiện để tương tác với các vi mạch điện tử khác.*

tôi thấy có người nói về một cái đồng hồ thông minh mà người làm ra nó tùy hứng cho ai muốn viết phần mềm gì trên đó thì viết. Đây là thiết bị mở chạy Linux do kỹ sư Andrew Huang thiết kế, chức năng làm đồng hồ hay vật trang trí thông minh tựa như Amazon Echo Show[1] vậy. Quả thực vậy, tôi thấy có người bán trên Ebay chỉ 20 đô la một chiếc. Nó có tên là Insignia Infocast (tên khởi nguồn là Chumby). Tôi lên eBay mua luôn một chiếc và mày mò dịch phần mềm để chạy trên đó. Tôi vào wiki để xem, cuối cùng nhận ra mình có thể viết được cả một giao diện đơn giản dựa trên công nghệ web để hiển thị điều khiển thiết bị trong phòng học.

Infocast để chạy trình duyệt và đọc từ cảm biến nhiệt

Tôi viết bản kiến nghị IT nhà trường đầu tư kinh phí mua hàng trăm thiết bị như vậy và được chấp thuận. Tôi nhớ có hôm đang làm việc thì người trong trường gọi ra để đón một xe tải chất đầy hàng trăm hộp Infocast. Và thế là tôi dành phần lớn thời gian phát triển phần mềm trên Infocast để làm một việc hoàn toàn khác với ý tưởng ban đầu của người kỹ sư làm ra nó. Điều này khiến mọi người trong phòng IT rất thích thú. Cùng lúc đó, tôi có email cho kỹ sư Huang để chia sẻ khám phá của mình. Ông gửi email lại động viên, rồi bảo lúc nào cần việc thì nhắn. Sau này, nhờ những trao đổi định mệnh ấy mà kỹ sư Huang giới thiệu tôi qua Singapore thực tập.

Sau công việc này, tôi bỏ lập trình nhúng để học tiếp lên sau Đại học, nghiên cứu ngành Tin - Sinh học, nhưng cơ duyên với những thiết bị nhúng vẫn theo gót.

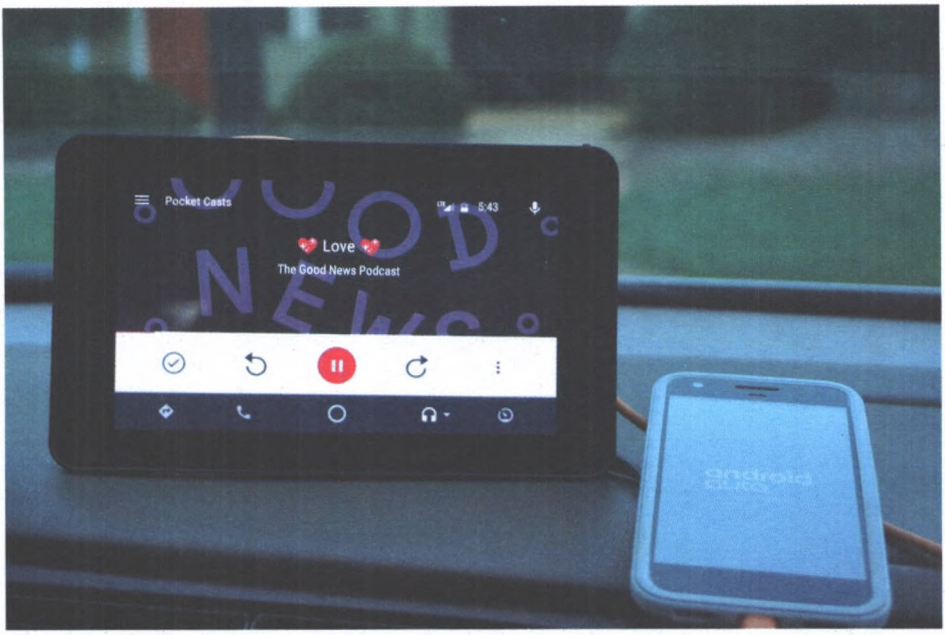
Vào cuối tuần hay lúc nào chán, tôi mở cái rương đầy đồ điện tử và Raspberry Pi ra để viết phần mềm, gắn cái nọ với cái kia. Thời gian ấy, tôi nhận ra một dự án phần mềm có rất nhiều tiềm năng. Phần mềm này giúp người dùng cài một chương trình kết nối smartphone Android tới ô tô để chiếc xe trở thành trung tâm giải trí thông minh y như điện thoại mà không cần phải với điện thoại gây mất tập trung. Chương trình này được phát hành tự do và có thể chạy trên cả Raspberry Pi. Như vậy, chỉ cần mua thêm một màn hình to là ai cũng có thể biến một chiếc xe cũ thành xe thông minh. Điều đáng tiếc là người đó chỉ phát hành phần mềm mà không đóng gói gì cả. Tôi bèn nghiên cứu đóng gói một hệ điều hành (đúng hơn là một distro[2]) dựa trên Linux để cấu hình phần mềm đó sao cho chỉ cần cắm là chạy. Tôi gọi đó là phần mềm Crankshaft (cái tên tôi google ra vì tôi chẳng biết tên bộ phận nào của ô tô), phát hành mã nguồn theo giấy phép tự do. Lúc tôi bấm nút phát hành bản beta[3] của distro này là lúc tôi làm việc một tuần thâu đêm suốt sáng.

---

[1] *Thiết bị gia đình thông minh kết hợp giữa một chiếc loa và và khả năng nhận diện giọng nói.*

[2] *"Distro": Hệ điều hành bao gồm lõi Linux và các chương trình phụ trợ trên đó để chạy các ứng dụng.*

[3] *Phiên bản dùng thử, nhằm tung ra để người dùng sử dụng và kiểm tra lỗi, phản hồi*



Máy tính Raspberry Pi chạy Crankshaft lắp trong ô tô

May mắn đã mỉm cười với tôi, vì tôi không những không bị đuổi khỏi trường khi cả một tuần không làm gì, mà phần mềm đó được hàng ngàn người đón nhận ngay tuần đầu tiên ra mắt. Tôi được nhiều công ty mời phỏng vấn, một trong số đó là Tesla - công ty làm ra những chiếc xe thông minh nhất. Tôi nghe theo lời một người bạn thân hiểu mình, phỏng vấn với họ và được mời làm việc ở đó (tôi làm việc cho họ từ đó đến nay). Ngày ngày, tôi cắm cái nọ vào cái kia, làm việc với phần cứng, phần mềm, USB, Bluetooth, Linux. Những việc ấy, tôi nghĩ mình rất sẵn lòng làm miễn phí như trước giờ vẫn làm.

Nghề và Người rồi sẽ quay trở lại với nhau

Tôi chưa từng được học một lớp lập trình nhúng nào trong đời, và cũng chưa bao giờ được đào tạo về ngành điện tử. Tôi không nghĩ đến một ngày lại có người trả tiền cho mình để làm những công việc như vậy, và càng không nghĩ tôi được làm việc tại một trong những công ty điện tử và phần mềm hàng đầu thế giới.

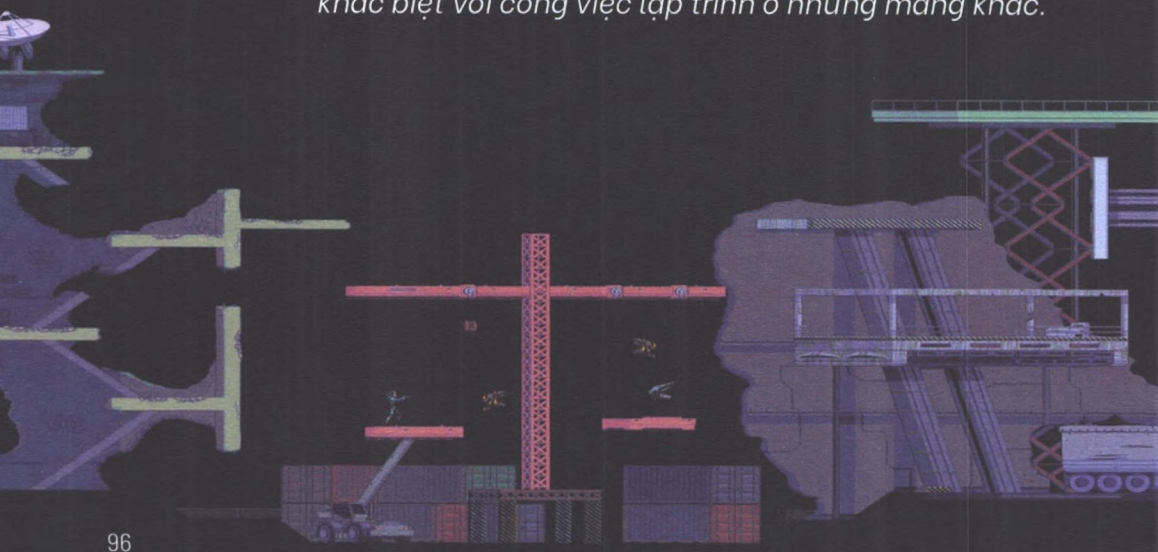
Tôi mong rằng nếu bạn có một công việc khiến bạn cảm thấy mình được sáng tạo, và có khả năng làm tốt, thì đừng ngại bắt đầu tìm hiểu và làm thử. Trong khi viết bài này, tôi nhìn ra tay bên phải thấy một chiếc router chạy Linux mình đóng góp viết, bên trái thấy chiếc điện thoại WebOS mình root[1]. Tôi nghĩ cuối cùng mình cũng làm tốt được một số việc và yêu một số thứ: những thiết bị nhỏ, tốn ít điện năng, đơn giản và làm được đúng việc mình cần. Đó chính là lập trình nhúng.

[1] *giành quyền kiểm soát hệ thống của thiết bị*

LÀM GAME sướng hay khổ

Tác giả: **Tống Tùng Giang**
Gameplay Programmer

Tên mình là Tống Tùng Giang, năm nay 25 tuổi, mình đã làm việc trong ngành game được 5 năm. Mình từng trải qua công việc ở Hiker Games - studio nổi tiếng với tựa game 7554 - Sống lại những ký ức hào hùng. Mình cũng từng làm việc ở studio game nhỏ hơn, làm qua mạng với một team quốc tế, hay làm những dự án cá nhân một mình. Hy vọng bài viết này sẽ cho các bạn góc nhìn sơ lược về ngành game, vai trò của một lập trình viên trong ngành và nó có gì khác biệt với công việc lập trình ở những mảng khác.



Mình đã đến với lập trình game như thế nào?

Những năm 2000, mình học cấp 1, nhà có cái đầu đĩa VCD của Trung Quốc. Hằng ngày, công dụng của nó là bật đĩa Xuân Mai cho bà ngoại dỗ em gái mình ăn cháo. Mình rất ít khi đụng vào cái đầu đĩa đấy, vì hầu hết thời gian mình ra ngoài đuổi nhau, đá bóng với bọn trẻ con cùng tuổi trong ngõ. Cho đến một ngày, mình phát hiện đầu đĩa có đi kèm hai cái tay cầm loại bốn nút, cùng với một đĩa điện tử 300 trò. Khỏi phải nói, với lũ trẻ ngày ấy, phát hiện đó không khác gì tìm ra kho báu. Thời đó, phương tiện giải trí không có nhiều, loanh quanh chỉ có tivi và sách báo. Sự khác biệt mà game mang lại tự dưng trở nên hấp dẫn hơn rất nhiều.

Khi mình lên cấp 2, bố mình được công trường cấp cho chiếc laptop IBM ThinkPad. Trước mình chỉ thấy máy tính để bàn toàn một màu trắng với cái màn CRT dày như tivi, giờ nhìn bố ngồi làm việc với cái máy tính xách tay đen tuyền bí ẩn, mình cảm thấy thèm muốn ghê lắm. Mỗi lần bố mình để máy ở nhà, mình lại nhảy vào táy máy đủ thứ trò, từ đơn giản như mở Paint ra vẽ rồi thay wallpaper, đến hì hục cài game vào trong máy, dĩ nhiên là trong sự lén lút, vì bố mình rất nghiêm, thấy con chơi điện tử là ăn roi luôn, không nói nhiều.

Năm mình học lớp 8, trường làng lần đầu tiên mở đội tuyển Tin học để đi thi học sinh giỏi, bên cạnh các môn truyền thống như Toán hay Lý. Mình với thằng bạn thân ngay lập tức nhảy từ đội tuyển Toán sang đội tuyển Tin với cùng niềm tin ngây thơ rằng vào đội tuyển Tin sẽ được... ngồi máy tính cả ngày. Mình hoàn toàn không ngờ rằng hành động bột phát đó là bước ngoặt lớn trong cuộc đời. Vào đội tuyển tin mình được học lập trình - hứng thú của mình với lập trình bắt đầu từ những tiết học đến 8h tối thời ấy.

Lần đầu tiên mình biết đến thuật ngữ "làm game" là năm lớp 11. Mình và thằng bạn thân, sau 2 năm gắn bó với đội tuyển ở trường làng cấp 2, tiếp tục đại diện trường cấp 3 đi thi cấp thành phố. Hồi ấy có 2 cuộc thi một năm: Cuộc thi do Sở Giáo dục tổ chức là cuộc thi kiểu nghiêm túc, dùng để tuyển chọn "gà" lên tham gia thi quốc gia, tổ chức ở trường Phan Đình Phùng - Hà Nội. Mình không khoái cuộc thi này lắm vì không có tham vọng thi giải quốc gia, chỉ tham gia để lên ngắm con gái trường Phan Đình Phùng (phải công nhận các bạn xinh thật, trường trên phố có khác). Cuộc thi thứ hai là Hội thi Tin học trẻ không chuyên, dành cho học sinh từ các trường cấp 3 không thuộc hệ thống trường chuyên (dân Tổng hợp hay Ams không bao giờ có mặt ở các cuộc thi này). Cuộc thi không chuyên vui hơn nhiều, chủ đề cũng rất sáng tạo. Năm đó, chủ đề là game bắn tàu. Đề bài mình nhớ mang máng là mỗi thí sinh có một số lượng cố định tàu to, tàu vừa và tàu bé. Thí sinh sắp xếp các tàu này vào một lưới kích thước cố định và phát triển thuật toán tấn công. Hội đồng sẽ sắp các thí sinh vào thành từng cặp đấu, cho hai bên bắn nhau theo lượt, bắn đến chết hết thì thôi. Hôm ấy, mình thắng đúng một trận rồi bị loại, nhưng mình bắt đầu thấy lập trình game có vẻ là cái gì đó rất vui.

Năm 2012, mình phải thi đại học. Năm ấy ngành kinh tế đi lên, ai ai cũng rủ nhau vào Kinh Tế, Ngoại Thương hay Ngân Hàng. Bố mẹ mình cũng muốn mình vào học khối ngành kinh tế, dù bố mình học Xây Dựng, mẹ mình học

Bách Khoa, cả hai đều là dân kỹ thuật. Lý do ông bà đưa ra là học kinh tế cho nhàn, lương lại cao, chứ làm thằng kỹ sư thì cả đời mày chỉ có làm thợ. Nhưng với sự bướng bỉnh của một thằng nhóc 18 tuổi chưa hiểu sự đời, mình vẫn quyết chí bỏ ngoài tai lời khuyên của bố mẹ. Thế là mình vào Đại học Bách Khoa Hà Nội, với ước mơ trở thành lập trình viên. Thời điểm ấy, mình vẫn chưa thực sự nghiêm túc với lập trình game, hay nói đúng hơn, mình cũng chưa biết rõ lập trình game là cái gì. Đến khi vào học các môn chuyên ngành, phải làm các đồ án môn học (bài tập lớn), mình bắt đầu lựa chọn các dự án game và gắn bó với nó cho đến bây giờ.

Vậy làm game thực sự như thế nào?

"Em đang học lớp 12, có hứng thú với lập trình game, không biết phải bắt đầu từ đâu, mong mọi người chỉ dạy."

"Em là sinh viên CNTT đang tập tành làm game, không biết mọi người hay làm đồ họa trong game bằng cái gì?"

"Em có thằng bạn thích làm game, mà nó không biết phải học ngôn ngữ lập trình gì, anh chị nào có kinh nghiệm cho em lời khuyên với ạ."

Mình gặp những câu hỏi như trên gần như hằng ngày. Mình cũng từng đặt những câu hỏi như vậy, chỉ khác là năm 2013 mình không có những group Facebook mà phải đem lên các forum như voz để hỏi. Trong phần này, mình sẽ "giải ngố" cho các bạn qua kinh nghiệm mình đã quan sát và học hỏi được sau vài năm trong ngành.

Đầu tiên, chúng ta sẽ nói về các vai trò trong ngành. Theo mình, có 4 nhóm vai trò chính: nhóm nội dung, nhóm sáng tạo, nhóm kỹ thuật và nhóm quản lý. Nhóm nội dung bao gồm những người sản xuất nội dung của game như đồ họa và âm thanh, ví dụ: các họa sĩ (modeler, animator, artist) hay các nhạc sĩ (sound designer, composer). Nhóm sáng tạo đảm nhiệm đủ thứ công việc nhằm đảm bảo trải nghiệm mà game mang lại độc đáo nhất có thể, từ mô tả kịch bản game, liệt kê các tính năng mà game cần có cho đến viết cốt truyện và cân bằng thông số. Nhóm này thường được gọi chung là game designer, đừng nhầm với graphic designer nhé, một bên là viết còn bên kia là vẽ, khác nhau đấy. Hai nhóm này cùng với nhóm kỹ thuật sẽ được theo dõi chặt chẽ bởi nhóm còn lại, nhóm quản lý (project manager, producer,...). Nhóm quản lý có nhiệm vụ đảm bảo mọi thứ diễn ra đúng với kế hoạch đề ra, cũng như làm việc trực tiếp với các bên liên quan mỗi khi có thay đổi. Theo quan sát của mình, việc này diễn ra thường xuyên.

Mình nói kỹ hơn về nhóm kỹ thuật. Nhóm này bao gồm 2 nhóm nhỏ, nhóm phát triển và nhóm đảm bảo chất lượng. Các lập trình viên thuộc nhóm đầu tiên, với đầu ra được giám sát bởi nhóm thứ hai gồm các tester và các chuyên viên QA (quality assurance). Nói một cách đơn giản, nhiệm vụ của chúng ta là xử lý các vấn đề kỹ thuật sao cho game xuất ra được, game chạy không bị lỗi và game chạy đủ "mượt".

Như các bạn thấy, có rất nhiều vai trò trong ngành game, tùy vào công ty và dự án mà một người có thể đảm nhận một hoặc nhiều vai trò. Với các dự án có quy mô nhỏ và nhân lực hạn chế, một người có thể đảm nhận nhiều vai trò: lập trình viên kiêm luôn vai game designer, hoặc một artist làm mọi thứ

từ dựng hình (modelling) đến diễn hoạt chuyển động (animating), và tất cả mọi người đều là tester. Trong khi đó ở các studio cỡ trung và cỡ lớn, việc phân vai trò theo nhóm thậm chí còn không đủ. Nếu bạn nhìn vào CV của một họa sĩ từng làm việc cho một studio như Naughty Dog[1] chẳng hạn, bạn sẽ thấy anh ta/cô ta có chức danh rất cụ thể, họa sĩ môi trường (environment artist) chẳng hạn.

Vậy với lập trình viên thì sao? Cũng tương tự, nếu ở studio nhỏ, bạn sẽ làm đủ mọi thứ từ việc to đến việc nhỏ. Nếu ở studio lớn, bạn sẽ đi rất sâu vào một mảng nào đó như phát triển công cụ (tools), mạng (networking), đồ họa (graphics), âm thanh, giao diện (user interface – UI), trí tuệ nhân tạo (artificial intelligence – AI), vật lý (physics),... Phát triển theo hướng chuyên môn hóa ở các ngành kỹ thuật cụ thể là con đường sự nghiệp tương đối phổ biến ở các lập trình viên, nhưng không phải duy nhất. Mình từng làm việc với các đàn anh có hướng đi khác. Có anh sau hơn 10 năm làm lập trình thì quyết định sử dụng khả năng phân tích vốn có cùng kinh nghiệm chơi game từ thời bé tí để chuyển sang nhóm game designer. Có anh lại nhảy sang mảng quản lý và rất xuất sắc ở vai trò mới. Có anh vẫn giữ vai trò kỹ thuật, nhưng lại ở một tầng tổng quát hơn với vị trí kiến trúc sư (architect) hoặc kỹ sư trưởng (lead engineer) thay vì đào sâu vào một mảng cụ thể.

Học gì để làm game?

Đọc đến đây, chắc các bạn có thể hình dung phần mềm nói chung và game nói riêng là những sản phẩm cực kỳ phức tạp và yêu cầu lượng kiến thức rất lớn. Vậy để làm lập trình game, bạn cần kiến thức gì?

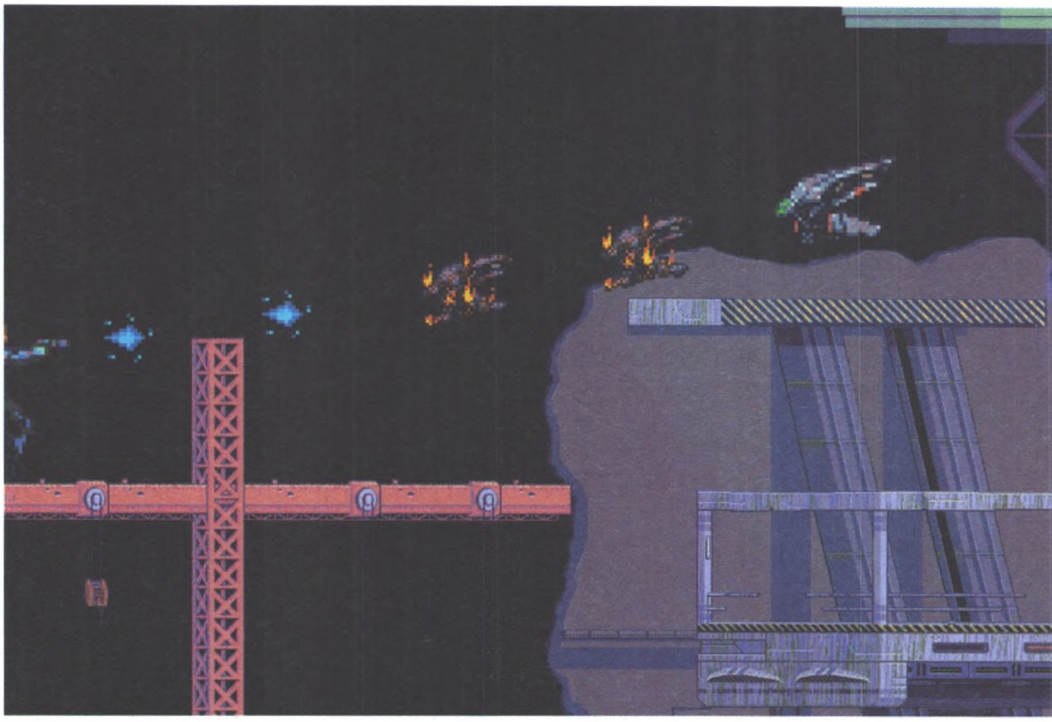
Đầu tiên, nếu có thể, các bạn hãy học Đại học. Gần đây mình thấy có xu hướng xem nhẹ việc học Đại học, nhất là trong ngành CNTT nói chung. Các trung tâm dạy nghề mọc lên như nấm với những quảng cáo hấp dẫn như: "Học 3 tháng có lương ngàn đô". Nghe rất hấp dẫn đúng không, nhưng toàn bánh vẽ mà thôi. Việc học Đại học sẽ trang bị cho các bạn kỹ năng tự học cùng một nền tảng kiến thức đủ vững để đối mặt với những thay đổi chóng mặt từ thế giới công nghệ.

Kể cả khi học Đại học, sẽ rất dễ để bạn sa vào các cám dỗ liên quan đến việc đuổi theo công nghệ mới. Đừng nóng ruột đốt cháy giai đoạn, hãy dành những năm đầu nắm thật vững các kiến thức cơ bản: Đại số, Xác suất Thống kê, Vật lý, Cấu trúc dữ liệu và giải thuật, Kiến trúc máy tính, Mạng máy tính, Lập trình hướng đối tượng. Tất cả những mảng kiến thức rời rạc ấy sẽ tạo thành một khối thống nhất khi bạn rèn luyện đủ nhiều sau một thời gian tích lũy kinh nghiệm, tạo tiền đề cho việc bạn học những thứ cao cấp hơn như Đồ họa máy tính hay Thiết kế kiến trúc phần mềm, hay những kiến thức đặc thù của những mảng mà bạn theo đuổi trong tương lai.

Khi bạn đã nắm vững các kiến thức cơ bản, việc học các công nghệ cụ thể sẽ rất dễ dàng. Ví dụ, các kiến thức về đồ họa máy tính có thể mang áp dụng từ Unity sang Unreal Engine[2], kiến thức về cấu trúc dữ liệu và giải thuật hoặc lập trình hướng

[1] *studio phát triển game tại Mỹ, nổi tiếng với các tựa game độc quyền cho hệ máy Playstation như Uncharted hay The Last of Us*

[2] *Hai công cụ phổ biến trong việc phát triển game trên thị trường hiện nay, do Unity Technologies và Epic Games phát triển*

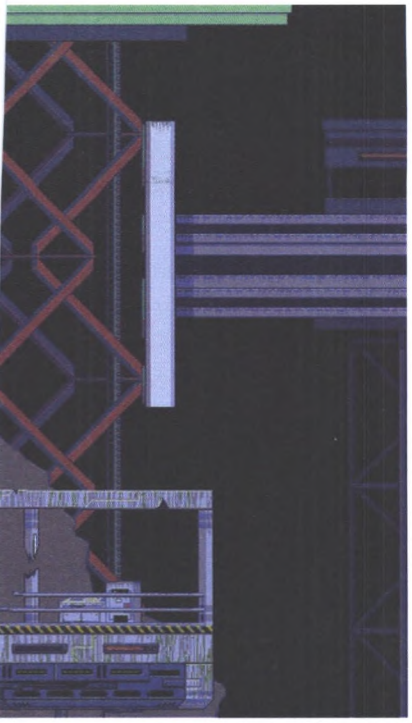


đối tượng vẫn y nguyên giữa C++ và C#, hay kiến thức toán khô khan mà bạn học trong môn Đại số và Xác suất sẽ là công cụ đắc lực cho bạn ở bất cứ tựa game nào. Trái với suy nghĩ của mọi người rằng công nghệ thay đổi liên tục, mình thấy các công nghệ mới vẫn được xây dựng dựa trên những kiến thức nền tảng này.

Cuối cùng, hãy thường xuyên theo dõi thông tin tuyển dụng của các game studio, ví dụ như như ở Hà Nội thì có Hiker, Zitga, OneSoft hay ở TP HCM thì có Glass Egg, Sparx*, Amanotes hay WolfFun. Việc này giúp bạn nắm được nhu cầu của thị trường là gì, yêu cầu kiến thức và kỹ năng cho vị trí mơ ước của bạn ra sao. Với nền tảng kiến thức trang bị từ trước, bạn hoàn toàn có đủ khả năng tự nghiên cứu và bồi đắp những mảng mình còn thiếu sót hoặc yếu kém.

Những thách thức của công việc lập trình game

Nhiều người vẫn hay đùa với mình rằng: "Làm game chắc sướng, suốt ngày được chơi". Đấy cũng là lý do khi mình đã học Bách Khoa, một lần nữa mình vấp phải sự phản đối của bố mẹ khi quyết định theo đuổi công việc lập trình game. Bố mẹ hoàn toàn không nghĩ rằng "làm game" là một công việc thực sự. Cũng khó mà trách được bố mẹ, vì thực ra quan điểm giống như bố mẹ mình ở xã hội Việt Nam không hiếm. Trong thực tế, làm game cũng là một công việc như bao công việc khác, cũng có lịch trình, cũng có áp lực, cũng có khó khăn. Nếu nghĩ làm game là chơi game thoải mái cả ngày thì thực sự sai lầm. Một khi đã đụng đến tiền, không có thứ gì có thể coi là làm cho vui được.



Lập trình game thực sự khó. Bạn phải liên tục chạy đua vì một khung hình chỉ có 16 ms (mili-giây) để chạy tất cả các thứ từ render cảnh 3D, hiển thị UI, xử lý input của người chơi, chạy logic cho AI, chạy các hiệu ứng hình ảnh và âm thanh,... Rồi phần cứng thay đổi, công nghệ thay đổi, nếu bạn làm một game trong vòng 2 năm, bạn phải tính cả đến tiêu chuẩn mà người dùng kỳ vọng ở game sẽ như thế nào sau 2 năm đó.

Tiếp theo là bug (các lỗi logic xảy ra trong quá trình chạy phần mềm), bất cứ mảng nào trong ngành phần mềm cũng phải đối diện với bug. Ngành game có những bug mà bạn không tin vào tai mình. Trong một dự án mình từng làm việc, đến gần ngày phát hành, tự dưng tester của dự án báo về một lỗi lạ. Khi mấy con bot đi được nửa đường, bỗng dưng nó bị ném đi đâu ở tít tận rìa của map và tắc tịt ở đấy. Bạn sẽ nghi ngờ tester báo cáo bậy bạ, nhưng kinh nghiệm cho mình thấy hầu hết đó là lỗi trong code của bạn.

Tuy nhiên, không giống các ngành phần mềm khác, ngành game rất thiếu quy trình và kế hoạch sản xuất chưa được tính toán chặt chẽ, nhất là ở các công ty nhỏ. Với một dự án phần mềm thông thường, yêu cầu và kiến trúc phần mềm sẽ được vạch ra chi tiết, nhưng với một dự án game, điều đó là không thể. Một dự án game thường bắt đầu với một nhúm các lập trình viên ngồi viết một bản chơi thử (prototype) để kiểm định tính khả thi của ý tưởng. Khi nào thấy ý tưởng (có vẻ) khả thi và chơi (có vẻ) vui, lúc ấy mới bắt tay vào sản xuất thật (production). Trong quá trình chuyển đổi giữa 2 giai đoạn, nhiều khi phải viết lại phần lớn game là chuyện bình thường.

Đấy là trong trường hợp prototype của bạn chơi hay. Hầu hết các prototype mình từng làm đều bị hủy bỏ trong vòng 2 tháng đầu tiên. Lý do rất đơn giản: Bạn không thể định nghĩa thế nào là game hay. Có những ý tưởng nghe rất hay ho trong đầu, nhưng khi thực sự cầm lên chơi lại thấy chúng chẳng thú vị lắm. Cá nhân mình, đau đớn nhất là dự án Arena of Survivors với Hiker Games. Dự án qua giai đoạn prototype, qua giai đoạn sản xuất, nhưng vẫn bị dừng sau 1 tháng open beta[1]. Thế mới thấy, đạt được thành công trong ngành game không đơn giản tí nào.

Nếu nghĩ làm game là chơi game thoải mái cả ngày thì thực sự sai lầm. Một khi đã đụng đến tiền, không có thứ gì có thể coi là làm cho vui được.

[1] *giai đoạn dùng thử với đông đảo người chơi trong thời gian nhất định*

Tại sao lại không đơn giản? Vì sự cạnh tranh trong ngành game thực sự rất kinh khủng. Thống kê cho thấy, năm 2018 có 9.000 game được ra mắt trên Steam, nghĩa là trung bình một tuần có hơn 170 game được ra mắt. Với các nền tảng di động như Android và iOS, con số thậm chí còn lên đến hàng trăm game một ngày. Cơ hội nào để game mà team bạn làm quần quật suốt cả năm trời được chú ý tới? Rồi khi người chơi đã chú ý tới game của team bạn rồi, liệu họ có sẵn sàng trả tiền mua game hoặc dành thời gian ra chơi?

Và cuối cùng là những cuộc chia tay. Ai tham gia ngành game cũng muốn được làm một thứ mà mình có hứng thú. Có người thấy dự án mới của team không hấp dẫn, dự án khác của một team khác có vẻ thú vị hơn. Có người thì thấy mệt mỏi với sự thay đổi liên tục của ngành game nên đi tìm thứ gì đó có thể đem lại sự ổn định. Có người lại chỉ đơn giản là đánh mất hứng thú. Những cuộc chia tay diễn ra thường xuyên trong bất cứ công ty nào mình từng làm việc. Sau khi đã làm việc cùng nhau được một thời gian, đã phát triển được một sự hòa hợp nhất định, thật khó để nói lời tạm biệt. Sau này, khi đã đứng ở cả hai bên, cả bên của người rời đi và người ở lại, mình bắt đầu học được cách chấp nhận đó là một phần của công việc.

Nhưng làm game thực sự rất thú vị

Năm cấp 2, khi gõ những dòng code đầu tiên, mình bắt đầu được trải nghiệm cảm giác phấn khích mỗi khi giải quyết xong vấn đề. Nó gần như chơi giải đố vậy. Hầu hết, các vấn đề trong lập trình sẽ liên quan đến việc bạn phải giải quyết một bài toán với ràng buộc về mặt thời gian hay bộ nhớ. Mình còn nhớ hồi làm Arena of Survivors, mình dành nguyên một buổi chiều tối ưu thuật toán Fog of War[1] để hiệu năng của nó giảm từ 15 ms xuống còn 2 ms. Sau những giờ tập trung cao độ, vật lộn với đám công thức, biến số và giá trị, nhìn thấy game chạy không còn giật nữa đúng là một liều kích thích tinh thần tuyệt vời. Đó là thứ làm cho công việc lập trình rất cuốn hút đối với mình, kể cả sau này nó dính với cơm áo gạo tiền chứ không đơn thuần là những tiết học trên lớp.

Như mình đã nói, lập trình game rất phức tạp. Mỗi dự án mình lại gặp phải rất nhiều vấn đề mới, hoặc có những vấn đề cũ nhưng yêu cầu một giải pháp mới vì cách giải cũ không còn phù hợp. Có những hôm mình không code dòng nào, chỉ lên mạng lục lọi và ngấu nghiến những tài liệu liên quan. Có những hôm mình xóa code cũ nhiều hơn là viết code mới. Có những hôm mình đến cửa công ty xong lại phải lộn về nhà kiếm mấy quyển sách cũ để dượt lại một phần kiến thức đã trót quên. Cơ hội học hỏi những điều mà mình chưa biết diễn ra hằng ngày trong ngành lập trình. Với lập trình game, có vô số thứ bạn có thể học: vật lý, đồ họa, tối ưu, kiến trúc phần mềm, kiến trúc máy tính,... Càng học mình càng nhận thấy kiến thức của mình ít ỏi như thế nào, để khi tiến bộ về một mảng nào đấy mình lại thấy vui.

[1] *thuật ngữ chỉ những màn sương che giấu hoạt động của kẻ địch hoặc các đơn vị của kẻ địch trong các tựa game theo lượt hoặc chiến thuật thời gian*

Làm game, có một cái vui nữa là mình hay được tiếp xúc đồ chơi mới. Đồ chơi này có thể là phần cứng - một thế hệ console[1] next-gen[2] chẳng hạn, hoặc một giải pháp phần mềm hoàn toàn mới. Mỗi lần tiếp xúc mình lại được mở mang tầm mắt rằng thế giới đã đi xa đến đâu, mình còn nhiều điều phải học thế nào, và sau cùng là mình thấy phấn khích. Với tất cả mấy thứ "ngầu đét" này, liệu mình có thể làm được gì với nó? Liệu mình có khả năng đạt được kết quả tương tự với giải pháp cũ và phần cứng cũ hay không? Những "đồ chơi" mới này liên tục thúc đẩy mình học hỏi và vượt qua giới hạn của bản thân.

Việc vượt qua giới hạn là điều rất cần thiết để bất cứ ai có thể trưởng thành, theo mình là vậy. Mình vẫn còn nhớ hồi đi phỏng vấn ở Emobi Games (tiền thân của Hiker Games) với tư cách là sinh viên thực tập, anh Huy, CEO của Emobi, có hỏi mình một câu về mục tiêu trong sự nghiệp của mình. "Em muốn tham gia vào làm một game như 7554." - mình trả lời ngắn gọn. Anh Huy cười. Có lẽ sau 6 - 7 năm trong ngành, anh đã quá quen với những câu trả lời như thế của bất cứ ai tham gia phỏng vấn, vì đây là Emobi, tác giả của 7554 cơ mà. Anh trả lời: "Làm như 7554 thì thường quá, thế hệ bọn anh đã làm rồi. Thế hệ của em phải làm được hơn thế". Sau này, trong suốt quãng thời gian làm việc ở Hiker Games, mình thực sự chứng kiến tinh thần ấy của toàn bộ studio, khi mọi người đều góp sức để làm nên một cái gì đó mới mẻ và thú vị.

Kết

Vậy đấy, làm game có quá nhiều thứ hay ho với mình. Có lẽ mình sẽ được nhiều người nể trọng hơn nếu làm một thầy giáo, sẽ ổn định hơn nếu làm kế toán cho một doanh nghiệp nhà nước, hay thậm chí là giàu hơn nếu vẫn làm lập trình ở mảng phần mềm khác. Nhưng mình không đánh đổi công việc làm game của mình cho bất cứ thứ gì ở trên.

Và còn một điều nữa, mình để lại đến cuối cùng. Cảm giác này là cảm giác tuyệt vời nhất mà bạn có thể có được khi làm việc trong ngành game. Sau khi trải qua cả đống dự án thất bại, rồi một thời gian dài đằng đẵng cày cuốc, những đêm OT đến tối muộn, những ngày về nhà trong sự uể oải, cuối cùng game của bạn cũng ra lò. Bạn liên tục vào xem thông số để biết game của bạn có bao nhiêu lượt tải. Bạn vào đọc từng comment xem người chơi nghĩ gì về công sức mà team bạn bỏ ra. Bạn lân la rình mò các tờ báo đưa tin về game với cảm giác hí hửng xen lẫn tự hào về thành quả lao động của bạn. Bạn hăm hở rủ bạn bè ra quán nước, chỉ để khoe game của bạn và bắt chúng nó chơi.

Nếu như bạn thấy thích thú với những trải nghiệm mà mình vừa kể, cũng như thấy những thách thức không đáng sợ lắm, có lẽ bạn hãy thử sức đi thôi. Biết đâu tương lai chúng ta sẽ là đồng nghiệp?

[1] *thiết bị chơi game*
[2] *thế hệ kế tiếp*

TRÍ TUỆ NHÂN TẠO
Cầu Nối Giữa
CON NGƯỜI & MÁY TÍNH

Khách mời: **Tiến sĩ Nguyễn Quang Uy**
AI Research Scientist, GRDAI, VNG
Biên soạn bài viết: **Ban biên tập**

Tháng 3/2016, giới công nghệ rúng động khi Lee Se-dol - nhà vô địch cờ vây Hàn Quốc với sự nghiệp lẫy lừng hơn 20 năm, từng chiến thắng hàng chục kỳ thủ hàng đầu thế giới - đã thua cuộc khi đấu với chương trình chơi cờ vây AlphaGo của Google. Sau trận đấu, anh nói: "Tôi vô cùng ngạc nhiên. Tôi không nghĩ mình sẽ thua nhưng không ngờ AlphaGo có thể chơi một ván đấu hoàn hảo đến vậy".

Đây được coi là bước phát triển đột phá của trí thông minh nhân tạo bởi cờ vây có sự thiên biến vạn hóa vô cùng phức tạp về chiến thuật. Vậy câu chuyện đằng sau những cỗ máy với "bộ não" tuyệt vời như vậy là gì?

Trí tuệ nhân tạo không còn là khoa học viễn tưởng

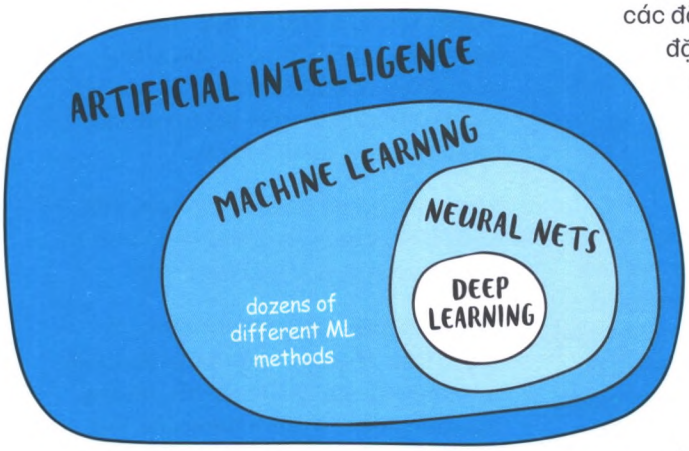
Trí tuệ nhân tạo, hay còn gọi là AI (Artificial Intelligence), thoạt nghe có vẻ phức tạp và... xa vời. Đến nay, đây vẫn là chủ đề mà nhiều nhà khoa học tranh luận, tùy từng trường phái mà có nhiều định nghĩa xoay quanh. Ví dụ, có quan điểm cho rằng: "Phàm thế nào để máy tính thông minh như con người, đấy là AI" hay "AI khiến máy móc bắt chước một hành vi nào đó của con người".

Các nhà khoa học cũng tranh cãi rất nhiều về chuyện AI đến ngày nào đó thay thế loài người? Và thậm chí quay lại tấn công chính chúng ta? Quan điểm của tôi là con người sẽ tiếp tục nghĩ ra các thuật toán mới, các hạ tầng tính toán không ngừng được cải tiến, cho phép xây dựng những thuật toán AI ngày càng tốt hơn. Nhưng khó có thể khẳng định chắc chắn về viễn cảnh AI sẽ thay thế hoàn toàn con người.

Vậy thực sự AI là gì?

Có thể hiểu đơn giản: AI là những gì làm cho máy tính trở nên thông minh và giải quyết được những công việc mà bình thường phải cần đến con người. Cụ thể hơn, trong AI có lĩnh vực thị giác máy tính giúp máy tính có thể nhìn, nhận biết màu sắc, chiều dài, chiều rộng, lĩnh vực nhận dạng âm thanh giúp máy tính có thể giao tiếp ngôn ngữ, chat, trò chuyện với nhau, nghe, hiểu, nhận dạng tiếng nói,...

Gần đây, khi đề cập đến AI, ta cũng thường nghe đến các thuật ngữ Machine Learning (ML), Artificial Neural Networks (ANN) và Deep Learning (DL), có thể phân biệt chúng một cách cơ bản như sau[1]:



AI: là tên gọi chung của cả lĩnh vực nghiên cứu, tương tự như cách ta gọi sinh học, hoá học,...

ML (Máy học): là một phần quan trọng (nhưng không phải tất cả) của AI. ML hướng đến câu chuyện làm thế nào để qua thời gian, máy tính có thể học và trở nên thông minh như loài người. Mục tiêu của ML là dự đoán các kết quả dựa trên các dữ liệu đầu vào. Vì thế, ML cần 3 nguyên liệu chính:

1. Dữ liệu (Data): Dữ liệu thu được càng đa dạng, kết quả đưa ra càng chính xác. Việc thu thập được một tập dữ liệu tốt không hề đơn giản, vì thế các công ty rất coi trọng dữ liệu (một số công ty có thể tiết lộ về thuật toán của họ, nhưng dữ liệu thì không).

2. Đặc tính (Features): Còn được gọi là tham số (parameters) hoặc các biến (variables). Một vài ví dụ về đặc tính như "số km đã đi của chiếc ô tô", "giới tính khách hàng", "giá cổ phiếu",... Đây là những yếu tố mà máy tính sẽ xem xét. Trong quá trình thực hiện ML, việc lựa chọn ra các đặc tính đúng với bài toán đặt ra ở thực tế tốn khá nhiều thời gian. Nó cũng là nguyên nhân chính gây ra sai sót bởi con người dễ bị chủ quan và lựa chọn những đặc tính mà họ thích hoặc cho rằng quan trọng.

3. Thuật toán (Algorithms): Hiển nhiên đây là yếu tố không thể thiếu. Mỗi bài toán đặt ra lại có nhiều lời giải khác nhau. Việc bạn lựa chọn giải pháp nào sẽ ảnh hưởng đến tính chính xác, mức độ hiệu quả và quy mô của mô hình. Tuy nhiên, cần lưu ý, nếu dữ liệu của bạn không "tốt" (có thể là quá ít hoặc đặc tính không liên quan đến yêu cầu đề bài) thì thuật toán tốt nhất cũng trở thành vô dụng. Vì thế, hãy cố gắng thu thập dữ liệu thật tốt trước khi tập trung vào độ chính xác của thuật toán.

[1] *tham khảo bài viết Machine Learning for Everyone: https://vas3k.com/blog/machine_learning*

ANN (Mạng neuron nhân tạo): là một trong những nhóm thuật toán học máy phổ biến nhất (nhưng không phải duy nhất, còn rất nhiều các thuật toán khác để lựa chọn) được xây dựng dựa trên cấu trúc mạng nơ ron của con người.

DL (Học sâu): là phương pháp học máy hiện đại và tiên tiến nhất hiện nay. DL được xây dựng dựa trên mạng nơ ron nhiều lớp với những kiến trúc mạng được thiết kế đặc thù cho từng lớp bài toán. Cụ thể hơn, có thể hiểu DL là mạng nơ ron nhân tạo nhưng có 3 thay đổi quan trọng: Thay đổi về kiến trúc mạng, thay đổi về thuật toán học và thay đổi về hàm kích hoạt trong mạng.

AI "cao siêu" như vậy liệu có dành cho quốc gia nhỏ bé như Việt Nam?

Trên thế giới, người ta liên tục nói về cách mạng công nghiệp 4.0. Nó bao hàm nhiều lĩnh vực, có IoT (Internet vạn vật), Blockchain, in 3D, công nghệ nano, công nghệ sinh học,... và AI có thể coi là bộ não của cuộc cách mạng này. Các quốc gia đang đầu tư vào nghiên cứu AI rất mạnh mẽ. Trung Quốc có những chương trình vài trăm tỷ đô từ nay đến 2025 để nghiên cứu AI. Ở Mỹ, họ thậm chí không gọi là cách mạng công nghiệp 4.0 mà gọi là "Kỷ nguyên trí tuệ nhân tạo".

Các nhà khoa học vẫn nói vui với nhau: "Hãy tìm cho tôi một lĩnh vực nào đó mà không bị ảnh hưởng bởi AI". Trong thời đại toàn cầu không còn chịu ảnh hưởng quá nhiều bởi khoảng cách địa lý, chúng ta không tìm đến thế giới thì thế giới cũng tìm đến chúng ta. Vì thế, tôi cho rằng tương lai 10 - 20 năm tới, AI sẽ ảnh hưởng rất nhiều đến đủ mọi lĩnh vực cũng như các hoạt động đời sống xã hội Việt Nam. Ví dụ như hệ thống xác thực người dùng, các cây xăng, trạm thu phí,... có thể hoàn toàn tự động. Thậm chí nay mai có thể ứng dụng nhiều quy trình khác như tự động đánh giá học sinh, xe tự lái,... Hiện tại, công việc của tôi và các cộng sự tại GRDAI[1] đang làm chính là tập trung ứng dụng AI vào những bài toán của Việt Nam và cung cấp cho cộng đồng, như: nhận dạng tiếng nói tiếng Việt chuyển thành văn bản; tổng hợp từ văn bản chuyển sang tiếng nói; đọc thông tin từ các chứng minh thư Việt Nam; nhận dạng biển số xe, hộ chiếu Việt Nam,... Với đội ngũ kỹ sư giàu kinh nghiệm và sự hỗ trợ về mặt nguồn lực đến từ VNG, tôi tự tin các sản phẩm của GRDAI có đủ tiềm năng để tạo nên đột phá trong việc giải quyết các bài toán thực tiễn tại Việt Nam.

Bên cạnh tương lai đầy hứa hẹn, Việt Nam cũng còn những khó khăn trong câu chuyện đưa AI vào thực tiễn.

Thứ nhất, các giải pháp sử dụng AI trong thực tế được những tập đoàn lớn trên thế giới giải quyết rất tốt, nhưng bài toán đặc thù dành riêng cho Việt Nam chưa được họ đầu tư nhiều. Việt Nam có các công ty lớn như Viettel, FPT, hay một số startup có xây dựng các dịch vụ về AI. Tuy nhiên, chưa công ty nào cung cấp hệ thống dịch vụ đầy đủ như hệ thống của Google, Amazon hay Microsoft,... mà chỉ cung cấp đơn lẻ một vài dịch vụ nào đó. Như Viettel có nhận dạng tổng hợp tiếng nói, FPT có nhận dạng tiếng nói và đọc thông tin từ Chứng minh thư (CMT),...

[1] *Cung cấp dịch vụ và sản phẩm được phát triển bởi nhóm nghiên cứu và phát triển dựa trên nền tảng trí tuệ nhân tạo, thuộc công ty VNG (grdai.vn)*

Thứ hai, khi triển khai, người sử dụng chưa hiểu đúng về AI. Họ luôn kỳ vọng và cho rằng AI hoàn hảo, không có sai số. Một điều rõ ràng là AI học các hành vi của con người. Vậy muốn AI đúng tuyệt đối đồng nghĩa với việc con người cũng phải tuyệt đối đúng. Trong thực tế, điều này là không thể. Tất nhiên, ở một số nghề cụ thể, có thể AI thực hiện tốt hơn con người, nhưng thông thường các thuật toán AI sẽ đạt đến ngưỡng xấp xỉ bằng con người, và dừng lại ở đó không cải tiến được thêm nhiều nữa. Sai số của AI tất nhiên không thể tránh được việc gây ra hậu quả, như có thể xác thực sai người dùng, để lọt một người nào đó vào ngân hàng gian lận đánh cắp thông tin, hoặc dùng AI chẩn đoán bệnh, nếu sai cũng rất nguy hiểm cho bệnh nhân.

Thứ ba đó là sự khan hiếm của dữ liệu để xây dựng các hệ thống AI. Chúng ta đều biết rằng, tất cả các hệ thống AI được xây dựng dựa trên dữ liệu do con người cung cấp. Dữ liệu càng nhiều càng có nhiều cơ hội để xây hệ thống AI thông minh hơn, đáp ứng tốt hơn yêu cầu thực tế. Tuy nhiên, thu thập và gán nhãn dữ liệu là công việc rất khó khăn và tốn kém. Thứ nhất, trong rất nhiều hệ thống, chẳng hạn như tài chính, ngân hàng hay an ninh quốc phòng, do tính nhạy cảm của dữ liệu, các nhà phát triển AI có rất ít cơ hội để sử dụng các dữ liệu để phát triển các mô hình AI. Thứ hai, ngay cả khi dễ dàng thu được dữ liệu người dùng (như lĩnh vực nhận dạng và tổng hợp tiếng nói, chúng ta có thể dễ dàng thu thập được dữ liệu từ các trang web hay kho dữ liệu video YouTube), việc gán nhãn dữ liệu (tức là ánh xạ giữa âm thanh và văn bản) đòi hỏi chi phí rất lớn, không dễ gì thực hiện được.

Vậy làm thế nào để dung hòa được sai số đó với nhu cầu sử dụng trong thực tiễn vốn luôn đòi hỏi tính đúng đắn? Điều này không hề đơn giản. Nó đòi hỏi người phát triển sản phẩm AI và người sử dụng cùng trao đổi với nhau, xem xét và bàn bạc nhiều yếu tố như khả năng sản phẩm, quy trình thực tế,... từ đó, đưa ra giải pháp áp dụng AI vào bước nào, chỉnh sửa cải thiện và kiểm tra ngẫu nhiên tính chính xác của AI ra sao,... Nhìn chung, hiện nay AI hầu hết được sử dụng như một công cụ trợ giúp thêm, còn quy trình cuối cùng vẫn phải do con người thực hiện.

Dù vậy, những vấn đề trên cũng mở ra cơ hội rất lớn cho các bạn trẻ đang muốn tham gia vào lĩnh vực AI. Bởi đến với một "mảnh đất còn sơ khai", hiển nhiên các bạn sẽ có nhiều "cơ hội khai thác". Năm 2019, Liên hiệp các cộng đồng AI ở Việt Nam chính thức ra mắt, với mục tiêu tạo ra và đẩy mạnh phát triển hệ sinh thái trí tuệ nhân tạo. Các đại diện ra mắt bao gồm: Câu lạc bộ Khoa - Trường - Viện Công nghệ Thông tin - Truyền thông Việt Nam FISU; Cộng đồng nghiên cứu, triển khai và ứng dụng Trí tuệ nhân tạo AI4Life; Cộng đồng Chuyển đổi số - Digital Transformation; Cộng đồng Machine Learning Cơ bản; Cộng đồng Google Developer; Cộng đồng Business Intelligence; Cộng đồng VietAI - Trí tuệ nhân tạo Việt và Tiến sĩ Cao Anh Tuấn - nhà sáng lập và CEO Công ty Genetica. Tôi nghĩ rằng người Việt có thiên hướng, khả năng làm tốt trong lĩnh vực này và đã tới lúc bước vào giai đoạn tương đối "chín" để chúng ta đưa AI lan sâu rộng trong thực tế.

Doanh nghiệp "đỏ mắt" tìm nhân lực ngành AI

So với mặt bằng chung, nhóm kỹ sư chuyên môn về AI có mức lương trung bình cao hơn các vị trí khác trong ngành CNTT. Tại GRDAI, các bạn mới tốt nghiệp ra trường có chuyên môn tốt có thể nhận mức lương khoảng 1.000 USD/tháng. Tuy mức lương tốt như vậy, các doanh nghiệp đang trong tình trạng thiếu nhân lực, không tuyển được người đủ tốt để làm việc.

Theo đánh giá của Google Brain, thế giới cần 1 triệu nhân lực trong lĩnh vực AI, nhưng hiện tại mới có khoảng 10.000 nhân lực chất lượng cao. Trong "Ngày hội Trí tuệ nhân tạo Việt Nam" tổ chức năm 2019, những người tham dự đã ngạc nhiên trước thông tin nguồn nhân lực AI hiện nay chỉ đáp ứng được 10% nhu cầu của thị trường. Dựa trên dữ liệu nửa đầu năm 2019 của trang tuyển dụng VietnamWorks, các kỹ sư trí tuệ nhân tạo được tuyển dụng với mức lương cao nhất (2.000 USD/tháng) và nhu cầu nhân lực AI dẫn đầu về tốc độ tăng trưởng (tăng 46% so với năm 2017). Tháng 1/2019, Nexus Frontier Tech công bố báo cáo "Toàn cảnh Trí tuệ nhân tạo tại Việt Nam năm 2018", trong đó, vấn đề thiếu hụt nhân tài là 1 trong 3 thách thức lớn nhất để phát triển công nghệ AI tại Việt Nam (chiếm 59%).

Hiện nay, số lượng các trường Đại học Việt Nam đào tạo AI còn rất ít. Năm 2018, Đại học Công nghệ Thông tin ĐHQG HCM đào tạo cử nhân ngành này với 50 sinh viên. Năm 2019, Đại học Bách Khoa Hà Nội mở ngành cử nhân chất lượng cao "Khoa học dữ liệu và Trí tuệ nhân tạo", đào tạo 40 sinh viên trong 5 năm. Ngoài ra, các trường cũng có đào tạo Thạc sĩ như Đại học Khoa học Tự nhiên ĐHQG HN (30 người), Đại học Bách Khoa Hà Nội (15 người) và Viện John von Newman ĐHQG HCM (35 người). Một số trường Đại học khác cũng đã xây dựng chương trình học về lĩnh vực này.

Tuy vậy, những sinh viên cũng cần 3 - 5 năm nữa mới tốt nghiệp ra trường, trong khi nhu cầu hiện nay là rất lớn. Tôi dự đoán ngành AI sẽ bùng nổ trong những năm tới.

Cần chuẩn bị gì để tham gia lĩnh vực AI?

Về định hướng phát triển, các bạn sẽ có 2 lựa chọn. Một là đi theo hướng phát triển sản phẩm về AI, hai là đi theo hướng nghiên cứu sâu hơn. Nhóm làm phát triển sản phẩm sẽ sử dụng các kết quả của nhóm nghiên cứu để tạo ra ứng dụng. Sau khi ra trường, các bạn có thể làm ở các phòng, trung tâm nghiên cứu phát triển và các doanh nghiệp với đa dạng những vị trí như: Kỹ sư phát triển ứng dụng AI; Chuyên gia nghiên cứu về trí tuệ nhân tạo; Kỹ sư dữ liệu; Kỹ sư phát triển hệ thống tự động hóa; Nghiên cứu ở các viện nghiên cứu, các trường Đại học,... Cơ hội học tiếp chuyên sâu ở nước ngoài cũng rất cao bởi hiện nay có khá nhiều học bổng dành cho nghiên cứu AI.

Nếu bạn muốn lựa chọn hoạt động về AI, tôi xin chia sẻ 5 kinh nghiệm rút ra từ bản thân, hy vọng giúp bạn cân nhắc định hướng rõ ràng hơn:

1. Có đam mê, thiên hướng về lĩnh vực này.

2. Giỏi lập trình do công việc phải sử dụng code nhiều.

3. Tư duy logic hoặc thuật toán tốt. Bạn không cần phải quá xuất sắc như đạt giải quốc gia hay quốc tế, nhưng nếu không tốt về toán, bạn sẽ gặp khó khăn trong việc học AI.

4. Giỏi công nghệ để tìm hiểu, sử dụng lại những thành tựu đã có trên thế giới.

5. Sự sáng tạo. Những công việc khác trong ngành CNTT thường có quy trình chuẩn (như làm web có framework), riêng AI thì không. Khi có vấn đề phải giải quyết, đầu tiên, bạn cần thử nghiệm những nghiên cứu sẵn có trên thế giới, đánh giá xem mức độ đáp ứng của nó như thế nào. Hầu hết những hệ thống có sẵn mà thế giới nghiên cứu và công bố sẽ không đủ tốt để đáp ứng bài toán thực tế. Vậy bạn cần lao vào để cải thiện, thay đổi sao cho phù hợp với bối cảnh. Đây chính là tính sáng tạo đầy thú vị nhưng cũng rất thách thức của lĩnh vực AI. Bản thân tôi cùng đội ngũ khi triển khai các giải pháp trong thực tế đều có sự sáng tạo trong sản phẩm. Ví dụ như việc xác thực người dùng, chúng tôi đã cải thiện giúp hệ thống không chỉ đọc thông tin từ CMT, hộ chiếu, mà còn có khả năng kiểm tra tính hợp lệ của những giấy tờ đấy, xem chúng có bị mất góc, hết hạn, bị mờ, chói,... hay không. Hoặc hệ thống nhận dạng khuôn mặt, phần sáng tạo lớn nhất là cải tiến để hệ thống có thể phát hiện người lạ (có thể là khách hoặc một ai đó xâm nhập vào).

Chinh phục những nấc thang mới trong lĩnh vực công nghệ là cả chặng đường dài, các bạn trẻ hãy rèn luyện và giữ gìn sức khỏe, luôn giữ được trái tim và bộ não nhiệt huyết, khỏe mạnh để bền bỉ bước trên con đường này.

Tại các trường Đại học đã mở chuyên ngành "Trí tuệ nhân tạo", các bạn sẽ được học nhiều môn học liên quan đến AI: khai phá dữ liệu, trí tuệ nhân tạo, học máy, thị giác máy tính, xử lý ngôn ngữ tự nhiên,... Đó đều là những môn tạo nền tảng rất tốt.

Tuy nhiên, tôi muốn lưu ý khả năng tự học là vô cùng cần thiết vì AI phát triển rất nhanh, chỉ 1 - 2 năm không cập nhật bạn có thể bị tụt hậu. Trên mạng hiện nay có nhiều khóa học về AI (cả miễn phí và trả phí) của các trường Đại học danh tiếng trên thế giới như Stanford, Oxford, MIT,... Nguồn tài liệu trên internet cũng rất nhiều. Hãy khai thác tốt các kênh đó trong quá trình tự học.

Bên cạnh việc học lý thuyết, một điểm quan trọng nữa là bạn hãy ngồi code, chạy thử chương trình trên dữ liệu thật. Việc này giúp bạn hiểu cặn kẽ được các thuật toán, mô hình, làm chủ được công nghệ và hiểu các yếu tố trong mô hình gây ảnh hưởng đến kết quả như thế nào. Lĩnh vực AI rất rộng nên bản thân mỗi người khó có thể làm chủ được hết các công nghệ. Trong mỗi giai đoạn lại có các xu hướng khác nhau, nên tùy từng thời điểm bạn sẽ quyết định tập trung vào công nghệ gì.

Song song với học ngôn ngữ lập trình phổ biến nhất dành cho AI là Python, các bạn đừng bỏ qua tiếng Anh. Ngoài việc giúp bạn tự tìm, đọc và học các tài liệu chuyên ngành, nó còn mở thêm cho bạn cơ hội việc làm ở những công ty nước ngoài.

Sau cùng, tôi muốn nhắn nhủ với các bạn trẻ: Chinh phục những nấc thang mới trong lĩnh vực công nghệ là cả chặng đường dài, các bạn trẻ hãy rèn luyện và giữ gìn sức khỏe, luôn giữ được trái tim và bộ não nhiệt huyết, khỏe mạnh để bền bỉ bước trên con đường này.

NGƯỜI LÀM SẢN PHẨM

"Ít tiếng" nhưng "có miếng"

Tác giả: **Husky - Product Owner**

Nếu phải tóm gọn những năm tháng làm ở vị trí Product (sản phẩm), mình sẽ dùng từ: transformation. Transformation dịch ra không chỉ là thay đổi thông thường, mà là sự thay đổi toàn diện về bản chất, từ đó giúp một người vươn tới được vị trí cao hơn trong cuộc sống. Công việc mình làm không chỉ mang lại giá trị cho khách hàng mà còn thay đổi cả con người mình, cách mình nhìn nhận về mọi thứ xung quanh. Nếu bạn là người dám chấp nhận thay đổi, chấp nhận sáng tạo bất chấp rủi ro thì vị trí Product sẽ khiến bạn rất thỏa mãn vì tìm được ý nghĩa trong công việc cũng như trong cuộc sống.

Thật ra, từ lúc còn học Đại học đến khi ra trường, mình không hề biết đến vị trí Product trong công ty công nghệ. Mình học về hệ thống quản lý thông tin và xác suất thống kê. Mình nghĩ khi ra trường nếu không làm ở vị trí phân tích dữ liệu thì ít nhất cũng là một kỹ sư công nghệ thông tin. Mình nghĩ đến ngôn ngữ lập trình cần học, các thuật toán cần biết, các cách vẽ đồ thị, các mô hình xác suất thống kê cần phải thông thạo. Mình xác định phải thông thuộc từ các công cụ cơ bản như Excel cho đến các công cụ nâng cao như Apache Spark.

Có rất ít người nói về vị trí Product trong công ty. Trước khi đi làm, bạn cũng nghe ít người nhắc đến nó so với những vị trí khác như kỹ sư full stack, nhà khoa học dữ liệu. Và ngay cả khi đi làm và bắt đầu gặp nhiều người trong ngành, bạn sẽ thấy vị trí Product thường không phải chủ đề mọi người hay bàn tán. Dường như ai cũng nhận thức có vị trí này trong một công ty công nghệ, nhưng thực sự ít ai hình dung được người làm Product đóng vai trò gì.

Nhưng mặc cho không được thường xuyên nhắc đến và ca tụng như các vị trí khác, mình có thể nói dựa trên kinh nghiệm của bản thân: *Người làm product là yếu tố then chốt giúp công ty thăng hoa hay thất bại.* Marty Cagan, một trong những người làm product tài ba nhất ở Thung lũng Silicon, đã giải thích điều đó như sau: "Tất cả mọi công ty đều phụ thuộc vào khách hàng. Và thứ khách hàng mua hay lựa chọn là sản phẩm của công ty đó. Sản phẩm của công ty được xây nên bởi đội ngũ kỹ sư và thiết kế, và người quyết định xem đội ngũ kỹ sư và thiết kế phải xây cái gì chính là người làm sản phẩm."

Cơ duyên đưa mình đến với sự nghiệp làm sản phẩm là khi bắt đầu bằng công việc Business Analyst (BA) lúc mới ra trường, sau đó chuyển sang vị trí Product Owner (PO). Hai vị trí này mặc dù có nhiều điểm tương đồng nhưng bản chất là khác nhau.

Một người làm tập trung vào Product có nhiệm vụ tối đa hóa giá trị của sản phẩm. Người đó đảm bảo các giải pháp được thực hiện sẽ cung cấp giá trị cho khách hàng đồng thời phù hợp với tầm nhìn của công ty. Các nhiệm vụ chính của người làm product là xác định hướng đi cho sản phẩm, lên danh sách các việc cần làm (product backlog), liên tục xem xét mức độ ưu tiên các việc cần làm, theo sát quá trình triển khai giải pháp đồng thời dự đoán trước nhu cầu khách hàng.

Còn người ở vị trí BA tập trung vào thu thập các yêu cầu của khách hàng, có thể gợi ý, lên yêu cầu hay đề xuất các giải pháp mà họ nghĩ khách hàng sẽ cần. Sau đó, người làm BA sẽ tổng hợp các thông tin có được thành một bộ tài liệu chi tiết và chuyển giao cho nhóm phát triển, bao gồm Project Manager (PM), Engineering (kỹ thuật) hay QC (kiểm soát chất lượng). Ngoài ra, người BA còn phải theo sát nhu cầu khách hàng để liên tục cập nhật những thay đổi trong yêu cầu của họ.

Như vậy, người BA tập trung vào yêu cầu của khách hàng, còn PO tập trung vào hoàn thiện sản phẩm để đáp ứng nhu cầu đó. Ở những công ty lớn, có thể 2 vị trí BA và PO tồn tại song song, nhưng ở các công ty nhỏ (như các startup quy mô 20 - 30 người, PO thường kiêm luôn cả vai trò BA).

Mình hiện đang làm về Product cho một startup ở Sài Gòn, chuyên cung cấp các giải pháp cho các quỹ đầu tư và các công ty luật trong lĩnh vực thương mại. Ban đầu khi thấy chữ "analyst" (phân tích), mình cho rằng sẽ đụng đến nhiều số liệu để giải quyết các vấn đề của khách hàng. Nhưng hóa ra đó là các phân tích định tính, tức phân tích nhu cầu của khách hàng nhiều hơn những con số. Mình dành phần lớn thời gian viết tài liệu, họp với các nhóm khác nhau trong công ty, vẽ quy trình, trò chuyện với khách hàng, tham khảo ý kiến của người dùng, xem các sản phẩm khác của đối thủ.

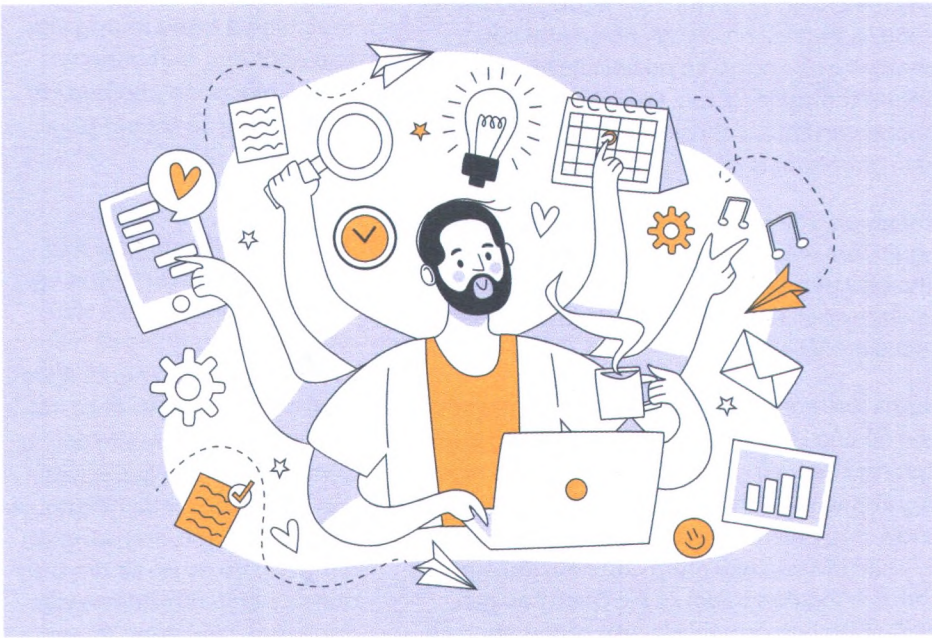
Mình nhận ra mình làm rất nhiều việc không tên, và làm với nhiều nhóm khác nhau. Phần lớn thời gian mình không trực tiếp làm ra sản phẩm. Đội thiết kế sản phẩm đảm trách thiết kế; đội kỹ sư xây nên sản phẩm bằng những dòng mã; đội QA (kiểm định chất lượng) kiểm tra những dòng mã xem có lỗi không, có đúng yêu cầu đề ra không; đội phát triển doanh nghiệp hay marketing trực tiếp quảng cáo sản phẩm cho khách hàng. Mình đã tự hỏi:

"Vậy người làm sản phẩm rốt cuộc là làm gì?"

Phải mất cả năm trời để mình thực sự tìm ra được câu trả lời cho vị trí đó.

Có rất nhiều người, ngay cả những người lâu năm trong ngành, vẫn tin sản phẩm của họ sẽ thành công nếu nó có công nghệ tân tiến hơn các sản phẩm của đối thủ. Thực tế, một sản phẩm chỉ thành công nếu nó cung cấp thứ mà khách hàng cần. Và để biết được rốt cuộc khách hàng cần gì, công ty sẽ cần một người phỏng vấn khách hàng, chắt lọc ra nhu cầu của họ, viết thành tài liệu hoàn chỉnh, từ tài liệu đó liệt kê danh sách những thứ cần phải làm, đồng thời mô tả chúng phải được làm như thế nào. Và vì một công ty (thường) luôn bị hạn chế bởi nhân lực, thời gian cũng như tài chính, người đó sẽ kiêm nhiệm vụ xác định trong danh sách những việc cần làm, cả nhóm phải ưu tiên làm thứ gì đầu tiên.

Người làm những việc đó chính là người làm Product.



Mặc dù nghe thật đơn giản nhưng việc xác định đúng nhu cầu của khách hàng và cung cấp giải pháp cho nhu cầu đó là vô cùng khó khăn. Hãy nghĩ đến những gã khổng lồ với các sản phẩm của họ. Ta có Google với một rừng các sản phẩm thất bại tiêu biểu như: Google+, Google Wave, Google Photos, Google Notebook. Tập đoàn thương mại điện tử hàng đầu Amazon cũng có một danh sách dài tương tự với các sản phẩm tiêu biểu: Amazon Fire Phone (mẫu điện thoại thông minh duy nhất họ làm đến nay), Amazon Local, Amazon Destinations, Amazon Wallet, cùng vô vàn các sản phẩm khác chết từ trong trứng nước.

Những sản phẩm đó được tạo ra bởi những kỹ sư hàng đầu thế giới, với các công nghệ tân tiến thời bấy giờ, được bơm những khoản tiền khổng lồ để phát triển, nhưng chúng đều thất bại theo nghĩa không mấy ai quan tâm và sử dụng.

Trong khi đó, thử nhìn lại những sản phẩm thành công và được sử dụng rộng rãi như: công cụ tìm kiếm và quảng cáo của Google, công cụ quảng cáo của Facebook, YouTube, Netflix, Grab, Tiki, Momo, chúng ta cảm thấy sự thành công đó đến thật tự nhiên vì chúng đáp ứng được nhu cầu của rất nhiều người. Nhưng ban đầu, các nhu cầu đó không hề hiển nhiên chút nào, và người làm sản phẩm (thông thường bao gồm cả CEO của công ty) sẽ phải đủ thông minh và sáng suốt để nhìn ra những nhu cầu của người dùng mà chính người dùng cũng không biết. Kinh nghiệm đọc các tài liệu phỏng vấn của khách hàng cho mình thấy phần lớn mọi người đều không nhận ra thực sự họ cần gì, họ có nhu cầu nhưng nó rất mơ hồ và họ chỉ có thể cung cấp những thông tin rời rạc. Điều đó đòi hỏi người phỏng vấn không chỉ giỏi lắng nghe mà còn hiểu được tâm lý của người đối diện.

Không chỉ xác định được nhu cầu ở hiện tại, người làm sản phẩm phải đoán trước được nhu cầu trong tương lai. Từ đó, họ có thể tạo ra danh sách những việc cần làm cho quãng thời gian dài sắp tới.

Người làm sản phẩm là người tìm ra nhu cầu, vấn đề của khách hàng và tìm ra hướng đáp ứng nhu cầu, xử lý vấn đề đó. Tuy nhiên, người này không nhất thiết phải là người nghĩ ra giải pháp. Sau nhiều năm làm việc, mình nhận ra phần lớn các giải pháp cho nhu cầu của khách hàng đến từ đội kỹ sư. Các kỹ sư giỏi, nhiều kinh nghiệm, thường rất thông minh và sáng tạo. Họ cũng biết giải pháp nào là phù hợp với nền tảng công nghệ mà công ty đang có. Người làm sản phẩm sẽ không cố gắng thay thế đội kỹ sư giải quyết vấn đề trong tất cả các trường hợp, mà thay vào đó, họ giúp điều phối mọi người, giúp các buổi bàn luận dẫn đến được giải pháp cần thiết.

Điều này dễ khiến mọi người đưa ra đánh giá người làm Product không đóng góp gì nhiều, chỉ giỏi nói, vì phần lớn các giải pháp không đến từ họ và họ cũng không trực tiếp thực hiện giải pháp đó. Tuy nhiên, thực chất vai trò của người làm Product rất quan trọng vì họ giúp cả nhóm hiểu được vấn đề cần phải giải quyết là gì, và hướng giải quyết như thế nào là hợp lý. Có rất nhiều cách để giải quyết một vấn đề, nhưng trong số các giải pháp được đưa ra sẽ có rất ít giải pháp vừa làm hài lòng khách hàng, lại vừa phù hợp với nguồn lực của nhóm.

Người làm product là người thực hiện 2 trong những việc khó nhất của một công ty: Tìm ra giải pháp phù hợp với khách hàng và thuyết phục được họ dùng nó.

Nhìn rộng ra, người làm Product là cầu nối giúp mọi người liên kết giữa những việc cả nhóm đang làm với tầm nhìn của công ty. Họ sẽ hiểu mình không chỉ là người được thuê để lập trình, thiết kế, rồi hưởng lương. Thay vào đó, họ đang bắt tay vào giải quyết vấn đề khó cho khách hàng, và được trả công tương xứng với công sức bỏ ra. Họ sẽ tìm thấy được ý nghĩa của những việc đang làm, hiểu được tại sao sản phẩm của công ty lại tồn tại và cần phải làm gì để giúp sản phẩm đó đi lên.

Do đó có thể nói người làm Product là người giữ lửa, giúp tinh thần cả nhóm đi lên và giúp cả nhóm sẵn sàng đổ tâm huyết vào phát triển sản phẩm.

Làm product và những chuyện vui buồn

Trong quãng thời gian mình làm nghề, vui nhất là nhận được lời khen của khách hàng rằng sản phẩm của công ty tạo ra dễ dàng sử dụng và giúp họ làm việc nhanh, hiệu quả hơn. Họ luôn cảm thấy thích thú khi có cảm giác sản phẩm này được tạo ra để cho họ dùng, vì nó giúp họ lấy được những gì họ cần. Những phản hồi tích cực như vậy tạo ra vòng lặp phản hồi tích cực, khiến mọi người gắn bó thêm với sản phẩm và bỏ công sức để hoàn thiện nó hơn.

Sự căng thẳng khi đi làm cũng có rất nhiều. Có lẽ nỗi buồn lớn nhất là một trong những sản phẩm của tụi mình không đáp ứng được nhu cầu của thị trường. Mọi người đã dành nhiều tháng để chăm chút cho sản phẩm đó, nhưng cuối cùng nó chỉ đáp ứng được nhu cầu tức thời của một nhóm nhỏ khách hàng và không được sử dụng rộng rãi như kỳ vọng. Mặc dù đi làm startup là phải chuẩn bị tinh thần cho những tình huống thế này, nhưng khi chuyện xảy ra, tinh thần mọi người, gồm cả mình, vẫn bị lung lay, khiến mình tự hỏi về ý nghĩa của việc mình và công ty đang làm, cũng như tiếc nuối công sức. Tuy nhiên, công ty của mình có nhiều sản phẩm và may mắn là những sản phẩm khác được đón nhận rất tốt và tụi mình đã nhanh chóng chuyển hướng dồn sức qua mảng đó.

Kỷ niệm khó quên nhất với mình có lẽ là khi cả nhóm đã hoàn thiện giải pháp sau nhiều tháng làm việc. Khách hàng của tụi mình là một trưởng phòng IT bên Singapore, và mình nhận nhiệm vụ bàn giao sản phẩm cho khách. Sản phẩm đó được kỳ vọng sẽ giúp phòng Nhân sự của công ty quản lý nhân viên hiệu quả hơn bởi nó có thể cập nhật tức thời dữ liệu về tình hình công việc cũng như chấm công của hàng trăm nhân viên một cách dễ dàng. Hiểu đơn giản, phòng IT đang thực hiện dự án chuyển đổi số cho phòng Nhân sự trong chính công ty họ. Nhưng trong buổi bàn giao, khi bác trưởng phòng IT giới thiệu sản phẩm, bác trưởng phòng Nhân sự bất ngờ lên tiếng từ chối với lý do nó tạo ra quá nhiều sự thay đổi trong quy trình làm việc và các nhân viên của bác chưa sẵn sàng cho việc này. Điều này tạo ra cú sốc lớn cho cả nhóm của mình cũng như bác khách hàng. Rất nhiều buổi họp sau đó được thực hiện để thuyết phục đội nhân sự sử dụng giải pháp này, và sếp mình đã phải trực tiếp tham dự.

Bài học mình rút ra là làm ra một sản phẩm tốt và phù hợp với công ty vẫn chưa đủ. Người làm sản phẩm còn phải giỏi giao tiếp, hiểu những mối lo lắng của khách hàng, từ đó có thể thuyết phục họ sử dụng chúng. Việc này không thể chỉ dựa hoàn toàn vào đội BD (business development) hay marketing

để thực hiện, bởi họ không thể hiểu rõ sản phẩm bằng chính người làm ra và dồn tâm huyết vào nó. Đồng thời, sự việc này cũng nhắc nhở mình phải luôn chuẩn bị cho mọi thứ bất ngờ có thể xảy ra.

Nhìn lại những gì đã trải qua, mình cảm thấy mình đã phát triển được rất nhiều khi làm ở vị trí phát triển sản phẩm. Mình rèn luyện được lối tư duy giải quyết vấn đề, luôn nhìn mọi thứ dưới khía cạnh vấn đề - giải pháp để đưa ra được các giải pháp tinh gọn, gạt đi những sự cầu kì không cần thiết, nhưng vẫn đáp ứng được nhu cầu của khách hàng. Mình cũng hiểu hơn cách làm việc nhóm hiệu quả, cũng như gạt bỏ cái tôi để lắng nghe các giải pháp tốt hơn từ những người khác trong nhóm.

Chuẩn bị hành trang và các hướng phát triển của người làm product

Để làm ở vị trí phát triển sản phẩm, bạn không nhất thiết phải xuất thân từ ngành học công nghệ thông tin. Bạn có thể đến từ lĩnh vực marketing, tài chính, kế toán, toán học. Bạn chỉ cần có tư duy giải quyết vấn đề cũng như kiến thức trong lĩnh vực bạn đang làm. Thường khi bắt đầu, bạn sẽ không hiểu rõ về lĩnh vực mình mới bước chân vô và hấp thu từ từ kiến thức chuyên ngành qua việc tìm hiểu khách hàng cũng như đọc các tin tức về lĩnh vực.

Tuy nhiên để cho công việc trở nên suôn sẻ và dễ hòa nhập hơn, bạn có thể tự học các khóa học sau:

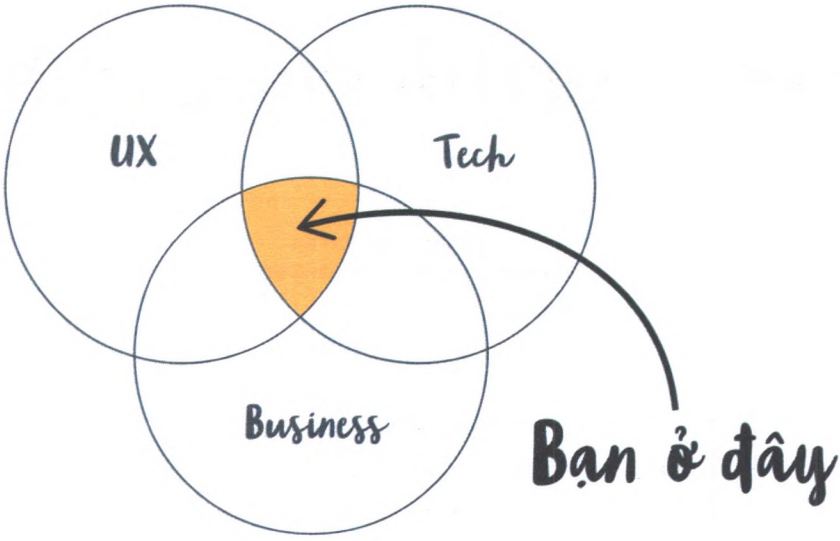
- Các khóa về lập trình cơ bản. Bạn có thể học các ngôn ngữ lập trình phổ biến như Python hay JavaScript. Bạn không cần trở thành chuyên gia trong các ngôn ngữ này, chỉ cần hiểu ở mức cơ bản.

- Học về cơ sở dữ liệu và cách dùng SQL. Cơ sở dữ liệu là xương sống của mọi sản phẩm công nghệ, hiểu về chúng sẽ giúp bạn nhanh chóng tìm ra hướng đi phù hợp cho nhiều vấn đề.

- Học về tài chính và quản trị doanh nghiệp. Điều này giúp bạn dễ dàng hiểu khách hàng hơn vì phần lớn các mối lo của họ sẽ xoay quanh hai thứ: chi phí vận hành và cơ cấu tổ chức hiện tại. Việc học về tài chính cũng giúp bạn tìm ra được cấu trúc giá cả (pricing structure) phù hợp với sản phẩm của mình.

Thường khi làm sản phẩm, bạn có thể bắt đầu từ những vị trí tập sự như Product Associate hoặc Junior Product Owner. Ở các vị trí đó bạn được giao phát triển các sản phẩm nhỏ của công ty, thúc đẩy nó phát triển. Nhìn chung, một công ty luôn có nhiều sản phẩm, với sản phẩm cốt lõi và sản phẩm hỗ trợ, và người mới làm bắt đầu ở những sản phẩm nhỏ. Lúc đó, bạn sẽ tập trung hoàn thiện các công việc được giao cũng như giúp cả nhóm phát triển tính năng mới bằng cách tạo ra danh sách những việc cần làm (gọi là backlog).

Khi có nhiều kinh nghiệm, bạn bắt đầu được giao các công việc có tính chất mở hơn như phỏng vấn khách hàng để thu thập thông tin. Thay vì được giao nhiệm vụ phát triển một tính năng cụ thể, bạn phải tự xác định xem tính năng tiếp theo cần làm là gì, tại sao phải làm như thế. Dần dần, bạn sẽ tự quyết định hướng đi cho sản phẩm của mình. Lúc đó, bạn chính thức trở thành Product Owner, người sẽ chịu trách nhiệm hoàn toàn cho sản phẩm bạn đang quản lý.



Khi cứng cáp hơn, bạn trở thành Product Manager – người quản lý toàn bộ sản phẩm của công ty. Bạn sẽ bắt đầu học về góc nhìn chiến lược, làm việc với khách hàng nhiều hơn trước. Bạn không chỉ xác định hướng đi của sản phẩm sẵn có mà còn xác định xem sản phẩm tiếp theo công ty cần phát triển là gì. Để làm điều đó, bạn cần hiểu rõ thị trường, hiểu rõ đối thủ cũng như các xu thế sẵn có. Bạn sẽ trở thành người có góc nhìn thấu đáo nhất trong công ty: vừa nhìn rộng cả thị trường lại vừa hiểu biết sâu về một góc của thị trường đó, nắm rõ tình hình nội bộ đồng thời biết được nhu cầu của khách hàng.

Lúc đó, bạn chính thức trở thành thuyền trưởng dẫn dắt sản phẩm vươn lên, giúp công ty đạt đến những tầm nhìn mà họ đặt ra. Sự thành công của công ty và sự hài lòng của khách hàng sẽ giúp bạn nhận ra những giá trị mình mang lại cho người khác. Và đó là lý do mình nói rằng:

"Nếu bạn thành công ở vị trí Product Manager,
bạn sẽ tìm thấy sự hài lòng trong cuộc sống của mình."

Thiết kế sản phẩm SẮC MÀU RIÊNG trong ngành công nghệ

Tác giả: **Hoàng Nguyễn**
Co-Founder & Design Coach GEEK Up

Chắc hẳn các bạn không còn xa lạ với Lazada, Shopee, Tiki,... - những cái tên nổi tiếng về Thương mại điện tử (TMĐT) ở Việt Nam. Nhưng bạn có biết, hơn 20 năm trước, "một nút bấm" ra đời đã tạo thay đổi cực kỳ to lớn trong lĩnh vực này. Lúc bấy giờ, người sử dụng các trang TMĐT đều bắt buộc phải "Đăng nhập", hoặc "Đăng ký" mới có thể hoàn thành thanh toán mua hàng. Yêu cầu này tưởng chừng hợp lý và đơn giản nhưng vô tình tạo ra rào cản vì người dùng mới rất ghét việc đăng ký, họ chỉ muốn mua hàng nhanh nhất có thể. Đội ngũ Designer sau đó đã đưa ra quyết định táo bạo: Đặt thêm 1 nút "Process as Guest" - cho phép người dùng thanh toán ngay mà không cần tạo tài khoản. Và kết quả thu được rất bất ngờ: Lượng thanh toán tăng 45% - Doanh số tháng đầu tiên tăng vọt lên 15 triệu đô. Sau 1 năm, nút bấm đơn giản này mang lại cho trang web đó 300 triệu đô. Những công việc tương tự như vậy có thể gọi là "Thiết kế sản phẩm - Product Design" (hay chính xác hơn là Digital Product Design).

Product Design là gì?

4 năm trước, mình được tham gia 1 dự án phát triển ứng dụng giao hàng ở Ấn Độ. Cũng giống các quốc gia đang phát triển khác, các bạn trẻ Ấn Độ sau tốt nghiệp Đại học có xu hướng ở lại thành phố lớn để phát triển sự nghiệp. Cha mẹ của họ - thế hệ lớn lên trong thời kỳ công nghệ chưa phổ biến - luôn hoài niệm và muốn nhắc nhở những đứa con của mình về quê hương bằng cách gửi lên những món đồ kỷ niệm, hoặc đồ ăn do chính họ chuẩn bị. Tuy nhiên, việc vận chuyển thật sự là trở ngại lớn và chưa có dịch vụ nào đủ tốt để giải quyết vấn đề này. Founder của dự án mong muốn tìm ra giải pháp hiệu quả, đơn giản dành cho mẹ của mình (thuộc tập người dùng không dễ làm quen với công nghệ).

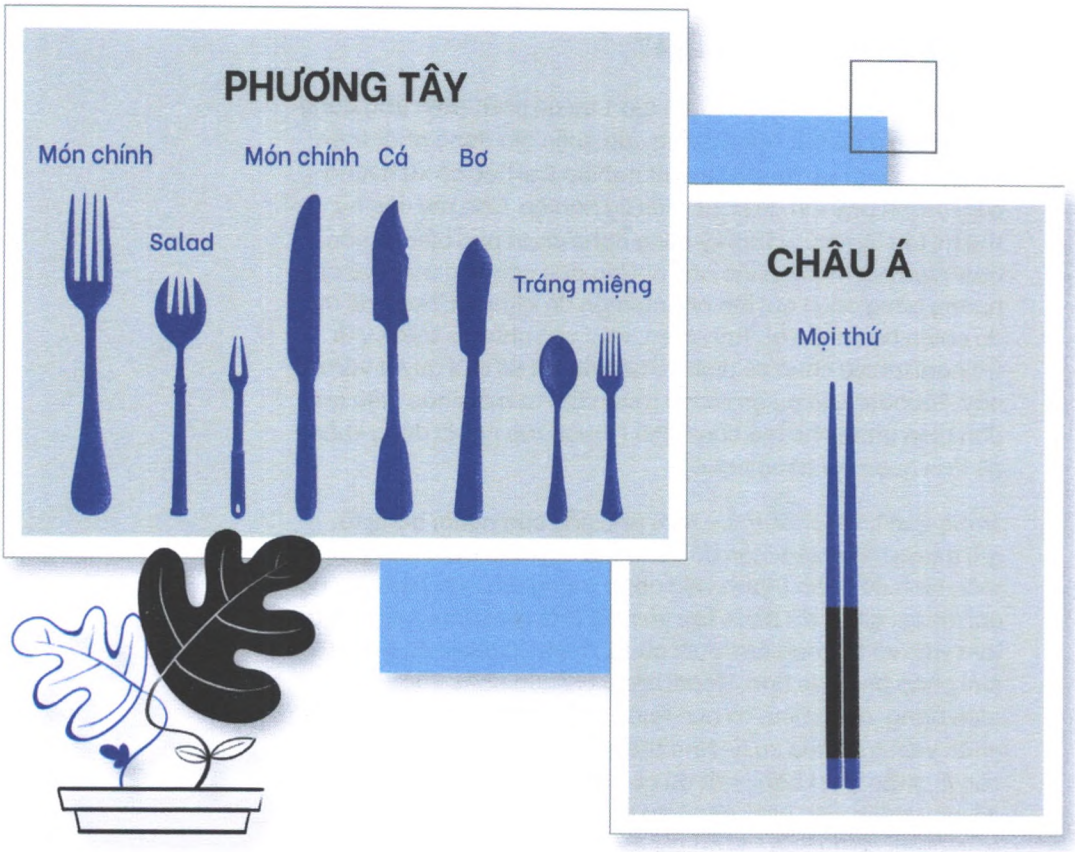
Trong quá trình phát triển, 80% nhu cầu của người dùng là gửi lượng lớn nhiều món đồ cùng lúc. Họ phải thao tác qua 14 màn hình để hoàn thành việc nhập thông tin, kích thước đóng gói, ngày, giờ,... rất phức tạp và mất thời gian. Sau nhiều ngày làm việc và thử nghiệm, cuối cùng, mình và team đã tìm ra giải pháp phù hợp hơn: Người dùng chỉ cần nhập vào số lượng món hàng, chụp hình và hẹn ngày, phần đóng gói còn lại sẽ do nhân viên trực tiếp xử lý. Nhờ thế, số màn hình giảm xuống chỉ còn 4, phần khó khăn nhất đã có nhân viên đảm trách. Thay đổi này khiến việc gửi hàng tiện hơn rất nhiều, đồng thời cũng giúp được nhiều cha mẹ có con cái ở xa quê hơn.

Không khó để nhận ra con người có các giới hạn, vấn đề và nhu cầu riêng. Có những vấn đề không tồn tại riêng lẻ ở mỗi cá nhân mà trên cả một cộng đồng nên cần được ưu tiên xử lý. Quá trình tìm ra các giải pháp khác nhau để giải quyết những vấn đề này là Product Design, người đảm nhiệm một vai trò trong quá trình đó là Product Designer.

Product Designer có 2 vai trò chính: Problem Solving - Tìm giải pháp thiết kế phù hợp và Building Experience - Xây dựng trải nghiệm. Nghĩa là, nếu có thể tìm ra giải pháp đúng, nhưng không thể khiến cho giải pháp đó dễ sử dụng thì Product cũng có thể thất bại.

Sự biến chuyển của Product Designer qua các thời kỳ

Thời gian đầu, Product Designer (Industrial Designer) là những người tạo ra các sản phẩm vật lý để tăng khả năng, loại bỏ giới hạn về thể chất trong lao động như: Cuốc, xẻng, dao, bàn, ghế,...



Dụng cụ trong ăn uống

Những sản phẩm dạng này bị giới hạn bởi khoảng cách địa lý. Đây là giai đoạn chúng chịu ảnh hưởng sâu sắc bởi đặc điểm từng địa phương, văn hóa, con người và nguyên vật liệu, nên dù có cùng một chức năng, sự khác biệt cũng rất rõ ràng dẫn đến sản phẩm có tính đa dạng cao.

Trong kỷ nguyên công nghệ, Product Designer (UX/UI Designer) bắt đầu đối mặt với các thử thách mới. *Trước tiên, đó là việc mất đi yếu tố khoảng cách.* Internet dần xoá bỏ giới hạn về mặt địa lý, điều này vừa là cơ hội, vừa là thử thách vì tập người sử dụng đa dạng hơn rất nhiều. Nó đòi hỏi Designer cần phải trang bị thêm kiến thức về các nền văn hoá và tập tính địa phương khác nhau. Thứ hai, các sản phẩm công nghệ thành công đã khiến cuộc sống của con người trở nên dễ dàng hơn và tạo ra thế hệ người dùng ngày càng đòi hỏi cao trong *trải nghiệm cảm xúc*. Giống như khi bạn đã sử dụng iPhone thì những chiếc điện thoại bàn phím của

Nokia ngày xưa trở nên thật lạc hậu. Trước đây, một sản phẩm vật lý có thể thông qua 5 giác quan của con người để tạo cảm xúc, nhưng digital (nền tảng số) chỉ có những thao tác khô khan trên màn hình thiết bị, kèm với một chuỗi các dịch vụ theo sau đó. Vì thế, việc *tạo cảm xúc* cho người dùng cũng ngày một khó.

UX và UI là 2 yếu tố luôn song hành

Có lẽ mọi người đã từng nghe qua vị trí UX/UI Designer, cách gọi này dễ khiến ta lầm tưởng chúng là một. Thực tế, chúng là 2 loại công việc khác nhau, đòi hỏi những kỹ năng khác nhau.

- *UX - Trải nghiệm người dùng*: là khái niệm đề cập tới cảm xúc và thái độ của người dùng khi họ sử dụng sản phẩm, dịch vụ hoặc hệ thống.

- *UI - Giao diện người dùng*: là những gì người dùng nhìn thấy và có thể tương tác thông qua các thiết bị.

Ví dụ một sản phẩm vật lý là cái ghế. Ghế được thiết kế đẹp, màu sắc bắt mắt phù hợp với phong cách nội thất trong phòng bạn. Những điều này có thể nhìn thấy được, gọi là UI. Nếu chiếc ghế được sử dụng với mục đích làm việc, nó phải được thiết kế chiều cao, độ cong và các chỉ số khác phục vụ cho mục đích đó, như: thêm tay gác, kê đầu, có bánh xe di chuyển,... Tất cả nhằm tạo cảm giác tiện dụng, dễ chịu và thoải mái, đây chính là UX.

Thiếu 1 trong 2 điều trên thì không thể tạo ra một sản phẩm tốt thật sự. Do đó, UX và UI cần phải song hành, gắn bó với nhau chặt chẽ trong Product Design.

Trong xã hội có nhịp độ gấp gáp như ngày nay, một trang web hoặc ứng dụng thành công cần phải đáp ứng nhanh và hiệu quả các nhu cầu của người sử dụng. Người dùng ngày một trở nên "lười" và khó tính. Dù sản phẩm bạn làm ra có tốt đến mấy, nếu trông nó "xấu xấu", họ sẽ chẳng thèm sử dụng. Nếu UX thiết kế tồi, quá rườm rà, khó sử dụng, họ sẽ bỏ cuộc ngay sau 5 phút dùng thử. Ứng dụng có UI đẹp và chuyên nghiệp dễ gây thu hút, có UX tốt sẽ chiếm được thiện cảm, được sử dụng thường xuyên và dễ được người dùng giới thiệu cho bạn bè.

Có thể nói, UI và UX góp phần quyết định sự thành bại của một sản phẩm!

Chính vì sự khác nhau rất rõ ràng giữa UX và UI, chúng ta cần phân biệt rõ sự khác biệt của 2 vị trí này:

UX Designer	UI Designer
Quan tâm tới sản phẩm sẽ được cảm nhận như thế nào	Quan tâm tới sản phẩm được trình bày ra sao
Wireframes, Storyboard, SiteMap	Sản phẩm thiết kế hoàn chỉnh để lập trình
Tham gia vào dự án ở giai đoạn đầu và tiếp theo để cải thiện	Có thể chỉ cần tham gia sau khi research xong
Nên bắt đầu tìm hiểu về hành vi, tâm lý con người	Làm quen các khái niệm về thiết kế: màu, sự cân bằng, tương phản, chữ, sự đồng nhất,...
Một vài cuốn sách hay: Don't make me think, The Design of Everyday Things, The Shape of Design,...	Tham khảo, theo dõi các designer nổi tiếng như: Gleb Kuznetsov, Eddie Lobanovskiy,... Tự rèn luyện khả năng về thẩm mỹ, ghi nhớ các quy tắc cơ bản của thiết kế đồ hoạ
Tìm hiểu UX Design Process: Clarify/Define Problem, User Research, Personas,...[1]	Tìm hiểu về UI Design Process: Discover, Define, Develop, Delivery [2]
Đọc blog về tech và design mỗi ngày để cập nhật thông tin công nghệ mới	Rèn luyện các công cụ hỗ trợ cho thiết kế như: Sketch, Figma, XD, Flinto,...
Học và tìm hiểu về Business/ Product mindset (Kinh doanh/Tư duy sản phẩm)	Tự nghĩ và thiết kế các dự án cá nhân để luyện tập và cập nhật xu hướng mới
Giải quyết vấn đề, học hỏi, giao tiếp tốt, biết lắng nghe, biết phân tích, chú ý vào chi tiết, đồng cảm với người dùng, nghĩ về bức tranh xa hơn	Suy nghĩ sáng tạo, có mắt thẩm mỹ, tỉ mỉ, kiên nhẫn, có trách nhiệm, thích thú khi có ý tưởng mới và chấp nhận thử thách

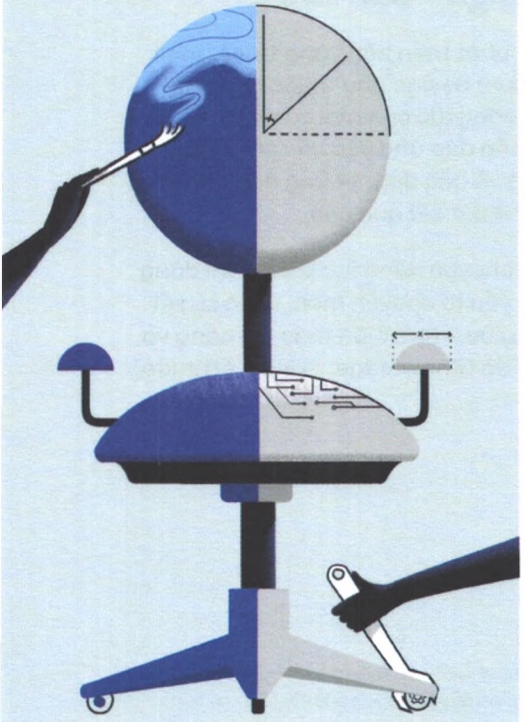
Hiện tại, Việt Nam vẫn chưa có trường lớp đào tạo bài bản nên sẽ khó hơn nếu bạn muốn bắt đầu với vị trí UX Designer, còn vị trí UI Designer thường sẽ từ Graphic Designer phát triển lên.

[1] *Quy trình thiết kế trải nghiệm người dùng: Xác định vấn đề, Nghiên cứu người dùng, Lên chân dung khách hàng,...*

[2] *Quy trình thiết kế giao diện người dùng: Tìm hiểu, Xác định ý tưởng, Phát triển ý tưởng, Thực thi*

Hành trình lặn lội 8 năm trong nghề

Mình chọn theo học Graphic Design khi vào Đại học dù gia đình không ủng hộ (cũng như các bậc phụ huynh khác muốn con cái có công việc ổn định), nhưng mình vẫn nghĩ công việc thiết kế có lẽ phù hợp với bản thân hơn. Khi còn đi học, mình nhận ra Design không chỉ dừng ở việc làm ra những thứ đẹp đẽ mà phải phục vụ một mục đích cụ thể. Mình thấy thích thú với cảm giác nghĩ ra một Ý tưởng/ Giải pháp nào đó cho bài toán gặp phải, sau đó thực hiện và nhìn kết quả. Sau khi tốt nghiệp, mình đưa ra quyết định trái ngược với bạn bè: từ chối lời mời của các agency danh tiếng và chọn vào làm ở một công ty thiết kế Website. Lúc đó, mình chỉ suy nghĩ đơn giản là tìm cơ hội học một thứ mới hơn, cũng như tin vào tiềm năng của lĩnh vực thiết kế cho Digital.



Trong 8 năm qua, có thể chia sự nghiệp của mình làm 3 giai đoạn:

2 năm đầu - Tích lũy kinh nghiệm: Say "Yes" to everything. Grinding your experience.

Lúc bắt đầu, mình chủ động làm nhiều công việc khác nhau một lúc: thiết kế web, branding, dàn trang,... Mình thu thập kinh nghiệm dàn trải nhất có thể, ứng dụng các kiến thức nền tảng vào công việc, nhằm mục đích chắt lọc và tìm ra những thứ phù hợp với bản thân. Đây cũng là giai đoạn cần xây dựng các mối quan hệ trong công việc, tìm những người đi trước để học hỏi, theo dõi những Designer đầu ngành và có khách hàng tiềm năng. Thời điểm này, mình nghĩ không nên quá quan trọng hay đặt nặng chuyện giá cả, chi phí. Có việc để làm, có kinh nghiệm để tích lũy là thứ cần ưu tiên hàng đầu.

4 năm tiếp theo - Xây dựng nền tảng: Learn to say "No". Storing your knowledge.

Sau một khoảng thời gian trải qua cả thành công lẫn thất bại, mình bắt đầu xây được quy trình làm việc phù hợp cho bản thân. Liên tục áp dụng và cải tiến quy trình này đã giúp mình ít lặp lại những sai lầm khiến dự án thất bại trước đó. Đồng thời, mình tự đặt ra "Tiêu chuẩn" tối thiểu về sản phẩm làm ra, nó giúp mình chọn lựa dự án dễ dàng hơn cũng như lấy đó làm thước đo cho bản thân khi làm việc. Đây là giai đoạn mình học cách từ chối những công việc nhàm chán, ít thử thách hoặc không phù hợp với chất lượng sản phẩm mà mình mong muốn. Điều này sẽ giúp chúng ta giữ được lửa trong công việc, cảm thấy tự hào với sản phẩm làm ra và có được portfolio đủ tốt như kỳ vọng.

Khi dự án kết thúc, mình tập thói quen note lại điều gì đã làm tốt, điều gì có thể làm tốt hơn. Những bài học này sẽ giúp chúng ta cải thiện hàng ngày.

Những năm sau đó - Hệ thống kiến thức: Share to learn. Learning from the best.

Sự khác biệt giữa Junior và Senior chính là tính trách nhiệm đi kèm với sự ổn định trong công việc. Trong đó, sự ổn định được thể hiện qua chất lượng sản phẩm. Vì thế, để tránh làm việc theo cảm hứng, mỗi Designer đều phải xây dựng quy trình làm việc phù hợp. Bên cạnh đó, bạn hãy tự hệ thống lại những kiến thức chuyên môn để áp dụng chúng vào từng dự án cụ thể. Kiến thức tuy đầy rẫy trên mạng, nhưng để biết được khi nào nên dùng lại là việc rất khó.

Mình cũng hay chia sẻ lại những kiến thức mình biết. Đây còn là cách để thu thập được nhiều góc nhìn khác nhau, lắng nghe có chọn lọc kinh nghiệm từ người khác. Việc hệ thống kiến thức cũng giúp chúng ta biết cách nói cho người khác hiểu được giá trị của từng công đoạn trong thiết kế, từ đó hỗ trợ ít nhiều cho khả năng thuyết phục khách hàng của một Designer.

Suốt 8 năm trong nghề, mình đã trải qua khá nhiều các câu chuyện vui, buồn. Tới giờ, kỷ niệm mình nhớ nhất là một dự án dành cho chó mèo năm 2017. Có 2 founder đã liên lạc với mình, đưa ra ý tưởng về một ứng dụng tạo cộng đồng cho những người yêu quý và có khả năng nhận nuôi chó mèo vô chủ. Ứng dụng cho phép đăng tải 1 video có sẵn, hoặc quay tại nơi mà người dùng thấy những chú chó mèo "cơ nhỡ". Dựa vào khoảng cách gần nhất, ứng dụng sẽ thông báo cho các tổ chức hoặc thành viên có điều kiện để giúp đỡ chúng nhanh nhất có thể. Ngoài ra, ứng dụng cũng giúp các gia đình đang có nhu cầu nhận nuôi tìm được một người bạn phù hợp. Vì cũng là người yêu động vật nên dù khi đó đang rất bận, mình vẫn nhận tham gia dự án và chi phí để họ quyết định. Kết quả là ứng dụng ra mắt thành công tốt đẹp trong buổi pitching[1] ý tưởng với một tổ chức bảo trợ động vật và được đầu tư. Mình nhớ lúc đó là 3h sáng, khi nhận được tin báo kèm câu: "We are going to double your rate for everything you've done" (Chúng tôi sẽ trả gấp đôi chi phí cho những gì bạn đã làm), mình đã vui tới mức không thể ngủ tiếp được. Đây chính là cảm giác mà nghề Product Design mang lại: vừa tạo ra giá trị cho xã hội, vừa tạo ra giá trị cho bản thân.

Đặt chân vào Product Design - bạn cần gì?

Việc phát triển bền vững trong nghề Product Design phụ thuộc rất nhiều vào năng lực của mỗi cá nhân. Năng lực cần đáp ứng yêu cầu của hoạt động và bảo đảm những hoạt động đó có thể đạt kết quả cao.

Câu chuyện năng lực không chỉ dừng lại ở yếu tố chuyên môn, nó là sự kết hợp giữa 3 thứ: Kiến thức, Kỹ năng và Thái độ (Knowledge, Skills & Attitude).

[1] *Pitching: một cá nhân/công ty thuyết phục khách hàng/nhà đầu tư rót vốn cho ý tưởng của mình*

Kiến thức: Hãy cố gắng hình thành những thói quen tốt như: tự học và nghiên cứu, đọc sách, theo dõi cập nhật thường xuyên các kiến thức mới về công nghệ,...

Kỹ năng chuyên môn:

1. *Research (Nghiên cứu)*: Kỹ năng thu thập và phân tích các thông tin khác nhau. Những thông tin này bao gồm nhiều khía cạnh như: Kinh doanh, thị trường, đặc biệt là người dùng. Hiểu được những mong muốn, suy nghĩ và hành vi của họ sẽ giúp chúng ta đưa ra giải pháp thiết kế phù hợp.

2. *UX & Prototype Design*[1]: Đây là kỹ năng chính của UX Designer. Kết hợp với những thông tin có được từ quá trình research, designer sẽ phối hợp với team tìm ra các giải pháp cân bằng giữa Lợi ích của Business (Công ty), Hữu ích với User (Người sử dụng) và có thể Thực thi về mặt Technical (Kỹ thuật).

3. *Sáng tạo, thẩm mỹ, tỉ mỉ*: giúp tìm ra giải pháp dễ dàng, thú vị hơn và hoàn thành thiết kế với độ hoàn thiện cao nhất. Những kỹ năng này tuy một phần thuộc về thiên khiếu, nhưng bạn hoàn toàn có thể tự trau dồi và nâng cao qua thời gian.

4. *UI Design*: Kết hợp giữa các kiến thức thẩm mỹ trong thiết kế đồ họa và hiểu biết về hành vi tương tác trên các thiết bị công nghệ để tạo ra giao diện bắt mắt, dễ sử dụng.

5. *Suy nghĩ logic*: là cầu nối cho quá trình thiết kế từ vấn đề đến giải pháp.

Kỹ năng mềm:

1. *Tự học*: Trong thời đại mà mọi thứ thay đổi tới chóng mặt, chỉ cần dừng tiếp thu kiến thức trong 1 năm, chúng ta đã bị tụt hậu.

2. *Critical Thinking*: Thiết kế nhìn chung là quá trình tìm ra giải pháp phù hợp cho từng vấn đề riêng biệt, vì thế, kỹ năng này giúp chúng ta biết làm thế nào xác định được vấn đề cần giải quyết và làm sao giải quyết nó hiệu quả nhất. Đây là kỹ năng không thể thiếu ngày nay.

3. *Giao tiếp*: Nếu không truyền tải được những kiến thức chuyên môn cho người ngoài chuyên môn hiểu, chúng ta sẽ khó thuyết phục được người khác về giải pháp thiết kế của mình. Ngoài ra, kỹ năng này giúp bạn dễ dàng hơn trong teamwork, bởi không một ai có thể làm tốt mọi việc, để giải quyết những vấn đề lớn chúng ta cần có một team.

4. *Quản lý công việc*: Quản lý quỹ thời gian và sắp xếp mức độ ưu tiên cho từng đầu việc cần làm. Bạn cũng cần khả năng tập trung bởi có một sự thật không thể chối cãi: Càng thiếu tập trung, chúng ta càng dễ phạm sai lầm.

---

[1] *UX & Prototype Design: Thiết kế bản dùng thử/bản nháp*

Thái độ:

1. *Sự kiên cường và cầu tiến*: để chấp nhận thử thách, học từ thất bại và chinh phục khó khăn. Hãy tin rằng mình luôn có thể làm tốt hơn, luôn có cơ hội để phát triển, nghiêm túc với kế hoạch phát triển bản thân.

2. *Tò mò và quan sát*: luôn tò mò cách mọi thứ xung quanh đang vận hành, quan sát và đặt những câu hỏi cho các vấn đề cần giải quyết.

3. *Đồng cảm với người dùng*: rất quan trọng, đặc biệt đối với UX Designer, vì chỉ có đồng cảm mới giúp designer có thể đưa ra các giải pháp thiết kế phù hợp. Đôi khi, để đồng cảm, bạn còn phải trực tiếp trải nghiệm xem "nỗi đau" mà người dùng đang trải qua là gì.

Cơ hội luôn đi kèm những sự đánh đổi

Chắc hẳn chúng ta đã được nghe nói khá nhiều về thời đại 4.0. Nhìn từ góc độ design, thời đại này được gọi tên: "Khi người dùng lên tiếng - Age of the empowered customer". Nghĩa là, các công ty cần phải định hình lại doanh nghiệp, chuyển dịch từ mô hình cũ "Business Centric[1]" sang "Customer Centric[2]", xây dựng quy trình và nghiệp vụ kinh doanh xung quanh "Hành trình khách hàng - Customer Journey". Đã bao giờ bạn tự hỏi: "Sao mình lại chọn quán cà phê A, thay vì quán B, cho dù quán A xa nhà hơn một chút?". Đó là bởi những trải nghiệm khác nhau của mỗi quán. Có thể là quán A có bãi giữ xe ở gần, nhân viên thân thiện, đồ uống ngon, không gian bài trí đẹp, bật nhạc phù hợp tâm trạng,...

Các công ty công nghệ thế giới xâm nhập vào Việt Nam và sở hữu một số lượng lớn người dùng. Việc sử dụng các sản phẩm đó hằng ngày làm mức độ đòi hỏi về trải nghiệm công nghệ ngày càng cao. Điều này bắt buộc các doanh nghiệp trong nước muốn cạnh tranh phải thay đổi góc nhìn về UX Design. Bằng chứng là các công ty dẫn đầu thị trường như: thegioididong, FPT, VNG, Viettel,... đã bắt đầu xây dựng đội ngũ UX/UI Designer riêng, kéo theo thị trường nhân sự của lĩnh vực này trở nên nóng hơn bao giờ hết. Các công ty phát triển sản phẩm cũng xuất hiện ngày một nhiều, giúp cho mặt bằng về chuyên môn cao hơn. Vì thế, đối với tất cả các bạn đam mê, hứng thú với công việc Product Design, cơ hội nghề nghiệp trong lĩnh vực này đang rộng mở.

Tất nhiên, bất kỳ công việc gì cũng luôn gặp phải những khó khăn nhất định. Điều quan trọng là bạn có dám đánh đổi sự thoải mái để phát triển bản thân hay không. Hãy tự hỏi và trả lời những câu hỏi:

[1] *tập trung vào công ty* / [2] *tập trung vào khách hàng*

- Mình có sẵn sàng đánh đổi những cuối tuần nghỉ ngơi thoải mái để thiết kế thêm một vài màn hình hoặc tìm đọc thêm kiến thức?
- Mình có chấp nhận những công việc sẽ tốt cho sản phẩm nhưng lại đầy khó khăn và áp lực và không liên quan đến thiết kế?
- Mình có đồng ý giảm bớt thời gian lướt Facebook/chơi game để tìm hiểu sâu hơn lĩnh vực mới cho dự án sắp tới?
- Mình có thể hạ cái tôi của bản thân để lắng nghe góp ý từ nhiều phía, tiếp thu và đối mặt với thất bại?
- Khi thiết kế bị từ chối, liệu có dám khách quan nhìn nhận những điều chưa tốt ở bản thân để cố gắng hơn hay không?

Vẫn còn rất nhiều câu hỏi khác tương tự với hy vọng giúp chúng ta chọn đúng thứ để đánh đổi và quyết tâm hơn với nó. Chúc các bạn luôn kiên cường trên con đường đã chọn!

BrSE Đũa phép chọn phù thuỷ

Tác giả: **Doãn Thiện**
Co-founder lovelock.one
Blockchain researcher icetea.io

Khoảng 10 năm trở lại đây, cụm từ BrSE được nhắc tới như một nghề "hái ra tiền". Vì thế, tôi muốn chia sẻ những kinh nghiệm của bản thân với các bạn còn đang bỡ ngỡ về công việc này. Hy vọng những chia sẻ của tôi có thể đem lại cho các bạn hình dung rõ hơn và đưa ra được lựa chọn phù hợp với bản thân mình.

BrSE là gì?

Br là viết tắt của Bridge, SE là viết tắt của System Engineer, dịch ra tiếng Việt gọi là "Kỹ sư cầu nối". Từ giờ, tôi xin dùng từ BrSE để gọi tên nghề này.

Nghe tới "kỹ sư cầu nối" chắc nhiều bạn hình dung nghề bên ngành cầu đường, thực ra, đây là công việc trong lĩnh vực công nghệ thông tin, cụ thể là trong các công ty outsourcing.

Công ty outsourcing là công ty được thuê để làm phần mềm hay một phần của hệ thống thông tin cho những công ty khác. Trong đó, BrSE là người làm việc giữa khách hàng và đội dự án ở Việt Nam (từ giờ sẽ gọi là team offshore) giúp cho công việc của hai bên được thông suốt. Thông thường, BrSE sẽ ngồi làm việc bên phía khách.

Sự thành công trong công ty outsourcing được đo bằng mức độ hài lòng của khách hàng. Như vậy vai trò của BrSE trong chuỗi sản xuất này là rất quan trọng.

Vị trí BrSE đặc biệt được sử dụng khi làm việc với khách hàng Nhật Bản, song hiện nay, công việc này cũng xuất hiện trong các dự án thực hiện với các đối tác Hàn Quốc, Trung Quốc, Âu Mỹ.

Tại sao BrSE lại quan trọng?

Khách hàng mà không có BrSE thì biết truyền đạt công việc cho team offshore thế nào khi không phải ai cũng biết tiếng Việt? Team offshore không có ai giải thích yêu cầu của khách thì biết làm sao? Nghe thoáng qua, nghề này có vẻ không quá khó khăn. Các bạn cần chuẩn bị đủ kiến thức và tâm lý để có thể ngồi cùng với khách hàng nghe yêu cầu hay đề xuất giải pháp, cũng như truyền đạt lại những yêu cầu đó tới team offshore một cách dễ hiểu để làm sao đưa ra được sản phẩm phù hợp với kỳ vọng của khách hàng.

Còn nhớ, hồi tôi còn là sinh viên đại học, cụm từ BrSE còn chưa xuất hiện. Vốn học Công nghệ Thông tin nên tôi cũng nghĩ mình sẽ làm mấy cái kiểu như phần mềm diệt virus Bkav. Ở trường, ngoài học tiếng Anh ra tôi còn "bị" học tiếng Nhật. Ban đầu, tôi đâu có thích tiếng Nhật, vài kì đầu học đuối lắm, nhưng luôn nhớ lời thầy dạy quân sự trong đầu: "Cũng chế độ ấy tại sao người ta làm được mà mình không làm được?" nên thấy chúng bạn học thì mình cũng ráng mà học.

Tốt nghiệp Đại học rồi đi làm, lúc này tôi mới biết là mình đã đặt một chân vào con đường BrSE. Sau 2 năm code cật lực ở offshore team, tôi được tham gia phỏng vấn với khách hàng và đi onsite (trở thành BrSE). Lúc mới vào team, tôi mong được cống hiến nên việc nào cũng muốn nhận, không ngại làm thêm giờ. Kỳ thực là càng làm thêm giờ tôi càng thấy nể đội BrSE bên kia chiến tuyến. Nhật Bản và Việt Nam chênh nhau 2 tiếng, nghĩa là ở Việt Nam mà 10h đêm thì bên Nhật đã sang ngày mới rồi. Có vài lần tôi mạnh dạn đề nghị để được làm BrSE nhưng chưa được chấp nhận vì khả năng kỹ thuật lúc đó còn chưa vững. Đúng thời điểm dự án của khách hàng mở rộng, đội BrSE thiếu người và tôi đã xin phỏng vấn khách hàng. Ơn trời, tôi được khách hàng chấp thuận.

Ban đầu nghe các sếp nói đi nước ngoài sướng lắm: nào là được làm với khách hàng lớn, được nâng cao kỹ năng mềm, được đi du lịch, rồi thu nhập ổn định,... Cũng đúng, nhưng đâu có ai trải thảm đỏ cho mình đi lấy những thứ đó đâu.

Mang tiếng có trong tay tấm bằng tiếng Nhật N2 mà đặt chân qua Nhật, khách hàng nói tôi chẳng hiểu gì. Áp lực công việc thì cực kì kinh khủng: báo cáo liên tục, công việc tới tấp, khách hàng mắng, offshore team giục. Thêm nữa ở nơi đất khách quê người, đi lạc vài phen, tiêu tiền thì cứ phải quy đổi ra tiền Việt làm tôi thấy căng như dây đàn. Sau vài tuần tự học thêm ngôn ngữ và quen với nhịp sống, công việc BrSE của tôi bắt đầu suôn sẻ.

Cũng nhờ có các anh chị đi trước chia sẻ "bí kíp" trở thành BrSE chuyên nghiệp nên tôi đã tiến bộ từng ngày. Liệu nghề này có hợp với bản thân không? Liệu không học công nghệ thông tin có trở thành BrSE được không? Cần những kỹ năng gì để trở thành BrSE?

Có 3 yếu tố các bạn cần đặc biệt quan tâm nếu muốn trở thành một BrSE chính là: Kỹ thuật lập trình, ngoại ngữ và kỹ năng mềm. Bên cạnh đó, khi làm BrSE, tôi còn hiểu thêm được chữ "nhẫn" hay tôi thường trêu với anh em là khả năng "lì đòn".

Vì sao lại là "nhẫn"?

Không có khả năng này thì bạn chắc cũng chẳng học nổi ngoại ngữ tới mức hiểu người bản xứ nói gì. Đơn cử như tiếng Nhật, làm BrSE thì chí ít bạn cũng cần có N2, mà N2 như tôi lúc ban đầu qua Nhật cũng như gà mắc tóc. Để có được N2 ấy hả? Nếu không

Khi làm **BrSE**, *tôi còn hiểu thêm được chữ "nhẫn" hay tôi thường trêu với anh em là khả năng "lì đòn".*

biết gì, bạn phải mất tầm 2 năm (với người bình thường nhé, còn các bạn siêu nhân học 9 tháng hay 1 năm thì tôi xin phép không nhắc tới).

Về vấn đề kỹ thuật, nghề BrSE cũng chia ra các cấp độ khác nhau (nghề nào cũng vậy). Những BrSE cấp độ cao (đi đề xuất rồi thương lượng với khách hàng) là người rất giỏi chuyên môn lập trình, số lượng này không nhiều. Tùy công việc cụ thể mà sinh ra yêu cầu cho người làm BrSE, ví dụ nếu khách hàng tốt ở kỹ thuật rồi thì BrSE chỉ làm việc theo hướng truyền đạt thông tin, báo cáo tình hình.

Ngoài ra, những kỹ năng mềm như quản lý thời gian, thuyết trình, làm báo cáo,... cũng mất khá nhiều thời gian. Chưa kể, muốn làm BrSE mà chỉ biết ngoại ngữ thì bạn cũng cần học thêm cả lập trình. Điều này cũng mất vài năm chứ không phải chuyện một sớm một chiều.

Cạm bẫy của chiếc "đũa phép" BrSE...

Khi bước chân vào nghề sẽ có một số cái bẫy mà sau này bạn mới nhận ra được. Như đã nói ở trên, bạn cần 3 yếu tố để trở thành một BrSE nhưng nó cũng dễ khiến bạn dễ rơi vào hoàn cảnh "cái gì cũng biết mà chẳng giỏi cái gì":

Tiếng Nhật nhàng nhàng, code cũng bình bình và kỹ năng mềm cũng không nổi trội. Có bạn sẽ hỏi: "Em không học nổi tiếng Nhật, nếu biết tiếng Anh có thể làm BrSE ở Nhật được không?" Chắc chắn là có. Đừng quên, bạn cũng cần giỏi cả kỹ thuật và kỹ năng mềm. Hiện tại cũng có nhiều trường hợp như vậy. Họ vẫn được khách hàng ưu ái, nhưng xét về dài hạn, bạn sẽ khá vất vả nếu sống ở Nhật mà không có tiếng Nhật.

Theo tôi, bạn cần có mindset (tư duy) làm việc, đặc biệt là tư duy làm phần mềm. Nghe hơi cao siêu nhưng tư duy đơn giản là đào sâu suy nghĩ, và làm BrSE thì phải suy nghĩ về hệ thống, về phần mềm rồi. Còn làm phần mềm có quy trình thế nào, các mô hình phát triển ra sao, bạn phải là người tìm hiểu. BrSE sẽ được tiếp xúc với rất nhiều dự án nên tuỳ dự án mà sẽ áp dụng ngôn ngữ lập trình, framework khác nhau. Nếu dự án rơi vào đúng ngôn ngữ và framework bạn biết, điều đó thật tuyệt. Nhưng nếu rơi vào thứ bạn không biết thì sao? "Đào sâu suy nghĩ" thôi nhỉ! Ví dụ, biết khách hàng này hay làm về Java thì tìm hiểu Java hay hệ thống của họ vẫn dùng COBOL thì ngại gì mà không học. Để tư vấn được cho khách, bạn cũng cần biết cách làm thiết kế. Đơn giản khách làm web mà cần thiết kế màn hình thì cũng cần biết chút HTML, CSS để vẽ sơ lược màn hình. Rồi hệ thống nào cũng có cơ sở dữ liệu, BrSE cần nắm được thao tác với cơ sở dữ liệu thế nào cho hợp lý. Vậy nên "nhẫn" để mà "đào sâu suy nghĩ".

Hãy nhớ phải nắm trong tay một kỹ năng mà mình thật giỏi và phải trau dồi nó mỗi ngày. Ngoại ngữ tốt thì hàng ngày hãy sử dụng nó. Lập trình tốt thì hàng ngày hãy vui với những dòng code. Kỹ năng mềm tốt thì hãy đi kết nối mọi người.

Khi bạn điên cuồng lao vào nghề BrSE sẽ có thêm một cái bẫy ngọt ngào hơn mà bạn phải đánh đổi: Sự mất cân bằng. Làm cây cầu nối những "bờ vui" mà mất cân bằng thì gay go lắm.

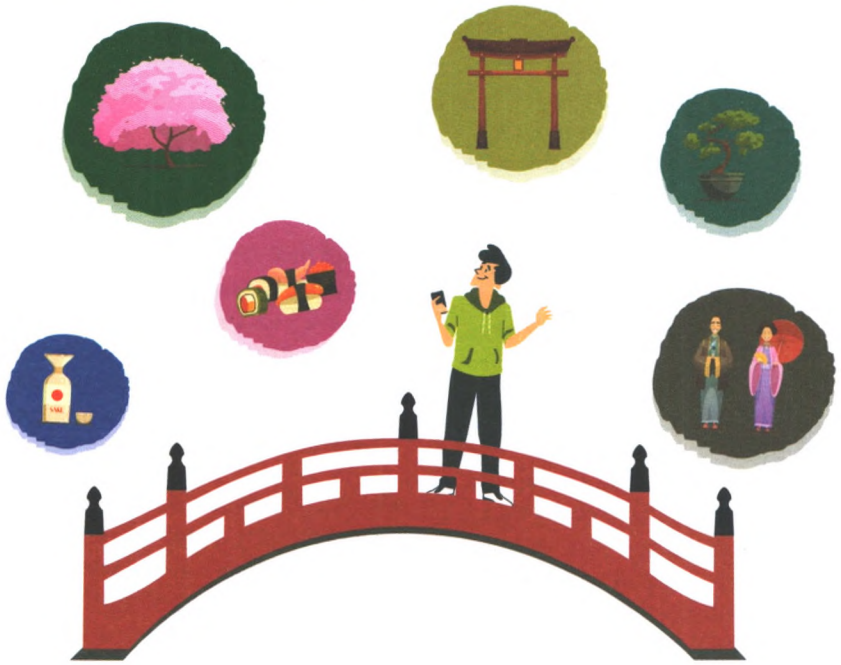
Là một BrSE thì việc OT (Overtime: làm thêm giờ) là điều không thể tránh khỏi. Hãy chú ý tới sức khoẻ nhé. Khoẻ để mà... OT. Khách hàng có mắng cũng hãy bình tĩnh mà truyền đạt lại cho team offshore, chứ "mất cân bằng" cảm xúc là khéo mất vui cả ngày.

Đọc tới đây chắc các bạn bắt đầu "sợ". Làm gì mà chẳng khó. Nếu bạn ước mơ thành BrSe, hãy cứ dấn thân. "Đũa phép" chọn "phù thuỷ" ấy mà, nếu thấy chưa hợp thì ta đi kiếm "đũa phép" khác.

...và cả những niềm vui

Các bạn sẽ được đi đây đi đó, trải nghiệm xem cuộc sống ngoài nước khác gì trong nước, được tự do vẫy vùng xa khỏi vòng tay ba mẹ, được "ăn cơm Tàu, ở nhà Tây" và biết đâu... "lấy được vợ Nhật".

Và tôi cũng suýt có "vợ Nhật". Thời gian đó là lúc tôi mới đặt chân tới Nhật, còn nhiều bỡ ngỡ nên được khách hàng và các tiền bối kèm cặp. Tôi may mắn được quản lý đội dự án 10 người (là team offshore tôi làm chung ở Việt Nam). Dự án này được khách ưu ái cho làm từ thiết kế hệ thống tới đưa lên môi trường thật. Làm việc với tôi và team offshore là một khách hàng nữ. Vì tính chất quan trọng của dự án mà tôi và khách họp với nhau tới 3, 4 lần một ngày. Sáng họp báo cáo chung với các dự án khác, trưa họp cập nhật tình hình với team offshore, chiều lại họp riêng để "khách" chỉ cho cách làm, thi thoảng còn họp để giải quyết vấn đề



bất ngờ xảy ra. Làm gần bạn "khách" đó vài tháng, tôi cũng rất có cảm tình, người đâu mà vừa năng động, nhiệt tâm lại còn xinh xắn. Ở trên công ty chỉ bàn chuyện công việc, nên khi về nhà tôi đã lẳng lặng tìm Line (ở Nhật hay dùng Line để tán gẫu giống Zalo ở Việt Nam) và Facebook của bạn ấy. Thật bất ngờ vì tôi tìm thấy Facebook và gửi yêu cầu kết bạn. Ngay hôm sau, khi kết thúc buổi họp sáng, tôi được kéo ra một góc và hỏi "Đây là cậu phải không?". "...Ừ, đúng rồi". Vậy là chúng tôi kết bạn. Cuối tuần, chúng tôi thường gặp nhau nhưng không phải là công việc nữa. Thế nhưng vào giai đoạn cuối của dự án, bạn ấy được công ty cử đi xa và tôi không thể gặp lại.

Chẳng còn cách nào khác, tôi lại tiếp tục cày cuốc một mình.

Về thu nhập thì miễn bàn. Đi làm ở nước ngoài vài năm sẽ dư tiền để tậu nhà, tậu xe, sinh con và có một cuộc sống gọi là "ổn". Ví dụ làm BrSE ở thị trường Nhật Bản, mới bước chân vào nghề thì thu nhập (tính theo tháng) của bạn sẽ khoảng từ 40 - 50 triệu đồng. Sau vài năm chinh chiến, kỹ năng bản thân được nâng cao, bạn có thể đạt tới mức lương 80 - 100 triệu đồng hoặc hơn, tuỳ thuộc vào khả năng của bạn.

Bạn nào muốn định cư ở nước ngoài, BrSE là một cơ hội vàng không thể bỏ qua. Nó là bước đệm để bạn có thể xin được visa vĩnh trú (visa dài hạn) hoặc xin đổi quốc tịch.

Những điều vừa kể trên là những điều được cho bản thân, nhưng có một điều vô cùng đặc biệt mà nghề BrSE tạo ra chính là công việc cho team offshore. Bạn càng tạo được niềm tin với khách hàng, team offshore càng nhiều việc. Tạo được thêm càng nhiều công việc cho người khác, có phải "tài khoản hạnh phúc" của bạn càng tăng cao không?

Công việc mà tạo ra nhiều hạnh phúc cho bản thân và cho cả người khác như vậy, liệu có nên làm? Tuổi trẻ mà, cứ xông pha đi!

AN TOÀN THÔNG TIN

Niềm tự hào không phô trương

Khách mời phỏng vấn: **Nguyễn Lê Thành**
Chief Security Officer VNG
Biên soạn bài viết: **Ban biên tập**

Năm 2016, tại khu vực làm thủ tục chuyến bay, màn hình hiển thị thông tin của các Sân bay Quốc tế Tân Sơn Nhất, Sân bay Quốc tế Nội Bài, Sân bay Quốc tế Đà Nẵng, Sân bay Phú Quốc đã bị hacker tấn công. Các màn hình bị chèn hình ảnh và câu chữ xúc phạm Việt Nam và Philippines, xuyên tạc về chủ quyền Biển Đông. Website của Vietnam Airlines cũng bị thay đổi giao diện, và tin tặc phát tán hơn 400.000 thông tin hành khách của hãng. Đây được đánh giá là cuộc tấn công lớn nhất từ trước đến nay vào hệ thống thông tin hàng không của Việt Nam.

Sau sự kiện này, nhiều người đặt câu hỏi về tính an toàn và bảo mật thông tin. Và đó là lúc vai trò của những người làm trong lĩnh vực này được quan tâm.

An toàn thông tin là làm gì?

Hiểu một cách đơn giản, an toàn thông tin (ATTT) liên quan đến tính an toàn, bảo mật thông tin, dữ liệu của các hệ thống ứng dụng công nghệ thông tin, trong đó, cần đảm bảo:

- Dữ liệu được an toàn, bảo mật.
- Dữ liệu không bị thay đổi để giữ tính toàn vẹn.
- Dữ liệu, dịch vụ luôn sẵn sàng để sử dụng bất cứ khi nào cần.

Tại Việt Nam, có thể chia các vị trí công việc ATTT thành 3 mảng chính: Tấn công; Phòng thủ và Đảm bảo tính tuân thủ. Cụ thể, tôi sẽ chia sẻ về vị trí công việc của các team tại VNG để bạn hình dung rõ hơn về lĩnh vực này:

1. Red team (tấn công, hack chủ động)

Công việc hằng ngày của team này là tìm lỗi trong những sản phẩm, hệ thống của công ty. Sau đó, họ khai thác lỗi để tấn công xem mức độ xâm nhập vào hệ thống nghiêm trọng đến đâu.

2. Blue team (phòng thủ, xây dựng các hệ thống phòng chống tấn công)

Họ có nhiệm vụ phát hiện những bất thường xảy ra trong sản phẩm, hệ thống. Khi có sự cố, họ sẽ kiểm tra xem có vấn đề gì không? Nếu có vấn đề thì gửi yêu cầu tiếp tục điều tra, xử lý. Team này có kỹ năng điều tra, tìm và phân tích mã độc.

Có thể so sánh như bạn lắp camera trong nhà, khi có báo động, Blue team sẽ ngồi soi camera xem có người lạ đi vào hay không, còn Red Team sẽ cố gắng xâm nhập vào nhà bạn. Trường hợp tốt nhất là phát hiện và ngăn chặn người lạ vào nhà ngay tại thời điểm đó, không để sự cố xảy ra. Trường hợp thứ 2 là người đó đã lọt vào bên trong, lúc này Blue team sẽ kiểm tra xem người này đã đi những đâu, làm những gì? Từ đó, họ tìm cách giảm thiểu thiệt hại.

3. Compliance & Risk Control (Đảm bảo tính tuân thủ)

Chức năng đầu tiên của nhóm này là xây dựng chính sách, quy định và đảm bảo quá trình vận hành tuân thủ đúng luật. Ví dụ khi công ty phát triển các sản phẩm về thanh toán như ZaloPay, team phải tìm hiểu về chính sách, các tiêu chuẩn thanh toán quốc tế, những luật lệ ở Mỹ hay Châu Âu khi áp vào Việt Nam sẽ ra sao? Hoặc quy định của Ngân hàng Nhà nước về đảm bảo ATTT cho thanh toán thẻ như thế nào? Sau khi nắm rõ những quy trình, quy chuẩn đó, họ sẽ tư vấn để team ZaloPay làm sản phẩm cho đúng.

Chức năng thứ hai là Risk control (quản trị rủi ro): nghĩa là xác định rủi ro trong sản phẩm, hệ thống và những gì cần làm để hạn chế nó xảy ra. Đội quản trị rủi ro phải liệt kê được tất cả các nguy cơ có thể xảy ra, như bị hacker tấn công hay mất dữ liệu,... Ngoài ra, phải tính cả những rủi ro nằm ngoài tầm kiểm soát nhưng luôn tồn tại như động đất, cháy nổ,... Họ phải trả lời câu hỏi: Nếu những vấn đề này xảy ra, phải đưa ra giải pháp gì để hạn chế tối đa thiệt hại? Và giải pháp đó đang thực hiện đến đâu? Độ rủi ro sau khi thực hiện giải pháp nằm ở mức nào?

4. R&D (Nghiên cứu và phát triển)

Team này phát triển các giải pháp đảm bảo an toàn thông qua việc xây dựng sản phẩm, các bộ công cụ phục vụ cho ATTT. Trong team có một số kiến trúc sư hiểu biết rộng về ATTT, họ đưa ra định hướng sản phẩm để các bạn phía dưới lập trình. Với các bạn lập trình viên này thì bên cạnh kỹ năng lập trình tốt, các bạn phải có tư duy về bảo mật.

ATTT liên quan đến tất cả các mảng khác trong ngành CNTT. Vì thế, nó rất rộng nhưng cũng rất hẹp: Rộng vì bất cứ sản phẩm nào liên quan tới công nghệ, được xây dựng trên nền tảng Internet đều chứa đựng rủi ro; Hẹp vì trước khi có thể tìm ra/quản trị/xử lý vấn đề bảo mật, chúng ta phải tìm hiểu rất sâu về sản phẩm và công nghệ đó.

Nhiều người cho rằng công việc ATTT chỉ liên quan đến các hệ thống máy tính. Cách hiểu này chưa đủ vì tính bảo mật liên quan đến cả con người (thậm chí con người là điểm yếu nhất). Các "vật thể tĩnh" như máy tính, phần mềm, có thể đảm bảo an toàn thông qua các chính sách, quy trình, nhưng con người thì luôn "động", nay khác mai khác. Khi không tấn công được vào hệ thống, những người vận hành, sử dụng hệ thống đó sẽ là đích nhắm của tin tặc.

Tố chất cần có của người làm ATTT

Khả năng tự học

Trong ngành ATTT, bạn cần nắm các kiến thức về lập trình, hệ điều hành, mạng máy tính, tìm lỗi phần mềm, dịch ngược mã phần mềm, điều tra số, mật mã hóa[1]. Hiện ở Việt Nam mới có 1 - 2 trường đào tạo chuyên ngành về ATTT, và các kiến thức trong trường cũng chỉ ở mức cơ bản, lý thuyết. Trong khi công nghệ phát triển ngày càng đa dạng, sản phẩm và ứng dụng rất bao la rộng lớn. Để có thể đi sâu và trở thành người làm nghề chuyên nghiệp, cách duy nhất

[1] *bạn có thể đọc blog của anh Dương Ngọc Thái (Kỹ sư ATTT tại Google) để tham khảo thêm về kiến thức, các cuốn sách cũng như khóa học về lĩnh vực ATTT: https://vnhacker.blogspot.com*

là bạn tự nghiên cứu độc lập. Nếu bạn muốn trở thành chuyên gia, ngoài việc biết rộng, bạn phải chọn đi rất sâu vào từng chuyên môn, và mỗi người lại chọn một thứ, không ai giống ai. Hơn 20 năm trong nghề, tôi đã đồng hành cùng nhiều thế hệ nhân sự, nhưng tất cả những gì tôi có thể làm chỉ là định hướng cho các bạn, không dạy được nhiều.

Tư duy "think out of the box[1]", khả năng suy luận, liên tưởng và kết nối các vấn đề với nhau (hacking mindset)

Làm ATTT đòi hỏi bạn phải có một số tố chất hơi... "dị biệt". Nếu thám tử Sherlock Holmes có thể nhìn và liên tưởng tới những thứ người khác không thể thì người làm ATTT cũng tương tự, bạn cần khả năng think out of the box, nhạy cảm, suy luận nhạy bén, liên tưởng phong phú.

Giả sử, bạn nhận một đề bài về security, đề bài rất rộng và bạn không biết đi vào bằng cách nào. Nếu thử từng phương pháp một, bạn có thể vẫn tìm ra lời giải nhưng thời gian và sức người chỉ có hạn. Đây là lúc bạn cần sự nhạy bén. Kinh nghiệm của tôi là hãy nhìn nhận vấn đề trên một diện rộng, sau đó dùng "cảm giác" để chọn ra một ngách nhỏ trong đó. Không ít trường hợp có người tìm cả năm trời không ra lỗi, nhưng một người khác mới vào chọn đúng điểm là phát hiện lỗi ngay lập tức.

Các tập đoàn lớn như Google, Microsoft có hàng ngàn nhân viên, kỹ sư hằng ngày tìm lỗi của sản phẩm mà vẫn nhận thêm các báo cáo lỗi từ người ngoài. Đó là minh chứng rõ ràng rằng các vấn đề trong ATTT quá rộng.

Bạn phải có trực giác, nhiều khi tôi gọi là hên (vì nó ngẫu nhiên mà). Chính sự nhạy bén này mới giúp bạn đi xa trong nghề và tạo ra khác biệt.

Kỹ năng giao tiếp

Đặc thù của nghề ATTT là chỉ ra vấn đề của người khác, hệ quả là bạn dễ gặp mâu thuẫn trong công việc. Đặc biệt ở Việt Nam, nơi văn hóa làm việc còn rất trọng tình cảm.

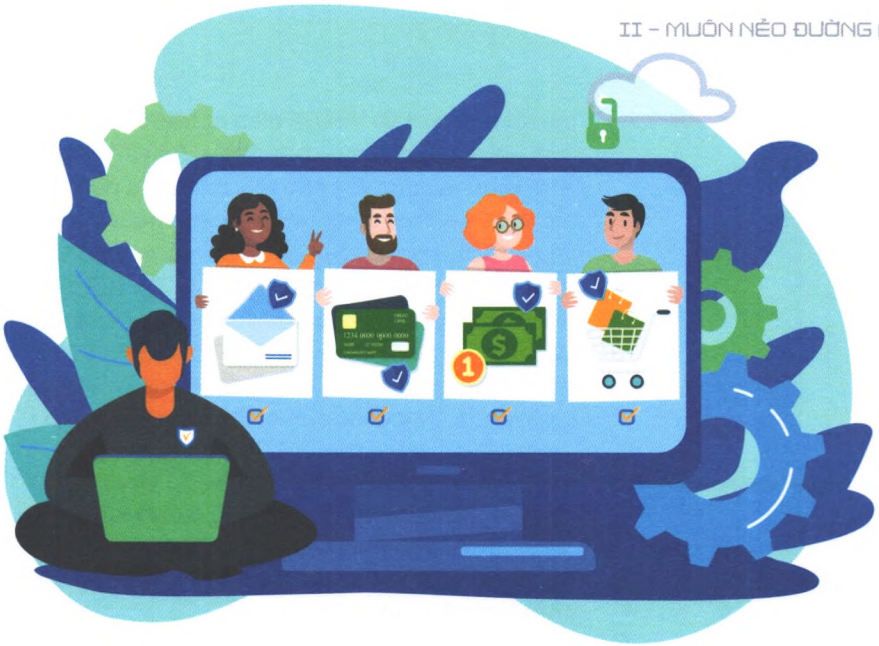
Bạn là người kẹt giữa sếp và các team làm sản phẩm. Một mặt, khi phát hiện ra lỗi, trách nhiệm của bạn phải báo cáo cho cấp trên. Mặt khác, bạn cần thuyết phục team làm sản phẩm thay đổi, chỉnh sửa lại cho phù hợp. Nhưng thuyết phục đâu phải chuyện dễ dàng khi trong mắt họ, bạn là một "gã săm soi khó tính, chuyên đi mách sếp".

Bạn sẽ phải tìm cách làm sao để cân bằng giữa việc phát triển sản phẩm và đảm bảo tính an toàn? Làm sao thuyết phục được các lãnh đạo về tầm quan trọng của ATTT? Làm sao cho các team sản phẩm hiểu rằng những lỗ hổng bạn chỉ ra thuần túy là công việc, không phải công kích cá nhân?

Đam mê

Hiện tại, lĩnh vực ATTT có mức thu nhập trung bình cao hơn các mảng khác trong ngành IT do mức độ nhận thức của thị trường đã thay đổi so với trước. Sau sự kiện Vietnam Airlines bị tấn công hay nhiều ngân hàng bị hack mất tiền, một số tập đoàn lớn như Viettel, VinGroup tách ra làm các công ty về security, từ đó đẩy lương mặt bằng chung của ngành này lên cao. Nhưng nếu lựa chọn ATTT chỉ vì mức lương, bạn hãy dè chừng.

[1] *tư duy vượt giới hạn*



Bước chân vào lĩnh vực này không quá khó: các bạn biết lập trình, tìm hiểu rồi đọc hướng dẫn là có thể "phá phách" chỗ này chỗ kia. Thế nhưng, để đạt được mức chuyên nghiệp thì không đơn giản vì ngành này đòi hỏi sự tập trung cao độ và quyết tâm tìm hiểu tới cùng về một lĩnh vực.

Công việc security đòi hỏi làm việc suốt ngày đêm, ngày này qua ngày khác bạn cần đi chuyên sâu một vài thứ. Thông thường, những vấn đề bạn gặp đều mới mẻ. Đôi khi, bạn sẽ rơi vào tình trạng không biết làm gì để giải quyết chúng? Bạn cần kiên trì thử nghiệm mọi ngóc ngách để tìm ra lời giải. Như vậy, bên cạnh các kỹ năng think out of the box, nghiên cứu độc lập,... đam mê sẽ giúp bạn không bỏ cuộc.

Đa số những người làm ATTT ở Việt Nam hiện nay không tốt nghiệp chuyên ngành này mà học lập trình, hệ thống mạng, các ngành về CNTT nói chung,... Sau đó, vì đam mê nên họ lựa chọn ATTT và tự học. Team security của VNG tập hợp đủ những con người có xuất phát điểm khác nhau: có người học Đại học Bách Khoa ngành Hóa, ra trường làm... pha sữa tại Vinamilk sau đó mới chuyển sang ATTT, có nhiều người còn chưa tốt nghiệp Đại học.

Các bạn trẻ đừng bỏ qua các cuộc thi về bảo mật

ATTT có rất nhiều mảng hẹp khác nhau, bạn chỉ nên chọn vài hướng đi chuyên sâu. Như bạn đã biết, ATTT không chỉ xoay quanh những người tìm lỗi, mà còn cả những người xây dựng giải pháp bảo vệ, người phòng thủ chống lại những cuộc tấn công,... Mỗi một vị trí lại có lộ trình phát triển và yêu cầu kỹ năng khác nhau, không có một công thức chung nào cho tất cả. Nó phụ thuộc rất nhiều vào định hướng của bạn.

Lời khuyên cho các bạn sinh viên là nên tham gia các cuộc thi về bảo mật công nghệ thông tin như Capture The Flag (CTF)[1]. Lần đầu tiên được tổ chức tại hội thảo bảo mật nổi tiếng DEFCON (Mỹ) năm 1997, đến nay, hằng năm

[1] *cuộc thi kiến thức chuyên sâu về bảo mật máy tính, được tổ chức theo mô hình trò chơi chiến tranh mạng, tập trung vào kỹ năng tấn công và phòng thủ mạng máy tính của người tham gia.*

có rất nhiều cuộc thi CTF diễn ra trên toàn thế giới với nhiều quy mô khác nhau. Tại Việt Nam, CTF thường được tổ chức quy mô trong nước và có thể thi online để cọ xát với team nước ngoài. Theo tổ chức CTFtime, Việt Nam nằm trong top 10 nước có nhiều đội tham gia thi CTF nhất. Một số nhóm CTF có thành tích được cộng đồng bảo mật thế giới biết đến như: Cửu Long Giáng Thế (CLGT, Bamboo-vn), CLGT$MeePwn PiggyBird, BabyPhd,... Ngoài ra, còn có 2 cuộc thi quy mô quốc tế là "Sinh viên với ATTT" do Hiệp hội An toàn Thông tin Việt Nam (VNISA) tổ chức (được công nhận là cuộc thi cấp quốc gia và năm 2019 đã mở rộng ra khu vực ASEAN) và cuộc thi WhiteHat Grand Prix do Bkav chủ trì (năm 2018 mở rộng với quy mô toàn cầu).

Bạn cũng đừng nghĩ chỉ 1 - 2 tháng là thành tài, hãy xác định đi thi vài năm mới thực sự tích lũy đủ kiến thức. Và bạn chỉ làm được việc đó nếu đam mê.

Đối với CTF, điều tuyệt vời nhất không phải là bạn giải được đề thi, mà nằm ở hành trình tìm kiếm lời giải. Ví dụ, đề bài yêu cầu bạn tìm lỗi của một chiếc khóa. Trước hết, bạn phải biết cái khóa đó được thiết kế và lập trình như thế nào? Nó sử dụng chip gì? Kết nối, giao tiếp ra làm sao? Bạn sẽ phải bóc tách và phân tích rất nhiều thứ trong chiếc khóa đó. Những kỹ năng và kiến thức bạn thu thập được trong quá trình này vô cùng bổ ích.

Ngoài ra, bạn còn được xem rất nhiều cách giải của những người khác để đa dạng hóa góc nhìn của bản thân về một vấn đề. Bạn sẽ hiểu tư duy của mình đã diễn biến như thế nào trong quá trình đi tìm lời giải? Tại sao người kia lại nhìn ở góc độ khác mình?

Trong cuộc thi, bạn sẽ gặp vấn đề ở nhiều mảng khác nhau, và bạn có thể tìm được mảng phù hợp với hướng đi của mình về lâu dài. Bên cạnh đó, bạn có cơ hội gặp và giao lưu với những người cùng chung đam mê và sở thích. Đôi khi bạn gặp người "tâm đầu ý hợp", nghĩ ra được ý tưởng, cùng làm một vài nghiên cứu, biết đâu sẽ thu được thành quả.

Với sinh viên, tôi nghĩ tham gia cuộc thi là cách học nhanh và bài bản. Bạn cũng đừng nghĩ chỉ 1 - 2 tháng là thành tài, hãy xác định đi thi vài năm mới thực sự tích lũy đủ kiến thức. Và bạn chỉ làm được việc đó nếu đam mê.

Dù công việc không hào nhoáng, nhưng chúng tôi vẫn vui và tự hào vì những đóng góp thầm lặng của mình đang giúp đảm bảo an toàn cho hàng triệu người dùng.

Liệu bạn có muốn làm một nghề không bao giờ biết đến thành công?

Nghe có vẻ phũ phàng nhưng thực sự, làm ATTT, tôi không biết thế nào gọi là thành công. Khi sự cố xảy ra, vậy là vai trò đảm bảo an ninh, an toàn của tôi và đội ngũ đã thất bại. Còn nếu mọi chuyện diễn ra êm đềm, chúng tôi sẽ hoài nghi: "Liệu không có vấn đề hay có vấn đề đấy nhưng mình không tìm ra?". Chúng tôi chỉ biết đã làm tất cả những gì có thể, chứ không có khái niệm "thành công" hay "xuất sắc". Dù công việc không hào nhoáng, nhưng chúng tôi vẫn vui và tự hào vì những đóng góp thầm lặng của mình đang giúp đảm bảo an toàn cho hàng triệu người dùng. Tôi cho rằng, đã chọn nghề thì phải chấp nhận không thể có sự hoàn hảo, dù ở quy mô một công ty hay bên ngoài xã hội cũng vậy.

Công nghệ đang ngày càng hoàn thiện và trở nên an toàn hơn, vì thế, lĩnh vực ATTT đòi hỏi kỹ năng ngày càng cao. Tuy nhiên, khi số lượng các công ty ứng dụng hệ thống công nghệ thông tin ngày càng lớn và đời sống con người ngày càng phụ thuộc vào công nghệ, vai trò của ATTT ngày càng trở nên rõ nét hơn bao giờ hết.

Tại VNG, đội security tuyển dụng những sinh viên từ năm 2, năm 3. Nhiều sinh viên chưa tốt nghiệp đã được ký hợp đồng làm kỹ sư chính thức. Điều này là bình thường vì chỉ cần bạn làm được, độ tuổi hay bằng cấp không còn quan trọng. Khi tuyển dụng, chúng tôi không chọn người có CV đẹp mà chọn người làm được việc và có định hướng phát triển của bản thân phù hợp với công ty.

Nếu bạn có niềm yêu thích với những câu chuyện xoay quanh bảo mật, sẵn sàng làm một nghề mà chẳng bao giờ được ca tụng về thành công, vậy ATTT có lẽ là sự lựa chọn đáng lưu tâm.

IT Outsourcing
Khi công việc hỗ trợ là trải nghiệm tuyệt vời

Khách mời phỏng vấn: **Giang Lê**
Operation Manager[1] Tek Experts
Biên soạn bài viết: **Ban biên tập**

Phần lớn mọi người hình dung công việc trong ngành CNTT có liên quan tới các vị trí phát triển phần mềm. Tuy nhiên làm "công nghệ" không nhất thiết là bạn luôn phải tạo ra các sản phẩm mới, mà việc duy trì, vận hành, và đảm bảo các hệ thống sẵn có luôn tối ưu hiệu quả hoạt động và phát sinh ít sai sót cũng không kém quan trọng. Trong bài viết này, anh Giang Lê từ Tek Experts (công ty chuyên cung cấp các dịch vụ hỗ trợ công nghệ thông tin - ITO và Professional Services) muốn chia sẻ với các bạn trẻ một thông điệp: Nếu bạn không chỉ đam mê công nghệ mà còn mong muốn giúp đỡ người khác giải quyết những vấn đề khó khăn của họ, ITO là một lĩnh vực đầy tiềm năng dành cho bạn.

Những "hiệp sĩ" giấu mặt

Năm 2014, trong một ngày làm việc bình thường, chúng tôi bỗng nhận được yêu cầu khẩn cấp từ phía khách hàng là một hãng hàng không. Phần mềm quản trị dịch vụ doanh nghiệp BSM (Business Service Management) của họ phát sinh lỗi môi trường production[2], khiến cho liên lạc giữa máy bay và trạm không lưu mặt đất bị gián đoạn. Sự cố này ảnh hưởng trực tiếp tới việc máy bay hạ cánh, và khiến cho hàng trăm con người đang ở "trên trời" phải bay lòng vòng để chờ kết nối với bộ phận mặt đất.

[1] *Quản lý vận hành*

[2] *"Môi trường" là một khái niệm trong ngành lập trình, bao gồm các yếu tố phần cứng, phần mềm, dữ liệu và các thiết lập cơ bản. Môi trường production là nơi chứa ứng dụng thật đang chạy với dữ liệu và người dùng thật.*

Nếu phải tham gia xử lý sự cố này, bạn sẽ suy nghĩ gì và hành động như thế nào? Mỗi phút giây bị lãng phí đồng nghĩa với việc rủi ro cho toàn bộ hành khách và phi hành đoàn tăng lên, chưa tính những thiệt hại kinh tế của hãng hàng không. Sự an toàn và uy tín của rất nhiều con người đang đặt lên vai bạn - người làm nhiệm vụ hỗ trợ kỹ thuật, hay còn gọi là IT support.

Đây chỉ là một trong nhiều kỷ niệm rất đáng nhớ của tôi và các đồng nghiệp tại Tek Experts. Ngày hôm đó, sau nhiều giờ tập trung căng thẳng, cuối cùng, đội ngũ chúng tôi đã phối hợp thành công cùng nhiều đồng nghiệp từ nhiều đơn vị khác trên thế giới, góp phần xử lý thành công lỗi hệ thống. Rất nhiều thiệt hại về vật chất có thể xảy ra đã được ngăn chặn kịp thời, nhưng quan trọng hơn cả là đảm bảo an toàn tính mạng của hàng trăm hành khách. Vậy thực sự công việc IT support/IT outsourcing là làm gì?

Hiểu một cách tổng quan, IT là lĩnh vực rất đa dạng, và mảng dịch vụ hỗ trợ hay còn gọi là ITO (Information Technology Outsourcing) chỉ là một mảnh ghép trong bản đồ ngành rộng lớn đó. Bạn hãy hình dung một tập đoàn lớn thường có rất nhiều sản phẩm khác nhau với hàng triệu người dùng trên khắp thế giới. Có quá nhiều nghiệp vụ kỹ thuật và phi kỹ thuật đi kèm khiến bản thân các doanh nghiệp này không thể tự mình xử lý hết được các vấn đề phát sinh. Do đó, để có thể tập trung vào những mảng kinh doanh cốt lõi, họ cần một bên thứ ba tham gia vào việc hỗ trợ vận hành, bảo trì hệ thống, xử lý các lỗi kỹ thuật phần mềm, chăm sóc và giải đáp các thắc mắc từ khách hàng,... Các bên thứ ba đó là những đơn vị cung cấp dịch vụ ITO.

Đối tác của chúng tôi thường là các công ty công nghệ lớn nhất trên thế giới và chúng tôi sẽ giúp khách hàng của họ - những người đang sử dụng các phần mềm dùng cho cá nhân (Windows, Office 365) hay các tổ chức đang sử dụng phần mềm doanh nghiệp như Dynamic 365, các gói phần mềm doanh nghiệp của Microfocus như Business Service Management, Network Node Manager, Data Protector, Functional Testing,... - giải quyết các vấn đề mà họ gặp phải. Trong khuôn khổ bài viết này, tôi muốn tập trung vào lĩnh vực hỗ trợ kỹ thuật (mảng chuyên môn của tôi). Tuy nhiên, bạn cũng đừng quên rằng ITO còn có những vị trí hỗ trợ khách hàng khác không đòi hỏi quá nhiều chuyên môn kỹ thuật.

Với tính chất hỗ trợ vận hành như vậy, có thể hình dung công việc của những người làm ITO chạm tới rất nhiều lĩnh vực chuyên môn khác nhau, từ mạng lưới (networking), dữ liệu (database), bảo mật (security) cho tới điện toán đám mây (cloud),... Do đó, các kỹ sư trong lĩnh vực IT support cũng sẽ phải là chuyên gia trong các mảng họ phụ trách. Ví dụ, nếu làm về bảo mật, bạn sẽ cần kiến thức về Linux, cần đạt được các chứng chỉ CNTT như CCNA (Cisco Certified Network Associate) hoặc CEH (Certified Ethical Hacker); làm về database hay cloud thì cần kiến thức về SQL, Oracle, hay chứng chỉ MCSA (Microsoft Certified Solution Associate),...

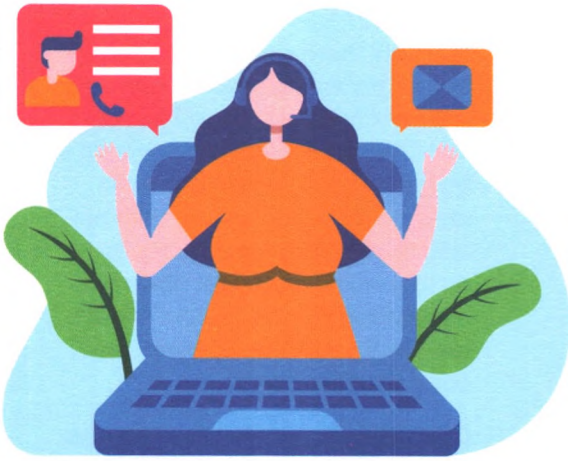
Nhiệm vụ của người làm IT support là giúp các hệ thống phần mềm phức tạp của khách hàng luôn hoạt động trơn tru và hiệu quả. Giả sử, khách hàng khi sử dụng phần mềm gặp phải một lỗi nào đó, họ sẽ tạo một "ticket", đội ngũ IT support nhận ticket này và giúp đỡ khách hàng xử lý lỗi đó nhanh chóng,

hiệu quả nhất. Yếu tố nhanh và hiệu quả luôn cần phải song hành, bởi đối với các tập đoàn, doanh nghiệp lớn, chỉ cần hệ thống phần mềm hoặc thiết bị đình trệ trong một vài giờ cũng có thể kéo theo thất thoát doanh thu lên tới vài triệu đô la. Giống như những "hiệp sĩ" giấu mặt, người làm IT support không mấy khi lộ diện nhưng lại thay mặt cho các đối tác xử lý nhiều vấn đề phức tạp.

Khác với các vị trí phát triển sản phẩm trong ngành công nghiệp phần mềm (đôi khi bạn chỉ cần đào sâu vào một mảng hoặc một bộ kỹ năng cụ thể như PHP, Java, .Net,... là có thể làm tròn vai), vị trí technical support cần kiến thức tổng quan rộng hơn và am hiểu nhiều lĩnh vực hơn. Tuy nhiên, điều đó không có nghĩa bạn chỉ cần nắm chuyên môn hời hợt ở từng mảng, bởi khi tiếp xúc với các vấn đề kỹ thuật, lỗi có thể xuất hiện ở một phần nhưng nguyên nhân sâu xa lại có thể nằm ở phần khác. Nếu không hiểu rộng và hiểu đủ sâu, người làm technical support sẽ không thể phán đoán được chính xác vấn đề đến từ đâu. Ngoài ra, vị trí này cũng đòi hỏi bạn có năng lực phản ứng nhanh với các tình huống, khả năng học hỏi tốt để nắm bắt được những bài toán mới. Trải qua nhiều năm kinh nghiệm, sự nhạy bén với các vấn đề của bạn sẽ gia tăng đáng kể.

Con đường phát triển của vị trí IT support

Tuỳ thuộc vào từng công ty mà cấu trúc nhân sự hay lộ trình thăng tiến trong lĩnh vực ITO có nhiều cách phân chia và tên gọi khác nhau. Về cơ bản, hướng đi trong mảng này có thể được chia thành 3 nhánh, phù hợp với các thiên hướng khác nhau. Tôi sẽ đưa ví dụ cụ thể tại Tek Experts để các bạn dễ hình dung:



- *Hướng thuần kỹ thuật (technical)*: có thể phân thành từng cấp bậc là Technical Support Engineer Level 1, Technical Support Engineer Level 2, Technical Support Engineer Level 3, Technical Lead, Subject Matter Expert;

- *Hướng kinh doanh (business enablement)*: sau khi hiểu và nắm vững về công việc Technical Support Engineer, bạn có thể phát triển theo hướng Đào tạo (trainer) hoặc Đảm bảo chất lượng (QA - Quality Assurance);

- *Hướng quản trị (management)*: các vị trí quản lý bắt đầu từ Service Delivery Manager, Team Manager cho đến Section Manager, Operation Manager.

Khi bắt đầu bước chân vào Tek Experts với vai trò kỹ sư Level 1, tôi đã trang bị cho mình một số chứng chỉ như CCNA (Cisco Certified Network Associate), CCNP (Cisco Certified Network Professional). Vì thế, tôi có thể ngay lập tức tham gia hỗ trợ khách hàng trong mảng hạ tầng, mạng lưới một cách tương đối thuận lợi. Tuy nhiên, thông thường với các vị trí tech support ở level ban đầu, bạn không nhất thiết phải có bằng cấp hoặc kinh nghiệm về

IT để làm việc. Yêu cầu cơ bản nhất là Tiếng Anh ở mức khá và có sự yêu thích công nghệ, cộng với tinh thần học hỏi, mong muốn giúp đỡ người khác. Khi gia nhập công ty, bạn sẽ được hướng dẫn những kiến thức chuyên môn cơ bản và học cách sử dụng các công cụ giúp phát hiện lỗi (troubleshoot), công cụ giao tiếp với khách hàng (ví dụ như phần mềm Webex) thông qua khóa đào tạo ban đầu (onboard training). Song song trong suốt quá trình làm việc, bạn sẽ tiếp tục nhận được hướng dẫn trực tiếp từ các "tiền bối", gọi là on-job training.

Để trở thành một người quản lý, bạn cần có kiến thức chuyên môn tương đối bao quát và quan trọng hơn là các kỹ năng về mặt con người: kỹ năng giao tiếp, trao đổi và thuyết phục khách hàng; kỹ năng truyền đạt, giải thích và hướng dẫn cho team nội bộ. Khả năng tương tác và phối hợp là một yêu cầu đặc biệt quan trọng.

Thuận lợi và khó khăn trong công việc IT support

Bất kỳ công việc nào cũng có những thuận lợi và khó khăn riêng, và mảng technical support không phải ngoại lệ. Bạn có bao giờ hình dung một ngày mình sẽ được truy cập vào hệ thống của các tập đoàn đa quốc gia hay các ngân hàng toàn cầu chưa? Sẽ khó có công việc nào cho bạn trải nghiệm thú vị như vậy: không chỉ tiếp cận cơ sở dữ liệu khổng lồ với hàng triệu khách hàng, bạn còn góp phần giải quyết những lỗi hệ thống giúp giảm thiểu thiệt hại của nhiều người, tiết kiệm nhiều tiền của. Ví dụ với một ngân hàng lớn, khi phần mềm bảo vệ dữ liệu (Data Protector) bị lỗi, ngân hàng sẽ không thể lưu lại các giao dịch trong ngày, đồng nghĩa với việc sao kê của khách hàng sẽ "biến mất", gây thiệt hại rất lớn. Cảm giác xử lý được những tình huống như vậy thực sự rất "cool ngầu".

Một ưu điểm lớn khác của công việc IT support còn đến từ cơ hội làm việc trong môi trường quốc tế. Khách hàng của chúng tôi đến từ khắp nơi trên thế giới, còn đồng nghiệp là những "công dân toàn cầu" từ Mỹ, Trung Quốc, Costa Rica, Bulgaria,... Dù muốn hay không bạn cũng phải sử dụng Tiếng Anh liên tục và do đó, trình độ ngoại ngữ được cải thiện đáng kể. Kỹ năng chuyên môn về sản phẩm được nâng cao qua từng nhiệm vụ (case) nhờ các "tiền bối" đi trước trực tiếp hỗ trợ; kỹ năng tư duy phản biện cũng sắc sảo hơn do đối mặt với nhiều trải nghiệm thực tế. Khác với hình dung của đa số mọi người rằng công việc tech support phải làm ngày làm đêm, ở Tek Experts, phần lớn thời gian chúng tôi chỉ làm việc trong khung giờ hành chính. Nếu hết ngày vẫn còn các vấn đề của khách hàng chưa được giải quyết, team sẽ bàn giao lại công việc cho đội ngũ ở múi giờ khác tiếp tục thực hiện. Thông thường, buổi sáng team Việt Nam sẽ nhận case từ Mỹ, buổi chiều bàn giao cho các bạn Châu Âu.

Tất nhiên không phải lúc nào công việc IT support cũng vui vẻ và màu hồng. Việc tiếp xúc với khách hàng cùng đội ngũ từ khắp nơi trên thế giới cũng phát sinh các yếu tố mâu thuẫn như văn hoá, rào cản ngôn ngữ, khác biệt trong cách tiếp cận vấn đề,... Những case khó kéo dài trong vài ngày sẽ liên đới tới rất nhiều team khác nhau và chỉ riêng việc trao đổi qua lại cũng đã tương đối tốn thời gian. Đôi khi, bạn sẽ gặp rất nhiều áp lực bởi chỉ một sai sót nhỏ có thể dẫn tới hậu quả khó lường, trong khi vấn đề đòi hỏi phải được xử lý hết sức nhanh chóng dưới sức ép về thời gian.

Công việc hỗ trợ khách hàng yêu cầu khả năng giao tiếp và kết nối với con người, do đó, bạn phải có sự nhẫn nại, lắng nghe và không được... "nổi khùng" bất kể yêu cầu của khách hàng là gì, dù họ là một người dùng cá nhân đang loay hoay chưa biết cách cài Windows hay một vị CTO[1] tai to mặt lớn. Đôi khi, việc này sẽ là thử thách không nhỏ với các bạn xuất thân thuần kỹ thuật, không giỏi giao tiếp, trò chuyện. Trong những tình huống khó khăn như vậy, kinh nghiệm của tôi là bạn hãy đặt mình vào vị trí khách hàng, hiểu và thông cảm với khó khăn của họ, từ đó, cố gắng hết sức để đưa ra giải pháp tốt nhất. Cho dù bạn có giải quyết được vấn đề ngay lập tức hay không thì một thái độ cầu thị, chân thành và khả năng giao tiếp tốt sẽ khiến cho khách hàng cảm thấy hài lòng.

"Mảnh đất vàng" còn nhiều tiềm năng và thỏa sức sáng tạo

Đến đây chắc các bạn đã hiểu, bên cạnh một phần lớn các doanh nghiệp đi theo con đường phát triển phần mềm, ITO cũng là lĩnh vực rất đáng thử sức. Tek Experts đã gia nhập thị trường Việt Nam từ năm 2013 và là một trong những đơn vị tiên phong về ITO. Khoảng vài năm trở lại đây, một số doanh nghiệp lớn khác cũng tham gia mảng này, như CMC hay FPT Software,...

Trên thế giới, lĩnh vực ITO đã cực kỳ phát triển. Theo báo cáo Spotlight on Viet Nam[2] của PwC, mảng dịch vụ thuê ngoài doanh nghiệp - BPO (Business Process Outsourcing, trong đó ITO chiếm tỷ trọng đáng kể) của Việt Nam năm 2015 đạt doanh thu 2 tỷ USD. So sánh với các quốc gia láng giềng - Philippines doanh thu 22 tỷ USD, Ấn Độ 143 tỷ USD,... có thể thấy thị trường Việt Nam còn rất nhiều tiềm năng tiến xa hơn.

Trong báo cáo của A.T.Kearney[3] năm 2019, chỉ số Dịch vụ Toàn Cầu theo quốc gia (Global Services Location Index) của Việt Nam xếp hạng 5 trong các thị trường mới nổi, với tốc độ tăng trưởng từ 20 - 25% mỗi năm. Gần đây, Việt Nam đã vượt qua Trung Quốc, trở thành đối tác outsource phần mềm lớn nhất của Nhật Bản. Những con số này cho thấy mảng IT support còn nhiều triển vọng phát triển, đặc biệt ở môi trường mới mẻ như Việt Nam. Dù lựa chọn các vị trí kỹ thuật hay phi kỹ thuật, các bạn cũng có cơ hội lớn để tìm được một công việc với mức đãi ngộ hấp dẫn.

Thỉnh thoảng, tôi vẫn nhận được thắc mắc của các bạn trẻ rằng mảng ITO chuyên về hỗ trợ khách hàng, hỗ trợ vận hành, liệu rằng có nhàm chán hơn các công việc liên quan tới phát triển sản phẩm hay không?

Xin được trả lời là không, "sáng tạo" hay "tối tạo" nằm rất nhiều ở thái độ chủ động và tư duy giải quyết vấn đề của bạn chứ không nằm ở đặc thù

---

[1] *Chief Technology Officer: Giám đốc Kỹ thuật*

[2] *PwC. 2017. Spotlight On Viet Nam. [online] Available at: <https://www.pwc.com/vn/en/publications/vietnam-publications/spotlight-on-vietnam.html> [Accessed 22 June 2020].*

[3] *Sethi, A., Gott, J. and Suman, V., 2019. The 2019 Kearney Global Services Location Index - Kearney. [online] Kearney.com. Available at: <https://www.kearney.com/digital-transformation/gsli/2019-full-report> [Accessed 22 June 2020].*

công việc. Người làm phát triển phần mềm mà có tư duy "copy", "clone" lại những sản phẩm có sẵn trên thị trường thì cũng không phải là sáng tạo. Ngược lại, người làm IT support nhưng luôn nỗ lực thử nghiệm để tìm ra giải pháp tối ưu trong việc xử lý lỗi, giúp tiết kiệm thời gian, công sức và tiền bạc cho khách hàng thì chính là sáng tạo. Chúng tôi thường tổ chức cuộc thi tên là "Amazing Race" kéo dài trong vòng 6 tháng để đội ngũ nhân sự "thi thố", đưa ra các sáng kiến trong việc hỗ trợ khách hàng hiệu quả nhất. Tôi nghĩ, cảm giác nhàm chán là dấu hiệu cho thấy bạn đang ở trong tình trạng không phát triển bản thân, chứ không liên quan tới việc bạn đang làm gì.

Đã hơn 7 năm theo đuổi công việc này, tôi luôn cảm thấy mỗi ngày là một thử thách hoàn toàn mới, một cột mốc mới vì đặc thù công việc luôn đòi hỏi xử lý các vấn đề của các đối tượng khác nhau: từ các tập đoàn hàng đầu thế giới cho đến những cụ ông, cụ bà không biết cách sử dụng phần mềm Word. Kiến thức và khả năng giao tiếp liên tục được mài giũa sắc bén hơn; nhưng điều quan trọng nhất là tôi luôn có cảm giác được giúp đỡ người khác mỗi ngày và nhận được sự trân trọng từ họ. Đó chẳng phải là niềm vui rất lớn hay sao?

"sáng tạo" hay "tối tạo" nằm rất nhiều ở thái độ chủ động và tư duy giải quyết vấn đề của bạn chứ không nằm ở đặc thù công việc.

NHÀ TÔI CÓ NUÔI MỘT ANH DEV

Tác giả: **Thu Phạm**
Vợ anh dev Spiderum

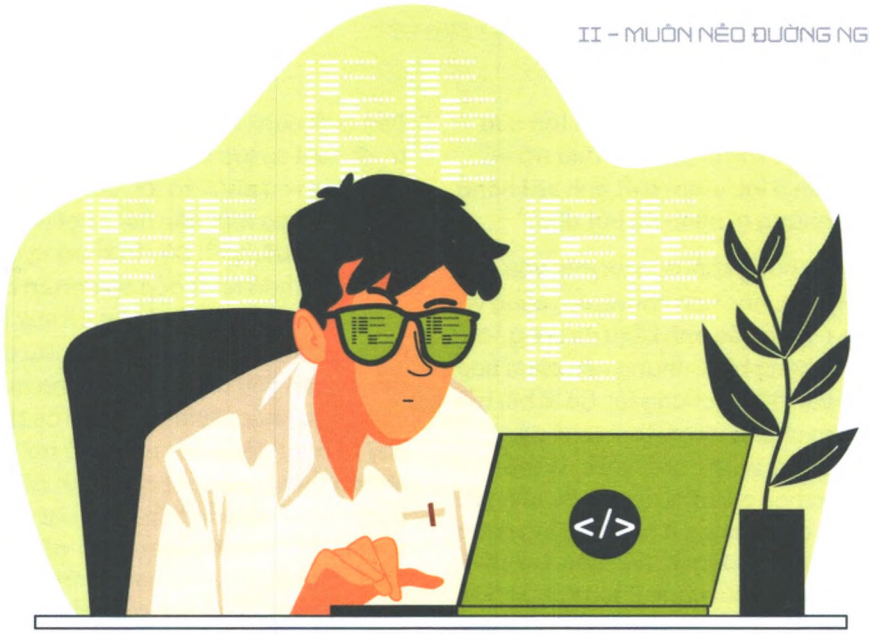
Có lẽ chúng ta thường nghĩ đến lập trình viên (dev) như những người khô khan, thường ngồi nhiều tiếng đồng hồ gõ những dòng code khó hiểu. Thế nhưng, ra khỏi công việc, họ cũng là những người chồng, người cha, người yêu, cũng có cảm xúc và thật nhiều những trắc trở đời thường. Hãy cùng chị Thu Phạm - một người vợ có chồng làm dev - khám phá thêm những khía cạnh "rất đời" của họ nhé!

Nhà tôi có nuôi một anh dev. Dáng người anh dong dỏng ngoài 1m70. Khi được hỏi anh luôn trả lời mình 1m72, như thể 2cm là một trò trống gì đó rất hệ trọng. Ngoài công việc chính là ngồi trước một chiếc màn hình đen sì, tành tạch gõ cả bài sớ dài như tấu chương cho Khang Hy thì anh còn là người yêu, là bạn thân, là chồng và là bố của hai đứa con tôi.

Nói chung, nhà tôi có nuôi một anh dev.

Khi tôi đang viết những dòng này thì anh dev của tôi vẫn đang ngồi "gõ sớ", còn đồng hồ đã điểm 2 giờ sáng. Chỉ mới cách đây 6 năm, anh vẫn còn sinh hoạt kiểu ông-cụ-bảy-mươi: 9 giờ tối đi ngủ, 7h sáng thức giấc và chăm chỉ tập thể dục như thể sợ nay mai sẽ ốm. Thế mà giờ anh thường làm việc đến 3 giờ sáng, thức giấc đâu đó vào khoảng 8 rưỡi, ngủ tranh thủ thêm vài giấc vặt vãnh trong ngày. Có lẽ đây gọi là tiến hóa ngược.

Người ta thường nghĩ gì khi nhắc đến một developer? Mắt cận, suốt ngày cắm mặt vào máy tính, khô khan và giao tiếp dở. Về cơ bản thì người tôi yêu đúng là như thế. Kính mắt dày ngang đít chai, thời gian dành cho máy tính nhiều hơn cả thời gian ăn với ngủ cộng lại, ăn nói (có vẻ) thật thà quá mức và rõ ràng không biết bày tỏ tình cảm thế nào cho phải.



Thế nhưng đâu chỉ có thế. Anh dev của tôi rất thích đọc truyện tranh. Tôi ngờ rằng chỉ vài năm nữa thôi, khi chồng tôi 35 tuổi còn con tôi 6 tuổi, có lẽ bố nó sẽ tranh đọc cả truyện của nó. Trước khi quen tôi khoảng 1 năm, anh cũng bắt đầu tự học chơi piano. Anh tập luyện không ngừng nghỉ cho đến khi đánh thuần thục một bài. Anh cũng chăm tập gym và chống đẩy vì muốn sở hữu thân hình vừa vặn với những chiếc sơ mi slimfit. Cho đến khi yêu tôi, anh còn học luôn cả thói cày phim truyền hình Mỹ dài tập nữa. Những 10 mùa phim Friends, How I met your mother, How to get away with murder, Breaking Bad,... anh đều xem trọn. Dev thì cũng như ai cả, cũng đều có những sở thích rất đời thường và đôi khi... rất con trẻ.

Sau 2 năm yêu nhau, anh quyết định nhảy việc. Có một ngày anh hỏi tôi: "Anh nên làm một công ty nổi tiếng với số lương rất cao nhưng bản chất công việc outsource không có gì mới mẻ hay nên làm một công việc mới mẻ nhưng lương không quá cao?". Tôi nói vốn anh đã có câu trả lời rồi, anh chỉ cần tôi khẳng định lại mà thôi. Vậy là anh chọn Spiderum.

Những ngày đầu đi làm về, anh bảo anh đang làm một nền tảng cho người thích viết lách, "hợp với em quá còn gì, anh làm cho em đấy". Chuyện làm cho tôi hay cho ai còn chưa hạ hồi phân giải thì tôi bị lôi đầu ra test sản phẩm. Một mạng xã hội cho người viết lách thì phải cho người hay viết test rồi còn gì, anh nghĩ thế như một lý do để ngang nhiên "tuyển dụng" tôi làm tester... không lương. Spiderum bắt đầu trở thành một ấp ủ của chính anh, với mong muốn nền tảng phát triển thành một mái nhà nơi thành viên có thể tự do viết và chia sẻ những kiến thức hữu ích.

Cuộc sống văn phòng của anh tại công ty mới rất vui: Nhân sự thì trẻ trung, "nhây" và rất dễ chịu thoải mái. Tôi nhớ thời điểm đó anh thường kể rất hào hứng về công việc anh phụ trách cũng như đồng nghiệp cùng chiến tuyến. Không may là chỉ sau 2 đến 3 tháng, Spiderum gặp khó khăn về vốn đầu tư. Anh lại một lần nữa đứng trước 2 lựa chọn: hoặc chuyển sang một dự án khác để đảm bảo mức lương hiện tại, hoặc ở lại cùng Spiderum với một số lương phù hợp với khả năng tài

chính của team hơn. Lại một lần nữa anh hỏi tôi. Tôi lại một lần nữa trả lời: "Anh nên ở lại, vì em biết anh sẽ không bỏ lại những gì mình đã bắt đầu."

Cả team hứng khởi dọn tới một văn phòng mới, nhỏ hơn nhưng ấm cúng vô cùng. Lương của anh kể từ đó cũng "ấm cúng" không kém, nhưng tiền chưa bao giờ là vấn đề với chúng tôi. Có lẽ bởi trời phú cho chúng tôi cái thói "chê tiền", có nhiều xài nhiều, có ít xài ít. Vì thế, tôi chưa bao giờ phải động viên anh về chuyện đồng lương ít ỏi, anh cũng chưa từng kêu ca không đủ tiền tiêu. Đói thì rủ nhau đi ăn bát cháo Bách Khoa chưa đến 10k, lại còn hào phóng rủ người yêu ăn thêm bát nữa: "Anh trả"; khát nước thì ngồi vỉa hè gọi 2 cốc nước mía, nói vài ba câu chuyện thường ngày. Mọi thứ vô cùng đơn giản như vậy thôi.

Đến năm thứ hai tập trung cho Spiderum, anh dev của tôi đã trở thành người đúng chuẩn của một startup. Thời điểm đó tôi bắt đầu bận bịu với công việc của riêng mình, thức giấc vào 7h sáng và chỉ về nhà khi đồng hồ điểm 8h. Bởi vậy, thay vì có tôi nấu cơm trưa cho anh đem đi, giờ anh phải tự lo cái bụng đói của mình. Không có tôi, anh chọn cách sống khổ hạnh như một nhà tu chân chính: Sáng dậy chạy xe ra đầu ngõ mua chiếc bánh mì pate trứng 10k đem đến văn phòng ăn sáng, gần trưa úp một bát mì tôm "không người lái", tráng miệng bằng 1- 2 gói Ostar kim chi. Đến chiều tối, khi tâm trí bắt đầu trở nên mờ mịt và có dấu hiệu mệt mỏi, tất cả con trai văn phòng lại rủ nhau thi chống đẩy. Chỉ tưởng tượng vài anh con trai cởi trần hùng hục hít đất, các anh còn lại hú hét ầm ĩ xung quanh, mồ hôi mồ kê ướt rượt, lại còn trong một chiếc phòng nhỏ nhỏ xinh xinh, tôi lại tự hỏi chẳng lẽ "hành xác" có thể đem lại một thứ gì đó tựa như niềm vui cuối ngày làm việc.

Đến cuối năm 2017, anh lại đứng trước một sự lựa chọn nữa. Thật kỳ lạ khi bạn cứ phải trả lời hết lần này đến lần khác một câu hỏi y hệt nhau: "Giữ hay buông?". Đây là khoảng thời gian khó khăn nhất của Spiderum khi thậm chí không còn đủ vốn để duy trì những hoạt động cơ bản nhất. Dù các thành viên đều đi kiếm cho mình một công việc khác, Spiderum vẫn có thể duy trì hoạt động nếu có sự hỗ trợ từ những thành viên cốt cán. Anh đứng trước lựa chọn vẫn tiếp tục phát triển Spiderum trong khi làm công việc mới, hoặc không. Vẫn như hai lần trước, câu trả lời chẳng có gì mới mẻ. Vẫn như lần trước, tiền với chúng tôi chưa bao giờ là vấn đề, cho đến một ngày đẹp trời nó bỗng dưng là vấn đề lớn.

Anh và tôi quyết định làm đám cưới. Anh và tôi sắp có một đứa con chào đời. Anh sắp làm cha.

Ban ngày anh lo hoàn thành công việc chính, sau giờ làm lại tập trung phát triển và duy trì Spiderum. Spiderum như một đứa con anh không nỡ bỏ, một gánh nặng anh không muốn buông. Đó là khoảng thời gian căng thẳng và mệt mỏi khi anh phải học cách cân bằng giữa công việc và đam mê của mình. Bạn biết đấy, ai cũng chỉ được ban cho một ngày 24 tiếng mà thôi, vì thế ai cũng phải đứng trước lựa chọn có được thứ này sẽ phải hy sinh thứ kia. Khi bạn đầu tư quá nhiều thời gian vào công việc, mọi thứ xung quanh đều trở nên ít quan trọng hơn, đôi khi đó là điều cần thiết nhưng nhiều khi sẽ là sai lầm khó có thể vãn hồi. Đó là bài học đầu tiên anh gặp phải trong cuộc sống hôn nhân: Công việc hay Gia đình. Chúng tôi chọn cách thỏa hiệp và cùng nhau làm mọi thứ nhiều nhất có thể. Điều đó có nghĩa, thỉnh thoảng tôi lại phải làm tester không công cho anh, dùng thử và chủ

yếu là phê bình sản phẩm của anh. Có khi anh còn gạ tôi học HTML chỉ để phụ anh làm những công việc nhỏ nhặt nữa.

Lần đầu tiên anh có cảm giác làm bố thực sự là khi anh chạy bộ 12 tầng bệnh viện chỉ để mau chóng lên gặp vợ và con trai đầu lòng. Ngày hôm đó, anh thức đến 2h sáng làm việc, vừa chợp mắt thì 3h sáng vội vội vàng vàng chở vợ đi viện, chờ đợi đến tận 3h chiều con mới chào đời. "Mệt, nhưng vợ còn mệt hơn" - anh nghĩ vậy. Chẳng thế mà không ngủ vẫn có sức chạy đến 12 tầng lầu. Rồi nhanh chóng, cuộc sống "bỉm sữa" len lỏi trong từng nhịp sống của anh. Cuộc sống của anh từ khi có con trai hóa ra không đáng sợ như anh từng nghĩ. Anh xung phong tắm cho con dù vẫn hơi ngượng tay và lần đầu còn cho con uống nguyên 1 ngụm nước, anh nhận rửa bình dù trước giờ chẳng phải rửa dù chỉ một chiếc bát. Họ hàng ai cũng bất ngờ vì bỗng dưng anh trở thành bố bỉm sữa "xịn" đến vậy. Nhưng một anh dev quen "gõ sớ" cũng gặp phải vài khó khăn nhất định khi chăm con. Nhiều khi Google cũng không cứu cánh nổi cho anh những lúc con "lên cơn" quấy. Trời phú cho con tôi cái tật "thính" - chỉ một tiếng động nhỏ cũng có thể khiến nó thức giấc, hoặc không thể ngủ được. Những lúc ấy, bản năng của một coder trỗi dậy. Anh test đủ bảy bảy bốn chín tư thế rung lắc ru con ngủ: từ ngoáy cháo đại pháp, sàng nia tam chiêu, xóc ốc liên hoàn chưởng đến đu thuyền chân kinh. Anh thậm chí còn thử bế con đi lên đi xuống cầu thang hòng "chuốc" con vào cơn buồn ngủ, rốt cục chỉ có hai cái chân giò của anh là "say mèm".

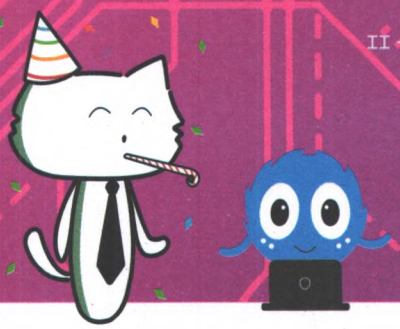
Ở thời điểm đó anh đã có một công việc remote khá ổn, làm tại nhà, lương tốt, sếp thân thiện, không phải quan tâm nhiều đến những chuyện râu ria công sở. Công việc dần đi vào ổn định, thu nhập khá lên rất nhiều, duy chỉ có nếp sống của tôi và anh chẳng mấy thay đổi. Chúng tôi vẫn đi ăn cháo Bách Khoa lúc rảnh, tuần nào cũng rủ nhau đi ăn nướng lụi khói mù ám đến từng chân tóc. Cuộc sống không mấy phải lo lắng về tiền bạc mới quay trở lại không bao lâu thì đùng một cái anh nhận tin có đứa thứ hai. Câu đầu tiên mà anh nói là "Que hỏng rồi, làm gì có chuyện đấy!". Hóa ra cái gì hỏng chứ que thì không hỏng. Lại một lần nữa anh phải suy nghĩ về tiền. Có lẽ khát khao cho con một cuộc sống như ý đã khiến anh lo lắng những điều trước kia mình chưa từng mảy may nghĩ tới.

Anh đặt tên đứa đầu là Sudo[1]. Không phải Cam, Táo, Mít, Quýt, Dừa,... Không phải Su Kem, Su Su mà là Sudo. Anh bảo Sudo là một lệnh trong "ngành gõ sớ" của anh. Mặc kệ cho tôi cười, anh dứt khoát chốt cái tên Sudo như thể gói ghém đặt cả vào đó bao mơ ước và kỳ vọng con trai sẽ học code. Tôi không biết liệu có phải developer nào cũng muốn con mình theo nghiệp của bố không, nhưng anh dev của tôi như muốn thổi bùng lên ngọn lửa đam mê code cho những đứa con một từ bé đôi còn chưa nói được. Mà có lẽ cha mẹ vốn vậy, thời đi học cứ trách sao bố mẹ ép con học nhiều quá, đến khi làm bố mẹ rồi lại muốn con sống tiếp ước mơ của mình. Anh nói đấy là điều tốt nhất anh có thể định hướng cho con, công nghệ phát triển từng ngày, mỗi lúc lại có một nghề mất đi vì sự phát triển đó, nhưng anh biết nghề của anh sẽ là nghề mất đi cuối cùng.

Tôi có biết nhiều người bạn và người anh làm kỹ sư phần mềm. Có rất nhiều người trong số đó đã sớm chuyển mình sang một nghề khác năng động hơn vì không thể chịu được việc ngồi mười mấy tiếng trước cái màn hình đơn sắc. 5 năm qua tôi đã quen với hình ảnh người bạn đồng hành của mình, ngày qua ngày, dù ngồi code cả ngày, vẫn tràn ngập trong tim niềm hứng khởi. Tôi nghĩ dù sao đi nữa, đam mê vẫn là chất xúc tác cho mọi công việc, là chất đốt cho mọi hoài bão.

Spiderum của anh hiện tại đã bắt đầu "khôn lớn". Cùng với sự giúp đỡ của nhiều thành viên khác nữa, Spiderum bước qua giai đoạn khó khăn, không còn ai phải ăn mì tôm với bim bim, số lượng người dùng đăng kí đã lên đến 50 nghìn người, hàng tháng có tới 1,5 triệu pageviews và đem lại thật nhiều giá trị cho cộng đồng. Có được những khởi sắc như vậy, có lẽ anh cũng như team, ai cũng mừng và có nhiều động lực để tiếp tục cố gắng. Anh và team cũng có cơ hội được phát triển thêm những tính năng mới mà mọi người từng ấp ủ.

Đôi khi tôi tự hỏi nếu anh học Đại học Thương mại thay vì FPT, hoặc nếu trường Thương mại có đủ điều hòa để năm đó nhập học anh không "chạy mất dép" vì... nóng, thì liệu giờ anh có là anh dev của tôi không. Tôi nghĩ có thể nghề đã chọn anh trước khi anh chọn nghề. Cũng có thể mỗi khi cuộc đời hỏi anh "Đi hay ở", anh chỉ giản đơn trả lời lại duy nhất một đáp án đã cũ: "Anh chọn nghề."



Đến với những câu chuyện đa chiều của các tác giả trong phần 2, có lẽ bạn đã mường tượng phần nào về các đặc thù công việc của từng vị trí khác nhau trong lĩnh vực IT. Mỗi vị trí đều có thuận lợi và khó khăn, có niềm vui và nỗi buồn rất riêng. Hành trình này tuy thách thức nhưng cũng đầy thú vị. Bạn hãy tự tin bước qua để trưởng thành!

Góc bật mí:

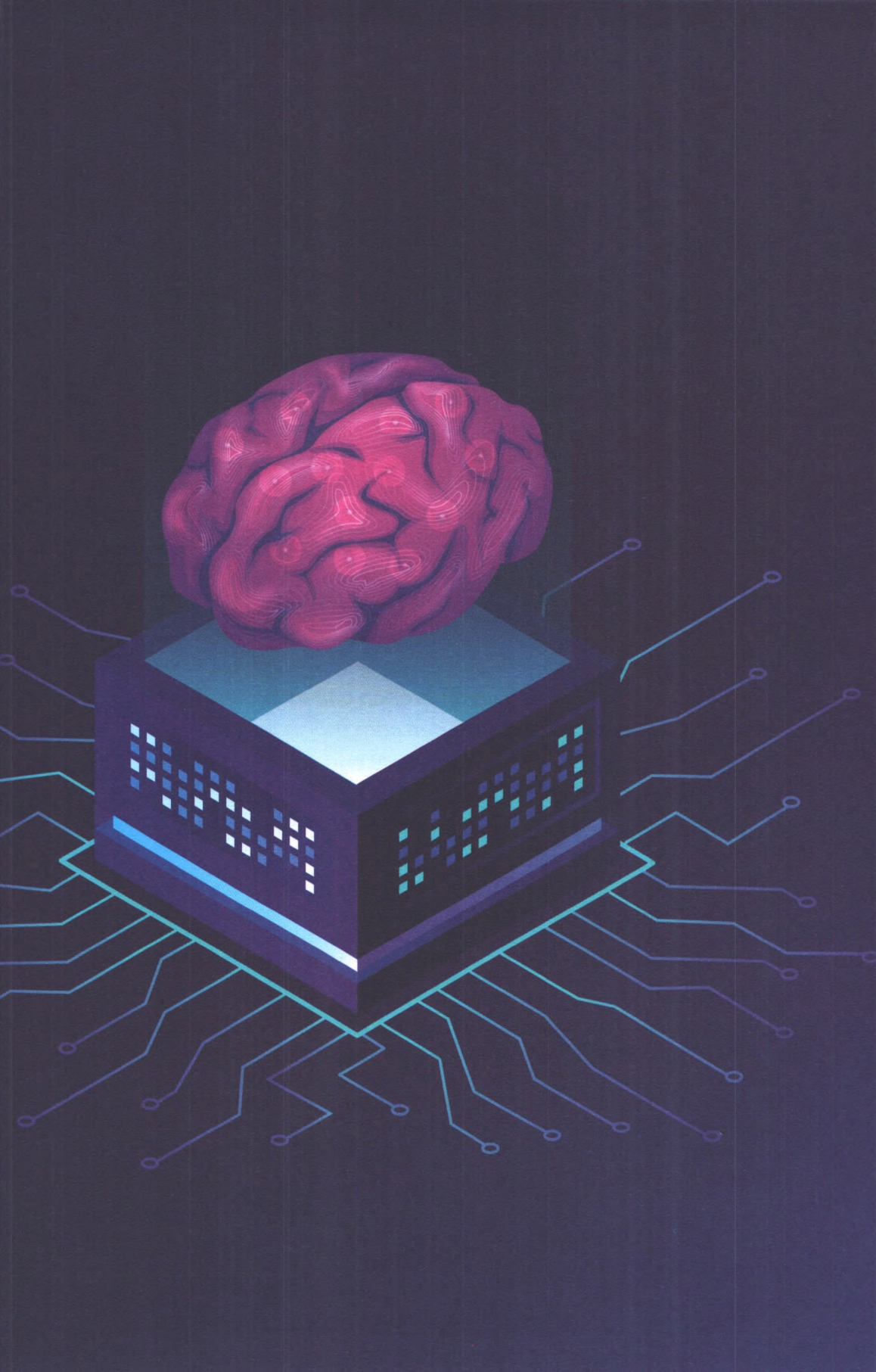
Để vững vàng trong chặng đường tiếp theo, bạn cần chuẩn bị cho mình một chiếc CV gây ấn tượng "cực mạnh" với nhà tuyển dụng.

Cùng tìm hiểu xem những cây viết trên Spiderum đã làm giàu vốn trải nghiệm của bản thân để "tích lũy" vào CV như thế nào và ghé thăm ứng dụng tạo CV chuyên nghiệp của TopCV dưới đây nhé:

Quét để tải ứng dụng

3

HÀNH TRANG VÀO NGÀNH



Để trở thành lập trình viên, bạn không cần phải là một "SIÊU SAO"

Tác giả: **Phạm Bình**
Full-stack Developer

Con đường để trở thành một lập trình viên tuy không thẳng tắp và bằng phẳng nhưng cũng không đến nỗi quá quanh co và gồ ghề như nhiều người vẫn nghĩ. Mình tin dù bạn là ai, có xuất phát điểm thế nào, chỉ cần đam mê và kiên trì, chắc chắn ước muốn trở thành lập trình viên sẽ thành hiện thực. Nếu bạn cảm thấy nghi ngờ về luận điểm trên, hãy đọc hết bài viết này. Đây là câu chuyện của chính mình - một cậu bé vô cùng bình thường trên con đường chinh phục ước mơ trở thành một lập trình viên chuyên nghiệp.

Trước tiên, mình xin giới thiệu một chút về bản thân để các bạn thấy rằng mình cũng có xuất phát điểm bình thường như nhiều đứa trẻ khác.

Mình tên đầy đủ là Phạm Quang Bình, sinh ra và lớn lên ở ngoại thành thành phố Uông Bí - Quảng Ninh, nơi mọi người phần lớn làm nông hoặc công nhân nhà máy. Tuổi thơ của mình gắn chặt với hình ảnh con trâu, cánh đồng lúa, với mấy trò chơi của tụi trẻ con vùng quê như thả diều, câu cá, bắn bi,...

12 năm ăn học của mình cũng không có thành tích gì nổi trội, không đứng cuối lớp nhưng cũng chẳng đứng đầu lớp, không cá biệt nhưng cũng chẳng ngoan ngoãn gì, cứ bình bình ở giữa. Mình chẳng thật sự giỏi một môn học nào, điểm phẩy các môn dao động trong khoảng 6,5 - 8,0.

Về gia đình, bố mình làm công nhân nhà máy, mẹ mình làm công việc tự do, cả gia đình không ai làm về IT, cũng không ai hướng cho mình theo nghề IT. Ngay cả bản thân mình lúc bé cũng không nghĩ sẽ theo nghề IT.

Thế nhưng, hiện tại (2020) mình đang là Full-stack Developer cho một công ty khá nổi tiếng ở Hà Nội, với mức thu nhập kha khá đủ để mình sống và tiếp tục theo đuổi đam mê.

Bạn thấy đấy, xuất phát điểm của mình không có gì đặc biệt cả, có may chăng mình là một thằng rất lì, không chịu bỏ cuộc cho tới khi đạt được và thỏa mãn điều gì đó.

Dưới đây là câu chuyện của mình kể từ lúc hình thành đam mê cho tới khi chinh phục ước mơ trở thành lập trình viên.

Trí tò mò dần dần hình thành đam mê

Trước khi hình thành đam mê với IT nói chung và lập trình nói riêng, mình là một người rất tò mò về các khái niệm công nghệ.

Mình nhớ khái niệm đầu tiên khiến mình tò mò là "virus máy tính". Nghe tới virus, mình liên tưởng tới thứ gì đó động đậy và sống ngoài môi trường tự nhiên, chứ sao lại sống trong máy tính (đọc đến đây xin các bạn đừng cười bởi lúc đó mình mới là học sinh tiểu học). Mình có thắc mắc với bố, nhưng nhận được câu trả lời là: "Ừ, đương nhiên, virus cũng có thể sống trong máy tính chứ". Không hiểu bố suy nghĩ như vậy thật, hay chỉ là câu trả lời để thằng con thỏa mãn rồi chịu nghe lời hơn. Nhưng dù gì đi nữa, mình vẫn không tin lắm.

Rồi hàng loạt những thắc mắc khác, như: Làm sao cái đĩa CD sặc sỡ bảy sắc cầu vồng kia lại lưu được hình ảnh? Làm sao một cái loa có thể phát ra tiếng với cục nam châm phía sau?,... Rất nhiều, rất nhiều các câu hỏi khác nữa mà đến hàng năm sau đó mình vẫn chưa có câu trả lời. Cho tới một ngày...

Cho tới một ngày, mình bắt đầu biết tới việc "lên mạng", hay chính là Internet. Mình được một thằng bạn giới thiệu cho trang google.com, nó bảo lên đấy tìm cái gì cũng có (đợt ấy mình cần tìm văn mẫu chép cho sướng, đỡ phải nghĩ). Bây giờ nghĩ lại, thật thầm cảm ơn nó, các kiến thức trên Internet với sự giúp sức của google đã "khai sáng" cho mình nhiều thắc mắc trước kia.

Sau khi tìm hiểu được nhiều kiến thức mới từ Internet, mình lại mong muốn có thể chia sẻ kiến thức lên đó. Thế là mình tiếp tục tìm hiểu về "cách tạo trang web cá nhân" để có thể viết bài chia sẻ tới mọi người. Hồi đó mình chưa biết đến Facebook (hoặc có khi Facebook chưa ra đời), các nền tảng blog như WordPress, Blogger, cũng chưa phát triển. Thế nên mình chơi lớn, tự tìm cách lập trình ra trang web riêng để viết bài chia sẻ cho tiện.

Mình bắt đầu dành hàng giờ đồng hồ trước màn hình máy tính (may mắn là lúc bố mình vừa sắm cái laptop), đốt không biết bao nhiêu tiền *D-com 3G* của bố để lên mạng tìm tài liệu học lập trình web. Chưa bao giờ mình cảm thấy thích thú với việc học đến thế, chưa bao giờ ngồi vào bàn học mà cảm tưởng thời gian trôi nhanh đến vậy. Càng tìm hiểu, mình càng thấy có nhiều kiến thức mới, lại càng ham, càng thôi thúc tìm hiểu nhiều hơn. Dần dần, mục tiêu "có thể viết bài chia sẻ tới mọi người" trở thành "có thể lập trình ra một trang web ổn định" từ lúc nào không hay.

Kể từ đó, đam mê lập trình, cụ thể là lập trình web, được hình thành.

Nghĩ lại, thấy đam mê đến với mình như một cái duyên. Nếu thằng bạn không giới thiệu cho trang google, nếu bố không sắm cái laptop đúng lúc, có khi bây giờ mình chẳng là một lập trình viên.

Bạn cũng vậy, nếu đang đọc bài viết này của mình, ít nhiều bạn cũng có duyên với nghề IT đó, hãy thử tìm hiểu xem sao.

Những bước chân đầu tiên chạm vào công nghệ

Sau một thời gian mày mò, mình có sản phẩm đầu tiên là một blog cá nhân, mục tiêu đã hoàn thành, cảm giác viên mãn vô cùng. Mình vui vẻ đem khoe hết người này tới người khác, khoe từ gia đình tới bạn bè,... Mấy thằng bạn mình tỏ vẻ thờ ơ, nhưng mình biết trong lòng chúng nó phục mình lắm, còn bố mẹ chắc cảm thấy tự hào, nhất là bố mình kiểu gì cũng cảm thấy "Tiền D-com không phí chút nào".

Không lâu sau, mọi người quen dần với việc mình có thể viết được mấy ký tự khó hiểu (code) trên máy tính, chẳng còn gì bất ngờ nữa. Mình cũng suy nghĩ nghiêm túc hơn về nghề nghiệp trong tương lai. Đây là thời điểm mình học cấp ba (lớp 10) - giai đoạn mà bất kỳ học sinh nào cũng phải suy nghĩ nghiêm túc hơn cho bản thân. Mình thoáng nghĩ trong đầu: "Hay trở thành lập trình viên nhỉ?".

Suy nghĩ đó dần xuất hiện nhiều hơn và nghiêm túc hơn. Nhìn lại bản thân, mình đã có lượng kiến thức hòm hòm nhờ tự tìm hiểu từ trước, mình cũng khá nhạy bén với công nghệ do game gì cũng biết chơi, lại được thêm tính lì lì, thích một mình. Tổng kết lại, trở thành lập trình viên có vẻ là lựa chọn sáng giá nhất lúc này. Dù vậy, mình vẫn chưa xác định được ước mơ. Nhưng không vì thế mà ngừng học thêm kiến thức lập trình, mình vẫn tìm hiểu, chỉ là với lượng thời gian ít hơn.

Thời gian cứ trôi, lớp 10, lớp 11 qua đi, đến năm cuối cấp ba, mình phải đưa ra lựa chọn để biết mình sẽ là ai trong tương lai cũng như chọn khối thi, chọn trường Đại học. Nghĩ lại, đã có lúc mình muốn trở thành lập trình viên: "Ừ thế thì trở thành lập trình viên đi!". Mình chốt hạ phương án cuối cùng trong phút chốc mà chẳng do dự, bởi mình hiểu rằng cuộc sống đôi khi

"đâm lao thì phải theo lao", thà cứ chọn đại mục tiêu rồi theo đuổi đến cùng, còn hơn chẳng có mục tiêu gì.

Kể từ đó, ước mơ trở thành lập trình viên được hình thành. Mình chia sẻ với bố mẹ về mục tiêu chọn ngành nghề, chọn khối thi và chọn trường thi, tất nhiên bố mẹ ủng hộ. Mẹ đã nghĩ tới việc sẽ đưa mình đi thi đại học thế nào (kiểu lo xa), còn bố sắm cho mình một chiếc laptop riêng thay vì phải dùng chung với bố.

Từ khi có máy tính riêng, động lực của mình tăng gấp đôi.

Lại miệt mài với việc tự học lập trình trên mạng, mình tò mò không biết công việc thật sự của một lập trình viên là thế nào? Cụ thể công việc của một web developer ra sao? Và mình còn thiếu những kiến thức gì để trở thành một web developer thực thụ? Mình tham khảo ý kiến trên nhiều diễn đàn, đọc bình luận, chia sẻ của các anh/chị đi trước, họ khuyên rằng: "Cách tốt nhất để học lập trình là học thông qua các dự án thực tế". Ok, vậy kiếm dự án thực tế về xem thôi.

Tiếp tục lang thang trên mạng, mình tìm được bộ mã nguồn mở của một team bên Nga, đây là mã nguồn dùng để xây dựng một diễn đàn mini, viết bằng PHP - MySQL. Mình download về, cài đặt, mở source code (mã nguồn) lên đọc, bất kỳ đoạn nào không hiểu mình đều lên mạng tra cứu và ghi chú lại cẩn thận. Không lâu sau, mình tích lũy được một lượng kiến thức đáng kể, có khi bằng 5, bằng 10 lần lượng kiến thức mình có trước kia, quả thực là mình còn rất nhiều thứ phải học.

Mình tham khảo thêm nhiều mã nguồn mở khác, không chỉ tham khảo các câu lệnh mới, mà cả thuật toán của họ hay cách họ tổ chức cấu trúc dự án,... Cái blog cá nhân của mình được đập đi làm lại đến cả chục lần, mỗi lần làm lại là một lần mình thực hành áp dụng những kiến thức mới vừa học được.

Nhoằng cái đã đến kỳ thi đại học, mình chọn khối A - Toán, Lý, Hóa. Nhưng thời gian vừa rồi mình tập trung học lập trình nhiều hơn học kiến thức trên lớp, kết quả là Toán biết sương sương, Hóa tàm tạm, còn Vật lý... gần như mù tịt. Làm thử đề thi Đại học những năm trước chưa bao giờ vượt quá 20 điểm, trong khi các trường Đại học về IT mình biết ít nhất cũng lấy 21 điểm. May mắn, mình quen một anh trên mạng và được giới thiệu Trường Đại học Công nghệ thông tin và Truyền thông Thái Nguyên, nghe qua tên trường là biết ngay đào tạo về IT - đúng ngành mình thích, xem qua mức điểm sàn thấy khá "dễ chịu". Vậy là mình nộp hồ sơ và thi đỗ nguyện vọng một vào đó.

Kể từ đây, một trang mới trên hành trình chinh phục mơ ước trở thành lập trình viên của mình được mở ra. Mình quen biết nhiều anh em bạn bè có chung đam mê, sở thích, mọi người cùng chia sẻ kiến thức lẫn nhau và nhờ đó, mình học thêm được nhiều điều mới. Lúc này mình mới thấy quyết định trở thành lập trình viên trước kia thật sáng suốt. Không! Phải là "cực kỳ sáng suốt" mới đúng. Nhưng mình cũng hiểu đây mới chỉ là bắt đầu, còn rất nhiều thử thách phía trước phải vượt qua trước khi chạm được tới đích cuối cùng: "Lập trình viên".

Tích lũy kiến thức và vượt qua "sức ì" của bản thân

Thời gian đầu ở môi trường mới, bên cạnh kiến thức trên giảng đường, mình vẫn tự học như cách làm trước kia. Nhưng một thời gian sau, mình bắt đầu chững lại do kiến thức trên Internet tuy nhiều nhưng đa phần ở mức cơ bản. "Để học kiến thức nâng cao hơn, chắc chỉ có đường xin thực tập ở một công ty phần mềm nào đó" - mình nghĩ vậy. Nhưng vừa mới chập chững lên Đại học, sợ chểnh mảng ảnh hưởng kết quả học tập, mình tạm thời hoãn ý định này. Thay vào đó, mình xin tham gia Câu lạc bộ Tin học ở trường.

Trường mình chuyên về IT, nên CLB Tin học hội tụ không ít "anh tài" với thành tích học tập vô cùng đáng nể, thậm chí có người còn lập cả công ty riêng khi mới là sinh viên năm 3. CLB chia làm 3 ban hoạt động: Ban giải thuật; Ban lập trình web; Ban lập trình ứng dụng; hoạt động xen kẽ các buổi tối trong tuần. Mình hoạt động chủ yếu ở ban lập trình web.

Hoạt động ở CLB đem đến cho mình nhiều kiến thức mới. Bởi cùng một bài toán nhưng lại có nhiều lời giải khác nhau, mọi người cùng góp ý, phân tích, tìm ra ưu nhược điểm của mỗi cách để đưa ra lựa chọn tốt nhất. Cảm giác được lắng nghe, trò chuyện với những người cùng đam mê thật tuyệt vời, khác hoàn toàn lúc lủi thủi tự học. Các hoạt động nhóm diễn ra thường xuyên giúp mình có thêm các mối quan hệ mới và học được cách phối hợp với người khác để hoàn thành nhiệm vụ. Sau một thời gian, mọi người khuyên mình nên đi thực tập ở các công ty từ sớm để tích lũy kinh nghiệm, sau này đi làm đỡ bỡ ngỡ. Rõ là mình từng có ý định như vậy nhưng sợ ảnh hưởng tới việc học nên lại thôi. Nhưng mọi người đã khuyên thế, mình cũng nên thử xem sao, hơn nữa mình cũng "cứng" hơn xưa nhờ hoạt động tích cực ở CLB rồi.

Mình quen một anh khóa trên, anh vừa startup một công ty phần mềm và ngỏ ý muốn mình về làm cùng (chẳng hiểu sao mình may mắn đến vậy, cứ lúc cần lại có người giúp đỡ), mình đồng ý ngay. Vậy là từ giờ, ngoài việc học trên giảng đường mình còn phải cân bằng cả thời gian đi làm nữa.

Đó là startup chuyên thiết kế website cho các doanh nghiệp. Công việc của mình là triển khai các dự án web theo yêu cầu của khách hàng - đương nhiên là yêu cầu cũng đơn giản thôi.

Từ ngày đi làm thêm, mình mới thật sự hiểu được thế nào là "không có thời gian": Buổi sáng đi học từ 6h30 - 11h45, buổi chiều từ 13h30 - 17h lên công ty làm việc, hôm nào học chiều thì đổi ngược lại, buổi tối tham gia hoạt động trên CLB, nếu không lại tiếp tục thói quen tự học. Nói chung, một ngày của mình kín mít từ 6h30 sáng cho tới 22h đêm, may chăng có chủ nhật được rảnh. Thời gian đầu thấy stress lắm, nhưng cứ nghĩ tới ước mơ, đam mê, tương lai, mình lại có thêm động lực để tiếp tục.

Thế rồi dần cũng quen, khi mọi thứ đã vào guồng mình lại cảm thấy bình thường, còn thấy đây là lợi thế lớn. Mình từng nghĩ việc làm thêm sẽ khiến bản thân mất tập trung học, nhưng không, nó khiến mình học dễ vào hơn bởi: Thứ nhất, đi làm cho mình kinh nghiệm, đi học cho mình kiến thức, và chúng bổ trợ cho nhau. Nhiều khi nghe thầy cô giảng mà mình cảm giác như đã học được điều này ở đâu đó trước kia rồi. Thứ hai, do có ít thời gian, nên mình phải tận dụng nó một cách hiệu quả, lên giảng đường là phải học nghiêm túc. Thứ ba, nó cho mình thấy so với đi làm thì đi học dễ hơn nhiều, bởi đi làm yêu cầu bài toán không rõ ràng như trong sách vở, và cũng không có đáp án để so sánh. Khi đi học, hỏi càng nhiều thầy cô giáo càng vui, nhưng khi đi làm, chỉ nên hỏi khi thật sự cần hỏi, bởi ai cũng có công việc của riêng họ.

Nhưng suy nghĩ kỹ, giờ chưa phải lúc để nghỉ, mình không cho phép bản thân được nhàn rỗi trong khi ước mơ chưa đạt được, đây là cơ hội tốt để thử thách bản thân

Mọi thứ cứ tiếp tục trôi qua như thế cho tới năm cuối Đại học. Startup mình làm việc vì nhiều lý do mà phải tạm dừng, mình không còn là cậu sinh viên năm nhất ngây thơ nữa, cũng không còn nhiều thời gian để chuẩn bị ước mơ. Mình cần phải làm điều gì đấy đột phá hơn.

Năm cuối cũng là khoảng thời gian mình có đợt thực tập tại doanh nghiệp. Mình phân vân giữa 2 quyết định: Một là chọn công ty gần trường với công việc quen thuộc để dễ dàng vượt qua đợt thực tập. Hai là tìm đến một công ty mới, xa lạ hoàn toàn, công việc thử thách thật sự.

Thời gian đầu, mình nghiêng hoàn toàn về lựa chọn thứ nhất, cho rằng đây là khoảng thời gian vừa làm vừa nghỉ ngơi. Nhưng suy nghĩ kỹ, giờ chưa phải lúc để nghỉ, mình không cho phép bản thân được nhàn rỗi trong khi ước mơ chưa đạt được, đây là cơ hội tốt để thử thách bản thân, mình đã chuẩn bị suốt bao lâu nay, bây giờ "đến thời" thì không được bỏ qua.

Vậy là mình rời trường, rời Thái Nguyên lên Hà Nội để tìm công ty phù hợp (trước đó mình có nộp CV vào 3 công ty ở Hà Nội và đều được hẹn phỏng vấn).

Đối diện những khó khăn và hiện thực hóa ước mơ

Nói thật, đặt chân đến đất Thủ đô, mình cảm giác như đi nhập học Đại học lần thứ hai vậy, chẳng quen ai, cũng chẳng biết đường đi lối lại. May là còn có "chị google" giúp mình tìm đường.

Phỏng vấn suôn sẻ, mình đỗ cả 3 công ty, và chọn một startup để làm việc ở vị trí PHP Developer. Lúc này mình rất vui sướng, còn sướng hơn cả cảm giác tạo được trang web đầu tiên, vì mình biết ước mơ của mình sắp thành hiện thực.

Chọn được công ty với vị trí công việc mong muốn, mình rời hẳn Thái Nguyên và chuyển đến sống ở Hà Nội.

Môi trường mới khiến mình gặp đôi chút khó khăn. Mình phải vất vả cả ngày mới tìm được phòng trọ ưng ý với giá thuê... gấp 4 lần giá thời sinh viên, trong khi các tiện ích chỉ tương tự. "Nhưng thôi, đây là Hà Nội mà, tuy đắt đỏ và chật chội, nhưng bù lại được cái tiện lợi, và cơ hội phát triển bản thân cũng rộng mở" - mình tự an ủi bản thân như vậy.

Thời gian đầu mình cảm thấy mệt mỏi, do cuộc sống khác thời sinh viên quá. Cuộc sống ở đây vội vã hơn, tắc đường nhiều hơn, khói bụi nhiều hơn, ồn ào hơn, đắt đỏ hơn. Mình bắt đầu có suy nghĩ chỉ thực tập ở Hà Nội 2 tháng rồi lại về Thái Nguyên làm việc cho "dễ thở". Nhưng nhớ lại những ngày đầu lên Đại học, môi trường cũng thay đổi, cũng khiến mình mệt mỏi, mà rồi mọi thứ cũng đâu vào đấy. Thế nên thay vì tìm cách trốn tránh, mình quyết định tìm cách thích nghi. Mình lập ra danh sách các địa điểm dịch vụ gần nhất như: quán ăn gần nhất, chợ gần nhất, tiệm thuốc gần nhất, bến xe gần nhất hay thậm chí đi thử vài chuyến xe bus xem thế nào. Mình cũng chỉ dám mua sắm những thứ cơ bản để có thể duy trì cuộc sống ở môi trường mới này.

Chưa quen với cuộc sống mới là một chuyện, trên công ty mình cũng chưa quen với công việc mới luôn. Công việc này đòi hỏi độ tỉ mỉ cao hơn, và cần một lượng kiến thức mới kha khá cả về mặt quy trình làm việc lẫn chuyên môn. Phòng mình có 6 người, 2 anh trưởng phòng dày dạn kinh nghiệm, 3 anh còn lại đều tốt nghiệp khoa CNTT của các trường có tiếng tại Hà Nội. Tất cả mọi người đều ngang ngang tuổi nhau, thuộc thế hệ 9x nên rất dễ nói chuyện. Mình mới vào, là người trẻ nhất nên được xem như em út và được mọi người nhường nhịn, chỉ bảo nhiệt tình (ngay cả khi có những câu hỏi "hơi ngu"). Mình rất hay hỏi những câu... có đáp án ngay trước mắt. Ví dụ, chạy chương trình mà thấy xuất hiện lỗi "Undefined variable: x on line 15", mình sẽ không nghĩ ngợi gì mà hỏi luôn: "Các anh ơi đây là lỗi gì ạ?", trong khi chỉ cần chú ý cái thông báo lỗi kia là hiểu ngay: "Biến x chưa được định nghĩa trên dòng số 15". Ban đầu mình cảm thấy tự ti lắm, rất thường xuyên phải hỏi mọi người, mà hỏi nhiều lại sợ phiền. Nhưng anh em trong phòng chắc cũng hiểu điều

này nên bảo mình: "Cứ hỏi đi đừng ngại". Nhờ thế, sau chừng 1 tháng, kết hợp giữa việc tự tìm hiểu và sự giúp đỡ nhiệt tình của anh em trong phòng, mình đã quen với công việc hiện tại.

Khi quen dần với môi trường làm việc, mình nhận ra nó cũng không đến nỗi khó, mà quan trọng nằm ở thái độ của bản thân đối với nó như thế nào. Cơ bản công việc vẫn là lập trình, vẫn là những kiến thức cơ bản giống hồi đầu mình tự học, có điều nó được triển khai theo quy trình phức tạp hơn, đòi hỏi các bước làm phải rõ ràng hơn.

Khi phải thay đổi thói quen hoặc tiếp thu những kiến thức mới, mình cần tích cực đón nhận, xem nó như một niềm vui thay vì cảm thấy mệt mỏi, khó chịu.

Một điều quan trọng nữa là biết cách phối hợp với mọi người trong nhóm để hoàn thành công việc. Dù có tài giỏi đến mấy, bạn cũng không thể tự mình làm hết mọi việc, và kể cả có làm được hết đi nữa thì cũng chẳng ai giao hết cho bạn cả vì thời gian thường có hạn. Vì thế, việc tương tác, trao đổi, phối hợp với mọi người trong nhóm sao cho hiệu quả cũng quan trọng không kém so với kiến thức được sử dụng trong lúc làm việc. Mình từng nghe thấy đâu đó họ nói rằng "Lập trình viên đâu phải chỉ biết code".

Có lần mình và một anh nữa đảm nhiệm dự án mới của công ty. Dự án này yêu cầu khách truy cập phải đăng ký/đăng nhập tài khoản mới có thể sử dụng được các tính năng khác. Mình đảm nhiệm tính năng đăng ký, còn anh kia đảm nhiệm tính năng đăng nhập. Do cả hai đều chủ quan, cho rằng tính năng này dễ, mặt khác mình với anh kia đều mới vào công ty nên cũng ngại trao đổi với nhau. Kết quả là thông tin ghi nhận từ tính năng đăng ký mình làm lại không khớp với thông tin khi đăng nhập. Với tính năng đăng ký, mình yêu cầu thông tin mật khẩu tối thiểu là 6 ký tự, còn khi đăng nhập lại yêu cầu tối thiểu là 8 ký tự. Dẫn đến một số tài khoản đăng ký xong nhưng không đăng nhập được. Chuyện đáng nói nhất ở chỗ, lỗi này được phát hiện trong lúc bọn mình đang demo sản phẩm cho sếp xem, và nó còn là tính năng đầu tiên nữa. Cảm giác sai ngay ở bước đầu khiến mình chỉ mong có cái lỗ chui xuống cho đỡ ngại. Nếu team chịu trao đổi với nhau hơn, lỗi này chưa chắc đã xảy ra. Cũng may là sếp chỉ nói đùa một câu: "Đến ông lớn như Apple khi demo còn sai, nên các em thế này cũng bình thường thôi". Đúng là bài học đáng nhớ.

Trong công việc, khó khăn mình gặp chủ yếu là về mặt công nghệ, như phải tìm hiểu một công nghệ mới trong một khoảng thời gian có hạn, hay điều tra một lỗi rất "ẩn" trong hệ thống hiện tại.

Về khó khăn trong công nghệ mới, mình nhớ nhất lần được giao tìm hiểu về big data (dữ liệu lớn) xem nó là gì, liệu có phù hợp để áp dụng cho dự án sắp tới không? Lúc ấy, khái niệm này như một mảnh đất khác mà mình chẳng có bất cứ kiến thức gì về nó cả. Trên mạng, tài liệu tiếng Việt về big data rất ít, đa phần đều hời hợt. Vậy là mình chuyển sang đọc tài liệu tiếng Anh, nhưng cũng không khá hơn là mấy, khi chúng chỉ nói nhiều lý thuyết (chưa kể tiếng Anh mình cũng chỉ đọc bập bõm). Suốt 1

tuần đầu tiên, mình chẳng thu được kết quả, cảm giác bế tắc, không biết phải làm gì tiếp theo bởi tất cả những gì nghĩ ra đều đã làm hết rồi. Nhưng may lúc đấy phòng mình vừa có một bạn mới, bạn này có chút kinh nghiệm làm việc với big data trước kia. Nhờ bạn giải thích, mình nhận ra được nguyên nhân sâu xa khiến mình hiểu sai vấn đề: Big data là kỹ năng của Data Engineer (kỹ sư dữ liệu), trong khi mình đang nhìn big data dưới góc độ của một Software Engineer (kỹ sư phần mềm). Vì đặt sai góc nhìn mà một số vấn đề mình hiểu sai trầm trọng. Từ việc hiểu sai, dẫn tới hướng tìm hiểu sai, và cuối cùng là không thu lại được gì. Sau khi nhận ra, thử đặt bản thân dưới góc độ một Data Engineer và tìm hiểu, cùng với sự trợ giúp của đồng nghiệp, trong 2 tuần tiếp theo mình đã có bài báo cáo về big data suôn sẻ.

Sau lần đó, bài học sâu sắc mình rút ra là: Khi tìm hiểu một công nghệ mới, phải chuẩn bị sẵn tâm lý của một người mới, đừng mang những kiến thức bản thân hiện có rồi áp đặt lên chúng. Giống như truyện kiếm hiệp, để học võ công mới, bạn thậm chí còn phải phế bỏ võ công hiện tại vậy.

Một trục trặc nữa cũng khiến mình nhớ đó là sản phẩm bên mình có tích hợp một cổng thanh toán bằng ví điện tử của bên thứ ba. Một lần, họ thông báo rằng giao dịch giữa hệ thống bên mình và bên họ không giống nhau. Bên họ ghi nhận 11 giao dịch, còn bên mình chỉ ghi nhận 10 giao dịch. Mình điều tra, đọc lại mã nguồn của đoạn ghi lại thông tin giao dịch bên mình thì không có gì đáng ngờ cả, nhờ họ kiểm tra lại hệ thống bên đó, họ cũng nói rằng hệ thống bên họ không có gì đáng ngờ. Thế là mình kiểm tra lại một lần nữa và phát hiện một trường hợp khá đặc biệt, đó là khi có một người thực hiện 2 giao dịch quá nhanh (gần như cùng một thời điểm), bên mình chỉ ghi nhận là 1 giao dịch, còn bên họ sẽ ghi nhận là 2 giao dịch. Điều này tạo ra sự ghi nhận chênh lệch giữa 2 hệ thống. Sau đó cả 2 bên cùng hợp tác để giải quyết lỗi này. Đúng là cần phải xem xét thật kỹ vấn đề trước khi kết luận.

Tính tới thời điểm đang viết bài này, mình vẫn đang gắn bó với công ty và đã làm việc được gần 2 năm. Quãng thời gian qua, tuy chưa có thành tích xuất sắc nào, nhưng cũng cảm thấy bản thân mình có vị trí trong nhóm, được đồng nghiệp và cấp trên tin tưởng. Các thuận lợi và khó khăn vẫn có, vẫn diễn ra, nhưng mình không bận tâm lắm bởi mình luôn chuẩn bị tâm lý để giải quyết chúng, luôn đón nhận chúng dưới góc độ tích cực, coi chúng như một thử thách mà bản thân phải vượt qua.

Trong tương lai, mình vẫn tiếp tục theo đuổi công việc hiện tại. Bởi từ bao giờ nó không còn đơn giản là ước mơ nữa, mà đã thành thói quen và trở thành một phần không thể thiếu trong con người mình.

Sau cùng, có vài lời mình muốn khẳng định với bạn đọc rằng: Để trở thành lập trình viên, bạn không nhất thiết phải là "siêu sao". Giống như con đường mình đã trải qua vậy, không có gì đặc biệt hết, chỉ cần sự nỗ lực, kiên trì, chấp nhận thay đổi đồng thời có thái độ tích cực khi gặp thách thức, mình tin bạn có thể đạt được ước mơ. Một ngày nào đó trong tương lai, biết đâu chúng ta trở thành đồng nghiệp?

Phát triển phần mềm: Những điều tôi học được

Tác giả: **Curly Rae Braces**
Kỹ sư phần mềm

Vì bạn đang cầm cuốn sách này trên tay, tôi nghĩ bạn là người quan tâm đến ngành công nghệ thông tin. Có thể bạn đang (hoặc có ý định) viết phần mềm, hay ít nhất là viết code, làm web, làm tự động hóa cho một công ty/dự án nào đó. Nếu như mong muốn của bạn là trở thành người viết phần mềm chuyên nghiệp, đây là bài viết dành cho bạn.

Khép lại thập kỷ 2010, thế giới đã thay đổi đáng kể, từ chỗ sử dụng máy tính được coi là một kỹ năng cộng thêm, bây giờ nó trở thành kỹ năng phải có để làm bất cứ việc gì. Chuyện lập trình được máy tính, hiểu và viết code để tự động hóa công việc của tôi, trong một chừng mực nào đó, cũng đang tiến trên con đường như vậy. Từ chỗ nó là một kỹ năng cộng thêm, làm vì thích, bây giờ nó trở thành một kỹ năng rất nên có, công việc nào cũng cần.

Bàn về chuyện chuyên nghiệp, trước tiên, tôi nghĩ quan trọng là nhận thức rằng việc làm chuyên nghiệp khác với nghiệp dư, và làm phần mềm không nằm ngoài quy luật đó. Tôi nói chúng khác nhau không phải để tâng bốc, quan trọng hóa cá nhân. Ví dụ như việc đi hát: Xung quanh ta có rất nhiều người hát được. Có thể bạn hát karaoke rất hay. Có thể bạn hát hay đến mức có người mời bạn hát đám cưới. Nếu cả đời bạn chỉ hát karaoke đám cưới cho vui thì không vấn đề gì. Nhưng nếu bạn nghĩ ca sĩ chỉ là một người hát karaoke hay hát đám cưới, rồi bỗng dưng may mắn có nhiều người nghe và biết tới là rất sai. Mặc dù nhiều ca sĩ được khám phá ra vì họ hát đám cưới hay, nhưng sẽ tốt hơn nếu chúng ta nhận ra khoảng cách giữa hai người đó rất xa nhau. Những ca sĩ chuyên nghiệp như Mỹ Tâm hay Taylor Swift chẳng hạn, họ cống hiến hàng chục năm mà vẫn có bài hát mới. Tôi chắc chắn một điều, phần lớn những thành công đó không nhờ họ có cảm hứng hay có may mắn.

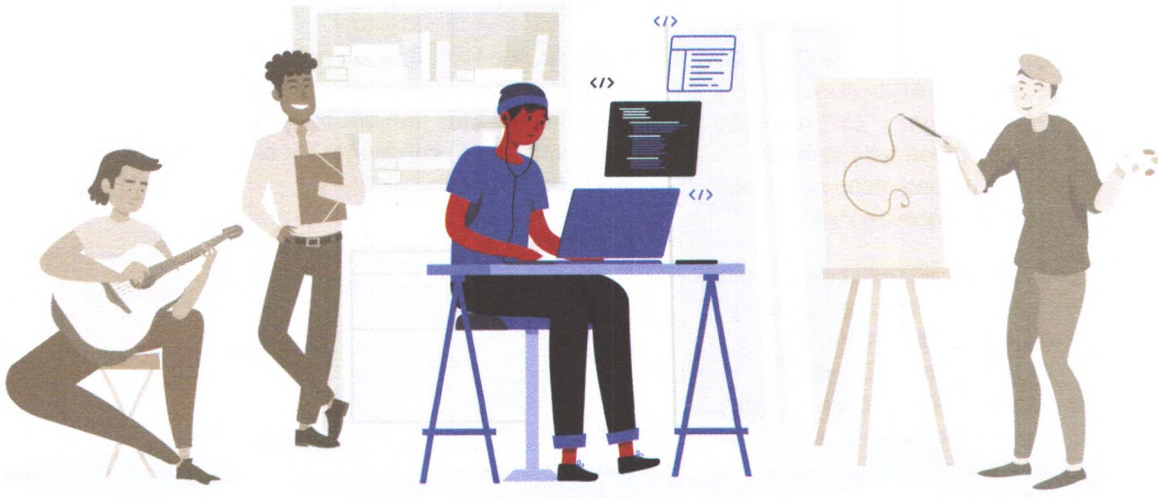
Điểm giống nhau giữa hát và code là rất nhiều người làm vì thích, đó là một phần của sự sáng tạo và đi lên từ việc làm nghiệp dư, làm cho vui. Cá nhân tôi viết phần mềm đầu tiên năm 12 tuổi. Tôi viết báo PCWorld về lập trình cho vui từ khi 14 tuổi để lấy tiền tiêu vặt. Phần mềm đầu tiên tôi viết "nguồn mở" cũng là vào những năm đó và nói chung, tôi đều viết code cho đến bây giờ. Đó là những việc làm chơi, nghiệp dư. Hiện tại, tôi được người ta trả tiền để viết phần mềm - đó là khi tôi biết bản thân đã chuyển từ việc làm chơi thành làm thật.

Từ thời gian đó đến giờ, rất nhiều khi tôi tự hỏi: "Khi biết mình thích một thứ, làm cách nào để biết mình đang làm đúng hoặc đi đúng hướng trên con đường đó? Làm cách nào để chuyển từ làm chơi thành làm thật?". Tôi từng có những hiểu sai về việc này, nghĩ rằng mình biết nhưng thực tế thì không. Nếu biết trước, tôi đã tiết kiệm được vô khối thời gian.

1. Ít nhất, bạn nên học hết Đại học chuyên ngành, và nếu có thể, học tiếp bằng Master.

Bạn không nên nghĩ rằng mình có thể "tự học" được tất cả mọi thứ bằng cách mày mò, và dành thời gian dư thừa để làm việc khác (ví dụ làm startup). Tuy "nghề" viết phần mềm nghe là lạ so với nhiều nghề khác, nhưng nó cũng không mới lắm. Và khi một con đường đã được người khác khai hoang từ lâu, ta hãy cứ đi con đường người khác đã biết và tạo lập sẵn, không cần khai phá lại. Bạn có thể nghĩ mình tự học, tự làm được, nhưng sẽ rất tốn thời gian, và phải đối mặt với muôn vàn câu hỏi không có câu trả lời (nhưng người khác có, và nếu bạn theo người ta, bạn đã không vướng phải).

Có nhiều điều bạn không biết là mình không biết nếu không học Đại học. Không ai bây giờ nghiên cứu vũ trụ bằng cách tự chế kính viễn vọng, rồi tự nhìn lên trời để phát minh ra định luật mới cho dù sách danh nhân viết



nhiều người đã làm như vậy (những việc đó bạn đã chậm chân 500 năm). Tương tự, bạn không nên nghĩ mình sẽ hiểu được máy tính bằng cách ngồi ở nhà và tự học. Đại học là nơi bạn được đào tạo chính quy, nghiêm chỉnh, luôn là con đường tốt nhất, ngay cả khi bạn đã có vốn liếng kha khá về kiến thức lập trình khi vào Đại học/Cao học.

Với cá nhân tôi, khi vào Đại học, tôi không ghi danh nhập học ngành máy tính vì nghĩ: "Hiểu được làm cách nào để code rồi thì học cái khác chứ học máy tính làm gì nữa?". Sau này, khi nhận ra mình phải học nghiêm túc vì có rất nhiều điều chưa biết, tôi chuyển qua học Khoa học Máy tính.

Nếu muốn làm startup thì bạn nên làm startup, nhưng nếu bạn không xuất chúng thiên hạ, bạn chỉ có thể chọn giữa làm startup hoặc làm phần mềm, chứ không phải cả hai. Nhiều người sẽ hỏi: "Vậy tại sao các ông trùm về công nghệ như Bill Gates hay Mark Zuckerberg đều là những kẻ bỏ học?". Tôi xin được trích lời Anh Phạm[1] : "Có bao nhiêu người bỏ học trở thành Bill Gates và bao nhiêu người bỏ học trở thành ăn xin?"

Ở đây, tôi giả thiết bạn vào được ngôi trường tốt, học chương trình tốt. Nếu học trường dở, thầy cô kém thì là chuyện khác. Tôi nghĩ giải pháp khôn ngoan là chuyển qua trường tốt. Giải pháp không khôn ngoan là bỏ học hẳn rồi tự học.

2. Đừng là nghệ sĩ, nhà hoạt động xã hội, kẻ khôn lỏi, hay người làm việc tùy hứng cùng lúc bạn viết code. Làm phần mềm là công việc kỹ thuật sáng tạo - một công việc kỹ thuật trước tiên.

Viết code không phải viết thơ hay viết văn. Viết code là viết lệnh mạch lạc để máy tính làm việc rõ ràng. Làm máy tính là phải học cách làm một người kỹ sư chứ không phải làm nghệ sĩ. Vì thế, về mặt kỹ thuật, viết code, viết tài liệu cho phần mềm, viết comment cho code, viết commit message đều phải mạch lạc, rõ ràng và đi vào gốc rễ của vấn đề. Mỗi dòng lệnh bạn viết cần có 1 mục đích cụ thể và duy nhất. Bạn sẽ sai rất nhiều và phải rất thẳng thắn nhận cái sai của mình.

Về công việc, bạn cần có kỷ cương và làm việc một cách chuyên nghiệp chứ không phải làm tùy hứng. Bạn cần nhận ra đâu là làm có hiệu quả và đâu là làm để lấy le với người khác (trong đó, làm không điều độ, thức đêm hôm là những việc điển hình của người làm việc nghiệp dư).

Khi viết code, người làm nghiệp dư thường mắc phải những lỗi mà họ nghĩ là mình "thông minh."

Ví dụ, lần đầu tiên tôi nhận ra bản thân không thông minh như mình tưởng là khi ông giáo sư dạy Khoa học Máy tính bắt tôi những lỗi như không viết dấu {} khi câu lệnh if chỉ có một dòng như thế này:

```
if (condition)
  doSomething();
return 0;
```

[1] *Ông Phạm Tuấn Anh hay còn được biết đến là Anh Phạm (Gấu) - phiên dịch của Tổng thống Mỹ Barack Obama trong chuyến thăm Việt Nam*

Nếu bạn không quen với ngôn ngữ như C, đoạn mã này sẽ gọi hàm doSomething() nếu như condition đúng. C cho phép chúng ta bỏ qua ngoặc nhọn nếu như ta chỉ gọi một hàm. Đoạn mã này tương đương với cách viết dài dòng thế này, khi tôi không biết, tôi tưởng rằng viết ngắn là tốt:

```
if (condition) {
  doSomething();
}

return 0;
```

Vấn đề của việc viết không có dấu {} là khi người khác làm việc trên code của tôi muốn làm thêm một việc khác nữa, mà viết như thế này:

```
if (condition)
  doSomething();
  doSomethingAfterwards();
return 0;
```

Như vậy, nếu gặp phải lỗi thì rất khó để nhận ra vấn đề nằm ở đâu. Vấn đề ở đây là đoạn mã này sẽ gọi doSomethingAfterwards() khi condition là sai vì nó tương đương viết như sau:

```
if (condition) {
  doSomething();
}

  doSomethingAfterwards();
return 0;
```

Người làm việc chuyên nghiệp không chỉ biết làm việc hiệu quả, mà còn biết làm việc để người khác hiểu được mình, kế tục được việc mình làm và ít bị bất ngờ nhất, chứ không phải người làm ma mãnh, khôn lỏi, thông minh nhất. Trường hợp người kế sau không biết tại sao những thay đổi về code đó đã diễn ra, diễn ra để làm gì, vấn đề nó khắc phục là gì, kết quả như thế nào, thì khi dự án trở nên rắc rối hơn, con người đông lên, việc một câu lệnh thực hiện chức năng gì sẽ vượt khỏi tầm tay và có thể xảy ra những hậu quả đáng tiếc.

Nếu bạn muốn tỏ ra thông minh nhất, hãy tham gia gameshow, đừng đi viết phần mềm.

3. Đừng là một kẻ đơn độc. Có rất nhiều người trong thế giới này giỏi hơn bạn và bạn sẽ học được rất nhanh khi làm việc với người giỏi hơn mình.

Một vấn đề khi làm vì ý thích là bạn thấy mình có thể tự làm nhiều thứ. Vì thế, bạn làm rất nhiều thứ một mình trong bóng tối. Bạn có thể nhốt mình trong căn phòng 24/7, không tiếp xúc với ai, không nói chuyện với ai, không ai biết bạn đang làm gì, và bạn nghĩ mình học được nhiều điều mới lạ. Bạn vẫn có những sáng tạo của riêng mình hàng ngày và thấy việc đó rất tốt. Bạn làm thế nào thì bạn cũng có niềm vui. Vấn đề là bạn sẽ không thấy điểm mù của bản thân và vùi đầu vào một việc không đáng làm, trong khi có người chỉ cho thì nhanh hơn rất nhiều. Điều này đặc biệt đúng với các bạn học sinh phổ thông hay sinh viên mới vào Đại học,

vì cảm thấy mình đơn độc khi có sở thích riêng, kỳ lạ, khác người và không chơi được với những người cùng trang lứa.

Để giải quyết câu chuyện "tự hát - tự thâu băng - tự đưa lên YouTube" này, giải pháp là học thầy và học bạn. Khi bạn có những người ít nhất là tốt bằng mình, bạn sẽ nhận ra cần phải làm gì rất nhanh. Để làm được việc đó, thay vì làm dự án cá nhân hay làm trong nhóm nhỏ, tốt nhất bạn nên tìm hiểu và tham gia vào một dự án phần mềm chuyên nghiệp. Các dự án này phải có đội ngũ làm nghiêm túc hẳn hoi. Tất cả các dự án có nhiều người làm ở các công ty phần mềm chuyên nghiệp (như Google, Facebook, Redhat) tham gia điều hành và gửi mã nguồn lên đều là những dự án "đạt chuẩn". Chúng không khó để tìm trên Internet, những dự án uy tín có thể kể ra như Debian, Linux Kernel, Buildroot, OpenWRT, KDE, GNOME, OpenSSL, Python đều là các dự án rất chuyên nghiệp và họ luôn tìm người tham gia. Khi bạn commit code vào những dự án đó, những người đi trước sẽ rất nhanh chóng chỉ cho bạn những lỗ hổng trong kiến thức và kỹ năng của mình. Cộng với việc bạn luôn có thể nói với mọi người rằng bạn đã tham gia đóng góp mã nguồn vào những dự án đó.

Nếu bạn tham gia các dự án nhỏ hơn của những cá nhân làm web, như các web framework, thư viện javascript hay ứng dụng nhỏ trên Github, trừ khi có lý do rõ ràng, tôi nghĩ rằng sẽ không hiệu quả cho việc phát triển cá nhân bằng dự án lớn. Bởi những người làm những framework hay dự án nhỏ cũng chỉ làm nghiệp dư, họ sẽ không có thời gian, công sức và kinh nghiệm để nắn sửa cho bạn khi bạn sai.

Nếu bạn muốn tham gia nghiên cứu, các lab uy tín ở trường Đại học lớn sẽ tốt hơn các lab ít người ở các trường Đại học nhỏ hơn. Các lab có nhiều sinh viên học Khoa học Máy tính sẽ có nhiều người để bạn học hỏi. Có nhiều trường hợp, như trường hợp của tôi, tôi đã làm việc Khoa học Máy tính trong một lab mà tôi là người duy nhất được đào tạo về máy tính. Trong thời gian đó, tôi không phát triển được nhiều kỹ năng làm phần mềm tốt cho bản thân.

4. Có tiền hay sự quan tâm của xã hội là tốt nhưng nó không phải là vấn đề.

Tôi nghĩ các bạn trẻ (teen) mắc phải cạm bẫy lớn là nghĩ việc mình làm rất tốt, rất quan trọng nếu nhận được tiền hoặc sự chú ý của ai đó. Bạn sẽ gây dựng nên Microsoft hay Facebook vì bạn thu được 1000 USD/tháng, hoặc vì bạn nghĩ ra điều gì đó mọi người đang cần. Tôi nghĩ sự nông nổi này sẽ giảm dần theo thời gian khi mỗi người lớn lên nên vị phụ huynh nào có con nhỏ đang như vậy cũng không nên lo quá. Nhưng có chút tiền không thôi thì không phải là lý do để không thèm học Đại học, không thèm mở rộng chân trời của mình. Mặt khác, đó có thể là lý do ngu ngốc nhất để không học tiếp.

Còn hack để kiếm tiền hay phục vụ mục đích của người khác thì luôn luôn sai, không cần bàn cãi. Nếu ai nghĩ rằng mình hoành tráng vì có thể hack, đó là người không chỉ không có khả năng, nghiệp dư, trẻ con, mà còn là một người độc hại cho cuộc sống.

Nói về chuyện tiền bạc, tôi rất yêu mến và muốn nhắc đến những người làm Phần mềm Tự do (Free Software), ở Việt Nam hay gọi là phần mềm mã nguồn mở (sở dĩ tôi dùng chữ "Phần mềm Tự do" như chữ của ông Richard Stallman[1], tức là coi trọng quyền người dùng chứ không coi trọng mã nguồn của nó). Điều đáng lo ngại là ngay cả những kỹ sư phần mềm ngày nay cũng ít khi để ý phát triển và tham gia vào các dự án Phần mềm Tự do. Có lẽ, bạn nghĩ những người phát triển Phần mềm Tự do sống khép kín, để râu xồm xoàm, sống dưới tầng hầm, vì viết phần mềm cho không thì đâu có tiền. Sự thực là phần lớn những người tôi biết tham gia phát triển Phần mềm Tự do đều có công việc rất tốt. Công việc thường tốt tới mức họ còn thời gian và sức lực phát triển Phần mềm Tự do trong tuần làm việc hoặc ngoài giờ làm việc. Rất nhiều trong số họ đang làm cho những công ty lớn, đặc biệt Google có lượng rất lớn các kỹ sư có dự án Phần mềm Tự do.

Bản thân tôi cũng là người đi lên từ Phần mềm Tự do. Có thể bạn biết tôi từ phần mềm Crankshaft, một phần mềm tự do cung cấp tính năng thông minh cho xe hơi đời cũ, nhưng tôi không bắt đầu từ đó. Nhờ tham gia vào thế giới phần mềm tự do mà tôi được nhiều người viết thư giới thiệu để vào trường Đại học và sau đó là chương trình học Tiến sĩ ở Mỹ. Cũng nhờ phát triển phần mềm tự do mà tôi nhận được rất nhiều lời mời phỏng vấn các công việc khác nhau trên khắp nước Mỹ. Tôi gia nhập Tesla cũng vì có người làm việc ở đó biết tới phần mềm tự do tôi viết. Để so sánh, chưa từng ai mời tôi việc gì vì họ thấy tôi có bằng cấp cao. Một phần mềm tự do tôi viết trong 3 tháng vì tôi thích có giá trị hơn một bằng cao cấp tôi làm trong 6 năm mà không có hứng thú nhiều vì đó là công việc.

Có vài yếu tố để một dự án Phần mềm Tự do hay việc đóng góp vào dự án Phần mềm Tự do là điểm rất mạnh trong hồ sơ xin việc của bạn:

- Thứ nhất, mã nguồn của bạn đã được rất nhiều người đọc và bạn có thể cung cấp mẫu thoải mái.
- Thứ hai, bạn không cần chứng minh mình có khả năng lập trình rất tốt.
- Thứ ba, bạn chứng tỏ mình là người làm việc được với người khác và hiểu người khác muốn làm gì.
- Và điều tối quan trọng: Bạn rất yêu thích công việc của mình, bạn sẵn sàng bước rất xa để làm những việc có thể không lợi lộc gì cho bản thân.

Như vậy, có rất nhiều lý do nếu bạn là sinh viên hay một người đang đi tìm việc để đóng góp cho các dự án Phần mềm Tự do.

4 điểm trên đây đều không dễ thay đổi, dễ làm trong ngày một ngày hai. Đó chính là lý do tôi viết bài này: Nếu tôi biết trước thì đã dễ dàng tiết kiệm được rất nhiều năm rồi. Nhưng tôi hy vọng, ai đọc được may ra có thể tiết kiệm được vài năm chăng?

Chúc chúng ta sẽ cùng trở thành những người tốt hơn trong thập kỷ mới!

---

[1] *Một nhà hoạt động vì phần mềm tự do, một hacker và một nhà phát triển phần mềm.*

Lập trình viên *có cần học Đại học?*

Tác giả: **Hien Nguyen - Software Engineer**

Tôi may mắn được Spiderum ưu ái đặt viết bài để thảo luận về một chủ đề xưa cũ nhưng luôn mang tính thời sự: "Lập trình viên có cần học Đại học?". Với vai trò (đã từng và hiện tại) là một sinh viên, giảng viên đại học, lập trình viên, nhà quản lý và trainer của những khoá học cho các lập trình viên hiện đại, tôi hy vọng bạn sẽ tìm thấy thông tin bổ ích dưới nhiều góc nhìn khác nhau về một chủ đề không xa lạ.

Trước khi đi vào vấn đề cụ thể, chúng ta cần đồng ý với nhau 2 điểm:

1. *Thời gian học Đại học (nếu có) rất quan trọng.*
2. *Thời gian và kết quả học Đại học không đủ quyết định thành công trong sự nghiệp.*

Thứ nhất, thời gian học Đại học là quan trọng. Sự nghiệp của một nhân sự trong ngành Công nghệ thông tin (CNTT) thường kéo dài khoảng 40 năm, như vậy, 4 năm học đại học chiếm khoảng 10%. Đặc biệt, đây là 10% thời gian bạn bắt đầu sự nghiệp, giai đoạn quan trọng tạo sức bật. Để dễ hình dung, nếu ví sự nghiệp của bạn như 1 năm, thì thời gian này tương đương 1 tháng sau Tết. 1 tháng dài hay ngắn? Bạn có thể đánh giá thông qua đợt dịch Covid-19 vừa qua. 1 tháng này quan trọng hay không? Bạn có thể đánh giá qua việc mọi kế hoạch năm bị điều chỉnh chỉ vì 1 tháng vừa qua, từ vĩ mô tới vi mô.

Thứ hai, 10% thời gian tất nhiên không đủ quyết định hoàn toàn tới thành công, bởi dù thế nào, bạn vẫn còn 90% thời gian. Dịch Covid vừa qua đã cuốn trôi 1 tháng (tương đương 10% thời gian trong năm) của các doanh nghiệp, cá nhân; song họ vẫn có thể đạt được 1 năm thành công nếu vượt lên mạnh mẽ các tháng sau đó. Tương tự, nếu trót có thời gian học Đại học "đáng bỏ đi", cơ hội để gây dựng sự nghiệp tốt vẫn còn nếu bạn biết điều chỉnh nỗ lực của mình.

Tất cả nằm ở sự nhận thức đúng đắn và cách thức điều chỉnh hành vi qua một số thông tin bài viết bàn luận dưới đây.

TRƯỜNG ĐẠI HỌC DẠY BẠN NHỮNG GÌ?

Những kiến thức, kỹ năng được các trường Đại học sắp xếp vào từng môn học, thường được tổ chức thành 3 cấu phần sau:

Những môn học "căn bản"

Mục tiêu của trường Đại học luôn là đào tạo kiến thức, kỹ năng căn bản. Mức độ căn bản thường được sắp xếp theo tiến trình như sau:

Những kiến thức "căn bản": Tập trung vào 1 - 2 năm học đầu tiên, bao gồm: toán học, vật lý, cách máy tính hoạt động, tiếng Anh,... Rất tiếc, những môn học này có tính phân cực cao nên ít tạo ra hứng khởi cho số đông. Trong khi tiếng Anh là kỹ năng cần thiết của 100% nhân sự CNTT thì toán học và vật lý lại chỉ thực sự hữu dụng cho 1% công việc sau này.

Những kiến thức "nền tảng ngành": Tập trung vào 2 - 3 năm học đầu tiên, bao gồm: lập trình, cấu trúc dữ liệu & giải thuật, cơ sở dữ liệu, hệ điều hành, mạng máy tính,... Đây là những kiến thức cực kỳ quan trọng vì dù bạn có làm việc trong lĩnh vực nào của CNTT thì đây là những kiến thức bắt buộc phải biết.

Những kiến thức "chuyên ngành": Đây là những môn học chuyên ngành, tập trung vào năm thứ 3 và thứ 4, bao gồm: cách phân tích và xây dựng một hệ thống thông tin, cách quản trị một hệ thống, cách làm cho máy tính "thông minh" hơn,...

Những môn học "hên xui"

Đây là những môn học mang lại giá trị nhiều hay ít phụ thuộc định hướng sự nghiệp của bạn, như: lập trình web, lập trình di động, quản trị hệ thống,... Nếu bạn mong muốn trở thành một quản trị viên hệ thống, việc hiểu biết sâu sắc về lập trình là không quá cần thiết. Đại học FPT là trường Đại học đặc biệt khi trang bị cho sinh viên 2 ngoại ngữ: tiếng Anh và tiếng Nhật. Với đa số sinh viên, đây là "hên" vì kỹ

năng tiếng Nhật giúp mở ra một lối đi mới trong sự nghiệp: Khả năng làm việc trực tiếp với khách hàng Nhật Bản. Phần "xui" dành cho những sinh viên mà định hướng nằm ở thị trường nói tiếng Anh, bởi họ có khoảng thời gian ác mộng (nếu bạn biết học tiếng Nhật khó ra sao).

Những môn "chỉ để học"

Dù thế nào, chúng ta cần thừa nhận rằng trường Đại học có khoảng ¼ số môn học chỉ phục vụ duy nhất việc "để học": triết học, kinh tế chính trị,... Giá trị mà chúng mang lại thường không tương xứng với thời lượng và công sức bạn bỏ ra.

Do chương trình đào tạo một chuyên ngành của trường Đại học phải được Bộ Giáo dục Đào tạo phê duyệt và đảm bảo tính nhất quán, nên chương trình của mọi trường sẽ luôn bao gồm 3 cấu phần kiến thức và kỹ năng kể trên. Điểm khác biệt giữa chương trình của các trường nằm ở tỉ lệ giữa các cấu phần. Những trường Đại học hàn lâm thường coi trọng và đặt tỉ lệ cấu phần "căn bản" ở mức cao với niềm tin rằng sinh viên sau khi ra trường có thể đi xa dựa trên nền tảng kiến thức vững chắc. Những trường Đại học mang tính công nghiệp hướng trọng tâm nhiều hơn vào kiến thức và kỹ năng giúp sinh viên bắt nhịp ngay với công việc sau khi ra trường. Không có gì đúng hay sai tuyệt đối: những sinh viên tốt nghiệp từ Đại học với xu hướng hàn lâm hay bị đánh giá là "không tạo ra giá trị đủ nhanh" ngay sau khi mới tốt nghiệp; ngược lại, những sinh viên từ Đại học mang tính công nghiệp bị đánh giá là "hạn chế trong việc giải quyết bài toán một cách sâu sắc" do yếu kiến thức và kỹ năng căn bản. Một xu hướng dễ thấy là các trường Đại học mang tính hàn lâm đang bổ sung thêm những môn học thực dụng hơn với ngành công nghiệp như lập trình web, lập trình mobile,... Trong khi những trường mang tính công nghiệp bổ sung thêm các môn học hàn lâm như hệ thống phân tán, trí tuệ nhân tạo,... Tất cả nhằm tìm kiếm điểm cân bằng trong kiến thức và kỹ năng giúp sinh viên sớm tạo giá trị ngay khi tốt nghiệp và có thể tiến xa trong sự nghiệp. Trong khi chờ đợi các trường Đại học tìm được sự cân bằng, sinh viên năng động luôn biết cách bổ sung điểm yếu từ trường của mình bằng các khoá học từ những trung tâm đào tạo CNTT khác - đó là điều đáng mừng.

TRƯỜNG ĐẠI HỌC CHO BẠN NHỮNG GÌ?

Những kiến thức, kỹ năng tôi đề cập ở trên giờ đây không phải lợi thế duy nhất chỉ thuộc về trường Đại học bởi bạn có thể dễ dàng học chúng qua những khóa học riêng biệt tại trung tâm hoặc online. Coursera, PluralSight là ví dụ: bạn có thể tìm thấy chương trình đầy đủ của một trường Đại học, từng môn học được cung cấp với chi phí rất rẻ (và miễn phí), thậm chí với chất lượng cao từ những Đại học hàng đầu thế giới như MIT, Stanford,... Tuy nhiên, có một điều chỉ trường Đại học mới có thể đem lại: *trải nghiệm đại học.*

Cộng đồng: Hằng năm, Đại học FPT biên một cuốn kỷ yếu, gọi là FU Alumni book, tập hợp bài viết về những kỷ niệm trong thời gian học tập. Những cuốn FU Alumni book đầu tiên khiến tôi khá bất ngờ khi ⅔ số bài viết có chung một câu chuyện: *thời gian học giáo dục quốc phòng tập trung.* Với tư cách là giảng viên, các thầy cô có thể

chạnh lòng khi những nỗ lực về kiến thức và kỹ năng được mình tâm huyết giảng dạy không phải điều ấn tượng nhất với sinh viên. Song từ góc nhìn của sinh viên, điều này rất dễ hiểu: chẳng phải mỗi khi họp lớp, chúng ta không mấy khi nhắc tới môn học, mà thường nhớ về những trải nghiệm cùng nhau đó sao? Những trải nghiệm tạo ra tình bạn không vụ lợi và bền vững theo thời gian. Một môi trường khác mà có lẽ chỉ trường Đại học cho bạn: ký túc xá và thư viện. Ký túc xá là nơi tuyệt vời để bạn học về "cách sống" khi chung sống với cả chục con người chứa đựng vô số định hướng, tính cách, lối sống,... khác nhau. Mỗi hành động của người này sẽ ảnh hưởng tới cuộc sống chung, nên xung đột là không tránh khỏi. Cách giải quyết những xung đột này luôn là kinh nghiệm và bài học quý báu cho phần đời còn lại của mỗi cá nhân.

Những cuộc thi: Nếu bạn yêu thích những cuộc thi, thử thách, trường Đại học gần như là lựa chọn duy nhất. Các trung tâm có thể tạo ra cuộc thi, sân chơi tốt, song tôi đang đề cập đến những cuộc thi "chính thống" và có uy tín như Olympic sinh viên, ACM-ICPC, Imagine Cup, Robocon,... Điểm lợi của việc tham gia những cuộc thi này là:

1. Bạn có trải nghiệm với thước đo "chuẩn" ở mức độ Quốc gia, Quốc tế;

2. Bạn có chứng nhận uy tín (nếu đoạt giải), điều mà những nhà tuyển dụng ưa thích;

3. Bạn có cơ hội giao lưu với những người cùng sở thích, đam mê ở diện rộng hơn (Quốc gia, Quốc tế).

Tất nhiên, những cuộc thi quy mô không bao giờ dành cho số đông, đó là câu chuyện của 5% (thậm chí là 1%) số sinh viên. Bạn nên cân nhắc giữa quyết tâm, năng lực, cơ hội và nhiều khi là may mắn. Gần đây, có một bạn sinh viên nhờ tôi tư vấn luyện tập để thi ACM - ICPC, tôi nói: "Với năng lực của em hiện giờ, em cần dành 3 - 4 giờ liên tục mỗi ngày trong 2 - 3 năm thì may ra mới có cơ hội đạt một giải ở quy mô khu vực" và bạn lập tức bỏ cuộc. Nhưng tôi cũng biết nhiều bạn sinh viên khác cố tình lưu ban 1 - 2 năm để được trải nghiệm thêm 1 - 2 kỳ Olympic hay ACM - ICPC, giống như trải nghiệm của ngày hội World Cup mà bất cứ cầu thủ nào cũng muốn góp mặt nhiều lần trong sự nghiệp.

Những mối quan hệ: Không thể phủ nhận rằng thành công về sự nghiệp của một cá nhân có đóng góp rất lớn từ chất lượng những mối quan hệ; và trường Đại học là nơi mang tới những quan hệ "chất lượng". Thứ nhất, hầu hết các giảng viên có chuyên môn và cách sống tốt; thứ hai, bạn bè trong trường Đại học cũng thường là những con người tốt. Có lẽ chỉ trường Đại học mới cho bạn cơ hội được học và làm việc cùng những người hàng đầu trong lĩnh vực của mình: những học sinh từng đạt giải Quốc gia, Quốc tế. Qua đó, bạn có thể biết họ xuất sắc (và cũng có thể là bình thường) ra sao so với bản thân mình. Chẳng dễ gì để có một thước đo tham chiếu tốt như vậy. Việc duy trì những mối quan hệ này còn rất tốt trong quãng sự nghiệp sau này. Các bạn học của tôi hiện có nhiều người làm việc tại những tổ chức lớn như Apple, Microsoft, Amazon, Google,... tôi luôn có nhiều điều học hỏi mỗi khi có dịp trò chuyện cùng họ.

Tôi biết lập trình từ khá sớm, điều đó từng dẫn tôi tới suy nghĩ không học Đại học để tiết kiệm 4 năm cuộc đời. May mắn là gia đình không cho phép tôi

Không thể phủ nhận rằng thành công về sự nghiệp của một cá nhân có đóng góp rất lớn từ chất lượng những mối quan hệ; và trường Đại học là nơi mang tới những quan hệ "chất lượng".

làm điều đó. Tôi thực sự biết ơn quyết định này. Với tôi, 4 năm Đại học là vô cùng giá trị; kể cả việc phải trải qua những thứ rất ngớ ngẩn vì chúng giúp tôi hiểu cuộc sống là như vậy: mọi thứ không hoàn hảo và ta phải chấp nhận một vài thứ để đạt được những giá trị lớn lao hơn.

TRƯỜNG ĐẠI HỌC KHÔNG CHO BẠN NHỮNG GÌ?

Bên cạnh những trải nghiệm đem lại lợi ích không thể chối bỏ, môi trường Đại học vẫn còn một số thiếu sót, theo tôi, là đáng tiếc.

Đạo đức: Đa số trường Đại học đào tạo về CNTT hiện nay bỏ ngỏ vấn đề đạo đức nghề nghiệp. Có rất ít trường đề cập tới nó một cách trực tiếp. Tôi được biết, thời điểm này chỉ có khoảng 5 trường Đại học tại Việt Nam có môn về đạo đức nghề nghiệp.

Bạn là quản trị viên của một hệ thống có thông tin 1 triệu khách hàng, bạn làm gì khi nhận được lời đề nghị 100 triệu để cung cấp thông tin 1 triệu khách hàng này cho bên thứ 3? Câu hỏi này khá dễ trả lời về mặt luật pháp. Bạn không được phép, đơn giản vì thông tin 1 triệu khách hàng là tài sản của công ty; thậm chí là tài sản của các khách hàng, và công ty bạn chỉ là người lưu trữ nhằm phục vụ họ.

Bạn làm việc cho một công ty xây dựng sản phẩm "chiếc nón kỳ diệu". Bạn xử lý thế nào khi vô tình biết sản phẩm "chiếc nón kỳ diệu" cho phép người chơi cược tiền và phục vụ mục đích đánh bạc online? Câu hỏi không dễ để trả lời, vì bạn không vi phạm pháp luật - bạn không phải người tổ chức đánh bạc và bạn không chính thức được biết vai trò của mình trong công việc đó. Lúc này, vấn đề đạo đức cần được xem xét.

Gần đây, có hơn 1.400 nhân viên Google đã ký tên vào bản kiến nghị, yêu cầu ban lãnh đạo Google giải trình về dự án Search Engine bởi họ nghi ngại ban lãnh đạo Google có thỏa thuận riêng với chính phủ Trung Quốc, qua đó những công việc họ đang thực hiện sẽ tạo ra một kết quả tìm kiếm theo mong muốn của chính phủ Trung Quốc và không khách quan như hệ thống Google thực hiện trên phần còn lại của thế giới. Cách hành xử của những nhân viên này đúng hay sai?

Sẽ có hàng trăm tình huống như vậy xảy ra trong quá trình hành nghề của bạn. Vậy đâu là quy tắc chung cho việc hành xử? Rất tiếc, trường Đại học đã không dạy bạn dù đây là một trong những trụ cột quan trọng chống đỡ ngôi nhà sự nghiệp của mọi cá nhân.

Chủ đề này cần tới hơn một cuốn sách, tôi chỉ muốn đưa ra những gợi mở để bạn quan tâm và tìm hiểu rõ hơn về code of ethics/code of conducts[1] chung cho ngành CNTT và cả riêng lĩnh vực của mình. Qua đó, bạn cũng sẽ biết việc đi nhậu suốt đêm với bạn bè, dù sáng hôm sau vẫn đến công ty sớm hơn 10 phút so với giờ quy định nhưng cả buổi uể oải, làm việc kém hiệu quả có phải là một hành động đúng hay không.

Cách học: Không chỉ ở trường Đại học mà suốt những năm tháng từ tiểu học tới phổ thông, phương pháp học chưa bao giờ là môn học chính thức, trong khi đây là một kỹ năng quan trọng. Sự khác biệt giữa việc học Đại học và học phổ thông là gì? Có lẽ là khả năng tự học, tự "bơi". Nhiệm vụ của các thầy cô trong trường Đại học là định hướng kiến thức, kỹ năng cần thiết và truyền tải một phần giáo trình; nhiệm vụ của sinh viên là "ngốn" hết giáo trình đó. Nhưng làm sao bạn "ngốn" hết số kiến thức và liên kết chúng lại với nhau? Việc này đòi hỏi khả năng tự học.

Ngoài ra, do đặc thù của ngành CNTT, những kiến thức và công nghệ mới sản sinh nhanh hơn các ngành khác, kỹ năng học thực sự quan trọng trong phát triển sự nghiệp.

Hai thập kỷ trước, kiến thức được hầu hết lập trình viên sử dụng là lập trình web. Một thập kỷ gần đây là lập trình mobile. Một thập kỷ tới có lẽ sẽ là IoT, Blockchain,... Thị trường công nghệ có tốc độ xoay chuyển của quá nhanh nên sự "chậm chạp" của các trường Đại học là điều dễ hiểu. Những kiến thức, kỹ năng "tương đối căn bản" được trang bị ở trường Đại học nhanh chóng trở nên lỗi thời. Trách nhiệm trang bị kỹ năng mặc nhiên thuộc về bạn. Và bạn chỉ có một công cụ duy nhất để gánh vác trách nhiệm này: phương pháp học.

Với kinh nghiệm của mình, tôi cho rằng kỹ năng tự học là điều rất quan trọng, thậm chí chiếm tới 60% thành công sự nghiệp của một nhân sự CNTT. Và kỹ năng này chính là điểm quyết định tới thành công trong sự nghiệp tôi đề cập ở trên. Nếu bạn đã có thời gian học Đại học "đáng bỏ đi", bạn có thể cứu vãn sự nghiệp của mình bằng việc bắt đầu lại từ viên gạch đầu tiên: phương pháp học. Đến đây, bạn có thể hiểu tại sao tôi liệt kê **Cách học** là kỹ năng quan trọng.

Tìm được cách học đúng đắn với bản thân không phải là điều dễ dàng, bạn có thể bắt đầu với một khoá học căn bản được cung cấp bởi GS. Barbara Oakley và GS. Terrence Sejnowski trên Coursera là "Học cách học" (Learning how to learn). Ngoài ra, việc tự học luôn gặp phải nhiều khó khăn: kỷ luật, lộ trình học tập, cách đánh giá trong quá trình học,... Những kỹ năng này bạn có thể tìm kiếm và thực hành qua những cuốn sách self-help; song tôi thường hướng dẫn các bạn trẻ tự học qua một số nguyên lý:

[1] *Bộ quy tắc đạo đức*

1. Thường xuyên theo dõi, đánh giá xu hướng công nghệ, kết hợp mong muốn, định hướng sự nghiệp của bản thân cùng việc trao đổi với đồng nghiệp, qua đó, tìm ra kiến thức, kỹ năng mình cần học.

2. Xây dựng một lộ trình học, và quản lý việc học như một dự án thực sự. Bạn có thể sẽ cần cơ chế thưởng, phạt cho bản thân giống như làm việc trong một dự án, bằng mọi giá khiến dự án thành công.

3. Đối với ngành CNTT, không gì tốt hơn là thực hành. Bạn có thể xây dựng những ứng dụng từ rất nhỏ như quản lý chi tiêu của bản thân tới một sản phẩm lớn hơn như nền tảng giao dịch bất động sản. Việc duy trì pet project[1] là rất cần thiết.

Định hướng nghề nghiệp: Trải nghiệm của bản thân với những lần làm mentor cho các bạn sinh viên và đồng nghiệp cho tôi thấy rằng: định hướng nghề nghiệp trong ngành CNTT không đơn giản, thậm chí cả với những người làm nghề lâu năm. Tôi nên làm lập trình viên hay quản trị viên hệ thống? Tôi nên làm kiến trúc sư giải pháp hay quản lý dự án?... Những câu hỏi như vậy xuất hiện quá thường xuyên. Trái ngược với những nghề nghiệp đã tồn tại hàng trăm năm như ngân hàng, y tế, xây dựng,... nơi hệ thống công việc và vai trò được hình thành rõ ràng và kiểm chứng qua thời gian dài, do đặc thù non trẻ và phát triển nhanh của ngành CNTT, công việc và vai trò sẽ luôn biến đổi, biến mất và sản sinh. Hai thập kỷ trước, vai trò lập trình viên mobile gần như không tồn tại; giờ đây lại trở thành phổ biến; vậy hai thập kỷ nữa vai trò này có biến mất? Vậy tôi nên làm gì? Cách nhìn nhận về ngành, nghề, về bản thân và từ đó định hướng sự nghiệp chính là lời giải. Sớm trang bị kỹ năng này giúp bạn phân loại được những môn học "hên xui" tôi đề cập ở trên, bạn sẽ tập trung đúng chỗ và thúc đẩy sự nghiệp tiến nhanh hơn. Tôi từng may mắn được làm việc cùng một cộng sự tài năng. Bạn sớm có định hướng rõ ràng: trở thành một lập trình viên mobile trong một công ty nước ngoài và làm sản phẩm, lấy đó là tiền đề trở thành lập trình viên trong công ty lớn tầm cỡ quốc tế. Định hướng nghề nghiệp rõ ràng giúp bạn liên tục có những bước tiến trong kỹ năng và sự nghiệp.

Bạn cộng sự của tôi có thể là một trường hợp đặc biệt (thủ khoa Đại học Bách Khoa), phần đông không có được kỹ năng này sớm như vậy. Các bạn có thể tham khảo phương pháp mà bạn đó đã sử dụng như sau:

1. Quan trọng nhất là hiểu biết về bản thân: biết mình thích thú với công việc gì và đạt năng lực cao nhất với công việc gì;

2. Tìm hiểu thị trường thông qua những báo cáo và những người đi trước;

3. Đặt mục tiêu và liên tục thực hiện những thử nghiệm để điều chỉnh lộ trình thông qua việc học tập có chủ đích.

---

[1] *dự án ngắn, nhỏ do cá nhân tự nghĩ ra*

Bạn có thể tìm hiểu thêm về IKIGAI - phương pháp gốc được bạn tôi sử dụng.

Giao tiếp & cộng tác: Tất cả những gì chúng ta từng coi là "kỹ năng mềm", giờ đây đã là một phần của "kỹ năng cứng". Tôi đánh giá cao một số ít trường có môn học về phát triển bản thân, bao gồm: kỹ năng giao tiếp, quản lý thời gian, làm việc nhóm, thuyết trình,... Những kỹ năng này tuy không được đào tạo trong trường Đại học nhưng lại đóng góp lớn vào thành công sự nghiệp. Thế kỷ 21 đòi hỏi các kỹ năng gắn với 4 chữ C: *Communication (giao tiếp), Collaboration (cộng tác), Critical thinking (tư duy phản biện), Creativity (sáng tạo)*. Chúng xuất hiện trong môn học nào tại trường Đại học của bạn?

Ngành CNTT đang mở ra nhiều cơ hội việc làm tốt hơn bao giờ hết bởi nó không đòi hỏi bằng cấp. Song bạn nên hiểu, bằng cấp và kỹ năng là hai chuyện hoàn toàn khác nhau. Học Đại học hay không? Và học Đại học theo cách nào? Đây luôn là những câu hỏi rất khó trả lời khi bạn dễ dàng tìm thấy hàng trăm ví dụ và phản ví dụ: Bill Gates và Mark Zuckerberg không học Đại học; Larry Page và Sergey Brin[1] lại là những sinh viên ưu tú - tất cả đều là những tên tuổi lẫy lừng trong ngành CNTT.

Trong quá trình viết, tôi chợt nhớ mình đã đề cập tới nội dung tương tự về "Lập trình viên và bằng Đại học" trên blog cá nhân. Sau 5 năm, quan điểm của tôi về việc này không thay đổi nhiều, bạn có thể tìm đọc thêm tại web gurunh.com hoặc scan QR code dưới đây:

Xin cảm ơn.

[1] *Hai nhà sáng lập Google*

Chuyện mình được nhận vào Google

Tác giả: **Scarlet**
Sinh viên năm nhất ngành Computer Science
tại University of British Columbia, Canada

Các công ty nổi tiếng trên toàn cầu như Google, Facebook, Microsoft,... luôn là điểm đến mơ ước của nhiều người theo đuổi lĩnh vực công nghệ thông tin. Chúng ta vẫn bắt gặp những bài báo ca ngợi về môi trường làm việc ở những nơi đó, nhưng ít khi hiểu được quá trình họ tuyển chọn nhân viên như thế nào. Tác giả Scarlet - cô sinh viên năm nhất vừa trúng tuyển vào thực tập sinh tại Google - sẽ kể lại trải nghiệm thú vị xuyên suốt hành trình phỏng vấn ra sao.

"Google chỉ thuê những đứa thực tập sinh từ những trường top như Ivy Leagues thôi, phải GPA cao tít tắp hay có CV cực đỉnh thì may ra."

"90% thực tập sinh của Google toàn trên năm 2 đó cậu ơi, năm nhất như tụi mình làm gì có cửa..."

"Năm ngoái anh xem thống kê, trên toàn Canada họ chỉ cho vào có tầm 10-15 người cho mỗi kỳ thực tập à, chưa kể trung bình mỗi trường Đại học họ chọn có 2 người là cao nhất..."

Thế là có con bé năm nhất lon ton chạy đi nộp đơn.

Mục đích của bài này là kể lại toàn bộ quá trình từ nộp đơn đến phỏng vấn của mình vào Google - chuyện không quá đặc biệt nhưng trong khoảng thời gian vừa rồi mình đã khám phá ra nhiều điều hay ho về công ty này.

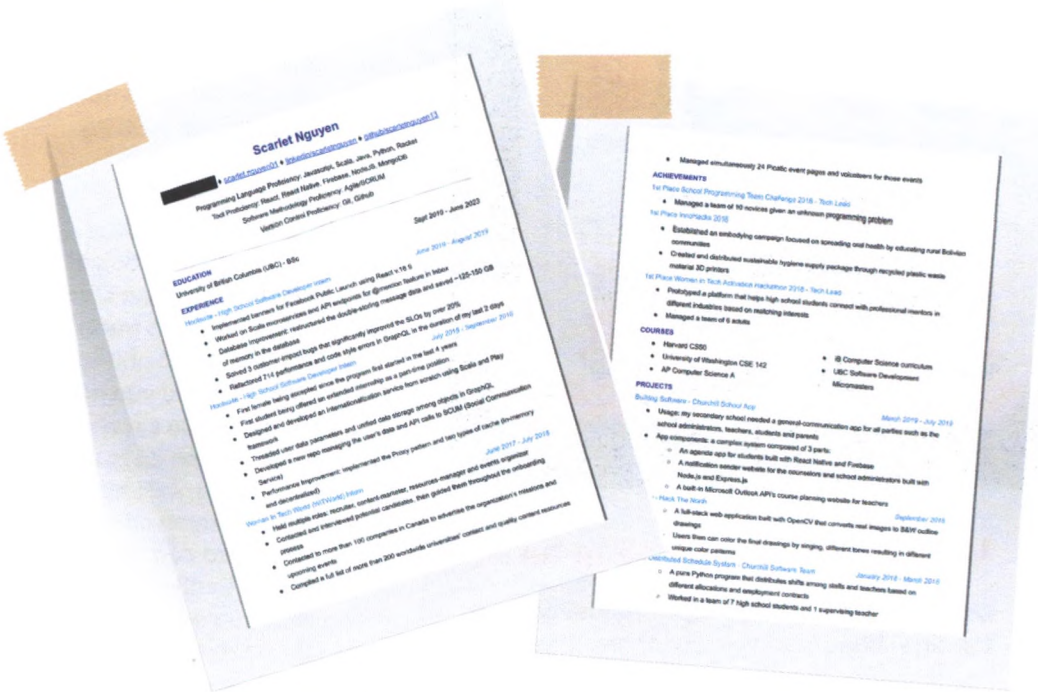
Vòng 1: Resume/CV (Nộp hồ sơ)

Google bắt đầu mở đơn tuyển internship (thực tập sinh) hằng năm vào giữa tháng 9, với lượng người nộp thường đông khủng khiếp. Theo những gì mình tìm hiểu được, vào năm 2018, họ nhận đến hơn 125,000 hồ sơ từ khắp nơi trên thế giới, nhưng chỉ tầm 1/3 trong số đó được chọn phỏng vấn và cuối cùng còn lại ~3,000 người nhận được offer thôi. Hiện nay, con số này còn tăng mạnh hơn vì ngành Computer Science/Computer Engineering (Khoa học Máy tính/Kỹ thuật Máy tính) đang là một trong những ngành hot nhất với vô số công nghệ nổi lên được mọi người nhắc tới mọi nơi như: deep learning, blockchain, quantum computing,... Ngay cả khối năm nhất của mình khi đó, dù chưa ai được chọn chuyên ngành cả nhưng hầu như đứa nào mình gặp cũng chăm chăm nhắm vào vùng đất màu mỡ này. Bạn bè mình đứa nào cũng hăng hái rải resume ở khắp nơi, trung bình đều nộp phải tầm 50 - 70 công ty gì đấy, còn mình thì lười nên nộp có mỗi Google...

Càng tìm hiểu, mình càng thấy quá trình thuê tuyển nhân sự trong những công ty lớn thật sự rất kỳ công. Ví dụ, để giải quyết vấn đề hằng năm nhận số lượng lớn hồ sơ như thế, họ phải dùng tới một hệ thống tự động hoá quá trình lọc resume/CV với một số keywords (từ khóa) nhất định nhằm loại bỏ những resume rác hoặc những người không đủ tiêu chuẩn ngay từ vòng đầu. Ngoài ra, chương trình internship bên này thường dựa vào rolling-basis (cuốn chiếu) - chỉ những ai quan tâm đủ tới internship mới có động lực nộp sớm, được xét trước và có khả năng được nhận sớm hơn. Môi trường cạnh tranh khốc liệt khiến đa số những thành phần đến sau vừa nộp đơn xong, còn chưa kịp nhấc mông đi phỏng vấn đã bị báo hết chỗ. Ngoài ra, mặt bất lợi nhất đối với mình có lẽ vẫn là việc hồ sơ của những đứa năm 1 sẽ tự động bị đẩy xuống chót của danh sách xét tuyển, nhường vị trí cao hơn cho những anh chị năm 2, 3 và 4 (những người có nhiều kinh nghiệm hơn hẳn). Google có thông báo thẳng rằng họ thậm chí không xét đơn của sinh viên năm nhất cho tới tận tháng 12.

Thế là mình còn mỗi cách cố gắng cải tiến resume bằng cách hỏi thăm một anh bạn lập trình viên bên Toronto góp ý giúp và nộp đơn sớm nhất có thể, tầm 3 ngày sau khi mở đăng ký (18/9/2019), với tâm trạng phấn khích xen lẫn hồi hộp.

Lúc nộp, mình nghĩ resume của mình được ngó qua thôi là vinh dự lắm rồi, chắc kiểu sẽ chuẩn bị tâm lý đợi tới tháng 12 để nghe thông báo bị từ chối là vừa. Đâu ngờ 3 ngày sau, mình nhận được email từ một cô recruiter (người tuyển dụng) cực kỳ xinh đẹp nhắn nhủ là:



Đây là cái resume mình dùng để nộp nè

"Chúc mừng em được chọn cho buổi phỏng vấn! Em có 4 ngày nữa cho đến lúc đó. Trước khi phỏng vấn, ký cho cô tầm 10 trang tài liệu với vài cái đơn NDA[1] này trong thời gian sớm nhất nhé. Chúc em may mắn."

Uhmmm, what.....?!

Tâm trạng của mình lúc đó rõ hỗn loạn. Vui cũng có, mà thú thật, khoảng thời gian gấp gáp khiến mình nhanh chóng trở nên hốt hoảng thì đúng hơn. Bình thường người ta chuẩn bị cho một buổi phỏng vấn cũng phải từ 2 tuần tới 1 tháng, với những công ty lớn, có người phải dành ra tới vài tháng, thậm chí tính theo năm luôn.

Và mình thì có 4 ngày.

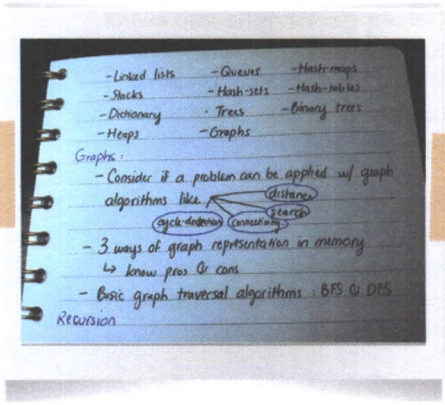
Chuyện bên lề: Interview Prep (Chuẩn bị phỏng vấn)

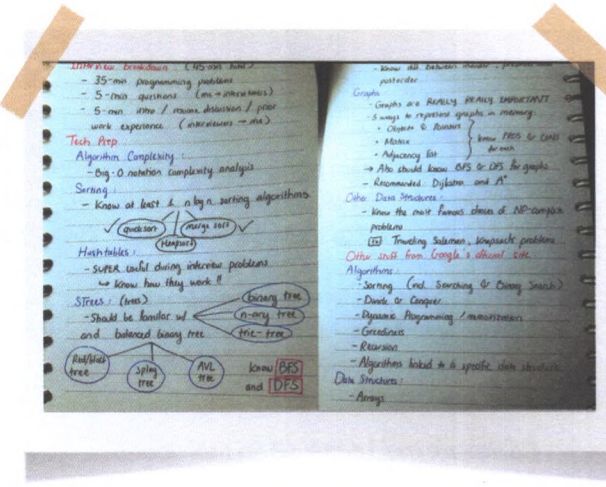
Technical interviews (phỏng vấn kỹ thuật) là loại interview phổ biến nhất trong giới lập trình, xoay quanh những câu hỏi nhằm đánh giá kiến thức và kỹ năng chuyên môn của bạn qua cách tiếp cận và giải quyết những vấn đề khó. Dựa vào trình độ của bạn, người phỏng vấn (thường là senior developers) đưa ra các câu hỏi có độ khó khác nhau, và ứng cử viên phải code giải pháp lên một cái bảng trắng ngay tại chỗ - hoặc trong trường hợp của mình, vừa giải thích qua điện thoại vừa code trên một cái shared Google Docs.

[1] *Non-disclosure agreement. Đây là một loại thỏa thuận không tiết lộ thông tin giữa ít nhất hai bên về tài liệu, kiến thức hay các thông tin bí mật mà các bên muốn chia sẻ với nhau vì mục đích chung nhưng cần hạn chế quyền truy cập bởi người thứ ba.*

Thế là có đứa lại lăng xăng đi tìm cái để học...

Tự dưng phát hiện ra, cơ bản mình sẽ phải chạy đi "ôn" những thứ mình sẽ học trong 4 năm tới. Mình còn tìm hiểu được Google là một trong những công ty coi trọng nhất khả năng mở rộng (scalability) và độ tối ưu (efficiency) trong một hệ thống. Vì thế, những câu hỏi của họ nổi tiếng khó, đòi hỏi phản ứng thật nhanh nhẹn với các thuật toán/phương trình được đưa ra và một người phải hiểu rõ độ phức tạp thời gian, không gian (time & space complexity) của đoạn code họ viết thế nào, từ đó tối ưu hoá ra câu trả lời tốt nhất dựa trên bối cảnh.





Nhưng như thế vẫn chưa đủ, ứng cử viên còn cần vừa tư duy vừa truyền đạt suy nghĩ của mình tới người phỏng vấn cùng một lúc, và đây có lẽ là yếu tố quan trọng nhất của một buổi phỏng vấn thành công: *giao tiếp (communication)*. Thường những câu hỏi sẽ bị xáo trộn và thiếu nhiều mảnh thông tin quan trọng, và công việc của một ứng cử viên tốt là liên tục hỏi người phỏng vấn cho tới khi nào nắm thật rõ đề bài thì thôi.
Các bước cơ bản thường sẽ nhìn như sau:

1. *Giải thích câu hỏi lại một lần nữa theo ý bản thân hiểu.*

2. *Hỏi những câu như "Tôi nghĩ là abcxyz... có phải vậy không?"*

3. *Đưa ra những ví dụ cho giải pháp của vấn đề, thường thì nên cho 1 trường hợp bình thường (normal case) và ít nhất 2 trường hợp hiếm gặp (edge/corner/boundary cases) → hỏi người phỏng vấn lại một lần nữa nếu bạn đang đi đúng hướng.*

4. *Quyết định cấu trúc dữ liệu (data structure) bạn sẽ dùng cho giải pháp.*

5. *Bắt đầu code, và trong khi code nên vừa viết vừa liên tục giải thích cho người phỏng vấn biết bạn đang áp dụng thuật toán nào hay mạch tư duy của bạn ra sao,...*

Mình đã đâm đầu vào học gần như ngày đêm cho sự kiện này, với sự giúp đỡ của các platform quen thuộc về lập trình như LeetCode và HackerRank. Thậm chí, mình còn nhờ vài người bạn giúp mình mô phỏng lại buổi phỏng vấn để quen hơn với kiểu vừa nói vừa động não liên tục. Nhớ những buổi đầu, mình còn lắp bắp mãi không biết nói gì, mặt mình lúc đấy nhìn ngu ngơ không chịu được làm đám bạn sau này cứ trêu mãi, haha.

Ngày 1/10/2019, mình có 2 buổi phỏng vấn trên điện thoại kéo dài 2 tiếng liên tục, mỗi buổi có một anh lập trình viên Google được chọn ngẫu nhiên từ khắp nơi trên nước Mỹ phỏng vấn mình. Lúc đó đã có con bé lo lắng đến nỗi phải dành ra tận 30 phút trước khi phỏng vấn để tự hít thở sâu trấn an bản thân, tạm bỏ lại những cảm xúc vụn vặt sau lưng và tập trung hết sức vào những gì sẽ xảy ra sắp tới.

*Cố lên, rồi mọi thứ rồi sẽ ổn thôi Scarlet. *tự vỗ vỗ vai**

Vòng 2: Interview

Phù, may là hai anh lập trình viên phỏng vấn mình ai cũng dễ thương thân thiện, tâm lý của mình được giải toả hơn rất nhiều.

Buổi phỏng vấn đầu tiên, mình được hỏi công nghệ mà mình đang yêu thích nhất là gì, nhằm mục đích "làm nóng người", ngay sau đó đi thẳng vào câu hỏi lập trình, không chút câu nệ tiểu tiết.

Khi nhận đề bài, việc đầu tiên là mình làm sáng tỏ đề bài bằng cách đặt thật nhiều câu hỏi và suy nghĩ thấu đáo những trường hợp cá biệt, rồi tiếp cận vấn đề bằng cách giải nguyên thuỷ nhất (brute force approach) để thiết lập một điểm bắt đầu. Việc liên tục nói ra suy nghĩ của mình khiến 1 tiếng trôi qua nhanh chóng. Một yếu tố chiếm kha khá thời gian nữa là mình phải liên tục nghĩ cách nâng cấp code ở phiên bản hiện tại chạy nhanh hơn/tốn ít dữ liệu hơn. Đặc biệt, mỗi khi mình gần hoàn thành đến nơi, anh phỏng vấn lại thêm vào "một vài vấn đề nhỏ" để tăng độ "thú vị", nên mình chỉ đành cười (khổ) và cố gắng giải tiếp. Tầm 5 phút cuối buổi, mình đã được hỏi anh đủ thứ về Google và nghe anh kể vài câu chuyện thật hay ho khiến mình vui đến nỗi cười toe toét, đâu phải lúc nào cũng có vinh dự được nói chuyện với kỹ sư Google đâu chớ.

Buổi phỏng vấn thứ 2 cũng gần giống vậy, chỉ là lần này thay vì được luyên thuyên chém gió về công nghệ, anh phỏng vấn đưa ngay đoạn code dài nhìn phát hoảng và hỏi mình:

- Đây là phần code reading. Em có biết khái niệm Promise trong lập trình không? Nếu không thoải mái anh có thể đổi câu hỏi.

- Dạ em không biết, mặt dù có nghe qua nhưng chưa dùng bao giờ. Nhưng mà em không ngại học cái mới, anh có thể giải thích sơ qua cho em khái niệm được không ạ? Em muốn thử sức.

Thế là tầm 3 phút tiếp theo mình phải học một kiến thức mới ngay tại chỗ và áp dụng nó cho đoạn code bên trên để viết ra kết quả (output) cuối cùng. Lần đầu tiên nhìn câu hỏi mình hoang mang kinh khủng, không biết đoạn code đang cố gắng làm gì cả nên thấy khó khăn vô cùng. May mà bình tĩnh lại sau đó, ngẫm nghĩ tầm vài phút sau thì mình nhìn ra cốt lõi vấn đề và viết đáp án đúng luôn. Anh phỏng vấn chắc bất ngờ nên thốt lên "Bingo!" và bật cười vui vẻ bảo không ngờ mình học nhanh

thế :). Sau đó, quá trình giải câu hỏi xảy ra tương tự như buổi phỏng vấn đầu, chỉ khác là mức độ khó hơn nhiều, nhưng may mắn là mình vẫn ra kết quả trong thời gian cho phép.

Kết thúc 2 cuộc phỏng vấn mình liền gục xuống bàn, còn tay chân thì bủn rủn hết cả. Trải nghiệm này vui và căng hơn nhiều so với dự đoán ban đầu, nhưng chung quy lại, nó vẫn vô cùng quý báu và có thể nói mình cảm thấy cực kỳ biết ơn khi một cơ hội hiếm hoi có thể đưa mình xa được đến mức này.

Vòng 3: Hiring Committee (Hội đồng tuyển dụng)

Thật lòng mà nói, đây là khoảng thời gian dàiii nhấttt trong đời mình từng trải qua. Tỉ lệ được chọn quá thấp khiến mình suốt ngày thấp thỏm không yên, càng không đành lòng nếu lỡ nhìn thấy thư từ chối mặc dù chưa bao giờ dám nghĩ bản thân có thể nhận được offer. Thế là tự làm bản thân lâm vào tình trạng thường xuyên giữ điện thoại bên người nhưng lại chẳng dám kiểm tra email...

Theo những gì mình tìm hiểu được thì sau buổi phỏng vấn, người phỏng vấn sẽ điền vào một đơn feedback dài và cực kỳ chi tiết về ứng cử viên, rồi gửi lại cho recruiter sớm nhất có thể. Thường quá trình này mất từ 2 - 3 ngày, và khi nhận được, recruiter sẽ bắt đầu soạn thảo, sắp xếp lại tất cả thông tin của ứng cử viên vào một packet (tệp), bao gồm:

- Resume
- Câu hỏi dùng trong phỏng vấn và câu trả lời của ứng viên
- Thư giới thiệu (nếu có)
- Một vài ghi chú nhỏ về ứng viên

và đưa nó xuống dưới cùng của danh sách ứng viên đang cần được duyệt. Ở Google, có nhiều hội đồng chuyên đưa ra quyết định thuê tuyển internship, được gọi là Hiring Committee (HC), họ thường là những người nắm giữ chức vị cấp cao (senior roles) và có kinh nghiệm lâu năm trong việc tuyển chọn một bộ phận nhất định tại công ty. Mỗi tuần, họ có một buổi họp (theo mình biết thường vào sáng thứ 3 hoặc thứ 5) để xét duyệt một lô tầm 7 - 10 ứng cử viên cho tuần đó và quyết định có đi tới offer hay không.

Quá trình từ recruiter đến HC thường sẽ xảy ra như sau:

- Trước cuộc họp 1 - 2 ngày, HR (bộ phận nhân sự) sẽ đăng tất cả tài liệu về lô ứng cử viên của tuần đó lên hệ thống và gửi email kèm link hồ sơ từng người tới mọi thành viên trong HC
- Khi nhận được email, mỗi thành viên HC sẽ phải cho một đánh giá chung chung (thuê/không thuê), mặc định điểm số họ cho là phù hợp từ 1 đến 4

  1 = Không nên thuê

  2 = Tôi nghĩ là không thuê, nhưng chúng ta vẫn nên xem xét kỹ hơn

  3= Tôi nghĩ đây là người phù hợp, nhưng chúng ta vẫn nên xem xét kỹ hơn

  4 = Nên thuê

(để được nhận, ứng cử viên phải cần điểm trung bình từ 3 trở lên)

và viết một nhận xét chi tiết về mỗi ứng cử viên.

Mọi thứ được thực hiện trên hệ thống một cách độc lập và ẩn danh trong khoảng thời gian này nhằm giảm thiểu thiên kiến từ đám đông lên một cá nhân

- Tới ngày họp, recruiter sẽ chiếu packet của từng ứng cử viên lên một màn hình lớn, và lúc này tất cả các đánh giá/điểm/nhận xét độc lập của từng thành viên trong HC tự làm trước đó sẽ được công khai cùng với hồ sơ nguyên bản của ứng cử viên

=> Thường 2/3 trong số ứng viên nhận được số điểm tương đối gần nhau từ các thành viên HC (khi xem xét độc lập) nên trong buổi họp, HC sẽ có quyết định nhanh chóng (khi xem xét tập thể) nếu họ muốn nhận ứng cử viên này hay không. Nếu một ứng viên nhận nhiều ý kiến trái chiều, HC sẽ dành nhiều thời gian hơn để phân tích và lắng nghe ý kiến từ mọi phía.

Có 4 yếu tố chính cần đánh giá một ứng viên:

1. *Khả năng phân tích và giải quyết vấn đề (qua buổi phỏng vấn)*
2. *Kinh nghiệm làm việc:*
   - *Viết code sạch, đẹp, hiệu quả cao, không quá chậm,... (từ buổi phỏng vấn)*
   - *Kinh nghiệm trong quá khứ (resume)*
3. *Khả năng lãnh đạo: tự biết phải làm gì khi không có nhiều chỉ dẫn từ cấp trên*
4. *Googliness: tính cách có phù hợp với văn hoá của Google hay không (qua cách nói chuyện trong lúc phỏng vấn và trao đổi email với HR)*

Trong trường hợp HC vẫn không quyết định được sau khi họp, sẽ có 2 hướng xảy ra:

1. *Xếp thêm một buổi phỏng vấn thứ 3 để có cái nhìn toàn diện hơn về ứng viên*
2. *Từ chối thẳng*

Trường hợp 2 phổ biến hơn vì Google có một văn hoá nặng về "false positive" - có nghĩa: thà lỡ người giỏi còn hơn thuê nhầm người tệ. Điều đó khiến HC quan niệm việc đang phân vân đồng nghĩa với "không nên thuê". Thêm vào đó, việc đặt lịch interview khá mất thời gian nên mọi người thường tránh trường hợp đầu.

- Sau buổi họp sẽ có kết quả ngay lập tức, và lúc đó recruiter bắt đầu gọi/email thông báo cho ứng cử viên về quyết định của HC

=> Dù được nhận hay từ chối thì ứng cử viên vẫn sẽ nhận được cuộc gọi

Kết quả

22/10 - đúng 3 tuần sau ngày phỏng vấn, mình lỡ cú điện thoại từ cô recruiter khi đang gật gù trong lớp Sinh học. Sau đó, khi phát hiện ra cuộc gọi nhỡ từ Google trên điện thoại, mình tá hỏa chạy như bay ra khỏi lớp và tìm chỗ nào có wifi tốt nhất để nhắn hỏi cô có thể gọi lại mình không. 10 phút sau đó, mình cứ cầm điện thoại, run thật run, không biết chuyện gì sẽ xảy đến đây nên cứ nhắm mắt cầu nguyện cái đã. Vì đời người ai cũng cần một chút may mắn...

"Em chào cô ạ"

"Chào buổi sáng, em cảm thấy thế nào?"

"Dạ tốt, cô thì sao?"

"Ừ cô vẫn khoẻ.... À, cô có tin tốt là,

CHÚC MỪNG EM ĐÃ ĐƯỢC NHẬN VÀO INTERNSHIP CỦA GOOGLE!

Em có thể dành ra 10 - 15 phút sắp tới để nghe về thông tin công việc cho hè năm sau không?"

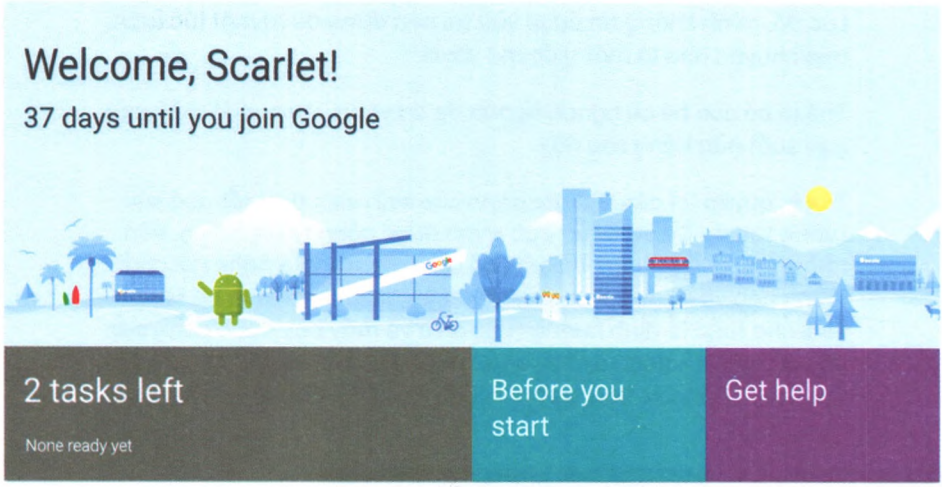
Lúc đó, mình không tin được vào tai nên đứng đờ ra một lúc luôn, mọi chuyện như là một giấc mơ vậy!!!

Thế là có con bé cứ ngoan ngoãn dạ dạ vâng vâng, mặt mày ngu ngơ suốt nửa tiếng sau đấy.

Trời ạ, quyền lợi của Google dành cho sinh viên thật tốt quá sức tưởng tượng. Cô recruiter bảo mình được công ty trả lương, tiền nhà và ăn cũng được bao hết (đồ ăn trong Google nghe nói chất lượng cao và ngon cực). Tuần đầu tiên, mình sẽ được qua New York cho tuần lễ định hướng (chỗ ở và vé máy bay được cung cấp đầy đủ luôn). Ngoài ra, ở Google, những thứ như gym, hồ bơi, dịch vụ mát xa hoặc xe đạp/xe hơi sẽ miễn phí hết cho nhân viên nữa.

Đây sẽ là bếp của tòa nhà mình làm việc vào hè năm nay

Còn đây là một trong những phòng giải trí



Họ còn có một đội chuyên đi thiết kế trải nghiệm cho internship để đảm bảo thực tập sinh có những trải nghiệm tuyệt nhất trong mùa hè như: đi xem phim, đấu bóng rổ, hoặc đi ăn cùng nhóm thực tập. Cô recruiter còn bảo mình sẽ có nhiều cơ hội gặp gỡ và học hỏi từ những kỹ sư lâu năm trong ngành qua những bài giảng và workshops dành riêng cho thực tập sinh nữa, nghe mà mình cười tít cả mắt lại luôn.

Vòng 4: Host Matching (Phân công dự án)

Sau khi ký hợp đồng xong hết, mình hiện đang ở giai đoạn chờ phỏng vấn các đội ở Google để chọn dự án làm việc cho 3 tháng hè năm nay.

Kết

Trải nghiệm nộp đơn cho Google của mình chỉ vỏn vẹn hơn 1 tháng thôi nhưng đã nhiều chuyện xảy ra, mình cũng học được rất nhiều thứ trong thời gian này. Nó phần nào chứng minh được không cần phải học trường quá cao siêu hay là thiên tài mới được nhận, vì những người biết mình từ trước sẽ rõ mình là một đứa cực kỳ bình thường, học hành cũng chẳng giỏi tới mức top hay thông minh xuất chúng gì cả, xuất phát điểm lại càng muộn so với bạn cùng lứa. Mình đã nỗ lực rất rất nhiều, thế nên thành công lần này do một phần nộp sớm và chuẩn bị kỹ, một phần kinh nghiệm làm việc mình có trước đó, và chắc chắn không thể thiếu một phần may mắn. Mình không rõ yếu tố may mắn chiếm bao nhiêu, nhưng có một điều mình chắc chắn: Nếu như mình sợ lúc đầu và tự ti rụt lại, sẽ không có bài này ở đây cho mọi người đọc đâu.

Hiện tại mình đang tập trung cho việc học trong trường nhưng sự háo hức dành cho mùa hè năm nay chưa bao giờ giảm. Nếu đủ thời gian thì mình cũng mong là được về thăm Việt Nam trước kỳ thực tập, vì một phần muốn được gặp vài anh chị mình cực kỳ mến trên Spiderum nữaaaa.

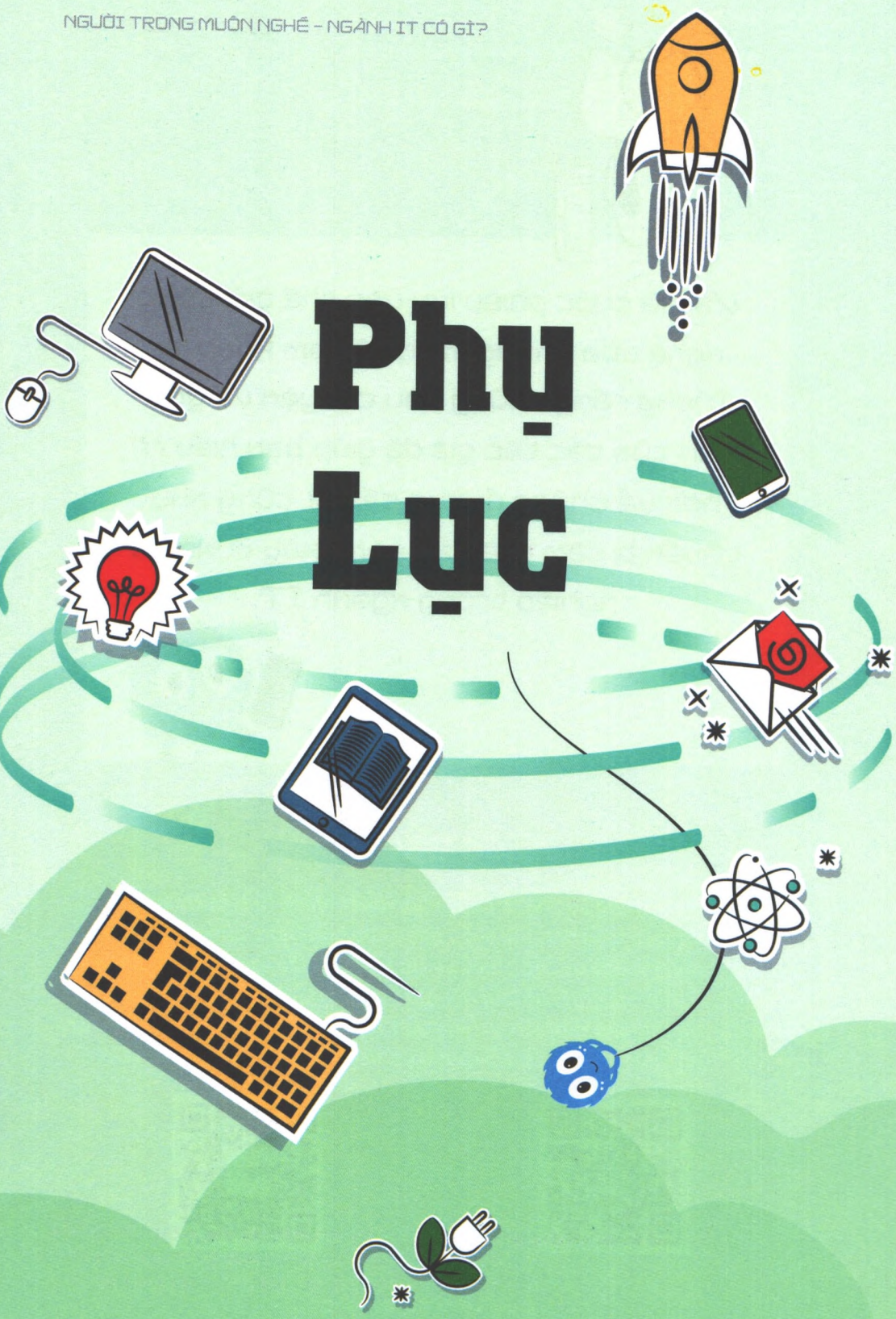
Cảm ơn mọi người vì đã đọc :)

Vậy là cuộc phiêu lưu vào thế giới công nghệ của chúng mình đã tạm khép lại. Mong rằng những câu chuyện và góc nhìn của các tác giả đã giúp bạn hiểu rõ hơn về chặng đường sắp đi, cũng như chuẩn bị tâm thế sẵn sàng xây dựng sự nghiệp trong ngành IT.

Góc bật mí:

Hành trang vào nghề chắc chắn nhất chính là kiến thức chuyên môn bạn thu thập được cho bản thân, cùng với những thông tin đầy đủ, đa chiều về thị trường việc làm. Để nâng cao kỹ năng tự học, hãy theo dõi những tác giả dạn dày kinh nghiệm từ Spiderum. Hoặc bạn có thể ghé thăm ứng dụng của TopCV để cập nhật những cơ hội việc làm mới nhất và trò chuyện trực tiếp với nhà tuyển dụng nhé:

Quét để tải ứng dụng



Phụ Lục

MỘT SỐ THUẬT NGỮ CHUYÊN NGÀNH

Fresher Sinh viên mới ra trường hoặc những nhân sự mới bắt đầu làm việc. Fresher thường chưa có kinh nghiệm và cần được đào tạo trước khi có khả năng làm việc thực tế.

Junior Những nhân sự ít hoặc chưa có kinh nghiệm chuyên môn. Junior đảm nhiệm các công việc không quá phức tạp, không yêu cầu chuyên môn cao. Khái niệm junior không hoàn toàn nhất quán mà phụ thuộc vào cách phân chia cấp độ của mỗi công ty.

Senior Những nhân sự dày dạn kinh nghiệm trong một lĩnh vực làm việc cụ thể, có khả năng xử lý các công việc yêu cầu chuyên môn cao. Tuy nhiên, giống như junior, khái niệm senior cũng không hoàn toàn nhất quán mà phụ thuộc vào cách phân chia cấp độ của mỗi công ty.

Freelancer Người làm việc tự do (freelance), không bị giới hạn về môi trường, địa điểm và thời gian làm việc.

Code/Source Code Mã nguồn của phần mềm

Git Phần mềm quản lý mã nguồn phân tán được phát triển bởi Linus Torvalds vào năm 2005. Hiện nay, Git trở thành một trong những phần mềm quản lý mã nguồn phổ biến nhất trên thế giới.

Library Thư viện (library) là tập hợp những đoạn code, chức năng đã được viết sẵn để sử dụng.

Framework Một bộ khung được cấu thành từ những đoạn code, thư viện, hay cả những giải pháp được tích hợp sẵn giúp lập trình viên tiết kiệm thời gian trong quá trình phát triển ứng dụng.

Build Quá trình chuyển đổi mã nguồn (source code) thành phần mềm

Demo Sản phẩm chưa hoàn chỉnh

Stackoverflow Website hỏi đáp về lập trình lớn nhất thế giới

Bug Lỗi hoặc hỏng hóc trong chương trình/hệ thống máy tính khiến kết quả tạo ra không chính xác hoặc hành xử theo những cách không lường trước.

Debug Quá trình tìm ra lỗi khiến chương trình máy tính hay hệ thống không hoạt động đúng. Bên cạnh đó, debug cũng giúp lập trình viên hiểu rõ quá trình thực thi của các mã lệnh.

Fixbug Sửa lỗi giúp chương trình hoặc hệ thống hoạt động đúng như mong muốn.

Backup Hành động tạo một bản sao của dữ liệu. Bản sao dữ liệu máy tính được lấy và lưu trữ ở nơi khác, có thể sử dụng để khôi phục bản gốc nếu bị mất dữ liệu.

Kick off Việc bắt đầu hay khởi động một dự án với mục đích để tất cả các thành viên chính thức liên quan đến nhau, cùng nhau hợp tác và chia sẻ những kỳ vọng chung của tất cả các bên.

Design pattern Một mẫu thiết kế phần mềm (design pattern) là một giải pháp chung, có thể tái sử dụng cho một vấn đề thường xảy ra trong bối cảnh nhất định khi thiết kế phần mềm.

Overtime (OT) Là khoảng thời gian làm thêm ngoài khung giờ làm việc trong quy định. Trong ngành CNTT, khi tiến độ dự án bị chậm, đội dự án thường phải OT để hoàn thành đúng thời gian dự kiến.

Clone Thuật ngữ thường được sử dụng trong cộng đồng lập trình viên, ám chỉ việc sao chép y hệt một đoạn mã code hoặc một phần mềm, hệ thống nào đó.

Deploy Việc triển khai phần mềm, bao gồm tất cả các hoạt động khiến một hệ thống phần mềm sẵn sàng cho sử dụng.

Outsourcing Là hình thức chuyển một phần công việc từ công ty ra thuê gia công bên ngoài. Trong ngành CNTT, các công ty nhận gia công phần mềm cho các công ty khác được gọi là công ty outsourcing để phân biệt với các công ty làm product - tự xây dựng và phát triển sản phẩm của chính mình.

Cache Bộ nhớ đệm chứa dữ liệu, các dữ liệu được nằm chờ yêu cầu từ ứng dụng hoặc phần cứng. Dữ liệu được chứa trong cache có thể là kết quả của tính toán trước đó, hoặc là sự trùng lặp dữ liệu được lưu trữ ở một nơi khác.

Virtual Private Server (VPS) Máy chủ ảo (Virtual Private Server) là dạng máy chủ được tạo ra bằng phương pháp phân chia một máy chủ vật lý thành nhiều máy chủ khác nhau có tính năng tương tự như máy chủ riêng, chạy dưới dạng chia sẻ tài nguyên từ máy chủ vật lý ban đầu.

Database Cơ sở dữ liệu (database) là một tập hợp tổ chức các thông tin hoặc dữ liệu có cấu trúc, thường được lưu trữ điện tử trong một hệ thống máy tính. Một cơ sở dữ liệu thường được kiểm soát bởi hệ thống quản lý cơ sở dữ liệu (DBMS).

Firewall Tường lửa (firewall) là hệ thống bảo mật mạng giám sát và kiểm soát lưu lượng mạng đến và đi dựa trên các quy tắc bảo mật được xác định trước. Một tường lửa thường thiết lập một rào cản giữa một mạng nội bộ đáng tin cậy và mạng bên ngoài không tin cậy, chẳng hạn như Internet.

Integrated Development Environment (IDE) Môi trường phát triển tích hợp (IDE) là phần mềm máy tính có công dụng giúp đỡ các lập trình viên trong việc phát triển phần mềm. IDE thông thường bao gồm công cụ viết mã nguồn (code editor), trình biên dịch/thông dịch và các công cụ kiểm tra lỗi (debugger).

Technical debt Nợ kỹ thuật (technical debt) là khối lượng công việc sẽ phải xử lý thêm khi lựa chọn giải pháp không tối ưu để đánh đổi lấy thời gian phát triển sản phẩm trong quá khứ.

Cloud computing Điện toán đám mây (cloud computing), còn gọi là điện toán máy chủ ảo, là mô hình điện toán sử dụng các công nghệ máy tính và phát triển dựa vào mạng Internet. Thuật ngữ "đám mây" ở đây là lối nói ẩn dụ chỉ mạng Internet và liên tưởng về độ phức tạp của các cơ sở hạ tầng chứa trong nó.

Internet of Things (IoT) Internet vạn vật (IoT) là hệ thống mạng lưới các vật thể vật lý, đồ vật được gắn với các cảm biến, phần mềm và các công nghệ khác nhằm mục đích kết nối và trao đổi dữ liệu với các thiết bị và hệ thống khác qua Internet.

Object-oriented programming (OOP) Lập trình hướng đối tượng (OOP) là phương pháp lập trình mà phần mềm được xây dựng xung quanh các đối tượng (object) thay vì chức năng và logic. Một đối tượng sẽ có dữ liệu được lưu trữ trong các thuộc tính (properties) và các hành vi (methods) của nó. Phương pháp này tập trung vào các đối tượng mà các nhà phát triển muốn thao tác hơn là logic cần thiết để thao tác chúng, nhờ đó, việc phân tích và xử lý các bài toán lớn trở nên hiệu quả hơn.

Software as a service (SaaS) Các phần mềm được nhà cung cấp xây dựng sẵn và phân phối qua Internet. Phần lớn người sử dụng có thể truy cập vào phần mềm ngay trên trình duyệt web, không cần cài đặt.

Platform as a service (PaaS) Mô hình mà nhà cung cấp không cung cấp phần mềm hay dịch vụ, thay vào đó PaaS cung cấp nền tảng cho việc tạo ứng dụng. Các nền tảng này được đưa ra qua Internet, giúp nhà phát triển có thể tập trung vào việc xây dựng phần mềm mà không cần quan tâm tới cơ sở hạ tầng.

Infrastructure as a service (IaaS) Mô hình mà nhà cung cấp xây dựng sẵn các cơ sở hạ tầng, cho phép người sử dụng thuê tài nguyên theo yêu cầu thay vì phải tự xây dựng phần cứng.

User Interface (UI) Giao diện người dùng (UI) là thuật ngữ ám chỉ những phần nhìn thấy của phần mềm, giúp người sử dụng phần mềm có thể tương tác với hệ thống.

User Experience (UX) Trải nghiệm người dùng (UX) là thuật ngữ ám chỉ những trải nghiệm như cảm xúc, thái độ của một người khi sử dụng phần mềm.

Refactor Là hoạt động chỉnh sửa lại mã nguồn đang có nhằm đạt được các tiêu chuẩn tốt hơn mà không làm thay đổi hành vi hệ thống về mặt chức năng.

Bottleneck Nghẽn cổ chai (bottleneck) là hiện tượng xảy ra khi khả năng của một ứng dụng hoặc hệ thống máy tính bị hạn chế nghiêm trọng bởi một thành phần duy nhất. Trong một máy tính thông dụng, các thành phần thường bị nghẽn cổ chai là card đồ họa, bộ xử lý và ổ cứng.

Deadlock Khóa chết (deadlock) là trạng thái xảy ra trong môi trường đa nhiệm (muti-threading) khi hai hoặc nhiều tiến trình đi vào vòng lặp chờ tài nguyên mãi mãi.

Real-time Thời gian thực (real-time) là thuật ngữ khoa học máy tính cho các hệ thống phần cứng và phần mềm chịu "ràng buộc thời gian thực". Nói một cách dễ hiểu, khi phát sinh yêu cầu tới một hệ thống thời gian thực, bạn sẽ nhận được phản hồi ngay lập tức.

Artificial Intelligence (AI) Trong khoa học máy tính, trí tuệ nhân tạo (AI) là trí thông minh được thể hiện bằng máy móc, trái ngược với trí thông minh tự nhiên con người thể hiện. Thông thường, thuật ngữ "trí tuệ nhân tạo" thường được sử dụng để mô tả các máy móc (hoặc máy tính) bắt chước các chức năng "nhận thức" mà con người liên kết với tâm trí, như "học tập" và "giải quyết vấn đề".

Machine Learning (ML) Học máy (ML) là một lĩnh vực của trí tuệ nhân tạo liên quan đến nghiên cứu và xây dựng kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Ví dụ, các máy có thể "học" cách phân loại thư điện tử xem có phải thư rác (spam) hay không và tự động xếp thư vào mục tương ứng. Học máy rất gần với suy diễn thống kê (statistical inference) tuy có khác nhau về thuật ngữ.

Data mining Khai phá dữ liệu (data mining) là quá trình tính toán để trích xuất các dữ liệu có thể sử dụng từ một tập hợp lớn hơn của bất kỳ dữ liệu thô nào. Khai phá dữ liệu được ứng dụng trong nhiều lĩnh vực, như khoa học và nghiên cứu.

Deep learning Học sâu (deep learning) là một tập con của học máy, trong đó sử dụng các mạng thần kinh nhân tạo (artificial neural networks) lấy cảm hứng từ bộ não con người, học từ một lượng lớn dữ liệu. Tương tự như cách chúng ta học hỏi từ kinh nghiệm, thuật toán học sâu sẽ thực hiện một nhiệm vụ nhiều lần, mỗi lần điều chỉnh một chút để cải thiện kết quả.

Big Data Dữ liệu lớn (big data) là việc xử lý một tập hợp dữ liệu rất lớn và phức tạp mà các ứng dụng xử lý dữ liệu truyền thống không xử lý được. Dữ liệu lớn bao gồm các thách thức như phân tích, thu thập, giám sát dữ liệu, tìm kiếm, chia sẻ, lưu trữ, truyền nhận, trực quan, truy vấn và tính riêng tư.

Minimum viable product (MVP) Một sản phẩm mới được phát triển với các tính năng cơ bản tối thiểu nhất đủ đáp ứng những người dùng đầu tiên. Bộ tính năng hoàn chỉnh cuối cùng chỉ được thiết kế và phát triển sau khi xem xét phản hồi từ người dùng ban đầu của sản phẩm.

Computer vision Thị giác máy tính (Computer vision) là lĩnh vực bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh và dữ liệu đa chiều từ thế giới thực để cho ra các thông tin số hoặc biểu tượng.

Blockchain Chuỗi khối (Blockchain) là cơ sở dữ liệu phân cấp lưu trữ thông tin trong các khối thông tin được liên kết với nhau bằng mã hóa và mở rộng theo thời gian. Mỗi khối đều chứa thông tin về thời gian khởi tạo và được liên kết tới khối trước đó, kèm một mã thời gian và dữ liệu giao dịch. Blockchain được thiết kế để chống lại việc thay đổi của dữ liệu: Một khi dữ liệu đã được mạng lưới chấp nhận sẽ không có cách nào thay đổi nó.

Software Requirement Specification (SRS) Đặc tả yêu cầu phần mềm (SRS) là mô tả về một hệ thống phần mềm sẽ được phát triển. Nó đưa ra các yêu cầu chức năng và phi chức năng, ngoài ra có thể bao gồm cả mô tả về cách thức người sử dụng tương tác với hệ thống trong các trường hợp (use cases) mà phần mềm phải cung cấp.

Test Chạy thử chương trình (để tìm ra bug)

Black-box testing Kiểm thử hộp đen (black-box testing) là phương pháp kiểm thử phần mềm, trong đó người kiểm thử coi phần mềm giống một "hộp đen". Với phương pháp này, người kiểm thử chỉ quan tâm tới kết quả của tính năng, không quan tâm tới phần mềm có cấu trúc, thiết kế, cách triển khai như thế nào để tạo ra kết quả đó.

White-box testing Kiểm thử hộp trắng (white-box testing) là phương pháp kiểm thử phần mềm, trong đó người kiểm thử phải biết rõ kiến trúc, mã nguồn bên trong của phần mềm. Người thực hiện white-box testing cần có kiến thức về lập trình.

Scrum Một trong những phương pháp phát triển phần mềm nhanh (agile) phổ biến nhất.

Agile Phát triển phần mềm linh hoạt (agile) là thuật ngữ chỉ các phương thức thực hiện các dự án công nghệ phần mềm, trong đó khuyến khích sự thay đổi khi phát triển dự án và đưa sản phẩm đến tay người dùng sao cho nhanh nhất. Khách hàng được gắn kết với quy trình phát triển phần mềm khi họ liên tục thực hiện các chu trình dùng thử và phản hồi.

Waterfall "Thác nước" (waterfall model) là mô hình của quy trình phát triển phần mềm truyền thống. Trong đó, quy trình phát triển giống như một dòng chảy, với các pha được thực hiện theo trật tự nghiêm ngặt, không có sự quay lui hay nhảy vượt pha.

Microservice Là một kiến trúc phần mềm trong đó ứng dụng lớn (ứng dụng nguyên khối) được chia ra thành các dịch vụ nhỏ với chức năng chuyên biệt có kết nối với nhau.

Docker Một dự án mã nguồn mở giúp tự động triển khai các ứng dụng Linux và Windows vào trong các container ảo hóa.

Milestone Các cột mốc (milestone) là các điểm cụ thể dọc theo dòng thời gian của dự án nhằm mục đích theo dõi sự tiến triển trên đường đi và đảm bảo các thành quả quan trọng đạt được theo thời gian.

Work breakdown structure (WBS) Phương pháp giúp phân tích và cấu trúc công việc. Trong đó, các phần việc lớn được phân chia thành các thành phần nhỏ hơn giúp việc quản lý trở nên hiệu quả, nhất là đối với các dự án lớn và phức tạp.

Malware Ghép lại từ 2 chữ malicious và software, có nghĩa: phần mềm độc hại. Nó được các tin tặc hay hacker tạo ra nhằm gây hại cho các máy tính.

Backdoor Là "cửa hậu" hay lối vào phía sau. Trong một hệ thống máy tính, "cửa hậu" là phương pháp vượt qua thủ tục chứng thực người dùng thông thường hoặc để giữ đường truy nhập từ xa tới một máy tính, trong khi cố gắng không bị phát hiện bởi việc giám sát thông thường.

Ransomware Mã độc tống tiền (ransomware) là phần mềm độc hại có mục đích tống tiền người dùng bằng cách xâm nhập vào máy tính và thao túng dữ liệu của nạn nhân.

Penetration Testing (Pentest/ Ethical Hacking) Thâm nhập thử nghiệm (Penetration Testing), hay còn gọi là pentest hay ethical hacking chỉ hành động mô phỏng cuộc tấn công vào hệ thống máy tính nhằm đánh giá mức độ an toàn cũng như tìm ra các lỗ hổng bảo mật có thể khai thác. Người thực hiện thâm nhập thử nghiệm phải được cho phép của người sở hữu hệ thống.

DANH SÁCH CÔNG TY CÔNG NGHỆ TIÊU BIỂU

Tek Experts

Tek Experts là công ty đa quốc gia chuyên về dịch vụ hỗ trợ CNTT và Professional Services toàn cầu có văn phòng tại 7 nước trên thế giới: Mỹ, Bulgaria, Costa Rica, Nigeria, Trung Quốc, Việt Nam và Rwanda. Có mặt tại Việt Nam từ năm 2013, sau 7 năm phát triển, công ty đã sở hữu 2 văn phòng hơn 4000m2 tại các tòa nhà hạng A ở trung tâm Hà Nội với hơn 700 nhân viên. Tek Experts đã và đang trở thành đơn vị cung cấp dịch vụ đầu ngành về hỗ trợ CNTT.

Tek Experts luôn chú trọng xây dựng môi trường làm việc tốt nhất để tạo điều kiện cho đội ngũ nhân lực phát triển về chuyên môn và trau dồi kỹ năng mềm. Nhân viên có cơ hội làm việc trong môi trường quốc tế, văn phòng làm việc được đầu tư trang thiết bị hiện đại với khu Sky Garden độc đáo, phòng trải nghiệm game cho nhân viên với đầy đủ thiết bị Xbox, Mixed Reality.

Là văn phòng có quy mô lớn thứ 3 trên thế giới, nhân viên tại Tek Experts Việt Nam luôn tự hào về môi trường làm việc thân thiện, sáng tạo với rất nhiều hoạt động sau giờ làm việc như: sinh nhật công ty, ngày hội gia đình, đổi rác điện tử lấy cây xanh, câu lạc bộ chia sẻ kiến thức, Green Forest, Morning Village Fair, ngày nam giới Men's Day, giải bóng đá Tek League, lễ Tạ ơn, Giáng sinh,...

VNG Corporation

Thành lập từ năm 2004, VNG là Tập đoàn Công nghệ và Internet hàng đầu Việt Nam với hệ sinh thái sản phẩm đa dạng, phục vụ nhu cầu của hơn 100 triệu khách hàng ở trong nước và quốc tế. Hiện nay, VNG đang tập trung phát triển 4 nhóm sản phẩm chính: Games, Nền tảng kết nối (Zalo, Zing,...), Thanh toán điện tử (ZaloPay), Điện toán đám mây (VNG CLOUD).

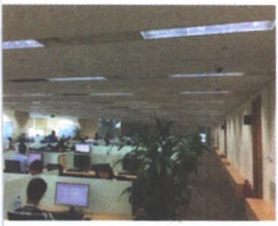
VNG cũng là doanh nghiệp đầu tiên tại Việt Nam được định giá trên 1 tỷ USD theo đánh giá World Startup Report và Google – Temasek Report, nằm trong Top 14 Kỳ lân công nghệ của Đông Nam Á năm 2019, theo báo cáo của Bain & Company.

Năm 2019, VNG lọt Top 50 thương hiệu hàng đầu Việt Nam, nhận giải thưởng "Nơi làm việc tốt nhất Châu Á" do HR Asia, tạp chí uy tín hàng đầu về nhân sự tại Châu Á tổ chức và bình chọn. Cũng trong năm 2019, VNG khánh thành trụ sở mới VNG Campus tại Quận 7, TP.HCM với tổng diện tích 52.440m2, áp dụng theo mô hình Campus hiện đại của thế giới: khu văn phòng tích hợp thư giãn, giải trí, thể thao.

Tổng Công ty Truyền thông VNPT - Media

Tổng công ty Truyền thông (Tên viết tắt: VNPT - Media) hoạt động trong lĩnh vực nghiên cứu phát triển, kinh doanh dịch vụ Truyền hình, dịch vụ Truyền thông đa phương tiện, dịch vụ Giá trị gia tăng và Công nghệ thông tin với 4 công ty trực thuộc: Công ty Phát triển Dịch vụ Truyền hình, Công ty Phát triển Dịch vụ Truyền thông; Công ty Phát triển Dịch vụ Giá trị gia tăng và Công ty Phát triển Phần mềm, 01 Trung tâm Dịch vụ Tài chính số cùng các Ban chức năng và các chi nhánh tại miền Trung và miền Nam.

VNPT - Media đặt mục tiêu xây dựng một hệ sinh thái tích hợp trọn gói các dịch vụ đa phương tiện trên nền tảng công nghệ và Internet lớn nhất Việt Nam, từ đó mang sản phẩm - dịch vụ đến thị trường quốc tế.



Viettel Telecom

Khối CNTT trực thuộc Tổng công ty (TCT) Viễn Thông Viettel (Viettel Telecom), chi nhánh Tập đoàn Công nghiệp Viễn thông Quân đội là đơn vị:

- Ứng dụng các kết quả phân tích dữ liệu (Data Analytics) dựa trên nền tảng dữ liệu lớn (Big Data) vào các bài toán kinh doanh nhằm tăng doanh thu và tạo ra các nguồn doanh thu mới.
- Trực tiếp thiết kế, phát triển các ứng dụng phần mềm phục vụ hoạt động điều hành, sản xuất kinh doanh toàn diện của TCT trên 63 tỉnh/ thành phố và 10 thị trường nước ngoài.
- Làm chủ và ứng dụng các công nghệ hàng đầu (AI, Machine Learning, Deep Learning, Big Data, Cloud, Microservices,..) vào các dự án, sản phẩm phần mềm nhằm thay đổi trải nghiệm của khách hàng với các sản phẩm dịch vụ, đổi mới cách điều hành sản xuất kinh doanh.
- Tiên phong trong việc ứng dụng các công cụ và quy trình phát triển phần mềm mới nhất nhằm tự động hóa và tối ưu quá trình phát triển phần mềm (DevOps, Agile,...).

Công ty Cổ phần Giải pháp Thanh Toán Việt Nam (VNPAY)

Thành lập tháng 03/2007, VNPAY là đơn vị cung cấp giải pháp thanh toán điện tử hàng đầu Việt Nam. Với nhiều năm kinh nghiệm trong lĩnh vực tài chính - ngân hàng, VNPAY hiện cung cấp dịch vụ cho hơn 40 ngân hàng, 05 công ty viễn thông và hơn 50.000 doanh nghiệp, sở hữu mạng lưới thanh toán QR code lớn nhất Việt Nam với hơn 70.000 điểm chấp nhận thanh toán.

VNPAY được vinh danh là Fintech tiêu biểu Việt Nam năm 2019, top 10 danh hiệu Sao Khuê 2019 với nhiều đột phá, sáng tạo công nghệ và tiên phong về các giải pháp thanh toán điện tử.

Các sản phẩm/dịch vụ tiêu biểu: Mobile Banking; Thanh toán VNPAY-QR; VNPAY POS; Website TMĐT Vban.vn; Thanh toán hóa đơn VnPayBill; Đặt vé máy bay VnTicket; Nạp tiền điện thoại VnTopup; SMS Banking,...

Công ty Cổ phần TOPCV Việt Nam

TopCV là Nền tảng Tuyển dụng Nhân sự hàng đầu Việt Nam, với sứ mệnh mang đến những cơ hội nghề nghiệp tốt nhất cho người Việt. Với 2 sản phẩm mũi nhọn là công cụ tạo CV online và kênh việc làm chất lượng cao, TopCV đã phục vụ xuất sắc cho 3.000.000+ người tìm việc trên topcv.vn, 120.000+ nhà tuyển dụng trên tuyendung.topcv.vn Trung bình mỗi tháng, TopCV giúp kết nối hơn 150.000 ứng viên với doanh nghiệp, trong đó có Viettel, Vingroup, FPT,...

Luôn chủ động và sáng tạo, lấy công nghệ làm cốt lõi để phát triển, TopCV nỗ lực từng ngày để cải thiện chất lượng nhân sự, thay đổi thị trường Tuyển dụng - Việc làm tại Việt Nam ngày một tốt hơn.

Công ty Cổ phần Base Enterprise

Được thành lập vào tháng 8/2016, Base là một trong những công ty công nghệ đi đầu trong lĩnh vực xây dựng nền tảng quản trị doanh nghiệp Software-as-a-Service (SaaS) tại Việt Nam.

Ứng dụng trên nền tảng Base được thiết kế chuyên sâu và tối ưu cho từng tác vụ cụ thể trong doanh nghiệp: từ quản lý công việc, thông tin, nhân sự, cho tới quản trị khách hàng và kinh doanh,... Tất cả được tích hợp với nhau trên một nền tảng chung. Base cũng có thể kết nối giải pháp của nhiều đơn vị cung cấp khác nhau, tạo nên bộ công cụ mạnh mẽ và toàn diện.

Hiện nay Base đã và đang phục vụ cho hơn 5000 doanh nghiệp tại Việt Nam.

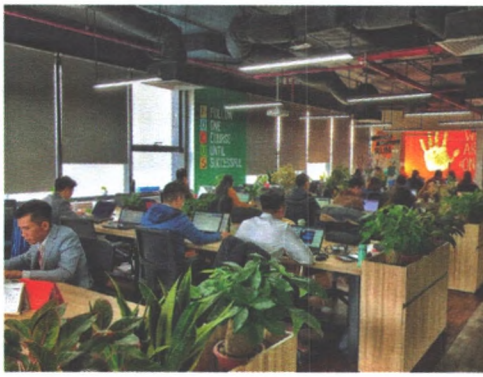
Công ty Cổ phần Phần mềm citigo

Được thành lập năm 2010 với khát khao đưa đến những giải pháp công nghệ tiết kiệm chi phí và nâng cao hiệu quả kinh doanh của khách hàng, Citigo không ngừng nỗ lực nghiên cứu, sáng tạo và phát triển để ngày một hoàn thiện và vươn xa hơn.

Citigo tự hào về những sản phẩm và dịch vụ (tư vấn, phát triển các sản phẩm, giải pháp về phần mềm và cung cấp đội ngũ các lập trình viên chuyên nghiệp) đang ngày càng trở nên phổ biến hơn với khách hàng, công ty đạt tốc độ tăng trưởng trung bình lên tới 200%/năm.

Công ty Cổ phần Workway (1Office)

Xây dựng nền tảng All In One: Tổng hợp tất cả các nghiệp vụ quản trị doanh nghiệp (như làm việc và giao tiếp nội bộ, quản lý nhân sự, marketing, bán hàng, chăm sóc khách hàng) trên MỘT phần mềm duy nhất giúp các CEO quản trị doanh nghiệp hiệu quả. Sau 5 năm hình thành và phát triển, sản phẩm đã được hơn 2000 doanh nghiệp trong nhiều lĩnh vực chọn lựa (tài chính, bán lẻ, du lịch, thương mại điện tử,...). Đồng thời, 1Office đã chạm mốc 100.000 người dùng, lọt top 3 nền tảng Quản trị Việt Nam được tìm kiếm nhiều nhất trên Google.



NextTech Group

Được thành lập từ năm 2001, NextTech Group là một hệ sinh thái khởi nghiệp với gần 20 dịch vụ, hoạt đông chủ yếu trong 4 lĩnh vực: E-commerce, Fintech, E-logistic & EduTech. NextTech là Tập đoàn về Công nghệ thông tin đang sở hữu những sản phẩm Công nghệ có tên tuổi trên thị trường như: NganLuong.vn; mPOS.vn; Vimo.vn; Boxme.asia; Học viện Teky và Ứng dụng Gọi xe Công nghệ Fastgo.

Với chặng đường gần 20 năm phát triển, NextTech hiện có gần 2.000 nhân viên hiện diện trên 8 quốc gia tại thị trường Đông Nam Á và Trung Quốc. Sứ mệnh của NextTech luôn mong muốn trở thành "đàn cá hổ" trong bể đại dương giúp tiện lợi hóa và điện tử hóa cuộc sống con người.

Công ty Cổ phần Công nghệ Sapo

Sapo (sapo.vn) là doanh nghiệp có nền tảng quản lý và bán hàng đa kênh được hơn 67.000 khách hàng trên toàn quốc tin tưởng và sử dụng. Sapo cung cấp giải pháp tổng thể từ online đến offline trong 5 lĩnh vực: Sapo POS - Phần mềm quản lý bán hàng; Sapo Web - Giải pháp thiết kế website bán hàng chuyên nghiệp; Sapo FNB - Phần mềm quản lý nhà hàng, quán cafe; Sapo Omnichannel - Nền tảng quản lý và bán hàng đa kênh từ Online đến Offline; Sapo Enterprise - Giải pháp thương mại điện tử cho doanh nghiệp lớn; Sapo Go - Giải pháp quán lý bán hàng online.

Hiện Sapo có trên 600 nhân sự với trình độ chuyên môn cao, môi trường làm việc vô cùng trẻ trung và năng động.

Tập đoàn Công nghệ Haravan

Haravan được thành lập vào năm 2014, là công ty chuyên cung cấp giải pháp vượt trội cho việc bán hàng, marketing và quản lý vận hành trong kinh doanh từ online đến offline.

Với tốc độ phát triển nhanh nhất trong lĩnh vực thương mại điện tử, Haravan được Facebook chọn là đối tác Việt Nam duy nhất trong việc phát triển kinh doanh nền tảng Facebook Messenger và là 1 trong 2 công ty khởi nghiệp tại Việt Nam được Google lựa chọn tham gia chương trình Launchpad Accelerator (Bệ Phóng Tài Năng).

Tự tin là người bạn đồng hành cùng các thương hiệu bậc nhất như: AEON, Vinamilk, Thiên Long, Couple TX, Kiềm Nghĩa, Juno, The Coffee House,... Haravan không ngừng tìm kiếm những nhân tài đầy nhiệt huyết, đam mê để cùng mang đến những trải nghiệm tuyệt vời cho khách hàng và thực hiện sứ mệnh "MAKE COMMERCE BETTER".

Công ty TNHH Cube System Vietnam

Là công ty 100% vốn đầu tư Nhật Bản hoạt động trong lĩnh vực công nghệ thông tin/phần mềm, Cube System Vietnam cung cấp dịch vụ phát triển phần mềm và giải pháp công nghệ cho các khách hàng hoạt động trong nhiều lĩnh vực từ sản xuất, logistics tới giáo dục, tài chính, may mặc,... tại thị trường Việt Nam và Nhật Bản. Hiện tại, công ty đang phát triển các hệ thống chính như Giải pháp Mobile xúc tiến bán hàng, Hệ thống nhân sự, Hệ thống quản lý trang thiết bị, Hệ thống quản lý hàng hóa,...

Kế thừa DNA của công ty mẹ - Cube System INC, với phương châm "Khách hàng là số một", "Chủ nghĩa trọng điểm", và "Toàn nhân viên tham gia kinh doanh", công ty nuôi dưỡng những kỹ sư Việt Nam ưu tú - "Kỹ sư hệ thống hướng đến khách hàng" nhằm cống hiến cho sự nghiệp phát triển của 2 quốc gia Việt - Nhật. Đi cùng phương châm đó, Cube System hằng năm tổ chức chương trình trao học bổng cho sinh viên của trường Đại học TPHCM, nhằm khuyến khích, hỗ trợ phát triển các nhân tài.

CUBE SYSTEM
VIETNAM CO.,LTD.

Công ty Cổ phần OWS Việt Nam

Hơn 6 năm phát triển, OWS Việt Nam đã thành công trong việc áp dụng công nghệ Streaming và Real-time vào những sản phẩm do chính công ty tạo ra: Daotaonoibo Platform, CloudClass, CloudMERO,... và nhận được sự tin tưởng của khách hàng ở hầu hết các lĩnh vực. Ngoài ra, OWS còn cung cấp các giải pháp công nghệ thông tin cho các doanh nghiệp với nguồn nhân lực chất lượng cao, thuyết phục các thị trường, đối tác khó tính trong việc đánh giá chất lượng sản phẩm như Nhật Bản.

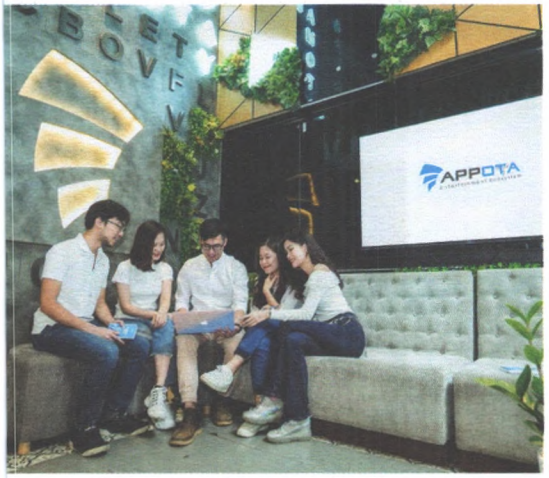
Công ty Cổ phần AZDIGI

AZDIGI chuyên cung cấp các giải pháp lưu trữ website (Web Hosting) chuyên nghiệp trên nền tảng Linux, tối ưu cho những mã nguồn mở như WordPress, Joomla, Drupal, Moodle và các website sử dụng PHP & MySQL khác. Công ty sử dụng những công nghệ tiên tiến nhất tích hợp trên các dịch vụ như LiteSpeed Webserver, CloudLinux, Imunify360, Redis Cache cùng các giải pháp liên quan: Đăng ký tên miền, Máy chủ ảo riêng (VPS), Chứng chỉ SSL,... Hoạt động từ năm 2016, đến nay, AZDIGI đã và đang phục vụ 3000 khách hàng doanh nghiệp, 12000 khách hàng cá nhân và đồng hành cùng một số đối tác lớn như: FPT Telecom, Viettel,...

Công ty Cổ phần Công nghệ Getfly Việt Nam

Getfly CRM mang đến giải pháp phần mềm quản lý và chăm sóc khách hàng toàn diện, chuyên nghiệp cho SMEs Việt, giúp tối ưu việc chăm sóc khách hàng, hỗ trợ quản lý tương tác giữa các phòng ban Marketing - Sales - Chăm sóc khách hàng trên một nền tảng. Ứng dụng giúp thống nhất quy trình, tiết kiệm thời gian, tối ưu nguồn lực, tăng hiệu suất làm việc và tự động hóa doanh nghiệp.

Đến nay, công ty đã triển khai cho 3000+ doanh nghiệp với 150+ ngành nghề.



Công ty Cổ phần Appota

Appota là đơn vị tiên phong cung cấp các giải pháp công nghệ và nội dung giải trí số tại thị trường Việt Nam. Công ty hiện sở hữu hệ sinh thái gồm 4 công ty thành viên: Gamota (Công ty phát hành game), Adsota (Công ty dịch vụ quảng cáo), AppotaPay (Công ty giải pháp thanh toán) và Kdata (Công ty giải pháp điện toán đám mây) với hơn 55 triệu người dùng trên Internet. Appota đồng thời là đối tác chính thức duy nhất phát triển mạng lưới Facebook Gaming Creator tại Việt Nam. Các lĩnh vực thế mạnh của công ty bao gồm: Sản xuất và phát hành nội dung giải trí trực tuyến, thể thao điện tử, quảng cáo, gaming creators, thanh toán trực tuyến, điện toán đám mây, IoT và giải pháp quản lý doanh nghiệp.

Công ty đã giành được nhiều thành tựu bao gồm giải thưởng "Startup toàn cầu đột phá nhất" bởi The Founder Institute vào năm 2012 và danh hiệu "Công ty tiềm năng nhất Việt Nam" năm 2013.

"Không sao chép nội dung khi chưa được sự đồng ý bằng văn bản"

NHÀ XUẤT BẢN THẾ GIỚI

Trụ sở chính:
Số 46. Trần Hưng Đạo, Hoàn Kiếm, Hà Nội
Tel: 0084.24.38253841

Chi nhánh:
Số 7. Nguyễn Thị Minh Khai, Quận I, TP. Hồ Chí Minh
Tel: 0084.28.38220102
Email: thegioi@thegioipublishers.vn
marketing@thegioipublishers.vn
Website: www.thegioipublishers.vn

**NGƯỜI TRONG MUÔN NGHỀ:
NGÀNH IT CÓ GÌ?**
(Tái bản)

Chịu trách nhiệm xuất bản
GIÁM ĐỐC - TỔNG BIÊN TẬP
PHẠM TRẦN LONG

Biên tập:
NGUYỄN TRUNG DŨNG

Vẽ bìa:
KHANG

Trình bày:
DUNG TIGONKHONGLO

Sửa bản in:
NGA LEVI, DŨNG EZ

LIÊN KẾT XUẤT BẢN
Công ty Cổ phần AHORA
Địa chỉ: Số 6 ngõ 186 đường Bưởi, Ba Đình, TP. Hà Nội

In 5.000 bản, khổ 16 cm x 24 cm tại Công ty Cổ phần In Viễn Đông
Địa chỉ: Km 19+400 Giai Phạm, huyện Yên Mỹ, tỉnh Hưng Yên

Số xác nhận ĐKXB: 3606-2021/CXBIPH/07-228/ThG.
Quyết định xuất bản số: 998/QĐ-ThG cấp ngày 19 tháng 10 năm 2021.
In xong và nộp lưu chiểu năm 2021. Mã ISBN: 978-604-345-064-4