13. Product Lifecycle

+ Coding Interviews

Logistics ⚙️

- Remaining assignments:
  - Due Dec 2
    - Roadmap
    - Specs doc
    - Peer feedback
  - Due Dec 9
    - Exit survey featuring another HackerRank challenge, should be <1 hour
- Final lecture will be a pizza party unless people yell at me about wanting something different?
  - Anyone have topic requests? I’ve got industry success tips but that’s not gonna take more than an hour.
- Homework 10 is due tonight! I’ve been trying to follow up with people who are falling behind, but PLEASE reach out to me if you’re having trouble.

Agenda 📅

1. Recap
2. How Software Gets Built
3. Technical Interviews

Poll: Specifications are

1. Written requirements that tell engineers why they’re building what they’re building
2. A written agreement about what exactly will be done as part of a project
3. A set of required technologies for a project (like a database, a language)

Poll: Specifications are

1. Written requirements that tell engineers why they’re building what they’re building
2. A written agreement about what exactly will be done as part of a project
3. A set of required technologies for a project (like a database, a language)

Recap: Functional/Non-functional Requirements

Name **one functional requirement** and one **non-functional requirement** of TikTok!

Recap: Critique a user story?

Story: User is able to view videos posted by people they're following and share them with their friends

Acceptance Criteria:

- Videos appear in a grid view on main page
- User is able to watch the videos that appear on the page
- Videos have a share button that r

Recap: Changing DB schemas?

Which of the following do you NOT have to do if you want to add or remove a column from a table in your SQLite database, using the tech stack from class?

1) Add the column as an instance variable of the model class in Python
2) Restart your app
3) Manually delete the table
4) Rename the table

Where do software products come from?

Or: you’ve got an idea for a feature or product, but now what?

Where do software products come from?

Or: you’ve got an idea for a feature or product, but now what?

Generally speaking, there are four common stages for software products:

1) Proof of Concept
2) Minimum Viable Product (MVP)
3) Product-Market Fit (PMF)
4) Go to Market (GTM)

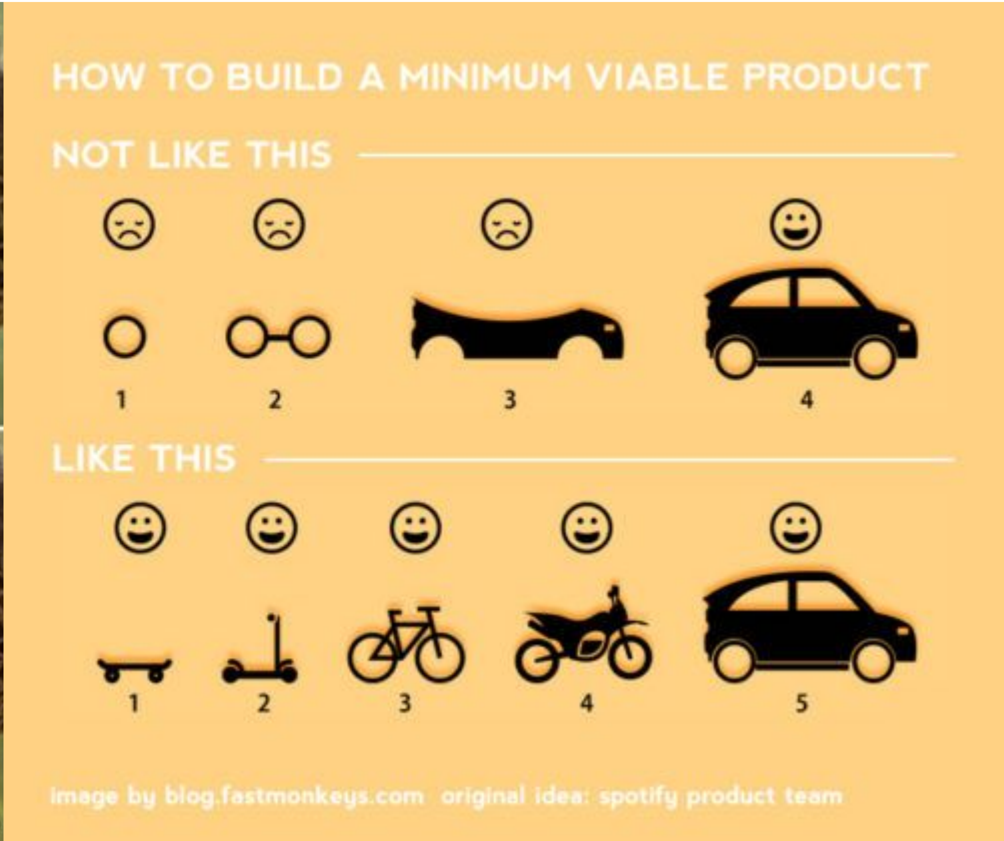
Heads up – this isn’t really how you’ll be thinking about your group project work, because you don’t have the same kind of market uncertainty that actual startups face.

Phase 0: Proof of Concept

- You need to show that your product is possible to build
- This can be an EXTREMELY rough first draft - a lot of times it's just a script that does the hard part(s) of your project
- If you were building Netflix, your proof of concept could just be showing that video streaming is possible

I don't need to write clean code.
This is just the proof of concept.
Once I know for sure this solution will work, I will go back and clean this mess up.
Well it works better than I anticipated and my boss wants it deployed ASAP. Maybe I'll rewrite it once it's been in prod for a few months.

Phase 1: Sprint to MVP (Minimum Viable Product)


HOW TO BUILD A MINIMUM VIABLE PRODUCT
NOT LIKE THIS
1
2
3
4
LIKE THIS
1
2
3
4
5
image by blog.fastmonkeys.com original idea: spotify product team

Phase 2: Product-Market Fit (PMF)

- This can be hard to define and varies depending on what you're working on
- In general, PMF refers to there being actual demand for your product (i.e. your product fills an actual need and people will pay to use it)
- Continuing the Netflix example: this would be the point where you have a decent number of subscribers who are regularly watching movies.

Phase 3: Go to Market (GTM)

- This part is about growing your user-base and scaling up your ability to serve customers
- For Netflix, this would essentially mean a major advertising push + potentially trying to expand internationally?
- This is also when you're figuring out what price point makes sense for what you're offering

LEMONADE! GET YOUR ICE-COLD LEMONADE! JUST FIVE DOLLARS A GLASS, WHILE IT LASTS!
HOW ARE SALES?
TERRIBLE. I DON'T UNDERSTAND IT.
IT SURE IS COLD OUT.
YEP. SEE, MY LEMONADE HAS "ALL-NATURAL" REFRIGERATION! WANT TO BUY A GLASS?
SORRY. ALL MY SAVINGS ARE IN BONDS.
MAYBE I SHOULD START CHARGING TEN DOLLARS SO I DON'T HAVE TO SELL AS MUCH.
WATTERSON 11-30
© 1990 Watterson/Distributed by Andrews McMeel Syndication

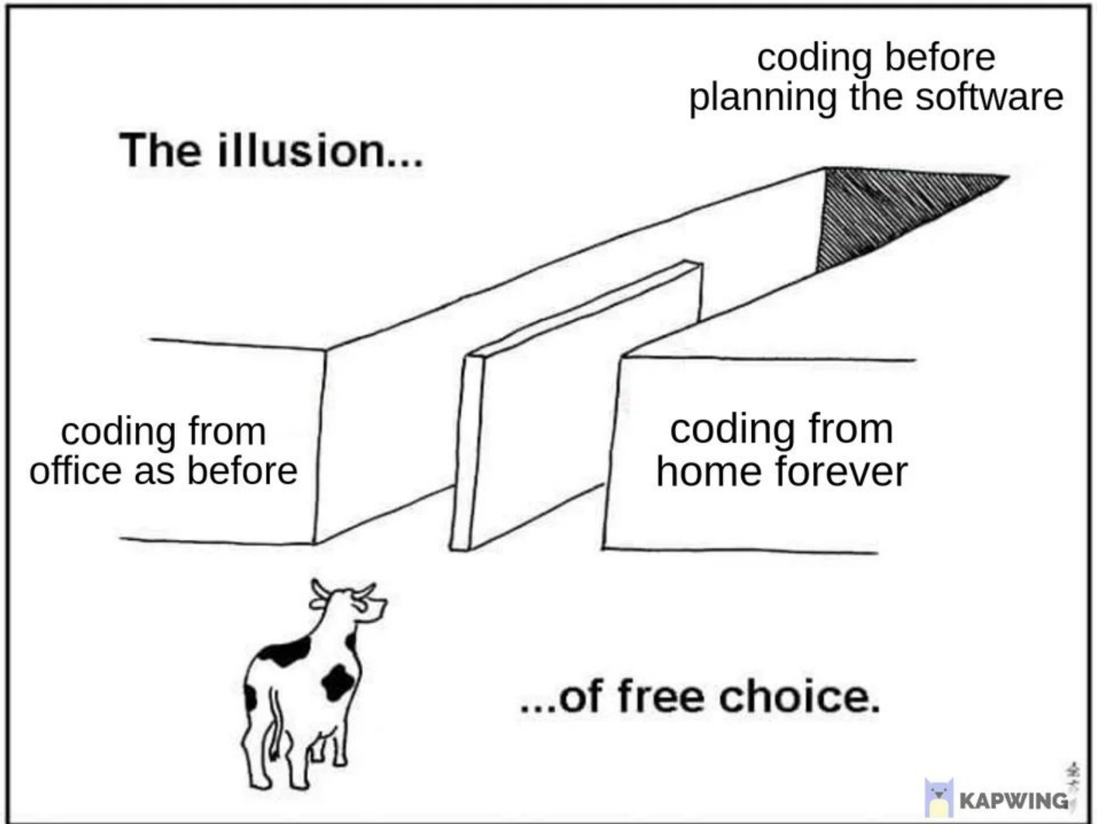
Case Study: Network Planning Products

- Located in Facebook Connectivity, which had a broad mandate to increase internet access and speed worldwide
- Started off serving internal customers but realized there might be an external need
- Built a tool from scratch!
  - Proof of concept was the core network planning algorithms
  - MVP was a simple portal partners could use to kick off those algorithms
  - We got stuck at product market fit and eventually dissolved. Sad!

Case Study: PERFAL

- Including this example to show that when you build internal tools, life can be a little different
- Built a system for tracking changes in CPU and disk usage, as well as network plan quality
- Customers were other members of my team, so it was super easy to spur adoption
- Stages:
  - Proof of concept was the set of queries that the system used
  - MVP was a Jupyter notebook chaining them all together
  - PMF was determined more or less in advance, but there was a little iteration to get it into a useful state for teammates
  - No GTM – there wasn’t a broader audience

Roadmapping



Roadmapping

- Whatever stage of product growth you’re in, you want to have some sense of where you’re going next.
- Sometimes in early product stages or with few people, there’s less need for a structured plan.
- However, most of the time, teams will want to have their next 4-12 weeks planned out
- This is something you’re going to practice in the group project.

Roadmapping

There are infinitely many ways to lay out this information, but you at least want to cover the following: what the tasks are, how long they'll take, who owns them, and priorities.

Task	Owner	Priority	Est. Level of Effort
Create auth flow	John Martin	P1	12 days
Add animations to UI	John Martin	P3	5 days
Allow user to link bank account	Someone Else	P2	8 days

Priorities

Quick guide to the priority of a feature:

- P0 means it has to be done or else there will not be a team/company/product. Often reserved for critical/emergency situations.
- P1 means it is a must-do.
- P2 means it is important, but doesn’t NEED to get done right now.
- P3 means it is a nice-to-have and probably won’t get done.

This isn’t the only system for prioritization, but these terms will pop up in a lot of conversations.

Tips for estimating time for tasks

- Try to write down what the specific steps you'll have to take to finish the feature are
- Compare with similar features in the past
- If you're new to this, double whatever estimate you come up with (at least)
  - It's really hard to plan for unknown unknowns - for example, no one factors in time they'll have to spend fixing their computer when it freezes and deletes a couple hours of work.
  - In my experience, it's rare for new engineers to OVER-estimate how long something will take.

Break

One other thing - feedback

- You’re going to be expected to give peer feedback at the end of this semester, and throughout next semester as group work really gets going
- “Feedback is a gift” ← a saying I’ve heard at literally every company I’ve worked for
  - (which admittedly is like two of them)
- Good feedback is HARD to give – it’s time-consuming to write, not because it’s hard to write but because it’s hard to really figure out what people could do better or why they’re good at what they do.
- It can also feel scary to give constructive feedback, which leads to communicating it poorly

One other thing - feedback

Takeaways:

- Take the group feedback form seriously! One of the best gifts you can give each other is a really meaningful evaluation of what they’re doing well and what they can do better.
- Share feedback directly if you’re feeling brave!

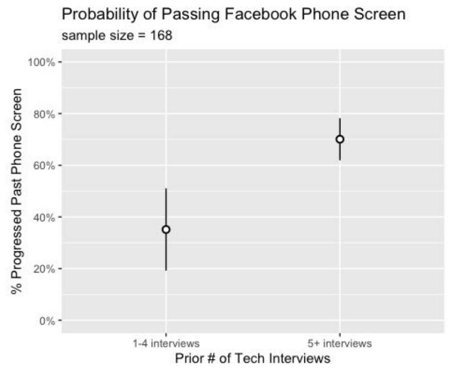
Ok, I want to talk about coding interviews

“I can’t pass a coding interview”

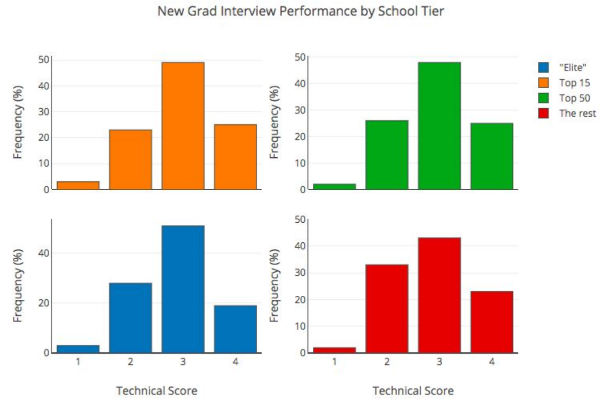
- “I’m not good at coding”
- “I can’t handle the pressure of the live interview”
- “I can’t find time to study”

“I can’t pass a coding interview”

- Not all companies have super rigorous coding interviews, or any coding interview at all.
- Interviewing is **not** an innate talent. It’s an arbitrary game that a bunch of companies have decided is the way to filter candidates, for now.
- Five practice interviews have been shown to dramatically improve your odds.



Practice matters way more than raw technical ability



“I can’t pass a coding interview”

- One other thing we need to talk about is dealing with nerves, which I think is really about dealing with rejection.
- **No one** bats 1.000 on coding interviews - only something like 20% of people are consistent performers at coding interviews. So no matter how good you are, you’ve got to be ready to accept that this is a numbers game.
- Anecdotal: in-interview mindsets from best to worst:
    - “That’s an interesting problem and I wonder what the solution is.”
    - “I am annoyed that I have to do a coding interview”
    - “I’m sleepy”
    - “I’m really afraid to mess up this interview because it is my ONE CHANCE at happiness”

Quick interview study guide:

1) Thoughts → code
    - Mental check: once I know how to solve the problem (i.e. what algorithm to use), can I confidently convert that idea into working code?
    - If not (or if you want to check), try making up practice problems with tasks you already know how to do (find the second smallest element in an array, binary search, etc) OR do problems where you've already looked up a solution.
2) (Extra Credit) Practice as many data structures and algorithms as you can:
    - Tier 1: Hashmaps (Dictionaries), Hashsets (Sets), Binary Search
    - Tier 2: Stacks, Queues, Binary Trees, Linked Lists
    - Tier 3: Graphs, DFS, BFS
    - Tier 4: Dynamic Programming

Later 👋

- Have a great break! See you for the last day of class!
- Please don’t do any work for this class during break
- Let me know if there’s anything you want me to try to cover in half of the last lecture - guest speakers? Topics? I wanna end on a good note :)