Curriculum design

Covering the fundamentals

Creating a cohesive curriculum 1 - Removing Silos

- Subjects should not be siloed, every module should know about every other module.

- E.g.
  - In all modules we should think about the performance implications with reference to DSA.
  - Or we should think about how the frontend might talk to the backend later etc…
  - We're always committing and reviewing code with github + git
  - Summary: all modules should blend with each other

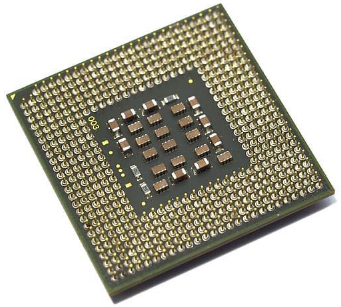
- If there's capacity in terms of well rounded trainers etc, you could complete an E2E project through the whole bootcamp.

Reduce complexity and super specific technologies

- If not absolutely needed by the client, we should eliminate bloated stacks
  - E.g. Prefer using something like (Python + Flask) over (NodeJS, Express, Axios, Javascript etc…) to teach backend server-side programming.

- Builds confidence and removes friction between students are core engineering concepts.

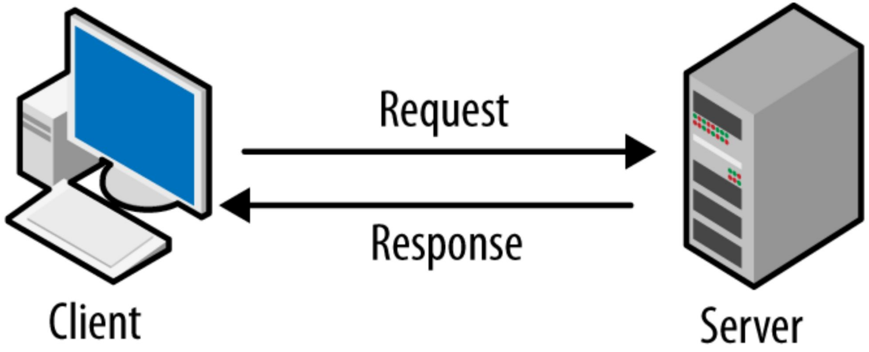
1. Computer science 101



1. Number systems (Intro to binary)
2. How Binary communicates with the bare metal
3. The life cycle of a computer programme and how it’s facilitated main PC components:
   - CPU (Cores, threads, cache transistors etc), GPU, Storage, RAM
4. Interpreted VS compiled and JVM Languages
5. Memory Leaks.
6. Using the above to work out the efficiency of algorithms and built-in Object methods in languages.
7. Generic DSA stuff: Log function, complexity, Trees, graphs etc...

2. Internet 101

1. Fundamentals about architecture design:
    a. Client-server architecture
    b. DNS querying
    c. IP addresses
    d. Ports
    e. Internet protocols
        i. SSH ⟶ If I had more time…
        ii. Leads perfectly to git
        iii. Explain architecture of GH
        iv. How we can use git with it
        v. Code reviews
    f. Etc, see Internet 101 slides...



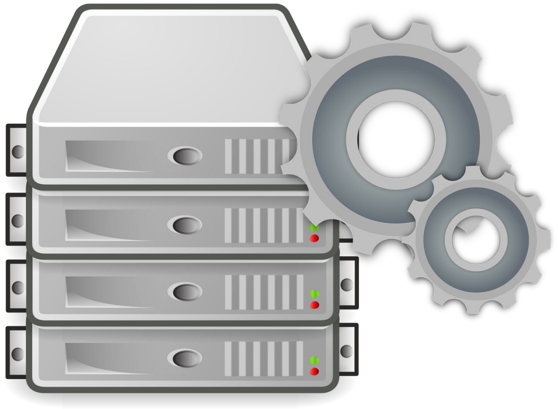
3. Web technologies Vanilla

1. Building the client side:
   a. Personal portfolio
   b. And then an interactive dynamic webpage that loads info from server.
   c. Learn how/when to use smaller 3P libraries and the motivations behind them e.g. bootstrap:
      i. SEO, serving of static files etc
   d. The lifecycle of the browser website (How the browser rendering engine works) ⟶ Need a strong understanding of the fundamentals.
   e. Client vs Server Side rendering ⟶ Intro to React…
   f. JS looks good, might be worth culling some of the more JS specific topics to make more time for some of the above?

4. Intro to web frameworks React JS

1. The motivations behind creation of React.
2. With differences does react make to the lifecycle of a website (The critical rendering path)
3. Using knowledge of the CRP to create performant websites
   a. Above the fold rendering
   b. Deferring scripts
   c. Minimising, bundling files etc
   d. Avoid render blocking code if it can be helped etc…
   e. 40% of users won't wait for a page that takes >= 3s to load.
4. Comparison to other frameworks (Vue, Angular)
   a. Make the call, when to use what? Languages and frameworks are just tools, no need for platform loyalty…
5. State management

5. Building backend systems



- The current Java course looks like has everything needed.

- Like all other modules, we just keep relating back to how the memory works under the hood.

6. System design - Storage Solutions (SQL vs NOSQL vs Caching etc)

1. This is mainly about SQL, we start with the motivations behind SQL for Data storage / when it’s best used.
2. Compare to data analysis using a programming language and a txt file.
3. Maybe the same for NoSQL?
4. Other storage types and possible architectures e.g. Redis caching are briefly looked at.
5. SQL basics e.g. Joins etc…

7. Security

1. Beware the React trap, with all the libs you add for functionality you add vulnerability.
2. Authentication, sessions, cookies JWT Tokens.
3. SQL injection exercises

8. Deployment, containerisation and CI/CD?

1. I don't have much experience here but a few days can be spent on the steps to deploying a finished products and different ways it can be deployed.
   a. Canarying etc
2. These are probably out of scope anyway
3. Have I left out any modules?

9. Developments in tech

1. Supplemental materials and/or high level lectures on the basics of the theory of technologies like:
   a. AI/ML models
   b. blockchain
   c. distributed systems
   d. game dev
   e. AR/VR etc…
   f. Learning how to:
      i. Consume tech news
      ii. Hold conversations
      iii. Be first movers in emerging technologies