Individual insights report – Leon

Collaborative software development

Developing software together in a team efficiently is hard. This already manifests itself when you start thinking about how to exactly define a Git / Github workflow such that the amount of parallelization conflicts is minimal. In retrospect, we should've been more strict about this and actually branch out every time a new feature is implemented or something is tried.

Dividing responsibilities

Next time when I work on a coding project, I will make sure that early on, we get a clear picture of the functionalities the product or MVP is supposed to have and that the entire team agrees upon it. Then, splitting up the tasks into disjoint sets with optimally clearly defined interfaces in between will be easier and less conflicts emerge from people trying to do the same thing but slightly differently.

Continuous integration / development pipelines

Github Actions is great when a product is continuously developed further to just push the changes from the production branch automatically to a server. I will definitely consider this for future projects.

Frontend architecture

As I have never worked on a fullstack pipeline before, I learned a ton about frontend development using Javascript but also html to actually design and build a website from the ground up. Essentially how to build a user interface from div blocks, how to incorporate functionality on button clicks or mouse hovering and also how to communicate with a backend via HTTP protocol. Also learned a ton about how to coordinate file structure and names between frontend and backend by simply exchanging JSON files.

Backend architecture

Similarly to frontend engineering, I have only worked on ML research projects so far and had no idea how to actually integrate models into a backend server that can actually send results to a frontend. Hence, using Python Flask as API for the server endpoints was also new territory. Moreover, working with the OpenAI API or other APIs as well as Llama Index for RAG was a really interesting learning experience and in hindsight taught me a lot about how to integrate models into an actual product as well as what LLMs and RAG is capable of and in which regards it is still limited.

Hosting a website

Finally, after most of the architecture and functionality is implemented, each software product has to be made accessible to customers either through an app or a website. In order to so, I had to dive into AWS to get an EC2 instance as backend server as well as a S3 bucket as frontend server running. I gained a lot of insights on how anything in general on the internet is hosted and how to configure AWS instances such that they can be accessed from the right endpoints. Moreover, since Magnus bought a website name on a DNS provider, I got to learn about routing settings such that the DNS actually points to the AWS S3 bucket and properly serves the frontend once called my a user.

Overall

In summary I really feel like a gained a lot of knowledge on the entire full-stack development process of building and releasing a product as well as using LLMs or potentially other models for interesting applications. Additionally, I also believe I learned something about how to effectively coordinate tasks in a team and work together towards the same direction to be able to iterate as fast as possible in a startup environment.