CHALLENGES AND RISKS

Conducting a project like ESE always carries many challenges and risks. Our group consisted of four (and later three) people, with only very little experience regarding application development. Therefore, many challenges were given by the sheer amount of new technologies and synergies to learn.

This document will list some of the most important general challenges that arose, as well as one more in depth review of a specific issue.

ORIENTATION WITHIN THE PROJECT

One of the earliest challenges were on one hand the communication within the group, and on the other hand having to learn and understand the involved technologies. These are listed together, as the way to efficiently solve them was one and the same: group meetings involving group/pair programming. We set up a slack[1] project, as well as a WhatsApp[2] group for internal communication, but especially in the beginning this wasn't enough.
We learned fast, that sitting together at the same table and going through our code line by line improved our understanding of the matter significantly.

TESTING ONLY ONE SIDE OF THE APPLICATION

During development there were several times, where the backend had the implementation ready, but the frontend not yet. We had to find a way to test backend methods without having a frontend to issue the commands. Pascal recommended us to use Postman[3], which we did. Postman allows you to directly send HTTP requests to the backend. At the same time, we had the challenge to use Postman correctly, which was another challenge itself. After many googling-sessions we learned that we must set the header "content-type" to "application/json" and the body has to be entered as "raw" in a specific kind of way.

COMMUNICATION BETWEEN BACKEND AND FRONTEND

The main challenge that was omnipresent during the whole project was the communication between backend and frontend. Even after validating through Postman that the backend handles HTTP requests as intended, there was still the task to let the frontend sent the requests.
We knew how the request should look like but sometimes the request would not work as intended. Another recommendation by Pascal, Fiddler[4], came to the rescue. With the program Fiddler, we were able to see exactly which request goes to which port, while displaying the header, body and so on. With that tool, we could narrow the problem down to bad requests or unhandled/missing attributes and many more.

HONORABLE MENTION

The last tool we used a lot was another recommendation by Pascal: DB Browser for SQLite[5]. We already knew that tool from the lecture about databases, but it came in handy. This was our only way of checking the database itself, when we didn't have any visual implementation of checking whether our commands actually altered the database or not.

[1] https://slack.com/intl/de-ch/
[2] https://www.whatsapp.com/
[3] https://www.getpostman.com/
[4] https://www.telerik.com/fiddler
[5] https://sqlitebrowser.org/

LEARNING OUTCOMES

SPLITTING INTO SUBTEAMS

As the instructors recommended, we split our team into frontend and backend subteams. We started with two people in each but had to alter this division rather quickly. One group member left the course after some weeks, and thus we had one person strictly in the backend, one person strictly in the frontend and one person a bit in-between.
Through this specialisation we had very different learning outcomes from this project. Our backend specialists now know for example how to handle incoming HTTPS requests and how to use express.js to set database commands. As well as the fine tuning of a token creation and password hashing
And our frontend specialists refreshed their memory of HTML and know the angular and ionic construct inside-out, while also learning about detailed work such as creating custom ionic colours.
All group members gained a good piece of knowledge over Typescript, and we were able to help each other out throughout the whole project.

DOCUMENTATION

It also came in handy to create an overview over all involved endpoints, most specifically for the HTTP requests. We created a table early on, with URLs and the involved parameters and so on. The file is inside the documentation folder of this project.

The biweekly progress reports were also helpful for all involved parties. The assistants knew at a glance where we stand, and we were able to summarize our progress. For us it was useful to have one place for the most important TODOs and completed tasks.

PRACTISE MAKES PERFECT

This last learning outcome is the same as for every other lecture involving programming: practice makes perfect. While we struggled to hit the ground running with this project, it was even more fulfilling when we finally did. New programming languages, frameworks and many more are always a challenge but when investing time and determination, you will get better each day and that is exactly what we did.

One semester, 13 lectures and many hours of hard work later we can deliver a final product, that we can stand behind. It was an interesting challenge and we learned a lot and we actually had some fun while doing it. Sometimes more, sometimes less. It was interesting and quite a ride.