Lappeenranta–Lahti University of Technology LUT
School of Engineering Science

Miro Hagelberg
0604373
CT70A9140 Software Development Skills: Full-Stack 2024-2025

Learning Diary, Full-stack Module

TASKLIST GITHUB: https://github.com/Hageli/fullstack_tasklist

PROJECT GITHUB: https://github.com/Hageli/fullstack_project

USED TUTORIAL FOR PROJECT: https://www.youtube.com/watch?v=K8YELRmUb5o

Date: 3.9.2024

Activity: Environment setup, tasklist part NodeJS

The setup for NodeJS was quite easy and didn't take me too long to get the versions used in the videos. I have previously used NodeJS for a full-stack project, but I used older versions at that time. Some modules like crypto were new to me. I had used bcrypt prior to this course, but crypto was simple enough to understand almost immediately. Routing and creating the server was clunky without ExpressJs and I am glad that later in this course I can use other tools to make building the project a lot easier. The fs, os, etc. demos did not include any new information to me, but it was good to go through them regardless to refresh my memories on some smaller specific things.

Date: 3.9.2024

Activity: Tasklist part MongoDB

This was considerably shorter than the previous part since it did not require much programming and again, I had already set up all the clients and accounts to MongoDB. I added a couple of screenshots to the GitHub folder about opening the client and adding some data to the new database. Most of the things in this part were also familiar to me so I did not learn much, but again, it was good to remind myself of the different possible functionalities of MongoDB.

Date: 3.9.2024

Activity: Tasklist part ExpressJS

This part was way more interesting than the previous two. Using the EJS view engine was completely new to me, I have only used other view engines before. This part also contained a lot more programming which I like, but most of the things I could have done without the tutorial as well. The most interesting part of this was how the functions and routes were organized within the code and this was something I would have probably done a lot

differently if I was working on my project. It was good to gain perspective on how others think about formatting code and what sort of arguments they support their choices with.
The colors package was also cool. The ability to make your logger more readable is always useful.

Date: 4.9.2024
Activity: Tasklist part React

I have also used React before on some other courses and I am quite familiar with the basics of it. I have however used only the newer versions and since this tutorial was a bit dated, some things had to be done in unfamiliar ways. Overall this part was quite simple again since it mostly focused on the front-end and did not bring in a server to communicate with the React components. The JSON-server package was fun, I do not recall using it before. I can imagine it being useful if the user has no experience in working with a full-stack project. In the next part where the complete MERN-stack is created, I imagine things are starting to get a bit more challenging and I cannot rely as much on previous experience as these other parts.

Date: 6.9.2024
Activity: Tasklist part MERN-stack

I completed the contents of the first video quite quickly. Express-async-handler was a new package to me, I do not recall using it before, but it seemed quite useful when building a backend. The usage of the colors package was a bit of an odd thing, maybe it is more included in the latter parts of this video series. In this part, it was simply imported but not accessed after that for anything other than connecting to notify of the MongoDB connection.

Part 2 of the video series was the most challenging so far since it required me to build the entire authentication and authorization side of the backend. One of the most challenging things in this course is the slight stylistic differences that other tutorials have had and how I like to do things myself so I have had to learn new ways of implementing essentially the same functionality. This is probably in part due to the different versions of the same packages used and the syntaxes changing over time as the packages are improved. I still got everything working as intended after a bit of troubleshooting. I decided to work with a localhost version

of MongoDB instead since I consider it to be easier to use and I do not have to spend time making a cluster online.

My plan of not using an online cluster was ruined almost immediately while doing the third part after an update to MongoDB compass broke my localhost connection completely. After an hour of troubleshooting, I switched to the cloud version so I could continue working. After this, I was able to make everything work. MongoDB compass was causing some weird errors which I thought was my fault because I had some difficulties understanding React Redux at first. It turned out that there were no issues with the code itself, just the requests were getting refused at times by the database. React Redux had a lot of similarities with React as one might expect, but at the same time, it had a lot of new components that I had to learn. Overall, React Redux seems like a powerful tool that has a steeper learning curve than React, and for example, knowing and controlling the state of the website with slices was interesting to learn about.

Finalizing the fourth part of the project was not too difficult since all I had to do was adding the functionality of the goals into the frontend. Adding the goal slices and services was easier this time since I had already once done it with the user slice and service. During part 3 I was a bit worried about learning React Redux, but it was nice to notice that the basics were quite easy to understand after a bit of practice. Next, I will strip away some of the functionality of this tutorial in order to modify it for my own project needs. I am not yet certain which type of project I will build, but I will use the working login and registration forms that I did in this MERN-stack part.

Date: 9.9.2024 - 11.9.2024
Activity: MERN-project

I accidentally stumbled upon a MERN-stack tutorial while scrolling Youtube. I decided to give it a look and it included a lot of interesting packages and details that were not included in the tasklist videos while still utilizing React Redux and all the other required parts. The main thing that interested me was using Redux Persist to store the state locally in order to not refresh everything in case you accidentally close a page. The tutorial also included other neat tools like Formik and Yup which made working with forms much easier. I also used Material UI for styling and elements in this website. After reviewing my options of how to approach

this project, I decided to not reuse code from the tasklist videos since there was little to no overlap between most of the server routes and frontend elements.

I began by working with the backend and it was mostly done in a similar format as many other servers I have built with NodeJS and Express. This time I decided to not even try the localhost connection to MongoDB since it had broken during the tasklist part and I had already set up a cluster online that I could use. Everything related to the backend went quite smoothly and I did not have to troubleshoot almost anything while building the routes and controllers.

On the client, many problems started arising. I wanted to use Axios for the requests on the client but there were some strange interactions with the server while using it. Therefore I only have some of the requests done using Axios and the rest of the project using the traditional fetch. I know that with time all of this could be fixed and every request could be done using Axios, but I did not want to break a functioning project just to change a few requests, since it could have taken me a considerable time. Most of the issues were a result of Axios returning the data in response.data variable while the fetch promise needs to be resolved using await response.json(). This formatting difference could have required quite large modifications to the state variable storing etc. so I ignored it this time. If I ever want to scale up this project, this is something that must be changed.

On the client, I also noticed that the tutorial started taking shortcuts and was doing some bug fixing off-camera in order to get the project to work. This caused me some unnecessary headaches and I had to look for solutions myself, for example, while trying to make the like toggle actually work without refreshing the page. The project was also poorly tested in the tutorial. Once the finalized version was coded, it was only run one time in the end without trying to do some specific actions that might break the project. I ended up noticing some of these bugs that caused the website to crash, for example, while liking one's own post on the profile page.

I also had issues with the formatting of variables and arrays passed between the pages and components. I do not know how the tutorial managed to fix these issues, but even while trying the direct solution from the tutorial, I did not get it working without finding my own solutions. After changing the variables and passing some functions from one widget to

another, I got the widgets working and the client was refreshing and getting updated as intended.

The project is finalized as a business networking platform that can be expanded into a full-scale production social media site. At this time the website does not include a profile search through the text box and it does not include comment input. Also for a real website, for example, the Navbar icons would have to include real private messaging or other functionality that would activate onClick(). To avoid the project scope from getting out of control I decided not to include these features in this course's project.