Learning diary for the course:

Software Development Skills: Full-Stack

By: Severi Arpinen

18.12.2024 - Starting the course

I started the course by going through the moodle-page, visiting all the tabs and checking out the general stuff, meaning what to do to pass the course. I created a new GitHub repository, where I would be committing my work from the assignments.

“Full-stack", is familiar to me, I know what it is, but I lack the experience on the technologies used in it. I'm looking forward to learning a bunch of new things, hopefully it will pay off for me in the future.

Next, I started watching the first module's video about Node.js. I installed node.js on my pc from the website. After that I started to follow the video, playing around with different JavaScript codes and figuring out how they work.

I started to build a simple http server, following the video. I learned a lot, I had some problems first with the JS syntax, but I got the hang of it pretty quickly. I created my own http server and played around with it.

I created a simple router for my http server, which will direct to homepage and about page, and if you try to access to any other page, we get status 404 and display a message “page not found”.

When i got the basic http server working, I started creating A simple API. I made some users with different id’s and made the API get the users name when inputting the correct id. For example, the request with /api/users/1, returns the users name with id 1.

Building the API was kind of difficult to understand, because i was not using any frameworks for it. I learned about middleware's and handlers in JS. Next, I added some functionality to the API by adding users with POST requests.

Lastly, I watched the video and learned about different kinds of modules that you can make, these were File System-, path-, OS-, URL- and Crypto-, Event Emitter- and Process Object Module. (I don't know if the process object a “module”, but I'm going with it)

19.12.2024 - MongoDB & ExpressJS

I started to work on the 2nd module of the course, MongoDB. I haven't heard of it before, so I will for sure learn a lot of new things. I started by installing MongoDB, creating an account and making a new cluster to try things out with. After getting everything to work, I followed the video and connected to the database and gave it some data to try to work with.

I messed around with it, creating new data and modifying/deleting it. I downloaded some sample data and used aggregations to find specific entries with the given limits (for example in movies, only the movies that have a runtime greater than 100min and were released before the year 1930).

Lastly, I got MongoDB working in my VSCode. Using MongoDB was not difficult, and the MongoDB Compass made it really easy to learn and use.

After learning about MongoDB, I started with the next module, ExpressJS.

Just like MongoDB, i have no prior experience with ExpressJS. I am looking forward to learning a lot. At first, I noticed that using the Express-framework is a lot simpler than just using node by itself. I created a simple server, and made it display messages. I created some JS objects, that could be accessed by "/api/posts", which was not difficult with Express. I also made it to support queries with limits. I learned how SQL-injections really work, and how to make your code "fool proof"

I made my simple server so that you can input it from a form –body. I also added functionality that you can update the earlier posts with a put-request or delete them. I also created a logger-middleware that will keep track of what requests and from where they came from in the console.

Next, I created an error handler, that would give a specific error message, if you would try to access a post that doesn't exist. I created it to be a middleware.

Next, I added a "colors" package, to make the terminal clearer. Now it displays different HTTP requests in different colors in the terminal.

Lastly, I created some Frontend-code that would fetch the data from the backend server I have created. I created a simple form, where you can add new posts in the frontend website, and it adds it to the backend server, displaying the new post.

20.12.2024 - React

I started the React-module, and again, I had no prior knowledge of React before starting this crash course.

I started the module by creating a new React app and following the video where I would start to build a simple UI. I first was surprised that in React, you can basically write a mix of html and JavaScript, which was a bit confusing at first.

I created a start for a simple “Task tracker” -app and after using react for a bit, I started to understand how it works. I created each of the components to be its own file in the components folder, which made the components reusable and the code cleaner.

I added some functionality to the app by making it possible to delete a task.

Lastly, I added more functionality by creating a component, from which you can add a new task by filling in a simple form.

Finally, I finished the module by learning how React works with a “mock” backend server. I didn't create my own server for this part, since I would be doing that next when starting the final project of the course. The routing from front- to back-end was not that difficult; I am hoping that the project will be as straightforward as this was.

21.12.2024 - MERN-Stack

Today I started to work on the final project of the course, where I would create a bigger project that connects all the learned technologies into one project.

The first component for the project was creating a rest-API. I started by creating a basic file structure and creating a few simple routes for /api/goals. I tested these routes with postman, and they worked as expected.

Next, I added some error-handling with an error middleware file.

After that, I created the database for my project with MongoDB. I connected it to my project and made funcitonality with it so i can create, update, get and delete goals. This was the final part for the rest-API.

Next thing for the project was to create JWT authentication, to secure my API.

I created a new controller and model for Users, where I made a user Schema and new routes for /users and /login. After that I created the “create a user” function, where I check that all of the fields are filled, and that an user with the same name doesn't exits. If so, I encrypt their password and create a new user in the database.

After that, I added validation with JWT tokens. This was a new thing for me, so I learned a lot about encryption and hashing.

Next thing I did was create a middleware function to protect my routes, that are private. It gets the token and verifies that it's correct, before it lets you access the /users/me page.

I also added this authorization feature to api/goals, so the user can only see its own goals.

Now, I have the backend API finished, with CRUD-operations that have authorization and authentication!

22.12.2024 - Frontend

Next thing I started was the Frontend-application with React. I created a new React-application on its own folder, and started working on the frontend.

I started by making the different pages, that are Dashboard, login and registration. Next, I made routes for them in the app.js file. I made a simple header, that contained the links for the pages.

Next, I created the Register-page, where I had 4 input fields for the name, email, password and confirm password, and one submit button.

For the login page, I just copied the register pages, and modified it to be without the name and confirm password fields.

Then I created an Auth Slicer, which was a bit confusing at first, but after some experimentation, I got the hang of it. I created many states for the registering user, that would check the email and that the passwords match.

I had some questions about how exactly connecting the backend to the frontend works, but I figured them out pretty quickly. The Redux toolkit was quite confusing to use for the first time.

Lastly, I made the login-page work, which was almost the same as the register, but instead of creating a new user, I just validated it.

Next, I started to watch the final video of the MERN stack tutorial series.

I started by working on the “Goals” section of the app. I created a goal slicer and goal service files, that worked the same as the auth service and slicer. The big difference was that I needed to authenticate the user with the token, to show its own goals, that's where I learned about thunkAPI, that could access the states and get the token.

After I got that working, I created a simple UI, where you see your created goals and can delete and create new ones.

After that, the basic functionality was done, I modified the index.css file to suit it more to my taste, and the only thing left was to deploy the site.

The given way to deploy a site was through “Heroku”, who unfortunately doesn’t allow for free deployments anymore.