Lappeenrannan teknillinen yliopisto

School of Business and Management

Software Development Skills

Miro Rahkonen, 0561542

LEARNING DIARY, <CHOSEN MODULE NAME> MODULE

**LEARNING DIARY**

16.03.2023  
  
To start off the course I went over the general information about the course to check how to complete the course and what needs to be done. Afterwards I started watching the first tutorial video about the example project and followed it to create a sample program of a working website. It will most likely change a lot as the tutorials go on, but I got it working like was shown in the video. Finally, I created a Git-repository, so the code is available online from GitHub.

I am familiar with creating websites from the Introduction and Advanced Web Programming courses, so Node and most of the other things were quite familiar. Though the methods shown in the made program were different from what I was used to. I am used to using Express and routers to send and receive requests, but it was interesting to see different ways to do them instead using other libraries. I learned a bit more about how requests and responses are done overall. I also tried to use Heroku to setup an online hosted website of the program, but it now requires a credit card to verify the account which I wasn’t comfortable with so I will have to find another method to host the website. GitHub has a way to host websites so I will try to figure out if it is possible using their service instead.

20.03.2023

Today I studied and followed the second video from the video tutorials which went over the use of MongoDB and different commands to use it. I am already familiar with mongoose from the Advanced Web Programming course, so I knew most of the commands already aside from a few. All of the commands shown were the same as the commands that you would input into the JavaScript code which I found helpful. In the Adv. Web programming course, we used mongoose inside of the JavaScript file, but in the tutorial the commands were done inside of the mongo terminal which is a bit different. I wasn’t previously familiar with the mongo terminal so it was nice to learn so that in the future I can create databases and collections from the terminal itself instead of creating them alongside the JavaScript program which is helpful.

What was new to me in the video was different ways to updating data inside of already stored objects in the database, which I wasn’t that familiar with before. I’ve had a couple instances where I had to update old data with new data, but it has been quite sparse. There were many good example commands to insert new key values or update existing key values, which will probably come in handy when programming the project.

I think this was a good tutorial to show how MongoDB is used in actual applications and servers, and I believe I learned the basics to be able to use it to create databases in future projects.

21.03.2023

Today I studied the next lecture which included some examples of Express and some of its libraries, as well as the handlebars template engine. There wasn’t much new to learn in this tutorial as I was familiar with most of the stuff from the advanced web programming course, so I knew how to create an Express server and a router, and how to handle requests and responses using them. Though I hadn’t previously used the handlebars template engine so it was new to me. I think I understood the fundamentals of how it works, and I was able to create a simple webpage with a form where you could submit new member data to be saved on the server and all the members would be displayed in a list on the website using the handlebars template.

24.03.2023

In today’s session I went over the first couple parts of the angular tutorial for the ‘Tour of Heroes’ program. In the tutorial I learned the basics of how to set up angular programs in Visual Studio Code, and different aspects of angular such as directives, components, services, and routing. I haven’t seen angular before, so this was all new to me. The example showed quite in-depth instructions and explanations of each command, so I think I understood how it works to some extent. I was able to complete the example program to the point where you’re able to use services to transfer data between the different components, and switch between different webpages to show different bits of information of the heroes and change them. I had a couple points where the program wasn’t working as was shown in the tutorial, but I was able to figure out the problems by going back and checking what wasn’t correct.

I learned how to create components and how its variable syntax works, and how to use services to send data to the components so the data will be displayed on the webpage. I learned quite a bit about how syntax with angular works, and how to use components and services together to create a functioning program. I will continue the tutorial at a later date on how to simulate a data server with HTTP calls.

I found angular quite interesting, and it might be useful to learn so I can use it in future job opportunities if it is required.

27.03.2023

Today I finished the Angular example program Tour of Heroes, where I added the functionality to send and receive data to an API, which would store and change the data stored on a database that stores information about the heroes. This section was quite complicated, and it took me quite a while to get it working. I wasn’t able to get the API URL ‘api/heroes’ to work, as it wasn’t set anywhere in the tutorial, and it was assigned dynamically according to some file. Fetching the URL like how was done in the tutorial didn’t return anything and none of the heroes were showing up on the website so I wasn’t able to test the functionality that was produced later in the tutorial. None of the fixes I found on online forums worked but eventually I figured out that the problem was a wrong import in the app.module.ts file. It was quite hard to find the problem as the program didn’t produce an error message and I couldn’t find which file the error was in. After I was able to fix this issue the rest of the functionality was easier to produce. I had prior experience of APIs on the Advanced Web Programming course, so I was familiar with the get, put, and delete fetches from APIs to change data on a database. Angular is quite different from regular JavaScript so it took me quite a bit of time to understand the fundamentals. I understood the basics of what was produced, but it will take a bit to be able to produce applications on Angular on my own without a tutorial. Being able to communicate between different files was pretty interesting to see, but I think it would take a lot more effort for me to understand it in detail. The one thing I didn’t like about Angular was how all of the code was distributed between so many files, and switching between the 40 different files was a bit of a pain.

Overall, I enjoyed using Angular but it was quite difficult to understand as it wasn’t similar to other template engines I’ve worked on, and the communication between the files seemed quite strict and one thing being wrong made the whole module not work. I would assume you get used to working on Angular as you create more programs using it so it might become a bit clearer eventually.

28.03.2023

For today I started the MEAN-stack portion of the course and finished the parts that were done in the first 4 videos. I produced the initial file structure and some features, such as the post route for creating an account, storing users in a mongoose database, and authentication with passport and JsonWebToken-tokens. I had a bit of difficulty with this section as the video is quite old and most of the fundamentals were outdated, especially with mongoose. Most of the functions of mongoose had changed last month, so that callbacks are no longer allowed on any of the search commands, such as findById() or findOne(). The issue is discussed in [this StackOverflow thread](https://stackoverflow.com/questions/75616051/mongooseerror-model-findone-no-longer-accepts-a-callback) that is made on the 2nd of March 2023, so it is quite a new issue. The problem boils down to no longer allowing the use of callbacks at the ends of mongoose functions, so you would want to do a command such as ‘findOne({username: req.body.username}(err,user)=>{})’, but the red portion is no longer allowed, so now you need to set the found object to a variable adding an await before the mongoose command. Example of a working command: let user = await findOne({username: req.body.username}.

This meant that most of the syntax used in the tutorial no longer worked so I had to modify the code quite a bit for it to work. Thankfully I had used passport and JWT-tokens before in web programming courses, so I was able to figure out a way to bypass this issue.

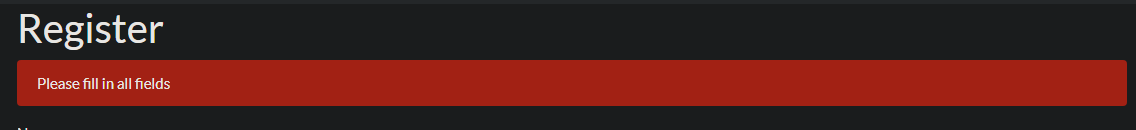
Despite the problem with mongoose, the other things taught in the tutorials thus far were quite familiar to me so I was able to follow along and complete the necessary features. I will continue the application at a later date and complete the rest of the features.

30.03.2023

Today I started working on the angular part of the program, and I finished the sections that were covered in videos 5 and 6. I had some difficulties with producing these parts of the program as the videos were quite old and the code was no longer valid in some parts. Especially with the FlashMessagesModule library, which I could not get to work at all. I used the fix that was told in the moodle page, but it did not work either. Putting the FlashMessagesModule into the app.module.ts imports always created an error and changing version numbers or other fixes online would not fix it. Eventually I gave up and decided not to do the feature. It was only a small feature that doesn’t compromise the functionality of the rest of the program. Aside from this particular module, I was able to find fixes for the rest of the errors that were due to outdated code, and I was able to create the necessary functionality for the program. This part was mostly dealing with angular and it was easy to follow along thanks to the previous angular tutorial earlier in the course. The functionality was quite straight forward to produce as all of the concepts were taught in the angular tutorial. I was able to remind myself of how angular works which is helpful in learning. I will continue adding more features to the program at a later date.

03.04.2023

Today I continued the program and completed the parts covered in part 7 of the tutorial. I learned more about for TypeScript works as I had some difficulties with this section due to the video being outdated and the syntax and some other things changing since the video was published. I created the account creation paths and validation, and I was able to create the functionality to register a new user to the server, so that it could be logged in using the provided information. I also created a similar solution to the FlashMessagesModule, which would show an error box at the top in case the provided information wasn’t corrected. Instead of the library used in the video, I created my own div into the HTML-file which would be shown if a variable ‘showDangerAlert’ was set to true in the module file. This provided a similar solution, and it works almost the same way, but it wouldn’t be brought over as a notice to the login page like was shown in the tutorial video. A screencap of the alert box is shown below, and it only appears once you click the submit button and some of the information is wrong.



The main issue I had in this section was the parameter type in the subscribe command as shown below.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedDue to changes in TypeScript, the code in the video is no longer allowed and you aren’t able to access the properties of an object parameter the same way as in JavaScript. Doing it according to the video produces an error ‘property “success” does not exist on type object’. Trying to find a fix for this problem made me google a bit about how objects in TypeScript work, and I think I learned a bit more about how it works differently from regular JavaScript. For some reason in TypeScript, you aren’t allowed to access the object properties of an object which comes as a parameter from another function. I found a fix for this problem and the solution was to create an interface for the response at the top of the file, which would be assigned to the data variable in the subscribe function. Below is the solution I created.

Text

Description automatically generatedThis allowed me to access the ‘success’ property of the data object since it was now of the interface type ‘response’, and it had a property named ‘success’ which is able to be found.

It took me a bit more time to produce this part then previous sections, but I learned a lot more about how to produce elements such as the alert box in angular, and it made me a bit more knowledgeable about TypeScript as well.

04.04.2023

Today I continued the part 9 of the angular tutorial and completed the necessary parts of it and also I created my own component of the FlashMessages service, which was made in the last video. In the last session I hard coded the div element into each file, but today I made it into its own component called <alertbox>, which I could add to the register, login pages, and to the navbar when logging out. Example of the component in the login page below.

Text

Description automatically generated

Alertbox-component inside of the login page

This was a bit easier to implement than the hard coded div, which mimicked the functionality. There was still a bit of work required to add this to each page, as there were a few lines of code that needed to be added to the component to make it work, including making the alert show up, set the alert message, and then finally make the alert disappear after a couple seconds. I used the setTimeout() method to set the showError variable to false after a couple seconds, so it would make it work somewhat similarly to the FlashMessagesModule service, which didn’t work last time.

There was one part in the video which I had a bit of difficulty producing, which was the ‘angular2-jwt’ library, which I could not get to work. When I tried to install the library with npm, it complained about some dependencies not being correct, and using the –force or –legacy-peer-deps options did not work either. I ended up using a newer library called ‘@auth0/angular-jwt’, which seems to be a newer version of the other one and it worked almost the same. There was some setting up required to get the functionality to work the same way as in the video, but the library documentation was easy to follow to make it work. The only thing that needed to be changed was to add a few lines of code into the app.module.ts file and it worked flawlessly. Other than this library, the rest of the features were easier to create and I was able to follow the video to make the functionality. I added the functionality to login and logout of the website, and in the case of logging in the JWT token would be saved to LocalStorage so that it can be retrieved with the angular-jwt service, to check if it’s valid. The logout feature would remove the JWT-token from LocalStorage and make you log out of the website. I also added checks for the buttons in the navbar, and made the login and register buttons disappear if the user was already logged in. Additionally the dashboard and profile buttons would be hidden if the user isn’t logged in, and logging in would make the buttons appear. I also created a simple layout of the profile and dashboard which were also produced in the video.

Most of the features in this part were quite simple to produce, as I had previous experience with a similar thing from the Advanced Web Programming course, where we also used JWT-tokens and LocalStorage to store and get the tokens for authentication. The skills from that course made it easy to understand what was needed to make this part of the program work.

05.04.2023

The MAIN-stack project seems to be about finished at this point and the only remaining thing in the tutorial was to deploy the page. Though I didn’t want to do the deployment with Heroku since the service has since changed to require a credit card to verify your account and get the option to deploy webpages. Instead of doing the deployment I decided to add some extra features into the application, so I added functionality into the dashboard page since it was left empty in the tutorial. Into the dashboard I added the functionality to add and remove notes, so you can save them and delete them later whenever you don’t need them anymore. I added the notes to be saved into the same mongoose model as the user information so that they would be saved into the same place as the user data, and it would be easy to retrieve. To make it possible to add and remove notes, I added some new routes into the app routers file, so that the application could make requests into the server and do the post and delete requests to manage specific notes from the user data. To make the notes show up on the dashboard page, I made a for-loop into an unordered list to display all of the notes in a list, and I made it show the date the note was submitted and the note’s message after. I also added a delete button at the end of the message, which after clicking will trigger a delete request to be sent to the server, which will remove that specific note. I was pretty familiar with how to produce new elements into the webpage thanks to the tutorial, so it was pretty straight forward to add this functionality. I think I have gotten the hang of how to use Angular to create working websites. Adding this functionality was a good test for me to see if I’ve gotten any better at using Angular, which I think I have after producing the functionality of the MEAN-stack tutorials.

I think at this point the project is almost finished and next time I will check how to make the application work from downloading it from the GitHub page and I will write the README-file for starting the application.