CT30A3203 Web Applications

Course Project Assignment

Petteri Mäkelä
0506167
16/12/19

This is the documentation for my course project assignment for the web applications course. The project served as an introductory project for full-stack web development. During the project and the course, I learned many fundamentals of web applications. The assignment was to make a social media platform, a microblogging service to be exact.

I chose to build the program with a react boilerplate. This gave a good start for the project, and it provided a working base to start improving. After initializing the folder with git and npm I was good to go. Libraries were added as the program grew. I saw a library or two that aren't used for the final product. This is because I did many things with a trial and error mentality. Overall though, the libraries suited my program well, and provide great functionality for it.

For the design, I started with the basics. After I had a good enough front-end, I made it functional with react. Then imported Mongo with the database. After all this was done, the program was functional, but it's front-end still quite basic. Sadly, I did not have time to finish the front end to give it a little more shine. It does however function well, and it checks all the minimum requirements for the course project.

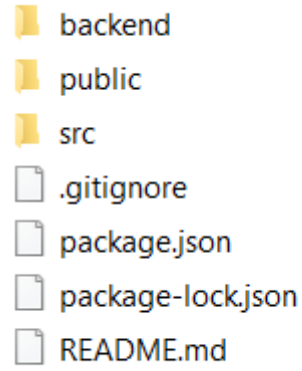
Program architecture is available on the second page. I've decided to show it as a picture of my main folders, which is everything there is to the program. Backend and front-end are their separate parts in the program. The backend runs simply with the server.js file. The data.js file is only a helping hand to setup our schema for Mongo. The front-end is run through parent and child components. App.js is the main file, which has the Feed child, which further divides into different components. This allows a functional react architecture and a clear and readable design.

There isn't much needed for the setup. I have whitelisted the mongo database, so anyone can access it. The app runs on local host, but internet access is still required for database access. The system should run with a simple npm command, which boots up both the front- and back-end. On terminal the backend runs with 'node server.js' command on backend folder, which is included in the start command. The front-end boots up with react-scripts start, which is also in the start command. The front-end is set to run on port 8080 for Rahti, which sadly I could not get working after adding database functionality.

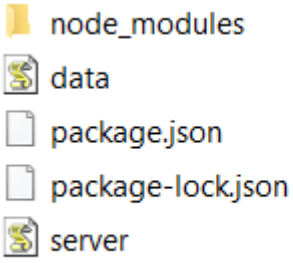
Minimum implementation	30
Using React.js front-end	15
Using a database, such as Mongo, Redis, or any SQL-compatible	5
Total	50

Program Architecture

Main file structure



Backend



src (Frontend)