CT30A3203 – Web Applications Project Work

Sara Heikkinen, 567012

Implemented features

Feature	Max Points
Basic features with well written documentation	25
Utilization of a frontside framework (React)	5
Use of a pager when more than 10 posts	2
Time stamp on posts and comments	2
Use of a syntax highlight	2
Vote (up or down) posts and comments (only one vote per user)	3
Provide a search that can filter out only those messages that have the searched keyword	2
404 page if user is on some random page	1
User can choose whether the app is on English or Finnish and change language while use	3
User can choose how many posts he wants to see in one page (updated pager)	1
User gets error messages in login and register if there is some error (not strong password etc.)	2
Side menu bar	2
Page has a footer in every page	1
A lot of CSS and MUI used	2
IN TOTAL	**53 (50)**

Technical choices

Server:
Node.js (mandatory), Express framework is used to create it. Express was most familiar to me and it was used also in week's 11 exercise.
Client:
Client side is made with React frontside framework. React felt the clearest way to implement the front-side of the application. In React one can implement HTML and JS in same file, and it felt so much easier way to execute the client side.
Database: For database I chose MongoDB because it was used in the course. I am not familiar with SQL databases, so MongoDB was a safe choice.
Styling for front-end: I used a little bit of Material UI components and then used basic CSS because it was easy to implement after the functionality.

Installation of the project:

This project requires node.js and node package manager (npm)

For client side one must install (cd client && npm install)

- @emotion/react
- @emotion/styled
- @mui/icons-material
- @mui/material
- i18next
- i18next-browser-languagedetector
- i18next-https-backend
- react
- react-dom
- react-i18next
- react-router
- react-router-dom
- react-scripts
- react-syntax-highligher

For server side one must install (cd server && npm install)

- bcryptjs
- cookie-parser
- cors
- debug
- dotenv
- express
- express-generatos
- express-validator
- jsonwebtoken
- mongodb
- mongoose
- morgan

User Manual

To get the app to run one must have two terminals open, in one you run the server side and in the other you run the client side.

Running the client side: in the root folder, write “npm run dev:client”

Running the server side: in the root folder, write “npm run dev:server”

In the app user can write posts and comment to them if he is logged in. If user is not logged in, he can only see other people's posts and comments on them. Also, when user is logged in, he can vote (like or dislike) on posts.

Registration:
User should submit their email-address, username (which should be unique) and password. The password is written twice so that it's right.
Password should follow strong password rules (at least 8 characters, must include upper and lowercase letters, numbers and at least one symbol). If it doesn't or the other inputs aren't valid, user will be notified.

Login:
User can login with username and password. Error message is displayed if login fails.

After logging in:
When logged in, user can post, comment, or vote existing posts and comments. User can add a new post from front-page's button. User will find the log out option from the same place as they logged in.

In the front-page user can choose how many posts they would like to see in front-page at once. Options are 5, 10 or 25. If there are more posts available than user likes to see at once, there is a pager and user can browse different pages of posts.

User can search with a specific keyword and app will return all the posts that contains that specific keyword. User can then read the searched posts. Search is available also for not logged in users.