Taru Haimi
0565878

Documentation for project work of Web Applications

This is a documentation for the project work of the course Web Applications. This document explains how the web application is made, how it can be installed and used, and which features can be found in the application.

Technology choices

Backend and frontend

This Web Application project has backend running on port 1234 and frontend running on 3000. Backend has been carried out with Express-generator and frontend is carried out with React. There is possibility to run them separately in two ports or together in the same port (1234) depending on how we want to run the project (in development or production mode). Basically, the project is built in the same way as exercises in week 11.

Authentication, authorization, and data management

User of this web application can register and log in. Authentication and authorization are carried out with JSON Web Token, which is saved to the local storage after user has logged in. Every user has the same rights (e.g., every authenticated user can post codes and comments and edit them). Users and their codes and comments are saved in three models to the MongoDB-database.

Responsive design

To make sure that the web application is responsive and usable for different devices (computer vs. mobile devices with different screen sizes), there is React and MUI libraries used and also CSS used. The usability is also tested by web browser's DevTools by changing devices and the screen dimensions (e.g., long texts scale to multiple rows and don't go over the screen borders).

How to install the web application and run it

When user wants to run this web application, no additional installations are needed. One needs only open the whole folder in the VS Code and start program in the terminal of VS Code. The commands for running the application are described below (assuming that operating system used is Windows).

This web application is available from Github: https://github.com/taruuwws/ProjectWork

Production mode

If user wants to run application in production mode (in one port), it starts with command:

- $env:NODE_ENV=”production”; npm run build; npm run start;

When user goes to localhost:1234, the application is running there. This mode is a default option for running this web application. npm run build is not a mandatory part of that command, but it needs to be run when changes are made in the client side. After setting the environmental variable to production, it is enough to use only command npm run start.

Development mode

If user for some reason wants to run the application in development mode (two ports), it is also possible by running it in two terminals with commands:

- $env:NODE_ENV=”development”; npm run dev:server
- npm run dev:client

One can open server side by going to the address localhost:1234 and client side should open automatically (thanks to React app) in the address localhost:3000. If one wants to run web application in two ports, they must also edit a little bit app.js-file in server-folder: rows 27 and 29 are commented away for running production mode, so one should take “//” away from those rows for running development mode.

User manual

On the website there is a navigation bar with buttons on top of the page. User can navigate with those buttons to all the subpages that the web application serves (in other words, user doesn’t need to write any URLs by themselves). Buttons showing are a bit different based on if the user is logged in or not. If user is not authenticated, there are links for homepage, registering page, login page, and tutorial page. User can also change the language of the web application between English and Finnish. If user is authenticated, registering and login buttons are not visible, but instead of them there is a logout-button. In addition, there is a link to the page where user can view and edit posts and comments which they have written themselves.

At homepage if user is not logged in, they can see other users’ posts, comments, and votes given to the posts and comments. If the user is logged in, they can in addition add new posts and comments, and also like or dislike every post and comment (only one like or dislike per post/comment). Posts and comments become visible when the page is refreshed, votes become visible automatically.

To register to the web application, one needs to fill their email, username, and password at the register-page. Email must be in the format of email (contains “@”) and password must be a strong password (minimum length 8 characters, and must contain at least one big letter, one small letter, one number and one special character). Username is free of choice. If email is already in use or password is not strong enough, error messages about them are shown in the terminal as console logs (but not on the website). Register has been successful if the page redirects automatically to the login-page.

To log in to the web application, one needs to fill correct email and password at login-page. If credentials are incorrect or user doesn’t exist, the error messages are shown again in the terminal (but not on the website). Logging in has been successful if the page redirects automatically to the homepage. Also the navigation bar changes in the header top of the page (changes in the header were described in the first paragraph of this chapter).

Taru Haimi
0565878

To log out authenticated user can simply click logout-button in the navigation bar.

Changing the language is also very simple, user just need to click FI or EN based on which language they want to view the web application.

Tutorials-page explains the same guidelines as this chapter, but a bit shorter.

If user is logged in, they can view their own posts and comments via the button in navigation bar. In that page user can also edit them.

Features implemented and points

Checking that all the mandatory requirements are fulfilled:

Feature:	**Done?**	**Notes:**
Implementation of backend with Node.js	Yes	Used Express generator
Utilization of database	Yes	MongoDB
Authenticated users can post new code snippets and comment on existing posts	Yes	Token for authorization is stored to the local storage
Non-authenticated users can see posts and comments	Yes	
Some page listing all the posts	Yes	Happens at the front page
After opening one post, comments are also listed	Yes	Every post has a button for showing its comments in the front page right below it.
Responsive design	Yes	Used MUI library.
Documentation	Yes	This paper

All features and the points I'm aiming at

Feature	**Notes**	**Points**
Mandatory features		25
Utilization of a frontside framework	React	5
Vote (up/down) posts and comments (only one vote per user)	There's separately like and dislike buttons, and user can click only either of them.	3
Users can edit their own posts/comments	Separate subpage for viewing them.	4
Own feature: internalization	The whole webpage can be viewed both in English and in Finnish, as all the texts have been translated in both languages. This is a very useful feature for a web application when it is used in the real life.	2
Together:		39