Project Work Documentation

Vili Raunola 0543366

Technology Choices

Server & database

The server is done in Node.js. I started the project by creating an Express.js app by running the express generator command. I chose it because it is a quick way to generate a solid foundation for the server that will provide the API (Application Programming Interface) point for my front end. I chose Express.js also since it is the only framework that I have used to create servers with Node.js so I was familiar with it.

For the database I chose to use MongoDB with the Node.js framework called mongoose.js. I chose to use MongoDB in my project because it is also familiar to me. I have used it before in the LUT's MEAN-stack course where I got to use it for the first time. I have started to like it a lot since it stores the data in JSON-like documents which in my opinion is great since it gives the developer a lot of flexibility. Another reason I chose to use MongoDB is mongoose.js. It provides a great framework to operate the MongoDB. In this project I chose the database to be on the web. I did this because I wanted to have the same database available to me whether I worked on the project from home on my desktop or on the go with my laptop. The only difference to locally hosted database is the connection URL that is given to mongoose when connecting to the database.

Authentication to the app was implemented with bcrypt.js. Bcyrpt.js is used to hash passwords. When the user first registers they give a password that is sent to the server. The server then hashes the password using the bcrypt library and generates a salt to them. This hash is then saved to the database and when the user wants to authenticate, they send their email and password to the server. Now the server does the hashing again to the received password and compares the hash to the hash that is found in the database. I used this method to authenticate the user since it is much safer to store the hashed version of the password in the database compared to saving the plain password. And in the case of the hashed password leaking, it would take a long time to find a matching hash for even a single user making it good against data leaks.

Authorization is done by combing Json Web Tokens and passport.js middleware. When the user has authenticated a JWT is sent to the browser's session storage. Whenever the user tries to access a protected API, passport middleware is ran. Passport checks if the JWT that is provided with the request is valid. Based on the result user is either allowed access to the API or denied it. I chose JWT because they are straight forward to use and provide a secure method to enable authorization. I chose passport.js to use as the

middleware to do the authorization since it had great documentation explaining how to implement it to my project and providing code examples.

Front end

The front end is done purely using React with a Materialize UI library. I have done one project previously using MEAN stack, meaning I am familiar with Mongo, Express, Angular and Node. I wanted to try something new for this project, so I chose to use React instead of Angular. To make the site look better and to have access to more components I decided to use MUI. MUI has great documentation with good examples so starting to use it was quite effortless. It also integrates well to the React which makes it intuitive to use. Also, thanks to the MUI library I was able to make the layout responsive since all the components that come from the library are designed to be responsive.

Installation Guide

Download & Install packages

- Download repository as zip file
- Unzip it
- Go to the project file aka the root file in which the server and client folders are located
- Command to install node modules (will install both server & client):
    - `npm run install`

Production mode

- Build the front end by running the command:
    - `npm run build`
- To run in production mode:
    - `SET NODE_ENV=production& npm start`
- Now the whole project is served at http://localhost:1234

Development mode

- To run in development mode open two terminals and run these commands, one in each terminal:
    - SET NODE_ENV=development& npm run dev:server
    - `npm run dev:client`
- Now the server is served at http://localhost:1234
- Front end is served at http://localhost:3000

To change the database location

- If you want to change the location in which the database is hosted, go to: /server/config/database.js
- In there change the database connection string to your own

User Manual

1. Users can register to the site using the Register page.
2. After registration user is redirected to the login page where they can login using the email and password that they provided.
3. After logging in users can:
    - Create a new post. They can choose a title and the content of the post
    - Vote on posts and comments in the single post view
    - Comment on posts in the single post view
    - Edit their post or comment in the single post view
    - Delete their post or comment in the single post view
    - Check their profile information
    - Logout.
4. All users including non-logged in can:
    - View all the posts in the Posts page.
    - If there are more than 10 posts, they can use the pager to go through the posts.
    - Click a post:
        - See vote counts
        - See comments
        - Click username to see information of the creator of the post/ comment.
    - Search posts with a key word that they expect to be in the post.
5. To use admin, user can log in with an email: admin@admin.fi password: admin
    - Has all the features as regular logged in user.
    - Admin can delete and edit all the posts and comments.

Features implemented

Feature	**Points**
Basics features & well written documentation	25
Users can edit their own comments/posts	4
Utilization of React	5
Pager when more than 10 posts are available	2
Admin with rights to delete & edit posts & comments	3
Search for posts with a key word in them	2
Vote comments and posts up and down. One vote per user	3
Can click username and see user profile page	2
Last edited timestamp is stored in database and shown with posts and comments	2
Total	**48**
My features	**Suggestion for points**
Navigation bar that shows only links that are available based on if the user is logged in	1
User's can delete their own posts and comments	1
My total	**2**
All combined	**50**