Stack

* Frontend: React with Material UI
* Backend: Node.js with Express
* Database: SQLite

I wanted to use a SQL-based relational database instead of a document-based one like MongoDB, mostly as a refresher on SQL. Since the website won’t be using any videos or other special data types, the database can use text only. Images can be hosted on the server and referenced with a URL.

I chose React and Material UI to make components reusable and responsive on mobile.

Installation:

Make sure you have Nodejs and npm installed

Install dependencies by using the npm script “npm run setup” on the root directory. If this fails you can also install dependencies manually by first running “npm install” in the root directory, then “npm install” in the “client” directory, then “npm install” in the “server” directory.

After dependencies are installed, run development server with command "npm run dev". You can also run the backend and frontend separately in different terminals with commands "npm run server" and "npm run client"

Tested with: Node.js version 19.6.0 React version 18.2.0

Usage:

**Logging in and registering:** Click on “Login” or “Register” on the top app bar. Enter your credentials and click Login/Register. If there are any errors, they will appear to the left of the button.

**Logging out:** Login, then click on your profile icon in the top app bar, and select “log out”.

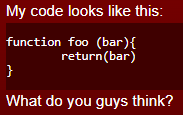
**Voting:** Log in and click on the arrow icons next to individual posts. You can only vote once per post. You can change your vote by selecting the other option, or cancel your vote by voting the same option again. You should see the vote score of the post change in real time.

**Create a post:** Log in, thenclick on the “Add post” button at the bottom of the front page. Enter the content of the post and click “Add post”

**Code highlighting:** Available in posts and comments by using <code> </code>  tags.

Example: My code looks like this: <code> function foo (bar){ return(bar) } </code>  What do you guys think?

Result:

[](https://user-images.githubusercontent.com/77271158/221673389-decb9e6d-a1a3-4692-84ae-8a046d4fe279.png)

**Edit a post:** Log in, thengo to one of your own posts. Click on the “Edit post” button and make your changes and publish the edit.

**Delete a post:** Log in, thengo to one of your own posts. Click on the “Edit post” button and click on “Delete post”

**Add a comment:** Log in, thengo to any post and write your comment in the comment field. Note that this is not visible when you’re not logged in.

**View your profile:** Login, then click on your profile icon in the top app bar, and select “profile”.

Features

Backend:

* Backend implemented with Node.js using Express framework.
* An SQLite database containing all persistent information
* API to access and modify the information in the database
* Use of parametrized queries to protect against SQL injections
* Encryption of passwords in database using bcrypt
* Use of SQL constraints to keep voting data valid after successive votes by same user

Frontend:

* Frontend implemented with React and Material UI
* Reactive UI usable with mobile devices
* Ability to login, logout and register as a user
* Password requirements (8 characters, one upper- and one lowercase character, one number and a symbol required)
* JWT-based authentication
* Authenticated users can create posts, comment on posts and vote on them
* Ability to edit and delete your own posts
* Pagination can be used when over 10 posts are in the database
* Code highlighting syntax for posts and comments
* Timestamp of post creation visible

|  |  |
| --- | --- |
| **Feature** | **Max points** |
| Basic features (as stated in the previous chapter)  with well written documentation | 25 |
| Users can edit their own posts | 4 |
| Utilization of a frontside framework, such as React, but you can also use Angular, Vue or some other | 5 |
| Use some highlight library for the code snippets, for example https://highlightjs.org/ | 2 |
| Use of a pager when there is more than 10 posts available | 2 |
| Vote (up or down) posts and comments (only one vote per user) | 3 |
| User can click username and see user profile page where name, register date, (user picture) and user bio is listed | 2 |
| Last edited timestamp is stored and shown with posts/comments | 2 |
| Password encryption in database | 1 |