Simple login – register app

First, thanks for taking the time to test my application and reading this document.

The app is based on two services: the client and the server.

On the server side (./server)

- Manages API calls
- Manages user authentication

The server has 2 different routes:

- /signup (POST) which is going to take the information to create a user in MongoDB, a boilerplate (model) of the necessary JSON properties that the body of the request need to have is on (./server/models).

  1) User submits input
  2) Check for duplicate email
  3) Create the user and give back a success response

- /Login (POST) which handles user authentication asking just the user email and password, this route query the MongoDB and makes sure credentials are correct, after that a Json Web Token is created with the user information as payload and given back in the response.

  1) User submits input
  2) Check if user exists
  3) Check if password is correct
  4) Create JWT and send it back in the response

  All the routes of the server are managed by controllers that can be found on (./Controllers).

  The server configuration consists in a .env file that must be created before running the application which is going to have 3 variables

NODE_ENV=production - Sets out environment
PORT={PORT_DESIRED} - What port the server is going to run on, if changing the port from the one given in the documentation a proxy property from the client must also be changed on the file client/package.json
MONGODB_URI={ Mongo Link connection }

For interaction with the MongoDB database, I used a library called mongoose an Object Data Mapper that helped me easily create the collection needed and query the database in a simple manner.

The application can run in two ways:

- Locally
- Virtually with the use of docker

More information on how to run the app can be found on the readme location in the GitHub repo: https://github.com/claudiojsaillant/LoginRegister

On the client side (./client)

- Provide views
- Secure routes
- Interacts with the API to create and authenticate users

Client utilities (./client/src/Utilities)

- Provide templates for the API calls needed for the app (./apiCalls.js)
- Provides the user session if the user is authenticated and can also log out a user that is currently logged in (./sessionManger.js)

Components (./client/src/Components)

- Provide reusable components that can be re-used multiple times
- Uses utilities to protect routes depending on the user authentication
    - PrivateRoute component checks if the user is logged in before rendering the view passed as an argument
    - PublicRoute component checks if the user is not logged in before rendering the view passed as an argument

Views (./client/src/views)

- Log In view saves token in a cookie after a successful login with Cookies.js
- All forms in views are handle by a function called handleInputChange which immediately puts any user input in the state of the Component

- Inputs are checked by regular expressions before making requests to the backend