Tutorial 10: User Authentication and Authorization

Task 1: Explore a sample project for building backend and frontend of a MERN stack application to handle user authentication and authorization (Sign in, Sign up and Sign out) using JSON Web Tokens

1. Refer to the two links below which provide detailed explanation on how to implement the backend and frontend of a MERN stack application
   a. Mastering User Authentication: MERN Stack Login Page (Part 1 — Backend): https://blog.stackademic.com/mern-creating-a-full-stack-authentication-system-using-jwt-and-bcrypt-part-1-b0c773ec4af0
   b. Mastering User Authentication: Building a Simple Login Page using MERN Stack (Part 2 — Frontend): https://medium.com/@sanjanashivananda07/mastering-user-authentication-building-a-simple-login-page-using-mern-stack-part-2-frontend-ad6602f7351d

2. Download a copy of the code from the author's GitHub: https://github.com/Sanjanaashivanand/LoginPageApplication

3. In your terminal, move to the project's **backend** folder to install all the dependencies using this command: `npm install`

   C:\Users\Chiam\Desktop\LoginPageApplication**backend**> `npm install`

4. In your terminal, move to the project's **frontend** folder to install all the dependencies using this command: `npm install`

   C:\Users\Chiam\Desktop\LoginPageApplication**frontend**> `npm install`

5. Create a **new database** in MongoDB, for example, you may name your database as '**mern-auth**' which has a collection called '**users**'.

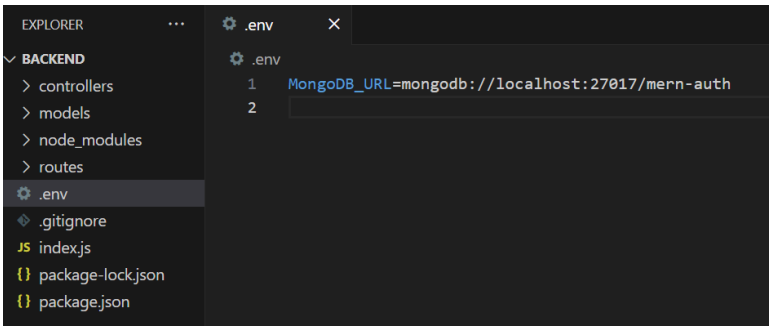
6. After installation, create a **.env** file in your **backend** directory to store environment variables. Open your **.env** file and create 1 environment variable called 'MongoDB_URL' with our database connection link that we need to pass to our application's environment like the example code below:

```
MongoDB_URL=mongodb://localhost:27017/mern-auth
```

   If you are using MongoDB Atlas, your database connection link should look like this: `mongodb+srv://<username>:<password>@<cluster>/<dbname>?retryWrites=true&w=majority`

   (**NOTE:** Replace `<username>:<password>` with your MongoDB Atlas's username and password)

   Screenshot of Backend's .env file:



7. Now, you can start your server by running npm run server in your terminal.

```
npm run server
```

If all these are successfully executed, your terminal should look like this:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Chiam\Desktop\LoginPageApplication\backend> npm run server

> backend@1.0.0 server
> nodemon server

[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server index.js`
app is running at 8000
Database connected successfully
```

8. Now, you can start your react app by running **npm start** in your terminal.

```
npm start
```

If all these are successfully executed, your terminal should look like this:

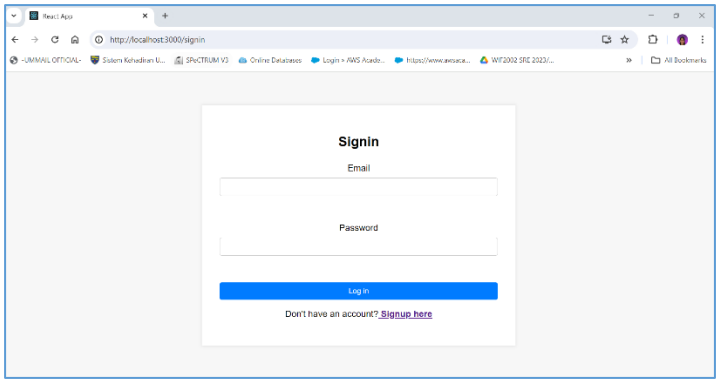
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Chiam\Desktop\LoginPageApplication\frontend> npm start

> frontend@0.1.0 start
> react-scripts start

Starting the development server...

One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.139:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

NOTE: To run the Frontend and Backend in one command only, you may refer to the following reference (Use either “concurrently” library or “npm-run-all”)

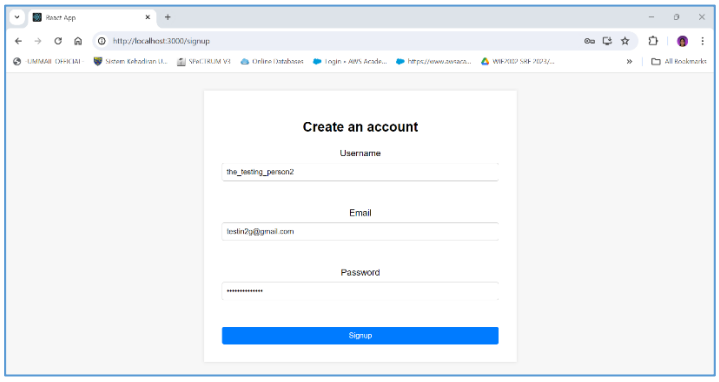
- https://medium.com/@rwijayabandu/how-to-run-frontend-and-backend-with-one-command-55d5f2ce952c

9. You should see this screen when you start the application: http://localhost:3000/

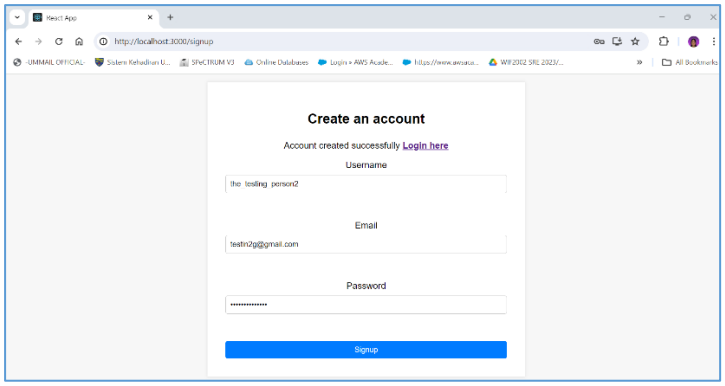
10. Test the application:
    a. Click the **sign in** link and you will see the **Signin** page



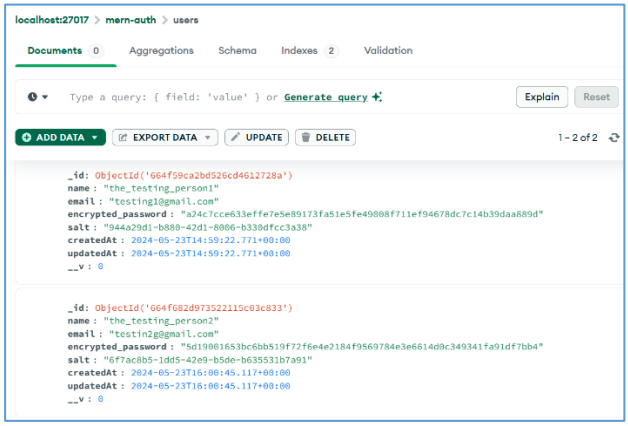
    b. Click the **Signup here** link to go to **Create an account**



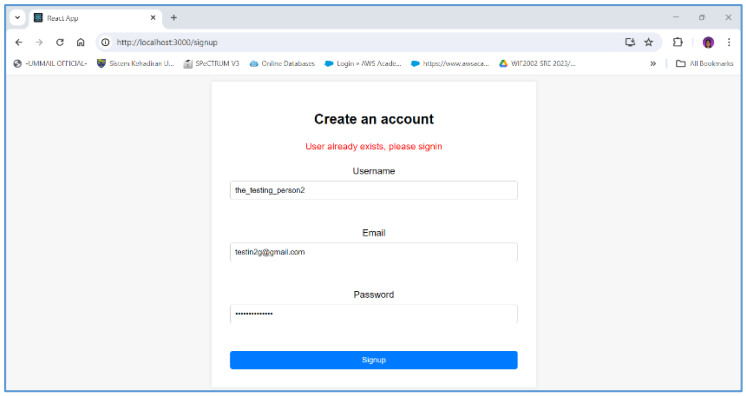
    c. You will see a message with "**Login here**" link if the account is created successfully



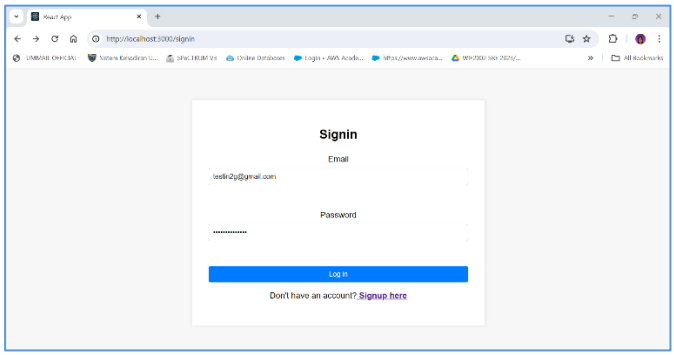
    d. A new user will be created in your MongoDB's database.



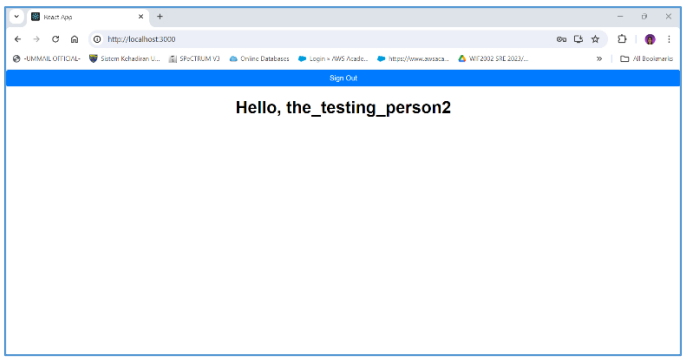
e. If the account exists, you will see the warning message: “User already exists, please signin”



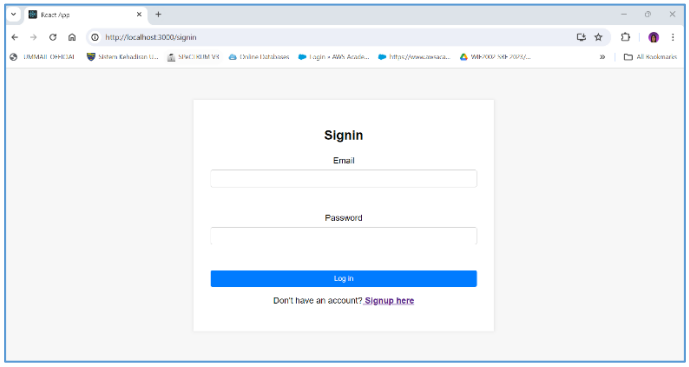
f. Click **Login here** link to go to Login page, enter email and password



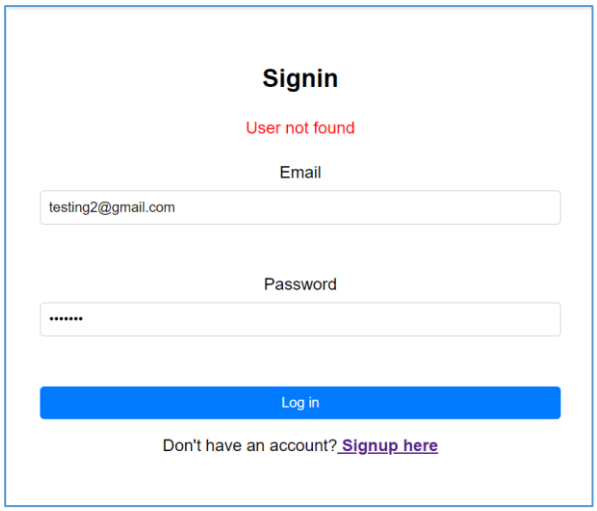
g. If login credentials are correct, you will be redirected to the **Dashboard** page with a **Sign Out** button.



h. Click **Sign Out** button and you will be redirected to **Signin** page.



i. If login failed, you will see a warning message “User not found”.



Task 2: Explore other authentication methods

There are other methods can be used for authentication in MERN stack applications.

You may explore:

1. Passport Js: https://www.passportjs.org/

The link below is explaining how to implement Google Social Authentication (OAuth) using Passport Js

- Social Authentication & Authorization in Node /Express Js Application using PassportJs: https://medium.com/@elijahechekwu/social-authentication-authorization-in-node-express-js-application-using-passportjs-part-1-db7fa622ea60

2. **Two-factor authentication**, or **2FA**, is a security mechanism that requires users to provide two different factors to verify their identity. These factors typically fall into three categories:

   - Something you know (e.g. Password)
   - Something you have (e.g. Mobile device)
   - Something you are (e.g. Biometrics)

   The link below is a step-by-step guide to implement two-factor authentication using Node.js:

   - https://levelup.gitconnected.com/go-beyond-passwords-secure-your-node-js-empire-with-two-factor-authentication-2fa-ff63c4b93112