

## 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
```

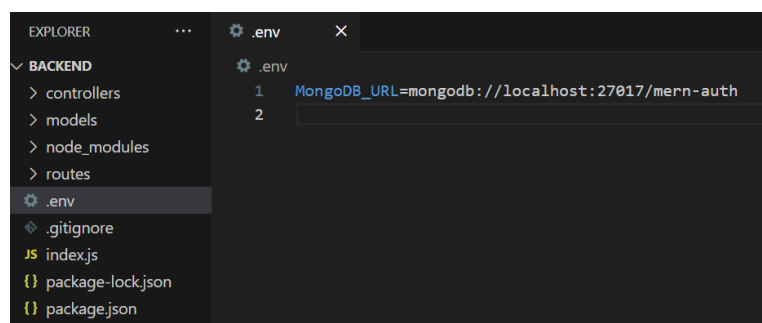
5. Create a **new database** in MongoDB, for example, you may name your database as '**mern-auth**' which has a collection called '**users**'.
6. After installation, create a **.env** file in your **backend** directory to store environment variables. Open your **.env** file and create 1 environment variable 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:**

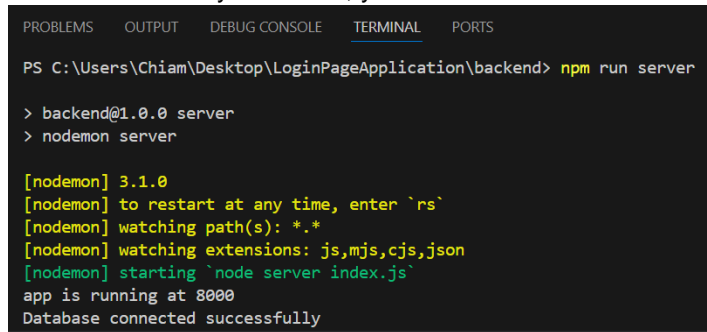


## WIF2003 Web Programming

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

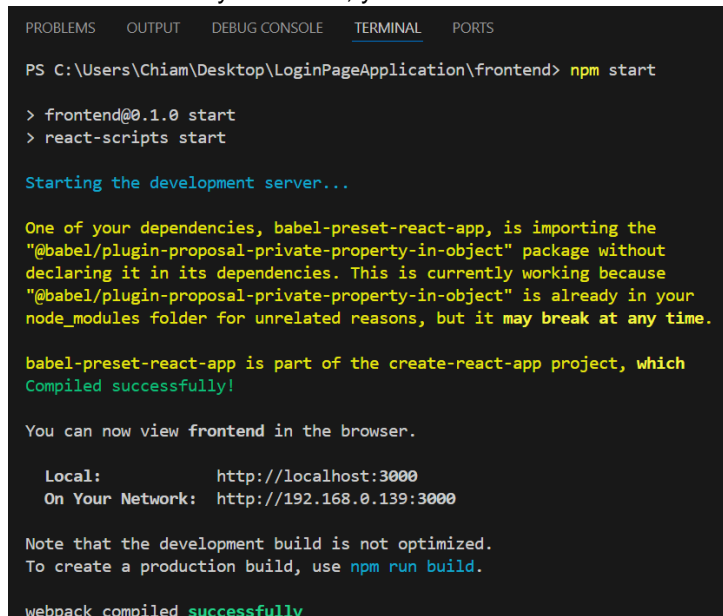
> backend@1.0.0 server
> nodemon server

[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

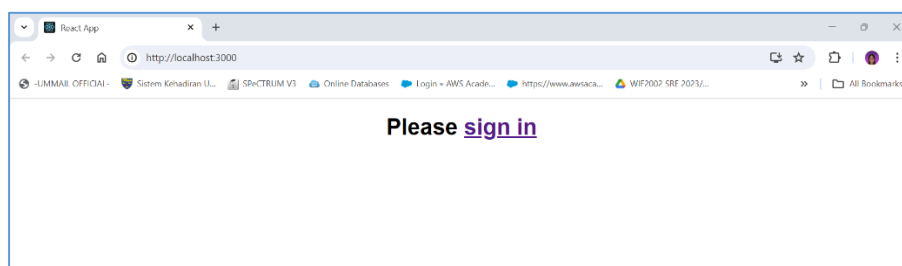
Note that the development build is not optimized.
To create a production build, use npm run build.

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/>



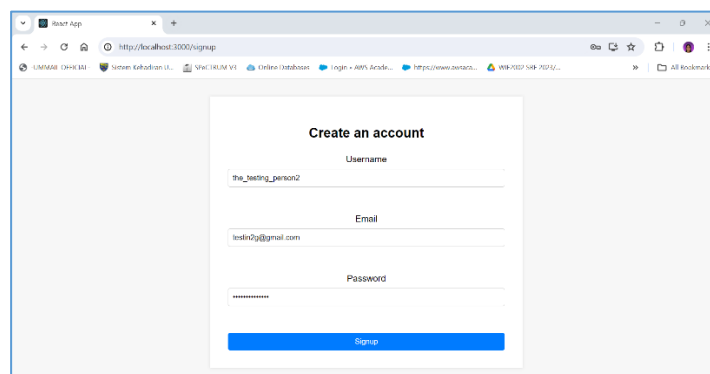
# WIF2003 Web Programming

## 10. Test the application:

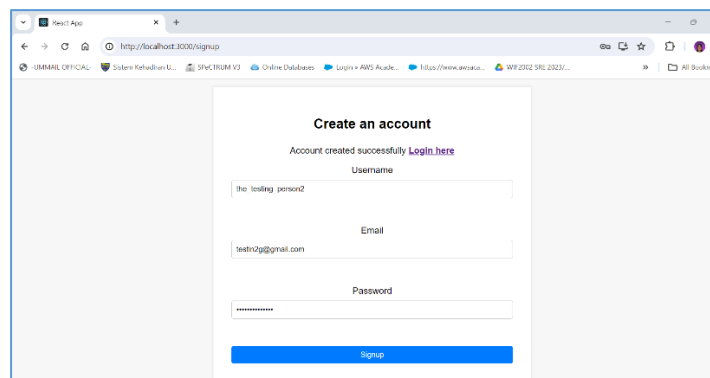
- a. Click the **sign in** link and you will see the **Signin** page

Signup here'." data-bbox="237 125 684 292"/>

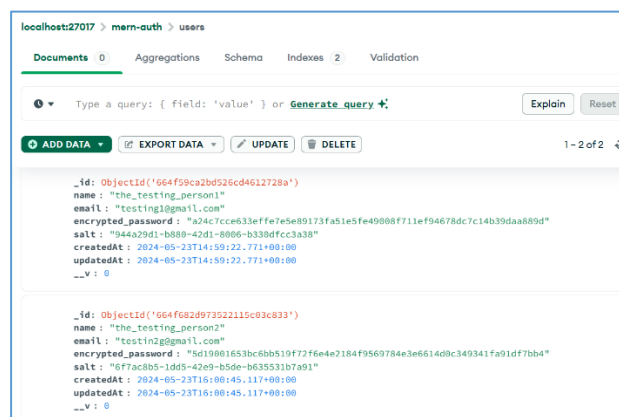
- b. Click the **Signup here** link to go to **Create an account**



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



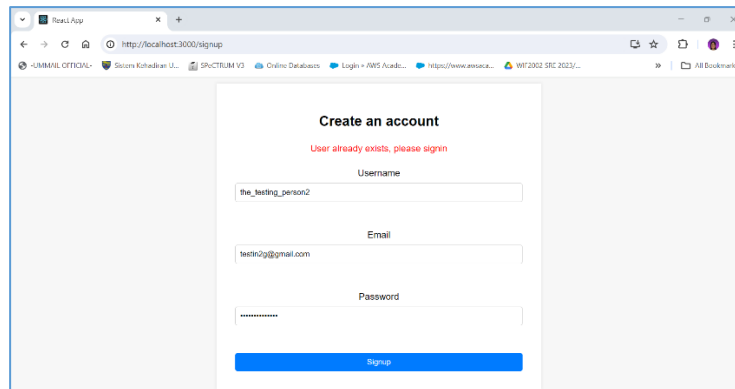
- d. A new user will be created in your MongoDB's database.



```
localhost:27017 > mern-auth > users
Documents 0 Aggregations Schema Indexes 2 Validation
Type a query: { field: 'value' } or Generate query
ADD DATA EXPORT DATA UPDATE DELETE 1-2 of 2
{
  "_id": ObjectId("664f59ca2bd526cd4612728a"),
  "name": "the_testing_person1",
  "email": "testing1@gmail.com",
  "encrypted_password": "a24c7cce633effe7e5e89173fa51e5fe49088f711ef94678dc7c14b39daa889d",
  "salt": "944a29d1-b088-42d1-6806-b330ffcc3a38",
  "createdAt": 2024-05-23T14:59:22.771+08:00,
  "updatedAt": 2024-05-23T14:59:22.771+08:00,
  "__v": 0
}
{
  "_id": ObjectId("664f682d97322115e03c833"),
  "name": "the_testing_person2",
  "email": "testing2@gmail.com",
  "encrypted_password": "5d19801653bc6bb519f72f6e4e2184f9569784e3e6614d8c349341fa91df7bb4",
  "salt": "6f7ac8b5-1dd5-42e9-b5de-b635531b7a91",
  "createdAt": 2024-05-23T16:08:45.117+08:00,
  "updatedAt": 2024-05-23T16:08:45.117+08:00,
  "__v": 0
}
```

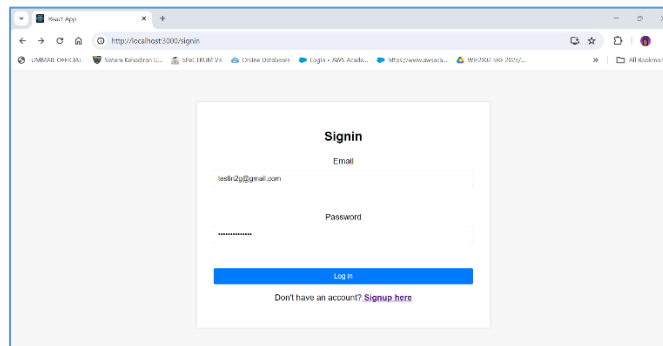
## WIF2003 Web Programming

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



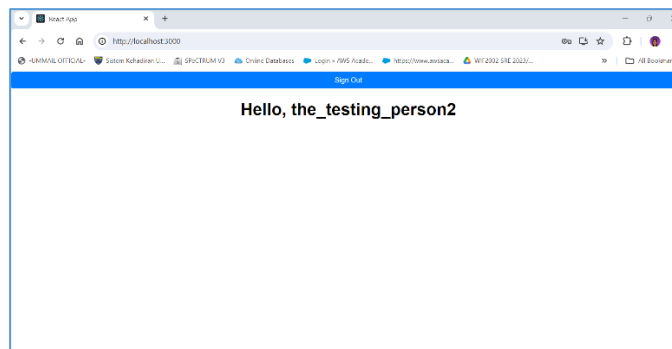
The screenshot shows a web browser window with the URL `http://localhost:3000/signup`. The page title is "Create an account". Below the title, a red error message reads "User already exists, please signin". There are three input fields: "Username" with the value "the\_testing\_person2", "Email" with the value "test123@gmail.com", and "Password" with masked characters. A blue "Signup" button is at the bottom.

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



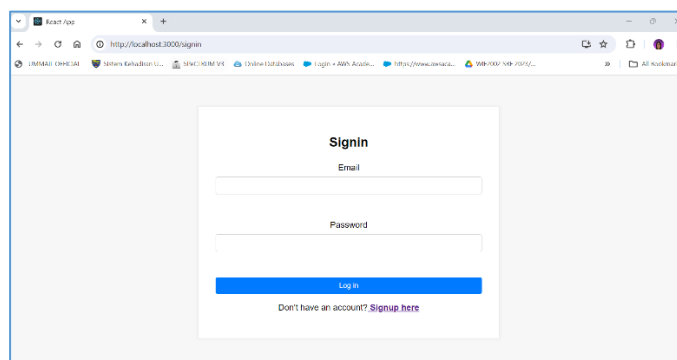
The screenshot shows a web browser window with the URL `http://localhost:3000/signin`. The page title is "Signin". There are two input fields: "Email" with the value "test123@gmail.com" and "Password" with masked characters. A blue "Log In" button is at the bottom. Below the button, a link reads "Don't have an account? [Signup here](#)".

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



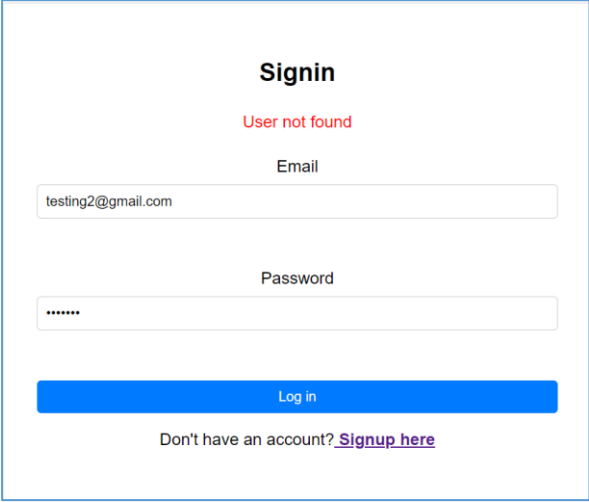
The screenshot shows a web browser window with the URL `http://localhost:3000`. The page has a blue header bar with a "Sign Out" button. Below the header, the text "Hello, the\_testing\_person2" is displayed.

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



The screenshot shows a web browser window with the URL `http://localhost:3000/signin`. The page title is "Signin". There are two input fields: "Email" and "Password", both empty. A blue "Log In" button is at the bottom. Below the button, a link reads "Don't have an account? [Signup here](#)".

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



The image shows a web form titled "Signin". Below the title, there is a red error message that says "User not found". The form has two input fields: "Email" and "Password". The "Email" field contains the text "testing2@gmail.com". The "Password" field is filled with asterisks. Below the input fields is a blue button labeled "Log in". At the bottom of the form, there is a link that says "Don't have an account? [Signup here](#)".

### 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>