Hands-on Lab: Set Up the React Environment with Backend Connectivity

Estimated time needed: **30** minutes

Introduction

In this lab, you will create a React project and establish database connectivity.

Objectives

After completing this lab, you will be able to:

- Create a React project
- Prepare the React project for backend (server side) connectivity
- Establish the database connectivity
- Test the website's server side and client side (front end) functionality

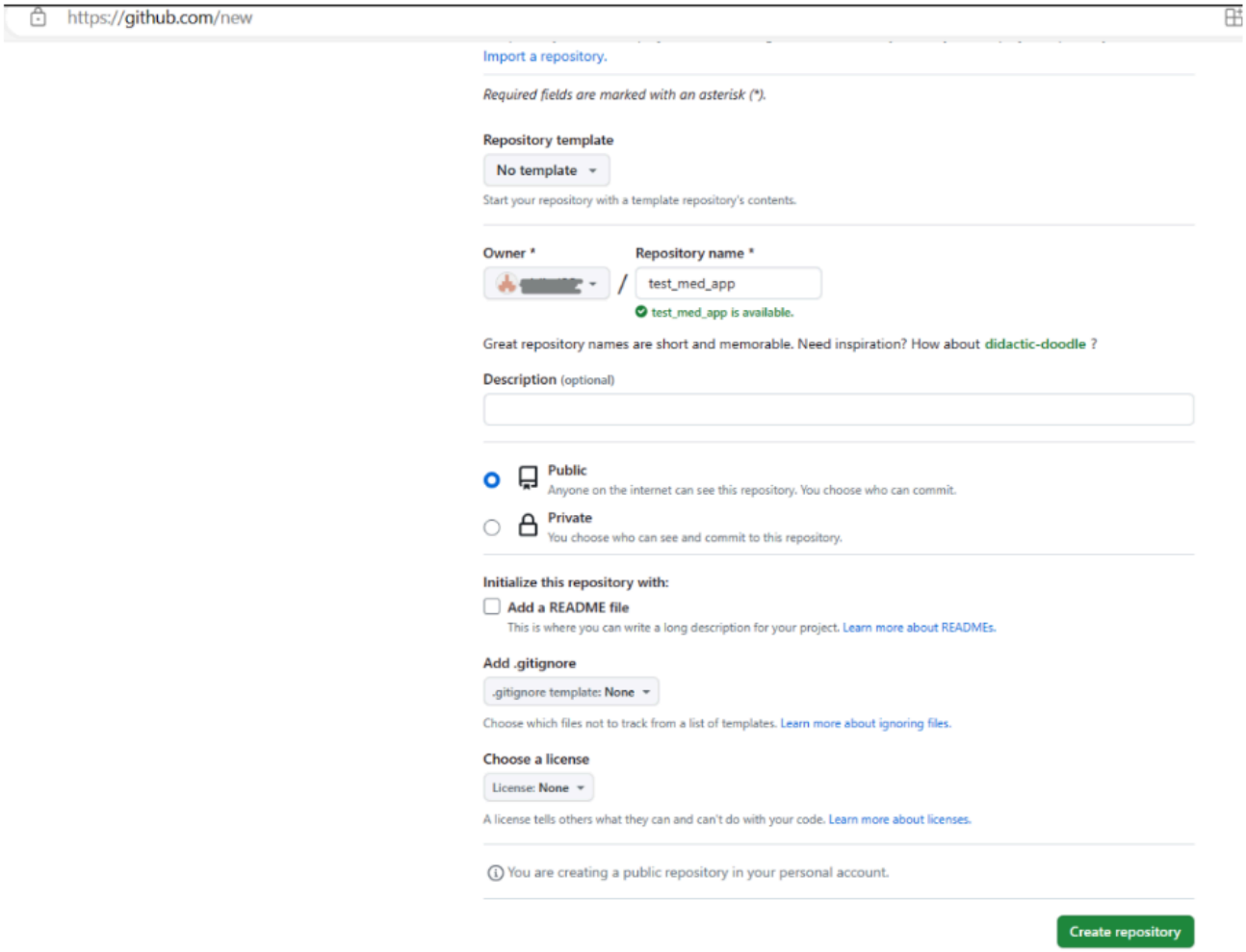
Prerequisites

- You must have completed the prerequisite courses, specially the **Getting Started with Git and GitHub** and **Developing Front-End Apps with React** courses.
- You must have completed the following labs:
  - Design Website Layouts
  - Create a GitHub Repository for your Project
  - Build Static Website Layouts

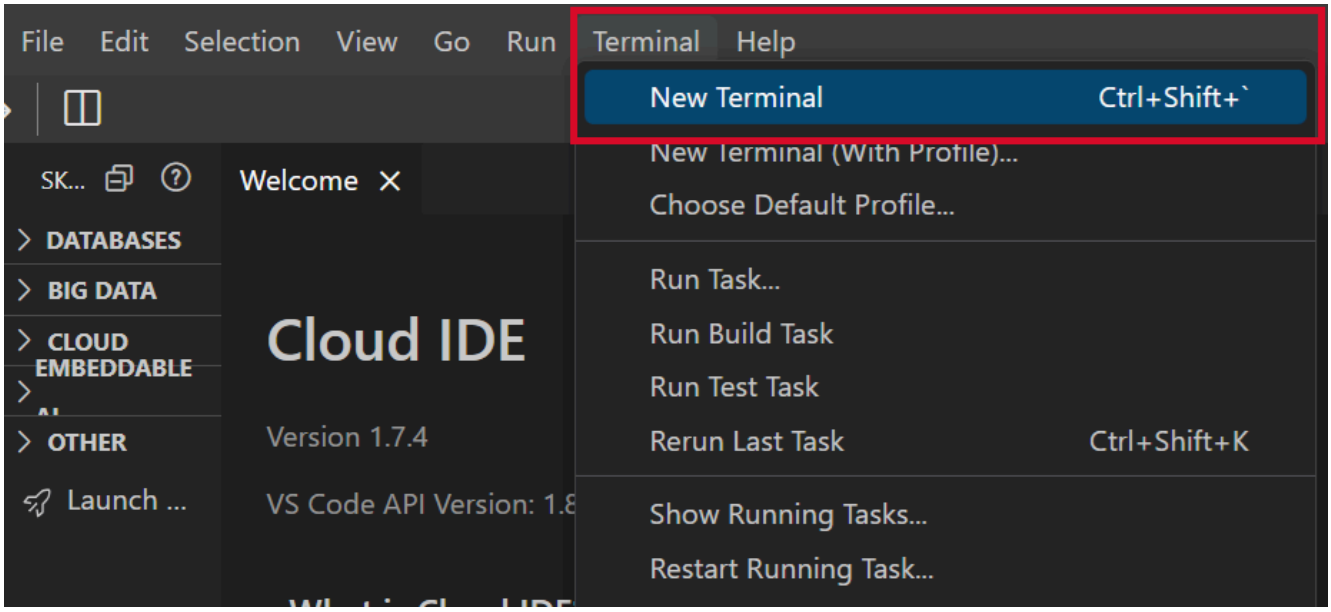
Exercise 1: Set up the React project

Let's create a React project and clone the repository.

1. Create a blank GitHub public repository in your GitHub account where you forked the repository in the previous exercise.



2. Open a new terminal in the Skills Network lab environment.



3. Create a React project.

```
theia@theiadocker-ritikaj:/home/project$ npx create-react-app test_med

Creating a new React app in /home/project/test_med.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1463 packages in 34s

242 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.

Installing template dependencies using npm...

added 69 packages, and changed 1 package in 5s

246 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...
```

▼ Click here for a **hint**.
npx create-react-app app-name

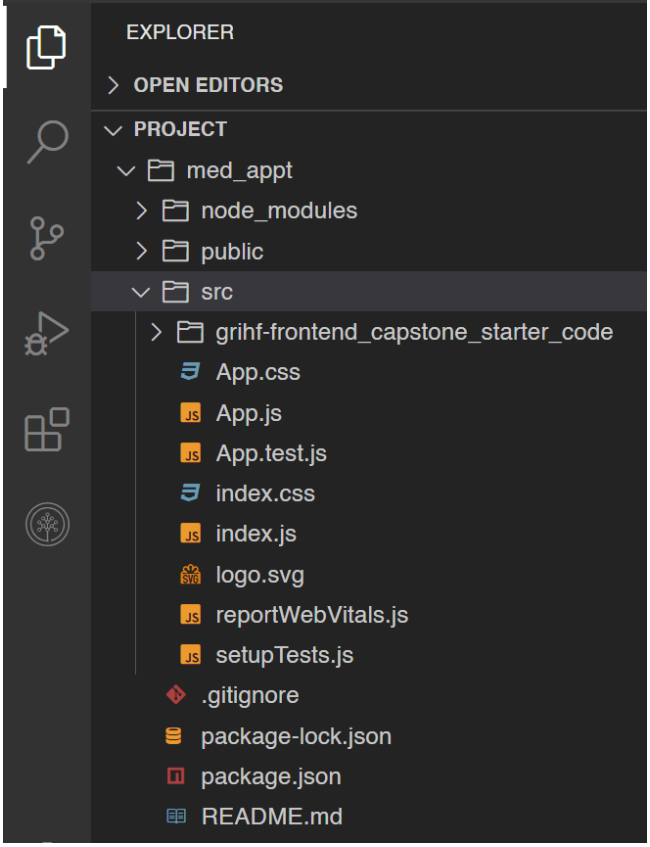
4. Navigate to the **src** folder in the React project.

▼ Click here for a **hint**.
cd app-name cd src

5. Clone your forked repository, **grihf-frontend_capstone_starter_code**, which contains the HTML and CSS files for the Navigation Bar, Sign Up form, and Login form inside the React project's **src** folder.

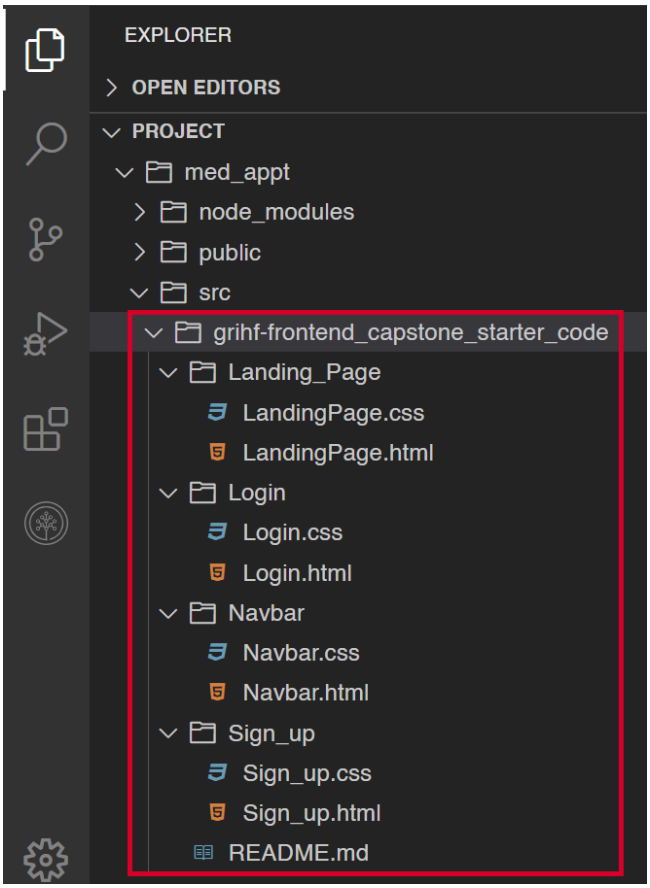
6. Validate that the cloning was completed properly in the React project folder. To do this:

    1. From the Explorer in the lab environment, navigate to the React project folder.
    2. Navigate to the **src** folder under the React project folder.



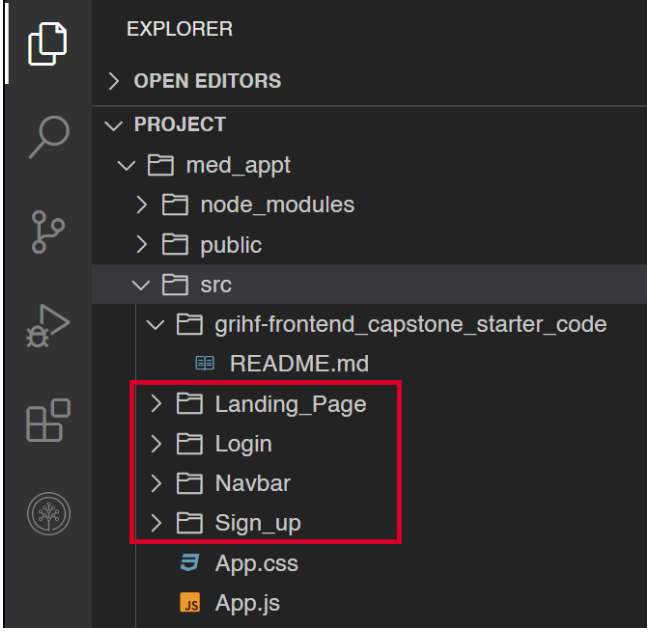
7. You should see the **grihf-frontend_capstone_starter_code** folder and several files, including **App.css**, **App.js**, **App.test.js**, **index.css**, **index.js**, **logo.svg**, **reportWebVitals.js**, and **setupTests.js**.

8. Ensure the complete folder structure from the previous labs is available under **grihf-frontend_capstone_starter_code**. The react project's **src** folder should contain folders and files for the Landing Page, the Navigation Bar, the Sign-Up form, and the Login form. Also, make sure that the README.md file that you created in a previous lab has been cloned.

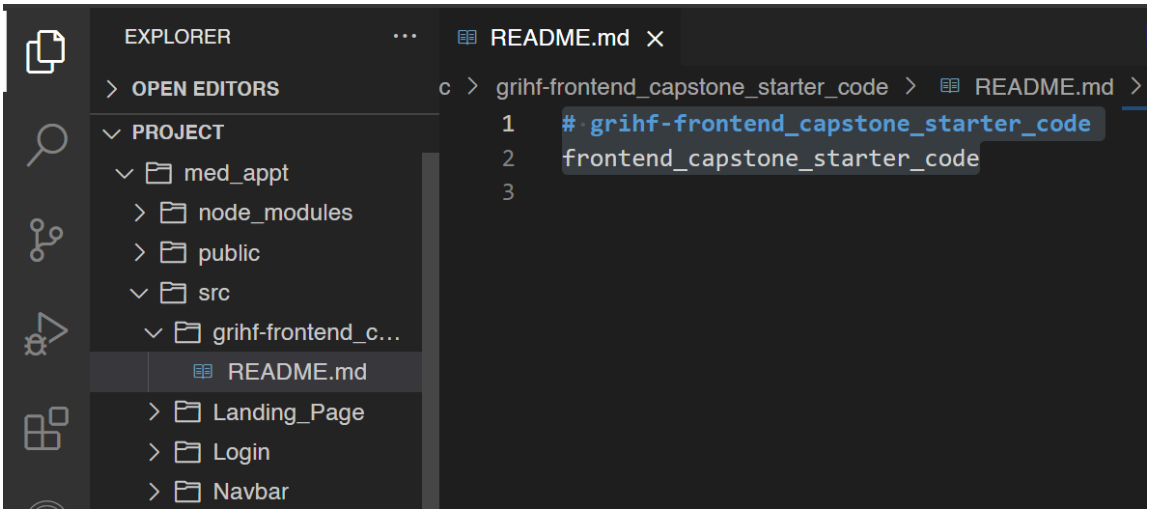


9. Move the folders available under the **grihf-frontend_capstone_starter_code** folder to be available directly under the **src** folder.

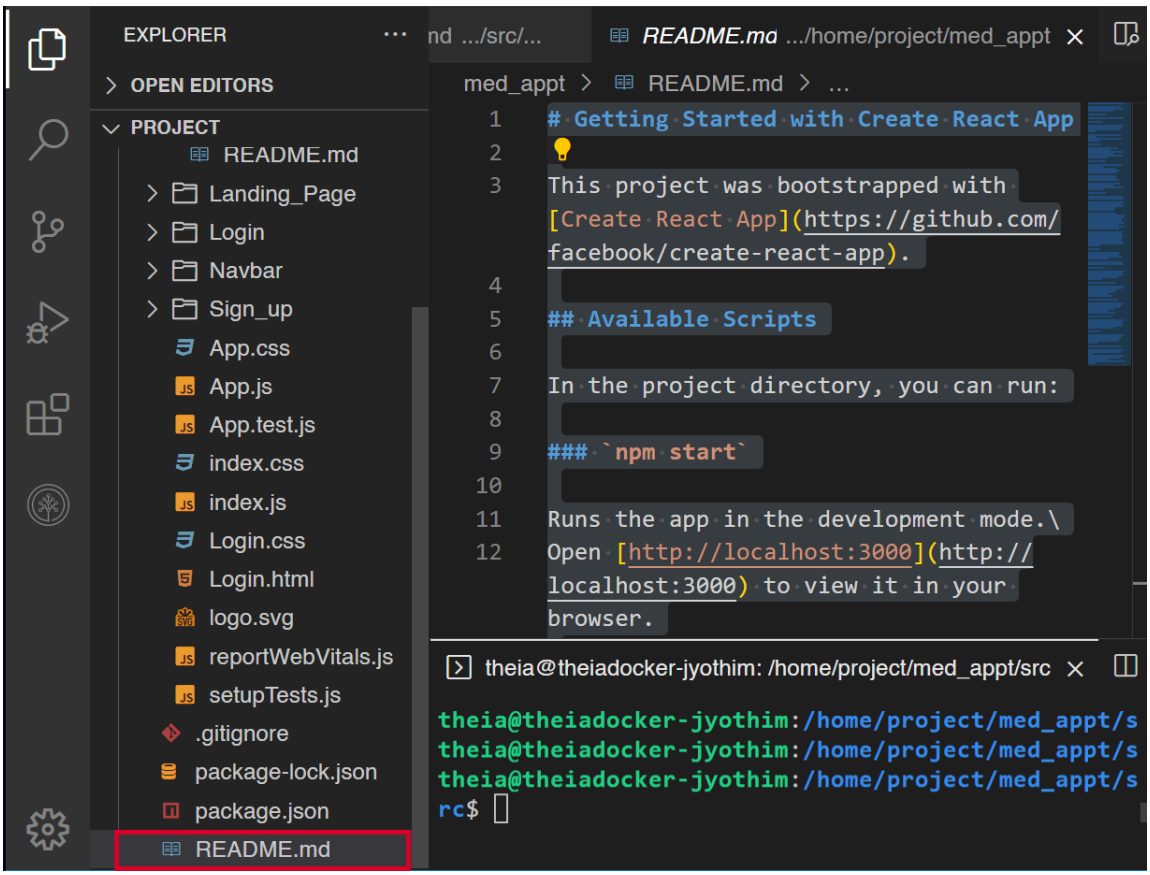
▼ Click here for a **hint**.
You can drag-and-drop the files via the Explorer.



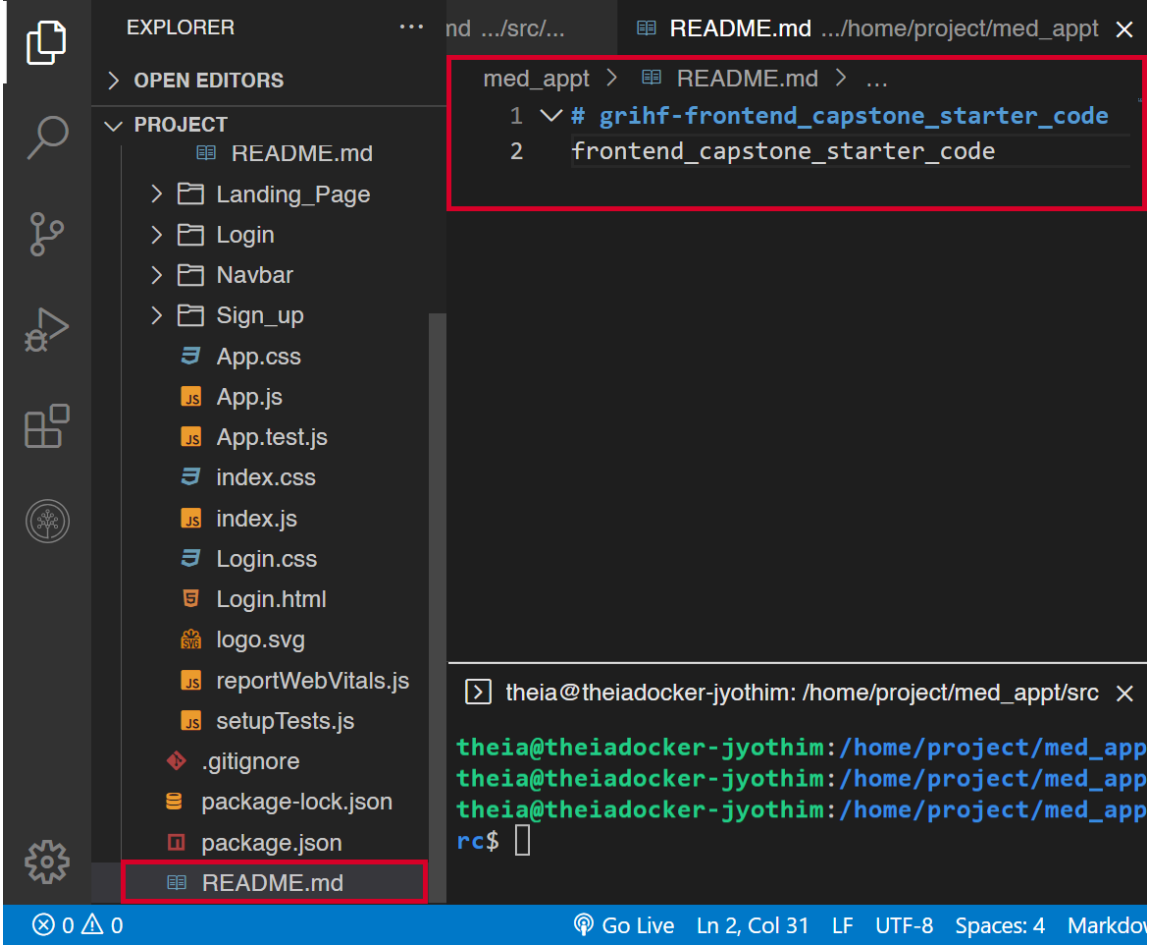
10. Copy the contents of the README.md file available under the **grihf-frontend_capstone_starter_code** folder.



11. Then, open the **README.md** file available in the **root** folder of the React project.



12. Replace the content with the content you copied in the previous step.



13. Delete the **grihf-frontend_capstone_starter_code** folder.

14. Navigate to index.html within public folder of react root folder and include these lines of code after title tag within `<head></head>`.

```
<link rel="preconnect" href="https://fonts.googleapis.com" />
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500;600;700;800;900&display=swap" rel="stylesheet">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-awesome.min.css">
```

15. Initialize the **React** project as a Git repository from the <app name> folder (for example, med_appt).

    ▼ Click here for a **hint**.
    git init

16. When prompted, enter the email and name as specified in the GitHub account.

    ▼ Click here for a **hint**.
    git config --global user.email "MY_NAME@example.com"

    git config --global user.name "name"

17. Add the files to your repository.

    ▼ Click here for a **hint**.
    git add --a

18. Commit the changes

    ▼ Click here for a **hint**.
    git commit -m "initial commit"

19. Push the changes using your Personal Access Token (PAT)

    ▼ Click here for a **hint**.
    git remote add origin2 <new react repository>

    git push origin2

Exercise 2: Prepare for server-side connectivity

1. After extracting to the local folder, drag and drop the necessary folders and files into the Theia lab environment.

2. To prepare for server-side connectivity, you must add the following folder and files to the React project's **src** folder:

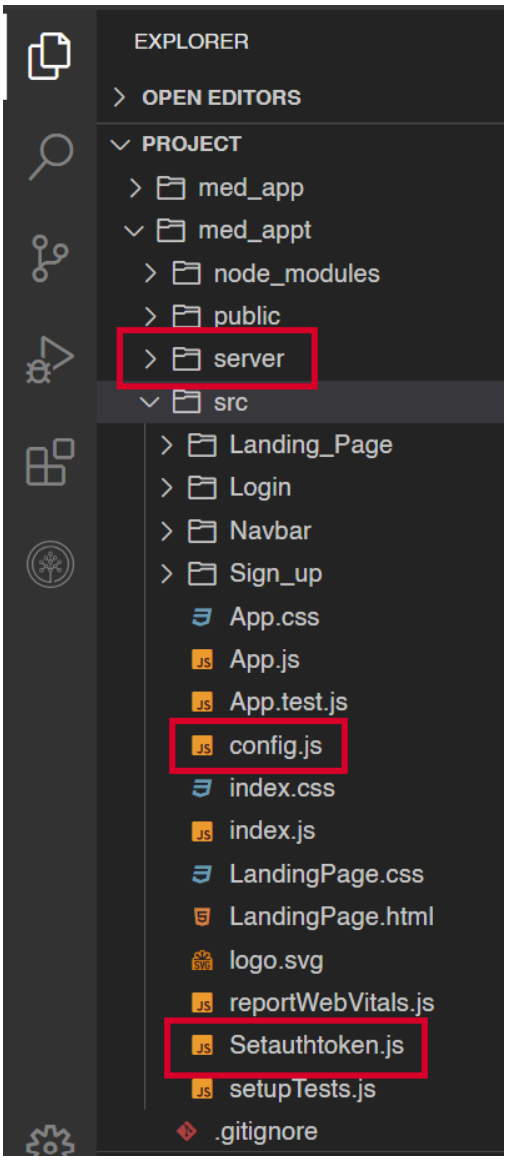
   - **server** folder
   - config.js
   - Setauthtoken.js

   To add the files:

   1. Download the **Server_Setup.zip** file using the link below and extract the files to a local folder.

Server_Setup

2. Add **config.js** and **Setauthtoken.js** in the **src** folder for react project. You must also add the **server** folder in the React project's root folder.



3. Open a new terminal in the lab environment and navigate to the **server** folder using `cd server` inside the **React** app folder.

4. Then, run `npm install` to install all the dependencies required for backend connectivity.

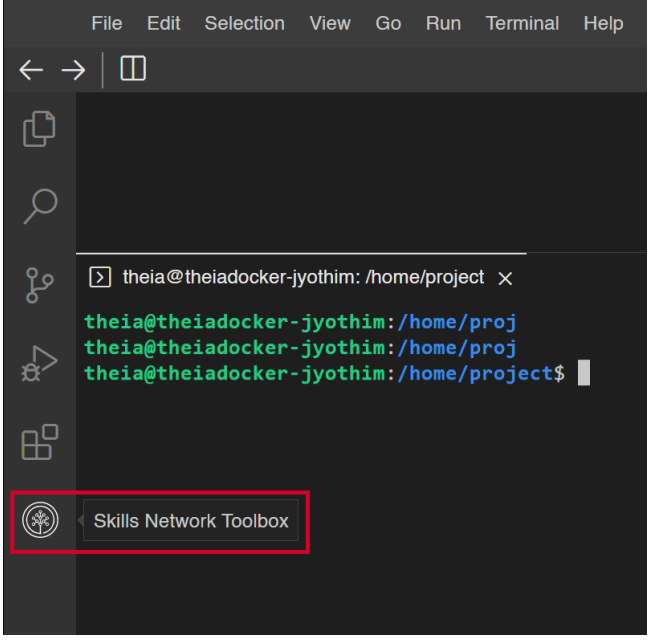
5. Perform `git add`, `git commit`, and `git push` commands to update changes into root repository of your React project's GitHub repository for proper code management.

   Refer to the **Capstone Project Reference: Git Commands** reading for details about Git command syntax and use relevant to the project.

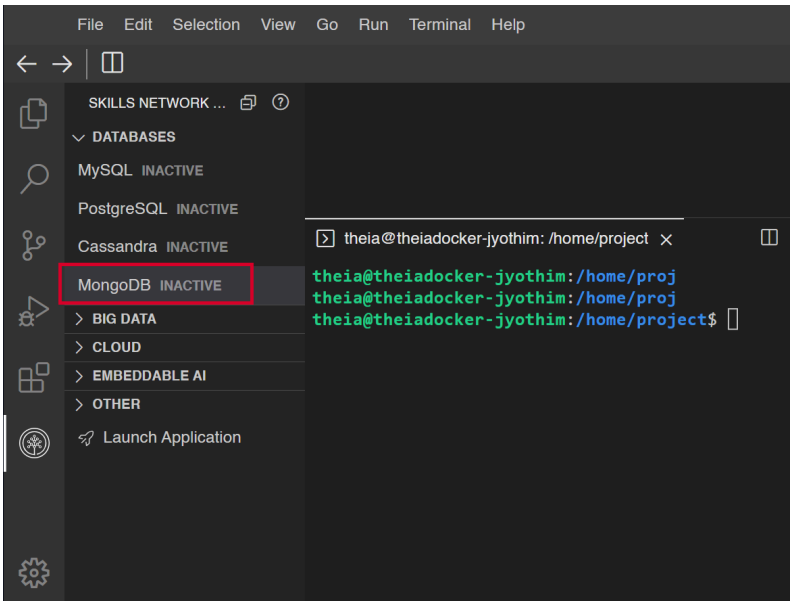
Exercise 3: Set up database connectivity

Let's set up the database connectivity to enable you to store the user information that is, for example, provided during sign up and used during login.

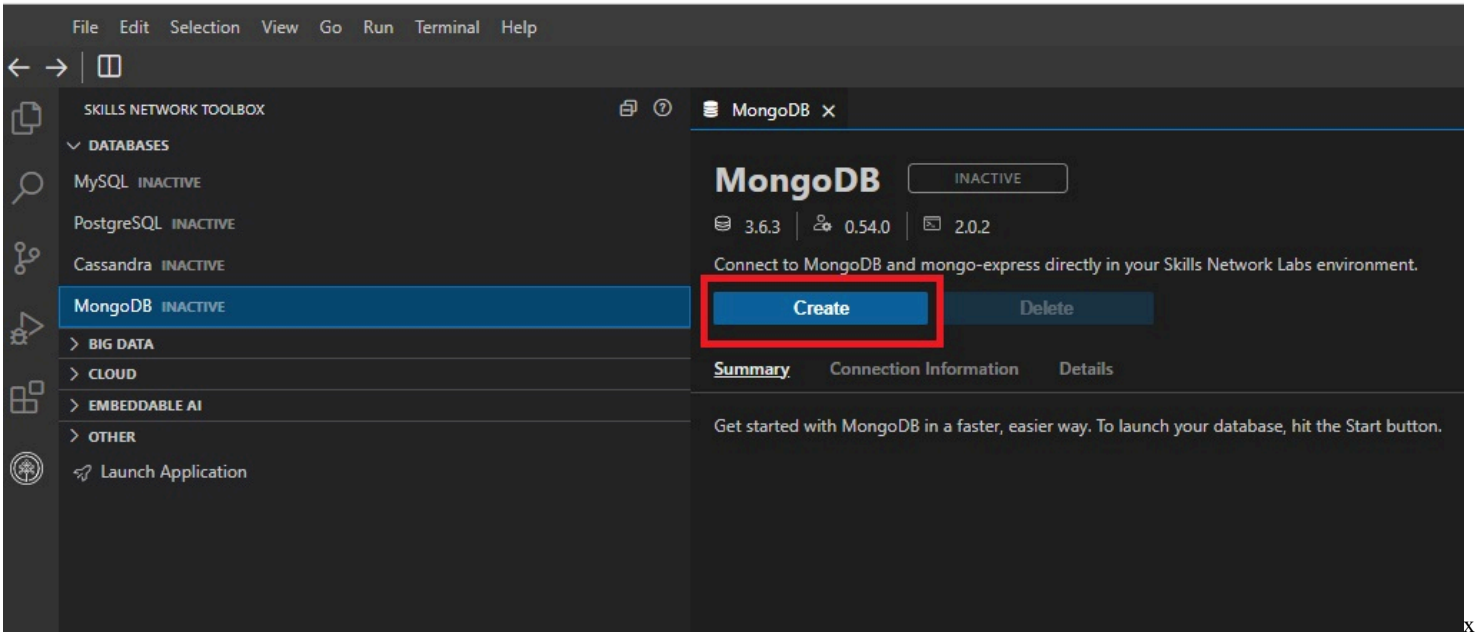
1. In the lab environment, click **Skill Network Toolbox**.



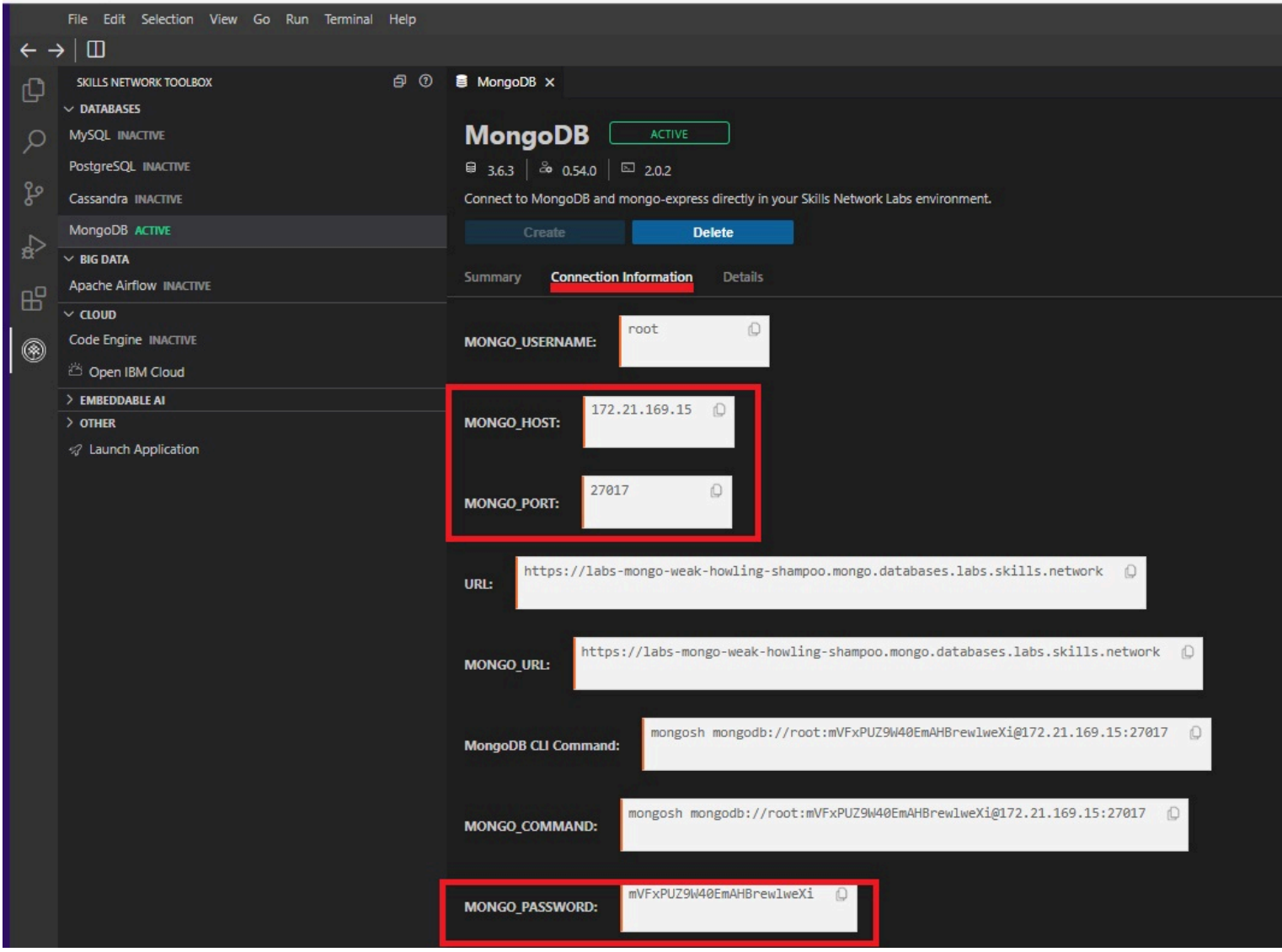
2. Click **mongoDB** from **DATABASES**.



3. Create the mongoDB service.



4. Copy the **MONGO_HOST**,**MONGO_PORT** ,**MONGO_PASSWORD** which is generated under connection information tab.



5. Navigate to the **server** folder in the React project's **root** folder.

6. Open the file named **db.js** and include that password at line number 2 for the const mongoURI after root: as displayed in the image below. Replace <your password>, MONGO_HOST and MONGO_PORT with the corresponding values.

Note: Every time that you start the MongoDB service, a new password and host ID is generated which needs to be updated in the db.js file.

7. Close the **db.js** file.

8. Perform `git add`, `git commit`, and `git push` commands on the React project's **root** folder to update changes into your React project's GitHub repository for proper code management.

Exercise 4: Test the website's server-side and client-side functionality

It's time to test the server-side and the client-side functionality in the lab environment. This will help you ensure both the server-side and client-side of the website launch successfully.

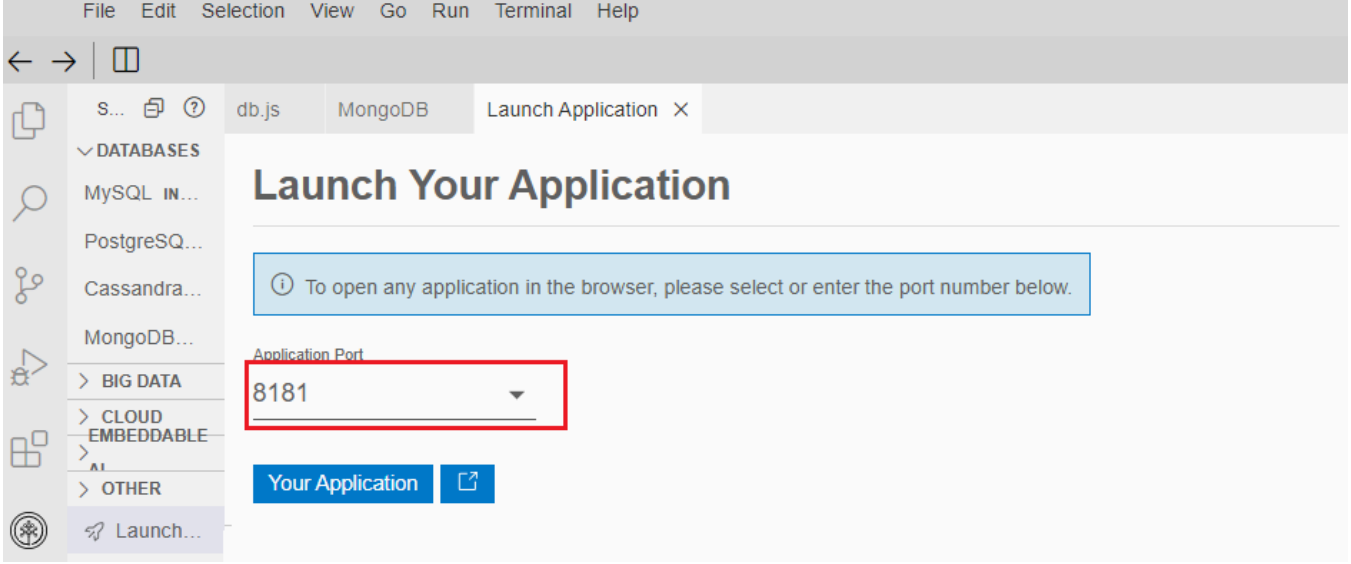
1. In a new terminal, navigate to the React project's **server** folder.

2. Then, execute the command `node index`.
   If successful, you will see the following message:

   Server is running on port localhost:8181
   Connected to Mongo Successfully

3. Click **Skills Network Toolbox**. Then, click **Launch Application**.

4. Specify the Application port as **8181** to launch server side. **8181** is the port number for the server side.



5. Then, click **Launch**  to launch the website's server-side.

6. The website will open in your default browser and display a message **Hello World!**.
   This indicates that the website's server side is working properly.

7. Copy the URL for the server side. Then, paste it in the **config.js** file replacing <add your server side url> under src folder.

   Note: Make sure to replace the URL both before and after the colon(:) to avoid compile time errors.

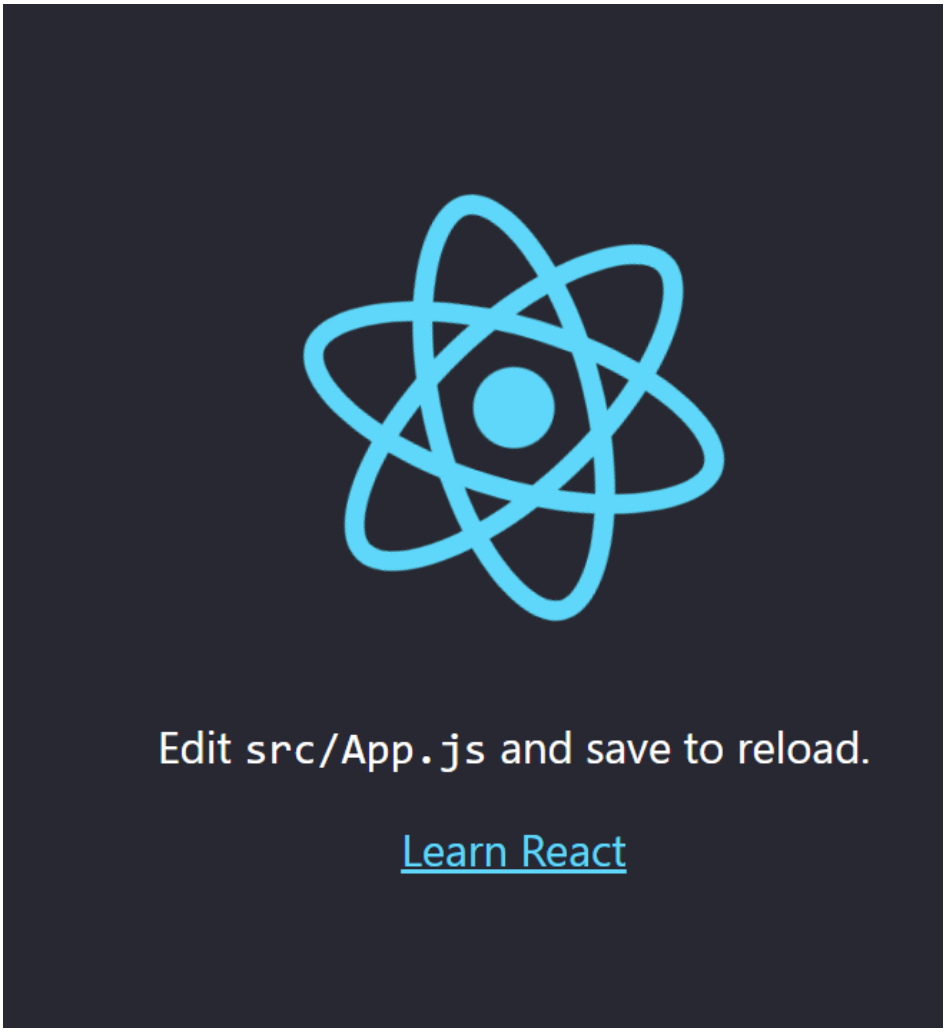
```
export const API_URL = window.location.hostname === "localhost" ? "<add your server side url>" :"<add your server side url>"
```

8. Open a new terminal. Navigate to the React project's **root** folder.

9. Execute `npm start`.

10. To launch the React project's client side, use port **3000**.

11. When you launch the website's client side, the default output of APP.js is displayed.



This concludes the lab on setting up the React project and establish database connectivity.

Note about data management and persistence

To ensure the proper management and persistence of your data in a GitHub repository, it is crucial to follow a few essential steps:

- **Regular Updates:** Whenever you make changes or add new components to your project, it is essential to add, commit, and push the updates to your GitHub repository. This ensures that your latest work is safely stored and accessible to collaborators.
- **Session Persistence:** During an active session, your data remains accessible. However, it's important to note that if your session expires or you log out, you will need to clone the repository again to resume work.
- **Ignoring node modules:** When pushing data to GitHub, it's best practice to exclude the node modules folder from both your server and client directories. This folder contains external dependencies and can be quite large, making the repository heavy and slowing down the process. By adding it to the .gitignore file, you prevent it from being pushed to the repository, keeping your commits cleaner and more focused.

By adhering to these guidelines, you can maintain a well-organized and efficient GitHub repository, ensuring that your work is securely stored and easily accessible to you and your collaborators.

Author(s)

Richa Arora