**Hospital Management Application**

Technologies used:

Frontend: **Visual code as ide for Ui**

Backend: **IntelliJ Ide for creating spring boot rest Api’s**

Database **: MySQL and MySQL workbench**

* **A Quick review how this project Works:** **MEDSWISS** system is a software application or platform designed to facilitate the booking of appointments to meet to related doctor to your problem.
* **Admin** is allowed to use the **crud** **operation** like **Adding , Updating , Deleting and Viewing** the **doctor and patient details** .

**Setting Up Backend:**

**Downloading Ide:**

* First we need and ide to setup and to execute the code.
* Here we are using **IntelliJ** **idea** from the **JetBrains** as an ide
* You can get the ide from the given url <https://www.jetbrains.com/idea/download/> click the link and it will redirects you to the page

**After Installation :**

* Open the intellij idea and click on button name new project the ide will automatically generate a the basic project structure for you.
* After the click on build menu to check whether the application is building without any errors.
* This is the basic set up for the ide.

**Some other important steps to know before getting into the project:**

* **Configure Project Settings (if necessary):** Depending on the project setup ,you might need to configure project SDK , language level ,project structure etc,.
* **Install Dependencies:** If the project relies on external dependencies managed by build tools like Maven or Gradle, IntelliJ will typically recognize them and prompt you to download them automatically. Otherwise, you might need to manually configure dependencies.

**Spring Boot Project:**

* After the installation process go spring initializer (<https://start.spring.io/>) website to download the basic spring project with dependencies you need for this project.

**Dependencies Required for this project :**

1. Spring boot JDBC
2. Spring web
3. Mysql connector(or any database you like)
4. Lombok
5. Spring Jpa
6. Spring boot validation
7. Spring Security
8. Jwt

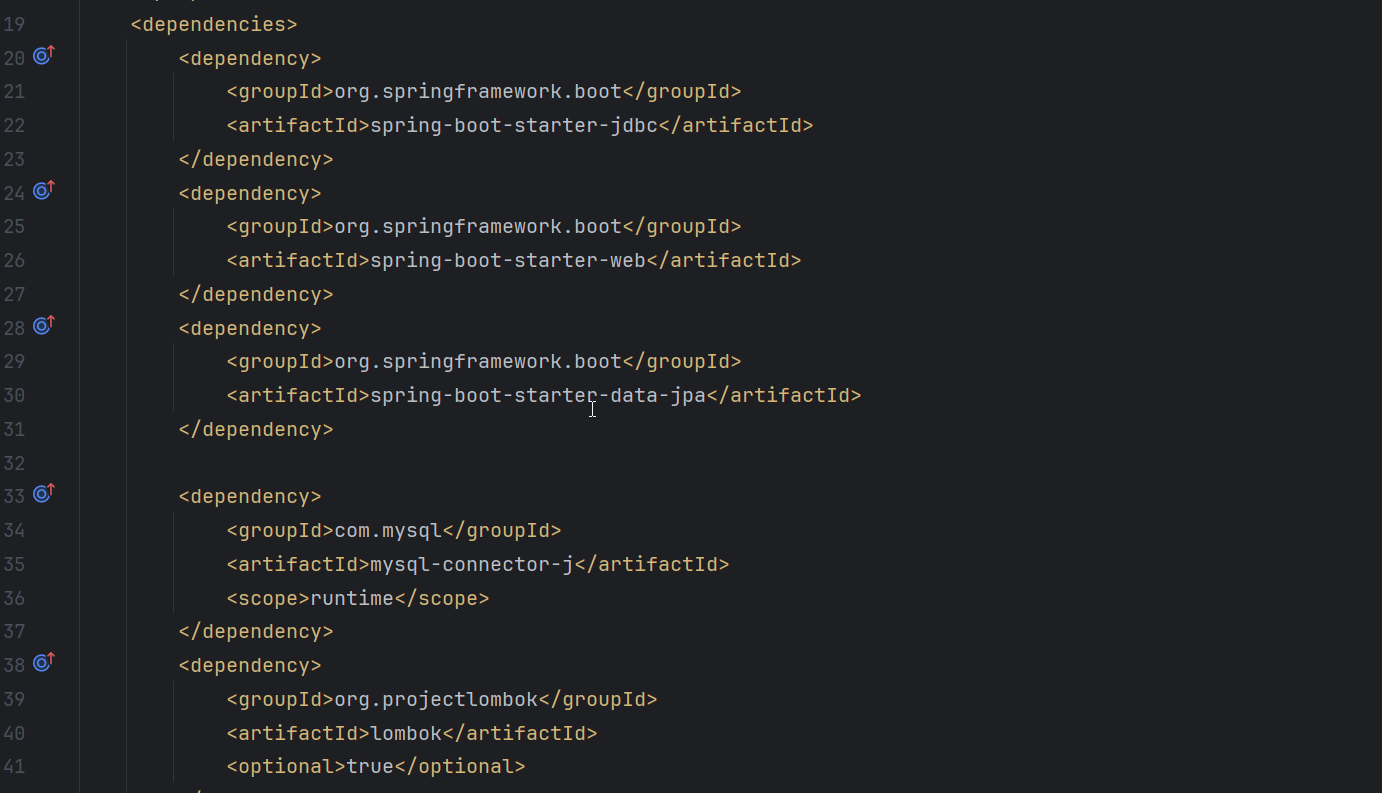
**Setting Up JPA (Java Persistence API) for Data Model Management:**

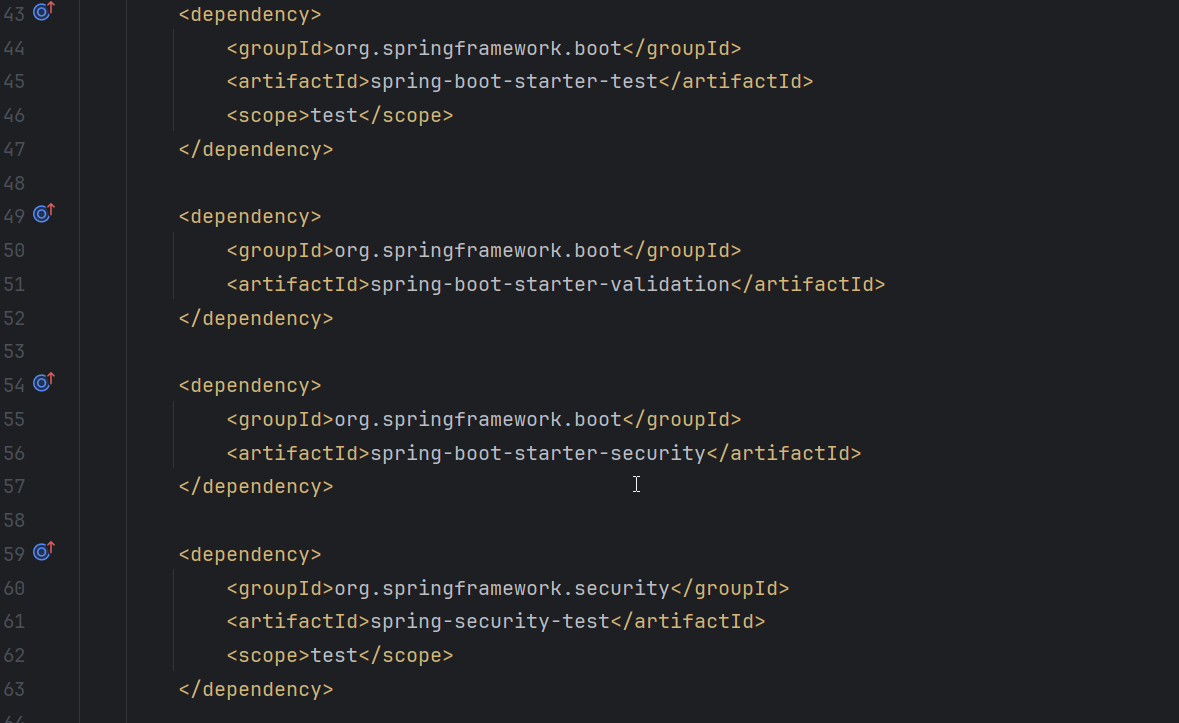
**What is JPA?**

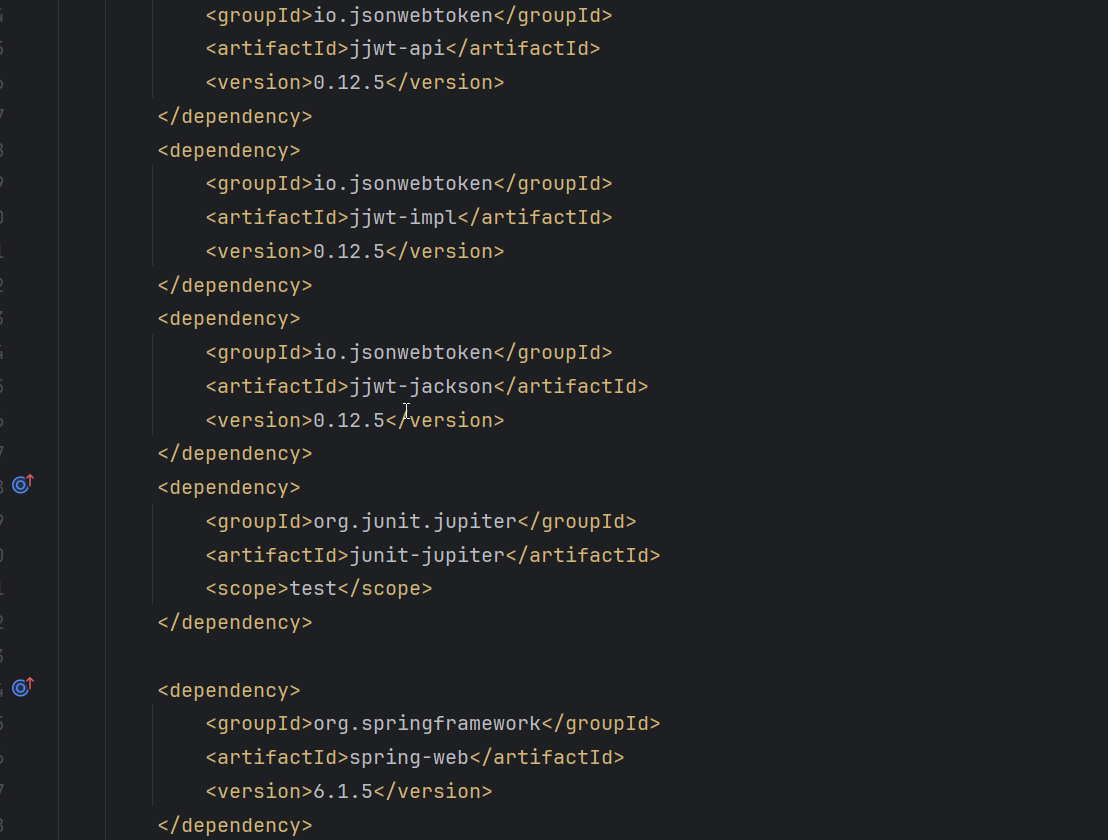
JPA (Java Persistence API) is a standard specification for managing relational data in Java applications. It provides an abstraction layer over JDBC (Java Database Connectivity) and allows developers to interact with databases using object-oriented concepts.

* **Steps to set up jpa:**

1. Add Dependencies:



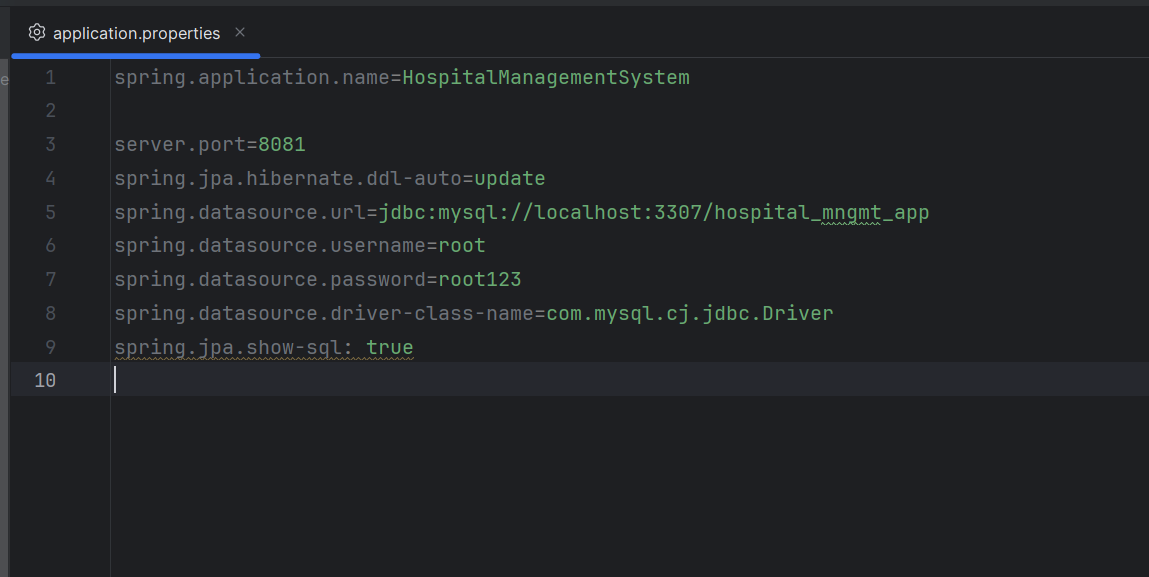




1. **Configure Data Source:**

* Define your database connection properties in application.properties.
* Specify the database URL, username, password, and other relevant settings.

**Application.properties:**



1. **Create Entity Classes:**

* An entity class represents a table in the database. Annotate your Java classes with @Entity to define them as JPA entities.
* Define fields, relationships, and annotations such as @Id, @GeneratedValue, and @Column.

1. **Repository Interfaces:**

* Create repository interfaces by extending JpaRepository<T, ID>.
* These interfaces provide CRUD operations (Create, Read, Update, Delete) for your entities

1. **Service Layer:**

* Implement business logic in service classes.
* Use repository methods to interact with the database.

**Implementing RestFul APIs for Hospital Data:**

There are 4 different classes for this project.

* Doctor
* Patient
* Appointment
* Medication

Only the admin can do the crud operations for this specific object.

**CRUD Operations Of Doctor: (Only for admin)**

1. **Create(post):**

* Create an endpoint as per your wish my endpoint for this post method is(“/api/doctor”) to add new doctor.
* Accept doctor data in the request body (JSON or form data).
* Save the data to the database using JPA repository.

1. **Read(GET):**

* Create an endpoint (“/api/doctor/id”) to retrieve doctor details by Id
* Implement a list endpoint (“/api/doctor”) to fetch all doctors.

1. **Update(put):**

* Create an endpoint (“/api/doctor/id”) to update doctor details by Id
* Accept updated data in the request body.
* Use JPA to update the corresponding record.

1. **Delete:**

* Implement an endpoint (“/api/doctor/id”) to delete an doctor by ID.
* Remove the record from the database.

**CRUD Operations Of Patient: (Only for admin)**

1. **Create(post):**

* Create an endpoint as per your wish my endpoint for this post method is(“/api/patient”) to add new patient.
* Accept patient data in the request body (JSON or form data).
* Save the data to the database using JPA repository.

1. **Read(GET):**

* Create an endpoint (“/api/patient/id”) to retrieve patient details by Id
* Implement a list endpoint (“/api/patient”) to fetch all patients.

1. **Update(put):**

* Create an endpoint (“/api/patient/id”) to update patient details by Id
* Accept updated data in the request body.
* Use JPA to update the corresponding record.

1. **Delete:**

* Implement an endpoint (“/api/patient id”) to delete an patient by ID.
* Remove the record from the database.

**CRUD Operations Of Appointment: (Only for admin)**

1. **Create(post):**

* Create an endpoint as per your wish my endpoint for this post method is(“/api/appointment”) to add new appointments.
* Accept appointment data in the request body (JSON or form data).
* Save the data to the database using JPA repository.

1. **Read(GET):**

* Create an endpoint (“/api/appointment/id”) to retrieve passenger details by Id
* Implement a list endpoint (“/api/appointment”) to fetch all appointments.

1. **Update(put):**

* Create an endpoint (“/api/appointment/id”) to update appointment details by Id
* Accept updated data in the request body.
* Use JPA to update the corresponding record.

1. **Delete:**

* Implement an endpoint (“/api/appointment/id”) to delete an appointment by ID.
* Remove the record from the database.

**CRUD Operations Of Medication: (Only for admin)**

1. **Create(post):**

* Create an endpoint as per your wish my endpoint for this post method is(“/api/medication”) to add new medications.
* Accept appointment data in the request body (JSON or form data).
* Save the data to the database using JPA repository.

1. **Read(GET):**

* Create an endpoint (“/api/medication/id”) to retrieve medication details by Id
* Implement a list endpoint (“/api/medication”) to fetch all medications.

1. **Update(put):**

* Create an endpoint (“/api/medication/id”) to update medication details by Id
* Accept updated data in the request body.
* Use JPA to update the corresponding record.

1. **Delete:**

* Implement an endpoint (“/api/medication/id”) to delete an medication by ID.
* Remove the record from the database.

**Utilizing Spring Security for Authentication and Authorization:**

Here we are using Jwt Token for the spring security.

**To login using a JWT (JSON Web Token), these are steps:**

1. **Authentication:** Authenticate the user using their credentials (username/email and password).
2. **Generate Token:** Upon successful authentication, generate a JWT containing relevant user information and any additional data necessary (e.g., user roles, permissions, etc.).
3. **Token Sending:** Send the JWT to the client (usually in the response body or headers).
4. **Client Storage:** Store the JWT securely on the client-side, typically in local storage or a cookie.
5. **Token Sending with Subsequent Requests:** Include the JWT in the headers of subsequent requests to the server for protected resources.
6. **Token Validation:** On the server-side, validate the JWT for each incoming request to ensure that it has not been tampered with and that it is still valid.
7. **Access Control:** Use the information within the JWT to determine whether the user has permission to access the requested resource.

***Dependencies for the jwt tokens are given prior to this topic.***

**Implementing Validation and Error Handling:**

**Validation:**

1. **Request Validation:**

* Validate input data (e.g., required fields, data formats) using annotations like (@Valid) and custom validators.
* Handle validation errors gracefully.

1. **Entity Validation:**

* Use JPA validation annotations (e.g., @NotEmpty, @Email) to validate entity fields.

**API Endpoints of the backend project:**

**Doctor:**

**Post:** [http://localhost:8081/api/doctor](http://localhost:8081/api/doctor%20)

**Get:** [http://localhost:8081/api/doctor](http://localhost:8081/api/doctor%20)

**GetById:** <http://localhost:8081/api/doctor/id>

**Put:** <http://localhost:8081/api/doctor/id>

**Delete:** <http://localhost:8081/api/doctor/id>

**Login:** <http://localhost:8081/api/auth/login>

**Register:** <http://localhost:8081/api/auth/register>

**Patient:**

**Post:** [http://localhost:8081/api/patient](http://localhost:8081/api/patient%20)

**Get:** [http://localhost:8081/api/patient](http://localhost:8081/api/patient%20)

**GetById:** <http://localhost:8081/api/patient/id>

**Put:** <http://localhost:8081/api/patient/id>

**Delete:** <http://localhost:8081/api/patient/id>

**Login:** <http://localhost:8081/api/auth/login>

**Register:** <http://localhost:8081/api/auth/register>

**Appointment:**

**Post:** [http://localhost:8081/api/appointment](http://localhost:8081/api/appointment%20)

**Get:** [http://localhost:8081/api/appointment](http://localhost:8081/api/appointment%20)

**GetById:** <http://localhost:8081/api/appointment/id>

**Put:** <http://localhost:8081/api/appointment/id>

**Delete:** <http://localhost:8081/api/appointment/id>

**Login:** <http://localhost:8081/api/auth/login>

**Register:** <http://localhost:8081/api/auth/register>

**Medication:**

**Post:** [http://localhost:8081/api/medication](http://localhost:8081/api/medication%20)

**Get:** [http://localhost:8081/api/medication](http://localhost:8081/api/medication%20)

**GetById:** <http://localhost:8081/api/medication/id>

**Put:** <http://localhost:8081/api/medication/id>

**Delete:** <http://localhost:8081/api/medication/id>

**Login:** <http://localhost:8081/api/auth/login>

**Register:** <http://localhost:8081/api/auth/register>

**Writing Unit Tests**

1. **Test Framework Selection:**

Choose a testing framework that integrates well with your project. For Java applications, popular choices include Junit.

1. **Test Class Creation:**

Create a test class for each class you want to test. For example, if you have an Doctor class you have create DoctorServiceTest class DoctorRepository class , DoctorController class

**Test Methods:**

* Write test methods within your test class.
* Each method should test a specific behavior or functionality of your code.

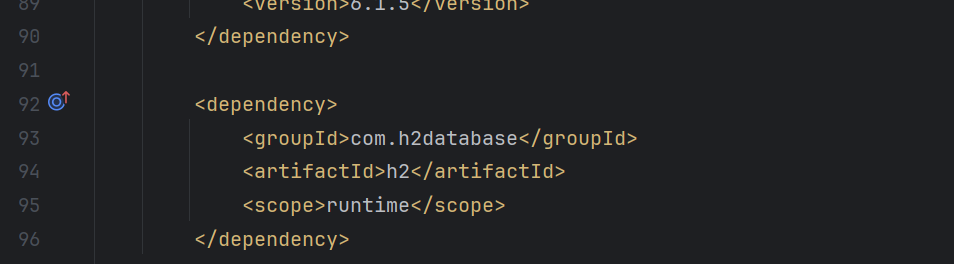
1. **Arrange-Act-Assert (AAA) Pattern:**

Follow the AAA pattern in your test methods:

* Arrange: Set up the necessary preconditions (create objects, initialize variables, etc.).
* Act: Invoke the method you want to test.
* Assert: Verify that the expected outcome matches the actual result**.**

1. **Use Assertions:**

Use assertions (e.g., assertThat) to validate the correctness of your code.

****

**Download From GitHub:**

Steps to download frontend project from GitHub in VS code

1. You can do this by pressing Ctrl + Shift + P (Windows/Linux) or Cmd + Shift + P (Mac). This will open a command palette at the top of the editor.
2. Clone Repository: In the command palette, type Git: Clone and select the option that appears. Alternatively, you can press Ctrl + Shift + P (Windows/Linux) or Cmd + Shift + P (Mac) and type Git: Clone and press Enter.
3. You will be prompted to enter the URL of the GitHub repository you want to clone. Paste the URL and press Enter.
4. Choose the local directory where you want to clone the repository. Once selected, press Enter.
5. If the repository is private or requires authentication, you may be prompted to provide your GitHub username and password or personal access token. Enter the required credentials and press Enter.
6. VSCode will start cloning the repository from GitHub to your local machine. You will see a progress indicator in the status bar at the bottom of the editor.
7. Once the cloning process is complete, VSCode will open the cloned repository in a new window automatically.
8. Enter cd HospitalManagement
9. And type npm dev command.

10.Open browser with <http://localhost:3000/>

**Steps to download backend project from GitHub in VS code**

1. Open the Project from Version Control: In the IntelliJ IDEA welcome screen, or if you have an existing project open, you can go to File > New > Project from Version Control > Git.

2. Clone Repository: In the URL field, enter the URL of the GitHub repository you want to clone. You can find this URL on the GitHub repository page. Then click Clone.

3. Select Destination Folder: Choose the local directory where you want to clone the repository. Once selected, click Clone.

4. Authenticate (if required): If the repository is private or requires authentication, IntelliJ IDEA will prompt you to provide your GitHub username and password or personal access token. Enter the required credentials and click OK.

5. Wait for Cloning to Complete: IntelliJ IDEA will start cloning the repository from GitHub to your local machine. You will see a progress indicator in the bottom right corner of the IntelliJ IDEA window.

6. Open Cloned Project: Once the cloning process is complete, IntelliJ IDEA will open the cloned project in a new window automatically.

7. Configure Project Settings (if necessary): Depending on the project, you may need to configure additional settings such as SDK, project structure, dependencies, etc.

8. Run HospitalManagementSystem file

9. Its runs in <http://localhost:8081>

**MySQL work bench set up(And add all the comments given below)**



**Login credentials:**

**Admin**

Username: admin

Password: admin

**Username**

Username: john

Password: john

You can add user by registering the user to login but admin credentials cannot be changed.

**FrontEnd:**

For the frontend here we are using vs code as an editor or ide.

**Steps for installing an react project :**

* Create a new folder Emp-Management-ui or the name the name you like to after creating a folder open the folder in vs code.
* Open the terminal in vs code manually or using the shortcut ctrl+j and use the command npm init vite which creates a new react default project.

**Required packages for the react-project:** (giving the terminal commands to install the packages):

* npm install axios –library
* npm install react-router-dom
* npm install bootstrap
* npm install sweetalert2
* npm i react-datepicker
* npm i react-select

**Axios:**

* Axios is a popular Promise-based HTTP client for JavaScript that works both in the browser and in Node.js environments.
* It provides a simple and intuitive API for making HTTP requests from client-side applications.
* Axios supports various features such as interceptors, request and response transformations, automatic JSON parsing, and more.
* It is widely used in modern web development for consuming APIs and handling asynchronous HTTP requests.

**React-Router-dom:**

* React Router DOM is a popular library for handling routing in React applications.
* It provides a declarative way to navigate between different components or pages in a single-page application (SPA).
* React Router DOM is built on top of React and uses its component-based architecture to manage routing and rendering.

**Bootstrap:**

* Bootstrap is an open-source front-end framework developed by Twitter.
* It provides a collection of CSS and JavaScript components for building responsive and mobile-first websites and web applications.
* Bootstrap simplifies the process of creating visually appealing and consistent user interfaces by offering pre-styled components, layout grids, and utility classes.

**SweetAlert2:**

* SweetAlert2 is a JavaScript library for creating beautiful, customizable, and responsive alert dialogs.
* It is an enhanced version of the original SweetAlert library with additional features and improvements.
* SweetAlert2 allows developers to create alerts, prompts, and confirmations with custom styles, animations, and functionalities.
* It is widely used in web development to enhance the user experience and provide visually appealing notifications.

**Local URL for the front end is :** <http://localhost:3000>

There are 4 main components for the Crud operations for the employees.

* Add
* View
* Update
* Delete

Only view page can be accessed by the user.

Admin can access all the operations.

**Login:**

Once you have logged in the browser will create a token will not be cleared until you logged out

**Login as an admin:**

Username: admin

Password: admin

**Login as an user:(**default)

Username: john

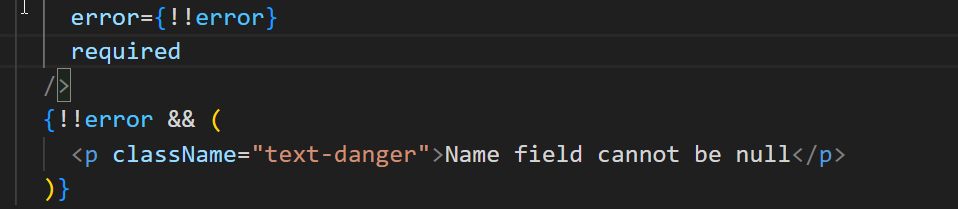
Password: john

New User also can be register and able to view the employee details

**Logout:**

* Once you click on to the logout button the session storage and the local storage will be cleared.
* Again when you login to the page a new token wil be create again

**Validation:**

****

* Each and every field will contain this required and the message .
* Required means the field cannot be null.