User Information Page

Tech stack: Spring Boot and Reactjs configuration, Nodejs,

DB-MySQL. To run the full stack application, Run the server: `$ mvn spring-boot:run` The react app by running: `$ npm run-script watch`

Now, it'll display in the browser(i.e)- http://localhost:8080/

DESCRIPTION:

- First, I have configured react js with spring boot.
- To configure spring boot I am using Maven which gives nice xml layout and for installing dependencies and plugins etc..
- Dependencies which mainly added are:
  - JPA-Data persistence in SQL with java persistence API
  - Thymeleaf- Server side java template engine
  - WEB- To build web,including RESTful, applications using Spring MVC.
  - **Mysql**-Database which is a **relational database**
  - Lombok- Annotation library to reduce boilerplate code.
- REST API Service:
  - Models-For perfoming **CRUD**(Create ,Read,Update,Delete) functionality.
  - Repositories-Repositories package which is an interface for **User Repository.**
  - The contactRepository extends JpaRepository which allow to access and persist data between java object class and relational database.
  - Controllers- Controller package with the class called **User Controller** and all the action feature required by our CRUD app (i.e) **GET,POST,PUT and DELETE actions**.
  - Exceptions- For clean and packaged.
  - **API TESTING-Tested using POSTMAN**
- Now, I have integrated react js by installing some packages and some dependencies like webpack ,babel ,axios, redux, react.

- **The setup of Spring Boot Restful API Server and configuration of React with Babel and Webpack is done.**

- UI section is divided into 3 components:
  - Header section
  - RegisterUserSection which is basically having input fields and a submit button.
  - Users Section
- Header section:
  - Location- /frontend/src/components/Header
  - This is where all the layouts of the UI has written.
  - Functional component called Header has been used. Then I have installed the core and icon dependencies.
  - Instead of using inline styling I have used with a variable just to keep the code clean and readable.
- ResisterUser Section:
  - I have created a class ResgisterUser.js component.
  - Constructor and binding method have been used.
  - Binding in constructor onSubmit method, Binding uses arrow functions onChange method. The onChange method constantly updates the state of the fields as we type some stuff in some areas.
  - Props- The onSubmit method is implemented in the root App.js component.
- App.js:
  - Bringing in components in render()-Header,RegisterUser, and Users component are painted on the DOM by the render functions,
  - Constructor with state- Defined a users array in state which is responsible for storing all users that are retrieved from the backend(Spring RESTful API).
- Users:
  - I have accessed props using this.props from App.js .Keyword and map function to iterate the list of data.
  - UserInfo- To keep the code clean and to pass user information down to a component called UserInfo.

- UserInfo:
  - Final component in the root folder called UserInfo.js.
  - Material icons imports- Icons dependencies which were installed before has been used here.
  - Passing argument to a prop- Onclick method implements removeUser callback function in the parent component App.js.Now, to remove specific user I have passed unique id in this case id. Now passing an argument inside a prop requires us to bind the argument thus in this case passing our arguments in this.props.removeUser.bind(this,id) binding method.
- CRUD operations in React with axios:
  - Finally, implemented the CRUD operations in app.js component.
  - First imported axios package
  - GET- Accessing the end point contact/all from the spring boot API to get the list of users from the DB.
  - DELETE-**removeUser callback** function is implemented here.The UI is upated using setState method and a spread operator inside the callback function.
  - The spread operator copies whatever is in the users state, then I have filtered all id which is not equal to one I am deleting.
  - POST- **addUser callback-** This method is hitting the "/contact/save"method from spring boot RESTful API and then returns that user. Then by using spread operator in the setState method, I have append the returned data.