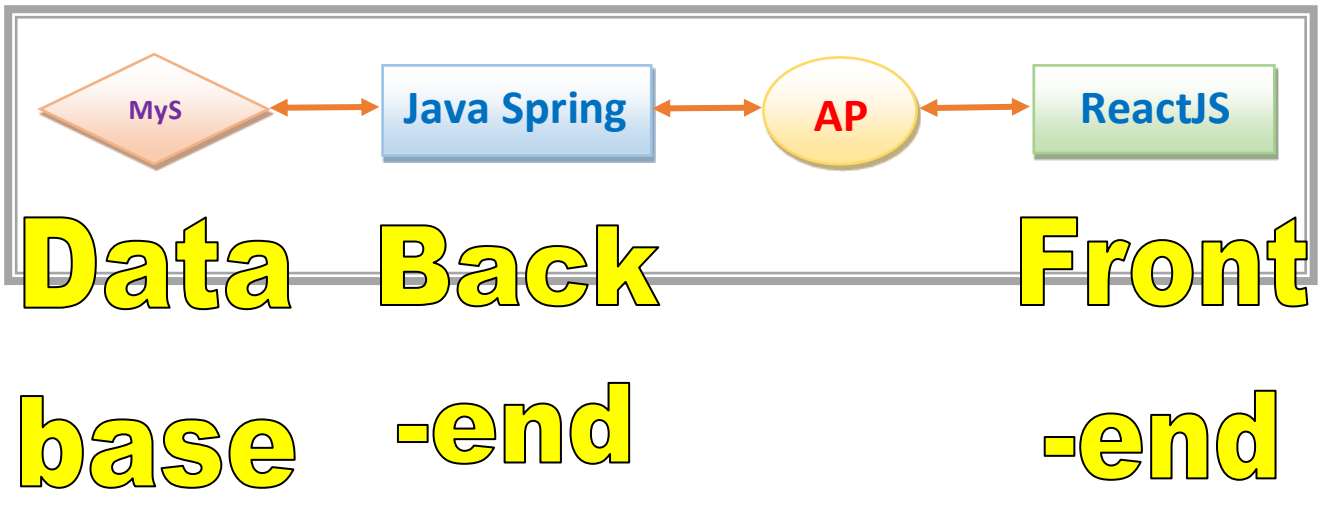
TUTORIAL 11 – CONSUME RESTFUL API WITH REACTJS

➢ Contents:

- Introduce about ReactJS library
- Consume API, created by Java Spring, using ReactJS

❖ Introduction:

- ReactJS is a JavaScript library for building user interfaces. It was developed and is maintained by Facebook and is widely used for building web applications. It allows developers to create reusable UI components and manage the state of their application, making it easier to build complex user interfaces.
- Some popular alternatives for ReactJS:
  - AngularJS
  - VueJS
- Some popular ways to consume Restful API in ReactJS:
  - *Axios: a promise-based HTTP client*
  - Fetch API: a browser built-in web API
- System architecture diagram:



➢ Instructions:

1. Open the previous Java Spring project, which creates Restful APIs *(Refer to Tutorial 5).* This project is used as back-end side.

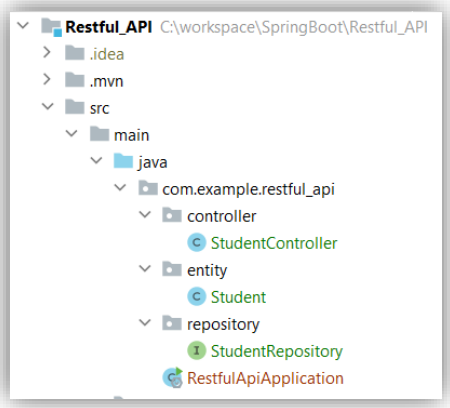


Figure 1 – Java Spring project structure (back-end)

2. **(VERY IMPORTANT)** Enable CORS (Cross-Origin Resource Sharing) either locally for each controller or globally for the whole project.

   NOTE: Without enabled CORS, the back-end & front-end sides can not communicate and share data to each other through APIs.

```java
@SpringBootApplication
public class RestfulApiApplication {

    public static void main(String[] args) {
        SpringApplication.run(RestfulApiApplication.class, args);
    }

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping( pathPattern: "/").allowedOrigins("http://localhost:3000");
            }
        };
    }
}
```

Figure 2 – Enable CORS globally - ***RestfulApiApplication.java***

```java
@CrossOrigin(origins = "http://localhost:3000")
@RestController
public class StudentController {
```

*Figure 3 - Enable CORS locally in each controller - **StudentController.java***

Note:

- *Default port for Java Spring is 8080*
- *Default port for ReactJS is 3000*

3. Start the web server to run Java Spring project then leave it on to share APIs to ReactJS. Do not *STOP* this server.

Note: Both servers for front-end & back-end must run at the same time. ⇨ Open 2 projects in 2 independent windows in IntelliJ or open Java Spring project in IntelliJ and ReactJS in Visual Studio Code (or alternatives)

4. Create a new ReactJS project, which consumes Restful APIs from previous project. This project is used as front-end side.

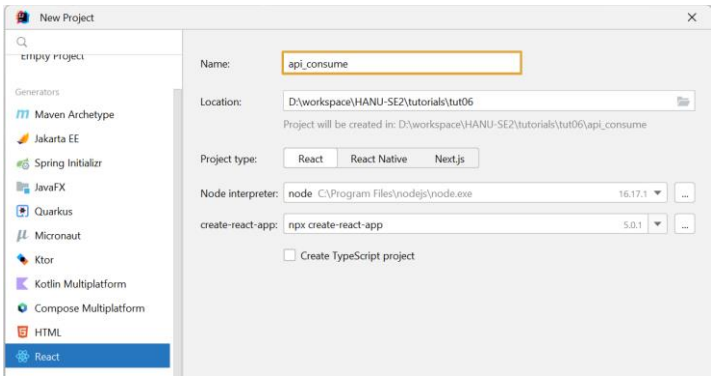


Figure 4 - Create new ReactJS project

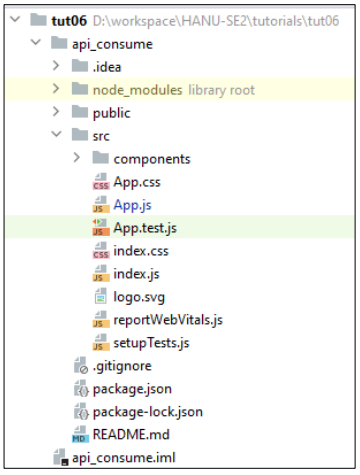


Figure 5 - ReactJS project structure (front-end)

5. Install Axios module by running below command in integrated terminal at bottom left corner

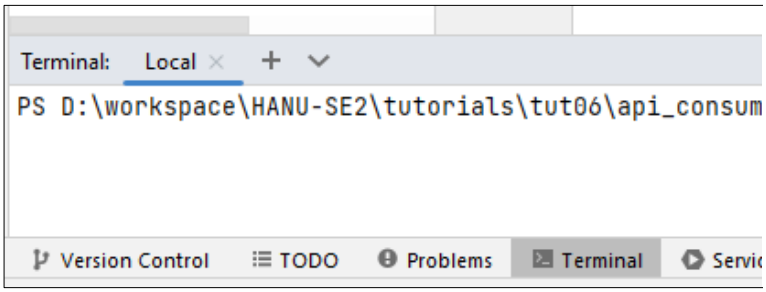


Figure 6 - Set default Terminal

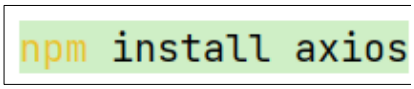


Figure 7 - Axios module installation

6. Create **components** package in **src** folder of ReactJS project

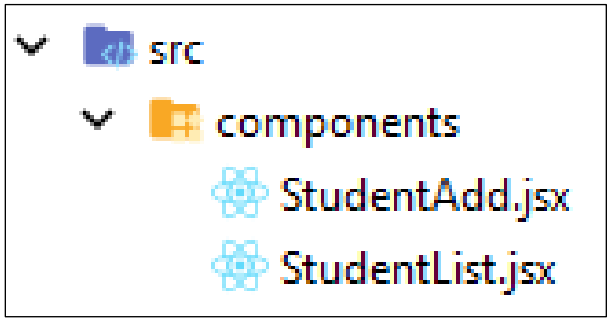


Figure 8 - Create ReactJS components

7. (*Optional*) Attach Bootstrap to decorate for user interface in **index.html** (located at **public** folder)

```
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
          rel="stylesheet" integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
```

*Figure 9 - **index.html***

8. Create module (component) **StudentList** to fetch student data from API

```
import React from 'react';
import axios from 'axios';

export default class StudentList extends React.Component {
    state = {
        students: []
    }

    url = "http://localhost:8080/"

    componentDidMount() {
        this.fetchStudentList();
    }

    componentDidUpdate(prevProps : Readonly<P> , prevState : Readonly<S> , snapshot : SS ) {
        if (this.props.reloadList !== prevProps.reloadList) {
            this.fetchStudentList();
        }
    }

    fetchStudentList = () => {
        axios.get(this.url).then((res : AxiosResponse<any> ) => {
            const students = res.data;
            this.setState( state: { students });
        });
    };
```

*Figure 10 - **StudentList.jsx (1)***

```
render() {
    return (
        <div className="container text-center mt-3">
            <table className="table table-primary">
                <thead>
                    <tr>
                        <th colSpan="4" className="h3 text text-danger bg-warning">STUDENT LIST</th>
                    </tr>
                    <tr className="h5 text text-success">
                        <th>Student Id</th>
                        <th>Student Name</th>
                        <th>Student Age</th>
                    </tr>
                </thead>
                <tbody>
                    {
                        this.state.students
                            .map(student =>
                                <tr key={student.id}>
                                    <td> {student.id} </td>
                                    <td> {student.name} </td>
                                    <td> {student.age} </td>
                                </tr>
                            )
                    }
                </tbody>
            </table>
        </div>
    )
}
```

*Figure 11 - **StudentList.jsx (2)***

9. Create module **StudentAdd** to add new student using form. Data from form will be passed to Spring Boot by API then be added to database.

```
import React from "react";
import axios from "axios";

export default class StudentAdd extends React.Component {
    state = {
        name: '',
        age: ''
    }

    url = "http://localhost:8080/"

    handleChange = event => {
        this.setState( state: {
            [event.target.id]: event.target.value
        });
    }

    handleSubmit = event => {
        event.preventDefault();

        event.target.reset();
        this.setState( state: {
            name: '',
            age: '',
        });

        const student = {
            name: this.state.name,
            age: this.state.age
        };

        axios.post( url: this.url + 'add', student)
            .then(res => {
                console.log(res);
                this.props.reloadStudentList();
            })
    }
```

*Figure 12 - **StudentAdd.jsx (1)***

```
render() {
    return (
        <div className="container text-center mt-3 mb-5">
            <h3 className="bg-warning text-primary p-2">ADD NEW STUDENT</h3>
            <form className="form card p-3 bg-light" onSubmit={this.handleSubmit} >
                <label className="form-label h5 text-success"> Student Name </label>
                <input className="form-control" type="text" id="name" minLength="3"
                       maxLength="20" required onChange={this.handleChange} />
                <label className="form-label h5 text-success"> Student Age </label>
                <input className="form-control" type="number" id="age" min="18" max="25"
                       required onChange={this.handleChange}/>
                <div className="text-center">
                    <button className="btn btn-primary mt-3 col-md-3" type="submit">
                        Add
                    </button>
                </div>
            </form>
        </div>
    );
}
```

*Figure 13 - **StudentAdd.jsx (2)***

10.Config file **App.js** to add 2 modules: **StudentList & StudentAdd**

```
import './App.css';
import StudentList from "./components/StudentList";
import StudentAdd from "./components/StudentAdd";
import {useState} from "react";

function App() {
    const[reloadList, setReloadList] = useState( initialState: false);

    const handleReloadList = () => {
        setReloadList(!reloadList);
    };
      return (
        <div className="container text-center card mt-3">
            <div className="row">
                <div className="col">
                    <StudentAdd reloadStudentList={handleReloadList}/>
                </div>
                <div className="col">
                    <StudentList reloadList={reloadList}/>
                </div>
            </div>
        </div>
      );
}

export default App;
```

*Figure 14 - **App.js***

11.Start the ReactJS project with **Shift + F10** or typing "**npm start**" in Terminal.

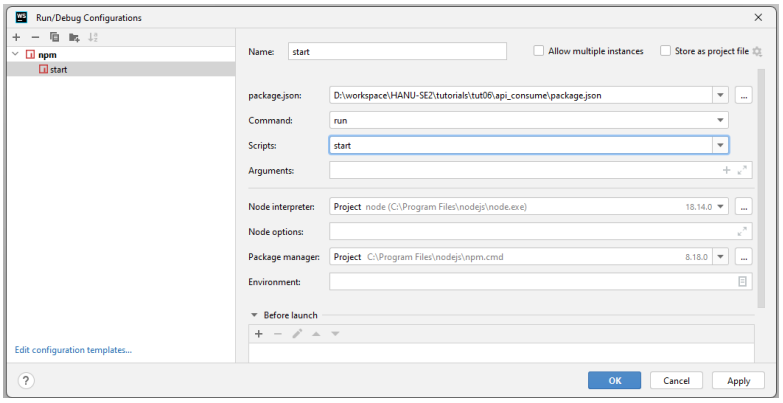


Figure 15 - Add npm configuration

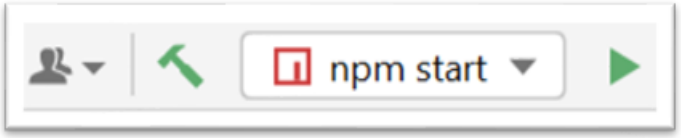


Figure 16 - Start the ReactJS project

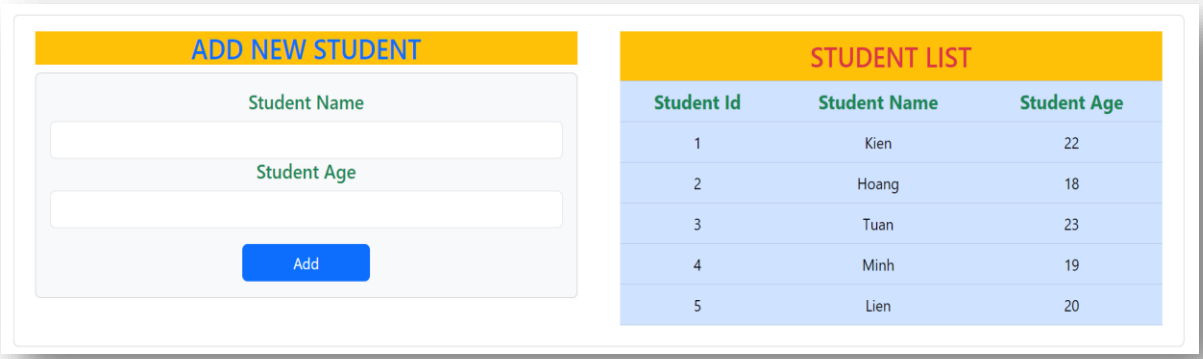


Figure 17 – Current result

Figure 18 - Final result

➢ TASKS:

- Write the code of UPDATE & DELETE to complete all CRUD features
- Create new modules and write proper codes to consume remained API
- Compress the whole project and submit to FIT LMS with file name syntax: *FullName_StudentID_SE1_Tut11.zip*