

LAB 5

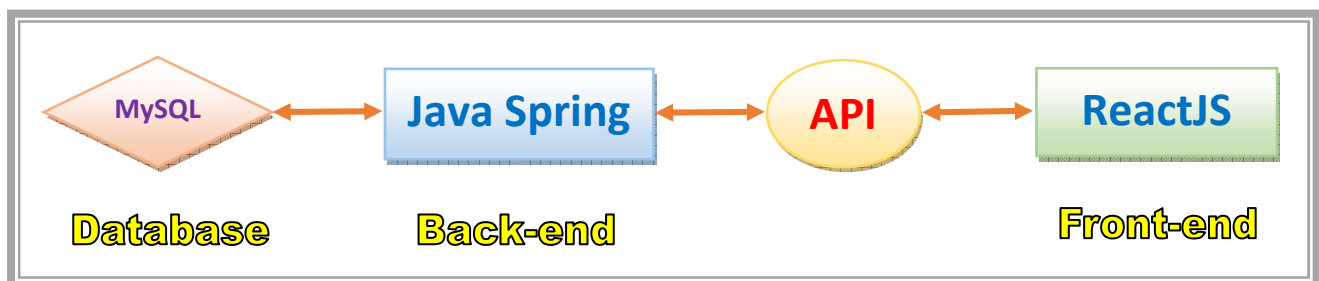
CONSUME RESTFUL API WITH REACTJS

➤ CONTENT

- Introduce to ReactJS
- Consume API with 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 alternatives to ReactJS:
 - AngularJS
 - VueJS
- Some ways to consume Restful API with ReactJS:
 - *Axios: a promise-based HTTP client*
 - *Fetch API: a browser built-in web API*
- System architecture:



➤ INSTRUCTION

1. Use previous Java Spring Boot project as back-end.

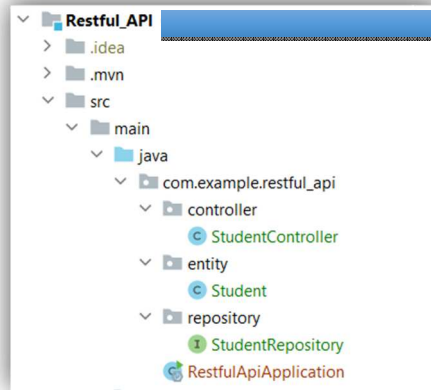


Figure 1 – Java Spring Boot project structure (back-end)

2. **Important:** Must 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 exchange data using API.

```
@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

```
@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 Boot is 8080
- Default port for ReactJS is 3000

3. Start the web server to run Java Spring Boot 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 IntelliJ windows or open Java Spring Boot project in IntelliJ and ReactJS in Visual Studio Code (or other alternative)

4. Create a new ReactJS project as front-end, which consumes Restful API

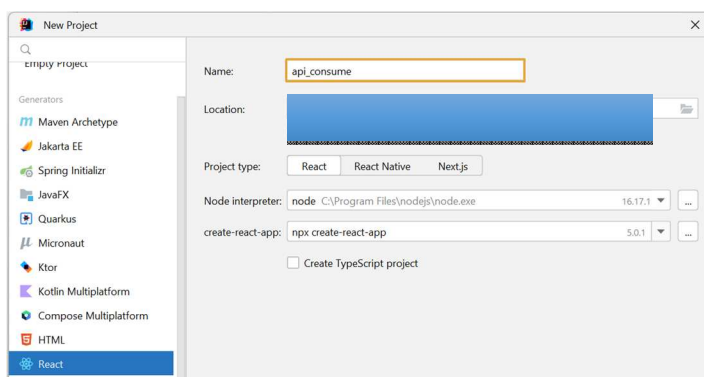


Figure 4 - Create new ReactJS project

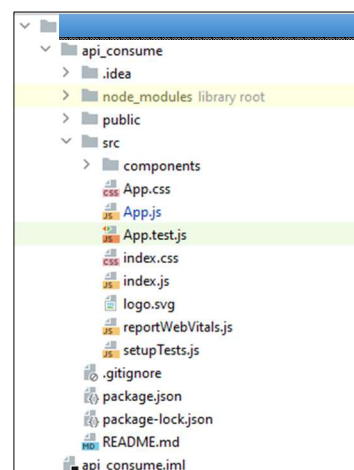


Figure 5 - ReactJS project structure (front-end)

5. Install Axios library (module/package) by running below command in integrated terminal at bottom left corner

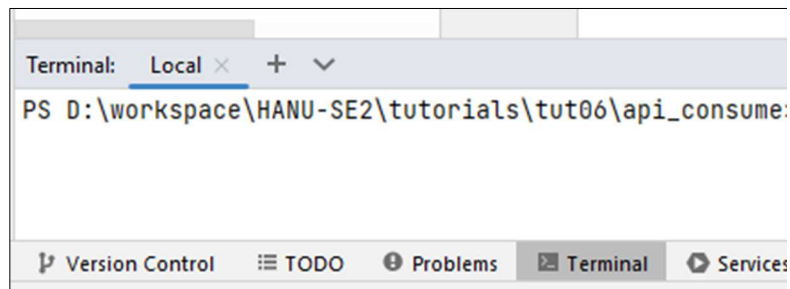


Figure 6 - Set default Terminal

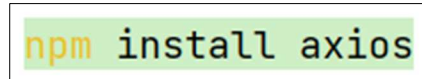


Figure 7 - Axios module installation

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

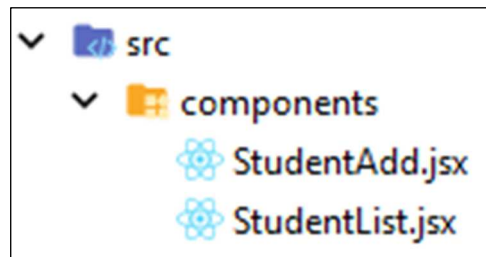


Figure 8 - Create ReactJS components

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



Figure 9 - index.html

8. Create 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>
            <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(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 these 2 components

```
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.

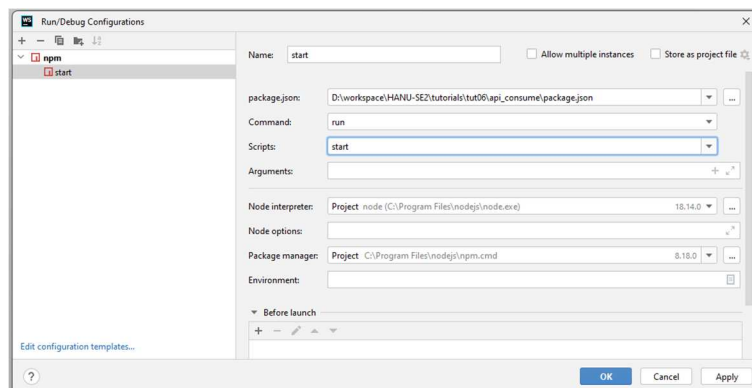


Figure 15 - Add **npm** configuration

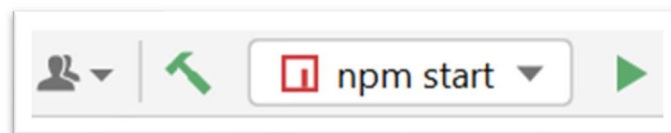


Figure 16 - Start the ReactJS project

ADD NEW STUDENT

Student Name

Student Age

Add

STUDENT LIST

Student Id	Student Name	Student Age
1	Kien	22
2	Hoang	18
3	Tuan	23
4	Minh	19
5	Lien	20

Figure 17 – Current result

STUDENT LIST				
Student Id	Student Name	Student Age	Update	Delete
1	Minh	18	<button>Update</button>	<button>Delete</button>
2	Hung	20	<button>Update</button>	<button>Delete</button>
3	Lien	22	<button>Update</button>	<button>Delete</button>

Figure 18 - Final result

➤ TO-DO

- Create UPDATE & DELETE components to complete CRUD features
- Create other components to consume remained APIs