Batch: D-2	**Roll No.:** 16010122151
Experiment / assignment / tutorial No.____	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Demonstrate axios to Create Mock API Server

AIM: To Implement the React Axios

Problem Definition:

Build a React application that interacts with a RESTful API using Axios to perform CRUD (Create, Read, Update, Delete) operations. The application should allow users to view, add, update, and delete data from the server. The application should allow users to view, add, update, and delete student data, with smooth navigation between different views using the `useNavigate` hook.

Requirements:

- Create a new React application using create-react-app.
- Install Axios using npm install axios.
- Install react-router-dom to handle navigation (npm install react-router-dom).

Data Fetching:

Create a component (StudentList.js) that fetches a list of students from a RESTful API endpoint (e.g., https://api.example.com/students) and displays them in a table or list. Handle loading states and errors during the fetch process.

Adding a New Student:

- Implement a form component (AddStudent.js) that allows users to add a new student record.
- Use Axios to send a POST request to the API with the new student data.
- Upon successful submission, navigate the user back to the student list view using useNavigate and display the newly added student in the list.

Updating Student Data:

- Implement an edit functionality in a separate component (EditStudent.js) that allows users to update an existing student's information.
- Use Axios to send a PUT request to the API with the updated student data.
- Upon successful submission, navigate the user back to the student list view using useNavigate, and reflect the updated student information in the list.

Deleting a Student:

- Add a delete button next to each student in the list.
- When the delete button is clicked, use Axios to send a DELETE request to the API.
- Upon successful deletion, the student should be removed from the list without requiring a page reload.

Navigation:

- Use useNavigate to smoothly navigate between different components/views (StudentList, AddStudent, EditStudent).
- Ensure that the browser’s back and forward buttons work correctly to navigate between the views.

Resources used:

Expected OUTCOME of Experiment:

CO 2:. Illustrate the concepts of various front-end, back-end web application development technologies & frameworks using different web development tools.

Books/ Journals/ Websites referred:

1. Shelly Powers Learning Node O' Reilly 2 nd Edition, 2016.

Pre Lab/ Prior Concepts:

Write details about the following content

- useNavigate
- Axios
- Routes in React

1. useNavigate

- **Purpose:** Programmatically navigate to different routes in a React application.
- **Usage:** `const navigate = useNavigate();`
- **Example:** `navigate('/path');`

2. Axios

- **Purpose:** HTTP client for making requests to a server.
- **Installation:** `npm install axios`
- **Usage:**

```
axios.get('/endpoint').then(response => console.log(response.data));
axios.post('/endpoint', data).then(response => console.log(response.data));
```

- **Features:** Supports request/response interceptors, automatic JSON transformation, and error handling.

3. Routes in React

- **Purpose:** Manage navigation and rendering of different views/pages.
- **Key Components:**
  - `<BrowserRouter>`: Manages the routing.
  - `<Routes>`: Holds `<Route>` elements.
  - `<Route>`: Defines paths and components.
  - `<Link>`: Creates navigation links.
- **Example:**

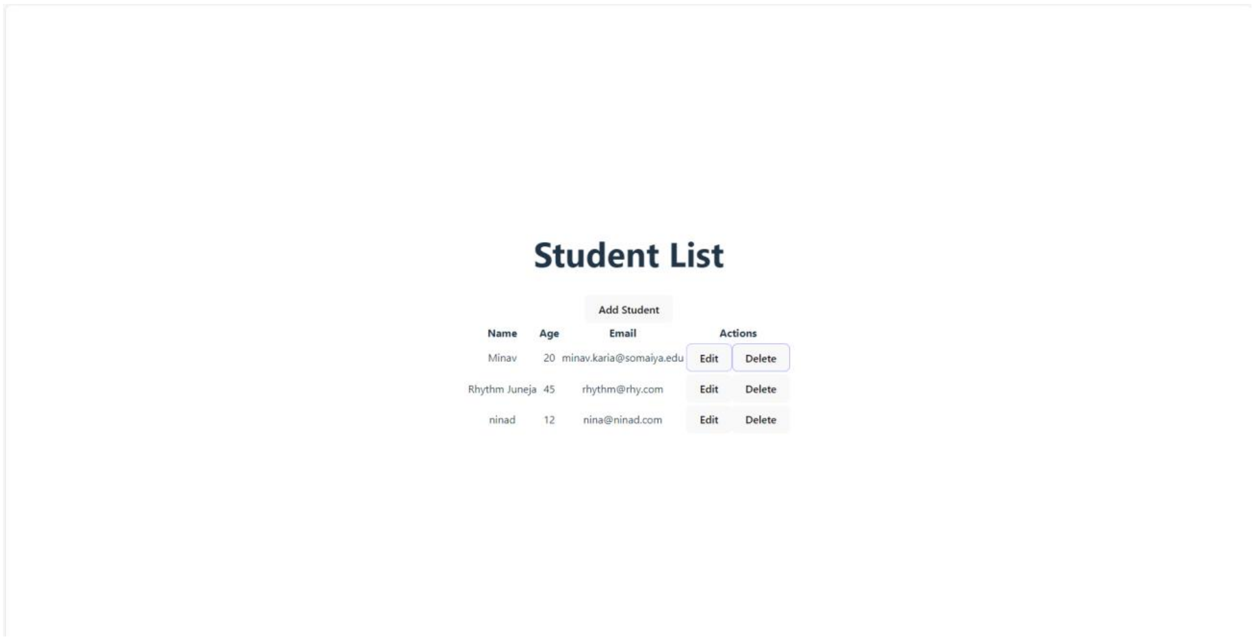
```
<Router>
  <nav><Link to="/">Home</Link><Link to="/about">About</Link></nav>
  <Routes>
    <Route path="/" element={<Home />} />
    <Route path="/about" element={<About />} />
```

```
      </Routes>
    </Router>
```

Implementation Details:

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { BrowserRouter as Router, Route, Routes, useNavigate,
useParams } from 'react-router-dom';
import './App.css';

const StudentList = () => {
  const [students, setStudents] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const navigate = useNavigate();

  useEffect(() => {
    const fetchStudents = async () => {
      try {
        const response = await
axios.get('http://localhost:5000/students');
        setStudents(response.data);
      } catch (err) {
        setError('Error fetching students');
      } finally {
        setLoading(false);
      }
    };
    fetchStudents();
  }, []);

  const handleDelete = async (id) => {
    try {
      await axios.delete(`http://localhost:5000/students/${id}`);
      setStudents(students.filter(student => student.id !== id));
    } catch (err) {
      setError('Error deleting student');
    }
  };

  if (loading) return <p>Loading...</p>;
  if (error) return <p>{error}</p>;
  return (
    <div>
      <h1>Student List</h1>
      <button onClick={() => navigate('/add')}>Add Student</button>
      <table>
        <thead>
          <tr>
            <th>Name</th>
            <th>Age</th>
            <th>Email</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {students.map(student => (
            <tr key={student.id}>
              <td>{student.name}</td>
              <td>{student.age}</td>
              <td>{student.email}</td>
              <td>
                <button onClick={() =>
navigate(`/edit/${student.id}`)}>Edit</button>
                <button onClick={() =>
handleDelete(student.id)}>Delete</button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};

const AddStudent = () => {
  const [name, setName] = useState('');
  const [age, setAge] = useState('');
  const [email, setEmail] = useState('');
  const [error, setError] = useState('');
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post('http://localhost:5000/students', { name, age,
email });
      navigate('/');
    } catch (err) {
      setError('Error adding student');
    }
  };

  return (
    <div>
      <h1>Add Student</h1>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Name</label>
          <input type="text" value={name} onChange={(e) =>
setName(e.target.value)} required />
        </div>
        <div>
          <label>Age</label>
          <input type="number" value={age} onChange={(e) =>
setAge(e.target.value)} required />
        </div>
        <div>
          <label>Email</label>
          <input type="email" value={email} onChange={(e) =>
setEmail(e.target.value)} required />
        </div>
        <button type="submit">Add Student</button>
        {error && <p>{error}</p>}
      </form>
    </div>
  );
};

// EditStudent Component
const EditStudent = () => {
  const [name, setName] = useState('');
  const [age, setAge] = useState('');
  const [email, setEmail] = useState('');
  const [error, setError] = useState('');
  const navigate = useNavigate();
  const { id } = useParams();

  useEffect(() => {
    const fetchStudent = async () => {
      try {
        const response = await
axios.get(`http://localhost:5000/students/${id}`);
        const student = response.data;
        setName(student.name);
        setAge(student.age);
        setEmail(student.email);
      } catch (err) {
        setError('Error fetching student');
      }
    };
    fetchStudent();
  }, [id]);

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.put(`http://localhost:5000/students/${id}`, { name,
age, email });
      navigate('/');
    } catch (err) {
      setError('Error updating student');
    }
  };

  return (
    <div>
      <h1>Edit Student</h1>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Name</label>
          <input type="text" value={name} onChange={(e) =>
setName(e.target.value)} required />
        </div>
        <div>
          <label>Age</label>
          <input type="number" value={age} onChange={(e) =>
setAge(e.target.value)} required />
        </div>
        <div>
          <label>Email</label>
          <input type="email" value={email} onChange={(e) =>
setEmail(e.target.value)} required />
```

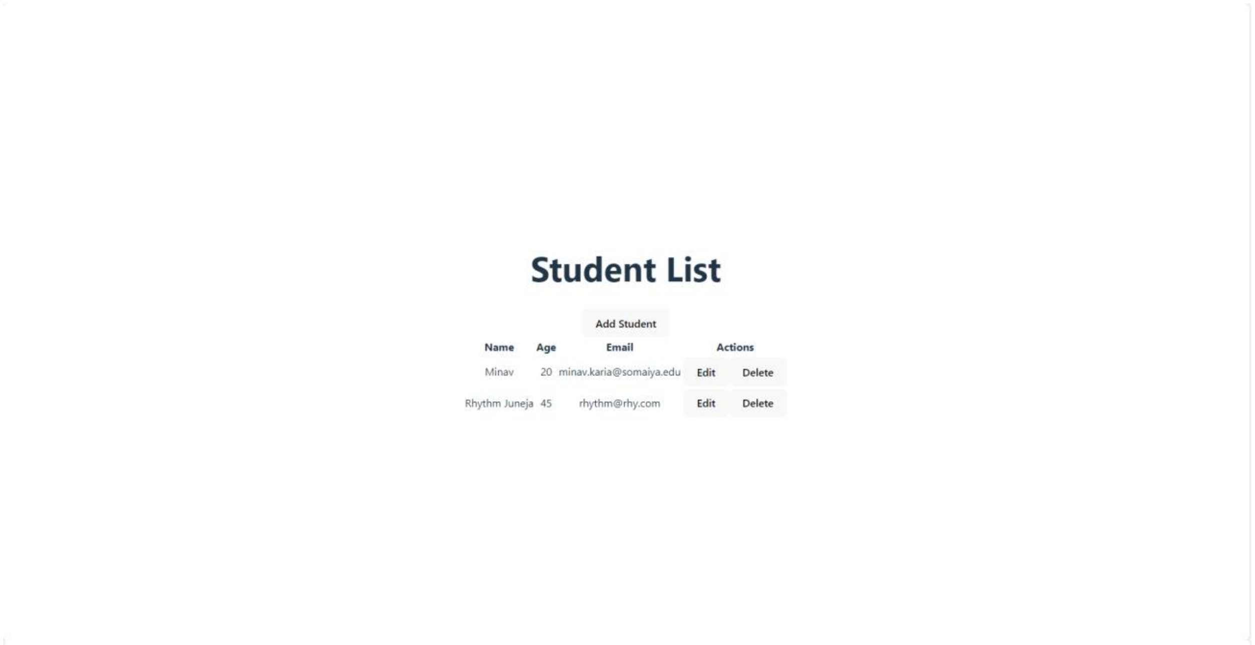
```
        </div>
        <button type="submit">Update Student</button>
        {error && <p>{error}</p>}
      </form>
    </div>
  );
};

const App = () => {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<StudentList />} />
        <Route path="/add" element={<AddStudent />} />
        <Route path="/edit/:id" element={<EditStudent />} />
      </Routes>
    </Router>
  );
};

export default App;
```



Student List

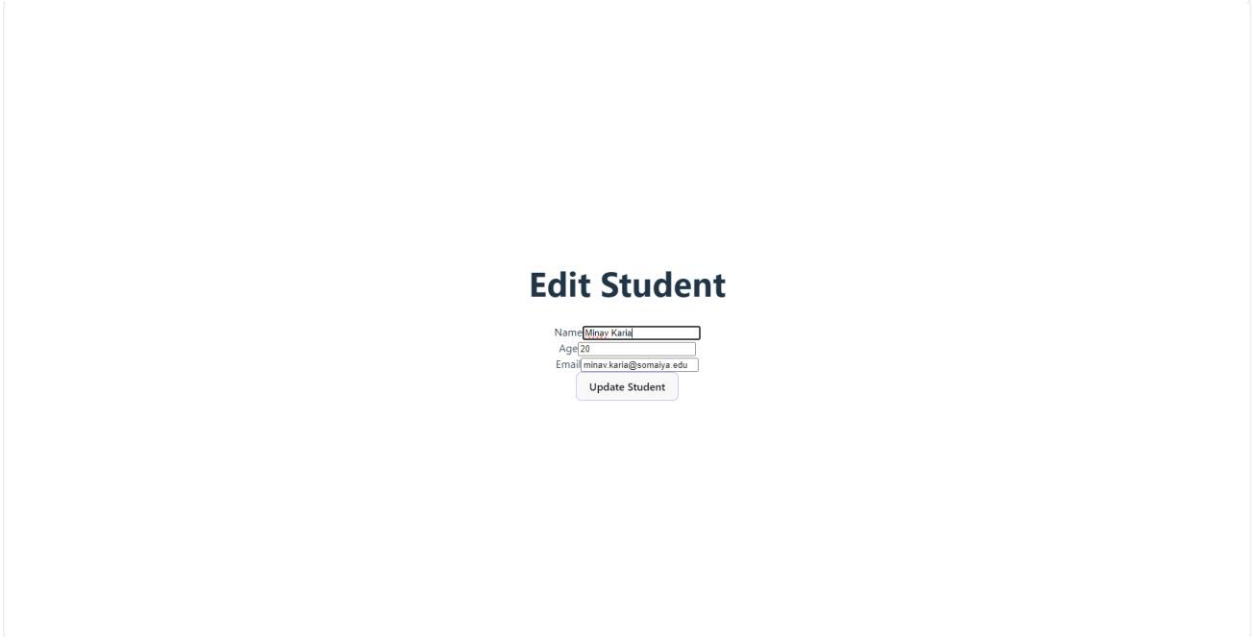
Add Student

Name	Age	Email	Actions	
Minav	20	minav.karia@somaiya.edu	Edit	Delete
Rhythm Juneja	45	rhythm@rhy.com	Edit	Delete
ninad	12	nina@ninad.com	Edit	Delete



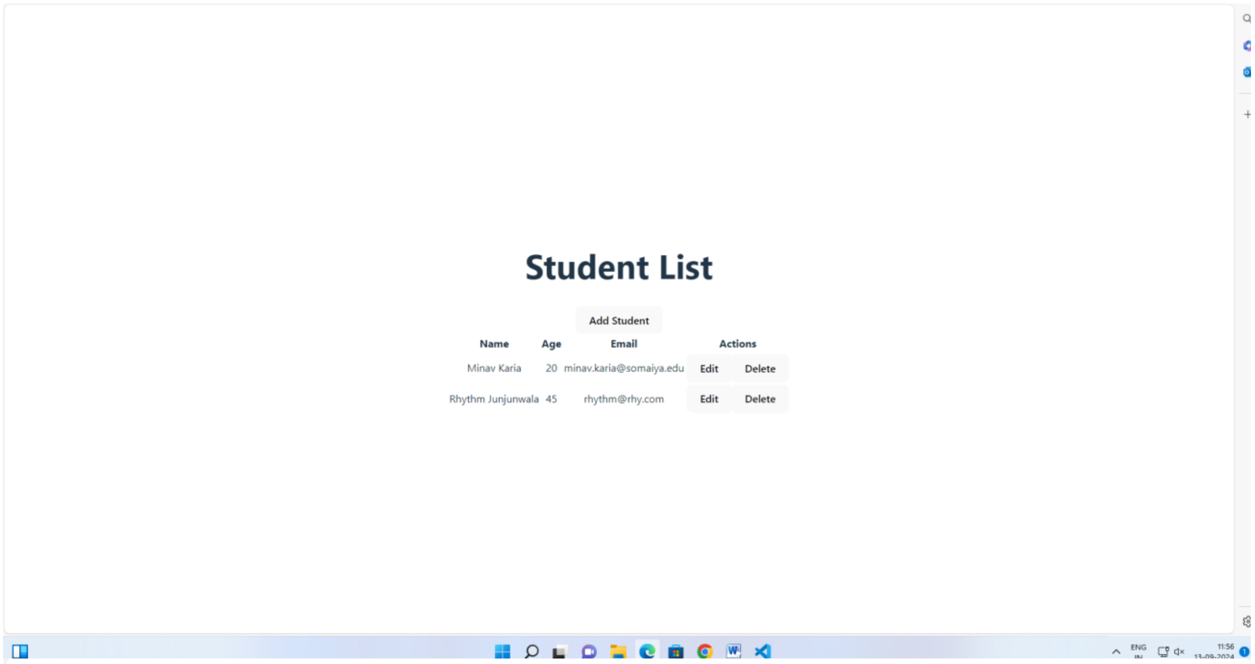
Student List

Name	Age	Email	Actions	
Minav	20	minav.karia@somaiya.edu	Edit	Delete
Rhythm Juneja	45	rhythm@rhy.com	Edit	Delete



Edit Student

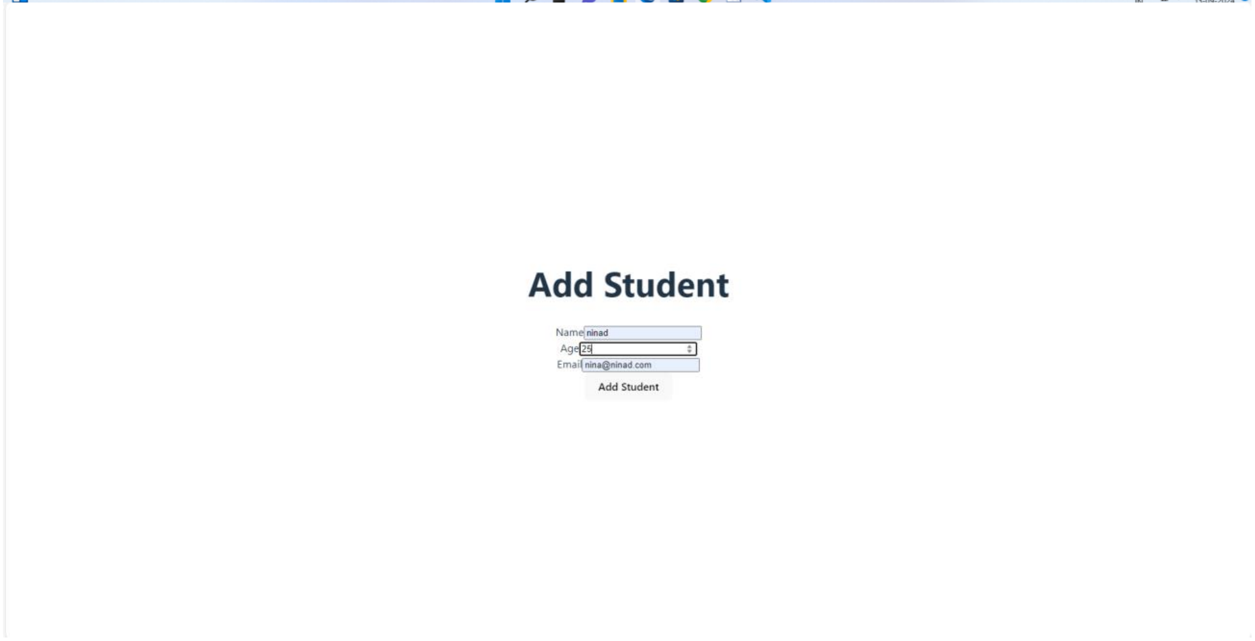
Name: Minay Karia
Age: 20
Email: minav.karia@somaiya.edu



Student List

Add Student

Name	Age	Email	Actions	
Minav Karia	20	minav.karia@somaiya.edu	Edit	Delete
Rhythm Junjunwala	45	rhythm@rhy.com	Edit	Delete



Add Student

Name	ninad
Age	25
Email	nina@ninad.com

Add Student

Conclusion:

We learned about axios, react-router-dom and how to implement it with the APIs with real life use case

Postlab questions:

1) Different ways to Add Api in React/Javascript with example.

1. Using Fetch API

- **Method:** `fetch()`
- **Example:**

```
useEffect(() => {
  fetch('https://api.example.com/data')
    .then(response => response.json())
    .then(data => setData(data));
}, []);
```

2. Using Axios

- **Installation:** `npm install axios`
- **Method:** `axios.get()`
- **Example:**

```
useEffect(() => {
  axios.get('https://api.example.com/data')
    .then(response => setData(response.data));
}, []);
```

3. Using `async/await` with Fetch

- **Method:** `async/await` syntax
- **Example:**

```
useEffect(() => {
  const fetchData = async () => {
    const response = await fetch('https://api.example.com/data');
    const result = await response.json();
    setData(result);
  };
  fetchData();
}, []);
```

4. Using React Query

- **Installation:** `npm install react-query`
- **Method:** `useQuery()`
- **Example:**

```
const { data, isLoading, error } = useQuery('data', () =>
  axios.get('https://api.example.com/data').then(res => res.data)
);
```

5. Using SWR

- **Installation:** `npm install swr`
- **Method:** `useSWR()`
- **Example:**

```
const { data, error } = useSWR('https://api.example.com/data', fetcher);
```