SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya School of Engineering
(formerly K J Somaiya College of Engineering)

K. J. Somaiya College of Engineering, Mumbai-77
Department of Computer Engineering

Batch: D1 Roll No.: 16010123298
Experiment / assignment / tutorial No.6
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

The useNavigate hook is part of React Router DOM v6 and is used for programmatic navigation within a React application.

- Purpose: It returns a function that lets you navigate to a different route (URL) imperatively, typically after an action like a form submission, a successful API call, or a button click. It replaces the useHistory hook from older versions of React Router.

Axios

Axios is a popular promise-based HTTP client for the browser and Node.js.

- Purpose: It simplifies making asynchronous HTTP requests (GET, POST, PUT, DELETE, etc.) to communicate with RESTful APIs.

- **Key Features**:
  - Promise-based: Naturally supports async/await syntax for cleaner asynchronous code.
  - Automatic JSON Transformation: Automatically converts the response data from JSON to a JavaScript object.
  - Interceptors: Allows you to modify requests and responses globally (e.g., adding an authentication token to every request).
  - Better Error Handling: Provides more detailed error objects than the native fetch API.
- **Basic Usage (CRUD)**:
  - Read (GET): axios.get(url)
  - Create (POST): axios.post(url, data)
  - Update (PUT/PATCH): axios.put(url/id, updatedData) or axios.patch(url/id, partialUpdatedData)
  - Delete (DELETE): axios.delete(url/id)

Routes in React

In React, routing is the process of navigating between different views (components) without actually reloading the entire page, which is fundamental to building a Single Page Application (SPA).

- Library: This is managed by the react-router-dom library.
- Core Components:
  - <BrowserRouter>: Wraps the entire application to enable client-side routing using clean URLs (e.g., /students).
  - <Routes>: Wraps all individual route definitions. It looks through its children (<Route>) and renders the first one whose path matches the current URL.

- <Route>: Maps a specific URL path to a React component.
  - Example: <Route path="/students" element={<StudentList />} />
- Dynamic Routes: Allows parts of the path to be variables (e.g., for editing a specific student).
  - Example: <Route path="/edit/:id" element={<EditStudent />} /> (The :id is a route parameter accessible via useParams()).

Methodology:

The experiment will follow these steps:

1. Environment Setup: Initialize a React project and install necessary libraries (axios, react-router-dom, and JSON Server for the mock API).
2. Mock API Creation: Set up a simple mock RESTful API for student data using JSON Server.
3. Routing Setup: Configure the main application routes (/, /add, /edit/:id) using react-router-dom.
4. Read Operation (StudentList): Implement data fetching using axios.get and useEffect to display the list of students.
5. Create Operation (AddStudent): Implement a form to capture student data and use axios.post to add a new record. Use useNavigate for redirection.
6. Update Operation (EditStudent): Implement a component to fetch existing data (axios.get for initial state) and update it using axios.put. Use useNavigate for redirection.
7. Delete Operation (StudentList): Implement a button to send an axios.delete request and update the UI state.

Implementation Details:

1. **Initial Setup and Dependencies**

- **Create React App:**

```
npx create-react-app react-axios-crud
cd react-axios-crud
```

- **Install Libraries:** `npm install axios react-router-dom json-server`
- **Setup JSON Server (Mock API):** Create a file named `db.json` in the root of your project:

```
{
  "students": [
    { "id": 1, "name": "Samiksha", "course": "Computer Science" },
    { "id": 2, "name": "Sambhav", "course": "Electrical Engineering" }
  ]
}
```

Base API URL for all requests: `http://localhost:5000/students`

2. Configure Routes in src/App.js:

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from
'react-router-dom';
import StudentList from './components/StudentList';
import AddStudent from './components/AddStudent';
import EditStudent from './components/EditStudent';

// Simple Navigation Bar
const NavBar = () => (
  <nav className="navbar navbar-expand-lg navbar-dark bg-primary">
    <div className="container-fluid">
      <Link className="navbar-brand" to="/">React Axios CRUD</Link>
      <div className="collapse navbar-collapse">
        <ul className="navbar-nav me-auto">
          <li className="nav-item">
            <Link className="nav-link" to="/">Student List</Link>
          </li>
          <li className="nav-item">
            <Link className="nav-link" to="/add">Add Student</Link>
          </li>
        </ul>
      </div>
    </div>
  </nav>
);
```

```
function App() {
  return (
    <Router>
      <NavBar />
      <div className="container mt-4">
        <Routes>
          <Route path="/" element={<StudentList />} />
          <Route path="/add" element={<AddStudent />} />
          <Route path="/edit/:id" element={<EditStudent />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

3. Student List Component (Read and Delete)

```
import React, { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import api from '../api/api';

const STUDENT_RESOURCE = '/';

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

  const fetchStudents = async () => {
    setLoading(true);
    setError(null);
    try {
      const response = await api.get(STUDENT_RESOURCE);
      setStudents(response.data);
    } catch (err) {
      const errorMessage = err.response
```

```
        ? `API Error: ${err.response.status} - Could not load students.`
        : 'Network Error: Check internet or MockAPI status.';
      setError(errorMessage);
    } finally {
      setLoading(false);
    }
  };

  useEffect(() => {
    fetchStudents();
  }, []);

  const handleDelete = async (id) => {
      if (!window.confirm("Are you sure you want to delete this student?"))
return;
    try {
      await api.delete(`/${id}`);
      setStudents(students.filter(student => student.id !== id));
    } catch (err) {
      setError('Error deleting student. Please refresh.');
    }
  };

  if (loading) return <div>Loading students...</div>;
  if (error) return <div className="alert alert-danger">Error: {error}</div>;

  return (
    <div>
      <h2>Student List</h2>
      <Link to="/add" className="btn btn-success mb-3">Add New Student</Link>
      <table className="table table-striped">
        <thead>
          <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Course</th>
            <th>Actions</th>
```

```
          </tr>
        </thead>
        <tbody>
          {students.map(student => (
            <tr key={student.id}>
              <td>{student.id}</td>
              <td>{student.name}</td>
              <td>
                {typeof student.course === 'object' && student.course !== null
                        ? (Object.keys(student.course).length === 0 ? 'N/A' :
JSON.stringify(student.course))
                    : student.course}
              </td>
              <td>
                   <Link to={`/edit/${student.id}`} className="btn btn-warning
btn-sm me-2">Edit</Link>
                <button
                  className="btn btn-danger btn-sm"
                  onClick={() => handleDelete(student.id)}
                >
                  Delete
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};

export default StudentList;
```

4. Add Students (Create Operation):

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import api from '../api/api';
```

```
const STUDENT_RESOURCE = '/';

const AddStudent = () => {
  const [student, setStudent] = useState({ name: '', course: '' });
  const [isSubmitting, setIsSubmitting] = useState(false);
  const navigate = useNavigate();

  const handleChange = (e) => {
    setStudent({ ...student, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setIsSubmitting(true);
    try {
      await api.post(STUDENT_RESOURCE, student);
      navigate('/');
    } catch (err) {
      alert('Error adding student. Check console for details.');
      setIsSubmitting(false);
    }
  };

  return (
    <div>
      <h2>Add New Student</h2>
      <form onSubmit={handleSubmit}>
        <div className="mb-3">
          <label className="form-label">Name</label>
          <input
            type="text"
            className="form-control"
            name="name"
            value={student.name}
            onChange={handleChange}
            required
```

```
          />
        </div>
        <div className="mb-3">
          <label className="form-label">Course</label>
          <input
            type="text"
            className="form-control"
            name="course"
            value={student.course}
            onChange={handleChange}
            required
          />
        </div>
                        <button   type="submit"   className="btn   btn-success"
disabled={isSubmitting}>
          {isSubmitting ? 'Adding...' : 'Add Student'}
        </button>
          <button type="button" className="btn btn-secondary ms-2" onClick={()
=> navigate(-1)}>
          Cancel
        </button>
      </form>
    </div>
  );
};

export default AddStudent;
```

5. Edit Student (Update, Delete Operations):

```
import React, { useState, useEffect } from 'react';
import { useParams, useNavigate } from 'react-router-dom';
import api from '../api/api';

const EditStudent = () => {
  const { id } = useParams();
  const [student, setStudent] = useState({ name: '', course: '' });
  const [loading, setLoading] = useState(true);
```

```
  const [isSubmitting, setIsSubmitting] = useState(false);
  const navigate = useNavigate();

  useEffect(() => {
    const fetchStudent = async () => {
      try {
        const response = await api.get(`/${id}`);
        setStudent(response.data);
      } catch (err) {
        alert('Error fetching student data for editing. Redirecting...');
        navigate('/');
      } finally {
        setLoading(false);
      }
    };
    fetchStudent();
  }, [id, navigate]);

  const handleChange = (e) => {
    setStudent({ ...student, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setIsSubmitting(true);
    try {
      await api.put(`/${id}`, student);
      navigate('/');
    } catch (err) {
      alert('Error updating student. Check console for details.');
      setIsSubmitting(false);
    }
  };

  if (loading) return <div>Loading student data for editing...</div>;

  return (
```

```
    <div>
      <h2>Edit Student (ID: {id})</h2>
      <form onSubmit={handleSubmit}>
        <div className="mb-3">
          <label className="form-label">Name</label>
          <input
            type="text"
            className="form-control"
            name="name"
            value={student.name}
            onChange={handleChange}
            required
          />
        </div>
        <div className="mb-3">
          <label className="form-label">Course</label>
          <input
            type="text"
            className="form-control"
            name="course"
            value={student.course}
            onChange={handleChange}
            required
          />
        </div>
                        <button   type="submit"   className="btn   btn-success"
disabled={isSubmitting}>
          {isSubmitting ? 'Updating...' : 'Update Student'}
        </button>
          <button type="button" className="btn btn-secondary ms-2" onClick={()
=> navigate(-1)}>
          Cancel
        </button>
      </form>
    </div>
  );
};
```

```
export default EditStudent;
```

Screenshots:

```
PS C:\Users\samik\Desktop\Coding>
PS C:\Users\samik\Desktop\Coding> npx create-react-app react-axios-crud
>> cd react-axios-crud
Need to install the following packages:
create-react-app@5.1.0
Ok to proceed? (y) y


Creating a new React app in C:\Users\samik\Desktop\Coding\react-axios-crud.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1324 packages in 1m

271 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.

Installing template dependencies using npm...

added 17 packages, and changed 1 package in 8s

271 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...


removed 1 package, and audited 1341 packages in 5s

271 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Created git commit.

Success! Created react-axios-crud at C:\Users\samik\Desktop\Coding\react-axios-crud
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
```

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya School of Engineering
(formerly K J Somaiya College of Engineering)

K. J. Somaiya College of Engineering, Mumbai-77
Department of Computer Engineering

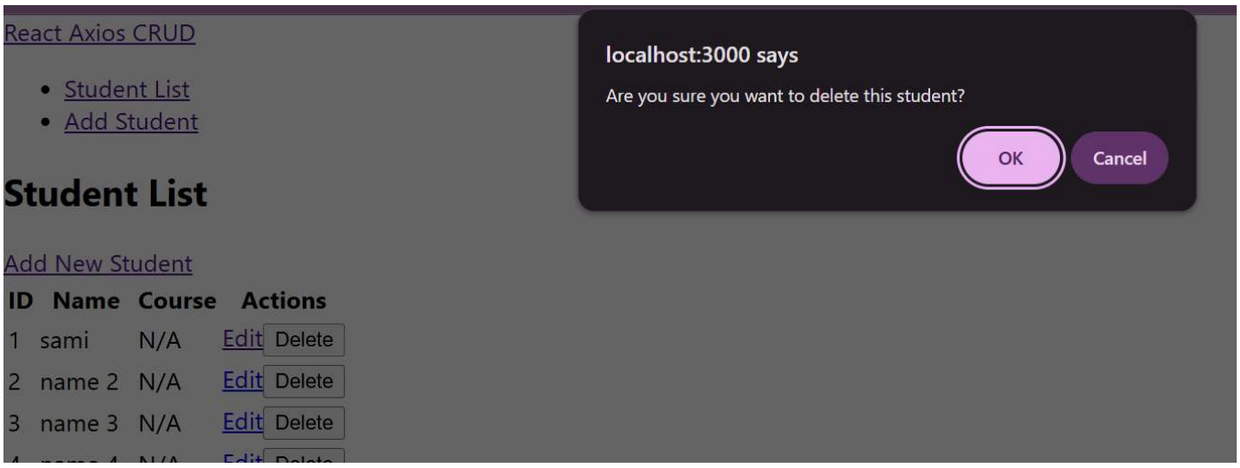
React Axios CRUD

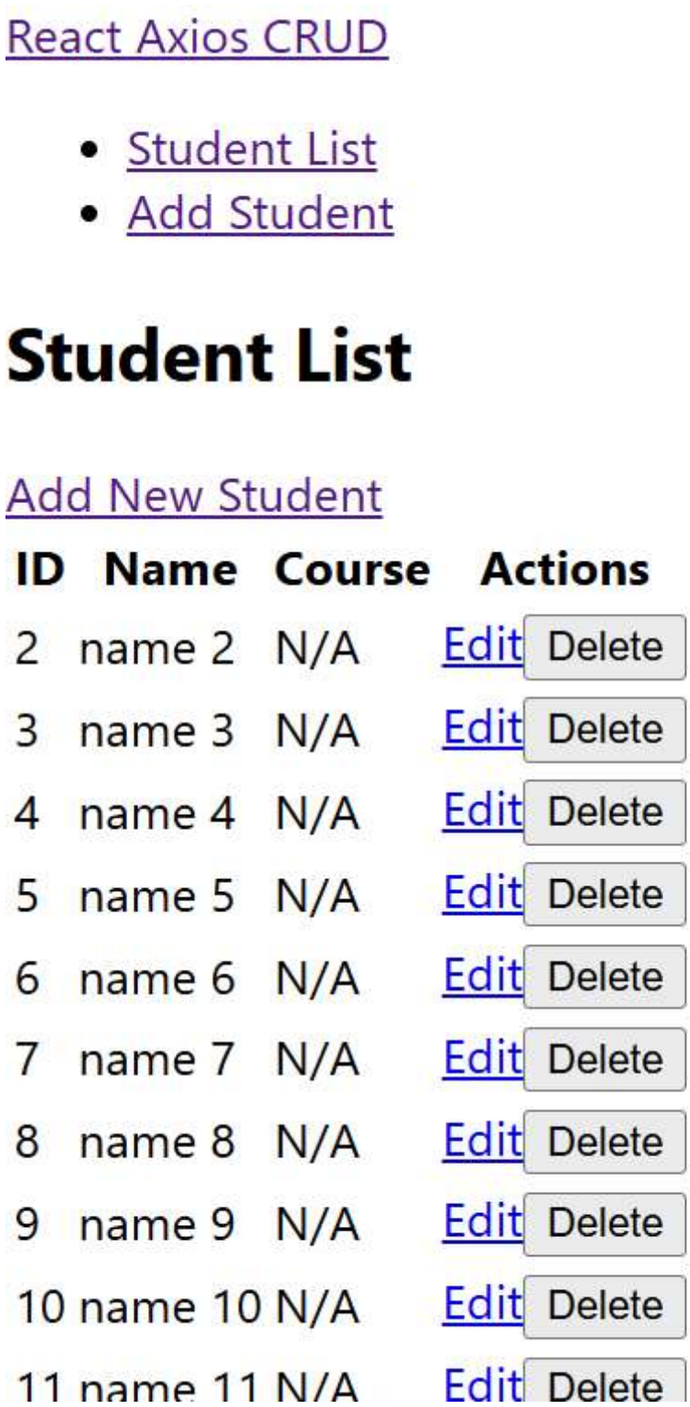
- Student List
- Add Student

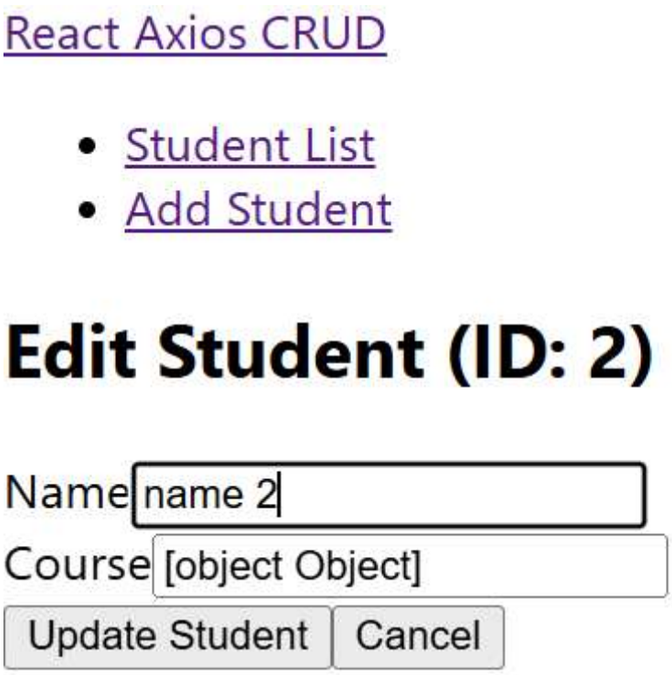
Student List

Add New Student

ID	Name	Course	Actions
1	sami	N/A	Edit Delete
2	name 2	N/A	Edit Delete
3	name 3	N/A	Edit Delete
4	name 4	N/A	Edit Delete
5	name 5	N/A	Edit Delete
6	name 6	N/A	Edit Delete
7	name 7	N/A	Edit Delete
8	name 8	N/A	Edit Delete
9	name 9	N/A	Edit Delete
10	name 10	N/A	Edit Delete
11	name 11	N/A	Edit Delete
12	name 12	N/A	Edit Delete
13	name 13	N/A	Edit Delete
14	name 14	N/A	Edit Delete


React Axios CRUD
Student List
Add Student
Student List
Add New Student
ID Name Course Actions
1 sami N/A Edit Delete
2 name 2 N/A Edit Delete
3 name 3 N/A Edit Delete
localhost:3000 says
Are you sure you want to delete this student?
OK
Cancel


React Axios CRUD
Student List
Add Student
Student List
Add New Student
ID Name Course Actions
2 name 2 N/A Edit Delete
3 name 3 N/A Edit Delete
4 name 4 N/A Edit Delete
5 name 5 N/A Edit Delete
6 name 6 N/A Edit Delete
7 name 7 N/A Edit Delete
8 name 8 N/A Edit Delete
9 name 9 N/A Edit Delete
10 name 10 N/A Edit Delete
11 name 11 N/A Edit Delete


React Axios CRUD
Student List
Add Student
Edit Student (ID: 2)
Name name 2
Course [object Object]
Update Student
Cancel

Edit Student (ID: 2)

Name Samiksha

Course [Maths, Science]

Update Student Cancel

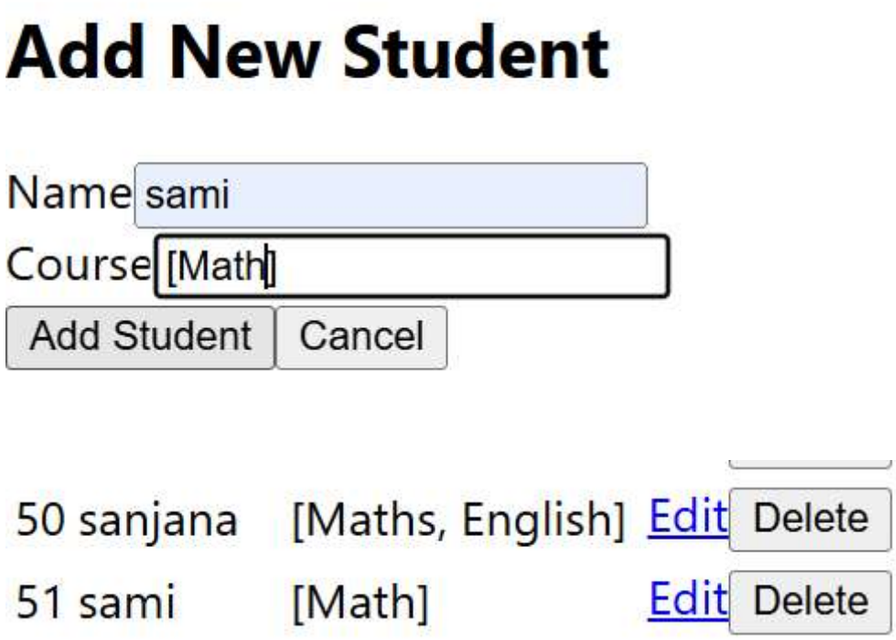
React Axios CRUD

- Student List
- Add Student

Student List

Add New Student

ID	Name	Course	Actions
2	Samiksha	[Maths, Science]	Edit Delete
3	name 3	N/A	Edit Delete
4	name 4	N/A	Edit Delete
5	name 5	N/A	Edit Delete
6	name 6	N/A	Edit Delete



Steps for execution:

Open two terminal windows in your project's root directory (`react-axios-crud`).
Start the Mock API Server (JSON Server) in the first terminal:

```
npm run server
```

Start the React Development Server in the second terminal:

```
npm start
```

Test the CRUD Operations:

- Read (GET): Verify the initial list of students is displayed on the homepage.
- Create (POST): Click "Add New Student," fill the form, and click "Add Student." The application should redirect to the list, and the new student should appear (demonstrates axios.post and useNavigate).

- Update (PUT): Click the "Edit" button next to a student. The form should pre-fill (demonstrates axios.get for initial state), and submitting the form should update the data and redirect back (demonstrates axios.put and useNavigate).
- Delete (DELETE): Click the "Delete" button next to a student. The student should instantly disappear from the list without a page reload (demonstrates axios.delete and local state update).

Conclusion:
The experiment successfully demonstrates the implementation of CRUD operations (Create, Read, Update, Delete) in a React Single Page Application (SPA) using the Axios library to interact with a Mock RESTful API (JSON Server).

Read (GET): The StudentList component effectively used axios.get within a useEffect hook to fetch and display the initial list of students, while managing loading and error states.

Create (POST): The AddStudent component used axios.post to send new student data and the useNavigate hook to redirect the user back to the list upon success.

Update (PUT): The EditStudent component utilized axios.get to pre-fill the form and axios.put to submit the modified data, also using useNavigate for redirection.

Delete (DELETE): The StudentList component implemented instant deletion using axios.delete and by updating the local state (setStudents) to reflect the change immediately without a full page reload.

This practical implementation illustrates the core concepts of front-end data management and RESTful API communication, directly achieving the learning objective CO 2 by demonstrating front-end application development using React and Axios.

Postlab questions:

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

In modern JavaScript and React development, there are primarily three ways to make HTTP requests to an API. All methods are **promise-based**, which simplifies asynchronous operations using async/await.

1. The Native fetch API

The **fetch API** is a modern, built-in JavaScript interface for making network requests, directly available in all modern browsers and Node.js environments.

- **Pros:** No need for external libraries, native standard.
- **Cons:** Does **not** automatically handle JSON response parsing (requires an extra .json() call), and it **only rejects the promise on network errors**, not on HTTP error statuses like 404 or 500 (you must manually check response.ok).

Axios is a third-party promise-based HTTP client that is highly popular for its feature set, which often simplifies the developer experience over the native fetch API.

- **Pros: Automatically transforms JSON data**, **rejects the promise on HTTP error statuses** (4xx, 5xx), includes features like request/response **interceptors**, and offers better **cancellation** support.
- **Cons:** Requires installing an external library.

3. Specialized Libraries (e.g., RTK Query / SWR)

These are not raw API clients like fetch or axios, but rather **data fetching and state management libraries** built specifically for React to handle complex API interactions, caching, and synchronization.

- **RTK Query (Redux Toolkit Query):** Built on top of Redux, it offers automatic caching, refetching, and state management for API data with zero-boilerplate.
- **SWR (Stale-While-Revalidate):** A lightweight library for data fetching focused on a cache-first approach.
- **Pros: Zero-boilerplate** for loading, error, and caching logic. **Automatic background revalidation** and optimized performance.
- **Cons:** Higher initial setup complexity, adds a new abstraction layer.