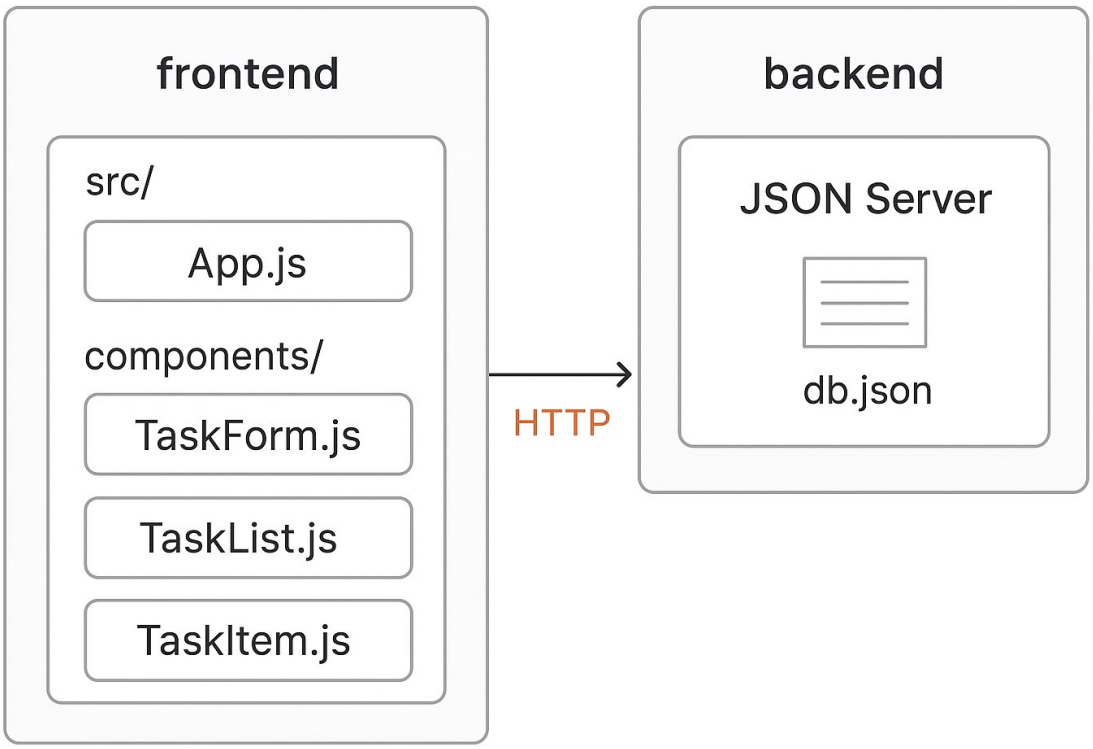
WEEK-8

Aim: To-Do List Application with React Hooks

Description:

The To-Do List Application is a simple yet powerful project built using React Hooks and a JSON Server backend to demonstrate full CRUD (Create, Read, Update, Delete) operations. The app allows users to manage their daily tasks efficiently by adding, viewing, updating, and deleting them through an intuitive user interface. It uses Axios to handle API requests between the React frontend and the local backend (http://localhost:5000/tasks), where all task data is stored in a JSON file. The project showcases key React concepts such as useState, useEffect, and event handling, while also emphasizing RESTful API communication. This setup makes it an excellent learning example for understanding modern front-end development, API integration, and basic backend data management using mock servers.



Tasks:

1. Create a functional React app with `useState` to manage the task list.
2. Allow users to add new tasks via a controlled form.
3. Implement list rendering to display tasks with keys.
4. Enable task completion toggling and deletion using event handlers.
5. Use conditional rendering to show a message when no tasks are present.
6. Implement all the crud operation using the data stored in a json file

Programs:

```
Week8/
│
├── backend/
│   ├── db.json              # JSON Server database file (stores tasks)
│
│
├── frontend/
│   │   ├── src/
│   │   ├── components/
│   │   │   ├── TaskForm.jsx     # Component for adding and editing tasks
│   │   │   ├── TaskList.jsx     # Component for rendering task list
│   │   │   └── TaskItem.jsx     # Component for individual task items
│   │   │
│   │   ├── App.jsx            # Main App component (uses useState, useEffect)
│   │   ├── App.css            # Styling for To-Do App
│   │   ├── api.jsx            # Axios configuration (CRUD functions)
│   │   ├──
│   │   │
```

Creating project

Creating Project:

Create one empty folder with name week8

And press in location path i.e cmd and you will get below path

1. **D:\Education\kluh\teaching\CC files\week8>**
2. **D:\Education\kluh\teaching\CC files\week8> code .**

3.
4. Click on terminal and create project as shown below
5. D:\Education\kluh\teaching\CC files\week8> npm create vite@latest week8
6.
7. D:\Education\kluh\teaching\CC files\week8> cd week8
8. D:\Education\kluh\teaching\CC files\week8>

or

npx create-react-app frontend

cd frontend

npm install axios

Programs:

TaskForm.jsx

```
import React, { useState } from "react";

const TaskForm = ({ onAddTask }) => {
  const [title, setTitle] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!title.trim()) return;
    onAddTask(title);
    setTitle("");
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        placeholder="Enter a new task..."
        value={title}
        onChange={(e) => setTitle(e.target.value)}
      />
      <button type="submit">Add Task</button>
    </form>
  );
};

export default TaskForm;
```

TaskList.jsx

```
import React from "react";
import TaskItem from "./TaskItem";

const TaskList = ({ tasks, onDelete, onToggle }) => {
  return (
    <ul>
      {tasks.map((task) => (
        <TaskItem
          key={task.id}
          task={task}
          onDelete={onDelete}
          onToggle={onToggle}
        />
      ))}
    </ul>
  );
};

export default TaskList;
```

TaskItem.jsx

```
import React from "react";


const TaskItem = ({ task, onDelete, onToggle }) => {

  return (

    <li className={task.completed ? "completed" : ""}>

      <span onClick={() => onToggle(task.id)}>{task.title}</span>

      <button onClick={() => onDelete(task.id)}>❌</button>

    </li>
```

```
  );
};

export default TaskItem;
```

App.jsx

```
import React, { useState, useEffect } from "react";
import { getTasks, addTask, deleteTask, updateTask } from "./api";
import TaskForm from "./components/TaskForm";
import TaskList from "./components/TaskList";
import "./App.css";

function App() {
  const [tasks, setTasks] = useState([]);

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

  const fetchTasks = async () => {
    const response = await getTasks();
    setTasks(response.data);
  };

  const handleAddTask = async (title) => {
    const newTask = { title, completed: false };
    const response = await addTask(newTask);
    setTasks([...tasks, response.data]);
  };

  const handleDeleteTask = async (id) => {
```

```
    await deleteTask(id);
    setTasks(tasks.filter((task) => task.id !== id));
  };

  const toggleTaskCompletion = async (id) => {
    const task = tasks.find((t) => t.id === id);
    const updatedTask = { ...task, completed: !task.completed };
    await updateTask(id, updatedTask);
    setTasks(tasks.map((t) => (t.id === id ? updatedTask : t)));
  };

  return (
    <div className="App">
      <h1>📝 To-Do List</h1>
      <TaskForm onAddTask={handleAddTask} />
      {tasks.length === 0 ? (
        <p>No tasks yet! Add your first one below 👇</p>
      ) : (
        <TaskList
          tasks={tasks}
          onDelete={handleDeleteTask}
          onToggle={toggleTaskCompletion}
        />
      )}
    </div>
  );
}

export default App;
```

App.css

```
.App {
  max-width: 500px;
  margin: 40px auto;
  text-align: center;
  font-family: Arial, sans-serif;
}

form {
  display: flex;
  justify-content: center;
  margin-bottom: 20px;
}
```

```
input {
  padding: 10px;
  width: 70%;
  border: 1px solid #ccc;
  border-radius: 6px;
}

button {
  margin-left: 8px;
  padding: 10px 14px;
  border: none;
  border-radius: 6px;
  background-color: #007bff;
  color: white;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}

ul {
  list-style: none;
  padding: 0;
}

li {
  background: #f1f1f1;
  margin: 8px 0;
  padding: 10px;
  border-radius: 6px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

li.completed span {
  text-decoration: line-through;
  color: gray;
}
```

api.jsx

```
import axios from "axios";
```

const API_URL = "http://localhost:5000/tasks";

export const getTasks = () => axios.get(API_URL);

export const addTask = (task) => axios.post(API_URL, task);

export const deleteTask = (id) => axios.delete(`${API_URL}/${id}`);

export const updateTask = (id, updatedTask) => axios.put(`${API_URL}/${id}`, updatedTask);

Operation	HTTP Method	Endpoint URL	Description / Example
Read (GET)	GET	http://localhost:5000/tasks	Fetches all tasks
Read (GET by ID)	GET	http://localhost:5000/tasks/1	Fetches a single task (ID = 1)
Create (POST)	POST	http://localhost:5000/tasks	Adds a new task
Update (PUT)	PUT	http://localhost:5000/tasks/1	Updates task with ID = 1
Delete (DELETE)	DELETE	http://localhost:5000/tasks/1	Deletes task with ID = 1

Step 1 — Run Backend:

cd backend

npx json-server --watch db.json --port 5000

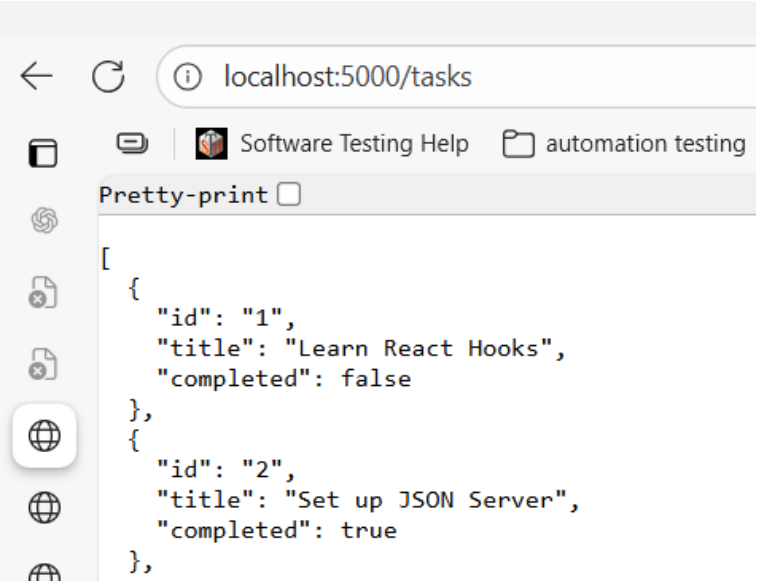
Step 2 — Run Frontend:

cd frontend

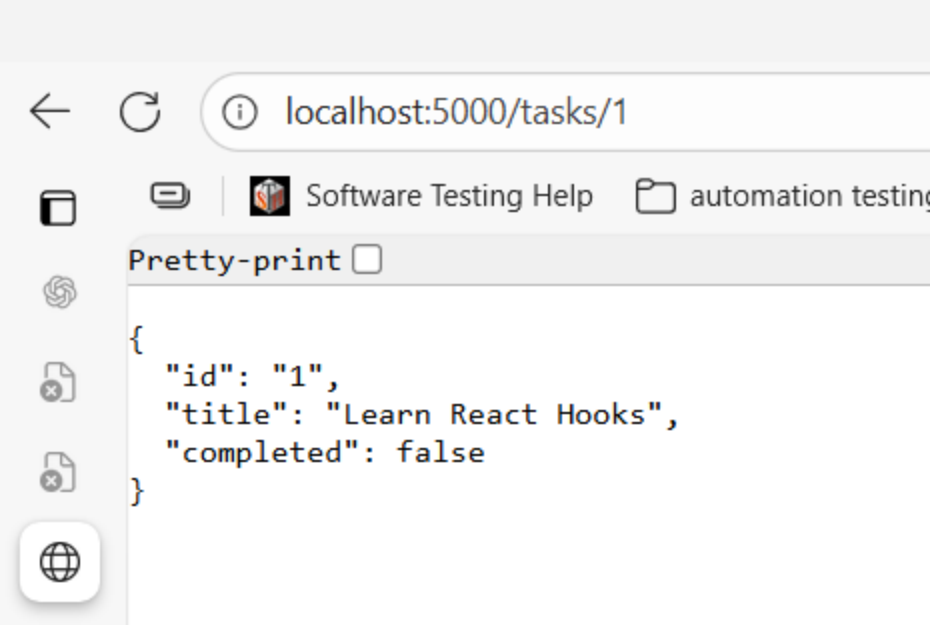
npm start

outputs:

http://localhost:5000/tasks



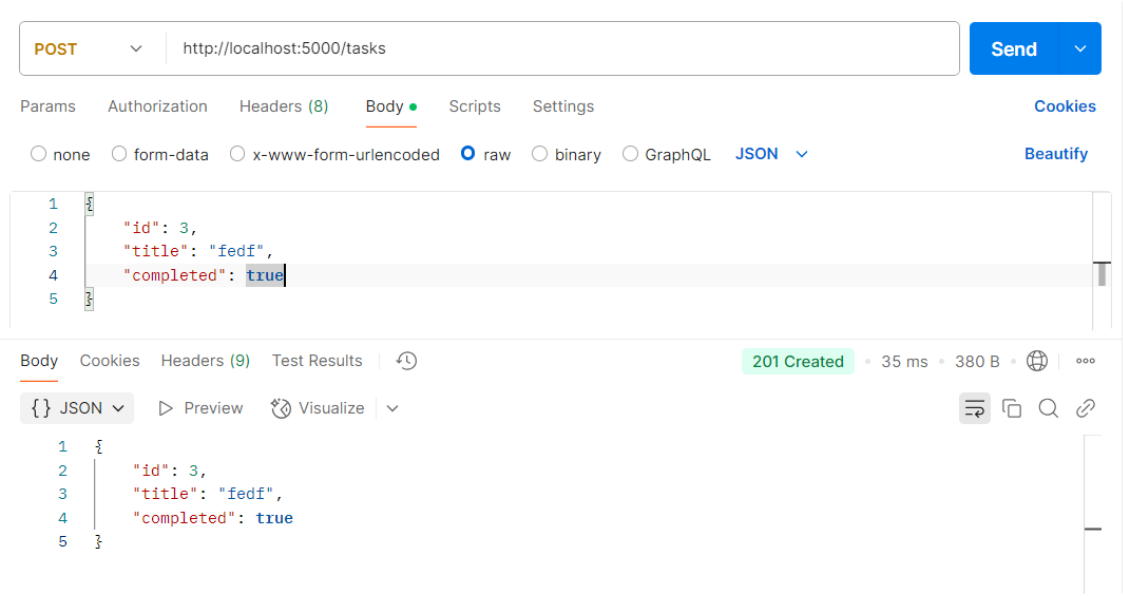
http://localhost:5000/tasks/1



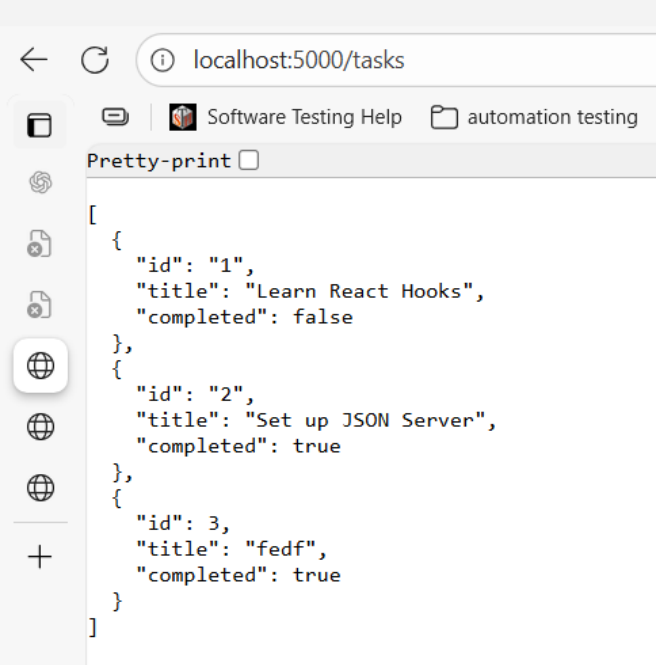
Use postman testing

Create (POST)

http://localhost:5000/tasks



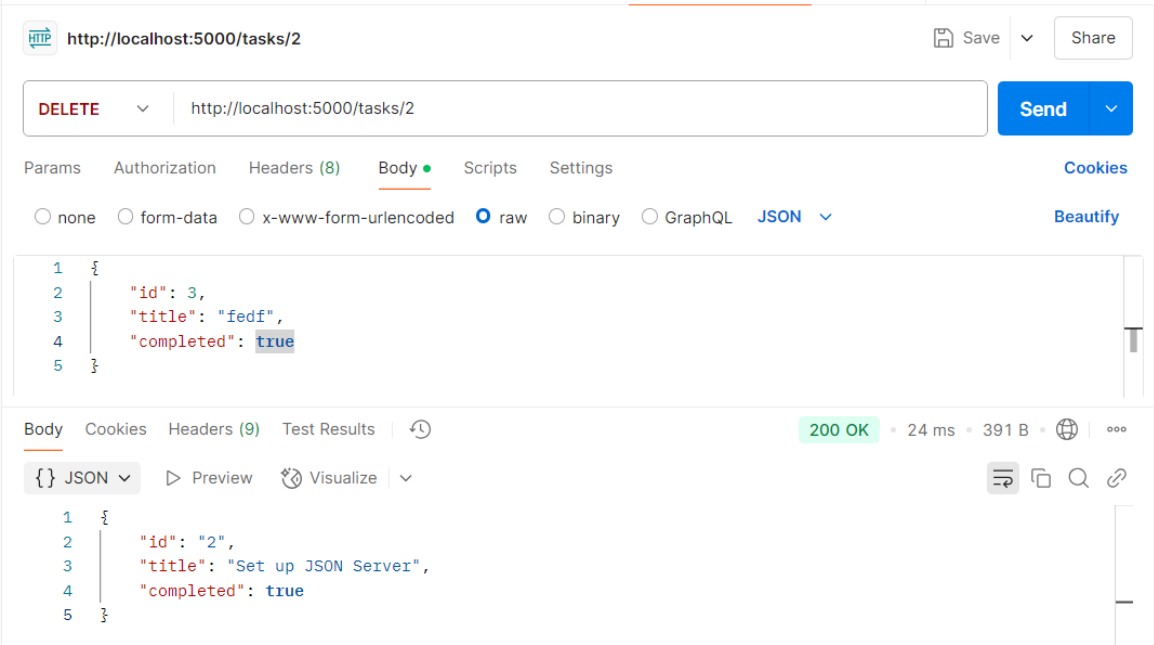
Successfully posted

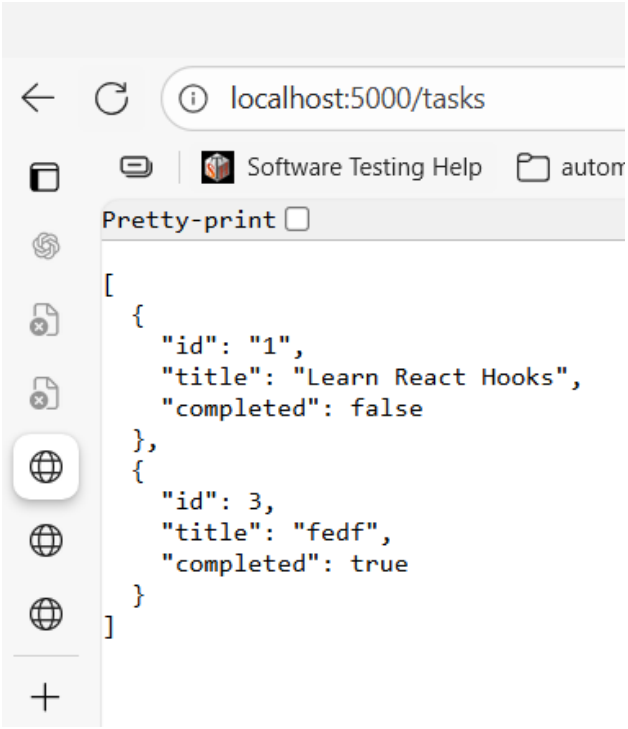


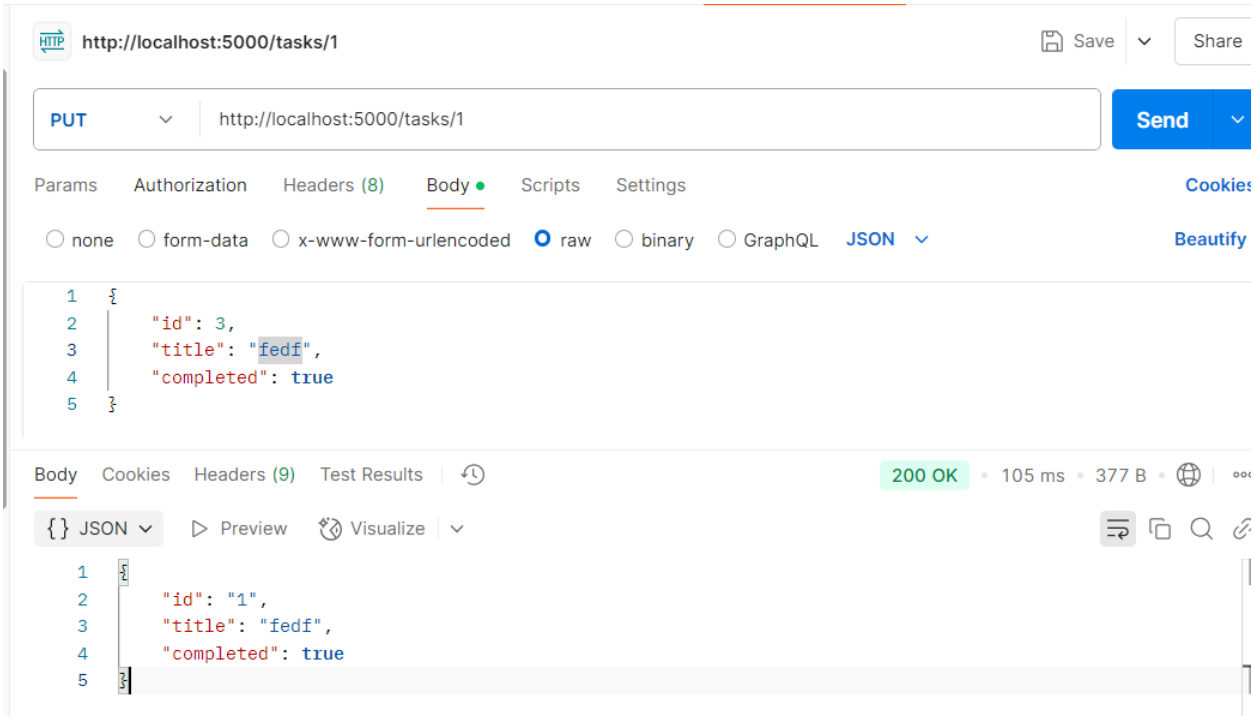
Delete (DELETE)

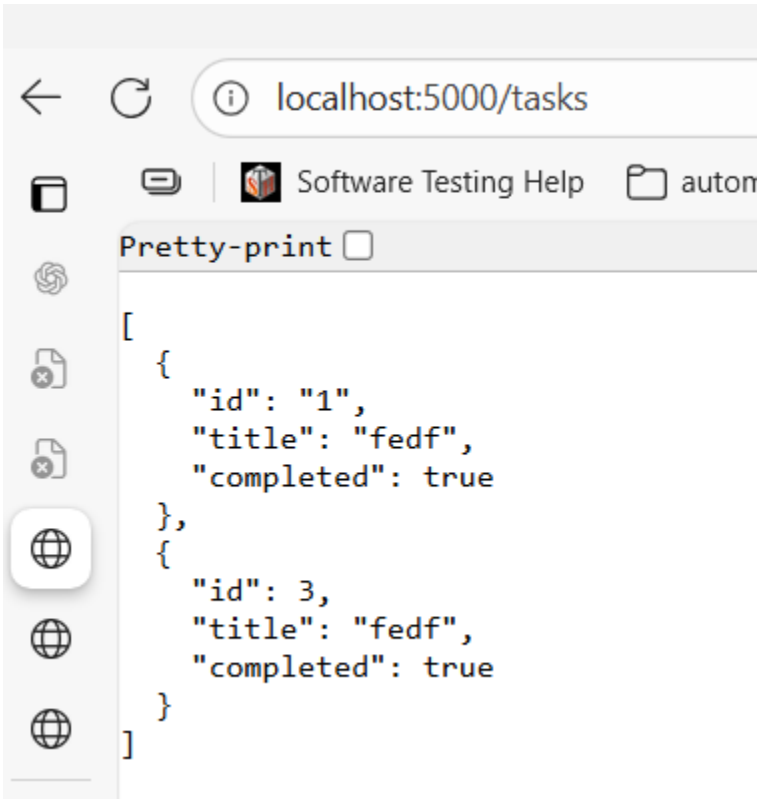
http://localhost:5000/tasks/1

Deletes task with ID = 2




localhost:5000/tasks
Software Testing Help
Pretty-print
[
{
"id": "1",
"title": "Learn React Hooks",
"completed": false
},
{
"id": 3,
"title": "fedf",
"completed": true
}
]


http://localhost:5000/tasks/1
Save
Share
PUT
http://localhost:5000/tasks/1
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id": 3,
"title": "fedf",
"completed": true
}
Body
Cookies
Headers (9)
Test Results
200 OK
105 ms
377 B
JSON
Preview
Visualize
{
"id": "1",
"title": "fedf",
"completed": true
}


localhost:5000/tasks
Software Testing Help
Pretty-print
[
{
"id": "1",
"title": "fedf",
"completed": true
},
{
"id": 3,
"title": "fedf",
"completed": true
}
]

Results:

The result of the To-Do List Application is a fully functional and responsive task management system that successfully performs all CRUD operations — creating, reading, updating, and deleting tasks — using a React frontend and a JSON Server backend. Users can easily add new tasks, view their list in real time, mark tasks as completed, edit task details, and delete them as needed. All changes are instantly reflected in the interface and persisted in the backend JSON file via Axios API calls. The project demonstrates seamless communication between the frontend and backend, validating the correctness and efficiency of the implemented RESTful architecture, React Hooks usage, and API integration.

VIVA QUESTIONS:

1. How does the useState hook help in managing the state of a task list in a functional React component?

   The useState hook helps in managing the state of a task list by allowing a React functional component to store and update the list dynamically. It keeps track of the current tasks and re-renders the component whenever the task list changes (e.g., adding or deleting a task).

2. What is a controlled component in React, and how is it used in forms to add new tasks?

   A controlled component in React is a form element (like an input) whose value is controlled by React state. In a to-do app, it is used to manage the input field for adding new tasks by updating the state on every keystroke and submitting the value through an event handler.

3. Why is it important to use a unique key prop when rendering a list of tasks in React?

   Using a unique key prop when rendering a list of tasks helps React identify which items have changed, been added, or removed. This improves rendering efficiency and prevents potential bugs during updates or re-renders.

4. How can event handlers be used to toggle task completion status or delete a task in a React component?

   Event handlers in React are used to perform actions such as toggling a task's completion status or deleting a task. For example, clicking a checkbox can trigger a function to update a task's "completed" state, while clicking a delete button can remove it from the task list.

5. What is conditional rendering in React, and how can it be used to display a message when the task list is empty?

   Conditional rendering in React means displaying content based on certain conditions. In a to-do app, it can be used to show a message like "No tasks available" when the task list is empty, and otherwise display the list of tasks when items are present.