Day 25 — Connecting React to Backend (Full CRUD with FastAPI)

Week 4 — Backend Integration & Deployment

Topic: Axios + Async/Await + API Service Layer + Error Handling

---

✅Why Connect React to a Backend?

Part	Role
Frontend	UI / What user interacts with
Backend	Data + Auth + Business Logic

➡ They communicate using **APIs** (HTTP requests)

✅Modern Best Practice (2025)

Method	Should Use?	Why
fetch()	❌No	messy JSON/error handling, no interceptors
Axios ✅	✅Yes	cleaner, handles JSON, supports tokens
Async/Await ✅	✅Required	readable async code

✅Install Axios

```
npm install axios
```

✅Create a Reusable Axios Instance

☸️src/api/axios.js

```
import axios from "axios";

const API = axios.create({
  baseURL: "http://localhost:8000", // Your FastAPI URL
});
```

```
// ✅ Optional: Auto attach token later
API.interceptors.request.use((req) => {
  const token = localStorage.getItem("token");
  if (token) req.headers.Authorization = `Bearer ${token}`;
  return req;
});

export default API;
```

✅Clean API calls ✅Scalable for authentication

✅CRUD API Service File

☸️src/api/tasks.js

```
import API from "./axios";

// ✅ Read All Tasks
export const getTasks = () => API.get("/tasks");

// ✅ Create New Task
export const addTask = (task) => API.post("/tasks", task);

// ✅ Update Task
export const updateTask = (id, task) => API.put(`/tasks/${id}`, task);

// ✅ Delete Task
export const deleteTask = (id) => API.delete(`/tasks/${id}`);
```

✅Centralized ✅Reusable ✅Cleaner Components

✅Using Async/Await in a Component

☸️src/components/Tasks.jsx

```
import { useEffect, useState } from "react";
import { getTasks, addTask, deleteTask } from "../api/tasks";

function Tasks() {
  const [tasks, setTasks] = useState([]);
  const [loading, setLoading] = useState(false);

  // ✅ Load tasks when component mounts
  useEffect(() => {
    fetchTasks();
  }, []);

  async function fetchTasks() {
    try {
      setLoading(true);
      const res = await getTasks();
      setTasks(res.data);
    } catch (error) {
      console.error("Failed to load tasks:", error);
    } finally {
      setLoading(false);
    }
  }

  async function handleAdd() {
    const newTask = { text: "New Task" };
    await addTask(newTask);
    fetchTasks();
  }

  async function handleDelete(id) {
    await deleteTask(id);
    fetchTasks();
  }

  return (
    <>
      <h2>Tasks from FastAPI ✅</h2>
      <button onClick={handleAdd}>Add Task</button>

      {loading && <p>Loading...</p>}

      <ul>
        {tasks.map(t => (
          <li key={t.id}>
            {t.text}
            <button onClick={() => handleDelete(t.id)}>❌</button>
          </li>
        ))}
      </ul>
    </>
  );
}

export default Tasks;
```

✅Real API Communication ✅UI Refresh

✅Error Handling Best Practice

```
catch (error) {
  if (error.response?.status === 401) {
    console.warn("Unauthorized – redirect to login soon");
  } else {
    console.error("API Error:", error);
  }
}
```

✅Secure ✅Debug-friendly ✅Handles auth cases

✅Axios vs Fetch (2025 Verdict)

Feature	fetch ❌	Axios ✅
Auto JSON parsing	❌	✅
Error messages	❌	✅
Interceptors	❌	✅
Timeout	❌	✅
Upload progress	❌	✅

☑️**Final: Use Axios for production apps**

🔝Exercise — Day 25 (Backend Connected Task Manager)

Feature	Requirement
✅Read tasks	GET /tasks
✅Add task	POST /tasks
✅Delete task	DELETE /tasks/{id}
✅Update Status	PUT /tasks/{id}
✅Loading UI	Show "Loading..."
✅Error UI	Show error message

✅Bonus Features

- Toast notifications (Success + Error)
- Material UI for UI elements & icons
- Optimistic UI (update UI before API response)

✅**Day 25 Completed!**