Day 3

Here's your 📅 **Day 3 – Frontend Integration (React + .NET Core)** schedule in the exact format you prefer.

You've already completed backend and boilerplate cleanup — today's goal is to connect your React UI with the backend and implement core task management features.

📅 Day 3 – React Frontend: Task List, Add & Delete

🎯 Goal: Display tasks, add new tasks, and delete them from React by integrating with the .NET Core API.

🕒 Time: ~5 hours

✅ Task 1: Create `services/api.js` for Axios Setup (~20 min)

🧠 Why it matters:

This helps **centralize all your API calls** — a clean pattern used in all real-world projects.

📘 Concepts explained:

- Axios instance
- Base URL reuse
- Request abstraction

🔨 Example:

```
// src/services/api.js
import axios from 'axios';

const api = axios.create({
  baseURL: 'https://localhost:5001/api', // Adjust port if needed
});
```

```
export default api;
```

✅ Task 2: Build `TaskList.jsx` to Fetch & Show Tasks (~1 hr)

🧠 Why it matters:

This is the **core display** logic of your app. Shows how React fetches data from your backend using `useEffect`.

📘 Concepts explained:

- `useState` to hold task data
- `useEffect` to fetch on load
- Conditional rendering
- `.map()` for lists

🔨 Example:

```
import { useState, useEffect } from 'react';
import api from '../services/api';

const TaskList = () => {
  const [tasks, setTasks] = useState([]);

  useEffect(() => {
    api.get('/task')
      .then(res => setTasks(res.data))
      .catch(err => console.error('Fetch error:', err));
  }, []);

  return (
    <ul>
      {tasks.map(t => (
        <li key={t.id}>{t.title} - {t.description}</li>
      ))}
    </ul>
  );
```

```
};

export default TaskList;
```

✅ Task 3: Create `AddTask.jsx` to Add New Tasks (~1 hr)

🧠 Why it matters:

This lets users **submit data to the backend** — teaches form handling and POST requests.

📘 Concepts explained:

- Controlled components (`value`, `onChange`)
- `onSubmit` handler
- Axios `POST` with form data

🔨 Example:

```
import { useState } from 'react';
import api from '../services/api';

const AddTask = () => {
  const [task, setTask] = useState({ title: '', description: '' });

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

  const handleSubmit = async (e) => {
    e.preventDefault();
    await api.post('/task', task);
    setTask({ title: '', description: '' }); // Reset form
  };

  return (
    <form onSubmit={handleSubmit}>
      <input name="title" value={task.title} onChange={handleChange} requi
```

```
red />
      <input name="description" value={task.description} onChange={handl
eChange} />
      <button type="submit">Add Task</button>
    </form>
  );
};

export default AddTask;
```

✅ Task 4: Implement Delete Task Feature in `TaskList.jsx` (~45 min)

🧠 Why it matters:

Lets users remove tasks — covers DELETE request and state manipulation (`.filter()`).

📘 Concepts explained:

- Axios `DELETE`
- Updating local state after deletion

🔨 Example:

```
const deleteTask = async (id) => {
  await api.delete(`/task/${id}`);
  setTasks(prev => prev.filter(task => task.id !== id));
};

{/* Inside .map() return: */}
<button onClick={() => deleteTask(t.id)}>Delete</button>
```

✅ Task 5: Combine Components in `App.jsx` (~15 min)

🧠 Why it matters:

This integrates the entire app — everything will be visible in one page.

📘 Concepts explained:

- React component composition
- Import/export best practices

🔨 Example:

```
import AddTask from './components/AddTask';
import TaskList from './components/TaskList';

const App = () => {
  return (
    <div>
      <h1>Task Manager</h1>
      <AddTask />
      <TaskList />
    </div>
  );
};

export default App;
```

✅ Task 6: Optional Styling with Bootstrap/Tailwind (~30 min)

🧠 Why it matters:

Looks matter! Recruiters and users will judge the **UI quality** as well.

📘 Concepts explained:

- Bootstrap grid & form classes
- Tailwind utility-first design

🔨 Example (Bootstrap):

```
npm install bootstrap
```

```
// main.jsx
import 'bootstrap/dist/css/bootstrap.min.css';
```

```
<form className="mb-3">
  <input className="form-control mb-2" />
  <button className="btn btn-primary">Add</button>
</form>
```

✅ Task 7: Push Final Code to GitHub (~30 min)

🧠 Why it matters:

You need this for **resume links**, recruiter proof, and deployment.

📘 Concepts explained:

- Git staging and commit best practices
- Branching (if needed)

🔨 Example:

```
git add .
git commit -m "feat: integrated frontend with backend, task CRUD complet
e"
git push
```

✅ By End of Day 3, You Will Have:

Feature	Status
Axios config setup	✅
Task fetching UI	✅
Task form to add new tasks	✅
Delete task from UI	✅
Bootstrap/Tailwind styling	✅
Full app wired + working in browser	✅

Feature	Status
Code pushed to GitHub	✅

Let me know when you're ready for **Day 4**, where we'll focus on:

- Hosting backend (Render)
- Hosting frontend (Vercel)
- Making resume-ready project links 🌍🚀