🧾 Project Documentation — Assignment 1: Task Manager

📘 Submitted To:

Pathlock Pvt. Ltd.

👨‍💻 Submitted By:

Sourav [2104105]

📅 Submission Date:

31 October 2025

🔷 1. Project Overview

This project is a **Task Management Web Application** built using:

- **ASP.NET Core (C#)** for the backend (RESTful API)
- **React + TypeScript** for the frontend
- **TailwindCSS** for responsive and modern UI design

It allows users to:

- Add tasks
- View task list
- Toggle task completion
- Delete tasks

The application demonstrates **full-stack integration**, **CORS handling**, and **frontend-backend communication** using **Axios**.

🔷 2. Objectives

The main objectives of this assignment are:

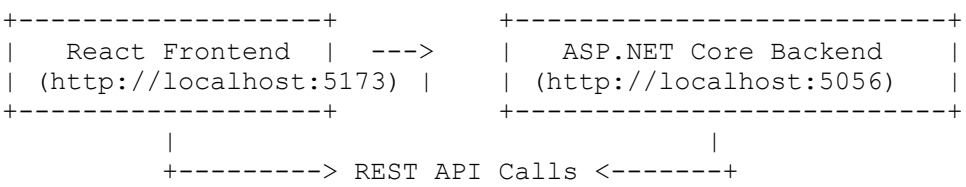
- To implement a **simple full-stack CRUD application**
- To understand **REST API design** using ASP.NET Core
- To integrate **frontend and backend** with **Axios**
- To apply **modern UI frameworks** (TailwindCSS) for responsive design

---

🔷 3. Tools & Technologies Used

Category	Technology
Frontend	React + TypeScript + Vite
Styling	TailwindCSS
Backend	ASP.NET Core 8 Minimal API
Language	TypeScript, C#
Libraries	Axios (for API calls)
Environment	Node.js 18+, .NET SDK 8.0
IDE/Editor	VS Code, Visual Studio / Rider

🔷 4. System Architecture



The frontend communicates with the backend via HTTP requests using Axios.
The backend maintains an **in-memory task list** and exposes **CRUD endpoints**.

🔷 5. Folder Structure

```
TaskManager/
│
├── backend/
│   ├── Program.cs
│   ├── backend.csproj
│
├── frontend/
│   ├── src/
│   │   ├── api/tasksApi.ts
│   │   ├── App.tsx
│   │   ├── index.css
│   │   └── components/TaskList.tsx
│   ├── package.json
│   └── tailwind.config.ts
│
└── Documentation_Assignment1.pdf
```

🔷 6. Backend Overview

✅ Features

- Exposes REST API using ASP.NET Core Minimal APIs
- Stores tasks in a C# list
- Supports CORS for React frontend

✅ API Endpoints

Method	Endpoint	Description
GET	`/api/tasks`	Retrieve all tasks
POST	`/api/tasks`	Create a new task
PUT	`/api/tasks/{id}/toggle`	Toggle completion status
DELETE	`/api/tasks/{id}`	Delete task

✅ Backend Setup

```
cd backend
dotnet restore
dotnet build
dotnet run
```

Backend runs on:
http://localhost:5056

🔷 7. Frontend Overview

✅ Features

- React-based interactive UI
- TypeScript interfaces for strong typing
- Axios API integration
- TailwindCSS for styling

✅ Frontend Setup

```
cd frontend
npm install
npm run dev
```

Frontend runs on:
http://localhost:5173

🔷 8. Workflow

1️⃣ **User adds a new task** → request sent via Axios to `/api/tasks`
2️⃣ **Backend saves the task** and returns it to the frontend
3️⃣ **UI refreshes** with updated task list
4️⃣ **User can toggle or delete** any task
5️⃣ All updates happen instantly via API calls

🔷 9. Code Snippets

🔸 Backend (Program.cs)

```
app.MapPost("/api/tasks", (TaskCreateDto newTask) =>
{
    var nextId = tasks.Count > 0 ? tasks.Max(t => t.Id) + 1 : 1;
    var task = new TaskItem { Id = nextId, Description =
newTask.Description, IsCompleted = false };
    tasks.Add(task);
    return task;
});
```

🔸 Frontend (tasksApi.ts)

```
export const createTask = async (description: string) => {
  const res = await api.post("/api/tasks", { description });
  return res.data;
};
```

🔸 Frontend UI (App.tsx)

```
<button
  className="bg-blue-500 hover:bg-blue-600 px-4 py-2 rounded text-white"
  onClick={handleAdd}
>
  Add Task
</button>
```

🔷 10. Sample API Responses

➤ GET `/api/tasks`

```
[
  { "id": 1, "description": "Sample task", "isCompleted": false }
]
```

➤ **POST `/api/tasks`**

```
{ "id": 2, "description": "New Task", "isCompleted": false }
```

🔷 11. Results

- Successfully implemented **CRUD Task Manager**
- Seamless communication between **React frontend** and **.NET backend**
- Responsive and modern interface with **TailwindCSS**
- Demonstrated strong understanding of **API development**, **frontend integration**, and **project structuring**

🔷 12. Future Enhancements

✅ Add task **due dates**
✅ Implement **task categories**
✅ Save data in **SQLite / SQL Server** instead of memory
✅ Add **user authentication**

🔷 13. Conclusion

This project demonstrates a clear understanding of:

- Backend REST API development using ASP.NET Core
- Frontend integration using React + Axios
- UI design using TailwindCSS
- Real-world full-stack development workflow

🔷 14. References

- React Documentation
- .NET 8 Documentation
- TailwindCSS Docs
- Axios Docs

✅ **End of Documentation**