Notely - Single-Tier Architecture Report

1. Architecture Overview

What is Single-Tier Architecture?

Single-tier architecture refers to a system where the frontend, backend, and database reside within the same project or environment. In this case, Notely, a note-taking application, follows this architecture by integrating the user interface, business logic, and database within a single ASP.NET Core Web API project.

Project Components

- Frontend (UI Layer): HTML, CSS, and JavaScript (served from wwwroot in the Web API project).
- Backend (Business Logic Layer): ASP.NET Core Web API to handle requests and data processing.
- Database Layer: SQL Server integrated with Entity Framework Core for data storage.

How It Works

1. Users access the Notely web interface served directly from the backend (wwwroot/index.html).
2. JavaScript interacts with the Web API using AJAX/fetch requests.
3. ASP.NET Core Web API processes the requests and communicates with the database.
4. Database stores and retrieves notes, returning results to the API.
5. Frontend updates the UI dynamically based on responses.

2. Implementation Steps

Step 1: Set Up the Web API Project

Create an ASP.NET Core Web API project.

Enable static file hosting (app.UseStaticFiles();).

Step 2: Define the Database and Model

Create ApplicationDbContext.cs using Entity Framework Core.

Define the Note model with properties Id, Title, and Content.

Step 3: Build API Endpoints (NotesController.cs)

GET /api/notes → Retrieve all notes.

POST /api/notes → Add a new note.

PUT /api/notes/{id} → Update a note.

DELETE /api/notes/{id} → Remove a note.

Step 4: Develop Frontend in wwwroot/

Create index.html, styles.css, and script.js.

Use fetch() in JavaScript to interact with the API.

Step 5: Run & Test the Application

Start the API using dotnet run.

Open http://localhost:5001/index.html to interact with the app.

3. Advantages and Challenges Faced

Advantages

- Simple Deployment – Everything runs within a single project, making deployment straightforward.
- Fast Development – No need to manage separate frontend and backend projects.
- Direct Communication – No network overhead between layers.

Challenges Faced

- Limited Scalability – As the application grows, maintaining all components in a single project can become complex.
- Security Concerns – Exposing everything in one system increases security risks.
- Code Maintainability – Mixing frontend, backend, and database logic in one project can make debugging difficult.

4. Comparison with Other Architectures

Architecture	Description	Pros	Cons
Single-Tier	All components in one system	Simple, fast, easy to develop	Hard to scale, security risks
Two-Tier	Frontend and Backend are separate	Better performance, structured	Increased complexity, potential latency
Three-Tier	UI, API, and Database separated	Highly scalable, secure	More complex to implement

Conclusion

Notely successfully implements a Single-Tier Architecture for a simple and efficient note-taking system. However, for larger applications requiring scalability and maintainability, a Two-Tier or Three-Tier Architecture would be preferable.