

Software Architecture Diagrams

Contents

Client Portal Web Application	2
1. System Architecture Diagram	2
2. Database Entity Relationship Diagram (ERD)	3
3. Authentication Flow Diagram	4
4. User Role Hierarchy	5
5. API Request Flow	6
6. Component Architecture	7
7. File Upload Flow	8
8. Real-Time Notification Flow	9
9. Project Workflow Diagram	9
10. Task Status State Machine	10
11. Deployment Architecture	10
12. Sequence Diagrams	11
12.1 Complete Project Creation Flow	11
12.2 Task Comment and Notification Flow	12
12.3 File Download Flow	13
12.4 Admin User Suspension Flow	14
Class Diagram	15
Use Case Diagram	16
Technology Stack Diagram	17
Summary	17
How to Use These Diagrams	17
Rendering These Diagrams	18

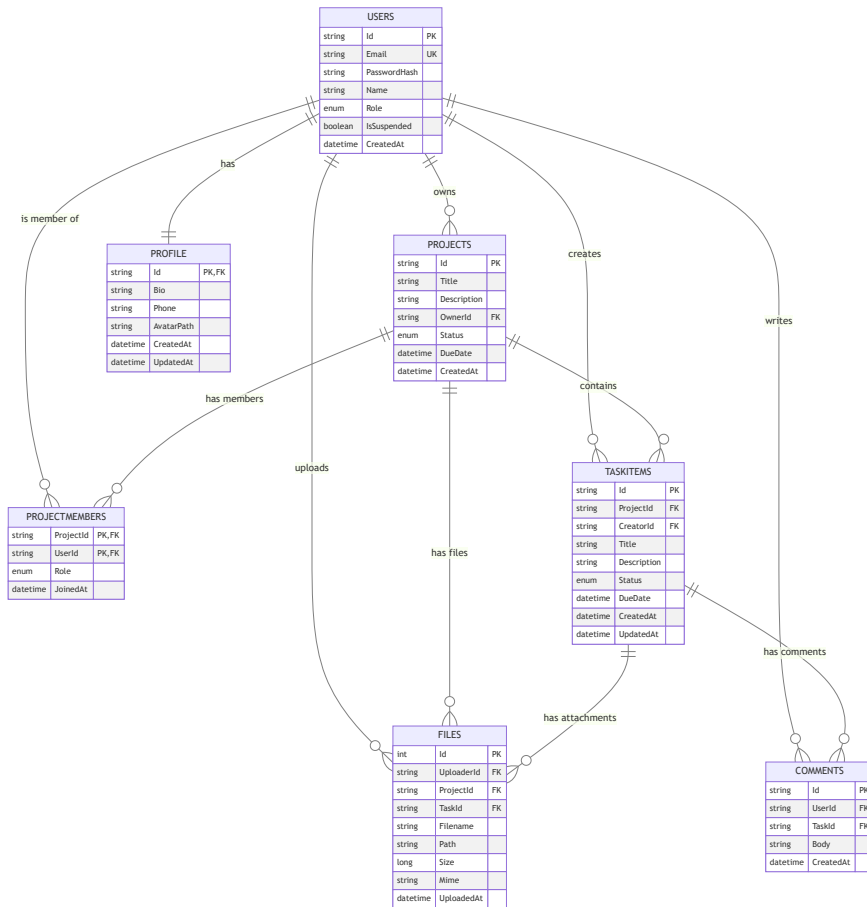
Version: 1.0
Date: 2025-10-26
Author: nizarLubbad

The diagram illustrates a layered architecture for a client portal, organized into six main layers:

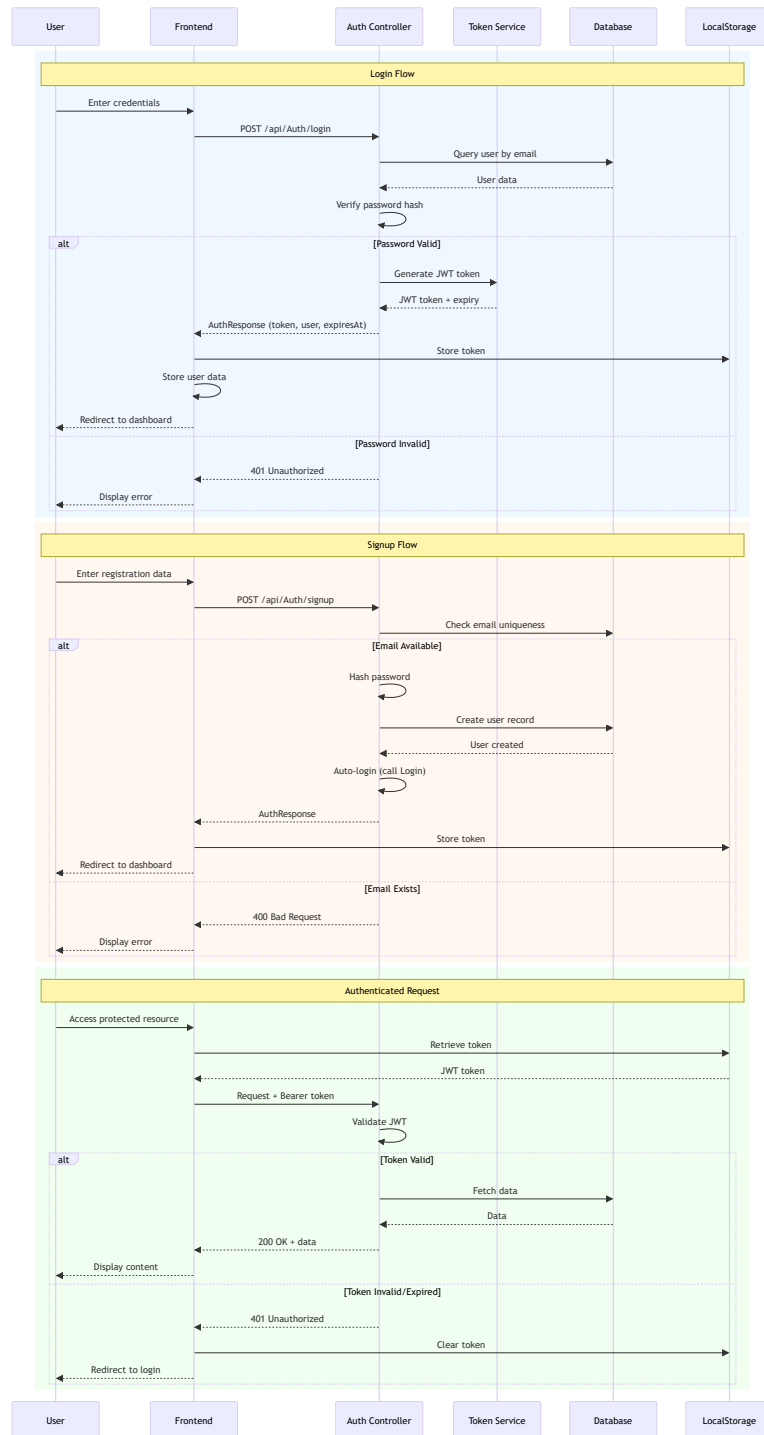
- Client Layer:** Contains the **React Frontend** (TypeScript + Vite). It connects to **File Management**, **Task Management**, **Project Management**, **Dashboard Components**, and **Login/Signup Pages**.
- Network Layer:** Contains **WebSocket SignalR** and **HTTPS/HTTP**. **File Management** and **Task Management** connect to **WebSocket SignalR**. **Project Management**, **Dashboard Components**, and **Login/Signup Pages** connect to **HTTPS/HTTP**.
- API Layer (ASP.NET Core):** Contains **Middleware**, **Exception Handling**, **CORS Policy**, **JWT Authentication**, **Controllers**, **AdminController**, **UserController**, **FilesController**, **TasksController**, **ProjectsController**, and **AuthController**. **Middleware** connects to **Exception Handling**, **CORS Policy**, and **JWT Authentication**. **Controllers** connects to **AdminController**, **UserController**, **FilesController**, **TasksController**, **ProjectsController**, and **AuthController**.
- Business Logic Layer:** Contains **Services**, **ProjectService**, **NotificationService**, **FileService**, and **TokenService**. **Services** connects to **ProjectService**, **NotificationService**, **FileService**, and **TokenService**.
- Data Access Layer:** Contains **Entity Framework Core**, **AppDbContext**, and **Migrations**. **Entity Framework Core** connects to **AppDbContext** and **Migrations**.
- Data Storage:** Contains **SQLite Database** (clientportal.db) and **File System** (wwwroot/uploads). **AppDbContext** connects to **SQLite Database**. **FileService** and **TokenService** connect to **File System**.

Additional connections include **SignalR Hub** (Real-Time Communication) connecting to **NotificationHub**, which then connects to **NotificationService** in the Business Logic Layer.

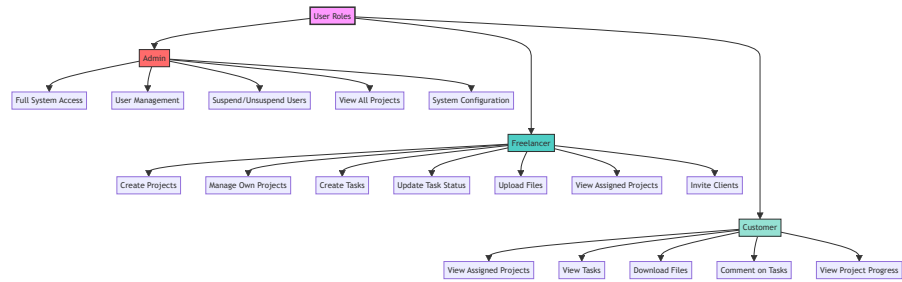
2. Database Entity Relationship Diagram (ERD)



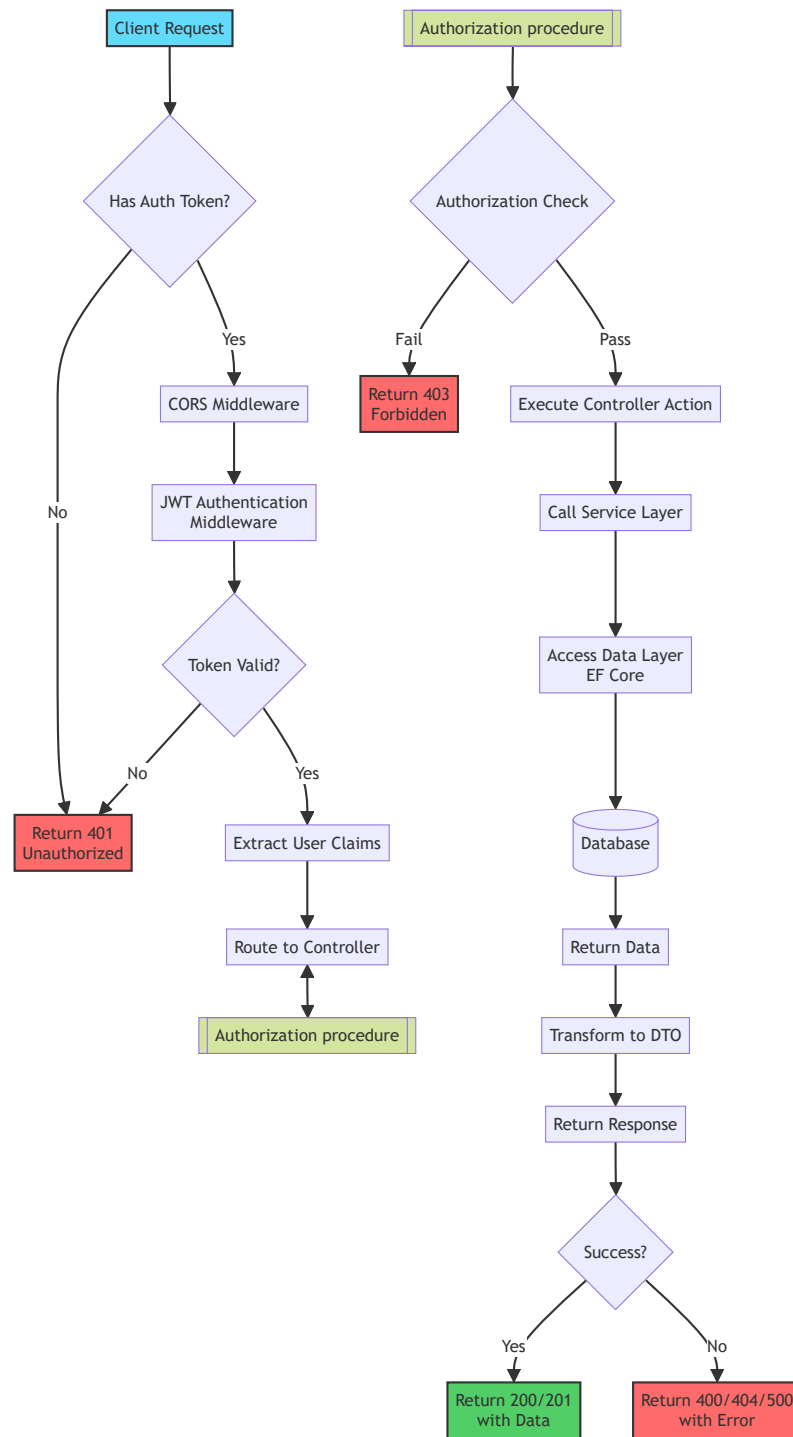
3. Authentication Flow Diagram



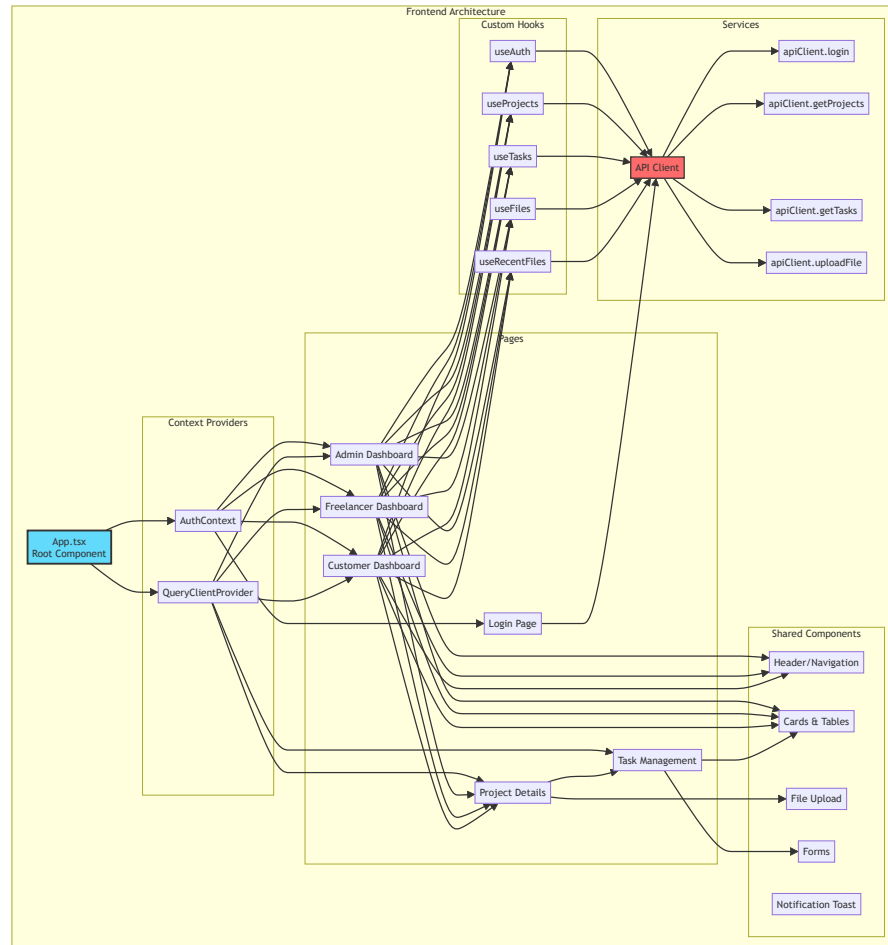
4. User Role Hierarchy



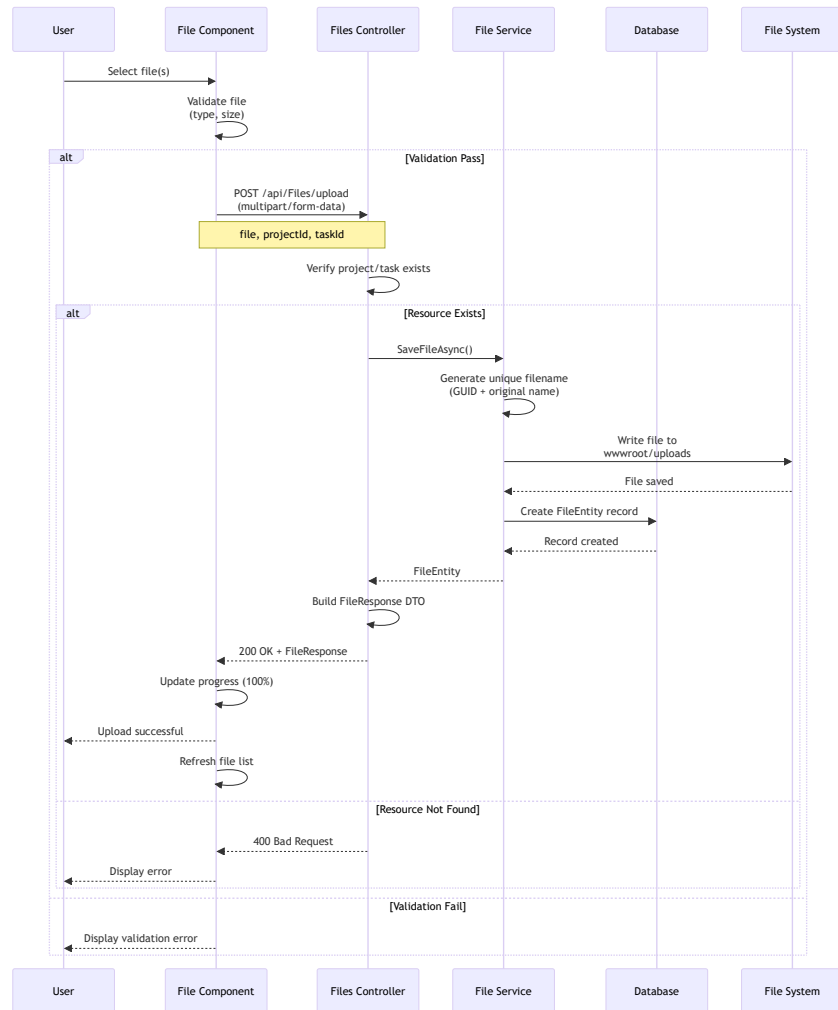
5. API Request Flow



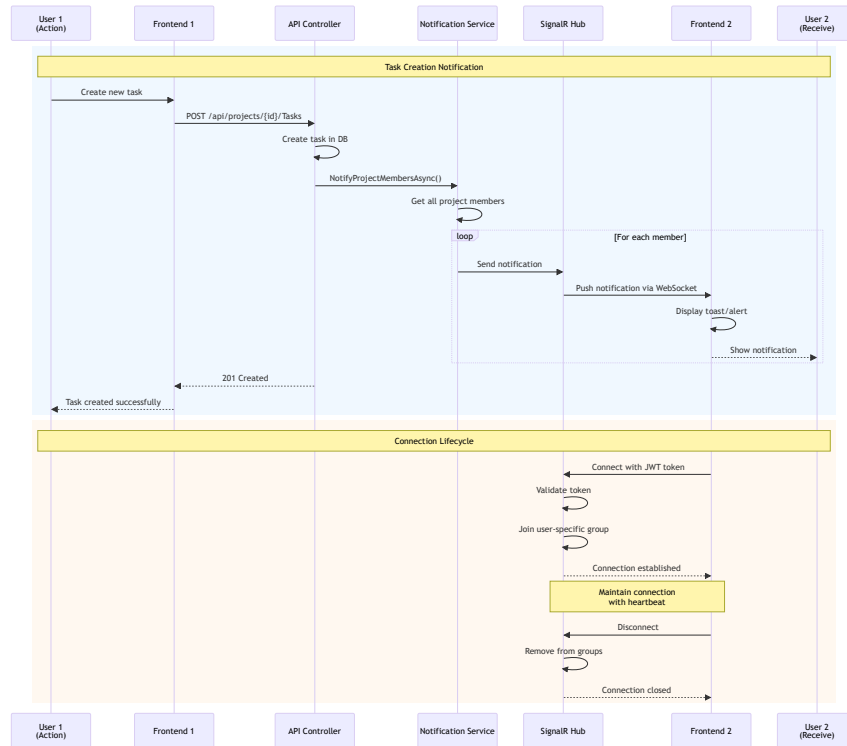
6. Component Architecture



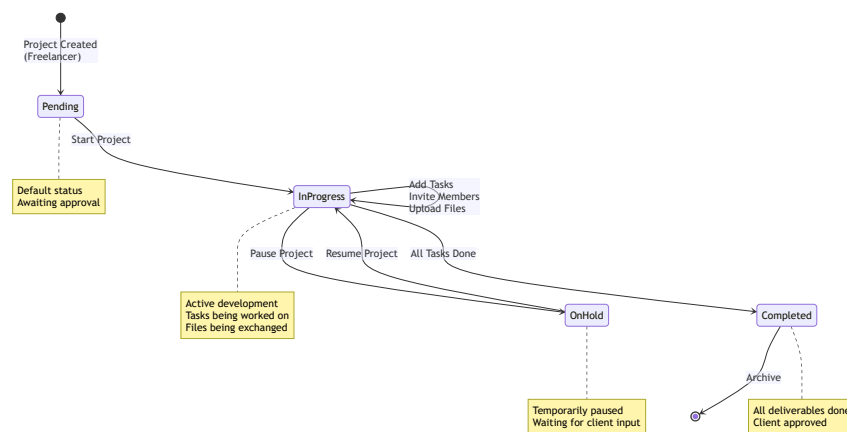
7. File Upload Flow



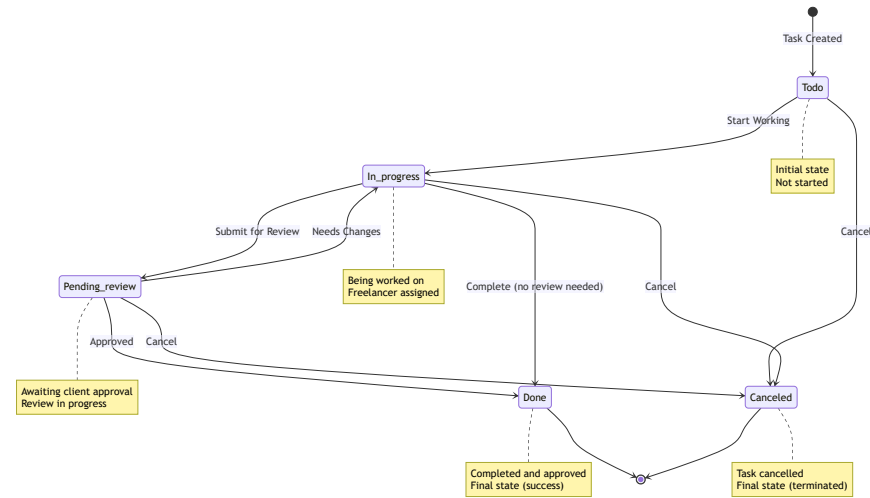
8. Real-Time Notification Flow



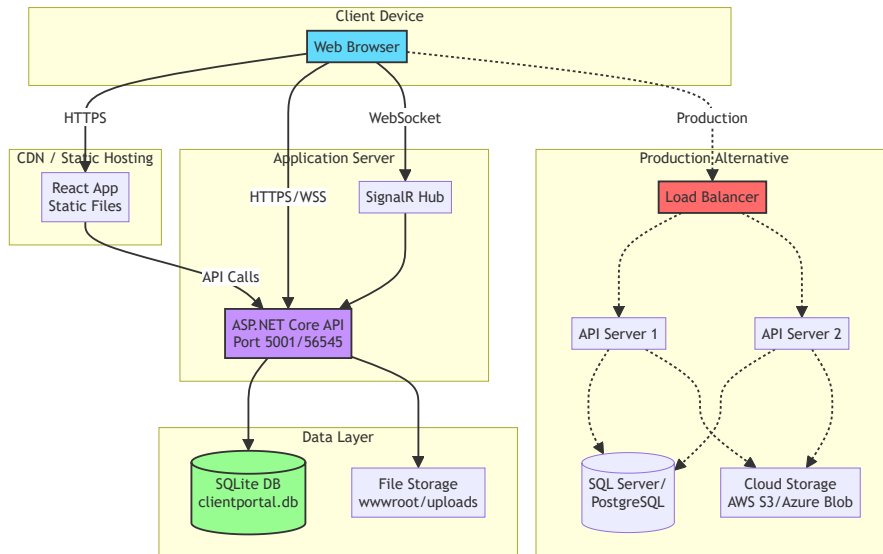
9. Project Workflow Diagram



10. Task Status State Machine

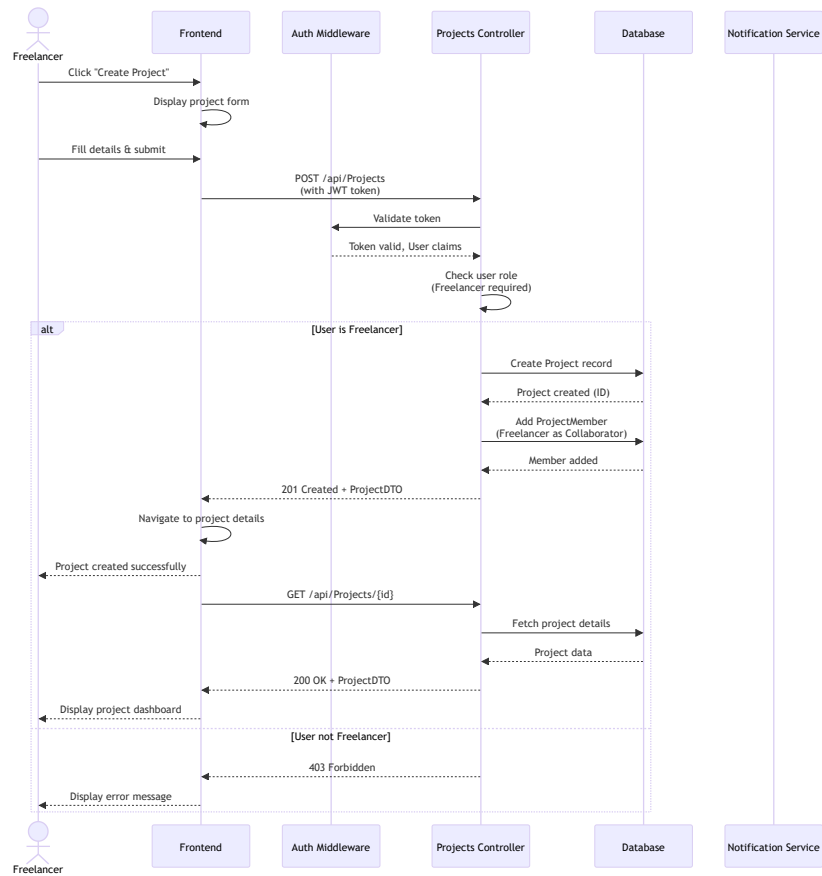


11. Deployment Architecture

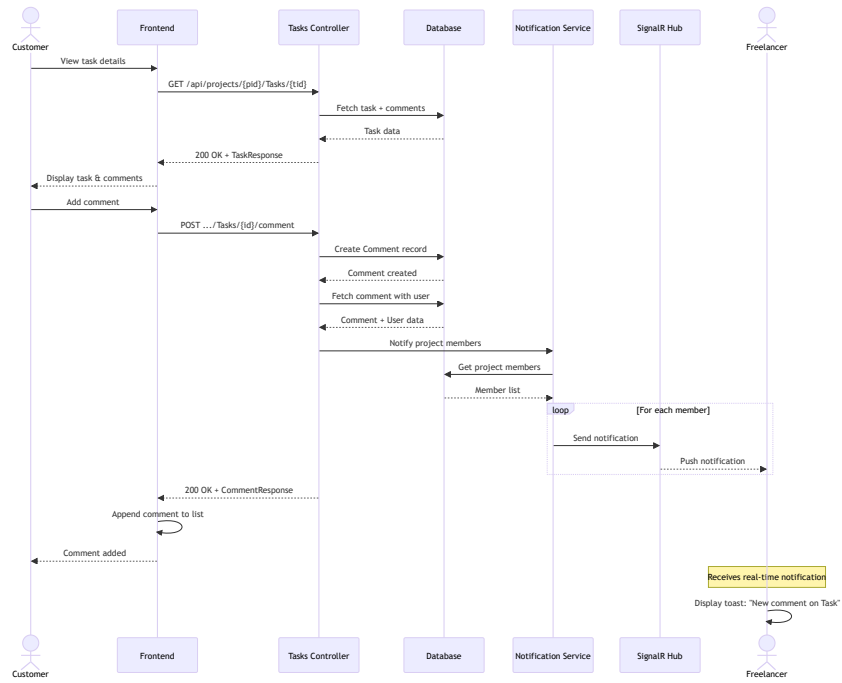


12. Sequence Diagrams

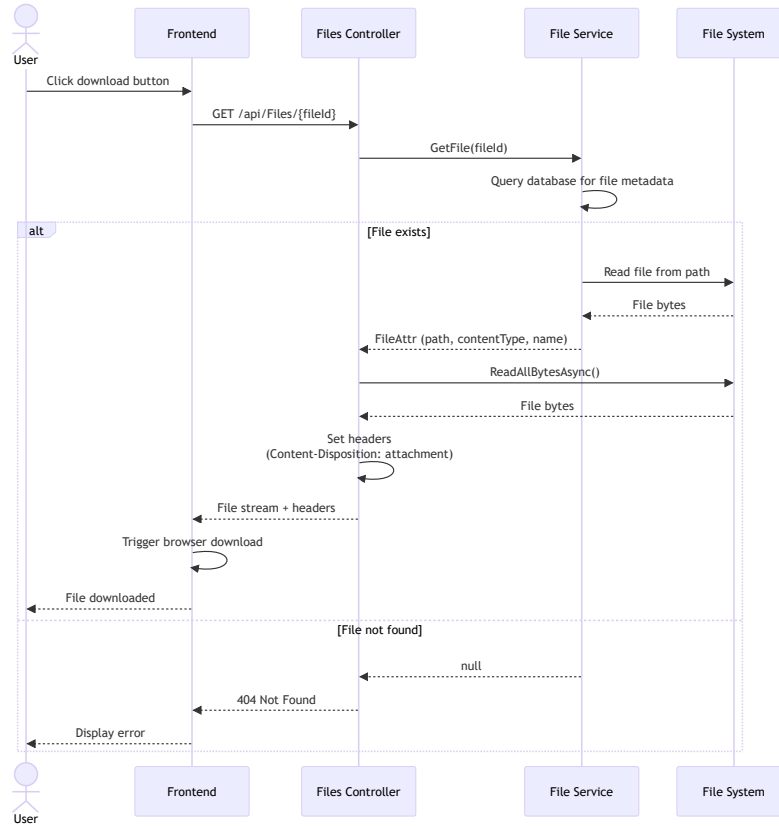
12.1 Complete Project Creation Flow



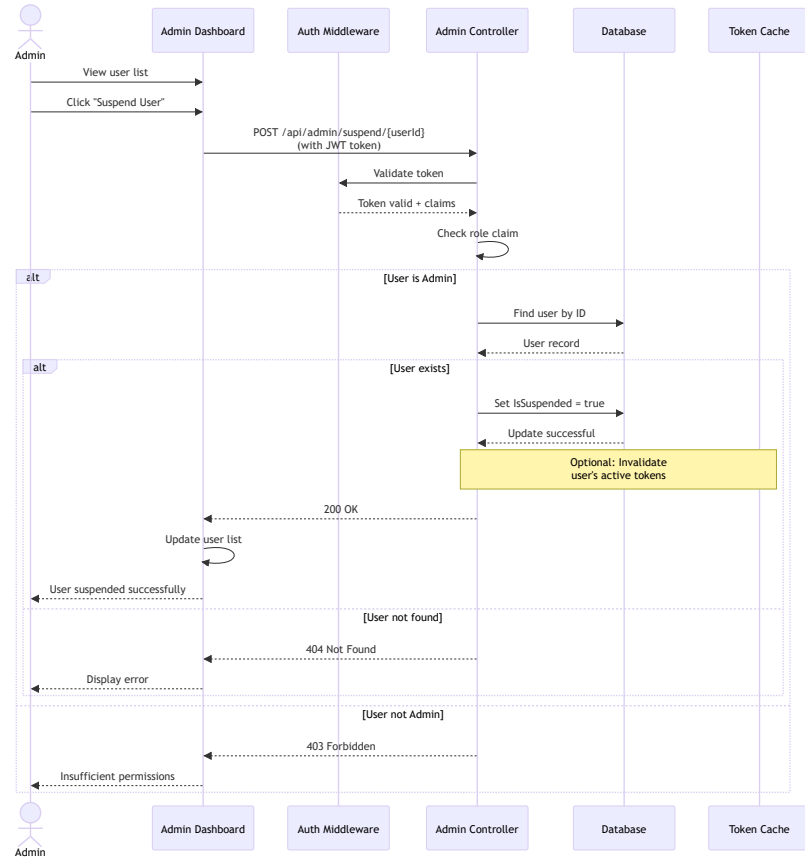
12.2 Task Comment and Notification Flow



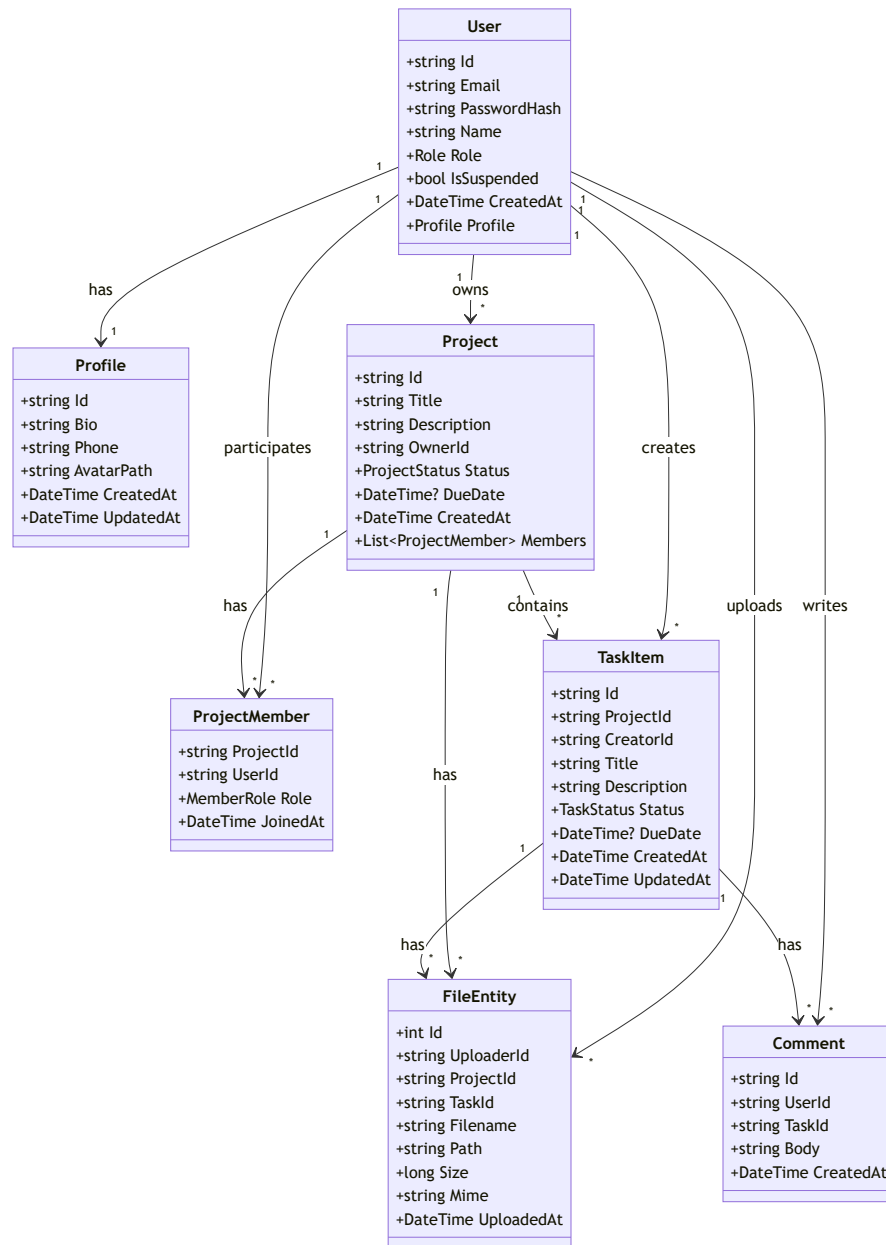
12.3 File Download Flow



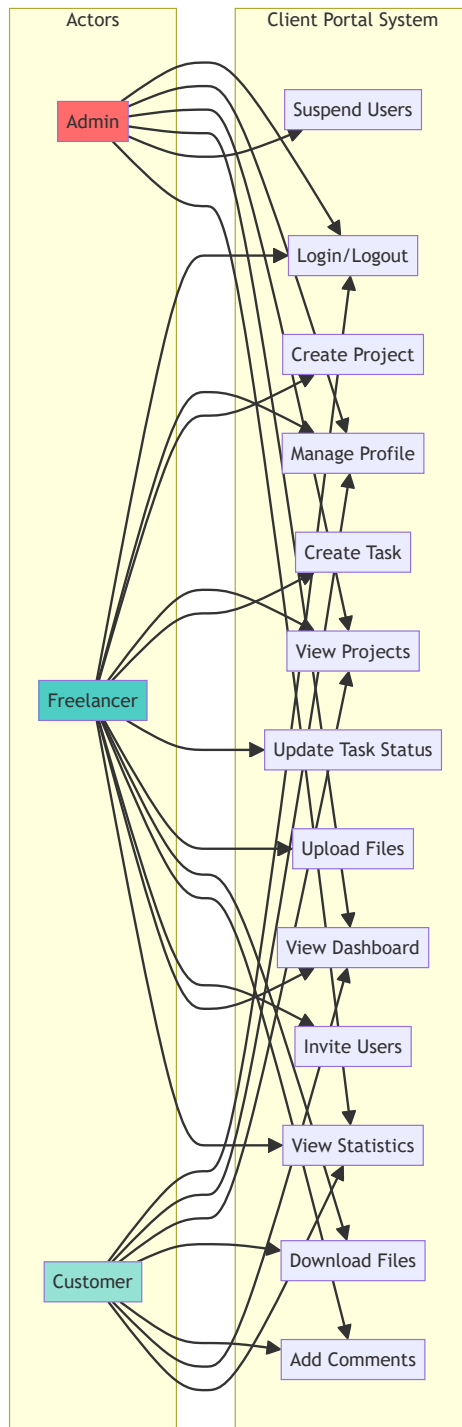
12.4 Admin User Suspension Flow



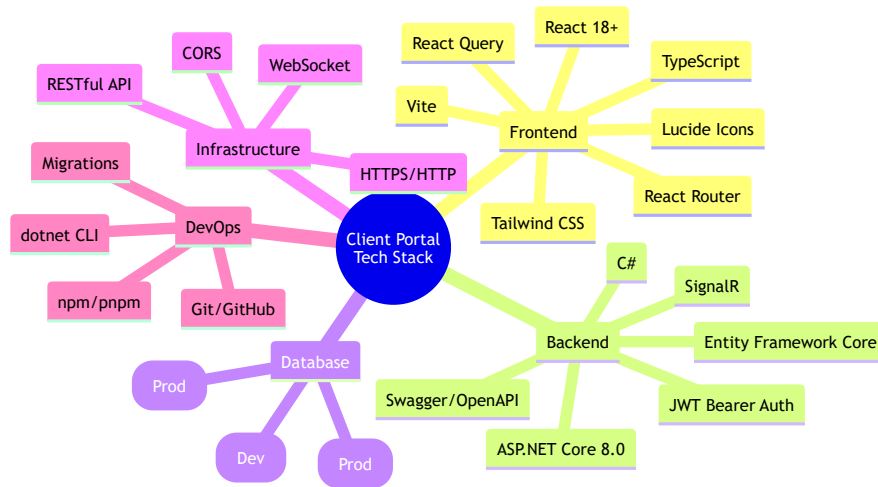
Class Diagram



Use Case Diagram



Technology Stack Diagram



Summary

These diagrams provide comprehensive visual documentation for:

- System Architecture** - High-level overview of all components
- Database Design** - Entity relationships and schema
- Authentication** - Security flow and token management
- User Roles** - Permission hierarchy
- API Flow** - Request/response lifecycle
- Component Structure** - Frontend architecture
- File Handling** - Upload/download processes
- Real-Time** - SignalR notification flow
- Workflows** - Project and task lifecycles
- Deployment** - Production architecture options
- Sequence Diagrams** - Detailed interaction flows
- Class Diagram** - Domain model structure
- Use Cases** - Role-based functionality

How to Use These Diagrams

- For Development:** Use sequence diagrams to understand feature implementation
- For Documentation:** Include in technical specifications and onboarding materials

3. **For Presentations:** Use architecture diagrams to explain system design
4. **For Database:** Reference ERD when writing queries or migrations
5. **For Testing:** Use state machines to create test cases

Rendering These Diagrams

These Mermaid diagrams can