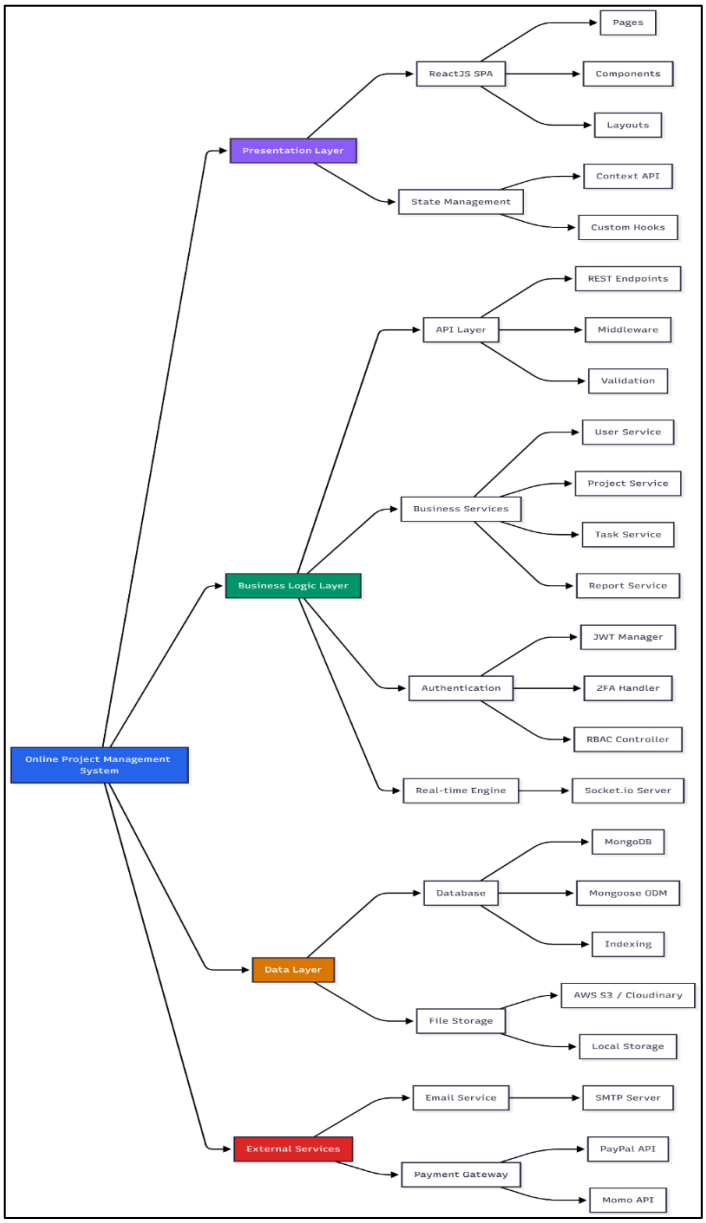
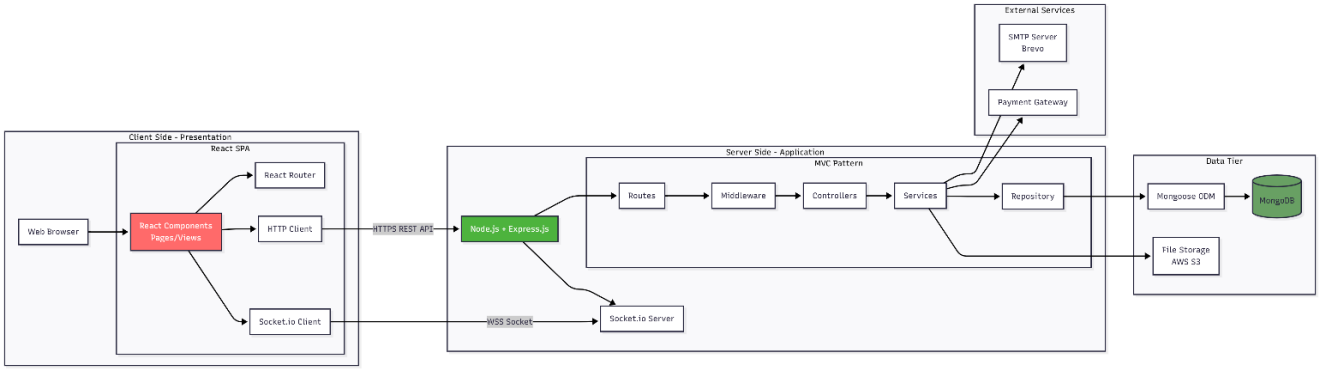
3 Architectural Design

3.1 Architecture Diagram



Picture: System decomposition tree diagram



Picture: MAIN components

3.1.1 Three-Tier Architecture

The system follows a strict three-tier architecture with clear separation of concerns:

- **Presentation Tier:** ReactJS Single Page Application (SPA) with Vite + TailwindCSS. Handles UI rendering, user interactions, and client-side state via React Context and custom hooks.
- **Business Logic Tier:** Node.js + Express.js server. Processes requests, enforces business rules, manages authentication/authorization, and orchestrates data operations.
- **Data Tier:** MongoDB database (Replica Set) with Mongoose ODM for schema validation and query abstraction. AWS S3/Cloudinary for file storage.

Communication occurs via RESTful HTTPS APIs between tiers, with WebSocket (Socket.io) for real-time features.

3.1.2 MVC Pattern

The Business Logic Tier implements Model-View-Controller (MVC):

- **Routes:** Map HTTP endpoints to controllers.
- **Controllers:** Handle requests, extract parameters, format responses.
- **Services:** Implement business logic and workflow orchestration.
- **Models:** Define MongoDB schemas using Mongoose.

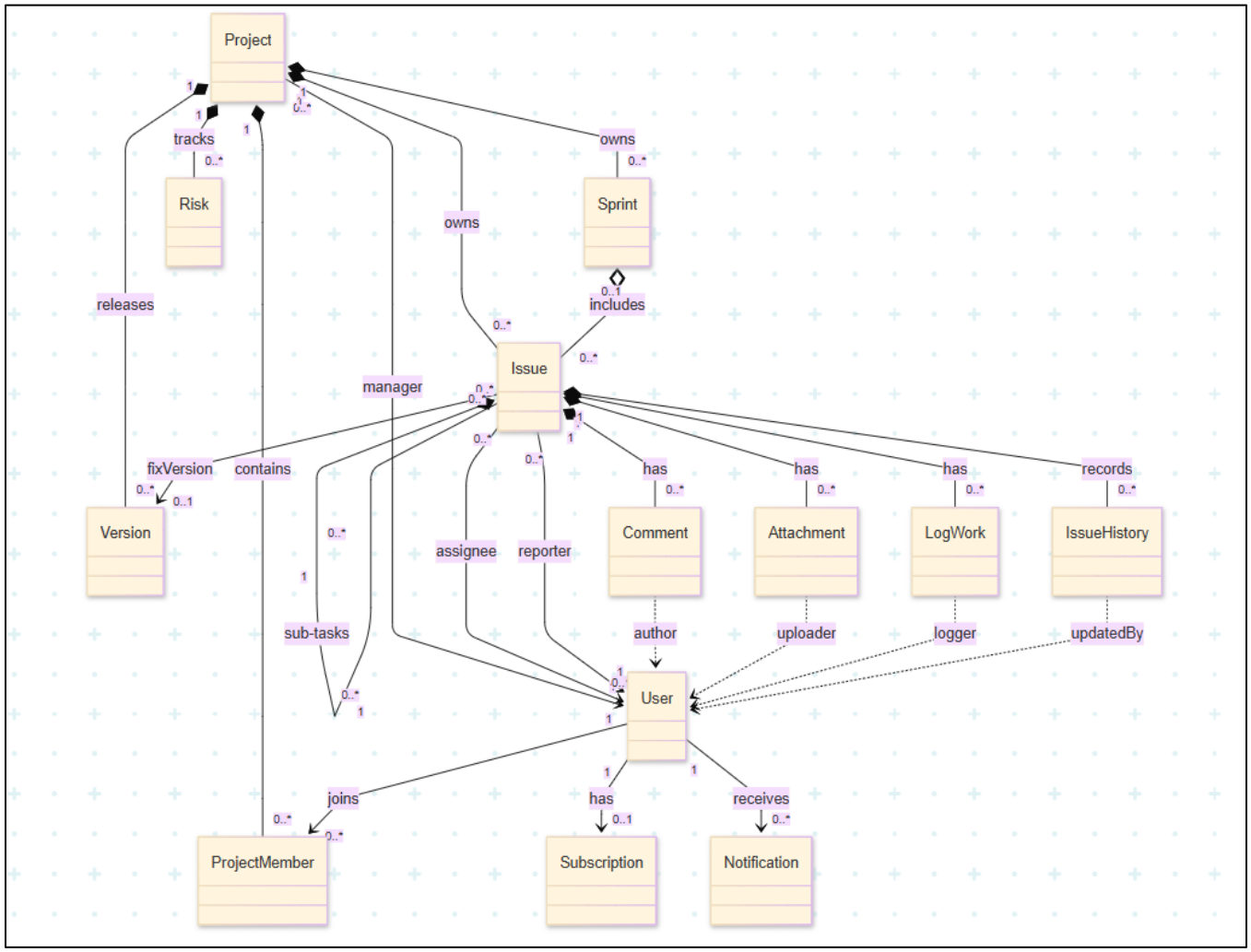
This separation ensures business logic is decoupled from HTTP concerns, making code testable and maintainable.

3.1.3 Repository Pattern

Purpose: Abstracts data access logic from business logic, providing a clean separation between the service layer and database operations.

- Business logic doesn't depend on MongoDB-specific syntax.
- Easier to write unit tests by mocking repositories.
- Database implementation can be changed without affecting services.
- Centralizes query logic (population, filtering).

3.2 Class Diagram



Picture: Class diagram