WellMind - Technical Design Document

Table of Contents

1. Introduction

WellMind is an innovative platform bridging the gap between patients and licensed therapists. It provides a seamless experience for mental health services with features like user authentication, therapist profile browsing, appointment scheduling, and secure communication. The platform emphasizes privacy, security, and accessibility.

2. System Overview

WellMind is a modern, web-based application built using the MERN stack (MongoDB, Express.js, React, Node.js) to provide scalability, maintainability, and a responsive user experience across devices.

3. Architecture

High-Level Architecture

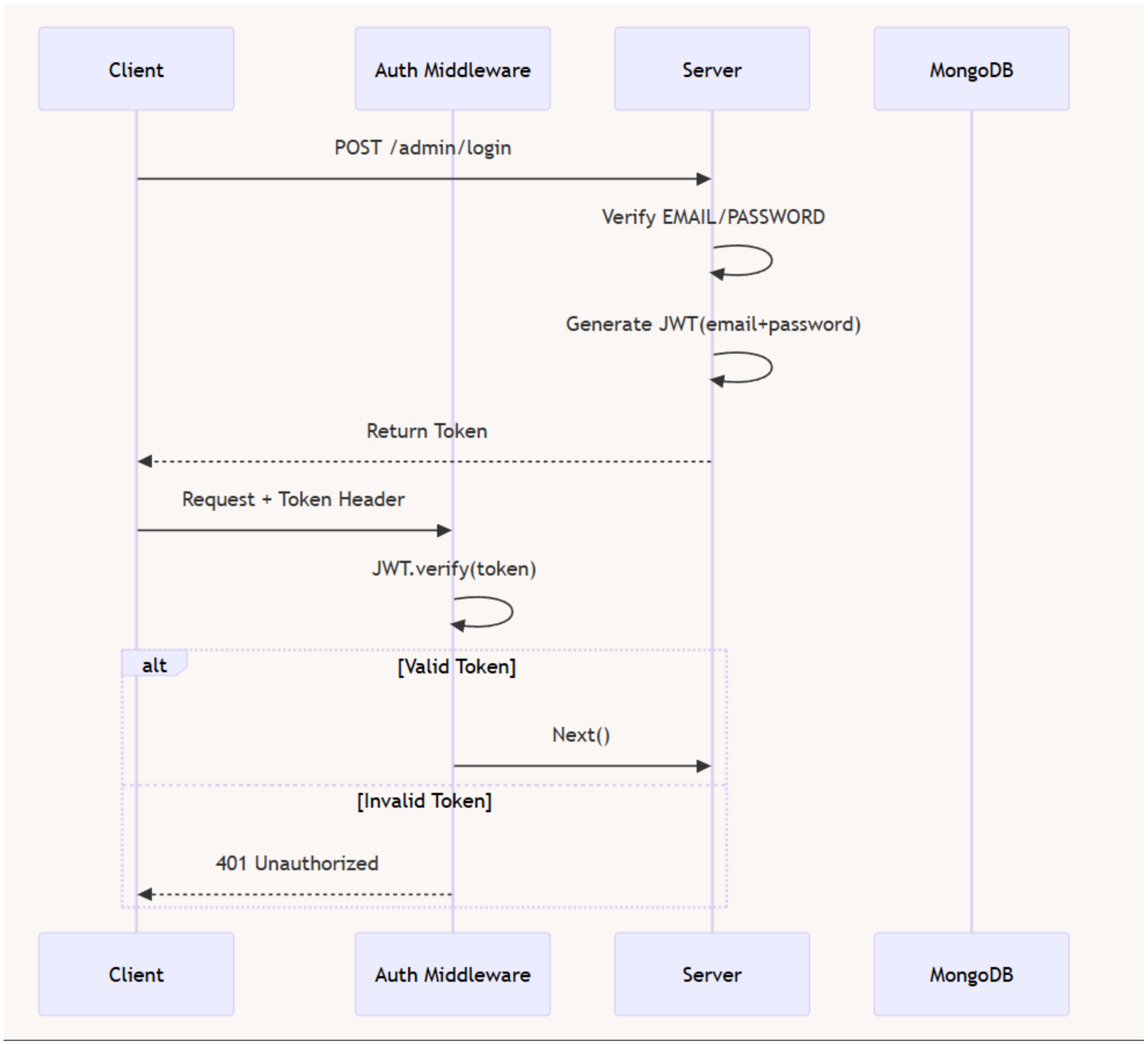
WellMind employs a three-tier architecture:

1. **Frontend**: Built using Vite-React, it is responsible for the user interface and interactions, integrating with Cloudinary to fetch media assets securely.
2. **Backend API**: Developed with Express.js, it handles server-side logic, authentication, and business rules. The backend directly manages media uploads and retrievals via Cloudinary APIs.
3. **Database**: MongoDB serves as the database for storing dynamic data such as user information, appointments, and therapist profiles, ensuring that the data structure aligns with media references stored in Cloudinary.

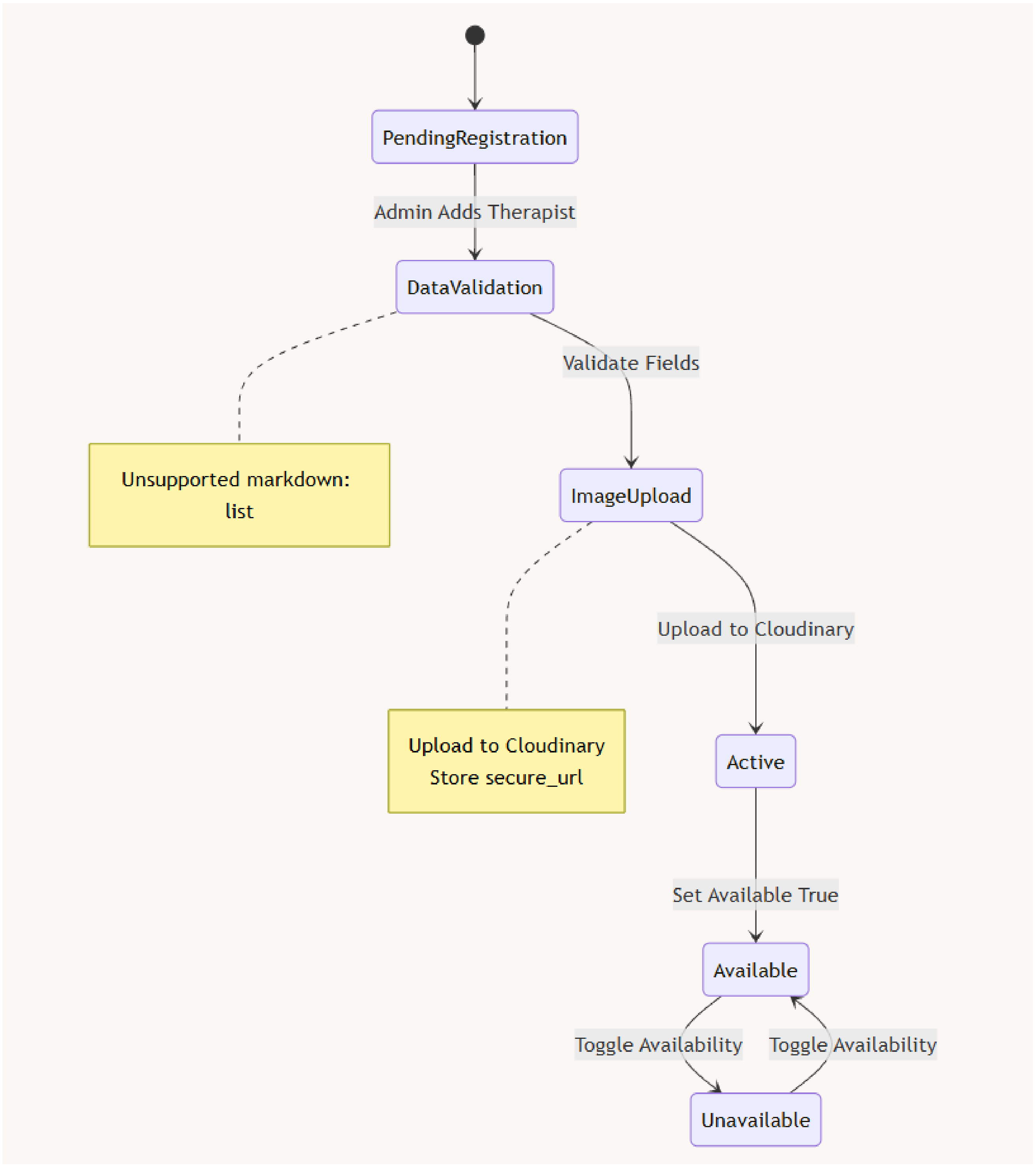
Component Interactions

- The frontend communicates with the backend API via RESTful endpoints.
- The backend interfaces with MongoDB to handle data storage and retrieval.
- Cloudinary is utilized for secure and efficient media storage (e.g., profile images).

Authentication Workflow



Therapist Management Flow



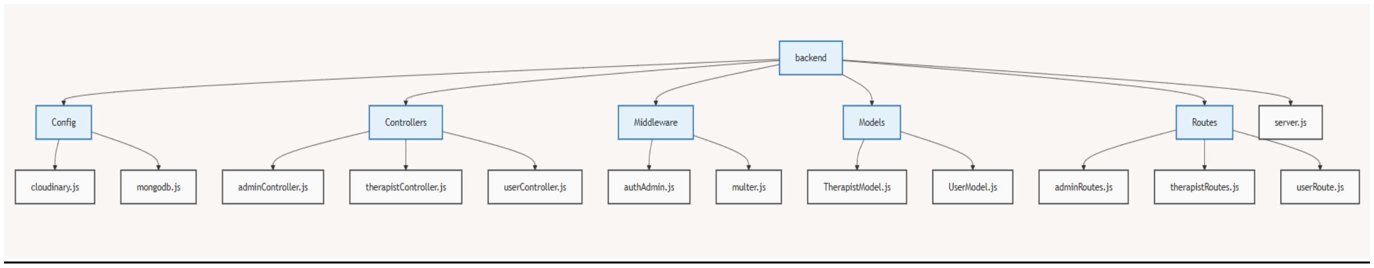
4. Backend Design

Technologies Used

- **Node.js**: JavaScript runtime environment.
- **Express.js**: Web application framework.
- **MongoDB**: NoSQL database.
- **Mongoose**: ODM library for MongoDB.
- **JWT**: For secure authentication.
- **bcrypt**: For password hashing.

Project Structure

/backend



API Design

User Endpoints

- **POST /auth/login**: Authenticates a user and issues a JWT.
- **POST /auth/register**: Registers a new user with role-based access.

Therapist Endpoints

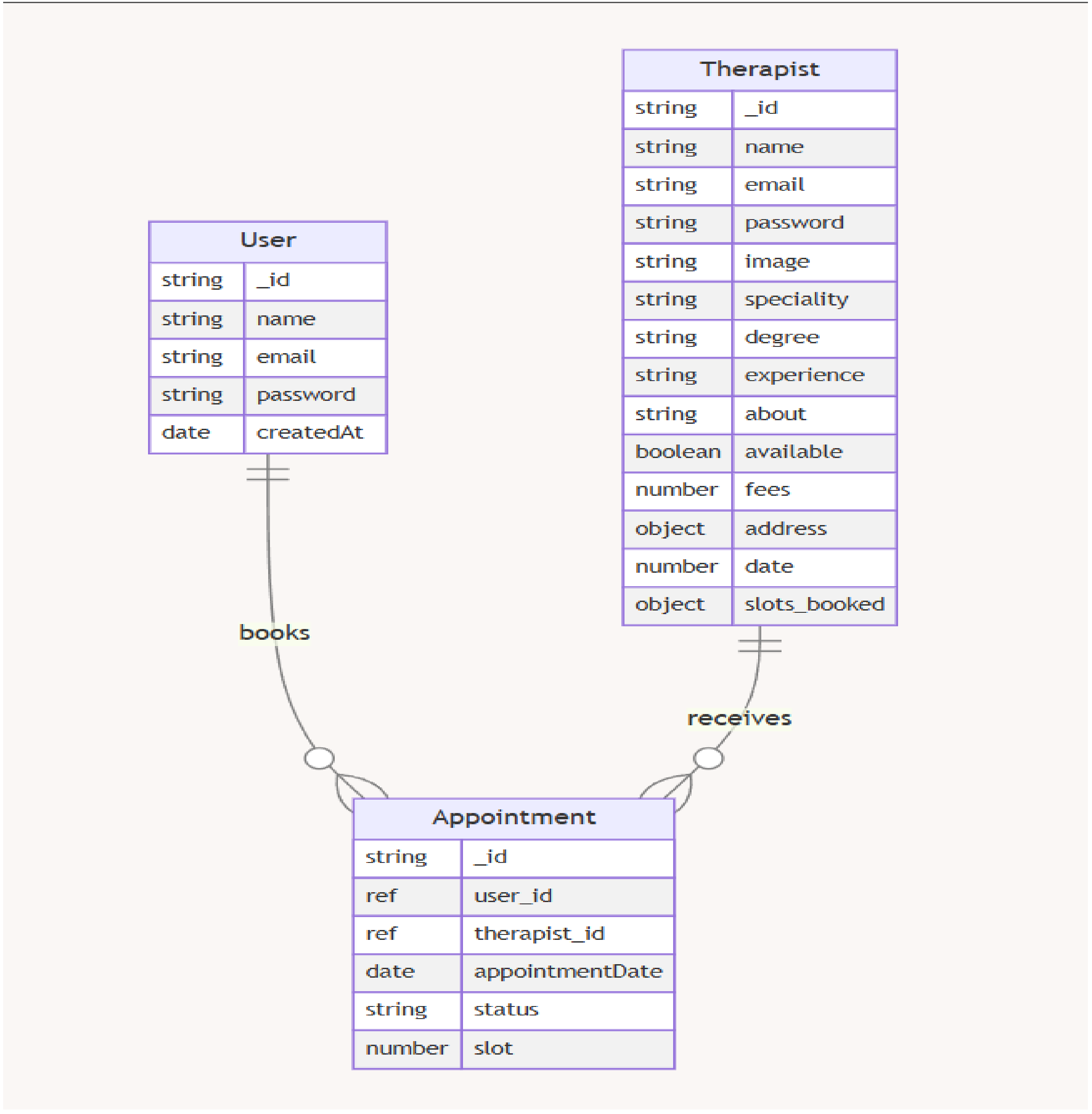
- **GET /therapists**: Fetches all available therapists with filters.
- **GET /therapists/:id**: Retrieves detailed information about a specific therapist.
- **POST /therapists**: Adds a new therapist (admin only).

Appointment Endpoints

- **POST /appointments**: Books a new appointment.
- **GET /appointments**: Retrieves user appointment history.

Database Schema

- **User Model**: Handles user information and credentials.
- **Therapist Model**: Stores therapist details, specialties, and availability.
- **Appointment Model**: Tracks booking details and statuses.



Middleware

- Authentication: Verifies JWT tokens.
- Error Handling: Centralized error responses for APIs.
- Input Validation: Sanitizes and validates request data.
- Logging and Monitoring: Captures request details and tracks performance metrics to enhance backend visibility.

---

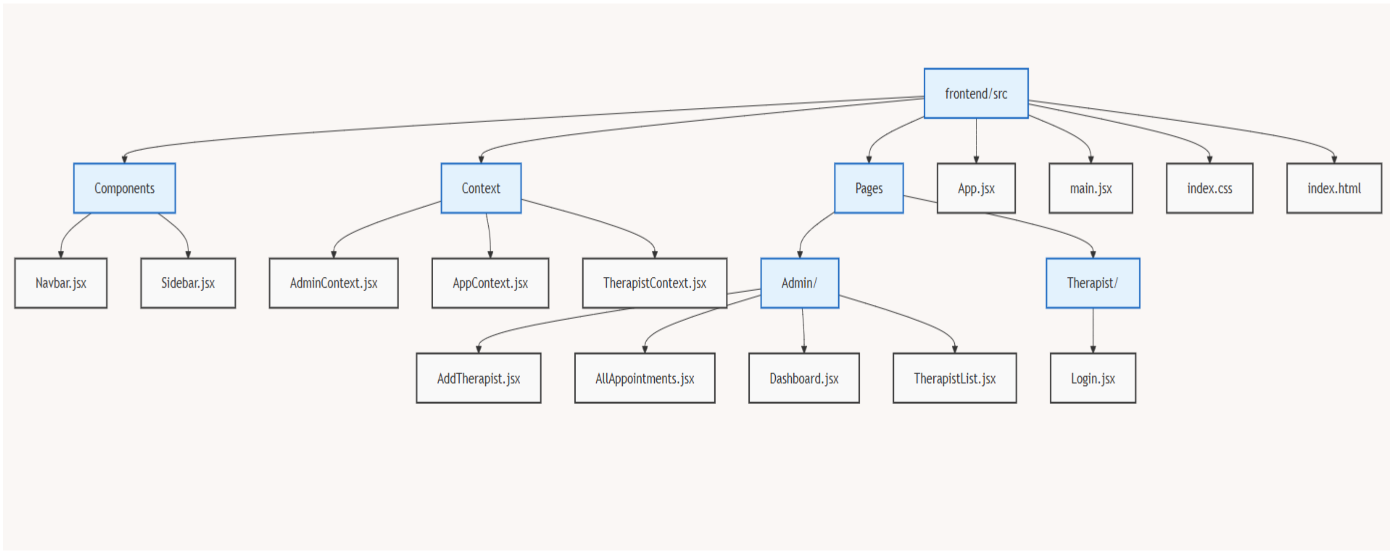
5. Frontend Design

Technologies Used

- **Vite-React**: Frontend framework.
- **Tailwind CSS**: For rapid and modern UI styling.
- **React Router DOM**: Client-side routing.
- **Axios**: For API requests.

Project Structure

/frontend



Routing

- **/**: Home Page
- **/login**: Login Page
- **/about**: About Us Page
- **/therapists**: Therapist Listing
- **/appointments**: User Appointment History

State Management

- **Local State**: Managed via React hooks.
- **Global State**: Context API for shared states like authentication and therapist data.

Key Components

- **Navbar**: Navigation menu with role-based access.
- **Sidebar**: Admin dashboard navigation.
- **TherapistList**: Displays therapist profiles with filters.
- **AppointmentForm**: Handles booking functionality.

---

6. Security Considerations

- **Authentication**: JWTs are securely signed and stored.
- **Password Security**: Uses bcrypt for hashing.
- **Authorization**: Protects routes and enforces role-based access.
- **CORS**: Restricts requests to trusted origins.
- **Data Encryption**: Sensitive data is encrypted during transmission.
- **Failed Login Management**: Implements account lockout after repeated failed attempts and sends alerts for suspicious activity to protect user accounts.

---

7. Deployment Plan

- **Frontend**: Deployed on AWS Amplify for scalable hosting and seamless integration with CI/CD pipelines.
- **Backend**: Hosted on AWS EC2 under the free tier, ensuring flexibility and control over server configurations.
- **Database**: MongoDB Atlas with restricted IP access.
- **Continuous Deployment**: CI/CD pipelines configured via AWS CodePipeline for automated deployments from GitHub.

---

8. Unit Testing

- **Frontend**: Jest and React Testing Library for component testing.
- **Backend**: Mocha and Chai for API testing.
- **Test Coverage**: Focus on critical features like authentication and booking.

---

9. Future Enhancements

1. **Integrated Payment System**: Add Stripe or PayPal for secure transactions.
2. **Telehealth Integration**: Enable video conferencing for virtual sessions.
3. **AI Recommendation**: Suggest therapists based on user preferences.
4. **Mobile App**: Develop Android and iOS applications for broader reach.
5. **Chat Feature**: Implement a real-time chat feature to facilitate seamless communication between therapists and patients.

---

10. Conclusion

WellMind provides a robust solution for mental health services, leveraging modern technologies and adhering to best practices in web development. Future iterations will enhance user experience, accessibility, and scalability.