Theramotion- Technical Design Document

Contents

1. Introduction

TheraMotion is a healthcare service platform that enables users to book healthcare services effortlessly. Built with modern web technologies, the platform offers features such as 'Join Our Team,' 'Meet Our Team,' 'Book an Appointment', ‘User Authentication’, and ‘Blog Engagement’.

Key Features:

- Secure Authentication System (JWT, protected routes, bcrypt, session management)
- Join our Team (mail to become part of team)
- Meet Team (searchable team member profiles with photos and bios)
- Book an Appointment (calendar integration, Razorpay, flexible cancellation)
- Blogs (informative and interactive healthcare topics)
- Modern UI/UX (fully responsive, user-friendly navigation)

The goal of this document is to offer a thorough guide to the project's architecture, components, technology stack, and design choices for developers, stakeholders, and contributors.

2. System Overview

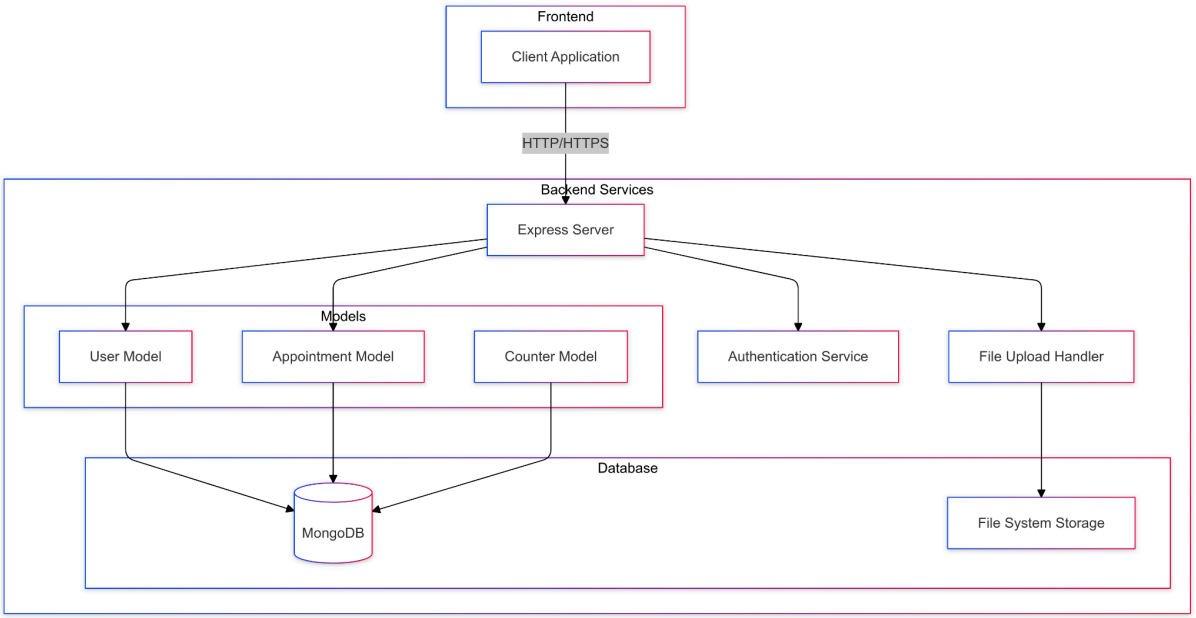
The MERN stack, which consists of MongoDB, Express.js, React, and Node.js, is used to build Theramotion. The application is designed to be scalable, maintainable, and responsive across multiple devices using current web development techniques.

The system comprises a backend API server using Express.js and MongoDB, and a user-facing frontend application.

3. Architecture

3.1. High-Level Architecture

The system architecture consists of three tiers: a frontend developed with React.js for user interface and interactions, a backend API built with Express.js and Node.js to manage server-side logic and API endpoints, and a MongoDB database for data storage.



4. Backend Design

4.1. Technologies Used

- **Node.js:** JavaScript runtime environment.
- **Express.js:** Web application framework for building APIs.
- **MongoDB:** NoSQL database for data storage.
- **Mongoose:** ODM (Object Data Modeling) library for MongoDB.

- **JWT:** JSON Web Tokens for authentication.
- **bcrypt:** Library for hashing passwords.
- **Razorpay:** Integration for secure online payments.

4.2. Project Structure

- **index.js:** Entry point of the server application.
- **config/:** Configuration files (e.g., database connection, constants,etc).
- **models/:** Mongoose schemas for User, Appointment,etc.
- **routes/:** Defines API endpoints for authentication, users, and appointment.
- **util/:** Utility functions and helpers (e.g., JWT verification, file upload configuration, and counter schema).

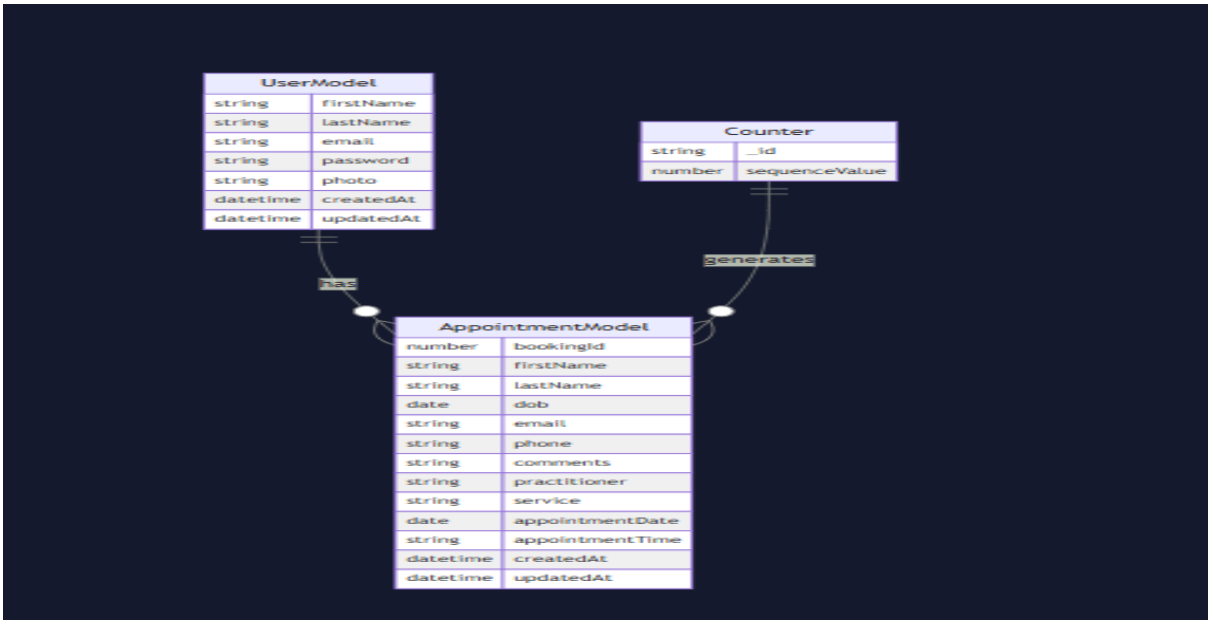
4.3. API Design

- Authentication (`/api/users`)
  - POST `/register`: User registration.
  - POST `/login`: User login and JWT token issuance.
  - GET `/currentUser`: Retrieve current user profile.
  - POST `/logout`: User logout.
  - GET `/:email`: Fetch user data by email (protected route, requires token).
  - PUT `/:email`: Update user information.
  - POST `/upload-photo`: Upload and update user profile photo (protected route).

- DELETE `/:email`: Delete a user account and related bookings after verifying password (protected route).

- Appointments (`/api/appointments`)
  - POST `/`: Create a new appointment.
  - GET `/:email/current`: Fetch all upcoming appointments for a user.
  - GET `/:email/previous`: Fetch all past appointments for a user.

4.4. Database Schema



- **User Model**

  Schema for managing user accounts and authentication:

  - **Fields:**
    - `firstName` (String, required): User's first name.
    - `lastName` (String): User's last name.
    - `email` (String, required, unique): User's email address, must be valid format.
    - `password` (String, required): Hashed password, minimum 6 characters.
    - `photo` (String): URL/path to user's photo, defaults to "/DefaultAvatar.png".
    - `timestamps`: Automatically managed creation/update timestamps.

  - **Methods:**
    - `getUser(req,successCallback, errorCallback)`: Retrieves user by email with token validation.
    - `signIn(user,successCallback,errorCallback)`: Authenticates user and generates JWT.
    - `addUser(user,successCallback,errorCallback)`: Creates new user with encrypted password.
    - deleteUserAndBookings(email, password): Deletes the user and their related bookings after verifying the password in a transaction.

- **Appointment Model**

  Schema for managing medical appointments.

  - **Fields:**
    - `bookingId` (Number, required, unique): Auto-generated booking identifier.
    - `firstName` (String, required): Patient's first name.
    - `lastName` (String): Patient's last name.
    - `dob` (Date, required): Patient's date of birth.
    - `email` (String, required): Patient's email.
    - `phone` (String, required): Patient's phone number.
    - `comments` (String): Additional appointment notes.
    - `appointmentDetails` (Object):
      - `practitioner` (String, required): Healthcare provider.
      - `service` (String, required): Type of service.
      - `date` (Date, required): Appointment date.
      - `time` (String, required): Appointment time.
    - `timestamps`: Automatically managed creation/update timestamps.
  - **Methods:**
    - `addAppointment(appointmentData,successCallback,errorCallback)`: Creates new appointment with auto-generated bookingId.

        - `getUserBookings(req,successCallback,errorCallback)`: Retrieves all appointments for a user's email.

- **Counter Model**

  Utility schema for generating sequential booking IDs.

  - **Fields:**
    - `_id` (String, required): Counter identifier.
    - `sequenceValue` (Number, required): Current sequence value.

5. Frontend Design

5.1. Technologies Used

- **React.js:** JavaScript library for building user interfaces.
- **React Router DOM:** Handling client-side routing.
- **Bootstrap:** Utility-first CSS framework for styling.
- **Font Awesome:** A library of scalable vector icons for enhancing web interfaces.
- **Vite:** Build tool for faster development.
- **ESLint:** A tool for maintaining code quality by identifying and fixing code issues.

5.2. Project Structure

- **index.html:** The main HTML file that loads the React application.
- **public/**: Contains global and static assets accessible by the public.
- **src/**: Source folder for the React application.
  - **components/**: Contains reusable React components.

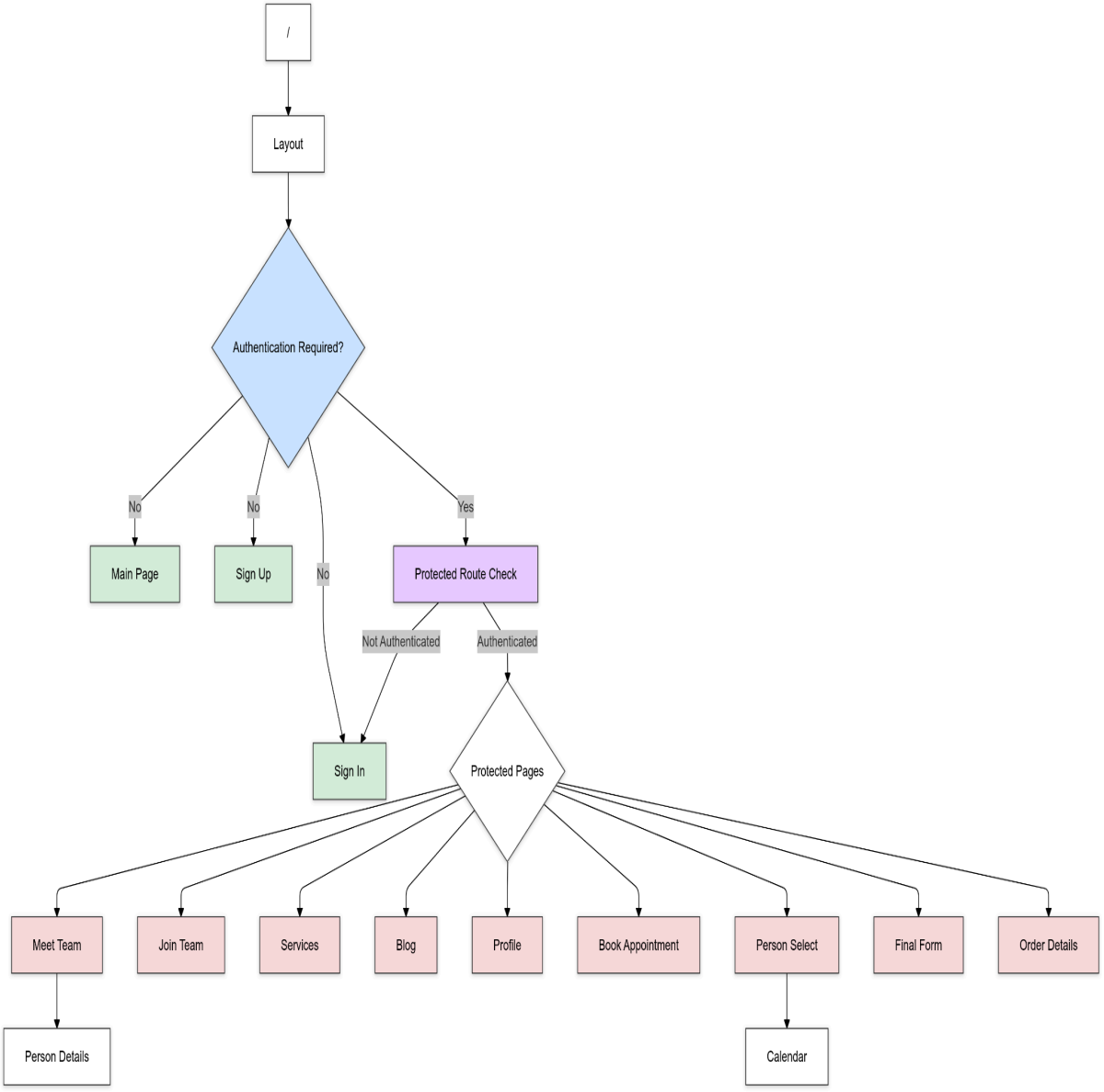
- **pages/**: Contains page-specific components that correspond to routes.
- **App.jsx**: The main application component that integrates all routes and layout.
- **Layout.jsx**: Layout component for a consistent user interface structure across pages.
- **main.jsx**: The entry point for rendering the React application.
- **routes.js**: Defines client-side routing and page rendering logic.

5.3. Routing

Used React Router for routing.

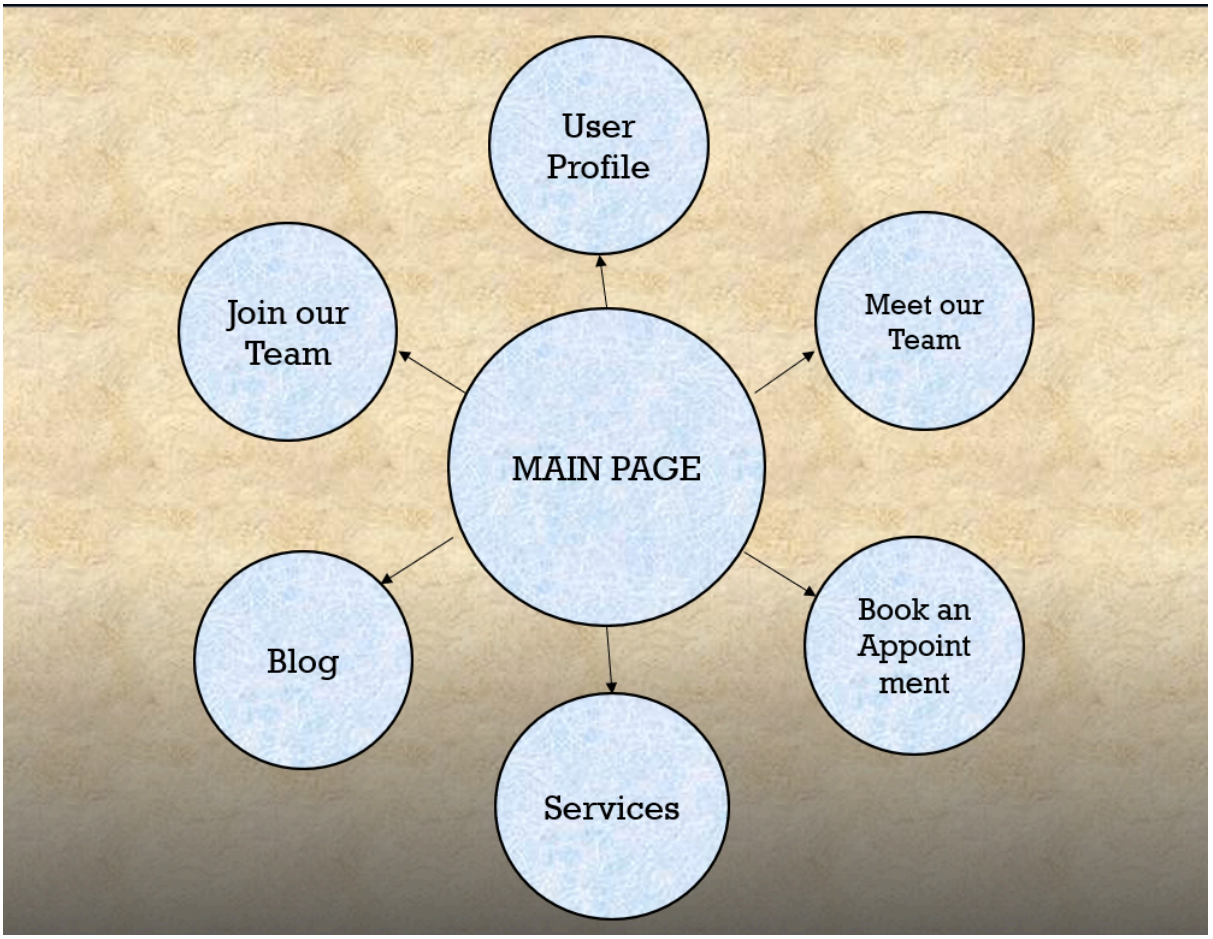
- `/`: Home page showcasing featured products.
- `/signup`: User registration page.
- `/signin`: User login page.
- `/meet-team`: Meet the team page with a list of team members.
- `/meet-team/:personId`: Detailed page for a specific team member.
- `/join-team`: Page for users to apply to join the team.
- `/services`: Services page outlining the available offerings.
- `/blog`: Blog page showcasing articles and updates.
- `/profile`: User profile page displaying user details and settings.
- `/book-appointment`: Appointment booking page for scheduling.
- `/person`: Page to select a person for an appointment or service.

- `/person/calendar`: Calendar page for managing appointments.
- `/form`: Final form page for submission or additional information.
- `/order`: Order details page showing the specifics of a booked appointment or service.



5.4. Pages

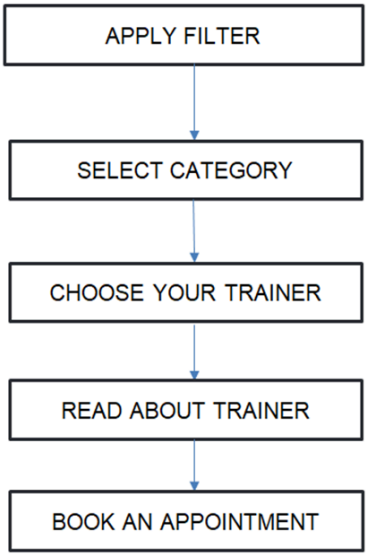
- **Schematic diagram**



- **Main Page:** The Main Page serves as the homepage of our service website, offering an overview of our offerings and guiding users to key areas of the site.
- **Meet Our Team Page:** The Meet Our Team page introduces visitors to the key members of our organization, highlighting their clinical interests, qualifications and locations. It aims to build trust and personalize the user experience.
- **Join Our Team Page:** The Join Our Team page invites talented candidates to explore career opportunities with our company.
- **Services Page:** The Services page provides detailed information about the various services we offer, helping

users understand our capabilities and make informed decisions.

- **Blog Page:** The Blog page serves as a central hub for sharing valuable content, updates, and insights related to our services. It aims to engage visitors, establish authority in our field, and enhance SEO efforts.
- **Book an Appointment Page:** This page is designed to offer a seamless user experience, ensuring that clients can quickly and efficiently book the services they need.



6. Security Considerations

6.1. Authentication:

- JWT tokens are used for secure implementation.
- Stored in user's localstorage.

6.2. Password Security:

- Passwords are encrypted using bcrypt.
- Passwords are not stored directly anywhere without encryption.

6.3. CORS Configuration:

- Requests are allowed from only trusted origins.
- Headers are set for allowed origin, headers, and methods.

7. Deployment Plan

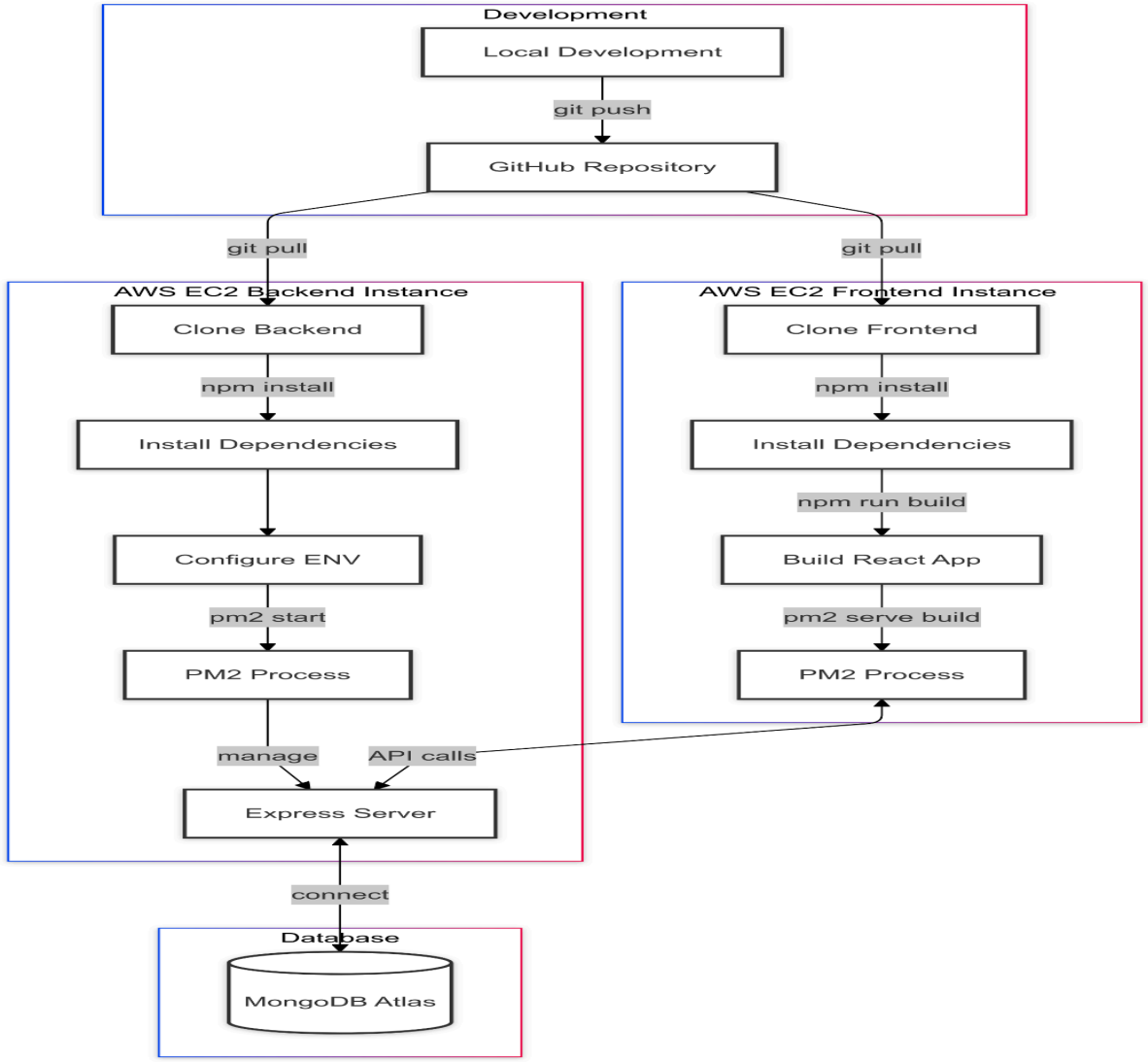
7.1. Environment Setup

```
# Backend (.env)
DB_CONNECTION_STRING=your_mongodb_connection_string
JWT_SECRET=your_jwt_secret_key
HashSalt=your_hashsalt
```

7.2. Deployment Steps

- **Backend Deployment:**
  - Host on platforms like AWS, Heroku, or DigitalOcean.
  - Use PM2 for process management.
- **Frontend Deployment:**
  - Build using npm run build.
  - Host on AWS, Vercel, or any other platform.
- **Database:**
  - Use managed MongoDB services like Atlas.
  - Enable IP whitelisting.
- **CI/CD Pipelines:**
  - Automate deployments using GitHub Actions or Jenkins.
- **Domain and SSL:**
  - Configure custom domains and SSL certificates.

7.3. Deployment Flowchart



8. Future Enhancements:

8.1. Feature Enhancements

- Enable users to publish their own blog posts.
- Allow users to cancel or update their bookings, and enable refund processing.

8.2. Technical Improvements

- Switch to TypeScript for better type safety.

- Implement Redux for more complex state management.
- Use Web Sockets for real time features like notifications and real time query solving.

9. Conclusion

Theramotion addresses gaps in healthcare access and service delivery, focusing on therapy and healing. The platform provides accurate information, promotes health literacy, and offers easy access to specialized therapeutic services, empowering individuals to manage their recovery and well-being. With expert therapists and modern healing techniques, it serves as a comprehensive solution for managing health conditions.

The project emphasizes affordability, accessibility, and quality, allowing users to connect with experienced specialists, book appointments, and access resources for healing. By promoting preventative care and holistic wellness, it helps users achieve long-term health and well-being. Theramotion aims to bridge healthcare disparities and empower individuals to live healthier, pain-free lives.