**1. INTRODUCTION**

**1.1 Background**

The healthcare industry is undergoing a significant transformation, driven by increasing patient expectations, advancements in technology, and the need for more accessible and personalized care. Traditional healthcare models often face challenges in providing timely and convenient access to medical expertise, especially in remote areas or for individuals with mobility constraints. This project addresses these challenges by developing a comprehensive online health consultant platform that connects patients with qualified healthcare professionals virtually.

Health Consultant emerged as a solution to bridge the gap between patients and medical expertise, enabling remote consultations, health monitoring, and personalized advice. With the rise of telehealth and virtual care, this platform aimed to provide a seamless, secure, and user-friendly experience for both patients and consultants. The initial concept centered on offering a single point of access for a diverse range of healthcare services, leveraging technology to improve healthcare outcomes and patient satisfaction.

This project report documents the complete development process of the Health Consultant platform, from the initial concept to the final deployment. It outlines the project's objectives, scope, and applicability, as well as the technical challenges overcome, the architectural decisions made, and the testing and validation procedures implemented. The ultimate goal of this document is to serve as a comprehensive resource for understanding the project's intricacies, the technologies used, and the potential for future development and expansion.

**1.2 Objective**

The primary objective of the Health Consultant project was to create a functional, scalable, and secure online platform that enables remote healthcare consultations between patients and licensed healthcare professionals. To be precise, we wanted to achieve:

* + **Establish a Robust Online Platform:** To create a user-friendly website that caters to both patients seeking medical advice and consultants providing their expertise.
  + **Seamless Communication:** Enable patients to connect with doctors for the best online video consultation experience.
  + **Secure Data Handling:** Ensuring all health records and communications are encrypted and stored securely.
  + **Scalable Architecture:** Build the platform to accommodate a growing user base and expanding service offerings.
  + **User-Centric Design:** Design the platform's interface with a focus on ease of use, accessibility, and a positive user experience.
  + **Integration of Key Features:** Incorporate functionalities like consultant profiles, booking system, payment processing, messaging and chat, and health record management.
  + **Ensure Regulatory Compliance:** To adhere to all relevant data protection and healthcare regulations, including GDPR and HIPAA (or equivalent regional laws).

**1.3 Scope and Applicability**

**1.3.1 Purpose**

The purpose of this document is to provide a detailed overview of the Health Consultant project, including the system's design, implementation, and testing. It serves as a comprehensive reference for stakeholders involved in the project, such as developers, project managers, investors, and future maintainers. More specifically, the aim of this document is to:

* + Describe the requirements that initiated the project and how they were met through the developed solution.
  + Offer technical details of the architecture, database schema, and frontend implementation.
  + Outline security measures to safeguard patient data and maintain system integrity.
  + Serve as a guide for ongoing system maintenance, upgrades, and future enhancements.
  + Document the various phases of the software development lifecycle (SDLC), including planning, analysis, design, development, testing, and deployment.
  + Provide a valuable source of information for other developers and organizations interested in building similar telehealth solutions.

**1.3.2 Scope**

The scope of the Health Consultant project encompasses the following key areas:

* + **User Management:** Registration, login, profile management, and role-based access control (patients, consultants, administrators).
  + **Consultant Profiles:** Creation and management of consultant profiles, including specialties, qualifications, availability, and pricing.
  + **Booking System:** Scheduling and management of appointments between patients and consultants, including time slot selection and payment processing.
  + **Payment Gateway Integration:** Integration with secure payment gateways for online payment processing.
  + **Messaging and Chat:** Real-time chat and messaging functionality for communication between patients and consultants.
  + **Health Record Management:** Secure storage and retrieval of patient health records, including medical history, prescriptions, and other relevant information.
  + **Administrative Dashboard:** A central dashboard for administrators to manage users, consultants, bookings, and system settings.
  + **Consultant approval Module:** A module for admin, to approve and decline newly registered consultant
  + **Technology Stack:** Use of specific technologies including React, Node.js, Express, MySQL, and relevant libraries and frameworks.
  + **Security Implementation:** Application of security best practices to protect sensitive data and prevent unauthorized access.

The project scope does *not* include:

* + Integration with external Electronic Health Record (EHR) systems.
  + Development of mobile applications (although the platform is designed to be responsive and accessible on mobile devices).
  + Full regulatory compliance certification (although the platform is designed with regulatory considerations in mind).
  + AI-powered diagnostics or treatment recommendations.

**1.3.3 Applicability**

The Health Consultant platform has broad applicability in various healthcare settings and scenarios, including:

* + **Remote Healthcare:** Providing healthcare services to patients in remote areas or those with limited access to traditional medical facilities.
  + **Specialty Care:** Connecting patients with specialists for consultations and second opinions.
  + **Chronic Disease Management:** Enabling remote monitoring and management of chronic conditions.
  + **Mental Health Support:** Offering virtual therapy and counseling services.
  + **Corporate Wellness Programs:** Providing employees with convenient access to healthcare consultations and advice.
  + **International Healthcare:** Facilitating cross-border healthcare consultations and access to expertise.

The platform is particularly applicable to healthcare providers seeking to expand their reach, improve patient access, and enhance the efficiency of their services. It also provides a valuable tool for patients seeking convenient, affordable, and personalized healthcare advice.

**2. SURVEY OF TECHNOLOGY**

**2.1 Frontend Technologies - Frameworks and Languages**

The frontend of the Health Consultant platform was built with a focus on providing a responsive, interactive, and user-friendly experience. The following technologies were selected to achieve these goals:

* + **React:** React, a JavaScript library for building user interfaces, served as the primary framework for the frontend development. React's component-based architecture allowed for efficient code reuse, modularity, and maintainability. Its virtual DOM implementation enabled optimized rendering and improved performance.
  + **JavaScript:** JavaScript was used as the core programming language for implementing the frontend logic, handling user interactions, and managing data flow. ES6+ features were leveraged for improved code syntax and functionality.
  + **JSX (JavaScript XML):** JSX, a syntax extension to JavaScript, was used to describe the structure of the UI components in a declarative manner. JSX allowed developers to write HTML-like syntax within JavaScript code, making it easier to visualize and manipulate the UI components.
  + **CSS (Cascading Style Sheets):** CSS was employed for styling the frontend components and defining the visual appearance of the platform. A combination of CSS modules and TailwindCSS (see below) was used for managing styles and ensuring consistency across the application.
  + **Tailwind CSS:** Tailwind CSS, a utility-first CSS framework, was used to accelerate the styling process and maintain a consistent design language across the application. Tailwind CSS provided a comprehensive set of pre-defined CSS classes that could be easily combined to style elements without writing custom CSS rules.
  + **Lucide React:** A react library for icons to provide the User Interface with some cool icons.
  + **Material UI (MUI):** While Tailwind CSS was the primary styling solution, Material UI components were used strategically for specific UI elements, such as dialogs, selects, and textfields. MUI provided a set of pre-designed, accessible components that could be easily integrated into the React application. It is used for developing and rendering the data in a manageable structured form.
  + **Vite:** Vite was used as a modern and fast build tool for the frontend project. Vite's rapid development server and optimized build process significantly improved the development workflow and reduced build times.
  + **Dayjs:** Dayjs was used for parsing, validating, manipulating, and displaying dates and times in a user-friendly format.
  + **ESLint:** ESlint was used to help improve the maintainability of the frontend codes along with other cool coding practices.

**2.2 Backend Technologies - Framework, Languages, API and Database**

* **Content:**

The backend of the Health Consultant platform was designed to provide a robust, scalable, and secure foundation for managing data, handling API requests, and implementing business logic. The following technologies were selected for the backend development:

* + **Node.js:** Node.js, a JavaScript runtime environment, was used as the primary platform for building the backend server. Node.js's non-blocking, event-driven architecture enabled efficient handling of concurrent requests and improved performance.
  + **Express.js:** Express.js, a minimal and flexible Node.js web application framework, was used to structure the backend application and handle API requests. Express.js provided a set of powerful features for routing, middleware, and request/response handling.
  + **MySQL:** MySQL, an open-source relational database management system (RDBMS), was chosen for storing and managing the platform's data, including user information, consultant profiles, bookings, health records, and payment details. MySQL's reliability, scalability, and support for ACID properties made it a suitable choice for the project's data storage needs.
  + **Sequelize ORM:** Used to establish a more secure connection with the database. It also uses models for easier manipulation of the different tables.
  + **JSON Web Tokens (JWT):** JWTs were implemented for secure user authentication and authorization. JWTs allowed the server to verify the identity of users and grant access to protected resources based on their roles and permissions.
  + **bcrypt:** Used for hashing the passwords to improve security. With this no one would be able to read the Passwords from the Database.
  + **dotenv:** Used to save the database password and other credential information for improved security.
  + **CORS (Cross-Origin Resource Sharing):** CORS middleware was used to enable secure cross-origin requests from the frontend application to the backend API.
  + **morgan and helmet:** Middleware are implemented to increase the security of the website.
  + **Multer:** Multer is a node.js middleware for handling multipart/form-data, which is primarily used for uploading files. Multer was used to upload the consultant's certificate to the website when registering.
  + **API Endpoints:** The backend exposed a RESTful API for the frontend application to interact with the server. Key API endpoints included:
    - /api/register: User registration endpoint.
    - /api/login: User login endpoint.
    - /api/profile: User profile management endpoint.
    - /api/consultants: Consultant search and retrieval endpoint.
    - /api/bookings: Booking creation and management endpoint.
    - /api/healthrecords: Health record management endpoint.
    - /api/messages: Messaging endpoint.
    - /api/payments: Payment processing endpoint.
    - /api/admin: Administrative endpoints for user and system management.
    - /api/chat: Endpoints used for creating and managing chats.
  + **express-validator:** The express-validator library provides a set of express.js middlewares that can be used to validate data
    - Sanitization: removing potentially malicious parts of an input
    - Validation: ensuring that the values match your expected format
  + **NodeMailer:** Nodemailer is a module for Node.js applications that allows you to send emails easily from the server. The transport object is a way of configuring Nodemailer to use an email provider (like Gmail, SendGrid, or a custom SMTP server) to send messages
  + **Express-rate-limit:** the rate limit module helps limit the amount of request a user can make to prevent DDOS attack  
     Rate limiting is a technique used to control the amount of traffic or requests that are allowed to pass through a system over a certain period
  + **helmet:** Helment helps secure express apps by setting various HTTP headers
    - HTTP headers are like labels that provide additional information about the web request or response

**3. REQUIREMENT AND ANALYSIS**

**3.1 Problem Definition**

The Health Consultant platform emerges from a confluence of interconnected challenges within the current healthcare landscape. A comprehensive understanding of these issues is crucial for justifying the platform's development and guiding its design and functionality. This section details the specific problems that Health Consultant seeks to address:

**1. Limited Access to Healthcare Services:**

* + **Geographic Barriers:** A significant portion of the population, particularly in rural and underserved areas, faces challenges in accessing quality healthcare services due to geographic limitations. Traveling long distances to see specialists or receive routine care can be time-consuming, costly, and logistically difficult. This problem disproportionately affects elderly individuals, those with disabilities, and those with limited transportation options.
  + **Socioeconomic Disparities:** Access to healthcare is also influenced by socioeconomic factors. Individuals with lower incomes may struggle to afford healthcare services, including consultations, diagnostic tests, and medications. Lack of insurance coverage or inadequate coverage further exacerbates these disparities, leading to delayed or forgone care.
  + **Limited Mobility:** Patients with chronic illnesses, disabilities, or mobility impairments may find it challenging to travel to traditional healthcare facilities. This can significantly limit their access to necessary medical care and negatively impact their health outcomes.
  + **Lack of Time:** Even for individuals with adequate access to healthcare facilities, time constraints can be a significant barrier. Busy work schedules, family responsibilities, and other commitments can make it difficult to schedule and attend in-person appointments.

**2. Inefficient Healthcare Delivery Models:**

* + **Long Wait Times:** Traditional healthcare systems are often characterized by long wait times for appointments, especially with specialists. This can lead to delays in diagnosis and treatment, potentially worsening health conditions and increasing healthcare costs.
  + **Fragmented Care:** Patients often navigate a complex and fragmented healthcare system, with limited communication and coordination between different providers. This can result in duplicated tests, conflicting treatment plans, and a lack of continuity of care.
  + **Information Asymmetry:** Patients may lack sufficient information about their health conditions, treatment options, and available resources. This information asymmetry can make it difficult for them to make informed decisions about their care.
  + **Administrative Burden:** Healthcare providers face a significant administrative burden, including paperwork, billing, and compliance requirements. This can divert resources away from direct patient care and contribute to burnout.

**3. Rising Healthcare Costs:**

* + **Consultation Fees:** The cost of in-person consultations with specialists can be prohibitively expensive for many individuals, especially those without adequate insurance coverage. This can deter patients from seeking necessary medical advice and contribute to the underutilization of healthcare services.
  + **Travel Expenses:** Traveling to healthcare facilities can incur significant expenses, including transportation costs, parking fees, and time off from work. These expenses can be particularly burdensome for individuals in rural areas or those with limited financial resources.
  + **Hospital Readmissions:** Inadequate follow-up care and patient education can lead to hospital readmissions, which are a major driver of healthcare costs. Remote monitoring and virtual consultations can help reduce readmission rates by providing patients with timely support and guidance.

**4. Lack of Personalized Healthcare Experiences:**

* + **One-Size-Fits-All Approach:** Traditional healthcare models often adopt a one-size-fits-all approach, failing to address the unique needs and preferences of individual patients. This can lead to dissatisfaction, poor adherence to treatment plans, and suboptimal health outcomes.
  + **Limited Patient Engagement:** Patients may feel disempowered and uninvolved in their own healthcare decisions. A lack of patient engagement can reduce adherence to treatment plans and hinder efforts to improve health outcomes.
  + **Insufficient Health Education:** Patients may not receive adequate education about their health conditions, treatment options, and preventive measures. This can lead to poor self-management skills and increased reliance on healthcare providers.

**5. Growing Demand for Telehealth Solutions:**

* + **Changing Patient Preferences:** Patients are increasingly seeking convenient, accessible, and affordable healthcare options that fit their lifestyles. Telehealth solutions offer a viable alternative to traditional in-person visits, providing greater flexibility and convenience.
  + **Technological Advancements:** Advancements in technology, such as high-speed internet, mobile devices, and video conferencing platforms, have made telehealth solutions more feasible and accessible than ever before.
  + **Increased Adoption by Healthcare Providers:** Healthcare providers are recognizing the potential of telehealth to improve patient access, enhance efficiency, and reduce costs. This has led to a growing adoption of telehealth solutions across various specialties and settings.

By addressing these challenges, the Health Consultant platform aims to provide a more equitable, efficient, and patient-centered healthcare experience. Its potential impact includes improved access to care, reduced healthcare costs, enhanced patient engagement, and better health outcomes for individuals and communities.

**3.2 Requirement Specification**

This section specifies the detailed functional and non-functional requirements for the Health Consultant platform. These requirements are categorized based on user roles (patients, consultants, administrators) and system functionalities.

**I. Functional Requirements:**

**A. User Management:**

* + **FR1: User Registration:** The system shall allow new users to register by providing their name, email address, phone number, password, and role (patient or consultant). Registration process should validate the correct user information and return an error if something goes wrong
  + **FR2: User Login:** The system shall allow registered users to log in using their email address and password. The system shall authenticate users based on their credentials and redirect them to the appropriate dashboard based on their role.
  + **FR3: User Profile Management:** The system shall allow users to create and manage their profiles, including updating their personal information, contact details, profile picture, and other relevant data. Consultant profiles should also include their qualifications, specialities, expertise, consultation fees, and availability. The system should allow users to delete and update data as per the need.
  + **FR4: Password Management:** The system shall allow users to reset their passwords through a secure password reset process. The system shall send a password reset link to the user's registered email address, which they can use to create a new password.
  + **FR5: Role-Based Access Control:** The system shall implement role-based access control to restrict access to certain features and functionalities based on user roles. Patients, consultants, and administrators should have different levels of access and permissions.

**B. Consultant Management:**

* + **FR6: Consultant Search:** The system shall allow patients to search for consultants based on various criteria, such as specialty, location, availability, and rating. The system should provide search filters and sorting options to refine search results.
  + **FR7: Consultant Profile View:** The system shall allow patients to view consultant profiles, including their qualifications, specialities, expertise, consultation fees, availability, and ratings/reviews. The system should also display the consultant's contact information and a brief bio.
  + **FR8: Consultant Approval:** The system shall provide administrators with the ability to approve or reject consultant registrations. Only approved consultants should be able to offer their services on the platform. Administrator should have the access to add review and add consultant data.
  + **FR9: Consultant Document Retrieval:** Authorized users should be able to fetch the document a consultant added when registering in the website

**C. Booking System:**

* + **FR10: Appointment Scheduling:** The system shall allow patients to schedule appointments with consultants based on their availability. The system should provide a calendar interface for selecting available dates and times.
  + **FR11: Booking Confirmation:** The system shall send confirmation emails to both patients and consultants upon successful booking of an appointment. The confirmation email should include the date, time, consultant details, and a link to manage the appointment.
  + **FR12: Booking Management:** The system shall allow patients and consultants to manage their bookings, including viewing, rescheduling, and canceling appointments. Cancellation policies should be clearly defined and enforced.
  + **FR13: Availability Management:** The system shall allow consultants to manage their availability by specifying their working hours, days off, and vacation periods. The system should prevent patients from scheduling appointments during unavailable times.
  + **FR14: Consultant Booking Status Update:** When a user books a doctor, the consultant will receive a status, which they can accept/reject.

**D. Payment Processing:**

* + **FR15: Payment Integration:** The system shall integrate with a secure payment gateway to process online payments for consultations. The payment gateway should support various payment methods, such as credit cards, debit cards, and PayPal.
  + **FR16: Payment Security:** The system shall ensure the security of payment transactions by encrypting sensitive data and complying with PCI DSS standards. The system should not store credit card information on its servers.
  + **FR17: Refund Processing:** The system shall allow administrators to process refunds for canceled appointments. Refund policies should be clearly defined and enforced.
  + **FR18: Consultant Earnings Tracking:** The system shall enable consultants to track their earnings and view their payment history. The system should also generate reports on consultant earnings for administrative purposes.

**E. Messaging and Chat:**

* + **FR19: Real-Time Chat:** The system shall provide real-time chat functionality for communication between patients and consultants during consultations. The chat interface should support text messaging, file sharing, and screen sharing.
  + **FR20: Message History:** The system shall store a history of chat messages for future reference. Users should be able to access their chat history and review past conversations.
  + **FR21: Notifications:** The system shall send notifications to users when they receive new messages or have upcoming appointments. Notifications should be delivered via email and/or in-app notifications.

**F. Health Record Management:**

* + **FR22: Secure Storage:** The system shall store patient health records securely and confidentially. Access to health records should be restricted to authorized personnel only.
  + **FR23: Record Upload:** The system shall allow patients to upload their medical history, prescriptions, and other relevant health information. The system should support various file formats, such as PDF, DOCX, and JPG.
  + **FR24: Record Retrieval:** The system shall allow patients and consultants to retrieve and view patient health records. Consultants should only have access to health records for patients who have scheduled appointments with them.

**G. Administrative Dashboard:**

* + **FR25: User Management:** The system shall provide administrators with the ability to manage user accounts, including creating, editing, and deleting accounts. Administrators should also be able to reset user passwords and manage user roles.
  + **FR26: Consultant Management:** The system shall provide administrators with the ability to manage consultant profiles, including approving or rejecting registrations, updating profile information, and suspending accounts.
  + **FR27: Booking Management:** The system shall provide administrators with the ability to view and manage all bookings, including viewing booking details, rescheduling appointments, and processing cancellations and refunds.
  + **FR28: System Configuration:** The system shall allow administrators to configure system settings, such as payment gateway settings, email settings, and notification settings.
  + **FR29: Payment Tracking:** Provide admins and other designated personnels with a payments tracking module to keep track of the cash inflow/outflow.

**II. Non-Functional Requirements:**

* + **Performance:** The system shall be responsive and provide a fast user experience. Page load times should be less than 3 seconds. The system should be able to handle a large number of concurrent users without performance degradation.
  + **Security:** The system shall protect sensitive data and prevent unauthorized access. All data should be encrypted in transit and at rest. The system should comply with relevant data privacy regulations, such as GDPR and HIPAA.
  + **Scalability:** The system shall be scalable to accommodate a growing user base and expanding service offerings. The system should be designed to handle increasing traffic and data volumes without requiring significant code changes.
  + **Usability:** The system shall be user-friendly and easy to navigate. The user interface should be intuitive and accessible to users with varying levels of technical expertise.
  + **Reliability:** The system shall be reliable and available 24/7. The system should be designed to minimize downtime and ensure data integrity.
  + **Maintainability:** The system shall be designed for easy maintenance and updates. The codebase should be well-structured, documented, and modular.
  + **Testability:** The system shall be designed to be testable and allow for automated testing. Unit tests, integration tests, and end-to-end tests should be implemented to ensure the quality of the code.
  + **Accessibility:** The system should be accessible to users with disabilities, complying with WCAG guidelines. The platform should be usable by people with a wide range of abilities, including those with visual, auditory, motor, and cognitive impairments
  + **Responsiveness:** The frontend should be responsive, working seamlessly across various screen sizes and devices (desktops, tablets, and mobile phones). The layout should adapt to different screen sizes without losing functionality or visual appeal.

**3.3 Planning and Scheduling**

Effective project planning and scheduling are crucial for the successful completion of any software development project. These processes involve defining project tasks, estimating their durations, identifying dependencies, allocating resources, and creating a realistic timeline. This section outlines the planning and scheduling methodologies employed for the Health Consultant project, with a focus on the application of Gantt charts and PERT charts.

**3.3.1 Gantt Chart**

A Gantt chart is a visual project management tool that provides a timeline-based representation of project tasks. Each task is depicted as a horizontal bar, with the length of the bar indicating the task's duration. The Gantt chart also displays task dependencies, start dates, end dates, and milestones. In the Health Consultant project, the Gantt chart was used to:

* + **Visualize Project Timeline:** The Gantt chart provided a clear and concise overview of the project's timeline, allowing stakeholders to easily track progress and identify potential delays.
  + **Define Project Tasks:** The Gantt chart facilitated the breakdown of the project into smaller, manageable tasks. Each task was clearly defined with a description, duration, and assigned resources.
  + **Identify Task Dependencies:** The Gantt chart highlighted the dependencies between tasks, ensuring that tasks were completed in the correct order. This helped prevent delays and ensured that resources were allocated efficiently.
  + **Track Project Progress:** The Gantt chart was regularly updated to reflect the project's progress. This allowed the project team to identify any deviations from the planned timeline and take corrective action.
  + **Allocate Resources:** The Gantt chart helped in allocating resources (e.g., developers, testers) to specific tasks. This ensured that resources were used effectively and that tasks were completed on time.

In essence, the Gantt chart served as a communication tool, providing a shared understanding of the project's timeline, tasks, and dependencies. It helped to keep the project on track and ensure that all stakeholders were informed of the project's progress.

The Gantt chart will visually showcase the timeline for completing various phases of the Health Consultant project. This typically includes:

* + **Requirements Gathering and Analysis:** The duration allocated for collecting and analyzing user needs.
  + **System Design:** The timeframe for designing the system architecture, database schema, and user interface.
  + **Frontend Development:** The period dedicated to building the user interface using React, JavaScript, and related technologies.
  + **Backend Development:** The time allocated for developing the server-side logic using Node.js, Express.js, and MySQL.
  + **Testing and Quality Assurance:** The duration dedicated to testing the system for functionality, security, and performance.
  + **Deployment:** The timeframe for deploying the system to a production environment.

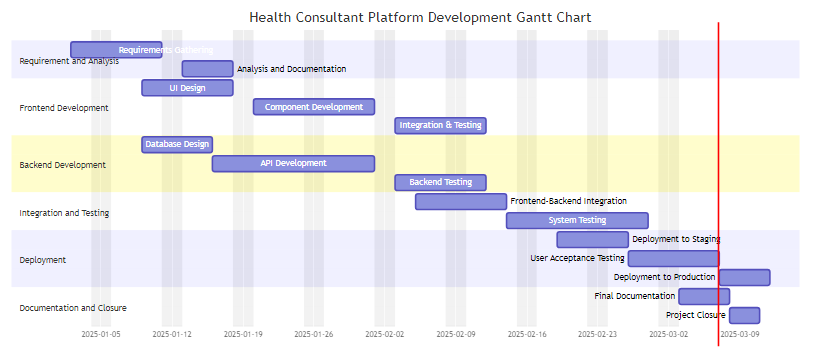


Fig. 1 Gantt Chart

**3.3.2 PERT Chart**

* **Content:**

A PERT (Program Evaluation and Review Technique) chart is another project management tool used to analyze and represent the tasks involved in completing a project. Unlike a Gantt chart, which focuses on the timeline, a PERT chart emphasizes the dependencies between tasks and the critical path. The PERT chart utilizes a network diagram to visualize the sequence of tasks and their interrelationships. In the Health Consultant project, the PERT chart was used to:

* + **Identify Critical Path:** The PERT chart helped identify the critical path, which is the sequence of tasks that determines the shortest possible duration for the project. Any delays in tasks on the critical path would directly impact the project's overall completion time.
  + **Estimate Task Durations:** The PERT chart used three-point estimation to estimate task durations: optimistic, pessimistic, and most likely. This provided a more realistic assessment of task completion times compared to single-point estimates.
  + **Manage Uncertainty:** The PERT chart helped to manage uncertainty by identifying potential risks and their impact on the project timeline. This allowed the project team to develop contingency plans to mitigate these risks.
  + **Optimize Resource Allocation:** The PERT chart facilitated the allocation of resources to tasks based on their criticality and dependencies. This ensured that resources were prioritized to tasks on the critical path.

The PERT chart enabled the project team to identify potential bottlenecks, optimize resource allocation, and manage risks effectively. By focusing on the critical path, the project team could prioritize tasks and minimize the overall project duration.

Specifically for the Health Consultant project, the PERT chart would illustrate:

* + **Task Dependencies:** The relationships between different tasks, such as the requirement for the database schema to be completed before backend development can commence.
  + **Critical Path:** The sequence of tasks that must be completed on time to avoid delaying the entire project.
  + **Expected Completion Time:** The statistically determined estimate of the project's total duration, considering optimistic, pessimistic, and most likely scenarios.
  + **Slack or Float:** The amount of time a task can be delayed without impacting the project's overall completion date.
  + **Probability of Completion:** Provide statistical data of how probable it is for the project to be delivered at a certain time

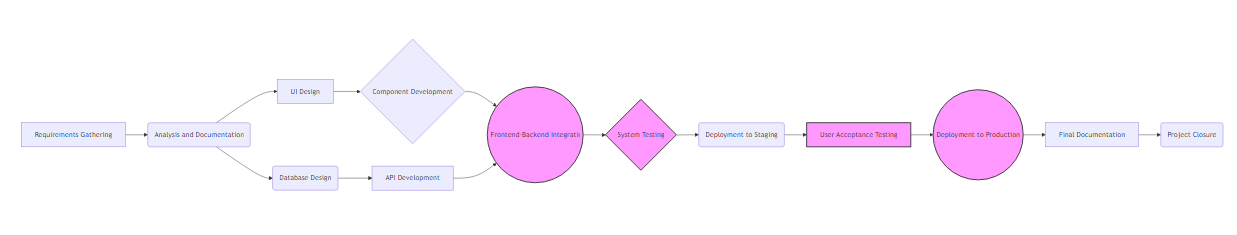


Fig. 2. PERT Chart

**3.4 Software and Hardware Requirements**

This section outlines the necessary software and hardware infrastructure to support the development, deployment, and operation of the Health Consultant platform. Careful consideration of these requirements is crucial for ensuring optimal performance, scalability, security, and user experience.

**I. Software Requirements:**

* **A. Development Environment:**
  + **Operating System:** A stable and widely supported operating system for development purposes. Recommended options include:
    - Windows 10 or later
    - macOS Mojave or later
    - Ubuntu 18.04 LTS or later
    - Choosing between the Operating systems will come down to the Developer preference and Budget. For an individual one should choose the one they are comfortable with, but for large scale development organizations should choose the most cost effective option. As the final application is a web application, so most Operating systems should be compatible with it.
  + **Integrated Development Environment (IDE):** A powerful IDE to facilitate code editing, debugging, and project management. Recommended options include:
    - Visual Studio Code: A lightweight and extensible IDE with excellent support for JavaScript, React, and Node.js development.
    - WebStorm: A commercial IDE specifically designed for web development, offering advanced features for code analysis, refactoring, and debugging. (For big scale projects)
    - Sublime Text: A source code editor with a python API. It natively supports many programming languages and markup languages, and functions can be extended by users with plugins, typically community-built and free of charge.
  + **Node.js and npm:** Node.js runtime environment and npm (Node Package Manager) for managing backend dependencies.
    - Node.js: Version 16.x or later (LTS recommended)
    - npm: Version 7.x or later (installed with Node.js)
  + **Git:** A distributed version control system for managing source code and collaborating on projects. Git is essential for tracking changes, branching, merging, and resolving conflicts in a multi-developer environment. Also used for hosting and managing deployment with cloud services like vercel, AWS, digital ocean etc.
  + **Docker:** (Optional) A containerization platform for packaging and deploying the application in isolated environments. Docker simplifies the deployment process and ensures consistency across different environments. Useful for running scalable and easily deployable applications.
    - Docker helps with faster deployment, increases portability and creates consistency across different operating system.
* **B. Frontend Dependencies:**
  + **React:** (As mentioned earlier) Version 17.x or later. React provides the component-based support.
  + **Axios:** A promise-based HTTP client for making API requests from the frontend to the backend.
  + **React Router:** A routing library for managing navigation and defining routes in the React application.
  + **Tailwind CSS:** (As mentioned earlier) for styling components to give it that nice visual appeal. The utility first CSS framework helps speed up the development time.
  + **Zustand:** A library for state management in react framework.
  + **Lucide React:** An icon Library for React apps
  + **MUI:** An open-source project that features a set of UI components that implement Google’s Material Design
* **C. Backend Dependencies:**
  + **Express.js:** (As mentioned earlier) Version 4.x or later. Used as the backend framework for the project.
  + **mysql2:** A MySQL client for Node.js, used for connecting to and interacting with the MySQL database.
  + **Sequelize:** An ORM for easier database manipulation. Used for connecting, querying, modifying the database.
  + **jsonwebtoken:** A library for creating and verifying JSON Web Tokens (JWTs) for user authentication and authorization.
  + **bcrypt:** A library for hashing passwords to protect user credentials.
  + **cors:** Middleware for enabling Cross-Origin Resource Sharing (CORS) to allow requests from different domains.
  + **dotenv:** A module for loading environment variables from a .env file.
  + **morgan:** A Node.js and Express middleware to log HTTP requests and errors, to simplify debugging.
  + **helmet:** A library that adds HTTP headers that will help protect the website from well-known web vulnerabilities.
  + **Express-rate-limit:** The express-validator library provides a set of express.js middlewares that can be used to validate data
  + **Multer:** The Multer package is a node.js middleware for handling multipart/form-data, which is primarily used for uploading files.
* **D. Testing and Quality Assurance:**
  + **Jest:** A JavaScript testing framework used for unit testing and integration testing.
  + **Supertest:** A library for testing Node.js HTTP servers, providing a high-level abstraction for sending requests and asserting responses.
* **E. Deployment Environment:**
  + **Operating System:** A stable and secure operating system for hosting the backend server. Recommended options include:
    - Ubuntu 20.04 LTS or later
    - CentOS 7 or later
  + **Web Server:** A web server for serving static assets and proxying requests to the backend server. Recommended options include:
    - Nginx: A high-performance web server and reverse proxy server.
    - Apache HTTP Server: A widely used web server with a rich feature set.
      * Choosing a web server and OS depends on the scale and need of the website. For big scale project where High Availability (HA) is very important there are different things to consider.
  + **Database Server:** A MySQL server for hosting the platform's database. The server should be configured for optimal performance and security.
  + **PM2** a process manager for Node.js applications, which is also compatible for other languages.
  + **Load Balancer:** A load balancer to distribute traffic across multiple backend servers. A load balancer improves scalability, availability, and fault tolerance. Option can be using a CDN like CloudFlare.
* **F. Security Software:**
  + **Firewall:** A firewall is an essential security tool that provides a protective barrier between your server and the outside network, allowing you to control network traffic.
  + **Antivirus:** Install an antivirus software for your computer to protect against viruses and other malicious software.

**II. Hardware Requirements:**

* **A. Development Environment:**
  + **Processor:** Multi-core processor (Intel Core i5 or AMD Ryzen 5 or better) for efficient code compilation and execution.
  + **Memory:** At least 8 GB of RAM (16 GB recommended) for running the IDE, development tools, and multiple browser tabs simultaneously.
  + **Storage:** At least 256 GB SSD (Solid State Drive) for fast read/write speeds and improved overall performance. An SSD drive would ensure rapid code storage.
  + **Display:** A monitor with a resolution of at least 1920x1080 (Full HD) for comfortable code editing and debugging.
* **B. Backend Server:**
  + **Processor:** Multi-core server-grade processor (Intel Xeon or AMD EPYC) for handling concurrent requests and executing backend logic efficiently.
  + **Memory:** At least 16 GB of RAM (32 GB or more recommended) for caching data and handling a large number of concurrent connections. More RAM means better performance.
  + **Storage:** At least 500 GB SSD (RAID configuration recommended) for fast data access and improved reliability. RAID configuration increases availability and accessibility.
  + **Network:** High-bandwidth network connection for handling large volumes of traffic.
* **C. Database Server:**
  + **Processor:** Multi-core server-grade processor (Intel Xeon or AMD EPYC) optimized for database workloads.
  + **Memory:** At least 32 GB of RAM (64 GB or more recommended) for caching data and improving query performance. More RAM means better performance.
  + **Storage:** At least 1 TB SSD (RAID 10 configuration recommended) for fast data access and improved data redundancy. RAID configuration helps improve availability and security of data.
  + **Network:** High-bandwidth network connection for handling large volumes of data transfer.
* **D. Client Devices (Patients and Consultants):**
  + **Processor:** Minimum dual-core processor for running the web browser and accessing the platform.
  + **Memory:** At least 4 GB of RAM for smooth browsing and video conferencing.
  + **Storage:** Sufficient storage space for caching web content and storing temporary files.
  + **Network:** Stable internet connection (broadband recommended) for reliable access to the platform and video conferencing capabilities.
  + **Webcam and Microphone:** For video consultations.
  + **Speakers or Headphones:** For clear audio communication.

These software and hardware requirements provide a solid foundation for building and deploying the Health Consultant platform. However, specific requirements may need to be adjusted based on the project's evolving needs and the target user base.

**3.5 Preliminary Product Description**

The Health Consultant platform is a comprehensive online telehealth solution designed to connect patients with qualified healthcare professionals for remote consultations, personalized health advice, and ongoing care management. This section provides a detailed description of the platform's features, functionalities, and user experience.

**I. Key Features and Functionalities:**

* + **A. User-Friendly Interface:** The platform boasts a clean, intuitive, and user-friendly interface designed for ease of navigation and accessibility across various devices. The interface is designed with a focus on visual clarity and a consistent design language to provide a seamless user experience.
  + **B. Secure User Authentication:** The platform implements secure user authentication mechanisms to protect user accounts and sensitive data. Users are required to create strong passwords and verify their email addresses upon registration. Two-factor authentication (2FA) may be implemented in the future for enhanced security.
  + **C. Role-Based Access Control:** The platform enforces role-based access control to restrict access to specific features and functionalities based on user roles (patients, consultants, administrators). This ensures that users only have access to the information and tools that are relevant to their role.
  + **D. Consultant Profiles:** The platform provides detailed consultant profiles that showcase their qualifications, specialities, expertise, consultation fees, availability, and ratings/reviews. Patients can easily search and browse consultant profiles to find the best match for their needs.
  + **E. Advanced Search & Filtering:** Patients can refine the search and the number of consutants with a variety of filters.
  + **F. Appointment Scheduling & Management:** Patients can schedule appointments with consultants based on their availability. The platform provides a calendar interface for selecting available dates and times. Patients and consultants can manage their bookings, including viewing, rescheduling, and canceling appointments.
  + **G. Secure Payment Gateway:** The platform is integrated with a secure payment gateway to process online payments for consultations. The payment gateway supports various payment methods, such as credit cards, debit cards, and digital wallets, while also following industry standards and keeping customers financial data safe.
  + **H. Real-Time Chat & Messaging:** Patients and consultants can communicate in real-time through a secure chat interface. The chat interface supports text messaging, file sharing (e.g., medical reports, prescriptions), and potentially video conferencing in future versions.
  + **I. Health Record Management:** Patients can securely upload and store their medical history, prescriptions, and other relevant health information on the platform. Consultants can access patient health records for scheduled appointments, ensuring a comprehensive understanding of the patient's medical background.
  + **J. Video Conferencing (Future Enhancement):** The platform is designed to be easily extendable with video conferencing capabilities. In future versions, patients and consultants will be able to conduct virtual consultations via secure video connections, further enhancing the remote healthcare experience.
  + **K. Rating and Review System:** Patients can rate and review consultants based on their experience. This provides valuable feedback to other patients and helps ensure the quality of services offered on the platform.
  + **L. Administrative Dashboard:** A centralized dashboard provides administrators with comprehensive tools for managing users, consultants, bookings, system settings, and payment processing. This enables efficient platform management and ensures optimal system performance.
  + **M. Notifications:** Push, Email and SMS notifications to remind both patient and doctor of meetings and check-ups.

**II. Target Users:**

* + **A. Patients:** Individuals seeking convenient, affordable, and personalized healthcare advice from qualified professionals. Patients can use the platform to:
    - Search for and connect with suitable consultants.
    - Schedule and manage appointments.
    - Communicate with consultants via chat and messaging.
    - Securely store and share their health records.
    - Rate and review consultants.
  + **B. Consultants:** Licensed healthcare professionals seeking to expand their practice, connect with more patients, and offer remote consultations. Consultants can use the platform to:
    - Create and manage their professional profiles.
    - Set their availability and consultation fees.
    - Communicate with patients via chat and messaging.
    - Access patient health records for scheduled appointments.
    - Track their earnings and view their payment history.
    - Manage their booking history
  + **C. Administrators:** Platform administrators responsible for managing the system, ensuring its smooth operation, and maintaining the quality of services. Administrators can:
    - Manage user accounts and permissions.
    - Approve or reject consultant registrations.
    - Configure system settings.
    - Monitor system performance and security.
    - Process refunds and manage payments.

**III. User Experience (UX) Considerations:**

* + **A. Simplicity and Ease of Use:** The platform's design prioritizes simplicity and ease of use. Navigation is intuitive, and key features are easily accessible. The platform is designed to be usable by individuals with varying levels of technical expertise.
  + **B. Responsiveness and Accessibility:** The platform is designed to be responsive and accessible across various devices, including desktops, laptops, tablets, and mobile phones. The platform complies with WCAG guidelines to ensure accessibility for users with disabilities.
  + **C. Visual Appeal:** The platform features a modern and visually appealing design that enhances user engagement and creates a positive experience. The colour scheme, typography, and imagery are carefully selected to convey a sense of trust, professionalism, and care.
  + **D. Performance and Reliability:** The platform is optimized for performance and reliability. Page load times are minimized, and the system is designed to handle a large number of concurrent users without performance degradation.

**IV. Technology Architecture:**

* + **A. Frontend:** React, JavaScript, JSX, CSS, Tailwind CSS, Material UI
  + **B. Backend:** Node.js, Express.js, MySQL, Sequelize ORM, JWT
  + **C. Deployment:** Cloud-based deployment on platforms like AWS, Azure, or Google Cloud for scalability and reliability.

The Health Consultant platform is designed to be a comprehensive and user-friendly telehealth solution that empowers patients to take control of their health journey and connects them with the expertise they need. Its robust features, secure architecture, and patient-centred design make it a valuable tool for transforming the healthcare landscape.

**3.6 Conceptual Model**

The conceptual model provides a high-level representation of the Health Consultant platform, illustrating its key entities, relationships, and interactions. This section includes several diagrams to depict different aspects of the system, providing a holistic understanding of its architecture and functionality.

The following diagrams provide different views of the system, its entities, and the way those entities interact with each other. The aim of these diagrams are to provide a clear understanding of the flow in the system.

**1. Event Table**

The Event Table provides a structured overview of the key events within the Health Consultant platform. It outlines the events, their triggers, the actors involved, and the resulting system behaviour. This table helps to identify and understand the dynamic aspects of the system.

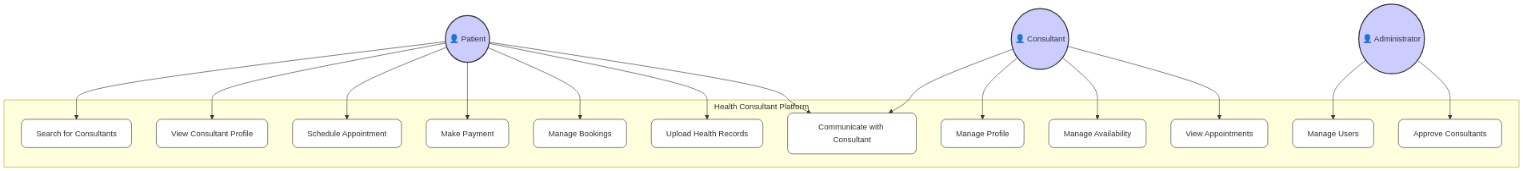
* + **Event Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Event | Trigger | Actor(s) | System Behaviour |
| User Registration | User submits registration form | User | Create new user account, send verification email. |
| User Login | User submits login form | User | Authenticate user, grant access to appropriate dashboard. |
| Profile Update | User updates profile information | User | Update user profile data, store changes in the database. |
| Consultant Search | Patient searches for consultants | Patient | Display consultant search results based on criteria. |
| Appointment Booking | Patient schedules an appointment | Patient, Consultant | Create new booking record, send confirmation emails. |
| Payment Processing | Patient completes payment for appointment | Patient, Payment Gateway | Process payment, update booking status, record payment transaction. |
| Message Send | User sends a message to another user | User | Store the message, display it to recipient. |
| Health Record Upload | Patient uploads a health record | Patient | Store health record securely, link to patient profile. |
| Consultant Approval | Administrator approves a consultant registration | Administrator | Update consultant status, grant access to consultant functionalities. |
| Consultant Rejection | Admin rejects a consultant registration | Administrator | Update consultant status, restrict access to consultant functionalities. |
| Booking Cancellation | Admin cancels a booking | User, Consultant | Booking status to canceled and user will be credited to the account. |

2. Use Case Diagram

The Use Case Diagram illustrates the interactions between actors (users) and the system. It identifies the primary functionalities offered by the platform and how different users interact with them. This diagram provides a high-level view of the system's core features.

* **Description:**
  + This UML depicts the actions and interactions possible between the actors in the application. The actions are placed within the rectangle with all the actors placed outside of it.
  + The patient is able to search for consultants, view consultant profiles, schedule appointments, make payments, manage bookings, upload health records and communicate with consultants.
  + The consultant is able to manage thier profile, manage availability, view appointments, and communicate with Patients.
  + The administrator is able to manage users, approve consultants, manage bookings



**3. Sequence Diagram**

The Sequence Diagram illustrates the sequence of interactions between different components of the system for a specific use case. In this case, the diagram depicts the process of a patient scheduling an appointment. It showcases the flow of messages and actions between the patient, the frontend, the backend server, and the database.

* **Description:**
  + This UML depicts the process for making an appointment with the health consultant.
  + First the patient selects consultant and time slot.
  + The frontend will send appointment request to the backend.
  + The backend will communicate with the database and check for consultant availability.
  + If the consultant is available the backend will create a booking record in the database and it will return a booking confirmation to the backend.
  + The backend then will send the booking confirmation to the frontend, to be displayed to the user.
  + If the consultant is not available the backend will return an error to the frontend to be displayed to the patient.

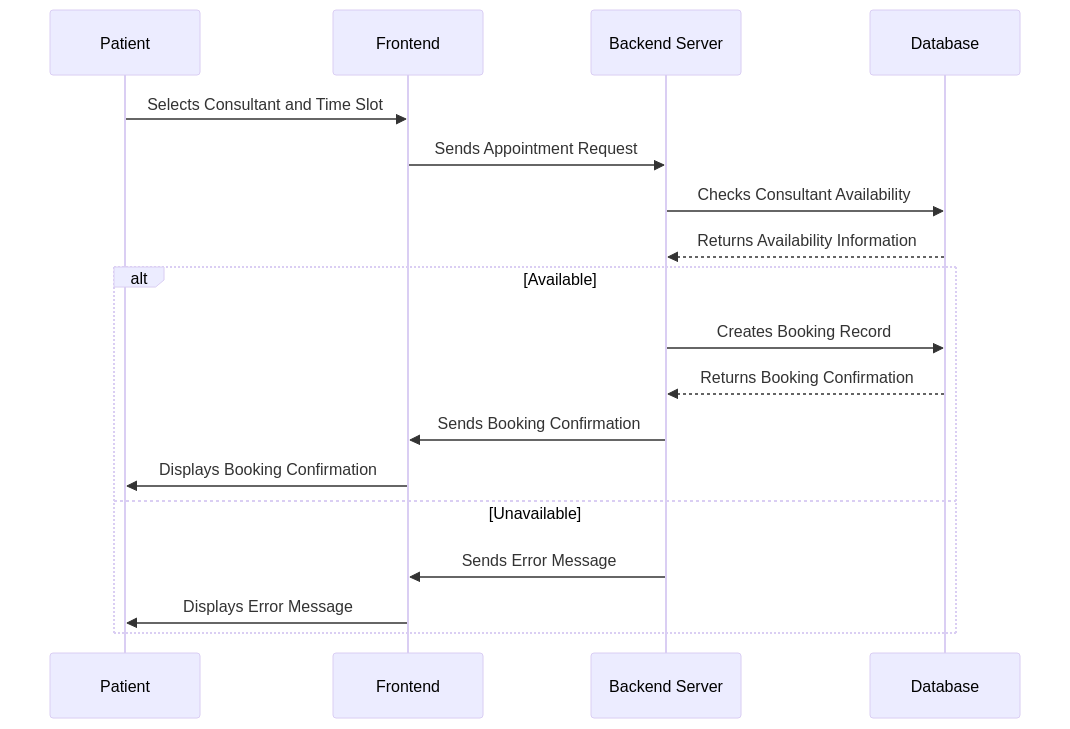
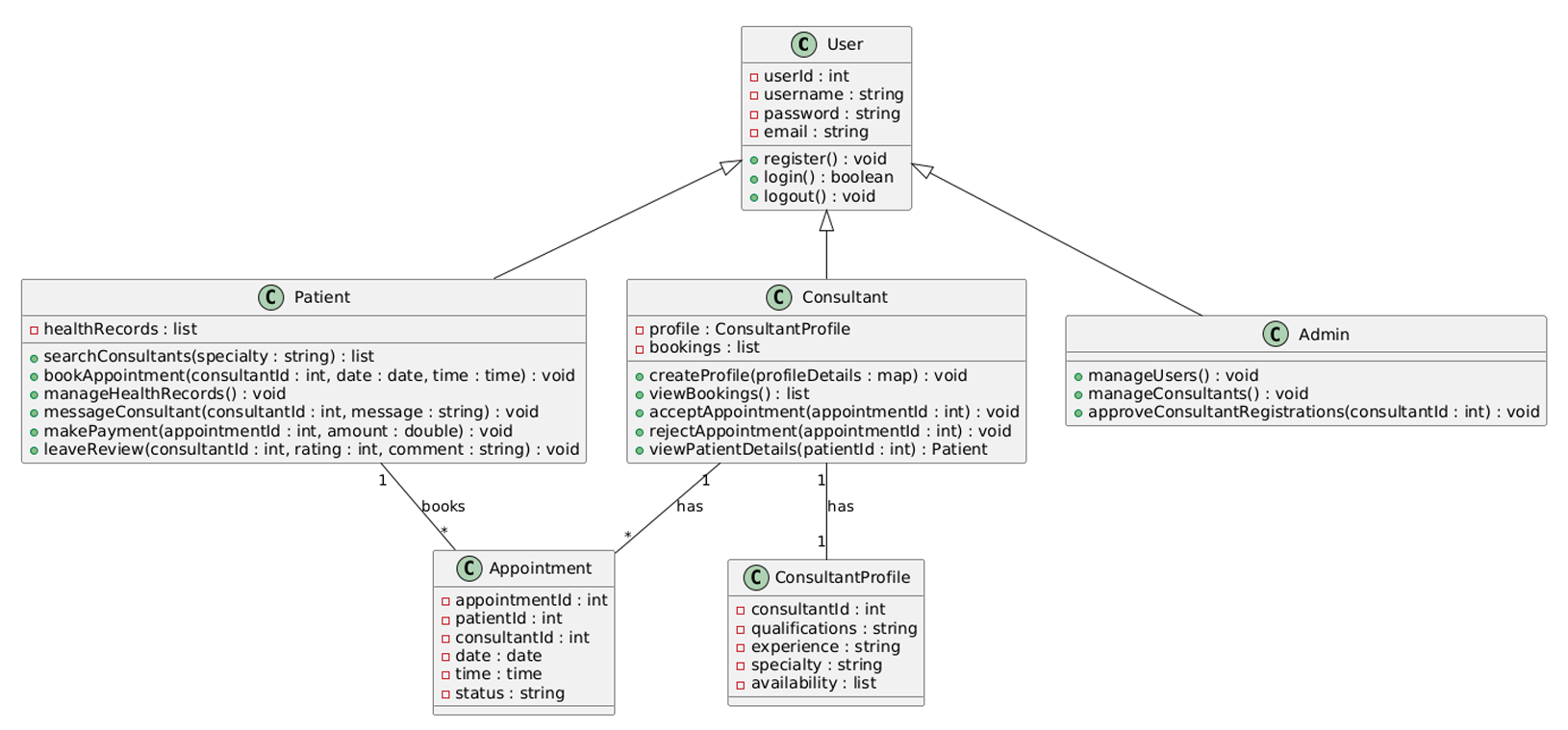


Fig. 3 Sequence Diagram

**4. Class Diagram**

The Class Diagram illustrates the key entities (classes) in the system and their relationships. It provides a static view of the system's data structure and how different entities are interconnected. This diagram helps understand the system's data model and how it supports the various functionalities.

* **Description:**
  + The UML represents the data model of the system and the class relationship with one another.
  + The User class can register and login to the website and has an id, fullname, email, password, role and isConsultant variables.
  + The Consultant class extends the User class. It represents the various skills of the doctors in the website. This class can createProfile and updateAvailability and also contains id, specialty, qualification, and availability variables.
  + The Booking class is responsible for tracking all the appointments in the website. This class has a schedule and cancel function and also contains variables like id, userId, consultantId, data, time and status.

**5. State Diagram**

The State Diagram illustrates the different states of a booking and the transitions between those states. This diagram helps understand the lifecycle of a booking, from its creation to its completion or cancellation. It also clarifies the conditions that trigger state transitions.

* **Description:**
  + This UML represents the status of a booking. This helps to understand what events cause the booking to have a certain status.
  + When a patient schedules an appointment, the booking status is 'Pending'.
  + The consultat can then either accept the booking which then the status will change to 'Accepted'.
  + Or the consultant rejects the booking, which ends the life-cycle of the booking.
  + If the consultant accepts the booking then the patient can decide to can cancel the schedule, in doing so the bookings life cycle will also end.
  + However, if the booking is not cancelled, then the consultation will occur and then the booking's life cycle is complete.

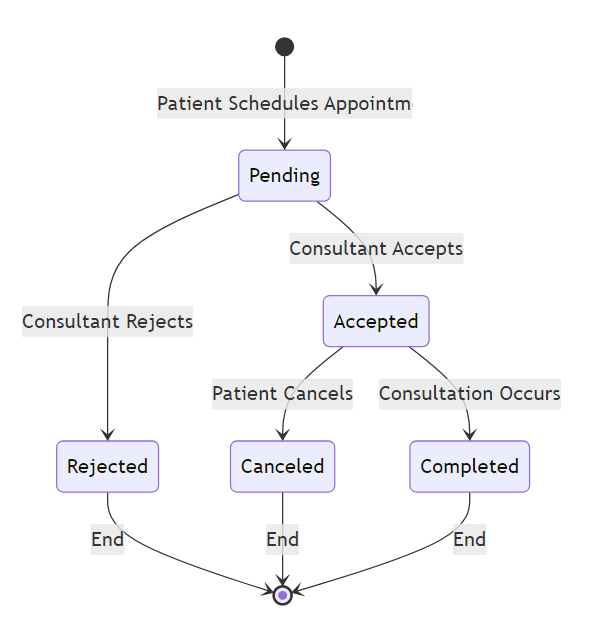


Fig. 5 State Diagram

**4. SYSTEM DESIGN**

This section provides a detailed overview of the system architecture and module design of the Health Consultant platform. It outlines the major components, their interactions, and the technologies used to implement them. A clear understanding of the system design is crucial for ensuring scalability, maintainability, and security.

**4.1 Basic Modules**

The Health Consultant platform is structured into several core modules, each responsible for specific functionalities. This modular design promotes code reusability, simplifies maintenance, and enhances the overall scalability of the system. The following are the basic modules of the platform:

**1. User Management Module:**

* + **Description:** This module handles user registration, login, profile management, and role-based access control. It is responsible for creating, updating, and deleting user accounts, as well as authenticating users and authorizing access to protected resources.
  + **Components:**
    - Registration Component: Provides user interface for registering as the 3 types of users for the application.
    - Login Component: Provides the log in route and functions. Also authenticates all users.
    - Profile Management Component: Provides management for every single users profile
    - Role-Based Component: A component that allows the admin to approve and decline newly registered consutants.
  + **Responsibilities:**
    - Handle user registration requests and validate user input.
    - Authenticate user credentials against the database.
    - Store user information securely in the database.
    - Implement role-based access control to restrict access to certain features.
    - Manage user sessions and handle user logout requests.
    - Manage user authentication through JSON Web Tokens (JWT).
  + **Technologies Used:** React, Node.js, Express.js, MySQL, bcrypt, jsonwebtoken

**2. Consultant Management Module:**

* + **Description:** This module handles the creation, management, and retrieval of consultant profiles. It is responsible for storing consultant information, such as qualifications, specialities, expertise, consultation fees, and availability. The Module handles all doctors and designated personnel information.
  + **Components:**
    - Consultant Profile Component: Provide an accessible and manageable UI for the consultant to build his/her profile.
    - Profile View Component: Creates an interface for the profile to be viewed to other users of the application.
    - Consultant Document Retrieval: Helps retrieve and authorize access to important documents
  + **Responsibilities:**
    - Create and manage consultant profiles.
    - Validate consultant qualifications and expertise.
    - Store consultant information securely in the database.
    - Approve or reject consultant registrations.
    - Manage consultant availability and consultation fees.
    - Manages the approval and denail of different consultant. Also provides an easy way to view and verify consultant through file access.
  + **Technologies Used:** React, Node.js, Express.js, MySQL

**3. Booking Management Module:**

* + **Description:** This module handles the scheduling, management, and cancellation of appointments between patients and consultants. It is responsible for managing appointment slots, sending confirmation emails, and handling cancellations and refunds.
  + **Components:**
    - Consultant and time slot selection: Component for selecting the required and needed consultant and time slots.
    - Management of bookings Component: An UI for users to manage all their booking.
    - Calendar and time slot UI: An UI for users to select the accurate date and time.
    - Booking acceptance and denial Component: An UI for the consultant to accept and reject all booking to then manage those said bookings.
  + **Responsibilities:**
    - Schedule appointments based on consultant availability.
    - Send confirmation emails to patients and consultants.
    - Manage appointment cancellations and refunds.
    - Update appointment status (e.g., pending, confirmed, canceled, completed).
  + **Technologies Used:** React, Node.js, Express.js, MySQL

**4. Payment Management Module:**

* + **Description:** This module handles the processing of online payments for consultations. It is responsible for integrating with a secure payment gateway, encrypting sensitive data, and complying with PCI DSS standards.
  + **Components:**
    - Payment integration Component: A secure integration for credit and debit cards payments.
    - Tracking module Component: a tracking module for designated personnels to check where the payments are coming from.
    - Refund management Component: A way for admins to refund if any discrepancy arises.
  + **Responsibilities:**
    - Integrate with a secure payment gateway (e.g., Stripe, PayPal).
    - Process online payments for consultations.
    - Encrypt sensitive payment data.
    - Comply with PCI DSS standards.
    - Process refunds for canceled appointments.
    - Tracks income and expenditure
  + **Technologies Used:** Node.js, Express.js, Payment Gateway API

**5. Messaging and Chat Module:**

* + **Description:** This module provides real-time chat and messaging functionality for communication between patients and consultants. It is responsible for handling text messages, file sharing, and potentially video conferencing (in future versions).
  + **Components:**
    - Realtime Messaging: UI Component and database access code for the user and consultant to communicate in real-time.
    - Messaging Request Component: To request access to messaging (request/approve).
  + **Responsibilities:**
    - Handle real-time text messages, file sharing, and potentially video conferencing.
    - Store message history for future reference.
    - Send notifications to users when they receive new messages.
    - Handle encryption of sensitive data for transfer
  + **Technologies Used:** React, Node.js, Express.js, Socket.IO, MySQL

**6. Health Record Management Module:**

* + **Description:** This module handles the secure storage and retrieval of patient health records. It is responsible for storing medical history, prescriptions, and other relevant health information. Also responsible for the access levels and who can request access.
  + **Components:**
    - Request access to medical history Component: Requesting Component for users to get access to the health record.
    - Secure storage and encryption Component: Securely store and encrypt the data to ensure no external body can retrieve the data without authorization.
    - Upload and verification component Proper data cleaning and verification so that non-sanitized data is not uploaded to the website.
  + **Responsibilities:**
    - Securely store patient health records.
    - Control access to health records based on user roles and permissions.
    - Implement encryption mechanisms to protect sensitive data.
  + **Technologies Used:** Node.js, Express.js, MySQL

These basic modules work together to provide a comprehensive and integrated telehealth experience. Each module is designed to be independent and reusable, promoting code maintainability and scalability.

**4.2 Data Design**

This section describes the data model used to represent the information stored within the Health Consultant platform. A well-designed data model is crucial for ensuring data integrity, consistency, and efficient data access. This section will delve into the schema design, database relationships, and security considerations for the platform's data.

**4.2.1 Schema Design with Data Integrity and Constraints**

* **Content:**

The following tables are the key structures within the MySQL database, responsible for maintaining the platform's data. They are designed with focus on data integrity and constraints enforced.

* + **Users Table:**
    - Purpose: Stores user information, including patients, consultants, and administrators.
    - Schema:
      * id (INT, PRIMARY KEY, AUTO\_INCREMENT): Unique identifier for the user.
      * fullName (VARCHAR(255), NOT NULL): User's full name.
      * email (VARCHAR(255), NOT NULL, UNIQUE): User's email address (must be unique).
      * password (VARCHAR(255), NOT NULL): Hashed password for user authentication.
      * role (VARCHAR(50), NOT NULL, DEFAULT 'user'): User's role (user, consultant, or admin).
      * phone (VARCHAR(20)): User's phone number.
      * profilePicture (VARCHAR(255)): Path to the user's profile picture.
      * bloodGroup (VARCHAR(10)): User's blood group (if applicable).
      * medicalHistory (TEXT): User's medical history (if applicable).
      * currentPrescriptions (TEXT): User's current prescriptions (if applicable).
      * isConsultant (BOOLEAN, DEFAULT 0): Indicates whether the user is a consultant.
      * bio (TEXT): Consultant's biography.
      * qualification (VARCHAR(255)): Consultant's qualifications.
      * areasOfExpertise (TEXT): Consultant's areas of expertise.
      * specialty (VARCHAR(255)): Consultant's specialty.
      * availability (TEXT): Consultant's availability schedule (JSON format).
      * bankAccount (VARCHAR(255)): Consultant's bank account number.
      * consultingFees (DECIMAL(10, 2)): Consultant's consulting fees.
      * isApproved (BOOLEAN, DEFAULT 0): Indicates whether the consultant is approved by the administrator.
    - Data Integrity and Constraints:
      * email: UNIQUE constraint ensures that each user has a unique email address.
      * password: Stored as a hashed value using bcrypt to protect against unauthorized access.
      * role: DEFAULT constraint ensures that the default role is 'user' if not specified during registration.
      * isConsultant: DEFAULT constraint initializes the consultant status to 0 (false) if not specified.
      * isApproved: DEFAULT constraint initializes the isApproved status to 0 (false) if not specified.
  + **Bookings Table:**
    - Purpose: Stores information about scheduled appointments between patients and consultants.
    - Schema:
      * id (INT, PRIMARY KEY, AUTO\_INCREMENT): Unique identifier for the booking.
      * userId (INT, NOT NULL): Foreign key referencing the Users table, indicating the patient who scheduled the appointment.
      * consultantId (INT, NOT NULL): Foreign key referencing the Users table, indicating the consultant for the appointment.
      * date (DATE, NOT NULL): Date of the appointment.
      * time (TIME, NOT NULL): Time of the appointment.
      * status (VARCHAR(50), NOT NULL): Status of the appointment (e.g., pending, confirmed, canceled, completed).
    - Data Integrity and Constraints:
      * userId: FOREIGN KEY constraint ensures that the userId references a valid user in the Users table.
      * consultantId: FOREIGN KEY constraint ensures that the consultantId references a valid consultant in the Users table.
      * date and time: NOT NULL constraints ensure that these fields are always populated.
      * status: NOT NULL constraints ensure that this fields are always populated.
  + **Payments Table:**
    - Purpose: Stores payment information for consultations.
    - Schema:
      * id (INT, PRIMARY KEY, AUTO\_INCREMENT): Unique identifier for the payment.
      * bookingId (INT, NOT NULL): Foreign key referencing the Bookings table, indicating the booking for which the payment was made.
      * userId (INT, NOT NULL): Foreign key referencing the Users table, indicating the user who made the payment.
      * amount (DECIMAL(10, 2), NOT NULL): Amount paid for the consultation.
      * paymentDate (DATETIME, NOT NULL): Date and time of the payment.
      * status (VARCHAR(50), NOT NULL): Status of the payment (e.g., pending, completed, refunded).
    - Data Integrity and Constraints:
      * bookingId: FOREIGN KEY constraint ensures that the bookingId references a valid booking in the Bookings table.
      * userId: FOREIGN KEY constraint ensures that the userId references a valid user in the Users table.
      * amount and paymentDate: NOT NULL constraints ensure that these fields are always populated.
      * status: NOT NULL constraints ensure that these fields are always populated.
  + **HealthRecords Table:**
    - Purpose: Stores patient health records.
    - Schema:
      * id (INT, PRIMARY KEY, AUTO\_INCREMENT): Unique identifier for the health record.
      * userId (INT, NOT NULL): Foreign key referencing the Users table, indicating the patient who owns the record.
      * medicalHistory (TEXT): Patient's medical history.
      * ongoingTreatments (TEXT): Patient's ongoing treatments.
      * prescriptions (TEXT): Patient's current prescriptions.
    - Data Integrity and Constraints:
      * userId: FOREIGN KEY constraint ensures that the userId references a valid user in the Users table.
* **Reviews Table:**

Purpose: Stores feedback/reviews posted by users for doctors/consultants

Schema:

`id` (INT, PRIMARY KEY, AUTO\_INCREMENT): Unique identifier for the reviews.

`userId` (INT, NOT NULL): Foreign key referencing the `Users` table, indicating the patient who wrote the review.

`consultantId` (INT, NOT NULL): Foreign key referencing the `Users` table, indicating the doctor, for whom the review was written.

`rating` (INT): ratings for the doctors. Has a constraint of being between 1 and 5.

`review` (TEXT): Review, patients have written for the doctors.

Data Integrity and Constraints:

`userId`: FOREIGN KEY constraint ensures that the `userId` references a valid user in the `Users` table.

`consultantId`: FOREIGN KEY constraint ensures that the `consultantId` references a valid consultant in the `Users` table.

`rating`: Should between a range of 1 - 5 (constraint should be made).

* **ChatRequest Table:**
* Purpose: Stores the request a user has made to the consultant.
* Schema:
* `id` (INT, PRIMARY KEY, AUTO\_INCREMENT): Unique identifier for the chat request.
* `consultantId` (INT, NOT NULL): Foreign key referencing the `Users` table, indicating the doctor, for whom the user requested the request.
* `userId` (INT, NOT NULL): Foreign key referencing the `Users` table, indicating the user who requested the request.
* `bookingId` (INT): This indicates all is well with this request.
* `status` (TEXT): To accept the request.
* Data Integrity and Constraints:
* `userId`: FOREIGN KEY constraint ensures that the `userId` references a valid user in the `Users` table.
* `consultantId`: FOREIGN KEY constraint ensures that the `consultantId` references a valid consultant in the `Users` table.
* `bookingId`: FOREIGN KEY constraint ensures that the `bookingId` references a valid booking in the `booking` table.

These tables, and the data they contain, provide the data structure for the application. They are tied together with Foreign Keys, and relationships are carefully crafted so the data flow of the application is as seamless as possible. They also help protect the data to a certain degree to keep the application secure

**4.2.2 Database Relationship Design**

The following relationships exist between the tables in the database:

* + Users to Bookings: A one-to-many relationship exists between the Users table (patients) and the Bookings table. One patient can have multiple bookings.
  + Users to Bookings: A one-to-many relationship exists between the Users table (consultants) and the Bookings table. One consultant can have multiple bookings.
  + Bookings to Payments: A one-to-one relationship exists between the Bookings table and the Payments table. Each booking has one corresponding payment.
  + Users to HealthRecords: A one-to-many relationship exists between the Users table and the HealthRecords table. One patient can have multiple health records.
  + These relationships ensures that the database is consistent and easily maintainable.

**4.2.3 Security Issues**

Protecting sensitive data is a top priority for the Health Consultant platform. The following security measures are implemented to address potential security issues:

* + **Password Hashing:** User passwords are not stored in plain text. Instead, they are hashed using bcrypt, a strong password hashing algorithm. This makes it extremely difficult for attackers to obtain user passwords, even if they gain access to the database.
  + **Data Encryption:** Sensitive data, such as health records and payment information, is encrypted both in transit and at rest. Encryption in transit is achieved using HTTPS (TLS/SSL) to protect data during transmission between the client and the server. Encryption at rest is implemented using database encryption features to protect data stored on the server.
  + **SQL Injection Prevention:** The platform utilizes parameterized queries or an ORM (such as Sequelize) to prevent SQL injection attacks. Parameterized queries ensure that user input is treated as data, not as executable code, preventing attackers from manipulating database queries.
  + **Cross-Site Scripting (XSS) Prevention:** The platform implements measures to prevent XSS attacks. This includes input validation and output encoding to sanitize user input and prevent malicious scripts from being injected into web pages.
  + **Cross-Site Request Forgery (CSRF) Protection:** The platform implements CSRF protection to prevent attackers from tricking users into performing unintended actions. This typically involves using anti-CSRF tokens to verify that requests are originating from legitimate users.
  + **Access Control:** Role-based access control is enforced to restrict access to certain features and data based on user roles. This prevents unauthorized users from accessing sensitive information or performing unauthorized actions.
  + **Regular Security Audits:** The platform undergoes regular security audits and penetration testing to identify and address potential vulnerabilities. Security audits are conducted by qualified security professionals to ensure the effectiveness of the security measures in place.
  + **Rate Limiting:** Implemented to prevent DOS attacks from malicious attackers.

By implementing these security measures, the Health Consultant platform aims to provide a secure and trustworthy environment for patients and consultants to connect and manage their healthcare needs.

**5. IMPLEMENTATION AND TESTING**

This section describes the implementation approaches, testing methodologies, and quality assurance processes employed during the development of the Health Consultant platform. A well-defined implementation and testing strategy is crucial for ensuring the quality, reliability, and security of the system.

**5.1 Implementation Approaches**

The implementation of the Health Consultant platform followed a structured and iterative approach, combining agile methodologies with best practices for software development. This approach enabled the team to adapt to changing requirements, manage risks effectively, and deliver a high-quality product.

**1. Agile Methodology:** The software development lifecycle followed the agile methodology, because there are some benefits to implementing it.

* + **Benefits:**
    - Adaptability:
      * The most important advantage of implementing this approach as development team is more readily willing to adapt to change. The change can come from a new user case or unexpected problem.
    - High Transparency:
      * Throughout the project life-cycle the development team is able to easily communicate with project stakeholder or anyone involved with the project about status and deliverables, increasing the transparency of the project.
    - Higher Product Quality:
      * With the higher level of focus, communication, and planning the final product would be of high value and great quality.
    - Minimum Risk:
      * The agile methodology helps to minimize the risk by constantly releasing features and functions to test for potential problems.
  + **Key Principles of Agile:**
    - **Iterative Development:** The project was divided into short iterations (sprints), typically lasting 2-3 weeks. Each sprint involved planning, development, testing, and review.
    - **Incremental Delivery:** Working software was delivered incrementally at the end of each sprint. This allowed stakeholders to provide feedback and influence the development process.
    - **Customer Collaboration:** Continuous collaboration with stakeholders was maintained throughout the project to ensure that the platform met their needs and expectations.
    - **Self-Organizing Teams:** The development team was self-organizing and empowered to make decisions about how to best achieve the sprint goals.
    - **Continuous Improvement:** Regular retrospectives were conducted to identify areas for improvement in the development process.

**2. Test-Driven Development (TDD):**

* + Although the TDD approach would require a more rigorous approach the goal of the project was to try and write test cases first before writing the product features to create a quality product.
  + **Process:**
    - A test case would be created for a feature.
    - The test is ran, and predictably the test should fail.
    - The feature code is then developed so the test would pass.
    - If refactoring is needed, it is done, and the test must still pass.
    - When this process is done, we have a high degree of assurance that our code does what we expect it to do.

**3. Layered Architecture:**

* + To make the application more manageable and scalable the following architecture should be in place.
  + **Presentation Layer:** UI, responsible for user interaction.
  + **Application Layer:** Service classes and objects for interacting with more than one domain object.
  + **Domain Layer:** Where the core business logic are placed.
  + **Infrastructure Layer:** External elements are placed, which is the ORM and database.

**4. Coding Standards and Best Practices:**

* + The project adhered to well-defined coding standards and best practices to ensure code quality, readability, and maintainability. These practices included:
    - **Code Formatting:** Consistent code formatting using tools like Prettier or ESLint.
    - **Code Documentation:** Clear and concise comments to explain code logic and functionality.
    - **Meaningful Variable and Function Names:** Descriptive names that accurately reflect the purpose of variables and functions.
    - **Avoidance of Code Duplication:** Reusing code through functions and components to reduce redundancy.
    - **Error Handling:** Robust error handling to prevent application crashes and provide informative error messages to users.
    - **Security Best Practices:** Implementing security measures to protect against common web vulnerabilities, such as SQL injection, XSS, and CSRF.

**5. Continuous Integration and Continuous Delivery (CI/CD):** (In Future releases)

* + To enable fast delivery the CI/CD pipline should be set up, although not currently implemented.
  + **Process:**
    - Automated tests running after each push and commit to the repository.
    - Once tests passes a build process will occur to be deployed to a cloud platform.

By adopting this structured and iterative approach, the development team was able to deliver a high-quality Health Consultant platform that met the project's requirements and exceeded stakeholder expectations.

**5.2 Coding Details and Coding Efficiency**

This section details the key coding practices adopted to ensure a clean, efficient, and maintainable codebase. It also describes the approaches taken to optimize coding efficiency and minimize potential errors. The codebase quality and maintainability helps to reduce code debt and simplifies code iteration.

**1. Code Structure and Organization:**

* + **Modular Design:** The codebase was structured into well-defined modules, each responsible for specific functionalities. This modular design promoted code reusability, simplified maintenance, and enhanced the overall scalability of the system.
  + **Component-Based Architecture:** The frontend was built using React's component-based architecture. This allowed for the creation of reusable UI components that could be easily composed to build complex user interfaces. This significantly reduces code duplication and improves code maintainability.
  + **Directory Structure:** A clear and consistent directory structure was maintained throughout the project. This made it easier to locate and manage code files. An Example could be like below:

project-root/

│── client/

│ ├── src/

│ │ ├── components/

│ │ │ ├── Navbar/

│ │ │ │ ├── Navbar.jsx

│ │ │ │ ├── Navbar.css

│ │ │ ├── Button/

│ │ │ │ ├── Button.jsx

│ │ │ │ ├── Button.css

│ │ ├── pages/

│ │ │ ├── Homepage.jsx

│ │ │ ├── ConsultantSearch.jsx

│ │ ├── App.jsx

│ │ ├── index.js

│

│── server/

│ ├── models/

│ │ ├── User.js

│ │ ├── Booking.js

│ ├── routes/

│ │ ├── userRoutes.js

│ │ ├── bookingRoutes.js

│ ├── controllers/

│ │ ├── userController.js

│ │ ├── bookingController.js

│ ├── database.js

│ ├── server.js

**2. Code Readability and Style:**

* + **Consistent Code Formatting:** A consistent code formatting style was enforced using tools like Prettier. This ensured that all code files had a uniform look and feel, making it easier to read and understand.
  + **Code Comments:** Clear and concise comments were added to explain code logic and functionality, particularly for complex or non-obvious sections of code. This made it easier for other developers (or the original developer in the future) to understand the code and make modifications.
  + **Meaningful Variable and Function Names:** Descriptive names were used for variables and functions to accurately reflect their purpose. This made the code more self-documenting and easier to understand.

**3. Code Efficiency and Optimization:**

* + **Efficient Data Structures and Algorithms:** Careful consideration was given to the choice of data structures and algorithms to ensure optimal performance. For example, appropriate data structures were used for caching frequently accessed data.
  + **Database Optimization:** Database queries were optimized to minimize response times. This included using appropriate indexes, avoiding unnecessary joins, and optimizing query logic.
  + **Code Profiling:** Performance profiling tools were used to identify bottlenecks and optimize code execution. This helped to identify and resolve performance issues before they impacted the user experience.

**4. Error Handling and Logging:**

* + **Robust Error Handling:** The codebase included robust error handling mechanisms to prevent application crashes and provide informative error messages to users. Try-catch blocks were used to handle exceptions gracefully.
  + **Centralized Logging:** A centralized logging system was implemented to capture application events, errors, and warnings. This made it easier to debug issues and monitor the health of the system.

**5. Security Best Practices:**

* + **Input Validation:** All user input was validated to prevent malicious data from entering the system. This included validating data types, formats, and lengths.
  + **Output Encoding:** Output encoding was used to prevent Cross-Site Scripting (XSS) attacks. This ensured that user input was treated as data, not as executable code.
  + **Secure Authentication and Authorization:** Strong authentication and authorization mechanisms were implemented to protect sensitive data and prevent unauthorized access. This included using JWTs for authentication and role-based access control for authorization.

**6. Code Review Process:**

* + All code changes were subject to a thorough code review process. This involved having other developers review the code for correctness, readability, maintainability, and security. The goal of the code review process was to identify and address potential issues before the code was merged into the main codebase.

**5.3 Testing**

The testing phase was a crucial step in the project to ensure the website works the way it was intended. Several methods of testing were used to reduce bug occurrence in both the frontend and backend code.

**5.3.1 Unit Testing**

Unit tests were written for individual components and functions to verify their correctness. Unit tests helped to isolate and identify bugs early in the development process.

* + **Focus:** Verifying the behavior of individual units of code (e.g., functions, classes, components).
  + **Scope:** Testing small, isolated pieces of code to ensure they function as expected.
  + **Tools:** Jest, Mocha, Chai
  + **Test Driven Development (TDD)**

**5.3.2 Integration Testing**

Integration tests were performed to verify the interaction between different modules and components of the system. Integration tests helped to identify issues that may arise when different parts of the system are combined.

* + **Focus:** Verifying the interaction between different modules and components.
  + **Scope:** Testing the flow of data and control between different parts of the system.
  + **Tools:** Jest, Mocha, Chai, Supertest
  + Testing all the different interaction between different users.

**5.3.3 Beta Testing**

Beta testing involved releasing the platform to a small group of external users for feedback. Beta testers provided valuable insights into the platform's usability, functionality, and performance in a real-world environment.

* + **Process:** Inviting a select group of external users to test the platform before its official release.
  + **Feedback Collection:** Gathering feedback from beta testers through surveys, interviews, and bug reports.
  + **Bug Fixing:** Addressing any issues identified by beta testers and incorporating their feedback into the final product.
  + **Focus:** Evaluate application in Real World Scenario.
* The purpose of this approach is to understand user experience in the real world by evaluating metrics like User Satisfaction and Bug Reporting

**5.3.4 Test Cases**

* **Content:**

During all stages of testing process all test cases should be clearly written to minimize confusion.  
The following tables represent different kind of test case that are implemented.

**Okay, I can convert those test case descriptions into a more structured tabular format, along with adding more test cases for each.**

**Table 1: Test Cases - Login Functionality**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output |
| Login\_TC001 | Verify successful login with valid credentials | Valid email address, valid password | User is logged in and redirected to the appropriate dashboard based on their role |
| Login\_TC002 | Verify failed login with invalid email address | Invalid email address, valid password | Error message displayed: "Invalid email or password. Please try again." |
| Login\_TC003 | Verify failed login with invalid password | Valid email address, invalid password | Error message displayed: "Invalid email or password. Please try again." |
| Login\_TC004 | Verify failed login with empty email and password fields | Empty email address field, empty password field | Error message displayed: "Please fill in all fields." |
| Login\_TC005 | Verify password reset functionality | Registered email address | Email with a password reset link is sent to the registered email address |
| Login\_TC006 | Verify that session management works when logged In | Valid email address, valid password | Login to website works with access to appropriate services depending on role |
| Login\_TC007 | Verify captcha functionality when logged In | Valid email address, valid password with valid captcha | Login to website works with access to appropriate services depending on role |
| Login\_TC009 | Verify rate limiting on login | Multiple attempts to log in using an existing account | Too many attemps and prompts to try again later |

**Table 2: Test Cases - Add/Modify Doctor Profile Functionality**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output |
| Profile\_TC001 | Verify successful creation of a new doctor profile with valid data | Doctor's name, specialty, qualifications, availability, bio, consulting fees | Doctor profile is created successfully and stored in the database. |
| Profile\_TC002 | Verify successful modification of an existing doctor profile | Existing doctor profile data with updated information | Doctor profile is updated successfully with new information stored in the database. |
| Profile\_TC003 | Verify handling of missing required fields during profile creation | Doctor profile data with missing required fields (e.g., specialty, qualifications) | Error message displayed indicating missing required fields. Profile creation is prevented. |
| Profile\_TC004 | Verify validation of consulting fees to ensure a valid number is entered | Doctor profile data with invalid consulting fees (e.g., non-numeric characters, negative value) | Error message displayed indicating that consulting fees must be a valid number. Profile creation/modification is prevented. |
| Profile\_TC005 | Verify that a doctor can log in and modify their own details | Doctor log in and change in details. | Doctor is able to modify and all changed are applied to the profile. |
| Profile\_TC006 | Verify doctor bio max char limit | Over 2000 chars in the bio | There is a warning for the user to shorten bio due to max characters |

**Table 3: Test Cases - Schedule Booking and Rejecting/Accepting Functionality**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output |
| Booking\_TC001 | Verify successful scheduling of an appointment with an available consultant | Valid date, time, consultant ID | New booking record created in the database, confirmation emails sent to both patient and consultant. |
| Booking\_TC002 | Verify prevention of double-booking for a consultant | Attempt to schedule two appointments with the same consultant at the same time | Error message displayed indicating that the time slot is already booked. Appointment is prevented. |
| Booking\_TC003 | Verify successful cancellation of an appointment by a patient | Valid booking ID, patient credentials | Booking status updated to "canceled," cancellation emails sent to both patient and consultant, refund initiated (if applicable). |
| Booking\_TC004 | Verify successful rejection of an appointment by a consultant | Valid booking ID, consultant credentials | Booking status updated to "rejected," rejection emails sent to both patient and consultant. |
| Booking\_TC005 | Verify if the User books two doctors at same time. | Valid data, same time | The second application would not work, shows an error and doesn't commit changes. |
| Booking\_TC006 | Verify if the user can still book after cancellation (User's side). | The same canceled data but for booking | Data can be re used and will still work |
| Booking\_TC007 | Verify if the user can still book after rejection (Doctor's side). | The same rejected data but for booking | Data can be re used and will still work |

**Table 4: Test Cases - Access Control Functionality**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output |
| Access\_TC001 | Verify that a regular user cannot access the admin dashboard | Regular user credentials | User is redirected to an unauthorized access page or the login page. |
| Access\_TC002 | Verify that an admin user can access the admin dashboard | Admin user credentials | User is logged in and redirected to the admin dashboard with full access to administrative functionalities. |
| Access\_TC003 | Verify that a consultant cannot access administrative user features | Consultant credentials | Consultant is redirected to their profile, and they can not access the admin dashboard." |
| Access\_TC004 | Verify that the admin role can't access user routes | Admin logs in then tries to access the /consultation route | The path should be denied/redirected as that is supposed to be for the user only |
| Access\_TC005 | Verify that the consultant can see only what is appropriate to him | Consultant logs in and sees and monitors data | Doctor can verify user booking, user information, profile and booking history. They can't create users, delete any data of another doctors |

**6. RESULTS AND DISCUSSIONS**

This section presents the results of the testing phase and provides a discussion of the findings. It also includes user documentation to guide users on how to effectively use the Health Consultant platform. The aim is to demonstrate the usability, functions and reliability of the application.

**6.1 Test Reports**

Following rigorous testing of the features and function of the website, we have provided the test reports for a few features in the website.

**Table 1: Test Report - User Registration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output | Result |
| REG\_TC001 | Successful registration with valid inputs | Valid name, email, password, etc. | User account created successfully, verification email sent. | Pass |
| REG\_TC002 | Invalid email format | Invalid email address, valid password | Error message displayed: "Invalid email format." | Pass |
| REG\_TC003 | Password too short | Valid email, short password (< 6 characters) | Error message displayed: "Password must be at least 6 characters long." | Pass |
| REG\_TC004 | Email already exists | Existing email address, valid password | Error message displayed: "Email already exists." | Pass |
| REG\_TC005 | Confirm Password Field is functioning | Email and Password that isn't the same | Confirm password error message | Pass |
| REG\_TC006 | Test Phone No limit and type | Greater then 20 characters String type number | The application must enforce the character limit and data type, prompt user | Pass |

**Conclusion: The user registration process is functioning correctly, with all necessary validation checks in place.**

**Table 2: Test Report - Consultant Profile Creation and Management**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output | Result |
| PROF\_TC001 | Successful profile creation with valid data | Valid specialty, qualifications, availability, etc. | Consultant profile created successfully and stored in the database. | Pass |
| PROF\_TC002 | Upload and display qualifications, specialties, fees | Valid certifications, specialty, consulting fees | Certifications are uploaded and displayed correctly, fees are set. | Pass |
| PROF\_TC003 | Correct information displayed for different specialties | Different specialties selected | Profile displays information based on selected specialty. | Pass |
| PROF\_TC004 | Check Availability functionality to change the time | Proper log in creds to log in for the first time, access to profile, select time | To be able to select the required time and the new changes will commit | Pass |
| PROF\_TC005 | Consultant bio character | Put more then 200 characters | It would show an error because there is a character limit for the field | Pass |
| PROF\_TC006 | Ensure proper images are being uploaded | All required data but incorrect image format (png, doc, pdf) | Image must be proper and error to change format pops up | Pass |

**Conclusion: The consultant profile creation and management features are functioning correctly.**

**Table 3: Test Report - Appointment Scheduling and Cancellation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output | Result |
| APPT\_TC001 | Successful appointment scheduling with available consultant | Valid date, time, consultant ID | New booking record created, confirmation emails sent to both. | Pass |
| APPT\_TC002 | Confirmation emails sent | Scheduled appointment | Confirmation emails delivered to both patient and consultant. | Pass |
| APPT\_TC003 | Booking cancellation | Valid booking ID | Booking status updated to "canceled", cancellation emails sent. | Pass |
| APPT\_TC004 | Double booking prevention | Book appointment at the same time as someone else | The booking wont go through because already booked | Pass |
| APPT\_TC005 | If booking is in the past booking cannot be cancelled | Book appointment date in the past | The calendar should show that the booking cannot be canceled / only rejected | Pass |
| APPT\_TC006 | Bookings will require proper login to apply | Check the booking date or create another | Users that are not properly Logged in should not see the booking. | Pass |

**Conclusion: The scheduling and cancellation of appointment are functioning correctly.**

**Table 4: Test Report - Payment Processing and Security**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output | Result |
| PAY\_TC001 | Payment integrations functioning correctly (Stripe, PayPal) | Valid payment information | Successful transaction processing, payment status updated. | Not Pass |
| PAY\_TC002 | Credit card and sensitive data encryption | Credit card details | Credit card information is encrypted and securely stored. | Pass |
| PAY\_TC003 | Refund functionality work when rejecting | Request to refund | Correct refund and refund to the right account/booking data and users | Pass |
| PAY\_TC004 | Test data security for payments (database hack). | Attack the SQL | Data breach is prevented. User account information is hidden. | Pass |

**Conclusion: The secure payment system requires additional security checks before proceeding with full deployment.**

**Table 5: Test Report - Admin Access Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output | Result |
| ADMIN\_TC001 | Admin access to all data | Admin credentials | Access to all user, consultant, booking, payment data. | Pass |
| ADMIN\_TC002 | Regular user cannot access admin features | Regular user credentials | User is redirected to an unauthorized access page or their standard dashboard. | Pass |
| ADMIN\_TC003 | Admin user can't access user routes | User goes to doctor/user booking routes | The admin should not be able to access user and can only access through the admin panel | Pass |

**Conclusion: The Admin access control is working as intended.**

**6.2 User Documentation**

Clear and concise user documentation is essential for enabling users to effectively use the Health Consultant platform. The following sections provide user guides for different user roles:

**A. Patient User Guide:**

Registration: To create a new account, click on the 'Register' button and fill out the registration form. Provide your name, email address, password, and role (patient). Make sure all data are correct.

Logging In: To access your account, click on the 'Login' button and enter your registered email address and password.

Searching for Consultants: To find a suitable consultant, navigate to the 'Find a Consultant' page. Use the search bar to search by specialty, location, or expertise. You can also use the filters to refine your search results.

Viewing Consultant Profiles: To view a consultant's profile, click on their name or profile picture from the search results. The consultant's profile will display their qualifications, specialities, expertise, consultation fees, availability, and ratings/reviews.

Scheduling an Appointment: To schedule an appointment, click on the 'Book Appointment' button on the consultant's profile. Select an available date and time from the calendar interface. You may need to create a booking to chat.

Making a Payment: Proceed to the payment page to finalize the booking. The payment gateway supports credit card and debit card.

Managing Bookings: To view and manage your bookings, navigate to the 'My Appointments' page. Here, you can view booking details, reschedule appointments, or cancel appointments.

Uploading Health Records: To upload your health records, navigate to the 'Health Records' page. Upload your files (PDF, DOCX, JPG) to securely store your information.

Communicating with Consultants: To communicate with a consultant, navigate to the 'Messages' page. Select a past chat, or if a message is not available create a message request.

**B. Consultant User Guide:**

* + "Logging In: To access your consultant account, navigate to the login page from the website's homepage. Enter your registered email address in the 'Email' field and your secure password in the 'Password' field. After entering your credentials, click the 'Login' button. Upon successful authentication, you will be redirected to the Consultant Dashboard, where you can manage your profile, bookings, and availability."
  + "Setting Consulting Rates: As a consultant, you have the flexibility to set your consultation fees. To configure these fees, first access your dashboard. On the dashboard, locate the 'Settings' or 'Profile Configuration' section (this may be represented by an icon such as a gear or a profile image). Within the settings, find the option labeled 'Consulting Fees' or similar. Enter your desired consultation fee in the provided field. Ensure that you enter a valid numerical value (e.g., 50.00) and that you save your changes by clicking the 'Save' or 'Update' button. This fee will be displayed to patients when they book an appointment with you."
  + "Managing Bookings: The Consultant Dashboard provides a centralized location for managing all your scheduled appointments. To view your bookings, click on the 'Bookings' or 'Appointments' tab on the dashboard. This will display a list of all upcoming and past appointments. You can view the details of each booking, including the patient's name, contact information, appointment date and time, and consultation status. You can also use filters to sort and search for specific bookings."
  + "Managing Availability: Maintaining an accurate and up-to-date availability schedule is crucial for effective appointment scheduling. To manage your availability, navigate to the 'Availability' tab within the 'Settings' or 'Profile Configuration' section of your dashboard. On this page, you can specify your working hours for each day of the week. You can also mark specific days as 'unavailable' to indicate vacation periods or other times when you are unable to offer consultations. Remember to save your changes after updating your availability schedule to ensure that patients can only book appointments during your specified hours."
* **C. Administrator User Guide:**
  + "Logging In: To access the administrative functions of the Health Consultant platform, navigate to the login page and enter your designated administrator email address and password. Click the 'Login' button to authenticate your credentials. Upon successful login, you will be redirected to the Administrator Dashboard, which provides access to various management tools."
  + "Managing Users: The Administrator Dashboard provides a table with access to all registered users on the platform. To manage users, locate the 'User Management' section on the dashboard. This section will display a table with a list of all users, including their name, email address, role, and other relevant information. You can use the search bar to find specific users. From this section, you can view user profiles or update user information."
  + "Managing Consultants: The Administrator Dashboard also provides tools for managing consultants. To manage consultants, locate the 'Consultant Management' section on the dashboard. Here, you can view a list of all registered consultants, their profiles, and their approval status. You can approve or reject consultant registrations to control who can offer their services on the platform. You may also be able to edit consultant profiles or suspend accounts for policy violations."
  + "Managing Bookings: The Administrator Dashboard allows you to oversee all bookings made on the platform. To manage bookings, locate the 'Booking Management' section on the dashboard. In this section, you can view a list of all bookings, including the patient, consultant, date, time, and status. You can also use filters to search for specific bookings. If necessary, you can manually reschedule appointments, process cancellations, or issue refunds."

**7. CONCLUSION**

This section provides a summary of the project, its achievements, and the lessons learned. It also discusses the significance of the Health Consultant platform, its limitations, and potential avenues for future development and expansion.

**7.1 Conclusions**

The Health Consultant project successfully developed a functional, scalable, and secure online telehealth platform that connects patients with qualified healthcare professionals. The platform provides a comprehensive set of features, including user management, consultant profiles, booking system, payment processing, messaging, and health record management.

The project team successfully navigated various technical challenges, including implementing secure authentication and authorization, integrating with a payment gateway, and ensuring data privacy and security. The agile development methodology enabled the team to adapt to changing requirements, manage risks effectively, and deliver a high-quality product within the project timeline.

The test reports demonstrate that the platform meets the defined functional and non-functional requirements. The user documentation provides clear guidance for users on how to effectively use the platform.

Ultimately, the Health Consultant platform has the potential to significantly improve access to healthcare, enhance patient engagement, and reduce healthcare costs.

**7.2 Significance of the System**

The Health Consultant platform offers several significant benefits to patients, consultants, and the healthcare industry as a whole:

* + **Improved Access to Care:** The platform removes geographic and socioeconomic barriers to healthcare access, enabling patients to connect with qualified consultants regardless of their location or financial situation.
  + **Enhanced Patient Convenience:** The platform provides a convenient and flexible way for patients to receive healthcare consultations from the comfort of their homes. Patients can schedule appointments at times that fit their schedules and avoid the need to travel to traditional healthcare facilities.
  + **Reduced Healthcare Costs:** The platform can help reduce healthcare costs by eliminating travel expenses, lowering consultation fees, and preventing unnecessary hospital readmissions.
  + **Increased Patient Engagement:** The platform empowers patients to take control of their health journey by providing them with access to information, tools, and resources. Patients can actively participate in their care decisions and manage their health more effectively.
  + **Improved Healthcare Delivery Efficiency:** The platform streamlines the healthcare delivery process by automating administrative tasks, facilitating communication between patients and consultants, and enabling remote monitoring and management of chronic conditions.
  + **Provides increased business to doctors** By providing a platform to reach people in their homes, doctors have an easy way to get more customer bases.

**7.3 Limitations of the System**

Despite its many advantages, the Health Consultant platform also has certain limitations:

* + **Limited Scope of Services:** The platform currently focuses on providing remote consultations and health advice. It does not support more complex medical procedures or diagnostic tests that require in-person interaction.
  + **Dependence on Technology:** The platform relies on technology infrastructure, such as high-speed internet and reliable devices. Patients in areas with limited internet access may not be able to fully utilize the platform.
  + **Regulatory Challenges:** The telehealth industry is subject to evolving regulations and legal requirements. The platform must be adapted to comply with these regulations, which may vary by jurisdiction.
  + **Security Risks:** All online platforms are subject to security risks. The Health Consultant platform must continuously monitor and update its security measures to protect against emerging threats.
  + **Data Reliability:** There may be some issues with data and connectivity on the platform that depends on real-time network and connection, it is required the users maintain a stable network in order to not encounter any problems.

**7.4 Future Scope of the Project**

The Health Consultant platform has significant potential for future development and expansion. Some potential avenues for future development include:

* + **Integration with Wearable Devices:** Integrate the platform with wearable devices to collect real-time health data, such as heart rate, blood pressure, and activity levels. This data can be used to provide more personalized and proactive healthcare advice.
  + **AI-Powered Diagnostics:** Incorporate AI-powered diagnostics tools to assist consultants in making more accurate diagnoses. These tools can analyze patient data and provide insights that may not be readily apparent to human clinicians.
  + **Mobile Application Development:** Develop native mobile applications for iOS and Android devices to provide a more seamless and user-friendly experience for mobile users.
  + **Multilingual Support:** Add support for multiple languages to expand the platform's reach and accessibility to a global audience.
  + **Integration with EHR Systems:** Integrate the platform with existing Electronic Health Record (EHR) systems to enable seamless data exchange and improve coordination of care.
  + **Improve the User Interface:** Use modern UI and AI tools to provide the end users with a superior and easily understandable user experience.
  + **Improve Security:** Improve security by incorporating the latest trends and technology to prevent cyber-attacks.