

**Internship Project**

**HELATHCARE MANAGEMENT**

**SYSTEM**

**Submitted By**

**RAHAVAN G**

**TAMILSELVAN N**

# TABLE OF CONTENTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | | | **PAGE NO.** |
|  | **ABSTRACT** | | | **3** |
|  | **LIST OF FIGURES** | | | **4** |
|  | **LIST OF ABBREVIATION** | | | **5** |
| **1** | **INTRODUCTION** | | | **6** |
|  | 1.1 | OVERVIEW | | 6 |
|  | 1.2 | OBJECTIVE | | 8 |
| **2** | **PROPOSED SYSTEM** | | | **9** |
|  | 2.1 | FLOW CHART | | 9 |
|  | 2.2 | EXPLANATION | | 10 |
| **3** | **SOFTWARE DESCRIPTION** | | | **12** |
|  | 3.1 | FRONTEND | | 12 |
|  |  | 3.1.1 | HTML | 12 |
|  |  | 3.1.2 | CSS | 13 |
|  |  | 3.1.3 | JAVA SCRIPTS | 14 |
|  |  | 3.1.4 | REACT JS | 16 |
|  | 3.2 | BACKEND | | 17 |
|  |  | 4.2.1 | DJANGO(PYTHON) | 17 |
|  | 3.3 | DATABASE | | 19 |
|  |  | 3.3.1 | MY SQL | 19 |
| **4** | **OUTPUT** | | | **21** |
| **5** | **CONCLUSION** | | | **31** |

# ABSTRACT

This abstract introduces a sophisticated Healthcare Management System (HMS) designed to streamline and enhance healthcare delivery processes. The system features three distinct login panels catering to administrators, doctors, and patients, facilitating seamless interaction and information exchange among stakeholders.Administrative login grants access to comprehensive administrative functionalities, enabling efficient management of disease details, medication records, checkup schedules, appointment bookings, and feedback mechanisms. Through this panel, administrators can oversee the entire healthcare ecosystem, ensuring smooth operations and optimal resource allocation. Doctor login provides healthcare professionals with a robust platform to manage patient health records, including disease specifics, medication details, and checkup schedules. Additionally, it facilitates video call consultations, enabling remote patient care and enhancing accessibility to medical expertise. Patient login offers individuals a personalized portal to access their medical history, schedule appointments, and provide feedback on healthcare services. This interactive interface empowers patients to actively engage in their healthcare journey, fostering a sense of ownership and accountability. The system's key features include centralized data management, real-time communication channels, and user-friendly interfaces, ensuring efficiency, accuracy, and convenience in healthcare service delivery. By integrating these functionalities, the Healthcare Management System promises to revolutionize patient care by optimizing processes, improving access to healthcare resources, and enhancing overall healthcare outcomes

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE**  **NO.** | **DESCRIPTION** | **PAGE NO.** |
| 2.1 | Flow Chart | 9 |
| 4.1 | Home Page | 21 |
| 4.2 | Contact Page | 21 |
| 4.3 | Footer Section | 22 |
| 4.4 | Admin Dashboard | 22 |
| 4.5 | Add Doctor Page | 23 |
| 4.6 | Patient View and Edit Page | 23 |
| 4.7 | Doctor Login Page | 24 |
| 4.8 | Doctor Dashboard | 24 |
| 4.9 | Add Medical, Health Issue and Checkups Schedule Page | 25 |
| 4.10 | Patient Appointment page | 25 |
| 4.11 | Patient List page | 26 |
| 4.12 | Medical record Page | 26 |
| 4.13 | Patient Login Page | 27 |
| 4.14 | Patient Dashboard | 27 |
| 4.15 | Appointment Status Page | 28 |
| 4.16 | Chekup Schedule Page | 28 |
| 4.17 | Health Issue Page | 29 |
| 4.18 | Medication Detail page | 29 |
| 4.19 | History Records Page | 30 |
| 4.20 | Book Appointment Page | 30 |

# LIST OF ABBREVIATIONS

|  |  |  |
| --- | --- | --- |
| **CSRF** | - | Cross Site Request Forgery |
| **CSS** | - | Cascading Style Sheet |
| **DOM** | - | Document Object Model |
| **GPM** | - | Gallons per Minute |
| **GSM** | - | Global System for Mobile Communications |
| **GUI** | - | Graphical User Interface |
| **HMS** | **-** | Healthcare Management System |
| **HTML** | - | Hyper Text Markup Language |
| **JSX** | - | Java Script XML |
| **MVC** | - | Model View Controller |
| **MVT** | - | Model View Template |
| **ORM** | - | Object Relational Mapping |
| **RDMS** | - | Relational Database Management System |
| **SPA** | - | Single-Page Application |
| **SQL** | - | Structured Query Language |
| **UI** | - | User Interface |
| **URL** | - | Uniform Resource Locator |
| **XML** | - | Extensible Markup Language |
| **XSS** | - | Cross Site Scripting |

# CHAPTER 1

# INTRODUCTION

# OVERVIEW

# The advancement of technology has revolutionized every aspect of our lives, and healthcare is no exception. In today's fast-paced world, where accessibility, efficiency, and quality are paramount in healthcare delivery, the need for a comprehensive Healthcare Management System (HMS) has become increasingly evident. This introduction sets the stage for the discussion of an integrated HMS, designed to address the complexities of modern healthcare systems and enhance patient care outcomes.

# Healthcare institutions are faced with multifaceted challenges, including the management of vast amounts of patient data, coordination among various stakeholders, and the need for seamless communication channels. Traditional methods of healthcare administration often fall short in meeting these demands, resulting in inefficiencies, errors, and compromised patient care experiences.

# In response to these challenges, our project introduces a robust HMS tailored to the specific needs of healthcare organizations, practitioners, and patients alike. At its core, the system features three distinct login panels catering to administrators, doctors, and patients, each equipped with a suite of functionalities aimed at optimizing healthcare management processes.

# The administrative login panel serves as the nerve center of the system, empowering administrators with comprehensive tools for managing patient records, medication details, appointment scheduling, and feedback mechanisms. By centralizing data management and streamlining administrative workflows, this panel enables healthcare institutions to operate more efficiently and effectively.

# For healthcare professionals, the doctor login panel offers a dedicated platform for accessing patient health records, scheduling checkups, conducting video call consultations, and collaborating with colleagues. This functionality not only enhances the quality of care delivery but also promotes greater accessibility to medical expertise, particularly in remote or underserved areas.

# Patients, too, benefit from the system's patient login panel, which provides a user-friendly interface for accessing personal health records, scheduling appointments, and providing feedback on healthcare services. By empowering patients to take an active role in their healthcare journey, the system fosters a sense of engagement, accountability, and trust between patients and healthcare providers.

# Through the integration of these functionalities, our HMS promises to revolutionize healthcare management by promoting efficiency, accuracy, and accessibility across the entire healthcare ecosystem. By leveraging technology to overcome the challenges of traditional healthcare administration, we aim to improve patient care outcomes, enhance the patient experience, and ultimately, contribute to the advancement of healthcare delivery in the digital age.

# OBJECTIVE

# Develop a Healthcare Management System (HMS) with three distinct login panels: Administrative, Doctor, and Patient.

# Enable efficient management of disease details, medication records, checkup schedules, appointment bookings, and feedback mechanisms.

# Facilitate seamless communication and information exchange among administrators, healthcare professionals, and patients.

# Empower administrators with comprehensive tools for centralized data management and streamlined administrative workflows.

# Provide healthcare professionals with a dedicated platform for accessing patient health records, scheduling appointments, and conducting video call consultations.

# Offer patients a user-friendly interface for accessing personal health records, scheduling appointments, and providing feedback on healthcare services.

# Enhance efficiency, accuracy, and accessibility across the entire healthcare ecosystem through the integration of HMS functionalities.

# CHAPTER 2

# PROPOSED SYSTEM

# 2.1 FLOW CHART

# 

# 

# Figure 2.1 FLOW CHART

# 2.2 EXPLANATION

# Dashboard:

# The dashboard of the proposed Healthcare Management System (HMS) serves as a centralized platform where users can access various features and navigate through different sections of the system. It provides an intuitive interface that presents key information and functionalities in a clear and organized manner. The dashboard offers a snapshot of important metrics, such as appointment schedules, patient demographics, and system notifications, allowing users to quickly assess the status of healthcare operations.

# Login/Sign-in Credentials:

# To ensure secure access to the system and protect sensitive patient information, users are required to log in using their credentials. This authentication process helps maintain confidentiality and integrity, ensuring that only authorized personnel can access patient records and perform administrative tasks within the HMS.

# Administrative Login:

# The administrative login panel is specifically designed for administrators who oversee the overall operation of the healthcare facility. Administrators have access to a wide range of functionalities that enable them to efficiently manage doctor and patient account details. This includes tasks such as registering new doctors and patients, verifying credentials, updating contact information, and managing access permissions.

# Doctor Login:

# Healthcare professionals access the system through the doctor login panel, which provides them with specialized functionalities tailored to their roles. Doctors can leverage the HMS to conduct video call consultations with patients, allowing for remote diagnosis, treatment, and follow-up care. This feature enhances accessibility to medical expertise, particularly for patients in remote or underserved areas.

# In addition to video call consultancy, doctors can use the system to manage their appointments, view patient lists, and access comprehensive medical records. This includes detailed information about patients' medical history, current diagnoses, prescribed medications, and upcoming checkup schedules. By centralizing patient data and streamlining administrative tasks, the HMS enables doctors to deliver high-quality care more efficiently and effectively.

# Patient Login:

# Patients interact with the HMS through the patient login panel, which provides them with access to personalized healthcare services and information. Patients can view detailed information about their diseases, including diagnoses, treatment plans, and medication instructions. They can also access their medical records, which contain a comprehensive history of past appointments, treatments, and health assessments.

# The HMS facilitates appointment booking for patients, allowing them to schedule consultations with doctors at their convenience. Patients can also participate in video call consultations, enabling them to seek medical advice and follow-up care remotely.

# CHAPTER 3

# SOFTWARE DESCRIPTION

# 3.1 FRONTEND

# 3.1.1 HTML

# HTML (Hypertext Markup Language) is the backbone of web development, defining the structure and content of web pages.

* Structure Defined: The markup language used by HTML, which is made up of items contained in tags, defines the structure of a web page.
* The terms "elements" refer to different sections of a webpage, including lists (<ul>,<ol>,<li>), and headers (<h1> to <h6>)paragraphs (<p>), and divisions (<div>).
* Semantic Markup: HTML provides developers and browsers with semantic components that express the meaning of material. , , , , , and are a few examples of elements that support SEO and accessibility.
* Multimedia material, such as pictures (<img>), videos (<video>), and audio (<audio>), may be integrated into web pages using HTML. These components successfully communicate information and increase user engagement.
* Hyperlinking: HTML enables hyperlinking through the <a> (anchor) element, allowing users to navigate between web pages or sections within the same page. Attributes like href specify the destination URL, and target can open links in new tabs or windows.
* Forms: HTML facilitates the creation of interactive forms for user input, using elements like <form>, <input>, <textarea>, <select>, and <button>. Attributes such as type, name, and value define form fields and their properties.
* Metadata: HTML includes metadata for search engines and browser behavior. The <meta> element provides information like character encoding, viewport settings, and authorship.
* Document Type Declaration (DOCTYPE): HTML documents begin with a DOCTYPE declaration, which specifies the HTML version and document type.
* Cross-Platform Compatibility: HTML ensures compatibility across different browsers and devices, enabling consistent rendering and user experience

**3.1.2 CSS**

* Presentation Control: CSS defines the presentation and styling of HTML elements on a webpage, allowing developers to control aspects such as layout, color, typography, and spacing.
* Selectors and Declarations: CSS uses selectors to target HTML elements and declarations to apply styling rules. Selectors can be based on element types, classes, IDs, attributes, or hierarchical relationships.
* Cascading and Specificity: CSS rules cascade, meaning styles from different sources (e.g., external stylesheets, internal <style> elements, inline styles) combine to determine the final appearance of elements. Specificity determines which styles take precedence when multiple rules apply.
* Box Model: CSS employs the box model to represent the layout of elements, comprising content, padding, border, and margin. Developers can adjust these properties to create various designs and spacing arrangements.
* Flexbox and Grid Layout: CSS offers layout models like Flexbox and CSS Grid for creating responsive and flexible designs. These layout tools enable developers to organize content within containers and manage alignment and distribution.
* Media Queries: CSS supports media queries, allowing developers to apply different styles based on characteristics like screen size, device orientation, and resolution. Media queries are essential for building responsive websites that adapt to various viewing environments.
* Transitions and Animation: CSS enables the creation of smooth transitions and animations to enhance user interaction and engagement. Properties like transition and animation define the timing and effects of element changes.
* Vendor Prefixes: CSS includes vendor prefixes (e.g., -webkit-, -moz-, -ms-) to ensure compatibility with different browsers and experimental features. While browser support improves, vendor prefixes help maintain consistency across platforms.
* Preprocessors: CSS preprocessors like Sass and Less extend the capabilities of CSS by introducing features like variables, nesting, mixins, and functions. Preprocessors enhance code maintainability and allow for more efficient stylesheet authoring.
* Compatibility and Performance: CSS ensures compatibility across browsers and devices, optimizing performance by minimizing file size, reducing rendering time, and utilizing modern CSS features responsibly.

# 3.1.3 JAVA SCRIPTS

# Dynamic Behavior: JavaScript is a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages. It enables developers to create interactive elements, respond to user actions, and manipulate content in real-time.

# Client-Side Scripting: JavaScript runs on the client side, meaning it executes in the user's web browser. This allows for instant feedback and interaction without needing to communicate with the server, enhancing the user experience.

# Event Handling: JavaScript enables event-driven programming, where functions are triggered in response to user actions such as clicks, mouse movements, key presses, and form submissions. Event listeners are used to detect and respond to these events.

# DOM Manipulation: JavaScript interacts with the Document Object Model (DOM), which represents the structure of an HTML document. Developers can manipulate DOM elements, attributes, and styles dynamically, enabling changes to the content and layout of web pages.

# Asynchronous Programming: JavaScript supports asynchronous programming paradigms, allowing tasks to execute independently of the main program flow. Asynchronous operations, such as fetching data from a server or performing animations, enhance responsiveness and efficiency.

# Data Manipulation and Validation: JavaScript facilitates data manipulation and validation on the client side, reducing the need for server requests and improving performance. It can validate user input, format data, and perform calculations within the browser.

# Browser Compatibility: JavaScript code is executed differently across various web browsers. Developers use techniques like feature detection and polyfills to ensure compatibility and consistent behavior across different browser environments.

# Libraries and Frameworks: JavaScript has a rich ecosystem of libraries and frameworks, such as jQuery, React, Angular, and Vue.js, which streamline development tasks, provide reusable components, and simplify complex interactions.

# Error Handling and Debugging: JavaScript includes mechanisms for error handling and debugging, such as try-catch blocks, console logging, and browser developer tools. These tools aid in identifying and resolving issues during development.

# 3.2.4 REACT JS

# Component-Based Architecture: React.js is a JavaScript library for building user interfaces (UIs) with a component-based architecture. Components are reusable, self-contained units that encapsulate UI elements and behavior, promoting code reusability and maintainability.

# Virtual DOM: React utilizes a virtual DOM (Document Object Model) to efficiently update the UI. Instead of directly manipulating the browser's DOM, React creates a virtual representation of the DOM in memory, compares it with the previous state, and applies only the necessary updates to the actual DOM, minimizing performance overhead.

# Declarative Syntax: React uses a declarative syntax to describe the desired UI state based on data changes. Developers define how the UI should look at any given time, and React handles the underlying DOM updates automatically, simplifying UI development and reducing boilerplate code.

# JSX (JavaScript XML): React introduces JSX, a syntax extension that allows developers to write HTML-like code directly within JavaScript. JSX enables the seamless integration of UI components and logic, improving code readability and maintainability.

# Unidirectional Data Flow: React follows a unidirectional data flow pattern, where data flows down from parent components to child components via props (properties). This ensures predictable data flow and facilitates debugging and testing.

# Component Lifecycle Methods: React components have lifecycle methods that allow developers to hook into different stages of a component's lifecycle, such as initialization, mounting, updating, and unmounting. Lifecycle methods enable developers to perform actions like data fetching, state updates, and cleanup operations.

# State Management: React components can manage internal state using the useState hook (for functional components) or the setState method (for class components). State enables components to maintain and update their data independently, leading to more dynamic and interactive UIs.

# Component Reusability: React promotes component reusability through composition, inheritance, and props. Components can be composed together to build complex UIs, and props allow for customization and configuration of component behavior.

# React Router: React Router is a popular library for handling navigation and routing in React applications. It enables developers to define routes, navigate between different views, and handle URL parameters, enhancing the user experience and enabling single-page application (SPA) development.

# Community and Ecosystem: React has a large and active community, with a rich ecosystem of libraries, tools, and resources. Developers benefit from a wide range of third-party packages, tutorials, and community support, accelerating development and fostering innovation.

# 3.2 BACKEND

# 3.2.1 DJANGO

# Full-Stack Web Framework: Django is a high-level Python web framework that enables rapid development of web applications with a clean and pragmatic design. It follows the Model-View-Controller (MVC) architectural pattern, emphasizing reusability, modularity, and scalability.

# Model-View-Template (MVT) Architecture: Django employs the Model-View-Template (MVT) architecture, where models represent the data structure, views handle request processing and response rendering, and templates define the presentation layer.

# Object-Relational Mapping (ORM): Django provides a built-in ORM that abstracts database interactions, allowing developers to define database models using Python classes. The ORM handles database queries, migrations, and transactions, facilitating database access and management without writing raw SQL queries.

# Admin Interface: Django includes a powerful admin interface for managing site content and user data. Administrators can perform CRUD (Create, Read, Update, Delete) operations on database records through a user-friendly interface generated automatically from model definitions.

# URL Routing: Django uses a URL routing mechanism to map URL patterns to view functions or class-based views. Developers define URL patterns in the URLconf (URL configuration) module, specifying the corresponding views to handle incoming requests.

# Template Engine: Django's template engine allows developers to create dynamic HTML templates using Django's template language, which includes template tags, filters, and template inheritance. Templates enable the presentation layer to render data dynamically and maintain consistency across web pages.

# Authentication and Authorization: Django includes robust authentication and authorization mechanisms for user authentication, permissions, and session management. Developers can integrate user authentication with minimal configuration, including support for user registration, login, logout, and password reset functionalities.

# Security Features: Django incorporates security features to protect web applications against common vulnerabilities, such as cross-site scripting (XSS), cross-site request forgery (CSRF), SQL injection, and clickjacking. It provides built-in middleware and utilities for implementing security best practices.

# 3.3 DATABASE

# 3.3.1 MY SQL

# Relational Database Management System (RDBMS): MySQL is a widely-used open-source relational database management system that stores and manages structured data. It follows the relational model, organizing data into tables consisting of rows and columns, with relationships defined between them.

# Structured Query Language (SQL): MySQL uses SQL, a powerful language for querying and manipulating relational databases. SQL enables developers to perform various operations, including data retrieval, insertion, updating, deletion, and schema modification.

# Data Types: MySQL supports a wide range of data types to represent different kinds of data, such as integers, floating-point numbers, strings, dates, and times. Data types ensure data integrity and optimize storage and processing efficiency.

# Table Management: MySQL enables the creation, modification, and deletion of tables through SQL commands or graphical user interfaces (GUIs) like MySQL Workbench. Tables define the structure of the database, including column names, data types, constraints, and indexes.

# Indexes and Constraints: MySQL allows developers to define indexes and constraints on tables to improve query performance and enforce data integrity. Indexes speed up data retrieval by facilitating quick lookup of records, while constraints ensure data consistency and enforce rules like uniqueness and referential integrity.

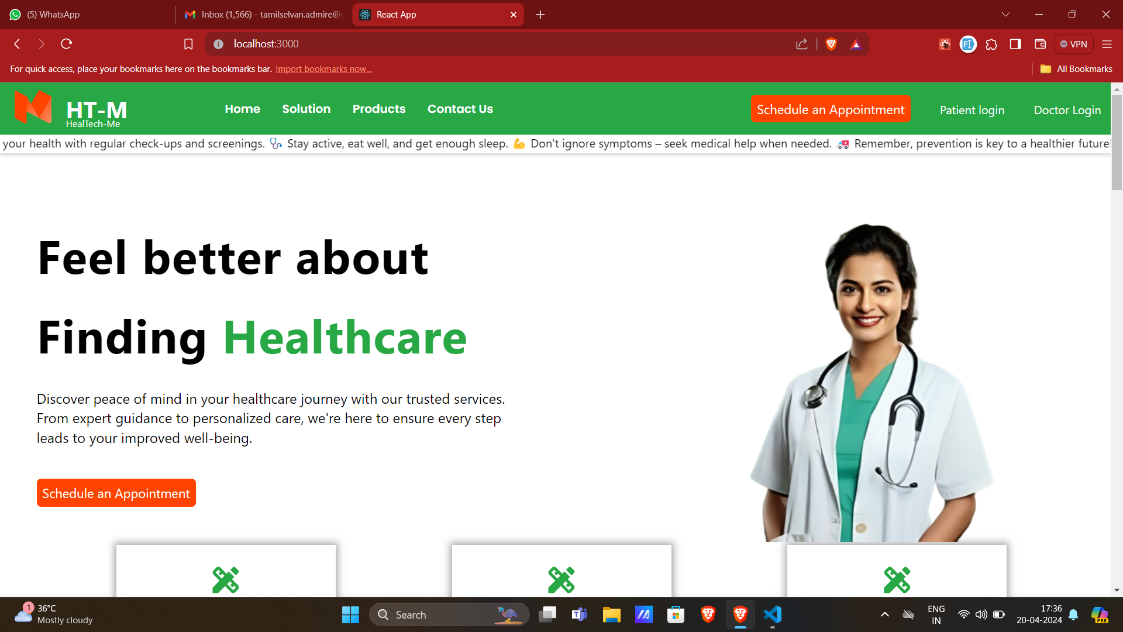
# Stored Procedures and Functions: MySQL enables the creation of stored procedures and functions, which are reusable blocks of SQL code stored in the database. Stored procedures and functions enhance database functionality, encapsulate business logic, and improve performance by reducing network overhead.

# Security Features: MySQL provides robust security features to protect sensitive data and ensure authorized access. It supports user authentication, access control, encryption, and auditing mechanisms to safeguard databases from unauthorized access, SQL injection attacks, and data breaches.

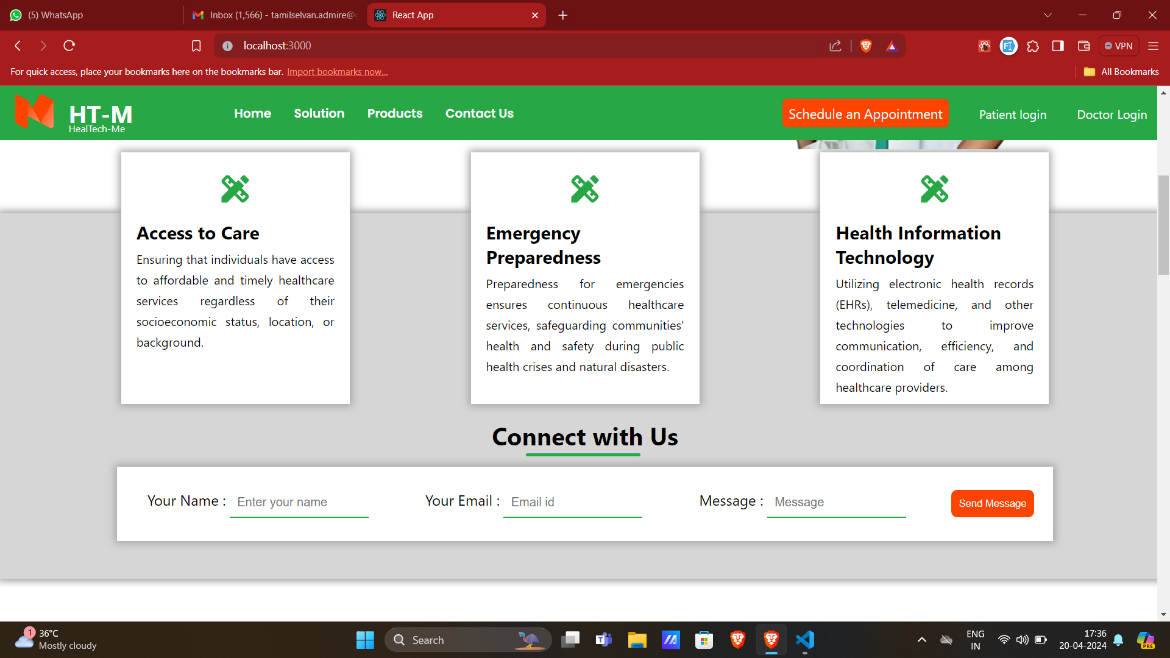
# Replication and High Availability: MySQL offers replication functionality to create copies of databases for redundancy, load balancing, and disaster recovery purposes. Replication ensures data consistency across multiple servers and enhances availability and scalability.

**CHAPTER 4**

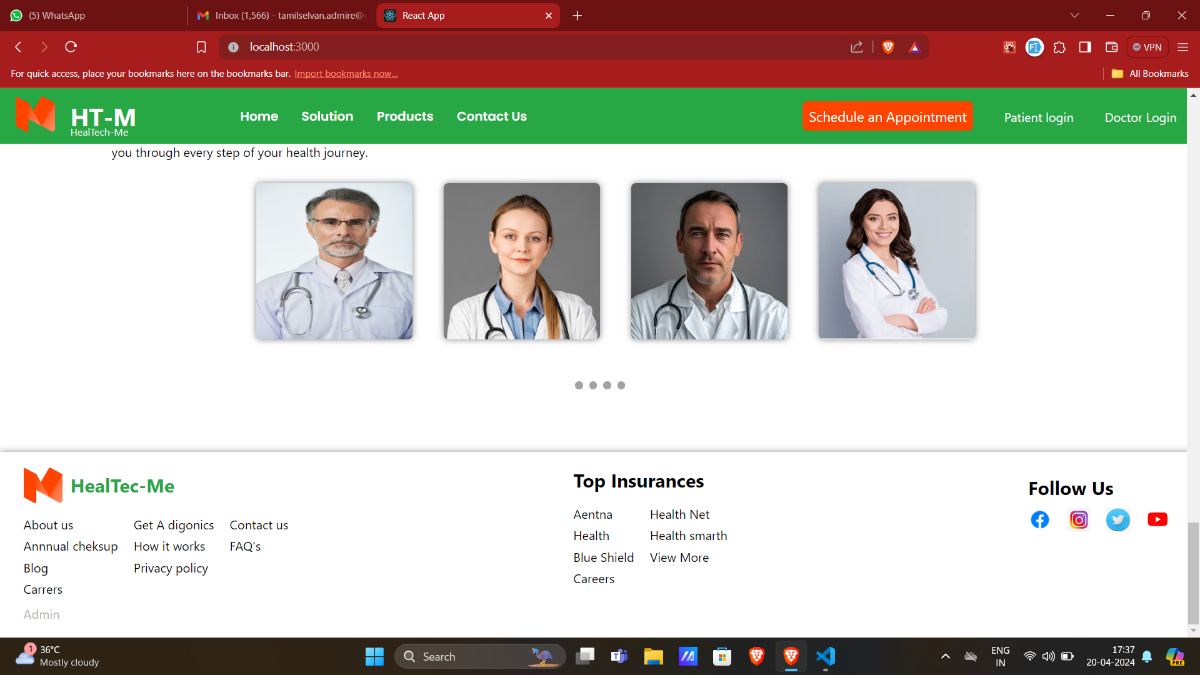
**OUTPUT**

****

**Figure 4.1 Home Page**

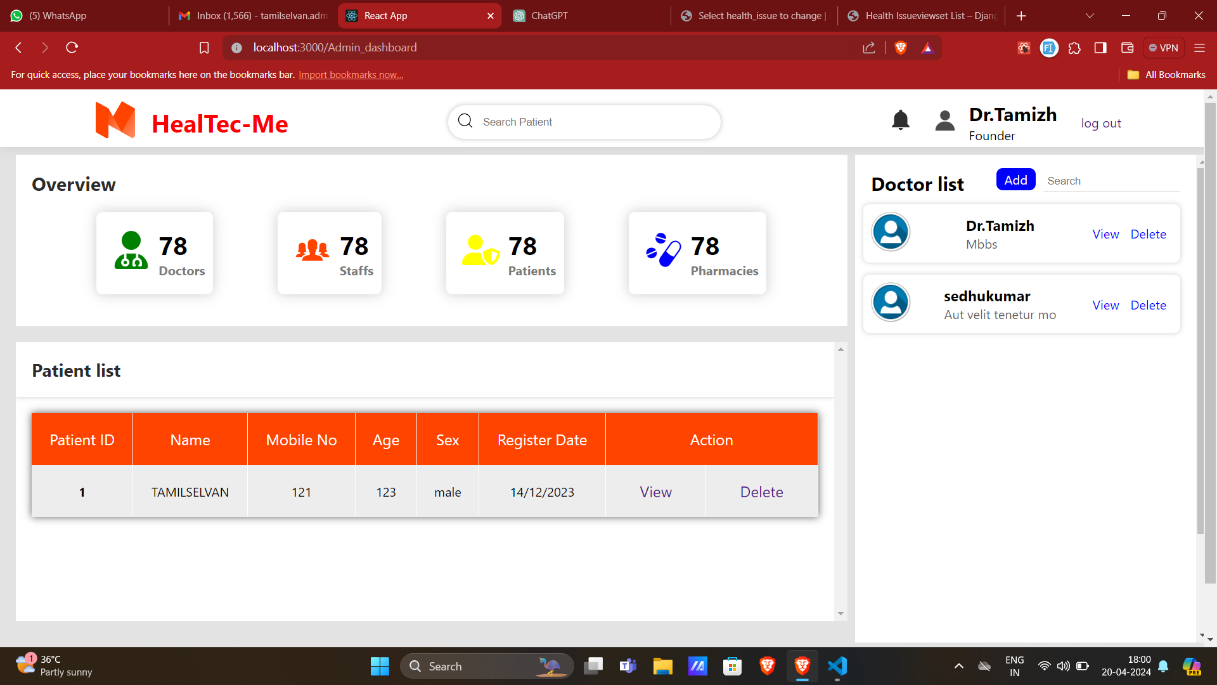
****

**Figure 4.2 Contact Page**

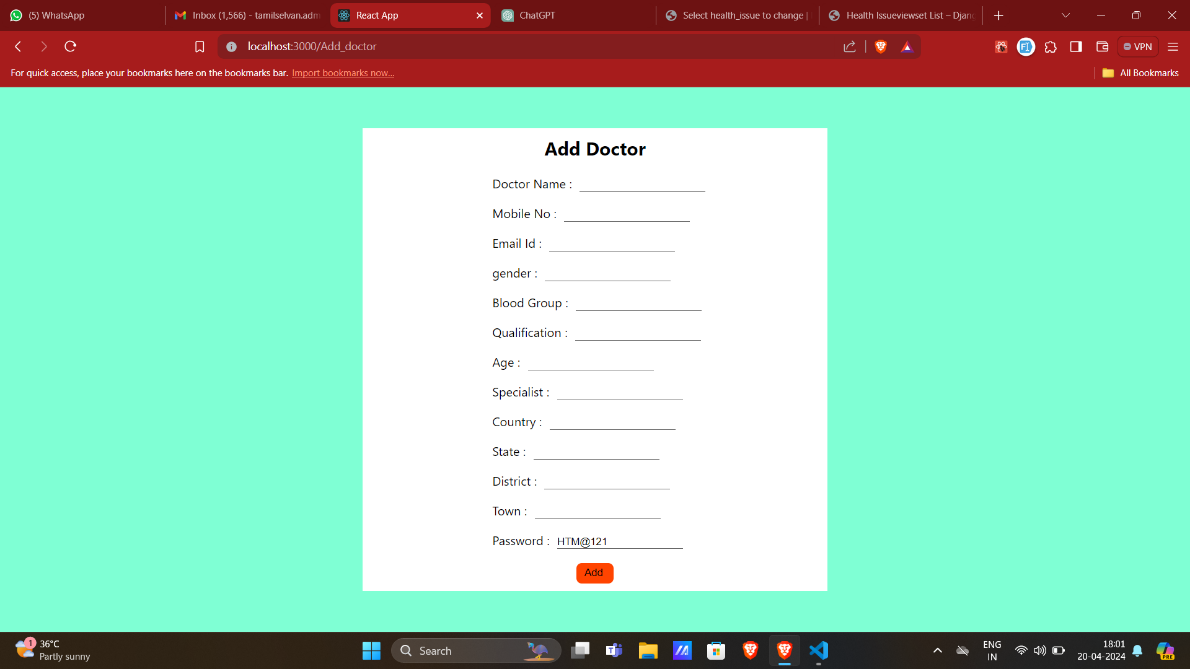
****

Admin login

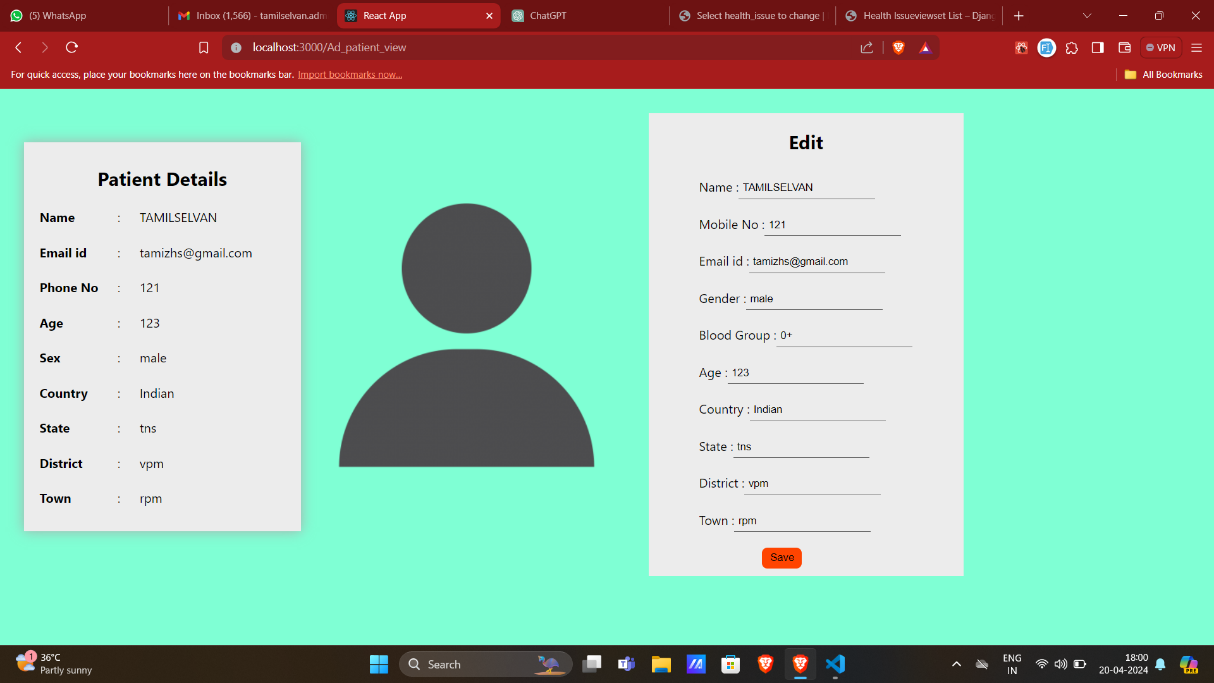
**Figure 4.3 Footer Section**

****

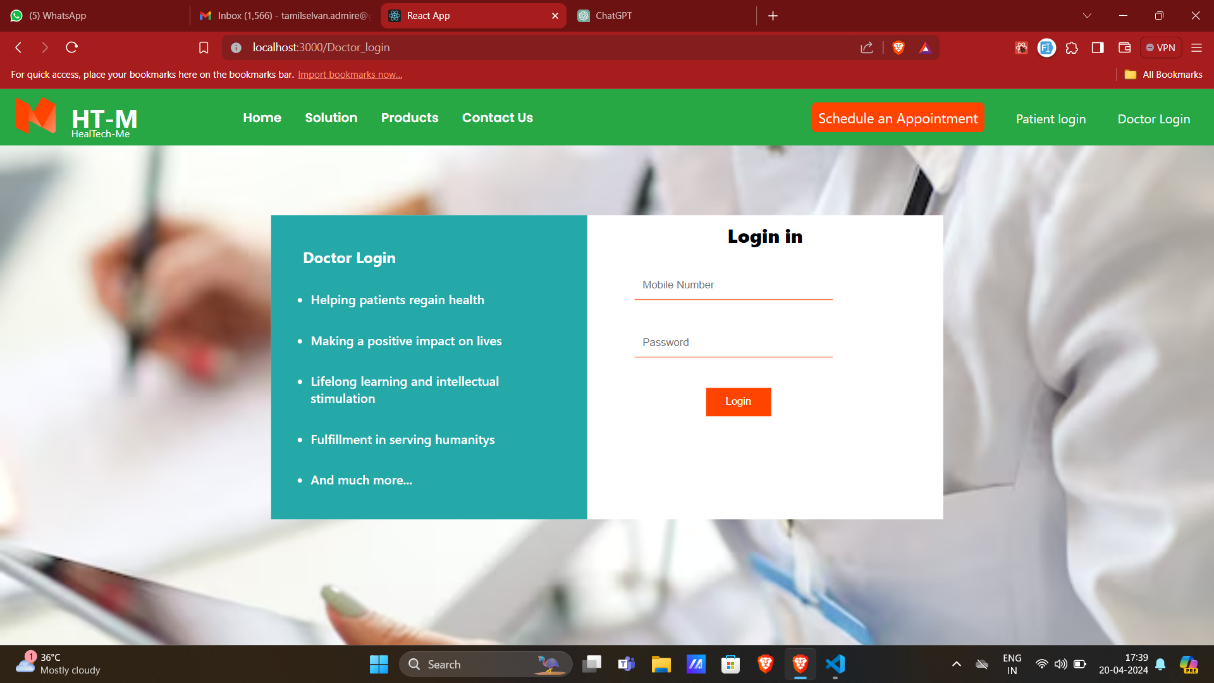
**Figure 4.4 Admin Dashboard**

****

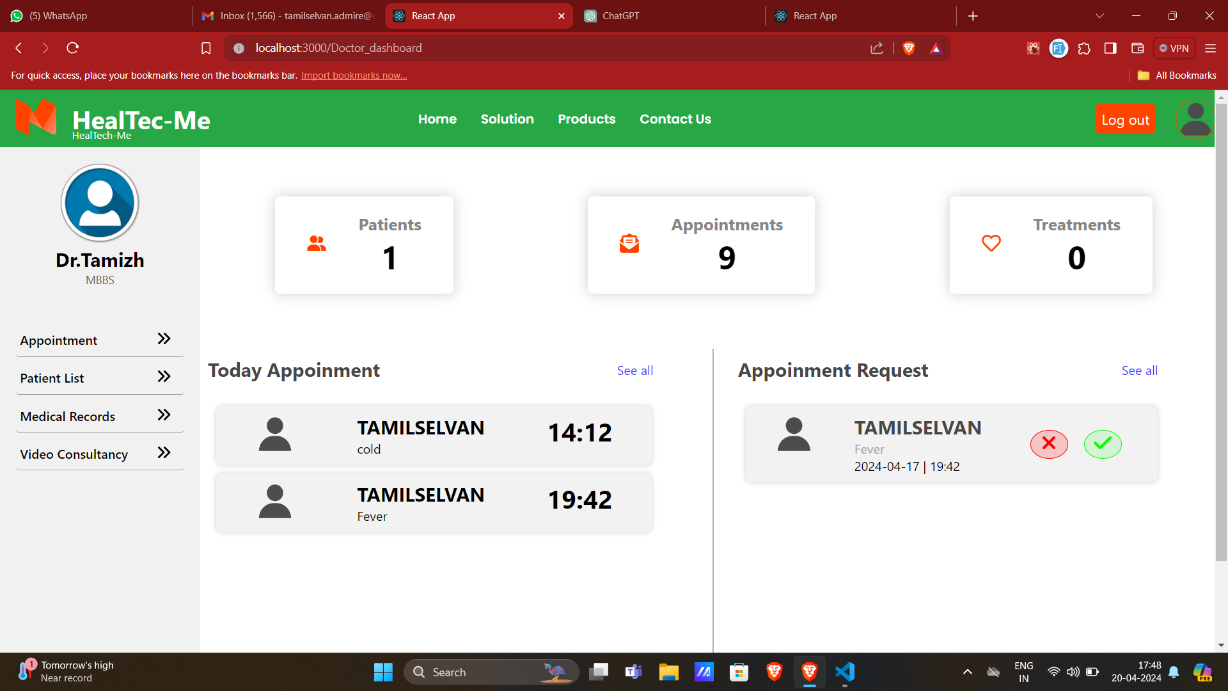
**Figure 4.5 Add Doctor Page**

****

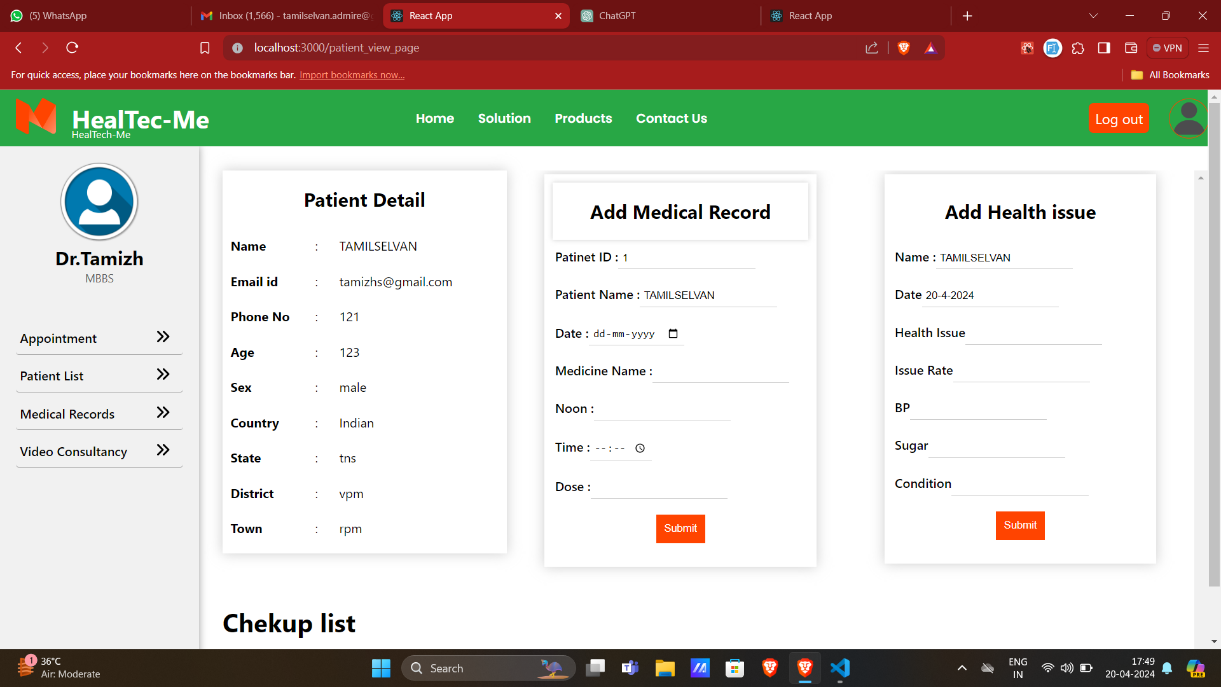
**Figure 4.6 Patient View and Edit Page**

****

**Figure 4.7 Doctor Login Page**

****

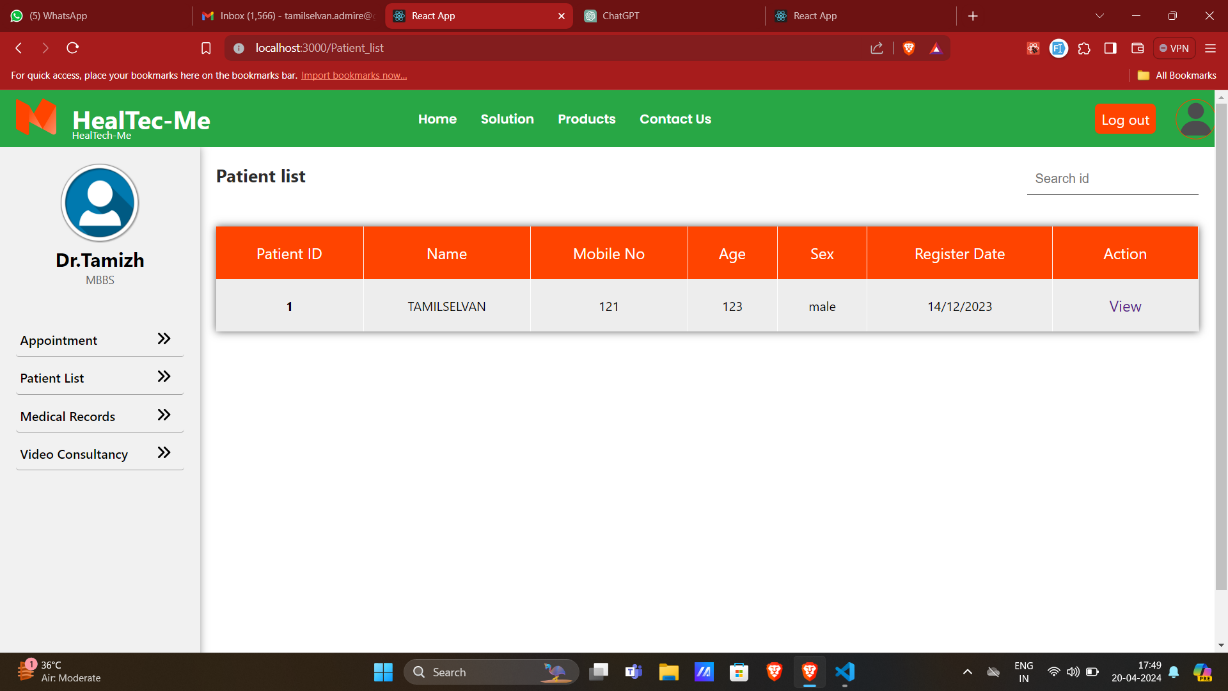
**Figure 4.8 Doctor Dashboard**

****

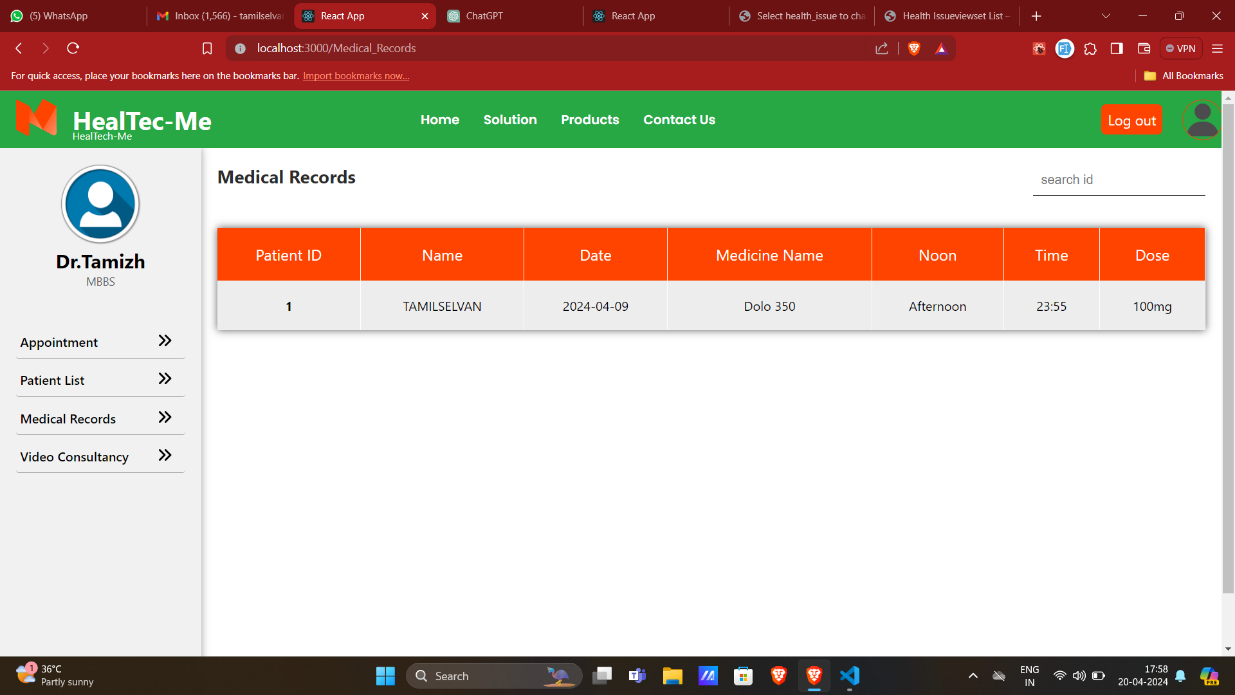
**Figure 4.9 Add Medical ,Health Issue And Cheup Schedule Page**

****

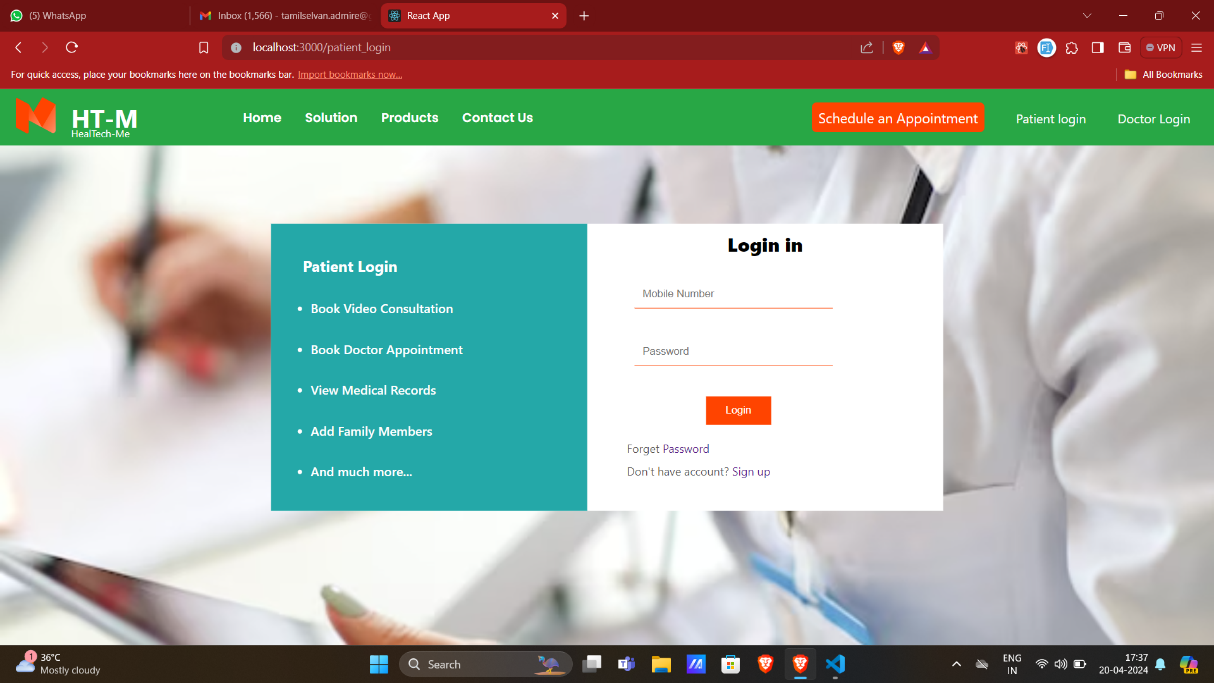
**Figure 4.10 Patient Appointment page**

****

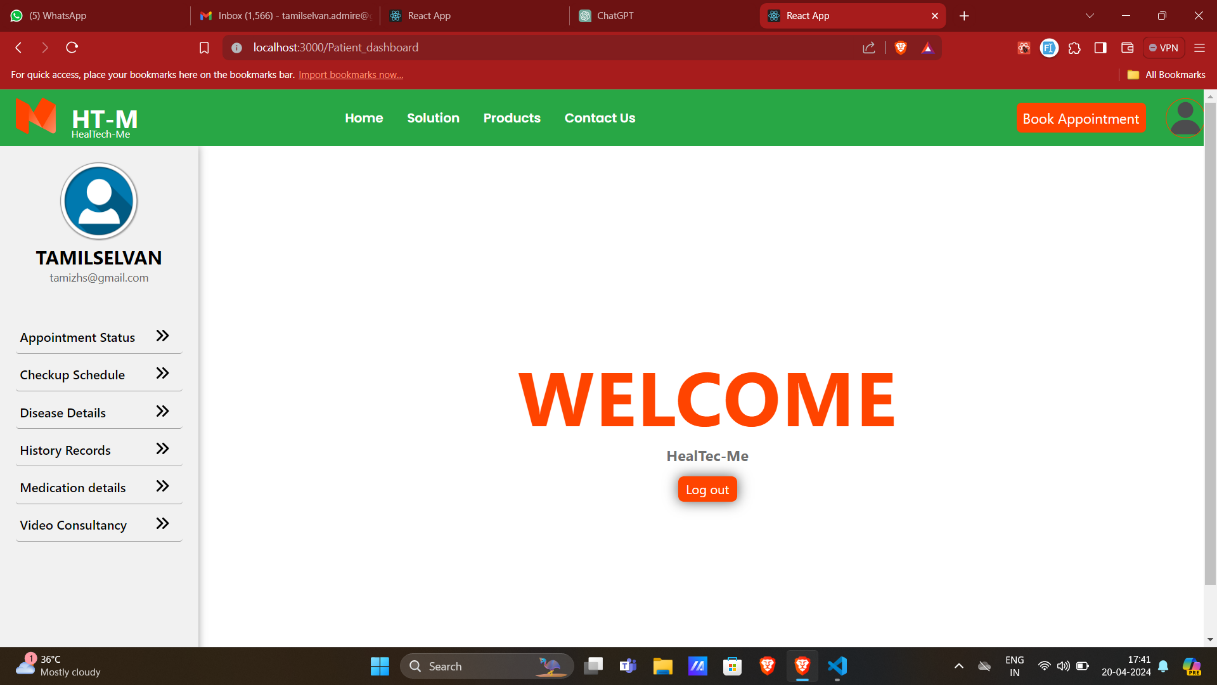
**Figure 4.11 Patient List Page**

****

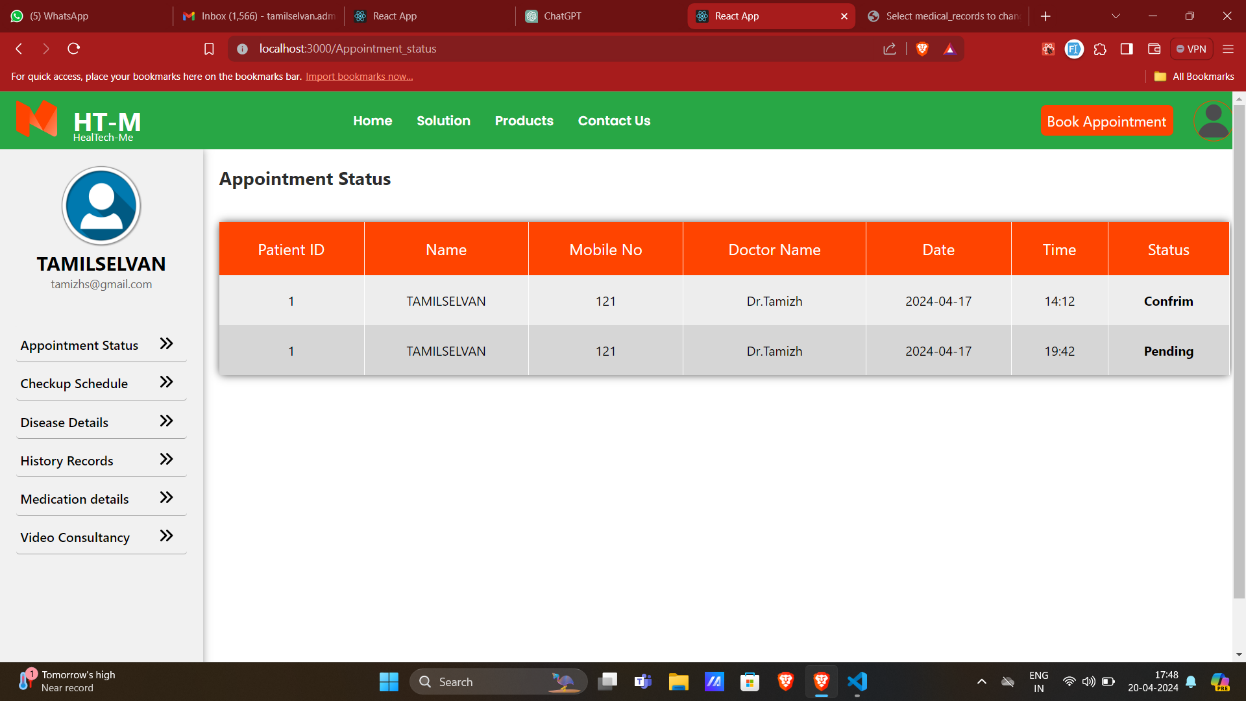
**Figure 4.12 Medical Record Page**

****

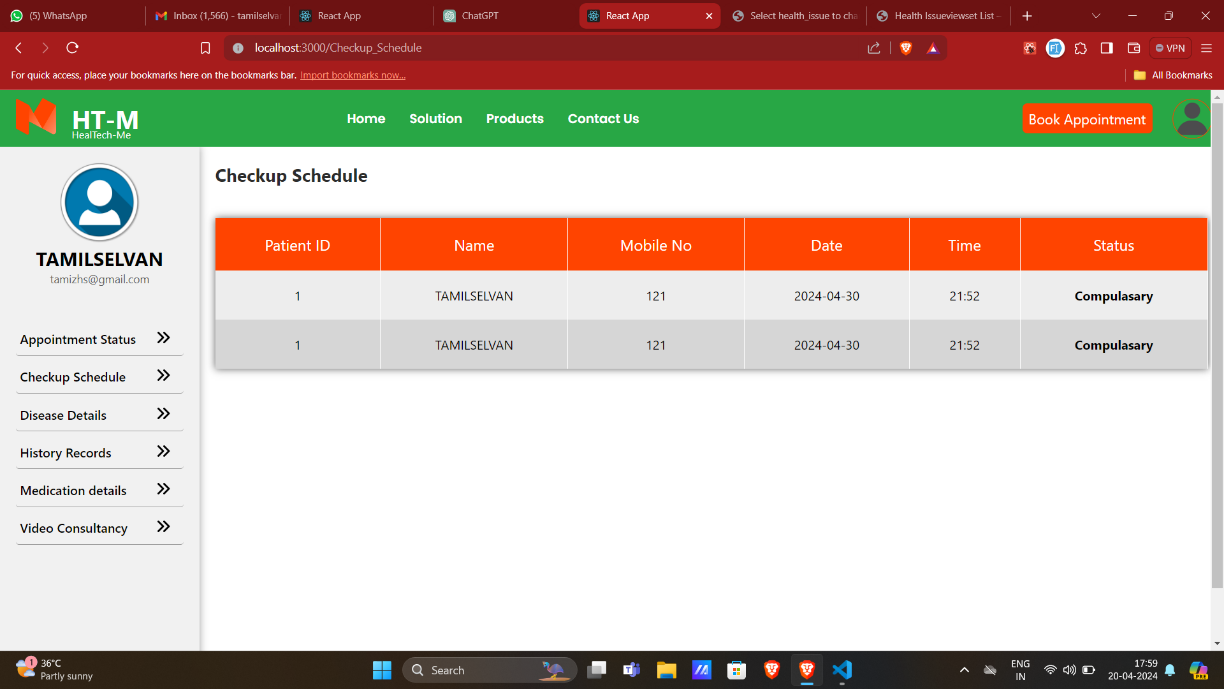
**Figure 4.13 Patient login Page**

****

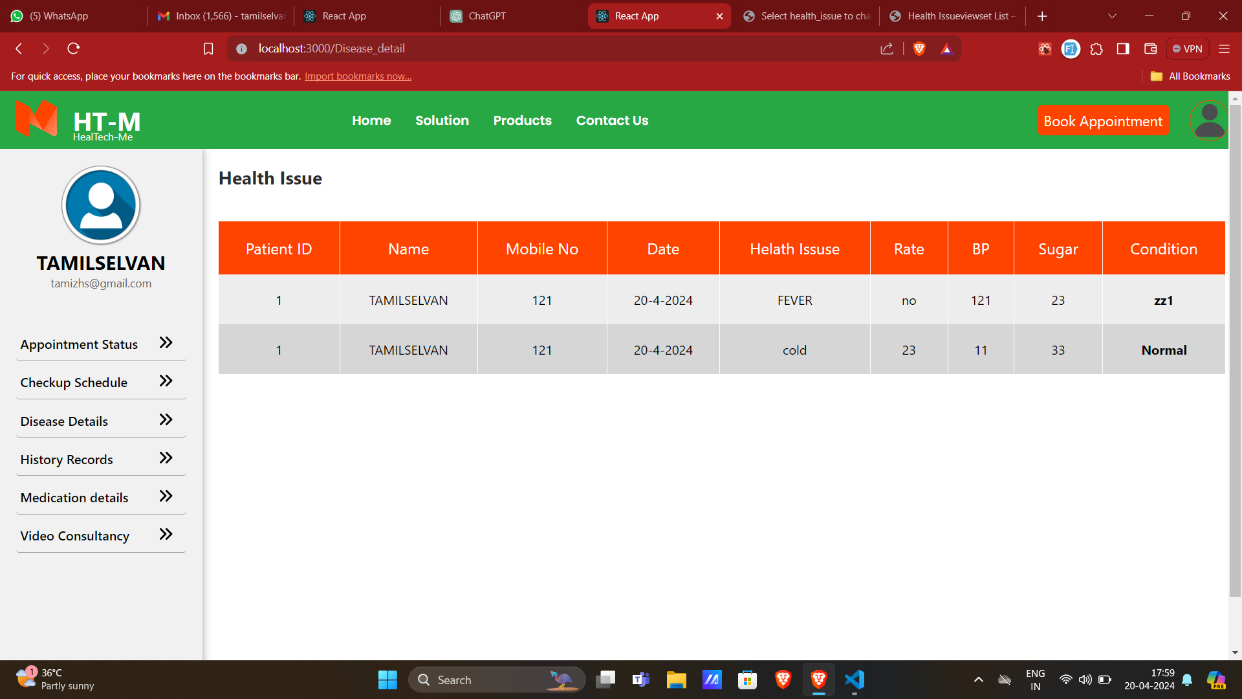
**Figure 4.14 Patient Dashboard**

****

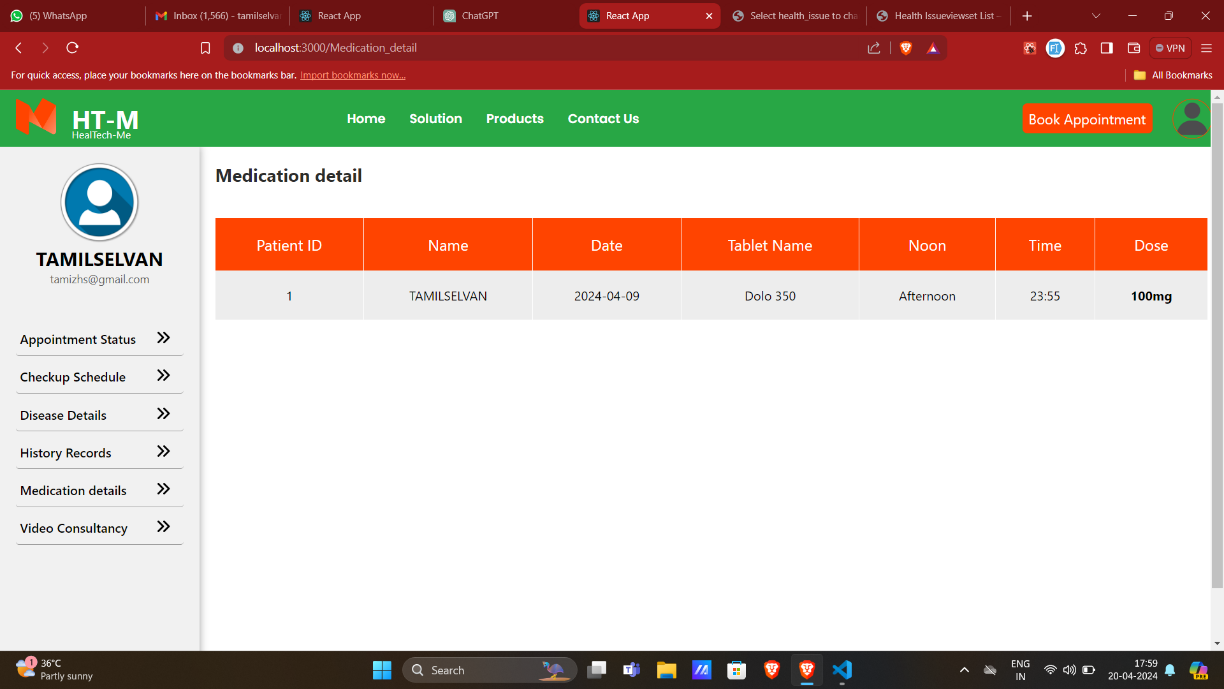
**Figure 4.15 Appointment Status Page**

****

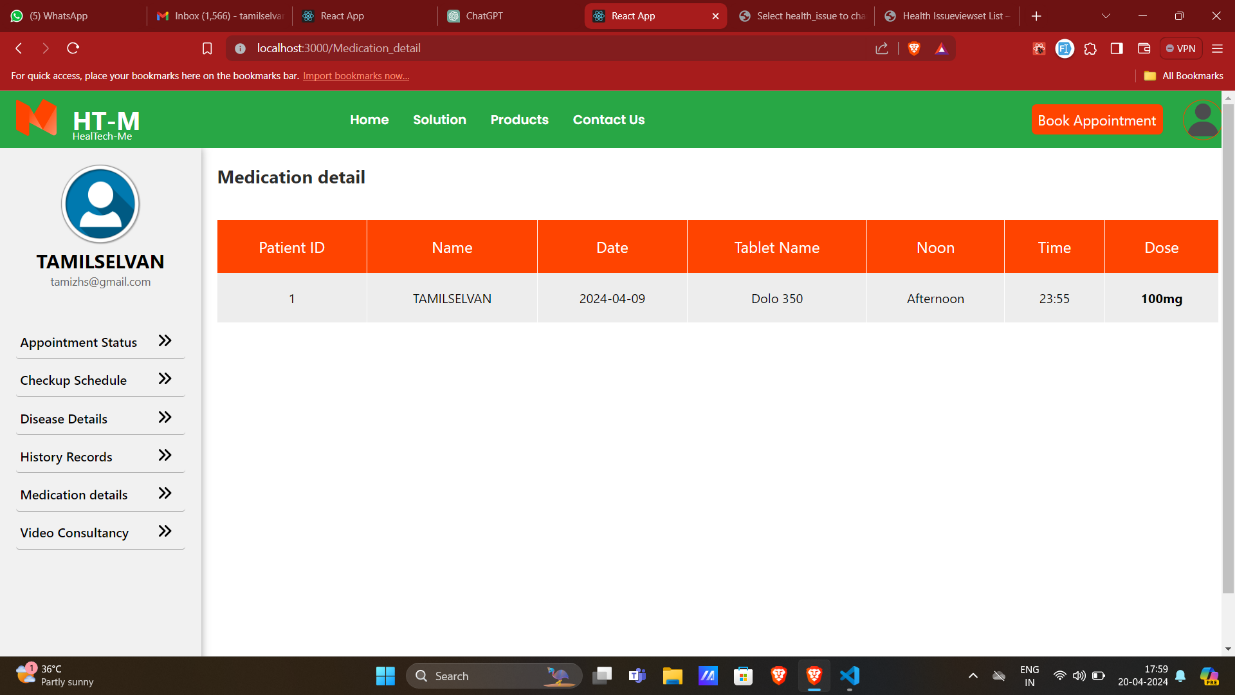
**Figure 4.16 Checkup’s Schedule Page**

****

**Figure 4.17 Health Issue Page**

****

**Figure 4.18 Medication Detail Page**

****

**Figure 4.19 History Records Page**

****

**Figure 4.20 Book Appointment Page**

**CHAPTER 5**

**CONCLUSION**

The Healthcare Management System (HMS) presented in this project represents a significant advancement in healthcare technology aimed at enhancing patient care outcomes and improving operational efficiency within healthcare facilities. Through its integrated features and user-friendly interfaces, the HMS addresses the complexities and challenges faced by healthcare organizations, practitioners, and patients in the modern healthcare landscape. By providing dedicated login panels for administrators, doctors, and patients, the system ensures tailored access to relevant functionalities, fostering seamless communication and collaboration among stakeholders. Administrators benefit from streamlined administrative processes, enabling efficient management of healthcare resources and compliance with regulatory standards. Doctors gain access to comprehensive patient information and tools for remote consultation, facilitating personalized care delivery and improving patient outcomes. Patients, in turn, enjoy enhanced access to healthcare services, with features such as appointment scheduling, medication management, and remote consultations, empowering them to take an active role in their healthcare journey. Through the implementation of the HMS, healthcare facilities can expect to see improvements in patient satisfaction, operational efficiency, and overall healthcare quality. By leveraging technology to optimize healthcare processes and facilitate patient-centered care, the system paves the way for a more effective and accessible healthcare delivery system.

**THANK YOU**