**HOSPITAL MANAGEMENT SYSTEM**

# **A PROJECT REPORT**

***Submitted by***

**KAVYA R (920422205053)**

**ABINAYA SELVI S (920422205004)**

**DEEPIKA M (920422205024)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECNOLOGY**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**KAMARAJ COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(An Autonomous Institution - Affiliated to Anna University, Chennai)**

**K.VELLAKULAM, VIRUDHUNAGAR - 625 701**

**NOVEMBER 2024**

**KAMARAJ COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(An Autonomous Institution- Affiliated to Anna University, Chennai)**

**K.VELLAKULAM, VIRUDHUNAGAR - 625 701**

**BONAFIDE CERTIFICATE**

Certified that the project report “**HOSPITAL MANAGEMENT SYSTEM** **USING MERN STACK”** is the bonafide work**of “KAVYA R (920422205053),ABINAYA SELVI S (920422205004),DEEPIKA M (920422205024)”** who carried out the project work under my supervision.

# **Dr. E. VAKAIMALAR** **Dr. R. ARTHY**

**Head of the Department,** **SUPERVISOR,**

Associate Professor, Assistant Professor,

Dept. of Information Technology, Dept. of Information Technology,

Kamaraj College of Engg & Tech, Kamaraj College of Engg &Tech, K.Vellakulam, K.Vellakulam,

Virudhunagar - 625 701. Virudhunagar - 625701.

# **INTERNAL EXAMINER** **EXTERNAL EXAMINER**

**ABSTRACT:**

This project is a full-stack web application built using the MERN (MongoDB, Express, React, Node.js) stack, designed to streamline user registration, login, and appointment management. The application provides a seamless workflow for users to register, log in, and manage appointments, offering an intuitive and efficient interface.

Upon accessing the home page, users have the option to either register as new users or log in if they already have an account. The registration process securely stores user details in a MongoDB database under the "register" collection. Once registered, users are redirected to the login page, where their credentials are validated using stored registration details. Successful login redirects users to the appointment page, where they can view and manage their appointments.

On the appointments page, users can add new appointments, which are stored in the MongoDB database under the "appointments" collection. The front end dynamically updates to display newly added appointments. The app emphasizes simplicity, avoiding complex authentication systems like bcrypt or JWT, and focuses instead on ensuring smooth functionality for managing user information and appointments.

This web application can be extended or modified to suit various business use cases, making it a versatile solution for any service requiring appointment management with a straightforward user experience.

**ACKNOWLEDGEMENT:**

I would like to express my sincere gratitude to Dr. E. Vakaimalar, HOD of the Department of Information Technology, and Dr. R. Arthy, my supervisor, for their invaluable guidance and support throughout the development Culinary Tales-RecipeBook project using the MEAN stack. Their insights and expertise were instrumental in helping me understand the practical implementation of web technologies, enhancing both my technical skills and problem-solving abilities. This project has provided me with hands-on experience in full-stack development, which will be immensely helpful in future endeavors.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | [**ABSTRACT**](bookmark://_Abstract)  **LIST OF FIGURES** | iii  vi |
| **1** | 1.1 HTML  1.2 CSS  1.3 JavaScript  1.4 MEAN Stack | **1**  1  2  3  4 |
| **2** | 2.1 Objective  2.2 Problem Statement  2.3 Block Diagram  2.4 Module Explanation | **7**  7  7  7  **8** |
| **3** | **RESULTS AND DISCUSSION** | **11** |
| **4** |  | **16** |
| **5** |  | **17** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 2.1 | Block Diagram | 7 |
| 3.1 | Home page | 12 |
| 3.2 | Products page | 12 |
| 3.3 | Cart page | 13 |
| 3.4 | About page | 13 |
| 3.5 | Getting details and placing order | 14 |
| 3.6 | Order Confirmation | 14 |
| 3.7 | Storing data in MongoDB | 15 |

**INTRODUCTION:**

**HTML:**

HTML is the foundational markup language used to create and structure content on the web. In this project, HTML is used to structure the various pages of the application, including the registration, login, home, and appointment pages.

**CSS:**

CSS is used for styling and layout of the application. It ensures that each page in the application, from registration to appointments, is visually appealing and responsive across different devices.

**JAVASCRIPT:**

JavaScript is a high-level programming language used in both the frontend and backend of this project. It enables dynamic interactivity on the client side (React.js) and powers server-side functionalities using Node.js and Express.js.

### **MERN STACK :**

The MERN stack is a full-stack JavaScript solution consisting of four main technologies: MongoDB, Express.js, React.js, and Node.js. These technologies work together to deliver a seamless and efficient development environment. Here's a detailed look at each component:

1. **MongoDB**:

* A NoSQL database that stores data in a flexible JSON-like format (BSON),allowing for dynamic and scalable storage of user information and appointment data.

1. **Express.js**:
   * A lightweight backend framework for building web applications in Node.js.
   * It provides essential features such as routing, middleware, and interaction with the MongoDB database.
2. **Angular**:

* A front-end library used for building dynamic and interactive user interfaces.
* React components are utilized in the creation of the registration, login, and appointment pages.

1. **Node.js**:

* A server-side runtime environment that executes JavaScript on the server.
* . It powers the backend of the application and facilitates interaction between the frontend and the database.

The MERN stack enables an end-to-end JavaScript development experience, making the stack highly efficient and cohesive for modern web applications.

**MongoDB (Database Layer)**

MongoDB is a NoSQL, document-oriented database that stores data in a flexible, JSON-like format. Unlike traditional relational databases, MongoDB does not rely on tables and schemas. Instead, it uses collections and documents, allowing for easier handling of large volumes of unstructured or semi-structured data.

**Key features of MongoDB include**:

* **Schema-less Structure**: Offers flexibility in data storage without a fixed schema, allowing developers to store varying types of data in the same collection.
* **Scalability**: Supports horizontal scaling, making it suitable for applications with growing datasets.
* **JSON Data Representation**: Stores data in BSON, seamlessly integrating with front-end JavaScript applications.

For example, an e-commerce website using MongoDB could efficiently store user data, product catalogs, and order details, evolving over time without needing extensive database modifications.

**Express.js (Backend Layer)**

Express.js is a lightweight web application framework that runs on Node.js. It simplifies the process of building server-side applications by providing robust features for managing routing, middleware, and HTTP requests.

**Key features of Express.js include:**

* **Routing**: Simplifies routing of HTTP requests to specific application endpoints. Developers can define routes for various functionalities, such as user login and data retrieval.
* **Middleware Support**: Middleware functions help handle request and response objects efficiently, such as authenticating users before granting access to certain routes.
* **Integration with MongoDB**: Works seamlessly with MongoDB for efficient data retrieval and storage.

In a typical MERN stack application, Express acts as the intermediary layer, connecting the React front end with the MongoDB database and managing API requests.

**React.js (Frontend Layer)**

React.js is a front-end library that allows developers to build dynamic, single-page web applications (SPAs). It provides tools for creating complex user interfaces and ensures a responsive user experience.

**Key features of React.js include**:

* **Component-Based Architecture**: Applications are structured as a collection of reusable components, making the codebase more maintainable and easier to debug.
* **Virtual DOM**: React uses a virtual DOM to optimize rendering, allowing for efficient updates and improved performance.
* **State Management**: React's state management allows components to manage their own data and update the UI dynamically in response to user interactions.

**Node.js (Server Layer)**

Node.js is a JavaScript runtime environment that enables developers to write server-side code using JavaScript. It allows for the development of fast and scalable server-side applications.

**Key features of Node.js include:**

* **Event-Driven Architecture**: Uses a non-blocking, event-driven architecture, allowing it to handle multiple requests efficiently, making it suitable for applications with high I/O operations.
* **Asynchronous Programming**: Operates in an asynchronous environment, enabling I/O operations to occur in the background without blocking the main execution thread.
* **NPM (Node Package Manager)**: Provides access to thousands of reusable libraries and modules that can be easily integrated into applications.

**How MERN Stack Works Together**

The four layers of the MERN stack interact as follows:

1. **Frontend Interaction**: Users interact with the React front end, responsible for rendering the UI and handling application elements. React makes asynchronous HTTP requests (API calls) to the backend server (Node.js with Express).
2. **Backend Processing**: Express.js processes these API requests on the server side. It determines the logic to execute based on the request and may interact with the MongoDB database to retrieve or store data.
3. **Database Operations**: MongoDB stores and retrieves data in a flexible JSON-like format, providing seamless integration with the rest of the JavaScript-based technologies in the stack.
4. **Response to Client**: Once the server (Node.js) has processed the request, it sends the appropriate response (data or confirmation) back to the React front end, updating the view dynamically for the user.

**METHODOLOGY:**

### **Objective**:

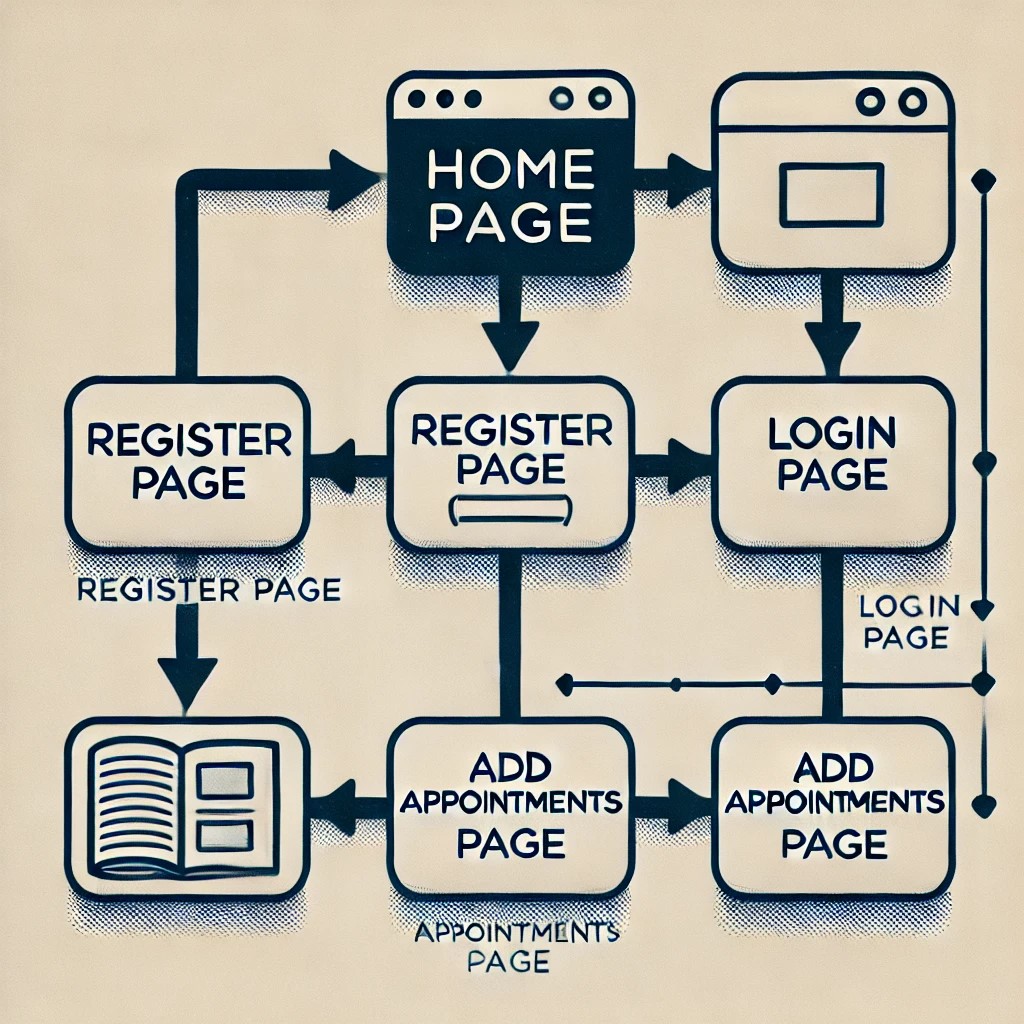
The goal of this project is to develop a comprehensive, dynamic, and scalable web application for managing appointments using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The system is designed to provide a seamless user experience, allowing users to register, log in, and manage their appointments in an intuitive manner. By leveraging the strengths of the MERN stack, the application will ensure that data is securely stored and retrieved from the MongoDB database, while providing a fast, responsive interface through React. The project also aims to highlight the benefits of using a unified JavaScript ecosystem across all layers of development, ensuring consistency, maintainability, and performance.

Additionally, the system focuses on key aspects such as scalability, security, and user-friendliness, ensuring that it can accommodate an increasing number of users and appointments without compromising on performance or data integrity. The project also aims to provide a foundation for future enhancements, such as notification systems, advanced search features, and additional user functionalities.

### **Problem Statement:**

The challenge addressed by this project is the creation of a reliable and efficient web platform that allows users to easily manage their appointments. Many users face difficulties with disorganized or inefficient appointment management systems that are either too complex or too limited in functionality. The objective is to design a system that allows users to effortlessly register, authenticate, and manage appointments in a streamlined and secure manner.

The platform will include features such as user authentication (register and login), appointment creation, and data storage in a secure MongoDB database. This will ensure that the system is not only user-friendly but also capable of handling multiple users and appointments simultaneously. The system addresses challenges such as real-time data updates, secure access to user information, and scalability to support an increasing user base while maintaining a smooth and efficient user experience. The final goal is to provide a simple yet robust solution for appointment scheduling that can be adapted to various industries and use cases.

**BLOCK DIAGRAM: **

**MODULE EXPLANATION:**

### 1. **User Authentication Module**

This module enables users to register and log in to the system. Registration information is validated and stored in the MongoDB database. The login module validates the credentials and ensures that only authenticated users can access the home and appointment pages.

### 2. **Home Page and Navigation Module**

The home page is the central hub after users log in. It offers navigation to other sections of the application, including the appointment management page. The home page serves as the starting point after successful login.

### 3. **Appointment Management Module**

In this module, users can add new appointments. The details of each appointment, such as time, date, and description, are stored in the backend MongoDB database. Users can view and manage their appointments on the appointment page, which dynamically updates as appointments are added.

### 4. **Database Management Module**

The MongoDB database is used to store and retrieve user and appointment data. This module ensures secure and efficient data management, handling everything from user registration details to the appointments that users create and manage.

### 5. **Alert and Notification Module**

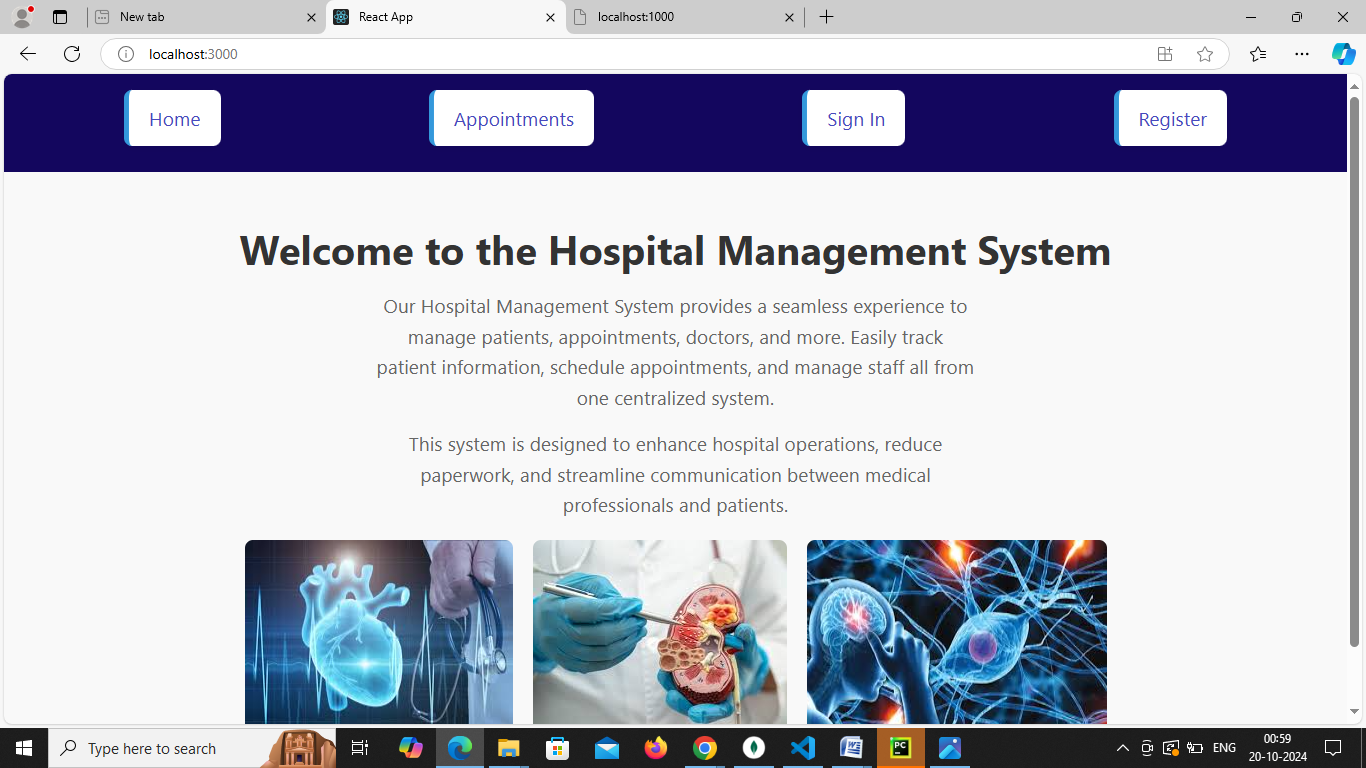
This module provides feedback to users in the form of alerts and notifications. For instance, upon successful registration or appointment creation, the system displays notifications to inform the user that the action was completed successfully.

### 6. **Backend API Module**

Using Express.js, this module handles all the communication between the frontend and the database. It processes API requests for user registration, login, appointment creation, and data retrieval, ensuring that the backend seamlessly interacts with both the frontend (React) and the database (MongoDB).

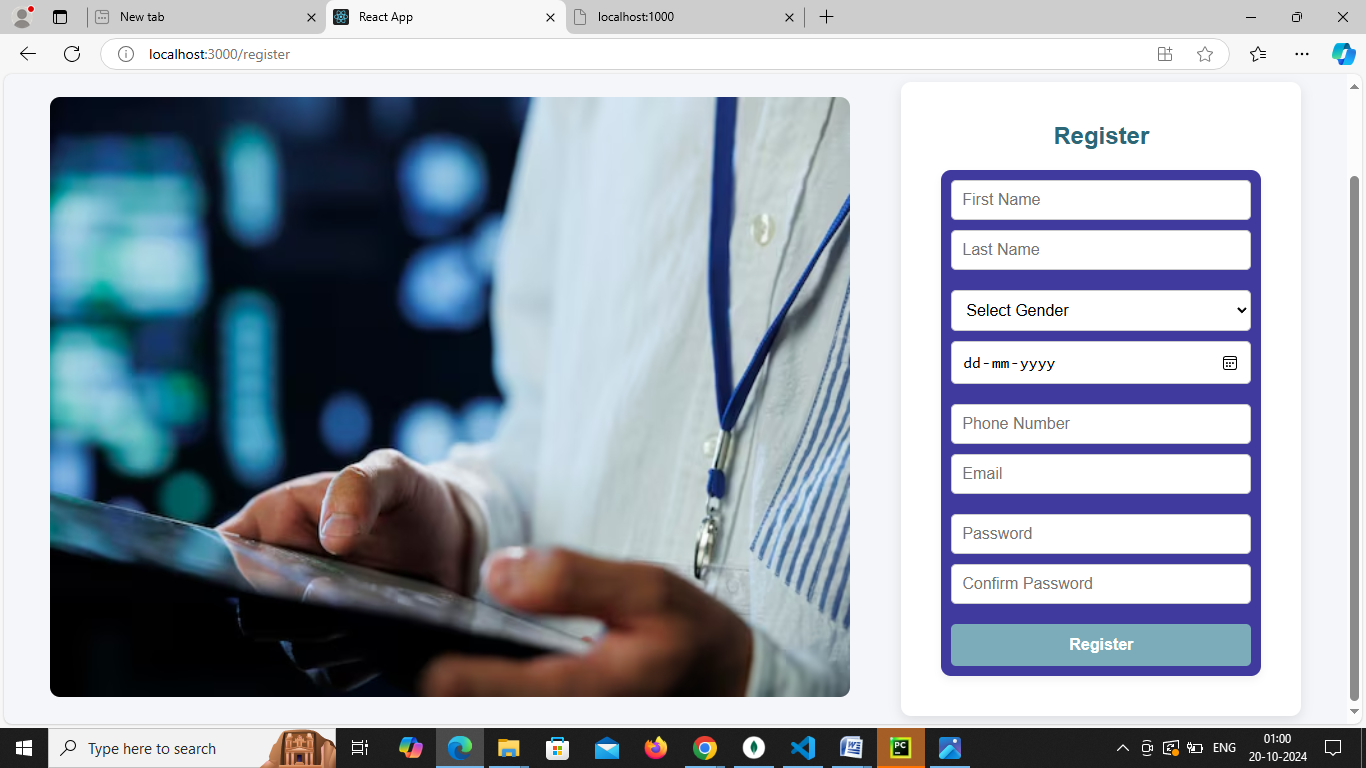
**RESULTS AND DISCUSSION:**

**SCREENSHOTS:**

****

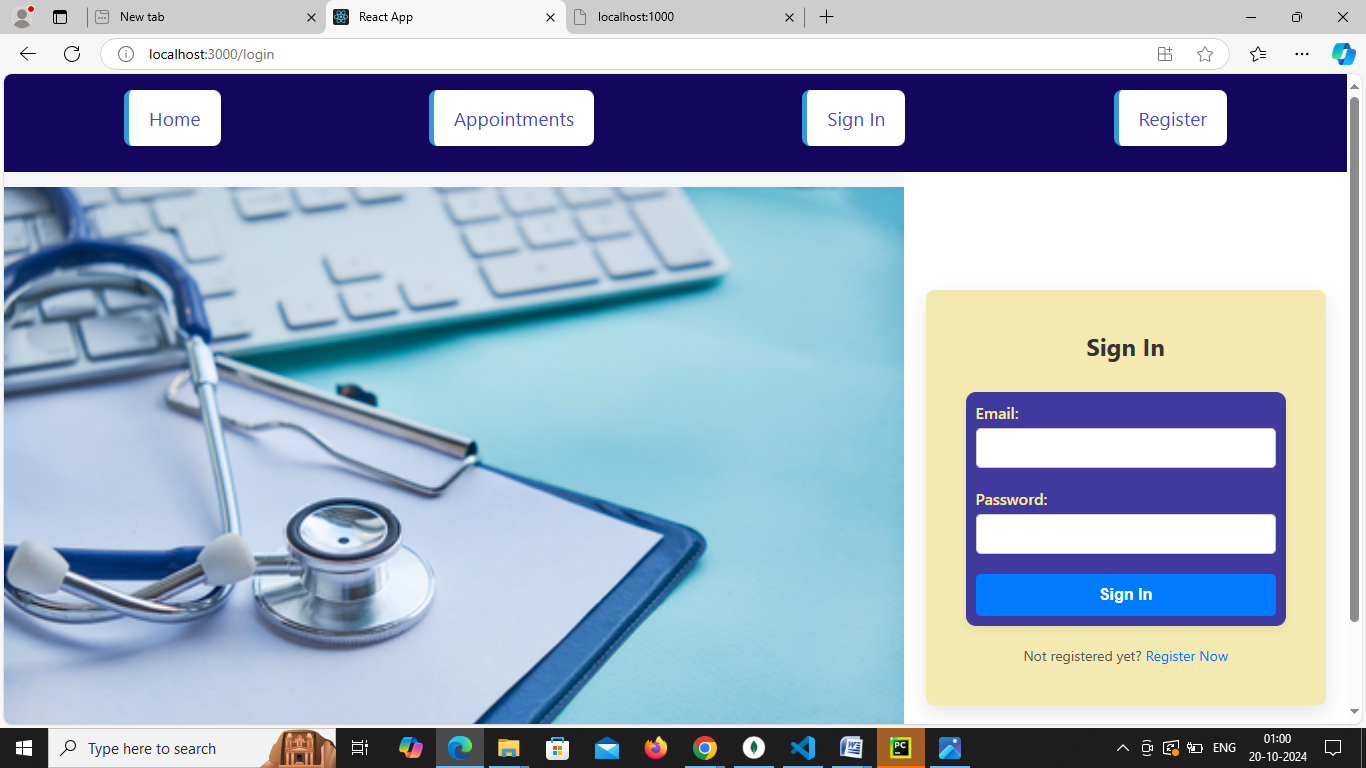
**Figure 1.1:** Index Page

The Figure 1.1 represents the home page of the Hospital Management App



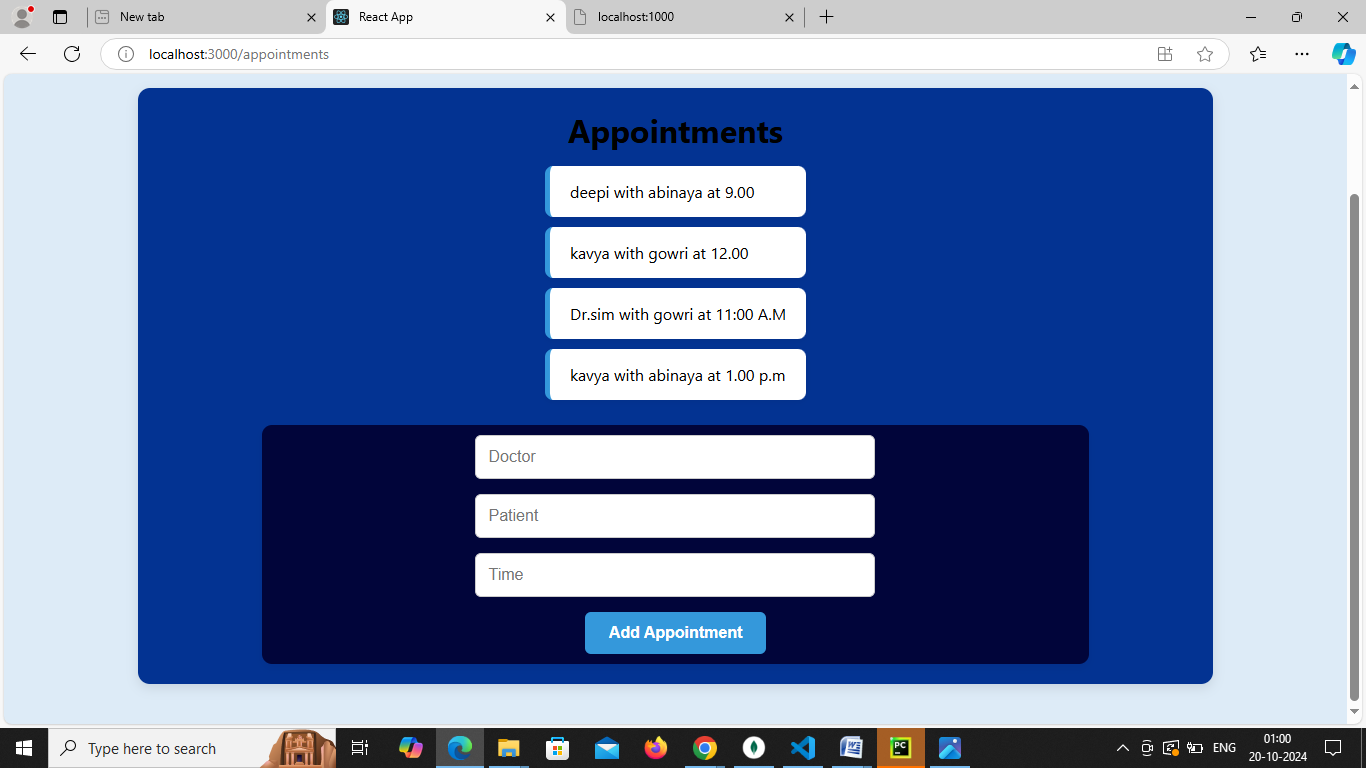
**Figure 1.2:** Register Page

The Figure 1.2 represents the registration page where the new users can register.

****

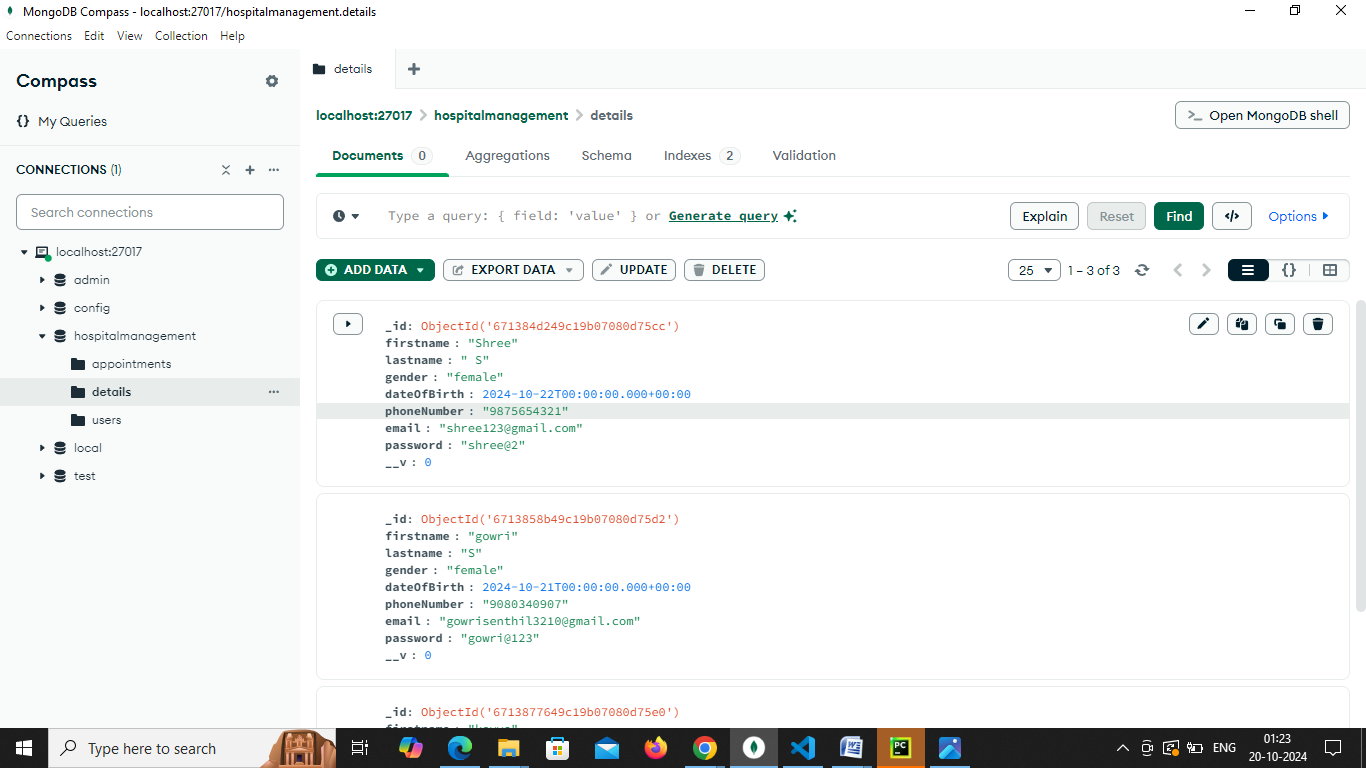
**Figure 1.3:** Sign in Page

The Figure 1.3 represents the Login page where you can login and navigate to appoinment page



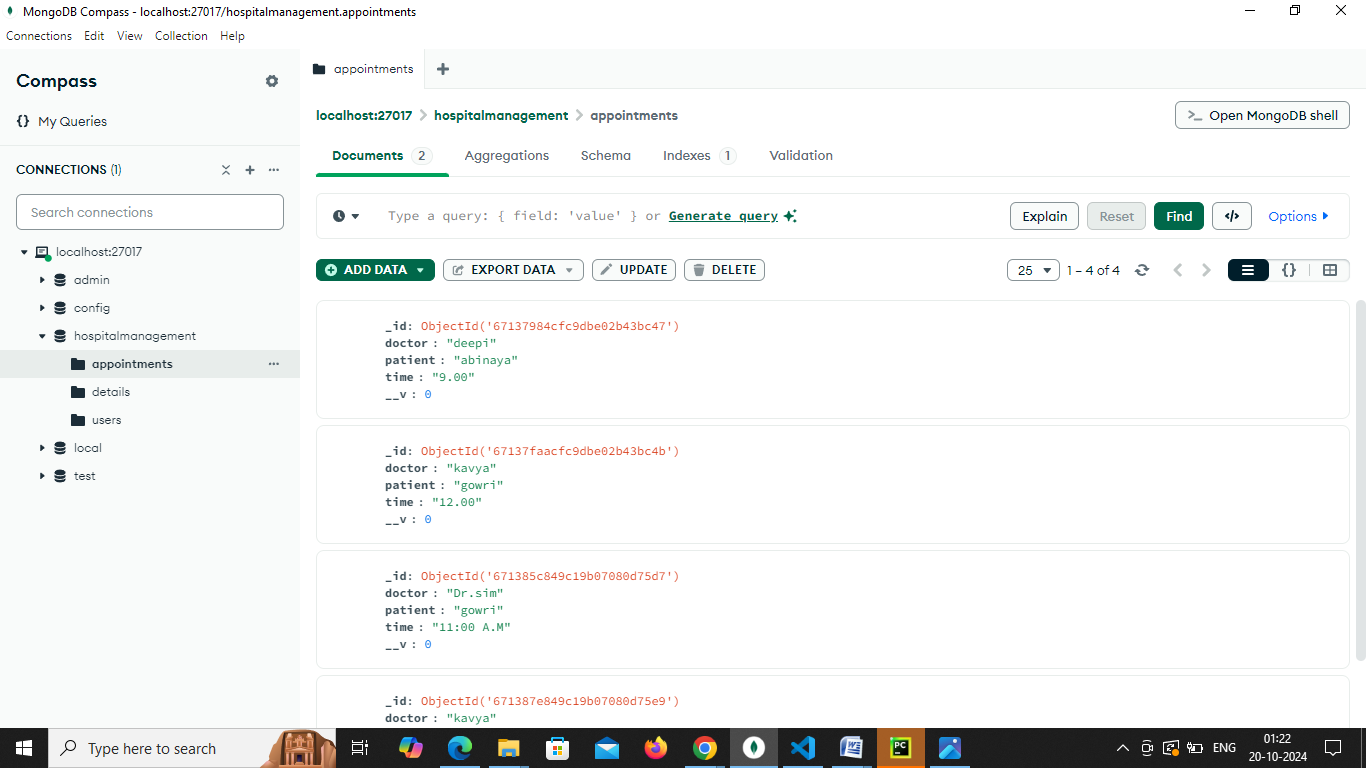
**Figure 1.4:** Appoinments Page

In Figure 1.4 you can add appointments with doctor in the respective times

****

**Figure 1.5:** Mongo DB Collection

The Figure 1.9 represents the registered users data collected from the front end and stored it in the collection.



**Figure 1.6:** Mongo DB Collection

The Figure 1.9 represents the Appoinments data collected from the front end and stored it in the collection.

**DISCUSSION:**

* **User Interaction:**

The React-based user interface in the project ensures a smooth and intuitive experience for users when registering, logging in, and managing appointments. With its dynamic nature, users can easily add appointments and view their details without the need for page reloads, increasing overall user satisfaction.

* **System Responsiveness:**

Node.js, in combination with Express.js, efficiently handles the flow of user data and requests from the frontend, such as registration and login submissions. The asynchronous, non-blocking architecture ensures that users experience minimal delays when interacting with different pages and services in the system, like storing user data in the MongoDB database and retrieving appointment information.

* **Data Integrity:**

MongoDB provides a flexible, schema-less data structure that makes it easier to handle registration and appointment data. It stores diverse data types from different users, ensuring quick and efficient access, while maintaining consistency across the platform. The database integration allows users to securely register, log in, and store appointment information without errors or loss of data.

* **User Satisfaction:**

The overall user satisfaction has been significantly improved by streamlining the flow between registration, login, and appointment management. The ability to quickly store and retrieve data from the backend without facing downtime or slow loading times enhances the user experience, and users are more likely to continue using the system.

* **Future Enhancements:** In future versions of the project, additional features like email notifications for upcoming appointments, integration with external calendars (e.g., Google Calendar), or advanced appointment filtering could be considered. Additionally, incorporating role-based access control or user profiles could further personalize the user experience.

**CONCLUSION:**

The project effectively demonstrates the power of the MERN stack in building a full-stack web application for appointment management. By utilizing MongoDB for flexible data storage, Express.js for backend routing, React.js for a dynamic user interface, and Node.js for server-side operations, the application ensures seamless interaction across all components. The unified JavaScript ecosystem allows for efficient data flow, scalability, and a smooth user experience. Future enhancements could further improve functionality, but the current system already offers a robust and user-friendly solution for appointment scheduling and management.

**REFERENCES:**

**FreeCodeCamp: Full-Stack MERN Tutorial**  
https://www.freecodecamp.org/news/mern-stack-tutorial-build-a-full-stack-app

**MDN Web Docs (Mozilla) - Web Development**  
 <https://developer.mozilla.org/en-US/docs/Learn>

**DigitalOcean: Building Modern Web Applications with the MERN Stack**  
https://www.digitalocean.com/community/tutorial\_series/building-modern-web-applications-with-the-mern-stack

**Dev.to: MERN Stack Resources**  
https://dev.to/t/mern