**Project Report**

**1. INTRODUCTION**

1.1 Project Overview

1.2 Purpose

**2. IDEATION PHASE**

2.1 Problem Statement

2.2 Empathy Map Canvas

2.3 Brainstorming

**3. REQUIREMENT ANALYSIS**

3.1 Customer Journey map

3.2 Solution Requirement

3.3 Data Flow Diagram

3.4 Technology Stack

**4. PROJECT DESIGN**

4.1 Problem Solution Fit

4.2 Proposed Solution

4.3 Solution Architecture

**5. PROJECT PLANNING & SCHEDULING**

5.1 Project Planning

**6. FUNCTIONAL AND PERFORMANCE TESTING**

6.1 Performance Testing

**7. RESULTS**

7.1 Output Screenshots

**8. ADVANTAGES & DISADVANTAGES**

**9. CONCLUSION**

**10. FUTURE SCOPE**

**11. APPENDIX**

GitHub & Project Demo Link

**1.INTRODUCTION**

**1.1 Project Overview:**

DocSpot – Seamless Appointment Booking for Health is a full-stack web application designed to simplify and digitize the process of booking medical appointments between patients and healthcare providers. Traditional appointment booking methods often involve long waiting times, manual scheduling, lack of transparency, and inefficient communication between patients and doctors. This project aims to address these challenges by providing a centralized online platform where users can easily search for doctors, check availability, and schedule appointments in real time.

Doctors can manage their schedules, confirm or reschedule appointments, and update consultation details. Admins oversee the overall platform operations, approve doctor registrations, and ensure system compliance. The application follows a modern client-server architecture using React.js for the frontend, Express.js and Node.js for backend services, and MongoDB for data management.

**1.2 Purpose:**

The primary purpose of the DocSpot application is to enhance accessibility, efficiency, and convenience in healthcare appointment scheduling through a digital solution. The project focuses on reducing manual effort, minimizing waiting times, and improving interaction between patients and doctors.

**Key objectives of the system include:**

* Providing an intuitive platform for users to discover healthcare professionals and book appointments online.
* Allowing doctors to efficiently manage their schedules and patient requests.
* Enabling administrators to maintain system governance and ensure secure platform operations.
* Delivering real-time appointment updates, notifications, and status tracking.
* Creating a scalable and user-friendly healthcare booking system using modern full-stack technologies.

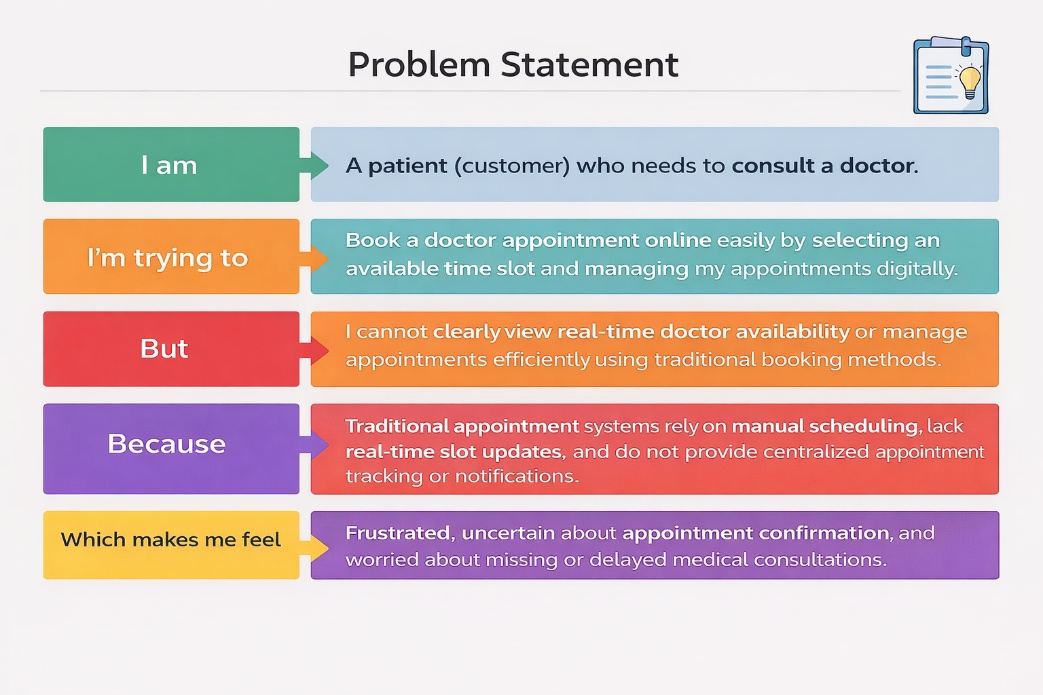
Overall, the project aims to bridge the gap between patients and healthcare services by offering a reliable and structured appointment management solution that improves user experience and operational efficiency.

**2.IDEATION PHASE**

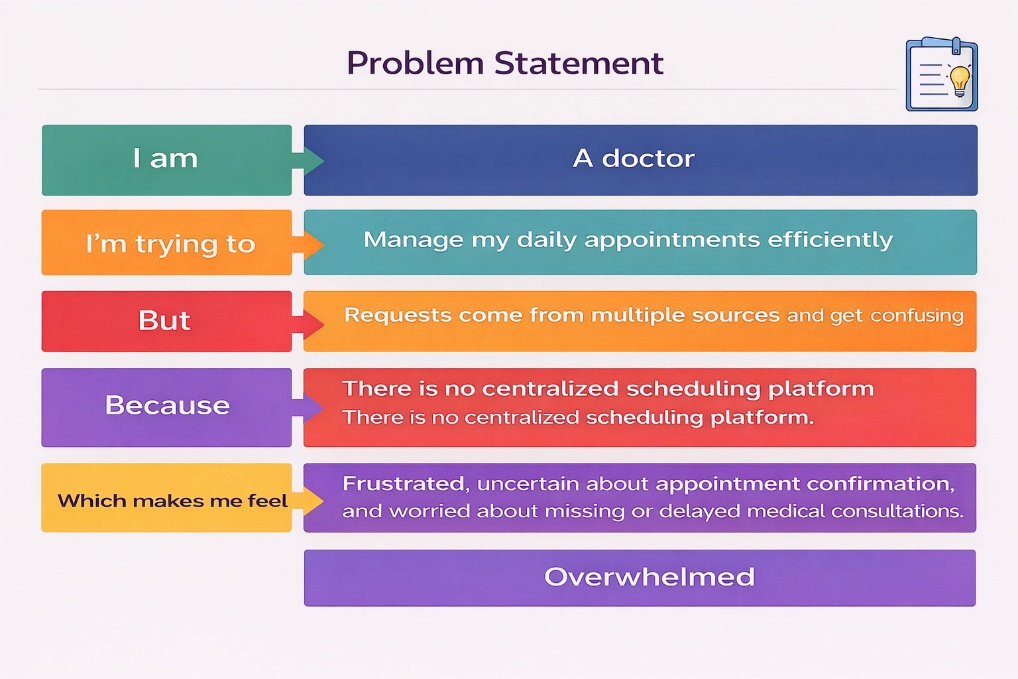
**2.1 Problem Statement:**

In today’s healthcare systems, booking and managing doctor appointments can be time-consuming and confusing for both patients and doctors. Many traditional methods do not provide real-time availability or a centralized scheduling system, which leads to missed updates, overlapping requests, and user frustration. The DocSpot platform aims to solve these challenges by providing a streamlined digital solution for efficient appointment booking and management.

**Problem Statement -1:**



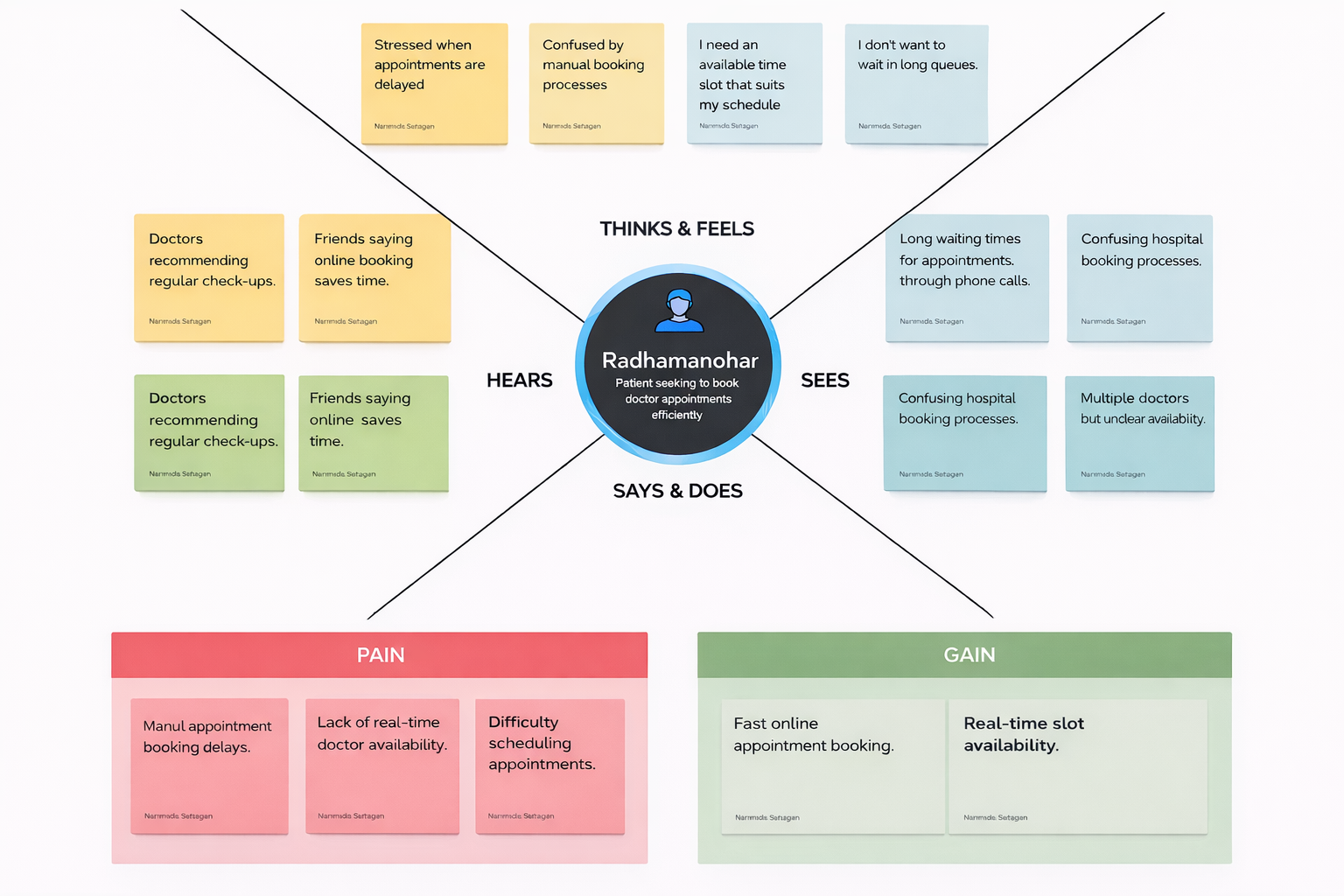
**Problem** **Statement -2:**

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Statement (PS)** | **I am** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | A patient who needs medical consultation. | Book a doctor appointment online easily by selecting an available time slot. | I cannot clearly view real-time doctor availability or appointment status. | Traditional booking methods do not provide updated slot availability or digital tracking. | Anxious, uncertain, and frustrated about appointment confirmation. |
| PS-2 | A doctor managing multiple patient consultations. | Manage and organize my daily appointments efficiently. | Appointment requests arrive from multiple sources and become difficult to track. | There is no centralized appointment scheduling and management platform. | Overwhelmed and stressed while handling appointments. |

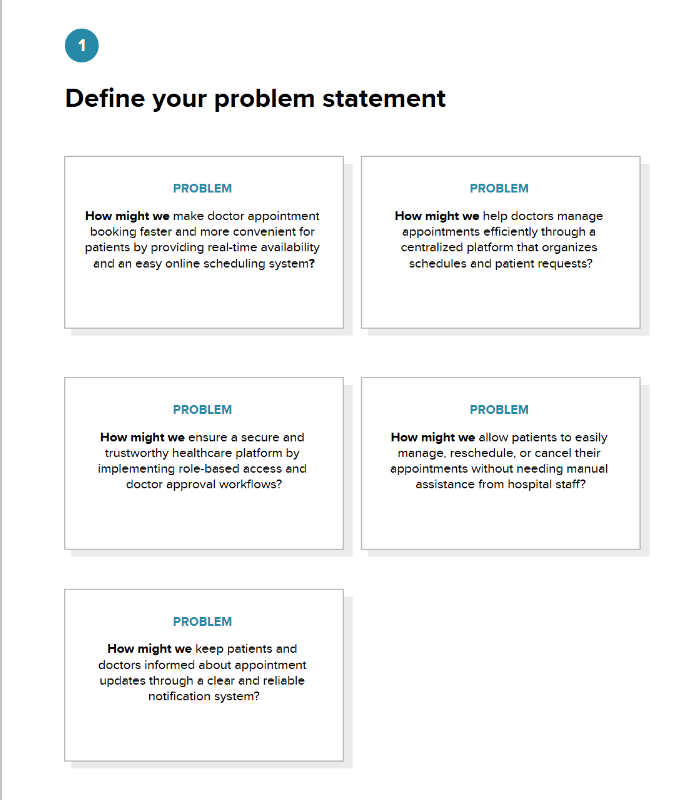
**2.2 Empathy Map:**

**User: - Radhamanohar Kumar Vasamsetti** (Working Professional Needs Routine Medical Consultation)

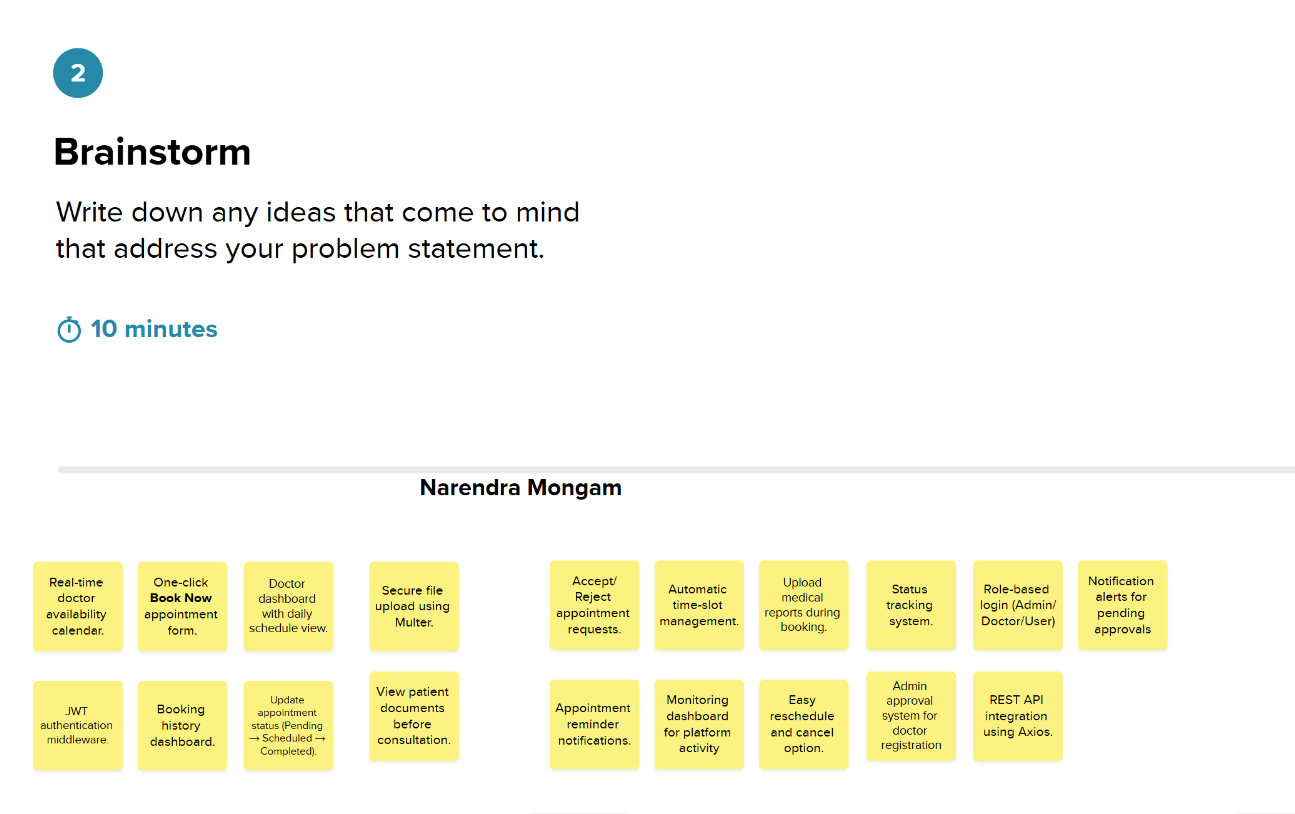


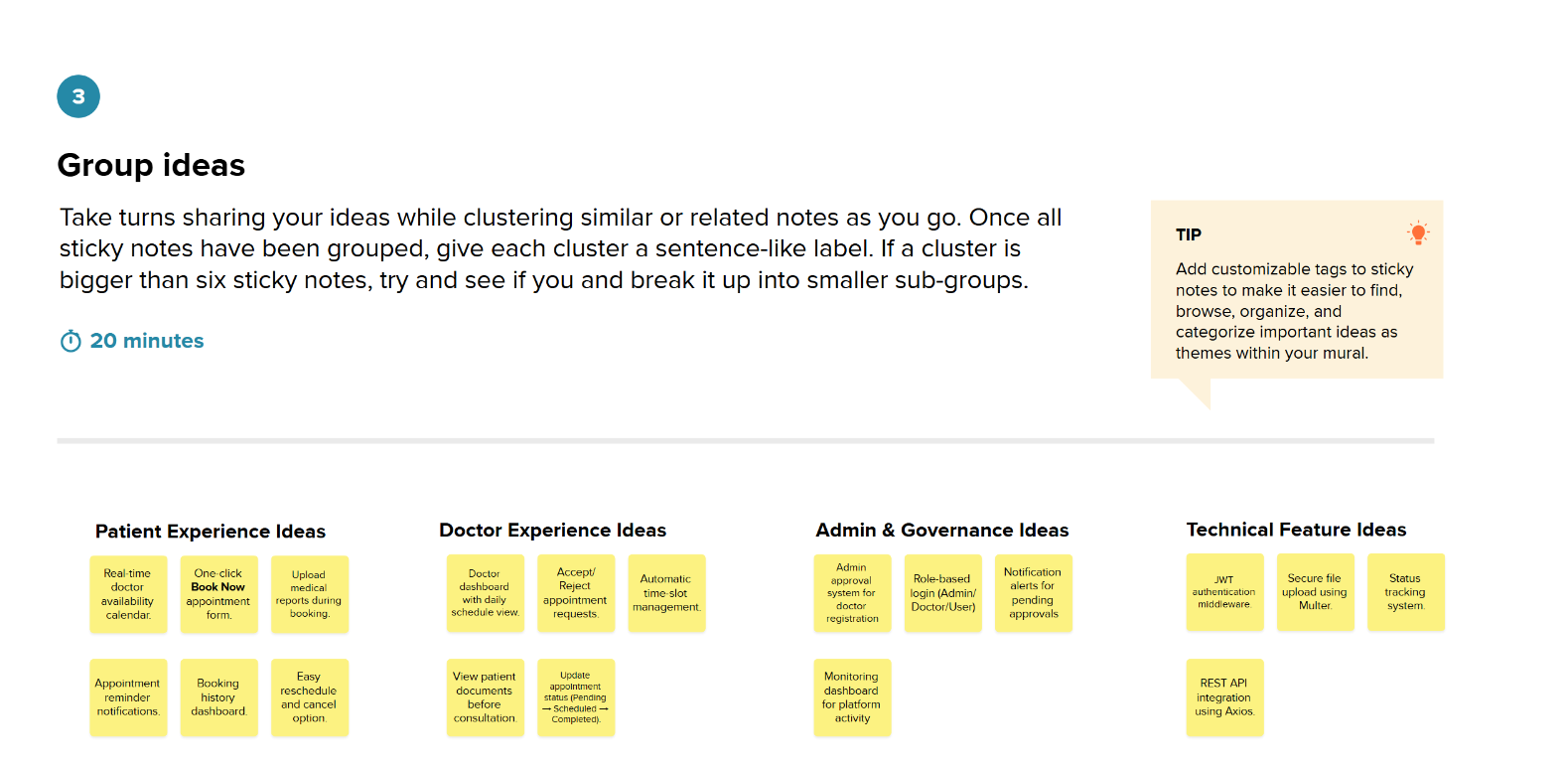
**2.2 Brainstorm & Idea Prioritization: -**

**Step-1: Team Gathering, Collaboration and Select the Problem Statement**



**Step-2: Brainstorm, Idea Listing and Grouping**

****

****

**Step-3: Idea Prioritization**



**REQUIREMENT ANALYSIS**

**3.1 Customer Journey map:**

****

**3.2 Solution Requirement**

**Functional Requirements:**

Following are the functional requirements of the proposed solution.

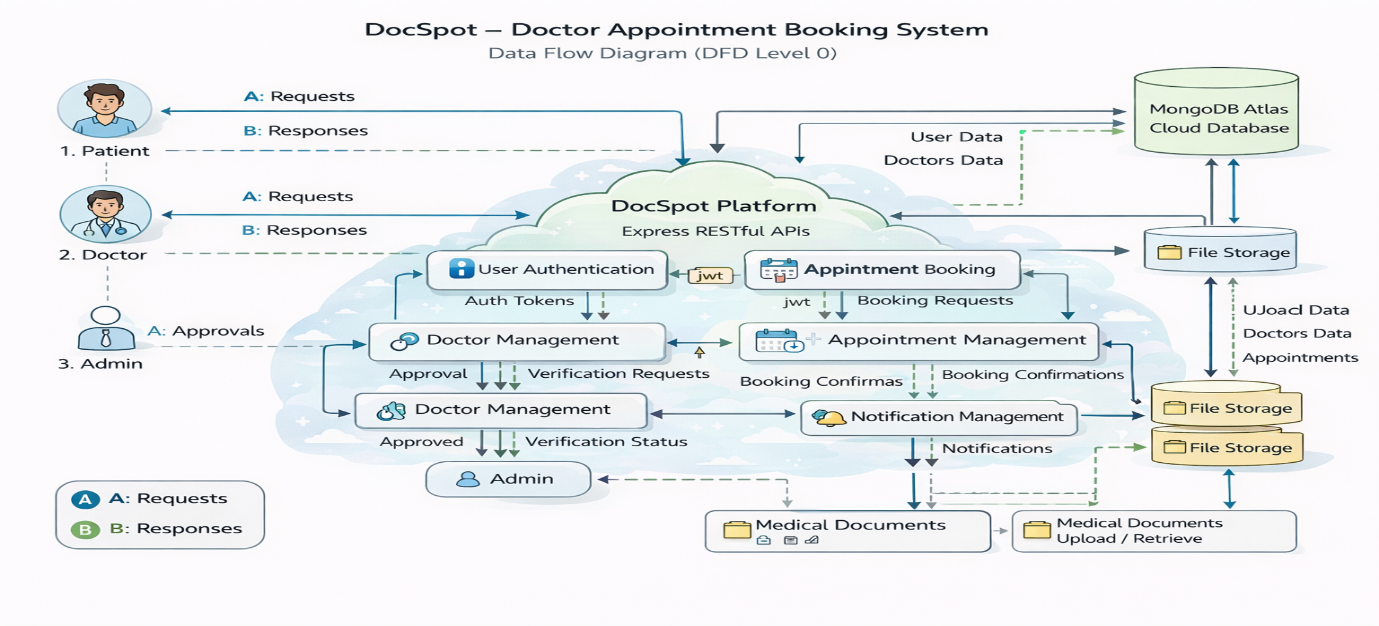
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Form  Role selection (User / Doctor)  Secure password creation |
| FR-2 | User Authentication | Login using Email & Password  JWT-based authentication & session validation |
| FR-3 | Doctor Browsing & Search | View doctor list on dashboard  Filter by specialization / availability |
| FR-4 | Appointment Booking | Select date & time slot  Upload medical documents  Submit application request |
| FR-5 | Appointment Management | View appointment history  Cancel or reschedule appointment  Track status (Pending / Scheduled / Completed) |
| FR-6 | Doctor Dashboard | View incoming appointment requests  Accept / Reject / Reschedule bookings |
| FR-7 | Admin Governance | Approve doctor registrations  Monitor platform activity |
| FR-8 | Notification System | Appointment confirmation alerts  Unread/read notification tracking |

**Non-functional Requirements:**

Following are the non-functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | Provides a simple and intuitive interface enabling patients and doctors to manage appointments without technical expertise. |
| NFR-2 | **Security** | JWT authentication, encrypted passwords using bcrypt, and secure document upload ensure protection of sensitive healthcare data. |
| NFR-3 | **Reliability** | |  | | --- | |  |   Ensures accurate appointment scheduling, consistent status updates, and prevention of booking conflicts. |
| NFR-4 | **Performance** | Fast dashboard loading and responsive booking operations through optimized REST API communication. |
| NFR-5 | **Availability** | Web-based platform accessible anytime for patients, doctors, and administrators. |
| NFR-6 | **Scalability** | MERN architecture with MongoDB Atlas enables scaling to support more users, doctors, and future healthcare features. |

**3.3 Data Flow Diagram: -**

****

**User Stories**

| **User Type** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| --- | --- | --- | --- | --- | --- | --- |
| Customer (Web User) | Registration | USN-1 | |  | | --- | |  |   As a user, I can register using email and password to access the platform. | Account created successfully and redirected to dashboard. | High | Sprint-1 |
| Customer (Web User) | Authentication | USN-2 | As a user, I can log in securely using my credentials. | Successful login generates JWT and opens user dashboard. | High | Sprint-1 |
| Customer (Web User) | Doctor Listing | USN-3 | As a user, I can view available doctors filtered by specialization and availability. | Doctor list loads from database with approved doctors only. | High | Sprint-1 |
| Customer (Web User) | Appointment Booking | USN-4 | As a patient, I can book appointment by selecting date & time slot. | Appointment created with status “Pending” | High | Sprint-2 |
| Customer (Web User) | Document Upload | USN-5 | As a user, I can upload medical documents while booking an appointment. | File uploads successfully and stored in local storage. | Medium | Sprint-2 |
| Doctor | Appointment Management | USN-6 | As a doctor, I can approve or reject appointment requests. | Appointment status updates to Scheduled or Rejected. | High | Sprint-3 |
| Administrator | Doctor Approval | USN-7 | As an admin, I can approve or reject doctor registration requests. | Approved doctors become visible to patients. | High | Sprint-3 |
| Administrator | Platform Management | USN- 8 | As an admin, I can monitor users, doctors, and appointments. | Admin dashboard displays system data correctly. | High | Sprint-3 |
| Customer Care Executive | Notifications | USN-9 | As a user, I receive appointment status notifications. | Notifications stored in DB with unread/read tracking. | Medium | Sprint-4 |
| Customer (Web User) | Appointment History | USN-10 | As a user, I can view my past and upcoming appointments. | Appointment history visible in dashboard. | Medium | Sprint-5 |
| Doctor | Schedule Management | USN- 11 | As a doctor, I can view my appointment schedule by time slots. | Doctor dashboard shows booked slots correctly. | Medium | Sprint-4 |
| Customer (Web User) | Appointment Cancellation | USN- 12 | |  | | --- | |  |   As a user, I can cancel my booked appointment. | Appointment status updated and doctor notified. | Medium | Sprint-5 |

**3.4 Technology Stack: -**

**Technical Architecture: -**

The DocSpot application follows a 3-Tier Client–Server Architecture:

* Presentation Layer (Frontend): React.js web interface where users, doctors, and admins interact with the system.
* Application Layer (Backend): Node.js and Express.js handle API requests, authentication, appointment processing, and role-based access control.
* Data Layer (Database): MongoDB stores user profiles, doctor details, appointment records, and notifications.

The frontend communicates with backend REST APIs using Axios, while JWT authentication secures protected routes. File uploads such as medical documents are handled through Multer middleware.

**Table-1: Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Component** | **Description** | **Technology** |
|  | User Interface | |  | | --- | |  |   Web interface for Patients, Doctors, and Admin dashboards enabling booking, management, and monitoring of appointments | React.js, HTML, CSS, Bootstrap, Material UI |
|  | Application Logic-1 (Authentication) | Handles user login, registration, role-based access, and secure authorization using tokens | Node.js, Express.js, JWT, bcryptjs |
|  | Application Logic-2 (Appointment Management) | Processes appointment booking, slot selection, scheduling, approval, and cancellation workflows | Express.js REST APIs, Node.js, Controllers |
|  | Application Logic-3 (Notification System) | Manages appointment status updates, alerts, and unread/read notification tracking | Node.js Controllers, REST APIs |
|  | Database (Primary Storage) | Stores users, doctors, appointments, schedules, and notification data | MongoDB, Mongoose ODM |
|  | Cloud Database | Cloud-hosted scalable database supporting remote access and persistent storage | MongoDB Atlas |
|  | File Storage | Stores uploaded medical documents associated with appointments | Local File System / Cloud Storage Service |
|  | API Communication Layer | Enables communication between frontend and backend services. | Axios (HTTP Client) |
|  | External API Integration | Not required for current project scope | Not Applicable |
|  | Machine Learning Model | AI/ML components are not used in this project | Not Applicable |
|  | Infrastructure / Deployment | Local server environment used for development and testing | Node.js Local Server |

**Table-2: Application Characteristics:**

| **S. No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Frameworks used for building scalable UI and backend APIs | React.js, Express.js, Node.js |
|  | Security Implementations | Secure authentication, encrypted password storage, and protected API routes | JWT Authentication, bcryptjs, Middleware |
|  | Scalable Architecture | Three-tier architecture separating frontend, backend, and database layers | REST Architecture, MongoDB Atlas |
|  | Availability | |  | | --- | |  |   Web application accessible anytime through browser-based interface | Node.js Server |
|  | Performance | Fast API responses and asynchronous communication | Axios, Express.js |
|  | Data Consistency | Ensures accurate appointment status and scheduling updates | MongoDB Transactions & Validation |
|  | Maintainability | Modular backend controllers and reusable frontend components | MERN Stack Modular Design |

**4. PROJECT DESIGN**

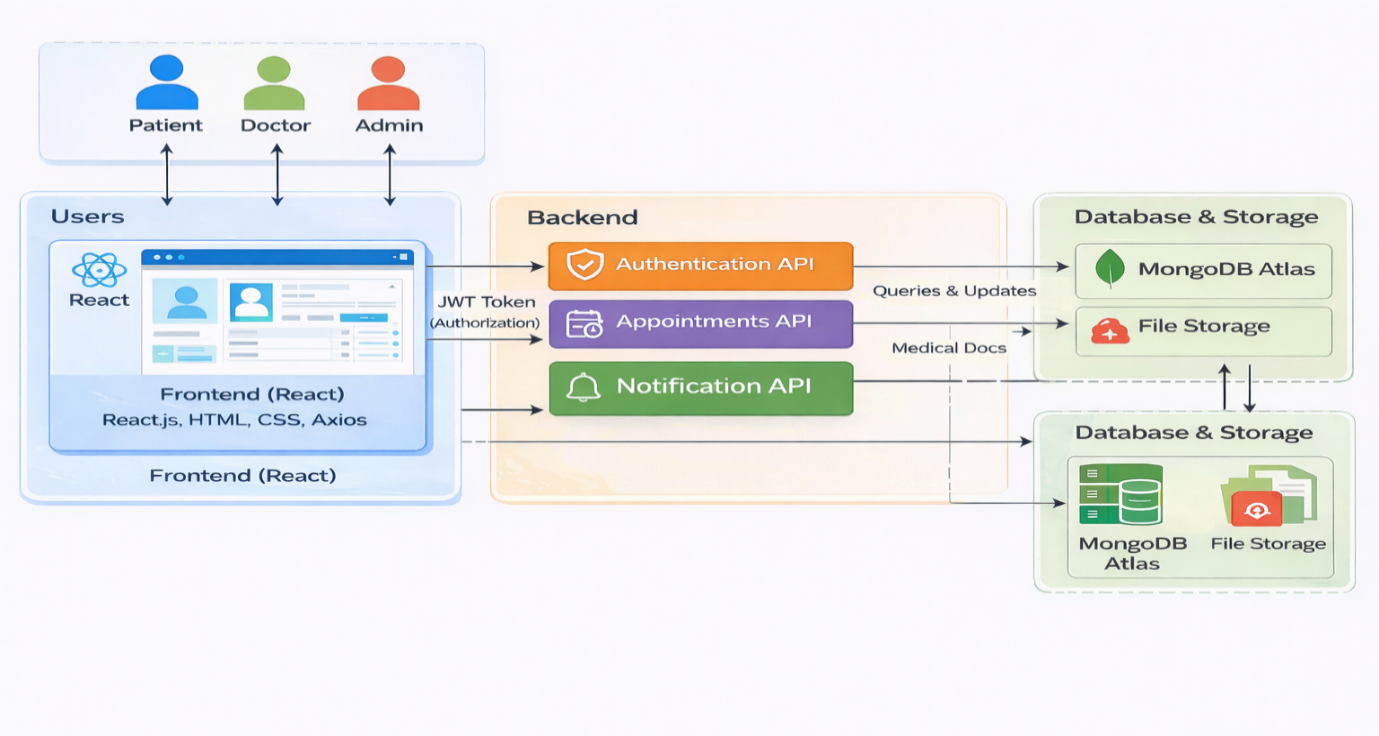
**4.1 Problem Solution Fit:**

****

**4.2 Proposed Solution:**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
|  | Problem Statement (Problem to be solved) | Patients experience difficulty booking doctor appointments due to manual scheduling processes, lack of real-time slot availability, and long waiting times. Doctors face challenges managing multiple appointment requests efficiently, while administrators require a centralized and secure system to verify doctors and maintain platform reliability. |
|  | Idea / Solution description | **DocSpot** is a MERN-stack web-based doctor appointment booking platform that enables patients to schedule appointments online through a centralized system. The platform provides role-based dashboards for Admin, Doctor, and Patient, real-time slot-based booking, secure JWT authentication, medical document upload, appointment status tracking, and REST API integration with MongoDB Atlas cloud database. |
|  | Novelty / Uniqueness | The system introduces a role-based healthcare workflow with admin-controlled doctor approval, real-time appointment tracking, secure document handling, and a simplified user-friendly booking interface designed specifically for efficient healthcare appointment management. |
|  | Social Impact / Customer Satisfaction | The platform reduces hospital waiting times, improves healthcare accessibility, ensures interaction with verified doctors, enhances appointment transparency, and increases patient satisfaction through digital appointment management and automated notifications. |
|  | Business Model (Revenue Model) | The proposed business model includes commission-based appointment bookings, subscription plans for doctors, premium doctor profile listings, and future expansion into telemedicine consultation services and healthcare partnerships. |
|  | Scalability of the Solution | The system follows a scalable 3-tier architecture consisting of a React.js frontend, Express.js/Node.js backend, and MongoDB Atlas cloud database. This architecture supports high scalability, efficient data handling, and future expansion to accommodate additional healthcare services and increased user traffic. |

**4.3 Solution Architecture:**

****

*Figure-1: Architecture and data flow of the DocSpot doctor appointment booking platform*

**5. PROJECT PLANNING & SCHEDULING**

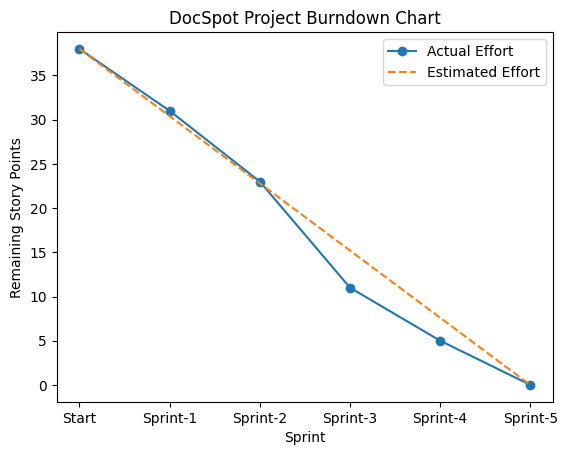
**5.1 Project Planning:**

**Product Backlog, Sprint Schedule, and Estimation:**

| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Registration | USN-1 | As a user, I can register using email and password to access the platform. | 2 | High | Team |
| Sprint-1 | Authentication | USN-2 | As a user, I can log in securely using my credentials. | 2 | High | Team |
| Sprint-1 | Doctor Listing | USN-3 | As a user, I can view available doctors filtered by specialization and availability. | 3 | High | Team |
| Sprint-2 | Appointment Booking | USN-4 | As a patient, I can book an appointment by selecting date and time slot. | 5 | High | Team |
| Sprint-2 | Document Upload | USN-5 | As a user, I can upload medical documents while booking an appointment. | 3 | Medium | Team |
| Sprint-3 | Appointment Management | USN-6 | As a doctor, I can approve or reject appointment requests. | 5 | High | Team |
| Sprint-3 | Doctor Approval | USN-7 | As an admin, I can approve or reject doctor registration requests. | 4 | High | Team |
| Sprint-3 | Platform Management | USN-8 | As an admin, I can monitor users, doctors and appointments. | 3 | High | Team |
| Sprint-4 | Notifications | USN-9 | As a user, I receive appointment status notifications with unread/read tracking. | 3 | Medium | Team |
| Sprint-5 | Appointment History | USN-10 | As a user, I can view my past and upcoming appointments. | 3 | Medium | Team |
| Sprint-4 | Schedule Management | USN-11 | As a doctor, I can view appointment schedules based on time slots. | 3 | Medium | Team |
| Sprint-5 | Appointment Cancellation | USN-12 | As a user, I can cancel a booked appointment. | 2 | Medium | Team |

**Project Tracker, Velocity & Burndown Chart:**

| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points Completed (as on Planned End Date)** | **Sprint Release Date (Actual)** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | 7 | 4 Days | 01 Feb 2026 | 04 Feb 2026 | 7 | 04 Feb 2025 |
| Sprint-2 | 8 | 4 Days | 05 Feb 2026 | 08 Feb 2026 | 8 | 08 Feb 2025 |
| Sprint-3 | 12 | 4 Days | 09 Feb 2026 | 12 Feb 2026 | 12 | 12 Feb 2025 |
| Sprint-4 | 6 | 3 Days | 13 Feb 2026 | 15 Feb 2026 | 6 | 15 Feb 2025 |
| Sprint-5 | 5 | 3 Days | 16 Feb 2026 | 19 Feb 2026 | 5 | 09 Feb 2026 |

****

**6. FUNCTIONAL AND PERFORMANCE TESTING**

**6.1 Performance Testing:**

**Project Overview:**

Project Name: DocSpot: Seamless Appointment Booking for Health

Project Description: DocSpot is a MERN-stack web application that enables patients to book doctor appointments online with real-time availability. The platform provides role-based dashboards for Admin, Doctor, and User, secure JWT authentication, document upload, and appointment status tracking using MongoDB Atlas.

Project Version: v1.0

Testing Period: 12 Feb 2025 to 19 Feb 2025

Testing Environment: Local Development Server

Application URL: http://localhost:5173

**Testing Scope:**

**Features Tested:**

* User Registration & Login
* Role-based Dashboard
* Appointment Booking
* Doctor Approval Workflow
* Document Upload
* Appointment Status Notifications

**User Stories Tested:**

* USN-1 Registration
* USN-2 Secure Login
* USN-3 View Available Doctors
* USN-4 Appointment Booking
* USN-5 Document Upload
* USN-6 Doctor Appointment Approval
* USN-7 Admin Doctor Approval
* USN-8 Platform Monitoring
* USN-9 Notifications
* USN-10 Appointment History
* USN-11 Doctor Schedule Management

USN-12 Appointment Cancellation

**Testing Environment:**

URL/Location: http://localhost:5173

Credentials (if required): User → testuser@gmail.com / \*\*\*\*\*\*

Doctor → doctest@gmail.com / \*\*\*\*\*\*

Admin → admin@gmail.com / \*\*\*\*\*\*

**Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Test Steps** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| TC-001 | User Registration | Step1: Open Register Page  Step2: Enter details Step3: Submit | User account created successfully | Account created | Pass |
| TC-002 | User Login | Step1: Enter email & password  Step2: Click Login | Redirect to dashboard | Dashboard loaded | Pass |
| TC-003 | View Doctors | Step1: Login  Step2: Open Dashboard | Doctor list displayed | Doctors visible | Pass |
| TC-004 | Book Appointment | Step1: Select doctor Step2: Choose date Step3: Submit | Appointment request created | Appointment pending | Pass |
| TC-005 | Upload Document | Step1: Attach file during booking | File uploaded successfully | File stored | Pass |
| TC-006 | Doctor Approval | Step1: Admin approves doctor | Doctor status updated | Approved successfully | Pass |
| TC-007 | Appointment Status Update | Step1: Doctor accepts request | Status changes to Scheduled | Status updated | Pass |
| TC-008 | Notification Generation | Application status updated | Notification appears in dashboard | Notification recieved | Pass |
| TC-009 | Application History | User opens history page | Previous booking displayed | History visible | Pass |

**Bug Tracking:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bug ID** | **Bug Description** | **Steps to reproduce** | **Severity** | **Status** | **Additional feedback** |
| BG-001 | Minor UI alignment issue in dashboard | Step-1: Login  Step-2: Open Dashboard | Low | Closed | Fixed using CSS update |
| BG-002 | File upload delay on slow internet | Upload large file | Medium | In Progress | Optimization suggested |

**Sign-off:**

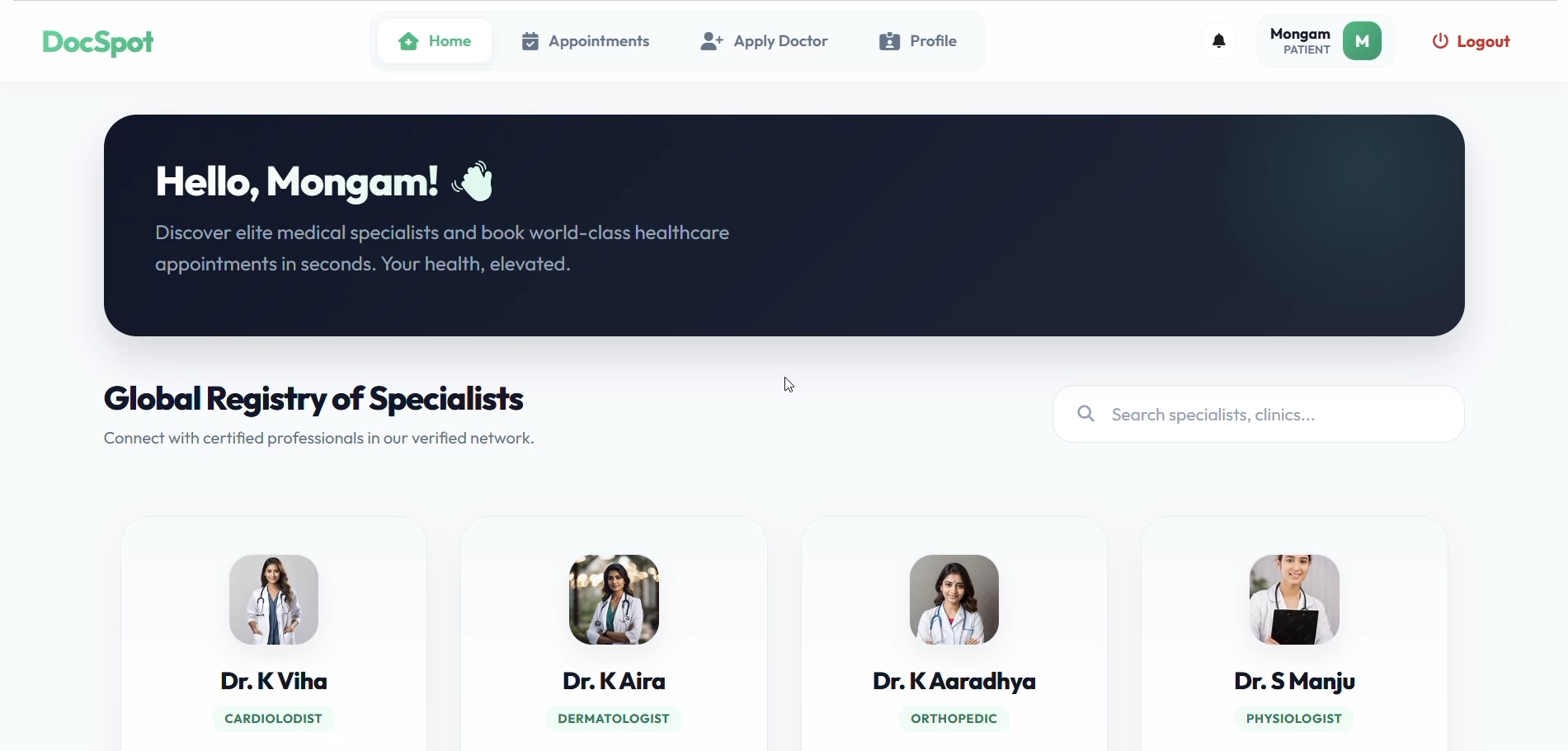
Tester Name: Vasamsetti Radhamanohar Kumar

Date: 19 February 2025

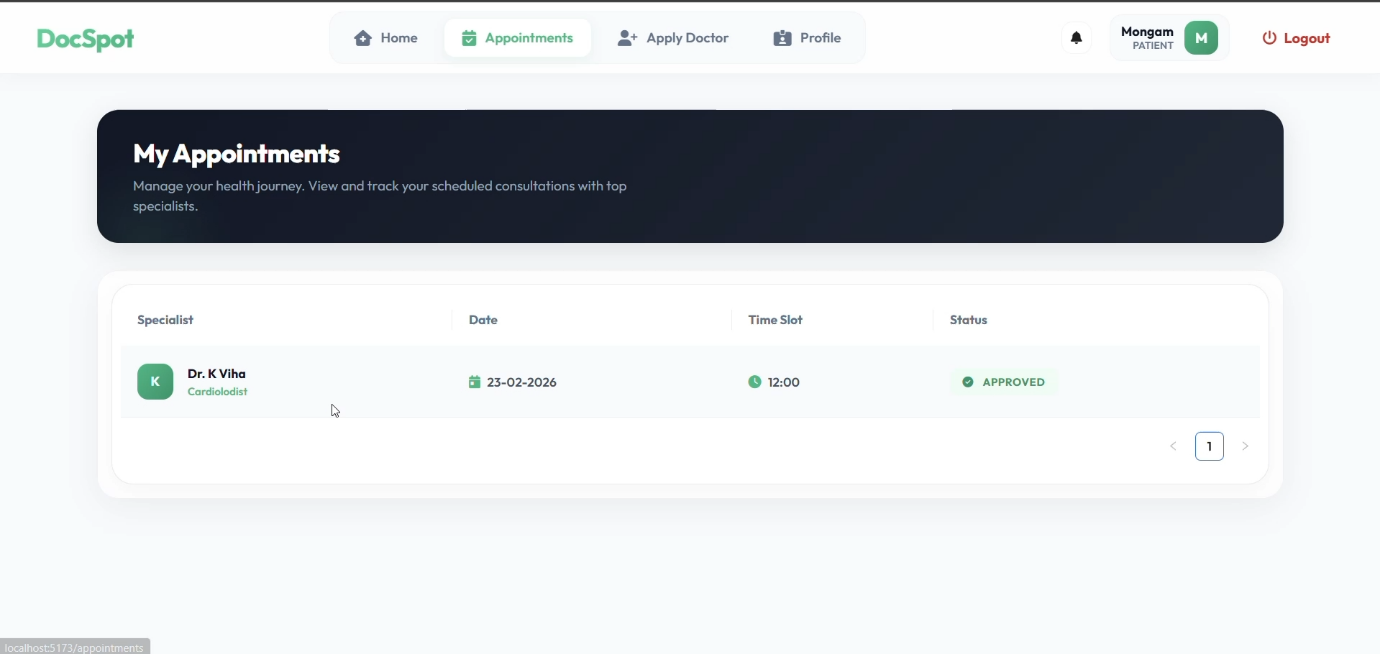
Signature: Vasamsetti Radhamanohar Kumar

**7. RESULTS**

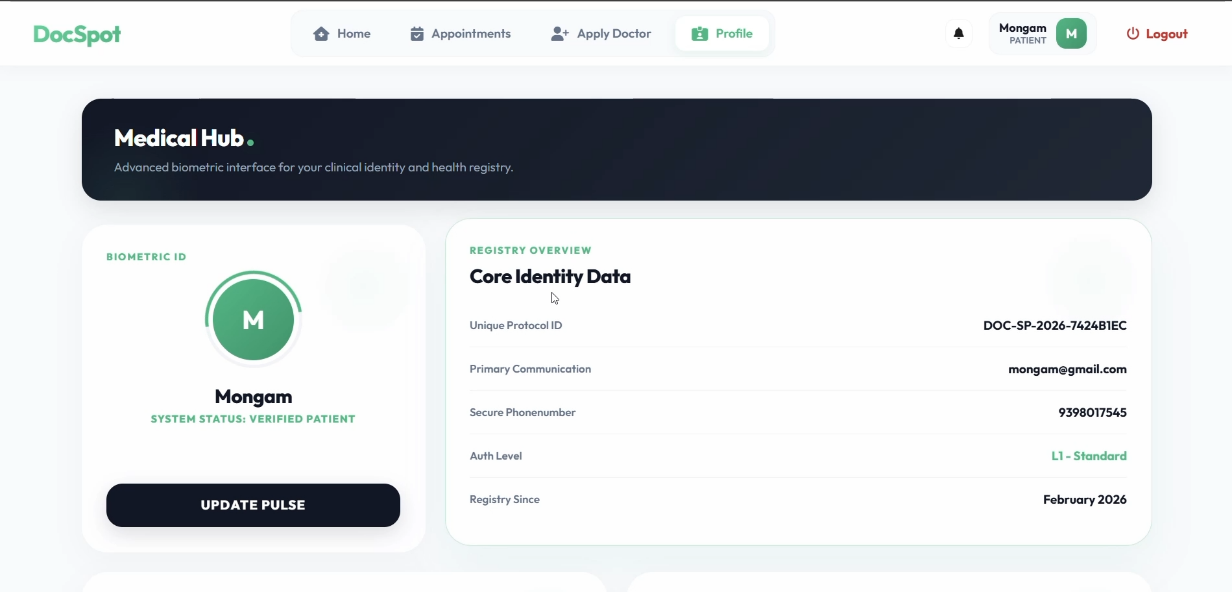
**7.1 Output Screenshots:**

****

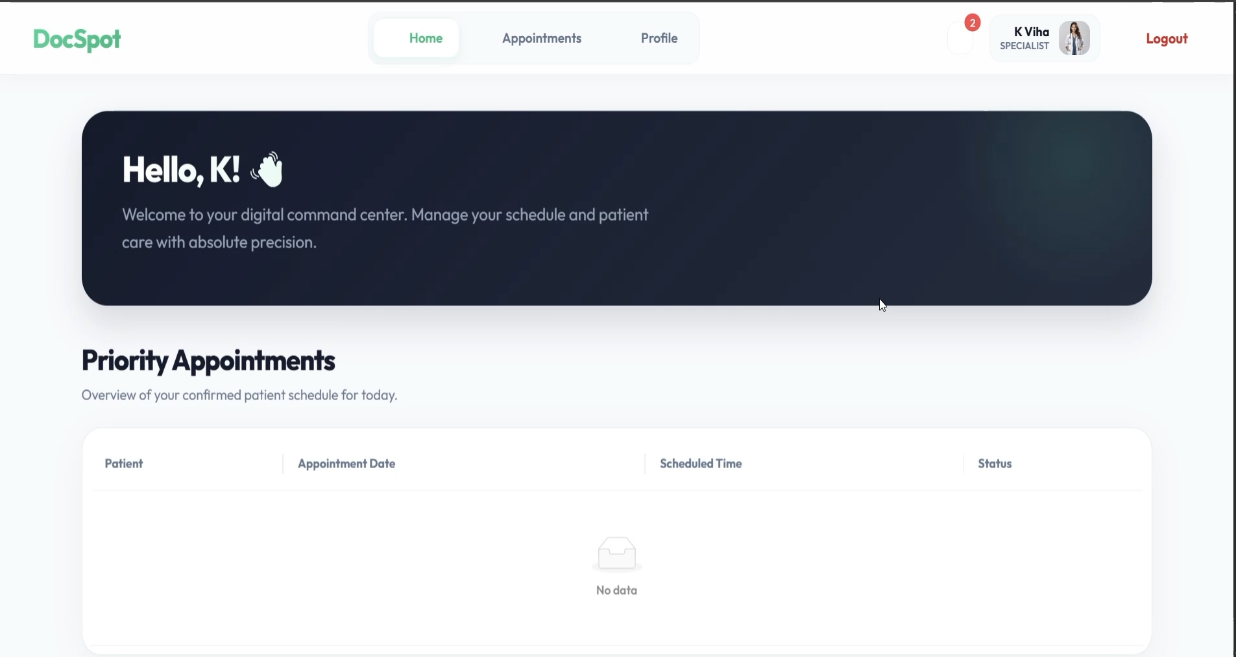
*Figure-2: User Dashboard*

****

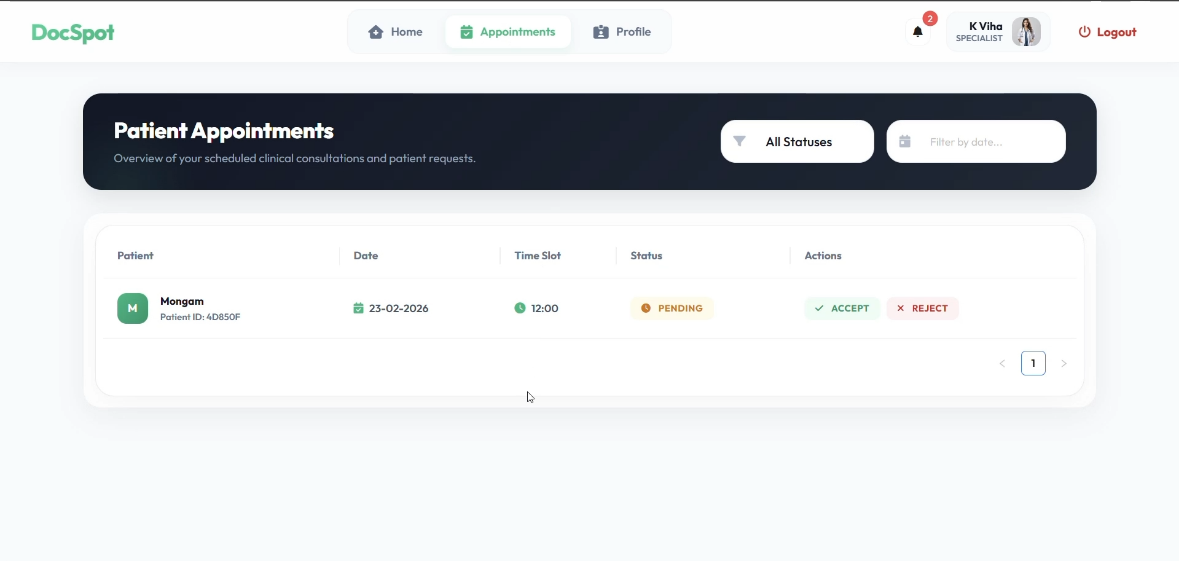
*Figure-3: Appointment History*



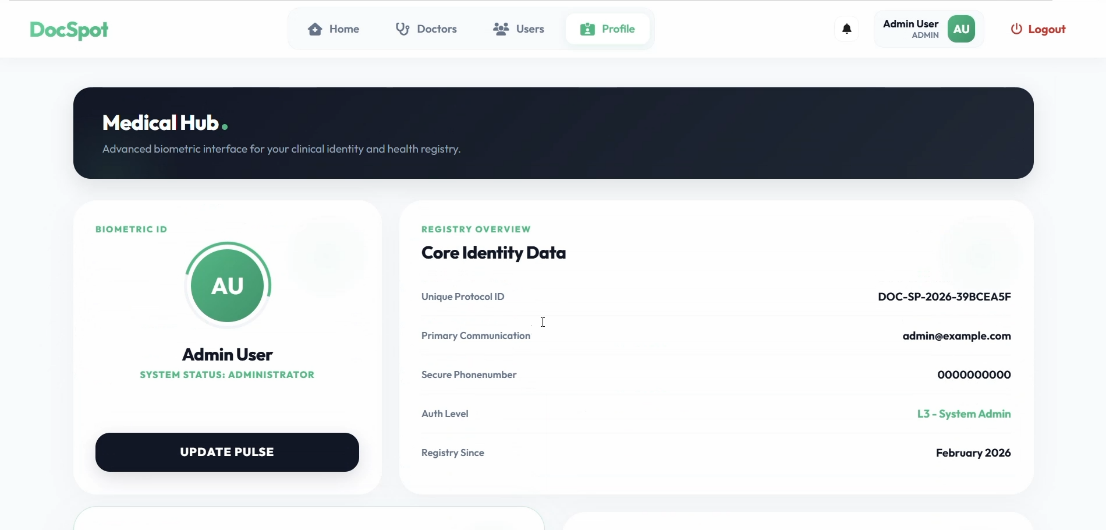
*Figure-4: User Profile*



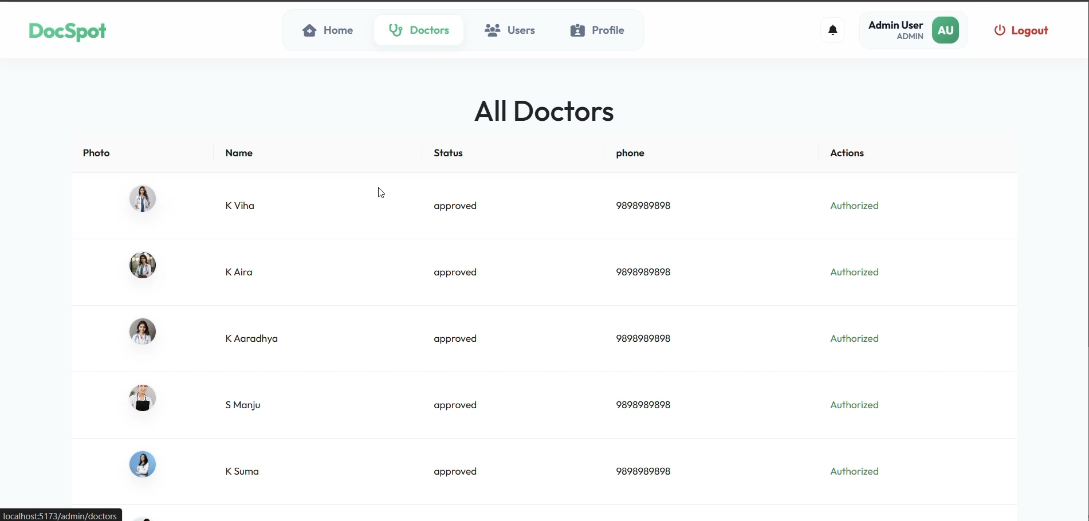
*Figure-5: Doctor Dashboard*



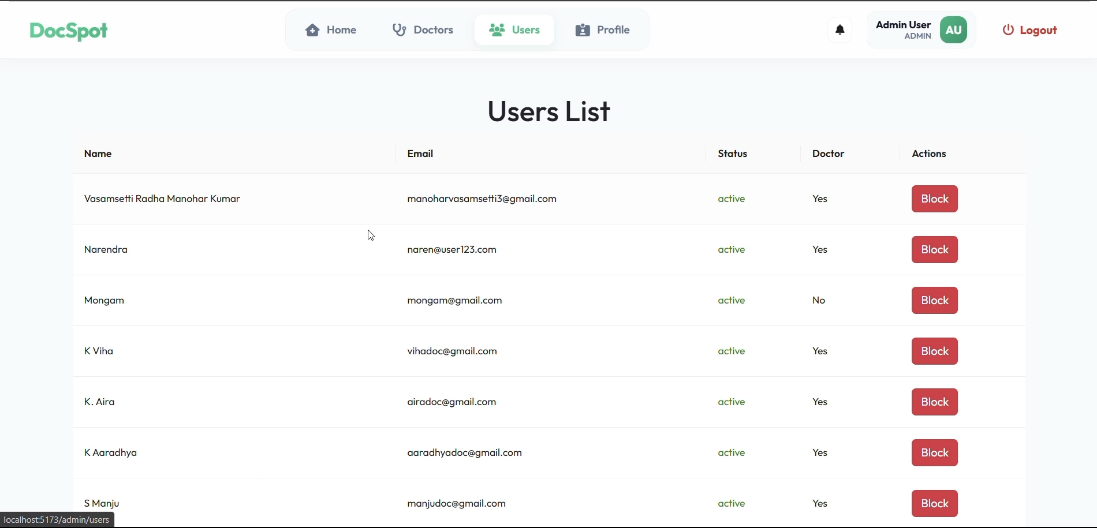
*Figure-6: Doctor Dashboard (Appointment History)*



*Figure-7: Admin Profile Page*



*Figure-8: Admin Dashboard*



*Figure-8: Admin Can Block Users & Doctors*

**8. ADVANTAGES & DISADVANTAGES**

**Advantages**

* **Real-Time Appointment Booking:** Patients can view available doctors and book appointments instantly, reducing manual delays.
* **Role-Based Access Control:** Separate dashboards for Admin, Doctor, and User improve workflow management.
* **Cloud Database Integration:** Using MongoDB Atlas ensures scalability, data security, and remote access.
* **Secure Authentication:** JWT-based login protects user data and restricts unauthorized access.
* **Digital Document Management:** Patients can upload medical records during booking, improving consultation efficiency.
* **User-Friendly Interface:** Built with React and modern UI libraries for smooth interaction.

**Disadvantages**

* Requires stable internet connectivity to access the platform.
* File upload performance may depend on network speed.
* Currently supports only web-based access (no mobile app).
* Real-time notifications may require further optimization.

**9. CONCLUSION**

The DocSpot Doctor Appointment Booking Platform successfully addresses the challenges of traditional healthcare scheduling by providing a centralized digital solution. Through the use of modern web technologies such as React.js, Node.js, Express.js, and MongoDB Atlas, the system enables seamless interaction between patients, doctors, and administrators.

The platform improves healthcare accessibility by reducing waiting time, enabling efficient appointment management, and ensuring secure handling of medical data. The project demonstrates practical implementation of full-stack development concepts, RESTful APIs, authentication mechanisms, and cloud database integration.

**10. FUTURE SCOPE**

* Integration of video consultation for telemedicine services.
* Mobile application development using React Native.
* AI-based doctor recommendation system.
* Real-time push notifications using WebSockets or Firebase.
* Payment gateway integration for online consultation fees.
* Advanced analytics dashboard for doctors and administrators.
* Automated appointment reminders via SMS or Email.

1. **APPENDIX**

**GitHub Repository:**

<https://github.com/mano-520/DocSpot-Seamless-Appointment-Booking-for-Health>

**Live Demo / Project Video:**

<https://drive.google.com/file/d/1RgBjcnsiStY2jNRKhx9suPu5UYB9TavD/view?usp=sharing>