**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.

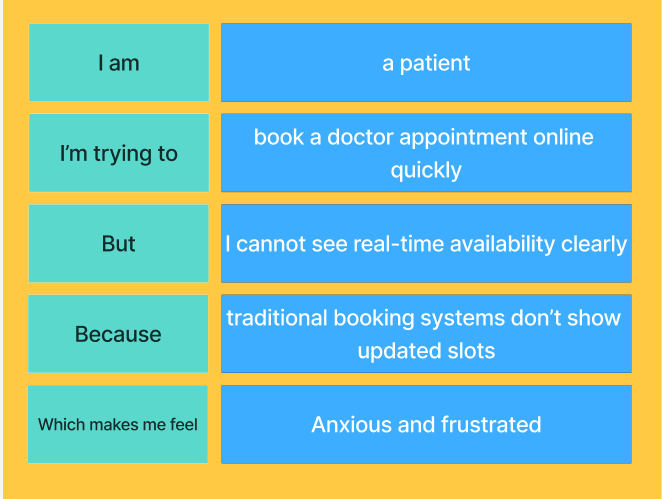
Overall, the project aims to bridge the gap between patients and healthcare services by offering a reliable and structured appointment management solution that improves user experience and operational efficiency.

**2.IDEATION PHASE**

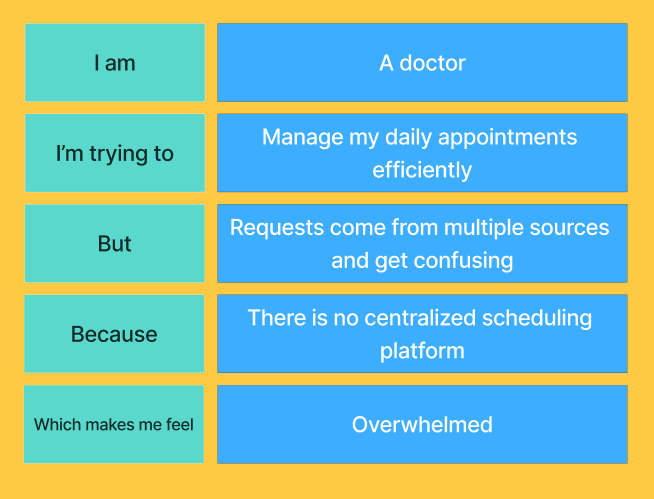
**2.1 Problem Statement:**

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:**



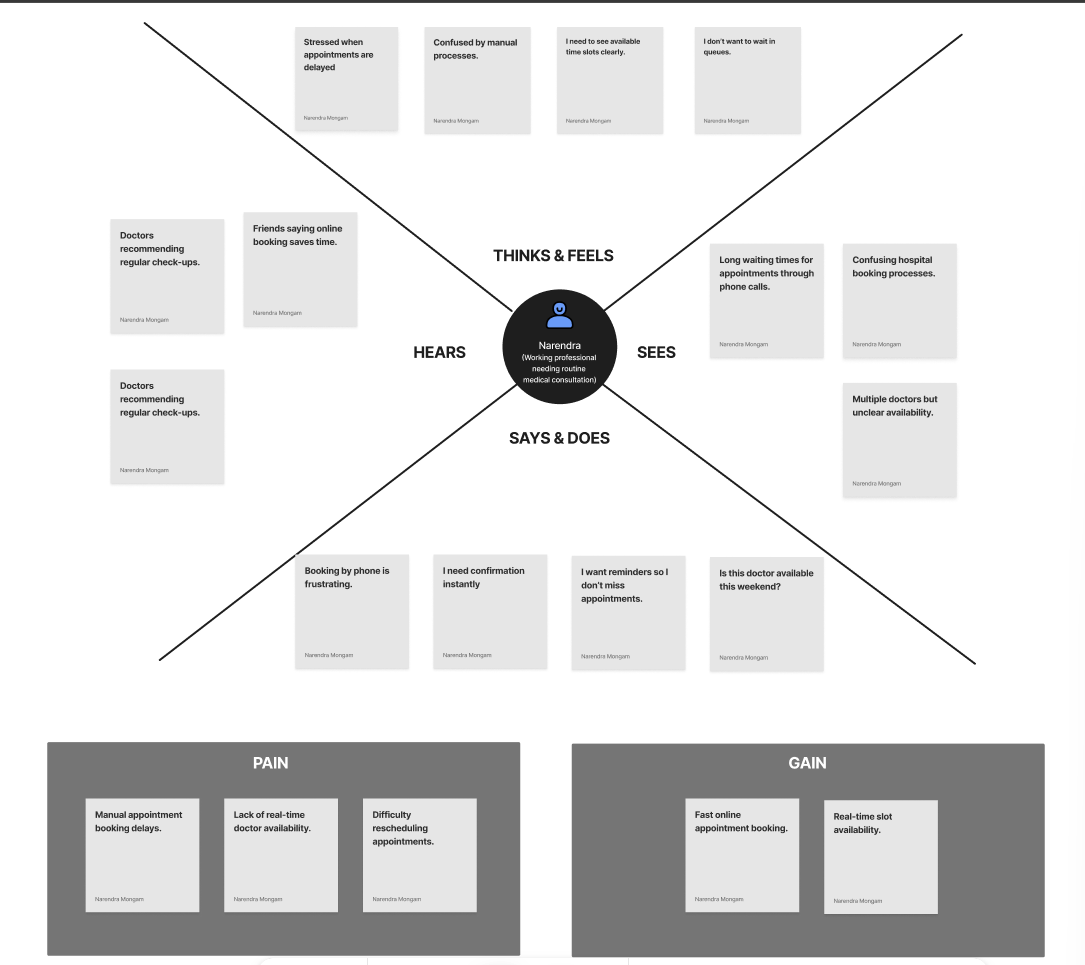
**Problem** **Statement -2:**

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Statement (PS)** | **I am** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | A patient | Book a doctor appointment online quickly | I cannot see real time availability clearly | Traditional booking systems don’t show updated slots | Anxious and frustrated |
| PS-2 | A doctor | Manage my daily appointments efficiently | Requests come from multiple sources and get confusing | There is no centralized scheduling platform | Overwhelmed |

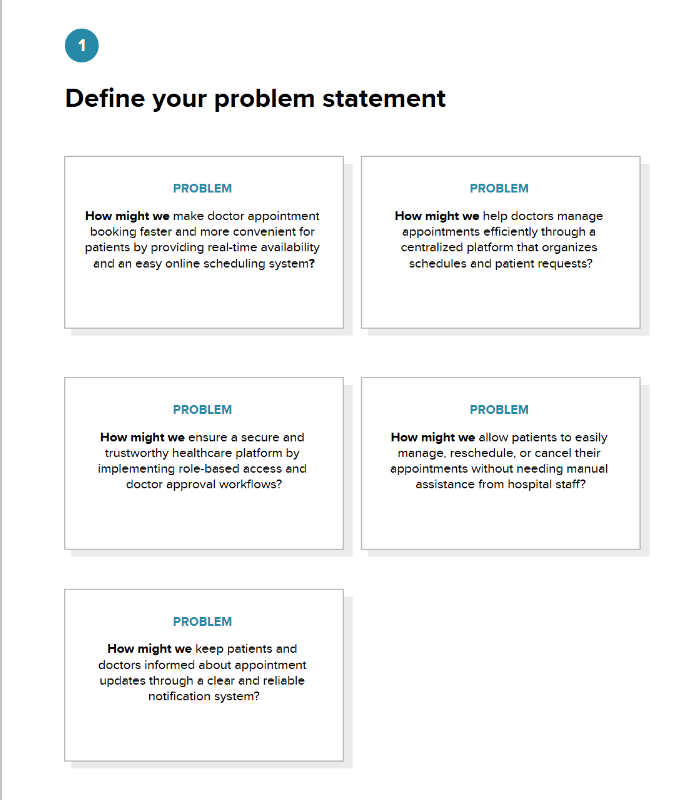
**2.2 Empathy Map:**

**User: - Katla Karthikeya** (Working Professional Needs Routine Medical Consultation)

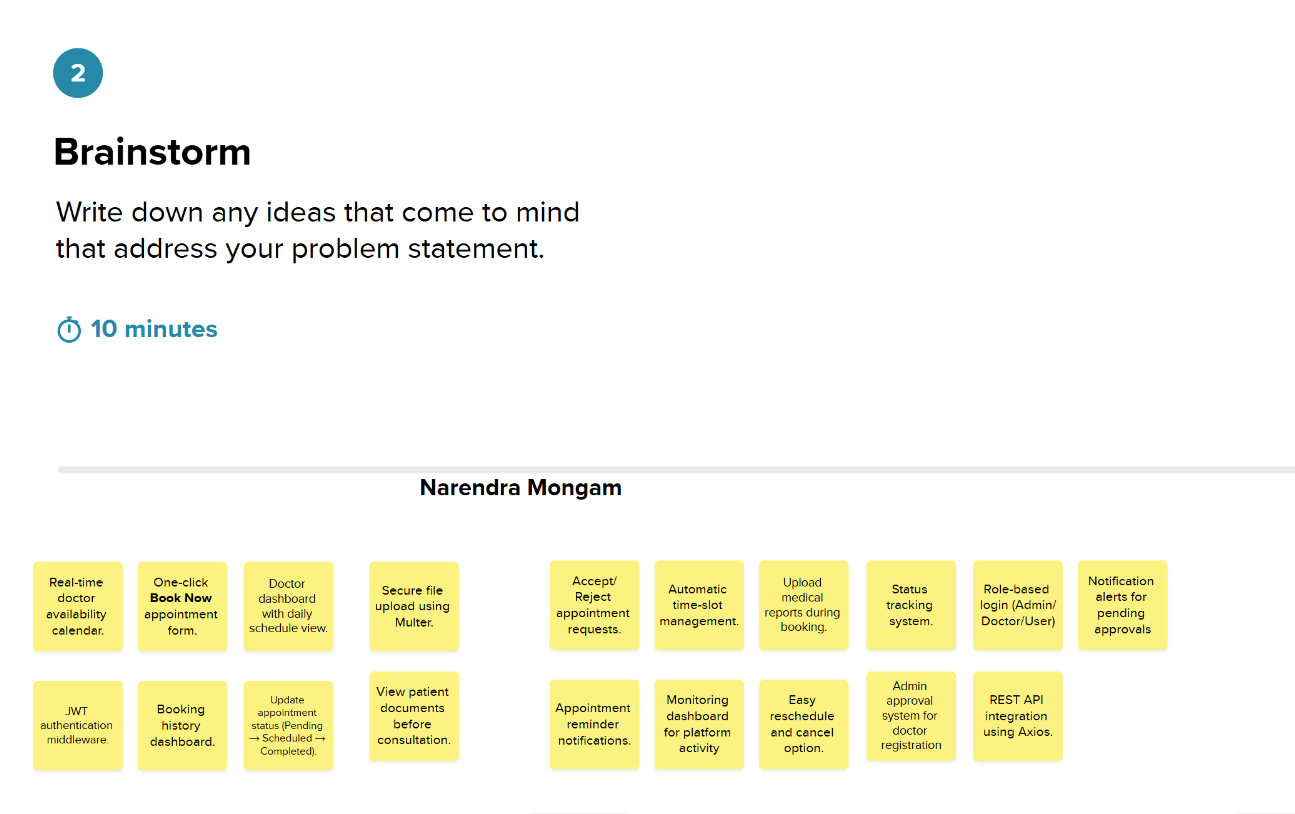


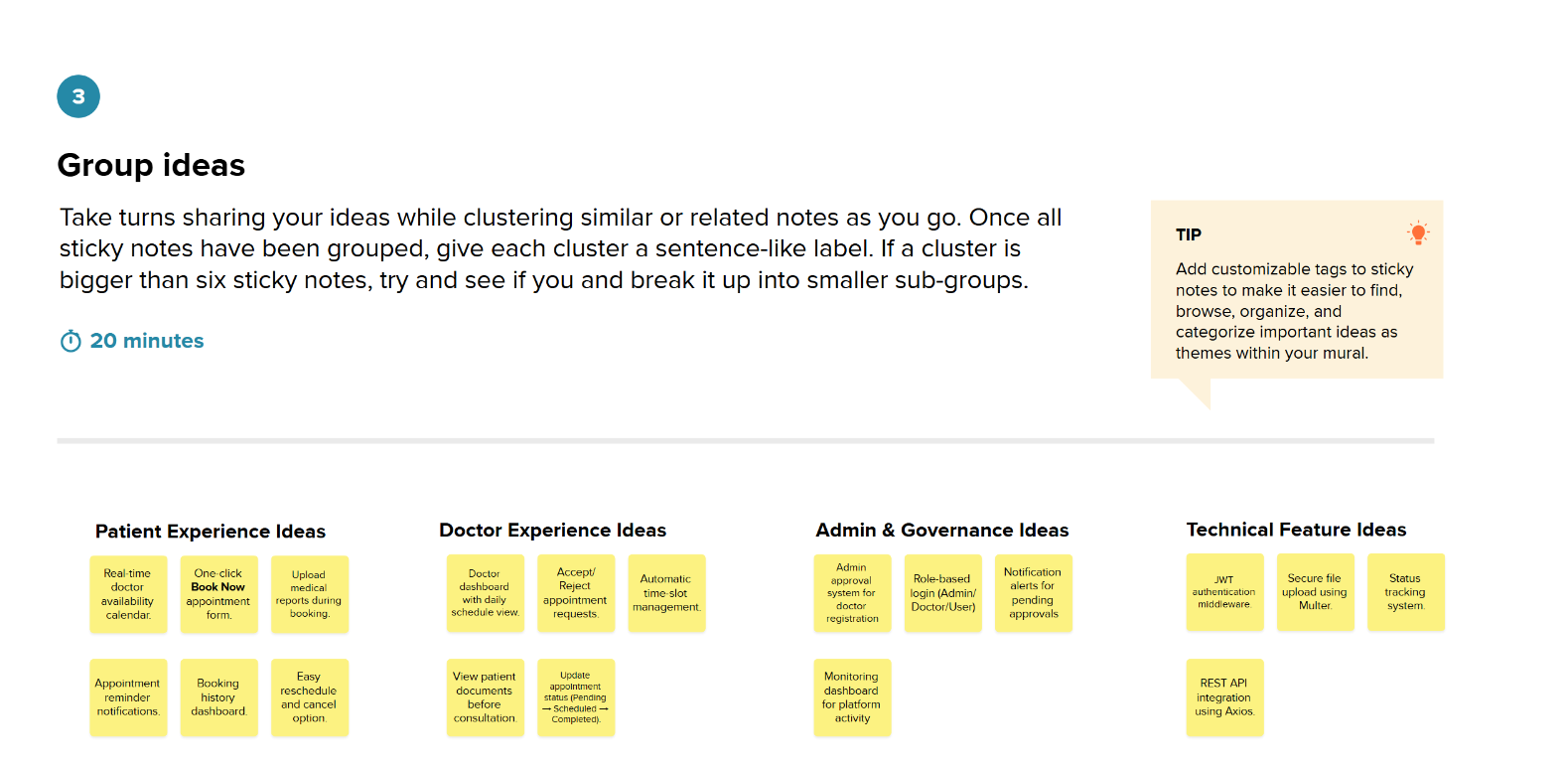
**2.2 Brainstorm & Idea Prioritization:-**

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



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

****

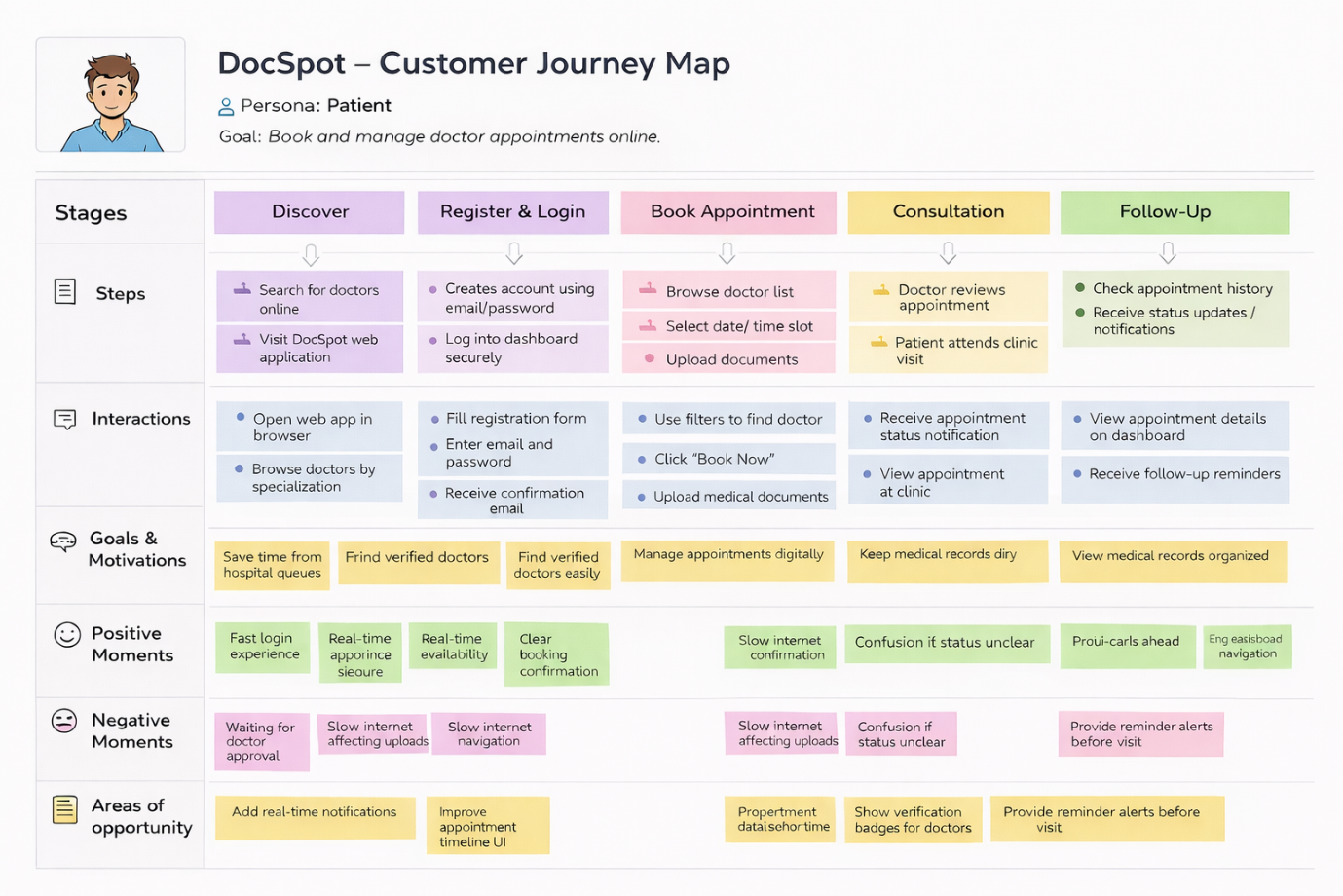
****

**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.

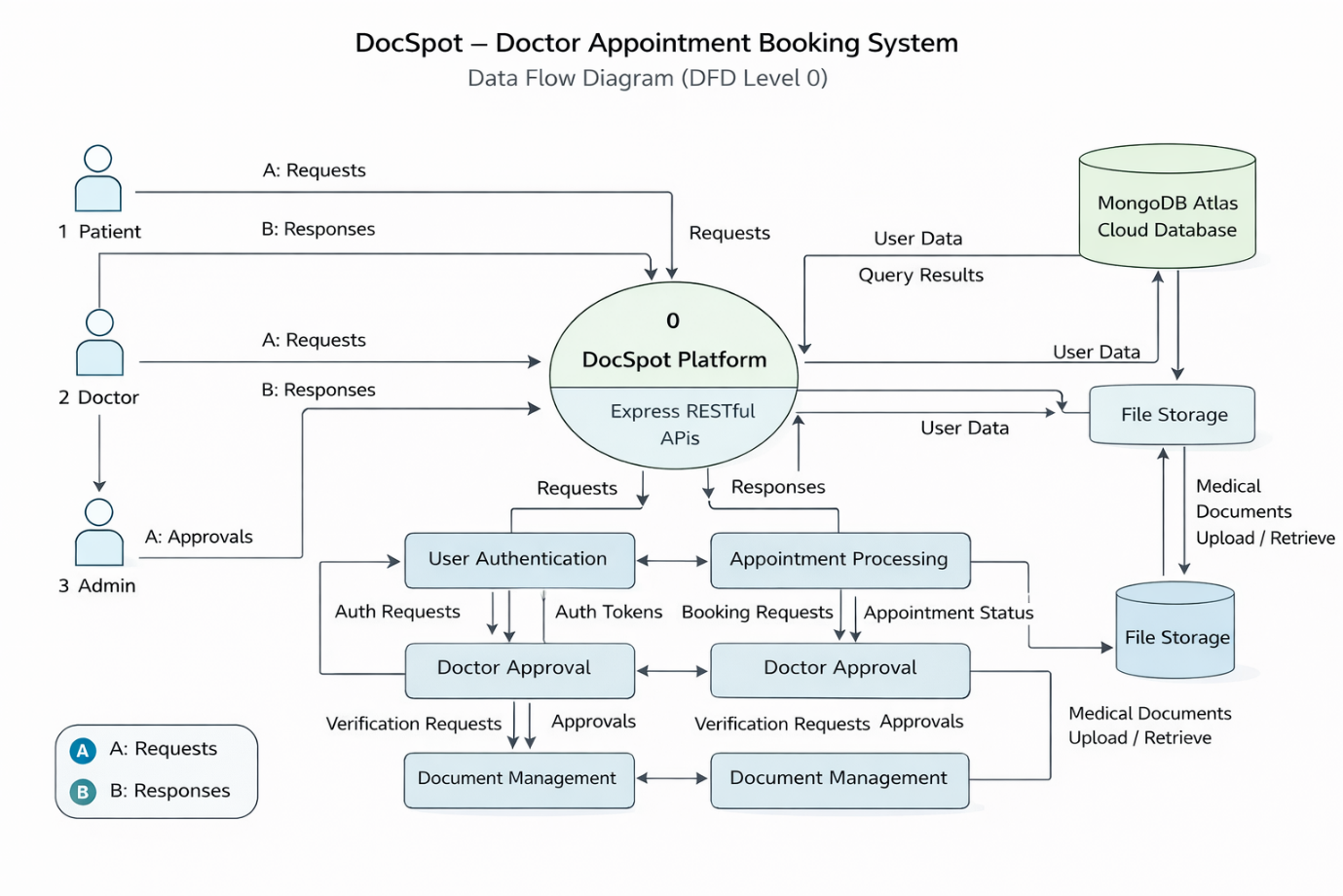
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Form  Role selection (User / Doctor)  Secure password creation |
| FR-2 | User Authentication | Login using Email & Password JWT-based authentication |
| 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 booking 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 | User receives confirmation updates  Appointment status notifications |

**Non-functional Requirements:**

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

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | Simple and user-friendly interface allowing patients to book appointments easily without technical knowledge. |
| NFR-2 | **Security** | JWT authentication, encrypted passwords, and secure document upload to protect user data and privacy. |
| NFR-3 | **Reliability** | System ensures accurate appointment scheduling and consistent status updates without data loss. |
| NFR-4 | **Performance** | Fast loading of dashboards and real-time booking actions through efficient API communication. |
| NFR-5 | **Availability** | Platform accessible online anytime for patients, doctors, and admins to manage appointments. |
| NFR-6 | **Scalability** | MongoDB and REST architecture allow future expansion to more users, doctors, and healthcare services. |

**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 | | Account created & dashboard accessible | High | Sprint-1 |
| Customer (Web User) | Authentication | USN-2 | As a user, I can log in securely | Successful login redirects to dashboard | High | Sprint-1 |
| Customer (Web User) | Doctor Listing | USN-3 | As a user, I can view available doctors | Doctor list loads from database | High | Sprint-1 |
| Customer (Web User) | Appointment Booking | USN-4 | As a patient, I can book appointment with date & time | Appointment status shows “Pending” | High | Sprint-2 |
| Customer (Web User) | Document Upload | USN-5 | As a user, I can upload medical documents during booking | File uploads successfully | Medium | Sprint-2 |
| Doctor | Appointment Management | USN-6 | As a doctor, I can accept or reject appointment requests | Status updates to Scheduled/Rejected | High | Sprint-3 |
| Administrator | Doctor Approval | USN-7 | As an admin, I can approve doctor registrations | Approved doctor visible in system | High | Sprint-3 |
| Customer Care Executive | Notifications | USN-8 | As a user, I receive appointment updates | Notification displayed in dashboard | Medium | Sprint-4 |
| Customer (Web User) | Appointment History | USN-9 | As a user, I can view booking history | Previous appointments visible | 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:**

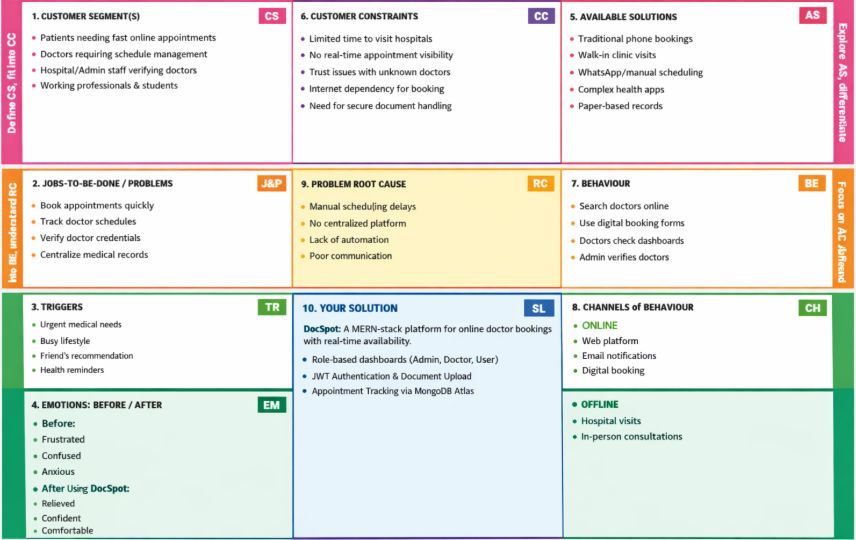
|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Component** | **Description** | **Technology** |
|  | User Interface | Web interface for patients, doctors, and admin dashboards | React.js, HTML, CSS, Bootstrap, Material UI |
|  | Application Logic-1 | Authentication & Role Management | Node.js, Express.js, JWT |
|  | Application Logic-2 | Appointment Booking & Scheduling | Express.js REST APIs |
|  | Application Logic-3 | Notification & Status Management | Node.js Controllers |
|  | Database | Stores users, doctors, appointments data | MongoDB, Mongoose |
|  | Cloud Database | Cloud-hosted NoSQL database for storing users, doctors, appointments, and notifications with remote access and scalability | MongoDB Atlas |
|  | File Storage | Medical document upload & storage | IBM Block Storage or Other Storage Service or Local Filesystem |
|  | External API-1 | |  | | --- | |  |  |  | | --- | | HTTP communication between frontend & backend | | Axios |
|  | External API-2 | Not Applicable | \_\_ |
|  | Machine Learning Model | Not used in this project | \_\_ |
|  | Infrastructure (Server / Cloud) | Local development deployment | Node.js Local Server |

**Table-2: Application Characteristics:**

| **S. No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Frameworks used to build UI & Backend APIs | React.js, Express.js, Node.js |
|  | Security Implementations | Role-based authentication, encrypted passwords, protected routes | JWT, bcryptjs, Middleware |
|  | Scalable Architecture | 3-Tier architecture separating UI, backend logic, and database | REST Architecture, MongoDB |
|  | Availability | |  | | --- | |  |  |  | | --- | | Web application accessible anytime through browser | | Node.js Server |
|  | Performance | Fast API responses and asynchronous communication | Axios, Express.js |

**4. PROJECT DESIGN**

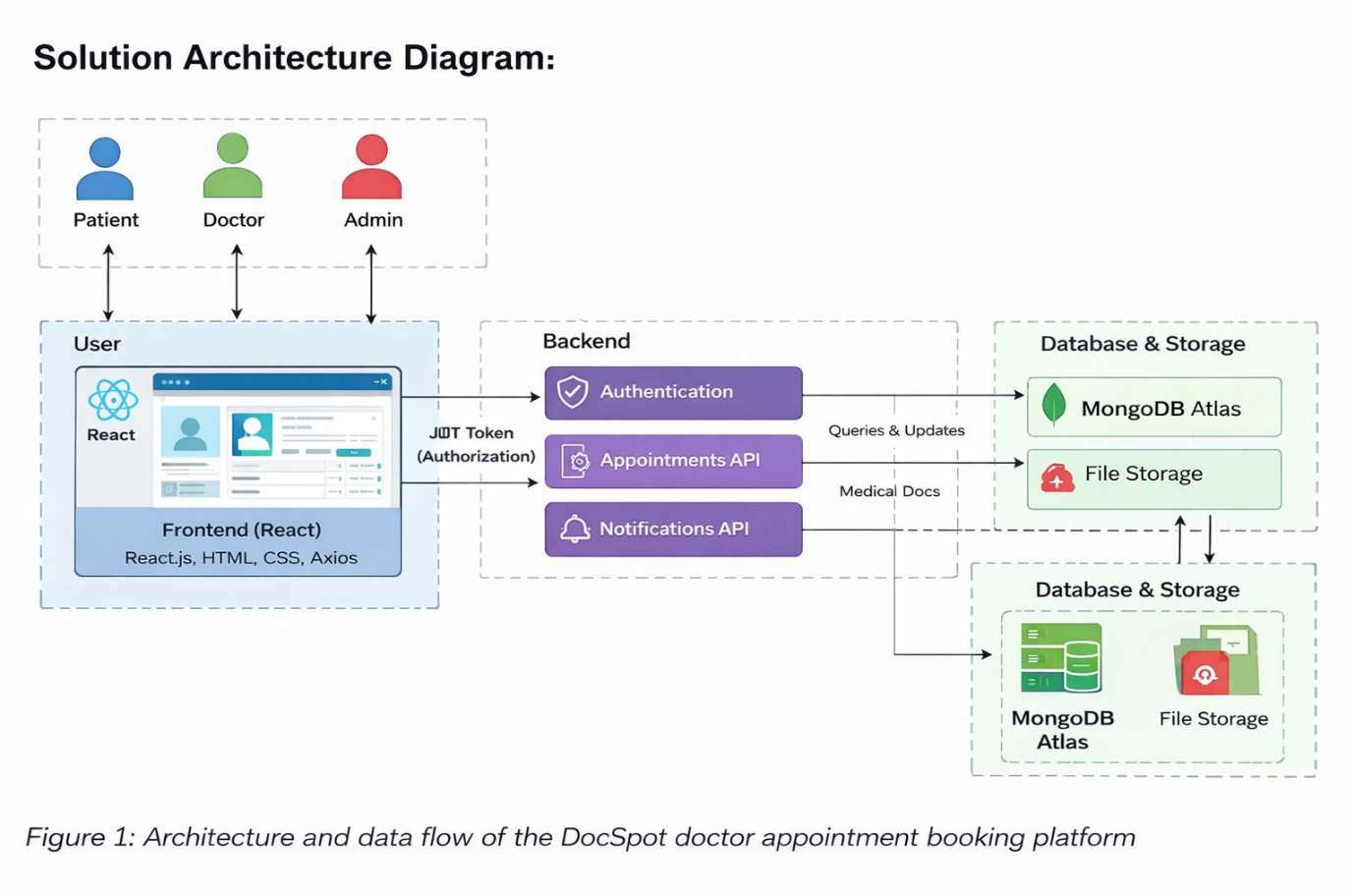
**4.1 Problem Solution Fit:**

****

**4.2 Proposed Solution:**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
|  | Problem Statement (Problem to be solved) | Patients face difficulty booking doctor appointments due to manual scheduling, lack of real-time availability, and long waiting times. Doctors need a centralized platform to manage appointments efficiently, while admins require a secure system to verify doctors and maintain platform trust. |
|  | Idea / Solution description | DocSpot is a MERN-stack web application that enables patients to book appointments online through a centralized system. It provides role-based dashboards for Admin, Doctor, and User, real-time booking, secure JWT authentication, document upload, and appointment status tracking using REST APIs connected to MongoDB Atlas. |
|  | Novelty / Uniqueness | Role-based healthcare workflow, admin-controlled doctor approval, real-time appointment tracking, secure medical document upload, and a simplified user-friendly booking interface. |
|  | Social Impact / Customer Satisfaction | Reduces waiting time, improves healthcare accessibility, ensures verified doctor interactions, and enhances patient convenience through digital appointment management. |
|  | Business Model (Revenue Model) | Commission-based booking model, subscription plans for doctors, premium doctor listing, and future telemedicine consultation services. |
|  | Scalability of the Solution | Built using 3-tier architecture (React frontend, Express backend, MongoDB Atlas cloud database), enabling scalable performance and easy expansion to support more users and healthcare services. |

**4.3 Solution Architecture:**

****

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

**5. PROJECT PLANNING & SCHEDULING**

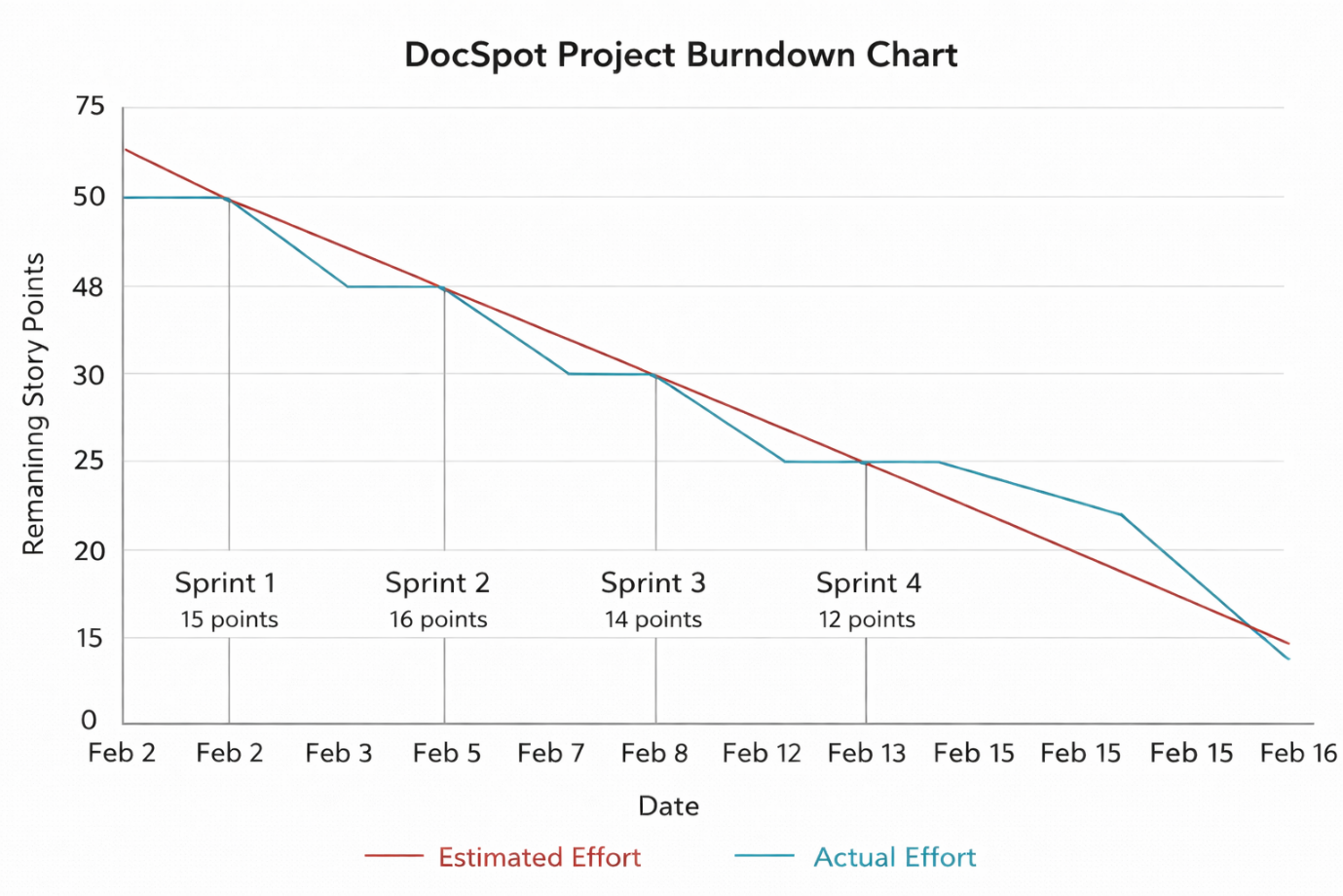
**5.1 Project Planning:**

**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 | | 2 | High | Team |
| Sprint-1 | Authentication | USN-2 | As a user, I can log in securely using JWT authentication | 2 | High | Team |
| Sprint-1 | Dashboard | USN-3 | As a user, I can view doctor list on dashboard | 3 | High | Team |
| Sprint-2 | Appointment Booking | USN-4 | |  | | --- | |  |  |  | | --- | | As a patient, I can book an appointment with date & time | | 5 | High | Team |
| Sprint-2 | Document Upload | USN-5 | As a user, I can upload medical documents during booking | 3 | Medium | Team |
| Sprint-3 | Doctor Panel | USN-6 | As a doctor, I can accept or reject appointment requests | 5 | High | Team |
| Sprint-3 | Admin Panel | USN-7 | As an admin, I can approve doctor registrations | 4 | High | Team |
| Sprint-4 | Notifications | USN-8 | As a user, I receive appointment status updates | 3 | Medium | Team |
| Sprint-4 | Appointment History | USN-9 | As a user, I can view and manage booking history | 3 | 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 | 15 | 4 Days | 02 Feb 2026 | 05 Feb 2026 | 15 | 05 Feb 2026 |
| Sprint-2 | 16 | 4 Days | 06 Feb 2026 | 09 Feb 2026 | 16 | 09 Feb 2026 |
| Sprint-3 | 14 | 4 Days | 10 Feb 2026 | 13 Feb 2026 | 14 | 13 Feb 2026 |
| Sprint-4 | 12 | 3 Days | 14 Feb 2026 | 16 Feb 2026 | 12 | 16 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: 13 Feb 2025 to 16 Feb 2025

**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 Login
* USN-4 Appointment Booking
* USN-6 Doctor Approval
* USN-8 Notifications

**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 |

**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:**

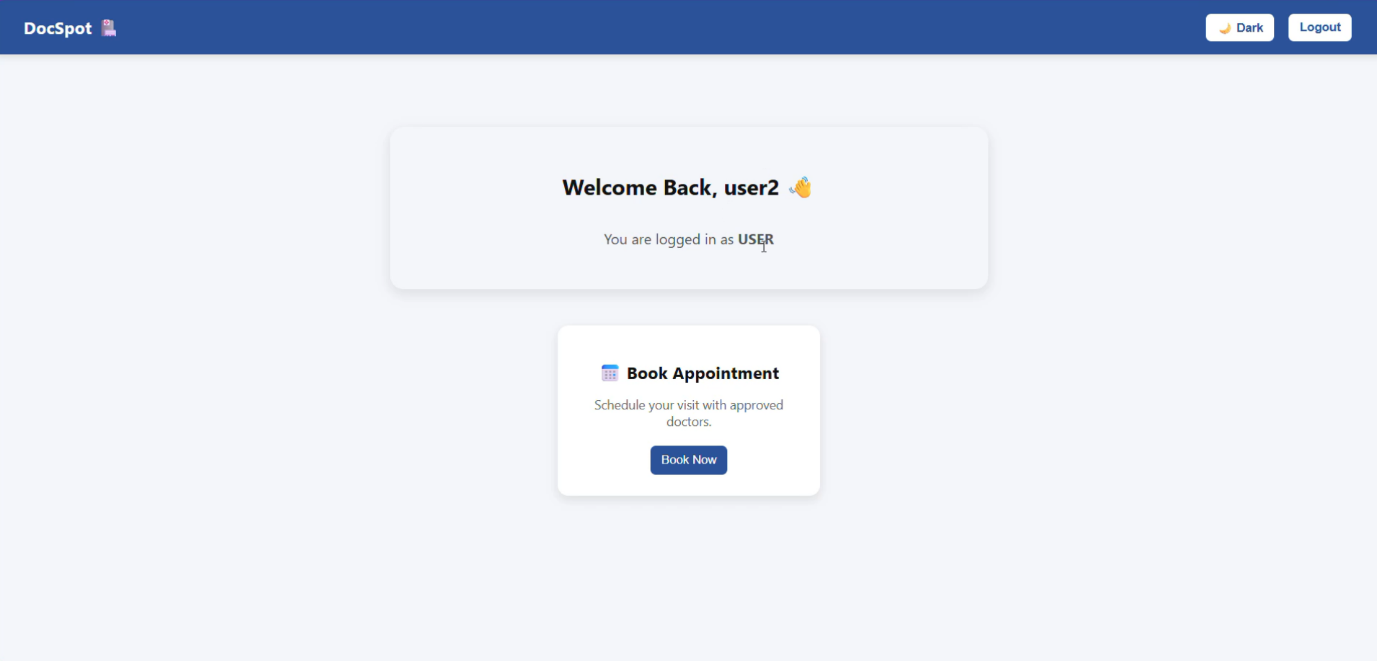
Tester Name: Katla Karthikeya

Date: 16 February 2026

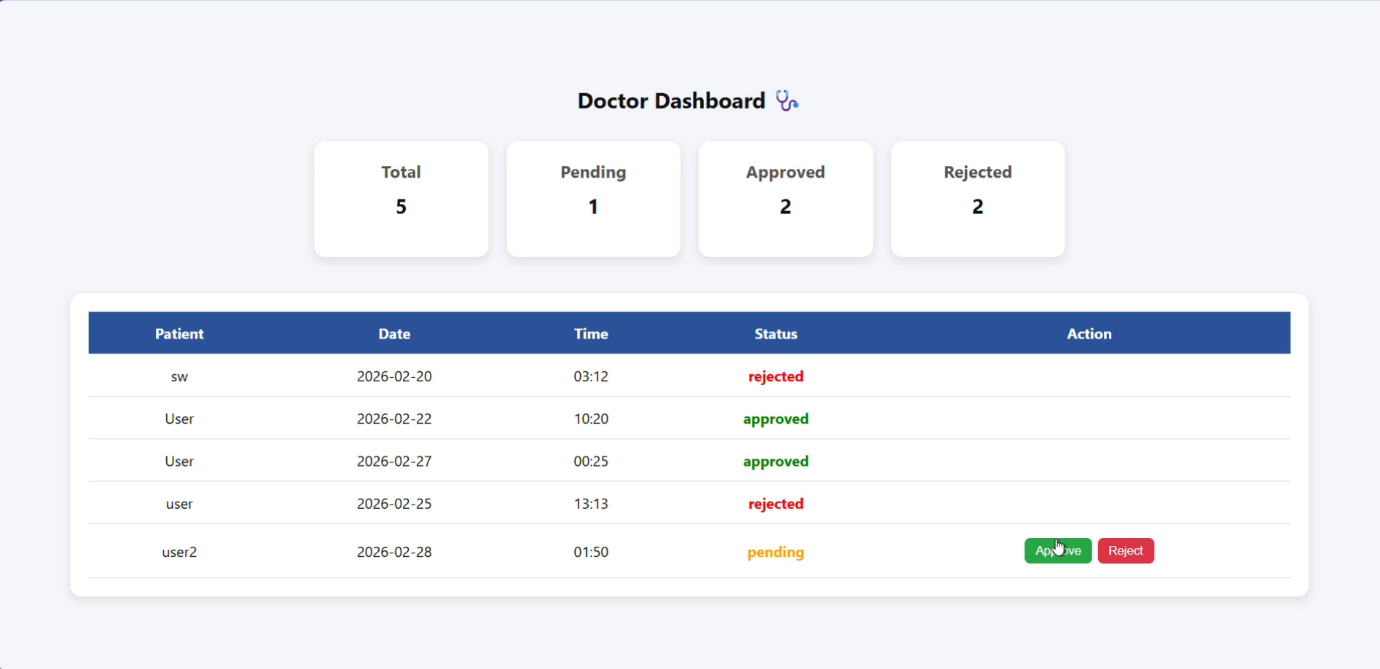
Signature: Mongam Narendra

**7. RESULTS**

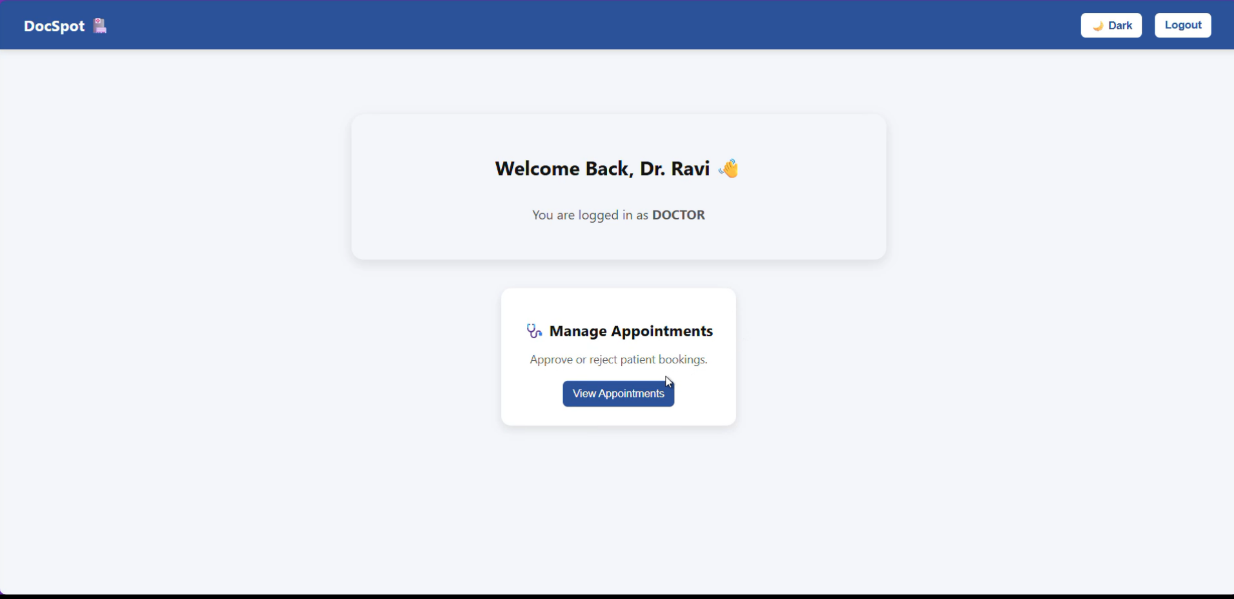
**7.1 Output Screenshots:**

****

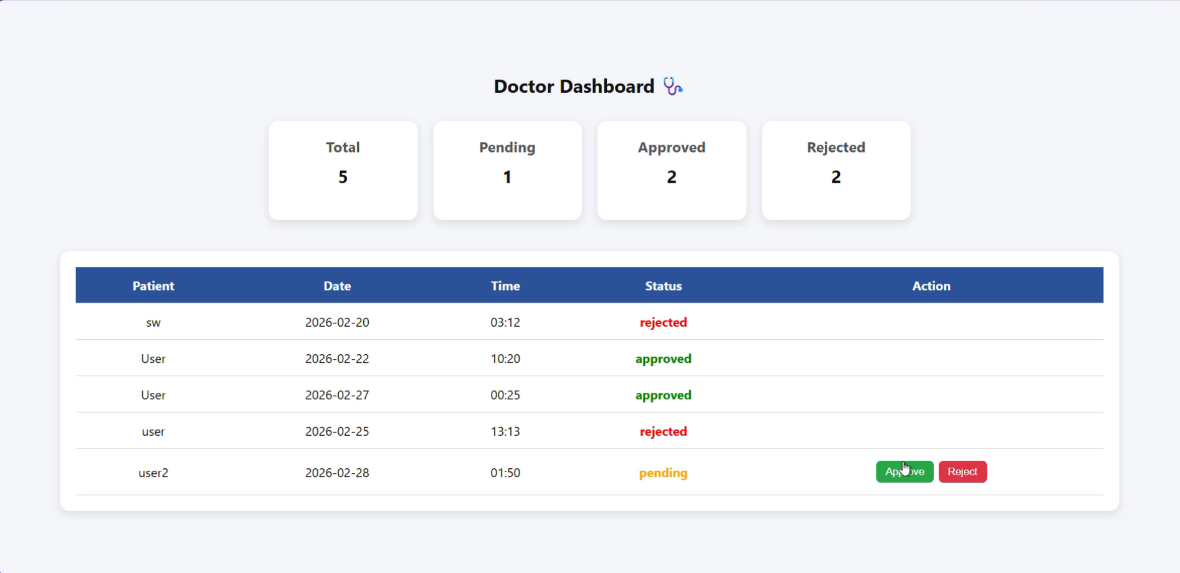
*Figure-2: User Dashboard*

****

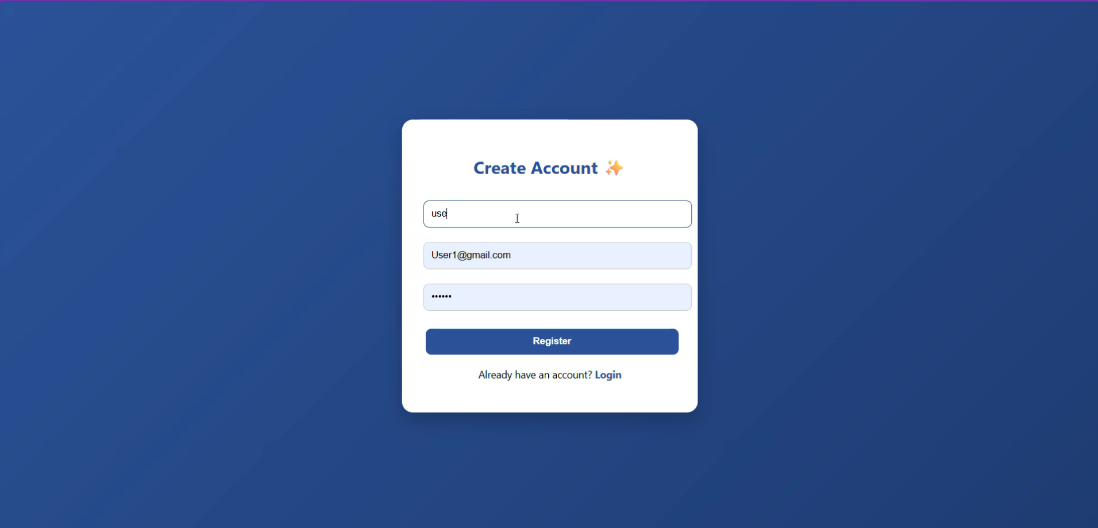
*Figure-3: Appointment History*



*Figure-4: Doctor Dashboard*



*Figure-5: Doctor Dashboard (Appointment History)*



*Figure-6: Registration Page*

**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:**

[Karthikeya20-k/DocSpot-MERN-App: Full Stack Doctor Appointment Booking System using MERN Stack](https://github.com/Karthikeya20-k/DocSpot-MERN-App)

**Live Demo / Project Video:**

<https://drive.google.com/drive/u/1/folders/1JiTZsbDeB8zd5TsV4c7uF9qgE01o-r17>