Full Stack Development with MERN

DocSpot: Seamless Appointment Booking for Health

Team ID: LTVIP2026TMIDS56110

Team Size: 4

Team Members and Roles

Name	Role
Balla Sri Divya Satya Joshna	Team Leader / Backend Architect
Dola Gowthami	Member / Frontend Developer
Bhargavi Satya Seetha Naga Lakshmi Nalam	Member / Full Stack Integration
Chillara Venkata Ramakrishna	Member / Database & Testing

1. INTRODUCTION

1.1 Project Overview

DocSpot is a web-based online doctor appointment booking system designed to simplify the process of scheduling medical consultations. The system allows patients to register, search for doctors, check availability, book appointments, and make online payments. Doctors can manage their schedules, and administrators can monitor and manage the entire system.
The application is built using the MERN stack (MongoDB, Express.js, React.js, Node.js) and follows a 3-tier cloud-based architecture.

1.2 Purpose

The purpose of this project is to:

- Reduce waiting time in hospitals
- Provide real-time doctor availability
- Enable online appointment booking
- Improve healthcare accessibility
- Digitize hospital appointment systems

2. IDEATION PHASE

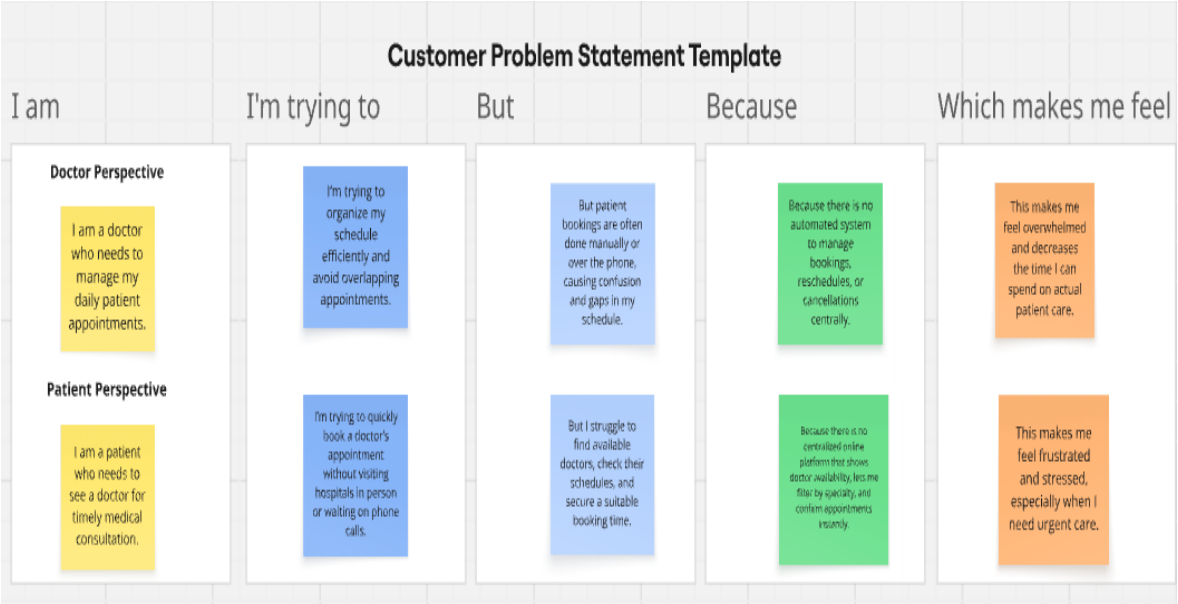
2.1 Problem Statement

Patients face difficulty in booking doctor appointments due to long queues, lack of real-time availability information, manual scheduling systems, and inefficient communication. There is a need for a digital platform that allows easy appointment booking and management.

In the ideation phase of the DocSpot project, the team focused on understanding the real challenges faced by patients and doctors in the current healthcare appointment system. Instead of directly jumping into solution development, we first identified and clearly defined the customer problems from different perspectives. This helped us empathize with users and design a solution that truly addresses their needs.

The primary users of the DocSpot platform are patients who require quick and easy access to medical consultation. Many patients face difficulties such as long waiting times, manual booking processes, lack of transparency in doctor availability, and confusion regarding appointment schedules. These issues create stress, especially during emergency situations.

Similarly, doctors and hospital administrators also experience challenges in managing appointments efficiently. Manual booking systems often result in scheduling conflicts, overcrowded waiting rooms, and missed appointments. Without a centralized digital system, it becomes difficult to track patient data and manage daily operations smoothly.



Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	a patient who needs medical consultation.	book an appointment with a suitable doctor quickly and easily.	I have to visit hospitals physically or call multiple clinics to check availability, which takes a lot of time and effort.	there is no centralized digital platform where I can view doctor details, availability, and book appointments instantly.	frustrated, stressed, and inconvenienced, especially during urgent medical situations.
PS-2	a doctor managing daily patient appointments.	organize my schedule efficiently and reduce overcrowding in the waiting area.	appointments are often booked manually, leading to scheduling conflicts, missed appointments, and long waiting times.	there is no proper digital system to manage bookings and patient records systematically.	overwhelmed and reduces the overall efficiency of my practice.
PS-3	a hospital administrator responsible for managing patient flow.	ensure smooth appointment scheduling and better service management.	manual systems create confusion, data mismanagement, and poor tracking of appointments.	there is no centralized, automated booking and management platform.	that operational efficiency and patient satisfaction are being negatively affected.

2.2 Empathy Map Canvas

Says:

- “I don’t want to wait in long queues.”
- “I need a quick appointment.”

Thinks:

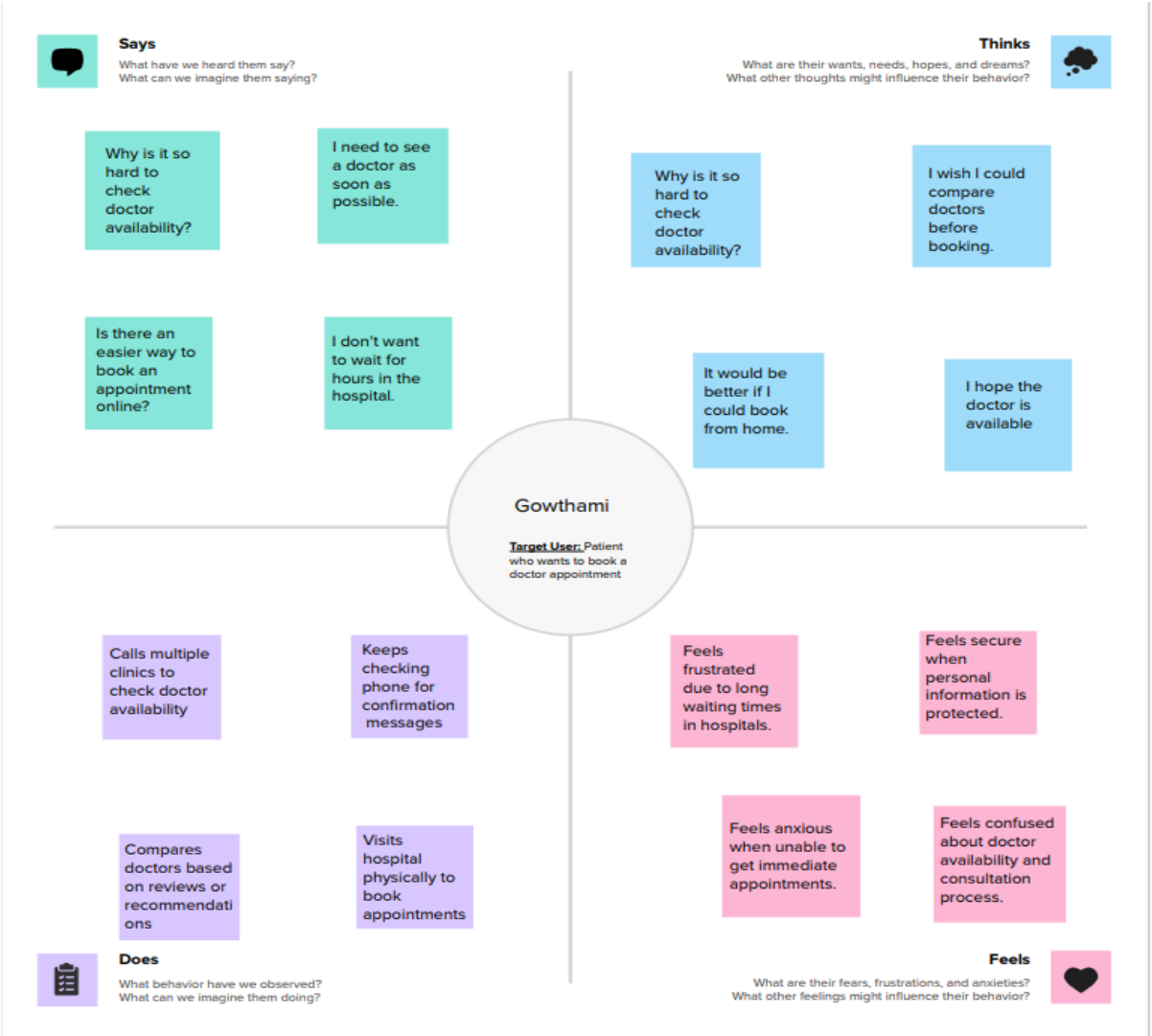
- “Is the doctor available?”
- “Will I get confirmation?”

Does:

- Calls hospital
- Visits hospital physically
- Searches online

Feels:

- Frustrated
- Confused
- Anxious when appointment is delayed



2.3 Brainstorming

Ideas generated:

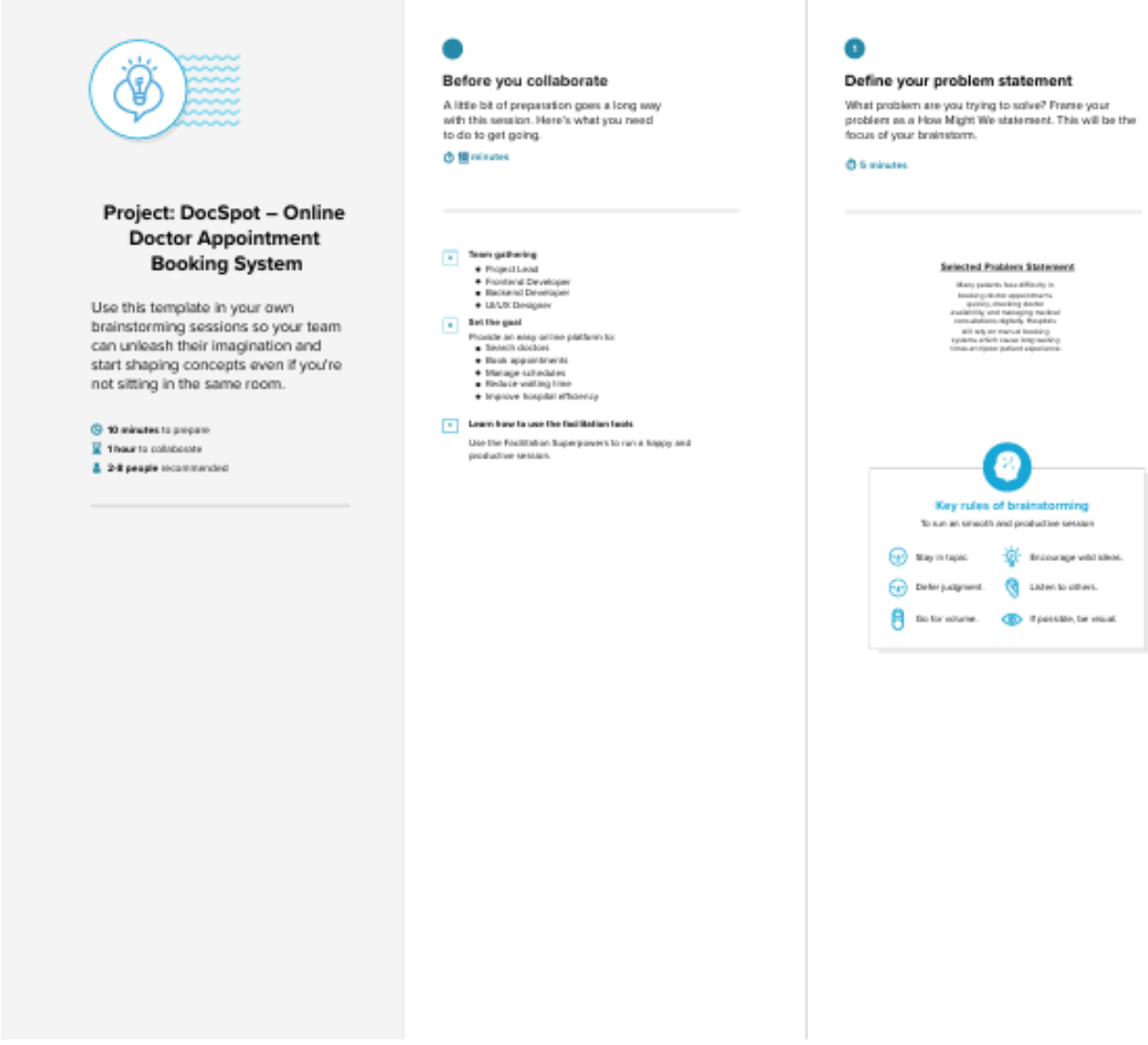
- Online doctor booking system
- Real-time slot availability
- Online payment system
- SMS/Email notification
- Teleconsultation option
- AI-based doctor recommendation

Final idea selected:

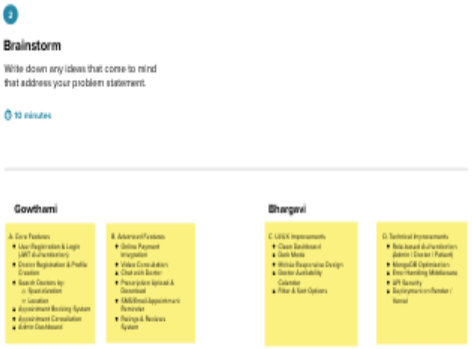
✓ Web-based Online Doctor Appointment System (DocSpot)

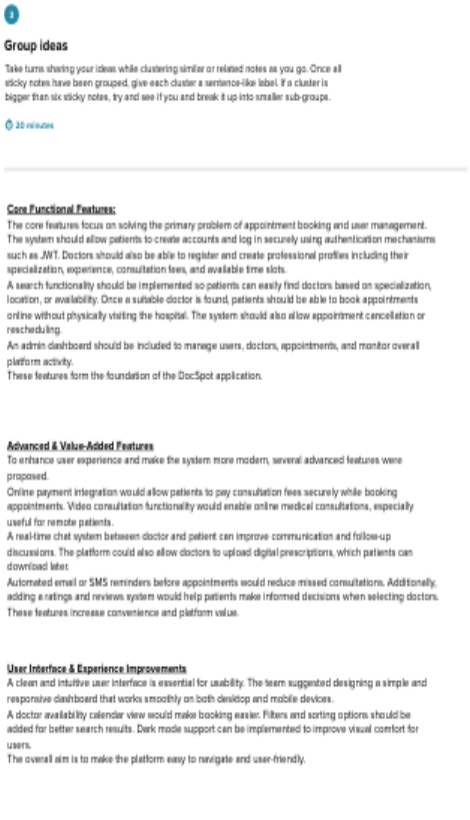
Step-1: Team Gathering, Collaboration and Select the Problem Statement

Many patients face difficulty in booking doctor appointments quickly, checking doctor availability, and managing medical consultations digitally. Hospitals still rely on manual booking systems which cause long waiting times and poor patient experience.

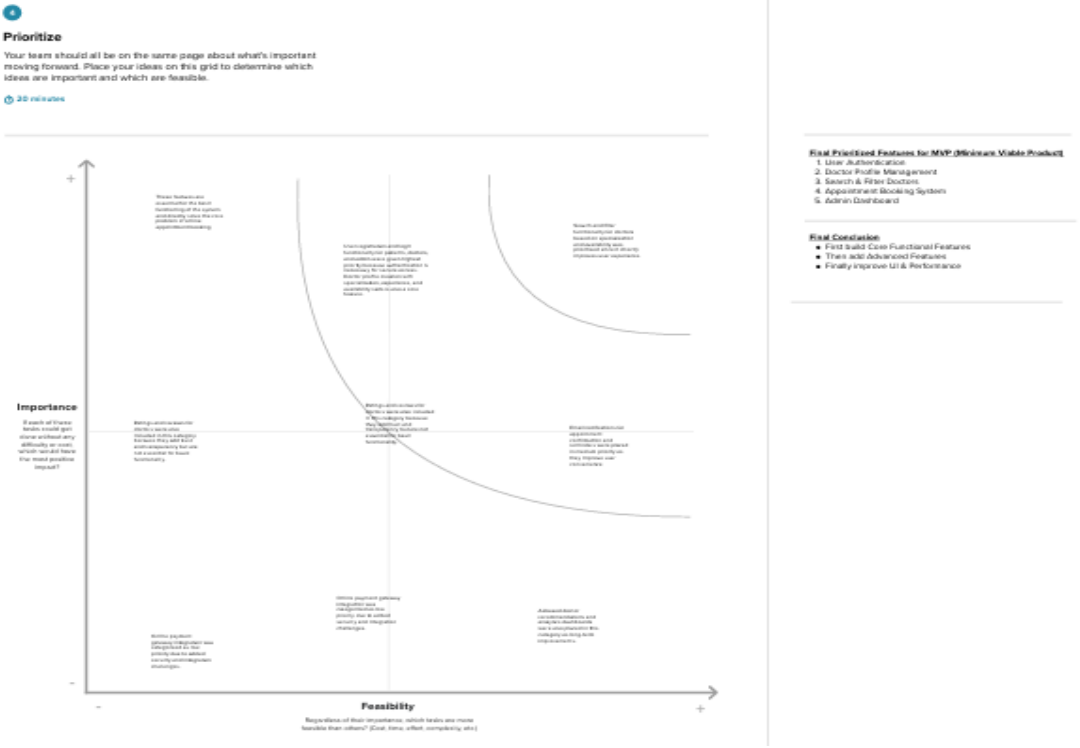


Step-2: Brainstorm, Idea Listing and Grouping





Step-3: Idea Prioritization

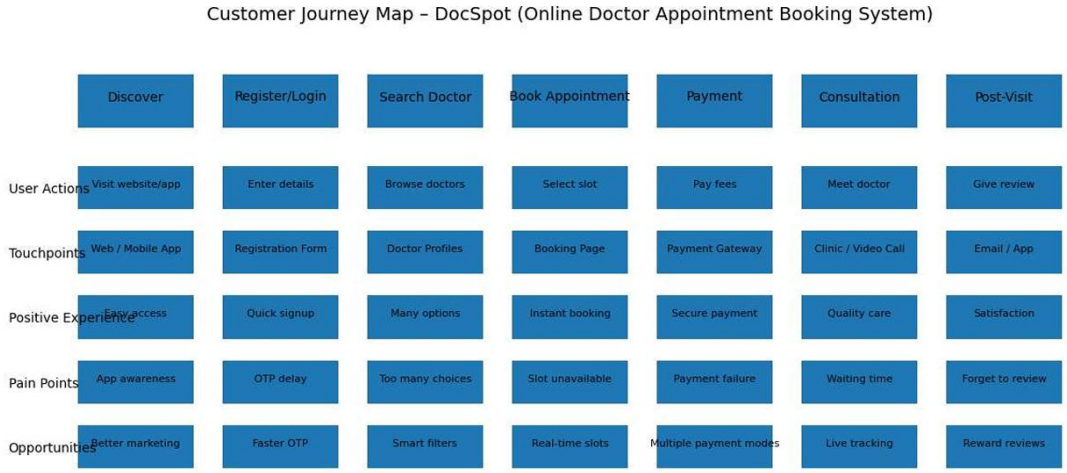


3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

Stages:

1. Discover platform
2. Register/Login
3. Search doctor
4. Select slot
5. Book appointment
6. Make payment
7. Visit doctor
8. Give review



3.2 Solution Requirement

Functional Requirements:

- User Registration & Login
- Doctor Search & Filter
- Appointment Booking
- Appointment Cancellation
- Payment Integration
- Admin Dashboard

Non-Functional Requirements:

- Security (JWT, bcrypt)
- Scalability
- Performance optimization
- Cloud deployment
- 24/7 Availability

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 (Name, Email, Password) Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login & Authentication	Login using Email & PasswordJWT Authentication Password Reset Option
FR-4	Doctor Profile Management	Add Doctor Details (Specialization, Experience, Fees) Update Profile Upload Profile Image
FR-5	Search & Filter Doctors	Search by Specialization Search by Location Filter by Availability
FR-6	Appointment Booking	Select Doctor Choose Date & Time Slot Book Appointment View Booking Confirmation
FR-7	Appointment Management	Cancel Appointment Reschedule Appointment View Appointment History
FR-8	Admin Dashboard	Manage Users Manage Doctors View Appointments Approve/Reject Doctor Profiles

Non-functional Requirements:

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

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	The system should have a simple, user-friendly interface that is easy to navigate for patients and doctors.
NFR-2	Security	User data must be protected using encryption, JWT authentication, and secure password storage.
NFR-3	Reliability	The system should function correctly without failures during booking or login processes.
NFR-4	Performance	The application should load pages within 2–3 seconds and handle multiple users efficiently.
NFR-5	Availability	The system should be accessible 24/7 with minimal downtime.
NFR-6	Scalability	The system should support increasing numbers of users and appointments without performance issues.

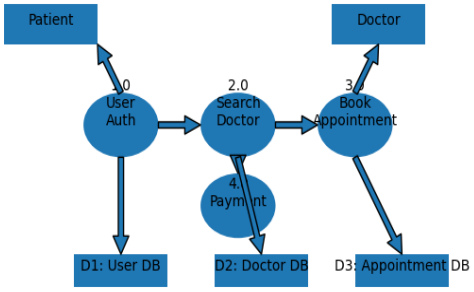
The functional requirements define what the system should do, while the non-functional requirements define how well the system should perform. Together, they ensure that DocSpot is efficient, secure, reliable, and user-friendly.

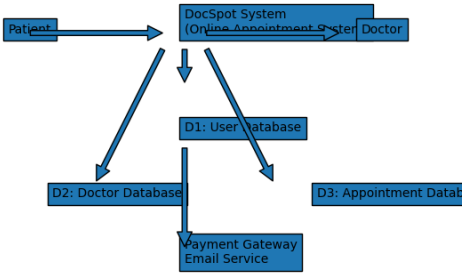
3.3 Data Flow Diagram

- User registers → Data stored in User DB
- User searches doctor → Fetch from Doctor DB
- User books appointment → Stored in Appointment DB
- Payment processed → Confirmation sent

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.





User Stories

- Use the below template to list all the user stories for the product.

Customer (Mobile/Web User – Patient)

User Type	Functional Requirement	User Story No	User Story	Acceptance Criteria	Priority	Release
Patient	Registration	USN-1	As a patient, I can register using email & password	I can access my dashboard after registration	High	Sprint-1
Patient	Registration	USN-2	As a patient, I receive confirmation email after registering	I receive and verify via email	High	Sprint-1
Patient	Registration	USN-3	As a patient, I can register using Gmail	I can login using Gmail successfully	Medium	Sprint-1
Patient	Login	USN-4	As a patient, I can login using email & password	I can access dashboard	High	Sprint-1

User Type	Functional Requirement	User Story No	User Story	Acceptance Criteria	Priority	Release
Patient	Dashboard	USN-5	As a patient, I can view available doctors	Doctor list is displayed	High	Sprint-1
Patient	Appointment	USN-6	As a patient, I can book appointment with doctor	Appointment confirmation is generated	High	Sprint-2
Patient	Appointment	USN-7	As a patient, I can cancel appointment	Appointment is removed from schedule	Medium	Sprint-2
Patient	Payment	USN-8	As a patient, I can pay consultation fees online	Payment confirmation is received	Medium	Sprint-2

Doctor

User Type	Functional Requirement	User Story No	User Story	Acceptance Criteria	Priority	Release
Doctor	Login	USN-9	As a doctor, I can login securely	Doctor dashboard is accessible	High	Sprint-1
Doctor	Manage Schedule	USN-10	As a doctor, I can update availability	Updated slots are visible to patients	High	Sprint-2
Doctor	Appointments	USN-11	As a doctor, I can view booked appointments	Appointment list is displayed	High	Sprint-2

Administrator

User Type	Functional Requirement	User Story No	User Story	Acceptance Criteria	Priority	Release
Admin	User Management	USN-12	As admin, I can view all users	User list is displayed	High	Sprint-1
Admin	Doctor Management	USN-13	As admin, I can approve/reject doctors	Doctor status updates	High	Sprint-2
Admin	Reports	USN-14	As admin, I can generate appointment reports	Reports are downloadable	Medium	Sprint-3

3.4 Technology Stack

Technical Architecture:

DocSpot follows a 3-Tier Architecture (Client–Server–Database Model):

1. **Presentation Layer (Frontend)**

- Built using React.js
- Provides UI for Patients, Doctors, and Admin
- Handles form submissions, appointment booking interface, dashboards

2. **Application Layer (Backend / API Server)**
    - Built using Node.js & Express.js
    - Handles business logic
    - Manages authentication, appointment processing, doctor management
    - Provides REST APIs
3. **Data Layer (Database)**
    - MongoDB (NoSQL Database)
    - Stores users, doctors, appointments, authentication data

Architecture Flow

1. User interacts with React frontend
2. Frontend sends API request to Express server
3. Backend processes request & interacts with MongoDB
4. Database returns response, Backend sends response to frontend
5. UI updates dynamically

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1	User Interface	Web-based UI for patients, doctors, admin	HTML, CSS, JavaScript, React.js
2	Application Logic-1	Authentication & User Management	Node.js, Express.js
3	Application Logic-2	Appointment Booking & Scheduling Logic	Node.js, Express.js
4	Application Logic-3	Role-Based Access Control (Admin/Doctor/Patient)	JWT, Middleware
5	Database	Stores user, doctor & appointment data	MongoDB (NoSQL)
6	Cloud Database	Managed cloud database service	MongoDB Atlas
7	File Storage	Store doctor profile images	Cloud Storage / Local File System
8	External API-1	Email Notification Service	Nodemailer / SMTP Service
9	External API-2	Payment Integration (Future Scope)	Razorpay API (Optional)
10	Machine Learning Model	(Future Scope) Doctor Recommendation System	Basic Recommendation Algorithm
11	Infrastructure (Server/Cloud)	Application deployment	Local Server / Render / Vercel / AWS

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology Used
1	Open-Source Frameworks	Uses open-source MERN stack technologies	React.js, Node.js, Express.js, MongoDB
2	Security Implementations	Secure authentication & data protection	JWT, bcrypt password hashing, HTTPS, CORS
3	Scalable Architecture	3-Tier architecture allows independent scaling of frontend & backend	MERN Stack, REST APIs
4	Availability	Cloud deployment ensures 24/7 availability	MongoDB Atlas, Cloud Hosting
5	Performance	Optimized API calls & database queries	Indexed MongoDB, Async/Await, Caching

4. PROJECT DESIGN

4.1 Problem Solution Fit

The solution directly addresses:

- Long hospital waiting times
- Lack of appointment transparency
- Manual record keeping

DocSpot provides a digital and automated solution.

Problem – Solution Fit Template:

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why

Purpose:

- ❑ Solve complex problems in a way that fits the state of your customers.
- ❑ Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.
- ❑ Sharpen your communication and marketing strategy with the right triggers and messaging.
- ❑ Increase touch-points with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.
- ❑ Understand the existing situation in order to improve it for your target group.

4.2 Proposed Solution

DocSpot is a cloud-based web application that allows:

- Patients to book appointments online
- Doctors to manage schedules

- Admin to manage users and reports

Proposed Solution Template:

Project team shall fill the following information in the proposed solution template.

S.No	Parameter	Description
1	**Problem Statement**	Patients face difficulty in booking doctor appointments due to long waiting queues, lack of real-time availability information, manual scheduling, and poor communication between hospitals and patients. Many people waste time visiting hospitals physically just to check doctor availability.
2	**Idea / Solution Description**	DocSpot is a web-based online doctor appointment booking system that allows patients to register, search for doctors based on specialization and availability, book appointments, make online payments, and receive instant confirmation. Doctors can manage their schedules, and administrators can monitor system activities through a centralized dashboard.
3	**Novelty / Uniqueness**	The system provides real-time slot availability, easy appointment rescheduling, role-based dashboards (Patient/Doctor/Admin), secure authentication (JWT), and cloud-based deployment. Future enhancements may include AI-based doctor recommendations and teleconsultation features.
4	**Social Impact / Customer Satisfaction**	DocSpot reduces hospital overcrowding, saves patient time, improves healthcare accessibility, and enhances patient satisfaction through convenience and transparency. It also supports digital healthcare transformation, especially in rural and semi-urban areas.
5	**Business Model (Revenue Model)**	Revenue can be generated through: • Commission per booked appointment • Subscription plans for hospitals/doctors • Premium doctor listing • Advertisement placements • Teleconsultation service fees
6	**Scalability of the Solution**	The system follows a scalable 3-tier MERN architecture. It can handle increasing users and appointments by scaling backend servers and cloud databases (MongoDB Atlas, AWS). The solution can expand to multiple cities, hospitals, and integrate telemedicine services in future.

4.3 Solution Architecture

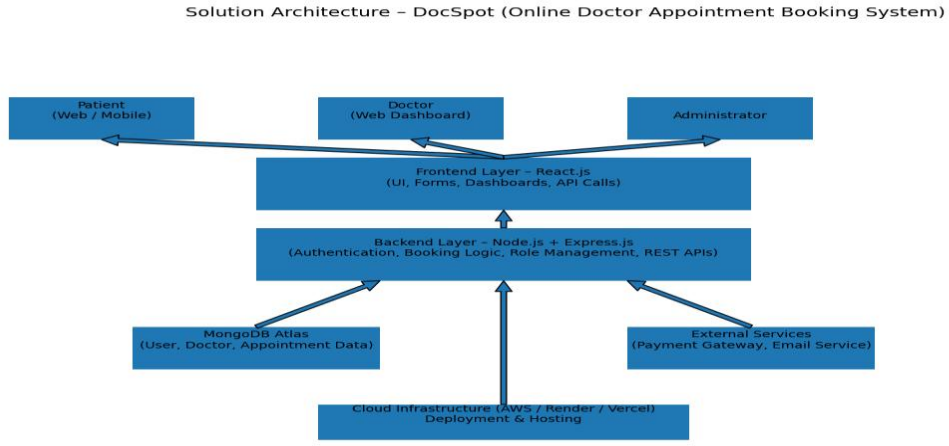
The system follows 3-tier architecture:

- Presentation Layer (React)
- Application Layer (Node + Express APIs)
- Data Layer (MongoDB)

External services include:

- Payment Gateway
- Email Service

Solution Architecture Diagram:



1. **Overview**

The DocSpot solution follows a 3-Tier Cloud-Based Architecture consisting of:

- Presentation Layer (Frontend)
- Application Layer (Backend APIs)
- Data Layer (Database)
- External Services Layer
- Cloud Infrastructure Layer

2. **Architecture Components**

User Layer
Patients, Doctors, and Administrators access the system through web browsers or mobile devices.
Frontend Layer
Developed using React.js, responsible for:

- User interface rendering
- Form handling
- Dashboard management
- API communication

Backend Layer
Built with Node.js & Express.js, responsible for:

- Authentication (JWT-based)
- Appointment booking logic
- Role-based access control
- REST API management

Database Layer
Uses MongoDB Atlas (Cloud NoSQL Database) to store:

- User details
- Doctor profiles
- Appointment records

External Services

- Payment Gateway integration
- Email notification service
- Future scope: Teleconsultation APIs

Infrastructure Layer

Application deployed on cloud platforms like:

- AWS / Render / Vercel
- Provides scalability, availability, and performance optimization

Architecture Benefits

✓ Scalable cloud deployment
✓ Secure authentication & data handling
✓ Modular and maintainable design
✓ Supports future enhancements like AI recommendations and telemedicine

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Phase 1: Requirement Gathering
Phase 2: UI/UX Design
Phase 3: Backend Development
Phase 4: Database Integration
Phase 5: Testing
Phase 6: Deployment
Development followed Agile methodology with sprint-based implementation.

A Sprint fixed period or duration in which a team works to complete a set of tasks
An **Epic** is a **big task or project** that is too large to complete in one sprint. It is broken down into **smaller tasks (stories)** that can be completed over multiple sprints.
A **Story** is a small task . It is part of an **Epic**.
A **Story Point** is a number that represents how much effort a story takes to complete. (usually in form of Fibonacci series)

1- Very Easy task
2- Normal task
3- Moderate task
5- Difficult task

Storypoint -5 (1,2,3,5)

Sprint 1

- **Epic 1: User Management**

User Story	Task	Story Points
US1	User Registration	3
US2	User Login	2
US3	Forgot Password	2

- **Epic 2: Doctor Management**

User Story	Task	Story Points
US4	Add Doctor Details	3
US5	View Doctor List	2

- **Epic 3: Appointment Booking (Basic)**

User Story	Task	Story Points
US6	Select Date & Time	3
US7	Book Appointment	5

Total Story Points in Sprint 1: 3 + 2 + 2 + 3 + 2 + 3 + 5 = 20

Sprint 2

- **Epic 4: Appointment Management**

User Story	Task	Story Points
US8	View Appointments	2
US9	Cancel Appointment	3
US10	Reschedule Appointment	3

- **Epic 5: Dashboard & Reports**

User Story	Task	Story Points
US11	Admin Dashboard	5
US12	Appointment Reports	3

- **Epic 6: Notification System**

User Story	Task	Story Points
US13	SMS Confirmation	3
US14	Email Notification	3

Total Story Points in Sprint 2: 2 + 3 + 3 + 5 + 3 + 3 + 3 = 22

Total Story Points

Sprint 1 = 20

Sprint 2 = 22

Total Story Points = 20 + 22 = 42

Number of Sprints = 2

Velocity Calculation

Velocity = Total Story Points Completed / Number of Sprints

Velocity = 42 / 2

21 Story Points per Sprint

Final Statement

The team completed 42 story points in 2 sprints.

Therefore, the team velocity is 21 Story Points per Sprint.

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story No	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register using email & password	3	High	Team A
Sprint-1	Registration	USN-2	As a user, I receive confirmation after registration	2	High	Team A
Sprint-1	Login	USN-3	As a user, I can login using email & password	2	High	Team B
Sprint-1	Doctor Management	USN-4	As admin, I can add doctor details	3	Medium	Team B
Sprint-1	Doctor Management	USN-5	As a user, I can view doctor list	2	High	Team C
Sprint-2	Appointment Booking	USN-6	As a user, I can select date & time	3	High	Team A
Sprint-2	Appointment Booking	USN-7	As a user, I can book appointment	5	High	Team B
Sprint-2	Appointment Management	USN-8	As a user, I can cancel appointment	3	Medium	Team C
Sprint-2	Dashboard	USN-9	As admin, I can view dashboard	4	Medium	Team B
Sprint-2	Notification	USN-10	As a user, I receive SMS confirmation	3	Medium	Team A

Total Story Points

- Sprint-1 = 12
- Sprint-2 = 18
- Total = 30 Story Points

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed	Sprint Release Date (Actual)
Sprint-1	12	6 Days	01 Feb 2026	06 Feb 2026	12	06 Feb 2026
Sprint-2	18	6 Days	08 Feb 2026	13 Feb 2026	18	13 Feb 2026
Sprint-3	16	6 Days	15 Feb 2026	20 Feb 2026	-	-
Sprint-4	16	6 Days	22 Feb 2026	27 Feb 2026	-	-

Velocity:

Total Story Points Completed

= 12 + 18

= 30

Number of Completed Sprints = 2

Velocity = 30 / 2

15 Story Points per Sprint

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint).

Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

Average Velocity Per Day

If Sprint Duration = 6 Days

Velocity = 15 Points per Sprint

Average Velocity per Day (AV)

= 15 / 6

= **2.5 Story Points per Day**

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

A Burndown Chart shows the remaining work versus time.

In this project:

- X-axis represents sprint days (1–6)
- Y-axis represents story points
- The ideal line decreases evenly from total story points to zero
- The actual line shows real progress

If work is completed on time, the actual line matches the ideal line.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

- Login functionality tested
- Appointment booking tested
- Payment workflow tested
- Load tested for multiple users
- Response time maintained under 2–3 seconds

The Online Doctor Appointment Booking System is a web-based application that allows users to register, log in, view doctors, book appointments, cancel appointments, and receive confirmations. It reduces waiting time and improves hospital management efficiency.
Project Version: 1.0
Testing Period: 15 February 2026 to 20 February 2026
Testing Scope:
Features & Functionalities to be Tested:

- User Registration
- User Login
- View Doctor List
- Book Appointment
- Cancel Appointment
- Admin Dashboard
- SMS/Email Notification

User Stories Tested:

- USN-1: User Registration
- USN-3: User Login
- USN-6: Select Date & Time
- USN-7: Book Appointment
- USN-8: Cancel Appointment

- USN-9: Admin Dashboard

Testing Environment:

URL/Location: http://localhost:3000

Credentials:

Username: testuser@gmail.com

Password: Test@123

Test Cases:

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
TC-001	User Registration	1. Open application 2. Click Register 3. Enter details 4. Submit	User account created successfully	Account created successfully	Pass
TC-002	User Login	1. Enter valid email & password 2. Click login	User redirected to dashboard	Redirected successfully	Pass
TC-003	Invalid Login	1. Enter wrong password 2. Click login	Error message displayed	Error message shown	Pass
TC-004	View Doctor List	1. Login 2. Click View Doctors	List of doctors displayed	Doctors displayed correctly	Pass
TC-005	Book Appointment	1. Select doctor 2. Choose date/time 3. Confirm	Appointment booked successfully	Appointment confirmed	Pass
TC-006	Cancel Appointment	1. Go to My Appointments 2. Click Cancel	Appointment cancelled successfully	Cancelled successfully	Pass
TC-007	Admin Dashboard	1. Login as admin 2. Open dashboard	Dashboard displays reports	Reports displayed	Pass

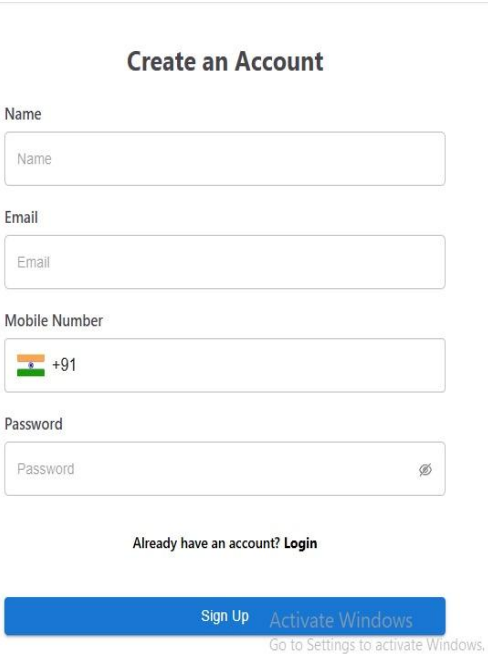
Bug Tracking:

Bug ID	Bug Description	Steps to Reproduce	Severity	Status	Additional Feedback
BG-001	Slow loading of doctor list	1. Login 2. Open doctor list	Medium	Closed	Optimized database query
BG-002	SMS notification delay	1. Book appointment	Low	In Progress	Needs API improvement

7. RESULTS

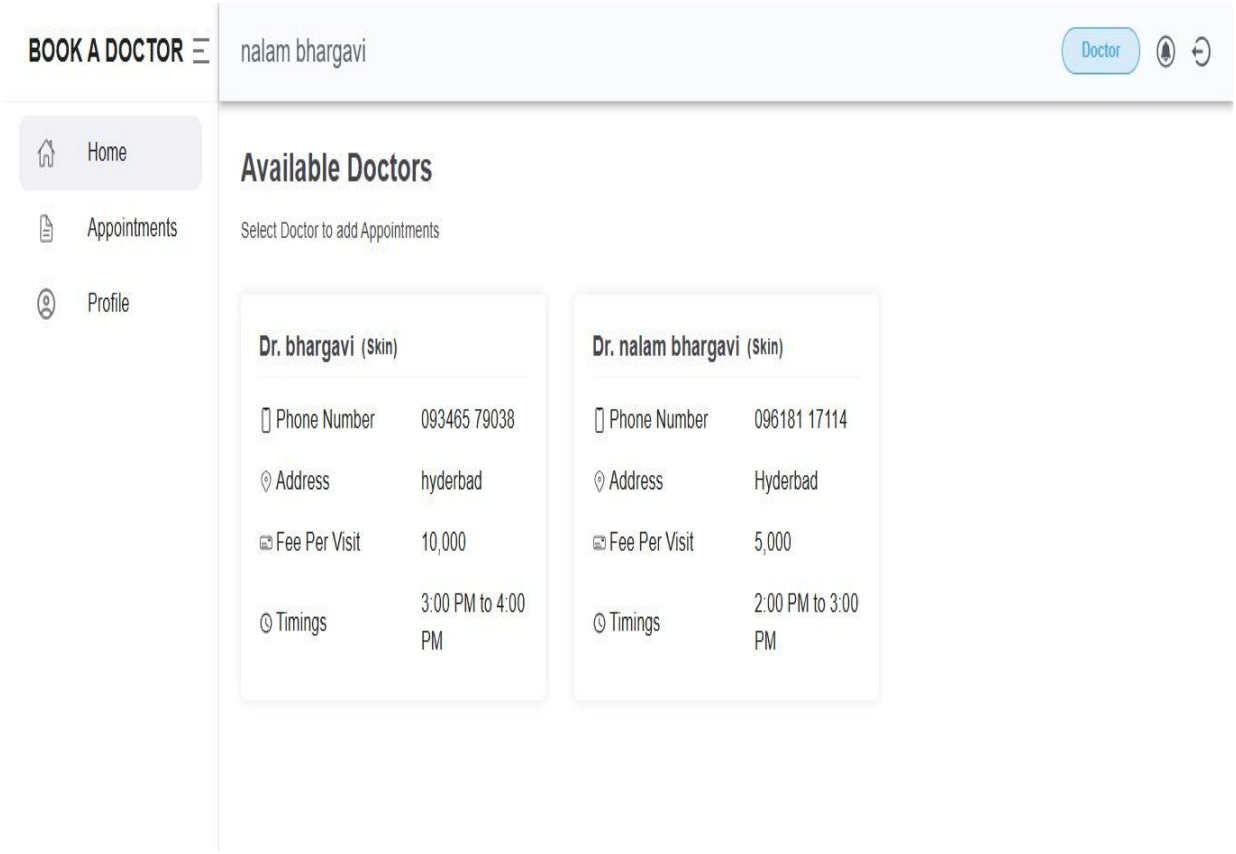
7.1 Output Screenshots

Registration Page

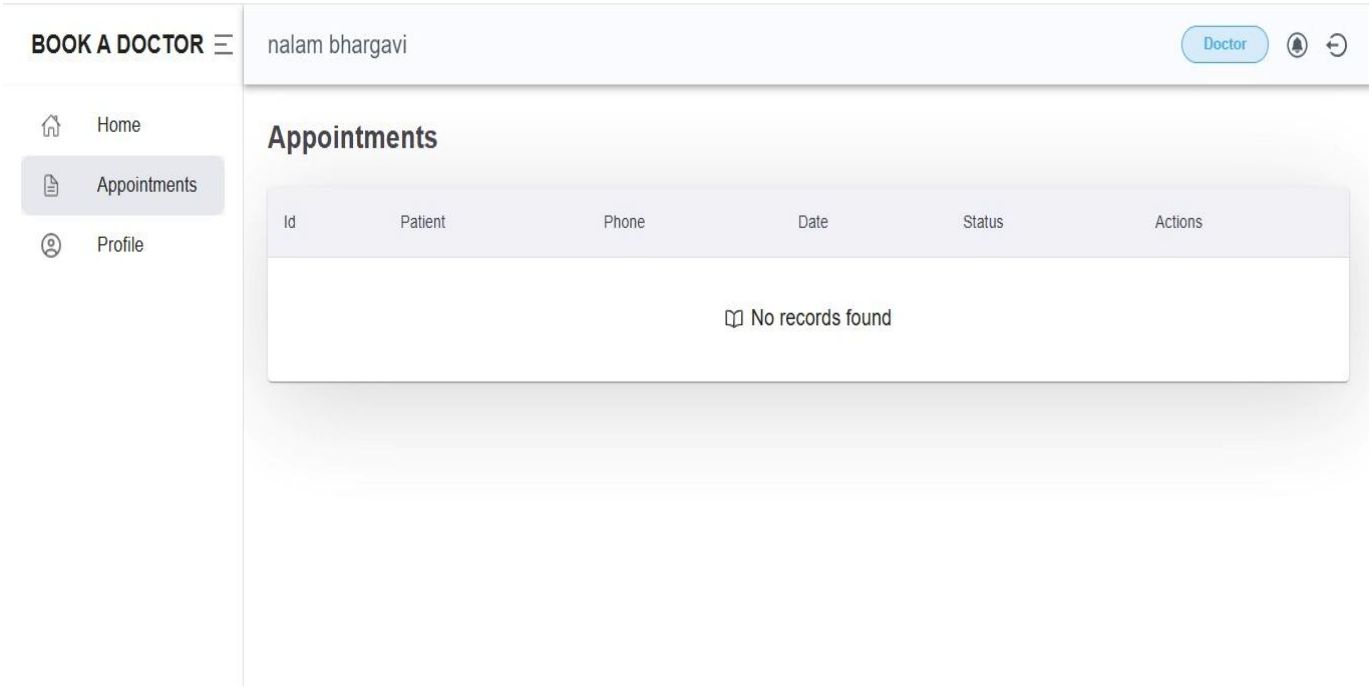


Login Page

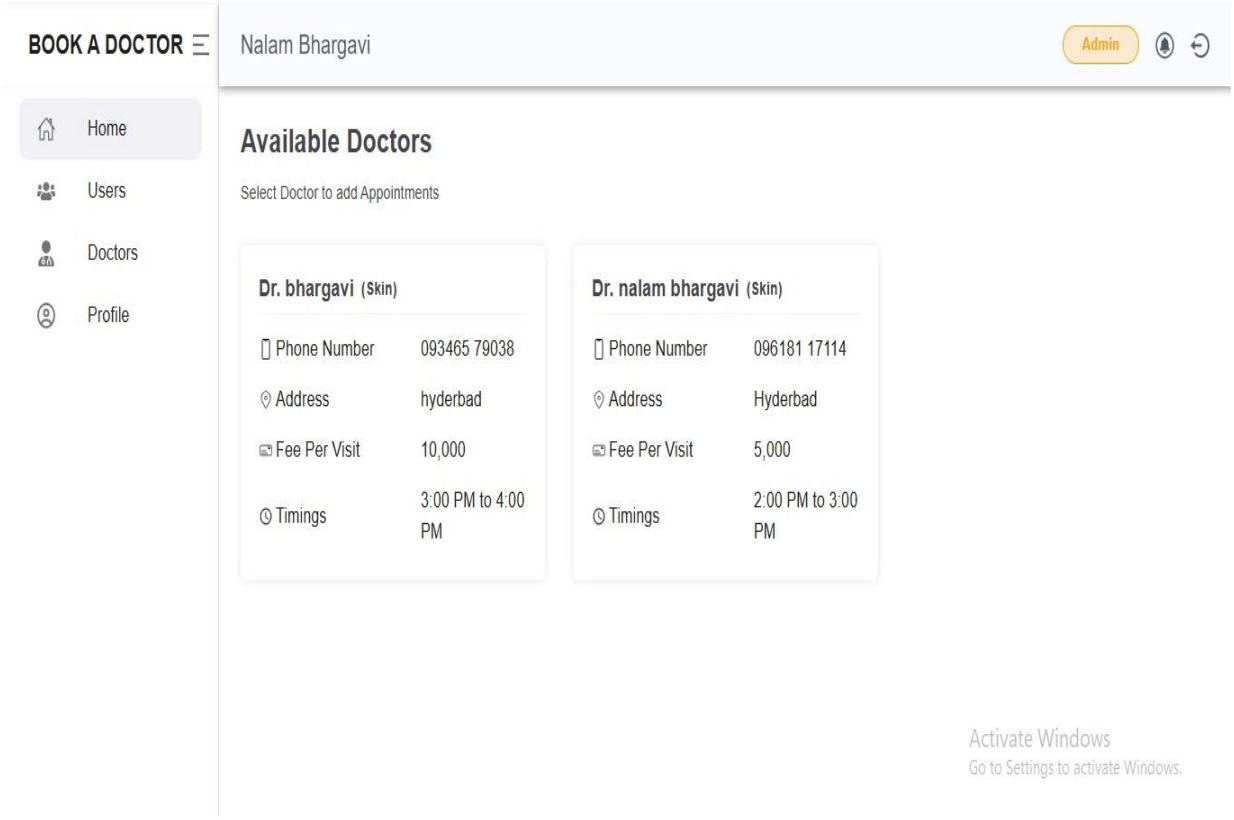
Doctor Dashboard



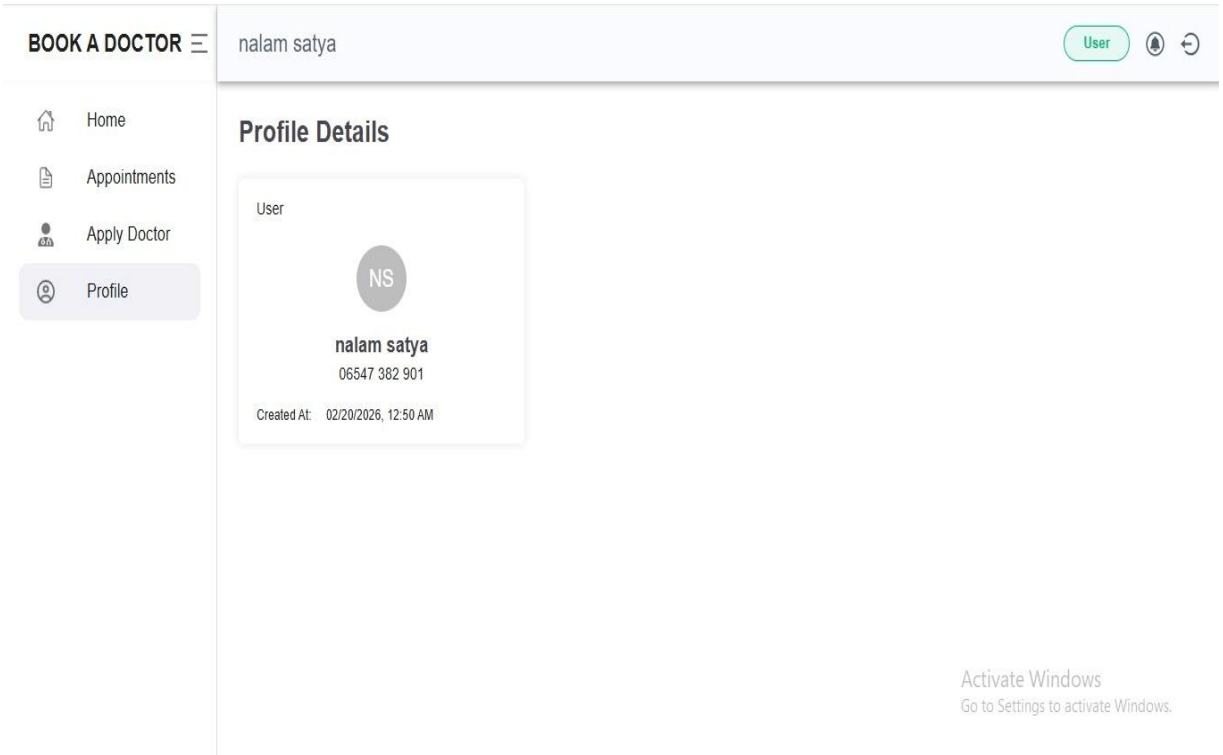
Appointment Booking Page



Admin Dashboard



User Dashboard



8. ADVANTAGES & DISADVANTAGES

Advantages:

- Saves time
- Reduces hospital crowd
- Easy access
- Secure system
- Cloud scalable

Disadvantages:

- Requires internet access
- Initial setup cost
- Users must be digitally literate

9. CONCLUSION

The Doctor Appointment Booking System developed using the MERN stack provides an efficient and user-friendly solution for managing medical appointments online. The system simplifies the booking process for patients while enabling doctors and administrators to manage schedules effectively.

This project demonstrates the practical implementation of full-stack development using modern web technologies. With secure authentication and modular architecture, the application serves as a strong foundation for future enhancements and real-world healthcare deployment.

10. FUTURE SCOPE

- Telemedicine video consultation
- AI-based doctor recommendation
- Mobile application version
- Integration with hospital management systems
- Multi-language support
- Online prescription storage

11. APPENDIX

- **Source Code**
  https://drive.google.com/drive/folders/1k8dAHWUpJ1poLIjZK_V4U3Pl5u8DkWo1?usp=sharing
- **GitHub Link**
  https://github.com/gowthamidola123/DocSpot-Seamless-Appointment-Booking-for-Health
- **Project Demo Link**
  https://drive.google.com/file/d/1OPdijfDUQ4LQoFS_qTth1KWH1CbK90Aq/view?usp=drive_link