INTRODUCTION

Project Title: DocSpot: Seamless Appointment Booking For Health

Team Members:
- PANDIRI BHAVAJNA : *Backend Developer*
- IMANDI SOUMYA : *Frontend Developer*
- LANKA KEERTHI : *Database & Testing Manager*

DocSpot is a web-based healthcare appointment booking system developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The application is designed to simplify the process of scheduling medical consultations by providing a digital platform where users can book appointments with doctors easily.

The system replaces traditional manual appointment methods with an efficient online solution. It allows users to register, log in, book appointments, and track their appointment status. Administrators can manage doctors and approve requests through a dedicated dashboard.

1. Project Overview

DocSpot provides a centralized platform that connects patients and doctors through an online booking system. The application includes user authentication, appointment scheduling, appointment tracking, and admin management features.

The frontend is built using React.js for a responsive and interactive user interface. The backend is developed using Node.js and Express.js to handle server-side operations, while MongoDB is used for secure data storage.

The system ensures smooth communication between users and administrators while maintaining security and reliability.

2. Purpose

The primary purpose of DocSpot is to reduce waiting time and improve the efficiency of hospital appointment management. It aims to provide a convenient and accessible platform where users can book medical appointments anytime and from anywhere.

The project also focuses on improving transparency in doctor availability and enhancing overall user experience through a secure and user-friendly interface. By digitizing the appointment process, DocSpot contributes to a more organized and efficient healthcare management system.

Ideation Phase
Define the Problem Statements

Date	31 January 2026
Team ID	LTVIP2026TMIDS66955
Project Name	Project Title : DocSpot: Seamless Appointment Booking for Health
Maximum Marks	2 Marks

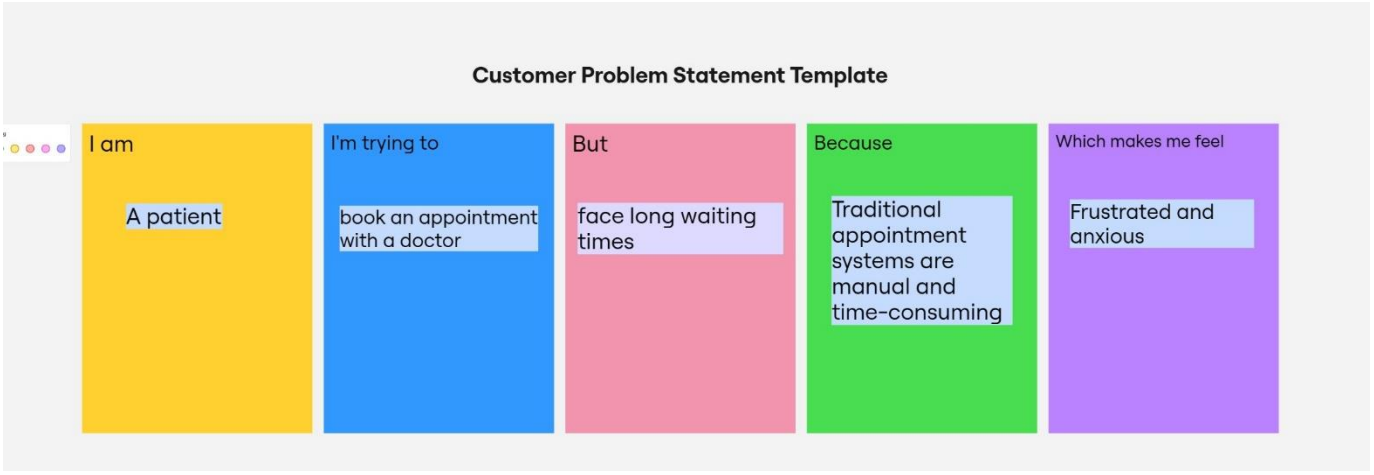
Customer Problem Statement Template:

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

I am	Describe customer with 3-4 key characteristics - *who are they?*	Describe the customer and their attributes here
I'm trying to	*List their outcome or "job" the care about - what are they trying to achieve?*	List the thing they are trying to achieve here
but	Describe what problems or barriers stand in the way – *what bothers them most?*	Describe the problems or barriers that get in the way here
because	Enter the "root cause" of why the problem or barrier exists – *what needs to be solved?*	Describe the reason the problems or barriers exist
which makes me feel	Describe the emotions from the customer's point of view – *how does it impact them emotionally?*	Describe the emotions the result from experiencing the problems or barriers

Reference: https://miro.com/templates/customer-problem-statement/



Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A patient who needs to consult a doctor for routine checkups or health issues	To book an appointment with a suitable doctor at a convenient time	I face long waiting times, unclear availability, and difficulty finding the right specialist	Traditional appointment systems are manual and time-consuming	Frustrated and anxious about getting timely medical care
PS-2	A doctor who wants to manage appointments efficiently	To organize my schedule and confirm patient bookings smoothly	I receive requests from multiple channels causing conflicts and missed updates	There is no centralized system for bookings and cancellations	Overwhelmed and unable to provide a smooth patient experience

Ideation Phase
Empathize & Discover

Date	31 January 2026
Team ID	LTVIP2026TMIDS66955
Project Name	Project Title : DocSpot: Seamless Appointment Booking for Health
Maximum Marks	4 Marks

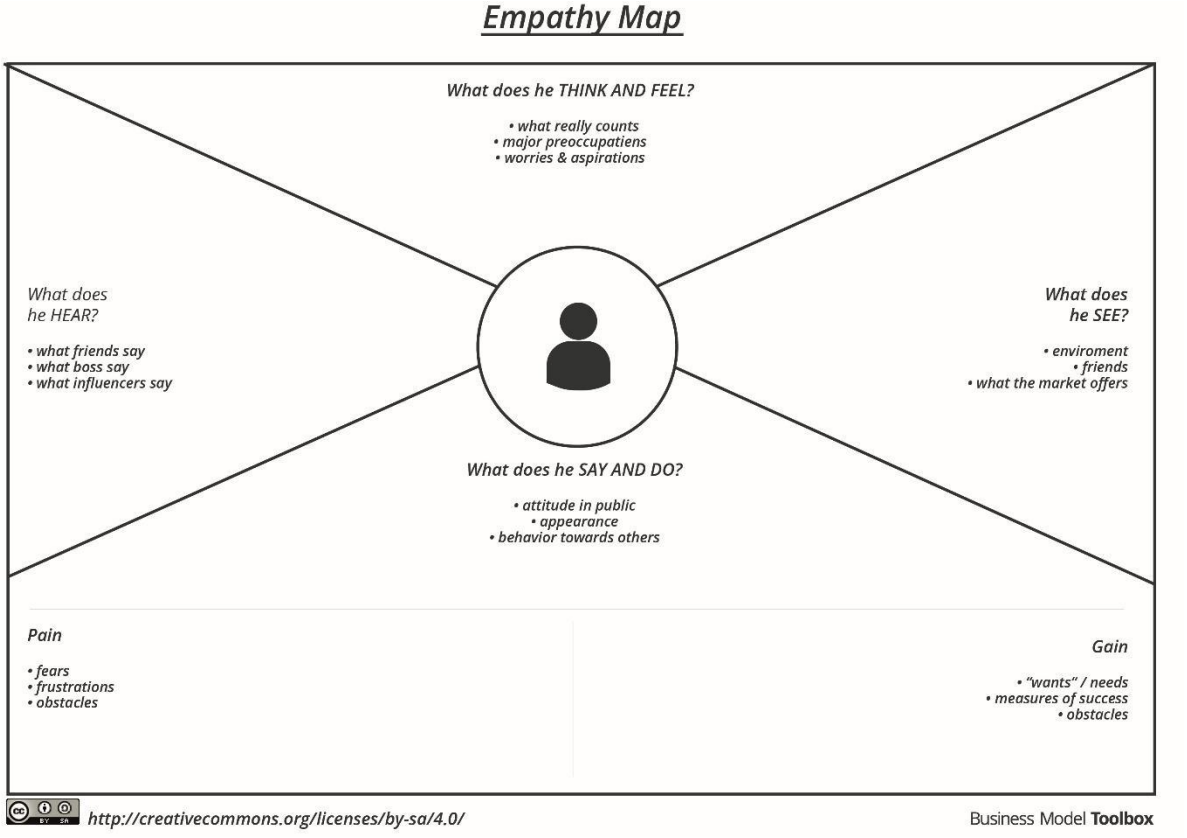
Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

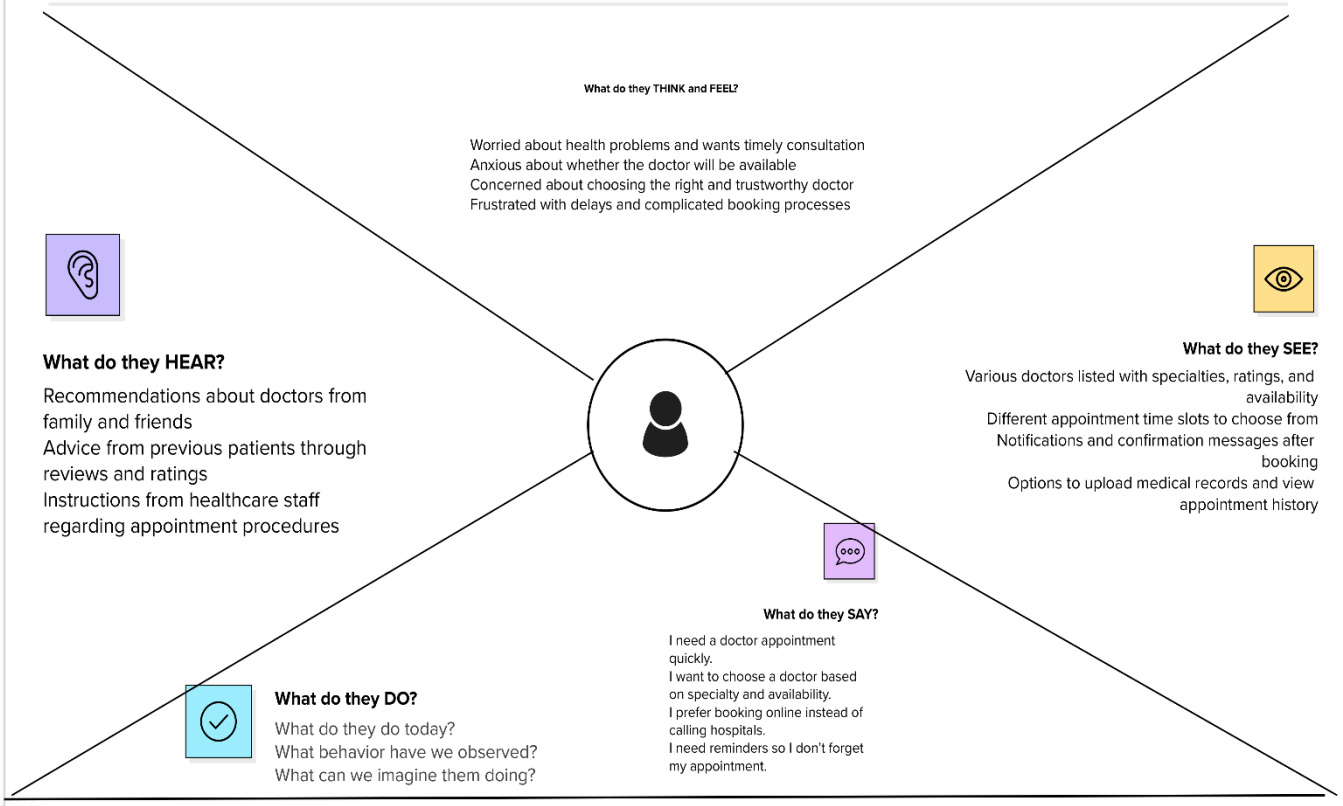
It is a useful tool to helps teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Example:



Reference: https://www.mural.co/templates/empathy-map-canvas



PAINS

Difficulty finding the right doctor quickly.
Long waiting time in traditional booking.
Lack of real-time availability information.

GAINS

Easy and quick appointment booking.
Real-time doctor availability.
Notifications and reminders.

Ideation Phase
Brainstorm & Idea Prioritization Template

Date	31 January 2026
Team ID	LTVIP2026TMIDS66955
Project Name	Project Title : DocSpot: Seamless Appointment Booking for Health
Maximum Marks	4 Marks

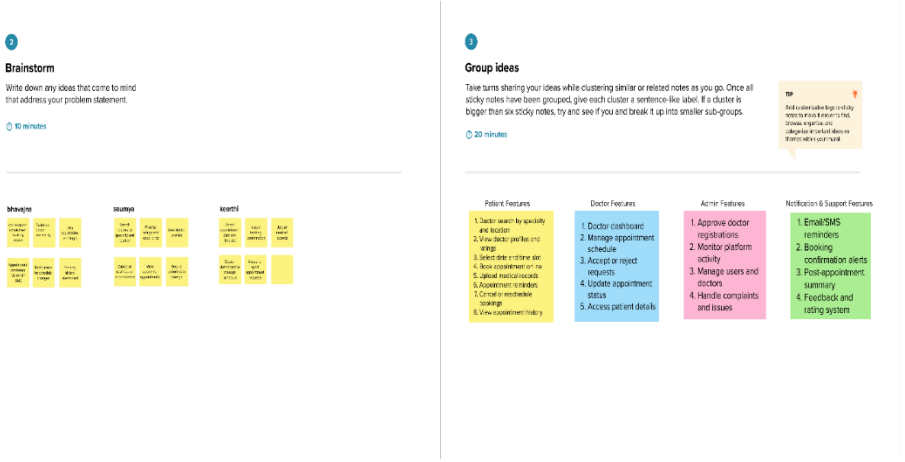
Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

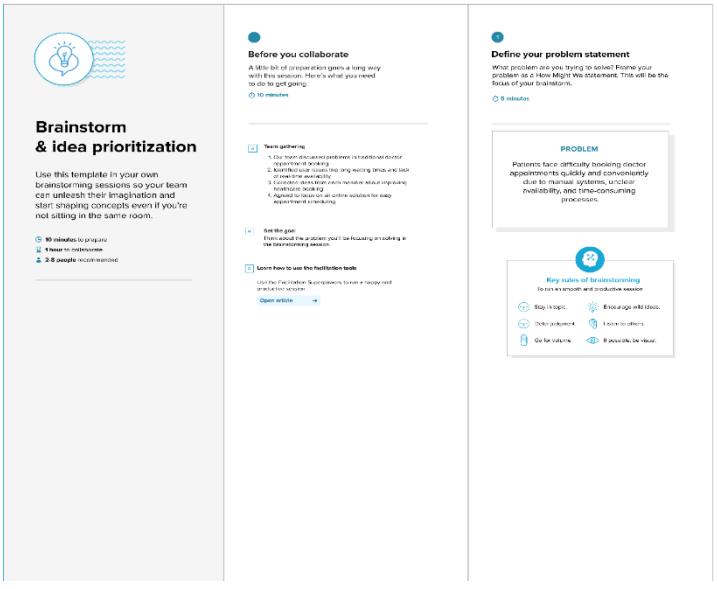
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Reference:

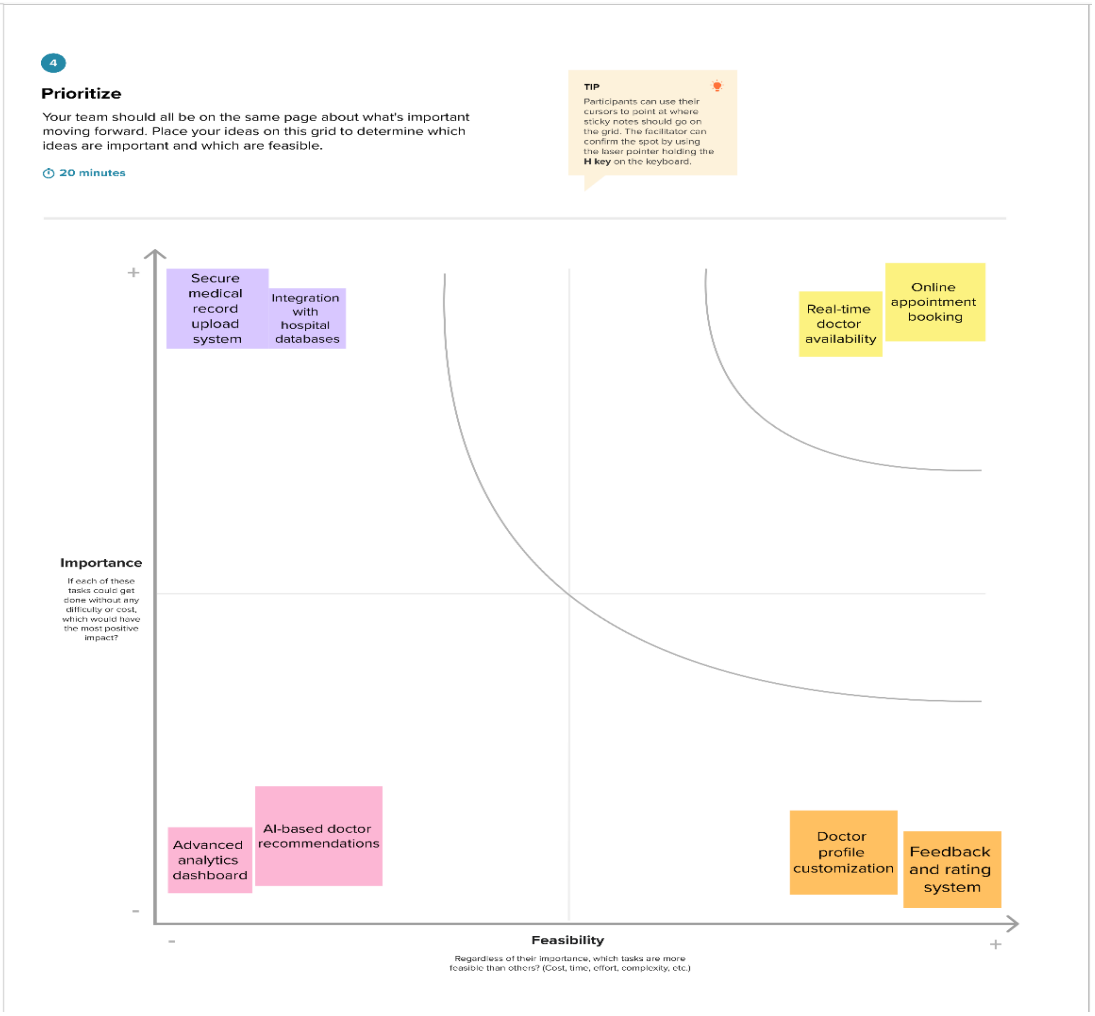
Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



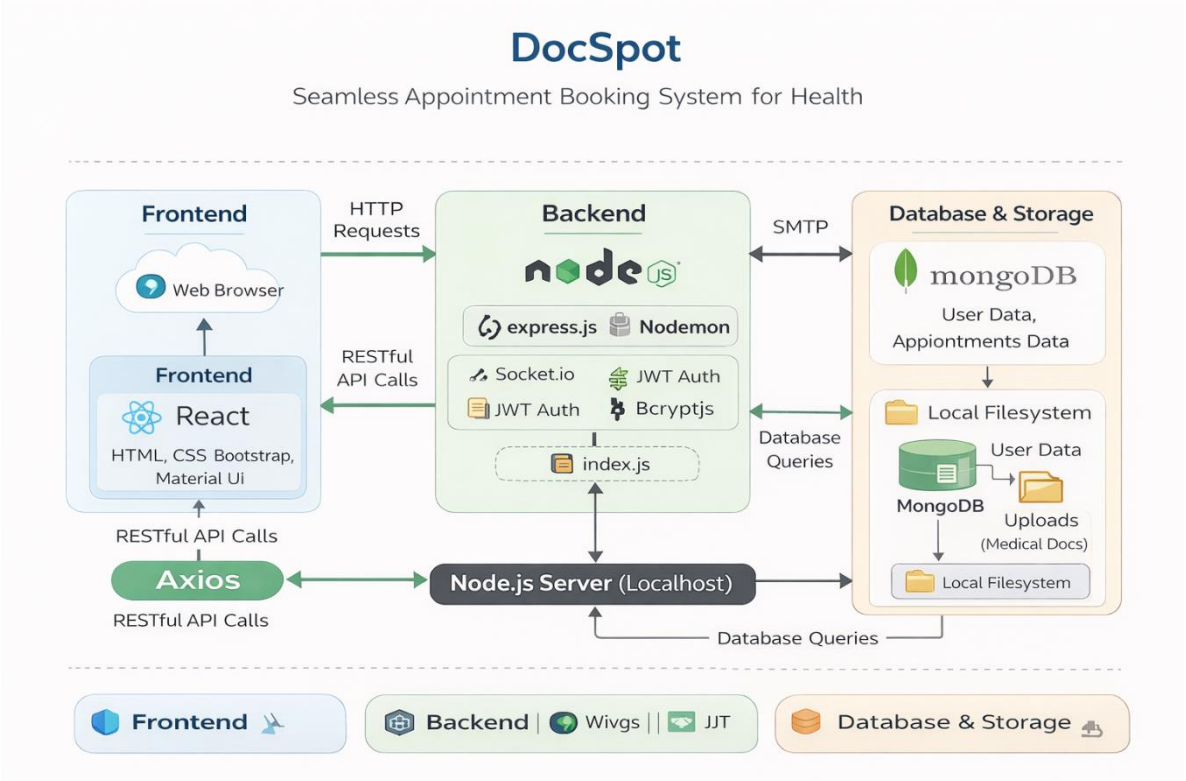
Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	31 January 2026
Team ID	LTVIP2026TMIDS66955
Project Name	DocSpot: Seamless Appointment Booking for Health
Maximum Marks	4 Marks

Technical Architecture:

The DocSpot application follows a client-server architecture. The frontend is developed using React and related UI libraries, which provides an interactive interface for patients, doctors, and admin users. The backend is implemented using Node.js and Express.js to handle application logic, authentication, and API services. MongoDB is used as the primary database for storing user data, doctor information, and appointment records. File uploads such as medical documents are handled using Multer and stored locally. The system uses RESTful APIs for communication between frontend and backend. Authentication and authorization are implemented using JSON Web Tokens (JWT). The application runs on a local server environment and can be deployed to cloud infrastructure if required.

Example: Order processing during pandemics for offline mode



Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
2	User Interface	Web interface for patients, doctors, and admin	React, HTML, CSS, JavaScript, Bootstrap, Material UI, Ant Design
3	Application Logic-1	Backend server logic and API handling	Node.js, Express.js
4	Application Logic-2	Authentication and authorization	JWT, bcryptjs
5	Application Logic-3	File upload and handling medical documents	Multer
6	Database	Stores users, doctors, appointments	MongoDB (Mongoose ODM)
7	Cloud Database	(Local database used)	------
8	File Storage	Stores uploaded medical documents	Local filesystem
9	External API-1	Email or notification services	SMTP / Email service
1	External API-2	Not Used	---------
1	Machine Learning Model	Not Applicable	---------
1	Infrastructure (Server / Cloud)	Application hosted on local server	Node.js Server (Localhost)

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	React, Express.js, Node.js, MongoDB, Bootstrap	MERN Stack
2	Security Implementations	Password hashing, authentication, protected routes	bcryptjs, JWTe.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3	Scalable Architecture	Modular architecture separating frontend and backend	REST API architecture
4	Availability	System accessible via web browser anytime	Web-based deployment
5	Performance	Efficient API communication and database queries	Axios, MongoDB indexing

References:

https://c4model.com/
https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/
https://www.ibm.com/cloud/architecture
https://aws.amazon.com/architecture
https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d

Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	31 January 2026
Team ID	LTVIP2026TMIDS66955
Project Name	DocSpot: Seamless Appointment Booking for Health
Maximum Marks	4 Marks

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 Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Doctor Search & Browsing	View list of available doctors Filter by specialty, location, availability View doctor profiles and ratings
FR-4	Appointment Booking	Select date and time slot Book appointment online Upload medical documents Receive booking confirmation
FR-5	Appointment Management	View upcoming appointments Cancel appointments Reschedule appointments
FR-6	Doctor Dashboard	View appointment requests Accept or reject appointments

Non-functional Requirements:

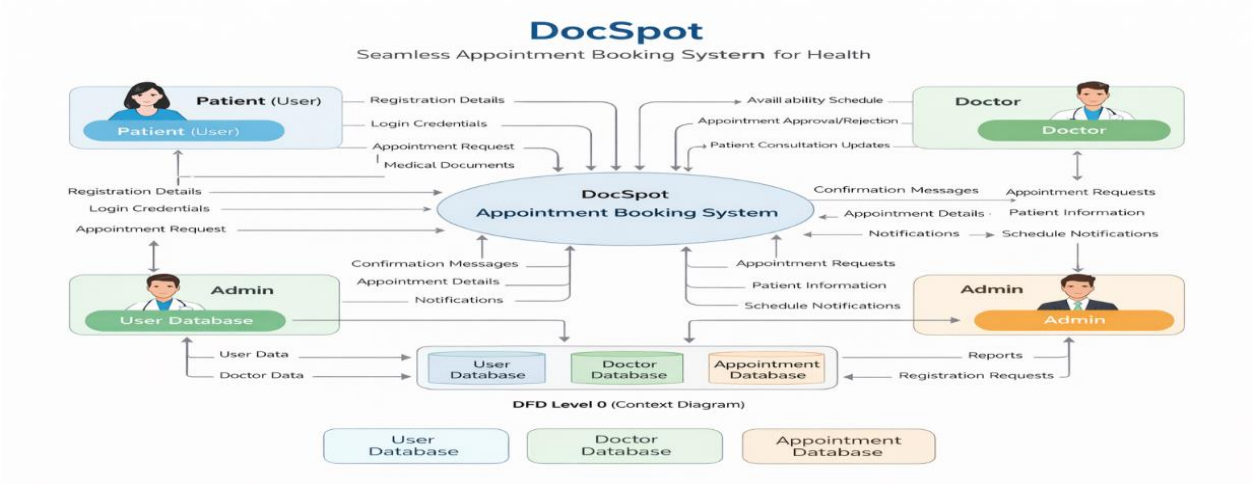
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Simple and user-friendly interface Easy navigation for patients, doctors, and admin Accessible on different devices
NFR-2	**Security**	Secure login and authentication Protection of patient medical data Encrypted data transmission
NFR-3	**Reliability**	System should function without failures Accurate appointment scheduling Proper error handling
NFR-4	**Performance**	Fast loading of pages Quick search and booking process Efficient handling of multiple users
NFR-5	**Availability**	System available 24/7 Minimal downtime Continuous access to booking services
NFR-6	**Scalability**	Ability to support increasing users and doctors Expandable database and server capacity

Project Design Phase-II
Data Flow Diagram & User Stories

Date	31 January 2026
Team ID	LTVIP2026TMIDS66955
Project Name	DocSpot: Seamless Appointment Booking for Health
Maximum Marks	4 Marks

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.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Patient)	Registration	USN-1	As a patient, I can register using email and password	User can access dashboard	High	Sprint-1
Customer (Patient)	Registration	USN-2	As a patient, I receive confirmation email after registration	Email received and verified	High	Sprint-1
Customer (Patient)	Login	USN-3	As a patient, I can log in using credentials	Dashboard opens successfully	High	Sprint-1
Customer (Patient)	Appointment Booking	USN-4	As a patient, I can search doctors by specialty and location	Relevant doctors displayed	High	Sprint-1
Customer (Patient)	Appointment Booking	USN-5	As a patient, I can book an appointment	Appointment confirmation shown	High	Sprint-1
Customer (Patient)	Appointment Booking	USN-6	As a patient, I can cancel or reschedule appointment	Status updated	Medium	Sprint-2
Doctor	Appointment Booking	USN-7	As a doctor, I can view appointment requests	Requests visible in dashboard	High	Sprint-1
Doctor	Appointment Booking	USN-8	As a doctor, I can approve or reject appointments	Status updated	High	Sprint-1
Administrator	Registrations	USN-9	As an admin, I can approve doctor registrations	Doctor added to system	High	Sprint-1
Administrator	Registrations	USN-10	As an admin, I can manage users and doctors	System updated successfully	Medium	Sprint-2

Project Design Phase
Solution Architecture

Date	15 February 2026
Team ID	LTVIP2026TMIDS66955
Project Name	DocSpot: Seamless Appointment Booking for Health
Maximum Marks	4 Marks

Solution Architecture:

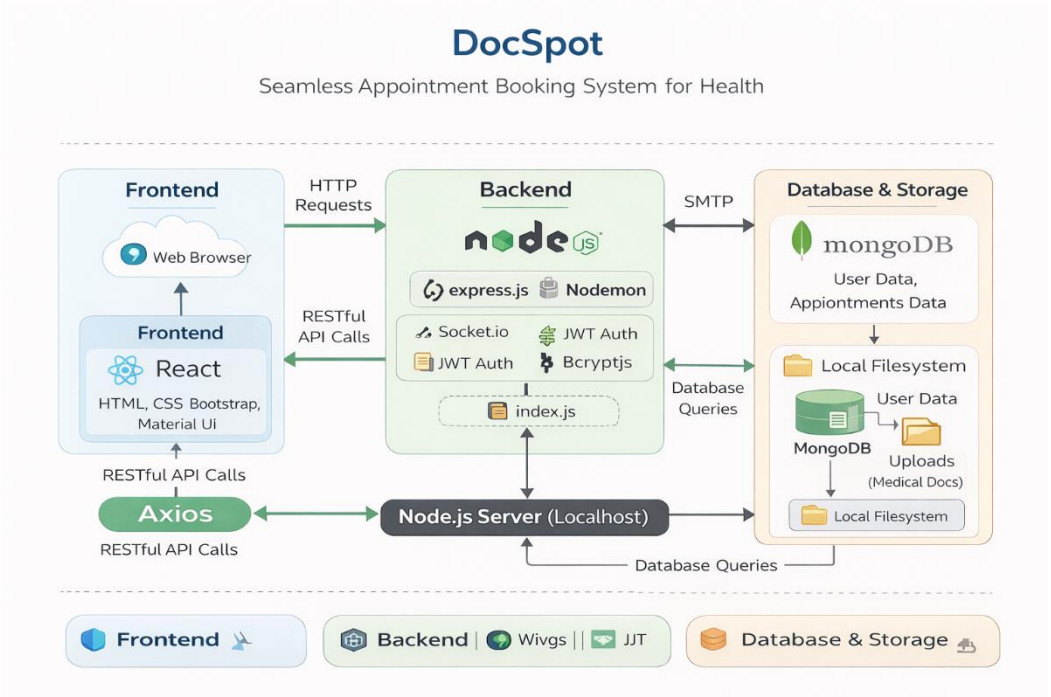
The DocSpot system follows a client-server architecture designed to connect patients, doctors, and administrators through a centralized platform. The frontend is developed using React, providing an interactive web interface for users. The backend is implemented using Node.js and Express.js, which handle application logic, authentication, and RESTful API communication. MongoDB is used as the primary database to store user data, doctor details, and appointment information.

Patients interact with the system through a web browser to register, search for doctors, book appointments, and manage bookings. Doctors access the platform to manage schedules and appointment requests after admin approval. Administrators oversee user management, doctor verification, and platform operations.

Authentication and security are implemented using JSON Web Tokens (JWT) and password hashing. File uploads such as medical documents are handled using Multer and stored locally. The architecture ensures efficient data flow, scalability, and reliable communication between components.

Example - Solution Architecture Diagram:

Figure 1: Architecture and data flow of the voice patient diary sample application



Project Design Phase
Proposed Solution Template

Date	15 February 2026
Team ID	LTVIP2026TMIDS66955
Project Name	DocSpot: Seamless Appointment Booking for Health
Maximum Marks	2 Marks

Proposed Solution Template:

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

S.No.	Parameter	Description
6.	Problem Statement (Problem to be solved)	Patients face difficulty booking doctor appointments due to manual processes, long waiting times, lack of real-time availability, and difficulty finding suitable specialists. Doctors also struggle to manage appointments efficiently, while administrators lack a centralized system to monitor operations
7.	Idea / Solution description	DocSpot is an online appointment booking platform that connects patients, doctors, and administrators in a single system. Patients can search for doctors, view real-time availability, book appointments, upload medical documents, and receive notifications. Doctors can manage schedules and appointment requests, while admins approve registrations and oversee platform operations.
8.	Novelty / Uniqueness	Centralized platform for patients, doctors, and admin Real-time appointment availability Integrated document upload for medical records Appointment status tracking and notifications Easy cancellation and rescheduling
9.	Social Impact / Customer Satisfaction	The solution improves access to healthcare by reducing waiting time and simplifying appointment booking. It enhances patient convenience, helps doctors manage schedules efficiently, and improves overall healthcare service delivery.
10.	Business Model (Revenue Model)	Commission from doctors or hospitals per appointment Subscription plans for premium features Advertisement from healthcare providers Partnership with hospitals and clinics
11.	Scalability of the Solution	The system can scale to support more users, doctors, and hospitals by expanding database capacity and deploying on cloud infrastructure. The modular architecture allows integration with additional healthcare services in the future.

Project Design Phase
Problem – Solution Fit Template

Date	15 February 2026
Team ID	LTVIP2026TMIDS66955
Project Name	DocSpot: Seamless Appointment Booking for Health
Maximum Marks	2 Marks

Problem :
Patients face difficulty booking doctor appointments due to manual processes, long waiting times, lack of real-time availability, and difficulty finding suitable specialists. Doctors also struggle to manage appointments efficiently, and admins need a centralized system to monitor operations.

Target Customers
1. Patients seeking quick and convenient doctor appointments
2. Doctors who want to manage schedules efficiently
3. Healthcare administrators managing users and approvals

Existing Alternatives
1. Phone-based appointment booking
2. Walk-in hospital visits
3. Manual register systems
4. Multiple disconnected hospital apps

Key Customer Pain Points
1. Long waiting times
2. Lack of real-time appointment availability
3. Difficulty finding the right doctor
4. Appointment scheduling conflicts
5. No centralized booking system

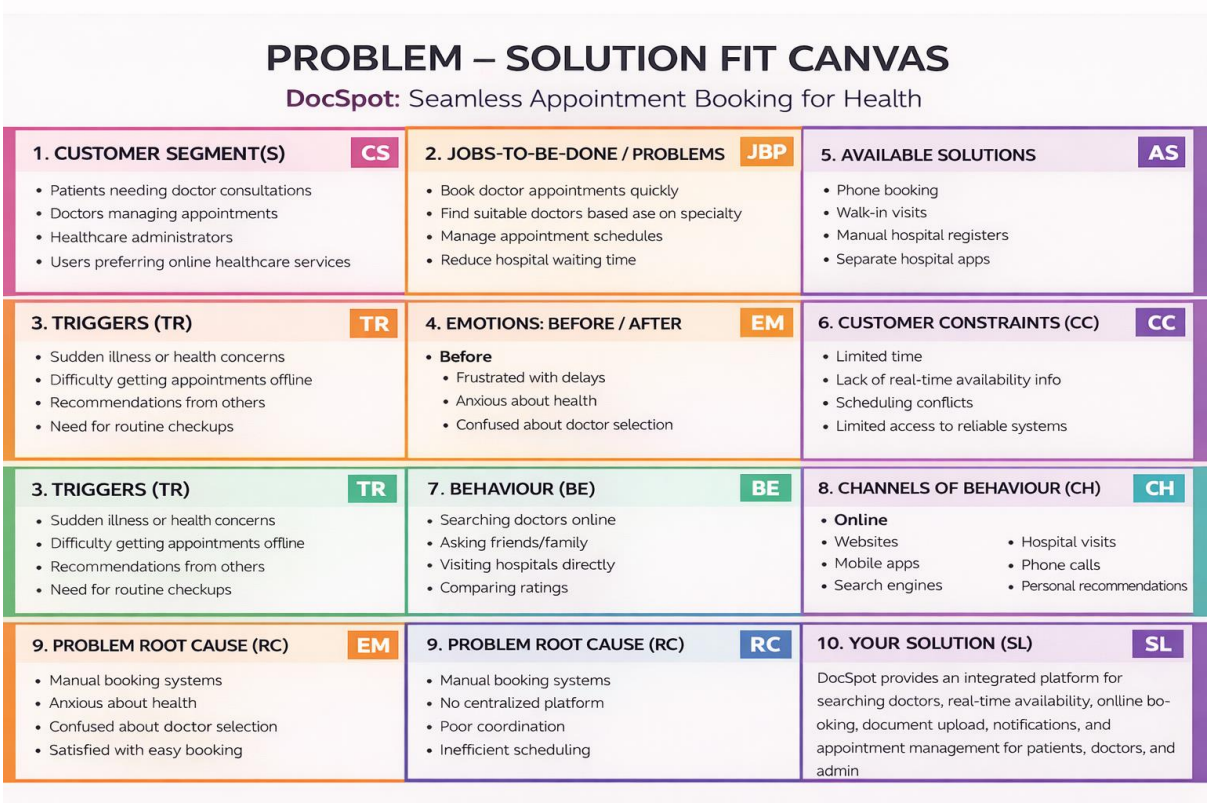
Proposed Solution

DocSpot is an online appointment booking platform that allows patients to search doctors, view real-time availability, book appointments, upload medical documents, and receive notifications. Doctors can manage schedules and appointments, while admins monitor the platform and approve registrations.

Value Proposition
1. Fast and convenient booking
2. Real-time availability
3. Centralized management system
4. Improved patient experience
5. Efficient appointment handling

How the Solution Fits the Problem

The DocSpot system directly addresses the limitations of traditional booking methods by providing an automated, user-friendly platform that connects patients, doctors, and administrators in one place, reducing delays and improving healthcare accessibility.



References:

6. https://www.ideahackers.network/problem-solution-fit-canvas/
7. https://medium.com/@epicantus/problem-solution-fit-canvas-aa3dd59cb4fe

Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	15 February 2026
Team ID	LTVIP2026TMIDS66955
Project Name	DocSpot: Seamless Appointment Booking for Health
Maximum Marks	5 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

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	PANDIRI BHAVAJNA
Sprint-1	Login	USN-2	As a user, I can login securely	1	High	IMANDI SOUMYA
Sprint-1	Security	USN-3	As a user, my password is encrypted	3	High	LANKA KEERTHI
Sprint-1	Dashboard	USN-4	As a user, I can access dashboard after login	3	Medium	PANDIRI BHAVAJNA
Sprint-1	Navigation	USN-5	As a user, I can navigate easily	3	Medium	IMANDI SOUMYA

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Doctor Module	USN-6	As a user, I can view doctor list.	2	High	LANKA KEERTHI
Sprint-2	Appointment	USN-7	As a user, I can book appointment.	3	High	IMANDI SOUMYA
Sprint-2	Appointment	USN8	As a user, I can view my appointments.	3	Medium	LANKA KEERTHI
Sprint-2	Status	USN-9	As a user, I can see appointment status.	4	Medium	PANDIRI BHAVAJNA
Sprint-2	Admin	USN-10	As a user, I can see appointment status.	5	High	IMANDI SOUMYA

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

12 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	12	6 Days	01 Feb 2025	06 Feb 2025	12	06 Feb 2025
Sprint-2	17	6 Days	08 Feb 2025	13 Feb 2025	17	13 Feb 2025

Velocity:

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)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Total Story Points Completed = 12 + 17 = 29

Number of Sprints = 2

Velocity = 29 / 2

Velocity = 14.5 ≈ 15 Story Points per Sprint

Burndown Chart:

A burndown chart is a graphical representation of the remaining work versus time in a sprint. It helps track the progress of the team and ensures tasks are completed within the planned timeline.

In the DocSpot project:

8. The X-axis represents time (days of sprint)
9. The Y-axis represents remaining story points
10. The ideal line shows planned completion
11. The actual line shows real progress

This helps monitor whether the team is ahead or behind schedule.

Reference:

- https://www.atlassian.com/agile/project-management
- https://www.atlassian.com/agile/tutorials/epics
- https://www.atlassian.com/agile/tutorials/sprints
- https://www.atlassian.com/agile/tutorials/burndown-charts

Acceptance Testing
UAT Execution & Report Submission

Date	16 February 2026
Team ID	LTVIP2026TMIDS66955
Project Name	DocSpot: Seamless Appointment Booking for Health
Maximum Marks	4 Marks

12. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT)

13. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

14. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

User Acceptance Testing (UAT) Template

Date	15 February 2026
Team ID	LTVIP2026TMIDS66955
Project Name	DocSpot: Seamless Appointment Booking for Health
Maximum Marks	5 Marks

Project Overview:
Project Name: DocSpot: Seamless Appointment Booking for Health
Project Description: DocSpot is a web-based healthcare appointment booking system developed using the MERN stack. The application allows users to register, log in, book doctor appointments, and track appointment status. It also provides an admin panel for managing doctors and approving appointments.
Project Version: Version 1.0
Testing Period: 10 February 2025 to 14 February 2025

Testing Scope:

Features Tested:
15. User Registration
16. User Login
17. Password Encryption
18. Dashboard Access
19. Doctor Listing
20. Appointment Booking
21. Appointment Status Viewing
22. Admin Login
23. Doctor Approval System

User Stories Tested:
24. As a user, I can register securely.
25. As a user, I can log in with valid credentials.
26. As a user, I can book an appointment.
27. As a user, I can view my appointment status.
28. As an admin, I can approve doctors.

Testing Environment:
URL/Location: http://localhost:3000
Backend Server: http://localhost:5000
Credentials:
User: testuser@gmail.com / password123
Admin: admin@gmail.com / admin123

Test Cases:

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
	User Registration	Enter valid details and click register	User account created successfully	Account created successfully	Pass
TC-002	User Login	Enter email & password	User redirected to dashboard	Redirected correctly	Pass
TC-003	Invalid Login	Enter wrong password	Error message displayed	Error message shown	Pass
TC-004	Book Appointment	Select doctor,date &time	Appointment booked successfully	Booking confirmed	Pass
TC-005	View Appointment	Click"My Appointments"	Appointment list displayed	Displayed correctly	Pass
TC-006	Admin Approval	Admin approves doctor	Status changes to Approved	Updated successfully	Pass

Bug Tracking:

Bug ID	Bug Description	Steps to reproduce	Severity	Status	Additional feedback
BG-001	Page refresh required to update appointment status	Book appointment and wait for approval	Low	Open	Can be improved using real-time updates
BG-002	Basic error message formatting	Enter invalid login details	Low	Closed	UI message improved

Bug Tracking:

Sign-off:
Tester Name: PANDIRI BHAVAJNA
Date: 14 February 2025
Signature: BHAVAJNA
Notes:

29. All major functionalities were tested for both positive and negative scenarios. The system performed as expected, and critical features such as authentication and appointment booking were validated successfully.
30. The application is ready for deployment with minor improvements suggested for future versions.

1. Introduction

Project Title: DocSpot: Seamless Appointment Booking For Health

Team Members:

- PANDIRI BHAVAJNA : *Backend Developer*
- IMANDI SOUMYA : *Frontend Developer*
- LANKA KEERTHI : *Database & Testing Manager*

2. Project Overview

Project Overview :

Purpose:

The purpose of DocSpot is to develop a web-based healthcare appointment booking system that simplifies and digitizes the medical consultation process.
DocSpot aims to eliminate traditional hospital appointment challenges such as long queues, unclear doctor availability, and manual record management.
The main goals of DocSpot are:

1. To provide an easy and quick online doctor booking system
2. To reduce waiting time in hospitals
3. To improve transparency in appointment scheduling
4. To enhance patient convenience
5. To create a secure and centralized healthcare platform

DocSpot ensures that patients can book appointments anytime and from anywhere.

Features :

DocSpot includes the following key features:

1. User Registration and Login
2. Secure Authentication System
3. Doctor Listing with Specialization Details
4. Online Appointment Booking
5. Appointment Status Tracking (Pending / Approved / Rejected)
6. Appointment History Management
7. Admin Dashboard for system control
8. Responsive and user-friendly interface

3. Architecture

Frontend :

The frontend of **DocSpot** is developed using **React.js** following a Single Page Application (SPA) approach.

1. Built using React.js for dynamic and responsive UI
2. Uses React Router for client-side routing
3. Communicates with backend using RESTful APIs
4. Axios is used for sending HTTP requests
5. Provides a user-friendly and responsive interface

Backend :

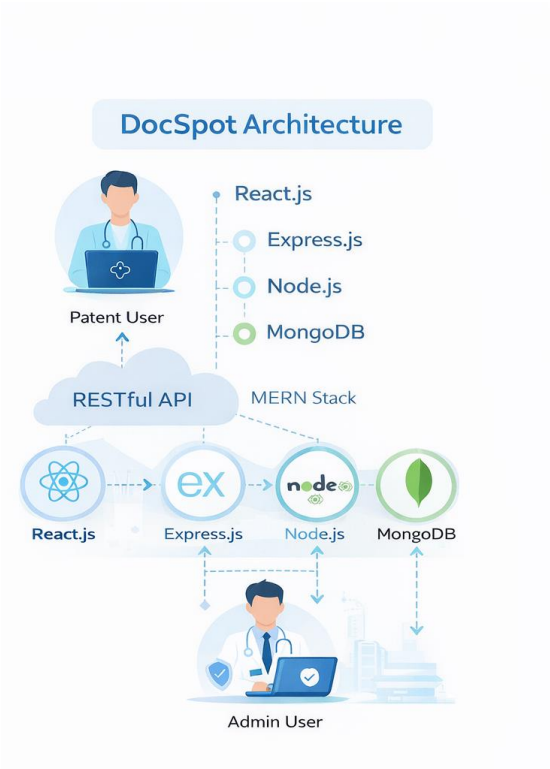
The backend of **DocSpot** is developed using **Node.js** with the **Express.js** framework.

- Built using Node.js and Express.js
- RESTful API endpoints handle client requests
- Business logic implemented on server side
- Authentication managed using JWT
- Middleware used for route protection and validation

Database :

DocSpot uses **MongoDB** as its database system.

- MongoDB stores application data
- Collections include: Users | Doctors | Appointments | Admins
- Mongoose is used for schema definition and data modeling
- CRUD operations are performed for managing records



4. Setup Instructions

Prerequisites:

Before running **DocSpot**, make sure the following software is installed:

1. Node.js (v16 or above recommended)
2. npm (comes with Node.js)
3. MongoDB (Local installation or MongoDB Atlas)
4. Git (for cloning the repository)
5. Code Editor (e.g., VS Code)

Installation:

Follow these steps to set up the project locally:

Step 1: Clone the Repository

Open terminal or command prompt and run:

git clone <repository-link>

Then navigate into the project folder:

cd docspot

Step 2: Install Backend Dependencies

cd server

npm install

Step 3: Install Frontend Dependencies

cd client

npm install

Step 4: Configure Environment Variables

*Create a .env file inside the **server** folder and add:*

MONGO_URL=your_mongodb_connection_string

JWT_SECRET=your_secret_key

PORT=5000

If using MongoDB Atlas, paste your connection string in MONGO_URL.

Step 5: Run the Application

Start Backend:

cd server

npm start

Start Frontend (open new terminal):

cd client

npm start

The application will run on:

http://localhost:3000

5. Folder Structure

Project Root Structure:

```
DocSpot/
|
├── backend/
└── frontend/
```

The project is divided into two main folders:

- **backend** → Handles server-side logic
- **frontend** → Handles user interface

Backend Structure:

```
backend/
|
├── config/
├── controllers/
├── middlewares/
├── node_modules/
├── routes/
├── schemas/
├── .env
├── index.js
└── package-lock.json
```


 config/
Contains configuration files such as database connection setup.
 controllers/
Contains business logic functions that handle request and response operations.
 middlewares/
Contains middleware functions such as authentication and route protection.
 routes/
Defines API endpoints for users, doctors, and appointments.
 schemas/
Contains MongoDB schema definitions for collections like Users, Doctors, and Appointments.
 .env
Stores environment variables such as MongoDB connection string and JWT secret.
 index.js
Main entry file of the backend server.
 package.json
Contains backend dependencies and scripts

Frontend Structure:

```
frontend/
│
├── node_modules/
├── public/
├── src/
├── .gitignore
├── package.json
├── package-lock.json
└── README.md
```

 public/
Contains static files like index.html.
 src/
Main React source folder containing components, pages, and routing logic.
 node_modules/
Installed frontend dependencies.
 package.json
Contains frontend dependencies and scripts.
 README.md
Project documentation file for frontend

6. Running the Application

Running the Application

To run the DocSpot application locally, both the frontend and backend servers must be started separately.

Start Backend Server

- Open terminal.
- Navigate to the backend folder:

cd backend

- Run the development server:

npm run dev

The backend server will start on:
http://localhost:5000

Start Frontend Server

1. Open a new terminal window.
2. Navigate to the frontend folder:

cd frontend

1. Start the React application:

npm start

The frontend application will run on:
http://localhost:3000

7. API Documentation

1.User APIs:

Register User:

Endpoint: /api/user/register
Method: POST
Request Body:

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "123456"
}
```

Example Response:

```
{
  "success": true,
  "message": "User registered successfully"
}
```

Login User:

Endpoint: /api/user/login
Method: POST
Request Body:

```
{
  "email": "john@example.com",
  "password": "123456"
}
```

Example Response:

```
{
  "success": true,
  "token": "jwt_token_here"
}
```

2.Doctor APIs:

Apply as Doctor:

Endpoint: /api/doctor/apply
Method: POST
Request Body:

```
{
  "name": "Dr. Smith",
  "specialization": "Cardiology",
  "experience": "5 Years"
}
```

Example Response:

```
{
  "success": true,
  "message": "Doctor application submitted"
}
```

Get All Doctors:

Endpoint: /api/doctor/get-all-doctors
Method: GET
Example Response:

```
{
  "success": true,
  "data": [
    {
      "name": "Dr. Smith",
      "specialization": "Cardiology"
    }
  ]
}
```

3.Appointment APIs:

Book Appointment:

Endpoint: /api/appointment/book
Method: POST
Request Body:

```
{
  "doctorId": "12345",
  "date": "2026-02-20",
  "time": "10:00 AM"
}
```

Example Response:

```
{
  "success": true,
  "message": "Appointment booked successfully"
}
```

Get User Appointments:

Endpoint: /api/appointment/user-appointments
Method: GET
Example Response:

```
{
  "success": true,
  "data": [
    {
      "doctor": "Dr. Smith",
      "date": "2026-02-20",
      "status": "Pending"
    }
  ]
}
```

Authentication:

All protected routes require a JWT token in the request header:
Authorization: Bearer <token>

8. Authentication

Authentication and authorization in **DocSpot** are implemented to ensure secure access to the application. The system uses **JWT (JSON Web Token)** based authentication for protecting routes and managing user sessions.

Authentication Process

1. The user registers or logs in using email and password.
2. The backend verifies the credentials.
3. If valid, a **JWT token** is generated.
4. The token is sent back to the client.
5. The token is stored on the client side (usually in localStorage).
6. For protected routes, the token is sent in the request header.

Password Security

- Passwords are encrypted using **bcrypt** before storing in the database.
- During login, the entered password is compared with the hashed password.
- This ensures that plain text passwords are never stored.

Authorization

Authorization ensures that only specific users can access certain routes.
DocSpot implements **role-based access control**:

- **User** → Can book appointments and view history.
- **Doctor** → Can manage appointment requests.
- **Admin** → Can approve doctors and manage users.

Middleware is used to:

1. Verify JWT token
2. Check user role
3. Allow or deny access to protected routes

Session Handling

DocSpot does not use traditional server-side sessions.
Instead, it uses **stateless authentication with JWT**, meaning:

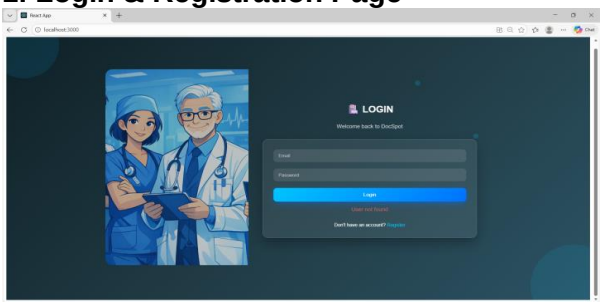
1. The server does not store session data.
2. Each request must include a valid token.
3. If the token expires, the user must log in again.

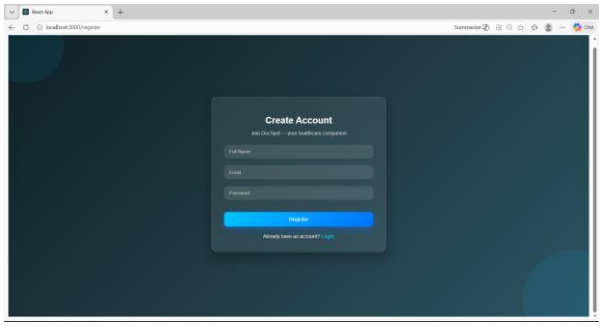
9. User Interface

The User Interface (UI) of DocSpot is designed to be clean, responsive, and user-friendly. The application follows a modern bluish theme and ensures smooth navigation for users, doctors, and administrators.

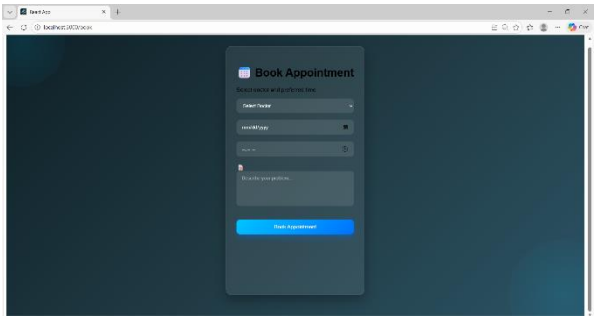
Below are screenshots showcasing different UI features of the system.

1. Login & Registration Page

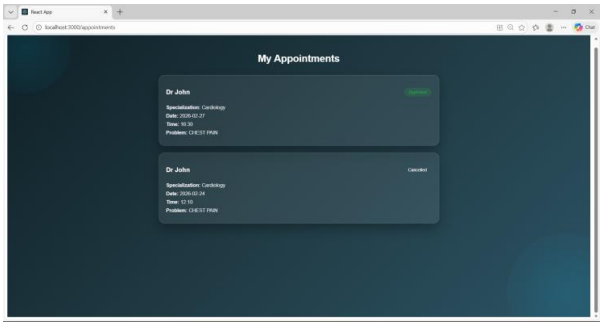




2. Doctor Listing Page



3. Appointment Booking Page



10. Testing

Testing is an important phase in the development of **DocSpot** to ensure the system functions correctly, securely, and efficiently. Different testing methods were applied to verify both frontend and backend functionalities.

1. Functional Testing

Functional testing was performed to verify that all features work as expected.

- User registration and login validation
- Doctor application submission
- Appointment booking process
- Appointment status updates
- Admin approval system

Each module was tested to ensure correct input handling and expected output responses.

2. API Testing

API testing was conducted using tools such as **Postman**.

- Verified all API endpoints
- Tested request and response formats
- Checked authentication token validation
- Validated error handling for invalid requests

3. Authentication Testing

1. Verified JWT token generation
2. Checked protected route access
3. Tested role-based access control
4. Ensured password encryption works properly

4. UI Testing

1. Checked responsiveness on different screen sizes
2. Verified navigation between pages
3. Tested form validations
4. Ensured proper error messages are displayed

5. Error Handling Testing

1. Tested invalid login credentials
2. Checked empty form submissions
3. Verified server error responses
4. Handled expired or invalid tokens

The testing process in DocSpot ensures that all system functionalities operate correctly and efficiently. Through functional, API, authentication, and UI testing, the application was verified to handle user interactions, appointment bookings, and role-based access securely and accurately.

Proper validation and error handling mechanisms were also tested to prevent system failures and unauthorized access. As a result, DocSpot provides a reliable, secure, and user-friendly healthcare appointment platform that performs consistently under normal usage conditions.

11. Screenshots or Demo

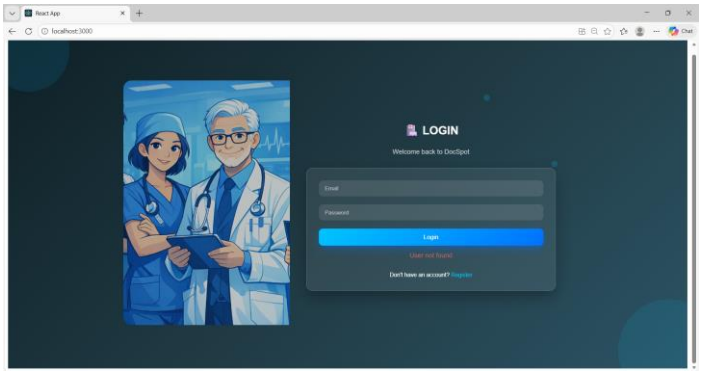
This section presents screenshots of the DocSpot application demonstrating its major functionalities and user interface components.

1. Login Page: The Login page allows registered users to securely access the DocSpot platform.

Features shown:

1. Email and Password input fields
2. Login button
3. Error message display ("User not found")
4. Register link for new users
5. Healthcare-themed illustration

This page ensures secure authentication and user validation.

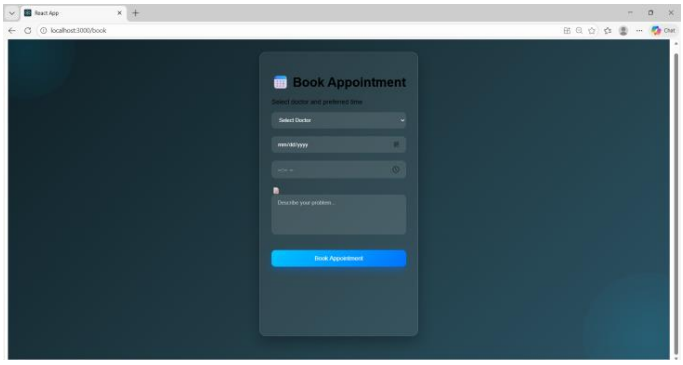


2. Registration Page: The Registration page allows new users to create an account.

Features shown:

1. Full Name input field
2. Email input field
3. Password input field
4. Register button
5. Login redirect option

This page securely collects user details for account creation.

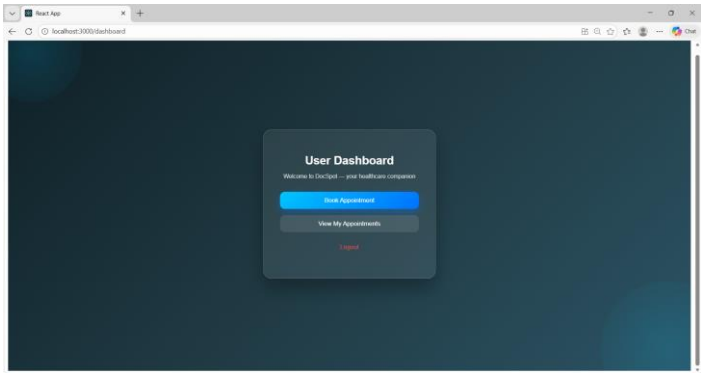


3. User Dashboard: The Dashboard acts as the central navigation page after login.

Features shown:

- Welcome message
- "Book Appointment" button
- "View My Appointments" button
- Logout option

It provides quick access to major functionalities of the application.

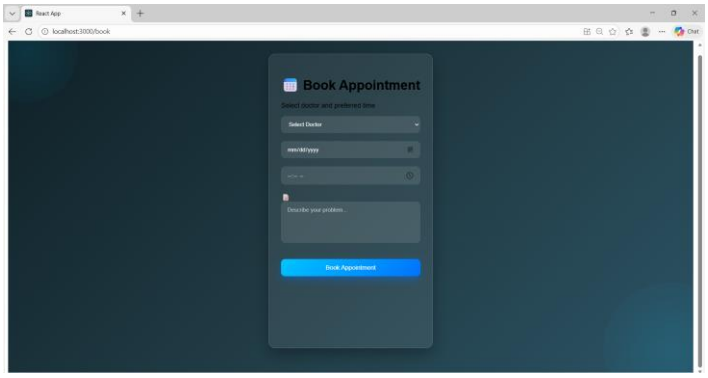


4. Book Appointment Page: This page allows users to schedule a consultation with a doctor.

Features shown:

1. Doctor selection dropdown
2. Date picker
3. Time selector
4. Problem description field
5. Book Appointment button

The interface is clean, responsive, and easy to use.

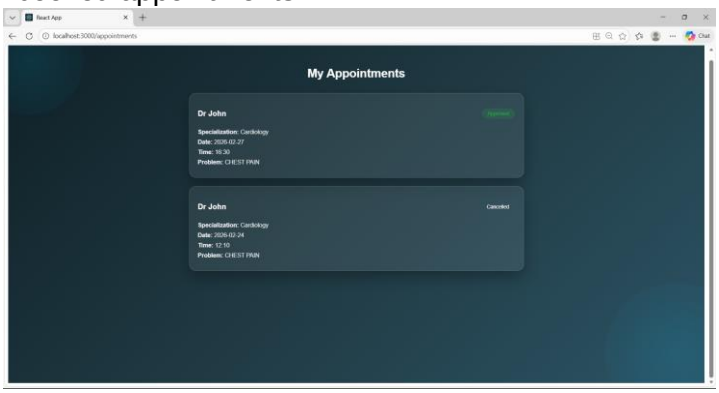


5.My Appointments Page: This page displays all booked appointments.

Features shown:

1. Doctor name and specialization
2. Appointment date and time
3. Problem description
4. Status indicator (Approved / Cancelled)

This page helps users track their appointment history and status.



12. Known Issues

While DocSpot is fully functional and stable for regular use, there are certain limitations and areas that can be improved in future versions.

- **No Email or SMS Notifications**

  The system does not currently send automated confirmation emails or reminders after booking an appointment. Users must manually check their appointment status.

- **No Real-Time Status Updates**

  Appointment approval or cancellation updates require manual page refresh. Real-time updates using WebSockets are not implemented.

- **Limited Doctor Filtering Options**

  The current system does not provide advanced filtering such as search by hospital location, ratings, or consultation fees.

- **No Online Payment Integration**

  There is no payment gateway integrated for paid consultations. All bookings are currently free and do not include transaction handling.

- **Basic Role Management**

  Role-based access control is implemented, but more granular permissions (like sub-admin roles) are not available.

- **Token Expiry Handling**

  If a JWT token expires, the user must manually log in again. Automatic token refresh functionality is not implemented.

- **No Appointment Rescheduling Feature**

  Users cannot modify an existing appointment. They must cancel and rebook instead.

- **Limited Admin Analytics**

  The admin dashboard does not currently include visual statistics, charts, or performance metrics.

- **No File Upload Support**

  Users cannot upload medical reports or documents while booking appointments.

- **Scalability Considerations**

  The application has been tested locally and is not yet optimized for high-traffic production deployment.

13. Future Enhancements

DocSpot has strong potential for expansion and improvement. The following enhancements can be implemented in future versions to improve functionality, scalability, and user experience.

- **Email and SMS Notifications**
  Implement automated appointment confirmations, reminders, and status updates through email and SMS services.
- **Real-Time Updates**
  Integrate WebSocket or real-time technologies to instantly reflect appointment approvals and cancellations without page refresh.
- **Online Payment Integration**
  Add payment gateway support (e.g., Razorpay, Stripe) to allow secure online consultation payments.
- **Advanced Doctor Search & Filters**
  Enable filtering based on specialization, experience, ratings, availability, and location.
- **Appointment Rescheduling Feature**
  Allow users to modify or reschedule appointments without canceling and rebooking.
- **Video Consultation Integration**
  Implement secure video calling functionality for online consultations.
- **Mobile Application Development**
  Develop Android and iOS mobile applications for better accessibility and convenience.
- **Admin Analytics Dashboard**
  Add charts and statistics for monitoring bookings, active users, and system performance.
- **File Upload Feature**
  Allow patients to upload medical reports or prescriptions during appointment booking.
- **AI-Based Doctor Recommendation System**
  Implement an intelligent system to suggest doctors based on user symptoms and history.

Conclusion

DocSpot is a full-stack healthcare appointment booking system developed using the MERN stack. The project successfully demonstrates the implementation of frontend and backend integration, secure authentication using JWT, and efficient data management using MongoDB. Through modules such as user registration, login, appointment booking, appointment tracking, and admin approval, the system provides a complete and functional digital healthcare solution.

The project highlights the practical application of agile planning, sprint management, testing, and user acceptance validation. Although there are areas for future enhancement, DocSpot effectively reduces manual effort, improves accessibility, and enhances user experience. Overall, the project achieves its objective of creating a secure, reliable, and user-friendly online doctor appointment platform.