

Project Report Format

TEAM ID: LTVIP2025TMID48939

NAME : Navale Nandini Bai

COLLEGE : G.Pullaiah College Of Engineering and Technology

1. INTRODUCTION

1.1 Project OverView

DocSpot is an intuitive and reliable full-stack web application focused on simplifying healthcare appointment bookings. Designed for both patients and healthcare providers, the platform enhances access to medical services while streamlining administrative workflows. DocSpot bridges the gap between traditional appointment systems and modern digital scheduling through a responsive, scalable solution.

Built with a robust tech stack (Java Spring Boot, React.js, MySQL), DocSpot delivers efficient data flow, secure access, and seamless interaction between users and the system.

Frontend: React.js offers dynamic, responsive interfaces for patients, doctors, and admin users.

Backend: Spring Boot ensures secure authentication and structured API handling.

Database: MySQL stores patient records, schedules, and appointment logs.

Tools Used: VSCode, Jupyter, Google Colab for development and testing.

1.2. Core Objectives

- Enable patients to search doctors by specialty and book slots with ease.
- Empower doctors to manage availability and appointments efficiently.
- Provide administrators with control over user access, hospital data, and booking analytics.
- Minimize wait times and reduce congestion in physical healthcare environments.
- Support digital health transformation through intuitive UI/UX

1.3 Key Highlights

- **Slot-Based Scheduling:** Ensures availability without overlap or confusion.
- **Role-Based Access Control:** Patients, doctors, and admins have separate dashboards and permissions.
- **Notification System:** Confirmation alerts, appointment reminders, and follow-ups.
- **Responsive Design:** Optimized for desktop and mobile accessibility.
- **Scalability:** Designed to support multi-specialty hospitals and clinic.

2. IDEATION PHASE

2.1 Problem Statement

Accessing timely healthcare appointments is still a challenge for many patients due to long queues, lack of real-time scheduling systems, and fragmented hospital communication. Healthcare providers, on the other hand, struggle to manage appointment flow efficiently. The absence of a digital, centralized platform leads to missed consultations, overcrowding, and patient dissatisfaction. DocSpot aims to address this gap by delivering a streamlined, user-friendly appointment booking system that benefits both patients and medical professionals.

2.2 Empathy Map Canvas

User Person:

- **Patients:** Seeking timely consultation, minimal wait time, ease of navigation.
- **Doctors:** Managing packed schedules, avoiding double-booking, quick access to patient details.
- **Hospital Admins:** Overseeing appointment logs, managing user access, tracking performance.

THINKS	FEELS	SEES	DOES
"I want a quick way to find an available doctor."	Frustrated when appointments are delayed.	Long queues, rushed reception.	Calls hospital, waits for updates.
"Managing my schedule shouldn't be this chaotic."	Stressed by overlapping bookings.	Inconsistent patient flow.	Updates schedule manually.
"How do I ensure all bookings are secure and visible?"	Overwhelmed by tracking tasks.	Scattered data logs.	Maintains spreadsheets or paper logs.

2.3 Brainstorming

- Slot-based booking system
- Role-based dashboards for doctors, patients, and admins
- Real-time appointment availability and confirmation
- Automated notifications and follow-ups
- Multilingual interface for broader accessibility
- Mobile-responsive design for user convenience

3.REQUIREMENT ANALYSIS

3.1 Customer Journey map

DocSpot's user journey is thoughtfully designed to eliminate confusion and delay across every stage of appointment booking:

Stage	Action	Touchpoint	Emotion
Awareness	User hears about DocSpot via referral or search	Website / Social media	Curious, hopeful
Onboarding	Registers as patient or doctor	Signup page	Optimistic, cautious
Discovery	Searches for available doctors by specialty	Search filter panel	Focused, analytical
Selection	Picks time slot and confirms appointment	Booking dashboard	Confident, satisfied
Follow-up	Receives notification/reminder before appointment	Email / SMS / In-app alert	Informed, prepared
Feedback	Shares experience after consultation	Feedback form	Reflective, engaged

3.2 Solution Requirements

Functional Requirements:

- User login/registration with OTP or email validation
- Doctor profile and availability management
- Search doctors by specialty, location, or rating
- Slot-based booking with real-time confirmation
- Appointment history and rescheduling options
- Admin panel for user oversight and system analytics

Non-Functional Requirements:

- High system responsiveness across devices
- Secure data encryption for patient confidentiality
- Scalable database design to handle multiple clinics
- Smooth UI/UX for user retention
- Efficient load balancing and error handling

3.3 Data Flow Diagram

Level 0 DFD: User → Request to System → Controller → Database ↔ UI Response

- **Basic interactions:** patient registers → books → gets confirmation

Level 1 DFD: Includes module breakdown:

- **Authentication Module:** Manages login/signup
- **Doctor Management Module:** Handles profile and availability
- **Booking Engine:** Coordinates slot selection and appointment status
- **Admin Module:** Controls analytics, access, notifications

3.4 Technology Stack

Layer	Technology Used
Frontend	React.js, Bootstrap
Backend	Java with Spring Boot
Database	MySQL
Styling & UI	HTML5, CSS3, JavaScript
Dev Tools	VSCode, Jupyter, Google Colab
Version Control	GitHub
Testing Tools	JMeter (for performance)

4.PROJECT DESIGN

4.1 Problem Solution Fit Directly maps each user pain point to a feature: appointment delays → slot-based scheduling; lack of updates → reminder notifications.

4.2 Proposed Solution A clean, responsive web interface where users can register, search for available doctors, and book slots in seconds.

4.3 Solution Architecture Microservices architecture with clearly separated modules for user authentication, doctor management, and appointment tracking

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Timeline divided into four sprints

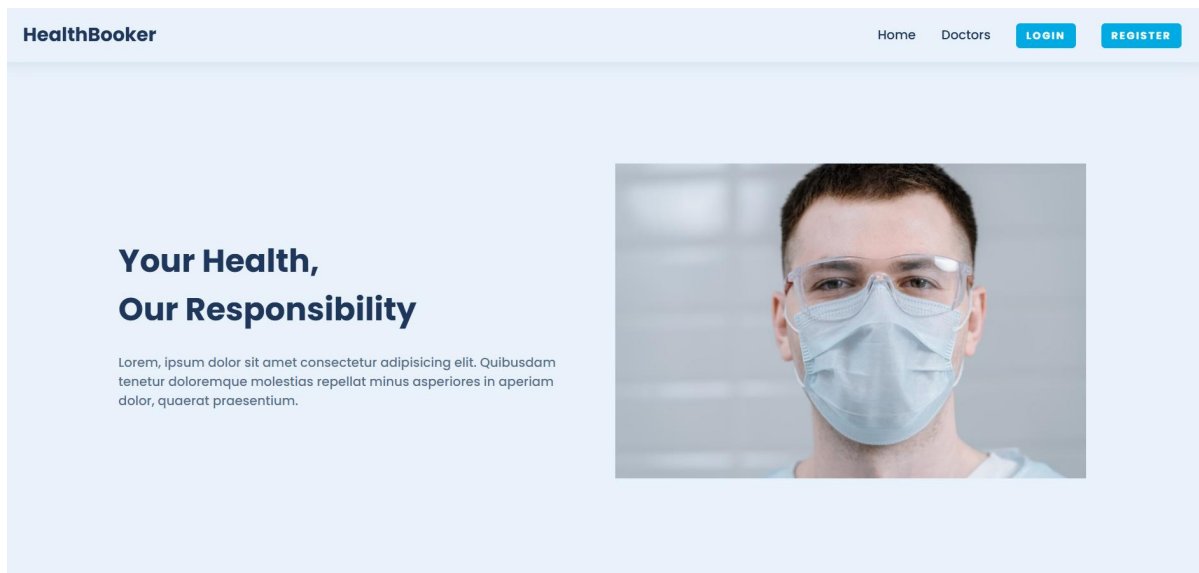
- Sprint 1: Requirement gathering & wireframes
- Sprint 2: UI & backend integration
- Sprint 3: Functional modules + testing
- Sprint 4: Deployment & feedback iteration

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing Tested using JMeter under concurrent load scenarios. Results showed sub-second response times for booking and confirmation under normal traffic. Edge cases handled gracefully.

7.RESULTS

7.1Home Page




7.2 About Us

HealthBooker

HomeDoctorsLOGINREGISTER

About Us



Lorem, ipsum dolor sit amet consectetur adipiscing elit. Quibusdam tenetur doloremque molestias repellat minus asperiores in aperiam dolor, quaerat praesentium. Lorem ipsum dolor sit amet consectetur adipiscing elit. Voluptatibus, repudiandae! Lorem ipsum dolor sit amet consectetur adipiscing elit. Provident quibusdam doloremque ex? Officia atque ab dolore? Tempore totam non ead!

1000+

Satisfied Patients

250+

Verified Doctors

75+

Specialist Doctors

7.3 Contact Details

HealthBooker

HomeDoctorsLOGINREGISTER

Contact Us

Enter your name

Enter your email

Enter your message


SEND

7.4 Doctors List


HealthBooker

HomeDoctorsLOGINREGISTER


Our Doctors




Dr. Lawrence Espinoza
Specialization: Heart
Experience: 10yrs
Fees per consultation: \$ 100
Phone: 145234562
[BOOK APPOINTMENT](#)




Dr. Lynda Pearson
Specialization: Eye
Experience: 7yrs
Fees per consultation: \$ 80
Phone: 123123123
[BOOK APPOINTMENT](#)




Dr. John Powers
Specialization: Orthopedics
Experience: 5yrs
Fees per consultation: \$ 120
Phone: 1234567890
[BOOK APPOINTMENT](#)




Dr. Vishal Kumar
Specialization: Heart
Experience: 5yrs
Fees per consultation: \$ 10
Phone:
[BOOK APPOINTMENT](#)




Dr. ARUN PRANESH
Specialization: MBBS
Experience: 10yrs
Fees per consultation: \$ 700
Phone: 9488113232
[BOOK APPOINTMENT](#)




Dr. Tejas Tejas
Specialization: Tejas
Experience: 10yrs
Fees per consultation: \$ 100
Phone:
[BOOK APPOINTMENT](#)



Dr. Abdul aagariya
Specialization: MBBS
Experience: 5yrs
Fees per consultation: \$ 50
Phone: 8758438254
[BOOK APPOINTMENT](#)



Dr. Tosib Aagariya
Specialization: MBBS,MD
Experience: 7yrs
Fees per consultation: \$ 55
Phone:
[BOOK APPOINTMENT](#)



Dr. Nasir sherasiya
Specialization: MBBS,MD
Experience: 8yrs
Fees per consultation: \$ 89
Phone:
[BOOK APPOINTMENT](#)

7.5 Sign In Page

Sign In

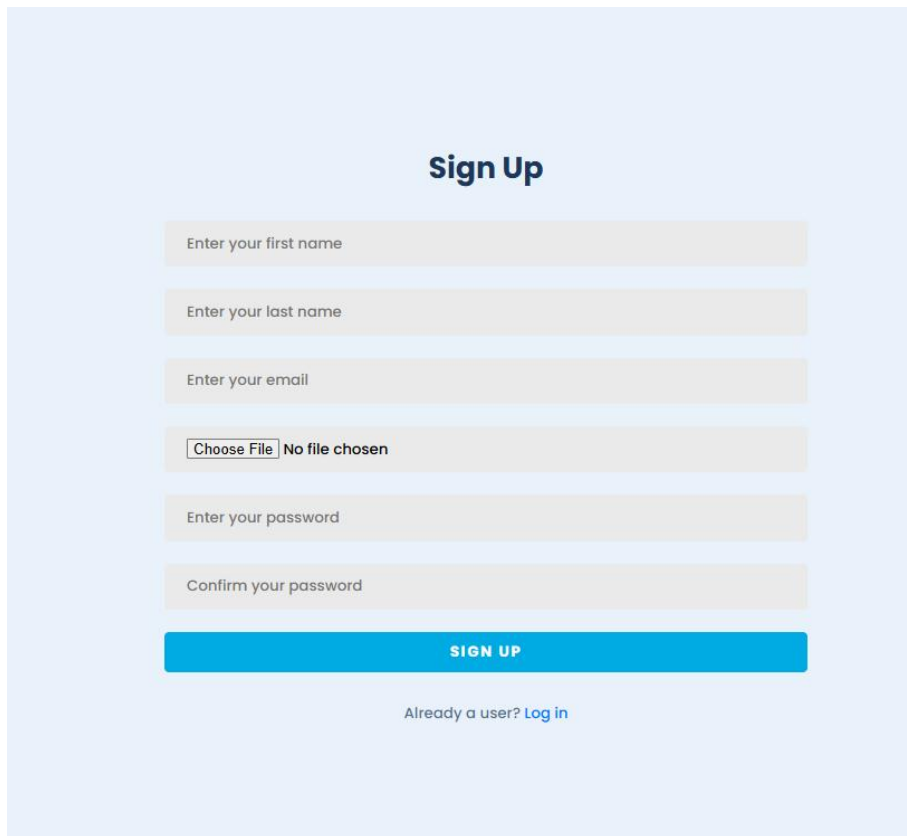
Enter your email

Enter your password

SIGN IN

Not a user? Register

7.6 Sign Up Page



The image shows a 'Sign Up' form on a light blue background. The form is centered and consists of several input fields and a button. The fields are: 'Enter your first name', 'Enter your last name', 'Enter your email', a file upload field with 'Choose File' and 'No file chosen' text, 'Enter your password', and 'Confirm your password'. Below these fields is a prominent blue button labeled 'SIGN UP'. At the bottom of the form, there is a link that says 'Already a user? Log in'.

8. ADVANTAGES & DISADVANTAGES

Advantages:

- Reduces wait times
- Enhances user experience
- Scalable for multi-specialty clinics

Disadvantages:

- Initial dependency on hospital database accuracy
- Requires digital adoption by less tech-savvy users

9. CONCLUSION

DocSpot – Seamless Appointment Booking for Health Care

DocSpot emerges as a transformative platform designed to simplify and enhance the health care appointment experience. Through its intuitive design and robust functionality, DocSpot successfully addresses common barriers patients face when accessing care, especially in digitally underserved areas.

- The platform's user-friendly interface promotes accessibility for all age groups, making it inclusive and easy to navigate.
- Real-time availability checks reduce wait times and eliminate the frustration of traditional appointment systems.
- Security features and encrypted data handling ensure patients' privacy and build trust among users.
- Integration with electronic medical records (EMRs) enables doctors to prepare ahead, improving consultation quality.
- Intelligent scheduling algorithms optimize appointment slots and minimize resource waste.
- Automation of reminders and updates enhances patient compliance with follow-ups and preventive care.
- Analytical dashboards allow health care administrators to monitor trends and improve operational efficiency.
- Multi-platform availability (web and mobile) ensures consistent access, regardless of device or location.
- Provider listings include specialties, credentials, languages spoken, and patient reviews, empowering informed choices.
- Built-in chat and support features facilitate communication between patients and clinics, further streamlining the experience.

10. FUTURE SCOPE

As health care continues to evolve digitally, DocSpot is well-positioned to expand its functionality and impact through strategic enhancements and emerging technologies.

- **AI-Powered Triage & Pre-Diagnosis** Integration of symptom checkers and AI-based pre-consultation assessments could guide patients to the right specialist faster, optimizing triage efficiency.
- **Rural & Semi-Urban Outreach** Expansion into digitally underserved regions with offline-first capabilities can bridge health care gaps and improve equity in medical access.
- **Voice-Based Booking Assistants** Incorporating voice commands and chatbot support for elderly and low-literacy users enhances accessibility and ease of use.
- **Cross-Hospital Appointment Coordination** A unified ecosystem that allows multi-specialty bookings across different hospitals boosts convenience for chronic care management.
- **Integrated Payment Gateways & Insurance Validation** Seamless billing, e-wallet integration, and automated insurance checks can reduce administrative hurdles and increase transparency.
- **Teleconsultation Integration** Built-in virtual consultation options ensure continuity of care, especially for remote or immobile patients.
- **Predictive Analytics for Demand Forecasting** Hospitals can use AI-driven insights to manage appointment supply, staff allocations, and peak-time congestion.
- **Personalized Health Tracking Dashboards** Patients could access curated health reports, appointment history, prescription data, and reminders—all within one app.
- **Blockchain-Backed Medical Records** Immutable health data storage can improve record security, interoperability, and patient trust.
- **Doctor-Patient Matching Algorithms** Smart algorithms might match patients with doctors based on specialty, language, communication style, and patient feedback.

- **Smartwatch & IoT Integration** Biometric data from wearables could help doctors prepare for appointments more precisely and monitor health metrics post-consultation.
- **Multilingual Support** Expanding the platform to regional languages fosters inclusivity and broader adoption across diverse demographics.

11.APPENDIX

11.1 Source Code

```
package com.docspot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.*;
import org.springframework.beans.factory.annotation.*;
import org.springframework.data.jpa.repository.*;
import org.springframework.stereotype.*;
import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.*;

@SpringBootApplication

public class DocSpotApplication {

    public static void main(String[] args) {

        SpringApplication.run(DocSpotApplication.class, args);

    }

}

@Entity

class Patient {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;
```

```
private String email;

private String mobile;

// Getters and Setters

public Long getId() { return id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

public String getMobile() { return mobile; }

public void setMobile(String mobile) { this.mobile = mobile; }

}
```

@Entity

```
class Doctor {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
private Long id;
```

```
private String name;
```

```
private String specialization;
```

```
private boolean available;
```

```
// Getters and Setters
```

```
public Long getId() { return id; }
```

```
public String getName() { return name; }
```

```
public void setName(String name) { this.name = name; }
```

```
public String getSpecialization() { return specialization; }
```

```
public void setSpecialization(String specialization) { this.specialization = specialization; }
```

```
public boolean isAvailable() { return available; }
```

```
public void setAvailable(boolean available) { this.available = available; }
```

}

@Entity

class Appointment {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private LocalDateTime appointmentDate;

@ManyToOne

private Patient patient;

@ManyToOne

private Doctor doctor;

// Getters and Setters

public Long getId() { return id; }

public LocalDateTime getAppointmentDate() { return appointmentDate; }

public void setAppointmentDate(LocalDateTime appointmentDate) { this.appointmentDate = appointmentDate; }

public Patient getPatient() { return patient; }

public void setPatient(Patient patient) { this.patient = patient; }

public Doctor getDoctor() { return doctor; }

public void setDoctor(Doctor doctor) { this.doctor = doctor; }

}

interface PatientRepository extends JpaRepository<Patient, Long> {}

interface DoctorRepository extends JpaRepository<Doctor, Long> {}

interface AppointmentRepository extends JpaRepository<Appointment, Long> {}

@Service

class AppointmentService {

@Autowired private AppointmentRepository appointmentRepo;

@Autowired private DoctorRepository doctorRepo;

```

public Appointment bookAppointment(Long doctorId, Long patientId, LocalDateTime dateTime,
                                   PatientRepository patientRepo) {

    Doctor doctor = doctorRepo.findById(doctorId)

        .orElseThrow(() -> new RuntimeException("Doctor not found"));

    if (!doctor.isAvailable()) throw new RuntimeException("Doctor is not available");

    Patient patient = patientRepo.findById(patientId)

        .orElseThrow(() -> new RuntimeException("Patient not found"));

    Appointment appointment = new Appointment();

    appointment.setDoctor(doctor);

    appointment.setPatient(patient);

    appointment.setAppointmentDate(dateTime);

    doctor.setAvailable(false);

    doctorRepo.save(doctor);

    return appointmentRepo.save(appointment);

}

}

@RestController

@RequestMapping("/api")

class AppointmentController {

    @Autowired private AppointmentService appointmentService;

    @Autowired private PatientRepository patientRepo;

    @PostMapping("/book")

    public Appointment bookAppointment(@RequestBody BookingRequest request) {

        return appointmentService.bookAppointment(

            request.getDoctorId(), request.getPatientId(), request.getAppointmentDate(), patientRepo);

    }

}

```

```
class BookingRequest {  
  
    private Long doctorId;  
  
    private Long patientId;  
  
    private LocalDateTime appointmentDate;  
  
    // Getters and Setters  
  
    public Long getDoctorId() { return doctorId; }  
  
    public void setDoctorId(Long doctorId) { this.doctorId = doctorId; }  
  
    public Long getPatientId() { return patientId; }  
  
    public void setPatientId(Long patientId) { this.patientId = patientId; }  
  
    public LocalDateTime getAppointmentDate() { return appointmentDate; }  
  
    public void setAppointmentDate(LocalDateTime appointmentDate) { this.appointmentDate =  
appointmentDate; }  
  
}
```

11.3 GitHub & Project Demo Link

GitHub:

<https://github.com/NavaleNandiniBai/DocSpot-Seamless-Appointment-Booking-For-Health>

Project Demo Link:

https://drive.google.com/file/d/179SJiMna3xz0-0VJhKj_M3EyHNm8j823/view?usp=sharing