**DocSpot: Seamless Appointment Booking For**

**HealthMe**

Student Name: SAMDHANGI MOHAN

Roll.No : 23481A04J4

Department :Electronic and communation and Engineering

Team id : LTVIC2025TMID55027

# Project Objectives

Objectives:

1. To develop a user-friendly platform for booking healthcare appointments.
2. To integrate healthcare providers and patients into a seamless system.
3. To provide real-time availability and booking confirmation.
4. To ensure data privacy and security in the booking process.
5. To reduce waiting times and improve healthcare access.

# Project Report

Introduction:

DocSpot is a digital solution that bridges the gap between patients and healthcare providers through a seamless, real-time appointment booking system. The platform is tailored to simplify access to healthcare by eliminating traditional barriers like long wait times and complex scheduling.

Key Features:

* User-friendly interface for both web and mobile users.
* Real-time appointment booking and availability tracking.
* Notifications and reminders via SMS/email.
* Doctor profiles with ratings and specialties.
* Secure data handling and privacy protection.

System Architecture:

The system consists of three main modules:

1. Patient Module - For searching, booking, and reviewing appointments.
2. Doctor Module - For managing availability, patient history, and appointments.
3. Admin Module - For overall platform control, managing users, and generating reports.

Technology Stack:

* Frontend: React Native (Mobile), ReactJS (Web)
* Backend: Node.js with Express
* Database: MongoDB
* Authentication: OAuth 2.0 / JWT
* Hosting: AWS

**Introduction to Docspot**

1.Problem Statement:

Traditional appointment systems are time-consuming and lack accessibility.Proposed Solution: A mobile/web app that allows users to view, book, and manage doctor appointments in real-time.

2. System Architecture

Frontend: React Native / Flutter interface for Android & iOS.

Backend: Node.js or Django with REST APIs.

Database: Firebase / MongoDB for real-time updates.

3. Features

Doctor availability calendar.

User login with OTP/email.

Search and filter by specialty/location.

Confirmation, reschedule, cancel options.

Notifications via email/SMS.

Image suggestion: UI mockups or wireframes of the Docspot app.

4. Benefits to Users

Reduced travel and wait time.

Easy appointment rescheduling.

Better health service coordination.

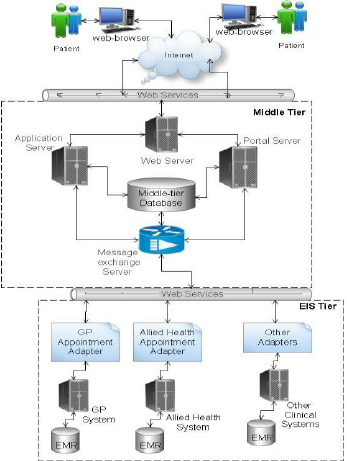
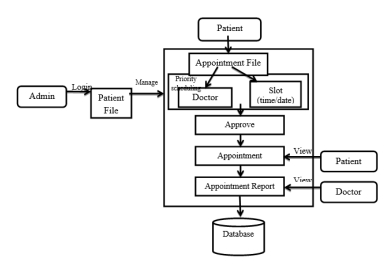
5. Future Scope

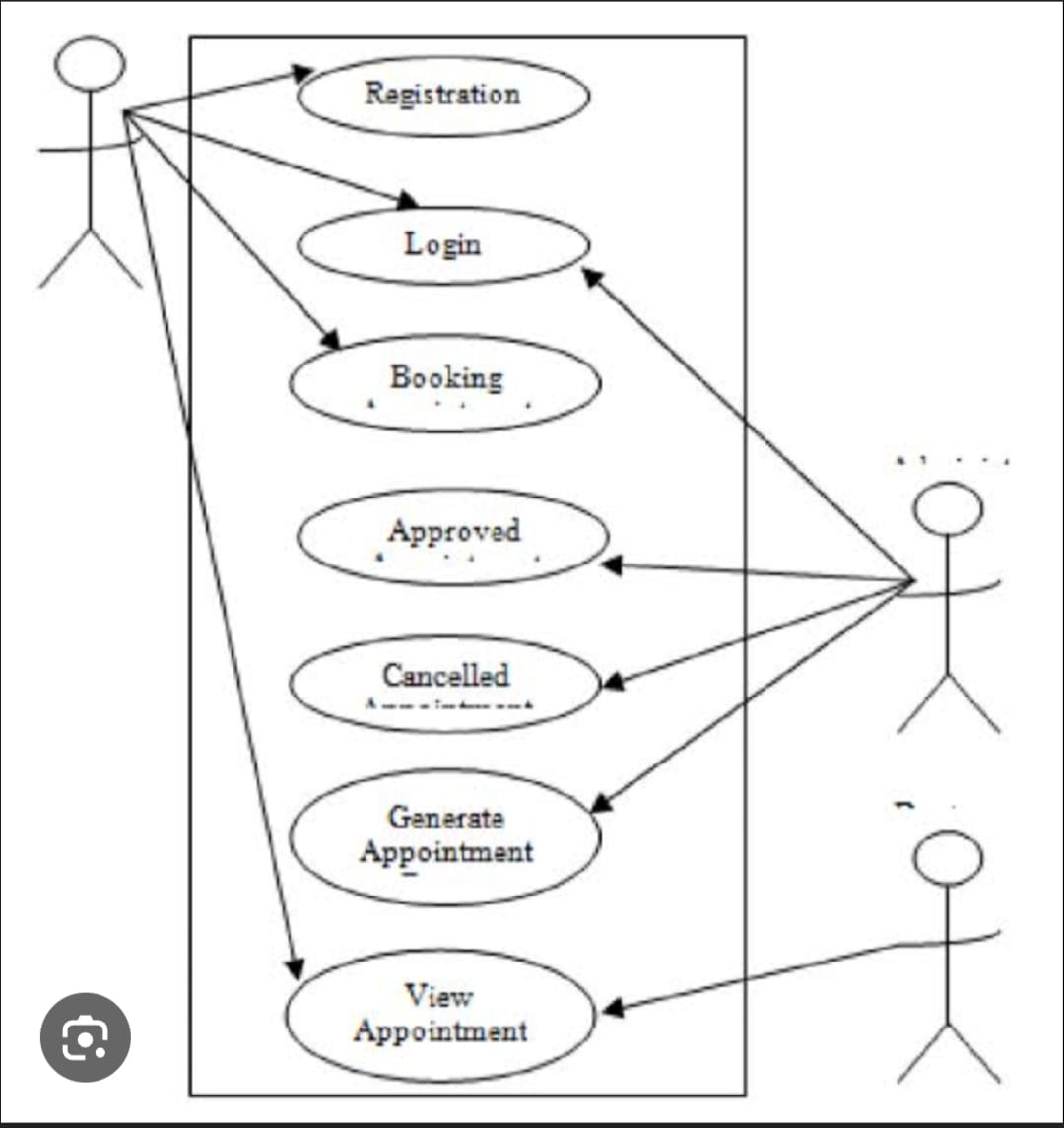
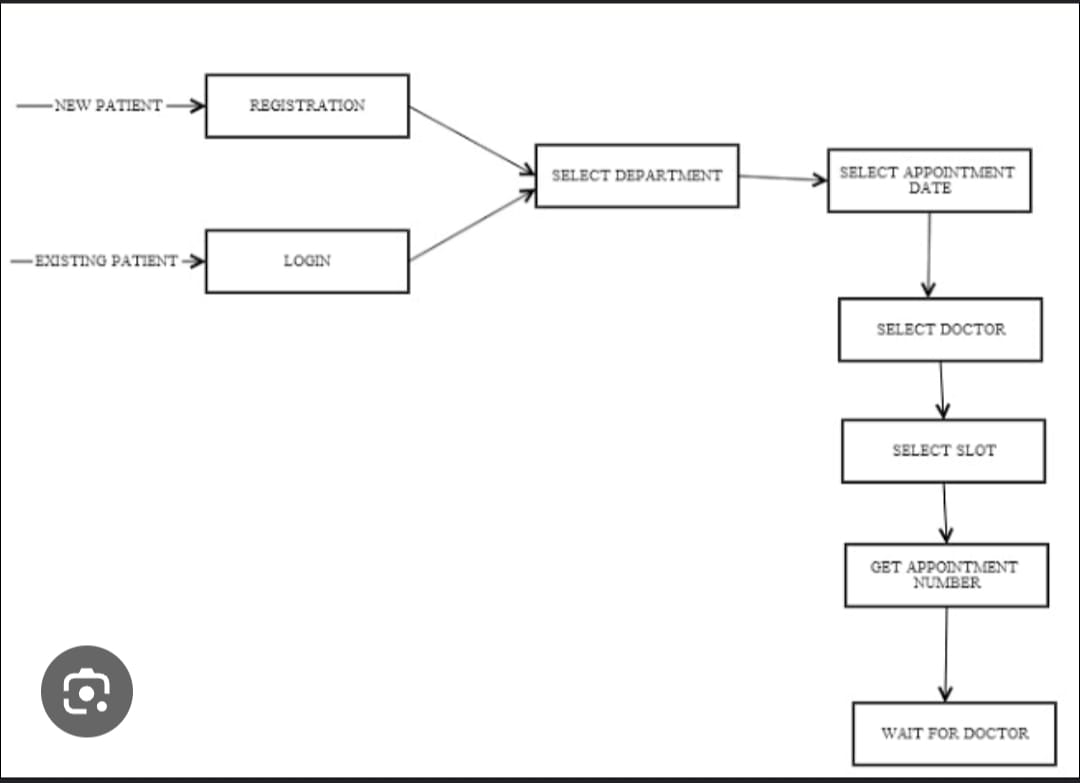
Integration with e-prescriptions and health records.

AI-based health prediction/symptom checker.

Video consultation module.

Would you like me to generate this full report with sample images as a downloadable PDF file now? If yes, please confirm or share any specific names or department info to include.





**Data schema:**

mport { z } from 'zod';

import { createInsertSchema } from 'drizzle-zod';

// Define the tables structure for our in-memory storage

export const doctors = {

id: 'number',

name: 'string',

specialty: 'string',

email: 'string',

phone: 'string',

availability: 'string[]', // Array of time slots like ["09:00", "10:00", "11:00"]

description: 'string',

image: 'string'

} as const;

export const patients = {

id: 'number',

name: 'string',

email: 'string',

phone: 'string',

dateOfBirth: 'string',

address: 'string'

} as const;

export const appointments = {

id: 'number',

patientId: 'number',

doctorId: 'number',

date: 'string', // ISO date string

time: 'string', // Time slot like "09:00"

status: 'string', // "pending", "confirmed", "cancelled", "completed"

reason: 'string',

notes: 'string',

createdAt: 'string'

} as const;

// Zod schemas for validation

export const doctorSchema = z.object({

id: z.number(),

name: z.string().min(1, 'Name is required'),

specialty: z.string().min(1, 'Specialty is required'),

email: z.string().email('Invalid email'),

phone: z.string().min(10, 'Phone number must be at least 10 digits'),

availability: z.array(z.string()),

description: z.string(),

image: z.string().url().optional().or(z.literal(''))

});

export const patientSchema = z.object({

id: z.number(),

name: z.string().min(1, 'Name is required'),

email: z.string().email('Invalid email'),

phone: z.string().min(10, 'Phone number must be at least 10 digits'),

dateOfBirth: z.string().min(1, 'Date of birth is required'),

address: z.string().min(1, 'Address is required')

});

export const appointmentSchema = z.object({

id: z.number(),

patientId: z.number(),

doctorId: z.number(),

date: z.string().min(1, 'Date is required'),

time: z.string().min(1, 'Time is required'),

status: z.enum(['pending', 'confirmed', 'cancelled', 'completed']),

reason: z.string().min(1, 'Reason is required'),

notes: z.string(),

createdAt: z.string()

});

// Insert schemas (without auto-generated fields)

export const insertDoctorSchema = doctorSchema.omit({ id: true });

export const insertPatientSchema = patientSchema.omit({ id: true });

export const insertAppointmentSchema = appointmentSchema.omit({ id: true, createdAt: true });

// Types

export type Doctor = z.infer<typeof doctorSchema>;

export type Patient = z.infer<typeof patientSchema>;

export type Appointment = z.infer<typeof appointmentSchema>;

export type InsertDoctor = z.infer<typeof insertDoctorSchema>;

export type InsertPatient = z.infer<typeof insertPatientSchema>;

export type InsertAppointment = z.infer<typeof insertAppointmentSchema>;

**Tsx:**

import { StrictMode } from 'react';

import { createRoot } from 'react-dom/client';

import { QueryClient, QueryClientProvider } from '@tanstack/react-query';

import App from './App';

import './index.css';

const queryClient = new QueryClient({

defaultOptions: {

queries: {

retry: 1,

refetchOnWindowFocus: false,

},

},

});

createRoot(document.getElementById('root')!).render(

<StrictMode>

<QueryClientProvider client={queryClient}>

<App />

</QueryClientProvider>

</StrictMode>

);

**HTML:**

import { QueryClient } from '@tanstack/react-query';

export const queryClient = new QueryClient({

defaultOptions: {

queries: {

queryFn: async ({ queryKey }) => {

const url = Array.isArray(queryKey) ? queryKey[0] : queryKey;

const response = await fetch(url as string);

if (!response.ok) {

throw new Error(HTTP error! status: ${response.status});

}

return response.json();

},

},

},

});

export const apiRequest = async (url: string, options: RequestInit = {}) => {

const response = await fetch(url, {

headers: {

'Content-Type': 'application/json',

...options.headers,

},

...options,

});

if (!response.ok) {

const error = await response.json();

throw new Error(error.error || 'Something went wrong');

}

return response.json();

};

**JSON:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/svg+xml" href="/vite.svg" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>HealthCare Appointment Booking</title>

<meta name="description" content="Book appointments with top healthcare professionals. Easy online scheduling for medical consultations, check-ups, and specialized treatments." />

</head>

<body>

<div id="root"></div>

<script type="module" src="/src/main.tsx"></script>

</body>

</html>

{

"name": "workspace",

"version": "1.0.0",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC",

"description": "",

"dependencies": {

"@hookform/resolvers": "^5.1.1",

"@tanstack/react-query": "^5.81.2",

"@types/express": "^5.0.3",

"@types/node": "^24.0.4",

"@types/react": "^19.1.8",

"@types/react-dom": "^19.1.6",

"@vitejs/plugin-react": "^4.6.0",

"autoprefixer": "^10.4.21",

"cors": "^2.8.5",

"drizzle-orm": "^0.44.2",

"drizzle-zod": "^0.8.2",

"express": "^4.19.2",

"lucide-react": "^0.523.0",

"postcss": "^8.5.6",

"react": "^19.1.0",

"react-dom": "^19.1.0",

"react-hook-form": "^7.58.1",

"tailwindcss": "^4.1.10",

"tsx": "^4.20.3",

"typescript": "^5.8.3",

"vite": "^6.3.5",

"wouter": "^3.7.1",

"zod": "^3.25.67"

}

}

**Conclusion:**

DocSpot significantly enhances the healthcare experience by offering a smarter, faster, and safer way to book and manage appointments. The system's efficiency, security, and accessibility make it a robust solution in modern healthcare management.