

# **MYDOC**

**A Project Report for Industrial Training and Internship**

**submitted by**

**TIASA MAHANTY  
RITAM CHAKRABORTY  
SWAGNIK GHOSH  
OINDRILLA MISHRA**

*In the partial fulfillment of the award of the degree of*

**B. Tech**

in the

**Department of Electronics and Communication Engineering**

**Of**

**B. P. Poddar Institute of Management and Technology**



At

**Ardent Computech Pvt. Ltd.**





# Ardent Computech Pvt. Ltd.

*Drives you to the Industry*

Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091  
[www.ardentcollaborations.com](http://www.ardentcollaborations.com)



## CERTIFICATE FROM SUPERVISOR

This is to certify that **Tiasa Mahanty, Ritam Chakraborty, Swagnik Ghosh, Oindrilla Mishra, 11500322044, 11500322042, 11500322038, 11500322050** have completed the project titled **MyDoc** under my supervision during the period from 4/7/25 to 16/8/25 which is in partial fulfillment of requirements for the award of the **B. Tech** degree and submitted to the Department of **B. Tech (ECE)** of **B.P.Poddar Institute of Management and Technology**.

---

### Signature of the Supervisor

Date: 17/08/25

Name of the Project Supervisor:





## Ardent Computech Pvt. Ltd. Drives you to the Industry

Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091  
[www.ardentcollaborations.com](http://www.ardentcollaborations.com)



## BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

*MyDoc* is the bonafide work of

**Name of the student:** *Tiasa Mahanty*

**Signature:**

**Name of the student:** *Ritam Chakraborty*

**Signature:**

**Name of the student:** *Swagnik Ghosh*

**Name of the student:** *Oindrilla Mishra*

### SIGNATURE

Name :

### PROJECT MENTOR

### SIGNATURE

Name:

### EXAMINERS

## Ardent Original Seal



## ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, **Mr. Subhajit Santra** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of **Ardent Computech Pvt. Ltd.** for their support.

# CONTENT PAGE

<b>1. COMPANY PROFILE</b>	<b>1</b>
<b>2. INTRODUCTION</b>	<b>2</b>
2A. OBJECTIVE -----	3
2B. SCOPE -----	4
1. User Registration & Login: -----	4
2. Doctor Profiles & Search: -----	4
3. Medical Records: -----	4
4. Appointment Booking & Slot Management:	4
5. Payment : -----	4
6. Admin Dashboard : -----	4
7. Cross-Platform Access-----	4
<b>3. SYSTEM ANALYSIS</b>	<b>5</b>
3A. IDENTIFICATION OF NEED -----	6
1. Accessibility to Healthcare:-----	6
2. Efficient Consultation Management: 6	
3. Integrated Communication Tools: 6	
4. Secure Medical Record Handling: 6-----	
3B.FEASIBILITY STUDY-----	7
3C.WORKFLOW-----	8
Waterfall Model Design: -----	8
Iterative Waterfall Design: -----	8
▪ Advantages:-----	9
▪ Disadvantages:-----	9
▪ Applications:-----	9
3D. STUDY OF THE SYSTEMS-----	10
• Patient & Doctor Registration Module:	10
• Login Module:	10
• Appointment Management Module:	10
• Payment Module: -----	10
• Admin Module:	11
3E.INPUT AND OUTPUT-----	12
INPUT: -----	12
OUTPUT: -----	12
<b>3F.SOFTWARE REQUIREMENT SPECIFICATIONS</b>	<b>13</b>
The developer is responsible for:-----	13
Functional Requirements:-----	13
B. Browse and Search:-----	13

C. Courses Display:.....	13
Hardware Requirements:.....	13
Software Requirements:.....	13
3G. SOFTWARE ENGINEERING PARADIGM APPLIED	14
<b>4. SYSTEM DESIGN</b>	<b>15</b>
<b>4A. DATA FLOW DIAGRAM</b>	<b>16</b>
DFD Notation:.....	17
DFD Example:.....	17
Database Input Output.....	17
Rules for constructing a Data Flow Diagram	18
● LEVEL 0 DFD OR CONTEXT DIAGRAM:.....	18
● LEVEL 1 DFD:.....	19
● LEVEL 1 DFD:.....	20
<b>4B. SEQUENCE DIAGRAM</b>	<b>21</b>
How to draw a Use Case Diagram?.....	25
<b>4D. SCHEMA DIAGRAM</b>	<b>27</b>
<b>5. UI SNAPSHOT</b>	<b>28</b>
❖ FRONTEND : .....	28
Login and register page: .....	28
✓ CODE.....	28
Main Page: .....	32
✓ CODE.....	33
3) ALL DOCTORS PAGE(for user)	45
4) ABOUT US:	48
✓ CODE.....	48
5) CONTACT US PAGE: .....	52
6) MY PROFILE PAGE:.....	54
7) MY APPOINTMENTS PAGE:	59
✓ CODE:.....	59
8) LOGIN PAGE (for admin and doctor):.....	64
✓ CODE:.....	64
9) ADMIN DASHBOAED PAGE.....	67
10) ADMIN ALL APPOINTMENT PAGE:.....	72
11)ADMIN-ADD-DOCTORS-PAGE.....	75
✓ CODE.....	75
12) ADMIN DOCTOR LIST PAGE:.....	81
✓ CODE:.....	81
13) DOCTOR DASHBOARD PAGE	83
✓ CODE:.....	83
14) APPOINTMENT PAGE FOR DOCTOR:	89
❖ BACKEND:.....	100
1) USER AND ADMIN DATA.....	100

2) DOCTOR DATA:.....	102
3) APPOINTMENT DATA.....	103
<b>6. CONCLUSION</b>	104
<b>7. FUTURE SCOPE &amp; FURTHER ENHANCEMENTS</b>	105
❖ Future scope:.....	105
❖ Further enhancement:.....	106
<b>8. BIBLIOGRAPHY</b>	107

## **1. COMPANY PROFILE**

ARDENT (Ardent Computech Pvt. Ltd.), formerly known as Ardent Computech Private Limited, is an ISO 9001:2015 certified Software Development and Training Company based in India. Operating independently since 2003, the organization has recently undergone a strategic merger with ARDENT Technologies, enhancing its global outreach and service offerings.

### **ARDENT Technologies**

ARDENT Technologies delivers high-end IT services across the UK, USA, Canada, and India. Its core competencies lie in the development of customized application software, encompassing end-to-end solutions including system analysis, design, development, implementation, and training. The company also provides expert consultancy and electronic security solutions. Its clientele spans educational institutions, entertainment companies, resorts, theme parks, the service industry, telecom operators, media, and diverse business sectors.

### **ARDENT Collaborations**

ARDENT Collaborations, the Research, Training, and Development division of ARDENT (Ardent Computech Pvt. Ltd.), offers professional IT-enabled services and industrial training programs. These are tailored for freshers and professionals from B.Tech, M.Tech, MBA, MCA, BCA, and MSc backgrounds. ARDENT (Ardent Computech Pvt. Ltd.) provides Summer Training, Winter Training, and Industrial Training to eligible candidates. High-performing students may qualify for stipends, scholarships, and additional benefits based on performance and mentor recommendations.

### **Associations and Accreditations**

ARDENT (Ardent Computech Pvt. Ltd.) is affiliated with the National Council of Vocational Training (NCVT) under the Directorate General of Employment & Training (DGET), Ministry of Labour & Employment, Government of India. The institution upholds strict quality standards under ISO 9001:2015 certification and is dedicated to bridging the gap between academic knowledge and industry skills through innovative training programs.

## **2. INTRODUCTION**

In today's healthcare system, accessibility, speed, and efficiency play a key role in keeping patients happy and ensuring effective medical services. Traditional appointment booking methods, like walk-ins and phone reservations, often lead to long waits, scheduling issues, and limited communication between patients and healthcare providers. These problems show the need for tech-based solutions for scheduling appointments and interacting with doctors.

MyDoc, the Doctor Appointment Booking System, is a full-stack MERN application that connects patients and healthcare professionals. The platform allows patients to register, search for doctors by specialty and location, see available consultation slots in real-time, and book appointments easily.

The system has three main user roles: Admin, Doctor, and Patient, each with its own dashboard and permissions. Patients can manage their health records and appointment histories, doctors can handle their schedules and patient consultations, and admins can oversee user management, appointments, and payment processing.

With built-in payment gateways, automated reminders, real-time notifications, and secure medical data storage, MyDoc aims to make healthcare services more accessible, efficient, and easy to use. The platform is responsive, works on various devices, and provides a solid foundation for future features like AI-based doctor recommendations and support for multiple languages.

## **2A.OBJECTIVE**

The main objectives of the MyDoc Doctor Appointment Booking System is to create a centralized, secure, and user-friendly platform that simplifies the process of scheduling, managing, and conducting medical consultations.

- Allows patients to search for doctors and book appointments online.
- Enables doctors to manage schedules and consultation slots.
- Supports teleconsultations via secure video calls and chat.
- Maintains patient medical history, prescriptions, and reports.
- Integrates online payment for consultation fees.
- Sends automated reminders and notifications.
- Ensures data privacy through encryption and role-based access.
- Provides dedicated dashboards for Admin, Doctor, and Patient.

## **2B.SCOPE**

Our project aims to create a mobile or web application that links patients with doctors through a central platform. This will allow for online appointment booking, teleconsultations, and secure management of medical records. The system is built to serve patients of all ages and healthcare needs, as well as doctors from different specialties and practice settings.

The scope of the MyDoc – Doctor Appointment Booking System includes:

1. **User Registration & Authentication** – Secure sign-up/login for patients, doctors, and admins with optional social media integration.
2. **Doctor Profiles & Search** – Detailed doctor information with search and filter options by specialty, location, availability, and ratings.
3. **Appointment Booking & Slot Management** – Real-time slot viewing, booking, rescheduling, and cancellation.
4. **Medical Records** – Secure storage and access to patient history, reports.
5. **Payment Integration** – Multiple online payment options with invoice generation.
6. **Admin Dashboard** – User and appointment management, reports, and statistics.
7. **Cross-Platform Access** – Responsive design accessible via web, mobile.

### **3. SYSTEM ANALYSIS**

## **3A.IDENTIFICATION OF NEED**

The development of the **MyDoc Doctor Appointment Booking System** addresses the growing demand for efficient, accessible, and secure healthcare services in the digital era. Traditional appointment booking often involves physical visits or lengthy phone calls, leading to patient inconvenience, scheduling conflicts, and administrative inefficiencies.

With the rapid adoption of smartphones, reliable internet access, and increased familiarity with online platforms, there is a clear need for a **centralized healthcare booking solution** that caters to both **patients** and **doctors** while ensuring **data security and ease of use**.

### **Key Needs Identified:**

#### **1. Accessibility to Healthcare:**

- Patients in rural or remote areas require a way to connect with qualified doctors without traveling long distances.
- Flexible booking options allow patients to schedule appointments according to their convenience.

#### **2. Efficient Consultation Management:**

- Doctors need a reliable system to manage appointment slots and reduce scheduling conflicts.
- Automated reminders can minimize no-shows and improve time management.

#### **3. Secure Medical Record Handling:**

- Centralized storage of patient history, and reports facilitates better treatment decisions.

### **3B. FEASIBILITY STUDY**

A feasibility study was conducted to check whether the MyDoc – Doctor Appointment Booking System can be developed successfully and used effectively.

From a technical point of view, the system is built using the MERN stack; MongoDB, Express.js, React.js with Vite, and Node.js, which provides speed, flexibility, and the ability to handle many users at the same time. Security is maintained through JWT authentication for safe logins. For payments, trusted gateways like Razorpay and Stripe are included.

From an economic perspective, the cost of development is low because open-source technologies are used. The system also has opportunities to earn revenue through consultation fees, and premium services.

From an operational perspective, the design is simple and user-friendly, so patients, doctors, and administrators can use it easily. Automated processes help reduce manual work and save time.

## **3C.WORKFLOW**

This document plays a vital role in the Software Development Life Cycle (SDLC) as it outlines the complete workflow and methodology used for developing the MyDoc – Doctor Appointment Booking System. It serves as a reference for the development team and is essential during the testing phase. Any future changes to the workflow or requirements will follow a formal change approval process.

The Waterfall Model was the first process model introduced in software engineering, also referred to as a linear-sequential life cycle model. It is simple to understand and use. In this model, each phase must be completed before moving to the next, with no overlapping phases. The Waterfall Model illustrates the software development process in a linear sequential flow, meaning a phase begins only after the previous phase is completed.

### **Waterfall Model Design:**

The Waterfall approach was one of the earliest and most widely used SDLC models. In this approach, the software development process is divided into distinct phases, where the output of one phase serves as the input for the next.

### **Iterative Waterfall Design:**

The Iterative Waterfall Model is a variation of the traditional Waterfall Model. It follows the same linear and sequential approach but introduces flexibility by allowing phases to be revisited and refined before moving forward. This combination of structure and adaptability ensures that feedback can be incorporated without restarting the entire process.

The sequential phases in the Iterative Waterfall Model for MyDoc are:

**Requirement Gathering and Analysis:** Gathering complete details from stakeholders (patients, doctors, admins) about system needs such as appointment booking, teleconsultations, payments, and medical record storage. These are documented in the requirement specification document.

**System Design:** Using the documented requirements to create the system's architecture, database schema, and user interface layouts. This stage defines both hardware and software requirements.

**Implementation:** Developing the system in smaller modules such as registration, slot booking, and payment processing. Each module is tested individually (unit testing).

**Integration and Testing:** Integrating all modules into the complete system and testing for errors, performance, and compliance with requirements.

**Deployment of the System:** Once testing is complete, the system is deployed for real-world use via the web and mobile platforms.

**Maintenance:** Fixing bugs, releasing patches, and adding new features based on feedback and evolving

healthcare needs.

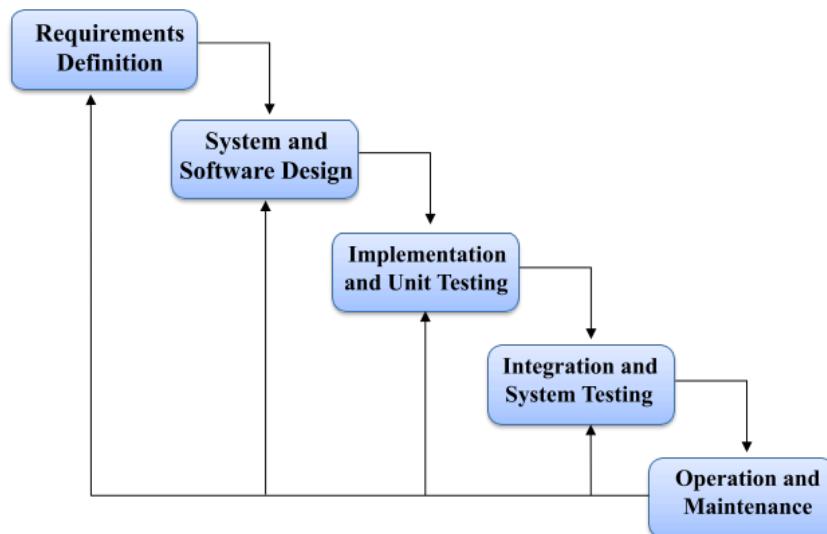
In this model, the phases are sequential but can be revisited in later iterations to make improvements, hence the term “Iterative Waterfall Model.”

- **Advantages:**

1. Flexibility: Iterations allow changes based on feedback.
2. Early Delivery: Core features can be released in stages for testing and feedback.
3. Risk Management: Problems can be identified and resolved early.

- **Disadvantages:**

1. Increased Complexity: Iterations add more steps to the process.
2. Potential for Scope Creep: Frequent changes may expand the project’s scope.
3. Resource Intensive: Revisiting stages may require additional time and effort.



- **Applications:**

The Iterative Waterfall Model is ideal for projects where requirements may evolve over time and feedback is essential. It is especially suitable for healthcare systems like MyDoc, where user feedback from patients and doctors can significantly improve the platform. Partial delivery of core features allows stakeholders to test functionalities early and suggest enhancements.

### **3D .STUDY OF THE SYSTEM**

**Modules:** The modules used in this software are as follows:

#### 1. Patient & Doctor Registration Module:

- Patient & Registration: Patients register by providing personal details such as name, email, password.
- Doctor Registration: Doctors registration is done by Admin providing personal details, qualifications, specialties, consultation fees, address, and available timings.

#### 2. Login Module:

- Patient Login: Patients log in to search doctors, book appointments, update their profile.
- Doctor Login: Doctors log in to manage schedules, accept/reject appointments, and update their profile.
- Admin Login: Admins login to manage appointments, register doctors and databases.

#### 3. Appointment Management Module:

- Patient Interface:
  - Search and filter doctors by name & specialty.
  - Book cancel appointments.
  - View appointment history and upcoming schedules.
- Doctor Interface:
  - Define and manage available time slots.
  - Accept or reject patient appointment requests,
  - View appointment list, total earnings
  - Update own profile.

#### 4. Payment Module:

- Integrates with payment gateways (Razorpay) for consultation fee payments.

- Generates payment receipts and stores transaction history for both patients and doctors.

## 5. Admin Module:

- Manage all registered patients and doctors.
- Can register new doctors.
- Monitor appointment statistics, payments, and overall system usage.
- Generate analytical reports for performance tracking

### **3E.INPUT AND OUTPUT**

The main inputs, outputs and the major function the details are:

**INPUT:**

1. Users can log in by entering their **credentials** on the login page such as name ,email,password etc.

**OUTPUT:**

1. Users can view the **available doctors, their slots ,book their appointments and go to the clinic.**
2. The **admin** can access a **centralized database** that includes **details of doctors, the appointments booked for each doctor, fees transaction** etc ensuring efficient system management.

## **3F.SOFTWARE REQUIREMENT SPECIFICATIONS**

Software Requirements Specification provides an overview of the entire project. It is a description of a software system to be developed, laying out functional and non-functional requirements. The software requirements specification document enlists enough necessary requirements that are required for the project development. To derive the requirements we need to have a clear and thorough understanding of the project to be developed. This is prepared after detailed communication with the project team and the customer.

### **The developer is responsible for:**

- Developing the system, which meets the SRS and solves all the requirements of the system.
- Demonstrating the system and installing the system at the client's location after acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.

### **Functional Requirements:**

#### **A. User Registration and Authentication:**

1. Patients should be able to login securely.
2. The system should authenticate the patients and manage login sessions.

#### **B. Appointment Booking System:**

1. This allows doctors to define and manage available time slots for consultations.
2. Doctors can accept or reject appointment request.

#### **C. Doctor and Clinic Management:**

1. This will create detailed profiles for healthcare providers including specialities, qualifications, experience, consultation fees and availability.
2. Displays reviews, ratings and patients feedback to aid in doctor selection .
3. Allow search and filter of doctors by speciality, location, availability, language and rating.

### **Hardware Requirements:**

- 1 . Computer has Intel I3 Processor
- 2 . 8 GB RAM
4. SSD-ROM Drive

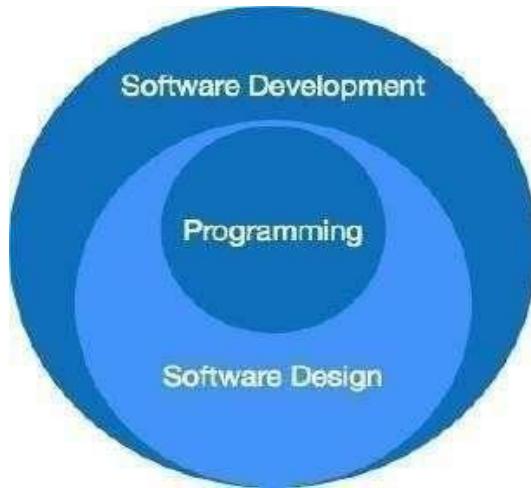
### **Software Requirements:**

1. Windows 11 OS
2. Visual Studio Code

### 3.Mongo DB Atlas

### **3G.SOFTWARE ENGINEERING PARADIGM APPLIED**

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another.



The programming paradigm is a subset of Software design paradigm which is further a subset of the Software development paradigm.

There are two levels of reliability. The first is meeting the right requirements. A careful and thorough systems study is needed to satisfy this aspect of reliability. The second level of systems reliability involves the actual work delivered to the user. At this level, the system's reliability is interwoven with software engineering and development.

There are three approaches to reliability.

**1. Error avoidance:** Prevents errors from occurring in software.

**2.Error detection and correction:** In this approach, errors are recognized whenever they are encountered, and correcting the error by the effect of the error of the system does not fail.

**3.Error to Clearance:** In this approach, errors are recognized whenever they occur, but enables the system to keep running through degraded performance or Applying values that instruct the system to continue process.

#### **4. SYSTEM DESIGN:**

## **4A. DATA FLOW DIAGRAM**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

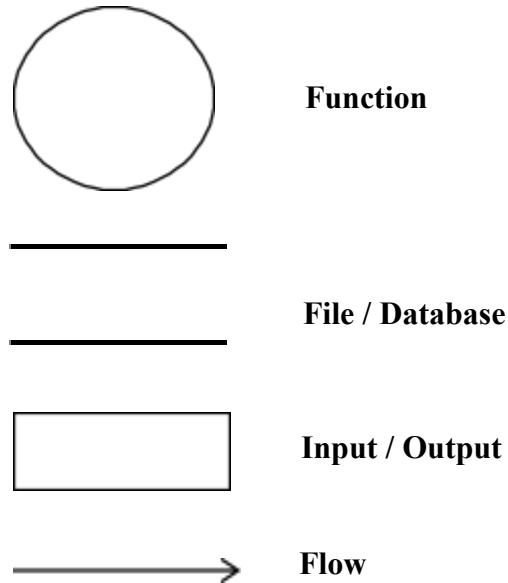
DFD can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of the process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present for the system to do its job and shows the flow of data between the various parts of the system.

Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users can visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's data-flow diagrams can be drawn up and compared with.

How any system is developed can be determined through a data flow diagram model. In the course of developing a set of leveled data flow diagrams, the analyst/designer is forced to address how the system may be decomposed into component sub-systems and to identify the transaction data in the data model. Data flow diagrams can be used in both the Analysis and Design phase of the SDLC. There are different notations to draw data flow diagrams. Defining different visual representations for processes, data stores, data flow, and external entities.

## **DFD Notation:**



## **DFD Example:**



## **Steps to Construct Data Flow Diagram:**

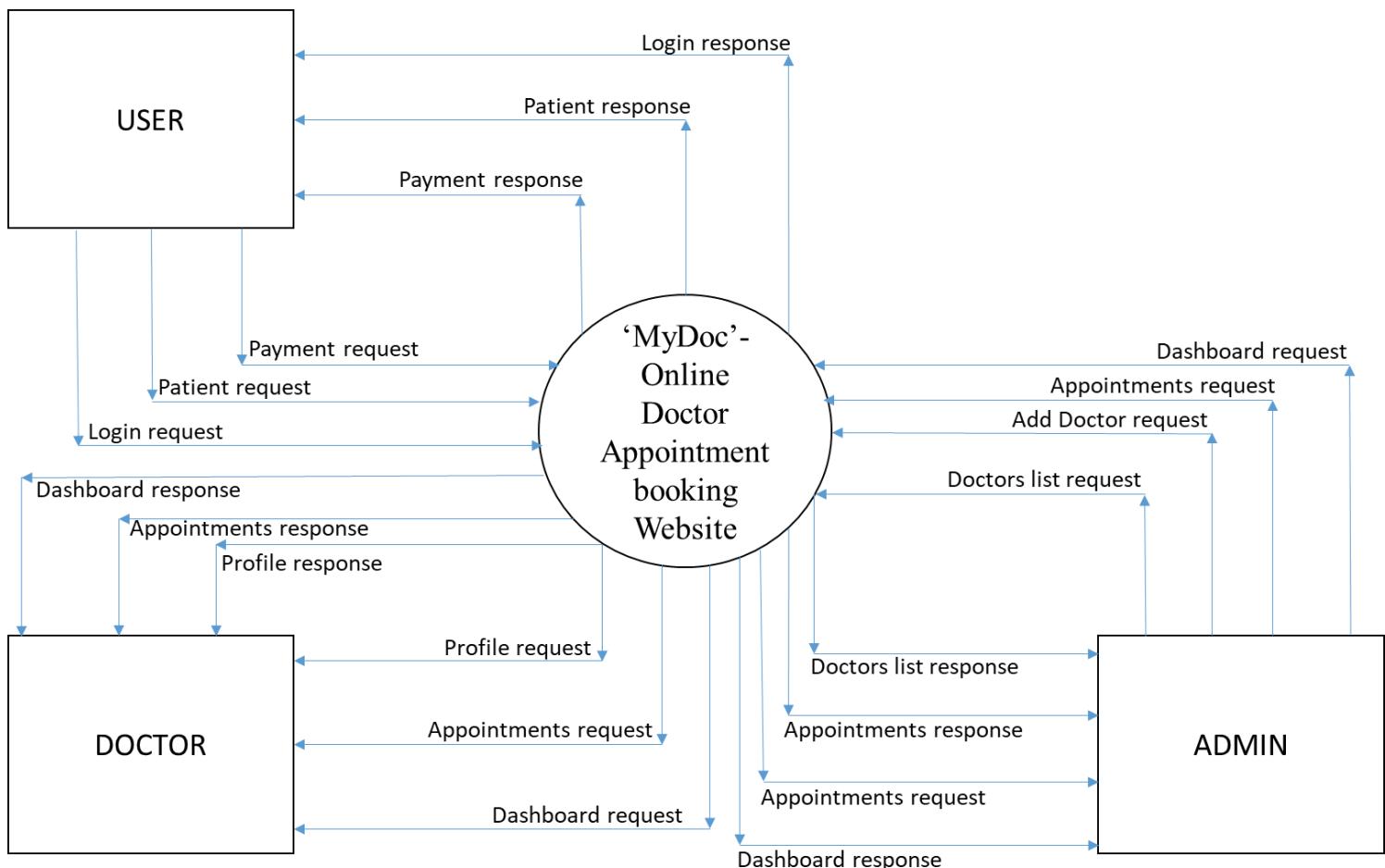
Four Steps are generally used to construct a DFD.

- Processes should be named and referred for easy reference. Each name should be representative of the reference.
- The destination of flow is from top to bottom and from left to right.
- When a process is distributed into lower-level details they are numbered.
- The names of data stores, sources, and destinations are written in capital letters.

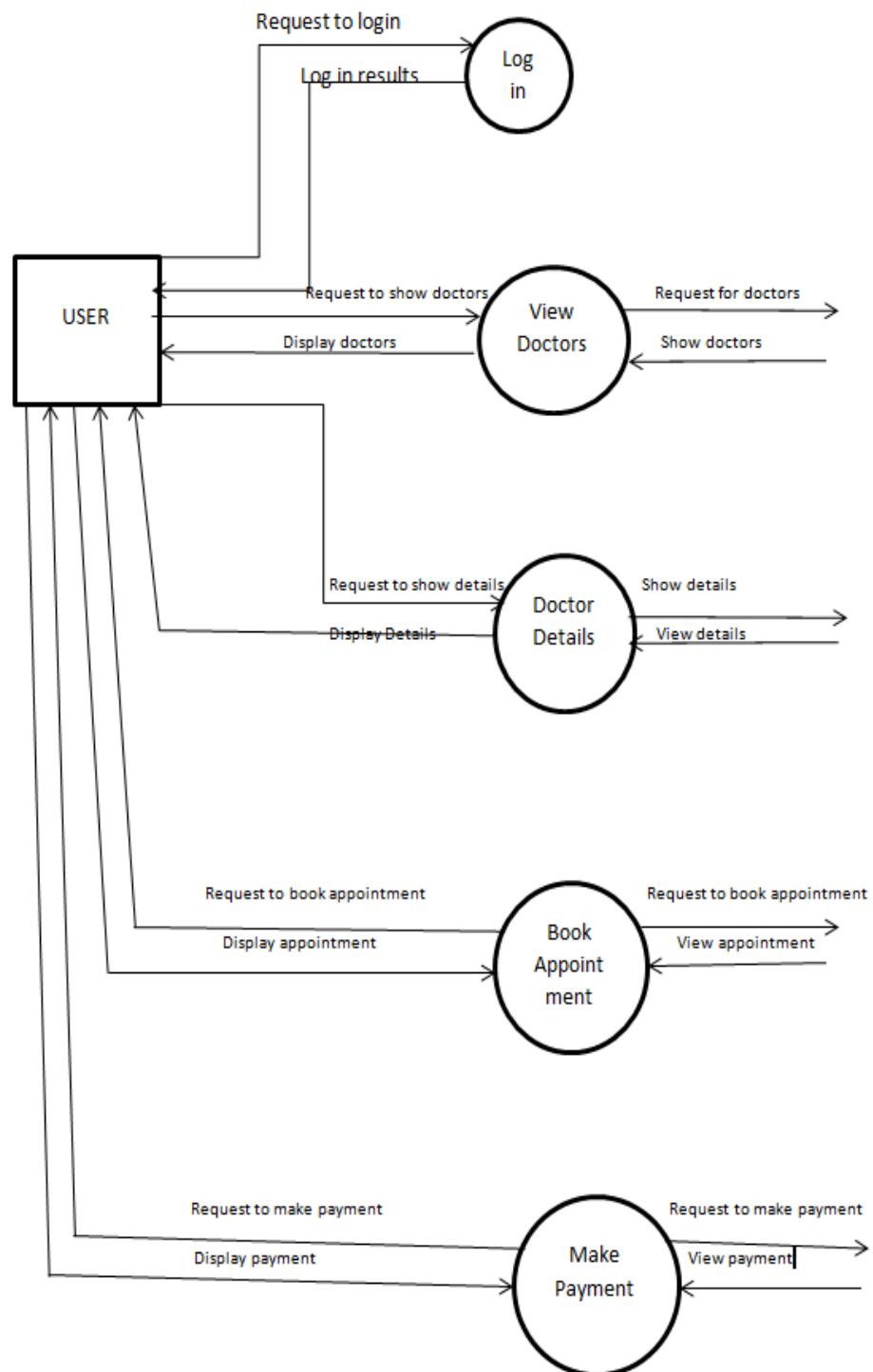
## Rules for constructing a Data Flow Diagram:

- Arrows should not cross each other.
- Squares, Circles, and Files must bear a name.
- Decomposed data flow squares and circles can have the same names.
- Draw all data flow around the outside of the diagram.

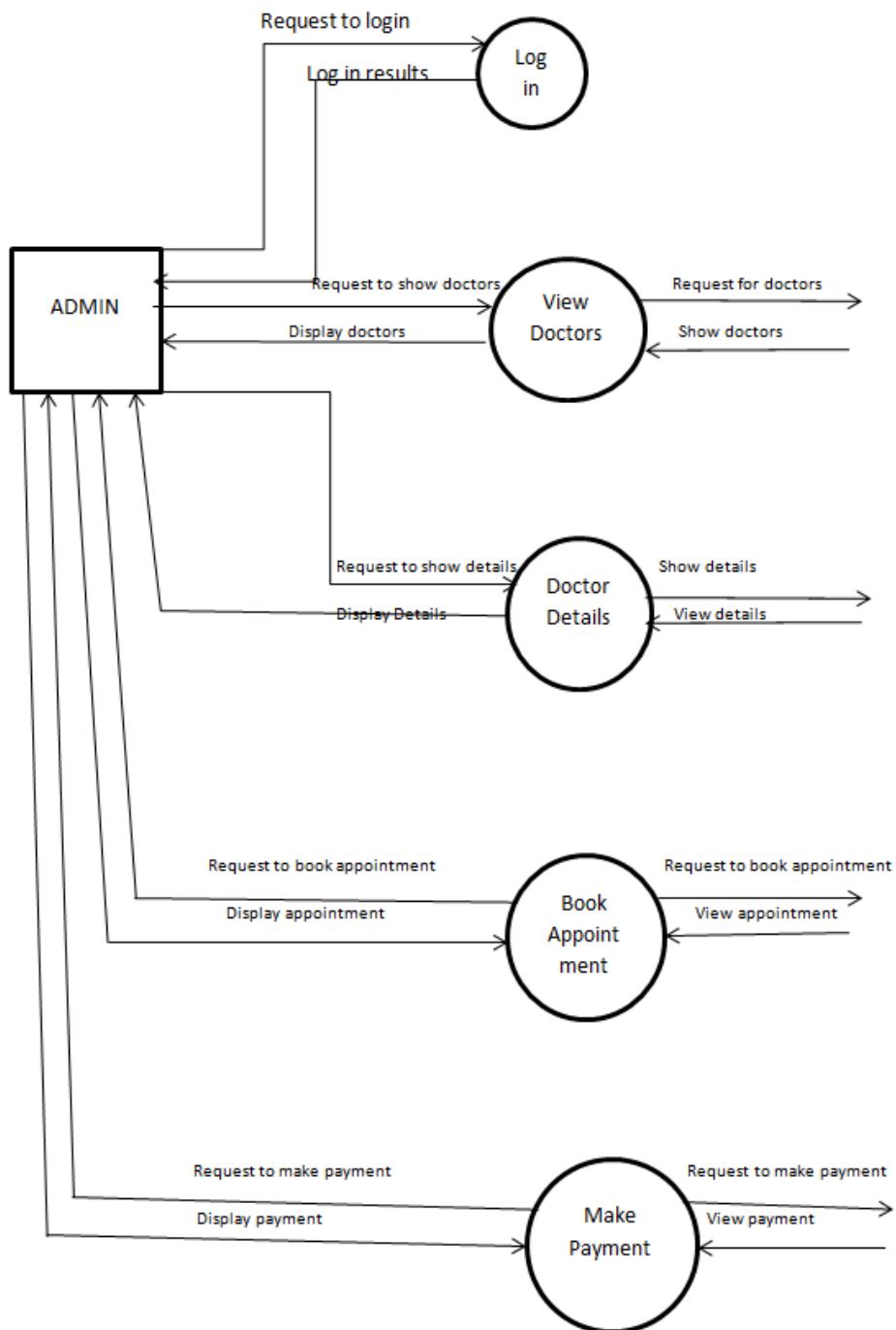
### • LEVEL 0 DFD OR CONTEXT DIAGRAM:



- LEVEL 1 DFD:



- LEVEL 1 DFD:



## **4B. SEQUENCE DIAGRAM**

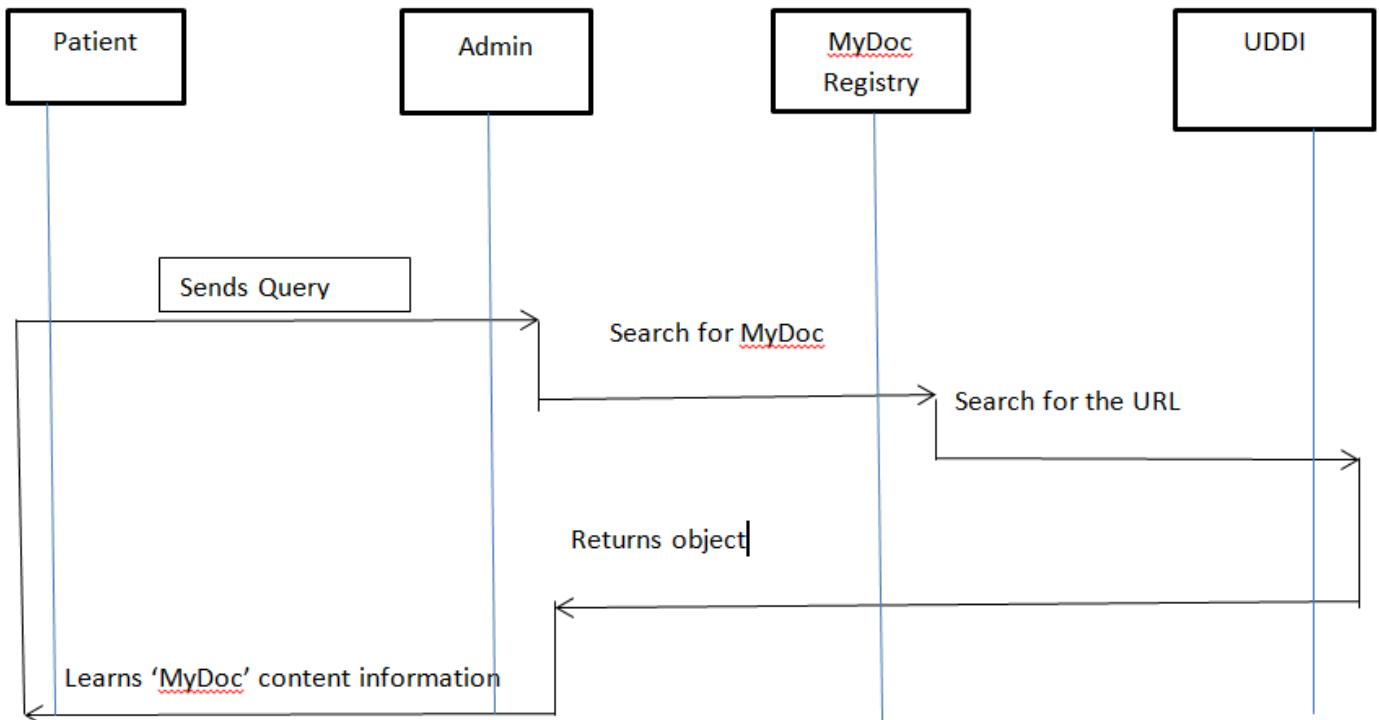
A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

A sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between several life lines .A sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, fragment, interaction, state invariant, continuation, and destruction occurrence.



A Use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

So only static behavior is not sufficient to model a system, rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and a use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams consist of actors, use cases, and their relationships. The diagram is used to model the system/subsystem of an application. A single-use case diagram captures a particular functionality of a system.

So, to model the entire system numbers of use case diagrams are used. The purpose of a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because the other four diagrams (activity, sequence, collaboration, and State chart) are also having the same purpose. So, we will look into some specific purpose that will distinguish it from the other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modeled to present the outside view. So, in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.

## **How to draw a Use Case Diagram?**

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed, the functionalities are captured in use cases.

So, we can say that use cases are nothing but the system functionalities written in an organized manner. Now the second thing which is relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

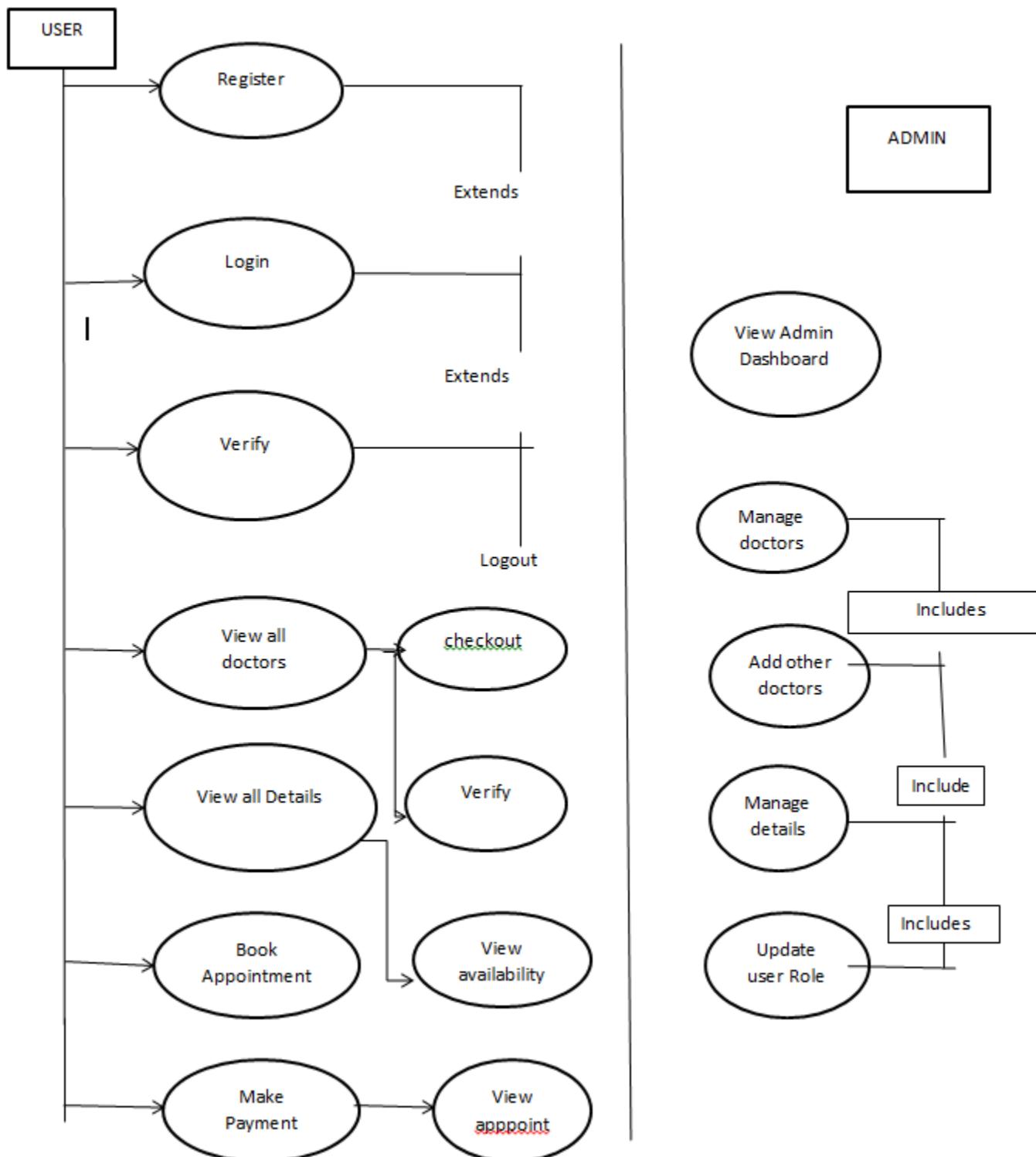
The actors can be human users, some internal applications or may be some external applications. So, in a brief when we are planning to draw a use case diagram, we should have the following items identified.

- Functionalities to be represented as a use case
- Actors
- Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So, after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

- The name of a use case is very important. So, the name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
- Use note whenever required to clarify some important point

- USE CASE DIAGRAM:

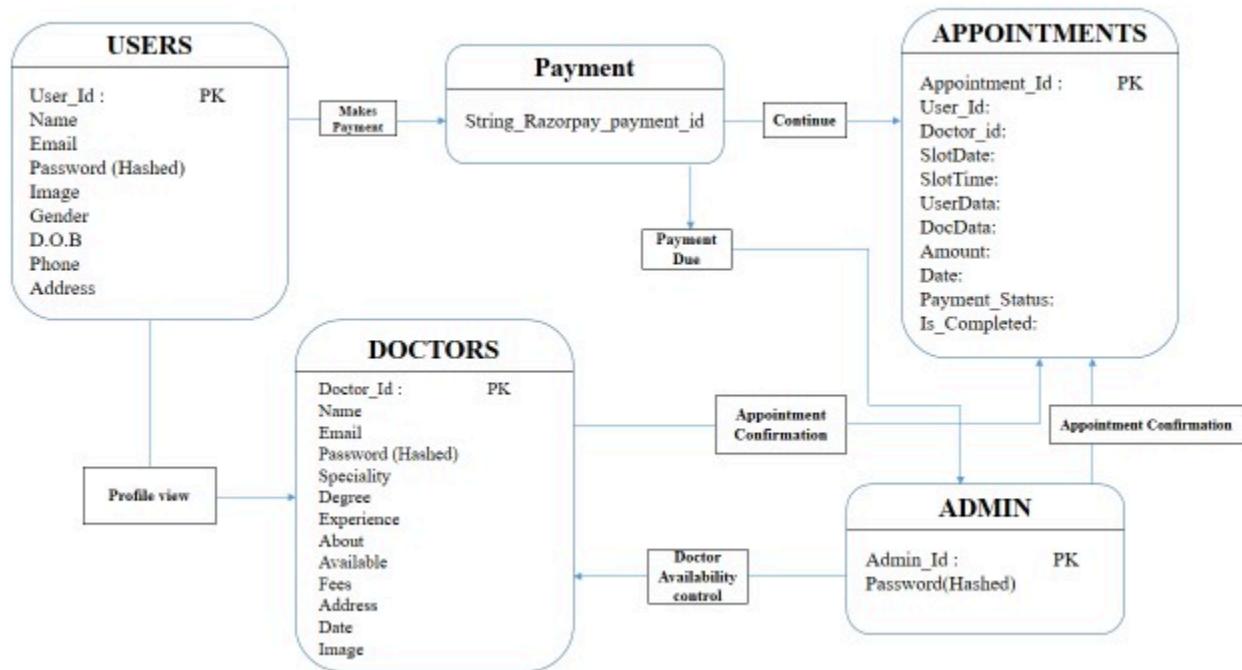


## 4D. SCHEMA DIAGRAM

The schema is an abstract structure or outline representing the logical view of the database as a whole. Defining categories of data and relationships between those categories, database schema design makes data much easier to retrieve, consume, manipulate, and interpret.

DB schema design organizes data into separate entities, determines how to create relationships between organized entities, and influences the applications of constraints on data. Designers create database schema to give other database users, such as programmers and analysts, a logical understanding of data.

- **SCHEMA DESIGN:**



## 5. UI SNAPSHOT

### ❖ FRONTEND :-

#### 1) Login & Register Page:

The image contains two screenshots of a mobile application interface for 'MDoc'. Both screenshots show a header with the 'MDoc' logo, navigation links for 'Home', 'All Doctors', 'About', and 'Contact', a search bar labeled 'Search doctors...', and a 'Create/Login' button.

**Screenshot 1: Login Page**

The first screenshot shows the 'Login' page. It features a title 'Login' and a sub-instruction 'Please log in to book appointment'. There are two input fields: 'Email' and 'Password', both represented by white input boxes with black outlines. Below these fields is a large purple rectangular button with the word 'Login' in white. At the bottom of the form, there is a small blue link that says 'Create a new account? Click here'.

**Screenshot 2: Create Account Page**

The second screenshot shows the 'Create Account' page. It has a title 'Create Account' and a sub-instruction 'Please sign up to book appointment'. It includes three input fields: 'Full Name' (with a placeholder 'John Doe'), 'Email' (with a placeholder 'john.doe@example.com'), and 'Password' (with a placeholder 'password123'). Below these is a purple 'Sign Up' button. Similar to the login page, it includes a 'Click here' link at the bottom for existing users.

### ✓ CODE:

```
import React, { useContext, useEffect, useState } from 'react'
import { AppContext } from '../context/AppContext'
import axios from 'axios'
import { toast } from 'react-toastify'
import { useNavigate } from 'react-router-dom'
```

```
const Login = () => {

  const {backendUrl, token, setToken} = useContext(AppContext)
  const navigate = useNavigate()
  const [state, setState] = useState('Sign Up')
  const [fullName, setFullName] = useState('')
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')

  const onSubmitHandler = async (event) => {
    event.preventDefault()

    try {
      if (state === 'Sign Up') {
        const {data} = await axios.post(backendUrl + '/api/user/register', {name
:fullName,password,email})
        console.log(data)
        if (data.success) {
          localStorage.setItem('token',data.token)
          setToken(data.token)
        } else {
          toast.error(data.message)
        }
      } else {
        const {data} = await axios.post(backendUrl + '/api/user/login', {password,email})
        if (data.success) {
          localStorage.setItem('token',data.token)
          setToken(data.token)
        } else {
          toast.error(data.message)
        }
      }
    } catch (error) {
      toast.error(error.message)
    }
    console.log({ fullName, email, password })
  }

  useEffect(()=>{
    if (token) {
      navigate('/')
    }
  },[token])
}
```

```
return (
  <div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-white via-purple-50 to-purple-100">
    <form
      onSubmit={onSubmitHandler}
      className="flex flex-col gap-4 p-8 rounded-2xl shadow-xl bg-white min-w-[340px] sm:min-w-96 border border-purple-100 text-zinc-600 text-sm">
      >
        <p className="text-2xl font-bold text-purple-600">
          {state === 'Sign Up' ? 'Create Account' : 'Login'}
        </p>
        <p className="text-sm text-zinc-500">
          Please {state === 'Sign Up' ? 'sign up' : 'log in'} to book appointment
        </p>

        {state === 'Sign Up' && (
          <div className="w-full">
            <p>Full Name</p>
            <input
              className="border border-zinc-300 rounded w-full p-2 mt-1 focus:outline-none focus:ring-1 focus:ring-purple-400"
              type="text"
              onChange={(e) => setFullName(e.target.value)}
              value = {fullName}
              required
            />
          </div>
        )}
      <div className="w-full">
        <p>Email</p>
        <input
          className="border border-zinc-300 rounded w-full p-2 mt-1 focus:outline-none focus:ring-1 focus:ring-purple-400"
          type="email"
          onChange={(e) => setEmail(e.target.value)}
          value = {email}
          required
        />
      </div>

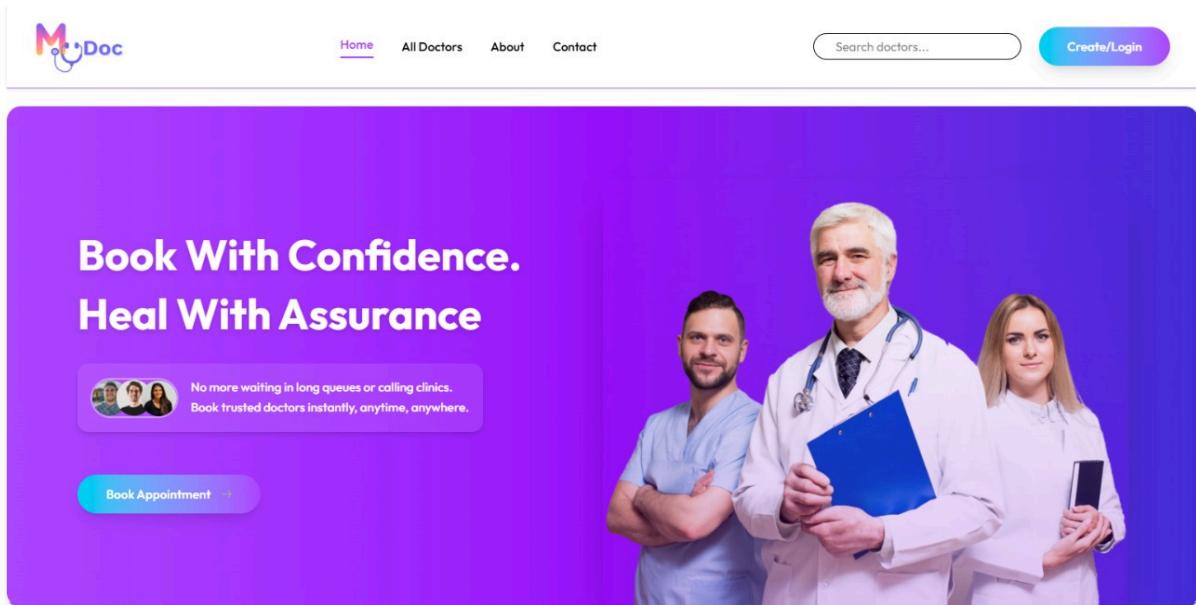
      <div className="w-full">
        <p>Password</p>
        <input
          className="border border-zinc-300 rounded w-full p-2 mt-1 focus:outline-none focus:ring-1 focus:ring-purple-400"
          type="password"
        />
      </div>
    </form>
  </div>
)
```

```
        onChange={(e) => setPassword(e.target.value)}
        value = {password}
        required
      />
    </div>

    <button
      type="submit"
      className="bg-purple-500 hover:bg-purple-600 text-white w-full py-2 rounded-md
text-base transition duration-300"
    >
      {state === 'Sign Up' ? 'Sign Up' : 'Login'}
    </button>

    <div className="text-sm text-center mt-2 text-zinc-500">
      {state === 'Sign Up' ? (
        <p>
          Already have an account?{' '}
          <span
            className="text-purple-500 font-medium cursor-pointer hover:underline"
            onClick={() => setState('Login')}
          >
            Login here
          </span>
        </p>
      ) : (
        <p>
          Create a new account?{' '}
          <span
            className="text-purple-500 font-medium cursor-pointer hover:underline"
            onClick={() => setState('Sign Up')}
          >
            Click here
          </span>
        </p>
      )}
    </div>
  </form>
</div>
)
}
```

## 2) MYDOC MAIN PAGE (user):



The screenshot shows the MyDoc main page. At the top, there is a navigation bar with the logo "MyDoc" on the left, followed by "Home", "All Doctors", "About", and "Contact". To the right of the navigation bar is a search bar containing the placeholder "Search doctors..." and a "Create/Login" button. The main content area features a purple background with the text "Book With Confidence. Heal With Assurance" in white. Below this text is a small circular icon showing three people and the text "No more waiting in long queues or calling clinics. Book trusted doctors instantly, anytime, anywhere." A blue "Book Appointment" button is located below the text. To the right of the text, there is a photograph of three medical professionals: a male nurse in blue scrubs, an older male doctor in a white coat holding a blue folder, and a female doctor in a white coat holding a black folder.

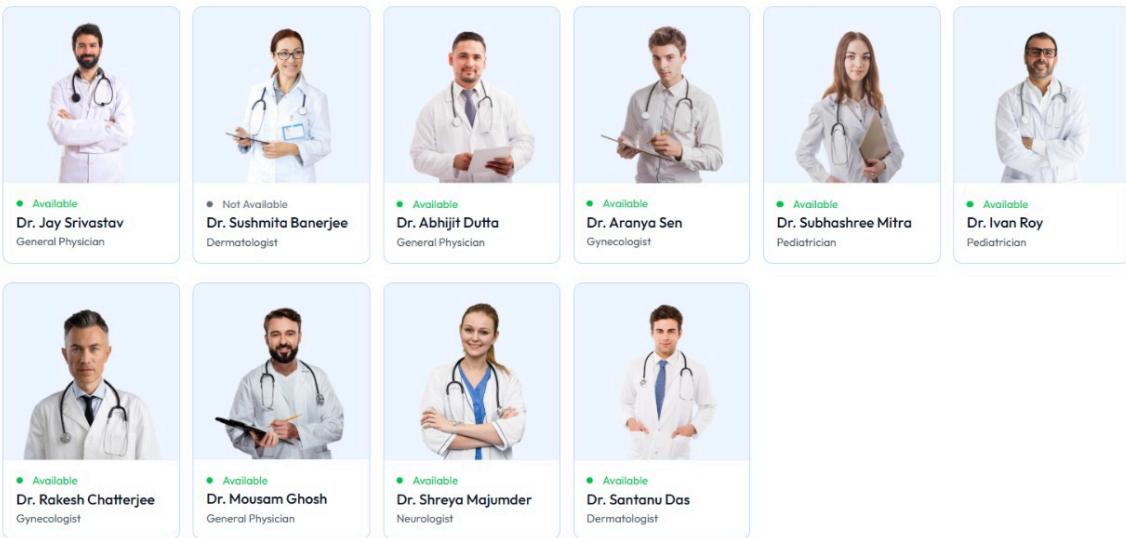
### Find by Speciality

Find the right doctor for your specific needs. Whether it's heart care, skin issues, child health, or routine check-ups, book top specialists like cardiologists, dermatologists, pediatricians, and more — all in one place. Get expert care, faster.



### Top Doctors to Book

Simply Browse through our extensive list of trusted doctors



The screenshot displays a grid of twelve doctor profiles, each in a separate card. The cards are arranged in two rows of six. Each card contains a doctor's photo, availability status (green dot for available, grey dot for not available), name, and specialty.

Row	Column	Doctor Name	Specialty	Status
1	1	Dr. Jay Srivastav	General Physician	Available
1	2	Dr. Sushmita Banerjee	Dermatologist	Not Available
1	3	Dr. Abhijit Dutta	General Physician	Available
1	4	Dr. Aranya Sen	Gynecologist	Available
1	5	Dr. Subhashree Mitra	Pediatrician	Available
1	6	Dr. Ivan Roy	Pediatrician	Available
2	1	Dr. Rakesh Chatterjee	Gynecologist	Available
2	2	Dr. Mousam Ghosh	General Physician	Available
2	3	Dr. Shreya Majumder	Neurologist	Available
2	4	Dr. Santanu Das	Dermatologist	Available

YOUR HEALTH, OUR PRIORITY

# Book Appointment With 100+ Trusted Doctors

Get expert medical care from top specialists across all fields – anytime, anywhere.

[Create Account](#)



MyDoc is your trusted partner in healthcare, simplifying the way you connect with medical professionals. Whether you're booking consultations, managing appointments, or accessing expert care, our platform ensures a seamless and secure experience. Prioritizing convenience, privacy, and reliability — we bring quality healthcare to your fingertips.

#### COMPANY

[Home](#)  
[About Us](#)  
[Contact Us](#)  
[Privacy Policy](#)

#### GET IN TOUCH

+91 9876543210  
[support@mydoc.com](mailto:support@mydoc.com)

Copyright 2025@MyDoc-All Right Reserved.

```
<!doctype html>
<html lang="en" class="scroll-smooth">
  <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>MyDoc</title>
  </head>
  <body>
    <div id="root"></div>
    <script src="https://checkout.razorpay.com/v1/checkout.js"></script>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

**Main.jsx code :-**

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'
import {BrowserRouter} from 'react-router-dom'
import ApplicationContextProvider from './context/AppContext.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <BrowserRouter>
      <ApplicationContextProvider>
        <App />
      </ApplicationContextProvider>
    </BrowserRouter>
  </StrictMode>
)
```

**Home.jsx code :-**

```
import Banner from "../components/Banner"
import Footer from "../components/Footer"
import Header from "../components/Header"
import SpecialityMenu from "../components/SpecialityMenu"
import TopDoctors from "../components/TopDoctors"

const Home = () => {
  return (
    <div>
      <Header/>
      <SpecialityMenu/>
      <TopDoctors/>
      <Banner />
    </div>
  )
}
```

```
}
```

```
export default Home
```

- components – Banner.jsx :-

```
import React from 'react'
import { assets } from '../assets/assets'
import { useNavigate } from 'react-router-dom'

const Banner = () => {
  const navigate = useNavigate()

  return (
    <div className='flex bg-gradient-to-r from-purple-500 to-indigo-700 rounded-lg px-6 sm:px-10 md:px-14 lg:px-8 my-20 md:mx shadow-lg'>
      {/*----- Left Side -----*/}
      <div className='flex-1 py-10 sm:py-16 lg:py-24 lg:pl-5 flex flex-col justify-center'>
        <p className='text-white text-sm uppercase tracking-wide mb-2'>Your Health, Our Priority</p>

        <div className='text-xl sm:text-2xl md:text-3xl lg:text-5xl font-bold text-white leading-snug'>
          <p>Book Appointment</p>
          <p>With 100+ Trusted Doctors</p>
        </div>

        <p className='text-white/90 mt-4 max-w-md text-sm sm:text-base'>
          Get expert medical care from top specialists across all fields – anytime, anywhere.
        </p>
        <button
          onClick={() => { navigate('/login'); scrollTo(0, 0) }}
          className='bg-gradient-to-r from-cyan-400 to-purple-500 text-white font-medium text-1 px-0 py-3 rounded-full mt-8 hover:scale-105 transition-all shadow-md hover:shadow-lg'>
          Create Account
        </button>
      </div>
    </div>
  )
}
```

```

        </div>

        {/*----- Right Side -----*/}
        <div className='hidden md:flex md:w-1/2 lg:w-[370px] relative justify-center items-end'>
            <img
                className='w-full max-w-md'
                src={assets.appointment_img}
                alt="Doctor Appointment Illustration"
            />
        </div>
    </div>
)
}

export default Banner

```

- **components – Footer.jsx :-**

```

import React from 'react'
import { assets } from '../assets/assets'

const Footer = () => {
    return (
        <div className='md:mx-10'>
            <div className='flex flex-row grid-cols gap-30 my-10 mt-40 text-sm'>

                {/*-----LeftSide-----*/}
                <div>
                    <img className='w-[130px] h-[75px] mb-4' src={assets.myDoc_logo} alt="" />
                    <p className='w-full md:w-2/3 text-gray-600 leading-6'>MyDoc is your trusted partner in healthcare, simplifying the way you connect with medical professionals. Whether you're booking consultations, managing appointments, or accessing expert care, our platform ensures a seamless and secure experience. Prioritizing convenience, privacy, and reliability – we bring quality healthcare to your fingertips.</p>
                </div>

                {/*-----Center-----*/}
                <div>

```

```

<p className='text-xl font-medium mb-5'>COMPANY</p>
<ul className='flex flex-col gap-2 text-gray-600'>
  <li>Home</li>
  <li>About Us</li>
  <li>Contact Us</li>
  <li>Privacy Policy</li>
</ul>
</div>
{/*-----RightSide-----*/}
<div>
  <p className='text-xl font-medium mb-5'>GET IN TOUCH</p>
  <ul className='flex flex-col gap-2 text-gray-600'>
    <li>+91 9876543210</li>
    <li>support@mydoc.com</li>
  </ul>
</div>
</div>
{/*-----CopyRight----- */}
<div>
  <hr />
  <p className='py-5 text-sm text-center'>Copyright 2025@MyDoc-All Right Reserved.</p>
</div>
</div> )
}
export default Footer

```

- **components – Header.jsx :-**

```

import React from "react";
import { assets } from "../assets/assets"

const Header = () => {
  return (
    <div className="flex flex-col md:flex-row flex-wrap bg-gradient-to-r from-purple-500 via-purple-600 to-indigo-700 relative rounded-2xl px-6 md:px-10 lg:px-20 overflow-hidden shadow-lg">

```

```

/* ----- Left Side ----- */



Book With Confidence. <br /> Heal With Assurance


</p>

<div className="flex flex-col md:flex-row items-center gap-4 text-white text-sm font-medium bg-white/10 p-4 rounded-xl backdrop-blur-md shadow-md">
![Profiles]({assets.group_profiles})
</p>
<p className="leading-relaxed">
    No more waiting in long queues or calling clinics.{ " " }
    <br className="hidden sm:block" />
    Book trusted doctors instantly, anytime, anywhere.
</p>
</div>

<div id="speciality" className="scroll-mt-20"></div>

<a
    href="#speciality"
    className="flex items-center gap-3 bg-gradient-to-r from-cyan-400 to-purple-500 px-8 py-3 rounded-full text-white text-sm font-semibold shadow-lg hover:shadow-xl hover:scale-105 transition-all duration-300 ease-in-out"
    >
    Book Appointment
    <img
        className="w-3 animate-pulse invert"
        src={assets.arrow_icon}
        alt="arrow"
    />


```

```

        </a>
    </div>

    {/* ----- Right Side ----- */}
    <div className="md:w-1/2 relative flex justify-center items-center z-10">
        <div className="relative w-full md:absolute bottom-0">
            <img
                className="w-full h-auto rounded-lg shadow-2xl transform hover:scale-105
transition-transform duration-500"
                src={assets.header_img}
                alt="Doctor illustration"
            />
            <div className="absolute inset-0 bg-gradient-to-t from-purple-700/20
to-transparent rounded-lg"></div>
        </div>
    </div>
</div>
)
}

export default Header

```

- **components – Navbar.jsx :-**

```

import React, { useContext, useState } from 'react'
import { assets } from '../assets/assets'
import { NavLink, useNavigate } from 'react-router-dom'
import { ApplicationContext } from '../context/AppContext'

const Navbar = () => {

    const { searchTerm, setSearchTerm } = useContext(ApplicationContext)
    const navigate = useNavigate()

    const handleSearch = (e) =>{
        if (e.key === "Enter"){
            navigate("/doctors")
        }
    }
}

```

```

}

const {token, setToken, userData} = useContext(AppContext)
const logout = () => {
  setToken(false)
  localStorage.removeItem('token')
}
const [isAdmin] = useState(true); // Change to false for non-admin
// const [username] = useState("Tiasa!!"); // Replace with actual username from
context/auth
const username = userData?.name || "Guest"; // Replaced with actual username

return (
  <div className='flex items-center justify-between text-sm py-4 mb-5 border-b
border-b-purple-400 bg-white sticky top-0 z-90 shadow-sm px-4 md:px-8'>
    <img onClick={() => navigate('/')} className='w-[100px] h-[60px] cursor-pointer'
src={assets.myDoc_logo} alt="logo"/>
    <ul className='hidden md:flex items-center gap-8 font-medium'>
      {[

        { name: 'Home', path: '/' },
        { name: 'All Doctors', path: '/doctors' },
        { name: 'About', path: '/about' },
        { name: 'Contact', path: '/contact' },
      ].map((link, index) => (
        <NavLink key={index} to={link.path} className={({ isActive }) => isActive ?
"text-purple-500 border-b-2 border-purple-500 pb-1" : "hover:text-purple-500"}>
          {link.name}
        </NavLink>
      )))
    </ul>

    <div className='flex items-center gap-5'>
      <input
        type="text"
        placeholder="Search doctors...">
        <span>
          {userData?.name}
        </span>
      </input>
    </div>
  </div>
)

```

```

        onChange={(e)=> setSearchTerm(e.target.value)}
        onKeyDown={handleSearch}/>

    {token && userData ? (
      <div className='flex items-center gap-2 cursor-pointer group relative'>
        {/* Greeting */}
        <p className='hidden md:block text-gray-600'>Hi,{username}</p>
        <img className='w-9 h-9.5 object-cover rounded-full ' src={userData.image} alt="profile" />
        <img className='w-2.5' src={assets.dropdown_icon} alt="dropdown" />
        <div className='absolute top-0 right-0 pt-14 text-base font-medium text-gray-600 z-20 hidden group-hover:block'>
          <div className='min-w-45 bg-stone-100 rounded flex flex-col gap-3 p-3 mt-4'>
            <p onClick={() => navigate('my-profile')} className='hover:text-black cursor-pointer'>My Profile</p>
            <p onClick={() => navigate('my-appointments')} className='hover:text-black cursor-pointer'>My Appointments</p>
            <p onClick={logout} className='hover:text-black cursor-pointer'>Logout</p>
          </div>
        </div>
      </div>
    ) : (
      <button
        onClick={() => navigate('/login')}
        className='bg-gradient-to-r from-cyan-400 to-purple-500 px-8 py-3 rounded-full text-white text-sm font-semibold shadow-lg hover:shadow-xl hover:scale-105 transition-all duration-300 ease-in-out'
      >
        Create/Login
      </button>
    )
  )
}

export default Navbar

```

- **components – TopDoctors.jsx :-**

```

import React, { useContext } from 'react'
import { useNavigate } from 'react-router-dom'
import Appointment from '../pages/Appointment'
import { ApplicationContext } from '../context/AppContext'

const TopDoctors = () => {

  const navigate = useNavigate()
  const {doctors} = useContext(ApplicationContext);

  return (
    <div className='flex flex-col items-center gap-4 my-16 text-gray-900 md:mx-10'>
      <h1 className='text-3xl font-medium'>Top Doctors to Book </h1>
      <p className='sm:w-1/3 text-center text-sm'>Simply Browse through our extensive
      list of trusted doctors</p>
      <div className='w-full grid grid-cols-[repeat(auto-fill,minmax(200px,1fr))] gap-4
      pt-5 gap-y-6 px-3 sm:px-0'>
        {doctors.slice(0,10).map((item,index)=>
          <div onClick={()=>{navigate(`/appointment/${item._id}`); scrollTo(0,0)}}>
            className='border border-blue-200 rounded-xl overflow-hidden cursor-pointer
            hover:translate-y-[-15px] transition-all duration-900' key={index}>
              <img className='bg-blue-50' src={item.image} alt="" />
              <div className='p-4'>
                <div className={`flex items-center gap-2 text-sm text-center
                ${item.available ? 'text-green-500': 'text-gray-500'}`}>
                  <p className={`w-2 h-2 ${item.available ? 'bg-green-500' :
                  'bg-gray-500' } rounded-full`}></p><p className='px-1'>{item.available ? 'Available':
                  'Not Available'}</p>
                </div>
                <p className='text-gray-900 text-lg font-medium'>{item.name}</p>
                <p className='text-gray-600 text-sm'>{item.speciality}</p>
              </div>
            </div>
          ))}
        </div>
        <button onClick={()=>{ navigate('/doctors'); scrollTo(0,0)}}
          className='bg-gradient-to-r from-cyan-400 to-purple-500 px-8 py-3 mt-10
          rounded-full text-white text-sm font-semibold shadow-lg hover:shadow-xl hover:scale-105
          transition-all duration-300 ease-in-out
          '>More</button>
      </div>
    )
  }
}

export default TopDoctors

```

- **components – Speciality.jsx :-**

```

import React from 'react'
import { specialityData } from '../assets/assets'

```

```

import { Link } from 'react-router-dom'

const SpecialityMenu = () => {
  return (
    <div className='flex flex-col items-center gap-5 py-16 text-gray-800' id='Speciality'>
      <h1 className='text-3xl font-medium'>Find by Speciality</h1>
      <p className='sm:w-2/3 text-center text-medium'>Find the right doctor for your specific needs. Whether it's heart care, skin issues, child health, or routine check-ups, book top specialists like cardiologists, dermatologists, pediatricians, and more – all in one place. Get expert care, faster. </p>
      <div className='flex sm:justify-center gap-8 pt-5 w-full overflow-scroll'>
        {specialityData.map((item,index)=>(
          <Link onClick={()=>scrollTo(0,0)} className='flex flex-col items-center text-xs cursor-pointer flex-shrink-0 hover:translate-y-[-10px] transition-all duration-500' key={index} to={`/doctors/${item.speciality}}>
            <img className='w-16 sm:w-24 mb-2' src={item.image} alt={item.speciality}>
          />
          <p>{item.speciality}</p>
        />
      )))
    </div>
  </div>
)
}

export default SpecialityMenu

```

- **components – RelatedDoctors.jsx :-**

```

import React, { useContext, useEffect, useState } from 'react'
import { AppContext } from '../context/AppContext'
import { useNavigate } from 'react-router-dom'

const RelatedDoctors = ({speciality,docId}) => {

  const {doctors}=useContext(AppContext)
  const navigate = useNavigate()

  const [relDoc,setRelDocs] = useState([])

  useEffect(()=>{
    if(doctors.length > 0 && speciality){
      const doctorsData = doctors.filter((doc)=> doc.speciality === speciality && doc._id !== docId)
      setRelDocs(doctorsData)
    }
  },[doctors,speciality,docId])

  return (

```

```

<div className='flex flex-col items-center gap-4 my-16 text-gray-900 md:mx-10'>
  <h1 className='text-3xl font-medium'>Top Doctors to Book </h1>
  <p className='sm:w-1/3 text-center text-sm'>Simply Browse through our extensive
list of trusted doctors</p>
  <div className='w-full grid grid-cols-[repeat(auto-fill,minmax(200px,1fr))] gap-4
pt-5 gap-y-6 px-3 sm:px-0'>
    {relDoc.slice(0,5).map((item,index)=>(
      <div onClick={()=>{navigate(`/appointment/${item._id}`); scrollTo(0,0)}}>
        className='border border-blue-200 rounded-xl overflow-hidden cursor-pointer
        hover:translate-y-[-15px] transition-all duration-900' key={index}>
          <img className='bg-blue-50' src={item.image} alt="" />
          <div className='p-4'>
            <div className={`flex items-center gap-2 text-sm text-center
${item.available ? 'text-green-500': 'text-gray-500'}`}>
              <p className={`w-2 h-2 ${item.available ? 'bg-green-500' :
'bg-gray-500' } rounded-full`}></p><p className='px-1'>{item.available ? 'Available':
'Not Available'}`</p>
            </div>
          </div>
        </div>
      ))}
    </div>
    <button onClick={()=>{ navigate('/doctors'); scrollTo(0,0)}}>
      className='bg-purple-500 text-white px-12 py-3 rounded-full mt-10 border-purple-950
      hover:translate-y-[-3px] transition-all duration-400'>More</button>
    </div>
  )}
export default RelatedDoctors

```

## 1) ALL DOCTOTS (for user):

✓ CODE:

```
import React, { useContext, useEffect, useState } from 'react'
import { Navigate, useNavigate, useParams } from 'react-router-dom'
import { ApplicationContext } from '../context/ApplicationContext';
import { doctors } from '../assets/assets';

const Doctors = () => {

  const { speciality } = useParams();

  const [filterDoc, setFilterDoc] = useState([])
  const navigate = useNavigate()

  const {doctors, searchTerm} = useContext(ApplicationContext)

  const applyFilter = () => {
    let filtered = doctors;

    if (speciality) {
      filtered = filtered.filter(doc => doc.speciality === speciality);
    }

    if (searchTerm) {
      filtered = filtered.filter(doc =>
        doc.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
        doc.speciality.toLowerCase().includes(searchTerm.toLowerCase())
      );
    }

    setFilterDoc(filtered);
  };

  useEffect(()=>{
    applyFilter()
  },[doctors, speciality])

  return (
    <div>
      <p className='text-gray-600'>Browse through the doctors specialiats</p>
      <div className='flex flex-col sm:flex-row items-start gap-9 mt-5'>
        <div className='flex flex-col gap-4 text-sm text-gray-600'>
          <p onClick={()=> speciality === 'General Physician' ? navigate('/doctors') : navigate('/doctors/General Physician')} className={`w-[94] sm:w-auto pl-3 py-1.5 pr-16 border border-gray-300 rounded transition-all hover:translate-y-[2px] cursor-pointer ${speciality === "General Physician" ? "bg-purple-300 text-black" : "" }`}>General Physician</p>
          <p onClick={()=> speciality === 'Gynecologist' ? navigate('/doctors') : navigate('/doctors/Gynecologist')} className={`w-[94] sm:w-auto pl-3 py-1.5 pr-16 border border-gray-300 rounded transition-all hover:translate-y-[2px] cursor-pointer ${speciality === "Gynecologist" ? "bg-purple-300 text-black" : "" }`}>Gynecologist</p>
          <p onClick={()=> speciality === 'Dermatologist' ? navigate('/doctors') : navigate('/doctors/Dermatologist')} className={`w-[94] sm:w-auto pl-3 py-1.5 pr-16 border border-gray-300 rounded transition-all hover:translate-y-[2px] cursor-pointer ${speciality === "Dermatologist" ? "bg-purple-300 text-black" : "" }`}>Dermatologist</p>
        </div>
      </div>
    </div>
  )
}
```

```

border-gray-300 rounded transition-all hover:translate-y-[2px] cursor-pointer
${speciality === "Dermatologist" ? "bg-purple-300 text-black" : "" }>Dermatologist</p>
    <p onClick={()=> speciality === 'Pediatrician' ? navigate('/doctors') :
navigate('/doctors/Pediatrician')} className={`w-[94] sm:w-auto pl-3 py-1.5 pr-16 border
border-gray-300 rounded transition-all hover:translate-y-[2px] cursor-pointer
${speciality === "Pediatrician" ? "bg-purple-300 text-black" : "" }>Pediatrician</p>
    <p onClick={()=> speciality === 'Neurologist' ? navigate('/doctors') :
navigate('/doctors/Neurologist')} className={`w-[94] sm:w-auto pl-3 py-1.5 pr-16 border
border-gray-300 rounded transition-all hover:translate-y-[2px] cursor-pointer
${speciality === "Neurologist" ? "bg-purple-300 text-black" : "" }>Neurologist</p>
    <p onClick={()=> speciality === 'Gastroenterologist' ? navigate('/doctors') :
navigate('/doctors/Gastroenterologist')} className={`w-[94] sm:w-auto pl-3 py-1.5 pr-16 border
border-gray-300 rounded transition-all hover:translate-y-[2px] cursor-pointer
${speciality === "Gastroenterologist" ? "bg-purple-300 text-black" : ""
`}>Gastroenterologist</p>

```

```

        </div>
        <div className='grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4
gap-5'>
            {
                filterDoc.map((item,index)=>(
                    <div onClick={()=> navigate(`/appointment/${item._id}`)} className='border
border-blue-200 rounded-xl overflow-hidden cursor-pointer hover:translate-y-[-15px]
transition-all duration-900' key={index}>
                        <img className='bg-blue-50' src={item.image} alt="" />
                        <div className='p-4'>
                            <div className={`flex items-center gap-2 text-sm text-center
${item.available ? 'text-green-500': 'text-gray-500'}>
                                <p className={`w-2 h-2 ${item.available ? 'bg-green-500' :
'bg-gray-500' } rounded-full`}></p><p className='px-1'>{item.available ? 'Available':
'Not Available'}</p>
                            </div>
                            <p className='text-gray-900 text-lg font-medium'>{item.name}</p>
                            <p className='text-gray-600 text-sm'>{item.speciality}</p>
                        </div>
                    </div>
                )))
            }
        </div>
    </div>
</div>
)
}

export default Doctors

```



## ABOUT US



At MY DOC, we believe healthcare should be simple, transparent, and accessible to everyone. Our platform was built with the vision of removing the stress and uncertainty from booking a doctor's appointment. Instead of juggling phone calls, long queues, and unclear schedules, MY DOC gives you a seamless way to search for trusted doctors, view their qualifications, check their availability, and book appointments online—all from the comfort of your home. Whether you're looking for a family physician, a specialist, or a diagnostic service, our goal is to connect you to the right healthcare professional at the right time.

We understand that healthcare is not just about treating illness—it's about building trust, maintaining long-term relationships, and ensuring timely care. That's why MY DOC partners with verified doctors and clinics to provide accurate information, real-time scheduling, and a safe environment for patients. Every doctor on our platform has a detailed profile so you can make informed choices based on expertise, experience, and patient feedback. We also ensure that our booking system is fast, secure, and user-friendly, so your healthcare journey begins without unnecessary delays.

### OUR MISSION

Our mission goes beyond appointment booking. We aim to empower patients with the knowledge and tools they need to take control of their health. From preventive check-ups to specialized treatments, MY DOC is here to guide you every step of the way. We are constantly innovating our platform to bring you new features, such as teleconsultations, prescription management, and health reminders, making quality healthcare more accessible than ever before. At MY DOC, your well-being is our priority, and we are committed to bridging the gap between patients and healthcare providers with convenience, care, and trust at the core of everything we do.

## Why Choose Us

We're here to make healthcare simple, secure, and accessible.

### Easy & Quick Booking

Find your doctor, check availability, and book in just a few clicks.

### Verified Doctors

We partner only with trusted, qualified healthcare professionals.

### Real-Time Availability

Choose appointment slots that work best for your schedule.

### Secure & Reliable

Your personal and health data is always protected with us.

### Patient Reviews

Read real feedback from other patients before you book.

### Beyond Booking

Get reminders, teleconsultations, and health management tools.



MyDoc is your trusted partner in healthcare, simplifying the way you connect with medical professionals. Whether you're booking consultations, managing appointments, or accessing expert care, our platform ensures a seamless and secure experience. Prioritizing convenience, privacy, and reliability — we bring quality healthcare to your fingertips.

### COMPANY

[Home](#)  
[About Us](#)  
[Contact Us](#)  
[Privacy Policy](#)

### GET IN TOUCH

+91 9876543210  
[support@mydoc.com](mailto:support@mydoc.com)

## CODE :

```
import React from 'react'
import { assets } from '../assets/assets'
const About = () => {
  return (
    <div>
      <div className='text-center text-2xl pt-10
text-blue-900'>
        <p> ABOUT <span className='text-2xl font-bold
text-blue-900 text-center'>US</span></p>
      </div>

      <div className='my-10 flex flex-col md:flex-row'
```

```
gap-12'>
    <img className='w-full md:max-w-[500px]'  
src={assets.about_image} alt="" />
    <div className='flex flex-col justify-center  
gap-6 md:w-2/4 text-sm text-gray-700'>
        <p>At MY DOC, we believe healthcare should be  
simple, transparent, and accessible to everyone. Our  
platform was built with the vision of removing the  
stress and uncertainty from booking a doctor's  
appointment. Instead of juggling phone calls, long  
queues, and unclear schedules, MY DOC gives you a  
seamless way to search for trusted doctors, view their  
qualifications, check their availability, and book  
appointments online—all from the comfort of your home.  
Whether you're looking for a family physician, a  
specialist, or a diagnostic service, our goal is to  
connect you to the right healthcare professional at the  
right time.</p>
        <p>We understand that healthcare is not just  
about treating illness—it's about building trust,  
maintaining long-term relationships, and ensuring  
timely care. That's why MY DOC partners with verified  
doctors and clinics to provide accurate information,  
real-time scheduling, and a safe environment for  
patients. Every doctor on our platform has a detailed  
profile so you can make informed choices based on  
expertise, experience, and patient feedback. We also  
ensure that our booking system is fast, secure, and  
user-friendly, so your healthcare journey begins  
without unnecessary delays.</p>
        <b className='text-gray-800'>OUR MISSION</b>
        <p>Our mission goes beyond appointment  
booking. We aim to empower patients with the knowledge  
and tools they need to take control of their health.  
From preventive check-ups to specialized treatments, MY  
DOC is here to guide you every step of the way. We are  
constantly innovating our platform to bring you new  
features, such as teleconsultations, prescription  
management, and health reminders, making quality  
healthcare more accessible than ever before. At MY DOC,  
your well-being is our priority, and we are committed  
to bridging the gap between patients and healthcare  
providers with convenience, care, and trust at the core  
of everything we do.
    </p>
    </div>
</div>

<div className="bg-blue-50 py-10 px-4 rounded-xl  
mt-10">
    <h2 className="text-2xl font-bold text-blue-900  
text-center">Why Choose Us</h2>
```

```
<p className="text-gray-600 text-center max-w-2xl mx-auto mt-2">
    We're here to make healthcare simple, secure, and accessible.
</p>

<div className="grid gap-6 sm:grid-cols-2 lg:grid-cols-3 mt-8">
    <div className="bg-white p-5 rounded-xl shadow hover:shadow-lg transition">
        <h3 className="font-semibold text-lg text-blue-900">Easy & Quick Booking</h3>
        <p className="text-gray-600 mt-1">
            Find your doctor, check availability, and book in just a few clicks.
        </p>
    </div>

    <div className="bg-white p-5 rounded-xl shadow hover:shadow-lg transition">
        <h3 className="font-semibold text-lg text-blue-900">Verified Doctors</h3>
        <p className="text-gray-600 mt-1">
            We partner only with trusted, qualified healthcare professionals.
        </p>
    </div>

    <div className="bg-white p-5 rounded-xl shadow hover:shadow-lg transition">
        <h3 className="font-semibold text-lg text-blue-900">Real-Time Availability</h3>
        <p className="text-gray-600 mt-1">
            Choose appointment slots that work best for your schedule.
        </p>
    </div>

    <div className="bg-white p-5 rounded-xl shadow hover:shadow-lg transition">
        <h3 className="font-semibold text-lg text-blue-900">Secure & Reliable</h3>
        <p className="text-gray-600 mt-1">
            Your personal and health data is always protected with us.
        </p>
    </div>

    <div className="bg-white p-5 rounded-xl shadow hover:shadow-lg transition">
        <h3 className="font-semibold text-lg text-blue-900">Patient Reviews</h3>
        <p className="text-gray-600 mt-1">
```

```

        Read real feedback from other patients before
you book.
    </p>
</div>
<div className="bg-white p-5 rounded-xl shadow
hover:shadow-lg transition">
    <h3 className="font-semibold text-lg
text-blue-900">Beyond Booking</h3>
    <p className="text-gray-600 mt-1">
        Get reminders, teleconsultations, and health
management tools.
    </p>
</div>
</div>
</div>
</div>
)
}
export default About

```

## ● CONTACT US PAGE (for user) :

**Contact Us**

Have questions or need help with your booking? Our team is here to assist you.

**Phone:** +91 98765 43210  
**Email:** support@mydoc.com  
**Address:** 123 Health Street, Kolkata, India

Your Name

Your Email

Your Message

Send Message

✓ CODE:

```
import { assets } from
"../assets/assets"

const Contact = () => {
  return (
    <div className="px-6 py-10">
      <div className="flex flex-col
md:flex-row items-center gap-8">
        <div className="md:w-1/3
w-full flex justify-center">
          <img
            src={assets.contact_image} alt="Contac
t" className="rounded-lg shadow-md
w-80 h-auto md:w-96"/>
        </div>

        <div className="md:w-2/3
w-full">
          <h2 className=" text-3xl
font-bold text-blue-900">Contact
Us</h2>
          <p className="text-gray-600
mt-2">Have questions or need help
with your booking? Our team is here
to assist you.</p>

          <div className="mt-6
space-y-2">
            <p
              className="text-gray-700"><strong>📞
Phone:</strong> +91 98765 43210</p>
            <p
              className="text-gray-700"><strong>✉️
Email:</strong> support@mydoc.com</p>
            <p
              className="text-gray-700"><strong>📍
Address:</strong> 123 Health Street,
Kolkata, India</p>
          </div>

          {/* Contact Form */}
          <form className="mt-6
space-y-4">
            <input
              type="text" placeholder="Your
Name" className="border
border-gray-300 p-3 rounded-lg w-full
focus:outline-none focus:ring-2
focus:ring-blue-500"/>
            <input
```

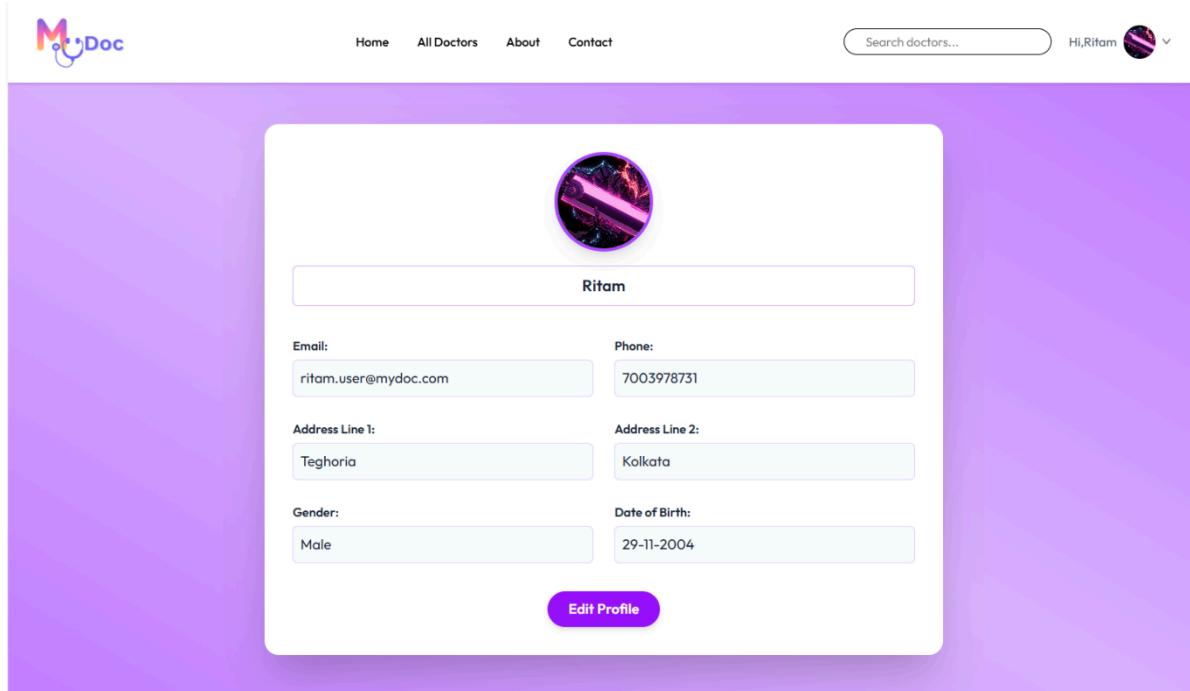
```

        type="email"placeholder="Your
Email"className="border
border-gray-300 p-3 rounded-lg w-full
focus:outline-none focus:ring-2
focus:ring-blue-500"/>
        <textarea
placeholder="Your
Message"rows="4"className="border
border-gray-300 p-3 rounded-lg w-full
focus:outline-none focus:ring-2
focus:ring-blue-500"></textarea>
        <button
type="submit"className="bg-purple-500
text-white py-3 px-6 rounded-lg
hover:bg-purple-700 transition"> Send
Message</button>
    </form>
</div>
</div>
</div>
)
}

```

```
export default Contact
```

## 1) My Profile Page :



## CODE :

```
export default Login
```

```
import React, { useContext, useState } from 'react'
```

```
import { ApplicationContext } from '../context/ApplicationContext'
import { assets } from '../assets/assets'
import axios from 'axios'
import { toast } from 'react-toastify'

const MyProfile = () => {
  const { userData, setUserData, token, backendUrl, loadUserProfileData } = useContext(ApplicationContext)
  const [isEdit, setIsEdit] = useState(false)
  const [image, setImage] = useState(false)
  const [loading, setLoading] = useState(false)

  const updateUserProfileData = async () => {
    setLoading(true)
    try {
      const formData = new FormData()
      formData.append('name', userData.name)
      formData.append('phone', userData.phone)
      formData.append('address', JSON.stringify(userData.address))
      formData.append('gender', userData.gender)
      formData.append('dob', userData.dob)

      if (image) formData.append('image', image)

      const { data } = await axios.post(
        `${backendUrl}/api/user/update-profile`,
        formData,
        { headers: { token } }
      )

      if (data.success) {
        toast.success(data.message)

        if (image) {
          setUserData(prev => ({ ...prev, image: URL.createObjectURL(image) }))
        }

        await loadUserProfileData()

        setIsEdit(false)
        setImage(false)
      } else {
        toast.error(data.message)
      }
    } catch (error) {
```

```

        console.error(error)
        toast.error(error.message)
    } finally {
        setLoading(false)
    }
}

const handleChange = (field, value) => {
    setUserData(prev => ({ ...prev, [field]: value }))
}

const handleAddressChange = (field, value) => {
    setUserData(prev => ({
        ...prev,
        address: { ...(prev.address || {}), [field]: value }
    }))
}

const handleToggleEdit = () => {
    if (isEdit) {
        updateUserProfileData() // Save changes
    } else {
        setIsEdit(true) // Enable edit mode
    }
}

return userData && (
    <div className="min-h-screen flex items-center justify-center bg-gradient-to-tr
from-purple-400 via-purple-300 to-purple-400 p-6">
    <div className="bg-white text-gray-800 rounded-2xl shadow-2xl p-8 w-full max-w-3xl">
        <div className="flex flex-col items-center">
            {isEdit ? (
                <label htmlFor="image">
                    <div className="relative w-28 h-28">
                        <img
                            src={image ? URL.createObjectURL(image) : userData.image}
                            alt="Profile"
                            className="w-28 h-28 rounded-full object-cover border-4 border-purple-500
shadow-lg"
                        />
                {!image && (
                    <img
                        src={assets.upload_icon}
                        alt="Upload"
                        className="absolute bottom-0 right-0 w-8 h-8 bg-white rounded-full p-1
border border-purple-300"
                )}
            ) : (
                <div>
                    <img
                        alt="Profile Picture"
                        className="w-28 h-28 rounded-full object-cover border-4 border-purple-500
shadow-lg"
                    />
                )}
            </div>
        </div>
    </div>
)

```

```

        />
    )}
</div>
<input
  type="file"
  id="image"
  hidden
  onChange={(e) => setImage(e.target.files[0])}
/>
</label>
) : (
<img
  src={userData.image}
  alt="Profile"
  className="w-28 h-28 rounded-full object-cover border-4 border-purple-500
shadow-lg"
/>
)}

<div className="w-full mt-4 text-center">
{isEdit ? (
<input
  type="text"
  className="w-full text-lg font-semibold text-center border border-purple-300
rounded-md p-2 focus:outline-none focus:ring-2 focus:ring-purple-500"
  value={userData.name}
  placeholder="NAME"
  onChange={e => handleChange("name", e.target.value)}
/>
) : (
<div className="w-full border border-purple-300 rounded-md p-2 text-lg
font-semibold text-center">
  {userData.name || 'NAME'}
</div>
)}
</div>
</div>

/* Profile Fields */
<div className="grid grid-cols-1 sm:grid-cols-2 gap-6 mt-8">
{[
  ['Email', 'email', 'email'],
  ['Phone', 'phone', 'text'],
  ['Address Line 1', 'line1', 'text', true],
  ['Address Line 2', 'line2', 'text', true],
  ['Gender', 'gender', 'select'],

```

```

        ['Date of Birth', 'dob', 'date']
    ].map(([label, key, type, isAddress]) => {
  const value = isAddress ? userData.address?.[key] || "" : userData[key] || ""
  return (
    <div key={key}>
      <label className="text-sm font-semibold">{label}:</label>
      {isEdit ? (
        type === 'select' ? (
          <select
            className="w-full mt-1 border border-purple-300 rounded-md p-2
focus:outline-none focus:ring-2 focus:ring-purple-500"
            value={userData.gender}
            onChange={e => handleChange("gender", e.target.value)}
          >
            <option>Male</option>
            <option>Female</option>
            <option>Other</option>
          </select>
        ) : (
          <input
            type={type}
            className="w-full mt-1 border border-purple-300 rounded-md p-2
focus:outline-none focus:ring-2 focus:ring-purple-500"
            value={value || ""}
            onChange={e =>
              isAddress
                ? handleAddressChange(key, e.target.value)
                : handleChange(key, e.target.value)
            }
            placeholder={label}
          />
        )
      ) : (
        <div className="w-full mt-1 border border-purple-200 rounded-md p-2
bg-gray-50 min-h-[42px]">
          {value || <span className="text-gray-400">{label}</span>}
        </div>
      )
    </div>
  )
})}
</div>

/* Button */
<div className="mt-8 text-center">
  <button

```

```

        onClick={handleToggleEdit}
        disabled={loading}
        className={`bg-purple-600 hover:bg-purple-700 text-white font-semibold px-6 py-2 rounded-full shadow-md transition duration-200 ease-in-out ${loading ? 'opacity-50 cursor-not-allowed' : ''}`}
      >
    {loading ? 'Saving...' : isEdit ? 'Save' : 'Edit Profile'}
  </button>
</div>
</div>
</div>
)
}

export default MyProfile

```

## 2) My Appointments Page:

The screenshot shows the 'My Appointments' section of the MyDoc app. It displays three appointment entries:

- Dr. Ivan Roy**: Pediatrician, Address: Action Area I, New Town, Kolkata. Date & Time: 16 August 2025 | 01:30 PM. Status: Confirmed. Buttons: Pay Online, Cancel Appointment.
- Dr. Abhijit Dutta**: General Physician, Address: City Centre, Durgapur, West Bengal. Date & Time: 18 August 2025 | 02:30 PM. Status: Confirmed. Buttons: Pay Online, Cancel Appointment.
- Dr. Abhijit Dutta**: General Physician, Address: City Centre, Durgapur, West Bengal. Date & Time: (not visible). Status: Appointment cancelled. Buttons: (disabled).

```

import React, { useContext } from "react"
import { AppContext } from "../context/AppContext"
import { useState } from "react"
import axios from "axios"
import { toast } from "react-toastify"
import { useEffect } from "react"
import { useNavigate } from "react-router-dom"

```

```
import { createStaticHandler } from "react-router-dom"

const MyAppointments = () => {
  const { backendUrl, token, getDoctorsData } = useContext(AppContext)
  const [appointments, setAppointments] = useState([])
  const months = [
    "", "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"
  ]

  const slotDateFormat =(slotDate)=>{
    const dateArray = slotDate.split('_')
    return dateArray[0]+ " "+months[Number(dateArray[1])]+ " " + dateArray[2]
  }

  const navigate = useNavigate()

  const getUserAppointments = async () =>{
    try {
      const {data} = await axios.get(backendUrl + '/api/user/appointments',
{headers:{token}})

      if (data.success) {
        setAppointments(data.appointments.reverse())
        console.log(data.appointments)
      }
    } catch (error) {
      console.log(error)
      toast.error(error.message)
    }
  }

  const cancelAppointment = async (appointmentId) => {
    try {
      const {data} = await axios.post(backendUrl + '/api/user/cancel-appointment',
{appointmentId}, {headers:{token}})
      if (data.success) {
        toast.success(data.message)
        getUserAppointments()
        navigate("/my-appointments")
      }
    } catch (error) {
      console.log(error)
      toast.error(error.message)
    }
  }
}
```

```

const initPay = (order)=> {

  const options = {
    key: import.meta.env.VITE_RAZORPAY_KEY_ID, // Enter the Key ID generated from the
Dashboard
    amount: order.amount, // Amount is in currency subunits. Default currency is INR.
    currency: order.currency,
    name: "Appointment Payment",
    description: "Appointment Payment for doctor",
    order_id: order.id, //This is the order_id created by you in the backend
    receipt: order.receipt,
    handler : async (response) => {
      console.log(response)

      try {
        const {data} = await axios.post(backendUrl + '/api/user/verifyRazorpay',response,
{headers:{token}})
        if (data.success) {
          toast.success(data.message)
          getUserAppointments()
          getDoctorsData()
        } else {
          toast.error(data.message)
        }
      } catch (error) {

      }
    }
  }

  const rzp = new window.Razorpay(options);
  rzp.open();
}

const appointmentRazorpay = async (appointmentId) => {
  try {
    const {data} = await axios.post(backendUrl + '/api/user/payment-razorpay',
{appointmentId}, {headers:{token}})
    if (data.success) {
      initPay(data.order)
      toast.success("Payment initiated")
    } else {
      toast.error(data.message)
    }
  }
}

```

```

    } catch (error) {
      console.log(error)
      toast.error(error.message)
    }
  }

useEffect(()=>{
  if (token) {
    getUserAppointments()
  }
},[token])

return (
  <div className="p-6 bg-gray-50 min-h-screen">
    <h1 className="text-3xl font-bold text-gray-800 mb-6 border-b pb-2">
      My Appointments
    </h1>

    {/* Appointment List */}
    <div className="space-y-6">
      {appointments.map((item, index) => (
        <div
          key={index}className="bg-white shadow-lg rounded-xl p-4 flex flex-col
md:flex-row items-center gap-6 hover:shadow-xl transition-shadow duration-300">
          {/* Doctor Image */}
          <div className="flex-shrink-0">
            <img src={item.docData.image} alt={item.name} className="w-28 h-28 object-cover
rounded-full border-4 border-blue-100"/>
          </div>

          <div className="flex-1">
            <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between">
              <div>
                <p className="text-xl font-semibold text-gray-800">{item.docData.name}</p>
                <p className="text-gray-500">{item.docData.speciality}</p>
                <p className="mt-2 text-gray-600 font-medium">Address:</p>
                <p className="text-gray-500">{item.docData.address.line1}</p>
                <p className="text-gray-500">{item.docData.address.line2}</p>
                <p className="mt-2 text-sm text-gray-600"><span
className="font-semibold">Date & Time:</span> {slotDateFormat(item.slotDate)} |<br/>
{item.slotTime}</p>
              </div>
            </div>
          </div>
        </div>
      )}
    </div>
  )
}

/* Appointment Status */
<span className="mt-2 sm:mt-0 px-3 py-1 rounded-full text-sm font-medium

```

```
bg-green-100 text-green-700">Confirmed</span>
        </div>
    </div>

    {/* Action Buttons */}
<div className="flex flex-col gap-2">
    {!item.cancelled && !item.isCompleted && item.payment && (
        <button className="bg-green-500 hover:bg-green-700 text-white py-2 px-4 rounded-lg text-sm transition-colors duration-300">
            Paid
        </button>
    )}

    {!item.cancelled && !item.isCompleted && !item.payment && (
        <button
            onClick={() => appointmentRazorpay(item._id)}
            className="bg-purple-500 hover:bg-purple-700 text-white py-2 px-4 rounded-lg text-sm transition-colors duration-300"
        >
            Pay Online
        </button>
    )}
}

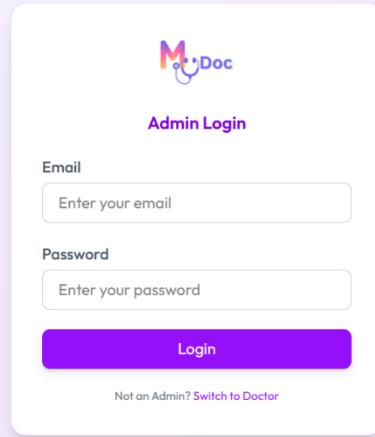
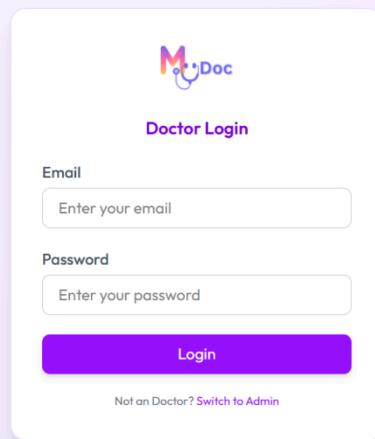
    {!item.cancelled && !item.isCompleted && (
        <button
            onClick={() => cancelAppointment(item._id, item.docData._id)}
            className="bg-red-500 hover:bg-red-700 text-white py-2 px-4 rounded-lg text-sm transition-colors duration-300"
        >
            Cancel Appointment
        </button>
    )}
}

    {item.cancelled && (
        <button className="sm:min-w-48 py-2 border border-red-500 rounded text-red-500">
            Appointment cancelled
        </button>
    )}
}

    {item.isCompleted && !item.cancelled && (
        <button className="sm:min-w-48 py-2 border border-green-500 rounded text-green-500">
            Appointment completed
        </button>
    )}
</div>
```

```
        </div>
    )}
</div>
</div>
)
}
export default MyAppointments
```

### **3) LOGIN PAGE (for admin and doctor):**



**CODE:**

```
import React, { useContext, useState } from "react"
import { assets } from "../assets/assets"
import { AdminContext } from "../context/AdminContext"
import axios from "axios"
import { toast } from 'react-toastify'
import { DoctorContext } from "../context/DoctorContext"
import {useNavigate} from "react-router-dom"

const Login = () => {
  const [state, setState] = useState("Admin")

  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const {setAToken, backendUrl} = useContext(AdminContext)
  const {setDToken} = useContext(DoctorContext)
  const navigate = useNavigate()
  const onSubmitHandler = async (event) => {
    event.preventDefault()

    try {
      if (state === 'Admin'){

        const {data} = await axios.post(backendUrl + '/api/admin/login',
{email,password})
        if (data.success){
          localStorage.setItem('aToken',data.token)
          setAToken(data.token)
          navigate("/admin-dashboard")
        }
        else {
          toast.error(data.message)
        }
      }
      else
      {
        const {data} = await axios.post(backendUrl + '/api/doctor/login',
{email,password})
        if (data.success){
          localStorage.setItem('dToken',data.token)
          setDToken(data.token)
          console.log(data.token)
          navigate("/doctor-dashboard")
        }
        else {
          toast.error(data.message)
        }
      }
    }
  }
}
```

```

        }
    }
    catch (error) {
        console.log(error)
    }
}

return (
    <form onSubmit={onSubmitHandler} className="min-h-[100vh] flex items-center justify-center bg-gradient-to-br from-cyan-200 via-purple-200 to-blue-300">
        <div className="flex flex-col gap-5 bg-white m-auto items-start p-8 w-full max-w-sm rounded-2xl shadow-xl border border-purple-200">

            <div className="w-full flex justify-center">
                <img
                    src={assets.myDoc_logo}
                    alt="Logo"
                    className="w-19 h-14 object-contain"
                />
            </div>

            <h2 className="text-lg font-semibold text-purple-700 w-full text-center">{state}</h2>
            Login </h2>
            <div className="w-full">
                <label className="block mb-1 text-gray-600 font-medium">Email</label>
                <input onChange={(e)=>setEmail(e.target.value)} value={email} type="email" required placeholder="Enter your email" className="w-full px-4 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500 border-gray-300 transition"/>
            </div>

            <div className="w-full">
                <label className="block mb-1 text-gray-600 font-medium">Password</label>
                <input onChange={(e)=>setPassword(e.target.value)} value={password} type="password" required placeholder="Enter your password" className="w-full px-4 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500 border-gray-300 transition"/>
            </div>
            <button type="submit" className="w-full py-2 bg-purple-600 text-white rounded-lg hover:bg-purple-700 transition transform hover:scale-[1.02] active:scale-[0.98] shadow-md">
                Login
            </button>
            {/* Switch user type */}
            <p className="text-xs text-gray-500 text-center w-full">
                Not an {state}?{" "}
                <span className="text-purple-600 cursor-pointer hover:underline" onClick={() => setState(state === "Admin" ? "Doctor" : "Admin")}> Switch to {state === "Admin" ? "Doctor" : "Admin"}</span>
            </p>
        </div>
    </div>
</form>

```

```

        </p>
      </div>
    </form>
  )
}

export default Login

```

#### 4) ADMIN DASHBOARD PAGE:

The screenshot displays the 'Admin Panel' interface. At the top, there's a navigation bar with the 'Logout' button. The main area is titled 'Admin Dashboard' and features three summary cards: '15 Doctors', '39 Appointments', and '12 Patients'. To the left, a sidebar contains links for 'Dashboard' (which is active), 'Appointments', 'Add Doctor', and 'Doctors List'. Below the sidebar is a section titled 'Latest Bookings' listing three recent appointments with their status (Pending or Completed).

#### ✓ CODE:

```

import React, { useContext, useEffect } from "react";
import { AdminContext } from "../../context/AdminContext";
import { assets } from "../../assets/assets";
import { AppContext } from "../../context/AppContext";

const Dashboard = () => {
  const { aToken, getDashData, dashData, cancelAppointment } =
    useContext(AdminContext);

  const { slotDateFormat } = useContext(AppContext);

  useEffect(() => {
    if (aToken) {
      getDashData();
    }
  }, [aToken, getDashData]);
}

```

```
return (
  dashData && (
    <div className="m-5 w-full">
      <h1 className="text-2xl font-bold mb-6 text-gray-800">
         Admin Dashboard
      </h1>

      {/* Top summary cards */}
      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6">
        {/* Doctors */}
        <div className="bg-white p-6 rounded-2xl shadow-md flex items-center gap-4
        hover:shadow-lg transition">
          <div className="bg-purple-100 p-4 rounded-full flex items-center
        justify-center">
            <img src={assets.doctor_icon} alt="Doctors" className="w-10 h-10" />
          </div>
          <div>
            <p className="text-3xl font-semibold text-purple-600">
              {dashData.doctors}
            </p>
            <p className="text-gray-500">Doctors</p>
          </div>
        </div>
      
```

```

        {/* Appointments */}
        <div className="bg-white p-6 rounded-2xl shadow-md flex items-center gap-4
        hover:shadow-lg transition">
          <div className="bg-blue-100 p-4 rounded-full flex items-center justify-center">
            <img
              src={assets.appointment_icon}
              alt="Appointments"
              className="w-10 h-10"
            />
          </div>
          <div>
            <p className="text-3xl font-semibold text-blue-600">
              {dashData.appointments}
            </p>
            <p className="text-gray-500">Appointments</p>
          </div>
        </div>
      
```

```

        {/* Patients */}
        <div className="bg-white p-6 rounded-2xl shadow-md flex items-center gap-4
        hover:shadow-lg transition">
          <div className="bg-green-100 p-4 rounded-full flex items-center justify-center">

```

```

        <img src={assets.patients_icon} alt="Patients" className="w-10 h-10" />
    </div>
    <div>
        <p className="text-3xl font-semibold text-green-600">
            {dashData.patients}
        </p>
        <p className="text-gray-500">Patients</p>
    </div>
    </div>
</div>

/* Latest Bookings */
<div className="bg-white mt-10 rounded-2xl shadow-md p-6">
    /* Header */
    <div className="flex items-center gap-2.5 mb-5">
        <img src={assets.list_icon} alt="" className="w-5 h-5" />
        <p className="font-semibold text-lg text-gray-800">Latest Bookings</p>
    </div>

    /* List */
    <div className="grid gap-4 max-h-[350px] overflow-y-auto pr-1">
        {dashData.latestAppointment.map((item, index) => (
            <div
                key={index}
                className="flex items-center justify-between p-4 bg-gray-50 rounded-xl
shadow-sm hover:shadow-md transition"
            >
                /* Left: doctor info */
                <div className="flex items-center gap-4">
                    <img
                        className="rounded-full w-12 h-12 border border-gray-200 object-cover"
                        src={item.docData.image}
                        alt={item.docData.name}
                    />
                    <div className="flex flex-col justify-center">
                        <p className="text-gray-800 font-medium">{item.docData.name}</p>
                        <p className="text-gray-500 text-sm">
                            {slotDateFormat(item.slotDate)}
                        </p>
                    </div>
                </div>
            </div>
        ))
    </div>

    /* Right: status or cancel */
    <div className="flex items-center">
        {item.cancelled ? (

```

```

        <p className="text-red-600 font-semibold">Cancelled</p>
      ) : item.isCompleted ? (
        <p className="text-green-600 font-semibold">Completed</p>
      ) : (
        <p className="text-yellow-600 font-semibold">Pending</p>
      )}
    </div>
  )
))}

</div>
</div>
</div>
)
);
};

export default Dashboard;

```

**component:- Navbar.jsx :**

```

import React, { useContext } from "react"
import { assets } from "../assets/assets"
import { AdminContext } from "../context/AdminContext.jsx"
import { useNavigate } from "react-router-dom"
import { DoctorContext } from "../context/DoctorContext.jsx"

const Navbar = () => {
  const { aToken, setAToken } = useContext(AdminContext)
  const { dToken, setDToken } = useContext(DoctorContext)

  const navigate = useNavigate()

  const logout = () => {
    navigate("/")
    aToken && setAToken("")
    aToken && localStorage.removeItem("aToken")
    dToken && setDToken('')
    dToken && localStorage.removeItem('dToken')
  }

};

return (
  <div className="flex items-center justify-between text-sm py-3 border-b border-b-purple-400 bg-white sticky top-0 z-90 shadow-sm px-4 md:px-8">
    <div className="flex items-center gap-130 text-2xl">
      <img

```

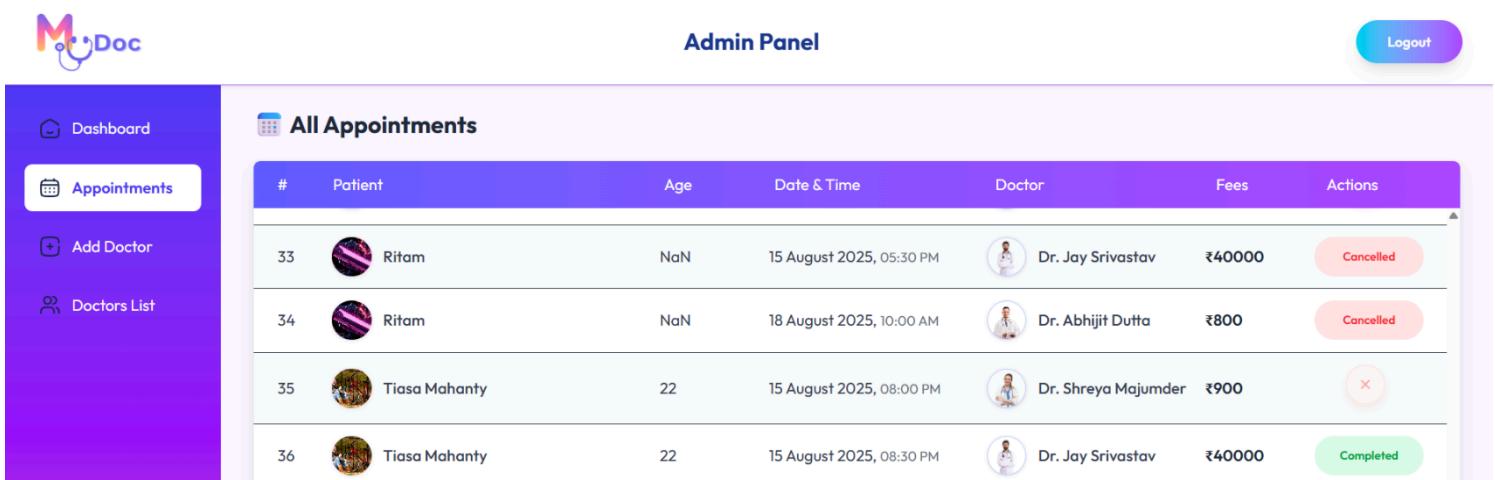
```

        className="h-16 sm:w-27 cursor-pointer"
        src={assets.myDoc_logo}
        alt=""
      />
      <p className=" px-8 py-3 text-blue-900 font-bold transition-all duration-200 ">
        {aToken ? "Admin" : "Doctor"} Panel
      </p>
    </div>
    <button
      onClick={logout}
      className='bg-gradient-to-r from-cyan-400 to-purple-500 px-8 py-3 rounded-full
text-white text-sm font-semibold shadow-lg hover:shadow-xl hover:scale-105 transition-all
duration-300 ease-in-out'
    >
      Logout
    </button>
  </div>
)
}

export default Navbar

```

## 5) ADMIN ALL APPOINTMENTS PAGE:



The screenshot shows the 'All Appointments' section of the Admin Panel. The left sidebar has 'Dashboard' (selected), 'Appointments' (selected), 'Add Doctor', and 'Doctors List'. The main area title is 'All Appointments'. The table lists four patient entries:

#	Patient	Age	Date & Time	Doctor	Fees	Actions
33	Ritam	NaN	15 August 2025, 05:30 PM	Dr. Jay Srivastav	₹40000	<span>Cancelled</span>
34	Ritam	NaN	18 August 2025, 10:00 AM	Dr. Abhijit Dutta	₹800	<span>Cancelled</span>
35	Tiasa Mahanty	22	15 August 2025, 08:00 PM	Dr. Shreya Majumder	₹900	<span>Cancel</span>
36	Tiasa Mahanty	22	15 August 2025, 08:30 PM	Dr. Jay Srivastav	₹40000	<span>Completed</span>

The top right has a 'Logout' button.

✓ CODE:

```
import React, { useContext, useEffect } from "react";
import { AdminContext } from "../../context/AdminContext";
import { AppContext } from "../../context/AppContext";
import { assets } from "../../assets/assets";

const AllAppointments = () => {
  const { aToken, appointments, getAllAppointments, cancelAppointment } =
    useContext(AdminContext);
  const { calculateAge, slotDateFormat, currency } = useContext(AppContext);

  useEffect(() => {
    if (aToken) {
      getAllAppointments();
    }
  }, [aToken]);

  return (
    <div className="w-full max-w-7xl mx-auto p-6">
      <h2 className="text-2xl font-bold text-gray-800 mb-5">
         All Appointments
      </h2>

      <div className="bg-white shadow-lg rounded-lg overflow-hidden border border-gray-200">
        {/* Table Header */}
        <div className="hidden sm:grid grid-cols-[0.5fr_3fr_1fr_2fr_2fr_1fr_1fr] bg-gradient-to-r from-indigo-500 to-purple-500 text-white py-3 px-6 sticky top-0">
          <p>#</p>
          <p>Patient</p>
          <p>Age</p>
          <p>Date & Time</p>
          <p>Doctor</p>
          <p>Fees</p>
          <p>Actions</p>
        </div>

        {/* Table Body */}
        <div className="max-h-[70vh] overflow-y-auto">
          {appointments.reverse().map((item, index) => (
            <div
              key={index}
              className={`sm:grid sm:grid-cols-[0.5fr_3fr_1fr_2fr_2fr_1fr_1fr] items-center text-gray-700 py-3 px-6 border-b
${
            item
          }}>
            <div>
              <img alt="Patient icon" data-bbox="100 100 120 120"/> {item.patient}</div>
              <div>
                {item.date} {item.time}</div>
              <div>
                {item.doctor}</div>
              <div>
                ${calculateAge(item.age)}</div>
              <div>
                ${slotDateFormat(item.slot)}</div>
              <div>
                ${currency(item.fees)}</div>
              <div>
                <button onClick={() => cancelAppointment(item._id)}>Cancel</button>
              </div>
            </div>
          ))}</div>
        </div>
      </div>
    )</div>
  );
}

export default AllAppointments;
```

```

        index % 2 === 0 ? "bg-gray-50" : "bg-white"
    } hover:bg-indigo-50 transition`}
>
/* Index */
<p className="max-sm:hidden">{index + 1}</p>

/* Patient */
<div className="flex items-center gap-3">
    <img
        className="w-10 h-10 rounded-full object-cover shadow-sm"
        src={item.userData.image}
        alt={item.userData.name}
    />
    <p className="font-medium">{item.userData.name}</p>
</div>

/* Age */
<p className="max-sm:hidden">{calculateAge(item.userData.dob)}</p>

/* Date & Time */
<p>
    {slotDateFormat(item.slotDate)}, {" "}
    <span className="text-sm text-gray-500">{item.slotTime}</span>
</p>
/* Doctor */
<div className="flex items-center gap-3">
    <img
        className="w-10 h-10 rounded-full object-cover border border-indigo-200
shadow-sm"
        src={item.docData.image}
        alt={item.docData.name}
    />
    <p className="font-medium">{item.docData.name}</p>
</div>

/* Fees */
<p className="font-semibold text-gray-800">
    {currency}
    {item.amount}
</p>

/* Actions */
{item.cancelled ?
    <p className="text-red-500 text-xs font-semibold bg-red-100 px-3 py-3
rounded-full text-center">
        Cancelled
    </p>
}

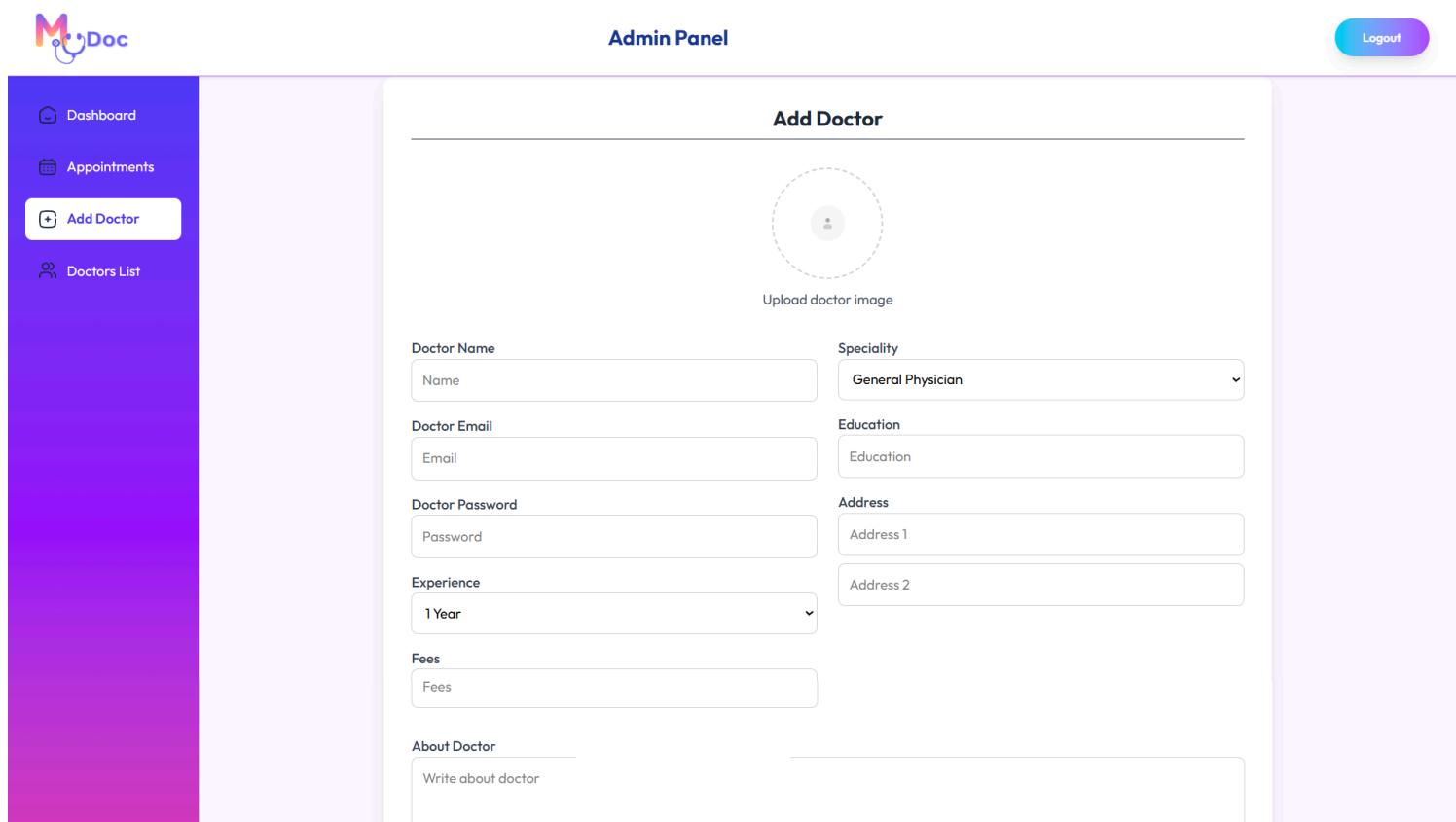
```

```

        </p> : item.isCompleted ? <p className="text-green-600 text-xs font-semibold bg-green-100 px-3 py-3 rounded-full text-center">Completed</p>
          :<img onClick={()=> cancelAppointment(item._id)}
            className="w-12 h-12 mx-7 cursor-pointer"
            src={assets.cancel_icon}
            alt="Cancel"
          />
        }
      </div>
    ))
</div>
</div>
</div>
);
};

```

## 1) ADMIN ADD DOCTORS PAGE:



The screenshot shows the Admin Panel interface. On the left, there's a sidebar with a purple gradient background containing navigation links: Dashboard, Appointments, Add Doctor (highlighted with a white background), and Doctors List. The main content area has a light blue header bar with the text "Admin Panel". Below this is a form titled "Add Doctor". The form includes fields for "Doctor Name" (with a "Name" input field), "Specialty" (with a dropdown menu set to "General Physician"), "Doctor Email" (with an "Email" input field), "Education" (with an "Education" input field), "Doctor Password" (with a "Password" input field), "Address" (with two input fields for "Address 1" and "Address 2"), "Experience" (with a dropdown menu set to "1 Year"), "Fees" (with an "Fees" input field), and an "About Doctor" text area with placeholder text "Write about doctor". In the top right corner of the main area, there is a "Logout" button.

## ✓ CODE

```
import React, { useContext, useState } from "react"
import { assets } from "../../assets/assets"
import { AdminContext } from "../../../context/AdminContext"
import { toast } from "react-toastify"
import axios from "axios"

const AddDoctor = () => {
  const [docImg, setDocImg] = useState(false)
  const [name, setName] = useState("")
  const [email, setEmail] = useState("")
  const [password, setPassword] = useState("")
  const [experience, setExperience] = useState("1 Year")
  const [fees, setFees] = useState("")
  const [about, setAbout] = useState("")
  const [speciality, setSpeciality] = useState("General Physician")
  const [degree, setDegree] = useState("")
  const [address1, setAddress1] = useState("")
  const [address2, setAddress2] = useState("")

  const {backendUrl, aToken } = useContext(AdminContext)

  const onSubmitHandler = async (event) => {
    event.preventDefault()

    try {
      if (!docImg) {
        return toast.error('Image Not Selected')
      }

      const formData = new FormData()

      formData.append('image',docImg)
      formData.append('name',name)
      formData.append('email',email)
      formData.append('password',password)
      formData.append('experience',experience)
      formData.append('fees',Number(fees))
      formData.append('about',about)
      formData.append('speciality',speciality)
      formData.append('degree',degree)
      formData.append('address',JSON.stringify({line1:address1, line2:address2}))

      //console.log(formData)
    }
  }
}
```

```

    formData.forEach((value, key) => {
      console.log(` ${key}: ${value}`);
    })

    const {data} = await axios.post(backendUrl + '/api/admin/add-doctor',
formData,{headers : {aToken}})

    if(data.success)
    {
      toast.success(data.message)
      setDocImg(false)
      setName("")
      setPassword("")
      setEmail("")
      setAddress1("")
      setAddress2("")
      setDegree("")
      setAbout("")
      setFees("")
    }
    else{
      toast.error(data.message)
    }

  } catch (error) {
    toast.error(error.message)
    console.log(error)
  }
}
return (
  <form onSubmit={onSubmitHandler} className="max-w-5xl mx-auto bg-white shadow-lg rounded-xl p-8 space-y-8">
    {/* Title */}
    <h2 className="text-2xl text-center font-bold text-gray-800 border-b pb-2">
      Add Doctor
    </h2>

    {/* Upload Area */}
    <div className="flex flex-col items-center gap-3">
      <label
        htmlFor="doc-img"
        className="w-32 h-32 rounded-full border-2 border-dashed border-gray-300 flex items-center justify-center cursor-pointer overflow-hidden hover:border-indigo-500 transition-all">
      </>
      {docImg ? (

```

```
<img
  src={URL.createObjectURL(docImg)}
  alt="Doctor"
  className="w-full h-full object-cover"
/>
) : (
  <img
    src={assets.upload_area}
    alt="Upload"
    className="w-10 h-10 object-contain"
  /
)
}

</label>
<input
  onChange={(e) => setDocImg(e.target.files[0])}
  type="file"
  id="doc-img"
  hidden
/>
{!docImg && (
  <p className="text-gray-600 text-center">Upload doctor image</p>
)
}
</div>

/* Two Column Form */
<div className="grid grid-cols-1 md:grid-cols-2 gap-6">
  /* Left column */
  <div className="space-y-4">
    <div>
      <p className="font-medium text-gray-700">Doctor Name</p>
      <input onChange={(e)=> setName(e.target.value)} value={name}
        type="text"
        placeholder="Name"
        required
        className="w-full border border-gray-300 rounded-lg p-3 focus:ring-2
focus:ring-indigo-500 outline-none"
      />
    </div>
    <div>
      <p className="font-medium text-gray-700">Doctor Email</p>
      <input onChange={(e)=> setEmail(e.target.value)} value={email}
        type="email"
        placeholder="Email"
        required
      />
    </div>
  </div>
</div>
```

```
        className="w-full border border-gray-300 rounded-lg p-3 focus:ring-2
focus:ring-indigo-500 outline-none"
      />
    </div>

<div>
  <p className="font-medium text-gray-700">Doctor Password</p>
  <input onChange={(e)=> setPassword(e.target.value)} value={password}
    type="password"
    placeholder="Password"
    required
    className="w-full border border-gray-300 rounded-lg p-3 focus:ring-2
focus:ring-indigo-500 outline-none"
  />
</div>

<div>
  <p className="font-medium text-gray-700">Experience</p>
  <select onChange={(e)=> setExperience(e.target.value)} value={experience}
className="w-full border border-gray-300 rounded-lg p-3 focus:ring-2 focus:ring-indigo-500
outline-none">
  {Array.from({ length: 10 }, (_, i) => (
    <option key={i + 1} value={`${i + 1} Year`}>
      {i + 1} Year
    </option>
  )))
  </select>
</div>

<div>
  <p className="font-medium text-gray-700">Fees</p>
  <input onChange={(e)=> setFees(e.target.value)} value={fees}
    type="number"
    placeholder="Fees"
    required
    className="w-full border border-gray-300 rounded-lg p-3 focus:ring-2
focus:ring-indigo-500 outline-none"
  />
</div>
</div>

/* Right column */
<div className="space-y-4">
  <div>
    <p className="font-medium text-gray-700">Speciality</p>
    <select onChange={(e)=> setSpeciality(e.target.value)} value={speciality}>
```

```
className="w-full border border-gray-300 rounded-lg p-3 focus:ring-2 focus:ring-indigo-500 outline-none">
    <option value="General Physician">General Physician</option>
    <option value="Gynecologist">Gynecologist</option>
    <option value="Dermatologist">Dermatologist</option>
    <option value="Pediatrician">Pediatrician</option>
    <option value="Neurologist">Neurologist</option>
    <option value="Gastroenterologist">Gastroenterologist</option>
</select>
</div>
<div>
    <p className="font-medium text-gray-700">Education</p>
    <input onChange={(e)=> setDegree(e.target.value)} value={degree} type="text" placeholder="Education" required
        className="w-full border border-gray-300 rounded-lg p-3 focus:ring-2 focus:ring-indigo-500 outline-none"
    />
</div>
<div>
    <p className="font-medium text-gray-700">Address</p>
    <input onChange={(e)=> setAddress1(e.target.value)} value={address1} type="text" placeholder="Address 1" required
        className="w-full mb-2 border border-gray-300 rounded-lg p-3 focus:ring-2 focus:ring-indigo-500 outline-none"
    />
    <input onChange={(e)=> setAddress2(e.target.value)} value={address2} type="text" placeholder="Address 2" required
        className="w-full border border-gray-300 rounded-lg p-3 focus:ring-2 focus:ring-indigo-500 outline-none"
    />
</div>
</div>

<div>
    <p className="font-medium text-gray-700">About Doctor</p>
    <textarea onChange={(e)=> setAbout(e.target.value)} value={about} placeholder="Write about doctor" rows={5} required>

```

```

        className="w-full border border-gray-300 rounded-lg p-3 focus:ring-2
focus:ring-indigo-500 outline-none"
      />
    </div>

    <div className="text-right">
      <button
        type="submit"
        className="bg-indigo-600 hover:bg-indigo-700 text-white px-6 py-3 rounded-lg
shadow-md font-medium transition-all">
        >
          Add Doctor
        </button>
    </div>
  </form>
)
}

```

```
export default AddDoctor
```

### ADMIN DOCTOR LIST PAGE:

The screenshot shows the 'All Doctors' section of the Admin Panel. The sidebar on the left has a purple background with white icons and text for Dashboard, Appointments, Add Doctor, and Doctors List (the current active tab). The main content area has a light blue background and displays ten doctor profiles in a grid. Each profile consists of a photo, the doctor's name, their title, and a green 'Available' status indicator with a checkmark. The doctors listed are Dr. Jay Srivastav, Dr. Sushmita Banerjee, Dr. Abhijit Dutta, Dr. Aranya Sen, Dr. Subhashree Mitra, and seven other doctors whose profiles are partially visible.

#### ✓ CODE:

```

import React, { useContext, useEffect } from 'react'
import { AdminContext } from '../context/AdminContext'

const DoctorsList = () => {
  const { doctors, aToken, getAllDoctors, changeAvailability } = useContext(AdminContext)

```

```

useEffect(() => {
  if (aToken) {
    getAllDoctors()
  }
}, [aToken])

return (
  <div className="p-6 w-full">
    <h1 className="text-3xl font-bold text-gray-800 mb-6">All Doctors</h1>
    <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 lg:grid-cols-4
lg:grid-cols-5 gap-6">
      {doctors.map((item, index) => (
        <div
          key={index}
          className="bg-indigo-50 hover:bg-indigo-200 rounded-2xl shadow-md
hover:shadow-lg transition-all duration-300 overflow-hidden border border-gray-100">
          <img src={item.image} alt={item.name} className="w-full h-48 object-cover"/>

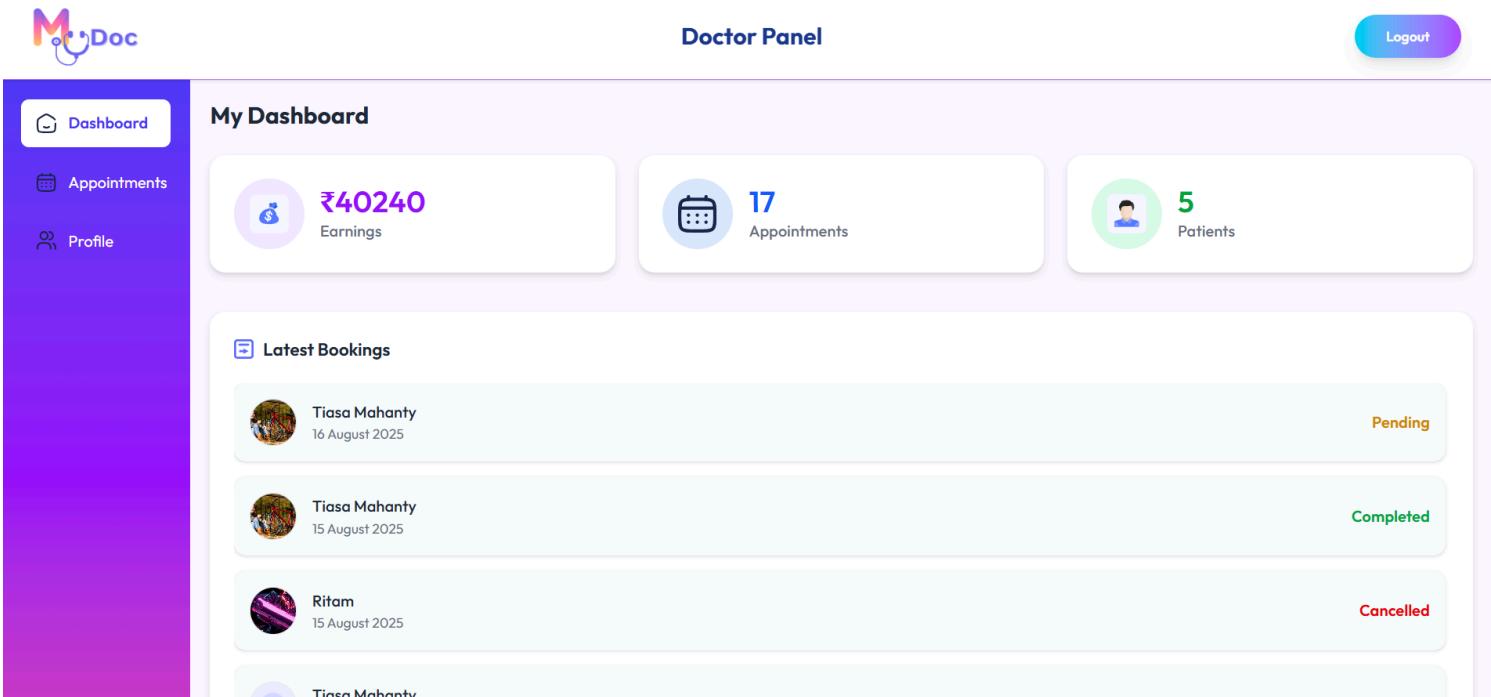
          <div className="p-4">
            <p className="text-lg font-semibold text-gray-800">{item.name}</p>
            <p className="text-sm text-gray-700 mb-3">{item.speciality}</p>

            <div className="flex items-center gap-2">
              <input type="checkbox" checked={item.available} onChange={()=>
changeAvailability(item._id)}
                    className="w-4 h-4 text-green-500 border-gray-300 rounded
focus:ring-green-400"/>
              <p className={`text-sm font-medium ${item.available ? 'text-green-600' :
'text-red-500'}`}>
                {item.available ? 'Available' : 'Not Available'}</p>
            </div>
          </div>
        </div>
      )))
    </div>
  </div>
)
}

export default DoctorsList

```

## 12) DOCTOR DASHBOARD PAGE:



The screenshot shows the Doctor Panel of the MyDoc app. At the top, there's a navigation bar with the 'MyDoc' logo, the title 'Doctor Panel', and a 'Logout' button. On the left, a sidebar menu includes 'Dashboard' (selected), 'Appointments', and 'Profile'. The main area is titled 'My Dashboard' and features three summary cards: 'Earnings' (₹40240), 'Appointments' (17), and 'Patients' (5). Below this, a section titled 'Latest Bookings' lists four entries with patient names, booking dates, and status indicators (Pending, Completed, Cancelled).

Booking ID	Patient Name	Date	Status
1	Tiasa Mahanty	16 August 2025	Pending
2	Tiasa Mahanty	15 August 2025	Completed
3	Ritam	15 August 2025	Cancelled
4	Tiasa Mahanty		

### ✓ CODE:

```
import React, { useContext, useEffect } from "react";
import { DoctorContext } from "../../context/DoctorContext";
import { assets } from "../../assets/assets";
import { AppContext } from "../../context/AppContext";

const DoctorDashboard = () => {
  const {
    dToken,
    dashData,
    getDashData
  } = useContext(DoctorContext);

  const { currency, slotDateFormat } = useContext(AppContext);

  useEffect(() => {
    if (dToken) {
      getDashData();
    }
  }, [dToken]);

  return (
    dashData && (
      <DoctorDashboard>
        <div>
          <div>
            <h2>My Dashboard</h2>
            <div>
              <div>
                <div>
                  <img alt="Earnings icon" />
                  ₹40240
                </div>
                Earnings
              </div>
              <div>
                <img alt="Appointments icon" />
                17
              </div>
              Appointments
            </div>
            <div>
              <img alt="Patients icon" />
              5
            </div>
            Patients
          </div>
          <div>
            <h3>Latest Bookings</h3>
            <table border="1">
              <thead>
                <tr>
                  <th>Booking ID</th>
                  <th>Patient Name</th>
                  <th>Date</th>
                  <th>Status</th>
                </tr>
              </thead>
              <tbody>
                <tr>
                  <td>1</td>
                  <td>Tiasa Mahanty</td>
                  <td>16 August 2025</td>
                  <td>Pending</td>
                </tr>
                <tr>
                  <td>2</td>
                  <td>Tiasa Mahanty</td>
                  <td>15 August 2025</td>
                  <td>Completed</td>
                </tr>
                <tr>
                  <td>3</td>
                  <td>Ritam</td>
                  <td>15 August 2025</td>
                  <td>Cancelled</td>
                </tr>
                <tr>
                  <td>4</td>
                  <td>Tiasa Mahanty</td>
                  <td></td>
                  <td></td>
                </tr>
              </tbody>
            </table>
          </div>
        </div>
      </DoctorDashboard>
    )
  );
}
```

```
<div className="m-5 w-full">
  <h1 className="text-2xl font-bold mb-6 text-gray-800">My Dashboard</h1>

  {/* Top summary cards */}
  <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6">
    {/* Earnings */}
    <div className="bg-white p-6 rounded-2xl shadow-md flex items-center gap-4
    hover:shadow-lg transition">
      <div className="bg-purple-100 p-4 rounded-full flex items-center
    justify-center">
        <img src={assets.earning_icon} alt="Earnings" className="w-10 h-10" />
      </div>
      <div>
        <p className="text-3xl font-semibold text-purple-600">
          {currency}{dashData.earnings}
        </p>
        <p className="text-gray-500 font-medium">Earnings</p>
      </div>
    </div>
  
```

```
  {/* Appointments */}
  <div className="bg-white p-6 rounded-2xl shadow-md flex items-center gap-4
  hover:shadow-lg transition">
    <div className="bg-blue-100 p-4 rounded-full flex items-center justify-center">
      <img src={assets.appointment_icon} alt="Appointments" className="w-10 h-10" />
    </div>
    <div>
      <p className="text-3xl font-semibold text-blue-600">
        {dashData.appointments}
      </p>
      <p className="text-gray-500 font-medium">Appointments</p>
    </div>
  </div>
```

```
  {/* Patients */}
  <div className="bg-white p-6 rounded-2xl shadow-md flex items-center gap-4
  hover:shadow-lg transition">
    <div className="bg-green-100 p-4 rounded-full flex items-center justify-center">
      <img src={assets.patients_icon} alt="Patients" className="w-10 h-10" />
    </div>
    <div>
      <p className="text-3xl font-semibold text-green-600">
        {dashData.patients}
      </p>
      <p className="text-gray-500 font-medium">Patients</p>
    </div>
```

```

        </div>
    </div>

    /* Latest Bookings */
    <div className="bg-white mt-10 rounded-2xl shadow-md p-6 w-full">
        <div className="flex items-center gap-2.5 mb-5 w-full">
            <img src={assets.list_icon} alt="" className="w-5 h-5" />
            <p className="font-semibold text-lg text-gray-800">Latest Bookings</p>
        </div>

        <div className="grid gap-4 max-h-[500px] overflow-y-auto pr-1">
            {dashData.latestAppointments.map((item, index) => (
                <div
                    key={index}
                    className="flex items-center justify-between p-4 bg-gray-50 rounded-xl
shadow-sm hover:shadow-md transition"
                >
                    /* User info */
                    <div className="flex items-center gap-4">
                        <img
                            className="rounded-full w-12 h-12 border border-gray-200 object-cover"
                            src={item.userData.image}
                            alt={item.userData.name}
                        />
                        <div className="flex flex-col justify-center">
                            <p className="text-gray-800 font-medium">{item.userData.name}</p>
                            <p className="text-gray-500 text-sm">{slotDateFormat(item.slotDate)}</p>
                        </div>
                    </div>
                </div>

                /* Status only */
                <div className="flex items-center">
                    {item.cancelled ? (
                        <p className="text-red-600 font-semibold">Cancelled</p>
                    ) : item.isCompleted ? (
                        <p className="text-green-600 font-semibold">Completed</p>
                    ) : (
                        <p className="text-yellow-600 font-semibold">Pending</p>
                    )}
                </div>
            </div>
        )))
    </div>
</div>

```

```

        )
    );
};

export default DoctorDashboard;

```

**component: Sidebar.jsx:-**

```

import React, { useContext } from "react"
import { AdminContext } from "../context/AdminContext"
import { NavLink } from "react-router-dom"
import { assets } from "../assets/assets"
import { DoctorContext } from "../context/DoctorContext"

const Sidebar = () => {
  const { aToken } = useContext(AdminContext)
  const { dToken } = useContext(DoctorContext)

  const navItemStyle =
    "flex items-center gap-3 px-4 py-3 rounded-lg w-full transition-all duration-300"

  return (
    <div className="flex min-h-screen w-55 bg-gradient-to-b from-indigo-600 via-purple-600
to-pink-500 p-5">
    {aToken && (
      <ul className="flex flex-col gap-3 w-full">
        <NavLink
          to="/admin-dashboard"
          className={({ isActive }) =>
            `${navItemStyle} ${{
              isActive
                ? "bg-white text-indigo-600 font-semibold"
                : "text-white hover:bg-white hover:text-indigo-600"
            }}`}
        >
          <img src={assets.home_icon} alt="Dashboard" className="w-5" />
          <p className="hidden md:block">Dashboard</p>
        </NavLink>

        <NavLink
          to="/all-appointments"
          className={({ isActive }) =>
            `${navItemStyle} ${{
              isActive
                ? "bg-white text-indigo-600 font-semibold"
                : "text-white hover:bg-white hover:text-indigo-600"
            }}`}
        >
    
```

```

        }
    >
    <img
        src={assets.appointment_icon}
        alt="Appointments"
        className="w-5"
    />
    <p className="hidden md:block">Appointments</p>
</NavLink>

<NavLink
    to="/add-doctor"
    className={({ isActive }) =>
        `${navItemStyle} ${{
            isActive
                ? "bg-white text-indigo-600 font-semibold"
                : "text-white hover:bg-white hover:text-indigo-600"
        }}`
    }
    >
    <img src={assets.add_icon} alt="Add Doctor" className="w-5" />
    <p className="hidden md:block">Add Doctor</p>
</NavLink>

<NavLink
    to="/doctor-list"
    className={({ isActive }) =>
        `${navItemStyle} ${{
            isActive
                ? "bg-white text-indigo-600 font-semibold"
                : "text-white hover:bg-white hover:text-indigo-600"
        }}`
    }
    >
    <img src={assets.people_icon} alt="Doctors List" className="w-5" />
    <p className="hidden md:block">Doctors List</p>
</NavLink>
</ul>
)}

```

```

{dToken && (
    <ul className="flex flex-col gap-3 w-full">
        <NavLink
            to="/doctor-dashboard"
            className={({ isActive }) =>
                `${navItemStyle} ${{

```

```

        isActive
          ? "bg-white text-indigo-600 font-semibold"
          : "text-white hover:bg-white hover:text-indigo-600"
      }` 
    }
  >
  <img src={assets.home_icon} alt="Dashboard" className="w-5" />
  <p className="hidden md:block">Dashboard</p>
</NavLink>

<NavLink
  to="/doctor-appointments"
  className={({ isActive }) =>
    `${navItemStyle} ${{
      isActive
        ? "bg-white text-indigo-600 font-semibold"
        : "text-white hover:bg-white hover:text-indigo-600"
    }}` 
  }
>
  <img
    src={assets.appointment_icon}
    alt="Appointments"
    className="w-5"
  />
  <p className="hidden md:block">Appointments</p>
</NavLink>

<NavLink
  to="/doctor-profile"
  className={({ isActive }) =>
    `${navItemStyle} ${{
      isActive
        ? "bg-white text-indigo-600 font-semibold"
        : "text-white hover:bg-white hover:text-indigo-600"
    }}` 
  }
>
  <img src={assets.people_icon} alt="Doctors List" className="w-5" />
  <p className="hidden md:block">Profile</p>
</NavLink>
</ul>
)}
</div>
)
}
export default Sidebar

```

### 13) APPOINTMENTS PAGE FOR DOCTOR:

✓ CODE:

#	Patient	Payment	Age	Date & Time	Fees	Action
1	Tiasa Mahanty	Cash	22	16 August 2025, 10:30 AM	₹400	
2	Tiasa Mahanty	Cash	22	15 August 2025, 08:30 PM	₹40000	Completed
3	Ritam	Cash	N/A	15 August 2025, 05:30 PM	₹40000	Cancelled
4	Tiasa Mahanty	Cash	N/A	15 August 2025, 01:30 PM	₹40000	
5	Sandipan	Cash	21	15 August 2025, 03:00 PM	₹40	Cancelled
6	Sandipan	Cash	21	15 August 2025, 06:00 PM	₹40	Completed
7	Sandy	Cash	N/A	15 August 2025, 08:00 PM	₹40	Completed
8	Sandy	Cash	N/A	15 August 2025, 07:00 PM	₹40	Completed

```

import React, { useContext, useEffect } from "react";
import { DoctorContext } from "../../context/DoctorContext";
import { AppContext } from "../../context/AppContext";
import { assets } from "../../assets/assets";

const DoctorAppointments = () => {
  const {
    dToken,
    appointments,
    getAppointments,
    completeAppointment,
    cancelAppointment,
  } = useContext(DoctorContext);

  const { calculateAge, slotDateFormat, currency } = useContext(AppContext);

  useEffect(() => {
    if (dToken) {
      getAppointments();
    }
  }, [dToken]);

  return (
    <div className="w-full p-6">
      <p className="mb-4 text-xl font-
        tor Appointments</p>
      <div className="bg-white border border-gray-200 rounded-lg shadow-md overflow-hidden">
    
```

```

/* Table Header */
<div className="sticky top-0 grid grid-cols-[60px_1.8fr_1fr_1fr_2fr_1fr_1fr] py-3 px-6 border-b text-white font-semibold bg-gradient-to-r from-[#6A11CB] to-[#2575FC] z-10">
  <p>#</p>
  <p>Patient</p>
  <p>Payment</p>
  <p>Age</p>
  <p>Date & Time</p>
  <p>Fees</p>
  <p>Action</p>
</div>

/* Table Rows */
{appointments.length > 0 ? (
  appointments
    .slice()
    .reverse()
    .map((item, index) => (
      <div
        key={index}
        className="grid grid-cols-[60px_1.8fr_1fr_1fr_2fr_1fr_1fr] items-center py-3 px-6 border-b border-gray-200 hover:bg-gray-50 transition"
      >
        /* Index */
        <p>{index + 1}</p>

        /* Patient */
        <div className="flex items-center gap-2">
          <img
            className="w-9 h-9 rounded-full object-cover border border-gray-300"
            src={item.userData.image || "/default-avatar.png"}
            alt=""
          />
          <p className="font-medium">{item.userData.name}</p>
        </div>

        /* Payment */
        <p className={item.payment ? "text-green-600" : "text-gray-600"}>
          {item.payment ? "Online" : "Cash"}
        </p>

        /* Age */
        <p>{calculateAge(item.userData.dob) || "N/A"}</p>

        /* Date & Time */
        <p>
          {slotDateFormat(item.slotDate)}, {item.slotTime}
        </p>

        /* Fees */
    
```

```

        <p className="font-medium">
          {currency}
          {item.amount}
        </p>

        {/* Action */}
        {item.cancelled ? (
          <p className="text-red-600 font-semibold">Cancelled</p>
        ) : item.isCompleted ? (
          <p className="text-green-600 font-semibold">Completed</p>
        ) : (
          <div className="flex gap-2">
            <img
              onClick={() => cancelAppointment(item._id)}
              className="w-8 h-8 cursor-pointer hover:scale-110 transition"
              src={assets.cancel_icon}
              alt="Cancel"
            />
            <img
              onClick={() => completeAppointment(item._id)}
              className="w-8 h-8 cursor-pointer hover:scale-110 transition"
              src={assets.tick_icon}
              alt="Complete"
            />
          </div>
        )}
      </div>
    ))
  ) : (
    <div className="p-6 text-center text-gray-500">
      No appointments found.
    </div>
  )
</div>
</div>
);
};

export default DoctorAppointments;});

 setLoading(false);
setLectures(data.lectures);
} catch (error) {
  console.log(error);
  setLoading(false);
}
}

async function fetchLecture(id) {
  setLecLoading(true);

```

```
try {
  const { data } = await axios.get(`/${server}/api/lecture/${id}` , {
    headers: {
      token: localStorage.getItem("token"),
    },
  });

  setLecture(data.lecture);
  setLecLoading(false);
} catch (error) {
  console.log(error);
  setLecLoading(false);
}
}

const changeVideoHandler = (e) => {
  const file = e.target.files[0]; const
  reader = new FileReader();
  reader.readAsDataURL(file);
  reader.onloadend = () => {
    setVideoPrev(reader.result);
    setVideo(file);
  };
};

const submitHandler = async (e) => {
  e.preventDefault(); setBtnLoading(true);
  const myForm = new FormData();

  myForm.append("title", title);
  myForm.append("description", description);
  myForm.append("file", video);

  try {
    const { data } = await axios.post(
      `${server}/api/course/${params.id}`
      , myForm,
    {
      headers: {
        token: localStorage.getItem("token"),
      },
    }
  );

  toast.success(data.message);
  setBtnLoading(false);
  setShow(false);
  fetchLectures();
  setTitle("");
  setDescription("");
  setVideo("");
  setVideoPrev("");
} catch (error) {
```

```
        toast.error(error.response.data.message);
        setBtnLoading(false);
    }
};

const deleteHandler = async (id) => {
    if (confirm("Are you sure you want to delete this lecture")) { try
    {
        const { data } = await axios.delete(`/${server}/api/lecture/${id}`, {
            headers: {
                token: localStorage.getItem("token"),
            },
        });
    }

    toast.success(data.message);
    fetchLectures();
} catch (error) {
    toast.error(error.response.data.message);
}
}
};

useEffect(() => {
    fetchLectures();
}, []);
return (
    <>
        {loading ? (
            <Loading />
        ) : (
            <>
                <div className="lecture-page">
                    <div className="left">
                        {lecLoading ? (
                            <Loading />
                        ) : (
                            <>
                                {lecture.video ? (
                                    <>
                                        <video
                                            controlsList="nodownload noreMOTEPLAYBACK"
                                            disablePic1              disableRemotePlayback
                                            autoPlay
                                        ></video>
                                    <h1>{lecture.title}</h1>
                                    <h3>{lecture.description}</h3>
                                    </>
                                ) : (
                                    <h1>Please Select Your Lecture</h1>
                                )}
                            </>
                        )}
                    </div>
                </div>
            </>
        )})
    </>
)
```

```

        </div>
        <div className="right">
            {user && user.role === "admin" && (
                <button className="add-btn mt-3" onClick={() =>setShow(!show)}>{show ? "Close" : "Add Lecture"}</button>
            )}
        
```

```

        {show && (
            <div className="lecture-form">
                <h2>Add Lecture</h2>
                <form onSubmit={submitHandler}>
                    <label htmlFor="text">Title</label>
                    <input
                        type="text"
                        value={title}
                        onChange={(e) => setTitle(e.target.value)}
                        required
                    />
                    <label htmlFor="text">Description</label>
                    <input
                        type="text" value={description}
                        onChange={(e) => setDescription(e.target.value)}
                        required
                    />
                    <input
                        type="file" placeholder="choose
                        video"
                        onChange={changeVideoHandler}
                        required
                    />
                </form>
                {videoPrev && (
                    <video src={videoPrev} width={"300"} controls></video>
                )}
            </div>
        )
    
```

#### **14) APPOINTMENTS PAGE FOR DOCTOR :**

✓ CODE:

```
import React, { useContext, useEffect, useState } from "react";
import { DoctorContext } from "../../context/DoctorContext";
import { ApplicationContext } from "../../context/ApplicationContext";
import axios from "axios";
import { toast } from "react-toastify";

const DoctorProfile = () => {
  const { dToken, profileData, setProfileData, getProfileData, backendUrl } =
    useContext(DoctorContext);
  const { currency } = useContext(ApplicationContext);
  const [isEdit, setIsEdit] = useState(false);

  const updateProfile = async () => {
    try {
      const updateData = {
        address: profileData.address,
        fees: profileData.fees,
        available: profileData.available,
      };
      const { data } = await axios.post(
        backendUrl + "/api/doctor/update-profile",
        updateData,
        { headers: { dToken } }
      );
      if (data.success) {
        toast.success(data.message);
        setIsEdit(false);
        getProfileData();
      } else {
        toast.error(data.message);
      }
    } catch (error) {
      toast.error(error.message);
      console.log(error);
    }
  };
  useEffect(() => {
    if (dToken) {
      getProfileData();
    }
  }, [dToken]);
}

return (
  profileData &&
```

```
<div className="max-w-4xl mx-auto p-6">
  {/* Profile Card */}
  <div className="bg-white rounded-2xl shadow-xl overflow-hidden border border-gray-100">
    {/* Top Banner */}
    <div className="bg-gradient-to-r from-indigo-500 to-purple-500 p-6 flex flex-col sm:flex-row items-center gap-6">
      <img
        className="w-40 h-40 rounded-full border-4 border-white shadow-lg object-cover"
        src={profileData.image}
        alt={profileData.name}
      />
      <div className="text-center sm:text-left text-white">
        <h1 className="text-3xl font-bold">{profileData.name}</h1>
        <p className="text-lg opacity-90">
          {profileData.degree} – {profileData.speciality}
        </p>
        <span className="inline-block mt-2 px-4 py-1 bg-green-100 text-green-800 rounded-full text-sm font-medium shadow-sm">
          {profileData.experience}
        </span>
      </div>
    </div>
  </div>

  {/* Info Section */}
  <div className="p-6 space-y-6">
    {/* About */}
    <section>
      <h2 className="text-lg font-semibold text-gray-800 border-b pb-1 mb-2">
        About
      </h2>
      <p className="text-gray-600 leading-relaxed text-sm">
        {profileData.about}
      </p>
    </section>

    {/* Appointment Fee */}
    <section>
      <h2 className="text-lg font-semibold text-gray-800 border-b pb-1 mb-2">
        Appointment Fee
      </h2>
      <p className="text-blue-600 font-medium">
        {currency}{" "}
        {isEdit ? (
          <input
            type="number"
            className="border border-gray-300 rounded-lg px-2 py-1 w-28"
            onChange={(e) =>
              setProfileData((prev) => ({

```

```

        ...prev,
        fees: e.target.value,
    }})
}
value={profileData.fees}
/>
) : (
    profileData.fees
)
</p>
</section>

/* Address */
<section>
<h2 className="text-lg font-semibold text-gray-800 border-b pb-1 mb-2">
    Address
</h2>
<p className="text-gray-600 text-sm">
    {isEdit ? (
        <>
            <input
                type="text"
                className="border border-gray-300 rounded-lg px-2 py-1 w-full mb-2"
                onChange={(e) =>
                    setProfileData((prev) => ({
                        ...prev,
                        address: { ...prev.address, line1: e.target.value },
                    }))
                }
            value={profileData.address.line1}
        />
        <input
            type="text"
            className="border border-gray-300 rounded-lg px-2 py-1 w-full"
            onChange={(e) =>
                setProfileData((prev) => ({
                    ...prev,
                    address: { ...prev.address, line2: e.target.value },
                }))
            }
            value={profileData.address.line2}
        />
    ) : (
        <>
            {profileData.address.line1}
            <br />
            {profileData.address.line2}
        </>
    )}

```

```
        </p>
    </section>

    {/* Availability */}
    <section>
        <div className="flex items-center gap-2">
            <input
                onChange={() =>
                    isEdit &&
                    setProfileData((prev) => ({
                        ...prev,
                        available: !prev.available,
                    }))
                }
                checked={profileData.available}
                type="checkbox"
                id="available"
                className="w-4 h-4"
            />
            <label htmlFor="available" className="text-gray-700">
                Available
            </label>
        </div>
    </section>

    {/* Buttons */}
    <div className="flex gap-3">
        {isEdit ? (
            <button
                onClick={updateProfile}
                className="px-5 py-2 bg-green-500 hover:bg-green-600 text-white rounded-lg shadow-md transition"
            >
                Save
            </button>
        ) : (
            <button
                onClick={() => setIsEdit(true)}
                className="px-5 py-2 bg-purple-500 hover:bg-purple-600 text-white rounded-lg shadow-md transition"
            >
                Edit Profile
            </button>
        )}
    </div>
</div>
</div>
</div>
)
);
```

};

```
export default DoctorProfile;
```

## ❖ BACKEND:-

### 1) USER AND ADMIN DATA:

The screenshot shows the MongoDB Compass interface. The left sidebar has a 'Data Explorer' tab selected, showing databases and collections. The main area displays the 'mydoc.users' collection with 12 documents. A document is expanded to show its fields: \_id, name, email, password, image, gender, dob, phone, and address. The address field is shown as an object.

```
_id: ObjectId('689b33c374d8bcf0d778913e')
name: "Ritam"
email: "ritam.user@mydoc.com"
password: "$2b$10$t5vdj1RD3zKfzFbna2PDunH/GEf5ZK.T9HZFmQywe79Rnl.geyu"
image: "https://res.cloudinary.com/dogjqdjer/image/upload/v1755028886/u8oa@wtk..."
gender: "Male"
dob: "29-11-2004"
phone: "7003978731"
__v: 8
address: Object
```

```
_id: ObjectId('689caacbbcc62fb0ae40da72')
name: "Puja Mahanty"
email: "user.puja@gmail.com"
password: "$2b$10$WFAv1RK8oEGIew2XCPaj.AE76cuhyWfWa90T.Gj1kD0odsT751SK"
image: "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAPAAAADwCAYAAAA+VemSAAA..."
```

System Status: All Good

- Database - mongodb.js:

```
import mongoose from 'mongoose';
const connectDB = async () => {
  mongoose.connection.on('connected', () => console.log("Database Connected"));
  await mongoose.connect(`.${process.env.MONGO_URI}/mydoc`)
}
export default connectDB;
```

- model: usermodel.js:-

```
import mongoose from "mongoose"

const userSchema = new mongoose.Schema(
{
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  image: { type: String, default:
"data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAPAAAAdwCAYAAAA+VemSAAAACXBIWXMAABCcAAAQnAEmz
To0AAAAAXNSR0IArsc6QAAAARnQU1BAACxjwv8YQUAA5uSURBVHgB7d0JchvHFc bxN+C+iaQolmzFsaWqHMA5QXID+
wZJTmDnBLZu4BvER4hvYJ/AvoHlimPAccIMOAYAQYcI8CAYwQYcIwAA44RYMAxAgw4RoABxwgv4BgBBhwjwIBjBBhwjA
ADjhFgwDECDDhGgAHHCDDgGAEGHCPAgG0zBlfanfzRNrv05o8Ls46e08Vdut3i966babz7rMfcjFmWP8/r0TM4Q4ADpj
CenZu18sCe52FtX9wcckGUAS+fb6IwK9Tzc/kHI/96gU9H8HiLAN0Wh/WsZXZ6fnfYpkEXCT30b0sjr8jf+SdkYb4I8w
wdruAQ4AAotCdnRbUdtcJ0g74XhbkMtCr08iJhDgkBrkmv0uWV9vgsrNDeRd/z31HxtSr0kIe6H1DjQhwxVRTD0+kfq
1n+v5b/Z91KQ/x8gJVuQ5Zc6fr5PrvllyzBvYuCvLZEKtEBZ6yFIJb0mkVD4JcHQI8JSkF9zqFWANyalYryJgeAjxh6p
Ac5ME90rOkawDu8LQI8+oSg13TQoAnSKPKe8d+RpWroHvZGrLundOsngYCPAGqurtH1/dL8S5VYnUnqMaTRYDHpL6uKk
zVs6Y8Kqux5nKrGjP3enwEeAwHp8VAFYaj8QG1VrbWaFKPi5dvBGoyvz4gvONQNX61X4wbYHQEeEj6403sp317aNI02N
c8KKbtMRqa0EPQXODmIf3dSdPtJrVqHiwbhkQFHpDC++aA8E6L+sW7R4YhUYEHcNy6XIWD6dGtJm1aoMEtRqgHQwW+B+
Gt1lo6GiBkic1gCPAdrq5/RXX0ut0cHgwBvkXZ50U9dJ+YEN+PAN9AA1UabWZ0c73UJ+YW090I8DX1JA1Gm80gW0xHp4
ZbEOBrdpnXHJz9RNdVD4IAx6G5zawoChMX1psR4L5yBw2ESeFlU0tdBNgu17khbGpGZRungvoAg4PkWgJ0iuGCd6dfC",
  address: { type: Object, default: {line1: '', line2: ''}},
  gender: { type: String, default: "Not Selected" },
  dob: { type: String, default: "Not Selected" },
  phone: { type: String, default: '0000000000' },
},
{ minimize: false }
);

const userModel = mongoose.models.user || mongoose.model("user", userSchema);

export default userModel;
```

## 2) DOCTOR DATA:

The screenshot shows the MongoDB Compass interface. The left sidebar has sections like Cluster, Overview, Data Explorer (which is selected), Real Time, Cluster Metrics, Query Insights, Performance Advisor, Online Archive, Command Line Tools, and Infrastructure as Code. The top navigation bar shows Organization: Ritam's Org - 2025-0..., Project: Project 0, and Cluster: Cluster0. The main area shows a database named 'mydoc' containing collections: appointments, doctors, users, and sample\_mflix. The 'doctors' collection is selected, showing its storage size (44KB), logical data size (10.4KB), total documents (15), and index size (72KB). There are tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A search bar at the top says 'Search Namespaces'. Below the collection details, there's a section for generating natural language queries. A query result table shows 1-15 of 15 documents, with one document expanded to show its fields. At the bottom, there's a system status message 'System Status: All Good' and links to MongoDB documentation.

- model – doctorModel.js :-

```
import express from "express";
import mongoose from "mongoose";

const doctorSchema = new mongoose.Schema(
{
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  image: { type: String, required: true },
  speciality: { type: String, required: true },
  degree: { type: String, required: true },
  experience: { type: String, required: true },
  about: { type: String, required: true },
  available: { type: Boolean, default: true },
  fees: { type: Number, required: true },
  address: { type: Object, required: true },
  date: { type: Number, required: true },
  slots_booked: { type: Object, default: {} },
},
{ minimize: false }
);
const doctorModel = mongoose.models.doctor || mongoose.model("doctor", doctorSchema);

export default doctorModel;
```

## 3) APPOINTMENT DATA

The screenshot shows the MongoDB Compass interface. The left sidebar has sections like Cluster, Overview, Data Explorer (which is selected), Real Time, Cluster Metrics, Query Insights, Performance Advisor, Online Archive, Command Line Tools, and Infrastructure as Code. The top navigation bar shows Organization: Ritam's Org - 2025-0..., Project: Project 0, and Cluster: Cluster0. The main area shows a database named 'mydoc' containing collections: appointments, doctors, users, and sample\_mflix. The 'appointments' collection is selected, showing its storage size (44KB), logical data size (10.4KB), total documents (9), and index size (72KB). There are tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A search bar at the top says 'Search Namespaces'. Below the collection details, there's a section for generating natural language queries. A query result table shows 1-9 of 9 documents, with one document expanded to show its fields. At the bottom, there's a system status message 'System Status: All Good' and links to MongoDB documentation.

**● model – appointmentModel.js :-**

```
import mongoose from "mongoose";

const appointmentSchema = new mongoose.Schema({
  userId: { type: String, required: true },
  docId: { type: String, required: true },
  slotDate: { type: String, required: true },
  slotTime: { type: String, required: true },
  userData: { type: Object, required: true },
  docData: { type: Object, required: true },
  amount: { type: Number, required: true },
  date: { type: Number, required: true },
  cancelled: { type: Boolean, default: false },
  payment: { type: Boolean, default: false },
  isCompleted: { type: Boolean, default: false },
});

const appointmentModel =
  mongoose.models.appointment ||
  mongoose.model("appointment", appointmentSchema);

export default appointmentModel;
```

## **6. CONCLUSION**

The **MyDoc – Doctor Appointment Booking System** is designed to bridge the gap between patients and healthcare providers by offering a secure, efficient, and user-friendly platform for managing medical consultations. Through features such as online appointment booking, real-time slot management, medical record storage, and integrated payment gateways, the system streamlines the healthcare process for all stakeholders.

Built using the **MERN stack** and enhanced with secure authentication, encryption, and compliance with privacy standards such as **HIPAA** and **GDPR**, MyDoc ensures both performance and data security. The adoption of the **Iterative Waterfall Model** during development has allowed for a structured yet flexible approach, enabling the incorporation of user feedback and system improvements throughout the process.

By automating administrative tasks, reducing appointment delays, and enabling remote consultations, MyDoc has the potential to improve healthcare accessibility, reduce operational workload, and enhance the overall patient experience. With continuous updates and feature enhancements, it can adapt to evolving medical needs, ensuring long-term relevance and usability.

## **7. FUTURE SCOPE & FURTHER ENHANCEMENTS**

### **Future scope :-**

The MyDoc – Doctor Appointment Booking System has the potential to expand beyond its current capabilities to meet emerging healthcare needs. In the future, the system can be scaled and adapted to:

- Expand User Base: Support larger networks of doctors, clinics, and hospitals across multiple cities and countries.
- Specialized Medical Services: Include dedicated modules for dental, physiotherapy, mental health, and other specialist appointments.
- Integration with Public Health Systems: Link with government health portals for vaccination drives, public health updates, and digital health IDs.
- Global Telemedicine Services: Enable cross-border consultations with multilingual support and international payment gateways.
- Advanced Teleconsultation Features: Incorporate AI-assisted diagnosis, electronic medical records sharing, and e-prescription automation for remote consultations.
- Research and Data Analytics: Use anonymized patient data for medical research, healthcare trend analysis, and policy-making.
- Preventive Healthcare Programs: Launch wellness plans, regular health check-up reminders, and lifestyle tracking tools.

## **Further enhancement:-**

The **MyDoc – Doctor Appointment Booking System** has been developed to meet the current requirements of patients, doctors, and administrators. However, as healthcare needs evolve and technology advances, the system can be further improved to provide more comprehensive services. Potential areas for enhancement include:

- **AI-Driven Recommendations:** Incorporating artificial intelligence to suggest doctors based on patient symptoms, medical history, and availability.
- **Pharmacy and Diagnostic Lab Integration:** Enabling patients to order prescribed medicines online and book lab tests directly from the platform.
- **Multi-Language Interface:** Offering support for multiple regional and global languages to ensure accessibility for a diverse user base.
- **Insurance and Billing Integration:** Partnering with insurance providers to facilitate easy claim processing and cashless treatments.
- **Offline Functionality:** Allowing appointment booking and record management in low-connectivity areas, with data syncing once internet access is restored.
- **Advanced Data Analytics:** Providing healthcare providers and admins with detailed insights into patient trends, appointment patterns, and treatment outcomes.

## **8. BIBLIOGRAPHY**

- 1)<https://www.mongodb.com/docs>
- 2)<https://expressjs.com/>
- 3)<https://razorpay.com/docs/>
- 4)<https://webrtc.org/>
- 5)<https://react.dev>
- 6)<https://nodejs.org/en/docs>