Final Year Project Report

Doctor Appointment Booking System

**BS(IT) (Session: 2019-23)**

**Project Supervisor**

Mr. Umar Daraz

Designation  
University of Sahiwal

**Submitted by**

Muhammad Bilal

**Roll NO.**

BsIT-M2-19-42

|  |  |  |
| --- | --- | --- |
|  |  |  |

DEPARTMENT OF COMPUTER SCIENCE

Preface

This project revolves around the creation and implementation of a Doctor Appointment Booking System web application. The primary objective of this system is to streamline the process of booking appointments with doctors, benefiting both patients and healthcare professionals alike.

In the modern world, where technology has become an integral part of our lives, it is essential to leverage it for enhancing the healthcare experience. The Doctor Appointment Booking System provides a user-friendly platform where patients can browse, search, and book appointments with doctors based on their specialization, location, and availability.

The idea behind this project is to simplify the appointment booking process, making it convenient for patients to access medical services. By having an online platform, patients can easily find doctors, view their profiles, and schedule appointments at their preferred date and time.

This report presents a comprehensive documentation of the project, detailing the conceptualization, design, implementation, and testing phases. It includes insights into the system architecture, data models, user interactions, functional descriptions, and the overall testing strategy.

The Doctor Appointment Booking System not only benefits patients in accessing healthcare services but also provides doctors with a centralized platform to manage their profiles, clinics, availability, and appointments efficiently.

We hope that this report serves as a comprehensive guide to understanding the concept and implementation of the Doctor Appointment Booking System. It is our aspiration that this web application contributes positively to the healthcare sector, improving the accessibility and efficiency of medical appointments for all users.

Lastly, we express our gratitude to our mentors, peers, and all those who have supported us throughout this journey of our Final Year Project. Without their guidance and encouragement, this endeavor would not have been possible.

ACknowledgment

First of all, we thank Almighty Allah who gives us the strength and ability to think, work and deliver what we are assigned to do. Secondly, we must be grateful to our internal supervisor **Mr. Umar Daraz** who guided us in this project. We also acknowledge our teachers who guided, taught and helped us during our study period. We would also like to thank all departmental staff and university staff, who had assisted us during our stay at the university.

**UNIVERISTY OF SAHIWAL, SAHIWAL**

**University Online:** [www.uosahiwal.edu.pk](http://www.uosahiwal.edu.pk)

**DEPARTMENT OF COMPUTER SCIENCE**

**PROJECT APPROVAL FORM**

CERTIFICATE OF COMPLETION

This is to certify that the following student

Student Name: Muhammad Bilal Roll No: BSIT-M2-19-42

have successfully completed their final year project titled

Doctor Appointment Booking System

in the partial fulfillment for the requirements of the Degree of Bachelor of Computer Science & Information Technology during the academic session 2019-2023.  
  
**Signatures: Signatures:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mr. Umar Daraz Dr. Shafique Hussain

Designation Chairman  
Department of Computer Science Department of Computer Science

University of Sahiwal University of Sahiwal

Abstract

The Doctor Appointment Booking System is a web application built using the Django framework that facilitates the process of scheduling appointments between doctors and patients. The system aims to streamline the appointment booking process, providing an efficient and convenient way for patients to find and schedule appointments with doctors of their choice.

The system includes multiple entities such as doctors, patients, clinics, and administrators, each with their own specific roles and functionalities. Doctors can create profiles, manage their availability, and provide information about their specialization, experience, services, degrees, and awards. Patients can register and search for doctors based on various criteria such as specialization, location, and availability. They can book appointments, manage their profiles, and engage in chat communication with doctors. Administrators have control over system management, including user management and clinic management.

The system incorporates features like appointment booking, appointment management, profile management, clinic management, and chat communication between doctors and patients. It ensures data security, user authentication, and authorization to maintain confidentiality and privacy.

By implementing the Doctor Appointment Booking System, the aim is to improve the overall efficiency and accessibility of healthcare services. It provides a user-friendly platform for patients to find suitable doctors and book appointments, reducing waiting times and enhancing patient satisfaction. The system also offers convenience for doctors by automating the appointment scheduling process and allowing efficient management of their schedules.

Overall, the Doctor Appointment Booking System enhances the doctor-patient interaction, streamlines the appointment booking process, and contributes to a more efficient healthcare ecosystem.

**REPORT PLAN**

**Notes:**

* 3 properly printed and bound reportS are to be submitted, 1 original and 2 copies (2 for university and 1 for internal EXAMINER)
* This is only template…modify main body contents as per your project requirements.
* The report can have minimum 10000 and max 15000 words or 80-100 pages
* Preface, Synopsis, Introduction to group members should be brief not more than 2 pages each
* Sequence of chapters must be followed
* “Figures” and “Tables” must be shown as Figure 1.1, Figure 1.2, Figure 2.1 and Table 1.1, Table 2.1 so on. Use these figure numbers [page number] where ever you want to reference it. Avoid redundancy in figures
* Level 3 headings can be increased as per requirement (single tab)
* Ensure that the Formatting style given is followed.
* You can Highlight important features if you like but don’t over do it
* Please ensure that page numbering must be correct.
* The students should follow the appropriate format as provided by the department.
* Throughout this report template you may find comments for your guidance. These guidelines should not be included in your final year report.
* One blank page should be left at the end of the bound report for remarks
* CD cover and layout is also provided at the end of this document. This CD will also be submitted on the day of your project submission
* Every single page from MAIN BODY and onwards should have the following Header/Footer format:

**Header:**

***Chapter # Chapter Name***

**Footer:**

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***University of Sahiwal Page #.***

**SUMMARY OF REPORT TEMPLATE**

* Headings as you can see in report consist of many levels. When writing the report any heading at any level, if consist of sub heading, then you have to write the overview of what will be discussed in those sub headings.
* The report must refer the project in past tense and not continuous tense as the project is complete.
* Please see that the headings should be specific to your project. E.g. 4.1 heading is general, replace the subsystem ‘n’ by the name of your specific sub system
* Chapter 1. Introduction: under this heading you have to give overview of what will be discussed in 1.1, 1.2, 1.3 so on. it will be introduction to report
* If you omit any section, you have to update the number accordingly
* The subsequent sub headings of chapter 1 must contain introduction to project. In 1.1 you have to write the aim of your project and what objectives you mentioned in your proposal. In 1.2 give an overview of your project describing the inputs and outputs of your system. You should target general computer oriented audience and therefore must not contain any low level detail. In 1.3 discuss what the domains this system is target for are and what its market prospect is. In 1.4 you have to discuss history of your project. Any prior work done on the subject. The work that you got inspiration from so on. 1.5 consists of the tools and technology used to build your project.
* Chapter 2 is all about User end, you have to discuss who are the intend users of the system, use cases (description as well as diagrams). How users are going to use the project and usage specifications of your system. E.g. any particular Operating System this system will run on. It also consists of limitations of your project e.g. any particular setup of environment needed to run the system
* Chapter 3 main heading will only discuss the main theme of this section. The heading should mention what each underlying subsection will contain. This chapter is all about functionality of the system and type of information that flows through the system starting right from the input information leading up to output information. All the main transformation of input data will be mentioned in this section. 3.1.1 Will actually discuss the architecture of your system. 3.1 will only refer to the up coming sub sections. The Architecture of system must be represented in system diagram along with the description. The architecture must clearly mention both in writing and diagram, the subsystem /modules of your system. Note that subsystems are presented in this section as black box. Discuss inputs, outputs and processing of system in general and interconnectivity between different sub systems. 3.3.1 and 3.3.2 contains details of system interfaces to other systems if any and other machines if any.
* Chapter 4 discusses the subsystems presented in 3.1.2 in detail. Chapter 4 main heading will again discuss what sub sections this chapter contains. 4.1 will again discuss what 4.1.1, 4.1.2 so on contains. 4.1 must be written for every sub system i.e. if there are 2 sub systems A and B then 4.1 will discuss A and 4.2 discusses B. Numbering should be altered accordingly. 4.1.2 discusses both in description and suitable diagram the flow of information of sub system/ modules under discussion. E.g. DFD level 1. 4.1.3 will present only the components of sub system under discussion. 4.1.4is like 4.1 just that it discusses components of each module in detail and repeated for every component. Please note that components are the core level details, do not mix it with sub system.
* Chapter 5 discusses behavior of the system. Depicted by event diagram. The different states the system can be found, and which important events shifts the state of system e.g. system is in idle state and on certain kind of event the system changes its state to running
* Chapter 6 is applicable if you built a prototype of your actual system or you simulated your system before you actually implemented it
* Chapter 7 discusses your estimations that you proposed earlier and the actual out comes after completion of projects. You have to give details of the estimation models you used and if you have used more then one model you have to give details for every model.
* Chapter 8 is all about testing you performed, test plan you developed, test cases you developed, testing methods you applied. Certain kind of testing is mandatory for certain projects. You have to analyze which best suits your requirement

PREFACE

ACKNOWLEDGEMENT

INTRODUCTION TO GROUP MEMBERS

CERTIFICATE OF COMPLETION BY PROJECT ADVISOR

ABSTRACT

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

*All the headings above must be included in Table of Contents (TOC) according to correct page numbers. Provided below is an exemplary TOC.*

*Provided After TOC is MAIN-BODY of Report. Select the heading or text and write your material in place of it. Don’t change the style and format*

*Some headings are generic and can be molded according to your heading name*

*Try to begin the chapter with new page. See that all chapters get mentioned in the header*

Table of COntents

[Preface ……………………………………………………………II](#_Toc140903370)

[Acknowledgment III](#_Toc140903371)

[CERTIFICATE OF COMPLETION IV](#_Toc140903372)

[Abstract V](#_Toc140903377)

[Table of Contents IX](#_Toc140903386)

[LIST OF FIGURES XII](#_Toc140903388)

[LIST OF TABLES XIII](#_Toc140903389)

[1 Introduction…… ……15](#_Toc140903390)

[1.1 Goals and objectives 15](#_Toc140903392)

[1.2 System statement of scope 15](#_Toc140903393)

[1.3 System context 16](#_Toc140903394)

[1.4 Theoretical Background (of project) 16](#_Toc140903395)

[1.5 Technology & Tools/hardware components 17](#_Toc140903396)

[1.1.1 Front-end Technologies: 17](#_Toc140903397)

[1.1.2 Back-end Technology: 18](#_Toc140903398)

[1.1.3 Database: 18](#_Toc140903399)

[2 Usage Scenario / User Interaction 21](#_Toc140903400)

[2.1 User profiles 21](#_Toc140903404)

[2.1.1 Patient: 21](#_Toc140903405)

[2.1.2 Doctor: 22](#_Toc140903406)

[2.1.3 Admin: 24](#_Toc140903407)

[2.2 Use-cases 25](#_Toc140903408)

[2.3 Special usage considerations 35](#_Toc140903409)

[3 Functional and Data Description 37](#_Toc140903410)

[3.1 System Architecture 37](#_Toc140903411)

[3.1.1 Architecture model 37](#_Toc140903412)

[3.1.2 Subsystem/modules overview 38](#_Toc140903413)

[3.2 Data Description 39](#_Toc140903414)

[3.2.1 Major data objects 39](#_Toc140903415)

[3.3 System level data model 42](#_Toc140903416)

[3.4 System Interface Description 43](#_Toc140903417)

[3.4.1 External machine interfaces 43](#_Toc140903418)

[3.4.2 External system interfaces 44](#_Toc140903419)

[4 Subsystem/module Description 46](#_Toc140903420)

[4.1 Description for Subsystem ‘n’ 46](#_Toc140903421)

[4.1.1 Subsystem scope 46](#_Toc140903422)

[4.1.2 Subsystem flow diagram/ desired UML diagram 46](#_Toc140903423)

[4.1.3 Subsystem ‘n’ components 47](#_Toc140903424)

[4.1.4 Description for sub system ‘n’ Component ‘k’ 47](#_Toc140903425)

[4.1.5 Design constraints 49](#_Toc140903426)

[5 Behavioral Model and Description 51](#_Toc140903427)

[5.1 Description for system behavior 51](#_Toc140903428)

[5.1.1 Events/interrupts 51](#_Toc140903429)

[5.1.2 States 52](#_Toc140903430)

[5.2 State Transition Diagrams 53](#_Toc140903431)

[5.3 Control specification 54](#_Toc140903432)

[6 System Prototype Modeling and Simulation Results 56](#_Toc140903433)

[6.1 Description of system modeling approach (if used) 56](#_Toc140903434)

[6.2 Simulation results 56](#_Toc140903435)

[6.3 Special performance issues 56](#_Toc140903436)

[6.4 Prototyping requirements 57](#_Toc140903437)

[7 System Estimates and Actual Outcome 59](#_Toc140903438)

[7.1 Historical data used for estimates 59](#_Toc140903439)

[7.2 Estimation techniques applied and results 59](#_Toc140903440)

[7.2.1 Estimation technique ‘m’ 59](#_Toc140903441)

[7.2.2 Estimate for technique ‘m’ 60](#_Toc140903442)

[7.3 Actual Results and Deviation from Estimates 60](#_Toc140903443)

[7.4 System Resources (Required and Used) 60](#_Toc140903444)

[7.4.1 System Resources Required 60](#_Toc140903445)

[7.4.2 System Resources Used 60](#_Toc140903446)

[8 Test Plan 63](#_Toc140903447)

[8.1 System Test and Procedure 63](#_Toc140903448)

[8.2 Testing strategy 63](#_Toc140903449)

[8.4.1 Unit testing 63](#_Toc140903450)

[8.4.2 Integration testing 65](#_Toc140903451)

[8.4.3 Validation testing 65](#_Toc140903452)

[8.4.4 High-order testing (a.k.a. System Testing) 66](#_Toc140903453)

[8.4.5 Stress testing 67](#_Toc140903454)

[9 Future Enhancements and Recommendations 69](#_Toc140903455)

[9.1 Payment Gateway: 69](#_Toc140903456)

[10 Conclusion / Summary 71](#_Toc140903458)

[11 *Reference* 72](#_Toc140903459)

[12 SNap shoots of the system 73](#_Toc140903460)

[12.1 Login Page 73](#_Toc140903461)

[12.2 Forgot Password 74](#_Toc140903462)

[12.3 Patient Dashboard 74](#_Toc140903463)

[12.4 Patient Dashboard 2 75](#_Toc140903464)

[12.5 Patient Profile 75](#_Toc140903465)

[12.6 Search Doctor 76](#_Toc140903466)

[12.7 View Doctor 76](#_Toc140903467)

[12.8 Booking Timing Schedule 77](#_Toc140903468)

[12.9 Doctor Dashboard 77](#_Toc140903469)

[12.10 Doctor Reviews 78](#_Toc140903470)

[12.11 Admin Dashboard 79](#_Toc140903471)

[12.12 Admin Profile 79](#_Toc140903472)

[12.13 Admin Appointment 80](#_Toc140903473)

[12.14 Admin Specialization 80](#_Toc140903474)

[12.15 Admin Doctor 81](#_Toc140903475)

[12.16 Admin Patient List 81](#_Toc140903476)

[12.17 Admin Reviews 82](#_Toc140903477)

[12.18 Admin Transaction 82](#_Toc140903478)

[12.19 Chat Page 83](#_Toc140903479)

LIST OF FIGURES

Figure 1.1: Flowchart for Traditional v/s Online Doctor Appointment system 17

Figure 2.1: Admin use case 26

Figure 2.2: Doctor use case 29

Figure 2.3: Patient use case 32

Figure 2.4: Combine use case 34

Figure 3.1: Architecture model 38

Figure 3.2: Admin Subsystem 39

Figure 3.3: ERD Diagram 42

Figure 3.3: Class Diagram 43

Figure 4.1: Subsystem Module 47

Figure 5.1: State Transition Diagram 53

LIST OF tABLES

Table 7.1: Estimated Cost 64

Table 8.1: Test Cases 49

Table 8.2: Security Test 68

Table 8.3: Stress Test 71

|  |
| --- |
| Chapter # 1 Introduction |

1. Introduction

The Doctor Appointment Booking System web application is a platform that enables patients to book appointments with doctors online. The application aims to streamline the appointment booking process and provide a convenient and efficient way for patients to schedule their medical consultations. This document serves as the complete project documentation for the Doctor Appointment Booking System web application.



## Goals and objectives

The goals and objectives of the Doctor Appointment Booking System web application are as follows:

**Goal 1**: Provide a user-friendly interface for patients to search for doctors and book appointments.

**Goal 2**: Enable doctors to manage their profiles, availability schedule, and appointments.

**Goal 3**: Facilitate efficient communication between doctors and patients through a chat feature.

**Objective 1**: Develop a responsive and visually appealing web frontend using the Doctor Appointment Booking System.

**Objective 2**: Implement user authentication and role-based access control to differentiate between doctors, patients, and administrators.

**Objective 3**: Create data models to represent entities such as users, doctors, clinics, appointments, and chat messages.

**Objective 4**: Design and implement the necessary subsystems and modules to support the functionalities of the application.

**Objective 5**: Conduct comprehensive testing to ensure the reliability, security, and performance of the system.

**Objective 6**: Provide documentation to guide users, developers, and administrators in understanding and utilizing the Doctor Appointment Booking System web application

## System statement of scope

The Doctor Appointment Booking System web application allows patients to browse and search for doctors based on various criteria such as specialization, location, and availability. Patients can view detailed information about doctors, including their profile, clinic details, services offered, and availability schedule. They can then book appointments with their chosen doctor at a convenient date and time. Doctors have access to a dedicated dashboard where they can manage their profiles, availability schedule, and appointments. Administrators can oversee the entire system and manage user accounts, doctor profiles, clinics, and appointments.

## System context

The Doctor Appointment Booking System web application is a standalone system that operates within the context of a web environment. It interfaces with web browsers used by patients, doctors, and administrators to access and interact with the application. The system stores and retrieves data from a relational database management system (DBMS) to manage user accounts, doctor profiles, clinics, appointments, and chat messages. The application utilizes the Django framework for web development, incorporating various libraries and tools to enhance functionality and user experience.

## Theoretical Background (of project)

This project is manly focus to investigate the efficacy of the web-based online booking system for outpatients. To deal with this problem an online booking system will be very necessary. Online booking system is accessed on internet by using devices which have internet connection at any time. Doctor Appointment booking system is a convenient and an efficient way to deal with this problem. The Doctor Appointment booking system will enable people access details of hospital and other services online and also do the booking online. The system will have all the necessary information about doctors on one single website which accessible by anyone who wants information or online booking

Doctor Appointment booking system is used to manage access to service providers. Many factors affect the performance of appointment systems which include arrival and service time variability, patient and provider preferences, available information technology and the experience level of the scheduling staff. Thus a proper scheduling system has to develop by considering all these factors which will increase patient satisfaction, which in turn increases profit. An Doctor Appointment booking system allows individuals to conveniently and securely book their appointments online. Compared to the usual queuing method, the web-based booking system could significantly increase patient's satisfaction with registration and reduce total waiting time effectively.

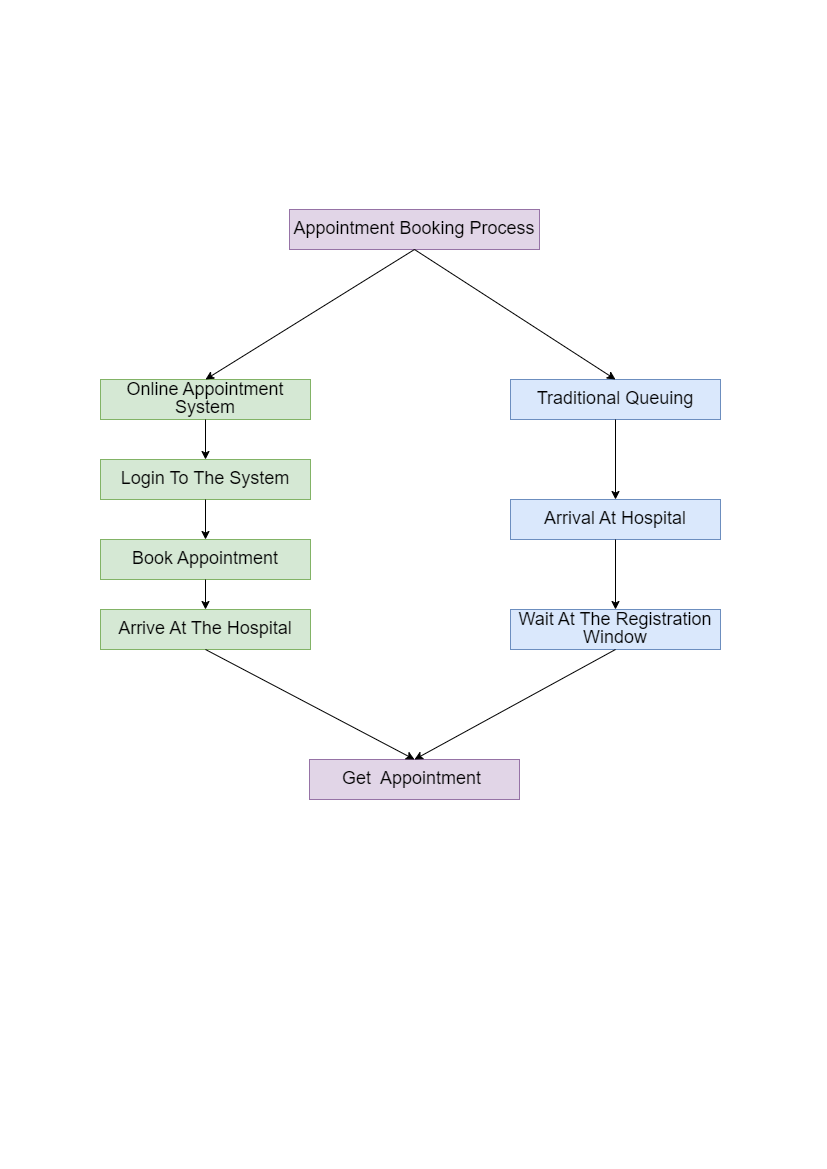


Figure:1.1 Flowchart for Traditional v/s Online Doctor Appointment system

## Technology & Tools/hardware components(used in the Project)

### Front-end Technologies:

HTML**:**

The standard markup language for creating the structure and content of web pages.

CSS**:**

Used for styling and formatting the visual presentation of web pages.

Bootstrap**:**

A popular CSS framework that provides pre-designed components and a responsive grid system, enabling faster and consistent front-end development.

JavaScript**:**

A versatile programming language used for adding interactivity and dynamic functionality to web pages.

Jquery:

A lightweight JavaScript library that simplifies DOM manipulation, event handling, and Ninja template, streamlining front-end development tasks.

The chosen front-end technologies provide a solid foundation for building a responsive and visually appealing user interface, enhancing the user experience and facilitating efficient interaction within the Doctor Appointment Booking System platform.

### Back-end Technology:

Python Django**:**

A robust and widely adopted Python web framework known for its elegant syntax, MVC (Model-View-Controller) architecture, and extensive ecosystem of libraries and packages. Django offers features like routing, database management, authentication, and caching, making it an ideal choice for developing scalable and maintainable web applications.

Django simplifies the development process by providing a structured framework, enabling faster development, and promoting code reusability. Its rich feature set and active community support make it well-suited for building the complex functionality required by Doctor Appointment Booking System.

Git: Version control system for managing source code.

### Database:

**MySQL:**

An open-source relational database management system (RDBMS) known for its performance, reliability, and ease of use. MySQL provides robust data storage and retrieval capabilities, making it a suitable choice for managing user data, project details, and other relational information within Doctor Appointment Booking System.

MySQL's widespread adoption, compatibility with Python Django, and support for essential database operations ensure efficient data management and scalability for the platform.

These technology choices are based on their popularity, extensive community support, and suitability for web application development. By leveraging these technologies, the Doctor Appointment Booking System project aims to deliver a stable, scalable, and feature-rich online platform for streamlined work.

Django Templates: Used for rendering dynamic HTML pages.

Web browsers: Users access the application through web browsers, such as Google Chrome, Mozilla Firefox, or Safari.

Deployment Platform: A server or hosting provider to deploy the web application.

|  |
| --- |
| Chapter # 2 **Usage Scenario / User Interaction** |

1. Usage Scenario / User Interaction

This section provides a usage scenario for the system. It organized information collected during requirements elicitation into use-cases.



## User profiles

The Doctor Appointment Booking System web application supports the following user profiles:

### Patient:

Patient who are seeking medical appointments with doctors,

Register:

The patient can create a new account by providing their necessary details like name, email, contact number, and password. Upon successful registration, the patient's information is stored securely in the database.

Create Profile:

After registering, the patient can complete their profile by adding additional information like age, gender, address, medical history, and any allergies they may have. A complete profile helps the system to provide personalized healthcare services.

Login:

Once registered, the patient can log in using their email/username and password.

Upon successful login, the system grants access to the patient's account and personal information.

Modify Profile:

The patient can update their profile information, such as contact details, medical history, or address. These modifications will be saved in the database for future reference.

View Doctor Profile:

The patient can browse through a list of doctors registered in the system.

Each doctor profile contains relevant information such as qualifications, specialties, availability, and patient reviews.

Appointment Booking:

The patient can schedule an appointment with their preferred doctor based on the doctor's availability. They can select a suitable date and time for the appointment.

Manage Appointment:

The patient can view, reschedule, or cancel their upcoming appointments.

The system notifies both the patient and the doctor about any changes to the appointment.

View Transaction:

The patient can access their transaction history, which includes details of payments made for appointments, prescriptions, or any other services.

Send Review:

After the appointment, the patient can provide feedback and rating about the doctor's service. These reviews will be visible to other patients looking to book appointments with the same doctor.

Chat with Doctor after Accepting the Appointment:

Once the doctor accepts the appointment, the patient can communicate with the doctor through a secure messaging platform within the system.

The chat feature allows the patient to discuss any concerns or queries before the actual appointment.

Contact with Admin:

The patient can contact the system administrator for any issues, complaints, or queries that cannot be resolved through the regular functionalities.

Change Password:

The patient can change their account password through the system.

This adds an extra layer of security to the patient's account.

Forget Password:

In case the patient forgets their password, they can use the "Forget Password" option.

The system will send a password reset link to their registered email, allowing them to set a new password.

Logout User:

The patient can log out from their account when they finish using the system.

Logging out ensures the security of their information and prevents unauthorized access.

### Doctor:

Register:

The scenario begins with a user (let's call them "User") who wants to use the system. The User initiates the registration process by providing necessary information like name, email, and password.

The system verifies the provided information and creates a new account for the User.

The User is now registered and can log in to access further functionalities.

Create Profile:

After registration, the User can create their profile, adding details like personal information, contact information, and any relevant professional information (if applicable).

This step helps in building a comprehensive profile for the User, such as an admin or a healthcare professional.

Login:

The User can log in using their registered email and password to access their account and other functionalities.

Modify Profile:

Once logged in, the User has the option to modify their profile information. They can update personal details, contact information, or any professional credentials that might change over time.

View Patient Profile:

As an admin or healthcare professional, the User can view patient profiles stored in the system.

Patient profiles may contain medical history, previous appointments, allergies, and other relevant information.

Confirm Appointment:

Patients can request appointments through the system.

As an admin or healthcare professional, the User can confirm or reject these appointment requests based on availability and other factors.

Manage Appointment:

After confirming appointments, the User can manage the schedule, including rescheduling or canceling appointments when needed.

Patients and the User receive notifications about any changes to the appointments.

Add Payments:

The User can handle the payment process for appointments and other services provided. Patients can make payments through the system, and the User can keep track of payment status and records.

View My Patients:

The User, particularly healthcare professionals, can view a list of their assigned patients or patients they have treated. This feature allows the User to have easy access to patient information and history.

View Review:

The system may allow patients to leave reviews or feedback after an appointment or service. The User can view these reviews to assess the quality of service and make improvements if necessary.

Chat with Patient after Accepting the Appointment:

After accepting an appointment, the User can initiate a chat or communication channel with the patient through the system.

This facilitates easy communication and follow-ups before or after the appointment.

Contact with Admin:

If there are any issues or concerns, the User can use the system to contact the admin for assistance or support.

Change Password:

The User can change their account password through the system's settings for security purposes.

Forget Password:

In case the User forgets their password, there should be a password recovery mechanism. This may involve email verification or security questions to reset the password.

Logout User:

The User can log out of their account once they have completed their tasks or when they want to secure their account.

### Admin:

Responsible of Description:

The Medical Appointment Management System is a web application designed to facilitate efficient scheduling and management of appointments between patients and doctors. The system has an administrative interface that allows the admin to perform various tasks related to user management, appointment management, review management, and system configuration.

**Admin**: The system administrator responsible for managing the overall functioning of the Medical Appointment Management System.

Login:

The admin can log in to the system using valid credentials (username and password).

Precondition: The admin must have a registered account in the system.

Manage Profile:

The admin can view and edit their profile information.

This includes details like name, email, contact information, etc.

View Doctor Profile:

The admin can view the profiles of registered doctors.

The profile may contain information about the doctor's name, specialization, contact details, etc.

Appointments Accept/Reject:

The admin can view incoming appointment requests from patients.

The admin can either accept or reject appointment requests based on doctor availability or other criteria.

Manage Appointments:

The admin can view and manage the existing appointments.

This includes scheduling new appointments, rescheduling existing ones, and canceling appointments if necessary.

View Transactions:

The admin can view the transaction history related to appointment fees, payments, etc.

Manage Reviews:

The admin can view and manage patient reviews and ratings for doctors.

They can delete inappropriate reviews or respond to patient feedback.

Add Specialization:

The admin can add new specializations to the system.

This is useful when new types of doctors are registered with unique specializations.

Manage Specialization:

The admin can view and edit existing specializations.

They can update or remove specializations if needed.

Manage Contact:

The admin can manage the contact information displayed on the website or app.

This includes updating phone numbers, emails, addresses, etc.

Change Password:

The admin can change their login password for security purposes.

Forget Password:

The admin can reset their password in case they forget it.

The system will provide a password recovery mechanism like email verification.

Logout:

The admin can log out of the system, ending the current session.

## Use-cases

The Doctor Appointment Booking System web application facilitates the following use cases:

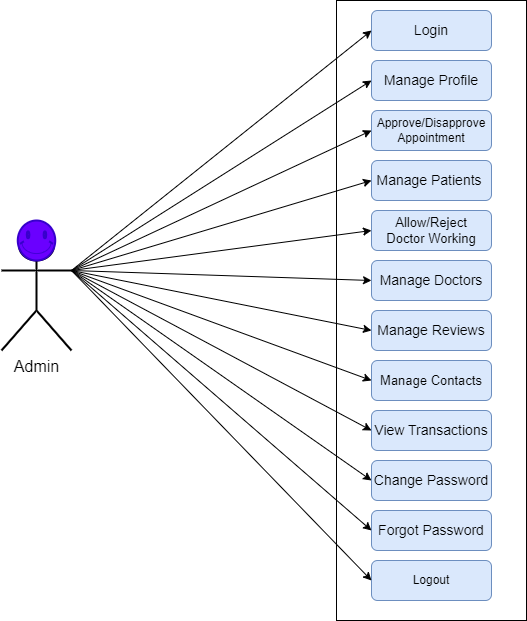


Figure: 2.1 Admin use case

## Admin:

Login as Admin

* Description: This use case enables admin to register or log in to their accounts.
* Actors: Admin
* Pre-conditions: The admin is accessing the system.
* Post-conditions: The admin is logged in to their account.

Manage Specialization

* Description: This use case enables the admin to add a new specialization to the platform.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: A new specialization is added to the system.
* Post-conditions: A existing specialization is updated in the system.
* Post-conditions: A specialization is deleted the system.

View All Patient

* Description: This use case allows the admin to view all registered patients within the platform.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The admin can see a list of all registered patients.

View All Doctor

* Description: This use case allows the admin to view all registered doctors within the platform.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The admin can see a list of all registered doctors.

Manage Doctor Working

* Description: This use case allows the admin to approve or reject the doctor profile within the platform.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The admin can see and approve or reject the doctor profile.

Accept or Reject Appointment

* Description: This use case allows the admin to accept or reject the patient appointment within the platform.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The admin can see and accept or reject the patient appointment.

User Contact

* Description: This use case allows the admin to receive and respond to user inquiries and messages.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The admin can communicate with users and provide necessary assistance.

Manage reviews

* Description: This use case allows the admin to view and react on the patient reviews.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The admin can view and react on the patient reviews about doctor.

View Payment Transaction

* Description: This use case allows the admin to view transactions.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The admin can view list of transactions.

Update Profile

* Description: This use case allows the admin to update here profile.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The admin can update profile.

Change Password

* Description: This use case enables the admin to change their password for enhanced account security.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The admin's password is changed successfully.

Logout

* Description: This use case allows the admin to securely log out of their account.
* Actors: Admin
* Pre-conditions: The admin is logged in to the system.

**Post-conditions**: The admin is successfully logged out of their account.

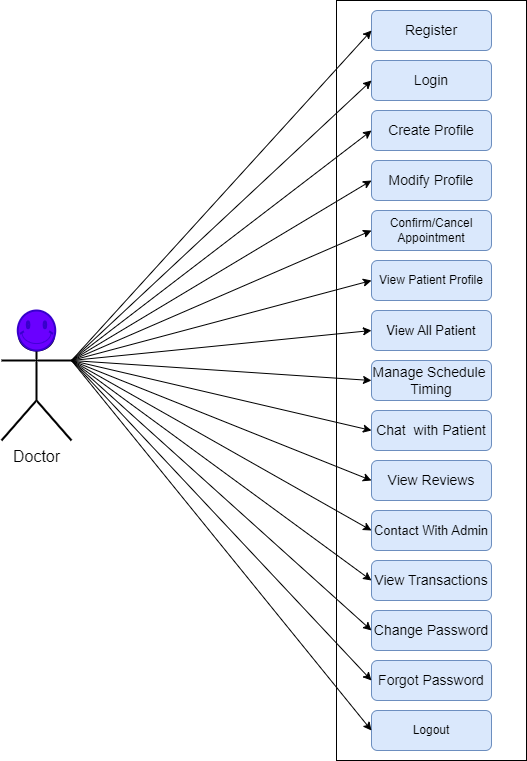


Figure: 2.2 Doctor use cases

## Doctor:

Register/Login as Doctor

* Description: This use case enables doctor to register or log in to their accounts.
* Actors: Doctor
* Pre-conditions: The doctor is accessing the system.
* Post-conditions: The doctor is registered or logged in to their account.

Create Profile

* Description: This use case enables the doctor to add details about profile to the platform.
* Actors: Doctor
* Pre-conditions: The doctor is logged in to the system.
* Post-conditions: A new profile is added to the system.

View My Patient

* Description: This use case allows the doctor to view all patients related to doctor within the platform.
* Actors: Doctor
* Pre-conditions: The doctor is logged in to the system.
* Post-conditions: The doctor can see a list of all view all patients related to doctor.

Confirm or Cancel Appointment

* Description: This use case allows the doctor to confirm or cancel the patient appointment within the platform.
* Actors: Doctor
* Pre-conditions: The doctor is logged in to the system.
* Post-conditions: The doctor can see and confirm or cancel the patient appointment.

Chat Patient

* Description: This use case allows the doctor to receive and send message and inquiries to Patient within the platform.
* Actors: Doctor
* Pre-conditions: The doctor is logged in to the system.
* Post-conditions: The doctor can receive or send message and inquiries to Patient in chat.

View Reviews

* Description: This use case allows the doctor to view the patient reviews.
* Actors: Doctor
* Pre-conditions: The doctor is logged in to the system.
* Post-conditions: The doctor can view the patient reviews.

Add Payment

* Description: This use case allows the doctor to add payment received by patient Manual.
* Actors: Doctor
* Pre-conditions: The doctor is logged in to the system.
* Post-conditions: The doctor can add payment received by patient Manual.

View Payment Transaction

* Description: This use case allows the patient to view transactions.
* Actors: Patient
* Pre-conditions: The patient is logged in to the system.
* Post-conditions: The patient can view list of transactions.

Update Profile

* Description: This use case allows the doctor to update here profile.
* Actors: Doctor
* Pre-conditions: The doctor is logged in to the system.
* Post-conditions: The doctor can update profile.

Change Password

* Description: This use case enables the doctor to change their password for enhanced account security.
* Actors: Doctor
* Pre-conditions: The admin is logged in to the system.
* Post-conditions: The doctor password is changed successfully.

Logout

* Description: This use case allows the admin to securely log out of their account.
* Actors: Doctor
* Pre-conditions: The doctor is logged in to the system.

**Post-conditions**: The doctor is successfully logged out of their account.

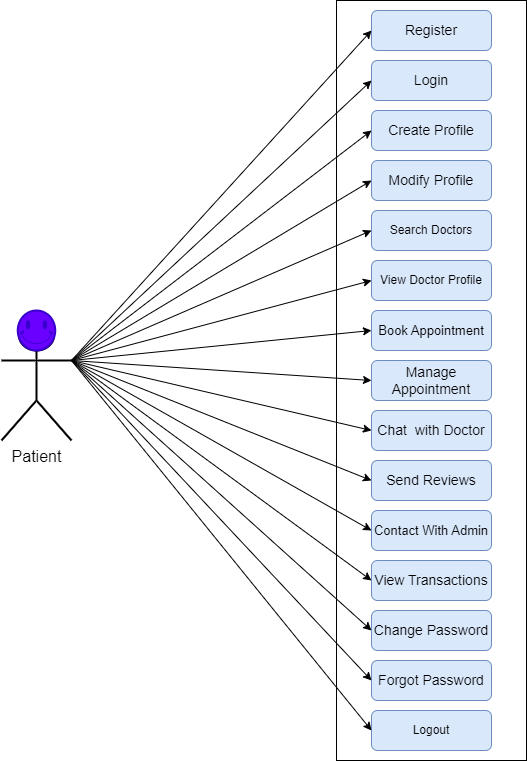


Figure: 2.3 Patient use case.

## Patient:

Register/Login as Patient

* Description: This use case enables patient to register or log in to their accounts.
* Actors: Patient
* Pre-conditions: The patient is accessing the system.
* Post-conditions: The patient is registered or logged in to their account.

Create Profile

* Description: This use case enables the patient to add details about profile to the platform.
* Actors: Patient
* Pre-conditions: The patient is logged in to the system.
* Post-conditions: A new profile is added to the system.

Book or Cancel Appointment

* Description: This use case allows the patient to book or cancel the appointment with the doctor.
* Actors: Doctor
* Pre-conditions: The doctor is logged in to the system.
* Post-conditions: The doctor can see and confirm or cancel the patient appointment.

Chat Doctor

* Description: This use case allows the patient to receive and send message and inquiries to doctor within the platform.
* Actors: Patient
* Pre-conditions: The patient is logged in to the system.
* Post-conditions: The patient can receive or send message and inquiries to doctor in chat.

Send Reviews

* Description: This use case allows the patient to send reviews.
* Actors: Patient
* Pre-conditions: The patient is logged in to the system.
* Post-conditions: The patient can send the reviews.

Update Profile

* Description: This use case allows the patient to update here profile.
* Actors: Patient
* Pre-conditions: The patient is logged in to the system.
* Post-conditions: The patient can update profile.

Change Password

* Description: This use case enables the patient to change their password for enhanced account security.
* Actors: Patient
* Pre-conditions: The patient is logged in to the system.
* Post-conditions: The patient password is changed successfully.

Logout

* Description: This use case allows the patient to securely log out of their account.
* Actors: Patient
* Pre-conditions: The patient is logged in to the system.
* Post-conditions: The patient is successfully logged out of their account

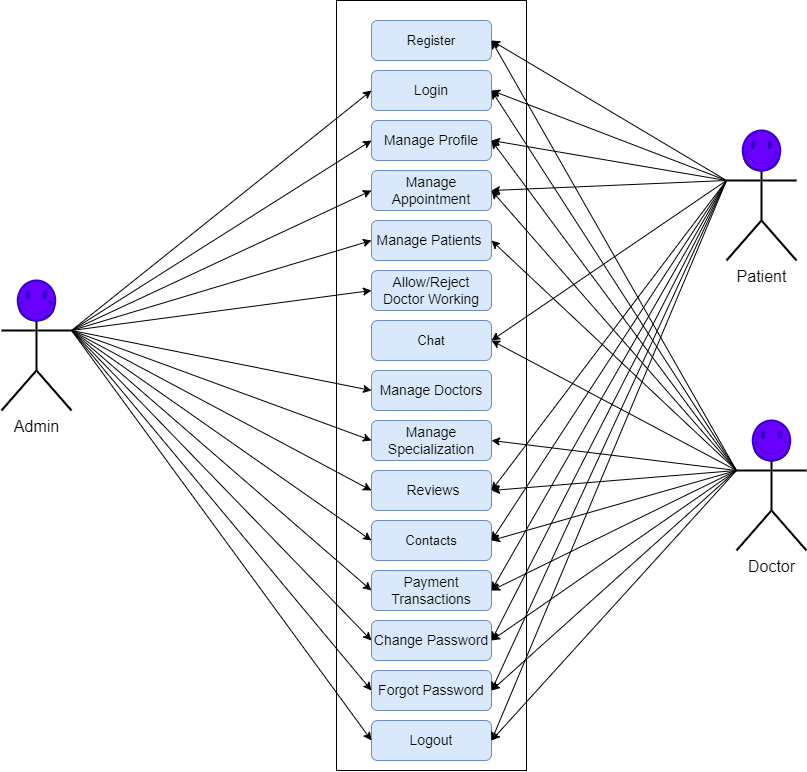


Figure: 2.4 Patient use case.

## Special usage considerations

The Doctor Appointment Booking System web application addresses the following special usage considerations:

**Security**: The application should ensure the privacy and confidentiality of user information, including personal details and medical history.

**Accessibility**: The application should be designed to accommodate users with disabilities and provide a seamless experience for all users.

**Scalability**: The system should be scalable to handle a large number of users, appointments, and chat messages concurrently.

**User-Friendly Interface**: The application should have an intuitive and user-friendly interface to facilitate easy navigation and interaction for all users.

|  |
| --- |
| Chapter # 3 **Functional and Data Description** |

1. Functional and Data Description

This section will provide an overview of the system model, its constituent’s parts and major data objects and their relation ship that flows in the system. Data object is any data that is being transformed by our system. It can be different forms of image or signal or software data or electronic signal etc. Only brief overview of what this section will contain should be mentioned in this heading

## System Architecture

The Doctor Appointment Booking System web application follows a three-tier architecture, consisting of the following components:

**Presentation Layer**: Handles user interaction and displays the user interface using HTML, CSS, and JavaScript. The Doctor Appointment Booking System is utilized for the frontend implementation.

**Application Layer**: Contains the business logic and processes user requests. It interacts with the presentation layer for receiving and returning data.

**Data Layer**: Manages data persistence and storage. It interacts with the application layer to perform database operations.

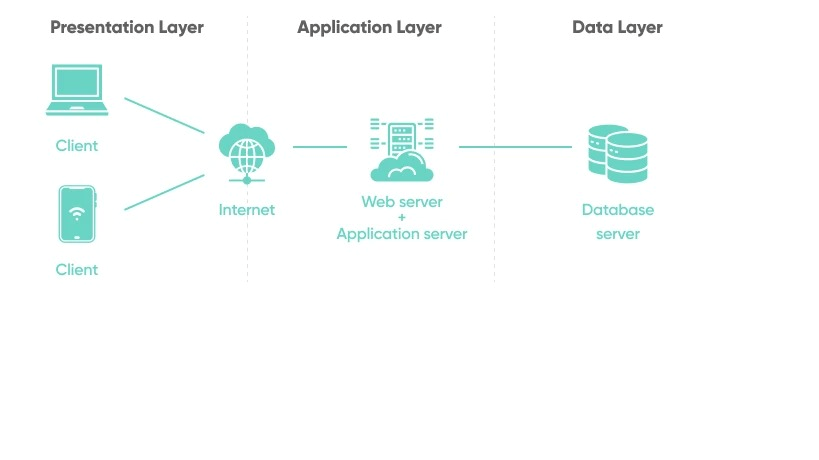
### Architecture model

The architecture model used in the Doctor Appointment Booking System web application is the Model-View-Controller (MVC) pattern. This pattern separates the application into three interconnected components:

**Model**: Represents the data and business logic of the application. It includes the data models for user accounts, doctor profiles, clinics, appointments, and chat messages.

**View**: Handles the presentation layer and user interface. It includes HTML templates and frontend components for rendering data.

**Controller**: Manages the flow of data and coordinates interactions between the model and view components. It handles user requests, updates the model, and renders the appropriate view.

 Figure: 3.1 Architecture model

### Subsystem/modules overview

The Doctor Appointment Booking System web application consists of the following subsystems/modules:

**User Authentication**: Handles user registration, login, and session management.

**User Management**: Allows administrators to manage user accounts, including adding, updating, or deactivating user profiles.

**Doctor Profile Management**: Enables doctors to create and manage their profiles, including personal information, clinic details, specialization, services offered, experience, degrees, awards, and availability schedule.

**Clinic Management**: Supports the management of clinics associated with doctors, including clinic information and images.

**Appointment Booking**: Facilitates the booking, approval, rescheduling, and cancellation of appointments by patients and doctors.

**Chat**: Provides a chat feature for doctors and patients to communicate with each other securely and efficiently.

**Feedback Management:**

The Rating and Review Management module allows buyers and sellers to rate and review each other's work. Patients can send feedback about doctorsdegrres, communication, and professionalism. Doctor can also reply patients on their timeliness, responsiveness, and payment. This allows Patient and Doctors to make informed decisions about who to work with in the future.

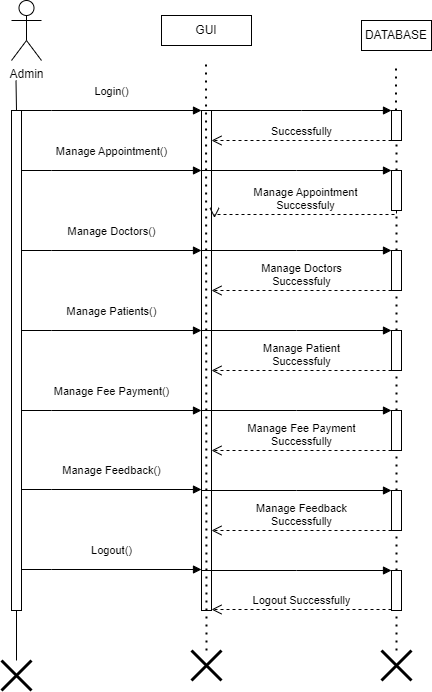


Figure: 3.2 Admin subsystem

## Data Description

### Major data objects

The Doctor Appointment Booking System web application utilizes the following major data objects:

User:

The User model represents the registered users of the system. Users can be classified as doctors, patients, or administrators based on the boolean fields is\_doctor, is\_patient, and is\_admin, respectively.

Specialization:

The Specialization model stores information about various medical specializations. Each specialization is associated with a name and an optional image (special\_image) representing the respective medical field.

Doctor:

The Doctor model corresponds to the registered doctors on the platform.

Each doctor is linked to a User object via a one-to-one relationship, allowing them to access the system functionalities.

Doctors have attributes such as gender, phone, biography, address, city, country, state, postal\_code, date\_birth, price, doc\_image, and status.

The specialization attribute connects the doctor to their chosen medical specialization using a foreign key relationship to the Specialization model.

Clinic:

The Clinic model represents the clinics associated with doctors on the platform.

Each clinic is linked to a User object (doctor) via a foreign key relationship, indicating the owner of the clinic.

The attributes of Clinic include clinic\_name, clinic\_image, and clinic\_address, which provide relevant information about the clinic.

Experience:

The Experience model keeps track of the professional experiences of doctors.

Doctors can add their previous experiences, such as hospital\_name, designation, start\_date, and end\_date.

Services:

The Services model allows doctors to list the medical services they offer.

Doctors can define the services they provide using the service attribute.

Degrees:

The Degrees model stores the educational degrees of doctors.

Doctors can add information about their degrees, including degree, institute, and year.

Award:

The Award model records any awards or accolades received by doctors.

Doctors can add details about their awards using attributes such as awards and award\_year.

Patient:

The Patient model represents the registered patients on the platform.

Each patient is linked to a User object via a one-to-one relationship, allowing them to access the system functionalities.

Attributes of Patient include date\_birth, blood\_group, address, city, country, state, postal\_code, phone, gender, and patient\_image.

Subadmin:

The Subadmin model represents sub-administrators of the system. Sub-admins are linked to a User object via a one-to-one relationship, allowing them to access specific administrative functionalities.

The attributes of Subadmin include date\_birth, blood\_group, address, city, country, state, postal\_code, phone, gender, and admin\_image.

Appointment:

The Appointment model stores information about scheduled appointments between doctors and patients. Each appointment is linked to a Doctor object and a Patient object using foreign key relationships.

Attributes of Appointment include booking\_date, appt\_date, appt\_time, and status, representing the status of the appointment (New, Accepted, Confirmed, Rejected).

Doctor Schedule:

The Doctor Schedule model allows doctors to define their availability for appointments on different days of the week.

Each schedule entry is linked to a Doctor object via a foreign key relationship.

The attributes include day\_of\_week, start\_time, and end\_time.

Payment:

The Payment model records payments made by patients to doctors for their services.

Each payment is associated with a Doctor object and a Patient object using foreign key relationships.

Attributes of Payment include payment\_date, amount, and status, representing the payment status (Paid).

Review:

The Review model allows patients to leave reviews and ratings for doctors.

Each review is associated with a Doctor object and a Patient object using foreign key relationships.

Attributes include date, message, title, and rating, representing the date of the review, the review message, the review title, and the rating given by the patient.

Chat:

The Chat model facilitates communication between users through chat messages.

Each chat message is sent by a User object (sender) and received by another User object (receiver).

Attributes include message, timestamp, representing the content of the message and the timestamp when it was sent.

Contact:

The Contact model is used for handling user inquiries and feedback.

Users can submit a contact form with their name, email, message, and the form submission date.

## System level data model

The system level data model for the Doctor Appointment Booking System web application includes the aforementioned data objects and their relationships. The model represents the structure and associations between different entities in the application.

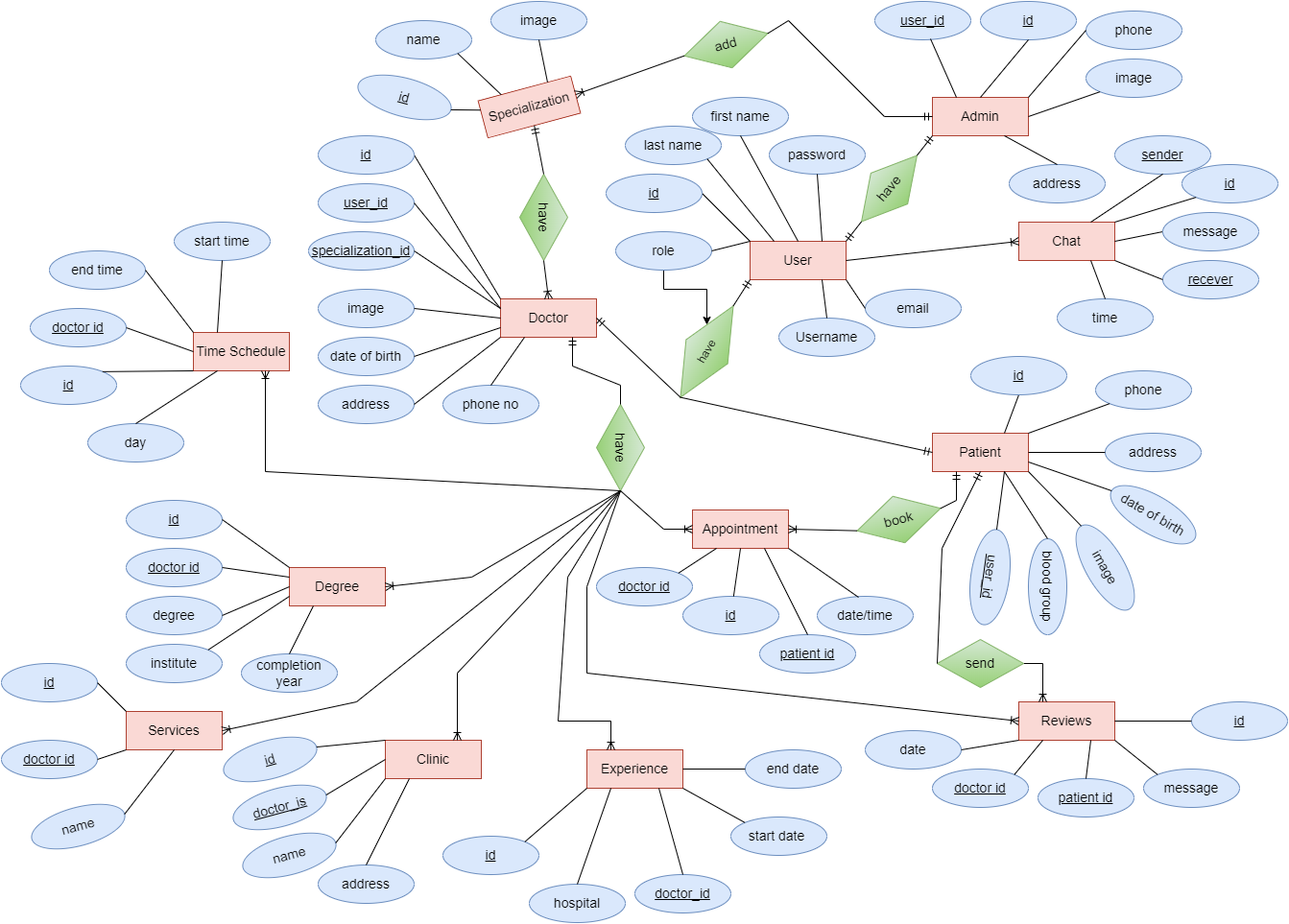


Figure: 3.3 ERD Diagram

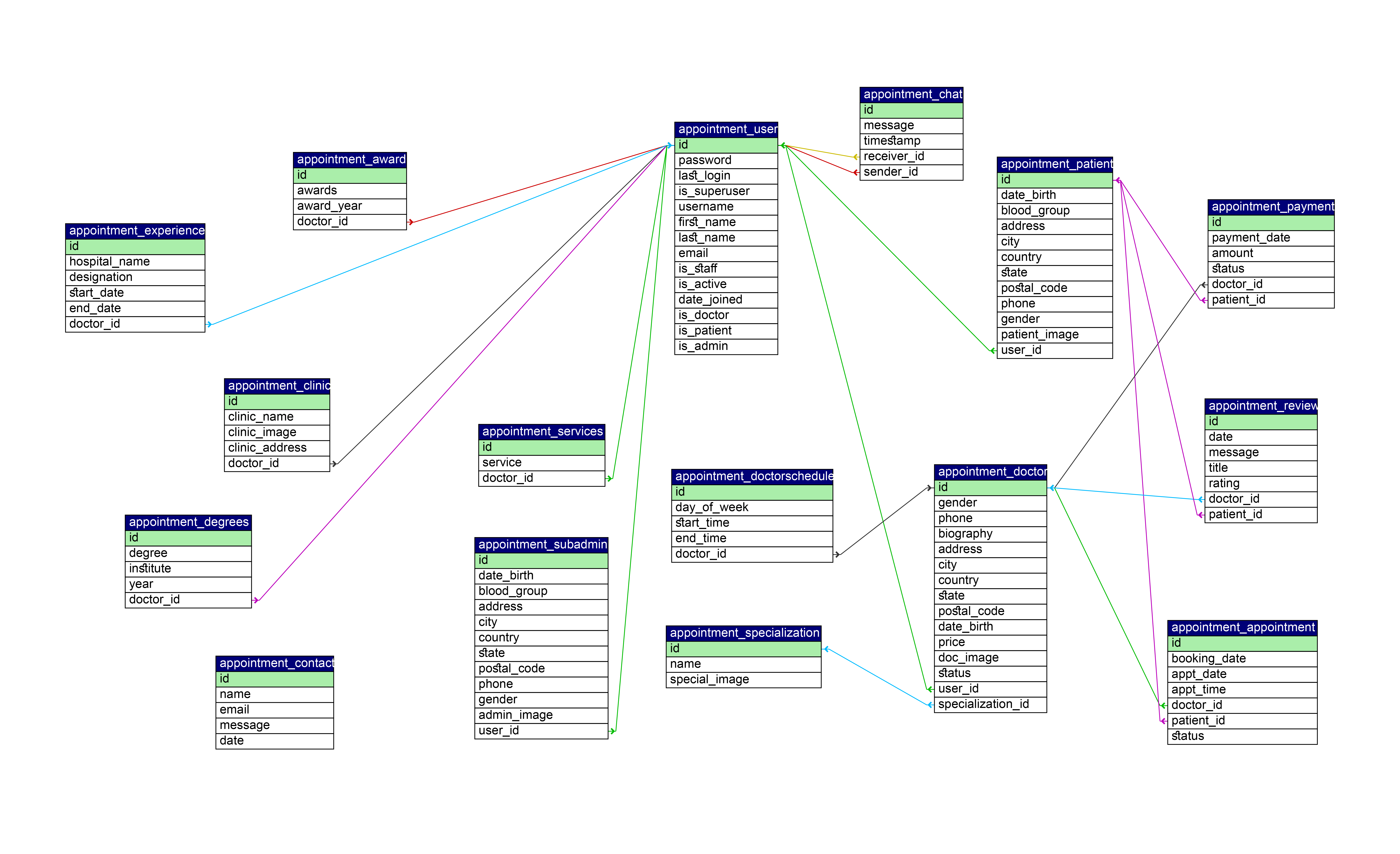


Figure: 3.4 Class Diagram

## System Interface Description

The System’s interface(s) to the outside world are described.

### External machine interfaces

The Doctor Appointment Booking System web application interfaces with the following external machine interfaces:

**Web Browser**: Users interact with the application through a web browser, which sends HTTP requests to the server and receives HTML responses.

**Database Management System**: The application interfaces with a relational database management system MySQL to store and retrieve data related to user accounts, doctor profiles, clinics, appointments, and chat messages.

### External system interfaces

The Doctor Appointment Booking System web application may integrate with external systems for additional functionalities, such as:

**Email Service**: The system can integrate with an email service provider to send notifications and reminders to users about their appointments.

|  |
| --- |
| Chapter # 4 **Subsystem/module Description** |

1. Subsystem/module Description

A description of each subsystem is presented.

## Description for Subsystem ‘n’

A detailed description of each subsystem is presented. Section 4.1 is repeated for each of ‘n’ subsystems i.e. this section is repeated for all subsystems.

### Subsystem scope

The Subsystem 'n' in the Doctor Appointment Booking System web application is responsible for managing the chat functionality between doctors and patients. It enables secure and efficient communication between the two parties within the application.

### Subsystem flow diagram/ desired UML diagram

The subsystem flow diagram or UML diagram depicts the components and their relationships within the chat subsystem. Please refer to the following textual description and create the appropriate diagram:

Chat Manager: Manages the overall functionality of the chat subsystem, including message handling and communication between doctors and patients.

Message Model: Defines the data model for chat messages, including the doctor, patient, message content, and date time.

User Interface: Provides a user-friendly interface for doctors and patients to view and send chat messages.

Chat Controller: Handles the incoming and outgoing chat messages, performs message validation, and triggers notifications for new messages.

Security Module: Implements security measures to ensure the privacy and confidentiality of chat messages, including encryption and access control.

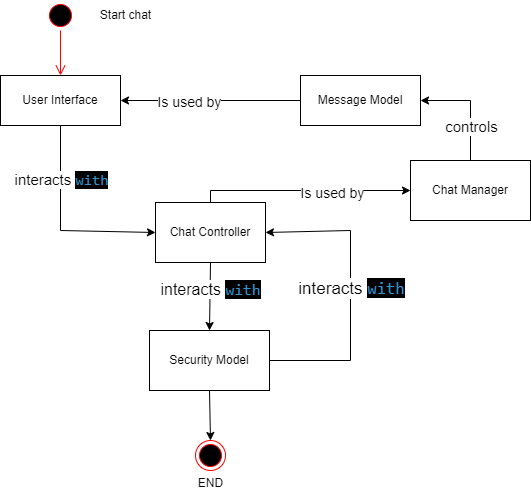


Figure: 4.1 Subsystem Module

### Subsystem ‘n’ components

The chat subsystem includes the following components:

Chat Manager

Message Model

User Interface

Chat Controller

Security Module

### Description for sub system ‘n’ Component ‘k’

A detailed description and diagram (if applicable) of each sub system component contained within the architecture is presented. Section 4.2 is repeated for each of (k) components.

#### Component ‘k’ interface description.

Chat Manager

start\_chat(doctor\_id: int, patient\_id: int) ->chat\_id: int: Initiates a chat session between a doctor and a patient and returns the unique chat identifier.

send\_message(chat\_id: int, sender\_id: int, message: str) -> success: bool: Sends a chat message from the sender (doctor or patient) to the specified chat.

get\_chat(chat\_id: int) ->chat\_data: dict: Retrieves the chat data, including the list of messages, participants, and chat details.

Message Model

doctor: ForeignKey(User): Reference to the doctor involved in the chat message.

patient: ForeignKey(User): Reference to the patient involved in the chat message.

message: CharField(max\_length=20): Content of the chat message.

datetime: DateTimeField: Date and time of the chat message.

User Interface

The user interface component provides a user-friendly chat interface for doctors and patients to view and send chat messages. It includes chat message display, input field, and options for sending messages and managing chat settings.

Chat Controller

The chat controller component handles incoming and outgoing chat messages. It performs message validation, ensures proper formatting, and triggers notifications for new messages. It interacts with the chat manager and message model to manage the flow of messages.

Security Module

The security module ensures the privacy and confidentiality of chat messages. It implements encryption algorithms and access control mechanisms to prevent unauthorized access to chat content.

### Restrictions/limitations

The chat subsystem has the following restrictions/limitations:

Chat messages are limited to a maximum length of 20 characters.

The chat system does not support file attachments or multimedia messages.

Chat messages are not automatically translated or localized.

### Performance issues

Performance issues related to the chat subsystem may include:

Handling a large number of concurrent chat sessions.

Ensuring real-time message delivery and synchronization.

Optimizing message retrieval and display for a smooth user experience.

### Design constraints

Design constraints for the chat subsystem may include:

Compatibility with different browsers and devices.

Integration with authentication and authorization mechanisms.

Ensuring scalability and high availability to handle increased chat load.

|  |
| --- |
| Chapter # 5  **Behavioral Model and Description** |

1. Behavioral Model and Description

The Doctor Appointment Booking System web application exhibits the following system behavior:

## Description for system behavior

User The doctor appointment booking system web application allows users (doctors and patients) to schedule and manage appointments. The system provides the following behaviors:

User Registration: Users can register as doctors, patients, or administrators.

User Authentication: Users can log in to the system using their credentials.

Doctor Profile Management: Doctors can update their personal information, such as contact details, biography, address, specialization, experience, services, degrees, awards, and clinic information.

Patient Profile Management: Patients can update their personal information, such as contact details, address, and medical history.

Patient Profile Management: Patients can update their personal information, such as contact details, address.

Appointment Booking: Patients can search for available doctors and book appointments based on doctor availability and specialization.

Appointment Management: Doctors and Patient can view and manage their appointments, including approving or rejecting appointment requests, rescheduling appointments, and marking appointments as completed.

Doctor-Patient Chat: The system provides a chat functionality for doctors and patients to communicate securely.

Change Password: Patients ,Doctors and Admin can change their password.

Feedback Management: Patients can send feedback on doctor performance, communication and Behavior admin monitoring the feedbacks.

### Events/interrupts

User Registration Event: Triggered when a user registers a new account.

User Login Event: Triggered when a user logs in to the system.

Doctor Profile Update Event: Triggered when a doctor updates their profile information.

Patient Profile Update Event: Triggered when a patient updates their profile information.

Appointment Booking Event: Triggered when a patient books a new appointment.

Appointment Approval Event: Triggered when a doctor approves an appointment request.

Appointment Rejection Event: Triggered when a doctor rejects an appointment request.

Appointment Rescheduling Event: Triggered when a doctor reschedules an appointment.

Appointment Completion Event: Triggered when a doctor marks an appointment as completed.

Chat Message Event: Triggered when a doctor or patient sends a chat message.

### States

User Registration State: The system is waiting for user registration information.

User Login State: The system is waiting for user login credentials.

Doctor Profile Update State: The system is waiting for doctor profile update information.

Patient Profile Update State: The system is waiting for patient profile update information.

Appointment Booking State: The system is waiting for appointment booking details.

Appointment Approval State: The system is waiting for the doctor's approval decision.

Appointment Rejection State: The system is waiting for the doctor's rejection reason.

Appointment Rescheduling State: The system is waiting for the new appointment details.

Appointment Completion State: The system is waiting for the doctor's confirmation of appointment completion.

Chat Message State: The system is waiting for the chat message content.

## 

## State Transition Diagrams

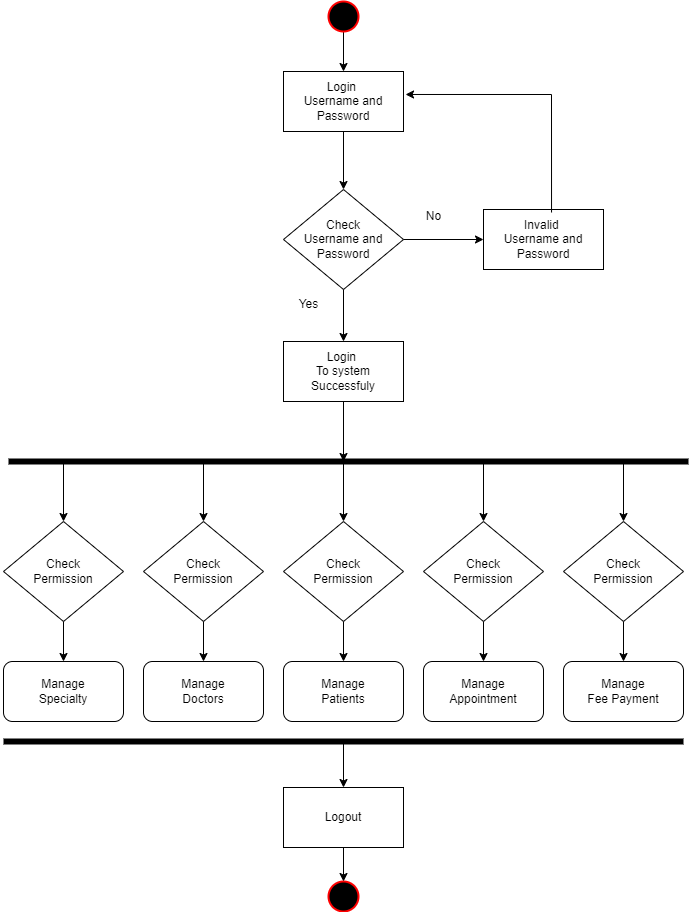


Figure: 5.1 State Transition Diagram

## 

## Control specification

The Doctor Appointment Booking System implements control specifications to govern system behavior. These include:

Appointment Validation: Ensuring the selected appointment date and time are within the doctor's availability schedule.

User Authentication: Verifying user credentials during login and restricting access to specific functionalities based on user roles.

Message Validation: Verifying the length and format of chat messages before sending or displaying them.

Appointment Status Updates: Updating the status of appointments based on doctor actions, such as approval, rescheduling, or cancellation.

|  |
| --- |
| Chapter # 6  **System Prototype Modeling and Simulation Results** |

1. System Prototype Modeling and Simulation Results

System prototype modeling and simulation results are used to evaluate the performance of a system before it is actually built. This can be done by creating a model of the system and simulating its behavior under different conditions. The results of the simulation can then be used to identify potential problems with the system and to make necessary changes before the system is built.

## Description of system modeling approach (if used)

For the doctor appointment booking system web application, a prototype model was created using Django, a Python-based web framework. The model follows the Model-View-Controller (MVC) architectural pattern, where the data models represent the entities and their relationships, views handle the user interface, and controllers manage the business logic and interactions with the database.

## Simulation results

The results of any system simulation some simulation results that could be used to evaluate the doctor appointment booking system web application:

**Performance:** The performance of the application could be evaluated by simulating the application with a large number of doctors and different types of Patients. The results of the simulation could then be used to determine how well the application scales and to identify any potential bottlenecks. For example, the simulation could be used to determine how long it takes for the application to load a appointment page when there are a large number of users logged in.

**Security:** The security of the application could be evaluated by simulating attacks on the application and evaluating how well the application's security features protect it from attack. For example, the simulation could be used to determine how easy it is for a hacker to gain access to the application's database.

**Usability:** The usability of the application could be evaluated by simulating users interacting with the application and evaluating how easy it is to use. For example, the simulation could be used to determine how long it takes for a user to complete a task in the application.

The results of the simulation to make changes to the application before it is built. This could help to ensure that the application is performance, secure, and easy to use.

.

## Special performance issues

The doctor appointment booking system web application should be designed and optimized to handle the following performance issues:

**Scalability**: The system should be able to handle a growing number of users, doctors, patients, and appointments without significant performance degradation.

**Response Time**: The system should provide quick response times for actions such as appointment booking, profile updates, and chat message delivery.

**Concurrent Usage**: The system should support multiple concurrent users accessing and interacting with the application without negatively impacting performance or data integrity.

## Prototyping requirements

If The system prototype requires the following resources and dependencies:

**Django framework**: The web application is developed using the Django framework, which provides the necessary tools and libraries for building the application.

**Database Management System**: The system requires a database management system MySQL to store and retrieve data.

**Web Frontend**: The system utilizes a web frontend template (referenced in the project description) for the user interface.

|  |
| --- |
| Chapter # 7  **System Estimates and Actual Outcome** |

1. System Estimates and Actual Outcome

During the development of the "Doccur An Online Platform for Doctor Appointment Booking system" application, it is important to establish estimates for various aspects of the system. These estimates serve as a basis for planning, resource allocation, and tracking progress. Here is an overview of the estimated and actual outcomes for the system:

## Historical data used for estimates

The historical data was used to create a baseline estimate for the cost, time, and resources required for Doctor Appointment Booking system. This baseline estimate was then used to create a more detailed estimate. The detailed estimate took into account the specific features of Doctor Appointment Booking system and the specific needs of the project.

## Estimation techniques applied and results

In the case of Doctor Appointment Booking system, the top-down estimation technique used to estimate the overall cost of the project. This estimate could then be broken down into smaller components, such as the cost of development, testing, and deployment.

In the case of Doctor Appointment Booking system, the bottom-up estimation technique used to estimate the cost of development, testing, and deployment. These estimates could then be added up to get the overall cost of the project.

In the case of Doctor Appointment Booking system, the parametric estimation technique used to estimate the cost of Doctor Appointment Booking system by using historical data from similar projects. This data could be used to estimate the cost of development, testing, and deployment.

In the case of Doctor Appointment Booking system, the Monte Carlo simulation technique used to generate a range of possible estimates for the cost of Doctor Appointment Booking system. This range of estimates could be used to get a better understanding of the uncertainty around the cost of the project.

### Estimation technique ‘m’

Parametric estimation: Parametric estimation is a cost estimation technique that uses historical data to estimate the cost of a project. This technique is often used for projects that are similar to previous projects.

In the case of Doctor Appointment Booking system, the parametric estimation technique could be used to estimate the cost of Doctor Appointment Booking system by using historical data from similar projects. This data could be used to estimate the cost of development, testing, and deployment.

### Estimate for technique ‘m’

Estimate generated for technique ‘m’.

## Actual Results and Deviation from Estimates

The final cost for the project (at this point in time) is presented here. A deviation chart should be presented to illustrate the comparison between actual cost of system and estimated cost.

|  |  |  |
| --- | --- | --- |
| ****Item**** | ****Estimated Cost**** | ****Actual Cost**** |
| Development | Rs.50,000 | Rs.10,000 |
| Testing |  |  |
| Deployment |  |  |
| Total |  |  |

. Table 7.1 Estimated Cost

## System Resources (Required and Used)

People, hardware, software, tools, and other resources proposed to build the software are noted here. On the bases of which cost estimations were performed

### System Resources Required

System resources that could be required and used by the Doctor Appointment Booking system application:

* **Software:** The Doctor Appointment Booking system project required the following software:
  + MySQL
* **Tools:** The Doctor Appointment Booking system project required the following tools:
  + Visual Studio Code
* People: The Doctor Appointment Booking system project required
  + Developer
* Hardware:
  + Good Processor
  + Lot of RAM
  + Large Hard drive

### System Resources Used

**People:** The Doctor Appointment Booking system project is actually built by one developer.

**Hardware:**

* Good processor
* **16GB** of RAM, and a
* 500GB hard drive.

**Software:** The Doctor Appointment Booking system project was built using the following software:

* + The MySQL database management system

**Tools:** The Doctor Appointment Booking system project was built using the following tools:

* + The Visual Studio code editor

|  |
| --- |
| Chapter # 8  **Test Plan** |

1. Test Plan

This section describes the overall testing strategy and issues that are required to properly execute effective tests. At least one of the testing strategy should be followed:

## System Test and Procedure

To test the complete application as a whole, system testing has been used. It is beneficial to check whether the application meets its requirements and fulfill Quality Standards.

## Testing strategy

The overall strategy for software testing is described.

### Unit testing

The software units in an application are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches to the type and size supported by python. The various controls are tested to ensure that each performs its action as required.

Commonly used method is White-Box Testing method. Every time a component of the program is changed, it can be run for testing that is the biggest and famous benefit of this testing phase. Issues that are arise during this phase, allowing to be resolved as quickly as possible. Unit testing is familiar by software developers. It allows them to test their application units before move them to testers for formal testing.

User Acceptance Testing:

User acceptance of an application is the key factor for the success of any application. The application under consideration is tested for user acceptance by constantly keeping in touch with the application users at time of developing and making changes whenever required.

Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Description | Expected Results | Actual Results | Pass/Fail |
| A buyer from an international country posts a job or project. | A buyer from an international country creates a new job or project on Doctor Appointment Booking system. The buyer enters all of the required information, including the job title, description, budget, and deadline. The buyer then submits the job or project. | The job or project is successfully created and the buyer is redirected to the job or project page. | The job or project is not created or the buyer is not redirected to the job or project page. | Pass |
| A seller in Pakistan bids on a job or project. | A seller in Pakistan views a job or project that has been posted by a buyer from an international country. The seller reviews the job or project description and budget, and then submits a bid. | The bid is successfully submitted and the seller is notified that their bid has been received. | The bid is not submitted or the seller is not notified that their bid has been received. | Pass |
| A buyer and seller communicate with each other. | A buyer and seller who have worked together on a job or project communicate with each other through the Doctor Appointment Booking system messaging system. The buyer and seller send messages to each other and receive messages from each other. | The buyer and seller are able to communicate with each other through the Doctor Appointment Booking system messaging system. | The buyer and seller are not able to communicate with each other through the Doctor Appointment Booking system messaging system. | Pass |
| A buyer pays a seller for completed work. | A buyer pays a seller for completed work through the Doctor Appointment Booking system payment system. The buyer enters the amount to be paid and the seller's bank account information. The buyer then submits the payment. | The payment is successfully processed and the seller is notified that they have been paid. | The payment is not processed or the seller is not notified that they have been paid. | Pass |

Table 8.1 Test Case

.

### Integration testing

The integration testing strategy and procedure is specified. This section includes a discussion of the order of integration by System function.

### Validation testing

The validation testing strategy and procedure is specified. This section includes a discussion of the order of validation by system.

### High-order testing (a.k.a. System Testing)

The high-order testing strategy for Doctor Appointment Booking system will focus on the following areas performance of Doctor Appointment Booking system will be tested to ensure that it can handle the expected load of users. The security of Doctor Appointment Booking system will be tested to ensure that it is protected from unauthorized access and malicious attacks

#### Security testing

**Security Testing** is a type of software testing that intends to uncover vulnerabilities of the system and determine that its data and resources are protected from possible intruders.

**Focus Areas**

There are four main focus areas to be considered in security testing (Especially for web sites/applications):

* **Network security:** This involves looking for vulnerabilities in the network infrastructure (resources and policies).
* **System software security:** This involves assessing weaknesses in the various software (operating system, database system, and other software) the application depends on.
* **Client-side application security:** This deals with ensuring that the client (browser or any such tool) cannot be manipulated.
* **Server-side application security:** This involves making sure that the server code and its technologies are robust enough to fend off any intrusion.
* Security Testing is defined as a type of Software Testing that ensures software systems and applications are free from any vulnerabilities, threats, risks that may cause a big loss. Security testing of any system is about finding all possible loopholes and weaknesses of the system which might result into a loss of information.
* The goal of security testing is to identify the threats in the system and measure its potential vulnerabilities, so the system does not stop functioning or is exploited. It also helps in detecting all possible security risks in the system and help developers in fixing the problems through coding.

Table 8.2 Security Test

|  |  |
| --- | --- |
| **Test Type** | **Result (Pass or Fail)** |
| Security Scanning | Pass |
| Penetration Testing | Fail |
| Risk Assessment | Fail |
| Ethical Hacking | Pass |
| Security Auditing | Pass |

## Stress testing

In the Stress testing involve loading testing. Load testing is carried out to know the behavior of the system under the specific expected load. Example: The software is tested in load of 20 users accessing resources at the same time; the system response time was good.

Load testing lets you measure your website's [quality of service](https://en.wikipedia.org/wiki/Quality_of_service) performance based on actual customer behavior. Nearly all the load testing tools and frame-works follow the classical load testing paradigm: when customers visit your web site, a script recorder records the communication and then creates related interaction scripts. A load generator tries to replay the recorded scripts, which could possibly be modified with different test parameters before replay. In the replay procedure, both the hardware and software statistics will be monitored and collected by the conductor, these statistics include the CPU, memory, disk IO of the physical servers and the response time, throughput of the system under test, etc. And at last, all these statistics will be analyzed and a load testing report will be generated. Load and performance testing analyzes [software](https://en.wikipedia.org/wiki/Software) intended for a multi-user audience by subjecting the software to different numbers of virtual and live users while monitoring performance measurements under these different loads. Load and performance testing is usually conducted in a test environment identical to the production environment before the software system is permitted to go live.

|  |  |  |
| --- | --- | --- |
| **Test Environment** | **No. of Users** | **Sustainability** |
| Web | 6 | Yes |

|  |
| --- |
| Chapter # 9  **Future Enhancements and Recommendations** |

1. Future Enhancements and Recommendations

## Payment Gateway:

## If payment processing is required for booking appointments or services, the application can integrate with a payment gateway service to securely handle online transactions.

|  |
| --- |
| Chapter # 10 **Conclusion / Summary** |

1. Conclusion / Summary

The doctor appointment booking system is an efficient and user-friendly web application developed using the Django framework. It provides a seamless experience for both doctors and patients, streamlining the appointment booking process and enhancing communication between healthcare providers and their patients.

Throughout the development of the system, various key features have been implemented to ensure its effectiveness. Users can register and log in as doctors, patients, or administrators, each with their own specific roles and functionalities. Doctors can create and manage their profiles, including personal information, contact details, biographies, and professional experiences. They can also add clinics, specify their availability, and offer specialized services.

Patients, on the other hand, can search for doctors based on various criteria such as specialization, location, and availability. They can view doctor profiles, check their schedules, and book appointments conveniently. The system provides an intuitive interface for appointment booking, allowing patients to select suitable dates and times.

The system also incorporates a chat functionality that enables direct communication between doctors and patients. This feature facilitates easy interaction, allowing patients to discuss their concerns, ask questions, and receive timely responses from doctors. Additionally, patients can provide feedback to doctors, helping to improve the quality of healthcare services.

Overall, the doctor appointment booking system enhances the efficiency and convenience of scheduling appointments in the healthcare industry. It reduces administrative tasks, improves communication, and enhances the patient experience. The system provides a reliable and secure platform for doctors and patients to connect, ensuring effective healthcare management.

In conclusion, the doctor appointment booking system proves to be a valuable tool for healthcare providers and patients alike. Its intuitive design, comprehensive features, and efficient functionality make it an essential component of modern healthcare services. With its successful implementation, it has the potential to revolutionize the way appointments are scheduled and managed, leading to improved patient care and satisfaction.

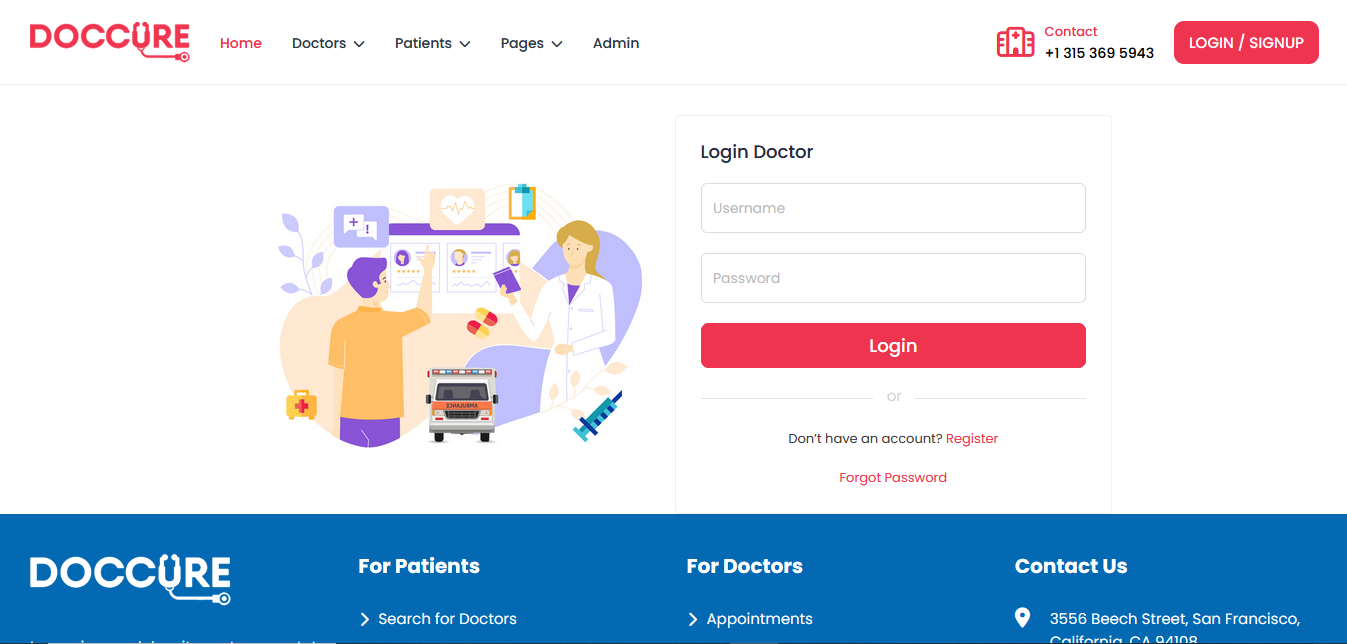
# ***Reference***

1. List Bailey NTJ. A study of queues and appointment systems in hospital out- patient departments, with special reference to waiting times. J Royal Stat Soc1952;14:185–99
2. Cayirli, T, E. Veral, and H. Rosen. (2006). Designing appointment scheduling systems for ambulatory care services. Health Care Management Science 9, 47–58.
3. Adebayo Peter Idowu., OlajideOlusegunAdeosun., and KehindeOladipoWilliams.,“Dependable Online Appointment Booking System for Outpatientin Nigerian Teaching Hospitals" International Journal of Computer Science & Information Technology (IJCSIT) Vol.6(4),pp.109-116,2014.
4. Arthur Hylton III and Suresh Sankaranarayanan “Application of Intelligent Agents in Hospital Appointment Scheduling System”, International Journal of Computer Theory and Engineering, Vol. 4, August 2012, pp. 625-630.
5. Yeo Symey, Suresh Sankaranarayanan, SitiNurafifahbintiSait “Application ofSmartTechnologiesforMobile Patient Appointment System”, International Journal of Advanced Trends in Computer Science and Engineering, august 2013
6. JagannathAghav, SmitaSonawane, and HimanshuBhambhlani “HealthTrack: Health Monitoring and Prognosis System using Wearable Sensors”, IEEE International Conference on Advances in Engineering & Technology Research 2014, pp. 1-5.

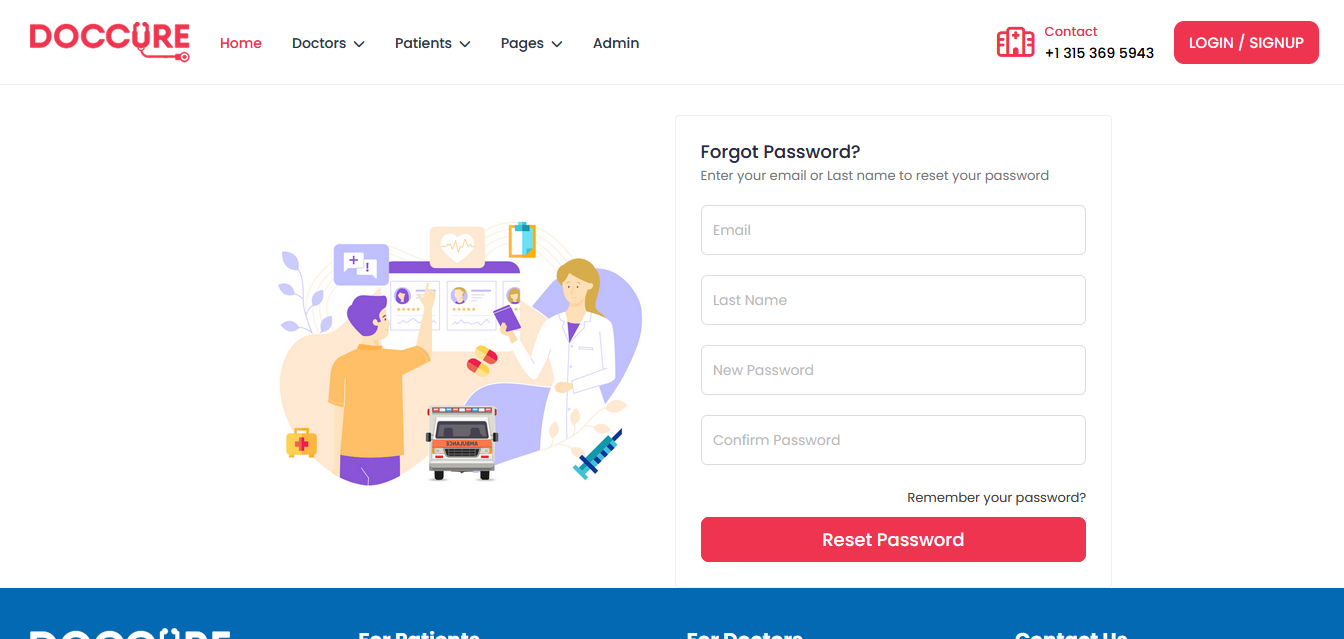
.

# **SNap shoots of the system**

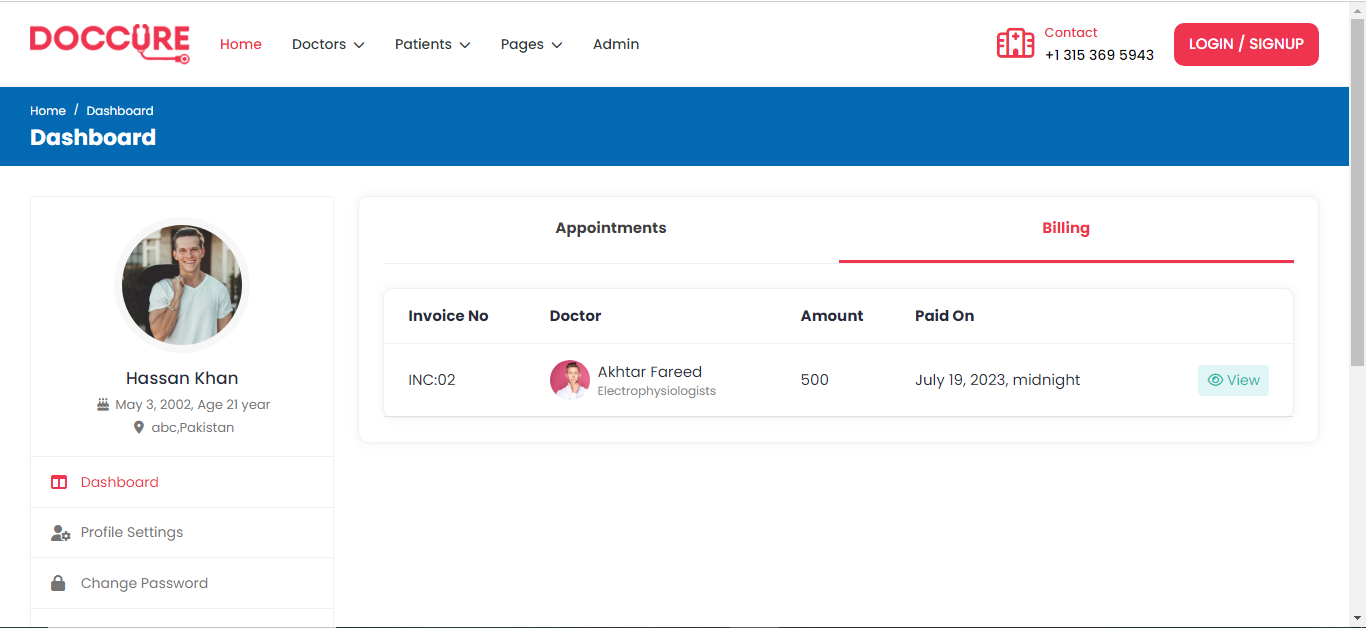
## Login Page

****

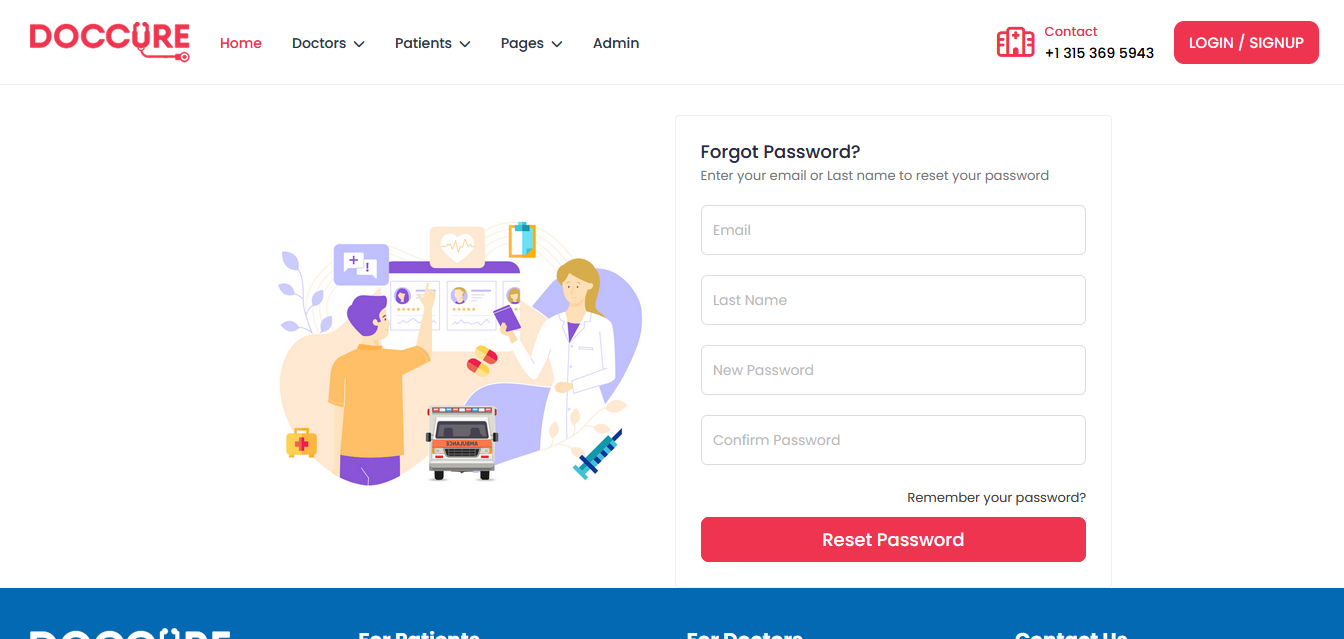
## Forgot Password



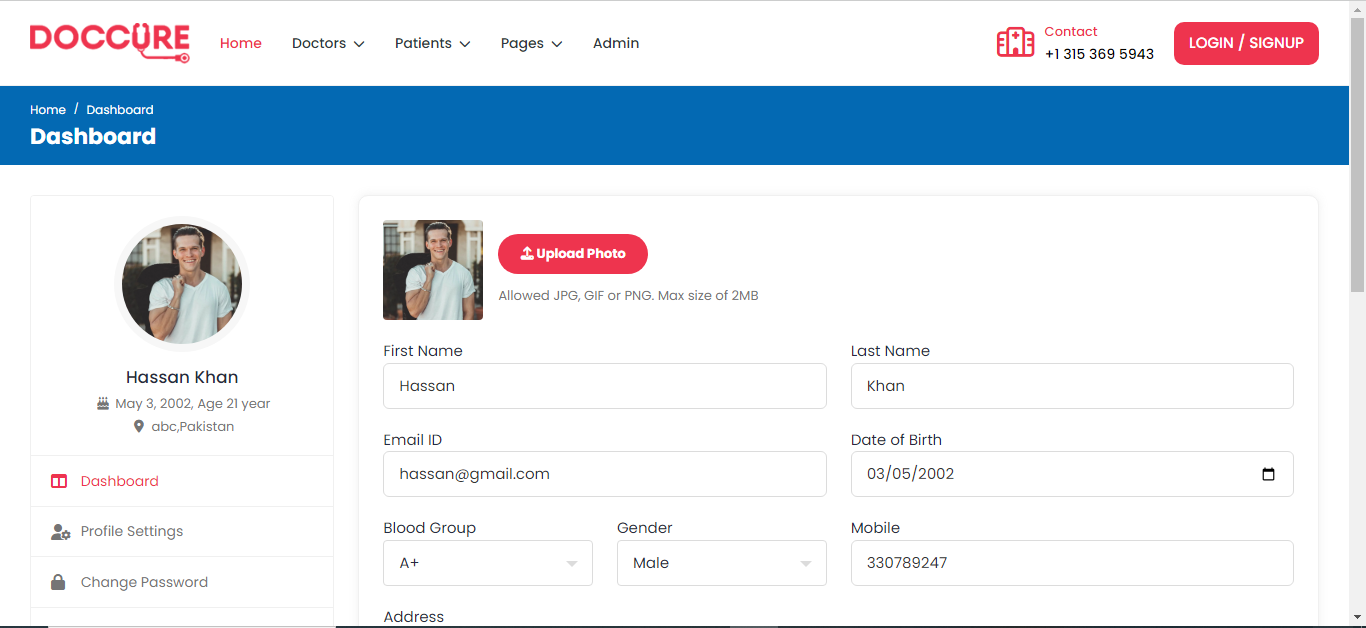
## Patient Dashboard



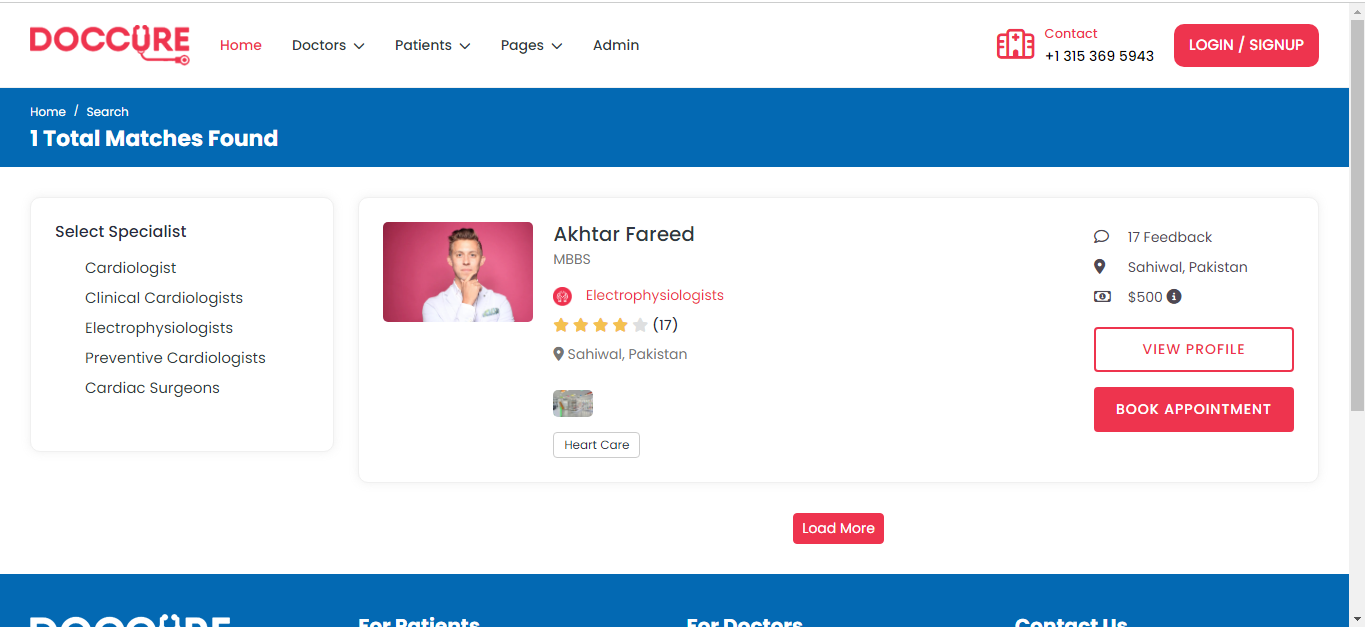
## Patient Dashboard 2



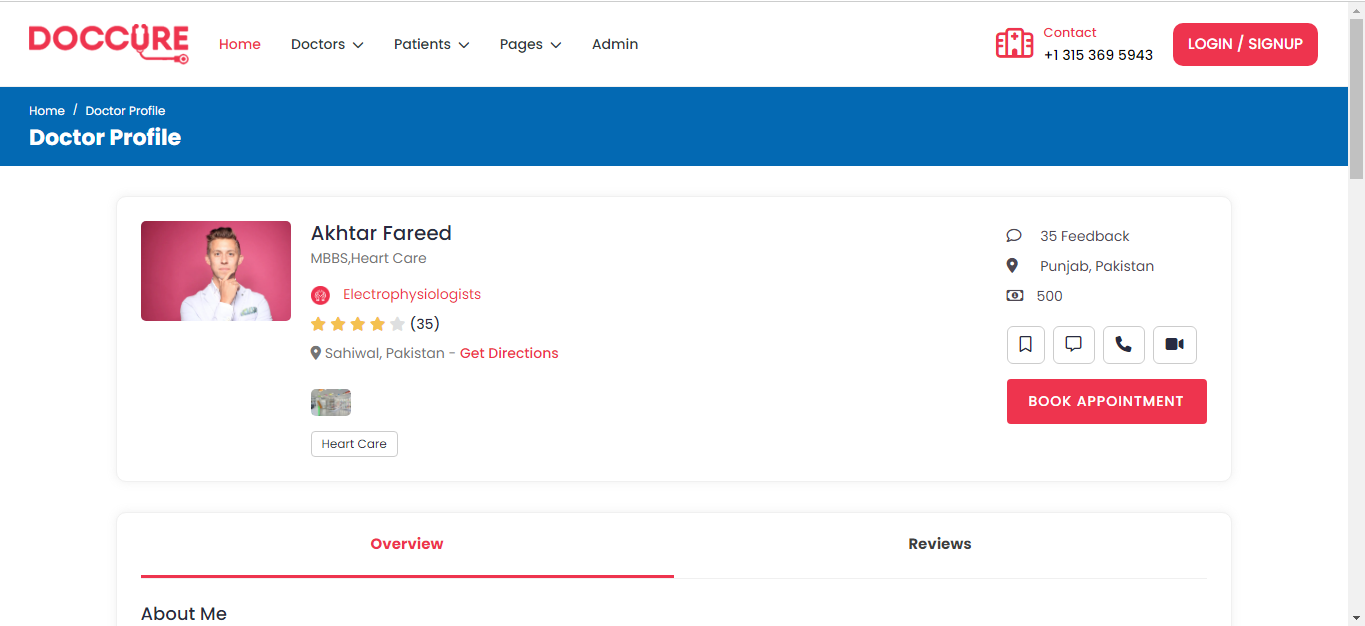
## Patient Profile



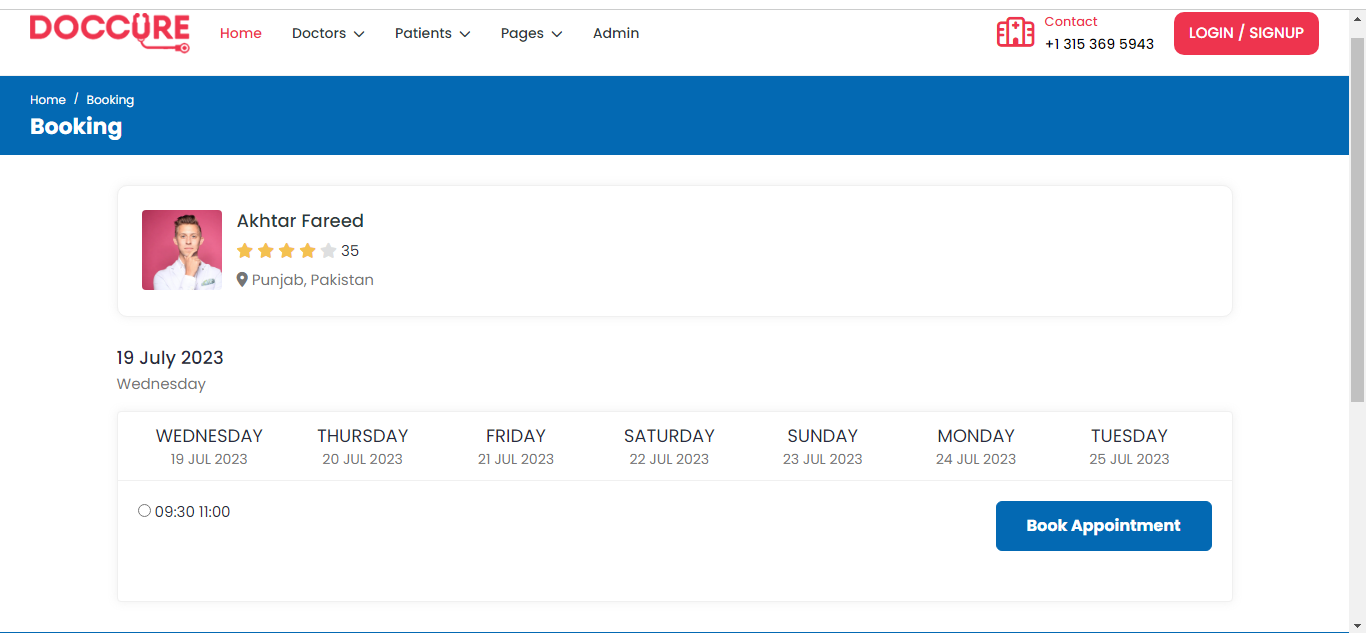
## Search Doctor



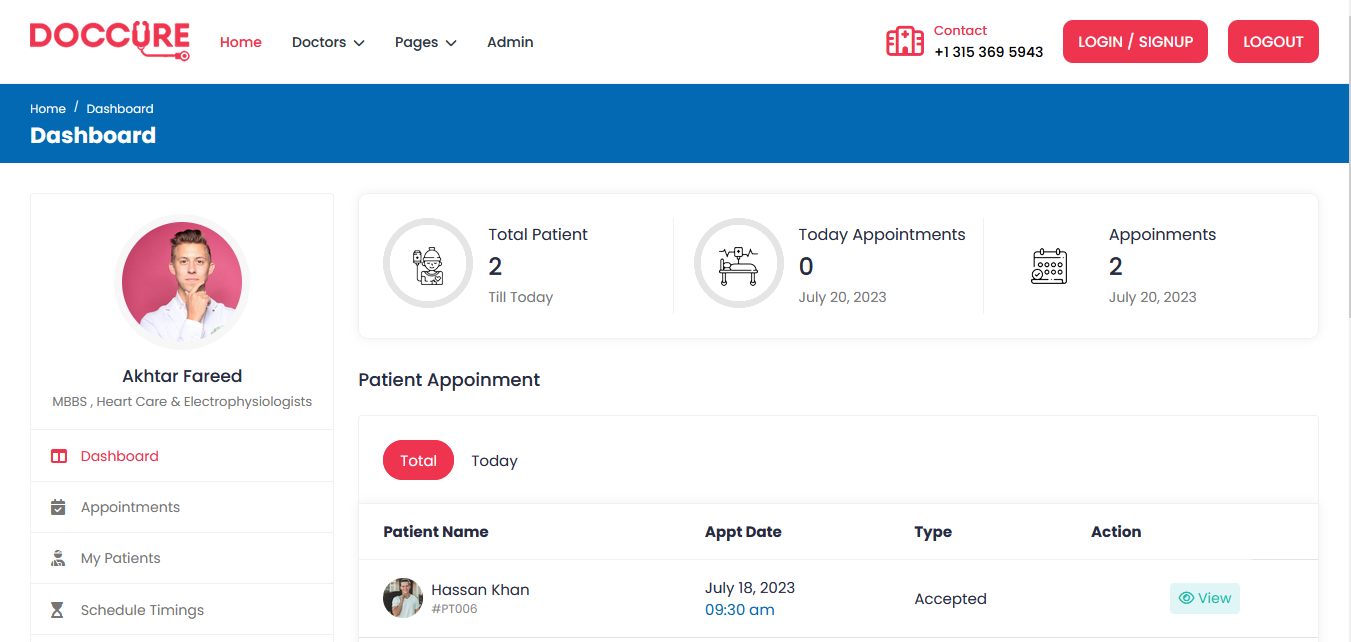
## View Doctor



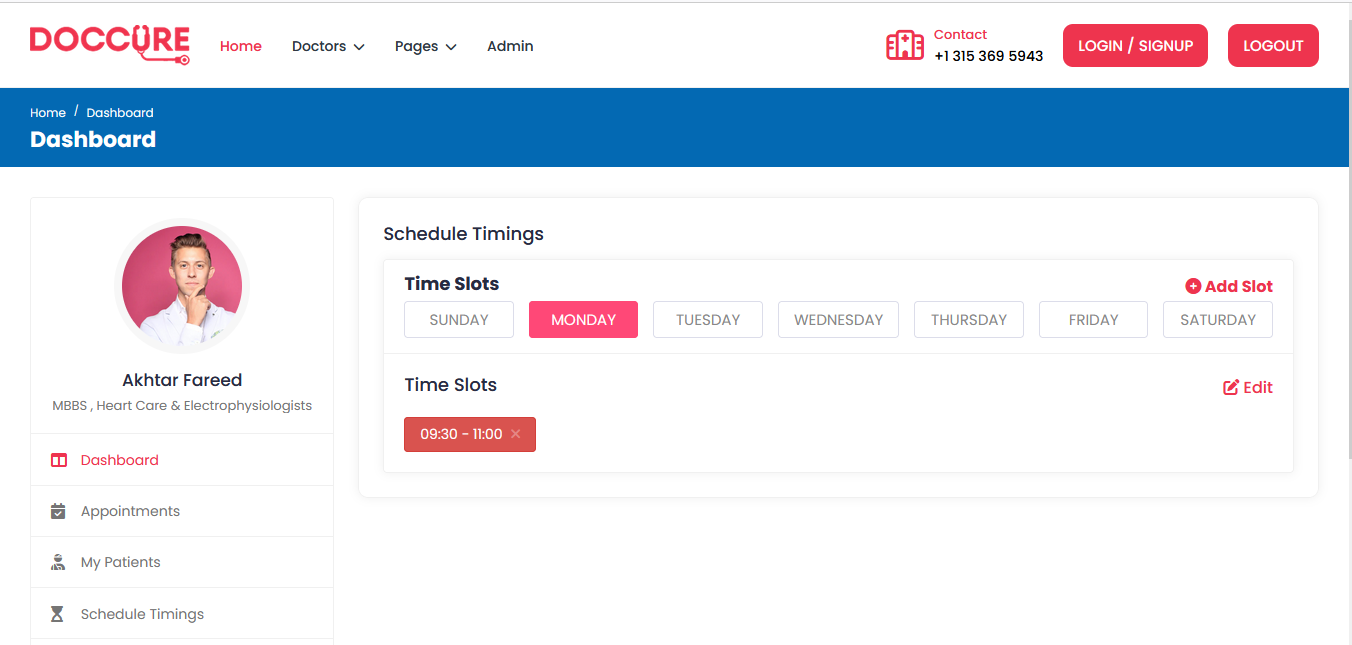
## Booking Timing Schedule



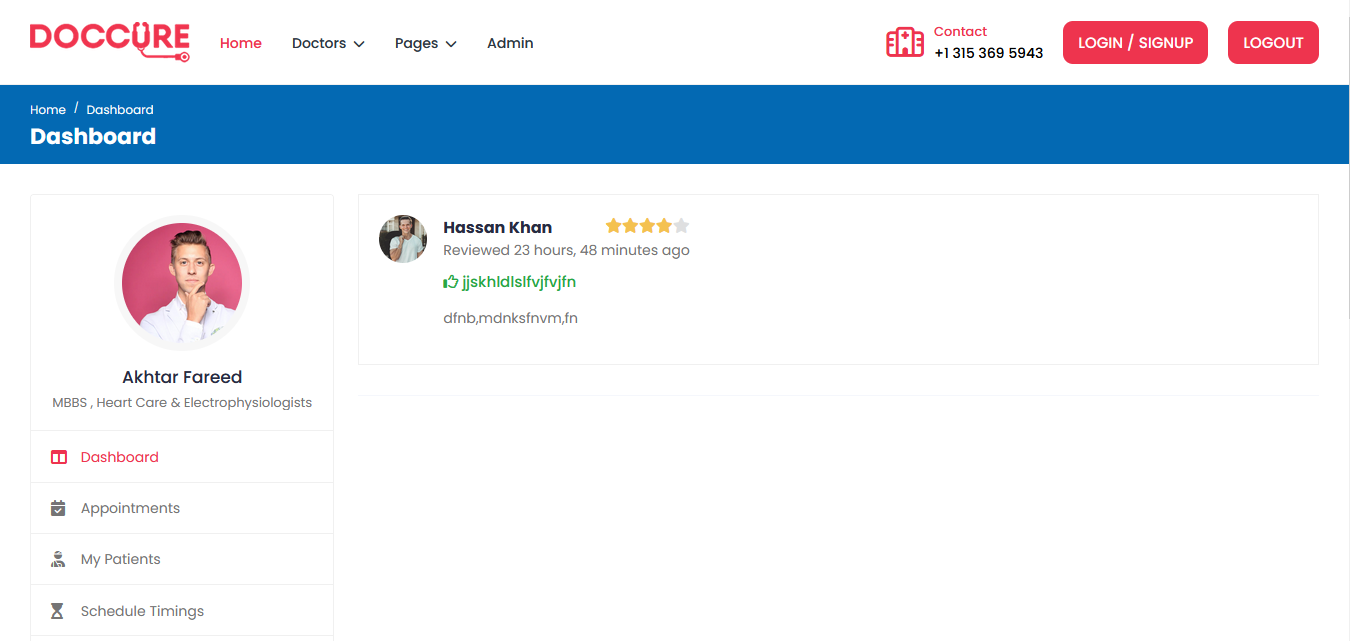
## Doctor Dashboard



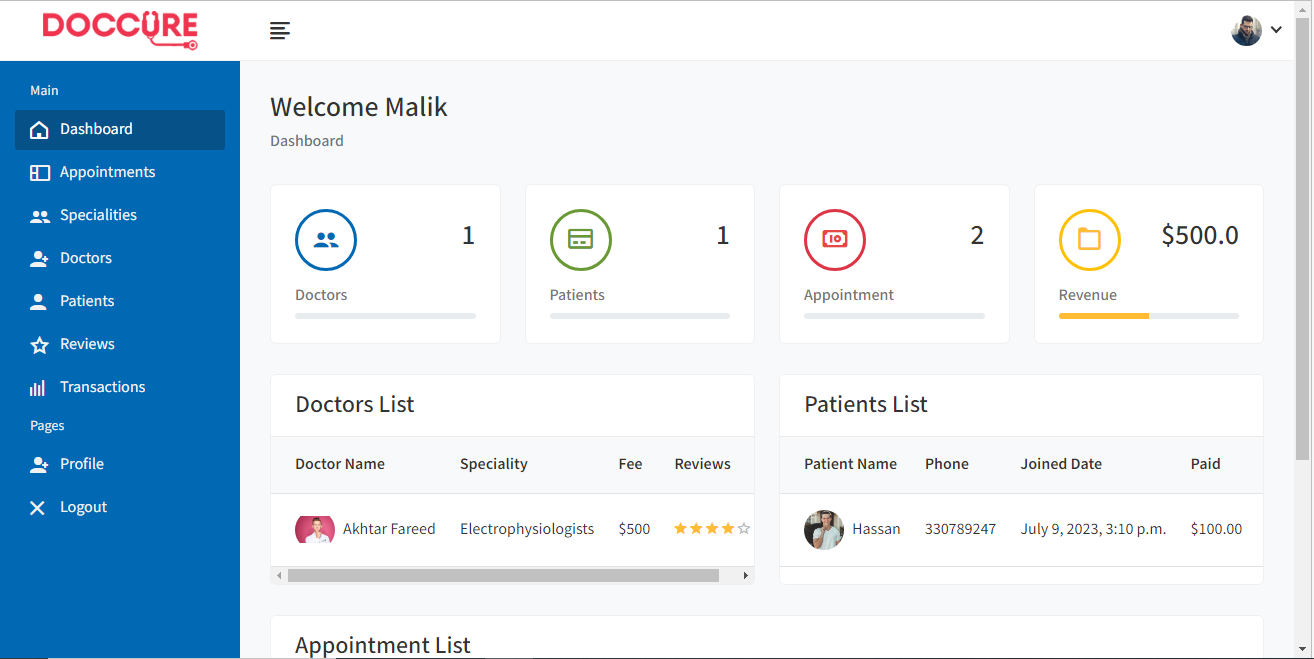
Doctor Time Scheduling



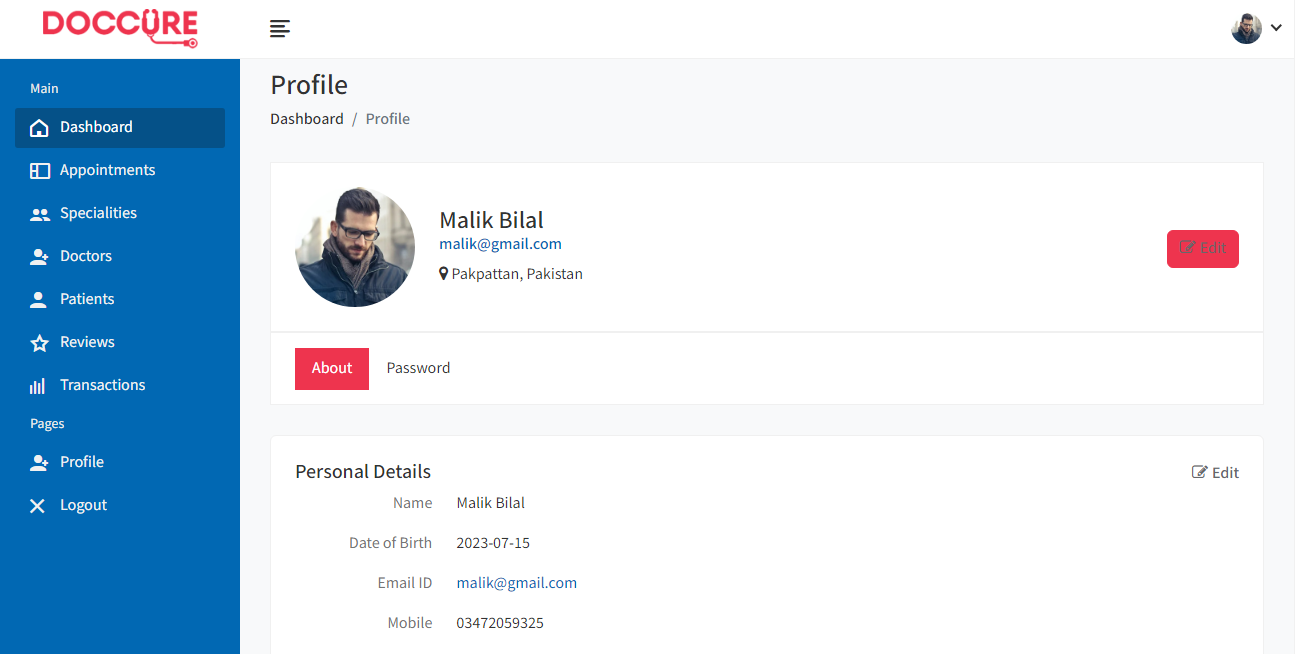
## Doctor Reviews



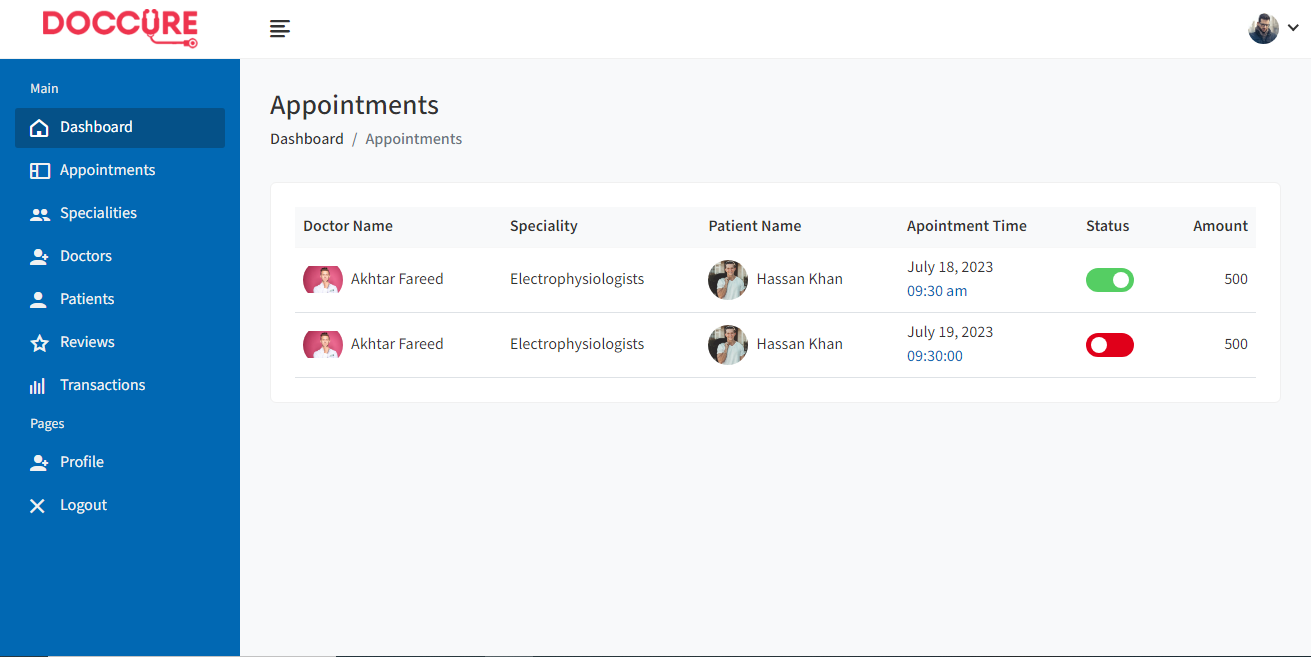
## Admin Dashboard



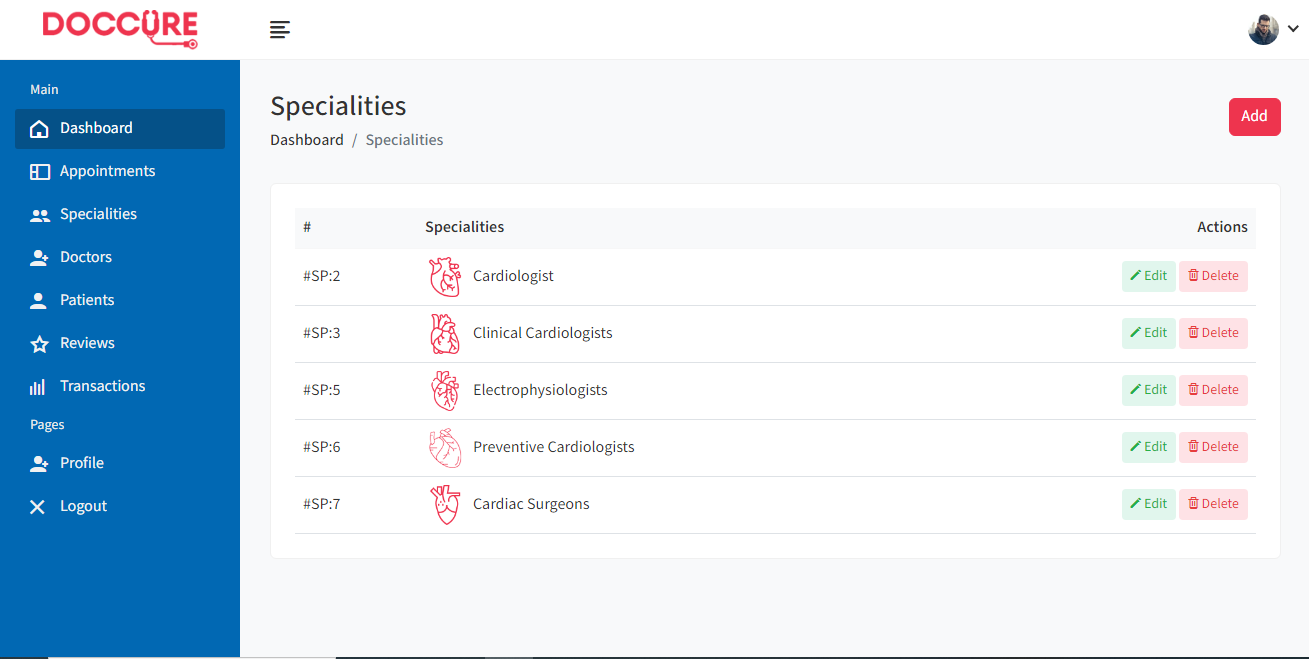
## Admin Profile



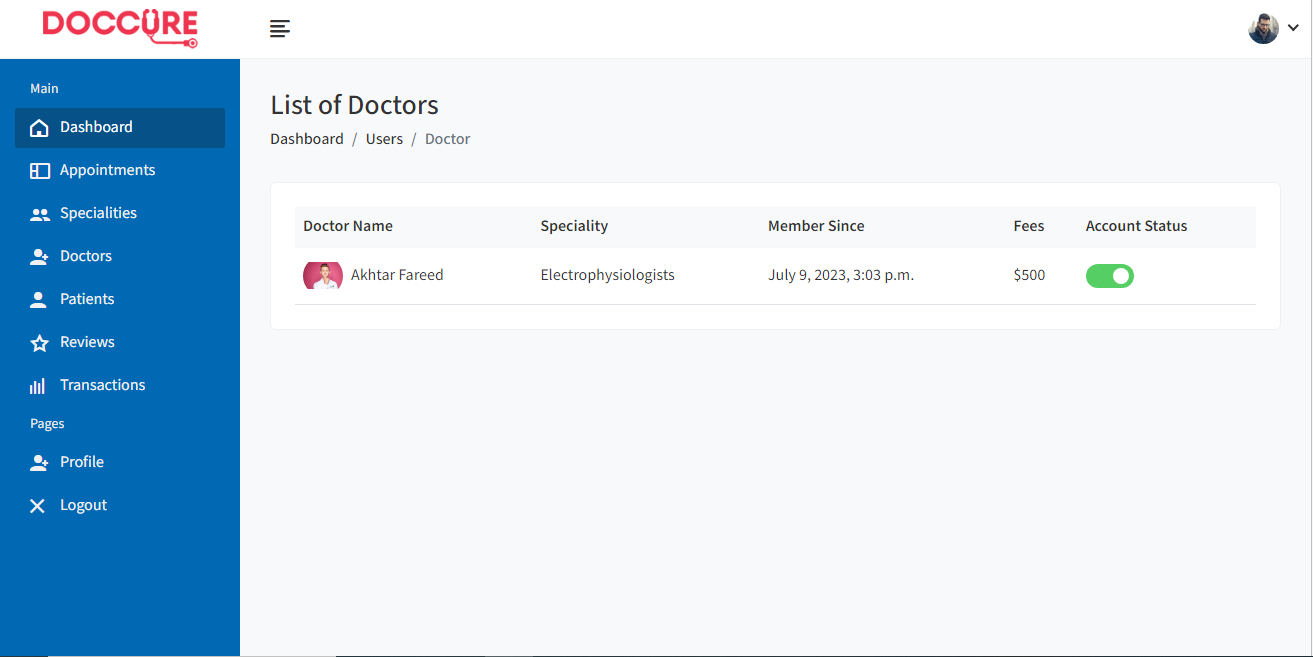
## Admin Appointment



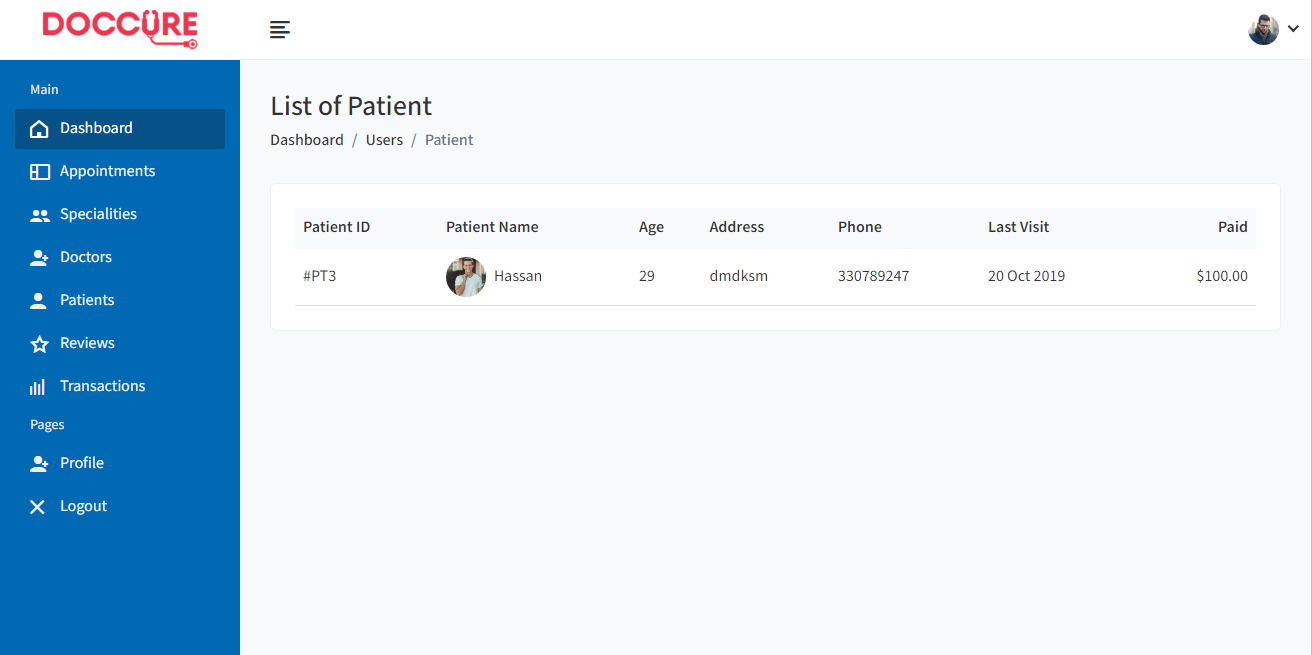
## Admin Specialization



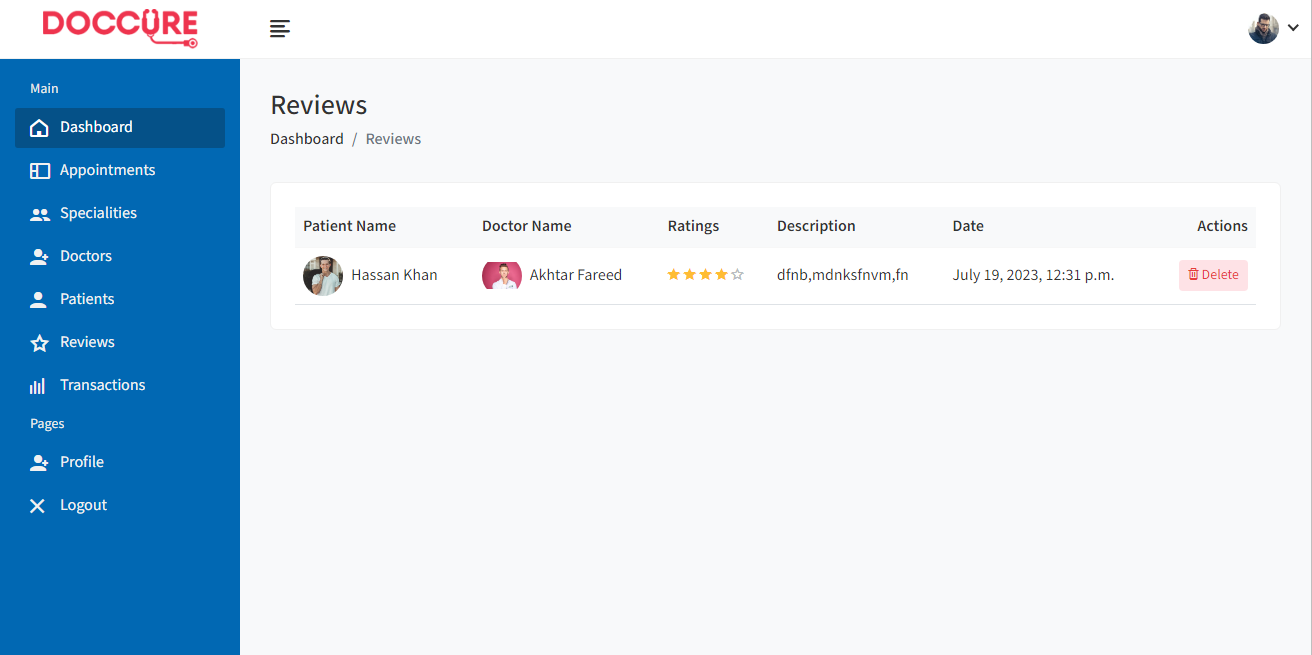
## Admin Doctor



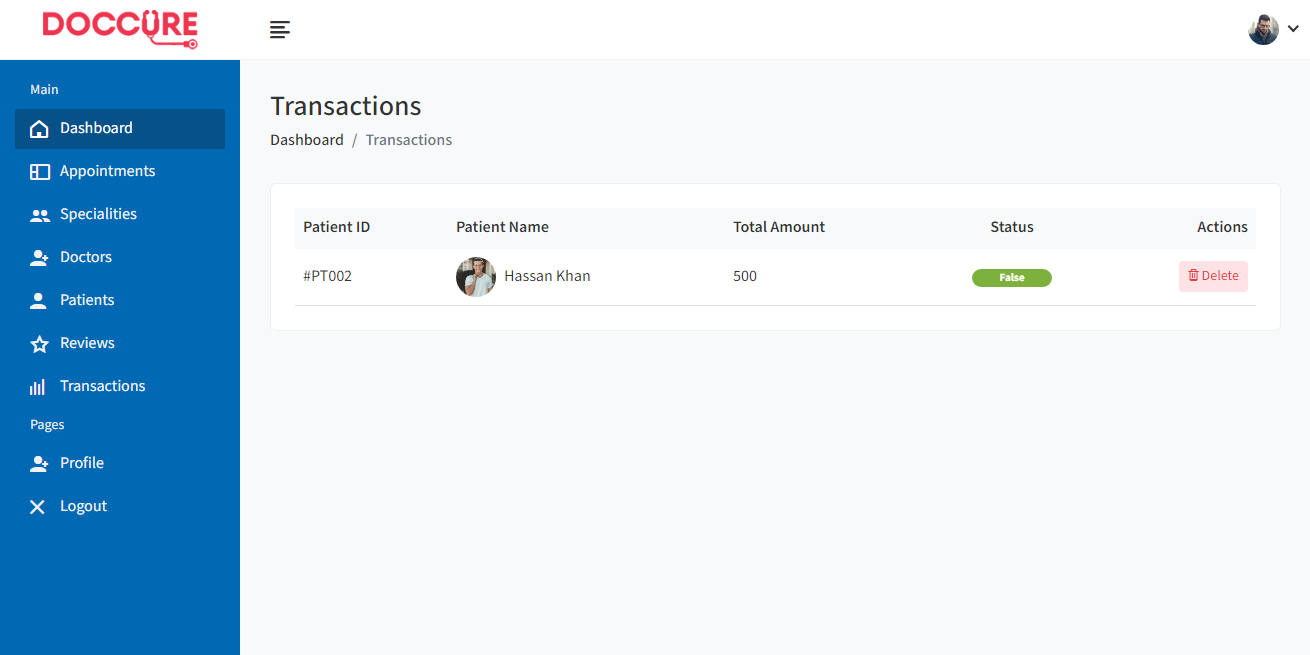
## Admin Patient List



## Admin Reviews



## Admin Transaction



## Chat Page

