# Hospital Management System

Nepal Business Collage,

Affiliated to Lincoln University

Final Year Project

By Sujit Kumar Sharma

# Acknowledgment

I, Sujit Kumar Sharma, would like to express my deepest gratitude to all those who have supported and guided me throughout the development of this Hospital Management System project. This endeavor would not have been possible without the collective efforts, guidance, and encouragement of several individuals.

First and foremost, I extend my heartfelt thanks to my project coordinators, Omar Farooque and Abhay Das. Their valuable suggestions and insights have been instrumental in shaping this project. Their expertise, constructive criticism, and unwavering support have been indispensable in navigating the challenges encountered during the project development. I am deeply grateful for their commitment and dedication, which have significantly contributed to the project's success.

I am also immensely grateful to my friend, Mr. Jeeten Mandel, for his constant support and encouragement throughout the project. His assistance and moral support have been a source of great motivation. Whether it was discussing ideas, offering feedback, or simply providing words of encouragement, his involvement has been crucial in keeping me focused and driven.

A special thanks to Rujal Baniya, whose assistance in building and debugging the project has been invaluable. His technical expertise, dedication, and willingness to help have greatly contributed to the successful completion of this project. Rujal's ability to troubleshoot and solve complex problems has been particularly beneficial, ensuring the project runs smoothly and efficiently.

Additionally, I would like to acknowledge all those who indirectly contributed to this project. The insights and knowledge gained from various online resources, forums, and communities have been invaluable. The collaborative spirit and wealth of information available online have been instrumental in overcoming numerous hurdles during the development process.

This project would not have been possible without the collective efforts and support of these individuals. I am truly thankful for their contributions, guidance, and unwavering support. Their belief in my abilities and their encouragement have been a driving force behind this project. I am proud to have had the opportunity to work with such dedicated and talented individuals, and I am deeply appreciative of all the assistance and guidance I have received.

# Abstract

The primary objective of the Hospital Management System project was to improve the management and operational efficiency of my town's hospital, which faced various daily challenges. This system streamlines hospital administration and enhances patient care through distinct functionalities for Admins, Doctors, and Patients.

For **Admins**, the system includes:

* **Doctor Management**: Register, view, approve, reject, or delete doctors.
* **Patient Management**: Admit, view, approve, reject, and discharge patients.
* **Invoice Management**: Generate and download PDF invoices based on treatment costs.
* **Appointment Management**: View, book, and approve patient appointments.

For **Doctors**, the system provides:

* **Job Application**: Apply for hospital jobs and access only after admin approval.
* **Patient Management**: View assigned patients' details and access discharged patient lists.
* **Appointment Management**: View and manage appointments booked by the admin.

For **Patients**, the system offers:

* **Account Management**: Create an account for hospital admission with admin approval.
* **Doctor Information**: View assigned doctor’s details.
* **Appointment Management**: Check appointment status and book appointments with admin approval.
* **Invoice Management**: View and download PDF invoices after discharge.

Developed using Python Django, HTML, CSS, JavaScript, and xhtml2pdf, this project significantly enhances hospital management by providing a structured, efficient, and user-friendly interface, improving overall functionality and service quality. Future enhancements will continue to build on this foundation to ensure sustained improvement in hospital operations.

# Table of Content

[Hospital Management System 1](#_Toc24835)

[Acknowledgment 2](#_Toc30400)

[Abstract 3](#_Toc32137)

[Table of Content 5](#_Toc15075)

[1. Project Overview 7](#_Toc20648)

[1.1 Introduction 7](#_Toc7437)

[1.2 Objectives: 7](#_Toc26929)

[1.3 Features and Functionalists: 7](#_Toc8569)

[2. System Architecture Document 9](#_Toc20602)

[2.1 Data Flow Diagram 9](#_Toc29215)

[2.2 Component Descriptions: 10](#_Toc7844)

[3. Requirements Document 13](#_Toc19555)

[3.1 Introduction 13](#_Toc12848)

[3.2 Resource Requirements: 13](#_Toc22754)

[3.3 Functional Requirements Of User: 13](#_Toc11504)

[3.4 Assumptions and Dependencies 14](#_Toc9994)

[4.Test Case Document 14](#_Toc10718)

[4.1. Introduction 15](#_Toc4648)

[4.2. Test Case Format 15](#_Toc31335)

[4.3. Test Cases 15](#_Toc17411)

[4.3.1 User Authentication 15](#_Toc28442)

[4.3.2 Admin Dashboard 17](#_Toc26948)

[4.3.3 Doctor Dashboard 21](#_Toc3817)

[4.3.4 Patient Dashboard 23](#_Toc3646)

[5.Design Document 25](#_Toc31070)

[5.1. Database Schema 25](#_Toc3796)

[5.1.1 User Model (Django's default User model) 25](#_Toc21350)

[5.1.2 Doctor Model 25](#_Toc6785)

[5.1.3 Patient Model 25](#_Toc23469)

[5.1.4 Appointment Model 25](#_Toc5598)

[5.1.5 PatientDischargeDetails Model 25](#_Toc27294)

[5.2. ER Diagrams 27](#_Toc11541)

[6. User Manual 28](#_Toc889)

[6.1 How to Install 28](#_Toc18207)

[6.1.1 Clone the Repository 28](#_Toc1835)

[6.1.2 Navigate to the Project Directory 28](#_Toc18368)

[6.1.3 Create and Activate a Virtual Environment 28](#_Toc24261)

[6.1.4 Install Required Packages: 28](#_Toc16031)

[6.1.5 Migrate the Database: 29](#_Toc25282)

[6.1.6 Create a Superuser 29](#_Toc21491)

[6.1.6 Run the Development Server 29](#_Toc11655)

[6.2 How to Use 29](#_Toc1929)

[Admin User 29](#_Toc15030)

[Doctor User 30](#_Toc15683)

[Patient User 31](#_Toc592)

[6.3 User Interface Guide 32](#_Toc10810)

[6.3.1 Login and Signup Pages 32](#_Toc30141)

[6.3.2 Admin Dashboard 32](#_Toc4125)

[6.3.3 Doctor Dashboard: 32](#_Toc13870)

[6.3.4 Patient Dashboard 33](#_Toc24728)

[6.4 Troubleshooting 33](#_Toc22263)

[6.4.1 Common Issues: 33](#_Toc28854)

[Debugging Tips: 34](#_Toc21458)

[Getting Help: 34](#_Toc25765)

[7. Project Report 35](#_Toc25132)

[Executive Summary 35](#_Toc32322)

[Detailed Project Description 35](#_Toc15854)

[1. Introduction 35](#_Toc3412)

[2. System Architecture 35](#_Toc21846)

[3. Modules 35](#_Toc30310)

[4. Database Design 36](#_Toc5345)

[5. User Interface 36](#_Toc11675)

[6. Implementation Details 36](#_Toc23391)

[Conclusion 37](#_Toc11503)

[Future Enhancements 37](#_Toc8086)

# Project Overview

### 1.1 Introduction

The Hospital Management System is a web application designed to manage hospital operations efficiently. The system is built using Django for the back-end and HTML, CSS, and JavaScript for the front-end. The application provides distinct functionalists for three types of users: Hospital Admin, Doctor, and Patient.

### ****1.2 Objectives:****

* To provide an intuitive platform for hospital administration, allowing for streamlined processes.
* To create a system where doctors can apply for positions and manage their patient appointments effectively.
* To enable patients to book appointments and manage their hospital interactions seamlessly.
* To automate billing and invoicing for discharged patients.

### ****1.3 Features and Functionalists:****

1. **Admin:**

· **Doctor Management:** Register, view, approve, reject, or delete doctors applying for positions within the hospital.

· **Patient Management:** Admit, view, approve, reject, and discharge patients (discharge patients when treatment is completed).

· **Invoice Management:** Generate and download PDF invoices based on medicine costs, room charges, doctor fees, and additional expenses.

· **Appointment Management:** View, book, and approve patient appointments (approve appointments requested by patients).

1. **Doctor:**

· **Job Application:** Apply for a job within the hospital and log in only after approval from the hospital admin.

· **Patient Management:** View details of patients assigned by the admin, including symptoms, names, and mobile numbers. Access a list of patients who have been discharged (by admin).

· **Appointment Management:** View appointments booked by the admin. Delete appointments after attending to the patient.

1. **Patient:**

· **Account Management:** Create an account to seek admission to the hospital and log in only after approval from the hospital admin.

· **Doctor Information:** View assigned doctor’s details, including specialization, mobile number, and address.

· **Appointment Management:** Check the status of booked appointments (pending/confirmed by admin). Book appointments (approval required from admin).

· **Invoice Management:** View and download PDF invoices only after being discharged by the admin.

# System Architecture Document

### Data Flow Diagram

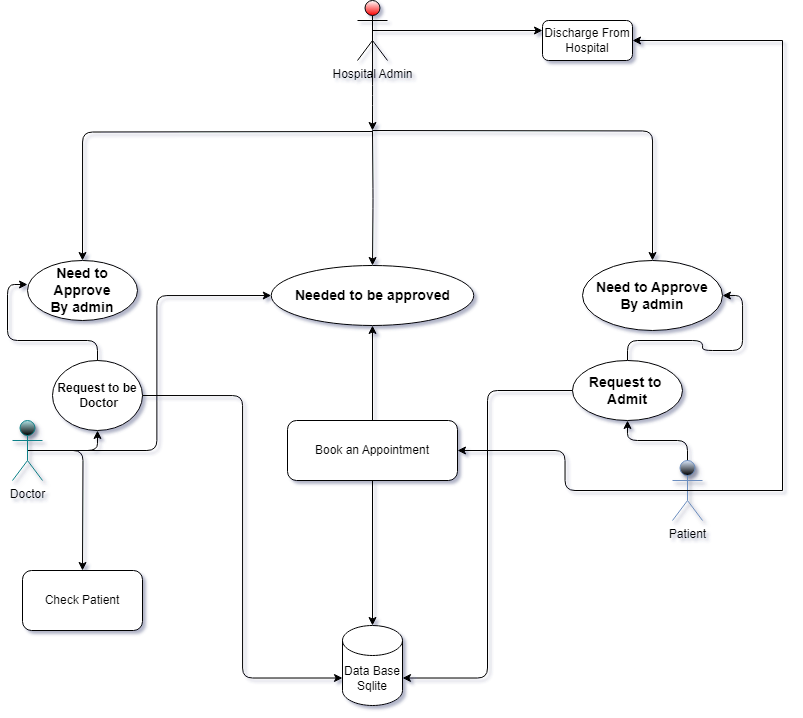


Fig: Data Flow diagram of Hospital Management System

### Component Descriptions:

#### 2.2.1 Front-End:

- **Technologies:** HTML, CSS, JavaScript

**- Description:** The Front end is responsible for providing the user interface for all types of users. It includes pages for user authentication, dashboards for admins, doctors, and patients, appointment booking, patient discharge, and invoice viewing/downloading.

**- Key Components:**

* + Login and Registration Forms
  + Admin Dashboard
  + Doctor Dashboard
  + Patient Dashboard
  + Appointment Booking System
  + Invoice Viewing and Downloading
  + Viewing the records
  + Managing the User

#### 2.2.2 Back-End:

* **Technology**: Django (Python)
* **Description:** The Back-end handles the business logic, database interactions, and user management. It processes requests from the Front-end, performs necessary computations, and interacts with the database to store and retrieve data.
* **Key Components:**
  + **User Management:**

1. Admin sign up & Login
2. Doctor sign up & Login
3. Patient sign up & Login
   * **Admin Management:**
     + Admin Dashboard View
     + Doctor Record view (Approve, Update, Delete, reject, add)
     + Patient Record view (Approve, Update, Delete, reject, add) With (discharge, Create Invoice )
     + Approve Appointment
   * **Doctor Management:**
     + Doctor Dashboard View
     + Patient view (view\_discharge, delete\_appointment, approve\_appointment )
     + Search bar
   * **Patient Management:**
     + Patient Dashboard View
     + Book Appointment with choice of (Doctor, Symptoms of Illness, Description )
     + Download Invoice

#### 2.2.2 Database :

* **Technology:** SQLite
* **Description:** The relational database stores all the data related to users (admins, doctors, patients), appointments, invoices, and other related entities.
* **Key Components:**
  + (Admin, Doctor, Patient) Tables are Linked with User Table
  + Appointment Table
  + Patient Discharge Table

#### 2.2.3 External Services:

· **PDF Generation Service:** Used for generating download-able invoices in PDF format. Libraries like Report Lab can be used for this purpose.

· **Security:**

* **Authentication:** Managed using Django's built-in authentication system, which includes user login, registration, and password management.
* **Authorization:** Role-based access control (RBAC) to ensure that only authorized users can access certain functionalists (e.g., only admins can approve doctors and patients).
* **Data Protection:** Implement SSL/TLS for secure data transmission, and follow best practices for data protection and privacy.

# Requirements Document

## 3.1 Introduction

This document outlines the requirements for the Hospital Management System. The system will manage the interactions between hospital admins, doctors, and patients, ensuring efficient administration of hospital activities.

## 3.2 Resource Requirements:

1. **Tools & Technology**:
2. Python
3. Django
4. Pillow, colorama, cryptography, pyparsing, xhtml2pdf, requests, qrcode, django-widget-tweaks
5. js, Css

## 3.3 Functional Requirements Of User:

**A. User Authentication**

* + 1. Admin, doctor, and patient signup.
    2. Login and logout functionalities.
    3. Redirect users to appropriate dashboards after login.

1. **Admin Dashboard**
   * + 1. View counts of doctors, patients, and appointments.
       2. Manage doctor and patient accounts (view, approve, reject, delete).
       3. Add new doctors and patients.
       4. Discharge patients and generate invoices.
       5. Manage appointments (view, approve, reject, add).
2. **Doctor Dashboard:**
   1. View patient details (symptoms, name, mobile).
   2. View discharged patients.
   3. View and manage appointments.
   4. Search for patients by symptoms or name.
3. **Patient Dashboard**
   1. View assigned doctor details (specialization, mobile, address).
   2. Book appointments with doctors.
   3. View appointment status.
   4. View and download invoices.
4. **Appointment Management**
   1. Patients can book appointments with doctors.
   2. Admin can approve or reject appointment requests.
   3. Doctors can view and delete their appointments.
5. **Invoice Generation**
   1. Admin can generate and download PDF invoices for discharged patients.
   2. Invoice includes medicine costs, room charges, doctor fees, and other charges.

## 3.4 Assumptions and Dependencies

1. The system will be developed using Django, HTML, CSS, and JavaScript.
2. The database used will be SQLite.
3. The system will use external services for email notifications and PDF generation.
4. Users will have access to the internet to use the system.
5. The system will be deployed on a web server with SSL/TLS enabled.

# 

# 4.Test Case Document

## 4.1. Introduction

This document outlines the test cases for the Hospital Management System. The purpose of these test cases is to ensure that all functionalities of the system work as expected and meet the specified requirements.

## 4.2. Test Case Format

Each test case is described using the following format:

* **Test Case ID:** Unique identifier for the test case.
* **Test Description:** Brief description of the test case.
* **Preconditions:** Conditions that must be met before the test case is executed.
* **Test Steps:** Step-by-step instructions to execute the test.
* **Expected Result:** The expected outcome of the test case.
* **Actual Result:** The actual outcome after executing the test case.
* **Status:** Pass/Fail based on whether the actual result matches the expected result.

## 4.3. Test Cases

### 4.3.1 User Authentication

#### # TC001

Test Case ID: TC001  
 **Test Description:** Admin Signup  
 **Preconditions:** None  
 **Test Steps:**

1. Navigate to the admin signup page.
2. Fill in the required details (username, password, email, etc.).
3. Submit the signup form.  
   **Expected Result:** Admin account is created, and the user is redirected to the admin login page.

#### # TC002

**Test Case ID:** TC002  
**Test Description:** Admin Login  
**Preconditions:** Admin account exists  
**Test Steps:**

1. Navigate to the admin login page.
2. Enter valid username and password.
3. Submit the login form.  
   **Expected Result:** Admin is logged in and redirected to the admin dashboard.

#### # TC003

**Test Case ID:** TC003  
**Test Description:** Doctor Signup  
**Preconditions:** None  
**Test Steps:**

1. Navigate to the doctor signup page.
2. Fill in the required details.
3. Submit the signup form.  
   **Expected Result:** Doctor account request is created, and the user is redirected to the doctor login page with a pending approval status.

#### # TC004

**Test Case ID:** TC004  
**Test Description:** Doctor Login  
**Preconditions:** Doctor account exists and is approved by admin  
**Test Steps:**

1. Navigate to the doctor login page.
2. Enter valid username and password.
3. Submit the login form.  
   **Expected Result:** Doctor is logged in and redirected to the doctor dashboard.

#### # TC005

**Test Case ID:** TC005  
**Test Description:** Patient Signup  
**Preconditions:** None  
**Test Steps:**

1. Navigate to the patient signup page.
2. Fill in the required details.
3. Submit the signup form.  
   **Expected Result:** Patient account request is created, and the user is redirected to the patient login page with a pending approval status.

#### # TC006

**Test Case ID:** TC006  
**Test Description:** Patient Login  
**Preconditions:** Patient account exists and is approved by admin  
**Test Steps:**

1. Navigate to the patient login page.
2. Enter valid username and password.
3. Submit the login form.  
   **Expected Result:** Patient is logged in and redirected to the patient dashboard.

### 4.3.2 Admin Dashboard

#### # TC007

**Test Case ID:** TC007  
**Test Description:** View Doctor List  
**Preconditions:** Admin is logged in  
**Test Steps:**

1. Navigate to the admin dashboard.
2. Click on the "Manage Doctors" link.  
   **Expected Result:** A list of all doctors (approved and pending) is displayed.

#### # TC008

**Test Case ID:** TC008  
**Test Description:** Approve Doctor  
**Preconditions:** Admin is logged in, and there are pending doctor requests  
**Test Steps:**

1. Navigate to the "Approve Doctors" page.
2. Click on the approve button next to a pending doctor.  
   **Expected Result:** The doctor’s status changes to approved.

#### # TC009

**Test Case ID:** TC009  
**Test Description:** Reject Doctor  
**Preconditions:** Admin is logged in, and there are pending doctor requests  
**Test Steps:**

1. Navigate to the "Approve Doctors" page.
2. Click on the reject button next to a pending doctor.  
   **Expected Result:** The doctor’s account request is rejected, and the doctor is removed from the list.

#### # TC010

**Test Case ID:** TC010  
**Test Description:** Add New Doctor  
**Preconditions:** Admin is logged in  
**Test Steps:**

1. Navigate to the "Add Doctor" page.
2. Fill in the required details.
3. Submit the form.  
   **Expected Result:** A new doctor is added to the hospital, and the doctor’s account is created.

#### # TC011

**Test Case ID:** TC011  
**Test Description:** Delete Doctor  
**Preconditions:** Admin is logged in, and there are approved doctors  
**Test Steps:**

1. Navigate to the "Manage Doctors" page.
2. Click on the delete button next to a doctor.  
   **Expected Result:** The doctor is deleted from the hospital, and the doctor’s account is removed.

#### # TC012

**Test Case ID:** TC012  
**Test Description:** View Patient List  
**Preconditions:** Admin is logged in  
**Test Steps:**

1. Navigate to the admin dashboard.
2. Click on the "Manage Patients" link.  
   **Expected Result:** A list of all patients (approved and pending) is displayed.

#### # TC013

**Test Case ID:** TC013  
**Test Description:** Approve Patient  
**Preconditions:** Admin is logged in, and there are pending patient requests  
**Test Steps:**

1. Navigate to the "Approve Patients" page.
2. Click on the approve button next to a pending patient.  
   **Expected Result:** The patient’s status changes to approved.

#### # TC014

**Test Case ID:** TC014  
**Test Description:** Reject Patient  
**Preconditions:** Admin is logged in, and there are pending patient requests  
**Test Steps:**

1. Navigate to the "Approve Patients" page.
2. Click on the reject button next to a pending patient.  
   **Expected Result:** The patient’s account request is rejected, and the patient is removed from the list.

#### # TC015

**Test Case ID:** TC015  
**Test Description:** Add New Patient  
**Preconditions:** Admin is logged in  
**Test Steps:**

1. Navigate to the "Add Patient" page.
2. Fill in the required details.
3. Submit the form.  
   **Expected Result:** A new patient is added to the hospital, and the patient’s account is created.

#### # TC016

**Test Case ID:** TC016  
**Test Description:** Delete Patient  
**Preconditions:** Admin is logged in, and there are approved patients  
**Test Steps:**

1. Navigate to the "Manage Patients" page.
2. Click on the delete button next to a patient.  
   **Expected Result:** The patient is deleted from the hospital, and the patient’s account is removed.

#### # TC017

**Test Case ID:** TC017  
**Test Description:** Discharge Patient  
**Preconditions:** Admin is logged in, and there are patients ready for discharge  
**Test Steps:**

1. Navigate to the "Discharge Patients" page.
2. Click on the discharge button next to a patient.
3. Calculate and confirm the discharge expenses.  
   **Expected Result:** The patient is discharged, and a final bill is generated.

#### # TC018

**Test Case ID:** TC018  
**Test Description:** Download Invoice PDF  
**Preconditions:** Admin is logged in, and there are discharged patients with generated invoices  
**Test Steps:**

1. Navigate to the "Discharged Patients" page.
2. Click on the download button next to a patient's invoice.  
   **Expected Result:** The invoice is downloaded as a PDF file.

#### # TC019

**Test Case ID:** TC019  
**Test Description:** View Appointment List  
**Preconditions:** Admin is logged in  
**Test Steps:**

1. Navigate to the admin dashboard.
2. Click on the "Manage Appointments" link.  
   **Expected Result:** A list of all appointments (approved and pending) is displayed.

#### # TC020

**Test Case ID:** TC020  
**Test Description:** Approve Appointment  
**Preconditions:** Admin is logged in, and there are pending appointment requests  
**Test Steps:**

1. Navigate to the "Approve Appointments" page.
2. Click on the approve button next to a pending appointment.  
   **Expected Result:** The appointment’s status changes to approved.

#### # TC021

**Test Case ID:** TC021  
**Test Description:** Reject Appointment  
**Preconditions:** Admin is logged in, and there are pending appointment requests  
**Test Steps:**

1. Navigate to the "Approve Appointments" page.
2. Click on the reject button next to a pending appointment.  
   **Expected Result:** The appointment request is rejected, and the appointment is removed from the list.

#### # TC022

**Test Case ID:** TC022  
**Test Description:** Add New Appointment  
**Preconditions:** Admin is logged in  
**Test Steps:**

1. Navigate to the "Add Appointment" page.
2. Fill in the required details.
3. Submit the form.  
   **Expected Result:** A new appointment is added to the hospital's schedule.

#### # TC023

**Test Case ID:** TC023  
**Test Description:** Delete Appointment  
**Preconditions:** Admin is logged in, and there are approved appointments  
**Test Steps:**

1. Navigate to the "Manage Appointments" page.
2. Click on the delete button next to an appointment.  
   **Expected Result:** The appointment is deleted from the hospital's schedule.

### 4.3.3 Doctor Dashboard

#### # TC024

**Test Case ID:** TC024  
**Test Description:** View Assigned Patients  
**Preconditions:** Doctor is logged in  
**Test Steps:**

1. Navigate to the doctor dashboard.
2. Click on the "Manage Patients" link.  
   **Expected Result:** A list of patients assigned to the doctor is displayed.

#### # TC025

**Test Case ID:** TC025  
**Test Description:** Search Patient by Symptoms or Name  
**Preconditions:** Doctor is logged in, and there are patients assigned to the doctor  
**Test Steps:**

1. Navigate to the "Search Patients" page.
2. Enter symptoms or patient's name in the search bar.
3. Submit the search form.  
   **Expected Result:** A list of patients matching the search criteria is displayed.

#### # TC026

**Test Case ID:** TC026  
**Test Description:** View Discharged Patients  
**Preconditions:** Doctor is logged in, and there are discharged patients  
**Test Steps:**

1. Navigate to the "View Discharged Patients" page.  
   **Expected Result:** A list of patients discharged by the doctor is displayed.

#### # TC027

**Test Case ID:** TC027  
**Test Description:** View Appointments  
**Preconditions:** Doctor is logged in  
**Test Steps:**

1. Navigate to the doctor dashboard.
2. Click on the "Manage Appointments" link.  
   **Expected Result:** A list of appointments assigned to the doctor is displayed.

#### # TC028

**Test Case ID:** TC028  
**Test Description:** Delete Appointment  
**Preconditions:** Doctor is logged in, and there are appointments assigned to the doctor  
**Test Steps:**

1. Navigate to the "Manage Appointments" page.
2. Click on the delete button next to an appointment.  
   **Expected Result:** The appointment is deleted from the doctor's schedule.

### 4.3.4 Patient Dashboard

#### # TC029

**Test Case ID:** TC029  
**Test Description:** View Assigned Doctor Details  
**Preconditions:** Patient is logged in  
**Test Steps:**

1. Navigate to the patient dashboard.
2. Click on the "View Doctor" link.  
   **Expected Result:** Details of the assigned doctor (specialization, mobile, address) are displayed.

#### # TC030

**Test Case ID:** TC030  
**Test Description:** Book Appointment  
**Preconditions:** Patient is logged in  
**Test Steps:**

1. Navigate to the "Book Appointment" page.
2. Fill in the required details (date, time, doctor, symptoms).
3. Submit the form.  
   **Expected Result:** The appointment request is created and awaits admin approval.

#### # TC031

**Test Case ID:** TC031  
**Test Description:** View Appointment Status  
**Preconditions:** Patient is logged in, and there are booked appointments  
**Test Steps:**

1. Navigate to the "View Appointments" page.  
   **Expected Result:** A list of booked appointments with their status (pending/confirmed) is displayed.

#### # TC032

**Test Case ID:** TC032  
**Test Description:** Download Invoice PDF  
**Preconditions:** Patient is logged in, and the patient has been discharged with a generated invoice  
**Test Steps:**

1. Navigate to the "View Invoice" page.
2. Click on the download button next to the invoice.  
   **Expected Result:** The invoice is downloaded as a PDF file.

# 5.Design Document

## 5.1. Database Schema

### 5.1.1 User Model (Django's default User model)

* username
* password
* first\_name
* last\_name
* email

### 5.1.2 Doctor Model

* User\_id: OneToOneField(User)
* profile\_pic: ImageField
* address: CharField
* mobile: CharField
* department: CharField (choices: Cardiologist, Neurologist, Orthopedic, etc.)
* status: BooleanField

### 5.1.3 Patient Model

* user: OneToOneField(User)
* profile\_pic: ImageField
* address: CharField
* mobile: CharField
* symptoms: CharField
* assignedDoctorId: PositiveIntegerField
* admitDate: DateField
* status: BooleanField

### 5.1.4 Appointment Model

* patientId: PositiveIntegerField
* doctorId: PositiveIntegerField
* patientName: CharField
* doctorName: CharField
* appointmentDate: DateField
* description: TextField
* status: BooleanField

### 5.1.5 PatientDischargeDetails Model

* patientId: PositiveIntegerField
* patientName: CharField
* assignedDoctorName: CharField
* address: CharField
* mobile: CharField
* symptoms: CharField
* admitDate: DateField
* releaseDate: DateField
* daySpent: PositiveIntegerField
* roomCharge: PositiveIntegerField
* medicineCost: PositiveIntegerField
* doctorFee: PositiveIntegerField
* otherCharge: PositiveIntegerField
* total: PositiveIntegerField

## 5.2. ER Diagrams

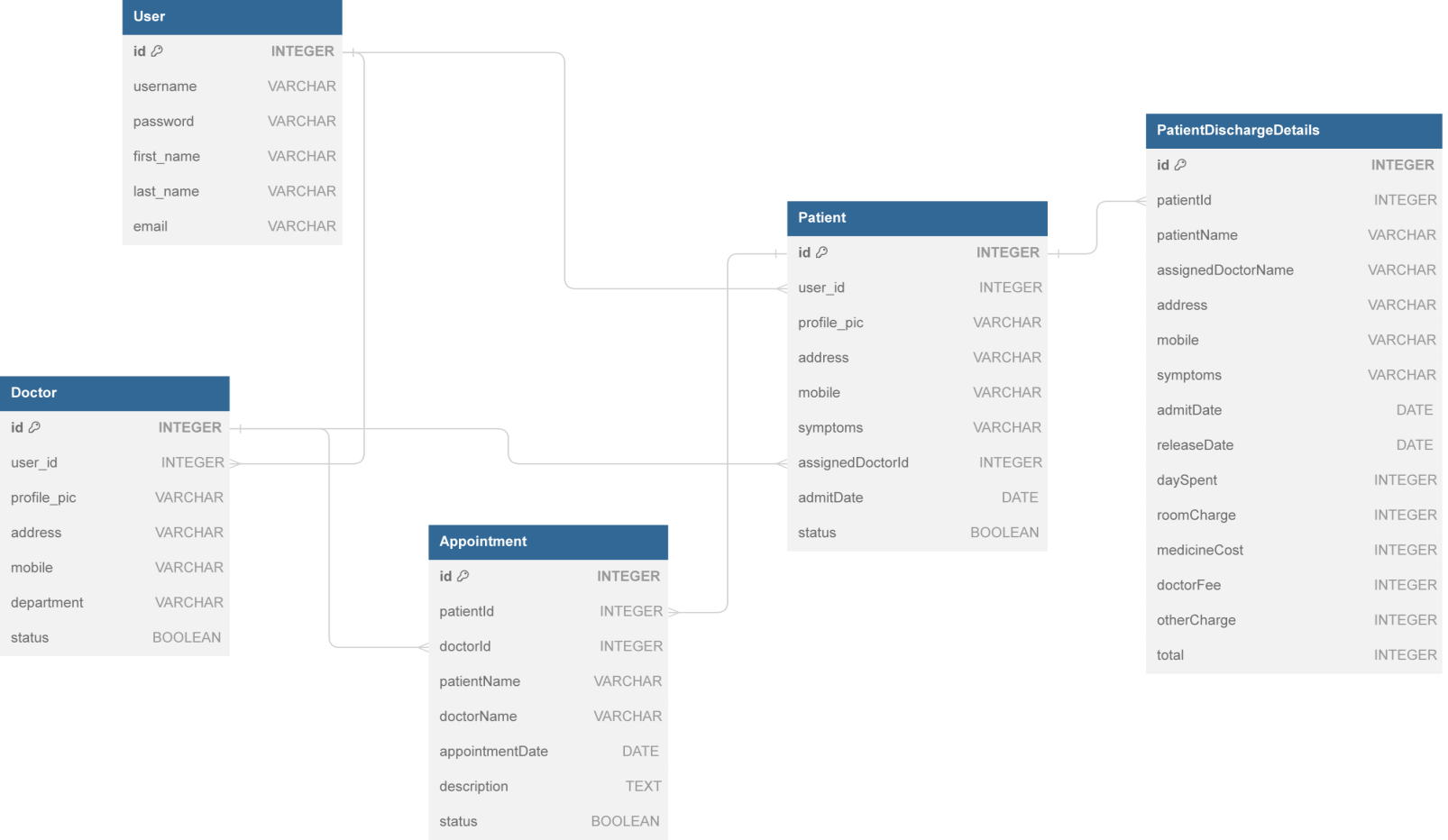


Fig: ER diagram of Hospital Management System

# User Manual

## 6.1 How to Install

### 6.1.1 Clone the Repository

* Open your terminal or command prompt.
* Navigate to the directory where you want to clone the repository.
* Run the following command:

`git clone https://github.com/0luv69/Hospital-Mangement-System.git`

### 6.1.2 Navigate to the Project Directory

* Change your directory to the newly cloned project:

`cd hospital-management-system`

### 6.1.3 Create and Activate a Virtual Environment

1. `python -m venv env`

#### In Windows :

1. `.\env\Scripts\activate`

#### In Mac/ Linux :

1. `source env/bin/activate`

### 6.1.4 Install Required Packages:

* Install the required Python packages:

1. ~pip install -r requirements.txt~

### 6.1.5 Mi****grate the Database:****

* Run the following commands to apply the database migrations

`python manage.py makemigrations`

` python manage.py migrate`

### 6.1.6 Create a Superuser

* Create an admin account to access the admin panel:

`python manage.py createsuperuser~

### 6.1.6 Run the Development Server

* Start the Django development server

`python manage.py runserver`

Note: Open your web browser and navigate to http://127.0.0.1:8000 to access the application.

& to off the server you can press Ctrl+z

## 6.2 How to Use

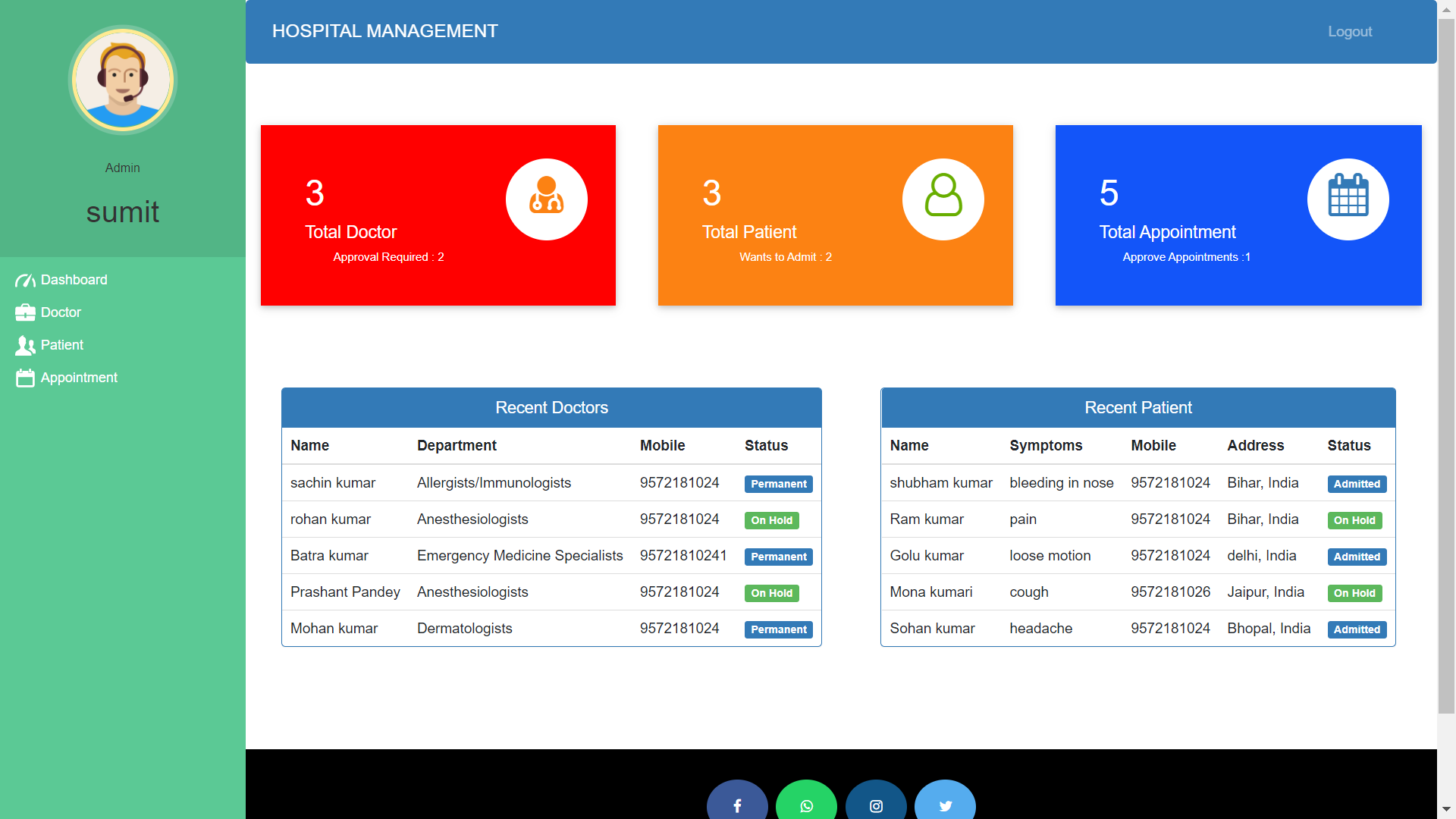
### Admin User

#### Signup/Login:

* + 1. Navigate to the admin signup page to create an admin account.
    2. Use the admin login page to log in with the created admin account.

#### Dashboard:

* + 1. Manage doctors, patients, and appointments.
    2. Approve or reject doctor and patient requests.
    3. Discharge patients and generate invoices.
    4. Approve or reject appointment requests



##### Fig : Dashboard of Admin User

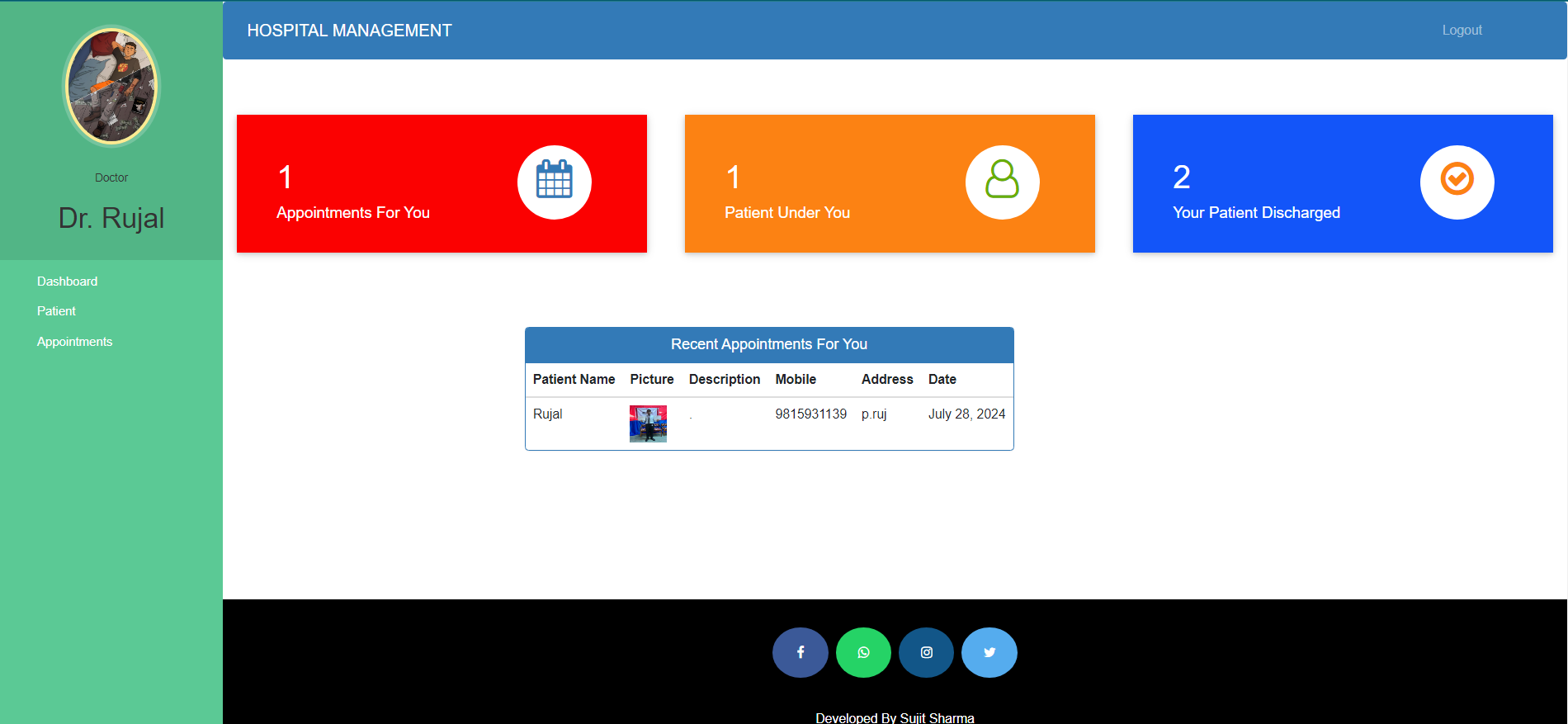
### Doctor User

#### Signup/Login:

* + 1. Navigate to the doctor signup page to apply for a job.
    2. Use the doctor login page to log in after approval by the admin.

#### Dashboard:

* + 1. View assigned patients and their details.
    2. Manage appointments.
    3. View discharged patients.



##### Fig : Dashboard of Doctor User

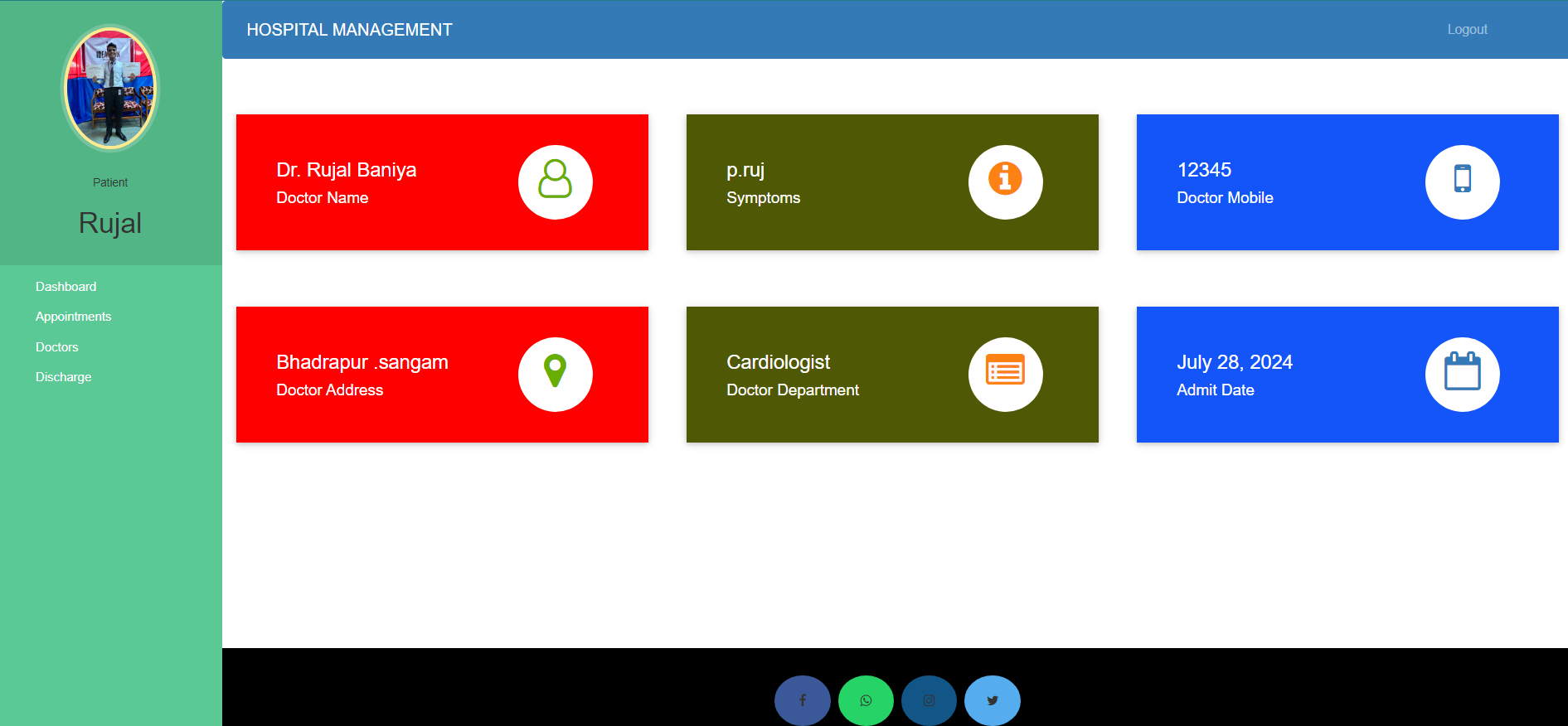
### Patient User

#### Signup/Login:

* + 1. Navigate to the patient signup page to create an account.
    2. Use the patient login page to log in after approval by the admin.

#### Dashboard:

* + 1. View assigned doctor details.
    2. Book and manage appointments.
    3. View and download invoices after discharge.



##### Fig : Dashboard of Patient User

## 6.3 User Interface Guide

### 6.3.1 Login and Signup Pages

1. Separate login and signup forms for Admin, Doctor, and Patient.
2. Simple and intuitive forms for user input with validation messages.

### 6.3.2 Admin Dashboard

#### 6.3.2.1 Manage Doctors:

* + 1. View, approve, reject, and delete doctors.
    2. Add and update doctor details.

#### 6.3.2.2 Manage Patients:

* + 1. View, approve, reject, discharge, and delete patients.
    2. Add and update patient details.

#### 6.3.2.3 Manage Appointments:

* + 1. View, approve, reject, and manage appointments.

#### 6.3.2.4 Generate Invoices:

* + 1. Calculate charges and generate downloadable invoices.

### 6.3.3 Doctor Dashboard:

#### 6.3.3.1 View Patients:

View list of assigned patients with details.

#### 6.3.3.2 Manage Appointments:

View and manage appointments.

Delete attended appointments.

#### 6.3.3.3 View Discharged Patients:

View list of discharged patients with details.

### 6.3.4 Patient Dashboard

#### 6.3.4.1 View Doctor Details:

* + 1. View assigned doctor's contact and specialization details.

#### 6.3.4.2 Book Appointments:

* + 1. Book appointments by specifying symptoms and selecting doctors.
    2. View appointment status (pending/confirmed).

#### 6.3.4.3 View Invoices:

* + 1. View and download discharge invoices.

## 6.4 Troubleshooting

### 6.4.1 Common Issues:

#### Server Not Starting:

* + 1. Ensure you have activated the virtual environment.
    2. Check if all required packages are installed.
    3. Verify database migrations are applied correctly.

#### Database Errors:

* + 1. Ensure database configurations are correct in settings.py.
    2. Run migrations again using python manage.py migrate.

#### Page Not Found (404):

* + 1. Verify the URL patterns in urls.py.
    2. Ensure you are navigating to the correct URL.

### Debugging Tips:

1. Use Django's debug mode by setting DEBUG = True in settings.py to get detailed error messages.
2. Check the Django logs for any runtime errors or warnings.
3. Use print statements or Django's logging module to trace issues in the code.

### Getting Help:

1. Refer to the Django documentation for detailed guides and references.
2. Search for solutions on Stack Overflow or other Django communities.
3. Contact the project maintainer or contributor for specific issues related to the project.

# Project Report

## Executive Summary

The Hospital Management System is a comprehensive web application developed using Django, HTML, CSS, and JavaScript. The system aims to streamline the administrative and operational tasks of a hospital by providing distinct interfaces for three user types: Admin, Doctor, and Patient. Admins manage doctor and patient records, appointments, and invoices. Doctors view and manage their patients and appointments, while patients can book appointments and view their medical details. The system enhances efficiency, reduces paperwork, and improves patient care.

## Detailed Project Description

### ****1. Introduction****

* **Purpose**: To develop a system that efficiently manages hospital operations and improves patient care.
* **Scope**: The system covers user authentication, patient and doctor management, appointment scheduling, and invoice generation.

### ****2. System Architecture****

#### ****High-Level Architecture****:

The system follows an MVC (Model-View-Controller) architecture, ensuring separation of concerns and facilitating maintainability.

#### Components:

* + **Frontend**: HTML, CSS, JavaScript for creating responsive and user-friendly interfaces.
  + **Backend**: Django framework for handling business logic and data management.
  + **Database**: SQLite for storing user, patient, doctor, and appointment information.

### 3. Modules

#### Admin Module:

* + Signup/Login without approval.
  + Manage doctors (view, approve, reject, delete).
  + Manage patients (view, approve, reject, discharge).
  + Generate and download invoices.
  + Manage appointments (view, approve, reject).

#### Doctor Module:

* + Apply for a job and login upon admin approval.
  + View assigned patients' details.
  + Manage appointments.
  + View discharged patients.

#### Patient Module:

* + Create an account and login upon admin approval.
  + View assigned doctor's details.
  + Book and manage appointments.
  + View and download invoices upon discharge.

### 4. Database Design

* **User Model**: Utilizes Django's default user model for authentication.
* **Doctor Model**: Stores doctor-specific information like profile picture, address, mobile, department, and status.
* **Patient Model**: Stores patient-specific information like profile picture, address, mobile, symptoms, assigned doctor ID, admit date, and status.
* **Appointment Model**: Manages appointment details between doctors and patients.
* **PatientDischargeDetails Model**: Stores details about discharged patients, including charges and total expenses.

### 5. User Interface

* **Login/Signup Pages**: Separate interfaces for Admin, Doctor, and Patient with user-friendly forms.
* **Dashboards**:
  + **Admin Dashboard**: Overview of doctors, patients, and appointments with management functionalities.
  + **Doctor Dashboard**: Overview of assigned patients and appointments.
  + **Patient Dashboard**: Overview of assigned doctors and appointment statuses.

### 6. Implementation Details

* **Technologies Used**:
  + Python and Django for backend development.
  + HTML, CSS, JavaScript for frontend development.
  + Additional libraries like Pillow, colorama, cryptography, pyparsing, xhtml2pdf, requests, qrcode, django-widget-tweaks.

## Conclusion

The Hospital Management System successfully meets its objectives of streamlining hospital operations and improving patient care. By providing distinct roles and functionalists for admins, doctors, and patients, the system ensures efficient management of hospital resources and patient information. The intuitive user interfaces and robust backend architecture contribute to a seamless user experience and reliable system performance.

## Future Enhancements

1. **Enhanced Reporting**: Adding more comprehensive reporting features for hospital admin to monitor overall hospital performance.
2. **Patient History**: Implementing a detailed patient history feature that allows doctors and patients to view past medical records and treatments.
3. **Integration with External Systems**: Integrating the system with external healthcare systems for seamless data exchange.
4. **Mobile Application**: Developing a mobile application to provide users with access to the system on-the-go.
5. **AI-Powered Features**: Incorporating AI to predict patient conditions based on symptoms and historical data, and to optimize doctor-patient matching.