PATIENT RECORDS MANAGEMENT SYSTEM

NAME: MEGHA BHAT | SRN: PES1UG22CS344

NAME: MAHIMA A | SRN: PES1UG22CS322

SECTION: F

INTRODUCTION:

Our project **Patient Records Management System** includes registration of patients, storing their details into the system, and also booking and cancelling their appointments with doctors. Our software has the facility to give a unique id for every patient and stores the details of every patient and the doctors/ admins automatically. User can search for a doctor and the details of a patient using the id. This system can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can edit or delete data into and from the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast.

IMPLEMENTATION:

Backend (MySQL):

- **Database:** MySQL is used to store all data, such as owners, tenants, employees, rooms, payments, and complaints.
- **Data Links:** The database has connections between tables to keep data accurate, like linking rooms to owners and payments to tenants.

Backend Logic (Flask and PyMySQL):

- **Main Routing (hello.py)**: Flask runs the backend logic, handles user requests, connects to the database using PyMySQL, and updates or retrieves data.
- **User Sessions**: Sessions ensure only logged-in users can access their dashboards safely.
- **Forms**: WTForms is used to create and validate forms for user input.

Frontend (HTML Templates):

- **Templates**: HTML templates show the web pages users see, filled with data from the backend.

- **User Dashboards**: There are separate login pages and dashboards for doctors, admins and patients.
- **Feedback Messages**: Flash messages notify users about their actions (e.g., login success, errors).

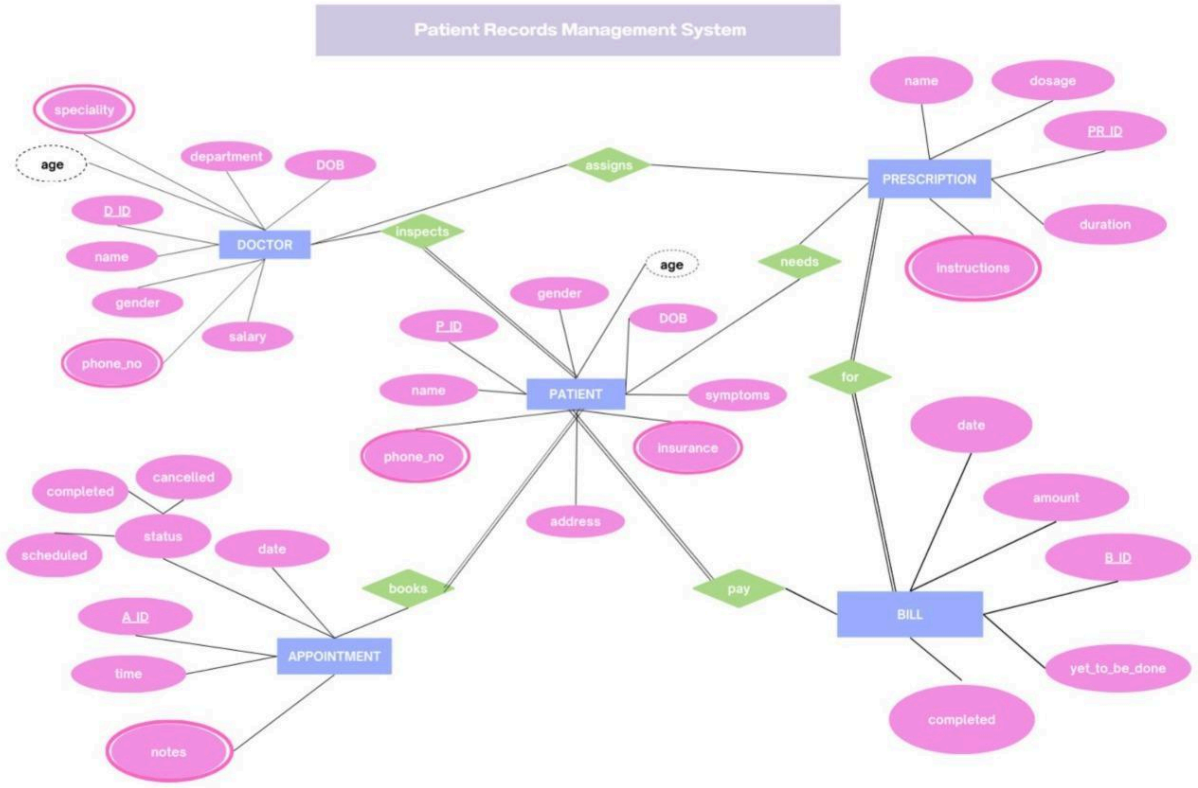
RUNNING THE CODE FILE:

In command prompt (terminal) of the same directory of the project-

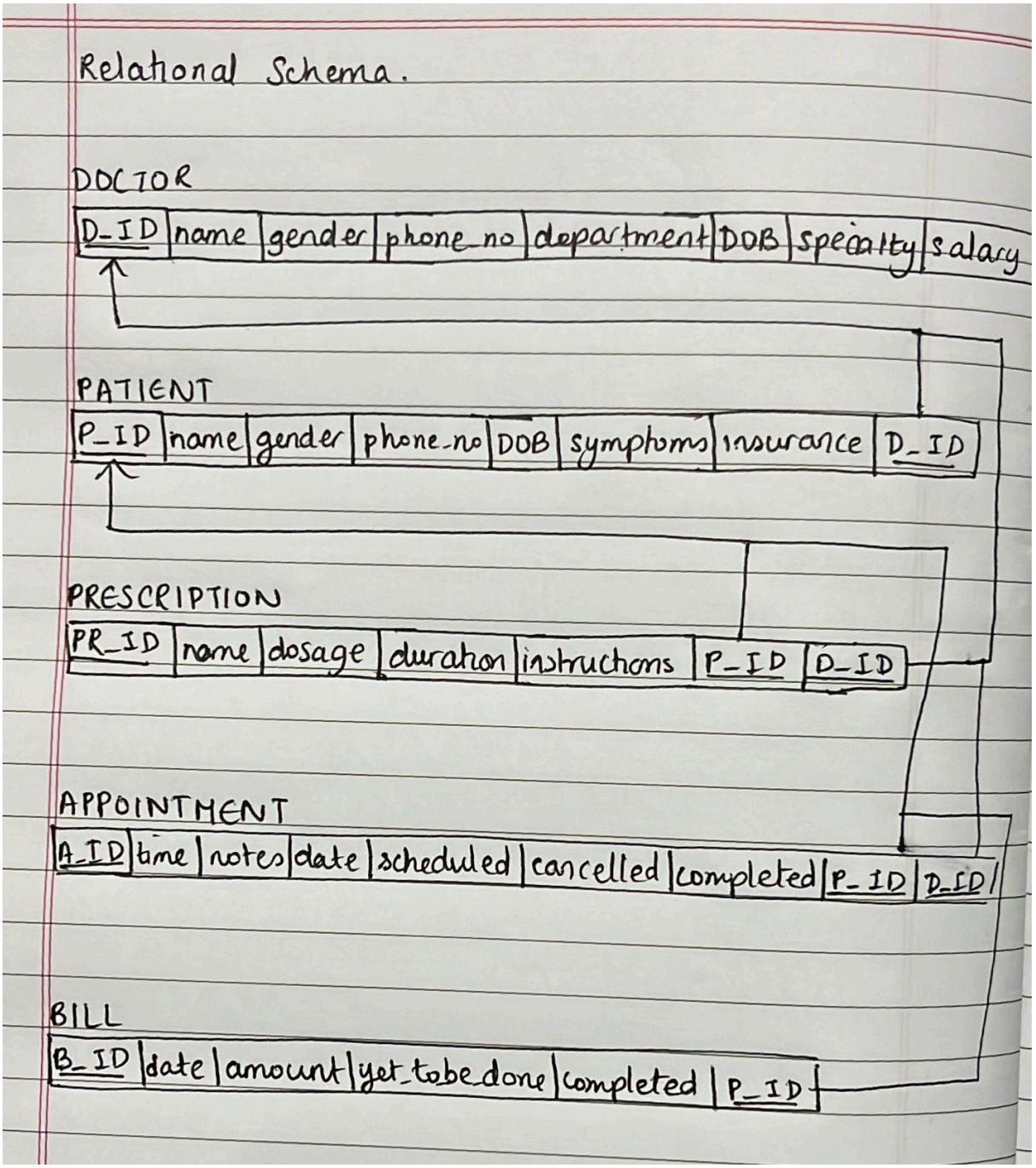
Python app.py

This can be run even on VSCode or any other IDE (integrated Development Environment) that supports Python programming language.

ER DIAGRAM:



RELATIONAL SCHEMA:



DATABASE CREATION:

```
CREATE DATABASE IF NOT EXISTS HospitalDB;
```

USE HospitalDB;

```sql
-- Create Database if it doesn't exist
CREATE DATABASE IF NOT EXISTS HospitalDB;
USE HospitalDB;
```

TABLES:

1) Patient Table (Auto Increment P_ID)

CREATE TABLE IF NOT EXISTS Patient (

P_ID INT AUTO_INCREMENT PRIMARY KEY, -- Auto increment P_ID

name VARCHAR(50) NOT NULL,

gender VARCHAR(10),

DOB DATE,

Age INT,

symptoms TEXT, -- Symptoms column added

address VARCHAR(128),

phone_no VARCHAR(15),

blood_group VARCHAR(5),

insurance VARCHAR(50),

email VARCHAR(255),

password VARCHAR(255)

);

```sql
-- Patient Table (Auto Increment P_ID)
CREATE TABLE IF NOT EXISTS Patient (
    P_ID INT AUTO_INCREMENT PRIMARY KEY,  -- Auto increment P_ID
    name VARCHAR(50) NOT NULL,
    gender VARCHAR(10),
    DOB DATE,
    Age INT,
    symptoms TEXT,  -- Symptoms column added
    address VARCHAR(128),
    phone_no VARCHAR(15),
    blood_group VARCHAR(5),
    insurance VARCHAR(50),
    email VARCHAR(255),
    password VARCHAR(255)
);
```

2) Doctors Table (AUTO_INCREMENT for D_ID)

CREATE TABLE IF NOT EXISTS doctors (

D_ID INT AUTO_INCREMENT PRIMARY KEY, -- Ensure D_ID is AUTO_INCREMENT

name VARCHAR(50) NOT NULL,

phone_no VARCHAR(15),

email VARCHAR(255),

job_title VARCHAR(100),

degree VARCHAR(100),

year INT,

employer VARCHAR(255),

password VARCHAR(255),

specialty VARCHAR(50),

DOB DATE,

Age INT,

salary DECIMAL(10, 2)

);

```
-- Doctors Table with AUTO_INCREMENT for D_ID
CREATE TABLE IF NOT EXISTS doctors (
    D_ID INT AUTO_INCREMENT PRIMARY KEY,  -- Ensure D_ID is AUTO_INCREMENT
    name VARCHAR(50) NOT NULL,
    phone_no VARCHAR(15),
    email VARCHAR(255),
    job_title VARCHAR(100),
    degree VARCHAR(100),
    year INT,
    employer VARCHAR(255),
    password VARCHAR(255),
    specialty VARCHAR(50),
    DOB DATE,
    Age INT,
    salary DECIMAL(10, 2)
);
```

3) Administrator Table

CREATE TABLE IF NOT EXISTS Administrator (

id INT AUTO_INCREMENT PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL

);

```sql
-- Administrator Table
CREATE TABLE IF NOT EXISTS Administrator (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL
);

-- Initial Data for Administrator
INSERT INTO Administrator (username, password)
VALUES ('meg', 'meg')
ON DUPLICATE KEY UPDATE password = 'meg';
```

4) Appointment Table

CREATE TABLE IF NOT EXISTS Appointment (

A_ID INT AUTO_INCREMENT PRIMARY KEY, -- Auto-increment integer ID

time DATETIME NOT NULL, -- Date and Time of Appointment

notes TEXT, -- Additional Notes for Appointment

status ENUM('Scheduled', 'Concluded', 'Completed', 'Cancelled') DEFAULT 'Scheduled', -- Appointment Status

P_ID INT NOT NULL, -- Patient ID

D_ID INT NOT NULL, -- Doctor ID

FOREIGN KEY (P_ID) REFERENCES Patient(P_ID) ON DELETE CASCADE, -- Ensure relational integrity

FOREIGN KEY (D_ID) REFERENCES doctors(D_ID) ON DELETE CASCADE -- Ensure relational integrity

);

```sql
-- Appointment Table
CREATE TABLE IF NOT EXISTS Appointment (
    A_ID INT AUTO_INCREMENT PRIMARY KEY,  -- Auto-increment integer ID
    time DATETIME NOT NULL,               -- Date and Time of Appointment
    notes TEXT,                           -- Additional Notes for Appointment
    status ENUM('Scheduled', 'Concluded', 'Completed', 'Cancelled') DEFAULT 'Scheduled', -- Appointment Status
    P_ID INT NOT NULL,                    -- Patient ID
    D_ID INT NOT NULL,                    -- Doctor ID
    FOREIGN KEY (P_ID) REFERENCES Patient(P_ID) ON DELETE CASCADE, -- Ensure relational integrity
    FOREIGN KEY (D_ID) REFERENCES doctors(D_ID) ON DELETE CASCADE  -- Ensure relational integrity
);
```

5) Prescription Table

CREATE TABLE IF NOT EXISTS Prescription (

Pr_ID VARCHAR(20) PRIMARY KEY,

name VARCHAR(50),

dosage VARCHAR(50),

instructions TEXT,

duration INT

);

```sql
-- Prescription Table
CREATE TABLE IF NOT EXISTS Prescription (
  Pr_ID VARCHAR(20) PRIMARY KEY,
  name VARCHAR(50),
  dosage VARCHAR(50),
  instructions TEXT,
  duration INT
);
```

6) Bill Table

CREATE TABLE IF NOT EXISTS Bill (

B_ID VARCHAR(20) PRIMARY KEY,

date DATE,

amount DECIMAL(10, 2),

payment_status ENUM('Completed', 'Yet-to-be-done')

);

```sql
-- Bill Table
CREATE TABLE IF NOT EXISTS Bill (
  B_ID VARCHAR(20) PRIMARY KEY,
  date DATE,
  amount DECIMAL(10, 2),
  payment_status ENUM('Completed', 'Yet-to-be-done')
);
```

TRIGGERS:

1) Patient Table: Before Insert to Update Age Based on DOB

DROP TRIGGER IF EXISTS before_insert_patient;

```
DELIMITER //

CREATE TRIGGER before_insert_patient

BEFORE INSERT ON Patient

FOR EACH ROW

BEGIN

 IF NEW.DOB IS NOT NULL THEN

  SET NEW.Age = TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE());

 END IF;

END;

//

DELIMITER ;
```

```sql
-- Triggers for Patient Table: Before Insert to Update Age Based on DOB
DROP TRIGGER IF EXISTS before_insert_patient;
DELIMITER //
CREATE TRIGGER before_insert_patient
BEFORE INSERT ON Patient
FOR EACH ROW
BEGIN
  IF NEW.DOB IS NOT NULL THEN
    SET NEW.Age = TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE());
  END IF;
END;
//
DELIMITER ;
```

2) To ensure that an appointment cannot have conflicting statuses or be scheduled in the past.

```
DELIMITER //

CREATE TRIGGER before_insert_appointment

BEFORE INSERT ON Appointment

FOR EACH ROW

BEGIN

  -- Ensure the appointment date is in the future

  IF NEW.time <= NOW() THEN

    SIGNAL SQLSTATE '45000'

    SET MESSAGE_TEXT = 'Appointment cannot be scheduled in the past.';

  END IF;
```

END;

//

DELIMITER ;

```
-- Trigger to ensure that an appointment cannot have conflicting statuses or be scheduled in the past.
DELIMITER //
CREATE TRIGGER before_insert_appointment
BEFORE INSERT ON Appointment
FOR EACH ROW
BEGIN
    -- Ensure the appointment date is in the future
    IF NEW.time <= NOW() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Appointment cannot be scheduled in the past.';
    END IF;
END;
//
DELIMITER ;
```

PROCEDURES:

1) To fetch all appointments for a specific doctor, including details about the patient.

```
DELIMITER //

CREATE PROCEDURE GetAppointmentsByDoctor (IN doctor_id INT)

BEGIN

  SELECT

    a.A_ID AS Appointment_ID,

    a.time AS Appointment_Time,

    a.status AS Appointment_Status,

    p.name AS Patient_Name,

    p.phone_no AS Patient_Phone,

    p.email AS Patient_Email,

    a.notes AS Appointment_Notes

  FROM

    Appointment a

  INNER JOIN

    Patient p ON a.P_ID = p.P_ID

  WHERE

    a.D_ID = doctor_id

  ORDER BY
```

a.time ASC;

END;

//

DELIMITER ;

```
-- Procedure to fetch all appointments for a specific doctor, including details about the patient.
DELIMITER //
CREATE PROCEDURE GetAppointmentsByDoctor (IN doctor_id INT)
BEGIN
    SELECT
        a.A_ID AS Appointment_ID,
        a.time AS Appointment_Time,
        a.status AS Appointment_Status,
        p.name AS Patient_Name,
        p.phone_no AS Patient_Phone,
        p.email AS Patient_Email,
        a.notes AS Appointment_Notes
    FROM
        Appointment a
    INNER JOIN
        Patient p ON a.P_ID = p.P_ID
    WHERE
        a.D_ID = doctor_id
    ORDER BY
        a.time ASC;
END;
//
DELIMITER ;
```

2) Retrieves the list of patients assigned to a specific doctor, including their details.

DELIMITER //

CREATE PROCEDURE GetPatientsByDoctor(IN doctor_id INT)

BEGIN

SELECT

p.P_ID AS Patient_ID,

p.name AS Patient_Name,

p.gender AS Gender,

p.DOB AS Date_of_Birth,

p.phone_no AS Phone,

p.email AS Email,

a.time AS Appointment_Time,

a.status AS Appointment_Status

FROM Patient p

JOIN Appointment a ON p.P_ID = a.P_ID

WHERE a.D_ID = doctor_id

ORDER BY a.time DESC;

END //

DELIMITER ;

```sql
-- Procedure retrieves the list of patients assigned to a specific doctor, including their details.
DELIMITER //
CREATE PROCEDURE GetPatientsByDoctor(IN doctor_id INT)
BEGIN
    SELECT
        p.P_ID AS Patient_ID,
        p.name AS Patient_Name,
        p.gender AS Gender,
        p.DOB AS Date_of_Birth,
        p.phone_no AS Phone,
        p.email AS Email,
        a.time AS Appointment_Time,
        a.status AS Appointment_Status
    FROM Patient p
    JOIN Appointment a ON p.P_ID = a.P_ID
    WHERE a.D_ID = doctor_id
    ORDER BY a.time DESC;
END //
DELIMITER ;
```

QUERIES:

1) Fetch appointments for the logged-in doctor

cur.execute("""

SELECT a.A_ID, a.time, a.status, a.notes, p.name AS patient_name

FROM Appointment a

JOIN Patient p ON a.P_ID = p.P_ID

WHERE a.D_ID = %s

""", (doctor_id,))

appointments = cur.fetchall()

```sql
# Fetch appointments for the logged-in doctor
cur.execute("""
    SELECT a.A_ID, a.time, a.status, a.notes, p.name AS patient_name
    FROM Appointment a
    JOIN Patient p ON a.P_ID = p.P_ID
    WHERE a.D_ID = %s
""", (doctor_id,))
appointments = cur.fetchall()
```

2) Fetch bills linked to the doctor's appointments

cur.execute("""

SELECT b.B_ID, b.date, b.amount, b.payment_status, a.A_ID

FROM Bill b

JOIN Appointment a ON b.B_ID = a.A_ID

WHERE a.D_ID = %s

""", (doctor_id,))

bills = cur.fetchall()

```
# Fetch bills linked to the doctor's appointments
cur.execute("""
    SELECT b.B_ID, b.date, b.amount, b.payment_status, a.A_ID
    FROM Bill b
    JOIN Appointment a ON b.B_ID = a.A_ID
    WHERE a.D_ID = %s
""", (doctor_id,))
bills = cur.fetchall()
```

3) Editing and updating doctor info

cur.execute("""

UPDATE doctors

SET name=%s, phone_no=%s, email=%s, job_title=%s, degree=%s, year=%s, employer=%s

WHERE D_ID=%s

""", (name, phone_no, email, job_title, degree, year, employer, doctor_id))

connection.commit()

cur.close()

```
# Editting and updating doctor info
cur.execute("""
    UPDATE doctors
    SET name=%s, phone_no=%s, email=%s, job_title=%s, degree=%s, year=%s, employer=%s
    WHERE D_ID=%s
""", (name, phone_no, email, job_title, degree, year, employer, doctor_id))
connection.commit()
cur.close()
```

4) Display all appointments for the doctor including patient information for each appointment and any prescription given.

cur.execute("""

SELECT a.A_ID, a.time, a.status, a.notes, p.name AS patient_name, pr.name AS prescription_name

FROM Appointment a

JOIN Patient p ON a.P_ID = p.P_ID

LEFT JOIN Prescription pr ON a.A_ID = pr.Pr_ID

WHERE a.D_ID = %s

""", (doctor_id,))

```
# Display all appointments for the doctor including patient information for each appointment and any prescription given.
cur.execute("""
    SELECT a.A_ID, a.time, a.status, a.notes, p.name AS patient_name, pr.name AS prescription_name
    FROM Appointment a
    JOIN Patient p ON a.P_ID = p.P_ID
    LEFT JOIN Prescription pr ON a.A_ID = pr.Pr_ID
    WHERE a.D_ID = %s
""", (doctor_id,))
appointments = cur.fetchall()
```

5) List of prescriptions doctor has provided during appointments.

cur.execute("""

SELECT p.Pr_ID, p.name AS prescription_name, p.dosage, p.instructions, p.duration,

a.time AS appointment_time, pat.name AS patient_name

FROM Prescription p

JOIN Appointment a ON p.Pr_ID = a.A_ID

JOIN Patient pat ON a.P_ID = pat.P_ID

WHERE a.D_ID = %s

""", (doctor_id,))

prescriptions = cur.fetchall()

cur.close()

```
# List of prescriptions doctor has provided during appointments.
cur.execute("""
    SELECT p.Pr_ID, p.name AS prescription_name, p.dosage, p.instructions, p.duration,
           a.time AS appointment_time, pat.name AS patient_name
    FROM Prescription p
    JOIN Appointment a ON p.Pr_ID = a.A_ID
    JOIN Patient pat ON a.P_ID = pat.P_ID
    WHERE a.D_ID = %s
""", (doctor_id,))
prescriptions = cur.fetchall()
cur.close()
```

RELATIONSHIP TABLES:

1) Associates doctors (D_ID) with patients (P_ID) to represent the patients who need care from specific doctors.

CREATE TABLE IF NOT EXISTS Needs (

D_ID INT,

P_ID INT, -- P_ID references INT for Patient

PRIMARY KEY (D_ID, P_ID),

FOREIGN KEY (D_ID) REFERENCES doctors(D_ID),

FOREIGN KEY (P_ID) REFERENCES Patient(P_ID)

);

```
106  -- Associates doctors (D_ID) with patients (P_ID) to represent the patients who need care from specific doctors.
107  CREATE TABLE IF NOT EXISTS Needs (
108      D_ID INT,
109      P_ID INT,  -- P_ID references INT for Patient
110      PRIMARY KEY (D_ID, P_ID),
111      FOREIGN KEY (D_ID) REFERENCES doctors(D_ID),
112      FOREIGN KEY (P_ID) REFERENCES Patient(P_ID)
113  );
```

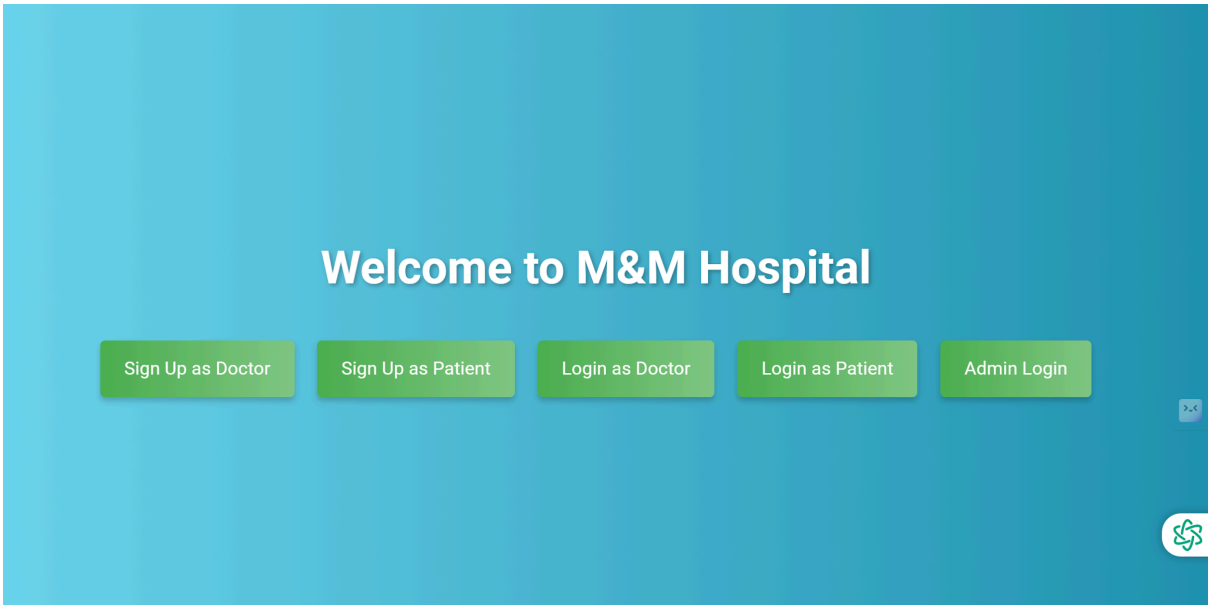
HTML PART:

1) home.html

Home page for the user – doctor, admin and patient

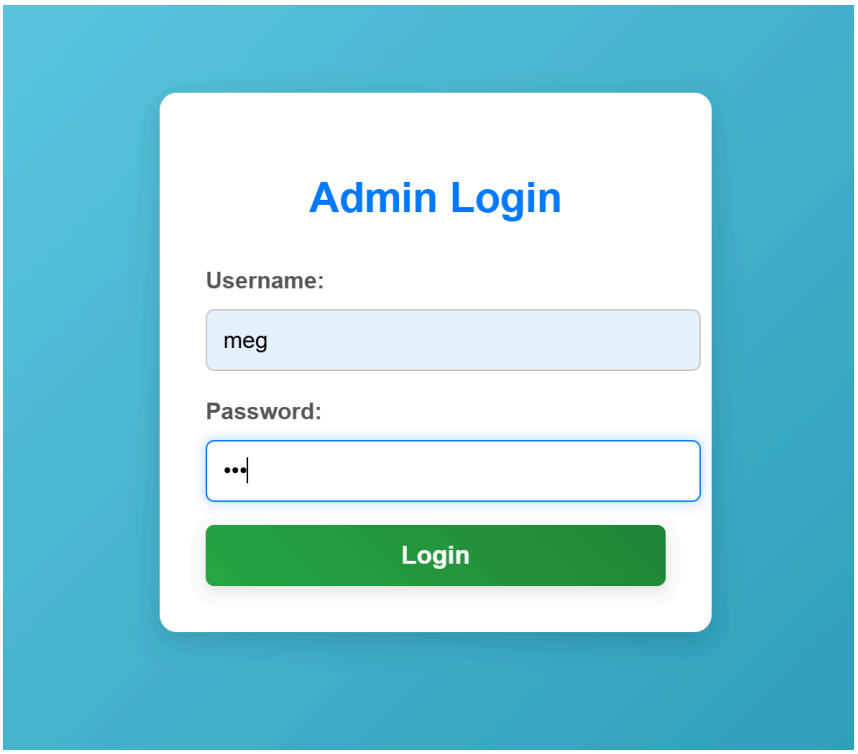
2) index.html

Main page where user can signup/ login for their respective category – doctor, admin and patient

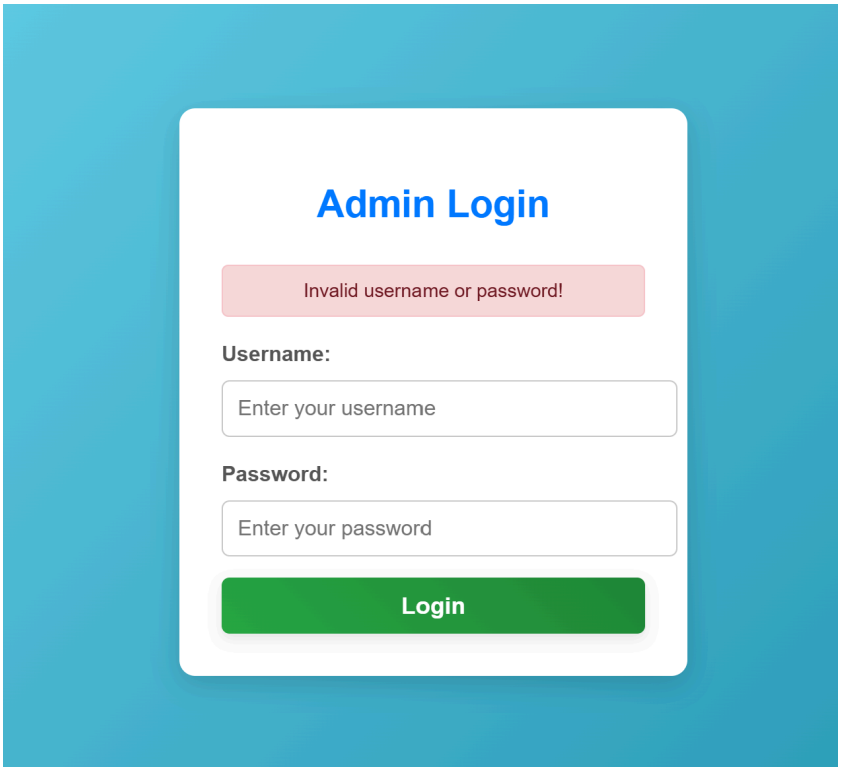


3) admin_login.html

Admin logs into their account here. Username = meg ; password = meg

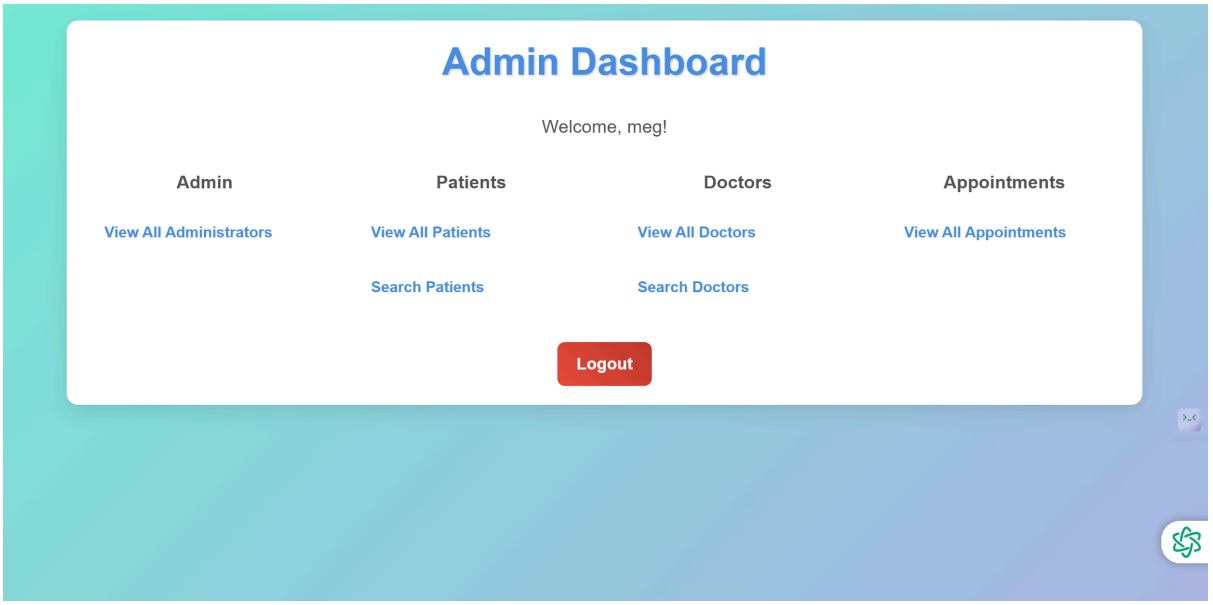


Invalid username/ password-



4) admin_dashboard.html

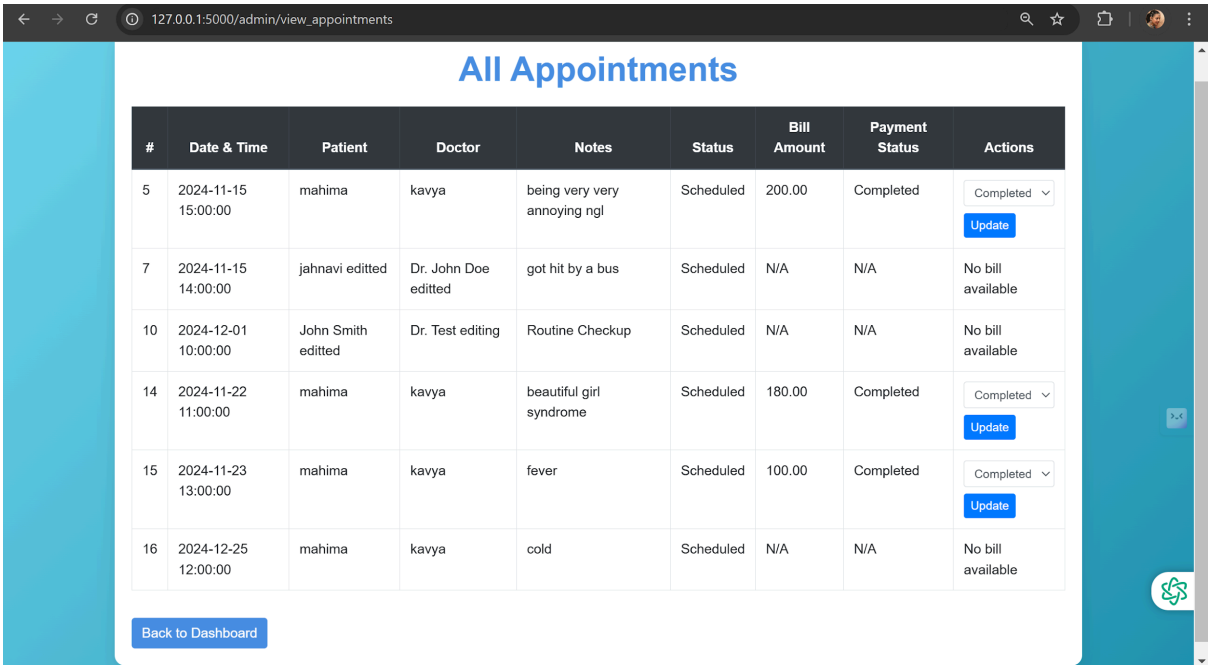
Page for the admin where they can click on the information they want to see like view and search for patients and doctors, view admins, and view all appointments made with their corresponding prescriptions and bills.



5) admin_view_appointments.html

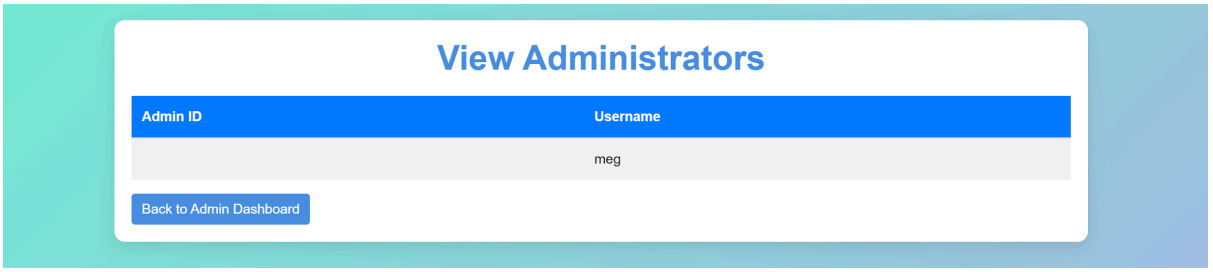
Page where the admin can view all the appointments made by patients with the prescription and bill the doctor has provided.

The admin can also change the status of bill payment to completed here.



6) view_administrators.html

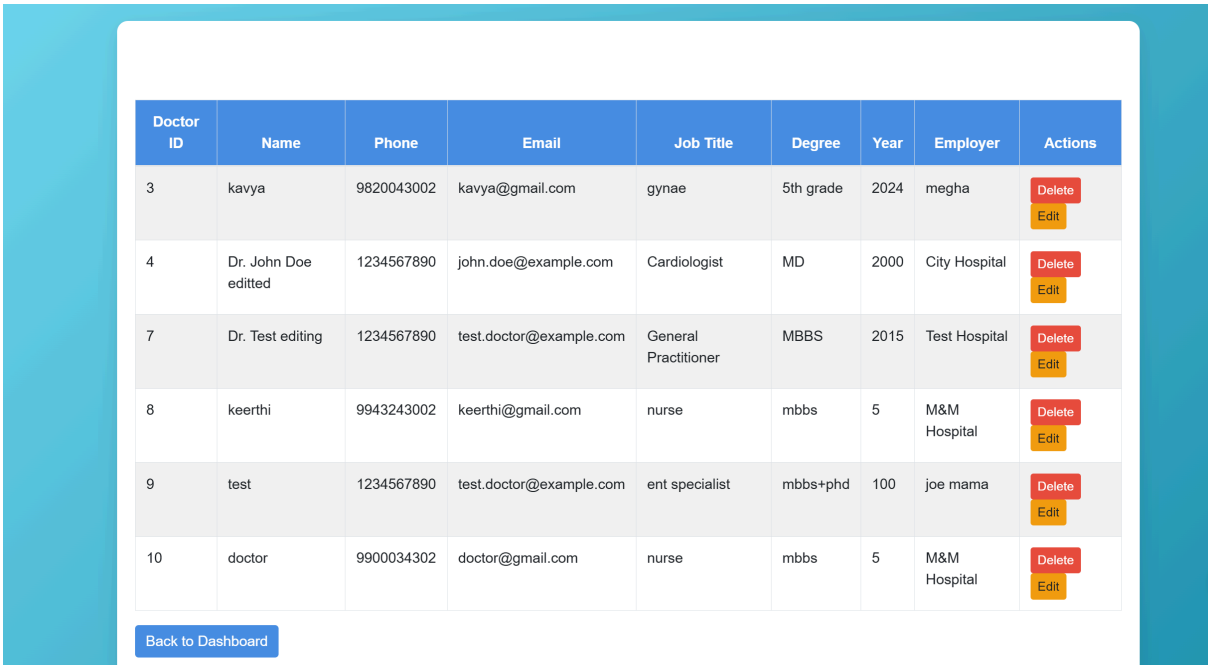
Admin can view all admins in the M&M Hospital here.



7) view_doctors.html

Admin can view all doctors and their information in this page.

Admin can also edit or delete their information.



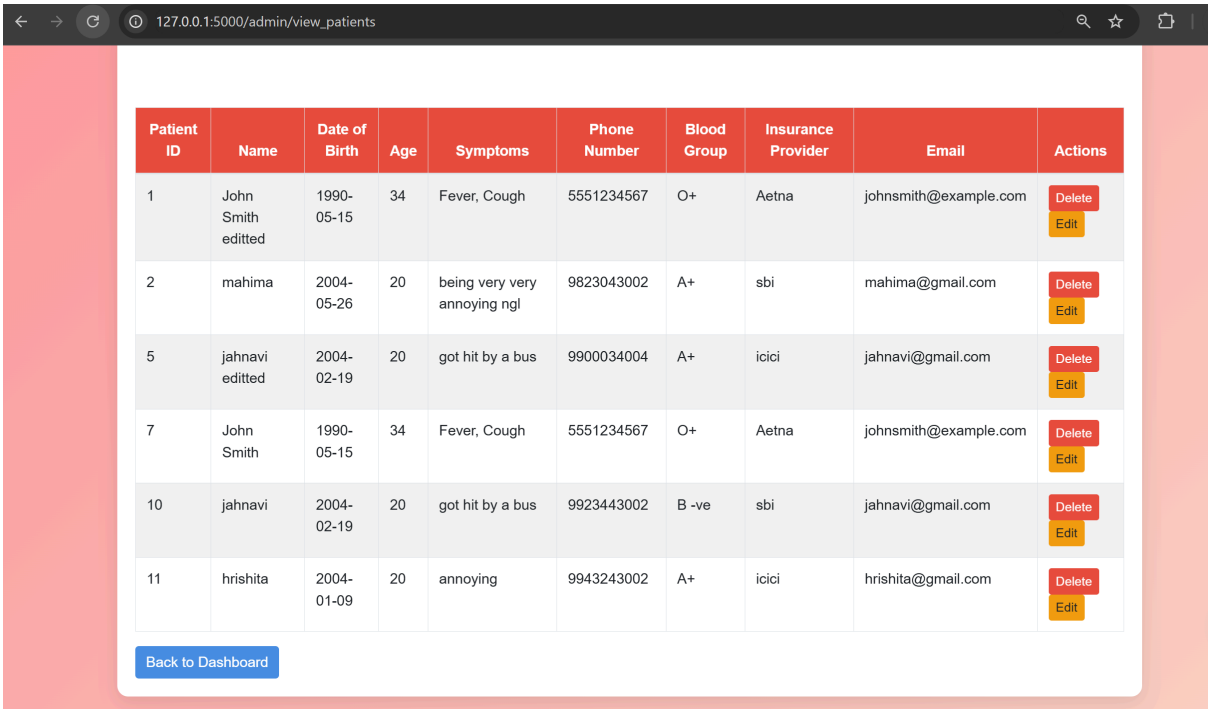
Doctor ID	Name	Phone	Email	Job Title	Degree	Year	Employer	Actions
3	kavya	9820043002	kavya@gmail.com	gynae	5th grade	2024	megha	Delete Edit
4	Dr. John Doe editted	1234567890	john.doe@example.com	Cardiologist	MD	2000	City Hospital	Delete Edit
7	Dr. Test editing	1234567890	test.doctor@example.com	General Practitioner	MBBS	2015	Test Hospital	Delete Edit
8	keerthi	9943243002	keerthi@gmail.com	nurse	mbbs	5	M&M Hospital	Delete Edit
9	test	1234567890	test.doctor@example.com	ent specialist	mbbs+phd	100	joe mama	Delete Edit
10	doctor	9900034302	doctor@gmail.com	nurse	mbbs	5	M&M Hospital	Delete Edit

Back to Dashboard

8) view_patients.html:

Admin can view all patients and their information in this page.

Admin can also edit or delete their information.



Patient ID	Name	Date of Birth	Age	Symptoms	Phone Number	Blood Group	Insurance Provider	Email	Actions
1	John Smith editted	1990-05-15	34	Fever, Cough	5551234567	O+	Aetna	johnsmith@example.com	Delete Edit
2	mahima	2004-05-26	20	being very very annoying ngl	9823043002	A+	sbi	mahima@gmail.com	Delete Edit
5	jahnavi editted	2004-02-19	20	got hit by a bus	9900034004	A+	icici	jahnavi@gmail.com	Delete Edit
7	John Smith	1990-05-15	34	Fever, Cough	5551234567	O+	Aetna	johnsmith@example.com	Delete Edit
10	jahnavi	2004-02-19	20	got hit by a bus	9923443002	B -ve	sbi	jahnavi@gmail.com	Delete Edit
11	hrishita	2004-01-09	20	annoying	9943243002	A+	icici	hrishita@gmail.com	Delete Edit

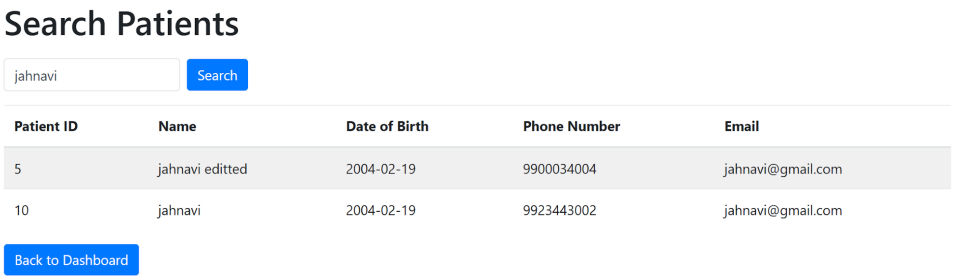
Back to Dashboard

9) search_doctors.html

Admin can search a doctor to get their information through their name.

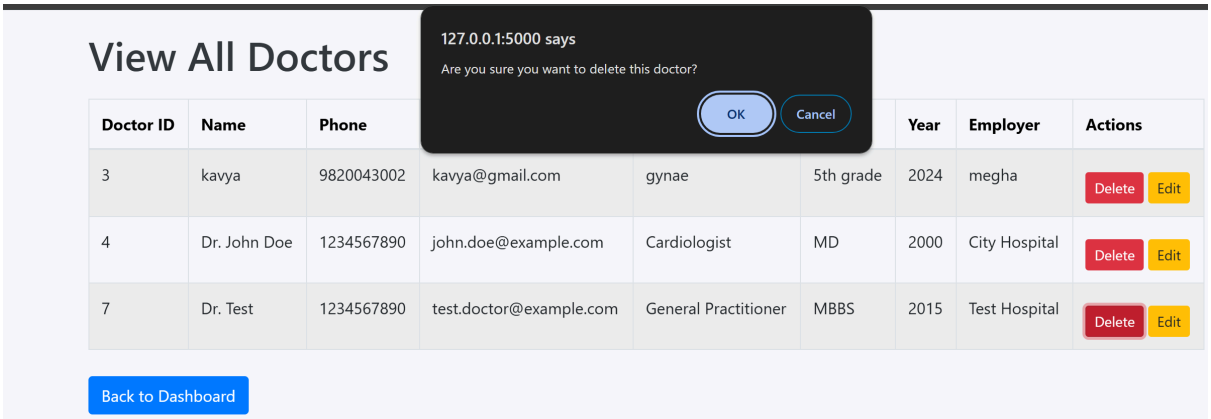
10) search_patients.html

Admin can search a patient to get their information through their name.



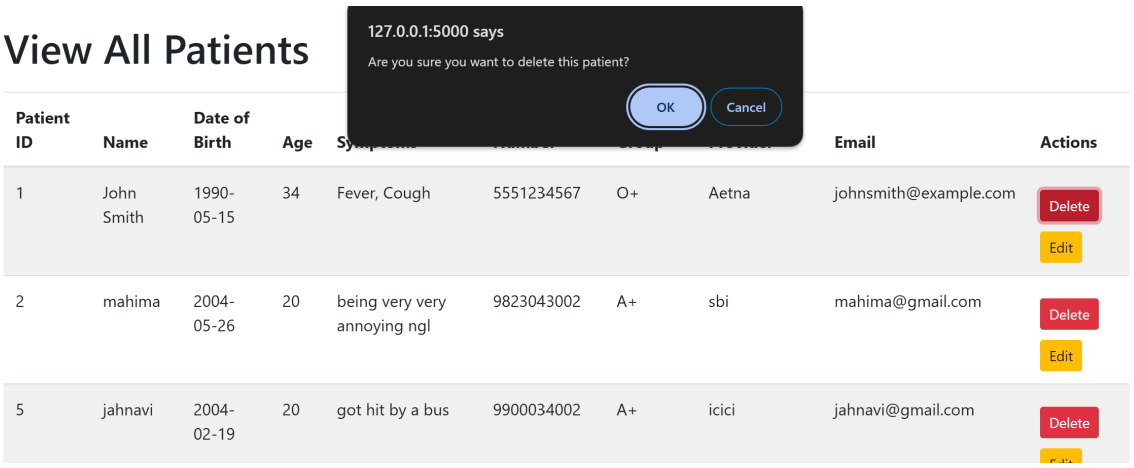
11) delete_doctors.html

Helps the admin to delete doctor information in the view_doctors page.



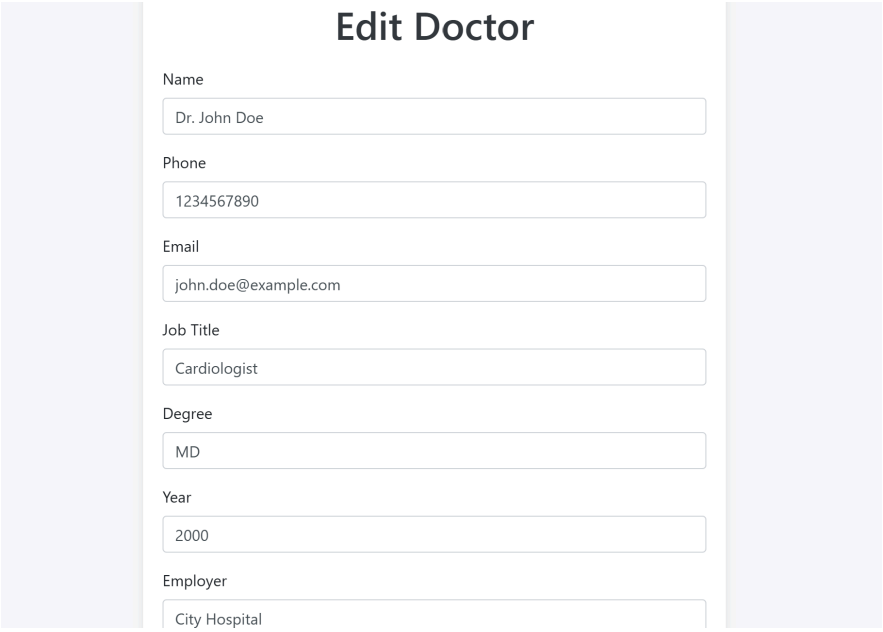
12) delete_patients.html

Helps the admin to delete patient information in the view_patients page.

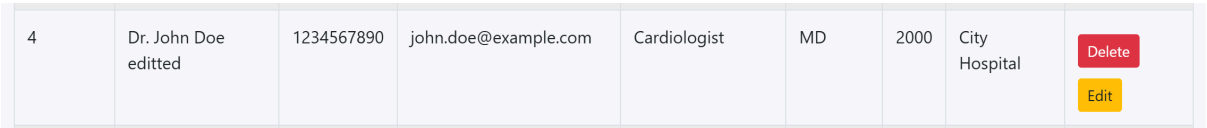


13) edit_doctors.html

Helps the admin to edit doctor information in the view_doctors page.

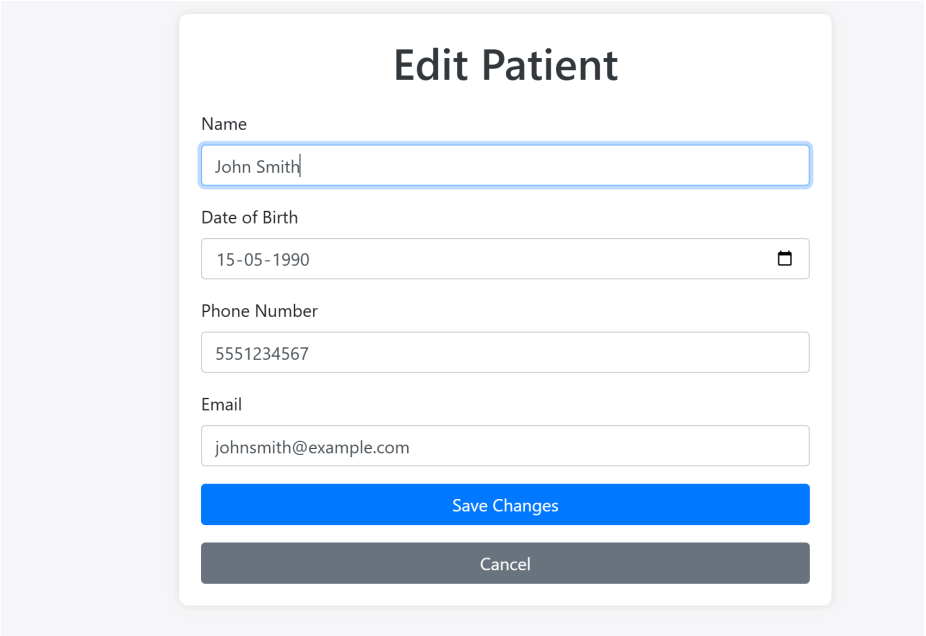


After editting-

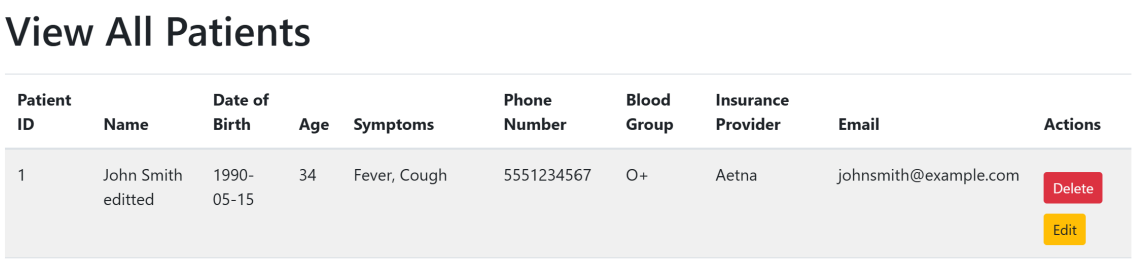


14) edit_patients.html

Helps the admin to edit patient information in the view_patients page.

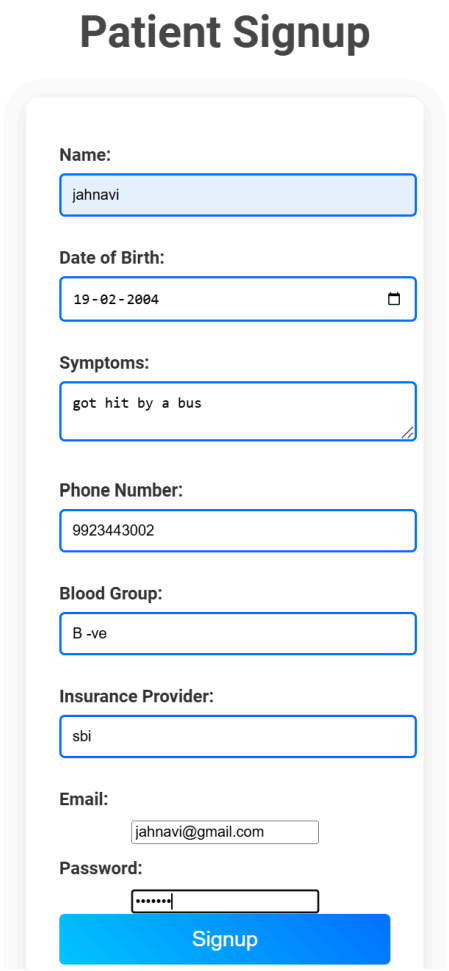


After editing-



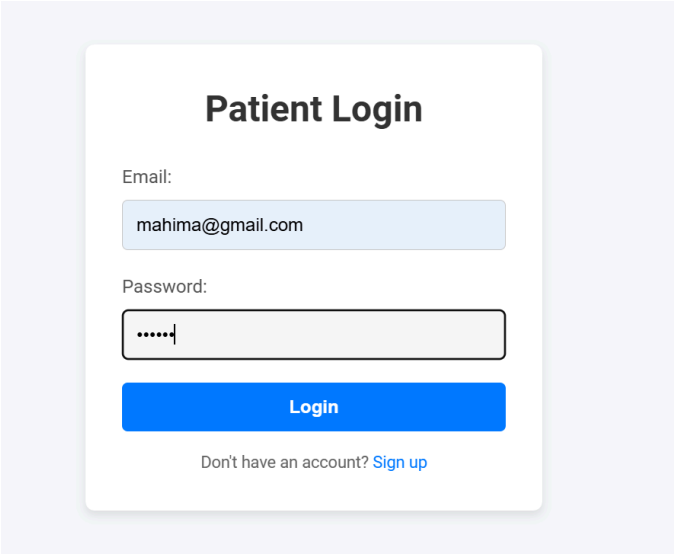
15) patients_signup.html

New patients can make their account in the M&M Hospital by signing up and giving their information like name, dob, symptoms, phone number, blood group, insurance, email and making a password.



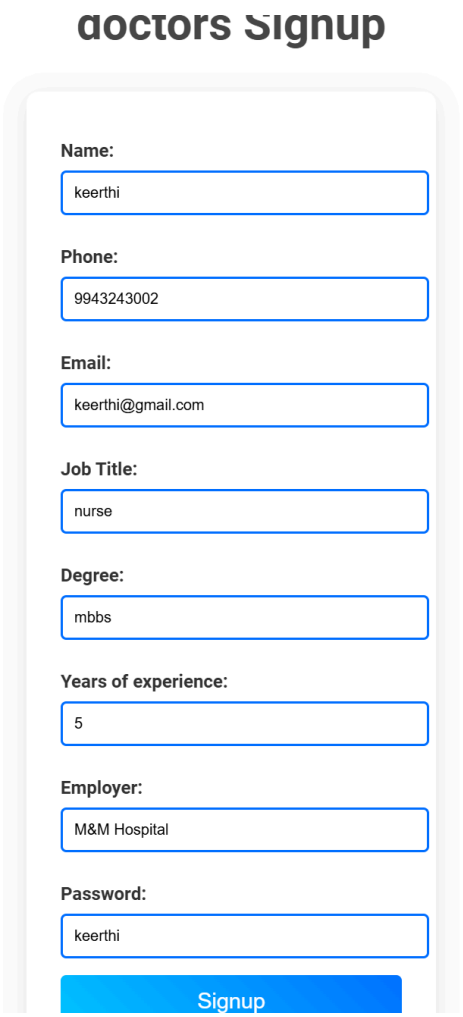
16) patients_login.html

Patients can login to their account by writing their email and corresponding password.



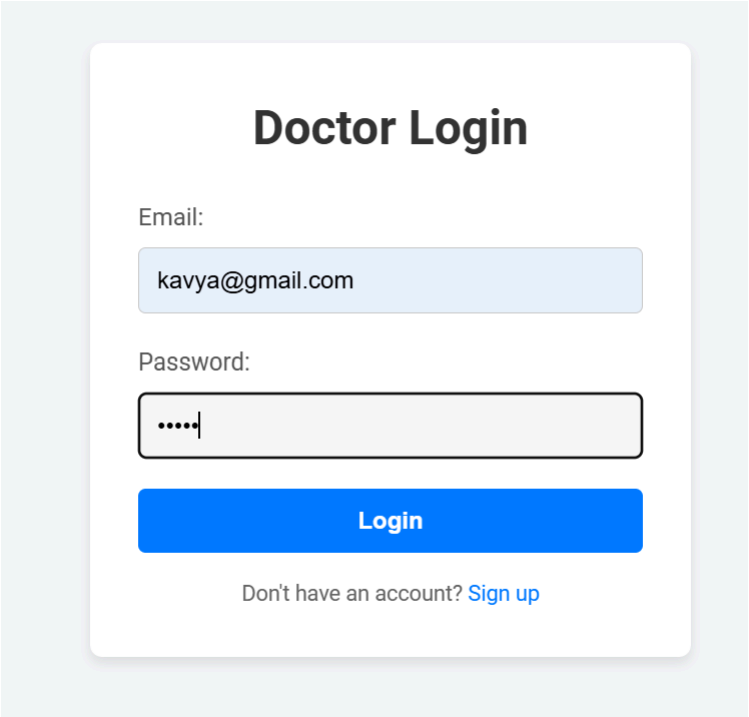
17) doctors_signup.html

New doctors can make their account in the M&M Hospital by signing up and giving their information like name, phone number, email, job title, degree, years of experience, employer and making a password.



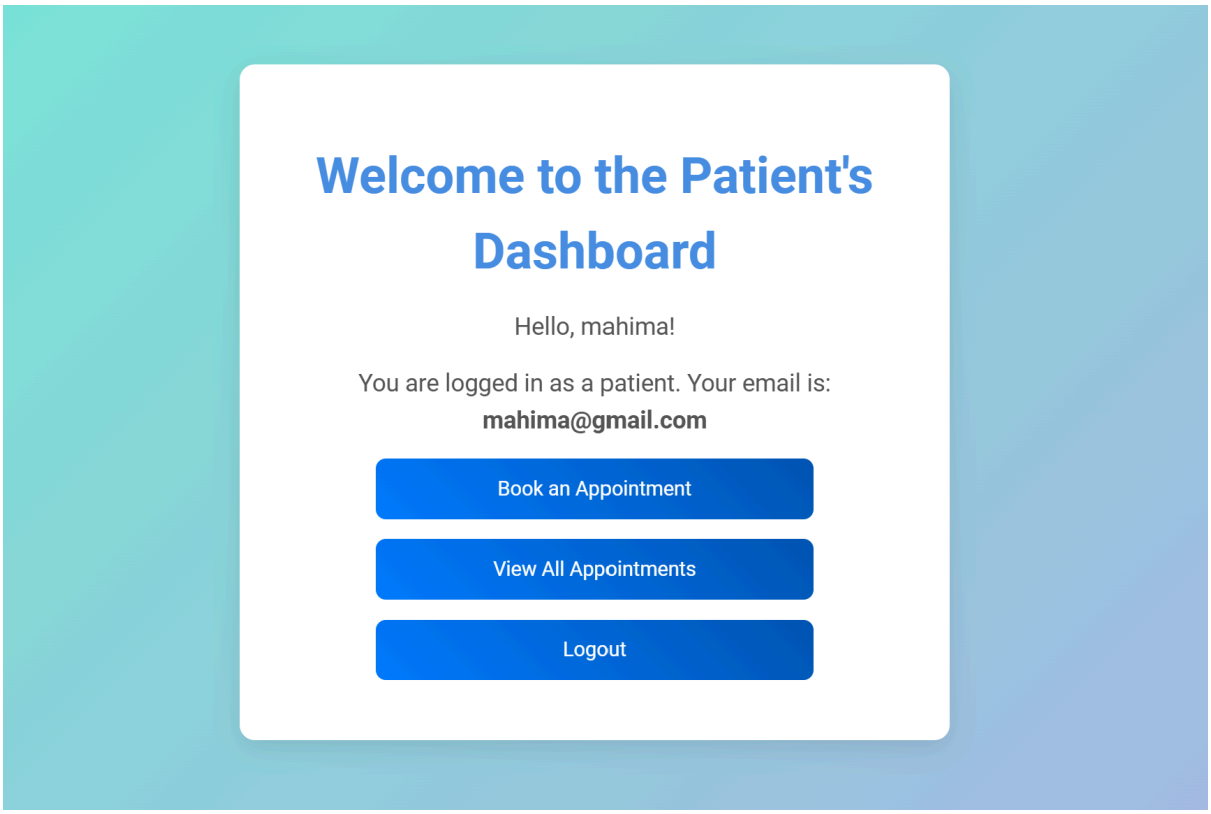
18) doctors_login.html

Doctors can login to their account by writing their email and corresponding password.



19) patients_welcome.html

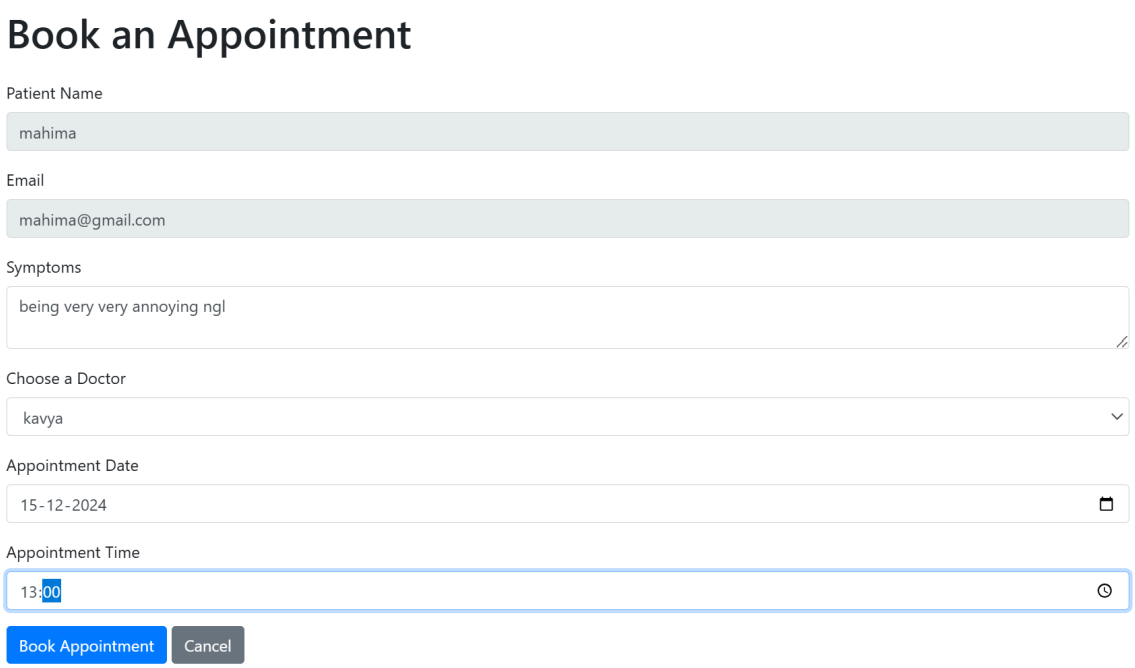
Patients can either click on ‘book an appointment’ or ‘cancel an appointment’ button here.



20) book_app.html

Patients can edit their symptoms here and can book an appointment by giving information like the doctor they want and their preferred date and time.

Booking appointments doesn't work when we book for the past.



21) view_app.html

When the patients books an appointment they can view that information here.

If the patient has finished consulting a doctor and if the doctor has given a prescription and bill, then that patient can view that in this page.

The patient can also cancel an appointment here.

Your Appointments

#	Date & Time	Doctor	Notes	Status	Prescription	Bill	Actions
1	2024-11-15 13:00:00	kavya	being very very annoying ngl	Scheduled	**dolo** Dosage: 1 tablet 1-0-1 Instructions: take sensibly	Amount: 500.00 Status: Completed	Cancel
5	2024-11-15 15:00:00	kavya	being very very annoying ngl	Scheduled	**crocin 2.0** Dosage: 10ml 1-0-1 Instructions: dont take too much just because it tastes good	Amount: 200.00 Status: Completed	Cancel
11	2024-12-13 13:00:00	kavya	being very very annoying ngl	Scheduled	No prescription added	No bill generated	Cancel
12	2024-12-15 13:00:00	kavya	being very very annoying ngl	Scheduled	No prescription added	No bill generated	Cancel

Back to Dashboard

After patient cancels appointment-

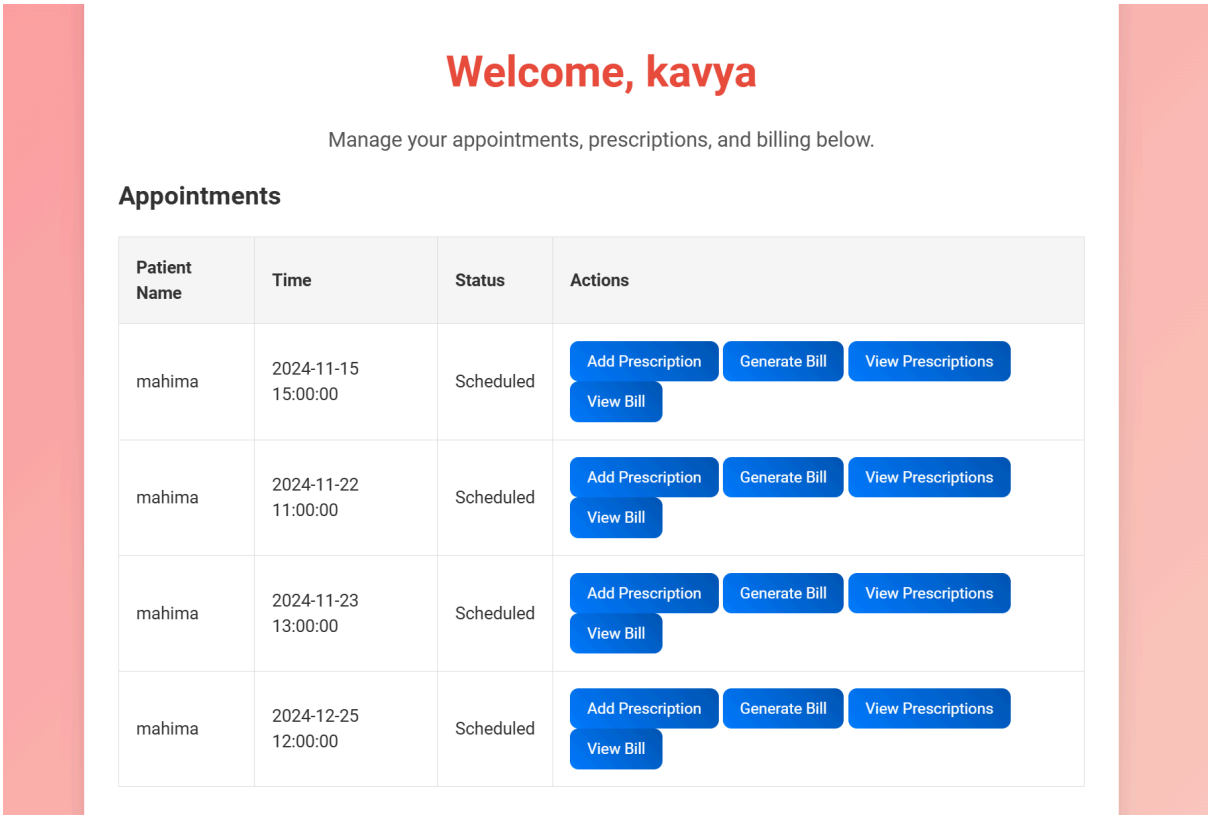
Your Appointments

#	Date & Time	Doctor	Notes	Status	Prescription	Bill	Actions
1	2024-11-15 13:00:00	kavya	being very very annoying ngl	Scheduled	**dolo** Dosage: 1 tablet 1-0-1 Instructions: take sensibly	Amount: 500.00 Status: Completed	Cancel
5	2024-11-15 15:00:00	kavya	being very very annoying ngl	Scheduled	**crocin 2.0** Dosage: 10ml 1-0-1 Instructions: dont take too much just because it tastes good	Amount: 200.00 Status: Completed	Cancel
11	2024-12-13 13:00:00	kavya	being very very annoying ngl	Scheduled	No prescription added	No bill generated	Cancel

Back to Dashboard

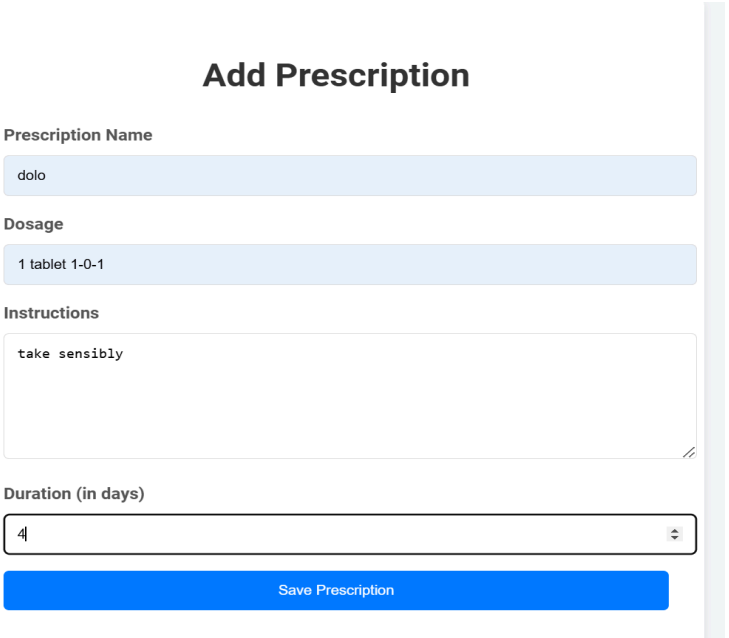
22) doctors_welcome.html

Doctors can view the patients that have booked an appointment in this page and can click to give/ view prescription and bill.



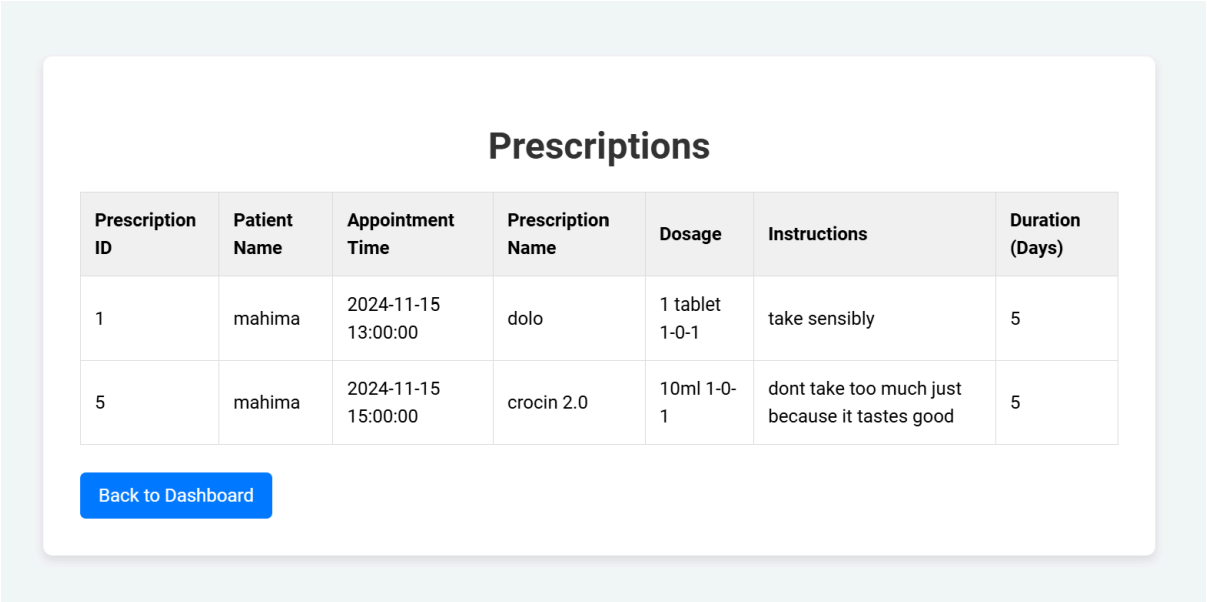
23) add_prescription.html

Doctors can give prescription to that patient by entering the medicine name, dosage, days of take, etc here.



24) view_prescriptions.html

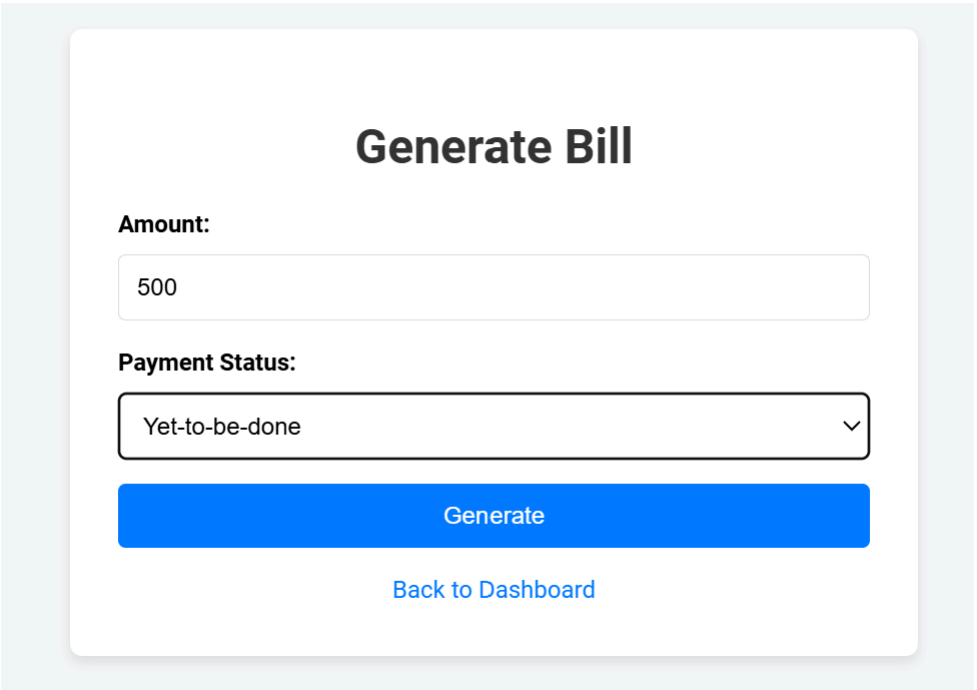
Doctors can view the prescription they have given to that patient.



25) generate_bill.html

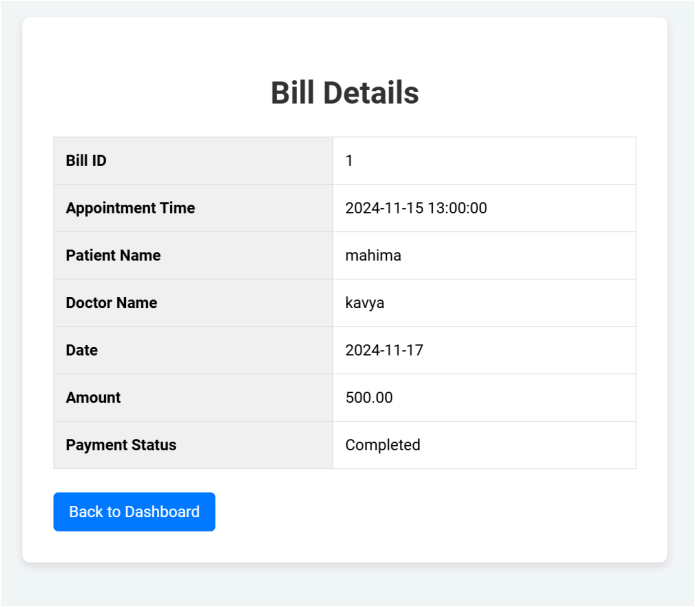
Doctors can give the amount to be paid by the patient here.

If the patient pays that time itself they can keep payment status as ‘completed’ or else doctors can change it to ‘yet to be done’.



26) view_bill.html

Doctors can view the bill they have given to the patient here with the payment status included.



THANK YOU