

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.

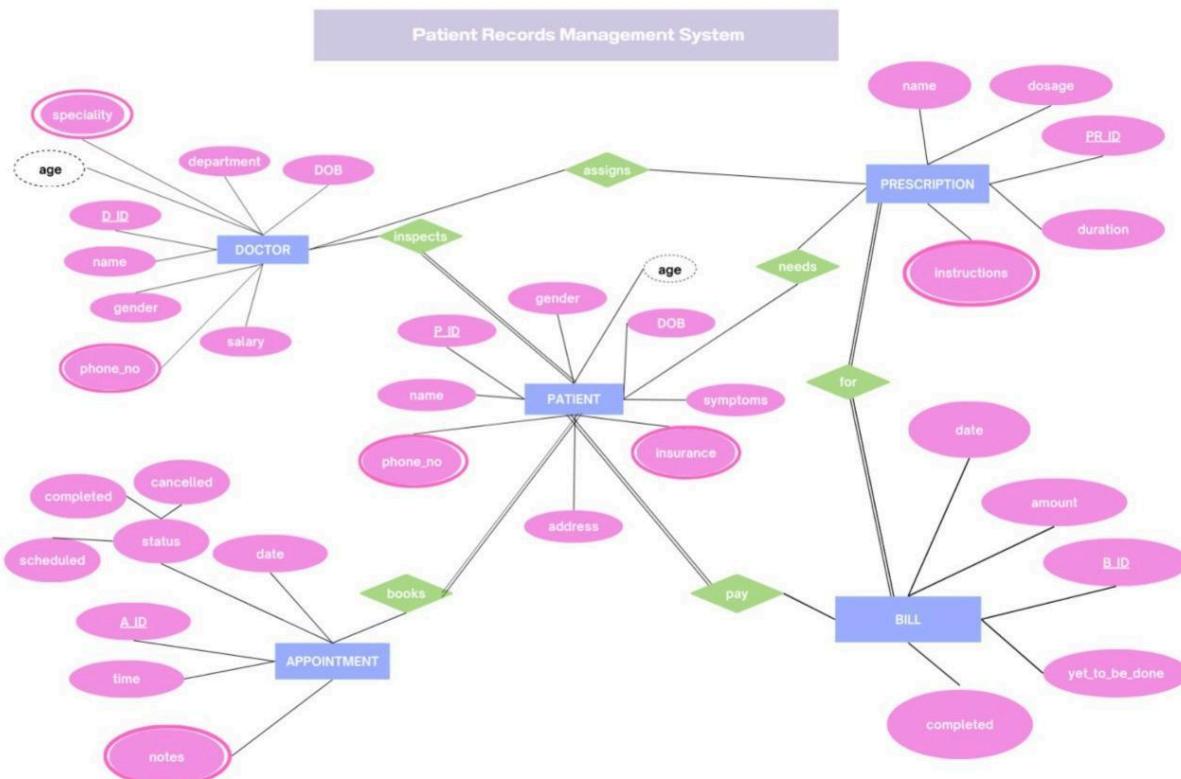
- **User Dashboards:** There are separate login pages and dashboards for doctors, admins and patients.
- **Feedback Messages:** Flash messages notify users about their actions (e.g., login success, errors).

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:

Relational Schema.

DOCTOR

D-ID	name	gender	phone no	department	DOB	speciality	salary

PATIENT

P-ID	name	gender	phone.no	DOB	symptoms	insurance	D-ID

PRESCRIPTION

PR-ID	name	dosage	duration	instructions	P-ID	D-ID

APPOINTMENT

A-ID	time	notes	date	scheduled	cancelled	completed	P-ID	D-ID

BILL

B-ID	date	amount	yet_to_be_done	completed	P-ID

DATABASE CREATION:

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

```

USE HospitalDB;

-- 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)
);

```

```

38   -- Patient Table (Auto Increment P_ID)
39   CREATE TABLE IF NOT EXISTS Patient (
40       P_ID INT AUTO_INCREMENT PRIMARY KEY, -- Auto increment P_ID
41       name VARCHAR(50) NOT NULL,
42       gender VARCHAR(10),
43       DOB DATE,
44       Age INT,
45       symptoms TEXT, -- Symptoms column added
46       address VARCHAR(128),
47       phone_no VARCHAR(15),
48       blood_group VARCHAR(5),
49       insurance VARCHAR(50),
50       email VARCHAR(255),
51       password VARCHAR(255)
52   );

```

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)
);

55  -- Doctors Table with AUTO_INCREMENT for D_ID
56  CREATE TABLE IF NOT EXISTS doctors (
57      D_ID INT AUTO_INCREMENT PRIMARY KEY, -- Ensure D_ID is AUTO_INCREMENT
58      name VARCHAR(50) NOT NULL,
59      phone_no VARCHAR(15),
60      email VARCHAR(255),
61      job_title VARCHAR(100),
62      degree VARCHAR(100),
63      year INT,
64      employer VARCHAR(255),
65      password VARCHAR(255),
66      specialty VARCHAR(50),
67      DOB DATE,
68      Age INT,
69      salary DECIMAL(10, 2)
70 );
```

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
);

19   -- Administrator Table
20   CREATE TABLE IF NOT EXISTS Administrator (
21       id INT AUTO_INCREMENT PRIMARY KEY,
22       username VARCHAR(50) UNIQUE NOT NULL,
23       password VARCHAR(255) NOT NULL
24   );
25
26   -- Initial Data for Administrator
27   INSERT INTO Administrator (username, password)
28   VALUES ('meg', 'meg')
29   ON DUPLICATE KEY UPDATE password = 'meg';
30

```

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
);


```

```

73   -- Appointment Table
74   CREATE TABLE IF NOT EXISTS Appointment (
75       A_ID INT AUTO_INCREMENT PRIMARY KEY, -- Auto-increment integer ID
76       time DATETIME NOT NULL,           -- Date and Time of Appointment
77       notes TEXT,                      -- Additional Notes for Appointment
78       status ENUM('Scheduled', 'Concluded', 'Completed', 'Cancelled') DEFAULT 'Scheduled', -- Appointment Status
79       P_ID INT NOT NULL,               -- Patient ID
80       D_ID INT NOT NULL,               -- Doctor ID
81       FOREIGN KEY (P_ID) REFERENCES Patient(P_ID) ON DELETE CASCADE, -- Ensure relational integrity
82       FOREIGN KEY (D_ID) REFERENCES doctors(D_ID) ON DELETE CASCADE -- Ensure relational integrity
83   );

```

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
);
```

```
87      -- Prescription Table
88  CREATE TABLE IF NOT EXISTS Prescription (
89      Pr_ID VARCHAR(20) PRIMARY KEY,
90      name VARCHAR(50),
91      dosage VARCHAR(50),
92      instructions TEXT,
93      duration INT
94  );
95
```

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')
);
```

```
96      -- Bill Table
97  CREATE TABLE IF NOT EXISTS Bill (
98      B_ID VARCHAR(20) PRIMARY KEY,
99      date DATE,
100     amount DECIMAL(10, 2),
101     payment_status ENUM('Completed', 'Yet-to-be-done')
102 );
```

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 ;

```

115  -- Triggers for Patient Table: Before Insert to Update Age Based on DOB
116  DROP TRIGGER IF EXISTS before_insert_patient;
117  DELIMITER //
118  CREATE TRIGGER before_insert_patient
119  BEFORE INSERT ON Patient
120  FOR EACH ROW
121      BEGIN
122          IF NEW.DOB IS NOT NULL THEN
123              SET NEW.Age = TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE());
124          END IF;
125      END;
126      //
127  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 ;

130  -- Trigger to ensure that an appointment cannot have conflicting statuses or be scheduled in the past.
131  DELIMITER //
132  CREATE TRIGGER before_insert_appointment
133  BEFORE INSERT ON Appointment
134  FOR EACH ROW
135  BEGIN
136      -- Ensure the appointment date is in the future
137      IF NEW.time <= NOW() THEN
138          SIGNAL SQLSTATE '45000'
139          SET MESSAGE_TEXT = 'Appointment cannot be scheduled in the past.';
140      END IF;
141  END;
142  //
143  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 ;  

145  -- Procedure to fetch all appointments for a specific doctor, including details about the patient.  

146  DELIMITER //  

147  CREATE PROCEDURE GetAppointmentsByDoctor (IN doctor_id INT)  

148  BEGIN  

149      SELECT  

150          a.A_ID AS Appointment_ID,  

151          a.time AS Appointment_Time,  

152          a.status AS Appointment_Status,  

153          p.name AS Patient_Name,  

154          p.phone_no AS Patient_Phone,  

155          p.email AS Patient_Email,  

156          a.notes AS Appointment_Notes  

157      FROM  

158          Appointment a  

159      INNER JOIN  

160          Patient p ON a.P_ID = p.P_ID  

161      WHERE  

162          a.D_ID = doctor_id  

163      ORDER BY  

164          a.time ASC;  

165  END;  

166  //  

167  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 ;

171  -- Procedure retrieves the list of patients assigned to a specific doctor, including their details.
172  DELIMITER //
173  CREATE PROCEDURE GetPatientsByDoctor(IN doctor_id INT)
174  BEGIN
175      SELECT
176          p.P_ID AS Patient_ID,
177          p.name AS Patient_Name,
178          p.gender AS Gender,
179          p.DOB AS Date_of_Birth,
180          p.phone_no AS Phone,
181          p.email AS Email,
182          a.time AS Appointment_Time,
183          a.status AS Appointment_Status
184      FROM Patient p
185      JOIN Appointment a ON p.P_ID = a.P_ID
186      WHERE a.D_ID = doctor_id
187      ORDER BY a.time DESC;
188  END //
189  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()

# 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  );
```

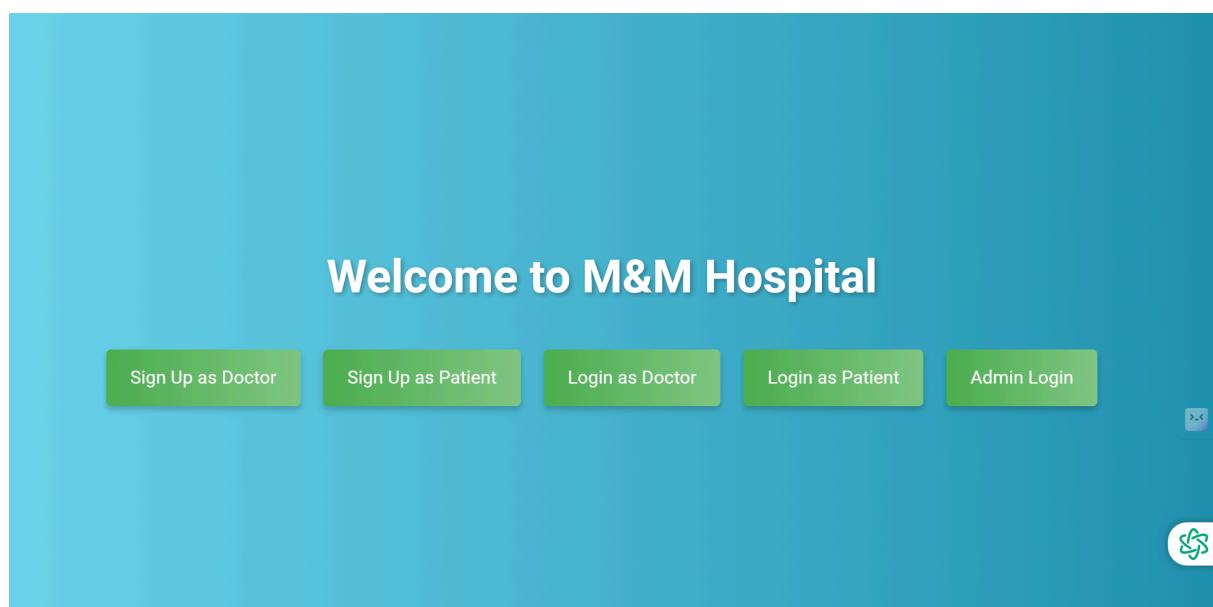
HTML PART:

1) home.html

Home page for the user – doctor, admin and patient

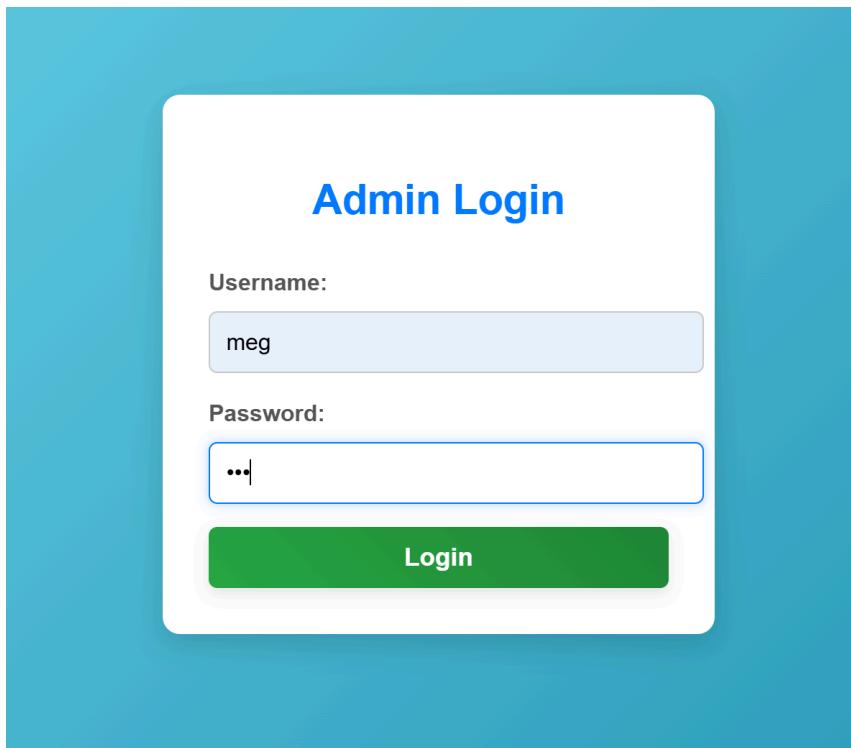
2) index.html

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

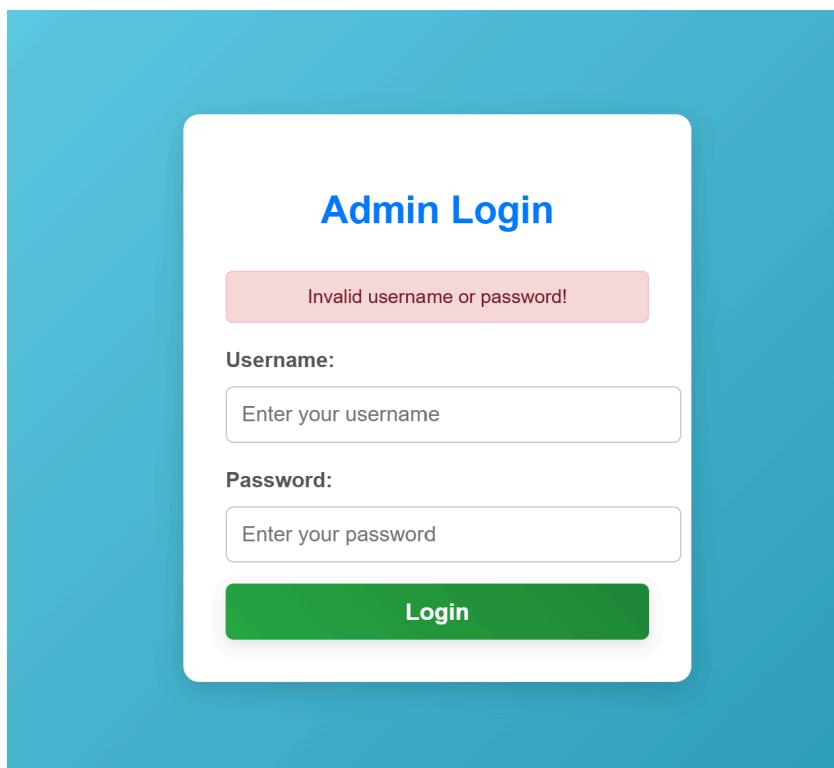


3) admin_login.html

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

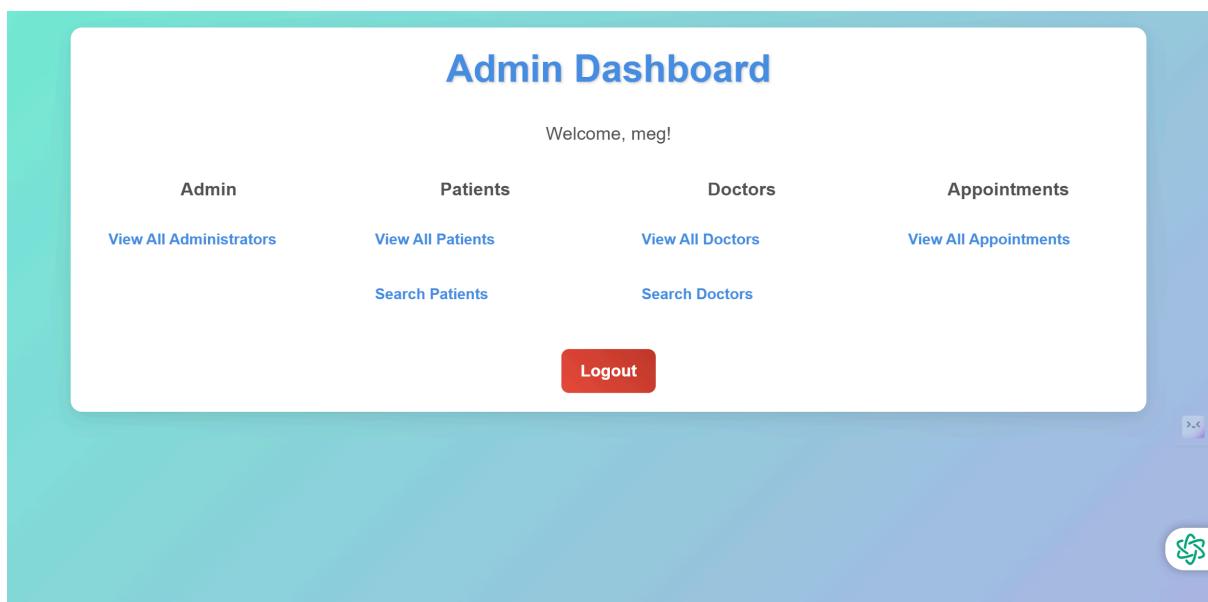


Invalid username/ password-



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.

The screenshot shows a web browser window with the URL 127.0.0.1:5000/admin/view_appointments. The title of the page is "All Appointments". Below the title is a table with the following data:

#	Date & Time	Patient	Doctor	Notes	Status	Bill Amount	Payment Status	Actions
5	2024-11-15 15:00:00	mahima	kavya	being very very annoying ngl	Scheduled	200.00	Completed	<input type="button" value="Completed"/> <input type="button" value="Update"/>
7	2024-11-15 14:00:00	jahnavi edited	Dr. John Doe edited	got hit by a bus	Scheduled	N/A	N/A	No bill available
10	2024-12-01 10:00:00	John Smith edited	Dr. Test editing	Routine Checkup	Scheduled	N/A	N/A	No bill available
14	2024-11-22 11:00:00	mahima	kavya	beautiful girl syndrome	Scheduled	180.00	Completed	<input type="button" value="Completed"/> <input type="button" value="Update"/>
15	2024-11-23 13:00:00	mahima	kavya	fever	Scheduled	100.00	Completed	<input type="button" value="Completed"/> <input type="button" value="Update"/>
16	2024-12-25 12:00:00	mahima	kavya	cold	Scheduled	N/A	N/A	No bill available

[Back to Dashboard](#)

6) view_administrators.html

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

The screenshot shows a web browser window with the URL 127.0.0.1:5000/admin/view_administrators. The title of the page is "View Administrators". Below the title is a table with the following data:

Admin ID	Username
	meg

[Back to Admin Dashboard](#)

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	<button>Delete</button> <button>Edit</button>
4	Dr. John Doe edited	1234567890	john.doe@example.com	Cardiologist	MD	2000	City Hospital	<button>Delete</button> <button>Edit</button>
7	Dr. Test editing	1234567890	test.doctor@example.com	General Practitioner	MBBS	2015	Test Hospital	<button>Delete</button> <button>Edit</button>
8	keerthi	9943243002	keerthi@gmail.com	nurse	mbbs	5	M&M Hospital	<button>Delete</button> <button>Edit</button>
9	test	1234567890	test.doctor@example.com	ent specialist	mbbs+phd	100	joe mama	<button>Delete</button> <button>Edit</button>
10	doctor	9900034302	doctor@gmail.com	nurse	mbbs	5	M&M Hospital	<button>Delete</button> <button>Edit</button>

[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 edited	1990-05-15	34	Fever, Cough	5551234567	O+	Aetna	johnsmith@example.com	<button>Delete</button> <button>Edit</button>
2	mahima	2004-05-26	20	being very very annoying ngl	9823043002	A+	sbi	mahima@gmail.com	<button>Delete</button> <button>Edit</button>
5	jahnavi edited	2004-02-19	20	got hit by a bus	9900034004	A+	icici	jahnavi@gmail.com	<button>Delete</button> <button>Edit</button>
7	John Smith	1990-05-15	34	Fever, Cough	5551234567	O+	Aetna	johnsmith@example.com	<button>Delete</button> <button>Edit</button>
10	jahnavi	2004-02-19	20	got hit by a bus	9923443002	B -ve	sbi	jahnavi@gmail.com	<button>Delete</button> <button>Edit</button>
11	hrishita	2004-01-09	20	annoying	9943243002	A+	icici	hrishita@gmail.com	<button>Delete</button> <button>Edit</button>

[Back to Dashboard](#)

9) search_doctors.html

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

Search Doctors

kavya

Name	Phone	Email	Job Title	Degree	Year	Employer
kavya	9820043002	kavya@gmail.com	gynae	5th grade	2024	megha

10) search_patients.html

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

Search Patients

jahnavi

Patient ID	Name	Date of Birth	Phone Number	Email
5	jahnavi editted	2004-02-19	9900034004	jahnavi@gmail.com
10	jahnavi	2004-02-19	9923443002	jahnavi@gmail.com

11) delete_doctors.html

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

View All Doctors

127.0.0.1:5000 says
Are you sure you want to delete this doctor?

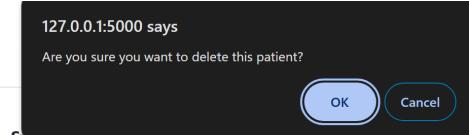
Doctor ID	Name	Phone	Email	Job Title	Degree	Year	Employer	Actions
3	kavya	9820043002	kavya@gmail.com	gynae	5th grade	2024	megha	<input type="button" value="Delete"/> <input type="button" value="Edit"/>
4	Dr. John Doe	1234567890	john.doe@example.com	Cardiologist	MD	2000	City Hospital	<input type="button" value="Delete"/> <input type="button" value="Edit"/>
7	Dr. Test	1234567890	test.doctor@example.com	General Practitioner	MBBS	2015	Test Hospital	<input type="button" value="Delete"/> <input type="button" value="Edit"/>

12) delete_patients.html

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

View All Patients

Patient ID	Name	Date of Birth	Age	Symptoms	Phone Number	Blood Type	Hospital	Email	Actions
1	John Smith	1990-05-15	34	Fever, Cough	5551234567	O+	Aetna	johnsmith@example.com	<button>Delete</button> <button>Edit</button>
2	mahima	2004-05-26	20	being very very annoying ngl	9823043002	A+	sbi	mahima@gmail.com	<button>Delete</button> <button>Edit</button>
5	jahnavi	2004-02-19	20	got hit by a bus	9900034002	A+	icici	jahnavi@gmail.com	<button>Delete</button> <button>Edit</button>



13) edit_doctors.html

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

Edit Doctor

Name

Phone

Email

Job Title

Degree

Year

Employer

After editing-

4	Dr. John Doe edited	1234567890	john.doe@example.com	Cardiologist	MD	2000	City Hospital	<button>Delete</button> <button>Edit</button>
---	---------------------	------------	----------------------	--------------	----	------	---------------	--

14) edit_patients.html

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

Edit Patient

Name
John Smith

Date of Birth
15-05-1990

Phone Number
5551234567

Email
johnsmith@example.com

After editing-

View All Patients

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	<input type="button" value="Delete"/> <input type="button" value="Edit"/>

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.

Patient Signup

Name:

Date of Birth:

 Symptoms:

Phone Number:

Blood Group:

Insurance Provider:

Email:

Password:

16) patients_login.html

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

Patient Login

Email:

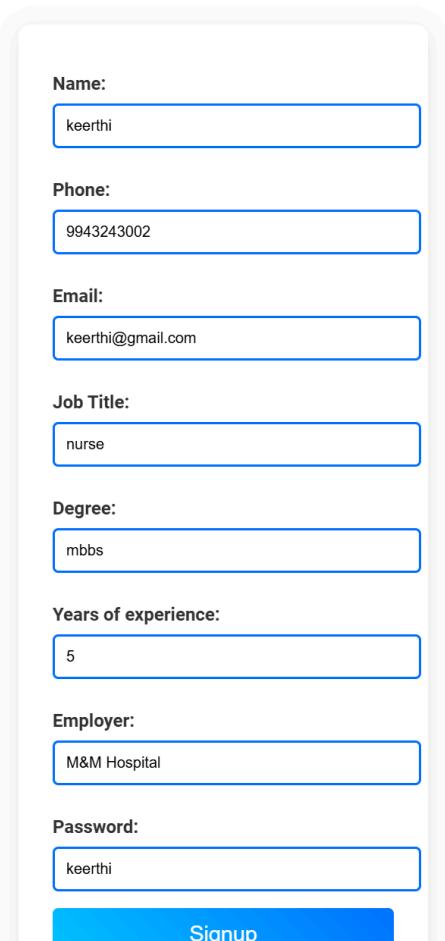
Password:

Don't have an account? [Sign up](#)

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.

doctors Signup



The screenshot shows a mobile-style sign-up form for doctors. The fields and their values are:

- Name: keerthi
- Phone: 9943243002
- Email: keerthi@gmail.com
- Job Title: nurse
- Degree: mbbs
- Years of experience: 5
- Employer: M&M Hospital
- Password: keerthi

A blue "Signup" button is at the bottom right of the form area.

18) doctors_login.html

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

Doctor Login

Email:

Password:

Login

Don't have an account? [Sign up](#)

19) patients_welcome.html

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

Welcome to the Patient's Dashboard

Hello, mahima!

You are logged in as a patient. Your email is:
mahima@gmail.com

Book an Appointment

View All Appointments

Logout

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.

Book an Appointment

Patient Name

Email

Symptoms

Choose a Doctor

Appointment Date
 

Appointment Time
 

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.

Welcome, kavya

Manage your appointments, prescriptions, and billing below.

Appointments

Patient Name	Time	Status	Actions
mahima	2024-11-15 15:00:00	Scheduled	Add Prescription Generate Bill View Prescriptions View Bill
mahima	2024-11-22 11:00:00	Scheduled	Add Prescription Generate Bill View Prescriptions View Bill
mahima	2024-11-23 13:00:00	Scheduled	Add Prescription Generate Bill View Prescriptions View Bill
mahima	2024-12-25 12:00:00	Scheduled	Add Prescription Generate Bill View Prescriptions View Bill

23) add_prescription.html

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

Add Prescription

Prescription Name

dolo

Dosage

1 tablet 1-0-1

Instructions

take sensibly

Duration (in days)

4

[Save Prescription](#)

24) view_prescriptions.html

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

Prescriptions

Prescription ID	Patient Name	Appointment Time	Prescription Name	Dosage	Instructions	Duration (Days)
1	mahima	2024-11-15 13:00:00	dolo	1 tablet 1-0-1	take sensibly	5
5	mahima	2024-11-15 15:00:00	crocin 2.0	10ml 1-0-1	don't take too much just because it tastes good	5

[Back to Dashboard](#)

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'.

Generate Bill

Amount:

Payment Status:

Yet-to-be-done

Generate

[Back to Dashboard](#)

26) view_bill.html

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

Bill Details

Bill ID	1
Appointment Time	2024-11-15 13:00:00
Patient Name	mahima
Doctor Name	kavya
Date	2024-11-17
Amount	500.00
Payment Status	Completed

[Back to Dashboard](#)

THANK YOU