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).

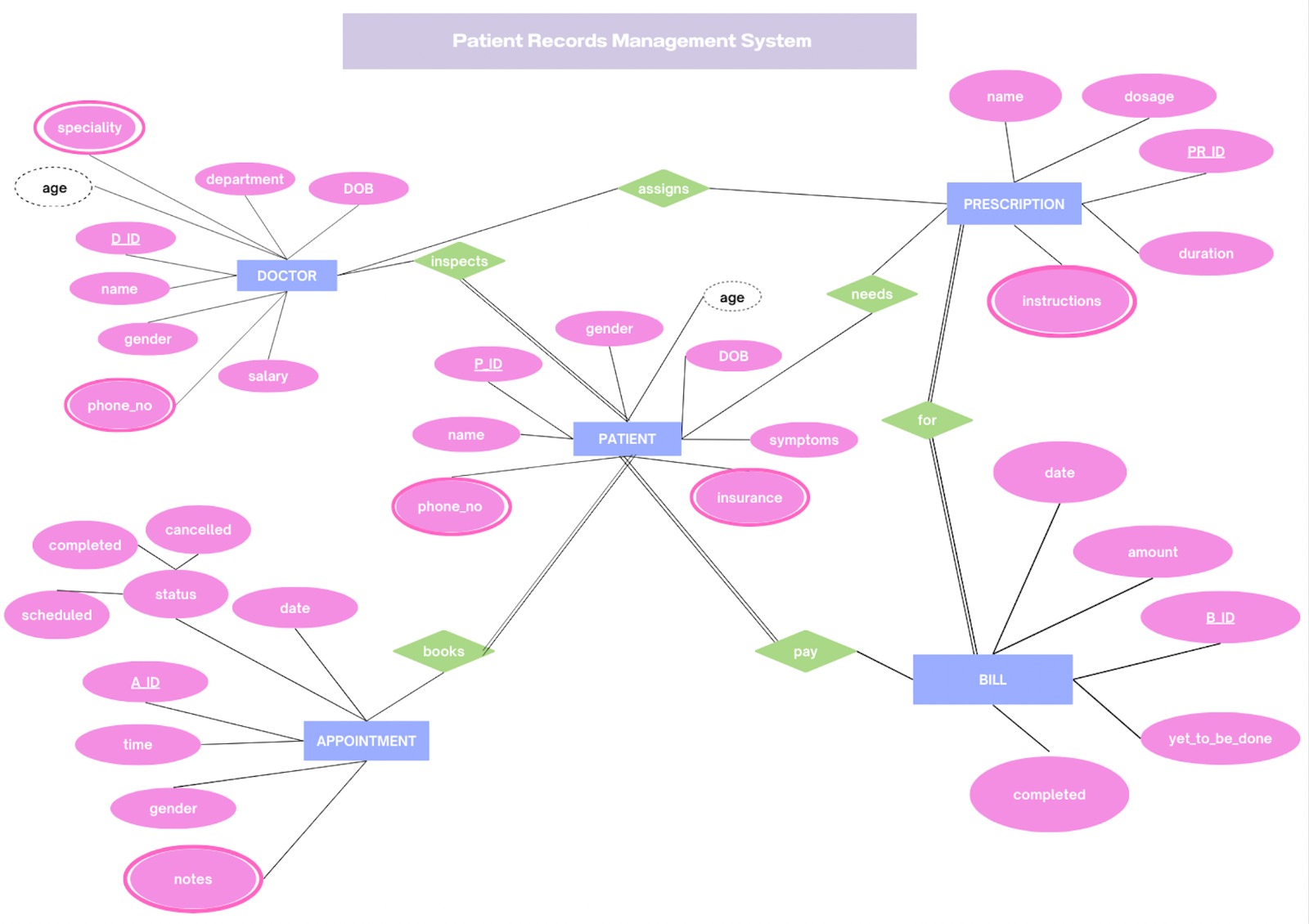
RUNNING THE CODE FILE:

**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:

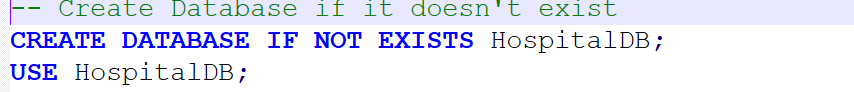


RELATIONAL SCHEMA:

DATABASE CREATION:

CREATE DATABASE IF NOT EXISTS HospitalDB;

USE HospitalDB;



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),

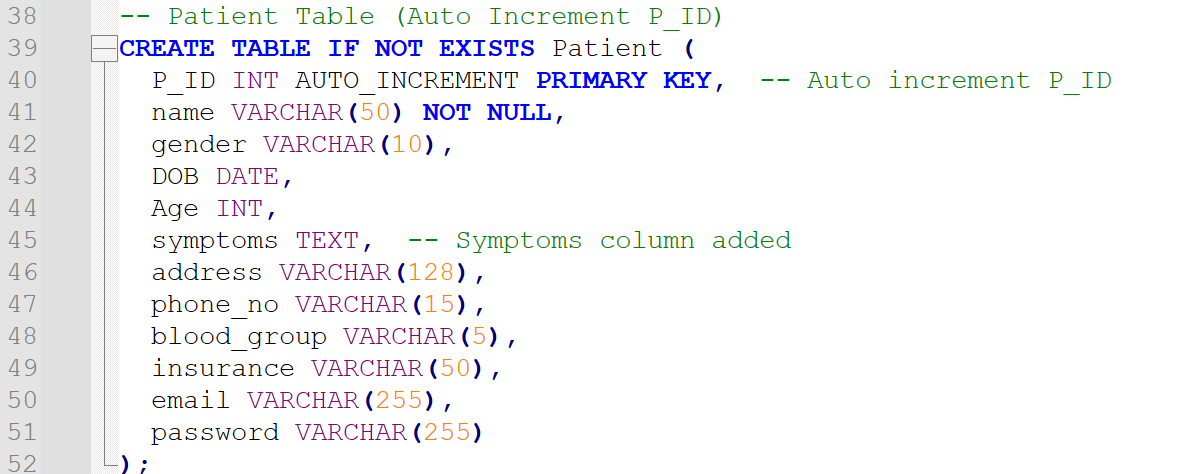
blood\_group VARCHAR(5),

insurance VARCHAR(50),

email VARCHAR(255),

password VARCHAR(255)

);



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),

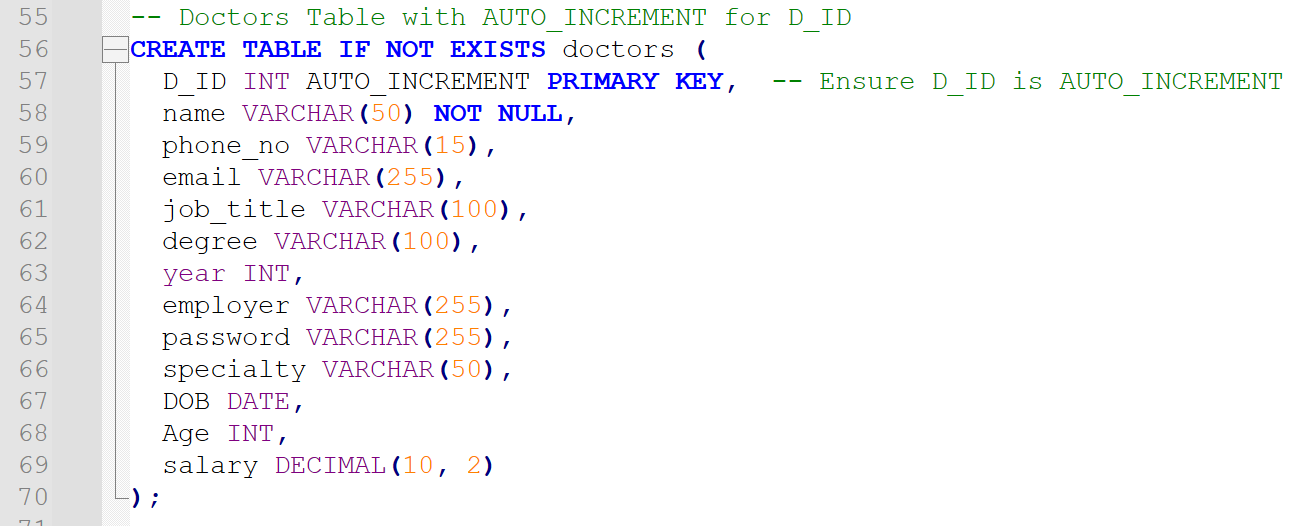
specialty VARCHAR(50),

DOB DATE,

Age INT,

salary DECIMAL(10, 2)

);



3) Administrator Table

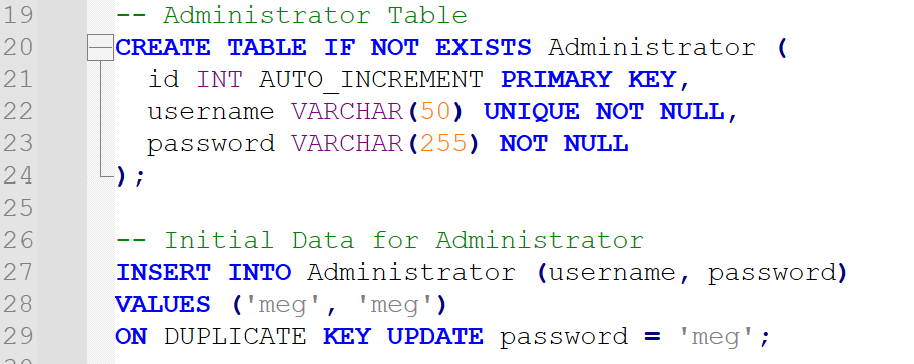
CREATE TABLE IF NOT EXISTS Administrator (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL

);



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

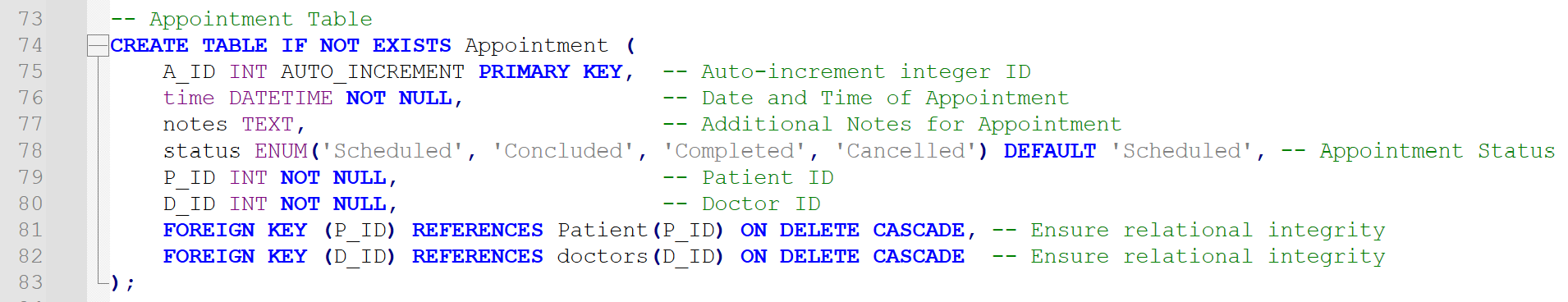
P\_ID INT NOT NULL, -- Patient ID

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

);



5) Prescription Table

CREATE TABLE IF NOT EXISTS Prescription (

Pr\_ID VARCHAR(20) PRIMARY KEY,

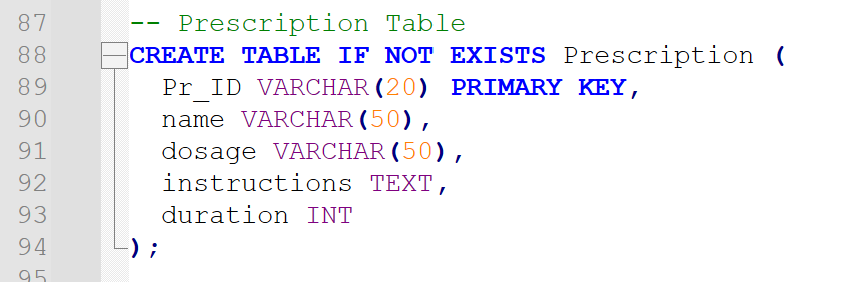
name VARCHAR(50),

dosage VARCHAR(50),

instructions TEXT,

duration INT

);



6) Bill Table

CREATE TABLE IF NOT EXISTS Bill (

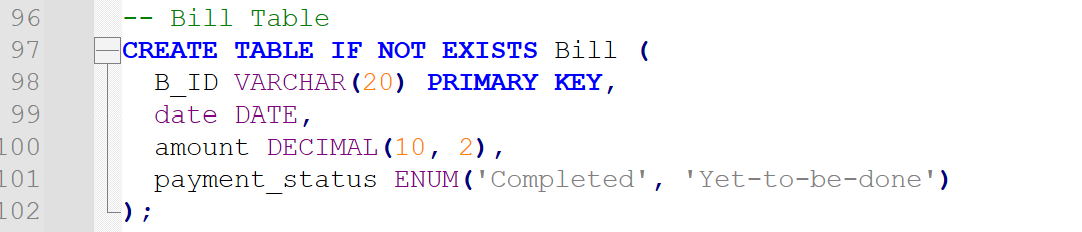
B\_ID VARCHAR(20) PRIMARY KEY,

date DATE,

amount DECIMAL(10, 2),

payment\_status ENUM('Completed', 'Yet-to-be-done')

);



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

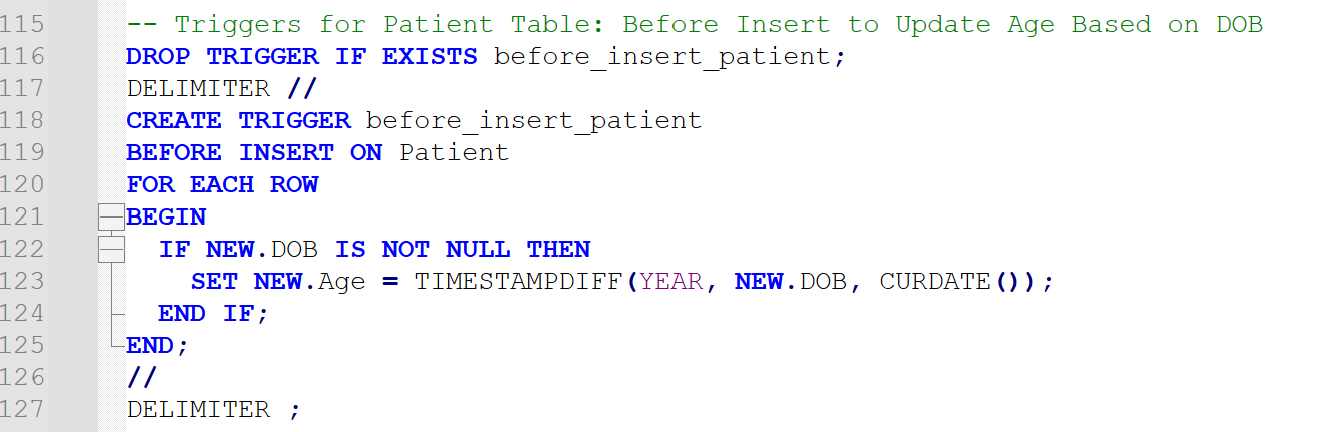
SET NEW.Age = TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE());

END IF;

END;

//

DELIMITER ;



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'

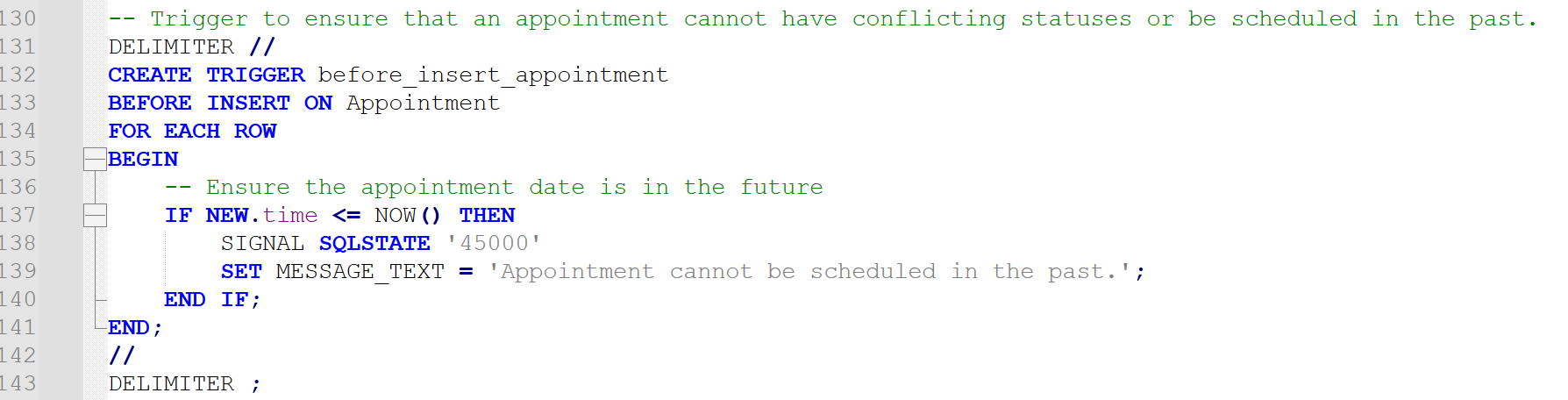
SET MESSAGE\_TEXT = 'Appointment cannot be scheduled in the past.';

END IF;

END;

//

DELIMITER ;



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

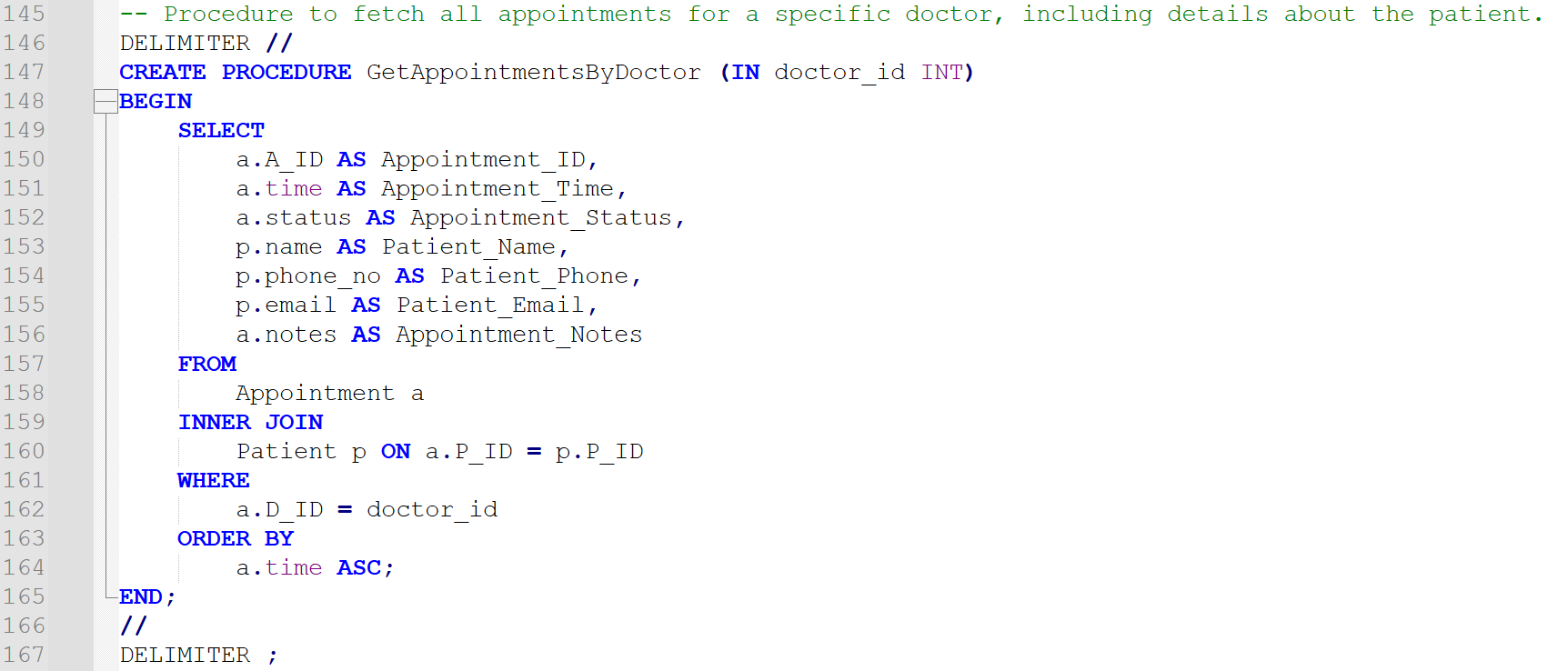
ORDER BY

a.time ASC;

END;

//

DELIMITER ;



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

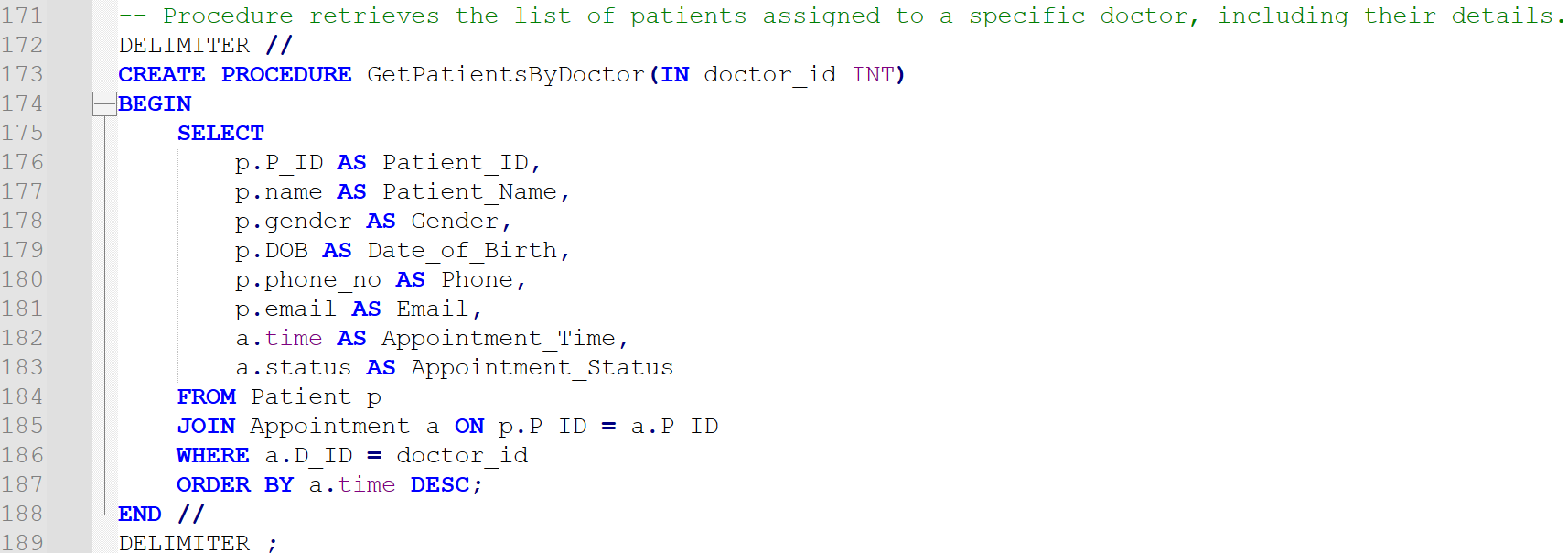
JOIN Appointment a ON p.P\_ID = a.P\_ID

WHERE a.D\_ID = doctor\_id

ORDER BY a.time DESC;

END //

DELIMITER ;



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

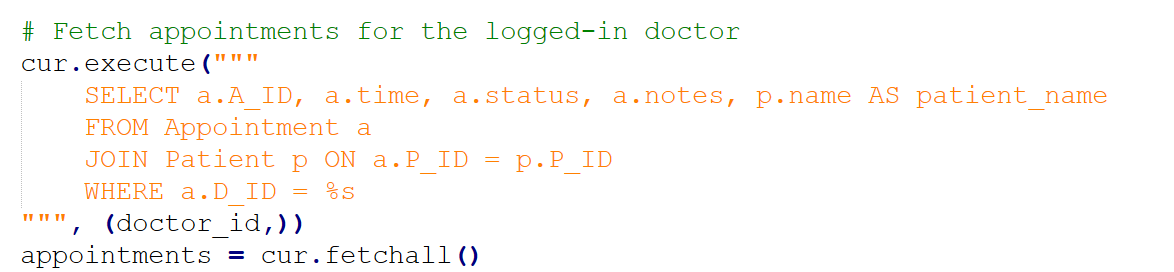
FROM Appointment a

JOIN Patient p ON a.P\_ID = p.P\_ID

WHERE a.D\_ID = %s

""", (doctor\_id,))

appointments = cur.fetchall()



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

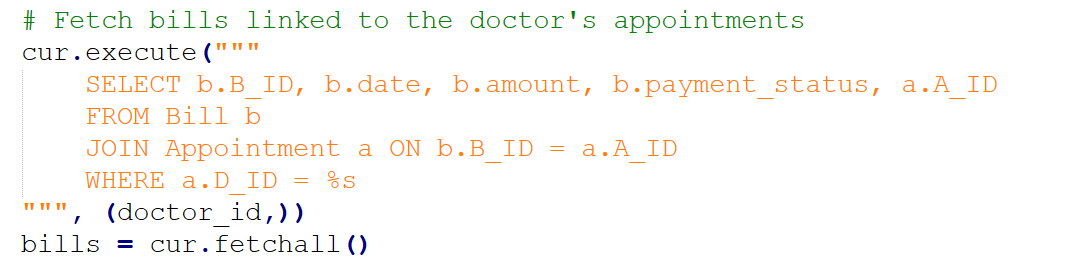
FROM Bill b

JOIN Appointment a ON b.B\_ID = a.A\_ID

WHERE a.D\_ID = %s

""", (doctor\_id,))

bills = cur.fetchall()



3) Editing and updating doctor info

cur.execute("""

UPDATE doctors

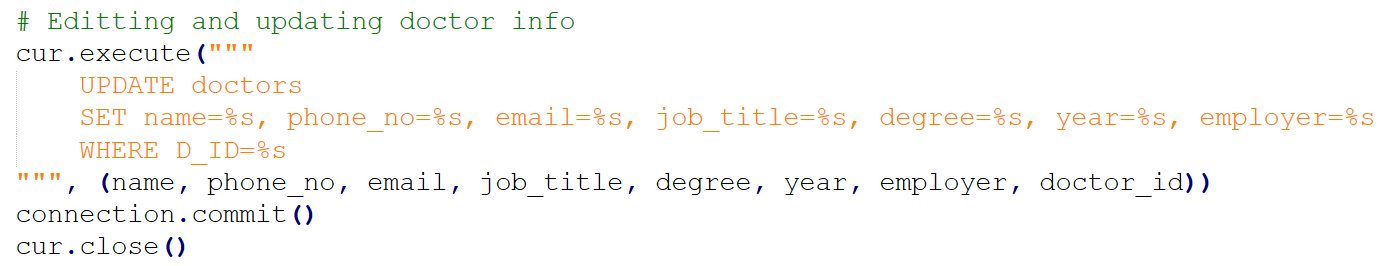
SET name=%s, phone\_no=%s, email=%s, job\_title=%s, degree=%s, year=%s, employer=%s

WHERE D\_ID=%s

""", (name, phone\_no, email, job\_title, degree, year, employer, doctor\_id))

connection.commit()

cur.close()



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

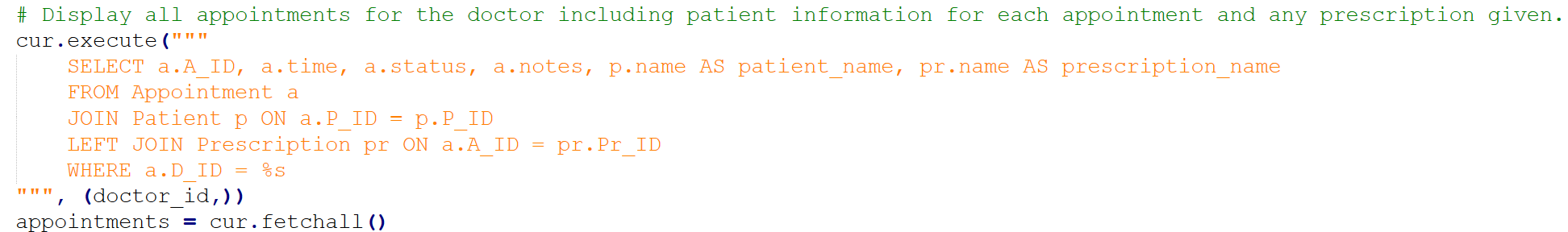
FROM Appointment a

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,))



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

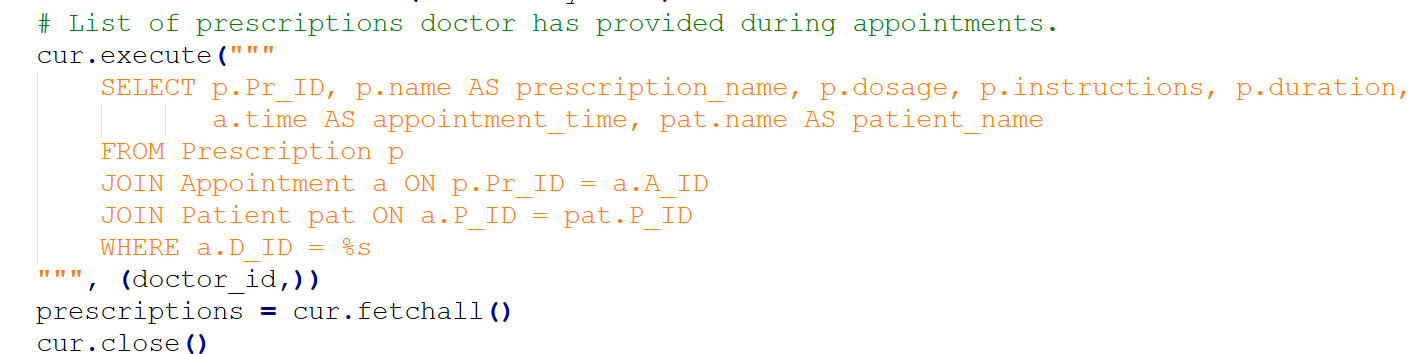
JOIN Patient pat ON a.P\_ID = pat.P\_ID

WHERE a.D\_ID = %s

""", (doctor\_id,))

prescriptions = cur.fetchall()

cur.close()



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,

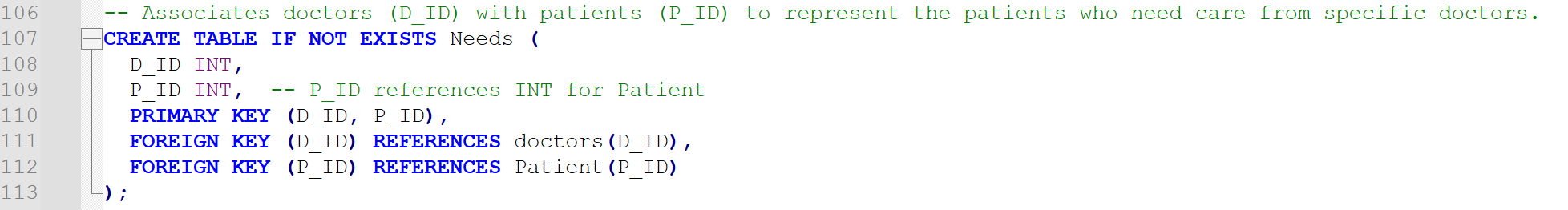
P\_ID INT, -- P\_ID references INT for Patient

PRIMARY KEY (D\_ID, P\_ID),

FOREIGN KEY (D\_ID) REFERENCES doctors(D\_ID),

FOREIGN KEY (P\_ID) REFERENCES Patient(P\_ID)

);



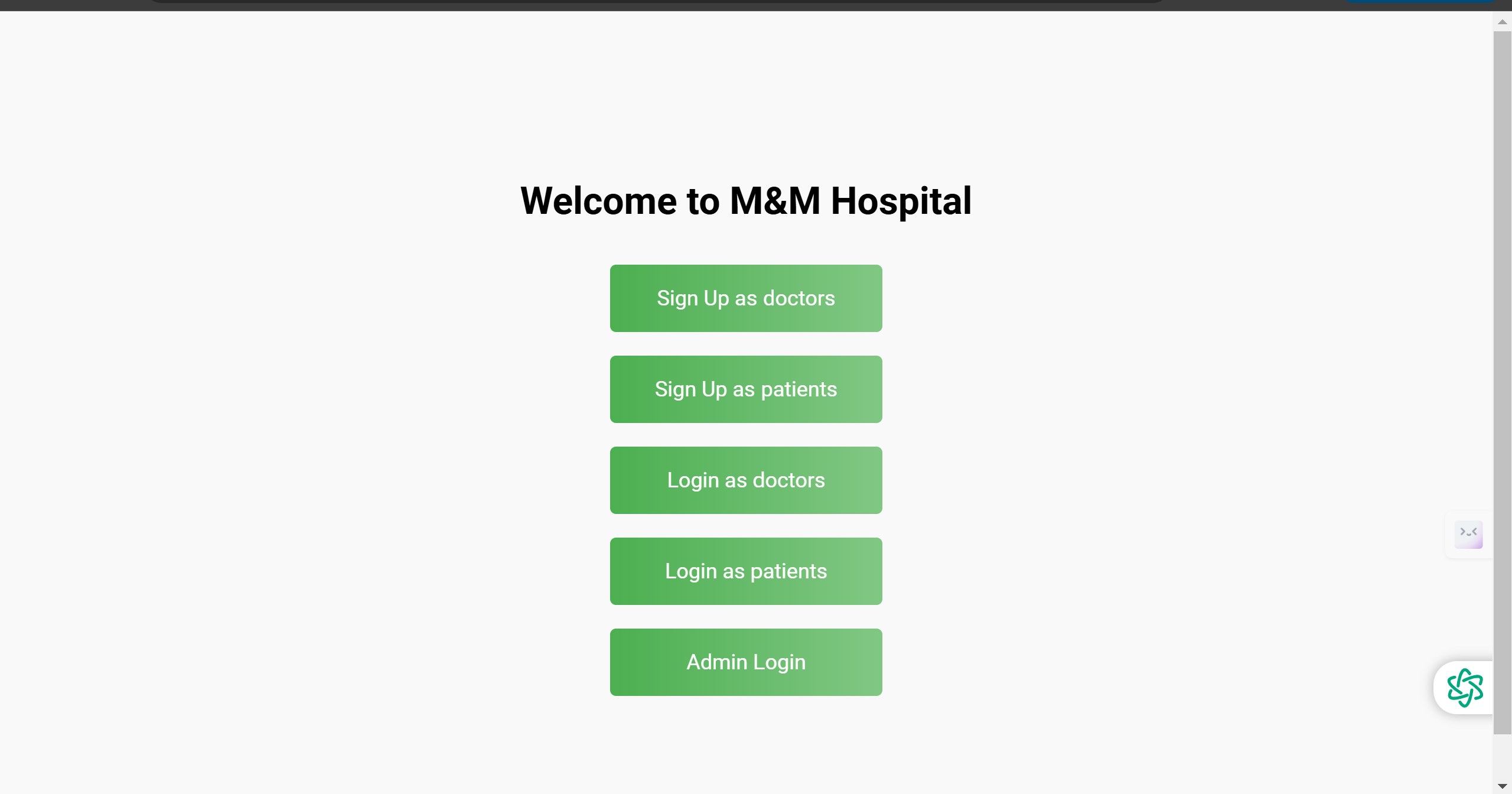
HTML PART:

1) home.html

Home page for the user – doctor, admin and patient

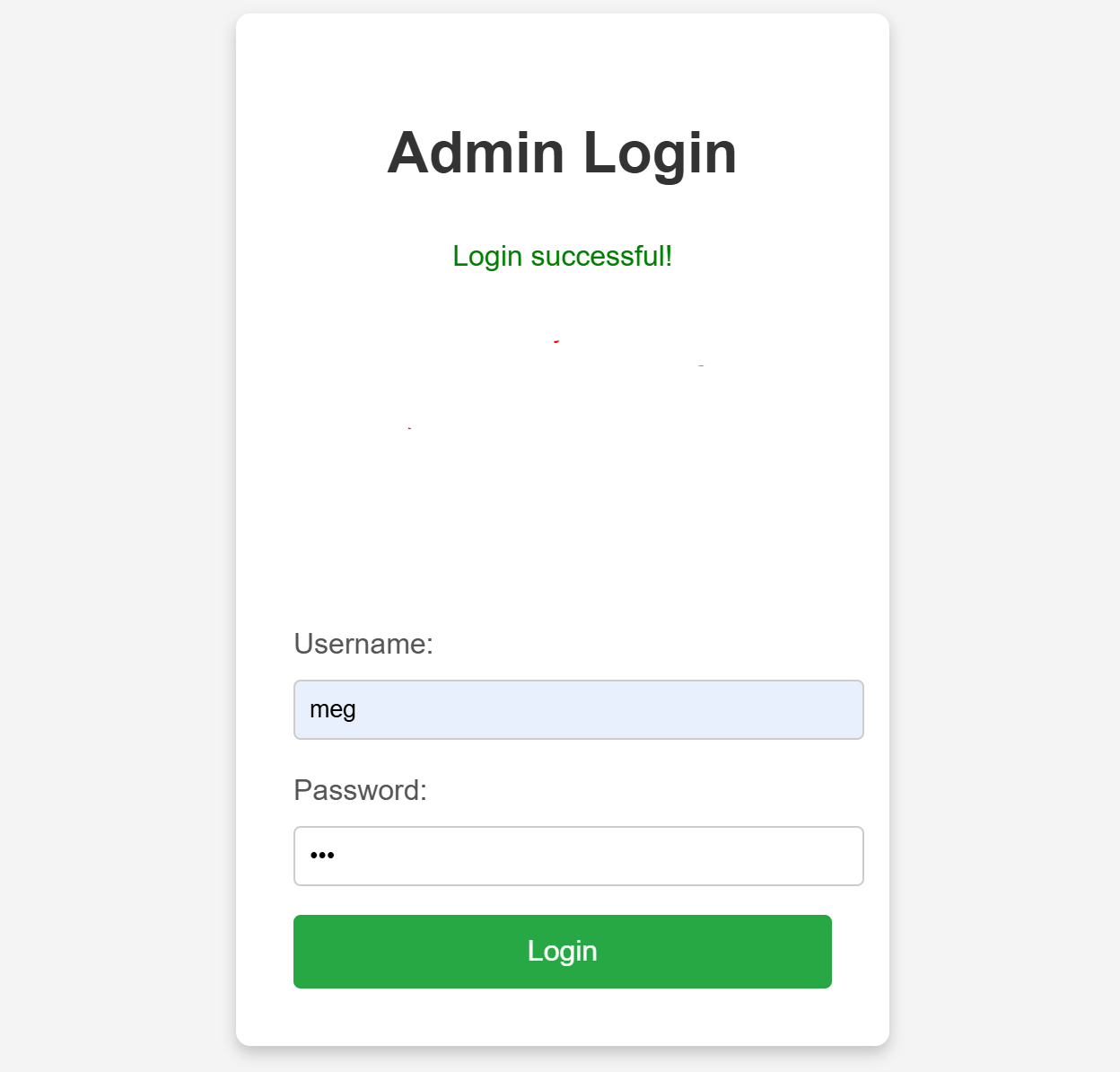
2) index.html

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

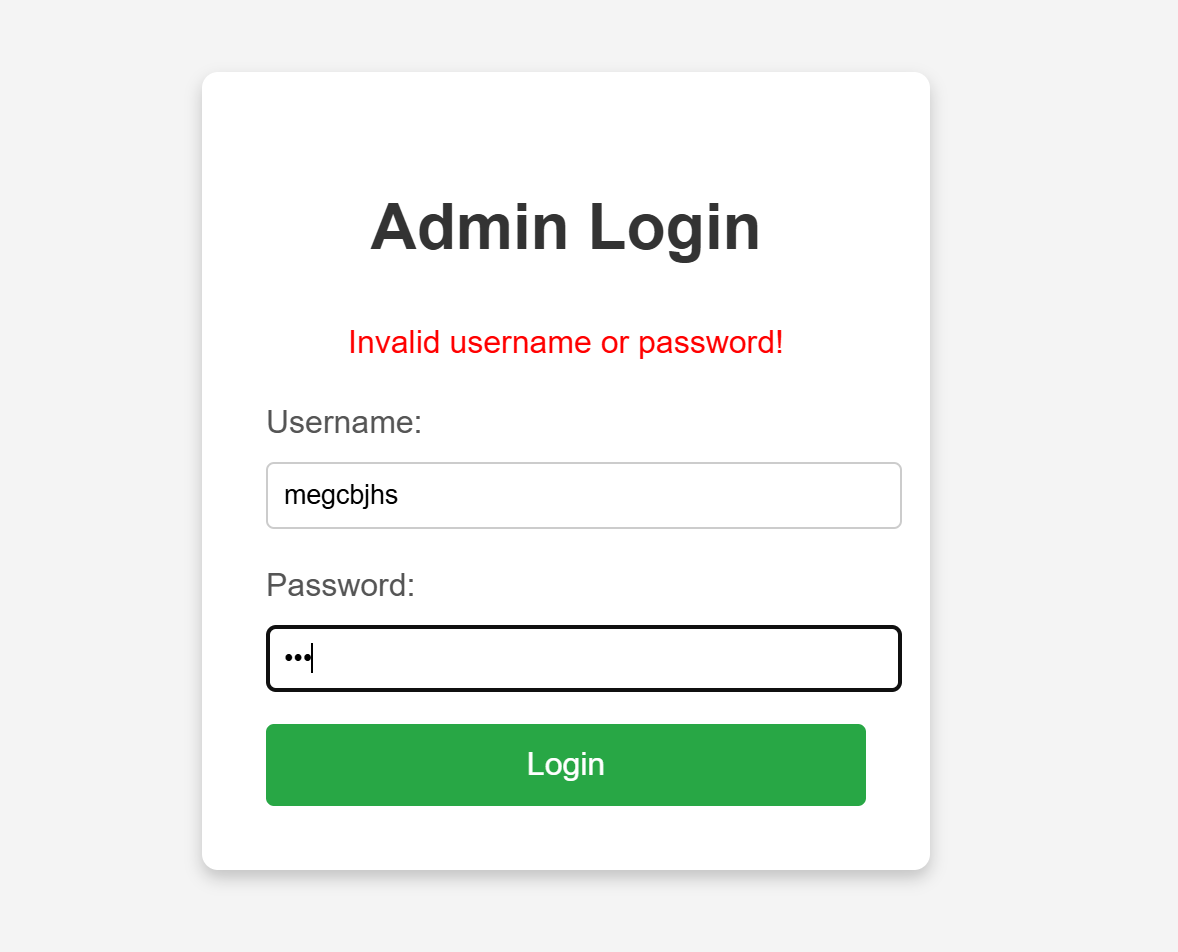


3) admin\_login.html

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

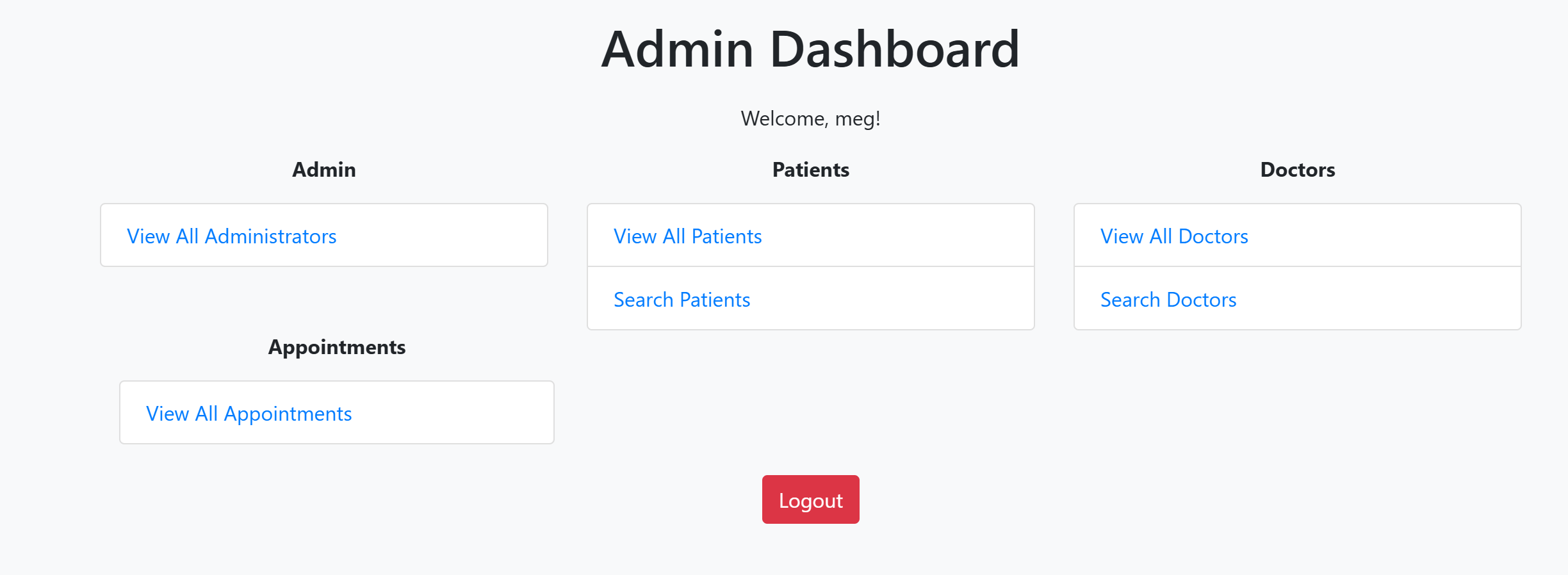


Invalid username/ password-



4) admin\_dashboard.html

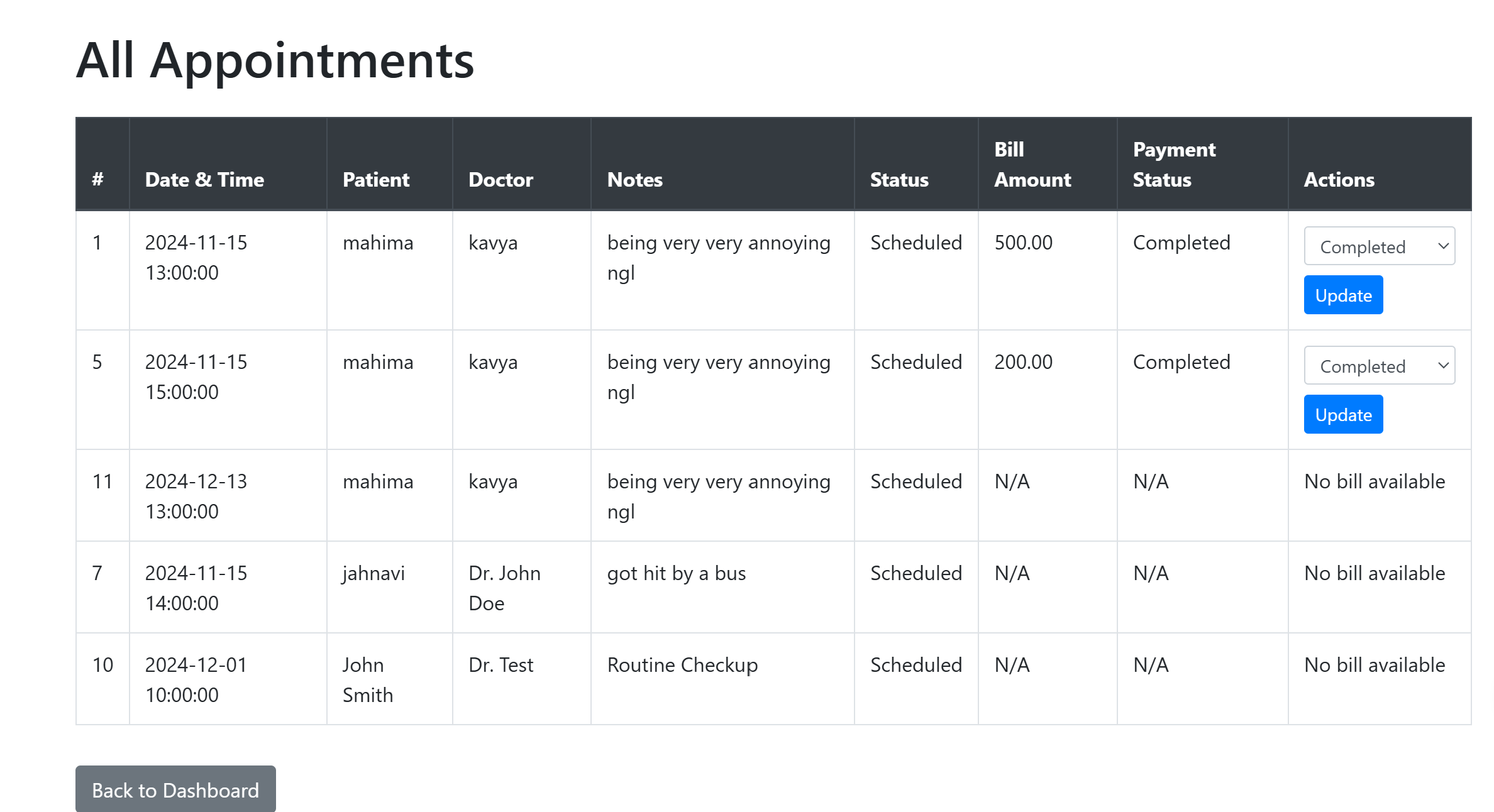
Page for the admin where they can click on the information they want to see like view and search for patients and doctors, view admins, and view all appointments made with their corresponding prescriptions and bills.



5) admin\_view\_appointments.html

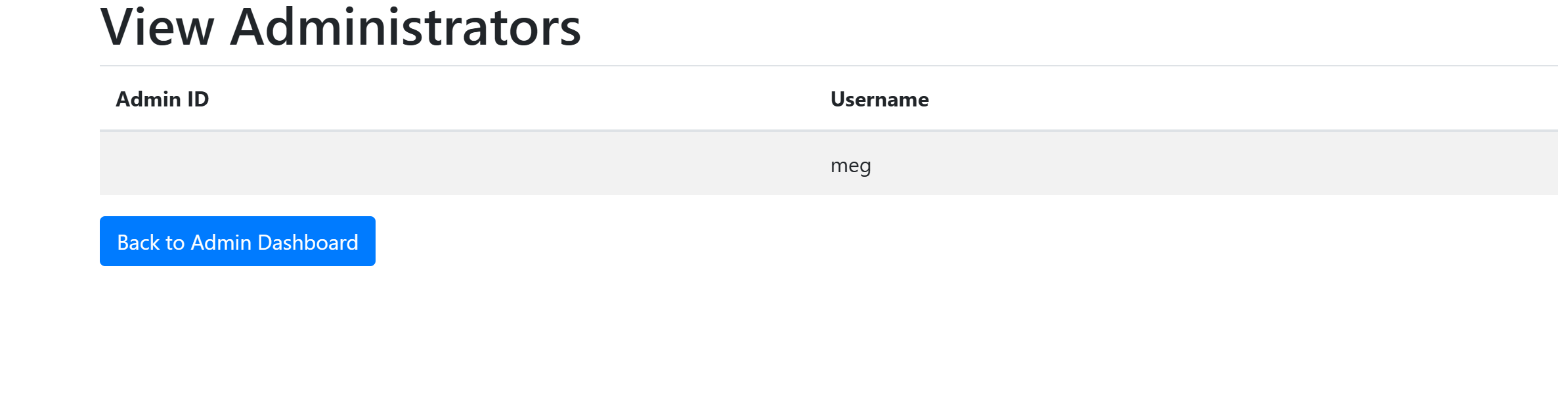
Page where the admin can view all the appointments made by patients with the prescription and bill the doctor has provided.

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



6) view\_administrators.html

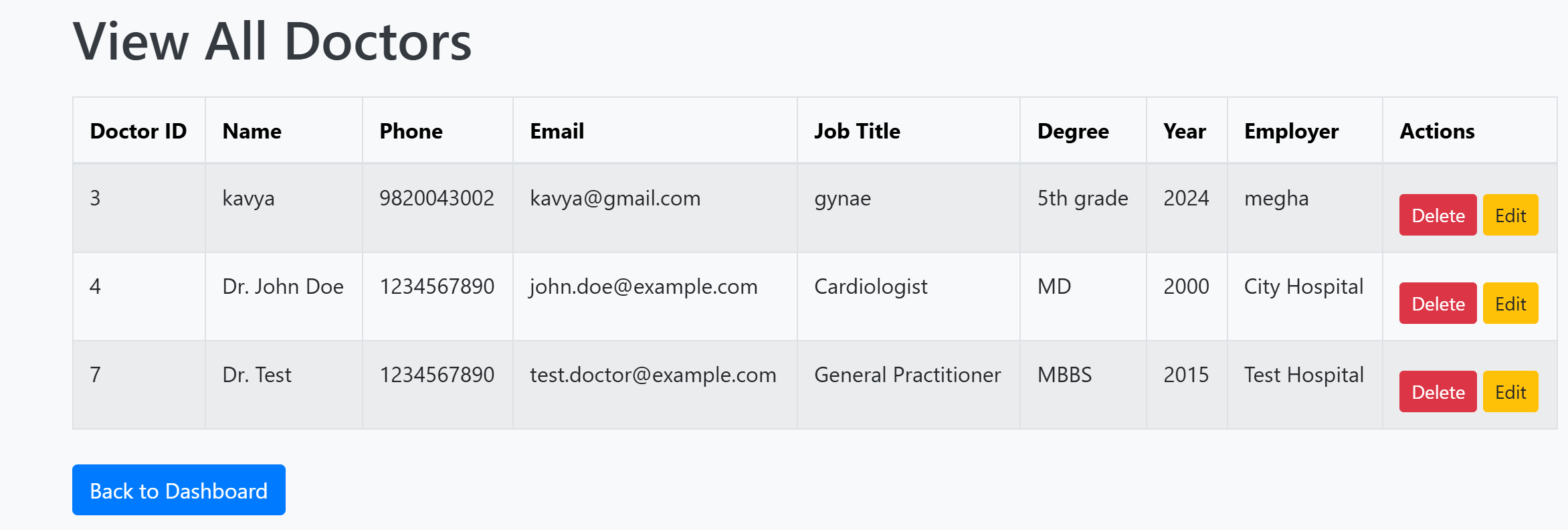
Admin can view all admins in the M&M Hospital here.



7) view\_doctors.html

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

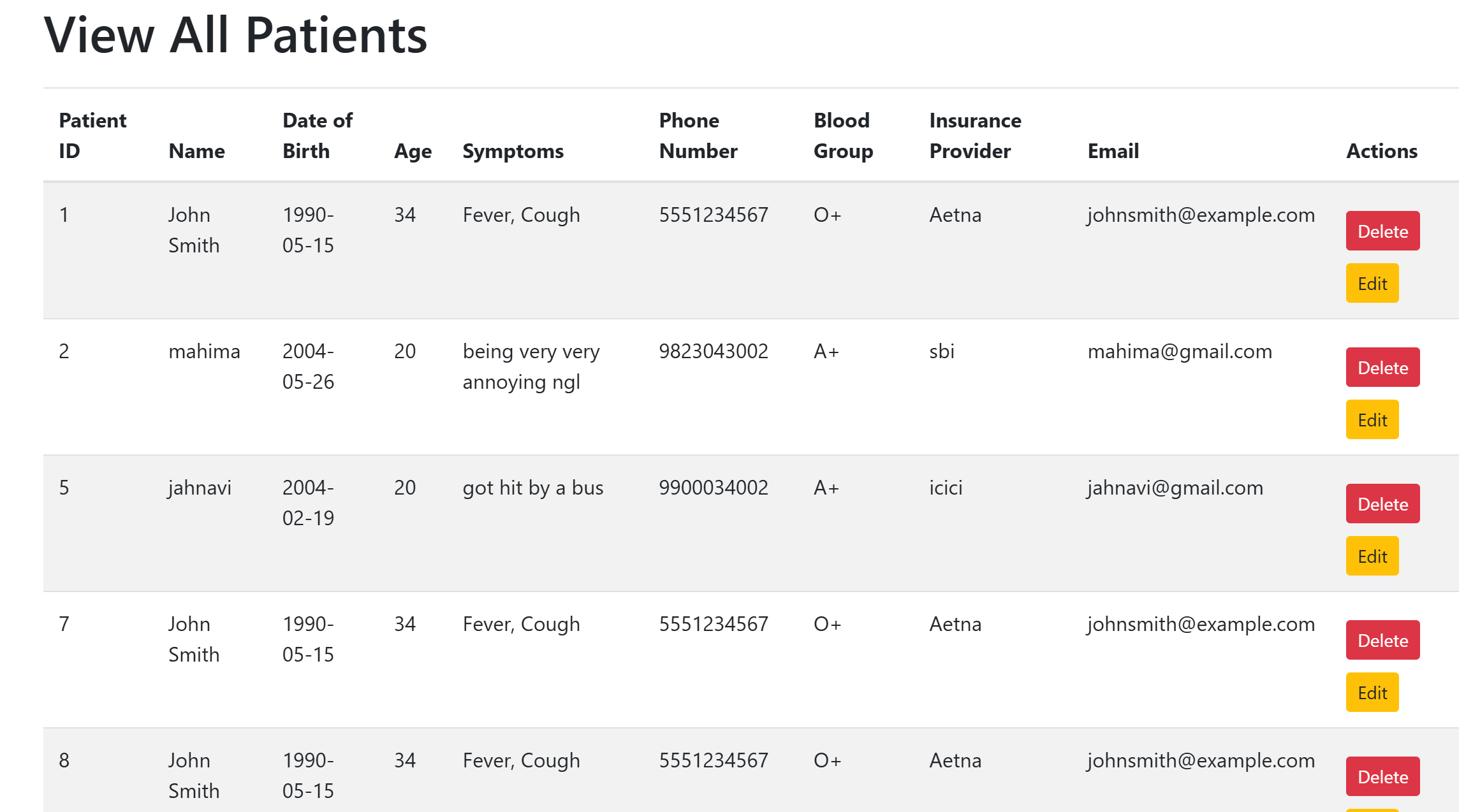
Admin can also edit or delete their information.



8) view\_patients.html:

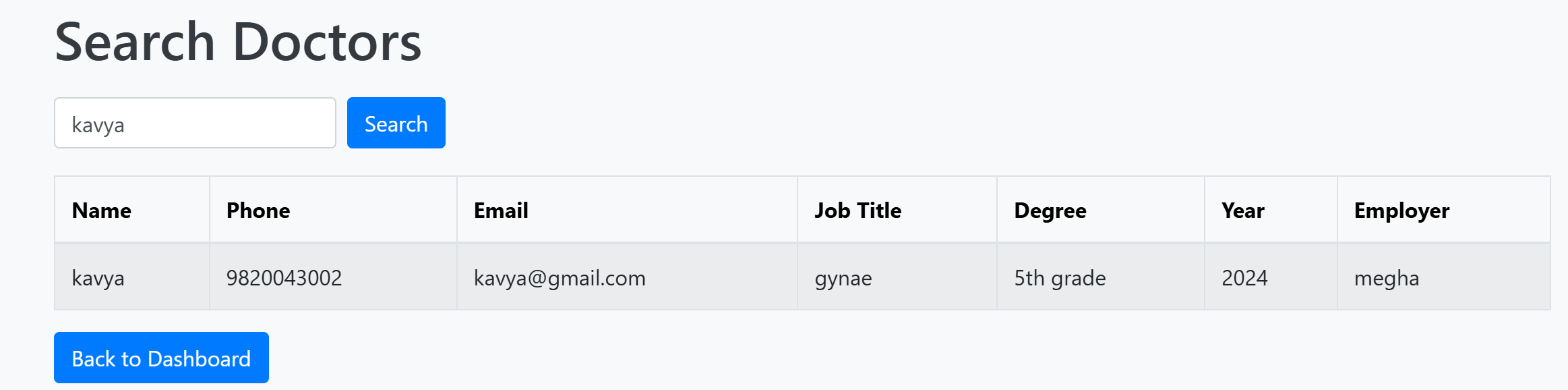
Admin can view all patients and their information in this page.

Admin can also edit or delete their information.



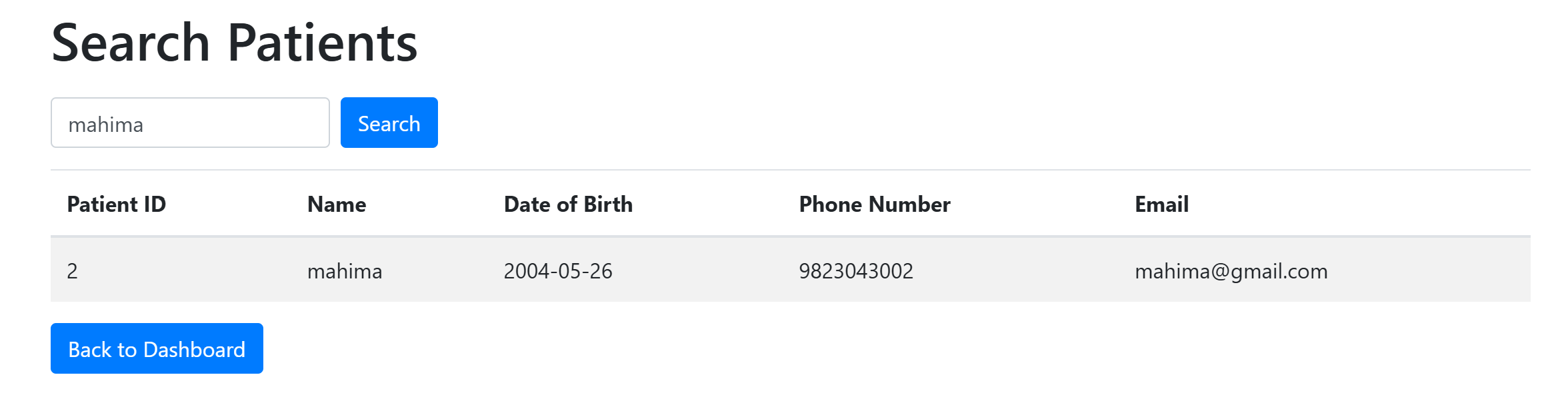
9) search\_doctors.html

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



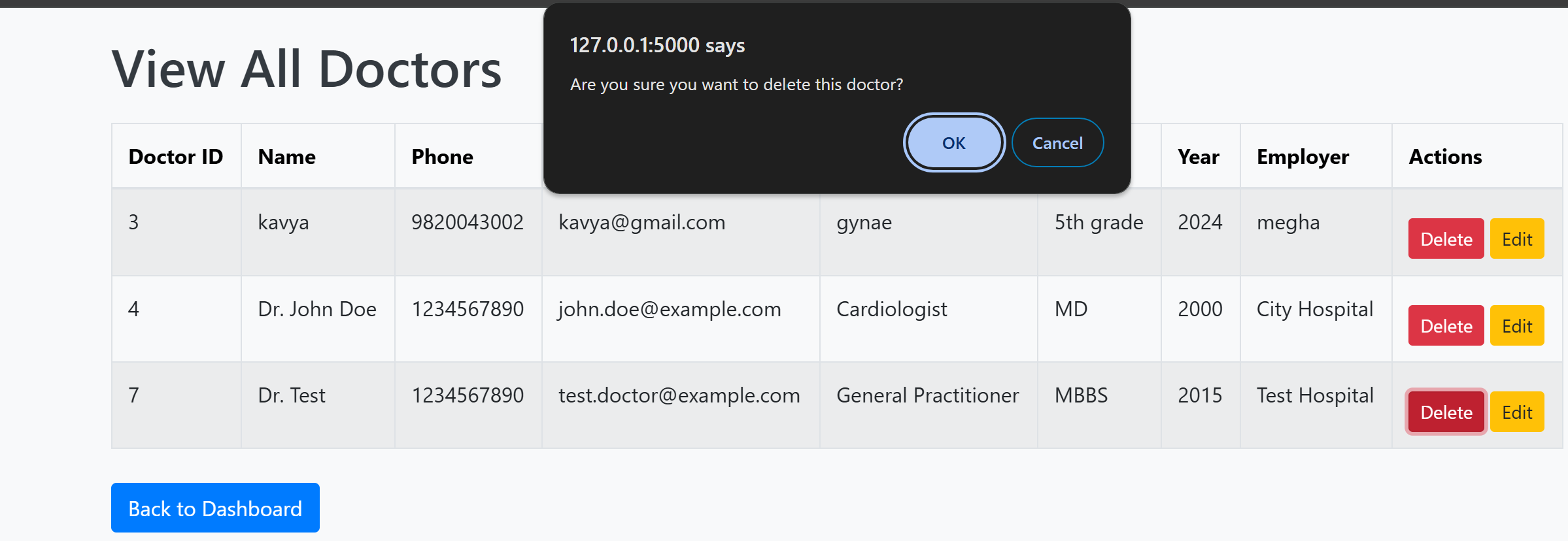
10) search\_patients.html

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



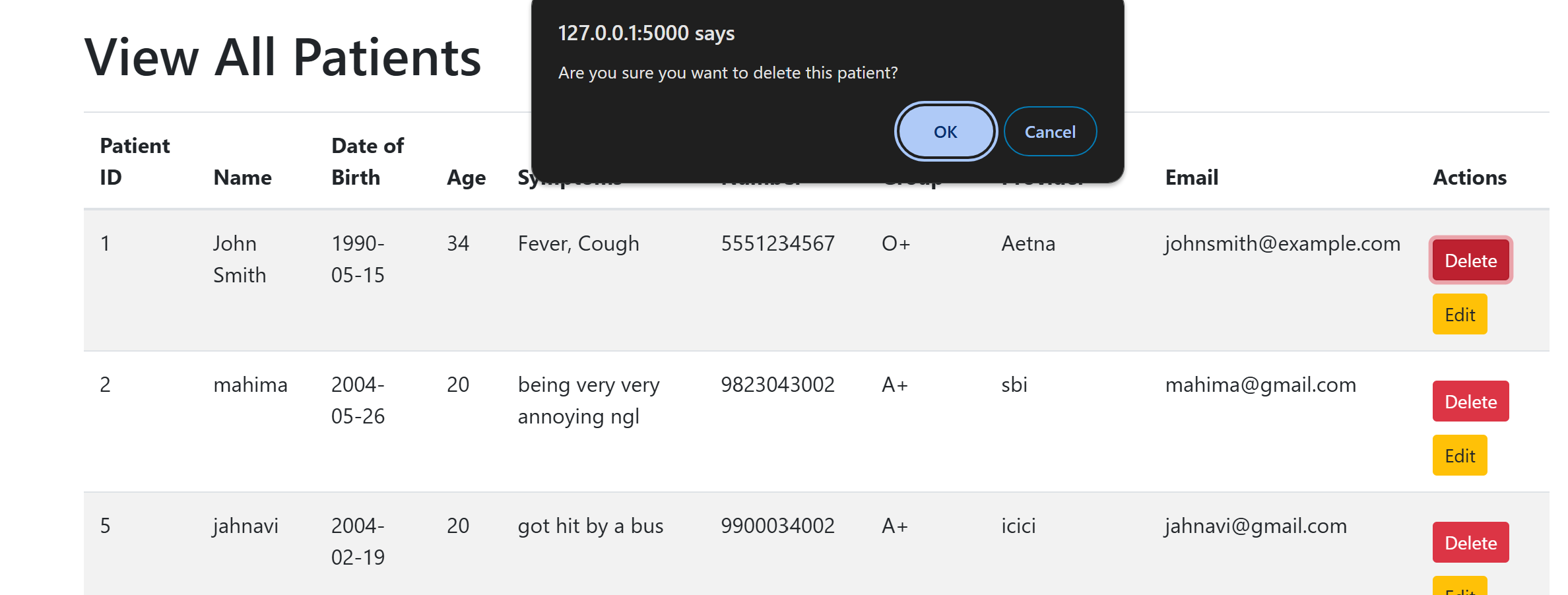
11) delete\_doctors.html

Helps the admin to delete doctor information in the view\_doctors page.



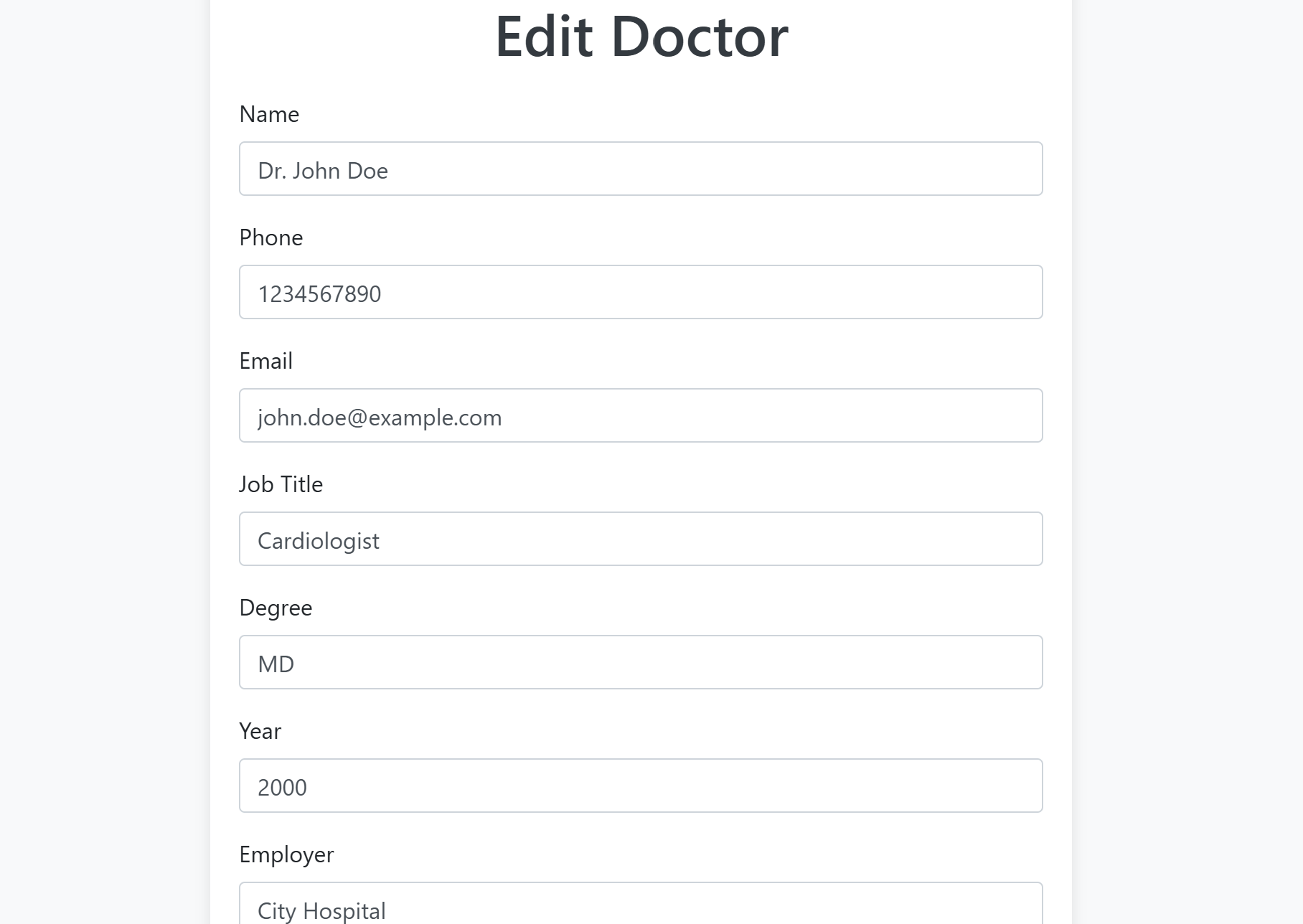
12) delete\_patients.html

Helps the admin to delete patient information in the view\_patients page.

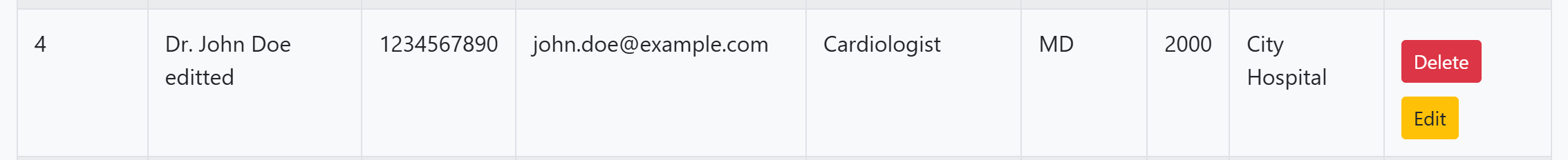


13) edit\_doctors.html

Helps the admin to edit doctor information in the view\_doctors page.

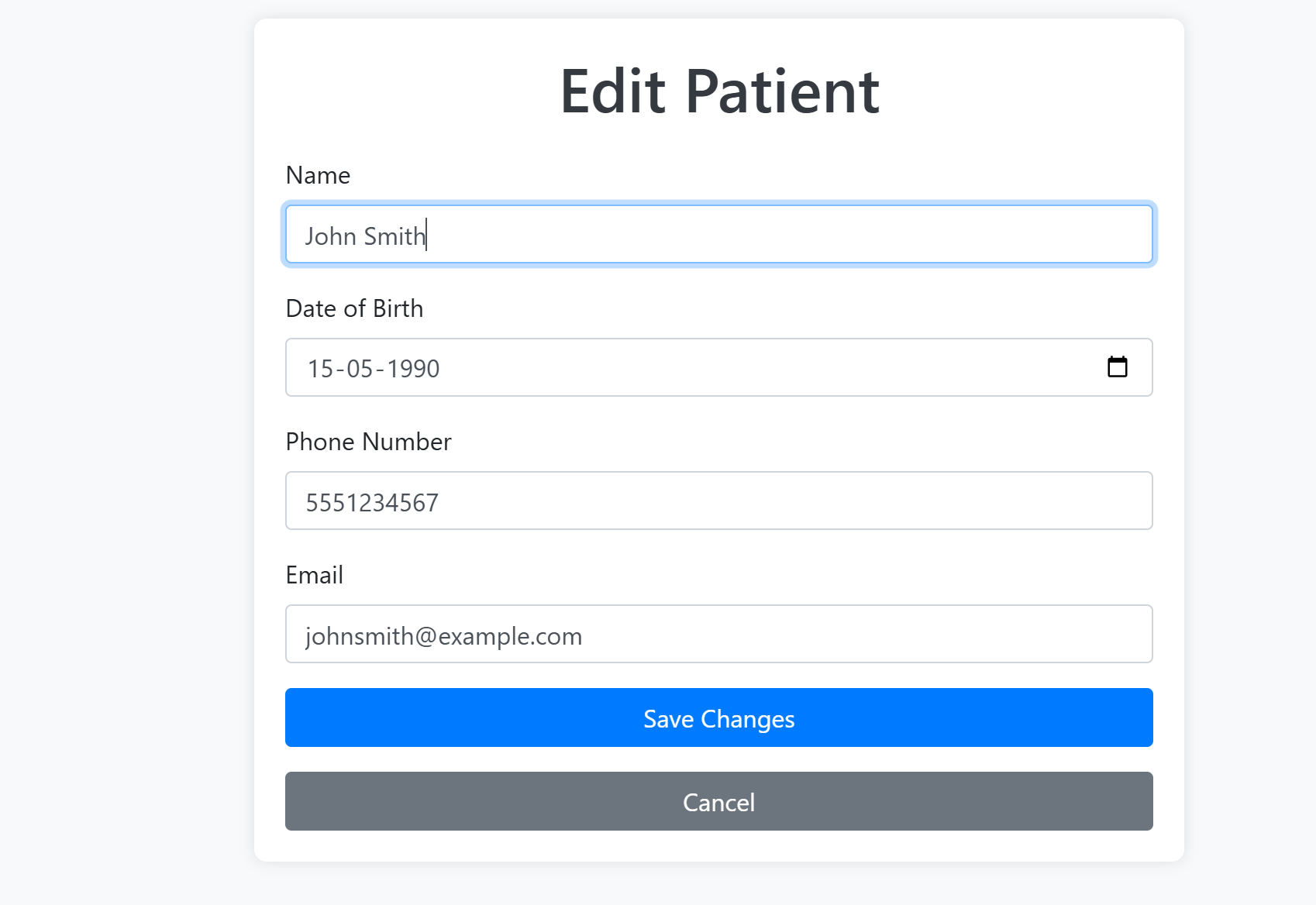


After editting-

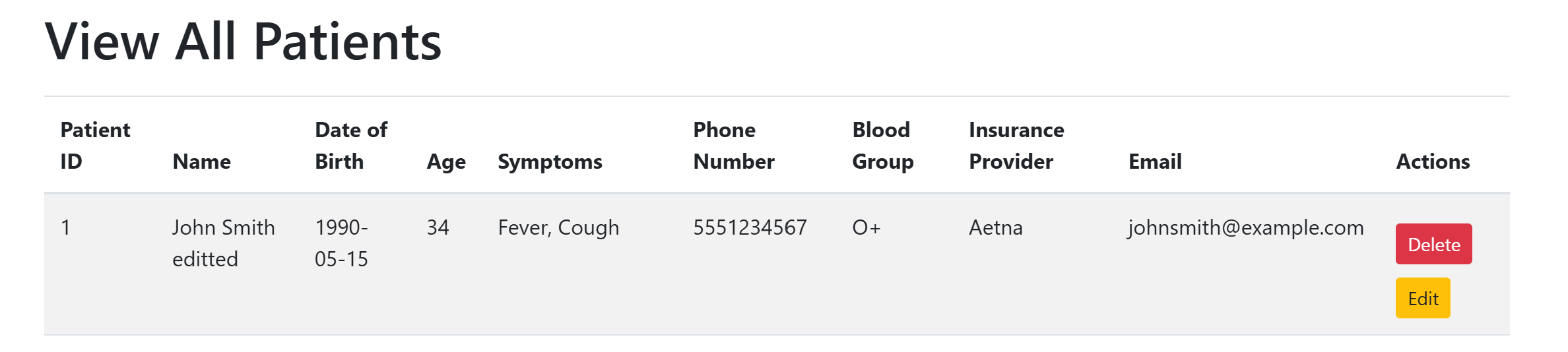


14) edit\_patients.html

Helps the admin to edit patient information in the view\_patients page.

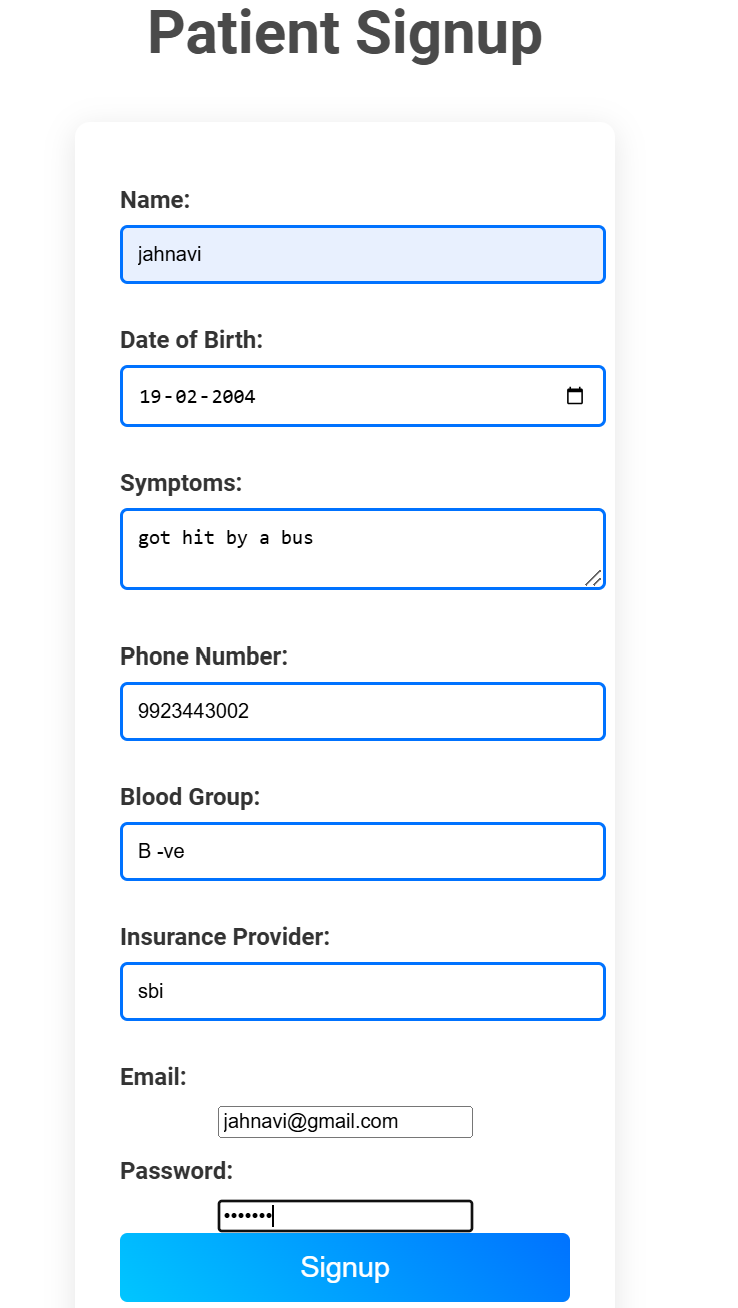


After editing-



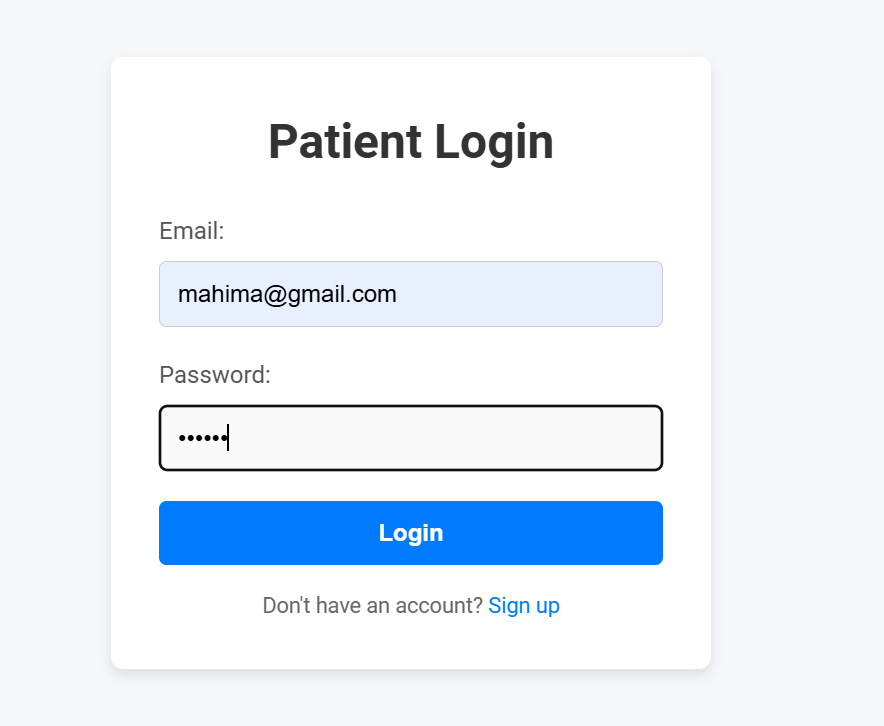
15) patients\_signup.html

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.



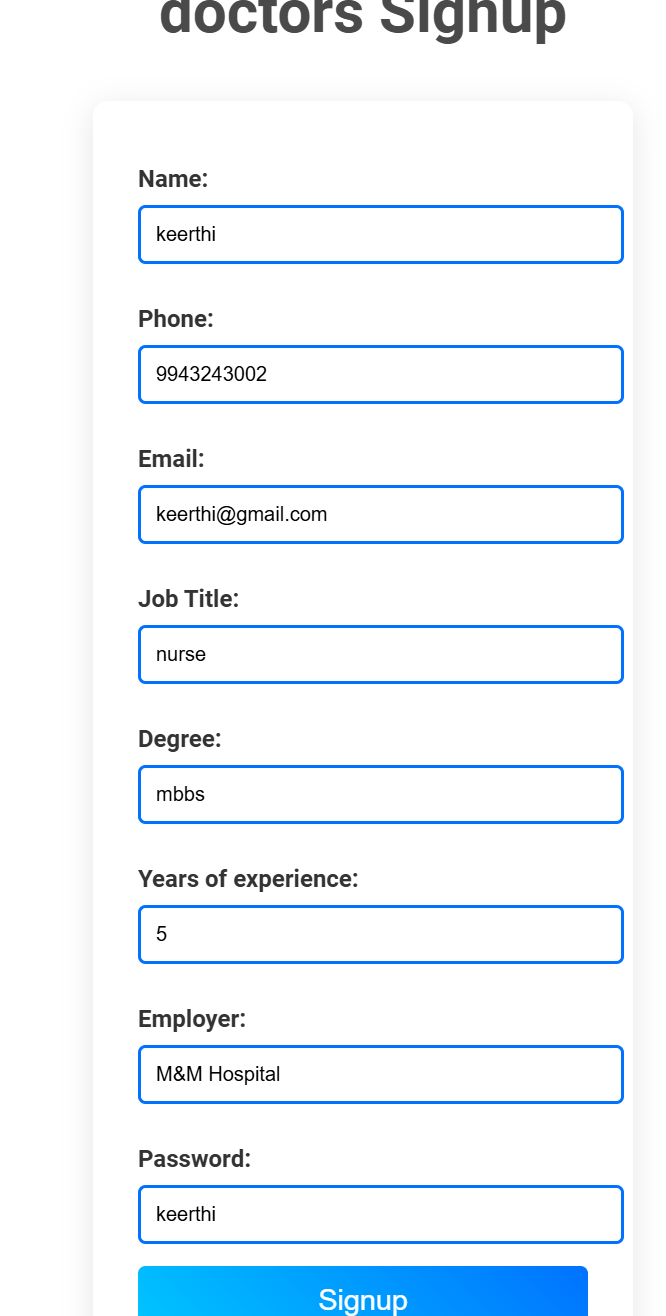
16) patients\_login.html

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



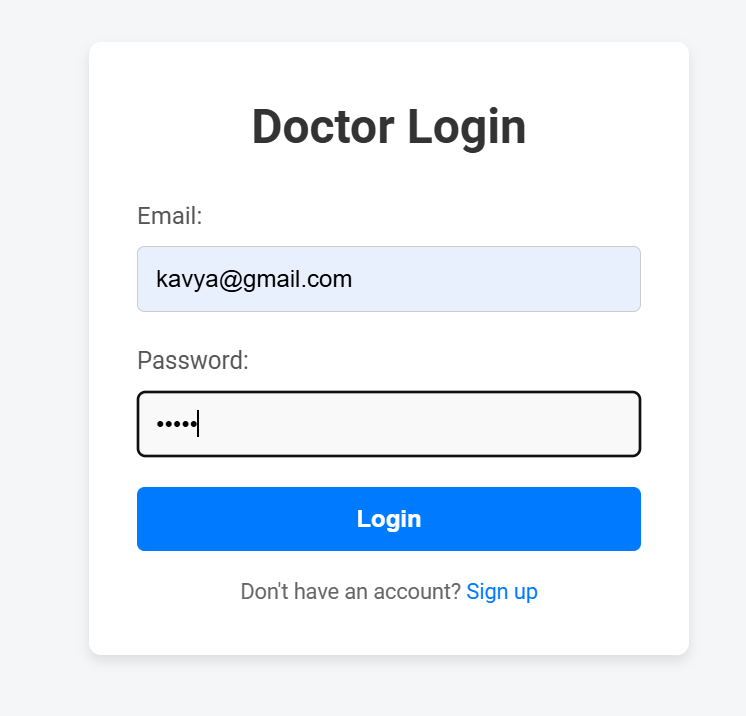
17) doctors\_signup.html

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.



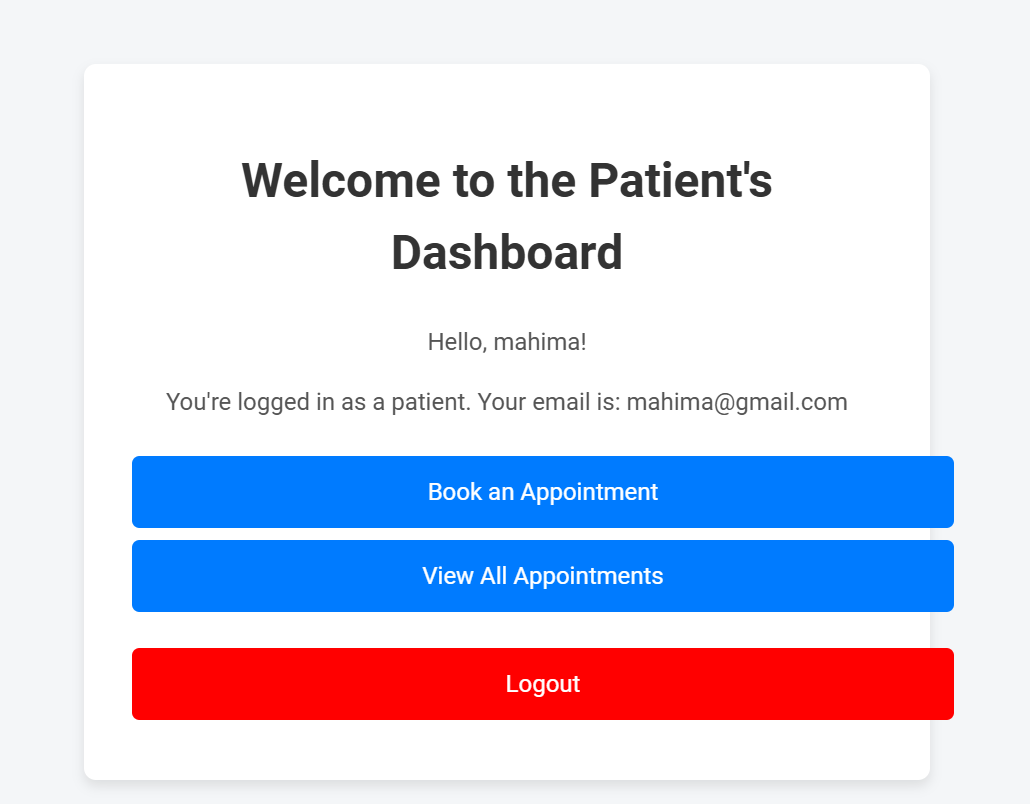
18) doctors\_login.html

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



19) patients\_welcome.html

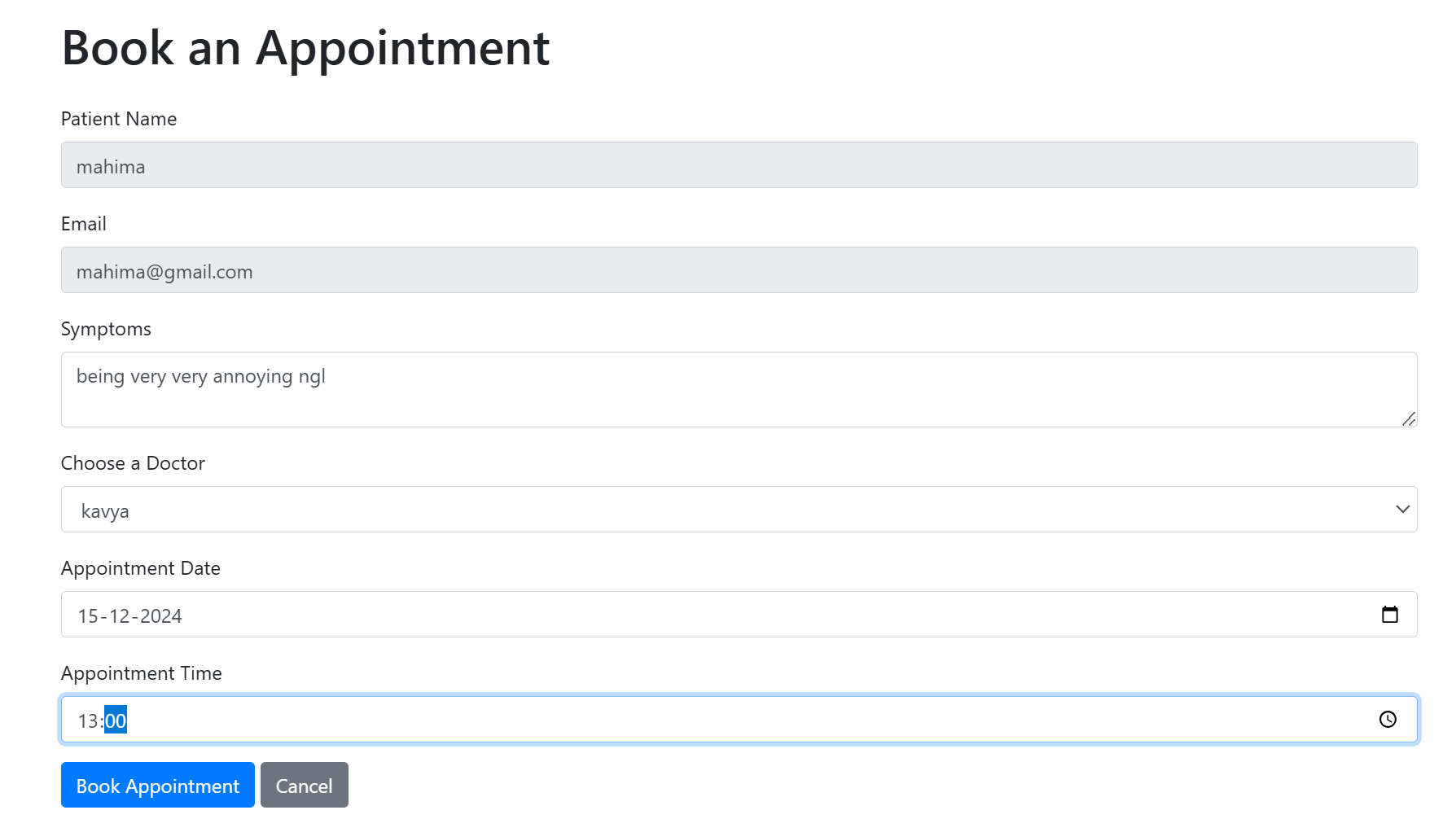
Patients can either click on ‘book an appointment’ or ‘cancel an appointment’ button here.



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.

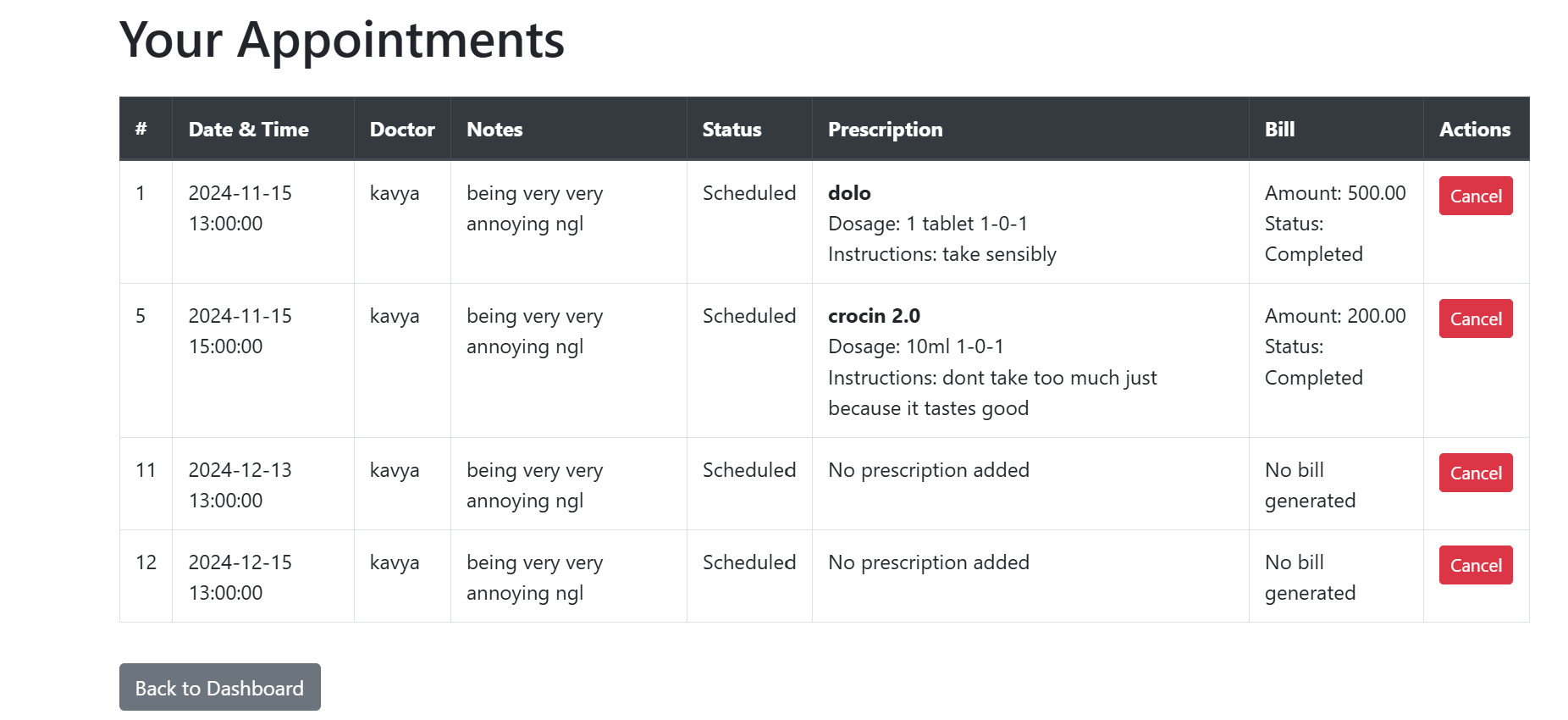


21) view\_app.html

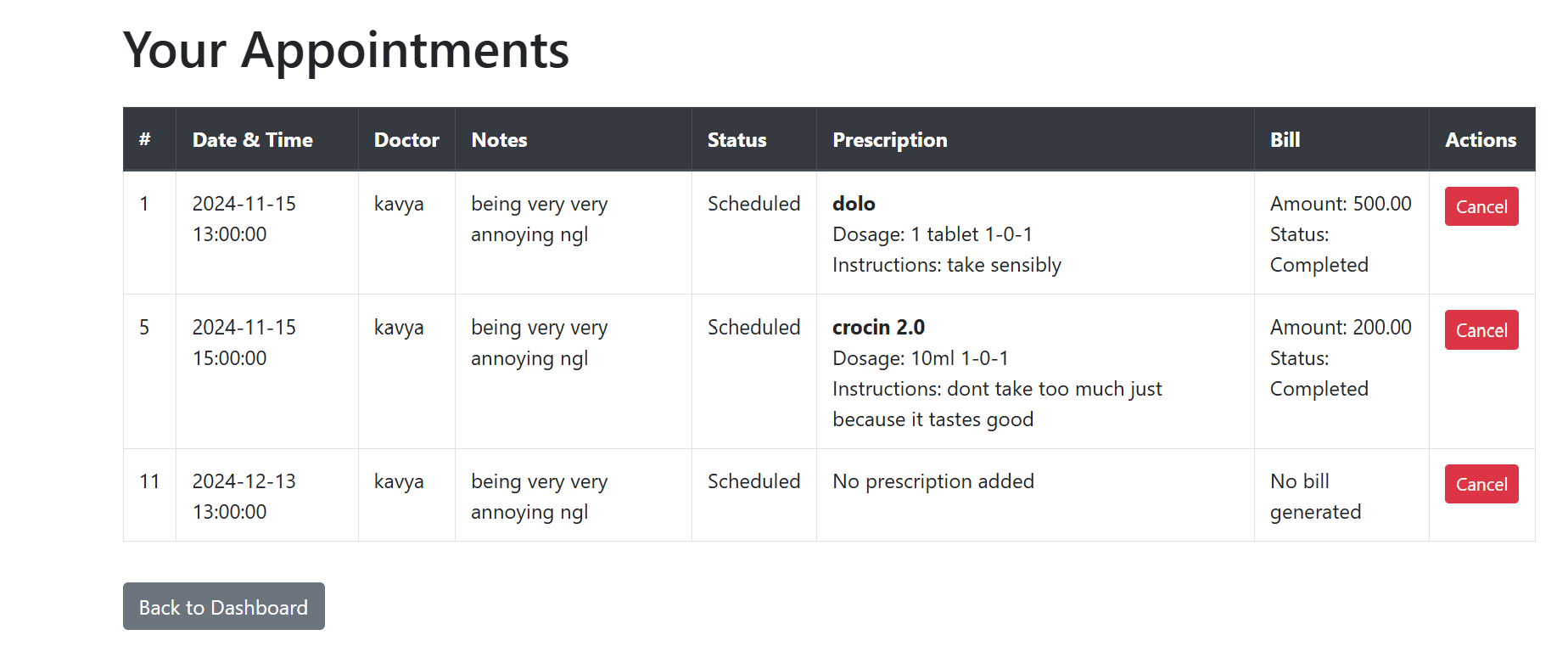
When the patients books an appointment they can view that information here.

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.

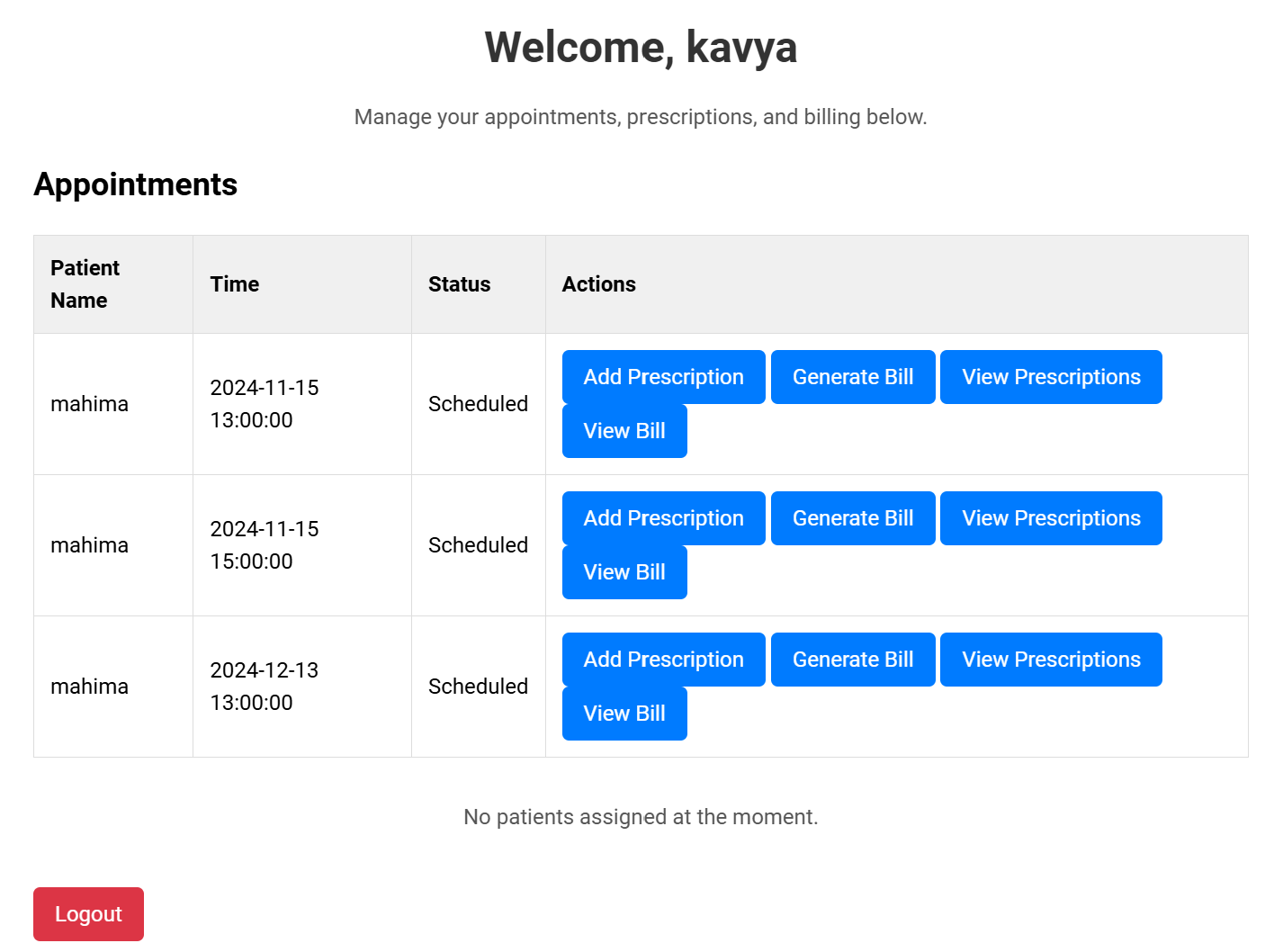


After patient cancels appointment-



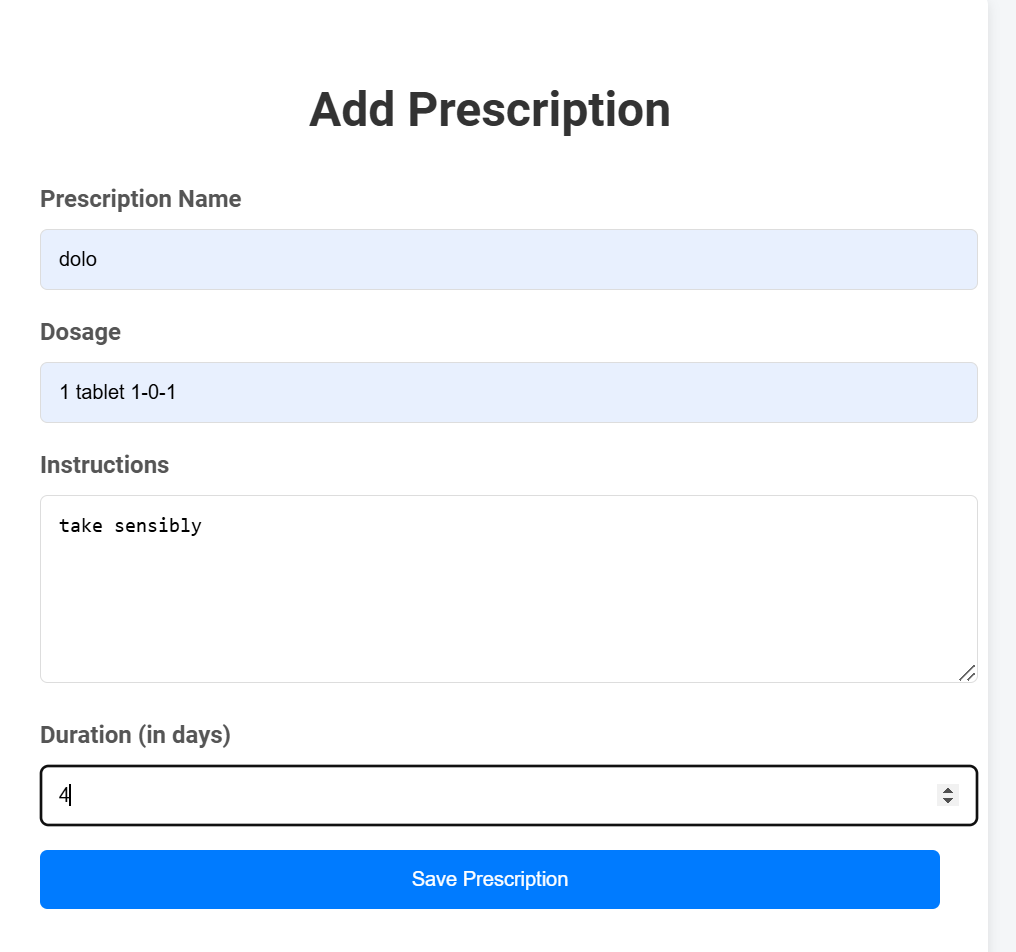
22) doctors\_welcome.html

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



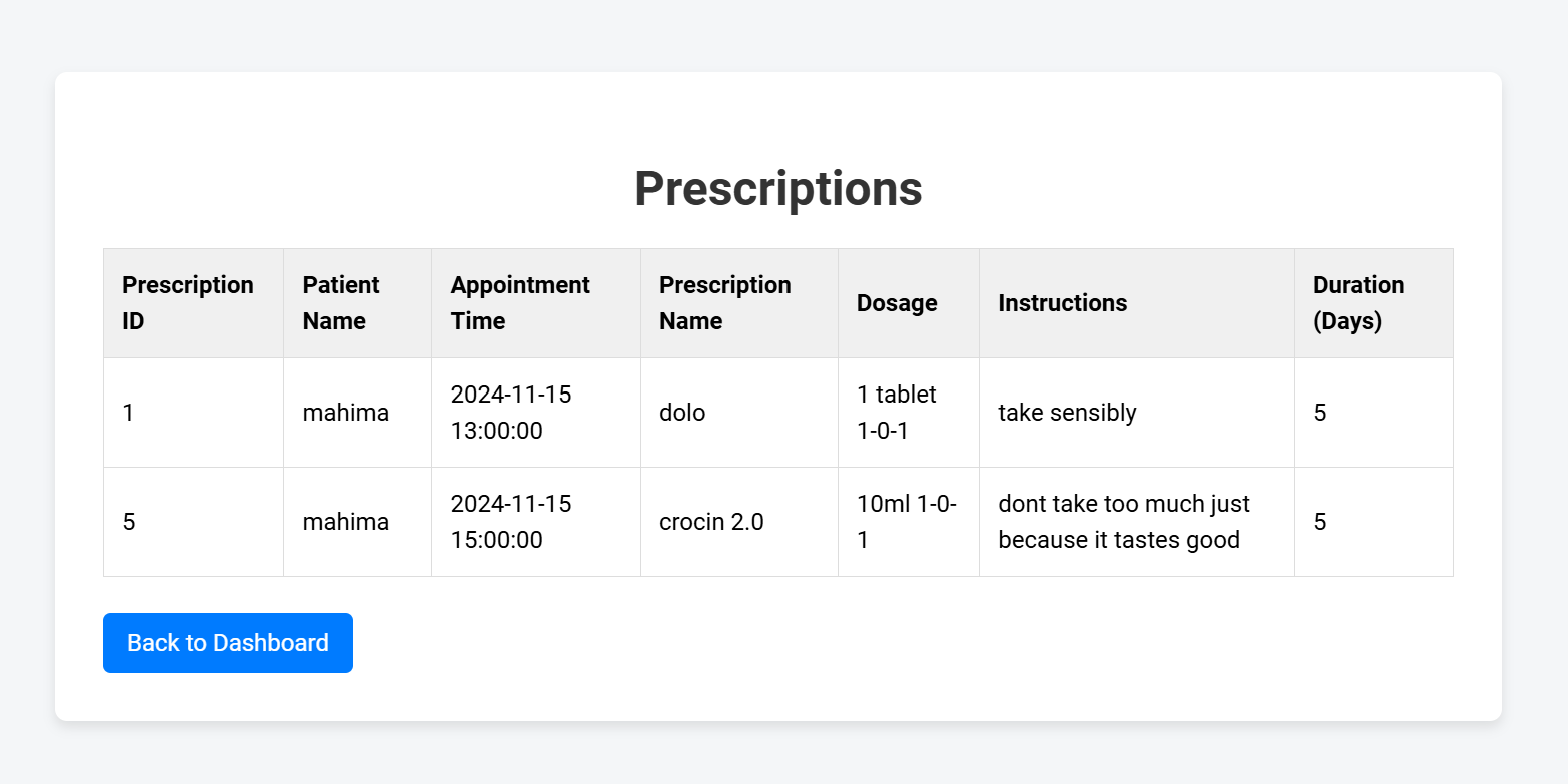
23) add\_prescription.html

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



24) view\_prescriptions.html

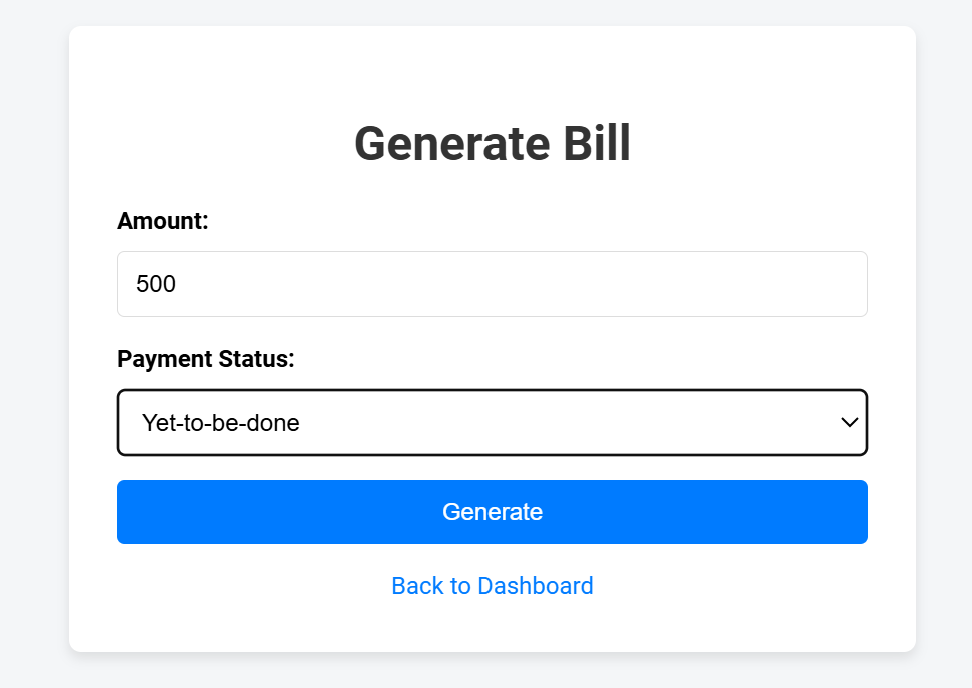
Doctors can view the prescription they have given to that patient.



25) generate\_bill.html

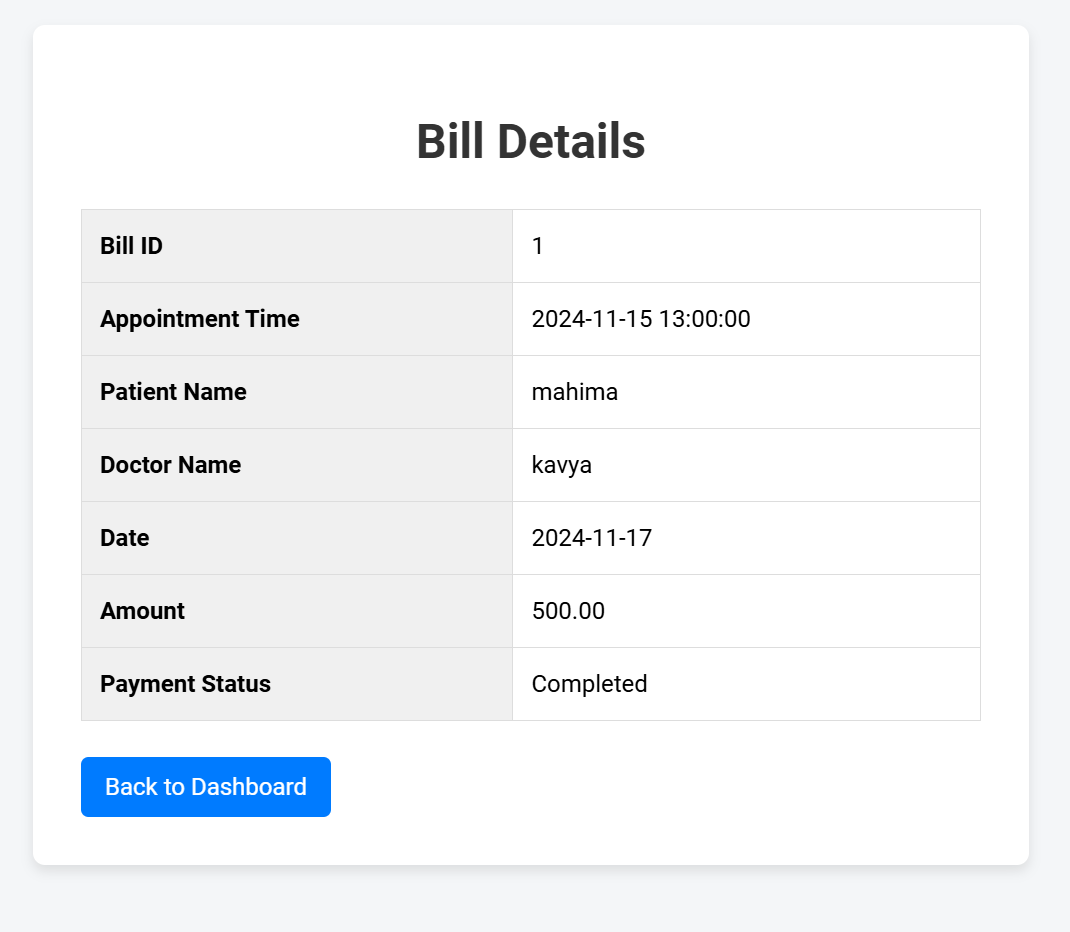
Doctors can give the amount to be paid by the patient here.

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