North South University

Department of Electrical & Computer Engineering

FALL 2025

Course Name: Database Management System Lab

Faculty Name: SLF

Project Name:

Healthcare Appointment & Prescription Tracking System

Submitted To: Moshiur Rahman

Submitted By		Score
Student Name	ID	
Md Sanaul Islam Zihad	2311774042	
Tanzidur Rahman	2232075642	

PROJECT SCOPE DESCRIPTION

Project Introduction

The *Healthcare Appointment and Prescription Management System* is designed to streamline clinical operations by centralizing patient information, appointment scheduling, medical history, diagnoses, prescriptions, and medicine management into a single integrated platform. This system ensures efficient communication between patients, doctors, and administrators while maintaining accuracy, security, and traceability of clinical data.

The system adopts a well-structured Entity–Relationship Design that incorporates advanced ERD concepts such as specialization/generalization (WebUser → Patient/Doctor/Admin), multivalued attributes (e.g., patient allergies), derived attributes (e.g., patient age), weak/dependent entities (appointments, visit summaries, prescription medicines), associative entities for many-to-many relationships, and supporting audit and access logs to ensure system accountability.

Project Objectives

1. Digitalize patient management by maintaining accurate patient records, allergies, medical history, and visit summaries.
2. Simplify doctor–patient interactions through a structured appointment system with available schedules, reminders, and status tracking.
3. Enable safe and standardized prescription generation using diagnoses reference tables and associated medicines.
4. Ensure system security and compliance through password reset mechanisms, access logs, and audit trails for every critical operation.
5. Provide a scalable and normalized data structure by applying ERD concepts such as
    - generalization (WebUser hierarchy),
    - dependent entities,
    - multivalued attribute conversion (Patient_Allergies),
    - derived attributes (age),
    - identifying and non-identifying relationships,
    - many-to-many resolution using junction entities.
6. Improve healthcare efficiency by allowing administrators to manage doctor schedules, medical resources, and user accounts effectively.
7. Maintain data accuracy and consistency through the use of foreign key constraints, reference tables Medicines

1.Entities and Relationships

In this project, we identified different entities, which are the main objects in the healthcare system. Each entity stores information about something important.

Entities Identified:

1. **Patient** – stores patient details 2.

Doctor – stores doctor details

3. **Admin** – stores admin account
4. **Schedule** – doctor’s available time slots
5. **Specialties** – doctor specialization (e.g., cardiology)
6. **Medicines** – medicine details
7. **Prescriptions** – prescription given to a patient
8. **Diagnoses** – list of possible diagnoses
9. **Appointment**
10. **Prescription_Medicines**
11. **Medical_History**
12. **Visit_Summaries**
13. **Record_Access_Log**
14. **Audit_Trail**

Main Relationships Identified:

1. Patients book appointments with doctors
2. Doctors create schedules
3. Appointments generate prescriptions and visit summaries
4. Prescriptions contain multiple medicines
5. Patients have medical history
6. System logs every access and update

2.Relationship Types (1:1, 1:N, M:N)

One-to-One (1:1):

Each appointment has one prescription and one visit summary.
So:

1. Appointment → Prescription (1:1)
2. Appointment → Visit Summary (1:1)

One-to-Many (1:N):

1. **Doctor → Schedule**
   One doctor can have many schedules.
2. **Schedule → Appointment**
   One schedule can have many appointments.
3. **Patient → Appointment**
   One patient can book many appointments.
4. **Patient → Medical History**
   One patient can have many medical history records.
5. **Patient → Prescriptions**
   One patient can receive many prescriptions.
6. **Doctor → Prescriptions**
   A doctor can issue many prescriptions.
7. **Specialties → Doctor**
   One specialty can have many doctors.
8. **Doctor → Patient (primary doctor)**
   One doctor can be primary doctor for many patients.

Many-to-Many (M:N):

- **Prescriptions ⇔ Medicines**
  A single prescription can include many medicines, and the same medicine can appear in many prescriptions.

This is implemented through the **Prescription_Medicines** table.

3.Generalization / Specialization

The system has different users—patients, doctors, and admins. To avoid repeating common fields, we use **generalization**.

Superclass:

1. **WebUser**
   1.1. email

   1.2. usertype (‘p’, ‘d’, ‘a’)

Subclasses:

1. **Patient**
2. **Doctor**
3. **Admin**

How it works:

WebUser stores **common information**.
The subclasses store **extra information** specific to each role.

This avoids duplication and makes the system easier to maintain.

4.Role Definitions

Doctor Roles:

1. **Treating Physician**
   In the Appointment entity, the doctor provides treatment.
2. **Prescriber**
   In the Prescription entity, the doctor issues the prescription.

Patient Roles:

1. **Appointment Holder**
   The patient is the one who books and attends the appointment.
2. **Prescription Recipient**

The patient receives medicines and instructions.

7. Weak Entities and Identifying Relationships

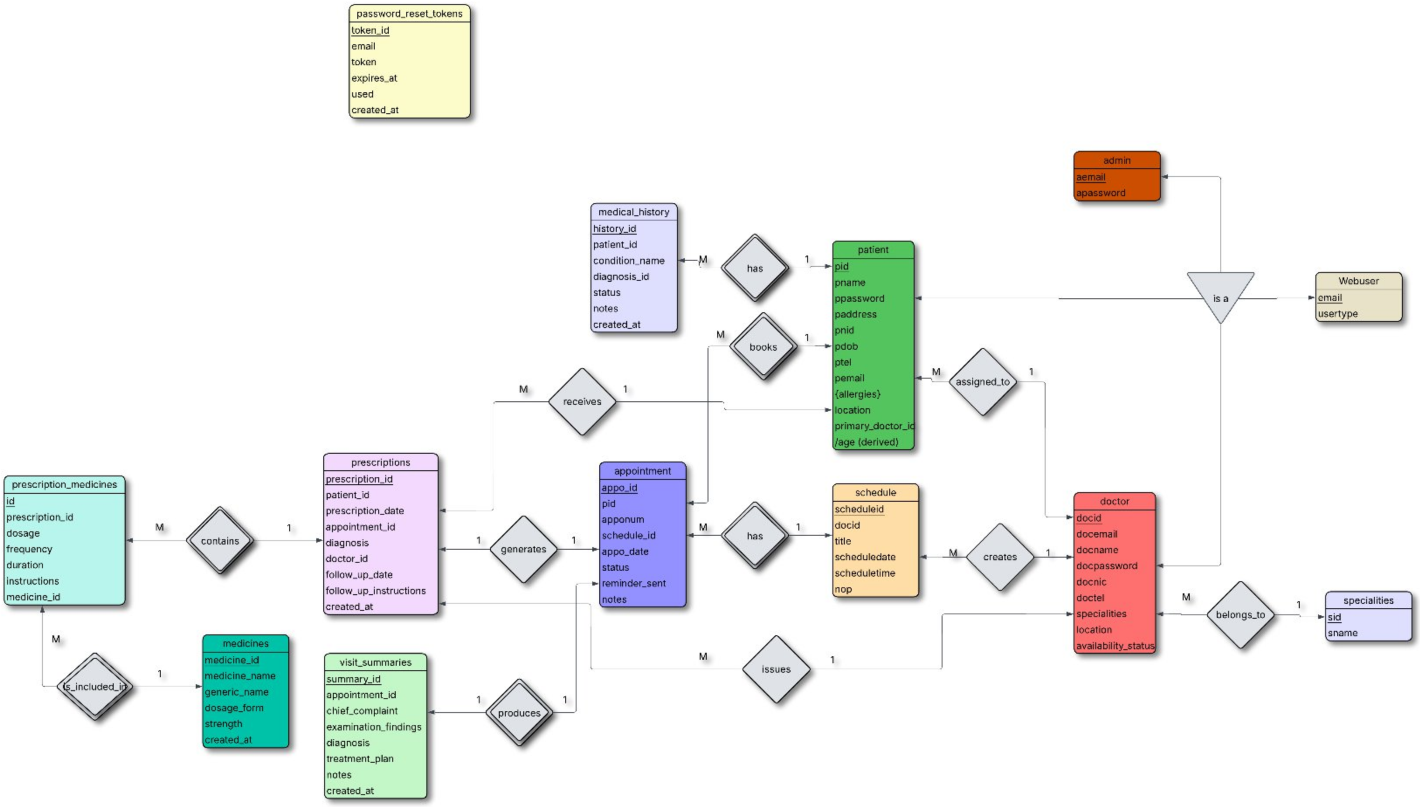
Weak Entities (dependent on others):

1. **Appointment** – depends on Patient + Schedule
2. **Prescription_Medicines** – depends on Prescription + Medicine
3. **Medical_History** – depends on Patient
4. **Visit_Summaries** – depends on Appointment
5. **Record_Access_Log** – created only when a record is accessed
6. **Audit_Trail** – created only when a record is modified

Identifying Relationships:

- Appointment is identified through the patient and the schedule it belongs to.
- Prescription_Medicines is identified by its related prescription and medicine.
- Medical_History entries are identified through the patient they belong to.
- Visit_Summaries are identified by the appointment they are linked with.
- Record_Access_Log and Audit_Trail entries depend on the specific record that was accessed or changed.

ERD


password_reset_tokens
token_id
email
token
expires_at
used
created_at
medical_history
history_id
patient_id
condition_name
diagnosis_id
status
notes
created_at
M
has
1
patient
pid
pname
ppassword
paddress
pnid
pdob
ptel
pemail
{allergies}
location
primary_doctor_i
/age (derived)
admin
aemail
apassword
is a
Webuser
email
usertype
M
books
1
M
assigned_to
1
M
receives
1
prescription_medicines
id
prescription_id
dosage
frequency
duration
instructions
medicine_id
M
contains
1
prescriptions
prescription_id
patient_id
prescription_date
appointment_id
diagnosis
doctor_id
follow_up_date
follow_up_instructions
created_at
1
generates
1
appointment
appo_id
pid
apponum
schedule_id
appo_date
status
reminder_sent
notes
M
has
1
schedule
scheduleid
docid
title
scheduledate
scheduletime
nop
M
creates
1
doctor
docid
docemail
docname
docpassword
docnic
doctel
specialities
location
availability_status
M
belongs_to
1
specialities
sid
sname
M
is_included_in
1
medicines
medicine_id
medicine_name
generic_name
dosage_form
strength
created_at
visit_summaries
summary_id
appointment_id
chief_complaint
examination_findings
diagnosis
treatment_plan
notes
created_at
1
produces
1
M
issues
1

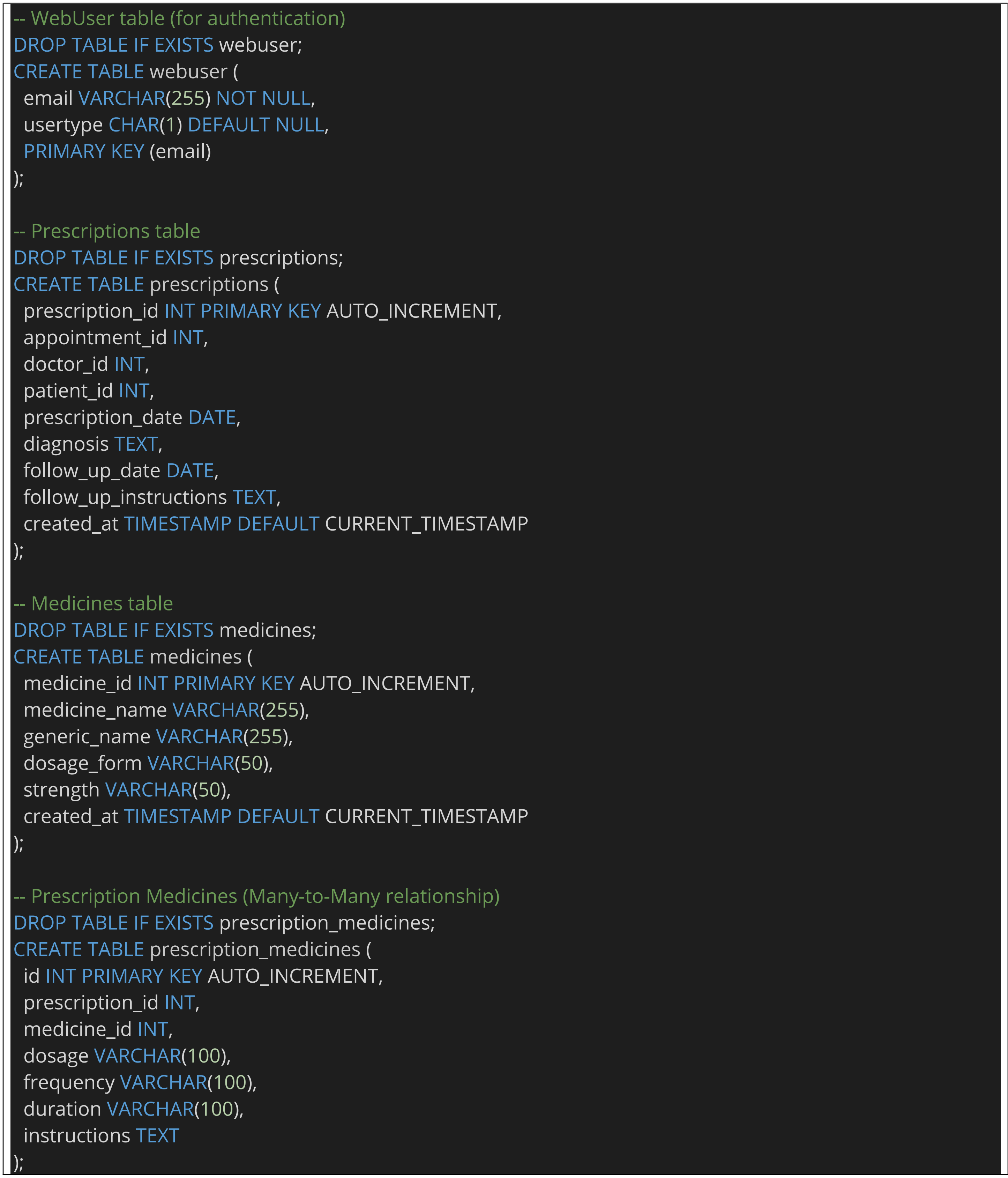
Relation Schema

Table Name	Attributes	Primary Key
WebUser	email, usertype, password	email
Patient	pid, pemail, pname, ppassword, paddress, pnic, pdob, ptel, location, primary_doctor_id	pid
Patient_Allergies	pid, allergy	pid
Doctor	docid, docemail, docname, docpassword, docnic, doctel, location, availability_status	docid
Specialties	sid, sname	sid
Doctor_Specialties	docid, sid	(docid, sid)
Schedule	scheduleid, docid, title, scheduledate, scheduletime, nop, duration_minutes	scheduleid
Appointment	appoid, pid, scheduleid, apponum, appodate, status, reminder_sent, notes	appoid
Prescriptions	prescription_id, patient_id, doctor_id, appointment_id, diagnosis_id, prescription_date, diagnosis_text, follow_up_date, follow_up_instructions, created_at	prescription_id
Prescription_Medicines	id, prescription_id, medicine_id, dosage, frequency, duration, instructions	id
Medicines	medicine_id, medicine_name, generic_name, dosage_form, strength, created_at	medicine_id
Medical_History	history_id, patient_id, diagnosis_id, condition_name, diagnosis_date, status, notes, created_at	history_id
Visit_Summaries	summary_id, appointment_id, chief_complaint, examination_findings, diagnosis_id, treatment_plan, notes, created_at	summary_id
Password_Reset_Tokens	token_id, email, token, expires_at, used, created_at	token_id

DDL

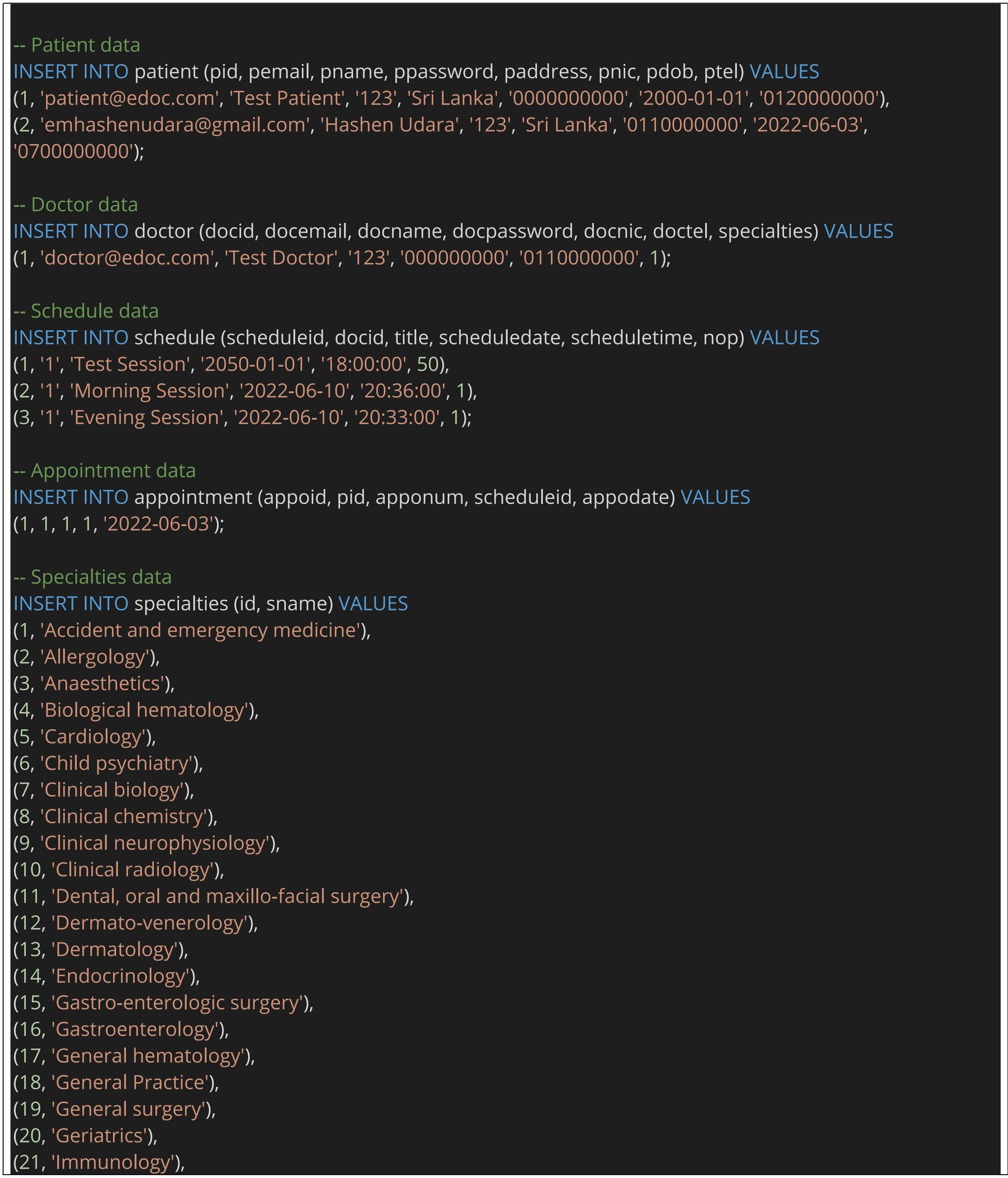
```
-- Healthcare Appointment & Prescription Tracking System
-- Create database if not exists
CREATE DATABASE IF NOT EXISTS edoc;
USE edoc;


-- Admin table
DROP TABLE IF EXISTS admin;
CREATE TABLE admin (
  aemail VARCHAR(255) NOT NULL,
  apassword VARCHAR(255) DEFAULT NULL,
  PRIMARY KEY (aemail)
);

-- Patient table
DROP TABLE IF EXISTS patient;
CREATE TABLE patient (
  pid INT(11) NOT NULL AUTO_INCREMENT,
  pemail VARCHAR(255) DEFAULT NULL,
  pname VARCHAR(255) DEFAULT NULL,
  ppassword VARCHAR(255) DEFAULT NULL,
  paddress VARCHAR(255) DEFAULT NULL,
  pnic VARCHAR(15) DEFAULT NULL,
  pdob DATE DEFAULT NULL,
  ptel VARCHAR(15) DEFAULT NULL,
  allergies TEXT,
  primary_doctor_id INT,
  location VARCHAR(255),
  PRIMARY KEY (pid)
);

-- Doctor table
DROP TABLE IF EXISTS doctor;
CREATE TABLE doctor (
  docid INT(11) NOT NULL AUTO_INCREMENT,
  docemail VARCHAR(255) DEFAULT NULL,
  docname VARCHAR(255) DEFAULT NULL,
  docpassword VARCHAR(255) DEFAULT NULL,
  docnic VARCHAR(15) DEFAULT NULL,
  doctel VARCHAR(15) DEFAULT NULL,
```

```sql
  specialties INT(2) DEFAULT NULL,
  location VARCHAR(255),
  availability_status VARCHAR(50) DEFAULT 'available',
  PRIMARY KEY (docid),
  KEY specialties (specialties)
);

-- Schedule table
DROP TABLE IF EXISTS schedule;
CREATE TABLE schedule (
  scheduleid INT(11) NOT NULL AUTO_INCREMENT,
  docid VARCHAR(255) DEFAULT NULL,
  title VARCHAR(255) DEFAULT NULL,
  scheduledate DATE DEFAULT NULL,
  scheduletime TIME DEFAULT NULL,
  nop INT(4) DEFAULT NULL,
  PRIMARY KEY (scheduleid),
  KEY docid (docid)
);

-- Appointment table
DROP TABLE IF EXISTS appointment;
CREATE TABLE appointment (
  appoid INT(11) NOT NULL AUTO_INCREMENT,
  pid INT(10) DEFAULT NULL,
  apponum INT(3) DEFAULT NULL,
  scheduleid INT(10) DEFAULT NULL,
  appodate DATE DEFAULT NULL,
  status VARCHAR(50) DEFAULT 'pending',
  reminder_sent BOOLEAN DEFAULT FALSE,
  notes TEXT,
  PRIMARY KEY (appoid),
  KEY pid (pid),
  KEY scheduleid (scheduleid)
);

-- Specialties table
DROP TABLE IF EXISTS specialties;
CREATE TABLE specialties (
  id INT(2) NOT NULL,
  sname VARCHAR(50) DEFAULT NULL,
  PRIMARY KEY (id)
);
```



```sql
-- WebUser table (for authentication)
DROP TABLE IF EXISTS webuser;
CREATE TABLE webuser (
  email VARCHAR(255) NOT NULL,
  usertype CHAR(1) DEFAULT NULL,
  PRIMARY KEY (email)
);

-- Prescriptions table
DROP TABLE IF EXISTS prescriptions;
CREATE TABLE prescriptions (
  prescription_id INT PRIMARY KEY AUTO_INCREMENT,
  appointment_id INT,
  doctor_id INT,
  patient_id INT,
  prescription_date DATE,
  diagnosis TEXT,
  follow_up_date DATE,
  follow_up_instructions TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Medicines table
DROP TABLE IF EXISTS medicines;
CREATE TABLE medicines (
  medicine_id INT PRIMARY KEY AUTO_INCREMENT,
  medicine_name VARCHAR(255),
  generic_name VARCHAR(255),
  dosage_form VARCHAR(50),
  strength VARCHAR(50),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Prescription Medicines (Many-to-Many relationship)
DROP TABLE IF EXISTS prescription_medicines;
CREATE TABLE prescription_medicines (
  id INT PRIMARY KEY AUTO_INCREMENT,
  prescription_id INT,
  medicine_id INT,
  dosage VARCHAR(100),
  frequency VARCHAR(100),
  duration VARCHAR(100),
  instructions TEXT
);
```

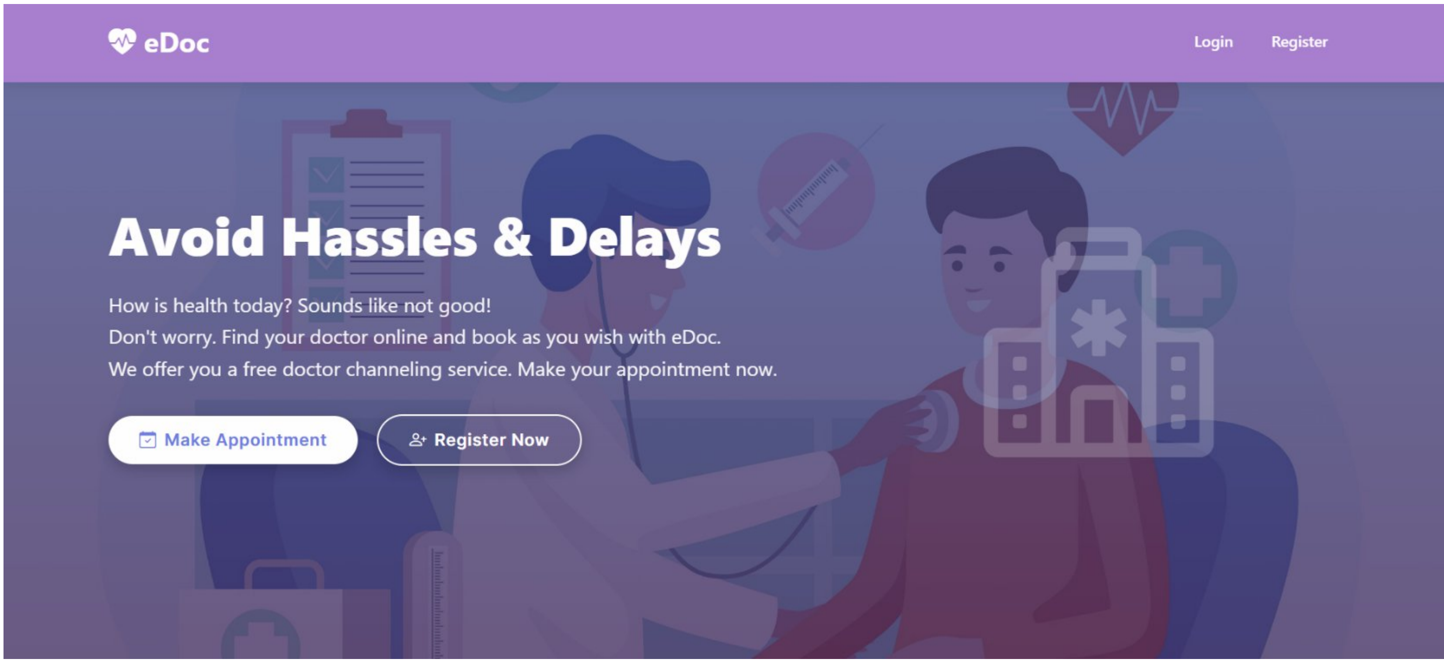
```
-- Medical History table
DROP TABLE IF EXISTS medical_history;
CREATE TABLE medical_history (
  history_id INT PRIMARY KEY AUTO_INCREMENT,
  patient_id INT,
  condition_name VARCHAR(255),
  diagnosis_date DATE,
  status VARCHAR(50),
  notes TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Visit Summaries table
DROP TABLE IF EXISTS visit_summaries;
CREATE TABLE visit_summaries (
  summary_id INT PRIMARY KEY AUTO_INCREMENT,
  appointment_id INT,
  chief_complaint TEXT,
  examination_findings TEXT,
  diagnosis TEXT,
  treatment_plan TEXT,
  notes TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (appointment_id) REFERENCES appointment(appoid) ON DELETE CASCADE
);

-- Add index for better performance
CREATE INDEX idx_appointment_id ON visit_summaries(appointment_id);

-- Password Reset Tokens table
DROP TABLE IF EXISTS password_reset_tokens;
CREATE TABLE password_reset_tokens (
  token_id INT PRIMARY KEY AUTO_INCREMENT,
  email VARCHAR(255),
  token VARCHAR(255),
  expires_at TIMESTAMP,
  used BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Admin data
INSERT INTO admin (aemail, apassword) VALUES
('admin@edoc.com', '123');
```



```
-- Patient data
INSERT INTO patient (pid, pemail, pname, ppassword, paddress, pnic, pdob, ptel) VALUES
(1, 'patient@edoc.com', 'Test Patient', '123', 'Sri Lanka', '0000000000', '2000-01-01', '0120000000'),
(2, 'emhashenudara@gmail.com', 'Hashen Udara', '123', 'Sri Lanka', '0110000000', '2022-06-03',
'0700000000');

-- Doctor data
INSERT INTO doctor (docid, docemail, docname, docpassword, docnic, doctel, specialties) VALUES
(1, 'doctor@edoc.com', 'Test Doctor', '123', '000000000', '0110000000', 1);

-- Schedule data
INSERT INTO schedule (scheduleid, docid, title, scheduledate, scheduletime, nop) VALUES
(1, '1', 'Test Session', '2050-01-01', '18:00:00', 50),
(2, '1', 'Morning Session', '2022-06-10', '20:36:00', 1),
(3, '1', 'Evening Session', '2022-06-10', '20:33:00', 1);

-- Appointment data
INSERT INTO appointment (appoid, pid, apponum, scheduleid, appodate) VALUES
(1, 1, 1, 1, '2022-06-03');

-- Specialties data
INSERT INTO specialties (id, sname) VALUES
(1, 'Accident and emergency medicine'),
(2, 'Allergology'),
(3, 'Anaesthetics'),
(4, 'Biological hematology'),
(5, 'Cardiology'),
(6, 'Child psychiatry'),
(7, 'Clinical biology'),
(8, 'Clinical chemistry'),
(9, 'Clinical neurophysiology'),
(10, 'Clinical radiology'),
(11, 'Dental, oral and maxillo-facial surgery'),
(12, 'Dermato-venerology'),
(13, 'Dermatology'),
(14, 'Endocrinology'),
(15, 'Gastro-enterologic surgery'),
(16, 'Gastroenterology'),
(17, 'General hematology'),
(18, 'General Practice'),
(19, 'General surgery'),
(20, 'Geriatrics'),
(21, 'Immunology'),
(22, 'Infectious diseases'),
(23, 'Internal medicine'),
(24, 'Laboratory medicine'),
(25, 'Maxillo-facial surgery'),
(26, 'Microbiology'),
(27, 'Nephrology'),
(28, 'Neuro-psychiatry'),
(29, 'Neurology'),
(30, 'Neurosurgery'),
(31, 'Nuclear medicine'),
(32, 'Obstetrics and gynecology'),
(33, 'Occupational medicine'),
(34, 'Ophthalmology'),
(35, 'Orthopaedics'),
(36, 'Otorhinolaryngology'),
(37, 'Paediatric surgery'),
(38, 'Paediatrics'),
(39, 'Pathology'),
(40, 'Pharmacology'),
(41, 'Physical medicine and rehabilitation'),
(42, 'Plastic surgery'),
(43, 'Podiatric Medicine'),
(44, 'Podiatric Surgery'),
(45, 'Psychiatry'),
(46, 'Public health and Preventive Medicine'),
(47, 'Radiology'),
(48, 'Radiotherapy'),
(49, 'Respiratory medicine'),
(50, 'Rheumatology'),
(51, 'Stomatology'),
(52, 'Thoracic surgery'),
(53, 'Tropical medicine'),
(54, 'Urology'),
(55, 'Vascular surgery'),
(56, 'Venereology');

-- WebUser data
INSERT INTO webuser (email, usertype) VALUES
('admin@edoc.com', 'a'),
('doctor@edoc.com', 'd'),
('patient@edoc.com', 'p'),
('emhashenudara@gmail.com', 'p');

-- Sample medicines
```

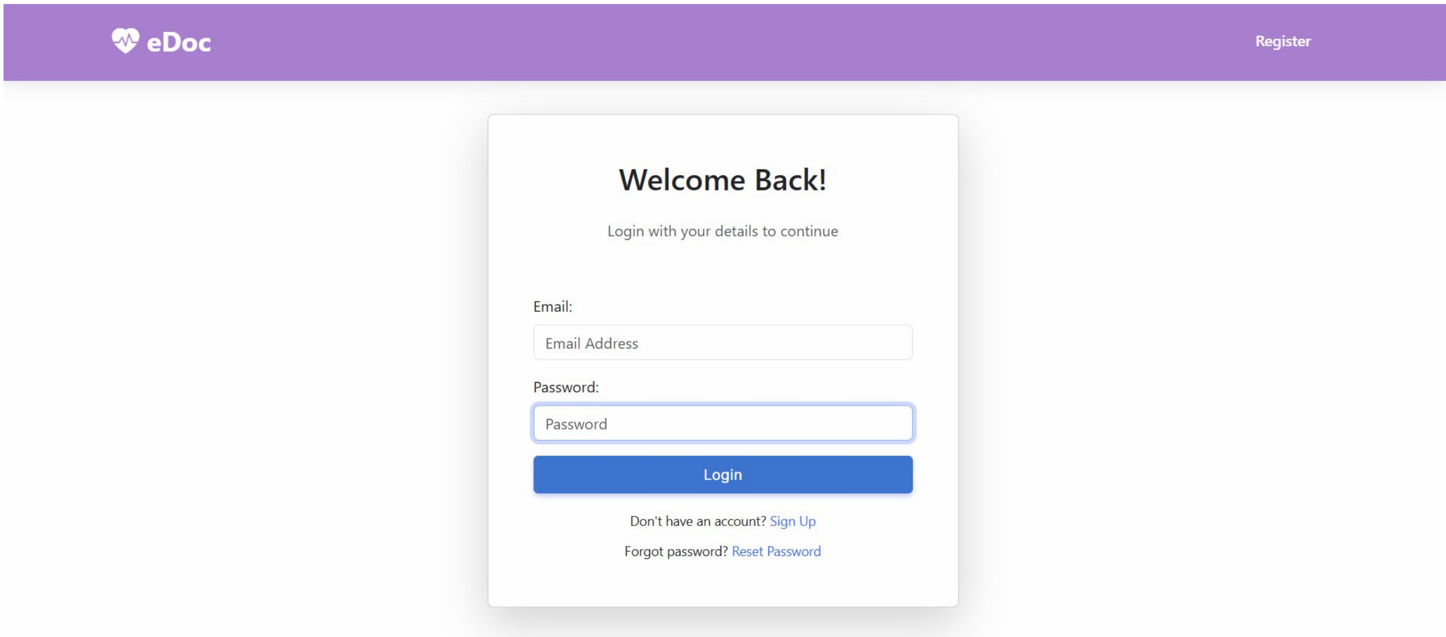
```
INSERT INTO medicines (medicine_name, generic_name, dosage_form, strength) VALUES
('Paracetamol', 'Acetaminophen', 'Tablet', '500mg'),
('Amoxicillin', 'Amoxicillin', 'Capsule', '250mg'),
('Ibuprofen', 'Ibuprofen', 'Tablet', '400mg'),
('Cetirizine', 'Cetirizine', 'Tablet', '10mg'),
('Omeprazole', 'Omeprazole', 'Capsule', '20mg');
```

Screenshots



Avoid Hassles & Delays

How is health today? Sounds like not good!
Don't worry. Find your doctor online and book as you wish with eDoc.
We offer you a free doctor channeling service. Make your appointment now.



Welcome Back!

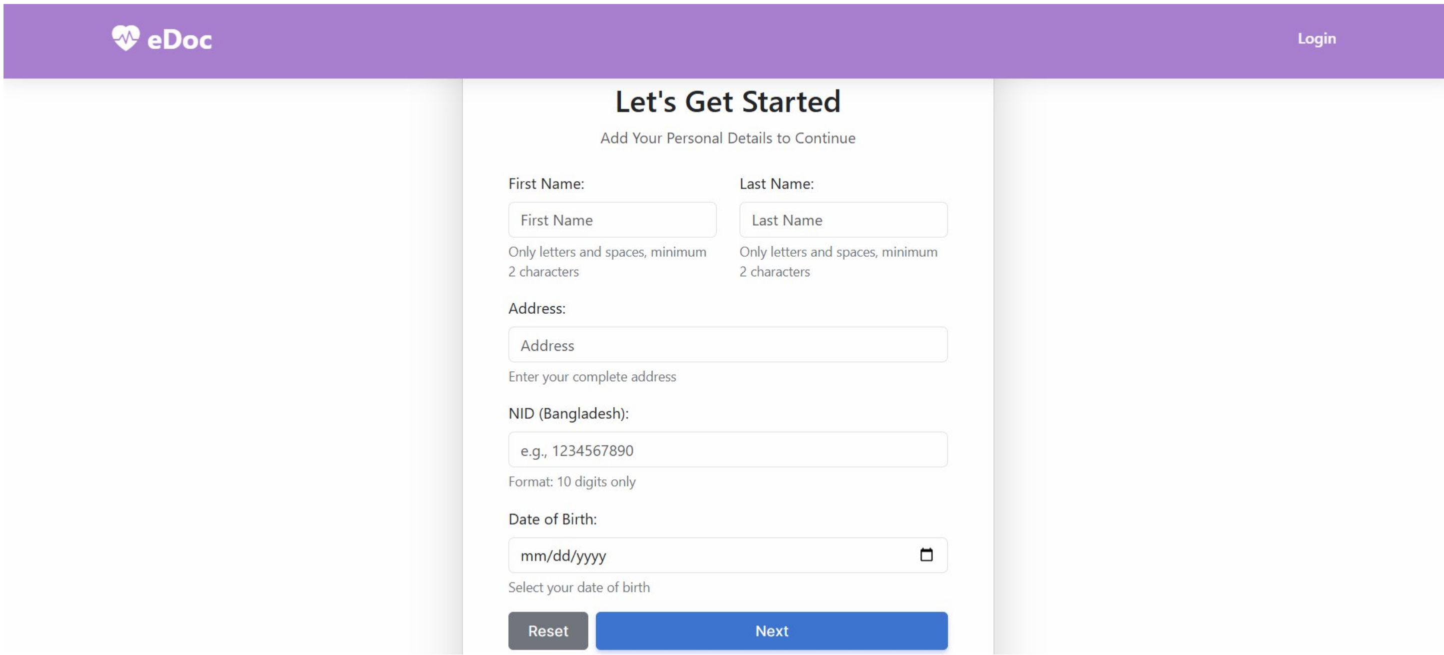
Login with your details to continue

Email: Email Address

Password: Password

Don't have an account? Sign Up

Forgot password? Reset Password



Let's Get Started

Add Your Personal Details to Continue

First Name: First Name

Only letters and spaces, minimum 2 characters

Last Name: Last Name

Only letters and spaces, minimum 2 characters

Address: Address

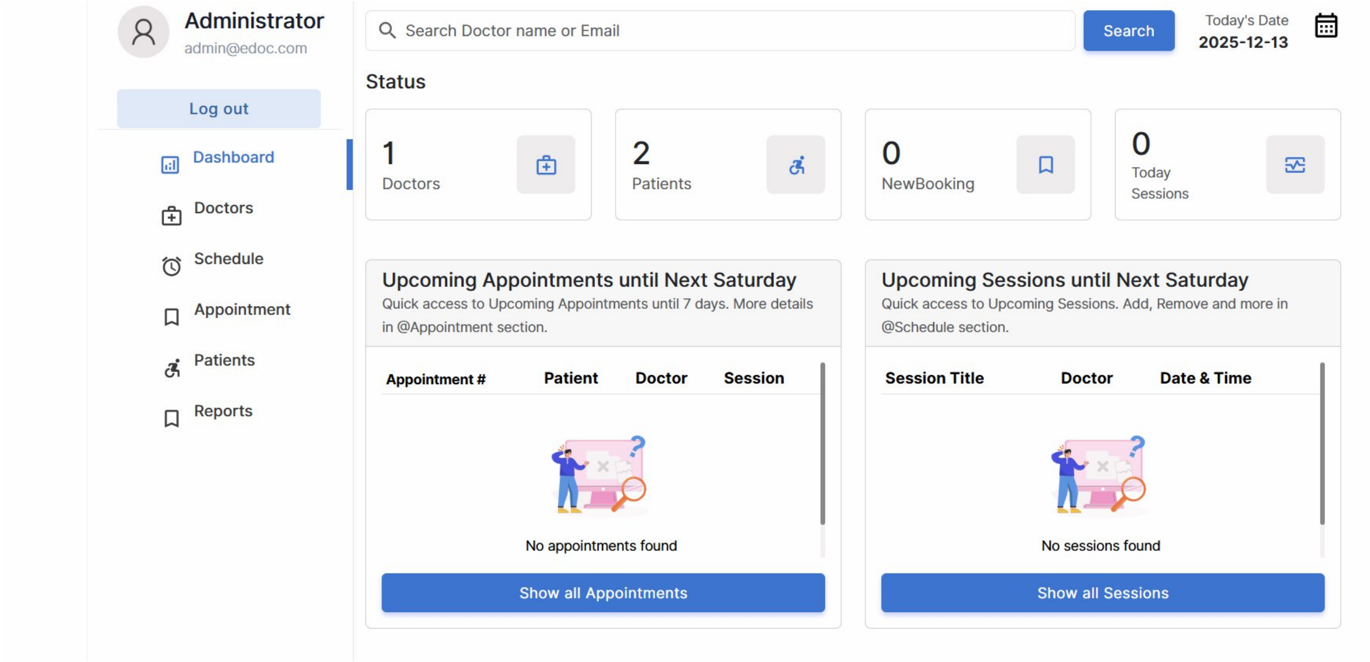
Enter your complete address

NID (Bangladesh): e.g., 1234567890

Format: 10 digits only

Date of Birth: mm/dd/yyyy

Select your date of birth



Status

Upcoming Appointments until Next Saturday

Quick access to Upcoming Appointments until 7 days. More details in @Appointment section.

Appointment #	Patient	Doctor	Session

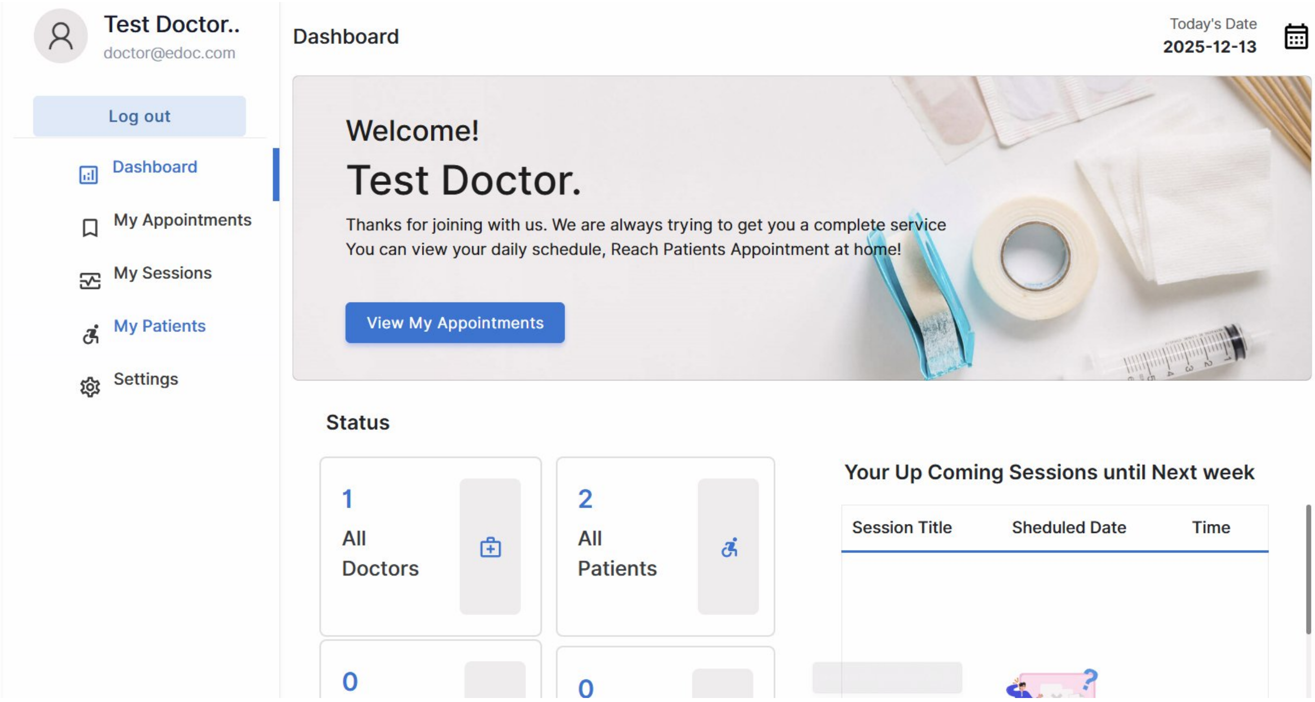
No appointments found

Upcoming Sessions until Next Saturday

Quick access to Upcoming Sessions. Add, Remove and more in @Schedule section.

Session Title	Doctor	Date & Time

No sessions found



Welcome!

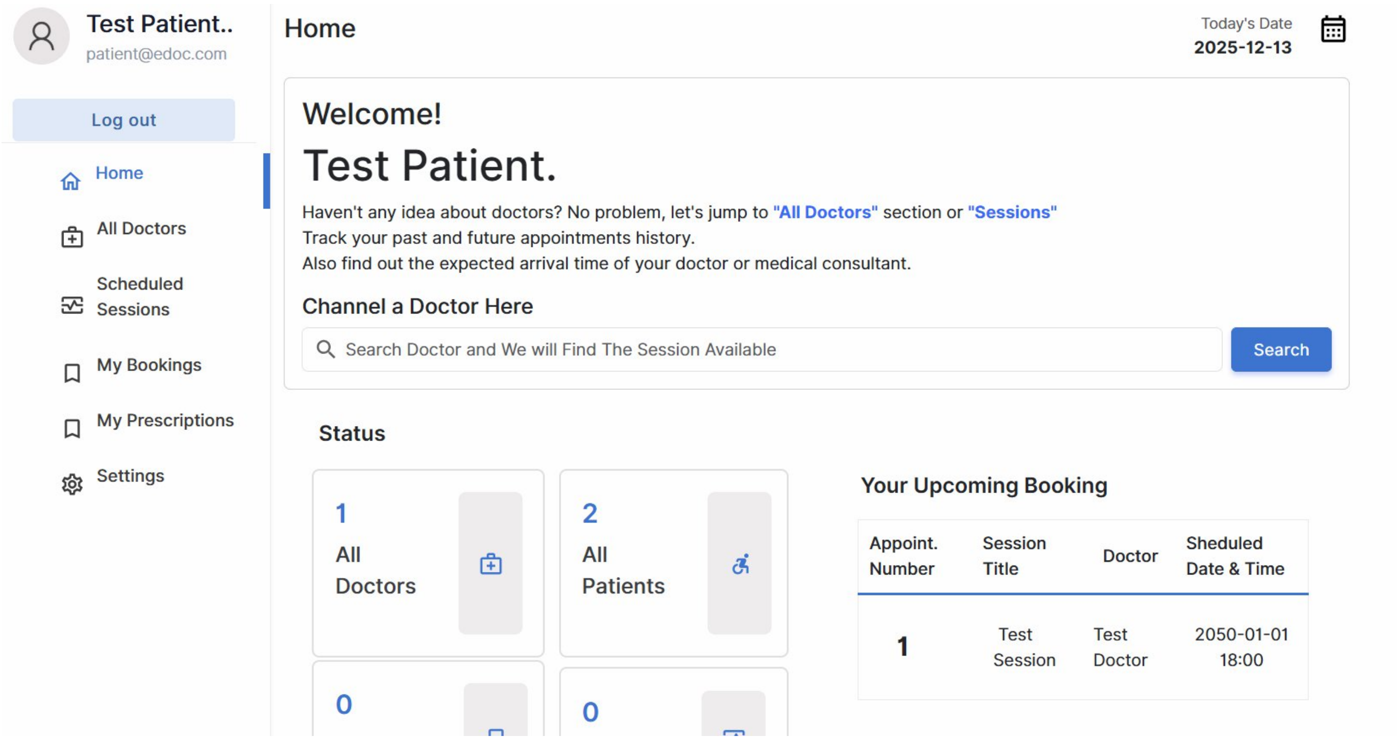
Test Doctor.

Thanks for joining with us. We are always trying to get you a complete service
You can view your daily schedule, Reach Patients Appointment at home!

Status

Your Up Coming Sessions until Next week

Session Title	Sheduled Date	Time



Welcome!

Test Patient.

Haven't any idea about doctors? No problem, let's jump to "All Doctors" section or "Sessions"
Track your past and future appointments history.
Also find out the expected arrival time of your doctor or medical consultant.

Channel a Doctor Here

Search Doctor and We will Find The Session Available | Search

Status

1 All Doctors

2 All Patients

0

0

Your Upcoming Booking

Appoint. Number	Session Title	Doctor	Sheduled Date & Time
1	Test Session	Test Doctor	2050-01-01 18:00

Contribution

Zihad

1. Patient Dashboard & Features

- Patient Registration
- Patient Login System
- Patient Dashboard
- Appointment Booking System
- Appointment Management
- View all appointments
- Cancel appointments
- Reschedule appointments
- Doctor Search
- Schedule Viewing
- Patient Settings

- Account Deletion

2. Authentication & Security

- Password Reset System
- Session Management

3. Database Tables

- `patient` table
- `webuser` table (authentication)
- `password_reset_tokens` table

Tanzidur

1. Doctor Dashboard & Features

- Doctor Dashboard
- Doctor Appointment Management
- Prescription System
- Create prescriptions
- Add multiple medicines
- Dosage, frequency, duration tracking
- Visit Summary
- Patient Records View
- Schedule Management
- Doctor Settings

2. Admin Dashboard & Management

- Admin Dashboard
- Doctor Management

- Patient Management
- Schedule Management
- Appointment Management
- Reports & Analytics - Statistics dashboard
- Doctor workload reports
- Prescription analytics
- Patient visit frequency

3. Advanced Features

- Medical History System
- Prescription Viewing for Patients
- Overlap Prevention in Booking
- Database Schema Design
- ERD Implementation

4. Database Tables

- `doctor` table
- `admin` table
- `schedule` table
- `appointment` table - `prescriptions` table
- `medicines` table
- `prescription_medicines` table
- `medical_history` table
- `visit_summaries` table

Conclusion

The Healthcare Appointment and Prescription Management System successfully demonstrates the application of database design principles, ER modeling, normalization, and practical implementation using PHP, MySQL, and XAMPP. Throughout the project, a complete end-to-end workflow was developed to manage users, appointments, schedules, prescriptions, medicines, diagnosis references, and patient medical history. The system integrates important database concepts such as weak entities, multivalued attributes, derived attributes, specialization/generalization, and many-to-many relationships, all of which were properly implemented in the final ERD and relational schema.

Key administrative and clinical functions including appointment booking, schedule management, prescription creation, medical history tracking, and audit logging were built to reflect real-world healthcare processes. The addition of audit trails, record-access logs, and password-reset mechanisms improved system security and accountability, making the database more robust and realistic. The implementation of CRUD operations further demonstrates practical understanding of database manipulation through a web interface.

Overall, this project provided valuable hands-on experience in database analysis, design, and implementation. It strengthened technical skills in SQL, relational modeling, and web-based system development while also highlighting the importance of accuracy, data integrity, and security in healthcare applications. The final outcome meets the functional requirements and serves as a strong foundation for future enhancements such as analytics, role-based access control, or full EMR (Electronic Medical Record) integration.