# Database Schema

## 1. Users Table

This table stores basic details about both patients and physicians. We can distinguish them using a role column.

| Column Name | Data Type | Description |
|---|---|---|
| user_id | INT (PK) | Unique identifier for each user. |
| name | VARCHAR(255) | Name of the user (either patient or physician). |
| email | VARCHAR(255) | Email address for user login. |
| password | VARCHAR(255) | Hashed password for login. |
| phone_number | VARCHAR(20) | Contact number of the user. |
| role | ENUM('PATIENT', 'PHYSICIAN') | Role of the user in the system. |
| address | TEXT | Address of the user (optional for physicians). |
| created_at | TIMESTAMP | Account creation timestamp. |

## 2. Patient Table

Stores additional information specific to patients.

| Column Name | Data Type | Description |
|---|---|---|
| patient_id | INT (PK) | Unique identifier for each patient (foreign key from Users table's user_id). |
| health_card | TEXT | Health card details (only for patients, can be NULL for physicians). |
| height | TEXT | Height of the patient. |
| weight | TEXT | Weight of the patient. |
| occupation | TEXT | Occupation of the patient. |
| drug_allergies | TEXT (Nullable) | Information about drug allergies. |
| pre_existing_conditions | TEXT (Nullable) | Information about pre-existing medical conditions. |

## 3. Physician Table

Contains details specific to physicians.

| Column Name | Data Type | Description |
|---|---|---|
| physician_id | INT (PK) | Unique identifier for each physician (foreign key from Users table's user_id). |
| specialization | TEXT (Nullable) | Specialization of the physician. |
| license | TEXT | License number of the physician. |
| accepting_patients | BOOLEAN | Indicates if the physician is accepting new patients. |

|  |  |  |
|---|---|---|
| clinic_id | INT (FK) | Foreign key reference to the Clinic table. |

## 4. Appointments Table

Stores appointment details for scheduling, modification, and cancellation.

| Column Name | Data Type | Description |
|---|---|---|
| appointment_id | INT (PK) | Unique ID for the appointment. |
| patient_id | INT (FK) | ID of the patient from the Users table. |
| physician_id | INT (FK) | ID of the physician from the Users table. |
| appointment_time | DATETIME | Scheduled appointment time. |
| status | ENUM('BOOKED', 'CANCELED', 'COMPLETED', 'ONGOING') | Current status of the appointment. |
| rescheduled_time | DATETIME | If rescheduled, the new appointment time (NULL if not rescheduled). |
| booking_time | TIMESTAMP | The time the appointment was booked. |
| appointment_end_time | DATETIME | End time of the appointment. |

## 5. Clinic Table

Stores clinic-related information.

| Column Name | Data Type | Description |
|---|---|---|
| clinic_id | INT (PK) | Unique ID for each clinic. |
| name | VARCHAR(255) | Name of the clinic. |
| address | TEXT | Address of the clinic. |
| contact | VARCHAR(20) | Phone number of the clinic. |
| email | VARCHAR(255) | Email address of the clinic. |

## 6. AppointmentNote Table

Stores symptoms and diagnosis information for each appointment.

| Column Name | Data Type | Description |
|---|---|---|
| appt_dtl_id | INT (PK) | Unique identifier for appointment notes. |
| appointment_id | INT (FK) | Reference to the appointment where diagnosis are noted. |
| symptoms | TEXT | Description of symptoms entered by the patient or physician. |
| diagnosis | TEXT | Description of diagnosis entered by the patient or physician. |

## 7. Prescription Table

Records prescription details associated with an appointment.

| Column Name | Data Type | Description |
|---|---|---|
| prescription_id | INT (PK) | Unique ID for each prescription entry. |
| appointment_id | INT (FK) | Reference to the appointment where prescription is noted. |
| prescription | TEXT | Description of prescription entered by the patient or physician. |
| created_at | TIMESTAMP | Timestamp when the prescription was entered. |

## 8. Payments Table

Handles payment information, including method and status.

| Column Name | Data Type | Description |
|---|---|---|
| payment_id | INT (PK) | Unique ID for the payment. |
| appointment_id | INT (FK) | Reference to the appointment for which payment is made. |
| amount | DECIMAL(10, 2) | Amount paid for the appointment. |
| payment_method | ENUM('CARD', 'ONLINE') | Method of payment (e.g., card, online). |
| payment_status | ENUM('PENDING', 'COMPLETED', 'FAILED') | Current status of the payment. |

|  |  |  |
|---|---|---|
| transaction_id | VARCHAR(255) | Payment gateway transaction ID. |
| created_at | TIMESTAMP | Payment timestamp. |

## Security:

1. We will use data encryption to encrypt sensitive fields like password, health_card, and drug_allergies before storing it in the table and decrypt it only after getting out of database in the application layer. It's best to use the AES_ENCRYPT() function to store encrypted data and AES_DECRYPT() to retrieve it.
2. We will implement role based access controls for using various resources of our application. Use MySQL user roles to limit access to tables based on user roles. For example:
   - Patients should have access only to their own records and appointment details.
   - Physicians should have access to their patients' records and appointment information.
3. In the application code, use prepared statements to prevent SQL injection.

Future Scope for Security:

1. Implement RLS and CLS - For RLS, we will tag each of the rows in all of the tables to their corresponding authorized roles so that when someone tries to access certain row, the access should only be granted if the accessing user roles matches one of the authorized roles for that particular row. Ex. If [patient_role, physician_role] are the tags associated with a particular row, then both patient and physician can access that row (roles can be different for view, update and delete).