**My sql part codes**

CREATE TABLE hospital (

hospital\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL DEFAULT 'Pratapgarh\_jila\_hospital',

address VARCHAR(255) NOT NULL DEFAULT 'Pratapgarh ranapratap choraha',

phone VARCHAR(15) NOT NULL DEFAULT '9628453433',

email VARCHAR(100) NOT NULL DEFAULT 'pbh@gmail.com'

);

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

hospital\_id INT NOT NULL,

username VARCHAR(50) UNIQUE NOT NULL,

password VARCHAR(50) NOT NULL,

user\_type ENUM('patient', 'doctor') NOT NULL,

FOREIGN KEY (hospital\_id) REFERENCES hospital(hospital\_id)

);

CREATE TABLE patients (

patient\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

hospital\_id INT NOT NULL,

name VARCHAR(100),

address VARCHAR(255),

phone VARCHAR(15),

date\_of\_birth DATE,

email VARCHAR(100),

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (hospital\_id) REFERENCES hospital(hospital\_id)

);

CREATE TABLE doctors (

doctor\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

hospital\_id INT NOT NULL,

name VARCHAR(100),

specialization VARCHAR(100),

phone VARCHAR(15),

email VARCHAR(100),

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (hospital\_id) REFERENCES hospital(hospital\_id)

);

CREATE TABLE tokens (

token\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT NOT NULL,

doctor\_id INT NOT NULL,

hospital\_id INT NOT NULL,

issue\_date DATETIME NOT NULL,

status ENUM('pending', 'completed') DEFAULT 'pending',

visit\_time DATETIME,

FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id) ON DELETE CASCADE,

FOREIGN KEY (doctor\_id) REFERENCES doctors(doctor\_id) ON DELETE CASCADE,

FOREIGN KEY (hospital\_id) REFERENCES hospital(hospital\_id) ON DELETE CASCADE

);

CREATE TABLE appointments (

appointment\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT NOT NULL,

doctor\_id INT NOT NULL,

hospital\_id INT NOT NULL,

appointment\_date DATETIME NOT NULL,

FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id),

FOREIGN KEY (doctor\_id) REFERENCES doctors(doctor\_id),

FOREIGN KEY (hospital\_id) REFERENCES hospital(hospital\_id)

);

CREATE TABLE medical\_records (

record\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT NOT NULL,

doctor\_id INT NOT NULL,

hospital\_id INT NOT NULL,

diagnosis TEXT NOT NULL,

treatment TEXT NOT NULL,

record\_date DATE NOT NULL,

FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id) ON DELETE CASCADE,

FOREIGN KEY (doctor\_id) REFERENCES doctors(doctor\_id) ON DELETE CASCADE,

FOREIGN KEY (hospital\_id) REFERENCES hospital(hospital\_id) ON DELETE CASCADE

);

-- Create Billing Table

CREATE TABLE billing (

bill\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT NOT NULL,

hospital\_id INT NOT NULL, -- Add hospital\_id column

amount DECIMAL(10, 2) NOT NULL,

payment\_status ENUM('paid', 'unpaid') DEFAULT 'unpaid',

billing\_date DATETIME NOT NULL,

FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id) ON DELETE CASCADE,

FOREIGN KEY (hospital\_id) REFERENCES hospital(hospital\_id) ON DELETE CASCADE -- Add foreign key constraint

);

**"summary" or "description"**

**Hospital Management System in Python**

**Description**

* This project is a comprehensive Hospital Management System developed in Python with a MySQL database backend. It is designed to streamline various aspects of hospital operations, including user management, appointment scheduling, token issuance, medical records maintenance, and billing processes. The system ensures secure and efficient handling of patient and doctor information, facilitating smooth interaction and record-keeping.

**Features**

**User Registration:**

* Patients and doctors can create accounts by providing their personal details.
* Unique usernames and secure passwords are enforced.

**Login and Authentication:**

* Secure login system to authenticate patients and doctors.
* Role-based access control to differentiate between patient and doctor functionalities.

**Appointment Booking:**

* Patients can book appointments with their preferred doctors.
* Doctors can view and manage their appointment schedules.

**Token Management:**

* Doctors can issue tokens for patient visits, indicating the order of consultations.
* The status of tokens (pending/completed) can be updated by doctors.

**Display Appointments:**

* Patients can view their upcoming and past appointments.
* Doctors can see their scheduled appointments for better planning.

**Display Tokens:**

* Patients can check the status of their issued tokens.
* Doctors can monitor the pending tokens and manage patient flow.

**Medical Records:**

* Comprehensive management of patient medical records, including diagnosis and treatment history.
* Doctors can add and update medical records for their patients.

**Billing:**

* Billing module to manage patient bills, including amount and payment status.
* Patients can view their billing history and payment statuses.

**Technologies Used**

* **Programming Language:** Python
* **Database:** MySQL
* **Libraries:**
  + mysql-connector-python for database connectivity
  + Other standard Python libraries for date and time handling
* **SQL Table Structure**

The system is built on a robust relational database structure with the following key tables:

**Hospital Table:**

* Stores basic information about hospitals.
* Ensures there is at least one default hospital entry.

**Users Table:**

* Manages user information for both patients and doctors.
* Links users to a specific hospital.

**Patients Table:**

* Contains detailed information about patients.
* Linked to the Users table for authentication and hospital association.

**Doctors Table:**

* Stores details about doctors, including specialization.
* Linked to the Users table for authentication and hospital association**.**

**Tokens Table:**

* Manages tokens issued by doctors to patients for consultations.
* Includes status tracking and visit times.

**Appointments Table:**

* Records appointments booked by patients with doctors.
* Tracks appointment dates and associations with hospitals.

**Medical Records Table:**

* Maintains detailed medical records for patients.
* Includes diagnosis, treatment, and record dates.

**Billing Table:**

* Handles billing information for patients.
* Tracks payment status and billing dates.

**Default Hospital Entry**

The application includes a default hospital entry to ensure there is always at least one hospital in the system:

* **Hospital ID:** 1
* **Name:** Pratapgarh\_jila\_hospital
* **Address:** Pratapgarh ranapratap choraha
* **Phone:** 9628453433
* **Email:** pbh@gmail.com