****

**Dhaka International University**

Department of CSE

Batch: E-87, Semester: 12th

Course Code: CSE- 425, Course Name: Project Work

**Project Title: Paperless Patient Management System.**

**Short description of project Work:**

|  |
| --- |
| The "**Paperless Patient Management System**" is a Python and Django-based web application using MySQL Database with modules for Patients, Doctors, Pathology, and Pharmacy. Each module allows online registration with unique credentials. Patients can view their profile, doctor and appointment lists, and test/medicine history. Doctors can manage their profile, patient list, appointments, and patient history. Pathologists and pharmacists access patient history via token ID to track tests and medications prescribed by doctors. |

**Project Features:**

* **Programming:** Python with Django Web Framework (Backend)
* **Database:** MYSQL
* **Front-end:** HTML, CSS, JS, Bootstrap and Ajax
* **Modules:** Patient, Doctor, Pathology and Pharmacy.

**1. Project Overview**

The Paperless Patient Management System is a web-based application developed using Python's Django framework with MySQL as the database backend and HTML, CSS, and JavaScript for the frontend. This system facilitates streamlined management of patient data and interactions with healthcare professionals. It comprises four primary modules:

* **Patient Module**
* **Doctor Module**
* **Pathology Module**
* **Pharmacy Module**

Each module allows users to register online and access specific features and information tailored to their role in the system.

**2. Technologies Used**

* **Backend Framework**: Django (Python)
* **Database**: MySQL
* **Frontend**: HTML, CSS, JavaScript
* **Additional Libraries**: Django Rest Framework (Optional for API)

**3. System Features**

**General Features**

* Online registration for each user role with separate login credentials.
* Role-based access control for data visibility and manipulation.
* Token-based patient history access for Pathology and Pharmacy.

**4. Module Descriptions**

**4.1. Patient Module**

**Functionality**:

* **Registration/Login**: Patients can register themselves and log in using their credentials.
* **Profile Management**: Patients can update personal details.
* **Doctor List**: View a list of available doctors.
* **Appointment List**: View their upcoming and past appointments.
* **Test and Medicine History**: Access a detailed record of tests taken and medicines prescribed.

**Key Features**:

* Easy appointment scheduling.
* Comprehensive view of medical history.

**Database Tables**:

* Patient: Stores patient details like name, address, contact information, etc.
* Appointment: Stores appointment details between patients and doctors.
* TestHistory: Contains records of tests taken by patients.
* MedicineHistory: Stores information on medicines prescribed to patients.

**4.2. Doctor Module**

**Functionality**:

* **Registration/Login**: Doctors can create accounts and log in with their credentials.
* **Profile Management**: Doctors can manage and update their profiles.
* **Patient List**: Access a list of patients who have booked appointments.
* **Appointment Management**: View and manage both upcoming and past appointments.
* **Patient History**: View medical test and medicine history of patients.

**Key Features**:

* Ability to view patients' complete medical history.
* Organized appointment scheduling system.

**Database Tables**:

* Doctor: Stores doctor information such as specialization, availability, and contact details.
* Appointment: Links patients with doctors for specific dates and times.
* TestHistory and MedicineHistory: Allows doctors to view patient records.

**4.3. Pathology Module**

**Functionality**:

* **Login**: Pathologists log in using their credentials.
* **Patient History Access**: Pathologists can access patient history using the Token ID.
* **Test Records**: Pathologists can see a complete list of tests taken by the patient.

**Key Features**:

* Ability to search for patients using a Token ID.
* View detailed patient test records.

**Database Tables**:

* TestHistory: Contains information on all tests associated with patients.

**4.4. Pharmacy Module**

**Functionality**:

* **Login**: Pharmacists can log in with their credentials.
* **Patient History Access**: Pharmacists access patient history using the Token ID.
* **Medicine Records**: Pharmacists can see which medicines have been prescribed by doctors for a particular patient.

**Key Features**:

* Token-based access for viewing patient medicine history.
* Easy identification of prescribed medicines for patients.

**Database Tables**:

* MedicineHistory: Stores details on which medicines were prescribed by doctors to patients.

**5. System Architecture**

**5.1. Frontend Design**

* **HTML/CSS/JavaScript**: The frontend is responsible for the user interface, allowing each user role to navigate the system seamlessly. Features such as form validation, dropdowns, and data visualization enhance the user experience.

**5.2. Backend Design**

* **Django Framework**: Handles the logic of the system, including routing requests, managing sessions, and ensuring secure authentication for each user type.
* **MySQL Database**: Stores all records securely, including patient, doctor, test, and medicine data.

**5.3. API Integration (Optional)**

* For API interaction between the modules or external services, Django REST Framework can be integrated to provide a RESTful API interface.

**6. User Guide**

**6.1. Patient Registration and Login**

* Step-by-step guide for patient registration, including required fields.
* Instructions for logging in and accessing the profile, doctor list, and appointment history.

**6.2. Doctor Registration and Login**

* Instructions for doctors on how to register and log in.
* Guide to managing their patient list and appointments.

**6.3. Pathologist Access**

* Instructions on logging in and using the Token ID to access patient test history.

**6.4. Pharmacist Access**

* Instructions on logging in and using the Token ID to access patient medicine history.

**7. Security Considerations**

* **Role-based Access Control**: Each module has restricted access to specific data, ensuring data privacy and compliance with healthcare regulations.
* **Encryption**: Sensitive data such as passwords and personal information should be encrypted.
* **Session Management**: Ensure sessions are handled securely to prevent unauthorized access.

**8. Database Schema**

**8.1. Main Tables:**

* Patient(id, name, address, contact\_info, profile\_info)
* Doctor(id, name, specialization, contact\_info, profile\_info)
* Appointment(id, patient\_id, doctor\_id, date, time)
* TestHistory(id, patient\_id, test\_name, result, date)
* MedicineHistory(id, patient\_id, medicine\_name, dosage, prescribed\_by, date)

**8.2. Relationships:**

* Patients and doctors are linked via the Appointment table.
* TestHistory and MedicineHistory are linked to patients by patient\_id.

**9. Deployment**

**9.1. Server Requirements**

* **Python**: Django framework.
* **MySQL**: Database server.
* **Web Server**: Nginx or Apache for serving the application.
* **Optional**: Cloud deployment on AWS, Google Cloud, or other platforms.

**9.2. Steps for Deployment**

* Set up the Django environment on the server.
* Configure MySQL database and connect it to Django.
* Migrate the database using Django's migration system.
* Serve the application using a web server.

**10. Future Enhancements**

* **Mobile App Integration**: Develop a mobile app for better accessibility.
* **Notification System**: Integrate email or SMS notifications for appointments and test results.
* **AI Integration**: Leverage AI for predictive analysis in patient treatment.

This documentation provides an organized structure for the project, making it easier for other developers and users to understand and interact with the system. You can modify or expand on any section based on specific implementation details.