**Doctor-Patient Appointment System - Internship Project Documentation**

**1. Introduction**

This is a **mini-project** for internship students to build a **Doctor-Patient Appointment System** using **Flask, SQLite, HTML, CSS, and Flask-Login**. The system allows **patients** to book appointments with **doctors**, while doctors can manage their schedules.

**2. Project Objectives**

* Patients can **register**, **login**, and **book appointments** with doctors.
* Doctors can **view their appointments** but cannot book new ones.
* Secure **user authentication** using Flask-Login and password hashing with Flask-Bcrypt.
* CRUD (Create, Read, Update, Delete) functionality for **managing appointments**.
* Responsive UI using **HTML, CSS (static files), and Bootstrap**.

**3. Technologies Used**

|  |  |
| --- | --- |
| **Technology** | **Purpose** |
| Flask | Web framework (Python) |
| Flask-SQLAlchemy | ORM for database handling |
| SQLite | Lightweight database |
| Flask-Login | User authentication |
| Flask-Bcrypt | Password hashing for security |
| HTML, CSS | Frontend UI |
| Bootstrap | Responsive design |
| Jinja2 | Templating engine for Flask |

**4. System Requirements**

* **Python 3.8+**
* Flask and required dependencies (install via requirements.txt)
* SQLite (included with Python)

**5. How to Set Up the Project**

**Step 1: Install Dependencies**

pip install -r requirements.txt

**Step 2: Initialize Database**

Run the following command inside Python:

from app import db

db.create\_all()

**Step 3: Run the Flask Application**

python app.py

The app will run on **http://127.0.0.1:5000/**.

**6. Database Structure**

**User Table**

|  |  |  |
| --- | --- | --- |
| Column | Type | Description |
| id | Integer (Primary Key) | Unique user ID |
| username | String | Unique username |
| password | String | Hashed password |
| role | String | 'doctor' or 'patient' |

**Appointment Table**

|  |  |  |
| --- | --- | --- |
| Column | Type | Description |
| id | Integer (Primary Key) | Unique appointment ID |
| patient\_name | String | Name of patient |
| doctor\_name | String | Name of doctor |
| date | String | Appointment date |
| time | String | Appointment time |
| user\_id | Integer (Foreign Key) | ID of the patient |

**7. Features Implemented**

✅ **User Authentication**

* Secure **user registration** and **login/logout** using Flask-Login.
* Passwords are **hashed** using Flask-Bcrypt.

✅ **Appointment Booking**

* **Patients** can **book appointments** by selecting a doctor, date, and time.
* **Doctors** can **view their appointments**.

✅ **Appointment Management**

* Patients can **edit** or **delete** their own appointments.
* Doctors can **view** but **not modify** appointments.

✅ **Role-Based Access Control**

* Patients **see a "Book Appointment" button**.
* Doctors **only see their scheduled appointments**.

**8. Folder Structure**

flask-appointment-system/

│── app.py # Main Flask app

│── database.db # SQLite database

│── requirements.txt # Dependencies

│── README.md # Documentation

│

├── templates/ # HTML templates

│ ├── index.html

│ ├── login.html

│ ├── register.html

│ ├── appointment.html

│ ├── view\_appointments.html

│

├── static/ # Static files (CSS, JS, images)

│ ├── css/styles.css

│

└── \_\_pycache\_\_/ # Compiled Python files

**9. Future Improvements**

💡 **Additional Features to Implement:**

* ✅ **Appointment Status:** Doctors can mark appointments as **"Completed" or "Pending".**
* ✅ **Add Doctor and Patient’s name:** Add and store Doctor and Patient name in appointments.
* ✅ **Enhanced UI:** Improve **mobile responsiveness** with Bootstrap.
* ✅ **Email Notifications (Optional):** Send **email reminders** for appointments.

**10. Internship Guidelines**

📌 **Tasks for Students:**

1. **Set up the project** on your local machine.
2. **Understand the database structure** (User and Appointment models).
3. Implement **role-based access** for patients and doctors.
4. **Modify the UI** to make it more user-friendly.
5. **Extend the features** by adding appointment status, notifications, or reports.
6. **Submit your final code** and document your work.