Hospital Management System

Objective

Develop a simplified Hospital Management System using Django. This system will allow authenticated users to manage doctors, patients, and medical records. The application will focus on creating, updating, deleting, and retrieving these entities. This limited scope is designed to be completed within two days.

Additional reading

1. https://docs.djangoproject.com/en/5.1/topics/auth/#user-authentication-in-django
2. https://docs.djangoproject.com/en/5.1/topics/auth/default/
3. https://docs.djangoproject.com/en/5.1/topics/auth/default/#module-django.contrib.auth.views

Business Logic and Requirements:

1. User Roles:
    a. Admin: Manages doctors, patients, and medical records. Admins have full control over the system.
    b. Doctor: Can view and manage their own medical records and patients.
2. Entities and Models:
    a. Patient:
        i. name: Full name of the patient.
        ii. email: Contact email of the patient (unique).
        iii. phone_number: Contact number of the patient.
        iv. date_of_birth: Date of birth of the patient.
        v. gender: Gender of the patient.
        vi. created_at: Timestamp for when the patient was registered.
        vii. updated_at: Timestamp for the last update to the patient record.
    b. Doctor:

i. name: Full name of the doctor.
ii. email: Contact email of the doctor (unique).
iii. phone_number: Contact number of the doctor.
iv. specialization: Field of specialization (e.g., Cardiology, Dermatology).
v. created_at: Timestamp for when the doctor was registered.
vi. updated_at: Timestamp for the last update to the doctor record.

c. Appointment:
i. doctor: ForeignKey to the Doctor model.
ii. patient: ForeignKey to the Patient model
iii. scheduled_at: timestamp for when the appointment is scheduled
iv. created_at
v. updated_at

d. Medical Record:
i. patient: ForeignKey to the Patient model.
ii. doctor: ForeignKey to the Doctor model.
iii. visit_date: Date of the visit.
iv. diagnosis: Diagnosis given by the doctor.
v. treatment: Treatment prescribed to the patient.
vi. notes: Additional notes from the doctor.
vii. report: Optional field to store any report.
viii. created_at: Timestamp for when the medical record was created.
ix. updated_at: Timestamp for the last update to the medical record.

3. Views:
a. Patient Management Views:
i. Patient List View: Displays a list of all patients. Admins can view, search, and filter patients.
ii. Patient Detail View: Displays detailed information about a specific patient.

iii. Create/Update Patient View: Allows creation or modification of a patient's information.
iv. Delete Patient View: Allows deletion of a patient's record (admin only).

b. Doctor Management Views:
    i. Doctor List View: Displays a list of all doctors. Admins can view, search, and filter doctors by specialization.
    ii. Doctor Detail View: Displays detailed information about a specific doctor.
    iii. Create/Update Doctor View: Allows creation or modification of a doctor's information.
    iv. Delete Doctor View: Allows deletion of a doctor's record (admin only).
c. Medical Record Management Views:
    i. Medical Record List View: Displays a list of all medical records associated with a specific patient.
    ii. Medical Record Detail View: Displays detailed information about a specific medical record, including diagnosis and treatment.
    iii. Create Medical Record View: Allows doctors to create a new medical record for a patient after a visit.
    iv. Update Medical Record View: Allows modification of an existing medical record.

4. Admin Site:
    a. Customize Django's admin site to manage doctors, patients, and medical records. The admin should be able to easily navigate and manage relationships between these entities.
5. Basic Functionalities:
    a. Patient Management: Admins can create, update, delete, and view patient information.

b. Doctor Management: Admins can create, update, delete, and view doctor information.
c. Medical Records: Doctors can create and update medical records for patients, while admins can manage all records.

Acceptance Criteria:

1. User Authentication: Routes are secured with Django's authentication system.
2. CRUD Operations: Doctors, patients, and medical records can be managed (created, updated, deleted, viewed).
3. Form Validation: Data submitted through forms is validated properly, and errors are handled gracefully.
4. Admin Site: The admin site is customized for easy management of doctors, patients, and medical records.
5. Error Handling: Errors are handled with appropriate responses and user feedback.

Directory structure

```
repository
|_ requirements.txt
|_ README.md
|_ src/
        |_ settings/
                |_ __init__.py
                |_ urls.py
                |_ wsgi.py
                |_ asgi.py
                |_ settings.py
        |_ web/
                |_ users/
                        |_ __init__.py
                        |_ models.py
                        |_ views.py
                        |_ urls.py
                        |_ utils.py
                        |_ admin.py
                |_ patients/
                        |_ __init__.py
                        |_ models.py
                        |_ views.py
                        |_ urls.py
                        |_ utils.py
                        |_ admin.py
        |_ manage.py
```