Zewail City of Science, Technology and Innovation

University of Science and Technology

School of Computational Sciences and Artificial Intelligence

CSAI 203 - Fall 2025

Introduction to Software Engineering

Clinic Management System

System Design Document

Team Number: 3

Team Members:

Mazen Mohamed Elbaz	202402281
Yasmeen Ayman Ebrahim	202402432
Sara Taha Tawfik	202400986

Representative Contact info:

s-yasmeen.ebrahim@zewalicity.edu.eg

1. Introduction

1.1. Purpose of the Document

This Design Document explains how the Clinic Management System (CMS) will be built during the development phase. It provides the system architecture, design models, MVC structure, UI layout, data models, and detailed interaction flows.

1.2. Scope of the Design Phase

The design phase covers the system's:

- **Architecture**
- **Module interactions**
- **MVC mapping**
- **UML diagrams**
- **UI wireframes**
- **Database schema**
- **Data dictionary**

1.3. Intended Audience

This document is intended for:

- **The course instructor and Tas**
- **Developers implementing the system**
- **Team members reviewing architecture**
- **Stakeholders validating the system's structure**

1.4. Overview of the Contents

This document contains:

- **A system overview**
- **System architecture**
- **MVC explanation**
- **UML diagrams**
- **UI wireframes**
- **Data design**
- **Conclusion summarizing design decisions**

---

2. System Overview

2.1 Brief Description of the System

The Clinic Management System is a web-based solution that allows patients to book appointments, doctors to manage diagnoses and patient records, and assistants to manage schedules. The system includes secure login, appointment booking, diagnosis entry, medical file management, and record viewing.

2.2 Key Design Goals and Constraints

Design Goals:

- **Scalability**
- **Simple and intuitive UI**
- **Secure role-based access**
- **Maintainable MVC structure**
- **Clear data organization**

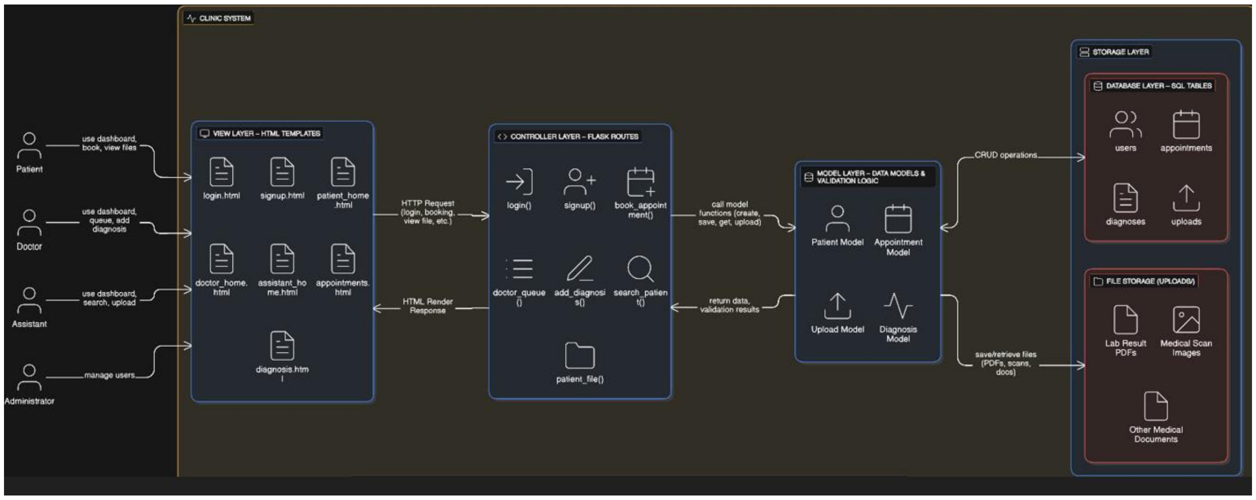
Constraints:

- **Must use Flask (Python)**
- **Must follow MVC**
- **Only basic HTML/CSS allowed**
- **System must run in a browser**

---

3. Architectural Design

3.1 System Architecture Diagram



3.2 Discussion of Architectural Style and Components

The system follows a Monolithic Architecture with a clear MVC separation.
All components run within a single Flask application:

- **View Layer: HTML templates rendered via Jinja**
- **Controller Layer: Flask routes handling logic and requests**
- **Model Layer: Classes and DB operations for Patients, Appointments, Diagnoses, Uploads**
- **Database Layer: MySQL storing all structural data**
- **File Storage: Uploaded test results and scans**

3.3 Technology Stack and Tools

- **Frontend: HTML, CSS**
- **Backend: Python Flask**
- **Database: MySQL**
- **File Storage: Local uploads folder**
- **UML Tools: Draw.io / PowerPoint**
- **IDE: VS Code / Pycharm**

---

4. Detailed Design

4.1.1 Description of MVC Pattern

The MVC pattern divides the system into three core components:

- **Model: Business logic + Database interaction**
- **View: The UI the user sees**
- **Controller: Processes requests, validates input, updates models, and returns views**

4.1.2 Mapping Components to MVC

Model:

- **Patient Model**
- **Appointment Model**

- **Diagnosis Model**
- **Upload Model**

View (HTML Templates):

- **login.html**
- **signup.html**
- **patient_home.html**
- **doctor_home.html**
- **assistant_home.html**
- **appointments.html**
- **diagnosis.html**

Controller (Flask Routes):

- **/login**
- **/signup**
- **/book**
- **/doctor/queue**
- **/diagnosis/add**
- **/assistant/search**
- **/patient/file**

4.1.3 Responsibilities

Model Responsibilities:

- **Manage DB queries**
- **Data validation**
- **Business rules**
- **Store and retrieve data**

View Responsibilities:

- **Present information**
- **Display forms and tables**

Controller Responsibilities:

- **Receive input**
- **Validate requests**
- **Call Model functions**
- **Render HTML pages**

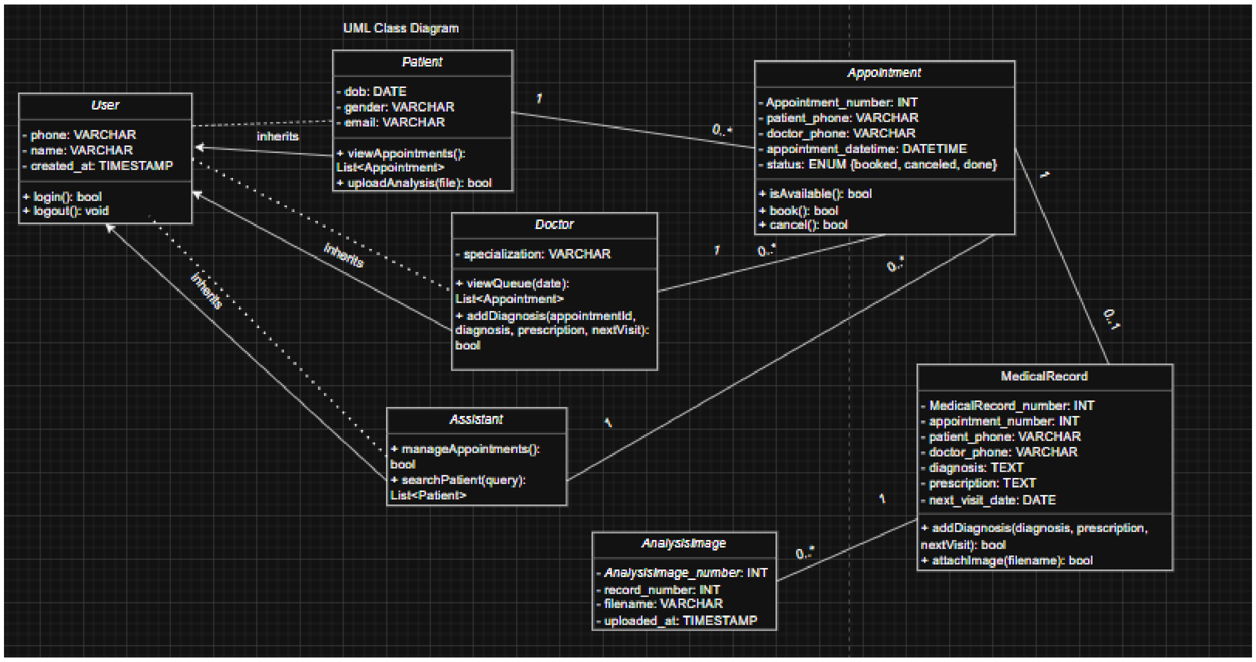
4.1.4 Interaction Between Components

Typical flow:

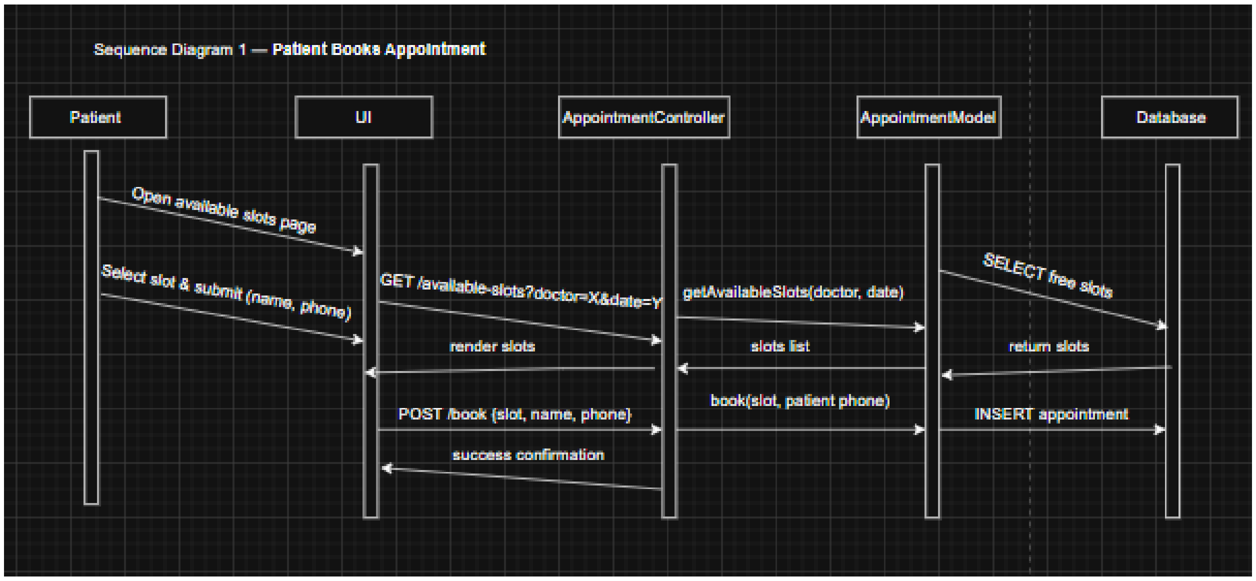
1. **User sends request → Controller**
2. **Controller validates and calls Model**
3. **Model returns data**
4. **Controller renders View**
5. **View displayed to user**

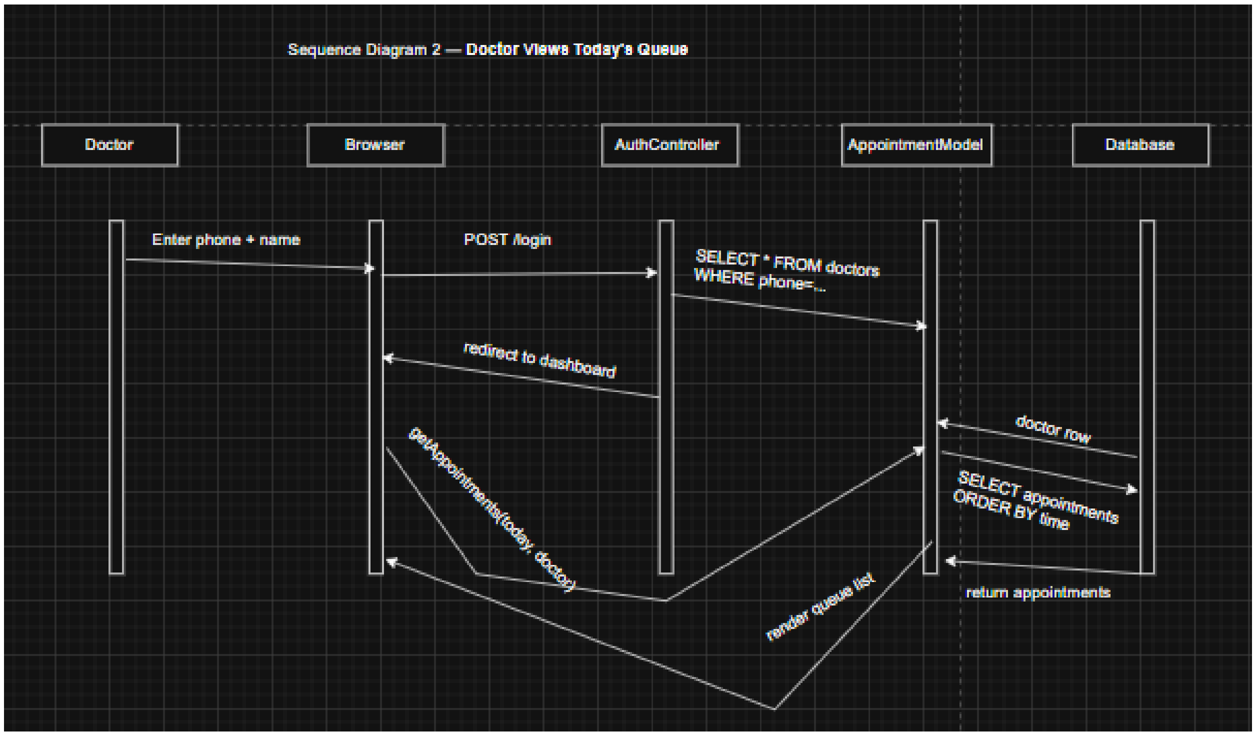
4.2 UML Diagrams

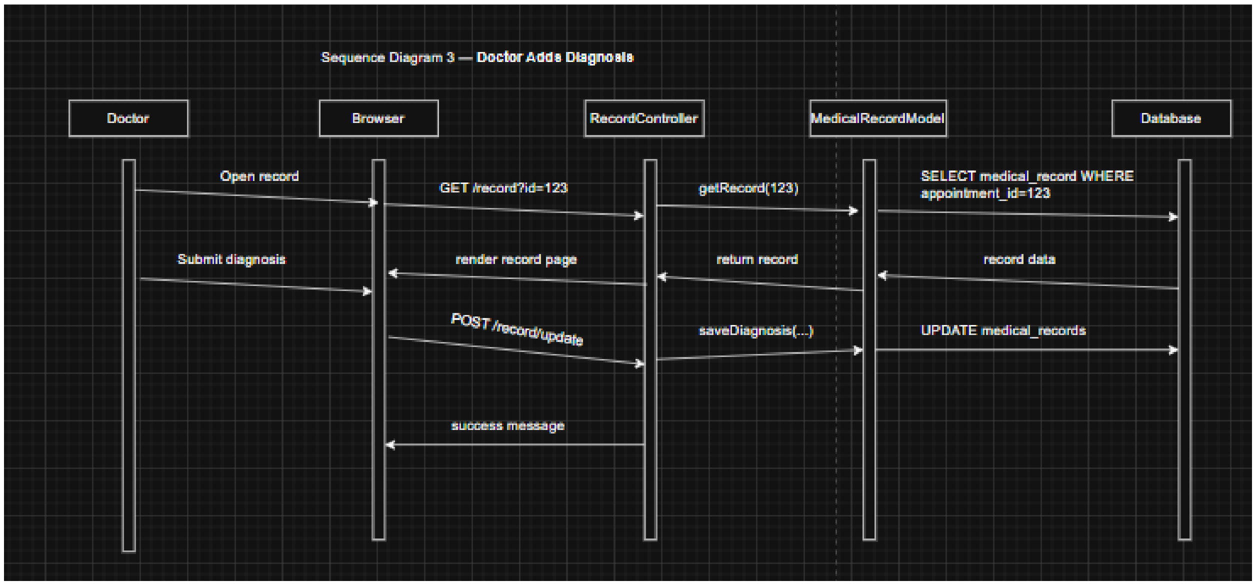
4.2.1 Detailed Class Diagram

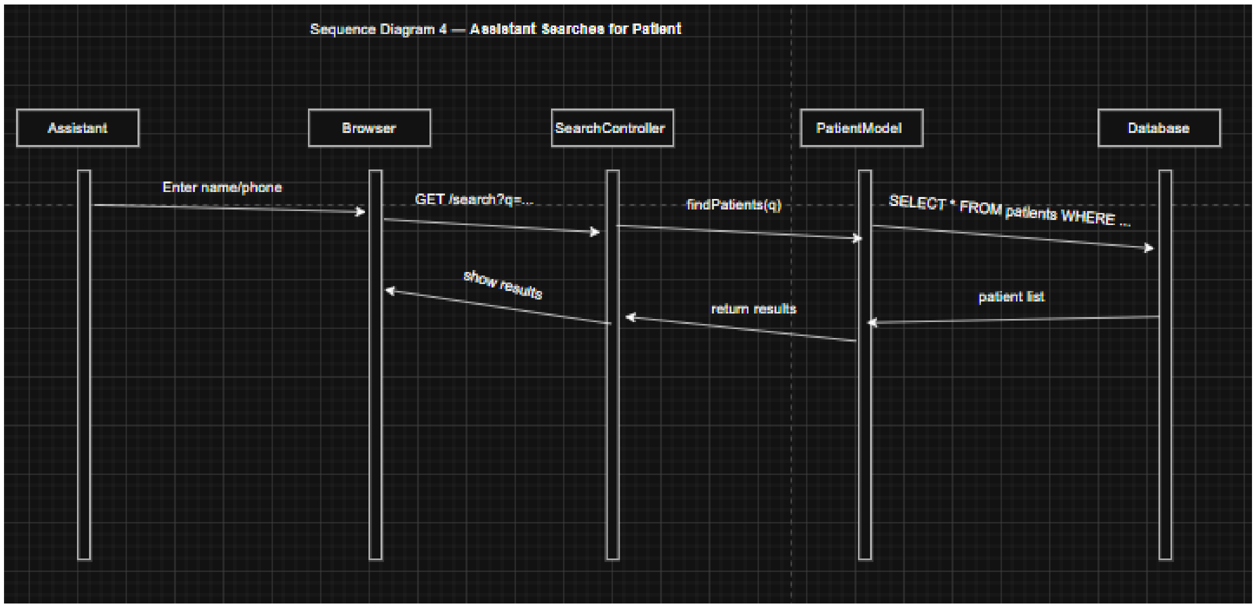


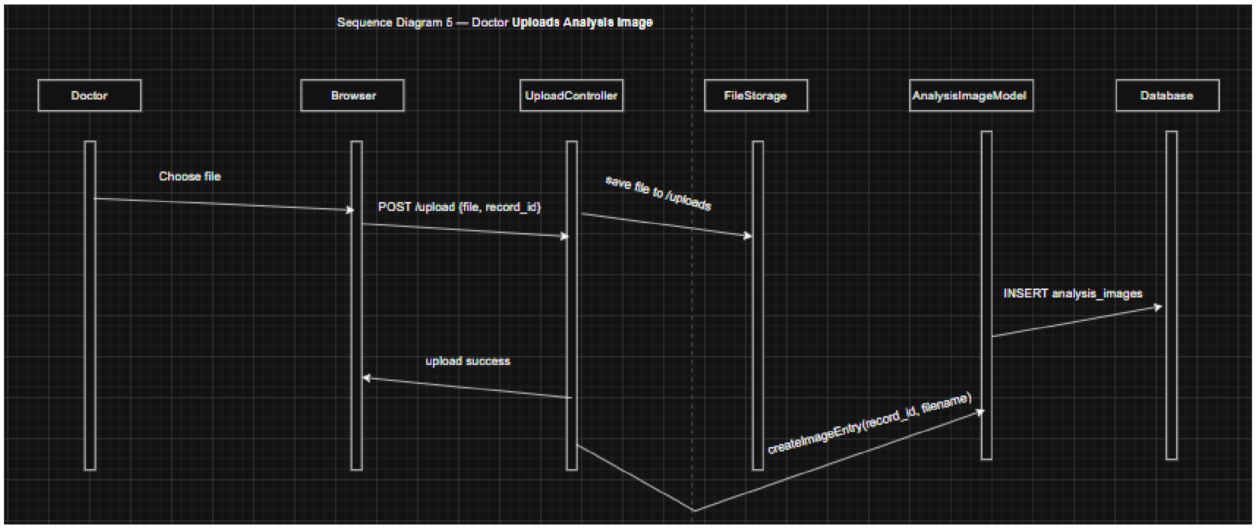
4.2.2 Sequence Diagrams


Sequence Diagram 1 — Patient Books Appointment
Patient
UI
AppointmentController
AppointmentModel
Database
Open available slots page
Select slot & submit (name, phone)
GET /available-slots?doctor=X&date=Y
getAvailableSlots(doctor, date)
SELECT free slots
render slots
slots list
return slots
POST /book {slot, name, phone}
book(slot, patient phone)
INSERT appointment
success confirmation

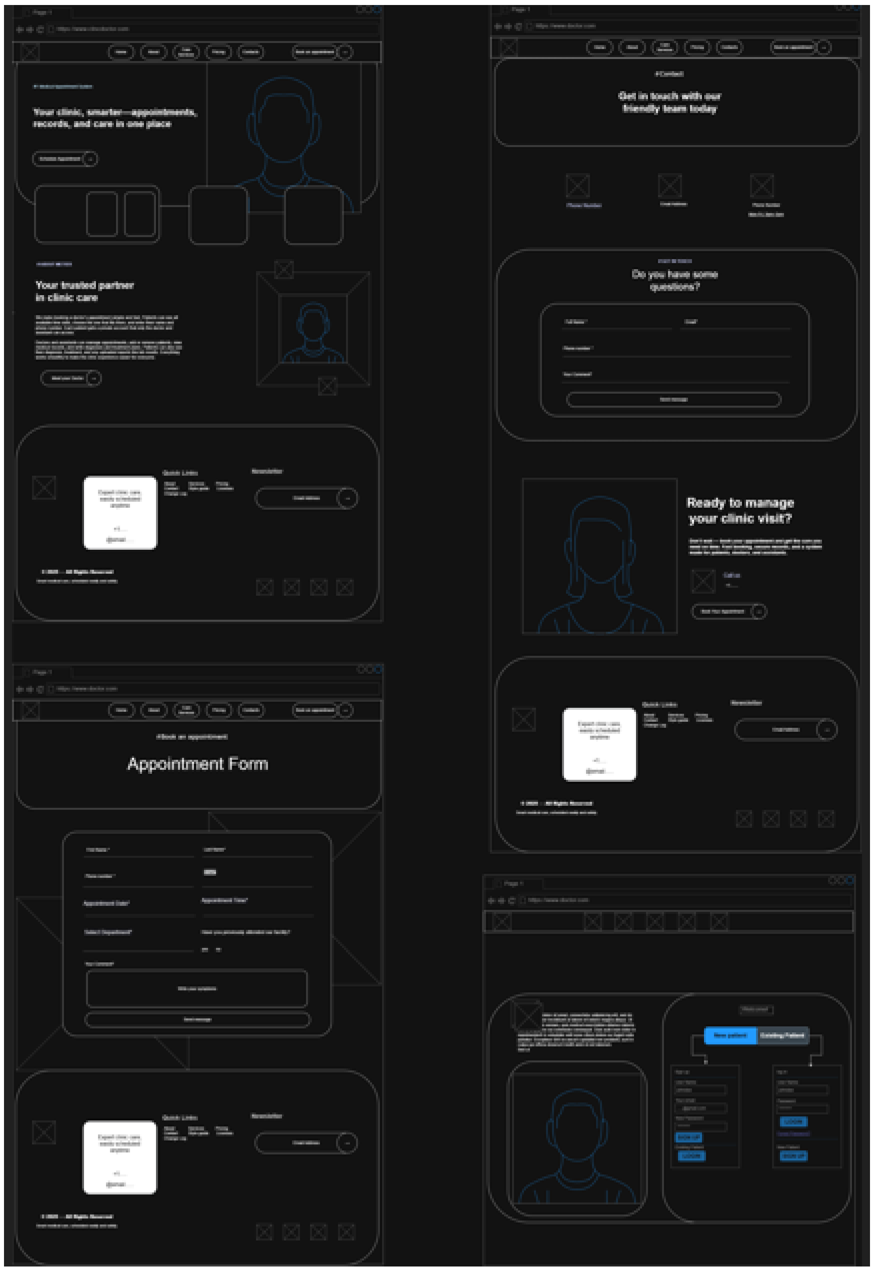

Sequence Diagram 2 — Doctor Views Today's Queue
Doctor
Browser
AuthController
AppointmentModel
Database
Enter phone + name
POST /login
SELECT * FROM doctors WHERE phone=...
redirect to dashboard
doctor row
getAppointments(today, doctor)
SELECT appointments ORDER BY time
return appointments
render queue list


Sequence Diagram 3 — Doctor Adds Diagnosis
Doctor
Browser
RecordController
MedicalRecordModel
Database
Open record
GET /record?id=123
getRecord(123)
SELECT medical_record WHERE appointment_id=123
Submit diagnosis
render record page
return record
record data
POST /record/update
saveDiagnosis(...)
UPDATE medical_records
success message


Sequence Diagram 4 — Assistant Searches for Patient
Assistant
Browser
SearchController
PatientModel
Database
Enter name/phone
GET /search?q=...
findPatients(q)
SELECT * FROM patients WHERE ...
show results
return results
patient list

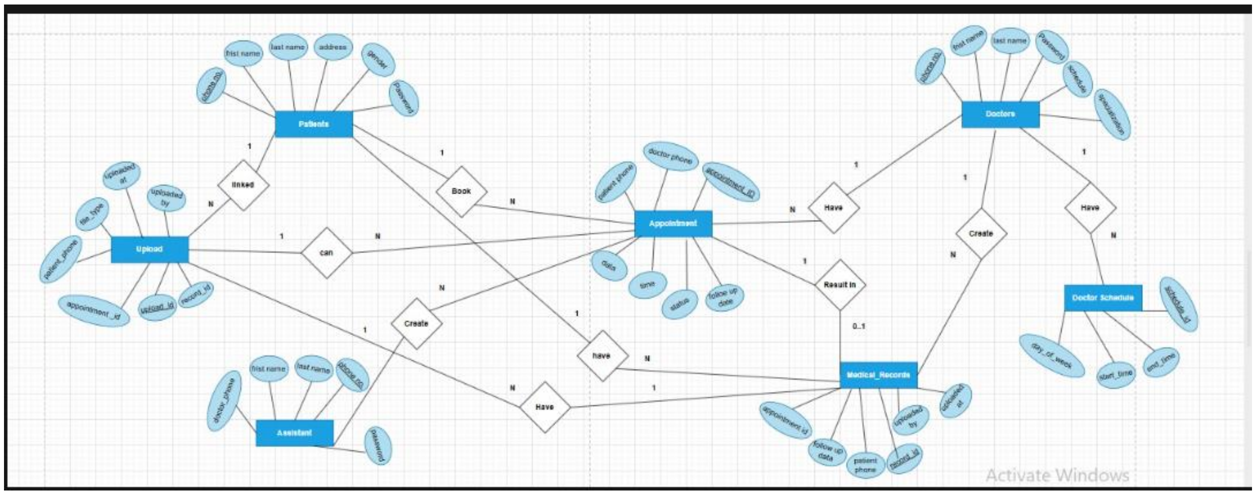


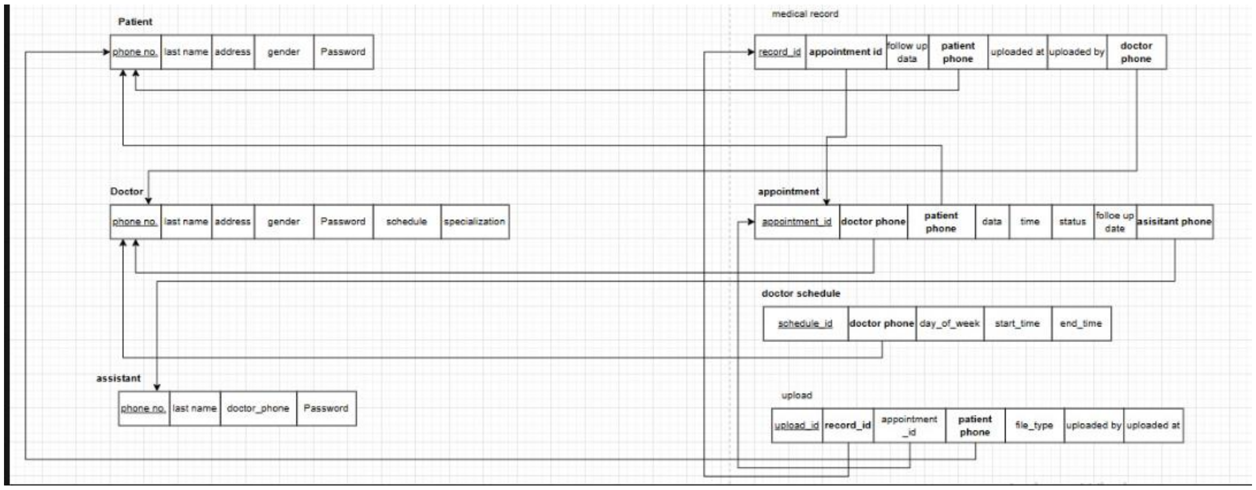
4.3 UI/UX Design


Your clinic, smarter—appointments, records, and care in one place
Your trusted partner in clinic care
Quick Links
Newsletter
Appointment Form
Get in touch with our friendly team today
Do you have some questions?
Ready to manage your clinic visit?
New patient
Existing Patient

4.4 Data Design

4.4.1 Database Schema / ER Diagram





4.4.2 File Structure / Data Storage Model

The system uses a structured file storage model to organize all uploaded files.

Files are stored inside the project directory under the `/uploads` folder.

Folder structure:

/uploads/

/uploads/labs/ → Laboratory analysis images uploaded by patients.

/uploads/scans/ → Medical scan images (X-rays, MRI, CT scans, etc.).

/uploads/docs/ → Additional documents such as prescriptions, reports, PDFs.

Each uploaded file is renamed using the format:

{phone number}_{timestamp}_{randomID}.{extension}

This ensures:

- No filename conflicts

- Easy retrieval of files

- Security and traceability

4.4.3 Data Dictionary

Table	Field	Type	Description
users	id	INT	Unique user ID
users	name	VARCHAR	User’s full name
users	phone	VARCHAR	Phone number
users	password	VARCHAR	Encrypted password
users	role	ENUM	doctor/patient/assistant
appointments	id	INT	Appointment ID
appointments	patient_id	INT	Linked patient
appointments	doctor_id	INT	Linked doctor
appointments	date	DATE	Booking date

Table	Field	Type	Description
appointments	time	TIME	Booking time
diagnoses	id	INT	Diagnosis entry
diagnoses	diagnosis	TEXT	Doctor diagnosis
diagnoses	prescription	TEXT	Treatment
uploads	id	INT	File ID
uploads	file_path	TEXT	Uploaded scan / PDF path

---

5. Conclusion

5.1 Summary of Design Phase

This document outlines the full design of the Clinic Management System, including architecture, MVC mapping, UML diagrams, UI layout, and data models. This design serves as the foundation for implementation in the next phase.