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

Software Requirements Specification (SRS)

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

The purpose of this Software Requirements Specification (SRS) document is to define the detailed requirements of the Clinic Management System (CMS). This system will help clinics manage their operations, including patient registration, appointment scheduling, doctor availability, medical record management, and billing. The document ensures all stakeholders share a clear understanding of the system before the design and implementation phases.

1.2 Scope

The Clinic Management System (CMS) is a web-based platform that allows patients to book appointments, doctors to manage schedules and records, and assistants to organize clinic operations. It provides login access for three roles: Doctor, Assistant, and Patient, ensuring each user can perform specific actions within the system.

Key functionalities include:

- Patient registration and login
- Appointment booking and management
- Doctor schedule management
- Medical record tracking
- Billing and payment tracking
- Admin dashboard for system control

The backend will be implemented using **Python Flask**, and the frontend will use **HTML**.

1.3 Definitions, Acronyms, and Abbreviations

Term	**Definition**
CMS	Clinic Management System

UI	User Interface
DB	Database
CRUD	Create, Read, Update, Delete
MVC	Model View Controller
Admin	Authorized clinic staff responsible for managing the system

1.4 References

- CSAI203 Course Project Guide (Fall 2025)
- IEEE SRS Standard Template
- Flask Documentation (flask.palletsprojects.com)

1.5 Overview

This document describes the system's functional and non-functional requirements, the use case model, and the domain model. It provides a comprehensive overview of how the CMS will operate, its constraints, and dependencies.

---

2. Overall description

2.1 Product Perspective

-The clinic management system is web-based system connected to data base that manages doctors , patient , assistant.

-Designed to manage a small clinic. To help clinics manage their daily operations.

àIt will have three main interfaces:

➢ Patient: To book appointments online with the doctor by viewing the available time slots.

➢ Doctor: To manage patient medical records, for viewing appointments and updates it.

➢ Assistant: To Schedule, reschedule, cancel and add appointments for patients.

2.2 Product Functions

• Login & Register for doctor , assistant and patient

• Book, cancel, or update appointments.

• Search functionality for patients by doctor and assistant.

• Feedback (Diagnosis & treatment) recording for patients by doctor.

• View & update patient records

• Add and view doctor schedule.

• Feedback and rating system.

2.3 User classes and characteristics

• Doctor : Manage view appointments, search patients , view & updates patient records.

• Patient : Books appointments , view patient records and rating system.

• Assistant: Help manage add & update doctor schedule, view rating ,search patient and view & updates appointments.

2.4 Operating Environment

✓ Front-end: HTML & CSS

✓ Back-end: Python Flask

✓ Database: MySQL

✓ Browser: Chrome, Firefox

2.5 Design and implementation constraints

❖ **Must use flask framework.**

❖ **Must use basics HTML & CSS only.**

❖ **Must follow MVC architecture.**

2.6 User documentation

- User manual / online help: It will be prepared to explain how to use the system step by step.

2.7 Assumptions and Dependencies

✦ **Users have good internet connection.**

✦ **The system depends on the flask web framework for backend logic**

✦ **The system stored information at the database by using MySQL**

---

3. Specific Requirements

3.1 Functional Requirements

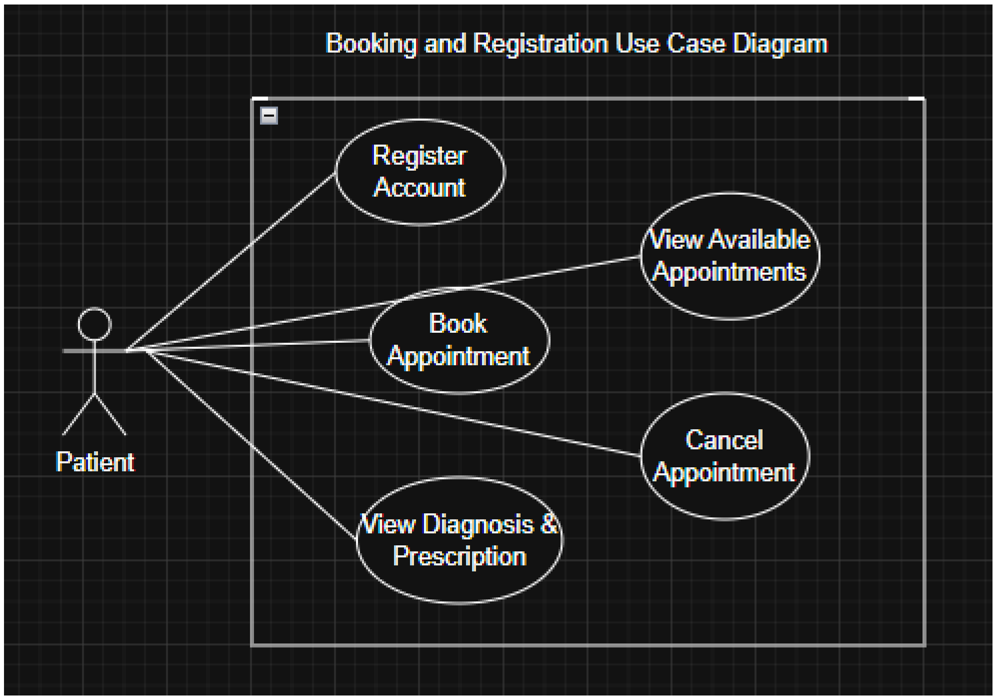
- **Patient Registration:** Patients can register using their name and phone number.
- **User Login:** All user types (doctor, assistant, patient) can log in securely.
- **View Available Appointments:** Users can view available time slots.
- **Book Appointment:** Patients can select a date and time to book.
- **Cancel Appointment:** Patients or assistants can cancel existing appointments.
- **View Patient Records:** Doctor and assistant can access patient details.
- **Add Diagnosis and Prescription:** Doctor can enter diagnosis, treatment, and follow-up date.
- **Upload Test Results:** Doctor or patient can upload medical images or test results.

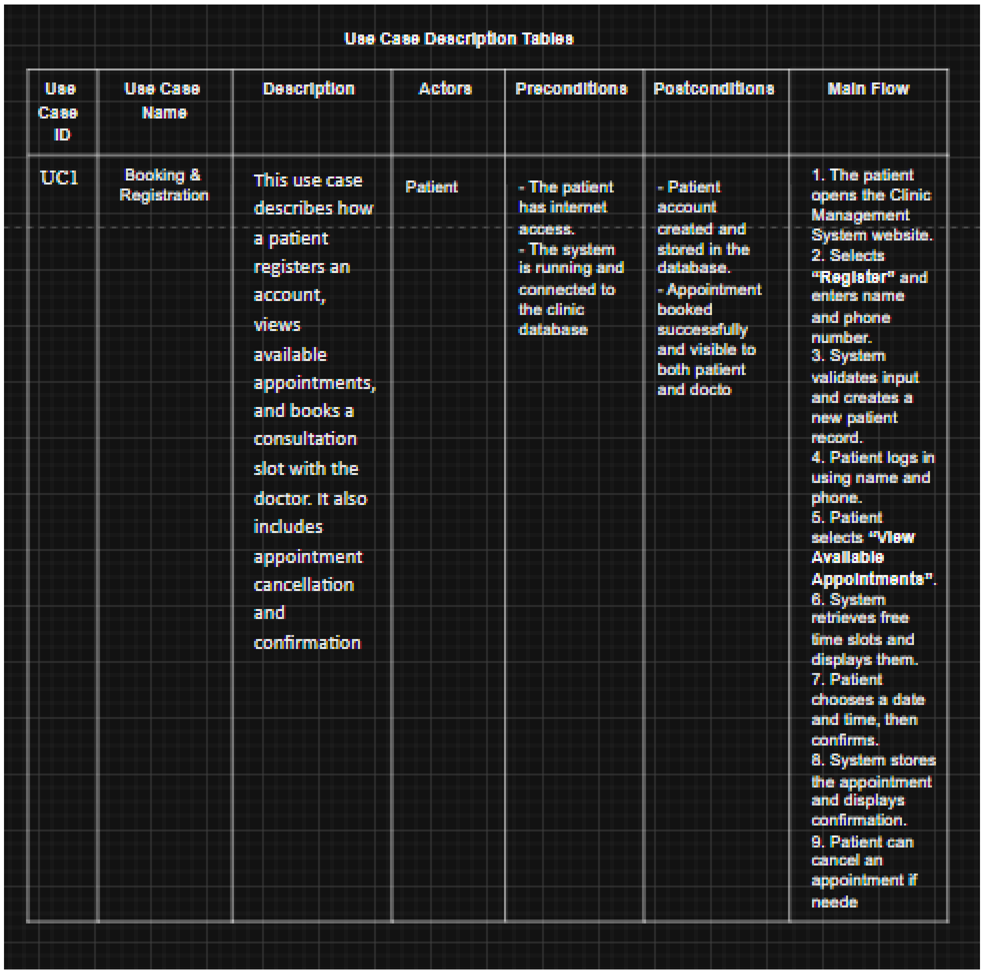
- **Search Patient:** Doctor and assistant can search patients by name or phone.
- **View Treatment History:** Doctor and patient can access past medical records

---

3.2 Use Case Model

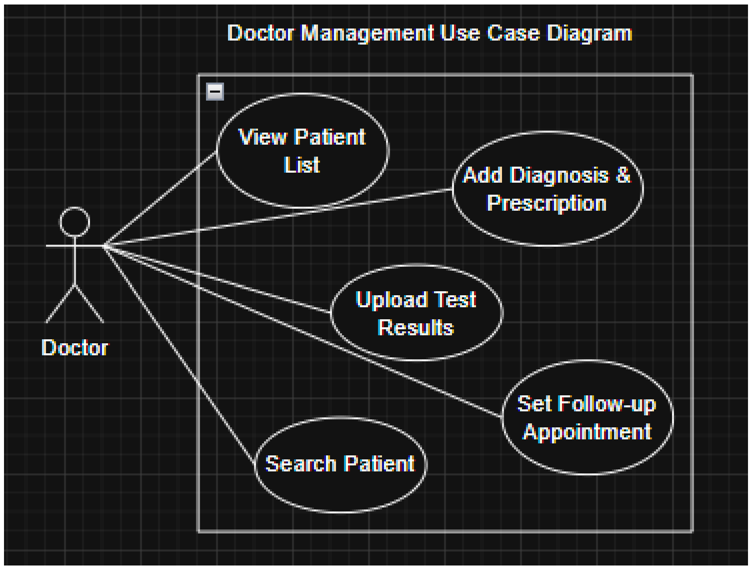
3.2.1 Use Case Diagram

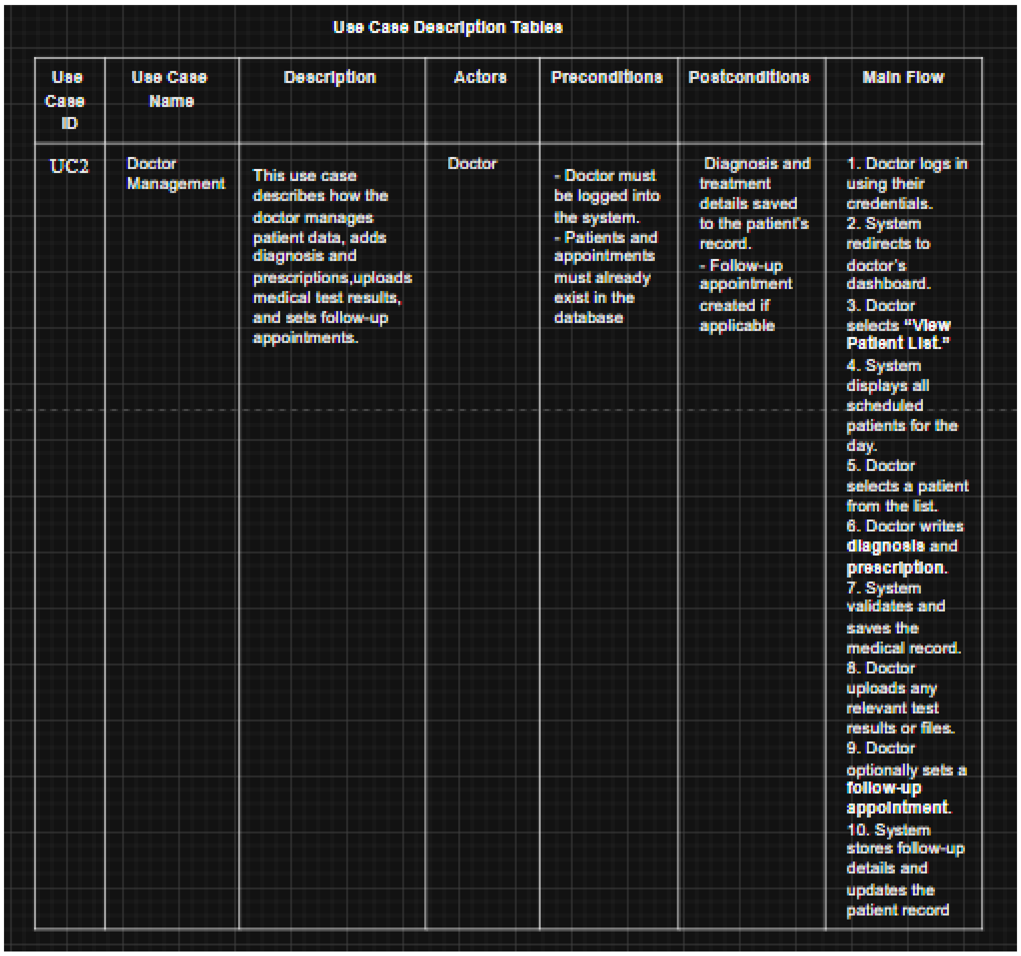

Booking and Registration Use Case Diagram
Register Account
View Available Appointments
Book Appointment
Cancel Appointment
View Diagnosis & Prescription
Patient



Use Case Description Tables

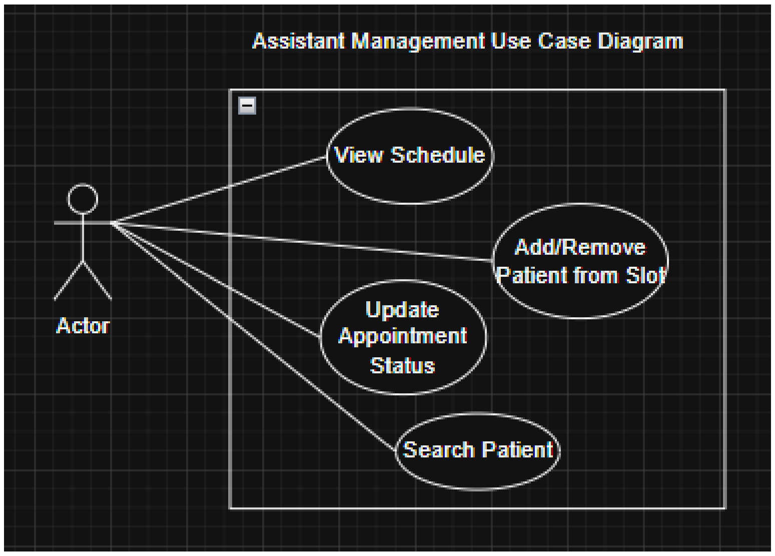
Use Case ID	Use Case Name	Description	Actors	Preconditions	Postconditions	Main Flow
UC1	Booking & Registration	This use case describes how a patient registers an account, views available appointments, and books a consultation slot with the doctor. It also includes appointment cancellation and confirmation	Patient	- The patient has internet access. - The system is running and connected to the clinic database	- Patient account created and stored in the database. - Appointment booked successfully and visible to both patient and docto	1. The patient opens the Clinic Management System website. 2. Selects **"Register"** and enters name and phone number. 3. System validates input and creates a new patient record. 4. Patient logs in using name and phone. 5. Patient selects **"View Available Appointments"**. 6. System retrieves free time slots and displays them. 7. Patient chooses a date and time, then confirms. 8. System stores the appointment and displays confirmation. 9. Patient can cancel an appointment if neede

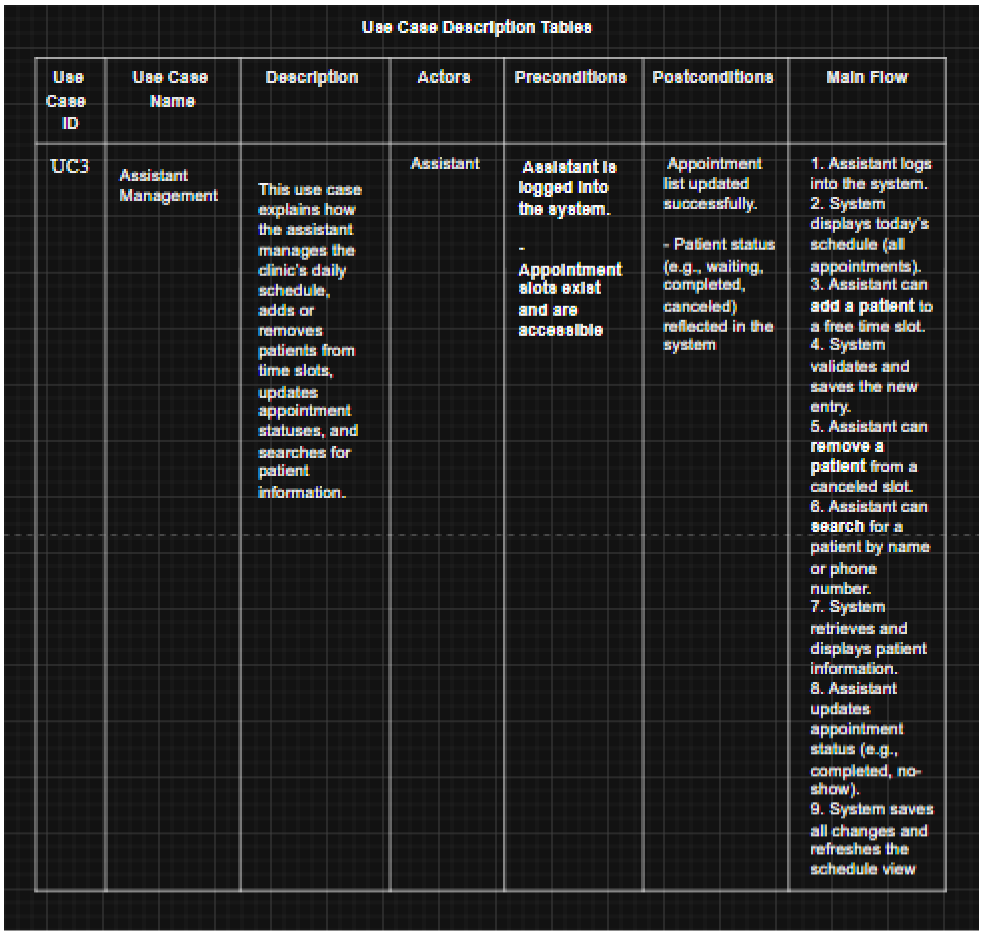

Doctor Management Use Case Diagram
Doctor
View Patient List
Add Diagnosis & Prescription
Upload Test Results
Set Follow-up Appointment
Search Patient



Use Case Description Tables

Use Case ID	Use Case Name	Description	Actors	Preconditions	Postconditions	Main Flow
UC2	Doctor Management	This use case describes how the doctor manages patient data, adds diagnosis and prescriptions,uploads medical test results, and sets follow-up appointments.	Doctor	- Doctor must be logged into the system. - Patients and appointments must already exist in the database	Diagnosis and treatment details saved to the patient's record. - Follow-up appointment created if applicable	1. Doctor logs in using their credentials. 2. System redirects to doctor's dashboard. 3. Doctor selects **"View Patient List."** 4. System displays all scheduled patients for the day. 5. Doctor selects a patient from the list. 6. Doctor writes **diagnosis** and **prescription**. 7. System validates and saves the medical record. 8. Doctor uploads any relevant test results or files. 9. Doctor optionally sets a **follow-up appointment**. 10. System stores follow-up details and updates the patient record


Assistant Management Use Case Diagram
Actor
View Schedule
Add/Remove Patient from Slot
Update Appointment Status
Search Patient



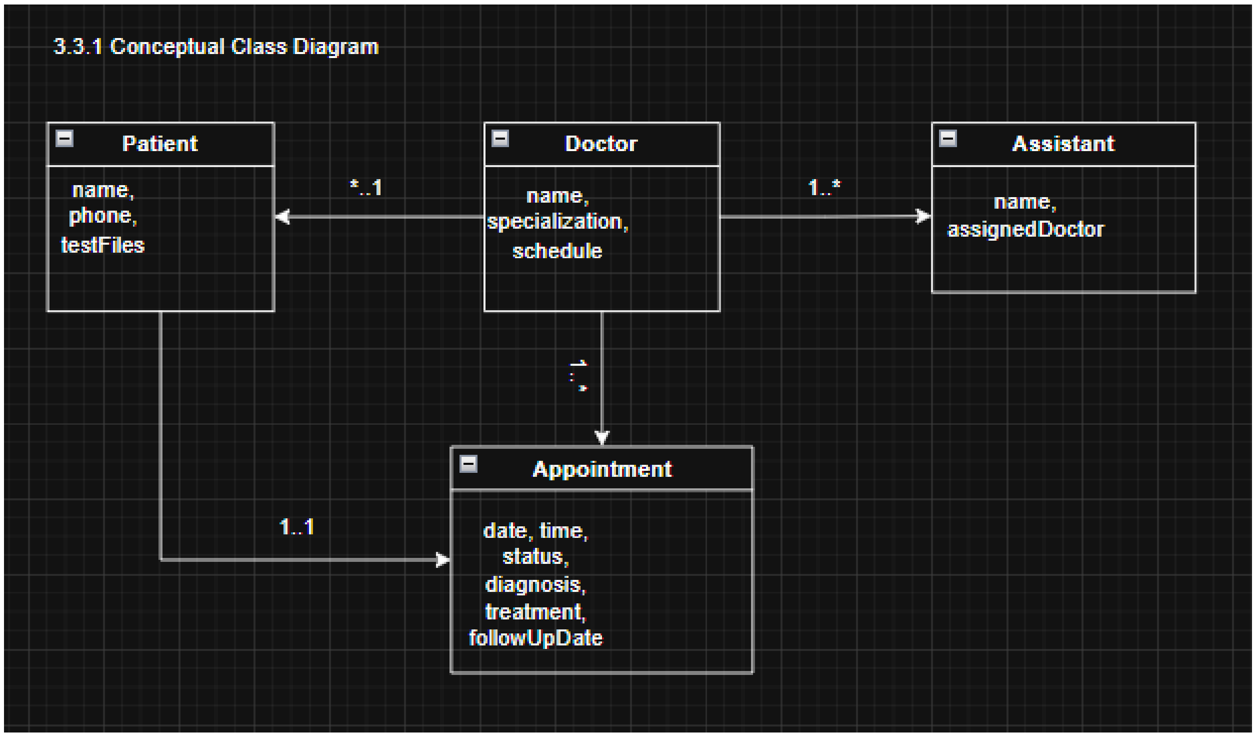
Use Case Description Tables

Use Case ID	Use Case Name	Description	Actors	Preconditions	Postconditions	Main Flow
UC3	Assistant Management	This use case explains how the assistant manages the clinic's daily schedule, adds or removes patients from time slots, updates appointment statuses, and searches for patient information.	Assistant	**Assistant is logged into the system.** - **Appointment slots exist and are accessible**	Appointment list updated successfully. - Patient status (e.g., waiting, completed, canceled) reflected in the system	1. Assistant logs into the system. 2. System displays today's schedule (all appointments). 3. Assistant can **add a patient** to a free time slot. 4. System validates and saves the new entry. 5. Assistant can **remove a patient** from a canceled slot. 6. Assistant can **search** for a patient by name or phone number. 7. System retrieves and displays patient information. 8. Assistant updates appointment status (e.g., completed, no-show). 9. System saves all changes and refreshes the schedule view

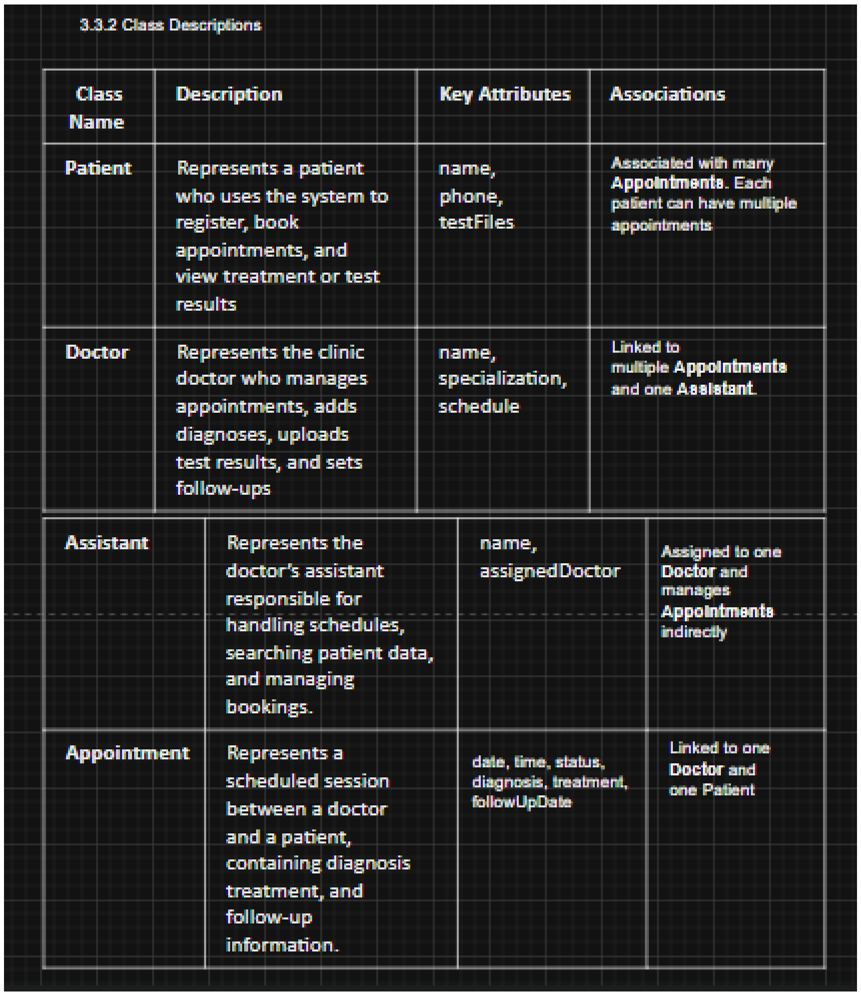
---

3.3 Domain Model

3.3.1 Conceptual Class Diagram



3.3.2 Class Descriptions



3.3.2 Class Descriptions

Class Name	Description	Key Attributes	Associations
Patient	Represents a patient who uses the system to register, book appointments, and view treatment or test results	name, phone, testFiles	Associated with many **Appointments**. Each patient can have multiple appointments
Doctor	Represents the clinic doctor who manages appointments, adds diagnoses, uploads test results, and sets follow-ups	name, specialization, schedule	Linked to multiple **Appointments** and one **Assistant**.
Assistant	Represents the doctor's assistant responsible for handling schedules, searching patient data, and managing bookings.	name, assignedDoctor	Assigned to one **Doctor** and manages **Appointments** indirectly
Appointment	Represents a scheduled session between a doctor and a patient, containing diagnosis treatment, and follow-up information.	date, time, status, diagnosis, treatment, followUpDate	Linked to one **Doctor** and one Patient

---

3.4 Non-Functional Requirements

Requirement	Description	Testing Method
Usability	Simple and intuitive interface for all users.	Usability testing with clinic staff.
Performance	Pages load within 2 seconds.	Performance test using browser tools.
Security	Role-based access control for doctor, assistant, and patient.	Access control tests.
Reliability	Data saved even if the connection is lost.	Simulated connection interruption tests.

3.5 External Interface Requirements

3.5.1 User Interface

Simple HTML forms and tables for:

- Login/Registration
- Appointment booking
- Medical record view
- Admin dashboard

3.5.2 Hardware Interface

Any computer capable of running a modern web browser.

3.5.3 Software Interface

- Flask web framework
- MySQL/SQLite database connection
- Python libraries (Flask, SQLAlchemy)

3.5.4 Communication Interface

HTTP protocol; localhost or deployed server environment.

---

4. Appendices

4.1 Appendix A: Data Dictionary

Table	Field	Type	Description
Patients	patient_id	INT	Unique patient ID
Doctors	doctor_id	INT	Unique doctor ID
Appointments	appointment_id	INT	Appointment details

Table	Field	Type	Description
MedicalRecords	record_id	INT	Patient diagnosis info
Billing	invoice_id	INT	Payment info

4.2 Appendix B: Glossary

Term	Meaning
Appointment	A scheduled meeting between doctor and patient
Invoice	Document showing patient’s payment details
Record	Medical history of a patient
