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

ClinicEase

Software Requirements Specification (SRS) Document

Team Number: 22

Team Members:

Name: Soha Essam ID: 202300335

Name: Farah Ahmed ID: 202300267

Name: Malak Mohamed ID: 202301650

Name: Mohamed Khaled ID: 202300282

Name: Nour Nader ID: 202300065

Representative Contact info:

s-nour.ayoub@zewailcity.edu.eg

8/11/2025

1. Introduction

- **1.1) Purpose:**

  This Software Requirements Specification (SRS) document is to describe the functional and non-functional requirements of the ClinicEase System. It explains the targets, characteristics, and limitations of the system to direct developers, testers, and stakeholders along the course of the project.

- **1.2) Scope:**

  A web-based clinic management system called ClinicEase was created to digitize patient information, visits, and prescription administration. Physicians can use the system to produce digital prescriptions, access patient records, and manage their schedules. In addition to viewing their medical records and scheduling or rescheduling appointments, patients can also get reminders. Administrators can oversee clinic operations, provide reports, and manage users. Technology decreases manual paperwork while increasing clinic efficiency, data quality, and communication.

- **1.3) Definitions:**

Term	Definition
SRS	Software Requirements Specification the document defining all functional and non-functional requirements of the system.
RBAC	Role-Based Access Control security model restricting actions based on user roles (Admin, Doctor, Patient).
CRUD	Create, Read, Update, Delete basic operations used to manage database entities.
UI (User Interface)	The visual elements and interaction components of the system
DB (Database)	A structured collection of data stored and managed by the system, such as patients, appointments, and prescriptions.
HTTP/HTTPS	Communication protocols used between the web client and the server for secure data transmission.

- **1.4) References:**
  - ❖ "CSAI_203 Course Project Guidelines-FALL 2025 (Dr.Mohamed Sami Rakhea)"
  - ❖ "IEEE Standard for srs (IEEE std 830-1998)"
  - ❖ "SQLite Documentation – https://www.sqlite.org/docs.html"
  - ❖ "Flask Documentation – https://flask.palletsprojects.com/"
- **1.5) Overview:**
  - This document outlines the requirements and design specifications for the *ClinicEase* system.
  - **Section 2** describes the overall system structure, users, and environment.
  - **Section 3** details the functional and non-functional requirements, use cases, and system models.
  - **Section 4** contains appendices such as the data dictionary and glossary of terms

2. Overall Description:

- **2.1 Product Perspective:**

  ClinicEase is a web-based system for patient records and appointments that digitizes the way clinics handle prescriptions, visits, and patient data. Doctors may manage schedules, write prescriptions, and access patient histories on the platform, while patients can schedule and reschedule appointments, examine medical records, and get reminders. Administrators are able to efficiently monitor clinic operations, create reports, and supervise both doctors and patients. Flask manages the backend logic, HTML/CSS/JS handle the frontend, and a relational database (MySQL/SQLite) stores all data.

- **2.2 Product Functions:**

  I. User Authentication (Login / Signup):

  The system allows patients, doctors, and administrators to create accounts and access the system securely through logging in and logging out process Authentication shall verify the identity of users through credential validation, ensuring that only authorized individuals can access system data.

  II. Role-Based Access Control (RBAC):

  The system supports Role-Based Access Control (RBAC) to manage access to system functionalities. Patients, doctors, and administrators shall have different access privileges, ensuring that each user can only view and perform actions relevant to their responsibilities.

III. Appointment Scheduling:

The system allows patients to schedule, reschedule, and cancel appointments with available doctors. Doctors shall be able to view, or modify their scheduled appointments to avoid conflicts and ensure efficient time management.

IV. Patient Records Management:

The system enables doctors to manage patient records by adding new entries, updating existing medical data, and reviewing historical health information. Each record shall include patient details, diagnosis, treatment history, and recommended treatments.

V. Prescription Management:

The system allows doctors to create, update, and manage digital prescriptions for patients. Patients shall be able to access and download their prescriptions securely through their accounts. The system shall store all prescriptions securely in the database, allowing doctors to review past prescriptions and patients to access them through their accounts. Only authorized users shall have permission to view or edit prescription data to maintain data integrity and confidentiality.

VI. Notifications:

The system should provide an automated notification feature that informs patients and doctors about appointment confirmations, reminders, cancellations, or updates. Notifications should also alert users of new prescriptions, lab results, and system messages. The system should support multiple channels such as email alerts to ensure timely communication.

VII. Dashboard:

The system should implement an interactive dashboard tailored to each user's role. The patient dashboard shall display appointment details, medical records, notifications, and prescriptions. The doctor dashboard shall provide tools for managing appointments, patient histories, and prescription creation. The administrator dashboard shall include features for user management, system monitoring, and report generation. Each dashboard offers real-time updates, easy navigation, and a summary of recent actions to improve user experience and efficiency.

VIII. Online Payment Integration:

The system shall provide a secure online payment feature that allows patients to pay consultation or service fees through integrated payment

gateways. Transactions shall be processed using secure protocols, and payment confirmations shall be recorded in the database. The system shall generate receipts automatically and make them accessible to both patients and administrators for reference.

IX. Feedback/ Rating System:

The system shall allow patients to submit feedback and rate their consultation experience after each appointment. Ratings and comments should be stored securely and made available to administrators for performance evaluation and quality improvement. Doctors shall be able to view summarized feedback to enhance service delivery while maintaining patient anonymity.

X. Chatbot:

The system should include an intelligent chatbot that assists users in performing basic tasks such as booking appointments, navigating the platform, and answering frequently asked questions. The chatbot should be available to both patients and doctors for quick assistance and should use predefined responses or integrated AI logic to provide relevant information in real time.

XI. Emergency Appointment Feature:

The system should provide an emergency appointment module that enables patients to request urgent or same-day medical consultations. The system shall automatically prioritize these requests and notify available doctors immediately. Administrators shall have the ability to monitor emergency bookings and allocate resources accordingly to ensure timely medical response.

XII. Inventory and Medicine Stock Tracking:

The system should provide a module for tracking medicine inventory and clinical supplies. Administrators and doctors shall be able to add, update, or remove stock items. The system shall automatically update stock quantities after usage or prescription issuance and generate alerts when items fall below a predefined threshold to prevent shortages.

XIII. Lab Test Results Tracking:

The system shall enable doctors to upload and manage laboratory test results associated with each patient. Patients shall be able to view and download their test results securely from their accounts. The system shall

ensure that all uploaded files are verified, properly linked to patient records, and protected against unauthorized access.

XIV. Search / filter system:

The system shall implement a search and filter mechanism that allows users to locate data across different modules, including appointments, patient records, prescriptions, and reports. Users shall be able to apply multiple filters such as patient name, appointment date, doctor specialization, or record ID. The system shall display search results in real time and ensure that only authorized users can view restricted information.

- **2.3) User Classes and Characteristics:**

User Class	Description	Technical Expertise
Doctor	Manages patient records, creates prescriptions, and views appointment schedules.	Intermediate; familiar with clinical workflows.
Patient	Books appointments, views prescriptions, manages profile, and receives reminders.	Basic computer or smartphone use.
Administrator	Oversees system management, user accounts, reports, and backups.	Intermediate to advanced technical skills.

- **2.4) Operating Environment:**
  - **Backend:** Python (Flask framework)
  - **Frontend:** HTML5, CSS3, JavaScript
  - **Database:** MySQL or SQLite
  - **Web Server:** Flask Development Server or Apache
  - **Operating System:** Windows / MacOS / Linux
  - **Browser Compatibility:** Google Chrome, Mozilla Firefox, Microsoft Edge

- **2.5) Design and Implementation Constraints:**
  - Must adhere to Flask and HTML.
  - Follows MVC architecture for code organization.
  - System must operate securely over HTTPS.
  - Database should comply with ACID properties

- No external paid APIs should be used for core functionality.

- **2.6) User Documentation:**
  - **User Manual (PDF):** Instructions for patients, doctors, and administrators.
  - **Online Help Section:** Built-in help pages explaining major functions.
  - **Quick Start Guide:** A concise guide for new users with screenshots.

- **2.7) Assumptions and Dependencies:**
  - Users have reliable internet connectivity and access to a modern web browser.
  - Doctors and patients must have valid registered accounts.
  - System depends on the backend server and connects database.
  - Notification and chatbot features depend on third-party email/SMS APIs or NLP tools (if implemented).

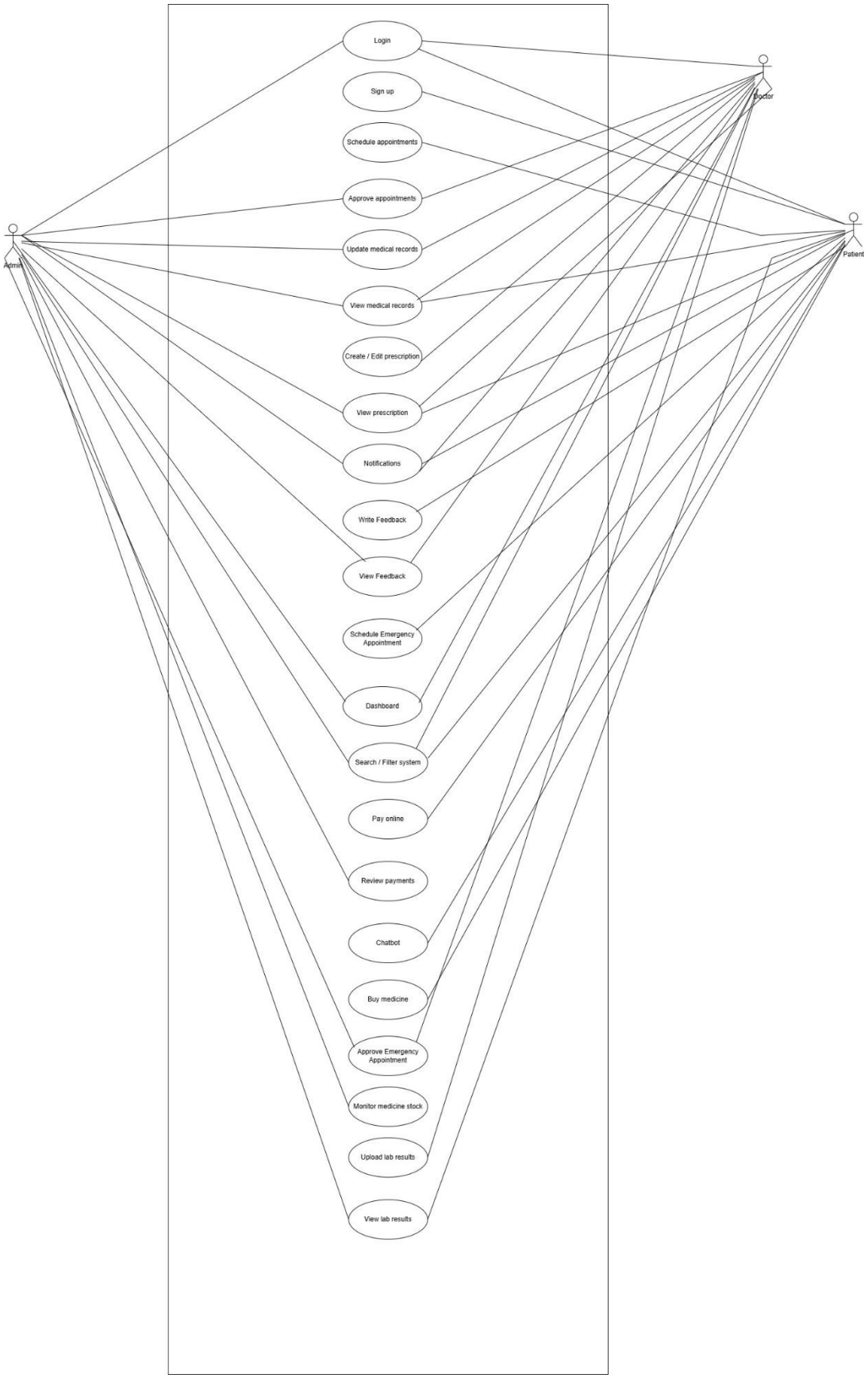
3. Specific Requirements:

- **3.1) Functional Requirements:**
  1. **User authentication (Login / Signup)**
     - Doctors, patients, and admins can securely log in and log out
     - New patients can sign up using valid personal and medical information
  2. **RBAC (Role based access control)**
     - Each type of user has access to only their respective functions and dashboard
  3. **Appointment scheduling**
     - Patients can book, cancel, or reschedule appointments with available doctors
     - Doctors can approve or reject appointment requests
  4. **Patient records management**
     - Doctors can view and update medical records
     - Records include diagnoses, prescriptions, and visit history
  5. **Prescription management**
     - Doctors can create, edit, and view prescriptions
     - Patients can access their prescriptions online
  6. **Notifications**

- Doctors receive notifications about new or canceled appointments
- Patients receive reminders about upcoming appointments

7. **Dashboard**
    - Doctor will have a dashboard that will represent the appointments details, total patients seen, etc..
    - Admin dashboard will represent the overall clinic performance.
8. **Search / filter system**
    - The patient will be able to search for a specific type of doctor
    - The doctor will be able to search for a certain patient based on the patient's name , id or date of appointment
    - The admin can search for any person or item in the clinic has access to search for almost everything about the clinic.
9. **Online payment integration**
    - Patients can pay for the appointment fees and any medicine they purchase from the clinic.
10. **Feedback / rating system**
    - After appointment patients can leave feedback or rate their doctor's service
11. **Chatbot**
    - For patients: General Inquiry Bot answers common questions (working hours, doctor list, location)
12. **Emergency appointment feature**
    - Patients can request emergency or same day appointments.
    - Doctors will receive a priority notification with the emergency appointment request
13. **Inventory and medicine stock tracking**
    - Admins can monitor medicine stock levels and receive low stock notification.
    - Patients can buy prescribed medicines through the system as well.
14. **Lab test results tracking**
    - Doctors can request lab tests for patients and upload the results when they are available.

- **3.2) Use Case Model:**
    - **3.2.1) Use Case Diagram:**


Login
Sign up
Schedule appointments
Approve appointments
Update medical records
View medical records
Create / Edit prescription
View prescription
Notifications
Write Feedback
View Feedback
Schedule Emergency Appointment
Dashboard
Search / Filter system
Pay online
Review payments
Chatbot
Buy medicine
Approve Emergency Appointment
Monitor medicine stock
Upload lab results
View lab results
Admin
Doctor
Patient

- **3.2.2) Use case description:**
  - Use Case 1: Login
    Actors: Admin, Doctor, Patient
    Description: The user enters his/her credentials to access the system.

  - Use Case 2: Sign up
    Actors: Patient
    Description: A new patient registers by providing personal details and creating credentials.

  - Use Case 3: Schedule appointment
    Actors: Patient
    Description: The patient selects a doctor, date, and time to book an appointment.
    Precondition: The patient must be logged in, and the doctor must be available.

  - Use Case 4: Approve appointment
    Actors: Doctor, Admin
    Description: The doctor reviews and approves or declines appointment requests.
    Precondition: Patient must have submitted an appointment request

  - Use Case 5: Update medical records
    Actors: Doctor, Admin
    Description: The doctor updates the patients' medical history.
    Precondition: The patients' medical records should be available to the doctor.

  - Use Case 6: View medical records
    Actors: Doctor, Patient
    Description: Users can access and view patients' medical history and past appointments
    Precondition: Patient must have a medical record

  - Use Case 7: Create / Edit prescription

Actors: Doctor
Description: Doctor creates / modifies a prescription
Precondition: Patient must have an appointment record

- Use Case 8: View prescription
  Actors: Patient, Doctor
  Description: Displays the list of prescriptions given for a certain patient
  Precondition: The prescription must exist in the system

- Use Case 9: Notifications
  Actors: Doctor, Admin, Patient
  Description: The system sends automatic notifications for appointment reminders, status changes, etc..

- Use Case 10: Write Feedback
  Actors: Patient
  Description: Patient submits feedback or rating after an appointment
  Precondition: Patient must have completed an appointment

- Use Case 11: View feedback
  Actors: Doctor, Admin
  Description: Doctors and admins review patient feedback
  Precondition: The feedback must exist

- Use Case 12: Schedule Emergency Appointment
  Actors: Patient
  Description: Patient requests an urgent same-day appointment.
  Precondition: Emergency slots must be available

- Use Case 13: Dashboard
  Actors: Admin, Doctor, Patient
  Description: Displays personalized summary information (appointments, notifications, statistics).
  Precondition: User is authenticated.

- Use Case 14: Search/Filter System

Actors: Admin, Doctor, Patient
Description: Users search and filter data (appointments, patients, doctors).
Precondition: Database contains relevant records.

- Use Case 15: Pay Online
  Actors: Patient
  Description: Patient pays consultation or service fees through online payment gateway.
  Precondition: Appointment must be confirmed.

- Use Case 16: Review Payments
  Actors: Admin
  Description: Admin monitors and reviews payment transactions.
  Precondition: Payments must exist in system.

- Use Case 17: Chatbot
  Actors: Patient
  Description: A virtual assistant answers FAQs and helps with appointment booking or navigation.
  Precondition: Patient is on the platform.

- Use Case 18: Buy Medicine
  Actors: Patient
  Description: Patient purchases prescribed medicines and is notified if any item is out of stock.
  Precondition: Prescription exists and medicine is listed in inventory.

- Use Case 19: Approve Emergency Appointment
  Actors: Doctor
  Description: Doctor approves or rejects emergency appointment requests.
  Precondition: The patient must have requested an emergency appointment

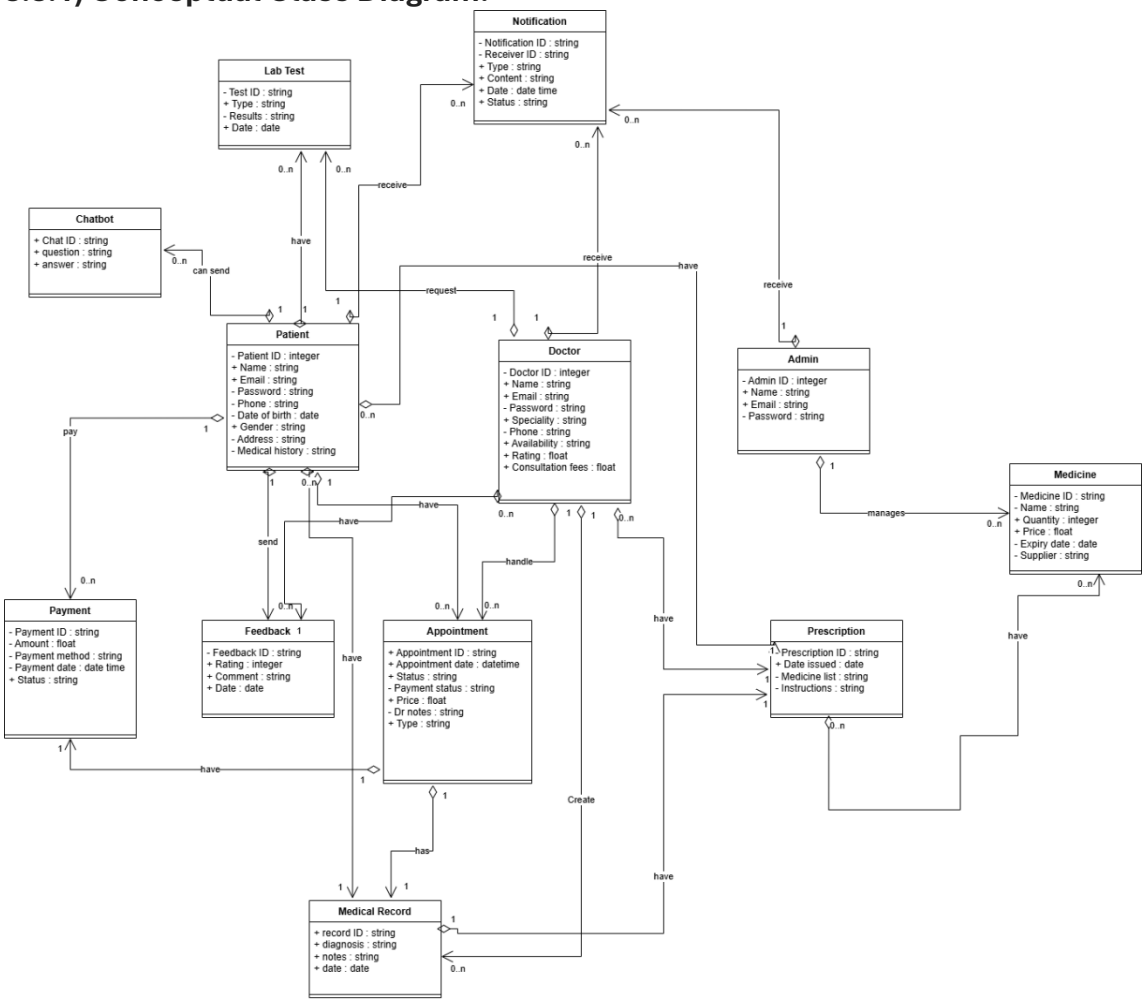
- Use Case 20: Monitor medicine stock
  Actors: Admin, Doctor

Description: System tracks available medicine quantities and alerts when stock is low.

- Use Case 21: Upload lab results
  Actors: Doctor, Admin
  Description: Doctor or admin uploads test results to the patients' medical record

- Use Case 22: View Lab Results
  Actors: Patient, Doctor
  Description: Displays uploaded lab test results for the selected patient.
  Precondition: Lab results must exist in the system.

- **3.3) Domain Model:**
  - **3.3.1) Conceptual Class Diagram:**



- **3.3.2) Class Descriptions:**

Relationships	Description	Multiplicity
Patient–Appointment	One patient can have many appointments.	1 → 0..n
Doctor–Appointment	One doctor can handle many appointments.	1 → 0..n
Appointment–Medical Record	Each appointment has exactly one medical record	1 → 1
Doctor–Medical Record	A doctor can create many medical records.	1 → 0..n
Patient–Medical Record	A patient can have many medical records.	1 → 0..n
Doctor–Prescription	A doctor can issue multiple prescriptions.	1 → 0..n
Patient–Prescription	A patient can receive many prescriptions.	1 → 0..n
Prescription–Medicine	Prescriptions can include multiple medicines, and a medicine can appear in multiple prescriptions.	0..n ↔ 0..n
Appointment–Payment	Each appointment is associated with one payment.	1 → 1
Patient–Payment	Each patient can make multiple payments.	1 → 0..n
Patient–Feedback	Each patient can send multiple feedbacks.	1 → 0..n
Doctor–Feedback	Each doctor can receive multiple feedbacks.	1 → 0..n
Doctor–Lab Test	A doctor can request multiple lab tests.	1 → 0..n
Patient–Lab Test	A patient can have multiple lab test results.	1 → 0..n
Admin–Medicine	Admin manages multiple medicines.	1 → 0..n
User–Notification	Every user can receive multiple notifications.	1 → 0..n
Patient–Chatbot	A patient can send multiple chatbot messages.	1 → 0..n
Doctor–Notification	A doctor can receive multiple notifications.	1 → 0..n
Admin–Notification	Admin can receive multiple notifications.	1 → 0..n

- **3.4) Non-functional Requirements:**
  - **Security**
    - User passwords are securely hashed, and all stored data is encrypted.
    - Role-based permissions ensure that only authorized users can carry out specific actions.
  - **Performance**
    - All dashboards and system operations should be loaded within 2 seconds during normal usage.
  - **Usability**
    - The interface should be responsive and user-friendly on both desktop and mobile devices.
    - Navigation should be intuitive, with color-coded dashboards to distinguish user roles.
  - **Reliability**
    - The system must function correctly even when multiple users access it at once.
    - It should include automatic error detection, handling, and recovery mechanisms.

- **3.5) External Interface Requirements:**
  - **3.5.1) User Interface:**
    - A web based application designed with HTML, CSS and basic JavaScript to ensure a responsive, user-friendly, and accessible interface.
  - **3.5.2) Hardware Interface:**
    - Standard PC or laptop, internet connection.
  - **3.5.3) Software Interface:**
    - Flask backend, SQLite/MySQL database.
  - **3.5.4) Communication Interface:**
    - HTTP/HTTPS protocol.

4. Appendices:

- **4.1) Appendix A: Data Dictionary**
  - **|Entity| Attribute Name| Type / Format| Description**
  - | **Patient** | **Patient_ID** | Integer (PK) | Unique patient identifier |

    | | Name | Varchar (100) | Patient's full name |

| | Email | Varchar (100) | Used for login and notifications |

| | Password | Varchar (255) | Encrypted password |

| | Phone | Varchar (15) | Patient’s contact number |

- | | Date_Of_Birth | Date | Patient’s date of birth |

| | Gender | Enum (Male/Female/Other) | Patient’s gender |

| | Address | Text | Residential address |

| | Medical_History | Text | Summary of previous illnesses and allergies |

- | **Doctor** | Doctor_ID | Integer (PK) | Unique doctor identifier |

| | Name | Varchar (100) | Doctor’s full name |

| | Email | Varchar (100) | Used for login and communication |

| | Password | Varchar (255) | Encrypted password |

| | Specialty | Varchar (100) | Doctor’s area of specialization |

| | Phone | Varchar (15) | Doctor’s contact number |

| | Availability | Text | Working hours and schedule |

| | Rating | Float | Average rating from patients |

- | **Admin** | Admin_ID | Integer (PK) | Unique admin identifier |

| | Name | Varchar (100) | Admin’s full name |

| | Email | Varchar (100) | Admin’s email for login |

| | Password | Varchar (255) | Encrypted password |

- | **Appointment** | Appointment_ID | Integer (PK) | Unique appointment identifier |

| | Patient_ID | Integer (FK) | Linked to patient |

| | Doctor_ID | Integer (FK) | Linked to doctor |

| | Appointment_Date | DateTime | Scheduled date and time |

| | Status | Enum (Pending/Approved/Cancelled/Completed) | Appointment status |

| | Payment_Status | Enum (Paid/Unpaid) | Indicates payment state |

| | Notes | Text | Doctor’s notes or remarks |

- | **Prescription** | Prescription_ID | Integer (PK) | Unique prescription ID |

| | Appointment_ID | Integer (FK) | Linked to appointment |

| | Doctor_ID | Integer (FK) | Linked to doctor |

| | Patient_ID | Integer (FK) | Linked to patient |

| | Date_Issued | Date | Date prescription created |

| | Medication_List | Text | List of prescribed medicines |

| | Instructions | Text | Dosage and usage notes |

- | **Feedback** | Feedback_ID | Integer (PK) | Unique feedback identifier |

| | Patient_ID | Integer (FK) | Linked to patient |

| | Doctor_ID | Integer (FK) | Linked to doctor |

| | Rating | Integer (1–5) | Patient’s rating for the doctor |

| | Comment | Text | Feedback or review content |

| | Date | Date | Date feedback submitted |

- | **Payment** | Payment_ID | Integer (PK) | Unique payment record |

| | Appointment_ID | Integer (FK) | Linked to appointment |

| | Amount | Decimal (10,2) | Paid amount |

| | Payment_Method | Varchar (50) | Example: “Credit Card”, “Cash” |

| | Payment_Date | DateTime | Date of payment |

| | Status | Enum (Success/Failed/Pending) | Payment transaction state |

- | **Inventory** | Medicine_ID | Integer (PK) | Unique medicine identifier |

  | | Name | Varchar (100) | Medicine name |

  | | Quantity | Integer | Available stock quantity |

  | | Price | Decimal (10,2) | Unit price of the medicine |

  | | Expiry_Date | Date | Medicine expiration date |

  | | Supplier | Varchar (100) | Supplier's name or source |

- | **Lab_Test** | Test_ID | Integer (PK) | Unique test identifier |

  | | Patient_ID | Integer (FK) | Linked to patient |

  | | Doctor_ID | Integer (FK) | Linked to doctor |

  | | Test_Type | Varchar (100) | Type of test (e.g., Blood, X-ray) |

  | | Results | Text | Uploaded test results |

  | | Date | Date | Date results were uploaded |

- | **Notification** | **Notification_ID** | Integer (PK) | Unique notification identifier |

  | | Sender_ID | Integer (FK) | Linked to the user who triggered the notification (doctor, admin, or system) |

  | | Receiver_ID | Integer (FK) | Linked to the user receiving the notification |

  | | Type | Varchar (50) | Type of notification (e.g., Appointment, Reminder, Payment, Emergency) |

  | | Message | Text | Content of the notification message |

  | | Date_Sent | DateTime | Date and time when the notification was sent |

  | | Status | Enum (Read/Unread) | Indicates whether the user has viewed the notification |

- | **Chatbot** | **Chatbot_ID** | Integer (PK) | Unique chatbot message identifier |

| | User_ID | Integer (FK) | Linked to the patient or user interacting with the chatbot |

| | Question | Text | The message or inquiry entered by the user |

| | Response | Text | The automated reply provided by the chatbot |

| | Date_Time | DateTime | Date and time of the interaction |

| | Intent | Varchar (100) | The detected purpose of the user's question (e.g., "Book Appointment", "Working Hours", "General Inquiry") |

| | Status | Enum (Resolved/Pending) | Indicates whether the chatbot was able to answer the question or escalate it to admin/doctor |

- **4.2) Appendix B: Glossary:**

Term	Definitions
Admin	A system user responsible for managing users, reports, and inventory.
Patient	A user who books appointments and views their medical history.
Doctor	A licensed practitioner who manages patient appointments and records.
Appointment	A scheduled meeting between a patient and a doctor.
Prescription	Digital record of medicines prescribed by the doctor
Inventory	A database of medicines and medical supplies tracked by the system.
Feedback	Patient rating or review of the doctor's service.
Lab Test	Medical analysis requested by the doctor, uploaded with results.
Dashboard	A personalized summary screen displaying relevant information like appointments, notifications, or clinic statistics, with a color-coded interface for user roles
Chatbot	Virtual assistant providing automated responses to common patient queries.
Role-Based Access Control (RBAC)	Security model restricting actions by user role.
CRUD	Create, Read, Update, Delete — basic database operations.
HTTP/HTTPS	Protocols used for communication between client and server.
SQLite/MySQL	Database management systems are used for storing data.
Flask	Python web framework used for the backend of Clinic Ease.
Credentials	The combination of a username and password used by a user (Admin, Doctor, or Patient) to securely log in to the system
Functional Requirements	Features the system must have to perform its core functions (e.g., Appointment Scheduling, User Authentication)
Non-Functional Requirements	Criteria used to judge the operation of the system, rather than specific behaviors (e.g., Security, Performance, Usability, Reliability)

SRS	Software Requirements Specification: The document detailing all the functional and non-functional requirements for the Clinic Ease system (i.e., this document).
Use Case	A description of a specific goal a user (or Actor) has when interacting with the system, detailing the steps and preconditions
User authentication	It is the fundamental security process that verifies a user's identity before granting them access to the Clinic Ease system.
Usability	This is a crucial Non-Functional Requirement (NFR) for the Clinic Ease system, as it determines how easy and pleasant the system is to use. A system that isn't usable will be rejected by its users, regardless of how powerful its features are
Reliability	is a critical Non-Functional Requirement (NFR) that defines the ability of the Clinic Ease system to perform its required functions under stated conditions for a specified period