OCL Constraints :

This document defines the business rules and constraints for the Hospital Management System (HMS) using Object Constraint Language (OCL).

Invariants

1.1 Patient Age Validation

- **Constraint**: Every patient must have a valid age greater than 0.
- context Patient inv: self.age > 0

1.2 Unique Email for Patients

- **Constraint**: Each patient must have a unique email address
- context Patient inv: Patient.all instances()->isUnique(p | p.email)

1.3 Unique Email for Staff

- **Constraint**: Each staff member must have a unique email address
- context Staff inv: Staff.all instances()->isUnique(s | s.email)

1.4 Appointment Slot Availability

- **Constraint**: Appointments cannot exceed the available slots for a doctor
- context Appointmen inv: self.slot <= self. doctor.available slots

1.5 Doctor Schedule Conflict Prevention

- **Constraint**: A doctor cannot have overlapping appointments
- in the context of doctor inv: self.appointments->forAll(a1, a2 | a1 <> a2 implies a1.endTime <= a2.startTime or a2.endTime <= a1.startTime)

2- Preconditions

2.1 Adding a New Patient

- **Constraint**: A patient must provide a name, email, and valid age before being added.
- context Patient::addPatient(name: String, email: String, age: Integer)
- pre: not name.isEmpty() and not email.isEmpty() and age > 0

2.2 Scheduling an Appointment

- **Constraint**: The patient must be registered in the system before scheduling an appointment.
- context Appointment::schedule(patient: Patient
- pre: Patient.all instances()->includes(patient)

3- Postconditions:

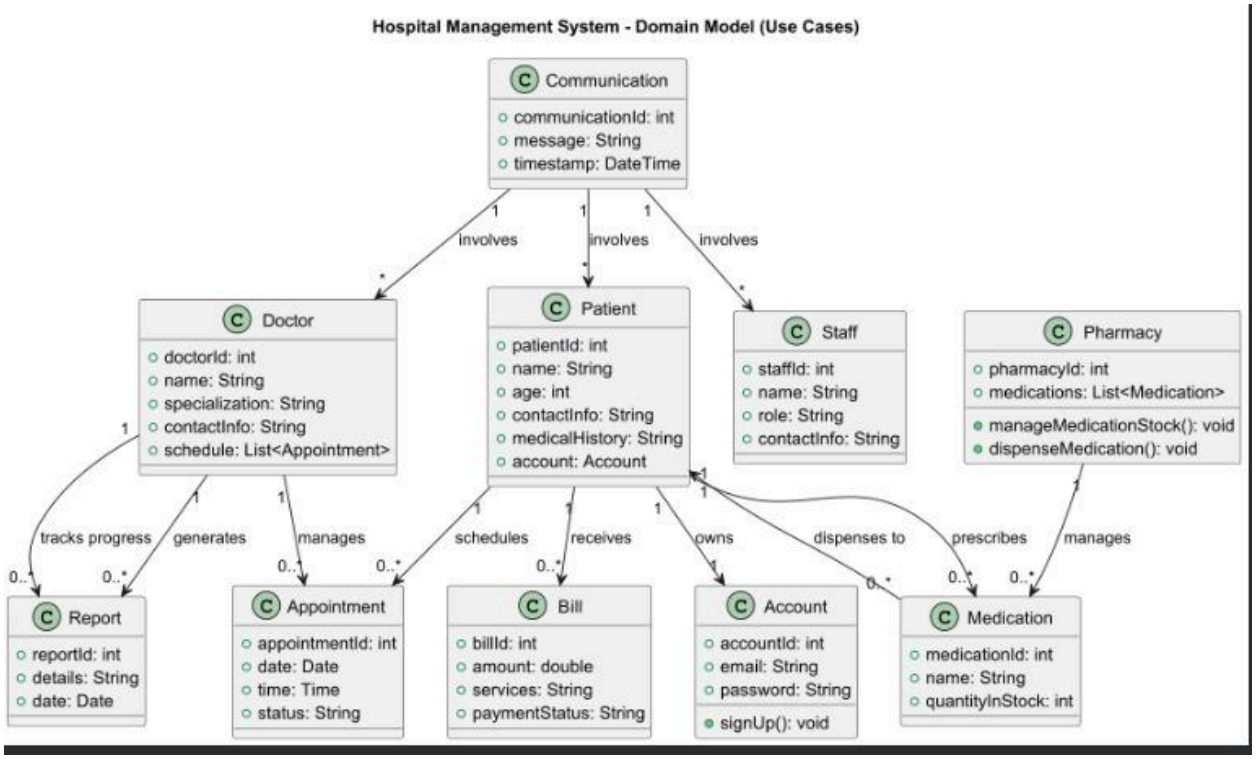
3.1 Bill Generation

- **Constraint**: After generating a bill, the total amount must equal the sum of all service costs.
- context Bill::generateBill()
- post: self.totalAmount = self.services->collect(s | s.cost)->sum()

3.2 Appointment Confirmation

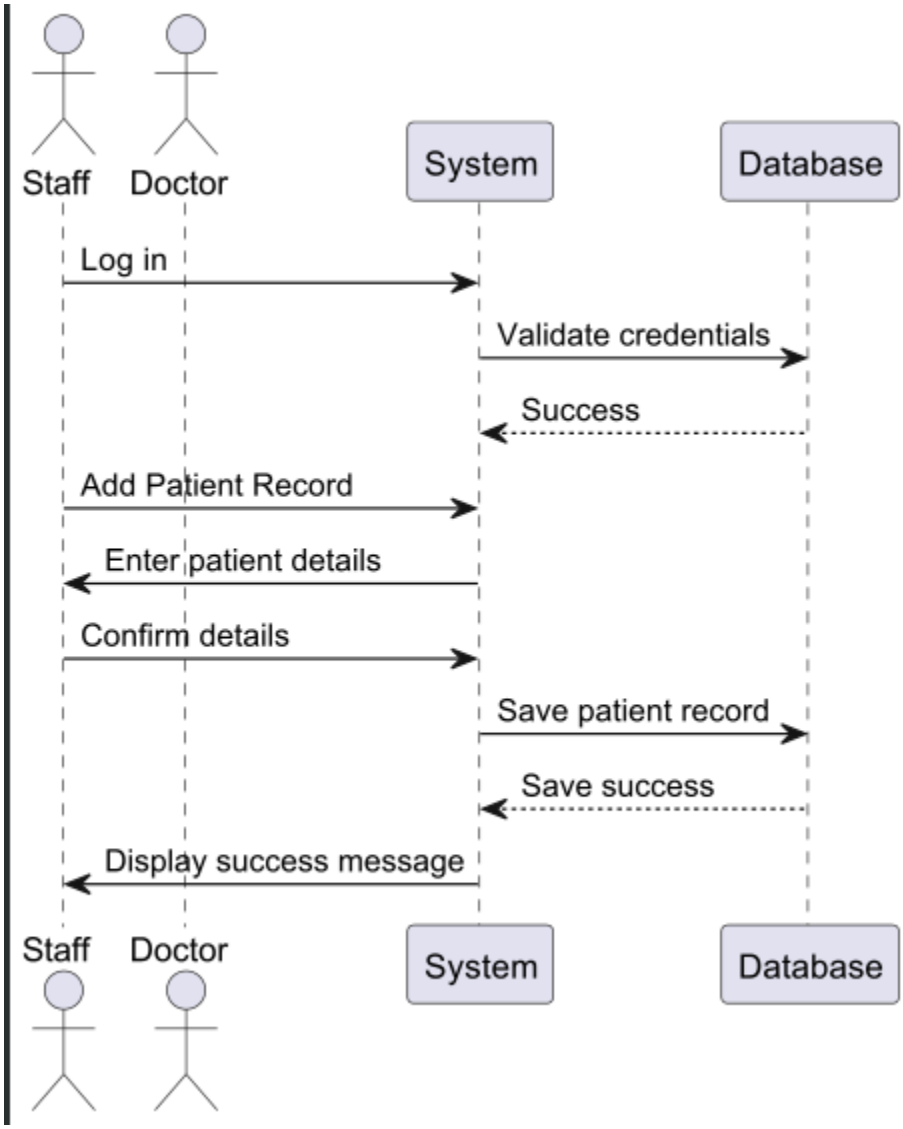
- **Constraint**: After scheduling an appointment, it must appear in the doctor's appointment list.
- context Appointment::schedule(patient: Patient, doctor: Doctor)
- post: doctor.appointments->includes(self)

2-Refined Class diagram :

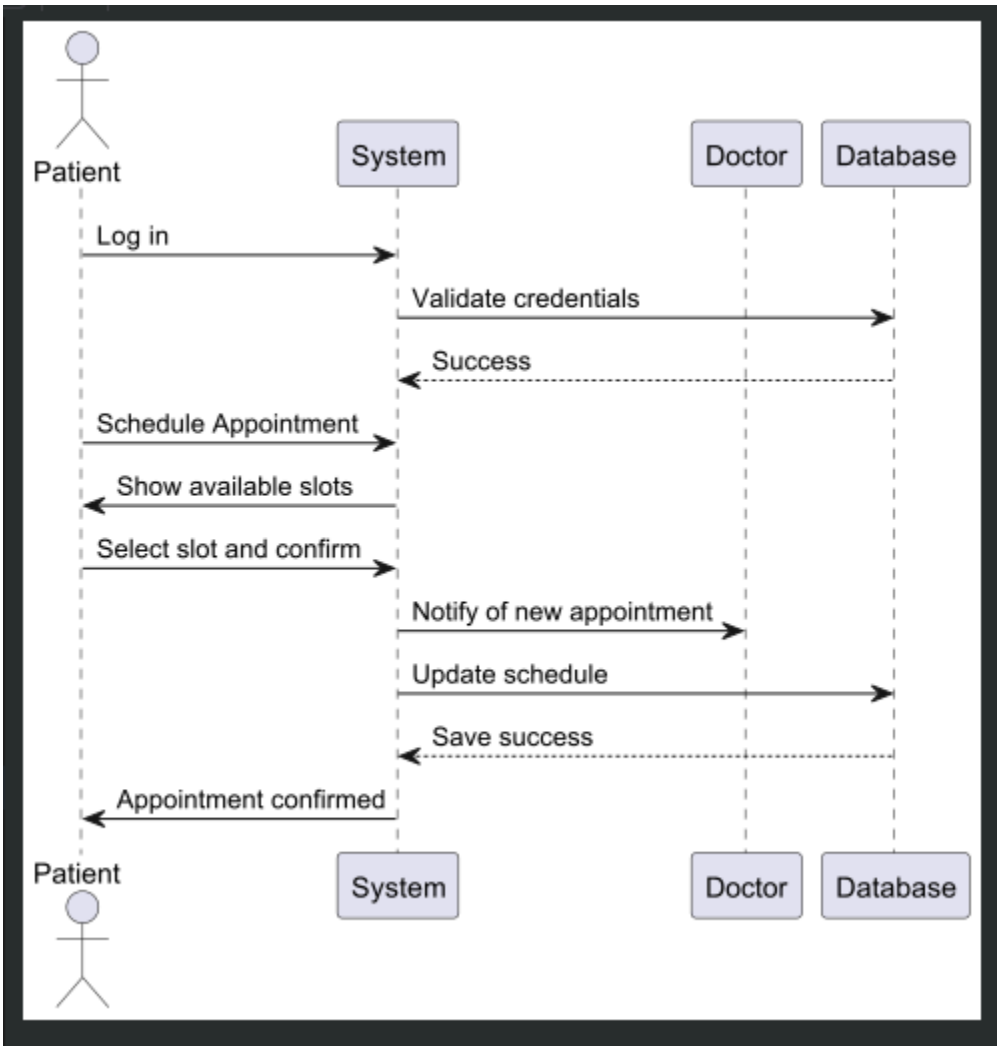


3-Interaction Diagrams:

3.1- add patient record

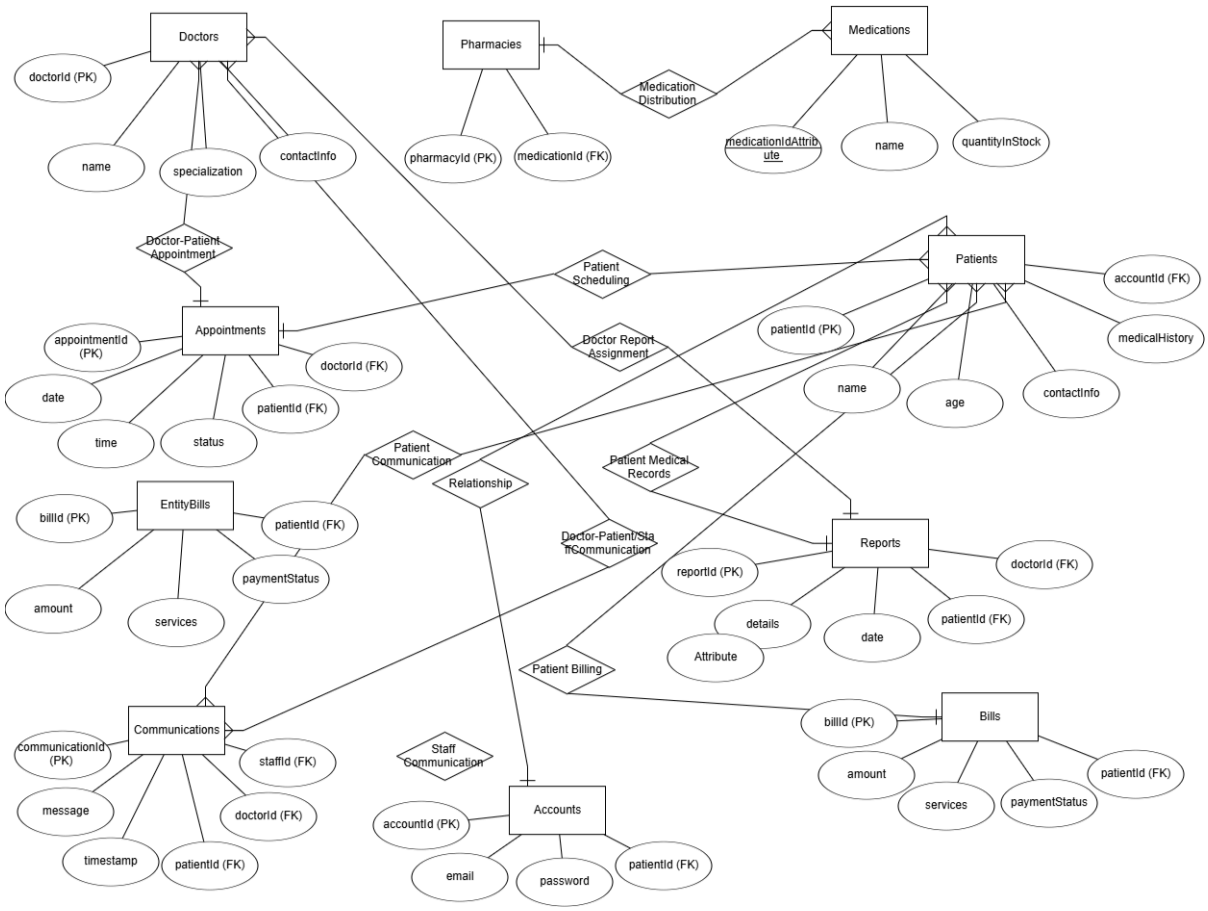


3.2-Schedule Appointment:



4-ER Diagram

-



5-UI Wireframes/Mockups

https://www.figma.com/proto/JVlX8xlR140t4zjVJuRdR9/Hospital-management-system-(Community)?node-id=1518-1935&t=FGTEyzZWSll0oVN7-1

6-Use Cases :

Use Case 1:Add Patient Record

Use Case ID	UC-001
Use Case Name	**Add patient Record**
Actors	Staff(Primary),Doctor(Secondary)
Precondition	1-The staff or doctor must be logged into the system. 2-The patient must provide valid details.
Basic Flow:	1-The actor selects "Add Patient Record" from the system menu. 2-The actor enters patient details, including name, age, contact information, and medical history. 3-The actor confirms the details. The system saves the patient record and displays a success message.

Alternative Flows:	If any mandatory fields are missing, the system displays an error message and prompts the actor to complete them.
Postconditions	The patient record is stored in the system and available for future use

Use Case 2: Schedule Appointment

Use Case Name	**Schedule Appointment**
Use Case ID	UC-002
Actors:	Patient (Primary), Doctor (Secondary)
Preconditions:	The patient must have an account and be logged into the system. Appointment slots must be available
Basic Flow:	1-The patient selects "Schedule Appointment" from the dashboard. 2-The patient chooses a doctor and views available time slots. 3-The patient selects a preferred slot and confirms. 4-The system notifies the doctor of the new appointment and updates the schedule.
Alternative Flows:	If the selected slot is no longer available, the system prompts the patient to choose another slot.
Postconditions:	The appointment is added to the system and visible to both the patient and the doctor.

Use Case 3: Generate Bill

Use case name	**generate bill**
Use Case ID	UC-003
Actor:	Staff (Primary), Patient (Secondary)
precondition:	1-The patient must have used hospital services. 2-Staff must be logged into the system
Main Flow:	1- actor select to generate bill 2- enter the name of the patient 3-actor review the data of the patient and check his services 4-system calculates the total bill amount 5-actor print the bill to the patient
Alternative Flows:	If the patient ID is invalid, the system displays an error and prompts for reentry.
Postcondition	The bill is generated and available for printing

UseCase 4 Generate Report

Use case name	**generate Report**
Use Case ID:	UC-004
Actor:	Doctor (Primary), Admin (Secondary)
Precondition:	the patient logged into the system and has already accounted for and dealt with this system

Main flow:	1-actor select generate Report 2-actor selects the patient and makes sure of data 3-actor can view, print, or email to the patient
Alternative flow:	If insufficient data exists, the system alerts the actor and cancels the operation.
Postconditions:	The report is generated and accessible to the actor

Use Case5: view patient Medical Data

Use Case Name	**View patient Medical Data**
Use Case ID	UC-005
Actors	Patient(Primary)
Precondition:	The patient already has an account and logged into the system
Main flow:	1- The patient logged into the system 2- The system verifies the patient's identity 3-system appears for the user all the treatments and next appointments 4-the patient can view or print data
Alternative flow:	if a name or email is already used system prompts for a different email.
Postconditions:	The patient successfully views their medical data.

Use Case 6: Manage Doctor Schedule

Use Case Name	Manage Doctor Schedule
Use Case ID:	UC-006
Actors:	Doctor (Primary)
Preconditions:	The doctor must be logged into the system.
Basic Flow:	1-The doctor selects "Manage Schedule" from the menu. 2-The doctor views the current schedule. 3-Doctor updates or cancels appointments as needed. 4-The system notifies affected patients of the changes.
Alternative Flows:	If the doctor tries to cancel a slot that has a booked appointment, the system prompts for confirmation.
Postconditions:	The updated schedule is saved and reflected in the system

Use Case 7:

Use Case Name	Sign Up
Use Case ID:	UC-007
Actors:	Patient (Primary), Staff (Secondary)
Preconditions:	The patient or staff must have access to the sign-up system.
Basic Flow:	.1. The patient selects "Sign Up" from the login screen. 2. The patient enters required details, such as name, email, and password. 3. The patient confirms the details and

	submits the form. 4. The system creates a new patient account and sends a confirmation email.
Alternative Flows:	If the email is already used, the system prompts for a different email.
Postconditions:	The patient account is successfully created and ready for use.

Use Case 8:

Use Case Name	View Doctor Availability
Use Case ID:	UC-008
Actors:	Patient (Primary)
Preconditions:	The patient must be logged into the system.
Basic Flow:	. 1. The patient selects "View Doctor Availability" from the menu. 2. The patient selects a doctor from the list. 3. The system displays available appointment slots.
Alternative Flows:	- If no slots are available, the system suggests an alternative doctor or time.
Postconditions:	The available slots are displayed to the patient.

Use Case 9:

Use Case Name	Update Patient Information*
Use Case ID:	UC-009
Actors:	patient (Primary)
Preconditions:	The patient must be logged into the system.
Basic Flow:	. 1. The patient selects "Update Information" from their profile. 2. The patient updates their contact information, address, or other details. 3. The patient saves the changes. 4. The system confirms the updates.
Alternative Flows:	- If mandatory fields are left incomplete, the system prompts the patient to fill them in.
Postconditions:	The patient information is updated successfully.

Use Case 10:

Use Case Name	Delete Patient Record*
Use Case ID:	UC-010

Actors:	Patient (Primary)
Preconditions:	Admin (Primary), Staff (Secondary)
Basic Flow:	1. The admin selects "Delete Patient Record". 2. The admin confirms the deletion. 3. The system removes the patient record.
Alternative Flows:	If the record is linked to ongoing treatments, the system warns the admin before deletion.
Postconditions:	The patient record is deleted from the system.

Use Case: 11

Use Case Name	Pharmacy Management
Use Case ID:	UC-011:
Actors:	- *Actors*: Pharmacist (Primary), Doctor (Secondary), Patient (Tertiary)
Preconditions:	1. The pharmacist must be logged into the system.
Basic Flow:	.1. The pharmacist selects "Manage Pharmacy". 2. The pharmacist views, adds, or removes medications in stock. 3. The pharmacist verifies patient prescriptions and dispenses medication. 4. The system updates the stock and sends notifications to the patient when medication is ready.

Alternative Flows:	- If medication stock is insufficient, the system alerts the pharmacist and prevents dispensing.
Postconditions:	Medication is dispensed, and the pharmacy stock is updated.

Use Case 12:

Use Case Name	Track Treatment Progress*
Use Case ID:	UC-012
Actors:	Doctor (Primary), Patient (Secondary)
Preconditions:	The doctor must be logged into the system.
Basic Flow:	. 1. The doctor selects "Track Treatment Progress" for a patient. 2. The doctor reviews the treatment history and notes progress. 3. The system updates
Alternative Flows:	**No Treatment Records Available** : The patient has no treatment history. 1. The system displays a message indicating no

	treatment records are available. 2. The doctor can choose to initiate a treatment plan by creating a new record.
Postconditions:	_The treatment history is updated with the latest observations and progress notes. The system ensures all changes are timestamped and saved in the patient's records. The updated treatment progress is made available for both doctor and patient for future reference.

7-Use Case Diagram:

8- Vision:

-The Hospital Management System (HMS) aims to streamline hospital operations by managing patient data, appointments, billing, and communication on a centralized platform. The system addresses the common issues of manual processes, reducing the potential for errors and delays, and improving data accuracy and patient care. The main goals are to improve operational efficiency, provide accurate patient data, and ensure smooth communication between staff, doctors, and patients.

Goals:

-Provide a centralized system for managing patient records, appointments, billing, and doctor schedules.

-Ensure seamless communication between patients, doctors, and administrative staff.

-Implement secure, real-time updates to ensure data accuracy and timely decision-making.

-Allow patients to access their medical data and schedule appointments through a user-friendly interface.

-Ensure that the system meets compliance and regulatory standards such as HIPAA for data privacy and security**.**

---

5. Supplementary Specs Document

1. Non-Functional Requirements:

- **Performance:** The system must handle up to 100 concurrent users without degradation in response time. Each user request should be processed within 3 seconds.
- **Scalability:** The system should support adding new patients, appointments, and records without affecting performance, even as the hospital grows.
- **Security:** All patient data must be encrypted both in transit and at rest. The system should implement role-based access control for different users (e.g., patients, doctors, staff).
- **Availability:** The system must have 99.9% uptime to ensure continuous access, especially during peak times.
- **Backup:** Data must be backed up daily to a secure off-site location and recoverable within 1 hour in case of failure.

6- Business/Domain Rules:

1. **Patient Management**
    - Each patient has a unique ID.
    - Medical history is updated after every visit.
2. **Appointments**
    - Only book appointments during doctors' available hours.
    - Canceled appointments notify patients and doctors immediately.
3. **Billing**
    - Bills are generated after treatment and must include all services.
    - Payment records must be secure and accessible for audits.
4. **Doctor Scheduling**
    - No overlapping appointments are allowed in a doctor's schedule.
    - Emergency appointments override regular schedules.
5. **Reports**
    - Only authorized users can generate patient reports.

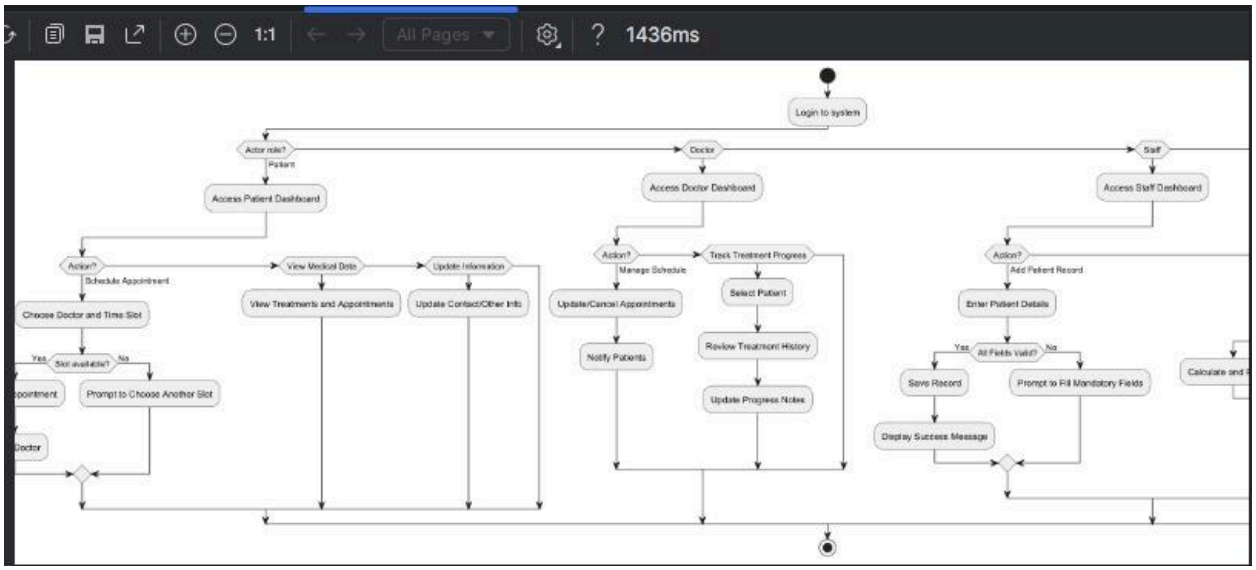
- Reports include treatment history and billing details.

7-Glossary:

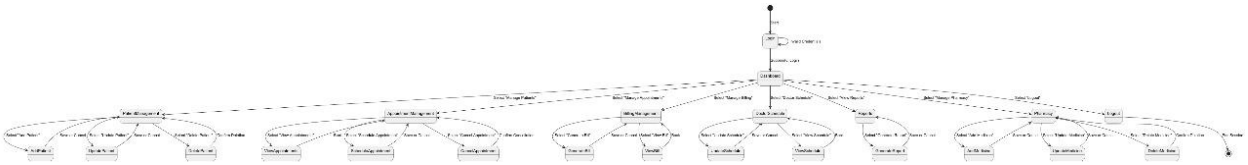
-Patient Record: A file containing all relevant medical information of a patient (e.g., medical history, treatments).
-Billing: The process of generating and managing invoices for hospital services.
-Doctor Schedule: A timetable listing all available time slots for patient appointments with doctors.
-Medical History: A record of all treatments and diagnoses that a patient has received.
-Schedule: timetable showing the availability of doctors and appointments
-System notification: An automated message sent to users about changes, reminders, or updates in the system.
-DataBackup: A process of copying and saving data to prevent loss in case of system failure
-uptime: the amount of time the system is operational and accessible to users

---

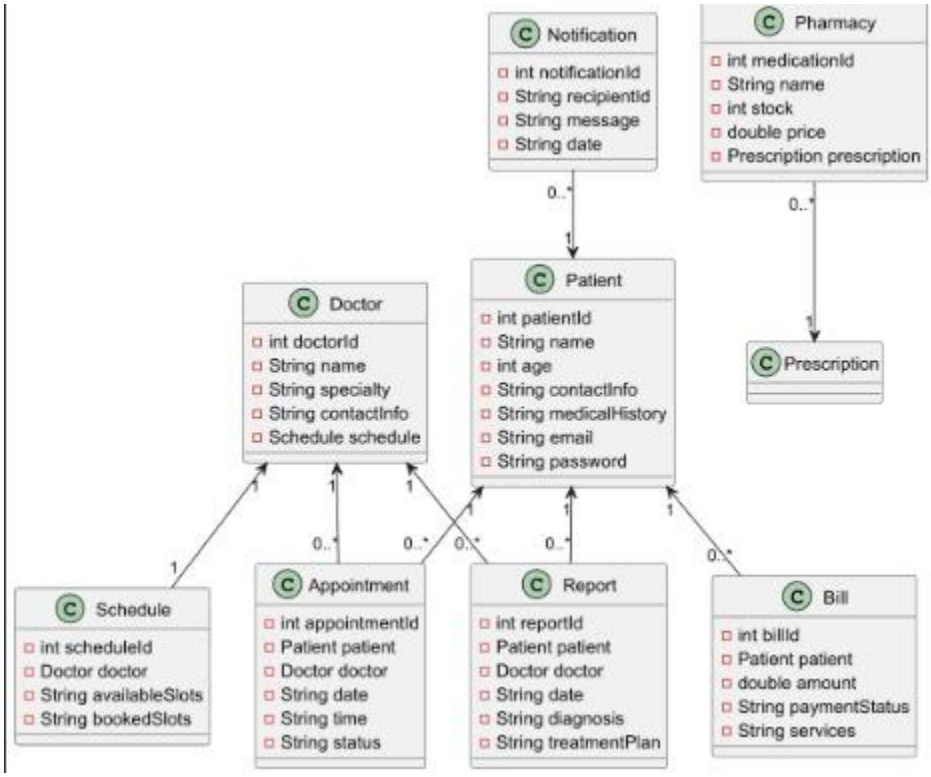
8. Activity Diagrams



9-State Machine Diagrams



10-Domain Model



Summary of Design Patterns in Your HMS Project

- Singleton Pattern: Ensures only one instance of the database connection is created and shared across the application.
- Factory Pattern: Abstracts the creation of different user types (patient, doctor, staff).
- Observer Pattern: Notifies multiple components (like patients, doctors, and staff) when a certain event, like an appointment change, occurs.

-These design patterns will help in making your system more modular, maintainable, and scalable.

Lessons Learnt

1. **Collaboration is Key:**
   Effective communication among team members was essential. Regular meetings helped clarify tasks and align goals.

2. **Time Management:**
   Setting clear deadlines for each phase of the project improved our efficiency. We learned to prioritize tasks better.

3. **Understanding OCL:**
   Working with Object Constraint Language (OCL) enhanced our understanding of business rules and constraints, crucial for system design.

4. **User-Centric Design:**
   Focusing on user experience during UI design led to a more intuitive interface, which will improve user satisfaction.

5. **Testing Importance:**
   Early and continuous testing helped identify issues before they escalated, ensuring a smoother deployment process.

6. **Adaptability:**
   Being flexible and ready to adapt to changes in project scope or requirements was vital for overcoming challenges.

7. **Documentation Value:**
   Maintaining thorough documentation throughout the project facilitated better onboarding for new team members and clarified project details.

(Who worked on which part of the project)

Sara :Interaction Diagrams.UI Wireframes/Mockups.. Domain Model . Use Cases12.Activity Diagrams (representing business flow)11.Sequence Diagrams (or Comm Diagrams)

Ahmed : Refined Class Diagram.
.Design Class DiagramsVision
. Supplementary Specs Document
Business/Domain Rules

Use Case Diagram

Basmalaa:Persistence Layer Design (ER Diagram, ORM details).
Well-commented and documented code.
State Machine Diagrams (especially for the UI Navigation)
Glossary