BANGLADESH UNIVERSITY OF PROFESSIONALS

FACULTY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (CSE)

Class Diagram

Submitted By:

Name	Roll
Mohaiminul Raju	2252421020
Shamoyeta Mourin Mouly	2252421036
Latifa Nishat Nishi	2252421062
Tahsina Tabassum Roza	2252421084
Raiyan Sarwar	2252421096

Section	: B
Semester	: 6th
Course Name	: Software Engineering Laboratory
Course Code	: CSE-3206
Date of Submission:	30.05.2025

Signature of Teacher

Class Diagram

1. Introduction

The Hospital Management System (HMS) is designed to streamline clinical and administrative processes across a hospital's operations. The class diagram presented outlines the primary entities and associations that form the backbone of the system. This structured design ensures modularity, clarity, and extensibility, ultimately promoting maintainable and scalable software architecture.

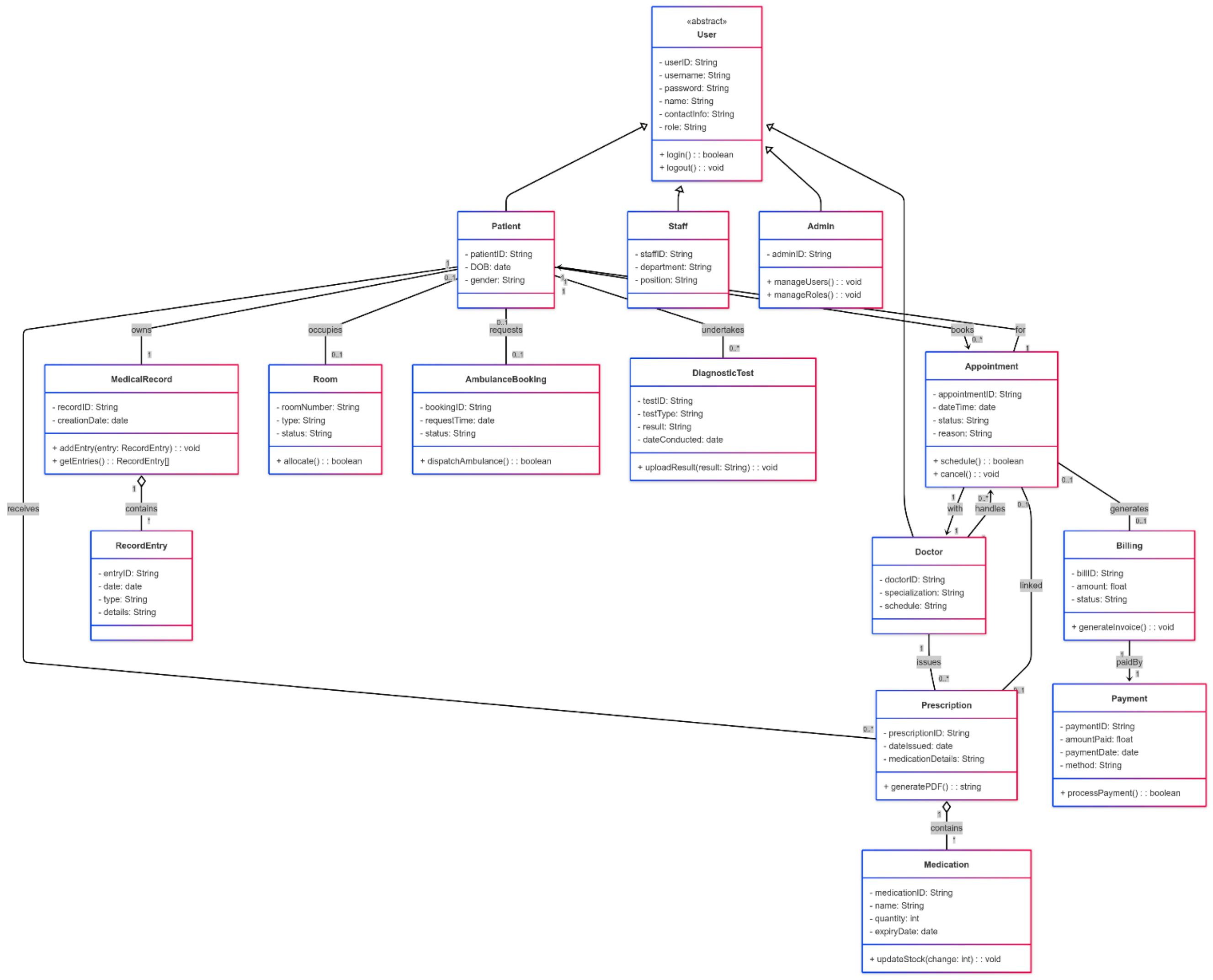

Fig 1: Class Diagram of Hospital Management System

2. Class Descriptions

2.1. User and Role Management

User (Abstract Class)

- **Purpose:** Serves as the base class for all system users. This abstraction captures common attributes and behaviors that every user—whether a patient, doctor, staff member, or administrator—must possess.
- **Attributes:**
    - **userID:** Unique identifier for the user.
    - **username:** Credential used for authentication.
    - **password:** Secure, encrypted password.
    - **name:** Full name of the user.
    - **contactInfo:** Contact details (phone, email, etc.).
    - **role:** Designates the type of user (e.g., patient, doctor).
- **Operations:**
    - **login():** Authenticates the user credentials and initiates a session.
    - **logout():** Safely terminates the user session.

Patient

- **Purpose:** Represents the end user seeking medical care. In addition to inheriting generic user properties, a patient record includes demographic details critical for personalized care.
- **Additional Attributes:**
    - **patientID:** Unique patient identifier.
    - **DOB:** Date of birth, used for age-dependent validations.
    - **gender:** Gender of the patient.

Doctor

- **Purpose:** Models the physicians who provide medical care. This class extends the base user with attributes that define their clinical profile.
- **Additional Attributes:**
    - **doctorID:** Unique identifier for the doctor.
    - **specialization:** Medical specialty or department.

    - **schedule:** Availability details and consultation timings.

Staff

- **Purpose:** Represents the administrative and supporting staff members. They contribute to non-clinical operations such as scheduling, patient registration, and facility management.
- **Additional Attributes:**
    - **staffID:** Unique identifier for the staff member.
    - **department:** Department or unit within the hospital.
    - **position:** Specific role or job title.

Admin

- **Purpose:** Embodies the system administrator responsible for managing users, roles, and maintaining overall system integrity. This role carries elevated privileges.
- **Additional Attributes:**
    - **adminID:** Unique identifier for the administrator.
- **Operations:**
    - **manageUsers():** Facilitates adding, updating, or removing user profiles.
    - **manageRoles():** Oversees the creation and assignment of user roles and permissions.

2.2. Core Functional Modules

Appointment

- **Purpose:** Captures information related to patient appointments. It defines the scheduled meeting between a patient and a doctor, including details such as date, time, and purpose.
- **Attributes:**
    - **appointmentID:** Unique number identifying the appointment.
    - **dateTime:** Scheduled date and time.
    - **status:** Indicates whether the appointment is active, cancelled, or rescheduled.
    - **reason:** Brief description of the appointment purpose.
- **Operations:**
    - **schedule():** Attempts to book the appointment, returns success status.

    - **cancel():** Cancels the appointment and updates the status accordingly.

<u>MedicalRecord</u>

- **Purpose:** Serves as the electronic repository for a patient's medical information. It aggregates the patient's ongoing clinical history.
- **Attributes:**
    - **recordID:** Unique identifier for the medical record.
    - **creationDate:** The date when the record was established.
- **Operations:**
    - **addEntry(entry: RecordEntry):** Appends a new record entry, capturing a consultation or treatment detail.
    - **getEntries():** Retrieves a collection of record entries.

<u>RecordEntry</u>

- **Purpose:** Represents individual entries within a medical record, such as consultation notes, diagnostic reports, or treatment plans.
- **Attributes:**
    - **entryID:** Unique identifier for the entry.
    - **date:** Date of the record entry.
    - **type:** Category of the record (e.g., lab report, diagnosis).
    - **details:** Detailed description of the entry.

<u>Prescription</u>

- **Purpose:** Models the prescription document issued by doctors. It captures medication instructions and relevant treatment details.
- **Attributes:**
    - **prescriptionID:** Unique identifier.
    - **dateIssued:** Date when the prescription was created.
    - **medicationDetails:** Detailed instructions for medication usage.
- **Operations:**
    - **generatePDF():** Converts the prescription into a PDF format for printing or electronic sharing.

Billing

- **Purpose:** Manages the financial charge details associated with appointments and hospital services.
- **Attributes:**
    - **billID:** Unique billing identifier.
    - **amount:** Charge amount.
    - **status:** Payment status (unpaid, paid, etc.).
- **Operations:**
    - **generateInvoice():** Compiles all service charges and prepares the invoice.

Payment

- **Purpose:** Tracks the financial transactions corresponding to bills, including payment method and confirmation information.
- **Attributes:**
    - **paymentID:** Unique identifier for the payment.
    - **amountPaid:** Amount transacted.
    - **paymentDate:** Date on which payment was made.
    - **method:** Mode of payment (credit, cash, online).
- **Operations:**
    - **processPayment():** Executes the payment procedure and returns a success or failure flag.

Room

- **Purpose:** Ensures efficient room allocation for admitted patients, tracking the type and availability of rooms within the hospital.
- **Attributes:**
    - **roomNumber:** Unique room designation.
    - **type:** Specifies if the room is ICU, general ward, or other.
    - **status:** Indicates whether the room is occupied, available, or under maintenance.
- **Operations:**
    - **allocate():** Assigns the room to a patient, updating its availability status.

AmbulanceBooking

- **Purpose:** Facilitates the booking and management of ambulance services, crucial for emergency transportation.
- **Attributes:**
    - **bookingID:** Unique identifier for the booking.
    - **requestTime:** Timestamp of the booking request.
    - **status:** Current state (e.g., dispatched, pending, cancelled).
- **Operations:**
    - **dispatchAmbulance():** Initiates the ambulance dispatch process.

DiagnosticTest

- **Purpose:** Represents the ordering and execution of diagnostic tests, along with the corresponding results.
- **Attributes:**
    - **testID:** Unique identifier for the test.
    - **testType:** Describes the kind of diagnostic test (e.g., blood test, X-ray).
    - **result:** Outcome of the test.
    - **dateConducted:** Date when the test was administered.
- **Operations:**
    - **uploadResult(result: String):** Updates the test record with the diagnostic result.

Medication

- **Purpose:** Functions as part of the medication inventory. This class tracks stock levels, expiry dates, and information about available pharmaceuticals.
- **Attributes:**
    - **medicationID:** Unique identifier for the medication.
    - **name:** Pharmaceutical name.
    - **quantity:** Current stock count.
    - **expiryDate:** Expiration date of the medication.
- **Operations:**

    - **updateStock(change: int):** Adjusts the medication quantity based on dispensing or restocking events.

3. Relationship Overview

- **User to Appointment**: A Patient may book multiple Appointments, and each Appointment is managed by one Doctor. This ensures that the patient's booking history and the doctor's schedule are consistently updated.
- **Patient to MedicalRecord:** Every Patient is associated with a single MedicalRecord. The MedicalRecord aggregates multiple RecordEntries that capture the longitudinal clinical data of the patient.
- **Prescription Flow**: Doctors issue Prescriptions to Patients. Prescriptions can also be optionally linked to specific Appointments and contain references to medications from the centralized Medication inventory.
- **Billing and Payment Integration:** Appointments may trigger the generation of Billing records, which are then reconciled through Payment transactions. This linkage ensures a seamless flow from service delivery to financial settlement.
- **Additional Services:** Optional relationships exist for Room Allocation (a patient may occupy a room) and AmbulanceBooking (a patient may request ambulance service). Furthermore, DiagnosticTests capture relevant patient testing data, contributing to a comprehensive electronic health record.

4. Conclusion

This class diagram and accompanying documentation encapsulate the fundamental structure of the Hospital Management System. By delineating clear responsibilities and relationships, the model supports design decisions that foster modularity, scalability, and security. The diagram serves as a blueprint for both implementation and further extension of system functionalities, ensuring that the HMS remains robust and responsive to evolving healthcare demands.

This documentation is designed to be a living document—subject to updates as the system evolves or as new requirements emerge. Feedback from development and testing phases will help guide future refinements.