**HEALTH MANAGEMENT SYSTEMS**

**USER STORIES**

**Clinical Module**

As a doctor, I want to be able to view a patient's medical history, so that I can make informed decisions about their treatment.

As a nurse, I need to record patient vital signs (e.g., temperature, blood pressure) in the system, so that the healthcare team can monitor the patient’s condition over time.

As a pharmacist, I want to see a patient's current medications and allergies, to safely dispense the correct drugs.

As a lab technician, I need to enter test results into the system, so that the treating physician can access and review them.

As a specialist (e.g., cardiologist), I want to access and review referrals and patient notes, to prepare for patient consultations.

As a patient, I want to view my test results and doctor's notes, to stay informed about my health.

As a healthcare administrator, I need to generate reports on patient outcomes and treatment efficacy, to improve hospital services.

**User Management Module**

As an IT administrator, I want to create and manage user accounts, to ensure that only authorized staff have access to the system.

As a staff member, I need to reset my password securely, so I can regain access to the system if I forget my password.

As a department head, I want to assign roles and permissions to my team members, to ensure they have access to the necessary system functionalities.

As an auditor, I need to view logs of user activities, to monitor and review access and usage of the system for compliance and security.

As a new employee, I need to undergo training and get system access, to start contributing to my department as soon as possible.

As a retiring staff member, my account should be deactivated, to maintain system security and data integrity.

As a system user, I want to customize my dashboard, to quickly access the information and tools I use most frequently.

These user stories aim to cover the basic functionalities and roles within the Clinical and User Management Modules of the hospital system.

**Functional Requirements**

**Clinical Module**

**Patient Management:** Create, read, update, and delete patient records, including personal information, medical history, and current health status.

**Appointment Scheduling:** Schedule, view, and manage appointments for patients with various healthcare professionals.

**Medical Records Access:** Secure access to patient medical records for authorized healthcare providers, including notes, lab results, and imaging studies.

**Medication Management:** Manage prescriptions, including medication ordering, dispensing, and monitoring for adverse reactions or interactions.

**Lab Integration:** Integrate with laboratory information systems for ordering tests and receiving results.

**Billing and Insurance Processing:** Handle billing, coding, and insurance claims processing within the clinical workflow.

**User Management Module**

**User Account Management:** Create, modify, deactivate, and delete user accounts, along with managing user roles and permissions.

**Authentication and Authorization:** Secure login processes, multi-factor authentication, and role-based access control.

**Audit Trails:** Maintain logs of user activities for security, compliance, and auditing purposes.

**Training and Support:** Provide onboarding, training, and ongoing support for system users.

**Data Privacy and Security:** Ensure compliance with healthcare data protection regulations (e.g., HIPAA, GDPR) for user and patient data.

**Non-Functional Requirements**

**Usability:** The system should be user-friendly and intuitive for various types of users, including medical staff, administrative personnel, and patients.

**Performance:** Ensure fast response times and handle large volumes of concurrent users and transactions without performance degradation.

**Scalability:** Ability to scale up to accommodate growth in patient numbers, user load, and data volume.

**Reliability and Availability:** High uptime and reliability, with backup and disaster recovery processes in place to prevent data loss.

**Security:** Implement strong security measures to protect sensitive data, including encryption, secure data storage, and regular security audits.

**Interoperability:** Ability to integrate with other hospital systems, electronic health records (EHRs), and third-party applications.

**Compliance:** Adhere to healthcare regulations and standards, such as HIPAA, GDPR, and others relevant to the healthcare industry and patient data protection.

**Development and Deployment Requirements**

**Technology Stack:** Choose a suitable technology stack that supports the system’s functional and non-functional requirements.

**Development Methodology:** Decide on a development methodology (e.g., Agile, Waterfall) that fits the project scope and team structure.

**Testing:** Comprehensive testing strategy including unit testing, integration testing, performance testing, and user acceptance testing (UAT).

**Training and Documentation:** Provide detailed documentation and training for end-users and administrators.

**Maintenance and Support:** Plan for ongoing maintenance, support, and future upgrades of the system.

**These requirements should be tailored to the specific needs of the hospital and refined through discussions with stakeholders to ensure that the system meets the operational, clinical, and administrative needs effectively.**

**Classes**

**Clinical Module**

**1. Patient**

* **Attributes:** patientID, name, dateOfBirth, gender, address, phone, medicalHistory, currentMedications,patientTypel
* **Methods:** createRecord(), updateRecord(), viewHistory(), addMedication(), scheduleAppointment(),admitPatient(),disChargePatient();

**2. Appointment**

* **Attributes:** appointmentID, patientID, doctorID, date, time, purpose
* **Methods**: schedule(), cancel(), reschedule(), notifyPatient(), notifyDoctor()

**3. Doctor**

* **Attributes:** doctorID, name, specialization, schedule, patientsList
* **Methods:** viewPatientRecord(), addNotes(), prescribeMedication(), orderLabTest(), viewSchedule(),checkBedAvailability() , assignHospitalization()

**4. Medication**

* **Attributes:** medicationID, name, dosage, sideEffects, contraindications
* **Methods:** addNewMedication(), updateMedicationInfo(), viewMedication()

**5. LabResult**

* **Attributes:** resultID, patientID, testType, date, result, normalRange
* **Methods:** recordResult(), updateResult(), viewResult()

**6. Bed**

* **Attributes**: bedID, roomNumber, isOccupied, patientID
* **Methods**: checkAvailability(), assignToPatient(), releaseBed()

**7. Hospitalization Class**

* **Attributes**: hospitalizationID, patientID, bedID, admissionDate, dischargeDate
* **Methods**: admitPatient(), dischargePatient()

**8.** **Visit class**:

* **Atrr**: id,status{ipd,opd},startdate,enddate

**User Management Module**

**1. User**

* **Attributes:** userID, name, role, email, password, lastLogin
* **Methods:** login(), logout(), resetPassword(), updateProfile()

**2. Role**

* **Attributes:** roleID, roleName, permissions
* **Methods:** createRole(), modifyRole(), assignToUser(), removeFromUser()

**3. Permission**

* **Attributes:** permissionID, name, description
* **Methods:** addPermission(), removePermission(), updatePermission()

**4. SystemAdmin**

* **Attributes**: adminID, name, email
* **Methods**: createUser(), disableUser(), updateUserPermissions()

**Additional Considerations**

* **Security:** Implement classes and methods for handling encryption, authentication, and authorization, such as Authenticator, Encryptor, and their respective methods like authenticateUser(), encryptData().
* **Integration:** Classes for handling integration with external systems, like LabSystemIntegration with methods like sendLabOrder(), receiveLabResults().
* **Notification:** A class like NotificationManager with methods such as sendEmailNotification(), sendSMSNotification() for communication with patients and staff.

**This structure provides a starting point, and additional classes and methods can be added based on the specific workflows, features, and requirements of the hospital system you are developing.**

**RELATIONSHIPS**

**Clinical Module**

1. **Patient** and **Appointment**
   * One-to-many relationship: A single patient can have multiple appointments, but each appointment is for one specific patient.
2. **Doctor** and **Appointment**
   * One-to-many relationship: A doctor can have multiple appointments with different patients, but each appointment is with one specific doctor.
3. **Patient** and **LabResult**
   * One-to-many relationship: A patient can have multiple lab results from different tests, but each lab result is associated with one patient.
4. **Doctor** and **Patient**
   * Many-to-many relationship: Doctors can see multiple patients, and patients may see multiple doctors, especially in a hospital setting with specialists.
5. **Doctor and Hospitalization**

* Many-to-many: A doctor can hospitalize multiple patients over time, and each hospitalization is overseen by multiple doctors.

1. **Patient** and **Medication**
   * Many-to-many relationship: A patient can be prescribed multiple medications, and each medication can be prescribed to multiple patients.
2. **Bed and patient**

* One-to-one for inpatients (each bed is assigned to only one patient at a time, and inpatients are assigned a specific bed).

1. **Bed and Hospitalization**

* **One-to-one during a patient's stay:** Each hospitalization record is linked to a specific bed that the patient occupies.

1. **Hospitalization Class**

* **Relationship with Patient:** One-to-many (a patient can have multiple hospitalization records over time).
* **Relationship with Bed:** One-to-one during the stay (each hospitalization record is linked to a specific bed).
* **Relationship with Visit:** One-to-One during the stay (each hospitalization record is linked to a visit bed).

**User Management Module**

1. **User** and **Role**
   * Many-to-many relationship: A user can have multiple roles (e.g., doctor, nurse, admin), and each role can be assigned to multiple users.
2. **Role and Permission**
   * Many-to-many relationship: A role can encompass multiple permissions (e.g., view, edit, delete), and each permission can be part of multiple roles.

**Cross-Module Relationships**

1. **User** (from User Management Module) and **Doctor/Patient** (from Clinical Module)
   * One-to-one relationship (in cases where one user corresponds to one role): A user account can be directly linked to a specific doctor or patient profile, especially in systems where users have distinct and singular roles.
2. **System Admin** and **User**
   * One-to-many relationship: A system administrator can manage multiple user accounts, but each user account is managed by one administrator.

**Additional Notes**

* **Aggregation and Composition**: These relationships can also represent more complex associations like aggregation and composition. For example, a doctor’s schedule (part of the **Doctor** class) may be a composition of multiple **Appointment** objects, indicating a stronger lifecycle dependency between doctor and appointments.
* **Inheritance**: There can be inheritance relationships where specific types of users (like **Doctor**, **Nurse**, or **SystemAdmin**) might inherit from a general **User** class, sharing common attributes like **userID**, **name**, and **email**, while also having role-specific attributes and methods.

These relationships are crucial for designing the database schema and object interactions within the system, ensuring that data flows logically and efficiently between different components of the hospital system.