

Prepared by : **SE1729\_NET** - **Group 5**

Lecturer : *Lê Mai Đông - DongLM6*

*Vũ Đình Khiêm* - HE170306

*Trình Ngọc Tuân* - HE163211

*Phạm Quang Thắng* - HE172671

*Trương Quang Lộc* - HE170736

*Lê Bá Hải Hà* - HE176636

– Hanoi, May 2024 –

# I. Requirement Modeling

## 1. List Actor

| No | Actor | Description |
| --- | --- | --- |
| 1 | Admin | * User management |
| 2 | Doctor | * View,Document and add or update patient medical records,report. * Request Tests and Imaging * Prescribe medication for patients to use during inpatient treatment or discharge from the hospital |
| 3 | Nurse | * View Document and add, update patient medical records. * Monitor symptoms and patient condition, reporting any changes for emergencies to doctors and to the system. |
| 4 | Receptionist | * Manage patient check-in and check-out,room locate processes, ensuring accurate record-keeping and smooth transitions. |
| 5 | Payment Gateway | * create payment transaction for patient banking fee |

## 2. Use case diagram

### 

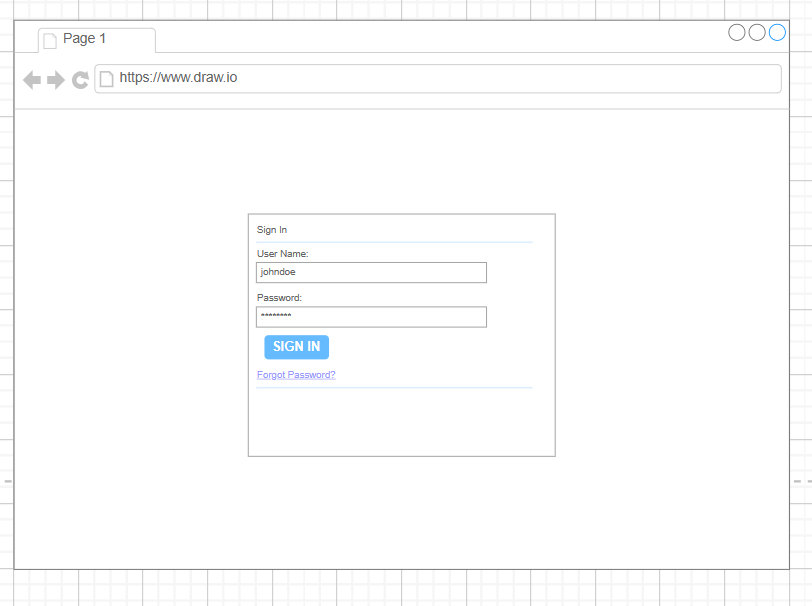
## 3. List Use Case

| **ID** | **Use case** | **Actor** | **Author** |
| --- | --- | --- | --- |
| UC-01 | Login | Admin, Doctor, Nurse, Receptionist | KhiemVDHE170306 |
| UC-02 | View Home Page | Admin, Doctor, Nurse, Receptionist | KhiemVDHE170306 |
| UC-03 | Logout | Admin, Doctor, Nurse, Receptionist | KhiemVDHE170306 |
| UC-04 | View Patient Reports | Doctor,Nurse | KhiemVDHE170306 |
| UC-05 | Prescribe Medicine | Doctor | KhiemVDHE170306 |
| UC-06 | Edit Patient Information | Doctor | KhiemVDHE170306 |
| UC-07 | View User List | Admin | TuanTNHE163211 |
| UC-08 | Active/Deactivate User | Admin | TuanTNHE163211 |
| UC-09 | Edit User | Admin | TuanTNHE163211 |
| UC-10 | Schedule Appointments | Admin | TuanTNHE163211 |
| UC-11 | Room Allocation | Receptionist | TuanTNHE163211 |
| UC-12 | Patient registration | Receptionist | ThangPQ21 |
| UC-13 | Patient Discharge | Receptionist | ThangPQ21 |
| UC-14 | View Receipt List | Receptionist | ThangPQ21 |
| UC-15 | Create Receipt | Receptionist | ThangPQ21 |
| UC-16 | Edit Receipt | Receptionist | ThangPQ21 |
| UC-17 | Delete Receipt | Receptionist | ThangPQ21 |
| UC-18 | Request Tests and Imaging | Doctor | HaLBHHE176636 |
| UC-19 | Create New User Account | Admin | TuanTNHE163211 |
| UC-20 | Authenticate User | Admin,Doctor,Nurse,Receptionist | Loctqhe170736 |
| UC-21 | View Patient List | Doctor,Nurse | Loctqhe170736 |
| UC-22 | Search Patient | Doctor,Nurse | Loctqhe170736 |
| UC-23 | View Patient Record | Doctor,Nurse | Loctqhe170736 |
| UC-24 | View Patient Record | Doctor,Nurse | Loctqhe170736 |
| UC-25 | Add Patient Record | Doctor,Nurse | Loctqhe170736 |
| UC-26 | Update Patient Record | Doctor,Nurse | Loctqhe170736 |
| UC-27 | View user details | Admin | TuanTNHE163211 |
| UC-28 | View List Role | Admin | HaLBHHE176636 |
| UC-29 | Create Role | Admin | HaLBHHE176636 |
| UC-30 | Update Role | Admin | HaLBHHE176636 |

## 4. Use Case Description

### 4.1 Login

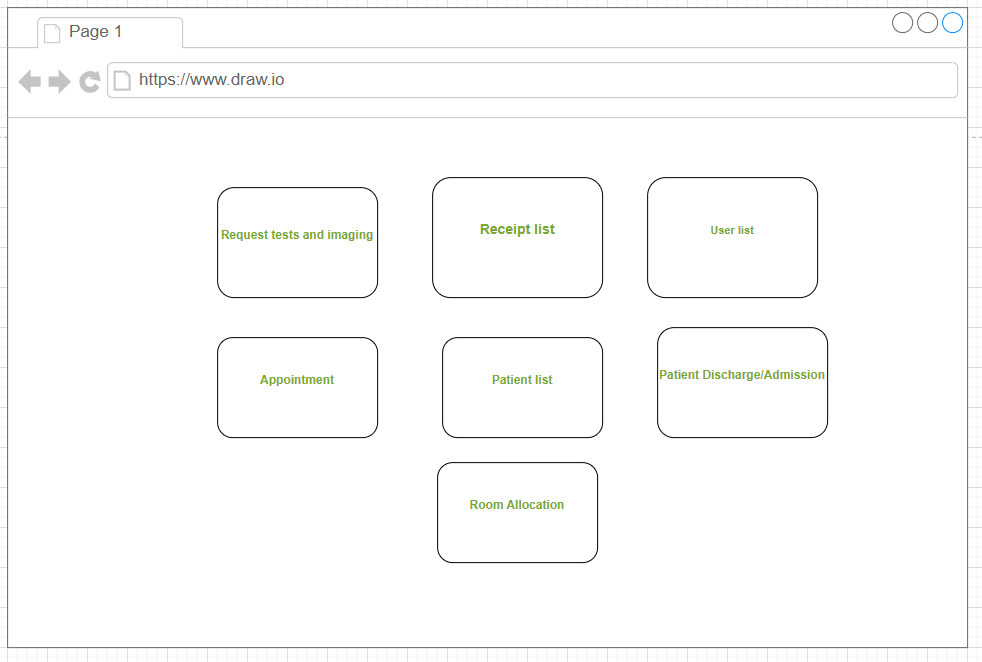
| UC ID and Name: | UC-01: Login |
| --- | --- |
| Summary | Users use their username and password to access the Hospital management system. |
| Dependency: | N/A |
| Actors: | Admin, Doctor, Nurse, Receptionist |
| Preconditions | The user must have a valid account in the system. |
| Main sequence: | 1. The user accesses the login page. 2. The system displays the login form. 3. The user enters their username and password. 4. The system validates the user's credentials. 5. If the credentials are valid, the system logs the user in and grants access. 6. Once logged in, the user can access the system's features and functions. |
| Alternative sequences: | 5. The credentials are invalid, an error message is displayed, and the user is prompted to re-enter their information. |
| Nonfunctional requirements: | **Security**: User passwords must be encrypted using **MD5 hashing**.  **Performance**: The system validates the user's credentials within equal or less than 5 seconds. |
| Postcondition: | The user is successfully logged into the hospital  management system. |
| Outstanding questions: | N/A |



### 4.2 View Home Page

#### 

| UC ID and Name: | UC-02: View Home Page |
| --- | --- |
| Summary | The home page displays room management, discharge process, patient report, bed charges and provides links to different sections of the site. Users can easily navigate to different parts of the website from the home page. |
| Dependency: | N/A |
| Actors: | Admin, Doctor, Nurse, Receptionist |
| Preconditions | The user must have a valid account in the system. |
| Main sequence: | 1.Include login use case.  2. Home page is shown and users can access the system's features and functions.  3. The home page displays room management, discharge process, patient report, bed charges options for users to access. |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | **Performance**: Access home page successfully within less than 5 seconds. |
| Postcondition: | The user gets access to features in the home page. |
| Outstanding questions: | N/A |

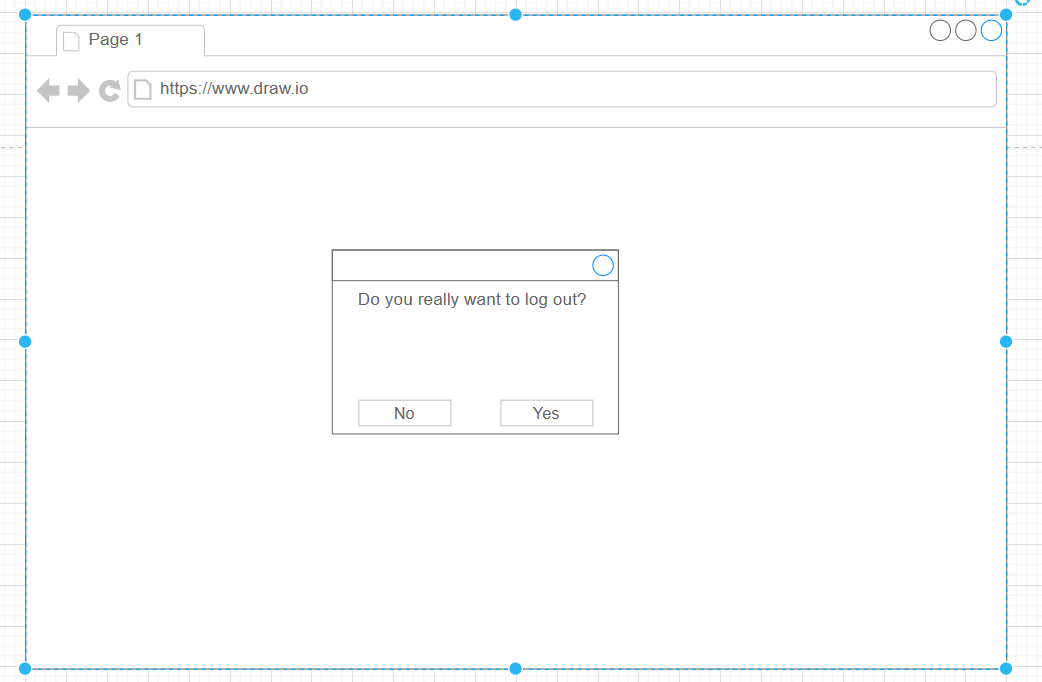


### 4.3 Logout

### 

#### 

| UC ID and Name: | UC-03: Logout |
| --- | --- |
| Summary | Users log out from the system |
| Dependency: | N/A |
| Actors: | Admin, Doctor, Nurse, Receptionist. |
| Preconditions | Users already login to the system. |
| Main sequence: | 1. User accesses the User Login screen.  2. User login successfully.  3. Home page is shown and users can access the system's features and functions.  4. User clicks on the " log out” symbol.  5. System prompts a message to ask the user to confirm.  6. User logout successfully and return to the login page. |
| Alternative sequences: | 4. After 2 hours using the system, the user logs out automatically. |
| Nonfunctional requirements: | **Security**: System requires user to logout and enter the credentials again after 2 hours using the system.  **Performance**: Logout successfully within less than 5 seconds. |
| Postcondition: | The user is successfully logged out from the system. |
| Outstanding questions: | N/A |



### 4.4 View Patient Daily Record

| UC ID and Name: | UC-04: View Patient Daily Report |
| --- | --- |
| Summary | System allows the Doctor/Nurse to view the information and medication of the patient. |
| Dependency: | N/A |
| Actors: | Doctor, Nurse |
| Preconditions | 1.The user successfully logs into the system with a doctor or nurse account.  2.There is at least 1 patient in the database of the system. |
| Main sequence: | 1.Home page is shown and the Doctor/Nurse can access the system's features and functions according to their account .  2.Doctor/Nurse access the “Patient list” page by clicking on the “Patient List button” in the home page to view the list of patients.  3.Doctor/Nurse choose a patient in the list and access the record of the patient in the list .  4.Doctor/Nurse choose the daily report section.  5.Doctor/Nurse successfully views daily reports of the chosen patient. |
| Alternative sequences: | 5. If the report of the patient is null. The system prompts a message to inform the Doctor/Nurse to write a report for a new patient. |
| Nonfunctional requirements: | N/A |
| Postcondition: | The user successfully views the daily report of the chosen patient. |
| Outstanding questions: | N/A |

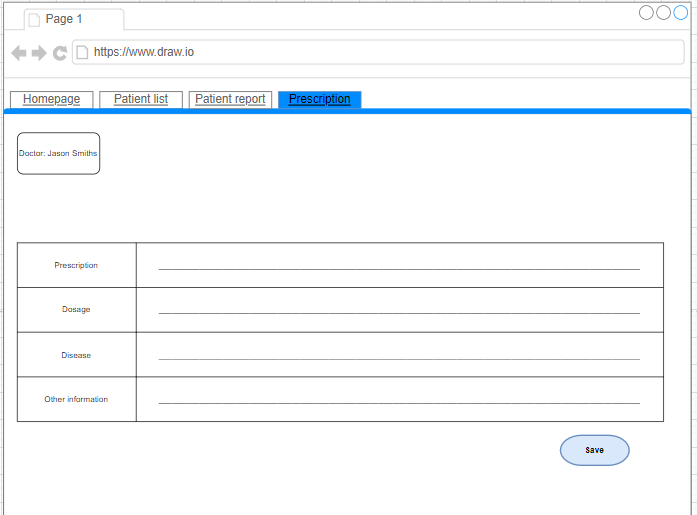
#### 

### 

### 

#### 

| UC ID and Name: | UC-05: Prescribe Medicine |
| --- | --- |
| Summary | System allows a doctor to write a medical prescription for a patient. |
| Dependency: | N/A |
| Actors: | Doctor |
| Preconditions | The user successfully logs into the system with a doctor account. |
| Main sequence: | 1. Doctor accesses the User Login screen.  2. Doctor logins successfully.  3. Home page is shown and the Doctor can access the system's features and functions.  4. Doctor access PatientList page from Homepage.  5. Doctor chooses a patient to view the patient record.  6. Doctor views patient medicine from the patient record.  7. Doctor chooses a medicine from a list of medicines and adds new medicine for the patient.  8. New medicine is added to the patient successfully. |
| Alternative sequences: | 6.1 There is no available medicine in storage.  7.1 Send notification “no available medicine” to the doctor.  6.2 the medicine added to patient is duplicate  7.2 Send notification “this medicine is already assigned to the patient” to the doctor. |
| Nonfunctional requirements: | N/A |
| Postcondition: | Patient’s medical prescription is updated successfully. |
| Outstanding questions: | N/A |



### 4.6 Edit Patient Information

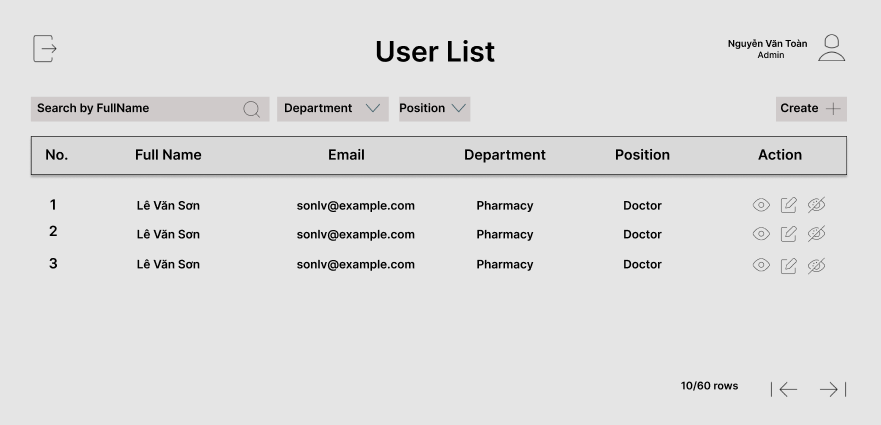
#### 

| UC ID and Name: | UC-06: Edit Patient Information |
| --- | --- |
| Summary | This use case describes the process by which a doctor or nurse wants to update patient information. |
| Dependency: | N/A |
| Actors: | Doctor, Nurse |
| Preconditions | The user successfully logs into the system with a doctor or nurse account. |
| Main sequence: | 1. 1.Home page is shown and the Doctor/Nurse can access the system's features and functions according to their account . 2. 2.Doctor/Nurse access the “Patient list” page to view the list of patients. 3. 3. Doctor/Nurse access patient record of the patient in the list. 4. 4.Doctor/Nurse successfully view patient information and medication in patient record. 5. 5.Doctor/Nurse access Patient Information page. 6. 6.Doctor/Nurse enter the new information of the patient to update. 7. 7.System updates the information and returns to the Patient Report page. |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | System updates the patient’s information within less than 3 seconds. |
| Postcondition: | N/A |

#### 

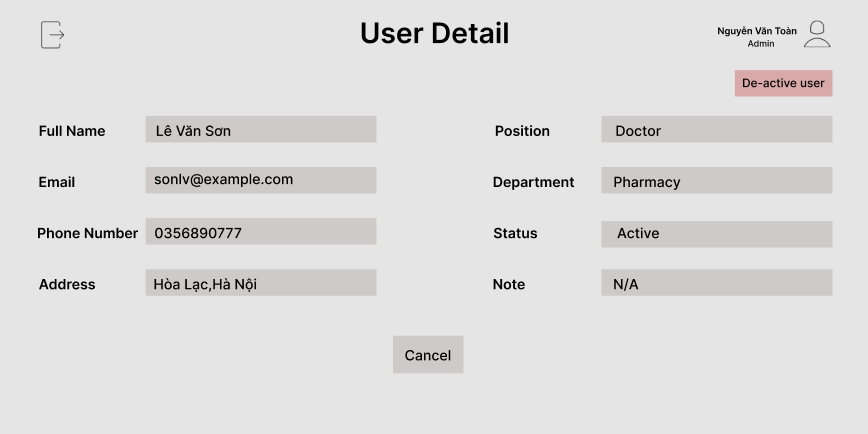
### 4.7 View User List

| UC ID and Name: | UC-07: View User List |
| --- | --- |
| Summary | This use case helps the admin view the list of users in the system. |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | Admin has logged in |
| Main sequence: | 1.Admin login into the system successfully.  2.User clicks the “User Management” menu.  3.System shows user list. |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | **Performance**: List users will be displayed after 3 seconds |
| Postcondition: | Admin can view the list of users. |
| Outstanding questions: | N/A |



### 4.8 Active/Deactive User

| UC ID and Name: | UC-08: Active/Deactive User |
| --- | --- |
| Summary | This use case allows admin to Activate/Deactivate user. |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | User has logged in as Admin |
| Main sequence: | 1.Admin login into the system successfully.  2.User clicks the “User Management” menu.  3.System shows user list.  4.User click on ban/unban-visibility button  5.User click icon Active/Deactivate button.  6.System sets user’s status to Active/Deactivate |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | **Performance**: status of user will updated immediately |
| Postcondition: | User account is activated/deactivated status |
| Outstanding questions: | N/A |



### 4.9 Edit User

| UC ID and Name: | UC-09: Edit User |
| --- | --- |
| Summary | This use case allows admin to edit user |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | User has logged in as Admin. |
| Main sequence: | 1.Admin login into the system successfully.  2.User clicks the “User Management” menu.  3.System shows user list and Admin choose user from list.  4.User click the “Edit” icon in the user list.  5.System shows Edit user information fields to screen.  6.User change information and click submit.  7.System save the changes. |
| Alternative sequences: | Step 6: Users fill invalid information into fields ,an error message is displayed, and the user is prompted to re-enter their information.System cannot return new information to Admin |
| Nonfunctional requirements: | **Performance**: information of the user will update and return userlist in 3 seconds. |
| Postcondition: | User account is updated |
| Outstanding questions: | N/A |



### 4.10 Schedule Appointments

| UC ID and Name: | UC-10: Schedule Appointments |
| --- | --- |
| Summary | This use case allows create appointments for patients with specific doctors or medical departments. |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | User has logged in as Admin |
| Main sequence: | 1.Admin login into the system successfully.  2.Admin clicks “Schedule Appointments”.in homepage  3.System shows schedule form.  4.Admin fills in time and appointment information.  5.Admin clicks the “Create” button.  6.The system will create an appointment between the doctor and the patient, and the doctor will also receive a notification containing that appointment information. |
| Alternative sequences: | Step 4: User fills input date which is in the past,an error message is displayed, and the user is prompted to re-enter their information. |
| Nonfunctional requirements: | **Performance**: System can continuous create 5 appointments for 5 doctors less or equal 1 minute |
| Postcondition: | An appointment schedule between doctor and patient is created |
| Outstanding questions: | N/A |

#### 

### 4.11 Room Allocation

| UC ID and Name: | UC-11: Room Allocation |
| --- | --- |
| Summary | This use case allows the Receptionist booking bed for patients with inpatient needs at the hospital. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | The user must have a valid account in the system.  The system or application is running and accessible. |
| Main sequence: | 1.Receptionist login into the system successfully.  2.User clicks the “Booking Room” in homepage.  3.User choose a room.  4.System required to enter information of patient and choose bed for patient.  5.User click save button.  6.System save the changes and navigate to room list. |
| Alternative sequences | Step 4: User chooses a room but the bed is full ,so an error message is displayed, and the user is prompted to choose another room. |
| Nonfunctional requirements: | **Performance**: The system can continuously book beds for 100 patients in 1 minute. |
| Postcondition: | The patient was successfully booked a bed. |
| Outstanding questions: | N/A |

#### 

### 4.12 Patient registration

#### 

| UC ID and Name: | UC-12: Patient registration |
| --- | --- |
| Summary | Receptionist adding a patient with that person's personal information into the list of patient in the system |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist * The number of patients in the system must not exceed the maximum allowed number |
| Main sequence: | 1. Receptionist login into website 2. Receptionist go to the Patient Registration site. 3. System show patient information form 4. Receptionist fills in patient information 5. Receptionist select create options 6. The system adds 1 patient with the same information entered as above to the list of patients and adds it to the system. |
| Alternative sequences: | Step 6: The system checks that the number of patients in the system has exceeded the allowed number, the screen displays a notification that the hospital has reached the maximum number of patients allowed and no more patients are allowed. |
| Nonfunctional requirements: | Performance: Check and add patients to the system in less than 10 seconds |
| Postcondition: | 1 patient will be added to the system |
| Outstanding questions: | N/A |

#### 

### 4.13 Patient Discharge

#### 

| UC ID and Name: | UC-13: Patient Discharge |
| --- | --- |
| Summary | Receptionist remove patient from their bed and change status to discharge |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist * There is at least 1 patient in the system |
| Main sequence: | 1. Receptionist go to patient list page 2. Receptionist select discharge options 3. Receptionist select option yes 4. The system will change this patient status to discharge in the list of patients. |
| Alternative sequences: | Step 3: Receptionist select option no  4. Back to homepage |
| Nonfunctional requirements: | Performance: Remove patients to the list in less than 10 seconds |
| Postcondition: | 1 patient will be removed from the patient list |
| Outstanding questions: | N/A |

#### 

### 4.14 View Receipt List

#### 

| UC ID and Name: | UC-14: View Receipt List |
| --- | --- |
| Summary | This use case describes the process of a receptionist viewing a list of receipts in the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | The user has logged into the system as a receptionist. |
| Main sequence: | 1. The receptionist accesses the receptionist page. 2. The system displays the receptionist page. 3. The receptionist clicks on the "Receipt List". 4. The system displays a list of receipts, including information such as receipt number, customer name, amount, and date created. |
| Alternative sequences: | Step 4: If there are no receipts in the system, the system displays a message that there are no receipts to display. |
| Nonfunctional requirements: | N/A |
| Postcondition: | The receipt list is displayed to the receptionist. |
| Outstanding questions: | N/A |

#### 

#### 

### 4.15 Create Receipt

#### 

| UC ID and Name: | UC-15: Create Receipt |
| --- | --- |
| Summary | The receptionist creates a receipt and adds the receipt to the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | The user has logged into the system as a receptionist. |
| Main sequence: | 1. The receptionist accesses the receptionist page. 2. The system displays the receptionist page. 3. The receptionist clicks on the "Receipt" item. 4. The system displays the receipt list. 5. The receptionist clicks on the "Add Receipt" option. 6. The system displays a form to enter receipt information. 7. The receptionist fills in the required information to submit form (e.g., services used, amount, date) 8. The system displays the receipt just created. |
| Alternative sequences: | 1. Step 7 : Receptionist fills in the form with receipt information. 2. Before submitting the form, the receptionist decides to cancel the creation process. 3. Receptionist clicks on the "Cancel" button. 4. The system prompts for confirmation to cancel the receipt creation. 5. Receptionist confirms the cancellation. 6. The system discards the form and returns to the receipt list without creating a new receipt. |
| Nonfunctional requirements: | N/A |
| Postcondition: | If the receipt is created successfully, the recipient will appear in View Receipt List. |
| Outstanding questions: | N/A |

### 4.16 Edit Receipt

#### 

| UC ID and Name: | UC-16: Edit Receipt |
| --- | --- |
| Summary | The receptionist edit a receipt and update the receipt to the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | The user has logged into the system as a receptionist. |
| Main sequence: | 1. The receptionist accesses the receptionist page. 2. The system displays the receptionist page. 3. The receptionist go to receipt page 4. The system displays the receipt list. 5. The receptionist choose edit receipt options 6. The system displays a form to edit receipt information. 7. The receptionist fills in the required information to submit form (e.g., services used, amount, date) 8. The system displays the receipt just created. |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | N/A |
| Postcondition: | If the receipt is created successfully, the recipient will appear in View Receipt List. |
| Outstanding questions: | N/A |

### 

### 4.17 Delete Receipt

#### 

| UC ID and Name: | UC-17: Delete Receipt |
| --- | --- |
| Summary | Receptionist delete a receipt in the system |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist |
| Main sequence: | 1. include UC 14 view receipt list 2. Receptionist clicks “Delete” beside the receipt 3. System show give notice if you are sure you want to delete it 4. Receptionist choose “yes” 5. This receipt will be removed from the system 6. Reload the receipt list |
| Alternative sequences: | Step 3: If the receptionist choose “no”, screen will back to list receipt screen |
| Nonfunctional requirements: | Performance: Delete receipt and reload less than 8 seconds |
| Postcondition: | A receipt is removed from system |
| Outstanding questions: | N/A |

### 4.18 Request Tests and Imaging

#### 

| UC ID and Name: | UC-18: Request Tests and Imaging |
| --- | --- |
| Summary | This use case describes the process by which a doctor requests tests and imaging for a patient. |
| Dependency: | N/A |
| Actors: | Doctor |
| Preconditions | User has logged in as Doctor |
| Main sequence: | 1. The doctor selects a patient from the list. 2. The system displays the patient's detailed information. 3. The doctor clicks on the option to request tests and imaging. 4. The system displays a form for requesting tests and imaging. 5. The doctor fills out the form with the necessary details for the tests and imaging. 6. The doctor submits the request. 7. The system saves the request and updates the patient's records. |
| Alternative sequences: | Step 1.1.a: If the doctor selects a patient from the list, the system will attempt to display the patient's details. However, if the system does not find the patient's record, it will display an error message stating that the patient's record does not exist. |
| Nonfunctional requirements: | NA |
| Postcondition: | **Patient Records Update:** Patient information, including details of tests and imaging performed, must be updated and accurately stored in the patient records system. |
| Outstanding questions: | N/A |

#### 

### 4.19 Add New User

#### 

| UC ID and Name: | UC-19: Add New User |
| --- | --- |
| Summary | This use case allows admin to create new user |
| Dependency: | **N/A** |
| Actors: | Admin |
| Preconditions | User has logged in as Admin |
| Main sequence: | 1.Admin login in the system successfully  2.User clicks “User Management” menu  3.System shows userlist  4.User click icon “Add” in the User list  5.System show Create user screen  6.User enter information and click submit.  7.System create new user account and send email to user |
| Alternative sequences | Step 6:User invalid information into fields ,an error message is displayed, and the user is prompted to re-enter their information |
| Nonfunctional requirements: | **Performance**: The system can continuously created 100 patients in 1 minute |
| Postcondition: | New account is created |
| Outstanding questions: | N/A |

#### 

#### 

#### 

#### 

#### 

#### 

#### 

### 4.20 Authenticate User

#### 

| UC ID and Name: | UC-20: Authenticate User |
| --- | --- |
| Summary | This use case describes the process system authenticate user so that defined the role is using system |
| Dependency: |  |
| Actors: | Doctor,Nurse,Admin,Receptionist |
| Preconditions | User have to login into an account that already exist in system |
| Main sequence: | 1.The user initiates the authentication process (e.g., clicks on a "Login" button).  2.The system prompts the user to enter their credentials.  3.The user enters their username/email and password.  4.The system validates the provided credentials against the stored user data.  5.If the credentials are valid, the system authenticates the user and grants access to authorized features and functionalities that fit with the role of the account.  6.The system Send-Redirect User to the Home Page |
| Alternative sequences: | 5.The Credential are invalid,system tell user that username/password is wrong and let user to stay in login window to re-input login credentials |
| Nonfunctional requirements: | User passwords must be stored as a secure hash instead of in plain text. |
| Postcondition: | User must login with an account have role Doctor or Nurse |
| Outstanding questions: |  |

#### 

### 4.21 View Patient List

#### 

| UC ID and Name: | UC-21: View Patient List |
| --- | --- |
| Summary | This use case describes the process of user view the Patient List that available in system |
| Dependency: |  |
| Actors: | Doctor,Nurse |
| Preconditions | There is already patient record exist in database  and user been authenticate by logging the system |
| Main sequence: | 1. The user accesses the login page.  2. The system displays the login form.  3. The user enters their username and password.  4. The system validates the user's credentials.  5. If the credentials are valid, the system logs the user in and grants access.  6. Once logged in,user will be redirect to the homepage the user can access the system's features and functions  7. User click on Patient List  8. System send redirect user to the Patient List tab and get all patient form a list |
| Alternative sequences: |  |
| Nonfunctional requirements: | * Only authorized medical personnel have access to patient lists.   Patient data must be encrypted and protected from unauthorized access.   * Patient List should be sorted by first character of patient name |
| Postcondition: | * All patient that exist in database should be display in the list ,each information should be visible in a row is * Name * Gender * Age * Status (That the patient is discharged or is located in a room) * Record ( To click and view record of that patient) * Report ( To click and view report of that patient) |
| Outstanding questions: | N/A |

#### 

### 4.22 Search Patient

#### 

| UC ID and Name: | UC-22: Search Patient |
| --- | --- |
| Summary | This use case describes the process by which users want to view a specific patient by using searching bar |
| Dependency: |  |
| Actors: | Doctor,Nurse |
| Preconditions | User |
| Main sequence: | 1. The user accesses the login page.  2. The system displays the login form.  3. The user enters their username and password.  4. The system validates the user's credentials.  5. If the credentials are valid, the system logs the user in and grants access.  6. Once logged in,user will be redirect to the homepage the user can access the system's features and functions  7. User click on Patient List  8. System send redirect user to the Patient List tab and get all patient form a list  9. User input a specific patient name into search bar  10.System return patient with name that contain all character in the search bar |
| Alternative sequences: | 8. There is no name that contain all character in the search bar so system return no result |
| Nonfunctional requirements: | Search response times must be fast and results accurate |
| Postcondition: | User must login with an account have role Doctor or Nurse |
| Outstanding questions: | N/A |
| Business rule: | * System should return all patient that name contain all the character user input in search bar * User Can search patient with patient name,patient phone, patient email |

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

### 4.23 View Patient Record

#### 

| UC ID and Name: | UC-23: View Patient Record |
| --- | --- |
| Summary | This use case describes the process by which users want to view a list of patient record |
| Dependency: |  |
| Actors: | Doctor,Nurse |
| Preconditions | There is already patient record exist in database |
| Main sequence: | 1. The user accesses the login page.  2. The system displays the login form.  3. The user enters their username and password.  4. The system validates the user's credentials.  5. If the credentials are valid, the system logs the user in and grants access.  6. Once logged in,user will be redirect to the homepage the user can access the system's features and functions  7. User click on Patient List  8. System send redirect user to the Patient List tab and get all patient form a list  9. Find patient want to view record list and click “view record”  10. System send-redirect user to that patient record list  11. View Patient Record |
| Alternative sequences: |  |
| Nonfunctional requirements: | All record must be sorted by date from newest to oldest |
| Postcondition: | User can see all record of patient that available in database |
| Outstanding  questions: |  |

#### 

**Business Rule :**   
Br1 : System should return all the record of the patient that are current being choose to view record, all record should be sort from newest to oldest

#### 

#### 

### 4.24 View Patient Record

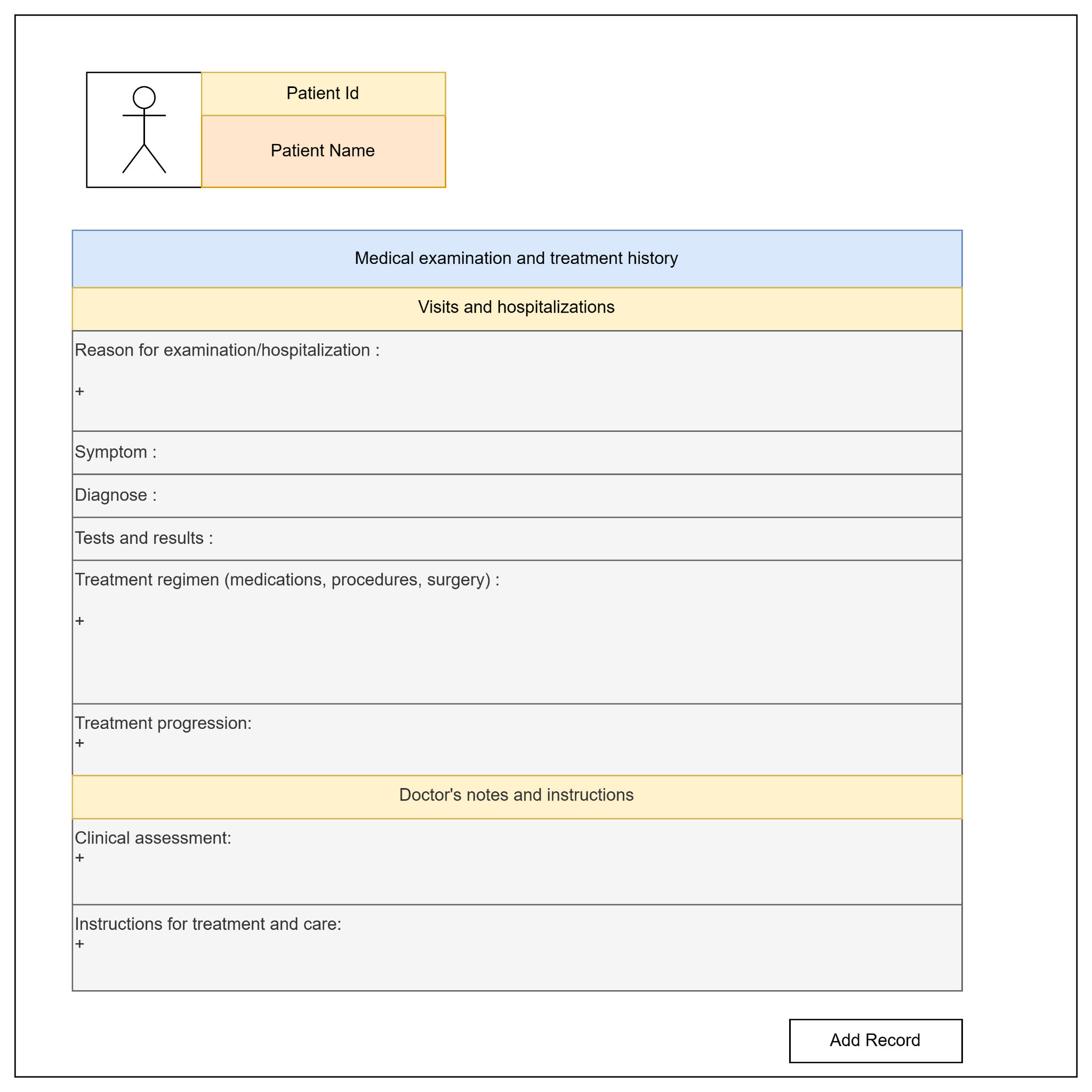
#### 

| UC ID and Name: | UC-24: View Patient Record |
| --- | --- |
| Summary | This use case describes the process by which users want to view a patient record that already exist in patient record list |
| Dependency: |  |
| Actors: | Doctor,Nurse |
| Preconditions | There is already patient record exist in patient record list |
| Main sequence: | 1. The user accesses the login page.  2. The system displays the login form.  3. The user enters their username and password.  4. The system validates the user's credentials.  5. If the credentials are valid, the system logs the user in and grants access.  6. Once logged in,user will be redirect to the homepage the user can access the system's features and functions  7. User click on Patient List  8. System send redirect user to the Patient List tab and get all patient form a list  9. User click “View Record” of a patient that need to see record list  10. System send redirect user to “Patient Record List”  11. Click on a Record  12. View Patient record |
| Alternative sequences: | 7.There is no record of a patient yet |
| Nonfunctional requirements: |  |
| Postcondition: | User must login with an account have role Doctor or Nurse |

#### 

### 4.25 Add Patient Record

| UC ID and Name: | UC-25: Add Patient Record |
| --- | --- |
| Summary | This use case describes the process by which users want to view a patient record that already exist in database |
| Dependency: |  |
| Actors: | Doctor,Nurse |
| Preconditions | When there is new information that need new record |
| Main sequence: | 1. The user accesses the login page.  2. The system displays the login form.  3. The user enters their username and password.  4. The system validates the user's credentials.  5. If the credentials are valid, the system logs the user in and grants access.  6. Once logged in,user will be redirect to the homepage the user can access the system's features and functions  7. User click on Patient List  8. System send redirect user to the Patient List tab and get all patient form a list  9. User click “View Record” of a patient that need to see record list  10. System send redirect user to “Patient Record List”  11.Click Add record  12.System will send redirect user to the add record tab  13.User input new record information  14.Click “Add Record”  15.System Will validate medicine info  16.System redirect user to patient record list and display new patient record |
| Alternative sequences: | 16. System cannot find medicine info  17. system return add fail message to user |
| Nonfunctional requirements: | All information should be save to database correctly and the same |
| Postcondition: | A new record will appear in database and system will display it in record list of that patient |
| Outstanding questions: |  |



### 4.26 Update Patient Record

#### 

| UC ID and Name: | UC-26: Update Patient Record |
| --- | --- |
| Summary | This use case describes the process by which users update a patient record that already exist in database |
| Dependency: | Include: UC-24 : View Patient Record |
| Actors: | Doctor,Nurse |
| Preconditions | There is already patient record exist in database that need to update |
| Main sequence: | 1. The user accesses the login page.  2. The system displays the login form.  3. The user enters their username and password.  4. The system validates the user's credentials.  5. If the credentials are valid, the system logs the user in and grants access.  6. Once logged in,user will be redirect to the homepage the user can access the system's features and functions  7. User click on Patient List  8. System send redirect user to the Patient List tab and get all patient form a list  9. User click “View Record” of a patient that need to see record list  10. System send redirect user to “Patient Record List”  11. Click on a Record  12.User click update button next to the record user want to update  13.System send redirect user to update record`s tab  14.User update patient record information and click “Update Record”  15.System update patient record to the latest information |
| Alternative sequences: | Step 14. User click back button so the update information will not change |
| Nonfunctional requirements: | The information should be update correctly and replace with the old information |
| Postcondition: | -Patient record should be change and update successfully into database  -User must login with an account have role Doctor or Nurse |
| Outstanding questions: |  |

#### 

### 4.27 View user details

| UC ID and Name: | UC-27: View user details |
| --- | --- |
| Summary | This use case allows admin to view user details page |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | User has logged in as Admin |
| Main sequence: | 1.Include UC 7 View Userlist  2.User click icon “View” button in the User list  3.System display user details screen for user to view |
| Alternative sequences: | Step 2: If the system cannot display data, an error message will be displayed. |
| Nonfunctional requirements: | **Performance**: The system can load all information of user in 1 second |
| Postcondition: | Admin can view user details page |
| Outstanding questions: | N/A |

#### 

### 4.28 View List Role

| UC ID and Name: | UC-28: View List Role |
| --- | --- |
| Summary | This use case describes the process by which an admin views the list of roles in the system. |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | The user has logged into the system as an admin. |
| Main sequence: | 1. The admin accesses the admin dashboard. 2. The system displays the admin dashboard. 3. The admin clicks on the "View Roles" option. 4. The system displays a list of all roles. |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | N/A |
| Postcondition: | The admin is able to view the list of roles in the system. |
| Outstanding questions: | N/A |

### 4.29 Create Role

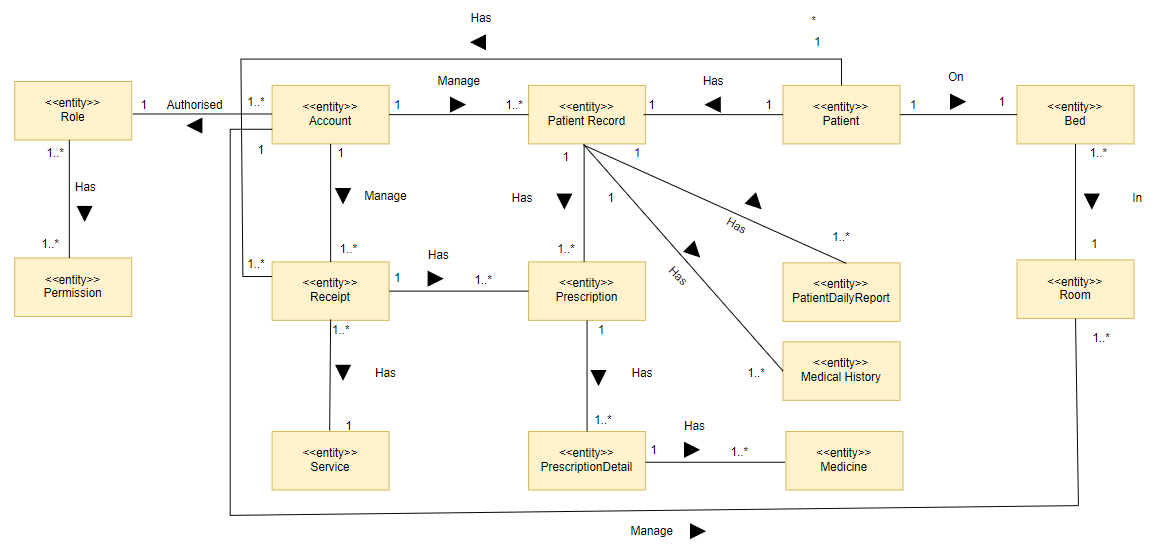
| UC ID and Name: | UC-29: Create Role |
| --- | --- |
| Summary | This use case describes the process by which an admin creates a new role in the system. |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | The user has logged into the system as an admin. |
| Main sequence: | 1. The admin clicks on the "Create Role" option. 2. The system displays a form to enter role information. 3. The admin fills in the required information (e.g., role name, permissions) and submits the form. 4. The system validates the information and creates the new role. 5. The system creates new roles successfully. |
| Alternative sequences: | Step 4.1a: If the admin cancels creating a new role while the role is being created, the system will display a confirmation message and display a list of current roles. |
| Nonfunctional requirements: | N/A |
| Postcondition: | The new role is created and available in the system. |
| Outstanding questions: | N/A |

### 4.30 Update Role

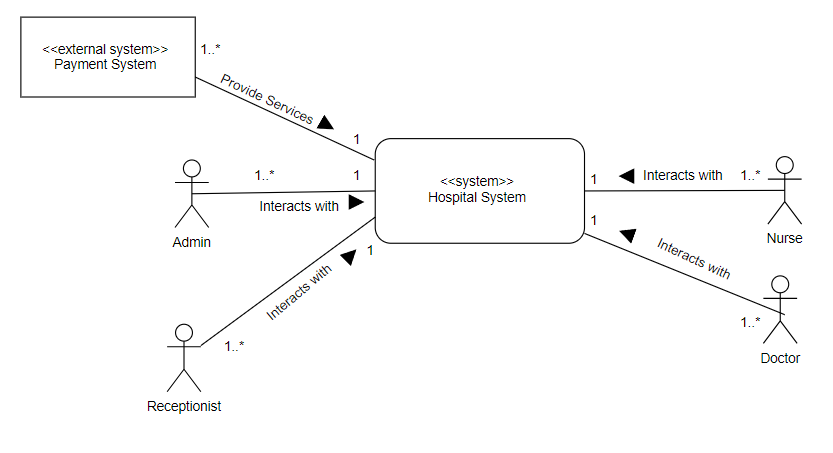
| UC ID and Name: | UC-30: Update Role |
| --- | --- |
| Summary | This use case describes the process by which an admin updates the permissions of an existing role in the system, including granting access to specific pages based on the role. |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | The user has logged into the system as an admin. |
| Main sequence: | 1. The admin clicks on the "Update Roles" option. 2. The system displays a form with the current role permissions. 3. The admin updates the permissions, including granting access to specific pages, and submits the form. 4. The system validates the updated permissions and saves the changes. 5. The system updates the role permissions to grant access to the specified pages. 6. The system displays "Update roles successfully." |
| Alternative sequences: | Step 2.a1.1.1: If the user clicks the "Update Roles" option and the system displays a form representing game permissions, but the user goes about 5 minutes without updating the role, the system logs the user in and displays a message that the current session has expired. The user need make a Edit Role |
| Nonfunctional requirements: | N/A |
| Postcondition: | The role is updated with the new information in the system, including the updated permissions for accessing specific pages. |
| Outstanding questions: | N/A |

# II. Analysis

## 1.Entity Diagram



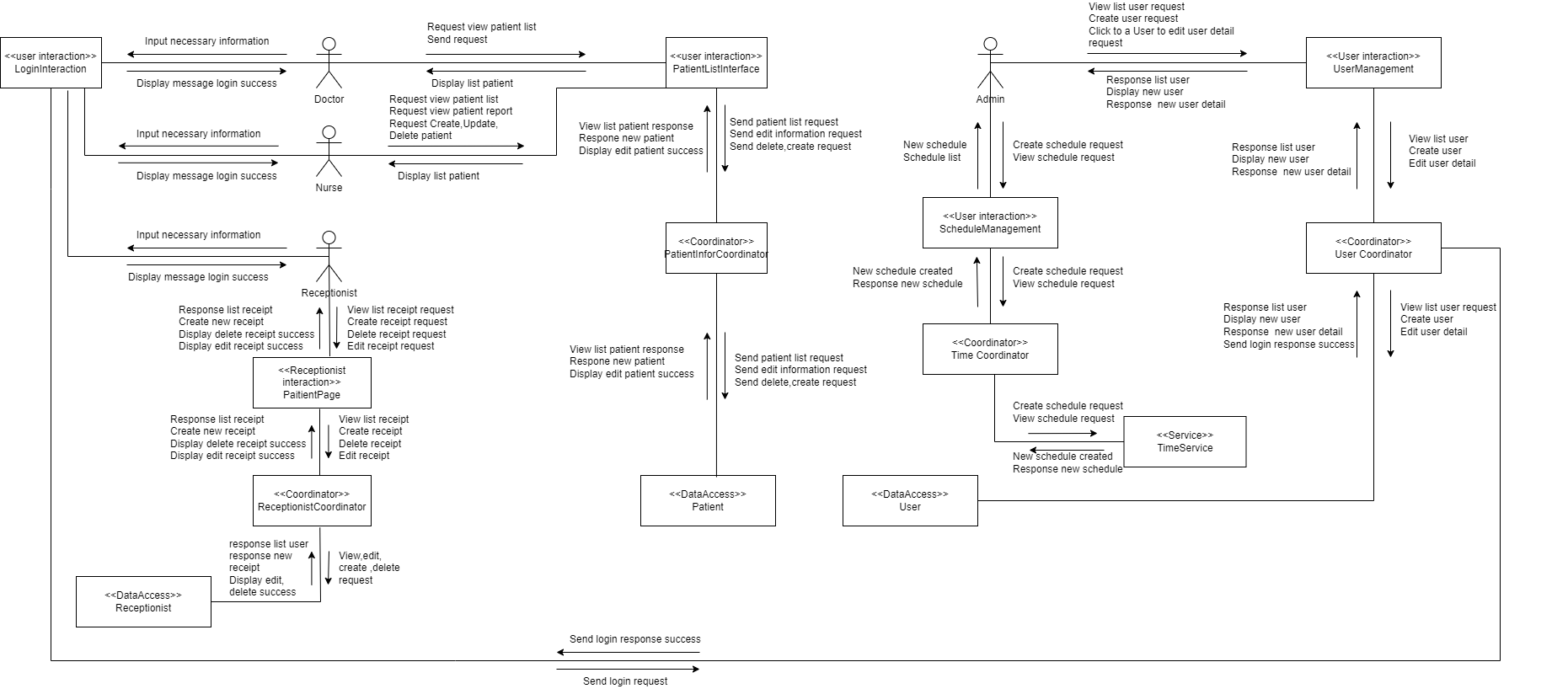
## 2.System Context Diagram



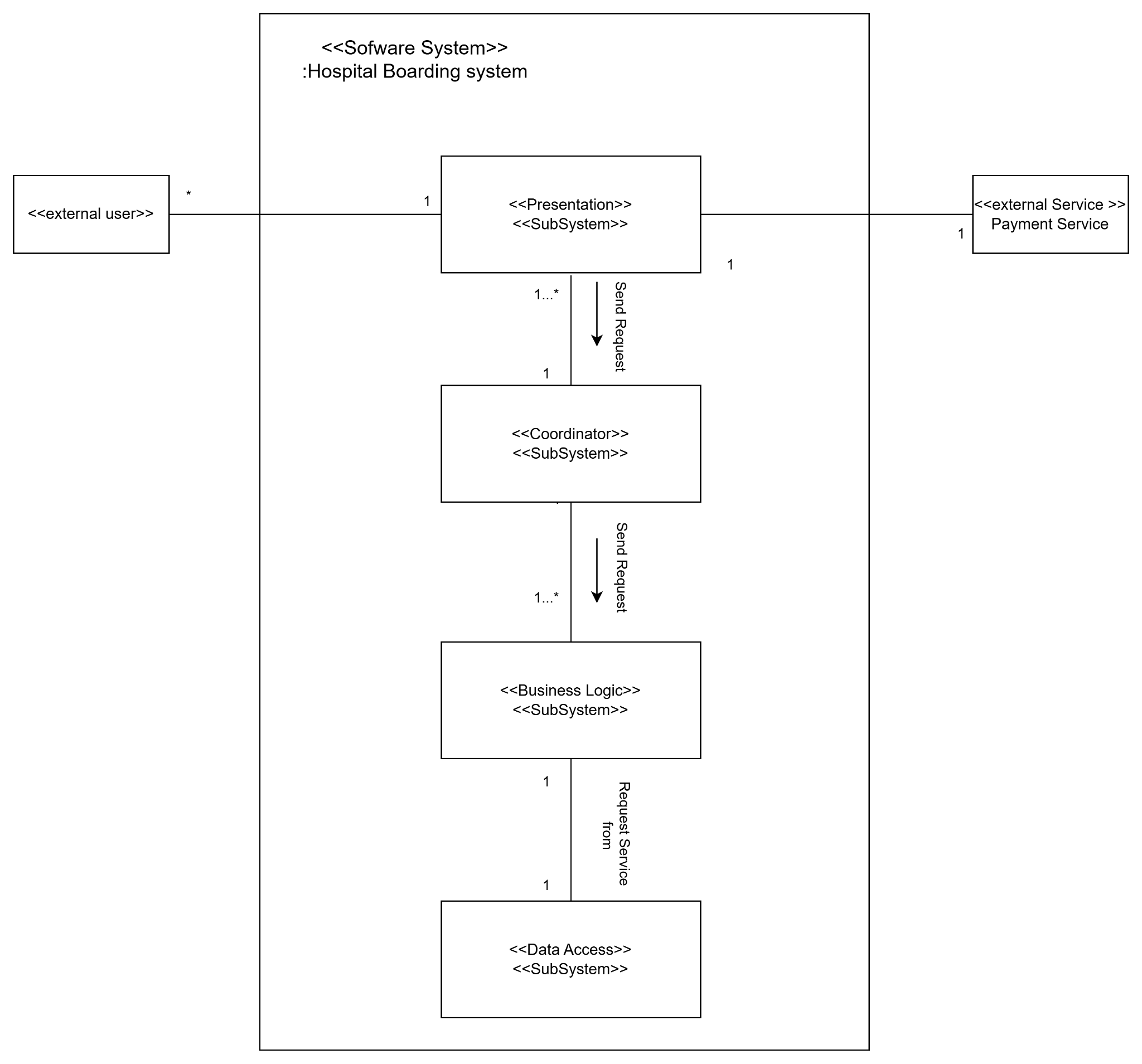
## 3.Boundary Class



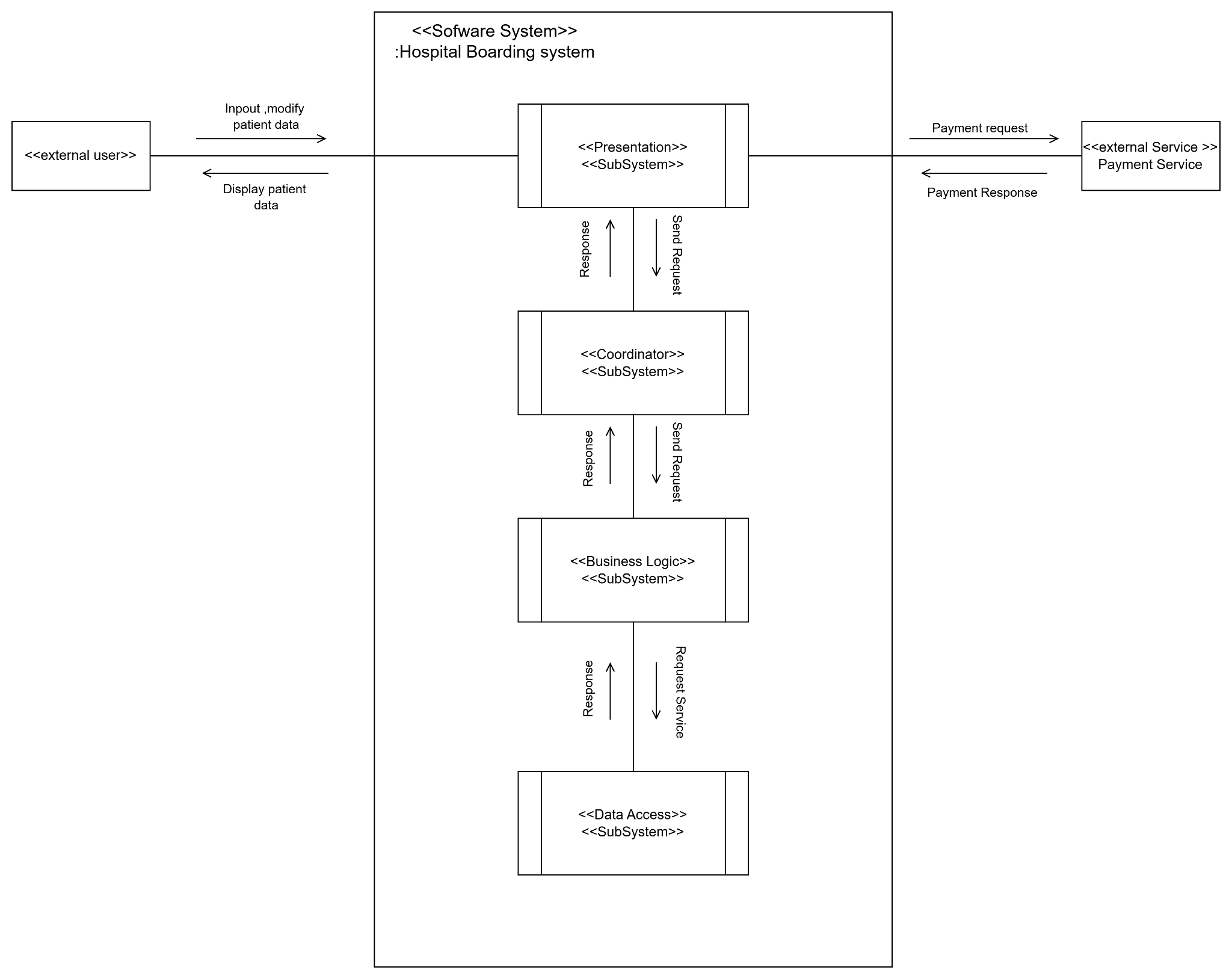
## 4.Integrated Communication Diagram



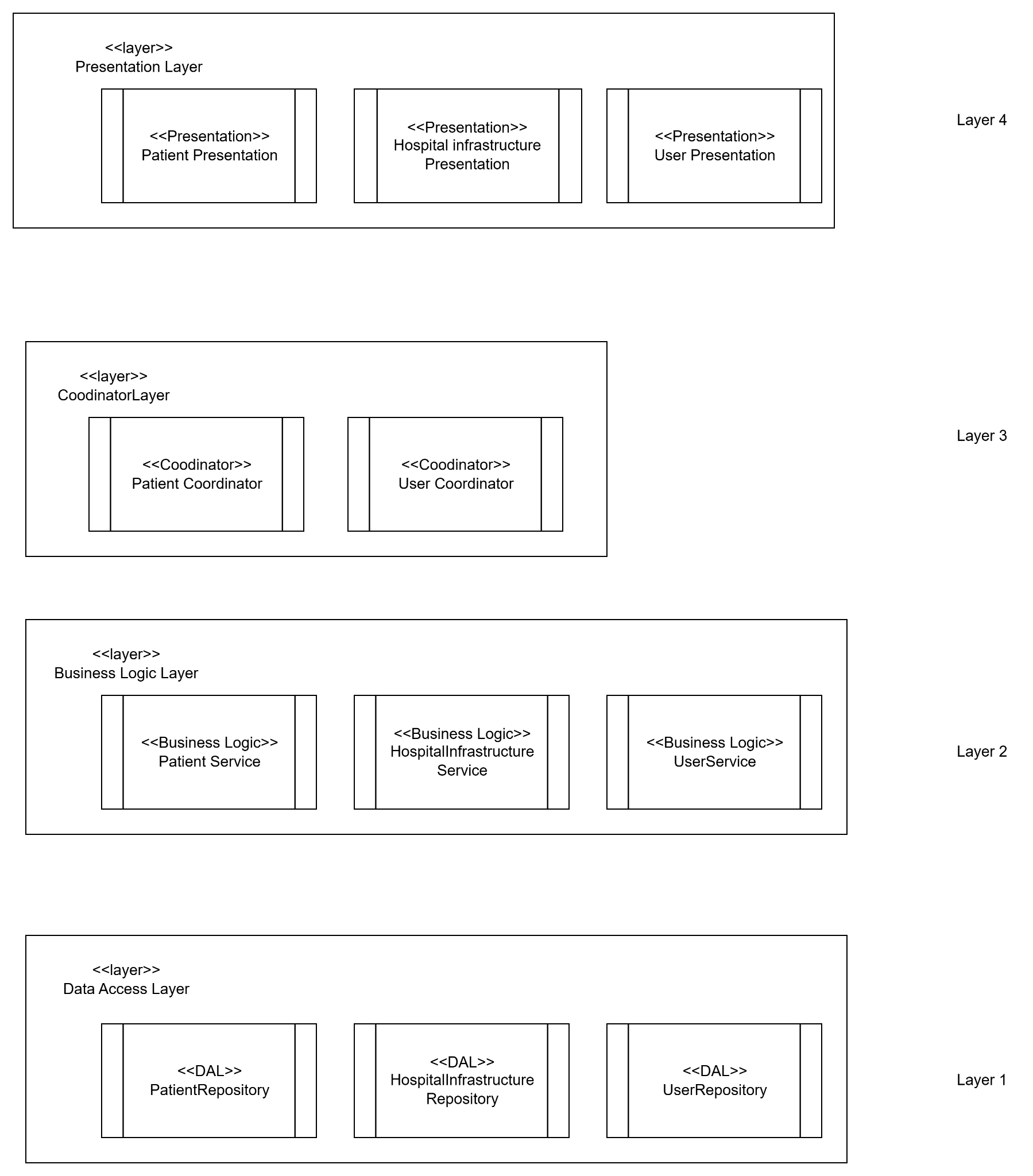
## 5. Structural View



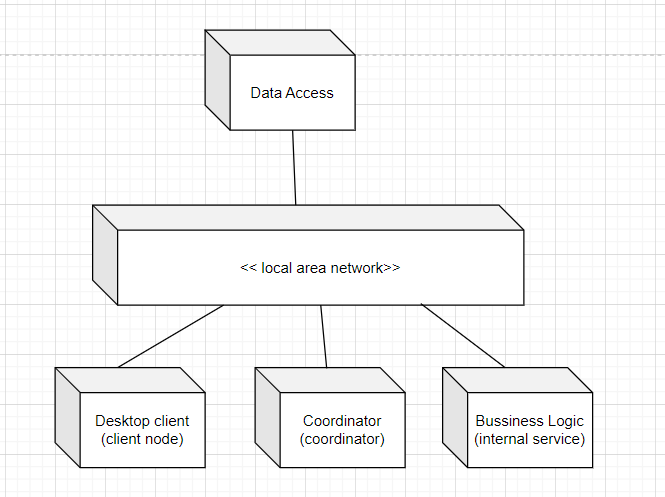
## 6.Dynamic View

****

## 7.Layer Of Abstraction



## 8. Deployment View



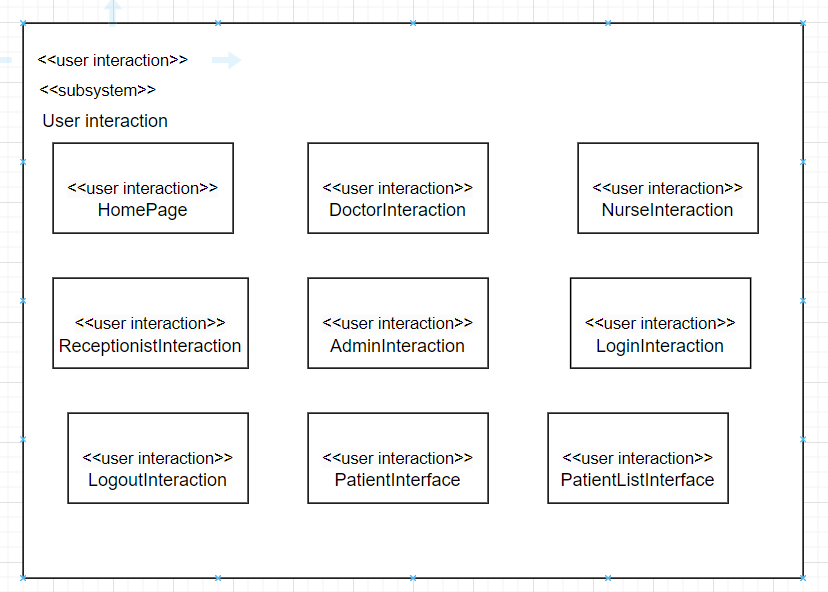
## 9. Class Diagram

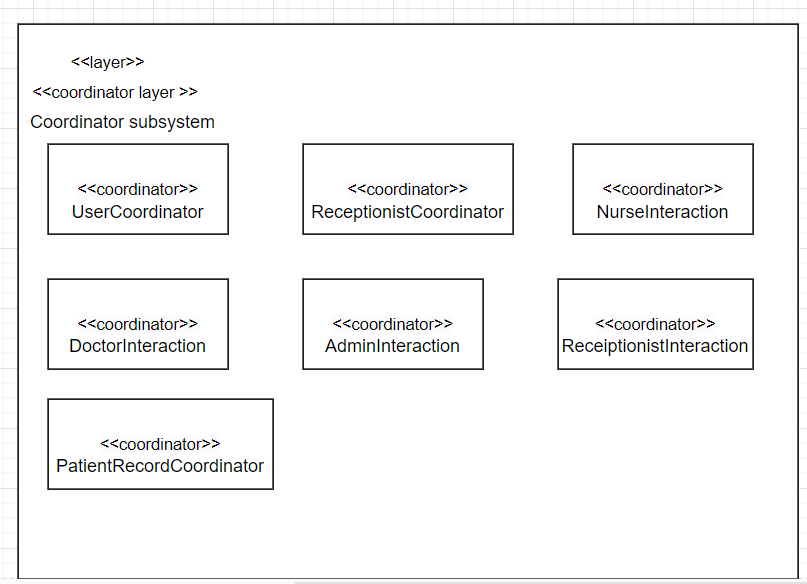
## 

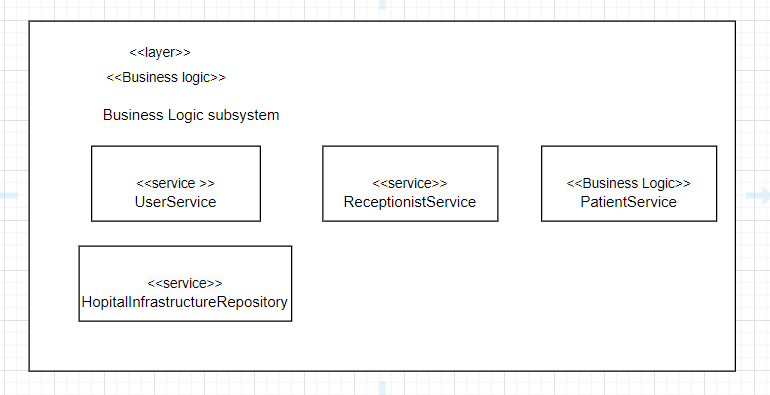
## 

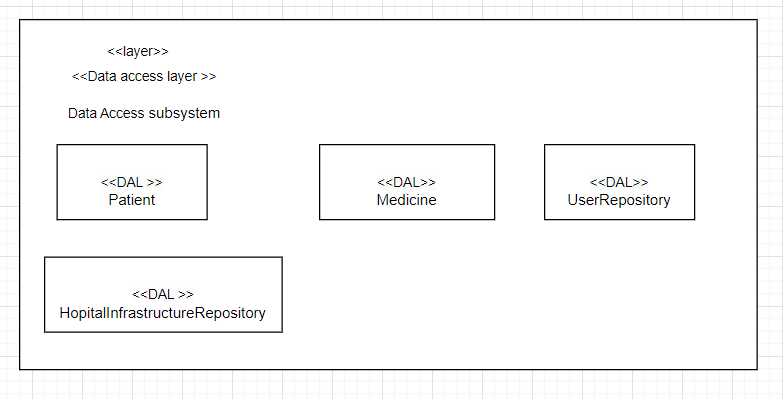
## 10. SubSystem

SUBSYSTEM









## 

## 11. Individual Part

### A. Vũ Đình Khiêm( He170306 )

### **1. Requirement**

4.1 [Login](#_kskjf9x3wud7)

4.2 [View Home Page](#_jadrkpagntd8)

4.3 [Logout](#_sgnkrcowgntu)

4.4 [View Patient Report](#_yzouv6h6b8f2)

4.5 [Prescribe Medicine](#_qmr8to1mwa9r)

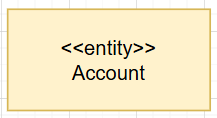
4.6 [Edit Patient Information](#_ykwu39vmm0kz)

### **2. Static Modeling**

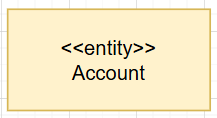
Entity Class

##### 

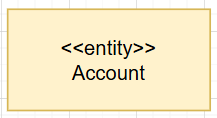
##### 2.1.1 Login



##### 2.1.2 View HomePage

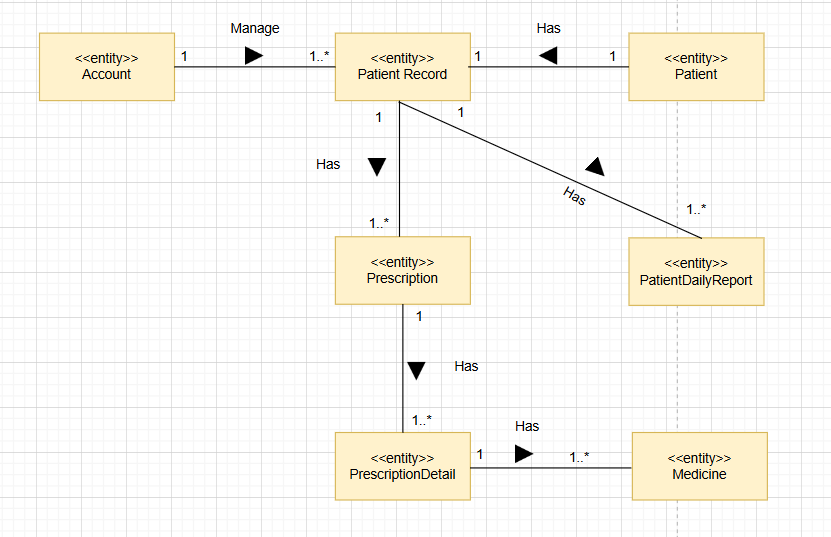


##### 2.1.3 Logout



##### 2.1.4 View PatientDailyReport

##### 



##### 2.1.5 Prescribe medicine

#### 

##### 2.1.6 Edit Patient Information

#### 

### **3. Dynamic Modeling**

### 

##### 3.1 Login

### 

| UC ID and Name: | UC-01: Login |
| --- | --- |
| Summary | Users use their username and password to access the Hospital management system. |
| Dependency: | N/A |
| Actors: | Admin, Doctor, Nurse, Receptionist |
| Preconditions | The user must have a valid account in the system. |
| Main sequence: | 1.The user accesses the login page.  2.The UserCoordinators display the login form.  3.The user enters their username and password.  4.The UserService validates the user's credentials.  5.If the credentials are valid, the UserCoordinators log the user in and grant access.  6.Once logged in, the user can access the system's features and functions. |
| Postcondition: | The user is successfully logged into the hospital  management system. |

**Identify boundaries and internal objects**

-Boundary object: LoginInteraction

-Internal object: UserCoordinators

-Service: UserService

**Messages**

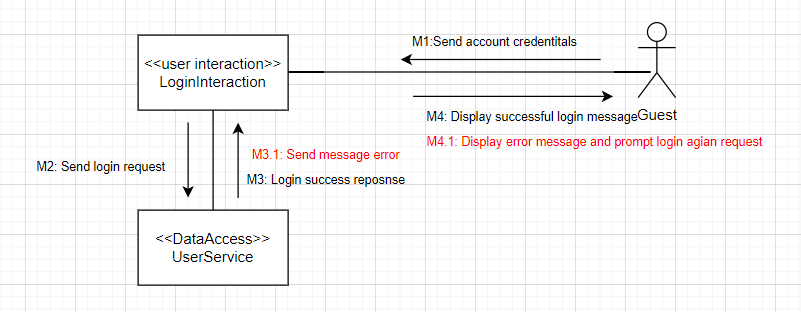
M1: Guest sends request to LoginInteraction

M2: LoginInteraction passes the login request to UserCoordinators for processing.

M3: User Coordinators send authentication requests for login information to the User

M4: Users Service validate information then send login success response to UserCoordinatiors

M5: UserCoordinators send response login success to LoginInteraction

M6: LoginInteraction show message login success 

**Alternative: [wrong information]**

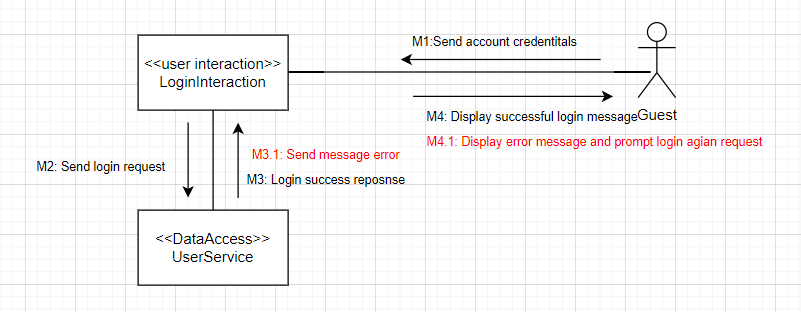
| UC ID and Name: | UC-01: Login |
| --- | --- |
| Summary | Users use their username and password to access the Hospital management system. |
| Dependency: | N/A |
| Actors: | Admin, Doctor, Nurse, Receptionist |
| Preconditions | The user must have a valid account in the system. |
| Alternative sequences: | 5a. UserService checked that the credentials are invalid,  6a. an error message is sent to UserCoordinators , and the user is prompted to re-enter their information. |
| Postcondition: | The user is successfully logged into the hospital  management system. |

**Messages**

M4A.1: User Service sends response validation failed to UserCoordinators.

M5A.2: UserCoordinators send message error to LoginInteraction.

M6A.3: LoginInteraction show [wrong input information] error message and prompt request to enter credential again.



##### 

##### 3.2 View Home Page

| UC ID and Name: | UC-02: View Home Page |
| --- | --- |
| Summary | The home page displays room management, discharge process, patient report, bed charges and provides links to different sections of the site. Users can easily navigate to different parts of the website from the home page. |
| Actors: | Admin, Doctor, Nurse, Receptionist |
| Preconditions | The user must have a valid account in the system. |
| Main sequence: | 1. Include login use case.  2. Home page is shown and users can access the system's features and functions.  3. The home page displays room management, discharge process, patient report, bed charges options for users to access. |
| Postcondition: | The user gets access to features in the home page. |

**Identify boundaries and internal objects**

-Boundary object: HomePageInteraction

-Internal object: UserService

**Messages**

**M1:** User sends a request to HomepageInteraction.

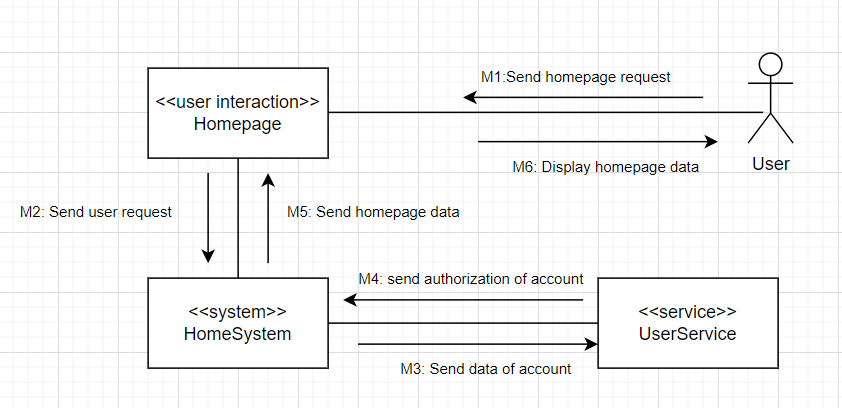
**M2:** HomepageInteraction sends a request to the HomeSystem to load the homepage.

**M3:** The HomeSystem sends data of the account to UserService

**M4:** User Service sends authorization of account to HomeSystem.

**M5:** HomeSystem sends homepage data to HomepageInteraction.

**M6:** HomepageInteraction displays the homepage to the customer.



##### 3.3 Logout

#### 

| UC ID and Name: | UC-03: Logout |
| --- | --- |
| Summary | Users log out from the system |
| Dependency: | N/A |
| Actors: | Admin, Doctor, Nurse, Receptionist. |
| Preconditions | Users already login to the system. |
| Main sequence: | 1. User accesses the User Login screen.  2. User login successfully.  3. Home page is shown and users can access the system's features and functions.  4. User clicks on the " log out” symbol.  5. System prompts a message to ask the user to confirm.  6. User logout successfully and return to the login page. |
| Alternative sequences: | 4. After 2 hours using the system, the user logs out automatically. |
| Nonfunctional requirements: | **Security**: System requires user to logout and enter the credentials again after 2 hours using the system.  **Performance**: Logout successfully within less than 5 seconds. |
| Postcondition: | The user is successfully logged out from the system. |
| Outstanding questions: | N/A |

**Identify boundaries and internal objects**

-Boundary object: LogoutInteraction

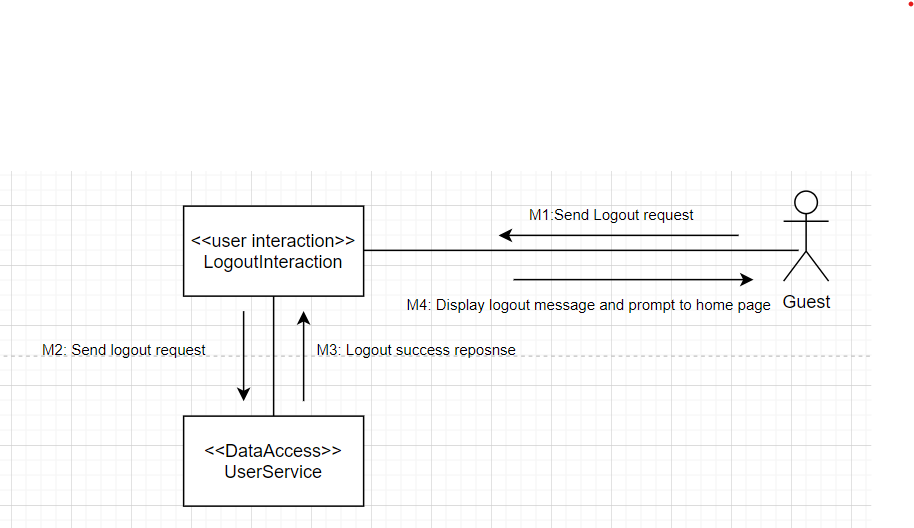
-Internal object: UserService

**Messages**

**M1:** User sends a request to LogoutInteraction.

**M2:** LogoutInteraction sends a request to the UserService

**M3:** The UserService sends a logout request.

**M4:** LogoutInteraction display logout message prompts user to homepage.

##### 3.4 View Patient Daily Report

### 

| UC ID and Name: | UC-04: View Patient Daily Report |
| --- | --- |
| Summary | System allows the Doctor/Nurse to view the information and medication of the patient. |
| Actors: | Doctor, Nurse |
| Preconditions | 1.The user successfully logs into the system with a doctor or nurse account.  2.There is at least 1 patient in the database of the system. |
| Main sequence: | 1.Home page is shown and the Doctor/Nurse can access the system's features and functions according to their account .  2.Doctor/Nurse access the “Patient list” page by clicking on the “Patient List button” in the home page to view the list of patients.  3.Doctor/Nurse choose a patient in the list and access the record of the patient in the list .  4.Doctor/Nurse choose the daily report section.  5.Doctor/Nurse successfully views daily reports of the chosen patient. |
| Postcondition: | The user successfully views the daily report of the chosen patient. |
| Outstanding questions: | N/A |

**Identify boundaries and internal objects**

-Boundary object: PatientInterface

-Internal object: PatientRecordCoordinator

**Messages**

M1: Select the patient from list

M2: Request patient information from list

M3: request medicine information

M4: response medicine information

M5: request patient information

M6: return patient information

M7: send patient record response

M8: shows patient record response

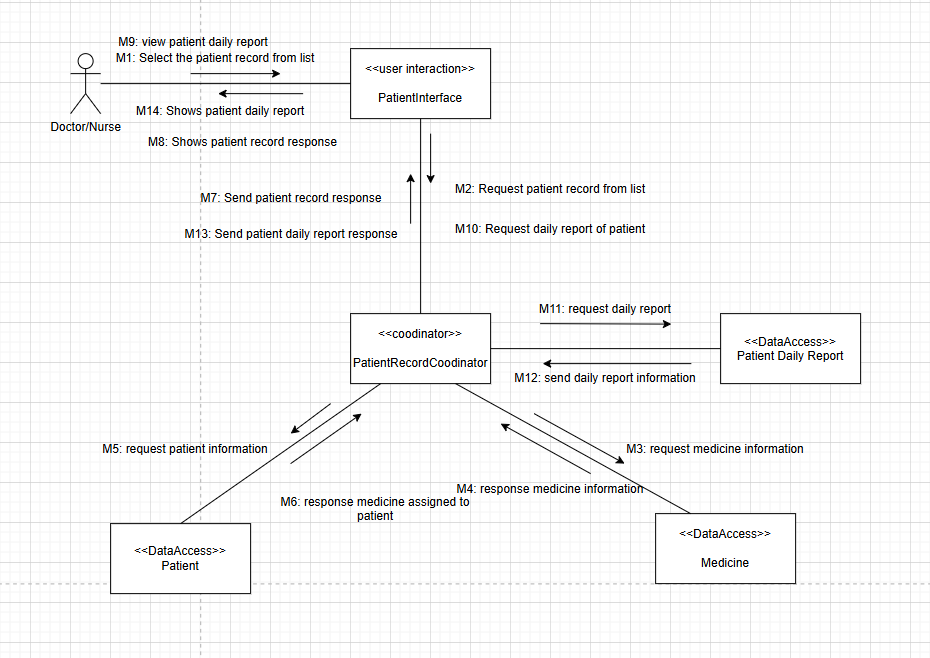
M9:View patient daily report

M10:request daily report of patient

M11:request daily report

M12:send daily report information

M13: Send patient daily report response

M14: Shows patient daily report

**Alternative: [wrong information]**

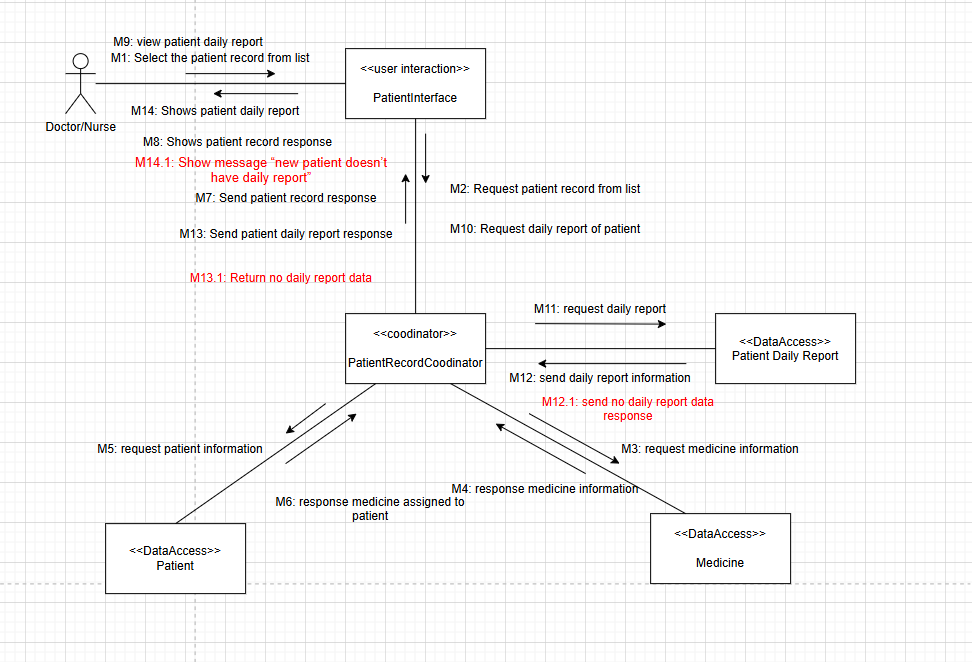
| UC ID and Name: | UC-04: View Patient Daily Report |
| --- | --- |
| Summary | System allows the Doctor/Nurse to view the information and medication of the patient. |
| Actors: | Doctor, Nurse |
| Preconditions | 1.The user successfully logs into the system with a doctor or nurse account.  2.There is at least 1 patient in the database of the system. |
| Alternative sequences: | 5a. there is no daily report of the new patient  6a. Send notification “new patient doesn’t have daily report” to the doctor. |

**Messages**

M12.1: send no daily report data response

M13.1: Return no daily report data

M14.1: Show message “new patient doesn’t have daily report”



##### 3.5 Prescribe Medicine

#### 

| UC ID and Name: | UC-05: Prescribe Medicine |
| --- | --- |
| Summary | System allows a doctor to prescribe medicine for a patient. |
| Actors: | Doctor |
| Preconditions | The user successfully logs into the system with a doctor account. |
| Main sequence: | 1. Home page is shown and the Doctor can access the system's features and functions.  2. Doctor access PatientList page from Homepage.  3. Doctor chooses a patient to view the patient record.  4. Doctor views patient information and medicine from the patient record.  5. Doctor chooses a medicine from a list of medicines and adds new medicine for the patient.  6. New medicine is added to patients successfully. |
| Postcondition: | Patient’s medical prescription is updated successfully. |

**Identify boundaries and internal objects**

-Boundary object: PatientInterface

-Internal object: PatientRecordCoordinator

**Messages**

M1: Select the patient record from list

M2: Request patient record of patient

M3: request medicine of patient

M4: return medicine

M5: request patient information

M6: return patient information

M7: send patient record response

M8: shows patient record response

M9: assign medicine to patient

M10: request assign medicine to patient

M11: request new medicine assigned to patient

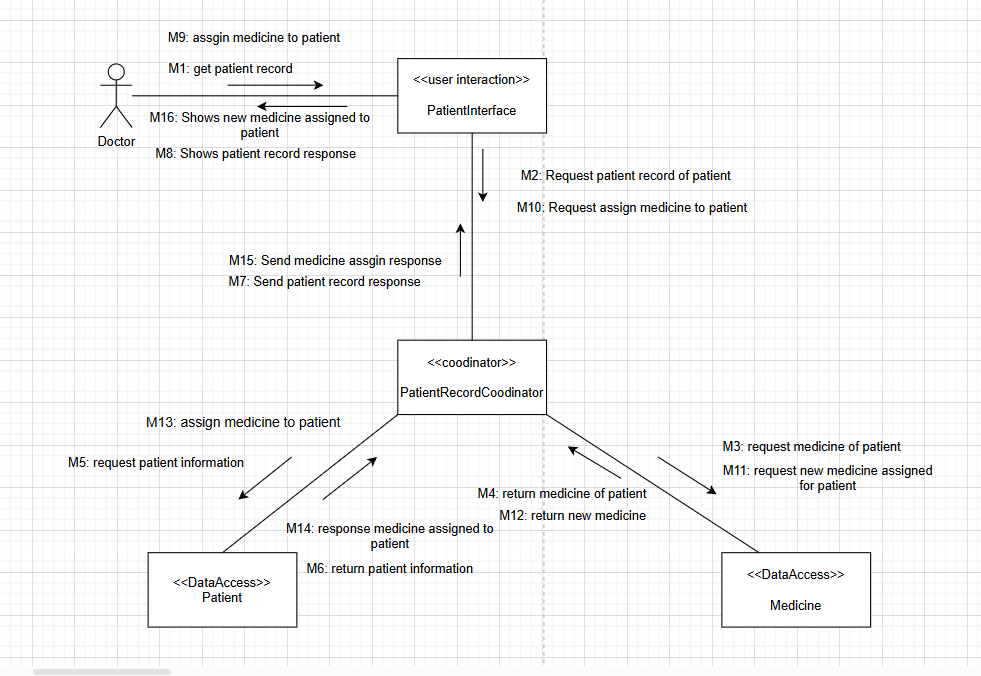
M12: return medicine

M13: assign medicine to patient

M14: response medicine assigned to patient

M15: Send medicine assign response

M16: Shows new medicine assigned to patient



**Alternative: [wrong information]**

#### 

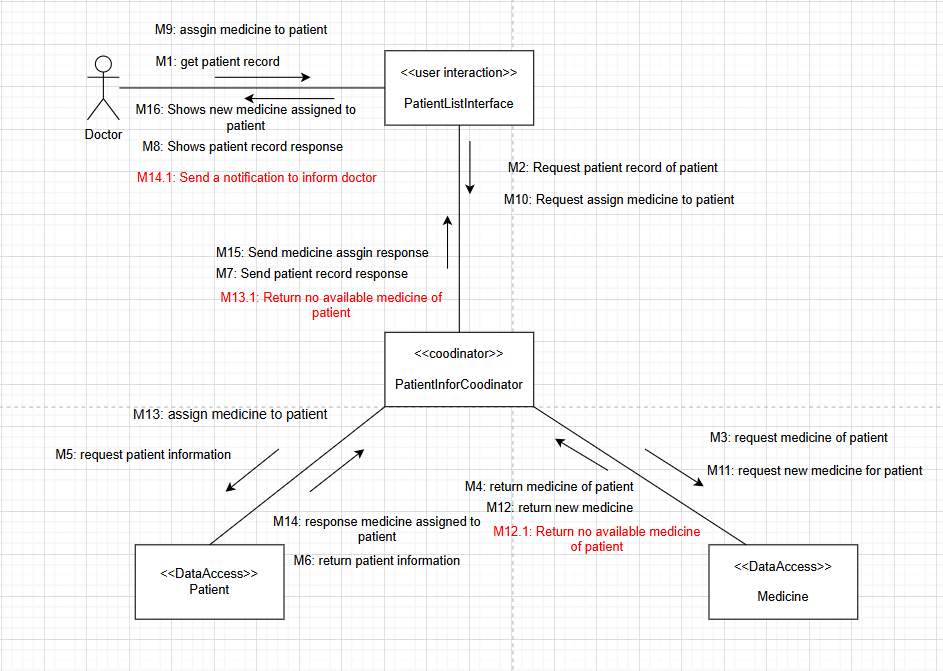
| UC ID and Name: | UC-05: Prescribe Medicine |
| --- | --- |
| Summary | System allows a doctor to write a medical prescription for a patient. |
| Dependency: | N/A |
| Actors: | Doctor |
| Preconditions | The user successfully logs into the system with a doctor account. |
| Alternative sequences: | 6.1 There is no available medicine in storage.  7.1 Send notification “no available medicine” to the doctor. |
| Postcondition: | Patient’s medical prescription is updated successfully. |

**Messages**

**M12.1: Return no available medicine of patient**

**M13.1: Return no available medicine of patient**

**M14.1: Send a notification to inform doctor**



##### 3.6 Edit Patient Information

#### 

| UC ID and Name: | UC-06: Edit Patient Information |
| --- | --- |
| Summary | This use case describes the process by which a doctor or nurse wants to update patient information. |
| Actors: | Doctor |
| Preconditions | The user successfully logs into the system with a doctor or nurse account. |
| Main sequence: | 1. Home page is shown and the Doctor/Nurse can access the system's features and functions according to their account.  2. Doctor/Nurse access the “Patient list” page to view the list of patients.  3. Doctor/Nurse access patient record of the patient in the list.  4. Doctor/Nurse successfully view patient information and medication in patient record.  5. Doctor/Nurse access Patient Information page.  6. Doctor/Nurse enter the new information of the patient to update.  7. System updates the information and returns to the Patient Report page. |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | System updates the patient’s information within less than 3 seconds. |
| Postcondition: | Patient’s information is updated successfully. |
| Outstanding questions: | N/A |

**Identify boundaries and internal objects**

-Boundary object: PatientInterface

-Internal object: PatientRecordCoordinator

**Messages**

M1: Select the patient record from list

M2: Request patient record of patient

M3: request medicine of patient

M4: return medicine

M5: request patient information

M6: return patient information

M7: send patient record response

M8: shows patient record response

M9: update patient information

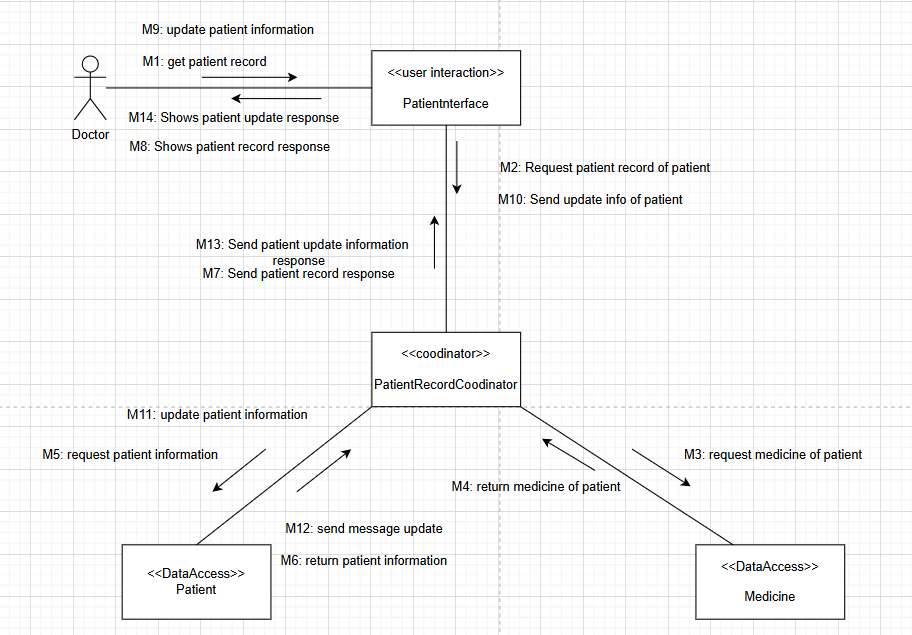
M10:send update info of patient

M11:update patient information

M12:send message update

M13:send patient update information

M14: show patient update response

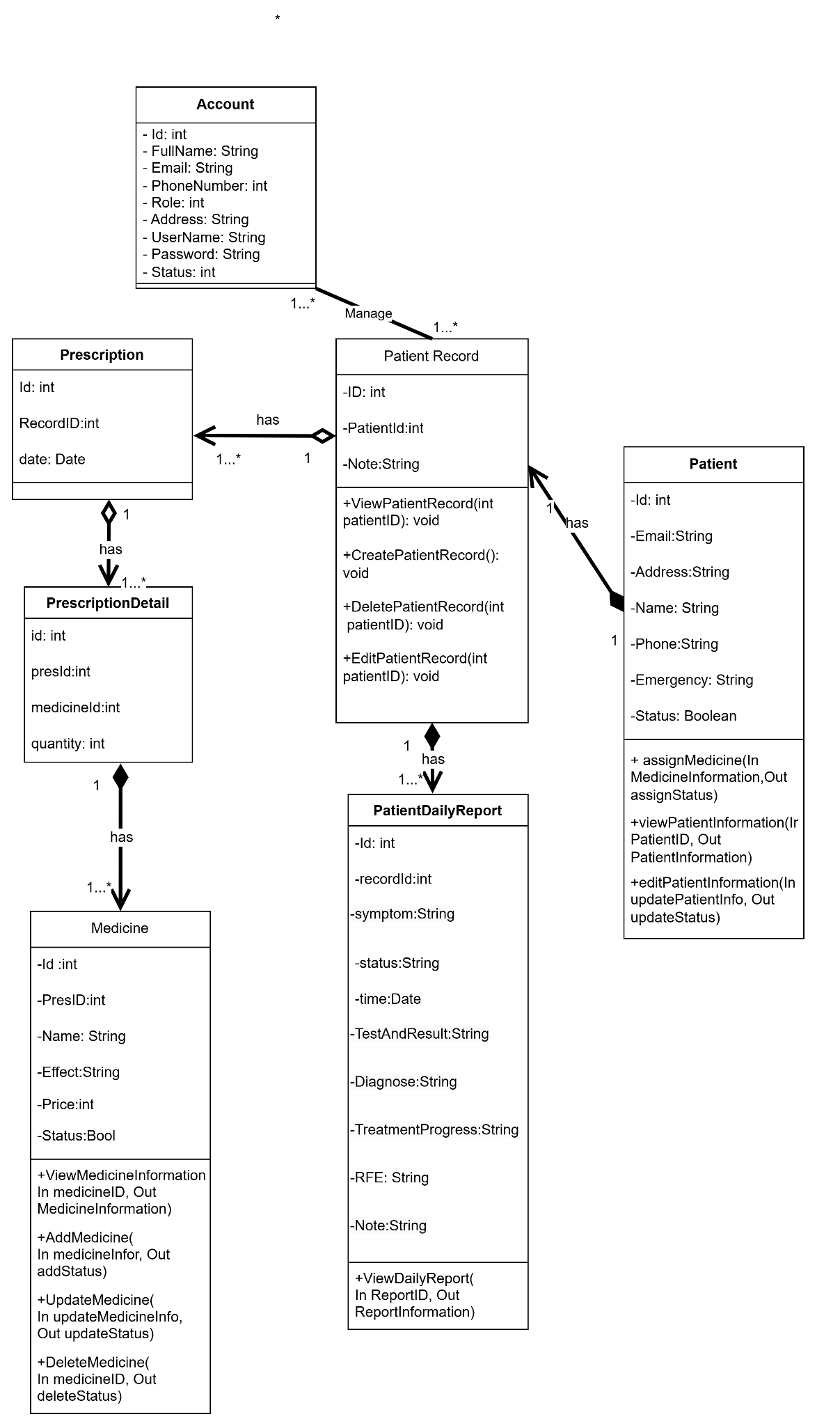


### **4. Architecture design**

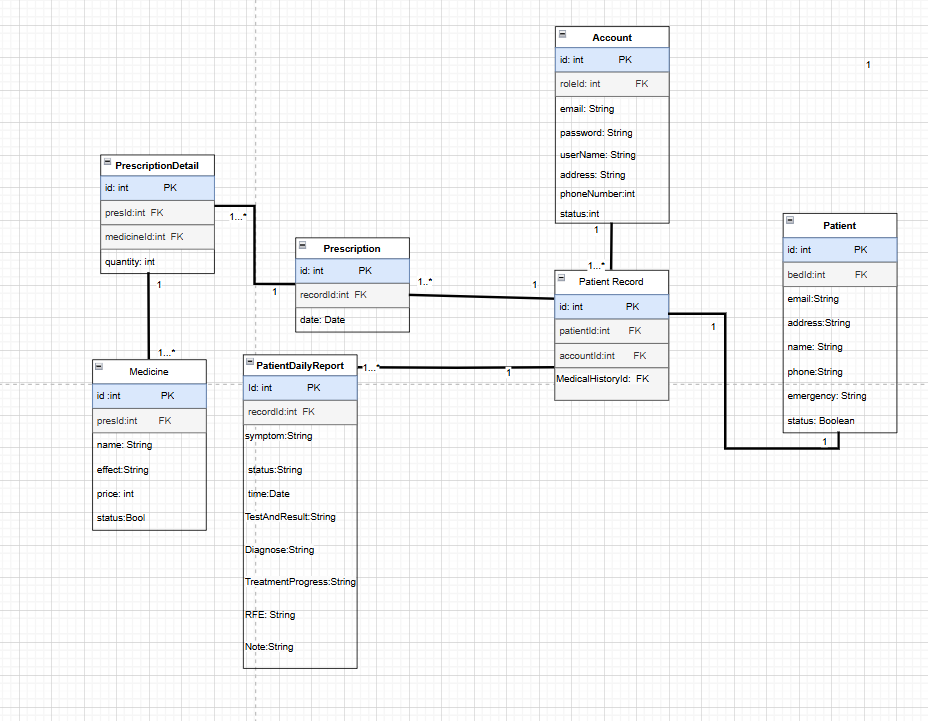
### 

### **5.Detail design**

#### 5.1.Class Diagram



#### 5.2.Relational Database



#### 5.3.Pseudocode

Patient:

* Attribute: id, Address, name, email,address,phone,emergency
* Operation:
* assignMedicine (In MedicineID,Out assignStatus):

**BEGIN**

foundMedicine <- findMedicineById(MedicineID)

foundPatient<- findPatientById(PatientID)

**IF** foundMedicine is NULL **THEN**

PRINT "Medicine not found"

Out assignStatus <- false

**ELSE IF** foundMedicine.status = false **THEN**

PRINT "Medicine not available"

Out assignStatus <- false

**ELSE**

Prescription.addMedicine <- foundMedicine

**THEN**

UpdatePrescription(prescription,foundPatient)

Out assignStatus <- true

**END IF**

**END**

* getPatientInformation (In PatientID,Out PatientInformation):

**BEGIN**

foundPatient <- NULL

**FOR EACH**

patient IN patientList DO

**IF** patient.id == In PatientID **THEN**

foundPatient <- patient

**BREAK**

**END IF**

**END FOR**

**IF** foundPatient == NULL **THEN**

PRINT "Patient not available"

**ELSE**

Out PatientInformation <- foundPatient

**END IF**

**END**

* editPatientInformation (In updatePatientInfo,Out updateStatus):

**BEGIN**

foundPatient <- NULL

**FOR EACH**

patient IN patientList DO

**IF** patient.id == In updatePatient.id **THEN**

foundPatient <- patient

**BREAK**

**END IF**

**END FOR**

**IF** foundPatient == NULL **THEN**

PRINT "Patient not available"

**ELSE**

foundPatient.Information <- updatePatientInfo

**END IF**

**END**

Medicine:

* Attribute: id, Address, name, effect, price, PresID,status
* Operation:
* getMedicineInformation (In MedicineID, Out MedicineInformation):

**BEGIN**

foundMedicine <- NULL

**FOR EACH**

medicine IN medicineList DO

**IF** medicine.id == In MedicineID **THEN**

foundMedicine <- medicine

**BREAK**

**END IF**

**END FOR**

**IF** foundMedicine == NULL **THEN**

PRINT "Medicine not available"

**ELSE**

Out MedicineInformation <- foundMedicine

**END IF**

**END**

* deleteMedicine (In MedicineID, Out deleteStatus):

**BEGIN**

foundMedicine <- NULL

**FOR EACH**

medicine IN medicineList DO

**IF** medicine.id == In MedicineID **THEN**

foundMedicine <- medicine

**BREAK**

**END IF**

**END FOR**

**IF** foundMedicine == NULL **THEN**

PRINT "Medicine not available"

**ELSE**

medicineList.Remove(foundMedicine)

updateMedicineTable()

**END IF**

**END**

* getMedicineFromPatient (In PatientID, Out ListOfMedicine):

**BEGIN**

foundPatient <- NULL

**FOR EACH**

patient IN paptientList DO

**IF** patient.id == In PatientID **THEN**

foundPatient <- patient

**BREAK**

**END IF**

**END FOR**

**IF** foundPatient == NULL **THEN**

PRINT "Patient not available"

**ELSE**

Out MedicineInformation <- foundPatient.ListMedicine()

**END IF**

**END**

* deleteMedicine (In MedicineID, Out deleteStatus):

**BEGIN**

foundMedicine <- NULL

**FOR EACH**

medicine IN medicineList DO

**IF** medicine.id == In MedicineID **THEN**

foundMedicine <- medicine

**BREAK**

**END IF**

**END FOR**

**IF** foundMedicine == NULL **THEN**

PRINT "Medicine not available"

**ELSE**

medicineList.Remove(foundMedicine)

updateMedicineTable()

**END IF**

**END**

* updateMedicine (In updateMedicineInfor, Out deleteStatus):

**BEGIN**

**FOR EACH**

medicine IN medicineList DO

**IF** medicine.id == In MedicineID **THEN**

medicine.Information= updateMedicineInfor

updateMedicineTable()

**BREAK**

**END IF**

**END FOR**

**END**

* addMedicine (In MedicineInformation, Out addStatus):

**BEGIN**

foundMedicine <- NULL

**FOR EACH**

medicine IN medicineList DO

**IF** medicine.information == In MedicineInformation **THEN**

foundMedicine <- medicine

**BREAK**

**END IF**

**END FOR**

**IF** foundMedicine == NULL **THEN**

createNewMedicine(MedicineInformation)

updateMedicineTable()

**ELSE**

PRINT "Medicine duplicate"

**END IF**

Patient Daily Report:

* Attribute: id, fullname, email, phoneNumber, address, userName, password, status
* Operation:
* ViewPatientDailyReport(In ReportId,Out ReportInformation)

**BEGIN**

foundReport <- NULL

**FOR EACH**

patient IN patientList DO

**IF** patient.id == In patientID **THEN**

foundPatient <- patient

**BREAK**

**END IF**

**END FOR**

**IF** foundPatient == NULL **THEN**

PRINT "Daily report not available"

**ELSE**

Out foundPatient.dailyReport<- foundReport

**END IF**

**END**

Account:

* Attribute: id, fullname, email, phoneNumber, address, userName, password, status
* Operation:
* login (In AccountCredentials,Out AccountInformation):

**BEGIN**

foundMedicine <- NULL

**FOR EACH**

medicine IN medicineList DO

**IF** medicine.id == In MedicineID **THEN**

foundMedicine <- medicine

**BREAK**

**END IF**

**END FOR**

**IF** foundMedicine == NULL **THEN**

PRINT "Medicine not available"

**ELSE**

Out MedicineInformation <- foundMedicine

**END IF**

**END**

* logout (In AccountCredentials,Out AccountInformation):

**BEGIN**

foundMedicine <- NULL

**FOR EACH**

medicine IN medicineList DO

**IF** medicine.id == In MedicineID **THEN**

foundMedicine <- medicine

**BREAK**

**END IF**

**END FOR**

**IF** foundMedicine == NULL **THEN**

PRINT "Medicine not available"

**ELSE**

Out MedicineInformation <- foundMedicine

**END IF**

**END**

* ViewHomePage (In Account,Out RoleFeature):

**BEGIN**

foundRole <- NULL

**FOR EACH**

role IN roleList DO

**IF** role.id == In Account.roleID **THEN**

foundRole <- role

**BREAK**

**END IF**

**END FOR**

foundRoleFeature<- NULL

featureOfRole<- role.listFeatureID

**FOR EACH**

featureID in featureOfRole DO

**FOR EACH**

feature IN featureList DO

**IF** feature.id == featureID **THEN**

foundRoleFeature.add(feature)

**BREAK**

**END IF**

**END FOR**

**END FOR**

Out RoleFeature<- foundRoleFeature

**END**

### **6.Change requirement**

#### 6.1 Requirements

#### 

| UC ID and Name: | UC-05: Prescribe Medicine |
| --- | --- |
| Summary | System allows a doctor to write a medical prescription for a patient. |
| Dependency: | N/A |
| Actors: | Doctor |
| Preconditions | The user successfully logs into the system with a doctor account. |
| Main sequence: | 1. Doctor accesses the User Login screen.  2. Doctor logins successfully.  3. Home page is shown and the Doctor can access the system's features and functions.  4. Doctor access PatientList page from Homepage.  5. Doctor chooses a patient to view the patient record.  6. Doctor views patient medicine from the patient record.  7. Doctor chooses a medicine from a list of medicines and adds new medicine for the patient.  8. New medicine is added to the patient successfully. |
| Alternative sequences: | 7.1 There is no available medicine in storage.  8.1 Send notification “no available medicine” to the doctor.  7.2 The medicine added to the patient is duplicate.  8.2 Send notification “this medicine is already assigned to the patient” to the doctor. |
| Nonfunctional requirements: | N/A |
| Postcondition: | Patient’s medical prescription is updated successfully. |
| Outstanding questions: | N/A |

#### 6.2 Static Modeling

Entity Class

#### 

#### 6.3 Dynamic Modeling

#### 

| UC ID and Name: | UC-05: Prescribe Medicine |
| --- | --- |
| Summary | System allows a doctor to prescribe medicine for a patient. |
| Actors: | Doctor |
| Preconditions | The user successfully logs into the system with a doctor account. |
| Main sequence: | 1. Home page is shown and the Doctor can access the system's features and functions.  2. Doctor access PatientList page from Homepage.  3. Doctor chooses a patient to view the patient record.  4. Doctor views patient information and medicine from the patient record.  5. Doctor chooses a medicine from a list of medicines and adds new medicine for the patient.  6. New medicine is added to patients successfully. |
| Postcondition: | Patient’s medical prescription is updated successfully. |

**Identify boundaries and internal objects**

-Boundary object: PatientInterface

-Internal object: PatientRecordCoordinator

**Messages**

M1: Select the patient record from list

M2: Request patient record of patient

M3: request medicine of patient

M4: return medicine

M5: request patient information

M6: return patient information

M7: send patient record response

M8: shows patient record response

M9: assign medicine to patient

M10: request assign medicine to patient

M11: request new medicine assigned to patient

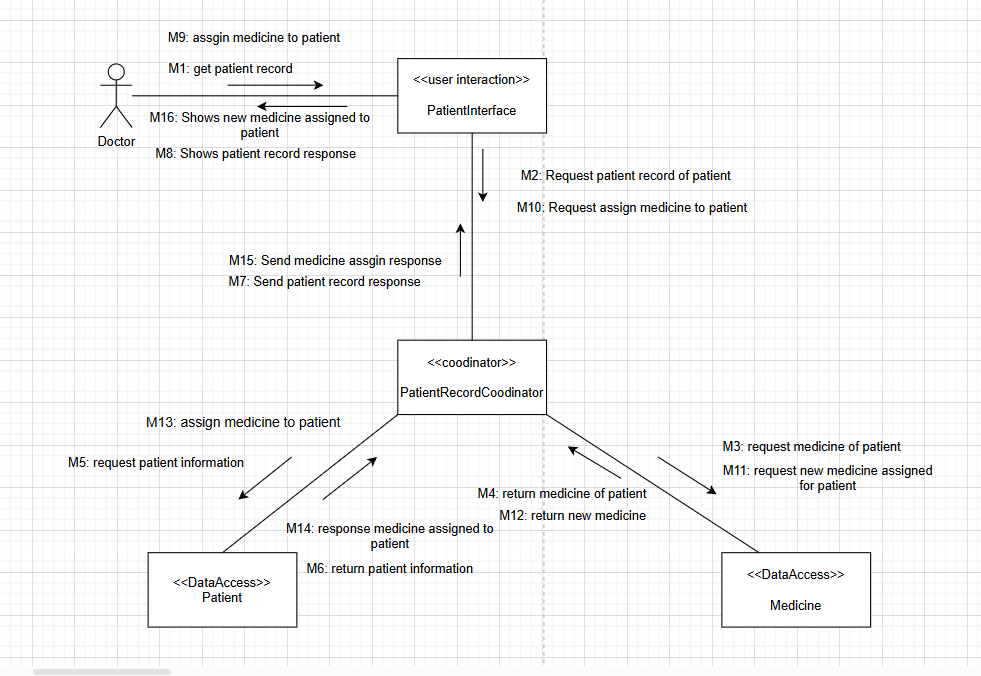
M12: return medicine

M13: assign medicine to patient

M14: response medicine assigned to patient

M15: Send medicine assign response

M16: Shows new medicine assigned to patient



**Alternative: [wrong information]**

#### 

| UC ID and Name: | UC-05: Prescribe Medicine |
| --- | --- |
| Summary | System allows a doctor to write a medical prescription for a patient. |
| Dependency: | N/A |
| Actors: | Doctor |
| Preconditions | The user successfully logs into the system with a doctor account. |
| Alternative sequences: | 7.1 There is no available medicine in storage.  8.1 Send notification “no available medicine” to the doctor.  7.2 the medicine added to patient is duplicate  8.2 Send notification “this medicine is already assigned to the patient” to the doctor. |
| Postcondition: | Patient’s medical prescription is updated successfully. |

**Messages**

**M12.1: Return no available medicine of patient**

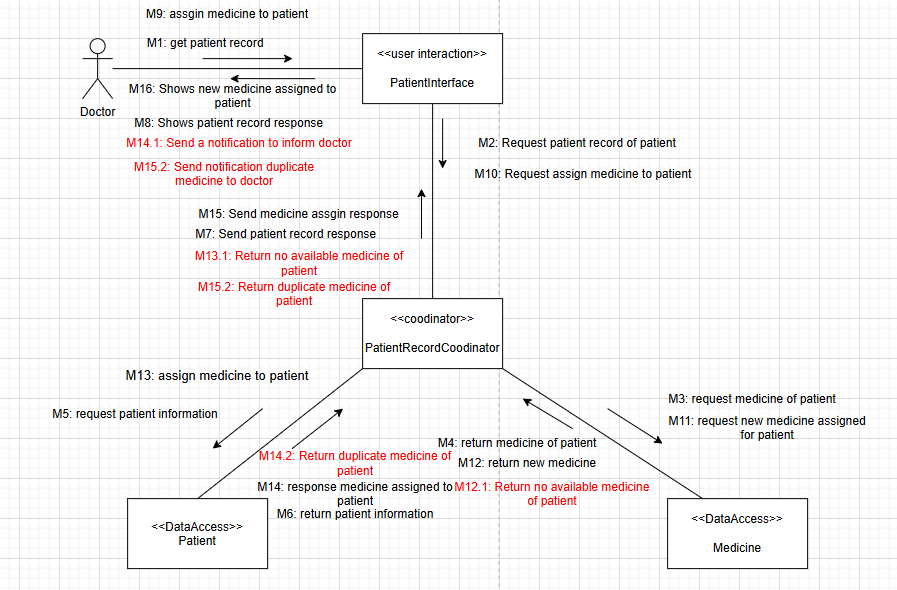
**M13.1: Return no available medicine of patient**

**M14.1: Send a notification to inform doctor**

**M14.2: Return duplicate medicine of patient**

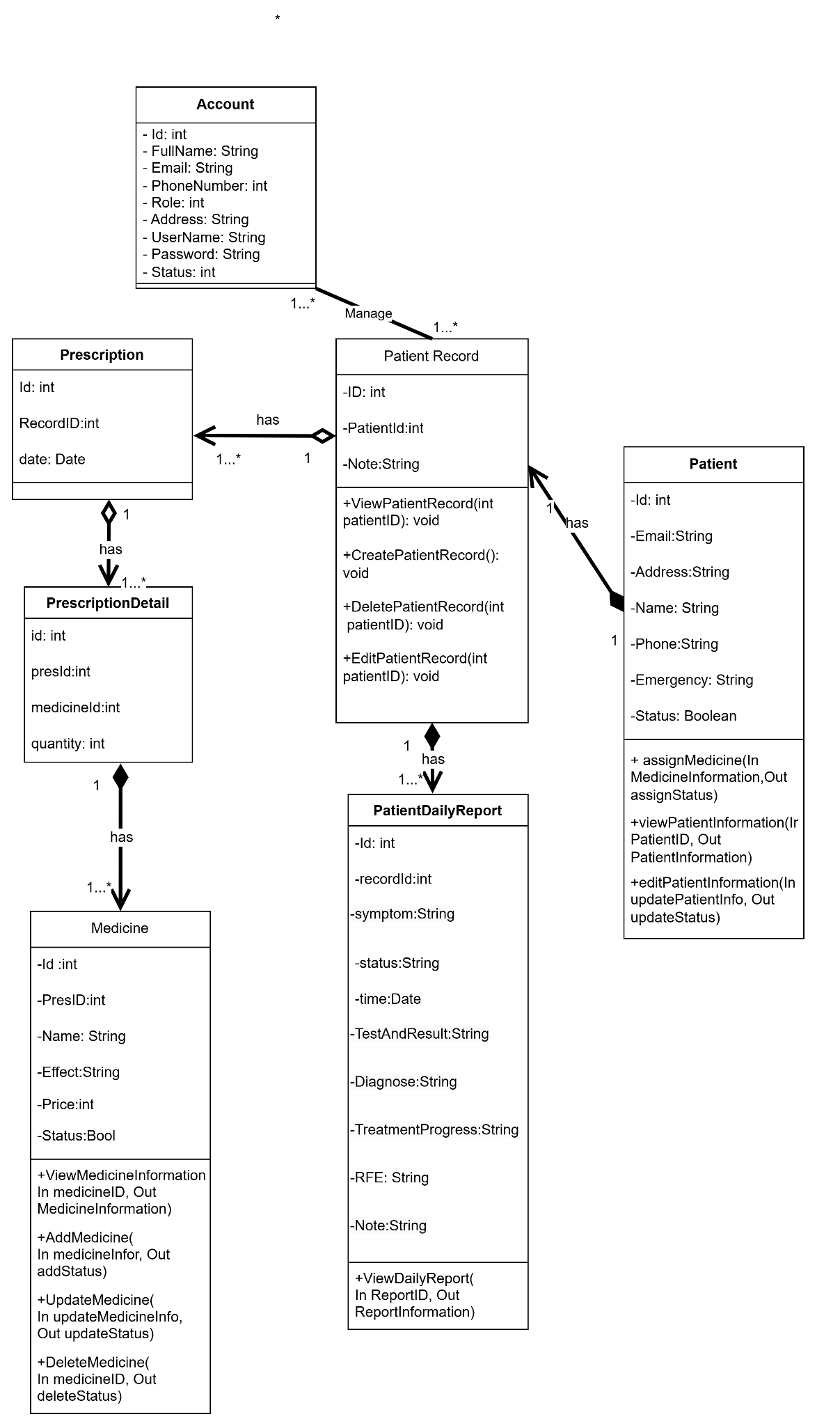
**M15.2: Return duplicate medicine of patient**

**M16.2: Send notification duplicate medicine to doctor**

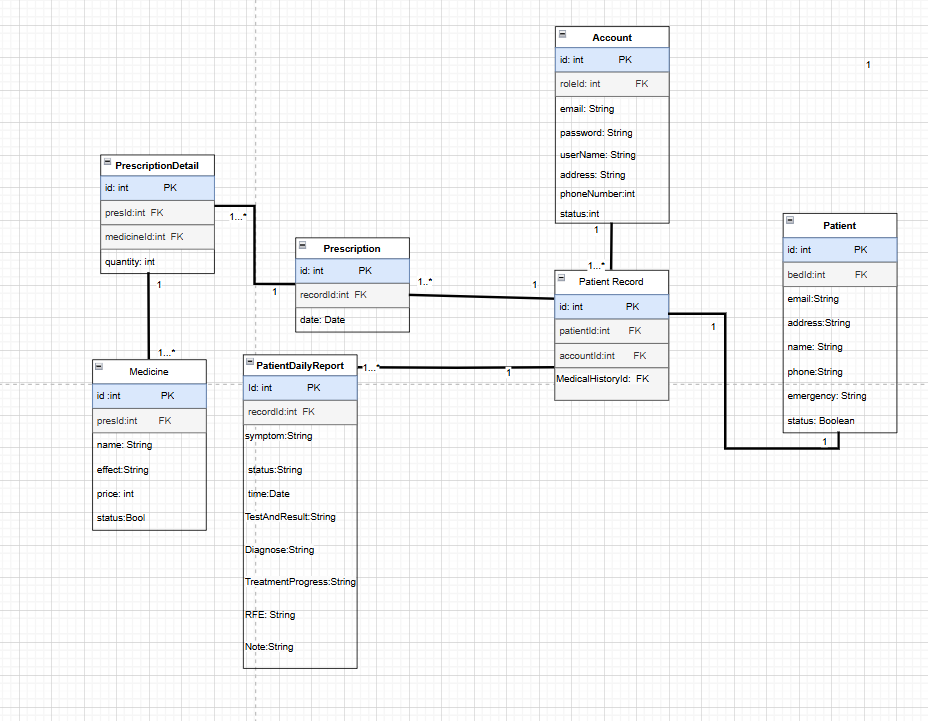


#### 6.4 Detail Design

###### 6.4.1 Class Diagram



###### 6.4.2 Relational Database



###### 6.4.3 Pseudocode

Patient:

* Attribute: id, Address, name, email,address,phone,emergency
* Operation:
* assignMedicine (In MedicineID,In PatientID,Out assignStatus):

**BEGIN**

foundMedicine <- findMedicineById(MedicineID)

foundPatient<- findPatientById(PatientID)

**IF** foundMedicine is NULL **THEN**

PRINT "Medicine not found"

Out assignStatus <- false

**ELSE IF** foundMedicine.status = false **THEN**

PRINT "Medicine not available"

Out assignStatus <- false

**ELSE IF**  foundPatient.status = true **THEN**

**FOR EACH**

medicine IN foundPatient.medicineList DO

**IF** medicine.id == In MedicineID **THEN**

Out assignStatus <- false

**BREAK**

**END IF**

**END FOR**

**ELSE**

Prescription.addMedicine <- foundMedicine

**THEN**

UpdatePrescription(prescription,foundPatient)

Out assignStatus <- true

**END IF**

**END**

### **7. Layers of Abstraction Pattern**

| Presentation Layer | viewDailyReportList.jsp, editDailyReport.jsp, addDailyReport.jsp,  deleteDailyReport.jsp,  login.jsp,  logout.jsp |
| --- | --- |
| Coordinator Layer | DeleteDailyReportServlet.java,  EditDailyReportServlet.java,  AddDailyReport.java,  ViewDailyReportServlet.java,  LoginServlet.java,  LogoutServlet.java, |
| Data Access Layer | PatientReportDailyDAO.java,  AccountDAO.java |

## 

### B. Lê Bá Hải Hà (HE176636)

#### 1. Requirement

1.1 [UC-18: Request Tests and Imaging](#_4e3e01z43qyo)

1.2 [UC-28: View List Role](#_rzegf73duo30)

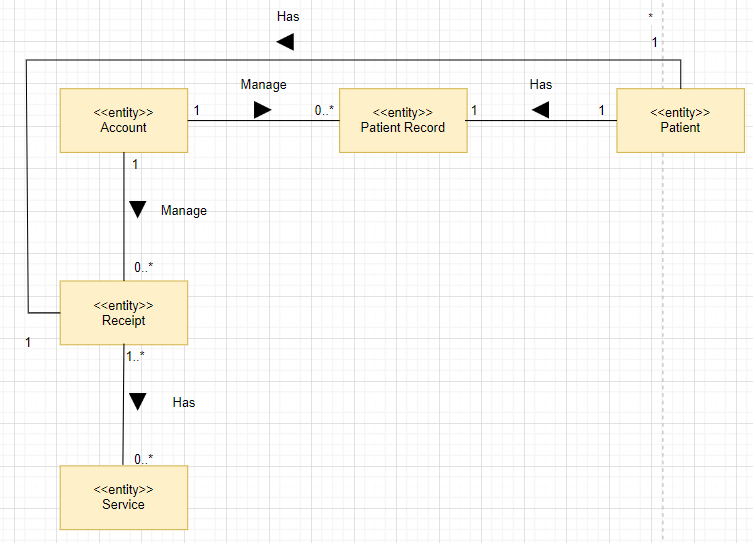
1.3 [UC-29: Create Role](#_krpgtn101bl)

1.4 [UC-30: Update Role](#_pa87ny8ymj4i)

#### 2. Static modeling

Entity Class

##### 2.1 Request Tests and Imaging



##### 2.2 View List Role

##### 2.3 Create Role

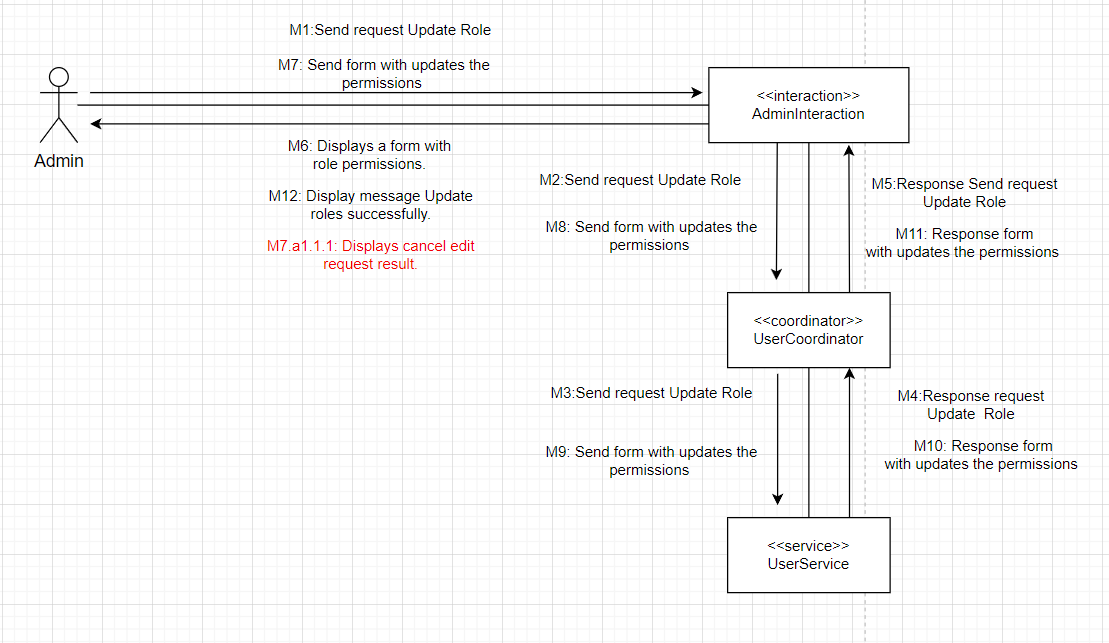
#### 

##### 2.4 Update Role

#### 

#### 3. Dynamic Modeling

##### 3.1 Request Tests and Imaging



###### 3.1.1 Identify Boundary and Internal objects:

* Boundary object: Doctor Interaction
* Internal object: UserCoordinator, UserService

###### 3.1.2 Messages

M1: Doctor send request test and imaging to DoctorInteraction

M2: DoctorInteraction send request test and imaging to UserCoordinator

M3: UserCoordinator send request test and imaging to UserService

M4: UserService response test and imaging to UserCoordinator

M5: UserCoordinator response test and imaging to DoctorInteraction

M6: DoctorInteraction display a form test and imaging for Doctor

M7: Doctor send request submit form to DoctorInteraction

M8: DoctorInteraction send request submit form to UserCoordinator

M9: UserCoordinator send request submit form to UserService

M10: UserService response submit form to UserCoordinator

M11: UserCoordinator response submit form to DoctorInteraction

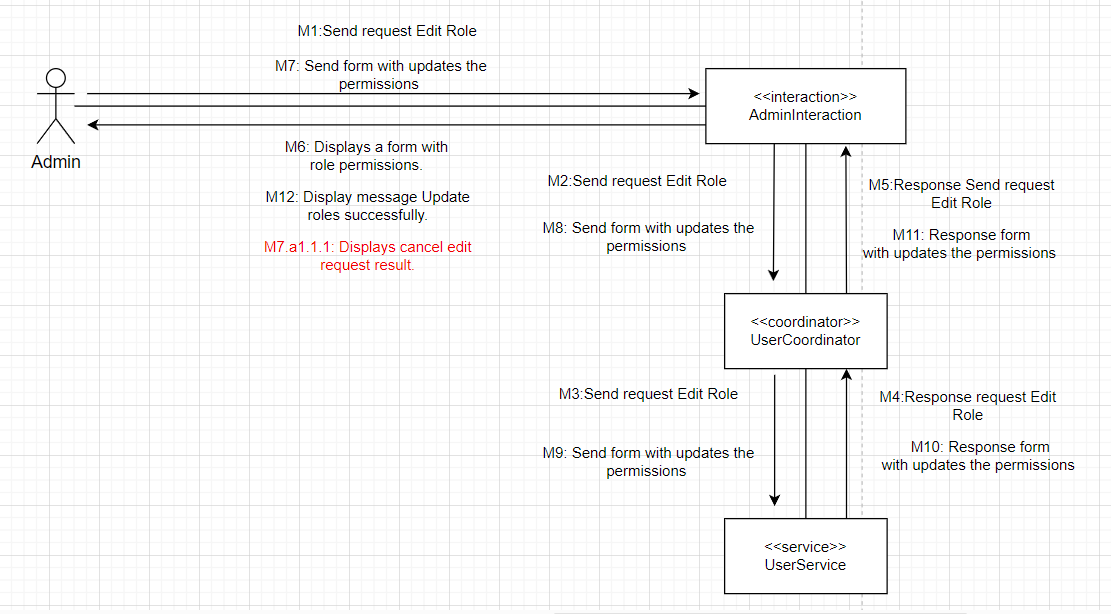
M12: DoctorInteraction display message request test and imaging successfully for Doctor

###### 3.1.3 Alternative

M1.1.a: Doctor send request test and imaging to DoctorInteraction

M2.1.a :DoctorInteraction display an error message for Doctor

##### 3.2 Update Role



###### 3.2.1 Identify Boundary and Internal objects:

* Boundary objects: AdminInteraction
* Internal objects: UserCoordinator, UserService

###### 3.2.2 Messages

M1: Admin send request Edit Role to AdminInteraction

M2: AdminInteraction send request Edit Role to UserCoordinator

M3: UserCoordinator send request Edit Role to UserService

M4: UserService response Edit Role to UserCoordinator

M5: UserCoordinator response Edit Role to AdminInteraction

M6: AdminInteraction displays a form with role permissions for Admin

M7: Admin send form with updates the permissions to AdminInteraction

M8: AdminInteraction send form with updates the permissions to UserCoordinator

M9: UserCoordinator send form with updates the permissions to UserService

M10: UserService response form with updates the permissions to UserCoordinator

M11: UserCoordinator response form with updates the permissions to AdminInteraction

M12: AdminInteraction display message Update roles successfully for Admin

###### 3.2.3 Alternative

M1: Admin send request Edit Role to AdminInteraction

M2: AdminInteraction send request Edit Role to UserCoordinator

M3: UserCoordinator send request Edit Role to UserService

M4: UserService response Edit Role to UserCoordinator

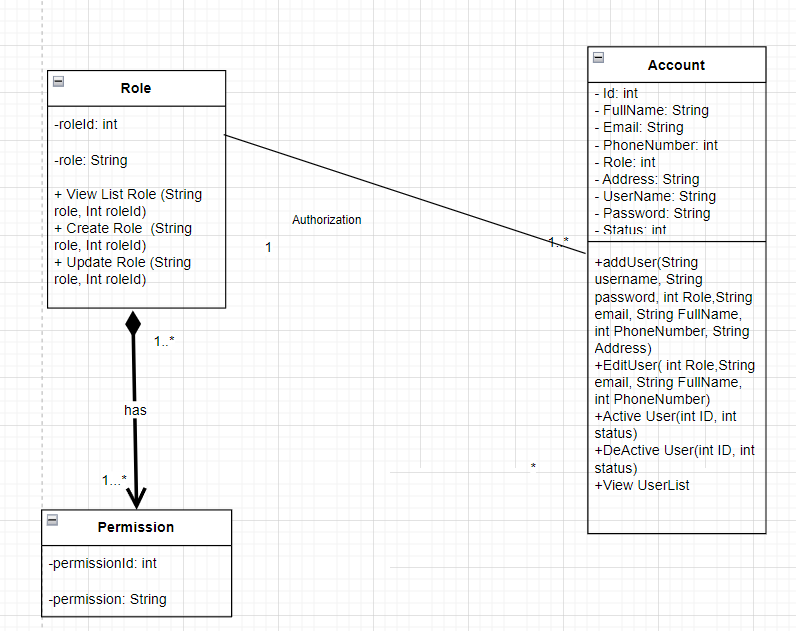
M5: UserCoordinator response Edit Role to AdminInteraction

M6: AdminInteraction displays a form with role permissions for Admin

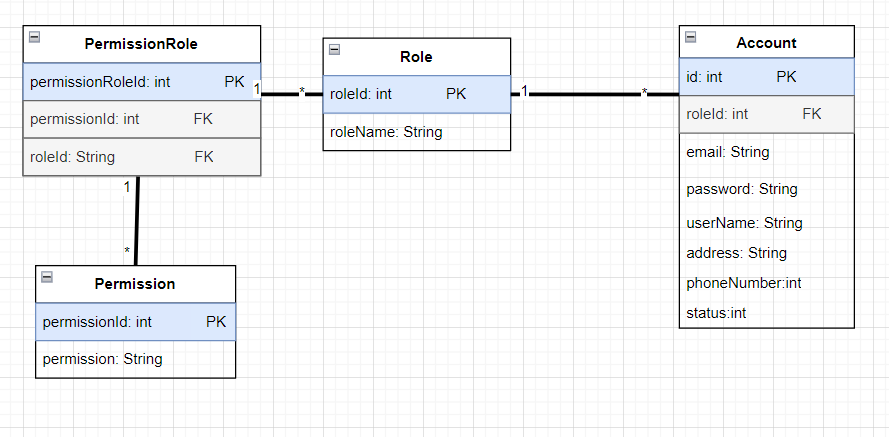
M7.a1.1.1: After 5m, AdminCoordinator will display cancel edit request result to Admin

### **4.Detail design**

#### 4.1.Class Diagram



#### 4.2 Relational Database



#### 4.3.Pseudocode

Role:

Attribute: roleId, roleName

Operation:

getterRoleId(OUT OrderId)

setterRoleId(IN orderId)

UpdateRole(IN roleName, IN RoleId) OUT Role

Begin;

private anRole;

setanRole.roleId = roleId;

setanRole.roleName = roleName;

return anRole;

End;

### 5.Change Requirement

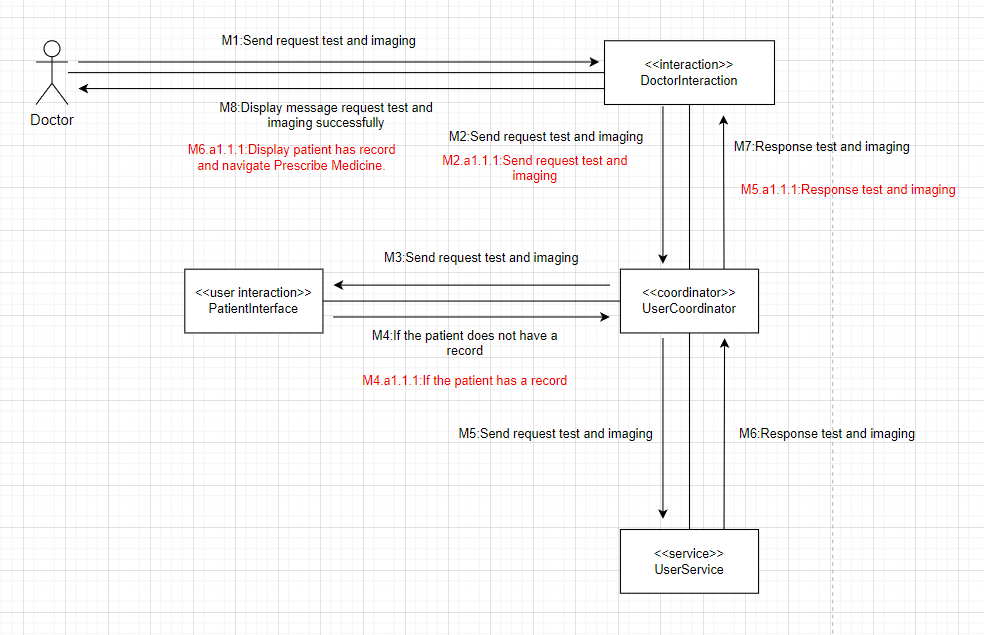
#### 5.1 Use Case model

### 

| UC ID and Name: | UC-18: Request Tests and Imaging |
| --- | --- |
| Summary | This use case describes the process by which a doctor requests tests and imaging for a patient. |
| Dependency: | N/A |
| Actors: | Doctor |
| Preconditions | * The user has logged in as Doctor. |
| Main sequence: | 1. The doctor selects a patient from the list. 2. If the patient does not have a patient record, the system displays the patient's detailed information. 3. The doctor clicks on the option to request tests and imaging. 4. The system displays a form for requesting tests and imaging. 5. The doctor submits the request. 6. The system saves the request and updates the patient's records. |
| Alternative sequences: | 2.a.1.1: If the patient has a patient record, the system will navigate to UC5: Prescribe Medicine. |
| Nonfunctional requirements: | Patient Records Update: Patient information, including details of tests and imaging performed, must be updated and accurately stored in the patient records system. |
| Postcondition: | N/A |
| Outstanding questions: | N/A |

#### 

#### 5.2 Dynamic modeling



#### 5.3 Messages

M1: Doctor send request submit the quest to DoctorInteraction

M2: DoctorInteraction send request submit the quest to UserCoordinator

M3: UserCoordinator send request submit the quest to PatientInterface

M4: PatientInterface validate and send request test and imaging to UserCoordinator

M5: UserCoordinator send request submit the quest to UserService

M6: UserService response request submit the quest to UserCoordinator

M7: UserCoordinator response request submit the quest to DoctorInteraction

M8: DoctorInteraction display message request test and imaging successfully for Doctor

#### 5.4 Alternative

M1: Doctor send request submit the quest to DoctorInteraction

M2: DoctorInteraction send request submit the quest to UserCoordinator

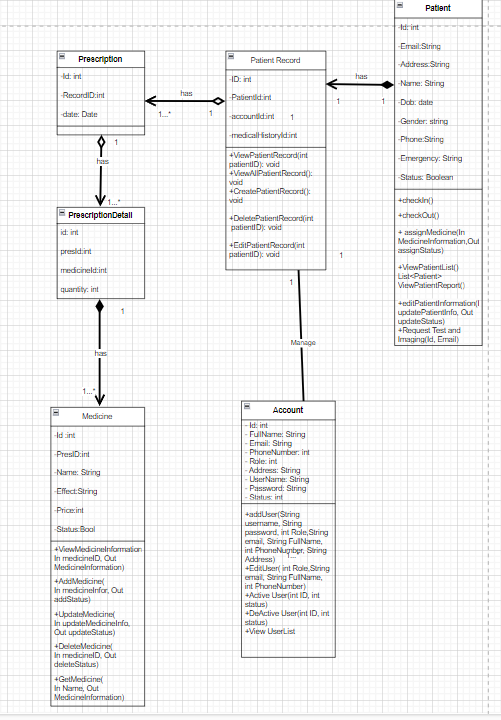
M3: UserCoordinator send request submit the quest to PatientInterface

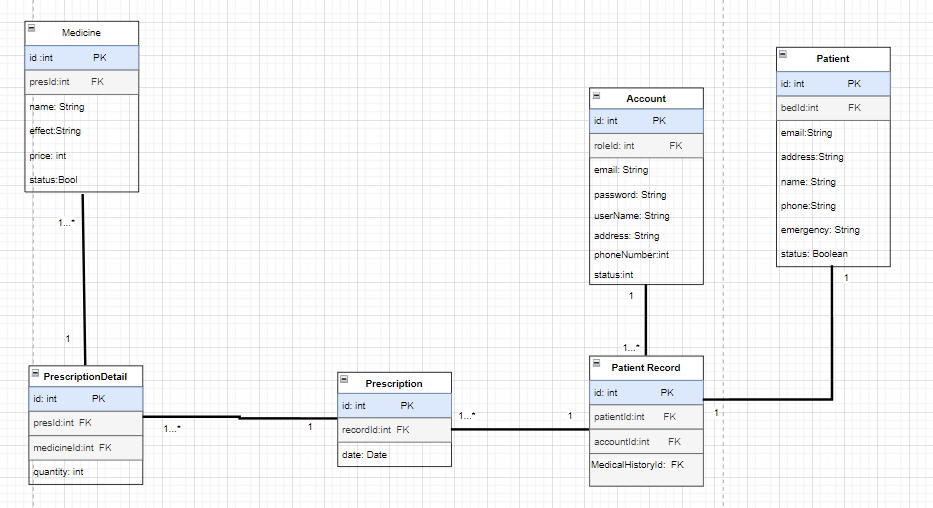
M4.a1.1.1: PatientInterface validate and response request submit the quest to UserCoordinator

M5.a1.1.1: UserCoordinator response request submit the quest to DoctorInteraction

M6.a1.1.1:DoctorInteraction display patient has record for Doctor and navigate Prescribe Medicine

#### 5.5 Class Diagram





#### 5.6 Pseudocode

Patient:

Attribute: id, name, email, address, phone, emergency

Operation:

requestTestsAndImaging (IN doctorId, IN patientId, IN testDetails, OUT requestStatus):

BEGIN

foundDoctor <- findDoctorById(doctorId)

foundPatient <- findPatientById(patientId)

IF foundDoctor is NULL THEN

PRINT "Doctor not found"

OUT requestStatus <- false

ELSE IF foundPatient is NULL THEN

PRINT "Patient not found"

OUT requestStatus <- false

ELSE

request <- createTestRequest(doctorId, patientId, testDetails)

IF request is NULL THEN

PRINT "Request creation failed"

OUT requestStatus <- false

ELSE

updatePatientRecords(foundPatient, request)

OUT requestStatus <- true

END IF

END IF

END

createTestRequest (IN doctorId, IN patientId, IN testDetails) OUT request:

BEGIN

private request

set request.doctorId = doctorId

set request.patientId = patientId

set request.testDetails = testDetails

return request

END

updatePatientRecords (IN patient, IN request):

BEGIN

// Update the patient's records with the new tests and imaging details

END

findDoctorById (IN doctorId) OUT doctor:

BEGIN

// Find and return doctor by id

END

findPatientById (IN patientId) OUT patient:

BEGIN

// Find and return patient by id

END

C. Trương Quang Lộc (HE170736)

#### 1.Requirement

[UC-21 : View Patient List](#_a71p2ejprzr8)

[UC-22 : Seach Patient](#_dvac48b833rz)

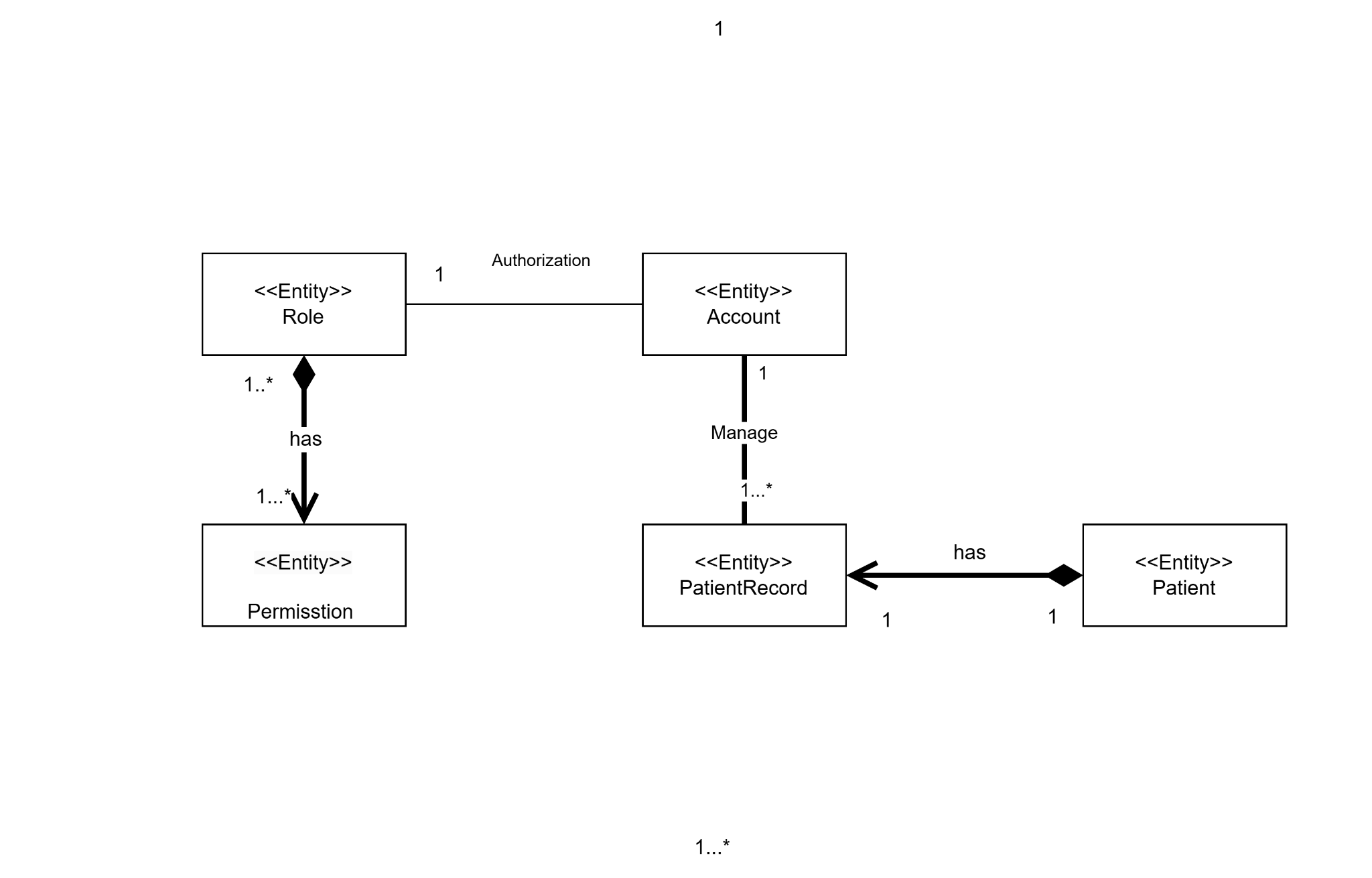
[UC-23 : View Patient Record](#_myob8tt074pu)

[UC-24 : View Patient Record](#_lpkmzkhys9t7)

[UC-25 : Add Patient Record](#_lpkmzkhys9t7)

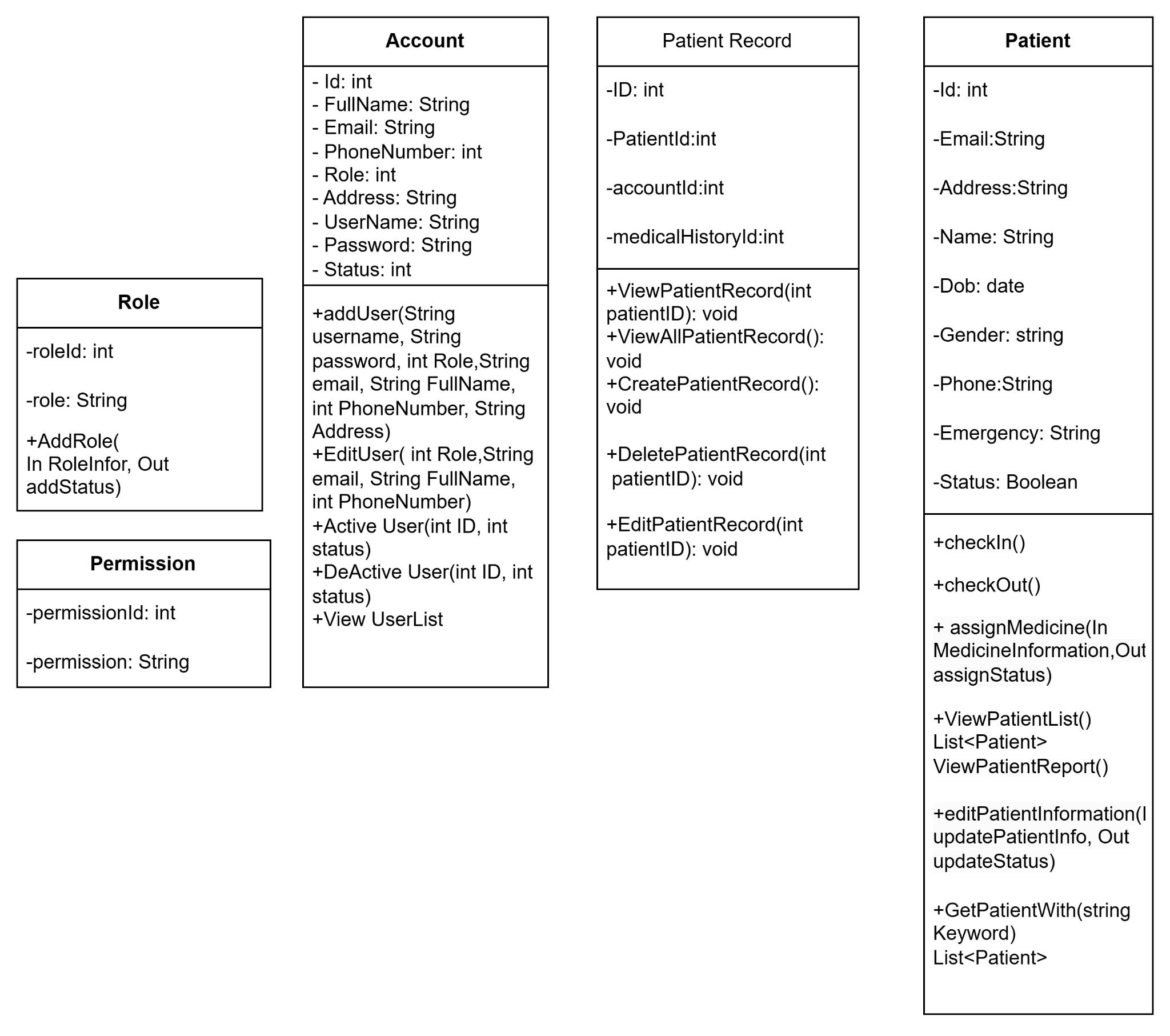
#### 2.Static Modeling

##### 2.1 Entity Class



##### 

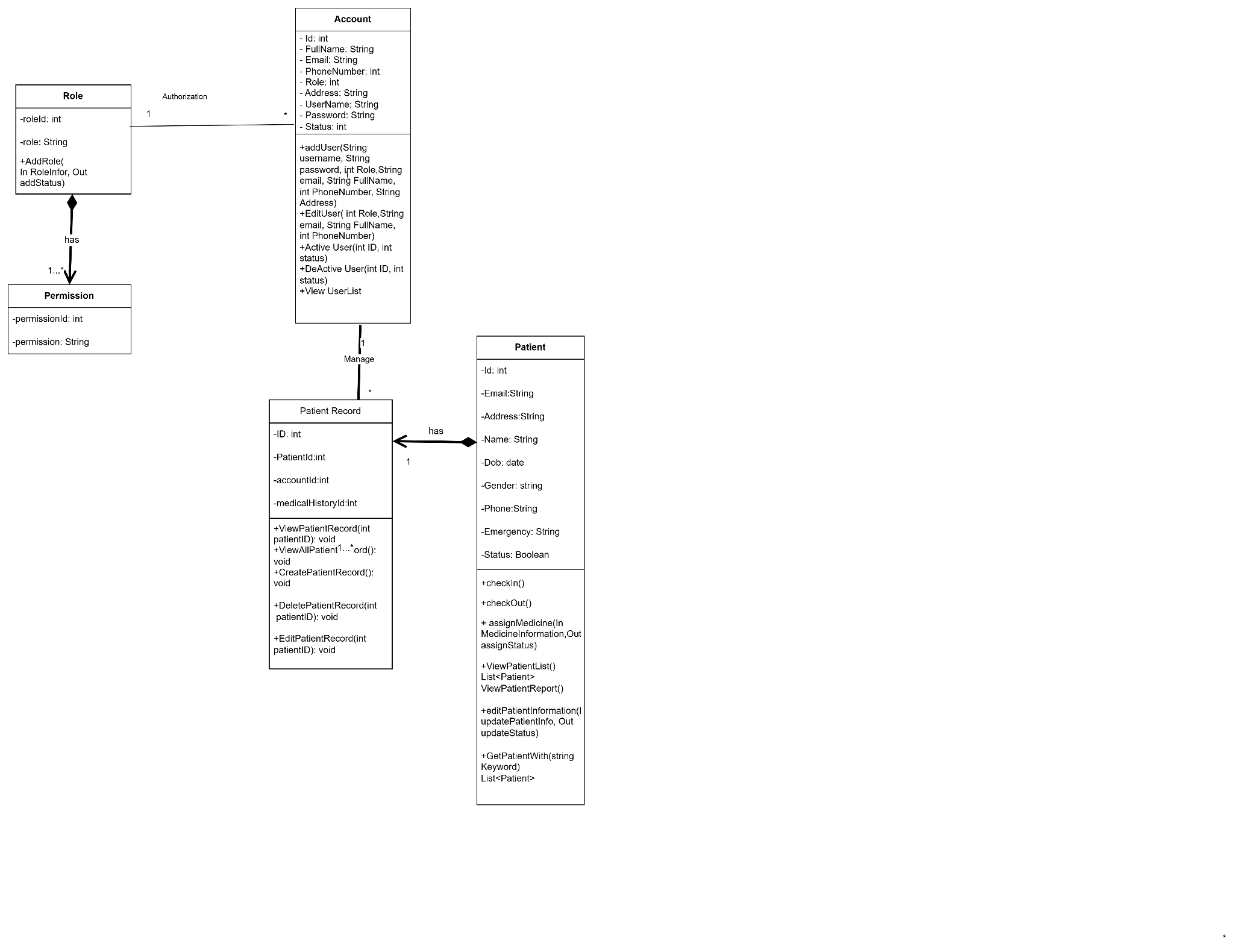
##### 2.2 Class Attribute



#### 

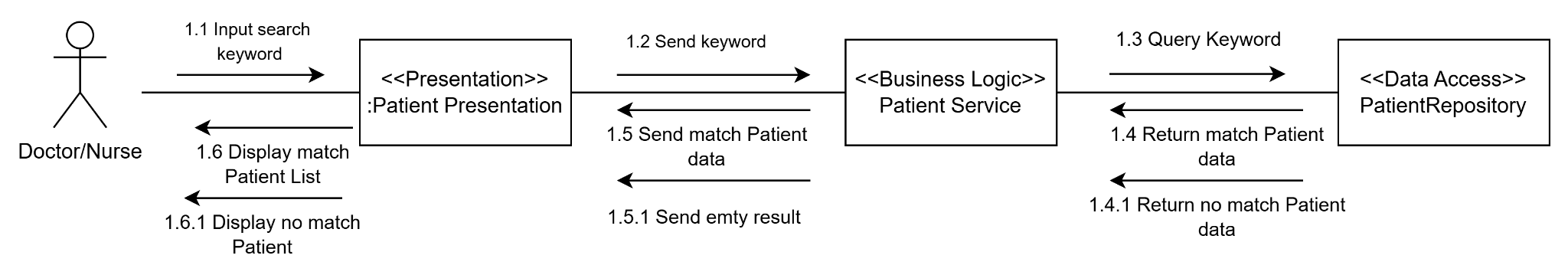
#### 3.Class Diagram

#### 4.Dynamic Modeling



##### 4.1 Search

a.Communication



**b. Identify boundaries and internal objects.**

Boundary object : Patient Presentation

Internal Object : User(Doctor/Nurse)

**c. Message**

M1.1: Input search keyword

M1.2: Send Keyword

M1.3: Query Keyword

M1.4: Return Match patient data

M1.5: Send match patient data

M1.6: Display Patient data

**d. Alternative sequence**

M1.4.1: Return no Match patient data

M1.5.1: Send empty patient data

M1.6.1: Display no match Patient

**e. Pseudocode**

**FUNCTION** searchPatientFunction

**ATTRIBUTE**

searchKeyWord : String

patientList : LIST = ["Patient A", "Patient B", "Patient C"]

found : bool

**Begin Function**

DISPLAY "Enter patient name to search:"

WAIT for user to enter “searchKeyWord” and click "Search" button

execute operation : searchPatient(searchKeyWord)

FOR each patient in patientList

IF patient.name matches searchName THEN

DISPLAY patient details

SET found to TRUE

END IF

END FOR

IF found is FALSE THEN

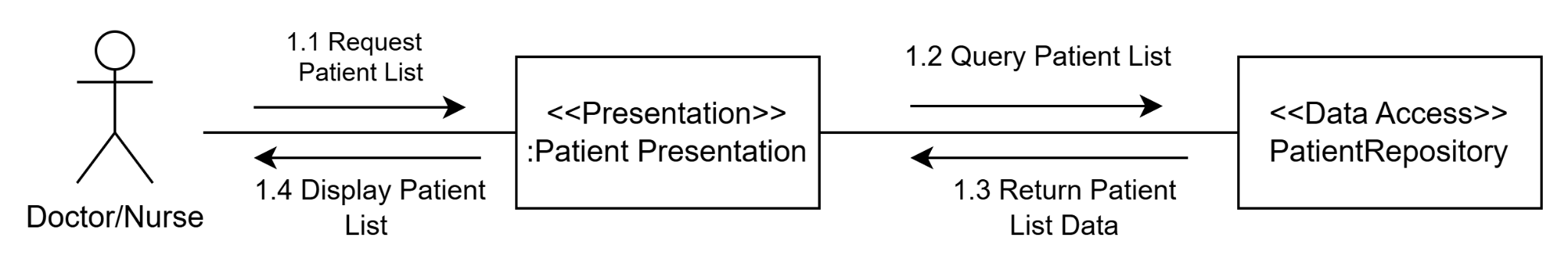
DISPLAY "No match Patient"

END IF

**END FUNCTION**

##### 4.2 View Patient List

a.communication diagram



**b. Identify boundary and internal objects.**

Boundary object : Patient Presentation

Internal Object : User(Doctor/Nurse)

**c. Message**

M1.1: Request patient list

M1.2: Send view request

M1.3: Send patient list data

M1.4: Display Patient list data

**e. Pseudocode**

**BEGIN**

**FUNCTION** viewPatientList

**ATTRIBUTE**

patientList : LIST = ["Patient A", "Patient B", "Patient C,..."]

**Begin Function**

WAIT for user to click "Patient List" button

patientList = ViewPatientList()

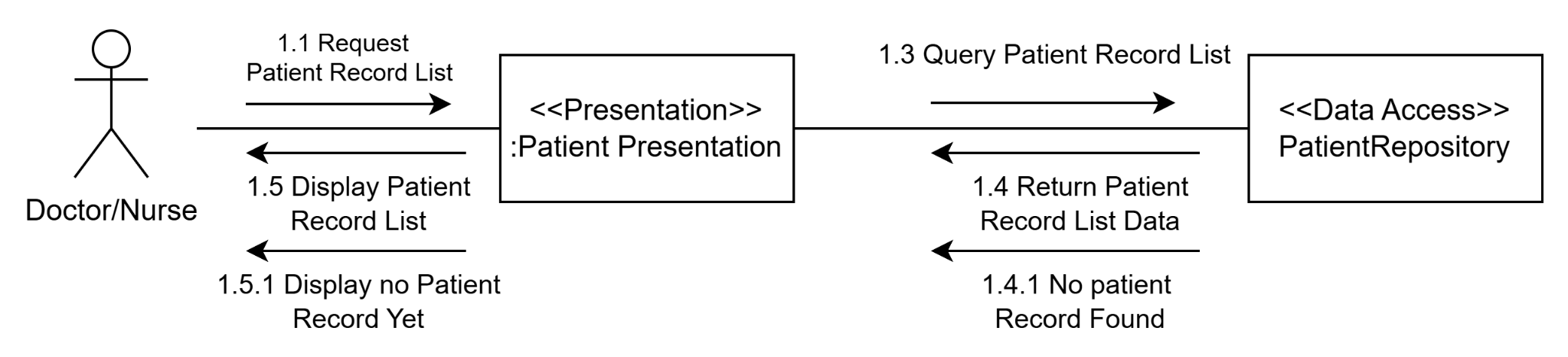
DISPLAY patientList

**END FUNCTION**

END

##### 4.3. View Patient Record List

a.communication diagram



**b. Identify boundary and internal objects.**

Boundary object : Patient Presentation

Internal Object : User(Doctor/Nurse)

**c. Message**

M1.1: Request patient list

M1.2: Send view request

M1.3: Query patient list

M1.4: Return patient list data

M1.5: Send patient list data

M1.6: Display Patient list data

**d. Alternative sequence**

M1.4.1: Return no patient record data

M1.5.1: Send empty patient record data

M1.6.1: Display no Patient record

**e. Pseudocode**

**BEGIN**

**FUNCTION** viewPatientRecordList

**ATTRIBUTE**

List<patientRecord>

**Begin Function**

WAIT for user to click "Patient Record List" button

IF patient already have record THEN

DISPLAY all record of that patient

ELSE

DISPLAY patient did not have record yet

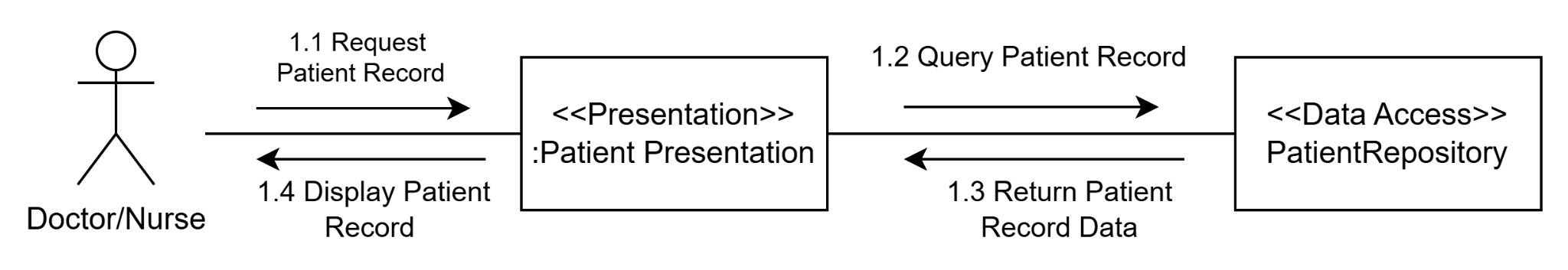
END IF

**END FUNCTION**

END

##### 4.4. View patient record

a.communication diagram

****

**b. Message**

M1.1: Request patient list

M1.2: Send view request

M1.3:Send patient list data

M1.4: Display Patient list data

**d. Alternative sequence**

**e. Pseudocode**

**BEGIN**

**FUNCTION** viewPatientRecord

**ATTRIBUTE**

patientRecord

PatientRecordID

**Begin Function**

WAIT for user to click "View Patient Record" button

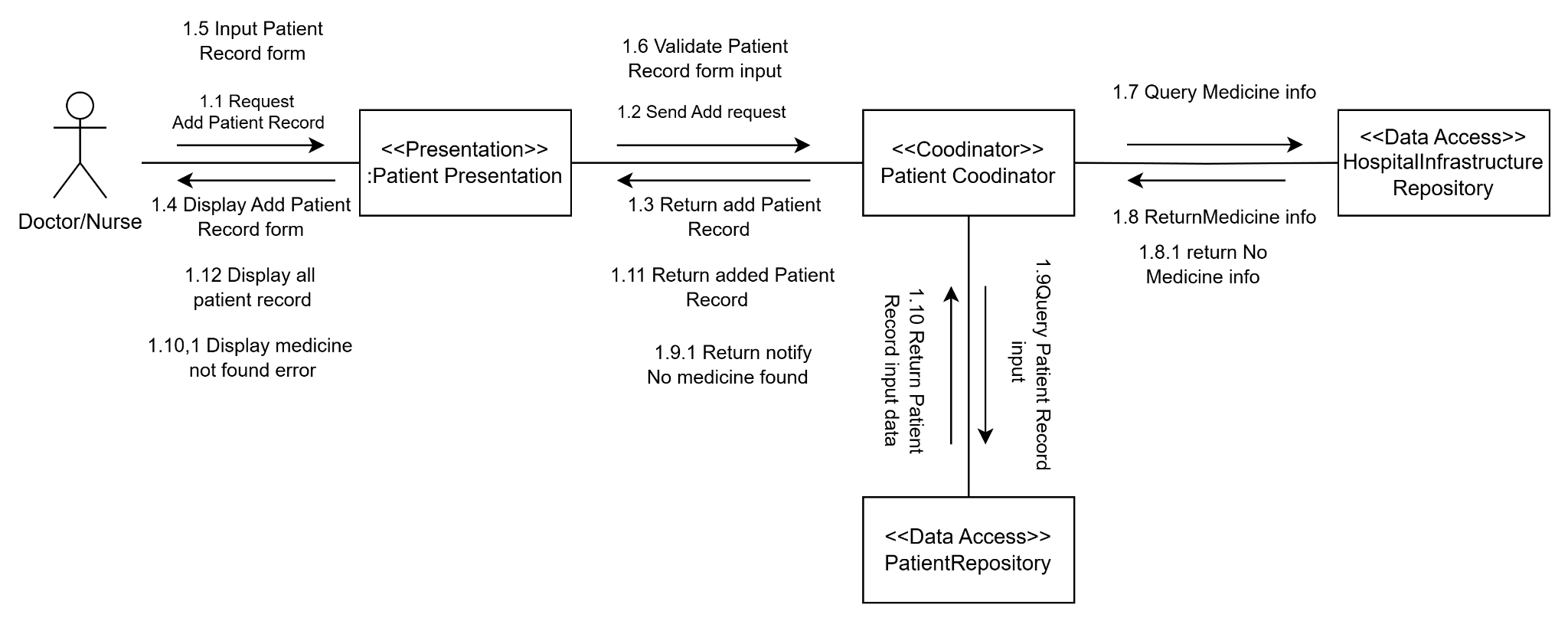
patientRecord = ViewPatientRecord(PatientRecordID)

DISPLAY patientRecord

**END FUNCTION**

##### 4.5 Add Patient Record

a.communication diagram



b. **Identify boundary and internal objects.**

Boundary object : Patient Presentation

Internal Object : Patient Coordinator,PatientRepository,HospitalInfrastructureRepository

c.Message

M1.1 Request Add Patient Record

M1.2 Send Add request

M1.3 Return add Patient Record

M1.4 Display Add Patient Record form

M1.5 Input Patient Record form

M1.6 Validate Patient Record form input  
M1.7 Query Medicine info

M1.8 Return Medicine info

M1.9 Query Patient Record input

M1.10 Return Patient Record input data

M1.11 Return added Patient Record

M1.12 Display all patient record

**d.Alternative**

M1.8.1 return No Medicine info

M1.9.1 Return No Medicine info

M1.10.1 Display medicine not found error

e.pseudocode

FUNCTION AddPatientRecord

ATTRIBUTE

patientRecordData

medicineInfo

allPatientRecords

Begin Function

DISPLAY "Add Patient Record Form"

WAIT for user to enter patientRecordData and click "Submit" button

SEND patientRecordData to PatientCoordinator

validationResult = PatientCoHospitalInfractureRepository ordinator.GetMedicine(“medicineInfo ” )

Wait the HospitalInfractureRepository to validate medicine in patientRecordData

Return validationResult

IF validationResult is TRUE THEN

SEND patientRecordData to PatientRepository

PatientRepository.CreatePatientRecord(patienpatientRecordData)

SEND add success notification to PatientCoordinator

DISPLAY savedData

ELSE

SEND add fail notification to PatientCoordinator

END IF

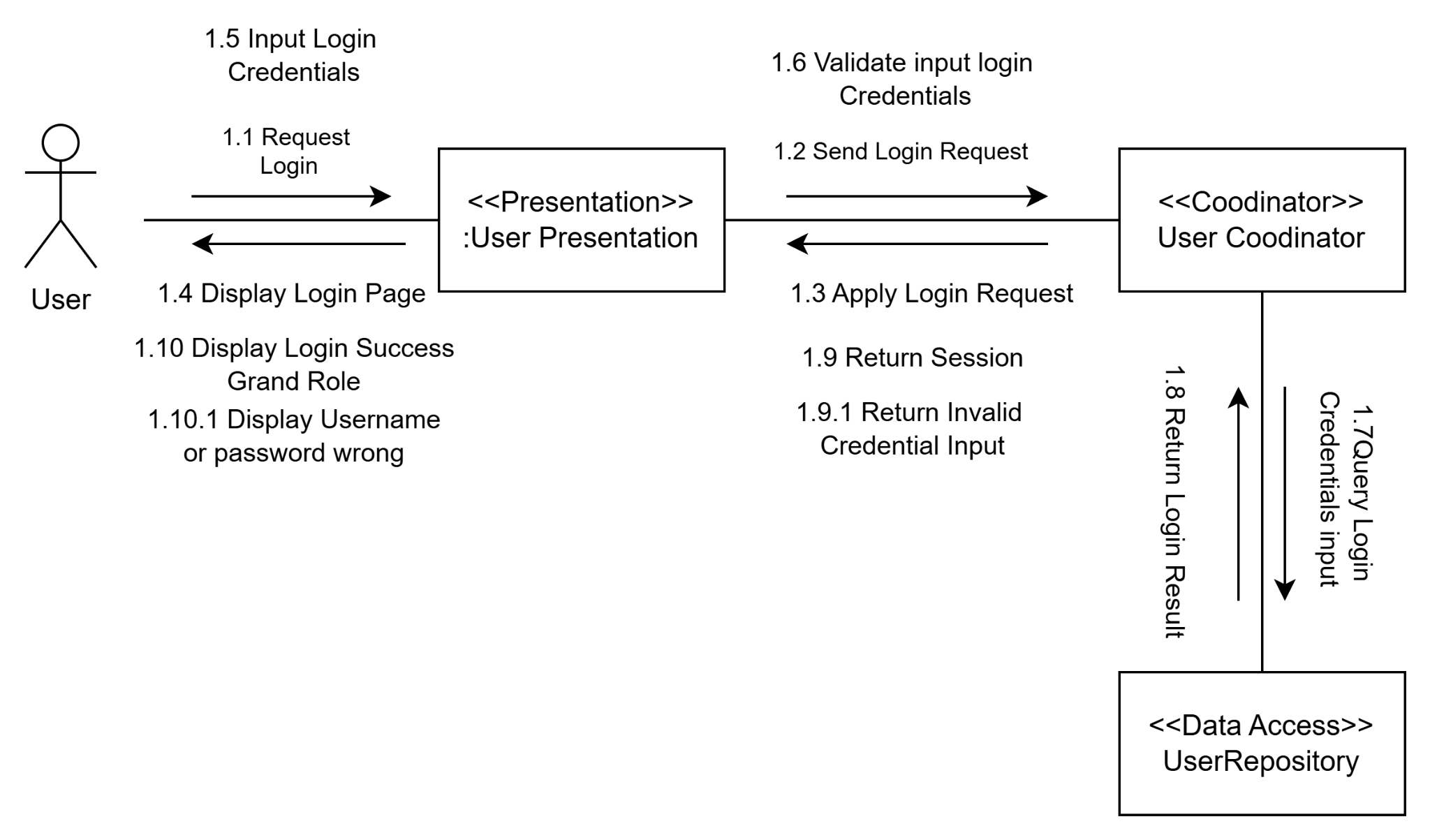
PatientCoordinator return notification to PatientPresentation

END IF

END FUNCTION

##### 4.6 Authenticate User

a.Communication Diagram



b.Boundary Object

External object : User

Boundary Object : User Presentation

c.Message

M1.1 Request Login

M1.2 Send Login Request

M1.3 Apply Login Request

M1.4 Display Login Page

M1.5 Input Login Credentials

M1.6 Validate INput Login Credentials

M1.7 Query Login Credentials input

M1.8 Return Login Result

M1.9 return Session

M1.10 Display Login Success

d.Alternative sequence

M1.9.1 return invalid credential input

M1.10.1 Display Login Success

e.Pseudocode

BEGIN DISPLAY loginPage

INPUT username, password

FUNCTION encryptData(data, publicKey):

RETURN RSA\_Encrypt(data, publicKey) publicKey = FETCH publicKey from server encryptedUsername = encryptData(username, publicKey) encryptedPassword = encryptData(password, publicKey)

SEND encryptedUsername, encryptedPassword TO server via HTTPS END

#### 5.Change Requirement

##### 5.1 Requirement

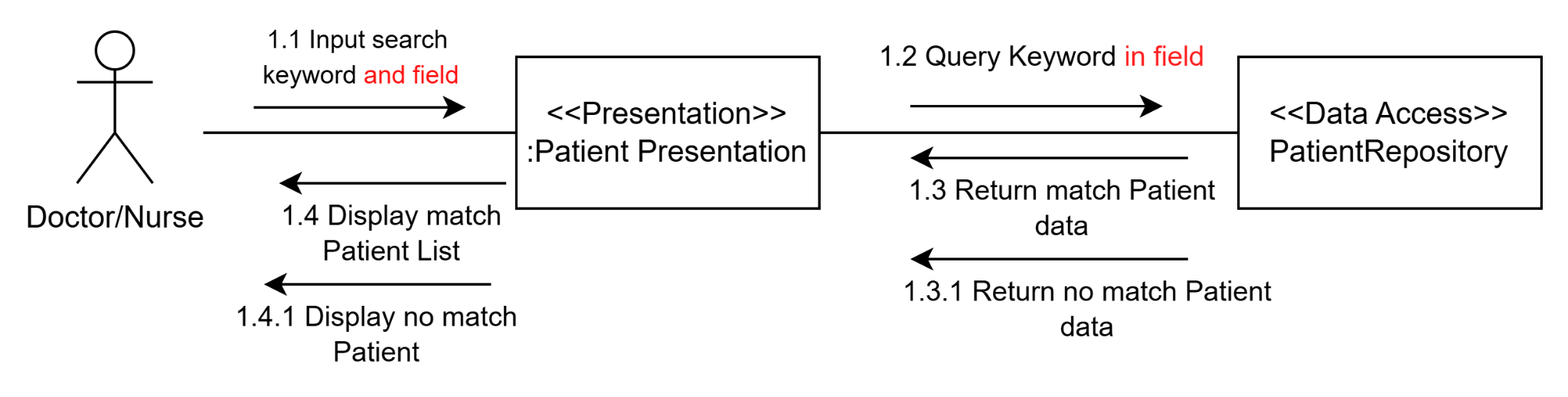
#### 

| UC ID and Name: | UC-22: Search Patient |
| --- | --- |
| Summary | This use case describes the process by which users want to view a specific patient by using searching bar |
| Dependency: |  |
| Actors: | Doctor,Nurse |
| Preconditions | User |
| Main sequence: | 1. The user accesses the login page.  2. The system displays the login form.  3. The user enters their username and password.  4. The system validates the user's credentials.  5. If the credentials are valid, the system logs the user in and grants access.  6. Once logged in,user will be redirect to the homepage the user can access the system's features and functions  7. User click on Patient List  8. System send redirect user to the Patient List tab and get all patient form a list  9..User choose what to search via a drop down box like (Email,name,phone)  10. User input a specific keyword into search bar  11.System return patient with the field that contain the keyword with field user input in the search bar |
| Alternative sequences: | 9. User does not choose what to search  10. User Input keyword into search bar  11. System query keyword with the default field is patient name  12. System return patient with the name contain keyword |
| Nonfunctional requirements: | Search response times must be fast and results accurate |
| Postcondition: | User must login with an account have role Doctor or Nurse |
| Outstanding questions: | N/A |
| Business rule: |  |

#### 

##### 5.2 Class Attribute

##### 5.3 Dynamic Modeling



**b. Identify boundaries and internal objects.**

Boundary object : Patient Presentation

Internal Object : User(Doctor/Nurse)

**c. Message**

M1.1: Input search keyword and field

M1.2: Query Keyword

M1.3: Return Match patient data

M1.4: Display Patient data

**d. Alternative sequence**

M1.3.1: Return no Match patient data

M1.4.1: Display no match patient

**e. Pseudocode**

**FUNCTION** searchPatientFunction

**ATTRIBUTE**

searchKeyWord : String

patientList : LIST = ["Patient A", "Patient B", "Patient C"]

found : bool

field

**Begin Function**

DISPLAY "Enter patient name to search:"

WAIT for user to enter “searchKeyWord” and Choose field then click "Search" button

execute operation : searchPatient(searchKeyWord ,field)

FOR each patient in patientList

IF patient.field matches searchKeyword THEN

DISPLAY patient details

SET found to TRUE

END IF

END FOR

IF found is FALSE THEN

DISPLAY "No match Patient"

END IF

**END FUNCTION**

### D. Phạm Quang Thắng (HE172620)

#### 1. Requirements

##### 1.1 Patient registration

#### 

| UC ID and Name: | UC-12: Patient registration |
| --- | --- |
| Summary | Receptionist adding a patient with that person's personal information into the list of patient in the system |
| Dependency: | Include: Login |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist * The number of patients in the system must not exceed the maximum allowed number |
| Main sequence: | 1. Extend login use case. 2. Receptionist go to the Patient Registration site. 3. System show patient information form 4. Receptionist fills in patient information 5. Receptionist clicks “Create” button 6. The system adds 1 patient with the same information entered as above to the list of patients and adds it to the system. |
| Alternative sequences: | Step 6: The system checks that the number of patients in the system has exceeded the allowed number, the screen displays a notification that the hospital has reached the maximum number of patients allowed and no more patients are allowed. |
| Nonfunctional requirements: | Performance: Check and add patients to the system in less than 10 seconds |
| Postcondition: | 1 patient will be added to the system |
| Outstanding questions: | N/A |

##### 1.2 Patient discharge

| UC ID and Name: | UC-13: Patient Discharge |
| --- | --- |
| Summary | Receptionist remove patient from their bed and change status to discharge |
| Dependency: | Extend: view patient list |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist * There is at least 1 patient in the system |
| Main sequence: | 1. Receptionist login in the system successfully 2. Receptionist go to list patient 3. System show list patient 4. Receptionist clicks “Discharge” button 5. Receptionist select option yes 6. The system will change this patient status to discharge in the list of patients. |
| Alternative sequences: | Step 5: Receptionist select option no  6. Back to homepage |
| Nonfunctional requirements: | Performance: Remove patients to the list in less than 10 seconds |
| Postcondition: | 1 patient will be removed from the patient list |
| Outstanding questions: | N/A |

#### 

##### 1.3 View Receipt List

#### 

| UC ID and Name: | UC-14: View Receipt List |
| --- | --- |
| Summary | This use case describes the process of a receptionist viewing a list of receipts in the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | The user has logged into the system as a receptionist. |
| Main sequence: | 1. The receptionist accesses the receptionist page. 2. The system displays the receptionist page. 3. The receptionist clicks on the "Receipt List". 4. The system displays a list of receipts, including information such as receipt number, customer name, amount, and date created. |
| Alternative sequences: | Step 4: If there are no receipts in the system, the system displays a message that there are no receipts to display. |
| Nonfunctional requirements: | N/A |
| Postcondition: | The receipt list is displayed to the receptionist. |
| Outstanding questions: | N/A |

#### 

##### 1.4 Create Receipt

#### 

| UC ID and Name: | UC-15: Create Receipt |
| --- | --- |
| Summary | The receptionist creates a receipt and adds the receipt to the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | The user has logged into the system as a receptionist. |
| Main sequence: | 1. The receptionist accesses the receptionist page. 2. The system displays the receptionist page. 3. The receptionist clicks on the "Receipt" item. 4. The system displays the receipt list. 5. The receptionist clicks on the "Add Receipt" option. 6. The system displays a form to enter receipt information. 7. The receptionist fills in the required information to submit form (e.g., services used, amount, date) 8. The system displays the receipt just created. |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | N/A |
| Postcondition: | If the receipt is created successfully, the recipient will appear in View Receipt List. |
| Outstanding questions: | N/A |

##### 1.5 Edit Receipt

#### 

| UC ID and Name: | UC-16: Edit Receipt |
| --- | --- |
| Summary | The receptionist edit a receipt and update the receipt to the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | The user has logged into the system as a receptionist. |
| Main sequence: | 1.The receptionist accesses the receptionist page.  2.The system displays the receptionist page.  3.The receptionist go to receipt page  4.The system displays the receipt list.  5.The receptionist choose edit receipt options  6.The system displays a form to edit receipt information.  7.The receptionist fills in the required information to submit form (e.g., services used, amount, date)  8.The system displays the receipt just created. |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | N/A |
| Postcondition: | If the receipt is created successfully, the recipient will appear in View Receipt List. |
| Outstanding questions: | N/A |

### 

##### 1.6 Delete Receipt

#### 

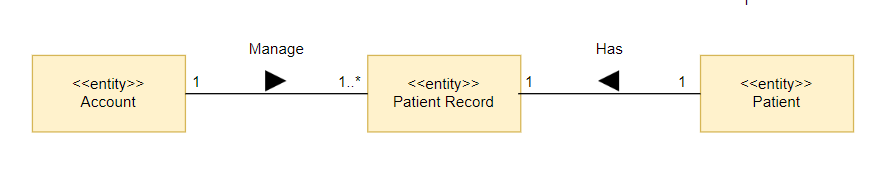
| UC ID and Name: | UC-17: Delete Receipt |
| --- | --- |
| Summary | Receptionist delete a receipt in the system |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist |
| Main sequence: | 1.include UC 14 view receipt list  2.Receptionist clicks “Delete” beside the receipt  3.System show give notice if you are sure you want to delete it  4.Receptionist choose “yes”  5.This receipt will be removed from the system  6.Reload the receipt list |
| Alternative sequences: | Step 3: If the receptionist choose “no”, screen will back to list receipt screen |
| Nonfunctional requirements: | Performance: Delete receipt and reload less than 8 seconds |
| Postcondition: | A receipt is removed from system |
| Outstanding questions: | N/A |

### 

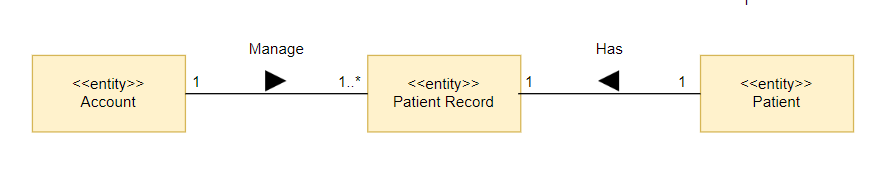
#### 2. Static Modeling

##### 2.1 Entity Class

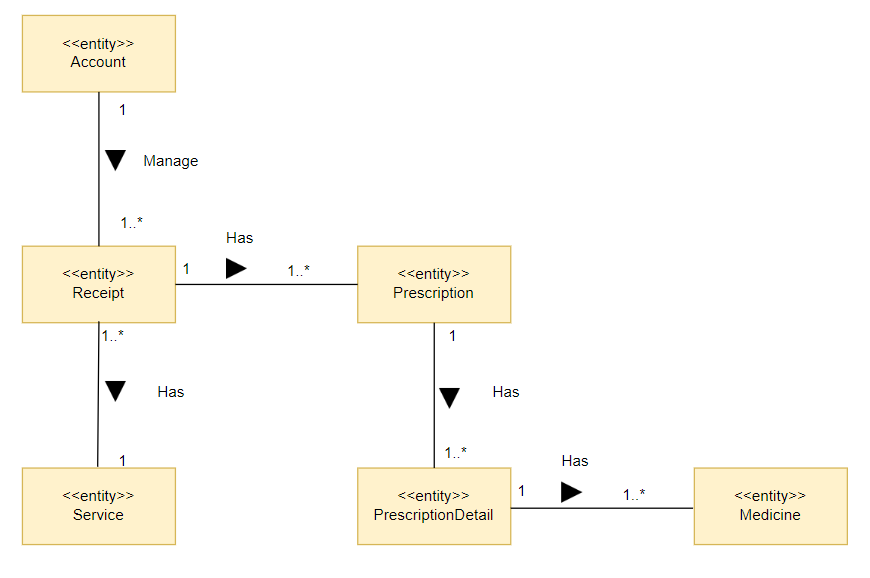
###### 2.1.1 Patient registration



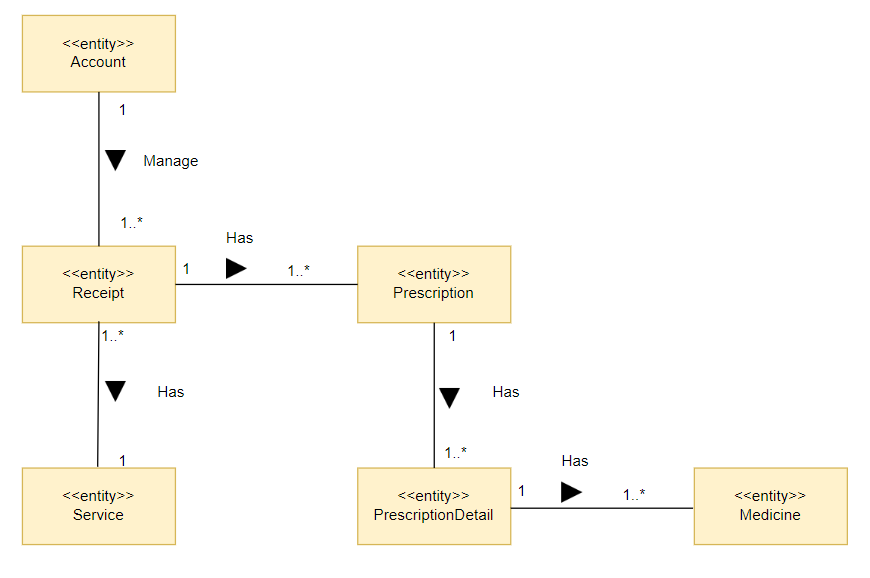
###### 2.1.2 Patient discharge



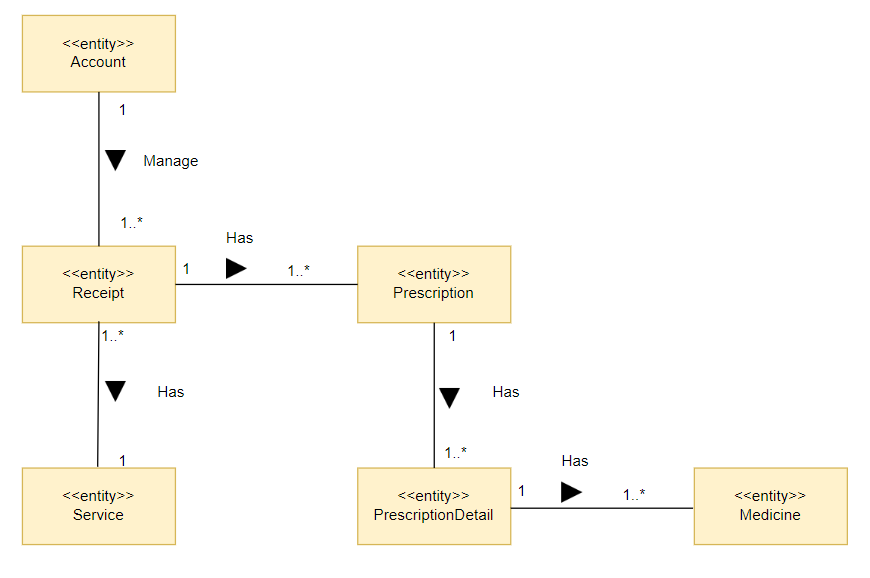
###### 2.1.3 View Receipt List



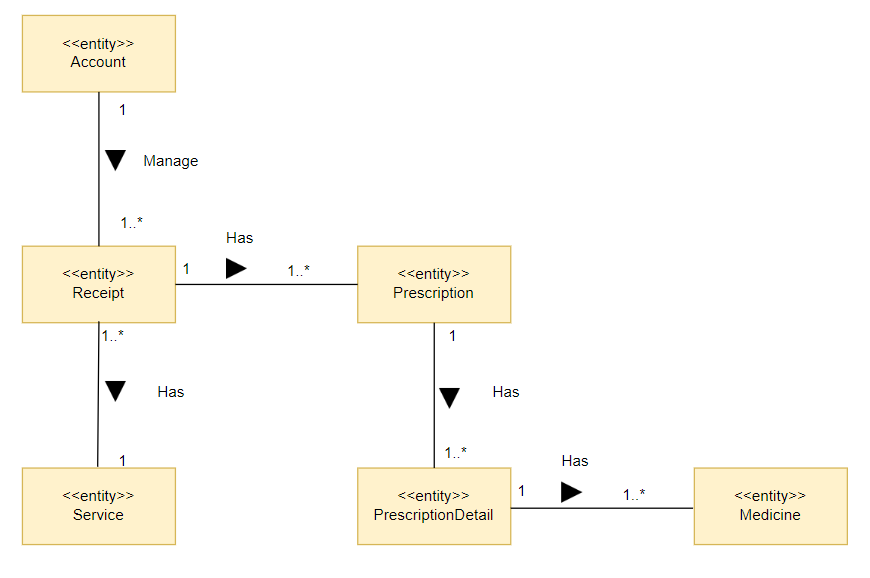
###### 2.1.4 Create Receipt



###### 2.1.5 Edit Receipt



###### 2.1.6 Delete Receipt



##### 

###### 

###### 

###### 

###### 

###### 

#### 3. Dynamic Modeling

##### 3.1 Patient registration

###### 3.1.1 Use Case model

#### 

| UC ID and Name: | UC-12: Patient registration |
| --- | --- |
| Summary | Receptionist adding a patient with that person's personal information into the list of patient in the system |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist * The number of patients in the system must not exceed the maximum allowed number |
| Main sequence: | * 1 Receptionist go to the Patient Registration site * 2 PatientRegistrationPage send information of patient to Receptionistcoordinator * 3 ReceptionistCoordinator send registration patient to ReceptionistService * 4 ReceptionistService send created patient information to ReceptionistCoordinator * 5 ReceptionistCoordinator send "successful" message to PatientRegistrationPage * 6 PatientRegistrationPage redirect receptionist to PatientListPag and display "successfully" message |
| Alternative sequences: | * 4 ReceptionistService check the number of patients has exceeded the number in the system and send error message * 5 ReceptionistCoordinator send error message to PatientRegistrationPage * 6 PatientRegistrationPage redirect receptionist to PatientListPage and display error notification |
| Nonfunctional requirements: | Performance: Check and add patients to the system in less than 10 seconds |
| Postcondition: | 1 patient will be added to the system |

###### *3.1.2 Identify boundaries and internal objects*

* Boundary object: PatientRegistrationPage
* Internal objects: ReceptionistService, ReceptionistCoordinator

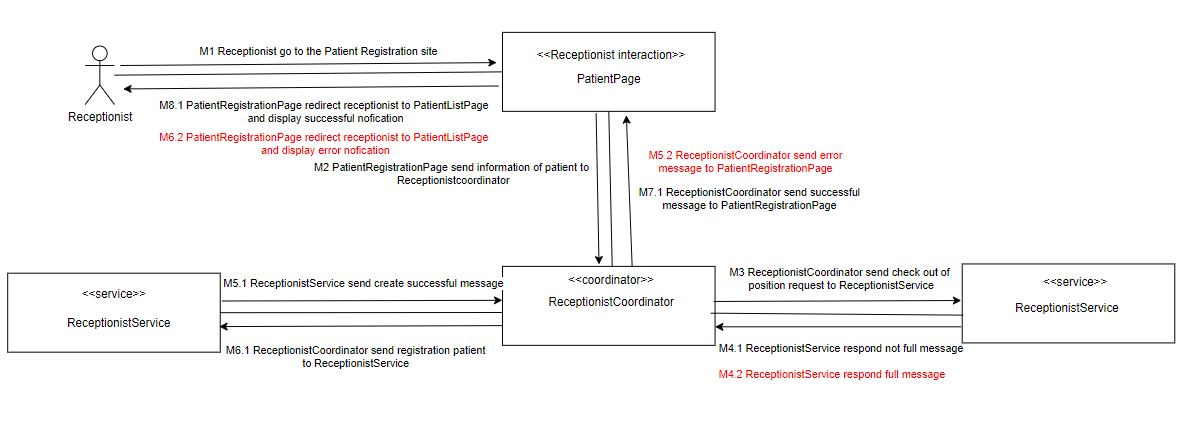
###### 3.1.3 Messages

**Main sequence:**

* M1 Receptionist go to the Patient Registration site
* M2 PatientRegistrationPage send information of patient to Receptionistcoordinator
* M3 ReceptionistCoordinator send check out of position request to ReceptionistService
* M4.1 Respond not full message
* M5.1 ReceptionistCoordinator send registration patient to ReceptionistService
* M6.1 ReceptionistService send created patient information to ReceptionistCoordinator
* M7.1 ReceptionistCoordinator send "successful" message to PatientRegistrationPage
* M8.1 PatientRegistrationPage redirect receptionist to PatientListPag and display "successfully" message

**Alternative sequence:**

* M1 Receptionist go to the Patient Registration site
* M2 PatientRegistrationPage send information of patient to Receptionistcoordinator
* M3 ReceptionistCoordinator send registration patient to ReceptionistService
* M4.2 ReceptionistService check the number of patients has exceeded the number in the system and send error message
* M5.2 ReceptionistCoordinator send error message to PatientRegistrationPage
* M6.2 PatientRegistrationPage redirect receptionist to PatientListPage and display error notification



##### 3.2 Patient Discharge

###### 3.2.1 Use Case model

#### 

| UC ID and Name: | UC-13: Patient Discharge |
| --- | --- |
| Summary | Receptionist remove patient from their bed and change status to discharge |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist * There is at least 1 patient in the system |
| Main sequence: | * 1 Receptionist go to the Patient Registration site * 2 PatientRegistrationPage send information of patient to Receptionistcoordinator * 3 ReceptionistCoordinator send registration patient to ReceptionistService * 4 ReceptionistService send created patient information to ReceptionistCoordinator * 5 ReceptionistCoordinator send "successful" message to PatientRegistrationPage * 6 PatientRegistrationPage redirect receptionist to PatientListPag and display "successfully" message |
| Alternative sequences: | * 4 ReceptionistService check the number of patients has exceeded the number in the system and send error message * 5 ReceptionistCoordinator send error message to PatientRegistrationPage * 6 PatientRegistrationPage redirect receptionist to PatientListPage and display error notification |
| Nonfunctional requirements: | Performance: Remove patients to the list in less than 10 seconds |
| Postcondition: | 1 patient will be removed from the patient list |

###### 3.2.2 Identify boundaries and internal objects

* Boundary object: PatientDischargePage
* Internal objects: ReceptionistService

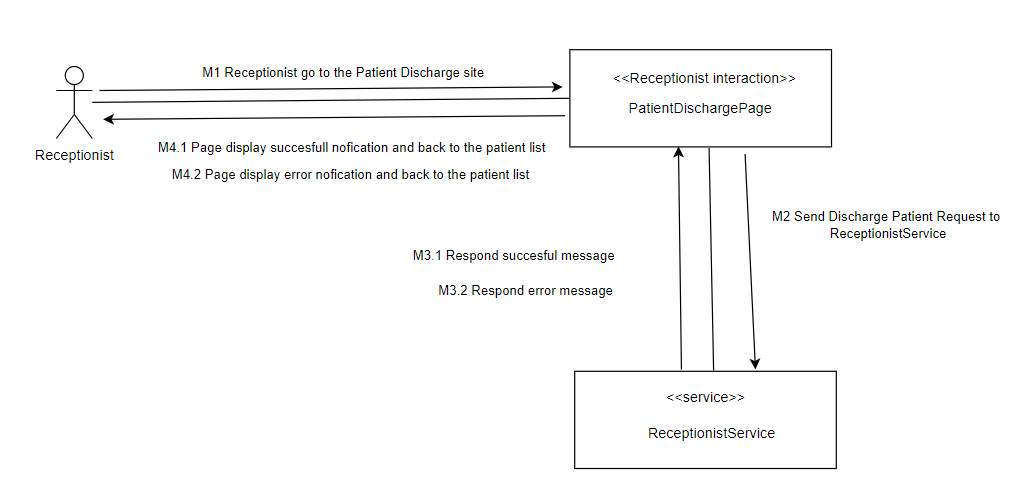
###### 3.2.3 Messages

**Main sequence:**

* M1 Receptionist go to the Patient Registration site
* M2 PatientRegistrationPage send information of patient to Receptionistcoordinator
* M3 ReceptionistCoordinator send check out of position request to ReceptionistService
* M4.1 Respond not full message
* M5.1 ReceptionistCoordinator send registration patient to ReceptionistService
* M6.1 ReceptionistService send created patient information to ReceptionistCoordinator
* M7.1 ReceptionistCoordinator send "successful" message to PatientRegistrationPage
* M8.1 PatientRegistrationPage redirect receptionist to PatientListPag and display "successfully" message

**Alternative sequence:**

* M1 Receptionist go to the Patient Registration site
* M2 PatientRegistrationPage send information of patient to Receptionistcoordinator
* M3 ReceptionistCoordinator send registration patient to ReceptionistService
* M4.2 ReceptionistService check the number of patients has exceeded the number in the system and send error message
* M5.2 ReceptionistCoordinator send error message to PatientRegistrationPage
* M6.2 PatientRegistrationPage redirect receptionist to PatientListPage and display error notification



##### 3.3 View Receipt List

###### 3.3.1 Use Case model

#### 

| UC ID and Name: | UC-14: View Receipt List |
| --- | --- |
| Summary | This use case describes the process of a receptionist viewing a list of receipts in the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * The user has logged into the system as a receptionist. |
| Main sequence: | * 1 Receptionist go to the Receipt page * 2 ReceptionistPage send view receipt request to ReceptionistService * 3 ReceptionistService send receipt information to ReceptionistPage * 4 PatientRegistrationPage redirect receptionist to Receipt list and display all receipt |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | Performance: Display all receipt in less than 5 seconds |
| Postcondition: | all receipt display in screen |

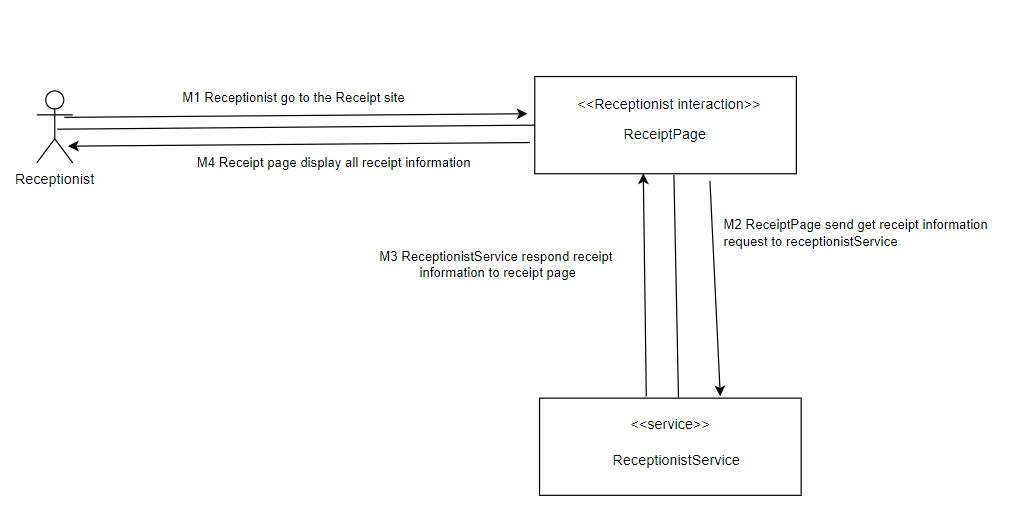
###### 3.3.2 Identify boundaries and internal objects

* Boundary object: ReceiptPage
* Internal objects: ReceptionistService

###### 3.3.3 Messages

**Main sequence:**

* M1 Receptionist go to the Receipt page
* M2 ReceptionistPage send view receipt request to ReceptionistService
* M3 ReceptionistService send receipt information to ReceptionistPage
* M4 PatientRegistrationPage redirect receptionist to Receipt list and display all receipt



##### 

##### 3.4 Create Receipt

###### 3.4.1 Use Case model

#### 

| UC ID and Name: | UC-15: Create Receipt List |
| --- | --- |
| Summary | This use case describes the process of a receptionist viewing a list of receipts in the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * The user has logged into the system as a receptionist. |
| Main sequence: | * 1 Receptionist go to the Receipt page * 2 ReceptionistPage send view receipt request to ReceptionistService * 3 ReceptionistService send receipt information to ReceptionistPage * 4 PatientRegistrationPage redirect receptionist to Receipt list and display all receipt |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | Performance: Display all receipt in less than 5 seconds |
| Postcondition: | all receipt display in screen |

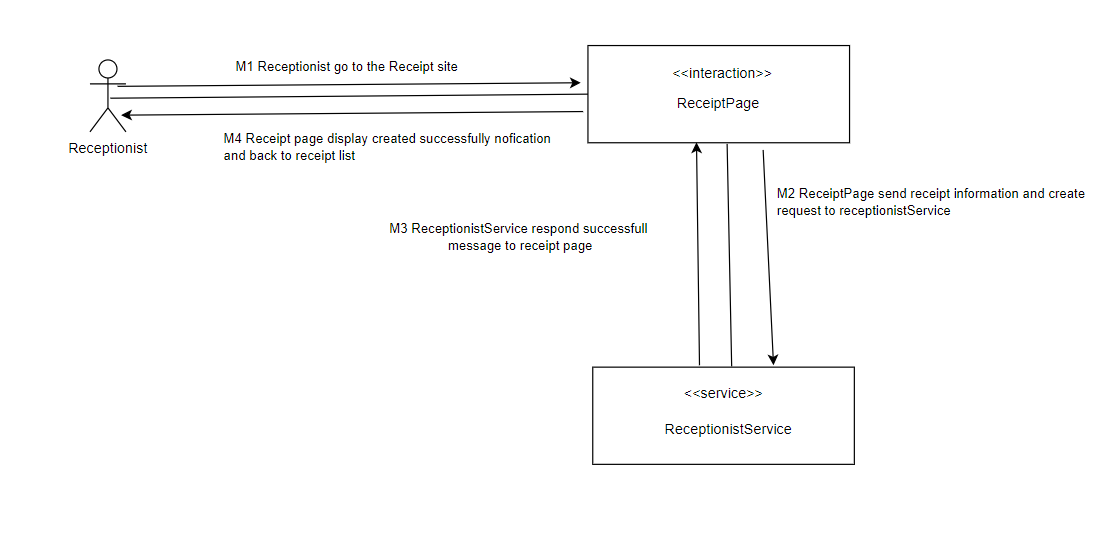
###### 3.4.2 Identify boundaries and internal objects

* Boundary object: ReceiptPage
* Internal objects: ReceptionistService

###### 3.4.3 Messages

**Main sequence:**

* M1 Receptionist go to the Receipt page
* M2 ReceptionistPage send view receipt request to ReceptionistService
* M3 ReceptionistService send receipt information to ReceptionistPage
* M4 PatientRegistrationPage redirect receptionist to Receipt list and display all receipt



##### 3.5 Edit Receipt

###### 3.5.1 Use Case model

#### 

| UC ID and Name: | UC-16: Edit Receipt List |
| --- | --- |
| Summary | This use case describes the process of a receptionist viewing a list of receipts in the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * The user has logged into the system as a receptionist. |
| Main sequence: | * 1 Receptionist go to the Receipt page * 2 ReceptionistPage send view receipt request to ReceptionistService * 3 ReceptionistService send receipt information to ReceptionistPage * 4 PatientRegistrationPage redirect receptionist to Receipt list and display all receipt |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | Performance: Display all receipt in less than 5 seconds |
| Postcondition: | all receipt display in screen |

###### 3.5.2 Identify boundaries and internal objects

* Boundary object: ReceiptPage
* Internal objects: ReceptionistService

###### 3.5.3 Messages

**Main sequence:**

* M1 Receptionist go to the Receipt page
* M2 ReceptionistPage send view receipt request to ReceptionistService
* M3 ReceptionistService send receipt information to ReceptionistPage
* M4 PatientRegistrationPage redirect receptionist to Receipt list and display all receipt



##### 3.6 Delete Receipt

###### 3.6.1 Use Case model

#### 

| UC ID and Name: | UC-17: Delet Receipt List |
| --- | --- |
| Summary | This use case describes the process of a receptionist viewing a list of receipts in the system. |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * The user has logged into the system as a receptionist. |
| Main sequence: | * 1 Receptionist go to the Receipt page * 2 ReceptionistPage send view receipt request to ReceptionistService * 3 ReceptionistService send receipt information to ReceptionistPage * 4 PatientRegistrationPage redirect receptionist to Receipt list and display all receipt |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | Performance: Display all receipt in less than 5 seconds |
| Postcondition: | all receipt display in screen |

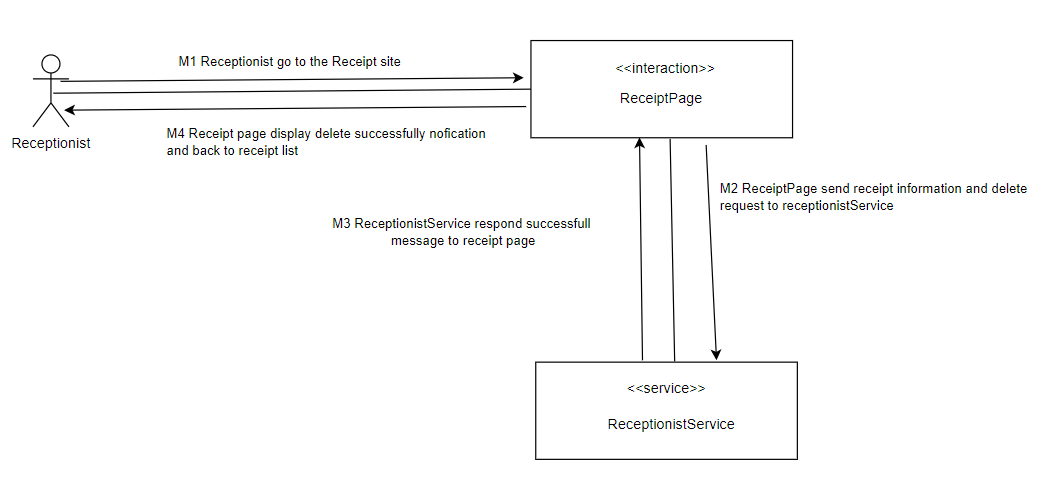
###### 3.6.2 Identify boundaries and internal objects

* Boundary object: ReceiptPage
* Internal objects: ReceptionistService

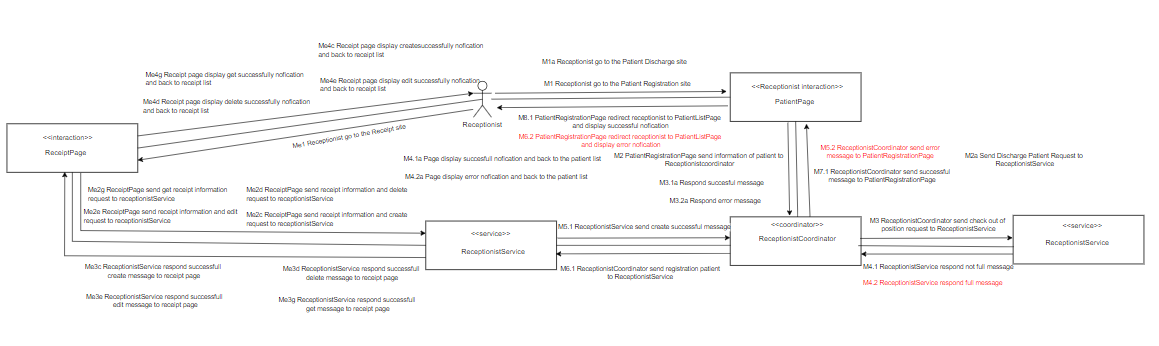
###### 3.6.3 Messages

**Main sequence:**

* M1 Receptionist go to the Receipt page
* M2 ReceptionistPage send view receipt request to ReceptionistService
* M3 ReceptionistService send receipt information to ReceptionistPage
* M4 PatientRegistrationPage redirect receptionist to Receipt list and display all receipt



#### 4. Architecture Design



#### 

#### 5. Detail Design

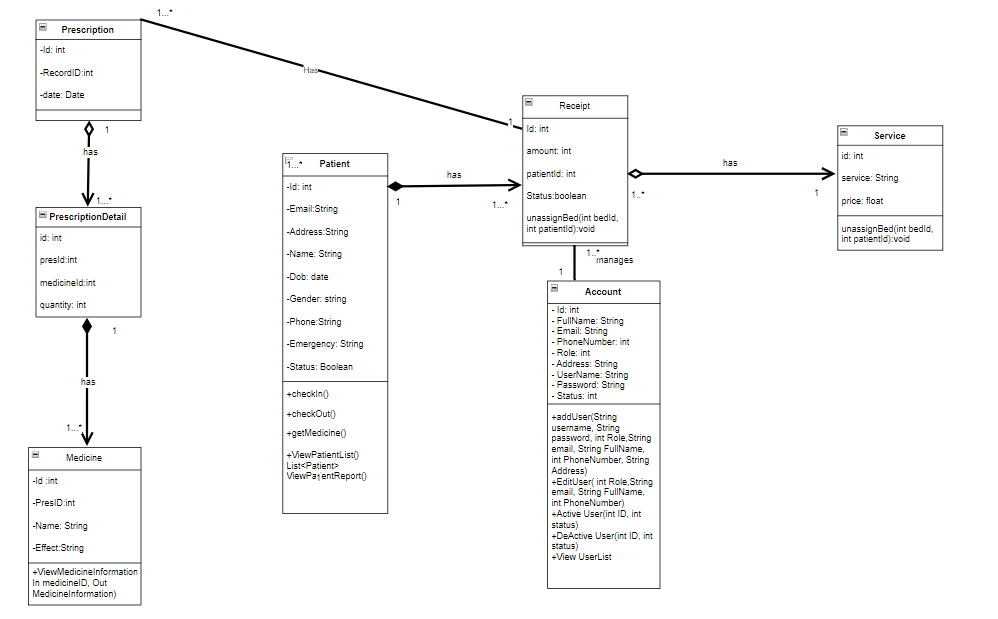
##### 5.1 Class Diagram

###### 5.1.1 Patient Registration

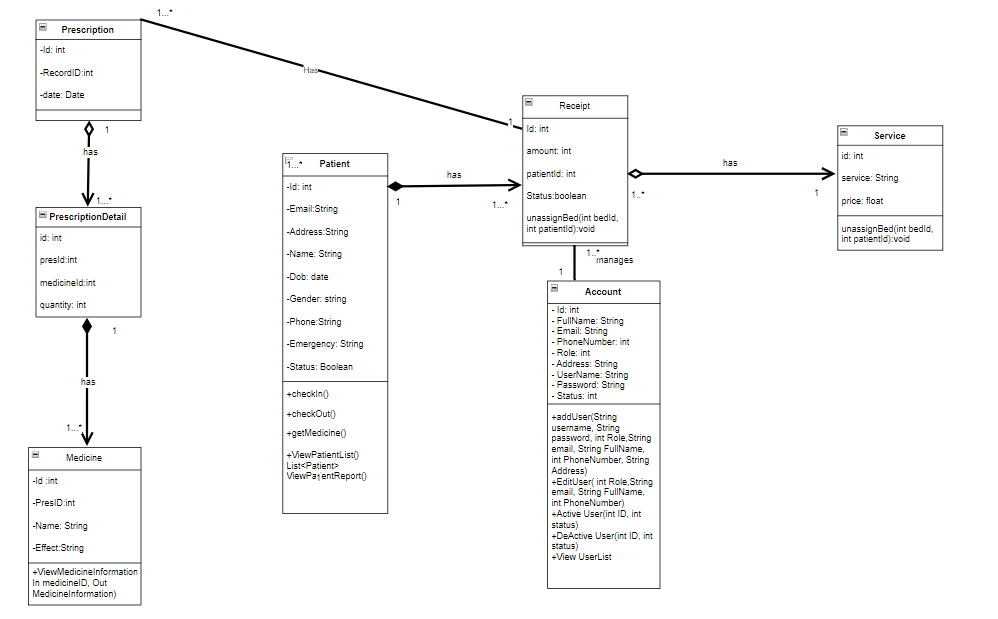
###### 5.1.2 Patient Discharge

###### 

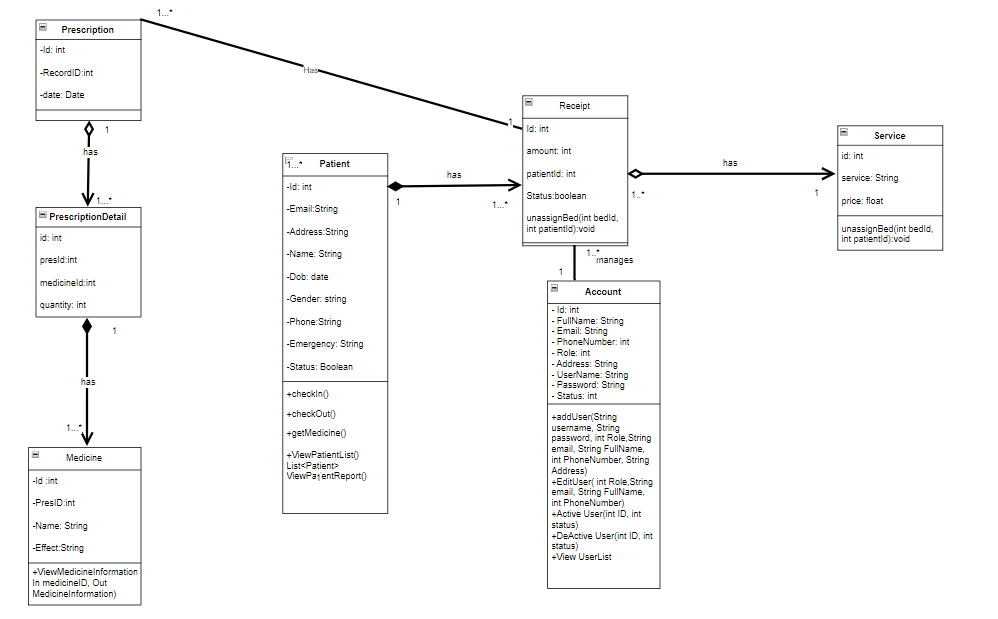
###### 5.1.3 View Receipt



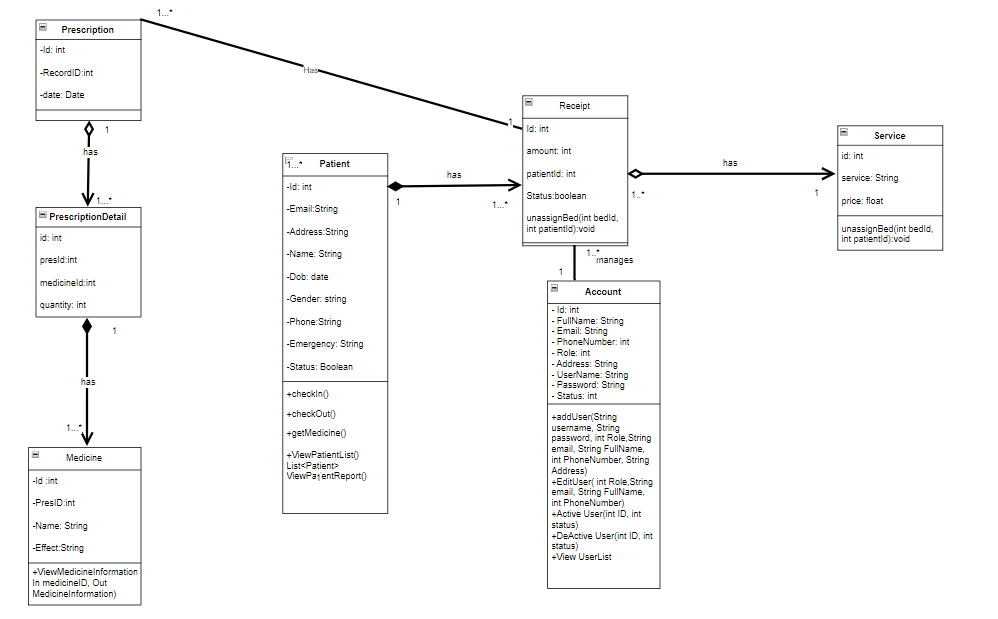
###### 5.1.4 Create Receipt



###### 5.1.5 Edit Receipt

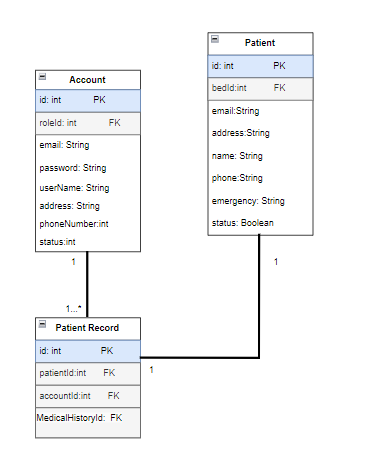


###### 5.1.6 Delete Receipt

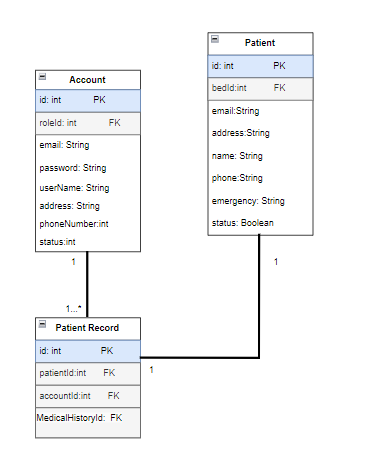


##### 5.2 Relational Database

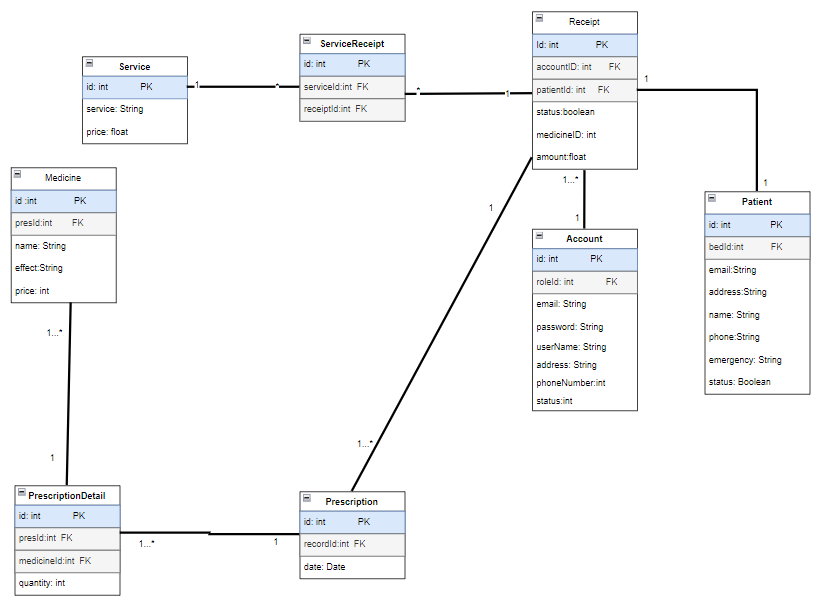
###### 5.2.1 Patient Registration



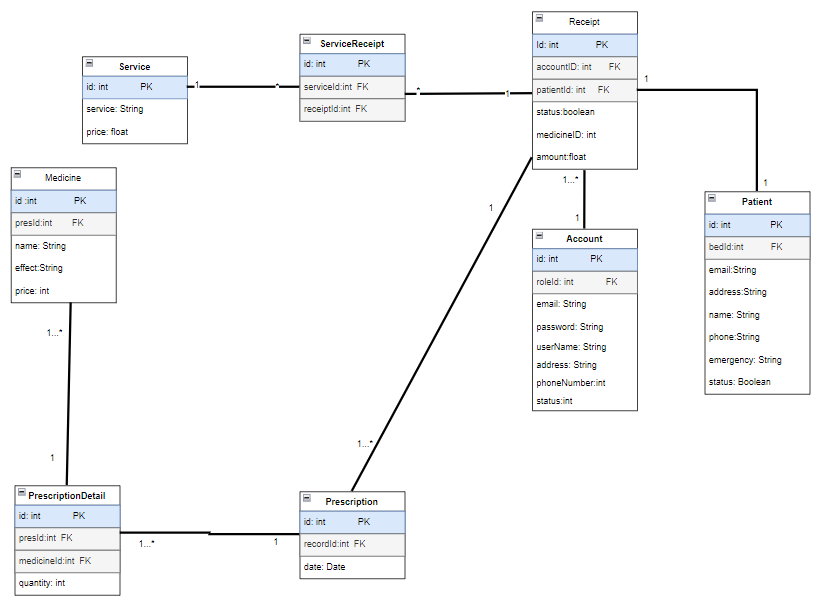
###### 5.2.2 Patient Discharge



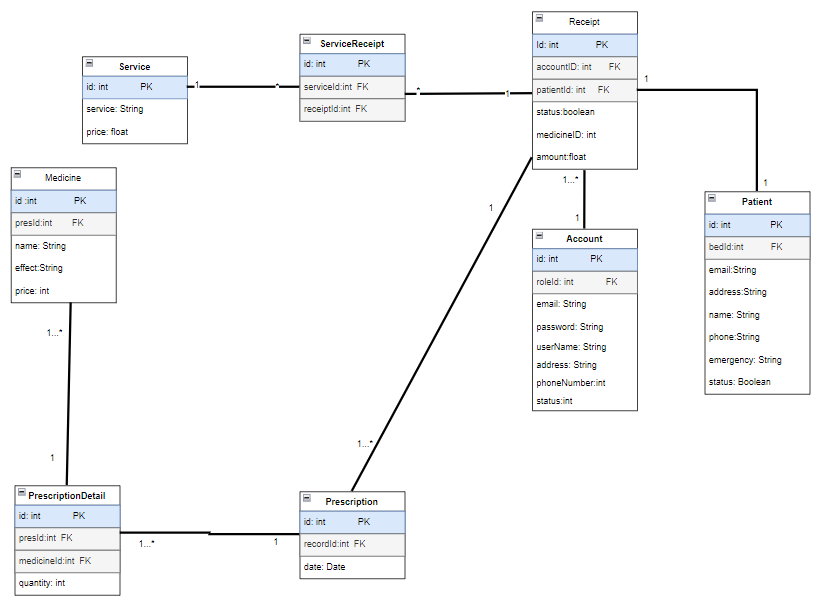
###### 5.2.3 View Receipt List



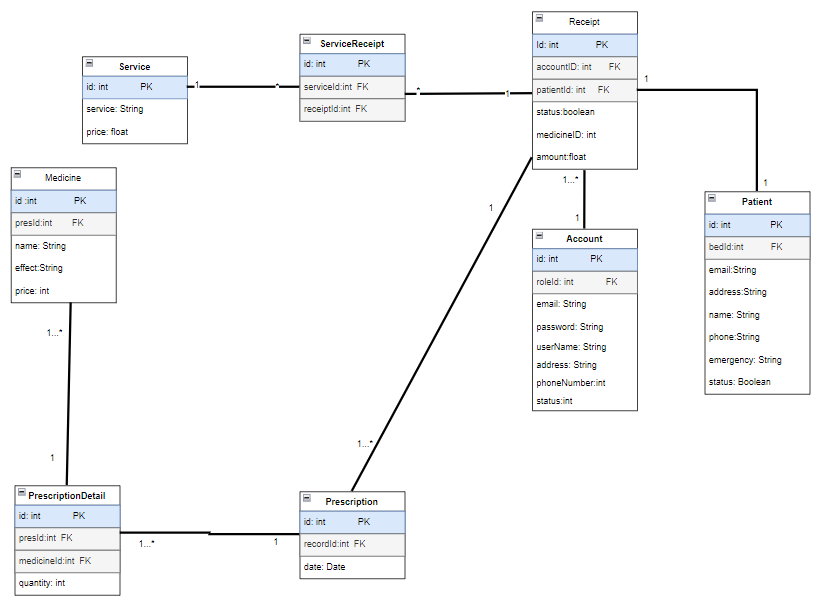
###### 5.2.4 Create Receipt



###### 5.2.5 Edit Receipt



###### 5.2.6 Delete Receipt



##### 5.3 Pseudocode

###### 5.3.1 Patient registration

* Attributes: patientId, bedId, Email, address, name, phone, emergency,status
* Operations:
* patientRegistration(In PatientInformation, Out bool)

**BEGIN**

// Validate the information

**IF** Name is empty **THEN**

**PRINT** "Name is required"

**RETURN** false

**END IF**

**IF** Gender is empty **THEN**

**PRINT** "Gender is required"

**RETURN** false

**END IF**

**IF** Status is empty **THEN**

**PRINT** "Status is required"

**RETURN** false

**END IF**

**IF** Phone is empty **THEN**

**PRINT** "Phone is required"

**RETURN** false

**END IF**

**IF** Emergency is empty **THEN**

**PRINT** "Emergency is required"

**RETURN** false

**END IF**

**IF** email format is invalid **THEN**

**PRINT** "Invalid email format"

**RETURN** false

**END IF**

**IF** email exists in database **THEN**

**PRINT** "Email already exists"

**RETURN** false

**END IF**

**IF** phone exists in database **THEN**

**PRINT** "Phone already exists"

**RETURN** false

**END IF**

// Check if there are available positions

**IF** checkOutOfPosition(Out bool) **THEN**

// **SAVE** Patient

**SAVE** savePatient(Name, Gender, Address, email, status, phone, emergency)

**PRINT** "Registration successfully."

**RETURN** true

**ELSE**

**PRINT** "Out of position."

**RETURN** false

**END IF**

**END**

###### 5.3.2 Patient Discharge

* Attributes: patientId, bedId, Email, address, name, phone, emergency,status
* Operation:

+dischargePatientById(IN id,OUT bool)

BEGIN

// Validate the id

IF id is empty THEN

PRINT "Patient ID is required"

RETURN

END IF

// Try to delete the patient record from the database

TRY

DELETE FROM database WHERE patientId = id

// Check if the deletion was successful

IF affectedRows > 0 THEN

PRINT "Patient record successfully deleted"

RETURN True

ELSE

PRINT "No patient found with the provided ID"

RETURN False

END IF

CATCH databaseException

PRINT "An error occurred while trying to delete the patient record"

END TRY

END

+getPatientByID(IN patientID, OUT patient)

BEGIN

IF patientID is empty THEN

PRINT "Patient ID is required";

RETURN;

END IF;

patient = query database for patient with patientID;

IF patient is NULL THEN

PRINT "Patient not found";

END IF;

RETURN patient;

END

###### 5.3.3 View Receipt

BEGIN

// Validate the receiptId

IF receiptId is empty THEN

PRINT "Receipt ID is required"

RETURN

END IF

// Query the database for the receipt with the provided receiptId

receipt = query database for receipt with receiptId

// Check if the receipt was found

IF receipt is NULL THEN

PRINT "Receipt not found"

END IF

RETURN receipt

END

###### 5.3.4 Create receipt

BEGIN

// Validate the required fields

IF patientId is empty OR amount is empty OR date is empty OR items is empty THEN

PRINT "All fields are required"

RETURN False

END IF

// Insert the new receipt into the database

TRY

INSERT INTO database (patientId, amount, date, items, status) VALUES (patientId, amount, date, items, 'active')

// Check if the insertion was successful

IF affectedRows > 0 THEN

PRINT "Receipt successfully created"

RETURN True

ELSE

PRINT "Failed to create receipt"

RETURN False

END IF

CATCH databaseException

PRINT "An error occurred while trying to create the receipt"

RETURN False

END TRY

END

###### 5.3.5 Edit receipt

BEGIN

// Validate the receiptId

IF receiptId is empty THEN

PRINT "Receipt ID is required"

RETURN False

END IF

// Update the receipt in the database

TRY

UPDATE database SET patientId = patientId, amount = amount, date = date, items = items WHERE receiptId = receiptId

// Check if the update was successful

IF affectedRows > 0 THEN

PRINT "Receipt successfully updated"

RETURN True

ELSE

PRINT "No receipt found with the provided ID"

RETURN False

END IF

CATCH databaseException

PRINT "An error occurred while trying to update the receipt"

RETURN False

END TRY

END

###### 

###### 5.3.6 Delete Receipt

BEGIN

// Validate the receiptId

IF receiptId is empty THEN

PRINT "Receipt ID is required"

RETURN False

END IF

// Try to delete the receipt from the database

TRY

DELETE FROM database WHERE receiptId = receiptId

// Check if the deletion was successful

IF affectedRows > 0 THEN

PRINT "Receipt successfully deleted"

RETURN True

ELSE

PRINT "No receipt found with the provided ID"

RETURN False

END IF

CATCH databaseException

PRINT "An error occurred while trying to delete the receipt"

RETURN False

END TRY

END

#### 6. Change requirement (UC - 12 Patient registration)

##### 6.0 Purpose

* Initial requirement: When the number of patients exceeds the number of beds and rooms, an error message will be displayed and patients cannot be added.
* It is possible that during an epidemic or disaster the number of patients suddenly increases and the number of patients is required to exceed the prescribed number, so they will be registered as additional patients if the number of patients exceeds the prescribed number. exceed the original allowable quantity

##### 6.1 Requirements

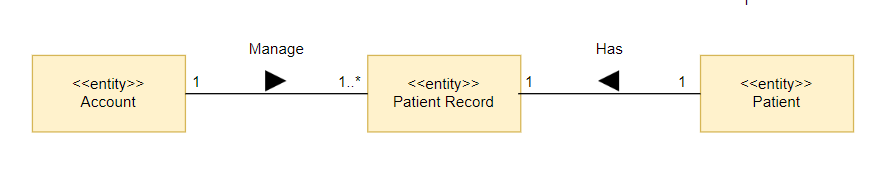
#### 

| UC ID and Name: | UC-12: Patient registration |
| --- | --- |
| Summary | Receptionist adding a patient with that person's personal information into the list of patient in the system |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist * The number of patients in the system must not exceed the maximum allowed number |
| Main sequence: | 1. Receptionist login to website. 2. Receptionist go to the Patient Registration site. 3. System show patient information form 4. Receptionist fills in patient information 5. Receptionist select create new patient option 6. The system adds 1 patient with the same information entered as above to the list of patients and adds it to the system. |
| Alternative sequences: | Step 6: The system checks that the number of patients in the system has exceeded the allowed number, the screen displays a notification that the hospital has reached the maximum number of patients allowed and creates additional patients. |
| Nonfunctional requirements: | Performance: Check and add patients to the system in less than 10 seconds |
| Postcondition: | 1 patient will be added to the system |
| Outstanding questions: | N/A |

##### 

##### 6.2 Static modeling

Entity Class



##### 6.3 Dynamic modeling

###### 6.3.1 Use Case model

#### 

| UC ID and Name: | UC-12: Patient registration |
| --- | --- |
| Summary | Receptionist adding a patient with that person's personal information into the list of patient in the system |
| Dependency: | N/A |
| Actors: | Receptionist |
| Preconditions | * User has logged in as Receptionist |
| Main sequence: | * 1 Receptionist go to the Patient Registration site * 2 PatientRegistrationPage send information of patient to Receptionistcoordinator * 3 ReceptionistCoordinator send check out of position request to ReceptionistService * M4.1 Respond not full message * M5.1 ReceptionistCoordinator send registration patient to ReceptionistService * M6.1 ReceptionistService send created patient information to ReceptionistCoordinator * M7.1 ReceptionistCoordinator send "successful" message to PatientRegistrationPage * M8.1 PatientRegistrationPage redirect receptionist to PatientListPag and display "successfully" message |
| Alternative sequences: | * M4.2 ReceptionistService check the number of patients has exceeded the number in the system and send exceed message * M5.2 ReceptionistCoordinator send registration additional patient to ReceptionistService * M6.2 ReceptionistService send created additional patient information to ReceptionistCoordinator * M7.2 ReceptionistCoordinator send "successful" message to PatientRegistrationPage * M8.2 PatientRegistrationPage redirect receptionist to PatientList and display "successfully" message |
| Nonfunctional requirements: | Performance: Check and add patients to the system in less than 10 seconds |
| Postcondition: | 1 patient will be added to the system |

###### 6.3.2 Identify boundaries and internal objects

* Boundary object: PatientRegistrationPage
* Internal objects: ReceptionistService, ReceptionistCoordinator

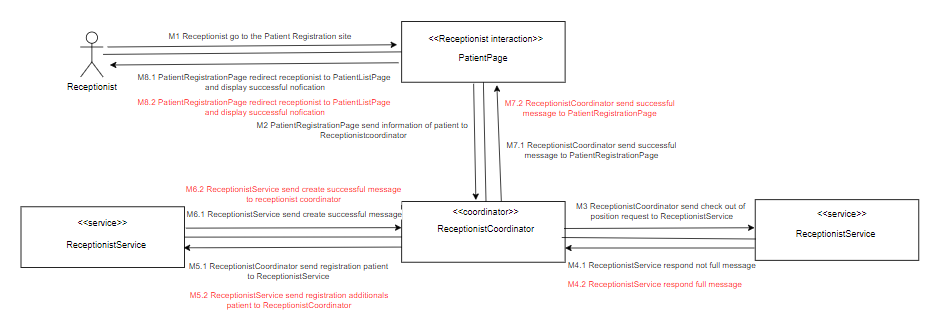
###### 6.3.3 Messages

**Main sequence:**

* M1 Receptionist go to the Patient Registration site
* M2 PatientRegistrationPage send information of patient to Receptionistcoordinator
* M3 ReceptionistCoordinator send check out of position request to ReceptionistService
* M4.1 Respond not full message
* M5.1 ReceptionistCoordinator send registration patient to ReceptionistService
* M6.1 ReceptionistService send created patient information to ReceptionistCoordinator
* M7.1 ReceptionistCoordinator send "successful" message to PatientRegistrationPage
* M8.1 PatientRegistrationPage redirect receptionist to PatientListPag and display "successfully" message

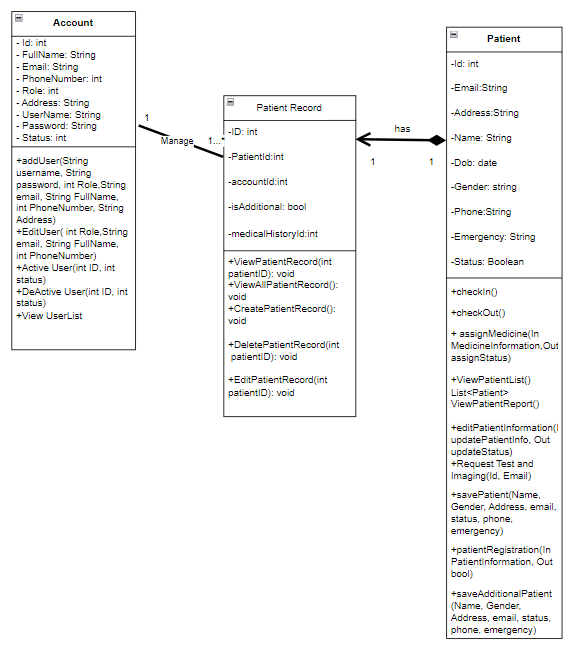
**Alternative sequence:**

* M1 Receptionist go to the Patient Registration site
* M2 PatientRegistrationPage send information of patient to Receptionistcoordinator
* M3 ReceptionistCoordinator send registration patient to ReceptionistService
* M4.2 ReceptionistService check the number of patients has exceeded the number in the system and send exceed message
* M5.2 ReceptionistCoordinator send registration additional patient to ReceptionistService
* M6.2 ReceptionistService send created additional patient information to ReceptionistCoordinator
* M7.2 ReceptionistCoordinator send "successful" message to PatientRegistrationPage
* M8.2 PatientRegistrationPage redirect receptionist to PatientList and display "successfully" message

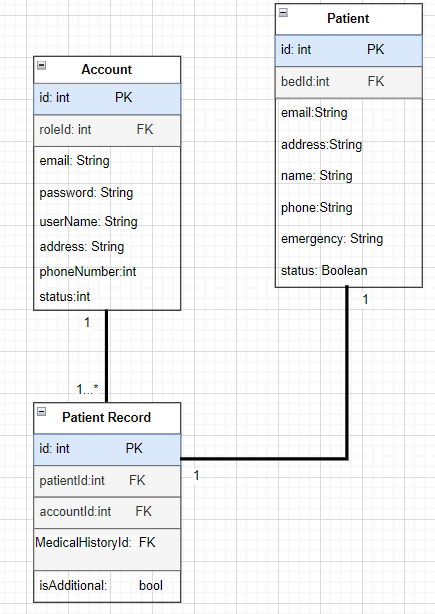


##### 6.4 Detail Design

###### 6.4.1 Class Diagram



###### 6.4.2 Relational Diagram



###### 6.4.3 Pseudocode

* Attributes: patientId, bedId, Email, address, name, phone, emergency,status
* Operations:
* patientRegistration(In PatientInformation, Out bool)

**BEGIN**

// Validate the information

**IF** Name is empty **THEN**

**PRINT** "Name is required"

**RETURN** false

**END IF**

**IF** Gender is empty **THEN**

**PRINT** "Gender is required"

**RETURN** false

**END IF**

**IF** Status is empty **THEN**

**PRINT** "Status is required"

**RETURN** false

**END IF**

**IF** Phone is empty **THEN**

**PRINT** "Phone is required"

**RETURN** false

**END IF**

**IF** Emergency is empty **THEN**

**PRINT** "Emergency is required"

**RETURN** false

**END IF**

**IF** email format is invalid **THEN**

**PRINT** "Invalid email format"

**RETURN** false

**END IF**

**IF** email exists in database **THEN**

**PRINT** "Email already exists"

**RETURN** false

**END IF**

**IF** phone exists in database **THEN**

**PRINT** "Phone already exists"

**RETURN** false

**END IF**

// Check if there are available positions

**IF** checkOutOfPosition(Out bool) **THEN**

// **SAVE** Patient

**SAVE** savePatient(Name, Gender, Address, email, status, phone, emergency)

**PRINT** "Registration successfully."

**RETURN** true

**ELSE**

// **SAVE** Patient

**SAVE** saveAdditionalPatient(Name, Gender, Address, email, status, phone, emergency)

**PRINT** "Registration additional patient successfully."

**RETURN** true

**END IF**

**END**

#### 7. Layers of Abstraction Pattern

###### 7.1 Patient Registration

| Presentation Layer | patient.jsp, login.jsp |
| --- | --- |
| Coordinator Layer | LoginController.java,  PatientRegistrationController.java |
| Data Access Layer | PatientDAO.java,  AccountDAO.java |

###### 7.2 Patient Discharge

| Presentation Layer | patient.jsp, login.jsp |
| --- | --- |
| Coordinator Layer | LoginController.java,  PatientRegistrationController.java |
| Data Access Layer | PatientDAO.java,  AccountDAO.java |

###### 7.3 View Receipt

| Presentation Layer | receipt.jsp, login.jsp |
| --- | --- |
| Coordinator Layer | LoginController.java,  ReceiptController.java |
| Data Access Layer | ReceiptDAO.java,  AccountDAO.java |

###### 

###### 7.4 Create Receipt

| Presentation Layer | receipt.jsp, login.jsp |
| --- | --- |
| Coordinator Layer | LoginController.java,  ReceiptCreateController.java |
| Data Access Layer | ReceiptDAO.java,  AccountDAO.java |

###### 7.5 Edit Receipt

| Presentation Layer | receipt.jsp, login.jsp |
| --- | --- |
| Coordinator Layer | LoginController.java,  ReceiptEditController.java |
| Data Access Layer | ReceiptDAO.java,  AccountDAO.java |

###### 7.6 Delete Receipt

| Presentation Layer | receipt.jsp, login.jsp |
| --- | --- |
| Coordinator Layer | LoginController.java,  ReceiptDeleteController.java |
| Data Access Layer | ReceiptDAO.java,  AccountDAO.java |

###### 

### E.Trình Ngọc Tuân (HE163211)

#### 1. Requirement

[4.7: UserList](#_irxen3e1qd95)

[4.8: Active/Deactive User](#_wcoth9x6chsk)

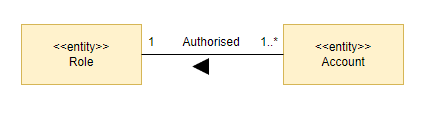
[4.9: Edit User](#_6ih1z6pj5cza)

[4.19: Add NewUser](#_gzzn1dhclz47)

[4.20: View UserDetail](#_7eu49dk3zjto)

#### 2. Static Modeling

##### 2.1 Entity Class



##### 

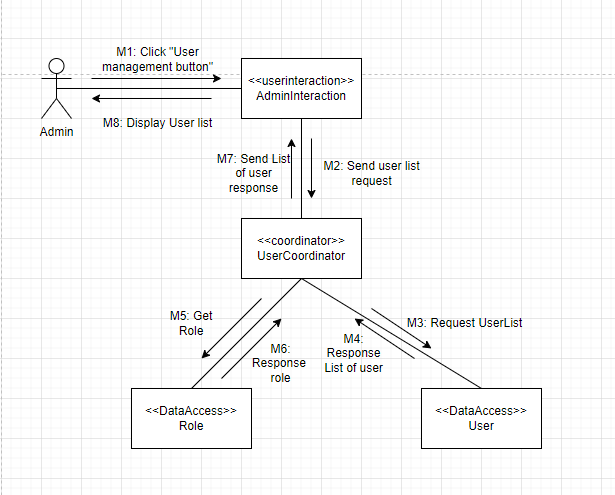
#### 3. Dynamic Modeling

##### 3.1 *UserList*

### 

| UC ID and Name: | UC-07: View User List |
| --- | --- |
| Summary | This use case helps the admin view the list of users in the system. |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | Admin has logged in |
| Main sequence: | 1.Admin login into the system successfully.  2.User clicks the “User Management” menu.  3.System shows user list. |
| Alternative sequences: | N/A |
| Nonfunctional requirements: | **Performance**: List users will be displayed after 3 seconds |
| Postcondition: | Admin can view the list of users. |

##### **a.communication diagram**



##### **b.Identify boundary and internal objects**

* Boundary objects: AdminInteration
* Internal objects:UserCoordinator

##### **c.Message**

M1: Click "User management button"

M2: Send user list request

M3: Request UserList

M4: Response List of user

M5: Get Role

M6: Response role

M7: Send List of user response

M8: Display User list

##### **d.Alternative sequence**

N/A

##### 3.2 *Active/Deactive User*

### 

| UC ID and Name: | UC-08: Active/Deactive User |
| --- | --- |
| Summary | This use case allows admin to Activate/Deactivate user. |
| Actors: | Admin |
| Preconditions | User has logged in as Admin |
| Main sequence: | 1.Admin login into the system successfully.  2.User clicks the “User Management” menu.  3.System shows user list.  4.User click on ban/unban-visibility button  5.User click icon Active/Deactivate button.  6.System show dialog to user confirm  7.System sets user’s status to Active/Deactivate |
| Alternative sequences: | N/A |
| Postcondition: | User account is activated/deactivated status |

##### **a.communication diagram**



##### **b.Identify boundary and internal objects**

* Boundary objects: AdminInteration
* Internal objects:UserCoordinator

##### **c.Message**

* M1:Request UserInformation
* M2:Request User status
* M3:Request Edit User status
* M4:Return new user status.
* M5:New user status.
* M6:Display new status of user

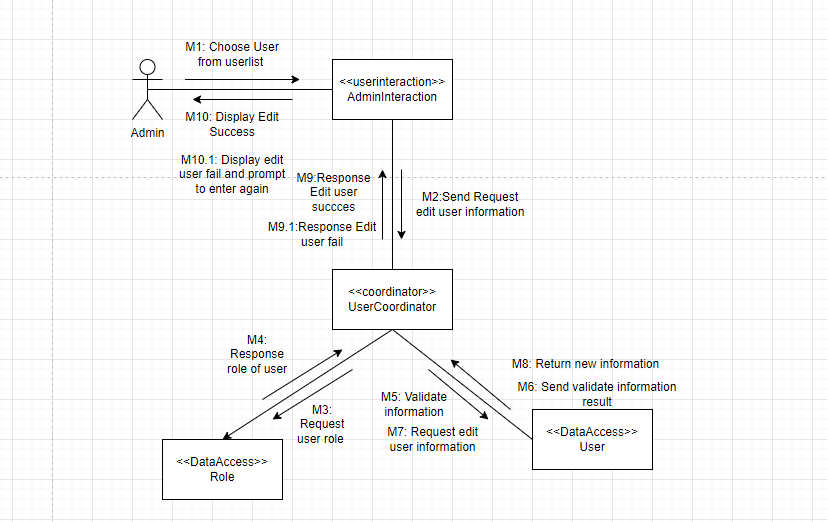
##### **d.Alternative sequence**

##### `3.3 *Edit User*

### 

| UC ID and Name: | UC-09: Edit User |
| --- | --- |
| Summary | This use case allows admin to edit user |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | User has logged in as Admin. |
| Main sequence: | 1.Admin login into the system successfully.  2.User clicks the “User Management” menu.  3.System shows user list and Admin choose user from list.  4.User click the “Edit” icon in the user list.  5.System shows Edit user information fields to screen.  6.User change information and click submit.  7.System save the changes and display edit success . |
| Alternative sequences: | Step 6: Users fill invalid information into fields ,an error message is displayed, and the user is prompted to re-enter their information. |
| Postcondition: | User account is updated |

##### **a.communication diagram**



##### **b.Identify boundary and internal objects**

* Boundary objects: AdminInteration
* Internal objects:UserCoordinator

##### **c.Message**

* M1: Choose User from list
* M2: Request user information
* M3: Request user role
* M4: Response role of user
* M5: Request edit user information
* M6: Validate information
* M7: Return Valid information
* M8: Edit user succces
* M9: Display Edit Success

##### **d.Alternative sequence**

* M7.1: Return InValid information
* M8.1: Edit user fail
* M9.1: Display edit user fail and prompt to enter again

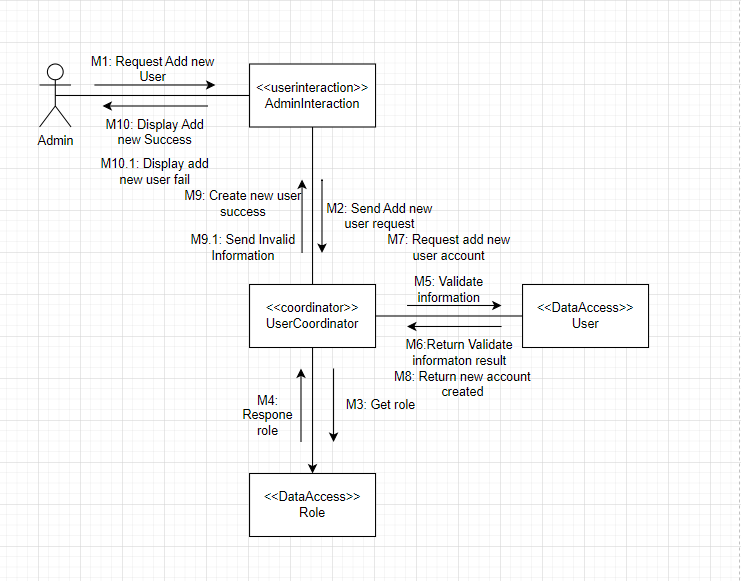
##### 3.5 *Add New User*

#### 

| UC ID and Name: | UC-19: Add New User |
| --- | --- |
| Summary | This use case allows admin to create new user |
| Dependency: | **N/A** |
| Actors: | Admin |
| Preconditions | User has logged in as Admin |
| Main sequence: | 1.Admin login in the system successfully  2.User clicks “User Management” menu  3.System shows userlist  4.User click icon “Add” in the User list  5.System show Create user screen  6.User enter information and click submit.  7.System create new user account and send email to user |
| Alternative sequences | Step 6:User invalid information into fields ,an error message is displayed, and the user is prompted to re-enter their information |
| Nonfunctional requirements: | **Performance**: The system can continuously created 100 patients in 1 minute |
| Postcondition: | New account is created |
| Outstanding questions: | N/A |

#### 

##### **a.communication diagram**



##### **b.Identify boundary and internal objects**

* Boundary objects: AdminInteration
* Internal objects:UserCoordinator

##### **c.Message**

M1: Request Add new User

M2: Send add new user request

M3: Get role to assign new user

M4: Respone role

M5: Validate information

M6: Check valid information

M7: If valid information ,request add new user account

M8: Send create new user success message

M9: Display Add new Success

##### **d.Alternative sequence**

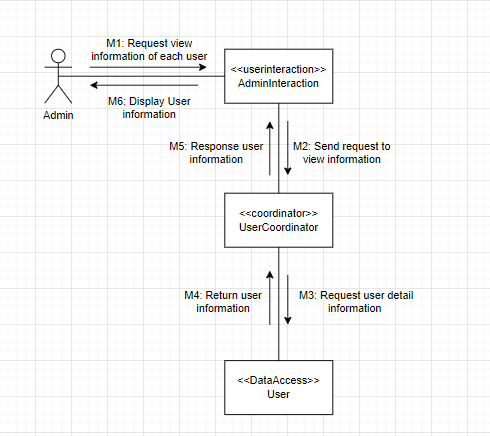
M6.1: If invalid information,cannot create new user account

M8.1: Send Invalid Information

M9.1: Display add new user fail

##### 3.6 *View User Detail*

##### **a.communication diagram**



##### **b.Identify boundary and internal objects**

* Boundary objects: AdminInteration
* Internal objects:UserCoordinator

##### **c.Message**

M1: Request view information of each user

M2: Send request to view information

M3: Request user detail information

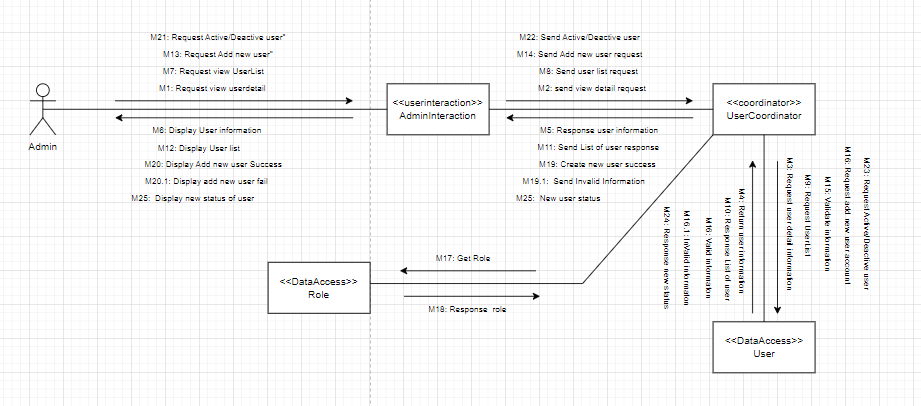
M4: Return user information

M5: Response user information

M6: Display User information

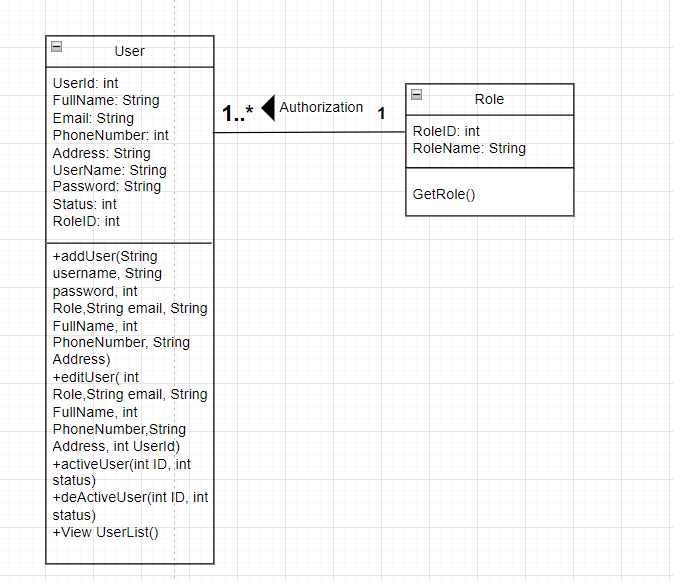
##### **d.Alternative sequence**

#### 4. Architecture Design



#### 5. Detail Design

##### 5.1.Class Diagram



##### 5.2.Relational Database



##### 5.3.Pseudocode

User:

* Attribute: Id,FullName,Emai,lPhoneNumber,Address,UserName,Password,Status,RoleID
* Operation:

+Edit User(In userId, In NewUserÌnormation,Out Display)

**BEGIN**

foundUser <- NULL

**FOR EACH**

user IN userList DO

**IF** user.Id == In UserID **THEN**

foundUser <- user

**BREAK**

**END IF**

**END FOR**

**IF** NewUserÌnormation.IS NOT ValidInformation **THEN**

Display <- "Invalid information, edit user fail"

**ELSE**

**IF** NewUserÌnormation.FullName IS NOT NULL **THEN**

foundUser.FullName <- NewUserData.FullName

**END IF**

**IF** NewUserÌnormation.Email IS NOT NULL **THEN**

foundUser.Email <- NewUserData.Email

**END IF**

**IF** NewUserÌnormation.PhoneNumber IS NOT NULL **THEN**

foundUser.PhoneNumber <- NewUserData.PhoneNumber

**END IF**

**IF** NewUserÌnormation.Address IS NOT NULL **THEN**

foundUser.Address <- NewUserData.Address

**END IF**

**IF** NewUserÌnormation.UserName IS NOT NULL **THEN**

foundUser.UserName <- NewUserData.UserName

**END IF**

IF NewUserÌnormation.Password IS NOT NULL **THEN**

foundUser.Password <- NewUserData.Password

**END IF**

IF NewUserÌnormation.Status IS NOT NULL **THEN**

foundUser.Status <- NewUserData.Status

**END IF**

**IF** NewUserÌnormation.RoleID IS NOT NULL **THEN**

foundUser.RoleID <- NewUserData.RoleID

**END IF**

Display <- "User information updated successfully"

**END IF**

**END**

Operation:

GetRole(out RoleList)

**BEGIN**

RoleList <- []

role1 <- new Role()

role1.setRoleID(1)

role1.setRoleName("Admin")

role2 <- new Role()

role2.setRoleID(2)

role2.setRoleName("Doctor")

role3 <- new Role()

role3.setRoleID(3)

role3.setRoleName("Nurse")

RoleList.append(role1)

RoleList.append(role2)

RoleList.append(role3)

**IF** RoleList is empty **THEN**

PRINT "No roles available"

**ELSE**

Out RoleList <- RoleList

**END IF**

**END**

+ AddUser( In FullName, In Email, In PhoneNumber, In Address, In UserName, In Password, In Status, In RoleId, Out Display)

**BEGIN**

newUser <- AddNewUser( FullName, Email, PhoneNumber, Address, UserName, Password, Status, RoleId)

**IF** newUser.IS NOT ValidInformation **THEN**

Display <- "Invalid user information, add user fail"

**ELSE**

userList.Add(newUser)

Display <- "User added successfully"

**END IF**

**END**

+ActiveUser(In userId, In status, Out Display)

**BEGIN**

foundUser <- NULL

**FOR EACH** user IN userList **DO**

**IF** user.Id == userId **THEN**

foundUser <- user

**BREAK**

**END IF**

**END FOR**

**IF** foundUser IS NULL **THEN**

Display <- "User not found"

**ELSE**

foundUser.Status <- status

Display <- "User status updated successfully"

**END IF**

**END**

+DeActiveUser(In userId, In status, Out Display)

**BEGIN**

foundUser <- NULL

**FOR EACH** user IN userList **DO**

**IF** user.Id == userId **THEN**

foundUser <- user

**BREAK**

**END IF**

**END FOR**

**IF** foundUser IS NULL **THEN**

Display <- "User not found"

**ELSE**

foundUser.Status <- status

Display <- "User status updated successfully"

**END IF**

**END**

+ViewUserDetail(In userId, Out Display UserInformation)

**BEGIN**

foundUser <- NULL

**FOR EACH**

user IN userList DO

**IF** user.id == In userId **THEN**

foundUser <- user

**BREAK**

**END IF**

**END FOR**

**IF** foundUser == NULL **THEN**

PRINT "User is not available"

**ELSE**

UserInformation<- foundUser

Display UserInformation

**END IF**

**END**

#### 6. Change requirement

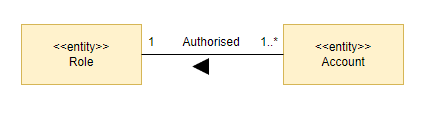
##### 6.1 Requirements

### 

| UC ID and Name: | UC-09: Edit User |
| --- | --- |
| Summary | This use case allows admin to edit user |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | User has logged in as Admin. |
| Main sequence: | 1.Admin login into the system successfully.  2.User clicks the “User Management” menu.  3.System shows user list and Admin choose user from list.  4.User click the “Edit” icon in the user list.  5.System shows Edit user information fields to screen.  6.User change information and click submit.  7.System save the changes. |
| Alternative sequences: | Step 6: Users fill invalid information into fields ,an error message is displayed, and the user is prompted to re-enter their information.System cannot return new information to Admin |
| Nonfunctional requirements: | **Performance**: information of the user will update and return userlist in 3 seconds. |
| Postcondition: | User account is updated |
| Outstanding questions: | N/A |

##### 6.2 Static modeling

Entity class



##### 6.3 Dynamic modeling

###### 6.3.1 Use Case model

| UC ID and Name: | UC-09: Edit User |
| --- | --- |
| Summary | This use case allows admin to edit user |
| Dependency: | N/A |
| Actors: | Admin |
| Preconditions | User has logged in as Admin. |
| Main sequence: | 1.Admin login into the system successfully.  2.User clicks the “User Management” menu.  3.System shows user list and Admin choose user from list.  4.User click the “Edit” icon in the user list.  5.System shows Edit user information fields to screen.  6.User change information and click submit.  7.System save the changes. |
| Alternative sequences: | Step 6: Users fill invalid information into fields ,an error message is displayed, and the user is prompted to re-enter their information.System cannot return new information to Admin |
| Nonfunctional requirements: | **Performance**: information of the user will update and return userlist in 3 seconds. |
| Postcondition: | User account is updated |
| Outstanding questions: | N/A |

###### 6.3.2 Identify boundaries and internal objects

* Boundary objects: AdminInteration
* Internal objects:UserCoordinator

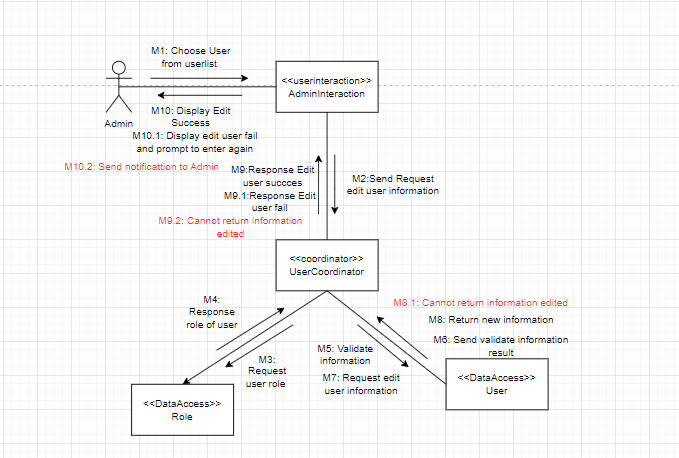
###### 6.3.3 Messages

**Main sequence:**

* M1: Choose User from list
* M2: Request user information
* M3: Request user role
* M4: Response role of user
* M5: Validate information
* M6: Send validate information result
* M7: Request edit user information
* M8: Return new information
* M9:Response Edit user succces
* M9.1:Response Edit user fail
* M10: Display Edit Success
* M10.1: Display edit user fail and prompt to enter again

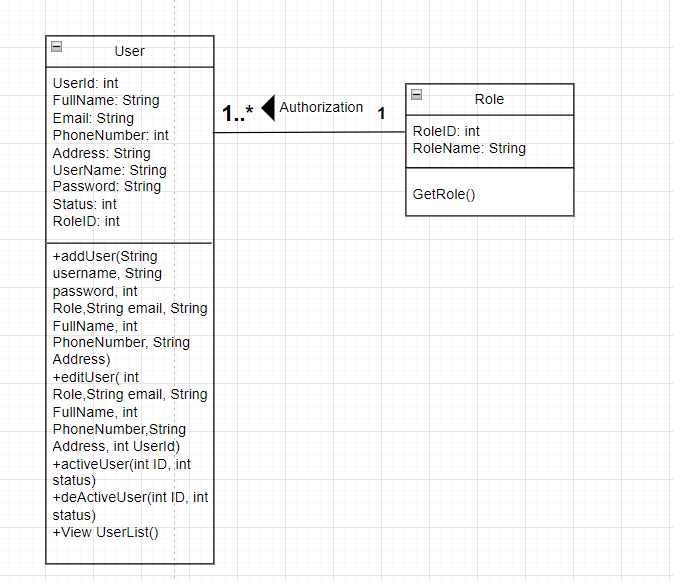
**Alternative sequence:**

* M8.1: Cannot return information edited
* M9.2 :Response cannot return information edited
* M10.2: Send notificattion to Admin



##### 6.4 Detail Design

###### 6.4.1 Class Diagram



###### 6.4.2 Relational Diagram



###### 6.4.3 Pseudocode

* Attribute: Id,FullName,Emai,lPhoneNumber,Address,UserName,Password,Status,RoleID
* Operation:

+Edit User(In userId, In NewUserÌnormation,Out Display)

**BEGIN**

foundUser <- NULL

**FOR EACH**

user IN userList DO

**IF** user.Id == In UserID **THEN**

foundUser <- user

**BREAK**

**END IF**

**END FOR**

**IF** NewUserÌnormation.IS NOT ValidInformation **THEN**

Display <- "Invalid information, edit user fail"

**ELSE**  **IF** NewUserÌnormation.IS NOT return ValidInformation **THEN**

Notification <-”Cannot return information edited”

**ELSE**

**IF** NewUserÌnormation.FullName IS NOT NULL **THEN**

foundUser.FullName <- NewUserData.FullName

**END IF**

**IF** NewUserÌnormation.Email IS NOT NULL **THEN**

foundUser.Email <- NewUserData.Email

**END IF**

**IF** NewUserÌnormation.PhoneNumber IS NOT NULL **THEN**

foundUser.PhoneNumber <- NewUserData.PhoneNumber

**END IF**

**IF** NewUserÌnormation.Address IS NOT NULL **THEN**

foundUser.Address <- NewUserData.Address

**END IF**

**IF** NewUserÌnormation.UserName IS NOT NULL **THEN**

foundUser.UserName <- NewUserData.UserName

**END IF**

IF NewUserÌnormation.Password IS NOT NULL **THEN**

foundUser.Password <- NewUserData.Password

**END IF**

IF NewUserÌnormation.Status IS NOT NULL **THEN**

foundUser.Status <- NewUserData.Status

**END IF**

**IF** NewUserÌnormation.RoleID IS NOT NULL **THEN**

foundUser.RoleID <- NewUserData.RoleID

**END IF**

Display <- "User information updated successfully"

**END IF**

**END**

Operation:

GetRole(out RoleList)

**BEGIN**

RoleList <- []

role1 <- new Role()

role1.setRoleID(1)

role1.setRoleName("Admin")

role2 <- new Role()

role2.setRoleID(2)

role2.setRoleName("Doctor")

role3 <- new Role()

role3.setRoleID(3)

role3.setRoleName("Nurse")

RoleList.append(role1)

RoleList.append(role2)

RoleList.append(role3)

**IF** RoleList is empty **THEN**

PRINT "No roles available"

**ELSE**

Out RoleList <- RoleList

**END IF**

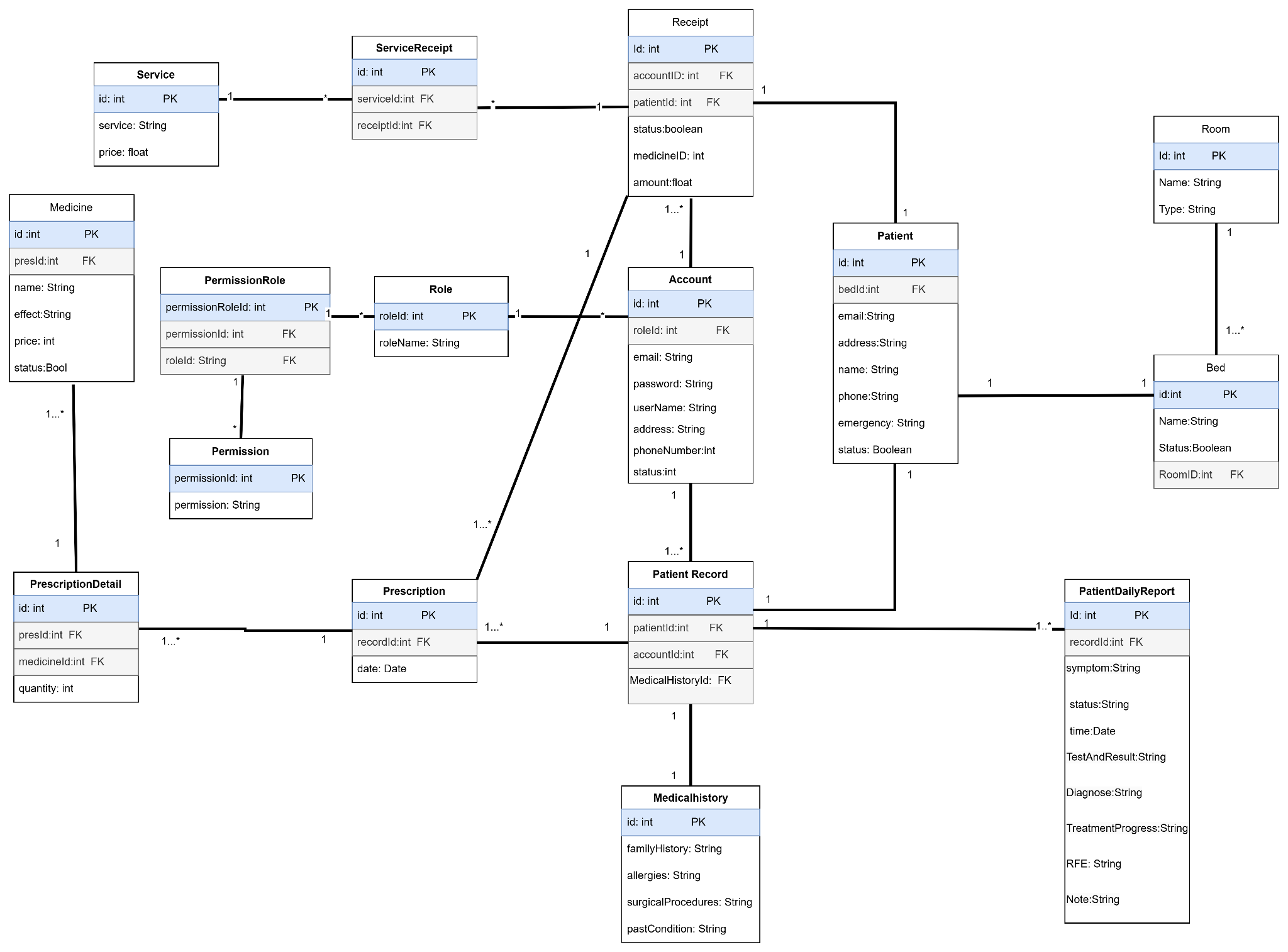
**END**

#### 7. Layers of Abstraction Pattern

| Presentation Layer | userList.jsp, editUser.jsp, viewUserDetail.jsp, addUser.jsp |
| --- | --- |
| Coordinator Layer | AddUserServlet.java,  EditUserServlet.java,  UserListServlet.java,  ViewUserServlet.java, |
| Data Access Layer | AccountDAO.java  RoleDAO.java |

## 

# III. Database



## 

# 