CS5373 TERM PROJECT REPORT

Healthcare System (HS)

Submitted by Group 22

Ayyappa Reddy Maramreddy - (R11888944)
Abhilash Reddy Gunukula- (R11885327)
Talha Jabbar Muhammad – (R11914715)
Hurt Natalie (D) – (R11924768)

Under the guidance of

Instructor: DR. Michael (Eonsuk) Shin, Ph.D
TA: Imran Pinjari, Namgyu Han

Texas Tech University
Department of Computer Science
Box 43104
Lubbock, TX 79409−3104

Table of Contents

Phase 1

Phase 2

PHASE 1

1. SYSTEM CONTEXT CLASS DIAGRAM

Objects used in health care system:

System:

1. Health Care System

Actors:

1. Patient
2. Doctor
3. Pharmacist
4. Insurance Company
5. Researcher
6. Bank

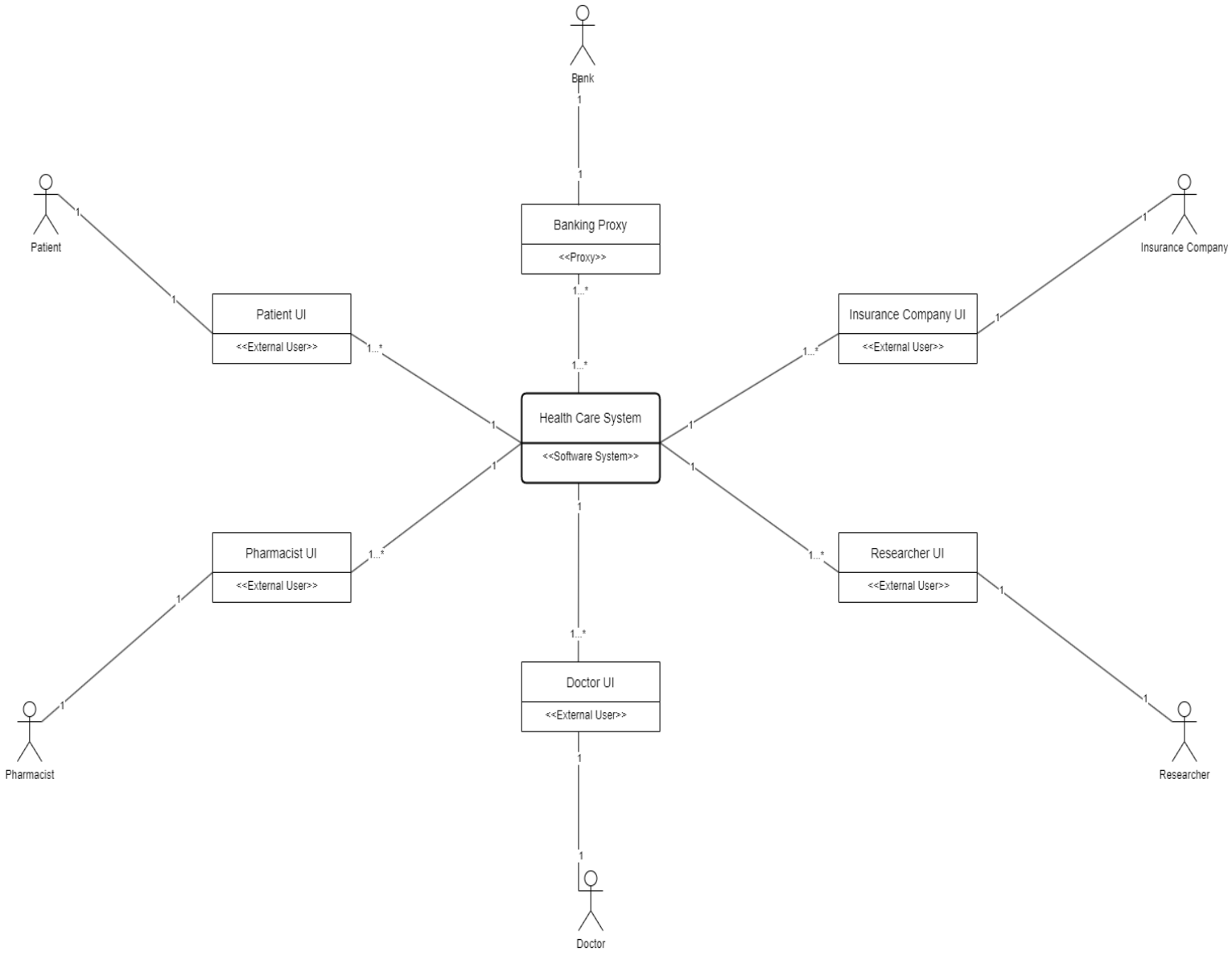
External Objects:

1. Patient UI
2. Doctor UI
3. Pharmacist UI
4. Insurance Company UI
5. Researcher UI

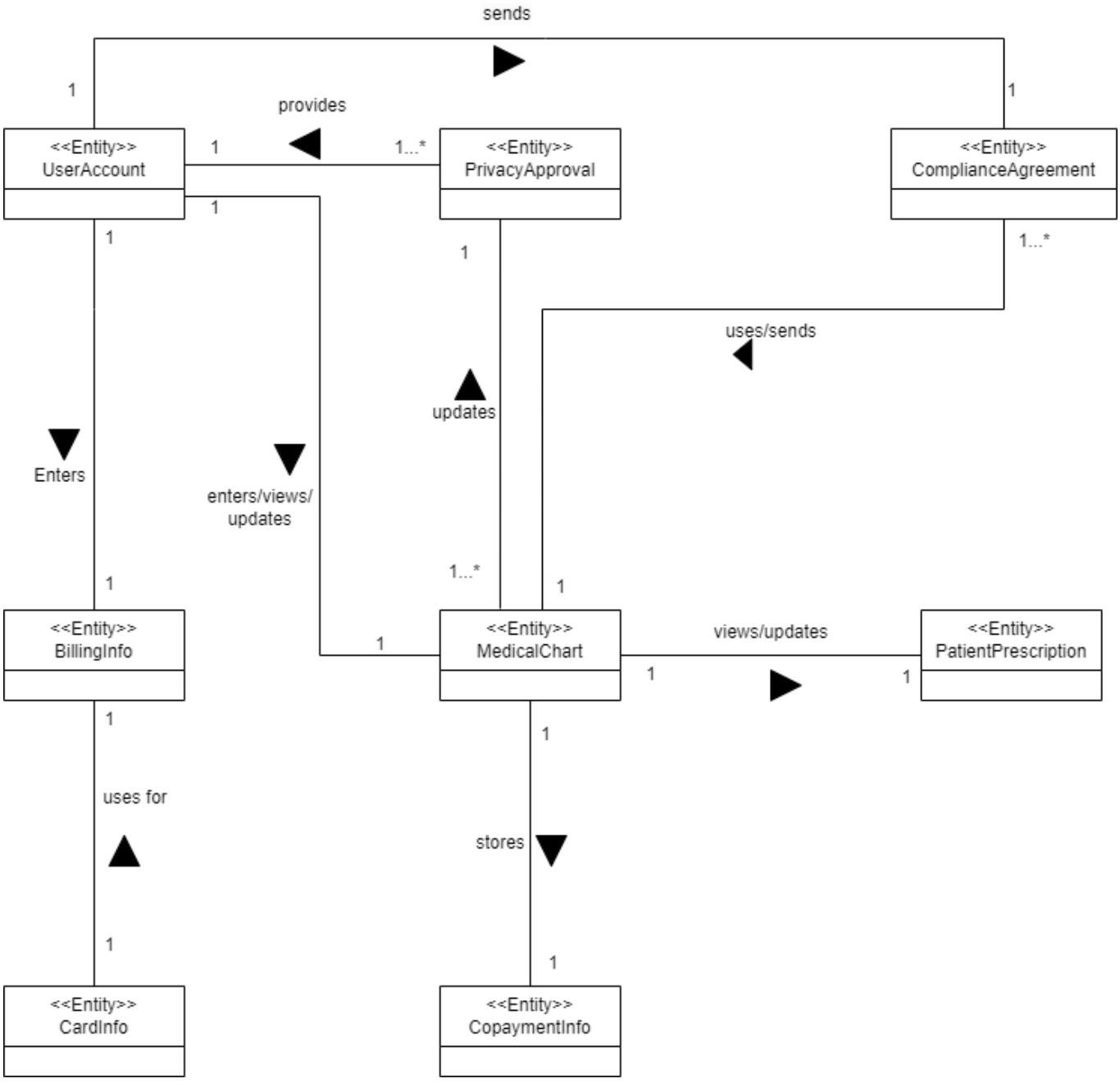
Proxies:

1. Banking Proxy

1.HEALTHCARE SYSTEM SOFTWARE CONTEXT CLASS DIAGRAM



2.CONCEPTUAL STATIC MODEL FOR HEALTHCARE SYSTEM -ENTITY CLASS



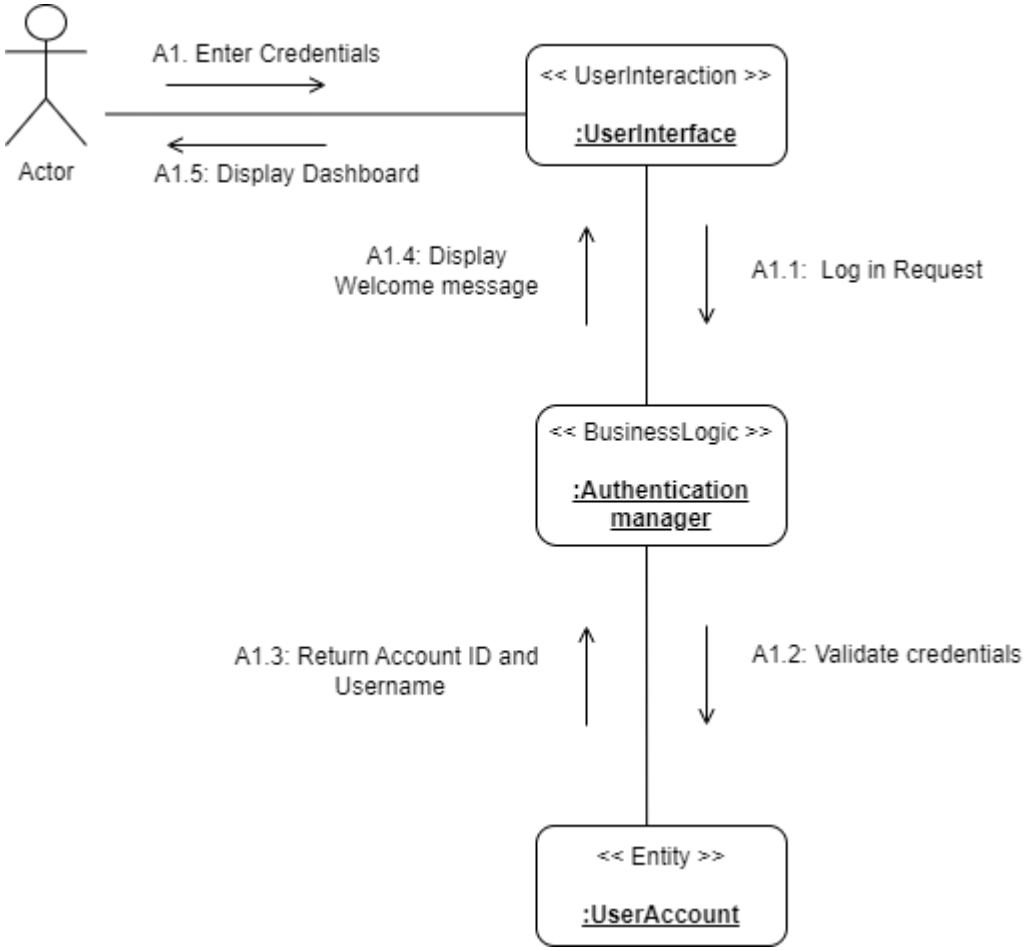
ENTITIES LIST

Assumptions Made:

1. In medical chart we have **treatmentNotes** attribute where it also contains disease name in it and when **researcher** search for a **disease** from blockchain it checks through treatment notes of medical chart db to get the patients list as per the requirement.
2. **ComplianceAgreement** is the entity data sent to a patient with an request approval message when research send the request.
3. When patient approves the request for trial research data then block chain system generates the permission code and saves in **PrivacyApproval** data with required details and those details will be sent to the researcher. So **PrivacyApproval** entity has been assumed with required attributes as per our design.
4. Since one patient can get many requests from many trial researchers so the relation between **ComplianceAgreement** and **MedicalChart** of patient can be many to many relations. Also, one researcher can get multiple requests approved from multiple patients so privacy approval to login entity maintains the many to one relation when login person is considered as researcher.

3. INTERCATION DIAGRAMS USING SEQUENCE OR COMMUNICATION DIAGRAM

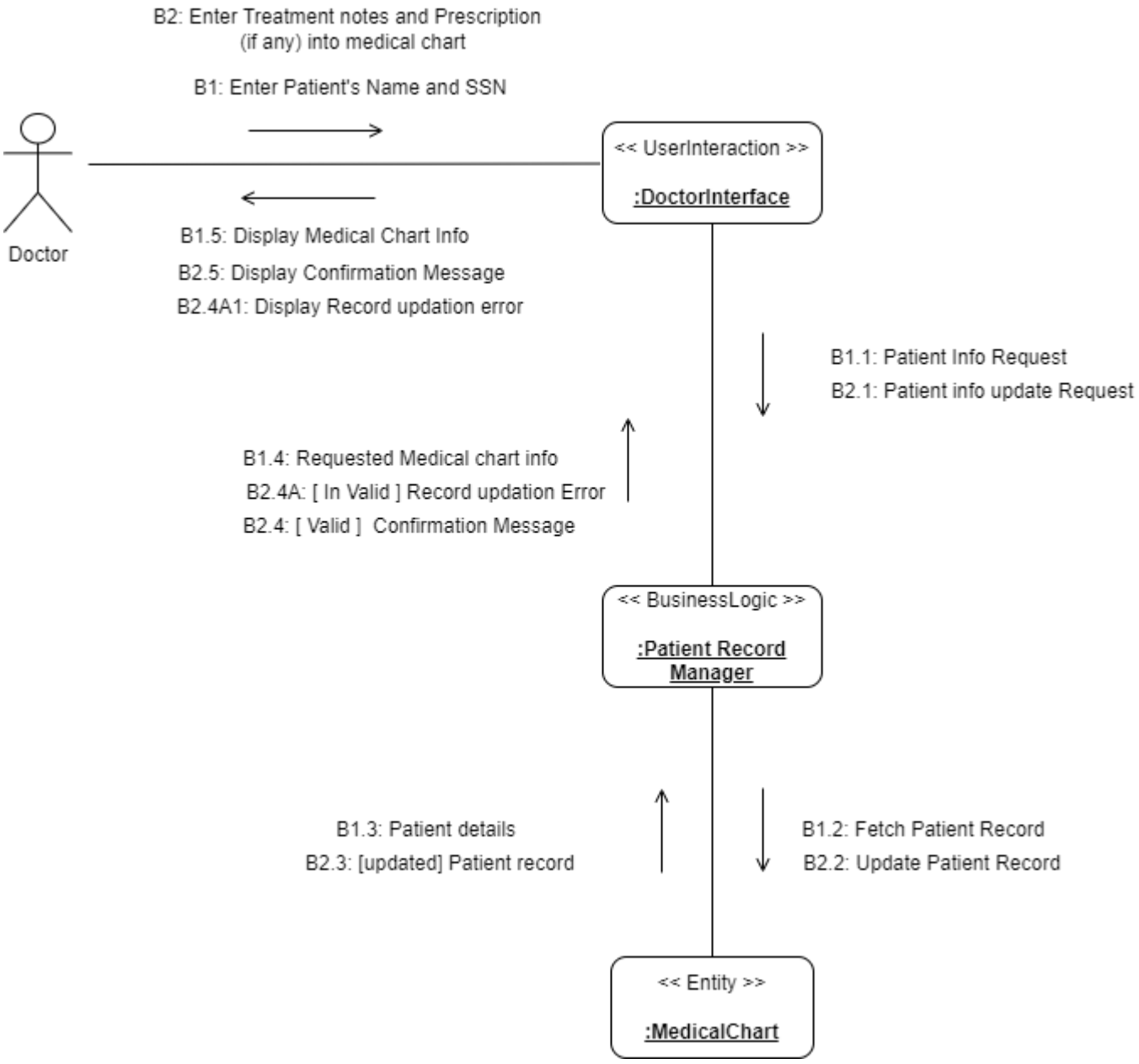
Use Case 1: Login



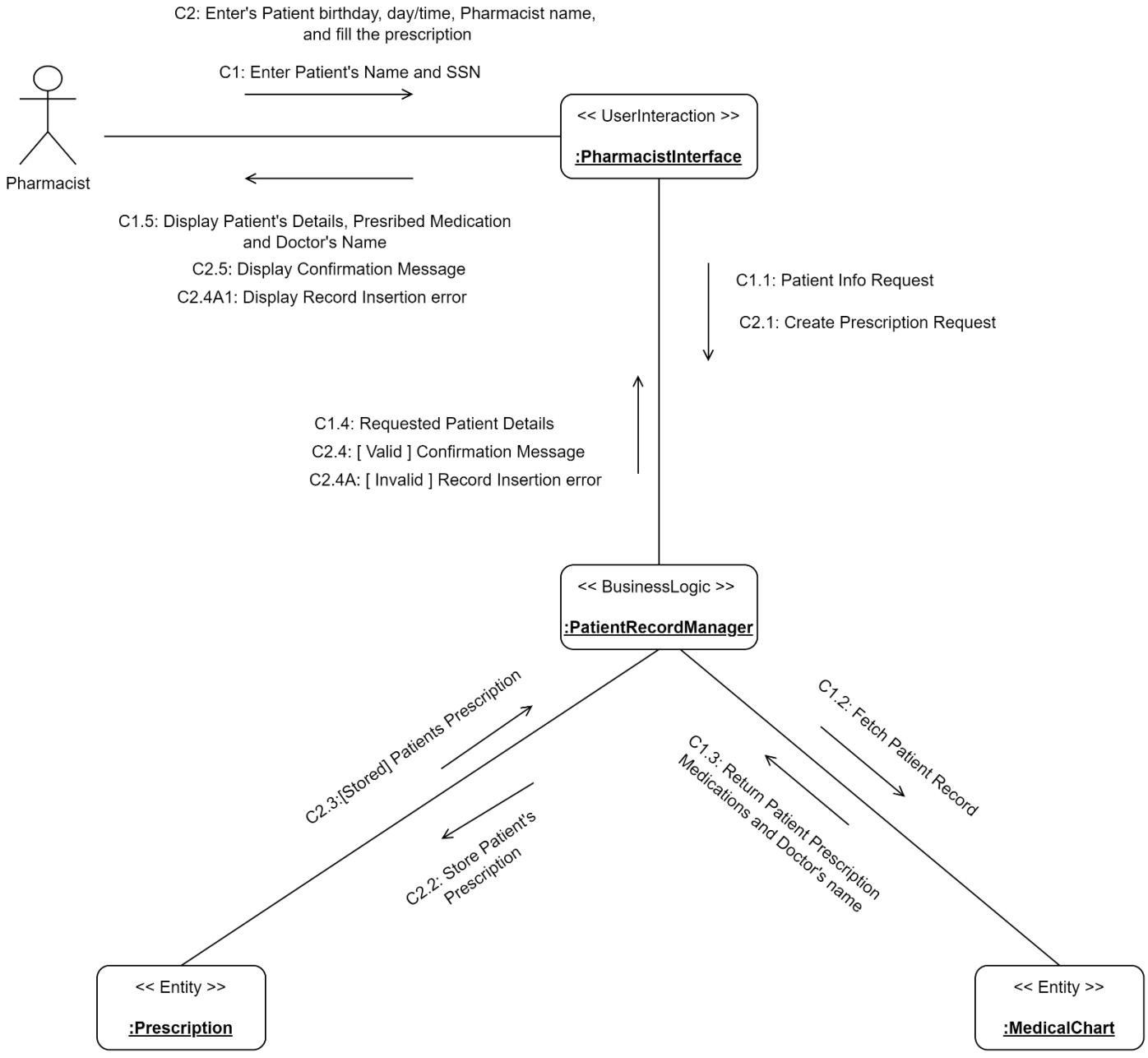
Assumptions made:

1. Here Actor can be any one from doctor, patient, pharmacist, researcher, insurance company who want to access the health care system.
2. When actor enters the credentials and validated. Based on the Id respective interface will be displayed with the welcome message.

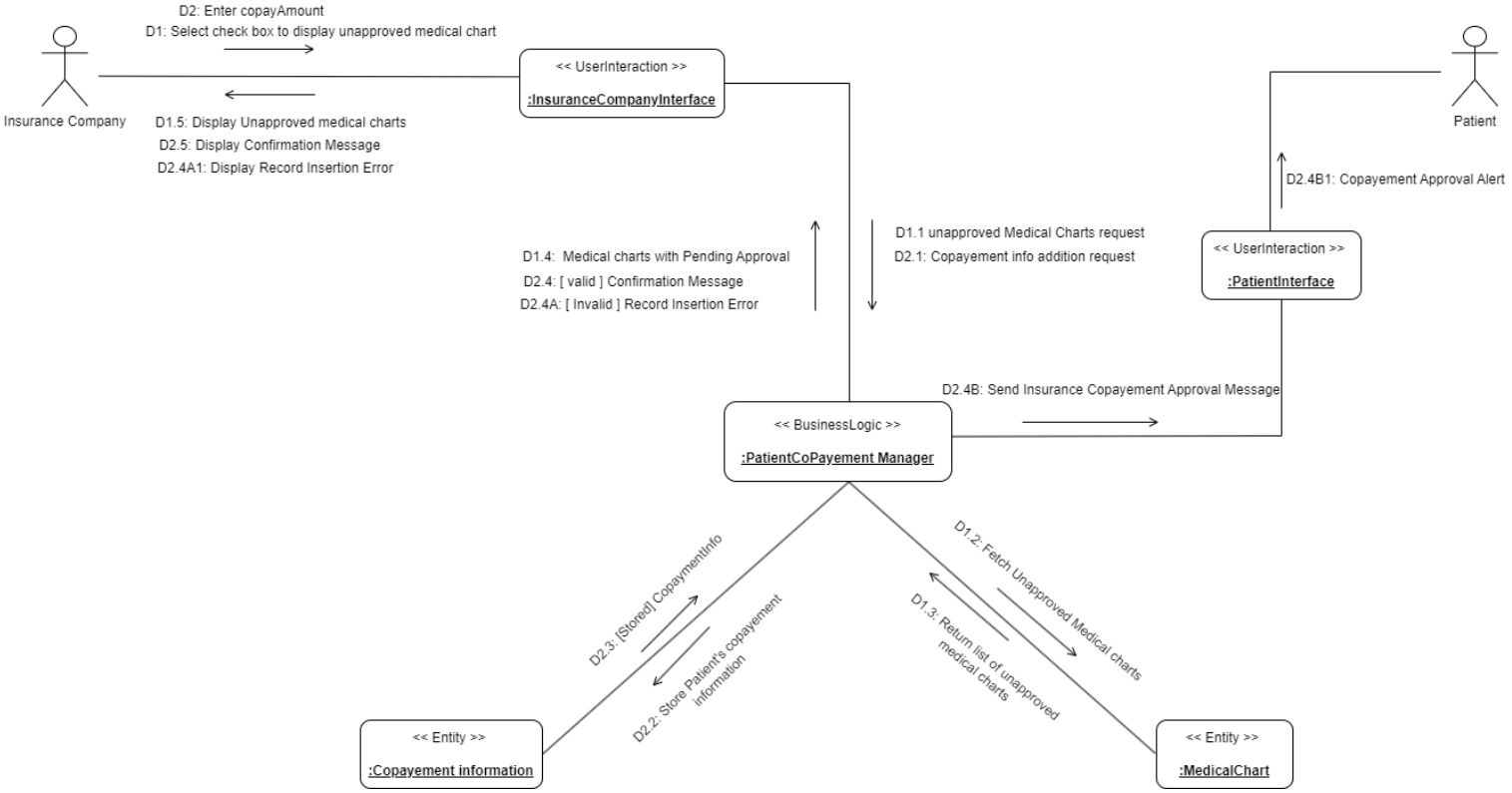
Use Case 2: Treat Patient



Use Case 3: Fill Prescription



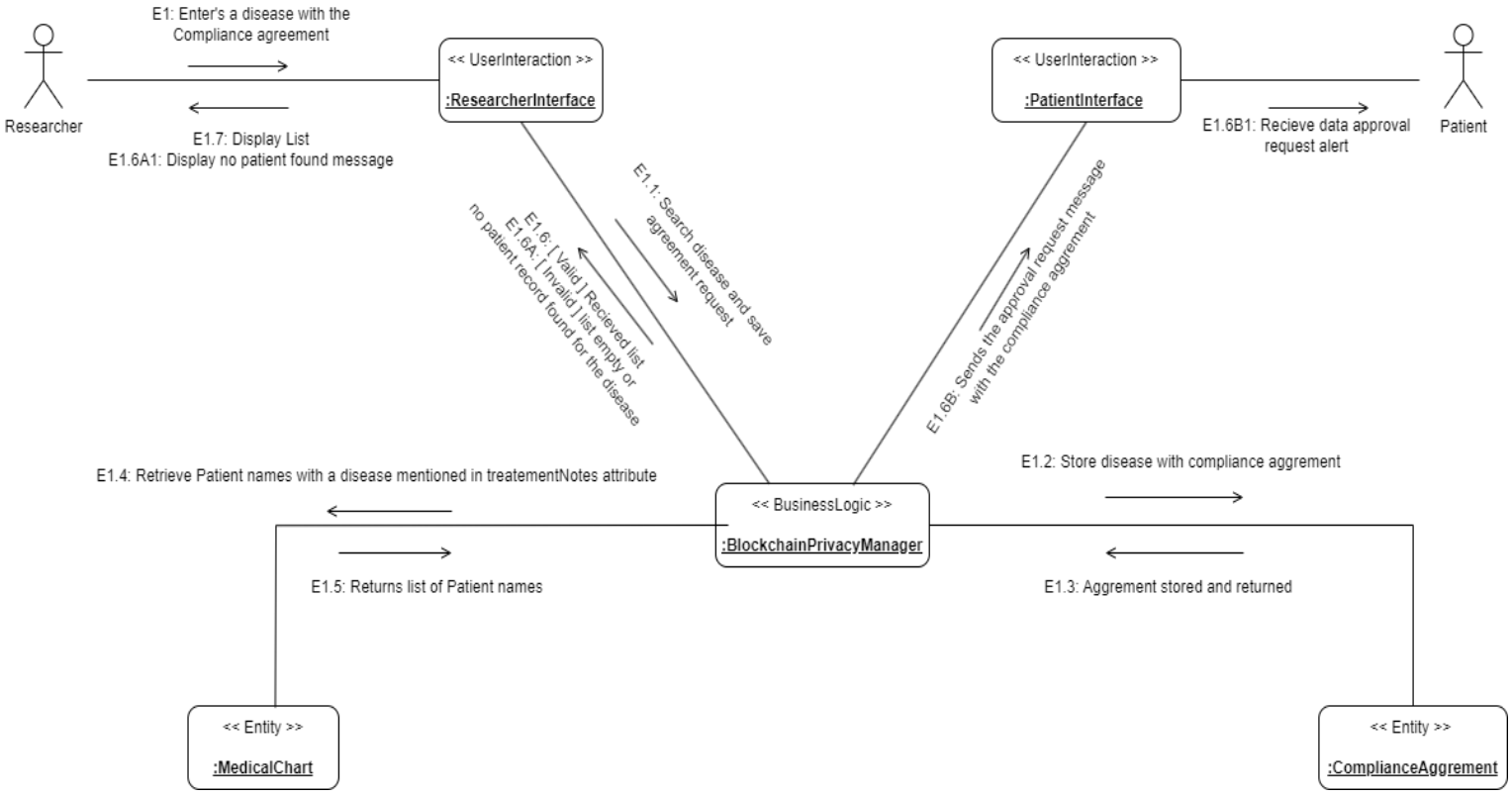
Use Case 4: Approve Copayment



Assumption made:

1. System credentials displays whole list of unapproved medical charts to insurance company where insurance company have feasibility to open each one record and then enter the copayment amount so that from interface all the required **copaymentInfo** related attributes will be sent to backend to store.

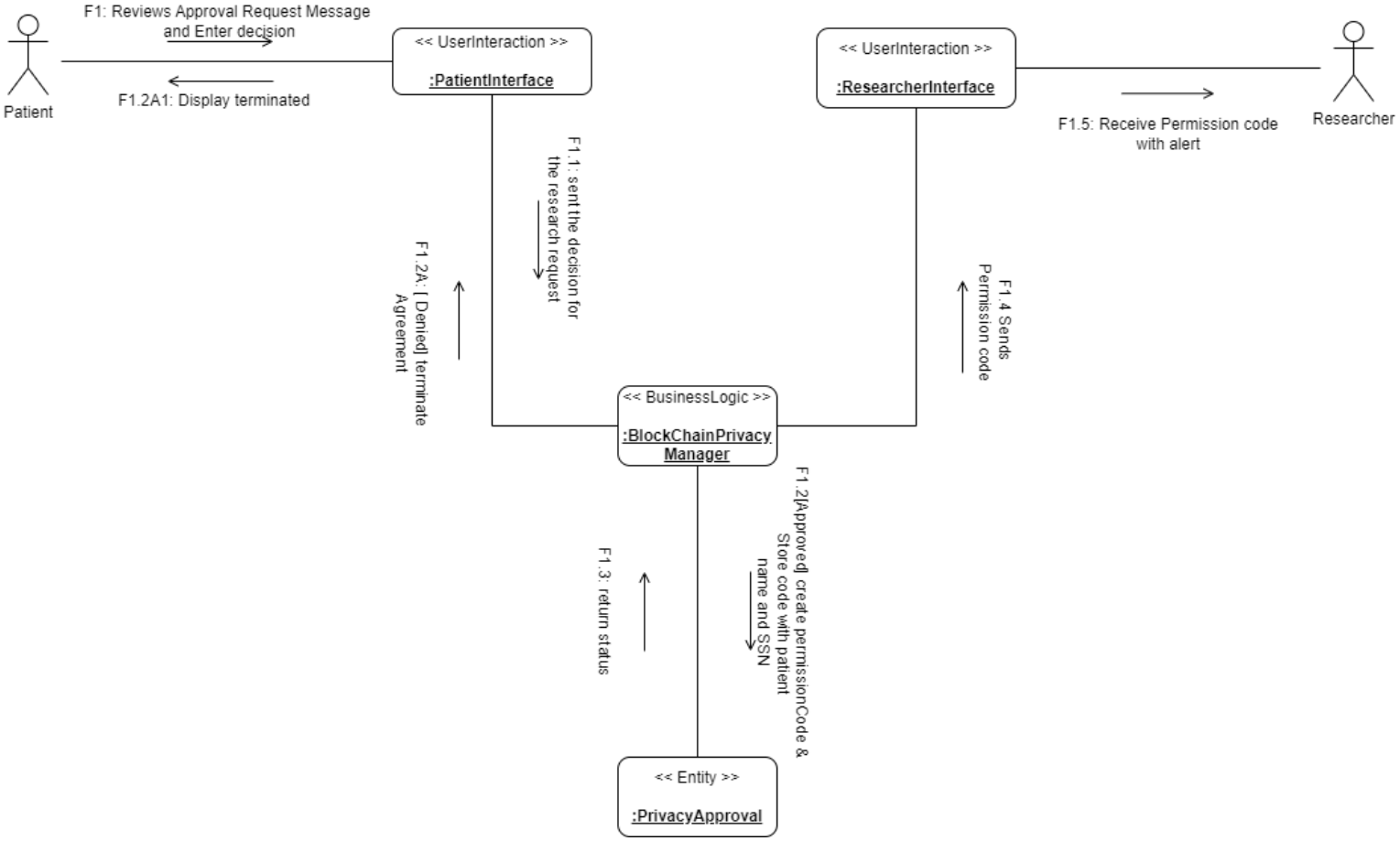
Use Case 5: Request Trial Research Data



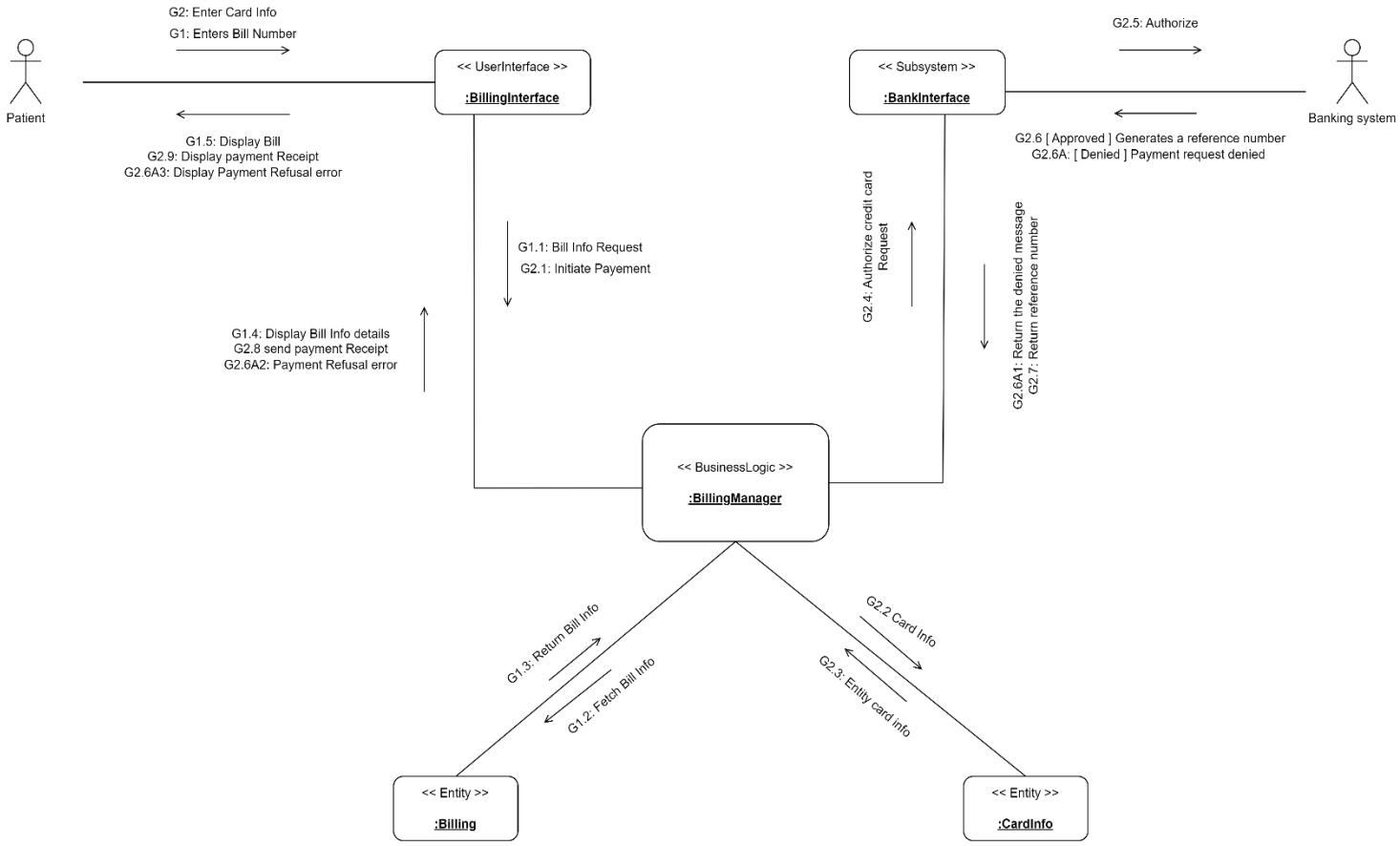
Use Case 6: Approve Trial Research Data Request

Assumption made:

1. Patient will review the approval request message and the compliance agreement and at the end of the review page patient have option to enter the decision to approve or deny the request.
2. Once decision entered interface sends the value to blockchain privacy manager were based on the decision it will either generates the permission code or terminates the interface.
3. In approval case we have separate entity where we are storing few patients related data, permission code, Boolean value of research approval
4. Also, in approval case we not showing anything in return to patient as use case mentioned we will alert researcher with code but nothing to display in patient perspective.



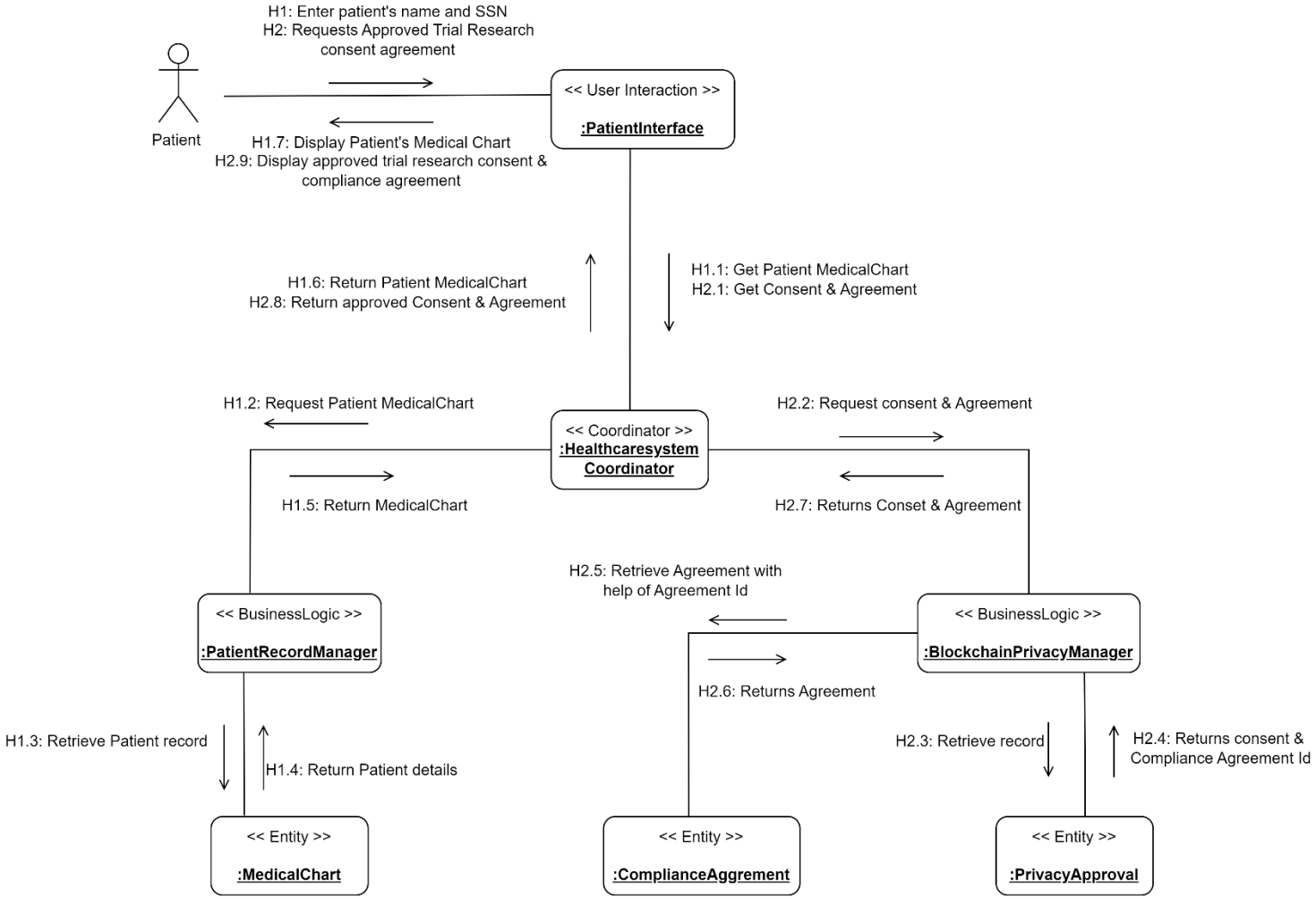
Use Case 7: Pay Bill



Assumptions made:

1. Bank interface is a subsystem of banking main system which recieves the cardInfor entity from the healthcare system and communicates with the main banking system to approve or deny the payment.

Use Case 8: View Medical Chart

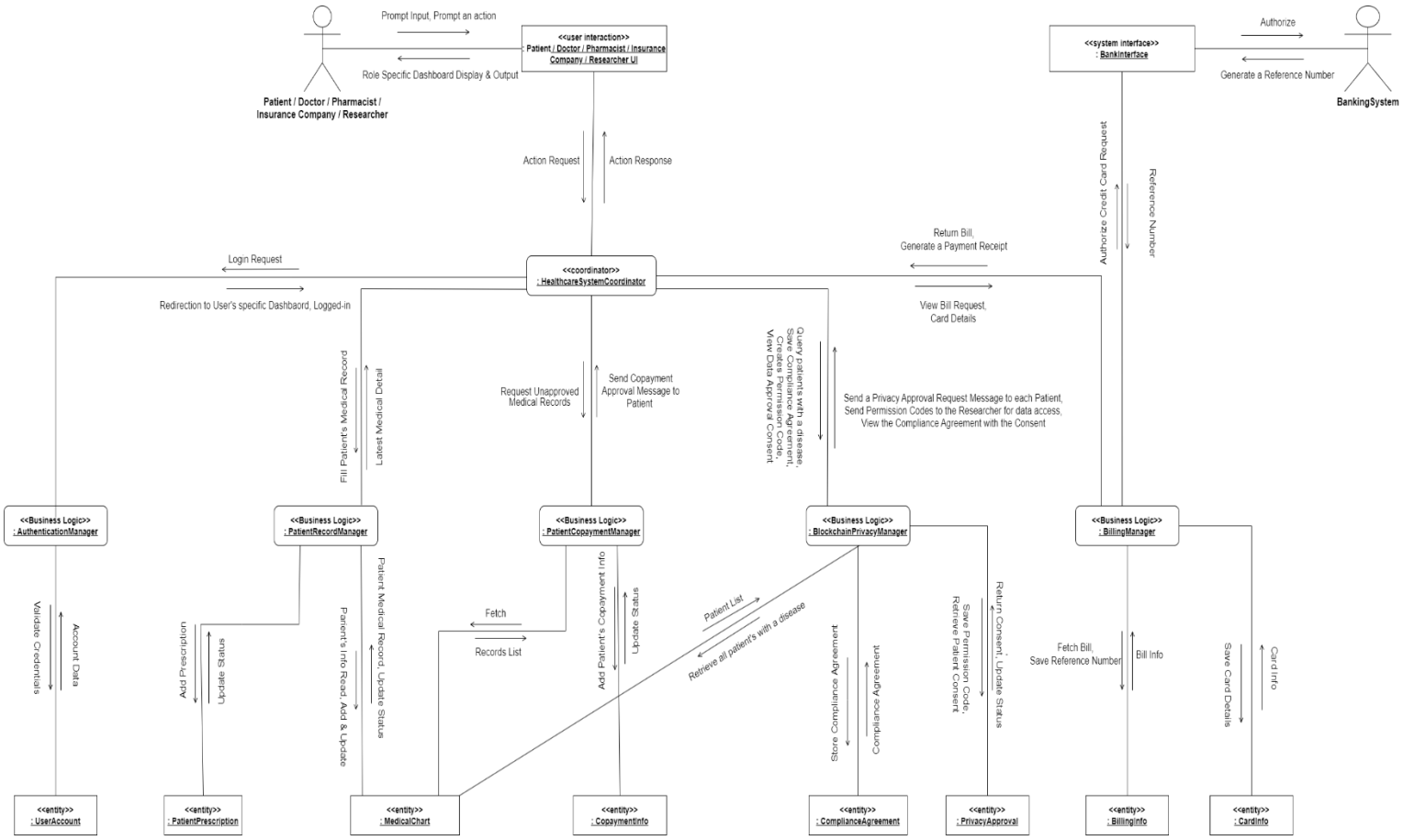


Assumptions made:

1. We are using two different entities to store some specific data with complianceAgreement and privacyApproval entity. So to show the approved trial research to the patient there is a need of calling these two entities.
2. These entities were already connected with the BlockchainPrivacyManger businees logic in our case. So it is the best practice to view approved trial research data blockchain logic only and that's reason we have used the coordinator logic to communicate with two different business logic to view medical chart and approved trial research data.

PHASE 2

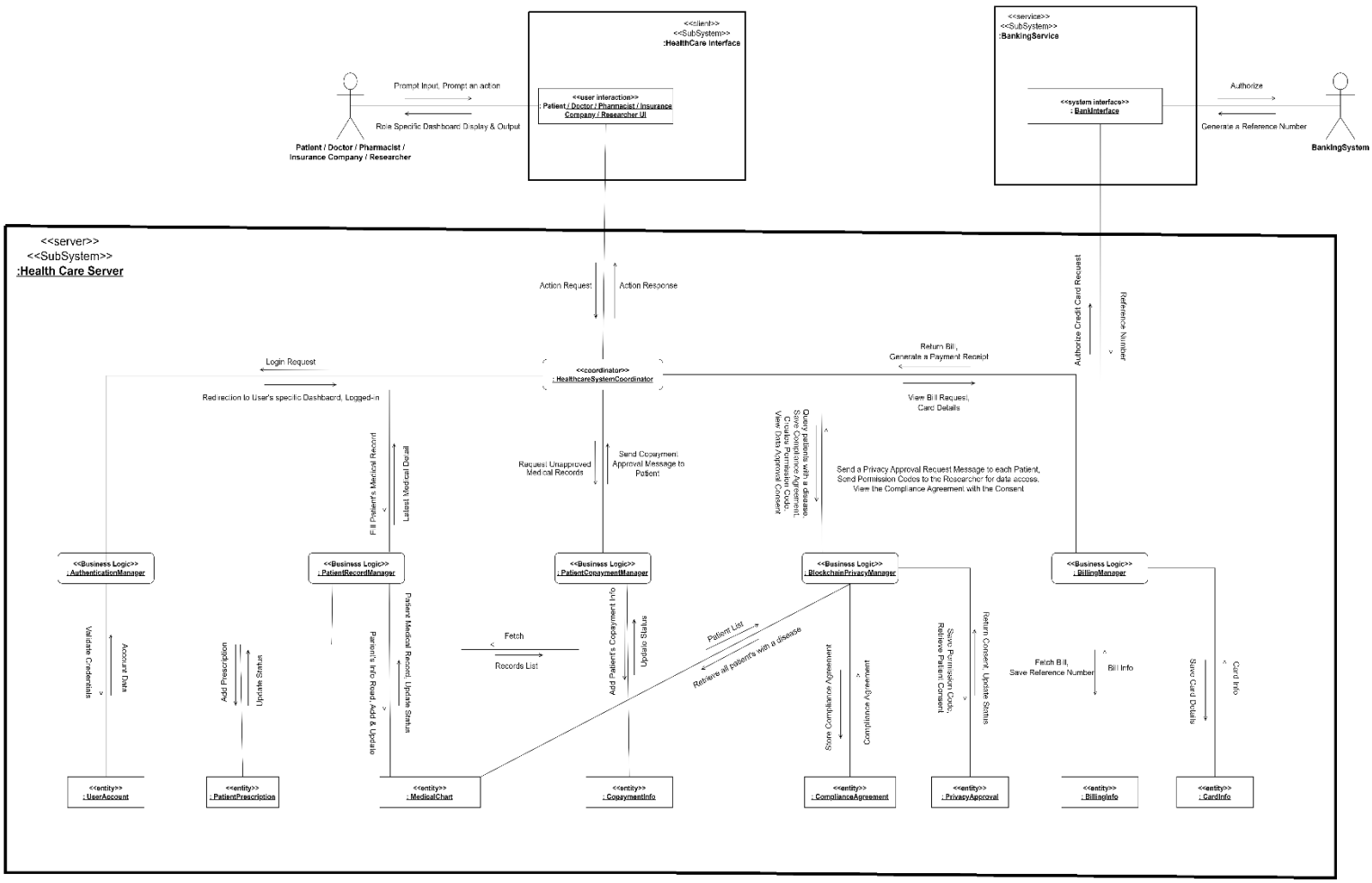
1. INTEGRATED COMMUNICATION DIAGRAM



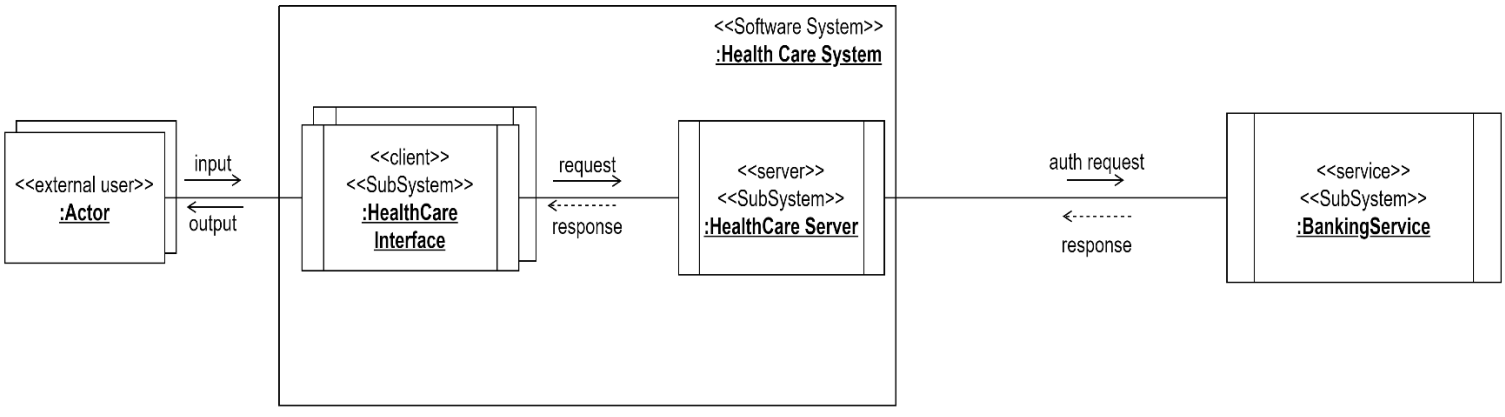
Here the user interaction can happen with anyone either doctor, patient, researcher, pharmacist, insurance company. Based on the user and action coordinator will decide to move required business logic.

2. SOFTWARE ARCHITECTURE DIAGRAM

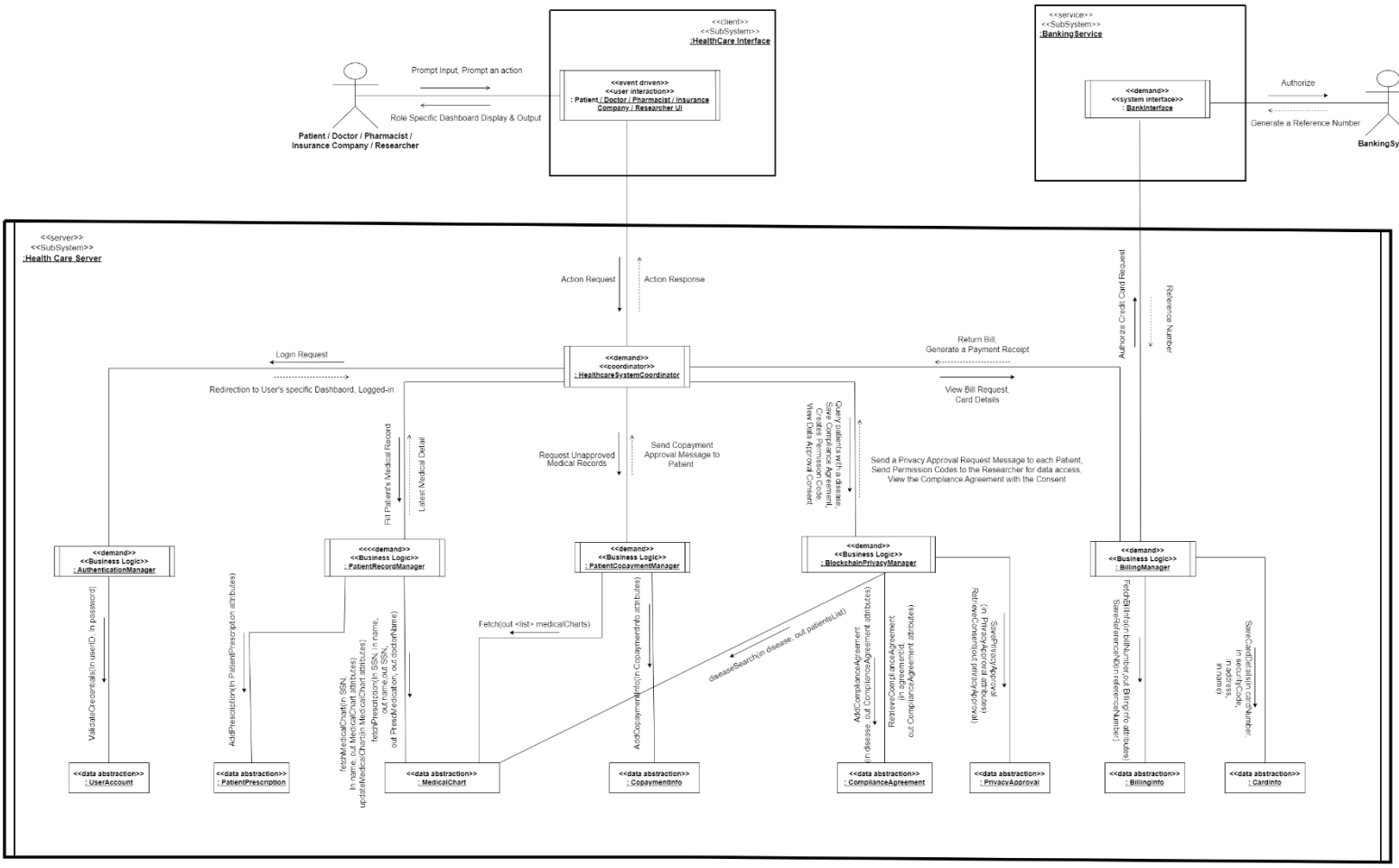
Software Architecture showing Client and Server of the System:



Message Communication Interfaces Between Client and Server:



3. TASK ARCHITECTURE:



Assumptions:

1. Insurance company has an option in UI with checkbox to get the unapproved medicalCharts that is where patientCopyamentManager applies query logic to fetch only unapproved medicalCharts by function call Fetch(out <list> medicalCharts).
2. Bank interface is a subsystem of banking main system which recieves the cardInfor entity from the healthcare system and communicates with the main banking system to approve or deny the payment.
3. Since we are using many entities with many number of attributes. It is tough to mention in the diagram, so we have mentioned as $entityName$ Attributes. This can say we are using all the attributes of the respective entity.
4. No matter who is the actor, but we are always showing required response as actor expected or the error messages. So, this error messages or success related data always will be forward from the business logic as the response. Hence all the calls between the coordinator and the business logic are considered as synchronous messages with reply.

CRITERIA USED FOR TASK STRUCTURING:

Event Driven User Interaction task:

- Patient UI
- Doctor UI
- Pharmacist UI
- Insurance Company UI
- Researcher UI

Demand Driven Co-Ordinator task:

- HealthcareSystemCoordinator

Demand Driven BusinessLogic task:

- AuthenticationManager
- PatientRecordManager
- PatientCopaymentManager
- BlockchainPrivacyManager
- BillingManager

Demand Driven Service SubSystem Interface task:

- BankInterface