Medalyze: Deliverable #3 – Domain Classes, Behavioral Models, and UML

1. Introduction

This report presents the analysis and modeling activities required for Deliverable #3. It includes:

- Domain class identification using the Noun Technique.
- Validation and refinement of use cases using the CRUD matrix.
- Behavioral modeling using Activity Diagrams, System Sequence Diagrams, and State Machine Diagrams.

2. Domain Classes (Noun Technique)

2.1 Step 1: Raw Noun Extraction

A comprehensive extraction of all nouns found in the Medalyze system description.

Noun	Context	Notes
Patient	**Book appointment** **View EHR &lab results** **Validate insurance claim**	
Doctor	**Book appointment**	

	Manage patient EHR **Send E-prescription**	
Appointment	**Book appointment** **Generate utilization report**	
Lab results	**View EHR & lab results**	
Date	**Book appointment**	
Time	**Book appointment**	
Pharmacy staff	**Processing E-prescription**	
Administrator	Generate Utilization Report Manage User Roles & Permissions	
Report	Generate Utilization Report	
Reminder	**Book appointment**	
Prescription	Send E-Prescription Process E-Prescription	
Medication	Send E-Prescription Process E-Prescription	
Medical record	**View EHR &lab results**	

	Manage patient EHR	
Patient record	**View EHR &lab results** **Manage patient EHR**	
Medical history	**View EHR &lab results** **Manage patient EHR**	
Allergy	**Manage patient EHR**	
Test history	**View EHR & lab results**	
System	**All use cases**	
Billing record	**Validate insurance claim**	
pharmacy	**Send E-prescription** **Process E- prescription**	
Notification	**Book appointment** **Send E-prescription** **Process E- prescription**	
Status	**Send E- prescription** **Process E- prescription** **Validate insurance claim**	
Abnormal value	**View EHR & lab results**	
Doctor notes	**View EHR & lab results**	

2.2 Step 2: Noun Classification

Nouns are classified into Candidate Classes, Attributes, Excluded Nouns, or Uncertain Nouns.

Noun	Category	Rationale
Patient	**Candidate class**	System stores patient accounts and medical information.
Doctor	**Candidate class**	Represents a primary user type with its own data.
Appointment	**Candidate class**	Central object tracked, created, modified, and reminded.
Lab results	**Candidate class**	Uploaded, notified, viewed; persists independently.
Date	**Attribute**	Belongs to Appointment. Not a standalone class.
Time	**Attribute**	Belongs to Appointment. Not a standalone class.
Pharmacy staff	**Candidate class**	Distinct actor who processes prescriptions; system tracks their actions.
Administrator	**Candidate class**	Manages roles & reports; persistent user entity
Report	**Candidate class**	System stores generated reports historically.

Reminder	**Excluded**	Automatic system action; not stored as entity.
Prescription	**Candidate class**	Sent, processed, tracked with statuses.
Medication	**Candidate class**	Separate entity with attributes like name, dose, etc.
Medical report	**Attribute**	**Part of the medical record**
Medical record **EHR**	**Candidate class**	Part of EHR containing patient history, notes, and details
Patient record	**Candidate class**	Subset of EHR; stores persistent medical details such as history, notes, labs, and diagnoses.
Medical history	**Attribute**	Narrative info inside the EHR; not created or managed as a separate entity.
Allergy	**Attribute**	Belongs to Patient profile (classification attribute)
Test history	**Attribute**	Collection of Lab Results, not a separate class.
System	**Excluded**	Not a domain class.
Billing record	**Candidate class**	Used in insurance validation

pharmacy	**Candidate class**	Actor with relevant stored data.
Notification	**Excluded**	System output, not stored as standalone data.
Status	**Attribute**	Indicates processing state (Filled, Out of Stock, Approved).
Abnormal value	**Attribute**	Part of Lab Result; not modeled independently.
Doctor notes	**Attribute**	Part of EHR content; not separate.

2.3 Step 3: Final Domain Classes

The final set of selected domain classes for modeling.

Domain Class	**Description**
Patient	Represents a system user receiving healthcare services. Stores personal info, history, allergies, and associated medical records.
Doctor	Healthcare provider managing appointments, EHR, and prescriptions.
Appointment	Scheduled meeting between Patient and Doctor with date, time, and status.
LabResult	Diagnostic result linked to a patient and their medical record.
PharmacyStaff	Staff member processing electronic prescriptions in the pharmacy system.

Administrator	System user responsible for managing roles, permissions, and reports.
Report	Utilization and administrative reports generated by the system.
Prescription	Electronic prescription issued by a doctor and processed by pharmacy staff.
Medication	Drug entity with dosage, quantity, and related instructions.
MedicalRecord	Consolidated EHR record containing patient history, lab results, notes, and diagnoses.
PatientRecord	Subset of medical information focused on patient-specific historical entries.
BillingRecord	Stores billing details and insurance validation information for a patient encounter.
Pharmacy	Entity representing a pharmacy where prescriptions are processed.

2.4 Step 4: Identified Attributes

Patient

Attribute	Type	Rationale
patientID	String	Unique identifier.
name	String	Required personal information.
dateOfBirth	Date	Medically relevant.
contactInfo	String	Used for communication.
allergies	String	Affects prescriptions & safety.
medicalRecordID	String	Links patient to their EHR.

Doctor

Attribute	Type	Rationale
doctorID	String	Unique identifier.
specialty	String	Needed for scheduling/filtering.
contactInfo	String	Communication details.

Appointment

Attribute	Type	Rationale
appointmentID	String	Unique identifier.
date	Date	Mandatory for scheduling.
time	Time	Mandatory for scheduling.
status	String	Tracks appointment lifecycle (Confirmed, Cancelled, etc.).

LabResult

Attribute	Type	Rationale
labResultID	String	Unique identifier.
resultType	String	Type of diagnostic test.
resultValue	String	Output of the test.
abnormalFlag	Boolean	Indicates abnormal values.

PharmacyStaff

Attribute	Type	Rationale
staffID	String	Unique identifier.
role	String	Differentiates pharmacist/assistant.

Administrator

Attribute	Type	Rationale
adminID	String	Unique identifier.
role	String	Used for permissions and system management.

Report

Attribute	Type	Rationale
reportID	String	Unique identifier.
reportType	String	e.g., Utilization, performance.
generatedDate	Date	Timestamp for tracking.

Prescription

Attribute	Type	Rationale
prescriptionID	String	Unique identifier.
issueDate	Date	Required for pharmacy processing.
status	String	Drafted, Sent, Dispensed, Expired.

Medication

Attribute	Type	Rationale
medicationID	String	Unique identifier.
name	String	Required medication info.
dosage	String	Medical dosage instruction.
quantity	Integer	Amount to dispense.

MedicalRecord

Attribute	Type	Rationale
recordID	String	Unique identifier.
doctorNotes	String	Notes added by doctor.
historySummary	String	Summary of medical history.

PatientRecord

Attribute	Type	Rationale
recordEntryID	String	Unique identifier.
entryType	String	e.g., Diagnosis, Visit Note.
entryDate	Date	Timing of record entry.

BillingRecord

Attribute	Type	Rationale
billingID	String	Unique identifier.
amount	Double	Payment amount.
insuranceStatus	String	Approved, Denied, Pending.

Pharmacy

Attribute	Type	Rationale
pharmacyID	String	Unique identifier.
name	String	Pharmacy identity.
location	String	Linking prescriptions to pharmacy.

2.5 Step 5: Relationships and Association Types

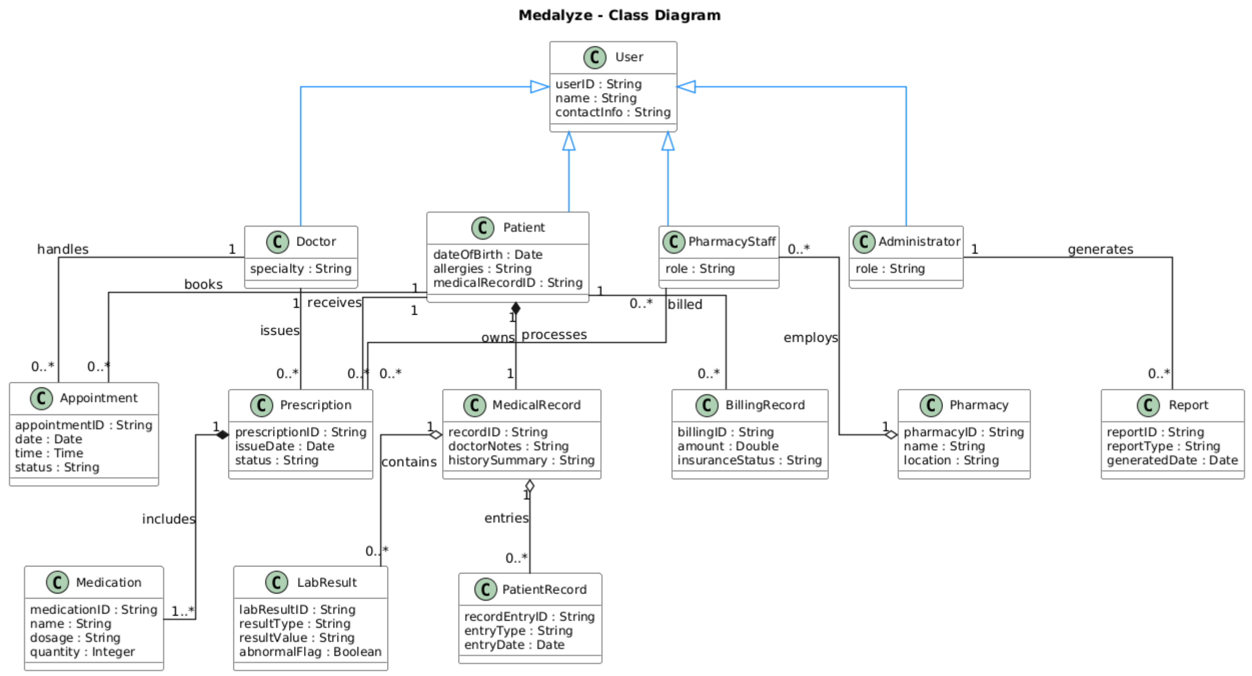
Relationship	Type	Classes	Rationale
Patient–Appointment	Association (1..*)	Patient ↔ Appointment	A patient can have many appointments; each appointment belongs to one patient.
Doctor–Appointment	Association (1..*)	Doctor ↔ Appointment	A doctor handles many appointments; each appointment is assigned to one doctor.
Patient–MedicalRecord	Composition (1–1)	Patient → MedicalRecord	MedicalRecord cannot exist without the patient.

MedicalRecord–LabResult	Aggregation (1..*)	MedicalRecord ↔ LabResult	Lab results attach to the medical record but can exist independently in storage.
MedicalRecord –PatientRecord	Aggregation (1..*)	MedicalRecord ↔ PatientRecord	Patient record attach to the medical report but can exist independently in storage
Doctor–Prescription	Association (1..*)	Doctor ↔ Prescription	Doctor issues many prescriptions.
Prescription–Medication	Composition (1..*)	Prescription → Medication	Medication list is part of the prescription.
Patient–Prescription	Association (1..*)	Patient ↔ Prescription	A prescription is issued for a specific patient.
PharmacyStaff–Prescription	Association (0..*)	PharmacyStaff ↔ Prescription	Staff process prescriptions.
Administrator–Report	Association (1..*)	Administrator ↔ Report	Admin generates many reports.
Patient–BillingRecord	Association (1..*)	Patient ↔ BillingRecord	BillingRecord belongs to a single patient.
Pharmacy–PharmacyStaff	Aggregation (1..*)	Pharmacy ↔ PharmacyStaff	Staff belong to a pharmacy.

Describing all identified relationships, including:

- Associations
- Cardinalities
- Generalization / Inheritance
- Aggregation
- Composition

2.6 UML Class Diagram



3. CRUD Technique and Use Case Validation

3.1 CRUD Matrix

The following CRUD matrix maps each domain class to the refined minimum use case set, ensuring full CRUD coverage while removing non-essential or redundant actions.

Domain Class	**Book Appt**	**View EHR**	**Edit EHR**	**Send Rx**	**Process Rx**	**Gen Report**	**Manage Roles**	**Validate Claim**
Patient	R	R	R	R	R	–	–	–
Doctor	R	R	R	R	R	–	–	–
Appointment	C R	–	–	–	–	R	–	–
LabResult	–	R	U	–	–	–	–	–

PharmacyStaff	–	–	–	–	R	–	–	–
Administrator	–	–	–	–	–	R	U	–
Report	–	–	–	–	–	C R	–	–
Prescription	–	–	–	C R	R U	–	–	–
Medication	–	–	–	R	R	–	–	–
MedicalRecord	–	R	U	–	–	–	–	–
PatientRecord	–	R	C U	–	–	–	–	–
BillingRecord	–	–	–	–	–	–	–	R U
Pharmacy	–	–	–	R	R	–	–	–

3.2 CRUD Analysis Results

Missing Use Cases (Based on CRUD Gaps)

- Patient / Doctor Account Update: No creation or editing supported.
- Lab Result Upload: Lab results lacked creation mechanism.
- Billing Record Creation: Previously read/updated only.
- Appointment Rescheduling & Cancellation: Updates missing.
- Prescription Modification: No update or cancel.
- Insurance Claim Creation: Only validation existed.
- Medication / Pharmacy Inventory Actions: Missing supporting use cases.

Overloaded Use Cases

- *Manage Patient EHR*: Split into View EHR, Edit Medical Record, Add Record Entry.
- *Process E-Prescription*: Split into Verify Prescription, Dispense Prescription.
- *Manage Roles & Permissions*: Narrowed to role and permission management only.

Actions Not Covered Before Refinement

- Deleting appointments, prescriptions, or record entries
- Creating lab results
- Creating billing records

- Updating user accounts
- Creating insurance claims
- Patient access to billing/insurance status

3.3 Updated Use Case Diagram

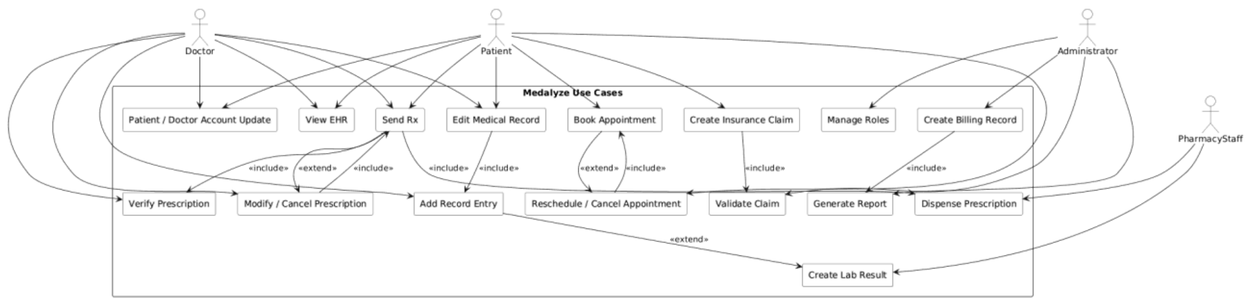


Figure X: Updated Use Case Diagram

4. Activity Diagrams (Three Complex Use Cases)

4.1 Activity Diagram: Book Appointment

Description:
This activity diagram details the end-to-end workflow for a Patient booking an appointment through the Medalyze system. The process begins with the Patient logging into the portal and navigating to the appointment page to select a specialty or doctor. The Patient then chooses a desired date and time. The System performs an availability check; if the slot is not available, the system displays alternative slots for the Patient to select. If a slot is available, the Patient confirms the selection. Upon confirmation, the System verifies the patient's eligibility, creates the appointment record, updates the doctor's schedule, and sends a confirmation notification. Finally, the Doctor/Clinic is notified and views the updated schedule, completing the process.

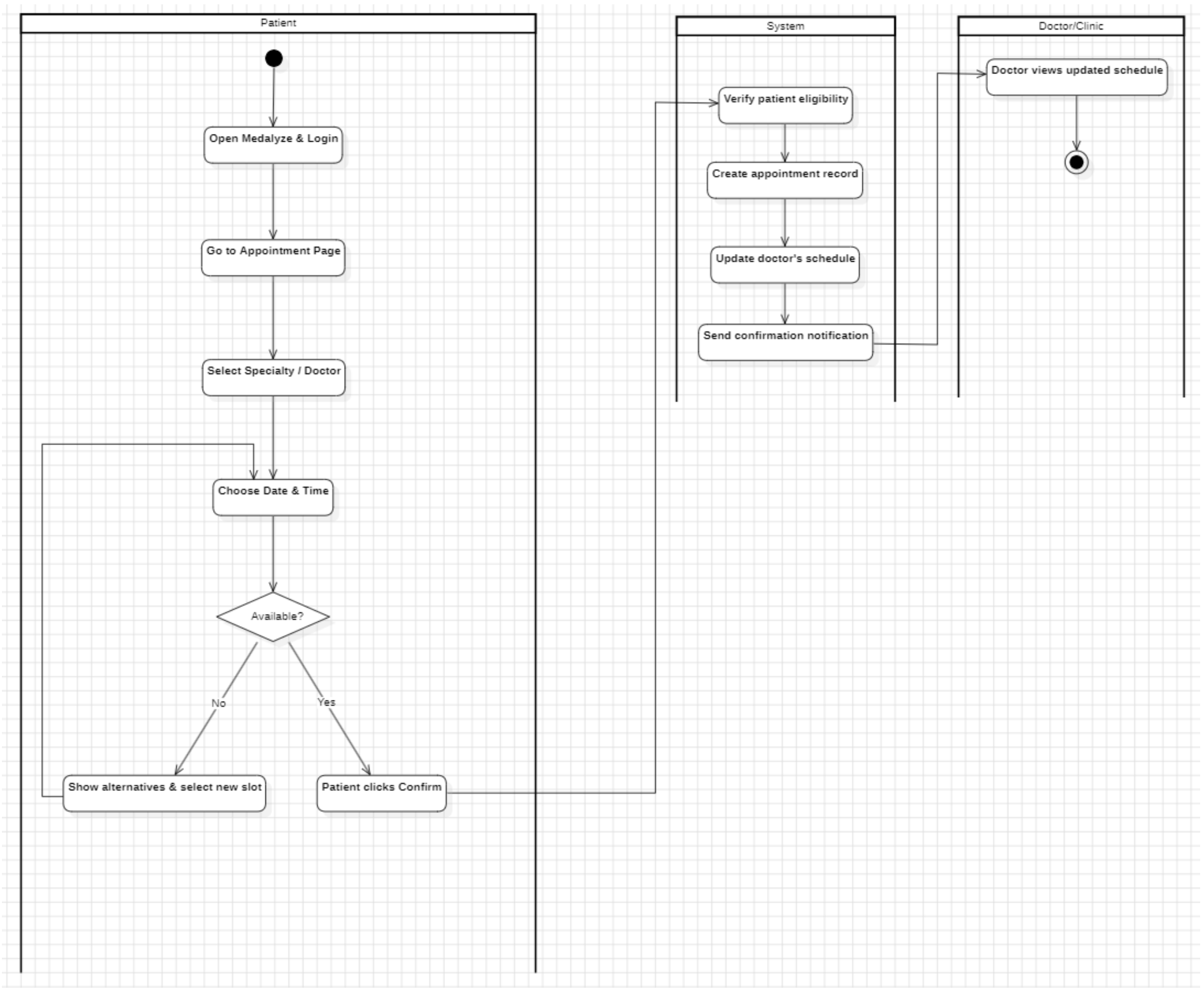

Patient
System
Doctor/Clinic
Open Medalyze & Login
Go to Appointment Page
Select Specialty / Doctor
Choose Date & Time
Available?
No
Yes
Show alternatives & select new slot
Patient clicks Confirm
Verify patient eligibility
Create appointment record
Update doctor's schedule
Send confirmation notification
Doctor views updated schedule

Figure X: Activity Diagram for Book Appointment.

4.2 Activity Diagram: View EHR & Lab Results

Description:
This diagram illustrates the workflow for a Patient accessing their electronic health records (EHR) and lab results. The Patient initiates the process by logging into the portal and navigating to their records to select either lab results or EHR. The System validates the Patient's access permissions. If permission is granted, the System retrieves the EHR summary and lab result history from the Medical Records / Lab Service. The retrieved data, including any abnormality flags, is displayed to the user. If abnormal values are detected, the System displays an advisory message. The Patient can then choose to open a specific

result for detailed review or exit the view. If access permissions are denied, the process terminates.

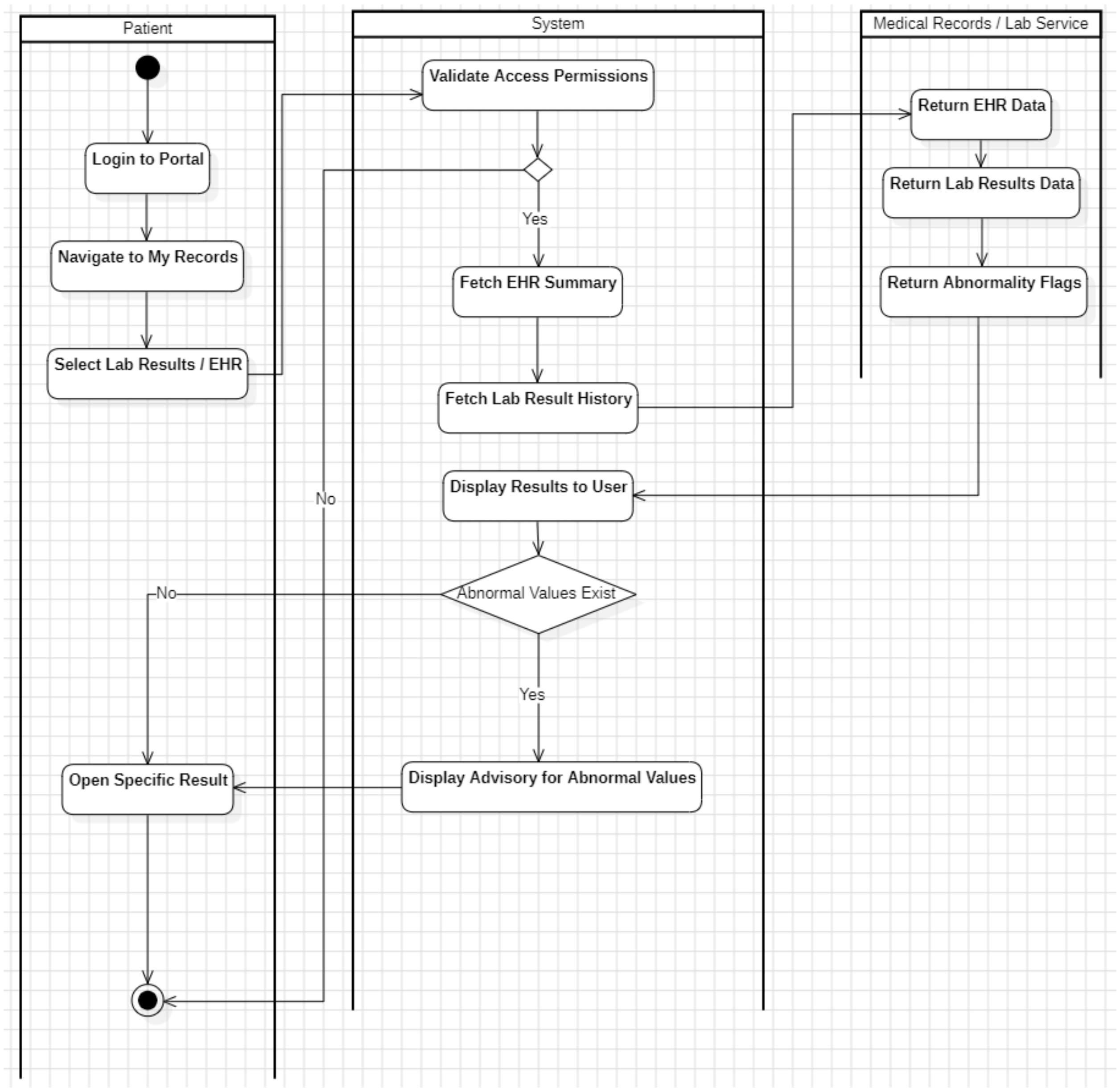


Figure X: Activity Diagram for Viewing EHR & Lab Results.

4.3 Activity Diagram: Process E-Prescription

Description:
This activity diagram models the workflow for processing an electronic prescription. It begins when the Doctor sends an e-Prescription. The System transmits the prescription to the Pharmacy and adds it to the pharmacy's processing queue. The Pharmacy Staff then views the prescription in the queue and opens it for processing. A decision point determines whether the medication is available ("Yes") or not ("No"). If available, the prescription is marked as "Completed," and the System notifies both the Patient and the Doctor of completion. If the medication is out of stock, the prescription is marked as "Out of Stock," and the System notifies the Doctor. In both scenarios, the process concludes after the appropriate notifications are sent.

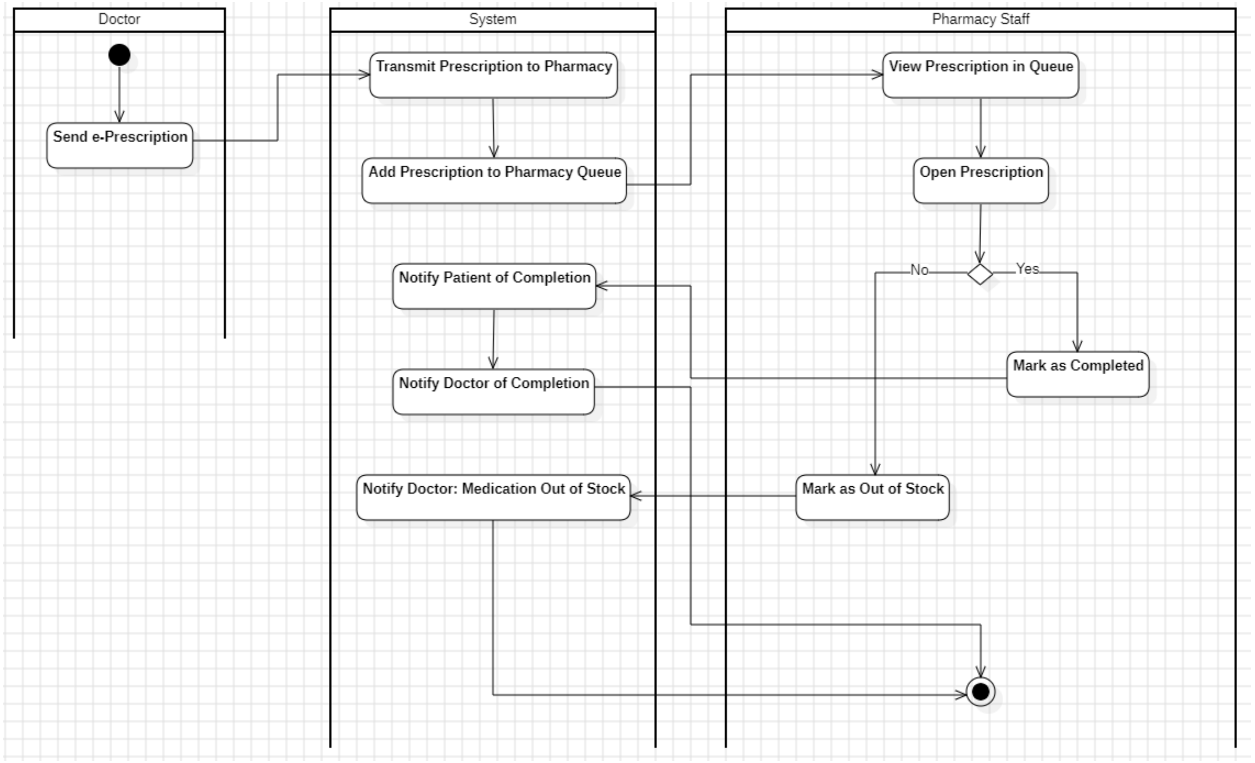
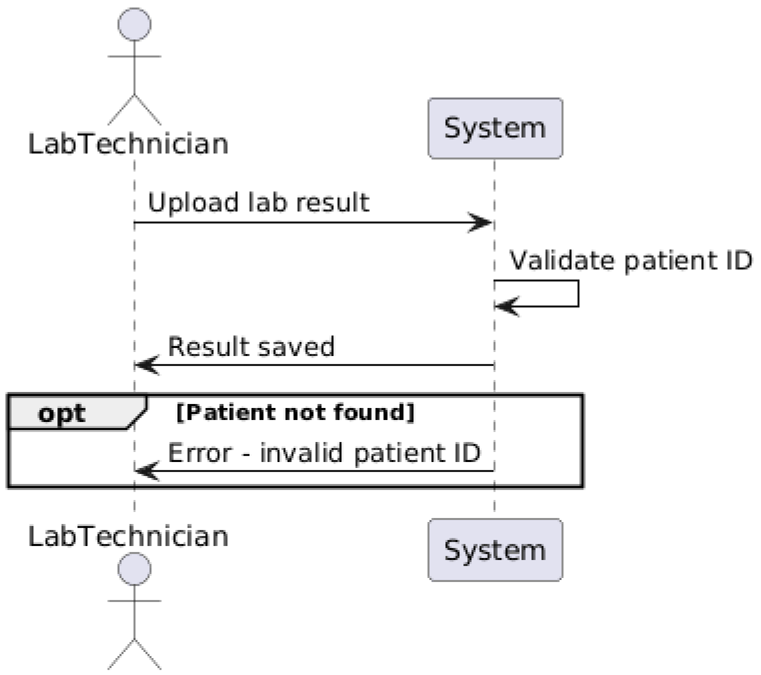
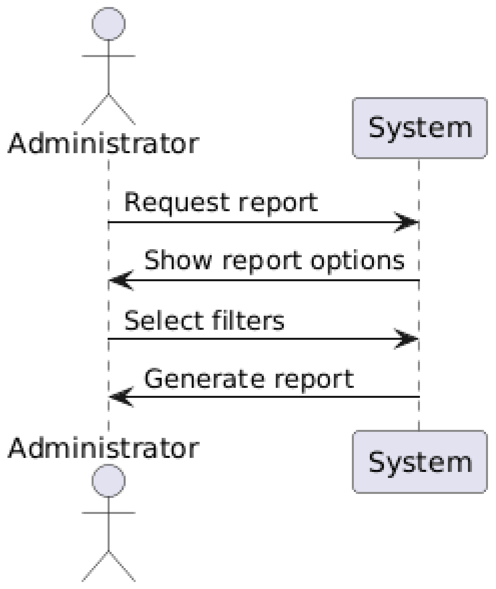
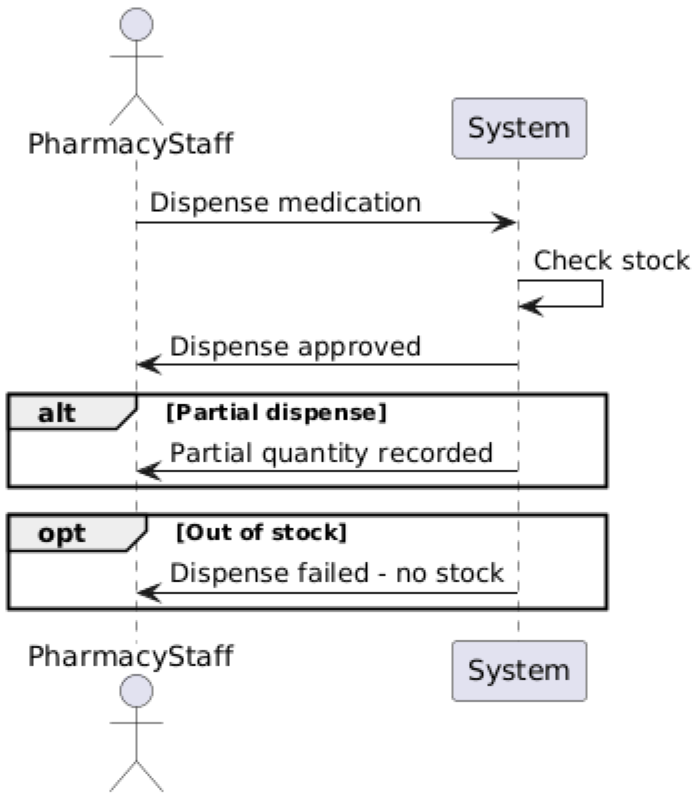

Doctor
System
Pharmacy Staff
Send e-Prescription
Transmit Prescription to Pharmacy
Add Prescription to Pharmacy Queue
View Prescription in Queue
Open Prescription
No
Yes
Notify Patient of Completion
Notify Doctor of Completion
Mark as Completed
Notify Doctor: Medication Out of Stock
Mark as Out of Stock

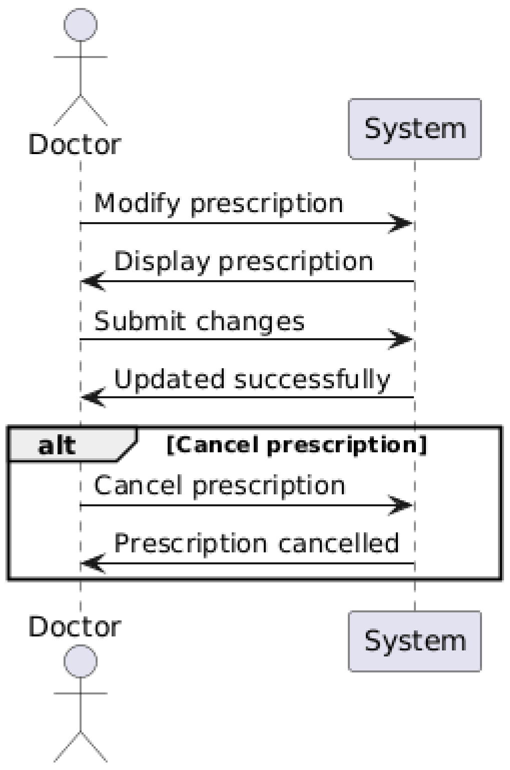
Figure X: Activity Diagram for Processing E-Prescription.

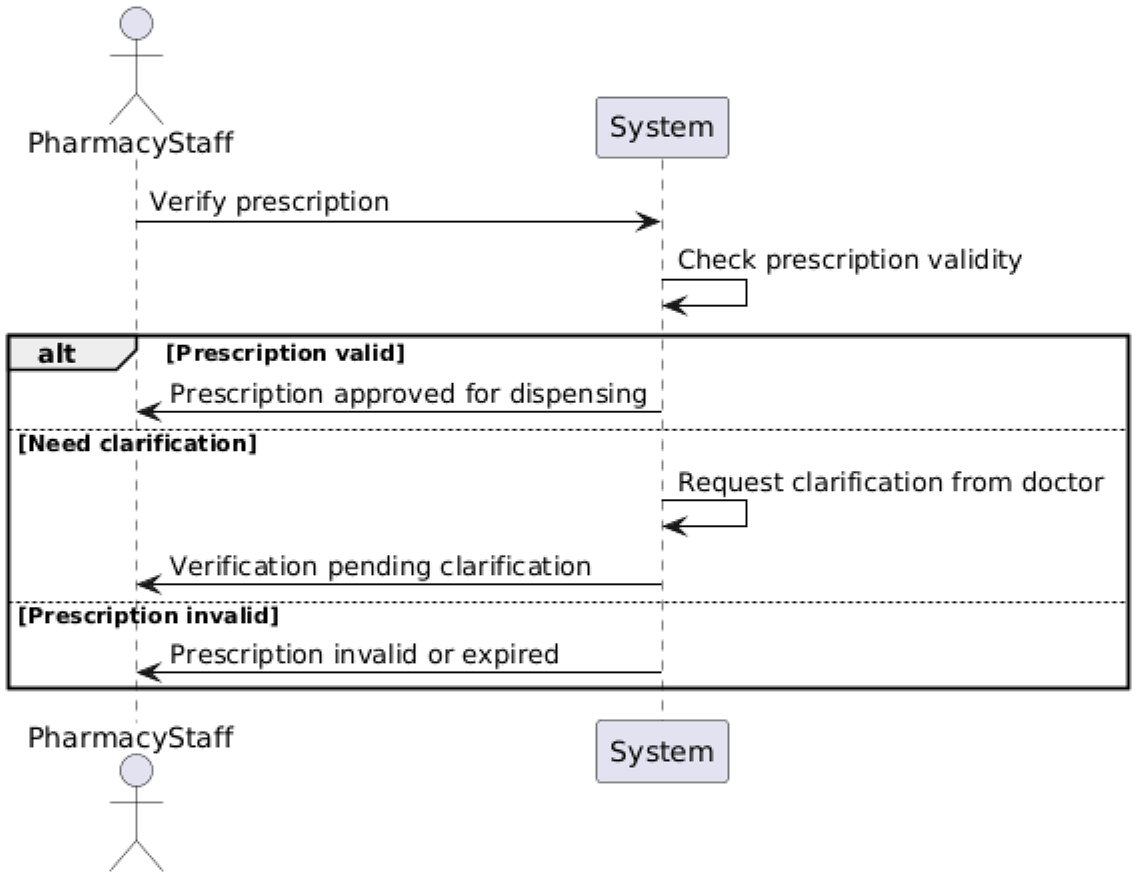
5. System Sequence Diagrams (SSDs)

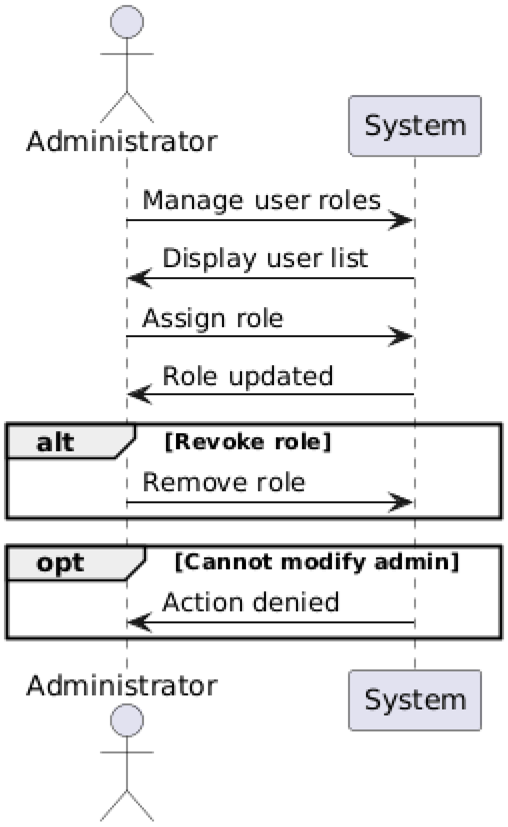


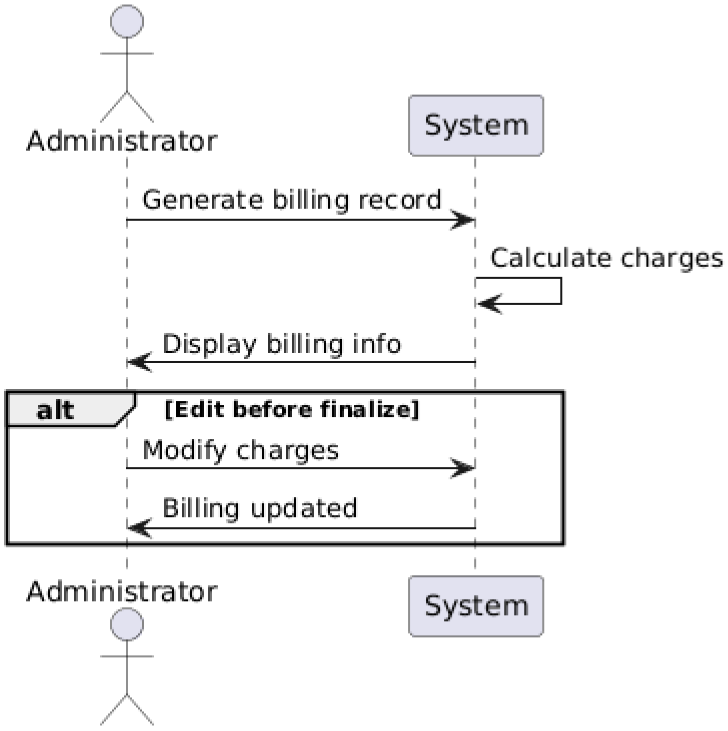


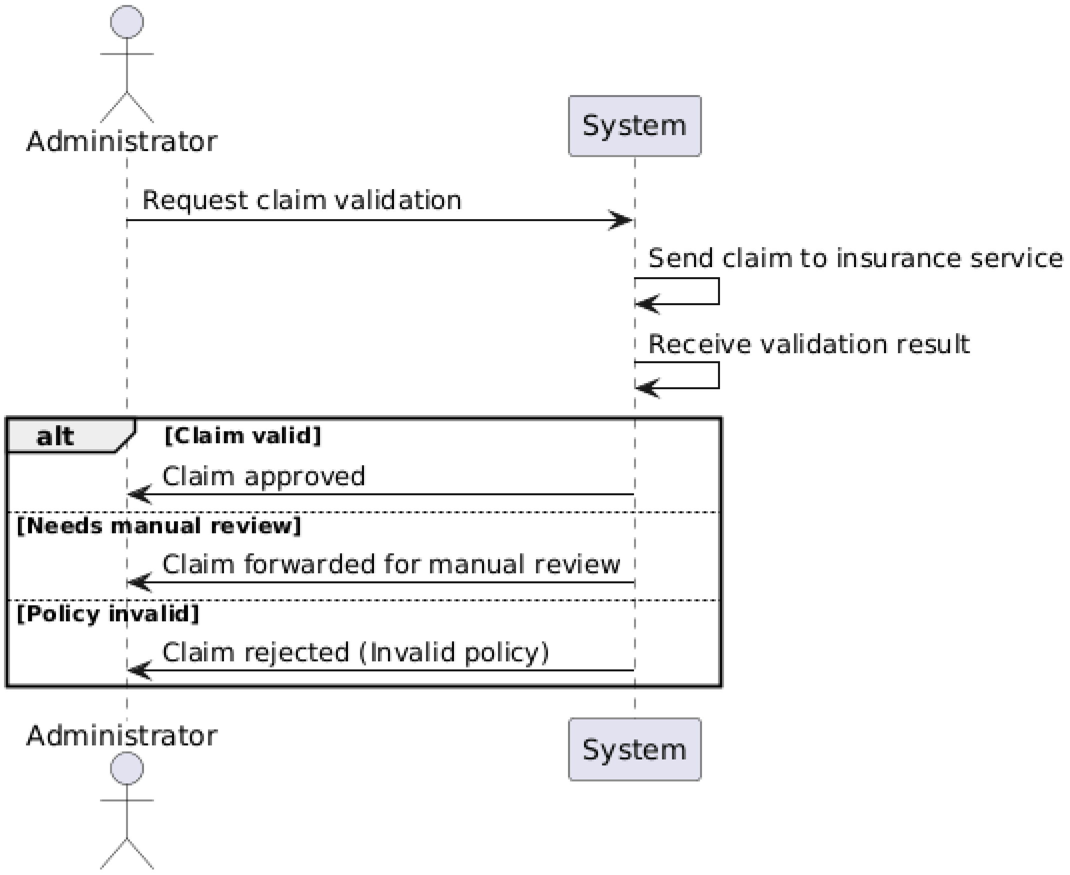


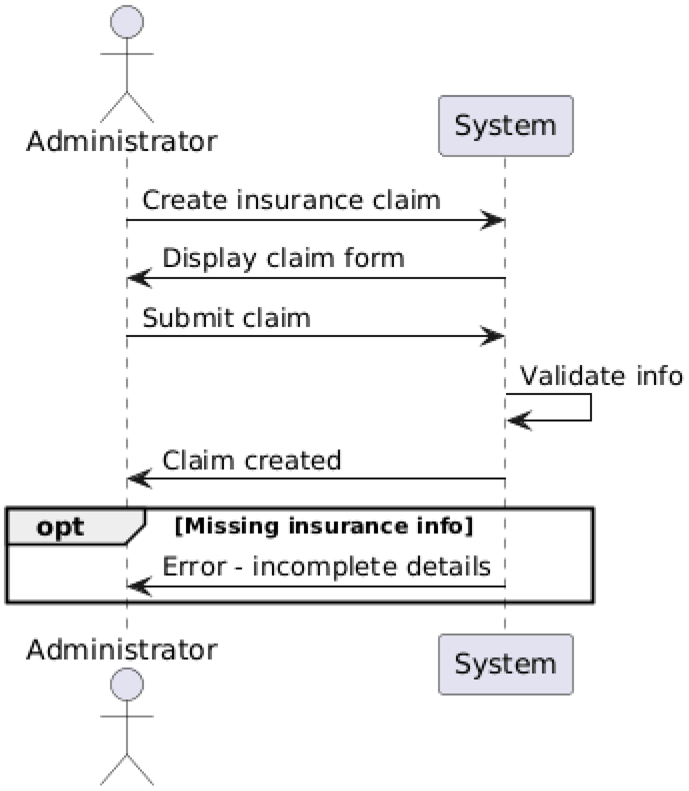

Doctor
System
Modify prescription
Display prescription
Submit changes
Updated successfully
alt
[Cancel prescription]
Cancel prescription
Prescription cancelled
Doctor
System

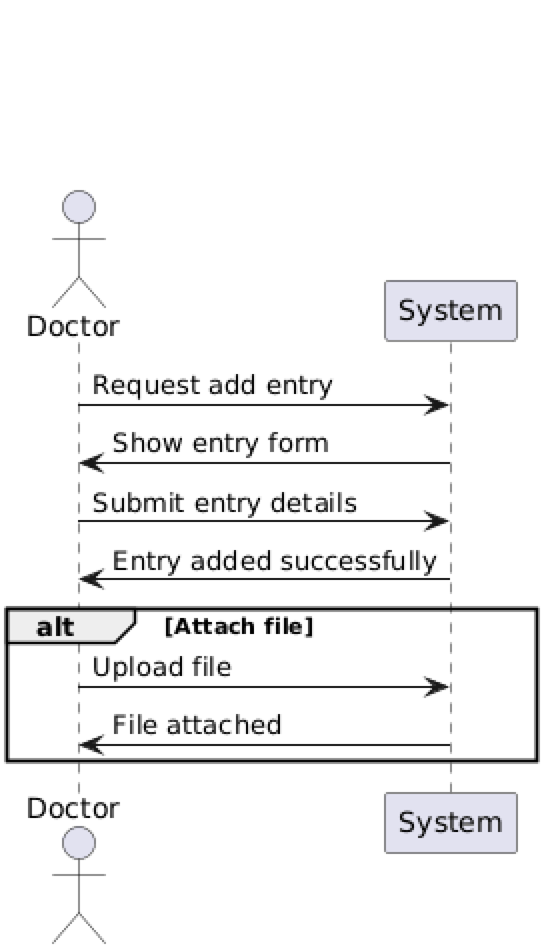

PharmacyStaff
System
Verify prescription
Check prescription validity
alt
[Prescription valid]
Prescription approved for dispensing
[Need clarification]
Request clarification from doctor
Verification pending clarification
[Prescription invalid]
Prescription invalid or expired
PharmacyStaff
System

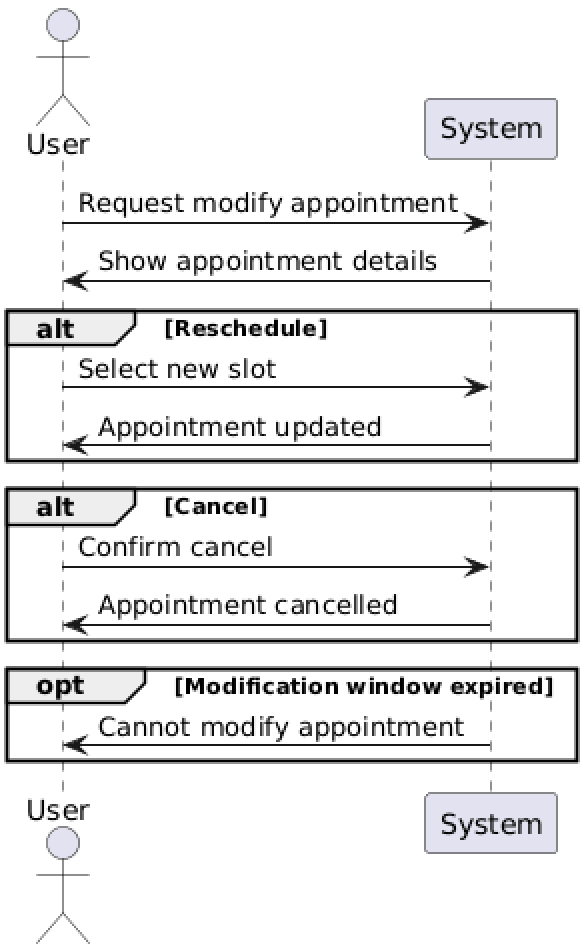

Administrator
System
Manage user roles
Display user list
Assign role
Role updated
alt
[Revoke role]
Remove role
opt
[Cannot modify admin]
Action denied
Administrator
System

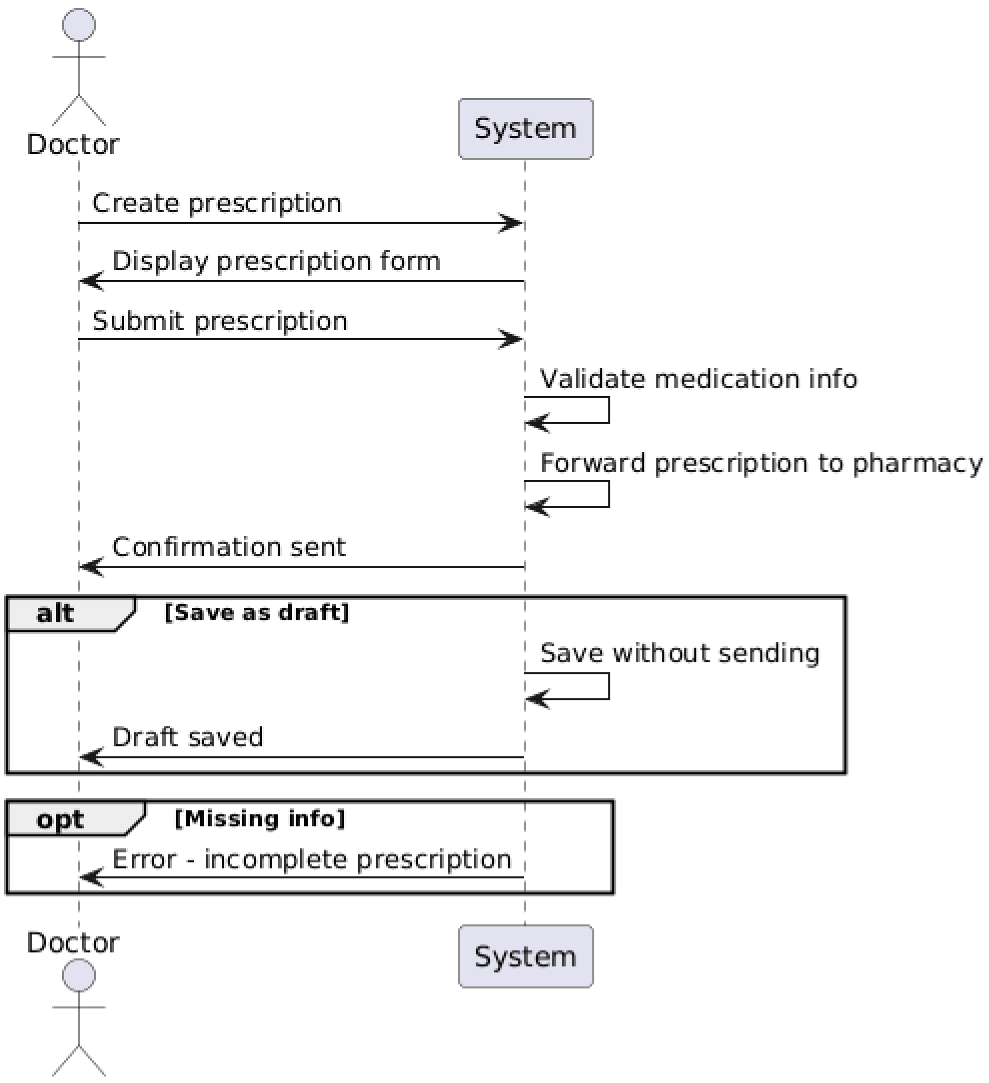

Administrator
System
Generate billing record
Calculate charges
Display billing info
alt
[Edit before finalize]
Modify charges
Billing updated
Administrator
System

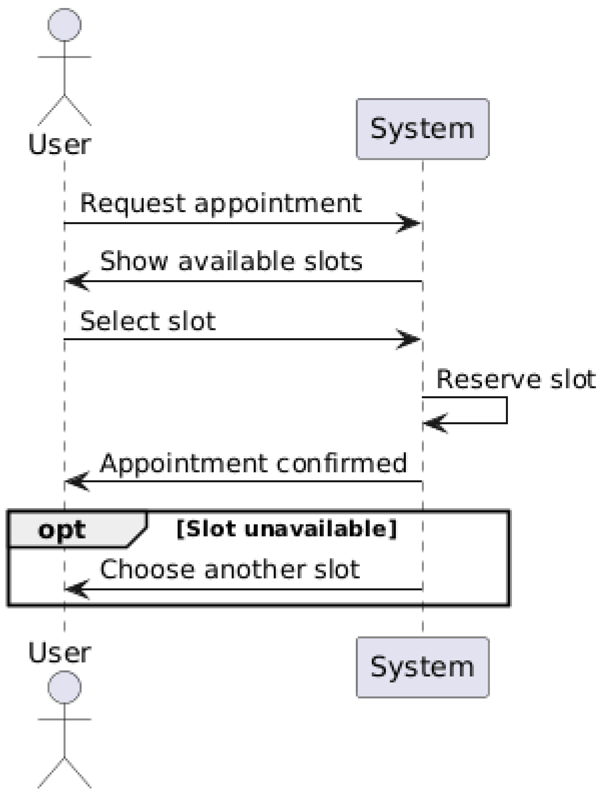

Administrator
System
Request claim validation
Send claim to insurance service
Receive validation result
alt [Claim valid]
Claim approved
[Needs manual review]
Claim forwarded for manual review
[Policy invalid]
Claim rejected (Invalid policy)
Administrator
System

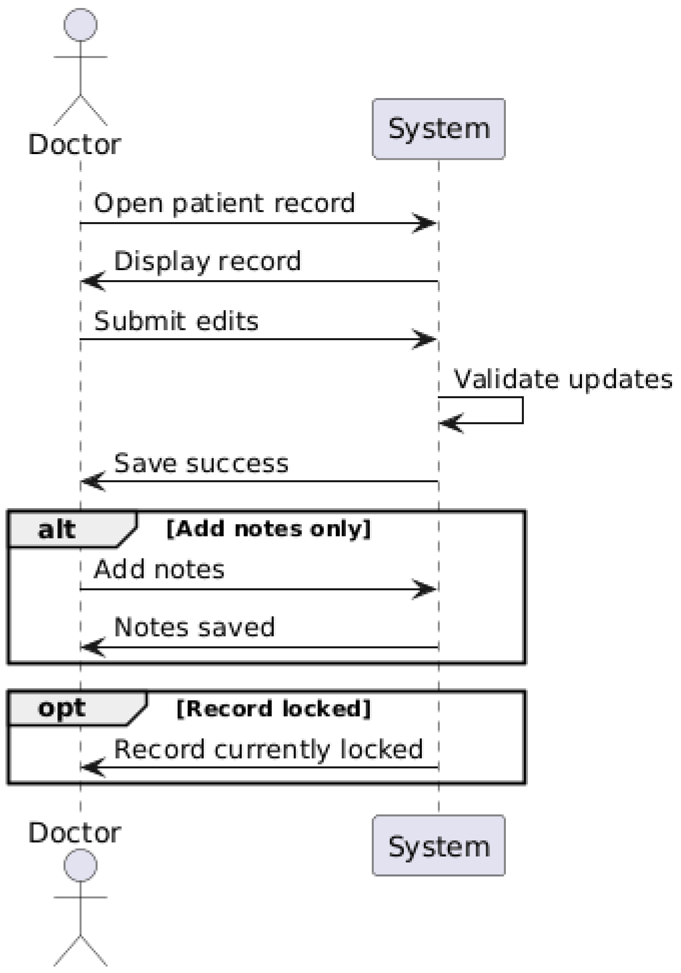

Administrator
System
Create insurance claim
Display claim form
Submit claim
Validate info
Claim created
opt [Missing insurance info]
Error - incomplete details
Administrator
System

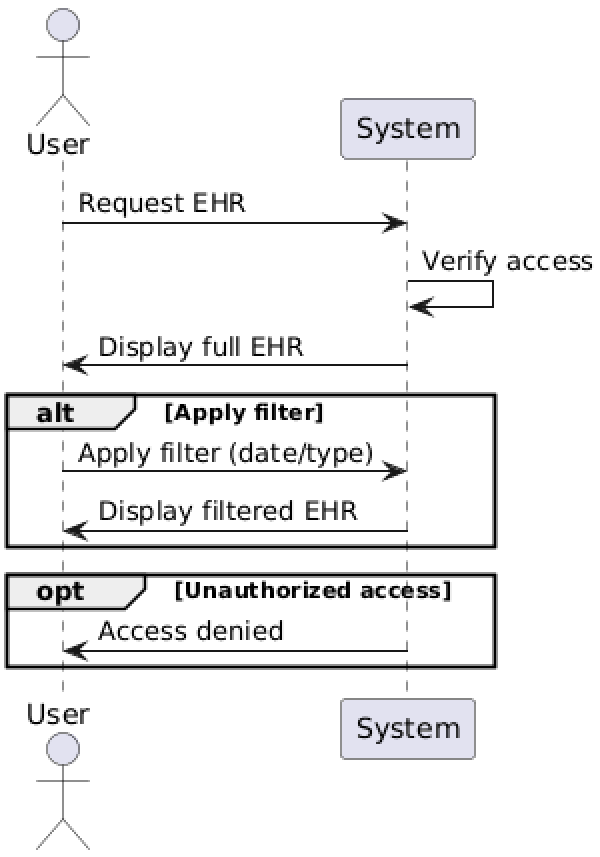

Doctor
System
Request add entry
Show entry form
Submit entry details
Entry added successfully
alt [Attach file]
Upload file
File attached
Doctor
System

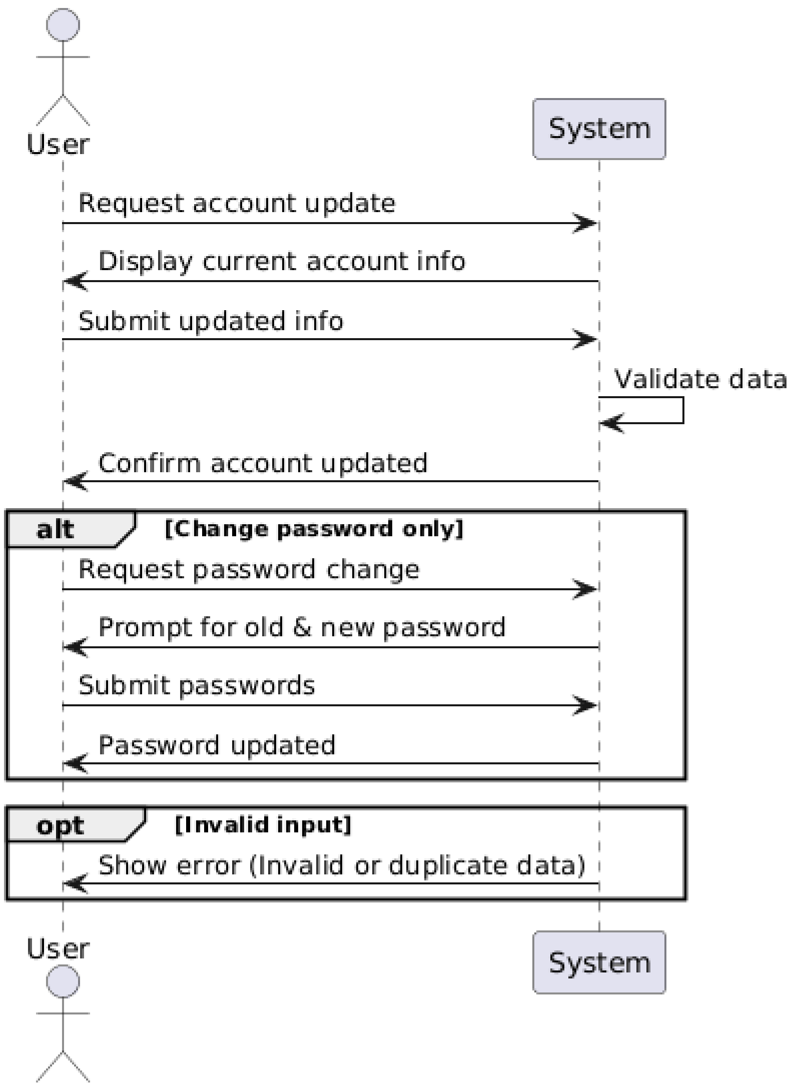

User
System
Request modify appointment
Show appointment details
alt [Reschedule]
Select new slot
Appointment updated
alt [Cancel]
Confirm cancel
Appointment cancelled
opt [Modification window expired]
Cannot modify appointment
User
System


Doctor
System
Create prescription
Display prescription form
Submit prescription
Validate medication info
Forward prescription to pharmacy
Confirmation sent
alt
[Save as draft]
Save without sending
Draft saved
opt
[Missing info]
Error - incomplete prescription
Doctor
System


User
System
Request appointment
Show available slots
Select slot
Reserve slot
Appointment confirmed
opt
[Slot unavailable]
Choose another slot
User
System


Doctor
System
Open patient record
Display record
Submit edits
Validate updates
Save success
alt [Add notes only]
Add notes
Notes saved
opt [Record locked]
Record currently locked
Doctor
System


User
System
Request EHR
Verify access
Display full EHR
alt
[Apply filter]
Apply filter (date/type)
Display filtered EHR
opt
[Unauthorized access]
Access denied
User
System



6. State Machine Diagrams (Two Complex Objects)

6.1 State Machine: Appointment

Rationale for Selecting Appointment

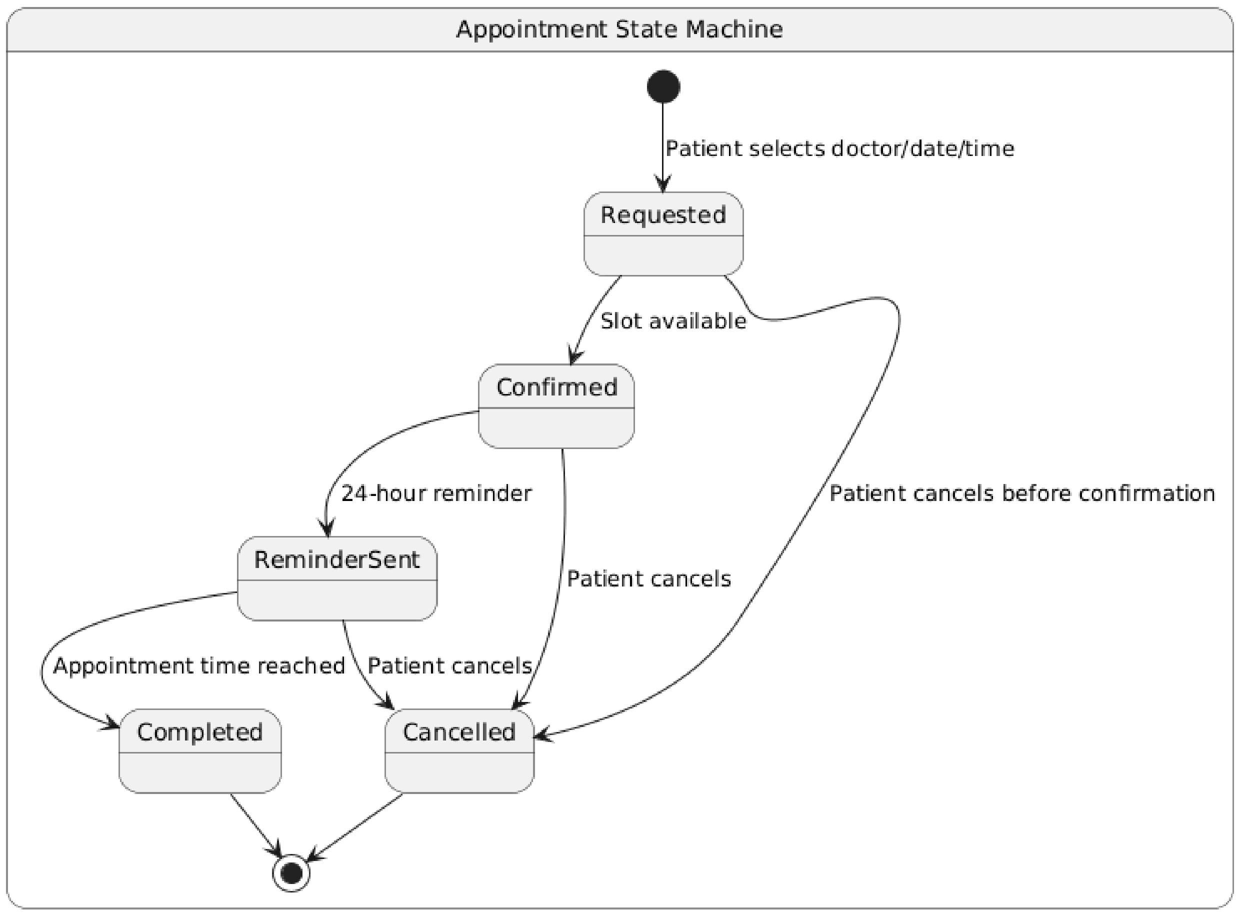
The **Appointment** object has one of the most dynamic lifecycles in the Medalyze system. It interacts with multiple actors (Patient, Doctor, System) and progresses through several operational stages. Important time-dependent actions—such as system reminders and patient-initiated modifications—create multiple branching transitions, making it ideal for behavioral modeling.

Key reasons:

- Multiple user-driven transitions (request → confirm → cancel)
- System-triggered actions (automatic reminder)
- Terminal states dependent on real-world events (completion vs cancellation)
- Clear primary and alternative flows

This complexity justifies modeling Appointment with a dedicated state machine diagram.

Medalyze - State Machine Diagram 1 - Appointment



6.2 State Machine: Prescription

Rationale for Selecting Prescription

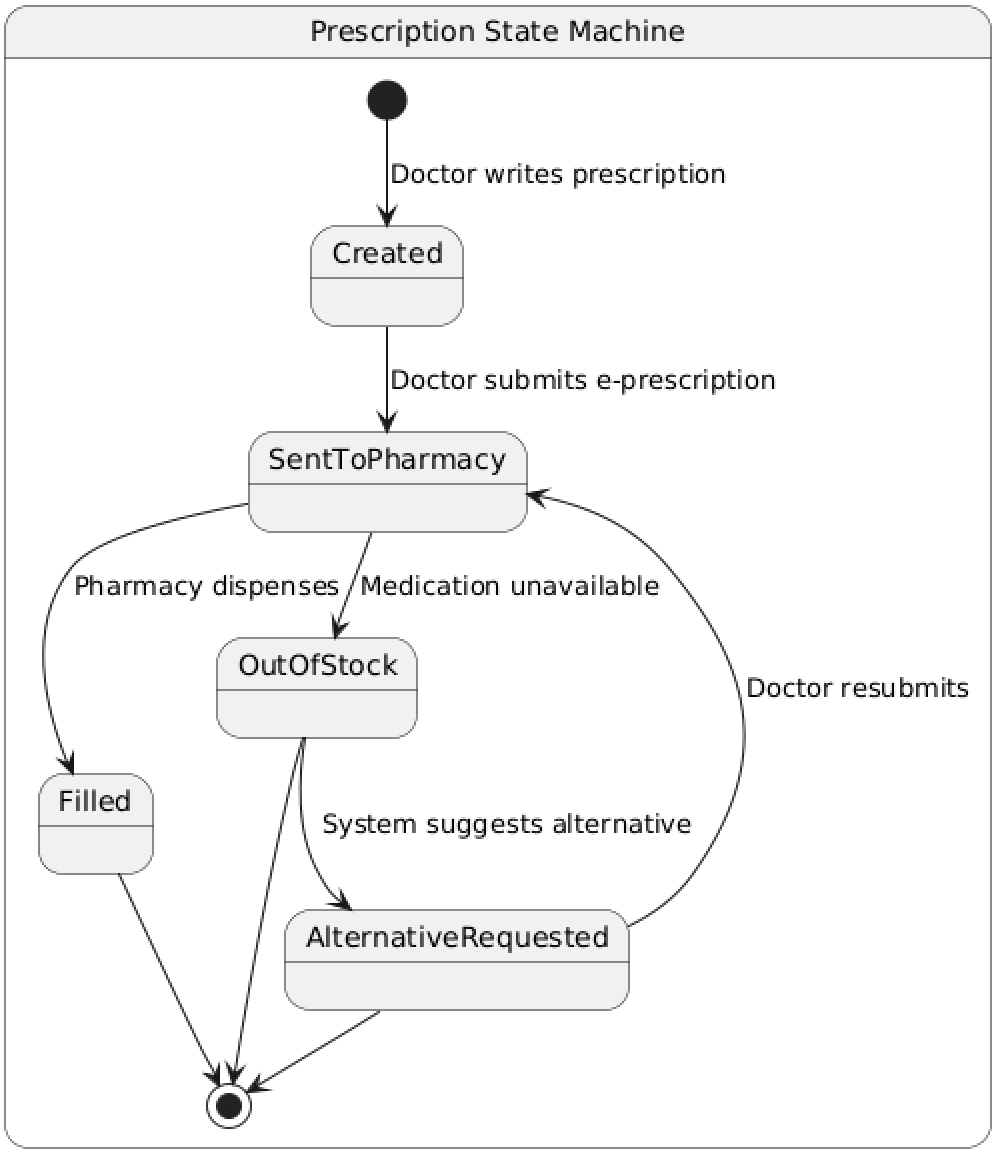
The **Prescription** object also demonstrates a complex workflow that spans both clinical and pharmacy operations. Its lifecycle includes conditional branching, error handling, and feedback loops triggered by inventory shortages. It is influenced by decisions from multiple roles (Doctor, Pharmacy Staff, System), and includes recovery paths—such as alternative medication requests—making it highly suitable for state-based behavioral analysis.

Key reasons:

- Multi-actor transitions (Doctor → Pharmacy Staff → System)
- Conditional outcomes (Filled vs OutOfStock)
- Exception handling (alternative medication loop)
- System-driven notifications and callback behavior
- A well-defined terminal state that varies based on outcomes

These characteristics make Prescription an excellent candidate for the second state machine diagram.

Medalyze - State Machine Diagram 1 (Prescription)



7. Conclusion

Deliverable #3 consolidates the core analytical and behavioral modeling work required to define Medalyze as a robust, well-structured healthcare information system. Through systematic noun extraction and classification, the final domain model accurately captures all essential entities involved in appointments, prescriptions, records, billing, and administrative operations. The CRUD matrix validated these domain classes against the refined use case set, revealing gaps that led to meaningful corrections—such as adding missing creation/update actions and splitting overloaded use cases into clearer, more maintainable ones.

Behavioral models—including Activity Diagrams, System Sequence Diagrams, and State Machine Diagrams—provided deeper insight into how the system responds to user actions and internal events. These models clarified workflow logic, validated process consistency, and highlighted critical system behaviors in dynamic objects like Appointments and Prescriptions.

Overall, this deliverable transforms the Medalyze system from a high-level concept into a detailed, behavior-driven design blueprint. It establishes the structural and dynamic foundations needed for the next development phases, ensuring that future design and implementation activities are guided by complete, accurate, and validated models.