Final Report

Smart Clinic

A Streamlined Patient Management System for Small Clinics

Group 3

Areej Asiri

Minahil Haroon Khalid

Ram Kishore Karuppiah Nadar Venkateswaran

College of Computing and Informatics, Drexel University

INFO 620-001 Information Systems Analysis & Design

Dr. Chad Peiper

1. The Problem Statement

1.1 Context:

Many small clinics in the U.S. still operate using paper records or overly complex systems designed for large hospitals. These small facilities often face high costs and steep learning curves when attempting to digitize operations. The result is inefficient patient management, lack of historical data, and administrative overhead.

1.2 Goals:

To develop a cost-effective, user-friendly patient care clinic system that allows a small clinic to manage basic functionalities such as appointments, patient information, medical histories (symptoms, diagnoses, treatments), and payments.

1.3 Importance:

A streamlined system will reduce administrative workload, improve care quality through accessible patient records, and enable better scheduling and revenue tracking. It supports small clinics transitioning to digital platforms without incurring high costs or steep learning barriers.

1.4 Scope:

1.4.1 In-Scope Functionalities:

- Patient registration and profile management onboarding.
- Appointment scheduling and calendar view.
- Recording of medical symptoms, treatments and diagnoses.
- Payment tracking (from patients and insurance).
- Limited user roles (doctors, nurses, admin staff).
- Basic reporting (e.g., upcoming appointments, patient visit history).
- Web portal for users.

1.4.2 Out-of-Scope Functionalities:

- Integration with insurance companies.
- Prescription management.
- External lab system integration.
- Complex billing procedures.
- Mobile application version.

2. Requirements:

2.1 Functional Requirements:

- Add, update, delete patient records.
- Schedule, modify, and cancel appointments.
- Record visit details: symptoms, diagnoses, treatments.
- Record and view payment transactions.
- User authentication and role-based access
- Web portal for patients.

2.2 Data Requirements:

- **Patient table:** ID, Name, DOB, Contact, Address, Insurance Info.
- **Appointment table:** ID, Patient ID, Doctor ID, Date, Time, Status.
- **Medical History table:** Visit ID, Patient ID, Date, Symptoms, Diagnosis, Treatment.
- **Payment table:** Payment ID, Patient ID, Amount, Date, Payment Type (Patient/Insurance).
- **User table:** ID, Role, Username, Password

2.3 Business Rules and Logic:

- Appointments cannot be scheduled in the past.
- A doctor cannot have overlapping appointments.
- Patients must complete the onboarding process before scheduling an appointment.
- Only authorized roles can edit patient medical records.

2.4 Non-Functional Requirements:

- Security: Basic login system with encrypted passwords and met MFA criteria.
- Usability: Clean UI with calendar-based appointment management.
- Performance: Fast response time for operations involving up to 10,000 records.
- Reliability: 99.5% uptime; basic backup functionality.
- Maintainability: Modular design to support future features.

2.5 Important Assumptions:

- The clinic has at most three doctors, five nurses and fewer than five administrative staff.
- One centralized database.
- Minimal training required for staff.
- Medical coding handled externally (ICD9 reference database).

3. Examples of System Input and Output Scenarios

Scenario 1: Scheduling an Appointment:

- **Input:** Patient ID, Doctor ID, Date, Time.
- **Output:** Confirmation message; Calendar updated.

Scenario 2: Recording Medical History:

- **Input:** Visit ID, Patient ID, Symptoms, Diagnosis, Treatment.
- **Output:** Record Stored; Medical history updated.

Scenario 3: Payment Recording:

- **Input:** Patient ID, Amount, Payment Type.
- **Output:** Transaction added; Balance updated.

Scenario 4: Viewing Appointment Calendar:

- **Input:** Doctor ID, Date Range.
- **Output:** List of scheduled appointments.

4. Knowledge Acquisition

Learning Approach:

- Research through academic and industry sources on small clinic operations.
- Interviews or questionnaires with local clinic staff (if possible).
- Review of existing minimal electronic health record systems.

Problem Origin:

- Based on real challenges faced by small clinics that lack affordable and simple digital tools.

5. Proposed Deliverables and Work Plan

Deliverables:

- Proposal.
- User Stories.
- Use Case Diagram.
- Use Case Descriptions.
- Class Diagram (domain model).
- Design Class Diagram.
- System Sequence Diagram.
- Sequence Diagrams.
- Relational Database Schema.
- Analysis Selection.

Work Plan:

Week 1: Complete Project Proposal.

Week 2: Write user stories and create a use case diagram and its explanation.

Week 3: Write use case descriptions and create a class diagram (domain model) and its explanation.

Week 4: Create system sequence, sequence, and design class diagrams and write the explanation for the design class diagram.

Week 5: Create a relational database schema and write the analysis of selection.

6. Actors and Goals

ACTORS	GOALS/USE CASE

Receptionist	Process Patient Check-In, Set Appointment, Reschedule Appointment, Cancel Appointment, Process Payment.
Patient Portal	Fill Out Forms, Book/Reschedule/Cancel Appointments, Access Medical Records, Pay Bills.
Nurse	Record Vitals, Perform Diagnostic Tests, Administer Medication.
Doctor	Write Doctor Notes (prescriptions, tests, referrals, treatments), Order Tests, Diagnose, Write Prescriptions, Write Referrals.
Billing Specialist	Bill Insurance Company, Process Claims.
Pharmacy	Fills Prescriptions Ordered by Doctor.
External Medical Code Database	Provide Up-to-Date Medical Codes for Billing and Claims.
External Cloud Storage Host	Store Protected Patient PHI Securely, Provide Backup for Data.
Bank	Process Payments via Credit, Debit, or Check, Authorize Patient Payments.
Insurance Company	Approve/Deny Claims.

6.1. Main Success Scenarios (User Stories):

1. **Doctor Writes Visit Notes:**

   The doctor fills in a pre-made template of routine questions asked during the office visit (e.g., verifying social history, general medical history, current medications). After the office visit, the doctor adds any additional notes, such as prescriptions, tests, referrals, and treatments. These records are stored securely and accessible to authorized healthcare providers.

2. **Nurse Supports Doctor:**

   The nurse plays a support role in preparing the patient for the doctor's exam by taking vital signs and preparing preliminary information such as health history and medications. If the patient requires any diagnostic tests or vaccinations, the nurse will perform these as directed by the doctor.

3. **Receptionist Processes Payment:**

   At the end of the visit, the patient pays their bill at the front desk. Payments can be made via credit card, debit card, cash, or check. The receptionist processes the payment, generates an invoice, and updates the system.

4. **Receptionist Checks-In Established Patient:**

   The receptionist verifies if there are any changes in the patient's information (e.g., contact details, insurance) and updates it accordingly. For new patients, the receptionist ensures they fill out the required forms and enters the data into the system.

5. **Insurance Claim Submission:**

   The accountant reviews the patient's record and submits the bill to the insurance company. The medical codes for the patient's treatments are verified using the external medical code database. The insurance company then approves or denies the claim.

6. **Patient Accesses Their Medical Records:**

   The patient logs into the patient portal to view their visit summaries, treatments, medications, lab results, and upcoming appointments. The system ensures that all information is available to the patient in compliance with HIPAA regulations.

7. **Receptionist Cancels Appointments in Advance:**

   If a patient requests to cancel an appointment, the receptionist ensures that the request is

processed at least 24 hours prior to the scheduled time. If the doctor has a scheduling conflict, the receptionist offers alternative appointment times.

8. **Handling Medical Confidentiality:**

   Only authorized medical staff (under HIPAA guidelines) can access patient medical records. The system ensures that non-medical staff, such as receptionists, cannot view confidential health information.

9. **Tracking Appointment Lateness:**

   If a patient is repeatedly late for their appointments, the system automatically flags the lateness and notifies the receptionist. After three late occurrences, the patient may be asked to review the clinic's policies or risk dismissal from the clinic.

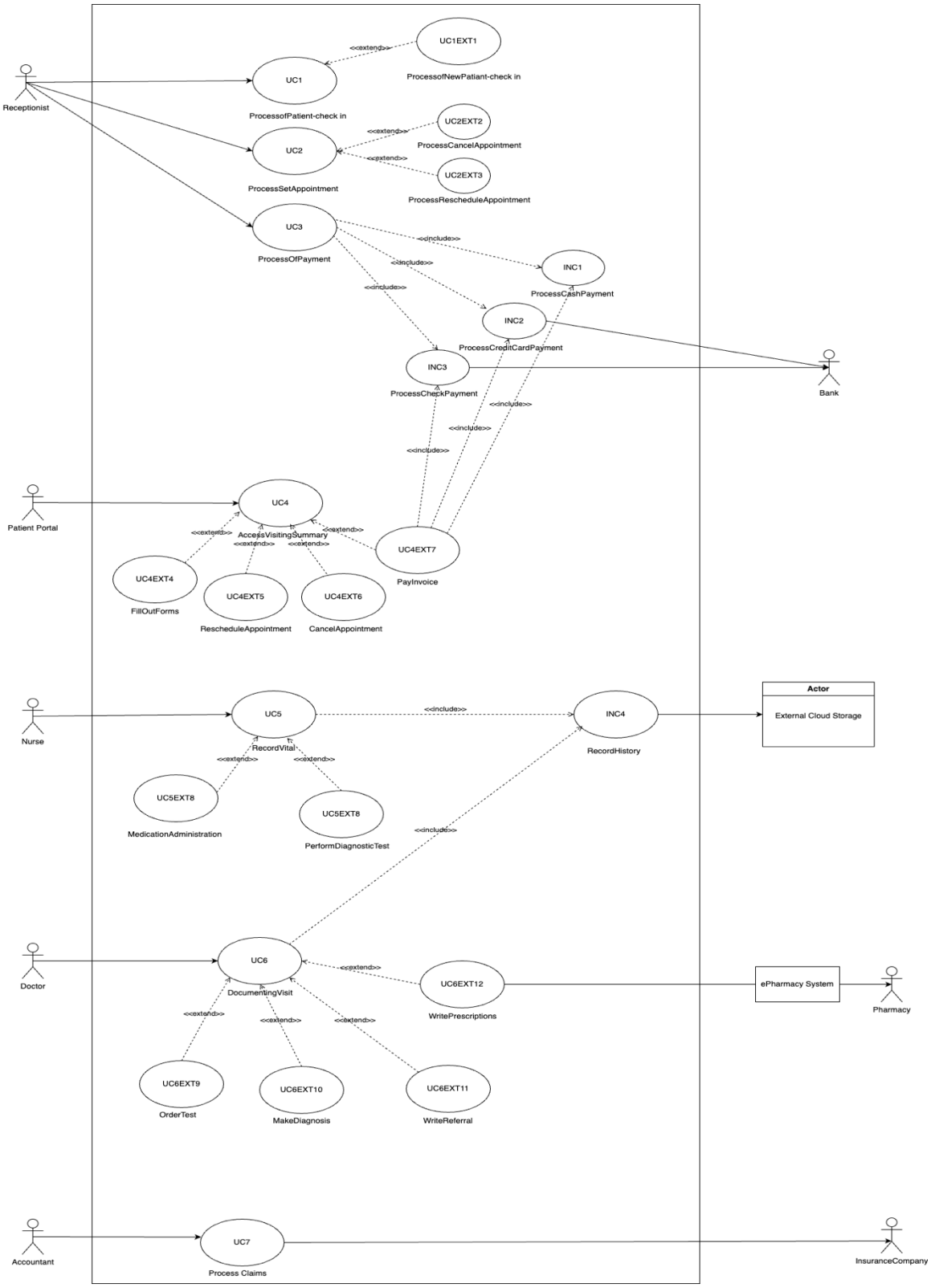
10. **Backup and Recovery in Case of System Failure:**

    In the event of a system failure where the clinic cannot access patient data for more than 15 minutes, the system will automatically initiate a recovery process. All patient data is backed up securely in compliance with HIPAA.

11. **Automatic Data Encryption, Upload, and Storage:**

    When new Protected Health Information (PHI) is entered by the doctor or nurse, it is temporarily stored on the device. The data is then encrypted and uploaded to a secure cloud storage, ensuring that only authorized users can access it.

6.2. Use Case Diagram:


Receptionist
UC1
ProcessofPatient-check in
UC1EXT1
ProcessofNewPatiant-check in
<<extend>>
UC2
ProcessSetAppointment
UC2EXT2
ProcessCancelAppointment
UC2EXT3
ProcessRescheduleAppointment
UC3
ProcessOfPayment
<<include>>
INC1
ProcessCashPayment
INC2
ProcessCreditCardPayment
INC3
ProcessCheckPayment
Bank
Patient Portal
UC4
AccessVisitingSummary
UC4EXT4
FillOutForms
UC4EXT5
RescheduleAppointment
UC4EXT6
CancelAppointment
UC4EXT7
PayInvoice
Nurse
UC5
RecordVital
UC5EXT8
MedicationAdministration
UC5EXT8
PerformDiagnosticTest
INC4
RecordHistory
Actor
External Cloud Storage
Doctor
UC6
DocumentingVisit
UC6EXT12
WritePrescriptions
ePharmacy System
Pharmacy
UC6EXT9
OrderTest
UC6EXT10
MakeDiagnosis
UC6EXT11
WriteReferral
Accountant
UC7
Process Claims
InsuranceCompany

USE CASE	TYPE
UC1- ProcessofPatient-check in	base
UC1EXT1- ProcessofNewPatiant-check in	<<extend>>
UC2 - ProcessSetAppointment	base
UC2EXT2 - ProcessCancelAppointment	<<extend>>
UC2EXT3 - ProcessRescheduleAppointment	<<extend>>
UC3 - ProcessOfPayment	base
INC1- ProcessCashPayment	<<include>>
INC2 - ProcessCreditCardPayment	<<include>>
INC3 - ProcessCheckPayment	<<include>>
UC4 - AccessVisitingSummary	base
UC4EXT4 - FillOutForms	<<extend>>
UC4EXT5 - RescheduleAppointment	<<extend>>
UC4EXT6 - CancelAppointment	<<extend>>
UC4EXT7 - PayInvoice	<<extend>>
UC5 - RecordVital	base
UC5EXT8 - MedicationAdministration	<<extend>>
UC5EXT8 - PerformDiagnosticTest	<<extend>>
INC4 - RecordHistory	<<include>>
UC6 - DocumentingVisit	base
UC6EXT9 - OrderTest	<<extend>>
UC6EXT10 - MakeDiagnosis	<<extend>>

UC6EXT11 - WriteReferral	<<extend>>
UC6EXT12 - WritePrescriptions	<<extend>>
UC7 - Process Claims	base

6.3. Use Case Diagram Explanation:

The Doctor's primary goals are related to examining the patient and documenting the findings, such as prescribing medication, ordering tests, writing referrals, and documenting patient information. Some use cases extend to performing tests, ordering prescriptions, and writing referrals, which are connected to the Pharmacy for prescription fulfillment.

The Nurse supports the doctor by performing diagnostic tests, recording patient vitals, and administering medications or vaccinations. The nurse's tasks are closely tied to patient care and are documented in the patient's medical record.

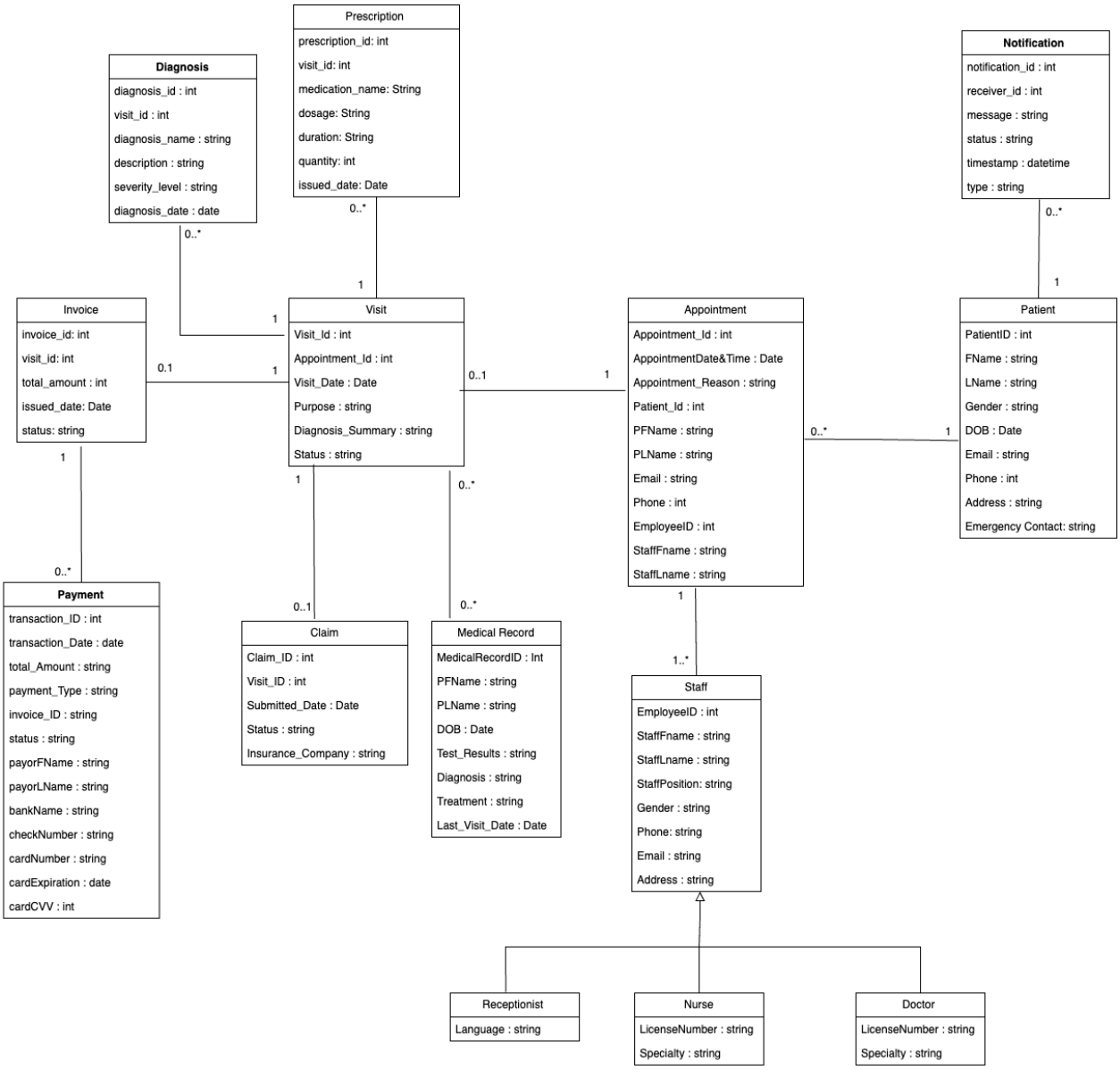
The Receptionist is tasked with managing patient appointments, including cancellations and rescheduling. The receptionist also handles patient check-ins and processes payments through the front desk.

The accountant reviews the medical codes and submits claims to the Insurance Company. The system uses an external medical code database to ensure that the appropriate codes are applied.

The Patient interacts with the system through the patient portal, where they can access visit summaries, view bills, schedule or cancel appointments, and manage payments.

External systems like External Cloud Storage and External Medical Code Database play key roles in securely storing PHI and providing updated medical codes for insurance claims processing.

7. Patient Care Clinic System – Class Diagram:



This class diagram represents the main structure of the Patient Care Clinic System (PCCS). It defines the relationships between various components of a healthcare visit, including patients, appointments, visits, billing, payments, claims, and prescriptions. The diagram focuses on capturing the key classes, their attributes, and the relationships between them to support system development and documentation.

The system begins with the Patient class, which stores personal information such as name, gender, date of birth, contact details, and emergency contact information. A patient can book multiple Appointments, each of which includes details such as the appointment date, reason, and links to the patient and the assigned staff member. Each appointment is handled by a staff member and can result in a Visit, which records the actual encounter between the patient and the healthcare provider, including the purpose, diagnosis summary, and status.

From a visit, the system can generate an Invoice for billing purposes. Each invoice includes the total amount, the issued date, and the payment status. Patients can settle their invoices through one or more Payments. The Payment class is further divided into three specialized payment types: Cash, Check, and Credit Card, each with its specific attributes. For instance, Cash includes the received amount and change, while Check stores the check number and bank details, and Credit Card handles card type, number, expiration date, and CVV.

Each Visit may also create an Insurance Claim, which records the claim's submission date, status, and associated insurance company. In addition, a visit can issue one or more Prescriptions, which include information such as medication name, dosage, duration, and quantity.

The Staff class represents all employees and includes general attributes like name, position, contact details, and address. It serves as a parent class for three specialized roles: Receptionist, Nurse, and Doctor, each with additional attributes like language, license number, or specialty. Staff members are responsible for managing appointments and updating Medical Records, which store diagnosis, test results, treatment plans, and visit history for each patient.

In summary, this class diagram captures the essential building blocks of a patient-centered healthcare system, showing how administrative, medical, and financial components work together in a structured and logical way.

7.1 Use Case Descriptions:

USE CASE #	**UC1**
USE CASE Name	Process of Patient Check In
ACTOR	Receptionist
Goal (1 phrase)	To check in a patient (new or existing) upon arrival at the healthcare facility.
Overview and scope	This use case describes how the receptionist performs the patient check-in process, which includes verifying patient information and logging their arrival. If the patient is new, additional information is collected. This use case ensures that patients are properly registered before being seen by medical staff.
Level	Base
Preconditions	• The receptionist is logged into the system. • The patient has an appointment scheduled in the system.

Postconditions in words (write in passive and past tense)	• **Appointment.Status** was updated to "Checked-in" to indicate the patient's arrival. • **Appointment.AppointmentDate&Time**, **Appointment.Appointment_Reason**, and patient details (**Appointment.PFName**, **Appointment.PLName**, **Appointment.Email**, **Appointment.Phone**) were confirmed by the receptionist. • If the patient was new, **Patient.PatientID**, **Patient.FName**, **Patient.LName**, **Patient.Gender**, **Patient.DOB**, **Patient.Email**, **Patient.Phone**, **Patient.Address**, and **Patient.Emergency Contact** were captured during the registration process. • **Staff.EmployeeID**, **Staff.StaffFname**, **Staff.StaffLname**, **Staff.StaffPosition**, and **Staff.Gender** were recorded to associate the check-in action with the receptionist. • **Staff.Phone**, **Staff.Email**, and **Staff.Address** were stored for traceability and audit logging. • A timestamp was created and stored as part of system logs to confirm the check-in event (referenced implicitly

<table>
<tr><td></td><td colspan="2">in audit records, not shown in the diagram but mentioned in business rules).

• The system saved all updates in the database, completing the check-in process and readying the patient for the scheduled appointment.</td></tr>
<tr><td>Trigger</td><td colspan="2">• A patient arrives at the reception for a scheduled appointment.</td></tr>
<tr><td>Included Use Cases</td><td colspan="2">None</td></tr>
<tr><td>Extending Use Cases</td><td colspan="2">UC1EXT1</td></tr>
<tr><td rowspan="4">MAIN SUCCESSFUL SCENARIO <u>for this Use Case</u>
in numbered sequence

Reference “included use cases” in this section using INCLUDE ius_name</td><td>Actor Action</td><td>System Action</td></tr>
<tr><td>Step 1. The receptionist Initiates the check-in process.</td><td>Step 2. System prompts for patient identification.</td></tr>
<tr><td>Step 3. Receptionist enters patient’s name or ID.</td><td>Step 4. System retrieves patient data.</td></tr>
<tr><td>Step 5. The receptionist confirms appointment and details.</td><td></td></tr>
</table>

	Step 6. The receptionist verifies the appointment time and confirms the patients identity.	Step 7. System updates appointment status to “Checked-in”.
	Step 8. If the patient is new, the receptionist triggers new patient registration.	Step 9. System saves the check-in data.
		Step 9. System confirms the check-in is complete.
	Actor Action	**System Action**
UNSUCCESSFUL SCENARIOS (erroneous situations)	Step 1: The patient arrives without a scheduled appointment.	Step 2: The system fails to locate any appointments for the patient.
EXTEND UC1EXT1 - Processof New Patient Check-in:	Step 3: The receptionist attempts to access appointment data.	Step 4: The system fails to respond due to connectivity issues.

	Step 5: The receptionist asks for identity verification.	
	Step 6: The patient is unable to provide valid identification.	Step 7: The system prevents check-in without identity confirmation.
Priority in scheduling	High	
Frequency	Frequently	
Business rules and data logic	• Identity must be confirmed via ID or system profile. • A check-in cannot proceed if the appointment is canceled or expired. • New patients must complete registration before check-in is finalized. • Each check-in must be timestamped and logged in audit records.	
Other non-functional requirements	• Check-in must be processed within 2 minutes. • System must display confirmation immediately after check-in.	
Superordinates	None	
Developer	Group 3 – Areej, Ram, Minahil	

Creation date and last modified date	Created: 05/19/2025 Updated: 06/08/2025
Other Comments	• Ensure fallback workflows exist for offline check-in in case of system outage. • Enable exporting daily check-in logs for administrative reports.

USE CASE #	**UC2**
USE CASE Name	Process of Setting Appointment
ACTOR	Receptionist
Goal (1 phrase)	To schedule a new appointment for a patient.
Overview and scope	This use case describes how the receptionist sets up a new appointment for a patient using the clinic's appointment system. The process includes selecting the patient, choosing the date and time, assigning a provider if needed, and confirming the appointment details.
Level	Base
Preconditions	• The receptionist selects the option to set an appointment for a patient.

Postconditions in words (write in passive and past tense)	• **Appointment.Appointment_ID** was generated to uniquely identify the new appointment. • **Appointment.PFName**, **Appointment.PLName**, **Appointment.Email**, and **Appointment.Phone** were retrieved and associated with the selected patient. • **Appointment.AppointmentDate&Time** was recorded to reflect the selected schedule for the visit. • **Appointment.Appointment_Reason** was captured to describe the purpose of the visit. • **Appointment.EmployeeID**, **Appointment.StaffFname**, and **Appointment.StaffLname** were saved to link the assigned provider or staff member responsible for the appointment. • **Staff.EmployeeID**, **Staff.StaffFname**, **Staff.StaffLname**, **Staff.StaffPosition**, and **Staff.Gender** were stored to associate the appointment action with the receptionist who performed the scheduling. • **Staff.Phone**, **Staff.Email**, and **Staff.Address** were logged to support communication and audit trail. • **Patient.PatientID**, **Patient.FName**, **Patient.LName**, and **Patient.Email** were used to match and confirm patient identity before scheduling.

<table>
<tr><td></td><td colspan="2">• The new appointment was saved into the system with status implicitly set to active (status attribute not explicitly present but implied as part of successful booking).
• Timestamp and scheduling metadata were recorded in the system logs to confirm the successful scheduling event.</td></tr>
<tr><td>Trigger</td><td colspan="2">• The receptionist selects the option to set an appointment for a patient.</td></tr>
<tr><td>Included Use Cases</td><td colspan="2">None</td></tr>
<tr><td>Extending Use Cases</td><td colspan="2">UC2EXT2 – Process Cancel Appointment
UC2EXT3 – Process Reschedule Appointment</td></tr>
<tr><td rowspan="3">MAIN SUCCESSFUL SCENARIO <u>for this Use Case</u>
in numbered sequence</td><td>Actor Action</td><td>System Action</td></tr>
<tr><td>Step 1. Receptionist initiates the “Set Appointment” process.</td><td>Step 2. System prompts for patient ID or search.</td></tr>
<tr><td>Step 3. Receptionist selects the patient.</td><td>Step 4. System displays available time slots.</td></tr>
</table>

Reference “included use cases” in this section using INCLUDE *ius_name*	Step 5. The receptionist Selects the date and time of the appointment.	
	Step 6. The receptionist Confirms and saves the appointment.	Step 7. System creates the appointment and stores it.
		Step 8. System confirms the appointment has been scheduled.
OTHER SUCCESSFUL SCENARIOS (Specify any successful variations of thenormal execution path, including any extension points Using EXTEND eus_name) ***Extend UCEXT 2: Process Cancel Appointment***	**Actor Action**	**System Action**
	Step 1: The patient or receptionist decides to cancel the appointment instead of setting up one.	
	Step 2: The cancellation process is initiated (see UC2EXT2).	

OTHER SUCCESSFUL SCENARIOS (Specify any successful variations of thenormal execution path, including any extension points Using EXTEND eus_name)	**Actor Action**	**System Action**
	Step 1: The patient requests a different appointment time.	
EXTEND UC2EXT3: Process Reschedule Appointment	Step 3: The rescheduling process is initiated (see UC2EXT3).	
UNSUCCESSFUL SCENARIOS (erroneoussituations)	Step 1: The receptionist searches for available slots.	Step 2: The system returns no available appointments.
No available time slots for requested date/time.	Step 3: The receptionist informs the patient and suggests alternative dates.	

UNSUCCESSFUL SCENARIOS (erroneoussituations)	Step 1: The receptionist attempts to book an appointment.	Step 2: The system fails to save the appointment
System error during booking.	Step 3: The receptionist informs the patient of the error and retries later.	
Priority in scheduling	High	
Frequency	Frequently	
Business rules and data logic	Appointment times must fall within working hours and avoid conflicts with other appointments. • Receptionist cannot schedule appointments for inactive or blocked patients. • Time slot availability is updated in real-time to prevent double booking.	
Other non-functional requirements	• User interface must be responsive and update availability without page reloads. • Appointment confirmation should be displayed immediately and optionally sent via email/SMS	

Superordinates	None
Developer	Group 3 – Areej, Ram, Minahil
Creation date and last modified date	Created: 05/19/2025 Updated: 06/08/2025
Other Comments	• Consider adding filters (e.g., provider specialty, appointment type) to simplify time slot selection.

USE CASE #	**UC2EXT2**
USE CASE Name	Process cancel appointment
ACTOR	Receptionist
Goal (1 phrase)	To cancel an existing appointment upon request.
Overview and scope	This use case describes how the receptionist cancels a previously scheduled appointment for a patient. The system allows the receptionist to search for the appointment, verify its details, and cancel it. The appointment is then removed from the active schedule and marked as canceled. This action may trigger notifications to the patient and the assigned provider if applicable.
Level	Extend
Preconditions	• The receptionist is logged into the system. • There is an active appointment scheduled in the system.

Postconditions in words (write in passive and past tense)	• **Appointment.Appointment_ID** was identified and retrieved based on patient search criteria. • **Appointment.PFName**, **Appointment.PLName**, and **Appointment.AppointmentDate&Time** were confirmed to validate the correct appointment. • **Appointment.Appointment_Reason**, **EmployeeID**, **StaffFname**, and **StaffLname** were reviewed for context before cancellation. • **Appointment status** was updated internally (though not explicitly shown in the class diagram, this status change is implied in the use case) to "Canceled" to reflect the action. • **Notification.Notification_ID** was generated and sent to **Notification.receiver_id**, which included either the **Patient.PatientID** or **Staff.EmployeeID**, depending on who needed to be notified. • **Notification.message** included the cancellation alert message and **Notification.timestamp** recorded the date and time of notification. • **Notification.status** was marked as "Sent" and **Notification.type** was labeled as "Cancellation".

	• **Staff.EmployeeID**, **Staff.StaffFname**, **Staff.StaffLname**, and **Staff.StaffPosition** were logged to record who canceled the appointment. • **Staff.Phone**, **Email**, and **Address** were stored for audit and future reference purposes. • The canceled appointment was removed from the active appointment schedule in the system and logged for audit purposes.	
Trigger	• The receptionist selects the cancel option during appointment management.	
Included Use Cases	None	
Extending Use Cases	UC1EXT1	
***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**	**Actor Action**	**System Action**
	Step 1. The receptionist Initiates the check-in process.	Step 2. System prompts for patient identification.

Reference “included use cases” in this section using INCLUDE *ius_name*	Step 3. Receptionist enters patient’s name or ID.	Step 4. System retrieves patient data.
	Step 5. The receptionist confirms appointment and details.	
	Step 6. The receptionist verifies the appointment time and confirms the patients identity.	Step 7. System updates appointment status to “Checked-in”.
	Step 8. If the patient is new, the receptionist triggers new patient registration.	Step 9. System saves the check-in data.
		Step 10. System confirms the check-in is complete.
	Actor Action	**System Action**
UNSUCCESSFUL	Step 1: The patient arrives without a scheduled	Step 2: The system fails to locate any appointments for the patient.

SCENARIOS (erroneous situations) ***EXTEND UC1EXT1 - Processof New Patient Check-in:***	appointment.	
	Step 3: The receptionist attempts to access appointment data.	Step 4: The system fails to respond due to connectivity issues.
	Step 5: The receptionist asks for identity verification.	
	Step 6: The patient is unable to provide valid identification.	Step 7: The system prevents check-in without identity confirmation.
Priority in scheduling	High	
Frequency	Frequently	
Business rules and data logic	• Identity must be confirmed via ID or system profile. • A check-in cannot proceed if the appointment is canceled or expired. • New patients must complete registration before check-in is finalized. • Each check-in must be timestamped and logged in audit records.	

Other non-functional requirements	• Check-in must be processed within 2 minutes. • System must display confirmation immediately after check-in.
Superordinates	None
Developer	Group 3 – Areej, Ram, Minahil
Creation date and last modified date	Created: 05/19/2025 Updated: 06/08/2025
Other Comments	• Ensure fallback workflows exist for offline check-in in case of system outage. • Enable exporting daily check-in logs for administrative reports.

USE CASE #	**UC2EXT3**
USE CASE Name	Process Reschedule Appointment
ACTOR	Receptionist
Goal (1 phrase)	To update the date and/or time of an existing appointment.
Overview and scope	This use case describes how the receptionist reschedules an existing appointment for a patient. The process involves identifying the current appointment, selecting a new available time slot, and updating the system records. This action helps accommodate patient requests or scheduling conflicts while ensuring provider availability is maintained.
Level	Extend
Preconditions	• The receptionist is logged into the system. • A valid appointment exists in the system. • The new desired time slot is available.

Postconditions in words (write in passive and past tense)	• **Appointment.Appointment_ID** was identified and retrieved based on patient search criteria. • **Appointment.PFName**, **Appointment.PLName**, and **Appointment.AppointmentDate&Time** were confirmed to validate the correct appointment. • **Appointment.Appointment_Reason**, **EmployeeID**, **StaffFname**, and **StaffLname** were reviewed for context before cancellation. • **Appointment status** was updated internally (though not explicitly shown in the class diagram, this status change is implied in the use case) to "Canceled" to reflect the action. • **Notification.Notification_ID** was generated and sent to **Notification.receiver_id**, which included either the **Patient.PatientID** or **Staff.EmployeeID**, depending on who needed to be notified. • **Notification.message** included the cancellation alert message and **Notification.timestamp** recorded the date and time of notification. • **Notification.status** was marked as "Sent" and **Notification.type** was labeled as "Cancellation".

<table>
<tr><td></td><td colspan="2">Staff.EmployeeID, Staff.StaffFname, Staff.StaffLname, and Staff.StaffPosition were logged to record who canceled the appointment.
Staff.Phone, Email, and Address were stored for audit and future reference purposes.
The canceled appointment was removed from the active appointment schedule in the system and logged for audit purposes.</td></tr>
<tr><td>Trigger</td><td colspan="2">• The receptionist selects the reschedule option from the appointment management screen.</td></tr>
<tr><td>Included Use Cases</td><td colspan="2">None</td></tr>
<tr><td>Extending Use Cases</td><td colspan="2">None</td></tr>
<tr><td rowspan="4">MAIN SUCCESSFUL SCENARIO <u>for this Use Case</u> in numbered sequence

Reference “included use cases” in this section using INCLUDE ius_name</td><td>Actor Action</td><td>System Action</td></tr>
<tr><td>Step 1. Receptionist selects “Reschedule Appointment”.</td><td>Step 2. System displays existing appointments.</td></tr>
<tr><td>Step 3. Receptionist selects the appointment to modify.</td><td>Step 4. System shows available future time slots.</td></tr>
<tr><td>Step 5. Receptionist selects a new date and time.</td><td></td></tr>
</table>

	Step 6. Receptionist confirms the new schedule.	Step 7. System updates the appointment record.
		Step 8. System sends updated appointment notifications.
		Step 9. System displays a confirmation message.
OTHER SUCCESSFUL SCENARIOS *Patient uses a split payment method (e.g., part cash, part card)*	**Actor Action**	**System Action**
	Step 1: Patient chooses to split payment	
	Step 2: Receptionist processes each portion using appropriate include cases.	Step 3: System confirms full amount covered by multiple payments.
		Step 4: Single receipt is generated showing all methods used.
OTHER USUCCESSFUL SCENARIOS (erroneous	**Actor Action**	**System Action**
	Step 1: Receptionist discusses payment options with the patient.	Step 2: System allows marking the invoice as unpaid or partially paid.

situations) ***Condition: Patient does not have the means to pay***	Step 3: Payment plan or billing follow-up is noted.	
OTHER USUCCESSFUL SCENARIOS (erroneous situations) ***Condition: System error during payment processing.***	Step 1: Receptionist initiates transaction.	Step 2: Payment module fails to respond or crashes.
		Step 3: Receptionist informs the patient and retries or escalates.
OTHER USUCCESSFUL SCENARIOS (erroneous	Step 1: Receptionist processes card/check.	Step 2: System flags invalid card or returned check.

situations) ***Condition: Invalid card or bounced check.***		
		Step 3: Receptionist requests alternate payment method from the patient.
Priority in scheduling	High	
Frequency	Occasionally	
Business rules and data logic	• Appointments can only be rescheduled to available future time slots. • Original appointment record should retain a log of the change. • Notifications must be sent immediately after rescheduling.	
Other non-functional requirements	• Interface should show live availability to avoid booking conflicts. • System must confirm changes within 3 seconds.	
Superordinates	None	

Developer	Group 3 – Areej, Ram, Minahil
Creation date and last modified date	Created: 05/19/2025 Updated: 06/08/2025
Other Comments	• Recommend allowing optional reason input for auditing why the reschedule occurred.

USE CASE #	**UC3**
USE CASE Name	Process of Cash Payment
ACTOR	Receptionist
Goal (1 phrase)	To collect and process a patients payment using cash
Overview and scope	This use case outlines how the receptionist processes payments from patients following a visit. The system supports multiple payment types including cash, credit card, and check. Here the receptionist selects the cash payment method, enters the amount, and confirms the transaction and the system updates the patient's balance. A receipt is generated, and the payment is logged in the system. It ensures accurate tracking of physical currency transactions at the front desk.
Level	Base
Preconditions	The receptionist is logged into the system. The patient has a payable amount in the system.

Postconditions in words (write in passive and past tense)	• **Payment.Payment_ID** was generated to uniquely identify the cash transaction. • **Payment.PatientID** was associated with the patient who made the payment. • **Payment.AmountPaid** was set to the amount of cash received from the patient. • **Payment.PaymentMethod** was assigned the value "Cash". • **Payment.Timestamp** was recorded with the exact date and time of the transaction. • **Payment.ReceivedBy (EmployeeID)** was logged as the receptionist who processed the payment. • **Payment.Status** was marked as "Completed" upon successful entry. • **Invoice.Invoice_ID** was updated to reflect the new payment status. • **Invoice.PatientID** remained linked to the payment and invoice.

	• **Invoice.TotalAmountDue** was updated, with any balance reduced by **Payment.AmountPaid**. • **Invoice.Status** was marked as "Paid" if full amount was received, or "Partially Paid" if only a portion was paid (if partial payments allowed). • **Receipt.Receipt_ID** was generated and linked to **Payment_ID**. • **Receipt.PatientID** and **Receipt.EmployeeID** were associated with the respective patient and receptionist. • **Receipt.AmountReceived**, **Receipt.ChangeGiven**, and **Receipt.PaymentMethod** were populated based on the transaction. • **Receipt.Timestamp** was logged at the moment of receipt generation. • **Receipt.DeliveryMethod** was recorded as "Printed" or "Emailed" based on patient preference.

<table>
<tr><td></td><td colspan="2">

- **CashRegisterLog.Transaction_ID** was created for audit purposes (implied from system logging requirement).
- **CashRegisterLog.EmployeeID**, **AmountIn**, and **ChangeOut** were recorded to ensure audit and compliance.
- **Notification (optional)** may have been generated if system settings triggered confirmation via SMS/email.

</td></tr>
<tr><td>Trigger</td><td colspan="2">The receptionist selects the payment option at the end of a patient visit.</td></tr>
<tr><td>Included Use Cases</td><td colspan="2">None</td></tr>
<tr><td>Extending Use Cases</td><td colspan="2">INCLUDE INC1 – Process Credit Card Payment
INCLUDE INC2 – Process Check Payment</td></tr>
<tr><td rowspan="3">MAIN SUCCESSFUL SCENARIO for this Use Case in numbered sequence</td><td>Actor Action</td><td>System Action</td></tr>
<tr><td>Step 1. Receptionist selects “Process Cash Payment”.</td><td>Step 2. System displays patient billing details.</td></tr>
<tr><td>Step 3. Receptionist confirms total amount due.</td><td>Step 4. System loads the cash payment screen.</td></tr>
</table>

Reference “included use cases” in this section using INCLUDE *ius_name*	Step 5. Receptionist enters the amount received from the patient.	Step 6: System calculates and displays change due
	Step7: Receptionist provides correct change to the patient.	Step 8. System records the transaction as a cash payment.
		Step 9. System updates the patient’s balance and marks the invoice as paid.
		Step 10. System generates and prints or sends the receipt.
OTHER SUCCESSFUL SCENARIOS (Specify any successful variations of thenormal execution path, including any extension points Using EXTEND eus_name)	**Actor Action**	**System Action**
	Step 1: Receptionist records amount received greater than total due.	Step 2: System calculates and displays change due.
	Step 3: Receptionist provides correct change.	Step 4: System finalizes the transaction and issues a receipt.

Extend UCEXT 2: ***Overpayment Handling:***		
UNSUCCESSFUL ***SCENARIOS*** ***(erroneoussituations)*** ***Insufficient cash received***	Step 1: Receptionist counts money and identifies a shortfall.	Step 2: System prevents payment from being recorded.
	Step 3: Receptionist informs patient and requests full amount.	
UNSUCCESSFUL ***SCENARIOS*** ***(erroneoussituations)*** ***System error during transaction entry***	Step 1: Receptionist attempts to log payment..	Step 2: Payment module fails to record transaction.
	Step 3: Receptionist retries or escalates to IT support.	
Priority in scheduling	High	
Frequency	Frequently	

Business rules and data logic	• Payments must match or exceed the total due unless a partial payment is allowed. • Each payment must be associated with a payment method and timestamp. • A receipt must be generated for every successful payment. • For cash: • Change due must be calculated and issued immediately. • Cash receipts must show received amount, totals, and change. • Cashier ID must be logged with the transaction.
Other non-functional requirements	• Must support encrypted, secure payment data transmission. • System must log all payment attempts and outcomes. • Support for printing and/or emailing receipts. • Receipt must be issued within 1 minute of cash entry.
Superordinates	None
Developer	Group 3 – Areej, Ram, Minahil

Creation date and last modified date	Created: 05/19/2025 Updated: 06/08/2025
Other Comments	Consider adding support for installment plans or insurance claims integration in future releases.

USE CASE #	**INC 1**
USE CASE Name	Process Credit Card Payment
ACTOR	Receptionist
Goal (1 phrase)	To process a patient's payment using a credit card.
Overview and scope	This use case describes how the receptionist processes a payment using a credit card. The receptionist selects the credit card option, enters the card details, and submits the transaction through the system's integrated payment gateway. Once the transaction is verified and approved, the system updates the billing record and generates a receipt.
Level	<<include>>

Preconditions	• The receptionist is logged into the system. • The patient has a pending invoice and chooses to pay via credit card.

Postconditions in words (write in passive and past tense)	• **Payment.Payment_ID** was generated to uniquely identify the credit card transaction. • **Payment.PatientID** was linked to the patient making the payment. • **Payment.AmountPaid** was set to the amount paid using the credit card. • **Payment.PaymentMethod** was assigned the value "Credit Card". • **Payment.Timestamp** was logged at the time of successful transaction. • **Payment.ReceivedBy (EmployeeID)** was recorded as the receptionist processing the payment. • **Payment.Status** was set to "Completed" after verification by the payment gateway. • **Invoice.Invoice_ID** was updated to reflect the payment. • **Invoice.PatientID** was maintained to associate the payment with the correct invoice. • **Invoice.TotalAmountDue** was reduced by **Payment.AmountPaid**. • **Invoice.Status** was updated to "Paid" or "Partially Paid" depending on whether the full balance was covered.

	• **Receipt.Receipt_ID** was generated and linked to the **Payment_ID**. • **Receipt.PatientID** and **Receipt.EmployeeID** were associated with the appropriate entities. • **Receipt.AmountReceived**, **Receipt.PaymentMethod**, and **Receipt.Timestamp** were recorded accurately. • **Receipt.DeliveryMethod** was marked as "Printed" or "Emailed" depending on patient preference. • **CreditCardTransaction.Transaction_ID** (if defined separately) was logged for gateway tracking. • **CreditCardTransaction.AuthorizationCode** was stored to reflect payment gateway approval. • **CreditCardTransaction.Status** was marked "Approved" upon gateway confirmation. • **AuditLog.Entry** was created to reflect that a credit card payment occurred, associated with **EmployeeID**, **PatientID**, and **Timestamp**. • **Sensitive card data** (e.g., card number, CVV) was not stored, as per PCI-DSS compliance. • **Transaction time** was within the 30-second response window (non-functional compliance).
Trigger	The receptionist selects “Credit Card” as the payment method

	during the payment process.	
Included Use Cases	None	
Extending Use Cases	None	
	Actor Action	**System Action**
MAIN SUCCESSFUL SCENARIO for this Use Case **in numbered sequence** Reference “included use cases” in this section using INCLUDE *ius_name*	Step 1. Receptionist selects "credit card" as the payment method in the system	
	Step 2. Receptionist inserts the card into the reader	Step 3. The terminal reads the card and communicates with the payment gateway for verification.
		Step 4. System receives approval response.
		Step 5. System records the payment and updates the invoice.
		Step 6. System generates and prints/sends the receipt.

	Actor Action	System Action
SUCCESSFUL SCENARIOS (erroneous situations)		Step 1: Terminal prompts for PIN or signature.
Card requires PIN entry or signature	Step 2: Patient enters pin or signs.	Step 3: Transaction proceeds after successful input.
	Step 4: System marks invoice as paid and issues receipt.	Step 5: System finalizes the transaction and issues a receipt.
UNSUCCESSFUL SCENARIOS (erroneous situations)	**Action Action**	**System Action**
		Step 1: Terminal displays "Card Declined."
Card is Declined	Step 2: Receptionist informs the patient and requests an alternate payment method.	Step 3: Receptionist informs patient and requests full amount.
	Action Action	**System Action**
UNSUCCESSFUL		STEP 1: Payment terminal fails to connect to payment gateway.

SCENARIOS (erroneous situations) ***Network Error***	STEP 2: Receptionist retries transaction or switches to another terminal.	
UNSUCCESSFUL SCENARIOS (erroneous situations) ***Patient Removes Card too early***	**Action Action**	**System Action**
		STEP 1: Transaction is interrupted.
	STEP 2: Receptionist re- initiates payment process and advises patient.	
Priority in scheduling	High	
Frequency	Frequently	
Business rules and data logic	The credit card must be valid and active. • System must mask sensitive card data after entry.	

	• Payment status must be updated only on successful authorization.
Other non-functional requirements	Transaction should complete within 30 seconds. All transmissions must be encrypted The system must store payment logs securely.
Superordinates	UC3
Developer	Group 3 – Areej, Ram, Minahil
Creation date and last modified date	Created: 05/19/2025 Updated: 06/08/2025
Other Comments	• The system must comply with PCI-DSS standards for card handling.

USE CASE #	**INC 2**
USE CASE Name	Process Check Payment
ACTOR	Receptionist
Goal (1 phrase)	To process a patient's payment using a personal check.
Overview and scope	This use case describes how the receptionist records a check payment from a patient. The receptionist collects the physical check, enters the required details such as check number, bank name, account holder, and check date, then confirms the transaction. The system stores the information, updates the invoice balance, and generates a receipt.
Level	<<include>>
Preconditions	• The receptionist is logged into the system. • The patient has a pending invoice and chooses to pay via check.

Postconditions in words (write in passive and past tense)	• **Payment.Payment_ID** was generated to uniquely identify the check transaction. • **Payment.PatientID** was associated with the patient making the payment. • **Payment.AmountPaid** was set to the total amount written on the check. • **Payment.PaymentMethod** was assigned the value "Check". • **Payment.Timestamp** was logged at the time the check payment was entered. • **Payment.ReceivedBy (EmployeeID)** was recorded as the receptionist who processed the check. • **Payment.Status** was set to "Pending Clearance" after entry. • **Check.CheckNumber** was stored and verified to be unique per patient. • **Check.BankName**, **Check.AccountHolder**, and **Check.CheckDate** were entered and stored as part of the check record. • **Check.Status** was set to "Pending" (to be updated later to "Cleared" or "Returned" after bank processing).

	• **Invoice.Invoice_ID** was updated to reflect the pending payment. • **Invoice.TotalAmountDue** was reduced by **Payment.AmountPaid**. • **Invoice.Status** was updated to "Pending" (until check is cleared). • **Receipt.Receipt_ID** was generated and linked to the **Payment_ID**. • **Receipt.PatientID** and **Receipt.EmployeeID** were stored. • **Receipt.AmountReceived**, **Receipt.PaymentMethod**, and **Receipt.Timestamp** were recorded. • **Receipt.ClearanceNote** indicated the payment is pending bank confirmation. • **Receipt.DeliveryMethod** was marked as "Printed" or "Emailed" based on patient preference. • **AuditLog.Entry** was created to log the check transaction attempt with **EmployeeID**, **PatientID**, **CheckNumber**, and **Timestamp**. • **System.Logs** securely stored the check details as per privacy policy.
Trigger	The receptionist selects “Check” as the payment method during

	the payment process.	
Included Use Cases	None	
Extending Use Cases	None	
	Actor Action	**System Action**
MAIN SUCCESSFUL SCENARIO <u>for this Use Case</u> **in numbered sequence** Reference “included use cases” in this section using INCLUDE *ius_name*	Step 1. Receptionist selects "check" as the payment method in the system.	
	Step 2. Receptionist enters check details: check number, bank name, account holder name, and check date.	
	Step 3. Receptionist confirms the transaction.	Step 4. System records the check payment in the transaction log.
		Step 5. System updates the invoice as pending for confirmation.

		Step 6. System generates and prints/sends the receipt.
		Step 7. System sends the check for bank verification.
	Actor Action	**System Action**
SUCCESSFUL SCENARIOS (erroneous situations) ***Patient pays for multiple invoices with a single check***	Step 1: Patient indicates that the check covers multiple visits.	
	Step 2: Receptionist selects multiple invoices and enters combined check amount.	Step 3: System splits and applies payment accordingly.
		Step 4: System marks all selected invoices as pending clearance.
		Step 5: A consolidated receipt is generated and printed.
UNSUCCESSFUL	**Action Action**	**System Action**

SCENARIOS (erroneous situations) ***Check has missing or incorrect details***	Step 1: The Receptionist identifies the issue (e.g., unsigned check).	Step 2: System does not accept the entry until issue is corrected.
UNSUCCESSFUL SCENARIOS (erroneous situations) ***System Entry Error***	**Actor Action**	**System Action**
	Step 1: Receptionist inputs incorrect check number or amount.	Step 2: System prompts a mismatch error or prevents Submission.
	Step 3: Receptionist re-enters the correct data.	
UNSUCCESSFUL SCENARIOS (erroneous situations) ***Paatient Check is post Dated***	**Actor Action**	**System Action**
	Step 1: Receptionist identifies the future date.	Step 2: System prevents payment recording until the date is valid.
Priority in scheduling	Medium	
Frequency	Occassionally	
Business rules and data logic	All check information (check number, bank name,	

	account holder, and check date) must be entered before submission. • The same check number cannot be used more than once per patient. • Check payments remain pending until cleared by the bank.
Other non-functional requirements	• Check info must be stored securely and associated with patient records. • Receipt must indicate payment is pending clearance.
Superordinates	UC3
Developer	Group 3 – Areej, Ram, Minahil
Creation date and last modified date	Created: 05/19/2025 Updated: 06/08/2025
Other Comments	• Recommend tracking check status (cleared, pending, returned) in a future enhancement.

USE CASE #	**UC4**
USE CASE Name	Access Visiting Summary
ACTOR	Patient
Goal (1 phrase)	To access and review the summary of previous healthcare visits.
Overview and scope	This use case describes the system allows patients to view detailed summaries of their past visits, including doctor notes, prescribed treatments, and follow-up recommendations. While reviewing, patients can choose to fill out required forms, reschedule future appointments, cancel upcoming ones, or proceed with invoice payments related to their visits.
Level	Base
Preconditions	• Patient is authenticated and logged into the system. • Previous visit data is available in the system.

Postconditions in words (write in passive and past tense)	• Visit summaries were retrieved and displayed for the selected PatientID, including associated Visit.Purpose, Diagnosis_Summary, and Visit.Status. • Associated MedicalRecord.Test_Results, Diagnosis, Treatment, and Last_Visit_Date were accessed and shown. • Prescription.medication_name, dosage, duration, and quantity were displayed for visits that included issued prescriptions. • Doctor details including StaffLname, and StaffPosition were shown as part of the visit summary context. • If required, linked Forms were filled and submitted by the patient. • Future appointments were optionally rescheduled by updating Appointment.Appointment_DateTime and Appointment_Reason. • Appointments selected for cancellation had their Appointment.Status updated to “Cancelled”. • Associated visit invoices were retrieved and displayed, and related payments were completed, updating Invoice.Status and recording Payment.Transaction_ID and total_Amount.
Trigger	• Patient selects “Access Visiting Summary” from the system interface.

<table>
<tr><td>Included Use Cases</td><td colspan="2">None</td></tr>
<tr><td>Extending Use Cases</td><td colspan="2">• Fill Out Forms (UC4EXT4)
• Reschedule Appointment (UC4EXT5)
• Cancel Appointment (UC4EXT6)
• Pay Invoice (UC4EXT7)</td></tr>
<tr><td rowspan="7">MAIN SUCCESSFUL SCENARIO <u>for this Use Case</u> in numbered sequence

Reference “included use cases” in this section using INCLUDE ius_name</td><td>Actor Action</td><td>System Action</td></tr>
<tr><td>Step 1. Patient selects “Access Visiting Summary”.</td><td>Step 2. System retrieves and displays the visiting summary.</td></tr>
<tr><td>Step 3. Patient reviews the summary.</td><td></td></tr>
<tr><td>Step 4. Patient chooses to fill out forms (if needed).</td><td>Step5. EXTEND Fill Out Forms (UC4EXT4) is triggered.</td></tr>
<tr><td>Step 6. Patient chooses to reschedule an appointment (if needed).</td><td>Step 7. EXTEND Reschedule Appointment (UC4EXT5) is triggered.</td></tr>
<tr><td>Step 8. Patient chooses to cancel an appointment (if needed).</td><td>Step 9. EXTEND Cancel Appointment (UC4EXT6) is triggered.</td></tr>
<tr><td>Step 10. Patient chooses to pay an invoice (if needed).</td><td>Step 11. EXTEND Pay Invoice (UC4EXT7) is triggered.</td></tr>
</table>

Priority in scheduling	High
Frequency	Frequently
Business rules and data logic	• Patients can only access summaries of visits they have attended. • Sensitive medical data must be displayed in compliance with HIPAA regulations. • Optional actions (forms, rescheduling, canceling, payment) are available only if applicable to the selected visit. • Payments must be processed through secure, PCI-compliant gateways. • Patients cannot reschedule or cancel past appointments—only future ones.
Other non-functional requirements	• Secure authentication and session management. • System must provide responses within 3 seconds for summary retrieval. • Daily backups of patient data. • Audit logging for all user interactions related to medical records.
Superordinates	Patient Management System
Developer	Group 3: Areej, Minahil, Ram
Creation date and last modified date	Creation 05/10/2025 Updated 05/18/2025

Other Comments	• Ensure encryption of all personal and medical data in transit and at rest. • Consider adding multi-factor authentication for enhanced security.

USE CASE #	**UC4EXT4**
USE CASE Name	Fill Out Forms
ACTOR	Patient
Goal (1 phrase)	To complete and submit required forms related to a healthcare visit.
Overview and scope	The system allows patients to access, complete, and submit various forms associated with their visits. Forms may include medical history, feedback surveys, consent forms, or insurance information. This process ensures that all necessary documentation is completed before or after a visit.
Level	Extend
Preconditions	• A visiting summary or related appointment exists that requires form completion.

Postconditions in words (write in passive and past tense)	• Required forms were accessed, filled out, and submitted successfully by Patient.PatientID. • MedicalRecord.Test_Results, Diagnosis, Treatment, and Last_Visit_Date were updated when medical history or clinical input forms were submitted. • Consent responses and feedback were recorded and linked to the corresponding Visit.Visit_ID where applicable. • Insurance information such as Claim.Insurance_Company, Claim.Status, and Claim.Submitted_Date was updated if insurance-related forms were completed. • Submission timestamps and any associated metadata were stored to maintain audit trails. • System validation confirmed all required fields were completed before accepting each form. • Any missing or optional documentation was flagged for follow-up, and notifications were triggered via the Notification class when necessary.	
Trigger	• Patient selects the option to fill out forms while viewing the visiting summary.	
Included Use Cases	None	
Extending Use Cases	None	
	Actor Action	**System Action**

***MAIN SUCCESSFUL SCENARIO <u>for this Use Case</u>* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	Step 1. Patient logs into the system.	Step 2. System presents the main dashboard.
	Step 3. Patient selects "Fill Out Forms" from the visiting summary.	Step 4. System displays available forms.
	Step 5. Patient selects a specific form to fill.	Step 6. System presents the selected form.
	Step 7. Patient completes the form and submits it.	Step 8. System validates the form data and saves it.
		Step 9. System displays confirmation that the form was successfully submitted.
Priority in scheduling	Medium	
Frequency	Occasionally	
Business rules and data logic	• All mandatory fields must be completed before submission. • Data entered must pass format validations (e.g., correct date formats, valid contact details).	

	• Once submitted, forms cannot be edited—patients must contact support for corrections.
Other non-functional requirements	• Secure form data storage and transmission. • User-friendly interface with form auto-save feature.
Superordinates	Access Visiting Summary
Developer	Group 3: Areej, Minahil, Ram
Creation date and last modified date	Creation 05/10/2025 Updated 05/18/2025
Other Comments	

USE CASE #	**UC4EXT5**
USE CASE Name	Reschedule Appointment
ACTOR	Patient
Goal (1 phrase)	To change the date and/or time of an upcoming appointment.
Overview and scope	The system allows patients to reschedule their future appointments through the patient portal. Patients can select new available time slots based on their preferences and the availability of healthcare providers. This ensures efficient appointment management without requiring direct phone calls or in-person visits.
Level	Extend
Preconditions	• There is at least one upcoming appointment eligible for rescheduling.

Postconditions in words (write in passive and past tense)	• Appointment_DateTime was updated successfully to reflect the newly selected time slot. • Appointment_Reason was optionally modified if additional details were provided during the rescheduling process. • The Appointment.Status remained marked as "Scheduled", ensuring the continuity of the upcoming visit. • Notification.Message, Type, Timestamp, and Receiver_ID were generated and recorded to inform both the Patient.PatientID and the assigned Staff.EmployeeID about the schedule change. • System logs were updated to capture the modification timestamp and the actor responsible for the change for future auditing. • All updates were validated against staff availability and appointment conflicts before confirmation was completed.	
Trigger	Patient selects the option to reschedule an appointment from the visiting summary or appointment management interface.	
Included Use Cases	None	
Extending Use Cases	None	
MAIN SUCCESSFUL SCENARIO for this	**Actor Action**	**System Action**
	Step 1. Patient logs into the system.	Step 2. System presents the main dashboard.

***Use Case* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	Step 3. Patient selects "Reschedule Appointment."	Step 4. System displays upcoming appointments eligible for rescheduling.
	Step 5. Patient selects the appointment to reschedule.	Step 6. System displays available time slots.
	Step 7. Patient selects a new date and time.	
	Step 8. Patient confirms the new appointment time.	Step 9. System updates the appointment and sends confirmation notifications.
		Step 10. System displays a success message to the patient.
UNSUCCESSFUL SCENARIOS (*erroneous* situations*)*	**Conditions**	**Actions**
Priority in scheduling	High	

Frequency	Frequently
Business rules and data logic	• Rescheduling is only allowed for future appointments. • Patients cannot reschedule within 24 hours of the appointment time. • Available time slots are dynamically updated based on real-time provider availability. • Notifications must be sent immediately after a successful rescheduling.
Other non-functional requirements	• The system must provide real-time availability of appointment slots. • Confirmation notifications should be sent via both email and SMS if enabled. • System response time must be under 3 seconds for displaying availability.
Superordinates	Access Visiting Summary
Developer	Group 3: Areej, Minahil, Ram

Creation date and last modified date	Creation 05/10/2025 Updated 05/18/2025
Other Comments	• Consider adding a waitlist feature for fully booked schedules.

USE CASE #	**UC4EXT6**
USE CASE Name	Cancel Appointment
ACTOR	Patient
Goal (1 phrase)	To cancel a scheduled upcoming appointment.
Overview and scope	The system allows patients to cancel their upcoming appointments directly through the portal. This helps free up provider availability and ensures accurate scheduling. Upon cancellation, the system updates the schedule and sends notifications to both the patient and the healthcare provider.
Level	Extend
Preconditions	• There is at least one upcoming appointment eligible for cancellation.

Postconditions in words (write in passive and past tense)	• Appointment.Status was updated to "Cancelled" for the selected Appointment_ID. • The corresponding Appointment_DateTime and associated slot were released and marked as available for reassignment to other patients. • Notification.Message, Type, Timestamp, and Receiver_ID were generated and logged to inform both the PatientID and the Staff.EmployeeID of the cancellation. • The cancellation action was recorded in system logs with a timestamp for auditing and tracking purposes. • Any dependencies, such as linked visits or forms, were flagged for review to ensure system consistency after cancellation.	
Trigger	Patient selects the option to cancel an appointment from the visiting summary or appointment management interface.	
Included Use Cases	None	
Extending Use Cases	None	
***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**	**Actor Action**	**System Action**
	Step 1. Patient logs into the system.	Step 2. System presents the main dashboard.
	Step 3. Patient selects "Cancel Appointment."	Step 4. System displays a list of upcoming appointments.

Reference “included use cases” in this section using INCLUDE *ius_name*	Step 5. Patient selects the appointment to cancel.	Step 6. System asks for confirmation of the cancellation.
	Step 7. Patient confirms cancellation.	Step 8. System cancels the appointment and updates the schedule.
		Step 9. System sends notifications to both the patient and healthcare provider.
		Step 10. System displays a confirmation message to the patient.
UNSUCCESSFUL SCENARIOS (*erroneous* situations*)*	**Conditions**	**Actions**
Priority in scheduling	High	
Frequency	Occasionally	
Business rules and data logic	• Cancellations are only allowed for future appointments. • Cancellations within 24 hours of the appointment time are not permitted unless explicitly allowed by business rules. • Notifications must be sent to the patient and the healthcare provider immediately after cancellation.	

	• Cancelled appointment slots should be immediately marked as available for booking by other patients.
Other non-functional requirements	• Confirmation of cancellation must be provided within 2 seconds. • Notification messages must be sent via both email and SMS (if enabled). • System should maintain an audit log of all cancellation activities.
Superordinates	Access Visiting Summary
Developer	Group 3: Areej, Minahil, Ram
Creation date and last modified date	Creation 05/10/2025 Updated 05/18/2025
Other Comments	• Consider adding a prompt to gather the reason for cancellation to analyze patient behavior.

USE CASE #	**UC4EXT7**
USE CASE Name	Pay Invoice
ACTOR	Patient
Goal (1 phrase)	To pay outstanding invoices related to healthcare visits.
Overview and scope	The system allows patients to view and pay their pending invoices securely through the patient portal. Payments can be made using various payment methods, including credit cards, checks and cash. Upon successful payment, the system updates the financial records and sends confirmation notifications.
Level	Extend
Preconditions	• There is at least one unpaid invoice associated with the patient's account.

Postconditions in words (write in passive and past tense)	• Invoice.Status was updated to "Paid" for the selected Invoice_ID. • A new payment record was created, capturing Transaction_ID, Transaction_Date, Payment_Type (credit card, check, or cash), and Total_Amount. • The Payment.Invoice_ID foreign key linked the transaction directly to the relevant invoice for accurate financial tracking. • Notification.Message, Type, Timestamp, and Receiver_ID were generated and sent to the corresponding PatientID as confirmation of successful payment. • System logs were updated to include the payment entry and associated metadata for auditing purposes. • Any pending balances were recalculated, and the patient's financial history was updated in real time.	
Trigger	Patient selects the "Pay Invoice" option from the visiting summary or billing section.	
Included Use Cases	INC3, INC2, INC1	
Extending Use Cases	None	
***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**	**Actor Action**	**System Action**
	Step 1. Patient logs into the system.	Step 2. System presents the main dashboard.
	Step 3. Patient selects "Pay Invoice."	Step 4. System displays a list of unpaid invoices.

Reference “included use cases” in this section using INCLUDE *ius_name*	Step 5. Patient selects an invoice to pay.	**Step 6.** System displays payment options and prompts for payment details.
	Step 7. Patient enters payment details and confirms payment.	**Step 8.** System processes the payment through the payment gateway.
		Step 9. System confirms successful payment and updates financial records.
		Step 10. System sends payment confirmation notifications via email/SMS.
		Step 11. System displays a success message to the patient.
Priority in scheduling	High	
Frequency	Frequently	
Business rules and data logic	• Payments must be processed using secure methods. • Partial payments are not allowed. • Receipts must be generated and accessible to patients after payment. • System must prevent duplicate payments for the same invoice.	
Other non-functional requirements	• Payment confirmation must be provided within 5 seconds. • Support for multiple payment methods (Credit Card, Checks, Cash).	

	• All transactions must be encrypted and logged for auditing purposes.
Superordinates	Access Visiting Summary
Developer	Group 3: Areej, Minahil, Ram
Creation date and last modified date	Creation 05/10/2025 Updated 05/18/2025
Other Comments	• Implement reminder notifications for unpaid invoices.

USE CASE #	**UC5**
USE CASE Name	Record Vital
ACTOR	Nurse
Goal (1 phrase)	To record and update the patient's signs during or before a visit
Overview and scope	This use case allows healthcare staff to input and update vital signs such as temperature, heart rate, respiratory rate, blood pressure and oxygen saturation for a patient. The data can be recorded during triage, during an ongoing visit, or immediately after
Level	Base
Preconditions	• The patient is checked in for a visit. • The nurse is authenticated and has permission to record vitals.

Postconditions in words(write in passive and past tense)	• Vital signs were recorded for Visit.Visit_ID and linked to Visit.Appointment_ID. • The Visit.Status was updated to "Vitals Recorded", and Visit.Purpose was noted as part of the clinical workflow. • The Visit.Visit_Date captured the date and time of the entry. • The Nurse's Staff.EmployeeID, StaffFname, StaffLname, StaffPosition, and Gender were logged to indicate who recorded the vitals. • Contact details including Staff.Phone, Staff.Email, and Staff.Address were stored for audit and communication purposes.	
Trigger	• Nurse selects "Record Vital Signs" from the visit dashboard	
Included Use Cases	RecordHistory (INC4)	
Extending Use Cases	MedicationAdministration (UC5EXT8) PerformDiagnosticTest (UC5EXT8)	
	Actor Action	**System Action**

		Step 1. System presents the dashboard
	Step 2. Nurse selects the patient visit	Step 3..System loads visit details
***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**	Step 4. Nurse clicks on "Record Vital Signs"	**Step 5:System displays the vital entry form**
	Step 6. Nurse enters Vitals	Step 7:System validates and saves the data
Reference "included use cases" in this section using INCLUDE *ius_name*	Step 8: EXTEND MedicationAdministration (if medication is immediately administered).	
	Step 9: EXTEND PerformDiagnosticTest (if test is ordered during vitals recording).	
	Step 11:System confirms successful save.	
Priority in scheduling	High	
Frequency	Every patient visit	

Business rules and data logic	• Vitals must be recorded in medically accepted units and ranges. • Each entry must be associated with a valid Visit_ID and timestamp. • Only authorized clinical staff can modify or submit vital records.
Other non-functional requirements	• System must respond within 2 seconds. • Audit logs must capture who recorded each entry. • System must comply with medical regulations.
Superordinates	Patient Visit Management
Developer	Group 3
Creation date and last modified date	Created: 05/18/2025 Updated: 05/19/2025
Other Comments	Consider integrating with medical devices for automatic vital capture in the future.

USE CASE #	**UC5 EXT 8**
USE CASE Name	Medication Adminstration
ACTOR	Nurse
Goal (1 phrase)	To administer medication to a patient during a visit and update the system with administration details.
Overview and scope	**This use case allows a nurse or authorized staff to administer prescribed medication during a visit and document the details in the patient's record. It ensures that the right medication, dosage, and route are recorded and linked to the corresponding visit.**
Level	Extend
Preconditions	• The patient is checked in for a visit. • Medication orders exist in the system. • The nurse is authenticated and authorized.

Postconditions in words(write in passive and past tense)	• A new entry was created in Prescription.prescription_id and associated with Prescription.visit_id. • Prescription.medication_name, dosage, quantity, and duration were saved according to the administration record. • Prescription.issued_date captured the exact time the medication was administered. • The administering nurse was logged using Staff.EmployeeID, along with StaffFname, StaffLname, StaffPosition, and Gender for clinical traceability. • Contact details including Staff.Phone, Staff.Email, and Staff.Address were recorded to support follow-up and accountability.	
Trigger	• Nurse selects “Administer Medication” from the patient’s visit panel.	
Included Use Cases	None	
Extending Use Cases	Record Vital (UC5)	
	Actor Action	**System Action**

		Step 1. System presents the main dashboard
***MAIN SUCCESSFUL SCENARIO <u>for this Use Case</u>* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	Step 2. Nurse accesses the patient's active visit.	Step 3.System displays the visit information
	Step 4. Nurse selects "Administer Medication."	**Step 5. System displays list of prescribed medications.**
	Step 6. Nurse selects a medication and enters administration details (e.g., dosage, route, time).	Step 7. System verifies and records the administration.
	Step 8. System updates the patient's medication history.	Step 9. System confirms the medication was recorded successfully.
	Step 11:System confirms successful save.	

Priority in scheduling	High
Frequency	Frequently during patient visits involving prescriptions
Business rules and data logic	• Only prescribed medications can be administered. • Each entry must include route, dosage, and time.

	• Entries are immutable; corrections require audit trails.
Other non-functional requirements	• Allow barcode validation of medication. • Save within 2 seconds. • All actions must comply with HIPAA standards.
Superordinates	Record Vital (UC5)
Developer	Group 3
Creation date and last modified date	Created: 05/18/2025 Updated: 05/19/2025
Other Comments	Future implementation can include decision support alerts for drug interactions.

USE CASE #	**INC 4**
USE CASE Name	Record History
ACTOR	Nurse/Doctor
Goal (1 phrase)	**To record and update the patient's medical history as part of the visit documentation.**
Overview and scope	**This use case allows authorized clinical staff to enter or update the patient's medical history, including chronic conditions, past illnesses, allergies, previous surgeries, family history, and lifestyle details. This information supports clinical decision-making and is used throughout the patient's care lifecycle.**
Level	Include

Preconditions	• The patient has an active visit. • The user (nurse/doctor) is logged in and authorized to access medical records.

Postconditions in words(write in passive and past tense)	• Patient history was updated in MedicalRecord.Diagnosis and MedicalRecord.Treatment. • The entry was linked to the patient using MedicalRecord.MedicalRecordID. • Patient.FName and LName were referenced for clear identification during history update. • Patient.EmergencyContact was reviewed and updated if necessary for safety documentation. • MedicalRecord.PFName, PLName, and DOB were recorded to ensure correct patient identification and age-appropriate care. • MedicalRecord.Last_Visit_Date was updated to reflect the time of modification. • The staff member who updated the record was logged via Staff.EmployeeID, along with StaffFname, StaffLname, StaffPosition, and Gender. • Staff.Phone, Email, and Address were stored to maintain communication and audit logs.
Trigger	• The user selects “Record History” while updating the visit documentation.

Included Use Cases	None	
Extending Use Cases	• Record Vital (UC5) • Documenting Visit (UC6)	
***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	**Actor Action**	**System Action**
		Step 1. System displays the dashboard.
	Step 2. Staff selects the patient and accesses visit details	Step 3. System loads the current visit and medical history.
	Step 4. Staff selects "Record History."	**Step 5: System displays the history entry form.**
	Step 6. Staff enters or updates relevant medical history.	Step 7. System validates and saves the data.
	Step 8. System logs the entry with timestamp and staff ID.	Step 9. System confirms the update.
Priority in scheduling	Medium	

Frequency	Occasionally – typically at the first visit or when significant history changes occur
Business rules and data logic	• Only licensed healthcare providers may update medical history. • Sensitive history data must comply with HIPAA privacy standards. • Each entry must be traceable by staff ID and timestamp.
Other non-functional requirements	• Changes to history must be saved within 2 seconds. • All updates must be versioned and audit-logged. • The system must support importing history from previous visits.
Superordinates	Record Vital (UC5), Documenting Visit (UC6)
Developer	Group 3
Creation date and last modified date	Created: 05/19/2025 Updated: 05/19/2025
Other Comments	Option to import history from prior visits or external providers may be considered in future versions.

USE CASE #	**UC6**
USE CASE Name	Documenting Visit
ACTOR	Doctor
Goal (1 phrase)	**To document the observations, diagnosis, treatments, and notes for a patient's visit.**
Overview and scope	**This use case enables healthcare providers to comprehensively document a patient's visit. This includes symptoms, physical exam findings, diagnosis, treatments provided, and medical advice. The documentation ensures continuity of care and legal compliance and supports future decision-making.**
Level	Base
Preconditions	• Patient is actively checked in for a visit. • Doctor is logged in and authorized.

Postconditions in words(write in passive and past tense)	• Visit details were saved in Visit.Diagnosis_Summary and associated with Visit.Visit_ID and Visit.Appointment_ID. • Visit.Status was updated to "Completed", and Visit.Purpose was documented to reflect the consultation reason. • Visit.Visit_Date recorded the timestamp of the encounter. • MedicalRecord.Diagnosis and Treatment were updated, and MedicalRecord.Last_Visit_Date was refreshed. • Patient context was preserved using MedicalRecord.PFName, PLName, and DOB. • Doctor was logged using Staff.EmployeeID, along with StaffFname, StaffLname, StaffPosition, and Gender. • Doctor's LicenseNumber and Specialty were recorded to verify provider credentials. • Staff.Phone, Email, and Address were saved for documentation and communication traceability.
Trigger	• Doctor selects "Document Visit" during or after the patient consultation.
Included Use Cases	RecordHistory (INC4)
Extending Use Cases	• Order Test (UC6EXT9) Diagnosis (UC6EXT10)

	Write Referral (UC6EXT11) Write Prescriptions (UC6EXT12)	
***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**	**Actor Action**	**System Action**
		Step 1. System displays the dashboard.
		Step 2. System displays visit details and documentation panel.
Reference "included use cases" in this section using INCLUDE *ius_name*	Step 3. Doctor selects "Record History."	**Step 6:** System auto-saves entries at intervals.
	Step 7. INCLUDE RecordHistory (INC4) is triggered if history needs update.	
	Step 8. EXTEND Order Test (UC6EXT9) is triggered if tests are ordered. Step 9. EXTEND Diagnosis (UC6EXT10) is triggered if diagnosis is entered.	

	Step 10. EXTEND Write Referral (UC6EXT11) is triggered if referral is made Step 11. EXTEND Write Prescriptions (UC6EXT12) is triggered if medication is prescribed. Step 12. Doctor submits and finalizes visit documentation.	Step 13. System confirms save and updates visit status.
Priority in scheduling	High	
Frequency	Every completed patient visit	

Business rules and data logic	• Only authorized providers may complete visit documentation. • Diagnoses must use valid ICD-10 codes. • Finalized notes are version-controlled.
Other non-functional requirements	• Autosave every 60 seconds. • All documentation encrypted in storage and transit. • Full edit history should be available.
Superordinates	Patient Visit Management
Developer	Group 3
Creation date and last modified date	Created: 05/19/2025 Updated: 05/19/2025
Other Comments	Adding templates for common visit types (e.g., annual physical, follow-up) in future versions.
USE CASE #	**UC6EXT9 – Order Test,**
USE CASE Name	Order Test
ACTOR	Doctor
Goal (1 phrase)	To order laboratory or imaging tests for the patient during a visit.

Overview and scope	This use case allows healthcare providers to order diagnostic tests such as blood work, urine analysis, X-rays, and other lab or imaging procedures as part of the clinical assessment. Orders are logged in the system and routed to the appropriate department or external lab.
Level	Extend
Preconditions	• Patient visit is active. • The Doctor is logged . • Test catalog is available in the system.
Postconditions in words (write in passive and past tense)	• The ordered test was documented in MedicalRecord.Test_Results and linked to the corresponding Visit.Visit_ID and MedicalRecord.MedicalRecordID. • MedicalRecord.Last_Visit_Date was updated to reflect the test order timestamp. • Patient information including MedicalRecord.PFName, PLName, and DOB was included to ensure proper identification.

	• The doctor who ordered the test was logged using Staff.EmployeeID, StaffFname, StaffLname, StaffPosition, and Gender. • Doctor's LicenseNumber and Specialty were recorded for professional validation. • Contact details such as Staff.Phone, Staff.Email, and Staff.Address were included for further communication and accountability.	
Trigger	• Provider selects "Order Test" while documenting the patient visit.	
Included Use Cases	None	
Extending Use Cases	Documenting Visit (UC6)	
***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence** **Reference "included use cases" in this section**	**Actor Action**	**System Action**
	Step 1.Doctor is documenting a visit.	Step 2. System displays documentation tools.
	Step 3. . Provider clicks "Order Test."	Step 4.System displays list of available tests.
	Step 5. Provider selects one or more tests and adds notes.	**Step 6:** . System validates and creates test orders.

using INCLUDE *ius_name*	Step 7. System logs the order and links it to the patient visit.	Step 8. System notifies the lab or imaging department (if internal).
	Step 9. System confirms successful order creation.	

Priority in scheduling	High
Frequency	Often during diagnostic or first-time visits
Business rules and data logic	• Only valid tests from the catalog may be ordered. • Duplicate test orders within the same visit must trigger a warning. • Orders must include test type, urgency, and provider ID
Other non-functional requirements	• Orders must be logged within 2 seconds. • System must be HL7-compatible for integration with external labs. • Acknowledgement from lab must be trackable in system logs.
Superordinates	Documenting Visit (UC6)
Developer	Group 3

Creation date and last modified date	Created: 05/19/2025 Updated: 05/19/2025
Other Comments	Future releases may support test bundling or custom test panels.

USE CASE #	**UC6EXT10**
USE CASE Name	Diagnosis
ACTOR	Doctor
Goal (1 phrase)	To record one or more diagnoses based on clinical findings during the patient visit.
Overview and scope	**This use case enables the healthcare provider to document diagnostic conclusions after evaluating the patient. The diagnoses are coded (typically using ICD-10), stored in the patient's record, and used to guide treatment, billing, and follow-up care.**
Level	Extend

Preconditions	• Patient is undergoing or has completed a clinical evaluation. • The provider is authorized to assign diagnoses. • The system provides a list or search for valid ICD-10 codes.

Postconditions in words(write in passive and past tense)	• MedicalRecord.Diagnosis was updated with the doctor's clinical findings and linked to MedicalRecord.MedicalRecordID. • MedicalRecord.Last_Visit_Date was recorded to reflect the diagnosis time. • Patient identification was confirmed using MedicalRecord.PFName, PLName, and DOB. • The diagnosing doctor was logged using Staff.EmployeeID, StaffFname, StaffLname, StaffPosition, and Gender. • Doctor's LicenseNumber and Specialty were included for clinical traceability. • Staff.Phone, Email, and Address were documented to support communication and future coordination.
Trigger	• Provider selects "Add Diagnosis" during the documentation of a visit
Included Use Cases	None
Extending Use Cases	Documenting Visit (UC6)

***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	**Actor Action**	**System Action**
	Step 1. Provider is documenting the patient visit.	Step 2. System displays visit documentation tools.
	Step 3. Provider selects "Add Diagnosis."	Step 4. System opens diagnosis entry screen with ICD-10 search.
	Step 5. Provider searches for and selects appropriate codes.	**Step 6: System adds the codes and description to the record.**
	Step 7. Provider may add clinical notes or supporting info.	Step 8: System saves diagnosis and links it to the visit.
	Step 9: System confirms diagnosis entry.	
Priority in scheduling	High	
Frequency	Every visit involving a clinical assessment or follow-up	
Business rules and data logic	• Diagnoses must be mapped to standard coding systems (e.g., ICD-10). • The system must warn if an incomplete or invalid code is selected.	

	• Each diagnosis is associated with a provider and timestamp.
Other non-functional requirements	• System must support intelligent autocomplete for diagnosis search. • All entries should be saved in less than 3 seconds. • Diagnostic history must be accessible from the patient profile..
Superordinates	Documenting Visit (UC6)
Developer	Group 3
Creation date and last modified date	Created: 05/18/2025 Updated: 05/19/2025
Other Comments	Future versions may include AI-assisted diagnosis suggestions based on symptoms and history.

USE CASE #	**UC6EXT11**
USE CASE Name	Write Referal
ACTOR	Doctor
Goal (1 phrase)	To refer the patient to a specialist or external provider for further evaluation or treatment.
Overview and scope	**This use case enables healthcare providers to generate a referral to another physician, specialist, or external facility. The referral includes diagnosis, reason for referral, urgency level, and any supporting documents. The referral record is saved and optionally sent electronically to the receiving provider.**
Level	Extend

Preconditions	• The patient is undergoing or has completed a clinical evaluation. • The provider is authorized to issue referrals. • The referral directory or database is available.
Postconditions in words(write in passive and past tense)	• The doctor entered referral notes into Visit.Purpose, and the referral facility information was temporarily recorded in Visit.Diagnosis_Summary. • Appointment.Appointment_Date and Appointment.Reason were reviewed to ensure alignment with the referral purpose. • Visit.Status was updated to "Referred", and Visit.Visit_ID and Appointment_ID were linked to maintain continuity. • The doctor was logged using Staff.EmployeeID, StaffFname, StaffLname, StaffPosition, and Gender. • Doctor's LicenseNumber and Specialty were included to specify the referral source and provider qualifications. • Staff.Phone, Email, and Address were documented for referral correspondence and auditing purposes.
Trigger	• Provider selects "Write Referral" while documenting the patient visit.

Included Use Cases	None	
Extending Use Cases	Documenting Visit (UC6)	
***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	**Actor Action**	**System Action**
	Step 1. Doctor is documenting the visit.	Step 2. System displays documentation interface.
	Step 3. Provider selects "Write Referral."	Step 4. System displays referral entry form with specialist list.
	Step 5. Provider selects specialist type and enters reason.	**Step 6: System validates the input and displays optional details.**
	Step 7. Provider enters additional notes or uploads attachments.	Step 8: System saves the referral and links it to the visit.
	Step 9: System optionally sends notification to the recipient.	Step 10.System confirms referral creation.
Priority in scheduling	Medium	
Frequency	Occasionally – when specialist care is required	

Business rules and data logic	• Each referral must be linked to a valid diagnosis. • Only licensed providers may initiate referrals. • System must allow attaching relevant reports or visit summaries.
Other non-functional requirements	• Referrals must be saved and acknowledged within 5 seconds. • The referral system must support export to PDF or HL7-compatible format. • Audit trail must include referral status (e.g., sent, pending, accepted).
Superordinates	Documenting Visit (UC6)
Developer	Group 3
Creation date and last modified date	Created: 05/18/2025 Updated: 05/19/2025
Other Comments	In the future, referral tracking and feedback loops from the specialist can be added to close the care loop.

USE CASE #	**UC6EXT12**
USE CASE Name	Write Prescriptions
ACTOR	Doctor
Goal (1 phrase)	To prescribe medication for the patient during or after the visit.
Overview and scope	**This use case allows providers to create and issue prescriptions for medications. The prescription includes drug name, dosage, route, duration, and instructions. It may be printed or electronically sent to a pharmacy through an integrated e-prescription system.**
Level	Extend
Preconditions	• The provider is logged into the system with prescription privileges. • The patient is undergoing or has completed evaluation. • The medication database is available and up to date.

Postconditions in words(write in passive and past tense)	• A prescription was issued with Prescription.prescription_id and linked to the corresponding visit using Prescription.visit_id and Visit.Visit_ID. • Prescription.medication_name, dosage, quantity, and duration were recorded in the system. • Prescription.issued_date captured the timestamp of prescription creation. • The prescribing doctor was logged using Staff.EmployeeID, StaffFname, StaffLname, StaffPosition, and Gender. • Doctor's LicenseNumber and Specialty were saved to verify prescriber credentials. • Staff.Phone, Email, and Address were stored to facilitate pharmacy follow-up and auditing.
Trigger	• Provider selects "Write Prescription" during visit documentation.
Included Use Cases	None
Extending Use Cases	Documenting Visit (UC6)

***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	**Actor Action**	**System Action**
	Step 1. Doctor is documenting the visit.	Step 2. System shows visit documentation options.
	Step 3.Provider clicks "Write Prescription."	Step 4.System displays medication search and entry form.
	Step 5. Provider searches and selects a drug from the database.	**Step 6:** System displays standard dosage options and alerts.
	Step 7. Provider completes dosage, frequency, and instructions.	Step 8: System saves the prescription and links it to the visit.
	Step 9: Provider chooses print or e-prescribe option..	Step 10. System executes chosen method and confirms success..
Priority in scheduling	High	
Frequency	Frequently during visits involving treatment	
Business rules and data logic	• Only prescribable medications are shown. • Drug interaction alerts must be shown based on active medications.	

	• DEA and license validation is required for controlled substances.
Other non-functional requirements	• Prescription must be saved within 3 seconds. • E-prescription system must be compliant with regulatory standards (e.g., EPCS). • Audit trail must include prescriber ID, timestamp, and medication details.
Superordinates	Documenting Visit (UC6)
Developer	Group 3
Creation date and last modified date	Created: 05/18/2025 Updated: 05/19/2025
Other Comments	In future versions, integration with pharmacy networks can enable real-time availability and cost display. .

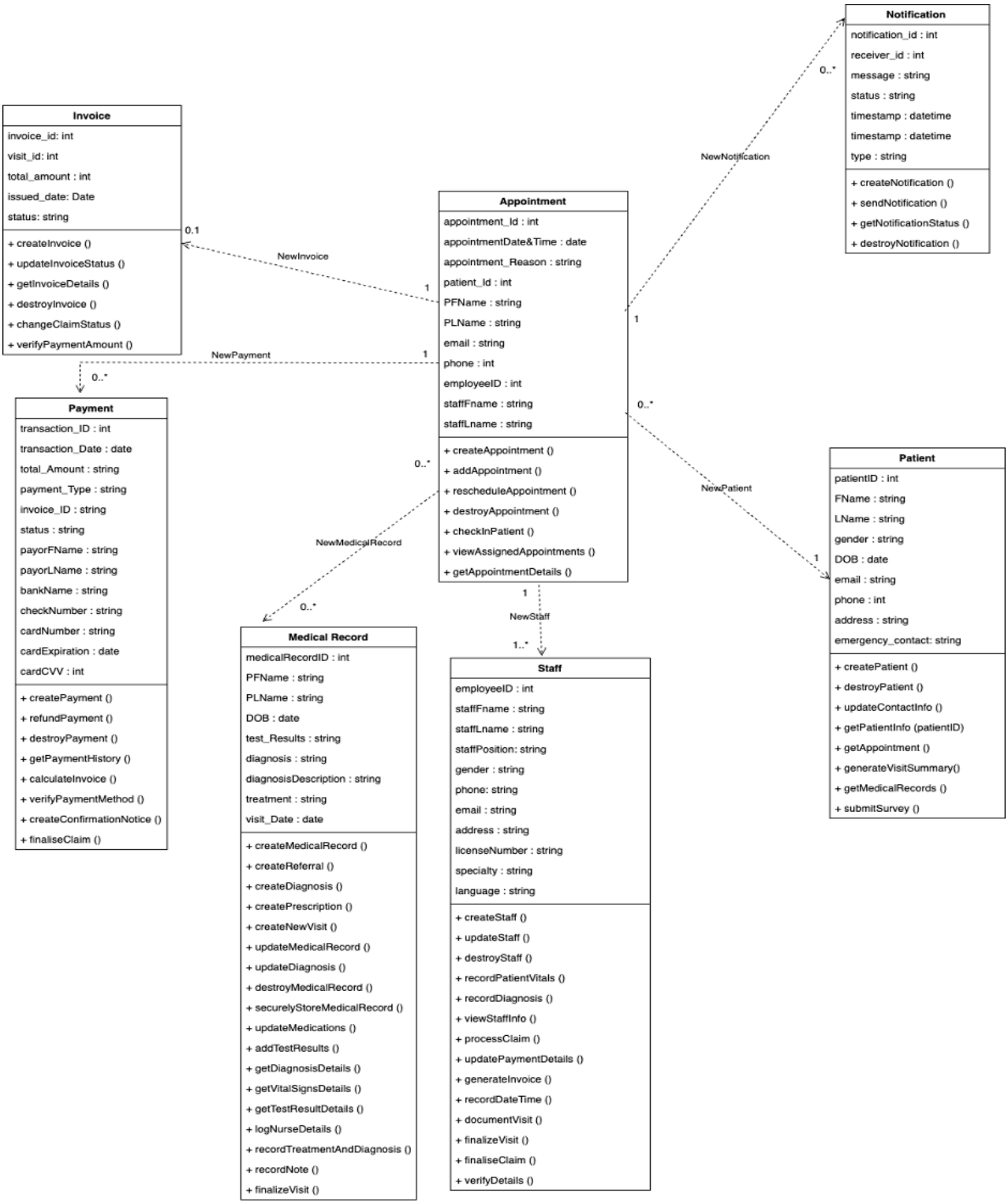
USE CASE #	**UC7**
USE CASE Name	Process Claims
ACTOR	Accountant
Goal (1 phrase)	To generate and process insurance claims for completed patient visits.
Overview and scope	**This use case allows billing personnel to generate claims based on services rendered during a visit. It includes compiling codes for procedures and diagnoses, verifying patient insurance details, and submitting the claim to the appropriate insurance provider. Responses such as approval or denial are recorded for follow-up.**
Level	Base
Preconditions	• The visit has been completed and documented. • Patient insurance information is validated. • Billing staff is authenticated.

Postconditions in words(write in passive and past tense)	• A claim was created with Claim.Claim_ID and linked to Visit.Visit_ID for billing purposes. • Claim.Submitted_Date recorded the time of claim submission, and Claim.Status was updated to reflect processing progress. • Claim.Insurance_Company was captured to indicate the responsible payer. • An Invoice.Invoice_ID was generated or updated and associated with both Claim and Visit. • Invoice.Total_Amount and Issued_Date reflected the billed service value and its creation time. • Invoice.Status was initialized as "Pending" for follow-up tracking. • The billing staff member was logged using Staff.EmployeeID, StaffFname, StaffLname, StaffPosition, and Gender. • Staff.Phone, Email, and Address were recorded for payment inquiries and audit trail maintenance.
Trigger	• Billing staff selects “Process Claims” from the billing dashboard.

Included Use Cases	None	
Extending Use Cases	None	
***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	**Actor Action**	**System Action**
		Step 1.System presents billing dashboard with eligible visits..
	Step 2. Staff selects a completed patient visit.	Step 3.System displays visit summary, services, and diagnoses
	Step 4. Step 4. Staff reviews and confirms claim details and verifies the payment amount and method.	**Step 5:** System generates claim form with coded procedures.
	Step 6. Staff submits the claim to the insurer.	Step 7: System sends the claim and logs submission timestam
	Step 8: System receives response from insurer.	Step 9. System updates claim status (e.g., submitted, approved).
Priority in scheduling	High	
Frequency	Daily – for every completed billable visit	

Business rules and data logic	• Each claim must contain valid CPT/ICD-10 codes. • Duplicate claim submissions must be prevented. • Claims must match the patient's coverage and provider agreements.
Other non-functional requirements	• Integration with clearinghouses and insurance APIs. • Claims must be logged within 2 seconds of submission. • A dashboard must show claim statuses and alerts for pending or denied claims. •
Superordinates	Billing and Revenue Cycle Management
Developer	Group 3
Creation date and last modified date	Created: 05/18/2025 Updated: 05/19/2025
Other Comments	Future enhancements could include AI-based claim validation and predictive rejection alerts.

8. Design Class Diagram:



The Design Class Diagram for the Patient Care Clinic System (PCCS) illustrates the main components and interactions involved in managing clinical operations such as patient registration, appointment scheduling, medical documentation, payments, and system notifications.

At the core of the system is the Appointment class, which connects patients and staff members. It contains appointment-specific attributes like date, time, reason, and participant details. Appointments can be created, rescheduled, checked in, and managed through various methods tied to this class.

The Patient class captures essential personal and contact information, including emergency contacts. Patients are able to register, update their profile, view appointments, generate visit summaries, and submit feedback through system interactions.

The Staff class represents clinic personnel including administrative and medical professionals. It includes attributes such as name, position, contact details, license number, and specialty. Staff members perform key system operations like recording vitals, updating medical records, processing claims, generating invoices, and managing appointment schedules.

The Medical Record class manages the clinical history of patients, covering test results, diagnoses, prescriptions, treatments, and visit notes. It supports secure creation, retrieval, and update of health data tied to individual appointments and staff actions.

The Invoice and Payment classes handle billing and transaction processing. Invoices track the total amount due for each visit, along with status and issue date. Payments record transaction

details such as payment method, payer information, and support functions like refunds, confirmations, and history tracking. Payments are directly linked to invoices for traceability.

The Notification class is responsible for sending system alerts to patients, such as reminders or medical updates. Each notification includes a message, timestamp, status, and type, and is associated with a specific recipient.

Overall, this design class diagram reflects a structured and cohesive architecture that supports the core functions of a small-to-medium size healthcare clinic, emphasizing maintainability, modularity, and real-world alignment.

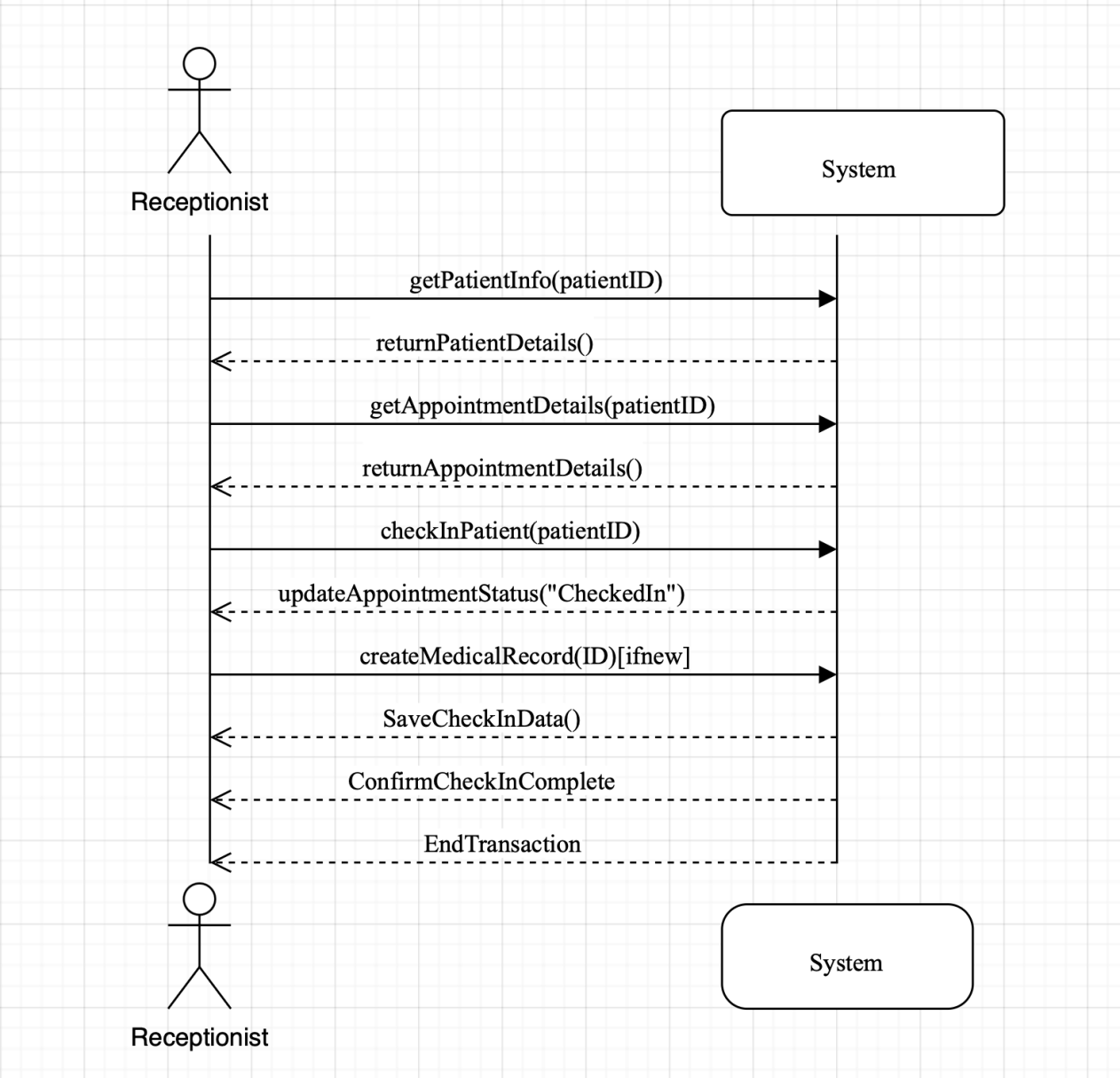
8.1 System Sequence Diagrams:

UC1: Document Patient Check-In

Actor: Receptionist

Description: The receptionist checks in a patient (new or existing) upon arrival at the healthcare facility. This includes verifying the patient's identity, confirming appointment details, and updating the system to reflect their arrival. If the patient is new, the receptionist initiates the new patient registration process. This ensures the patient is properly registered and ready to be seen by medical staff.

Diagram by Minahil:

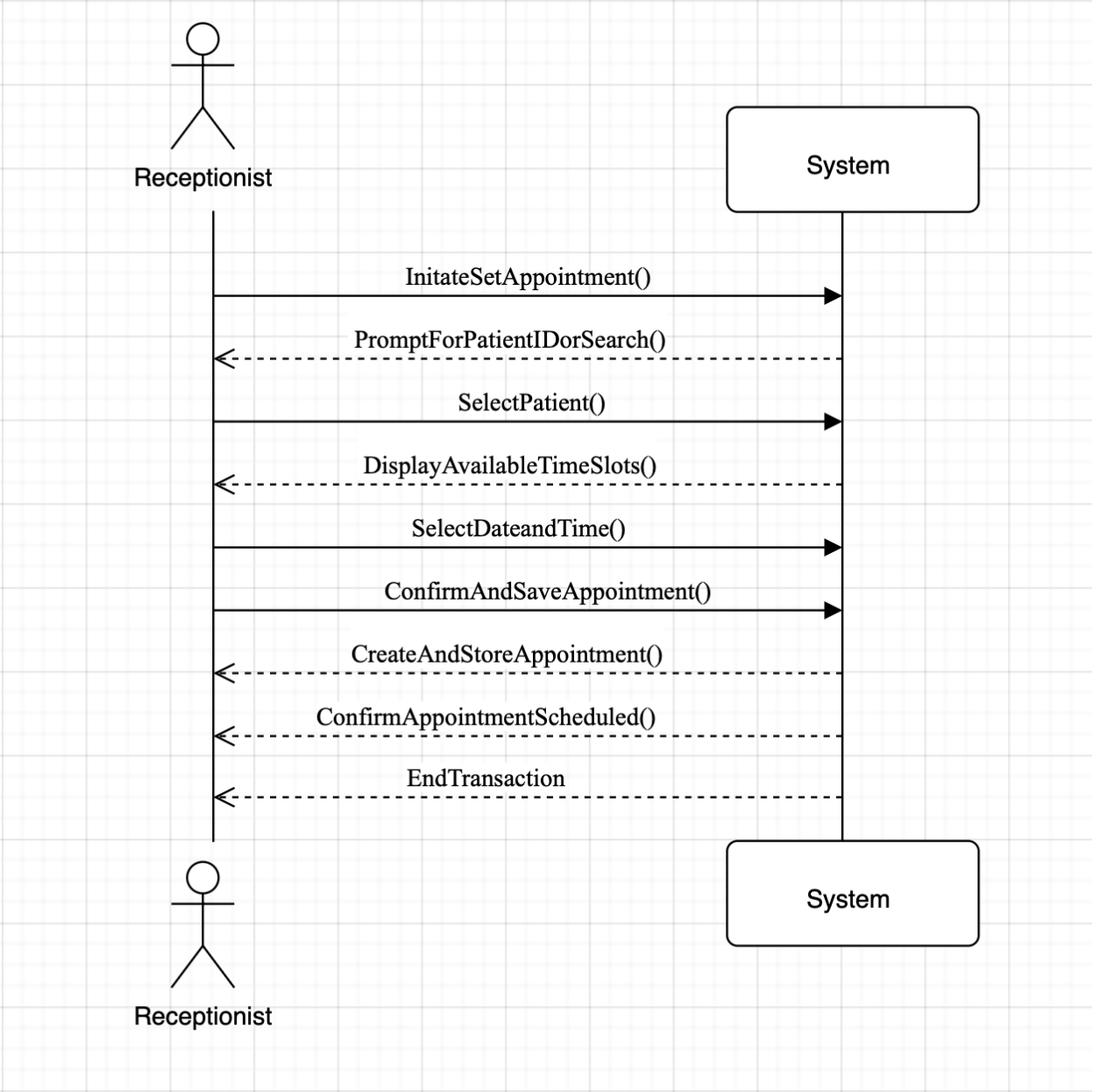


UC2: Set Appointment

Actor: Receptionist

Description: The receptionist schedules a new appointment for a patient using the clinic's appointment system. This process involves identifying the patient, selecting an available date and time, assigning a provider if necessary, and confirming the appointment details. Once confirmed, the system stores the appointment and notifies the patient. This ensures appointments are accurately recorded and patients are scheduled efficiently for future visits.

Diagram by Minahil:

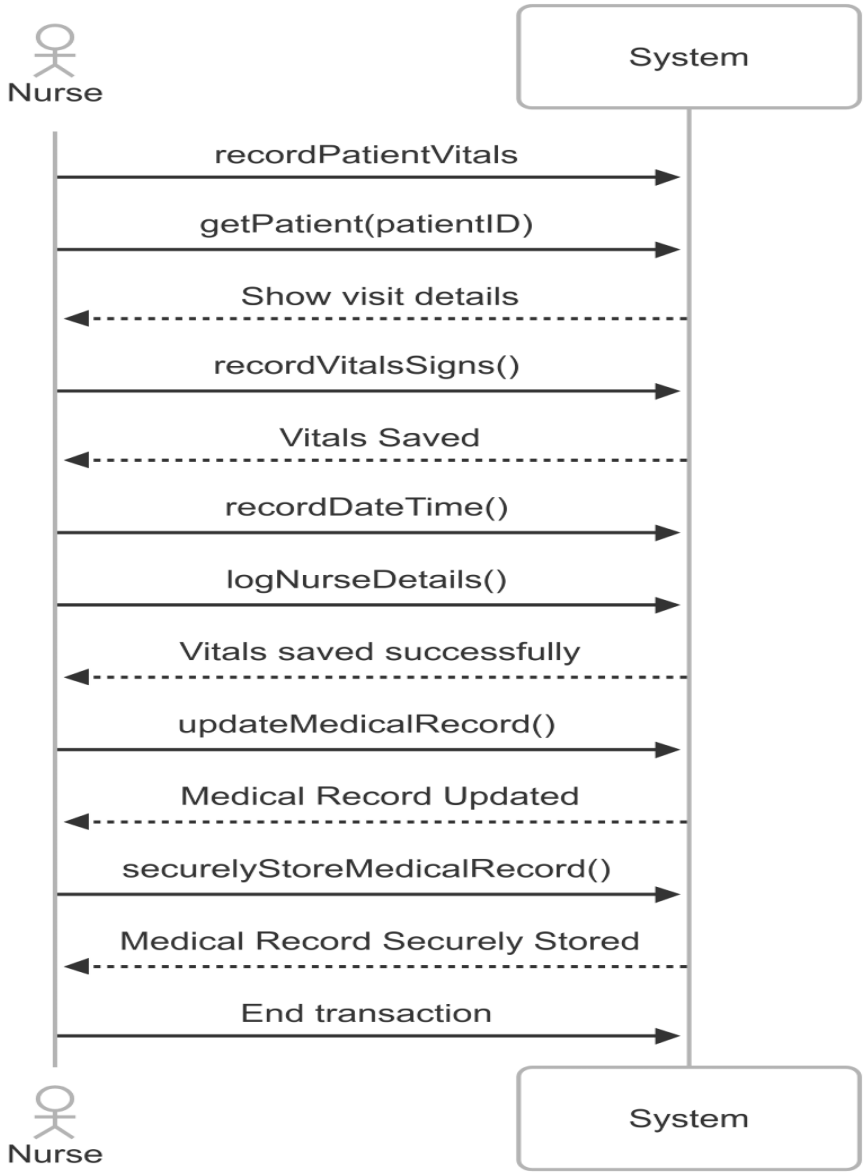


UC5: Record Vitals

Actor: Nurse

Description: The nurse records the patient's vital signs such as temperature, blood pressure, respiratory rate, etc.

Diagram by Ram:

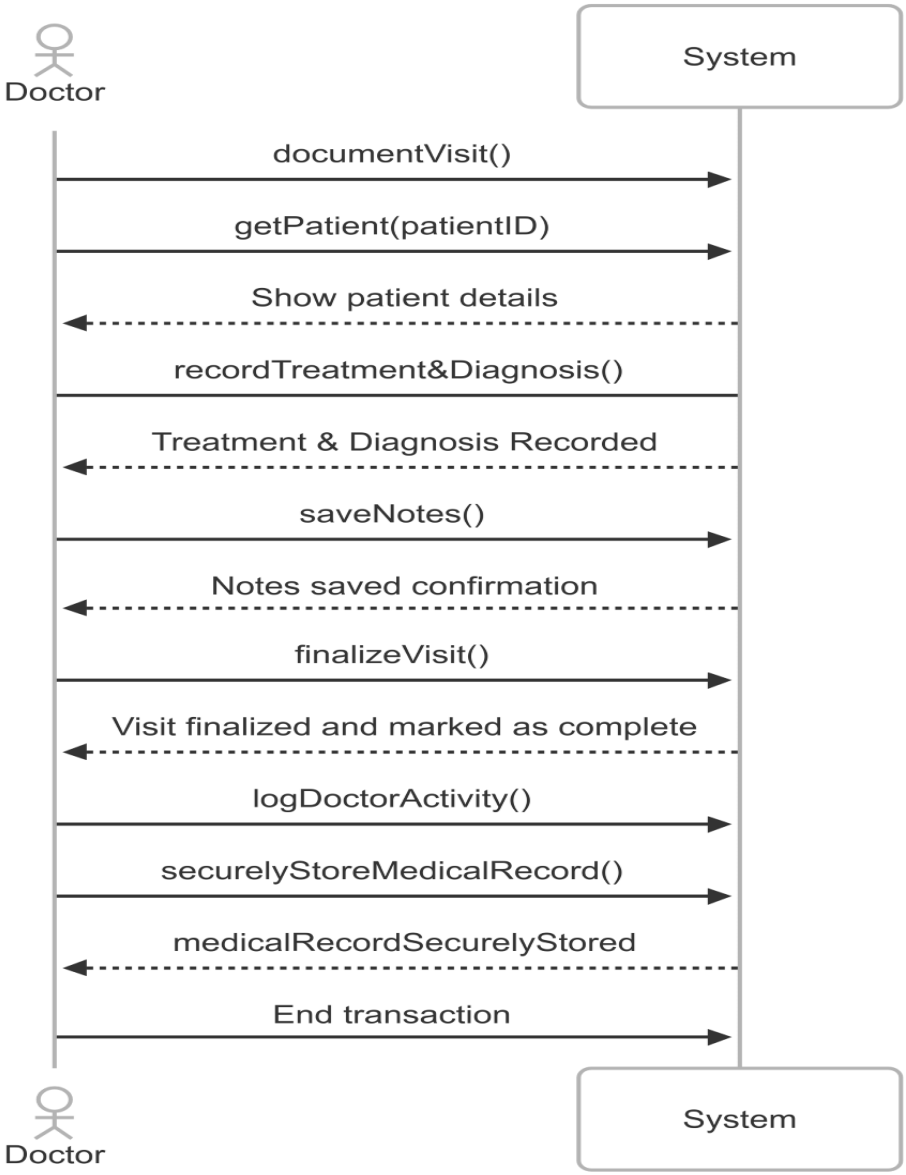


UC6: Document Visit

Actor: Doctor

Description: The doctor documents the patient's visit by entering symptoms, observations, treatment, and finalizing the visit note.

Diagram by Ram:

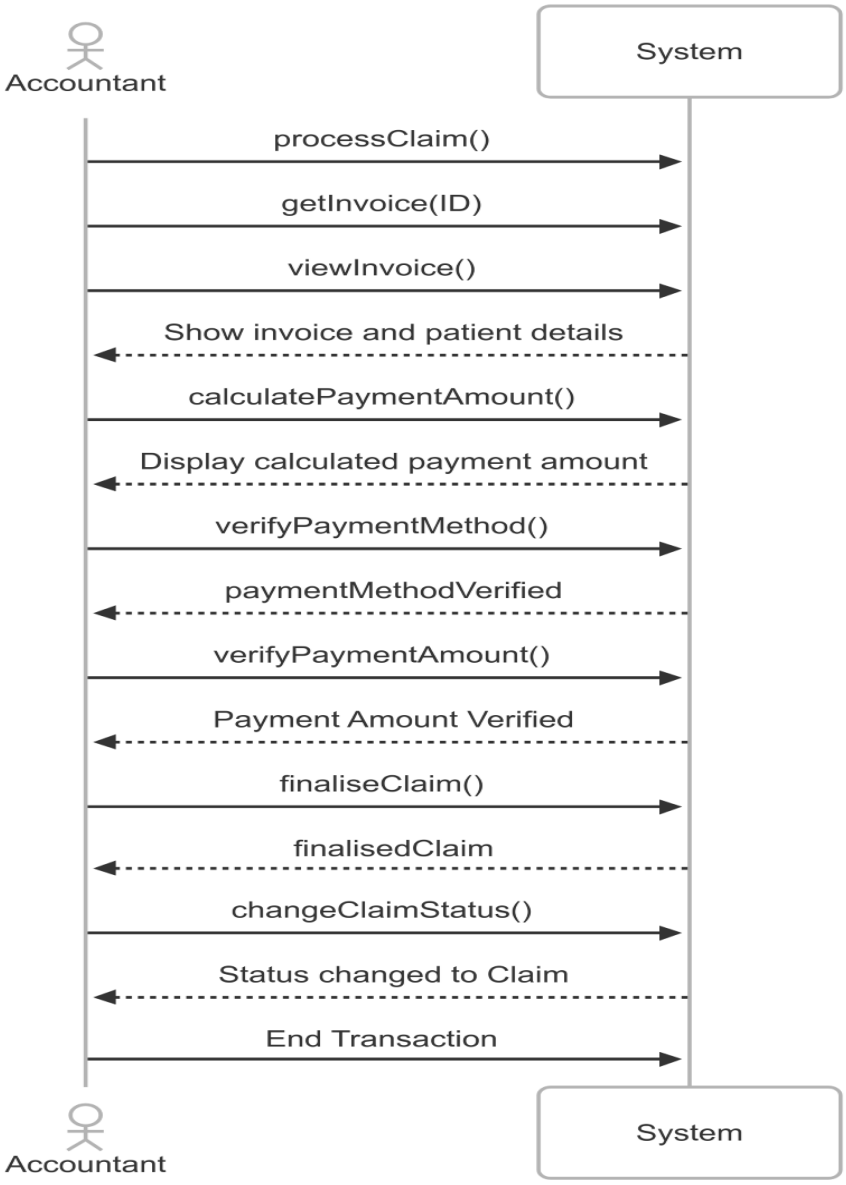


UC7: Process Claims

Actor: Accountant

Description: The accountant processes the claims by reviewing invoices, verifying payments, and submitting claims into the system.

Diagram by Ram:



8. 2 Sequence Diagrams:

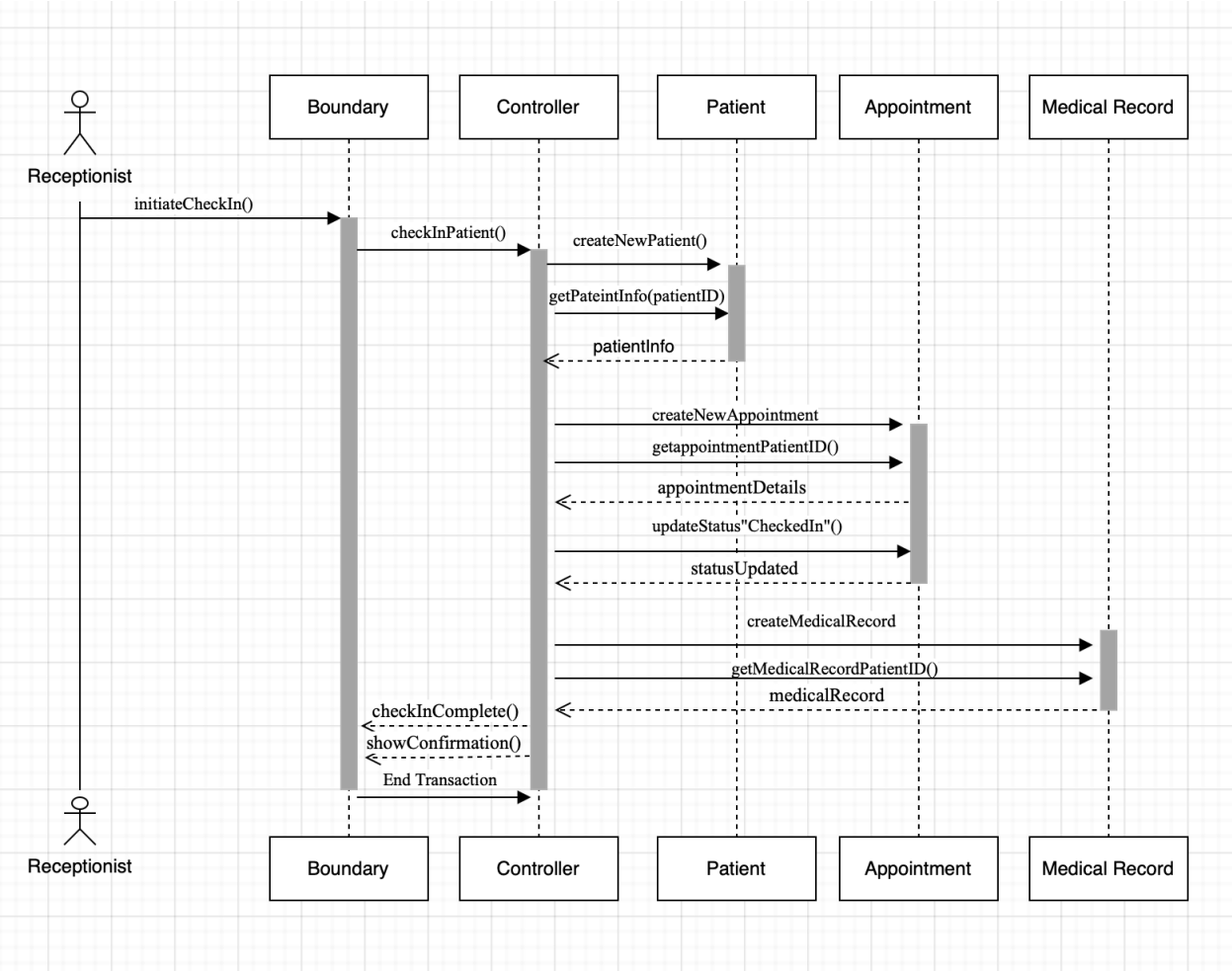
UC1: Patient Check-In

Actor: Receptionist

Goal: To check in an existing or new patient, confirm their appointment, and ensure medical

records are updated.

Diagram by: Minahil



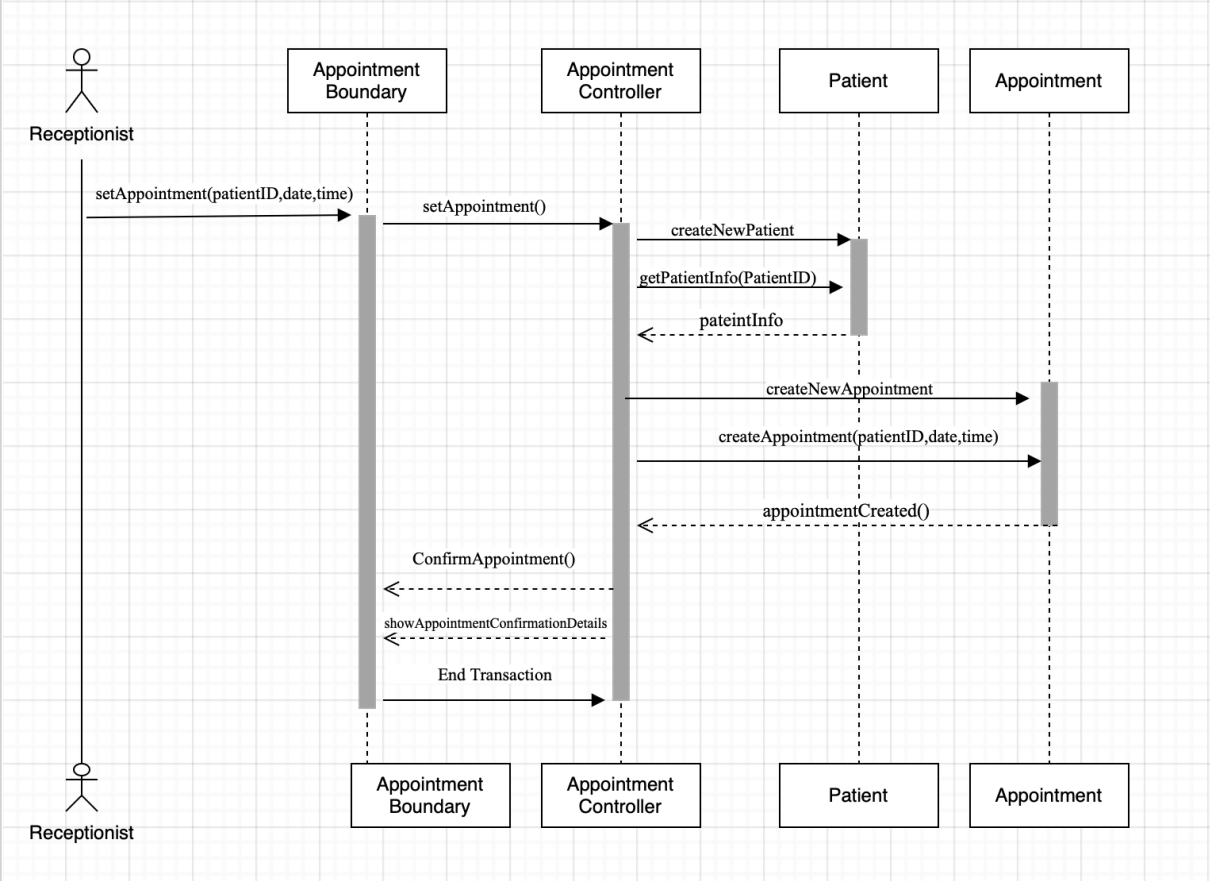
UC2: Set Appointment

Actor: Receptionist

Description: The receptionist schedules a new appointment for a patient using the system interface. The process begins when the receptionist initiates the appointment setup through the AppointmentBoundary. The AppointmentController then coordinates the retrieval of patient details from the Patient class and displays available time slots. Upon selecting the date, time, and

provider (if applicable), the controller creates a new appointment record in the Appointment class. Finally, the system confirms that the appointment has been successfully scheduled.

Diagram by: Minahil

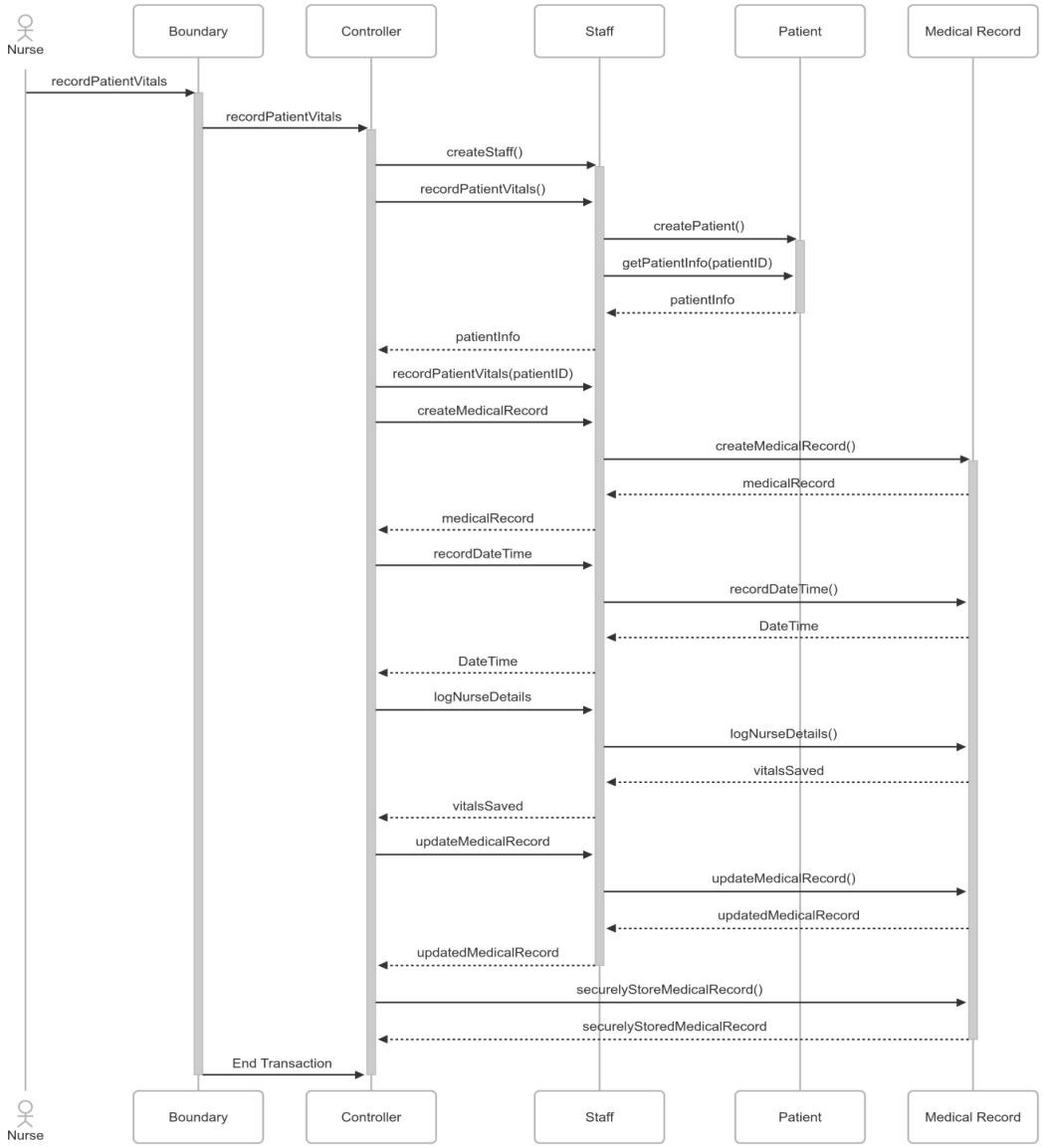


UC5: Record Vitals

Actor: Nurse

Description: The nurse records the patient's vital signs such as temperature, blood pressure, respiratory rate, etc.

Diagram by: Ram

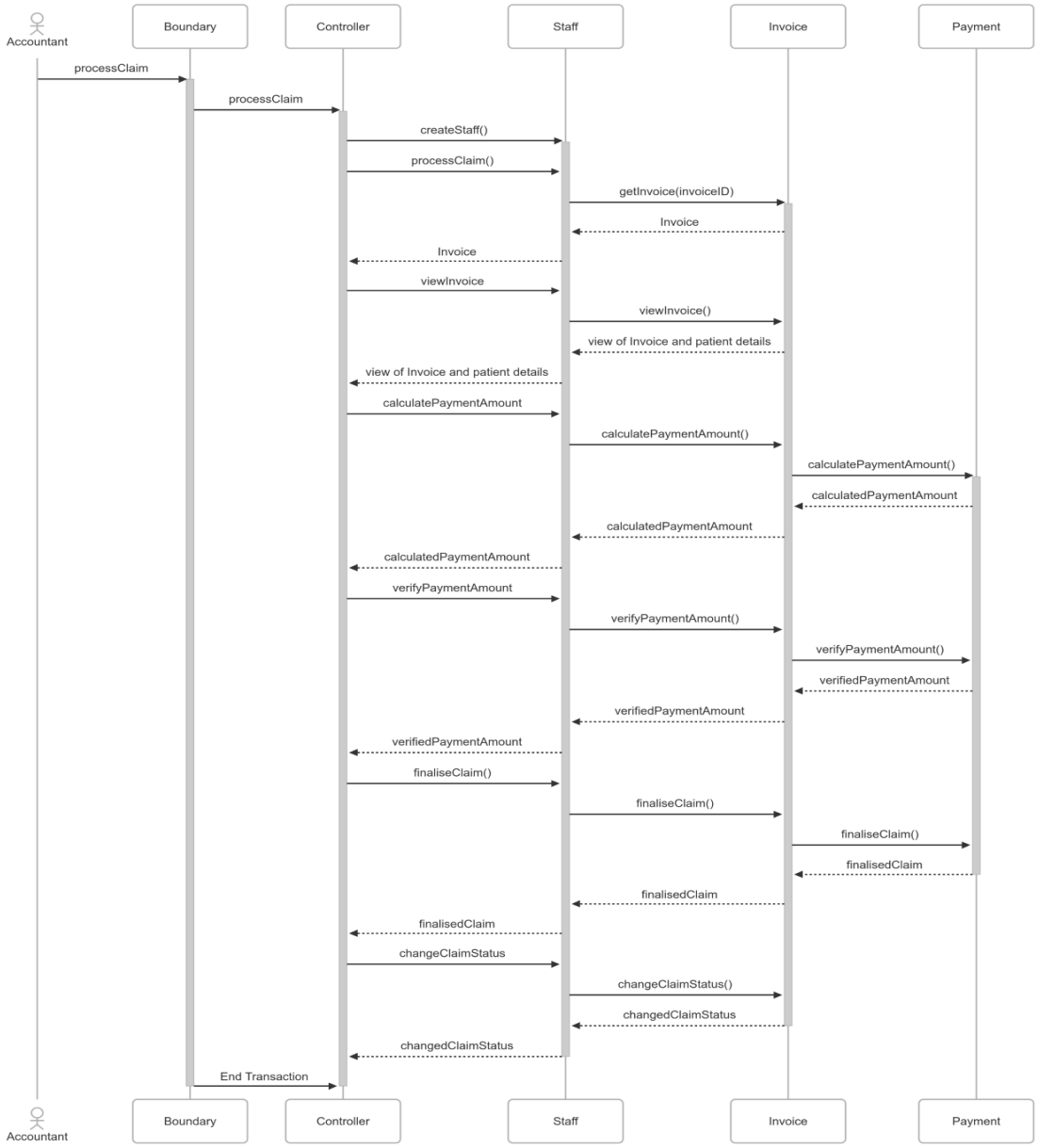


UC6: Document Visit

Actor: Doctor

Description: The doctor documents the patient's visit by entering symptoms, observations, treatment, and finalizing the visit note.

Diagram by: Ram

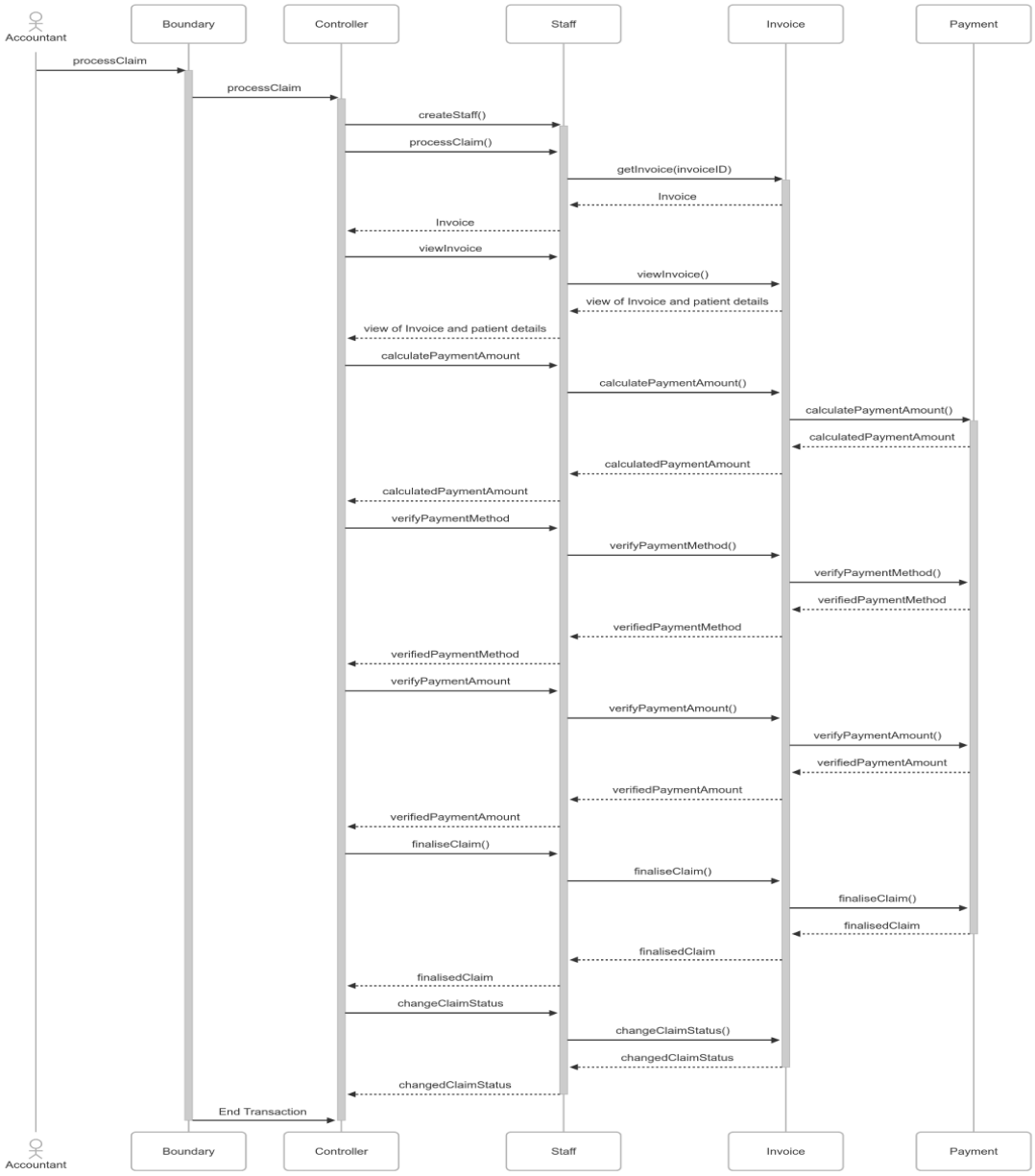


UC7: Process Claims

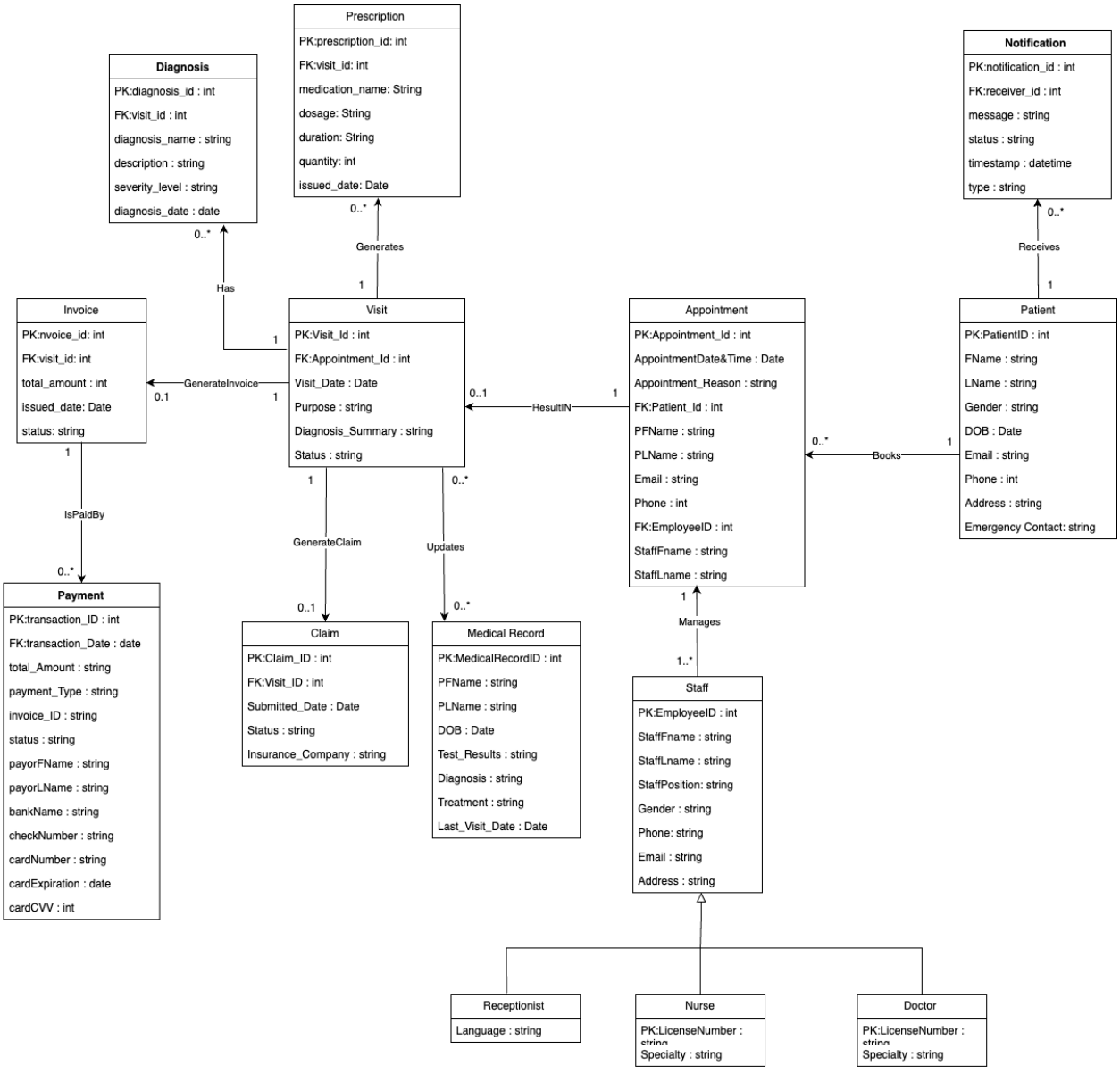
Actor: Accountant

Description: The accountant processes the claims by reviewing invoices, verifying payments, and submitting claims into the system.

Diagram by: Ram



9. Relational Database Schema:



10. Analysis section:

Working on the Smart Clinic system was a hands-on experience that bridged theory and real-world application in a healthcare context. From the outset, we moved through all stages of

system design, starting with defining key features and ending with a complete set of UML diagrams.

In Milestone 1, we identified the domain and developed comprehensive use cases through extensive brainstorming sessions. These covered critical clinical activities such as taking vitals, documenting visits, and filing insurance claims. We defined actors, goals, triggers, and success scenarios to mirror real clinical workflows.

By Milestone 2, we translated those use cases into full descriptions, clearly laying out event sequences and applying includes and extends where necessary. This pushed us to think structurally about system behavior and human interaction, improving clarity and consistency across our documentation.

Milestone 3 was the most technically demanding. We worked on domain class diagrams, system sequence diagrams, and detailed sequence diagrams. One major challenge was defining appropriate attributes and methods for each class without overcomplicating the design. We went through multiple iterations to fine-tune entities like Visit, Invoice, Staff, and MedicalRecord, making sure the relationships and behaviors reflected real-world clinical operations. Consistency in naming conventions and alignment with earlier documentation were top priorities.

Collaboration was tough. Working in a group meant navigating differing ideas, balancing responsibilities, and staying coordinated. But the challenge paid off. We split the workload smartly, regularly reviewed each other's contributions, and held each other accountable. It wasn't just a technical learning experience; it was a crash course in communication, delegation, and problem-solving under pressure.

Overall, this project sharpened our skills in system analysis and design and deepened our understanding of applying technical concepts in real-world scenarios. Just as important, it pushed us to grow as collaborators and thinkers, making the learning process both effective and meaningful.