# Software Quality Assurance Project Deliverable (SQAPD)

**PREPARED FOR**

John Doe
Health Connect

**PREPARED BY**

GioMonci, Joseph L, David R, Derek M
Dr. Buckley’s Quality Solutions

# Project Overview

Health Connect is a telehealth platform designed to streamline and secure asynchronous consultations between patients and healthcare providers. Key objectives include:

- Access: Enable registered patients to submit consultation requests and upload health information securely.
- Efficiency: Provide doctors with an organized dashboard for request management, tracking, and status updates.
- Security & Compliance: Ensure HIPAA-compliant data handling, role-based access, encryption, and comprehensive audit logging.
- Scalability: Support growth in user base and feature expansion through modular architecture.

Stakeholders: Patients, Physicians, Pharmacists, System Administrators, and IT Security Teams.

Technology Stack: Java Swing frontend, JDBC backend, PostgreSQL database, hosted on AWS.

**Signed as accepted by client:**

John Doe, CEO of Health Connect April 21st, 2025

[NAME], [TITLE] [DATE]

# Contents

# 1. Requirements Phase

## 1.1. Introduction

1.1.1. This section presents the results of the SQA Requirements Phase for Health Connect. It includes:

1.1.1.1. Inspection & Review of existing requirements.

1.1.1.2. Corrected Requirements, reformatted and enhanced.

1.1.1.3. Risk Matrix for all requirements with mitigation plans.

1.1.1.4. Evaluation & Assessment Process—checklists and metrics to minimize requirement defects.

## 1.2. Requirements Inspection & Review

1.2.1. Inspection Method:

1.2.1.1. Peer Reviews: Two independent reviewers examined each requirement. (Giovanni & Joseph)

1.2.1.2. Standards Referenced: ISO/IEC/IEEE 29148, SMART, CUPRIMDSO, HIPAA.

1.2.1.3. Quality Metrics: Completeness, Clarity, Atomicity, Testability, Consistency.

| Issue | Description | Impact | Severity |
|---|---|---|---|
| Missing Requirements | No use cases for Authentication, Medical History, Assign Doctor, Data Export | Gaps in core functionality. | High |
| Ambiguity | “Manage requests” is undefined (create vs. view vs. update). | Misinterpretation by developers. | Medium |

| | | | |
|---|---|---|---|
| Not Atomic | Combined actions in a single statement (e.g., “verify and sanitize input”). | Difficult to test; violates single-responsibility. | Medium |
| Lack of testability | No acceptance criteria or measurable thresholds (e.g., performance, notification delays). | Unclear when requirement is met. | High |
| Inconsistency | Terminology varies (“doctor” vs “nurse” vs. “assistant” vs. “user”). | Confusion; traceability issues. | Medium |
| Traceability Gaps | No mapping from requirements to risk or design artifacts. | Hard to track coverage and mitigate risk. | High |

## 1.3. Corrected & Reformatted Requirements:

### 1.3.1. Functional Requirements:

#### 1.3.1.1. User Profile Management:

1.3.1.1.1. The User Profile Management system shall allow a new Patient to register by providing full name, date of birth, email, and password.

1.3.1.1.2. The User Profile Management system shall allow a new Doctor to register by providing full name, medical license number, email, and password.

1.3.1.1.3. The User Profile Management system shall enforce role-based access control (RBAC), differentiating patients and doctors.

1.3.1.1.4. The User Profile Management system shall prevent duplicate registrations by checking email uniqueness.

1.3.1.1.5. The User Profile Management system shall allow users to update their profile except for unique identifiers.

1.3.1.1.6. The User Profile Management system shall secure credentials using salted password hashing and optional multi-factor authentication (MFA).

1.3.1.2. Patient Request Management:

1.3.1.2.1. Only registered Patients shall submit consultation requests.

1.3.1.2.2. Each consultation request shall receive a unique Request ID and timestamp upon submission.

1.3.1.2.3. Patients shall view only their own requests; attempts to access others' requests shall be denied.

1.3.1.2.4. The system shall enforce input validation and sanitization for request data to prevent injection attacks.

1.3.1.2.5. The system shall notify the assigned Doctor within 1 minute of request submission.

1.3.1.3. Doctor Request Handling:

1.3.1.3.1. Doctors shall see only unassigned Patient requests in their dashboard.

1.3.1.3.2. Upon opening a request, the system shall assign it exclusively to that Doctor.

1.3.1.3.3. Doctors shall update request status to In Progress or Closed, following the sequence Open → In Progress → Closed.

1.3.1.3.4. The system shall log every status change with timestamp, Doctor ID, and notes.

1.3.1.4. Request Status Management:

1.3.1.4.1. The system shall escalate requests remaining in Open state for over 24 hours by sending a reminder to a supervisor.

1.3.1.4.2. Closed requests shall be archived and locked against further edits.

1.3.1.4.3. The system shall provide filtering of requests by status (Open, In Progress, Closed) in the Doctor dashboard.

**1.3.2. Non-Functional Requirements:**

1.3.2.1. Performance:

1.3.2.1.1. The web interface shall respond to user actions within 2 seconds under normal load (< 100 concurrent users).

1.3.2.1.2. Notification delivery shall complete within 1 minute of trigger.

1.3.2.2. Security & Compliance:

1.3.2.2.1. All data at rest and in transit shall be encrypted using AES-256 and TLS

1.3.2.2.2. The system shall comply with HIPAA and ISO/IEC 27001 standards.

1.3.2.2.3. User sessions shall expire after 15 minutes of inactivity.

1.3.2.2.4. Access to API endpoints shall be restricted by user role.

1.3.2.3. Reliability & Availability:

1.3.2.3.1. The system shall achieve 99.5% uptime monthly, excluding scheduled maintenance.

1.3.2.3.2. Automatic backups of the database shall occur every 6 hours.

1.3.2.4. Maintainability & Scalability:

1.3.2.4.1. The architecture shall support horizontal scaling to add capacity for a 50% increase in users within 3 months.

1.3.2.4.2. All code modules shall include unit tests covering ≥ 80% of statements.

**1.3.3. Other Requirements:**

1.3.3.1. The system shall integrate with external EHR systems via HL7 messaging

1.3.3.2. Audit logs shall be retained for a minimum of 7 years

## 1.4. Risk Matrix

| ID | Risk Description | Prob | Impact | Exposure | Priority | Mitigation Plan | Owner |
|---|---|---|---|---|---|---|---|
| R1 | Unauthorized data access | 4 | 5 | 20 | High | Enforce AES-256/TLS; RBAC; monthly security audits | Security Lead |
| R2 | Request not escalated within SLA | 3 | 3 | 9 | Mid | Automated reminder service; | DevOps |
| R3 | Requirements omissions leading to feature gaps | 4 | 4 | 16 | High | Maintain traceability matrix; biweekly requirement reviews | QA Manager |
| R4 | Performance degradation under peak load | 3 | 4 | 12 | Mid | Load-test at 2 times expected load; enable autoscaling | DevOps |
| R5 | Data loss due to backup failure | 2 | 5 | 10 | Mid | Implement multi-region backups; weekly restore drills | Database Admin |

## 1.5. Evaluation & Assessment Process

### 1.5.1. Quality Metrics:

| Metric | Description | Scale | Threshold |
|---|---|---|---|
| Completeness | All system functions and edge cases covered | 1-5 | >4 |
| Clarity | Unambiguous, consistent terminology | 1-5 | >4 |
| Atomicity | Single action per requirement | 1-5 | 5 |
| Testability | Measurable criteria or acceptance test defined | 1-5 | >4 |
| Consistency | Terminology and style uniform across requirements | 1-5 | 5 |

### 1.5.2. Inspection Checklist:

| # | Checklist Item | Description |
|---|---|---|
| 1 | Traceability | Mapped in the Traceability Matrix to risks and design artifacts. |
| 2 | Single Action (Atomicity) | Contains exactly one verb phrase (one testable action). |
| 3 | Defined Terms | All domain-specific terms appear in the glossary or approved term list. |
| 4 | Measurable Criteria | Acceptance criteria or thresholds are explicitly defined (e.g., time, counts, states). |
| 5 | Consistent Terminology | Uses approved terminology uniformly (e.g., Patient, Doctor, Request). |

| | | |
|---|---|---|
| 6 | Complete Peer Comments | All reviewer comments have been addressed and closed in the Review Report. |
| 7 | Formatting & Syntax | Follows the hierarchical numbering and style guidelines consistently. |

### 1.5.3. Process Workflow:

1.5.3.1. Requirement Draft

1.5.3.1.1. Analysts author initial requirements (Artifact: Requirements document).

1.5.3.2. Peer:

1.5.3.2.1. The QA team applies the Inspection Checklist to each requirement.

1.5.3.2.2. Defects are logged in a document, referencing specific checklist items.

1.5.3.3. Defect:

1.5.3.3.1. QA Manager reviews logged defects, classified by severity (High/Med/Low), and assigns back to authors.

1.5.3.4. Corrections & Revision:

1.5.3.4.1. Authors revise requirements per defect log, document changes in Change Log document.

1.5.3.5. Verification Pass:

1.5.3.5.1. The QA team re-executes the checklist on revised requirements; all items must be ✓.

1.5.3.5.2. Any unresolved defects return to step 3.

#### 1.5.3.6. Final Approval:

1.5.3.6.1. The QA Manager reviews the Verification Report.

1.5.3.6.2. If all quality metrics meet thresholds, formal sign-off is captured in the Sign Off Approval document.

1.5.3.6.3. Requirements are baselined and locked.

#### 1.5.3.7. Changes:

1.5.3.7.1. Post-baseline changes require a Change Request (change form) and Impact Analysis.

1.5.3.7.2. Change Board evaluates, approves, and updates baseline as needed.

# 2. Design Phase

## 2.1. Inspection of UML diagrams

2.1.1. Scope: Class Diagram & Use Case Diagram

2.1.2. Checklist for Inspection:

2.1.2.1. Verify that all domain classes required by the FRs (Patient, Doctor, Supervisor, MedicalHistory, Request, Conversation, AuthenticationManager, NotificationService, ErrorHandler) are present.

2.1.2.2. Check each association for correct type (association vs. aggregation vs. composition), direction, and cardinality (e.g. Patient 1—* Request, Request 1—1 Doctor).

2.1.2.3. Ensure generalization hierarchies are modeled for shared behavior (e.g. UserView → PatientView / DoctorView).

2.1.2.4. Confirm use cases cover all FRs: RegisterPatient, RegisterDoctor, Login, AuthenticateUser (with optional MFA), GetMedicalHistory, CreateConsultationRequest, AssignDoctor, EscalateRequest, ViewProfile, ViewRequests.

2.1.2.5. Identify missing actors (e.g. Supervisor for escalations) and role distinctions (Patient vs. Doctor vs. System Admin).

2.1.2.6. Validate that error-handling flows and notification triggers are represented (ErrorHandler, NotificationService).

## 2.2. Review of Findings

2.2.1. Class Diagram Review (See in Appendix A)

2.2.1.1. Missing Domain Classes: Patient, Doctor, Supervisor, etc.

2.2.1.2. Repetition: PatientView and DoctorView share attributes/methods

2.2.1.3. Associations: Many lines unlabeled or missing cardinalities; font too small.

2.2.2. Use Case Diagram Review (See in Appendix A)

2.2.2.1. Coverage Gaps:

2.2.2.1.1. RegisterPatient / RegisterDoctor

2.2.2.1.2. GetMedicalHistory, AssignDoctor, EscalateRequest

2.2.2.1.3. Login / AuthenticateUser with <<extend>> for MFA

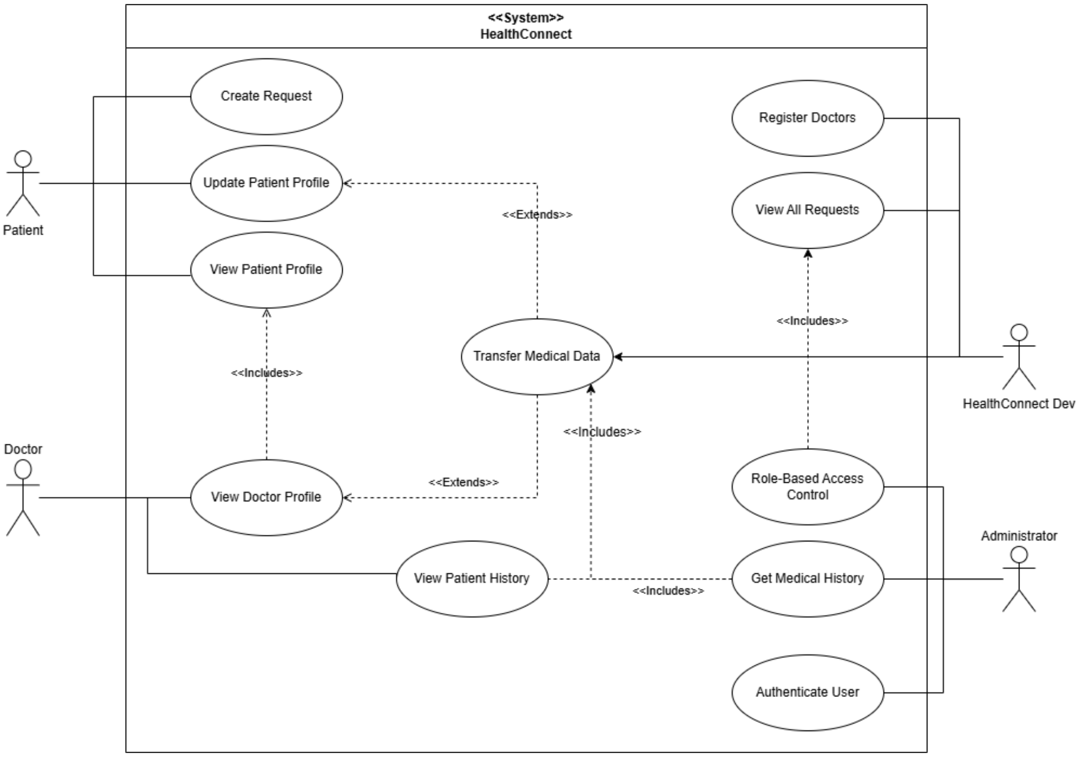
2.2.2.2. Actors: Missing Supervisor actor → needed for escalation flow.

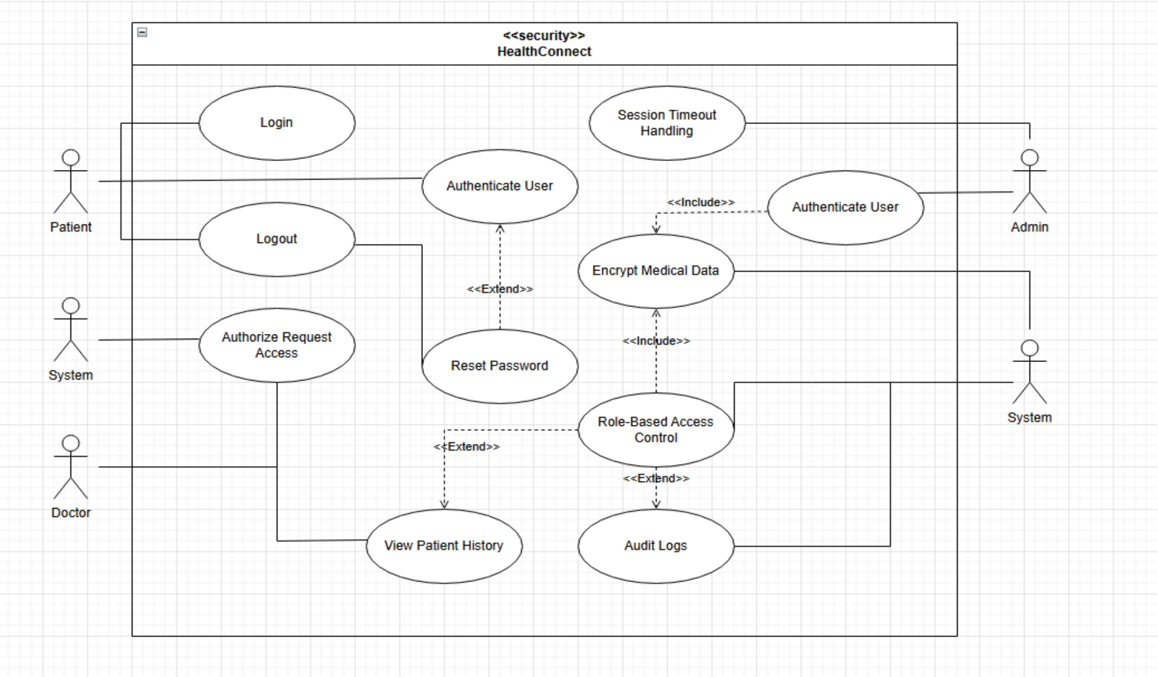
2.2.2.3. Include/Extend Relationships: None defined; critical for reuse (e.g. Login <<include>>, MFA <<extend>>).

2.2.2.4. Generalization: ViewProfile duplicated for multiple actors rather than being abstracted.

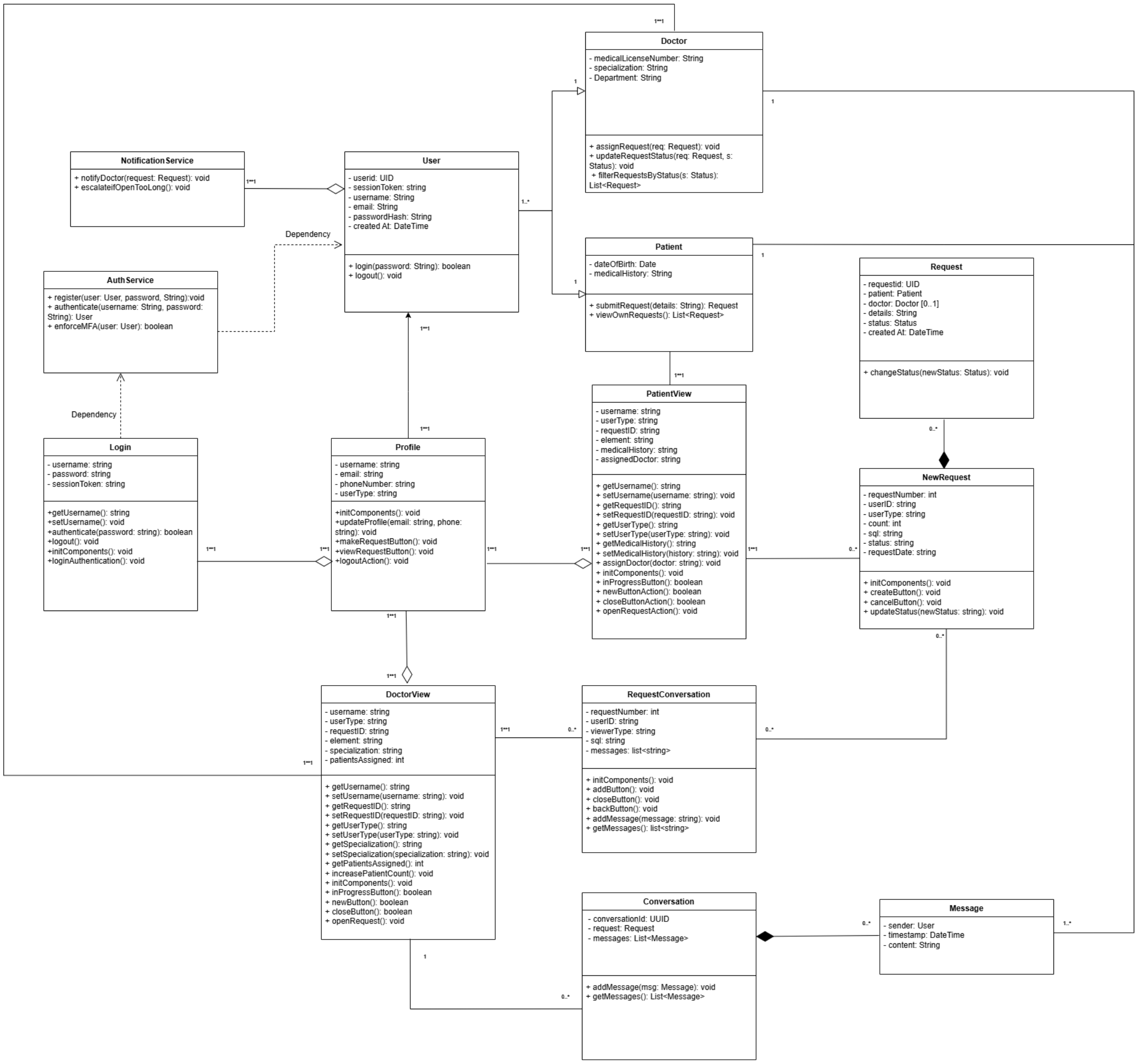
## 2.3. Corrected UML Diagrams:

### 2.3.1. Corrected Use Case Diagram:





### 2.3.1. Corrected Class Diagram:



## 2.4. Comprehensive Evaluation & Assessment Process

### 2.4.1. SQA Process Steps:

2.4.1.1. Requirement Alignment: Map each class/use case to Functional & Non-Functional Requirement

2.4.1.2. UML Compliance: Check against UML 2.5.1 notation (ISO/IEC 19501).

2.4.1.3. Relationship Accuracy: Validate aggregation/composition, remove redundancy.

2.4.1.4. Security & Error Handling: Ensure AuthenticationManager, MFA, input validation, ErrorHandler models exist.

2.4.1.5. Performance & Scalability: Annotate response-time expectations (2 s UI, 1 min notification).

2.4.1.6. Maintainability & Testability: Verify modular design and support for ≥ 80% unit-test coverage.

2.4.2. Checklist (Quantitative & Qualitative Metrics):

2.4.2.1. Graph

| Criterion | Metric | Target |
|---|---|---|
| FR Coverage | % of FRs mapped to model | 80% |
| Domain Classes | Count of classes | ≥ 8 |
| Associations Labeled | % labeled with cardinality | ≥ 90 % |
| Use Case Reuse | # include/extend relationships | ≥ 1 per flow |
| Security Modeling | Authentication & MFA present | ✓ |
| Error Handling Modeling | ErrorHandler class present | ✓ |
| Performance Annotations | Use cases annotated with NFRs | ✓ |

2.4.2.2. Qualitative Checks:

2.4.2.2.1. Clarity and readability of diagrams (font size, layout).

2.4.2.2.2. Consistency of naming conventions.

2.4.2.2.3. Completeness and specificity of association labels and use-case descriptions.

# 3. Process Model Phase

## 3.1. Defect Identification and Metrics

3.1.1. As per the SQA Project Deliverables, we must track defects originating in each phase and those passed to subsequent phases using the following parameters: POD (Phase Originated Defects), PD (Passed Defects), %FE (Percent Filtering Efficiency), RD (Removed Defects), CDR (Cost of Removing Defects), TRC (Total Cost of Defect Removal), and RM (Remaining Defects)

| Phase | POD | PD | %FE | RD | CDR | TRC | RM |
|---|---|---|---|---|---|---|---|
| Requirements | 12 | 8 | 33.3% | 4 | $400 | $1,200 | 8 |
| Design | 8 | 5 | 37.5% | 3 | $600 | $1,800 | 5 |
| Implementation | 5 | 2 | 60.0% | 3 | $900 | $2,700 | 2 |
| Testing | 2 | 0 | 100% | 2 | $1,400 | $4,100 | 0 |

3.1.1.1. Definitions

3.1.1.1.1. POD: Count of defects discovered within the phase.

3.1.1.1.2. PD: Subset of POD that escaped to the next phase.

3.1.1.1.3. %FE = (POD – PD) / POD × 100%.

3.1.1.1.4. RD: POD – PD (defects removed before next phase).

3.1.1.1.5. CDR: RD × average removal cost per defect in that phase (e.g., $100/defect in Requirements).

3.1.1.1.6. TRC: Cumulative sum of CDR from project starts through that phase.

3.1.1.1.7. RM: PD carried into the next phase.

3.1.2. Concrete Example (Requirements → Design):

3.1.2.1. During Requirements inspection, twelve defects were logged.

3.1.2.2. 4 were corrected before handing off to design (RD = 4, CDR = 4 × \$100 = \$400).

3.1.2.3. 8 defects passed into the Design phase (PD = 8, RM = 8).

3.1.2.4. Filtering efficiency was $(12 - 8)/12 = 33.3\%$

## 3.2. Evaluation & Assessment Process

3.2.1. Defect Logging Checklist:

3.2.1.1. Record each defect with: ID, description, severity, origin phase, detection phase, root cause.

3.2.1.2. Classify severity (Critical/Major/Minor) and log estimated removal cost.

3.2.1.3. Link to corresponding requirement or design artifact for traceability.

3.2.2. Defect Metrics Dashboard:

3.2.2.1. Automated reports of POD, PD, %FE, RD, CDR, TRC, RM by phase.

3.2.2.2. Trend analysis of filtering efficiency over iterations.

3.2.3. Quantitative Metrics:

3.2.3.1. Defect Density: POD per KLOC or per requirement.

3.2.3.2. Removal Efficiency: %FE per phase.

3.2.3.3. Cost of Quality: TRC as a percentage of total project budget.

3.2.4. Qualitative Reviews:

3.2.4.1. Root Cause Analysis: Monthly RCA sessions on critical defects.

3.2.4.2. Peer Inspection Effectiveness: Survey reviewers on checklist clarity and coverage.

3.2.5. Continuous Improvement:

3.2.5.1. Update inspection checklists quarterly based on defect trends.

3.2.5.2. Incorporate feedback into training sessions for analysts and designers.

# 4. Implementation Phase

## 4.1. Recommended Design Pattern

### 4.1.1. Model–View–Controller (MVC)

4.1.1.1. Why: Separates UI (Swing/JavaFX), business logic, and data access layers for maintainability.

4.1.1.2. Usage:

4.1.1.2.1. Model: Java domain objects (Patient, Request, Message).

4.1.1.2.2. View: GUI classes (NewJFrame, PatientView, etc.).

4.1.1.2.3. Controller: Service classes managing HTTP endpoints or UI actions, mediating between view inputs and model updates.

### 4.1.2. Data Access Object (DAO) + Repository

4.1.2.1. Why: Encapsulates all JDBC/SQLite operations, abstracts persistence logic.

4.1.2.2. Usage:

4.1.2.2.1. Define PatientDao, DoctorDao, RequestDao with CRUD methods.

4.1.2.2.2. Underlying implementation uses Prepared Statement to guard against SQL injection (REQ 3.1.2.4).

### 4.1.3. Singleton for Database Connection

4.1.3.1. Why: Ensures a single shared Connection instance or connection pool across the app.

4.1.3.2. Usage:

4.1.3.2.1. DatabaseConnection.getInstance() returns a thread-safe connection or Data Source.

### 4.1.4. Observer / Publish Subscribe for Notifications

4.1.4.1. Why: Decouples request submission from doctor notification logic (REQ 3.1.2.5).

4.1.4.2. Usage:

4.1.4.2.1. RequestService publishes an event; NotificationService subscribes and sends push/email within 1 minute.

4.1.5. Factory or Strategy for Request Processing

4.1.5.1. Why: Encapsulates varying behaviors (e.g., HL7 integration vs. simple in-app messaging, REQ 3.3.1 & 3.3.2).

4.1.5.2. Usage:

4.1.5.2.1. MessageHandlerFactory.getHandler(protocol) returns either InAppHandler or Hl7 Handler.

4.1.6. Decorator or Proxy for Security

4.1.6.1. Why: Applies cross-cutting concerns (encryption, RBAC, logging) around core services.

4.1.6.2. Usage:

4.1.6.2.1. Wrap DAO calls in a proxy that enforces role checks (REQ 3.1.1.3) and logs access.

## 4.2. Best Practices in Development

4.2.1. Layered Architecture & SOLID Principles

4.2.1.1. Enforce single-responsibility in controllers/services/DAOs.

4.2.1.2. Open/Closed: Extend notification channels without modifying core code.

4.2.2. Dependency Injection (DI)

4.2.2.1. Use Spring Framework or Google Guice to manage object lifecycles and facilitate testing (mocks for unit tests).

4.2.3. Automated Testing & CI/CD

4.2.3.1. Integrate JUnit/Mockito tests per Test Plan section 14.2.

4.2.3.2. Enforce coverage gates (Statement ≥ 95%, Branch ≥ 80%) with JaCoCo in your CI pipeline.

4.2.4. Secure Coding Practices

4.2.4.1. Parameterize all SQL (prevent injection, REQ 3.1.2.4).

4.2.4.2. Hash & salt passwords (REQ 3.1.1.6).

4.2.4.3. Enforce HTTPS/TLS 1.3 for all endpoints (REQ 3.2.2.1).

4.2.5. Consistent Error Handling & Logging

4.2.5.1. Define global exception handlers returning meaningful HTTP statuses (400, 403, 409).

4.2.5.2. Use SLF4J + Logback for structured logs, including audit trails (REQ 3.3.2).

4.2.6. Configuration Management

4.2.6.1. Externalize database URLs, encryption keys, SMTP settings in application.properties or environment variables.

4.2.6.2. Use a secrets manager for sensitive configs.

4.2.7. Code Reviews & Pair Programming

4.2.7.1. Enforce pull request reviews to catch defects early.

4.2.7.2. Rotate pair programming partners to spread domain knowledge.

4.2.8. Documentation & API Contracts

4.2.8.1. Publish Open API (Swagger) definitions for all REST endpoints.

4.2.8.2. Keep UML diagrams (from Chapter 3) coordinated with implemented code.

## 4.3. Recommended Standards

4.3.1. Coding Standards

4.3.1.1. Java: Google Java Style Guide or Oracle’s JSR-14 conventions.

4.3.1.2. SQL: Lowercase keywords, snake case identifiers, clear naming for tables/columns.

4.3.2. API & Data Exchange

4.3.2.1. RESTful Design: Use proper HTTP verbs, resource-based URLs, and versioning (e.g. /API/v1/requests).

4.3.2.2. HL7 Messaging Standard for EHR integration (REQ 3.3.1).

4.3.3. Security & Compliance

4.3.3.1. HIPAA for patient data privacy.

4.3.3.2. ISO/IEC 27001 for information security management (REQ 3.2.2.2).

4.3.3.3. OWASP Top 10 as baseline for web app security

4.3.4. Testing & Quality

4.3.4.1. JUnit 5 for unit tests, Mockito for mocking.

4.3.4.2. JaCoCo coverage thresholds in CI.

4.3.4.3. OWASP ZAP or Burp Suite scans for security.

4.3.5. Documentation & Archiving

4.3.5.1. Javadoc for all public classes/methods.

4.3.5.2. UML artifacts stored in version control.

4.3.5.3. Audit Logs: Retention ≥ 7 years (REQ 3.3.2).

4.3.6. Data Encryption & Transmission

4.3.6.1. AES-256 for data at rest; TLS 1.3 for data in transit (REQ 3.2.2.1).

4.3.6.2. Password Hashing: BCrypt or Argon2.

# 5. Testing Phase

## 5.1. Test Plan Overview

5.1.1. Scope: Unit and integration testing of all Functional Requirements (3.1.1–3.1.4) and Nonfunctional Requirements (3.2.1–3.3).

5.1.2. Objectives: Detect and remove defects early; verify each requirement.

5.1.3. Tools & Environment:

5.1.3.1. Unit: JUnit (Java), Mockito

5.1.3.2. Integration: Spring Test / Test Containers (SQLite)

5.1.3.3. Coverage: JaCoCo for Statement & Branch metrics

5.1.3.4. Performance: JMeter for load tests

5.1.3.5. Security: OWASP ZAP for injection & auth testing

## 5.2. Test Case Specification

5.2.1. For each functional requirement, we provide one unit and one integration test case (minimum). Coverage targets: ≥ 95% statements, ≥ 80% branches.

| Req. ID | TC ID | Level | Scenario | Preconditions | Steps | Expected Result | Coverage |
|---|---|---|---|---|---|---|---|
| **3.1.1.1** | TC-UPM-1.1 | Unit | Positive register | Empty DB | Call registerPatient(validProfile) | Returns success; DB contains new patient record | Statement, Branch |
| | TC-IPM-1.1 | Integration | End-to-end register | App running; DB empty | HTTP POST /api/patient/register with valid JSON | HTTP 201 Created; Location header; DB row exists | Statement, Branch |
| **3.1.1.2** | TC-UPM-1.2 | Unit | Missing license | In-memory UserService | Call registerDoctor(noL | Throws ValidationException | Statement, Branch |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | icense) | | |
| | TC-IPM-1.2 | Integration | Duplicate email | One doctor registered with email X | HTTP POST /api/doctor/register with same email | HTTP 409 Conflict; error message “Email already in use” | Statement, Branch |
| **3.1.1.3** | TC-UPM-1.3 | Unit | RBAC enforcement | Mocked user with role=“PATIENT” | Call access(“/doctor/dashboard”) | Throws AccessDeniedException | Statement, Branch |
| | TC-IPM-1.3 | Integration | Patient → doctor UI | Patient logged in | Visit /doctor/dashboard via browser | HTTP 403 Forbidden | Statement, Branch |
| **3.1.1.4** | TC-UPM-1.4 | Unit | Check email uniq. | Two User objects with same email in repo | Call checkEmailUnique (email) | Returns false | Statement, Branch |
| | TC-IPM-1.4 | Integration | Duplicate patient | One patient exists | POST /api/patient/register with same email | HTTP 409 Conflict | Statement, Branch |
| **3.1.1.5** | TC-UPM-1.5 | Unit | Update allowed | Patient profile in repo | Call updateProfile({email, name=“New”}) | Repo updated; unique fields unchanged | Statement, Branch |
| | TC-IPM-1.5 | Integration | Attempt ID change | Patient logged in | PUT /api/patient/{id} with changed id | HTTP 400 Bad Request; error “Cannot change identifier” | Statement, Branch |
| **3.1.1.6** | TC-SEC-1.6U | Unit | Password hashing | Raw password “P@ssw0rd” | Call hashAndStore(“P@ssw0rd”) | Stored hash ≠ plain; verify(plain, hash) succeeds | Statement, Branch |
| | TC-SEC-1.6I | Integration | MFA flow | Patient enabled MFA | Login flow with OTP code | Requires OTP step; HTTP 200 on valid OTP | Statement, Branch |
| **3.1.2.1** | TC-REQ-2.1 | Unit | Unregist | No user in | Call | Throws | Statement |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | U | | ered block | session | submitRequest(...) | AuthenticationException | , Branch |
| | TC-REQ-2.1I | Integration | Submit request | Patient logged in | POST /api/requests with valid payload | HTTP 201 Created; JSON with requestId, timestamp | Statement, Branch |
| **3.1.2.2** | TC-REQ-2.2U | Unit | ID/timestamp gen. | In-memory RequestService | Call createRequest(data) | Returns Request{id, timestamp} | Statement, Branch |
| | TC-REQ-2.2I | Integration | DB persistence | App & DB running | Submit request via REST; read back from DB | DB row has non-null id and created_at | Statement, Branch |
| **3.1.2.3** | TC-SEC-2.3U | Unit | Unauthorized view | Two requests by different patients | Call getRequests(patientB) | Returns only B's requests | Statement, Branch |
| | TC-SEC-2.3I | Integration | Direct URL hack | Patient A logged in | GET /api/requests?patientId=B | HTTP 403 Forbidden | Statement, Branch |
| **3.1.2.4** | TC-SEC-2.4U | Unit | SQL injection block | Input "' OR '1'='1" | Call sanitize(requestText) | Returns escaped text; no injection risk | Statement, Branch |
| | TC-SEC-2.4I | Integration | Injection attempt | App & DB | POST /api/requests with malicious payload | HTTP 400 Bad Request; payload sanitized | Statement, Branch |
| **3.1.2.5** | TC-NOTIF-2.5U | Unit | Async notify | Mock RequestService | Call onNewRequest(...) | notificationService.notify() invoked | Statement, Branch |
| | TC-NOTIF-2.5I | Integration | End-to-end notify | App + Notification broker | Submit request; wait 1 min | Doctor receives push/email within ≤ 60 s | Statement, Branch |
| **3.1.3.1** | TC-REQ-3.1U | Unit | Unassigned only | Three requests: two assigned, one open | Call getDashboard(doctor) | Returns only unassigned request | Statement, Branch |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | TC-REQ-3.1I | Integration | Dashboard fetch | Doctor session | GET /api/doctor/requests?status=NEW | JSON list contains only NEW requests | Statement, Branch |
| **3.1.3.2** | TC-REQ-3.2U | Unit | Exclusive assign | One unassigned request | Call openRequest(reqId, doctorId) | Request.status=In Progress; owner=doctorId | Statement, Branch |
| | TC-REQ-3.2I | Integration | Race condition | Two doctors simultaneously open same request | Both POST /api/requests/{id}/open | One succeeds 200; other receives 409 Conflict | Statement, Branch |
| **3.1.3.3** | TC-REQ-3.3U | Unit | Status sequence | Request.status=Open | Call updateStatus(reqId, Closed) | Throws “InvalidTransition” | Statement, Branch |
| | TC-REQ-3.3I | Integration | Valid lifecycle | Doctor owns request | PUT /api/requests/{id}/status within Progress→Closed | HTTP 200; status changed in DB | Statement, Branch |
| **3.1.3.4** | TC-LOG-3.4U | Unit | Log entry created | Mock statusChange | Call logChange(...) | LogRepository saved record with timestamp, doctorId, notes | Statement, Branch |
| | TC-LOG-3.4I | Integration | Persistence of log | DB & log service | Change status via API | logs table has new row with correct data | Statement, Branch |
| **3.1.4.1** | TC-ESC-4.1U | Unit | Check escalation | One request.Open >24 h | Call findStaleRequests() | Returns list containing stale request | Statement, Branch |
| | TC-ESC-4.1I | Integration | Scheduler job | App scheduler running | Advance clock >24h; run job | Supervisor receives reminder email | Statement, Branch |
| **3.1.4.2** | TC-ARC-4.2U | Unit | Archive lock | Request.status=Closed | Call addMessage(reqId, ...) | Throws “ArchivedRequestException” | Statement, Branch |
| | TC-ARC-4.2 | Integratio | Archive | Close | Attempt POST | HTTP 423 Locked | Statement |

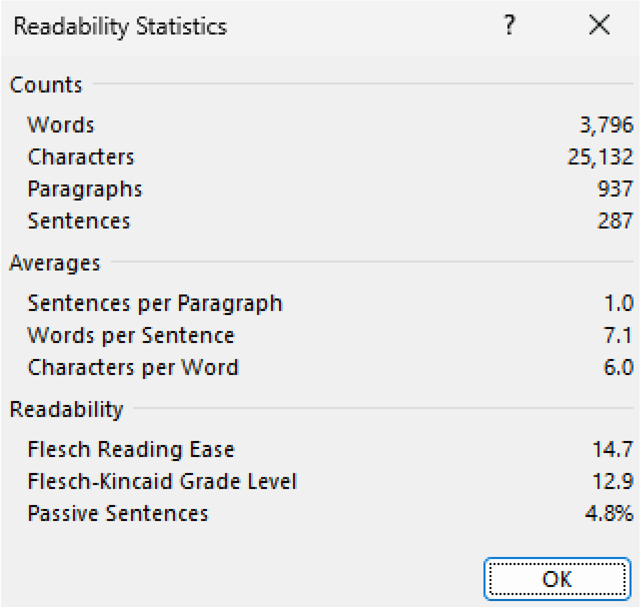
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | I | n | endpoint | request via API | /api/requests/{id}/message | | , Branch |
| **3.1.4.3** | TC-FLT-4.3U | Unit | Filter by status | Request list with mixed statuses | Call filterRequests("Closed") | Returns only Closed requests | Statement, Branch |
| | TC-FLT-4.3I | Integration | UI filter | Doctor dashboard loaded | Click “Closed” filter | UI shows only Closed requests | Statement, Branch |
| **3.2.1.1** | TC-PERF-1.1 | Performance | Response time | 50 concurrent virtual users | HTTP GET /api/dashboard under load | 95th percentile ≤ 2 s | Statement |
| **3.2.1.2** | TC-NOTIF-1.2 | Performance | Notify SLA | Submit request | Measure time from POST to doctor notification | ≤ 60 s | Statement |
| **3.2.2.x** | TC-SEC-2.x | Security | Encryption & Auth | TLS 1.3, AES-256 configured | Inspect network traffic; session timeout | All data encrypted; sessions expire after 15 min; role-based access | Statement |
| **3.2.3.1** | TC-AVAIL-3.1 | Reliability | Uptime check | Monitoring in place | Simulate failure & recovery | Monthly uptime ≥ 99.5% | Statement |
| **3.2.3.2** | TC-BACKUP-3.2 | Reliability | Backup schedule | DB in production | Inspect backup logs over 24 h | Backups every 6 h | Statement |
| **3.2.4.1** | TC-SCALE-4.1 | Scalability | Horizontal scale | Kubernetes cluster | Increase load by 50% | System scales with no errors | Statement |
| **3.2.4.2** | TC-COV-4.2U | Maintainability | Unit coverage | Test suite | Run coverage report | Statement ≥ 80% (for modules) | Statement |
| **3.3.1** | TC-EHR-5.1U | Integration | HL7 integration | Mock EHR endpoint | Send/receive HL7 message | Correct HL7 ACK received | Statement, Branch |
| **3.3.2** | TC-AUD-5.2U | Compliance | Log retention | Old logs >7 years | Query logs older than 7 y | Logs still present; read-only | Statement |

## 5.3. Requirements Traceability Matrix

5.3.1. The Requirements Traceability Matrix (RTM) below links each functional and non-functional requirement to its corresponding project risk, assigned severity, and the test cases defined in the Test Plan. This ensures that every requirement is verifiable and that high-risk items are fully covered.

| Req. ID | Description | Severity | Test Cases |
|---|---|---|---|
| 3.1.1.1 | Patient registration | Medium | TC-UPM-1.1, TC-IPM-1.1 |
| 3.1.1.2 | Doctor registration | Medium | TC-UPM-1.2, TC-IPM-1.2 |
| 3.1.1.3 | Role-based access control | High | TC-UPM-1.3, TC-IPM-1.3 |
| 3.1.1.4 | Prevent duplicate registrations | High | TC-UPM-1.4, TC-IPM-1.4 |
| 3.1.1.5 | Profile updates (no ID change) | Medium | TC-UPM-1.5, TC-IPM-1.5 |
| 3.1.1.6 | Credential security (hashing, MFA) | High | TC-SEC-1.6U, TC-SEC-1.6I |
| 3.1.2.1 | Only registered Patients submit | High | TC-REQ-2.1U, TC-REQ-2.1I |
| 3.1.2.2 | Unique Request ID & timestamp | Low | TC-REQ-2.2U, TC-REQ-2.2I |
| 3.1.2.3 | Patients view only own requests | High | TC-SEC-2.3U, TC-SEC-2.3I |
| 3.1.2.4 | Input validation/sanitization | High | TC-SEC-2.4U, TC-SEC-2.4I |
| 3.1.2.5 | Notify doctor within 1 min | Medium | TC-NOTIF-2.5U, TC-NOTIF-2.5I |
| 3.1.3.1 | Doctors see only unassigned | Medium | TC-REQ-3.1U, TC-REQ-3.1I |
| 3.1.3.2 | Exclusive assignment on open | Medium | TC-REQ-3.2U, TC-REQ-3.2I |
| 3.1.3.3 | Status lifecycle (Open→In Prog→Closed) | Medium | TC-REQ-3.3U, TC-REQ-3.3I |
| 3.1.3.4 | Log every status change | Low | TC-LOG-3.4U, TC-LOG-3.4I |
| 3.1.4.1 | Escalate open > 24 h | Medium | TC-ESC-4.1U, TC-ESC-4.1I |
| 3.1.4.2 | Archive & lock closed | Low | TC-ARC-4.2U, TC-ARC-4.2I |
| 3.1.4.3 | Filter requests by status | Low | TC-FLT-4.3U, TC-FLT-4.3I |
| 3.2.1.1 | UI response ≤ 2 s | Medium | TC-PERF-1.1 |
| 3.2.1.2 | Notification ≤ 1 min | Medium | TC-NOTIF-1.2 |
| 3.2.2.x | Encryption, session expiry, RBAC | High | TC-SEC-2.x |
| 3.2.3.1 | 99.5% uptime | Medium | TC-AVAIL-3.1 |
| 3.2.3.2 | Backups every 6 h | Low | TC-BACKUP-3.2 |
| 3.2.4.1 | Horizontal scaling | Medium | TC-SCALE-4.1 |
| 3.2.4.2 | ≥ 80% unit-test statement coverage | Low | TC-COV-4.2U |
| 3.3.1 | HL7 EHR integration | Medium | TC-EHR-5.1U |
| 3.3.2 | Audit log retention 7 years | Low | TC-AUD-5.2U |

# 6. Readability Statistics



Counts

| | |
|---|---|
| Words | 3,796 |
| Characters | 25,132 |
| Paragraphs | 937 |
| Sentences | 287 |

Averages

| | |
|---|---|
| Sentences per Paragraph | 1.0 |
| Words per Sentence | 7.1 |
| Characters per Word | 6.0 |

Readability

| | |
|---|---|
| Flesch Reading Ease | 14.7 |
| Flesch-Kincaid Grade Level | 12.9 |
| Passive Sentences | 4.8% |

# 7. NASA ARM REPORT

7.1. Due to their incredible length, we uploaded the ARM reports to our GitHub Repository. There you can see the Full ARM report PDFs without it affecting the length and quality of our SQA document. Here are the most important parts.

https://github.com/GioMonci/Software-Quality-Assurance/blob/main/SQA%20Final%20Deliverable/NASA-ARM-TOOL/arm.laplante.io_ARMTool.cgi.pdf

## 7.2. Imperative

```
IMPERATIVE                                                          OCCURRENCE
----------                                                                ----------
ARE APPLICABLE                                                            0
ARE TO                                                                    0
IS REQUIRED TO                                                            0
MUST                                                                      0
RESPONSIBLE FOR                                                           0
SHALL                                                                     31
SHOULD                                                                    0
WILL                                                                      0

                                                                    TOTAL          31
```

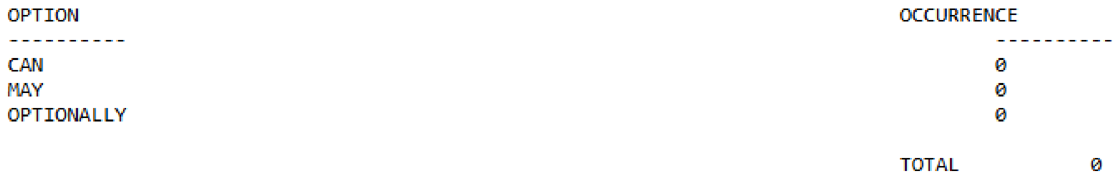
## 7.3. Continuance

```
CONTINUANCE                                                         OCCURRENCE
----------                                                                ----------
:                                                                         0
AND                                                                       11
AS FOLLOWS:                                                               0
BELOW:                                                                    0
FOLLOWING:                                                                0
IN PARTICULAR:                                                            0
LISTED:                                                                   0
SUPPORT:                                                                  0

                                                                    TOTAL          11
```

## 7.4. Directives

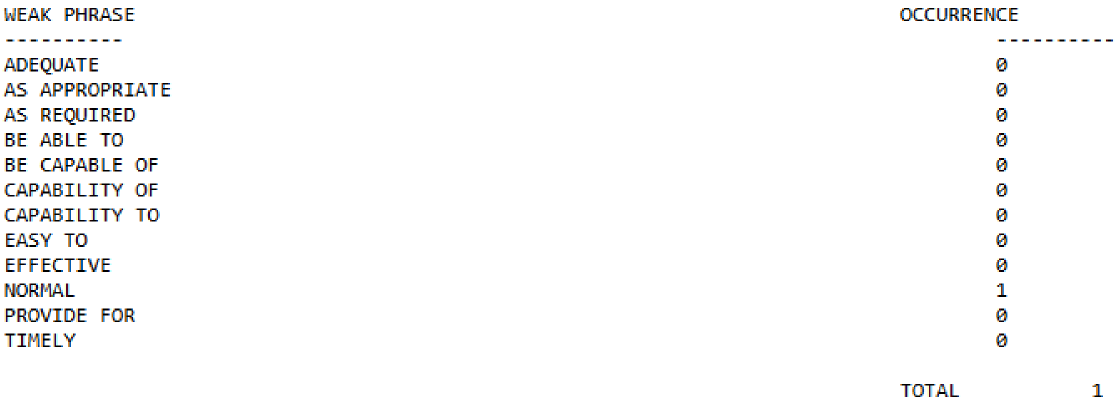
```
DIRECTIVE                                                           OCCURRENCE
----------                                                                ----------
E.G.                                                                      0
FIGURE                                                                    0
FOR EXAMPLE                                                               0
I.E.                                                                      0
NOTE:                                                                     0
TABLE                                                                     0

                                                                    TOTAL          0
```

## 7.5. Option



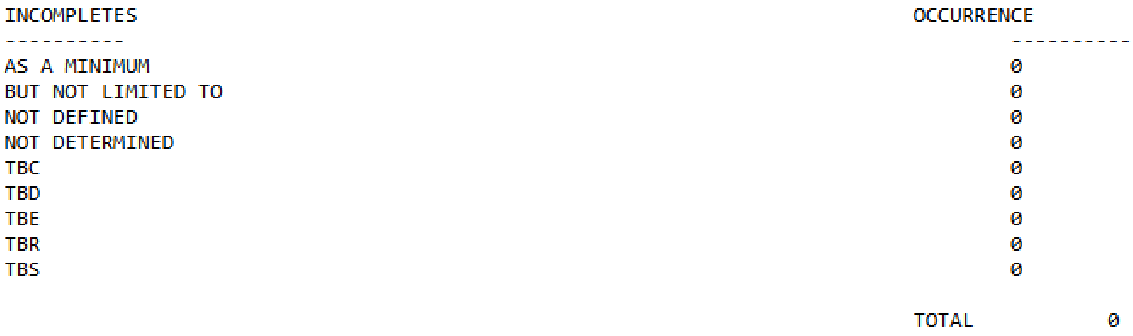
```
OPTION                                                                     OCCURRENCE
----------                                                                         ----------
CAN                                                                                0
MAY                                                                                0
OPTIONALLY                                                                         0

                                                                           TOTAL           0
```

## 7.6. Weak Phrase



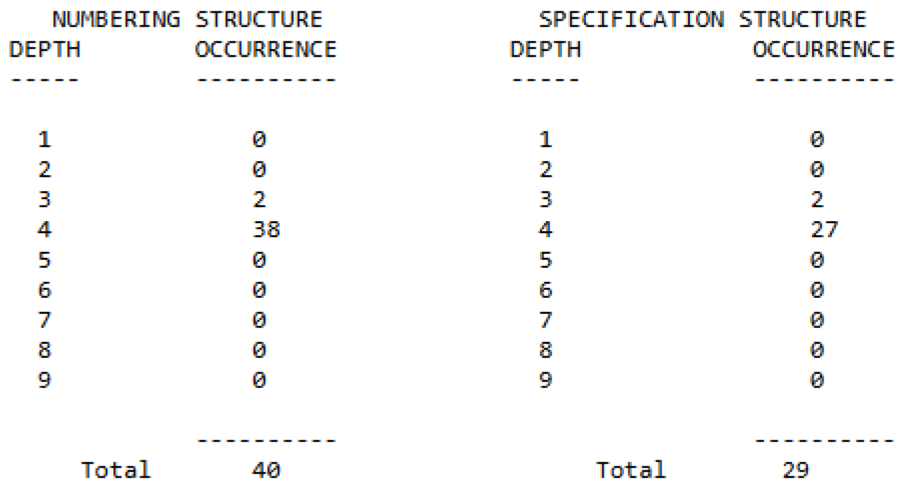
```
WEAK PHRASE                                                                OCCURRENCE
----------                                                                         ----------
ADEQUATE                                                                           0
AS APPROPRIATE                                                                     0
AS REQUIRED                                                                        0
BE ABLE TO                                                                         0
BE CAPABLE OF                                                                      0
CAPABILITY OF                                                                      0
CAPABILITY TO                                                                      0
EASY TO                                                                            0
EFFECTIVE                                                                          0
NORMAL                                                                             1
PROVIDE FOR                                                                        0
TIMELY                                                                             0

                                                                           TOTAL           1
```

## 7.7. Incompletes



```
INCOMPLETES                                                                OCCURRENCE
----------                                                                         ----------
AS A MINIMUM                                                                       0
BUT NOT LIMITED TO                                                                 0
NOT DEFINED                                                                        0
NOT DETERMINED                                                                     0
TBC                                                                                0
TBD                                                                                0
TBE                                                                                0
TBR                                                                                0
TBS                                                                                0

                                                                           TOTAL           0
```

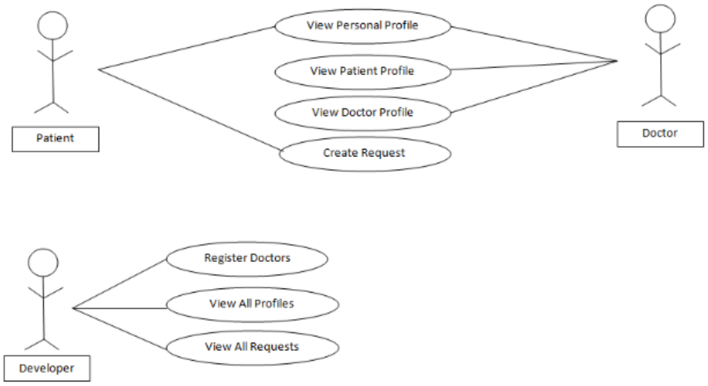
## 7.8. Depth



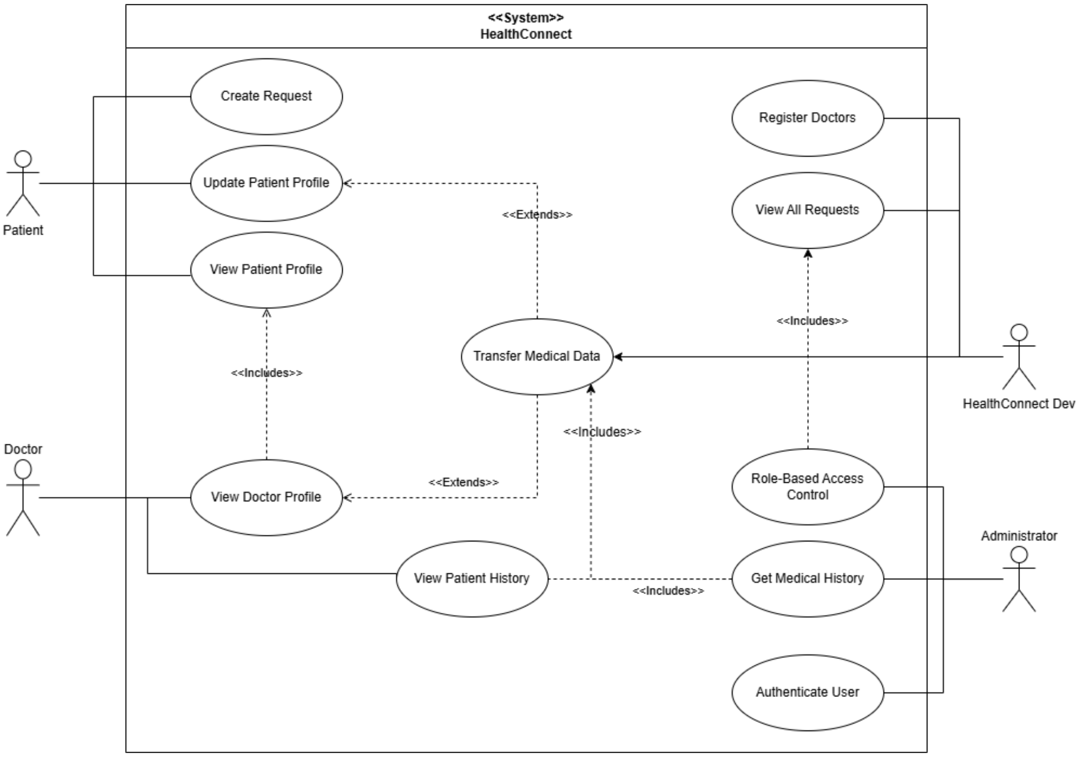
```
   NUMBERING STRUCTURE               SPECIFICATION STRUCTURE
DEPTH        OCCURRENCE            DEPTH            OCCURRENCE
-----        ----------            -----            ----------

  1              0                   1                  0
  2              0                   2                  0
  3              2                   3                  2
  4              38                  4                  27
  5              0                   5                  0
  6              0                   6                  0
  7              0                   7                  0
  8              0                   8                  0
  9              0                   9                  0

             ----------                             ----------
     Total       40                      Total        29
```

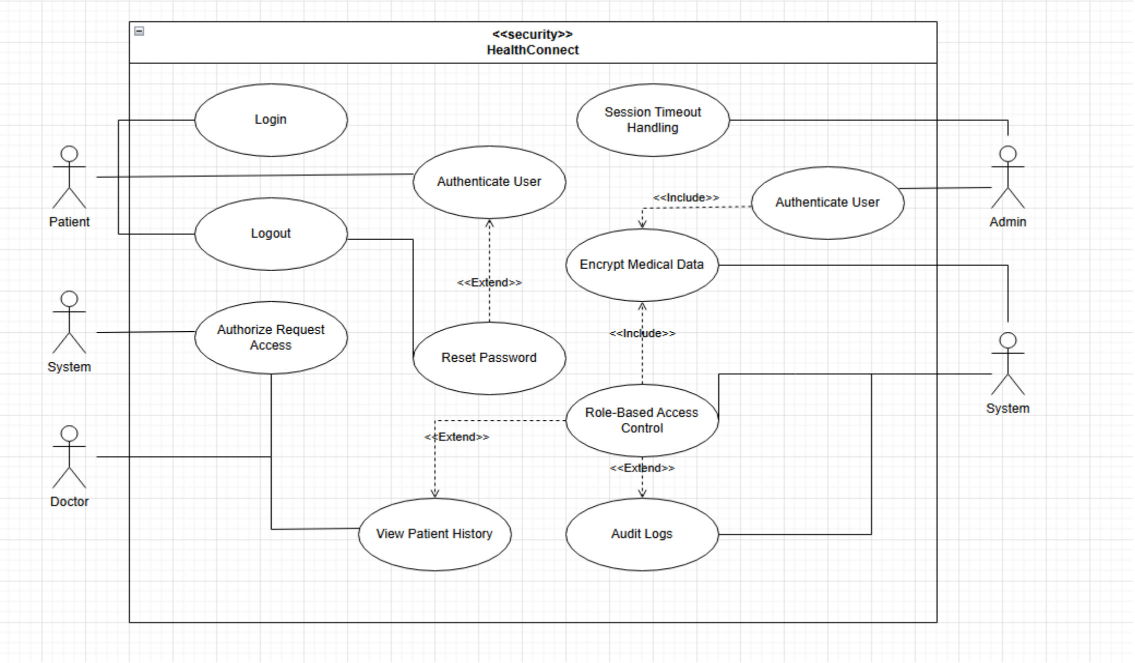
# 8. Appendix A: Analysis Models

## 8.1. Old Use Case Diagram

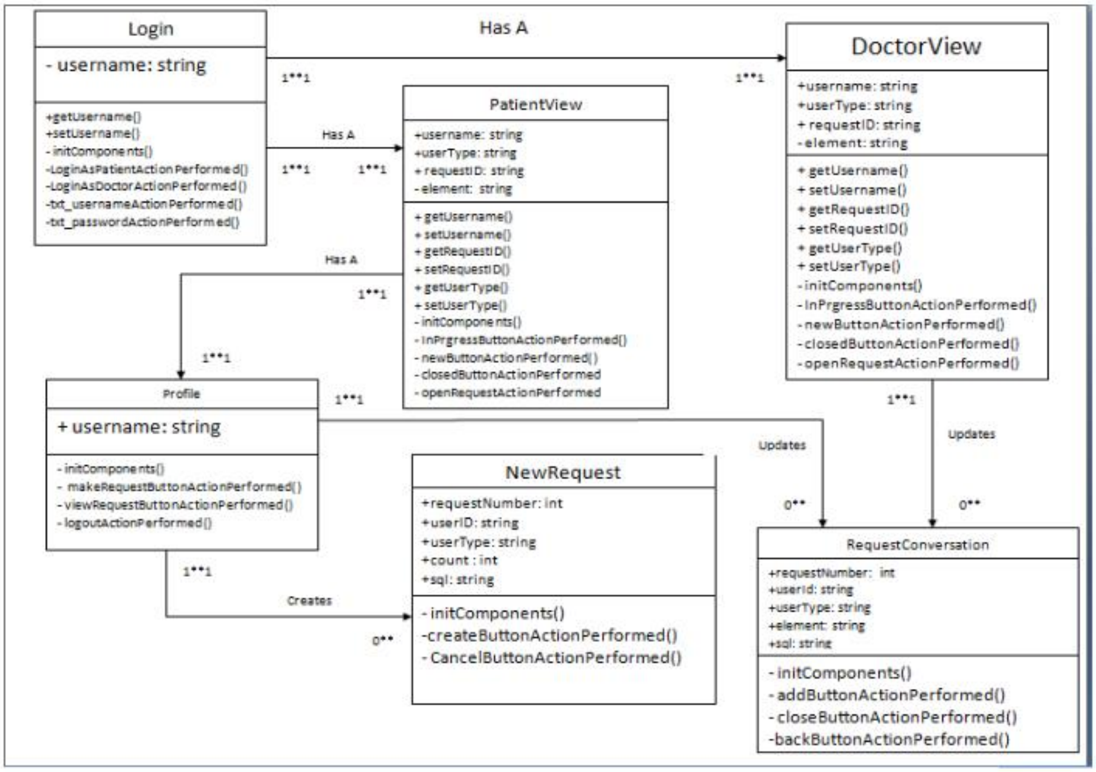


## 8.2. New Use Case Diagram

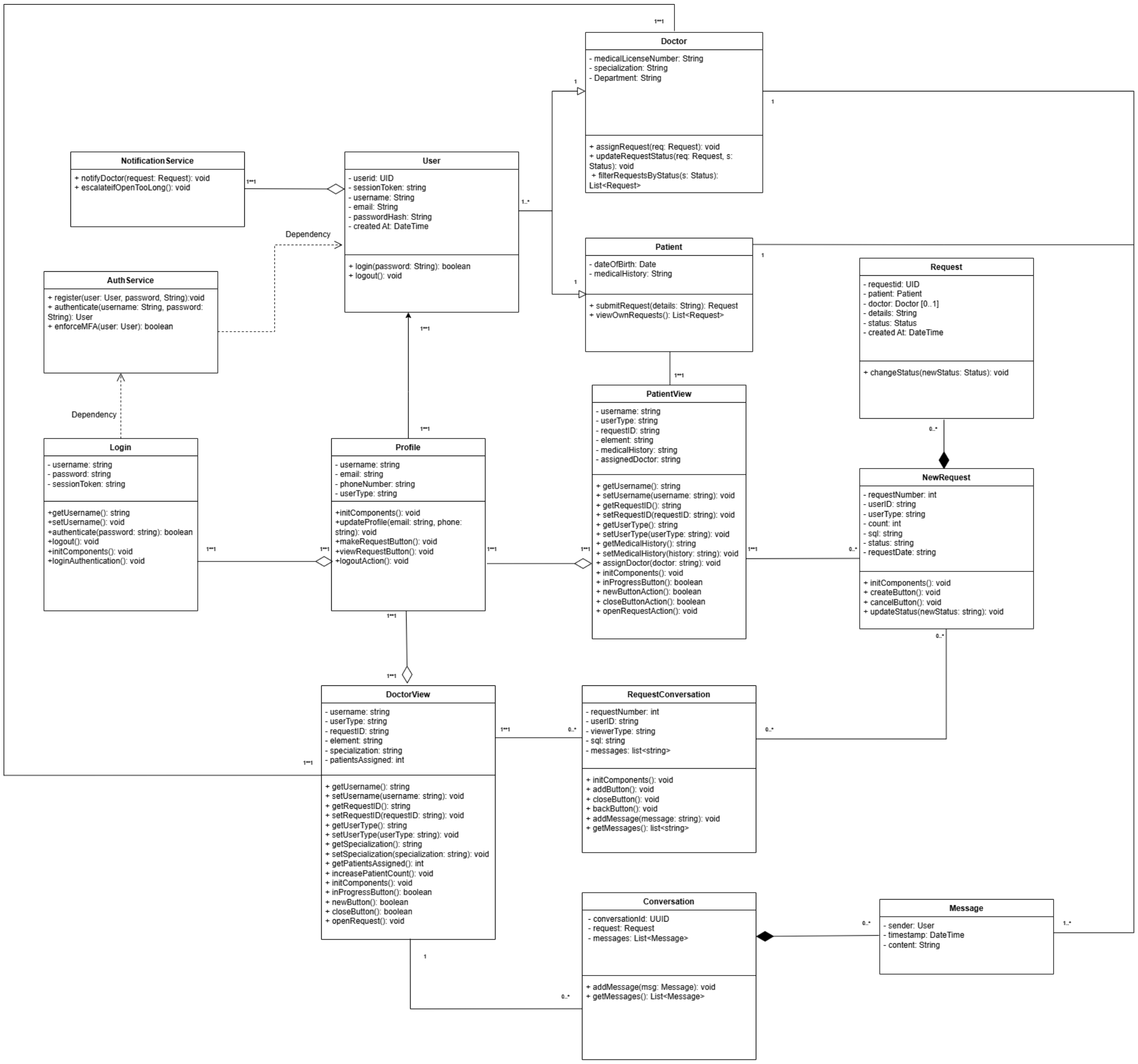



<<security>>
HealthConnect
Login
Session Timeout Handling
Authenticate User
Authenticate User
<<Include>>
Logout
Encrypt Medical Data
<<Extend>>
Authorize Request Access
Reset Password
<<Include>>
Role-Based Access Control
<<Extend>>
<<Extend>>
View Patient History
Audit Logs
Patient
System
Doctor
Admin
System

## 8.3. Old Class Diagram



## 8.4. New Class Diagram



# 9. Appendix C: Cited Sources

[1] IEEE Standard for System, Software, and Hardware Verification and Validation, **IEEE Std 1012-2016**, IEEE, 2016.

[2] International Organization for Standardization, **ISO/IEC 25010:2011, Systems and Software Quality Requirements and Evaluation (SQuaRE)** – **System and Software Quality Models**, Geneva, Switzerland: ISO, 2011.

[3] Object Management Group (OMG), **UML 2.5.1 Specification**, formal-17-12-05, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1

[4] G. Booch, J. Rumbaugh, and I. Jacobson, **The Unified Modeling Language User Guide**, 2nd ed. Boston, MA: Addison-Wesley, 2005.

[5] M. Fowler, **UML Distilled: A Brief Guide to the Standard Object Modeling Language**, 3rd ed. Boston, MA: Addison-Wesley, 2004.

[6] I. Sommerville, **Software Engineering**, 10th ed. Boston, MA: Pearson, 2015.

[7] B. Meyer, **Object-Oriented Software Construction**, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1997.