Software Requirements Specification

for

Medical Encryption System

Version 1.0

Prepared by

Group Name: Team-5

C Srikanth	SE22UCSE062	se22ucse062@mahindrauniversity.edu.in
Dev M Bandhiya	SE22UCSE078	se22ucse078@mahindrauniversity.edu.in
G Aditya Vardhan	SE22UCSE092	se22ucse098@mahindrauniversoty.edu.in
K Sai Bharat Varma	SE22UCSE125	se22ucse125@mahindrauniversoty.edu.in
K Harpith Rao	SE22UCSE141	se22ucse141@mahindrauniversity.edu.in
Punith Chavan	SE22UCSE310	se22ucse310@mahindrauniversoty.edu.in
Harsha B	SE22UCSE321	se22ucse321@mahindrauniversity.edu.in

Instructor: ***Vijay Rao***

Course: **Software Engineering**

Lab Section: ***CS-1 & CS-2***

Teaching Assistant: ***Sravanthi Acchugatla***

Date: **10-3-2025**

Contents ii

Revisions ii

1 Introduction 1

1.1 Document Purpose 1
1.2 Product Scope 1
1.3 Intended Audience and Document Overview 1
1.4 Definitions, Acronyms and Abbreviations 1
1.5 Document Conventions 1
1.6 References and Acknowledgments 2

2 Overall Description 2

2.1 Product Overview 2
2.2 Product Functionality 3
2.3 Design and Implementation Constraints 3
2.4 Assumptions and Dependencies 3

3 Specific Requirements 4

3.1 External Interface Requirements 4
3.2 Functional Requirements 4
3.3 Use Case Model 5

4 Other Non-functional Requirements 6

4.1 Performance Requirements 6
4.2 Safety and Security Requirements 6
4.3 Software Quality Attributes 6

5 Other Requirements 7

Appendix A – Data Dictionary 8

Appendix B - Group Log 9

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft V1.0	C Srikanth, Dev M Bandhiya, G Aditya Vardhan,K Sai Bharat Varma,K Harpith Rao,Punith Chavan, Harsha B	Initial draft of the SRS for the Chikitsa Medical Data Encryption System, detailing system scope, requirements, security policies, and architecture."	10/03/2025

1 Introduction

1.1 Document Purpose

The purpose of this document is to define the software requirements for the Chikitsa Medical Data Encryption System, version 1.0. This system is designed to securely store and manage patient medical records while ensuring role-based access for patients, doctors, and administrators. It outlines the encryption mechanisms, access control policies, and data validation procedures necessary to protect sensitive healthcare information.

This SRS covers the entire system, detailing all major functionalities, including encrypted storage, secure access management, and data retrieval processes. It describes the user roles and their permissions, the security measures in place, and the technologies used for development, such as **AES-GCM encryption, MongoDB, ReactJS, and JWT authentication**. The document serves as a reference for the development team, outlining the scope, objectives, and expected deliverables while ensuring that all stakeholders have a clear understanding of the system's capabilities and constraints.

1.2 Product Scope

*The **Chikitsa Medical Data Encryption System** is a secure platform designed to store and manage patient medical records with end-to-end encryption. It ensures that sensitive healthcare data remains confidential while allowing authorized access for patients, doctors, and administrators. The system leverages **AES-GCM encryption** to safeguard medical information, providing both security and integrity. Its primary objective is to enhance data protection in healthcare environments while enabling seamless and secure access to patient records when needed.*

*This system offers significant benefits, including **enhanced security, role-based access control, and improved data integrity**. By encrypting records before storage, it mitigates risks associated with unauthorized access and cyber threats. The platform also facilitates efficient medical record sharing between healthcare professionals, ensuring quick and reliable access without compromising privacy. Ultimately, **Chikitsa** aims to provide a **scalable, user-friendly, and highly secure** medical data management solution, addressing the growing concerns around healthcare data breaches and compliance with privacy regulations.*

1.3 Intended Audience and Document Overview

Document Overview

This **Software Requirements Specification (SRS)** document for the **Chikitsa Medical Data Encryption System** is structured as follows:

1. Introduction

Provides a high-level overview of the project, detailing its objectives and importance in securing medical records. This section also outlines the intended audience, key definitions, document conventions, and references.

- **1.1 Document Purpose** – Describes the purpose of this document, including the software version and scope of requirements.
- **1.2 Product Scope** – Defines the functionality and security objectives of the system.
- **1.3 Intended Audience and Document Overview** – Identifies the target readers and how they should approach this document.
- **1.4 Definitions, Acronyms, and Abbreviations** – Lists and defines key terms used in the document.
- **1.5 Document Conventions** – Specifies the formatting standards and conventions followed in this document.
- **1.6 References and Acknowledgments** – Includes any supporting documents, research sources, or acknowledgments.

2. Overall Description

Explains the system's high-level functionality, including role-based access control for patients, doctors, and administrators.

- **2.1 Product Overview** – Describes the system's purpose, features, and significance in healthcare data security.
- **2.2 Product Functionality** – Summarizes the system's core features, such as authentication, encrypted record storage, and data access controls.
- **2.3 Design and Implementation Constraints** – Lists any constraints related to technology, compliance, and performance.
- **2.4 Assumptions and Dependencies** – Outlines external dependencies that may affect the system's implementation.

3. Specific Requirements

Details the functional and non-functional requirements that define the system's behavior.

- **3.1 External Interface Requirements** – Describes how users interact with the system through UI/UX and APIs.
- **3.2 Functional Requirements** – Specifies authentication, data encryption, prescription uploads, and administrative oversight.
- **3.3 Use Case Model** – Provides use case diagrams to illustrate system interactions and workflows.

4. Other Non-Functional Requirements

Outlines additional system constraints and operational requirements.

- **4.1 Performance Requirements** – Defines expected system performance, response time, and scalability.
- **4.2 Safety and Security Requirements** – Details security measures such as **AES-GCM encryption** and role-based access controls.
- **4.3 Software Quality Attributes** – Specifies usability, reliability, maintainability, and compliance with healthcare regulations.

5. Other Requirements

Includes any additional system requirements, such as legal, regulatory, and compliance standards that must be met.

Suggested Reading Sequence

For a structured approach:

- **Clients and Professors** should start with **Section 1 (Introduction)** and **Section 2 (Overall Description)** to understand the system's purpose, scope, and security framework.
- **Developers** should focus on **Section 3 (Specific Requirements)** and **Section 4.2 (Safety and Security Requirements)** to ensure proper implementation of encryption and access controls.
- **Testers** should review **Section 4 (Non-Functional Requirements)** and **Section 5 (Other Requirements)** to validate system security, performance, and compliance.

This document provides a comprehensive guide for all stakeholders, ensuring that the **Chikitsa Medical Data Encryption System** meets security, functionality, and performance requirements for efficient and secure medical record management.

1.4 Definitions, Acronyms and Abbreviations

- ***AES-GCM*** *– Advanced Encryption Standard - Galois/Counter Mode (a secure encryption method providing confidentiality and integrity).*
- ***API*** *– Application Programming Interface (a set of functions allowing communication between different software components).*
- ***DBMS*** *– Database Management System (software used for storing, retrieving, and managing data).*
- ***DDoS*** *– Distributed Denial of Service (a type of cyberattack that disrupts system functionality by overwhelming it with traffic).*
- ***DPDP Act*** *– Digital Personal Data Protection Act (India's data protection law that governs the processing and protection of personal data, including healthcare information).*
- ***EHR*** *– Electronic Health Records (digital version of a patient's medical history and records).*
- ***IT Act, 2000*** *– Information Technology Act, 2000 (Indian law that provides legal recognition for electronic transactions and includes provisions for data protection and cybersecurity).*
- ***JWT*** *– JSON Web Token (a secure way to transmit information between parties as a compact, self-contained token).*
- ***MERN*** *– MongoDB, Express.js, ReactJS, Node.js (a full-stack JavaScript framework for web development).*

- ***RBAC*** *– Role-Based Access Control (a security model that restricts system access based on user roles).*
- ***SRS*** *– Software Requirements Specification (a document detailing the software system's functionalities and constraints).*
- ***UI/UX*** *– User Interface / User Experience (design aspects that enhance usability and interaction).*

1.5 Document Conventions

*This document adheres to the **IEEE** formatting requirements to ensure clarity and consistency. The following standards and typographical conventions have been followed throughout the **Software Requirements Specification (SRS)**:*

1. Formatting Conventions

- ***Font Style and Size****: The document uses **Arial, size 11 or 12** for all text except for headings.*
- ***Spacing and Margins****: Text is **single-spaced** with **1-inch margins** on all sides.*
- ***Section and Subsection Titles****: Titles follow the provided template, using bold formatting for **main sections** and subheadings.*

2. Text Formatting

- ***Italics****: Used for comments, notes, and clarifications.*
- ***Bold****: Applied to section headers and key terms for emphasis.*
- ***Monospace****: Used for code snippets, file names, or system commands where applicable.*

3. Naming Conventions

- ***Acronyms and Abbreviations****: Defined in the **Definitions, Acronyms, and Abbreviations** section upon first use.*
- ***File Naming****: Documents and files related to the system follow the format: ProjectName_DocumentType_VersionNumber (e.g., Chikitsa_SRS_V1.0).*

1.6 References and Acknowledgments

The following documents and standards serve as references for the Chikitsa Medical Data Encryption System:

- National Digital Health Mission (NDHM) – Ayushman Bharat Digital Mission (ABDM) https://abdm.gov.in/
- Information Technology (Reasonable Security Practices and Procedures and Sensitive Personal Data or Information) Rules, 2011 https://www.meity.gov.in/
- Electronic Health Record (EHR) Standards for India, 2016 https://www.nhp.gov.in/NHPfiles/ehr_2016.pdf

- Data Security Council of India (DSCI) – Security Best Practices for Healthcare Data https://www.dsci.in/
- Aadhaar Data Security Guidelines – UIDAI https://uidai.gov.in/
- MongoDB Security Best Practices https://www.mongodb.com/docs/manual/security/
- JWT Authentication Best Practices https://auth0.com/blog/

2 Overall Description

2.1 Product Overview

The Medical Data Encryption System is a secure web platform designed to encrypt and manage medical records of all users. It implements role-based access for different users – patients, doctors, *and admins. It is a new product designed to ensure data privacy and security to sensitive medical information.*
This system is not built as a replacement for an existing system but works as an independent solution which can be integrated with existing hospital management software through APIs. Using web technology, encryption standards, and authentication provides us with a secure, user-friendly medical record management system.

2.2 Product Functionality

Our system will perform the following major functions:
1. **User-authentication**: Secure login implemented using JWT token authentication with role-based permission.
2. **Medical Record Encryption:** The patient data is encrypted using AES-GCM scheme before storing it in MongoDB.
3. **Patient Dashboard**: Allows patients to log in, view, and download their medical history. (Prescription, Test reports, Appointment history and more)
4. **Doctor Dashboard**: Enables doctors to login using medical ID, view patient records and add prescriptions.
5. **Admin Dashboard:** Allows admins to perform maintenance and ensure data validity.
6. **Data logs**: Maintain logs of data access by all users for consistency and rollback.

2.3 Design and Implementation Constraints

The design and implementation constraints can be divided into hardware and software groups, as follows:

Hardware Constraints:
1. The system should run efficiently on cloud servers or on existing hospital servers.
2. The encryption/decryption processes should be optimized for performance, but keeping security in mind.

Software Constraints:
1. The system must be built using MERN stack, which includes ReactJS, NodeJS, and MongoDB.
2. The security requirements are the AES-GCM encryption scheme and JWS authentication.
3. The system should use RESTful APIs for external integration.

Development Standards:
1. The software must follow the COMET method for the designing process.
   Reference: Gomaa, H. (2011). “Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures”.
2. The design must be modeled using UML diagrams.

2.4 Assumptions and Dependencies

The following assumptions are taken as valid for the development of this project:

1. All users will have secure login credentials and it will not be shared with unauthorized people.
2. All hospitals will comply with data security, IT & medical regulations and laws.
3. MongoDB is secure for all data storage and it supports the encryption system at rest.
4. Role-based is strictly never violated.
5. Assumes availability of a cloud server on-site to host the application.

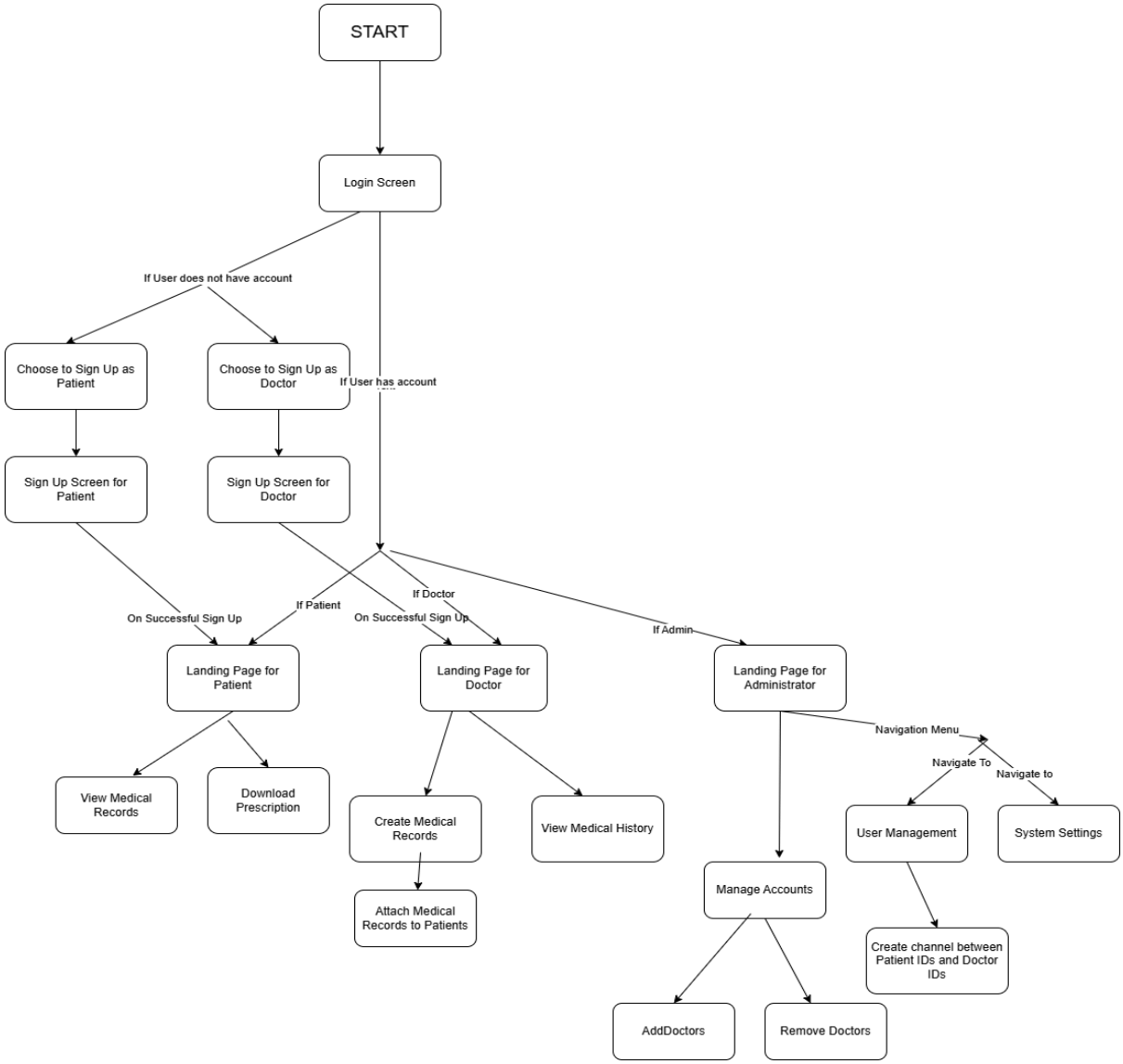
Dependencies are listed as follows:

1. The system relies on ReactJS for frontend development.
2. MongoDB Atlas is used for cloud-based storage.
3. Node.js handles backend logic and API management.
4. APIs for hospital management may be needed for future integration.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces



The system has three main interfaces:

1. Patient Interface
2. Doctor Interface
3. Administrator Interface

The logical characteristics of each of the interfaces between the software and the users are:

1. Patient Interface
    - **Login & Authentication**: Patients can log into the website using their unique ID and password
    - **View Access for Prescriptions**: Patients are authorized to view their encrypted prescriptions. They cannot edit / modify the data.
    - **Option to Download**: Patients can download the encrypted prescriptions if they wish.
    - **Filter Prescriptions:** Patients can filter prescriptions for viewing using criteria like Date Range, Sort by Oldest to Newest, or Status (Active / Expired)
2. Doctor Interface
    - **Login & Authentication**: Doctors can log into the website using their unique ID assigned by the administrator.
    - **Edit Access for Prescriptions**: Doctors are authorized to modify dosage, frequency, and prescriptions.
    - **View Prescription / Medical History:** Doctors can access patients' prescription history.
    - **Filter Prescriptions:** Doctors can filter prescriptions for viewing using criteria like Date Range, Sort by Oldest to Newest, or Status (Active / Expired)
3. Administrator Interface
    - **Login & Authentication**: Doctors can log into the website using their admin ID, in order to prevent unauthorized access
    - **Navigation Menu:** Administrators have access to different sections like User Management and System Settings.
    - **Adding Doctors and Providing Unique IDs:** Administrators can add doctors to the system, and automatically generate unique IDs.
    - **Manage Accounts:** Administrators can disable those accounts of doctors who resign / are no longer authorized, and create and modify accounts.
    - **Manage Database System:** Administrators can access the database and ensure records are properly stored.

3.1.2 Hardware Interfaces

The system has three main interfaces:

1. Patient Interface
2. Doctor Interface
3. Administrator Interface

The logical and physical characteristics of each of the interfaces between the software and hardware products are:

1. Patient Interface
   Supported Devices
    - **Personal Computers & Smartphones**: Access through a web browser

   Hardware Interactions
    - **Keyboard Input:** Patients can enter credentials to login / filter prescriptions
    - **Printers:** Patients can print their prescriptions (if needed)
    - **Display Screen**: Shows prescription details and medical history
2. Doctor Interface
   Supported Devices

- **Hospital Workstations & Authorized Laptops** - Secure terminals within hospitals, extended to clinics.

Hardware Interactions

- **Keyboard Input:** Doctors can enter credentials to login / filter prescriptions
- **Printers:** Doctors can print their prescriptions (if needed)
- **Display Screen**: Shows prescription details and medical history

3. Administrator Interface

Supported Devices

- **Admin Workstations:** Used within secure environments
- **Secure Servers:** Back-end access for database / encryption management

Hardware Interactions

- **Server Console Access:** Admins may interact with backend servers for maintenance and security updates.
- **Keyboard Input:** Admins can enter login credentials / navigate records.

3.1.3 Software Interfaces

The connections between this product and the software components are:

1. Website Interface
   - Patients, doctors, and admins interact with the system through a secure website.\
   - Website sends login attempt, data modification requests to the backend system
2. Encryption / Decryption
   - Requests and Prescriptions are encrypted at the doctor's end, and decrypted at the patient's end
   - Patient data is encrypted using AES-GCM Scheme.
3. Authentication / Authorization
   - Allows user to choose whether they are logging in as Patient / Doctor
   - Role based access ensures that patients can only view, doctors can edit prescriptions, and admins can manage users.

3.2 Functional Requirements

F1: User Authentication

The system shall allow users to securely log in using their credentials. Authentication will be managed using JWT tokens with role-based access control.

F2: Medical Record Encryption

The system shall encrypt all patient medical records using the AES-GCM encryption scheme before storing them in the database.

F3: Patient Dashboard

The system shall allow patients to view their encrypted medical records and download prescriptions, test reports, and appointment history.

F4: Doctor Dashboard

The system shall allow doctors to log in using their unique medical ID, view patient records assigned to them, and add new prescriptions

F5: Admin Dashboard
The system shall allow administrators to manage user accounts and resolve user access issues

3.3 Use Case Model

The Use Case diagram for this system works based on role-based access where:

1. **Patients** can log in and securely view their medical history.
2. **Doctors** can log in, access patient records, and add prescriptions.
3. **Administrators** can manage data validation and oversee system integrity.
4. **The Encryption Service** ensures secure storage and retrieval of patient data using AES-GCM encryption.

3.3.1 Use Case #1

Author – Srikanth C

Purpose - To allow all users to log in and access their respective dashboards securely.

Requirements Traceability -

1. The system must provide authentication for different users.
2. User must be verified using JWT tokens.

Priority - High

Preconditions - User must have a registered account on the portal.

Post conditions - The user is authenticated and redirected to their respective dashboard.

Actors – Patients, Doctors, Administrators.

Extends – None

Flow of Events

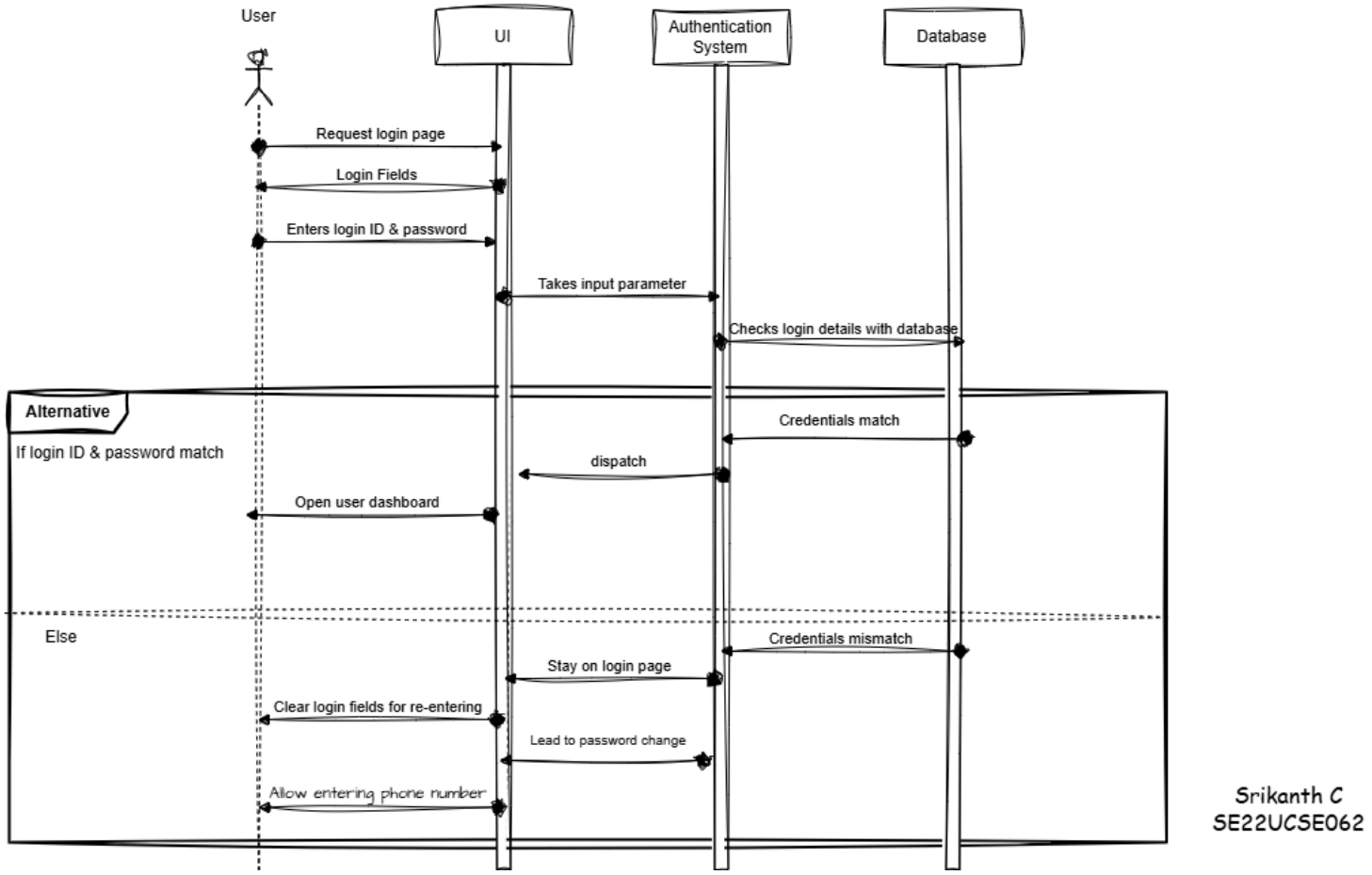
1. Basic Flow -
    - User enters login credentials
    - System validates credentials against stored ones in the database
    - If valid, a JWT token is assigned to the user and he/she is redirected to the dashboard.
2. Alternative Flow -
    1. If the user enters wrong credentials, an error message is shown and the page is reset.
    2. If users forget passwords, they can reset them via OTP verification.
3. Exceptions - System failure.

Includes: None

Notes/Issues - Ensure encryption system is working optimally

Sequence Diagram

Chikitsa
Medical Records Encryption System



3.3.2 Use Case #2

Author – Bharat Varma

Purpose - To allow patients & doctors to view medical records securely

Requirements Traceability:

1. Patients must be able to view their records.
2. Doctors must be able to view patient records they are authorized for.
3. Data must be decrypted only when accessed.

Priority – High

Preconditions - User must be authenticated, Medical record must exist in the database

Postconditions - The requested medical records are displayed to the authorized user.

Actors:

- Patient
- Doctor

Extends: Decrypt Data (U6)

Flow of Events:

Basic Flow:

1. The user selects the "View Medical Records" option.
2. The system checks user authorization.
3. If authorized, the system retrieves and decrypts the medical record.
4. The system displays the records securely.

Alternative Flow:

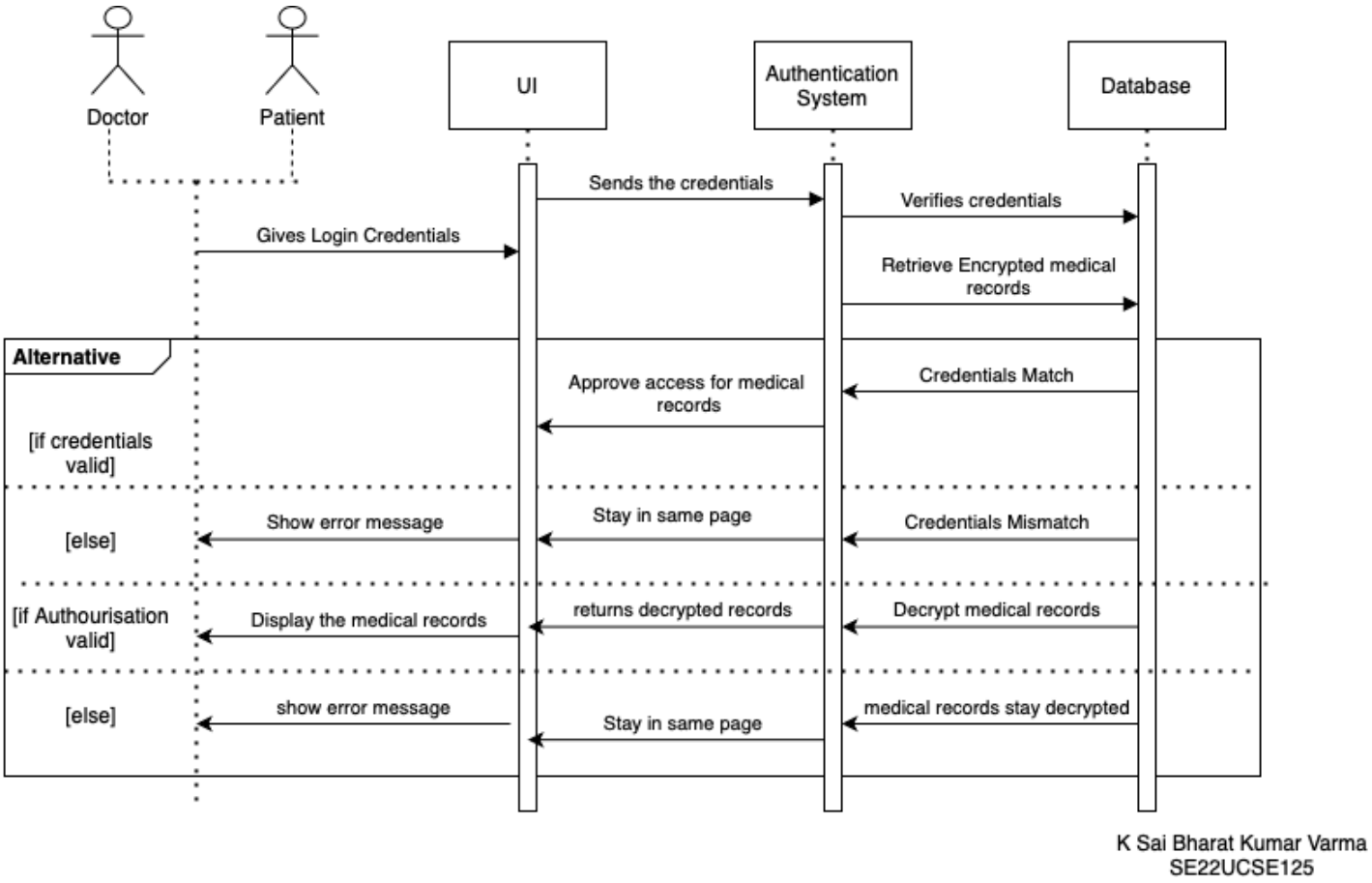
- If no records exist, the system displays a "No records found" message.

Exceptions:

- Unauthorized access attempt: Log the event and display an error message.

Includes: None

Notes/Issues: Ensure access is strictly role-based.



3.3.3 Use Case #3

Author - Dev M. Bandhiya

Purpose - Allow a doctor to update an existing medical records or make a new one

Requirements Traceability

- The system must enable doctors to modify or add medical records securely.
- Data integrity and encryption must be ensured.

Priority: High

Preconditions:

- The doctor must be authenticated and logged into the system.
- The patient's record must exist in the system for updates.

Postconditions:

- The medical record is successfully updated or created.
- Changes are stored securely and reflected in the patient's file.

Actors:

- Doctors

Extends: None

Flow of Events

Basic Flow:

1. The doctor selects a patient from the database.
2. The system retrieves the patient’s existing medical records.
3. The doctor updates the medical details.
4. The system validates and securely stores the changes.
5. A success message is displayed, and the updated record is accessible.

Alternative Flow:

1. The doctor selects a patient from the database.
2. The system retrieves the patient’s existing medical records.
3. The doctor makes a new record.
4. The system validates and securely stores the changes.
5. A success message is displayed, and the updated record is accessible.

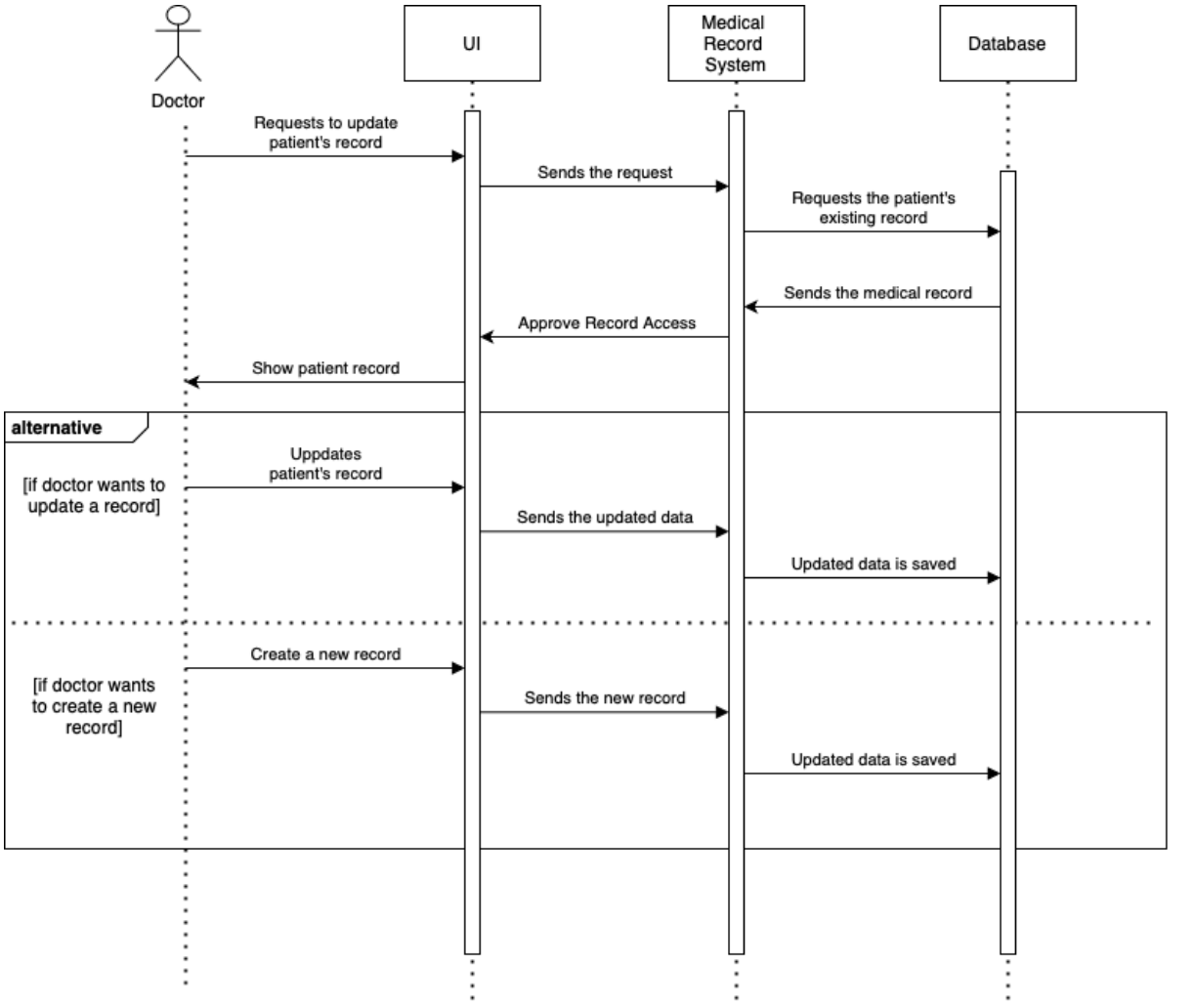
Exceptions:

- System failure or database connectivity issues.
- Unauthorized access attempts.

Includes: None

Notes/Issues:

- Ensure encryption is applied to sensitive medical data.
- Implement an audit log for tracking modifications.



3.3.4 Use Case #4

Author – Gandi Aditya Vardhan

Purpose - To allow an Admin to securely link a Patient and a Doctor so that the doctor can upload prescriptions and the patient can view them.

Requirements Traceability:

- The Admin must be able to link a patient and a doctor using their account IDs.

- The Admin must be able to create a secure channel between a patient and a doctor.
- The channel must ensure that only the assigned doctor can upload prescriptions.
- The patient should only be able to view prescriptions assigned to them.
- The system must encrypt and securely store the prescription data.

Priority: High

Preconditions:

- The Admin must be authenticated.
- Both the Doctor and Patient accounts must exist in the system.

Postconditions:

- A secure channel is created between the doctor and patient.
- The doctor can upload prescriptions securely.
- The patient can view prescriptions assigned to them.

Actors:

1. Admin: Initiates and manages the doctor-patient link.
2. Users: Interact with the channel

Flow of Events:

Basic Flow:

1. The Admin logs into the system.
2. The Admin selects "Create Doctor-Patient Channel."
3. The system prompts for the Patient account ID and the Doctor account ID.
4. The Admin enters the Patient ID and Doctor ID.
5. The system validates both IDs and verifies that a doctor-patient relationship exists or can be assigned.
6. If valid, the system creates a secure channel linking the patient and doctor.
7. A confirmation message is sent to both the doctor and patient.
8. The doctor can now upload prescriptions, and the patient can view them securely.

Alternative Flow:

- If the patient or doctor ID is invalid, the system notifies the admin and rejects the request.
- If the doctor already has an assigned patient list limit, the system prevents further assignments.
- If a channel already exists, the system notifies the admin and prompts for reassignment confirmation.

Exceptions:

- Unauthorized Admin Access: The system denies access and logs the event.
- Database Connection Failure: The system notifies the admin and retries the request.
- Patient/Doctor Deletion: If a linked account is deleted, the system removes the channel and notifies the admin.

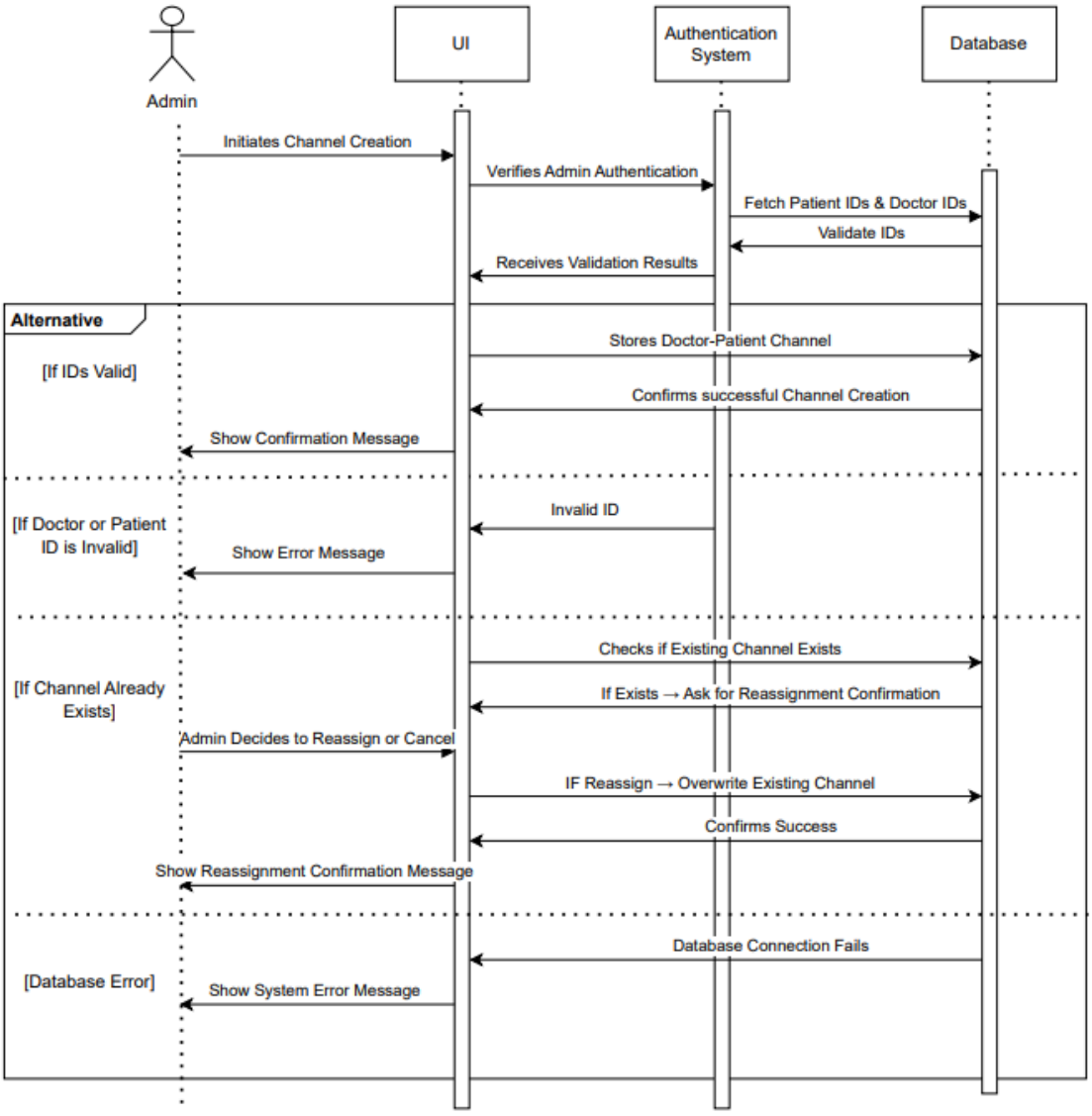
  Includes:

- Validate User Accounts (U5): Ensures the patient and doctor accounts exist.
- Encrypt Prescription Data (U6): Ensures all prescription uploads are encrypted before storage.

Notes/Issues:

- The system should enforce role-based access control (RBAC) to prevent unauthorized channel creation.
- The admin should be able to remove or update existing doctor-patient links when necessary.
- Logs should track who creates, modifies, or removes doctor-patient links for audit purpose.

Name: G Aditya Vardhan
Roll No.: SE22UCSE092



Use Case #5

Author – Punith Chavan

Purpose - To log in as admin and monitor all overall access to the medical records

Requirements Traceability:

1. To appoint a patient to a doctor
2. Admin cannot view or modify medical history
3. Admin can verify if records are uploaded correctly
4. Admin can roll back in case of inconsistencies

Priority: High

Preconditions:

1. Admin must have a valid login credential(given by system owners)

2. Patient and doctor accounts must already exist.

Actors:

1. Admin

Extends: none

Flow of Events

1. Basic Flow

1. Admin logs in to the system.

2. Admin assigns a patient to a doctor

3. System updates the database and notifies the doctor.

4. Admin verifies if medical records are being uploaded correctly.

5. If everything is fine, the process continues normally.

2. Alternate Flow

1. Admin assigns a patient who was previously assigned to another doctor.

2. System confirms reassignment and updates database.

3. Previous doctor is notified of reassignment.

Exceptions:

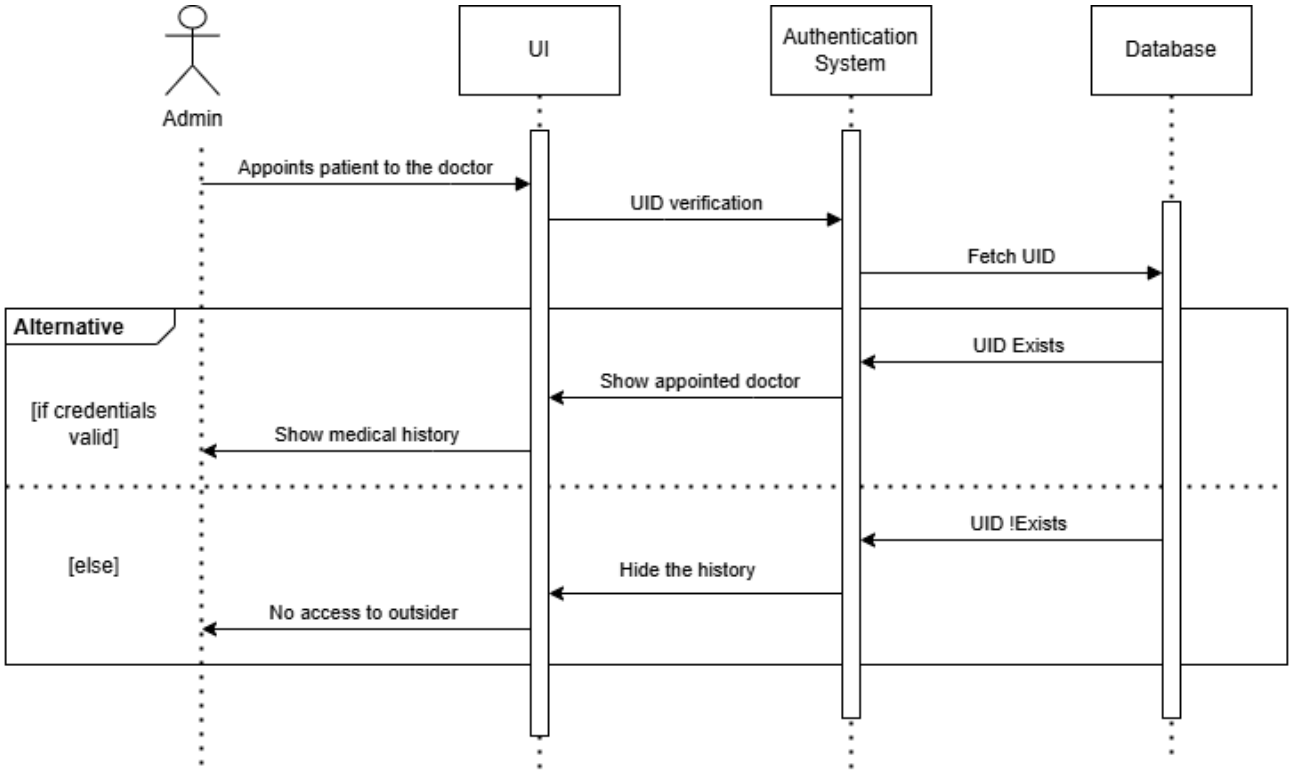
1. Login Failure
2. Patient Already Assigned
3. Database Rollback Failure

Includes:

1. Login and Authentication
2. Database Transaction Management

Notes/Issues:

1. Admin should not have the ability to modify patient records directly.
2. Define strict rollback policies to avoid data corruption



3.3.6 Use Case #6

Author: K. Harpith Rao
Purpose: Download Patient Medical Report

Requirements Traceability:
Patients must be able to download their medical records in a secure format.
The system should generate a PDF or encrypted file.
Priority: High

Preconditions:
The patient must be authenticated.
The requested medical record must exist in the system.

Postconditions:
The patient successfully downloads the requested medical report.

Actors:
Patients

Extends:
Decrypt Data (U6)

Flow of Events:
Basic Flow:

1. The patient selects the "Download Medical Report" option.
2. The system checks authorization.
3. The system retrieves and decrypts the record.
4. The system generates a downloadable file (e.g., PDF, encrypted ZIP).
5. The patient downloads the file.

Alternative Flow:

- If no records exist, a "No records found" message is displayed.

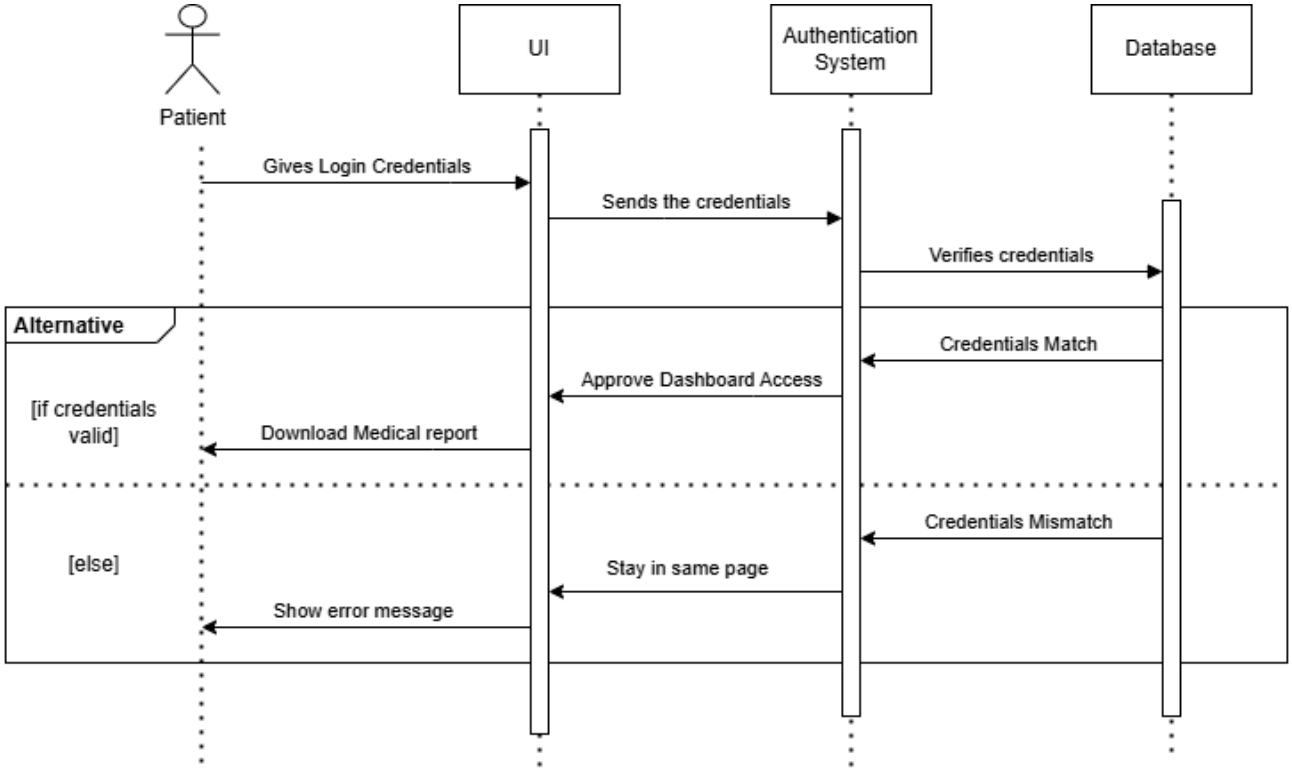
Exceptions:

- Unauthorized access attempt: Log the event and display an error message.
- System failure: Show an error message.

Includes: None

Notes/Issues:

- Ensure encryption for downloaded records.
- Implement download tracking for security.



3.3.7 Use Case #7

Author - Harsha Biruduraju

Purpose - Ensure database consistency by maintaining accurate and validated health data

Requirements Traceability:

- Admins should be able to create or modify existing medical record entries
- System must prevent duplicate, or inconsistent medical data

Priority: High

Preconditions:

- Medical records remain accurate, complete and free from inconsistencies
- Admin must be authenticated and have appropriate permissions

Postconditions:

- Modifications are properly saved without duplication or corruption
- System maintains integrity bt detecting and preventing conflicting / missing medical data

Actors:

- Admin

Extends:

- Data Management

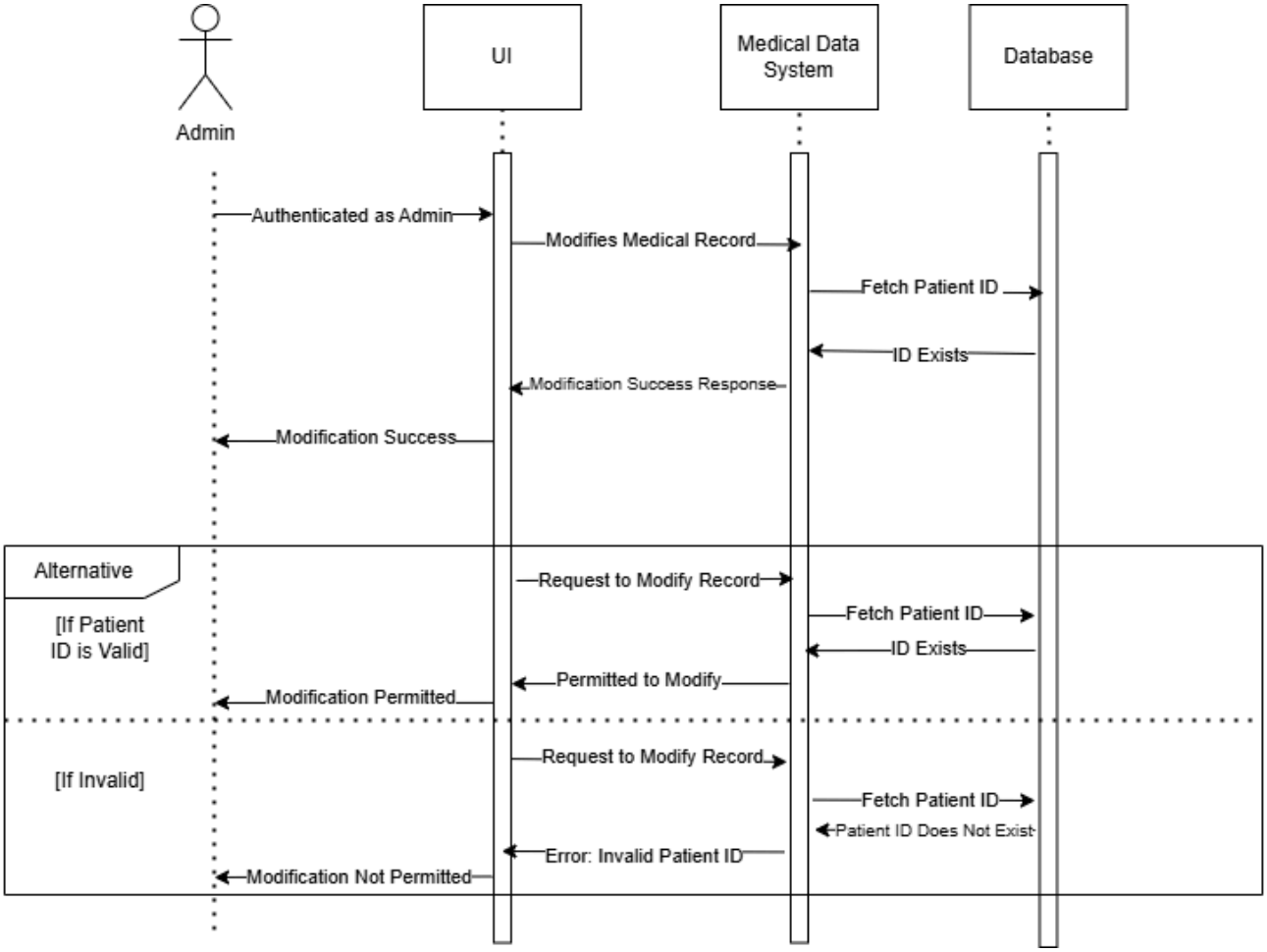
Flow of events:

Basic Flow:

1. Admin searches for a medical record using patient details
2. System verified existence and correctness of record
3. If no errors are found, system allows modifications / additions
4. System updates the database while ensuring data consistency
5. System confirms the successful update of the medical record

Alternative Flow:

1. Admin searches for a medical record
2. System checks for duplicate / inconsistent data
3. If errors are found, system generates an error message
4. Admin is prompted to correct the inconsistencies before proceeding
5. Flow is repeated from Step 1 until the medical record is verified and consistent


Admin
UI
Medical Data System
Database
Authenticated as Admin
Modifies Medical Record
Fetch Patient ID
ID Exists
Modification Success Response
Modification Success
Alternative
[If Patient ID is Valid]
Request to Modify Record
Fetch Patient ID
ID Exists
Permitted to Modify
Modification Permitted
[If Invalid]
Request to Modify Record
Fetch Patient ID
Patient ID Does Not Exist
Error: Invalid Patient ID
Modification Not Permitted

Other Non-functional Requirements

4.1 Performance Requirements

P1: Authentication Response Time
The system shall authenticate users within 2 seconds for their convenience

P2: Minimal Downtime
The system shall maintain minimal downtime and almost always be functioning

P3: Encryption and Decryption Speed
Encryption and decryption of medical records should be limited to 500 milliseconds per request, which can ensure data security

P4: Dashboard Loading Time
The dashboard shall load within 2 seconds when displaying encrypted medical records

P5: Data Logging Performance
There should be a maximum 1-second delay between an event and its log entry

4.2 Safety and Security Requirements

S1: Encryption of Medical Records
Patient records can be encrypted using AES-GCM encryption before being stored in the database

S2: Compliance with Digital Personal Data Protection Act (DPDP Act), 2023
The medical encryption system must comply with the DPDP Ac, which governs the processing and storage of medical data

S3: Data Loss Prevention
The system shall implement automated backups of encrypted patient records from time to time to prevent data loss.

S4: Role-Based Access Control
Patients can only view their own records
Doctors can view and edit assigned patient records
Administrators can manage accounts, and ensure that the system is functioning well

S5: Preventing Unauthorized Access
5 failed login attempts should the lock the user out for 1 hour, to ensure data security

S6: Audit Logs for Authentication
The system must maintain detailed logs of all login attempts. These must include timestamps, users' respective IP addresses, and device details

4.3 Software Quality Attributes

4.3.1 Reliability

The system shall have 99.9% uptime and ensure that the patients records are readily accessible.
This will be achieved by:

- Automated backup every 2 days
- Detecting errors and resolving them by error handling and logging methods.

4.3.2 Maintainability

The system shall follow architecture principles, allowing easy updates and bug fixes.
This will be achieved by:

- Backend divided into microservices, where different modules(authentication, records, logs) can be updated independently
- Well-documented APIs for future extensions

4.3.3 Usability

The system shall ensure ease of use for the users, and shall be made very simple to navigate in the website.
This will be achieved by:

- Intuitive UI made very easy to navigate
- Search and filter options for quick access to patient records.
- Clear error messages which guide users on how to fix issues they are facing

4.3.4 Robustness

The system shall be designed to handle unexpected inputs and failures without crashing the website
This will be achieved by:

- Input validation to check if the input is meaningful and is of the correct format
- Automated recovery to restore system functionality if unexpected shutdowns / crashes occur
- Protection against data corruption by implementing checksums and logging errors

4.3.5 Scalability

The system shall be designed to handle increasing loads efficiently, ensuring smooth performance as the number of users and stored medical records grows.
This will be achieved by:

- Efficiently structured database queries to minimize processing time.
- Flexible architecture that supports future enhancements and feature additions.

Appendix A – Data Dictionary

Name	Type	Description	Possible Values	Operations	Requirements
User Module					
userID	UUID	Unique identifier for a user	Auto-generated	Create, Read, Update, Delete	Must be unique, encrypted
fullName	String	User's full name	Alphanumeric, 3-50 chars	Create, Update, Read	Encrypted
email	String	User email address	Valid email format	Create, Update, Read	Encrypted, required
passwordHash	String	Hashed password	Encrypted string	Create, Update	Minimum 12 chars, hashed
role	Enum	User role	Admin, Doctor, Patient	Assign, Update, Read	Default: Patient
publicKey	String	Asymmetric encryption public key	Base64-encoded string	Store, Read	Must match stored private key
privateKey	String	Encrypted private key	Base64-encoded string	Store, Read	AES-256 Encrypted
Patient Data					
patientID	UUID	Unique identifier for patient	Auto-generated	Create, Read, Update	Linked to User table
dob	Date	Date of Birth	YYYY-MM-DD	Read, Store	Encrypted
medicalHistory	JSON	Encrypted patient history	JSON structure	Read, Update	AES-256 Encrypted
allergies	JSON	List of allergies	JSON Array	Read, Update	Encrypted
emergencyContact	JSON	Emergency contact details	Name, Phone, Relation	Read, Update	Encrypted
Doctor Data					
doctorID	UUID	Unique identifier for doctor	Auto-generated	Create, Read, Update	Linked to User table
specialization	String	Doctor's field of expertise	Cardiology, Neurology, etc.	Read, Update	Encrypted
hospitalID	UUID	Associated hospital/clinic	References Hospital Table	Read, Update	Required

Medical Records					
recordID	UUID	Unique medical record identifier	Auto-generated	Create, Read, Delete	Encrypted, Immutable
patientID	UUID	Associated patient	References patientID	Read, Validate	Must exist in Patient table
doctorID	UUID	Issuing doctor	References doctorID	Read, Validate	Must exist in Doctor table
diagnosis	String	Diagnosis summary	Free text, encrypted	Read, Update	AES-256 Encrypted
prescriptions	JSON	List of prescribed medicines	JSON structure	Read, Update	Encrypted, digitally signed
testResults	JSON	Medical test results	JSON structure	Read, Update	Encrypted
encryptionKey	String	Unique AES-256 key per record	Base64-encoded	Generate, Store, Read	Must be securely stored
Encryption & Security Logs					
logID	UUID	Unique log identifier	Auto-generated	Create, Read	Required
event	String	System event description	Login, Logout, Encryption, etc.	Store, Read	Required
timestamp	DateTime	Event timestamp	ISO 8601 format	Store, Read	Auto-generated
userID	UUID	User responsible for event	References userID	Read, Validate	Required
encryptionStatus	Enum	Encryption status of data	Encrypted, Decrypted, Error	Read, Update	Logged for audits

Appendix B - Group Log

Date	Agenda	Discussion Summary	Action Items	Next Steps
2025-01-29	Project Kickoff for Interactive Apartment Search Application	Defined project scope, objectives, and security goals. Assigned roles.	C Srikanth [SE22UCSE062] - Frontend Development Dev M. Bandhiya [SE22UCSE078] - UI/UX Interface G Aditya Vardhan [SE22UCSE092] - Backend, UI/UX Interface. K Sai Bharat Kumar Varma [SE22UCSE125] -, Frontend Development. K.Harpith.Rao [SE22UCSE141] - Frontend , Backend Development. Punith Chavan [SE22UCSE310] - Frontend Development. Harsha B [SE22UCSE321] - Frontend Development, Deploy.	Understand the workings of similar applications in the industry
2025-02-06	Meeting to create Statement of Work for Apartment Search Application	Named the application (MoveIN) and decided to begin implementation	Bharat - Started with the Statement of Work, and ensured that everyone actively contributed Dev - Formally distributed the workload, and wrote it in the Statement of Work	Implement an approach to create the application, and the timeline
2025-02-08	Whether to Change Projects	Researched about a project idea called ‘Medical Encryption System’ and analysed the pros and cons. Went to professors for further details about the project	Dev - Explained the project idea and why we should pursue it Srikanth - Provided some ideas for implementation, and the significance of Encryption Systems. Aditya - Made us aware of certain difficulties in designing a medical encryption system application.	Resubmit the statement of work for the new project idea. Understand and analyse well-known encryption algorithms
2025-02-10	Implementation Plan and Structure of Project and Pages to Design	Worked on the Statement of Work together and discussed about workload distributions	Punith - Structured the functionalities of login and signup pages, and the idea for separate landing pages for doctors, patients and admins Harsha - Documented what patients, doctors, and admins are authenticated to do	Design the Login and Signup Pages, and specify the requirements in login / signup like Aadhaar Number

2025-02-17	Specify the Navigation for Users in the Application	Extensively discussed about how each of the users (patients, doctors, admins) were going to navigate from Login -> Signup -> Landing Page	Harpith - Designed the Login Page. Upon typing the URL, users are redirected to the login page. Dev - Designed the Figma for the login page, ensuring that all the fields were mentioned in the layout	Structure the Doctor-Patient-Admin Relationships in the Application
2025-02-24	Structuring Doctor-Patient Relationship in the Application	Decided on what the doctors, and patients would be authenticated for in the application. Finalized Encryption Algorithm	Srikanth - Established the way doctors and patients would sign up. (Doctors will be given Unique IDs by the Admin) Bharat - Finalized AES-GCM Encryption Algorithm to be implemented for encryption medical records	Establish what the Admin is authenticated to do in the Application
2025-03-03	Specify the Powers of Admin, and his Influence on Other Users	Discussed about how the Admin-User relationship becomes bureaucratic. Established what he can be authorized to do	Harpith - Established how the admin could manage accounts, modify the records, and remove doctors from the system. Punith - Completed designing the Sign Up Page Aditya - Gave a plan for Admins to provide links between Doctor IDs and Patient IDs.	Design Use Case Diagrams and Sequence Diagrams to gain a structured understanding of the systems interactions and workflows
2025-03-10	Design Use Case Diagrams, Sequence Diagrams. Work on SRS Template and Resolve Issues Faced	Went through the document to check individual contributions, and made suggestions where required	Dev - Ensured everything was properly mentioned in the Use Case Diagrams Harsha - Designed the Graphic Flowchart for how data encryption, decryption, and login in authentications occur within the medical encryption system	Design the Landing Pages for Each User