Sri Lanka Institute of Information Technology

Project Proposal

Innobot Hospital Management System

Module: **Information Technology Project (IT2080)**
Batch ID: **Y2.S2.WE.IT.02**
Group No: **ITP24_B2_W10**

	Name with Initials (Surname first)	**Registration Number**	**Contact Phone Number**	**Email**
1.	S. A. N. Bamunusinghe	IT22515612	0772187484	it22515612@my.sliit.lk
2.	D. M. E Wimalagunasekara	IT22917270	0768952222	it22917270@my.sliit.lk
3.	S A T Nayanapriya	IT22892058	0725203742	it22892058@my.sliit.lk
4.	Obeyesekere A D	IT22332080	0774801500	it22332080@my.sliit.lk
5.	Chamikara M G S	IT22905840	0712905241	it22905840@my.sliit.lk
6.	M F M Farsith	IT22354556	0770494812	it22354556@my.sliit.lk
7.	D S I Gamage	IT22907516	0779705099	it22907516@my.sliit.lk
8.	W A Malsha Haren	IT22307576	0788760703	it22307576@my.sliit.lk

DECLARATION

We declare that this project report or part of it was not a copy of a document done by any organization, university, any other institute, or a previous student project group at SLIIT and was not copied from the Internet or other sources.

ABSTRACT

This final project report documents the development of a comprehensive Practice Management System tailored for Innobot Health, with the primary objective of optimizing hospital operations.

The system incorporates essential modules for inventory management, staff administration, appointment scheduling, patient records, insurance management, and claims processing. Leveraging modern technologies including Spring Boot, React.js, Docker, and MongoDB, the solution ensures scalability, security, and adherence to healthcare standards.

Employing agile methodologies, the development process prioritized iterative requirement gathering, modular design, rigorous testing, and seamless integration. The report encapsulates the challenges encountered, innovative solutions devised, and the resultant benefits accrued.

It also outlines a roadmap for future enhancements and maintenance, ensuring the system remains aligned with evolving healthcare needs.

Validation of the system's success is achieved through compliance audits, performance metrics, and user satisfaction evaluations, guaranteeing its efficacy in meeting the dynamic demands of healthcare providers.

ACKNOWLEDGEMENT

Undertaking the development of a Practice Management System for Innobot Health as part of our university project has been an invaluable opportunity for our team to apply our theoretical knowledge in a practical setting. As members of the development team, we extend our sincere gratitude to all those who have supported and guided us throughout this endeavor.

We would like to express our heartfelt gratitude to **Dr. Samantha Rajapaksha** for his unwavering support and expert guidance throughout the project. His mentorship has been invaluable in shaping our understanding of complex concepts and driving our progress towards achieving our goals.

We extend our deepest appreciation to Natasha Schlinkert and Mario Weerakone, our esteemed clients at Innobot Health, for entrusting us with the task of enhancing hospital operations through our Practice Management System. Their vision and collaboration have been pivotal in guiding our efforts and ensuring the project's success.

Special thanks are also due to, our project evaluators, for dedicating their time and expertise to assess our work and provide constructive feedback. Their insights have been instrumental in refining our approach and identifying areas for improvement.

We are deeply thankful to our parents, colleagues, and other individuals who have provided constant support and encouragement throughout the project. Their guidance and encouragement have been instrumental in keeping us motivated and focused on delivering a successful outcome.

Finally, we extend our heartfelt appreciation to each member of our team who has dedicated their time and effort to bring this project to fruition. Their dedication and teamwork have been the cornerstone of our success, and we are immensely proud of the achievements we have made together.

Table of Contents

Table of Figures

Table of Tables

LIST OF ACRONYMS AND ABBREVIATIONS

Abbreviation	Description
MERN	MongoDB, Express, React Js, Node Js
ER	Entity Relationship Diagram
DBMS	Database Management System
SD	Sequence Diagram
SDLC	Software Development Life Cycle

Table 1

Chapter 1 - Introduction

1.1 Company Background

Innobot health is an US based company with subsidiaries in Colombia, Sri Lanka and India with a collective staff about 40 people. For our university project we chose Innobot as our client.
The company intends on building a Revenue Cycle Management based Practice Management System to automate and ease the tasks of hundreds of thousands of billers in USA.
Innobot Health is a trailblazer in healthcare technology, led by experts with a deep understanding of the industry. Their mission is to simplify healthcare processes, and our university project with them focuses on developing a cutting-edge Practice Management System. Innobot Health’s intelligent automation solutions, utilizing AI and machine learning, optimize revenue cycle management. The company’s collaborative approach tailors’ solutions to specific stakeholder challenges. Our joint effort aims to contribute to advancing healthcare through automation, aligning with Innobot Health’s commitment to excellence.

1.2 Problems and Motivations

1.2.1 Problems

During the requirements gathering phase of the project we discovered that, the existing healthcare information technology infrastructure faces notable challenges, marked by inefficiencies and limitations in managing various aspects of healthcare practices. The lack of a comprehensive system hinders healthcare providers in efficiently managing patient information, scheduling appointments, and handling billing processes, and effectively interacting with insurance systems. Additionally, patients often experience challenges accessing and managing their health records seamlessly. The current scenario lacks a unified solution that promotes interoperability with external healthcare systems and third-party applications. In recognizing these challenges, the development of the 'Innobot' healthcare system is imperative to address these issues and usher in a more streamlined, patient-centric, and interoperable healthcare information technology solution.

1.2.1.1 Current Process and Identified Problems

- **Fragmented Practice Management:**
  Issue: Healthcare providers currently rely on fragmented systems for practice management, leading to inefficiencies in patient record management, appointment scheduling, and billing processes. This fragmentation complicates day-to-day operations and may result in errors and delays.

- **Limited Patient Access and Communication:**
  Issue: Patients face challenges in accessing their personal health records seamlessly and communicating securely with healthcare providers. The current

communication channels may not be user-friendly, limiting patient engagement and satisfaction.

- **Non-Interoperable Systems:**
  Issue: The lack of interoperability with external healthcare systems and third-party applications restricts the seamless exchange of information. This limitation hampers collaborative efforts, data sharing, and the overall efficiency of healthcare operations.

- **Inefficient Insurance Management:**
  Issue: Healthcare providers encounter challenges in efficiently managing insurance-related processes. The current system may lack robust tools for verifying insurance details, processing claims, and ensuring timely reimbursement, leading to financial and administrative burdens.

1.2.2 Motivations

By addressing the identified problems and implementing the 'Innobot' healthcare system, Clients stand to gain several significant benefits,

- **Streamlined Practice Management:**
  Healthcare providers will experience streamlined practice management with integrated tools for patient record management, appointment scheduling, and billing processes. This will lead to increased operational efficiency, reduced errors, and smoother day-to-day operations.

- **Enhanced Patient Engagement:**
  The portal's patient-centric features, including online access to personal health records and secure communication channels, will empower healthcare providers to enhance patient engagement. This increased interaction and accessibility contribute to improved patient satisfaction and adherence to treatment plans.

- **Efficient Insurance Management:**
  Healthcare providers will benefit from robust tools within the 'Innobot' system for insurance management. This includes streamlined processes for verifying insurance details, processing claims, and ensuring timely reimbursement. Improved efficiency in insurance management reduces financial and administrative burdens, allowing healthcare providers to focus more on patient care.

- **Increased Accuracy and Compliance:**
  The automated features of the 'Innobot' system contribute to increased accuracy in patient data management, billing processes, and insurance-related tasks. This

not only minimizes errors but also ensures compliance with healthcare regulations and standards, mitigating potential risks and liabilities

- **Time and Cost Savings:**
  The efficiency gains from the 'Innobot' system translate into time and cost savings for healthcare providers. Automation of repetitive tasks, streamlined workflows, and reduced manual interventions contribute to optimized resource utilization, allowing healthcare professionals to allocate more time to patient care.

- **Enhanced Data Security:**
  The 'Innobot' system prioritizes data security, providing healthcare providers with a secure platform for managing patient information. Improved data security measures safeguard against unauthorized access, ensuring compliance with data protection regulations and instilling trust among patients.

1.3 Aims & Objectives

1.3.1 AIM

This project aims to develop a comprehensive system administration software tailored to the needs of Innobot Health, with a primary focus on enhancing the management processes within hospital operations. This software will streamline and optimize system administration tasks, bolster security measures, and elevate overall efficiency in managing the company's IT infrastructure, particularly within the context of hospital management workflows and procedures.

1.3.2 Objectives

- *Process Optimization:* Identify and streamline management processes within hospital operations, such as patient admissions, discharge procedures, inventory management, and staff scheduling, through the implementation of efficient system administration software.

- *Integration with Hospital Management Systems*: Develop seamless integration capabilities with existing hospital management systems to ensure smooth data flow and interoperability, enabling comprehensive management oversight and analysis.

- *Patient Data Security Enhancement:* Enhance security measures within the software to safeguard sensitive patient data, ensure compliance with healthcare regulations such as HIPAA, and bolster trust in the management of medical records.

- *Operational Efficiency Improvement:* Implement features and functionalities aimed at improving operational efficiency within hospitals, such as automated appointment scheduling, real-time bed management, and optimized resource allocation based on demand forecasting.

- *Decision Support System Implementation:* Integrate decision support functionalities into the software to provide actionable insights and recommendations to hospital administrators, enabling informed decision-making and strategic planning.

- *Customization for Healthcare Environment:* Tailor the system administration software to meet the unique requirements and workflows of healthcare environments, including specialized modules for clinical workflows, diagnostic imaging, and electronic health record management.

- *Performance Monitoring and Reporting:* Develop robust monitoring and reporting capabilities to track key performance indicators within hospital operations, allowing for data-driven analysis, performance optimization, and compliance reporting.
- *Stakeholder Engagement and Feedback Incorporation:* Engage stakeholders, including hospital administrators, healthcare providers, and IT staff, throughout the development process to gather feedback and incorporate user-driven improvements, ensuring that the software meets the evolving needs and expectations of the healthcare industry.

1.4 System Overview

1.4.1 Introduction to the System

The Practice Management System being specified in this Proposal serves as a pivotal component within Innobot's suite of healthcare information technology (HIT) solutions. It is a new, self-contained product aimed at enhancing the efficiency and effectiveness of healthcare providers' administrative tasks, patient management, and overall practice management.

1.4.2 Product Feature

The Practice Management portal for Innobot, encompasses a range of robust features tailored to optimize healthcare information technology solutions. Key functionalities include comprehensive practice management tools for healthcare providers, facilitating seamless patient record management, appointment scheduling, and billing processes. Additionally, the portal offers patient-centric features, empowering individuals with online access to personal health records and secure communication channels. Interoperability is a cornerstone, allowing integration with external healthcare systems and third-party applications.

1.4.3 Operating Environment

The Innobot Practice Management System will operate in a secure and scalable environment tailored to meet the specific needs of healthcare information technology. The software is designed to be platform-independent, supporting a range of hardware configurations. The recommended operating systems include Windows Server 2016 and above.

The web portal seamlessly coexists with common web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge, ensuring accessibility for users across different platforms. It is optimized for compatibility with the latest versions of these browsers to provide an optimal user experience.
To ensure optimal performance, the web portal is recommended to be hosted on servers with sufficient processing power, memory, and storage capacity, and it should be deployed within a network infrastructure that prioritizes data integrity, confidentiality, and availability. Regular updates and patches for the underlying operating system and software components are essential to maintain a secure operating environment for the web portal.

The operating environment emphasizes security and compliance, aligning with industry standards such as **HIPAA** for the protection of sensitive patient information.

1.4.4 Scope Of Work

The project includes the development and deployment of a Healthcare Practice Management System, covering all specified subsystems. It excludes major hardware upgrades or network infrastructure changes outside the immediate system requirements.

1.4.4.1 Project Objective

- Implement a fully functional Healthcare Practice Management System.
- Improve appointment scheduling processes for both staff and patients.
- Enhance inventory management to optimize stock levels and reduce shortages.
- Streamline staff management for efficient onboarding, allocation, and communication.
- Facilitate seamless claim management and integration with external billing services.
- Provide a user-friendly patient management system for accurate record-keeping.
- Implement a notification system for real-time communication and alerts.
- Enable insurance management with features for creating, updating, and reporting.

1.4.4.2 Key Features and Components

- Centralized appointment scheduling
- Staff onboarding and performance tracking
- Inventory tracking with barcode integration
- Claim processing and external system integration
- Patient registration and record management
- Real-time notifications and alerts
- Insurance record creation and management

1.5 System Diagram

We divided our system into 8 main subsystems. These are Appointment Management System, Staff Management System, Claim Management System, Inventory Management System, Procedure/ DX management System, Notification Management System, Patient Management System, Insurance Management System.

Front end is implemented using React.JS. From the frontend, an API request (rest API) will be sent to the backend which is made of Spring Boot. The API validators in the backend will receive the API request and once it's accepted from there, it will go to the authorization middleware, and later it will be sent to the endpoint's service. The service has the function for that endpoint. Finally, to access the DB, a database API request will be sent to the DB (MongoDB)

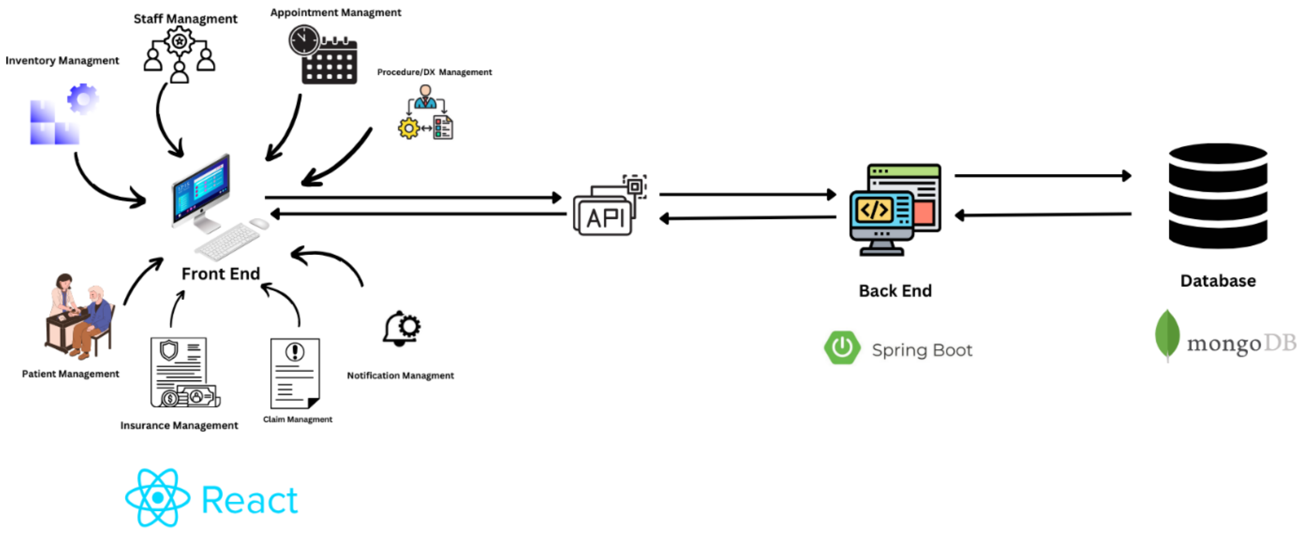


Figure 0-1 – System Design

1.6 Functional Requirements

1.6.1 Inventory Management

This module allows healthcare organizations to efficiently manage their supplies. Users can add, edit, and delete supplier information, including contract details, while monitoring supplier performance through key metrics. Inventory functions include adding, editing, and deleting items, with robust search and filtering options. Seamless integration with barcode scanning technology enhances data accuracy and streamlines tracking. The system supports multi-location inventory management, enabling accurate tracking, movement, and replenishment across various warehouses or facilities.

1) Manage supplier information by adding, editing, and deleting details, upload and oversee contracts, and track supplier performance through key metrics.
2) Perform inventory management functions by adding, editing, and deleting items, and implement search and filtering options for inventory data based on various criteria.

3) The system should integrate seamlessly with barcode scanning technology to enhance accuracy in data entry, streamline the tracking process, and facilitate efficient inventory management.
4) The system should support the management of inventory across multiple locations, allowing for accurate tracking of stock levels, movement, and replenishment in various warehouses or facilities.

1.6.2 Notification Management

Users can customize notification preferences based on their roles, selecting from options like email, SMS, or in-system notifications. The system enables the definition of triggers for key events within the Revenue Cycle Management (RCM) system, ensuring timely notifications for registration, appointment scheduling, claim submission, payment posting, and denial notifications. Notifications are delivered in real-time or near real-time, with a comprehensive audit trail to track all communication within the system.

1) Allow users to set their notification preferences based on their role and responsibilities within the healthcare organization.
2) Options should include email notifications, SMS alerts, in-system notifications.
3) Define triggers for various events within the RCM system, such as registration, appointment scheduling, claim submission, payment posting, or denial notifications.
4) Ensure that notifications are delivered in real-time or near real-time to users to keep them informed of important events and updates.
5) Maintain an audit trail of all notifications sent and received within the system.

1.6.3 Staff Management

This subsystem facilitates the onboarding of new practices with dedicated staff management systems. It includes functionalities for allocating coordinators and staff during initial setup, granting specific access and privileges, allowing staff to control visibility of their profiles, sending messages to all logged users by selected users, viewing staff activity logs, monitoring Key Performance Indicators (KPIs), and managing staff data (creating, deleting, and updating user profiles and privileges).

1) Enrolling new practices with dedicated staff management systems.
2) Allocating coordinators and staff during the initial setup.
3) Granting specific access and privileges to staff members.
4) Staff ability to see/hide their staff base.
5) Sending messages to all logged users by selected users.
6) Viewing staff activity logs.
7) Viewing KPIs (Key performance Indicators)
8) Managing staff data (Creating Users, Deleting Users, Updating Users, Updating User privileges, Read Users)

1.6.4 Appointment Scheduling Management

This comprehensive module manages patient appointments seamlessly. It stores and manages patient information, allowing patients to submit appointment requests. The system verifies patient eligibility based on insurance information and checks healthcare provider availability, displaying available slots or suggesting alternatives. It supports appointment cancellations, sends timely reminders to patients and providers, and integrates with Electronic Health Records (EHR) for optimized coordination.

1) The system should be able to store and manage patient information, including personal details, contact information, and relevant medical history.
2) The system should allow patients to submit appointment requests.
3) The system must verify the eligibility of patients for appointments based on insurance information.
4) The system should check the availability of the requested healthcare provider and display available appointment slots to the patient.
5) If the requested time is unavailable, the system should suggest alternative appointment slots.
6) The system should support the cancellation of appointments by patients and update the records accordingly.
7) The system must send timely reminders to both patients and healthcare providers leading up to the scheduled appointment.

1.6.5 Insurance Management

This module enables users to create, delete, or archive insurance records. It allows updates and modifications to existing insurance records and empowers administrators to define and manage fee schedules for different CPT codes. Users with appropriate permissions can update fee schedules to reflect changes. The system generates insurance records reports and provides analytics features for identifying trends and patterns.

1) System should allow users to create new insurance records.
2) Users should be able to delete or archive insurance records when necessary.
3) Users should be able to update and modify details of existing insurance records.
4) System should enable administrators to define and manage fee schedules for different CPT codes.
5) Users with the appropriate permissions should be able to update fee schedules to reflect changes in allowed amounts.
6) The system should generate insurance records reports and provide analytics features for users to identify trends and patterns.

1.6.6 Claim Management

Role-based access control ensures users have access only to relevant functionalities and data. The system tracks the status of submitted claims, integrates with external systems for seamless data exchange, and allows customizable workflows for claim approval. Comprehensive reports on claim status, processing times, and financial performance aid in monitoring and decision-making. implement role-based access control to ensure that the users only have access to functionalities and data relevant to their roles.

1) Ability to tack status of submitted claims, including approvals, rejections, or pending requests for additional information.

2) Functionality to integrate with external systems, such as insurance databases and billing services, to facilitate seamless data exchange and interoperability.
3) Establish customizable workflows for claim approval, allowing organizations to define approval processes based on their specific policies and procedures.
4) Ability to generate comprehensive reports to claim status, processing times and financial performance.

1.6.7 Procedure and Diagnosis Management

For Procedure Codes, the system allows adding, updating, and deleting codes with associated details. Users can search, display a list, and associate procedure codes with diagnosis codes. Similar functionalities apply to Diagnosis Codes. This ensures accurate documentation and facilitates efficient billing processes.

- **Procedure Codes**

  a. Add new procedure codes with descriptions, including mandatory fields like code, name, and description.
  b. Update existing procedure codes with new descriptions or other relevant information.
  c. Delete procedure codes.
  d. Search for procedure codes by code, name, keyword, or description.
  e. Display a list of all existing procedure codes with their details.
  f. Associate procedure codes with diagnosis codes when applicable.

- **Diagnosis Codes**

  a. Add new diagnosis codes with descriptions, including mandatory fields like code, name, and description.
  b. Update existing diagnosis codes with new descriptions or other relevant information.
  c. Delete diagnosis codes.
  d. Search for diagnosis codes by code, name, keyword, or description.
  e. Display a list of all existing diagnosis codes with their details.
  f. Associate diagnosis codes with procedure codes when applicable.

1.6.8 Patient Management

This module involves the registration of new patients, including details verification. Users can search and access patient information, view and update patient records, manage patient records (including handling duplicates), and efficiently store and organize patient details. The system ensures the centralized and secure management of patient records.

1) Register new patients by entering patient details.
2) Verify insurance details.

3) Search and access patient information.
4) View, update patient records as needed.
5) Manage patient records (Delete duplicated records).
6) Store and manage patient records (Details).

1.7 Non-Functional Requirements

- **Performance**

Response Time: The system should respond to user actions within acceptable time limits, ensuring efficient interaction.

Scalability: The system should be scalable to accommodate an increasing number of users, patients, and data without significant degradation in performance.

Throughput: The system should handle a specified number of transactions or operations per unit of time.

- **Reliability**

Availability: The system should be available for use during specified operational hours, with minimal downtime for maintenance or unexpected issues.

Fault Tolerance: The system should be resilient to hardware or software failures, ensuring continuous operation and data integrity.

Backup and Recovery: Regular backups of data should be performed, and a robust recovery mechanism should be in place to restore the system in case of data loss or system failure.

- **Security**

Data Encryption: All sensitive data, including patient records and financial information, should be encrypted to prevent unauthorized access.

Access Control: The system should implement role-based access control to restrict access to functionalities and patient data based on user roles.

Audit Trails: Detailed logs should be maintained to track user activities within the system for security and compliance purposes.

- **Scalability**

Horizontal Scalability: The system should support the addition of new servers or nodes to the infrastructure to handle increased load.

Vertical Scalability: The system should efficiently utilize increased resources on a single server to accommodate growing requirements.

- **Usability**

User Interface Design: The user interface should be intuitive, with a user-friendly design that minimizes the learning curve for new users.

Accessibility: The system should adhere to accessibility standards, ensuring that users with disabilities can effectively use the software.

- **Compatibility**

Browser Compatibility: The system should be compatible with a range of browsers to ensure a consistent user experience across different platforms.

Integration Compatibility: The system should seamlessly integrate with other healthcare systems, such as electronic health records or laboratory information systems.

- **Maintainability**

Modularity: The system's architecture should be modular, allowing for easy updates, additions, or replacements of components without affecting the entire system.

Documentation: Comprehensive documentation should be provided for system architecture, code, and configurations to facilitate maintenance and updates.

- **Compliance**

Regulatory Compliance: The system should comply with relevant healthcare regulations and standards, such as HIPAA, to ensure the privacy and security of patient information.

1.8 Technical Requirements

- **Hardware:** ￼
  The system should have hardware components that meet the required specifications, such as processing power, memory, and storage capacity.
- **Operating System:**
  The system should be designed to work with a specific operating system, such as Windows, MacOS, or Linux.
- **Database:**
  The system should have a database to store and manage data efficiently.
- **User Interface:**
  The system should have an easy-to-use interface that allows users to interact with the system and perform necessary tasks.
- **Data Security:**
  The system should have appropriate security measures in place to protect data from unauthorized access, such as encryption and user authentication.

- **Networking:**
  The system should be able to connect to a network to enable data transfer and communication with other devices.
- **Performance:**
  The system should be designed to perform efficiently and effectively to meet the needs of the users.
- **Compatibility:**
  The system should be compatible with other hardware and software components that are required for its operation.
- **Scalability:**
  The system should be scalable and able to handle increasing amounts of data and users as the system grows.
- **Maintenance:**
  The system should be easy to maintain, update, and troubleshoot to ensure its continued operation and optimal performance.

1.9 Literature Review

For the literary review we did research some similar Revenue management-based hospital management services like, DOC 990, E-Channeling and composed with our web application which is intergraded based on the Revenue Cycle Management System (RCM), this review will analyze existing literature on:

- Key functionalities of online PMS in healthcare, including patient management, staff management, inventory management, pharmacy management, procedure and diagnosis systems, notification management, and insurance management.
- Benefits of online PMS for RCM, such as improved efficiency, reduced errors, faster claim processing, and increased revenue collection.
- Challenges associated with implementing online PMS, including data security concerns, integration with existing systems, and user adoption.
- Best practices for successful implementation and utilization of online PMS for improved RCM.

- Functions	DrChrono	eClinicalWorks	AthenaOne	NextGen Office	PatientNow
Inventory Management	✓	✓	✓	✗	✗
Staff Management	✗	✓	✗	✗	✓
Appointment Management	✓	✓	✓	✓	✓
Patient Management	✓	✓	✗	✓	✓
Notification Management	✓	✗	✗	✓	✓
Claim Management	✓	✓	✓	✓	✗
Insurance Management	✓	✗	✓	✓	✗
Procedure and Diagnosis Management	✓	✗	✓	✓	✓

1.10 Key Functionalities

- Patient Management: Online PMS can streamline patient registration, appointment scheduling, electronic health record (EHR) integration, and communication with patients through appointment reminders, billing notifications, and online portals.
- Staff Management: PMS can automate scheduling, payroll, and performance tracking for staff, improving efficiency and communication within the practice.
- Inventory Management: Real-time inventory tracking helps optimize stock levels, reduce waste, and ensure timely ordering of medical supplies.
- Pharmacy Management: Online PMS can integrate with pharmacy systems to automate prescription management, medication renewals, and insurance verification.
- Procedure and Diagnosis System: PMS can facilitate accurate coding and billing for procedures and diagnoses, reducing claim denials and improving revenue collection.
- Notification Management: Automated appointment reminders, lab results notifications, and billing updates improve patient communication and satisfaction.
- Insurance Management: Online PMS can verify insurance eligibility, submit claims electronically, and track payment status, streamlining the insurance billing process.

1.11 Methodology

1.11.1 Tools and Technologies

Overview of the System Architecture

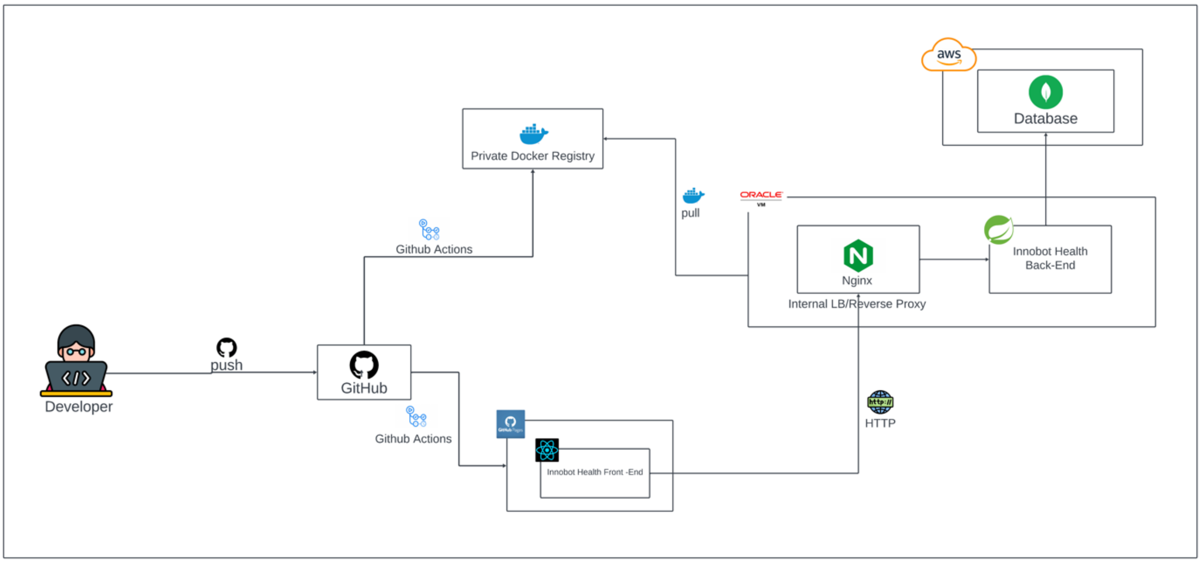


Figure 2 - Overview of the System Architecture

Tools and Services	Technologies
Git/GitHub (version control)	Spring boot (open-source, micro-service-based Java web framework)
Monday.com (work and project management)	React.js (open-source java script library)
Figma, Balsamiq (UI and wireframes)	Docker (containerizing technology)
IntelliJ Idea (integrated development environment)	MongoDB (An open-source document database that provides high performance and scalability)
Nginx (load balancer/ reverse proxy)	
Oracle Could (cloud computing platform)	
AWS (cloud computing platform)	
Postman (API platform)	
Termius (SSH platform)	
MongoDB Compass (GUI for MongoDB)	

1.11.2 Spring Boot

Spring Boot is a powerful framework for building Java-based applications with ease. It provides a simplified configuration and setup process, allowing developers to quickly create production-ready applications. With micro service support for auto-configuration, embedded servers, and dependency management, Spring Boot simplifies the development process, enabling developers to focus more on writing business logic rather than boilerplate code. Its convention-over-configuration approach and extensive ecosystem of plugins and starters make it a popular choice for building microservices, web applications, and APIs.

Advantages:

a) **Simplified Configuration**
Spring Boot provides sensible defaults and auto-configuration, reducing the need for manual setup and configuration.

b) **Rapid Development**
Its streamlined development process allows developers to quickly prototype and build applications, saving time and effort.

c) **Embedded Servers**
Spring Boot includes embedded servers like Tomcat, Jetty, or Undertow, simplifying deployment and eliminating the need for external server setup.

d) **Dependency Management**
It manages dependencies effectively, reducing version conflicts and ensuring compatibility between libraries.

Disadvantages:

a) **Opinionated**
While its conventions can speed up development, they may not always align with specific project requirements, leading to limitations or workarounds.

b) **Learning Curve**
Beginners may find Spring Boot's extensive features and configurations overwhelming, requiring time to grasp its concepts fully.

c) **Performance Overhead**
While Spring Boot offers convenience, its auto-configuration and additional layers may introduce some performance overhead compared to lightweight frameworks.

Why did we choose spring boot?

a) **Security**
Spring Boot provides robust security features, essential for handling sensitive health data and complying with HIPAA regulations.

b) **Scalability**
Its support for microservices architecture enables scalability, allowing the system to handle increasing loads and data volumes effectively.

c) **Integration**
With its extensive ecosystem and support for RESTful APIs, Spring Boot facilitates seamless integration with other healthcare systems and third-party services.

d) **Reliability**
Spring Boot's mature ecosystem, community support, and proven track record make it a reliable choice for building mission-critical applications like RCM systems.

e) **Productivity**
By reducing boilerplate code and simplifying configuration, Spring Boot enables developers to focus more on implementing business logic and healthcare-specific functionalities.

1.11.3 React.js

React.js is a JavaScript library for building user interfaces, particularly for single-page applications. Developed by Facebook, it focuses on creating reusable UI components that efficiently update and render when the data changes. React's virtual DOM allows for fast and efficient updates to the user interface by minimizing direct manipulation of the actual DOM. With its declarative and component-based architecture, React.js simplifies the process of building interactive and dynamic web applications, making it a popular choice for front-end development.

Advantages:

a) **Reusable Components**
Enhance code reusability and maintainability.

b) **Declarative Syntax**
Simplifies UI development by expressing how the UI should look at any given time.

c) **Virtual DOM**
Boosts performance by reducing actual DOM manipulation and enhancing rendering speed.

d) **Component-Based Architecture**
Facilitates modular development and easier code organization.

e) **Strong Ecosystem**
Access to a vast ecosystem of libraries, tools, and community support.

Disadvantages:

a) **Learning Curve**
Beginners may find its concepts, such as JSX and component lifecycle, challenging initially.

b) **Tooling Complexity**
Setting up a React project with additional tools like Babel and Webpack can be complex.

c) **SEO Challenges**
Single-page applications built with React may face SEO challenges due to initial server-side rendering requirements.

Why we choose React.js?

a) **Enhanced User Experience**
React's efficient rendering and dynamic UI capabilities improve user interaction, vital for a smooth RCM system.

b) **Scalability**
Component-based architecture allows for easy scaling and maintenance of the application as the RCM system grows.

c) **Community Support**
A large and active community provides resources, best practices, and support, ensuring the success of the project.

d) **Integration**
React seamlessly integrates with other JavaScript libraries and frameworks, facilitating integration with existing healthcare systems and APIs.

1.11.4 Docker

Docker is a containerization platform that simplifies the process of building, deploying, and managing applications in isolated environments called containers. With Docker, applications and their dependencies can be packaged into a single unit, ensuring consistency across different environments.

Advantages:

a) **Portability:** Docker containers can run on any platform that supports Docker, providing consistent behavior across development, testing, and production environments.

b) **Isolation:** Containers isolate applications and their dependencies, preventing conflicts and ensuring reliability.

c) **Efficiency:** Docker's lightweight containers consume minimal resources and start quickly, improving efficiency and scalability.

d) **Consistency:** Docker eliminates the "it works on my machine" problem by ensuring that applications run consistently in any environment.

e) **Scalability:** Docker's container-based architecture allows applications to scale up or down easily, adapting to changing demands.

f) **DevOps Integration:** Docker integrates seamlessly with DevOps practices, enabling continuous integration, delivery, and deployment pipelines.

Disadvantages:

a) **Complexity:** Docker introduces additional complexity, especially for teams new to containerization, requiring time to learn and adapt to new concepts and tools.

b) **Orchestration Overhead:** Managing containerized applications at scale may require additional tools and overhead for orchestration, such as Kubernetes or Docker Swarm.

c) **Security Concerns:** Improperly configured Docker containers can pose security risks, requiring careful attention to best practices for securing containerized environments.

Why we choose docker?

a) **Consistency**
Docker ensures consistent deployment and runtime environments, critical for maintaining compliance and reliability in healthcare systems.

b) **Isolation**
Containerization isolates applications and dependencies, reducing the risk of conflicts and ensuring data privacy and security.

c) **Scalability**

Docker's container-based architecture allows the RCM system to scale efficiently to handle increasing workloads and data volumes.

d) Flexibility
Docker's portability enables easy deployment across different environments, including on-premises servers and cloud platforms, providing flexibility in infrastructure choices.

e) DevOps Alignment
Docker aligns well with DevOps practices, facilitating automation, continuous integration, and deployment workflows, improving agility and collaboration in RCM system development and operations.

1.11.5 MongoDB

MongoDB is a NoSQL database management system that offers flexibility, scalability, and performance for modern application development. It uses a document-oriented data model, storing data in flexible JSON-like documents, making it suitable for a wide range of use cases, including real-time analytics, content management, and mobile applications.

Advantages:

a) **Flexible Data Model:** MongoDB's document-oriented approach allows for dynamic and schema-less data structures, accommodating evolving application requirements.
b) **Scalability:** MongoDB's distributed architecture supports horizontal scaling across multiple servers, enabling seamless scaling as data volume and user load grow.
c) **Performance:** With features like indexing, sharding, and in-memory storage engine, MongoDB delivers high-performance queries and data retrieval operations.
d) **Replication and High Availability:** MongoDB provides automatic replication and failover capabilities, ensuring data availability and reliability in case of server failures.
e) **Rich Query Language:** MongoDB Query Language (MQL) supports a wide range of query operations, including CRUD operations, aggregation, text search, and geospatial queries.
f) **Community Support:** MongoDB has a vibrant community of users, contributors, and experts, providing resources, documentation, and support for developers.

Disadvantages:

a) **Data Consistency:** MongoDB sacrifices some aspects of data consistency in favor of scalability and performance, requiring careful consideration of application requirements and data integrity.
b) **Learning Curve:** Developers familiar with relational databases may face a learning curve when adapting to MongoDB's document-oriented data model and query language.
c) **Operational Complexity:** Managing a distributed MongoDB cluster with multiple nodes requires expertise in deployment, configuration, monitoring, and maintenance, adding complexity to operations.

Why did we choose MongoDB?

a) **Flexible Data Model**
MongoDB's schema-less design accommodates the complex and evolving data structures often encountered in healthcare systems, such as patient records and medical histories.

b) **Scalability**
MongoDB's horizontal scaling capabilities allow the RCM system to handle growing data volumes and user loads without sacrificing performance.

c) **Performance**
MongoDB's indexing, sharding, and in-memory storage engine ensure fast query response times, critical for real-time data processing and analytics in healthcare.

d) **High Availability**
MongoDB's replication and failover features ensure data availability and reliability, minimizing downtime and ensuring continuous operation of the RCM system.

e) **Community Support**
MongoDB's active community provides resources, best practices, and support, aiding developers in building and maintaining the RCM system effectively.

We used the following technologies to develop the web application.

React is a declarative, efficient, and flexible JavaScript library for building user interfaces

MongoDB is a document database used to build highly available and scalable internet applications. With its flexible schema approach, it's popular with

Java Spring Boot (Spring Boot) is a tool that makes developing web application and microservices with Spring Framework faster and easier through three core

Apart from the above tech Stack we use the following tools and technologies in our project.

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control.

GitHub is a code hosting platform for version control and collaboration. It lets us to work together on projects from anywhere.

1.12 Methods

1.12.1 Requirements Engineering Methods

1) **Use Case Modeling**
   Describe system interactions and behaviors from the perspective of users.
2) **User Stories**
   Define system features and functionalities from the end user's viewpoint.
3) Requirement Workshops
   Gather stakeholders to elicit and prioritize system requirements collaboratively.

1.12.2 Design Methods

1) **Domain-Driven Design (DDD)**
   Focus on modeling the healthcare domain and its complexities to design a relevant system architecture.
2) **Component-Based Design**
   Break down the system into modular components for easier development and maintenance.
3) **Responsive Web Design**
   Ensure the RCM system's user interface is accessible and usable across different devices and screen sizes.

1.12.3 Testing Methods

1) **Unit Testing**
   Test individual components and functions to ensure they work as expected.
2) **Integration Testing**
   Verify the interaction and integration between different modules and components.
3) **End-to-End Testing**
   Test the entire system workflow to ensure all components work together seamlessly.

1.12.4 Integration Methods

1) **RESTful APIs**
   Enable seamless integration with external systems and services.
2) **Message Queues (RabbitMQ – Not included in the first stage)**
   Facilitate asynchronous communication and integration between different parts of the system.

Git Repository

BackEnd - https://github.com/DulangaMW/Innobothealth-Access-Management-System.git
FrontEnd- https://github.com/InnobotHealth/InnobotHealthFE.git

CHAPTER 2 – REQUIREMENTS

2.1 Stakeholder Analysis

1. Healthcare Providers and Administrators:

- Interest: They aim to streamline administrative tasks, enhance patient care, and optimize resource allocation.

- Influence: They have significant influence as primary users and decision-makers in the healthcare organization.

- Engagement: Critical for requirements gathering, user testing, and ensuring system alignment with operational needs.

2. Patients:

- Interest: Seeking efficient appointment scheduling, seamless record management, and improved healthcare experiences.

- Influence: Indirect influence through demand for quality care and feedback mechanisms.

- Engagement: Essential for usability testing, feedback collection, and ensuring patient-centric design.

3. Insurance Providers:

- Interest: Interested in accurate record management, efficient claims processing, and compliance with regulations.

- Influence: Have regulatory and financial influence, impacting reimbursement processes.

- Engagement: Collaboration needed for system integration, compliance checks, and data exchange protocols.

4. Suppliers and Vendors:

- Interest: Concerned with inventory management, system integration, and timely payments.

- Influence: Influence inventory processes and supply chain efficiency.

- Engagement: Collaboration necessary for seamless integration, supplier performance tracking, and feedback on system usability.

5. IT Staff:

- Interest: Ensuring system security, scalability, and compatibility with existing infrastructure.

- Influence: Key in system implementation, maintenance, and ensuring data integrity.

- Engagement: Essential for technical assessments, system deployment, and ongoing support.

6. Regulatory Authorities:

- Interest: Ensure compliance with healthcare regulations, data privacy laws, and industry standards.

- Influence: Direct influence on system requirements, data handling protocols, and security measures.

- Engagement: Collaboration vital for regulatory assessments, compliance audits, and ensuring adherence to legal frameworks.

2.2 Requirements Analysis

Requirements Analysis for Innobot Hospital Management System:

1. Inventory Management:

- Functionality: Add, edit, and delete supplier information and inventory items.

- Integration: Seamless barcode scanning technology integration for accurate tracking.

- Multi-location Support: Ability to manage inventory across multiple facilities or warehouses.

2. Notification Management:

- Customization: Allow users to customize notification preferences based on roles.

- Real-time Delivery: Deliver notifications in real-time or near real-time.

- Audit Trail: Maintain an audit trail of all communication within the system.

3. Staff Management:

- On boarding: Facilitate on boarding of new staff and allocation of roles.

- Access Control: Role-based access control for staff privileges and visibility settings.

- Performance Monitoring: Track staff activity logs and monitor key performance indicators.

4. Appointment Scheduling Management:

- Patient Information: Store and manage patient details, including medical history and contact information.

- Eligibility Verification: Verify patient eligibility based on insurance information.

- Real-time Updates: Send timely reminders to patients and providers and suggest alternative appointment slots.

5. Insurance Management:

- Record Management: Create, delete, update, and archive insurance records.

- Fee Schedule Management: Define and manage fee schedules for different CPT codes.

- Analytics: Generate reports and analytics for trend identification and performance evaluation.

6. Claim Management:

- Access Control: Role-based access control for claim processing functionalities.

- Integration: Integrate with external systems for seamless data exchange.

- Customizable Workflows: Define customizable workflows for claim approval processes.

7. Procedure and Diagnosis Management:

- Procedure Codes: Add, update, delete, search, and associate procedure codes.

- Diagnosis Codes: Add, update, delete, search, and associate diagnosis codes.

8. Patient Management:

- Registration: Register new patients, verify insurance details, and manage duplicates.

- Record Management: Search, access, view, update, and organize patient records securely.

2.3 Requirements Modeling

Requirement Modelling for Innobot Hospital Management System:

2.3.1 1. Inventory Management

- The system shall allow users to manage supplier information (add, edit, delete) including contract details.
- The system shall enable users to track supplier performance through key metrics.
- The system shall allow users to manage inventory items (add, edit, delete).
- The system shall provide search and filter functionalities for inventory data.
- The system shall support multi-location inventory management with accurate tracking across facilities.

2.3.2 2. Notification Management

- The system shall allow users to set notification preferences (email, SMS, in-system) based on their roles.
- The system shall enable definition of triggers for key RCM events (registration, appointments, claims etc.).
- The system shall deliver notifications in real-time or near-real-time.
- The system shall maintain an audit trail for all communication within the system.

2.3.3 3. Staff Management

- The system shall facilitate the boarding of new practices with dedicated staff management functionalities.
- The system shall allow allocation of coordinators and staff during initial setup.
- The system shall enable the granting of specific access and privileges to staff members.
- The system shall allow staff to control the visibility of their profiles.
- The system shall enable the sending of messages to all logged users.
- The system shall provide functionalities to view staff activity logs and KPIs.
- The system shall allow management of staff data (create, delete, update user profiles and privileges).

2.3.4 4. Appointment Scheduling Management

- The system shall store and manage patient information (personal details, contact information, medical history).
- The system shall allow patients to submit appointment requests.
- The system shall verify patient eligibility based on insurance information.
- The system shall check healthcare provider availability and display available slots.
- The system shall allow cancellation of appointments by patients and update records accordingly.
- The system shall send timely reminders to patients and providers for appointments.

2.3.5 5. Insurance Management

- The system shall allow users to create, delete, or archive insurance records.
- The system shall allow updating and modifying existing insurance records.
- The system shall enable administrators to define and manage fee schedules for CPT codes.
- The system shall allow authorized users to update fee schedules.
- The system shall generate insurance records reports and provide analytics features.

2.3.6 6. Claim Management

- The system shall implement role-based access control for claim functionalities.
- The system shall track the status of submitted claims (approved, rejected, and pending).
- The system shall integrate with external systems for data exchange.
- The system shall allow for customizable claim approval workflows.

- The system shall generate reports on claim status, processing times, and financial performance.

2.3.7 7. Procedure and Diagnosis Management

- The system shall allow adding, updating, and deleting codes with descriptions.
- The system shall allow searching and displaying a list of codes.
- The system shall allow associating procedure codes with diagnosis codes (when applicable).

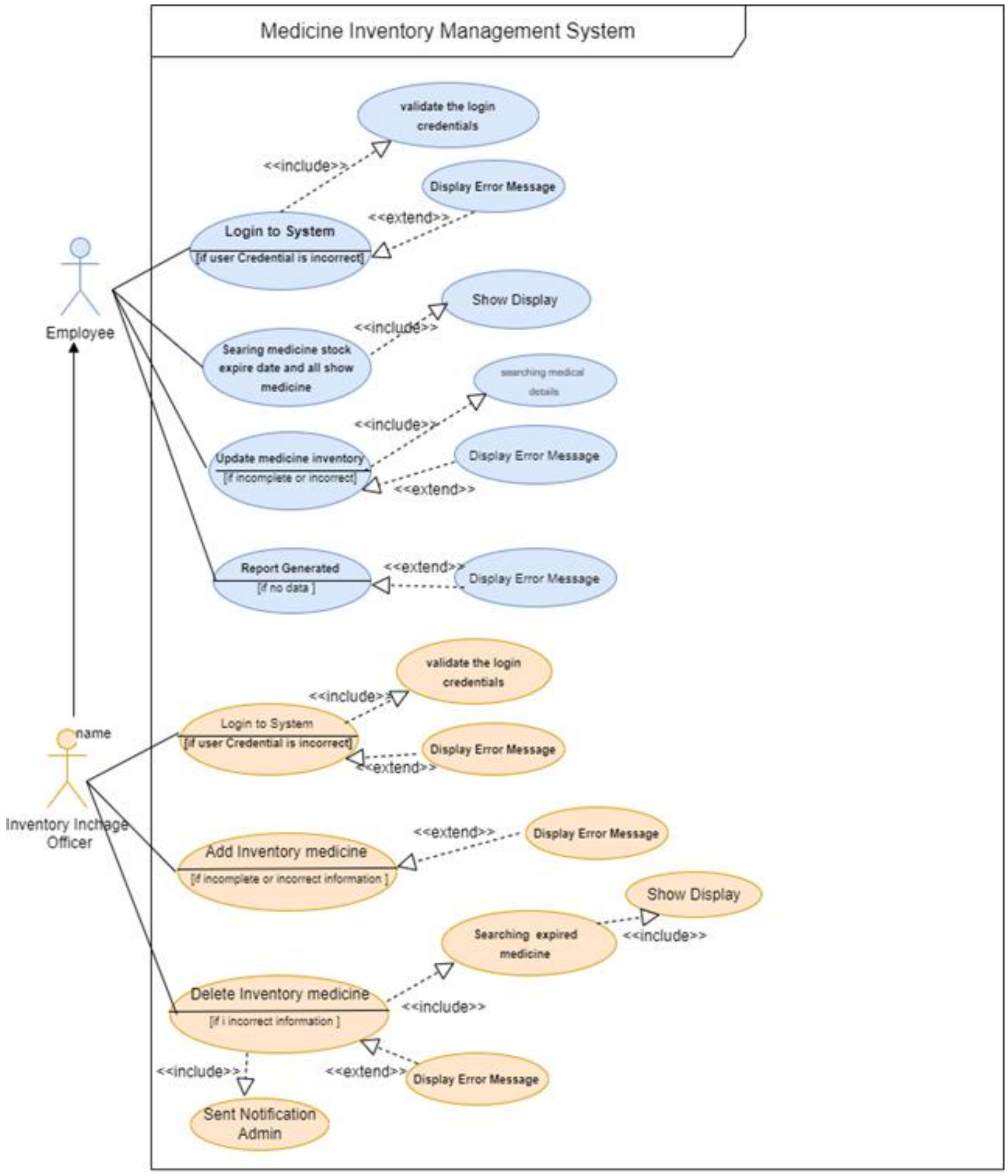
2.3.8 8. Patient Management

- The system shall allow registration of new patients with detailed verification.
- The system shall allow searching and accessing patient information.
- The system shall allow viewing and updating patient records.
- The system shall allow managing patient records (including handling duplicates).
- The system shall securely store and manage patient details.

Chapter 3 - Design and Development

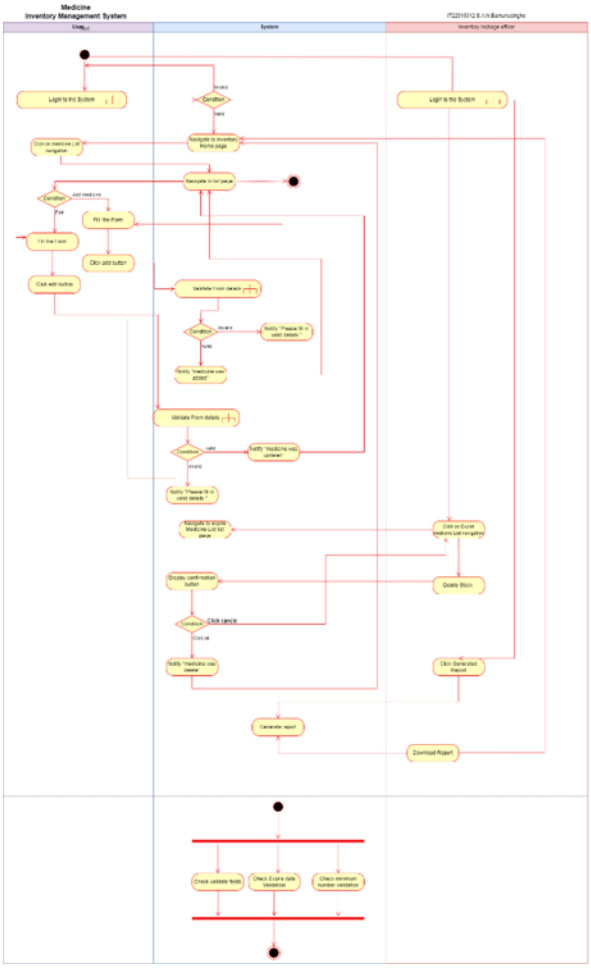
3.1 Diagram of Components

Use case diagram IT22515612 S.A.N. Bamunusinghe


Medicine Inventory Management System
Employee
validate the login credentials
<<include>>
Display Error Message
<<extend>>
Login to System
[if user Credential is incorrect]
Show Display
<<include>>
Searing medicine stock expire date and all show medicine
searching medical details
<<include>>
Update medicine inventory
[if incomplete or incorrect]
Display Error Message
<<extend>>
Report Generated
[if no data]
<<extend>>
Display Error Message
validate the login credentials
<<include>>
name
Login to System
[if user Credential is incorrect]
Display Error Message
<<extend>>
Inventory Inchage Officer
<<extend>>
Display Error Message
Add Inventory medicine
[if incomplete or incorrect information]
Show Display
Searching expired medicine
<<include>>
Delete Inventory medicine
[if i incorrect information]
<<include>>
<<include>>
<<extend>>
Display Error Message
Sent Notification Admin

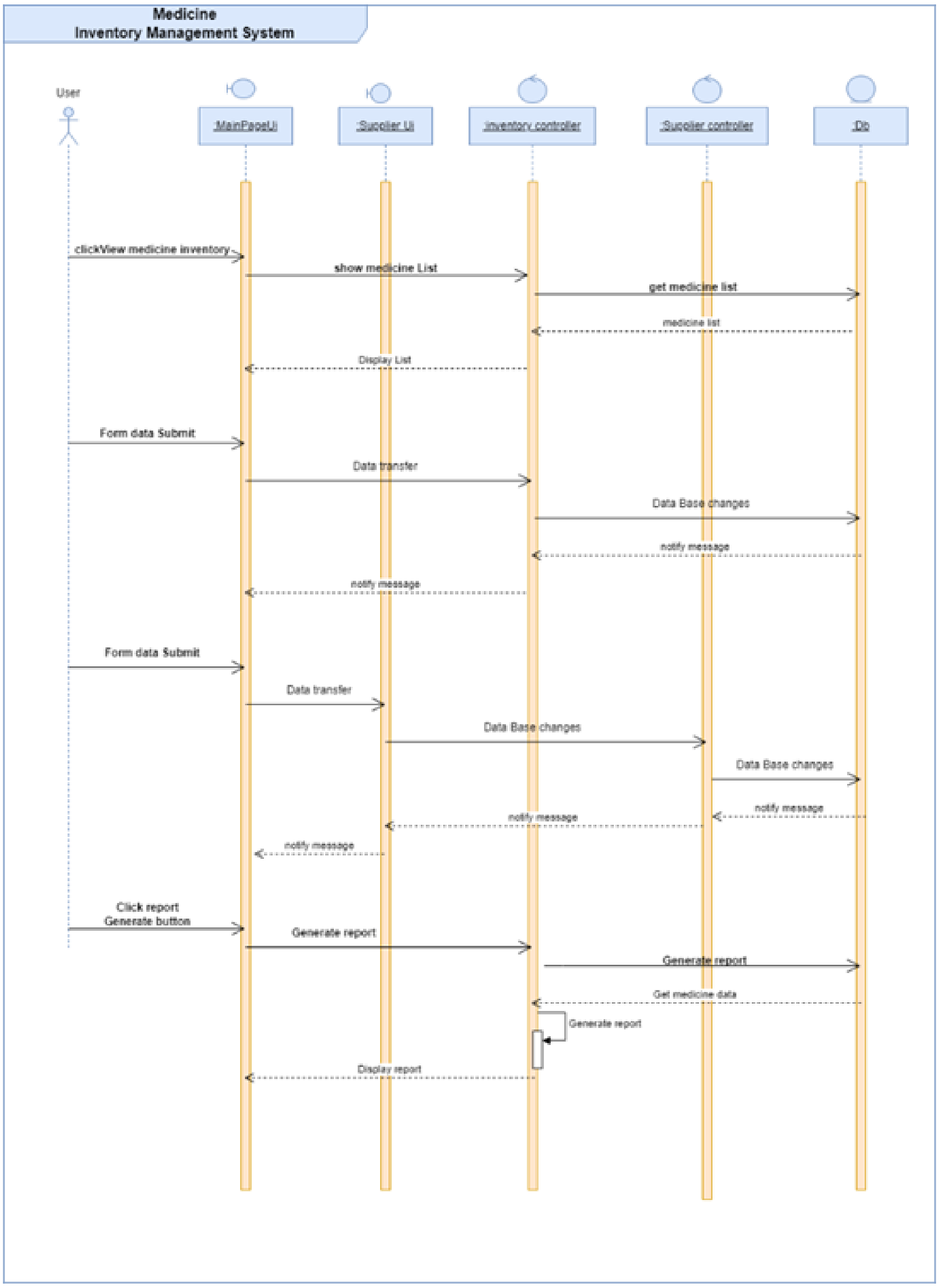
Activity diagram IT22515612 S.A.N. Bamunusinghe

https://drive.google.com/file/d/1POUY-jHZDG8yLq_PLwMUgJQzx0MBHdKJ/view?usp=sharing

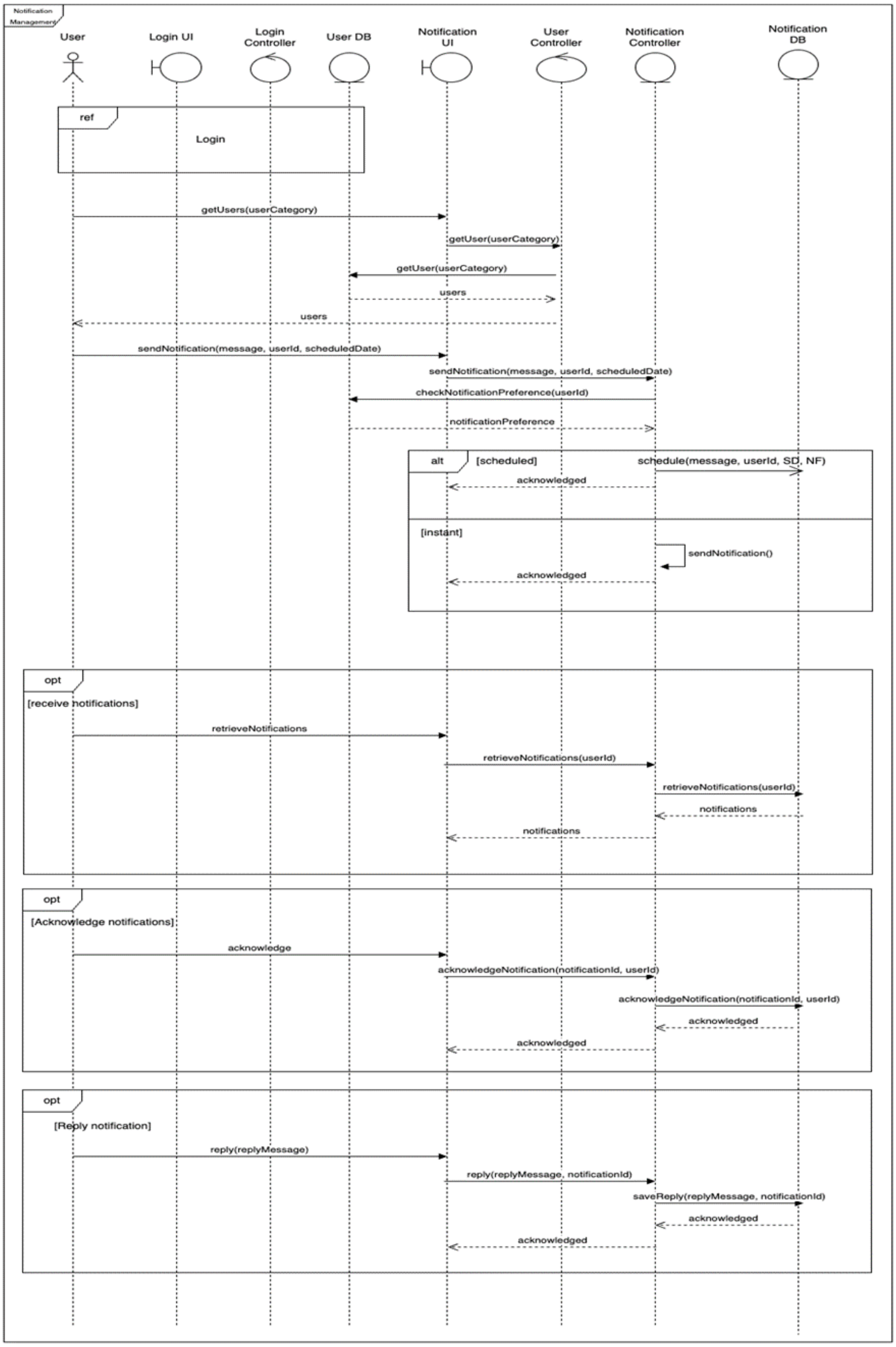

Medicine
Inventory Management System
IT22515612 S.A.N Bamunusinghe
User
System
Inventory Incharge officer
Login to the System
Condition
invalid
valid
Login to the System
Click on medicine List navigation
Navigate to inventory Home page
Navigate to list page
Condition
Add medicine
Edit
Fill the Form
Fill the Form
Click add button
Click edit button
Validate From details
Condition
invalid
Notify "Please fill in valid details "
valid
Notify "medicine was added"
Validate From details
Condition
valid
Notify "medicine was updated"
invalid
Notify "Please fill in valid details "
Navigate to expire Medicine List list page
Click on Expire medicine List navigation
Display confirmation button
Delete Stock
Condition
Click cancle
Click ok
Notify "medicine was delete"
Click Generated Report
Generate report
Download Report
Check validate fields
Check Expire date Validation
Check minimum number validation

Sequence diagram

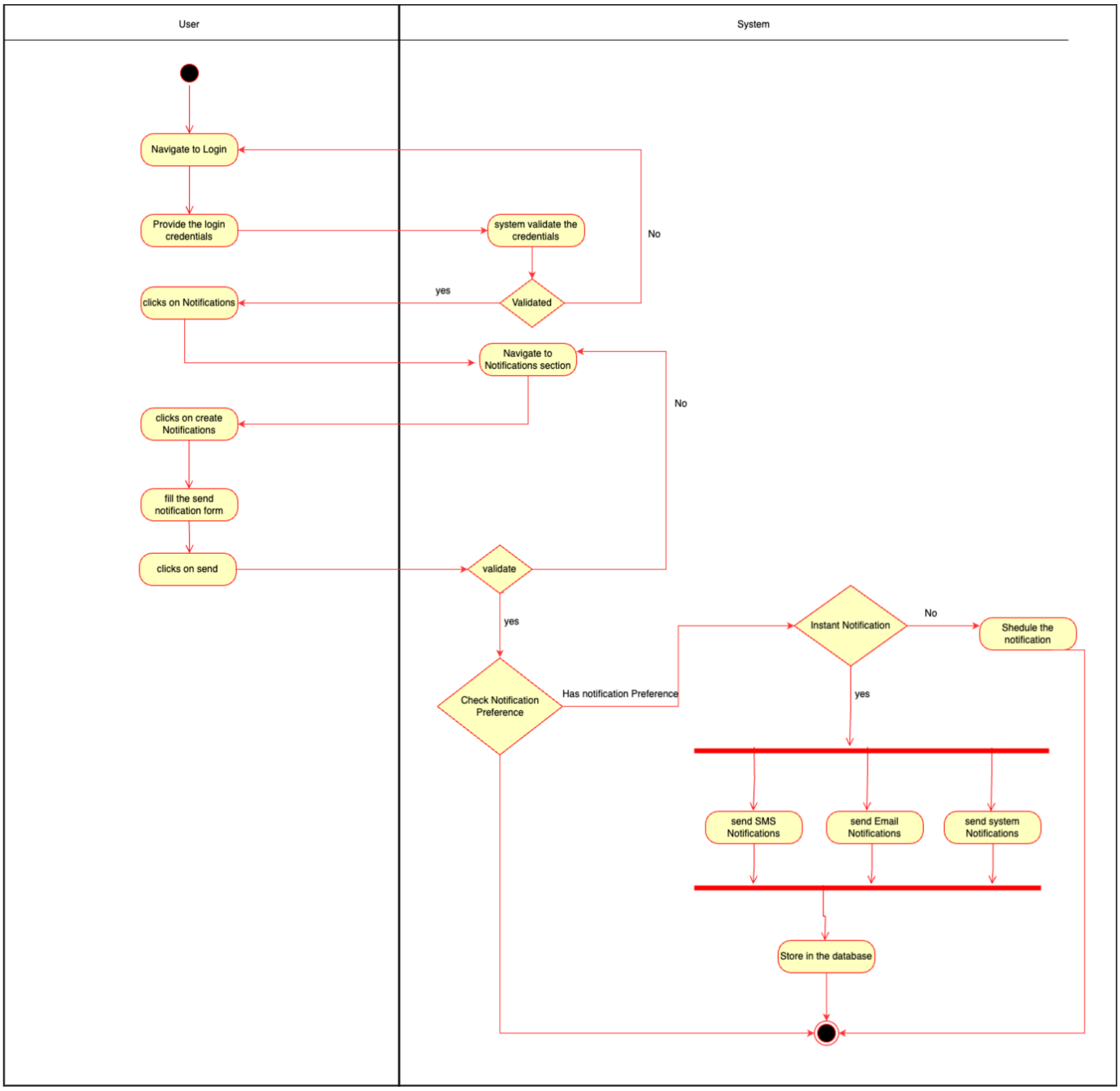
IT22515612 S.A.N. Bamunusinghe



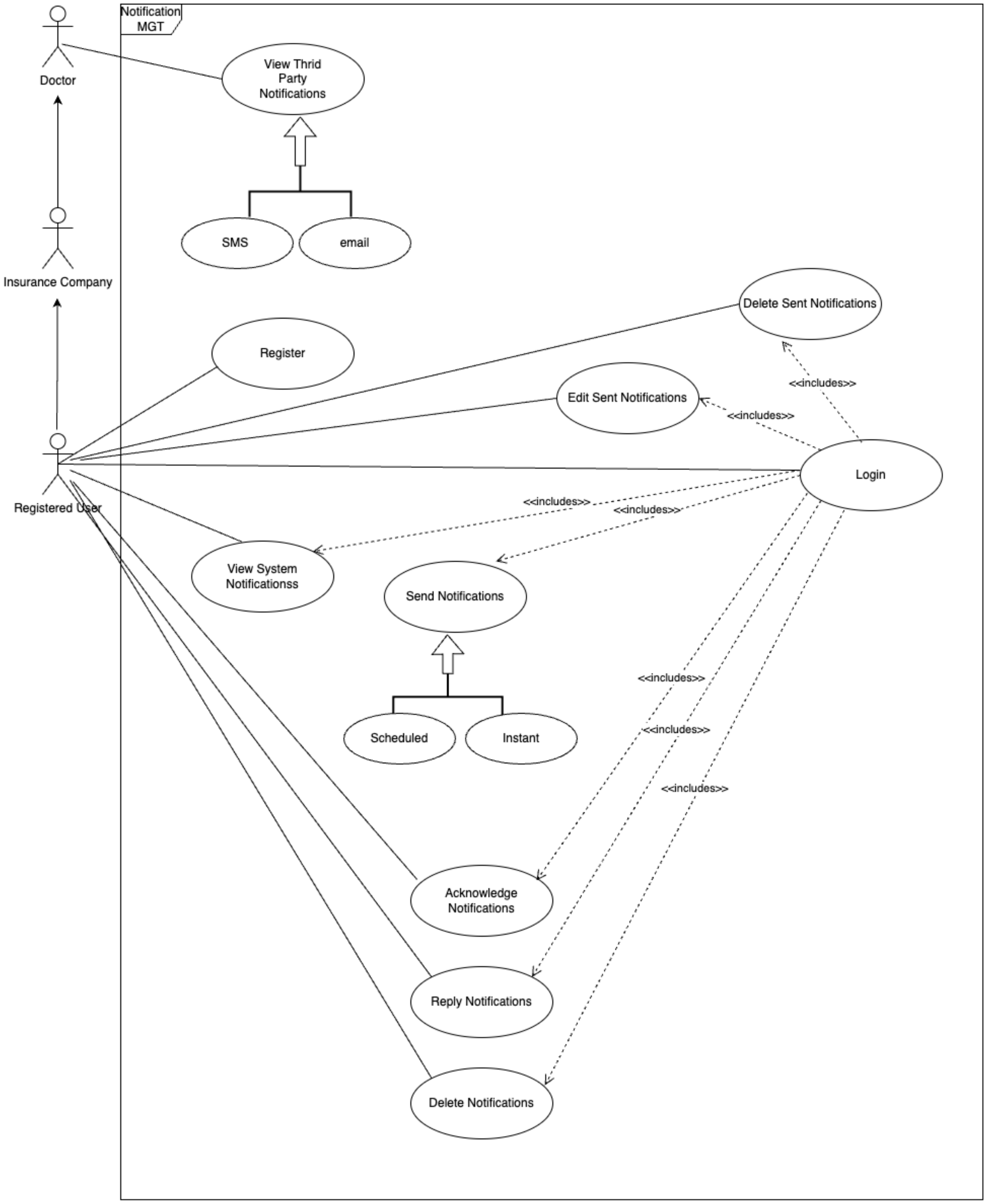
DME Wimalagunasekara - Sequence Diagram IT22917270



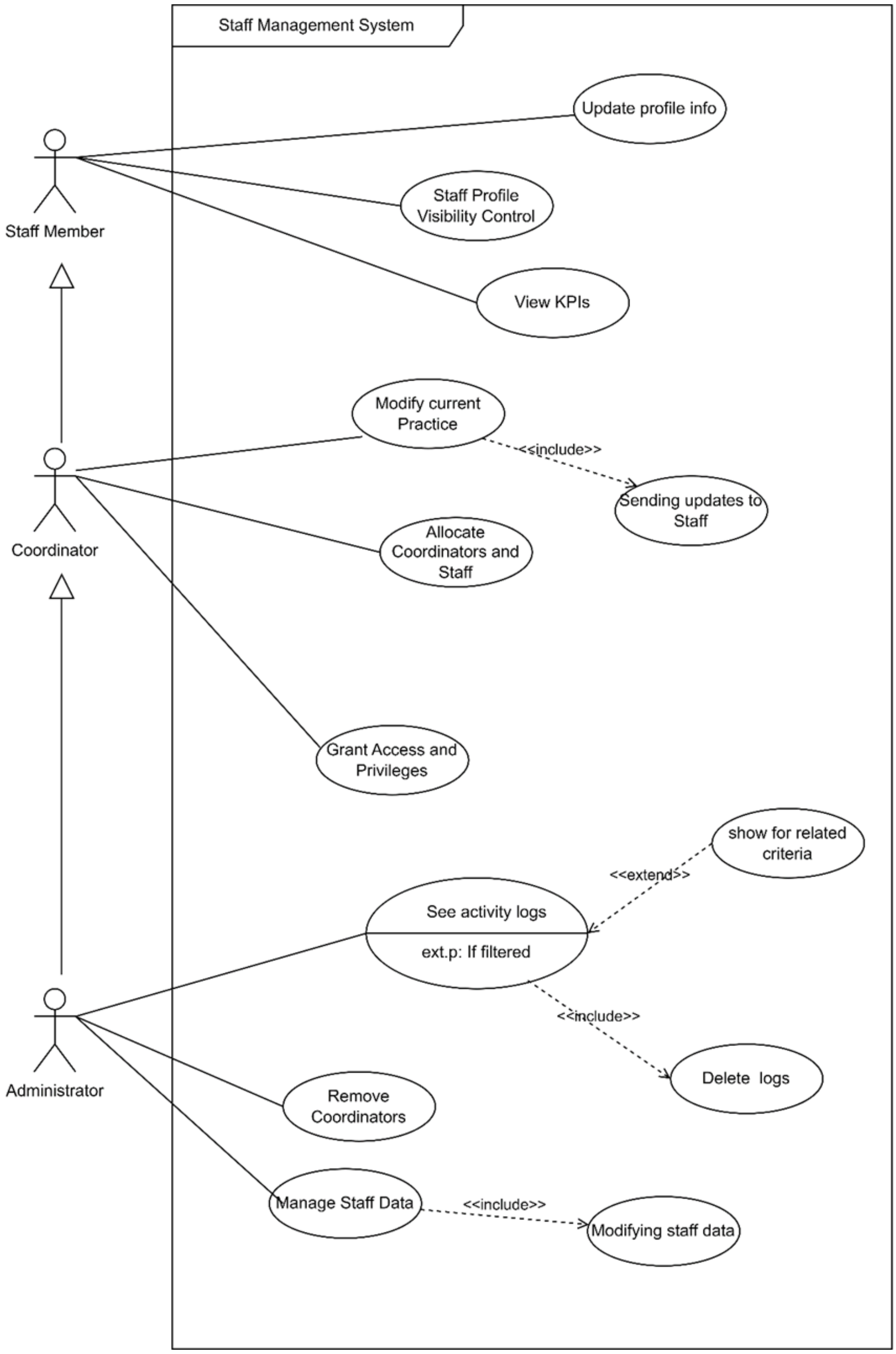
DME Wimalagunasekara - Activity Diagram IT22917270

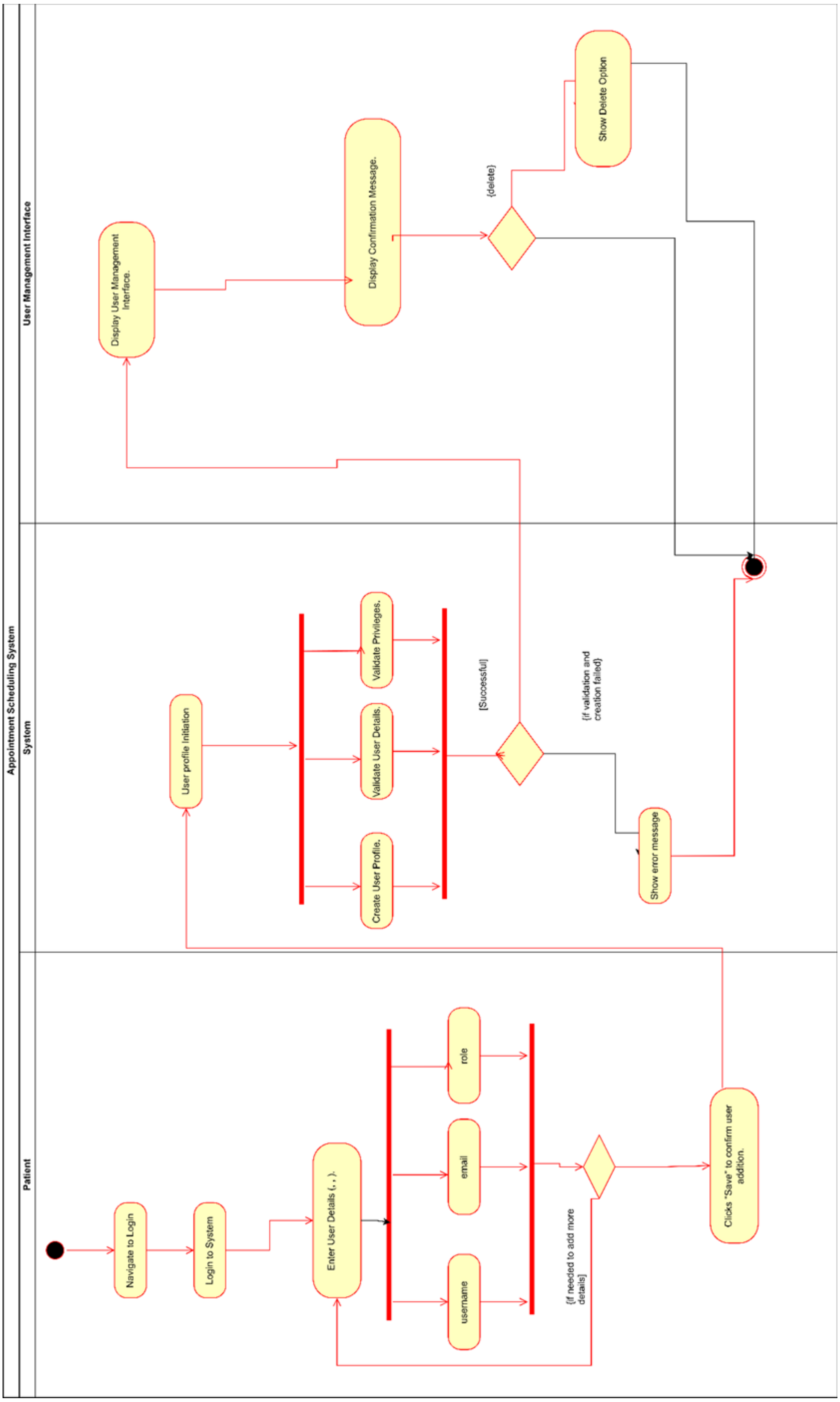

User
System
Navigate to Login
Provide the login credentials
system validate the credentials
No
yes
clicks on Notifications
Validated
Navigate to Notifications section
No
clicks on create Notifications
fill the send notification form
clicks on send
validate
yes
No
Instant Notification
Shedule the notification
Check Notification Preference
Has notification Preference
yes
send SMS Notifications
send Email Notifications
send system Notifications
Store in the database

DME Wimalagunasekara -Use Case Diagram IT22917270

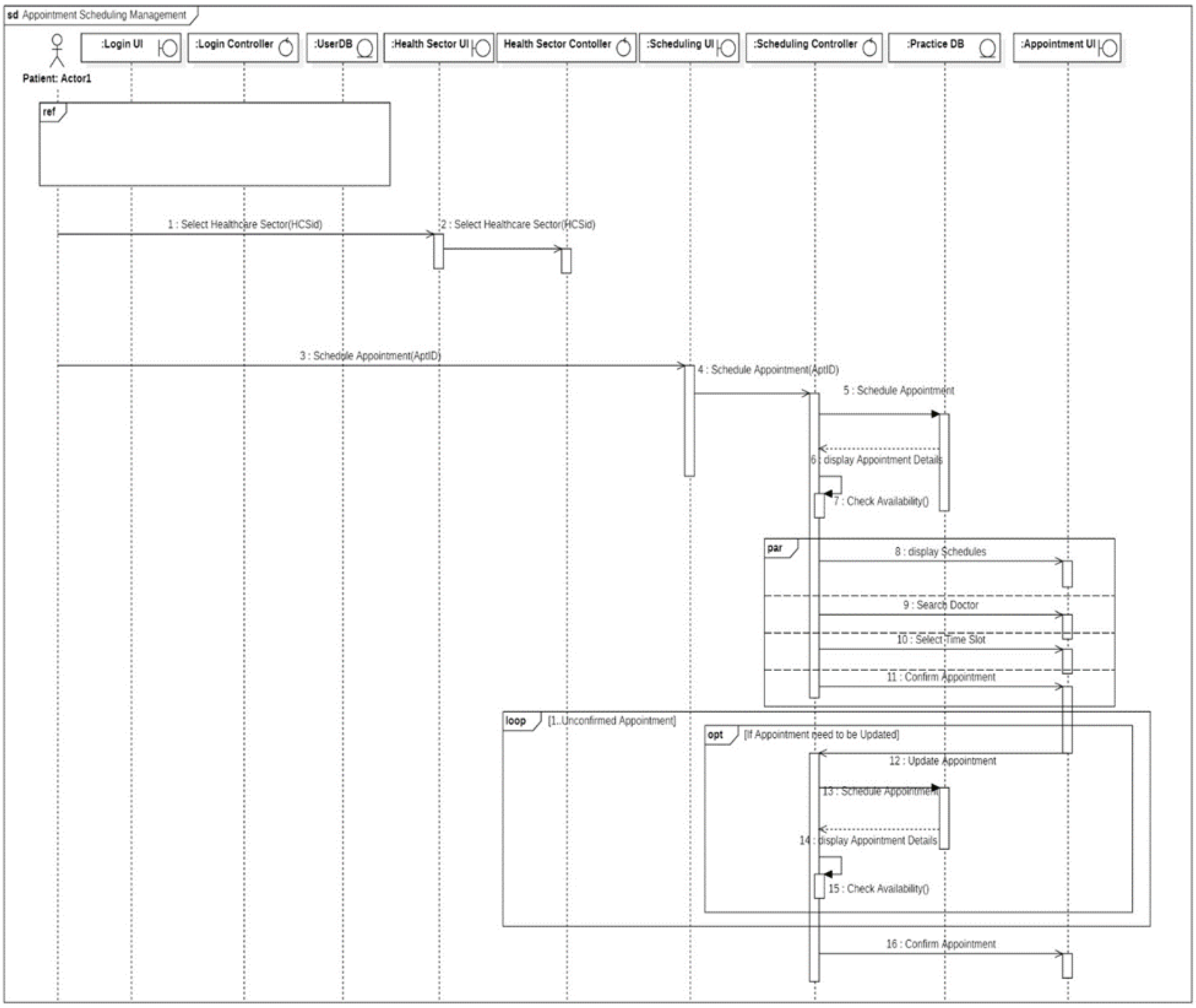


S A T Nayanapriya - Use Case Diagram IT22892058

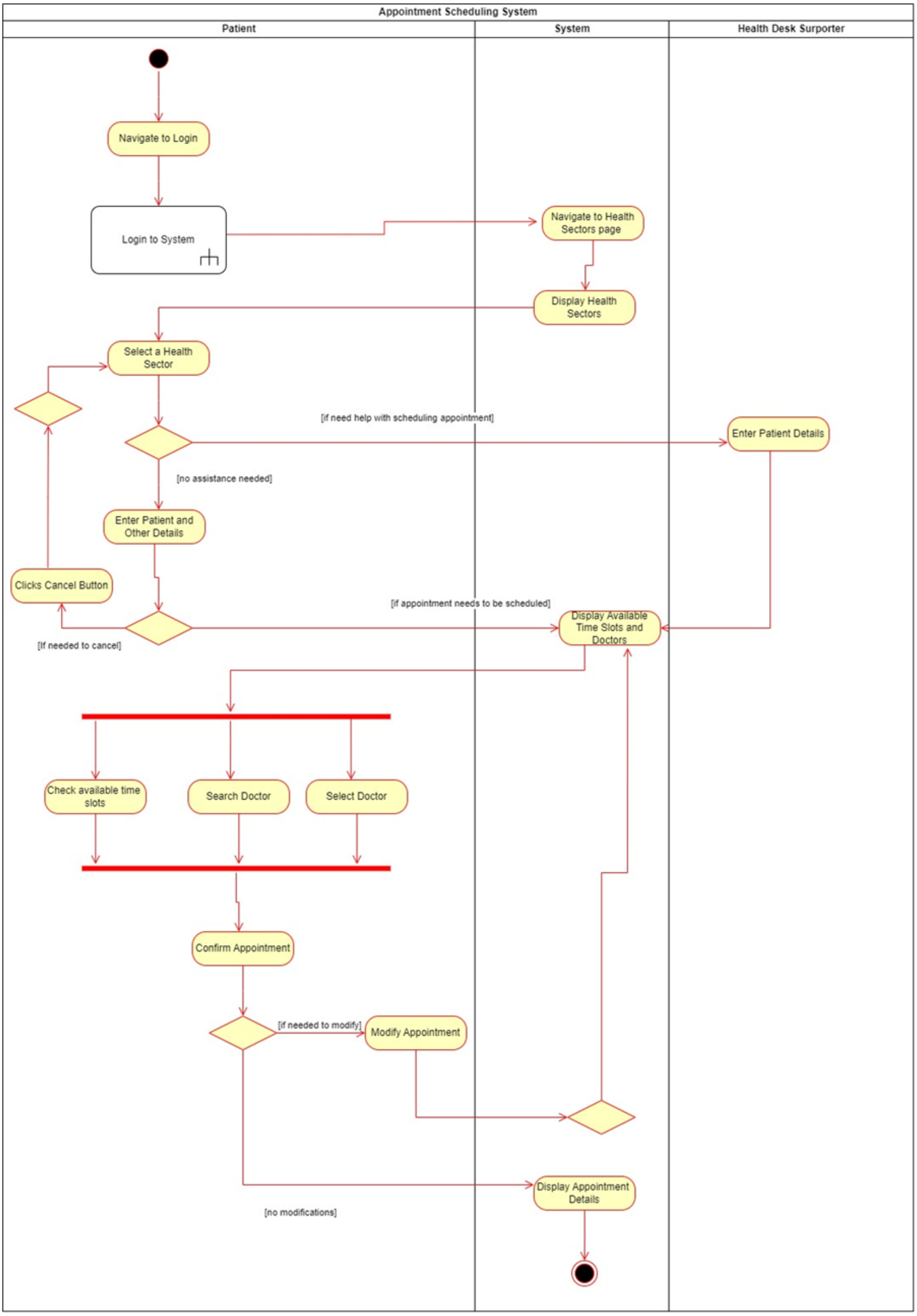



Appointment Scheduling System
Patient
System
User Management Interface
Navigate to Login
Login to System
Enter User Details (., ., .).
username
email
role
[if needed to add more details]
Clicks "Save" to confirm user addition.
User profile Initiation
Create User Profile.
Validate User Details.
Validate Privileges.
[Successful]
[if validation and creation failed]
Show error message
Display User Management Interface.
Display Confirmation Message.
{delete}
Show Delete Option

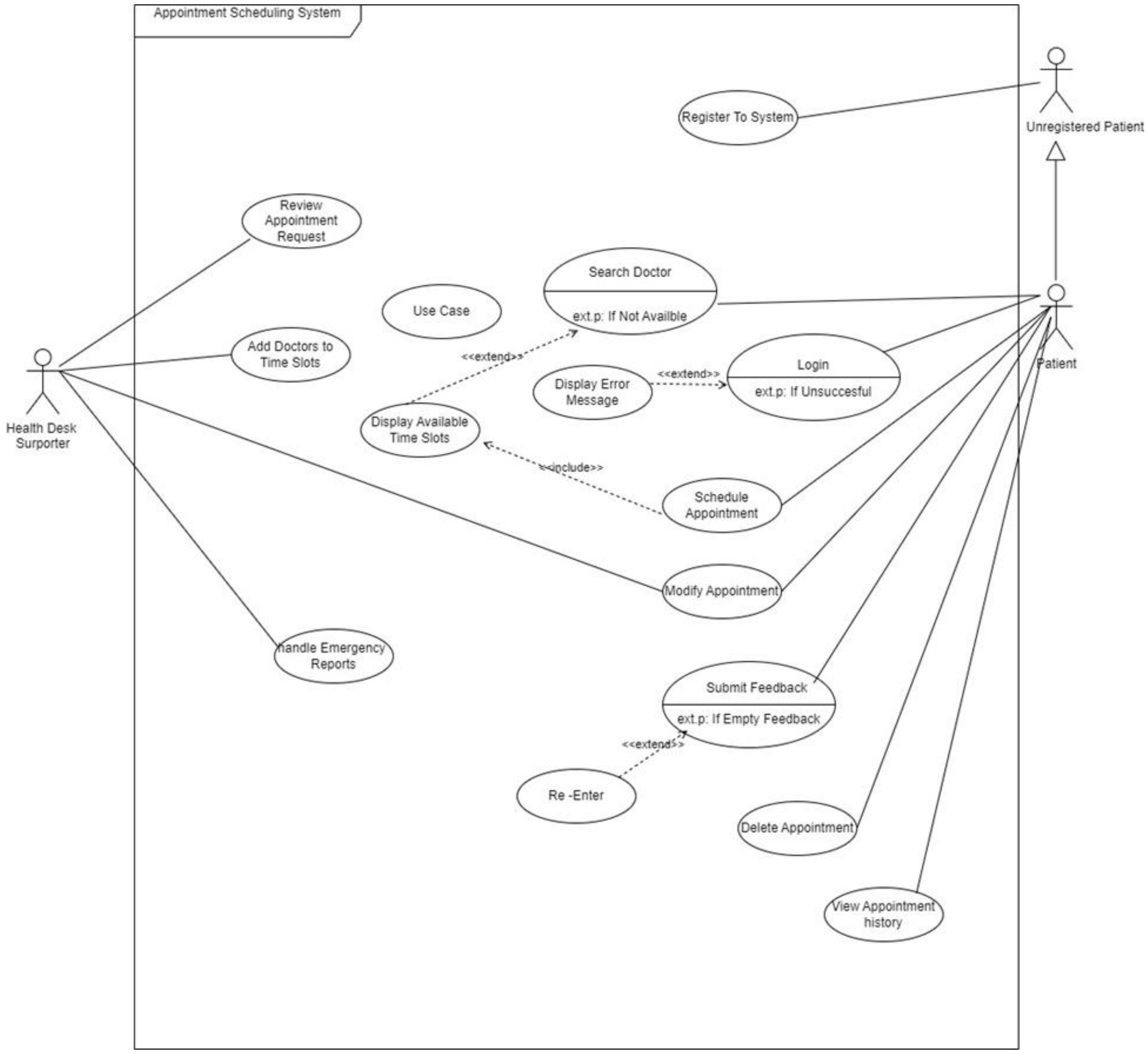
Obeyesekere A D -Sequence Diagram IT22332080


sd Appointment Scheduling Management
Patient: Actor1
:Login UI
:Login Controller
:UserDB
:Health Sector UI
Health Sector Contoller
:Scheduling UI
:Scheduling Controller
:Practice DB
:Appointment UI
ref
1 : Select Healthcare Sector(HCSid)
2 : Select Healthcare Sector(HCSid)
3 : Schedule Appointment(AptID)
4 : Schedule Appointment(AptID)
5 : Schedule Appointment
6 : display Appointment Details
7 : Check Availability()
par
8 : display Schedules
9 : Search Doctor
10 : Select Time Slot
11 : Confirm Appointment
loop
[1..Unconfirmed Appointment]
opt
[If Appointment need to be Updated]
12 : Update Appointment
13 : Schedule Appointment
14 : display Appointment Details
15 : Check Availability()
16 : Confirm Appointment

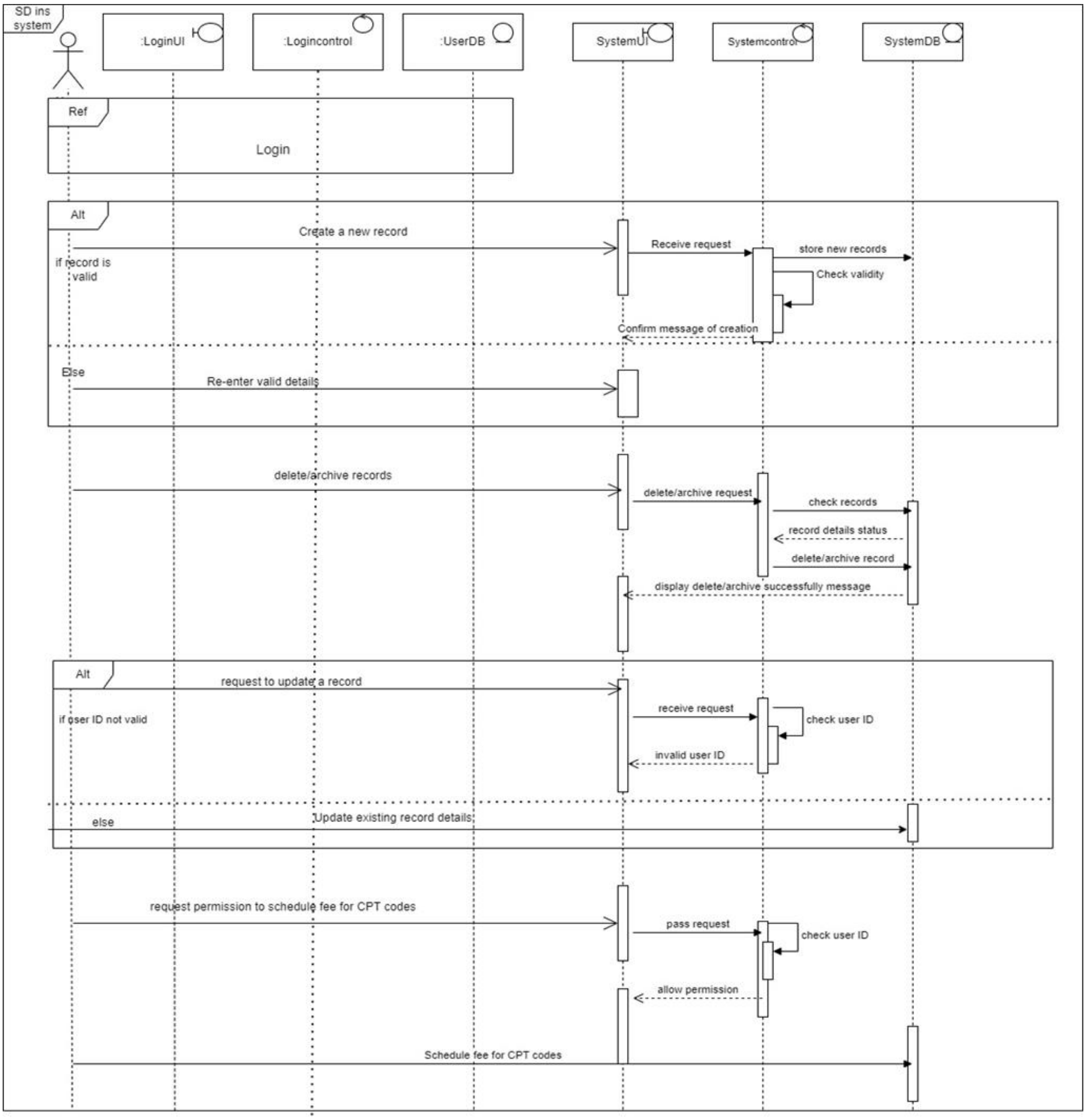
Obeyesekere A D -Activity Diagram IT22332080

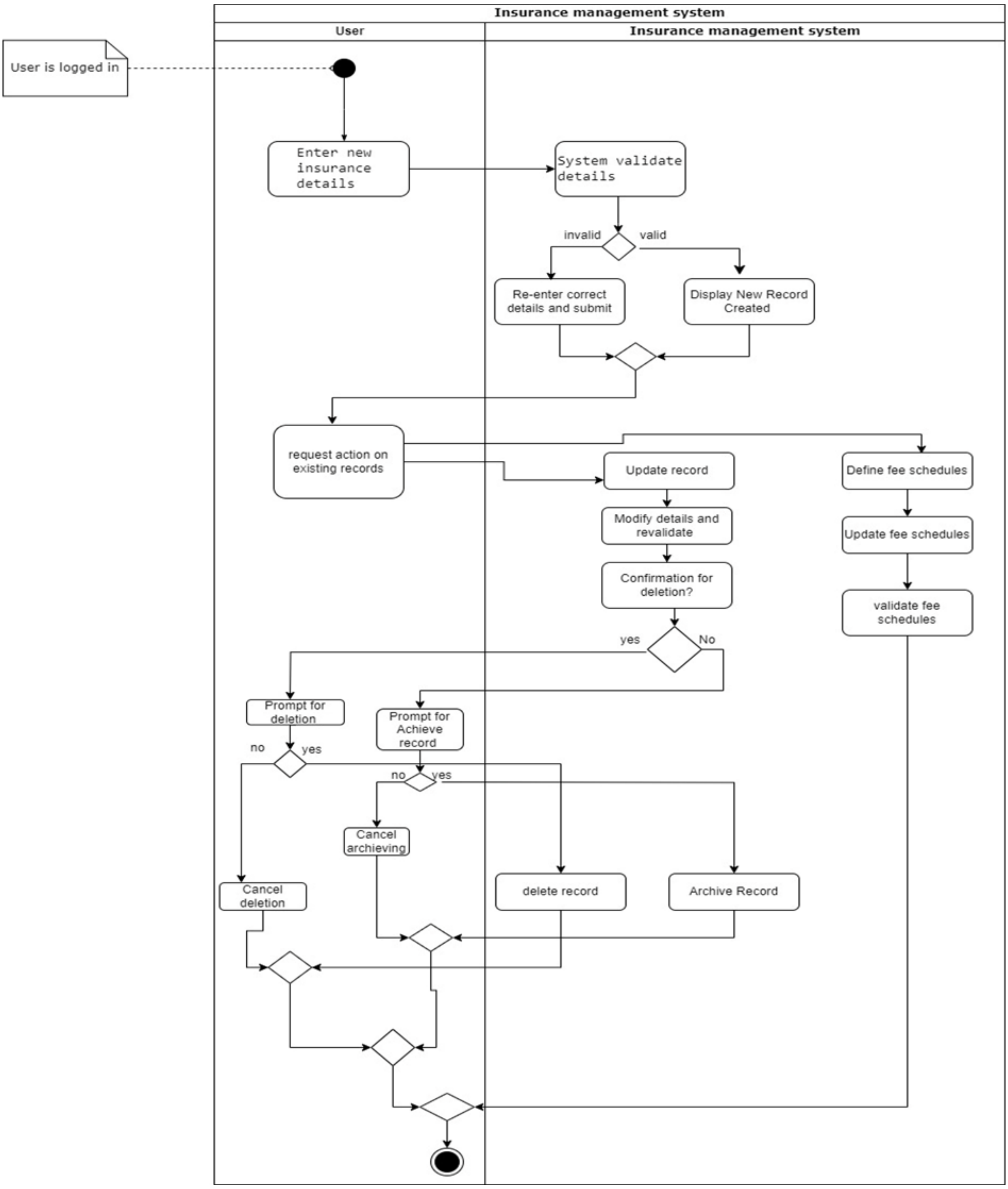


Obeyesekere A D -Use Case Diagram IT22332080

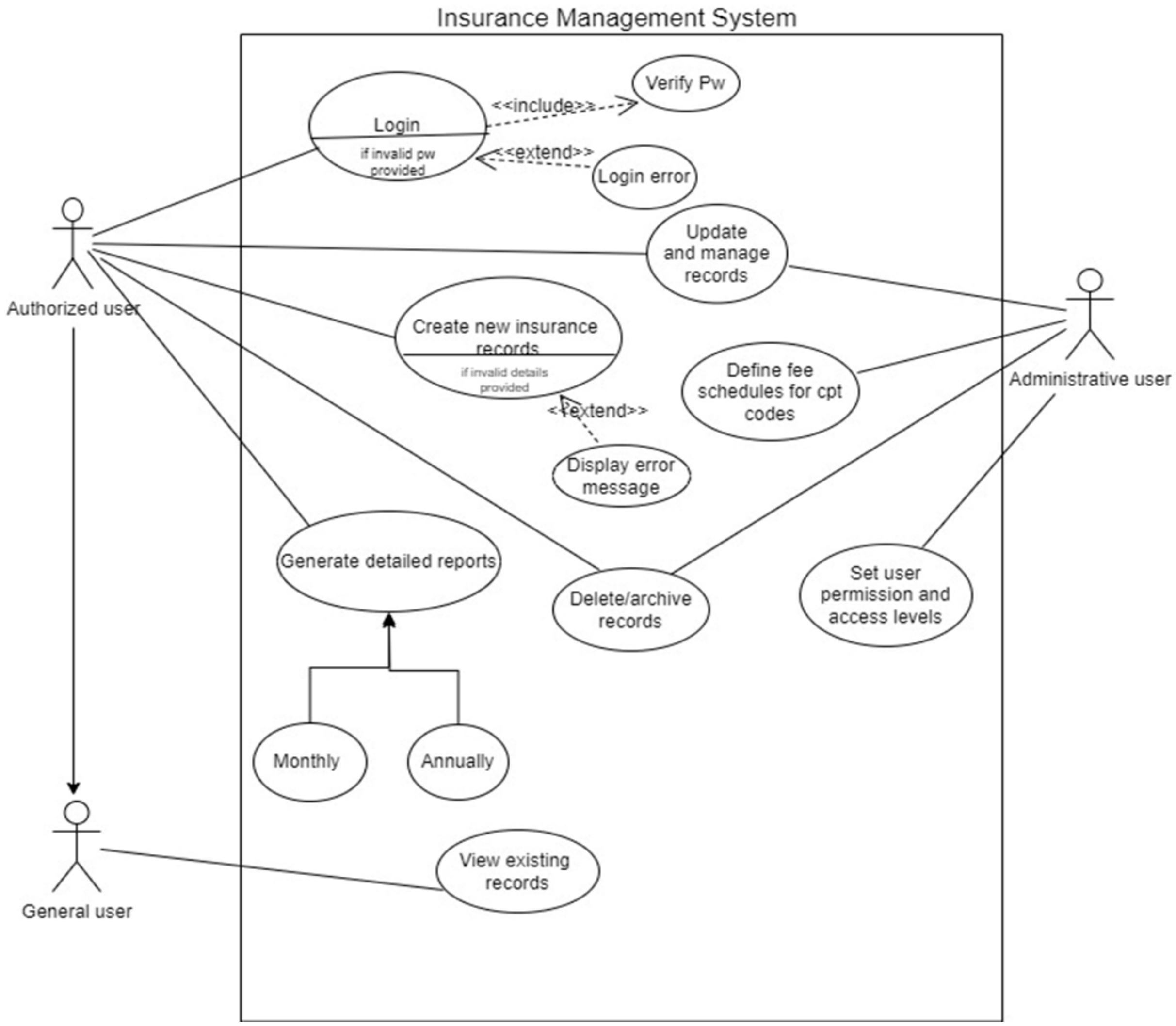


Chamikara M G S -Sequence Diagram IT22905840

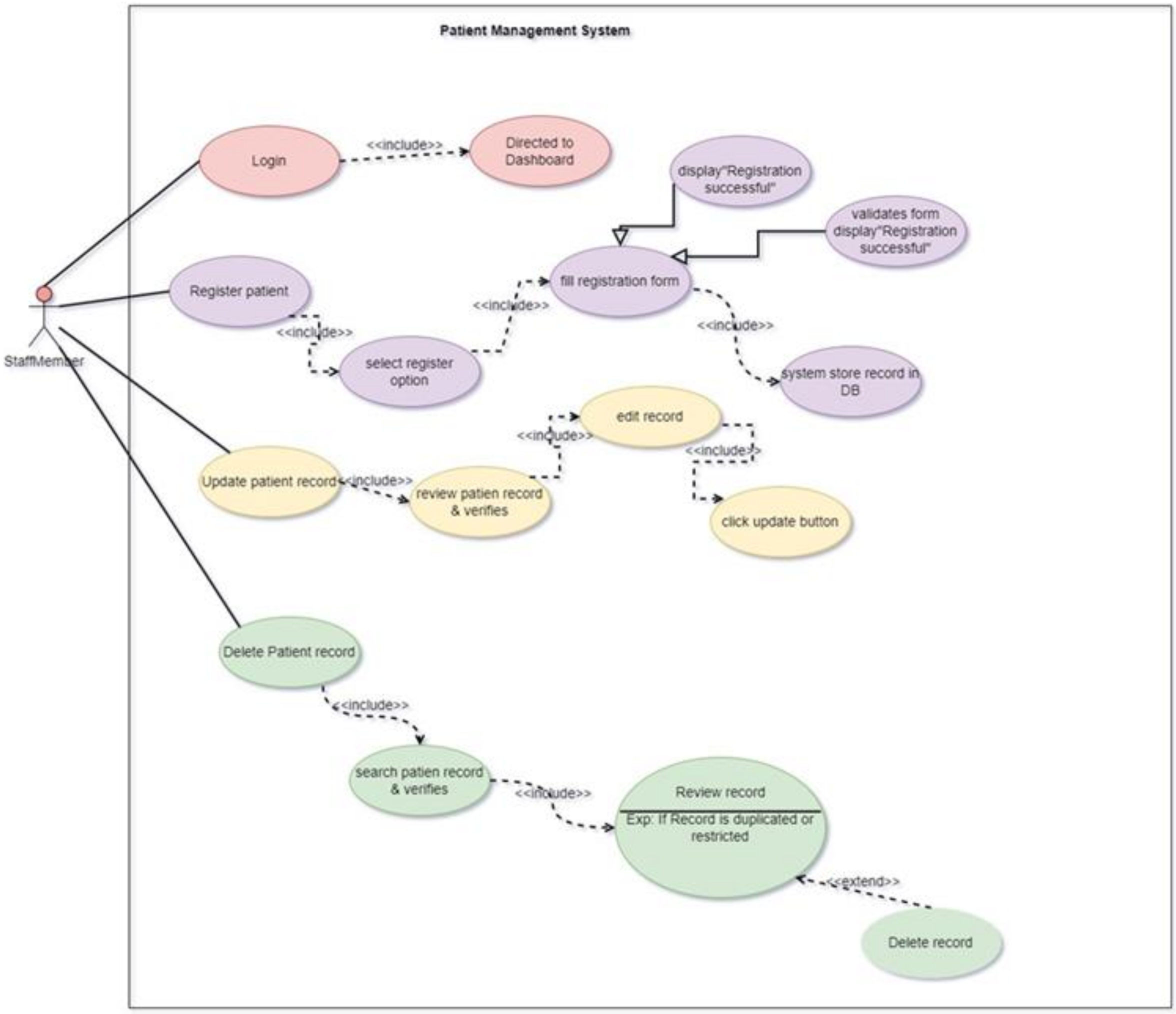



Insurance management system
User
Insurance management system
User is logged in
Enter new insurance details
System validate details
invalid
valid
Re-enter correct details and submit
Display New Record Created
request action on existing records
Update record
Modify details and revalidate
Confirmation for deletion?
yes
No
Define fee schedules
Update fee schedules
validate fee schedules
Prompt for deletion
no
yes
Prompt for Achieve record
no
yes
Cancel archieving
Cancel deletion
delete record
Archive Record

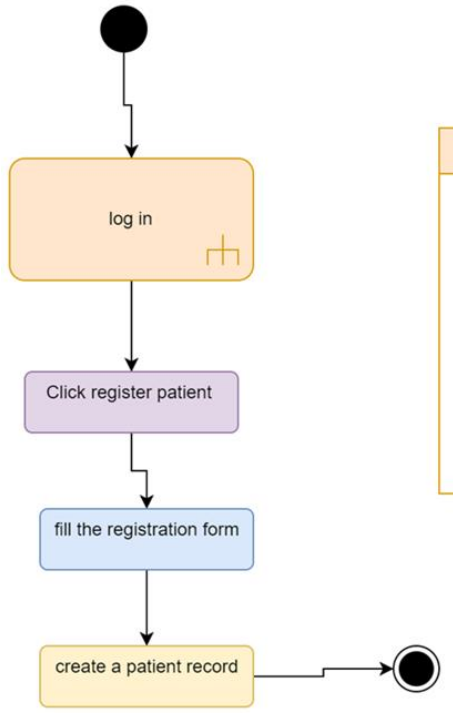
Chamikara M G S -Use Case Diagram IT22905840

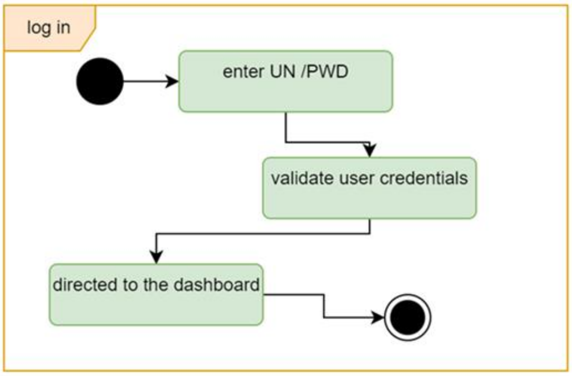

Insurance Management System
Login
if invalid pw provided
<<include>>
Verify Pw
<<extend>>
Login error
Update and manage records
Authorized user
Create new insurance records
if invalid details provided
<<extend>>
Display error message
Define fee schedules for cpt codes
Administrative user
Generate detailed reports
Delete/archive records
Set user permission and access levels
Monthly
Annually
General user
View existing records

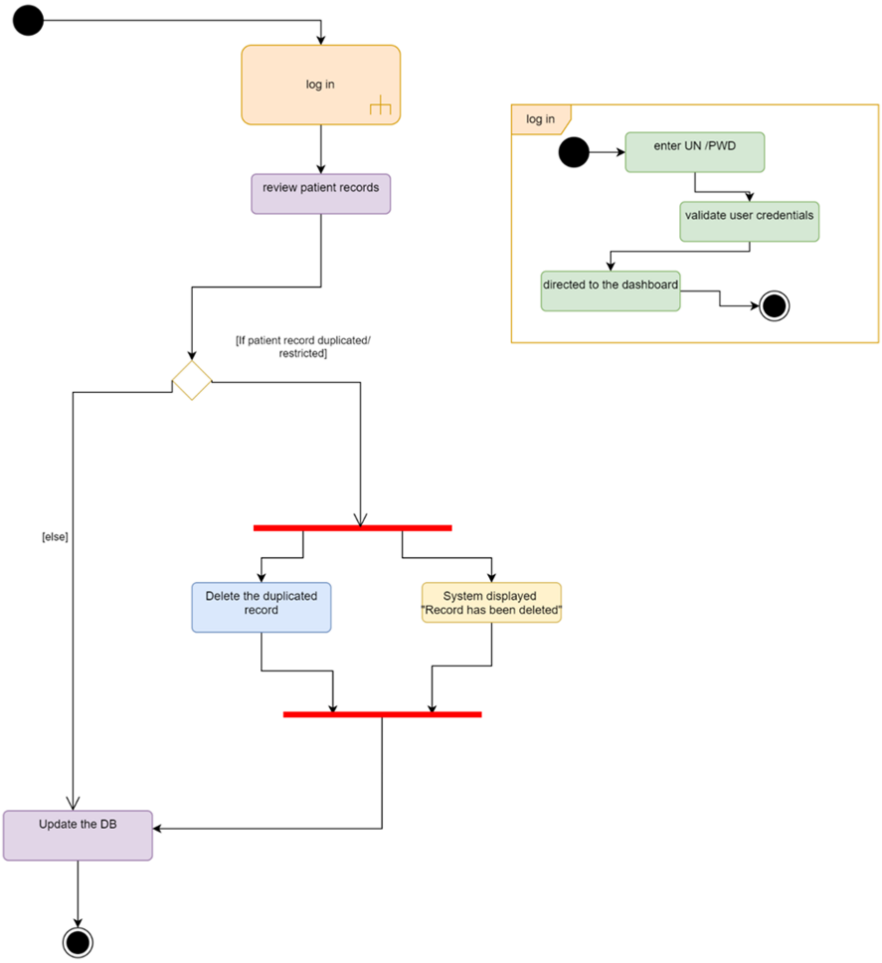
W A M Haren – IT 22307576 Use case diagram

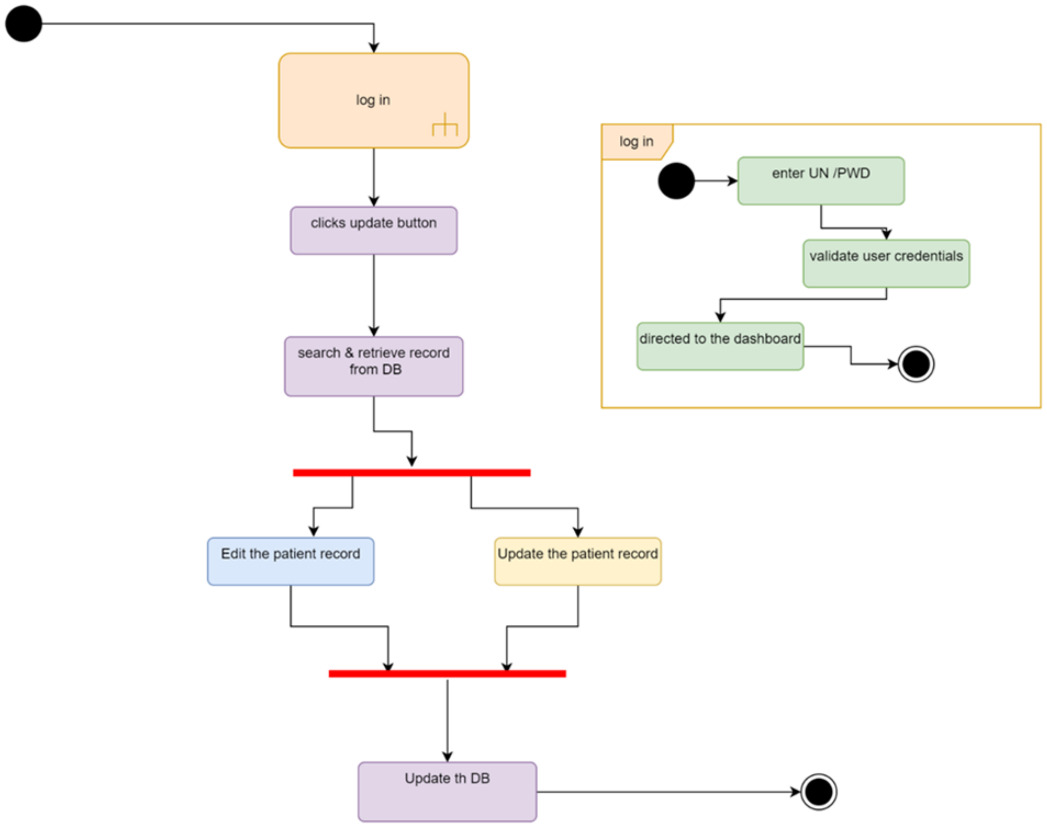


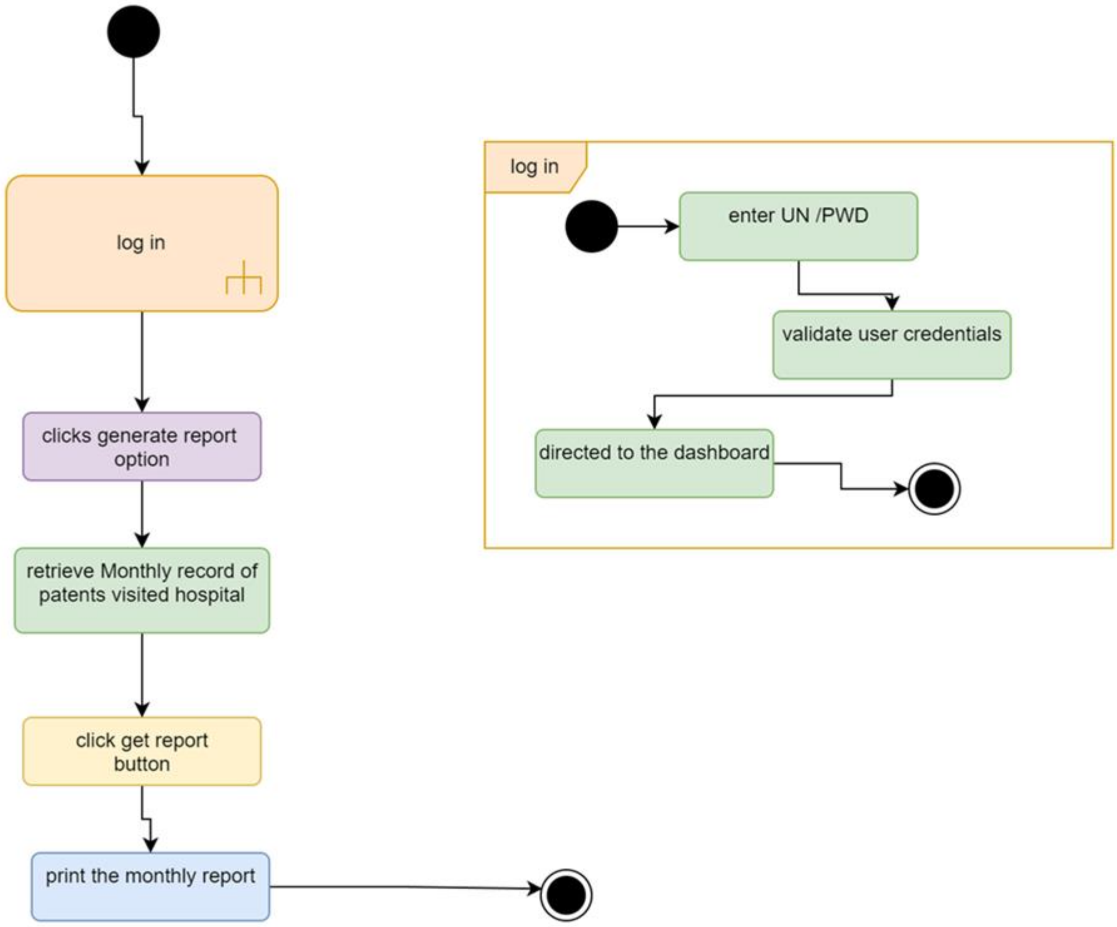
W A M Haren –It22307576 Activity Diagram



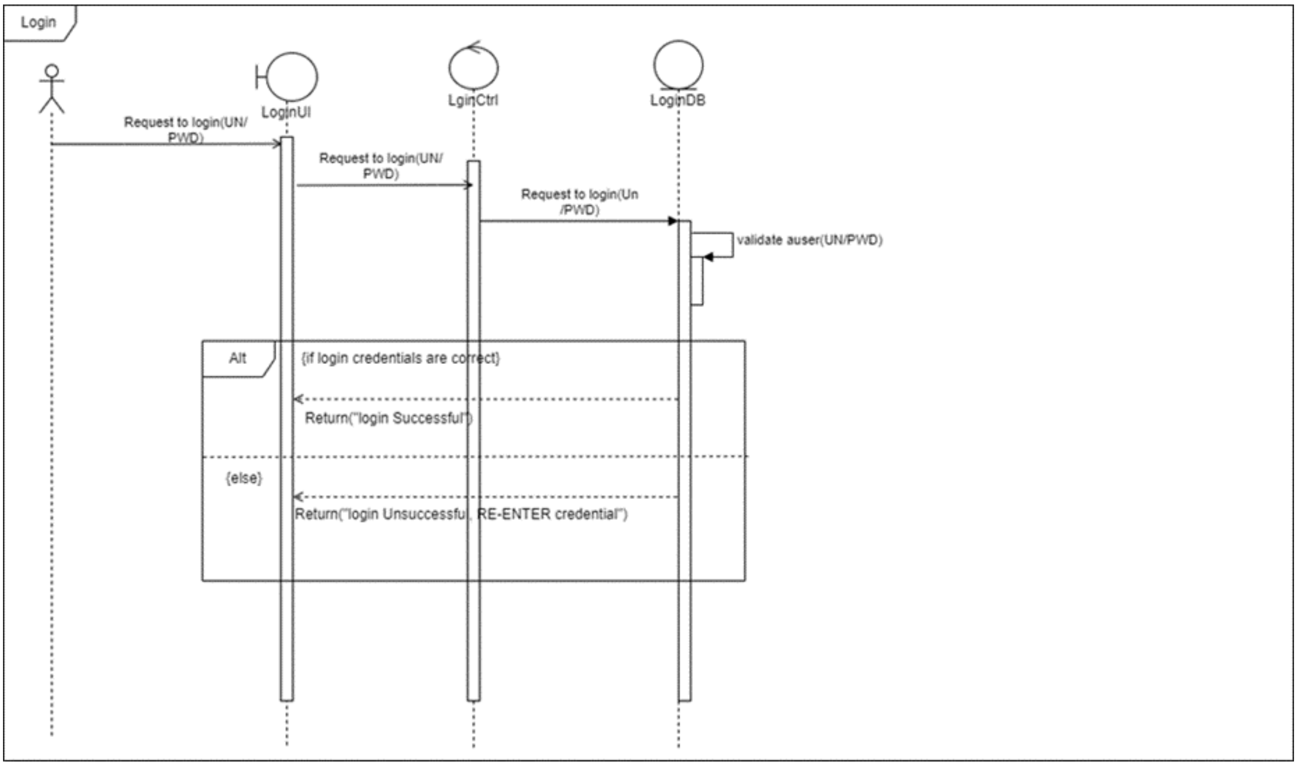



log in
review patient records
[If patient record duplicated/ restricted]
[else]
Delete the duplicated record
System displayed "Record has been deleted"
Update the DB
log in
enter UN /PWD
validate user credentials
directed to the dashboard

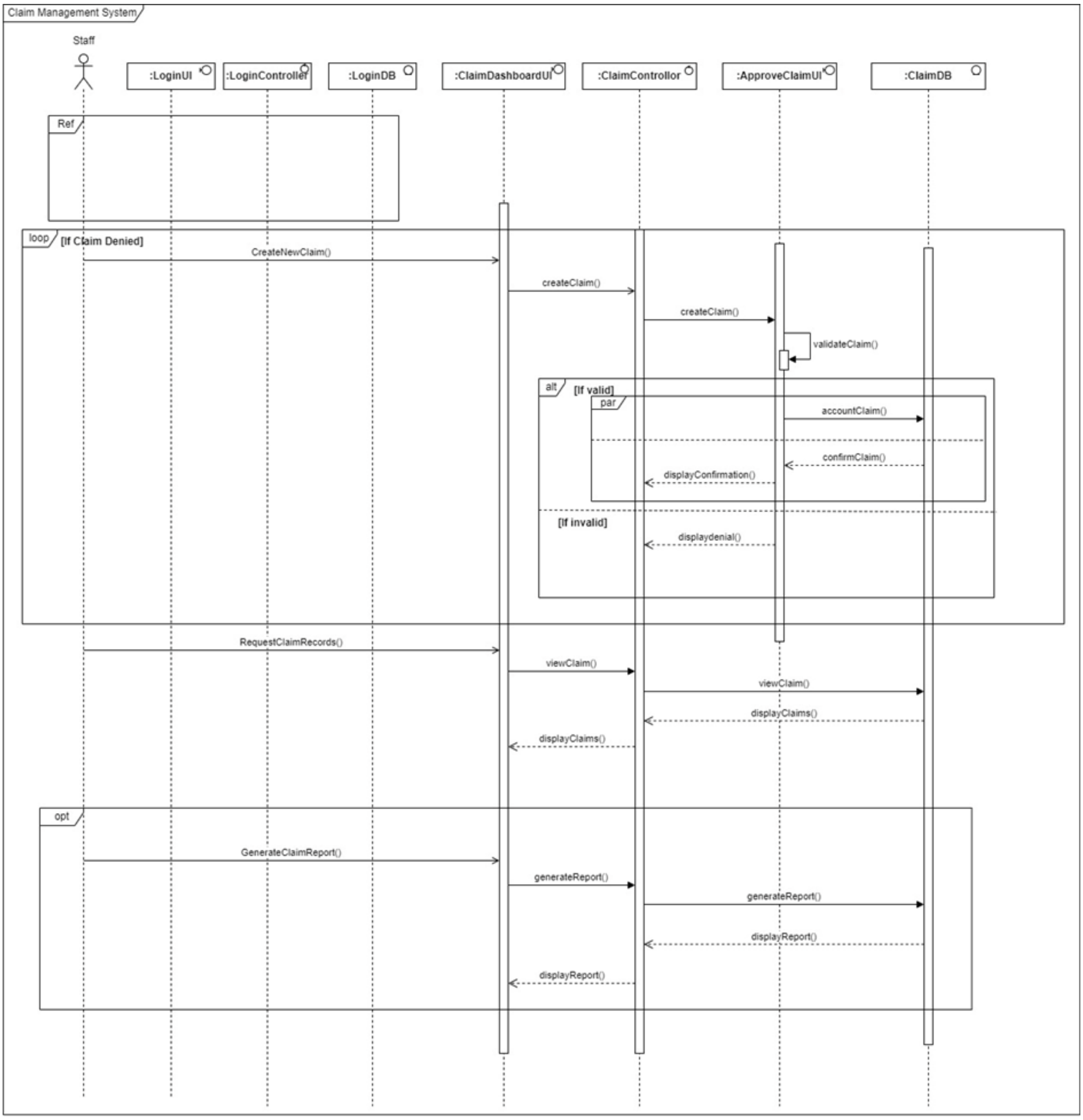

log in
clicks update button
search & retrieve record from DB
Edit the patient record
Update the patient record
Update th DB
log in
enter UN /PWD
validate user credentials
directed to the dashboard


log in
clicks generate report option
retrieve Monthly record of patents visited hospital
click get report button
print the monthly report
log in
enter UN /PWD
validate user credentials
directed to the dashboard

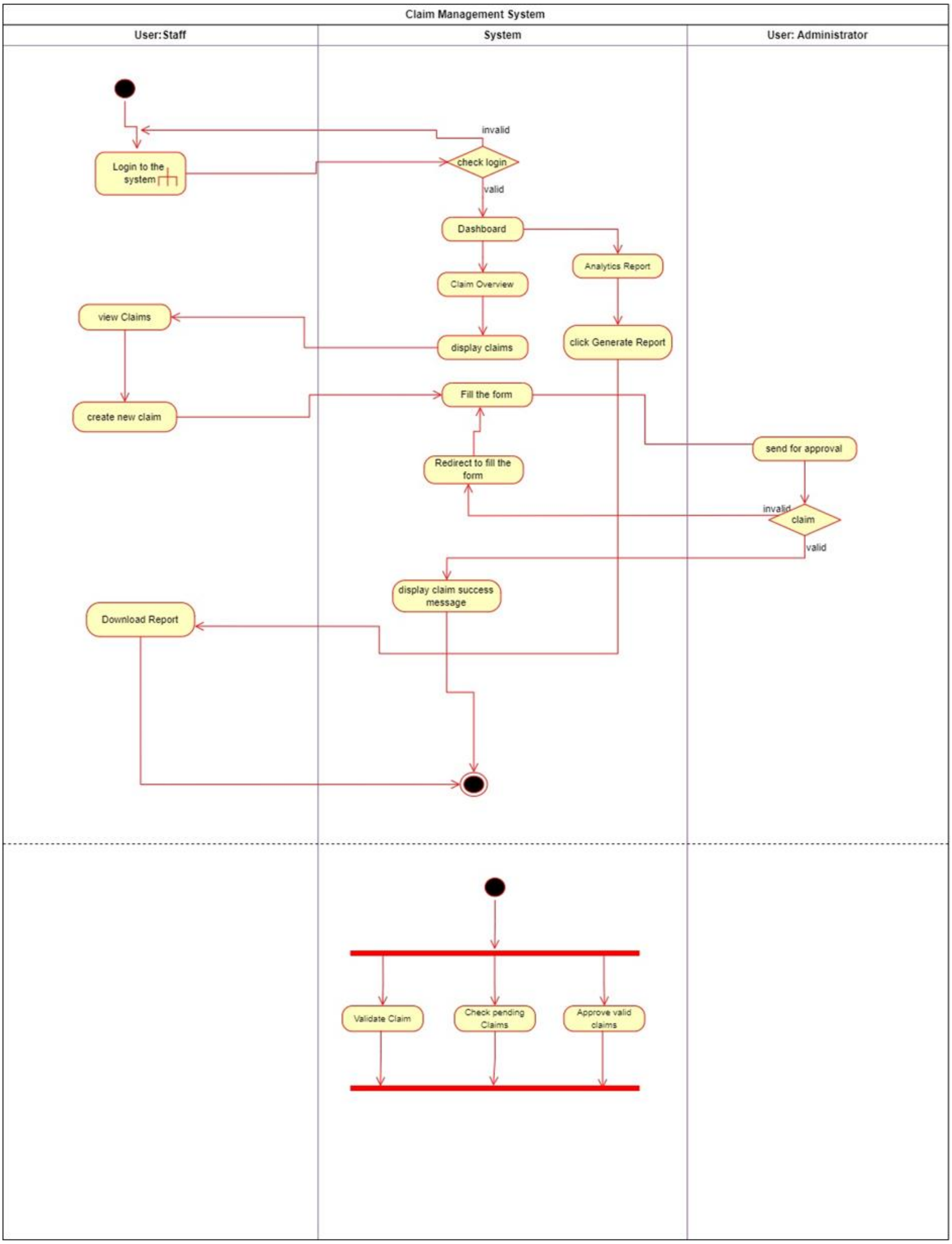
W A M Haren – IT22307576 Sequence diagram



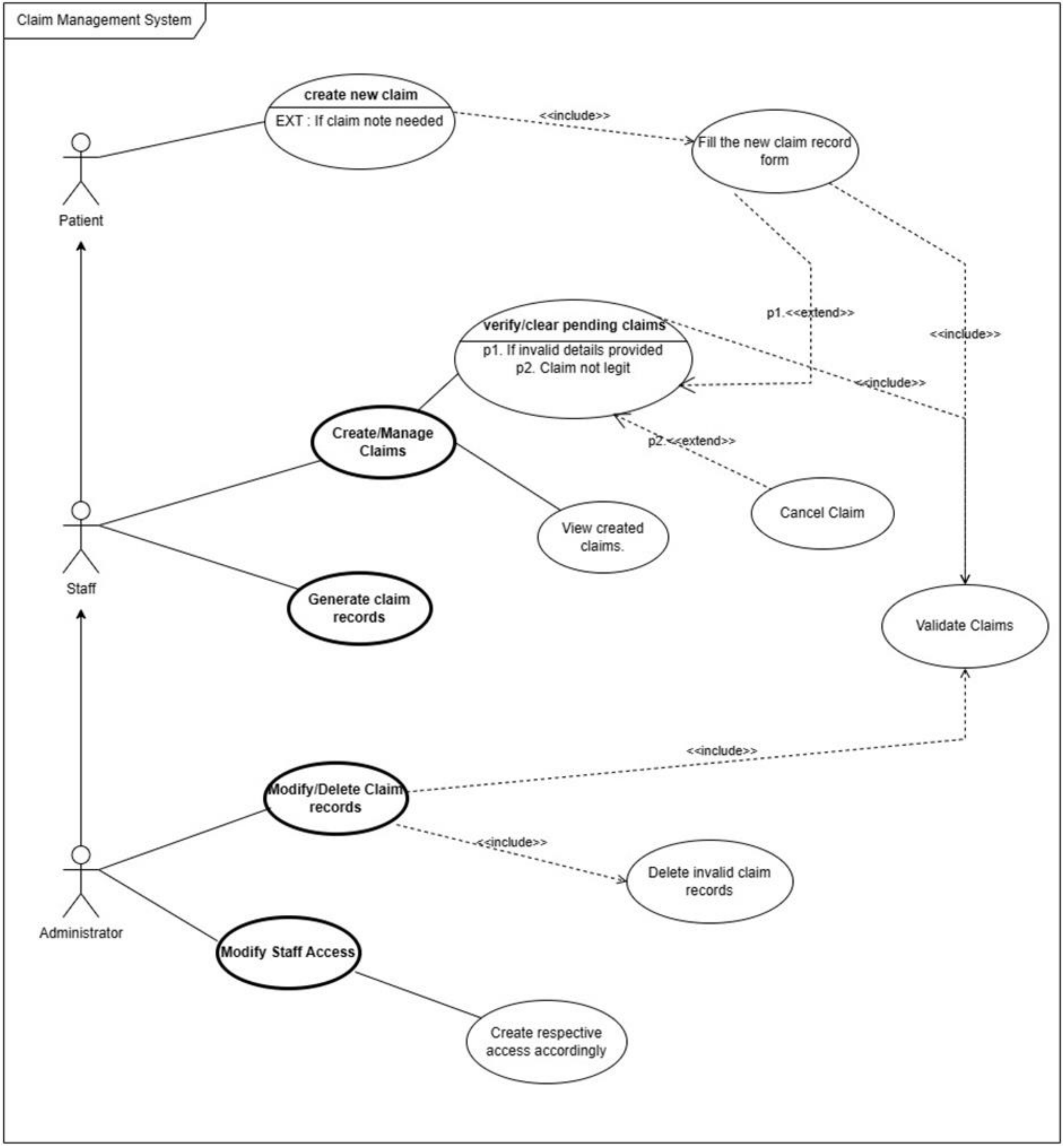
M F M Farsith -Sequence Diagram IT22354556



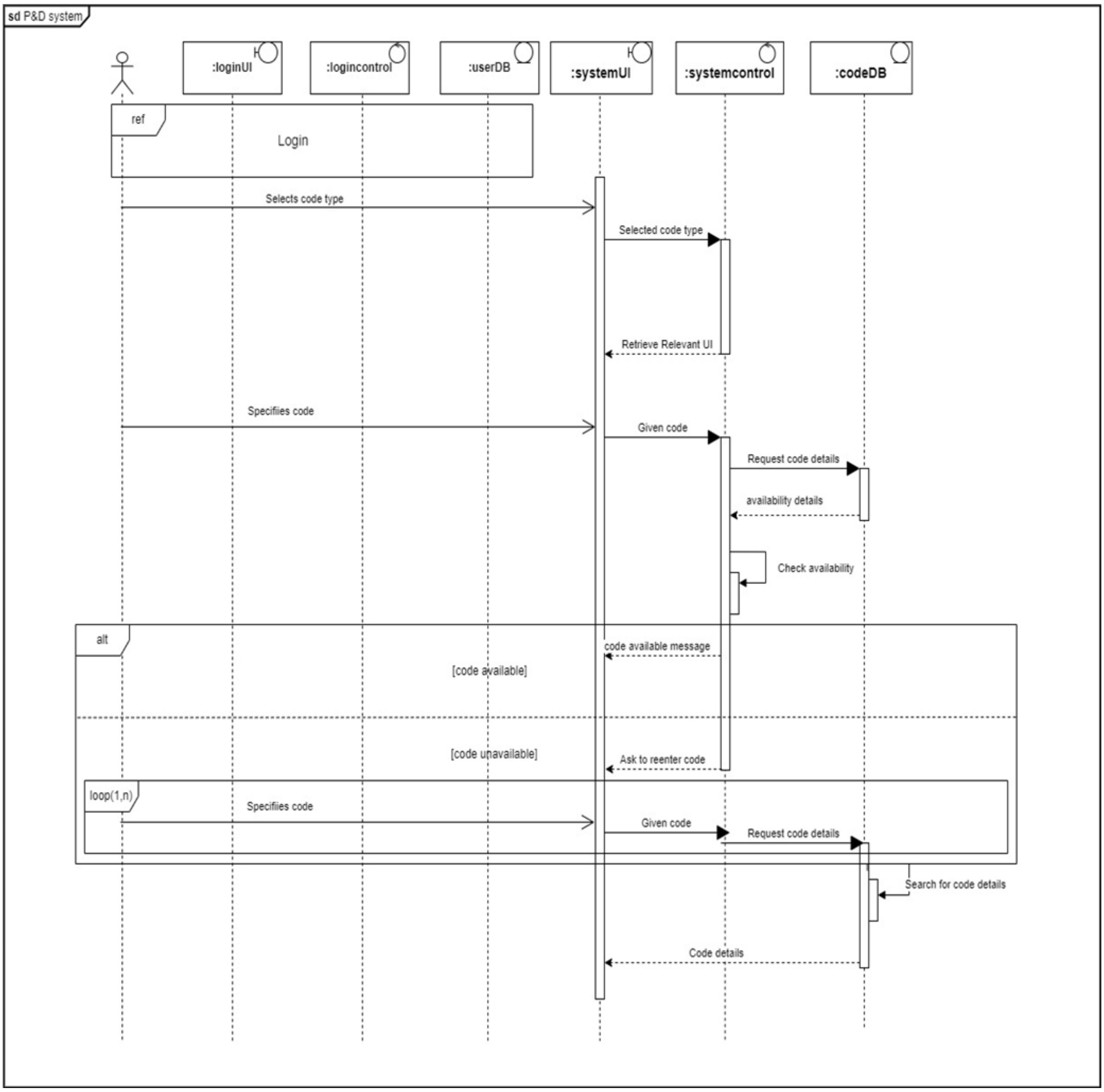
M F M Farsith -Activity Diagram IT22354556



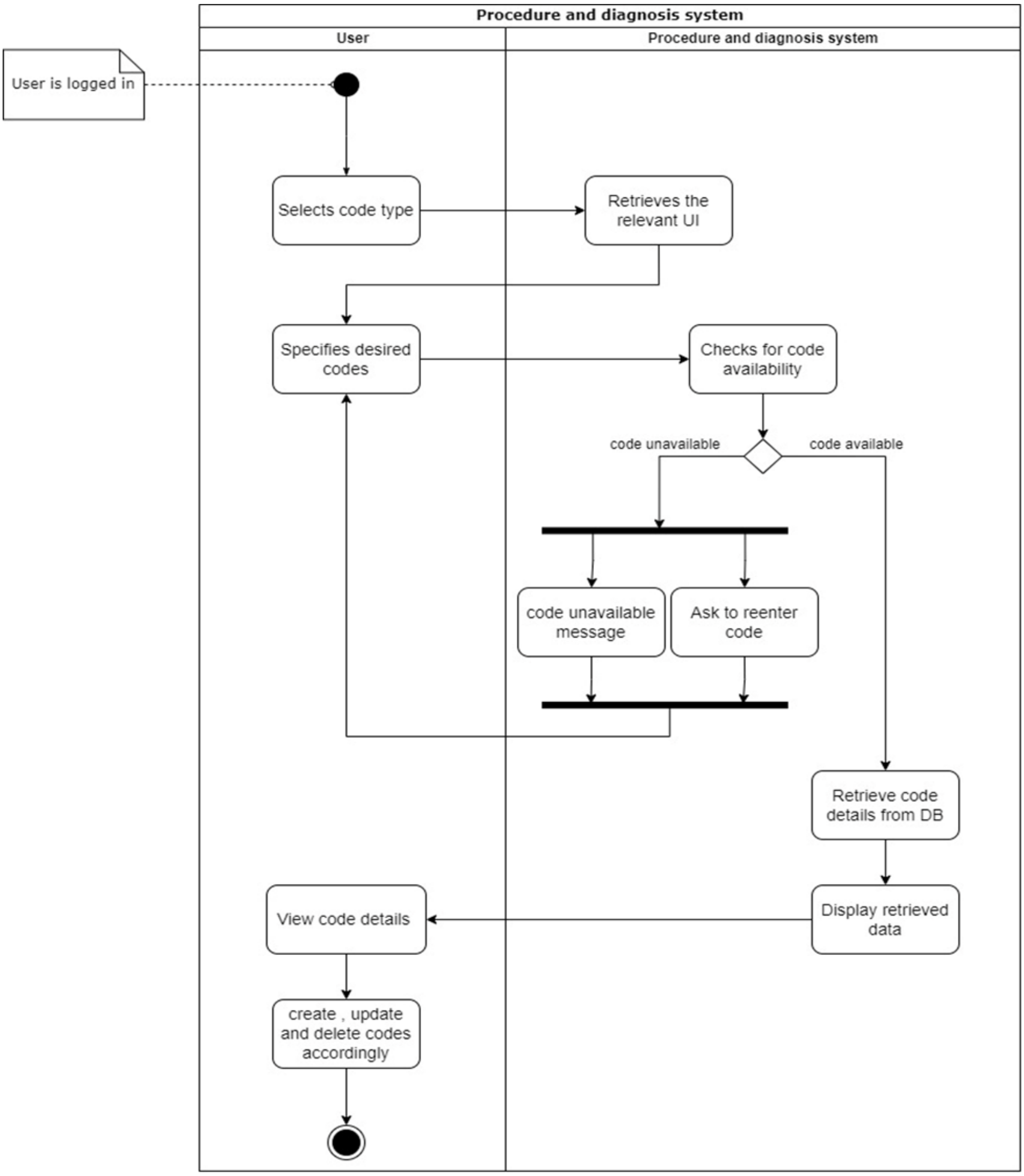
M F M Farsith -Use Case Diagram IT22354556



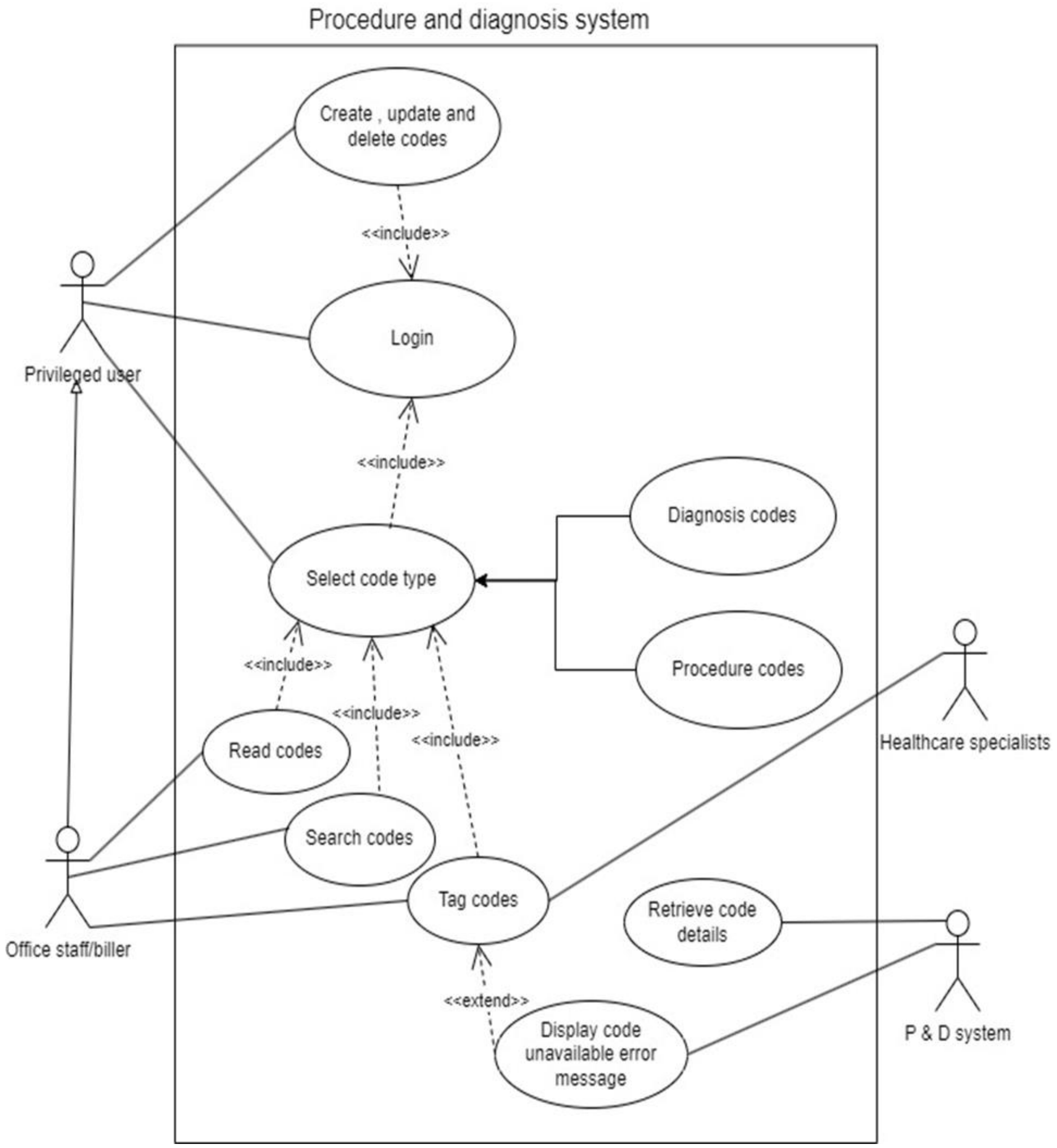
D S I Gamage -Sequence Diagram IT22907516



D S I Gamage -Activity Diagram IT22907516


Procedure and diagnosis system
User
Procedure and diagnosis system
User is logged in
Selects code type
Retrieves the relevant UI
Specifies desired codes
Checks for code availability
code unavailable
code available
code unavailable message
Ask to reenter code
Retrieve code details from DB
Display retrieved data
View code details
create , update and delete codes accordingly

D S I Gamage -Use case Diagram IT22907516



3.2 NON-FUNCTIONAL REQUIREMENTS

Functionality:

The system must meet the users' functional requirements, which may include specific features or capabilities that it must have to meet the user's needs.

Performance:

The system must be able to perform its intended functions in a timely and efficient manner, with fast response times and minimal downtime.

Scalability:

The system must be able to scale to handle increasing amounts of data, users, or transactions without sacrificing performance or stability.

Reliability:

The system must be reliable and able to operate continuously without failure or unexpected downtime.

Compatibility:

The system must be compatible with other systems or software that the user may use, with the ability to exchange data or integrate with other systems, as necessary.

Usability:

The system must be easy to use and navigate, with an intuitive interface that is accessible to users with varying levels of technical expertise.

Cost:

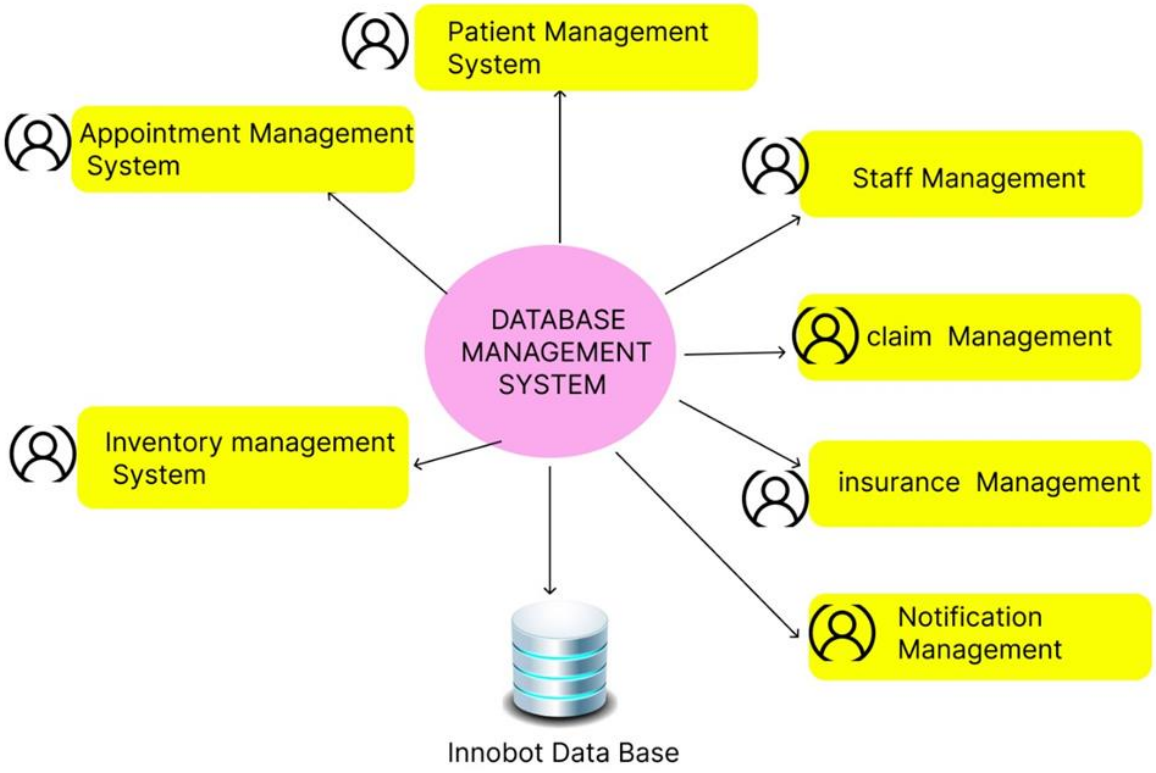
The system must be cost-effective, with a reasonable cost that is commensurate with the value that it provides to the user.

3.3 PERFORMANCE REQUIREMENTS

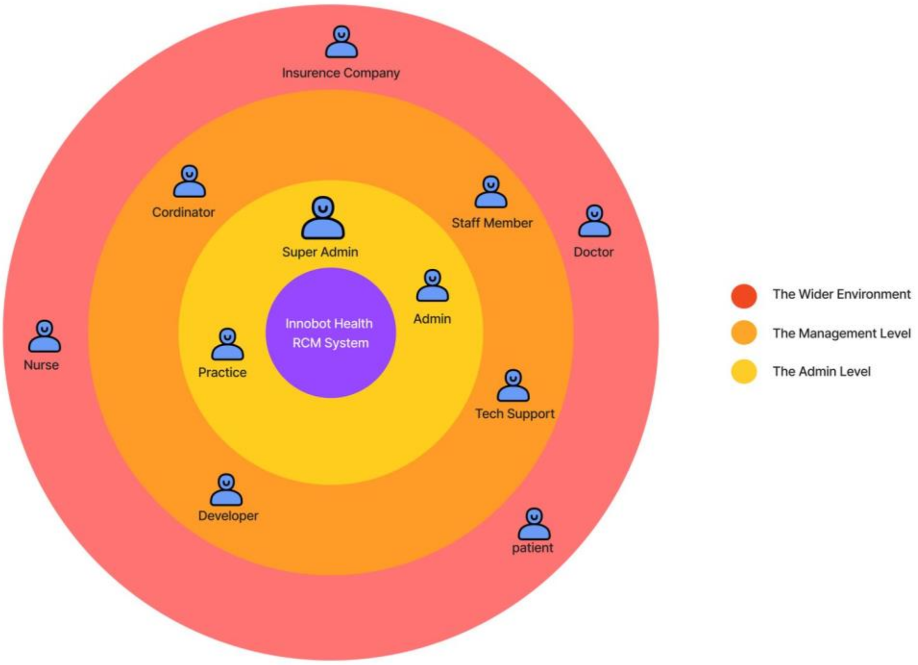
In performance requirements, we consider the request time and response time between the client side and other client-side functions that operate in the browser.

- Components should load within a second.
- Database objects should retrieve the results within 2 seconds.
- User authentications should be done in less than 2 seconds.

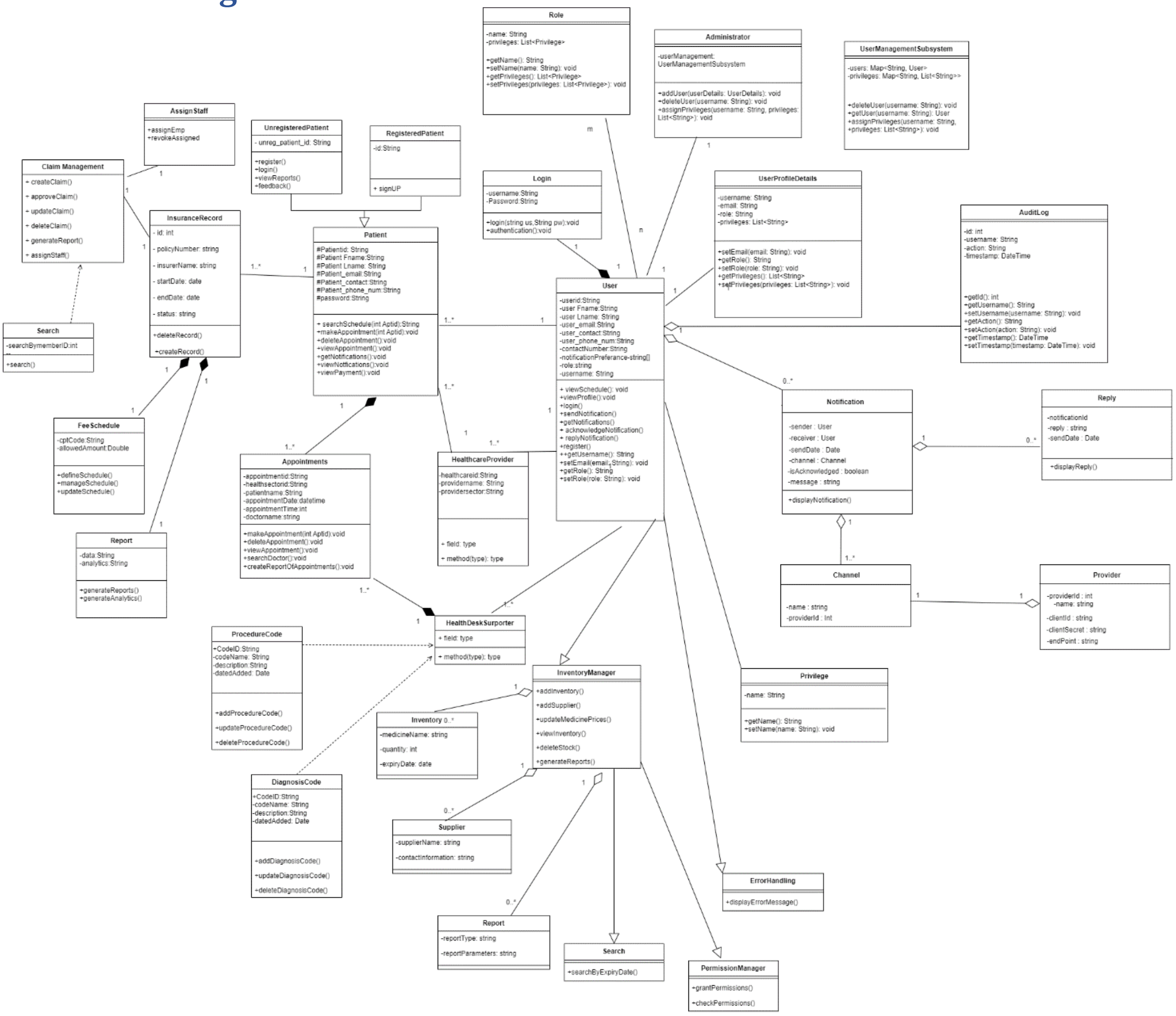
3.4 High-level Architecture Diagram


Patient Management System
Appointment Management System
Staff Management
DATABASE MANAGEMENT SYSTEM
claim Management
Inventory management System
insurance Management
Notification Management
Innobot Data Base

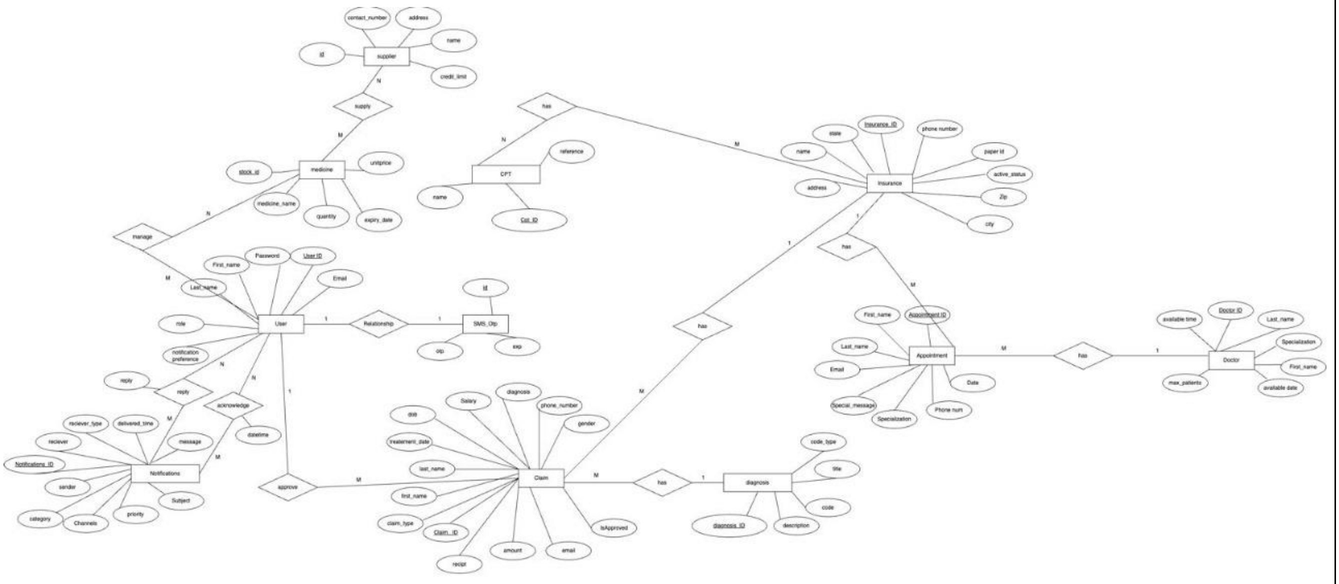
3.5 Onion Diagram



3.6 Class Diagram

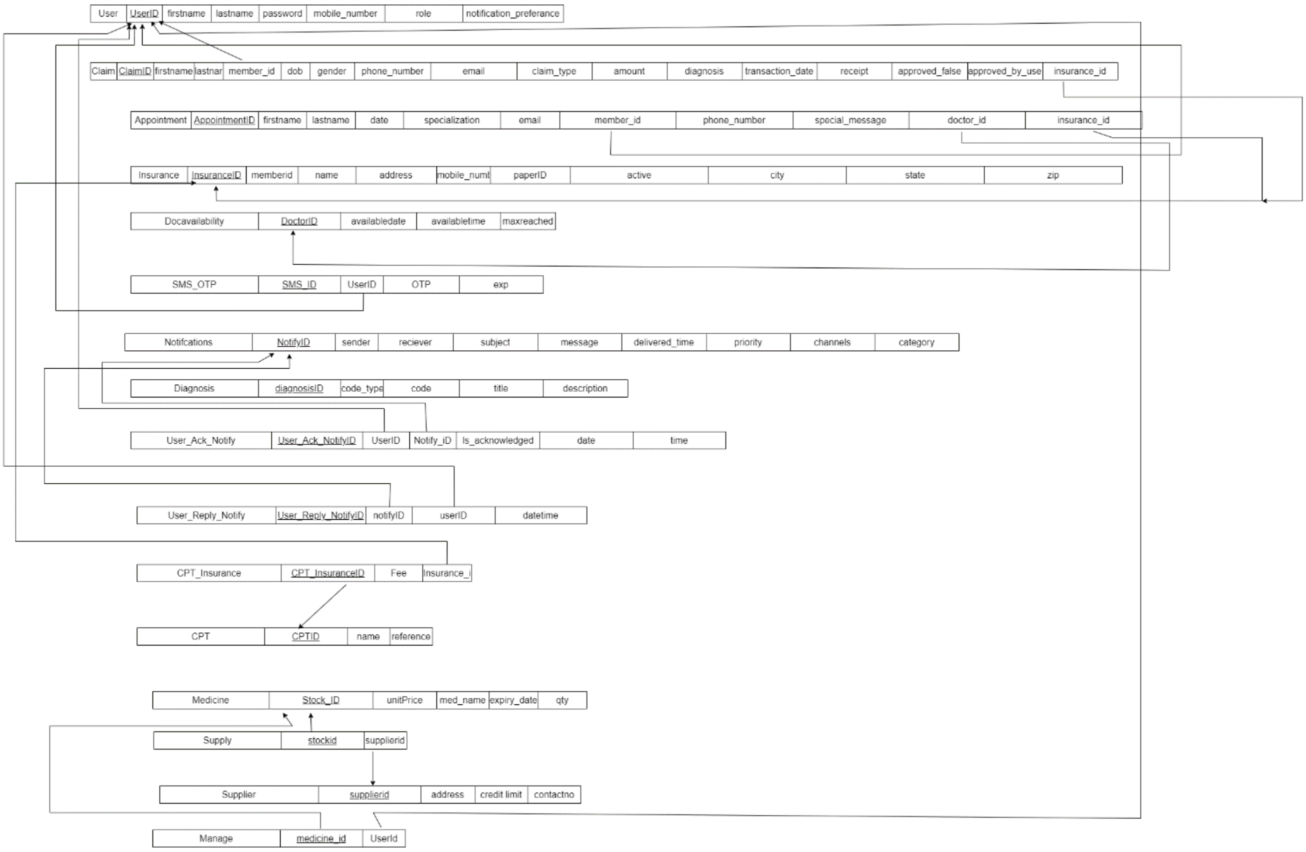


3.7 ER Diagram



3.8 DATABASES

3.8.1 Database schema – Relational Model



User | UserID | firstname | lastname | password | mobile_number | role | notification_preferance

Claim | ClaimID | firstname | lastnar | member_id | dob | gender | phone_number | email | claim_type | amount | diagnosis | transaction_date | receipt | approved_false | approved_by_use | insurance_id

Appointment | AppointmentID | firstname | lastname | date | specialization | email | member_id | phone_number | special_message | doctor_id | insurance_id

Insurance | InsuranceID | memberid | name | address | mobile_numl | paperID | active | city | state | zip

Docavailability | DoctorID | availabledate | availabletime | maxreached

SMS_OTP | SMS_ID | UserID | OTP | exp

Notifcations | NotifyID | sender | reciever | subject | message | delivered_time | priority | channels | category

Diagnosis | diagnosisID | code_type | code | title | description

User_Ack_Notify | User_Ack_NotifyID | UserID | Notify_iD | Is_acknowledged | date | time

User_Reply_Notify | User_Reply_NotifyID | notifyID | userID | datetime

CPT_Insurance | CPT_InsuranceID | Fee | Insurance_i

CPT | CPTID | name | reference

Medicine | Stock_ID | unitPrice | med_name | expiry_date | qty

Supply | stockid | supplierid

Supplier | supplierid | address | credit limit | contactno

Manage | medicine_id | UserId

3.9 DEVELOPMENT ASPECTS

1 PLANNING
2 DESIGN
6 MAINTENANCE
3 DEVELOPMENT
JAVA
CSS3
PHP
HTML5
5 DELIVERY
4 TESTING

Chapter 4 – Testing

4.1 Test Cases & Results

4.1.1 Inventory Management System.

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Medicine Inventory System	
Testing Function: Add New Medicine	
Test Case ID: TC_ADD_001	**Project Name: Test Adding Valid Medicine**
Test Priority: High	
Test Description: This test case verifies that a new medicine can be added to the inventory with all required fields correctly filled out.	
Test Steps: Step 1: Navigate to the 'Add New Medicine' section. Step 2: Enter all required patient details into the form. Step 3: Click the "Save" button to submit the information. Step 4: Check for a confirmation message with the assigned patient ID.	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
ADD_01	Name: Paracetamol Description: Pain reliever and a fever reducer Quantity: 100 Price: $5.00 Supplier Information: ABC Pharmaceuticals	A confirmation message stating "Medicine added successfully with ID: [UniqueID]"	A confirmation message stating "Medicine added successfully with ID: MED987654321"	PASS	Medicine was added successfully, and a unique ID was generated

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Medicine Inventory System	
Testing Function: Update Medicine Inventory	
Test Case ID: TC_UPDATE_001	**Project Name: Test Updating Medicine Quantity**
Test Priority: Medium	
Test Description: This test case checks if the medicine quantity can be updated in the inventory.	
Test Steps: Step 1: Navigate to the 'Update Medicine Inventory' section. Step 2: Search for the medicine by name 'Paracetamol'. Step 3: Select the 'Quantity' field and enter the new quantity. Step 4: Click the 'Update' button and verify the confirmation message.	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
UPD_01	Medicine Name: Paracetamol New Quantity: 150	A confirmation message stating, "Medicine updated successfully."	A confirmation message stating, "Medicine updated successfully."	PASS	Quantity updated correctly in the database.

4.1.3 Inventory Management System.

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Staff Management System	
Testing Function: Adding Staff	
Test Case ID: 1	
Test Priority: High	
Test Description: Verify that staff can be added successfully with all required details, and the data is saved in the MongoDB database.	
Test Steps: a. Step 1 : Fill in all required fields (username, first name, last name, email, role). b. Step 2 : Submit the form to add the staff.	

Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
	Valid values for username, first name, last name, email, and role	Staff should be added to the user list in the front-end. Data should be saved in the MongoDB database	Verify if the staff is displayed in the user list after submission. Check the MongoDB database to ensure the staff data is stored correctly	Pass if the staff is successfully added and data is saved in the database; fail otherwise	Check for error messages if any fields are missing or if the username/email already exists.

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Staff Management System	
Testing Function: Editing Staff	
Test Case ID: 2	**Project Name:**
Test Priority:	
Test Description: Verify that staff details can be edited and saved successfully, and the updated data is reflected in the MongoDB database.	
Test Steps: Step 1: Click on the "Edit" button for a staff member in the user list. Step 2: Modify the staff details. Step 3: Save the changes	

Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
	Modified values for staff details (e.g., username, first name, last name, email, role).	Staff details should be updated in the user list with the modified values. Updated data should be saved in the MongoDB database	Verify if the staff details are updated after saving changes. Check the MongoDB database to ensure the updated data is stored correctly.	Pass if the staff details are successfully updated and data is saved in the database; fail otherwise.	Ensure that the staff details are correctly reflected in the user list and database after editing

<table>
<tr><td colspan="2"><b>Project ID:</b> <b>TP_2024_Y2_S2_WE10</b></td></tr>
<tr><td colspan="2"><b>Project Name: Staff Management System</b></td></tr>
<tr><td colspan="2"><b>Testing Function: Deleting Staff</b></td></tr>
<tr><td><b>Test Case ID: 3</b></td><td><b>Project Name:</b></td></tr>
<tr><td><b>Test Priority: High</b></td><td></td></tr>
<tr><td colspan="2"><b>Test Description:</b> Verify that staff can be deleted from the system, and the deleted data is removed from the database.</td></tr>
<tr><td colspan="2"><b>Test Steps:</b>
Step 1: Click on the "Delete" button for a staff member in the user list.</td></tr>
<tr><td colspan="2"></td></tr>
</table>

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
	Selection of a staff member to delete.	Staff details The selected staff member should be removed from the user list. Deleted data should be removed from the MongoDB database.	Verify if the staff member is no longer present in the user list after deletion. Check the MongoDB database to ensure the deleted data is removed. .	Pass if the staff member is successfully deleted and data is removed from the database; fail otherwise	Ensure that the deletion process does not cause any unexpected behavior or errors.

4.1.4 Appointment Schedule Management System

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Appointment Scheduling Management System	
Testing Function: Add an Appointment to the System	
Test Case ID: 1	**Project Name:**
Test Priority:	High
Test Description: Add an Appointment to a Doctor in a relevant health sector registered in the system.	
Test Steps: Step 1: Start from the homepage, the central hub of the system Step 2: navigate to the dashboard for an overview of Healthsectors. Step 3:Enter the patient details and the appointment details. Step 4: Display Overview of the available Doctors, Time and Dates Step 5: Find healthcare providers based on specialty or location. Step 6: Confirm the Appointment	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
1	First name Last name Date Specialization email member_id phone_number special_message doctor_id	The Appointment added must be saved to the database, and later displayed with the doctor selected and patient details entered as Appointment Scheduled.	Expected Output	Pass if Appointment details added successfully to DB, else Fail	Adding new appointment function worked

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Appointment Scheduling Management System	
Testing Function:Update Appointment Scheduled	
Test Case ID: 2	**Project Name:**
Test Priority:	High
Test Description: Update a Scheduled appointment.	
Test Steps: Step 1: Start from the homepage, the central hub of the system Step 2: navigate to the dashboard for an overview of Healthsectors. Step 3: Enter the patient details and the appointment details. Step 4: Display Overview of the available Doctors, Time and Dates Step 5: Find healthcare providers based on specialty or location. Step 6: Confirm the Appointment. Step 7. Update the Appointment.	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
2	First name Last name Date Specialization email member_id phone_number special_message doctor_id	The Appointment added must be saved to the database,and should be able to update the appointment details.	Expected Output	Pass	Adding new appointment function worked

4.1.6 Patient Management System

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Patient Management System	
Testing Function: Add Patient	
Test Case ID: TC_ADD_01	**Project Name: Test Add New Patient**
Test Priority: High	
Test Description: This test case checks if a new patient can be added to the system with all required information.	
Test Steps: Step 1: Navigate to the "Add Patient" section. Step 2: Enter all required patient details into the form. Step 3: Click the "Save" button to submit the information. Step 4: Check for a confirmation message with the assigned patient ID	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
ADD_01	Name: John Doe ID: (Should be auto-generated) Address: 123 Health St. Phone Number: 555-1234 Insurance Information: HealthInsure Co. Policy #98765	A confirmation message saying "Patient successfully added with ID: [UniquePatientID]"	A confirmation message saying "Patient successfully added with ID: P123456"	PASS	The patient was successfully added, and a unique ID was generated and displayed.

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Patient Management System	
Testing Function: Delete Patient	
Test Case ID: TC_DEL_01	**Project Name: Test Delete Existing Patient**
Test Priority: Medium	
Test Description: This test case verifies that an existing patient can be deleted from the system after confirmation.	
Test Steps: Step 1: Use the search function to find the patient with ID P123456. Step 2: Select the patient record found. Step 3: Click the "Delete" button. Step 4: Confirm the deletion when prompted.	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
DEL_01	Patient ID: P123456	A confirmation message saying "Patient record deleted successfully."	A confirmation message saying "Patient record deleted successfully."	PASS	The patient record was successfully deleted after confirmation

4.1.7 Claim Management System

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Claim Management System	
Testing Function: Create Claim	
Test Case ID: 1	**User : Staff**
Test Priority: High	
Test Description: Create Claim for a Patient	
Test Steps: Step 1: Login to the system Step 2: Enter the Claim Overview Interface Step 3: Click on the “Create Claim” buttom Step 4: Click the agreement checkbox and create claim. Step 5: Confirm by getting the prompt Modal.	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
ClaimTest01	MemberId, FirstName, LastName, DOB, Amount, Receipt	Confirmation Modal appearing and details stored in DB	Confirmation Modal appearing and details stored in DB	Pass	Checking on the possibility of constructing a Modal to be printed.

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Claim Management System	
Testing Function: Approve Claim	
Test Case ID: 2	**User : Staff**
Test Priority: High	
Test Description: Approve Pending Claims for Patients	
Test Steps: Step 1: Login to the system Step 2: Enter Approve Claim Interface Step 3: Check on the unapproved claims. Step 4: Validate the claim by double verification Step 5: Approve the claim and notify the patient	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
Approve Claim Test01	MemberId, ClaimId, Name, Amount, Receipt	Change of the claim status in the overview interface after the approval.	Change of the claim status in the overview interface after the approval.	Fail	Checking on the progress of the current approval and verification methods.

4.1.8 Insurance Management System

Project ID: **ITP_2024_Y2_S2_WE10**					
Project Name: Health Care Management System “Innobot Health”					
Testing Function: Create New Insurance Record					
Test Case ID: 1			**Test Case Designed By,** **ID: IT22905840** **Name:** Chamikara M G S		
Test Priority:			High		
Test Description: verifies the functionality to create a new insurance record.					
Test Steps: Step 1: Login to the system Step 2: Go to the Create New Record function Step 3: Input valid insurance record details Step 4: Call the Create New Insurance Record() function					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
1	Insurance Provider Coverage Details Effective Date Expiry Date Premium Amount	Display “Insurance record create successfully”	Expected output	pass	

Project ID: **ITP_2024_Y2_S2_WE10**	
Project Name: Health Care Management System “Innobot Health”	
Testing Function: Delete/Archive Insurance Record ()	
Test Case ID:2	**Test Case Designed By,** **ID: IT22905840** **Name:** Chamikara M G S
Test Priority:	High
Test Description: Verify that an existing insurance record can be deleted or archived successfully	
Test Steps: Step 1: Call the delete/Archive Insurance Record() function Step 2: Provide the ID of the insurance record need to delete Step 3: Confirm the deletion/archival action when prompted Step 4: Verify that the insurance record is successfully deleted or archived	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
2	Insurance Record ID	Display "Insurance record deleted/archived successfully"	Expected output	pass	delete/archive function worked.

4.2 Notification Management System

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Notification Management System	
Testing Function: Send Notifications	
Test Case ID: 1	**Project Name: Send new notification**
Test Priority: 1	
Test Description: create a new notification and send	
Test Steps: Step 1: log into the system by using email and password Step 2: Navigate to the notifications section and tap on '+create new' Step 3: Fill the forms as per to the validations. Step 4: Click and 'send notification'	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
1	Category:custom Receiver_type: Doctor Receiver: all Subject: Test bulk notification Message: Test Notification Annonuynous: not selected Priority: High Scheduled/Instant: Instant	202 response status, and the notificatin should be sent to all the doctors available in the system instantly.	The notification has been sent to the doctors based on their notification preference.	PASS	The record has also been created in the database successfuly.

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Notification Management System					
Testing Function: Receive Notifications					
Test Case ID: 2			**Project Name: Receive Notifications**		
Test Priority: 2					
Test Description: Users receive system notifications once logged in.					
Test Steps: Step 1: Log into the system with email and password Step 2: Click on the notification icon in the top bar.					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
1	User's access token	The system notifications received by the user should be listed down.	The notifications received by the logged in user listed down with acknowledge and reply to buttons.	PASS	User's id has been embeded in the JWT access token and BE system authorized the user by using that.

4.2.1 Procedure & Diagnosis Management System

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Procedure and Diagnosis System	
Testing Function: Add New Code	
Test Case ID: TC_ADD_001	**Project Name: Test Adding Valid Codes**
Test Priority: High	
Test Description: This test case verifies that a new code can be added to the inventory with all required fields correctly filled out.	
Test Steps: Step 1: Navigate to the 'Add New Code' section. Step 2: Enter all required patient details into the form. Step 3: Click the "Save" button to submit the information. Step 4: Check for a confirmation message with the assigned patient ID.	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/ Fail)**	**Comments**
ADD_01	Name: Paracetamol Description: Pain reliever and a fever reducer Quantity: 100 Price: $5.00 Supplier Information: ABC Pharmaceuticals	A Confirmation message stating "Code Added successfully with ID: [UniqueID]"	A confirmation message stating "Code added successfully with ID:MED987654 321"	PASS	Code was added successfully, and a unique ID was generated

Project ID: **TP_2024_Y2_S2_WE10**	
Project Name: Procedure and Diagnosis System	
Testing Function: Update Code	
Test Case ID: TC_UPDATE_001	**Project Code Name: Test Updating**
Test Priority: Medium	
Test Description: This test case checks if the code type can be updated in the inventory.	
Test Steps: Step 1: Navigate to the 'Update Code ' section. Step 2: Search for the code by name 'ACL255'. Step 3: Select the 'Type' field and enter the new type. Step 4: Click the 'Update' button and verify the confirmation message.	

Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
UPD_01	Code Name: ACL255 New type: Diagnosis	A confirmation message stating, "Code updated successfully."	A confirmation message stating, "Code updated successfully."	PASS	Quantity updated correctly in the database.

Chapter 5 – Evaluation & Conclusion

The implementation of the Innobot Hospital Management System marks a significant milestone in the evolution of healthcare administration at our institution. This evaluation and conclusion segment will offer a comprehensive assessment of the system's performance, its impact on hospital operations, and the overarching benefits it has conferred upon our organization.

The Innobot Hospital Management System stands out as a sophisticated solution tailored to meet the complex demands of modern healthcare facilities. With its extensive suite of features, the system addresses various facets of hospital management, including patient registration, appointment scheduling, medical records management, billing, and inventory control. Its seamless integration with medical devices and electronic health records (EHR) systems further enhances its utility and interoperability within our healthcare ecosystem.

At Innobot Hospital, the implementation of this advanced management system has yielded remarkable results. Its robust functionality, intuitive interface, and reliability have significantly elevated operational efficiency and patient care standards. The system efficiently manages patient information, streamlines appointment scheduling processes, and facilitates accurate billing and invoicing. Moreover, its real-time reporting capabilities enable data-driven decision-making, thereby fostering improved resource allocation and clinical outcomes.

To ensure the sustained effectiveness and relevance of the Innobot Hospital Management System, proactive measures must be undertaken. Regular updates and system maintenance are imperative to address any identified issues promptly and to incorporate advancements in healthcare technology. Additionally, soliciting feedback from healthcare professionals, administrative staff, and patients will provide invaluable insights for ongoing system enhancements and optimization.

In conclusion, the Innobot Hospital Management System has emerged as a cornerstone in revolutionizing hospital administration at Innobot Hospital. Its implementation has catalyzed a paradigm shift towards greater efficiency, accuracy, and patient-centricity in our operations. By seamlessly integrating with existing workflows and leveraging cutting-edge technology, the system has become an indispensable tool for optimizing resource utilization, enhancing patient experiences, and ultimately improving healthcare outcomes. Through its continued evolution and innovation, the Innobot Hospital Management System reaffirms our commitment to excellence in healthcare delivery and organizational advancement.

5.1 References

Visual Studio Code, [Online]. Available: https://code.visualstudio.com/docs.

React JS, [Online]. Available: https://reactjs.org/

React Validation [onlie] .Available: https://react-hook-form.com/get-started

MongoDB, [Online]. Available: https://www.mongodb.com/atlas/database.

NPM packege manger [Online]. Available:https://www.npmjs.com/

Maven Repository [Online]. Available. https://mvnrepository.com/

"What is agile methodology? Modern software development explained," 06 Apr 2022.
[Online].
Available: https://www.infoworld.com/article/3237508/what-is-agile-methodology-modernsoftware-development-explained.html.
Spring[Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
Build Rest Api [Online] Available : https://hevodata.com/learn/spring-boot-rest-api/
Docker [Online] . Availble : https://docs.docker.com/
Deployed server . Availble : https://www.docker.com/blog/how-to-deploy-containers-to-azure-aci-using-docker-cli-and-compose/
"What is Mongo DB?," MongoDB, [Online]. Available: https://www.mongodb.com/what-ismongodb.
"What is a Hospotal Management System?," [Online]. Available:
https://uk.indeed.com/career-advice/career-development/hospitality-management-system#:~:text=Hospitality%20management%20systems%20help%20hotels,
competitive%20travel%20and%20tourism%20industry.
"Gym Management Software," g2.com, [Online]. Available:
https://www.g2.com/search?utf8=%E2%9C%93&query=hospital+Management+Software