WACHEMO UNIVERSITY

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF SOFTWARE ENGINEERING

Project title: Stock Management System

Group members

<u>**Name**</u>	<u>**ID**</u>
Eyobed Solomon……..……………...	WCU172510
Fetiha Oumer………………..............	WCU171211
Minase Mengesha…………………	WCU17D6911
Ruhama Alemu…………..……….	WCU172019
Yamilaksira Ashenafi……………...	WCU17D4021

Submission date:17/04/2018E.C

ACKNOWLEDGEMENT

First and foremost, we thank God for the strength, guidance, and wisdom granted throughout the development of this project. We extend our sincere gratitude to our instructors for their guidance, feedback, and support, which helped shape this work in the right direction. We also thank Kidemu Mini Market for allowing us to collect real-world information that made this project practical and relevant. Finally, we appreciate everyone who supported and encouraged us during the completion of this project.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ACRONYM

WCU – Wachemo University

E.C – Ethiopian Calendar

ID – Identification Number

DB – Database

SQL – Structured Query Language

OS – Operating System

UI – User Interface

UML – Unified Modeling Language

E-R – Entity Relationship

ERD – Entity Relationship Diagram

PK – Primary Key

FK – Foreign Key

CRUD – Create, Read, Update, and Delete

1NF – First Normal Form

2NF – Second Normal Form

3NF – Third Normal Form

BCNF – Boyce–Codd Normal Form

4NF – Fourth Normal Form

5NF – Fifth Normal Form

UAT – User Acceptance Testing

POS – Point of Sale system

EXECUTIVE SUMMARY

This project is about designing a Stock Management System for Kidemu Mini Market, a small retail shop that currently uses notebooks and papers to manage products, stock, sales, and expiry dates. The manual system makes daily work difficult and slow. It often causes mistakes such as wrong stock counts, missed expiry dates, and incorrect sales calculations. Preparing reports also takes a lot of time and effort.

To solve these problems, this project proposes a simple computer-based stock management system. The new system stores all product and stock information in a database. It allows the Admin to add and update products, and record sales. The system updates stock levels after each sale and gives alerts when products are low in stock or close to expiry.

The system supports three user roles: Admin, Owner, and Accountant. Each user has limited access based on their responsibility. This improves security and keeps the system organized. The Accountant can easily view income and cost without changing any stock data.

To design the system clearly, different UML diagrams were used, including use case diagrams, activity diagrams, sequence diagrams, class diagrams, and an ER diagram. These diagrams help explain how the system works and how data flows inside it. The database design ensures that data is stored safely and correctly.

The proposed system reduces human errors, saves time, improves accuracy, and makes business management easier. It is simple to use and suitable for small businesses like Kidemu Mini Market. Overall, this project provides a strong foundation for implementing an effective and reliable stock management system.

CHAPTER ONE

INTRODUCTION

1.1 Background of the Organization

Kidemu Mini-market is an owner-operated retail establishment located in Hosana around Bezabh Petros square, Ethiopia. Founded and managed by Kidmu Nasru, the business has served the local community since 1992E.C as mini store. The enterprise currently operates by the owner, who oversees all daily operations.

Kidemu Mini Market currently depends on a fully manual inventory system. The owner records products, sales, expiry dates, and stock levels inside notebook and also slip. To check availability, he manually checks shelves and estimates what is running out. This manual approach slows down decision-making and easily leads to errors such as missing expiry dates, miscounting stock, or failing to notice low-stock items.

As the number of products grows, the notebook-based method becomes unreliable. There is a need for a simple digital system that can track products, update stock automatically after sales, and generate quick reports.

1.2 Statement of the Problem

Kidemu Mini Market currently uses a manual notebook-based system to manage product stock, sales records, and expiry dates. Although the shop is small, this manual approach creates serious operational and management challenges.

The main problems include:

- Frequent calculation errors in stock quantities and sales records due to handwritten entries
- Lack of automatic tracking for product expiry dates, leading to expired items remaining unnoticed
- No real-time stock balance, making it difficult to know exact quantities at any time
- Difficulty in tracking sales, cost, especially over long periods
- Manual report preparation, which is time-consuming, inconsistent, and prone to mistakes

A digital stock management system will eliminate these problems and improve accuracy and efficiency.

1.3 Objective of the project

1.3.1. General Objective

To design and develop a digital stock management system to replace the manual management system used at Kidemu Mini Market.

1.3.2 Specific Objectives

Identify and document functional and non-functional requirements of the existing manual stock management system.

Analyze the limitations and inefficiencies of the current notebook-based stock management process.

Design a proposed stock management system using appropriate UML models such as use case, activity, sequence, class, ER, state chart, component, and deployment diagrams.

Design a logical database structure that represents system entities, relationships, and constraints accurately.

Design a physical database schema that ensures data integrity, consistency, and efficient storage.

Develop a stock management system that supports product management, stock tracking, sales recording, alert generation, and reporting.

Implement role-based access for Admin, Owner, and Accountant according to system responsibilities.

Test system functions to verify correctness of stock updates, alerts, and report generation.

Prepare clear technical documentation to support future system maintenance and enhancement.

1.4. Methodologies

1.4.1 Requirement Elicitation Methodology

To understand the needs of Kidemu Mini Market, the following methods were used:

Observation: Directly observing how the Owner records stock, checks shelves, and prepares reports.

Interview: Asking the Owner about challenges with the current system.

Document Review: Examining the existing notebook and papers used for stock records, sales, and expiry information.

Problem Analysis: Identifying issues in accuracy, speed, and reliability based on gathered information.

These methods ensured that the system requirements reflect real shop operations.

1.4.2 Requirement Analysis and Modeling

Requirement analysis for the project was conducted by studying the existing manual stock management process at Kidemu Mini Market and identifying system users, operations, and data requirements. The gathered requirements were then translated into system models that clearly describe system behavior, structure, and data flow.

To achieve this, several diagrams were prepared based on the proposed system.

Use Case Diagram

The use case diagram models the interactions between the system and its users: Admin, Owner, External payment system, and Accountant. It identifies all major system functions such as product management, stock handling, sales recording, alert viewing, and report generation, while also defining access boundaries for each user role.

Activity Diagrams

Activity diagrams were developed to represent the internal workflow of major system processes such as adding products, updating product information, deleting products, recording expired items, and viewing alerts. These diagrams show the sequence of actions and decision points involved in completing each task.

Sequence Diagrams

Sequence diagrams were created to illustrate how system components interact during operations like updating products, generating reports, recording expired items, viewing products, and viewing alerts. They show the order of messages exchanged between the user interface, control logic, and database.

Class Diagram

The class diagram defines the static structure of the system by identifying key classes, their attributes, operations, and relationships. It supports core functions such as stock updates, sales recording, report generation, and user management, and aligns with the database design.

Entity Relationship Diagram (ERD)

The ERD models the logical database structure of the system. It defines entities such as Product, Stock, User, Transaction, Report, Supplier, and Order, along with their relationships. This diagram ensures proper data organization and supports accurate stock tracking and reporting.

State Chart Diagram

The state chart diagram represents the lifecycle of a product within the system. It shows how a product transitions through states such as Registered, In Stock, Low Stock, Expired, and Out of Stock based on system events and conditions.

Component Diagram

The component diagram illustrates the high-level system modules and how they interact. It shows components such as the User Interface module, Business Logic module, and Database module, helping to visualize system organization.

Deployment Diagram

The deployment diagram describes how the system will be physically deployed. It shows the relationship between the application, database, and the hardware environment, including the user's computer and database server.

Together, these models provide a complete and consistent representation of the proposed system and serve as a blueprint for implementation.

1.4.3 System Implementation Methods

The system will be implemented using a simple, structured, and iterative approach, based on the tools and programming concepts learned during the course. Development will be carried out step by step, where each part of the system is implemented, tested, and refined before moving to the next.

Implementation Approach

The system will be developed as a web-based application using C++.

A SQL-based database will be used to store products, stock details, users, transactions, and reports.

The implementation will strictly follow the requirements, system models, and diagrams prepared during the analysis and design phase.

Development will follow an iterative approach, allowing changes and improvements based on testing and feedback.

Database Implementation

Create database tables based on the ER diagram.

Define primary keys to uniquely identify records.

Establish relationships between tables to maintain data consistency.

Store product information, stock quantities, expiry dates, and transaction records.

Use basic SQL operations such as INSERT, UPDATE, and SELECT to manage data.

Application Development

System functionality will be implemented using basic C++ programming concepts, including:

Functions

Conditional statements

Loops

Arrays

Simple menus or forms will be developed to support:

Login

Product management

Stock updates

Sales recording

Recording expired items

Viewing alerts and reports

Business Logic Implementation

The system will enforce rules that control how data is processed, including:

Automatically reducing stock after a sale is recorded.

Monitoring stock levels to detect low-stock items.

Tracking expiry dates to identify expired or near-expiry products.

Enforcing user roles:

Admin: manages product and system operations

Owner: manage stock, views stock status and alerts

Accountant: views reports only

Testing

Test each function individually using sample data.

Verify that stock quantities update correctly after sales and stock additions.

Check that alerts are generated for low stock and expired items.

Confirm that users can only access functions permitted to their role.

Deployment and Maintenance

The system is hosted on a local Windows machine.

Enter initial product and stock data.

Provide basic guidance to users.

Perform periodic backups to protect data.

Handle basic input errors to avoid incorrect records.

1.5. Scope and Limitation of the Project

1.5.1 Scope

The system covers:

- Product registration
- Stock updates
- Recording sales
- Expiry date alerts
- Low-stock alerts
- Report generation for income, cost, expiry, and sales

The project focuses only on basic stock management for a single mini market location.

1.5.2 Limitations

The system does not include:

The system does not support barcode scanning. All product identification and sales entry are performed manually using product names or IDs.

The system requires an active internet connection and is accessed through a web interface. Offline mode and mobile applications are not supported.

Supplier information is stored only for reference during ordering. Advanced supplier management features such as supplier performance tracking or contract management are not implemented.

The system supports only a single mini-market location. It does not handle multiple branches or centralized stock synchronization.

Product prices are updated manually by the Admin. Automatic price changes based on market conditions or supplier price feeds are not supported.

Payment processing is handled by an external system and is outside the scope of this project.

1.6. Significance and Beneficiaries of the project

Significance

The system brings several improvements:

- Reduces stock errors from handwritten notes
- Helps prevent expired items from being sold
- Saves time spent on shelf checking
- It helps to produces accurate financial reports
- Supports better stock planning

Beneficiaries

Owner:

Gets real-time visibility of stock levels.

Receives alerts for low stock and expiry.

Can make better purchasing and restocking decisions.

Saves time and avoids financial loss from expired products.

Accountant:

Can access reports online without being physically present at the mini market.

Reviews sales, income, and cost reports remotely.

Saves time by avoiding repeated visits to the shop for report collection.

1.7 Feasibility Analysis

1.7.1 Operational/Organizational Feasibility

The system fits well with daily operations because the users (Owner and Admin) already understand the workflows. The system is simple and easy to operate.

1.7.2 Technical Feasibility

The system requires a basic computer, simple database software, and a lightweight interface.

1.7.3 Economic Feasibility

The cost of development is medium because free tools and simple technologies are used.

1.7.4 Schedule Feasibility

The project can be completed within the semester timeline because the functions are small and well-defined (product entry, stock updates, reporting, alerts).

1.7.5 Legal Feasibility

The system does not violate any legal requirements. It manages only internal stock data and does not involve customer personal information.

1.7.6 Others (Social Feasibility)

The system improves customer service by ensuring products are available and not expired. It also reduces workload for workers.

1.8 Risk Assessment

Risk Category	Risk Description	Impact	Mitigation Strategy
	Internet Connectivity Problems	Since the system is online, unstable or slow internet access may prevent users from accessing the system	**Mitigation:** Perform system operations when internet connectivity is available and resume work once the connection is restored. Keep the system simple to reduce data usage.
	System Bugs or	Errors in C++ logic or	**Mitigation:**

	Application Errors	database queries may cause incorrect stock updates or crashes.	Test each function with sample data. Use input validation to prevent invalid entries. Fix issues during iterative development cycles.
	Data Loss or Corruption	Data may be lost due to server failure, software errors, or accidental deletion.	**Mitigation:** Perform regular database backups. Restrict delete operations to authorized users only. Store backups on external storage or cloud services.
	Incorrect Data Entry by Users	Users may enter wrong quantities, prices, or expiry dates.	**Mitigation:** Use validation rules (e.g., no negative stock, required fields). Provide clear error messages. Keep input forms simple.
	Power Interruption (Very realistic for Ethiopia)	Power outages can interrupt system access or cause incomplete operations.	**Mitigation:** Auto-save data after each transaction. Use backups to recover data after interruptions. Resume work when power is restored
	Limited Technical Skills of Users	Users may struggle to use the system correctly.	**Mitigation:** Design a simple and clear user interface. Provide short training sessions for Owner, Admin, and Accountant. Use meaningful labels and alerts.
	Unauthoriz	Unauthorized	**Mitigation:**

	ed Access	users may try to access sensitive data.	Design a simple and clear user interface. Provide short training sessions for Owner, Admin, and Accountant. Use meaningful labels and alerts.

Table 1.1: Risk assessment

1.9 Development Tools

Programming Language: C++

Database: SQL (SQL Server)

Development Environment: Windows Operating System

Code Editor: (Code::Blocks)

Modeling and Design Tools:

To modeling UML Diagrams

Tools Enterprise Architect

Documentation Tool: Microsoft Word

Testing Tools: Manual Testing

1.10 Work Breakdown

1.10.1 Project Plan Activities (Schedule)

Phase/Major Task	Sub Tasks	Date
Project initiation	1.1 Topic selection	-
	1.2 Team formation	-
	1.3 Project planning	11/03/2018 E.C-12/03/2018 E.C
Requirement gathering	2.1 Stakeholder identification	12/03/2018 E.C -16/03/2018 E.C
	2.2 Data collection	
	2.3 Current system study	
	2.4 Problem identification	

	2.5 Requirement validation	
Requirement analysis	3.1 Functional requirements	16/03/2018 E.C -20/03/2018E.C
	3.2Non-functional requirements	
	3.3Requirement prioritization	
System design	4.1 System architecture	20/03/2018 E.C -28/03/2018E.C
	4.2 Use case diagram	
	4.3 Class diagram	
	4.4 ER diagram	
	4.5State, sequence & activity diagrams	
Database design	5.1 Table design	28/03/2018 E.C -02/04/2018E.C
	5.2 Keys and relationships	
	5.3 SQL scripts	
System implementation	6.1 Development	02/04/2018 E.C -10/04/2018E.C
	6.2 Database integration	
	6.3 Stock operations	
Interface design	7.1 user interface	10/04/2018 E.C -11/04/2018E.C
Testing	8.1 Unit testing	11/04/2018 E.C -13/04/2018E.C
	8.2 Integration testing	
	8.3 User acceptance testing	
Deployment	9.1 System installation	13/04/2018 E.C -14/04/2018E.C
	9.2 Data initialization	
Documentation	9.1 Project report	14/04/2018 E.C -16/04/2018E.C
	9.2 User manual	

	9.3 Presentation slides	
Maintenance	10.1 Bug fixing	TBD
	10.2 Future enhancement Sub Tasks	

Table 1.2: Project Plan Activities (Schedule)

Note: “-“indicates tasks that are not made by the team.

“TBD” indicates activities with flexible schedules and planned for later system versions.

1.10.2 Project Organization

The Stock Management System project was organized as a small student software development project carried out by a team of five members. The project organization was structured to divide responsibilities clearly while allowing collaboration across tasks.

The project followed a task-oriented organization, where work was grouped according to major phases of the software development life cycle, including requirement analysis, system design, database design, implementation, testing, and documentation.

Each phase of the project was planned and executed with coordination among team members to ensure consistency between requirements, design models, and implementation results. Although responsibilities were assigned, team members supported each other to maintain progress and resolve technical or documentation challenges.

This organization helped the team manage workload efficiently, reduce errors, and ensure that all project components were completed on time.

1.10.3 Team Organization

Name	ID	Role
Eyobed Solomon	WCU172510	Developer
Fetiha Oumer	WCU171211	System design
Minase Mengesha	WCU17D6911	Requirement analyst and documentation
Ruhama Alemu	WCU172019	Database designer
Yamilaksira Ashenafi	WCU17D4021	Project manager

Table 1.3: Team Organization

1.11 Budget Plan

Item Category	Item Description	Quantity	Estimated Cost (Birr)
Hardware	Laptops	2	-
	Desktop Computer	1	50,000
Development Tools	C++,SQL server , UML Tools	1	Free
Supplies	Papers	200 Sheets	400
	Pens	5	125
Operational Costs	Printing & Binding	100 sheets	1200
	Transport	1	150
	Cable,	3	400
Labor/Team Payment	Project Manager	1	15000
	Requirement Analyst and documentation	1	12000
	System design	1	10000
	Database Designer	1	10000
	Developer	1	10000
Total Costs			109,275

Table 1.4: Budget Plan

CHAPTER TWO

REQUIREMENT ANALYSIS AND SPECIFICATIONS

2.1. Description of the Current System

The mini market operates using a fully manual record-keeping approach. The owner writes new products in notebook and papers updates quantities after sales, and occasionally notes expiry dates. To check availability, the owner manually checks the shelves. This method is slow, error-prone, and relies heavily on memory.

The accountant calculates income and cost at the end of each month using the same notebook and papers making the process even more prone to errors.

2.2 Players/Actors in the Existing System

In the current manual system, there are only two people involved in managing and reviewing stock-related information:

Owner:

The Owner performs all daily activities, including:

Recording new products in the notebook and papers

Updating stock after sales

Removing unusable or expired items

Checking shelves to confirm availability

Identifying low-stock products

Manually checking expiry dates

Maintaining the only record book

The Owner is the main actor who handles every operation in the existing system.

Accountant:

The Accountant is not involved in daily activities but works periodically. They:

Review the notebook and papers to calculate monthly income

Check cost and income

Prepare monthly financial summaries

The accountant depends on the Owner's handwritten entries.

2.3 Use Case Diagram for the Existing System

Owner Use Cases:

Add New Stock

Update Stock after Sales

Remove Stock

Check Stock Availability

Identify Low Stock

Check Expiry Dates

Record Expired Items

Accountant Use Cases:

Review Monthly Records

Calculate Income

No system automation, include, or extend relationships exist because the process is fully manual.

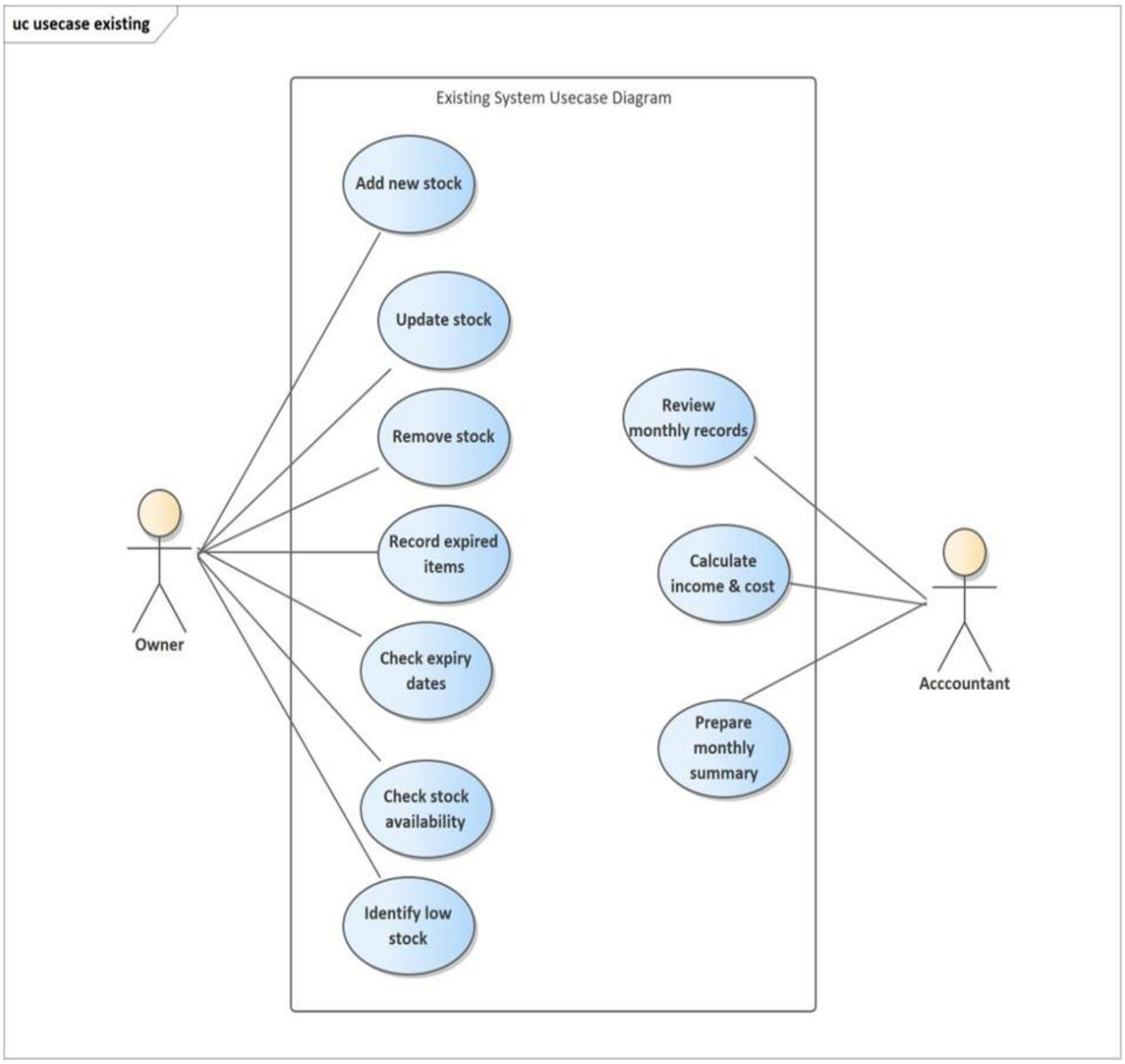


Figure_2.1: Use case diagram of existing system

2.4 Forms and Documents Used in the Existing System

The existing system does not use any digital forms. All records are kept manually using Notebook and Papers.

This is the main document where:

New products are written

Stock updates are recorded

Sales are written manually

Expiry information (sometimes) is noted

Monthly records are checked by the Accountant

Physical Shelf Inspection: to check availability, the Owner must visually inspect items.

Accountant Calculations: the Accountant uses the same notebook to prepare financial summaries manually.

There are no digital records, templates, or backup documents, making the system disorganized and unreliable.

2.5 Business Rules of the Existing System

The existing system follows several informal rules:

1. All product, stock, and sales information must be written by hand in a notebook or on paper.
2. Stock quantity changes only when the Owner manually updates the notebook after sales or purchases.
3. Sales are recorded manually after they happen, based on memory or paper notes.
4. Expiry dates are checked by physically reading product labels on shelves.
5. Expired items are recorded on paper only when the Owner notices them.
6. There is no written rule to automatically warn about low stock or expiring products.
7. Monthly income and cost are calculated manually using notebook records.
8. If records are missing, unclear, or forgotten, values are estimated by the Accountant.
9. There is no backup of records; loss or damage of notebooks and papers results in permanent data loss.

2.6 Proposed System

2.6.1 Overview

The proposed system is a simple computer-based stock management system that replaces the handwritten notebook and papers used in the current manual process. It automates stock recording, sales updates, expiry tracking, and report generation. The system allows the Owner to manage all stock-related activities digitally, while the Accountant can view records without modifying stock data.

The new system improves accuracy, reduces human error, provides real-time information, and saves time spent on manual counting and checking shelves. It also introduces automatic alerts for low-stock items and upcoming expiry dates, which are not possible in the current manual system.

The proposed system replaces the notebook and paper with a computer-based stock management system. It allows the owner and admin to register products, update stock, record sales, and generate expiry or low-stock alerts automatically.

Key Improvements:

Quick product searches

Expiry and low-stock notifications

Accurate and reliable reports

2.6.2 Business Rules of the Proposed System

The proposed system follows the following new business rules:

1. Products must be registered in the system before any stock operation is performed.
2. Stock quantity can only be increased using the Add Stock function.
3. Stock quantity must not be allowed to go below mini-stock.
4. Products with expiry dates must have expiry information recorded.
5. The system must generate alerts for low-stock and expired products.
6. Expired products must not be allowed for sale.
7. All users must log in to access the system.
8. The Admin manages products and system operations.
9. The Owner views stock, manage stock status and alerts.
10. The Accountant can only view reports and financial summaries.
11. The system must generate simple reports for sales, stock status, and expired items.
12. The system must store data securely and support basic backup.

2.6.3 Functional Requirements

1. The system shall allow users to log in using a username and password.
2. The system shall allow the Admin to add new products.
3. The system shall allow the Admin to update product information.

4. The system shall allow the Admin to add product quantities.

5. The system shall allow recording of sales transactions.

6. The system shall allow recording of expired products.

7. The system shall display low-stock alerts.

8. The system shall display expiry alerts.

9. The system shall allow viewing of current stock information.

10. The system shall generate reports for stock, sales, and expired items

11. The system shall allow the Accountant to view financial reports only.

12. The system shall allow users to log out of the system.

2.6.4 Non-Functional Requirements

Usability:

The system should be easy to use for users with basic computer knowledge. Menus, labels, and alerts should be clear so that the Owner, Admin, and Accountant can perform their tasks without confusion or extensive training.

Performance:

The system should respond quickly to user actions such as logging in, viewing stock, or generating reports. Delays in response may affect daily operations, especially when checking alerts or updating stock.

Reliability:

The system should store and retrieve data accurately without loss or duplication. Stock quantities, sales records, and reports must remain consistent even after repeated use.

Security:

The system should protect data using login authentication and role-based access. Each user (Admin, Owner, and Accountant) must only access the functions permitted to their role to prevent misuse or unauthorized changes.

Availability

Since the system is online, it should be accessible whenever internet connectivity is available. Users should be able to access stock information and reports from any location with proper login credentials.

Maintainability:

The system should be designed in a simple and modular way so that future updates or fixes can be made easily. This helps in correcting errors, improving features, or adapting the system as the business grows.

Data Backup and Recovery:

The system should support regular data backup to prevent loss of information. In case of system failure or unexpected errors, backed-up data can be restored to continue operations with minimal disruption.

2.6.5 Actor and Use Case Identification

Admin Use Case:

Login

Manage product (delete, update, and add product)

Update account

Owner Use Cases:

Login

Record Sales

View Stock

Manage Stock (delete, add, and update stock)

View Reports

View Low-Stock Alert

View Expiry Alert

Generate report

Accountant Use Cases:

Login

Check Income and Cost

External payment system:

Record Sales

Payment Integration

2.6.6 System Model

2.6.6.1 Use Case Model

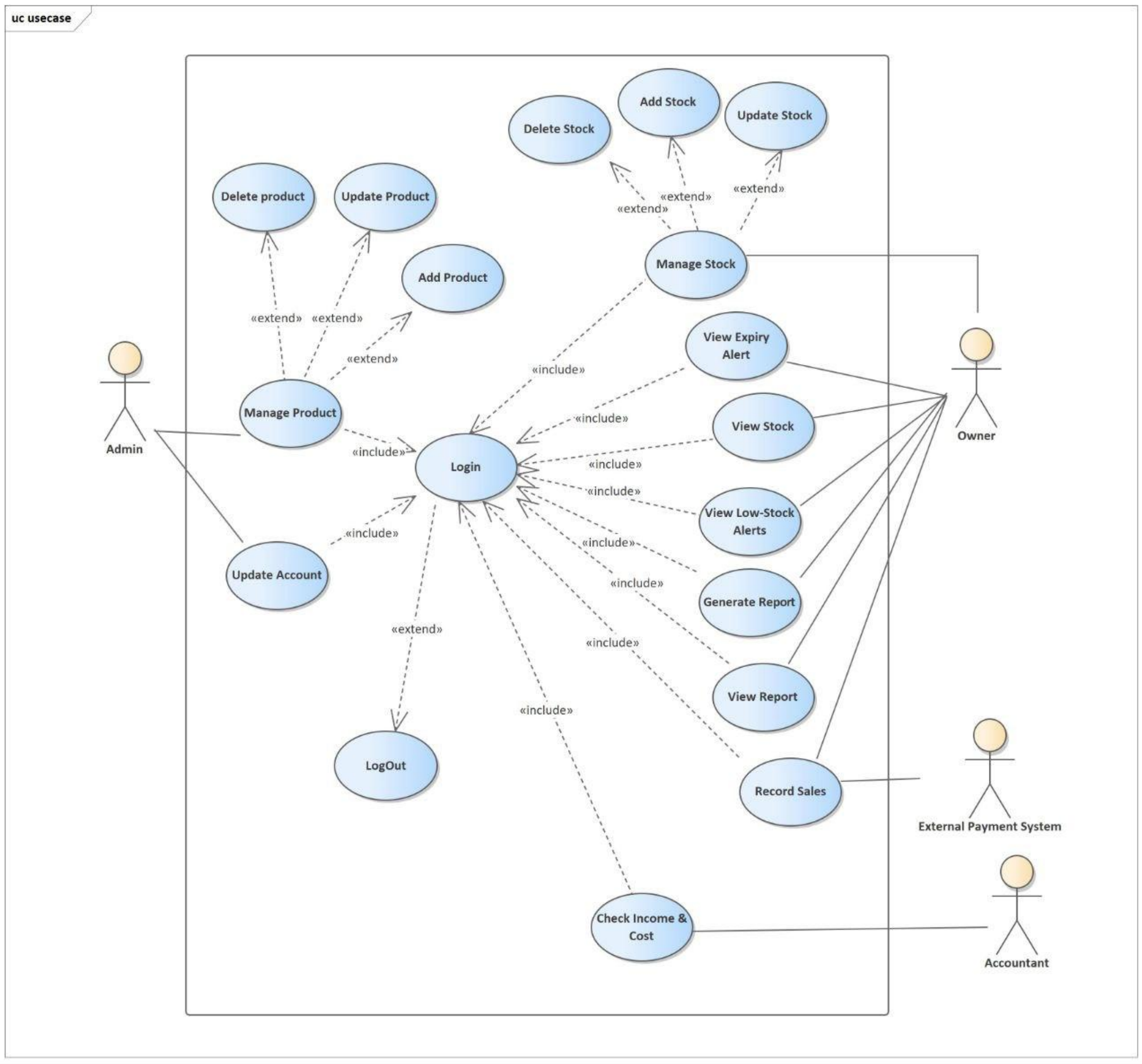


Figure_ 2.2: use case of proposed system

2.6.6.1.1 Use Case Descriptions

Login

Actors	Admin, Owner, Accountant
Description	Allows system users to access the system using valid credentials
Precondition	User account must exist

Main Flow	User enters username and password → System validates credentials → Access granted
Alternative Flow	Invalid credentials → Error message shown
Post condition	User is logged into the system

Table_2.1_ Use case description for use case login

Add Product

Actor	Admin
Description	Allows the Admin to register a new product into the system
Precondition	Admin must be logged in to the system
Main Flow	Admin enters product details → System validates → Product saved
Alternative	Flow Missing/invalid data → Entry rejected
Post condition	Product is added to the stock system

Table_2.2_ Use case description for use case Add Product

Update Product

Actor	Admin
Description	Allows the Admin to update details of an existing product
Precondition	Admin logged in and product exists
Main Flow	Admin selects product → Updates product details → System saves changes
Alternative Flow	Invalid input → Update rejected
Post condition	Product information is updated

Table_2.3_ Use case description for use case Update Product

Delete Product

Actor	Admin
Description	Allows the Admin to remove a product from the system
Precondition	Admin logged in and product exists
Main Flow	Admin selects product → Confirms deletion → System deletes product

Alternative Flow	Deletion cancelled → Product remains unchanged
Post condition	Product removed from the system

Table_2.4_ Use case description for use case Delete Product

Add Stock

Actor	Owner
Description	Allows the owner to increase stock quantity of a product
Precondition	owner logged in and product exists
Main Flow	owner selects product →Enters quantity →System updates stock
Alternative Flow	Invalid quantity → update rejected
Post condition	Stock quantity increased

Table_2.5_ Use case description for use case Add Stock

Update Stock

Actor	Owner
Description	Allows the owner to correct of existing stock values
Precondition	Owner logged in to the system
Main Flow	Owner selects stock record →Modifies quantity → System saves update
Alternative Flow	Invalid value → update cancelled
Post condition	Stock record updated

Table_2.6_ Use case description for use case Update Stock

Delete Stock

Actor	Owner
Description	Allows removal of stock record when necessary
Precondition	Owner logged in to the system
Main Flow	Owner selects stock entry → Confirms deletion → System removes record
Alternative Flow	Owner cancels operation
Post condition	Stock record deleted

Table_2.7_ Use case description for use case Delete Stock

Record Sale

Actor	Owner, external payment system
Description	Records product sales and reduces stock automatically
Precondition	Owner or external payment system logged in and sufficient stock available
Main Flow	Owner or external payment system records sale→ System deducts stock→ Transaction saved
Alternative Flow	Insufficient stock → sale rejected
Post condition	Sale recorded and stock updated

Table_2.8_ Use case description for use case Record Sale

View Stock

Actors	Owner
Description	Allows the owner viewing of current stock levels
Precondition	Owner logged in to the system
Main Flow	Owner opens stock page→ System displays stock list
Alternative Flow	No stock available
Post condition	Stock information viewed

Table_2.9_ Use case description for use case View Stock

View Report

Actor	Owner
Description	Allows the Owner to view generated system reports
Precondition	Owner logged in to the system
Main Flow	Owner selects report type → System displays report
Alternative Flow	No data available
Post condition	Report viewed

Table_2.10_ Use case description for use case View Report

Generate Reports

Actor	Owner
Description	Allows the owner to generate financial and stock reports
Precondition	Owner logged in to the system
Main Flow	Owner requests report → System retrieves data → Report generated
Alternative Flow	Missing data → report not generated
Post condition	Report available for viewing

Table_2.11_ Use case description for use case Generate Reports

Check Income and Cost

Actor	Accountant
Description	Allows viewing of summarized income and cost information
Precondition	Accountant logged in to system
Main Flow	Accountant selects income/cost option → System displays summary
Alternative Flow	No transactions recorded
Post condition	Income and cost information displayed

Table_2.12_ Use case description for use case Check Income and Cost

View Low Stock Alert

Actor	Owner
Description	Notifies the Owner when stock levels fall below threshold
Precondition	Owner logged in to the system
Main Flow	Owner opens alerts → System shows low-stock items
Alternative Flow	No low-stock items
Post condition	Owner informed of low stock

Table_2.13_ Use case description for use case View Low Stock Alert

View Expiry Alert

Actor	Owner

Description	Alerts Owner about products nearing or past expiry
Precondition	Owner logged in to the system
Main Flow	Owner opens expiry alerts → System displays expiring items
Alternative Flows	No expiry alerts
Post condition	Owner informed of expiry risks

Table_2.14_ Use case description for use case View Expiry Alert

Update Account

Actor	Admin
Description	Allows the Admin to manage user accounts and system-related account operations. This use case covers account updates, password recovery, password reset, and basic system maintenance related to user access
Precondition	Admin must login and have permission to update the specific account.
Main Flow	Admin selects a user account → Admin chooses an action (update account details, reset password, or perform maintenance) →System validates the request → System applies the changes successfully
Alternative Flows	User not found → system shows error message → Invalid account data → update rejected → Password reset fails → system requests retry
Post condition	User account information or access settings are updated successfully

Table_2.15_ Use case description for use case Update Account

Logout

Actor	Admin, Owner, Accountant
Description	Allows a logged-in user to securely end their system session
Precondition	User must be logged in
Main Flow	User selects logout option → System terminates the session → User is redirected to login page
Post condition	User is logged out of the system

Table_2.16_ Use case description for use case Logout

2.6.6.2 Sequences diagram

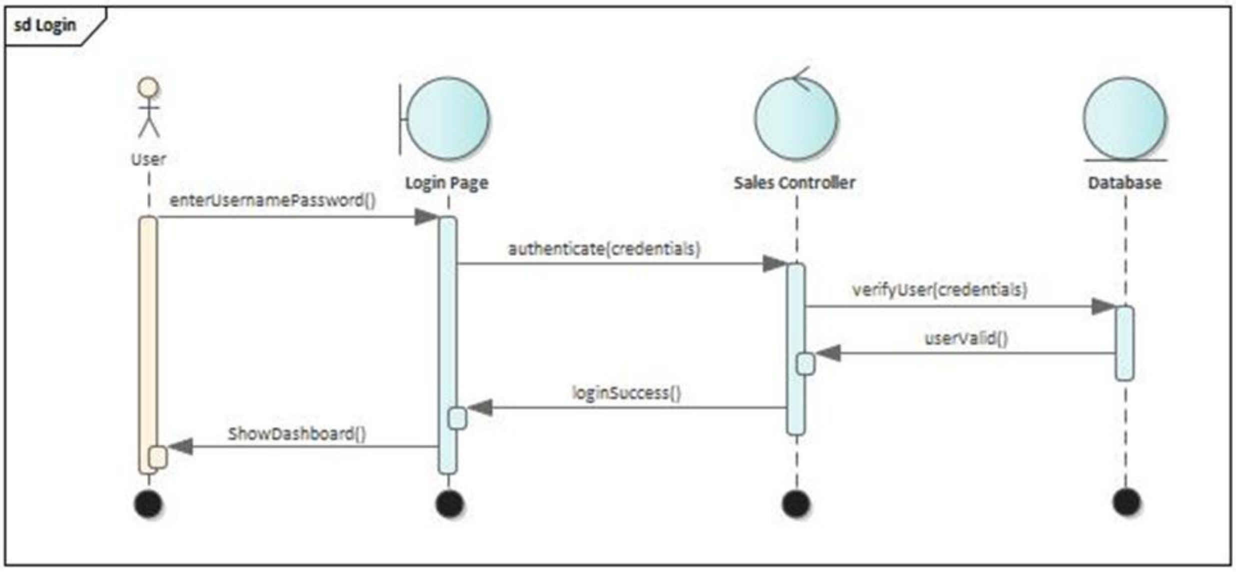


Figure_ 2.3: Sequences diagram for use case Login.

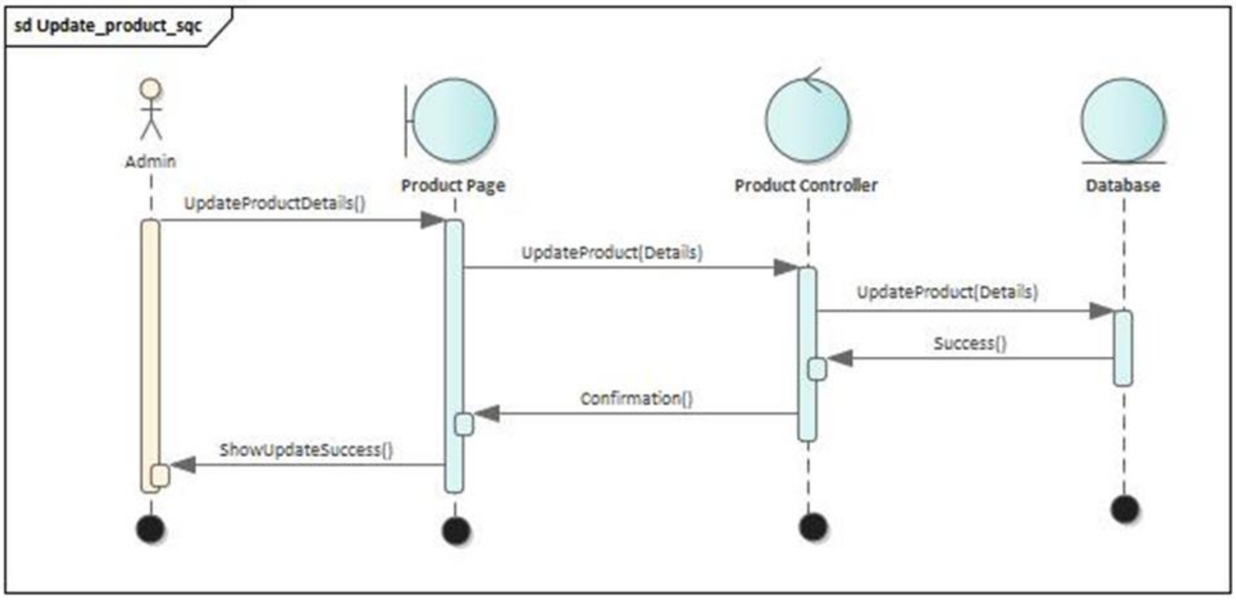


Figure_ 2.4: Sequences diagram for use case update product.

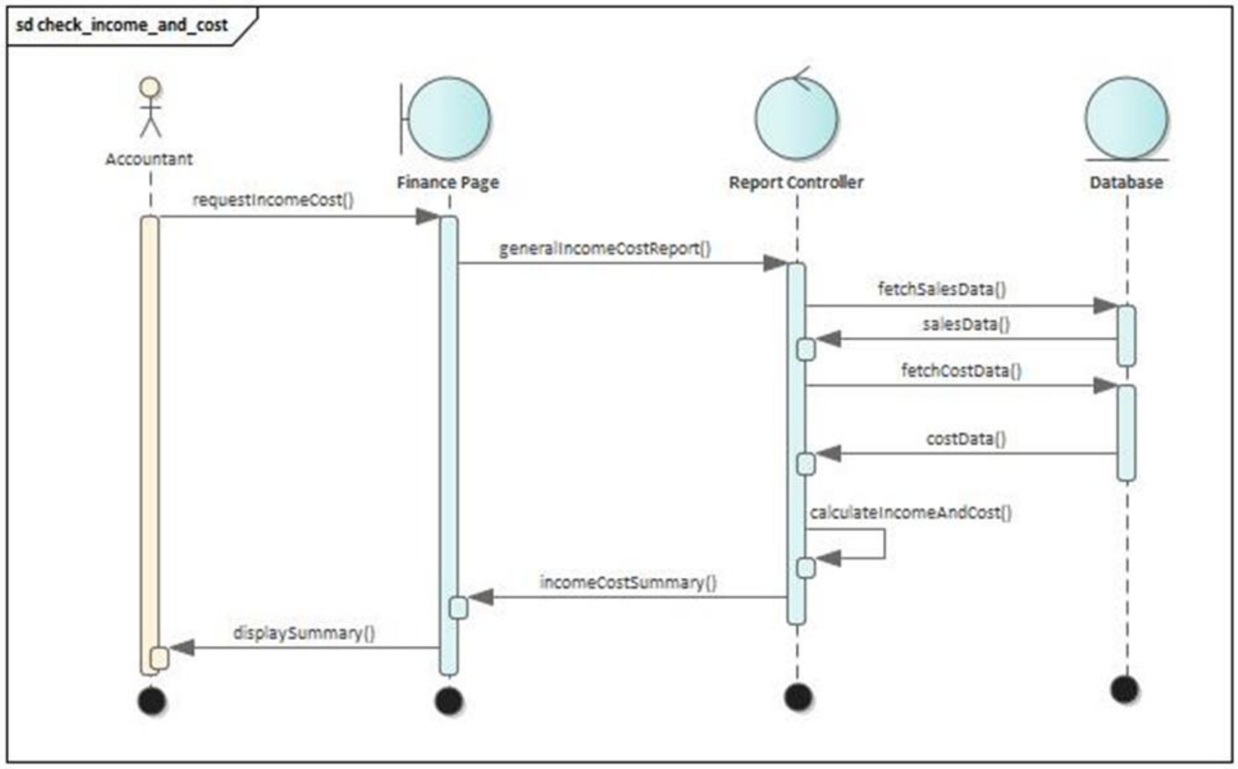


Figure_ 2.5: Sequences diagram for use case check income and cost.

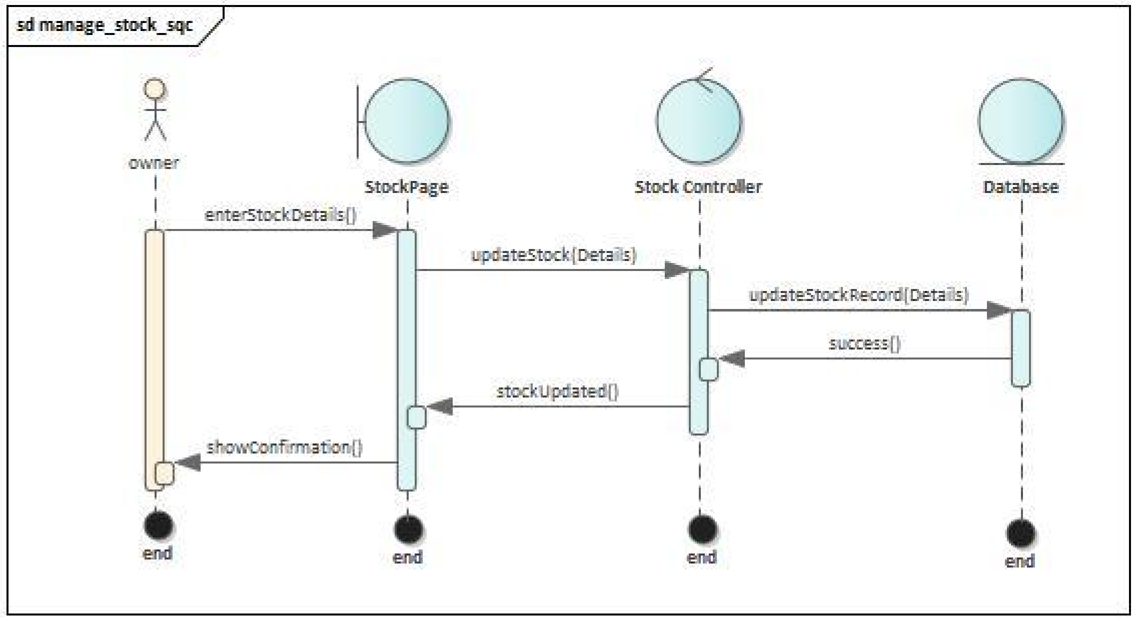


Figure_ 2.6: Sequences diagram for use case manage stock.

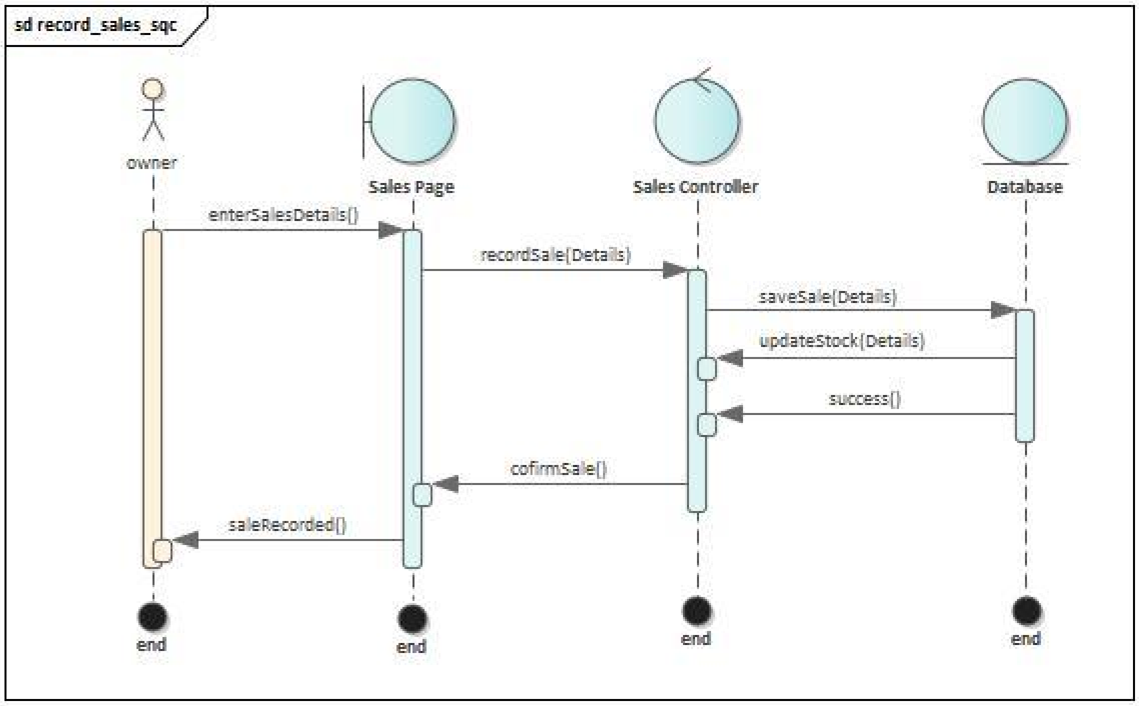


Figure_ 2.7: Sequences diagram for use case record sales.

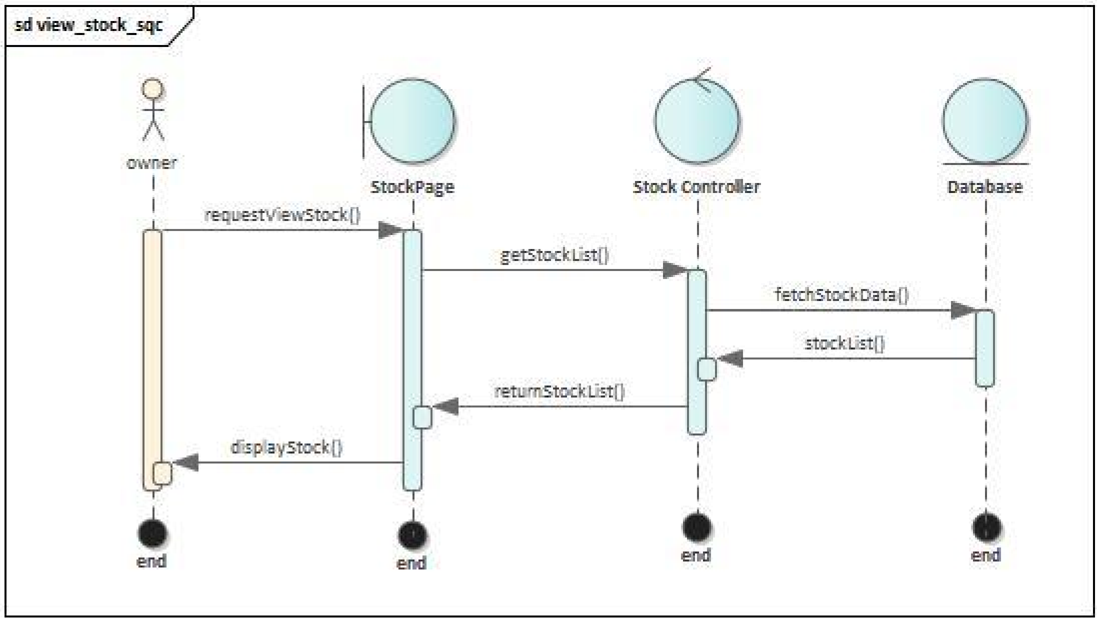


Figure_ 2.8: Sequences diagram for use case view stock.

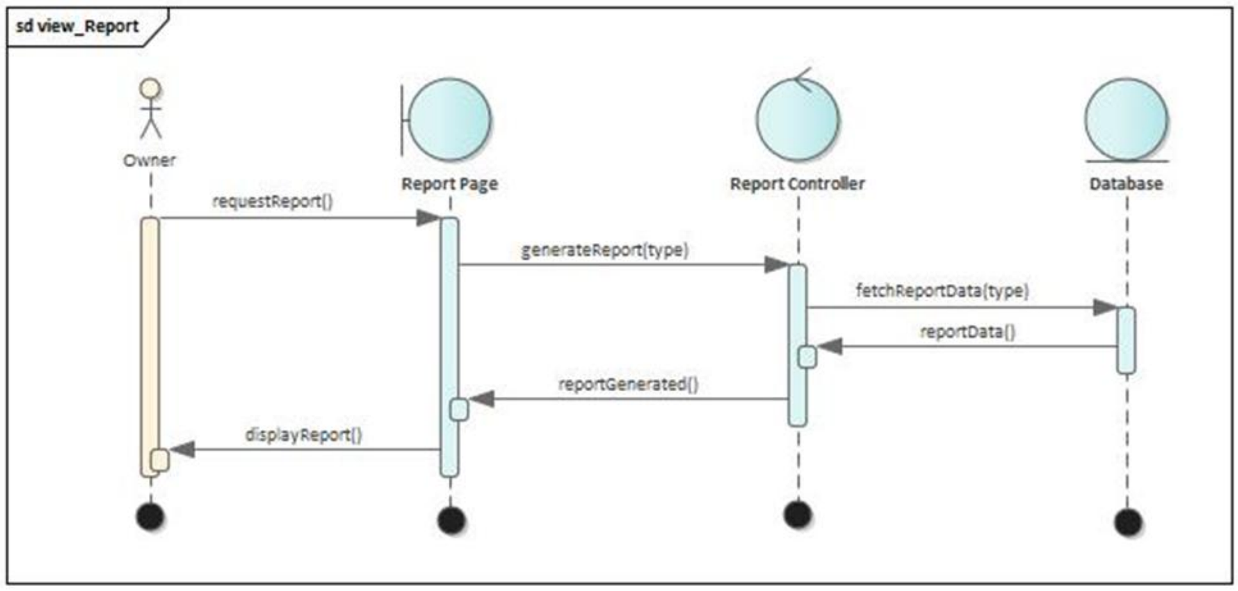


Figure_ 2.9: Sequences diagram for use case view report.

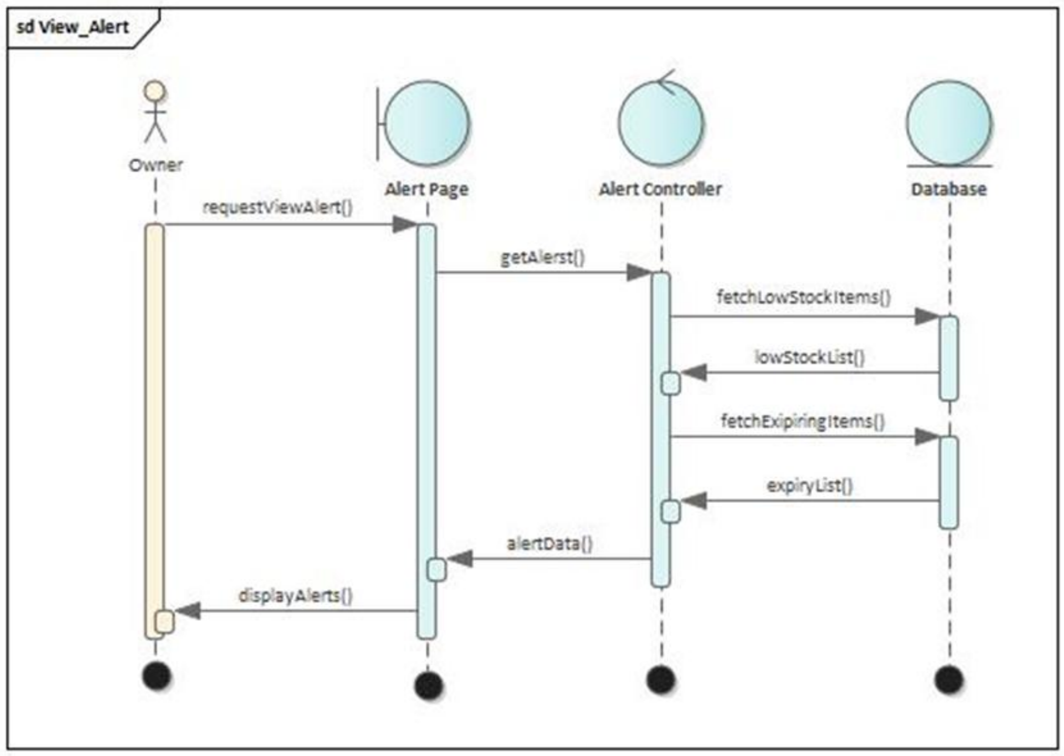


Figure_ 2.10: Sequences diagram for use case view alert.

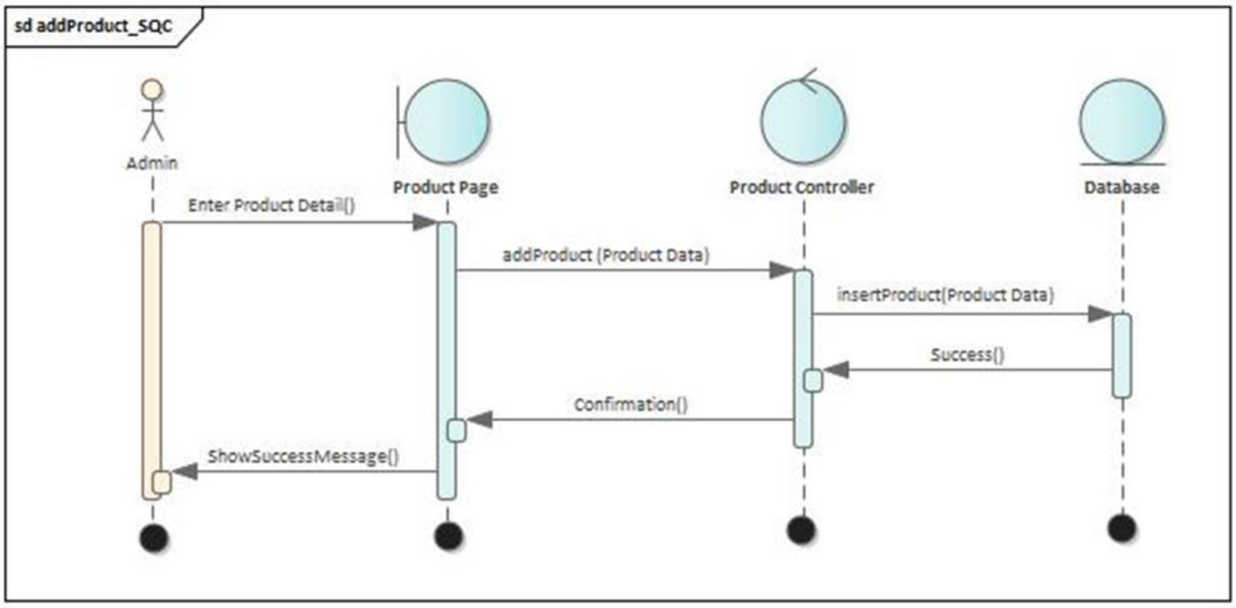


Figure_ 2.11: Sequences diagram for use case add product.

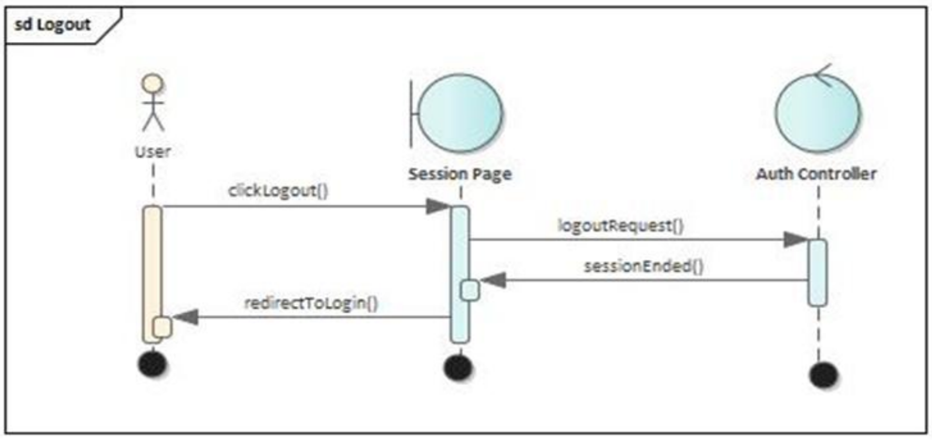


Figure_ 2.12: Sequences diagram for use case Logout.

2.6.6.3 Activity Diagrams

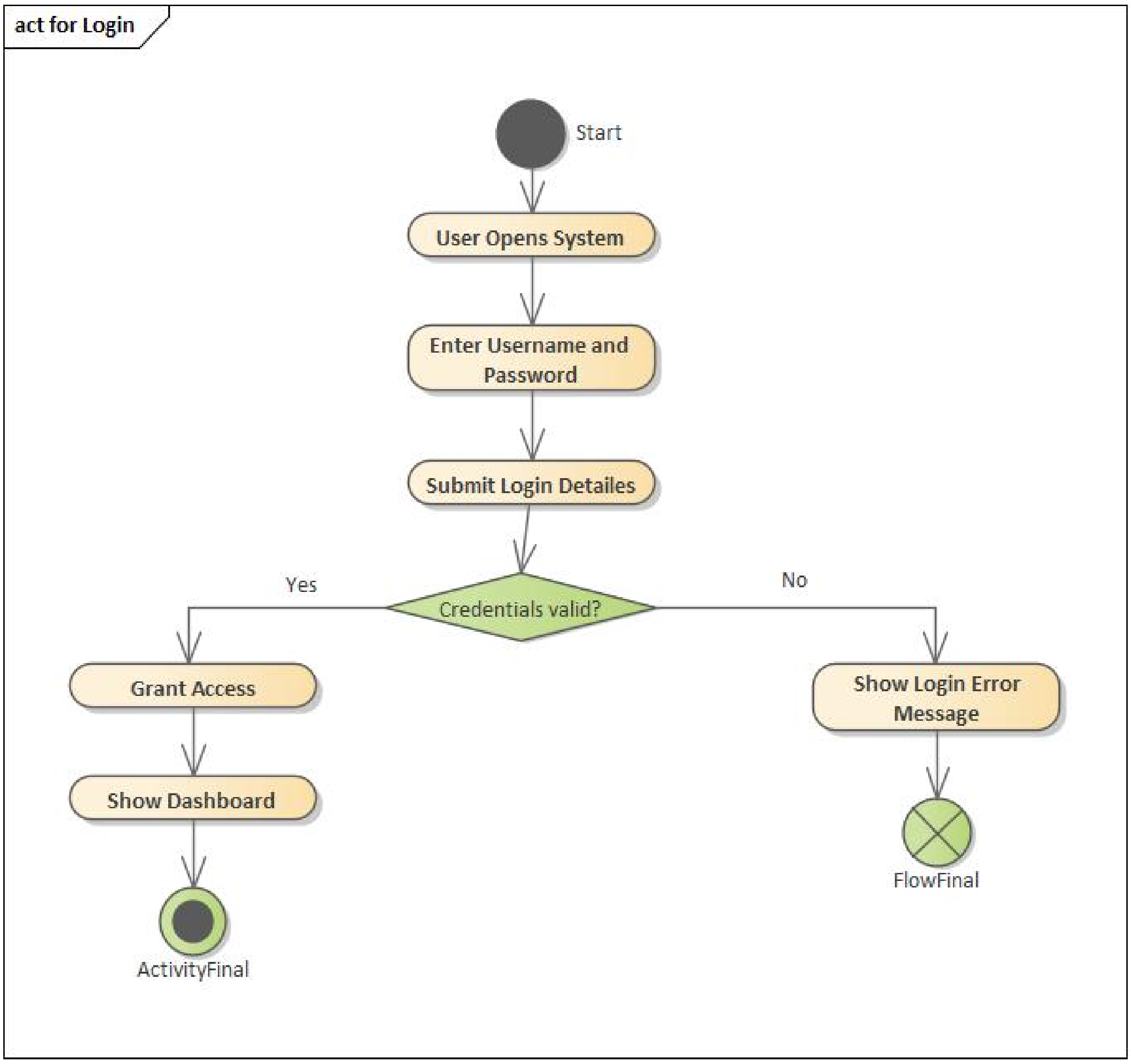


Figure_ 2.13: Activity diagram for use case login.

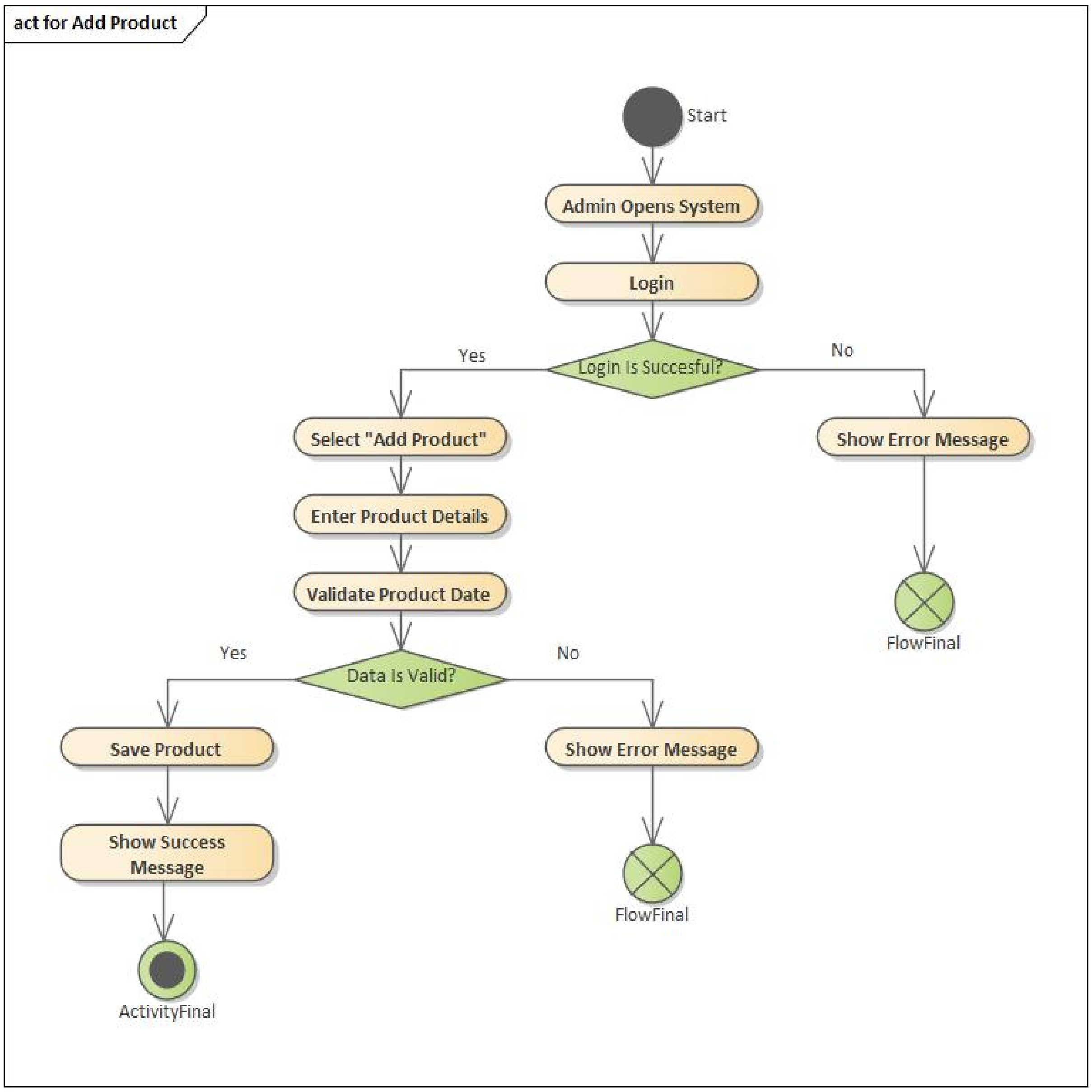


Figure_ 2.14: Activity diagram for use case add product.

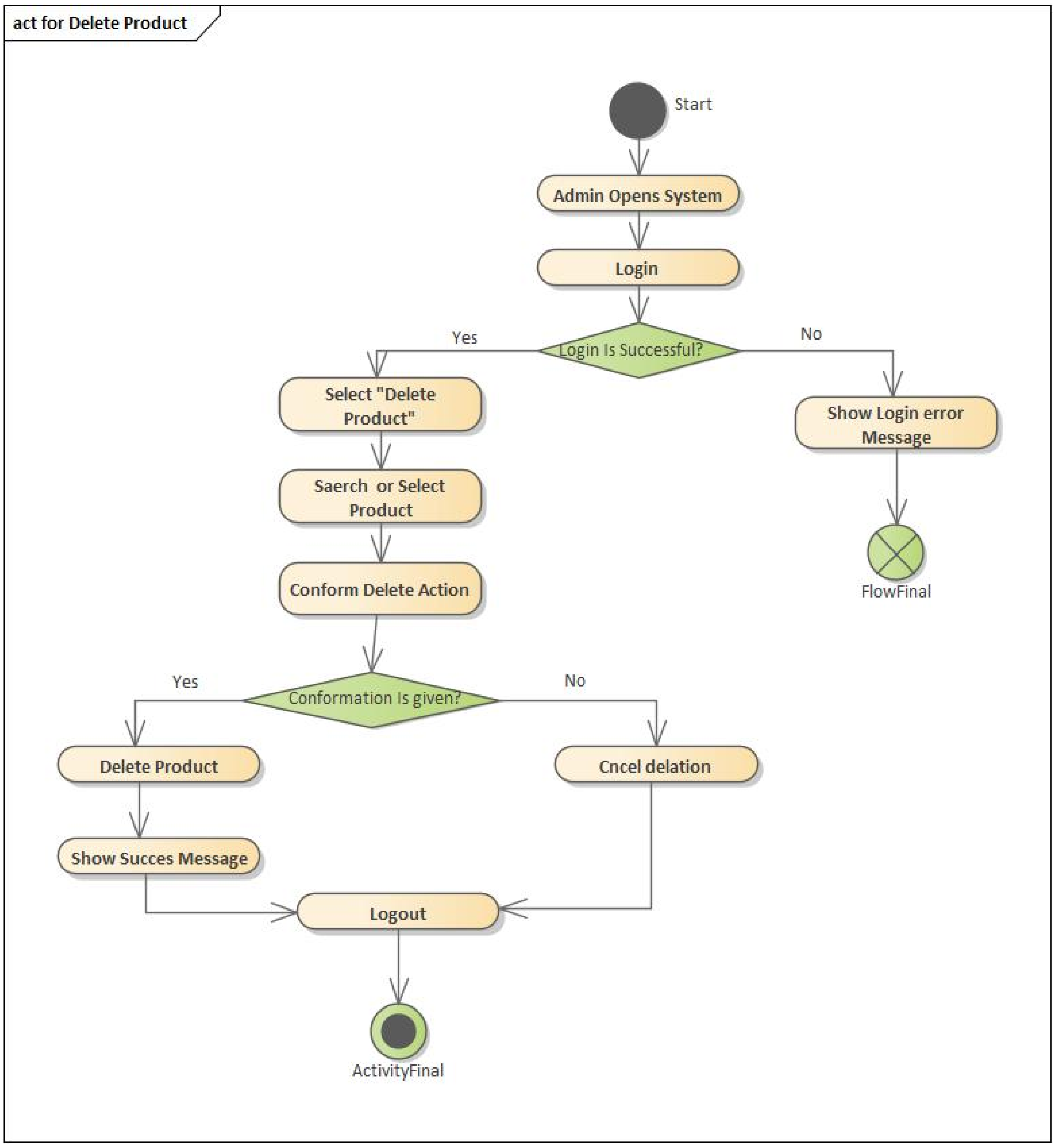


Figure_ 2.15: Activity diagram for use case delete product.

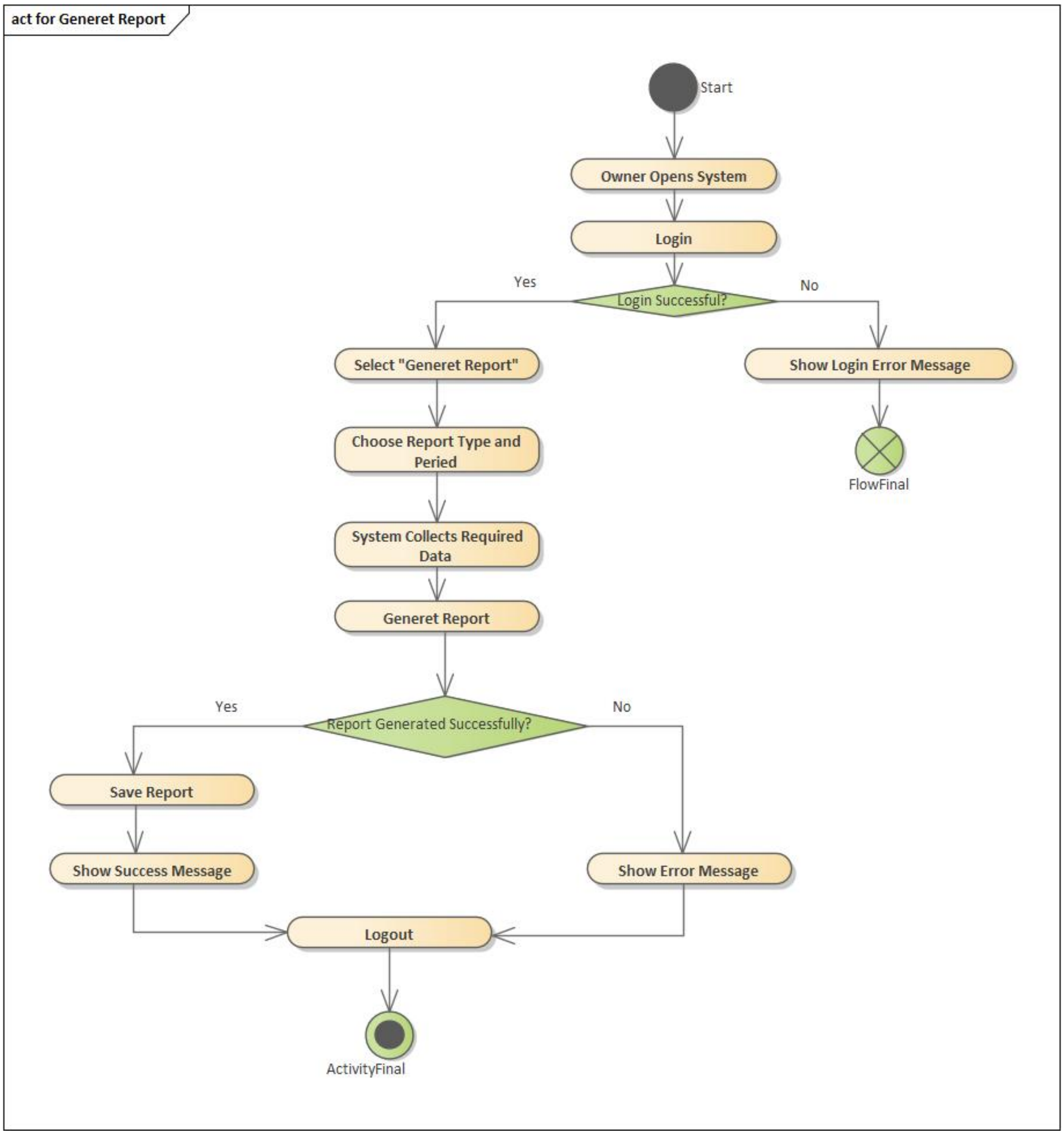


Figure_ 2.16: Activity diagram for use case generate report.

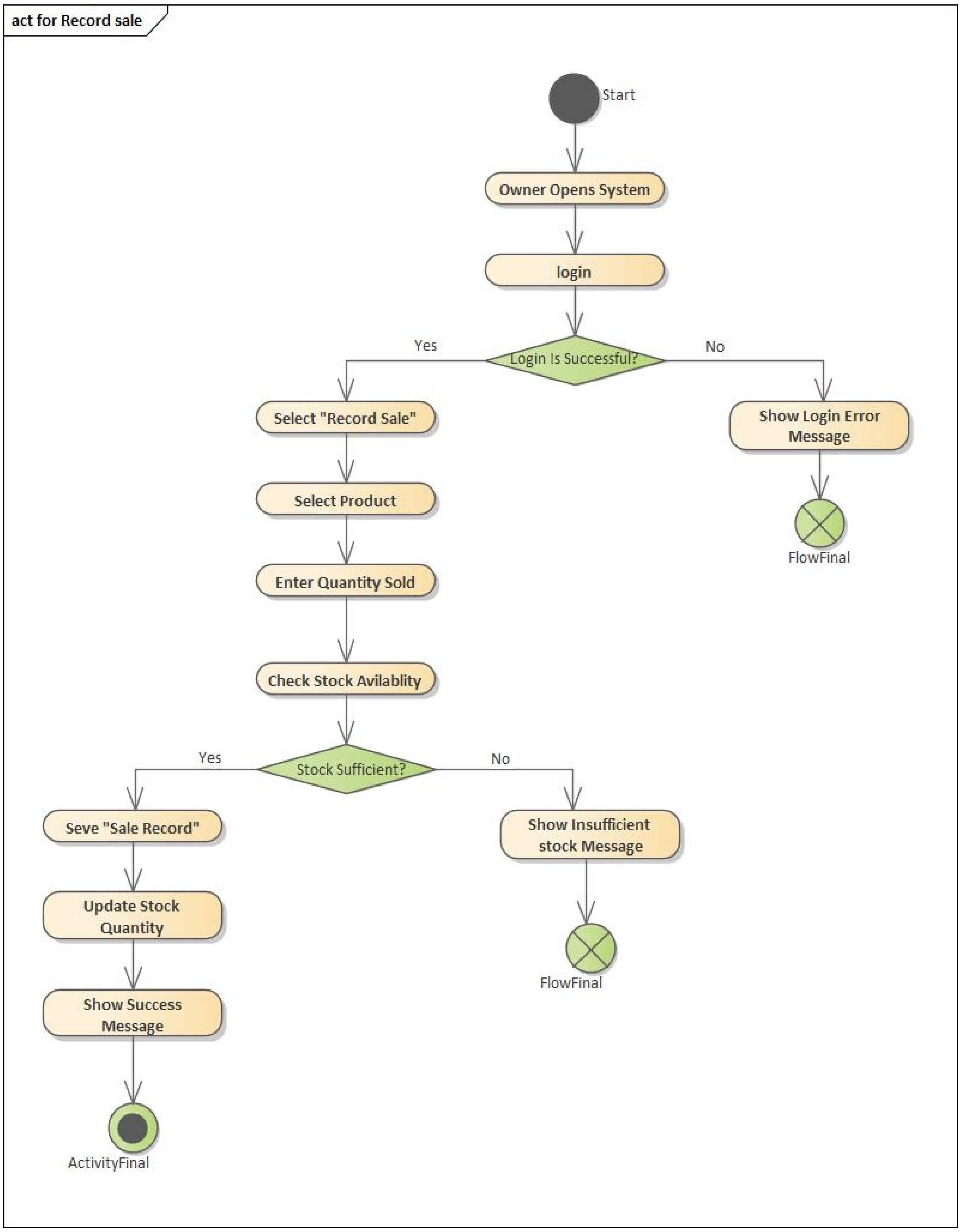


Figure_ 2.17: Activity diagram for use case record sales.

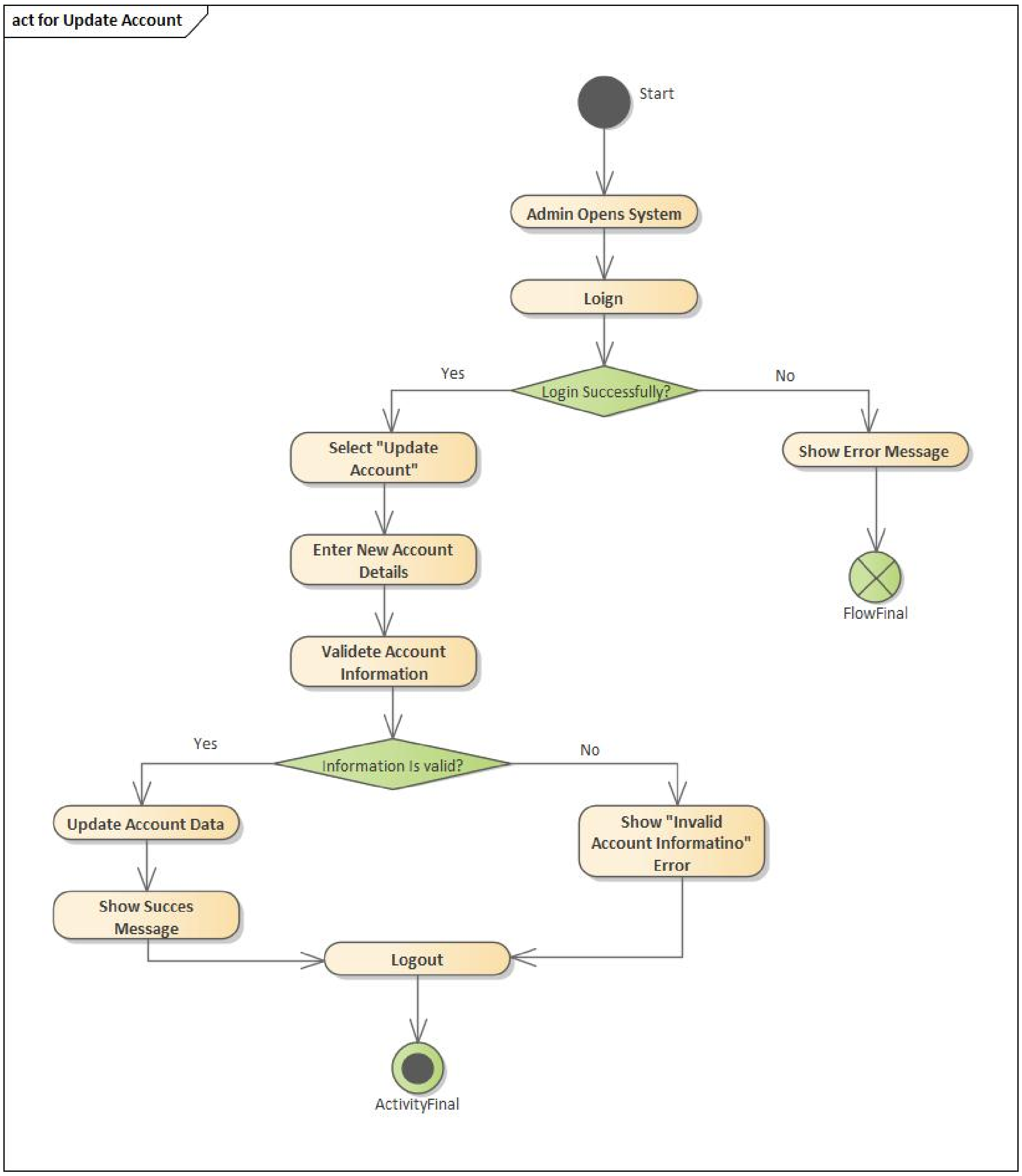


Figure_ 2.18: Activity diagram for use case update account.

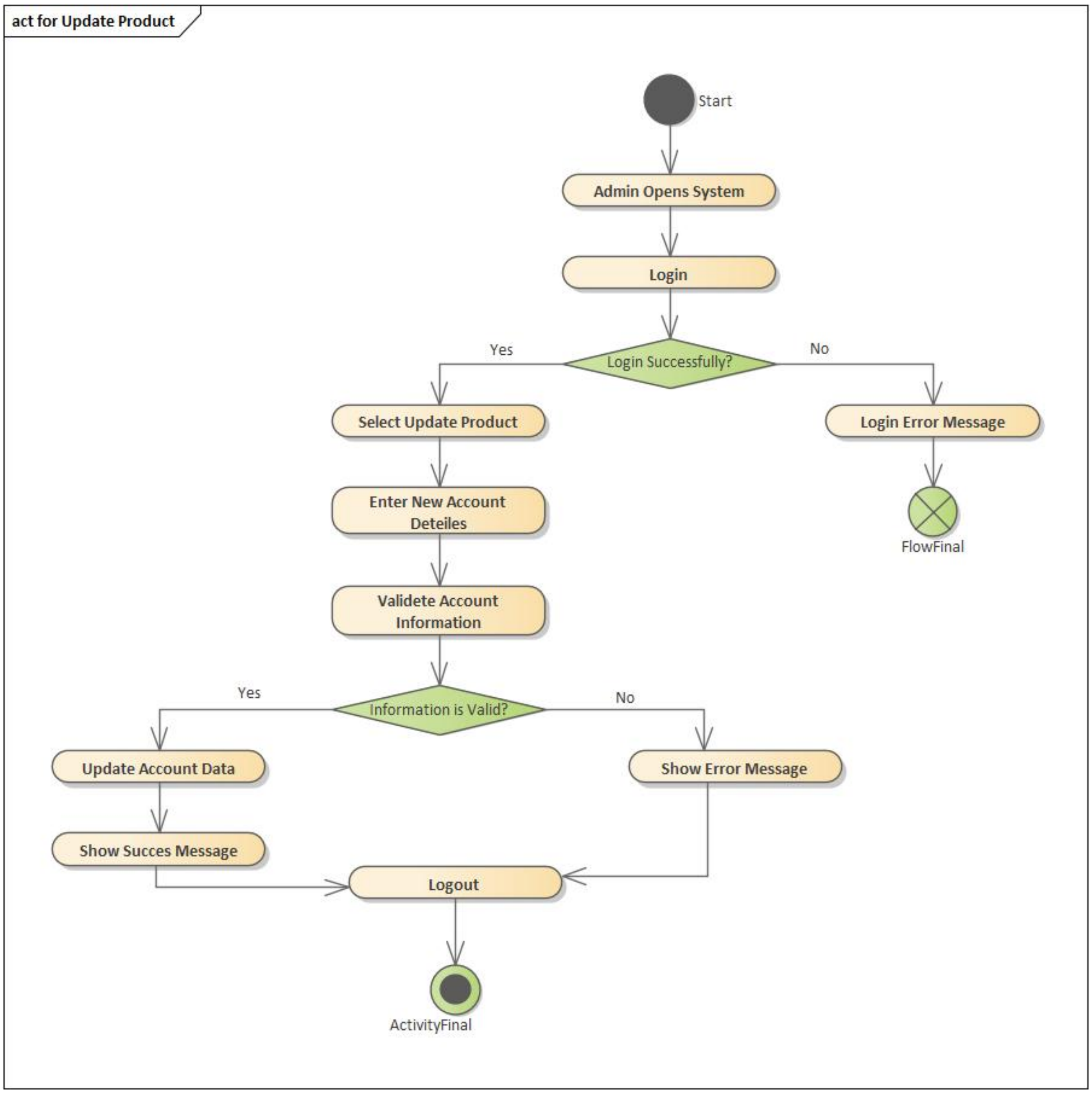


Figure_ 2.19: Activity diagram for use case update product.

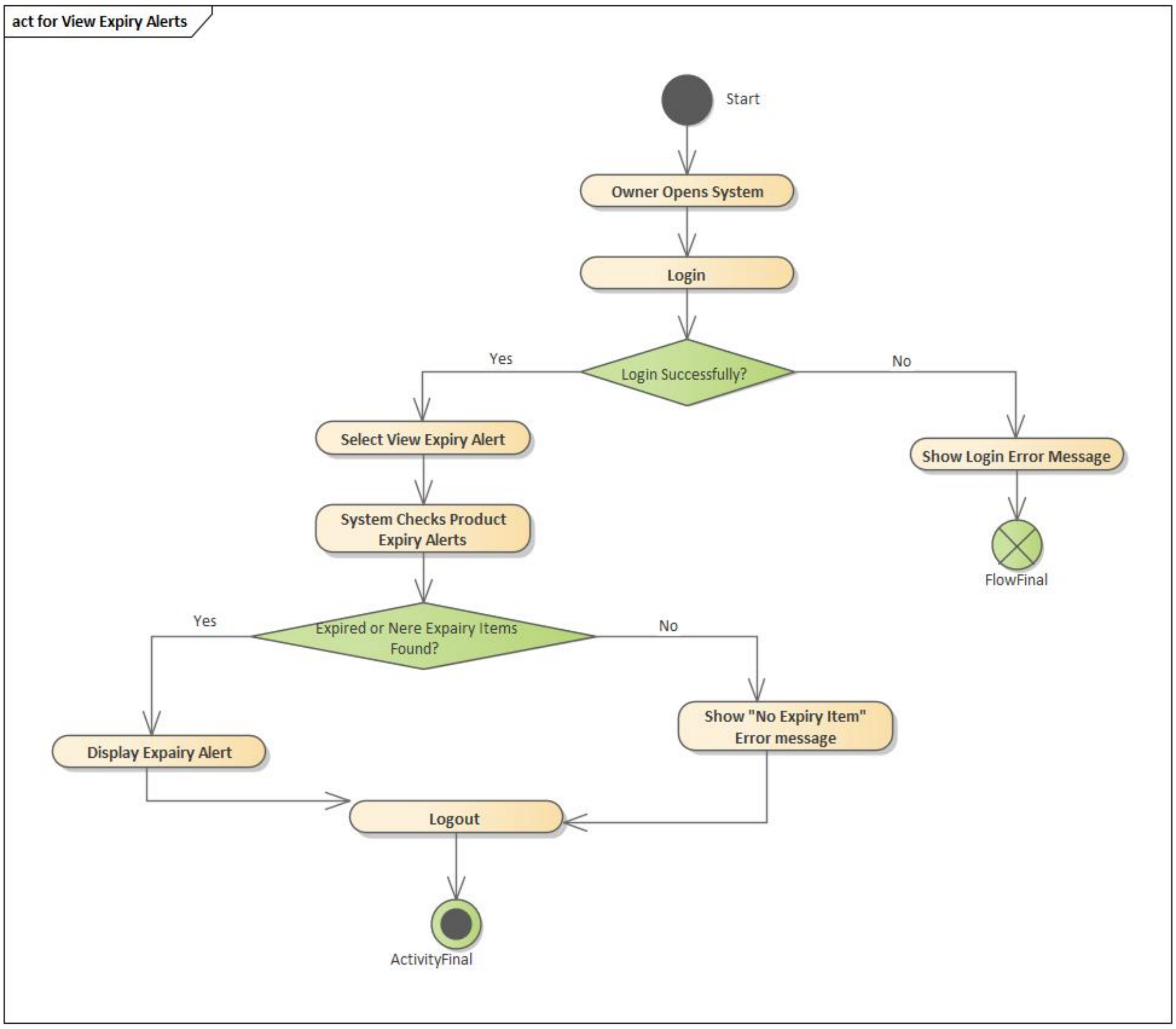


Figure_ 2.20: Activity diagram for use case view expiry alert.

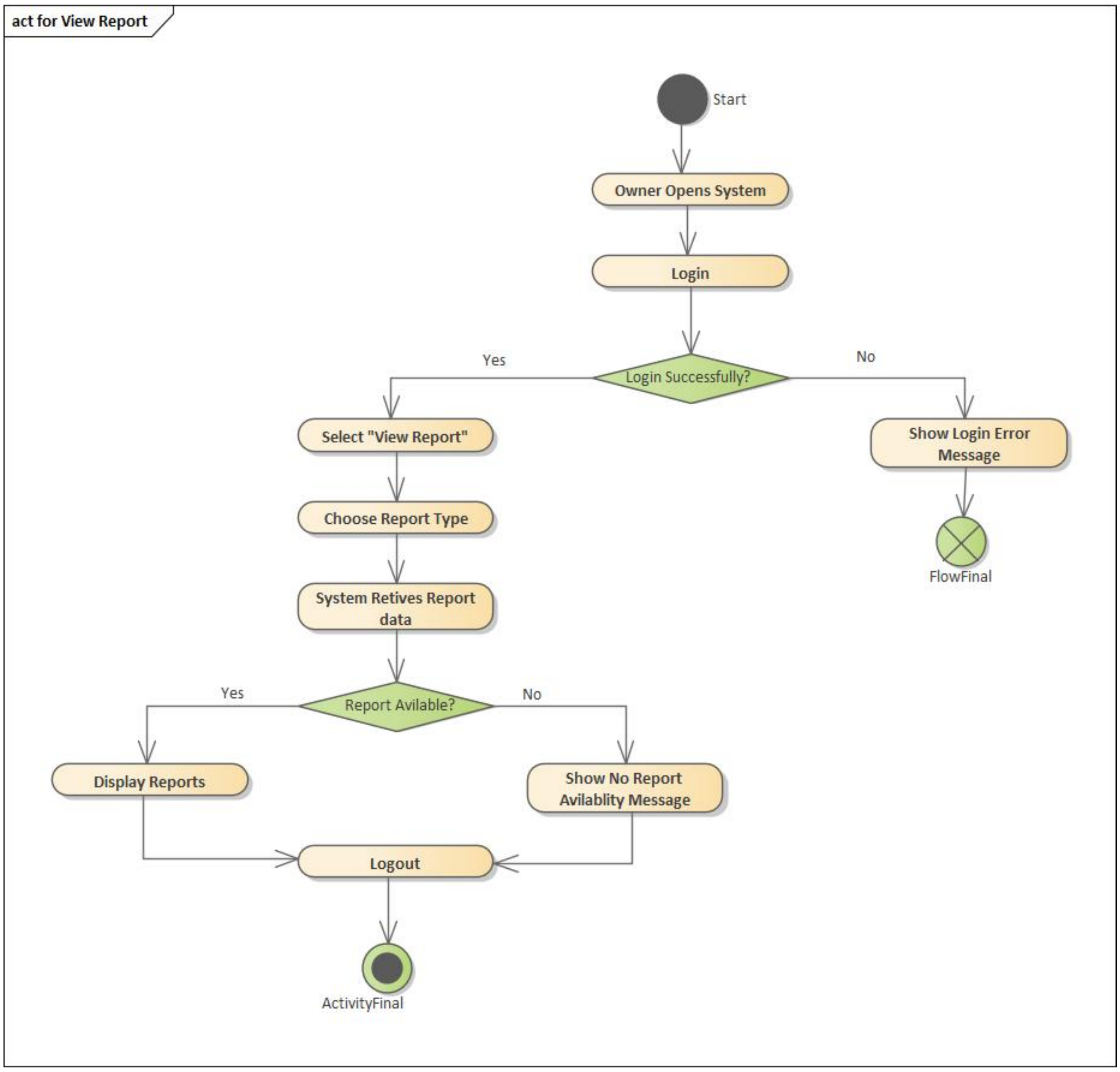


Figure_2.21: Activity diagram for use case view report.

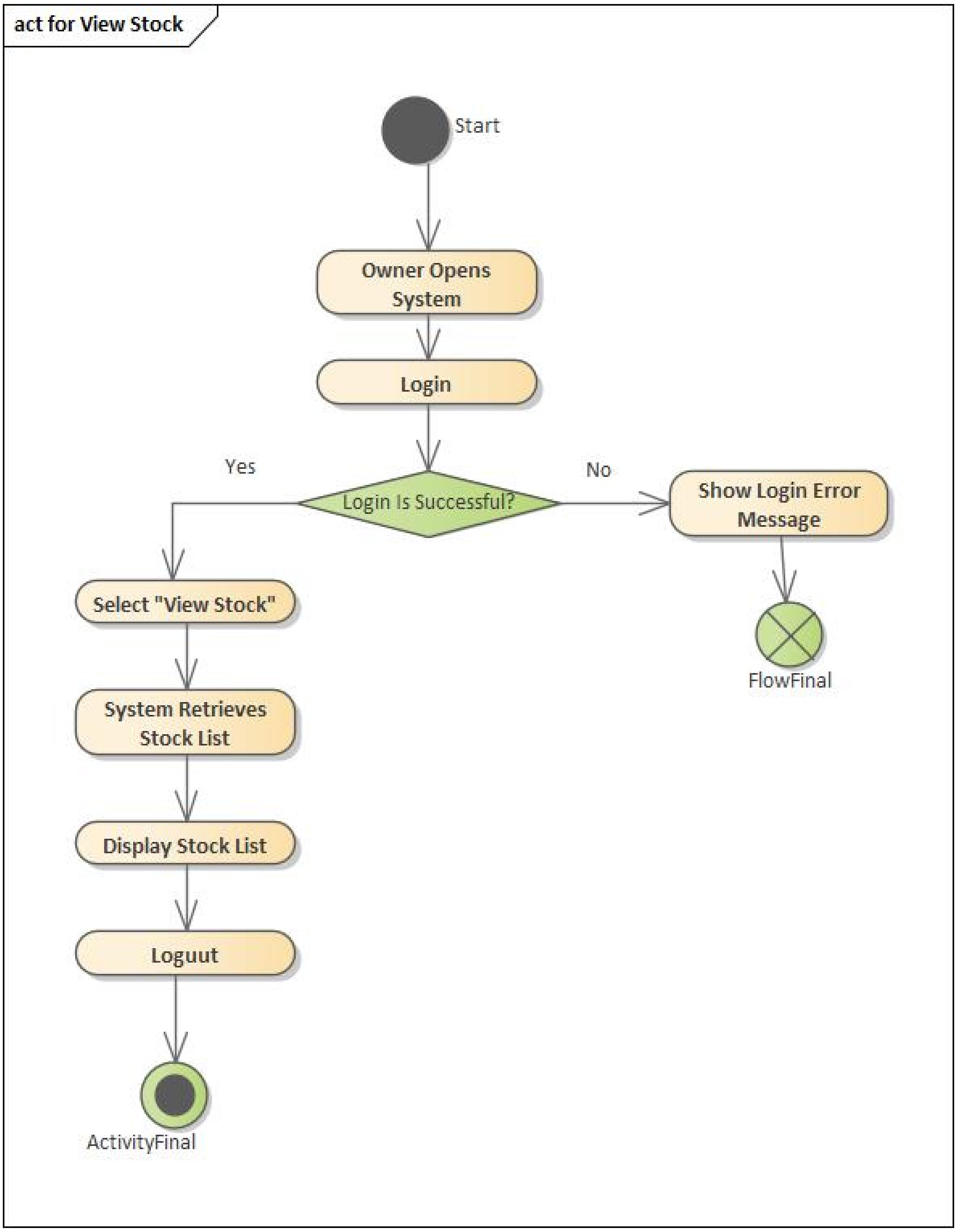


Figure_ 2.22: Activity diagram for use case view stock.

2.6.6.4 State Chart Diagram

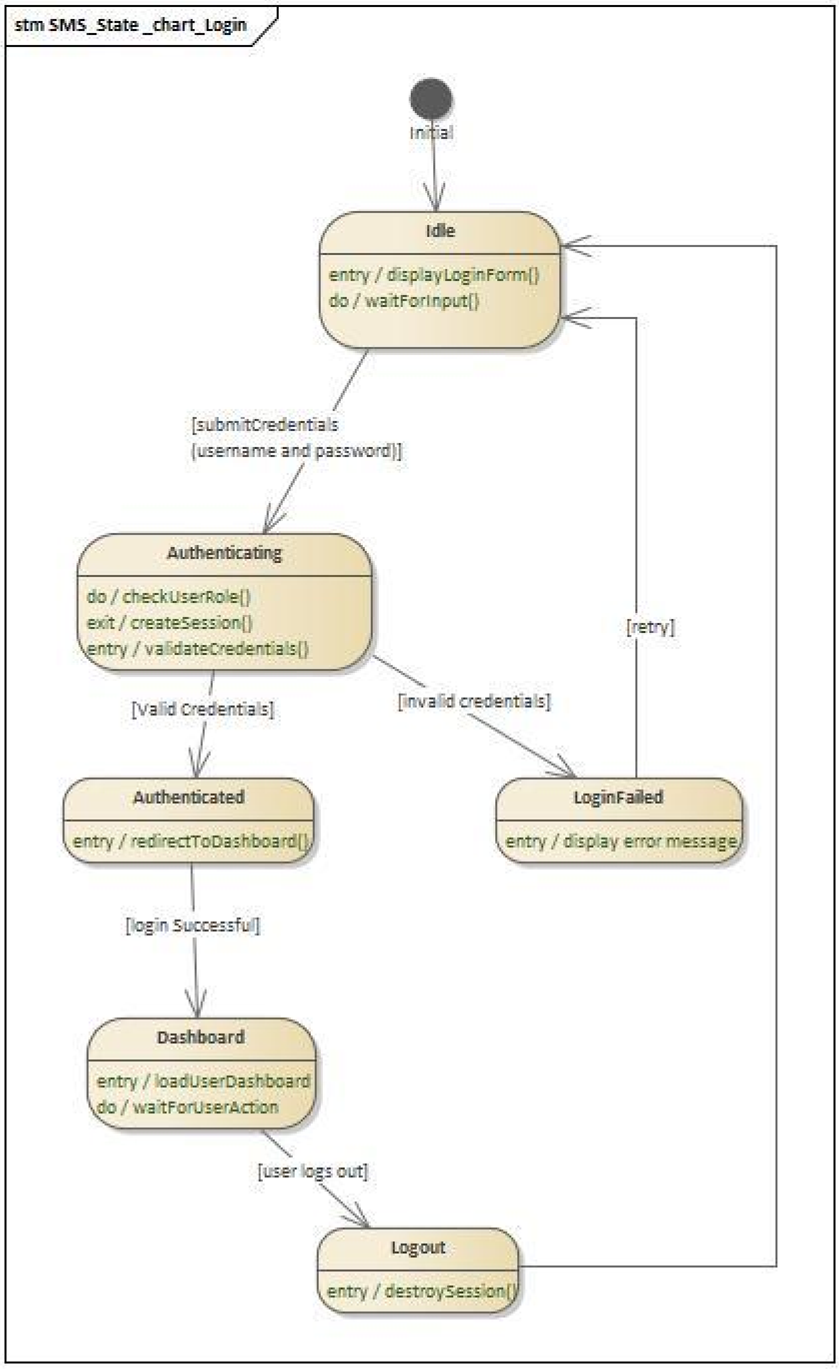


Figure_ 2.23: State chart diagram for use case login.

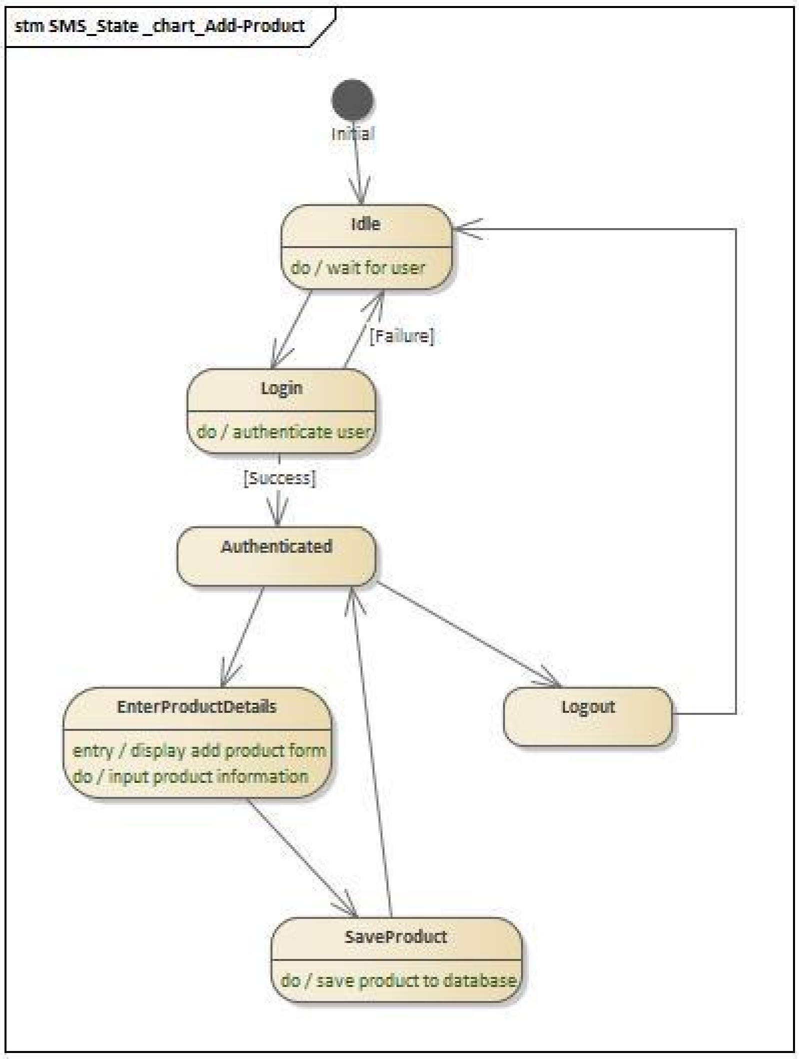

stm SMS_State _chart_Add-Product
Initial
Idle
do / wait for user
[Failure]
Login
do / authenticate user
[Success]
Authenticated
EnterProductDetails
entry / display add product form
do / input product information
Logout
SaveProduct
do / save product to database

Figure_ 2.24: State chart diagram for use case add product.

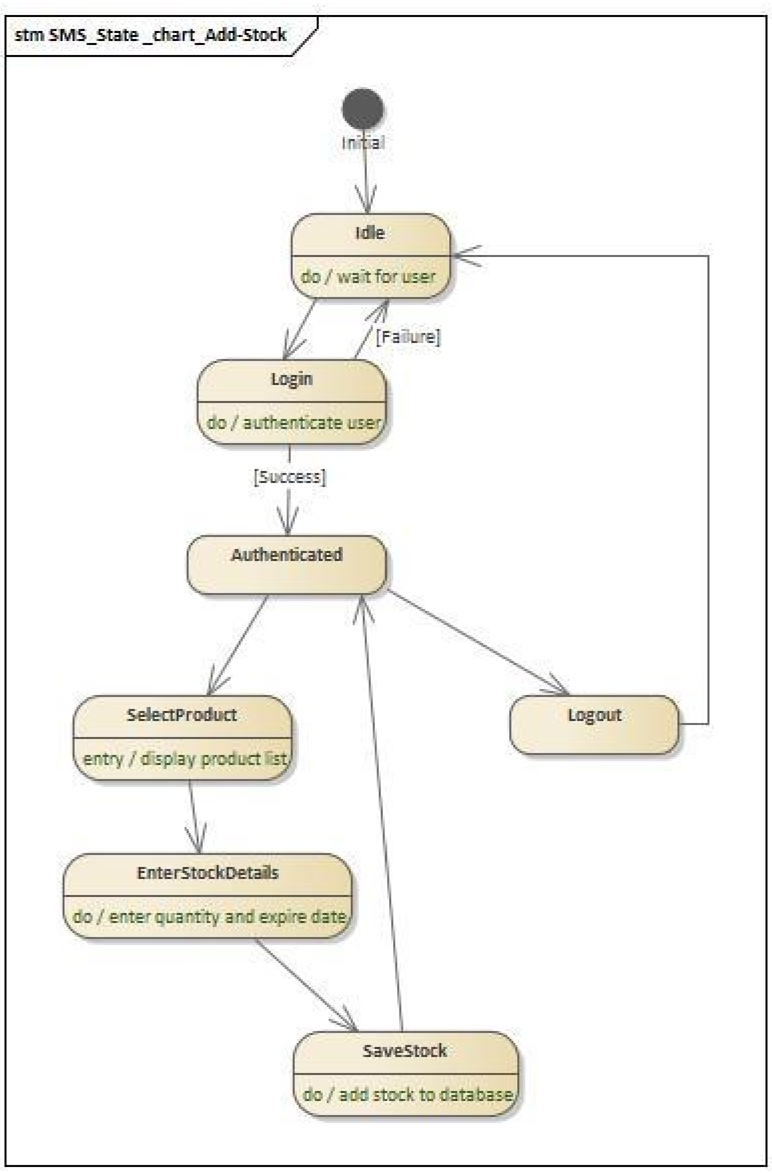


Figure_ 2.25: State chart diagram for use case add stock.

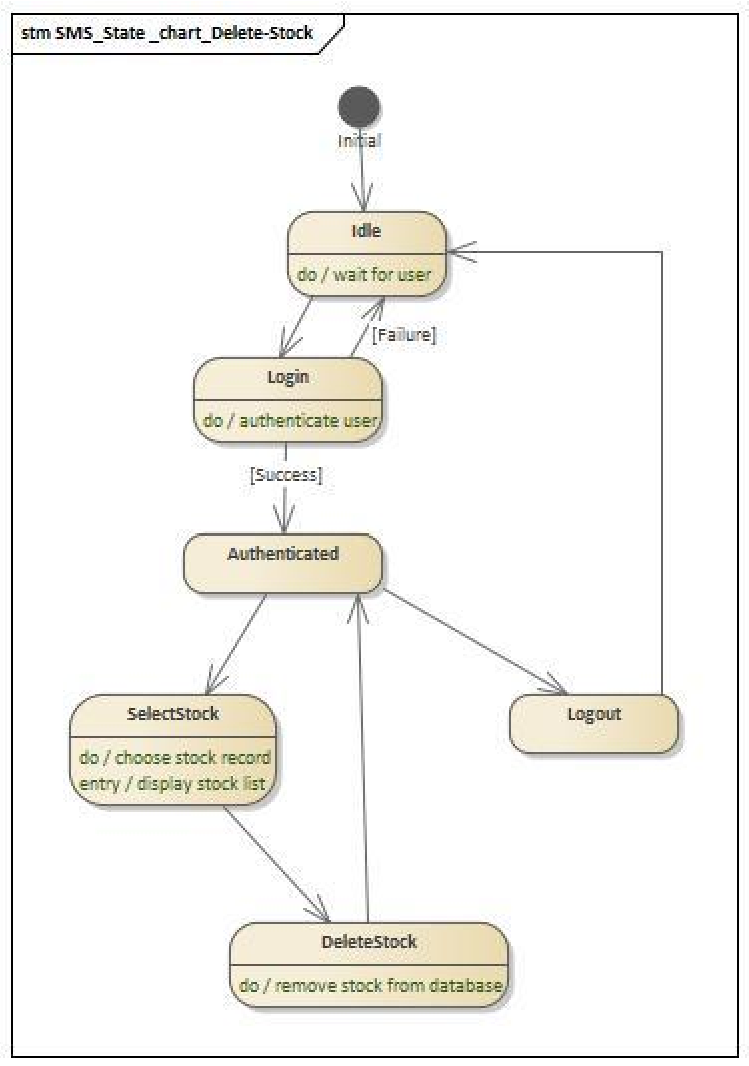


Figure_ 2.26: State chart diagram for use case delete stock.

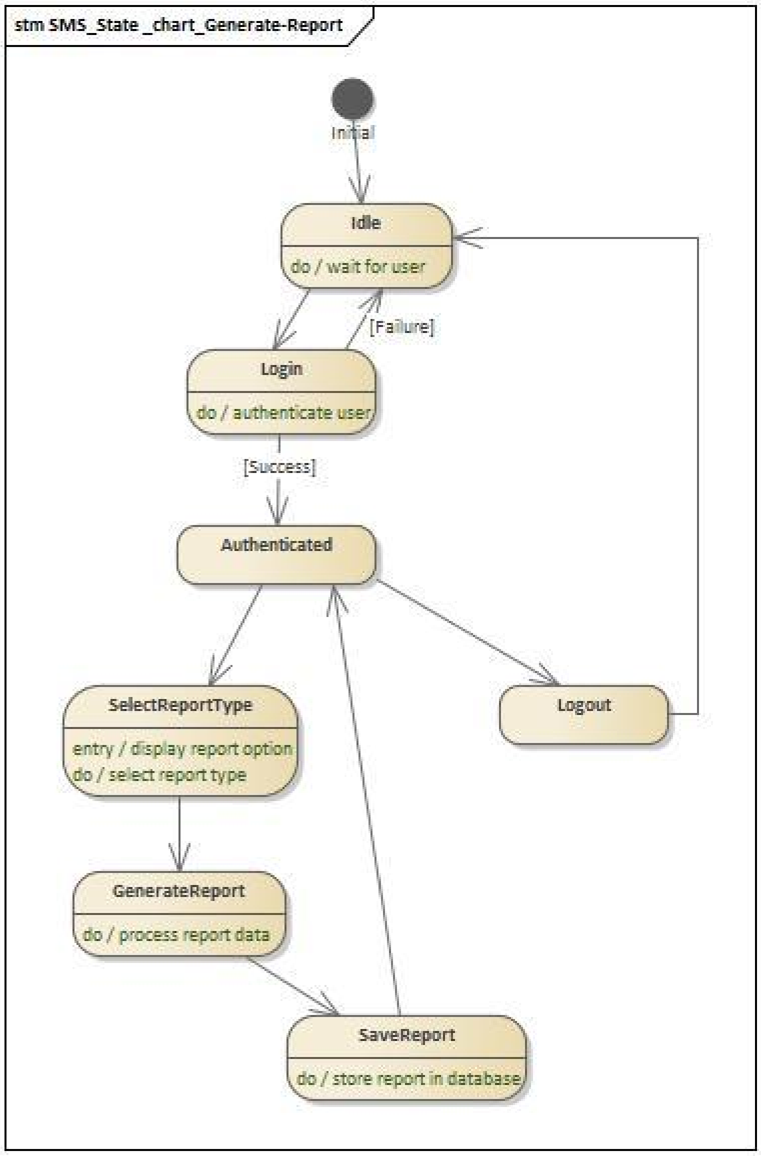


Figure_2. 27: State chart diagram for use case generate report.

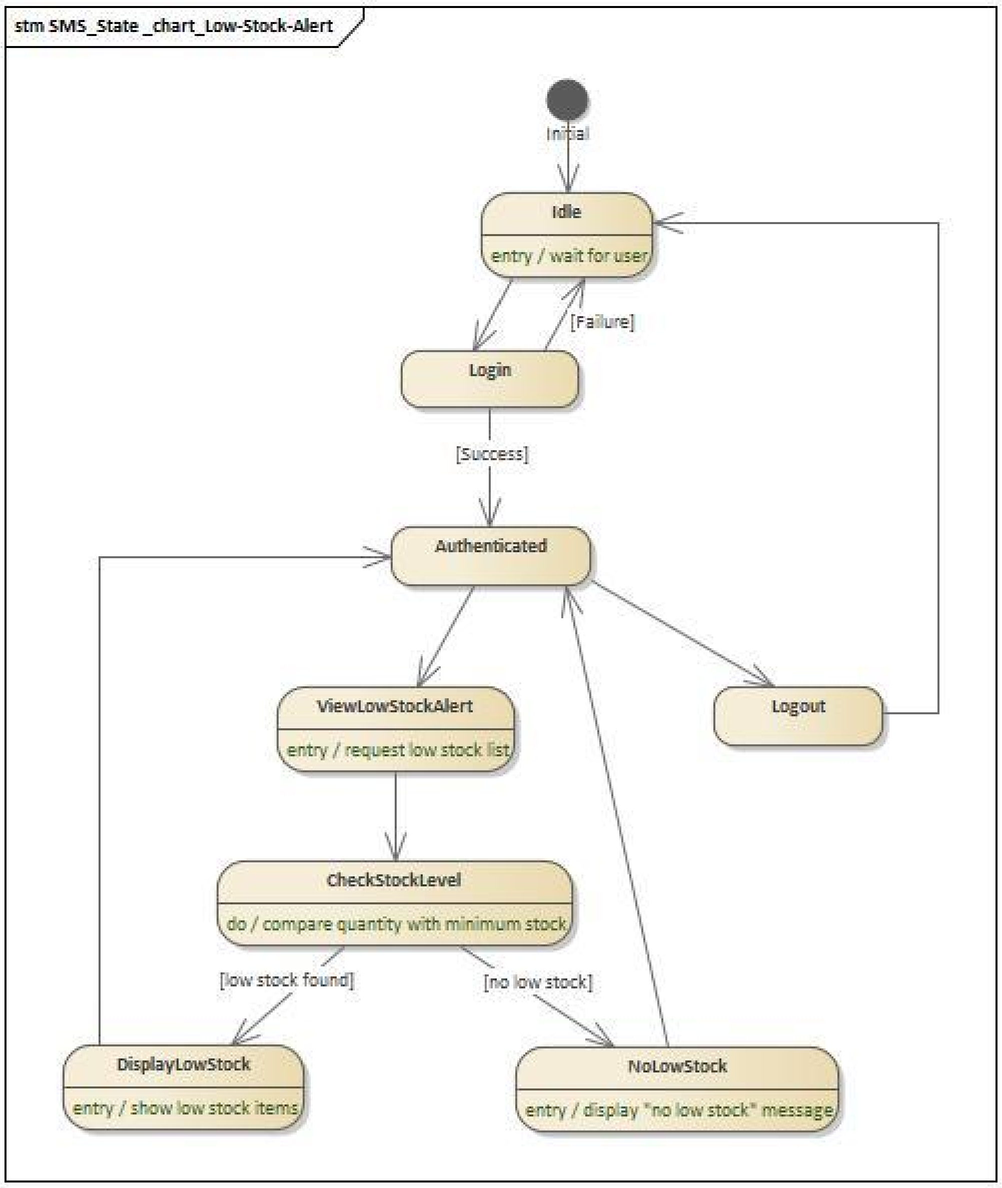

stm SMS_State _chart_Low-Stock-Alert
Initial
Idle
entry / wait for user
[Failure]
Login
[Success]
Authenticated
ViewLowStockAlert
entry / request low stock list
Logout
CheckStockLevel
do / compare quantity with minimum stock
[low stock found]
[no low stock]
DisplayLowStock
entry / show low stock items
NoLowStock
entry / display "no low stock" message

Figure_2.28: State chart diagram for use case low stock alert.

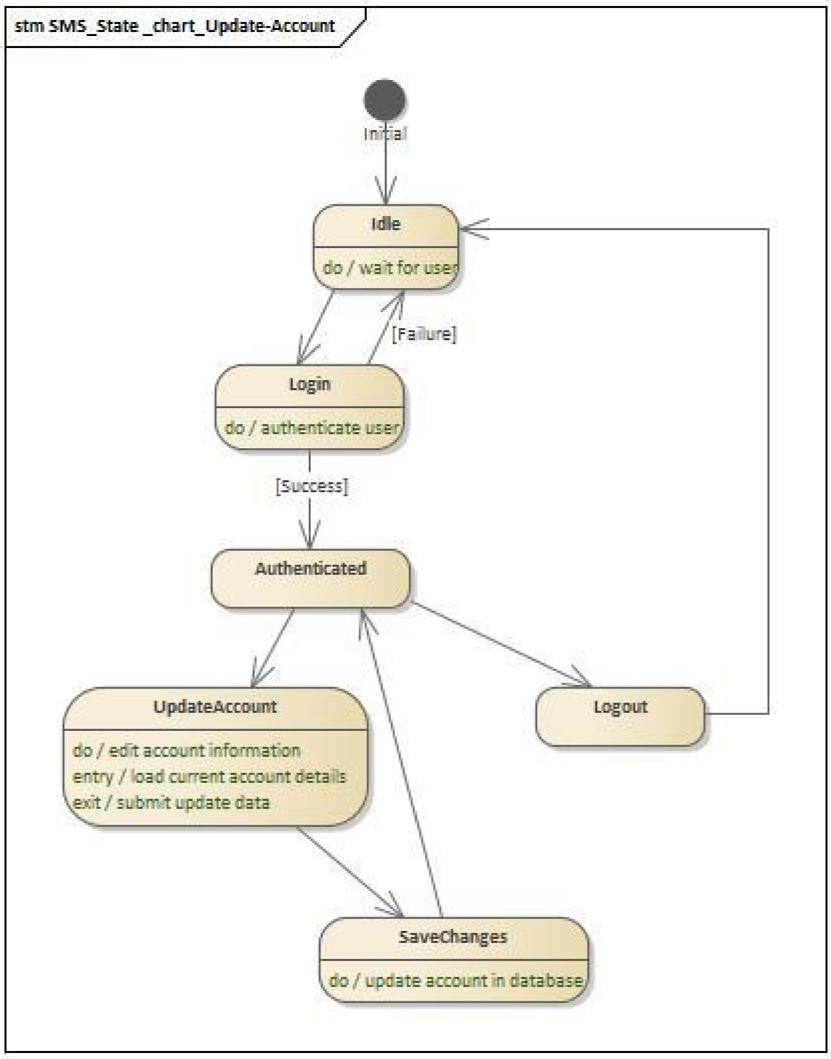

stm SMS_State _chart_Update-Account
Initial
Idle
do / wait for user
[Failure]
Login
do / authenticate user
[Success]
Authenticated
UpdateAccount
do / edit account information
entry / load current account details
exit / submit update data
Logout
SaveChanges
do / update account in database

Figure_ 2.29: State chart diagram for use case update account.

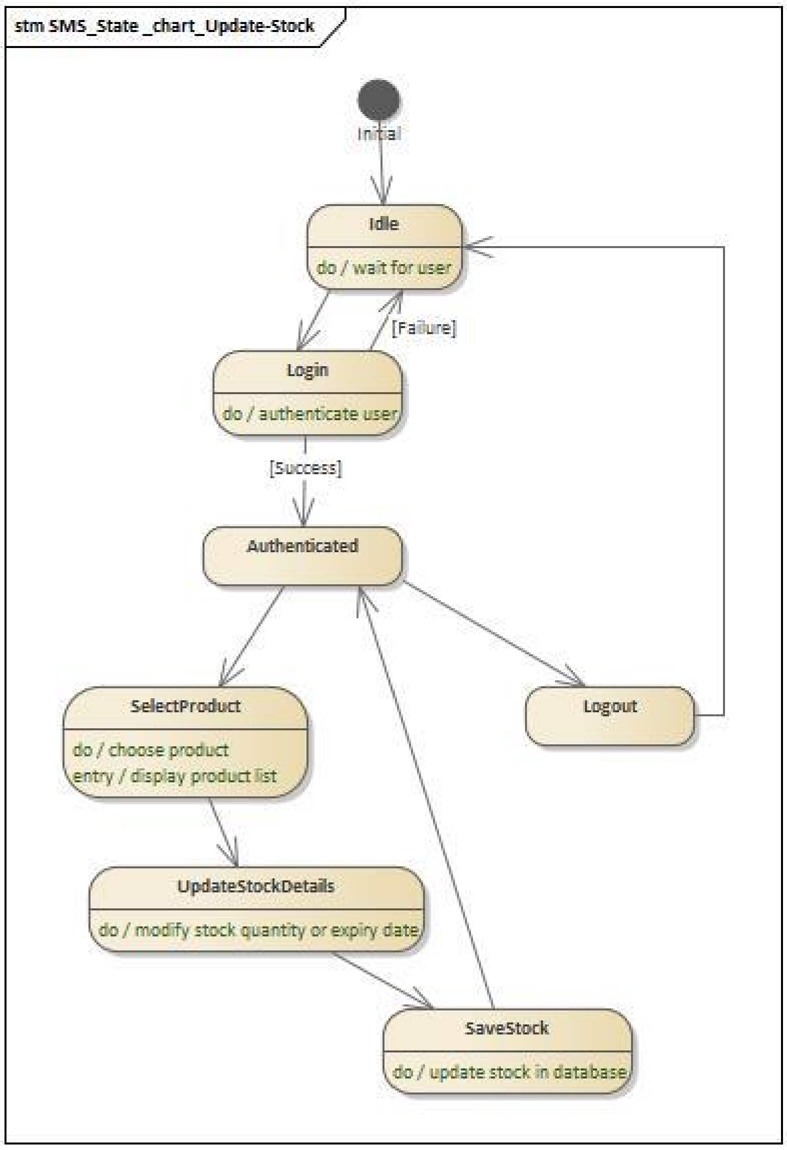


Figure_ 2.30: State chart diagram for use case update stock.

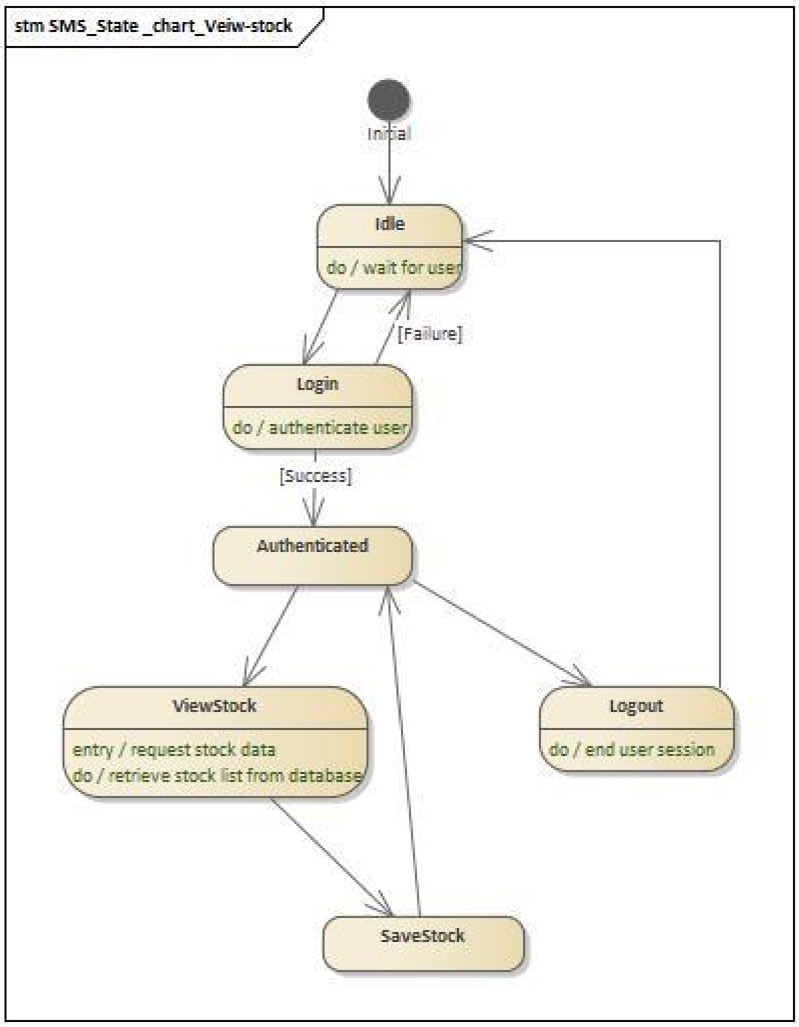


Figure_ 2.31: State chart diagram for use case view stock.

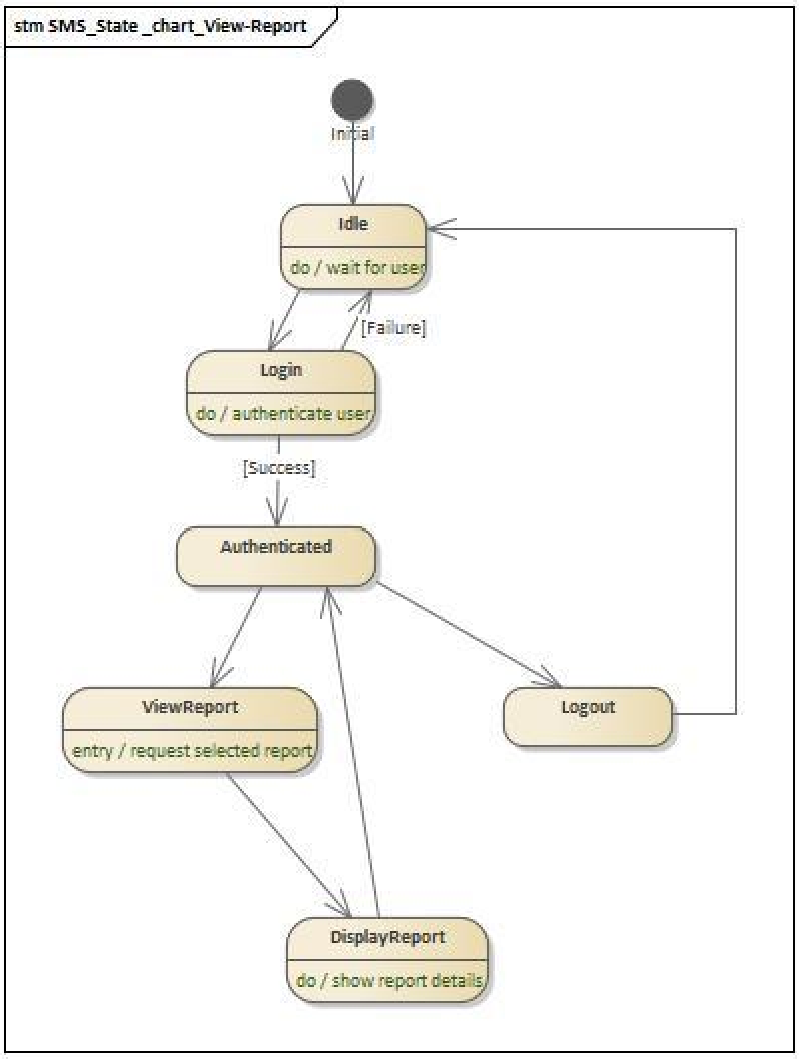


Figure_ 2.32: State chart diagram for use case view report.

2.6.6.5 Class Diagram

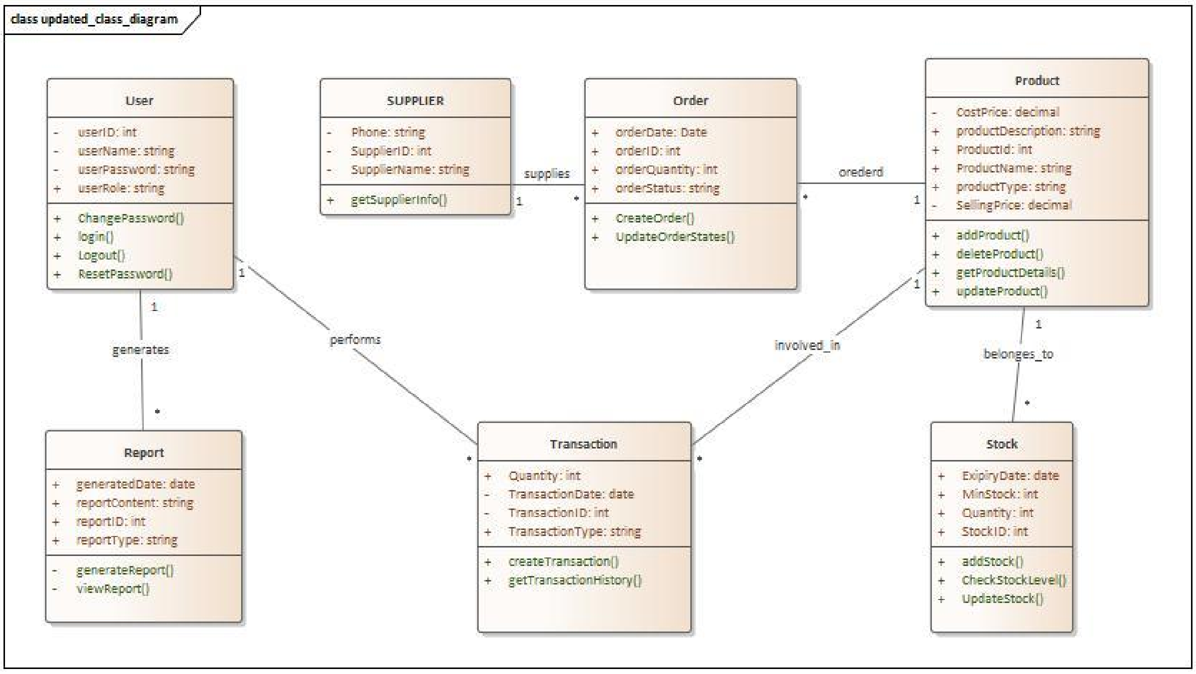


Figure_ 2.33: Class diagram

2.6.6.6 User Interface Prototyping

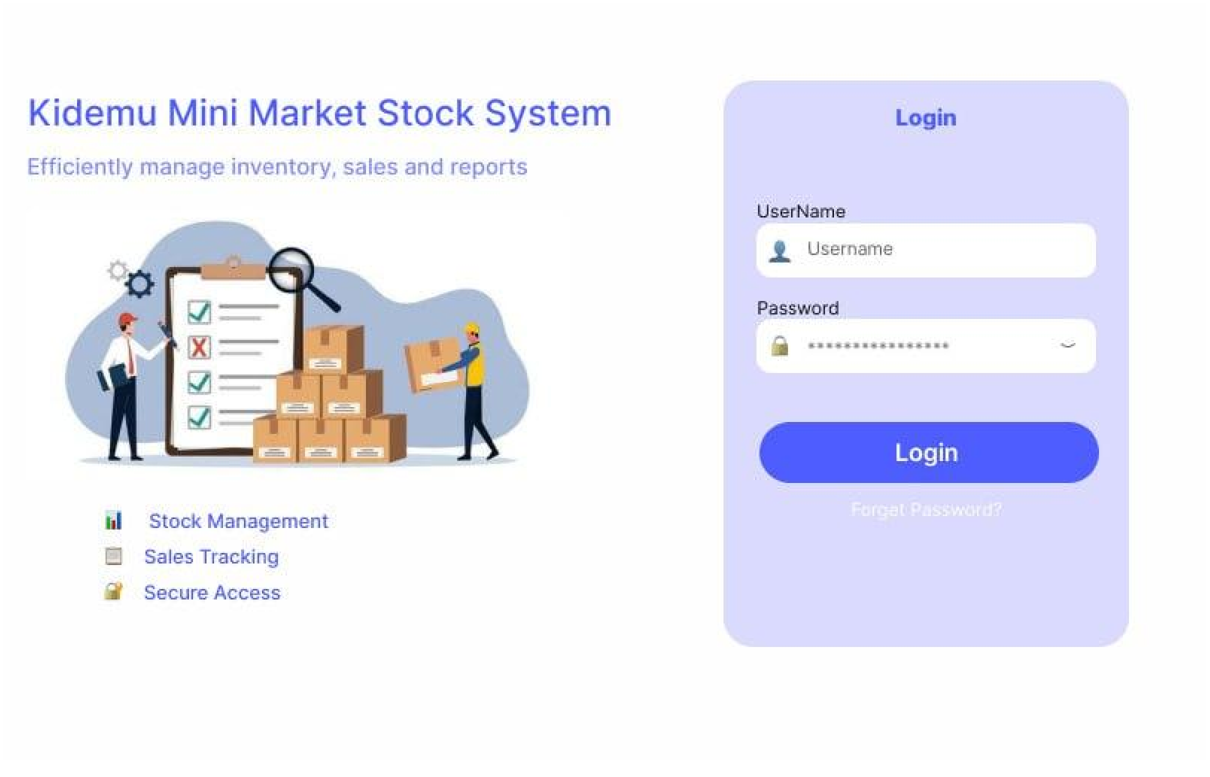


Figure_2.34: User interface prototype

CHAPTER THREE

DATABASE DESIGN

3.1 Conceptual Database Design

3.1.1 Entity Identification and Description

1. User

Description**:**

Represents individuals who are authorized to access and operate the Stock Management System. Users are categorized into predefined roles: Admin, Owner, and Accountant.

Why it exists (real systems):

In real-world systems, user entities are mandatory for authentication, authorization, and accountability. Every critical action such as recording sales, generating reports, or updating stock must be traceable to a specific user to ensure control and security.

2. Supplier

Description**:**

Represents external suppliers who provide products to Kidemu Mini Market.

Why it exists:

Stock management systems must track product sources for accountability, restocking, and auditing purposes. Even in small retail businesses, supplier information is essential for managing orders and resolving supply issues.

3. Product

Description**:**

Represents items that are sold or managed by the system, including their pricing and classification details.

Why it exists:

The product entity is the core of the stock management system. All other entities such as stock, orders, and transactions depend on product data. Without a product entity, inventory control is impossible.

4. Stock

Description:

Represents inventory information related to each product, including quantity, minimum stock level, and expiry date.

Why it exists:

In professional systems, stock information is separated from product details to allow dynamic quantity updates, expiry tracking, and alert generation without modifying product master data.

5. Order

Description:

Represents purchase orders made to suppliers for acquiring products.

Why it exists:

Order records allow the system to track restocking activities, supplier transactions, and incoming inventory. This supports accountability and ensures accurate stock updates when goods are received.

6. Transaction

Description:

Represents all stock-affecting activities such as sales, stock additions, and adjustments.

Why it exists:

Transactions provide a complete audit trail of inventory movement. In real systems, stock levels are never changed directly; they are updated through transactions to preserve historical accuracy.

7. Report

Description:

Represents system-generated outputs such as stock reports, income summaries, and alert reports.

Why it exists:

Reports transform raw data into meaningful information for decision-making. They also provide accountability by recording when and by whom reports were generated.

3.1.2 Attribute Identification and Description

User

User ID – Unique identifier for each user

User Name – Login name of the user

User Password – User authentication credential

User Role – Defines access level (Admin, Owner, Accountant)

Supplier

Supplier ID – Unique identifier of the supplier

Supplier Name – Full name of the supplier

Supplier Phone – Contact number

FName, LName – Supplier personal name components

Product

Product ID – Unique product identifier

Product Name – Name of the product

Product Type – Category/type of product

Cost Price – Purchase price

Selling Price – Selling price

Product Description – Additional product details

Stock

Stock ID – Unique identifier for stock record

Product ID – References the product

Quantity – Current available quantity

Mini Stock – Minimum allowed stock level

Expire Date – Expiration date of product

Order

Order ID – Unique order identifier

Product ID – Ordered product

Supplier ID – Supplier providing the product

Order Quantity – Quantity ordered

Order Date – Date of order

Status – Order status (pending, received)

Transaction

Transaction ID – Unique transaction identifier

Product ID – Product involved

UserID – User who performed the transaction

Transaction Date – Date of transaction

Transaction Type – Sale, stock update, adjustment

Quantity – Quantity affected

Report

Report ID – Unique report identifier

Report Type – Type of report (stock, income, alerts)

Generate Date – Date report was generated

Report Content – Report data summary

User ID – User who generated the report

3.1.3 Relationship Identification and Description

User – Generates – Report

Description:

A user can generate multiple reports, but each report is generated by one user.

Relationship Type: One-to-Many

Supplier – Supplies – Order

Description:

A supplier can supply many orders, but each order is associated with one supplier.

Relationship Type: One-to-Many

Order – Includes – Product

Description:

Each order includes one product, while a product can appear in many orders.

Relationship Type: One-to-Many

Product – Belongs To – Stock

Description:

Each product has a stock record that tracks quantity and expiry details.

Relationship Type: One-to-One (or One-to-Many depending on batches)

<u>User – Performs – Transaction</u>

Description:

A user can perform many transactions, but each transaction is performed by one user.

Relationship Type: One-to-Many

<u>Product – Involved In – Transaction</u>

Description:

A product can be involved in many transactions over time.

Relationship Type: One-to-Many

3.1.4 ER Diagram

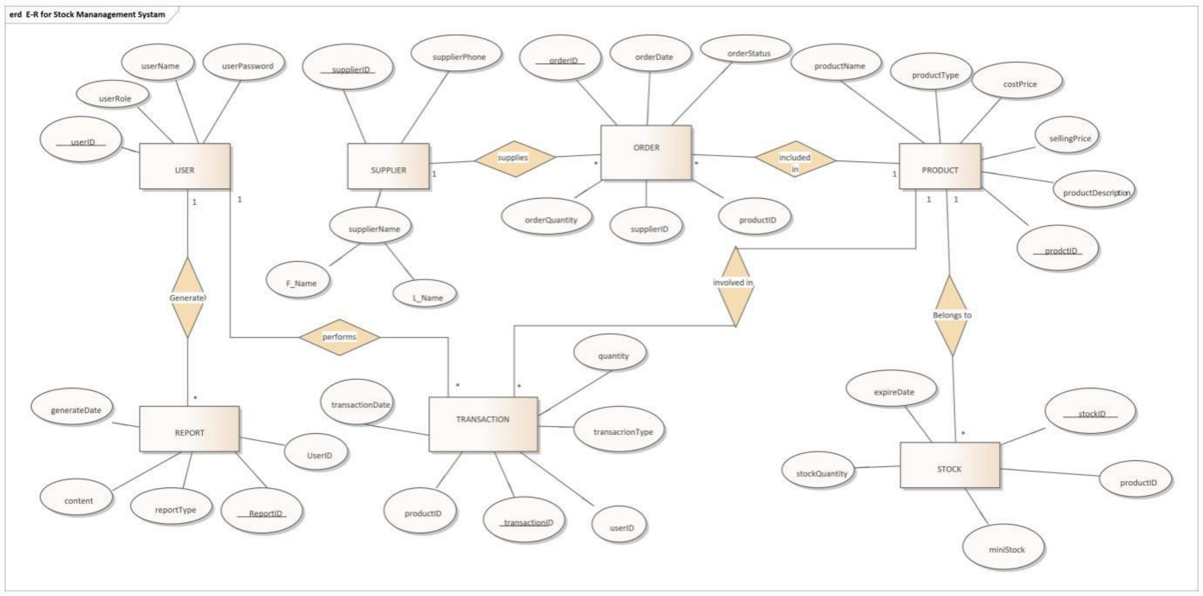
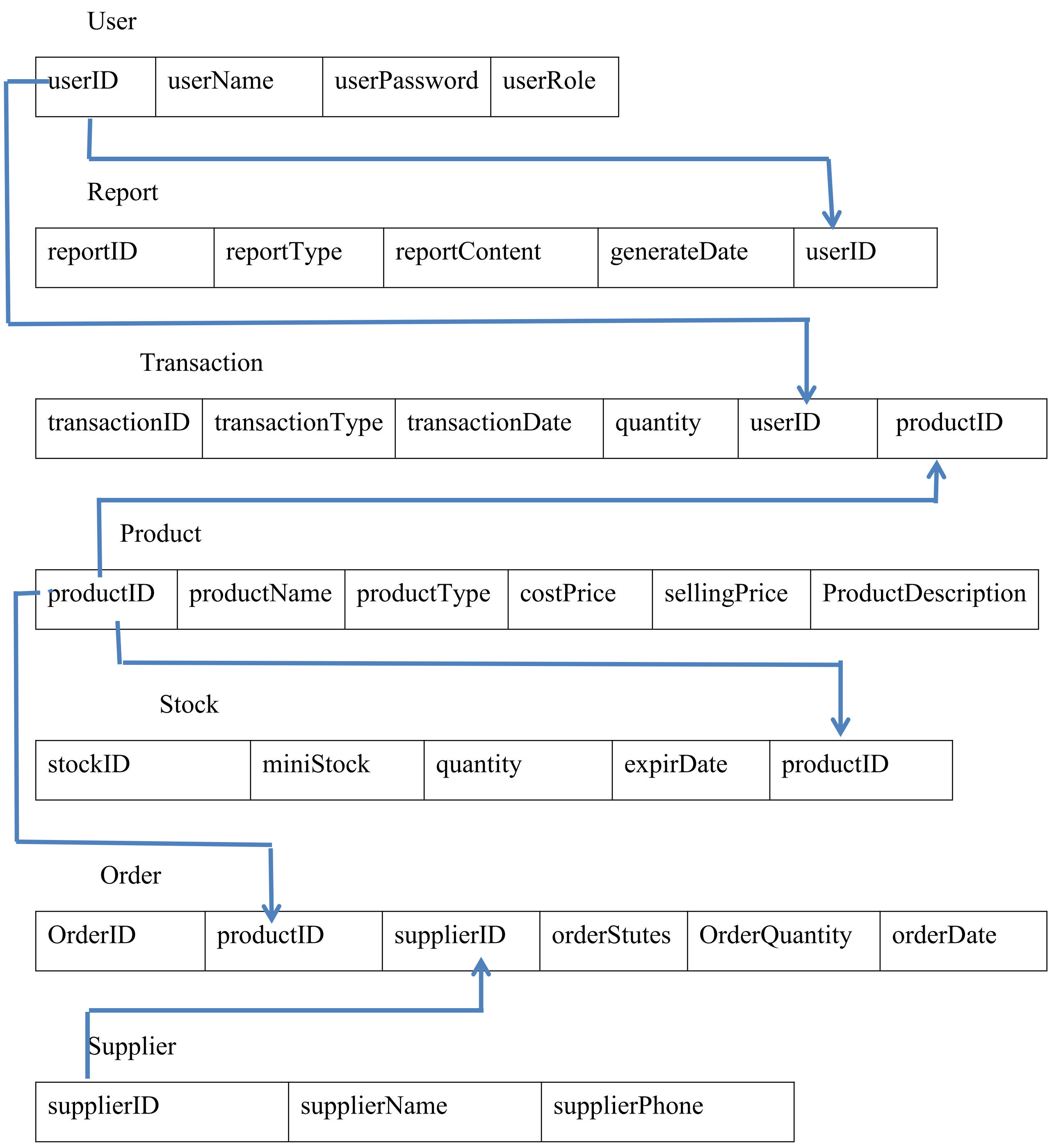


Figure_ 3.1: E-R diagram

3.2. Logical Database Design of the New System

3.2.1. ER to Table Mapping description



User

userID	userName	userPassword	userRole

Report

reportID	reportType	reportContent	generateDate	userID

Transaction

transactionID	transactionType	transactionDate	quantity	userID	productID

Product

productID	productName	productType	costPrice	sellingPrice	ProductDescription

Stock

stockID	miniStock	quantity	expirDate	productID

Order

OrderID	productID	supplierID	orderStutes	OrderQuantity	orderDate

Supplier

supplierID	supplierName	supplierPhone

Table 3.1_ E_R to table mapping

3.2.2. Validate Model using Normalization

User

userID	userName	userpassword	userRole

Report

reportID	reportType	reportContent	generateDate	userID

Transaction

transactionID	transactionDate	transactionType	quantity	userID	productID

Product

productID	productName	productType	productDisciption	costprice	sellingprice

Stock

stockID	quantity	expireDate	miniStock	productID

Order

orderID	orderStatus	orderQuantity	orderDate	productID	supplierID

Supplier

supplierID	supplierName	supplierPhone

Table 3.2_ Unnormalized table

3.2.2.1. First Normal Form (1NF)

The User, report, transaction, product, stock and order tables are already in First Normal Form because all attributes contain atomic values, there are no repeating groups, and the table has a primary key.

1NF for supplier table

supplierID	supplierFname	suplierLname	supplierPhone

Table 3.3_ Supplier1NF table

3.2.2.2. Second Normal Form (2NF)

All tables satisfy Second Normal Form

Because the tables are already in 1NF, each table has a single-attribute primary key and therefore has no partial dependency.

3.2.2.3. Third Normal form (3NF)

Report, transaction, product order and supplier tables are satisfy Third Normal Form:

The tables are already in 1NF

No partial dependencies

No transitive dependencies

All non-key attributes depend only on the primary key

3NF for user table

userID(PK)	userName	userPassword	roleID(FK)

Role

roleID(PK)	roleName

3NF for stock table

stockID(PK)	quantity	expiredate	productID(FK)

Table 3.4_ User, role and stock 3NF table

3.2.2.4. Other NF

BOYCE-CODD Normal Form (BCNF)

All tables in the system are designed following database normalization rules.

Each table is in First Normal Form (1NF) because all attributes contain atomic values and there are no repeating groups.

They are in Second Normal Form (2NF) since each table has a single-attribute primary key, so no partial dependency exists.

After removing transitive dependencies (such as separating roles and moving minimum stock to the Product table), all tables satisfy Third Normal Form (3NF).

All functional dependencies have determinants that are candidate keys, so the tables also conform to Boyce–Codd Normal Form (BCNF).

FOURTH NORMAL FORM (4NF)

1. User Table – No multi-valued attributes. Already in 4NF

2. Product Table – No multi-valued attributes. Already in 4NF

3.Stock Table – Each stockID links to one product, quantity, expiryDate. Already in 4NF

4. Transaction Table – One transactionID per product per user per transaction. Already in 4NF

5. Order Table – Currently, one orderID → one product → one supplier.

Potential issue: If an order can contain multiple products (one orderID for multiple products), this creates a multi-valued dependency.

Fix: Split into Order + OrderItem table. Then 4NF is satisfied

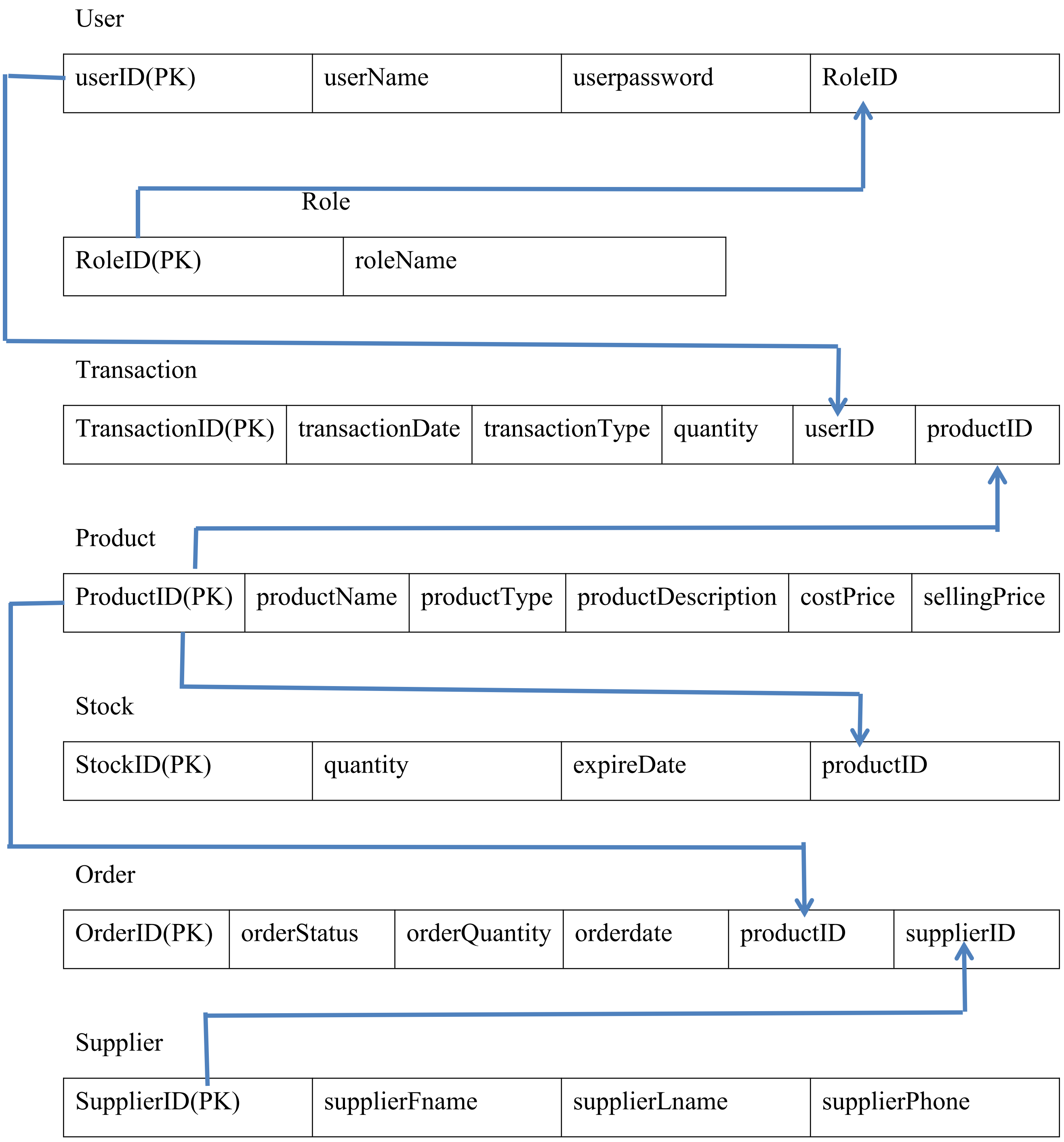
6. Supplier Table – No multi-valued attributes. Already in 4NF

7. Report Table – Each reportID maps to one content, type, and date. Already in 4NF

FIFTH NORMAL FORM (5NF)

User, Product, Stock, Transaction, Supplier, Report – Simple relationships, no complex join dependencies. Already in 5NF

3.2.3. Relational Schema with Referential Integrity after normalization



User

userID(PK)	userName	userpassword	RoleID

Role

RoleID(PK)	roleName

Transaction

TransactionID(PK)	transactionDate	transactionType	quantity	userID	productID

Product

ProductID(PK)	productName	productType	productDescription	costPrice	sellingPrice

Stock

StockID(PK)	quantity	expireDate	productID

Order

OrderID(PK)	orderStatus	orderQuantity	orderdate	productID	supplierID

Supplier

SupplierID(PK)	supplierFname	supplierLname	supplierPhone

Table 3.5_ Relational schema with referential integrity after normalization

3.3. Physical Database Design of the New System

3.3.1. Physical Design Strategy

The strategy for physical design of the database centers around simplicity, data integrity and consistency, and conformity with both system requirements and database course concepts.

The system has adopted a centralized relational database strategy that ensures storage of all system data in one database instance. This manner is most appropriate for a small retail business and it eases the workload of system maintenance, backup, and access control.

Normalization principles are applied to a level that is appropriate for the specific case, in order to diminish data redundancy and to uphold the accuracy of the whole trail. Related data is separated into different tables and linked using foreign keys. For instance, product info is kept separate from stock and transaction records.

Automated primary keys are utilized to enhance performance and simplify referencing among the tables. Data indexing is implicitly done through primary keys and unique constraints to support faster data retrieval during operations like viewing stock and generating reports.

Access control measures are taken at the database level by limiting direct access to the tables and protecting sensitive data such as user passwords and preventing direct access at the database level. User input control and database as a support for reliable system behavior through constraint enforcement are working in parallel.

This design approach guarantees that the database will be user-friendly, resource-efficient, and able to support future enhancements, and able to hold up future enhancements in case the system grows.

3.3.2. Hardware Implementation

A dedicated computer that acts as the database server hosts the database for the Stock Management System. All system data is stored by this server and the application also gets supported by the database operations.

The system is meant to be used in a small business context, which implies that powerful server hardware wouldn't be necessary. A regular computer with enough processing power, memory, and storage is sufficient to host the SQL Server database and allow system users' concurrent access.

Admin, Owner, and Accountant have the access to the system through client devices like desktop or laptop computers. The client machines are connected to the database through the application layer, which guarantees that the users will not interact with the database directly.

Data loss is prevented by basic hardware reliability measures such as secure storage and regular data backups. In addition, the architecture allows for future upgrades, like moving the database to a more powerful server if system usage increases.

This hardware arrangement is a very affordable, reliable, and suitable environment for running the Stock Management System in a mini-market setting.

CHAPTER FOUR

SYSTEM DESIGN

4.1 Introduction

This chapter presents the system design of the proposed Online Stock Management System for Kidemu Mini Market. The design phase translates the requirements and analysis defined in Chapters One and Two into a clear technical structure that can be implemented.

The system design explains:

How the system is structured

How components interact with each other

How users access system functionality

How data is processed and stored

To clearly describe system behavior and structure, UML models such as use case diagrams, class diagrams, sequence diagrams, activity diagrams, state chart diagrams, component diagrams, and deployment diagrams are used. These models act as a blueprint for system implementation and ensure alignment with both functional and non-functional requirements.

4.2 Purpose of the System

The purpose of the Online Stock Management System is to replace the existing manual notebook and paper-based stock handling process used by Kidemu Mini Market with a reliable and centralized online system.

Specifically, the system is designed to:

Manage product information and stock quantities digitally

Track expiry dates and generates alerts for low stock and expiring items

Record sales and stock transactions consistently

Generate reports such as stock reports, sales summaries, and income/cost reports

Support multiple user roles (Admin, Owner) with controlled access

By achieving these purposes, the system reduces human error, saves time, improves visibility of stock status, and supports better business decisions.

4.3 Design Goals

The design of the system is guided by the following goals:

Simplicity

The system is designed to be easy to use for users with basic computer skills. Interfaces are kept clear and straightforward.

Accuracy and Consistency

Stock levels, transactions, and reports are automatically processed to ensure accurate and consistent data.

Modularity

The system is divided into independent subsystems such as authentication, stock management, alerts, and reporting, making it easier to maintain and extend.

Security and Access Control

Role-based access ensures that:

Admin manages system operations

Owner manages stock, views alerts and stock status

Accountant accesses reports read only

Efficiency

The system responds quickly to daily operations such as viewing stock, recording sales, and generating reports.

Scalability and Maintainability

Although designed for one mini market, the architecture allows future improvements without redesigning the entire system.

Reliability

Proper database relationships and backup support reduce data loss and operational risks.

4.4 Current Software Architecture

The current system used by Kidemu Mini Market is fully manual and non-computerized.

Architectural Overview

The existing architecture relies on human effort and physical records:

Input Layer

Product details, stock quantities, sales, and expiry dates are written manually in notebooks and papers.

Processing Layer

Calculations for stock updates, income, and cost are done manually or using calculators.

Storage Layer

Information is stored only in notebooks and papers, with no backup or security.

Output Layer

Reports are prepared manually by reviewing handwritten records.

Main Components

Owner

Records products, updates stock, checks shelves, and identifies expired items manually.

Accountant

Reviews handwritten records and prepares monthly summaries.

Limitations

High risk of human error

No real-time stock visibility

No alerts or automation

No security or backup

Time-consuming reporting

These limitations justify the need for a computerized online system.

4.5 Proposed Software Architecture

The proposed system is an online web-based Stock Management System designed using a simple layered architecture suitable for a small retail business.

Users access the system through a web browser, while business logic and data management are handled on the server side.

High-Level Architecture

The system is organized into three main layers:

1. Presentation Layer (User Interface)

Handles all user interactions through web pages.

2. Application Logic Layer

Implements business rules, validation, alerts, and report generation.

3. Data Layer

Manages persistent storage using a centralized database.

4.5.1. Subsystem Decomposition

The system is decomposed into the following subsystems:

1. User Interface Subsystem

Provides login pages, dashboards, forms, and report views

Displays low-stock and expiry alerts

Accepts input from Admin, Owner

Acts as the entry point to the system

2. Authentication and Authorization Subsystem

Validates user login credentials

Enforces role-based access control

Controls login, logout, and account access

3. Product Management Subsystem

Registers new products

Updates existing product information

Deletes products when required

Stores product details such as name, type, and pricing

4. Stock Management Subsystem

Manages stock quantities and minimum stock levels

Tracks expiry dates

Generates low-stock and expiry alerts primarily for the Owner

Ensures accurate inventory status

5. Transaction Management Subsystem

Records sales transactions

Updates stock after each sale or stock addition

Maintains transaction history

6. Report Management Subsystem

Generates stock, sales, income, cost, and expiry reports

Provides read-only access to reports for the Accountant

7. Database Subsystem

Stores all system data securely

Maintains relationships between entities

Supports data consistency and backup operations

4.5.2. Component Diagram

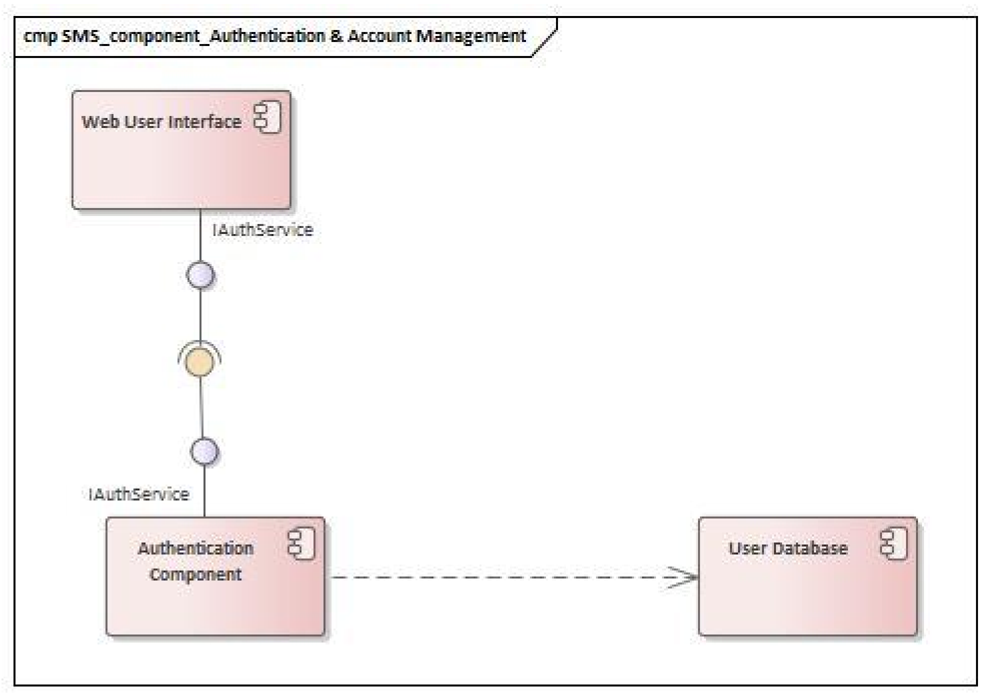


Figure 4.1: Component diagram for component authentication and account management

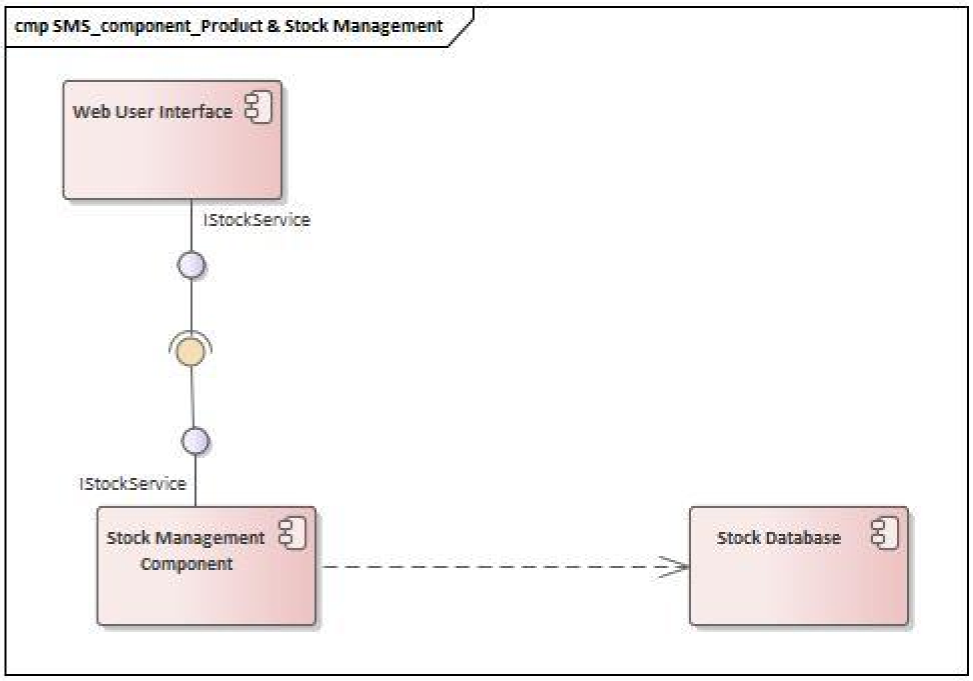

cmp SMS_component_Product & Stock Management
Web User Interface
IStockService
IStockService
Stock Management Component
Stock Database

Figure 4.2: Component diagram for component product and stock management

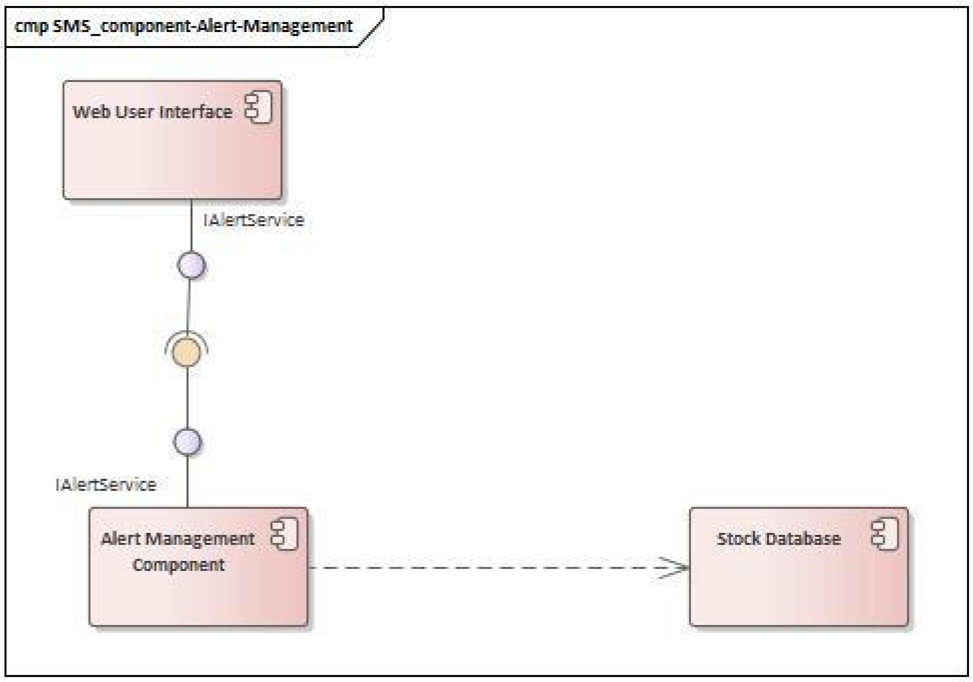

cmp SMS_component-Alert-Management
Web User Interface
IAlertService
IAlertService
Alert Management Component
Stock Database

Figure 4.3: Component diagram for component alert management

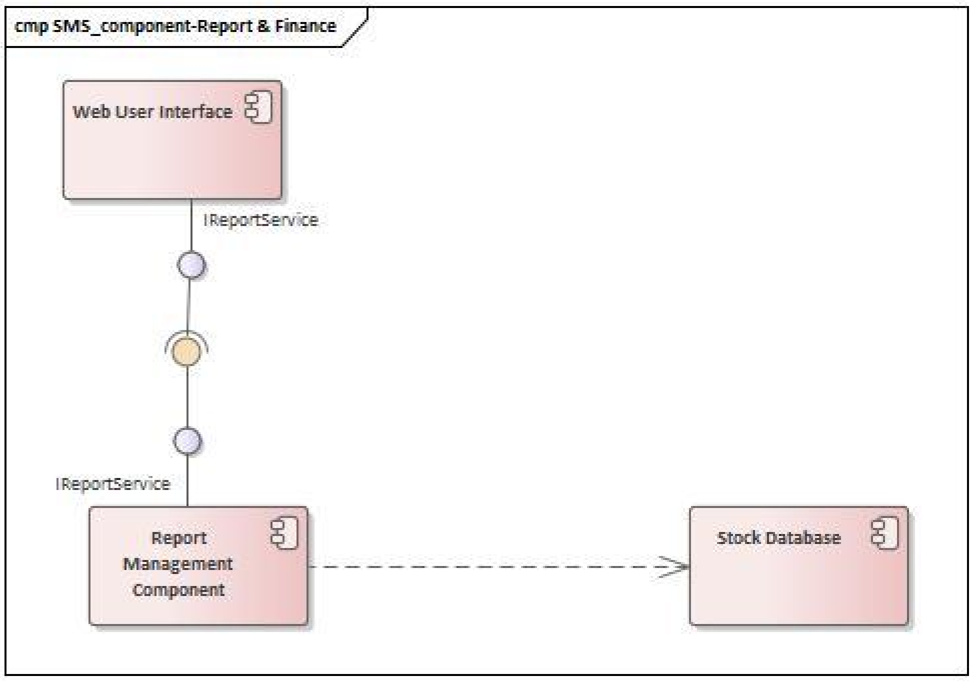

cmp SMS_component-Report & Finance
Web User Interface
IReportService
IReportService
Report Management Component
Stock Database

Figure 4.4: Component diagram for component report and finance

4.5.3. Deployment Diagram

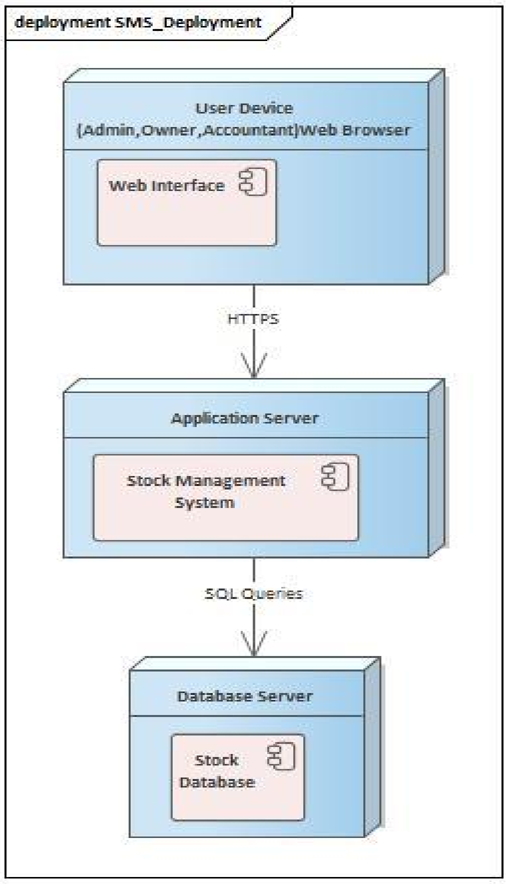


Figure 4.5: Deployment diagram

4.5.4. Database Diagram

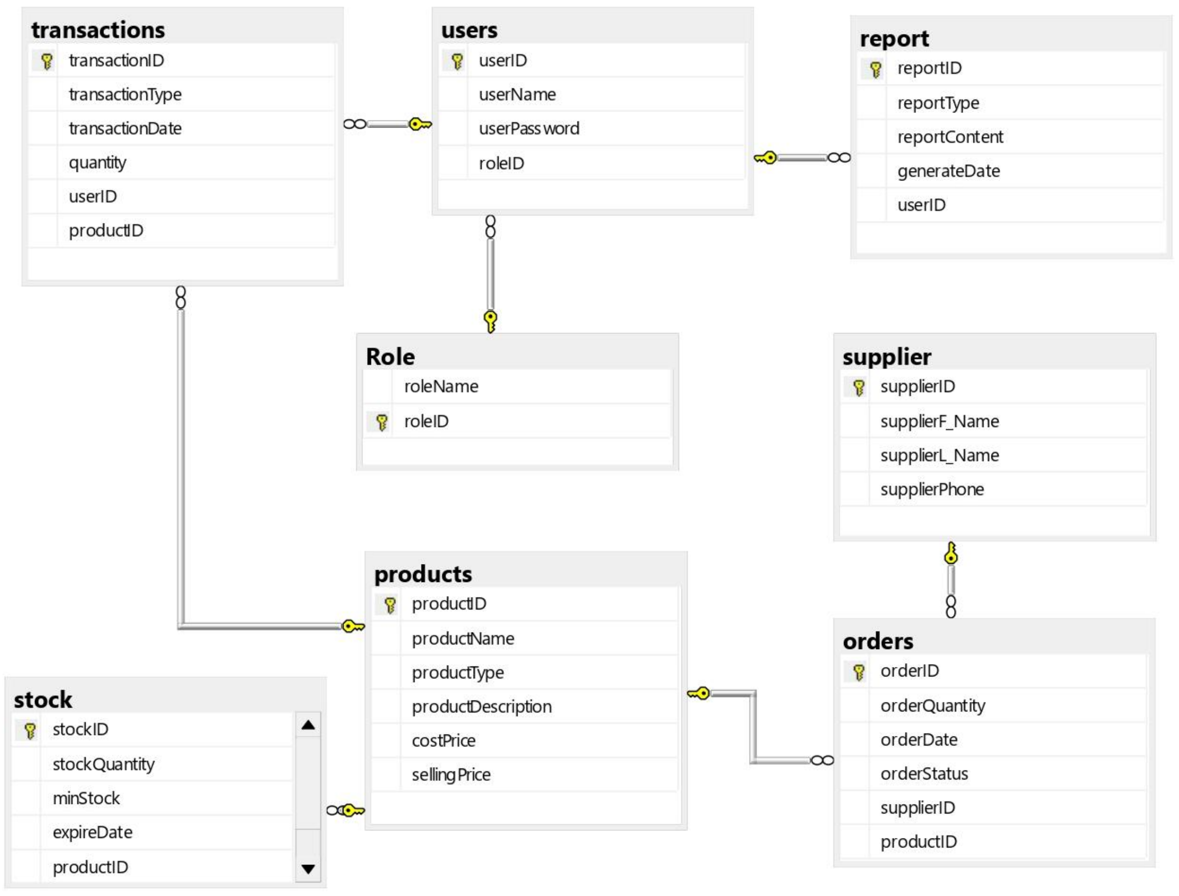


Figure 4.6: Database Diagram

4.5.5. Persistent Data Management

The system uses a relational database to store all persistent data.

CRUD (Create, Read, Update, and Delete) operations are performed through controlled application logic. Stock levels are updated after sales and stock changes to ensure accuracy.

Constraints and relationships are enforced to prevent invalid or inconsistent data.

4.5.6. Access Control and Security

Access control is implemented through authentication and role-based authorization.

Users must log in using valid credentials

Admin users have full system access

Owners can view stock and alerts

Accountants can view reports only

This approach protects sensitive data and prevents unauthorized operations.

4.5.7. Global Software Control

The system follows an event-driven control flow.

All operations are initiated by user actions such as logging in, recording a sale, or requesting a report. Each request is processed by the appropriate component and results are returned to the user interface.

This control structure ensures predictable and consistent system behavior.

4.5.8. Boundary Conditions

The system handles exceptional and limiting conditions gracefully.

Invalid login attempts are rejected with error messages

Sales are blocked when stock is insufficient

Alerts are shown when stock is low or products are near expiry

Report requests return informative messages when no data is available

These boundary conditions improve system reliability and user experience.

4.6. Refined ER Diagram

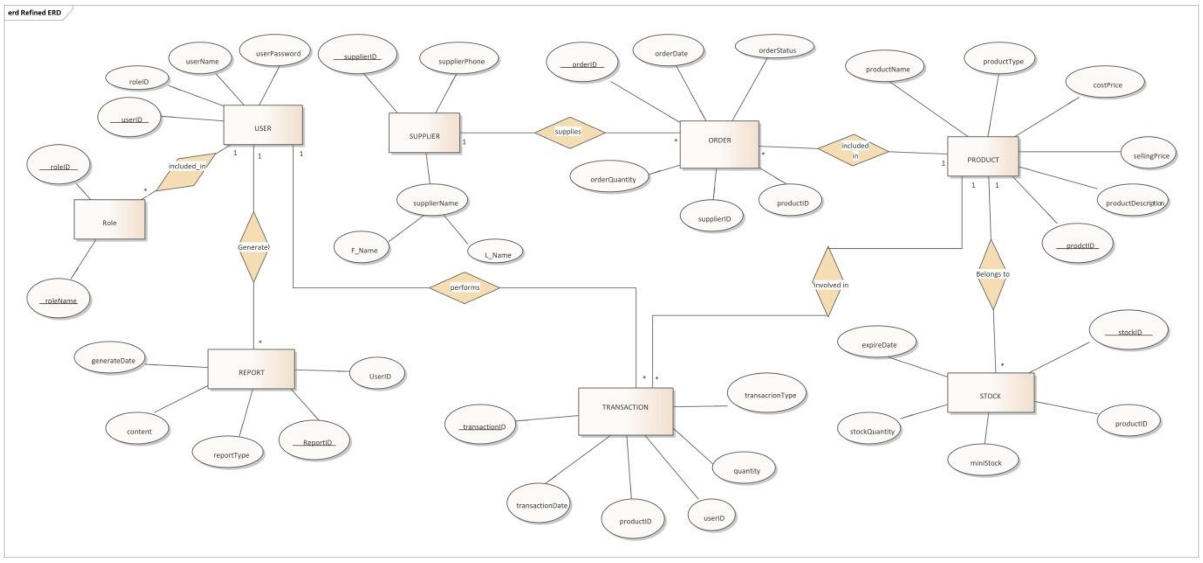


Figure 4.7: Refined ER Diagram

CHAPTER FIVE

IMPLEMENTATION AND TESTING

5.1 Test Procedure

Testing of the Stock Management System was carried out to confirm that the implemented features operate correctly and support the daily activities of a small retail shop. Testing focused on core system functions such as product management, stock updates, sales recording, and report generation.

Testing was performed using sample data that reflects the actual products, stock levels, and sales operations of the shop.

Unit Testing

Unit testing was performed by testing each major function of the Stock Management System separately.

Modules Tested:

User Login

Verified that only registered users can log in.

Tested incorrect usernames and passwords to ensure access is denied.

Product Management

Tested adding new products with valid data.

Tested invalid inputs such as negative prices and empty product names.

Verified that products are saved correctly in the database.

Stock Management

Tested updating stock quantities.

Verified that the system prevents negative stock values.

Tested minimum stock threshold validation.

Sales Recording

Tested recording sales for available products.

Verified that sales quantities reduce stock correctly.

Purpose:

To ensure each individual function works correctly before interacting with other parts of the system.

Integration Testing

Integration testing verified that related system components work together correctly.

Scenarios Tested:

When a sale is recorded:

The transaction is saved in the transactions table.

The corresponding product stock quantity is reduced automatically.

Low-stock alerts are triggered when minimum stock levels are reached.

When reports are generated:

Data is correctly retrieved from stock and transaction records.

Reports reflect accurate quantities and summaries.

Purpose:

To ensure that sales, stock management, and reporting modules operate as an integrated system.

System Testing

System testing evaluated the complete Stock Management System using realistic shop activities.

Test Scenario:

Login → Add / Update Products → Record Sales → Update Stock → View Alerts → Generate Reports → Logout

The system was tested using sample products, stock quantities, and sales similar to those used in the real shop.

Purpose:

To confirm that the system performs correctly as a whole and supports daily shop operations.

User Acceptance Testing (UAT)

User Acceptance Testing was conducted by the project team acting in the roles of system users (Owner and Accountant), based on defined system requirements.

Activities Performed:

Recording sales

Checking stock levels

Viewing income and cost reports

Verifying role-based access control

Outcome:

The system met all defined functional requirements and was considered suitable for managing stock and sales activities compared to the manual notebook system.

5.2 User Manual Preparation

A simple user manual was prepared to guide users in operating the Stock Management System.

Manual Contents:

System overview

Login and logout procedures

Product and stock management

Sales recording

Viewing alerts

Report generation

Data backup instructions

Common input errors and solutions

Design Approach:

Simple language

Step-by-step instructions

Screenshots of system screens

Purpose:

To allow users to operate the system independently with minimal training.

5.3 Training and Installation

Since the system is a prototype, installation was performed in a controlled environment:

Training

Training focused on explaining how the system should be used in daily shop operations.

Training Covered:

Navigating system menus

Managing products and stock

Recording sales

Viewing alerts and reports

Handling common input errors

Security practices such as password protection

Training Method:

Demonstration of system features

Practice sessions using sample data

Outcome:

Users were able to perform all required tasks using the system.

Installation

The system installation process included:

Installing the application on a Windows computer

Creating and configuring the database using SQL Server

Entering sample product, stock, and user data

Verifying system functionality after setup

5.4 Startup Strategy

A gradual startup strategy was defined to ensure smooth adoption of the Stock Management System.

Startup Steps:

Step 1: Initial Setup

System installation

Entry of existing product and stock data

Creation of user accounts

Step 2: Parallel Usage

Manual records and the digital system used together for a short period

Results compared to verify correctness

Step 3: Full Transition

Manual notebook usage discontinued

System used as the primary management tool

Step 4: Data Backup

Regular database backups planned to prevent data loss

Step 5: Support Period

Developers remain available for minor fixes and clarifications

Result:

The strategy minimizes disruption, reduces risk, and improves confidence in the system.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

6.1. Conclusion

This project was carried out to solve the problems faced by Kidemu Mini Market due to the use of a manual stock management system. In the existing system, all product records, sales, expiry dates, and stock quantities were written in notebooks and papers. This manual method caused many problems such as calculation errors, loss of records, difficulty in tracking expired products, and delays in preparing reports.

After studying the current system and collecting requirements through observation and interviews, a computerized Stock Management System was designed. The proposed system helps to manage products, stock, sales, and reports in a digital way. It updates stock after sales, shows alerts for low stock and expired products, and generates accurate reports.

Different UML diagrams such as use case diagrams, activity diagrams, sequence diagrams, class diagrams, and ER diagrams were used to clearly explain how the system works and how data is stored. These diagrams help developers understand the system structure and behavior before implementation.

The system also supports different user roles (Admin, Owner, and Accountant). Each user can only access the functions allowed for their role. This improves system security and prevents unauthorized changes.

In general, the proposed system successfully meets the project objectives. It reduces human errors, saves time, improves accuracy, and helps the owner make better business decisions. The project proves that even a simple digital system can greatly improve inventory management in small businesses.

6.2. Recommendation

Based on the results of this project, the following recommendations are suggested:

Use the System in Real Work

Kidemu Mini Market should use the proposed Stock Management System instead of the manual notebook system. This will help the business manage stock properly and avoid losses caused by expired or missing products.

Provide Basic Training

The Admin, Owner, and Accountant should receive simple training on how to:

Log in to the system

Add and update products

Record sales

View alerts and reports

Training will reduce mistakes and help users feel confident using the system.

Perform Regular Data Backup

To avoid data loss due to power interruption or system failure, regular database backups should be performed. Backups can be saved on external storage or cloud services.

Improve the System in the Future

In the future, the system can be expanded by adding:

Barcode scanning

Supplier management

Mobile or web version

POS system integration

Advanced reports and graphs

These improvements will make the system more powerful and flexible.

6.3 Reference

Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill.

Sommerville, I. (2016). Software Engineering. Pearson Education.

Kendall, K. E., & Kendall, J. E. (2011). Systems Analysis and Design. Pearson.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). Database System Concepts. McGraw-Hill.

Lecture notes from Department of Software Engineering, Wachemo University.

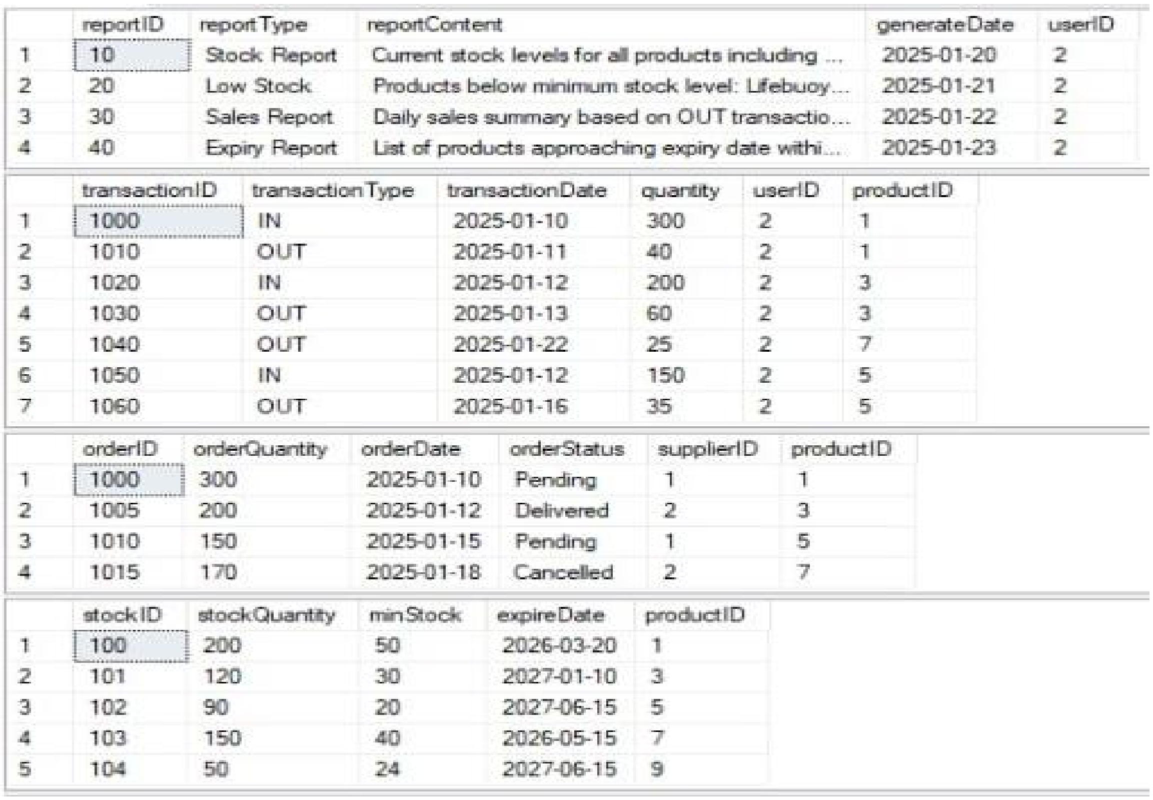
6.4. Appendix

```
create database Stock_Management_System_SQL_0
use Stock_Management_System_SQL_0
CREATE TABLE Role(
roleName VARCHAR(120) NOT NULL,
roleID int Identity(3,2) PRIMARY KEY not null
);
CREATE TABLE users(
userID INT Identity(2,2) PRIMARY KEY NOT NULL,
userName VARCHAR(30) NOT NULL unique,
userPassword VARCHAR(50) NOT NULL ,
roleID int references Role(roleID) NOT NULL
);
CREATE TABLE supplier(
supplierID INT Identity(1,1) PRIMARY KEY NOT NULL,
supplierF_Name VARCHAR(30) not null,
supplierL_Name VARCHAR(30) not null,
supplierPhone VARCHAR(15) not null
);
CREATE TABLE products(
productID INT Identity(1,2) PRIMARY KEY not null,
productName VARCHAR(50) NOT NULL,
productType VARCHAR(50) not null,
productDescription VARCHAR(255),
costPrice DECIMAL(10,2) check(costPrice>=0),
sellingPrice DECIMAL(10,2) check(sellingPrice>=0)
```

```
);
CREATE TABLE stock(
stockID INT Identity(100,1) PRIMARY KEY NOT NULL,
stockQuantity INT NOT NULL check(stockQuantity>=0),
minStock INT not null check(minStock>=0),
expireDate DATE not null,
productID int references products(productID)
);
CREATE TABLE orders(
orderID INT Identity(1000,5) PRIMARY KEY NOT NULL,
orderQuantity INT NOT NULL check(orderQuantity>=0),
orderDate DATE not null,
orderStatus varchar(20),
supplierID int references supplier(supplierID),
productID int references products(productID)
);
CREATE TABLE transactions(
transactionID INT Identity(1000,10) PRIMARY KEY NOT NULL,
transactionType VARCHAR(20) not null,
transactionDate DATE not null,
quantity INT NOT NULl check(quantity>=0),
userID int references users(userID),
productID int references products(productID)
);
CREATE TABLE report(
reportID INT Identity(10,10) PRIMARY KEY NOT NULL,
reportType VARCHAR(20),
reportContent varchar(200),
generateDate DATE,
userID int references users(userID)
);
INSERT INTO Role (roleName)
VALUES
('Owner,Sales,StockManager'),
('Admin,updateAccount,manageProdut'),
('Accountant,view');
INSERT INTO users (userName, userPassword, roleID)
VALUES
('KidmuNasr', 'Owner@123',3),
('BereketTeketel', 'admin@124',5),
('NahomDemsse', 'Acc@125',7);
INSERT INTO supplier (supplierF_Name, supplierL_Name, supplierPhone)
VALUES
('Abebe', 'Kebede', '+251911111111'),
('Almaz', 'Aschenaki', '+251911001111');
INSERT INTO products
(productName, productType, productDescription, costPrice, sellingPrice)
VALUES
('Oreo Biscuits', 'Bakery', 'Sweet cream-filled biscuits', 120, 160),
('Always Pads', 'Hygiene', 'Sanitary pad pack', 185, 220),
('Lifebuoy Soap', 'Hygiene', 'Antibacterial bath soap', 30, 50),
('KitKat Chocolate', 'Confectionery', 'Chocolate wafer pack', 185, 220),
('Closeup Toothpaste', 'Household', 'Dental care toothpaste', 170, 210);
INSERT INTO stock (stockQuantity, minStock, expireDate, productID)
VALUES
(200, 50, '2026-03-20', 1),
(120, 30, '2027-01-10', 3),
(90,  20, '2027-06-15', 5),
(150, 40, '2026-05-15', 7),
(50, 24,  '2027-06-15', 9);
INSERT INTO orders
(orderQuantity, orderDate, orderStatus, supplierID, productID)
VALUES
(300, '2025-01-10', 'Pending',   1, 1),
(200, '2025-01-12', 'Delivered', 2, 3),
(150, '2025-01-15', 'Pending',   1, 5),
(170, '2025-01-18', 'Cancelled', 2, 7);
INSERT INTO transactions
(transactionType, transactionDate, quantity, userID, productID)
VALUES
('IN',  '2025-01-10', 300, 2, 1),
('OUT', '2025-01-11', 40,  2, 1),
('IN',  '2025-01-12', 200, 2, 3),
('OUT', '2025-01-13', 60,  2, 3),
('OUT', '2025-01-22', 25,  2, 7),
('IN',  '2025-01-12', 150, 2, 5),
('OUT', '2025-01-16', 35,  2, 5);
INSERT INTO report
(reportType, reportContent, generateDate, userID)
VALUES
( 'Stock Report','Current stock levels for all products including Oreo, KitKat, Lifebuoy, Always, and Closeup.','2025-01-20',2),
( 'Low Stock', 'Products below minimum stock level: Lifebuoy soap and Closeup toothpaste.' '2025-01-21',2),
( 'Sales Report','Daily sales summary based on OUT transactions recorded by the system.',
  '2025-01-22',2),
('Expiry Report', 'List of products approaching expiry date within the next three months.',
  '2025-01-23', 2);
```

```
alter table products alter column productType varchar (90) not null;
update supplier set supplierF_Name = 'Bekalu', supplierL_Name = 'Ashenafi' where supplierPhone = '+251911111111' and supplierF_Name = 'Abebe';
select productID from products where productName = 'Always Pads';
delete from transactions  where productID = '2';
select * from products where productName LIKE 'k%';
Select expireDate, stockQuantity from stock where minStock between 20 and 30;
Select expireDate, minStock from stock where stockQuantity IN (50,200,90);
SELECT orderDate from orders where orderStatus is null;
select * from transactions where (productID='4' or userID='4') or (quantity in(150,200,40) and transactionType like 'I%') and transactionDate != '2025-01-12';
select minStock from stock where exists
(select minStock from stock where stockQuantity > 90);
select * from stock where stockQuantity > all
(select stockQuantity from stock where stockQuantity <=90);
select * from transactions where quantity >any
(select quantity from transactions where productID > 3);
select count (*) from orders where orderStatus = 'pending';
select sum (orderQuantity) from orders;
select avg (orderQuantity) from orders;
select max (orderQuantity) from orders;
select min (orderQuantity) from orders;
update products set sellingPrice = 199
where sellingPrice < (select Avg (sellingPrice) from products)
select p.productID, p.productName, p.productType, p.productDescription, costPrice, sellingPrice
from products p join orders o on p.productID = o.orderID;
SELECT * from products p LEFT OUTER JOIN orders o ON p.productID = o.productID;
SELECT * from products p RIGHT OUTER JOIN orders o ON p.productID = o.productID;
SELECT * FROM products p FULL OUTER JOIN orders o ON p.productID = o.productID;
```

```
select * from supplier order by supplierF_Name;
select * from products order by sellingPrice;
select costPrice, sum (sellingPrice) from products group by costPrice;
SELECT orderStatus, SUM(orderQuantity) AS TotalQuantity FROM orders GROUP BY orderStatus;
SELECT orderStatus, SUM(orderQuantity) AS TotalQuantity FROM orders GROUP BY orderStatus HAVING SUM(orderQuantity) > 150;
select * from report
select * from transactions
select * from orders
select * from stock
select * from products
select * from supplier
select * from users
```

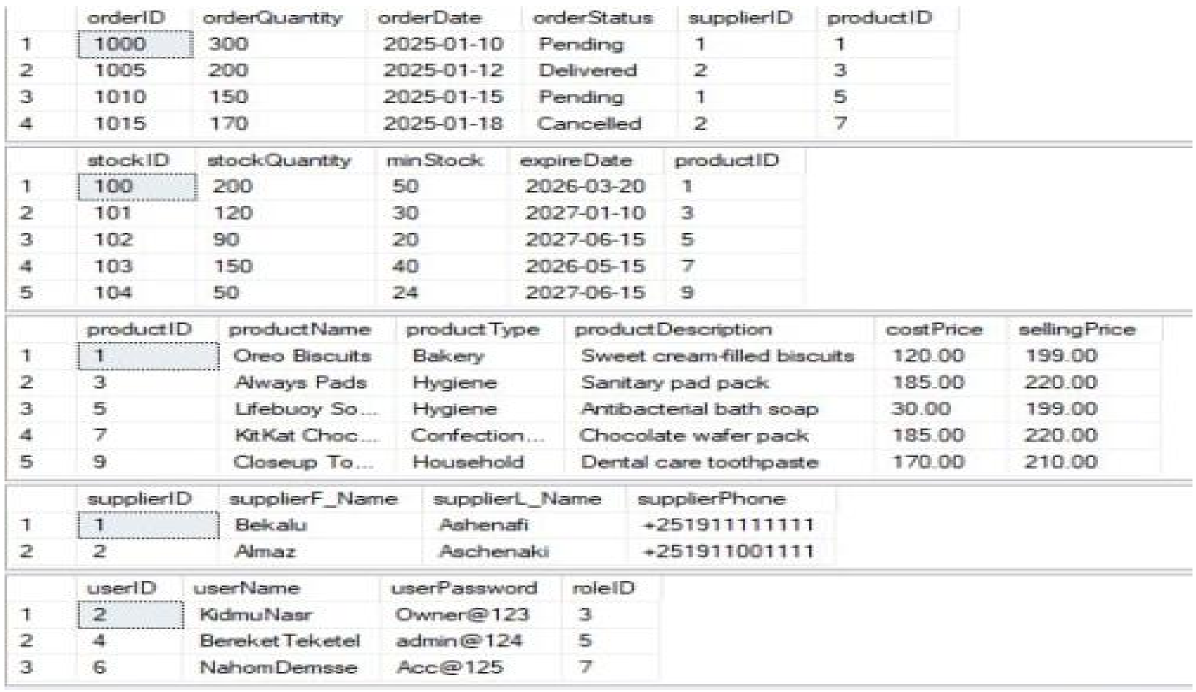


	reportID	reportType	reportContent	generateDate	userID
1	10	Stock Report	Current stock levels for all products including ...	2025-01-20	2
2	20	Low Stock	Products below minimum stock level: Lifebuoy...	2025-01-21	2
3	30	Sales Report	Daily sales summary based on OUT transactio...	2025-01-22	2
4	40	Expiry Report	List of products approaching expiry date withi...	2025-01-23	2

	transactionID	transactionType	transactionDate	quantity	userID	productID
1	1000	IN	2025-01-10	300	2	1
2	1010	OUT	2025-01-11	40	2	1
3	1020	IN	2025-01-12	200	2	3
4	1030	OUT	2025-01-13	60	2	3
5	1040	OUT	2025-01-22	25	2	7
6	1050	IN	2025-01-12	150	2	5
7	1060	OUT	2025-01-16	35	2	5

	orderID	orderQuantity	orderDate	orderStatus	supplierID	productID
1	1000	300	2025-01-10	Pending	1	1
2	1005	200	2025-01-12	Delivered	2	3
3	1010	150	2025-01-15	Pending	1	5
4	1015	170	2025-01-18	Cancelled	2	7

	stockID	stockQuantity	minStock	expireDate	productID
1	100	200	50	2026-03-20	1
2	101	120	30	2027-01-10	3
3	102	90	20	2027-06-15	5
4	103	150	40	2026-05-15	7
5	104	50	24	2027-06-15	9



	orderID	orderQuantity	orderDate	orderStatus	supplierID	productID
1	1000	300	2025-01-10	Pending	1	1
2	1005	200	2025-01-12	Delivered	2	3
3	1010	150	2025-01-15	Pending	1	5
4	1015	170	2025-01-18	Cancelled	2	7

	stockID	stockQuantity	minStock	expireDate	productID
1	100	200	50	2026-03-20	1
2	101	120	30	2027-01-10	3
3	102	90	20	2027-06-15	5
4	103	150	40	2026-05-15	7
5	104	50	24	2027-06-15	9

	productID	productName	productType	productDescription	costPrice	sellingPrice
1	1	Oreo Biscuits	Bakery	Sweet cream-filled biscuits	120.00	199.00
2	3	Always Pads	Hygiene	Sanitary pad pack	185.00	220.00
3	5	Lifebuoy So...	Hygiene	Antibacterial bath soap	30.00	199.00
4	7	KitKat Choc...	Confection...	Chocolate wafer pack	185.00	220.00
5	9	Closeup To...	Household	Dental care toothpaste	170.00	210.00

	supplierID	supplierF_Name	supplierL_Name	supplierPhone
1	1	Bekalu	Ashenafi	+251911111111
2	2	Almaz	Aschenaki	+251911001111

	userID	userName	userPassword	roleID
1	2	KidmuNasr	Owner@123	3
2	4	BereketTeketel	admin@124	5
3	6	NahomDemsse	Acc@125	7