**Vietnam General Confederation of Labor**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL PROJECT SOFTWARE ENGINEERING**

**MOBILE DISTRIBUTION MANAGEMENT SYSTEM**

*Instructor*: **Mr. PHAM THAI KY TRUNG**

*Student*: **Nguyen Hoang Duy – 521H0350**

*Class* **: 21H50202**

*Year* **: 21**

**HO CHI MINH CITY, 2023**

Vietnam General Confederation of Labor

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL PROJECT SOFTWARE ENGINEERING**

**MOBILE DISTRIBUTION MANAGEMENT SYSTEM**

*Instructor*: **Mr. PHAM THAI KY TRUNG**

*Student*: **Nguyen Hoang Duy – 521H0350**

*Class* **: 21H50202**

*Year* **: 21**

**HO CHI MINH CITY, 2023**

ACKNOWLEDGEMENT

The first sincere thanks I want to give to Mr. Pham Thai Ky Trung, who enthusiastically taught and worked tirelessly to give me enough tools and skills to complete this report. He played an important role in improving my mathematical logic and knowledge. The second thanks I would like to give to the teachers of the Department of Information Technology of Ton Duc Thang University for giving me the opportunity to do this report.

Because the impact of the epidemic is too great, my report will have some errors, I am very open to receiving feedback from teachers so that I can improve my report writing skills.

Finally, I wish you good health and success in your noble career.

**THIS PROJECT WAS COMPLETED AT**

**TON DUC THANG UNIVERSIY**

I fully declare that this is my own project and is guided by Mr. Pham Thai Ky Trung; The research contents and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

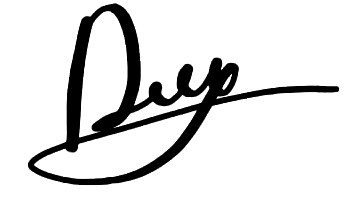
Besides that, the project also uses a number of comments, assessments as well as data from other authors, other agencies and organizations, with citations and source annotations.

**Should any frauds were found, I will take full responsibility for the content of my report.** Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

*Ho Chi Minh city, 6th May, 2023*

*Author*

*(Sign and write full name)*

**

*Nguyen Hoang Duy*

CONFIRMATION AND ASSESSMENT SECTION

**Instructor confirmation section**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh January, 2023*

*(Sign and write full name)*

**Evaluation section for grading instructor**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh January 2023*

*(Sign and write full name)*

SUMMARY

This is a report for the final project of Software Engineering subject. This report mainly about the process of creating, developing, managing and maintaining the Mobile Distribution Management Software.

INDEX

[ACKNOWLEDGEMENT i](#_Toc134304391)

[CONFIRMATION AND ASSESSMENT SECTION iii](#_Toc134304392)

[SUMMARY iv](#_Toc134304393)

[INDEX 1](#_Toc134304394)

[CHAPTER 1 – INTRODUCTION 3](#_Toc134304395)

[1.1 Purpose and scope 3](#_Toc134304396)

[1.1.1 Purpose 3](#_Toc134304397)

[1.1.2 Scope 3](#_Toc134304398)

[1.2 Product Overview 3](#_Toc134304399)

[1.3 Structure of the Document 3](#_Toc134304400)

[1.4 Terms, Acronyms, and Abbreviations 4](#_Toc134304401)

[CHAPTER 2 – PROJECT MANAGEMENT PLAN 5](#_Toc134304402)

[2.1 Project Organization 5](#_Toc134304403)

[2.2 Lifecycle model used 5](#_Toc134304404)

[2.3 Risk analysis 6](#_Toc134304405)

[2.4 Hardware and Software Resource Requirements 6](#_Toc134304406)

[2.5 Deliverables and Schedule 7](#_Toc134304407)

[2.6 Professional Standards 7](#_Toc134304408)

[2.7 Impact of the project on individuals and organizations 8](#_Toc134304409)

[CHAPTER 3 – REQUIREMENT SPECIFICATIONS 9](#_Toc134304410)

[3.1 Stakeholders for the system 9](#_Toc134304411)

[3.2 Use case list 9](#_Toc134304412)

[3.2.1 Graphical use case model 10](#_Toc134304413)

[3.2.2 Textual Description 13](#_Toc134304414)

[3.3 Functional requirements 22](#_Toc134304415)

[3.4 Non-functional requirements 23](#_Toc134304416)

[CHAPTER 4 – ARCHITECTURE 23](#_Toc134304417)

[4.1 Architectural styles used 24](#_Toc134304418)

[4.2 Architectural model 24](#_Toc134304419)

[4.3 Technology, software, and hardware used 25](#_Toc134304420)

[4.4 Rationale for your architectural style and model 25](#_Toc134304421)

[CHAPTER 5 – DESIGN 25](#_Toc134304422)

[5.1 GUI design 25](#_Toc134304423)

[5.2 Static model – class diagrams 26](#_Toc134304424)

[5.3 Dynamic model – sequence diagrams 27](#_Toc134304425)

[CHAPTER 6 – TEST PLAN 31](#_Toc134304426)

[6.1 Requirements/specifications-based system level test cases 32](#_Toc134304427)

[6.2 Traceability of test cases to use cases 32](#_Toc134304428)

[6.3 Techniques used for test generation 32](#_Toc134304429)

[6.4 Assessment of the goodness of your test suite 32](#_Toc134304430)

[CHAPTER 7 – DEMO 32](#_Toc134304431)

[7.1 Database 32](#_Toc134304432)

[7.2 Software output 32](#_Toc134304433)

[7.3 Github activites 32](#_Toc134304434)

[REFERENCES 32](#_Toc134304435)

CHAPTER 1 – INTRODUCTION

1.1 Purpose and scope

1.1.1 Purpose

This document provides a comprehensive overview of the architecture of the mobile distribution management software we designed, including the use of technology models. For the purpose of capturing and communicating the important architectural decisions that have been made about the modules, the document serves as a bridge between the software requirements and the detailed design of the application. It also helps programmers easily grasp the requirements of users and customers, avoiding unnecessary omissions.

1.1.2 Scope

This document defining software architecture that addresses project requirements in the following areas: functionality, usability, reliability, scalability, maintainability, manageability, and evolvability of our software.

1.2 Product Overview

The distributor requires software that can effectively manage inventory and orders. Resellers should be able to place orders, select payment methods keep track of their order status. Accountants should be able to to record goods received, create delivery notes, update order and payment statuses. The system should be able to maintain inventory levels, and generate monthly stock, sales, and revenue reports.

1.3 Structure of the Document

The structure of the document consists of seven parts including: introduction, project management plan, requirements specifications, architecture, design, test plan and demo. Each part plays an important role in creating, developing and managing the process of making this application

1.4 Terms, Acronyms, and Abbreviations

|  |  |  |
| --- | --- | --- |
| **#** | **Item** | **Description** |
| 1 | MVC | Model – View – Control architecture |
| 2 | 3LA | Three-layer architecture |
| 3 | BUS | Business layer |
| 4 | DAL | Data Access Layer |
| 5 | GUI | Graphical User Interface |
| 6 | UI | User Interface |
| 7 | MDMS | Mobile Distribution Management System |

CHAPTER 2 – PROJECT MANAGEMENT PLAN

2.1 Project Organization

The development team for this MDMS is organized in a way that maximizes the efficiency and productivity. This team consists of developers, designers, quality assurance personnel and project managers. The developers are responsible for coding and make sure that the software runs properly. The designers are responsible for creating user-friendly UI and ensure it run smoothly. The program is tested by the quality assurance team to make sure there are no bugs or faults in it. The project managers are in charge of supervising the whole project, directing the team, and making sure it achieves its aims and objectives.

The rationale of this structure is to make sure that all team members are clear about their role and duty. This will help to avoid duplicate effort, reduces errors, guarantees the project complete on time and fulfills client’s satisfaction.

2.2 Lifecycle model used

The lifecycle model used for the whole project is the Waterfall model. The development process consists of requirements analysis, design, implementation, testing and maintenance. The Waterfall model offers a disciplined method for developing software that makes sure that each stage is finished completely before going on the next stage.

The rationale of using this model is it provided a structured and disciplined approach to software development. Also, based on the scenario that we just have a small number of members and the amount of workload, we think that this will be the best suit model for a whole project.

* 1. Risk analysis
* Technical risk: the system may not work as expected due to technical issues such as bugs, compatibility problems or software conflicts.
* Risk arise possibility: moderate.
* Risk reduction strategy: the developers have to conduct the test throughout the development process.
* Timeline risk: the project may take longer time than expected to fully complete.
* Risk arise possibility: high.
* Risk reduction strategy: create a detail project plan with clear deadlines.
* Communication risk: team members communication may break down which result in misunderstanding, delays, …
* Risk arise possibility: high
* Risk reduction strategy: establish regular communication between team members. Use the same communication tools like discord, messenger.
* Resource risk: team members may experience resource constraint like shortage of members.
* Risk arise possibility: moderate to high
* Risk reduction strategy: provide team members with all necessary skills and knowledge. Adjust the timelines if any unexpected resource constraint occurs.
  1. Hardware and Software Resource Requirements
* Hardware Requirement:
* Computer workstation: minimum 8GB RAM, 16GB hard drive space.
* Sever: minimum 16GB RAM, 256GB hard drive space.
* Network equipment: routers, switches.
* Printers and scanners.
* Software requirements:
* Operating system: Windows 10.
* Database Management System: SQL Server.
* Version control software: Git/Github.
  1. Deliverables and Schedule
* Requirement gathering: gather requirement from the requests and others related software and document them in a requirements document. Time: 3 days.
* Design: design system architecture, create database, design GUI. Time: 1 week.
* Development: implement the software based on the design and requirements. Time: 2 weeks.
* Testing: test entire system, fix any arise bugs. Time: 2 days.
* Deployment: deploy the software on others devices (parallel with testing). Time: 2 days.
  1. Professional Standards
* Scholastic Honesty: team members are expected to maintain high levels of academic integrity and avoid plagiarism or cheating.
* Meeting schedule: team members are expected to join all scheduled meetings, actively in discussion.
* Quality Expectations: team members are expected to complete all the tasks meet the required quality standards.
  1. Impact of the project on individuals and organizations

For the distributor, the software will provide a more simplified and effective way to manage inventories and orders. This will reduce the time and effort required for manual tracking and management of goods received, delivery notes, and payments.

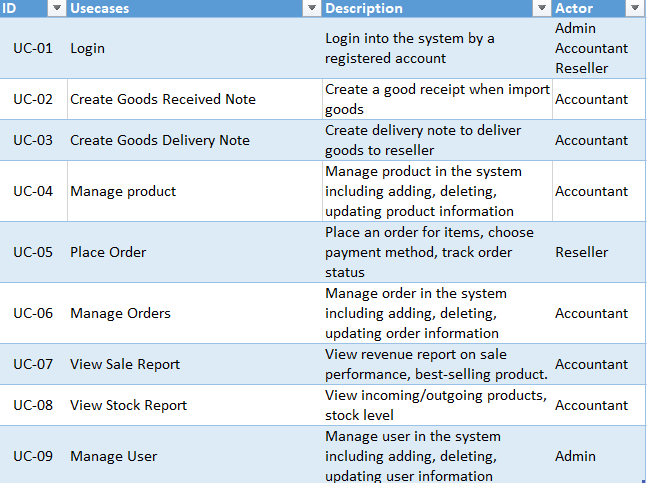
For the resellers, the software will provide a more user-friendly interface for placing orders, making payments, and tracking the status of orders. This will reduce the time and effort required for manual processing of orders, leading to increased efficiency and accuracy.

CHAPTER 3 – REQUIREMENT SPECIFICATIONS

3.1 Stakeholders for the system

* Admin: manage account, authorized account.
* Accountant: manage orders, manage inventory, receive goods, deliver products, view sale/stock report.
* Reseller: place orders, make payments and track their order status.

3.2 Use case list



3.2.1 Graphical use case model

* Login module:

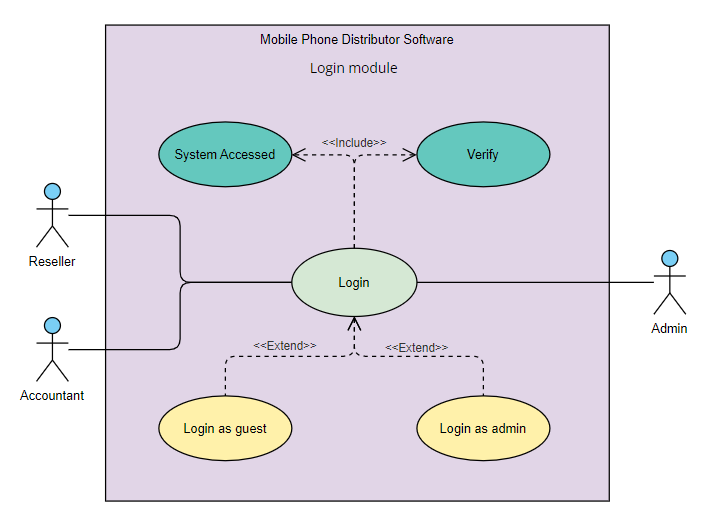


Figure 3.2.1.1 Use case diagram for Login module

* Admin module:

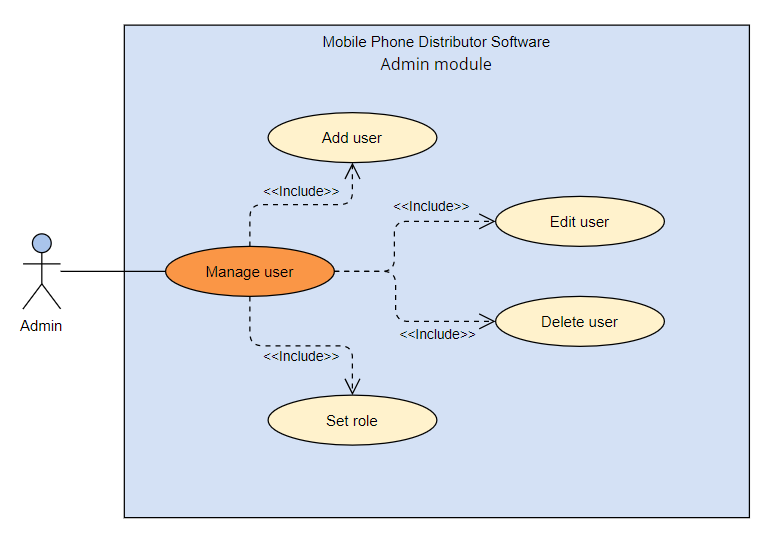


Figure 3.2.1.2 Use case diagram for Admin module

* Accountant module:

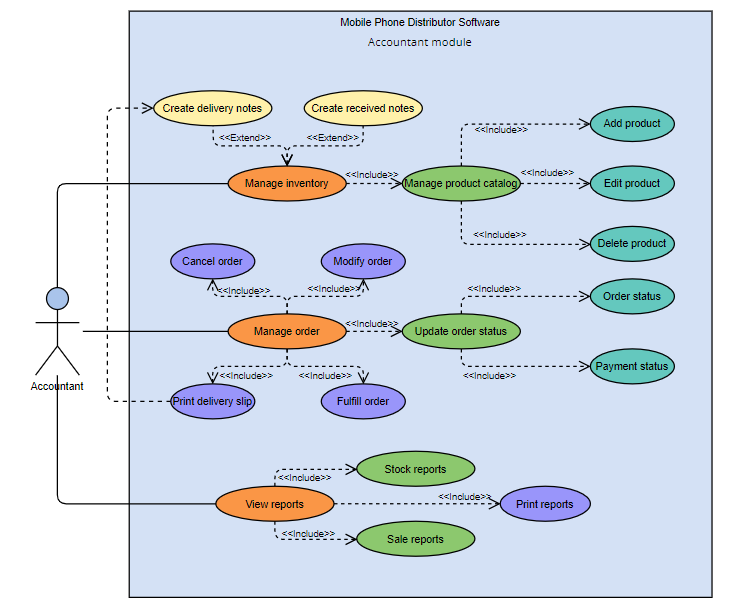


Figure 3.2.1.3 Use case diagram for Accountant module

* Reseller module:

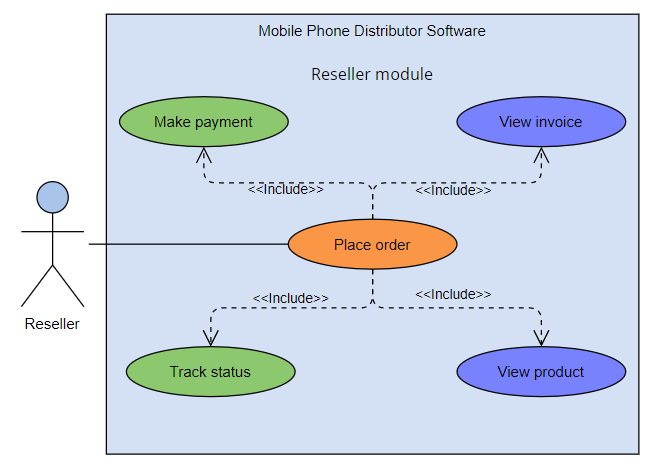


Figure 3.2.1.4 Use case diagram for Reseller module

3.2.2 Textual Description

* Description for use case Login:

|  |  |  |
| --- | --- | --- |
| **Use case name:** | Login | |
| **Trigger event:** | Application start | |
| **Brief description:** | Admin, accountant and reseller logged into the system by a registered account. | |
| **Actors:** | Admin, accountant and reseller. | |
| **Preconditions:** | - User must have a valid username and password.  - Reseller must have access to internet. | |
| **Post conditions:** | User is logged into the system, access to system features based on their role. | |
| **Flow of activities:** | Actor | System |
| 1. User start the application.  2. User enter their valid username and password and click Login button. | 1.1 MDMS display login form.  2.1 System verify user’s credential and logs them into the system based on their role.  2.2 Use case end. |
| **Alternative flow:** | 2.1 If user enter incorrect username or password, the system will display an error message. | |
| **Exception conditions:** | 2.1 System raise error if the connection to database is broken down. | |

* Description for use case Create Goods Received Note:

|  |  |  |
| --- | --- | --- |
| **Use case name:** | Create Goods Received Note | |
| **Trigger event:** | Import of goods | |
| **Brief description:** | Accountant create a goods receipt when they are imported and update inventory levels. | |
| **Actors:** | Accountant | |
| **Preconditions:** | - Goods are imported  - Accountant is logged into the system | |
| **Post conditions:** | - A goods received note is created and stored in the system.  - The inventory levels are updated. | |
| **Flow of activities:** | Actor | System |
| 1. User logs in to the system and navigate to Create Receipts section.  2. User select option, enter details such as name, quantity and to create a goods received note | * 1. MDMS display a receipt form.   2.1 System generates a goods received note, stores it in the system and updates stock levels.  2.2 Use case end. |
| **Alternative flow:** | 2.1 If user enter incomplete details, the error will be displayed. | |
| **Exception conditions:** | 2.1 System raise error if it failed to update the inventory levels. | |

* Description for use case Create Goods Delivery Note:

|  |  |  |
| --- | --- | --- |
| **Use case name:** | Create Goods Delivery Note | |
| **Trigger event:** | Good are ready to be delivered to reseller. | |
| **Brief description:** | Accountant create a goods delivery note to record the details of the goods being delivered to reseller. | |
| **Actors:** | Accountant | |
| **Preconditions:** | - The goods are ready to be delivered  - Reseller has placed an order for the goods  - Accountant has access to the system | |
| **Post conditions:** | - A delivery note is created in the system | |
| **Flow of activities:** | Actor | System |
| 1. User logs in to the system and navigate to create delivery note section.  2. User selects option, enters missing details and verifies the order details and reseller details. | 1.1MDMS display a delivery form with related order details and reseller details.  2.1 System generates a delivery note and stores it in the system.  2.2 Use case end. |
| **Alternative flow:** | 2.1 If reseller has not placed an order, the accountant will not be able to create a delivery note. | |
| **Exception conditions:** | 2.1 System raise error if it failed to create a delivery note. | |

* Description for use case Manage Product:

|  |  |  |
| --- | --- | --- |
| **Use case name:** | Manage Product | |
| **Trigger event:** | Accountant need to manage product information | |
| **Brief description:** | Accountant manage product includes adding, deleting and updaing product information. | |
| **Actors:** | Accountant | |
| **Preconditions:** | - Accountant has access to the system | |
| **Post conditions:** | - Product infomation is successfully added, deleted or updated in the system. | |
| **Flow of activities:** | Actor | System |
| 1. User logs in to the system and navigate to Products section.  2. If user selects to add product.  3. If user selects to delete product.  4. If user selects to edit product.  5. User save changes | 1.1MDMS display a list of product with options to add, delete and edit.  2.1 System display a form to add product information.  3.1 System display a message box to confirm the deletion.  4.1 System display a form to modify product information.  5.1 System update product infomation  5.2 Use case end. |
| **Alternative flow:** |  | |
| **Exception conditions:** | If their is an error in the system, the system displays error message. | |

* Description for use case Place Order:

|  |  |  |
| --- | --- | --- |
| **Use case name:** | Place Order | |
| **Trigger event:** | Reseller need to place an order for items. | |
| **Brief description:** | Reseller place an order for items, choose payment method and track order status. | |
| **Actors:** | Reseller | |
| **Preconditions:** | - Reseller is logged in to the system  - Reseller has selected the items to order  - Reseller has chosen the payment method. | |
| **Post conditions:** | - Order is placed in the system  - Reseller can track the order status | |
| **Flow of activities:** | Actor | System |
| 1. User logs in to the system  2. User select items, payment method and delivery method.  3. User comfirm the order | * 1. MDMS display an order form.   2.1 System verify the availability of item.  2.2 System calculate the total cost  3.1 System generate an order and display the status of order  3.2 Use case end. |
| **Alternative flow:** | 2.1 If item is not available, the system blocks reseller from selected that item. | |
| **Exception conditions:** | If their is an error while placing order, the system displays error message. | |

* Description for use case Manage Order:

|  |  |  |
| --- | --- | --- |
| **Use case name:** | Manage Order | |
| **Trigger event:** | Accountant want to manage the order in the system | |
| **Brief description:** | Accountant manage orders by adding, deleting and updating order information | |
| **Actors:** | Accountant | |
| **Preconditions:** | - Accountant is logged in to the system  - There are existing order in the system. | |
| **Post conditions:** | - The order information is updated in the system | |
| **Flow of activities:** | Actor | System |
| 1. User logs in to the system and navigate to the order list section.  2. If user selects add an order.  3. If user selects delete and order.  4. If user selects edit an order.  5. User save changes | * 1. MDMS display an order list with add, delete and edit options.   2.1 System display a form to add an order.  3.1 System display a message box to confirm deletion.  4.1 System display a form to edit an order.  5.1 System update order infomation.  5.2 Use case end. |
| **Alternative flow:** | If accountant cancel the action at any step, the system returns to the order list form. | |
| **Exception conditions:** | If their is an error in the system, the system displays error message. | |

* Description for use case View Sale Report:

|  |  |  |
| --- | --- | --- |
| **Use case name:** | View Sale Report | |
| **Trigger event:** | Accountant selects the option to view the sale report | |
| **Brief description:** | The system display a report containing revenue information and sale performance of the products. | |
| **Actors:** | Accountant | |
| **Preconditions:** | - Accountant is logged in to the system  - There are existed sale data in the system. | |
| **Post conditions:** | - The system displays the sale report | |
| **Flow of activities:** | Actor | System |
| 1. User logs in to the system and navigate to the sale report section. | * 1. MDMS calculate the data from the database and display a sale report.   1.2 Use case end. |
| **Alternative flow:** | If there are no sale data in the system, the system display a message to inform user. | |
| **Exception conditions:** | If their is an error in the system, the system displays error message. | |

* Description for use case View Stock Report:

|  |  |  |
| --- | --- | --- |
| **Use case name:** | View Stock Report | |
| **Trigger event:** | Accountant selects the option to view the stock report | |
| **Brief description:** | The system display a report containing incoming/outgoing products and current stock levers in the system. | |
| **Actors:** | Accountant | |
| **Preconditions:** | - Accountant is logged in to the system  - There are existed stock data in the system. | |
| **Post conditions:** | - The system displays the stock report | |
| **Flow of activities:** | Actor | System |
| 1. User logs in to the system and navigate to the stock report section. | * 1. MDMS calculate the data from the database and display a stock report.   1.2 Use case end. |
| **Alternative flow:** | If there are no stock data in the system, the system display a message to inform user. | |
| **Exception conditions:** | If their is an error in the system, the system displays error message. | |

* Description for use case Manage User:

|  |  |  |
| --- | --- | --- |
| **Use case name:** | Manage User | |
| **Trigger event:** | Admin wants to mange user infomation | |
| **Brief description:** | Admin manage user information by adding, deleting and updating user information. | |
| **Actors:** | Admin | |
| **Preconditions:** | - Admin is logged in to the system | |
| **Post conditions:** | - User information is updated in the system | |
| **Flow of activities:** | Actor | System |
| 1. User logs in to the system and navigate to the admin section.  2. If user selects add an order.  3. If user selects delete and order.  4. If user selects edit an order.  5. User save changes | * 1. MDMS display a list of users with add, delete and edit options.   2.1 System display a form to add an user.  3.1 System display a message box to confirm deletion.  4.1 System display a form to edit an user.  5.1 System update user infomation.  5.2 Use case end. |
| **Alternative flow:** | If admin enter invalid user information, the system displays error message | |
| **Exception conditions:** | If their is an error in the system, the system displays error message. | |

3.3 Functional requirements

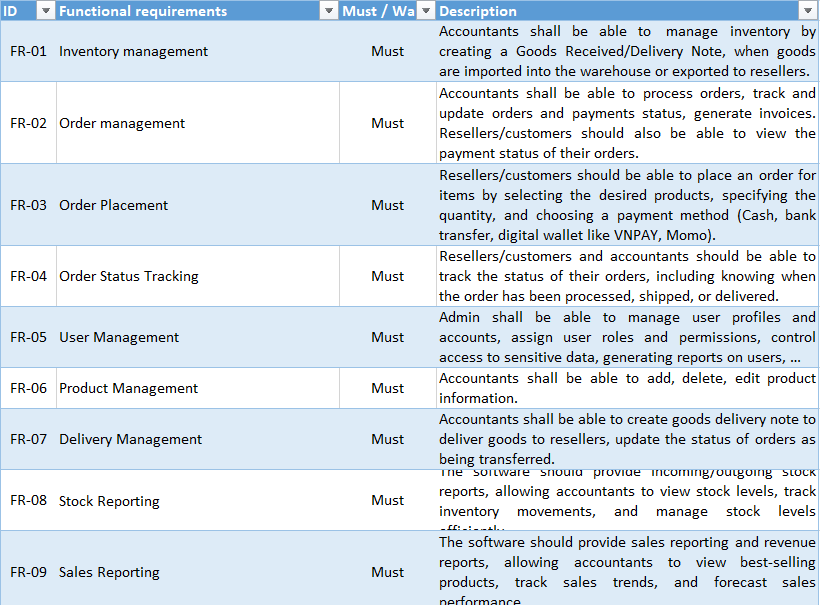


Figure 3.3.1 Functional requirements

3.4 Non-functional requirements

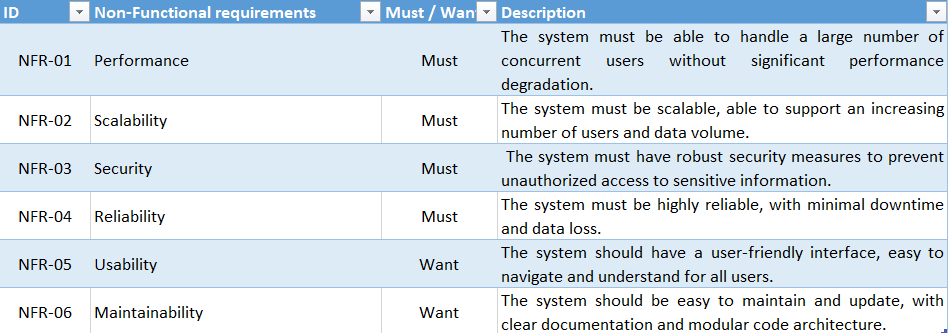


Figure 3.3.2 Non-functional requirements

CHAPTER 4 – ARCHITECTURE

4.1 Architectural styles used

- The MDMS is designed using a client-server architecture. The system consists of two main components: the client-side application used by resellers to place orders and server-side is used by accountant to manage the distribution process.

- Client-side application using MVC architecture includes model, view and model which allow resellers to place orders, make payment and track order status online.

- Server-side application using three-layer architecture includes presentation layer, business layer and data access layer which provide accountants and admin with a solution to manage inventory, orders and accounts.

4.2 Architectural model

- MVC (model – view- control) architecture: this architecture pattern separate the application into three distinct parts: Model, View and Control.

* Model represents the data, logic and rules of the application.
* View represents a complete user interface such as information, button and receive user input.
* Control accepts input from View, handle it and update View and Model if needed.

- Three-layer architecture: known as n-tier architecture. Three-layer pattern divided application into three main components: Presentation Layer, Business Logic Layer and Data Access Layer.

* Presentation Layer: handle user interaction and display data to user.
* Business Layer: contain all the application’s logic.
* Data Access Layer: handle the data storage. Carry out operations on database.

4.3 Technology, software, and hardware used

- Programming language:

* Winform application: C#.
* Webform application: HTML, CSS, Javascript, C#.

- Framework:

* Winform application: .NET, third-party libraries for additional functionalities.
* Webform application: ASP.NET MVC, Entity Framework.

- Database management system: SQL Server.

- Software requirements:

* Integrated Development Environment (IDE): Visual Studio 2019, Microsoft SQL Server Management Studio 2018 (SSMS).

4.4 Rationale for your architectural style and model

MVC and three-layer models were chosen because they provide a clear separation of the software and allow easy maintenance and scalability. They also provide the flexibility in the design of the application.

CHAPTER 5 – DESIGN

5.1 GUI design

- Login form:

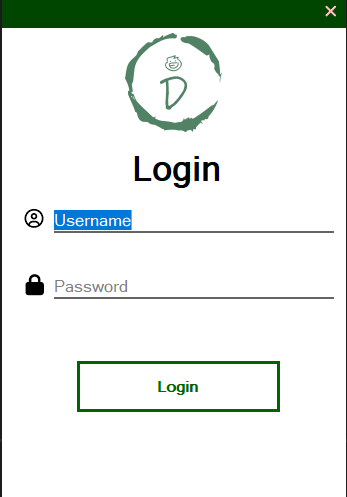


Figure 5.1.1 Login form

- Home/Dashboard form: we can click into the name to show the signout button.

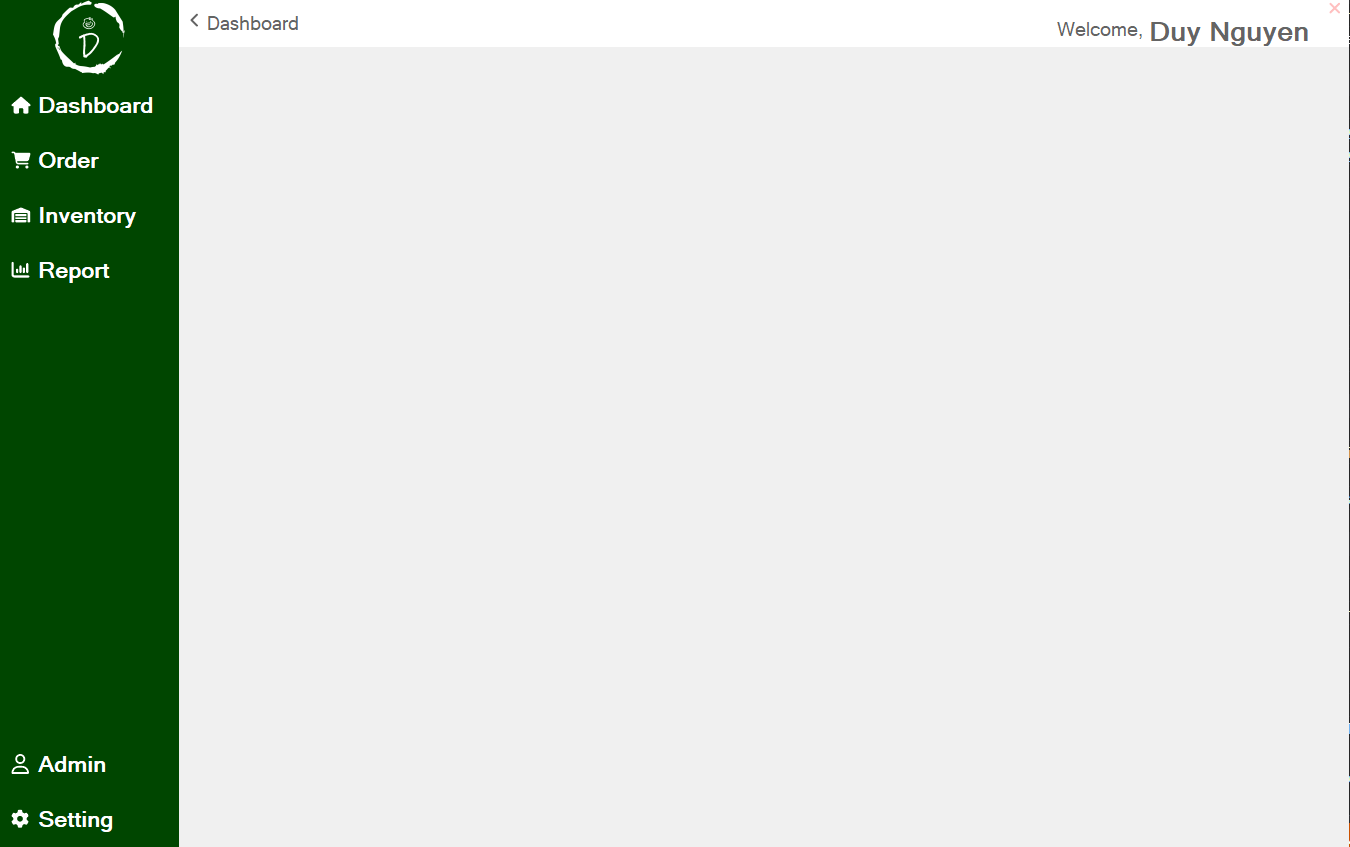


Figure 5.1.2 Home/Dashboard form

- Order List form: we can search orders by reseller name and switch to pending tab to view the pending orders. Double click into specific order to view that order details.

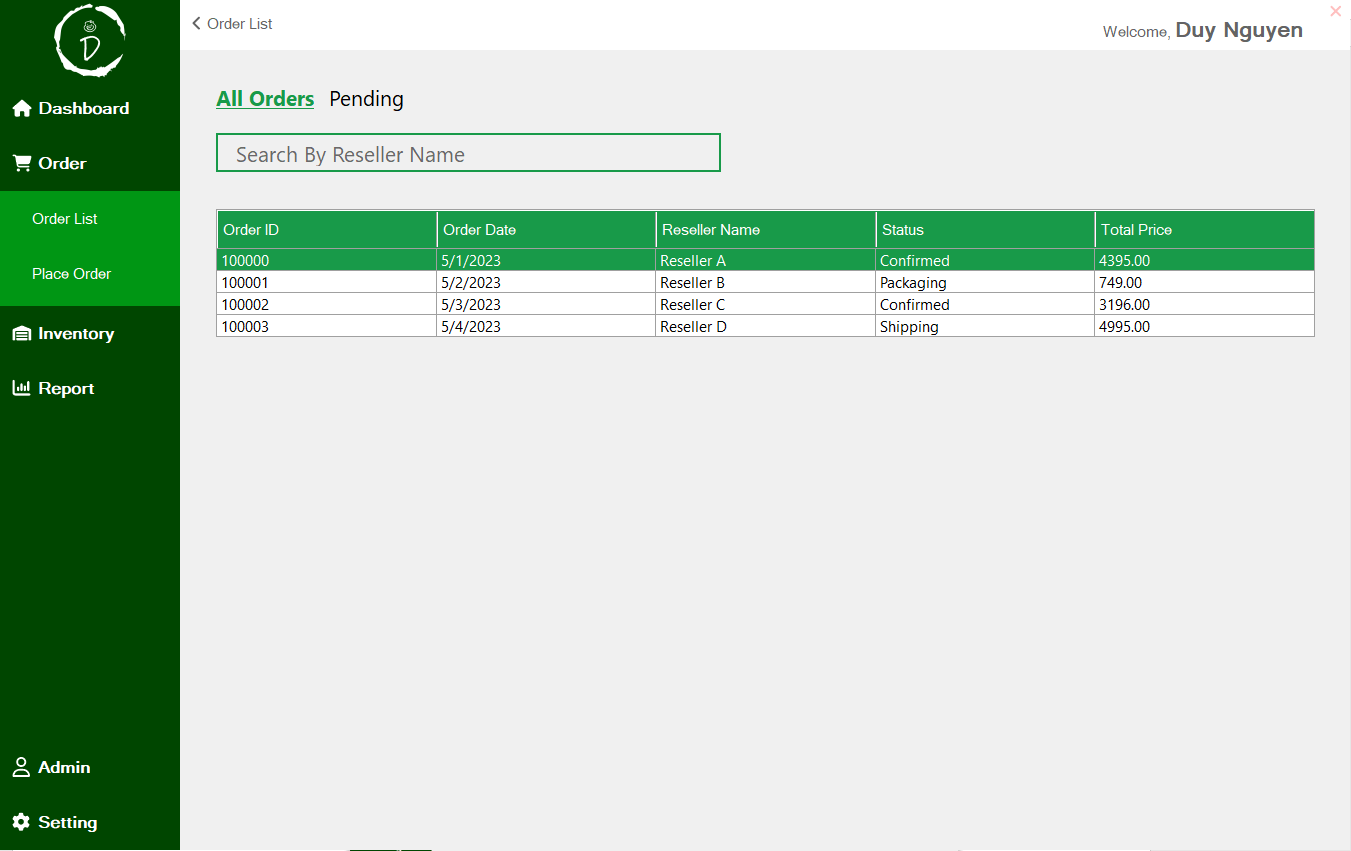


Figure 5.1.3 Order List form

- Order details form:

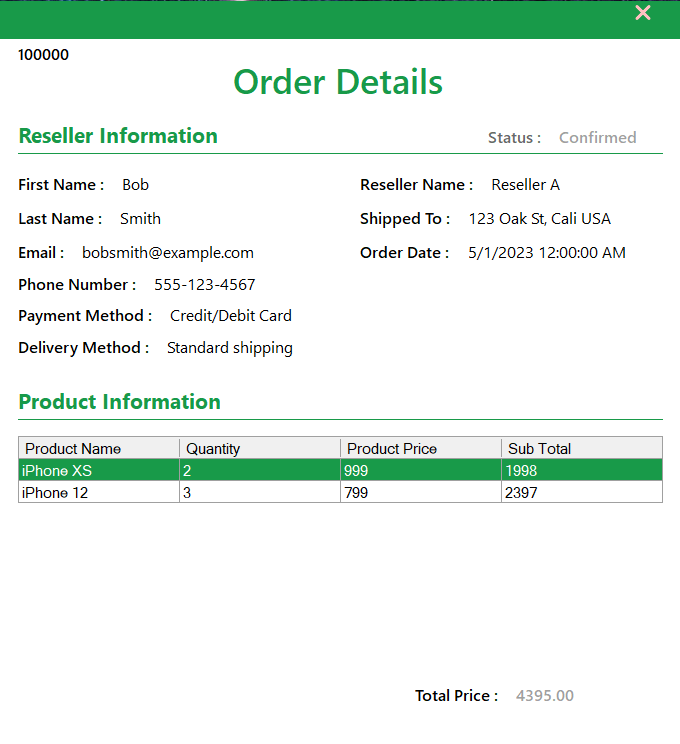


Figure 5.1.4 Order details form

- Product List form: we can search product by product name and perform create, delete, edit product actions with functional buttons.

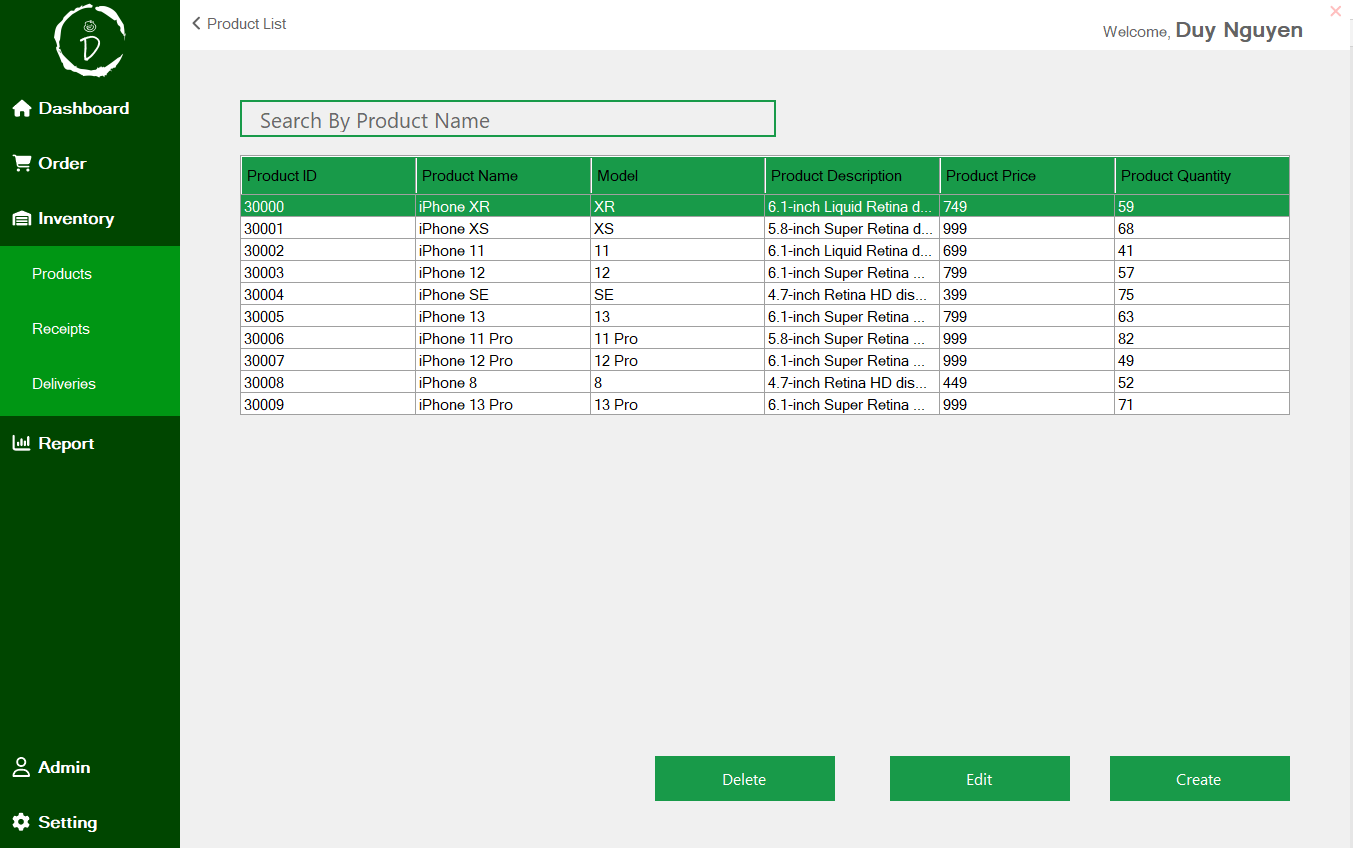


Figure 5.1.5 Product List form

- Product details form: perform either create and edit function. For edit, you need to select a product first.

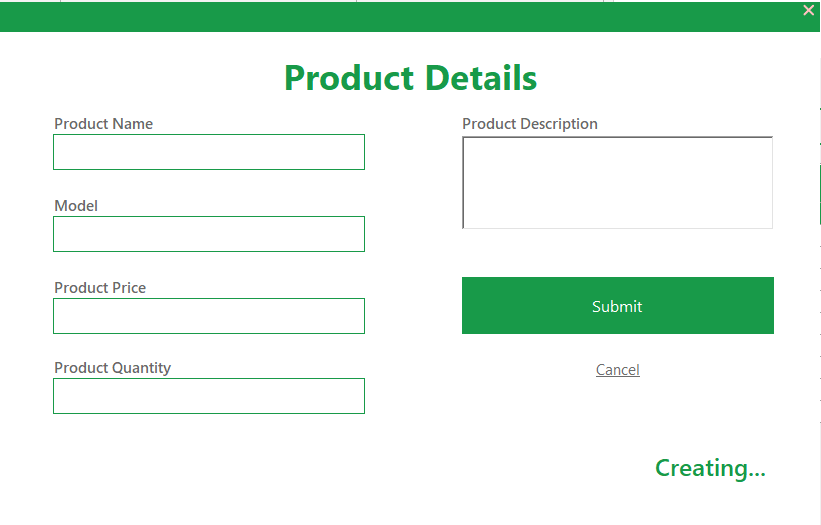


Figure 5.1.6 Product details on Create a new product

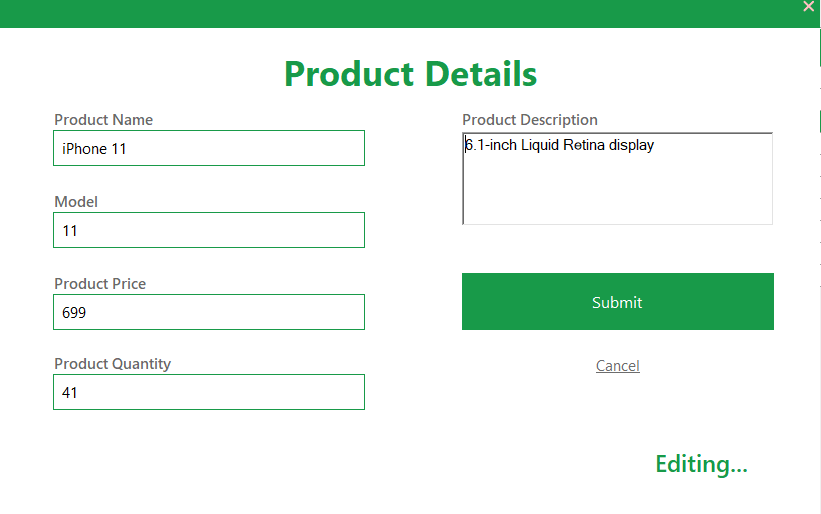


Figure 5.1.7 Products details form on Edit an existed product

- Goods Received Note form: we can search for a specific note by where we received products and perform create. edit, delete a note with functional buttons.

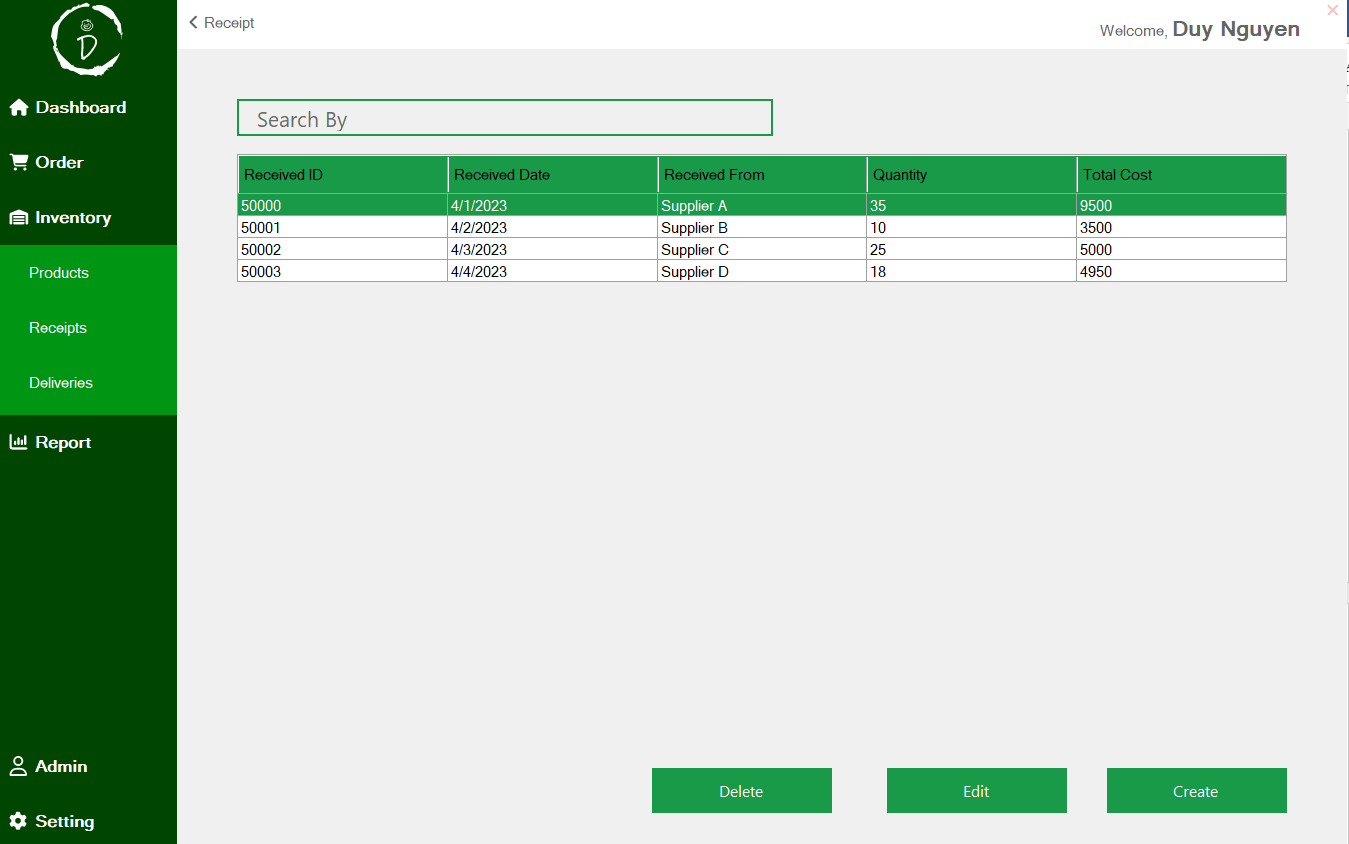


Figure 5.1.8 Goods Received Note form

- Receipt details form: perform create and edit function. For edit, you need to select a specific note first.

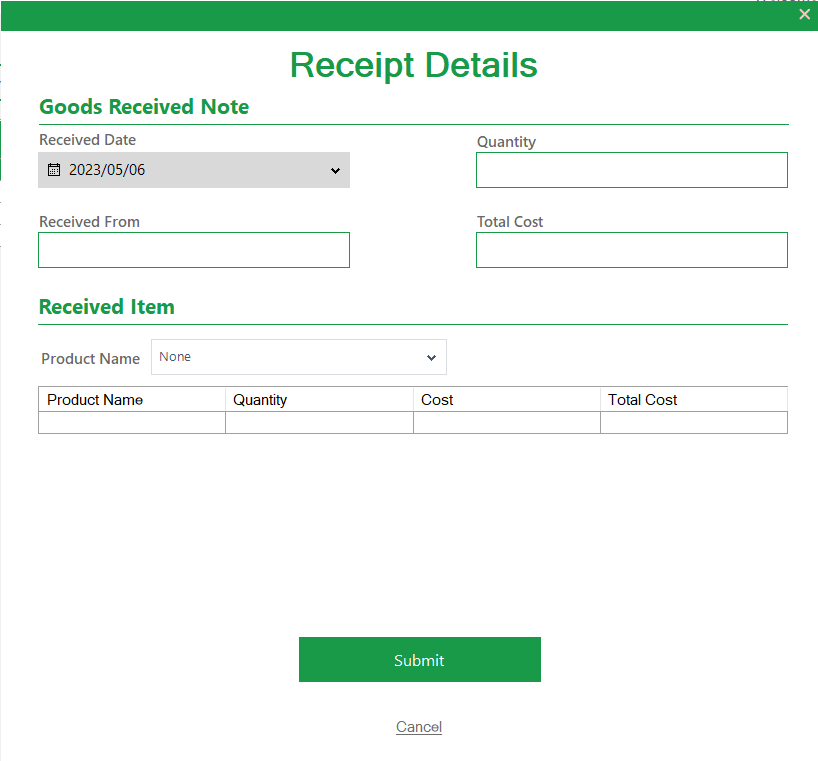


Figure 5.1.9 Receipt Details form on Create a new Goods Received Note

- Delivery Note form: double click in a specific note to view the details.

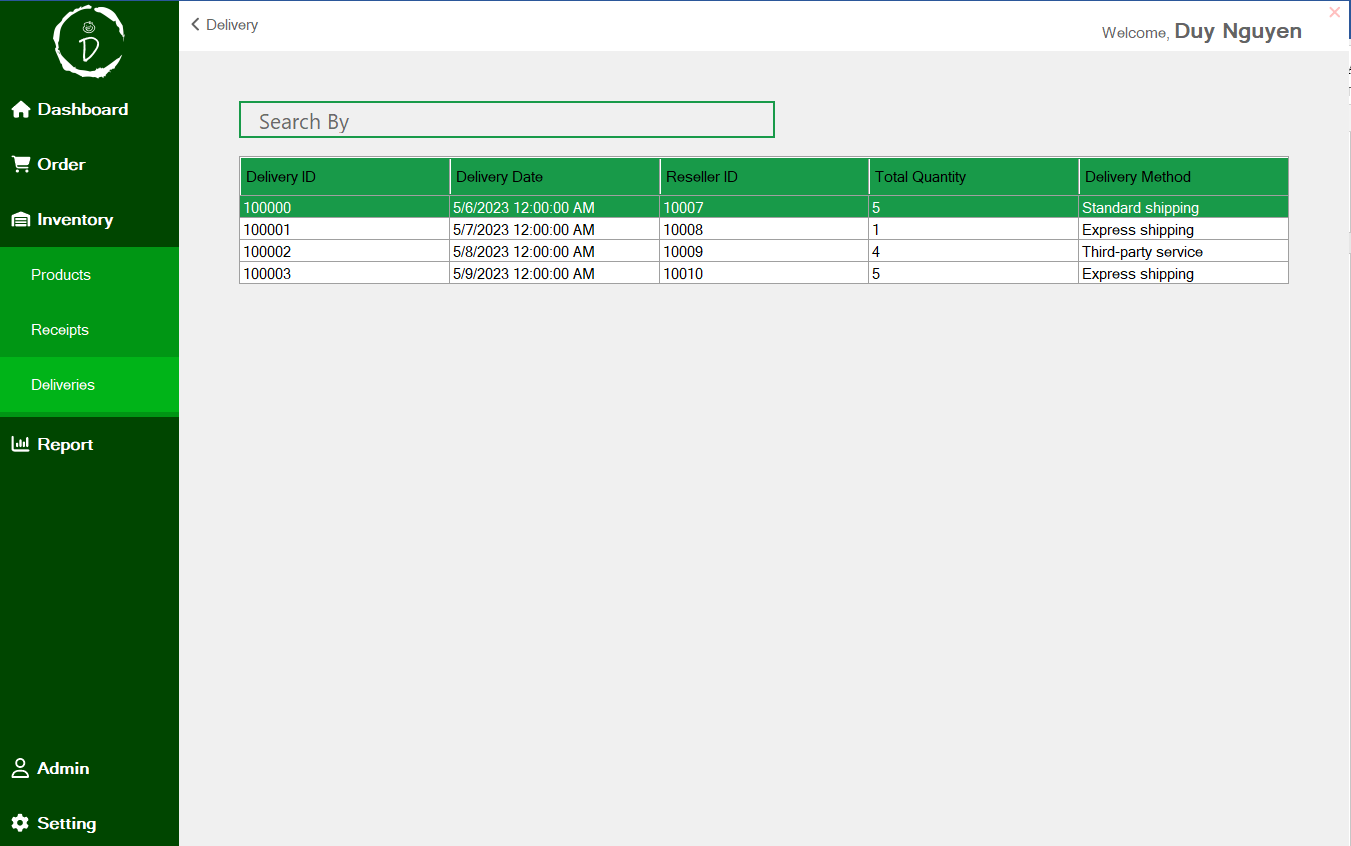


Figure 5.1.10 Delivery Note form

Figure 5.1.10 Delivery Note form

- Delivery details form:

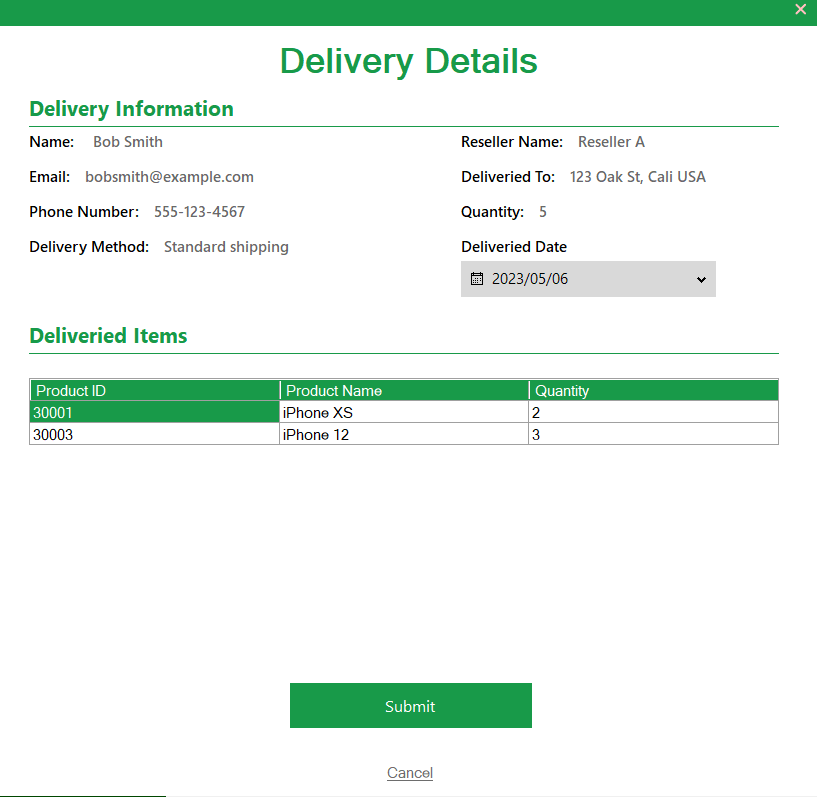


Figure 5.1.11 Delivery Details form

- Stock Report form: view the stock report on each product.

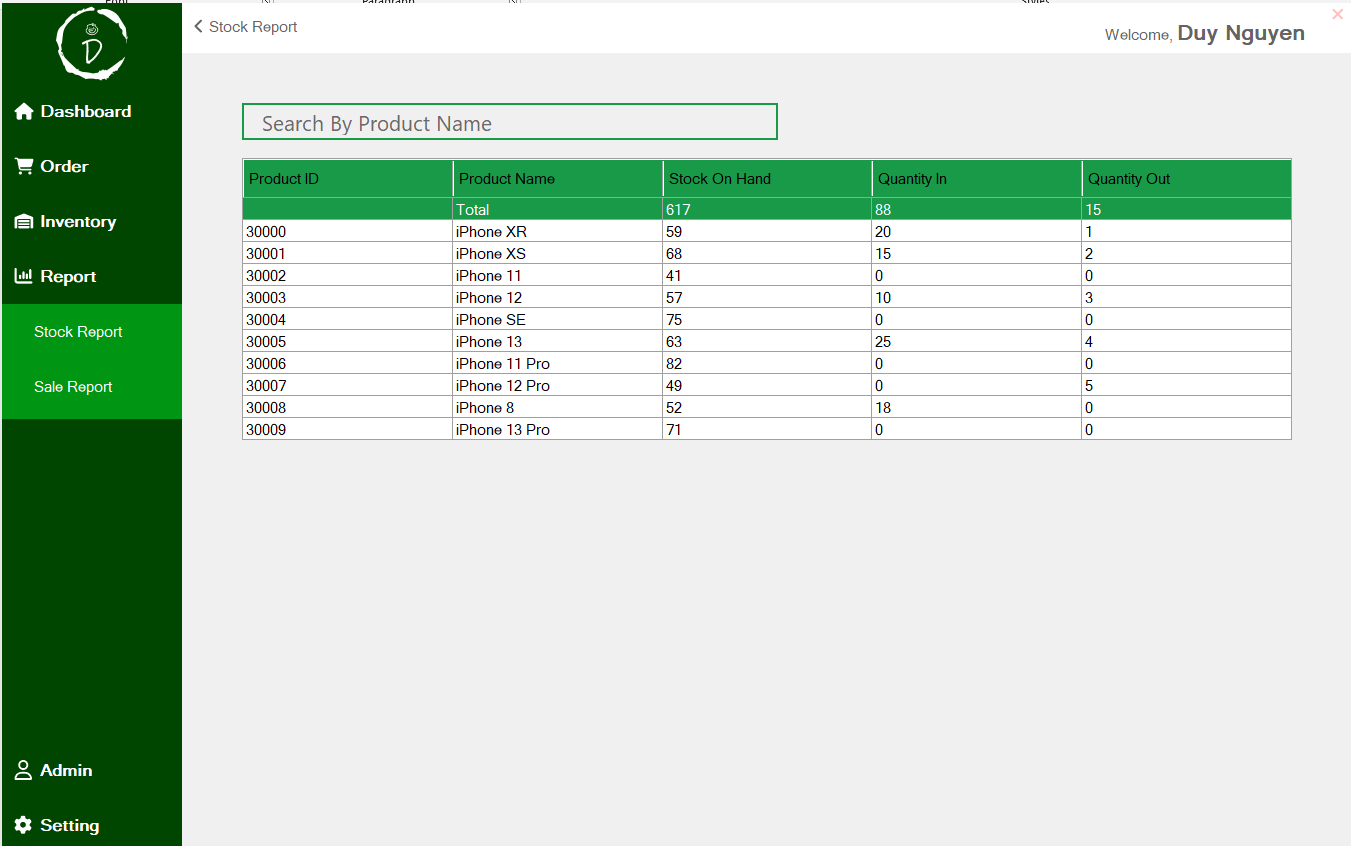


Figure 5.1.12 Stock Report form

- Sale Report form: view the sale report on each product.

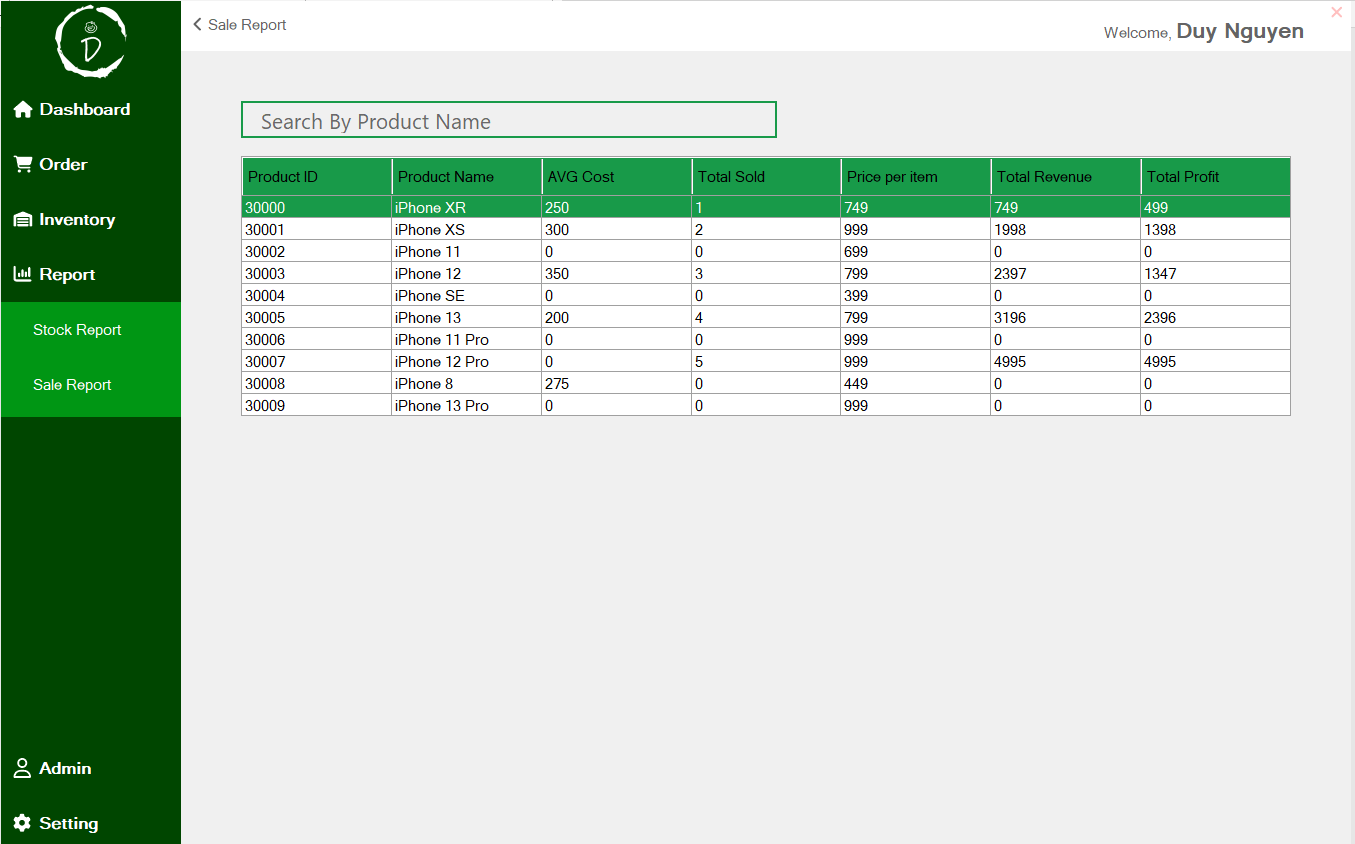


Figure 5.1.13 Sale Report form

- Admin form:

5.2 Static model – class diagrams

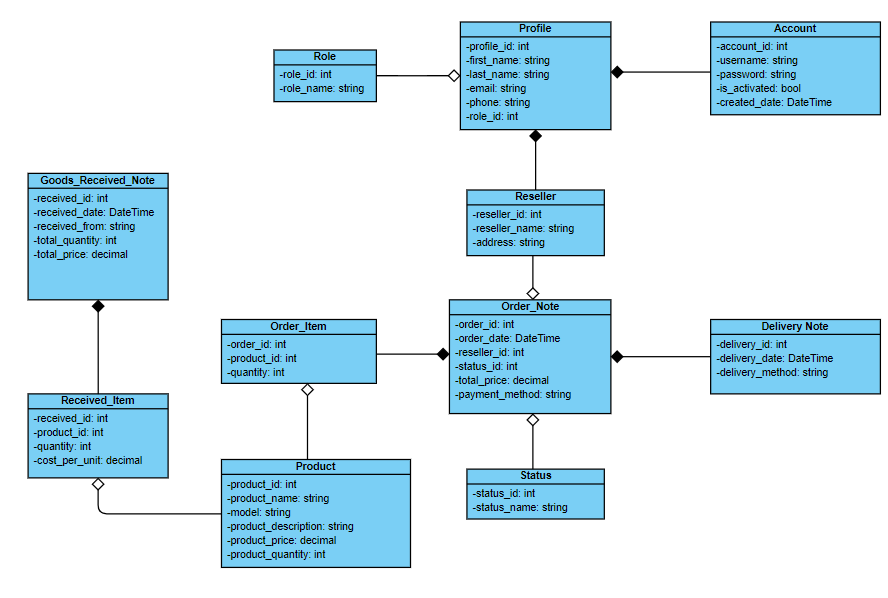


Figure 5.2.1 Class diagram

5.3 Dynamic model – sequence diagrams

* Sequence diagram for use case Login:

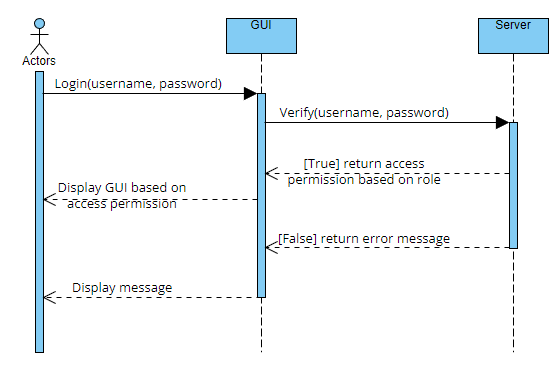


Figure 5.3.1 Sequence diagram for use case Login

* Sequence diagram for use case Create Goods Received Note:

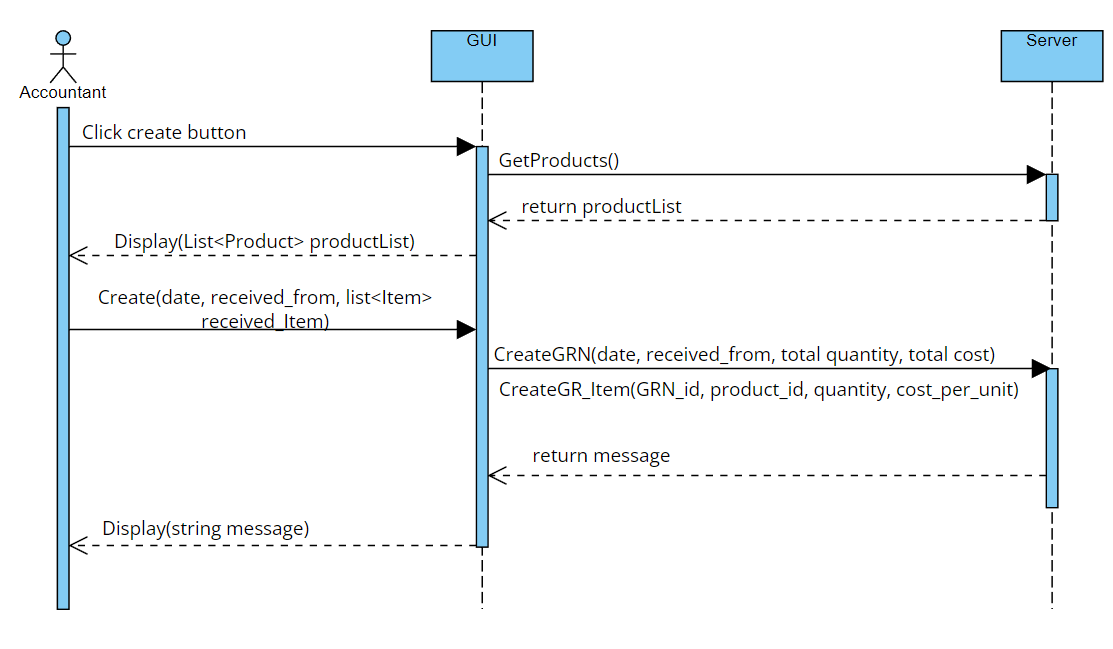


Figure 5.3.2 Sequence diagram for use case Create Goods Received Note

* Sequence diagram for use case Create Goods Delivery Note:

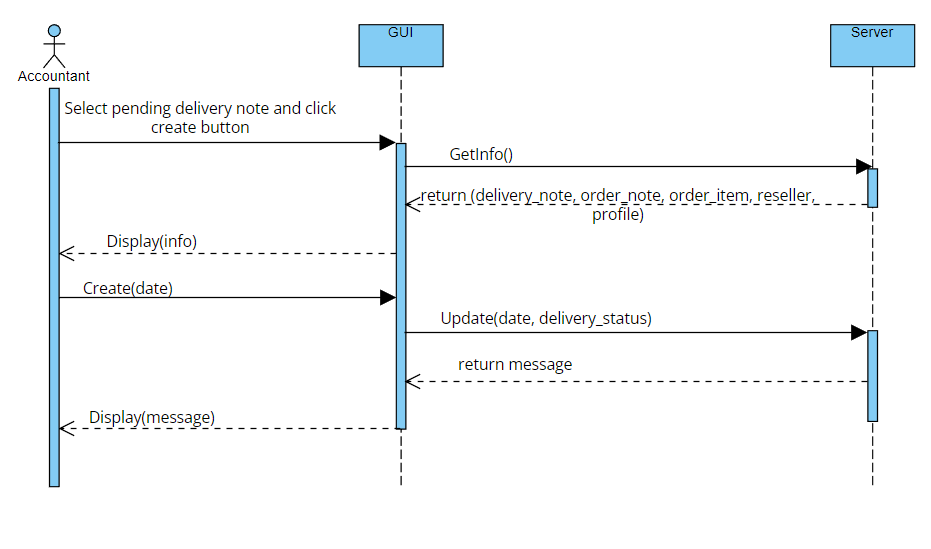


Figure 5.3.3 Sequence diagram for use case Create Goods Delivery Note

* Sequence diagram for use case Manage product:

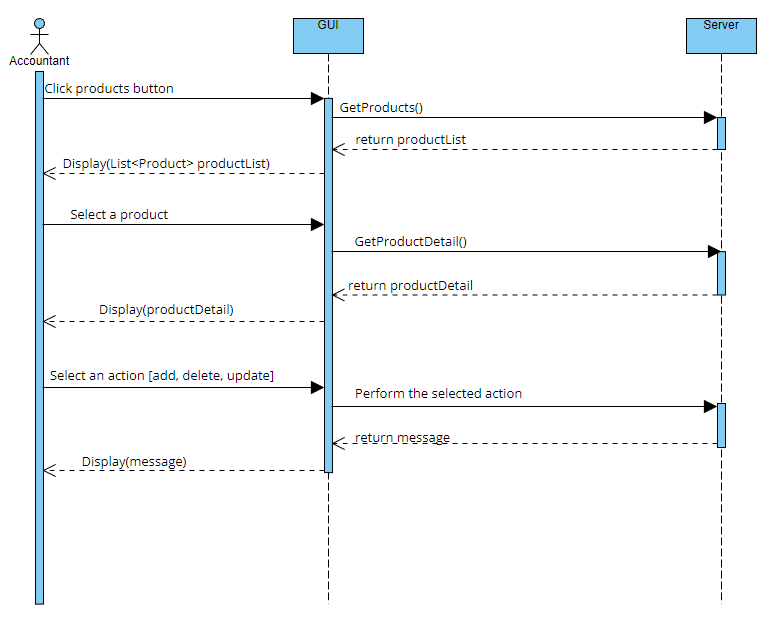


Figure 5.3.4 Sequence diagram for use case Manage product

* Sequence diagram for use case Place Order:

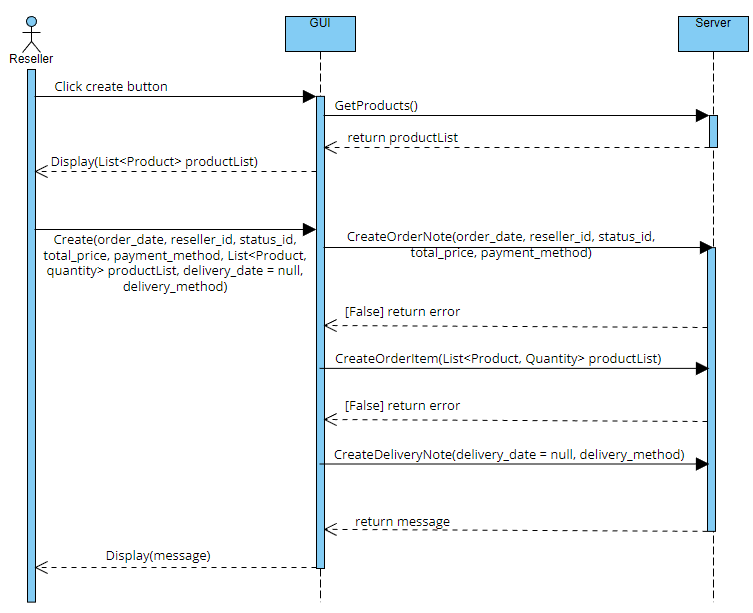


Figure 5.3.5 Sequence diagram for use case Place Order

* Sequence diagram for use case Manage Orders:

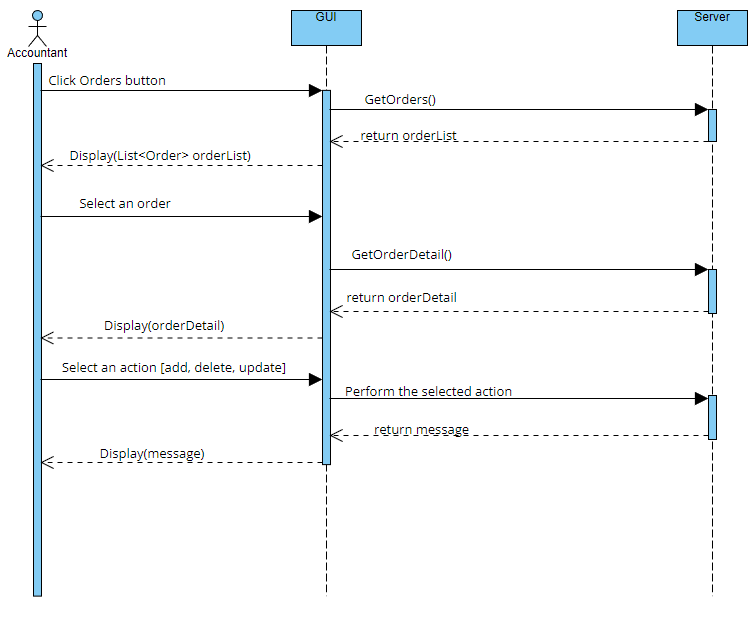


Figure 5.3.6 Sequence diagram for use case Manage Orders

* Sequence diagram for use case View Sale Report:

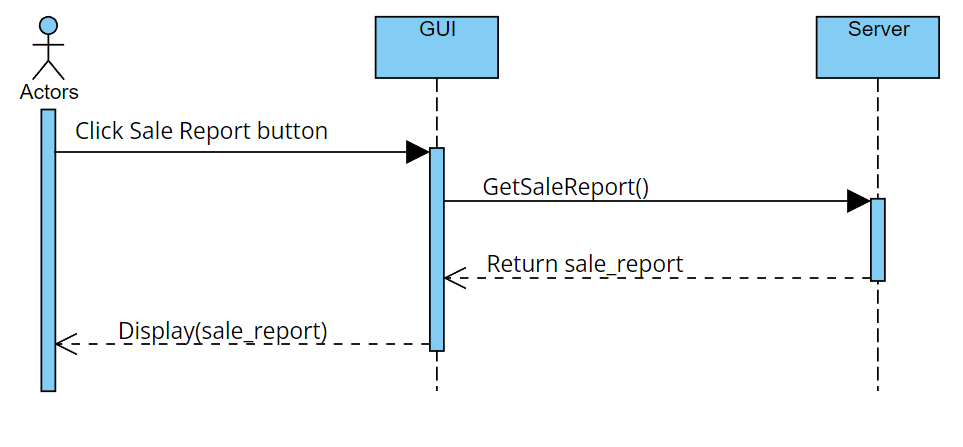


Figure 5.3.7 Sequence diagram for use case View Sale Report

* Sequence diagram for use case View Stock Report:

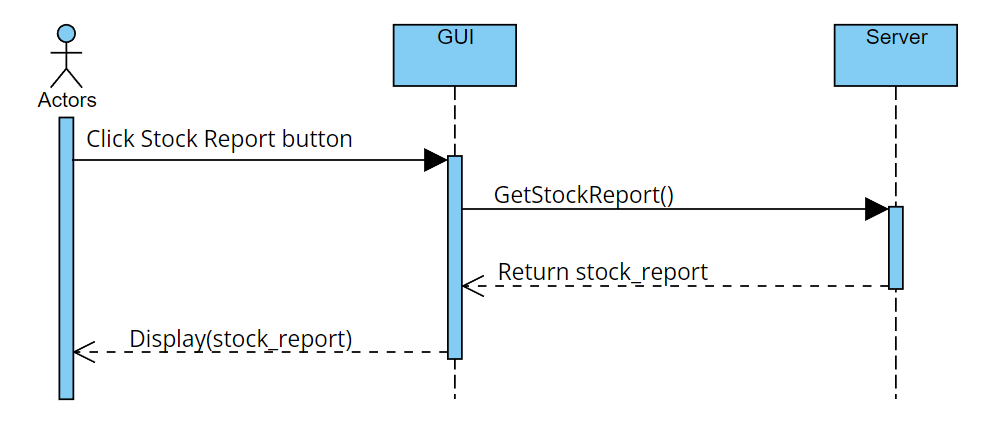


Figure 5.3.8 Sequence diagram for use case View Stock Report

* Sequence diagram for use case Manage User:

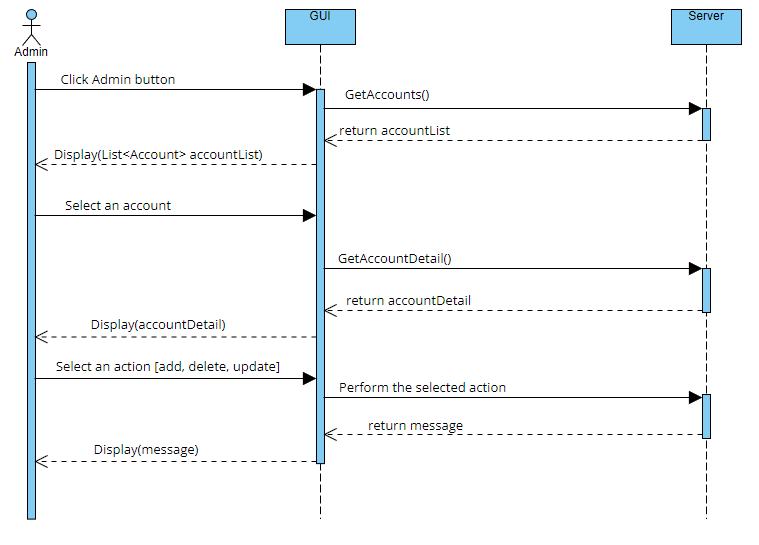


Figure 5.3.9 Sequence diagram for use case Mange User

CHAPTER 6 – TEST PLAN

6.1 Requirements/specifications-based system level test cases

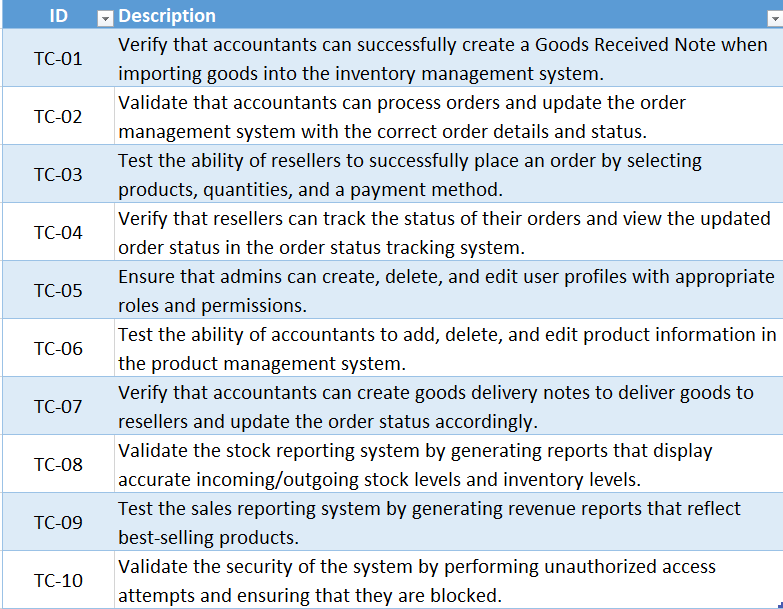


Figure 6.1.1 System level test cases

6.2 Traceability of test cases to use cases

* Analyze each use cases and identify the main functions and interactions with the system.
* Based on the functions and interactions, identify test scenarios and test cases that cover different conditions.
* Map each test case to the corresponding use case.

6.3 Techniques used for test generation

- In this test plan, black box-based testing is mainly used.

- With black-box based testing, it focuses on testing the external behavior of the system. Test cases are generated based on functional requirements, specifications, and expected outputs.

- With white-box based testing, it focuses on testing the internal structure. Test cases are generated based on the system implementation including code analysis, control flow and data flow.

- For measuring the quality of the tests, we can follow these criteria:

* Test coverage: evaluate how effectively the test cases cover the system's requirements, use cases, and different paths.
* Test effectiveness: evaluate how effectively the test cases detect system issues.
* Test maintainability: Consider how easy it will be to maintain and update test cases when the system evolves.
* Test traceability: Ensure that there is a clear traceability link between test cases, requirements, and use cases.

CHAPTER 7 – DEMO

7.1 Database

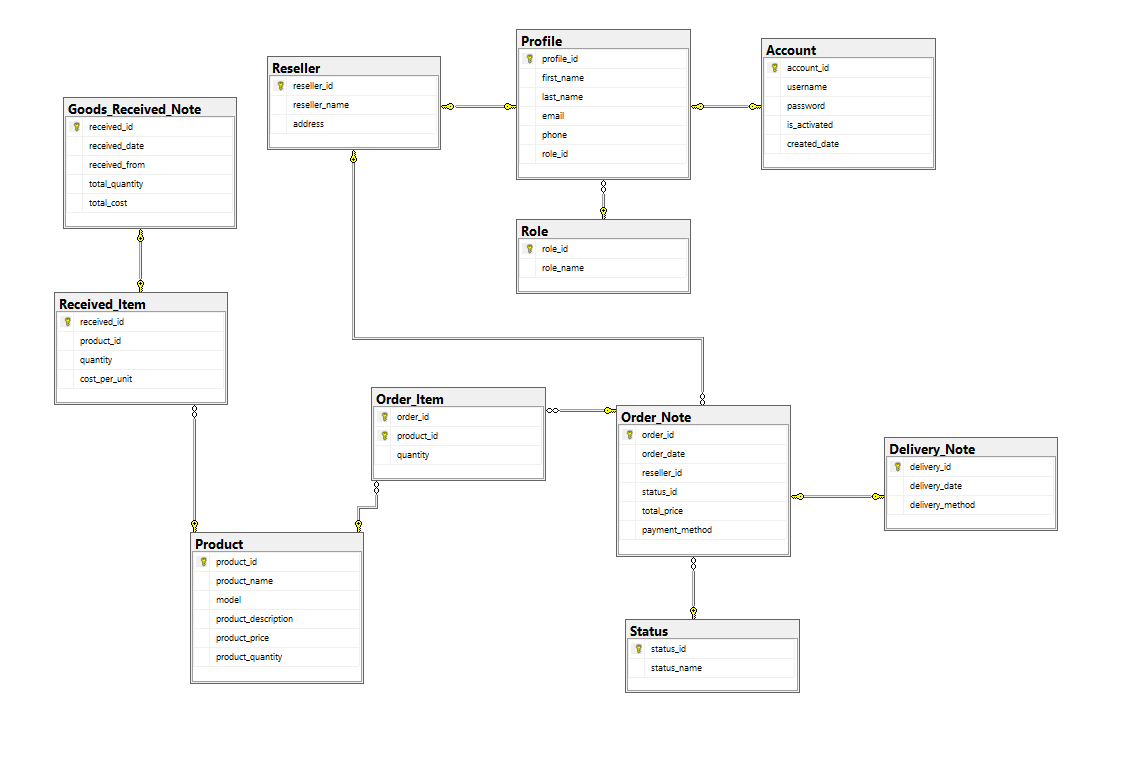


Figure 7.1.1 Relational Model

7.2 Software output

7.3 Github activites

REFERENCES

[1] TutorialsPoint, "SDLC - Waterfall Model," last modified January 8, 2021, accessed May 2, 2023, https://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm.

[2] Wikipedia, "Model-View-Controller," last modified March 22, 2023, accessed May 3, 2023, <https://en.wikipedia.org/wiki/Model-view-controller>.

[3] eCanarys, "3 Layered Architecture," last modified November 28, 2016, accessed May 4, 2023, <https://www.ecanarys.com/Blogs/ArticleID/76/3-Layered-Architecture>.