ELECTRONICS DEVICES PRODUCTS

Requirements Analysis & Design (RAD)
By Students:

1. **StudentID1 : Nguyen Van Hung**
2. **StudentID2 : Le Huynh My Duyen**

Reference: **Team_18_RAD_Requirements_Modelling_v0.1**

Audience: **Mr. Pham Thai Ky Trung** **Document Version:** **7 May, 2023**

Outcome: **Distributing system electronics devices products**

Abstract: This document provides an in-depth analysis of a distributor selling electronics devices producst system with the requirements modelled utilizing the UML framework. The document is a collaboration between the members of Team 18.

Intellectual Property

Table of Contents

Table of Figures

Executive Summary

This report summarizes the process of completing the Team 18 RAD final assignment. It presents the necessary diagrams when designing electronics devices products sales system. This report consists of five major sections:

- Business Requirements: Requirements and benefits of this system in business
- System Requirement Analysis: Analysis of the requirements required when designing the system
- System Requirement Design: The process which showed designing the database, the system code, UI,…
- System Requirement Implementation: The completed steps of the system design & build, show the UI design, SQL code,…
- System Testing: The result of testing, Show the test plan, Demo of system.

I. Business Requirements (1 point)

1.1 Gantt Chart

TON DUC THANG UNIVERISTY

Project Start: Fri, 4/14/2023

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Analysis Tasks			2 days	
Meeting with the Purchasing manager	Van Hung - My Duyen	75%	4/14/23	4/14/23
Meeting with several purchasing Agents	Van Hung - My Duyen	80%	4/14/23	4/14/23
Meeting with Accountants	My Duyen	75%	4/14/23	4/15/23
Meeting with the Purchasing manager	Van Hung	100%	4/15/23	4/15/23
Identify infor and use case	Van Hung - My Duyen	100%	4/15/23	4/15/23
Designs Tasks			4 days	
Design screen layouts	Van Hung - My Duyen	50%	4/16/23	4/17/23
Design database	Van Hung	60%	4/16/23	4/18/23
Design overall architecture	My Duyen	75%	4/16/23	4/19/23
Design program details	Van Hung - My Duyen	60%	4/16/23	4/20/23
Build Tasks			5 days	
Code and uni test GUI layer program	Van Hung	50%	4/20/23	4/25/23
Code and uni test Logic layer program	My Duyen	65%	4/20/23	4/25/23
Test and Deploy Tasks			1 days	
Perform system fuctionality tests	Van Hung - My Duyen	100%	4/26/23	4/26/23
Perform user acceptance test	Van Hung - My Duyen	100%	4/26/23	4/26/23
Finish the project	Van Hung - My Duyen	100%	4/26/23	4/26/23

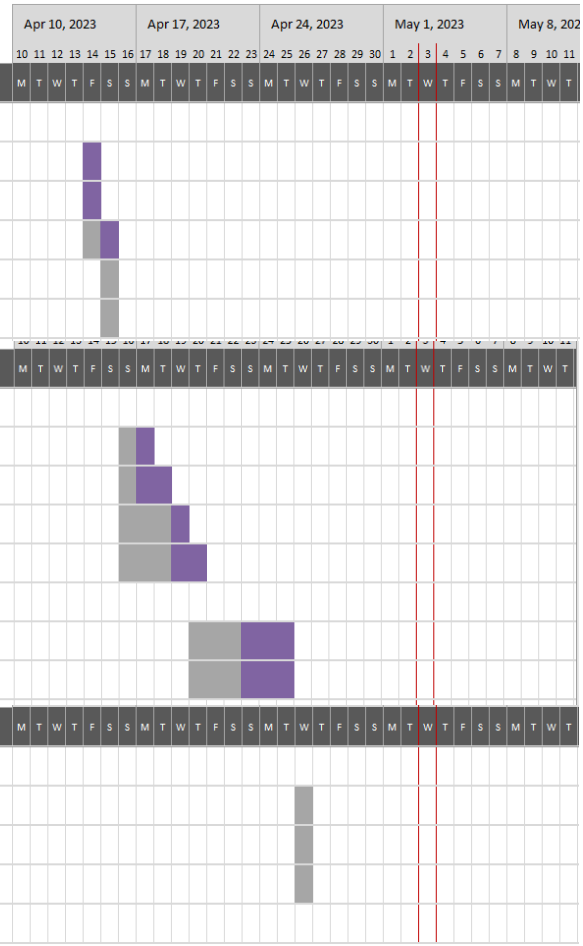


Figure 1: Gantt Chart

1.2 Business Modelling / Requirements

Business use – case diagram

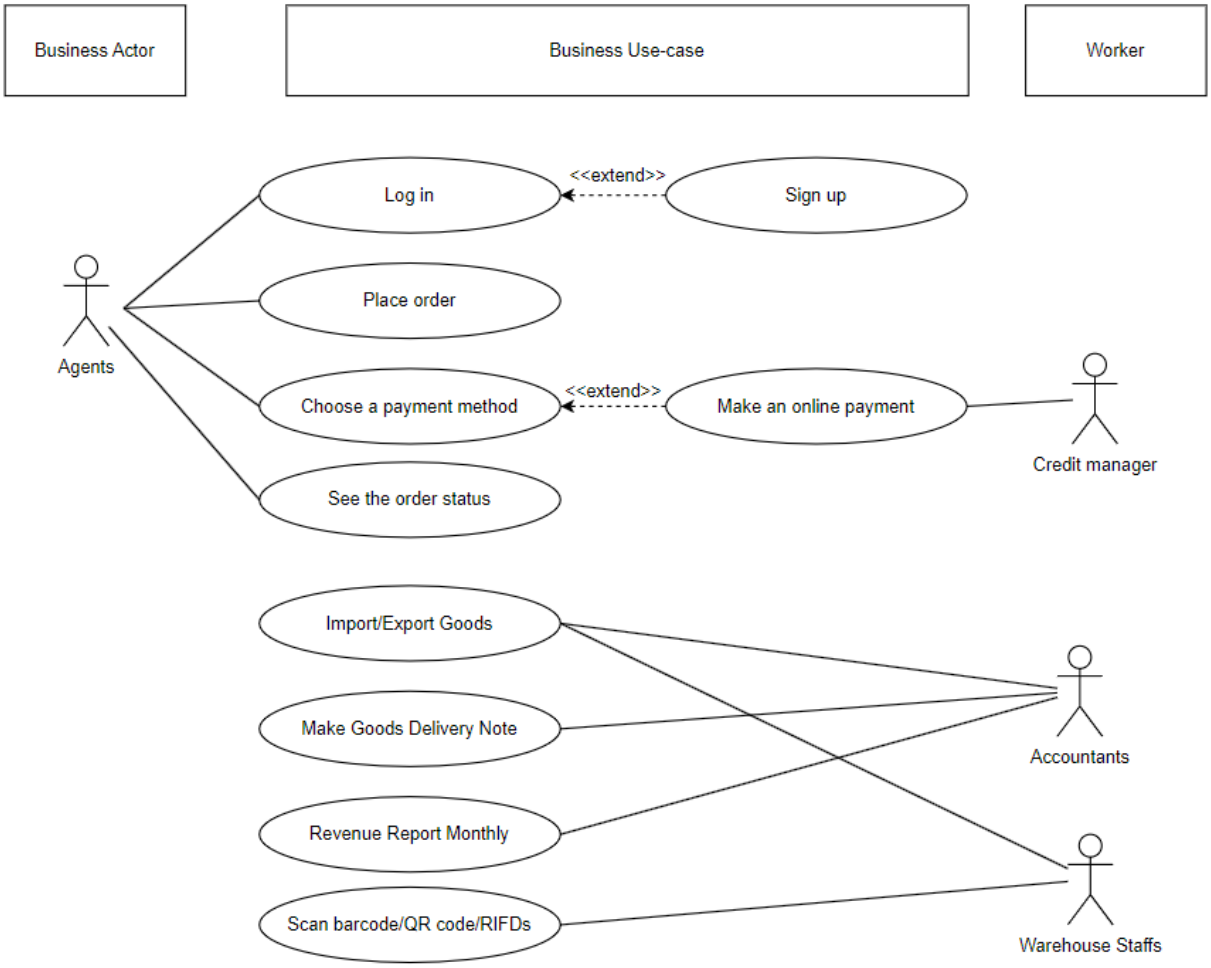
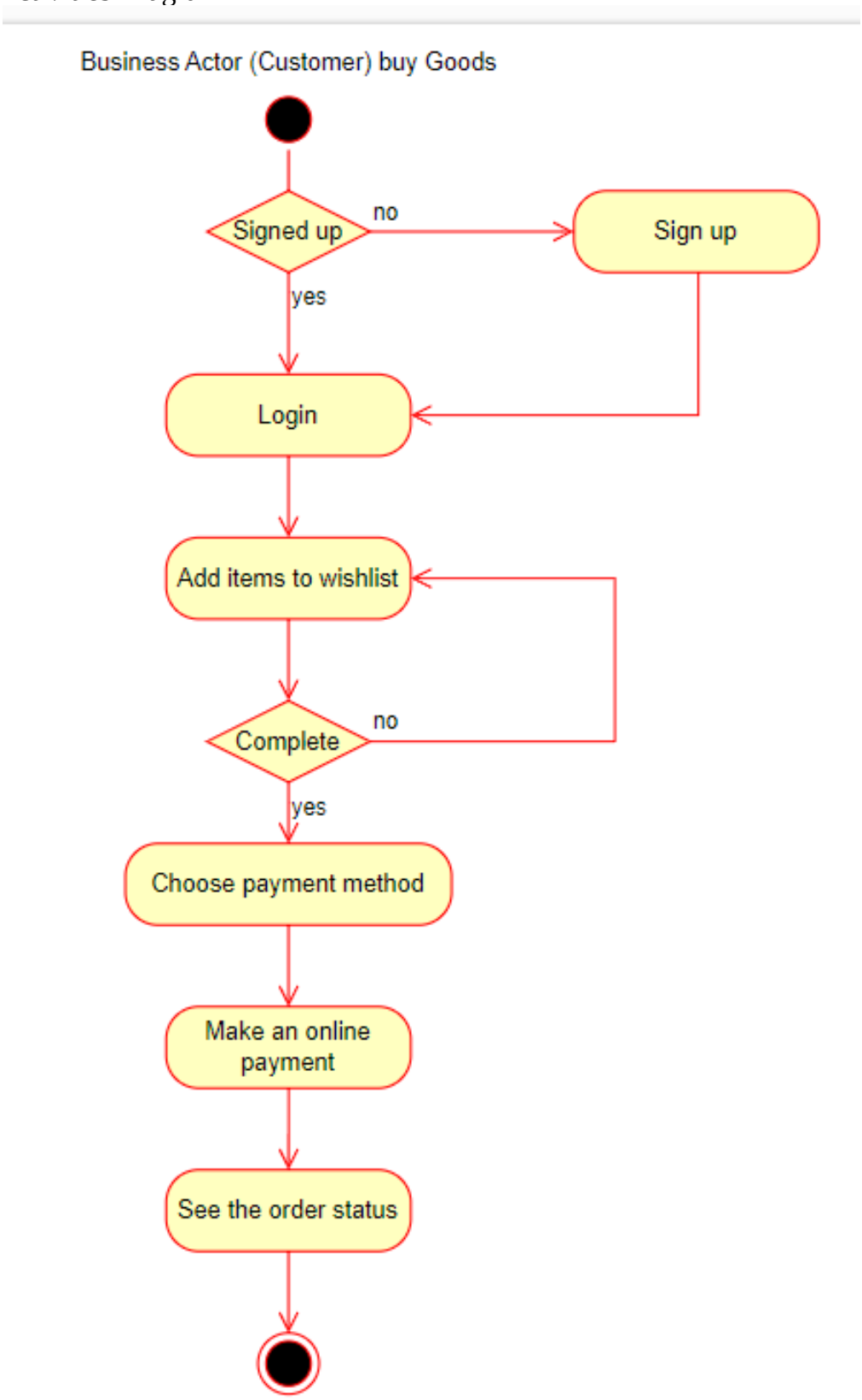
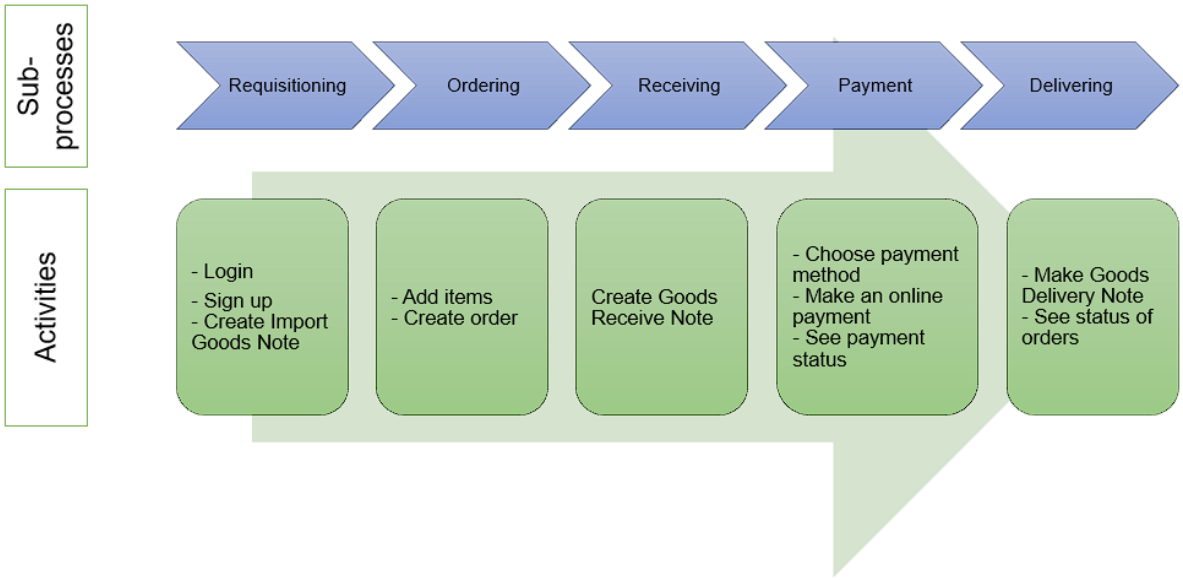


Figure 2: Business use – case diagram

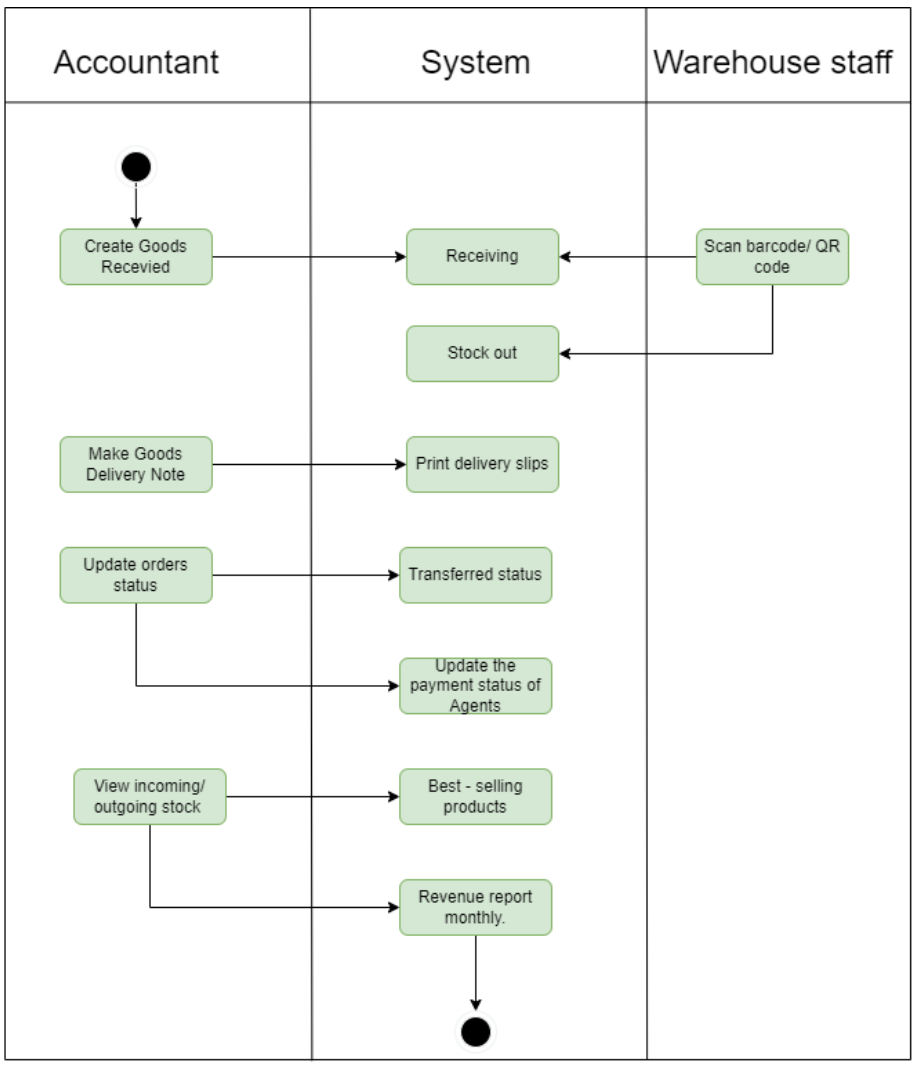
Business Activities Diagram



1.3 Business Processes / Flowchart of Requirements



1.4 Activity Diagram



1.5 List of Requirements

Requirements	Functional	Nonfunctional
The user's interface's background could be modified		x
The system must allow the user login to system after signup	x	
The system must allow accountants to add new Goods into database	x	
The system must allow the user to view the location of delivery	x	
The form to login and signup must be placed in the middle of system		x
The system must perform all the quantity of Goods in invetory	x	
The system must allow the user see the items in wishlist	x	
The system must allow the user see **the payment status**	x	
The system have a company's icon in the left corner		x
The system must be linked to database and update the changes in the quantity of Goods every 5 minutes		x
The system must allow the user choose a payment method	x	
The system allows accounting staff to make Goods Delivery Note	x	
The system allows accounting staff update the status of orders as being transferred and update the payment status of agents.	x	
The system must allow accountants view incoming/outgoing stock report (inventory movement), best-selling products and revenue report monthly.	x	

II. System Requirements Analysis (3.0 points)

2.1 Translate from Business Use Case

2.1.1. Problem Description

- There are many types electronic device, so it is hard to manage and store them in a way that can find, import, export easily.
- It is import to have a system, which can support Accountants and Agents access to information and manage/ order items. In addition, it is recommended that system can allows staff to make Goods Delivery Note in order to delivery exactly. Moreover, Agents could see the status of their orders by the system.
- Also, the system shouble be deployed on portable equipments.

2.1.2. System capabilities

A Distributor selling electronics devices products to authorized resller/ agents, the new system should be capable of:

- Accoutants shall be able to create Goods Received when the distributor imports goods (a warehouse receipt will include many items).
- When receiving goods at the warehouse, warehouse staff will scan barcode/ QR code, RIFDs to perform goods warehousing process.
- Reseller / Agents shall be able to place an order of items by themselves and choose a payment method (Cash, bank transfer, Momo...) and make an online payment and see the status of their orders.
- The system allows accounting staff to make Goods Delivery Note (based on previously placed orders) to deliver goods to agents (print delivery slips), update the status of orders as being transferred and update the payment status of agents. Also, the system allows warehouse staff make use of barcode/ QR code/ RFIDs for stock out.
- Accountants shall be able to view incoming/outgoing stock report (inventory movement), best-selling products and revenue report monthly.

2.1.3. Business benefits.

It is anticipated that the deployment of this new system will provide the following business benefits:

- Manage and store goods easily, so it is less time consumption to import/ export goods, thereby improving the quality and speed of purchase order decisions
- Maintain correct and current information about suppliers and see the status of their orders, thereby facilitating rapid communication between suppliers and agents
- Provide online payment method, thereby the payment is done quickly and conveniently

2.1.4. Users and their goals

User/ Actor	User Goal	Description
Accountant	Log in	Account must log in in the system to perform functions
	Create Goods Received	Enter new goods detail to import, or update the quanity when import goods in warehouse
	Make Goods Delivery Note	Accountant create new goods delivery note based on previously placed orders to transfer to Agents
	Update orders status	Choose items, choose payment method, enter agent information (address, phone number or email,…)
	Manage incoming/ ougoing stock	use inventory movement, calculate the average of the quanity of import/ export goods
Warehouse staff	Scan barcode/ QR code	when receiveing or delivering goods at the warehouse, warehouse staff will scan to perform the goods warehousing process
	Login	Warehouse staff must log in in the system to perform functions
Reseller/ Agents	Order of items	Agent shall able to place an order of items by themselves.
	Make Payment	When agents/ resellers enter products then could choose a payment method (cash, bank,…)
	View status of orders	see agent use payment information, status of order (transfer, stock out, received,….), see update the status of orders

2.1.5. List of Events

List of events and its use case:

EVENT	TRIGGER	SOURCE	USE CASE	RESPONE	DESITINATION
Accountant create delivery note	New note	Accoutant	Make Goods Delivery Note	Report details	Accoutant Warehouse staff
When delivery products, warehouse scan barcode/ QR code	Scan barcode/ Qrcode, RFIDs	Warehouse staff	Scan barcode/ QR code	Delivery/ recceived products in warehouse	Warehouse staff
Update quanity when received/ delivery products in warehouse	Update quanity	Accountant	Update quanity	Change quanity	Accoutant
Accoutant want to update orders status	Update order status	Accoutant	Update orders status	Update order stauts details	Accoutant

2.1.6. List of Actors

- **Accountant**: is who interacts with the system to create goods delivery note, goods received, update orders status and manage incoming/ outgoing stock
- **Warehouse staff**: is who interacts with the system to scan barcode/ QR code RFIDS of products to perform the goods warehousing process or stock out.
- **Agents/ Reseller**: is who able to place an order of items and choose a payment method and view the status of their order from a distributor.

2.1.7. List of Use Cases

- **Create Goods Recevied:** Enter new goods details to import, update the quanity when import goods in warehouse by Accountant.
- **Make Goods Delivery Note:** Accountant create new goods delivery note based on previously placed orders to transfer to Agents

- **Manage incoming/ outgoing stock:** use inventory movement, calculate the average of the quanity of import/ export goods
- **Update orders status:** when agents chooses items, choose payment method, enter agents information (address, phone or email,…), accountant allow orders status.
- **Scan barcode/ QR code, RIFDs:** when receiveing or delivering goods at the warehouse, warehouse staff will scan to perform the goods warehousing process.
- **Make payment:** Enter new goods details to import, reseller/ agents chooses a payment method (cash, bank,….)
- **View status of orders:** see agent use payment information, status of order (transfer, stock out, received,….), see update the status of orders.
- **Orders of items:** agent shall able to place an order of items by themselves.

2.1.8. Use Case Diagram

The use case diagram of Distributing Electonics Devices Products System can be draw-able as the following:

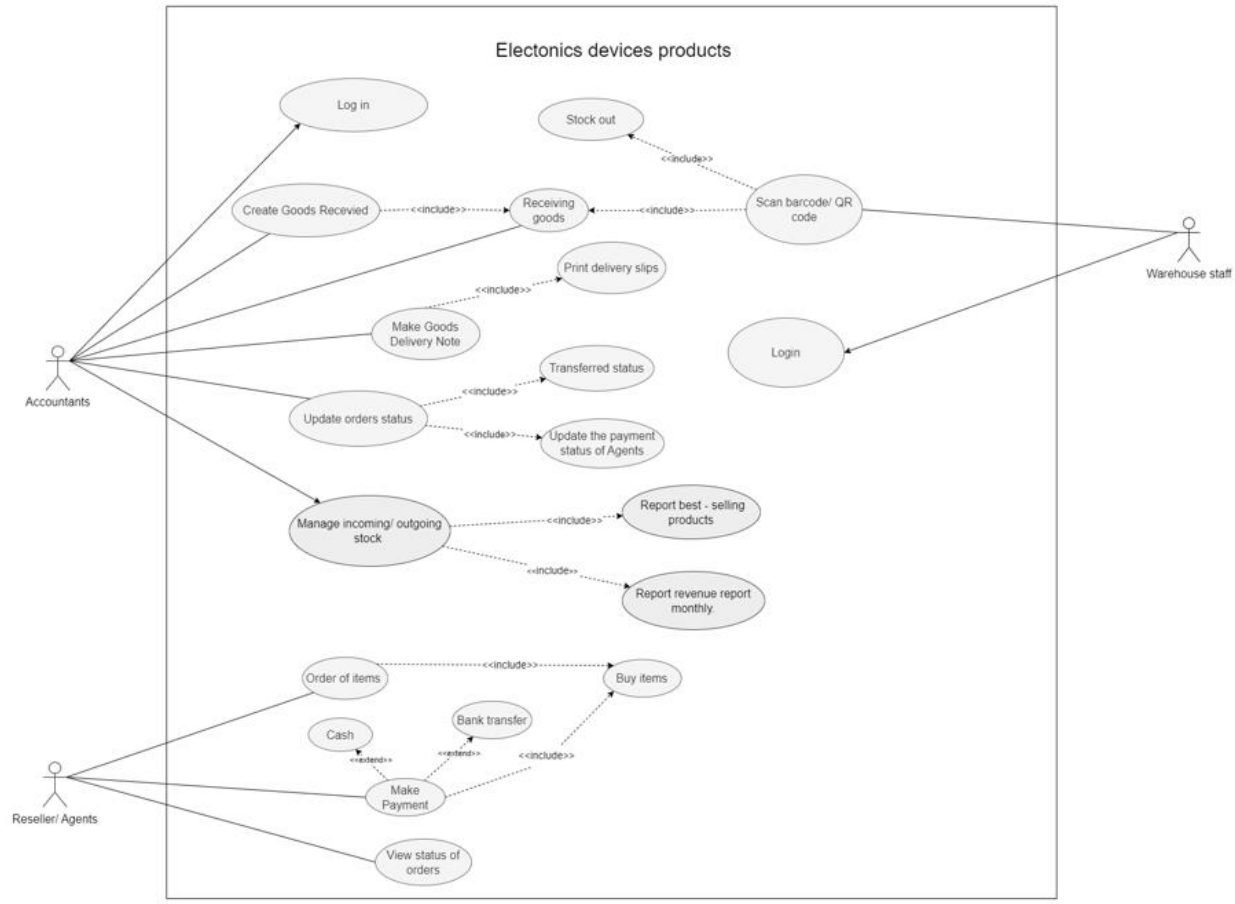


Figure: *Use case for System Diagram*

The use case of Make Goods Delivery Note subsystem diagram can be draw-able as the following:

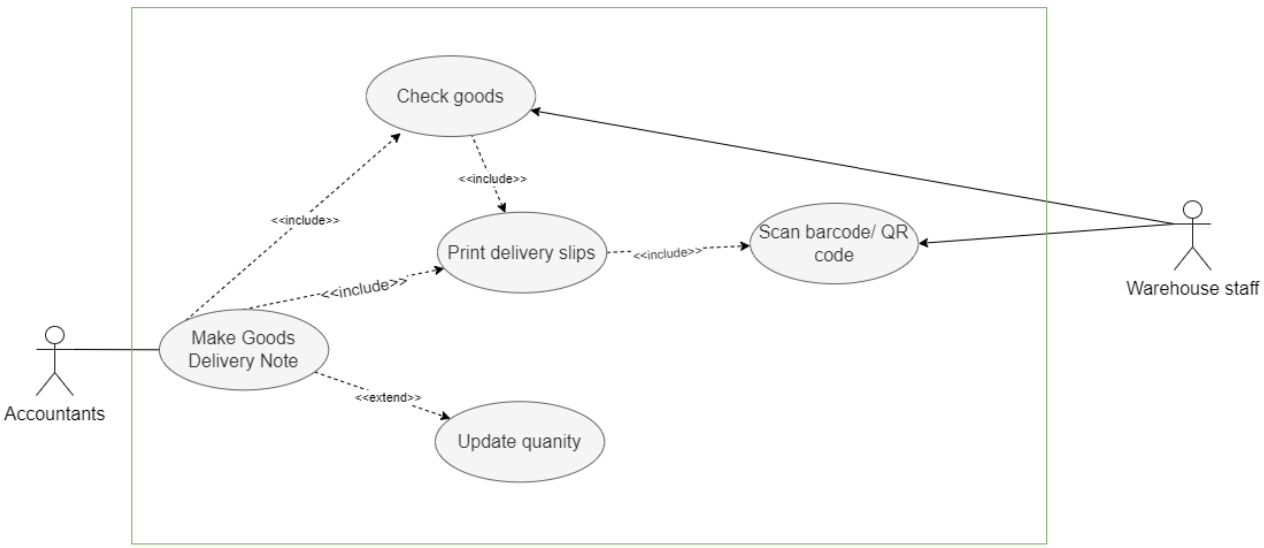


Figure: *Use case subsystem of Make Goods Delivery Note Diagram*

The use case of Make Goods Delivery Note subsystem diagram can be draw-able as the following:

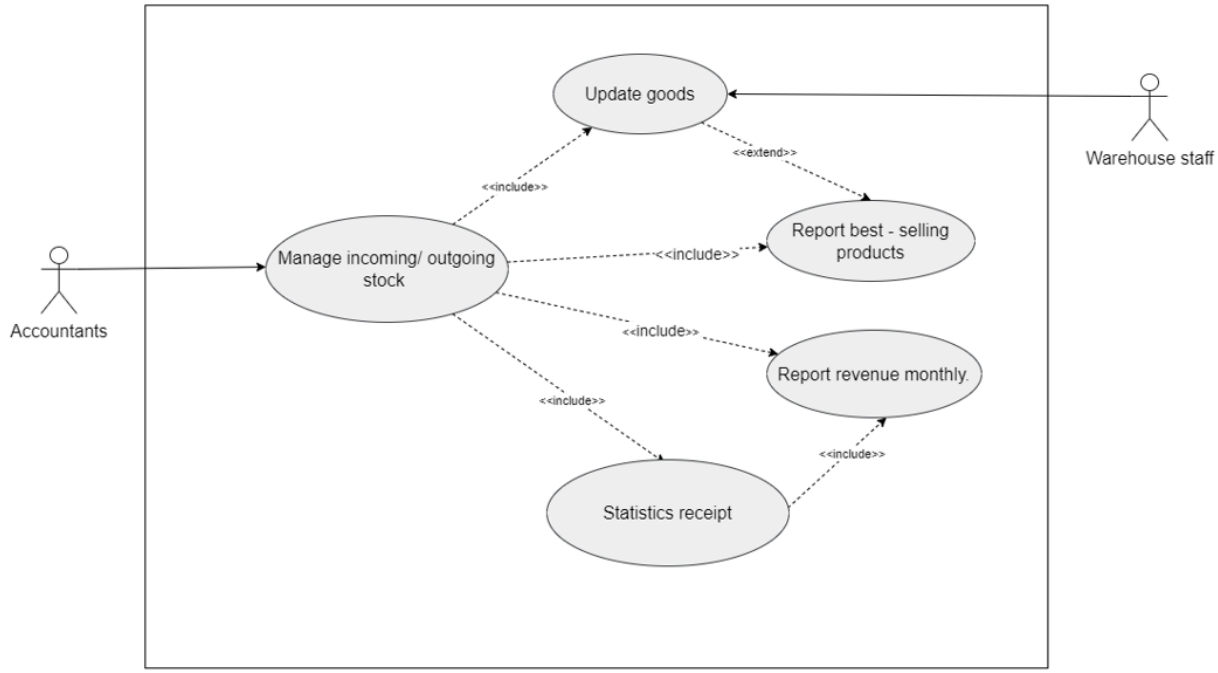
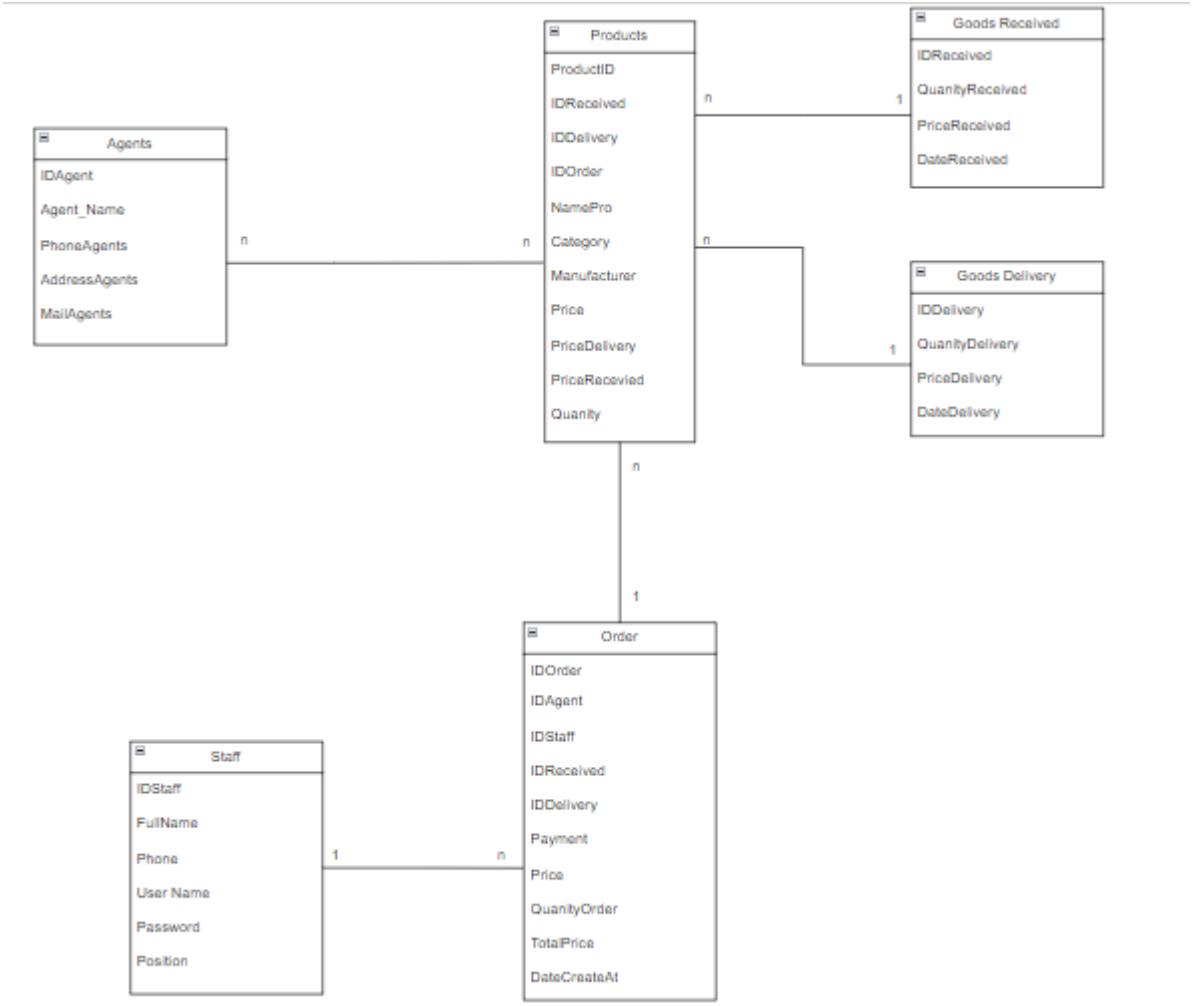


Figure: *Use case subsystem of Manage incoming/ outgoing stock Diagram*

2.1.9. Domain Class Model Diagram



Domain Model Class Make Goods Delivery for Staff Subsystem Diagram

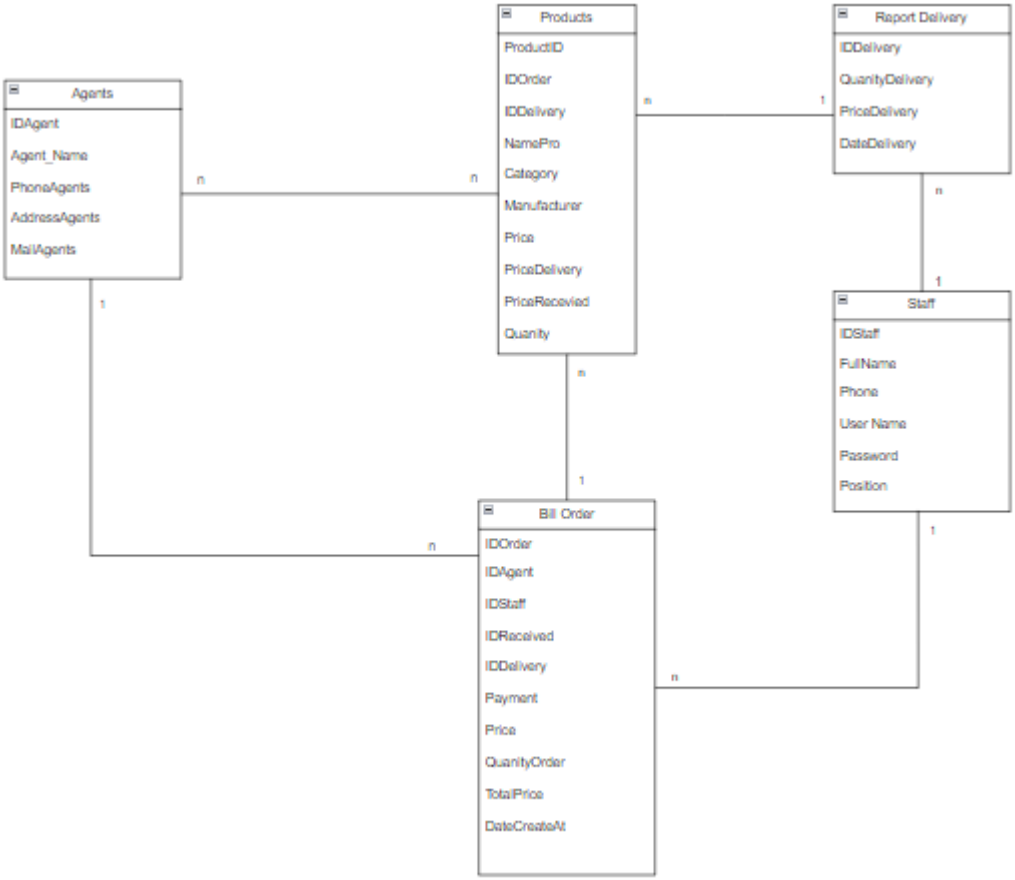


Figure: *Report Goods Delivery of Staff Subsystem Domain Model Class Diagram*

Domain Model Class Manage incoming/ outgoing stock for Staff Subsystem Diagram

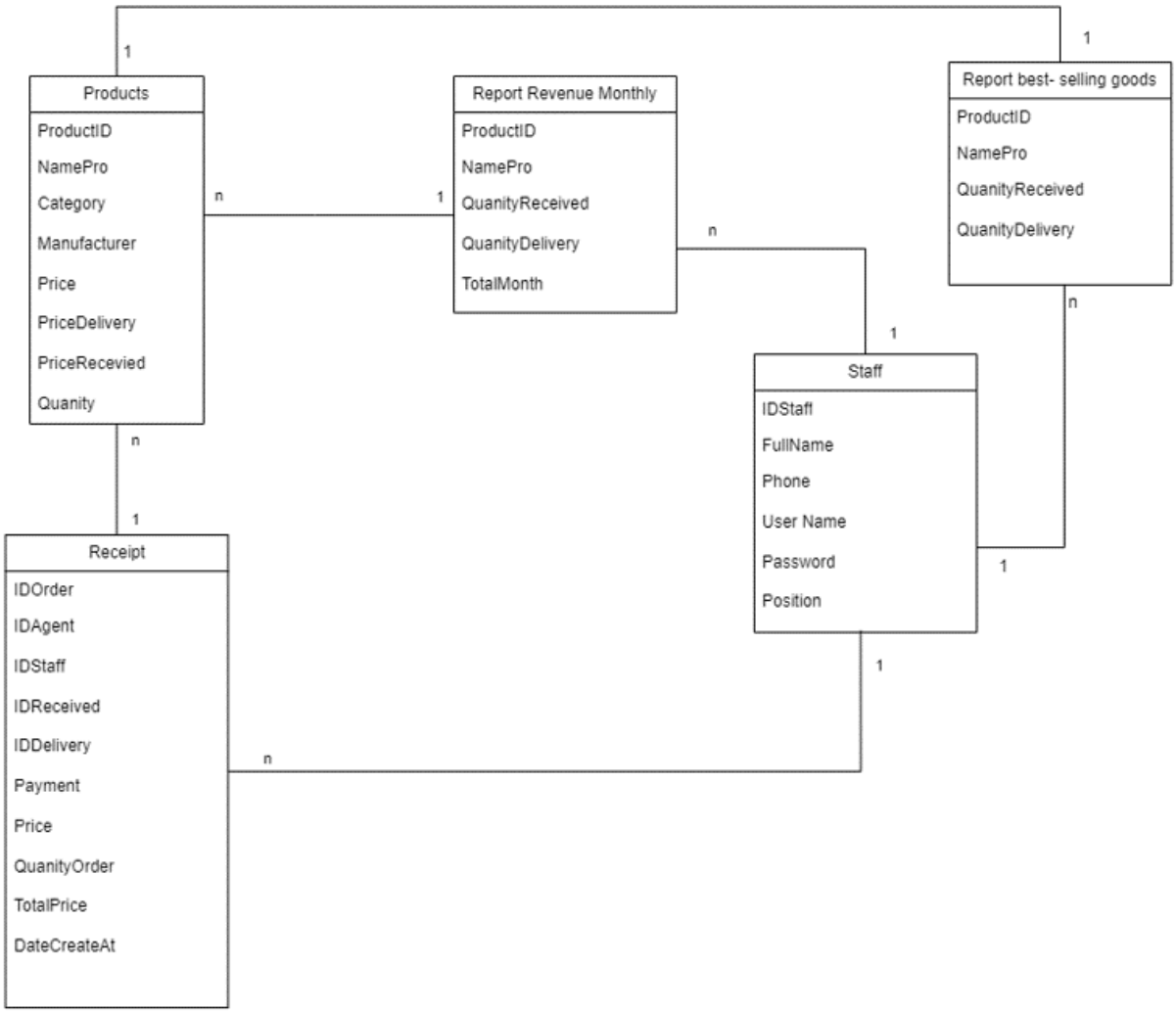
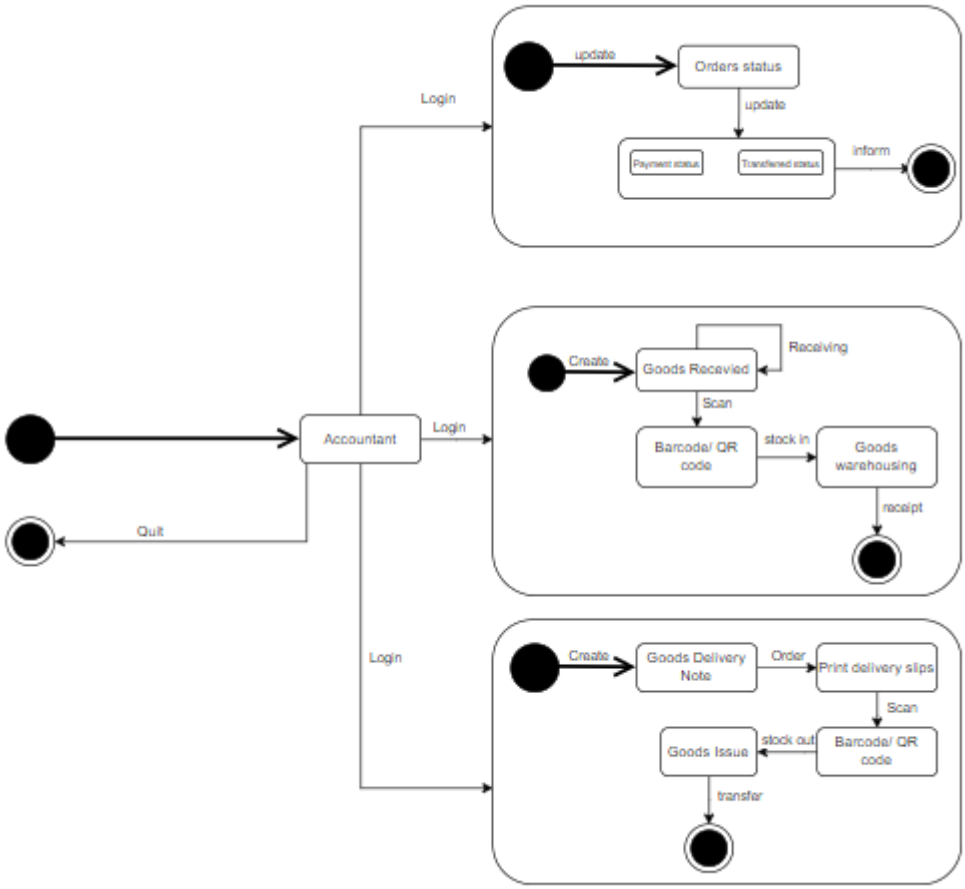


Figure: *Manage incoming/ outgoing stock of Staff Subsystem Domain Model Class Diagram*

2.1.10 State – chart diagram



2.2 Use Case Descriptions

2.2.1. Use Case: Make Goods Delivery Note

i. Use case : Make Goods Delivery Note

ID:	**Make Goods Delivery Note**
Title:	**Make Goods Delivery Note**
Description:	- When a Company wants to delivery more goods, - An Accountant staff to make Goods Delivery Note based on previously placed orders - To deliver goods to agents, account will print delivery slips
Primary Actor:	Accountant
Preconditions:	Reseller/ Agents were previously placed orders. Account must print delivery slips.
Postconditions:	When deliver goods to agens, accountant update the status of orders as being transferred and update payment status of agents.
Main Success Scenario:	Accountant create Goods Delivery Notes based on previously placed orders. Then print delivery slips and warehouse staff make use of barcode/QR to stock out.
Extensions:	The system also allows warehouse staff make use of barcode/QR for stock out.
Frequency of Use:	Frequently
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

ii. Use case: Print delivery slips

ID:	**Print delivery slips**
Title:	**Print delivery slips**
Description:	- When an accountants received previously placed order. - An accoutants makes Goods Delivery Note to deliver goods to agents - An accoutants prints delivery slips to transfering
Primary Actor:	Accountant
Preconditions:	List of products must be available before Reseller/ Agents were previously placed orders.
Postconditions:	Warehouse staff check goods and scan barcode/ QR to stock out
Main Success Scenario:	Accountant create Goods Delivery Notes based on previously placed orders. Then print delivery slips and warehouse staff make use of barcode/QR to stock out.
Extensions:	The system also allows warehouse staff make use of barcode/QR for stock out.
Frequency of Use	Frequently
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

iii. Use case: Check Goods

ID:	**Check Goods**
Title:	**Check Goods**
Description:	- When an accountants received previously placed order. - An accoutants makes Goods Delivery Note to deliver goods to agents to check products. - An accoutants prints delivery slips to transfering
Primary Actor:	Accountant
Preconditions:	List of products must be available before Reseller/ Agents were previously placed orders
Postconditions:	Reseller/ Agents were previously placed orders. Accoutant confirm the order and create Goods Delivery
Main Success Scenario:	Accountant create Goods Delivery Notes based on previously placed orders. Then print delivery slips and check products is avaiable. Next warehouse staff scan barcode/ QR to stock out
Extensions:	The system also allows warehouse staff make use of barcode/QR for stock out, couldn't check goods from accountant.
Frequency of Use	Frequently
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

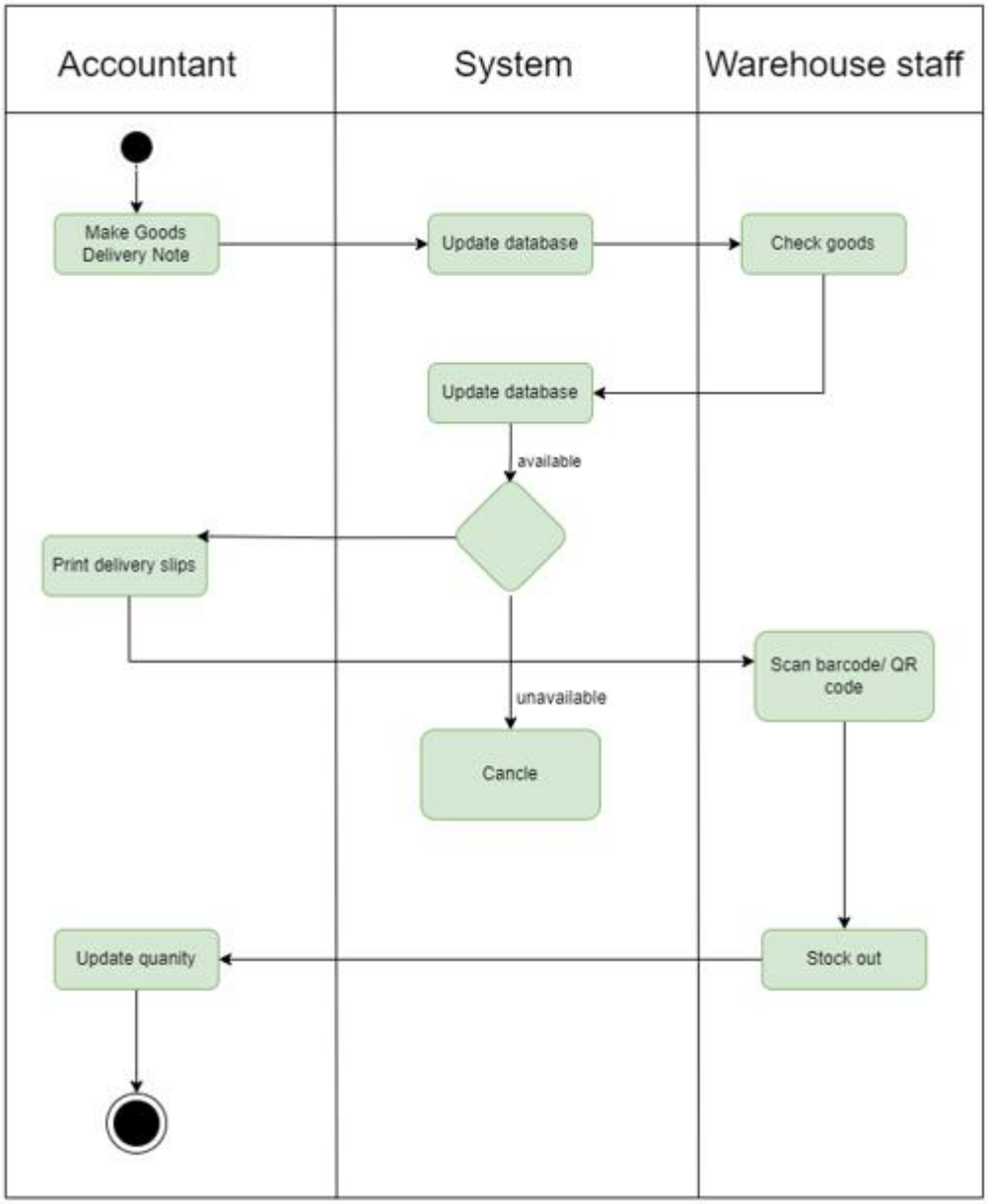
iv. Use case: Update quanity

ID:	**Update quanity**
Title:	**Update quanity**
Description:	- When an accountants received previously placed order. - An accoutants makes Goods Delivery Note to deliver goods to agents to check products. - An accoutants prints delivery slips to transfering and update quanity in warehousing.
Primary Actor:	Accountant
Preconditions:	When a distributor received or delivered products for agents
Postconditions:	List of products are avaiable or unvaible to tranfer to agents
Main Success Scenario:	Accountant update products after order delivered successfully or the distributor received goods.
Extensions:	Nothing
Frequency of Use	Frequently
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

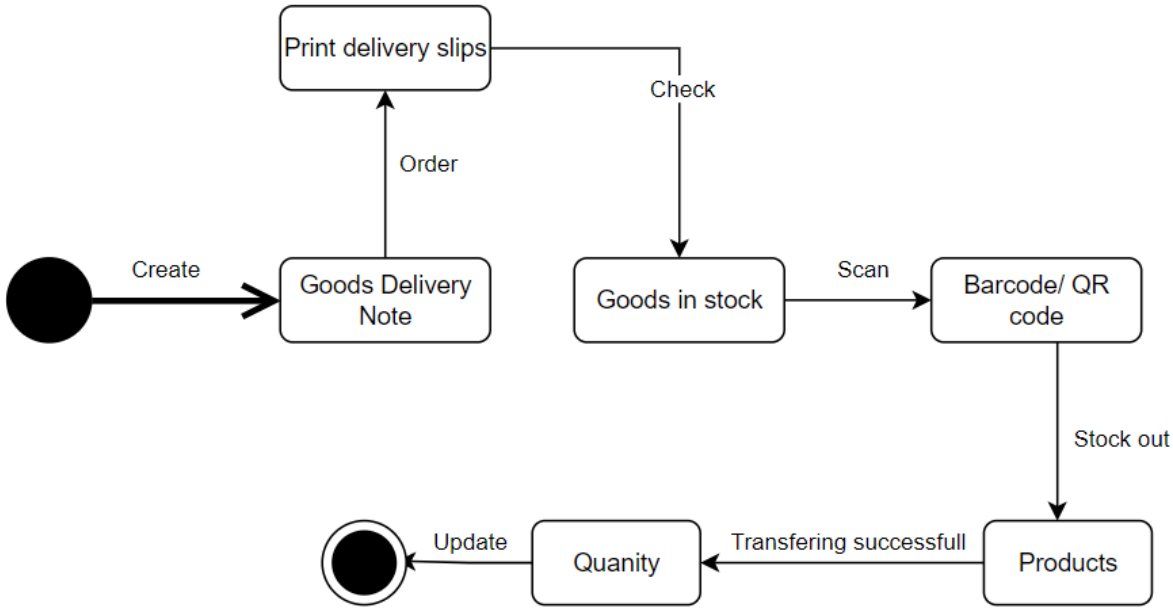
v. Use case: Scan barcode/ QR code

ID:	**Scan barcode/ QR code**
Title:	**Scan barcode/ QR code**
Description:	- When an accountants received previously placed order. - An accoutants makes Goods Delivery Note to deliver goods to agents to check products. - Warehouse staff perform to scan barcode/ QR code to stock
Primary Actor:	Accountant
Preconditions:	When the order need received and delivery, warehouse staff will scan barcode/ QR code
Postconditions:	The order was transferred to agents or import to warehousing
Main Success Scenario:	Accountant create Goods Delivery Notes or create Goods Received, warehouse staff will scan barcode/ QR code to stock out or stock in.
Extensions:	Products don't have available in stock
Frequency of Use	Frequently
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

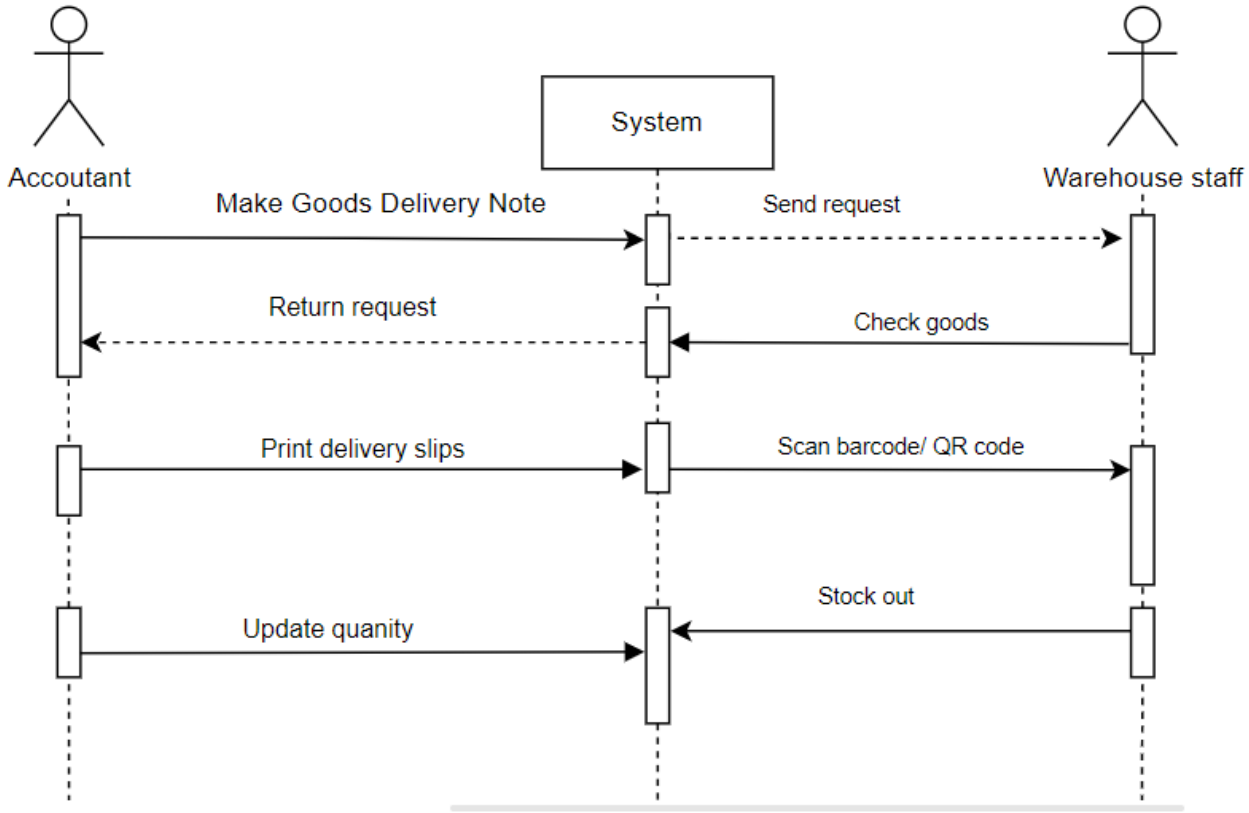
vi. Activity diagram for Use Case: Use Case Make Goods Delivery Note



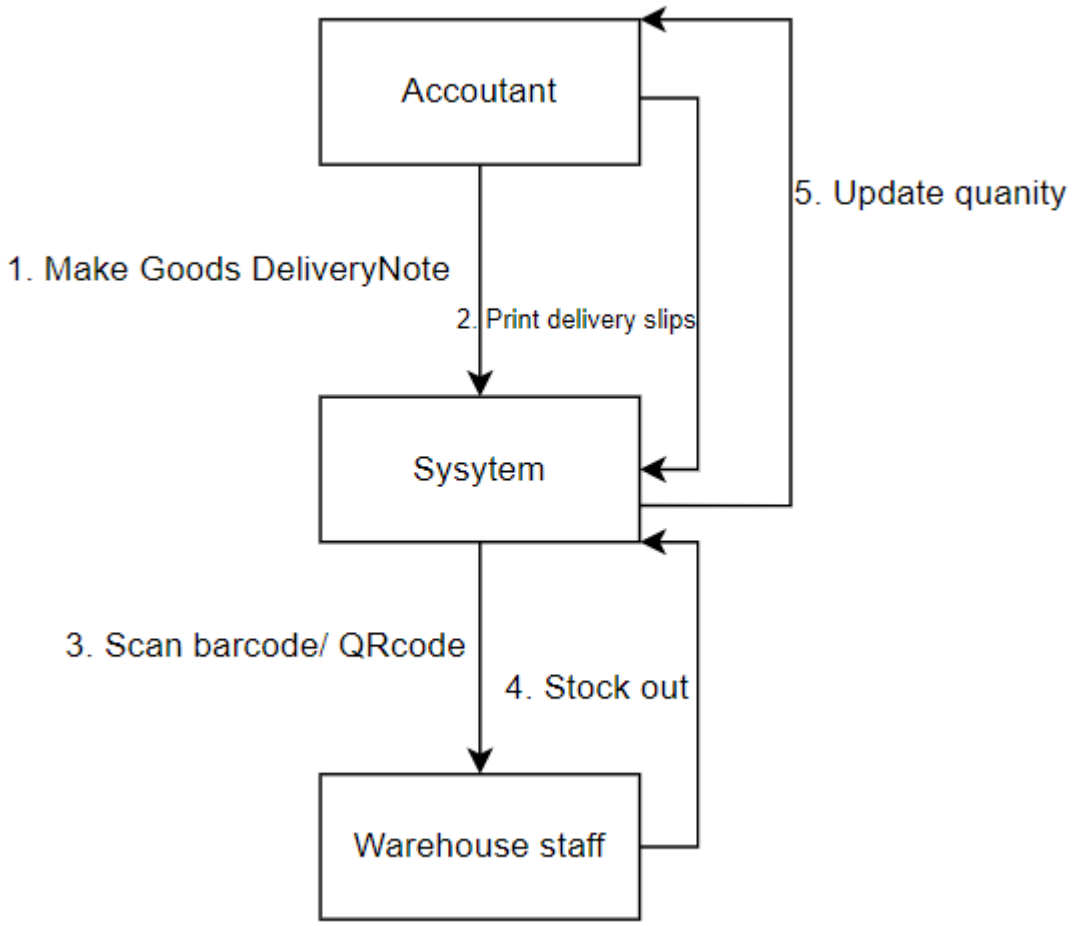
vii. State – chart for Use Case: Use case Make Goods Delivery Note



viii. System sequence diagram for Use Case: Use Case Make Goods Delivery Note



ix. Collaboration



2.2.2. *Use Case: Manage incoming/ outgoing stock*

i. Use case : Manage incoming/ outgoing stock

ID:	**Manage incoming/ outgoing stock**
Title:	**Manage incoming/ outgoing stock**
Description:	- When a distributor wants to know status of storage - Account view incoming/ outgoing stock report to inventory movement
Primary Actor:	Accountant
Preconditions:	The system is store of database and allow a accountant check status of warehouse
Postconditions:	The full information of revenue for the product when remove or add and quantity product which will be presented on the screen
Main Success Scenario:	An accountant could see information of Storage. An accountant check best selling and revenue report monthly
Extensions:	Nothing
Frequency of Use:	Not frequently, just few times every month
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

ii. Use case: Update Goods

ID:	**Update Goods**
Title:	**Update Goods**
Description:	- When an accountants manage incoming/ outgoing stock could update products - Calcultate best –selling products to received to agents
Primary Actor:	Accountant
Preconditions:	Report quanity of products
Postconditions:	Information products is best - selling Accoutant confirm the order and create Goods Delivery
Main Success Scenario:	Accountant create Goods Delivery Notes based on previously placed orders. Then print delivery slips and check products is avaiable. Next warehouse staff scan barcode/ QR to stock out
Extensions:	The system also allows warehouse staff make use of barcode/QR for stock out, couldn't check goods from accountant.
Frequency of Use	Frequently
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

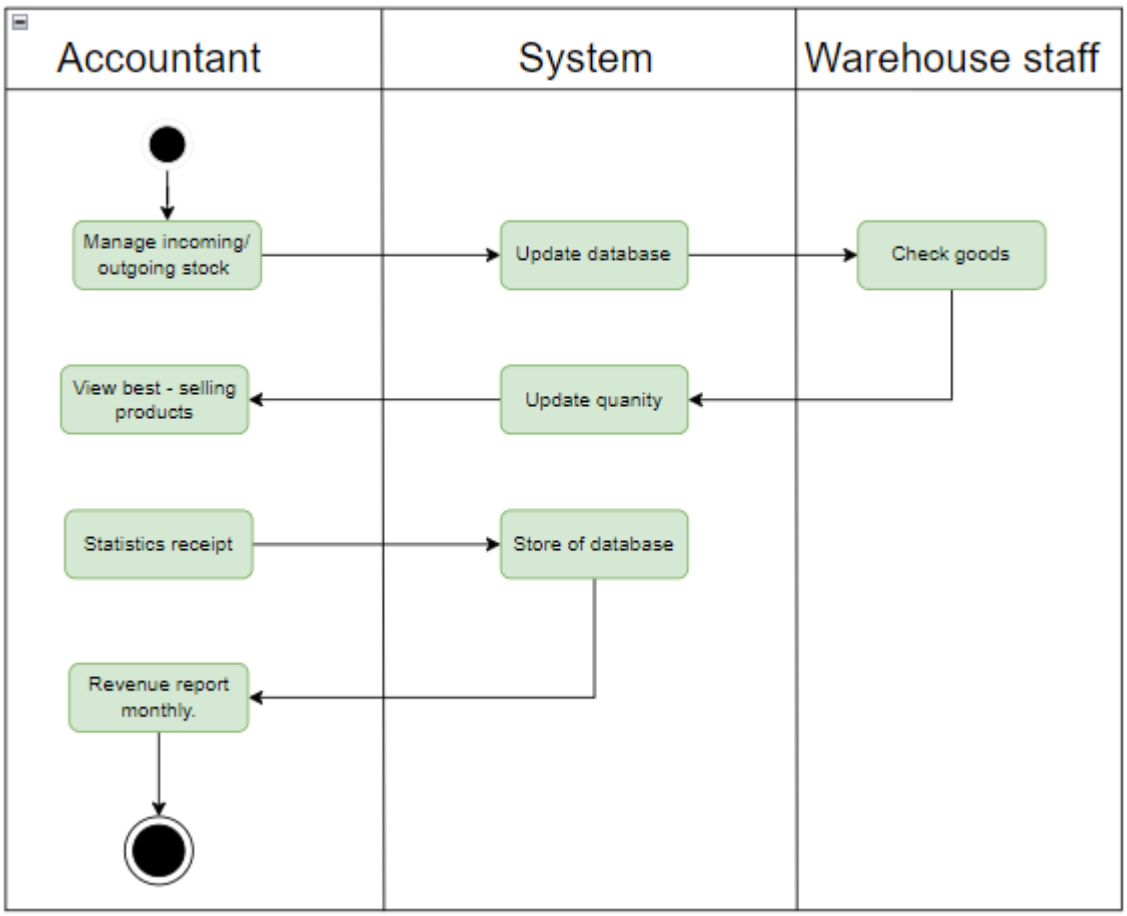
iii. Use case: Report best – selling products

ID:	**Report best-selling products**
Title:	**Report best – selling products**
Description:	- A accountant manage incoming/ outgoing stock then update goods every day - A accountant could report best – selling products every month
Primary Actor:	Accountant
Preconditions:	Accountant will update goods to report best – selling products
Postconditions:	Accountant have report best – selling products monthly
Main Success Scenario:	An accountant update incoming/ outgoing products then report best- selling goods every month.
Extensions:	Nothing
Frequency of Use:	Not frequently, just few times every month
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

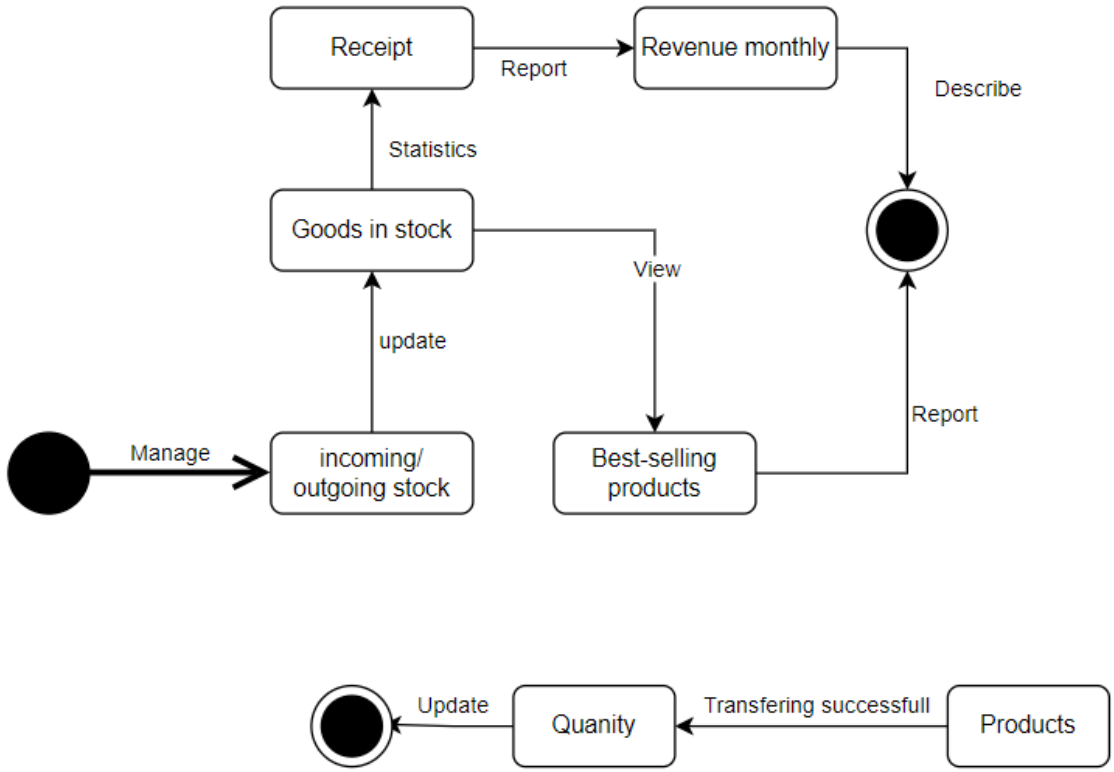
iv. Use case: Revenue report monthly

ID:	**Revenue report monthly**
Title:	**Revenue report monthly**
Description:	- A accountant manage incoming/ outgoing stock then statistics receipt - A accountant revenue report monthly for the distributors
Primary Actor:	Accountant
Preconditions:	Accountant have statistics receipt
Postconditions:	Accountant have revenue report monthly
Main Success Scenario:	An accountant statistics receipt , have revenue report monthly
Extensions:	Nothing
Frequency of Use:	Not frequently, just few times every month
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

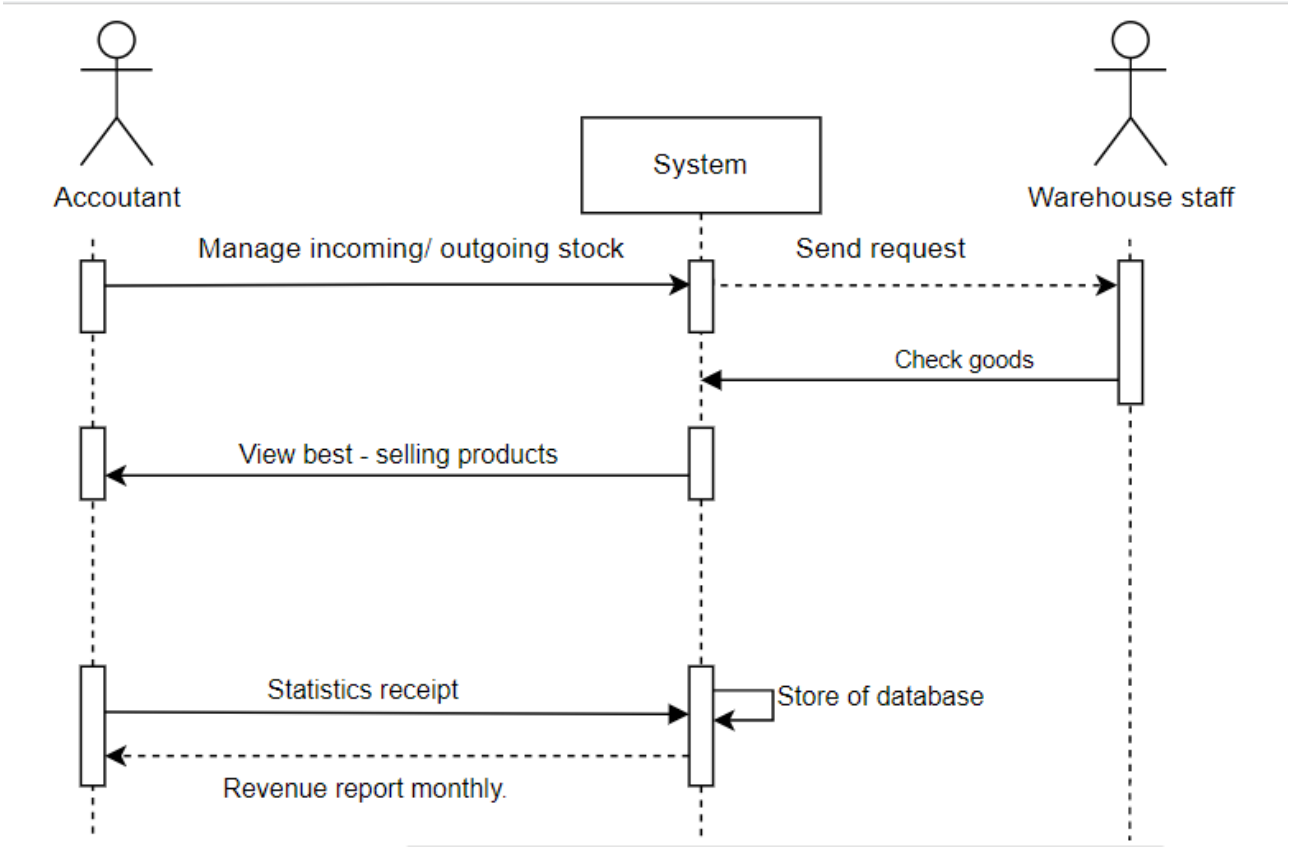
v. Activity diagram for Use Case: Manage incoming/ outgoing stock



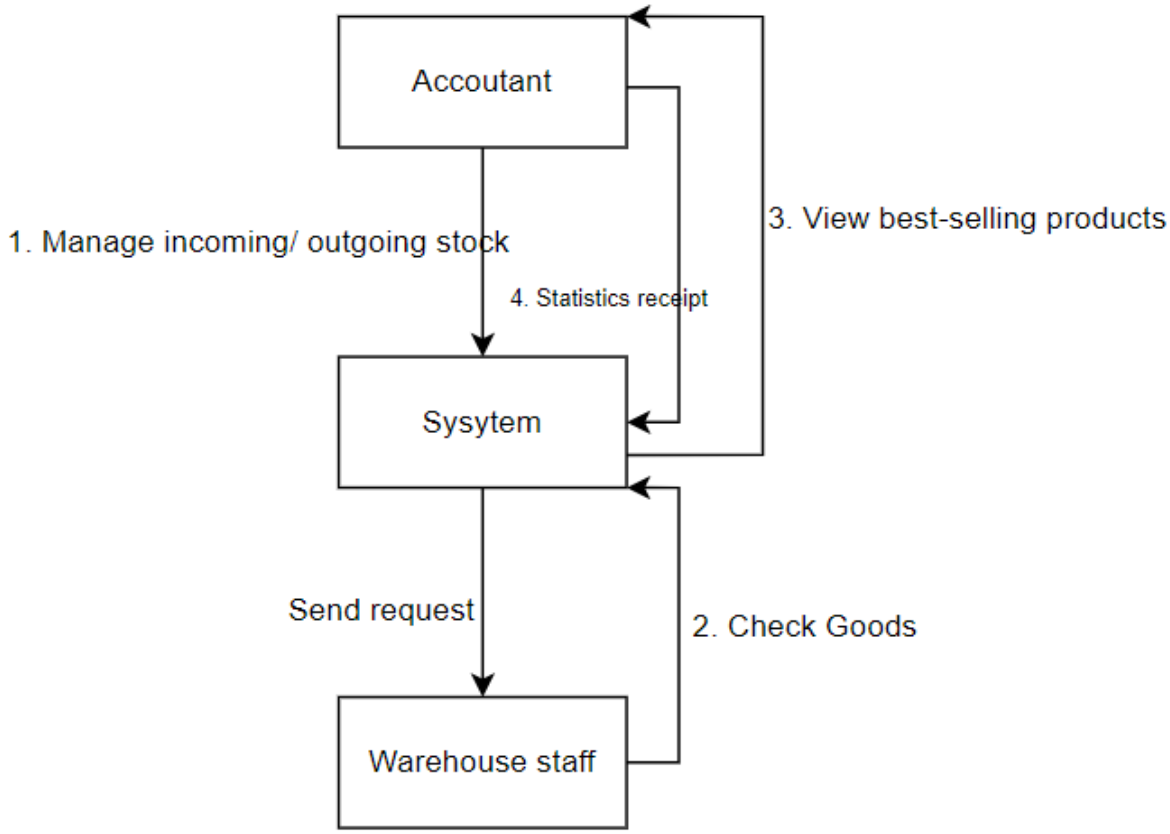
vi. State chart diagram for Use Case: Manage incoming/ outgoing stock



vii. System sequence diagram for Use Case: Manage incoming/ outgoing stock



viii. Collaboration

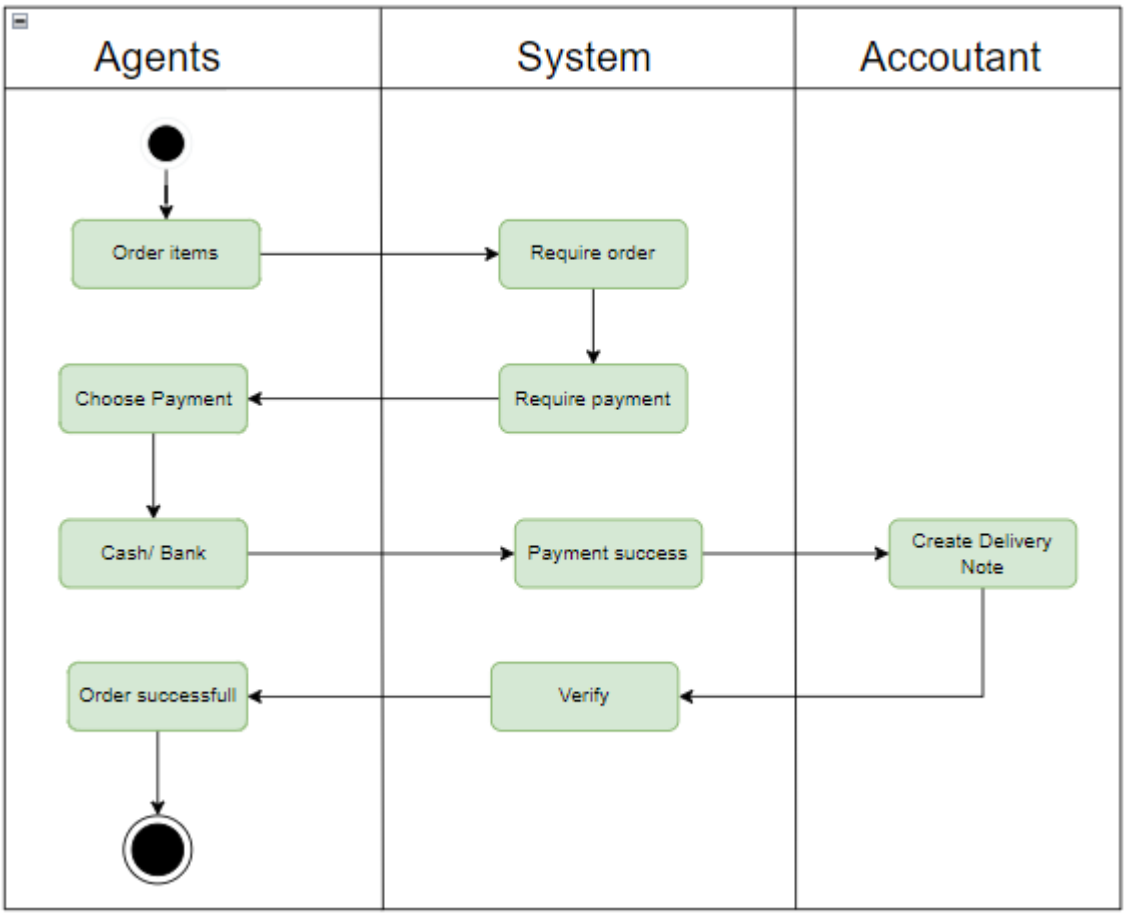


2.2.3. *Use Case: Payment*

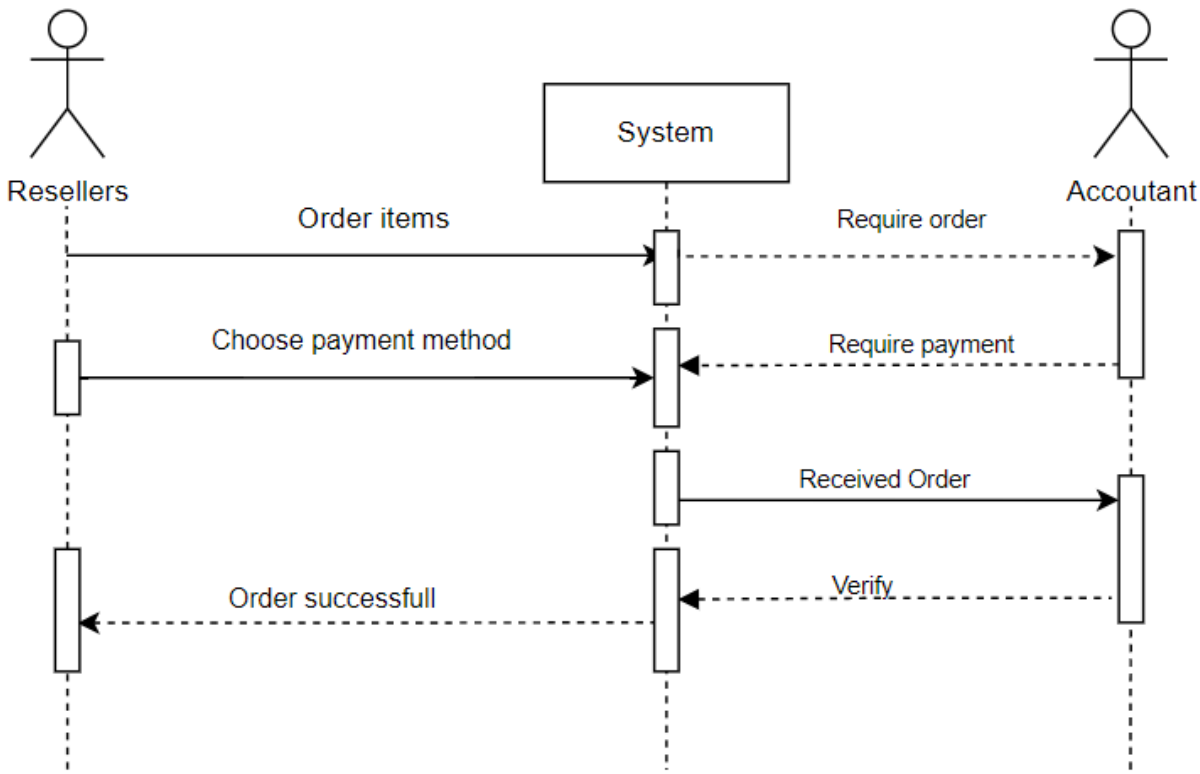
i. Use case : Payment fully description

ID:	**Payment**
Title:	**Payment**
Description:	- When Resellers/ Agents choose and order of items and to perform the payment method
Primary Actor:	Agents/ Reseller
Preconditions:	Agents order of items
Postconditions:	Agents choose payment method (cash, bank,….)
Main Success Scenario:	Reseller/ Agents order products and choose payment method in system.
Extensions:	Nothing
Frequency of Use:	Not frequently, just few times every week/ month
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

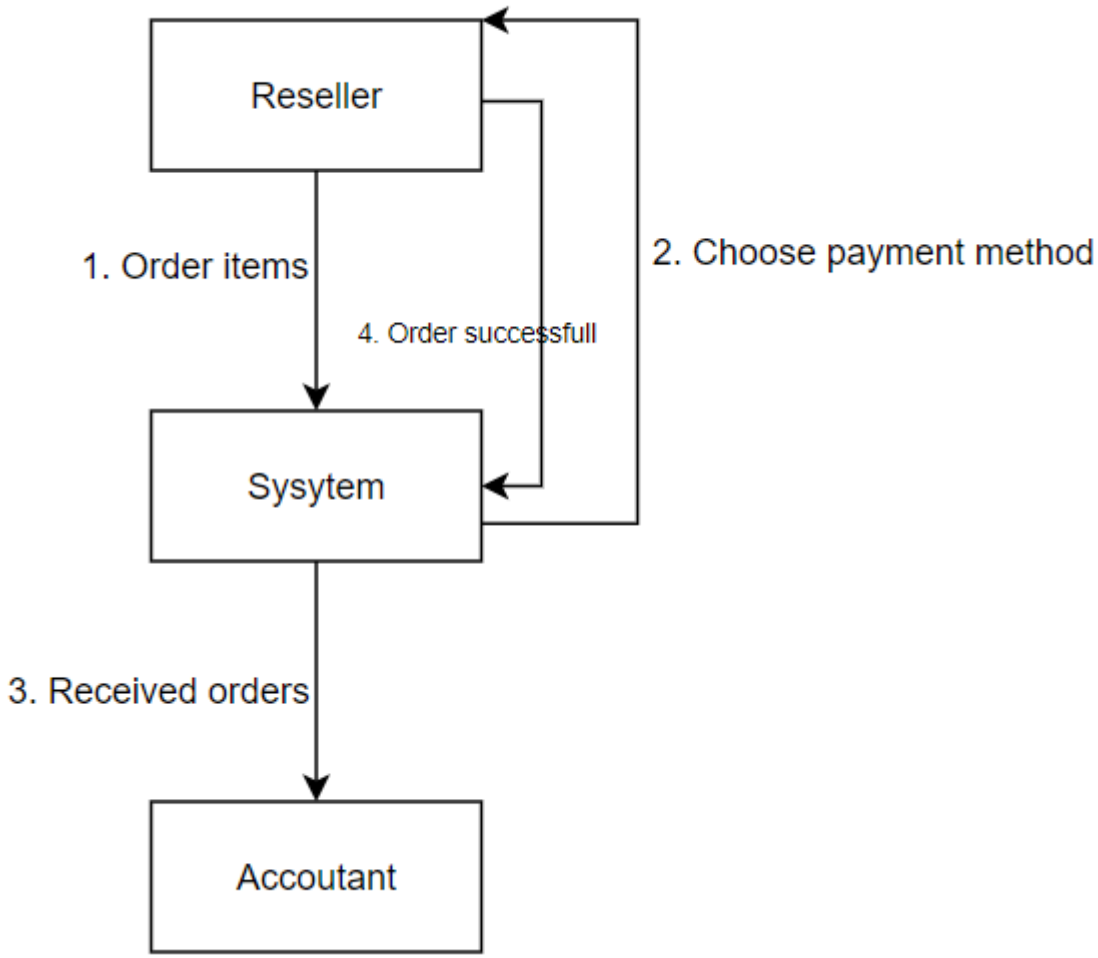
ii. Activity diagram for Use Case: Payment



iii. System sequence diagram for Use Case: Payment



iv. Collaboration: Payment

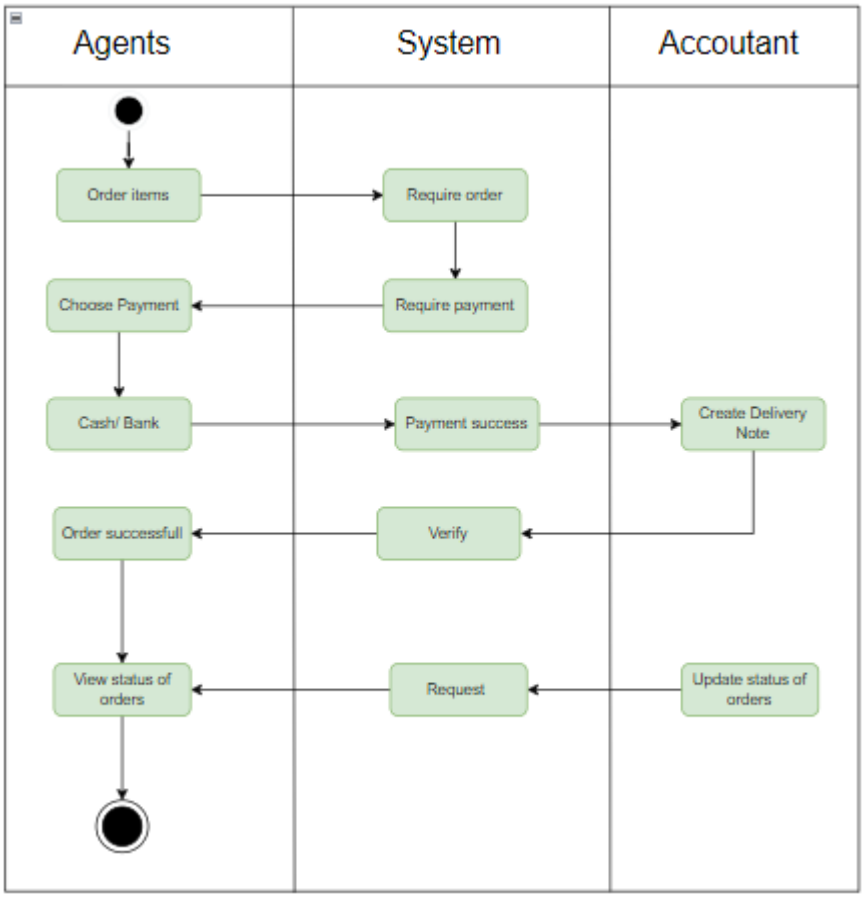


2.2.4. *Use Case: View status of orders*

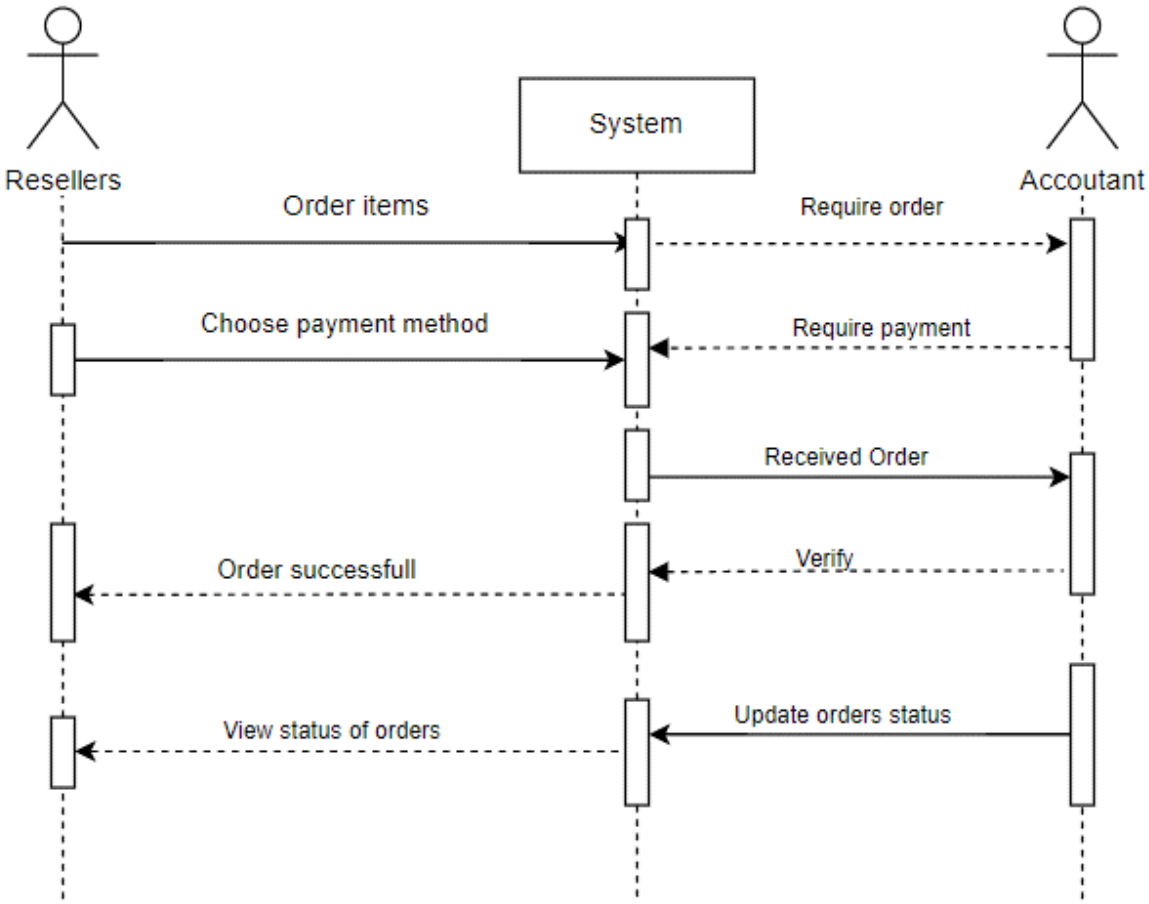
i. Use case : name 4 fully description

ID:	**View status of orders**
Title:	**View status of orders**
Description:	- When Resellers/ Agents order of items succesfully, accoutant update order status to agents could allow view status
Primary Actor:	Agents/ Reseller
Preconditions:	Agents/ Reseller order of product successfull
Postconditions:	Account update order status and agents could view status of orders
Main Success Scenario:	Resellers/ Agents order of items succesfully, accoutant update order status to agents could allow view status
Extensions:	Nothing
Frequency of Use:	Not frequently, just few times every week/ month
Status:	[Development status]
Owner:	Le Huynh My Duyen
Priority:	[Priority of this use case]

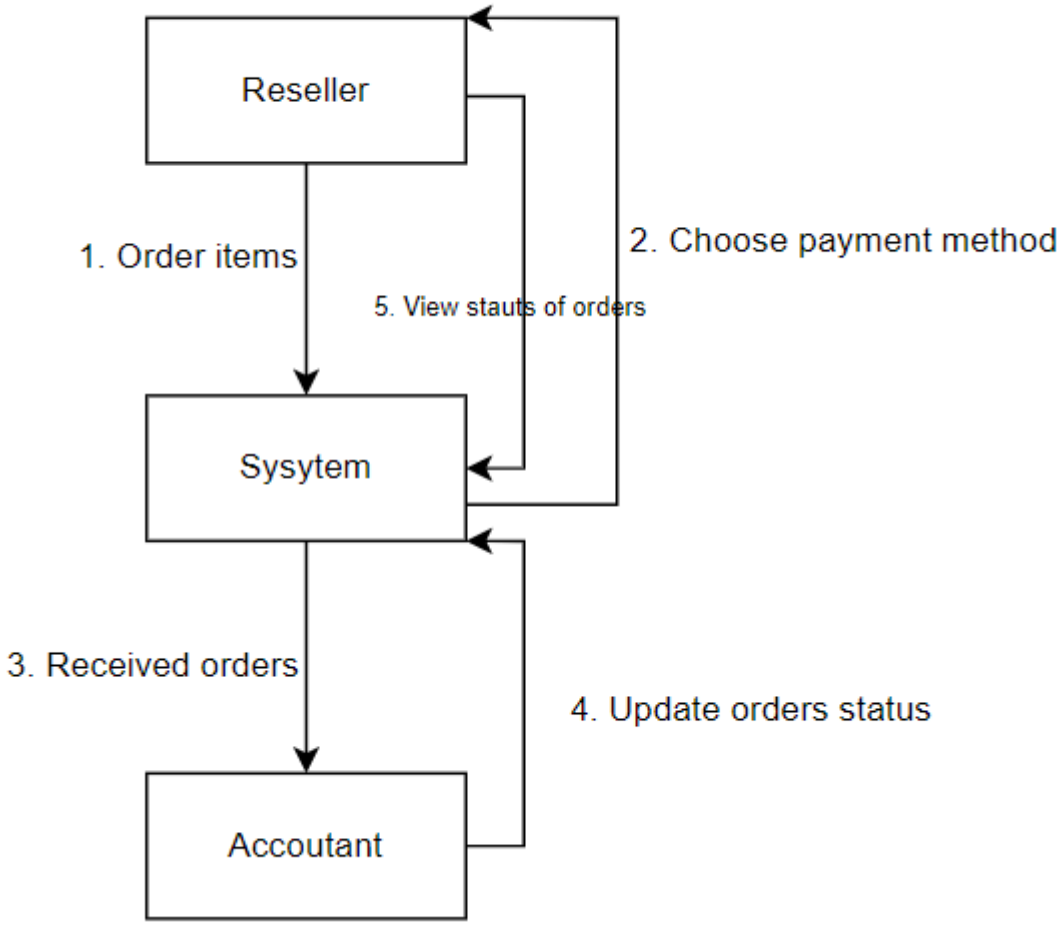
ii. Activity diagram for Use Case: View status of orders



iii. System sequence diagram for Use Case: View status of orders



iv. Collaboration of View status of Orders



2.3 Verifying use cases for Actor

2.3.1. Verifying uses cases: Accoutant

Data entity/domain class	C R U D	Verified use case
Accoutant	Create	Create Goods Received Make Goods Delivery Note
	Update	Update orders status
	Log in	Log in system

2.3.2. Verifying uses cases for Warehouse staff

Data entity/domain class	C R U D	Verified use case
Warehouse staff	Scan	Scan barcode/ QR code, RIFDs
	Log in	Log in system
	Update	Update Goods
	Check	Check Goods

2.3.3. Verifying uses cases for Agents

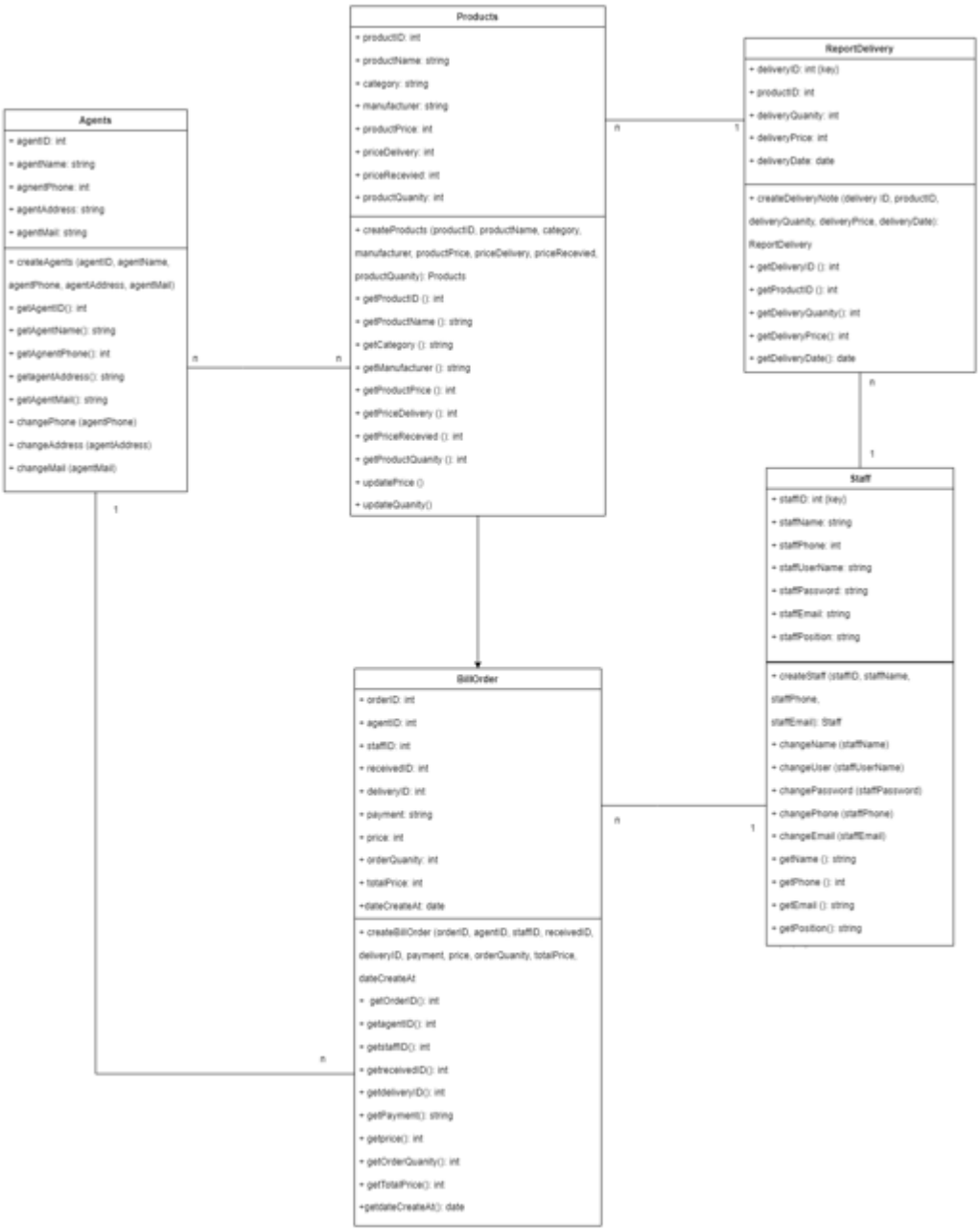
Data entity/domain class	C R U D	Verified use case
Agents	Order	Order of items
	Make	Make Payment
	View	View status of orders

III. System Requirements Design (3.0 points)

3.1 Design Class for Create Goods Delivery Note

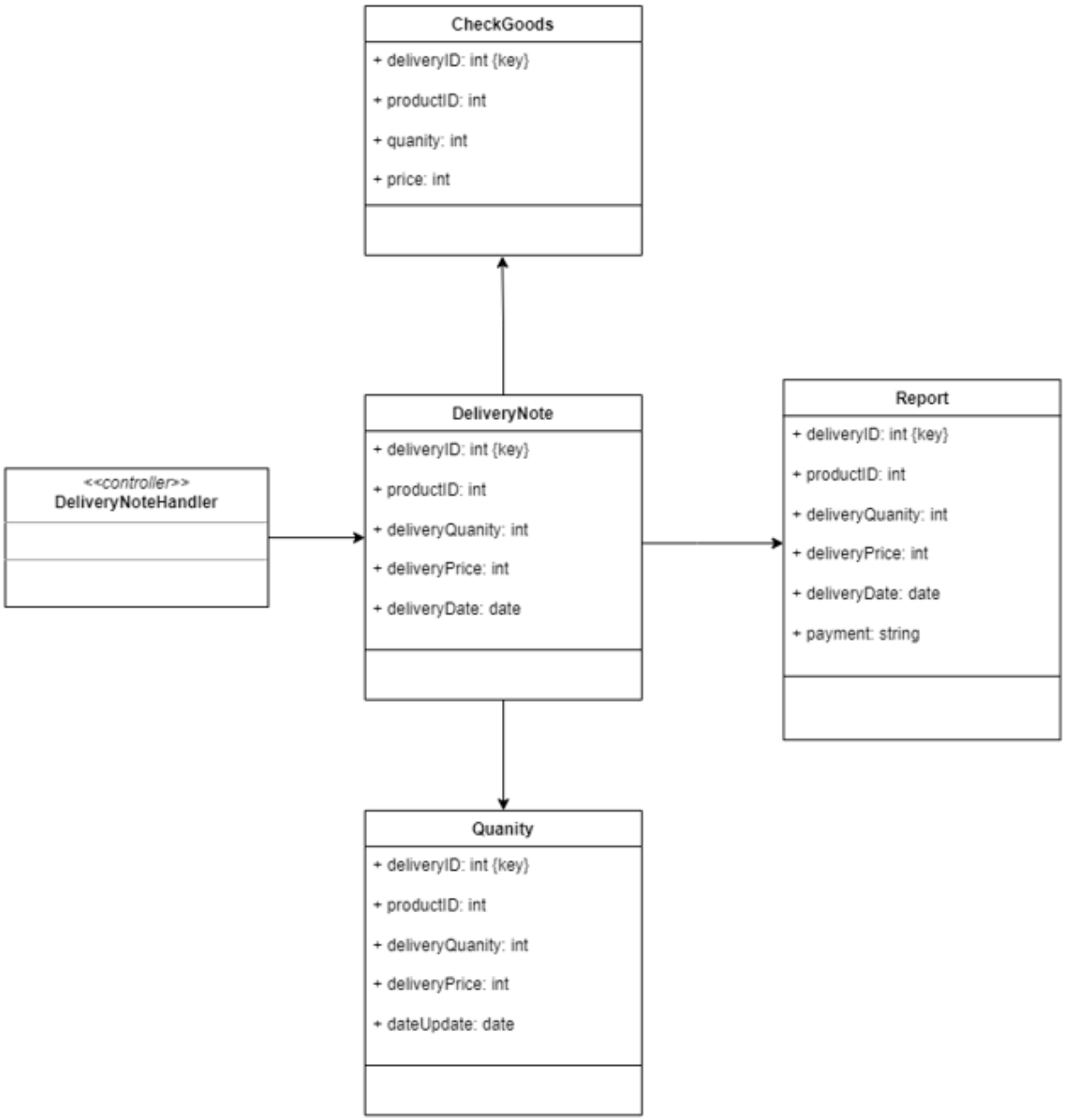
3.2.1. Design Classes in Detailed Design

Convert Domain Classes to Design Classes

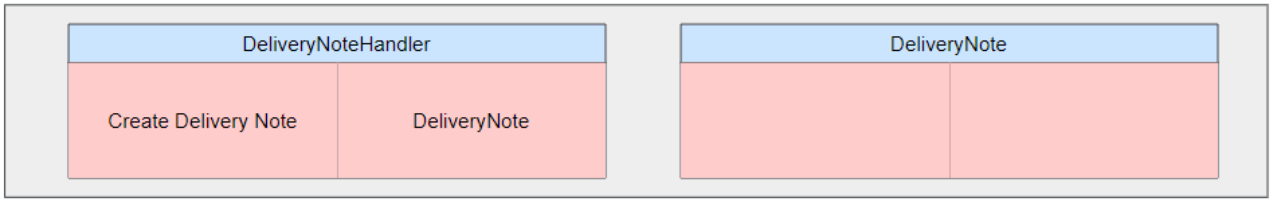


3.2.2. Design Class Diagram

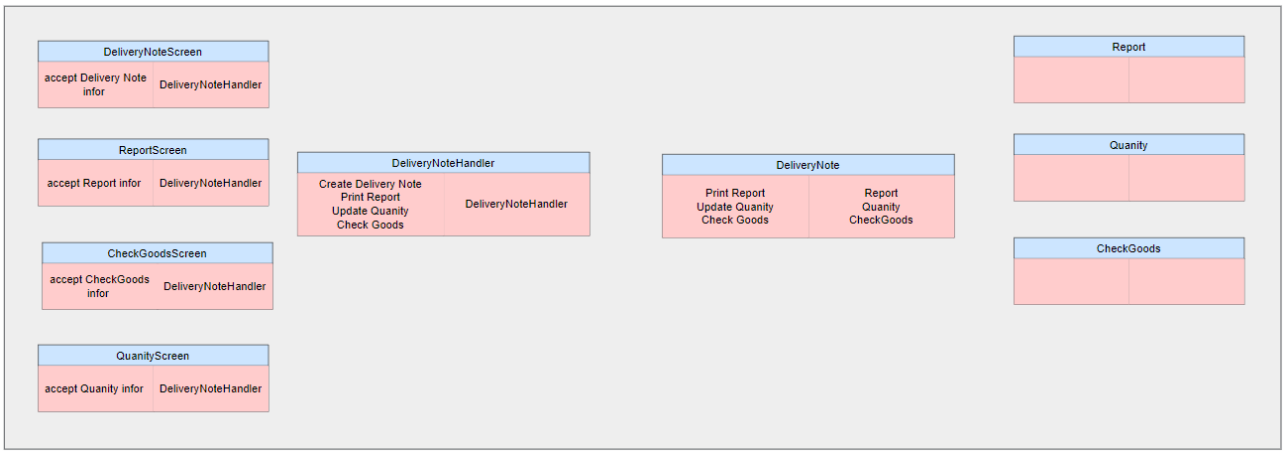
i. Domain Design Class



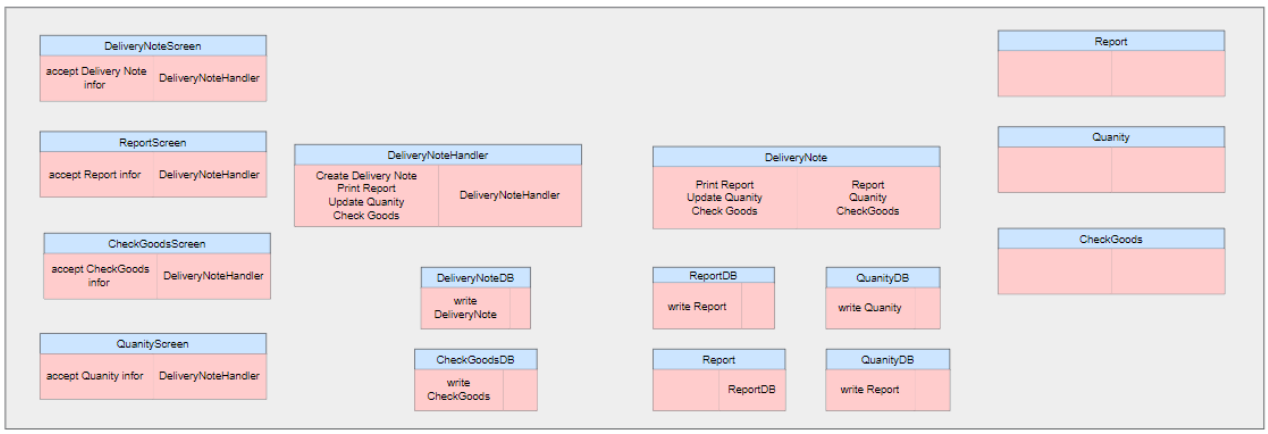
ii. Controller



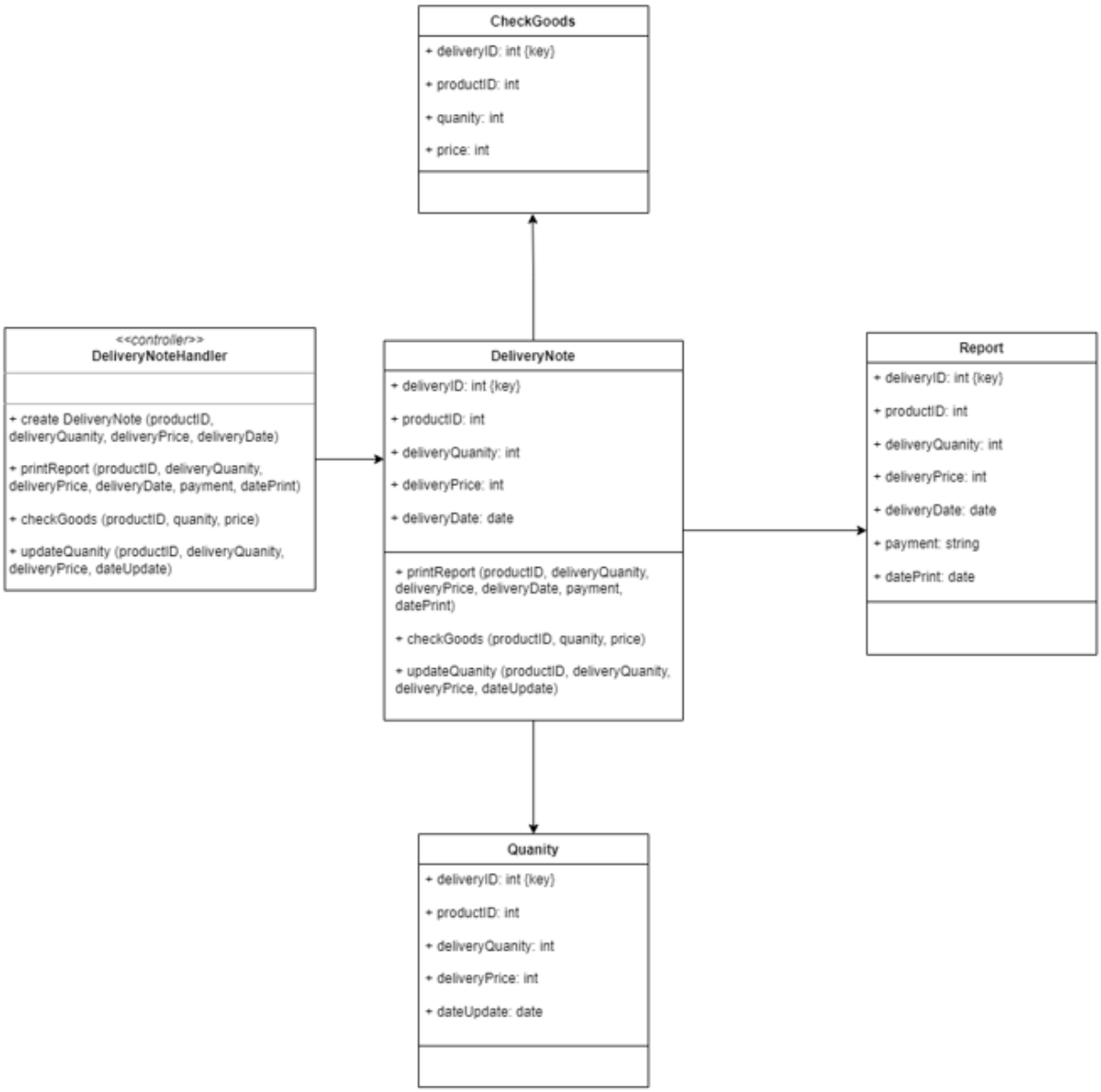
iii. UI



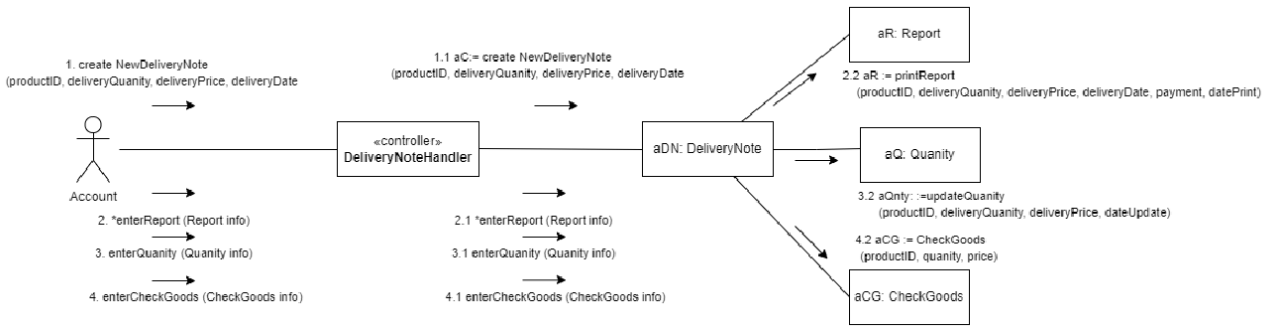
iv. Data Access



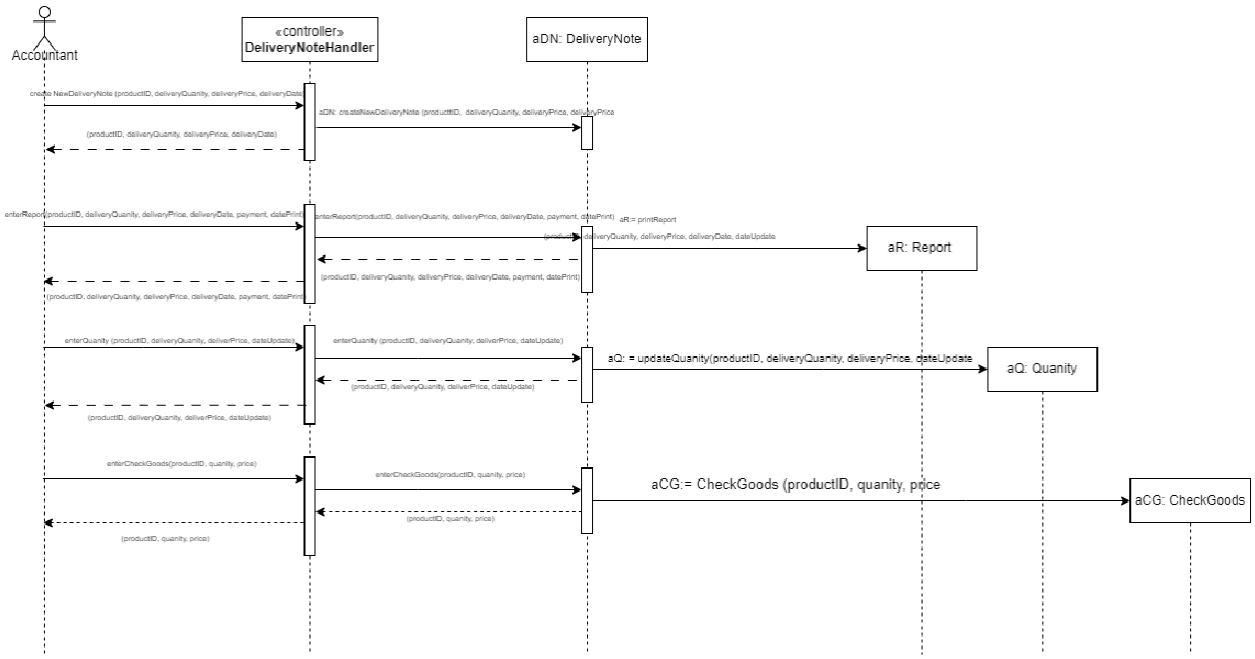
v. Design Class



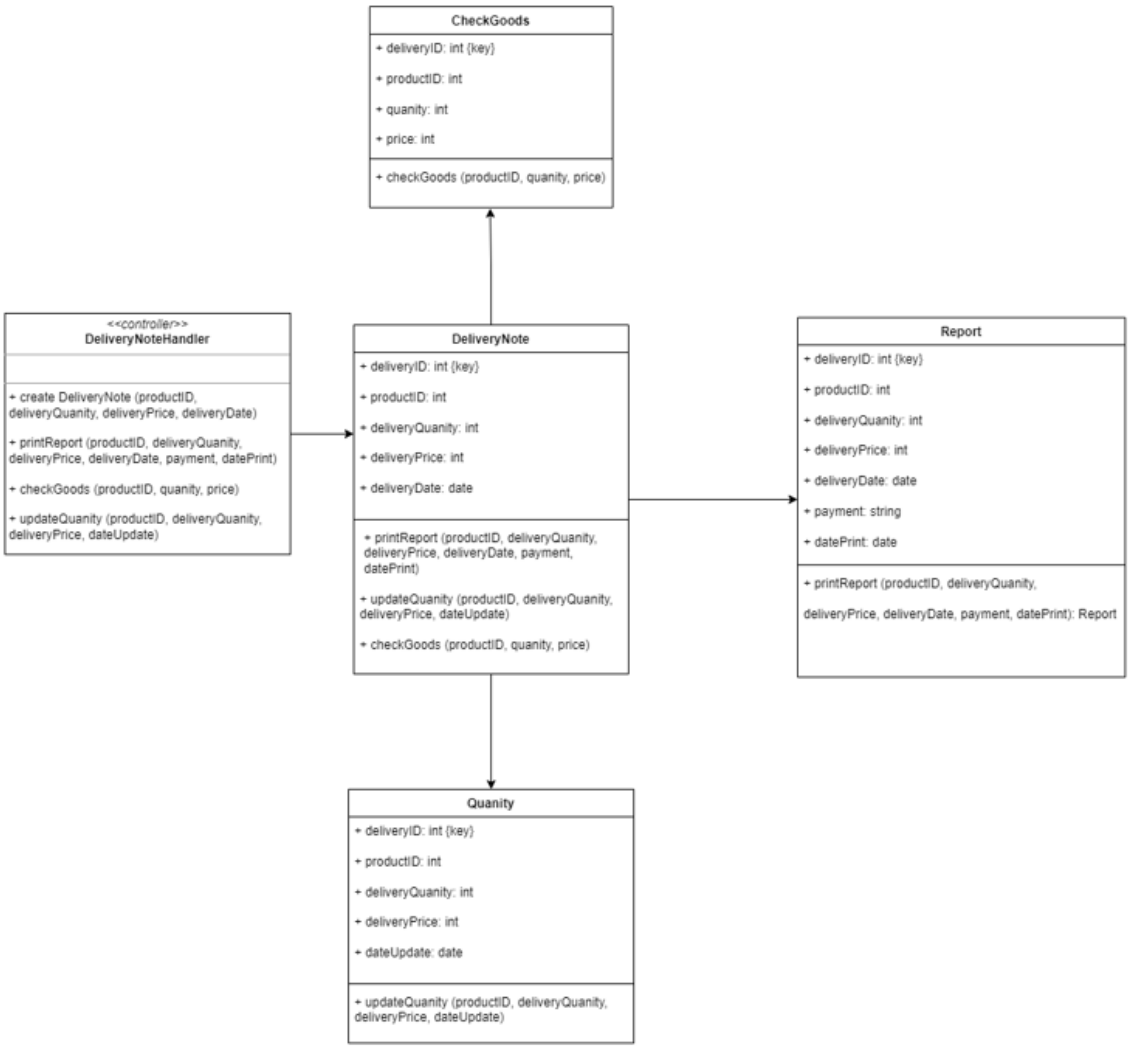
3.2.3. OOD with Communication



3.2.4. OOD with Sequence Diagram



3.2.5. Final Design Class Diagram

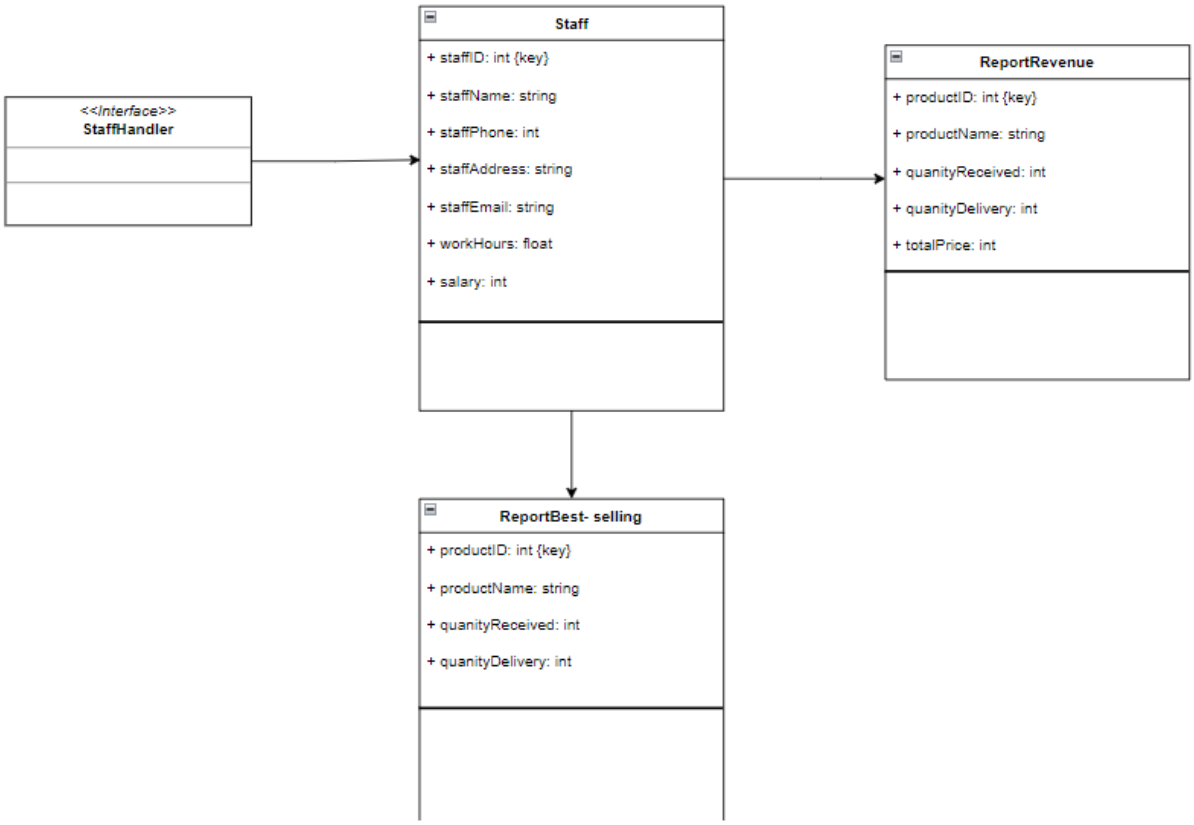


3.2 Design Class for Manage incoming/ outgoing stock

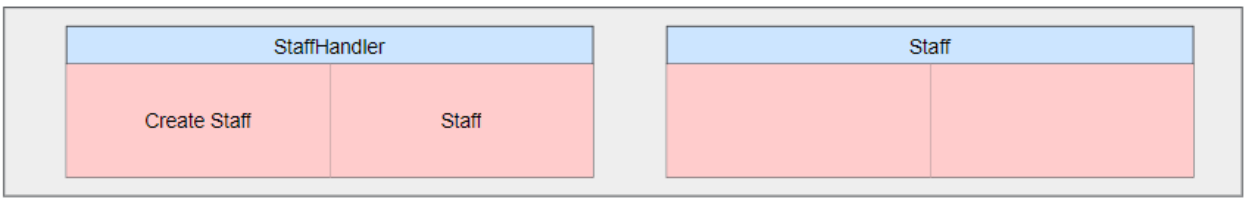
3.2.1. Design Classes in Detailed Design

3.2.2. Design Class Diagram

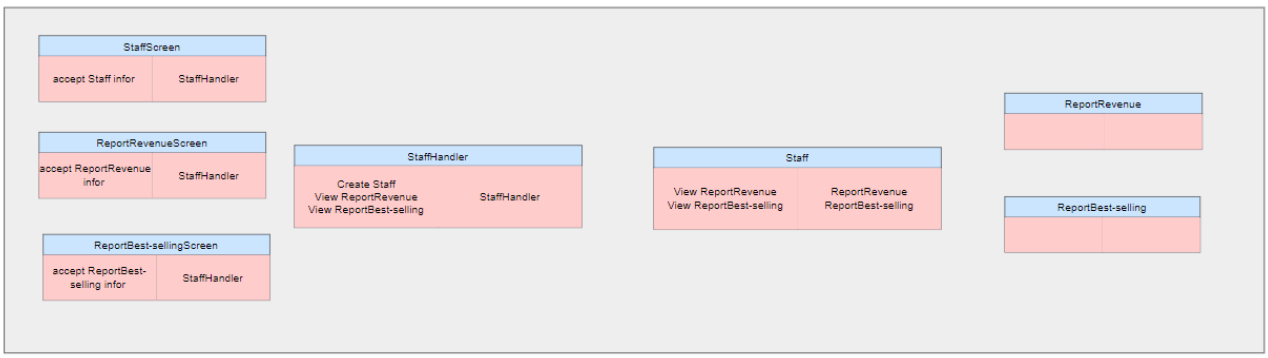
i. Domain Design Class



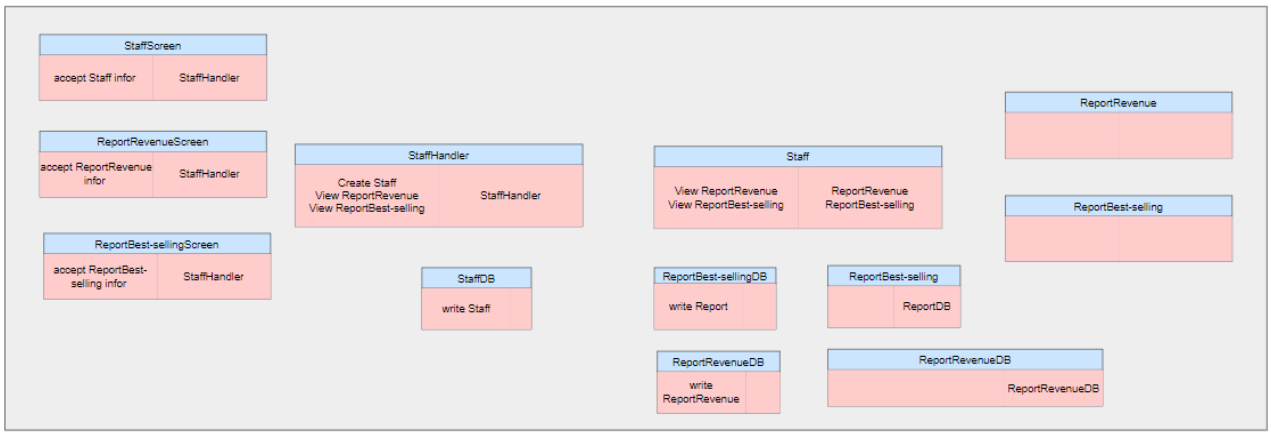
ii. Controller



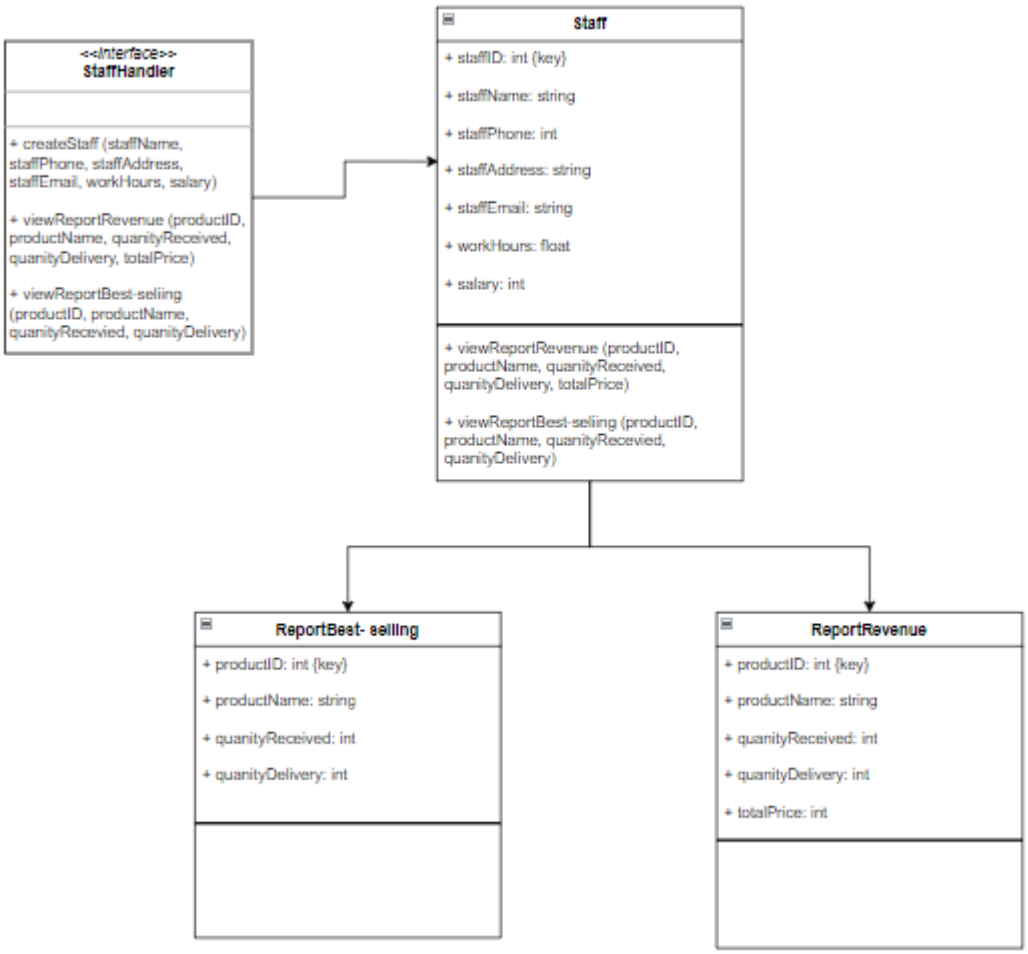
iii. UI



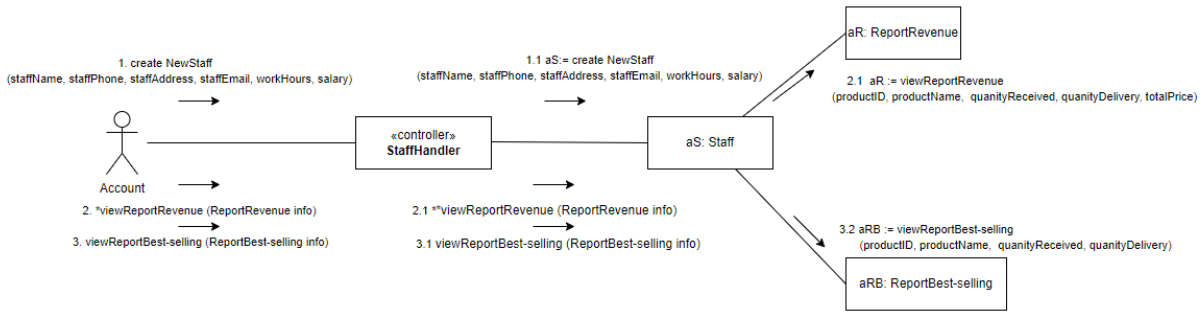
iv. Data Access



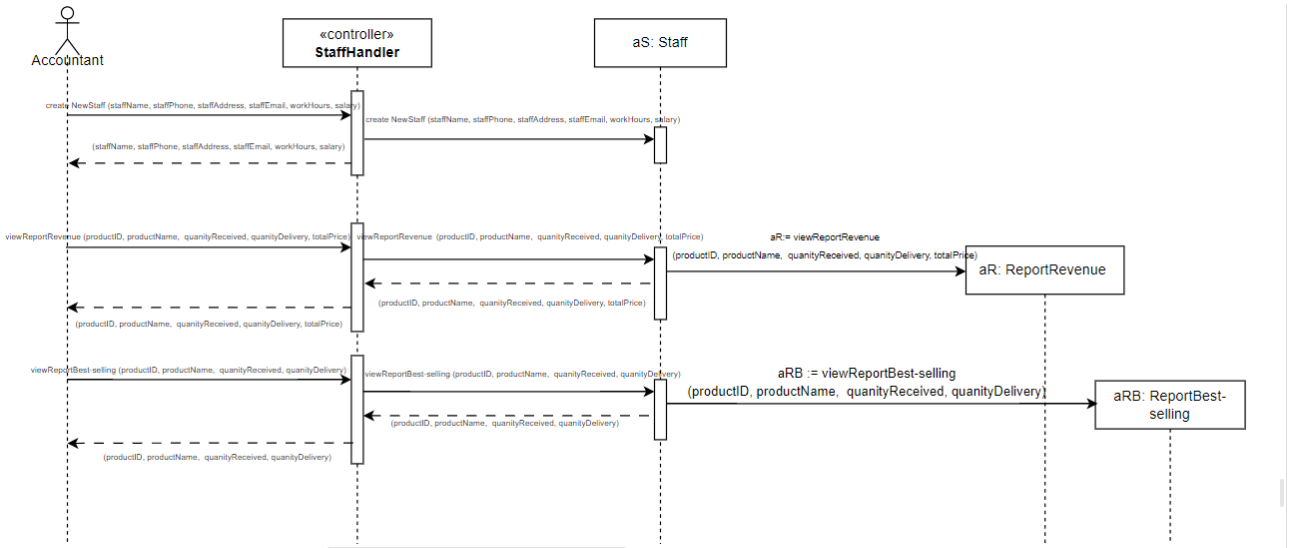
v. Design Class



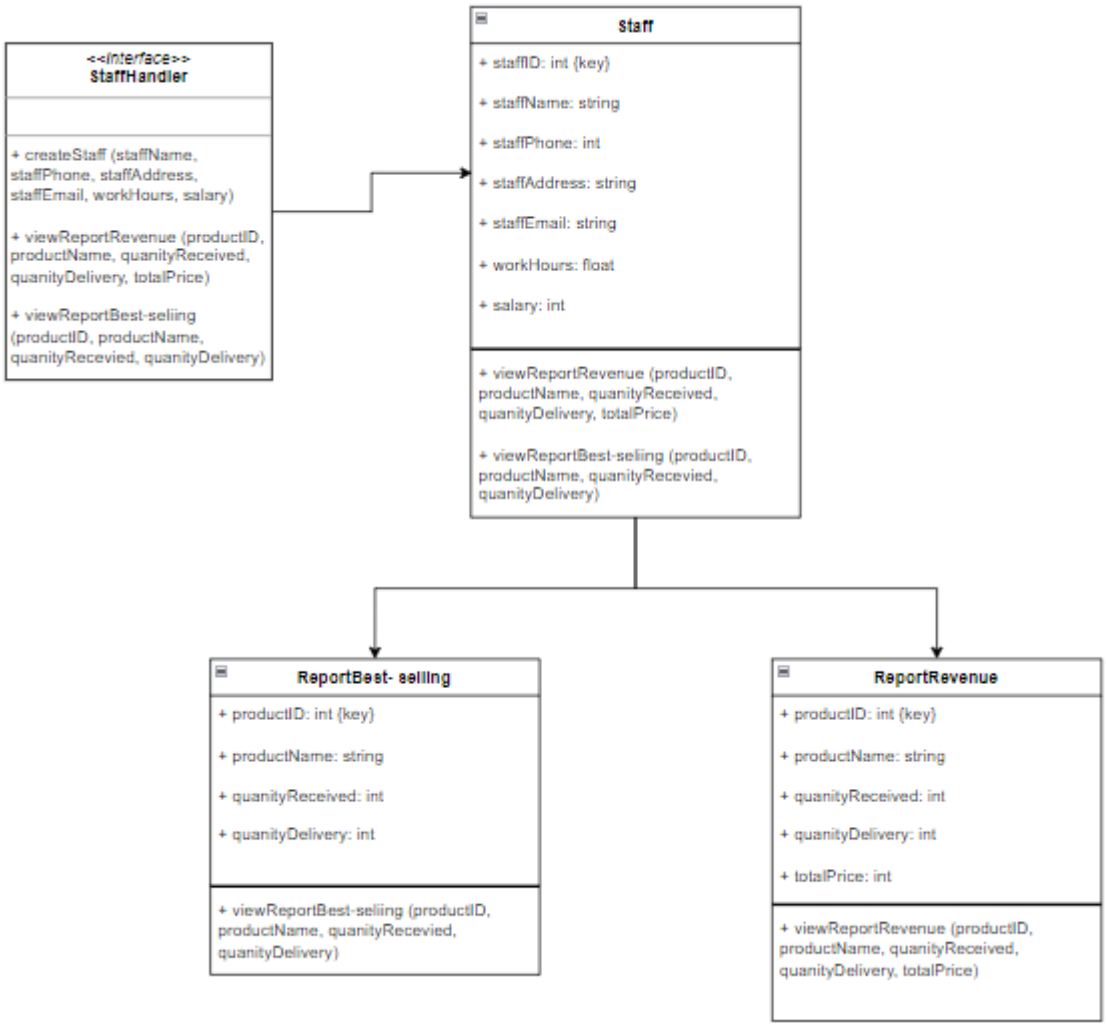
3.2.3. OOD with Communication



3.2.4. OOD with Sequence Diagram

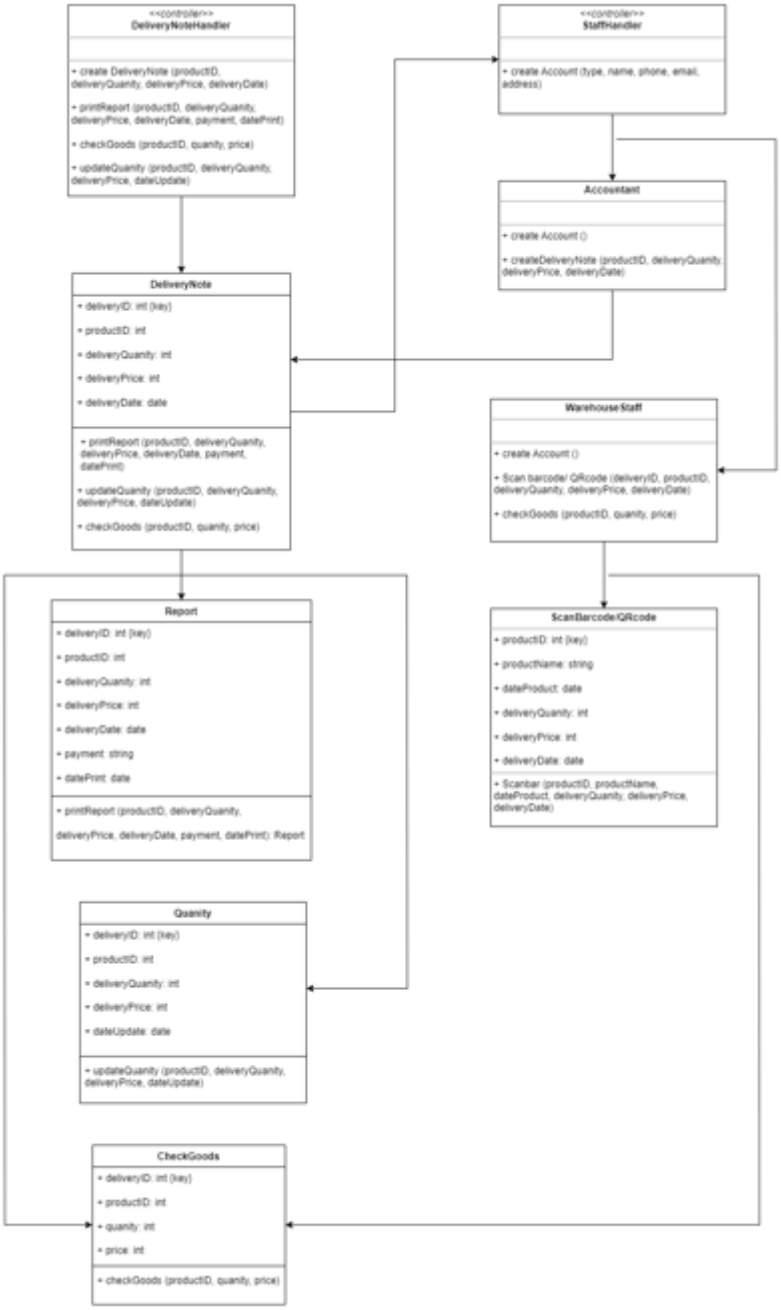


3.2.5. Final Design Class Diagram



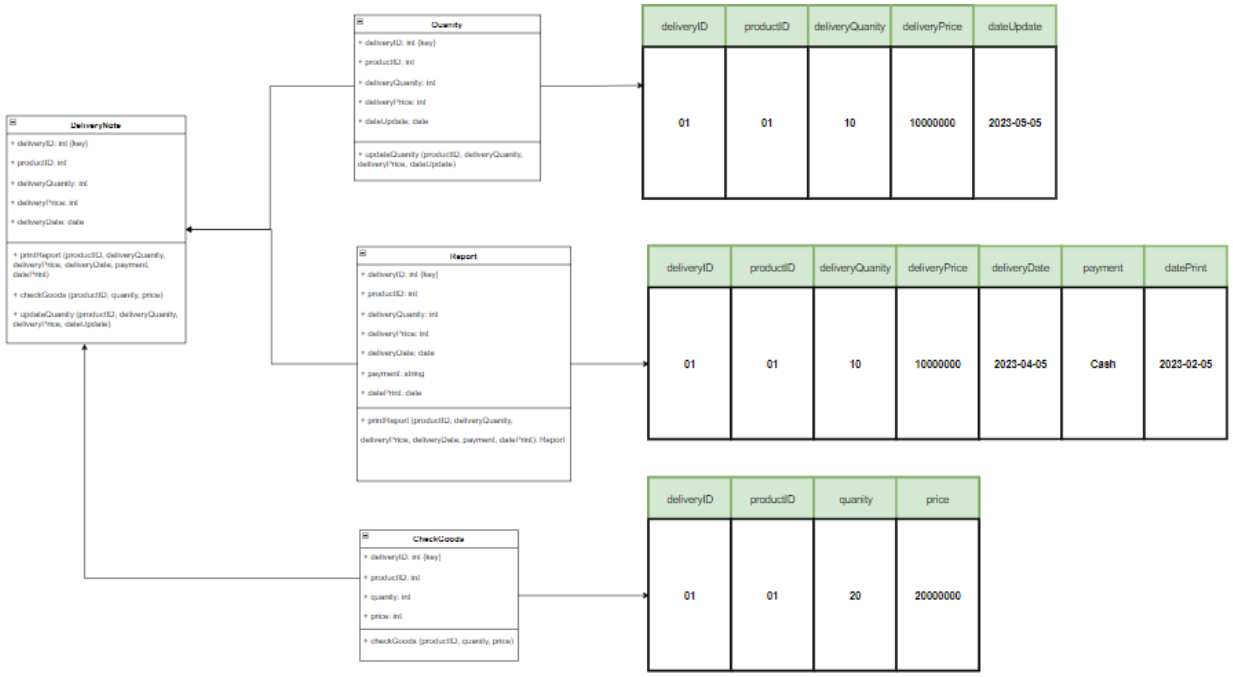
IV. System Requirements Implementation (2pts)

4.1 Design Class for Sub System



4.2 Implementation

4.2.1 Map persistent objects to the tables in a database



4.2.2 UI design

Figure: Log in Account

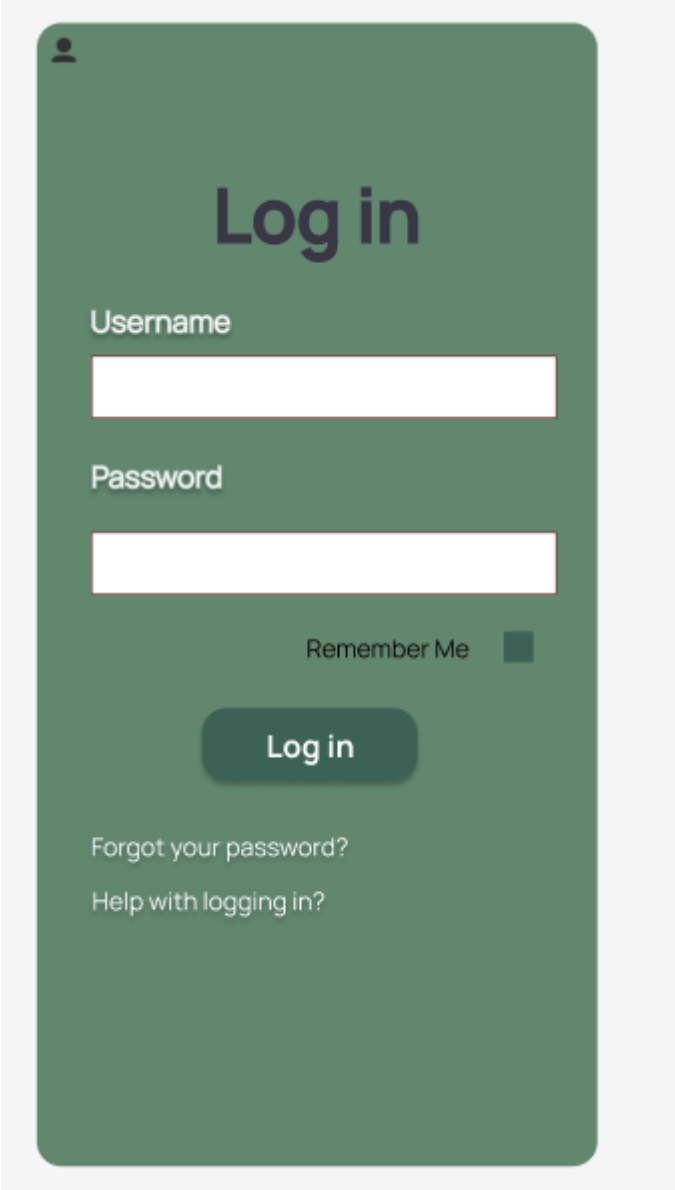


Figure: Screen Login

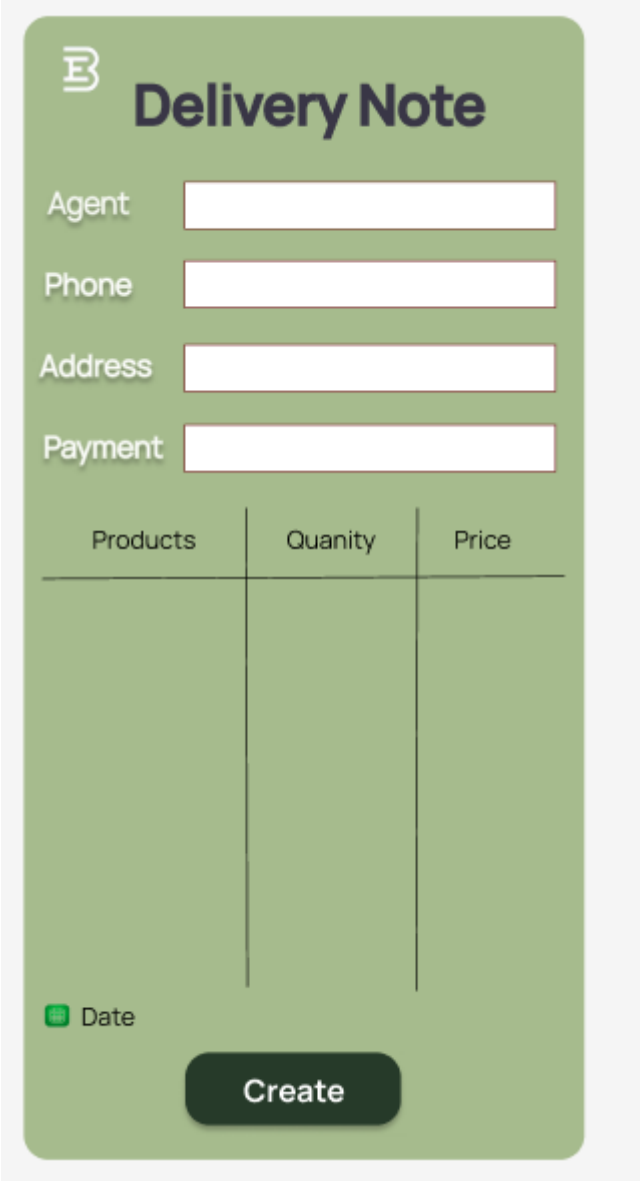


Figure: Screen Create Delivery Note

Figure: Print Receipt

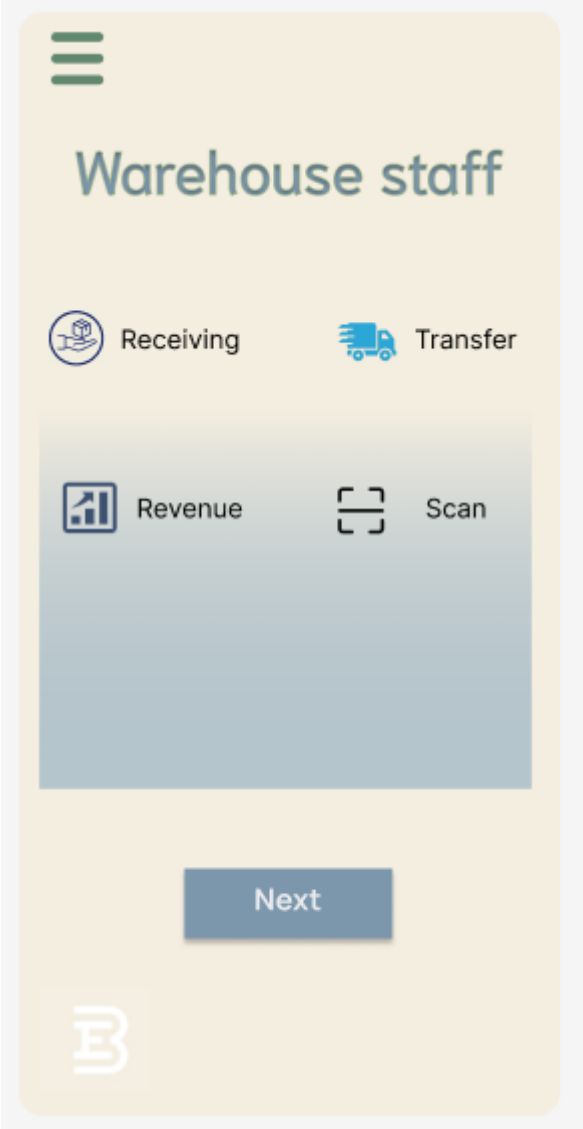


Figure: Screen Warehouse staff

Figure: Scan barcode/ QR code/ RIFDs

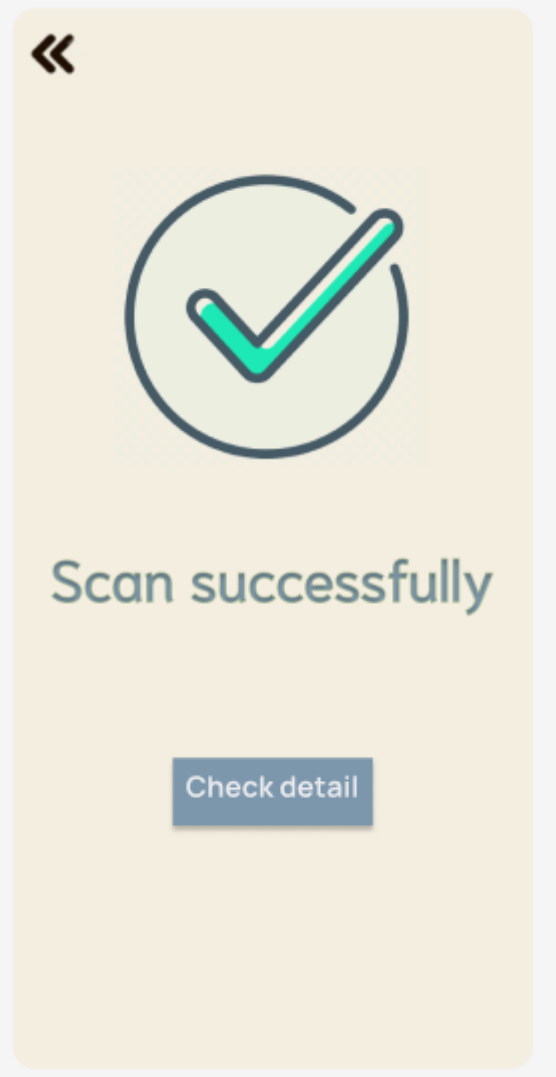
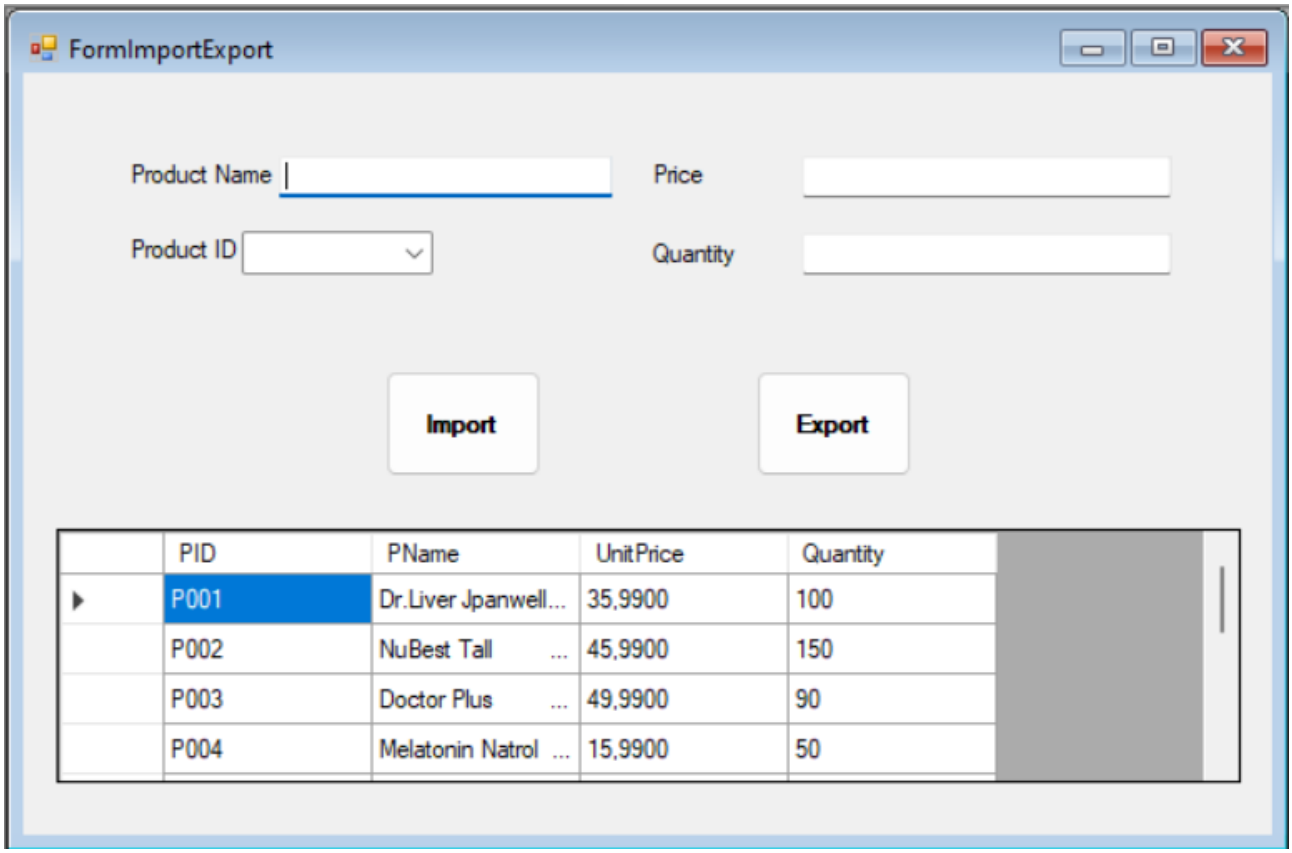
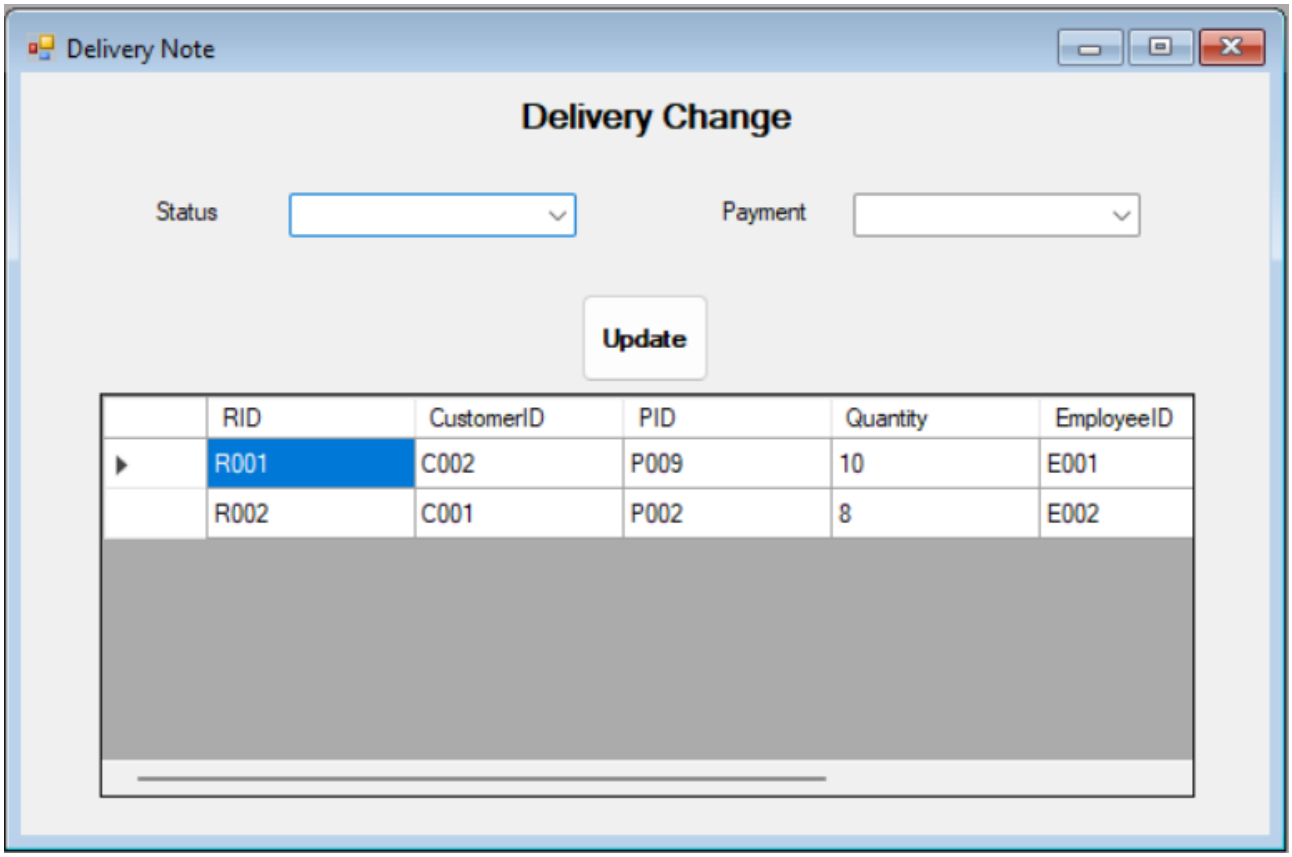


Figure: Scan successfully

4.2.3 Winform


FormImportExport
Product Name
Price
Product ID
Quantity
Import
Export
PID
PName
UnitPrice
Quantity
P001
Dr.Liver Jpanwell...
35,9900
100
P002
NuBest Tall ...
45,9900
150
P003
Doctor Plus ...
49,9900
90
P004
Melatonin Natrol ...
15,9900
50



4.3 SQL Code

i. Create Database

```
create database ManageElectricProducts
```

ii. Create Table

```
create database ManageElectricProducts

create table GoodReceived(
    IDReceived varchar(10) primary key,
    QuantityReceived int,
    PriceReceived int,
    DateReceived date
)

create table GoodDelivery(
    IDDelivery varchar(10) primary key,
    QuantityDelivery int,
    PriceDelivery int,
    DateDelivery date
)

create table Agents(
    IDAgent varchar(10) primary key,
    Agent_Name nvarchar(100),
    PhoneAgents varchar(15),
    AddressAgents nvarchar(100),
    MailAgents nvarchar(100)
)
```

```
create table Staff(
    IDStaff varchar(10) primary key,
    FullName nvarchar(100),
    Phone varchar(15),
    Username nvarchar(100),
    Password nvarchar(100),
    Position nvarchar(100)
)

create table OrderReport(
    IDOrder varchar(10) primary key,
    IDAgent varchar(10) foreign key references Agents(IDAgent),
    IDStaff varchar(10) foreign key references Staff(IDStaff),
    IDReceived varchar(10),
    IDDelivery varchar(10),
    Payment nvarchar(100),
    Price int,
    QuantityOrder int,
    TotalPrice int,
    DateCreateAt date,
)

create table Products(
    ProductID varchar(10) primary key,
    NamePro nvarchar(100),
    Category nvarchar(100),
    Manafacturer nvarchar(100),
    Price int,
    PriceDelivery int,
    PriceReceived int,
    Quantity int,
    IDReceived varchar(10) foreign key references GoodReceived(IDReceived),
    IDDelivery varchar(10) foreign key references GoodDelivery(IDDelivery),
    IDOrder varchar(10) foreign key references OrderReport(IDOrder)
)

create table Agents_Products(
    IDAgent varchar(10) foreign key references Agents(IDAgent),
    ProductID varchar(10) foreign key references Products(ProductID),
    primary key(IDAgent,ProductID)
)
```

```
insert into Agents values ('A01', N'Phong Vu', '18006867', N'Nguyen Thi Thap, Tan Phong, Q7', N'cskh@phongvu.vn'),
                          ('A02', N'Cell Phone S', ' 18002097', N'Dien Bien Phu, Binh Thanh', N'cskh@cellphones.vn'),
                          ('A03', N'GearVn', ' 18006975', N'78-80-82 Hoàng Hoa Thám, Phường 12, Quận Tân Bình', N'CSKH@GEARVN.COM')
```

```
insert into Staff values ('ST01',N'Staff 1', '0678345912', N'ST01','ST01123', N'Accountant'),
                         ('ST02',N'Staff 2', '0678321954', N'ST02','ST02123', N'Warehouser'),
                         ('ST03',N'Staff 3', '0678321945',N'ST03','ST03123', N'Warehouser')
```

```
insert into GoodReceived values (1, 3, 3600000, '2023-04-30'),
                                 (2, 5, 3500000, '2023-05-01'),
                                 (3, 6, 7800000, '2023-05-02')
```

```
insert into GoodDelivery values (1, 5, 7500000, '2023-05-06'),
                                 (2, 5, 4500000, '2023-05-08'),
                                 (3, 5, 7500000, '2023-05-09')
```

```
insert into Products values (1, 'Keyboard', 'Laptop Component', N'Edra' , 1200000,  1200000, 1500000, 12, 1, 1, 1),
                            (2, 'Mouse', 'Accessories',  N'PCFTECH',700000,  700000, 900000, 10, 2, 2, 2),
                            (3, 'Monitor', 'Computer Monitor', N'Atas', 1300000,  1300000, 1500000, 9, 3, 3, 3)
```

```
insert into OrderReport values (1, 'A01', 'ST01', 1, 1,  N'Cash', 5, 7500000, '2023-05-06'),
                               (2, 'A02', 'ST01', 2, 2,  N'Cash', 5, 4500000, '2023-05-08'),
                               (3, 'A03', 'ST01', 3, 3,  N'Cash', 5, 7500000, '2023-05-09')
```

```
insert into Agents_Products values ( 'A01', 1),
                                   ( 'A02', 2),
                                   ('A03', 3)
```

4.4 Software Classes Method Code

1. JAVA

```java
public interface DeliveryNoteHandler{
    public void updateQuantity(int productID, int deliveryQuantity, int deliveryPrice, int dateUpdate);
    public boolean checkGoods( int productID, int quantity, int price);
    public String printReport( int productID, int deliveryQuantity, int deliveryPrice, int deliveryDate, String payment, int datePrint);
}
```

```java
public abstract class DeliveryNote implements DeliveryNoteHandler{
    public int deliveryID;
    public int productID;
    public int deliveryQuantity;
    public int deliveryPrice;
    public int deliveryDate;

    public DeliveryNote() {
    }

    public DeliveryNote(int deliveryID, int productID, int deliveryQuantity, int deliveryPrice, int deliveryDate) {
        this.deliveryID = deliveryID;
        this.productID = productID;
        this.deliveryQuantity = deliveryQuantity;
        this.deliveryPrice = deliveryPrice;
        this.deliveryDate = deliveryDate;
    }

    public DeliveryNote(int deliveryID, int productID, int deliveryQuantity, int deliveryPrice) {
        this.deliveryID = deliveryID;
        this.productID = productID;
        this.deliveryQuantity = deliveryQuantity;
        this.deliveryPrice = deliveryPrice;
    }
```

```java
public int getDeliveryID() {
        return this.deliveryID;
    }

    public void setDeliveryID(int deliveryID) {
        this.deliveryID = deliveryID;
    }

    public int getProductID() {
        return this.productID;
    }

    public void setProductID(int productID) {
        this.productID = productID;
    }

    public int getDeliveryQuantity() {
        return this.deliveryQuantity;
    }

    public void setDeliveryQuantity(int deliveryQuantity) {
        this.deliveryQuantity = deliveryQuantity;
    }

    public int getDeliveryPrice() {
        return this.deliveryPrice;
    }

    public void setDeliveryPrice(int deliveryPrice) {
        this.deliveryPrice = deliveryPrice;
    }

    public int getDeliveryDate() {
        return this.deliveryDate;
    }

    public void setDeliveryDate(int deliveryDate) {
        this.deliveryDate = deliveryDate;
    }
```

```java
public void updateQuantity(int productID, int deliveryQuantity, int deliveryPrice, int dateUpdate){
        setProductID(productID);
        setDeliveryQuantity(this.deliveryQuantity + deliveryQuantity);
        setDeliveryPrice(deliveryPrice);
        getDeliveryDate(dateUpdate);
    }

    public boolean checkGoods(int productID, int quantity, int price) {
        // read database and check if goods are exist or not
        return true
    }

    public String printReport( int productID, int deliveryQuantity, int deliveryPrice, int deliveryDate, String payment, int datePrint){
        return "{" +
            " deliveryID='" + getDeliveryID() + "'" +
            ", productID='" + productID + "'" +
            ", quantity='" + deliveryQuantity + "'" +
            ", price='" + deliveryPrice + "'" +
            ", date='" + deliveryDate + "'" +
            ", payment='" + payment + "'" +
            ", datePrint='" + datePrint + "'" +
            "}";
    }

}
```

```java
public class Report extends DeliveryNote{
    public int deliveryID;
    public int productID;
    public int deliveryQuantity;
    public int deliveryPrice;
    public int deliveryDate;
    public String payment;
    public int datePrint;

    public Report() {
    }

    public Report(int deliveryID, int productID, int deliveryQuantity, int deliveryPrice, int deliveryDate, String payment, int datePrint) {
       super(deliveryID, productID, deliveryQuantity, deliveryPrice, deliveryDate);
    }
```

```
1
2     public int getDeliveryID() {
3         return this.deliveryID;
4     }
5
6     public void setDeliveryID(int deliveryID) {
7         this.deliveryID = deliveryID;
8     }
9
10    public int getProductID() {
11        return this.productID;
12    }
13
14    public void setProductID(int productID) {
15        this.productID = productID;
16    }
17
18    public int getDeliveryQuantity() {
19        return this.deliveryQuantity;
20    }
21
22    public void setDeliveryQuantity(int deliveryQuantity) {
23        this.deliveryQuantity = deliveryQuantity;
24    }
25
26    public int getDeliveryPrice() {
27        return this.deliveryPrice;
28    }
29
30    public void setDeliveryPrice(int deliveryPrice) {
31        this.deliveryPrice = deliveryPrice;
32    }
33
34    public int getDeliveryDate() {
35        return this.deliveryDate;
36    }
37
38    public void setDeliveryDate(int deliveryDate) {
39        this.deliveryDate = deliveryDate;
40    }
41
42    public String getPayment() {
43        return this.payment;
44    }
45
46    public void setPayment(String payment) {
47        this.payment = payment;
48    }
49
50    public int getDatePrint() {
51        return this.datePrint;
52    }
53
54    public void setDatePrint(int datePrint) {
55        this.datePrint = datePrint;
56    }
57
```

```java
public String printReport() {
        return "{" +
            " deliveryID='" + getDeliveryID() + "'" +
            ", productID='" + getProductID() + "'" +
            ", deliveryQuantity='" + getDeliveryQuantity() + "'" +
            ", deliveryPrice='" + getDeliveryPrice() + "'" +
            ", deliveryDate='" + getDeliveryDate() + "'" +
            ", payment='" + getPayment() + "'" +
            ", datePrint='" + getDatePrint() + "'" +
            "}";
    }
```

```java
public class CheckGoods extends DeliveryNote{

    public int deliveryID;
    public int productID;
    public int quantity;
    public int price;

    public CheckGoods(int deliveryID, int productID, int quantity, int price) {
       super(deliveryID, productID, quantity, price)
    }
```

```java
public int getDeliveryID() {
        return this.deliveryID;
    }

    public void setDeliveryID(int deliveryID) {
        this.deliveryID = deliveryID;
    }

    public int getProductID() {
        return this.productID;
    }

    public void setProductID(int productID) {
        this.productID = productID;
    }

    public int getQuantity() {
        return this.quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public int getPrice() {
        return this.price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

```

```java
  public boolean checkGoods(int productID, int quantity, int price) {
        // read database and check if goods are exist or not
        return true
    }
```

```java
public class Quantity extends DeliveryNote{

    public Quantity() {
    }

    public Quantity(int deliveryID, int productID, int deliveryQuantity, int deliveryPrice, int dateUpdate) {
        super(deliveryID, productID, deliveryQuantity, deliveryPrice, dateUpdate)
    }
```

```java
public int getDeliveryID() {
        return this.deliveryID;
    }

    public void setDeliveryID(int deliveryID) {
        this.deliveryID = deliveryID;
    }

    public int getProductID() {
        return this.productID;
    }

    public void setProductID(int productID) {
        this.productID = productID;
    }

    public int getDeliveryQuantity() {
        return this.deliveryQuantity;
    }

    public void setDeliveryQuantity(int deliveryQuantity) {
        this.deliveryQuantity = deliveryQuantity;
    }

    public int getDeliveryPrice() {
        return this.deliveryPrice;
    }

    public void setDeliveryPrice(int deliveryPrice) {
        this.deliveryPrice = deliveryPrice;
    }

    public int getDateUpdate() {
        return this.dateUpdate;
    }

    public void setDateUpdate(int dateUpdate) {
        this.dateUpdate = dateUpdate;
    }
```

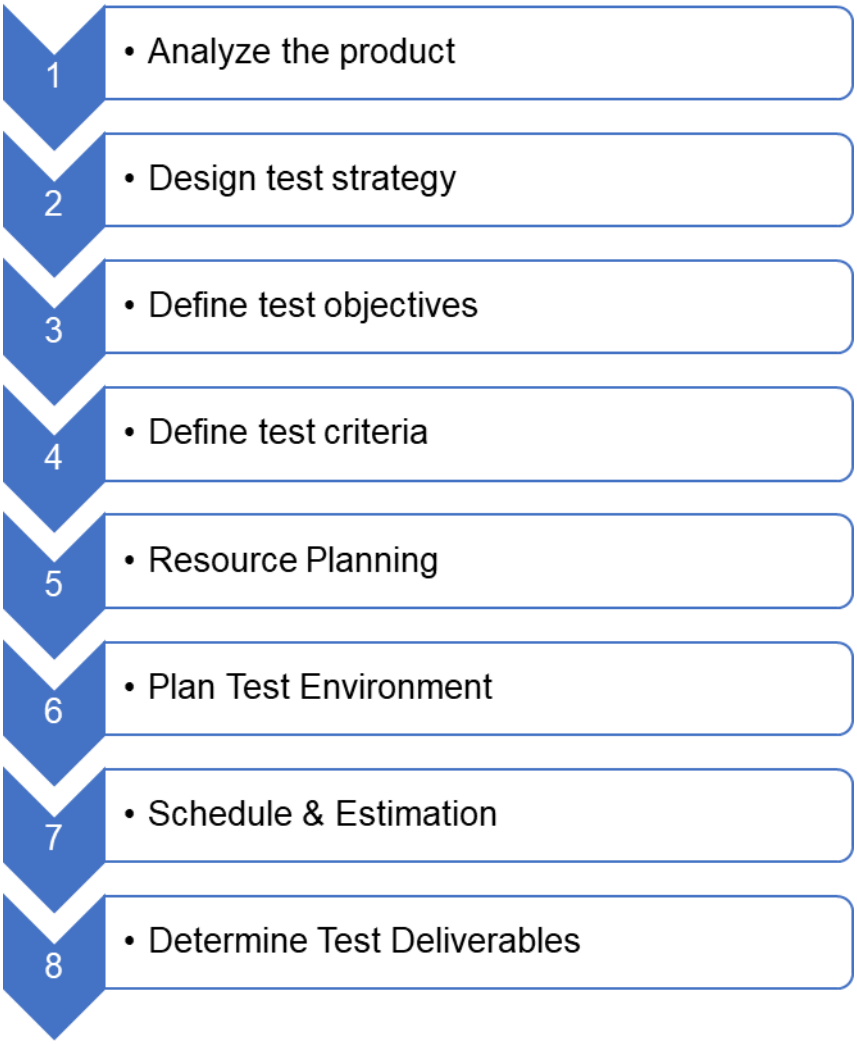
```
public void updateQuantity(int productID, int deliveryQuantity, int deliveryPrice, int
dateUpdate){
        setProductID(productID);
        setDeliveryQuantity(this.deliveryQuantity + deliveryQuantity);
        setDeliveryPrice(deliveryPrice);
        setDateUpdate(dateUpdate);
    }

```

V. SYSTEM TESTING, DEPLOYMENT AND DEMONSTRATION

5.1 Testing: Test plan & Test case

Test plan for this system, we follow the seven steps of IEEE 829:



- Test strategy:

Module Name	Applicable Role	Description
Sign up	Customer Manager Accountants Warehouse Staffs	Enter information and create a new account
Login	Customer Manager Accountants Warehouse Staffs	Enter account's username and password to log in to the system
Create Goods Received	Accountants	Accountants can fill the forms to create Goods Received when the distributor imports goods
Place an order	Customer	Customer can place an order of items by themselves and choose a payment method
Make Goods Delivery Note	Accountants	Accountants can make Goods Delivery Note to deliver goods to agents, update the status of orders as being transferred and update the payment status of agents.
Scan barcode/QR codes/RIFDs	Warehouse Staffs	Perform the import/export goods warehousing processes.
Inventory movement	Accountants	Accountants shall be able to view incoming/outgoing stock report

  - Risk and Issue:
    - Team member lacks of requires skill for testing.
    - Wrong time-estimation
    - Database (environment) has some problem and is not responding.

- Test objective
  - Focus on testing the system operation (Create Goods Received, Inventory movement,…) to guarantee all operations can work normally in company.

- Test criteria
    - Conditions to exit test criteria:
        - Run rate is mandatory to be 100%.
        - Pass rate is 100%, unless a clear reason is given.
- Resource planning:

System Resource	**Human Resource**
- SQL: SSMS to run SQL server - **Test tool** - **Network** - Computer: As least 2GB RAM and 64GB storage available	- Test Manager: Van Hung - Developer in Testing: Van Hung & My Duyen - SQA members

- Test case:

No.	Test case	Processes	Expected Result	Test
1	Customer signup with personal data	Enter username Enter password Enter phone number Press “Sign up”	Display “Successful!” and save account to database	Pass
2	Customer login with wrong account username	Enter wrong username Enter password Press “Login”	Display “The username or password is not valid”	Pass
3	Accountants create Goods Received	Click “Import Goods” Enter Product information Press “Import”	Display “Successful!” and update database	Pass
4	Place an order	Add items to wishlist Open wishlist Click “Buy”	Display payment form and allow user to choose payment method	Pass
5	Payment	Choose Payment method Enter payment information Click “Yes” in confirm form	Display “Payment successful!” and show receipt to customer, update database.	Pass
6	Scan barcode/QR codes/RIFDs	Scan codes Input information import/export products Press “Save”	Display “Successful!” and update database	Pass
7	Sign out	Press “Sign out”	Sign out of the system and display login/signup form	Pass

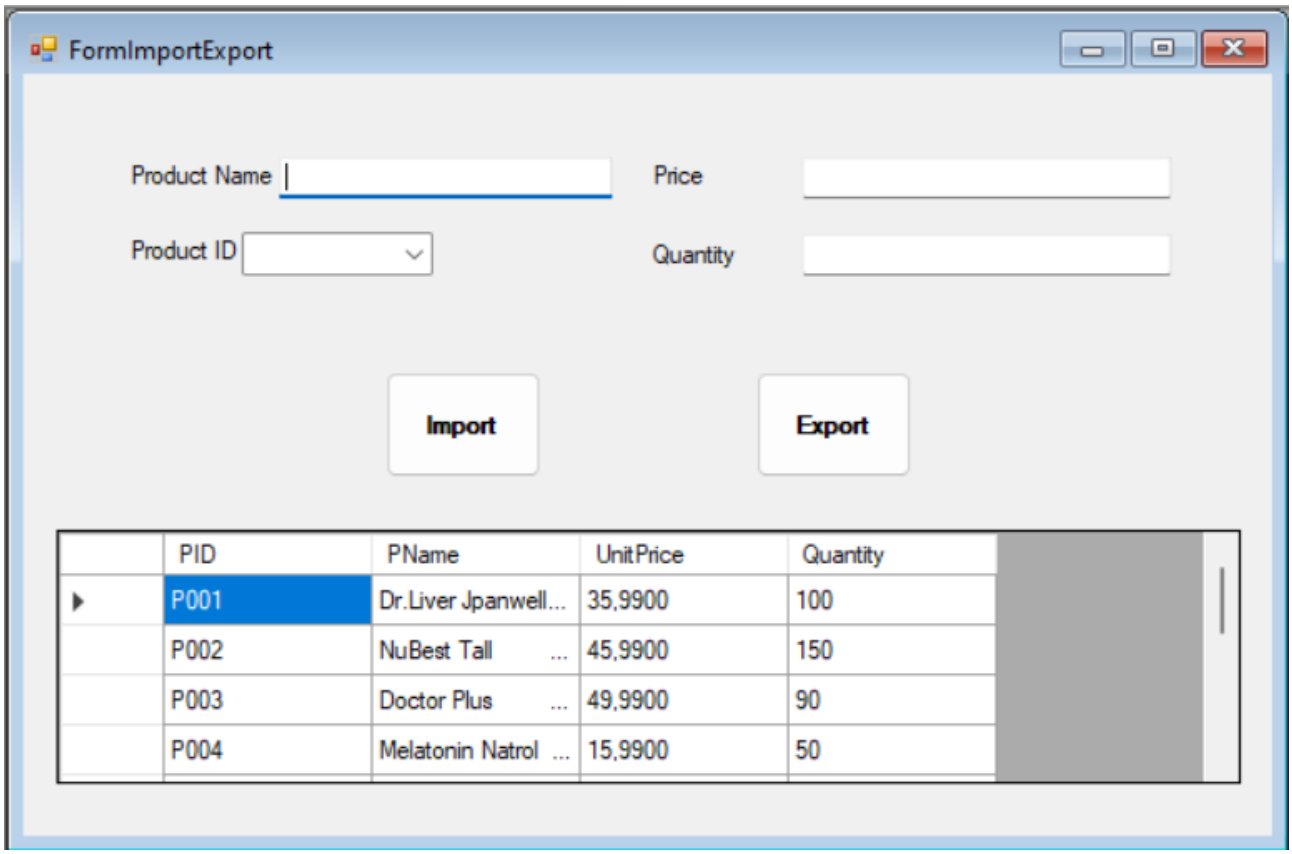
5.2 Deployment

We will improve the system by deploying more functions. We will develop it suitable for all age users. That will help the system get more comments from users, which helps us to improve further.

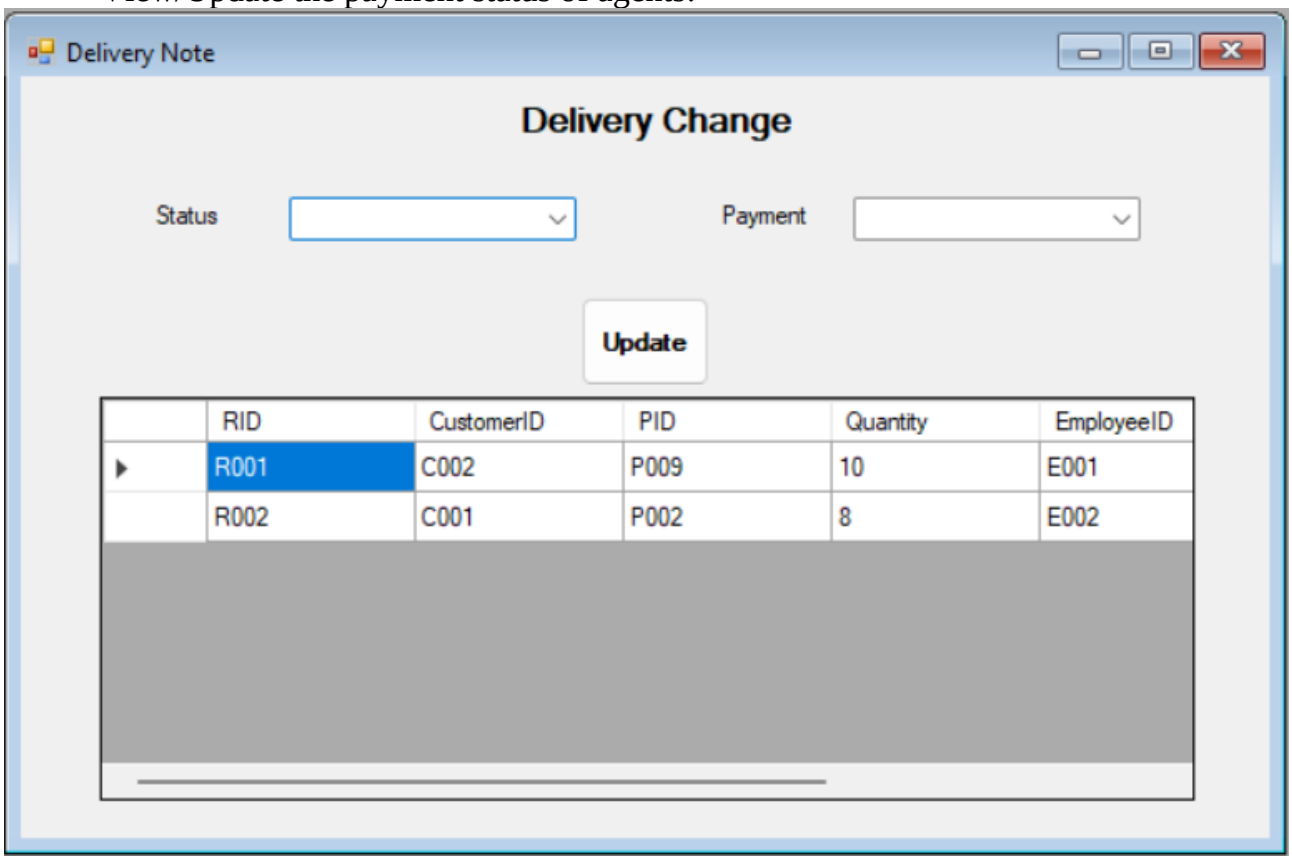
5.3 Demonstration

Demo of system:

- Create Goods Received:



- View/Update the payment status of agents:



Conclusions/ Recommendations

This system is just a functional process without complicated process so far. The identified actors play an important role to the full functioning of the business,….

References

B.1. *Slide* ***Requirements Modelling***

B.2. *Book-MultiAuthor, Gra*

Internet Link:

I.1. *http://www.uml-dox.net/Addison.Wesley-UML.for.Mere.Mo/0321246241/ch02lev1sec6.html*

I.2. *https://www.geeksforgeeks.org/difference-between-sequence-diagram-and-collaboration-diagram/?fbclid=IwAR3yvEb4HP4-KpJmYcHLYYBsUAYjtw5hNxDZmEcxhofWC-mfUo2PgpY1HAU*

I.3. *Test plan: https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html*

Assessment Rubric

Subject: Requirements Analysis and Design

Final project (50%)

For concrete final project report, the lecturer will give consideration to the following criteria:

Criteria	Marks range	1	2	3	Grade point range	Alpha grade range
	Point /10	**1/4 point (Poor)**	**1/2 point (Fair)**	**Full points (Good)**		
1/ Form of Document	**1.5**					
• Document Format	*0.5*	• Majority or all of the section does not follow the prescribed format (0.15)	• Some of the sections do not follow the prescribed format (0.25)	• All the sections of the document followed the prescribed format (0.5)	0.5	
• Composition and organization	*1.0*	• Make considerable effort to understand the underlying logic and flow of ideas. • Diagrams were absent or inconsistent with the text. • Grammatical and spelling errors made it difficult for the reader to interpret the text in places. ***(0.25)***	• The deliverable was organized and clearly written for the most part. In some areas the logic and/or flow of ideas were difficult to follow. • Words were well chosen with some minor expectations. • Diagrams were consistent with the text. • Sentences were mostly grammatical and/or only a few spelling errors were present but they did not hinder the reader. • **(0.5)**	• The deliverable was well organized and clearly written. The underlying logic was clearly articulated and easy to follow. • Words were chosen that precisely expressed the intended meaning and supported reader comprehension. • Diagrams or analyses enhanced and clarified presentation of ideas. • Sentences were grammatical and free from errors. **(0.75-1.00)**	1.0	

2/ Introduction and Conclusion about topics	**0.5**	• The output provided were not substantial and requires further study. (0.15)	• The output provided were substantial but some sections of the documents need more in-depth analysis. (0.25)	• The output provided were substantial that shows in depth understanding and research of the system requirement. (0.5)	0.5	
Criteria	**Marks range**	**1**	**2**	**3**	**Grade point range**	**Alpha grade range**
	Point /10	**1/4 point (Poor)**	**1/2 point (Fair)**	**Full points (Good)**		
3/ Analysis Presentation	**5**	• Some or Most of the output incorrectly used or integrated concepts learned.	• Some of the output incorrectly used or integrated concepts learned.	• Concepts learned were correctly integrated and used in the output.		
3.1 Structural Modelling (Class Diagram) (Deployment & Component)	*1.5*	• Some or Most of the Domain Class diagram incorrectly used or integrated concepts learned (0.25-0.5)	• Some the Domain Class diagram incorrectly used or integrated concepts learned (0.75-1.0)	• Domain Class diagram learned were correctly integrated and used in the output (1.25-1.5)	1.25	
	Deployment diagram: 0.25 (if correct); Component Diagram: 0.5(if correct)					
3.2 Behavioral Modelling	*3.5*	Some or Most of diagram incorrectly used: • Use Case Diagram: 0.15 • Use Case Description:0.25 • Sequence Diagram: 0.15 • Collaboration: 0.15 • State-chart diagram: 0.15 • Activity diagram: 0.15	Some diagram incorrectly used: • Use Case Diagram: 0.25 • Use Case Description: 0.5 • Sequence Diagram: 0.25 • Collaboration: 0.25 • State-chart diagram: 0.25 • Activity diagram: 0.25	Diagram learned correctly used: • Use Case Diagram: 0.5 • Use Case Description:1.0 • Sequence Diagram: 0.5 • Collaboration: 0.5 • State-chart diagram: 0.5 • Activity diagram: 0.5	3.25	

4/ Design Presentation	**2.0**	Some or Most of diagram incorrectly used: • Design Class Diagram: 0.25 • Map attribute to DB design: 0.05 • Add DB to sequence Diagram: 0.05 • Map method to UI design: 0.15	Some diagram incorrectly used: • Design Class Diagram: 0.5 • Map attribute to DB design: 0.15 • Add DB to sequence Diagram: 0.15 • Map method to UI design: 0.25	Diagram learned correctly used: • Design Class Diagram: 1.0 • Map attributes to DB design: 0.25 • Add DB to sequence Diagram: 0.25 • Map methods to UI design: 0.5	1.75	
5/ Implementation	**1.0**	Write the Code for the Design Class to Implement incorrectly 0.25	Write the Code for the Design Class to Implement somehow correctly 0.5	Write the Code for the Design Class to Implement correctly used 0.75-1.0	0.75	
Total grade point	**10**				9.0	