

A DISTRIBUTOR SELLING ELECTRONICS DEVICES PRODUCTS

Requirements Analysis & Design (RAD)

By Students:

1. 521H0504 : **Bui Huu Loc**

Reference: Team_01_RAD_Requirements_Modelling_v0.1

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

Outcome: A Distributor selling electronics devices products

Abstract: This document provides an in-depth analysis of a distributor selling electronics devices products system with the requirements modelled utilizing the UML framework.

Intellectual Property

Copyright 2023 Team 01.

The following documentation, the content therein and/or the presentation of its information is proprietary to and embodies the confidential processes, designs, technologies and otherwise of Team 01. All copyright, trademarks, trade names, patents, industrial designs, and other intellectual property rights contained herein are, unless otherwise specified, the exclusive property of Team 01.

The ideas, concepts and/or their application, embodied within this documentation remain and constitute items of intellectual property which nevertheless belong to Team 01

The information (including, but by no means limited to, data, drawings, specification, documentation, software listings, source and/or object code) shall not be disclosed, manipulated, disseminated or otherwise in any manner inconsistent with the nature and/or conditions under which this documentation has been issued.

The information contained herein is believed to be accurate and reliable. Team 01 accepts no responsibility for its use in any way whatsoever. Team Five shall not be liable for any expenses, damages and/or related costs which may result from the use of the information contained herein.

The information contained herein is subject to change without notice.

All Rights Reserved. Copyright herein is expressly protected at common law, statute and under various International and Multi-National Treatises (including, but by no means limited to, the Berne Convention for the Protection of Literary and Artistic Works).

Table of Contents

Intellectual Property	2
Table of Contents	2
Table of Figures	4
Executive Summary	6
I. Business Requirements	7
1.1 Organization Chart / Project Chart/Gantt Char	7

1.2 Business Modelling / Requirements	9
1.3 Business Processes / Flowchart of Requirements	10
1.4 Activity Diagram	11
1.5 List of Requirements	11
II. System Requirements Analysis	12
2.1 Translate from Business Use Case	12
2.1.1 System Narrative	12
2.1.2 Users and their goals	12
2.1.3 List of Events	14
2.1.4 List of Actors	16
2.1.5 List of Use Cases	16
2.1.6 Use Case Diagram	17
2.1.7 Domain Class Model Diagram	18
2.2 Use Case Descriptions	18
2.2.1 Use Case: Manage orders.	18
2.2.2 Use Case: Statistics	22
2.2.3 Use Case: Manage shopping cart	24
2.3.1 Verifying uses cases for Manage Order	28
2.3.2 Verifying uses cases for Statistics.	28
2.3.3 Verifying uses cases for Manage Shopping Cart	28
III. System Requirements Design (3.0 points)	30
3.1 Design Class for Use Case Manage Order	30
3.1.1 Design Classes in Detailed Design	30
3.1.2 Design Class Diagram	31
3.1.3 Final Design Class Diagram	34
3.2 Design Class for Use Case Statistic	35
3.2.1 Design Classes in Detailed Design	35
3.2.2 Design Class Diagram	36
3.2.3 OOD with Communication	38
3.2.3 Final Design Class Diagram	39
3.3 Design Class for Use Case Manage Shopping Cart	40
3.3.1 Design Classes in Detailed Design	40
3.3.2 Design Class Diagram	41

3.3.2 OOD with Communication	43
3.3.3 Final Design Class Diagram	44
IV. System Requirements Implementation	45
4.1 Design Class for Sub System	45
4.2 Implementation	46
4.2.1 Map persistent objects to the tables in a database	46
4.2.2 Modifying sequence diagrams.	48
4.2.3 UI design	48
4.2.4 SQL Code	51
4.2.5 Software Classes Method Code	54
V. SYSTEM TESTING, DEPLOYMENT AND DEMONSTRATION	59
5.1 Testing: Test plan & Test case	59
5.2 Deployment	63
5.3 Demonstration	64
Conclusions/ Recommendations	69
References	69

Table of Figures

Figure 1 Organization Chart	7
Figure 2 Business Modelling / Requirements	9
Figure 3 Business Processes / Flowchart of Requirements	10
Figure 4 Use case diagram	17
Figure 5 Customer Account Subsystem Domain Model Class Diagram	18
Figure 6 Activity Diagram for Use Case: Manage orders.	20
Figure 7 System sequence diagram for Use Case: Manage orders	21
Figure 8 Activity diagram for Use Case: Statistics	23
Figure 9 System sequence diagram for Use Case: Statistics	24
Figure 10 Activity diagram for Use Case: Make shopping cart	26
Figure 11 System sequence diagram for Use Case: Make shopping cart	27
Figure 12 Design Classes in Detailed Design	30
Figure 13 Domain Design Class	31
Figure 14 Controller	31
Figure 15 UI	32
Figure 16 Data access	33
Figure 17 Design class	33
Figure 18 Accountant class	35
Figure 19 Domain Design Class	36

Figure 20 use case statistic Controller	36
Figure 21 UI statistic	37
Figure 22 database access	37
Figure 23 OOD with Communication for use case 2	38
Figure 24 3 Final Design Class Diagram use case 2	39
Figure 25 Design Classes in Detailed Design use case 3	40
Figure 26 . Domain Design Class	41
Figure 27 Controller use case 3	41
Figure 28 UI use case 3	42
Figure 29 database access	42
Figure 30 OOD with Communication	43
Figure 31 3 Final Design Class Diagram	44
Figure 32 Design Class for Sub System	45
Figure 33 Mockup Order Form	48
Figure 34 Mockup OrderDetail Form	49
Figure 36 Mockup Statistic Form	50
Figure 37 Software Classes Method Code Class Product	54
Figure 38 Software Classes Method Code Class Order	55
Figure 39 Software Classes Method Code Class Order Product	56
Figure 40 Software Classes Method Code Class User	57
Figure 41 Software Classes Method Code Class Admin	58
Figure 42 Login form	64
Figure 43 Manage Order	65
Figure 44 Detail Order	66
Figure 45 Update Order	67
Figure 46 Statistic	68

Executive Summary

Today, the application of Information Technology in economic sectors and daily life has brought great benefits. The management of import and export goods is one of the important activities in any goods distribution unit. The purpose of this document is to provide an overview of the import and export management system, applying the knowledge learned in requirements analysis, web development, database design and optimization. optimize user experience to build a professional and effective information system. The program is built to serve the goods management of distributors of electronic goods for agents. The system serves for order management and import, export management, payment management between distributors and agents and suppliers, inventory management, management of best-selling items. most, manage sales revenue and show monthly revenue results.

The system meets the basic requirements in updating and processing goods import and export quickly and accurately such as updating data, searching for information, general report on import and export of inventory, ... User-friendly and easy interface.

I. Business Requirements

1.1 Organization Chart / Project Chart/Gantt Char

* Organization Chart:

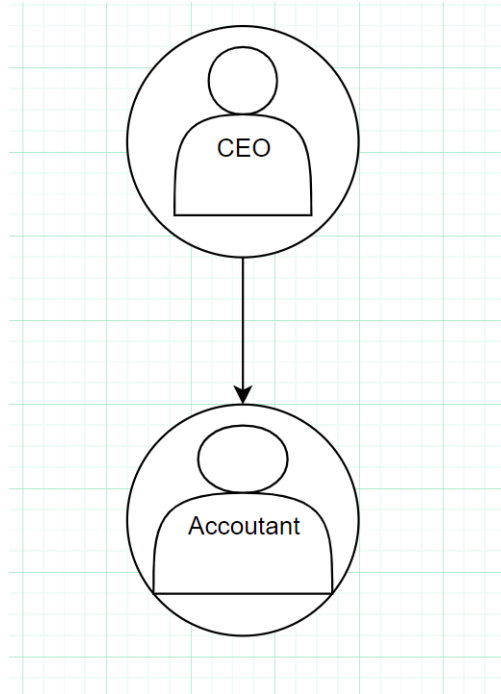


Figure 1 Organization Chart

* Project Chart:

Task Name	Duration	Start Date	End Date	Dependencies
Project Planning	3	2023-04-01	2023-04-03	-
System Analysis	4	2023-04-04	2023-04-07	1
Database Design	3	2023-04-08	2023-04-10	2
User Interface Design	3	2023-04-11	2023-04-13	2
Development	10	2023-04-14	2023-04-23	3,4
Quality Assurance Testing	4	2023-04-24	2023-04-27	5
Deployment and Training	3	2023-04-28	2023-05-01	6

This project chart shows a project that will begin on April 1, 2023, and the system will be deployed by May 1, 2023.

The project has seven tasks, each with an estimated time of completion with a start and end date. Task 1 is project planning, which includes goal setting, road mapping, and project planning.

Task 2 is Systems Analysis, where the Systems Analyst will work with stakeholders to define requirements and design solutions that meet the needs of the organization.

Task 3 is Database Design, where the Database Administrator will design and implement the database supporting the system.

Task 4 is User Interface Design, where UI Designers create user-friendly interfaces that link customers to company products or services.

Task 5 is Development, where the Developers will code the system based on the designs from task 2, 3 and 4, and do the installation.

Task 6 is Quality Assurance Testing where the Quality Assurance Specialist will test the system to ensure that it is working correctly and meeting the requirements.

Task 7 is Deployment and Training, where the system will be deployed to end users and training will be provided to accounting, sales and marketing staff.

The dependencies between tasks are shown in the last column. Task 2 depends on task 1, task 3 and 4 depends on task 2, and task 5 depends on task 3 and 4. By defining these dependencies, we can ensure ensures that tasks must be completed before their dependent tasks can begin.

* Gantt Chart

Task Name	Duration	Start Date	End Date	Progress
Project Planning	3	2023-04-01	2023-04-03	100%
System Analysis	4	2023-04-04	2023-04-07	100%
Database Design	3	2023-04-08	2023-04-10	100%
User Interface Design	3	2023-04-11	2023-04-13	100%
Development	10	2023-04-14	2023-04-23	100%
Quality Assurance Testing	4	2023-04-24	2023-04-27	100%
Deployment and Training	3	2023-04-28	2023-05-01	100%

This Gantt chart provides a graphical representation of a project's timeline, showing the start and end dates of each task, the duration of each task, and the progress achieved.

Task 1 (Project Planning) starts on April 1 and ends on April 3. Task 2 (System Analysis) starts on April 4 and ends on April 7. Task 3 (Database Design) starts on April 8th and ends on April 10th. Task 4 (User Interface Design) starts April 11th and ends on April 11th. April 13. Task 5 (Development) is the longest, starting on April 14 and ending on April 25. Task 6 (Quality Assurance Testing) begins April 26 and ends April 29. Task 7 (Implementation and training) begins April 30 and ends May 1.

1.2 Business Modelling / Requirements

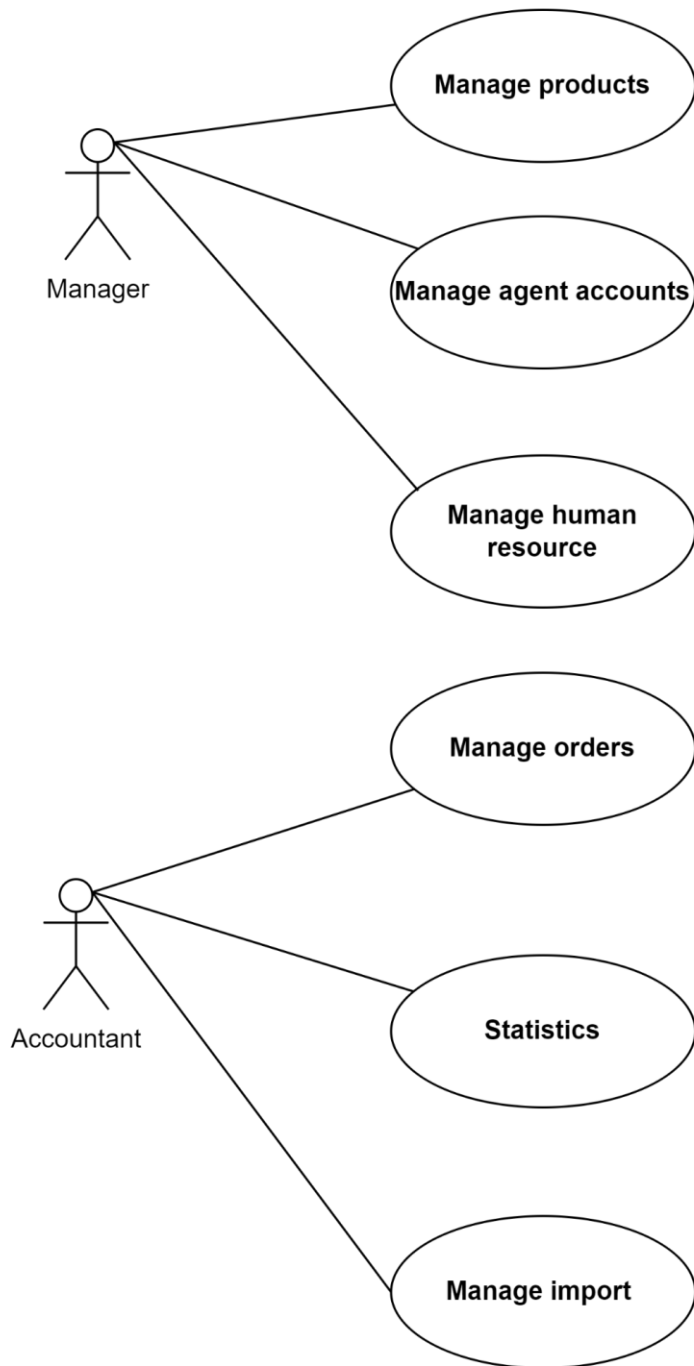


Figure 2 Business Modelling / Requirements

1.3 Business Processes / Flowchart of Requirements

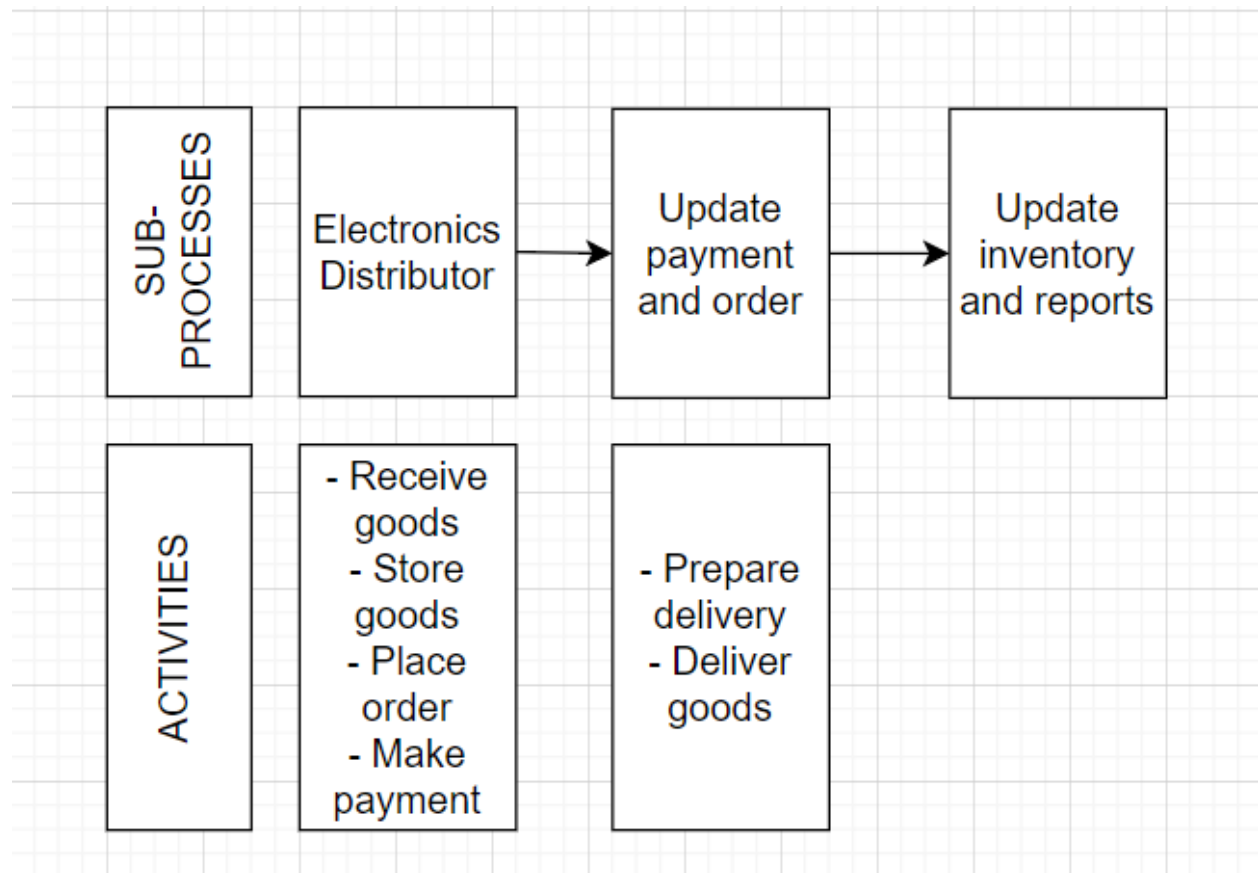
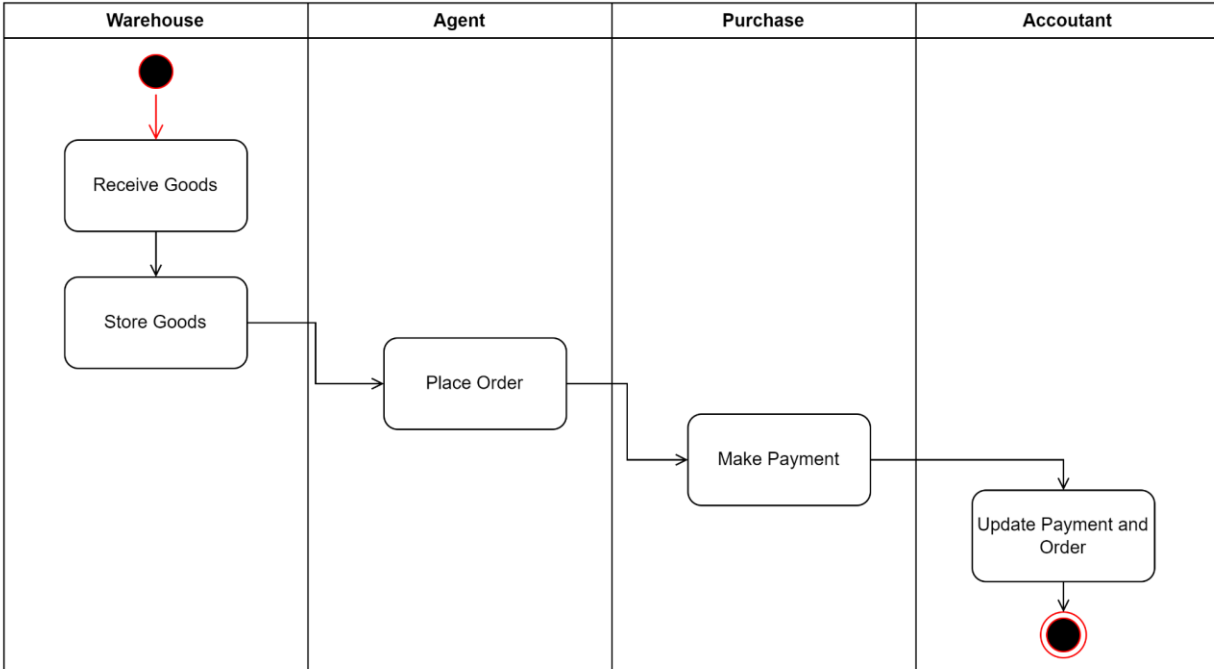


Figure 3 Business Processes / Flowchart of Requirements

1.4 Activity Diagram



1.5 List of Requirements

Usability	User-friendly interface, easy to understand, can be used by everyone.
Reliability	The software must be stable, reliable, and function as expected with no errors or system crashes.
Security	Software must have strong security features to protect information such as customer data, and financial transactions. Only authorized persons can access and modify to identify and correct vulnerabilities.
Performance	Requires software that will be able to handle a large number of visitors and transactions without significant performance loss. Processing speed must be efficient, along with responsiveness.

II. System Requirements Analysis

2.1 Translate from Business Use Case

2.1.1 System Narrative

The electronics distributor requires an information system to manage the sales process for their products. The system is expected to enable authorized resellers or agents to place orders for products and choose their preferred payment method. The system will also allow the accountant staff to perform goods receiving and warehousing processes by scanning barcode/QR code, RFID of incoming goods.

Accountants will be able to create Goods Received Notes when the distributor imports goods. These notes will include multiple items, and the accountant staff will scan the barcodes, QR codes, and RFID tags on each item during the warehousing process. The system will enable the accounting staff to create Goods Delivery Notes based on previous orders to deliver goods to agents. These delivery slips will be printed, and the order status will be updated to show that the items have been transferred. Additionally, the payment status of agents will be updated.

The system will also generate incoming/outgoing stock reports, best-selling product reports, and revenue reports monthly, enabling accountants to view inventory movement, identify top-selling products, and track revenue.

The system will feature a user-friendly interface with various functions that will enable employees to manage orders, products, imports, employees, and agents. The Manage Order function will allow employees to view all orders, search and filter orders, update order status, payment status, and print order details. The Manage Product function will enable employees to view all products, search, and filter products, create, update, and delete products. The Manage Import function will enable employees to view all imports, search and filter imports, print import details, and import products.

The system will include a Statistics function that generates reports on revenue and top products. The Manage Employee function will enable admins to view all employees, search and filter employees, lock and unlock accounts, and add new employees. Similarly, the Manage Agent function will allow admins to view all agents, search and filter agents, lock and unlock accounts, and add new agents.

Agents will be able to view all products, search and filter products, add products to a shopping cart, increase or decrease the quantity of items in the cart, fill in order details such as name, address, and phone number, and make payments through the payment gateway. They can view history orders.

2.1.2 Users and their goals

For the accounting staff, the goal is to accurately record incoming and outgoing goods, create goods received and delivery notes, and generate reports on inventory movement, revenue, and top-selling products.

Accountant staff aim to scan barcode/QR Code/RFID tags accurately during the goods receiving and warehousing process. Admins aim to manage the system effectively, including managing orders, products, imports, employees, and agents.

The user goal for the admin is to manage the system efficiently by performing tasks such as adding and managing employees, managing agents, managing orders, managing products, and managing imports. The admin aims to ensure that the system is running smoothly and that all user accounts are secure and well-managed.

The user goal for agents is to be able to view all available products, search and filter products, place orders, and choose their preferred payment method. Agents aim to have a smooth shopping experience by being able to add products to a shopping cart, increase or decrease the quantity of items in the cart, fill in order details such as name, address, and phone number, and make payments through the payment gateway, view history order.

2.1.3 List of Events

List of events and its use case

Event Name	Trigger	Source	Use Case	Response	Destination
Create Goods Received	New goods are imported into the warehouse	Accounting Staff	Manage imports	Generate a Goods Received record and update the inventory database	Accounting Staff
View Order	Accountant want to check orders	Agent	Manage orders	Display the orders	Agent
Process Order	Accountant want update order status, payment status	Agent	Manage orders	Display the processed order	Agent
View Revenue Report	Accounting Staff wants to view revenue	Accounting Staff	View statistics	Display a report of top-selling products by quantity and revenue	Accounting Staff
Manage Employee Account	Admin wants to manage employee accounts	Admin	Manage employees	Add, delete, or modify employee accounts	Admin
Manage Agent Account	Admin wants to manage agent accounts	Admin	Manage agents	Add, delete, or modify agent accounts	Admin
View Product Catalog	Agent wants to view available products	Agent	View products	Display a list of available products	Agent
Add Product to Cart	Agent wants to add a product to their cart	Agent	Manage shopping cart	Add the selected	Agent

				product to the shopping cart	
Fill in Order Information	Agent wants to fill in order details	Agent	Manage orders	Record the agent's order information	Accounting Staff
Make Payment	Agent wants to pay for the order	Agent	Order	Record the payment method for the order	Payment Gateway
View Order History	Agent clicks on the "Order History" button on their dashboard	Agent	View All Order History for Agent	System displays a list of all the agent's order history, including the order date, status, and payment information. The agent can search and filter the order history by various criteria.	

2.1.4 List of Actors

- Accountant Employee: Manage Order, Statistic, Manage Import, Manage Product
- Admin: Can do everything an Employee can do, Manage Employee, Manage Agent
- Agent: View Products, Manage Shopping Cart, Place Order, Make Payment, View History Order

2.1.5 List of Use Cases

- Manage Order: View All Orders, Search and Filter Orders, Update Order Status, Update Payment Status, Print Order Detail
- Manage Product: View All Products, Search and Filter Products, Create, Update, and Delete Products
- Manage Import: View All Imports, Search and Filter Imports, Print Import Detail, Import Products
- Statistic: View Revenue Statistics, View Top Selling Products
- Manage Employee: View All Employees, Search and Filter Employees, Lock and Unlock Employee Accounts, Add New Employees
- Manage Agent: View All Agents, Search and Filter Agents, Lock and Unlock Agent Accounts, Add New Agents
- View Products: Agents can View All Products, Search and Filter Products
- Manage Shopping Cart: Agents can Add Products to Cart, Increase or Decrease the Quantity of Products in Cart
- Place Order: Agents can Fill Out the Order Information, including Name, Address, and Phone Number
- Make Payment: Agents can Pay for the Order using the Payment Gateway
- View History Order: Agents should be able to view all their order history, including the order date, status, and payment information.

2.1.6 Use Case Diagram

The use case diagram of a System can be draw-able as the following :

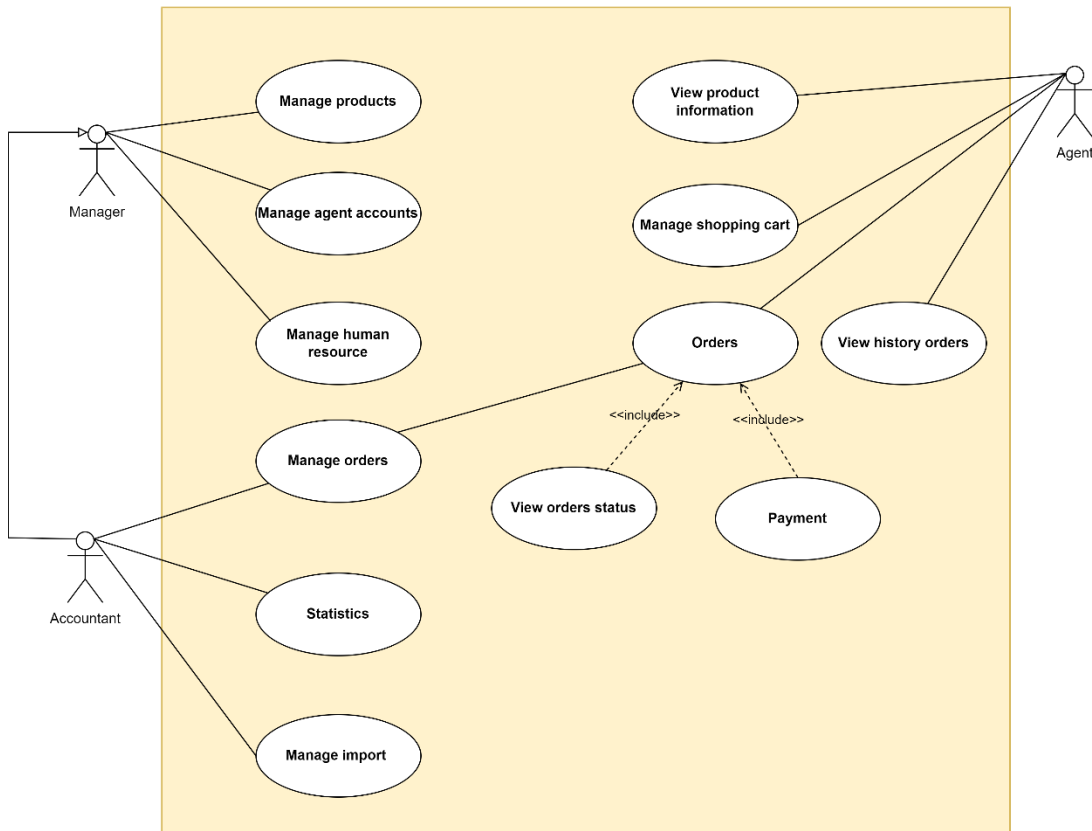


Figure 4 Use case diagram

2.1.7 Domain Class Model Diagram

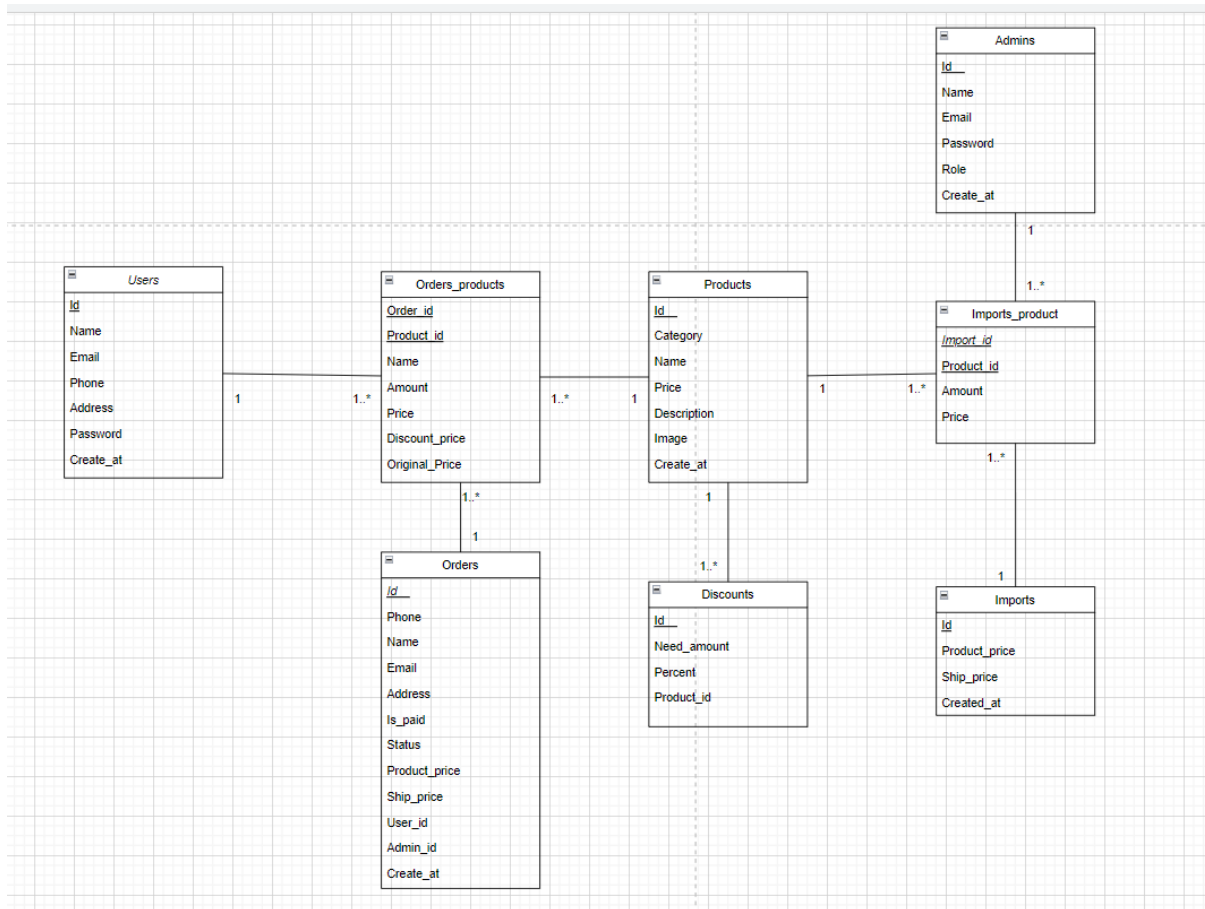


Figure 5 Customer Account Subsystem Domain Model Class Diagram

2.2 Use Case Descriptions

2.2.1 Use Case: Manage orders.

- i. Use case: Manage orders fully description

Use case name	Manage orders	
Participating Actors	Accountant	
Entry Condition(s)	The accountant has logged into the coffee sales management software. At least one order has been created in the system.	
The flow of Events	Accountant	System

	The accountant accesses the order management function.	The system displays a list of orders that have been created in the system, including information such as order number, creation date, customer name, order status, and payment status.
	The accountant can filter the list of orders by different criteria, such as creation date, customer name, order status, or payment status. The accountant selects a specific order from the list to view details.	The system displays detailed information about the order, including the products in the order, quantity, price, total amount, and shipping fee (if any).
	The accountant can update the order status of that order, such as in progress, completed, or canceled. The accountant can also update the payment status of the order, such as paid or unpaid.	The system update to the database
Exit Condition	The accountant has created a new purchase order and updated the inventory.	

ii. Activity Diagram for Use Case: Manage orders.

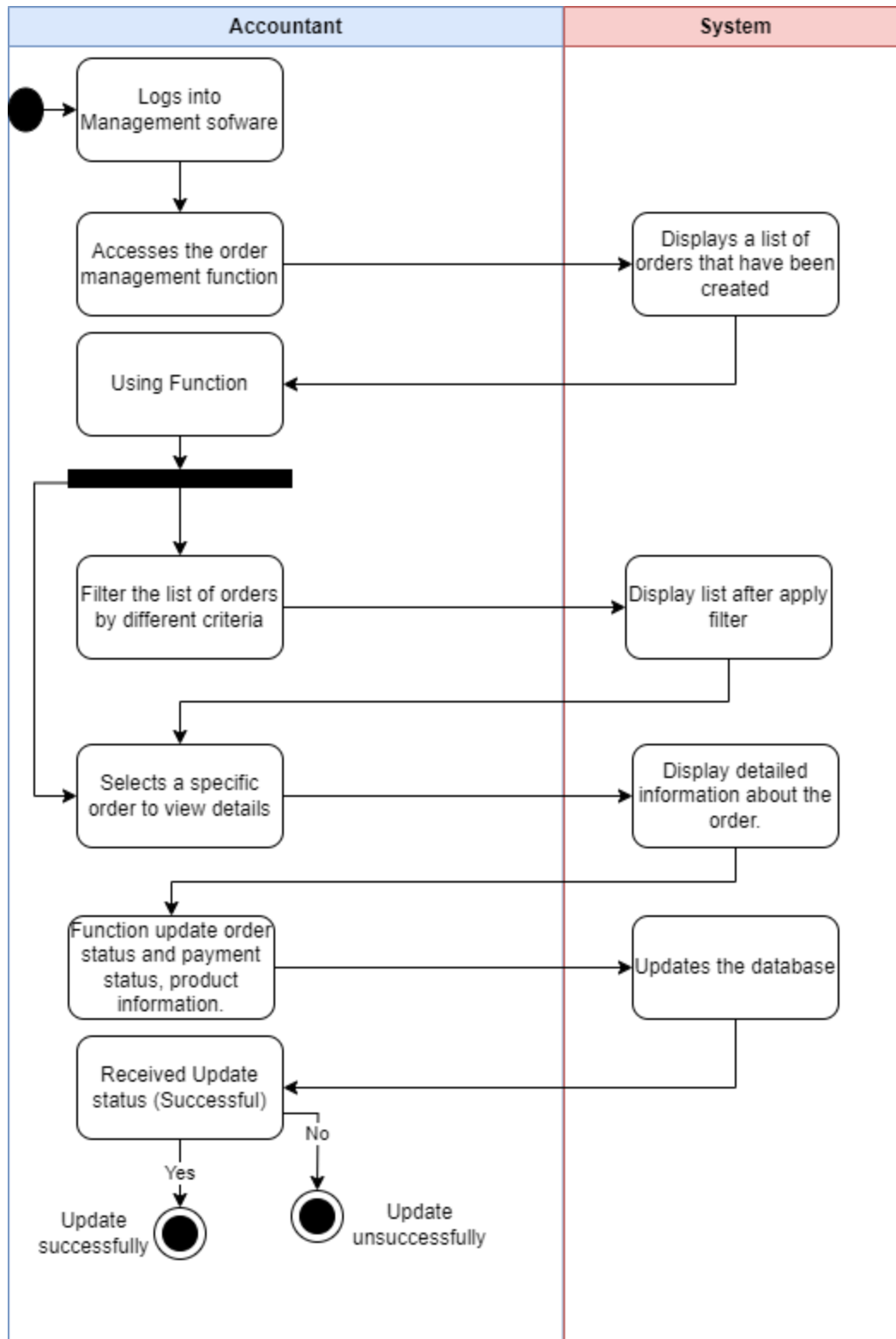


Figure 6 Activity Diagram for Use Case: Manage orders.

- iii. System sequence diagram for Use Case: Manage orders.

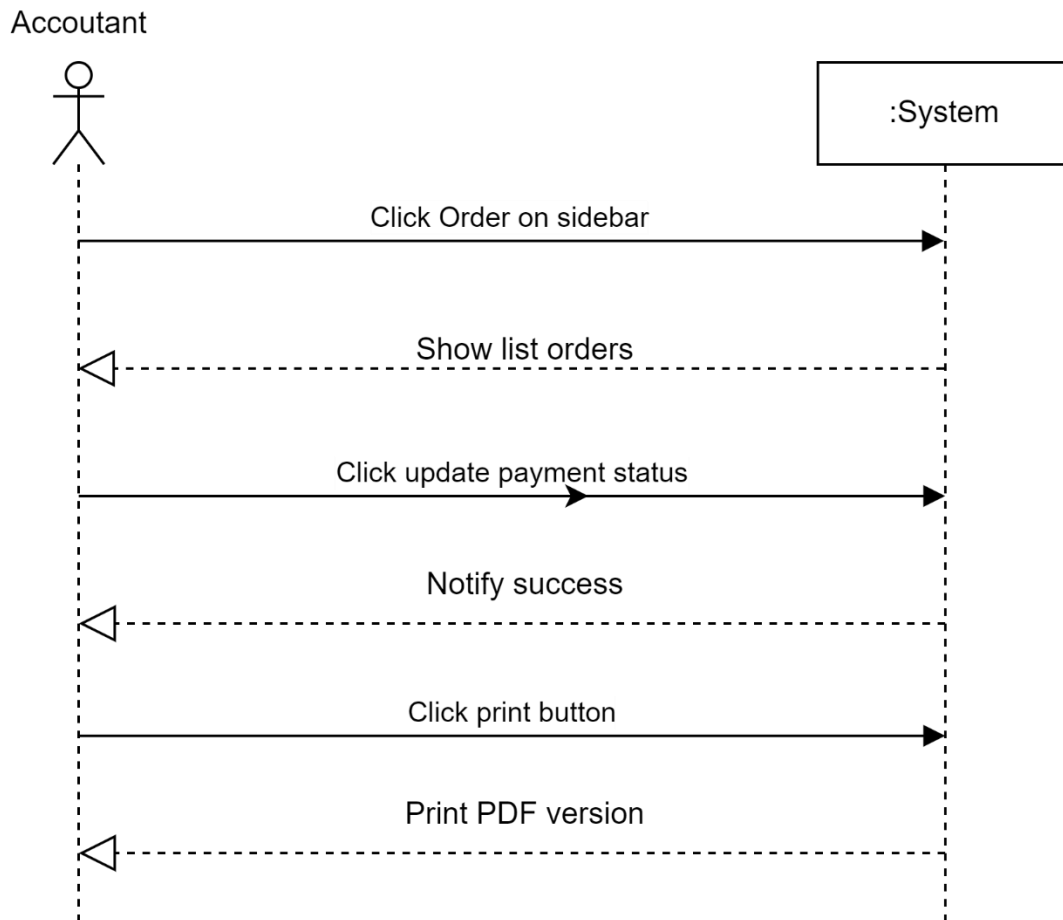


Figure 7 System sequence diagram for Use Case: Manage orders

2.2.2 Use Case: Statistics

- i. Use case: Statistics fully description.

Use case name	Statistics	
Participating Actors	Accountant	
Entry Condition(s)	The accountant has logged in to the system.	
The flow of Events	Accountant	System
	The accountant selects the function to review statistics on outgoing and incoming goods, the best-selling products, and revenue for each month.	The system displays the corresponding reports and forms.
	The accountant selects a report or form to view the details. The accountant can export the report or form to a file for storage or print it out for use.	The system displays the detailed information in the report or form.
Exit Condition	The accountant has viewed the statistics information and can use the necessary reports or forms.	
Exceptions	If there is no suitable data to display in the report or form, the system will display an error message to the accountant. If the accountant does not have access to the relevant reports or forms, the system will display an error message.	
Special Requirements	The system needs to provide all necessary reports and forms so that the accountant can easily analyze the statistics information. The system needs to ensure data security and only allow the accountant to access reports and forms related to their job.	

- ii. Activity diagram for Use Case: Statistics

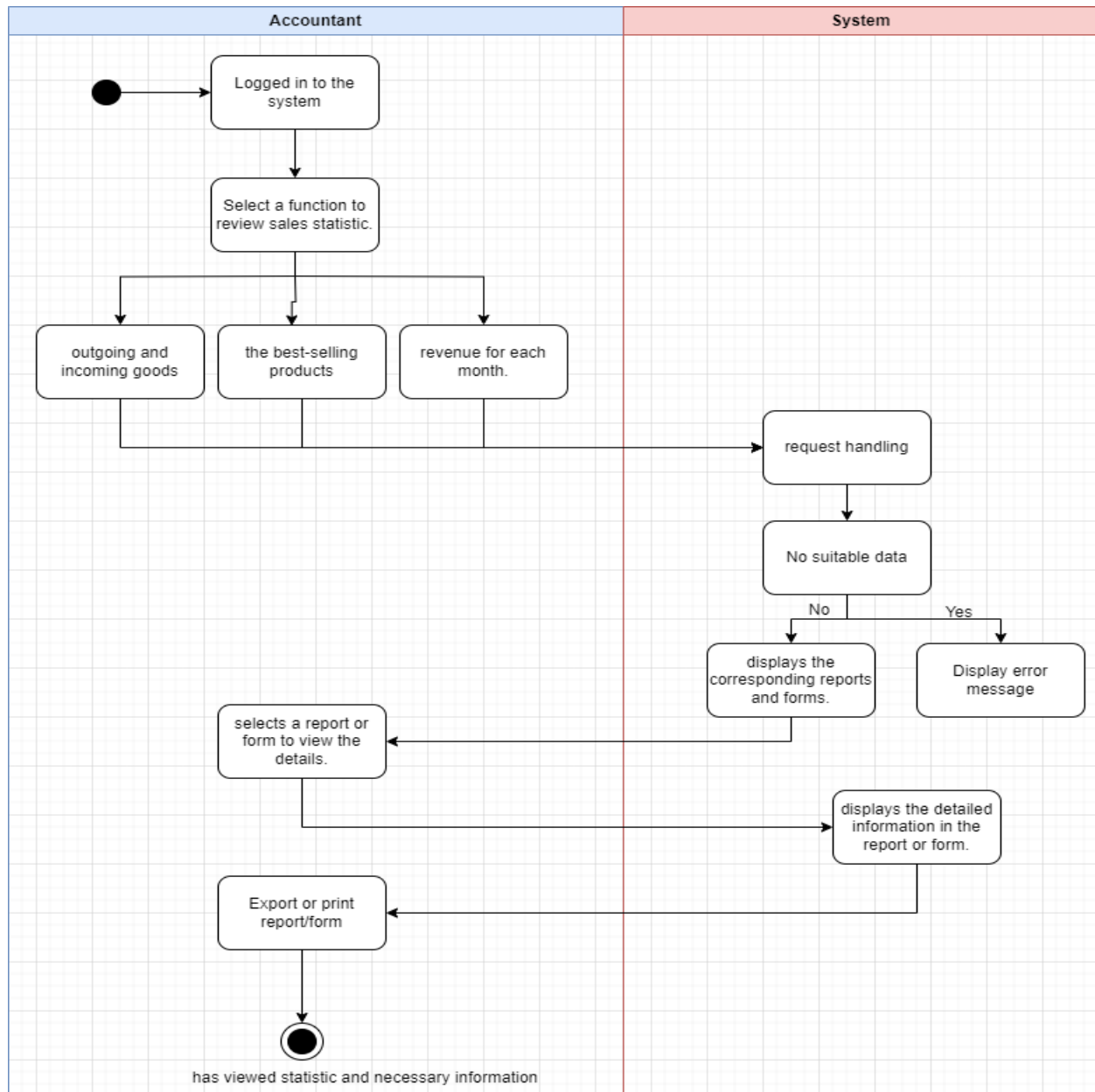


Figure 8 Activity diagram for Use Case: Statistics

iii. System sequence diagram for Use Case: Statistics

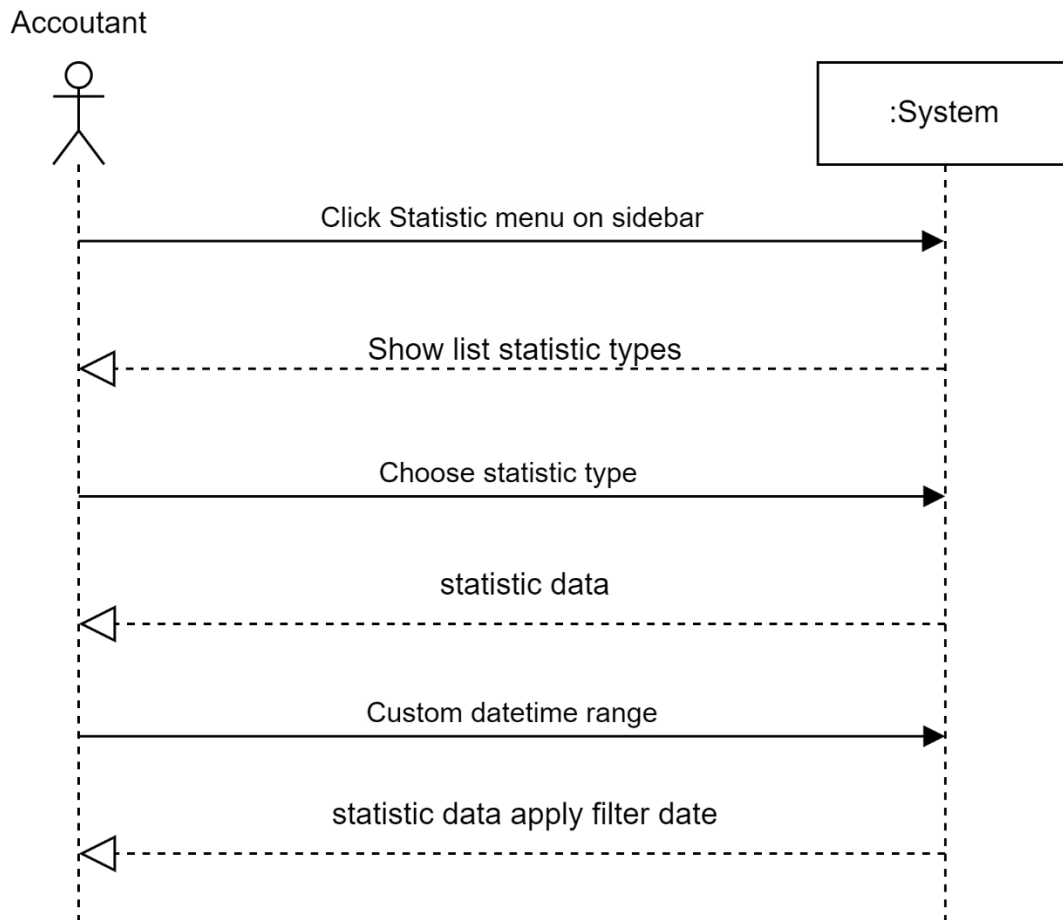


Figure 9 System sequence diagram for Use Case: Statistics

2.2.3 Use Case: Manage shopping cart

i. Use case: Make shopping cart fully description

Use case name	Manage shopping cart	
Participating Actors	Customer.	
Entry Condition(s)	The customer has logged in to the e-commerce website and is browsing products.	
The flow of Events	User	System

	The customer selects a product and clicks the "Add to Cart" button.	The system adds the selected product to the customer's shopping cart and displays the updated cart information to the customer.
	The customer can increase or decrease the quantity of the product in the cart by using the + and - buttons or entering a number in the quantity field.	The system updates the cart information accordingly and displays the updated information to the customer.
	The customer can continue shopping and repeat steps 1-4 to add more products to the cart.	
Exit Condition	The customer has completed their shopping and placed an order.	
Exceptions	<p>If the customer enters incorrect information in the checkout form, the system will display an error message and ask the customer to correct the information.</p> <p>If the customer tries to add a product that is out of stock to the cart, the system will display an error message and ask the customer to remove the product or wait until it is back in stock.</p>	
Special Requirements	<p>The system needs to update the inventory in real-time to ensure accurate product availability information.</p> <p>The system needs to calculate and display the total price of the products in the cart, including taxes and shipping costs.</p> <p>The system needs to allow the customer to remove products from the cart if they change their mind or if the product is no longer needed.</p>	

ii. Activity diagram for Use Case: Make shopping cart

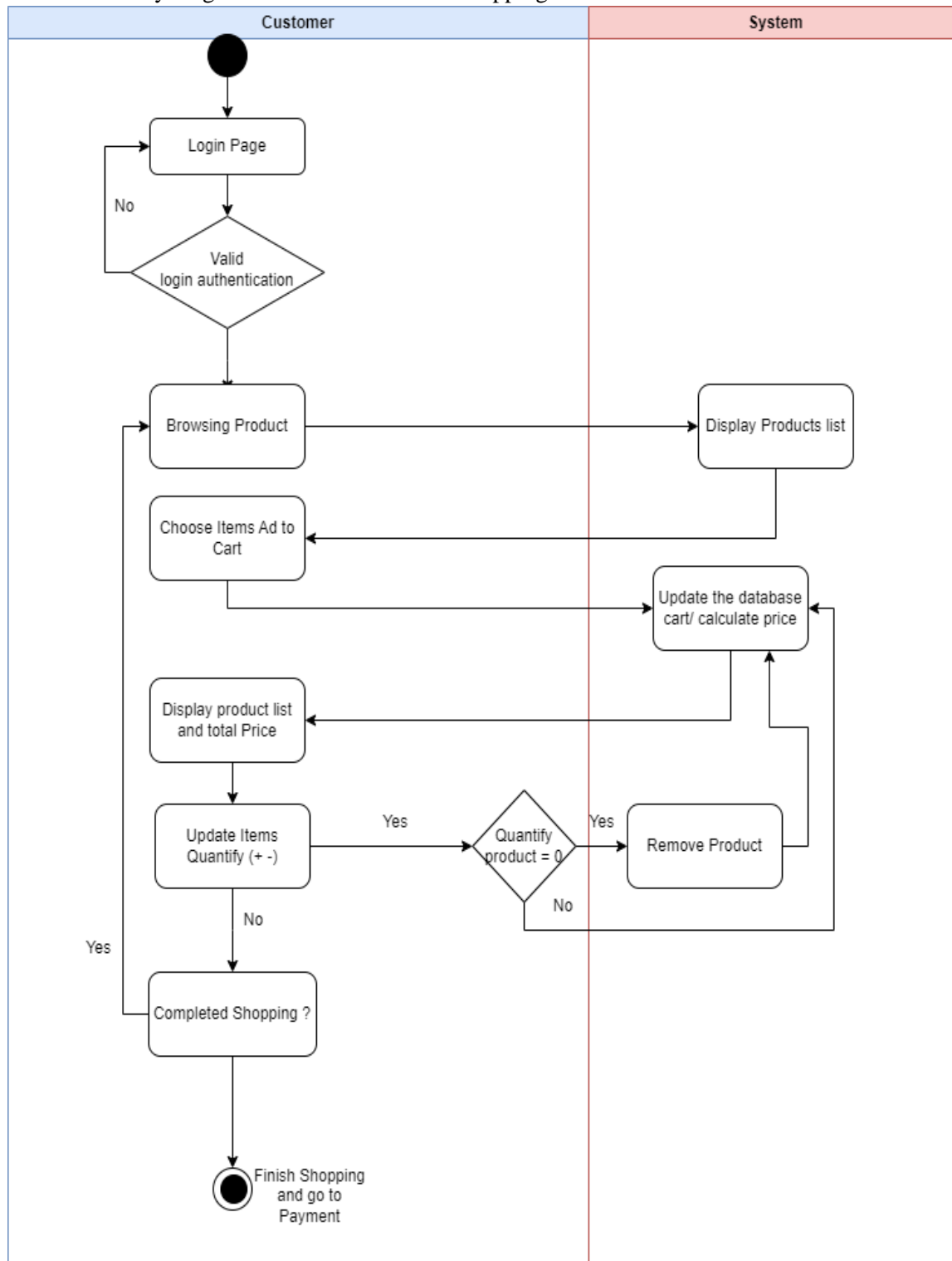


Figure 10 Activity diagram for Use Case: Make shopping cart

- iii. System sequence diagram for Use Case: Make shopping cart

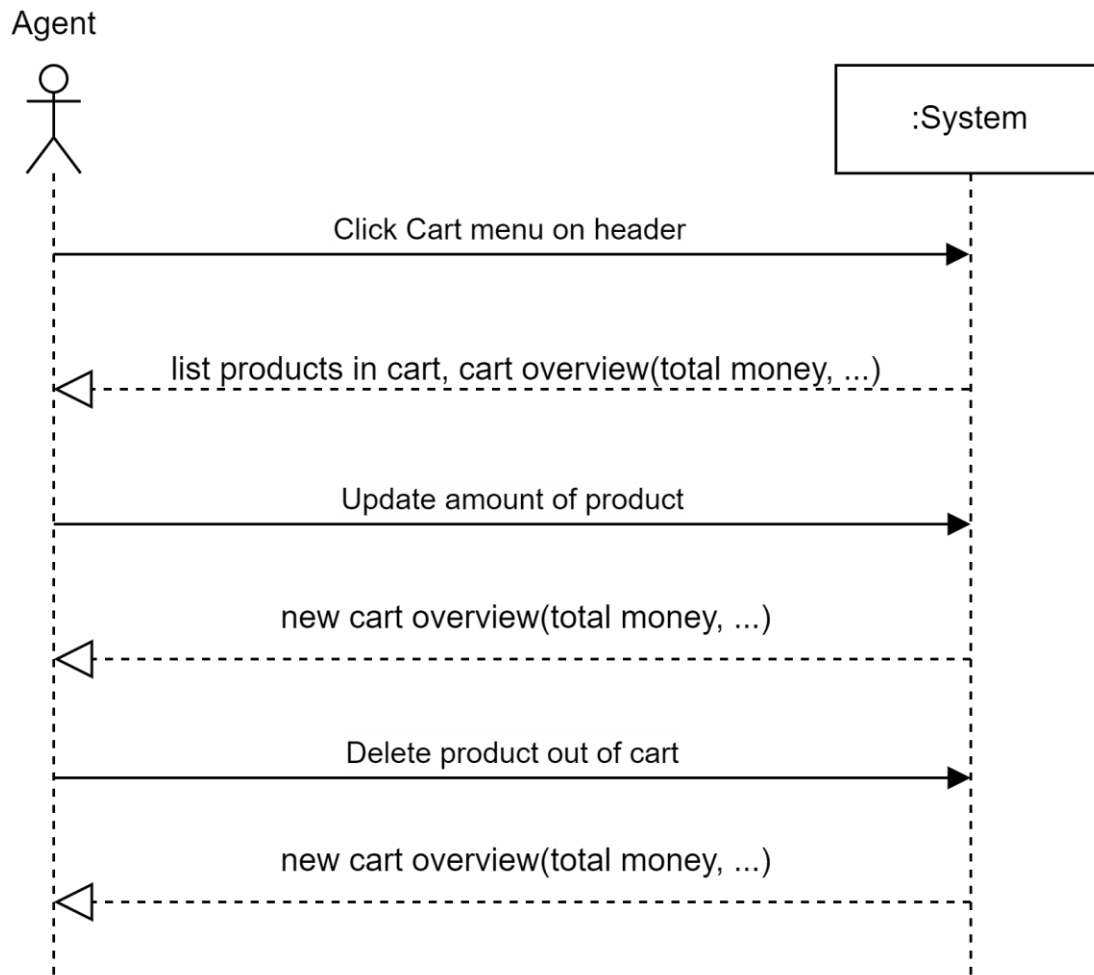


Figure 11 System sequence diagram for Use Case: Make shopping cart

2.3 Verifying use cases for Agent.

2.3.1 Verifying uses cases for Manage Order

Data entity/domain class	C R U D	Verified use case
Order	Create	View All
	Read/report	The accountant looks up the details of a specific order
Product	Update	The accountant updates payment status, order status
	Delete	The accountant cancels an order

2.3.2 Verifying uses cases for Statistics.

Data entity/domain class	C R U D	Verified use case
Sales	Create	The accountant can view the revenue report for a specific time period, showing the total revenue, revenue by product, and revenue by reseller/agent. The accountant can view the best-selling products for a specific time period, including the quantity sold and the revenue generated.

2.3.3 Verifying uses cases for Manage Shopping Cart

Data entity/domain class	C R U D	Verified use case
Cart	Create	The agent can add products to their shopping cart by clicking on the "Add to Cart" button on the product detail page.
	Read/View	The agent can view their shopping cart at any time to see which products have been added, the quantity of each product, and the total cost of the cart.
Product	Update	The agent can update the quantity of each product in their cart by changing the quantity field and clicking the "Update Cart" button.
	Delete	The agent can remove products from their shopping cart by clicking on the "Remove" button next to the product in the cart. Empty Cart: The agent can empty their entire cart by clicking on the "Empty Cart"

		button. This will remove all products from the cart.
--	--	--

III. System Requirements Design (3.0 points)

3.1 Design Class for Use Case Manage Order

3.1.1 Design Classes in Detailed Design

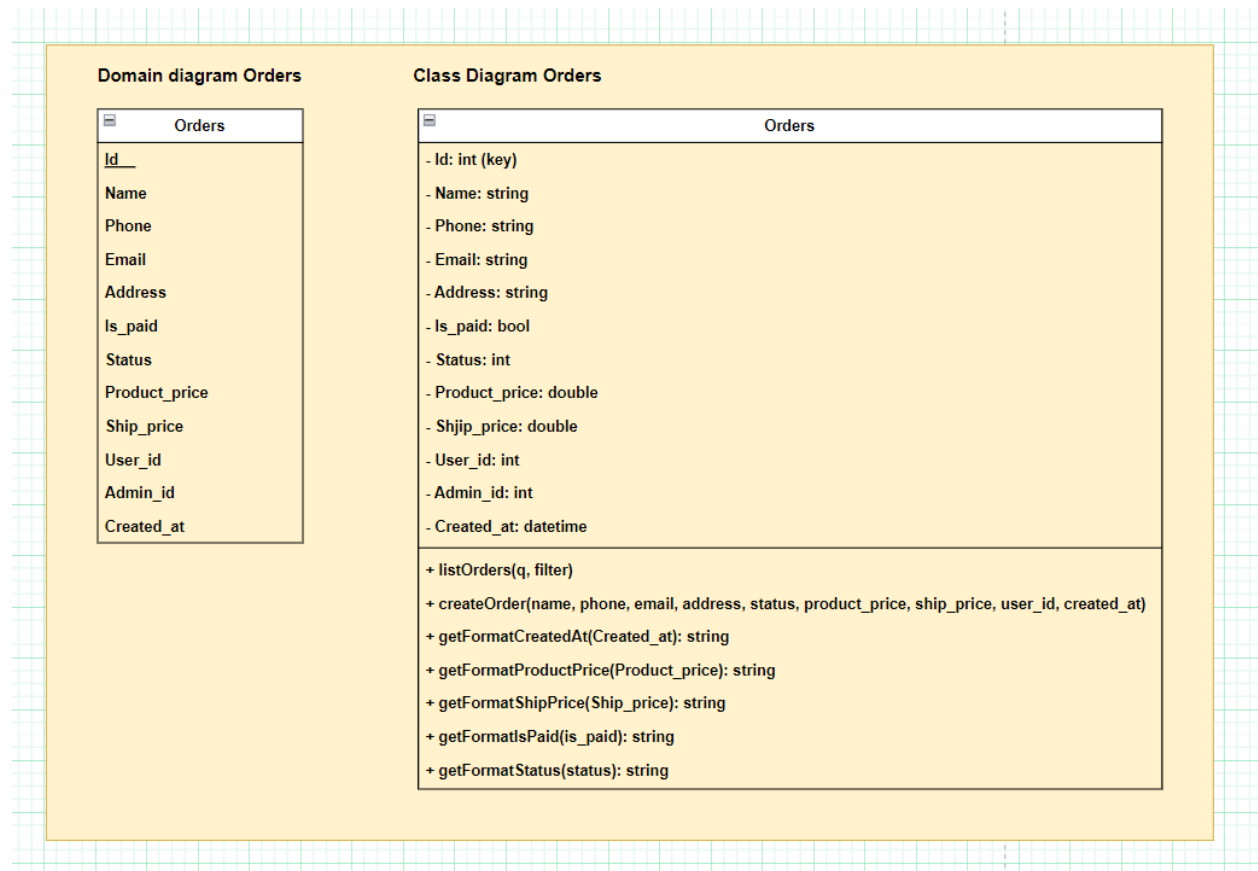


Figure 12 Design Classes in Detailed Design

3.1.2 Design Class Diagram

Domain Design Class

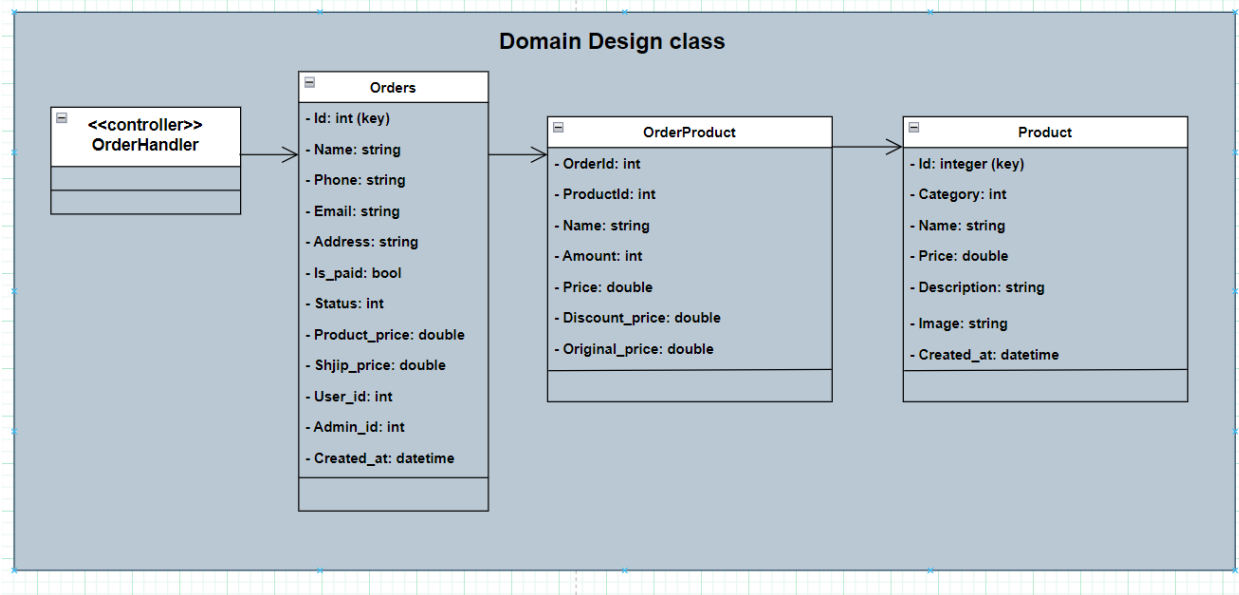


Figure 13 Domain Design Class

Controller

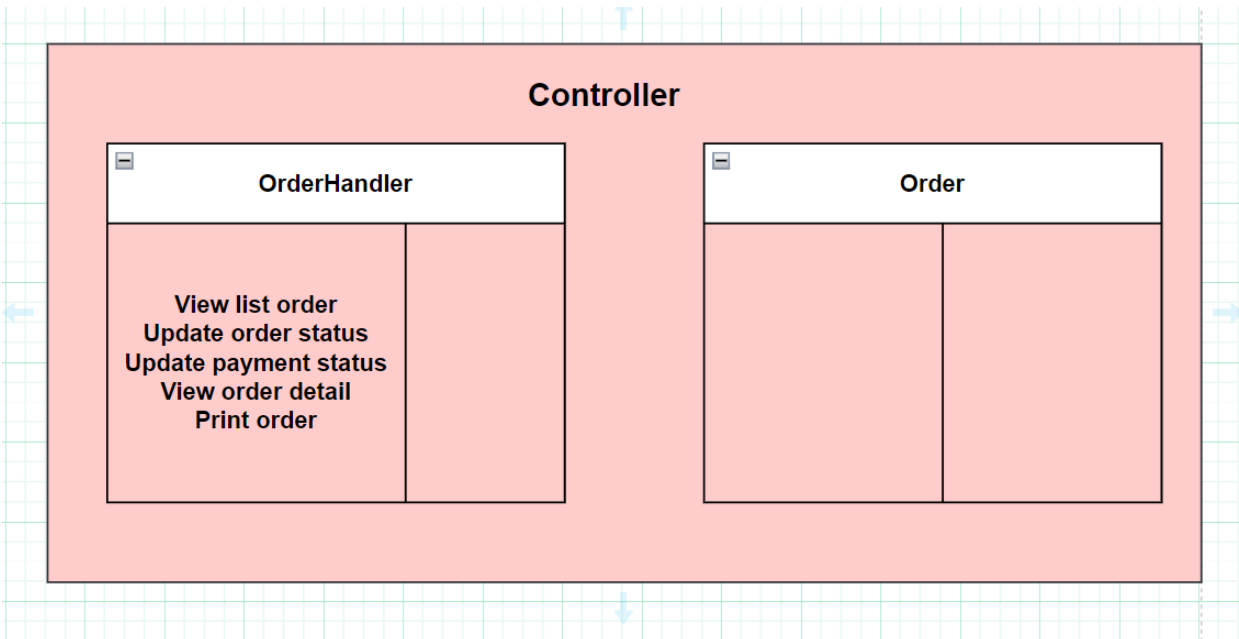


Figure 14 Controller

UI

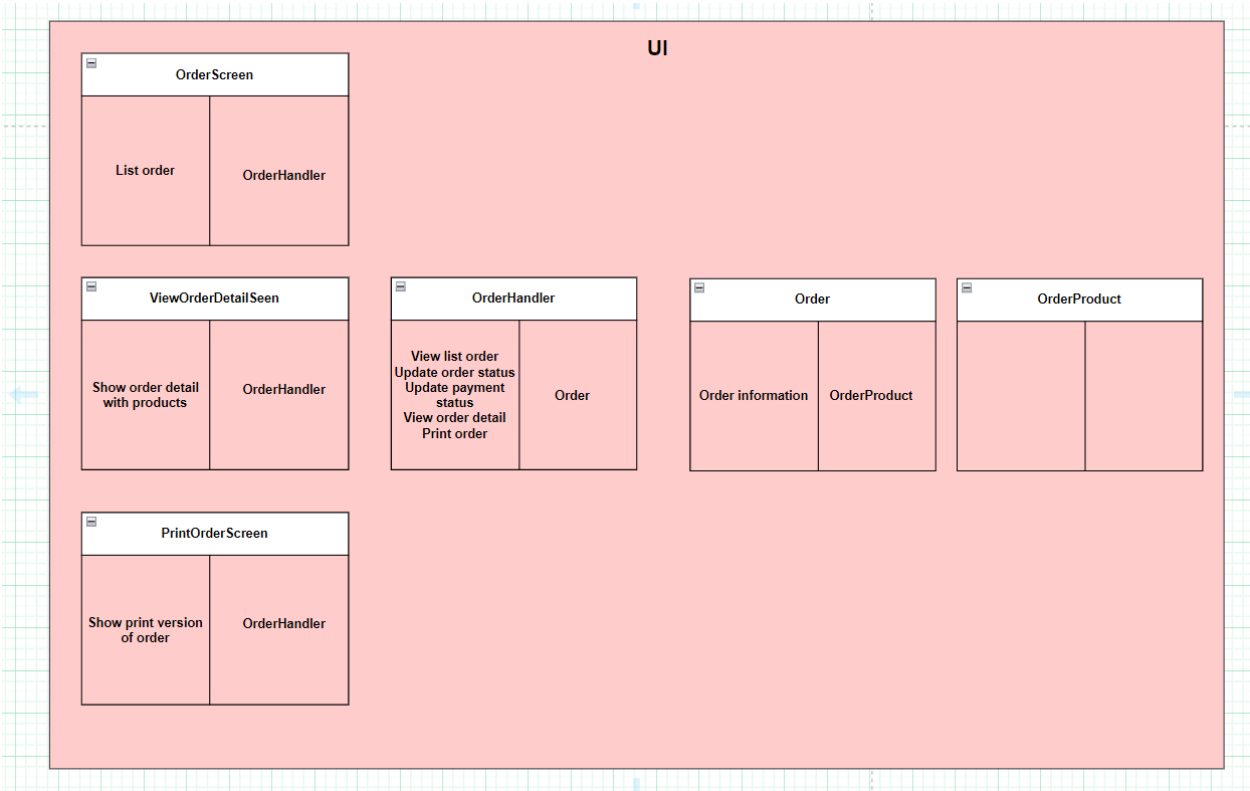


Figure 15 UI

Data Access

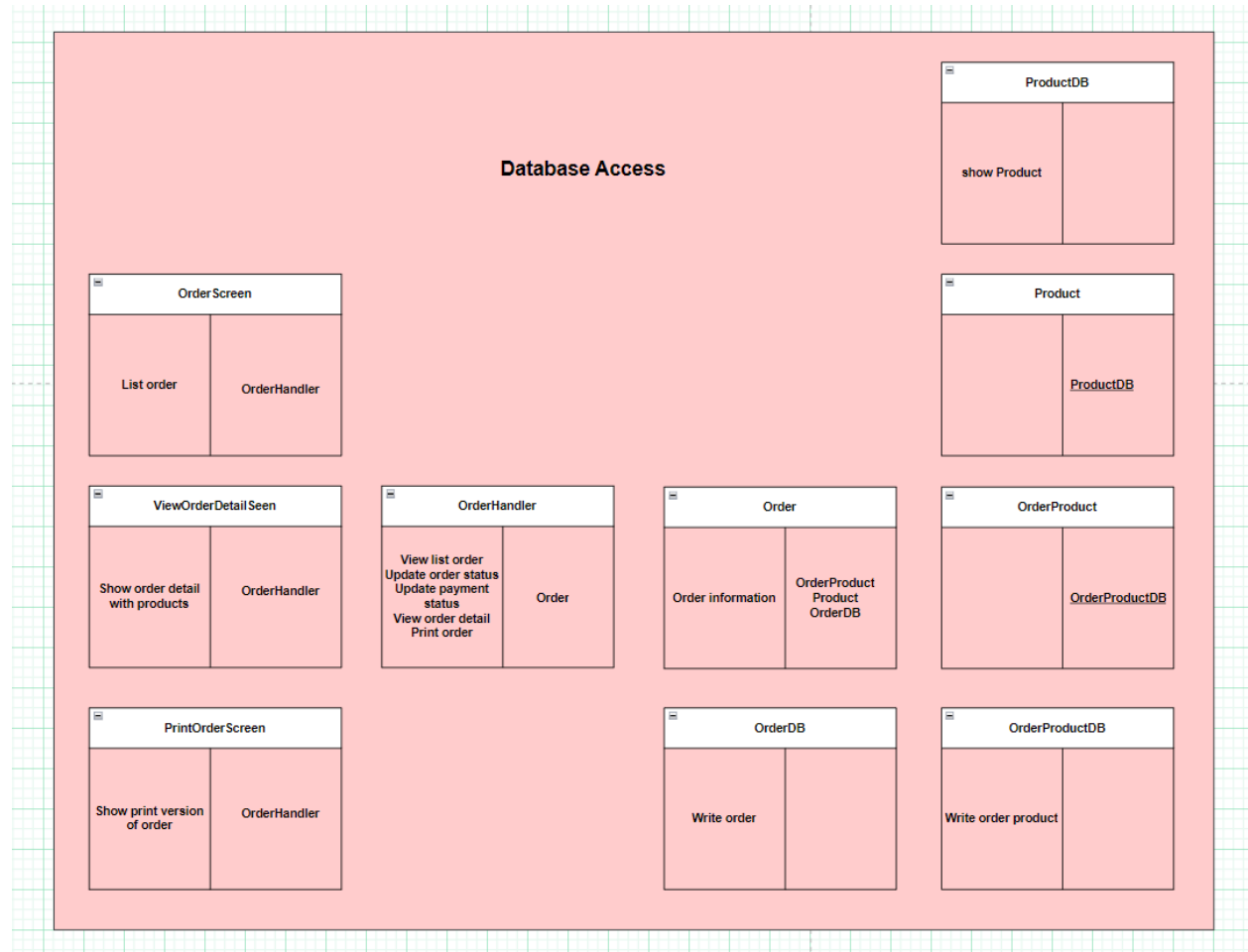


Figure 16 Data access

Design Class

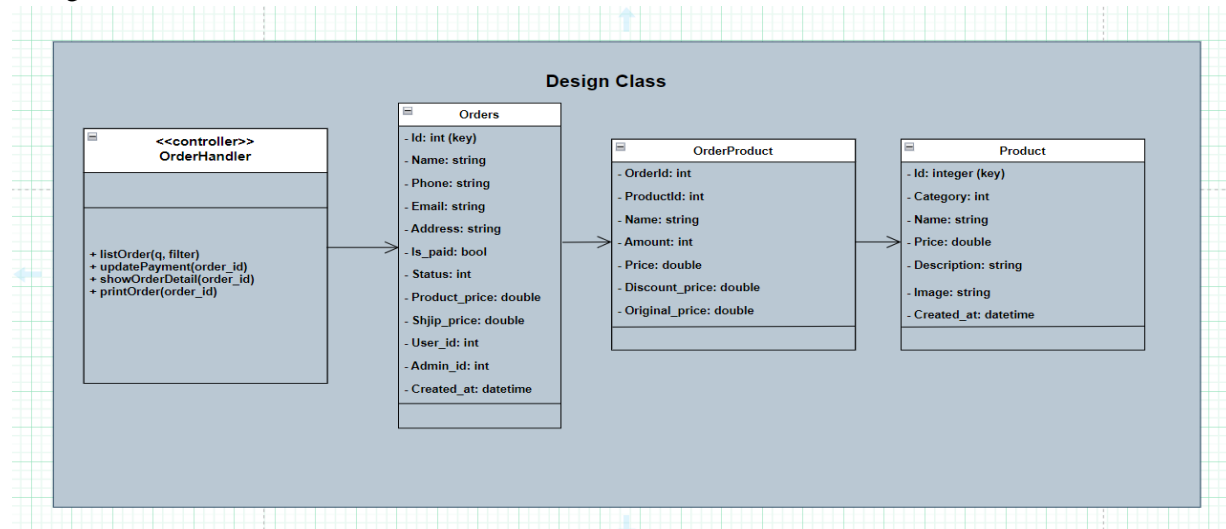
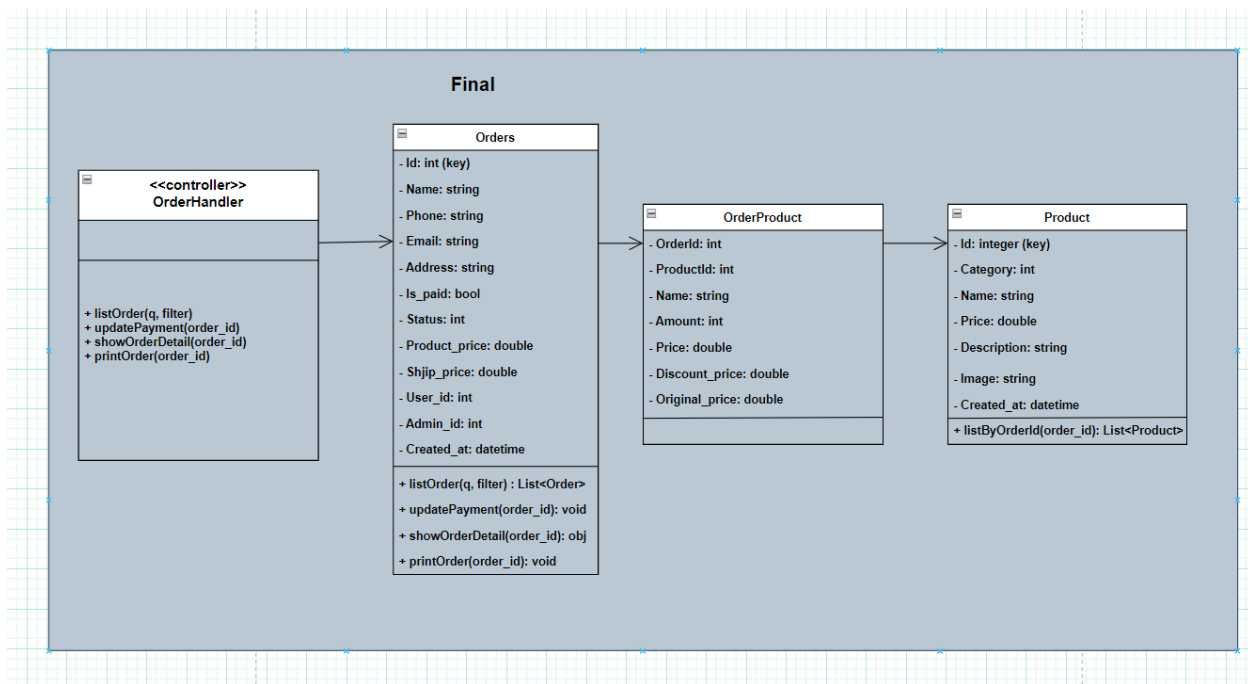


Figure 17 Design class

3.1.3 Final Design Class Diagram



3.2 Design Class for Use Case Statistic

3.2.1 Design Classes in Detailed Design

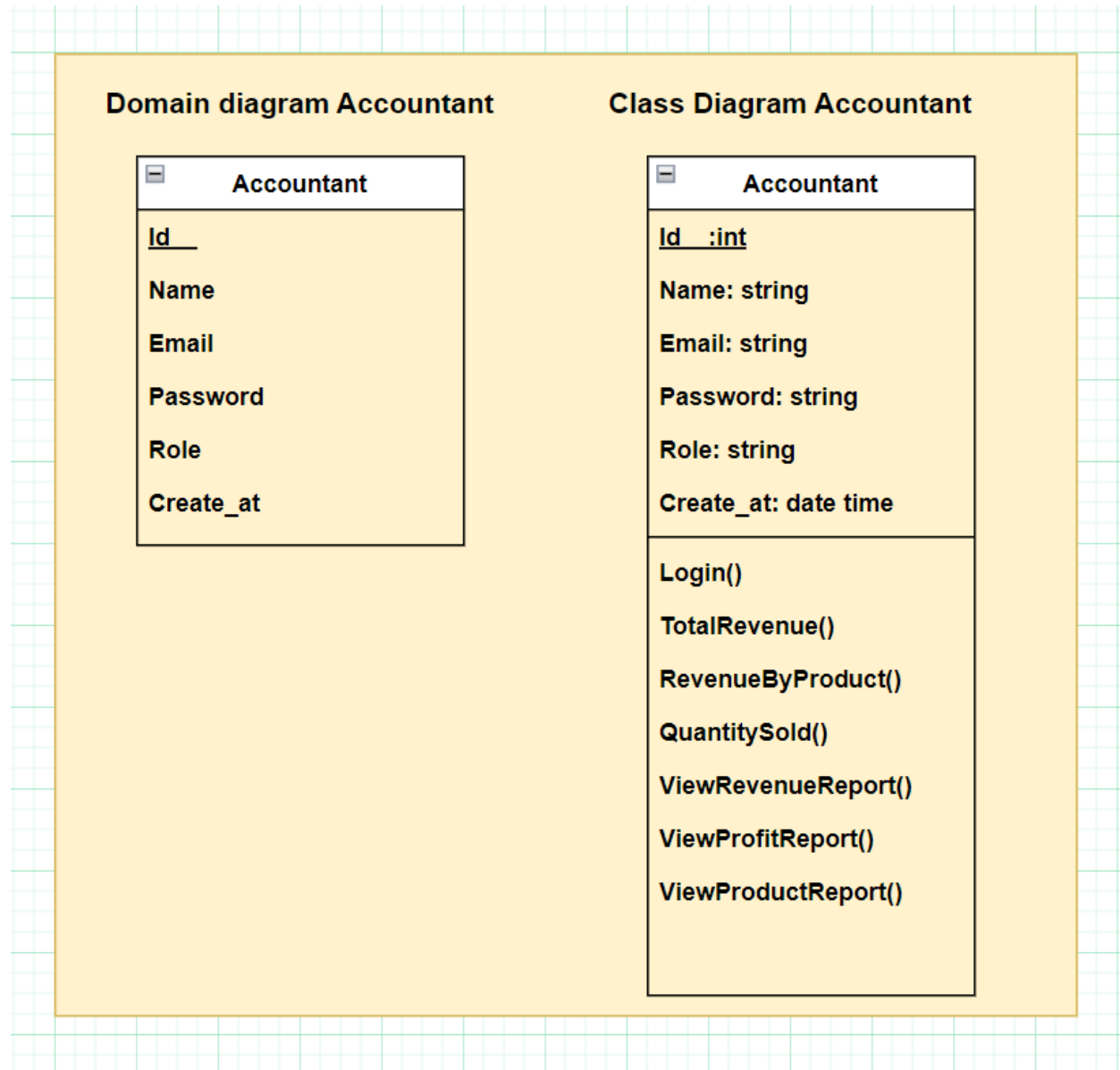


Figure 18 Accountant class

3.2.2 Design Class Diagram

Domain Design Class

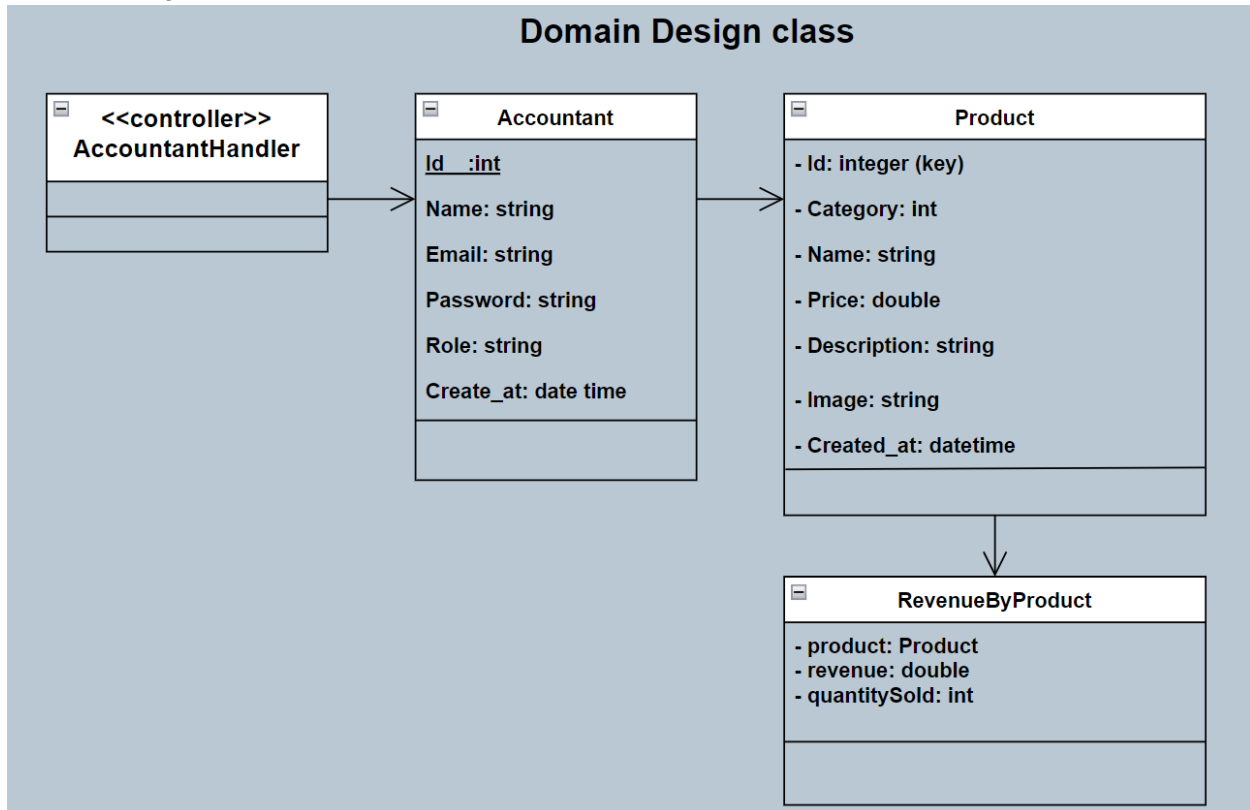


Figure 19 Domain Design Class

Controller

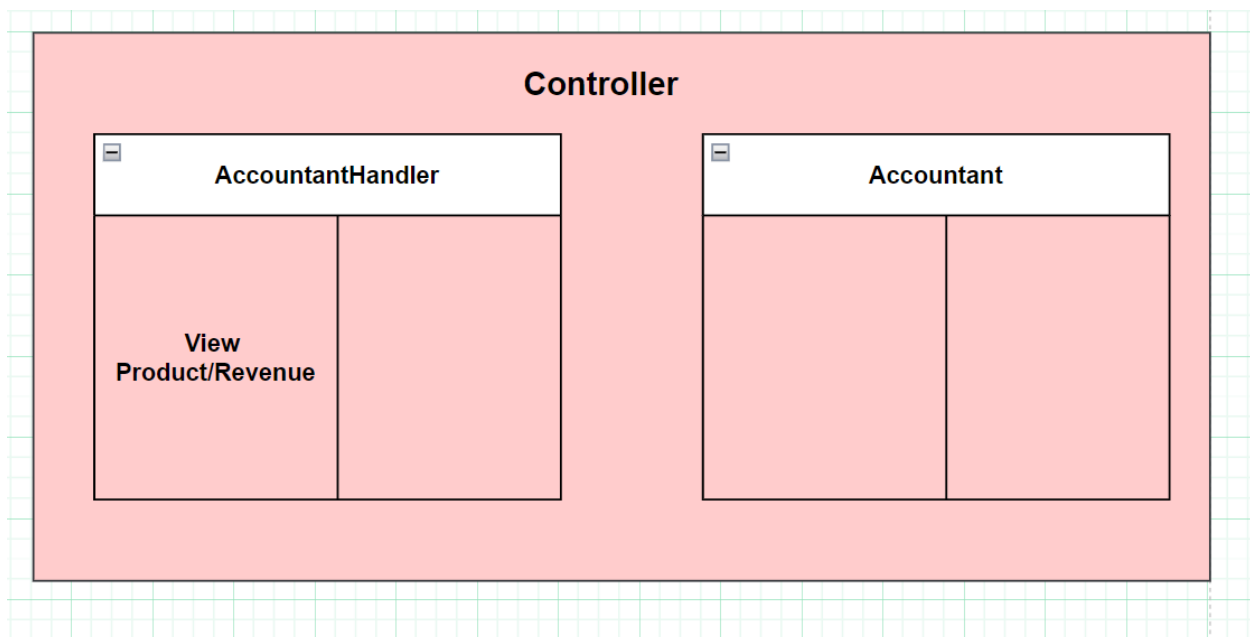


Figure 20 use case statistic Controller

UI

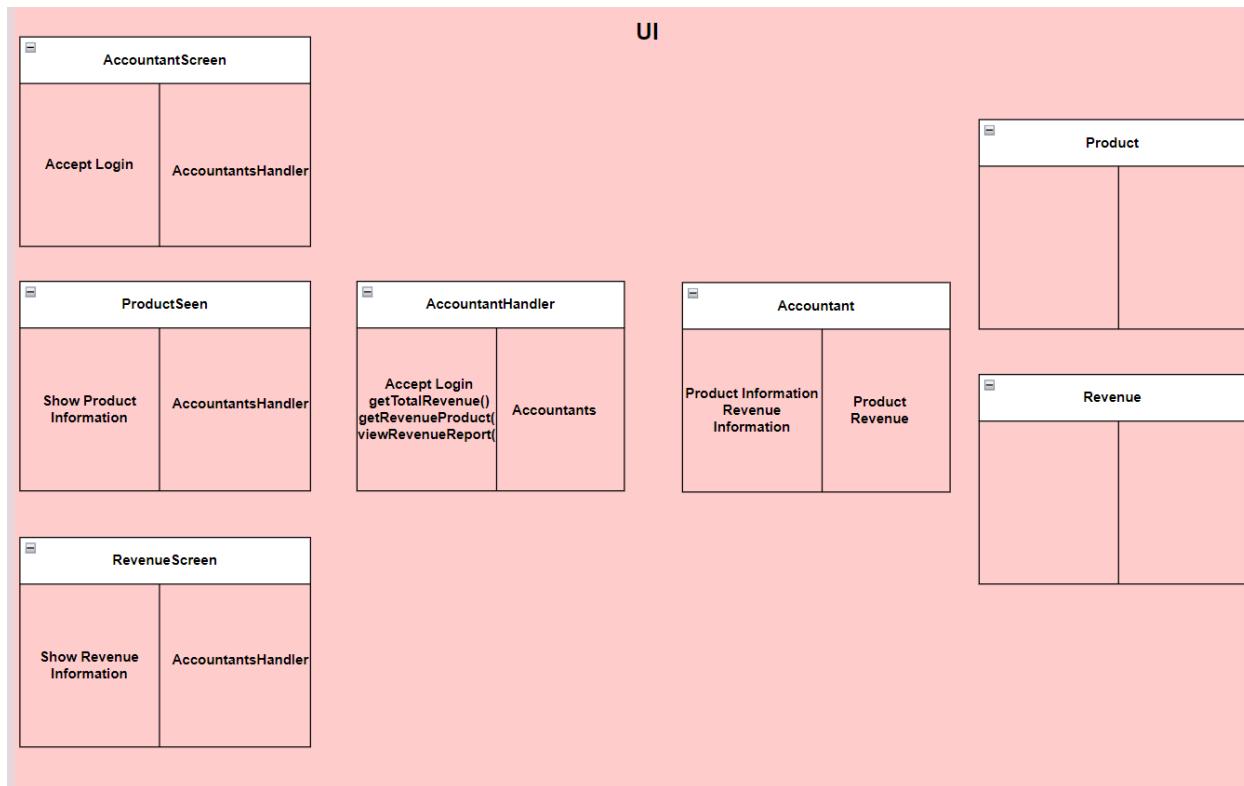


Figure 21 UI statistic

Data Access

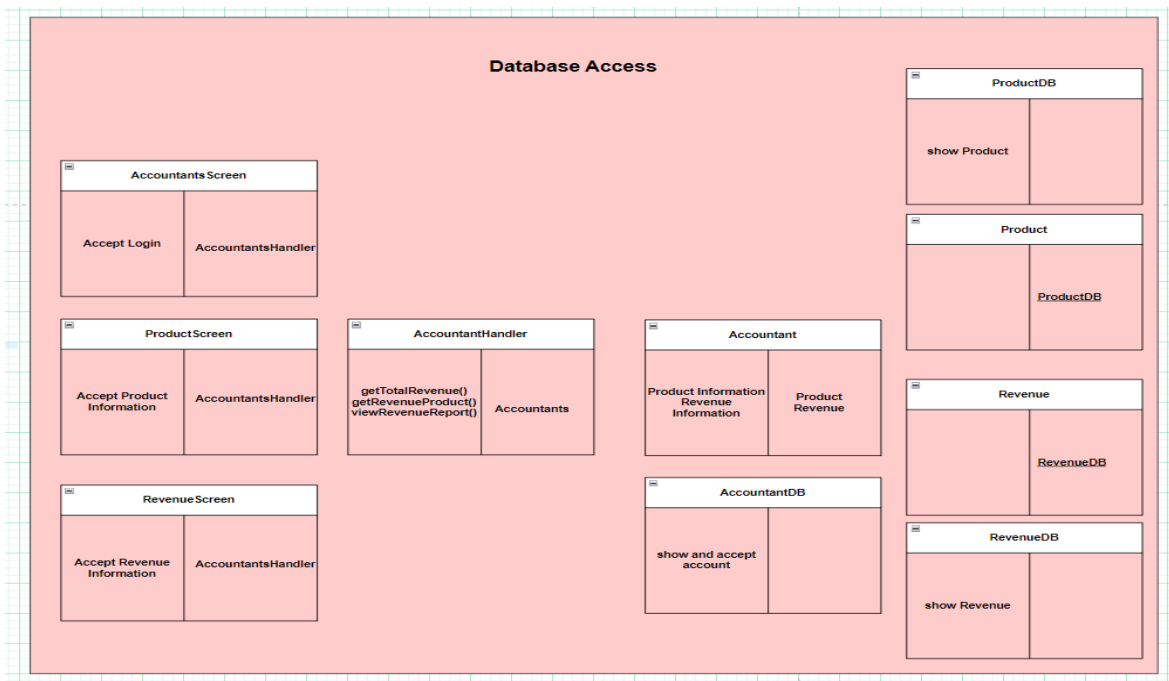
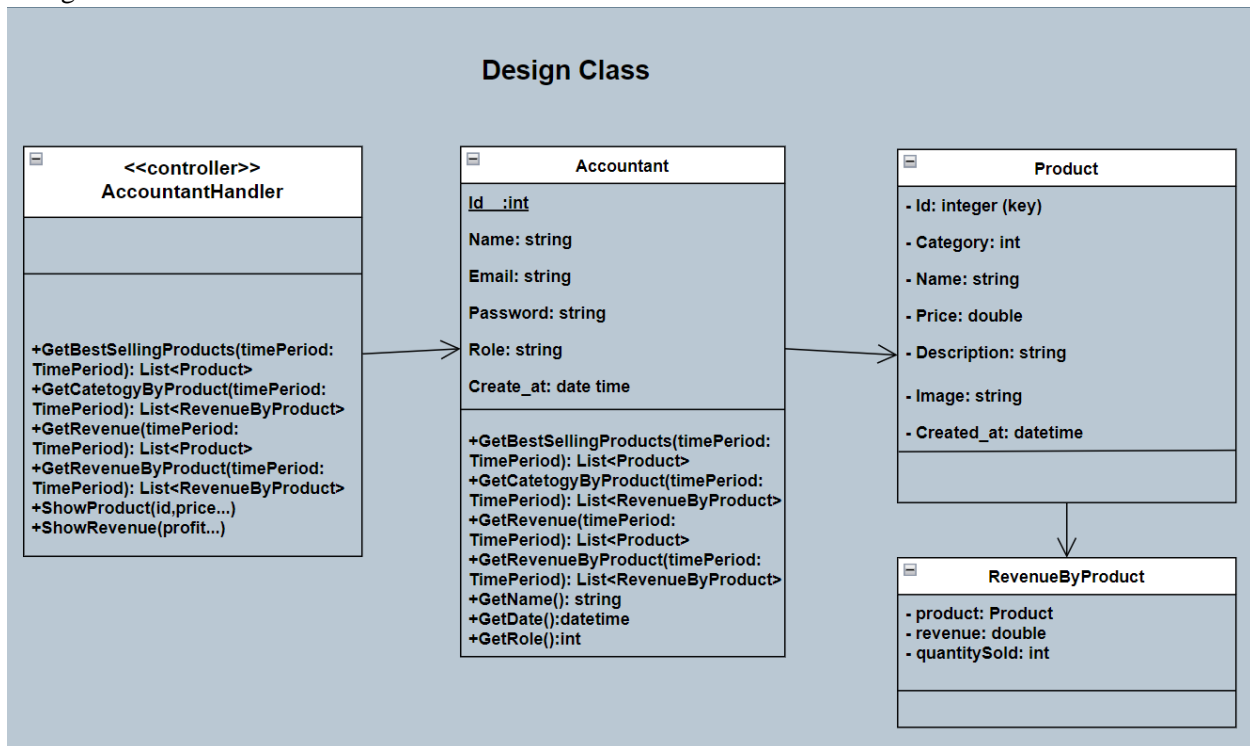


Figure 22 database access

Design Class



3.2.3 OOD with Communication

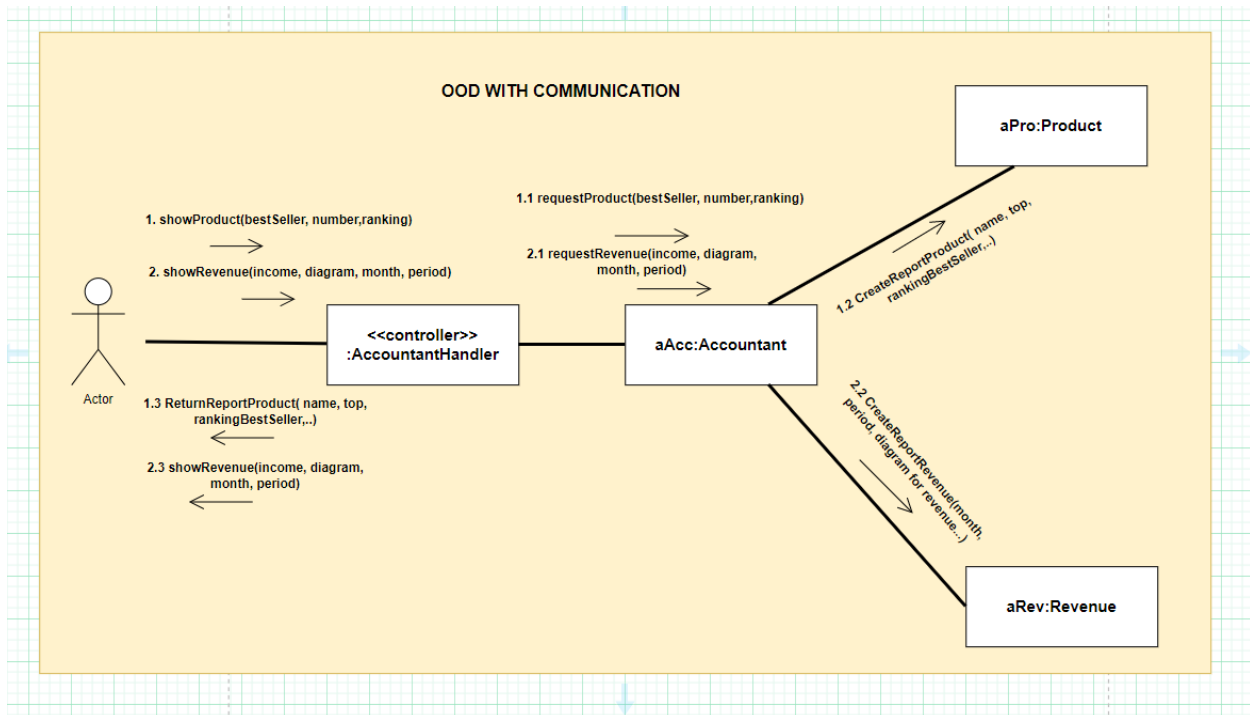


Figure 23 OOD with Communication for use case 2

3.2.3 Final Design Class Diagram

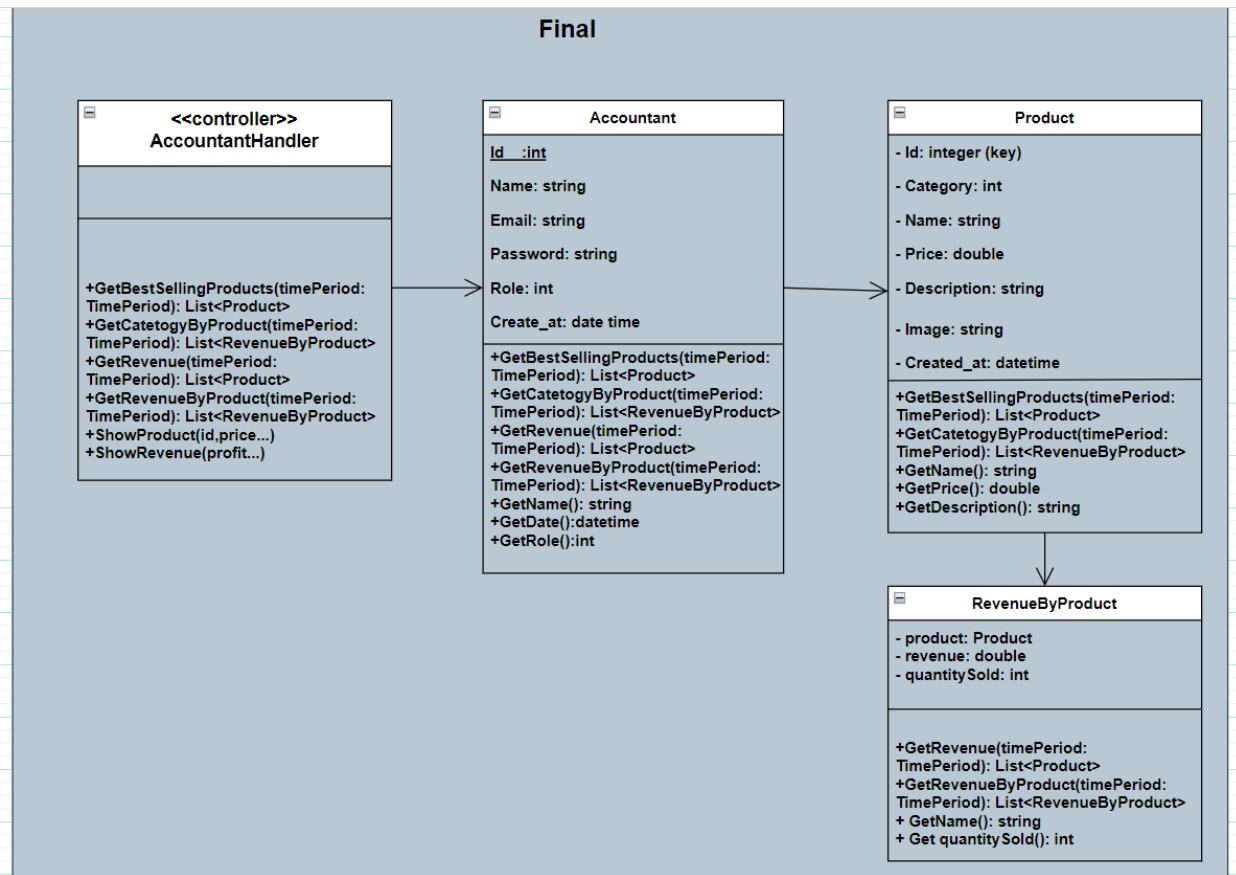


Figure 24 3 Final Design Class Diagram use case 2

3.3 Design Class for Use Case Manage Shopping Cart

3.3.1 Design Classes in Detailed Design

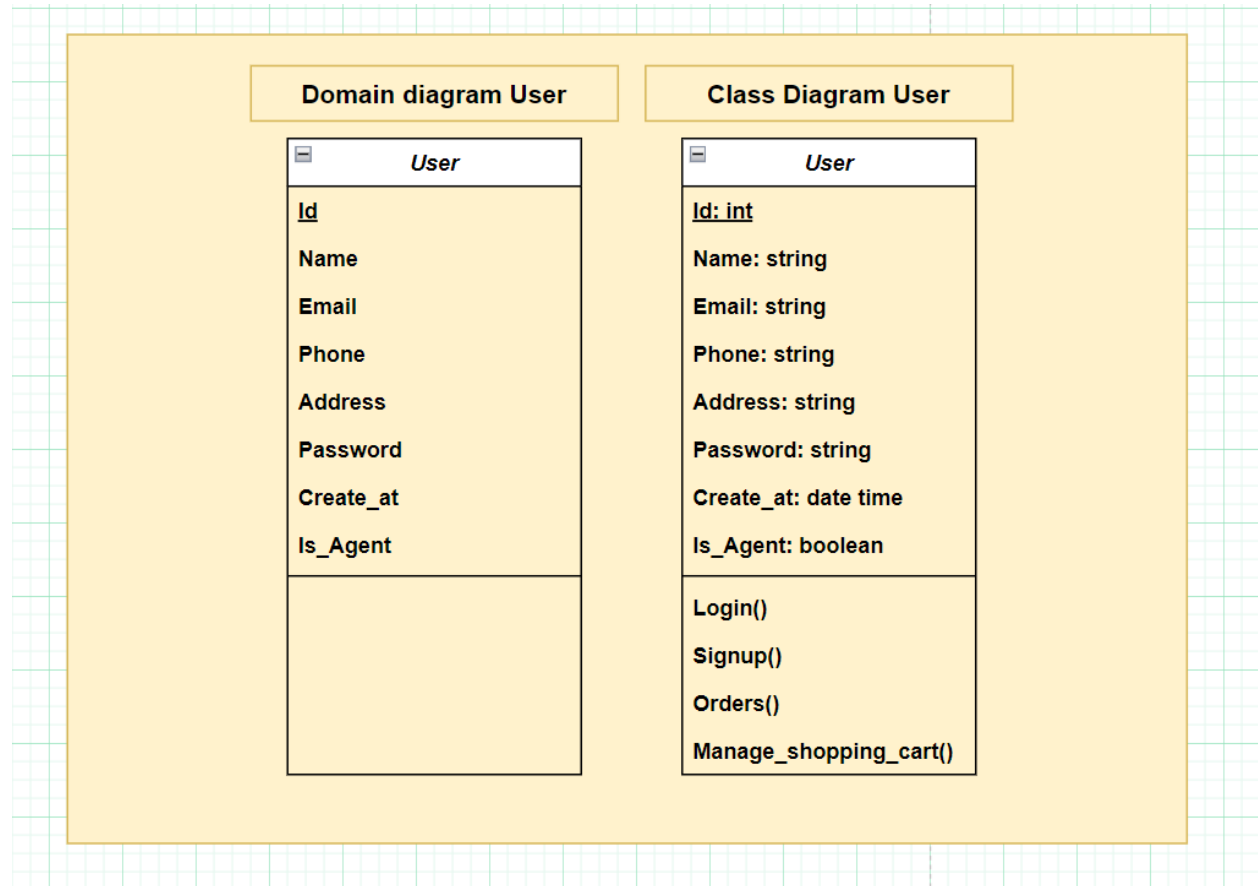


Figure 25 Design Classes in Detailed Design use case 3

3.3.2 Design Class Diagram

i. Domain Design Class

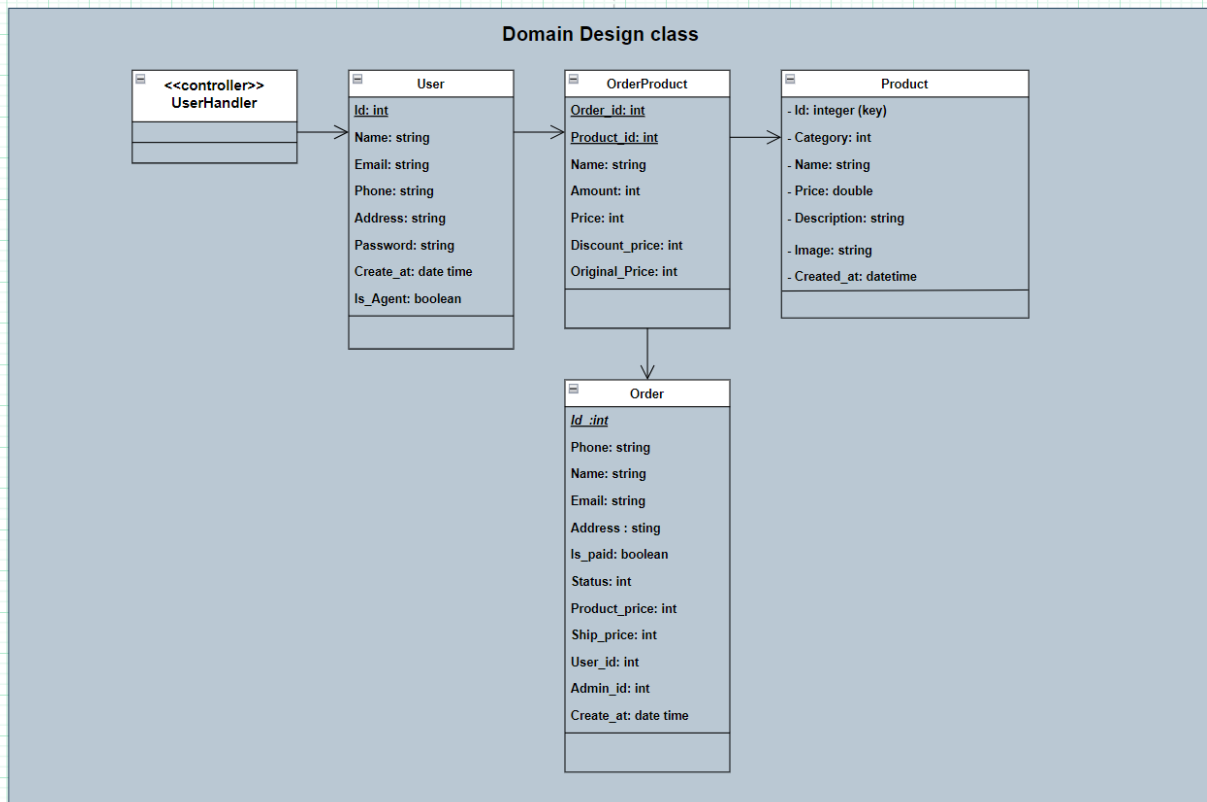


Figure 26 . Domain Design Class

ii. Controller

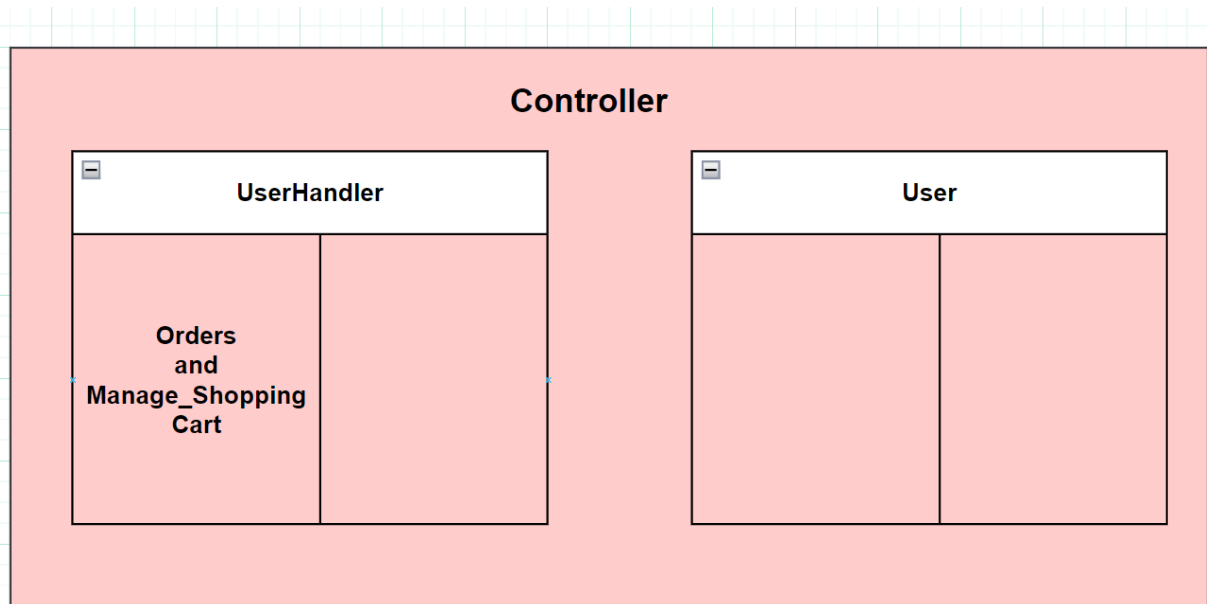


Figure 27 Controller use case 3

iii. UI

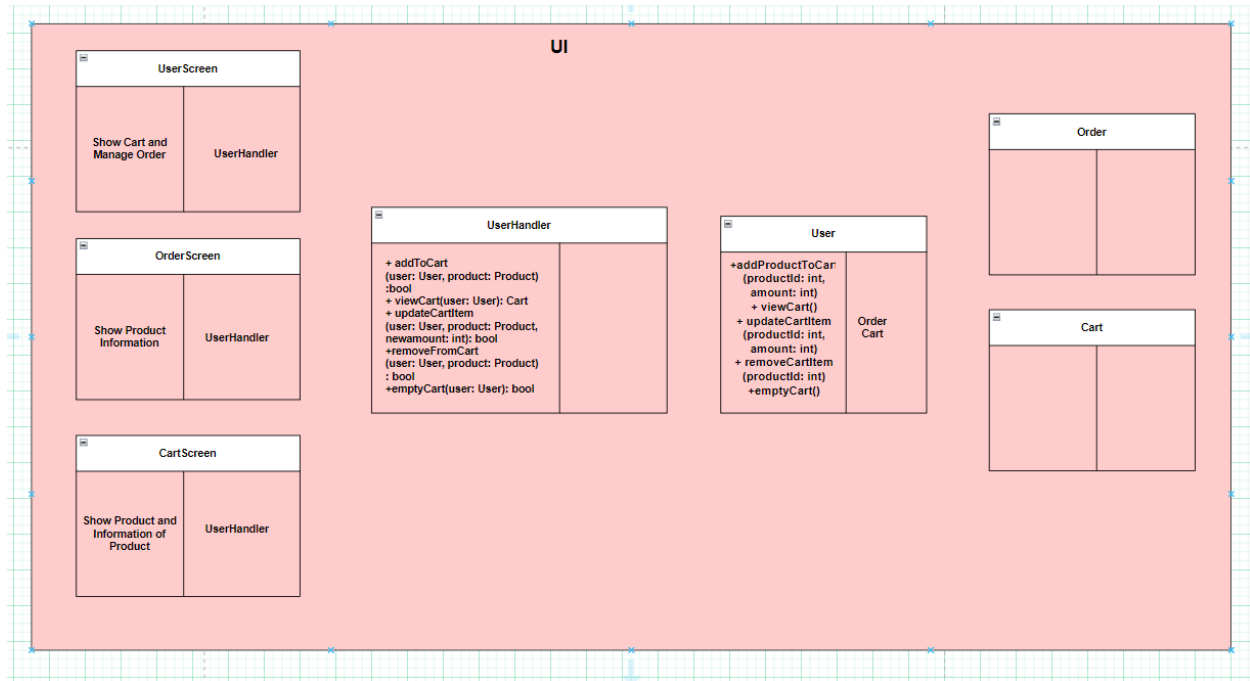


Figure 28 UI use case 3

iv. Data Access

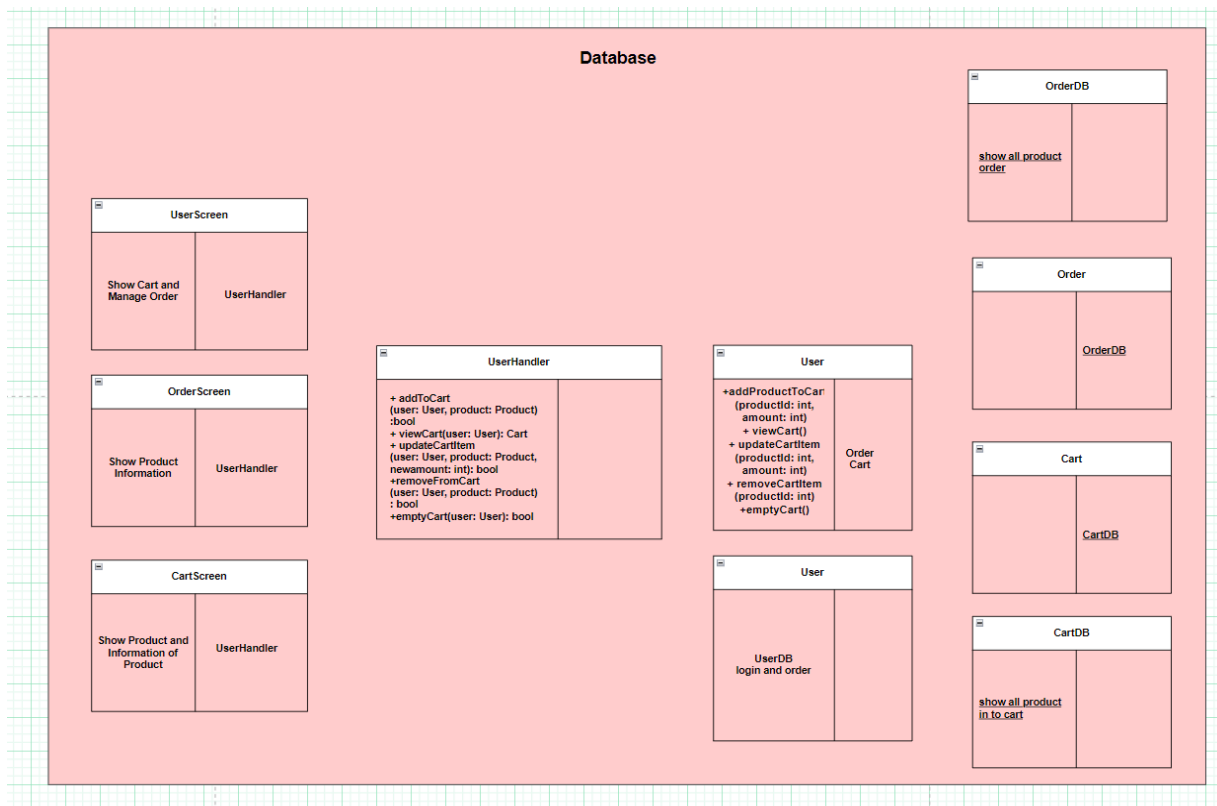
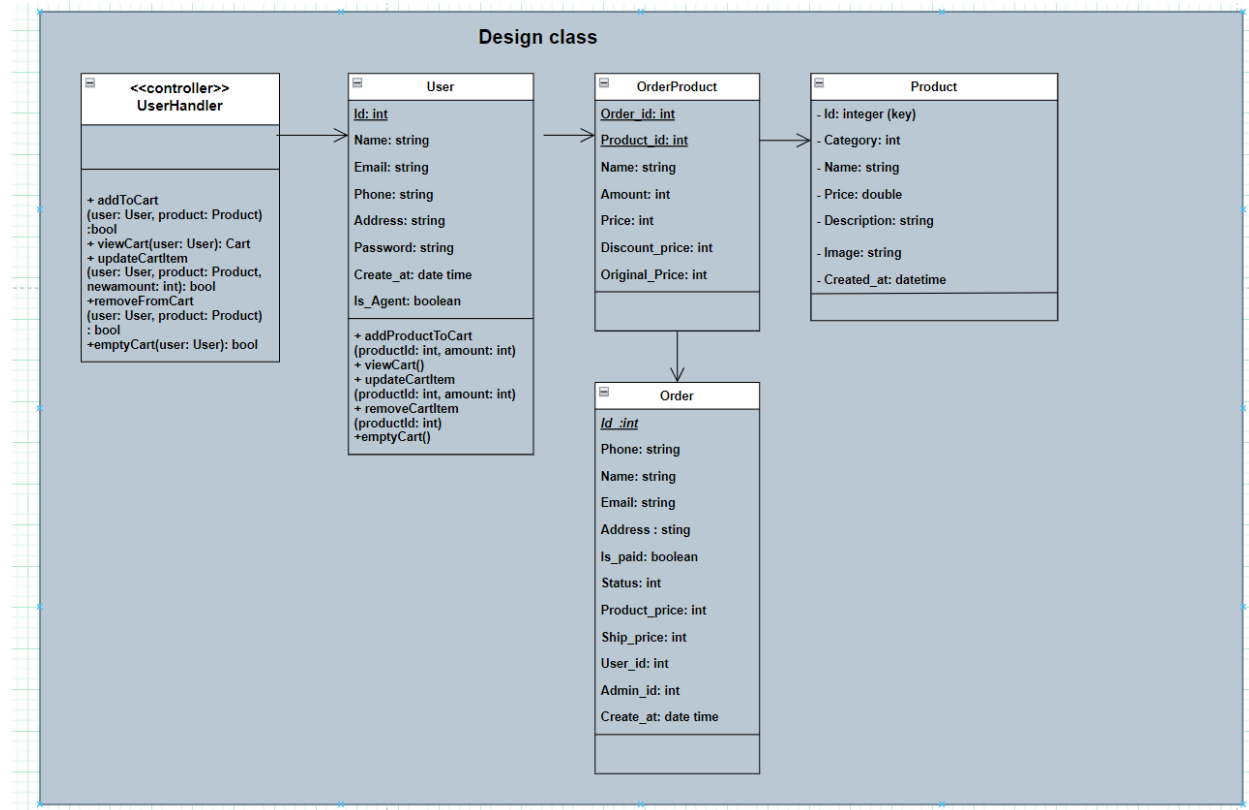


Figure 29 database access

v.Design Class



3.3.2 OOD with Communication

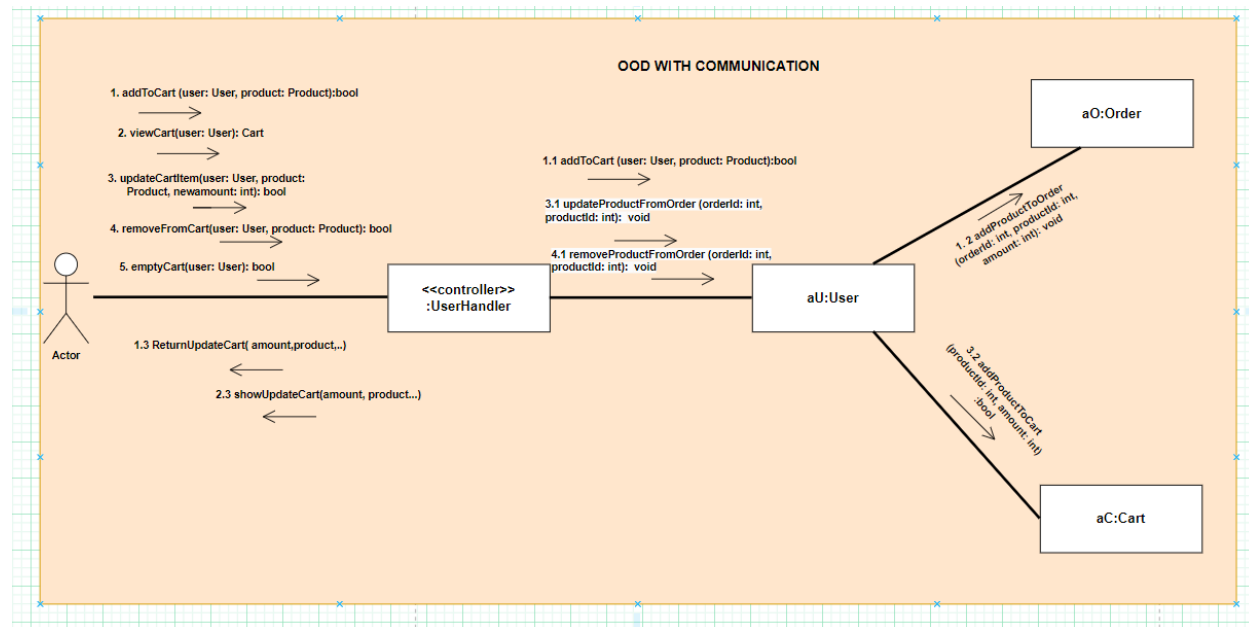


Figure 30 OOD with Communication

3.3.3 Final Design Class Diagram

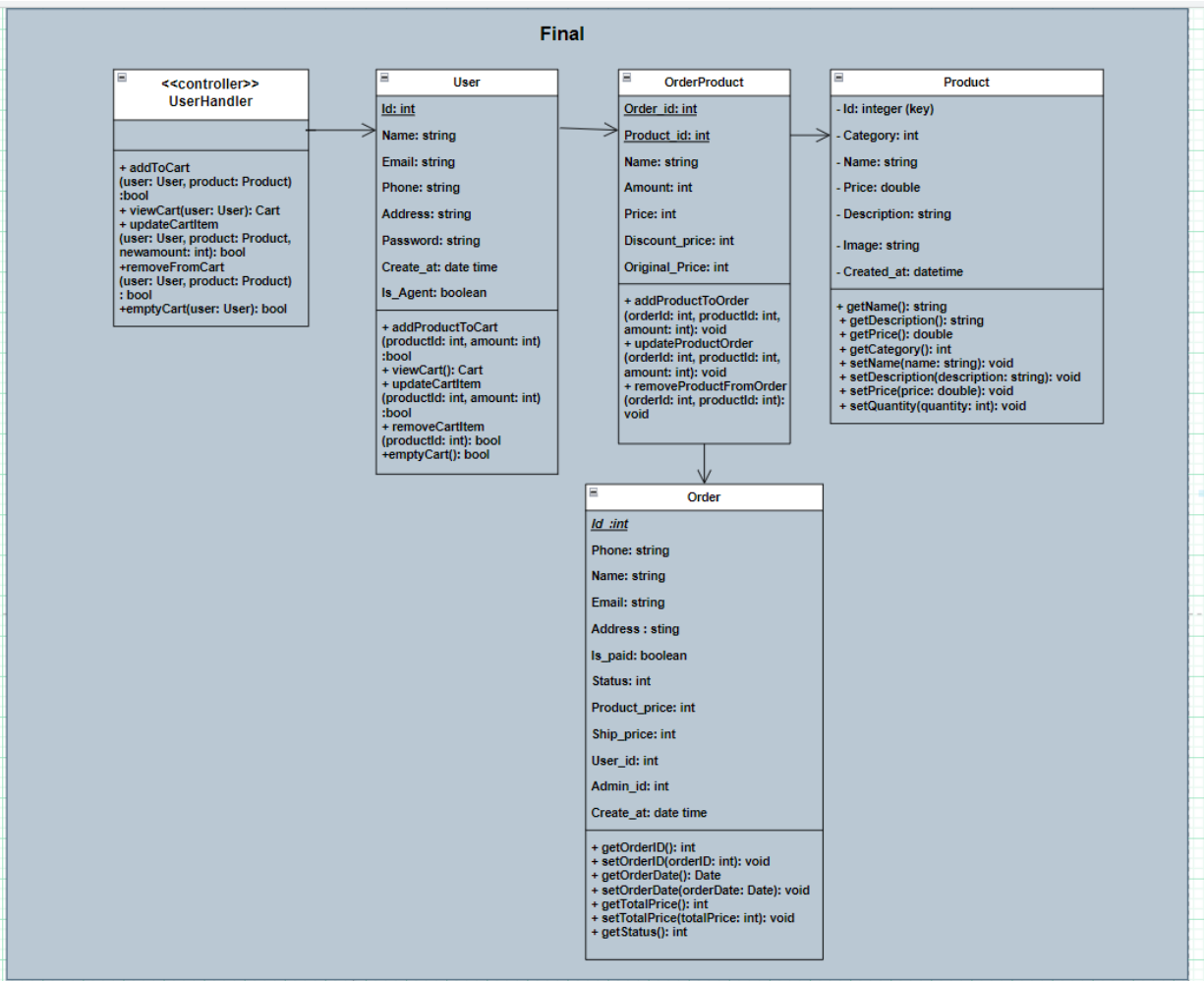


Figure 31 3 Final Design Class Diagram

IV. System Requirements Implementation

4.1 Design Class for Sub System

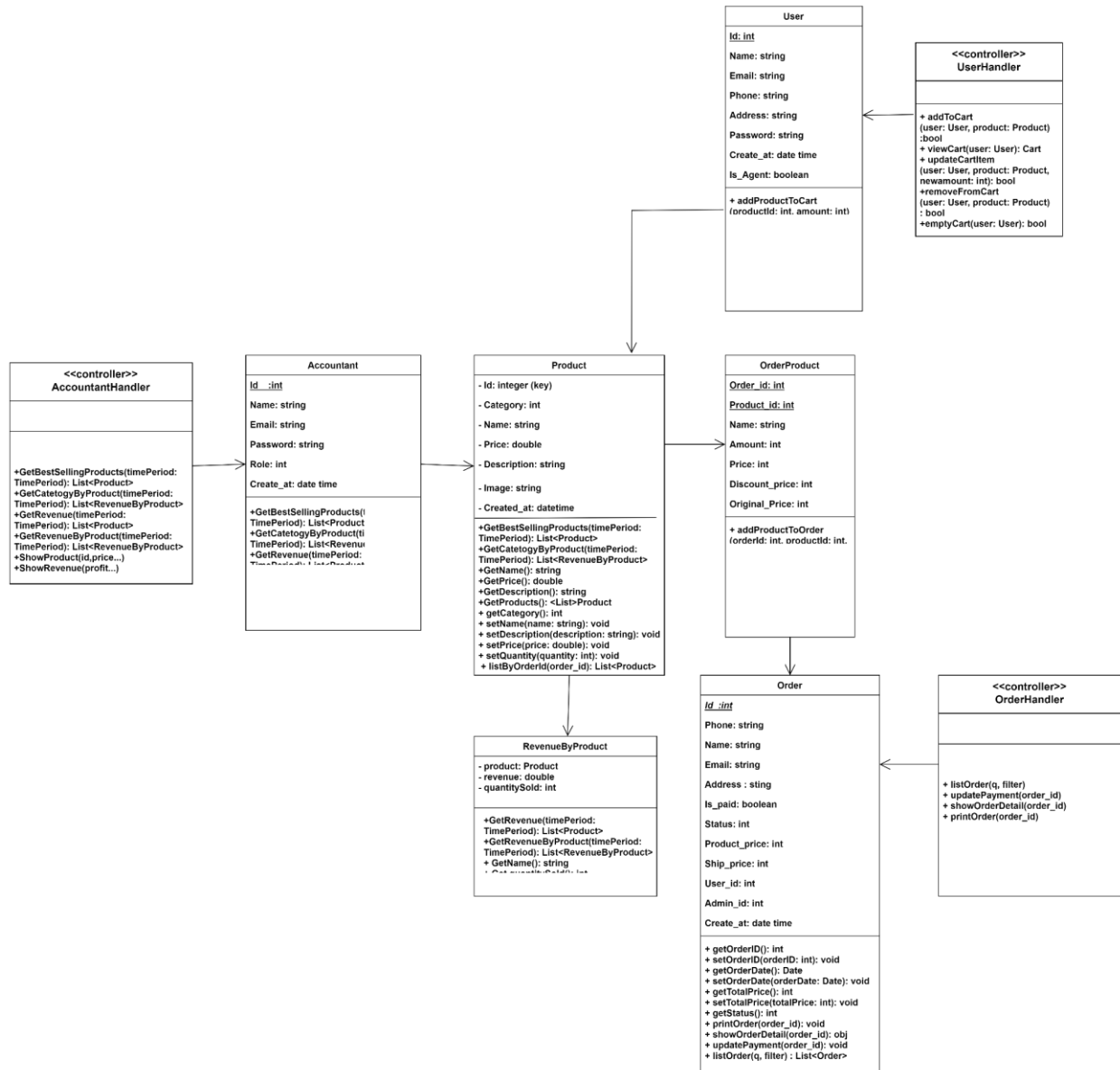


Figure 32 Design Class for Sub System

4.2 Implementation

4.2.1 Map persistent objects to the tables in a database

Product

Product
- Id: integer (key)
- Category: int
- Name: string
- Price: double
- Description: string
- Image: string
- Created_at: datetime
+ GetBestSellingProducts(timePeriod: TimePeriod): List<Product>
+ GetCategoryByProduct(timePeriod: TimePeriod): List<RevenueByProduct>
+ GetName(): string
+ GetPrice(): double
+ GetDescription(): string
+ GetProduct(): <List>Product
+ getCategory(): int
+ setName(name: string): void
+ setDescription(description: string): void
+ setPrice(price: double): void
+ setQuantity(quantity: int): void
+ listByOrder(order_id): List<Product>

id	category	name	price	description	image	created_at
1	2	Film Acc...	5,228,000	Tempore vi...	https://i.ibb...	2023-05-01 22:34:5...
2	3	Film Est ...	6,260,000	Veritatis ne...	https://via....	2022-05-12 10:13:3...
3	3	Film Cu...	8,650,000	Qui quo se...	https://via....	2022-08-01 21:31:5...
4	7	Film Sus...	3,127,000	Ea harum e...	https://via....	2023-03-23 01:49:3...

Order

Order
id: int
Phone: string
Name: string
Email: string
Address: string
is_paid: boolean
Status: int
Product_price: int
Ship_price: int
User_id: int
Admin_id: int
Created_at: datetime
+ getOrder(): int
+ setOrderID(orderID: int): void
+ getOrderCategory: Date
+ setOrderDate(orderDate: Date): void
+ getProductName(): int
+ setOrderProductPrice: int: void
+ getStatus(): int
+ productId(order_id): void
+ showOrderDetail(order_id): void
+ updateOrderStatus(order_id): void
+ listOrder(): List<Order>

id	name	address	email	phone	status	is_paid	ship_fee	product_price	ship_price	user_id	admin_id	created_at
1	Kleran...	6690 Ja...	nrnu@...	484.67...	3	1	0	11,488,000	50,000	12	1	2022-09-...
2	Rick H...	82859 C...	muham...	(769) 7...	1	0	0	77,416,000	58,000	18	1	2023-01-...
3	Deond...	391 Kon...	melyna...	+1-352...	4	0	0	63,327,000	36,000	15	1	2023-03-...
4	Veroni...	86262 C...	nienow...	+1 (68...	3	0	0	45,952,000	49,000	3	1	2022-04-...

OrderProduct

OrderProduct
Order_id: int
Product_id: int
Name: string
Amount: int
Price: int
Discount_price: int
Original_Price: int
+ addProductToOrder(orderId: int, productId: int, amount: int): void
+ updateProductOrder(orderId: int, productId: int, amount: int): void
+ removeProductFromOrder(orderId: int, productId: int): void

order_id	product_id	name	amount	price	discount_price	original_price
1	1	Film Accusantium accu...	1	5,228,000	0	4,574,500
1	2	Film Est neque et non.	1	6,260,000	0	5,216,666
2	1	Film Accusantium accu...	1	5,228,000	0	4,574,500
2	2	Film Est neque et non.	3	6,260,000	0	5,216,666
2	3	Film Cum sit et asperio...	4	8,650,000	0	7,688,888

Admin

Accountant	
Id: int	
Name: string	
Email: string	
Password: string	
Role: int	
Create_at: date time	
+GetBestSellingProducts(timePeriod: TimePeriod): List<Product> +GetCategoryByProduct(timePeriod: TimePeriod): List<RevenueByProduct> +GetRevenue(timePeriod: TimePeriod): List<Product> +GetRevenueByProduct(timePeriod: TimePeriod): List<RevenueByProduct> +GetName(): string +GetDate(): datetime +GetRole(): int	

id	name	email	password	created_at
10,002	Nhan vien	nhanvien@gmail.com	\$2a\$11\$7Nu96R7sR8B...	2023-04-30 21:36:17....
10,004	Ke toan	mkmvdfkvmv@dfbfd.bf...	\$2a\$11\$EA.JDGIASsw...	2023-05-01 22:35:58....
1	The Administrator	admin	\$2y\$10\$I2PUjePE50bL...	2023-04-27 07:08:25....
2	The Accountant	accountant	\$2y\$10\$T6cPvhb/Q.Q...	2023-04-27 07:08:25....

User

User	
Id: int	
Name: string	
Email: string	
Phone: string	
Address: string	
Password: string	
Create_at: date time	
Is_Agent: boolean	
+ addProductToCart (productid: int, amount: int) : bool + viewCart(): Cart + updateCartItem (productid: int, amount: int) : bool + removeCartItem (productid: int): bool + emptyCart(): bool	

id	name	phone	email	address	password	is_agent	created_at
1	Precio...	678-35...	noemy...	9376 Che...	\$2y\$10\$ey...	0	1985-09-21 15:19:32....
2	Keven...	(956) ...	torrey...	9972 Noe...	\$2y\$10\$6/i...	0	1972-02-04 12:35:40....
3	Espera...	1-234-...	obraun...	579 Vinni...	\$2y\$10\$yh...	0	1982-05-20 15:27:41....
4	Damio...	1-772-...	kmorar...	54920 Ke...	\$2y\$10\$Pt6...	0	2018-09-18 21:56:39....

4.2.2 Modifying sequence diagrams.

4.2.3 UI design

+ Order Form

MainForm

Order

Customer

Product

Import

HRM

Statistic

Log Out

Filter

☐ Order status
Unprocessed ▼

☐ Order time range
year/month/day year/month/day

☐ Payment status
☐ Is paid

Search

Filter

Status
Unprocessed ▼

☐ Payment status

Update

View

Order List

Id	Name	Phone	Email	Address	Status	PayStatus	ProductPrice	Ship

Figure 33 Mockup Order Form

+ OrderDetail Form

DetailOrder

Order detail

Name

Bao

Phone

123456798

Email

bao123456789@g

Address

154421 Tan Phong, District 7

Product Price

1520000

Ship price

20000

Total

1540000

Order At

1/1/2023 00:00:0

Transaction code

20000

Bank code

20000

Produc	ProductName	Amoun	Price	Sum	Discoun	OriginalPric
1	FilmAccusantiu	2	152000	154000	0	1000000

Print

Figure 34 Mockup OrderDetail Form

+ Statistic Form

MainForm

Order

Customer

Product

Import


HRM


Statistic

Log Out

Import Product

☐ Time Range

2021/01/01 

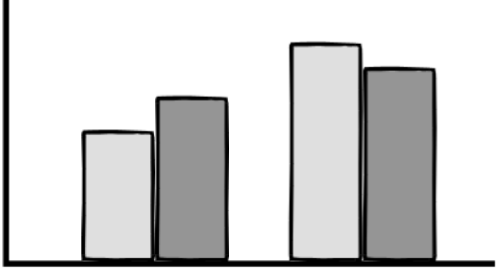
2023/02/02 

Total Revenue

Total Cost

Total profit

Revenue



Top Product

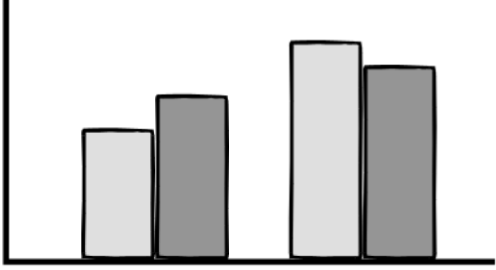


Figure 35 Mockup Statistic Form

4.2.4 SQL Code

- i. Create Database

```
CREATE DATABASE "project";  
USE "project";
```

- ii. Create Table

- + Admin

```
CREATE TABLE "Admin" (  
    "id" BIGINT NOT NULL,  
    "name" NVARCHAR(255) NOT NULL,  
    "is_admin_master" BINARY NOT NULL,  
    "email" NVARCHAR(255) NOT NULL,  
    "password" NVARCHAR(255) NOT NULL,  
    "active" BINARY NOT NULL DEFAULT "1",  
    "created_at" DATETIME2(0) NOT NULL,  
    PRIMARY KEY ("id")  
);
```

- + User

```
CREATE TABLE "User" (  
    "id" BIGINT NOT NULL,  
    "name" NVARCHAR(255) NOT NULL,  
    "phone" NVARCHAR(255) NULL DEFAULT NULL,  
    "email" NVARCHAR(255) NOT NULL,  
    "address" NVARCHAR(255) NULL DEFAULT NULL,  
    "password" NVARCHAR(255) NULL DEFAULT NULL,  
    "is_agent" BINARY NOT NULL,  
    "active" BINARY NOT NULL DEFAULT "1",  
    "created_at" DATETIME2(0) NOT NULL,  
    PRIMARY KEY ("id")  
);
```

+ Order

```
CREATE TABLE "Order" (  
    "id" BIGINT NOT NULL,  
    "name" NVARCHAR(255) NULL DEFAULT NULL,  
    "address" NVARCHAR(255) NULL DEFAULT NULL,  
    "email" NVARCHAR(255) NULL DEFAULT NULL,  
    "phone" NVARCHAR(255) NULL DEFAULT NULL,  
    "status" INT NOT NULL,  
    "is_paid" BINARY NOT NULL DEFAULT "0",  
    "ship_fee" FLOAT NOT NULL DEFAULT "0",  
    "product_price" FLOAT NOT NULL,  
    "ship_price" FLOAT NOT NULL DEFAULT "0",  
    "user_id" BIGINT NULL DEFAULT NULL,  
    "admin_id" BIGINT NULL DEFAULT NULL,  
    "bank_code" NVARCHAR(255) NULL DEFAULT NULL,  
    "transaction_code" NVARCHAR(255) NULL DEFAULT NULL,  
    "created_at" DATETIME2(0) NOT NULL,  
    PRIMARY KEY ("id"),  
    FOREIGN KEY,  
    FOREIGN KEY,  
    CONSTRAINT "order_user_id_foreign" FOREIGN KEY ("user_id") REFERENCES "User"  
("id") ON UPDATE NO_ACTION ON DELETE NO_ACTION,  
    CONSTRAINT "order_admin_id_foreign" FOREIGN KEY ("admin_id") REFERENCES  
"Admin" ("id") ON UPDATE NO_ACTION ON DELETE NO_ACTION  
);
```

+ Product

```
CREATE TABLE "Product" (  
    "id" BIGINT NOT NULL,  
    "category" INT NOT NULL,  
    "name" NVARCHAR(255) NOT NULL,  
    "price" FLOAT NOT NULL,  
    "description" NVARCHAR(max) NOT NULL,  
    "image" NVARCHAR(255) NOT NULL,  
    "created_at" DATETIME2(0) NOT NULL,  
    PRIMARY KEY ("id")  
);
```

+ OrderProduct

```
CREATE TABLE "OrderProduct" (  
    "order_id" BIGINT NOT NULL,  
    "product_id" BIGINT NOT NULL,  
    "name" NVARCHAR(255) NOT NULL,
```

```
"amount" INT NOT NULL,  
"price" FLOAT NOT NULL,  
"discount_price" FLOAT NOT NULL DEFAULT "0",  
"original_price" FLOAT NOT NULL,  
FOREIGN KEY,  
FOREIGN KEY,  
PRIMARY KEY ("order_id", "product_id"),  
CONSTRAINT "orderproduct_product_id_foreign" FOREIGN KEY ("product_id")  
REFERENCES "Product" ("id") ON UPDATE NO_ACTION ON DELETE NO_ACTION,  
CONSTRAINT "orderproduct_order_id_foreign" FOREIGN KEY ("order_id") REFERENCES  
"Order" ("id") ON UPDATE NO_ACTION ON DELETE NO_ACTION  
);  
  
CREATE INDEX "orderproduct_order_id_foreign" ON "OrderProduct" ("order_id");  
CREATE INDEX "orderproduct_product_id_foreign" ON "OrderProduct" ("product_id");
```

4.2.5 Software Classes Method Code

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class Product {
    private int id;
    private String name;
    private int category;
    private double price;
    private String description;
    private String image;
    private Date createdAt;

    public Product(int id, String name, int category, double price, String description, String image,
Date createdAt) {
        this.id = id;
        this.name = name;
        this.category = category;
        this.price = price;
        this.description = description;
        this.image = image;
        this.createdAt = createdAt;
    }

    public List<Product> getBestSellingProducts(TimePeriod timePeriod) {
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }

    public String getDescription() {
        return description;
    }

    public List<Product> getProducts() {
    }

    public int getCategory() {
        return category;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setDescription(String description) {
        this.description = description;
    }

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

    public void setQuantity(int quantity) {
    }

    public List<Product> listById(int orderId) {
    }
}
```

Figure 36 Software Classes Method Code Class Product

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class Order {
    private int id;
    private String name;
    private String phone;
    private String email;
    private String address;
    private boolean isPaid;
    private int status;
    private double productPrice;
    private double shipPrice;
    private int adminId;
    private Date createdAt;

    public Order(int id, String name, String phone, String email, String address, boolean isPaid, int
status,
                double productPrice, double shipPrice, int adminId, Date createdAt) {
        this.id = id;
        this.name = name;
        this.phone = phone;
        this.email = email;
        this.address = address;
        this.isPaid = isPaid;
        this.status = status;
        this.productPrice = productPrice;
        this.shipPrice = shipPrice;
        this.adminId = adminId;
        this.createdAt = createdAt;
    }

    public int getOrderID() {
        return id;
    }

    public void setOrderID(int orderID) {
        this.id = orderID;
    }

    public Date getOrderDate() {
        return createdAt;
    }

    public void setOrderDate(Date orderDate) {
        this.createdAt = orderDate;
    }

    public double getTotalPrice() {
        return productPrice + shipPrice;
    }

    public void setTotalPrice(double totalPrice) {
        this.productPrice = totalPrice;
    }

    public int getStatus() {
        return status;
    }

    public void printOrder(int orderId) {
    }

    public Object showOrderDetail(int orderId) {
    }

    public void updatePayment(int orderId) {
    }

    public List<Order> listOrder(String q, String filter) {
    }
}
```

Figure 37 Software Classes Method Code Class Order

```
import java.util.HashMap;
import java.util.Map;

public class OrderProduct {
    private int orderId;
    private int productId;
    private int amount;
    private String name;
    private double price;
    private double discountPrice;
    private double originalPrice;
    private Map<Integer, Integer> products;

    public OrderProduct(int orderId, int productId, int amount, String name, double
price, double discountPrice, double originalPrice) {
        this.orderId = orderId;
        this.productId = productId;
        this.amount = amount;
        this.name = name;
        this.price = price;
        this.discountPrice = discountPrice;
        this.originalPrice = originalPrice;
        this.products = new HashMap<>();
    }

    public void addProductToOrder(int orderId, int productId, int amount) {
    }

    public void updateProductOrder(int orderId, int productId, int amount) {
    }

    public void removeProductFromOrder(int orderId, int productId) {
    }
}
```

Figure 38 Software Classes Method Code Class Order Product


```
import java.util.ArrayList;
import java.util.List;

public class User {
    private int id;
    private String name;
    private String email;
    private String phone;
    private String address;
    private String password;
    private String created_at;
    private boolean is_agent;
    private Cart cart;

    public User(int id, String name, String email, String phone, String address, String password,
                String created_at, boolean is_agent) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.phone = phone;
        this.address = address;
        this.password = password;
        this.created_at = created_at;
        this.is_agent = is_agent;
        this.cart = new Cart();
    }

    public boolean addProductToCart(int productId, int amount) {
    }

    public Cart viewCart() {
    }

    public boolean updateCartItem(int productId, int amount) {
    }

    public boolean removeCartItem(int productId) {
    }

    public boolean emptyCart() {
    }
}
```

Figure 39 Software Classes Method Code Class User

```
import java.util.List;

public class Admin {
    private int id;
    private String name;
    private String email;
    private String password;
    private int role;
    private String created_at;

    public Admin(int id, String name, String email, String password, int role, String created_at)
    {
        this.id = id;
        this.name = name;
        this.email = email;
        this.password = password;
        this.role = role;
        this.created_at = created_at;
    }

    public List<Product> GetBestSellingProducts(TimePeriod timePeriod) {
    }

    public List<RevenueByProduct> GetCatetogyByProduct(TimePeriod timePeriod) {
    }

    public List<Product> GetRevenue(TimePeriod timePeriod) {
    }

    public List<RevenueByProduct> GetRevenueByProduct(TimePeriod timePeriod) {
    }

    public String GetName() {
        return name;
    }

    public String GetDate() {
        return created_at;
    }

    public int GetRole() {
        return role;
    }
}
```

Figure 40 Software Classes Method Code Class Admin

V. SYSTEM TESTING, DEPLOYMENT AND DEMONSTRATION

5.1 Testing: Test plan & Test case

+ Manage Order

Name	Test View all orders
Preconditions	The system is running, there are orders in the database
Input	Click on the "View all orders" button.
Expected Output	The system should display a list of all orders with their respective details (order number, date, customer name, total amount, status)

Name	Test Search and Filter Order
Preconditions	The system is running, there are orders in the database.
Input	Enter a search query in the search box or select a filter option (e.g. by date, customer name).
Expected Output	The system should display a list of orders that match the search or filter criteria.

Name	Test Update Order Status
Preconditions	The system is running, there are orders in the database.
Input	Click on the "Update order status" button for a specific order.
Expected Output	The system should allow the accountant employee to change the status of the order (e.g. from "Processing" to "Shipped").

Name	Test Update Payment Status
Preconditions	The system is running, there are orders in the database.
Input	Click on the "Update payment status" button for a specific order.
Expected Output	The system should allow the accountant employee to change the payment status of the order (e.g. from "Pending" to "Paid").

Name	Test Print Order Detail
Preconditions	The system is running, there are orders in the database.
Input	Click on the "Print order detail" button for a specific order.
Expected Output	The system should generate a printable document that contains all the details of the order (order number, date, customer name, products ordered, total amount, status).

+ Manage Shopping Cart

Name	Verify that the user can add a product to their shopping cart.
Preconditions	User is logged in and browsing the available products.
Input	User clicks "Add to Cart" button on a product. System adds the selected product to the user's shopping cart.
Expected Output	The selected product is added to the user's cart. The cart shows the product's name, image, price, and quantity.

Name	Verify that the user can update the quantity of a product in their shopping cart.
Preconditions	User has one or more products in their shopping cart.
Input	User clicks the "Update" button next to the quantity of a product. User enters a new quantity for the product. User clicks "Update Cart".
Expected Output	The user is able to change the quantity of the selected product. The cart updates to show the new quantity and the updated total cost.

Name	Verify that the user can remove a product from their shopping cart.
Preconditions	User has one or more products in their shopping cart.
Input	User clicks the "Remove" button next to a product.
Expected Output	The selected product is removed from the user's cart. The cart updates to show the new total cost.

Name	Verify that the user can continue shopping after adding a product to their cart.
Preconditions	User has added one or more products to their cart. User clicks the "Home" button.
Input	The user is returned to the product page.
Expected Output	The products in the cart are retained.

+ Statistic

Name	Test the revenue report generation
Preconditions	
Input	Sample data with sales transactions.
Expected Output	Verify the report generated by the system has the correct revenue.

Name	Test the top product report generation
Preconditions	
Input	Sample data with sales transactions.
Expected Output	Verify the report generated by the system has the correct top selling products in descending order.

Name	Test the statistic report generation by month
Preconditions	
Input	Sample data with sales transactions from different months.
Expected Output	Verify the report generated by the system has the correct statistics for each month, including revenue, top products and other relevant data.

Name	Test the filter function for revenue report
Preconditions	
Input	Apply a filter to the revenue report, such as date range or specific product.
Expected Output	Verify the report generated by the system shows only the data corresponding to the filter criteria.

Name	Test the filter function for top product report
Preconditions	
Input	Apply a filter to the top product report, such as date range or specific product category.
Expected Output	Verify the report generated by the system shows only the data corresponding to the filter criteria.

5.2 Deployment

Deploy a sub-system of a software system developed using C# .NET Framework Winform with LINQ and SQL Server.

Step 1: Pre-Deployment Preparation

- Ensure that all the necessary libraries and dependencies are included in the software system.
- Test the sub-system thoroughly to ensure that it is functioning correctly and without errors.
- Create a backup of the sub-system and the database to avoid any data loss during deployment.
- Check the hardware and software requirements of the sub-system and ensure that they are met in the deployment environment.

Step 2: Deployment Plan

- Determine the deployment environment: Decide where the sub-system will be deployed and ensure that the environment meets all the necessary requirements.
- Choose a deployment method: Deploy the sub-system using various methods such as manual deployment, batch scripts, or automated deployment tools.
- Set up the deployment process: Configure the deployment process to ensure that it runs smoothly and without errors.
- Test the deployment process: Perform a trial deployment in a non-production environment to identify any issues or errors in the deployment process.

Step 3: Deployment Process

- Stop the sub-system: Shut down the sub-system before deployment.
- Install the sub-system: Install the sub-system on the deployment environment.
- Configure the sub-system: Configure the sub-system to work with the deployment environment. This may include setting up database connections, configuring ports and protocols, etc.
- Start the sub-system: Start the sub-system and ensure that it is functioning correctly.
- Test the sub-system: Perform a series of tests to ensure that the sub-system is working correctly and without errors.

5.3 Demonstration

First we have to login

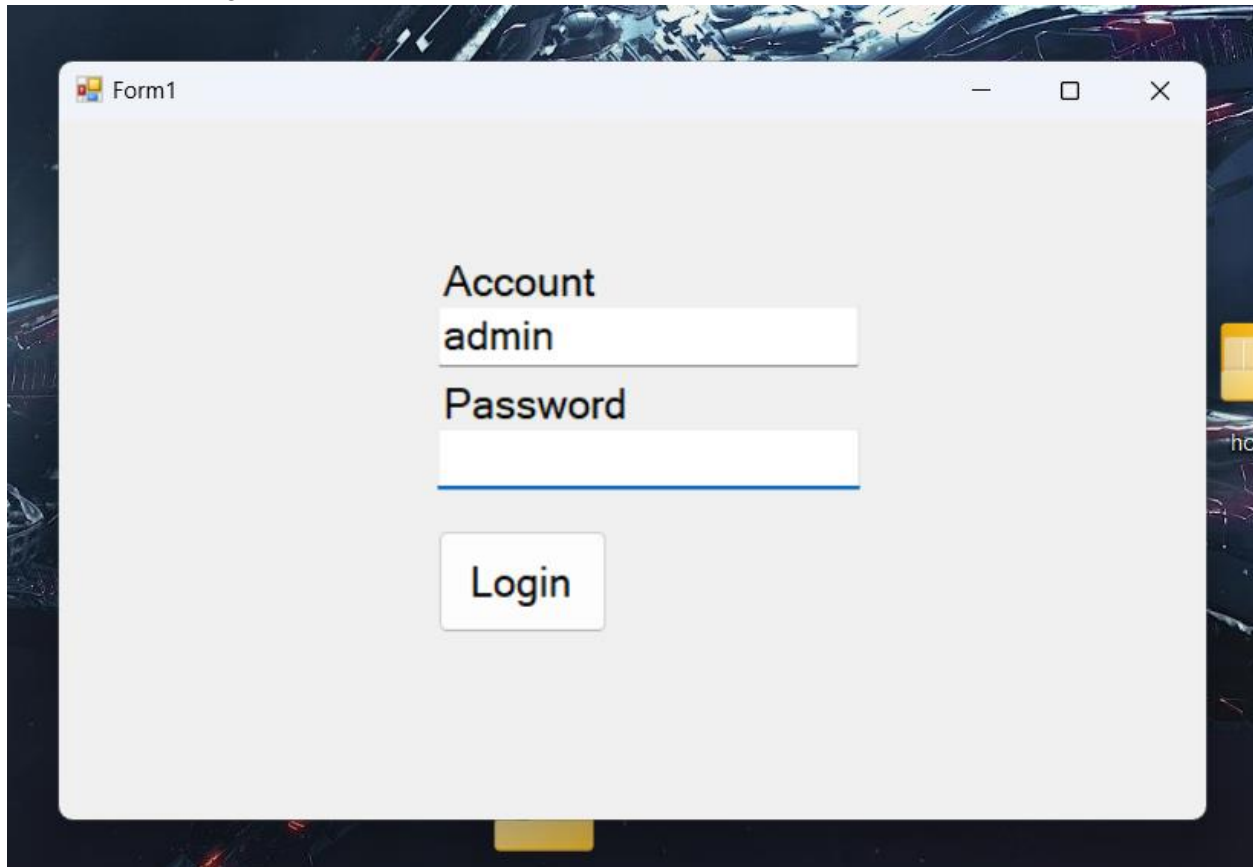
A screenshot of a Windows-style application window titled "Form1". The window has a light gray background and contains a login form. The form consists of two text input fields: the first is labeled "Account" and contains the text "admin"; the second is labeled "Password" and is empty. Below the input fields is a button labeled "Login". The window is set against a desktop background showing a car engine and some icons.

Figure 41 Login form

Manage Order

MainForm

Filter

☐ Order status ☐ Order time range ☐ Payment status

Unprocessed ▾ 2023/05/08 ▾ 2023/05/08 ▾ ☐ Is paid

Search

Action

Status: Unprocessed ▾ ☐ Payment status **Update**

View

Order List

	id	Name	Phone	Email	Address	Status
▶	10008	nj	kn	kjn	kjn - jk	Success..
	10007	bnkhbkh	kb	jkb	Ho Chi ...	Success..
	10006	bdfbdf	bdbdfb	dfbdfb	Ho Chi ...	Success..
	10005	b	bh	bjh	bjh - b	Success..
	10004	n	fqnfnfn	nfn	fqnfn - fn	Success..
	10003	Lok	0123465...	sdbvsdf...	333 Lê L...	Success..
	10002	LOk	0123468...	dfbf@bd...	333 Le L...	Success..
	174	Presley ...	435.900....	streich.r...	14014 Br...	Delivering

Figure 42 Manage Order

Detail order

DetailOrder

Order detail

Name: Lexie Wolf IV Phone: +1-551-402-8943 Email: mina03@gmail.com

Address: 9908 Robyn Plain Apt. 424Swaniawskiton, DE 23822

Product price: 26140000 Ship price: 34000 Total: 26140000

Ordered At: 4/17/2023 7:46:17 F Transaction code: Bank code:

	ProductId	ProductNa	Amount	Price	Sum
▶	1	Film Acc...	5	5228000	26140000

Print

Figure 43 Detail Order

Update Order

Filter

☐ Order status ☐ Order time range ☐ Payment status

Unprocessed 2023/05/08 2023/05/08 ☐ Is paid

Search

Action

Status

Cancelled Unprocessed Delivering Canceled Decline Successful Destroy

☒ Payment status Update

Order List

	id	Name	Phone	Email	Address	Status
	10002	LOk	0123468...	dfbf@bd...	333 Le L...	Success..
	174	Presley ...	435.900...	streich.r...	14014 Br...	Delivering
	22	Prof. Ro...	+148454...	fharber...	8935 Ha...	Cancellec
▶	96	Lexie W...	+1-551-4...	mina03...	9908 Ro...	Cancellec
	92	Marlin D...	1-708-66...	muller.s...	61124 Ci...	Cancellec
	177	Mr. West...	541.939...	collier.jul...	5259 Lin...	Delivering
	108	Dr. Colu...	412-238-...	ycummi...	27401 L...	Delivering
	128	Lenna T...	(224) 48...	macie.o...	3201 Zo...	Delivering

Figure 44 Update Order

Statistic

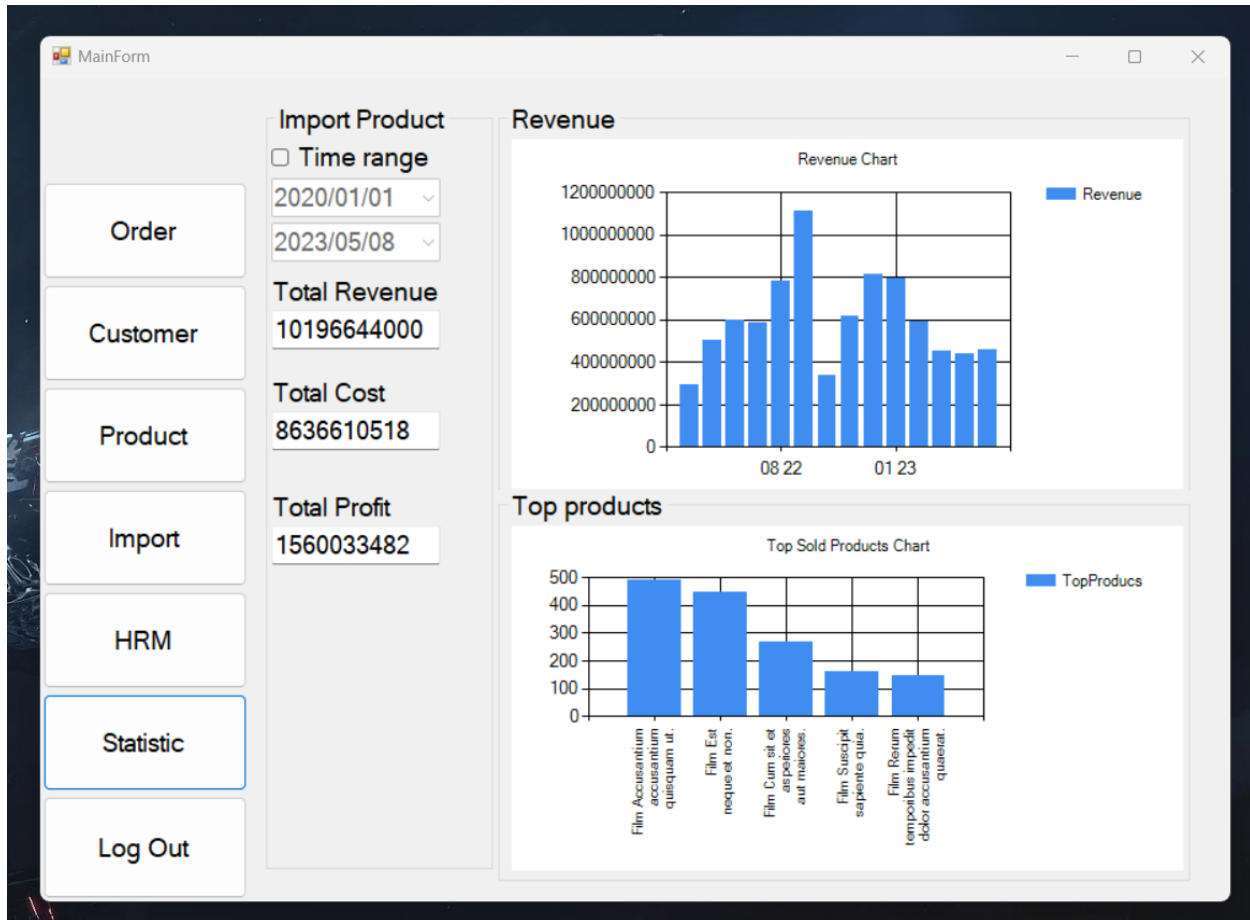


Figure 45 Statistic

Conclusions/ Recommendations

During the process of implementing the project, our team has acquired a lot of knowledge about information systems from planning, system analysis, design to building application programs. Regarding the analysis and design of the system, the business processes have been fully implemented as a basis for building practical application programs. The system helps to provide accurate, fast and complete information, well serving the import and export management process, supporting accounting staff in managing import and export of the company's goods. . The system has introduced the main functions of the program, but some parts are still not handled well.

The development direction of the topic is to update, upgrade and fix errors that arise during use, expand new functions to be able to fully meet the requirements of users, build programs. perfect to solve well the problems that reality posed.

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>