VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

FINAL PROJECT IN SOFTWARE ENGINEERING

The Mobile Shop

Instructor: **MSC.Pham Thai Ky Trung**

Executor: **Nguyen Tan Bao- MSSV: 521H0438**

Bui Huu Loc – MSSV: 521H0504

Class: **21H50203**

Course: **25**

HO CHI MINH CITY, 2023

VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

FINAL PROJECT IN SOFTWARE ENGINEERING

The Mobile Shop

Instructor: **MSC.Pham Thai Ky Trung**

Executor: **Nguyen Tan Bao- MSSV: 521H0438**

Bui Huu Loc – MSSV: 521H0504

Class: **21H50203**

Course: **25**

HO CHI MINH CITY, 2023

ACKNOWLEDGMENTS

In my pleasant, I thank Mr. Pham Thai Ky Trung, who taught me and help me to finish this project and Mr. Tran Thanh Phuoc taught me in the exercise room.

It is an honor to have the opportunity to study and work at Ton Duc Thang University and also thank the teachers of the Faculty of Information Technology for supporting me in previous subjects.

Finished, thanks to my friend Bui Huu Loc, who shared this project with me and help me when I have trouble.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1. INTRODUCTION

1.1. Purpose and Scope

1.1.1. Purpose

- Develop functions for accountants to create warehouse receipts when the company imports goods.
- Developed Web functionality that allows dealers to place item lists and choose a form of pay(Cash, bank transfer, Momo...). Agents can pay online and view order status.
- Develop functions for accountants to make warehouse slips to ship goods to agents (print export slips), update order status as in transit, and update agents' paystatus.
- Develop a function for accountants to review statistic of incoming goods, best-selling items, the revenue of each month.
- Website Development or Retail Moblie App to sell products

1.1.2 Scope

The scope of the project appears to involve developing various functions and web functionalities for a company involved in importing and selling goods. Specifically, the project includes the following tasks:

1. Developing functions for accountants to create warehouse receipts when the company imports goods.
2. Developing web functionality that allows dealers to place item lists and choose a form of payment, and agents to pay online and view order status.
3. Developing functions for accountants to make warehouse slips to ship goods to agents, update order status as in transit, and update agents' pay status.
4. Developing a function for accountants to review statistic of incoming goods, best-selling items, and revenue for each month.
5. Developing a website or mobile app for retail sales.

Overall, the project involves developing a system that streamlines the company's importing and selling processes, allowing for more efficient and accurate management of goods, payments, and orders.

1.1.3 Limitations

- Platform-specific limitations: The software needs to be developed as both a Windows Form Project and a Web Forms Project to cater to the needs of staff who work both in the office and from home. Each platform has its own set of limitations and constraints that need to be taken into account during development.

- pay gateway limitations: The software needs to integrate with VNPay to facilitate payments from agents. The pay gateway has its own set of limitations and restrictions that need to be considered during development.

- Communication limitations: The software needs to be able to send order confirmations to agents via Zalo or Email. These communication channels have their own limitations and constraints that need to be considered during development.

- Database limitations: The B2C Ecommerce Website/Mobile App needs to connect to the same MSSQL Server as the other software products. This means that the database schema and access controls need to be designed to accommodate the needs of all the software products.

- Security limitations: The software needs to have a login function to ensure that only authorized staff and agents can access the system. The login function needs to be secure and protect against common security threats such as brute force attacks, SQL injection, and cross-site scripting.

- Performance limitations: The software needs to be able to handle large volumes of data and transactions without impacting system performance. This means that the software architecture needs to be designed to scale horizontally or vertically to accommodate increasing loads.
- Usability limitations: The software needs to be user-friendly and easy to use for both staff and agents. This means that the user interface needs to be intuitive and accessible, with clear instructions and error messages to guide users through the system.

1.2. Product Overview (including capabilities, scenarios for using the product, etc.)

This is a description of a warehouse and sales management system. When a distributor imports goods, the employee will generate a code and update it into the database. Accountants can log into the system to create warehouse entry or exit orders, and update the status of orders, and the pay status of the distributor. Distributors can also log into the system to place orders, choose pay methods, and view the status of their orders.
In addition, accountants can also view reports on goods imported/customers, the best-selling products, and monthly revenue reports. To sell products to customers, a B2C website or mobile application needs to be developed to connect to the same MSSQL server and use VNPay for distributor payments.
Warehouse managers can manage personnel and user accounts, as well as product management.
Employees of the distributor use a Windows Form project while in the office and a Web Forms project when working remotely

1.3. Structure of the Document

Chapter	Description
1. Introduction	Project overview
2. Project Management Plan	Describe project planning and project management
3. Requirement Specifications	Analyze topic requirements
4. Architecture	Software design models and structures
5. Design	Interface and data design
6. Test Plan	Software retesting

CHAPTER 2 - PROJECT MANAGEMENT PLAN

2.1. Project Organization

Role:

- Design(Design UI): Design User Interface for a website, winform, webform
- PMO(Project management organization): Participant in the project, report
- Dev: Code for WinForms, web form, web B2C
- Directive: Leader in team
- Design Architecture: Design Architectural Model

Name	Role
Nguyen Tan Bao	Design UI, PMO, Dev
Bui Huu Loc	Directive, PMO, Dev, Design Architecture

- Communication for project:
-
    - ➔ 15-Minute Sprint Planning Meetings in Discord: call or chat when have any problem about the project or report .
    - ➔
- Management common:
    - ➔ Google Docs for the report.
    - ➔ Git for code.

2.2. Lifecycle Model Used

Use the agile model as this model can easily unify the ideas of team members due to many problems to be analyzed so there are many test cases that need to be tested again after completion. If the test case does not match the requirements, it can be corrected, by adding comments when correcting for team members. Easy to change when wrong request object. Scrum quickly gives users the freedom to deploy during application development. Easy to learn, easy to use. Changing operations is quick and easy. Minimize risk when building and developing products—support to optimize the work efficiency and efforts of the programmer team. Processing is fast and compact, helping customers to use the product in a shorter time.

2.3. Risk Analysis

- The first software project so there are many shortcomings.
- Group matching is quite late, so the time to do it is quite urgent.
- Poor problem analysis leads to repeated redrawing of use cases and analysis of use cases.
- Fixing use cases leads to UI fixes and UI forms fixes.

2.4. Hardware and Software Resource Requirements

Hardware:

- Processor: Intel Pentium core i5
- Hard Disk:4 GB
- RAM:512GB

Software:

- Operating System:- windows 10
- Front End: ASP.NET, ASP MVC with visual studio 2022, Laravel framework
- Back End:- SQL Server 2022 with azure data studio

2.5. Deliverables and Schedule

2.5.1. Deliverables

This section is all the documentation to make software from analysis to final product including

- Code: Winform, Webform, Web B2C
- Document: Final report
- Link: Figma

2.5.2. Schedule

Week 1	Week 2	Week 3	Week 4
The team will analyze the topic and plan the project	The team that started sprint 1 was Requirement Specifications	The team working on sprint 2 and sprint 3 is architecture and design	The team implementing sprint 4 is the test plan and review the project

2.6. Monitoring, Reporting, and Controlling Mechanisms

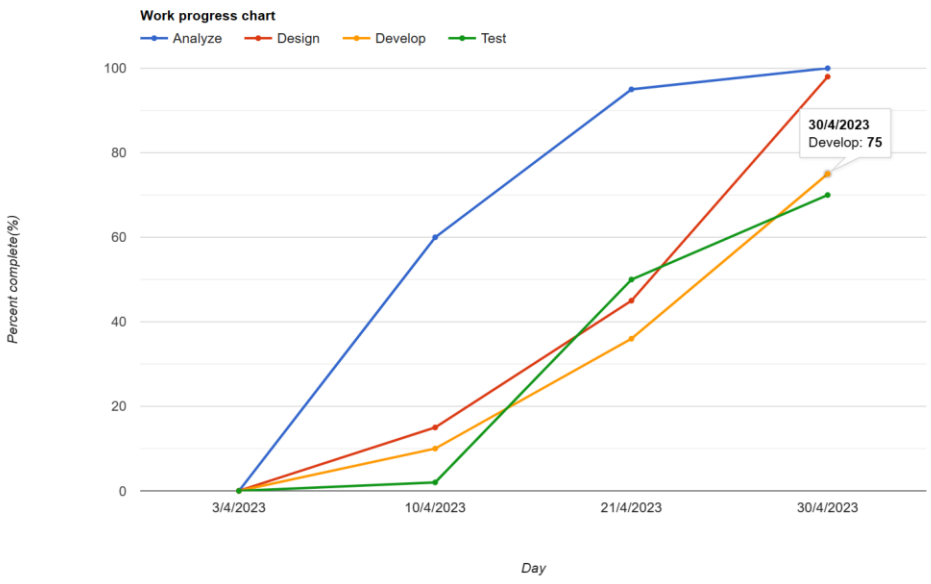


Figure 1 Work progress chart

2.7. Professional Standards

- Define requirements: Identify and document all requirements of the software, including features, functions, and user roles.
- Design architecture: Design the software architecture, including the database schema, application layers, and user interfaces. Decide which programming language and framework to use.
- Develop and test software: Develop the software according to the requirements and design. Test each feature to ensure it works correctly.
- Deploy software: Deploy the software to the production environment, making it available to users.
- Maintain software: Provide ongoing maintenance and support for the software, including bug fixes, security updates, and feature enhancements.
- Use version control: Use version control software to manage changes to the source code and collaborate with other developers.
- Use coding standards: Follow coding standards and best practices to ensure code quality, readability, and maintainability.
- Use testing standards: Use testing standards and best practices, such as unit testing and integration testing, to ensure the software works as expected.
- Use security standards: Use security standards and best practices to protect the software and its users from security threats.
- Document software: Document the software, including its requirements, design, implementation, and user guides, to make it easy to understand and use.

2.8. The evidence all the artifacts have been placed under configuration management.

Github repository

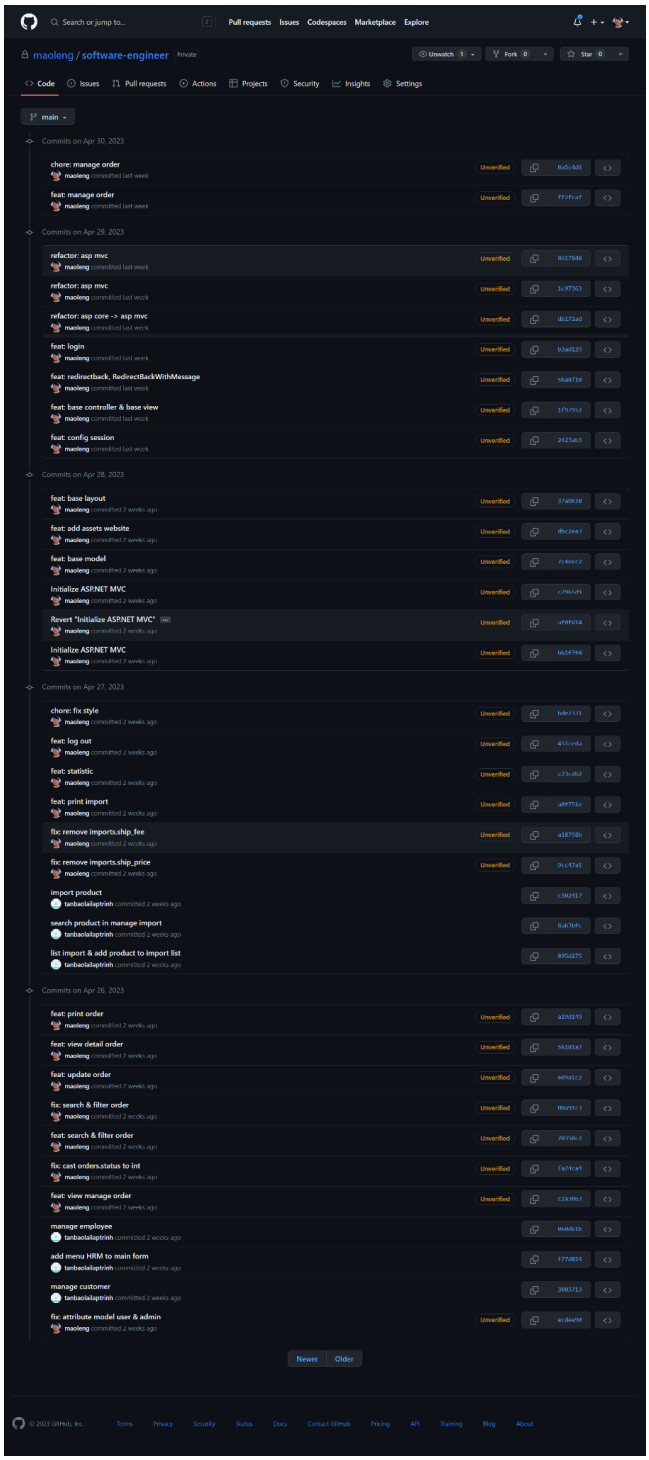


Figure 2 GitHub Activity

2.9. Impact of the project on individuals and organizations

- Eye-catching interface
- Easy to use for all ages when getting started.
- Enter data, update products easily.

CHAPTER 3 - REQUIREMENT SPECIFICATIONS

3.1. Stakeholders for the system

Actor	Use Case
Accountant	Log in, sign up, manage order, statistic, Manage Order
Customer	View product information, Manage Shopping Cart, Orders, View History order
Customer (without account)	View product information, Manage Shopping Cart, Orders
Agent	View product information, Manage Shopping Cart, Orders, View History Orders,
Manager	Manage Product, Manage Human Resource, Manage Account Agent

3.2. Use case model.

3.2.1. Graphic use case model

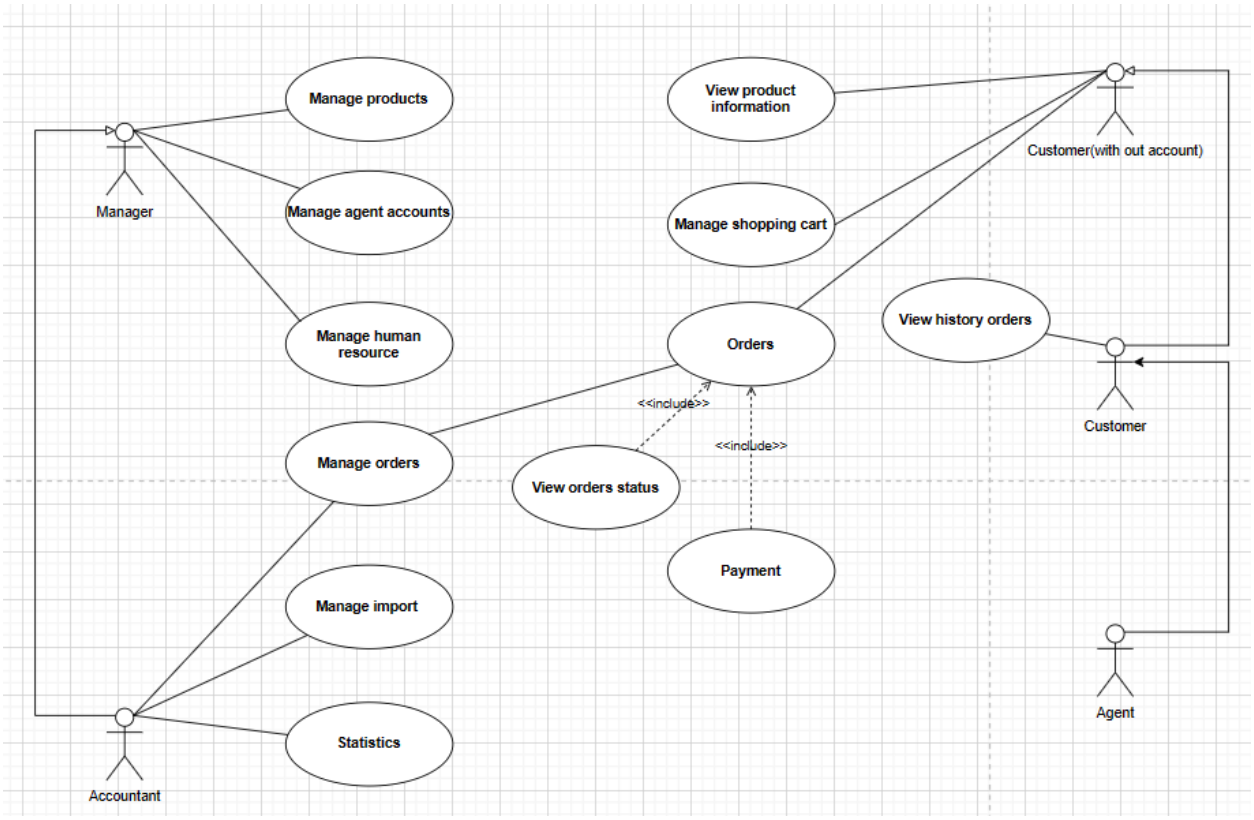


Figure 3 Use Case Diagram

3.2.2. Textual Description for each use case

<table>
<tr><td>Use case name:</td><td colspan="2">View order history.</td></tr>
<tr><td>Participating Actors:</td><td colspan="2">Customer has an account, agents</td></tr>
<tr><td>Entry Condition(s):</td><td colspan="2">The agent/customer has logged into the order management system.</td></tr>
<tr><td rowspan="3">The flow of Events:</td><td>Customer</td><td>System</td></tr>
<tr><td>The customer/agent accesses the "Order History" feature on the system interface.</td><td>The system displays a list of orderplaced in the past by the agent, in chronological order from newest to oldest.</td></tr>
<tr><td>The customer/agent can view the details of each order by clicking on it in the list.</td><td>The system shows detailed information such as product name, quantity, price, order date, order status, etc.</td></tr>
<tr><td>Exit Condition:</td><td colspan="2">The agent/customer has viewed their order history and can exit this feature to return to other features in the system.</td></tr>
<tr><td>Exceptions:</td><td colspan="2">If no orderhave been placed in the agent's past, the system will notify the customer and not display a list of orders.</td></tr>
<tr><td>Special Requirements:</td><td colspan="2">To use this feature, the customer needs to have a login account for the order management system and be granted access to this feature.</td></tr>
</table>

Use case name:	View product information	
Participating Actors:	Customer	
Entry Condition(s):	The customer has logged in to the product management system.	
The flow of Events:	**Customer**	**System**
	The customer accesses the "View Products" feature on the system interface.	The system displays a list of products available, including information such as product name, brand, price, and image.
	The customer can filter the list of products by brand or price range, or search for specific products by name using the search feature provided by the system. When the customer selects a product from the list.	The system displays detailed information such as product description, specifications, and availability.
Exit Condition:	The user has viewed the product information and can exit the feature to return to other components in the system.	
Exceptions:	If there are no products available, the system will notify the user and display an empty list.	
Special Requirements:	To use this feature, the user needs to have a logged-in account in the product management system and be granted access to this feature.	

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:	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. 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.	

Special Requirements:	The system needs to update the inventory in real-time to ensure accurate product availability information. The system needs to calculate and display the total price of the products in the cart, including taxes and shipping costs. 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.

Use case name:	Orders.	
Participating Actors:	Customer	
Entry Condition(s):	The Customer has selected the products they wish to purchase and added them to their cart.	
The flow of Events:	**User**	**System**
	The Customer selects the "Checkout" button to proceed with their order.	The system displays a form for the Customer to enter their shipping information and paydetails.
	The Customer enters their shipping information and selects their preferred paymethod - either direct payto the Agent or online payment.	If the Customer chooses direct payto the Agent, the system displays a message indicating that the total price will be cheaper when purchased through the Agent.
	The Customer confirms their order and submits the pay information.	The system processes the payand generates an order confirmation.
Exit Condition:		

Use case name:	Payment.	
Participating Actors:	Customer.	
Entry Condition(s):	The customer has selected one or more products for purchase.	
The flow of Events:	**Customer**	**System**
	The customer proceeds to checkout and is prompted to select a paymethod. The customer selects either "pay in-person" or "pay online".	If the customer selects "pay in-person", they are instructed to visit the physical store to complete the payment./(If the customer selects "pay online", they are directed to a secure paygateway to complete the transaction.)
Exit Condition:	The customer has successfully completed the payprocess.	
Exceptions:	If the customer's payis declined, they will be prompted to choose another paymethod or contact customer support for assistance. If the customer cancels the payprocess, they will be redirected back to the shopping cart.	
Special Requirements:	The paygateway must be secure and reliable to protect the customer's sensitive financial information.	

Use case name:	View order status	
Participating Actors:	Agents/Customer	
Entry Condition(s):	The actor order success and logged	
The flow of Events:	**Actor**	**System**
	The actor logs in to the system.	Check to validate the account
	The actor selects the function “view order status”	Show where is the status of the item(unprocessed, rejected, in transit, completed).
Exit Condition:	The actor logs out system	

Use case name:	manage order..	
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 orderthat have been created in the system, including information such as order number, creation date, customer name, order status, and paystatus.
	The accountant can filter the list of orderby different criteria, such as creation date, customer name, order status, or paystatus. 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 paystatus 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.	

Use case name:	Manage export.	
Participating Actors:	Accountant	
Entry Condition(s):	Accountant logged to system	
The flow of Events:	**Accountant**	**System**
	Accountant creates fill information about export orders	The system update to the database. The system prints the release slip.
Exit Condition:	The accountant logs out of the system	
Exceptions:	If the system is down or not accessible, the accountant will not be able to print the invoice at the time of shipment. In this case, the accountant will have to print the invoice later when the system is back online.	
Special Requirements:	The accountant must ensure that the invoice number is unique and sequential. The accountant must keep a copy of the invoice for record-keeping purposes.	

Use case name:	Manage product.	
Participating Actors:	Manager	
Entry Condition(s):	The manager is logged in to the system and has permission to manage product.	
The flow of Events:	**User**	**System**
	The manager selects the "Manage Product" option from the menu.	The system displays a list of existing products.
	The manager can choose to create a new product or edit/delete an existing one. If the manager chooses to create a new product, they are prompted to enter the product information such as name, description, price, and discount. If the manager chooses to edit an existing product, they can modify any of its information or delete the product entirely. The manager can also set a customized discount for each product.	After any changes are made, the system updates the product database.
Exit Condition:	The manager has successfully created, edited, or deleted a product and the product database has been updated accordingly.	

Exceptions:	If the manager enters invalid or incomplete information, the system prompts them to correct the errors before saving the changes. If the manager attempts to delete a product that is associated with existing orderor inventory records, the system prompts them to confirm the action or to cancel it to prevent data loss.
Special Requirements:	The product management system must be intuitive and easy to use to enable the manager to add, modify, or delete products efficiently. The customized discount feature must be flexible and allow for different levels of discounts based on product type or customer segment.

<table>
<tr><td>Use case name:</td><td colspan="2">manage customer</td></tr>
<tr><td>Participating:</td><td colspan="2">Manager.</td></tr>
<tr><td>Entry condition:</td><td colspan="2">The manager must have administrative access to the system.</td></tr>
<tr><td rowspan="3">The flow of events:</td><td>Manager</td><td>System</td></tr>
<tr><td>The manager logs into the system with their administrative credentials.

The manager navigates to the "manage customer" section.</td><td>The system displays a list of all agents and regular customers.</td></tr>
<tr><td>The manager selects the function block/reset password agent’s accounts.

The manager selects the function to add an account.

The manager selects the function to view all accounts.</td><td>The system updates the database

The system inserts the database..

The system selects all accounts in the database.</td></tr>
<tr><td>Exit condition.</td><td colspan="2">The manager completes the desired actions and logs out of the system.</td></tr>
<tr><td>Exceptions:</td><td colspan="2">If the manager does not have administrative access to the system, they cannot access the "manage customer" section.

If the manager enters incorrect or incomplete information when creating a new agent account, the system will display an error message and prompt them to correct the errors.</td></tr>
</table>

Use case name:	Manage the human resource.	
Participating Actors:	Manager	
Entry Condition(s):	The manager must have administrative access to the system.	
The flow of Events:	**Manager**	**System**
	The manager logs into the system with their administrative credentials. The manager navigates to the "Manage Staff Accounts" section.	The system displays a list of all staff.
	The manager selects function block/add/reset password	Update the information staff's accounts to the database
Exit Condition:	The manager logs out of system	

3.3. The rationale for your use case model

First, we will have 5 actors who will participate in this use case including 4 main actors who are accountants, managers, retail buyers, and agents and the main actor is the customer.

➔ use case manage employee and manage account customers.

The manager will be able to manage his clients along with the personnel in the company. In which, the manager can view information about the accounts of agents, ordinary customers, and employees in the company along with being able to create accounts, lock accounts or reset passwords (lock when no longer available). is an agent, the employee quits)

➔ use case Manage Product

Managers can manage product by viewing, deleting, adding, updating product information, and customizing discounts for each product.

➔ use case statistic

Accountants can view sales statistic, best-selling items, and import and export order printed in forms/reports or can be viewed on computers.

➔ use case manage import order and use case manage order

Accountants can control stock receipts by creating invoices in the receipt when importing products from suppliers and viewing orders. Along with that, you can control your order by being able to view orders, update orders, update pay status, and print release notes when shipping.

➔ use case /View product information/ Manage shopping cart/ Orders/ Payment

Agents and customers who are logged in and not logged in can view the product information by searching and filtering products by brand, name, and price,... after viewing the information, they can add it to their cart. item and can increase the number of items in your cart. After that, you can proceed to pay by clicking to order items in the cart including filling in personal information, choosing a pay method that can pay directly or online, and money at checkout of your account. Dealers will be cheaper.

- ➔ use case/ View order history / View order status / Cancel order.

Once a successful pay has been made, customers who already have an account (including agents and retailers) can view the history of the items they have ordered when they select a specific item. status of that order (where is the order processed or rejected, if it has been processed, you can see where the item arrives and when it is received the order status will be completed, yes Can cancel the order if the order is not completed by choosing to cancel the order)

3.4. Non-functional requirements

Non-functional requirements for the software:

- "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

‘

CHAPTER 4 - ARCHITECTURE

4.1. Architectural style(s) used

- Web form: MVC, Dependency Injection, Inversion of Control, Repository Pattern, Service Container, Routing, and Middleware
- Winform: MVC, Entity Framework

4.2. Architectural model

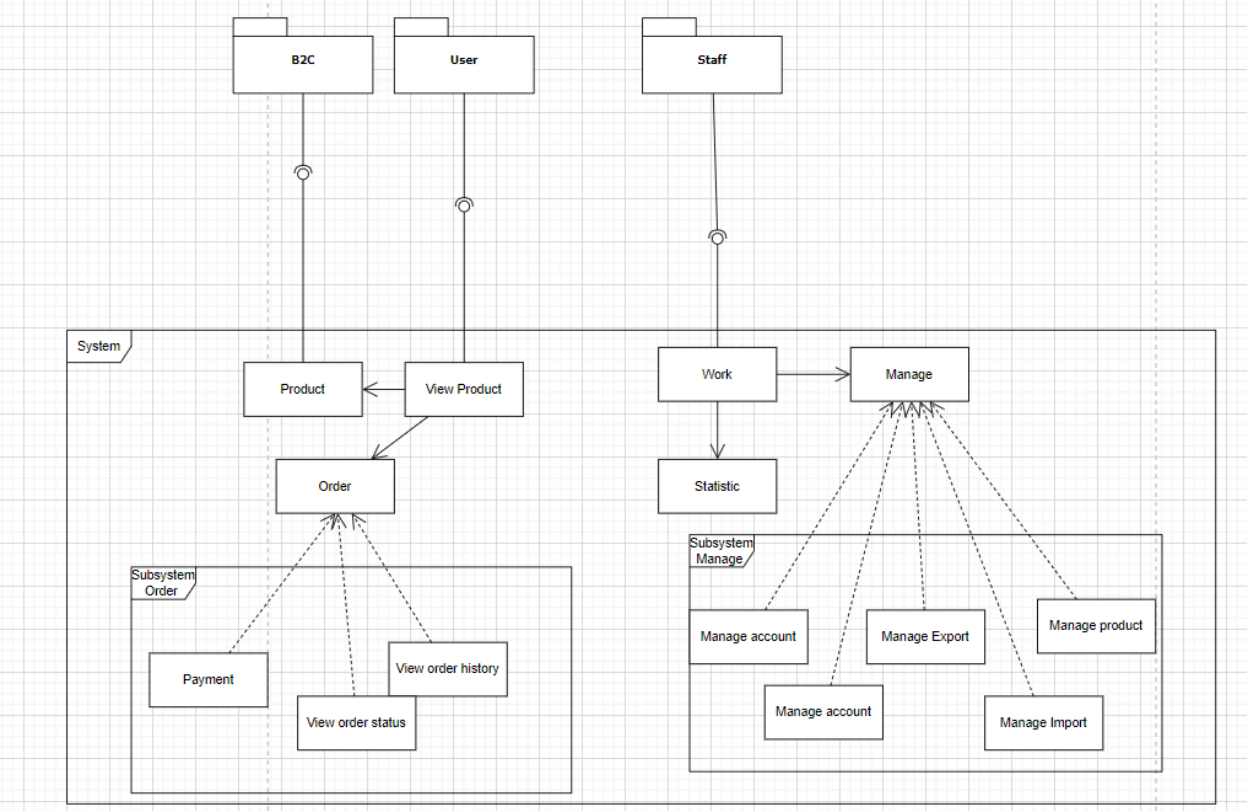


Figure 4 Architectural model

4.3. Technology, software, and hardware used

- B2C: Laravel 10.8
- Webform: ASP NET MVC
- Winform .netframework 4.7

4.4. Rationale for your architectural style and model

The architecture that could be suitable for the system is a client-server architecture. This is because the system has different types of clients (staff, agents, and customers), and each client has different requirements and needs access to different parts of the system. A client-server architecture will allow for a clear separation of concerns, where the client can focus on the user interface and the server can focus on the business logic and data storage.

Within the client-server architecture, a layered architectural style could be used. This style allows for a separation of concerns and abstraction of functionality into distinct layers. This could include presentation, application, business logic, and data storage layers. By separating these concerns, each layer can focus on its specific responsibilities, making the system more maintainable and scalable.

For the web application, the Laravel 10.8 framework could be used as it is a widely used PHP framework that provides a range of features, including authentication, routing, and templating. Laravel can also provide easy integration with different payment gateways and messaging services.

For the Windows Form project, the .NET Framework 4.7 could be used as it is a robust platform for developing Windows desktop applications. The .NET Framework also provides a range of features and libraries that can be used for developing enterprise-level applications.

CHAPTER 5 - DESIGN

5.1. GUI (Graphical User Interface) design

5.1.1 Mock up

5.1.1.1WinForm

Account: Admin
Password: 1234

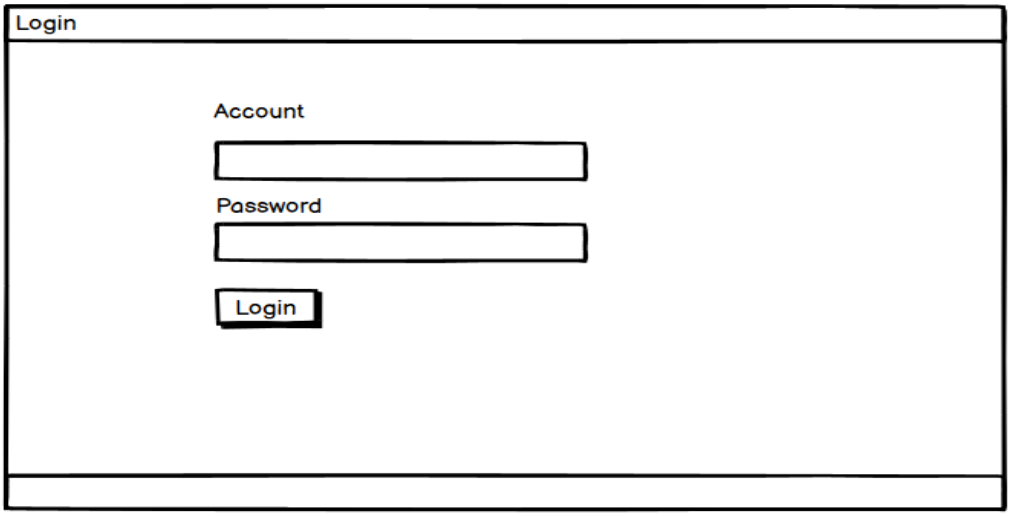


Figure 5Mockup Login

When you successfully log in, you will be redirected to the main form.
This is a main form (the order form). In this form you can filter Order status, Order time range, and pay status. You can search Order by name and can update status order.

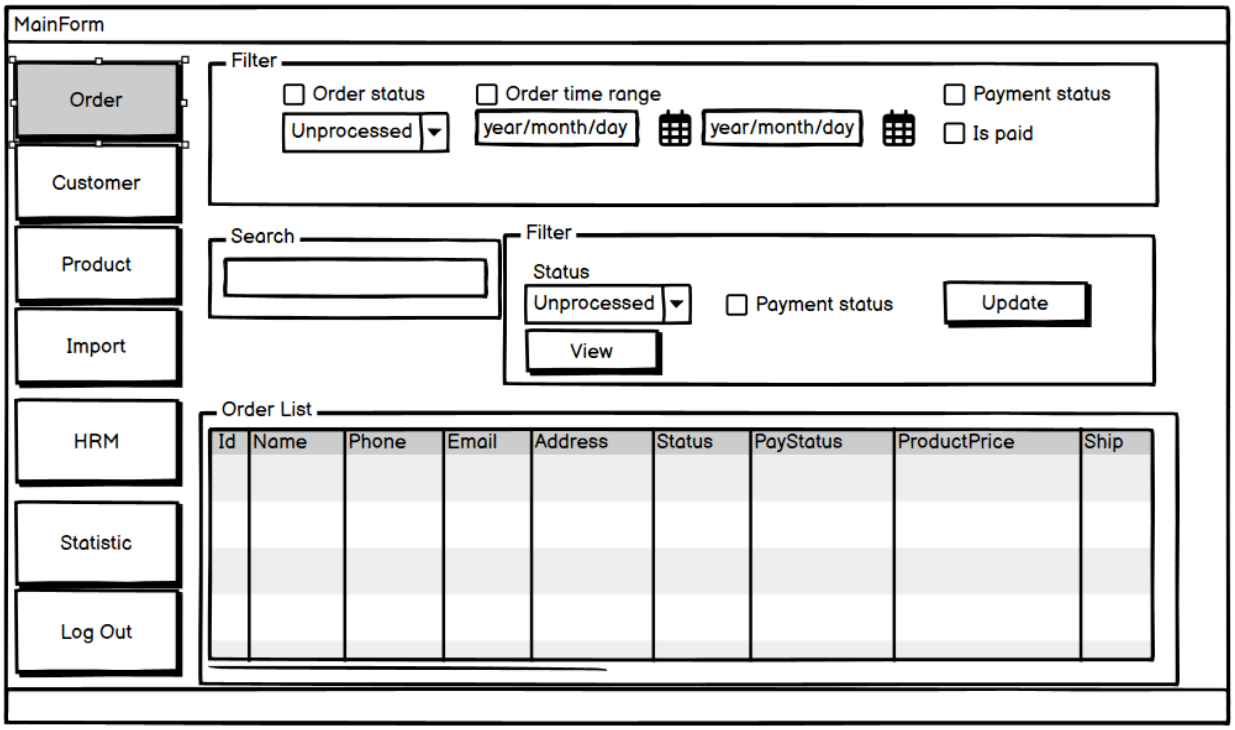


Figure 6 Mockup Order

When you click the view button, the detailed order form will be displayed. In this form permit Print Detail Order you choose

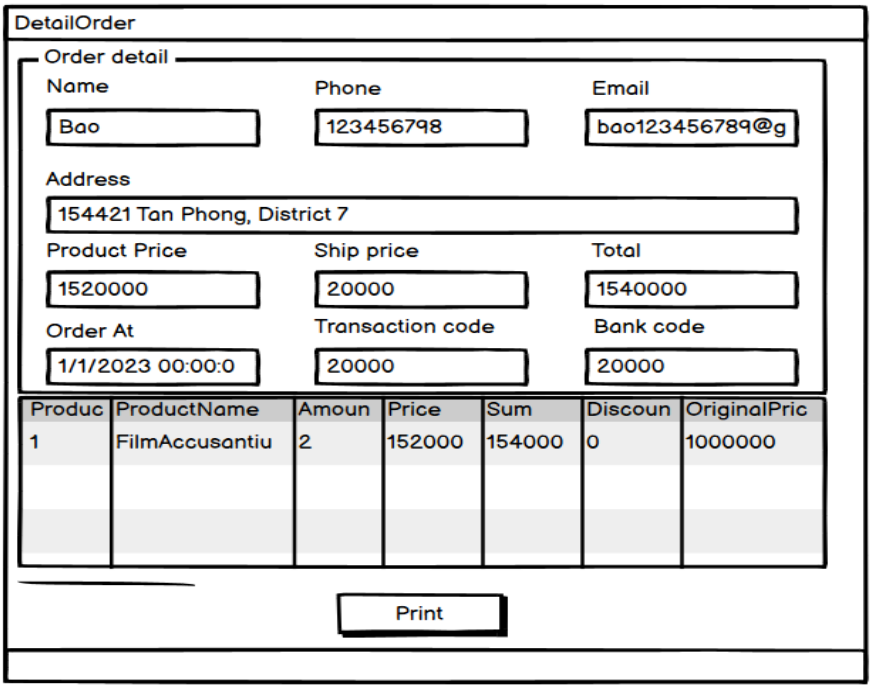


Figure 7: Mockup Order Detail

When you press the print button, the PDF file will be printed

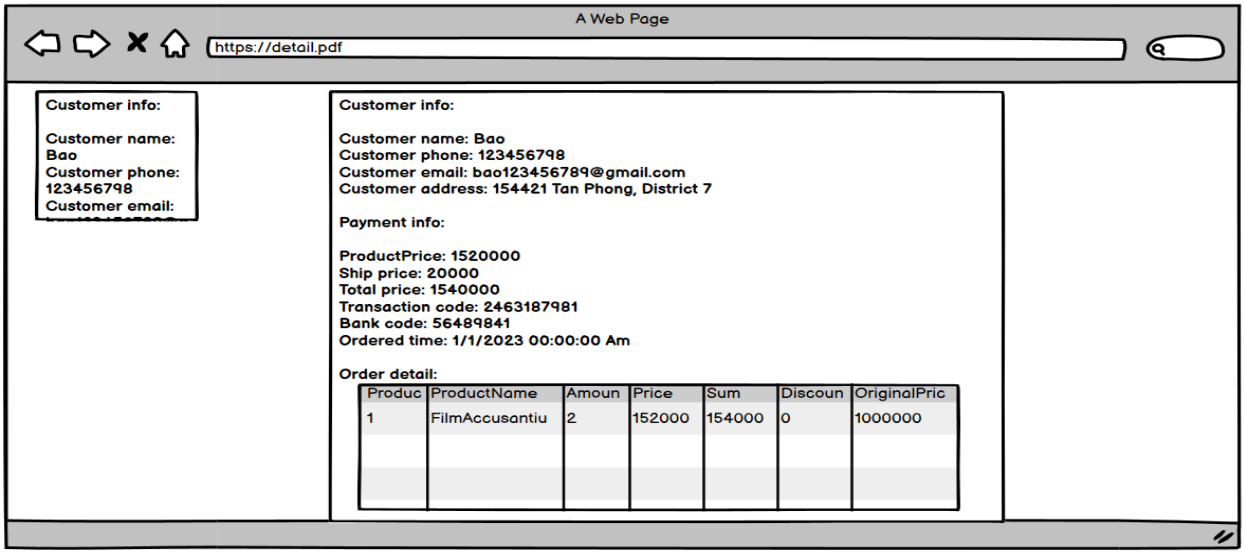


Figure 8 Mockup Print Order

When you click on the button Customer, form customer will be display in this form
When you the click button Initialize you can create new customer detail by enter name, phone, email, address.
Beside you can change password or update information

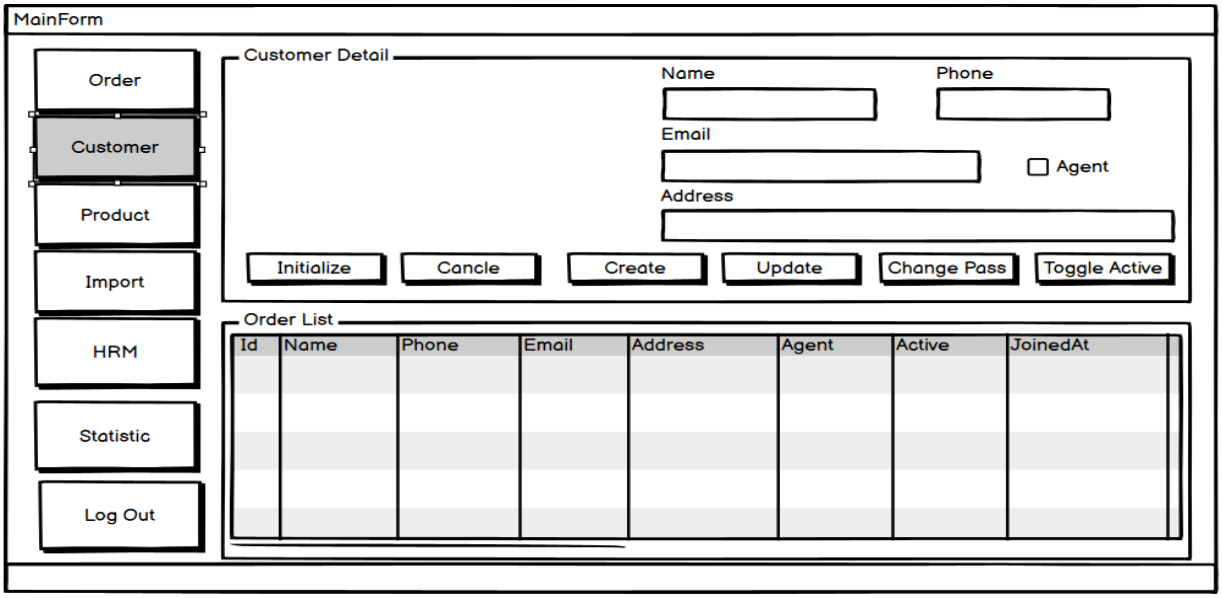


Figure 9 Mockup Manage Customer

When you click on the button Customer, the form customer will be displayed in this form When you click the button Initialize you can create new product detail by enter name, phone, mail, and address.
Besides you can change the name, image, or description by clicking on the update button. And remove the product by clicking on delete button

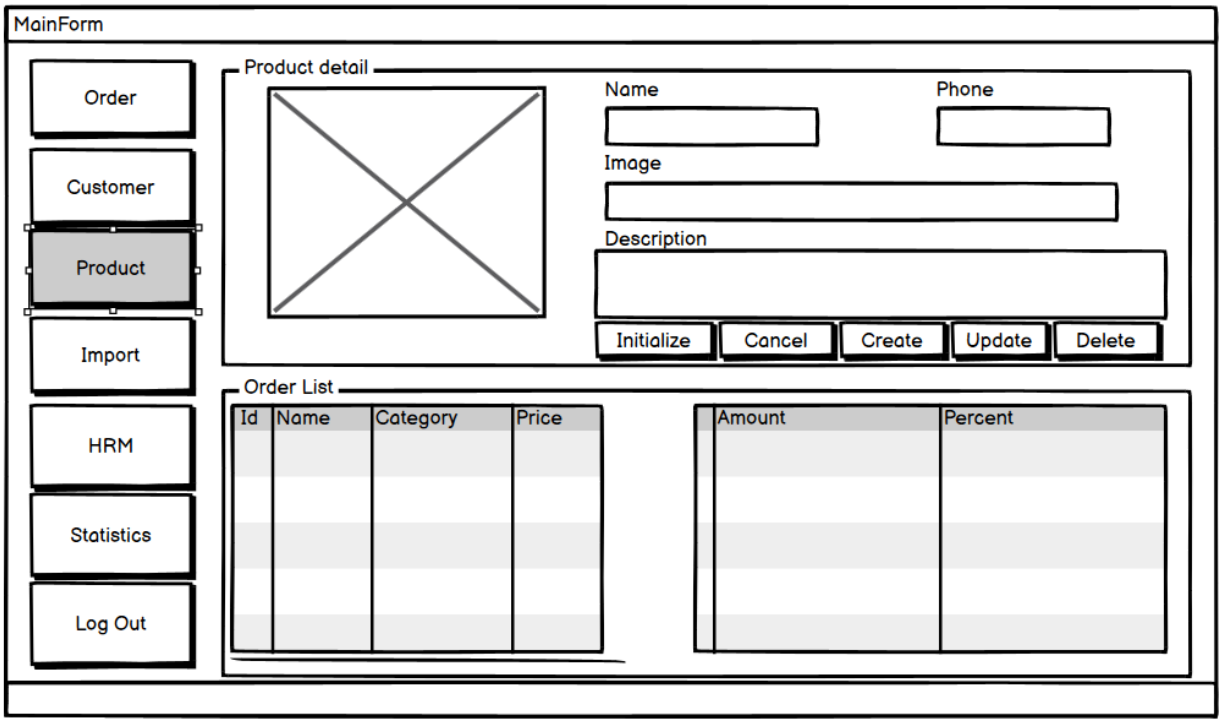


Figure 10 Manage Product

When you click on the DataGrid view below you can turn on edit Discount form.

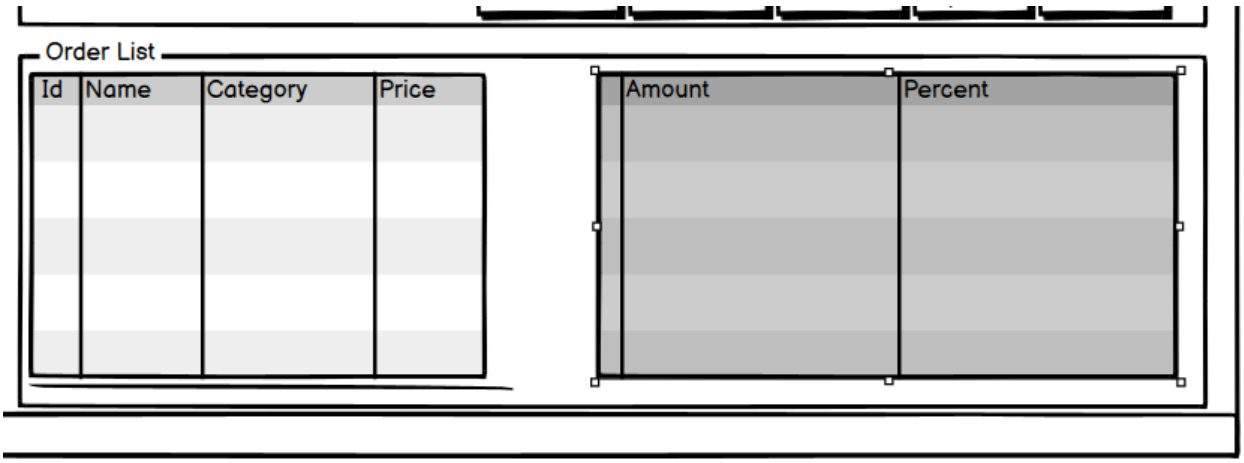


Figure 11Mockup Manage Discount

When you click on the button Import, the form Import will be displayed in this form. When you enter product id, amount, price and clicking on button add it will create new import.

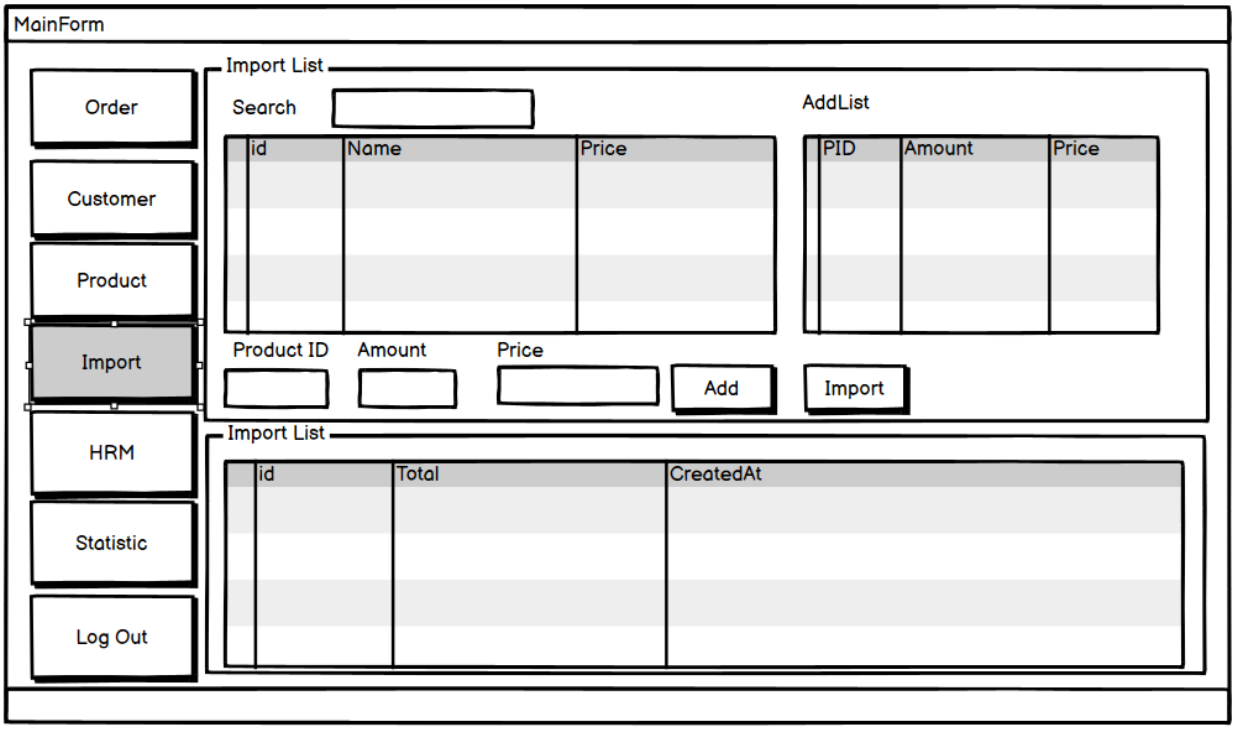


Figure 12 Mockup Manage Import

When you click on button import form detail import will display

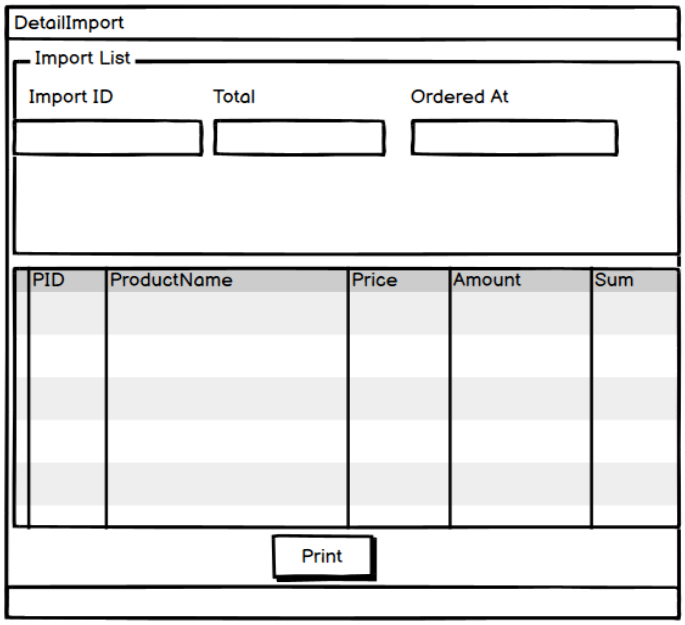


Figure 13Mockup Import detail

When you click on button HRM form statistic will be displayed

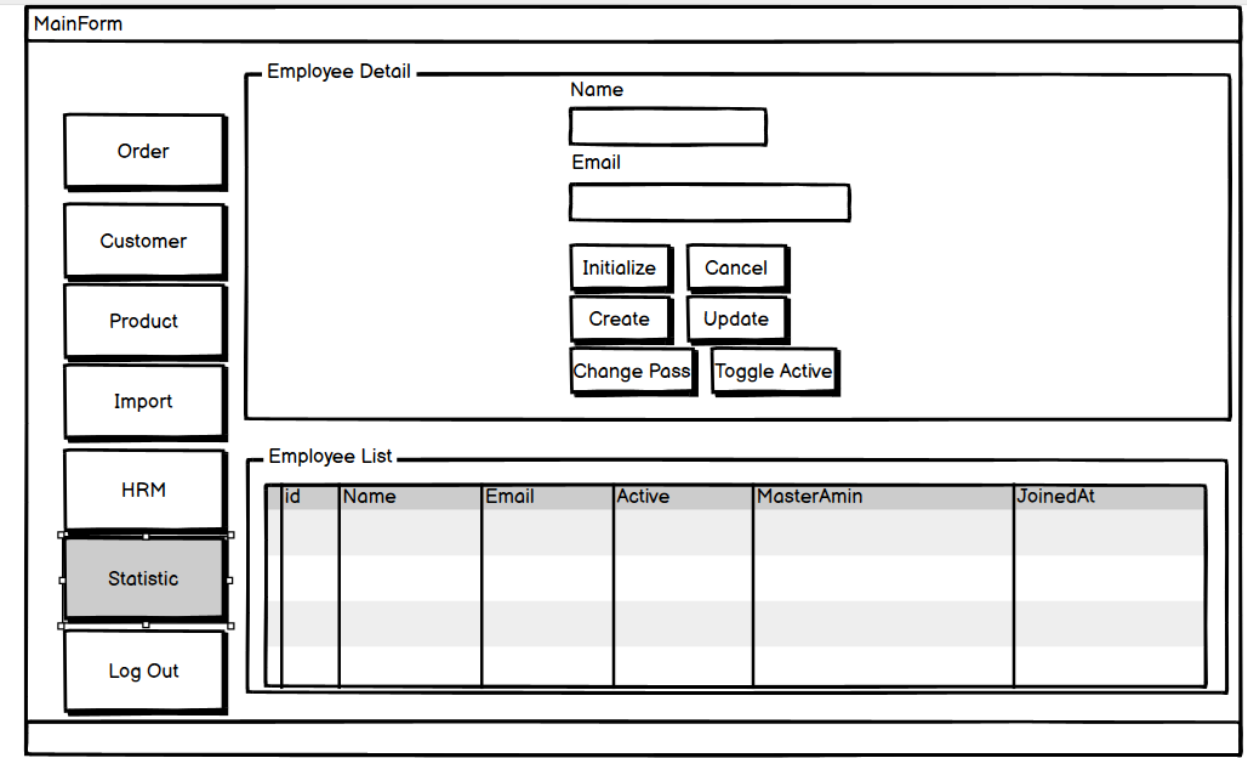


Figure 14: Mockup Manage employee

When you click on button Statistic form statistic will be displayed.

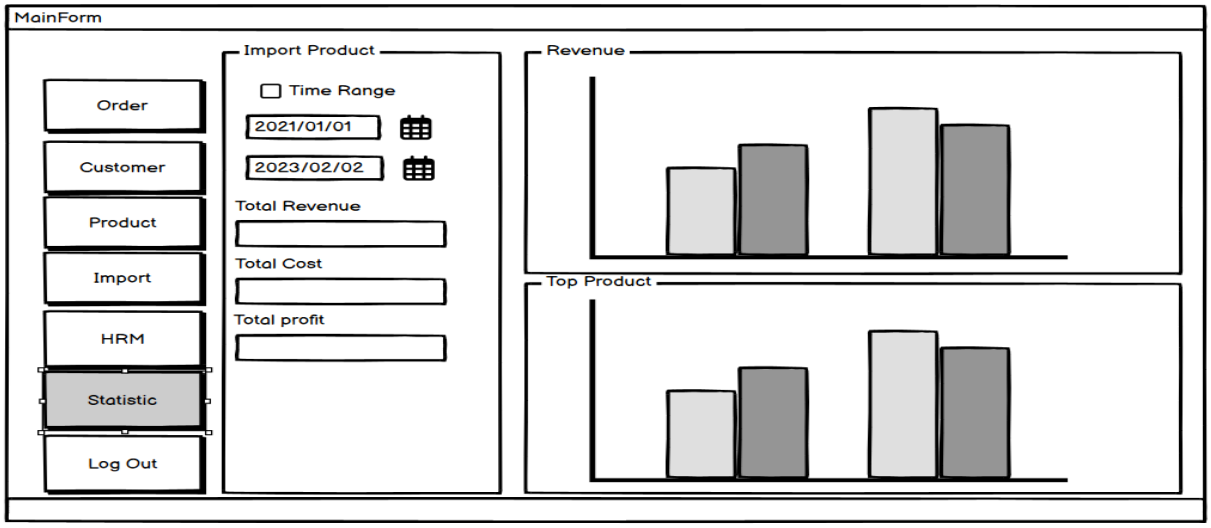


Figure 15Mockup Statistic

5.1.1.2 WebForm

Form Login

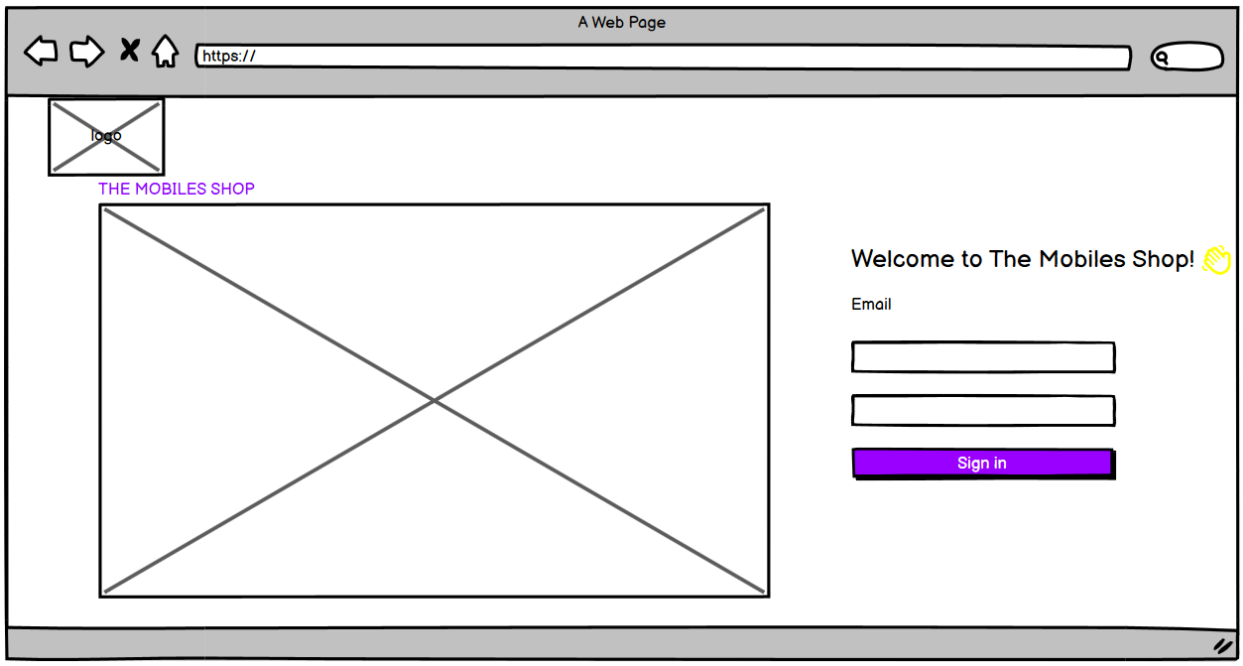


Figure 16 Mockup Login

Form Manage Order

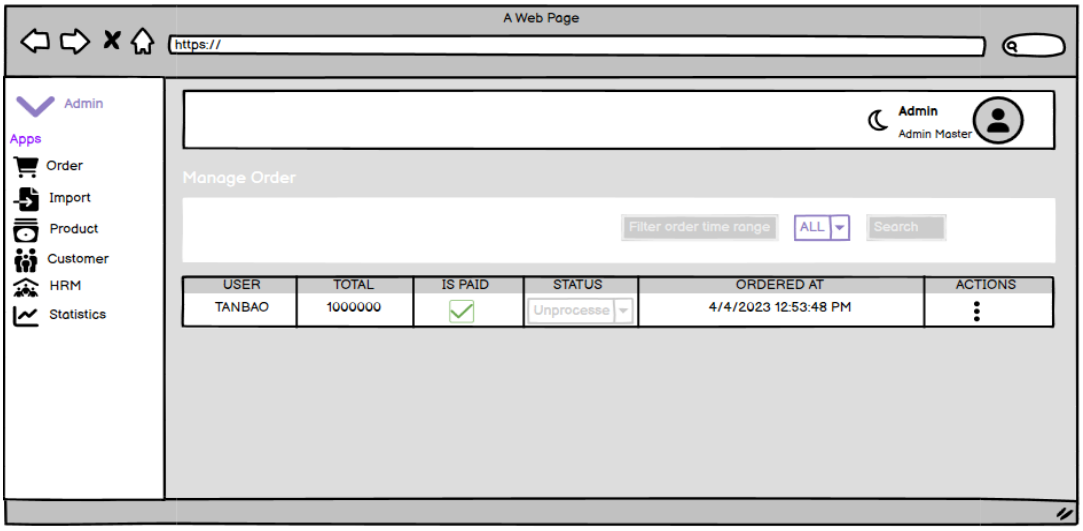


Figure 17: Mockup Manage Order

Form Detail Order

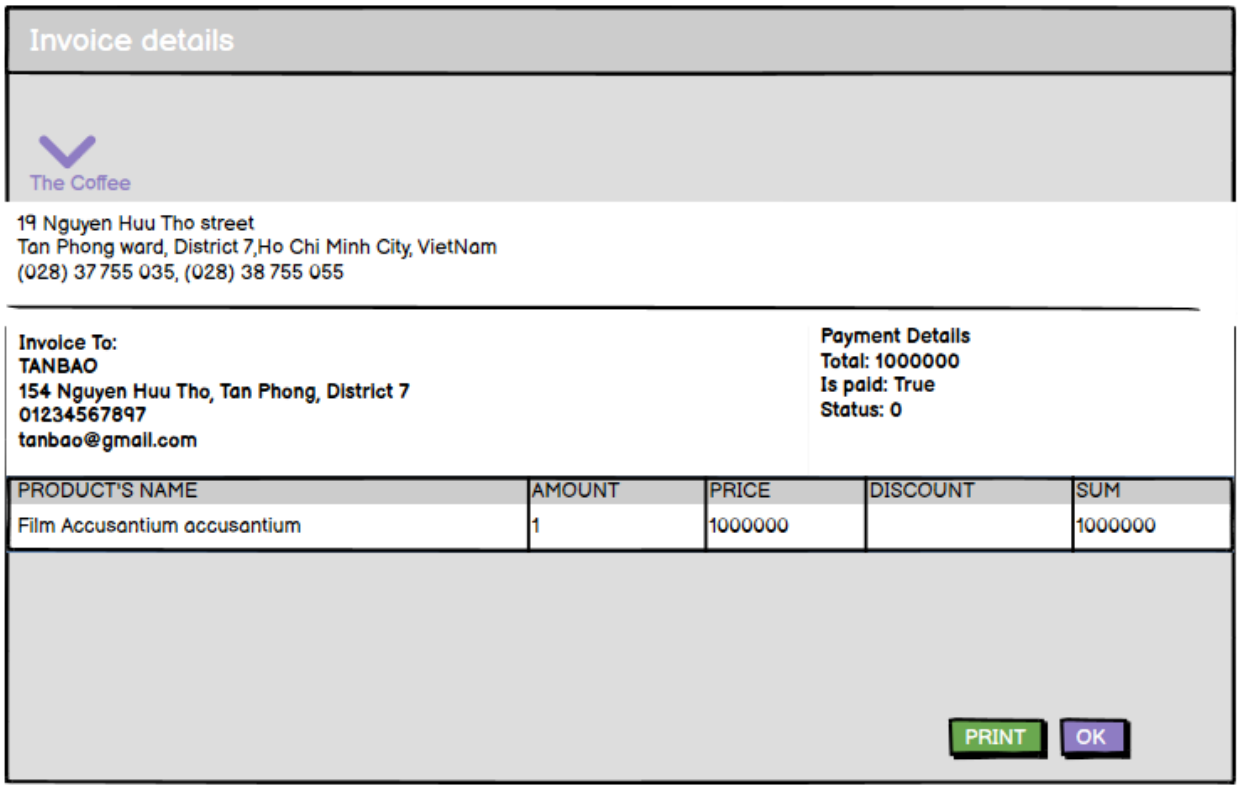


Figure 17: Form Detail Order

File print PDF Order

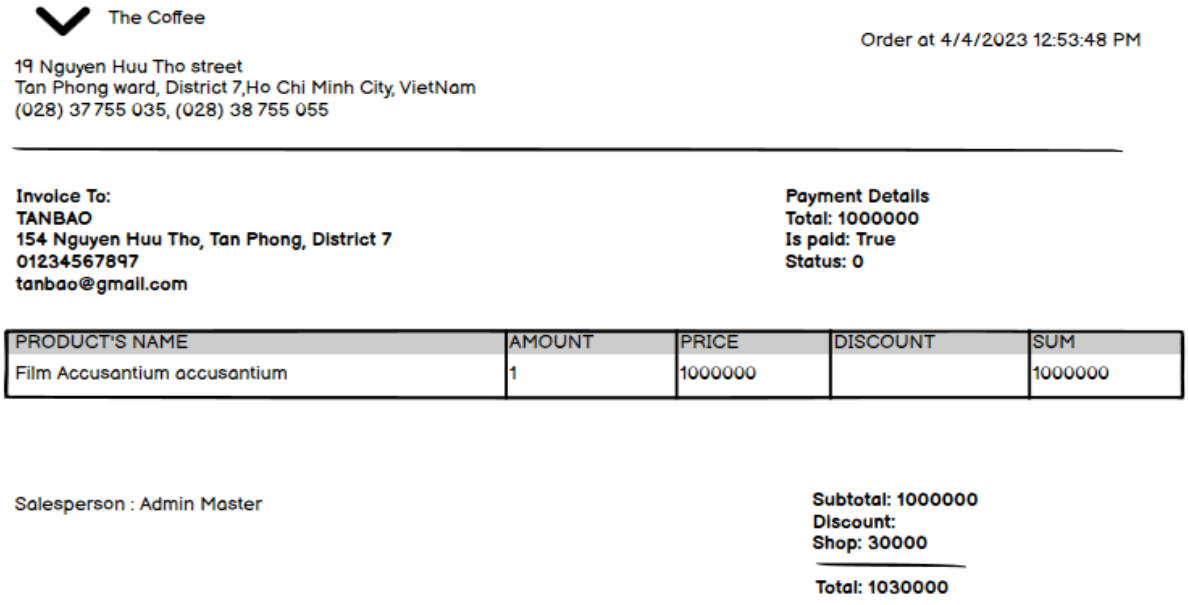


Figure 18: Form Detail Order

Form Manage Import

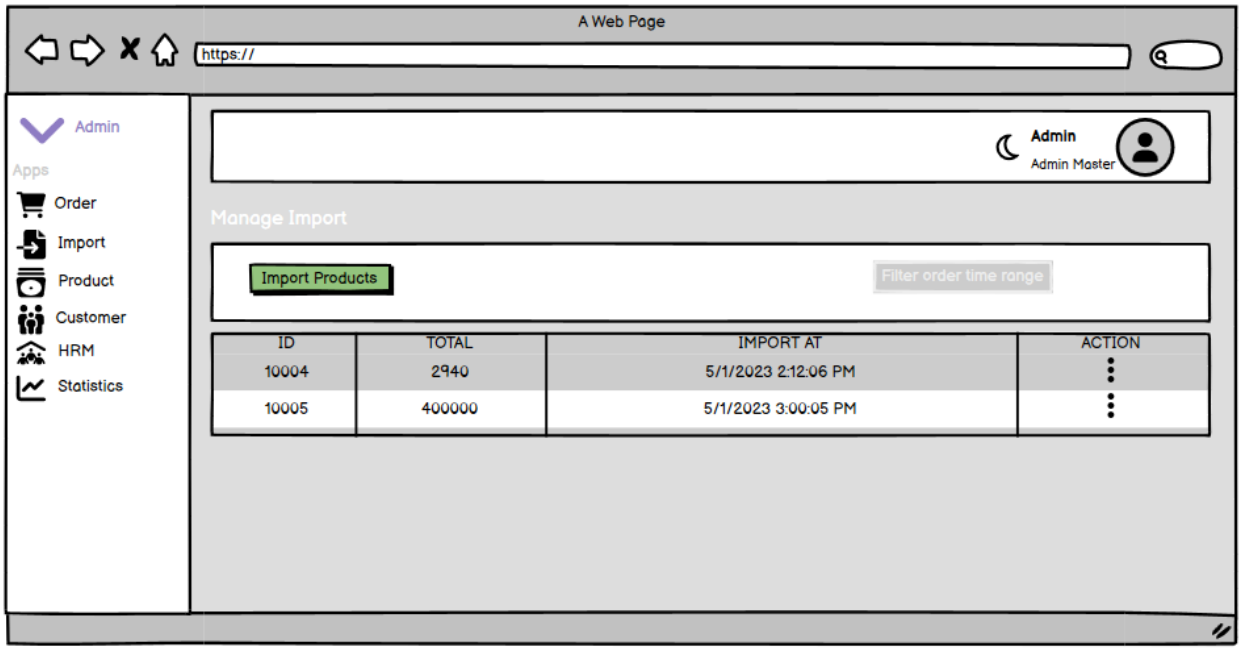


Figure 19: Form Manage Import

Form Import Product

Figure 20: Mockup Form Import Product

Form Import Details

Figure 21: Mockup Form Import Details

Form Print PDF Import Details

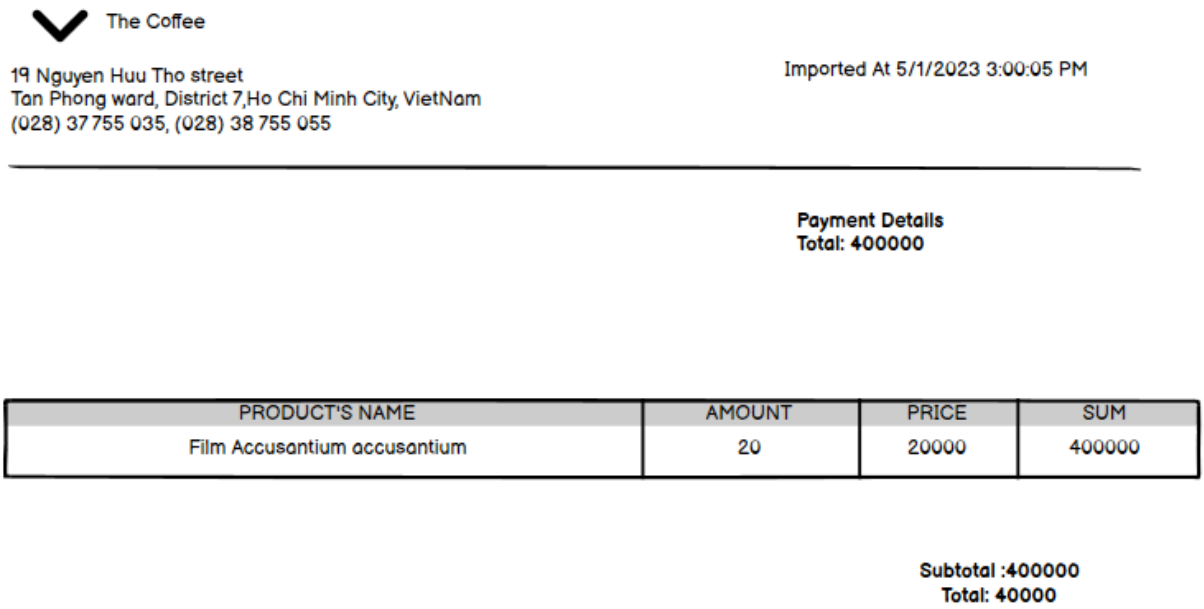


Figure 22: Mockup Form Print PDF Import Details

Form Manage Product

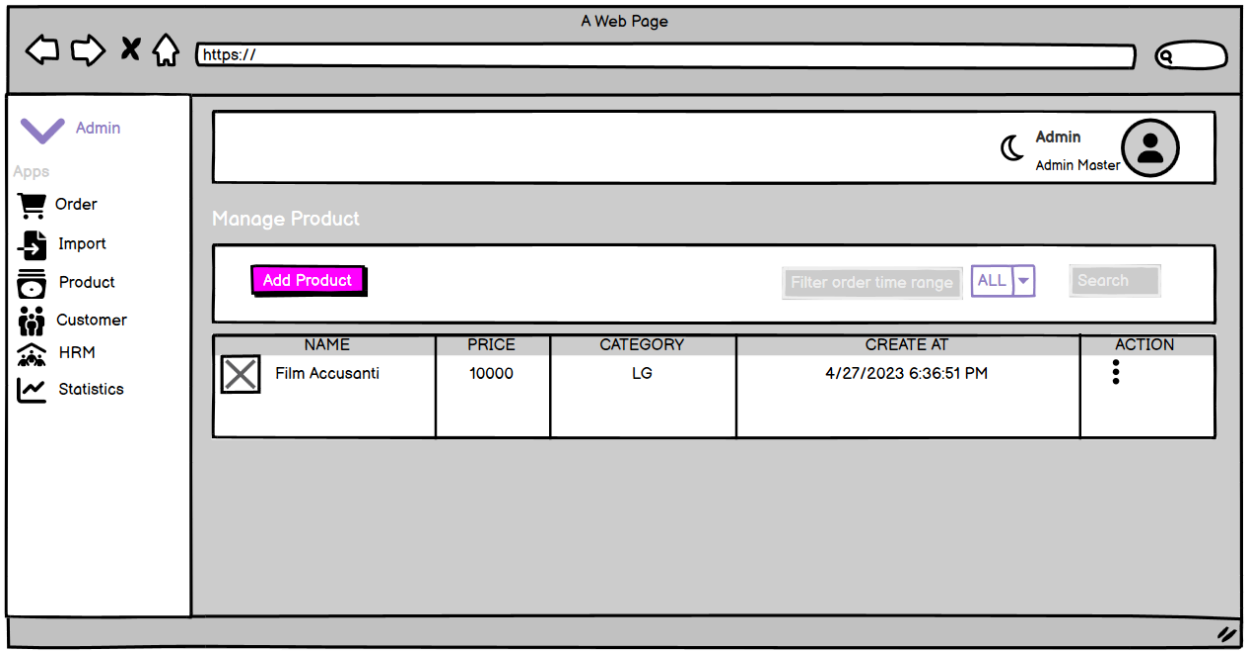


Figure 23: Mockup Form Manage Product

Form Add Product

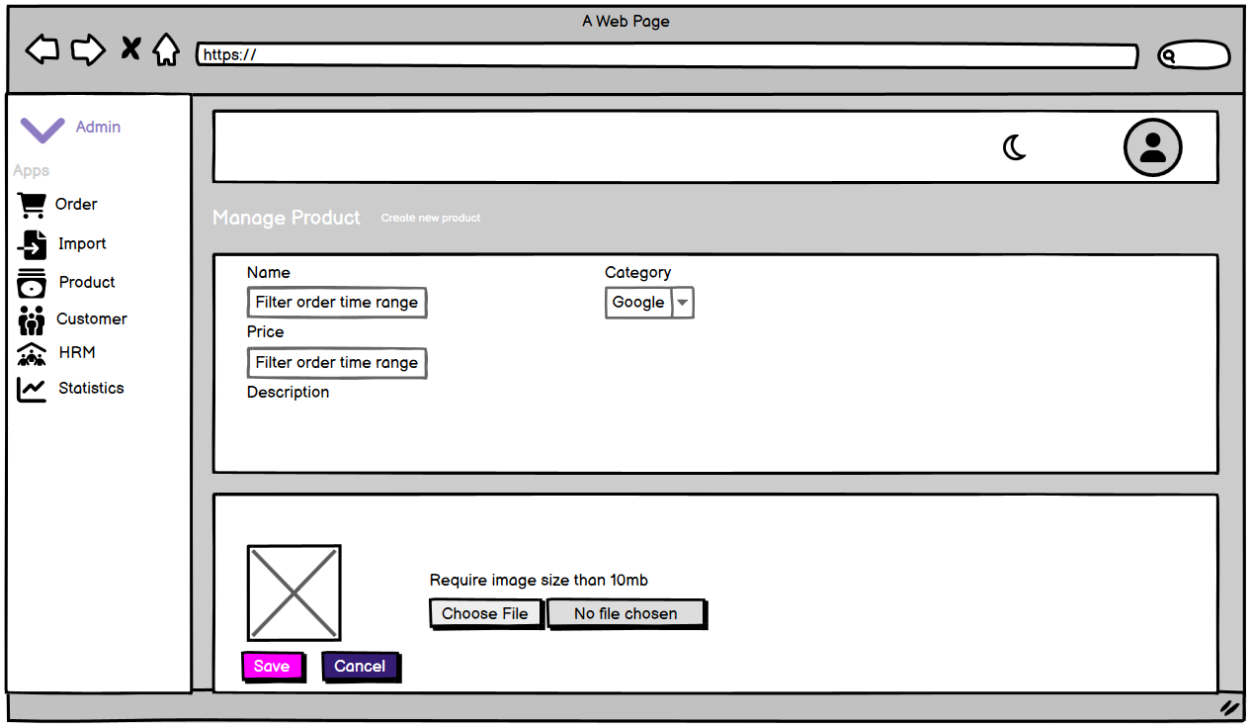


Figure 24: Mockup Form Add Product

Form Customer

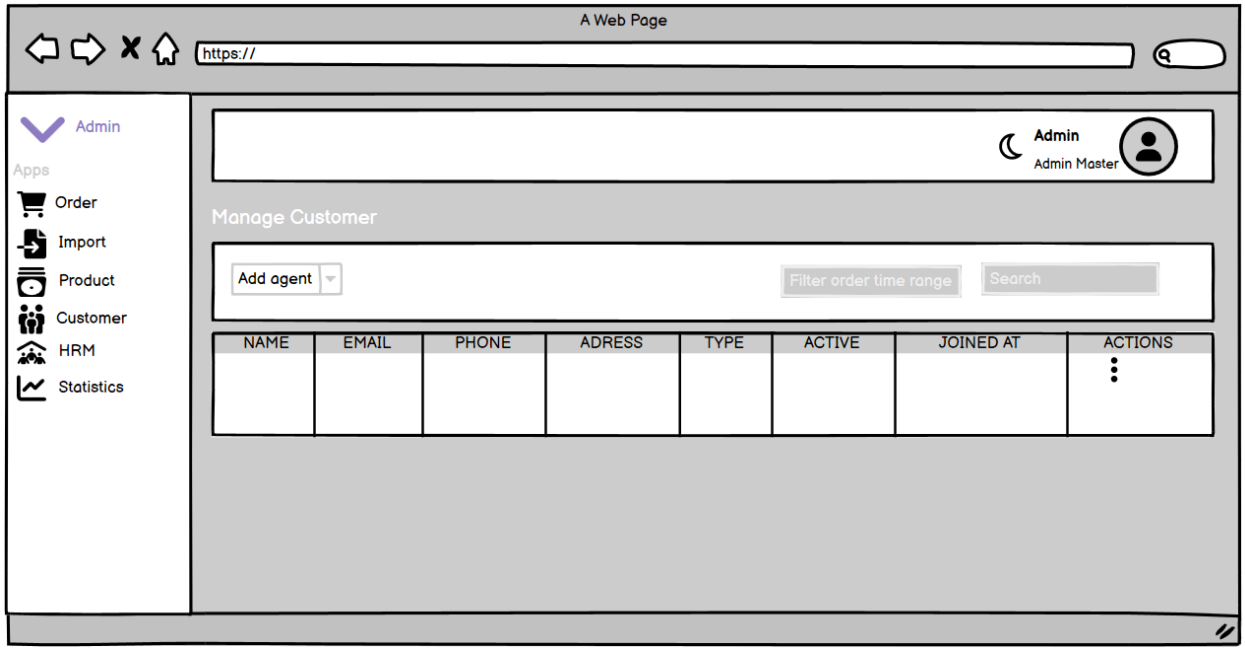


Figure 25: Form Customer

Form Add Agent

Figure 26: Form Add Agent

Form Manage HRM

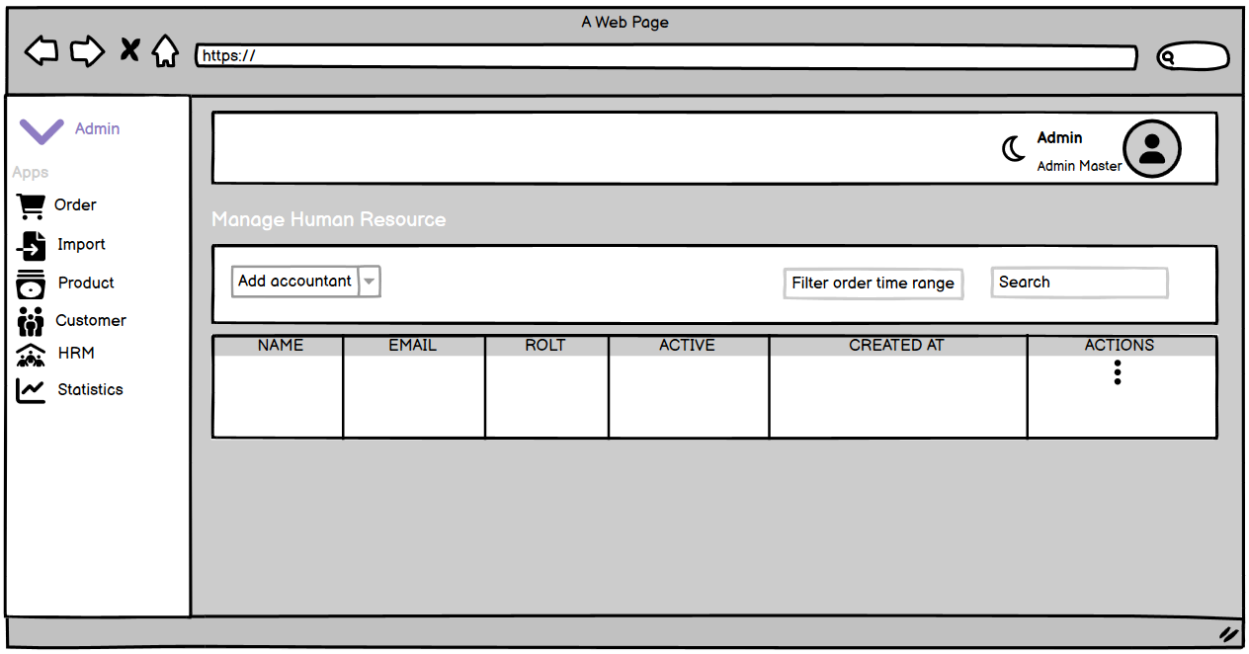


Figure 27: Form Manage HRM

Form Add Staff

Figure 28: Form add staff

Form statistic Product

Figure 29: Form statistic Product

Form statistic Revenue

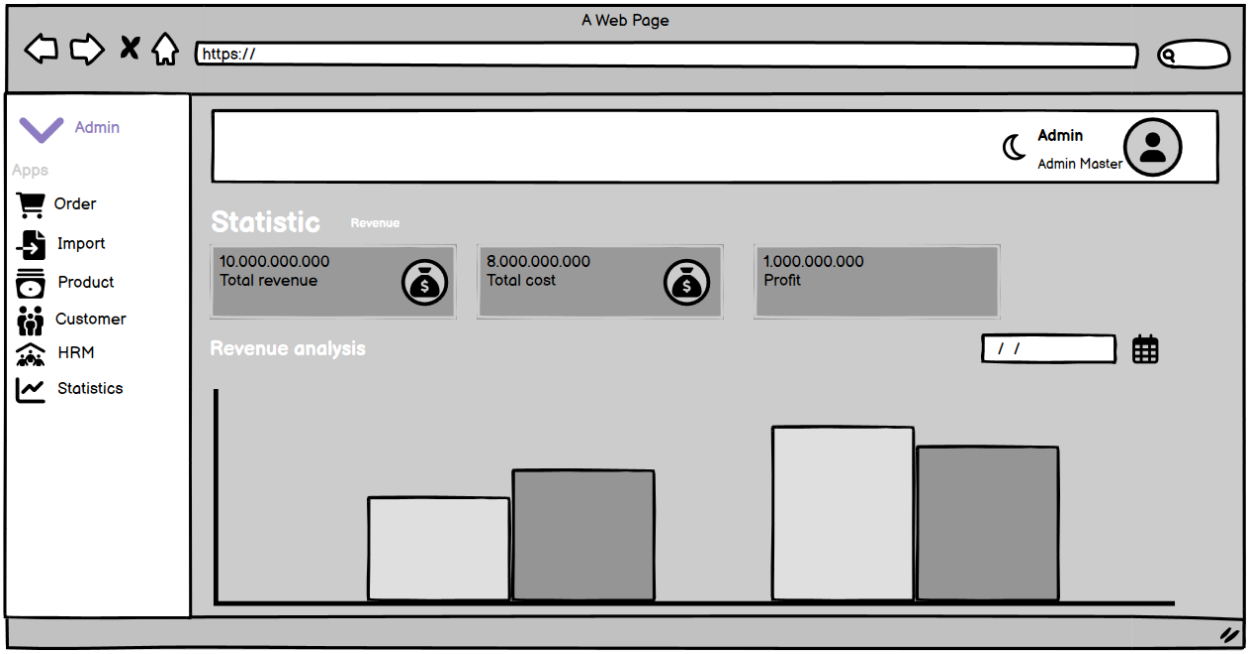


Figure 30: Form statistic Revenue

5.1.1.3 Web B2C

Homepage

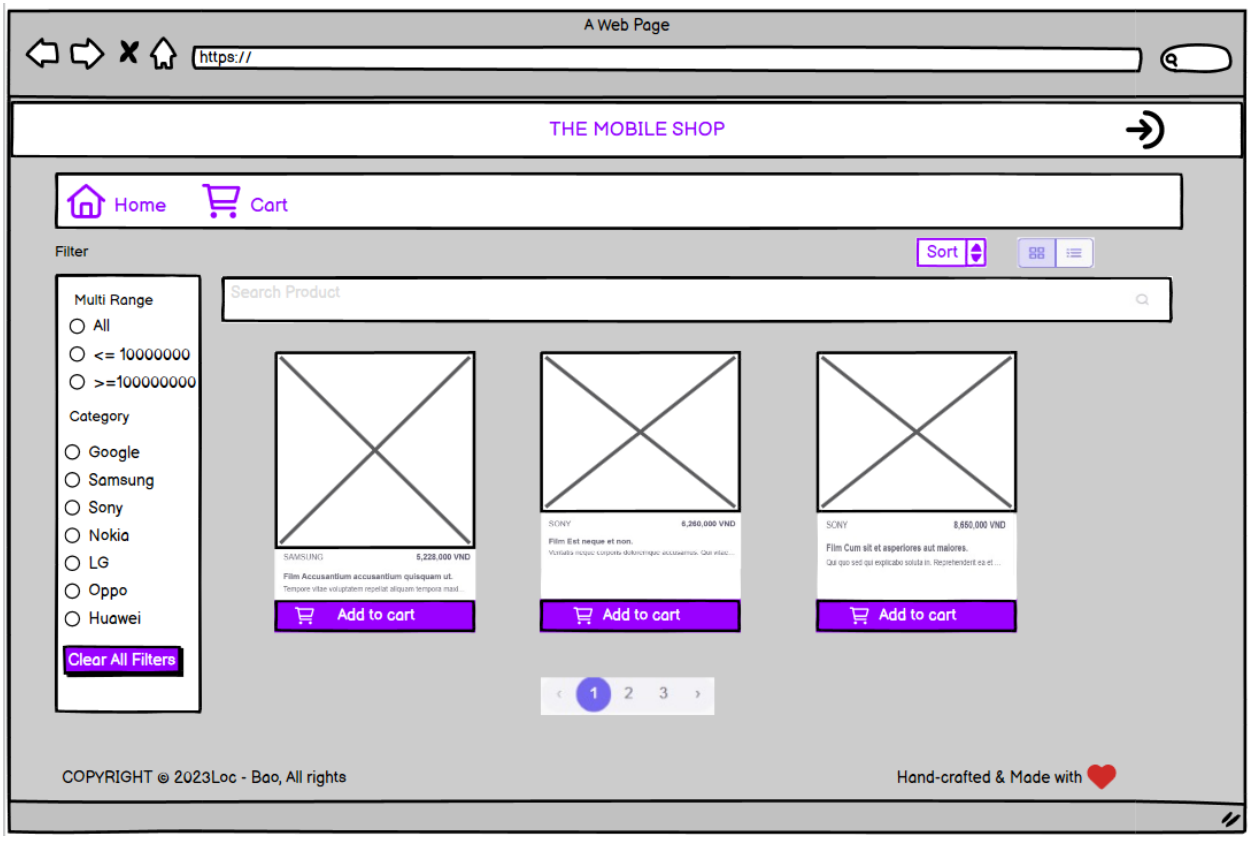


Figure 31: Mockup Homepage

Order History

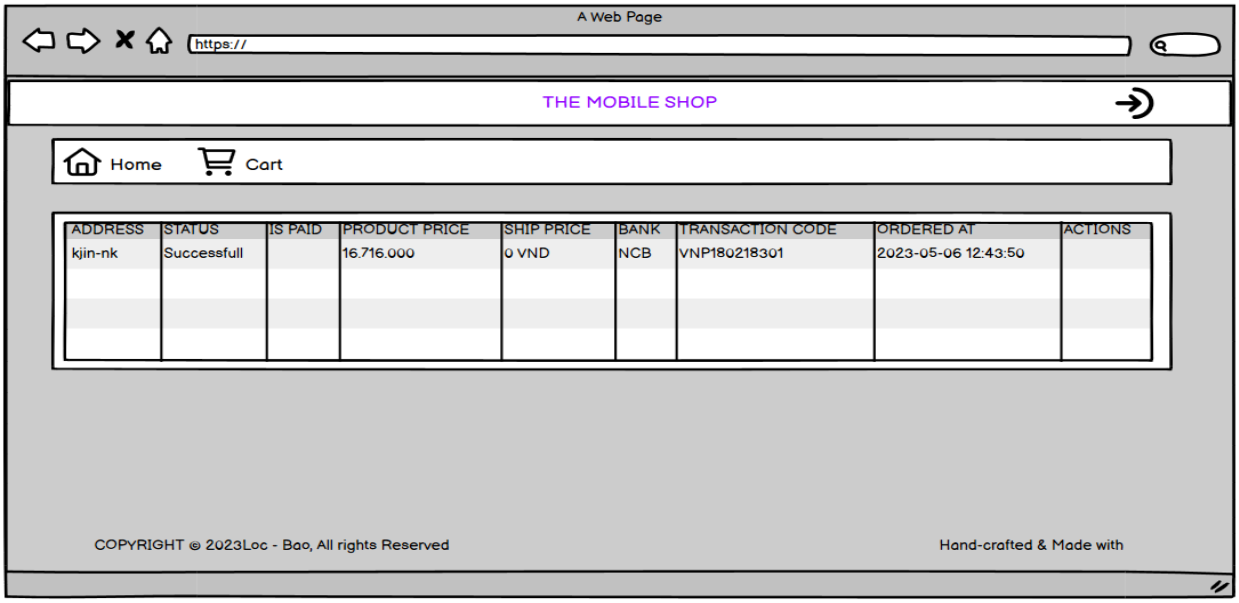


Figure 32: Order history

Detail Product

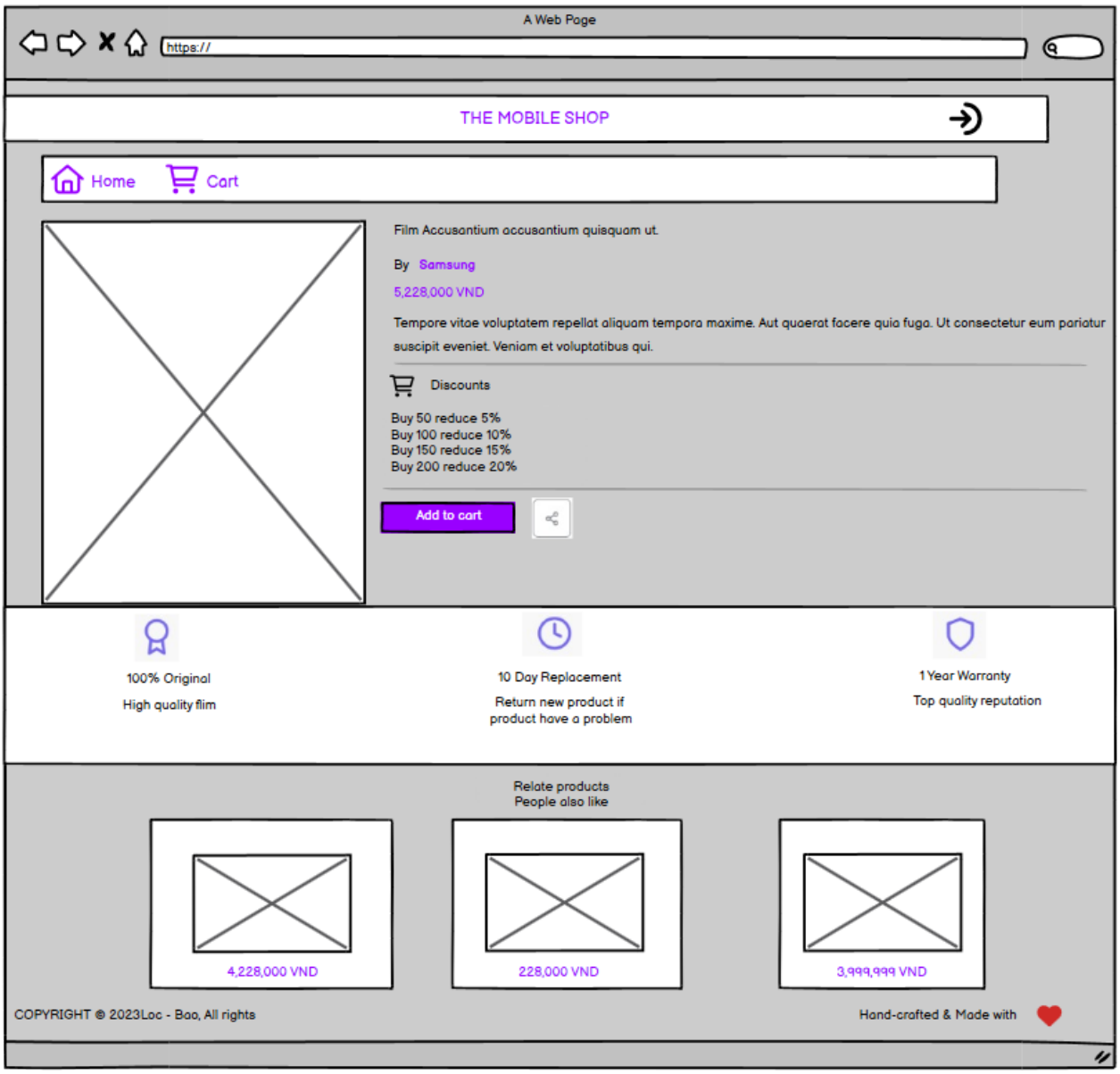


Figure 33: Detail Product

Form Login

Figure 34: Form Login

Cart Page

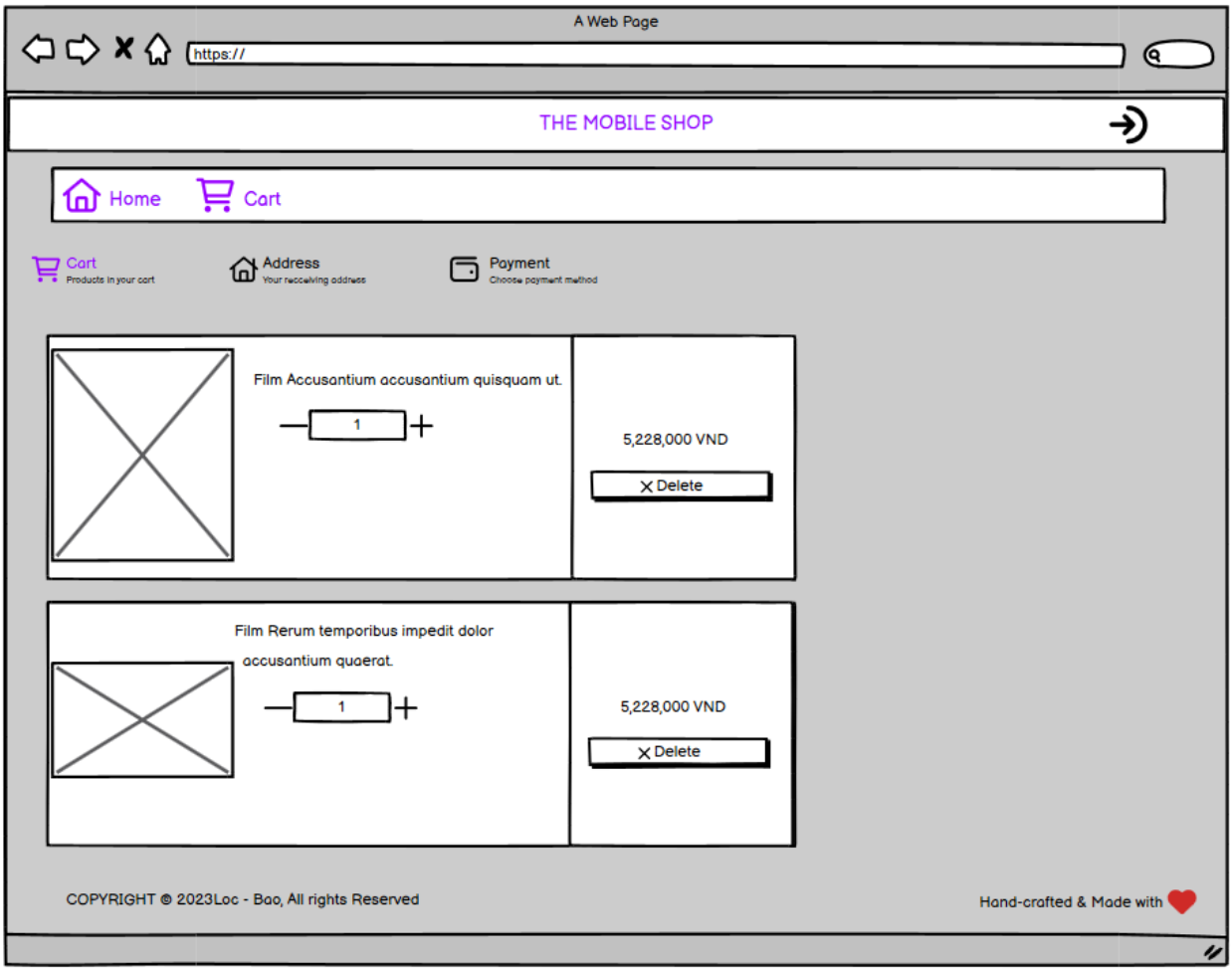


Figure 35: Cart Page

Cart Address

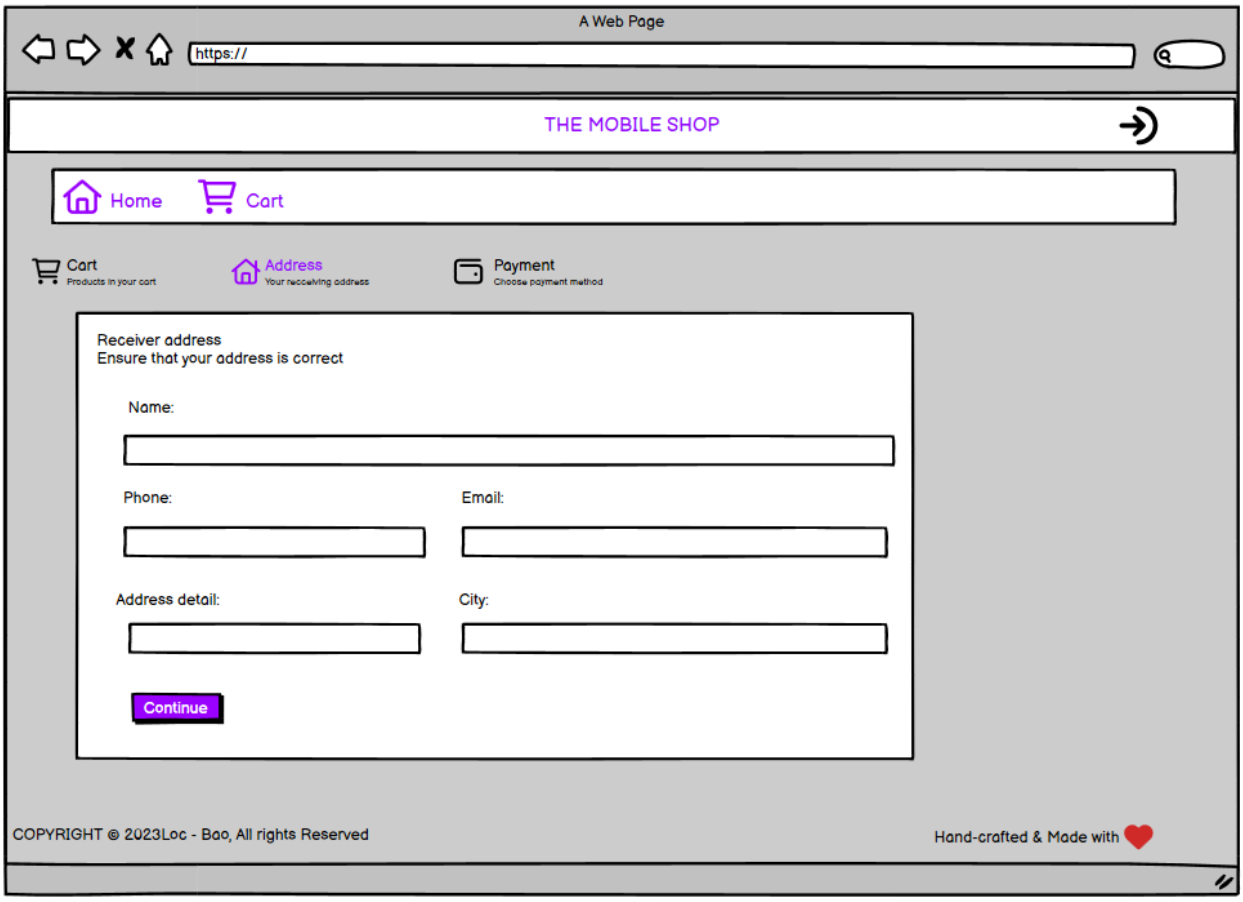


Figure 36: Cart Address

Cart Payment

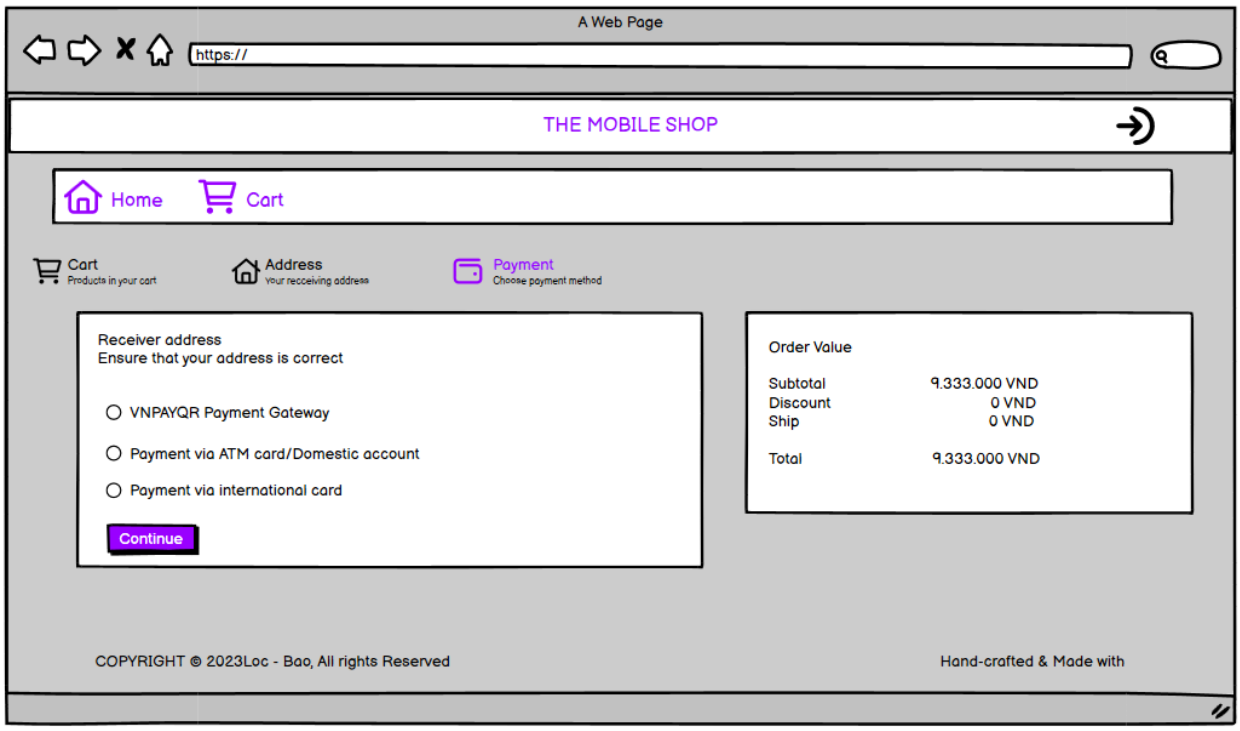
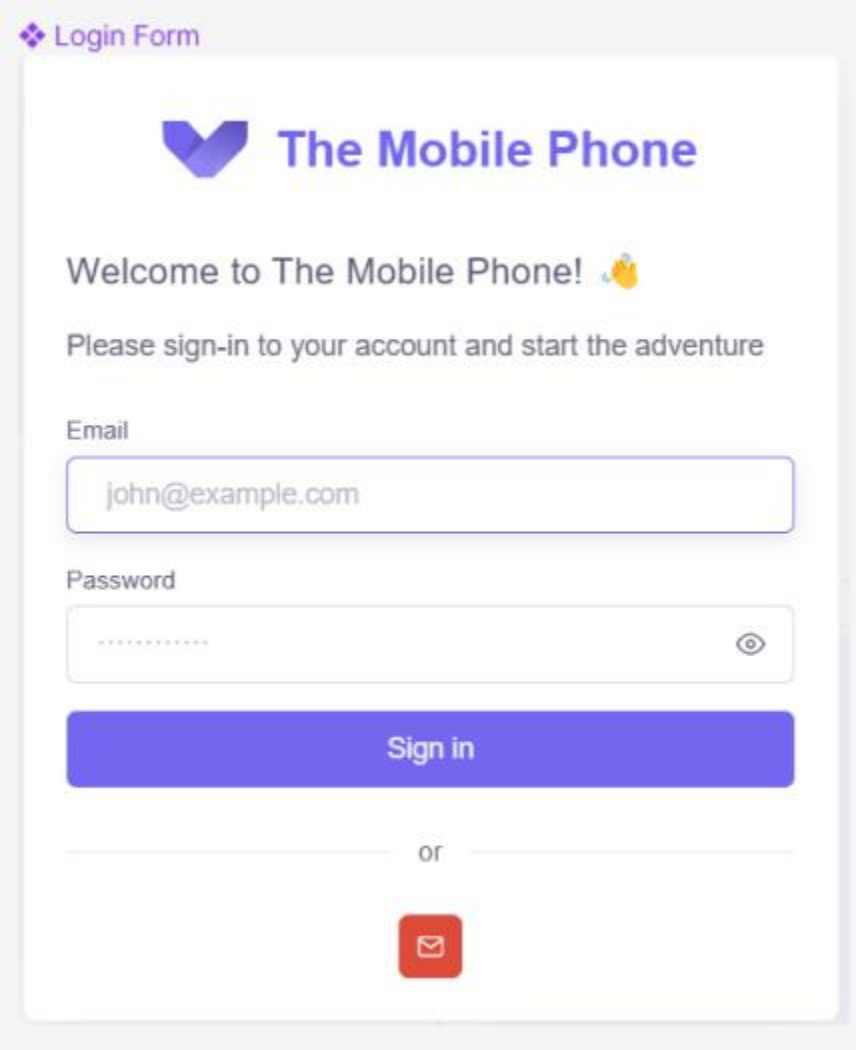


Figure 37: Cart Payment

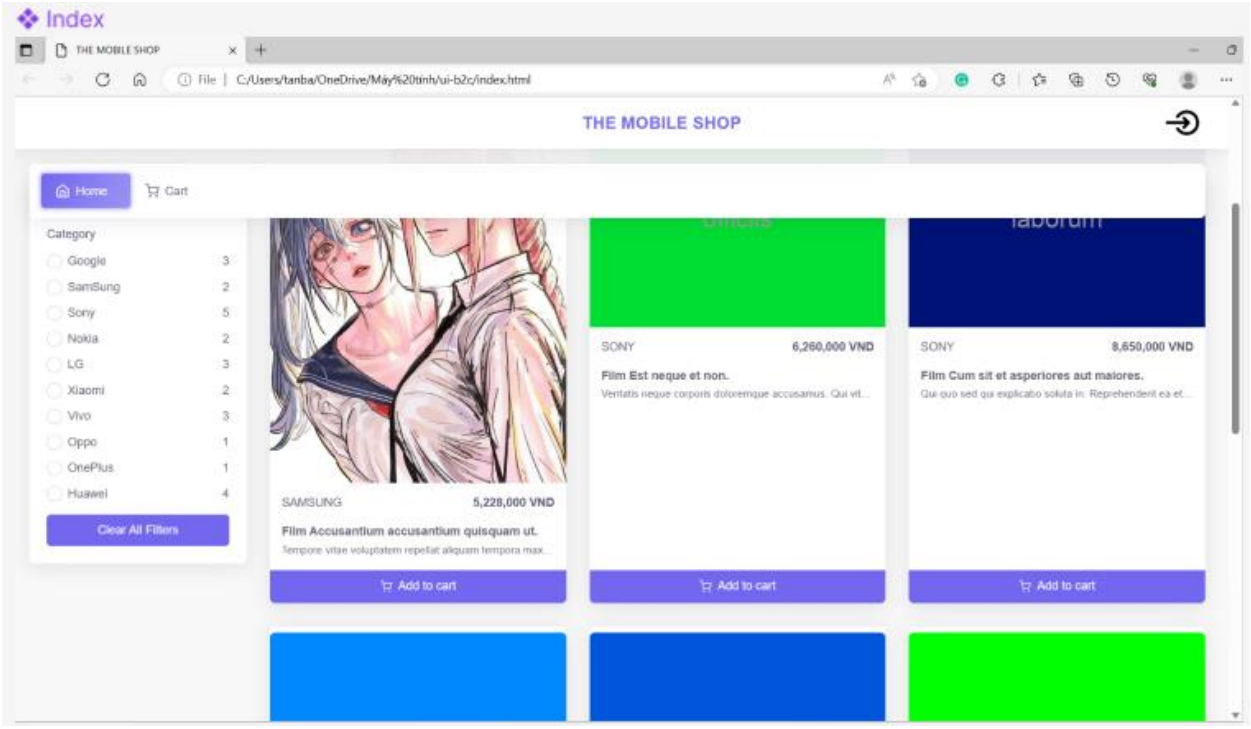
5.1.2 Prototype

5.1.2.1 WebB2C

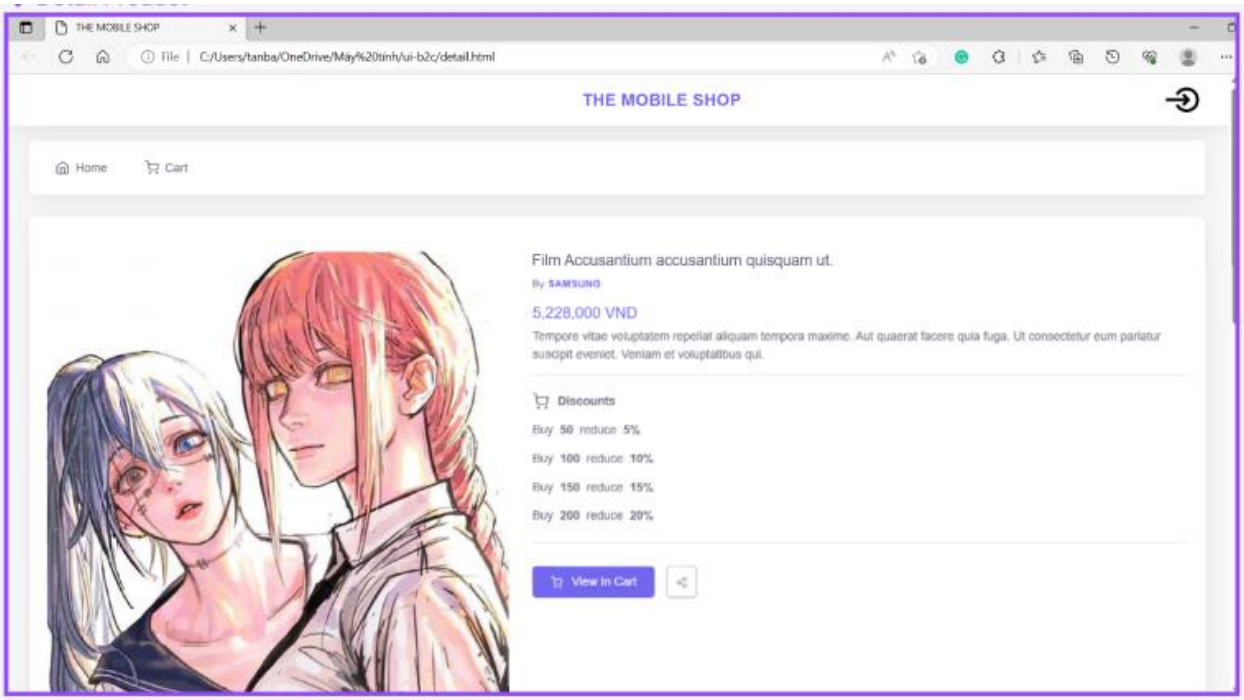
Form Login



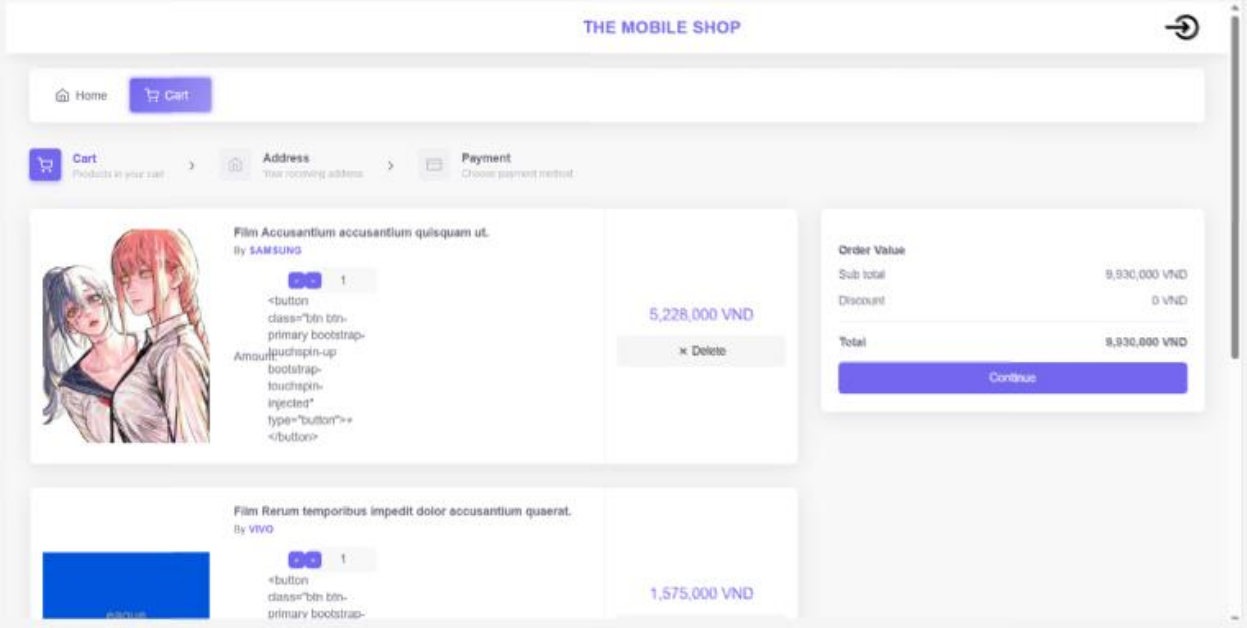
Onclick to Sign in, Index will be displayed



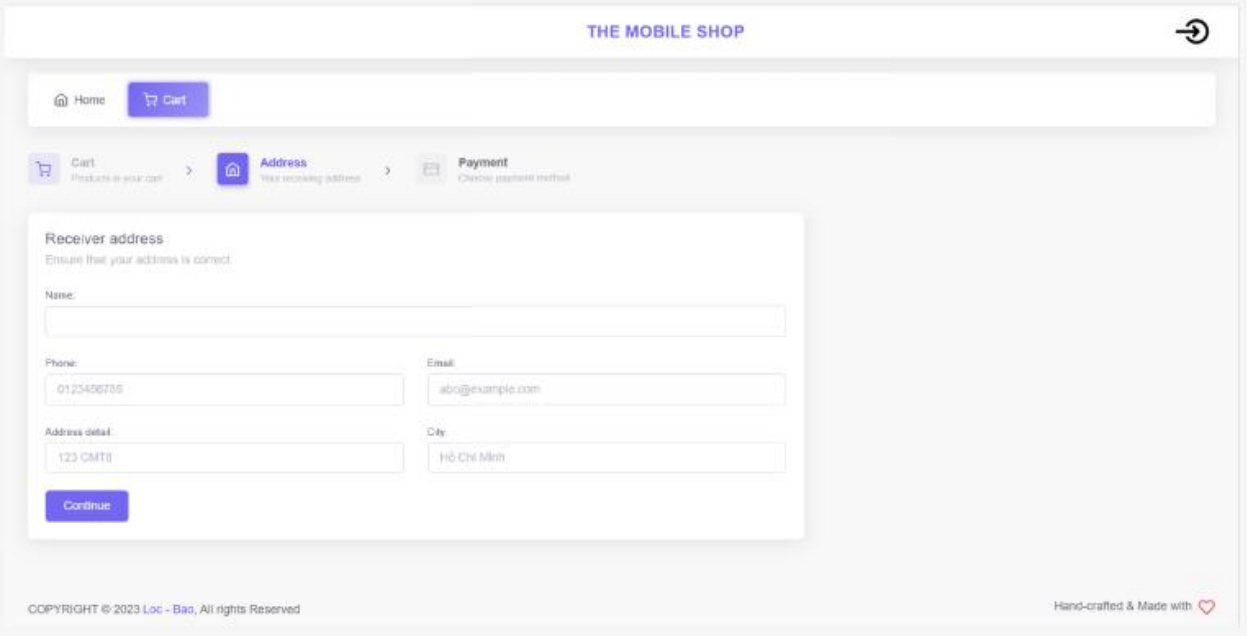
OnClick to Product(image), Detail Product web will be displayed



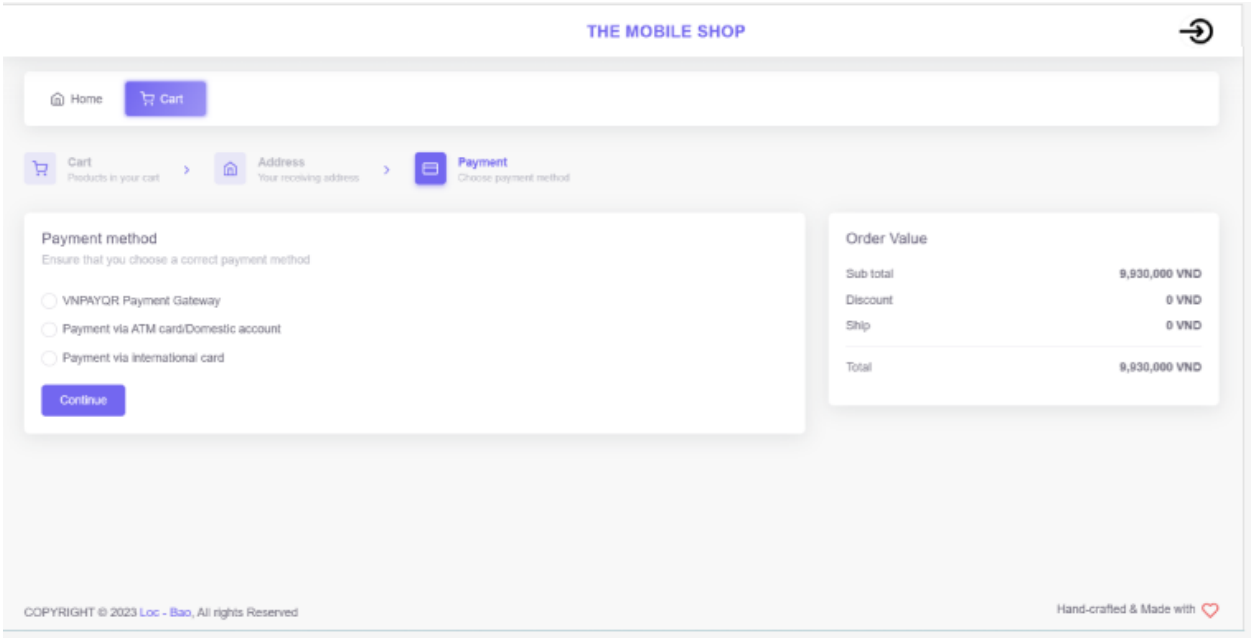
Onclick to button View In Cart, Cart web will be displayed



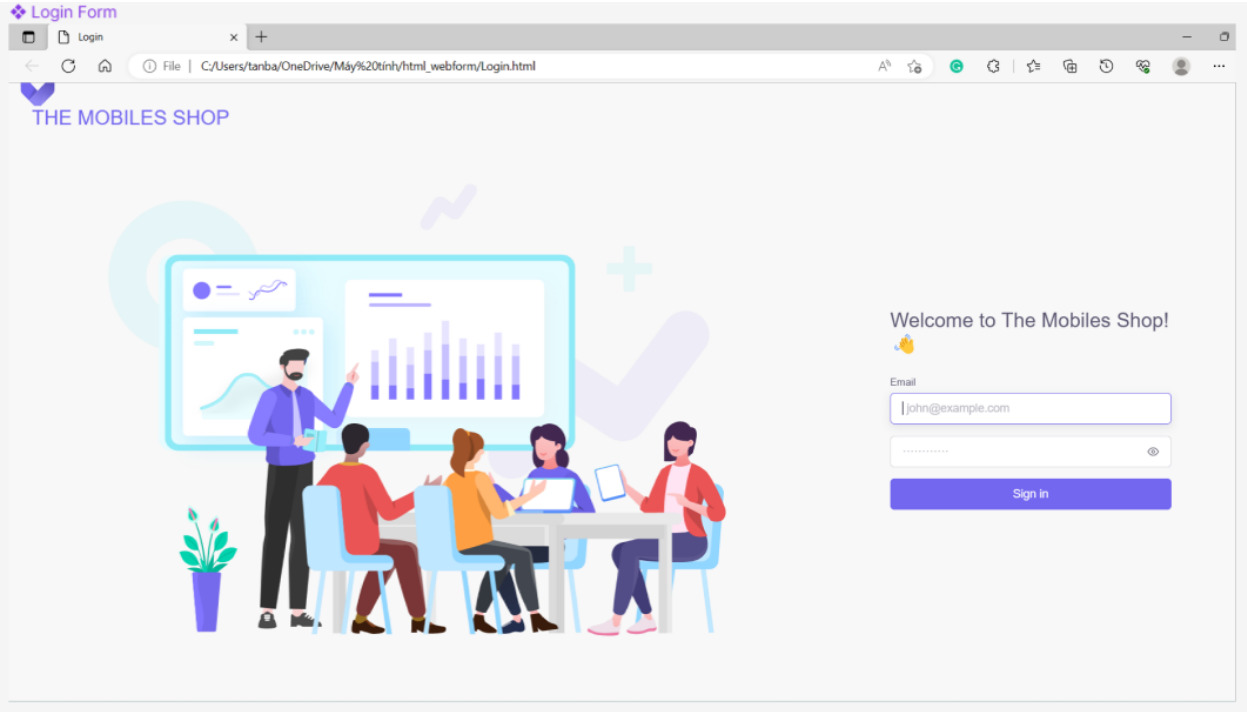
Onclick to Address, Address form will be displayed



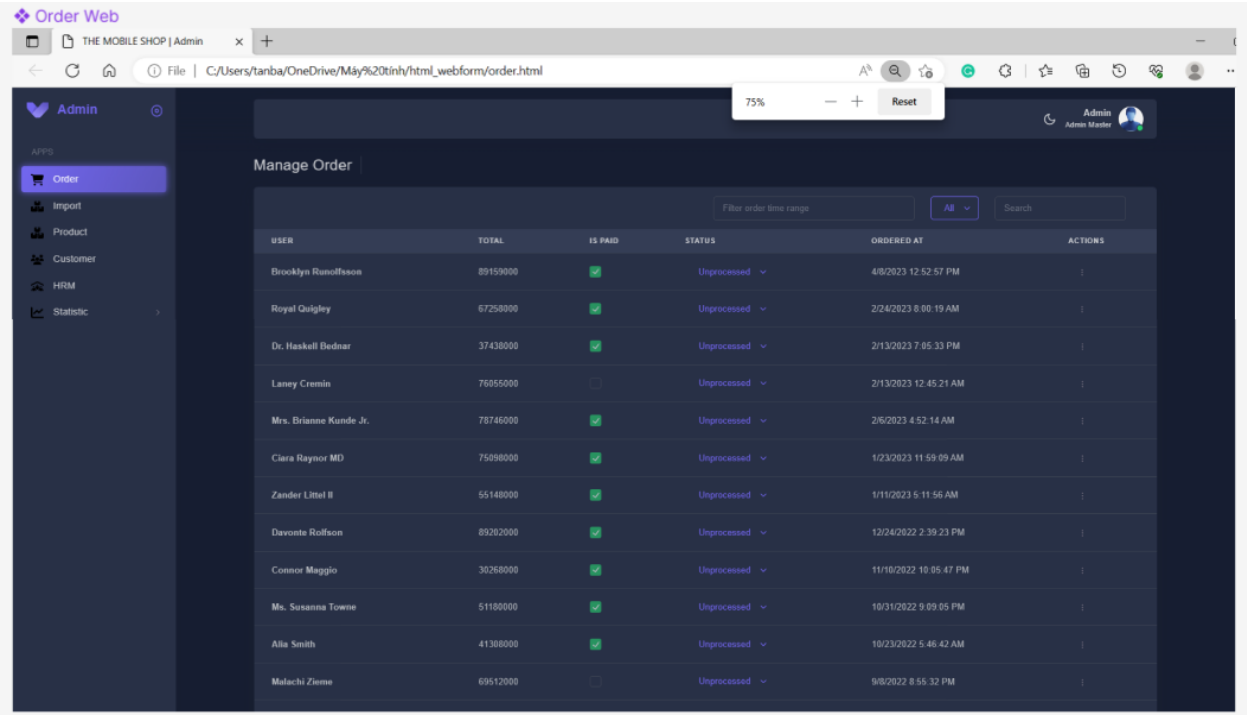
Onclick to Payment, Payment form will be displayed



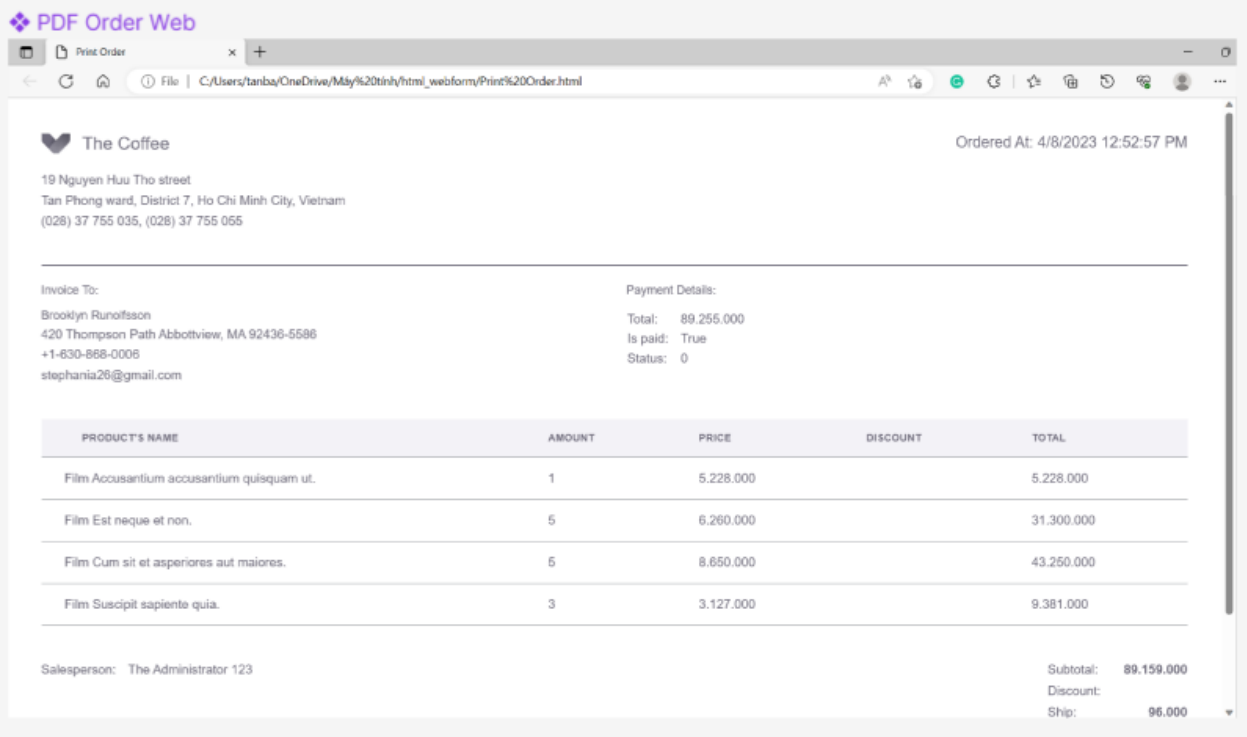
5.1.2.2 Webform



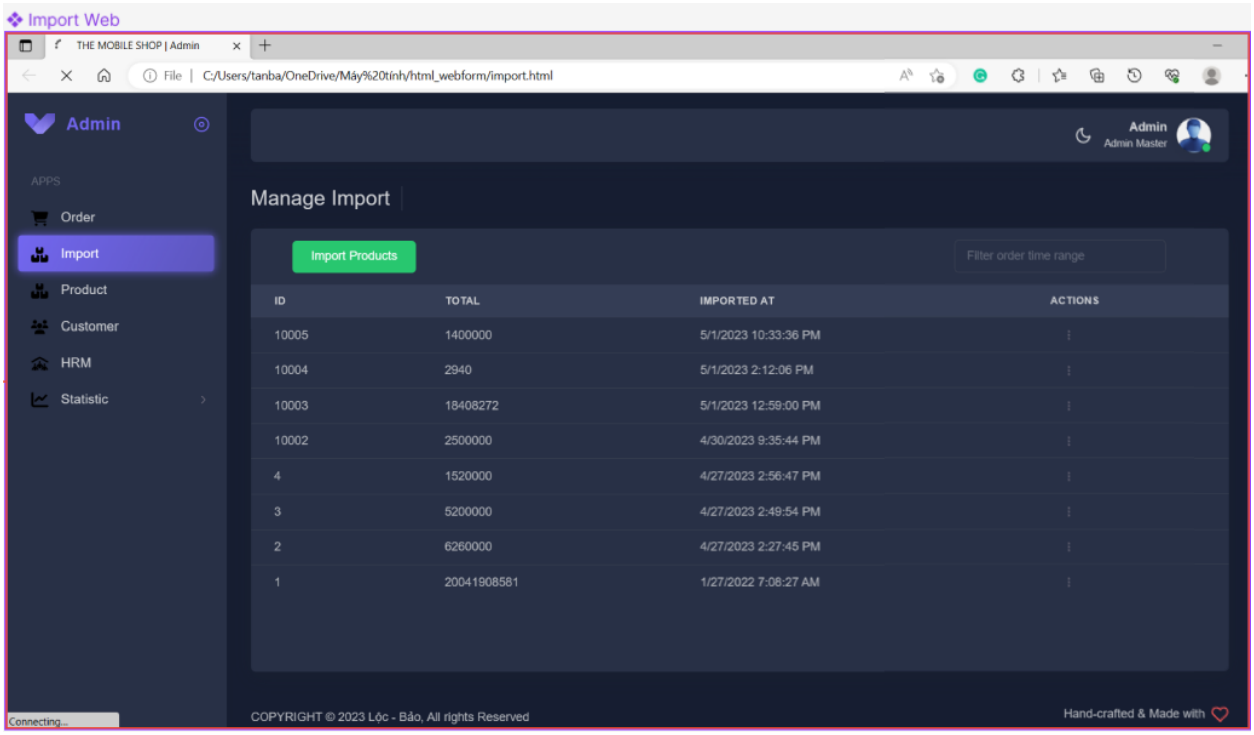
When onclick to Sign in, Order Web will be displayed



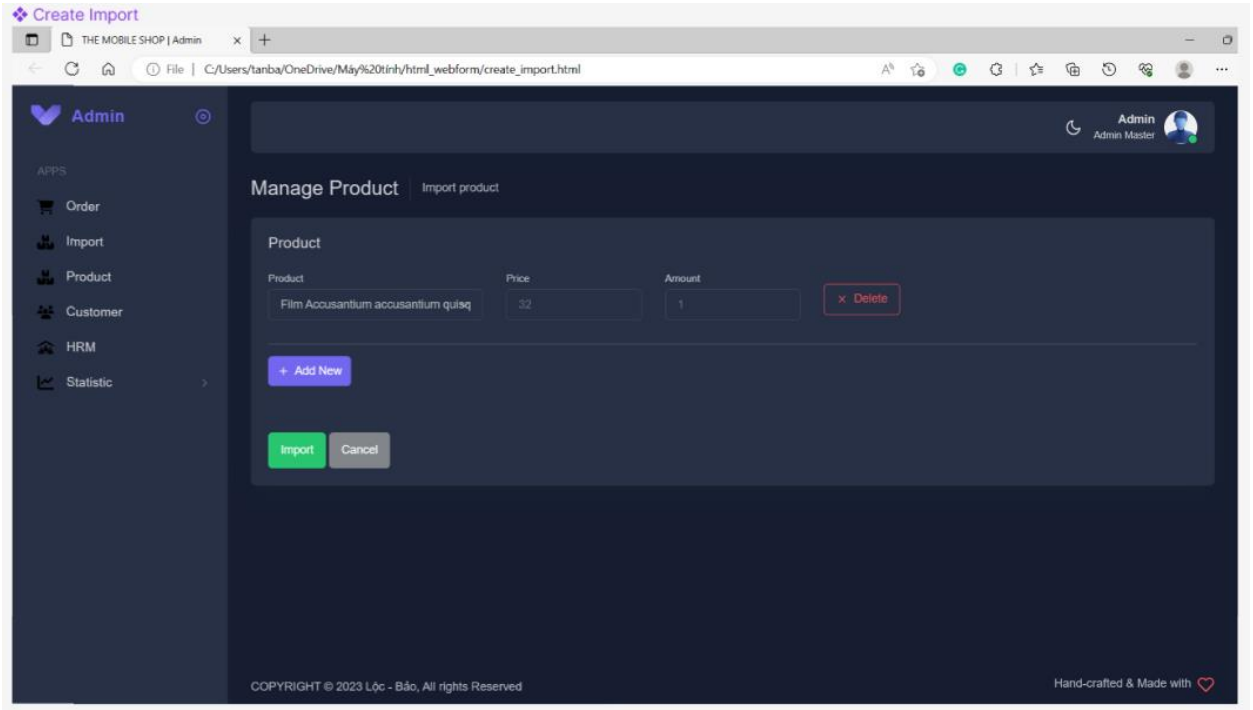
Onclick Print at Action in Order Web, Product was chosen will be print to PDF file


PDF Order Web
Print Order
File | C:/Users/tanba/OneDrive/Máy%20tính/html_webform/Print%20Order.html
The Coffee
Ordered At: 4/8/2023 12:52:57 PM
19 Nguyen Huu Tho street
Tan Phong ward, District 7, Ho Chi Minh City, Vietnam
(028) 37 755 035, (028) 37 755 055
Invoice To:
Brooklyn Runolfsson
420 Thompson Path Abbottview, MA 92436-5586
+1-630-868-0006
stephania26@gmail.com
Payment Details:
Total: 89.255.000
Is paid: True
Status: 0
PRODUCT'S NAME
AMOUNT
PRICE
DISCOUNT
TOTAL
Film Accusantium accusantium quisquam ut.
1
5.228.000
5.228.000
Film Est neque et non.
5
6.260.000
31.300.000
Film Cum sit et asperiores aut maiores.
5
8.650.000
43.250.000
Film Suscipit sapiente quia.
3
3.127.000
9.381.000
Salesperson: The Administrator 123
Subtotal: 89.159.000
Discount:
Ship: 96.000

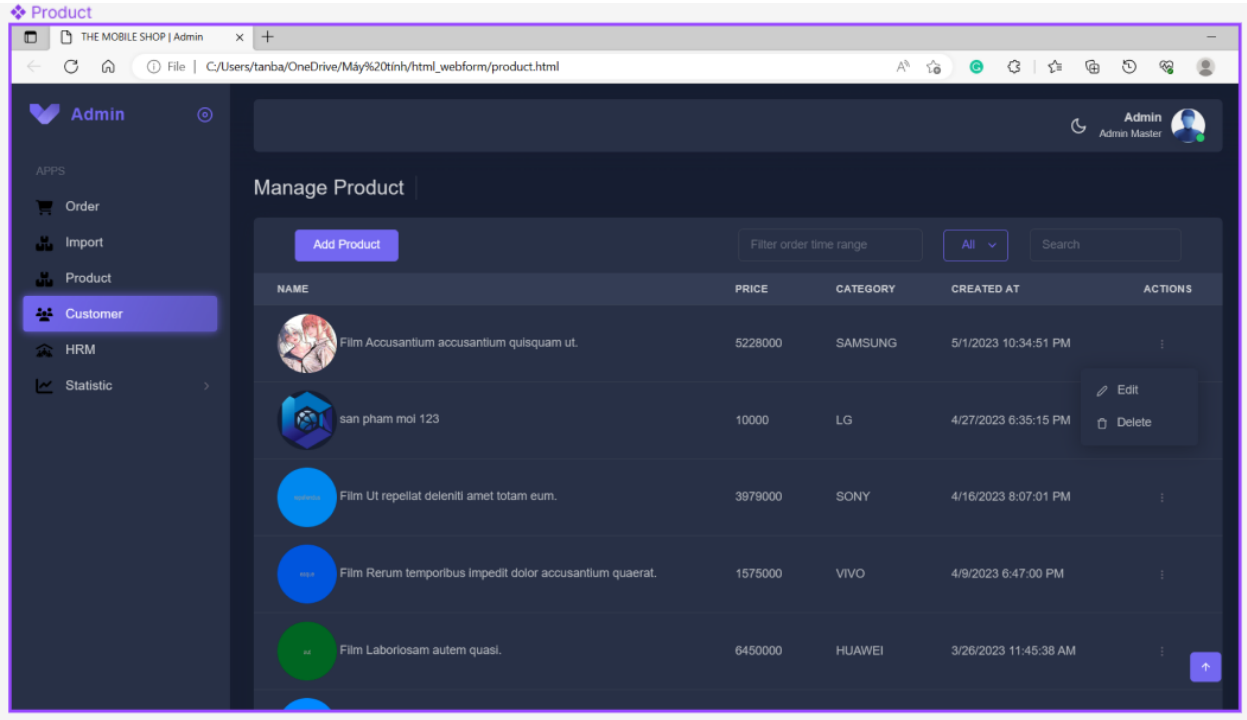
Onclick to Import, Import Web will be displayed



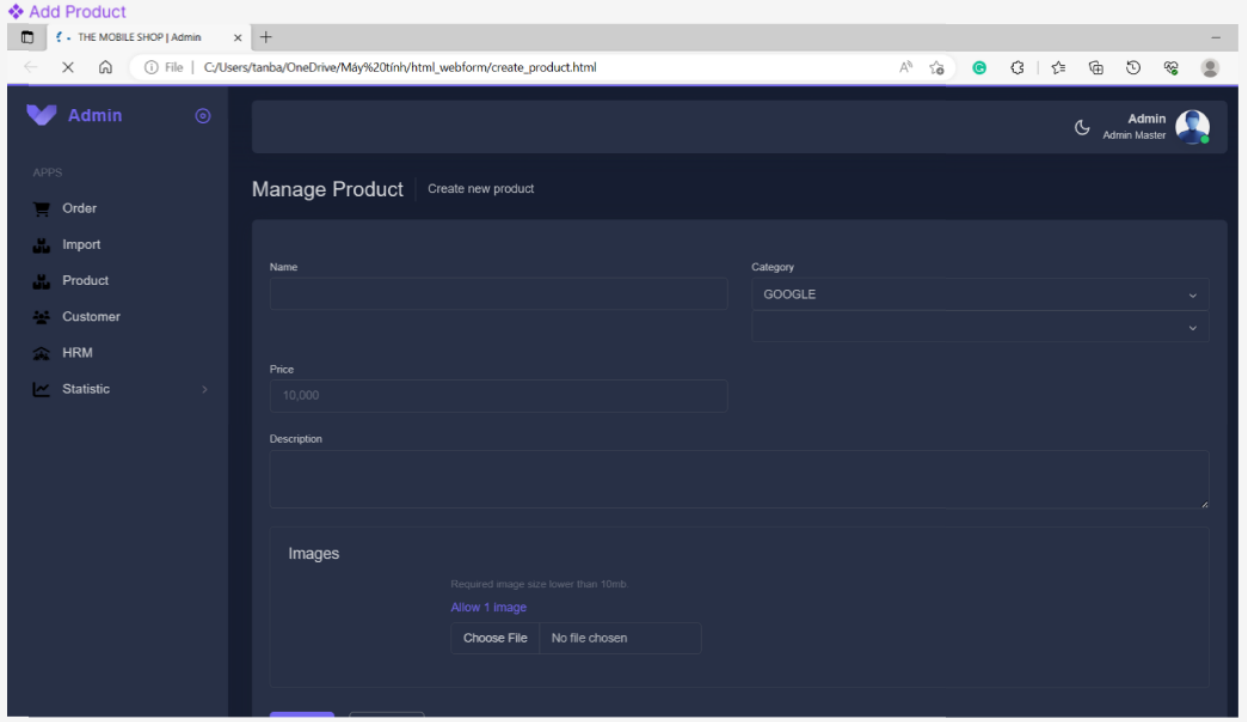
Onclick to Import Products button, Form Create Import will be displayed



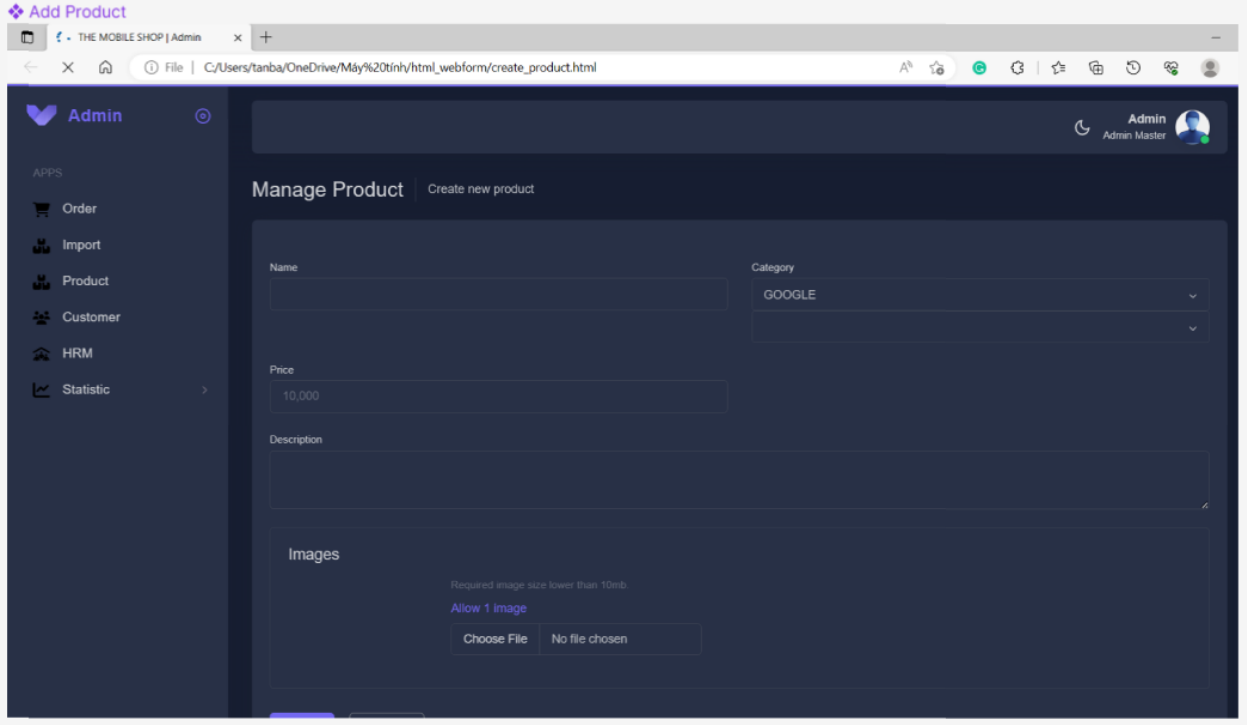
Onclick to Product, Product Web will be display



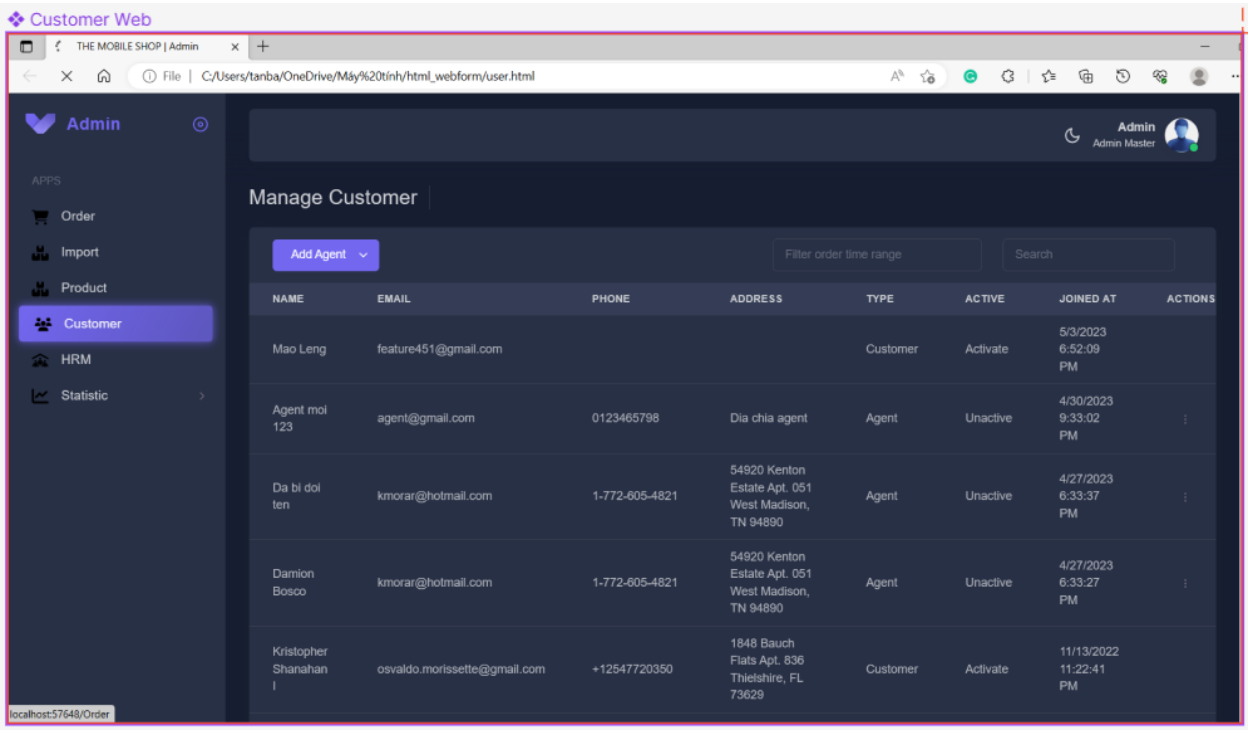
Onclick to button Add Product, form Create New Product will be displayed



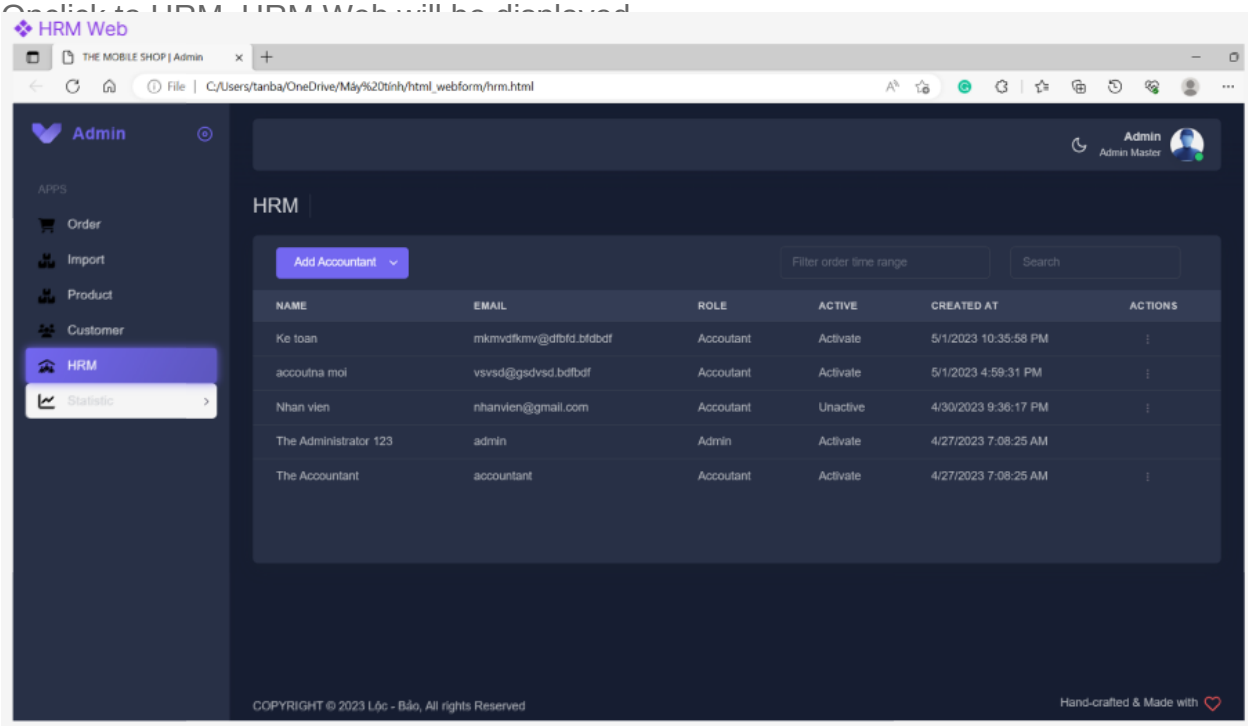
In Product Web, onclick to Edit in Action, form Edit product will be displayed



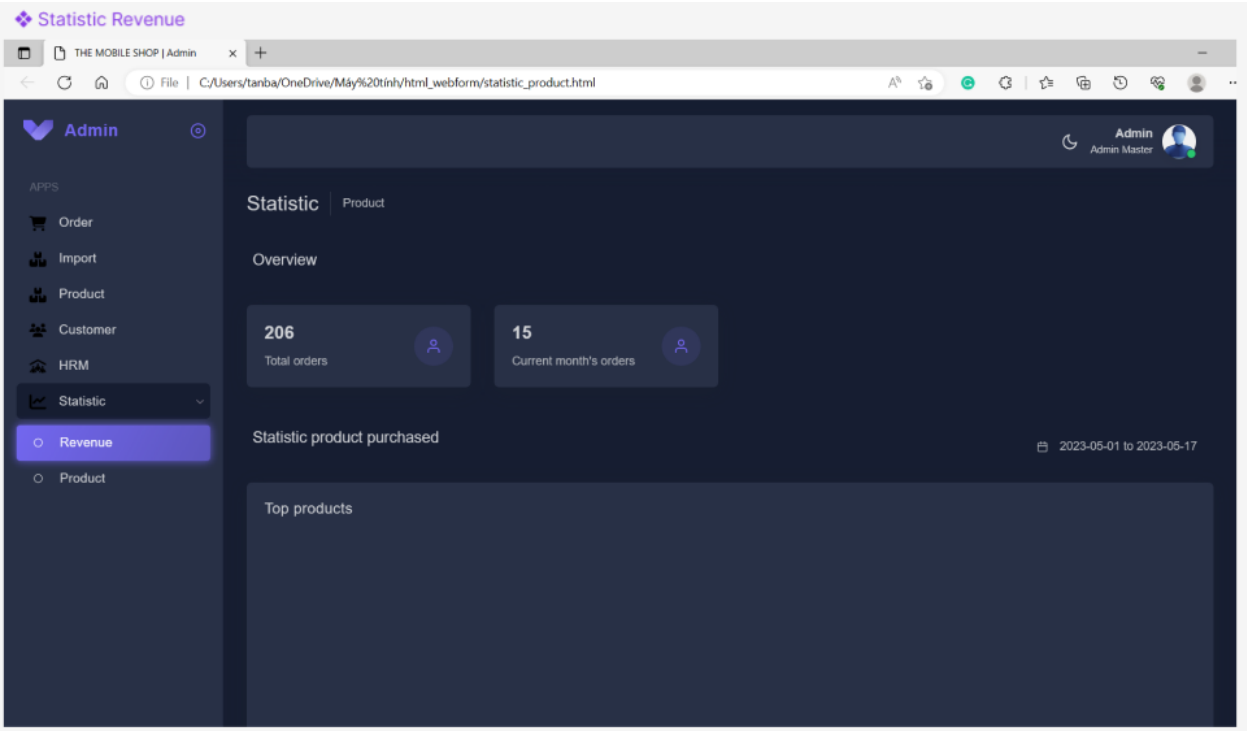
Onclick to Customer, Customer web will be displayed



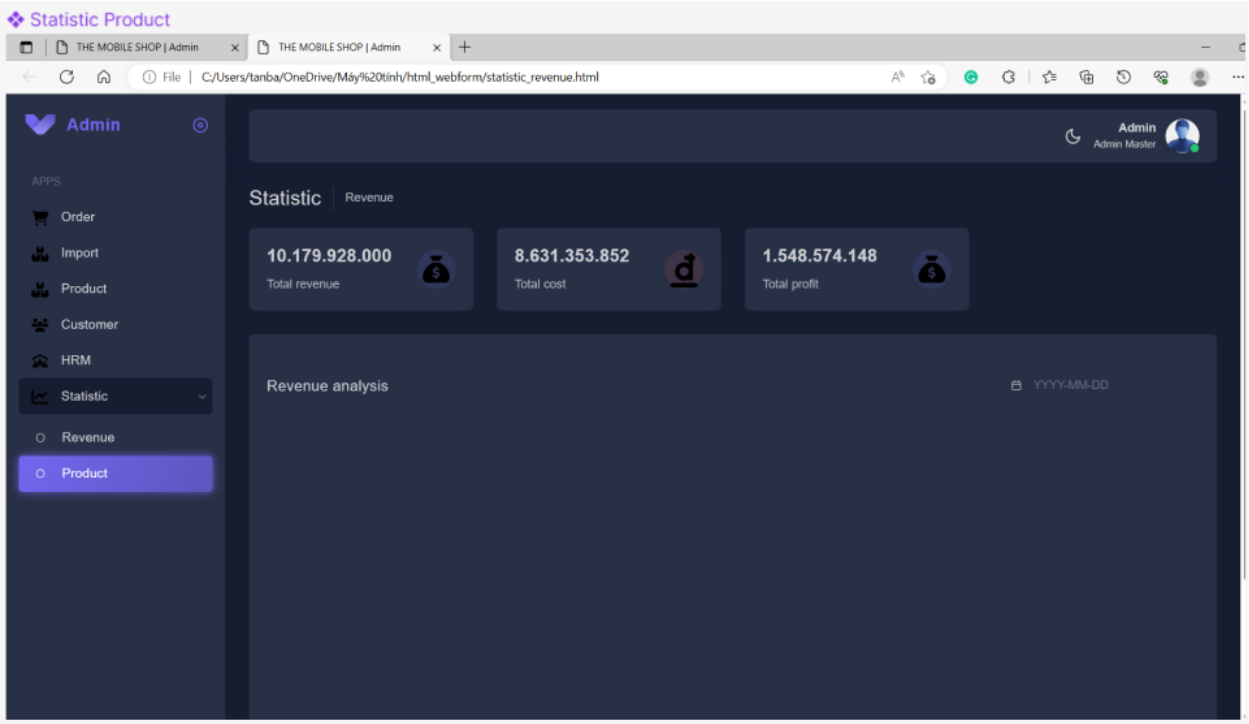
Onclick to HRM, HRM Web will be displayed



Choose Revenue in Statistic, Statistic Revenue will be displayed



Choose Product in Statistic, Statistic Product will be displayed



5.2. Static model – class diagrams

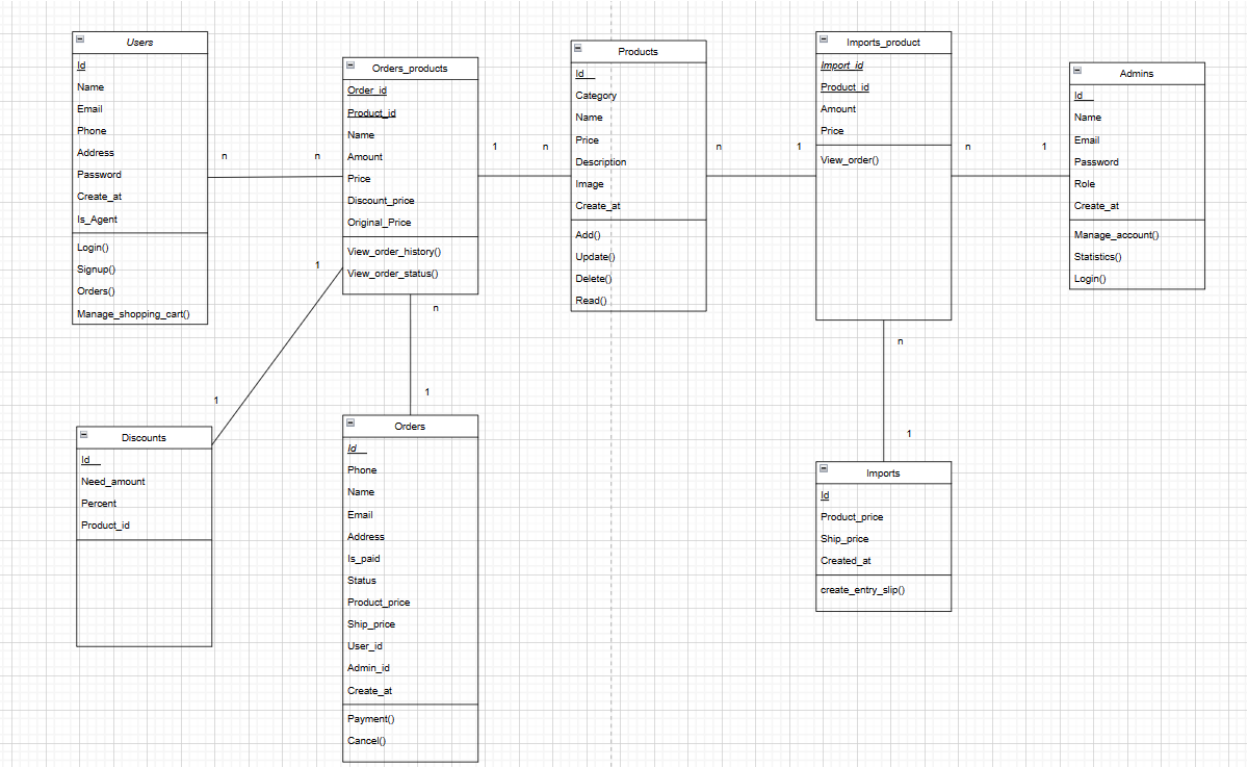


Figure 38: Class Diagram

5.3. Dynamic model – sequence diagrams

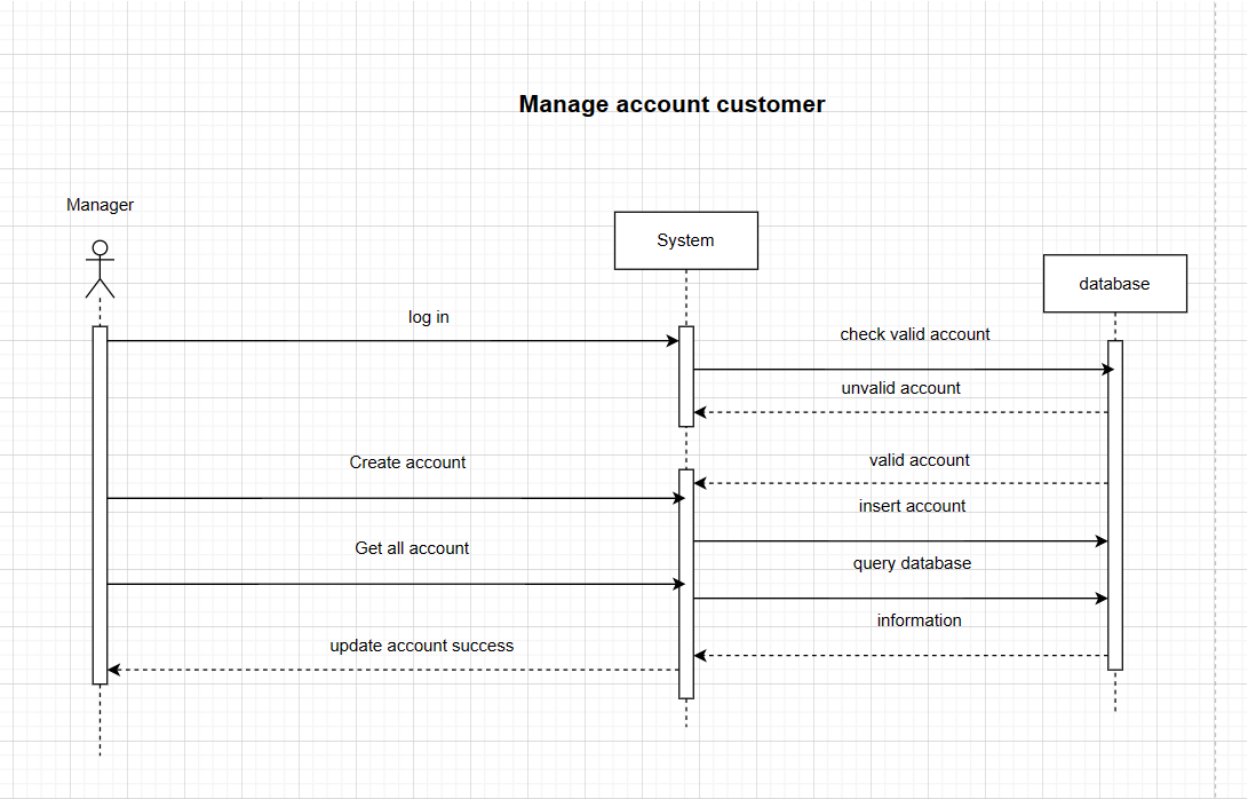


Figure 39: Sequence Diagram Manage account customer

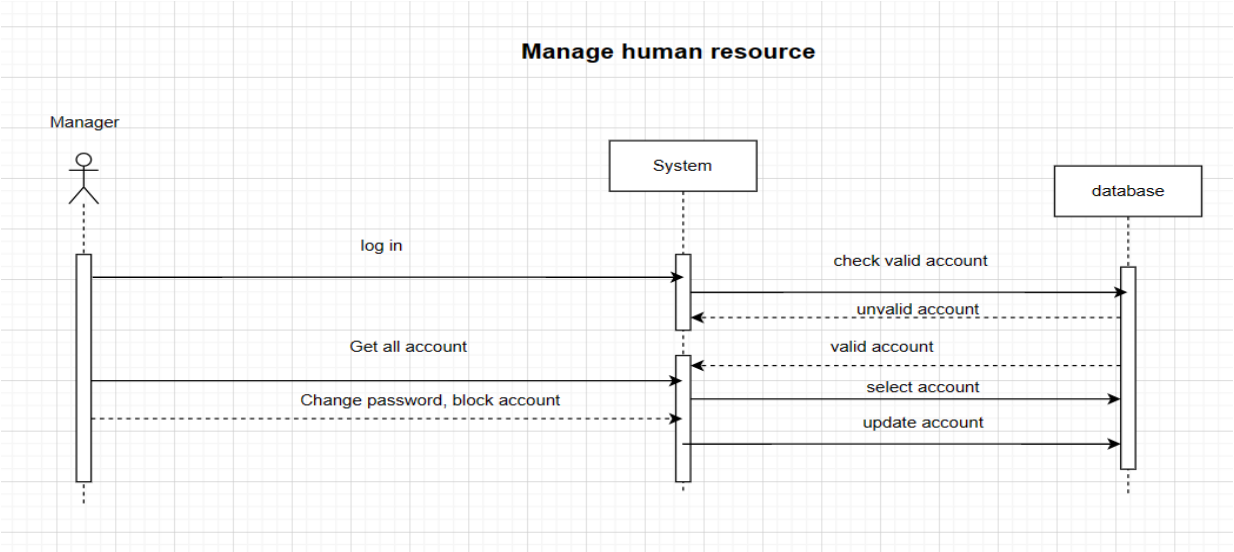


Figure 40: Sequence Diagram Manage Human Resource

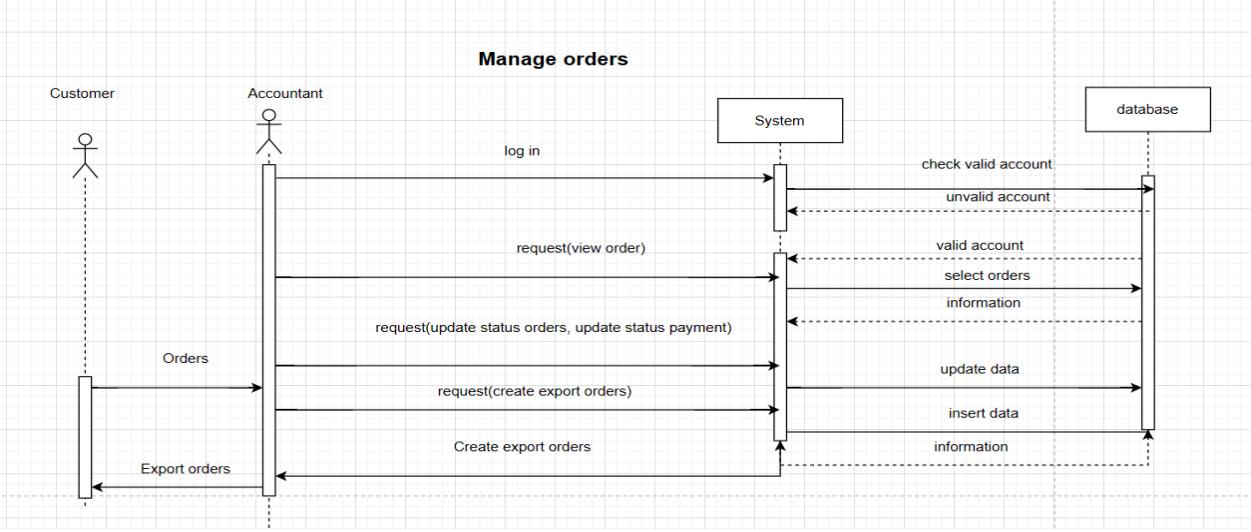


Figure 41: Sequence Diagram manage orders

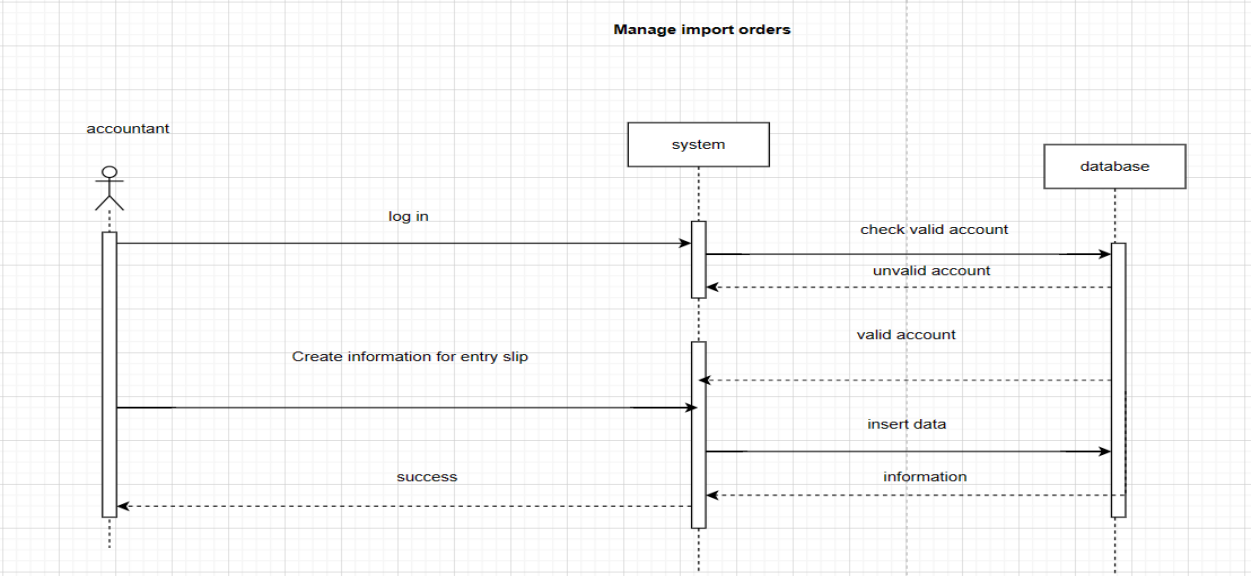


Figure 42: Sequence Diagram Manage Import Orders

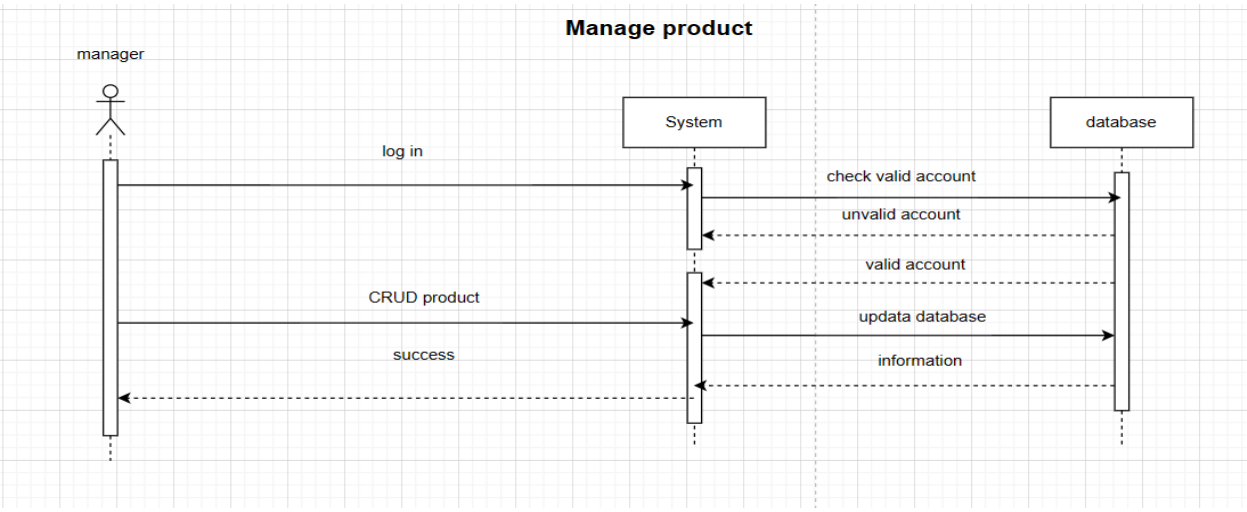


Figure 43: Sequence Diagram manage product

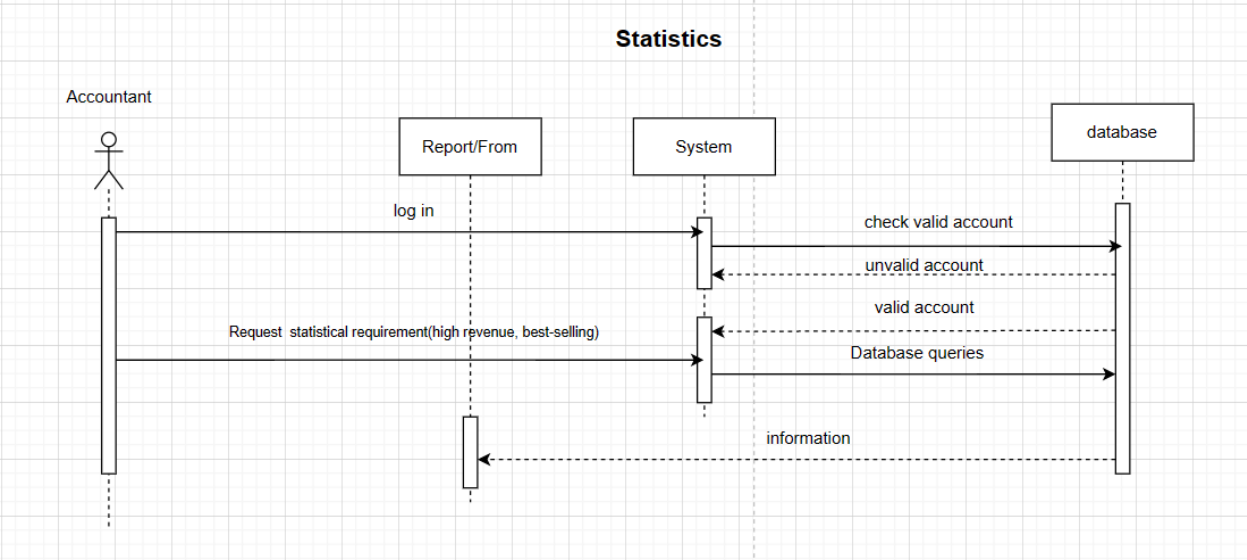


Figure 44: Sequence Diagram Statistic

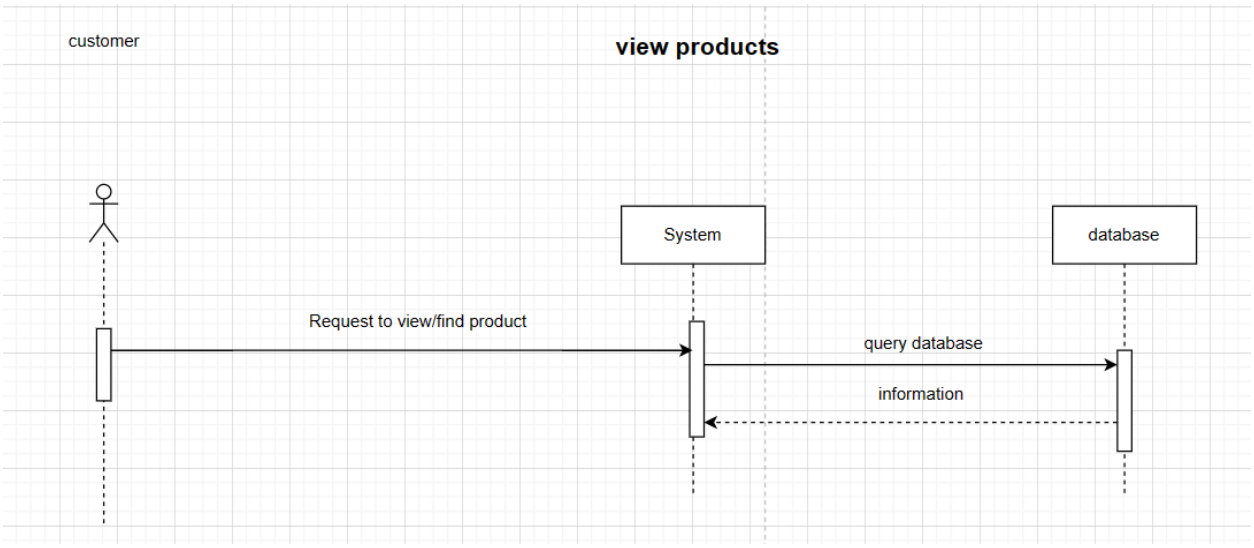


Figure 45: Sequence Diagram View Products

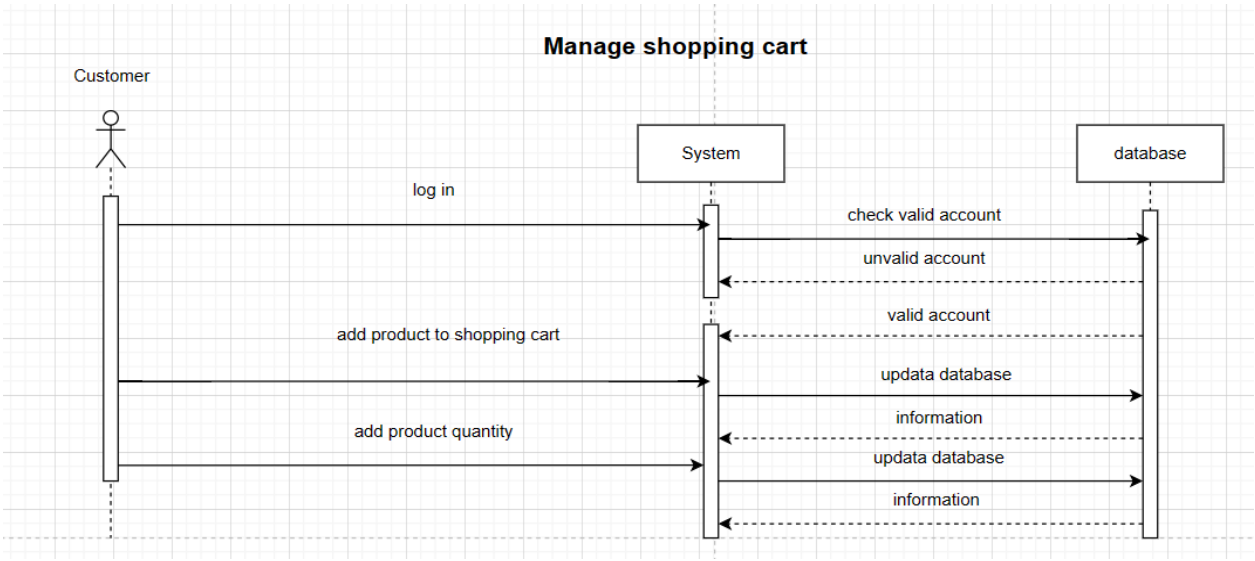


Figure 46: Sequence Diagram Manage Shopping cart

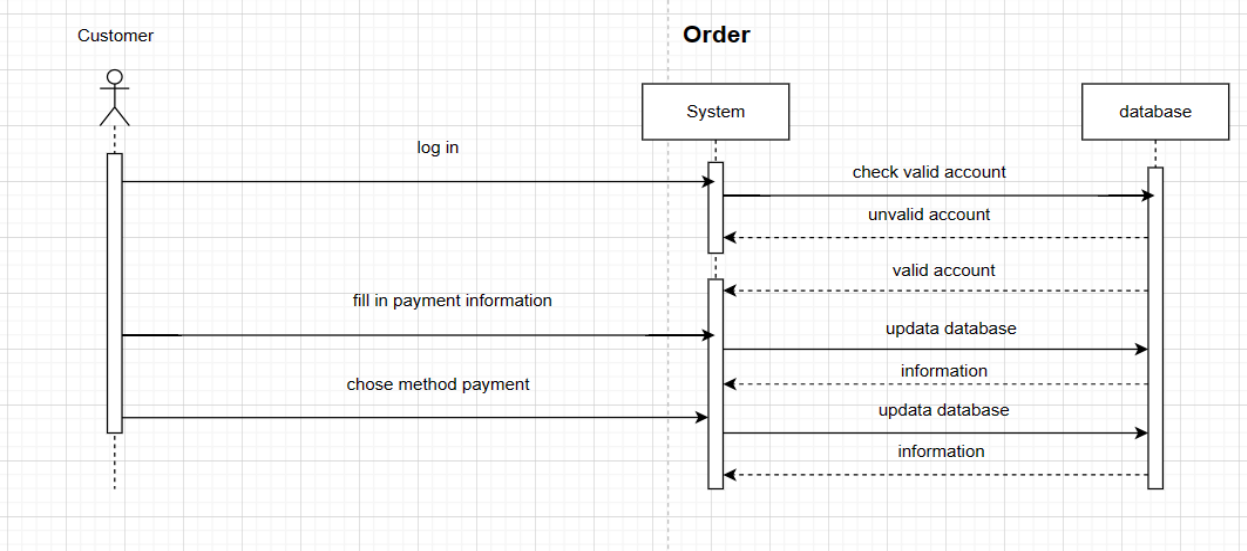


Figure 47: Sequence Diagram Order

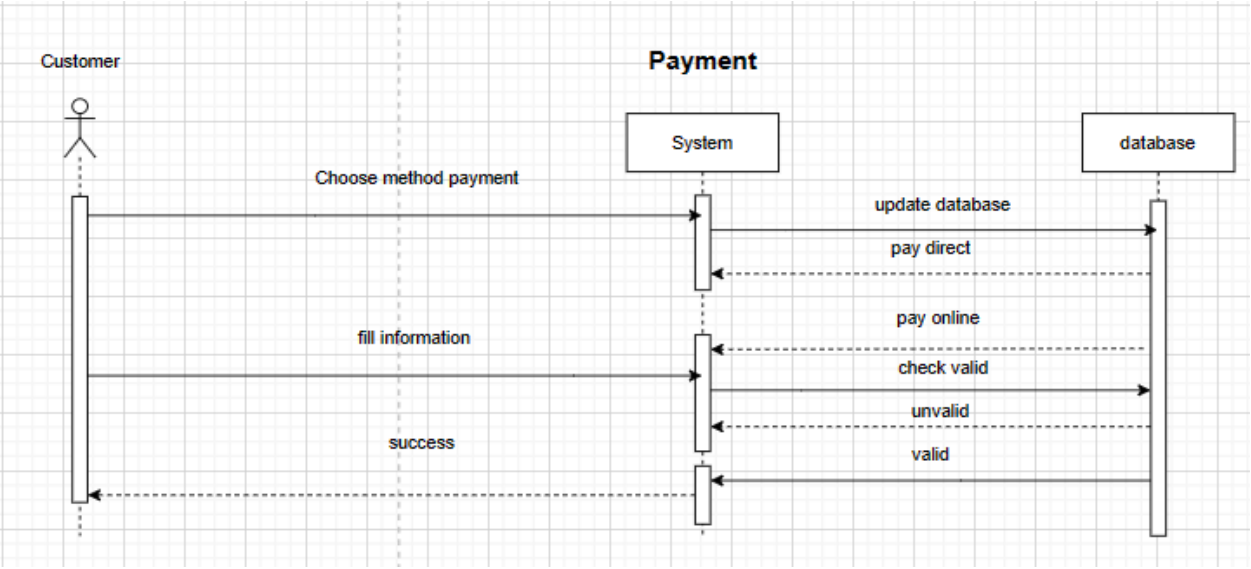


Figure 48: Sequence Diagram Payment

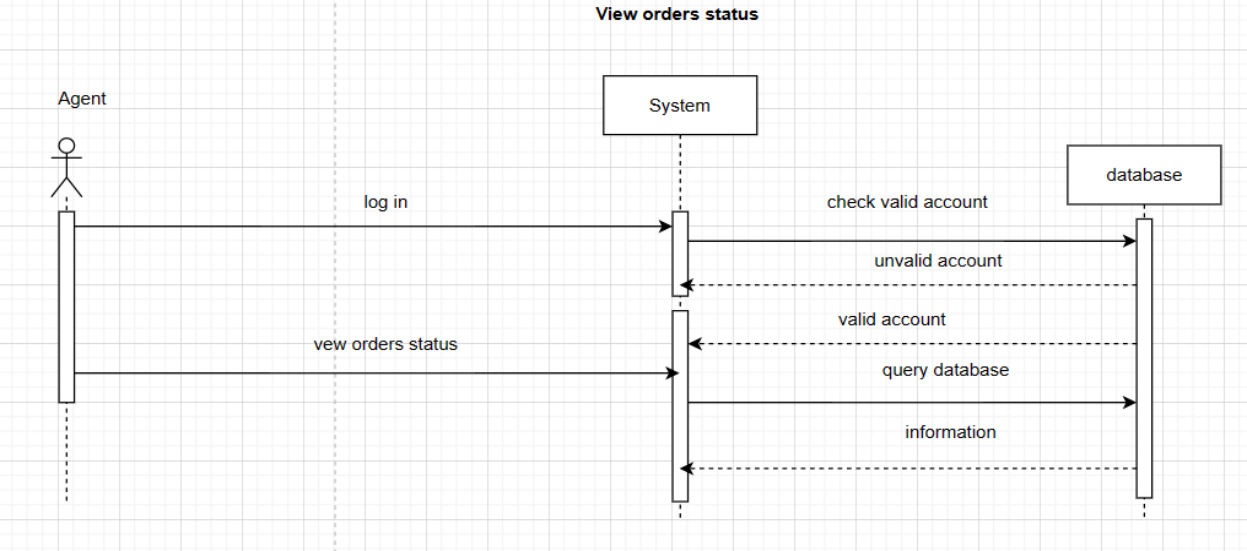


Figure 49: Sequence Diagram View orders status

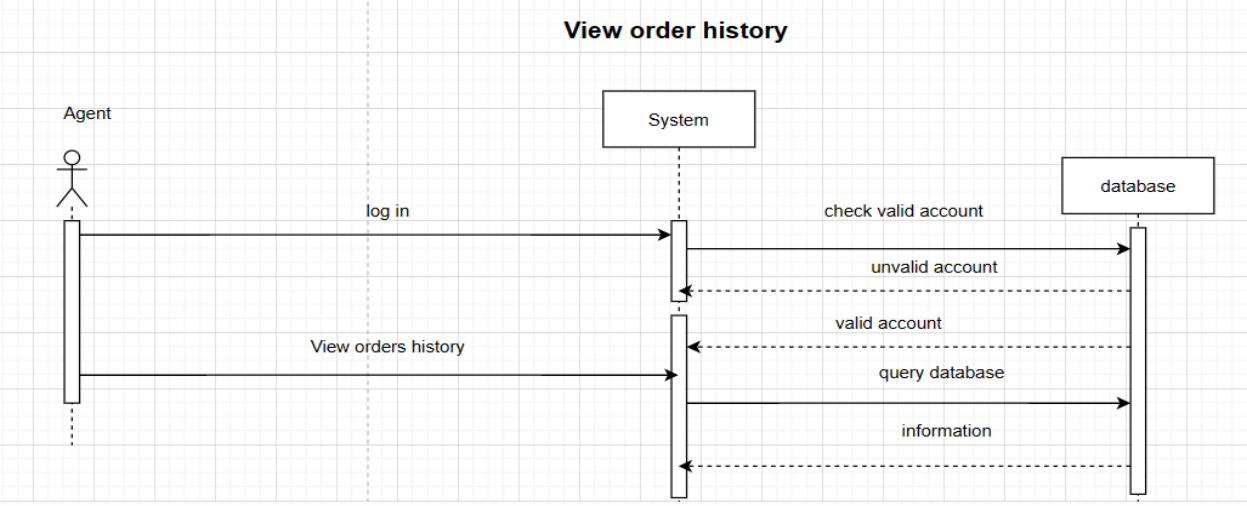


Figure 50: Sequence Diagram view order history

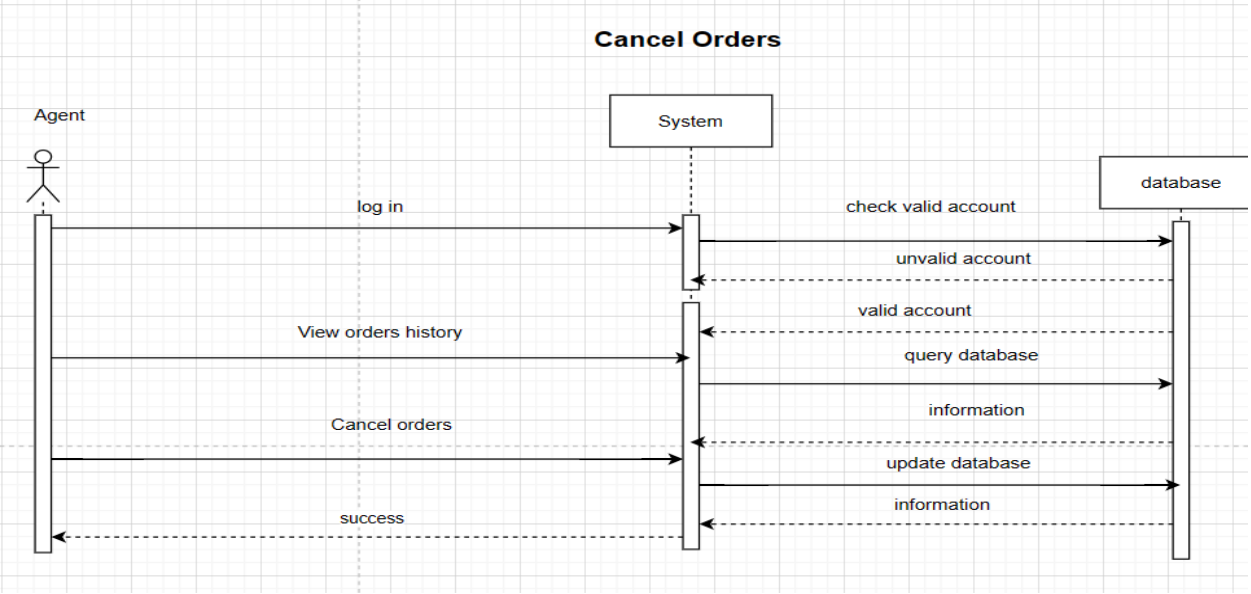


Figure 51: Sequence Diagram Cancel Orders

5.4. Rationale for your detailed design model

Confirming that the program complies with the demands: The software's implementation and compliance with the requirements are described in detail in the detailed design model. This guarantees that the software will be developed in line with the requirements of the client.

Facilitating communication: The detailed design model gives the members of the development team a shared understanding of the software's architecture and design. The team members' ability to communicate and work together is facilitated by this.

Error reduction: The detailed design model offers a thorough explanation of the architecture and design of the software. Errors are less likely to occur during the implementation stage as a result.

Facilitating testing: Software testing can be done on the basis of the detailed design model. The testing team can develop test cases and run software tests using the detailed design model.

5.5. Traceability from requirements to detailed design model

CHAPTER 6 - TEST PLAN

6.1. Requirements/specifications-based system-level test cases

Requirements/specifications-based system level test cases

★ Requirement: The system must allow distributors to place orders, choose payment methods, and view the status of their orders.

Test Case:

- Create a new distributor account in the system.
- Log into the system as the distributor.
- Place an order for a product and choose a payment method.
- Verify that the order is successfully submitted and the status of the order is updated in the system.
- View the order status and payment status in the system.

★ Requirement: The system must allow accountants to create warehouse entry or exit orders, update the status of orders, and the payment status of the distributor.

Test Case:

- Log into the system as an accountant.
- Create a new warehouse entry order and update the status of the order.
- Verify that the order is successfully created and the status of the order is updated in the system.
- Update the payment status of the distributor for the order.
- Verify that the payment status is updated in the system.

★ Requirement: The system must allow accountants to view reports on goods imported/customers, the best-selling products, and monthly revenue reports.

Test Case:

- Log into the system as an accountant.
- View the report on goods imported/customers.
- Verify that the report displays accurate and up-to-date information.
- View the report on the best-selling products.
- Verify that the report displays accurate and up-to-date information.
- View the monthly revenue report.
- Verify that the report displays accurate and up-to-date information.

★ Requirement: The system must allow warehouse managers to manage personnel and user accounts, as well as product management.

Test Case:

- Log into the system as a warehouse manager.

- Add a new employee account to the system.
- Verify that the employee account is successfully created and can log into the system.
- Add a new product to the system.
- Verify that the product is successfully added and can be managed in the system.

★ Requirement: The B2C website or mobile application must be able to connect to the same MSSQL server and use VNPay for distributor payments.

Test Case:

- Launch the B2C website or mobile application.
- Place an order for a product and choose VNPay as the payment method.
- Verify that the payment is successfully processed and the order status is updated in the system.
- Log into the system as an accountant.
- Verify that the payment status for the order is updated in the system.

6.2. Traceability of test cases to use cases

❖ Use Case: Manage Order
- ➔ Test Case: Update the status of the order.
- ➔ Test Case: Update the payment status of the order.

❖ Use Case: Statistic
- ➔ Test Case: View the report on goods imported/customers.
- ➔ Test Case: View the report on the best-selling products.
- ➔ Test Case: View the monthly revenue report.

❖ Use Case: View product information
- ➔ Test Case: View the product details page.

❖ Use Case: Manage Shopping Cart
- ➔ Test Case: Add a product to the shopping cart.
- ➔ Test Case: Remove a product from the shopping cart.
- ➔ Test Case: Update the quantity of a product in the shopping cart.

❖ Use Case: Manage Import
- ➔ Test Case: Generate a new import in the system.

❖ Use Case: View History Orders
- ➔ Test Case: View the order history page.

- ❖ Use Case: Order
  - ➔ Test Case: Place an order for a product and choose a payment method.
  - ➔ Test Case: View the order status and payment status in the system.

- ❖ Use Case: Manage Product
  - ➔ Test Case: Add a new product to the system.
  - ➔ Test Case: Update the details of an existing product.

- ❖ Use Case: Manage Employee
  - ➔ Test Case: Add a new employee account to the system.
  - ➔ Test Case: Update the details of an existing employee account.

- ❖ Use Case: Manage Customer
  - ➔ Test Case: Create a new agent account in the system.

6.3. Techniques used for test generation

Test Case: Update the details of an existing agent account.

- White Box Testing: This technique involves testing the system's internal structure, including its code and architecture. Test cases are designed based on knowledge of the system's internals, and are focused on verifying that the system operates correctly at a code level.

- Black Box Testing: This technique involves testing the system's functionality without considering the internal structure of the system. Test cases are designed based on the system's inputs and expected outputs, and are focused on verifying that the system behaves as expected from a user's perspective.

- White box :

Test case:	Generate a new import in the system
Test Object:	To verify that a new import can be successfully created and saved in the system.
Precondition:	The user has appropriate permissions to generate a new import. The database connection has been established. The system is functional.

Test Steps:	Open the application and log in to the system using valid credentials. Navigate to the "Import" menu on screen. Enter valid values for the import product details, including the product ID, amount, and price. Click the "Save" button to generate the new import. Verify that the new import record has been successfully created in the "imports" table with the correct data, including the ID, product price, and created_at timestamp. Verify that import product records have been successfully created in the "import_product" table with the correct data, including the import ID, product ID, amount, and price. Verify that the product stock has been successfully updated in the "products" table based on the amount of the product in the import. Verify that the total cost of the import has been correctly calculated by summing the cost of each product in the import. Verify that the application displays a success message indicating that the import has been generated successfully.
Test Data:	Product ID: 1 Amount: 100 Product Price: 100000
Expected Results:	The new import record should be created successfully in the "imports" table. Import product records should be created successfully in the "import_product" table. The product stock should be updated successfully in the "products" table. The total cost of the import should be calculated correctly. The application should display a success message indicating that the import has been generated successfully.
Actual Results:	The new import record is created successfully in the "imports" table. Import product records are created successfully in the "import_product" table. The product stock is updated successfully in the "products" table. The total cost of the import is calculated correctly. The application displays a success message indicating that the import has been generated successfully.
Test Result:	Pass

- Blackbox:

Test case:	Generate a new import in the system
Test Object:	To verify that a new import can be successfully created and saved in the system.
Precondition:	The user has appropriate permissions to generate a new import. The database connection has been established. The system is functional.
Test Steps:	Launch the system and navigate to the import page. Select a product from the available list and enter the necessary product details Enter the amount of the selected product to be imported. Click on the "Save" button. Verify that the system displays a success message indicating that the import was saved successfully. Verify that the new import is added to the list of imports in the system.
Test Result:	If the system successfully saves the new import and displays a success message, the test case passes. If the system fails to save the new import or displays an error message, the test case fails. The main difference between white box and black box testing, in the context of the use case "Generate a new import in the system", is the level of knowledge and access to the system's internals that the tester has. In white box testing, the tester has access to the system's source code and is familiar with the internal workings of the system. The tester can analyze the code and identify potential issues or areas of the system that need to be tested. This testing method is useful for verifying the correct implementation of the system and ensuring that all code paths are tested. In black box testing, the tester does not have access to the system's source code or internal workings. The tester focuses on testing the system's external behavior and functionality, without considering how the system works internally. This testing method is useful for ensuring that the system meets the requirements and behaves correctly from the end-user's perspective.

6.4. Assessment of the goodness of your test suite

(Which metrics were used for such assessment?)
Acknowledgment
References
(Must be complete, and correctly formatted using the standard for IEEE Conference Proceedings)

7. SourceCode
This code to get all employee, apply filter and search, then return a view

```
public ActionResult Index(string created_at, string q, int page = 1)
{
    if (Authed() == null) return RedirectToAction("Index", "Home");
    if (Authed().is_admin_master == false) return RedirectToAction("Index", "Home");

    int pageSize = 20;

    var builder = db.Admins.AsQueryable();
    if (!string.IsNullOrEmpty(created_at))
    {
        var split = created_at.Split(',');
        if (split.Length == 2)
        {
            var start = DateTime.Parse(split[0]);
            var end = DateTime.Parse(split[1]);
            builder = builder.Where(o => o.created_at >= start && o.created_at <= end);
        }
    }
    if (!string.IsNullOrEmpty(q))
    {
        builder = builder.Where(o =>
            o.name.Contains(q) ||
            o.email.ToString().Contains(q) ||
            o.created_at.ToString().Contains(q)
        );
    }
    var admins = builder.OrderByDescending(o => o.created_at)
                        .Skip((page - 1) * pageSize)
                        .Take(pageSize)
                        .ToList();

    ViewBag.Admins = admins;
    ViewBag.Paginate = new Paginate(builder.Count(), pageSize);

    return View("~/Views/Admin/Index.cshtml");
}
```

This code create a new employee

```
[HttpPost]
[Route("/hrm")]
0 references
public ActionResult Store(FormCollection data)
{
    if (Authed() == null) return RedirectToAction("Index", "Home");
    if (Authed().is_admin_master == false) return RedirectToAction("Index", "Home");

    if (string.IsNullOrEmpty(data["name"]) || string.IsNullOrEmpty(data["email"]))
    {
        TempData["error"] = "Field must not be empty";

        return RedirectBack();
    }

    db.Admins.Add(new Admin()
    {
        name = data["name"],
        email = data["email"],
        password = BCrypt.Net.BCrypt.HashPassword(data["email"]),
        is_admin_master = false,
        active = true,
        created_at = DateTime.Now,
    });
    db.SaveChanges();
    TempData["success"] = "Create accountant successfully";

    return RedirectBack();
}
```

This code list all imports and apply filter and search

```
[HttpGet]
[Route("/import")]
0 references
public ActionResult Index(string status, string created_at, string q, int page = 1)
{
    if (Authed() == null) return RedirectToAction("Index", "Home");

    int pageSize = 20;

    var builder = db.Imports.AsQueryable();
    if (!string.IsNullOrEmpty(created_at))
    {
        var split = created_at.Split(',');
        if (split.Length == 2)
        {
            var start = DateTime.Parse(split[0]);
            var end = DateTime.Parse(split[1]);
            builder = builder.Where(o => o.created_at >= start && o.created_at <= end);
        }
    }

    var imports = builder.OrderByDescending(o => o.created_at)
                         .Skip((page - 1) * pageSize)
                         .Take(pageSize)
                         .ToList();

    ViewBag.Imports = imports;
    ViewBag.Paginate = new Paginate(builder.Count(), pageSize);

    return View("~/Views/Import/Index.cshtml");
}
```

This code create import

```
0 references
public ActionResult Store(FormCollection form)
{
    if (Authed() == null) return RedirectToAction("Index", "Home");

    Dictionary<string, string> data = form.AllKeys.ToDictionary(k => k, k => form[k]);

    if (data.Count % 3 != 0)
    {
        TempData["error"] = "Product not found or not filled";

        return RedirectToAction("Import", "Product");
    }

    List<ExpandoObject> products = new List<ExpandoObject>();
    for (int i = 0; i < data.Count / 3; i++)
    {
        dynamic product = new ExpandoObject();
        product.product_id = data[$"products[{i}][product_id]"];
        product.amount = data[$"products[{i}][amount]"];
        product.price = data[$"products[{i}][price]"];
        products.Add(product);
    }

    var import = new Import
    {
        product_price = 0,
        created_at = DateTime.Now
    };
    db.Imports.Add(import);
    db.SaveChanges();

    var sync = new Dictionary<int, ImportProduct>();
    double total = 0;
    foreach (dynamic product in products)
    {
        int productId = int.Parse(product.product_id);
        double price = double.Parse(product.price);
        int amount = int.Parse(product.amount);
        total += price * amount;

        sync[productId] = new ImportProduct
        {
            price = price,
            amount = amount,
            import_id = import.id,
            product_id = productId
        };
    }

    foreach (var item in sync)
    {
        db.ImportProducts.Add(item.Value);
    }
    db.SaveChanges();

    import.product_price = total;
    db.SaveChanges();

    TempData["success"] = "Import products successfully";

    return RedirectToAction("Index", "Import");

}
```

This code list all orders and apply filter and search

```
[HttpGet]
[Route("/order")]
0 references
public ActionResult Index(string status, string ordered_at, string q, int page = 1)
{
    if (Authed() == null) return RedirectToAction("Index", "Home");

    int pageSize = 20;

    var builder = db.Orders.AsQueryable();
    if (!string.IsNullOrEmpty(status))
    {
        builder = builder.Where(o => o.status.ToString().Equals(status));
    }
    if (!string.IsNullOrEmpty(ordered_at))
    {
        var split = ordered_at.Split(',');
        if (split.Length == 2)
        {
            var start = DateTime.Parse(split[0]);
            var end = DateTime.Parse(split[1]);
            builder = builder.Where(o => o.created_at >= start && o.created_at <= end);
        }
    }
    if (!string.IsNullOrEmpty(q))
    {
        builder = builder.Where(o =>
            o.name.Contains(q) ||
            o.address.Contains(q) ||
            o.email.Contains(q) ||
            o.phone.Contains(q) ||
            o.ship_price.ToString().Contains(q) ||
            o.product_price.ToString().Contains(q) ||
            o.created_at.ToString().Contains(q)
        );
    }

    var orders = builder.Include(o => o.User)
                        .OrderBy(o => o.status)
                        .ThenByDescending(o => o.created_at)
                        .Skip((page - 1) * pageSize)
                        .Take(pageSize)
                        .ToList();

    ViewBag.Orders = orders;
    ViewBag.Paginate = new Paginate(builder.Count(), pageSize);

    return View("~/Views/Order/Index.cshtml");
}
```