|  |
| --- |
| **PROJECT REPORT**  Mini Supermarkets  **Semester: Programming Fundamentals**  **Class: PF08**  **Group: Group 4**  **Instructor: Nguyễn Xuân Sinh**  **Team members: NDE19021\_Nguyễn Quyết Thắng**  **NDE19064\_Nguyễn Vũ Huân**  **NDE19020\_Trần Văn Huân** |

Index

Project Name 1

Index 2

I. Project introduction 3

II. Analyze System Requirements 3

III. Design Details 16

IV. Test 22

V. Assign work to each team member 23

VI. Installation Instructions 24

I. Project introduction  
Process payment for orders in mini supermarkets

1. Proposed System

Actor Manager

Actor Casher

Manager function

1. Insert product
2. Update product

Cashier function

1. Create order
2. Update order

2. The scope of the project to be applied

Checkout and update invoice

Create and update product

3. System Name

Mini supermarkets management system

4. Deployment Environment

Mini supermarkets

5. Development Tools

Visual Studio Code

My SQl Work bench

# IntelliJ IDEA

6. Customer Requirements

(System features)

UC01\_Login

UC02\_Create Order

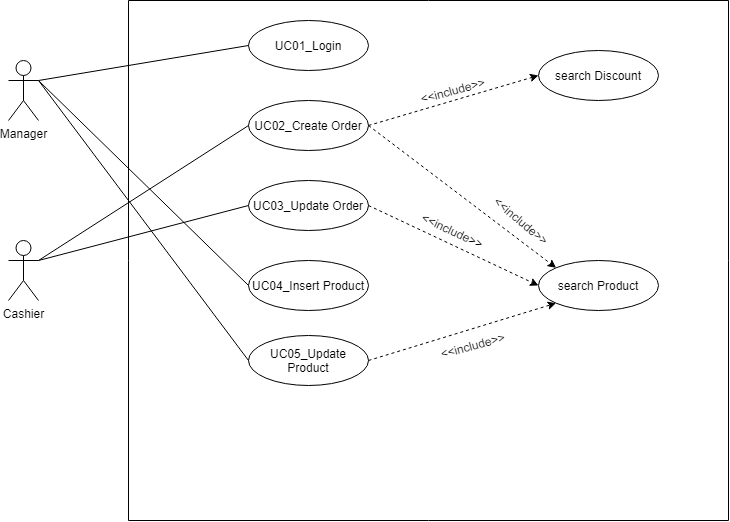
UC03\_Update Order

UC04\_Insert Product

UC05\_Update Product

II. Analyze System Requirements

1. Use case

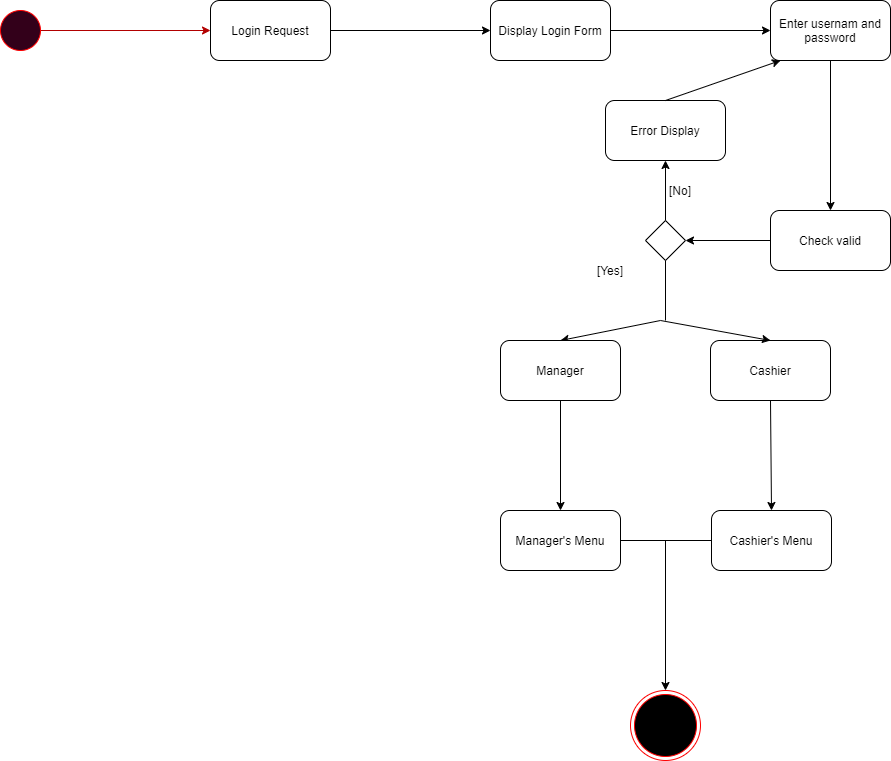


Use Case diagram

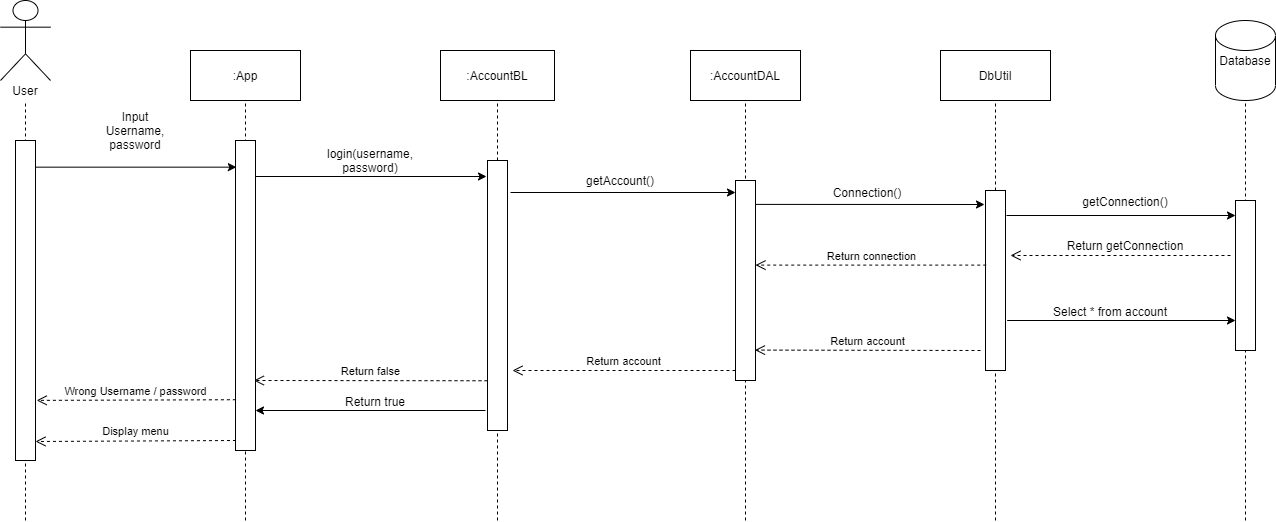
UC01\_Login (Description)

|  |  |
| --- | --- |
| Name | Login |
| ID | UC01 |
| Description | Manager and Cashier must login to system |
| Actor | Manager and Cashier |
| Organizational Benefits | Authentication |
| Frequency of Use | Often |
| Triggers | When a user wants to use the system |
| Preconditions |  |
| Postconditions | Display functions of the cashier /manager |
| Main Course | 1.User input user name and password  2.Valid user name/password  3. Display Functions of the user  3.1. If user is Manager (Manager's Menu) (AC3.1)  3.2. If user is Cashier (Cashier's Menu) (AC3.2) |
| Alternate Courses | AC3.1: Manager's Menu  1. Show:  -Insert product  -Update product  -Search product  AC3.2: Cashier's Menu  2. Show:  -Create order  -Update order |
| Exceptions | EX1. Valid Username:   * None special characters * Maximum 20 characters   EX2. Valid Password   * Minimum password length 8 characters * Minimum 1 uppercase character * Minimum 1 normal character * Minimum 1 numeric character   EX3. Username Password fails  - Return user to Main Course Step 1 |

UC01\_Login (ActivityDiagram)



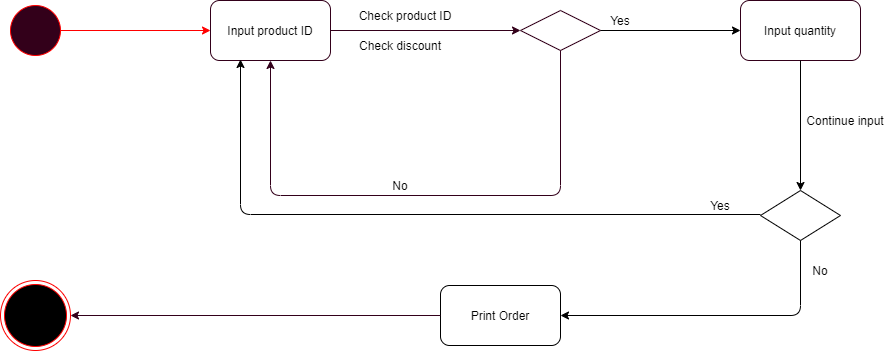
UC01\_Login (Sequence Diagram)



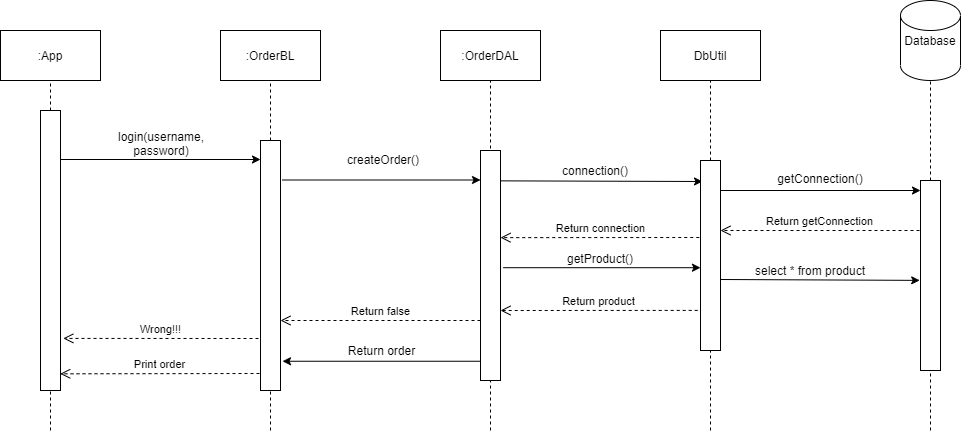
UC02\_Create Order (Description)

|  |  |
| --- | --- |
| Name | Create order |
| ID | UC02 |
| Description | Create new order |
| Actor | Cashier |
| Organizational  Benefit | Control action sale of the business |
| Frequency of Use | Often |
| Triggers | User selects the “Create Order” Function |
| Preconditions | User is Cashier was Login |
| Postconditions | Print Order |
| Main Courses | 1.Input product ID (AC1)  2.Search discount (AC2)  3.Input quantity  4.Print order |
| Alternate Courses | AC1. Check product ID  1.1: Product ID was existed in order  - Quantity + 1  1.2: Product ID doesn't in order  - Add product to order  AC2: Check discount type  2.1. Have discount  - Return product decrease-price  2.2. None have discount  - Return 0 |
| Exception | EX 1: Product doesn’t exist   1. System prompt user: Product doesn’t exist in system   EX3: Quantity < 0   1. Return user Main Course Step 3 |

UC02\_Create Order (Activity Diagram)



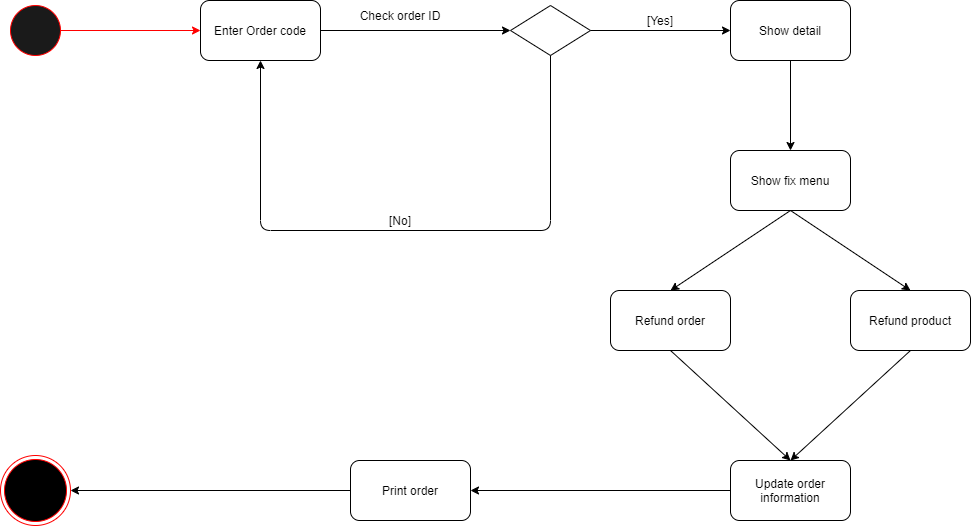
UC02\_Create Order (SequenceDiagram)



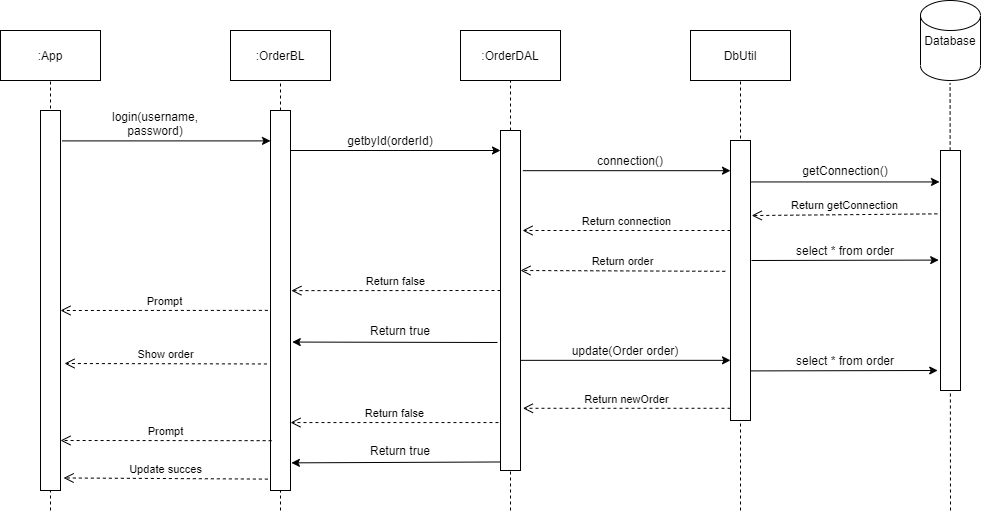
UC03\_Update Order (Description)

|  |  |
| --- | --- |
| Name | Update Order |
| ID | UC03 |
| Description | Edit about products on order product’s quantity , and refunded order |
| Actor | Cashier |
| Organizational  Benefit | Ensure the accuracy of the Order |
| Frequency of Use | Sometime |
| Triggers | User select the “Update Order” Function |
| Preconditions | Use is Cashier was login |
| Postconditions | Print new order |
| Main Courses | 1.Enter order ID  2.Show detail  3.Show fix menu (AC3)  4.Update order information  5.Print order updated |
| Alternate Courses | AC3: Fix content  3.1: Refund order  3.2: Refund product |
| Exception | EX1.1: Order ID doesn’t exist  1.Return Main Course Step 1  EX1.2: Valid product ID doesn’t exist  1. Prompt user ID product doesn’t exist in order  2. Return AC3.1  EX2: Only fix order within one hour   1. Prompt unable update order   EX5: System fails on saving order  1.System notifies user that an error has occurred  2.Return user to Main course step 1 |

UC03\_Update Order (Activity Diagram)



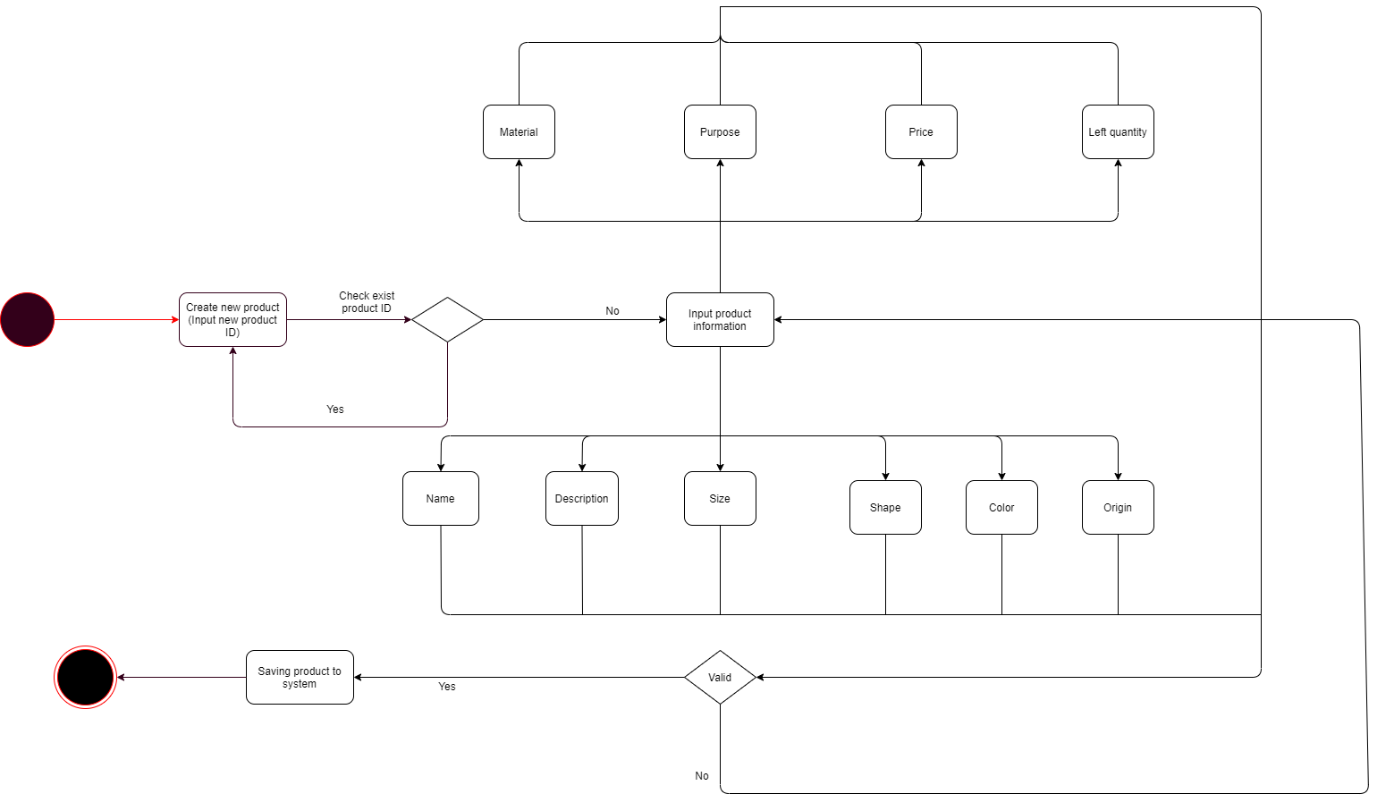
UC03\_Update Order (SequenceDiagram)



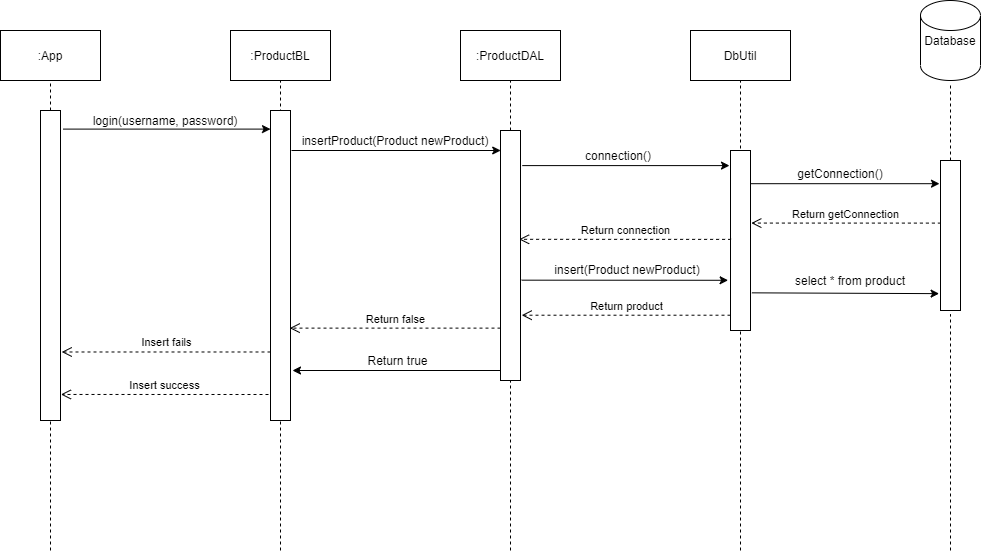
UC04\_Insert Product (Description)

|  |  |
| --- | --- |
| Name | Insert Product |
| ID | UC04 |
| Description | Add new product to system |
| Actor | Manager |
| Organizational  Benefit | Manage products in store |
| Frequency of Use | Often |
| Triggers | Selects ’Insert product’ |
| Preconditions | User is Manager |
| Postconditions | Insert success |
| Main Courses | 1.Create new product  2.Input product information  3.Saving product to system |
| Alternate Courses | AC2: Input product information:   * Name * Description * Size * Shape * Color * Origin * Material * Purpose * Price * Left Quantity |
| Exception | EX1: Product has Existed   1. Return user to Main courses step 1   EX2.1: Valid Name product   1. Name has existed  * Prompt user Name product existed in system   - Return Main course 2  EX2.2: Valid price  2.1. Price > 0  EX2.3: Valid quantity  3.1. Quantity > 0  EX2.4: Valid fails  Step 1: System prompt insert fails  Step 2: Return to Main Course step 1. |

UC04\_ Insert Product (Activity Diagram)



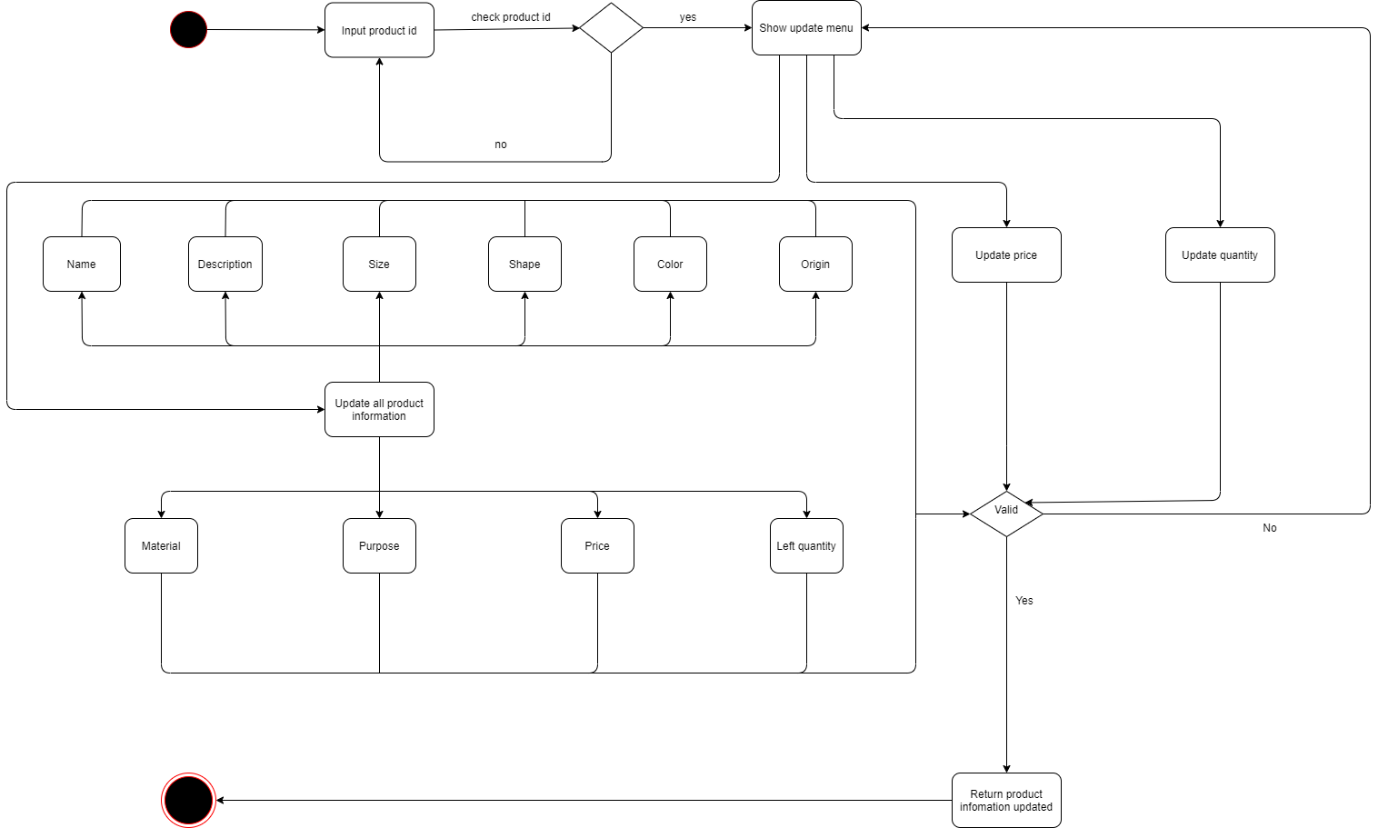
UC04\_Insert Product (SequenceDiagram)



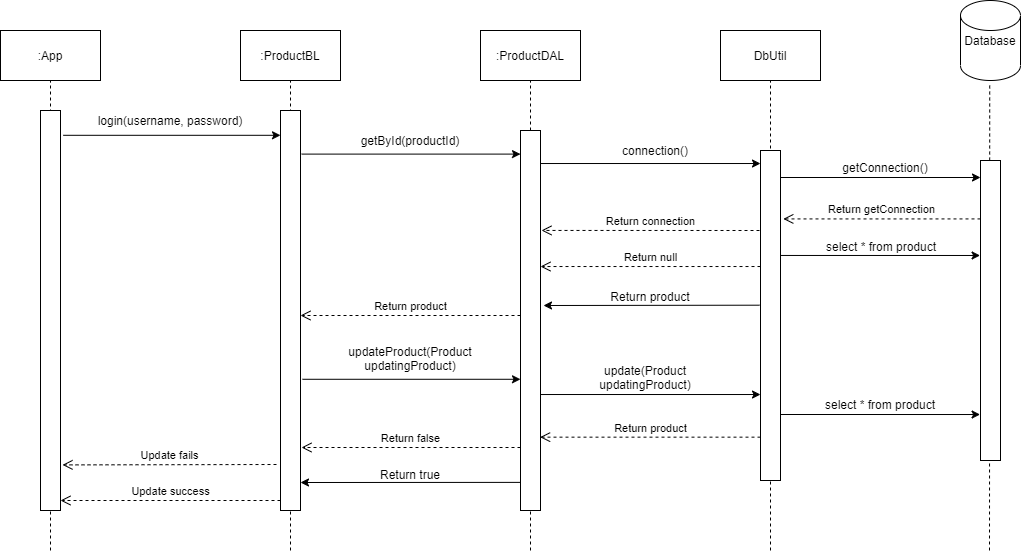
UC05\_Update Product (Description)

|  |  |
| --- | --- |
| Name | Update Product |
| ID | UC05 |
| Description | Update product information |
| Actor | Manager |
| Organizational Benefits | Update accurate product information |
| Frequency of Use | Sometime |
| Triggers | Select "Update Product" Function |
| Preconditions | User is Manager |
| Postcondition | System confirm update success |
| Main Course | 1. Input Product ID  2. Show update menu  3. Update product information  4. Prompt update success |
| Alternate Courses | AC2. Update Menu  1. Update price   1. Update product information:  * - Name * - Description * - Size * - Shape * - Color * - Origin * - Material * - Purpose * - Price * - Left quantity |
| Exceptions | EX1. Product ID doesn't exist  1. Return user to Main Course Step 1  EX2.1. Price < 0   1. Return Alternate Course Step 2   EX2.2. Quantity < 0   1. Return Alternate Course Step 3 |

UC05\_Update Product (Activity Diagram)

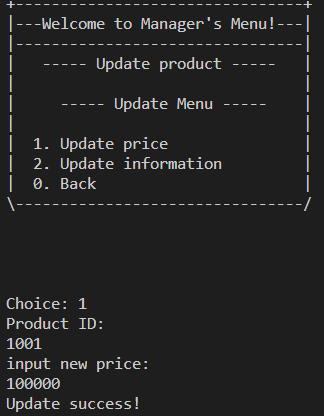
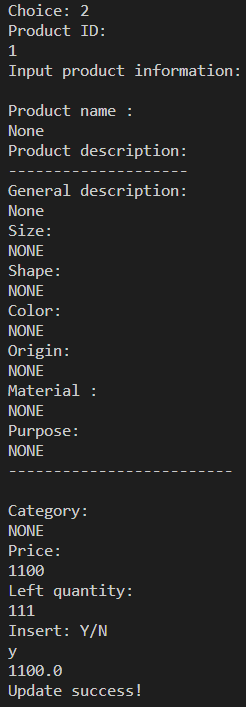
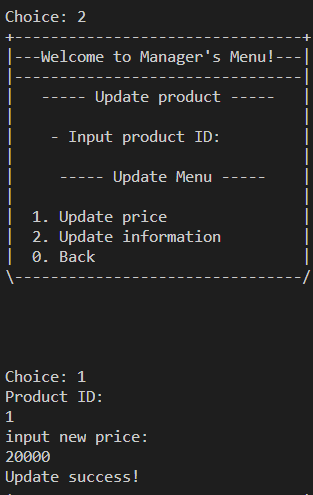
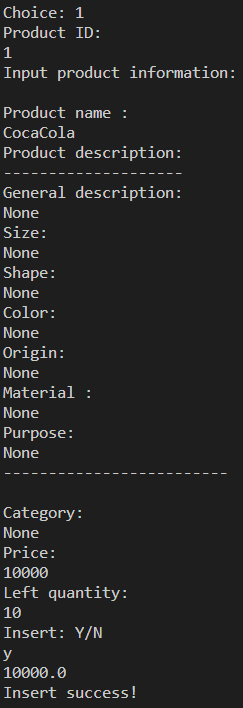
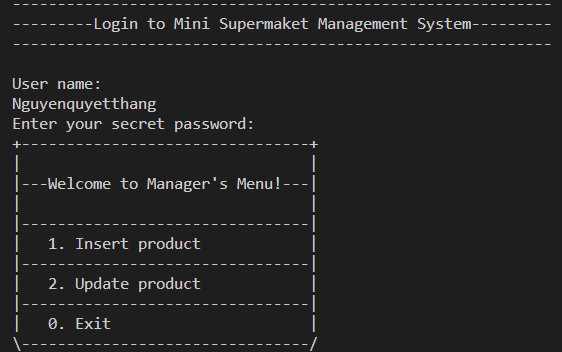
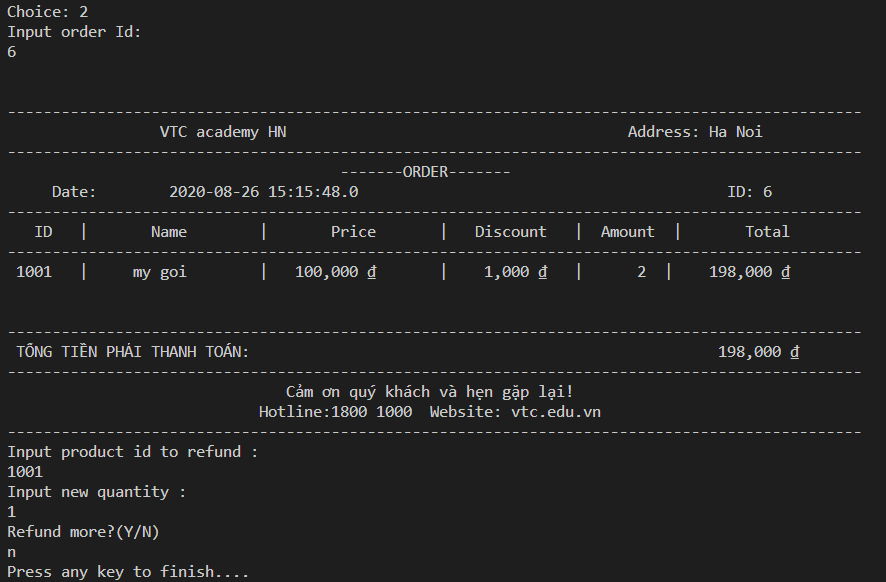
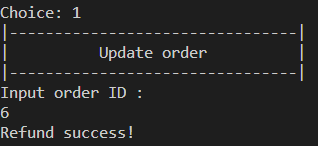
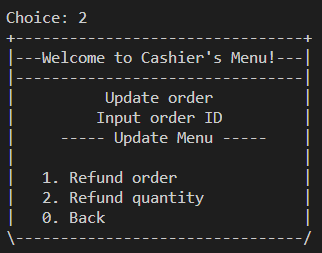
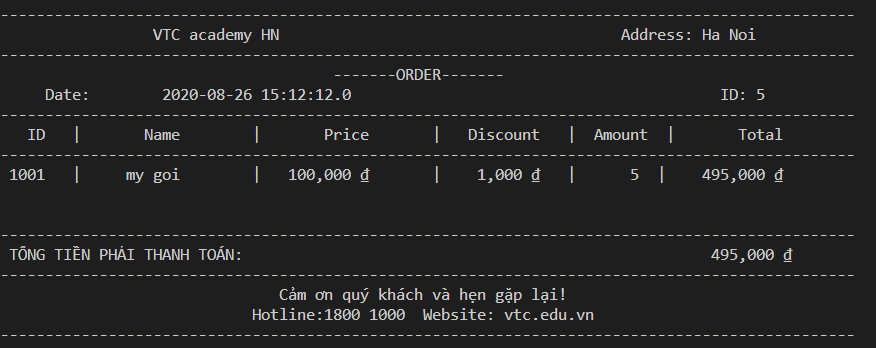
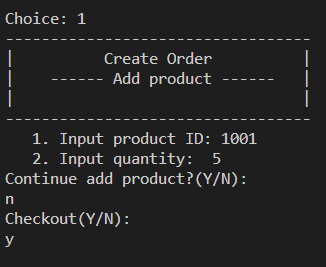
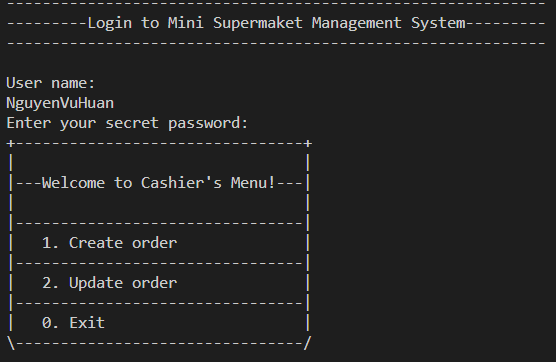


UC05\_Update Order(SequenceDiagram)



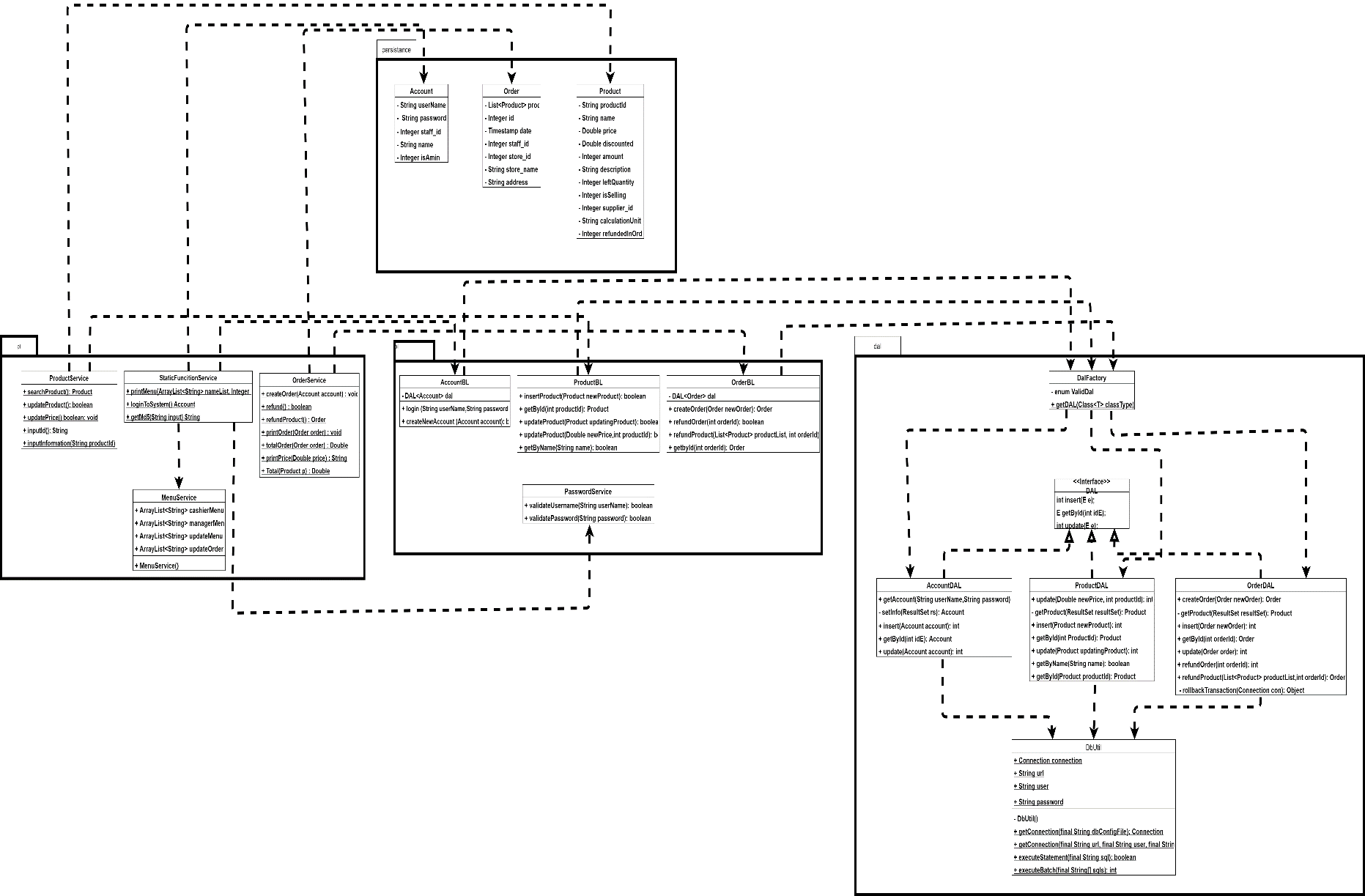
III. Design Details

1. UI Design



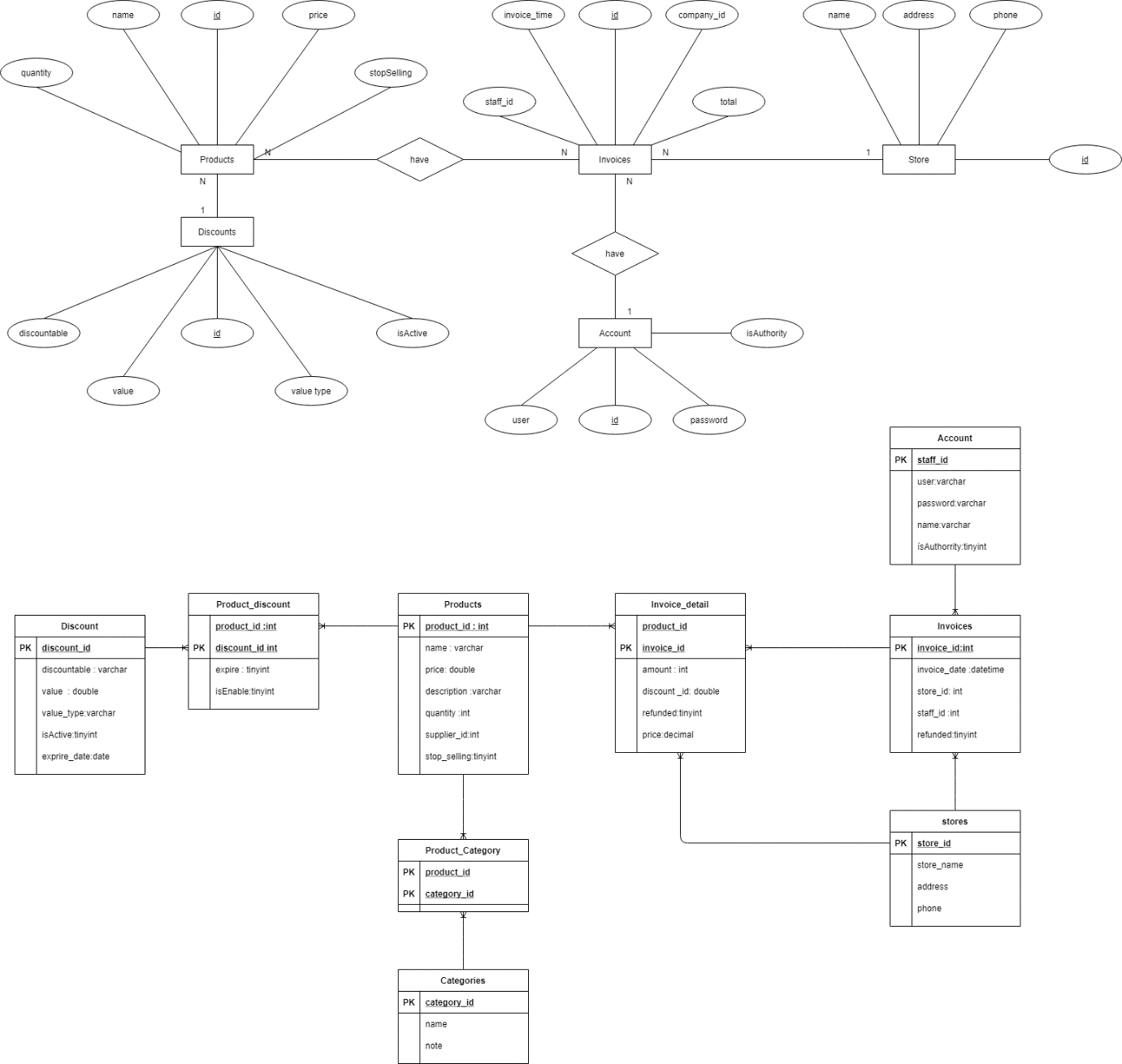
2. Code Design (Class Diagram)

(Class Diagram):

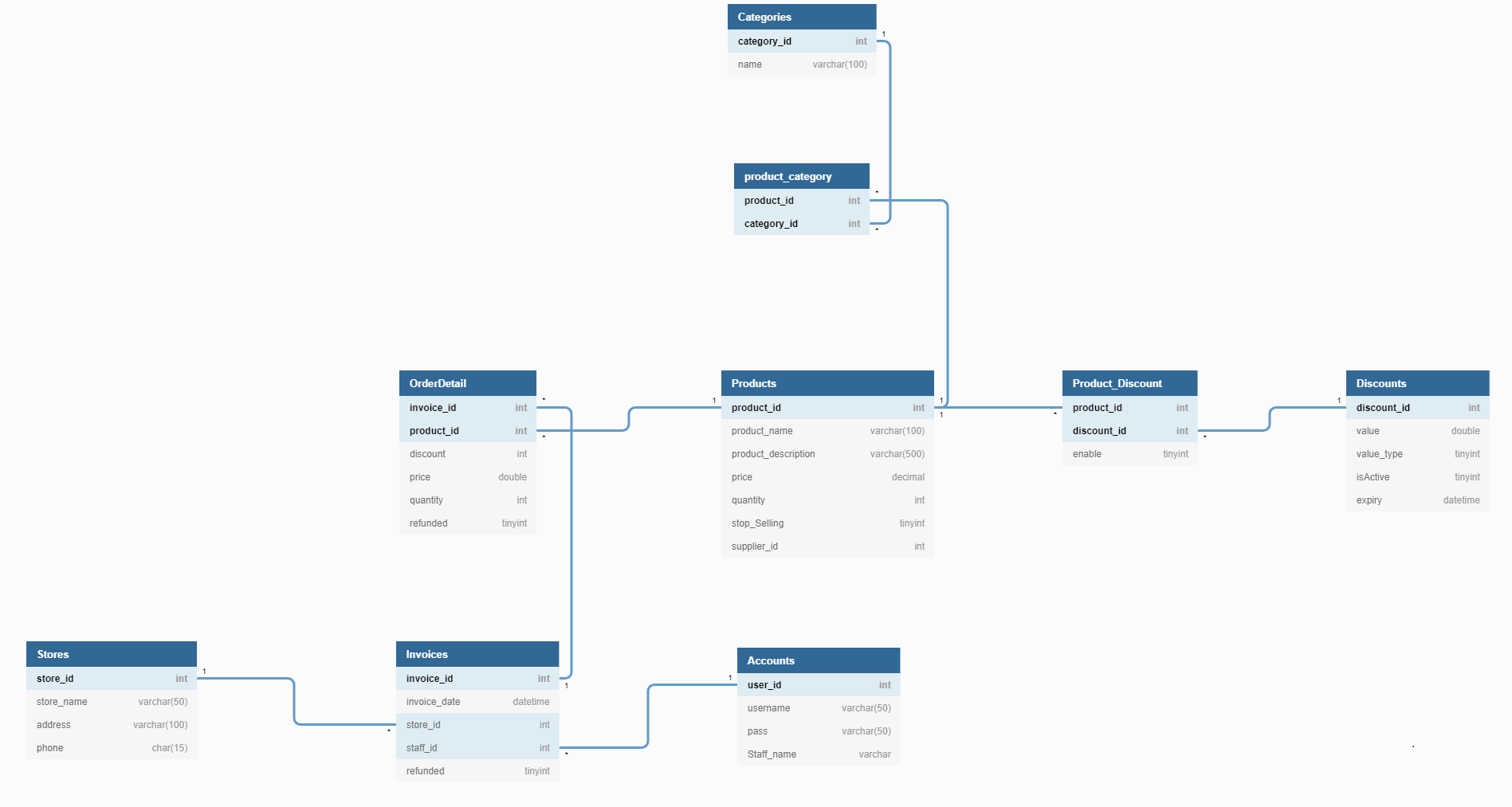


3. Database Design

a. Entity Relationship Diagram



b. Database Design Details



|  |  |  |  |
| --- | --- | --- | --- |
| **Products** | | | |
| **Column name** | **Data Type** | **Constraints** | **Description** |
| product\_id | int | Primary key |  |
| product\_name | Varchar(100) | unique |  |
| product\_description | Varchar(500) |  |  |
| price | decimal | >0 |  |
| quantity | int | >0 |  |
| stop\_selling | tinyint | NOT NULL,default(1) | 0: true 1: false |
| supplier\_id | int | NOT NULL, FK |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Discounts** | | | |
| **Column name** | **Data Type** | **Constraints** | **Description** |
| discount\_id | Int | NOT NULL, PK |  |
| value | double | NOT NULL |  |
| value\_Type | tinyint | NOT NULL |  |
| isActive | tinyint | NOT NULL, default (0) |  |
| expiry | datetime | NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Product\_Discount** | | | |
| **Column name** | **Data Type** | **Constraints** | **Description** |
| product\_id | Int | NOT NULL, PK, FK |  |
| discount\_id | int | NOT NULL, PK, FK |  |
| enable | tinyint |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Stores** | | | |
| **Column name** | **Data Type** | **Constraints** | **Description** |
| store\_id | int | Primay key |  |
| store\_name | Varchar(100) | NOT NULL |  |
| address | Varchar(200) |  |  |
| phone | Char(15) |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Accounts** | | | |
| **Column name** | **Data Type** | **Constraints** | **Description** |
| user\_id | int | Primay key ,NOT NULL |  |
| user\_name | varchar(50) | NOT NULL,unique |  |
| pass | Varchar(50) | NOT NULL |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Invoices** | | | |
| **Column name** | **Data Type** | **Constraints** | **Description** |
| Invoice\_id | int | Primary key ,NOT NULL |  |
| Invoice\_date | datetime | NOT NULL |  |
| Store\_id | int | FK |  |
| Staff\_id | int | FK |  |
| refunded | tinyint | NOT NULL, default(1) | 0: true, 1: fasle |

|  |  |  |  |
| --- | --- | --- | --- |
| **Order\_Detail** | | | |
| **Column name** | **Data Type** | **Constraints** | **Description** |
| Invoice\_id | int | PK, FK, NOT NULL |  |
| Product\_id | int | PK, FK, NOT NULL |  |
| Discount\_id | int | FK |  |
| price | double | >0 |  |
| quantity | int | >0 |  |
| refunded | tinyint | Default (1) | 0: true, 1: false |

IV. Test

1. Test login

|  |  |
| --- | --- |
| Test CaseNumber | 1.1 (getAccountTest1) |
| Test Case Name | Test login functionality |
| Test Case description | check the system's login ability |
| Preconditions |  |
| Test Case Input | "Nguyenquyetthang","16c0ce36e334e22fda8caca1  b10c2f9c", "Nguyenquyetthang",null,1,"thang",0 |
| Test Case expected output | True |
| Test Case Steps | Step1: create instance of acccountDAL  Step2: Login  Step3: Create expected Account  Step4: Check if the expected account is correct or not |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.2(getAccountTest2) |
| Test Case Name | Test login functionality |
| Test Case description | check the system's login ability |
| Preconditions |  |
| Test Case Input | “staff”, “123456” |
| Test Case expected output | True |
| Test Case Steps | Step1: create instance of acccountDAL  Step2: Login  Step3: Create expected Account  Step4: Check if the expected account is correct or not |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.3(getAccountTest3) |
| Test Case Name | Test login functionality |
| Test Case description | check the system's login ability |
| Preconditions |  |
| Test Case Input | “staff1”, “1234” |
| Test Case expected output | True |
| Test Case Steps | Step1: create instance of acccountDAL  Step2: Login  Step3: Create expected Account  Step4: Check if the expected account is correct or not |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.4(getAccountTest4) |
| Test Case Name | Test login functionality |
| Test Case description | check the system's login ability |
| Preconditions |  |
| Test Case Input | “staff1”, “1234” |
| Test Case expected output | True |
| Test Case Steps | Step1: create instance of acccountDAL  Step2: Login  Step3: Create expected Account  Step4: Check if the expected account is correct or not |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.5 (validdateUsername) |
| Test Case Name | Test username form |
| Test Case description | UserName mustn't have special charater and It contains at least 8 characters and at most 20 characters |
| Preconditions |  |
| Test Case Input | “Nguyenquyetthang” |
| Test Case expected output | True |
| Test Case Steps | Step1: Call validateUsername()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.6 (validdateUsername2) |
| Test Case Name | Test username form |
| Test Case description | Username have special character |
| Preconditions |  |
| Test Case Input | “Nguyenquyetthang123%” |
| Test Case expected output | False |
| Test Case Steps | Step1: Call validateUsername()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.7(validdateUsername3) |
| Test Case Name | Test username from |
| Test Case description | Username less than 8 character |
| Preconditions |  |
| Test Case Input | “Nguy” |
| Test Case expected output | False |
| Test Case Steps | Step1: Call validateUsername()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.8(validdateUsername4) |
| Test Case Name | Test username from |
| Test Case description | UserName more than 20 character |
| Preconditions |  |
| Test Case Input | “Nguyenquyetthang123asdvghasdvghasvghasvdhvas” |
| Test Case expected output | False |
| Test Case Steps | Step1: Call validateUsername()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.9(validdatePassword) |
| Test Case Name | Test password |
| Test Case description | It contains at least 8 characters and at most 20 characters.  It contains at least one digit.  It contains at least one upper case alphabet.  It contains at least one lower case alphabet.  It doesn’t contain any white space |
| Preconditions |  |
| Test Case Input | “Thangnguenquyet123” |
| Test Case expected output | True |
| Test Case Steps | Step1: Call validatePassword()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.10 (validdatePassword2) |
| Test Case Name | Test password |
| Test Case description | It wasn’t contains at least one digit. |
| Preconditions |  |
| Test Case Input | “Thangnguyenquyet” |
| Test Case expected output | False |
| Test Case Steps | Step1: Call validatePassword2()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.11(validdatePassword3) |
| Test Case Name | Test password |
| Test Case description | It wans’t contains at least upper case alphabet |
| Preconditions |  |
| Test Case Input | “thangnguyenquyet123” |
| Test Case expected output | False |
| Test Case Steps | Step1: Call validatePassword3()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.12(validdatePassword4) |
| Test Case Name | Test password functionality |
| Test Case description | It wans’t contains at least lower case alphabet |
| Preconditions |  |
| Test Case Input | “THANGNGUYENQUYET” |
| Test Case expected output | False |
| Test Case Steps | Step1: Call validatePassword4()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.13(validdatePassword5) |
| Test Case Name | Test password functionality |
| Test Case description | It contains at least 8 characters |
| Preconditions |  |
| Test Case Input | “THA” |
| Test Case expected output | False |
| Test Case Steps | Step1: Call validatePassword4()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.14(validdatePassword6) |
| Test Case Name | Test password functionality |
| Test Case description | It contains at more 20 characters |
| Preconditions |  |
| Test Case Input | "THAasdfasvgasbvghdvasghdasvasdvsavghasdgvdgasv" |
| Test Case expected output | False |
| Test Case Steps | Step1: Call validatePassword4()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 1.15(testMD5) |
| Test Case Name | Encrypt password |
| Test Case description | Ensure security for password |
| Preconditions |  |
| Test Case Input | “Thangnguyenquyet123” |
| Test Case expected output | 16c0ce36e334e22fda8caca1b10c2f9c |
| Test Case Steps | Step1: Call getMD5()  Step2: Input value  Step3: Create expected output  Step4: Compare result |
| Default Value preverving |  |

1. Test order

|  |  |
| --- | --- |
| Test CaseNumber | 2.1( getByIdTest) |
| Test Case Name | Test order functionality |
| Test Case description | wrong orderId |
| Preconditions |  |
| Test Case Input | 1001 |
| Test Case expected output | Null |
| Test Case Steps | Step1: Call getByidTest()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 2.2(getByIdTest2) |
| Test Case Name | Test order functionality |
| Test Case description | orrect id |
| Preconditions |  |
| Test Case Input | 1 |
| Test Case expected output | Null |
| Test Case Steps | Step1: Call getByIdTest2()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 2.3(createOrder) |
| Test Case Name | Test order functionality |
| Test Case description | correct |
| Preconditions |  |
| Test Case Input | 1001,"phobo",50000.,10000.,1 |
| Test Case expected output | Null |
| Test Case Steps | Step1: Call createOrder()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 2.4( createOrder1) |
| Test Case Name | Test order functionality |
| Test Case description | order null |
| Preconditions |  |
| Test Case Input |  |
| Test Case expected output | Null |
| Test Case Steps | Step1: Call createOrder1()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |  |
| --- | --- |
| Test CaseNumber | 2.5( createOrder2) |
| Test Case Name | Test order functionality |
| Test Case description | wrong product id |
| Preconditions |  |
| Test Case Input | 10010,"phobo",50000.,10000.,1 |
| Test Case expected output | Null |
| Test Case Steps | Step1: Call createOrder2()  Step2: Input value  Step3: Test |
| Default Value preverving |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 2.6( createOrder3) |  | Test Case Name | Test order functionality |  | Test Case description | List null |  | Preconditions |  |  | Test Case Input | “Ha Noi” |  | Test Case expected output | Null |  | Test Case Steps | Step1: Call createOrder3()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 2.7( createOrder3) |  | Test Case Name | Test order functionality |  | Test Case description | List null |  | Preconditions |  |  | Test Case Input | 10010,"phobo",50000.,10000.,1 |  | Test Case expected output | Null |  | Test Case Steps | Step1: Call createOrder3()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 2.8(refundProductTest) |  | Test Case Name | Test order functionality |  | Test Case description | List null |  | Preconditions |  |  | Test Case Input | 1001 |  | Test Case expected output | Null |  | Test Case Steps | Step1: Call refundProductTest ()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

1. Test product

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.1( getProductById) |  | Test Case Name | Test product functionality |  | Test Case description | corrrect id |  | Preconditions |  |  | Test Case Input | 1001,"phobo",20000.0,null,null,"da update",20,1,1,null |  | Test Case expected output | True |  | Test Case Steps | Step1: Call getProductById()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.2( getProductById2) |  | Test Case Name | Test product functionality |  | Test Case description | wrong id |  | Preconditions |  |  | Test Case Input | 1 |  | Test Case expected output | Null |  | Test Case Steps | Step1: Call getProductById2()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.3(insertProduct) |  | Test Case Name | Test product functionality |  | Test Case description | Already existed |  | Preconditions |  |  | Test Case Input | 1001,"my goi 5",10000.,"update from java app",10,1,"my goi" |  | Test Case expected output | Fale |  | Test Case Steps | Step1: Call insertProduct()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.4(updatePrice) |  | Test Case Name | Test product functionality |  | Test Case description | valid price |  | Preconditions |  |  | Test Case Input | 50000.,1001 |  | Test Case expected output | True |  | Test Case Steps | Step1: UpdatePrice()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.5(updatePrice2) |  | Test Case Name | Test product functionality |  | Test Case description | wrong price (price<0) |  | Preconditions |  |  | Test Case Input | -50000.,1001 |  | Test Case expected output | False |  | Test Case Steps | Step1: UpdatePrice2()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.6(updatePoduct) |  | Test Case Name | Test product functionality |  | Test Case description | update in desciption |  | Preconditions |  |  | Test Case Input | 1002,"banh my",15000.,"update from java app dbsjajdasjad",50,1,"my goi" |  | Test Case expected output | True |  | Test Case Steps | Step1: UpdateProduct()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.7(updatePoduct1) |  | Test Case Name | Test product functionality |  | Test Case description | update false: wrong id |  | Preconditions |  |  | Test Case Input | 1111,"banh my",15000.,"update from java app dbsjajdasjad",50,1,"my goi" |  | Test Case expected output | False |  | Test Case Steps | Step1: UpdateProduct1()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.8(updatePoduct2) |  | Test Case Name | Test product functionality |  | Test Case description | update false: wrong price (<0) |  | Preconditions |  |  | Test Case Input | 1002,"banh my",-15000.,"update from java app dbsjajdasjad",50,1,"my goi" |  | Test Case expected output | False |  | Test Case Steps | Step1: UpdateProduct2()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.9(updatePoduct3) |  | Test Case Name | Test product functionality |  | Test Case description | update false: duplicate name |  | Preconditions |  |  | Test Case Input | 1002,"phobo",15000.,"update from java app dbsjajdasjad",50,1,"my goi" |  | Test Case expected output | False |  | Test Case Steps | Step1: UpdateProduct3()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.10(updatePoduct4) |  | Test Case Name | Test product functionality |  | Test Case description | update false: null |  | Preconditions |  |  | Test Case Input | productBL.updateProduct(product) |  | Test Case expected output | False |  | Test Case Steps | Step1: UpdateProduct4()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.11(getByName) |  | Test Case Name | Test product functionality |  | Test Case description | exist :True |  | Preconditions |  |  | Test Case Input | “phobo” |  | Test Case expected output | True |  | Test Case Steps | Step1: getByName()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

|  |
| --- |
| Test |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseNumber | 3.12(getByName1) |  | Test Case Name | Test product functionality |  | Test Case description | exist :False |  | Preconditions |  |  | Test Case Input | “phobo2” |  | Test Case expected output | True |  | Test Case Steps | Step1: getByName2()  Step2: Input value  Step3: Test |  | Default Value preverving |  |  |

V. Assign work to each team member

|  |  |  |
| --- | --- | --- |
| Group  4 | Project Name |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | Task name | Description | Start Date | End Date | Member | Self assessment |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | System design | System model architecture | 24/07/2020 |  | Nguyễn Quyết Thắng |  |  | 2 | Use case design | Design system features | 30/07/2020 |  | Nguyễn Quyết Thắng,  Trần Văn Huân,  Nguyễn Vũ Huân |  |  | 3 | Entity relationships design | Identify the entities in the system and the relationships between them | 28/07/2020 | 29/07/2020 | Trần Văn Huân  Nguyễn Quyết Thắng |  |  | 4 | Entity relationships design details | Construct tables that represent properties of entities and the relationship between entities | 29/07/2020 | 04/08/2020 | Nguyễn Quyết Thắng |  |  | 5 | Design database | Database building | 30/07/2020 | 06/08/2020 | Nguyễn Quyết Thắng |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | Use case description | Describe in detail the use case | 01/08/2020 | 06/08/2020 | Nguyễn Quyết Thắng  Trần Văn Huân  Nguyễn Vũ Huân |  |  | 7 | Design activity diagram for each use case | Design the system's task processing flowchart for each use case | 01/08/2020 | 07/08/2020 | Nguyễn Vũ Huân  Trần Văn Huân  Nguyễn Quyết Thắng |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | Design sequence diagram for each use case | Design sequence diagrams for the system's workflow for each use case | 01/08/2020 | 25/08/2020 | Trần Văn Huân  Nguyễn Vũ Huân  Nguyễn Quyết Thắng |  |  | 9 | Class diagram for system | Designing detailed drawing class for the system | 03/08/2020 | 20/08/2020 | Trần Văn Huân Nguyễn Vũ Huân  Nguyễn Quyết Thắng |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | Coding | Write code based on class diagrams | 15/08/2020 | 26/8/2020 | Nguyễn Quyết Thắng  Trần Văn Huân  Nguyễn Vũ Huân |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | Bug fixes, optimizations, upgrades for the code | Fix errors, optimize code, upgrade system features | 15/8/2020 |  | Nguyễn Quyết Thắng |  |  | 12 | Deployment diagram | Drawing up the project environment | 01/08/2020 | 26/08/2020 | Nguyễn Quyết Thắng  Trần Văn Huân  Nguyễn Vũ Huân |  |  | 13 | Attach report | Report writing | 07/08/2020 | 26/08/2020 | Nguyễn Vũ Huân  Trần Văn Huân  Nguyễn Quyết Thắng |  |  | 14 | Wriet slides | Presentation material | 26/08/2020 | 26/08/2020 | Nguyễn Vũ Huân  Trần Văn Huân  Nguyễn Quyết Thắng |  |  |

VI. Installation Instructions

1. Deployment Diagram

Deployment diagram of a mini market management System

