

**PROJECT REPORT**

**Moto Store**

Semester: Programming Fundamentals

Class: PF08

Group: Group 5

Instructor: Nguyễn Xuân Sinh

Team member: Lê Văn Được

Đồng Đức Lân

Index

Project Name 1

Index 2

1. Project introduction 2
2. Analyze System Requirements 3
3. Design Details 30
4. Test 42
5. Assign work to each team member 60
6. Installation Instructions 61

Appendix

I.Project introduction

The system describes the functions that employees can manage the products in the store on the console, helps employees create invoices according to customers' requirements and update information from the menu. This product will help employees to easily meet the needs of customers

1. Proposed System

* Insert product
* Update product
* Create order
* Update order

1. The scope of the project to be applied

* Applicable for large displacement vehicle dealers

1. System Name

* Product management and invoice management software

1. Deployment Environment

* Desktop deployment environment (windows)
* Visual studio code
* IntelliJ Idea
* MySQL Workbenck 8.0.20

1. Development Tools

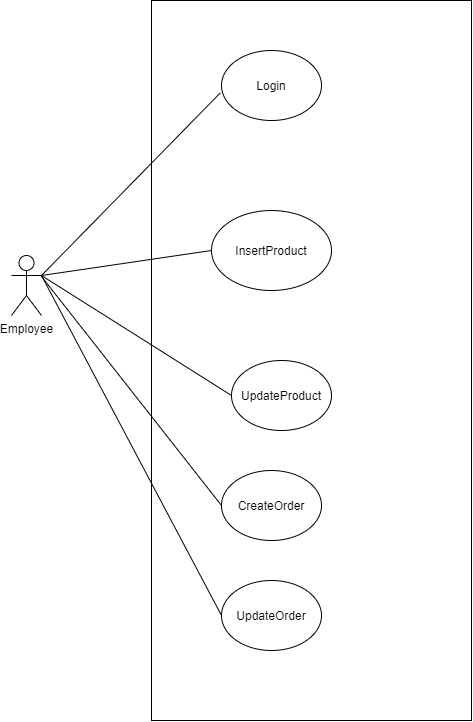
* Development tools in the form of java
* Jdk8
* Apache maven
* Junit

1. Customer Requirements(System features)

* The system can satisfy customer requirements when creating order
* If you have created an invoice that is not correct or information is incorrect, you can edit the details of each part, giving customers peace of mind when creating orders and making payments.

II. Analyze System Requirements

1. Use case



1.

|  |  |
| --- | --- |
| Name | Login |
| ID | UC\_001 |
| Description | Allows employee to access the system, using the functions of the system |
| Actors | Employee |
| Organizational Benefits | Giving employee access to system functions |
| Frequency of Use | Always |
| Trigger | Run app |
| Preconditions | * Employee account has been created * The device is connected to the Internet when logging in |
| Postconditions | * Employee successfully log into the system and can use the functions of the system |
| Main Course | 1. Input username 2. Input pasword(EX2) 3. Click login 4. Check username/password(EX4) 5. Show main menu(AC5.1) |
| Alternate Courses | AC5.1 Show menu  - Insert product  - Update product  - Create order  - Update order  - Exit |
| Exceptions | EX2: Valid password  - Password does not contain special characters  - Password must more than 8 characters, using special characters and numbers  - Return main course step 2  EX4:Login fail  - Return main course step 1 |

2.

|  |  |
| --- | --- |
| Name | Insert Products |
| ID | UC\_002 |
| Description | Allow logged-in employee to add products |
| Actors | Employee |
| Organizational Benefits | Employee add products to the system |
| Frequency of Use | Sometime |
| Trigger | Choose insert product |
| Preconditions | 1. The employee is logged into the system 2. Show main menu |
| Postconditions | 1. The products are saved to the system and displayed message 2. Return main menu |
| Main Course | 1. Show menu class product(AC1) 2. Choice class product(EX2) 3. Insert product information(AC3,EX3) 4. Show product new 5. Save product(AC5,EX5) 6. Want to continue or not and back to main menu(AC6,EX6) 7. Return main menu |
| Alternate Courses | AC1: Menu product   1. Motor 2. Clothes 3. Hat 4. Gloves 5. Exit   AC3:Insert product   * Name(EX3.1) * Description(EX3.2) * Color(EX3.3) * Size(EX3.4) * Price(EX3.5) * Time Warranty(EX3.6)   AC5:Save product   1. OK  * Return main course step 6  1. Cancel  * Return main course step 6   AC6:Continue or not   1. Yes  * Return main course step 1  1. No  * Return main course step 7 |
| Exceptions | EX2: Valid incorrect   * Return main course step 2   EX3: Product not found   * Return main course step3   EX3.1: Valid incorrect   * Return to AC3   EX3.2: Valid incorrect   * Return to AC3   EX3.3: Valid incorrect   * Return to AC3   EX3.4: Valid incorrect   * Return to AC3   EX3.5: Valid incorrect   * Return to AC3   EX3.6: Valid incorrect   * Return to AC3   EX4: Valid incorrect   * Return main course step 4   EX6: Valid incorrect   * Return main course step 6 |
|  |  |

3.

|  |  |
| --- | --- |
| Name | Update Product |
| ID | UC\_003 |
| Description | Allows employee to update product information in the data warehouse |
| Actors | Employee |
| Organizational Benefits | Employee can manage the product inside the system |
| Frequency of Use | Sometimes |
| Trigger | Choose update product |
| Preconditions | 1. The employee is logged into the system 2. Show main menu |
| Postconditions | 1. Update product information successfully and display message 2. Return main menu |
| Main Course | 1. Show menu category product(AC1) 2. Choice category product(EX2) 3. Show list product follow category 4. Input id product (EX4) 5. Show product 6. Input new information product(AC6) 7. Show product with new information 8. Update or not (AC8,EX8) 9. Want to continue or not and back to main menu(AC9,EX9) 10. Return main menu |
| Alternate Courses | AC1:Show menu class product   1. Moto 2. Hat 3. Clothes 4. Groves   AC6:Input information product   * Name(EX6.1) * Description(EX6.2) * Color(EX6.3) * Size(EX6.4) * Price(EX6.5) * Time Warranty(EX6.6)   AC8: Update or not   1. OK  * Return main course 9  1. Cancel  * Return main course 9   AC9:Continue or not and back to main menu   1. Yes  * Return main course step 1  1. No  * Return main course step 9 |
| Exceptions | EX2: Valid incorrect   * Return to main course step 2   EX4: Product not found   * Return to Main Course step 4   EX6.1: Valid information incorrect   * Return to AC6   EX6.2: Valid information incorrect   * Return to AC6   EX6.3: Valid information incorrect   * Return to AC6   EX6.4: Valid information incorrect   * Return to AC6   EX6.5: Valid information incorrect   * Return to AC6   EX6.6: Valid information incorrect   * Return to AC6   EX8: Valid incorrect   * Return main course step 8   EX9: Valid incorrect   * Return main course step 9 |

4. Create Order

|  |  |
| --- | --- |
| Name | Create Order |
| ID | UC\_004 |
| Description | Allows employee to create order when selling the products to customers. Orders generated are full of information and are saved |
| Actors | Employee |
| Organizational Benefits | 1. Manage products sold 2. Summary of revenue in cycles |
| Frequency of Use | Sometimes |
| Trigger | Choose create order |
| Preconditions | 1. Employee login seccessful to system 2. Show main menu |
| Postconditions | 1. Order created must have enough information 2. Print Order 3. Return main menu |
| Main Course | 1. Employee input information of customers (AC1). 2. Input category(EX2). 3. Show list product follow category 4. Input productID(EX4) 5. Input quantity(EX5) 6. Continue add product or not(AC6,EX6) 7. Save or not and back to main menu (AC7,EX7). 8. Print order. 9. Return main menu |
| Alternate Courses | AC1:Input information   * Name Customer(EX1.1) * Address Customer(EX1.2) * Phone Number(EX1.3)   AC6:Input product continue or not   1. Yes  * Return main course step 3  1. No  * Return main course step 7   AC7:Save order or not   1. OK  * Return main course step 8  1. Cancel  * Return main course step 9 |
| Exceptions | EX1.1:Valid incorrect   * Return main course step 1   EX1.2:Valid incorrect   * Return main course step 1   EX1.3:Valid incorrect   * Return main course step 1   EX2: Product doesn’t exist.   * Display “Product not found”. * Return main course step 2.   EX4: Valid incorrect   * Return main course step 4   EX5: Valid incorrect   * Return main course step 5   EX6: Valid incorrect   * Return main course step 6   EX7: Valid incorrect   * Return main course step 7 |

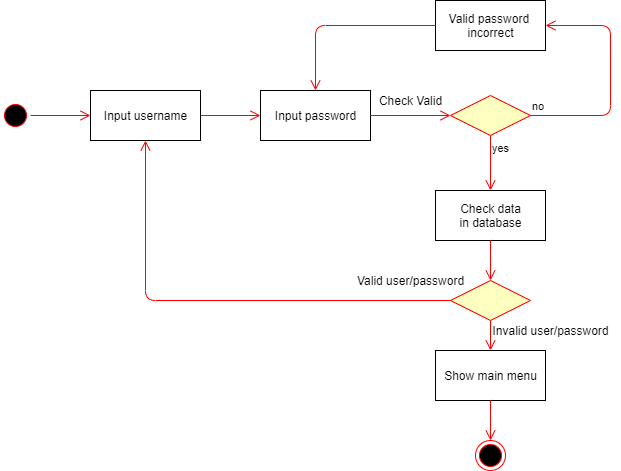
5. Update Order

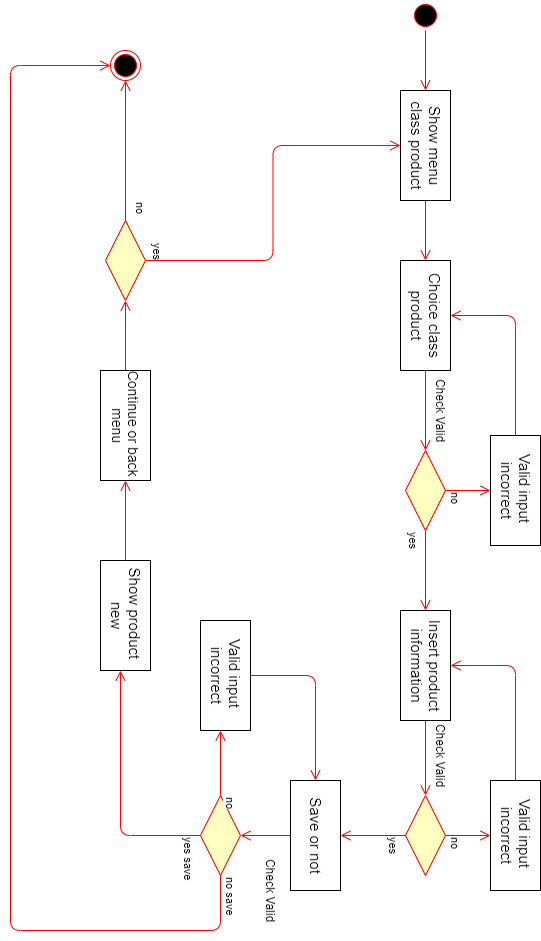
|  |  |
| --- | --- |
| Name | Update Order\_1 |
| ID | UC\_005.1 |
| Description | Allows employee to update invoice information in the data warehouse, edit customer information, product information. |
| Actors | Employee |
| Organizational Benefits | Updated invoice with wrong information |
| Frequency of Use | Hardly ever |
| Trigger | Choose update order |
| Preconditions | 1. Employee successfully logged into the system 2. Show main menu |
| Postconditions | 1. Update order information successfully 2. Save order 3. Return main menu |
| Main Course | 1. Input order id (EX1). 2. Show order 3. Show menu function(AC3) 4. Update order Customer(AC4). 5. Show order new update 6. Save or Not(AC5,EX5) 7. Print order new 8. Return main menu. |
| Alternate Courses | AC3: Menu function   1. Update customer 2. Update products   AC4: Employee update customer in order.   * Name Customer(EX4.1) * Address Customer(EX4.2) * Phone Number(EX4.3)   AC6: Save update order or not   1. OK  * Return main course step 7  1. Cancel  * Return main course step 8 |
| Exceptions | EX1: Valid incorrect   * Return main course step 1   EX4.1: Valid incorrect   * Return main course step 3   EX4.2: Valid incorrect   * Return main course step 3   EX4.3: Valid incorrect   * Return main course step 3   EX6: Valid incorrect   * Return main course step 6 |

|  |  |
| --- | --- |
| Name | Update Order\_2 |
| ID | UC\_005.2 |
| Description | Allows employee to update invoice information in the data warehouse, edit customer information, product information. |
| Actors | Employee |
| Organizational Benefits | Updated invoice with wrong information |
| Frequency of Use | Hardly ever |
| Trigger | Choose update order |
| Preconditions | 1. Employee successfully logged into the system 2. Show main menu |
| Postconditions | 1. Update order information successfully 2. Save order 3. Return main menu |
| Main Course | 1. Input order id (EX1). 2. Show order 3. Show menu function(AC3) 4. Update order Product. 5. Input categoryID(EX5). 6. Show list product follow category 7. Input productID(EX7) 8. Input quantity(EX8) 9. Continue add product or not(AC9,EX9) 10. Show order new update 11. Save or Not(AC11,EX11) 12. Print order new 13. Return main menu. |
| Alternate Courses | AC3: Menu function.   1. Update customer 2. Update products   AC9:Input product continue or not   1. Yes  * Return main course step 4  1. No  * Return main course step 9   AC11: Save update order or not   1. OK  * Return main course step 12  1. Cancel  * Return main course step 13 |
| Exceptions | EX1: Valid incorrect   * Return main course step 1   EX5: Valid incorrect   * Return main course step 5   EX7: Valid incorrect   * Return main course step 6   EX8: Valid incorrect   * Return main course step 8   EX9: Valid incorrect   * Return main course step 9   EX11: Valid incorrect   * Return main course step 11 |

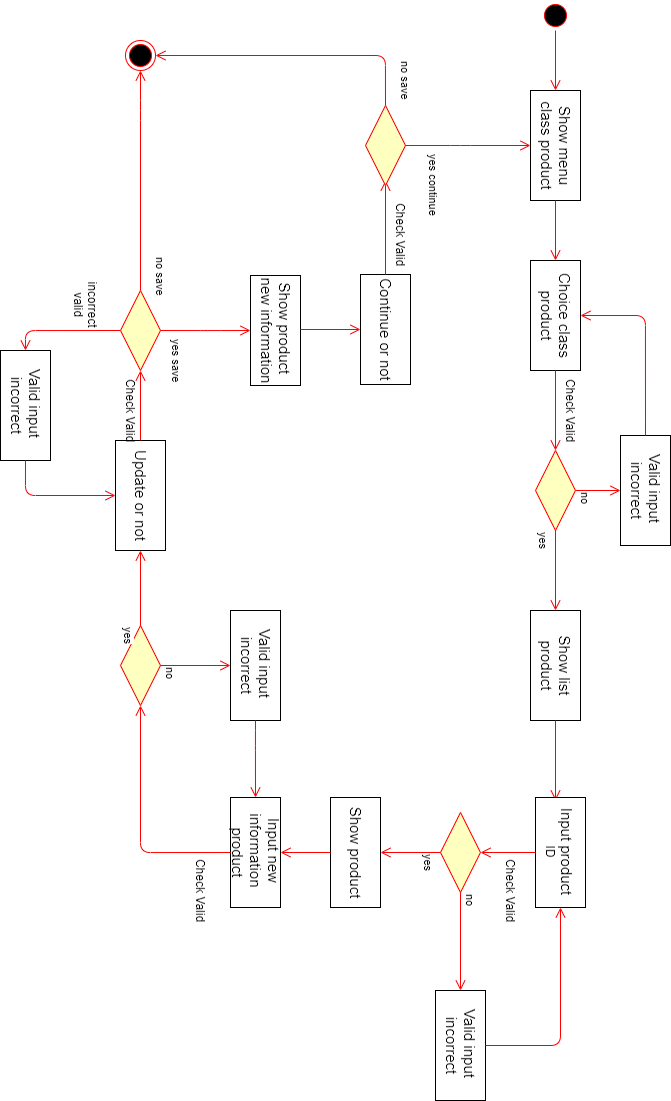
1. Activity Diagrams:­­­­­­

Login

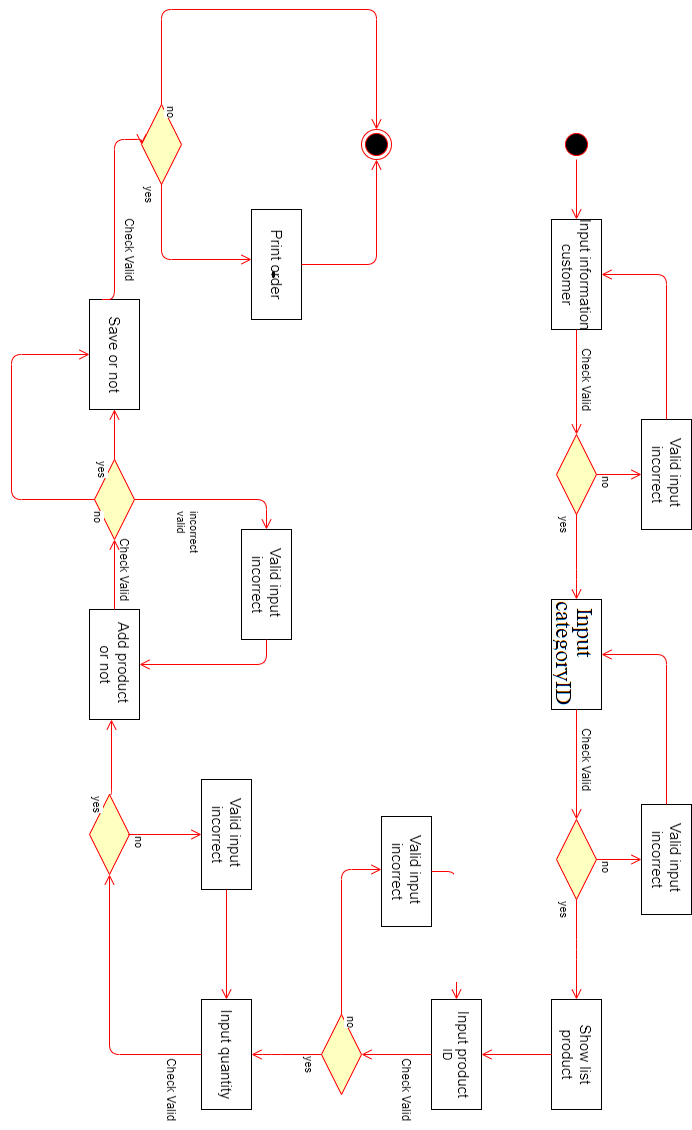


Insert Product:

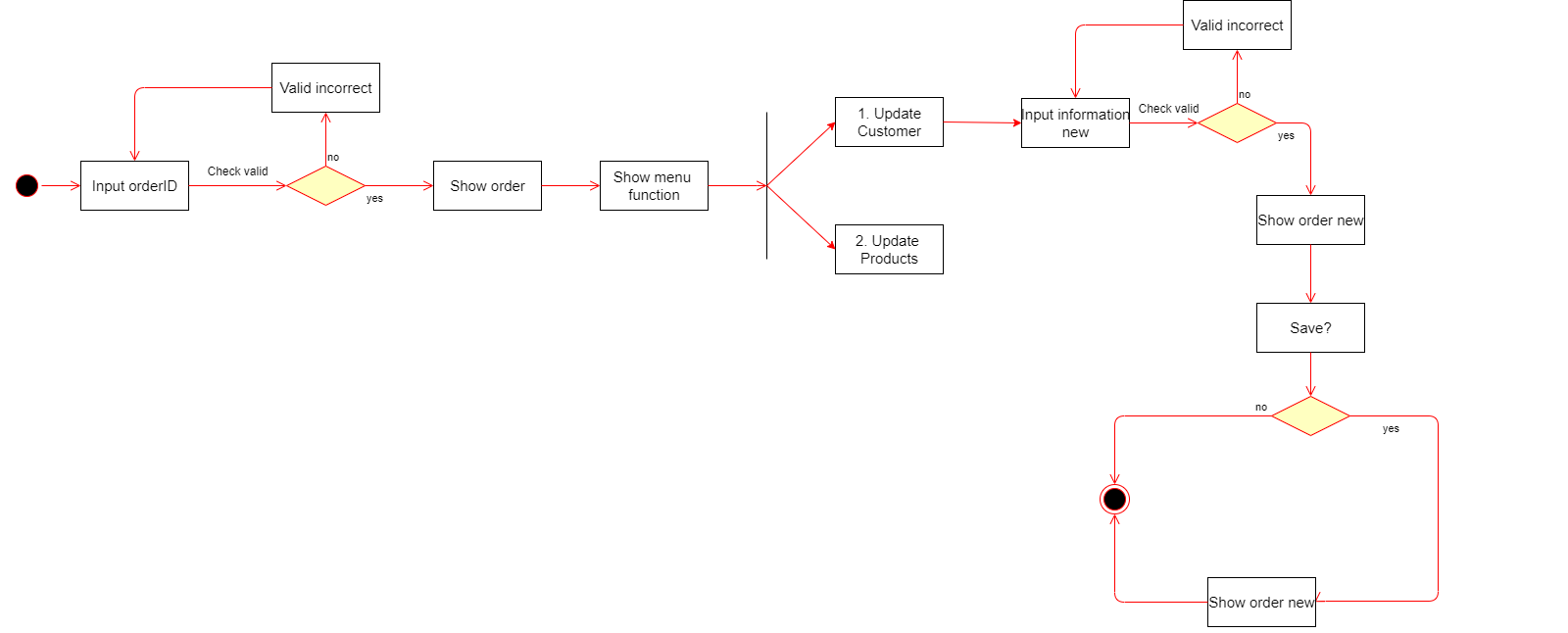
Update Product:

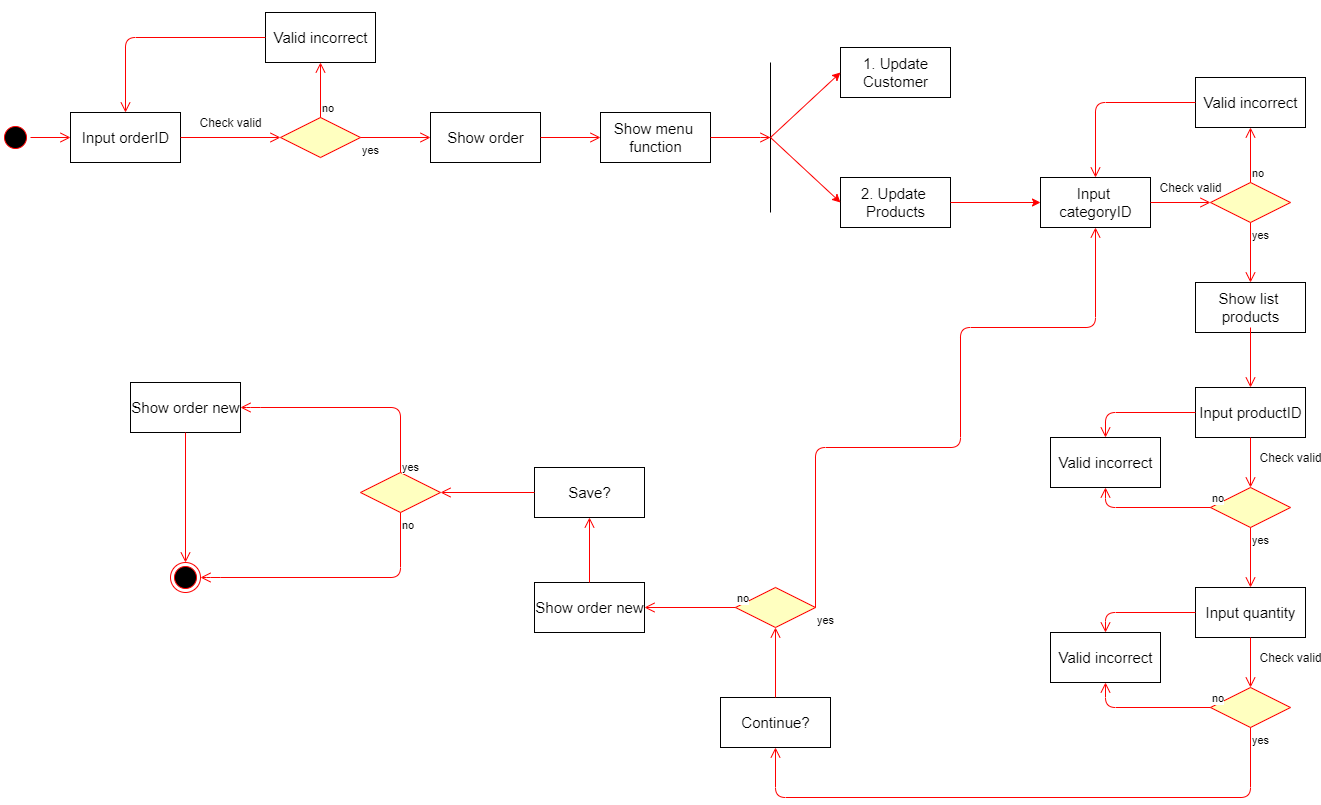


Create Order:

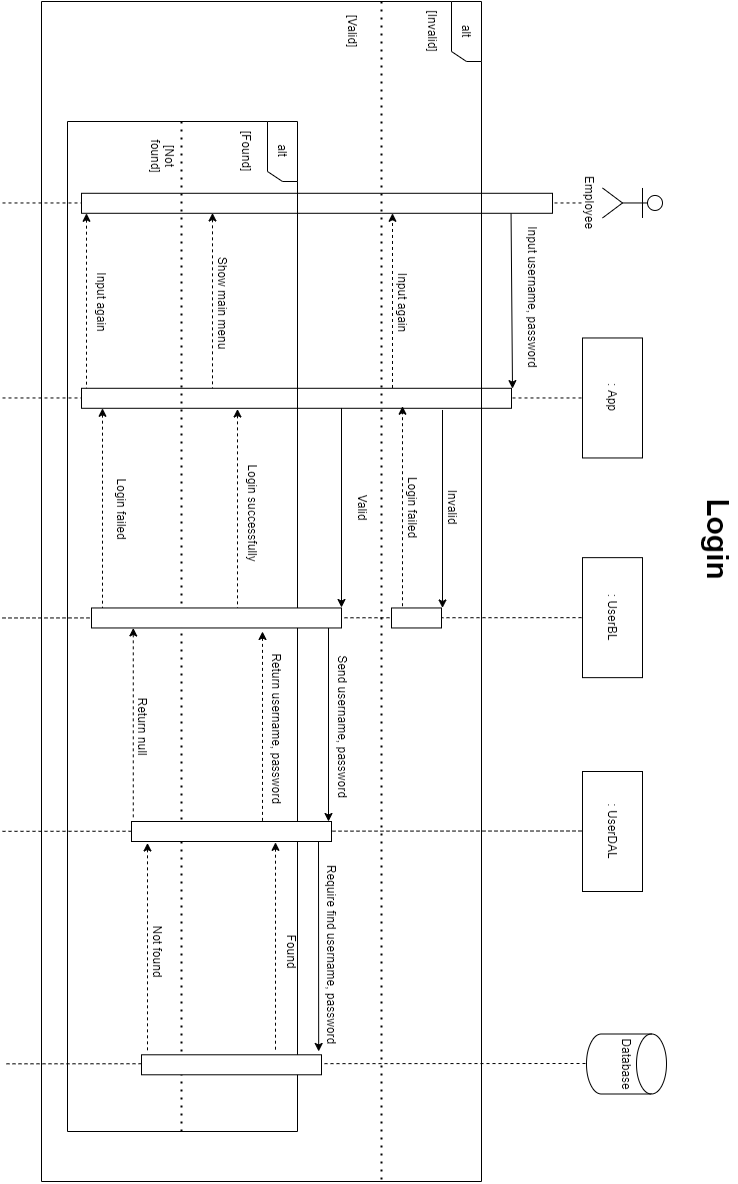


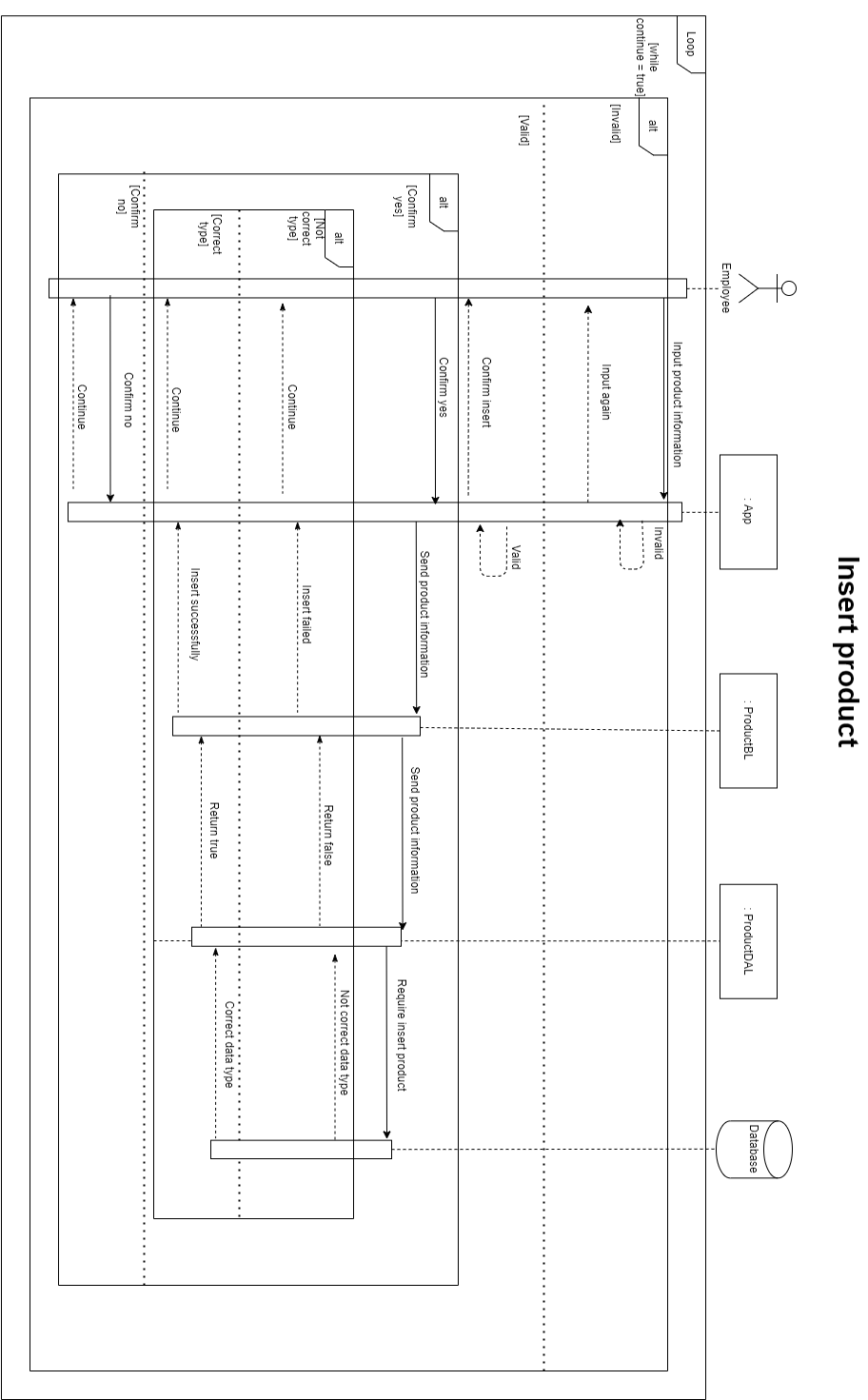
Update Order:

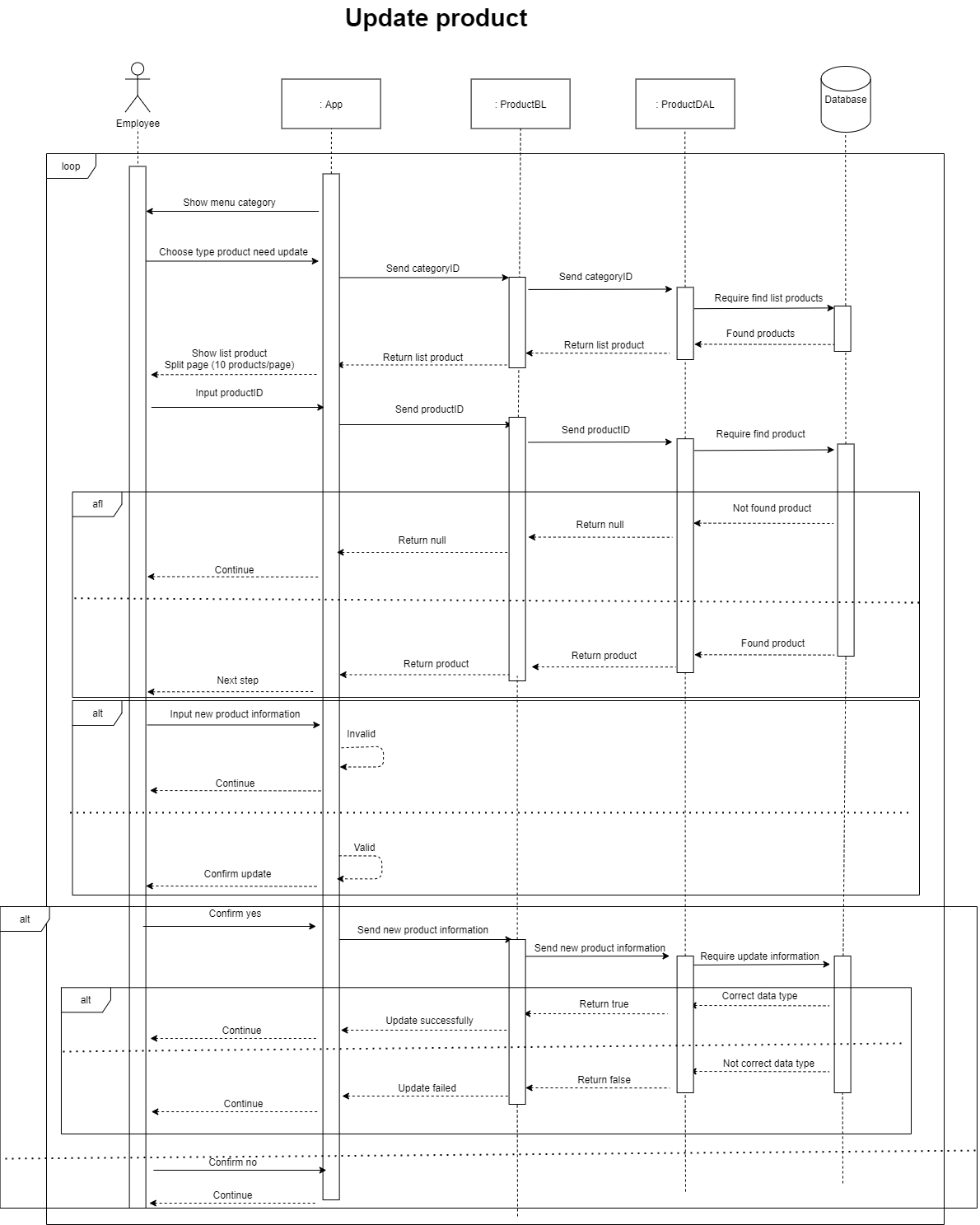


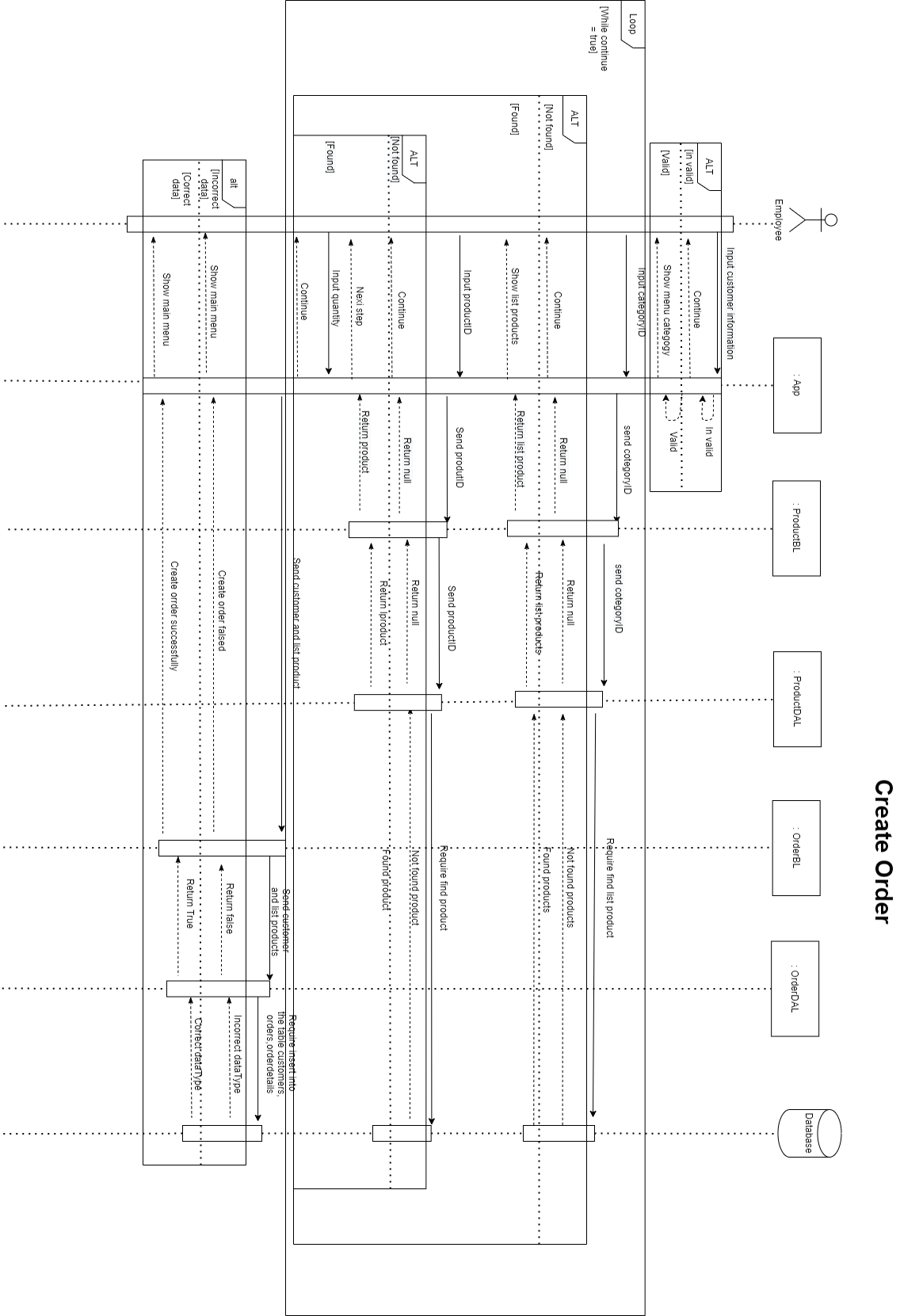


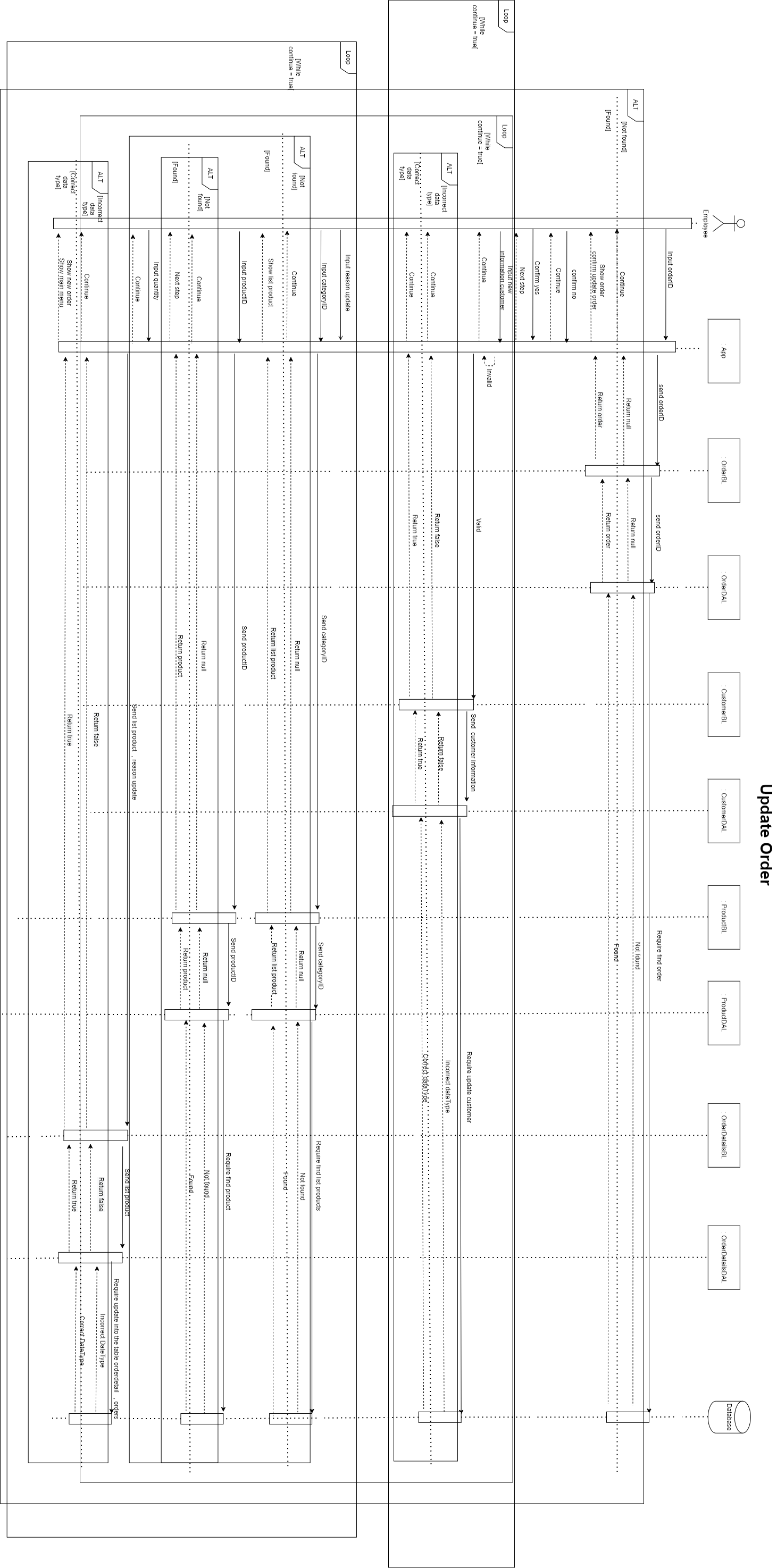
1. Sequence Diagrams:





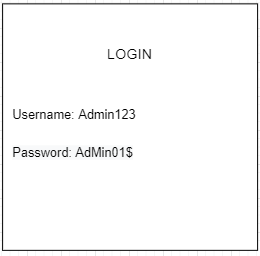




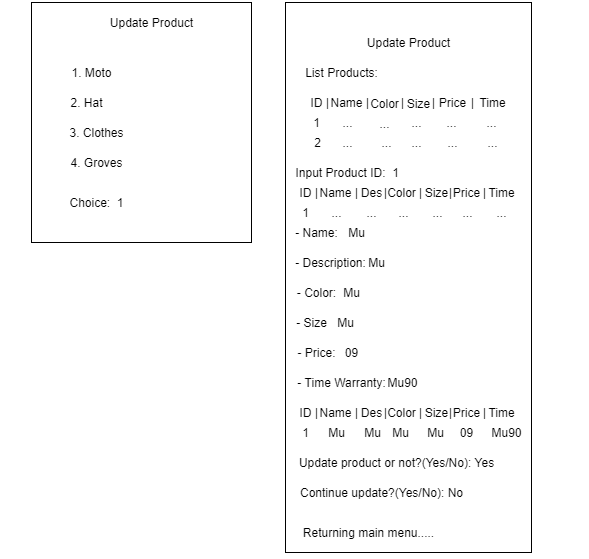


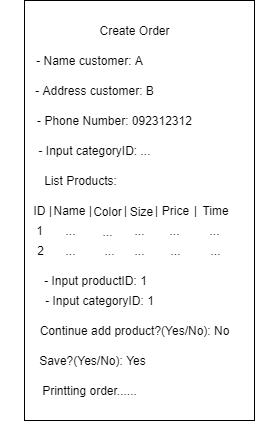
III. Design Details

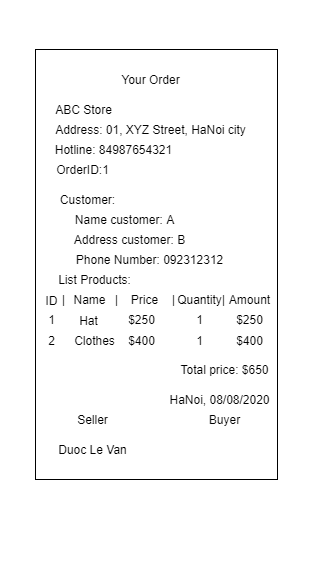
1. UI Design

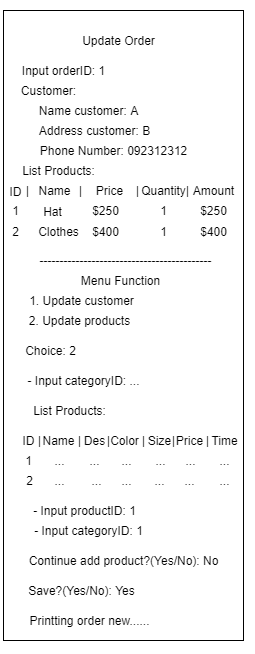
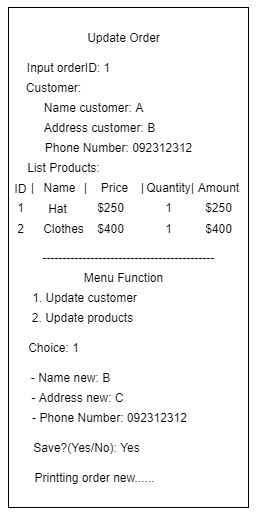




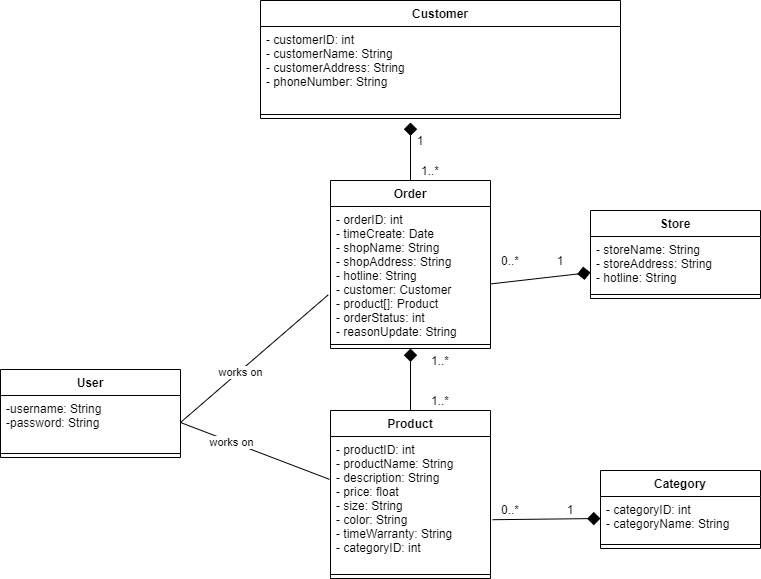


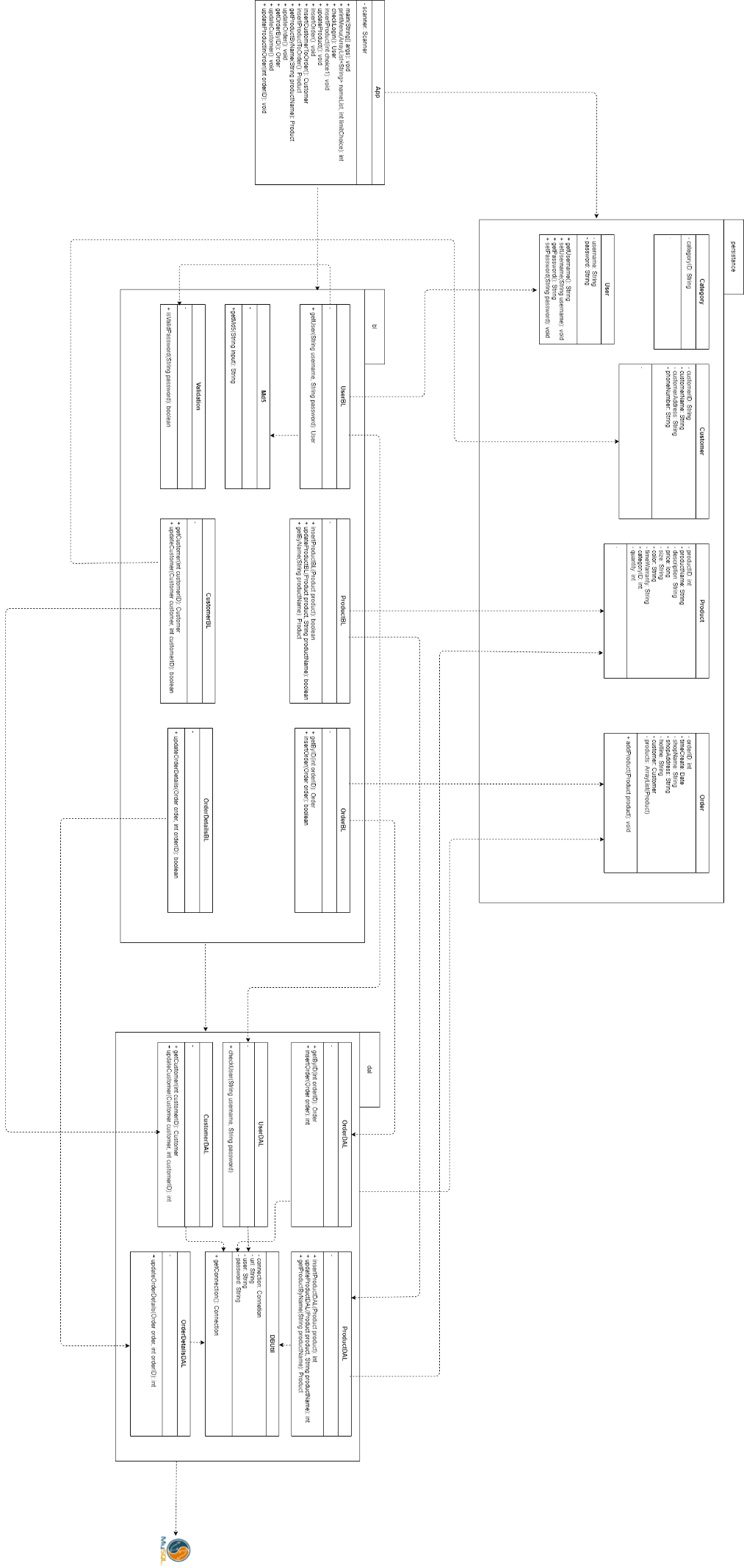




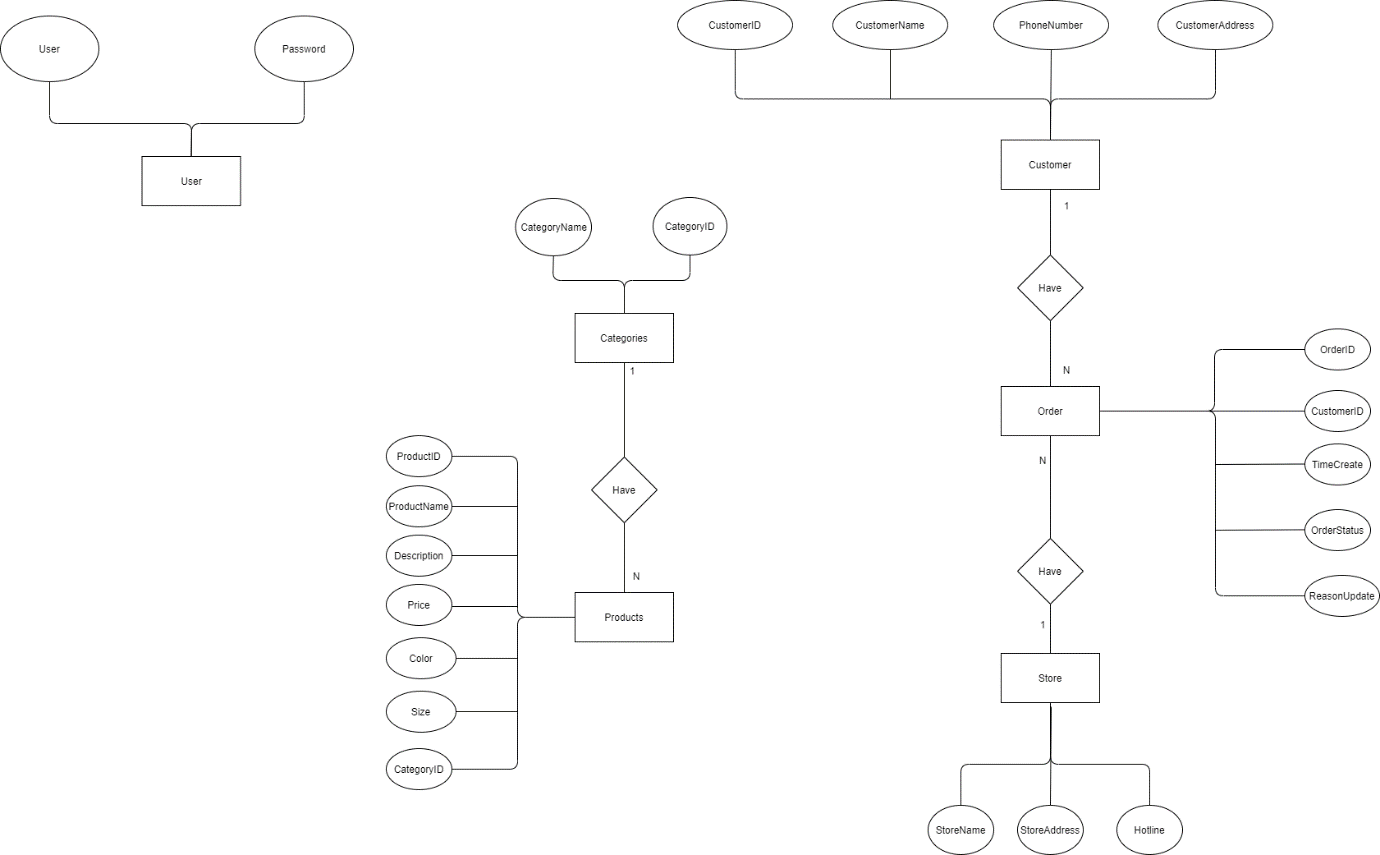


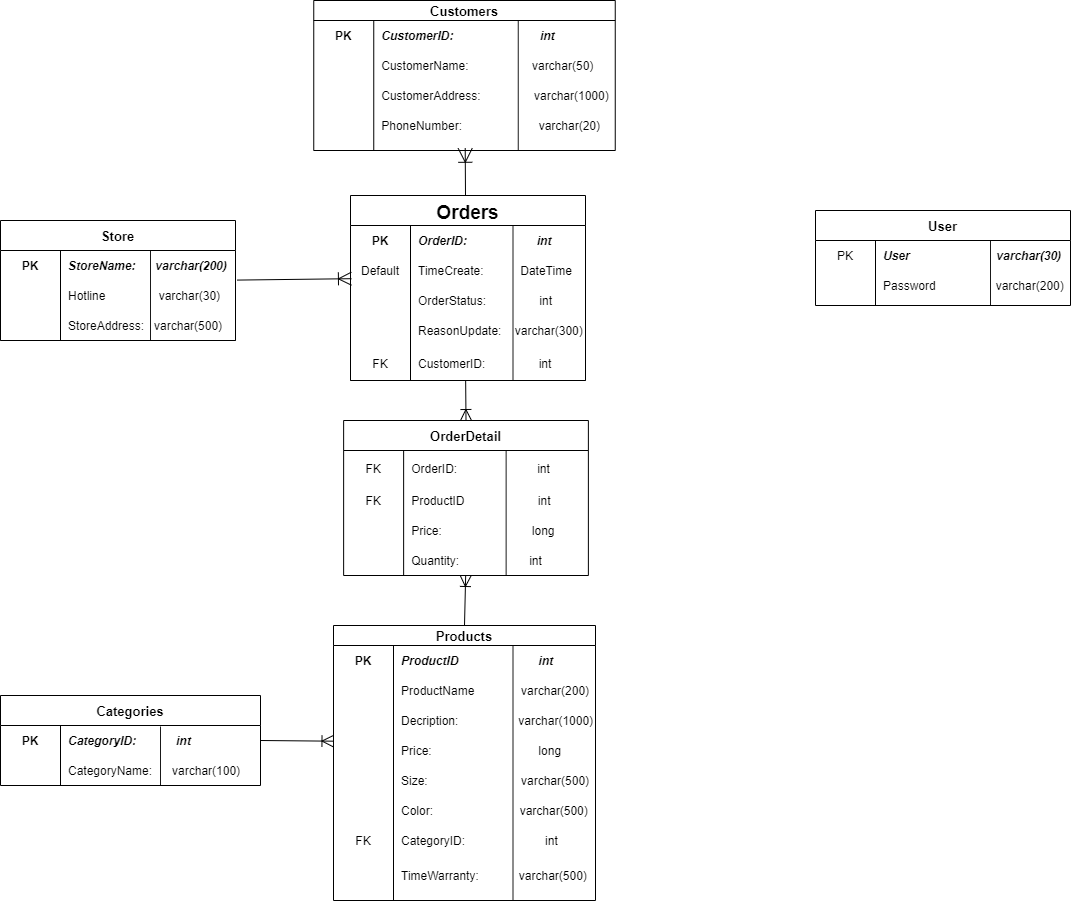
1. Code Design (Class diagram)



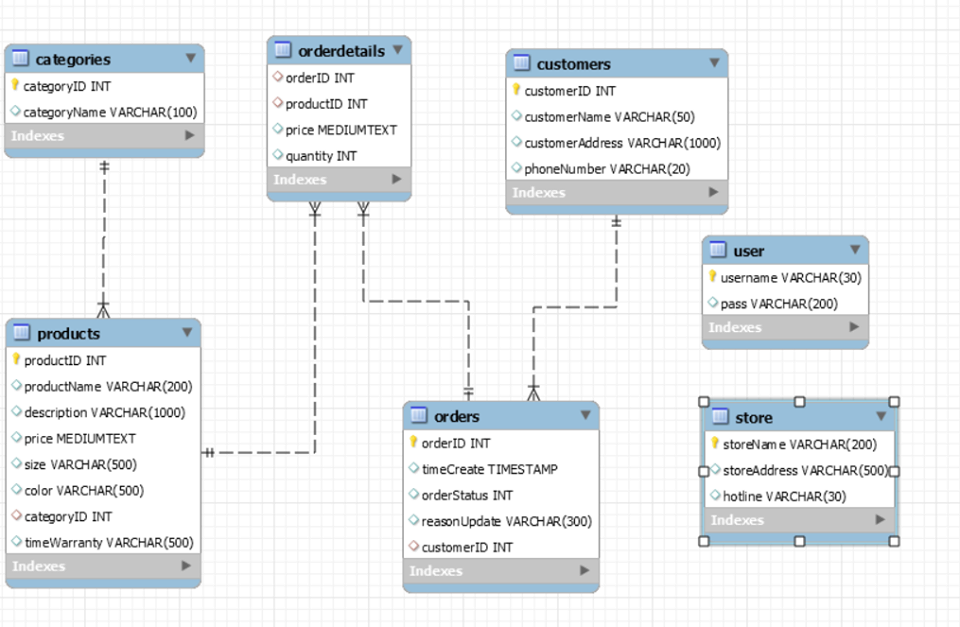


1. Database Design
   1. Entity Relationship Diagram





* 1. Database Design Details



|  |  |  |  |
| --- | --- | --- | --- |
| User | | | |
| Colum Name | Data Type | Constraints | Description |
| Username | Varchar(30) | Prymary key |  |
| Password | Varchar(200) | NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Order | | | |
| Colum Name | Data Type | Constraints | Description |
| OrderID | Int | Primary key |  |
| TimeCreate | DateTime | DateTime |  |
| OrderStatus | Int | NOT NULL |  |
| ReasonUpdate | Varchar(300) | NOT NULL |  |
| CustomerID | Int | Foreign key |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Products | | | |
| Colum Name | Data Type | Constraints | Description |
| ProductID | Int | Primary key |  |
| ProductName | Varchar(200) | NOT NULL |  |
| Description | Varchar(1000) | NOT NULL |  |
| Price | Long | NOT NULL |  |
| Size | Varchar(500) | NOT NULL |  |
| Color | Varchar(500) | NOT NULL |  |
| CategoryID | Int | Foreign key |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Store | | | |
| Colum Name | Data Type | Constraints | Description |
| StoreName | Varchar(200) | Primary key |  |
| PhoneNumberStore | Varchar(11) | NOT NULL |  |
| StoreAddress | Varchar(50) | NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Customer | | | |
| Colum Name | Data Type | Constraints | Description |
| CustomerID | Int | Primary key |  |
| CustomerName | Varchar(50) | NOT NULL |  |
| CustomerAddress | Varchar(1000) | NOT NULL |  |
| PhoneNumber | Varchar(10) | NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| OrderDetail | | | |
| Colum Name | Data Type | Constraints | Description |
| OrderID | Int | Foreign key |  |
| ProductID | Int | Foreign key |  |
| Price | Long | NOT NULL |  |
| Quantity | Int | NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Categories | | | |
| Colum Name | Data Type | Constraints | Description |
| CategoryID | Int | Primary key |  |
| CategoryName | Varchar(100) | NOT NULL |  |

IV. Test

|  |  |
| --- | --- |
| Test Case Number | 1.1.1 |
| Test Case Name | Test UserDAL |
| Test Case Description | Test Case check input correct username and password |
| Precondition |  |
| Test Case Input | Account, password is correct |
| Test Case Expected Output | Returns the account, password |
| Test Case Step | 1. Create DAL connection to database  2. Check the username, password on the databse  3. Create the resulting string array is the account, password  4. Create the desired result string array as account, password,  5. Compare the 2 arrays above |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.1.2 |
| Test Case Name | Test UserDAL |
| Test Case Description | Test Case check input wrong username and correct password |
| Precondition |  |
| Test Case Input | Wrong username, correct password |
| Test Case Expected Output | NULL |
| Test Case Step | 1. Create DAL connection to database  2. Check the username, password on the databse  3. Return NULL |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.1.3 |
| Test Case Name | Test UserDAL |
| Test Case Description | Test Case check input correct username and wrong password |
| Precondition |  |
| Test Case Input | Correct username, wrong password |
| Test Case Expected Output | Null |
| Test Case Step | 1. Create DAL connection to database  2. Check the username, password on the databse  3. Result Null |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.1.4 |
| Test Case Name | Test UserDAL |
| Test Case Description | Test Case check input username null and correct password |
| Precondition |  |
| Test Case Input | Null username, correct password |
| Test Case Expected Output | Null |
| Test Case Step | 1. Create DAL connection to database  2. Check the username, password on the databse  3. Result Null |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.1.5 |
| Test Case Name | Test UserDAL |
| Test Case Description | Test Case check input correct username and password null |
| Precondition |  |
| Test Case Input | Correct username, null password |
| Test Case Expected Output | Null |
| Test Case Step | 1. Create DAL connection to database  2. Check the username, password on the databse  3. Result Null |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.1.6 |
| Test Case Name | Test UserDAL |
| Test Case Description | Test Case check input null username and password null |
| Precondition |  |
| Test Case Input | Null username, null password |
| Test Case Expected Output | Null |
| Test Case Step | 1. Create DAL connection to database  2. Check the username, password on the databse  3. Result Null |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.2.1 |
| Test Case Name | Test Md5 |
| Test Case Description | Test Case check function encrypt password |
| Precondition |  |
| Test Case Input | Correct password |
| Test Case Expected Output | True |
| Test Case Step | 1. Input encrypted string 2. Create string wish result 3. Compare them   4. Result True |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.2.2 |
| Test Case Name | Test Md5.1 |
| Test Case Description | Test Case check function encrypt password |
| Precondition |  |
| Test Case Input | Wrong password |
| Test Case Expected Output | False |
| Test Case Step | 1. Input encrypted string 2. Create string wish result 3. Compare them   3. Result False |
| Default Value Prevering |  |
| Test Case Number | 1.3.1 |
| Test Case Name | Test ValidPassword |
| Test Case Description | Test Case check ability to format passwords |
| Precondition |  |
| Test Case Input | Correct format password |
| Test Case Expected Output | True |
| Test Case Step | 1. Input Valided string 2. Valid that string 3. Create array with wish result 4. Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.3.2 |
| Test Case Name | Test ValidPassword |
| Test Case Description | Test Case check ability to format passwords |
| Precondition |  |
| Test Case Input | String lack of special characters |
| Test Case Expected Output | False |
| Test Case Step | 1. Input Valided string 2. Valid that string 3. Create array with wish result 4. Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.3.3 |
| Test Case Name | Test ValidPassword |
| Test Case Description | Test Case check ability to format passwords |
| Precondition |  |
| Test Case Input | String lack of upper character |
| Test Case Expected Output | False |
| Test Case Step | 1. Input Valided string 2. Valid that string 3. Create array with wish result 4. Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.3.4 |
| Test Case Name | Test ValidPassword |
| Test Case Description | Test Case check ability to format passwords |
| Precondition |  |
| Test Case Input | String lack of lower character |
| Test Case Expected Output | False |
| Test Case Step | 1. Input Valided string 2. Valid that string 3. Create array with wish result 4. Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.3.5 |
| Test Case Name | Test ValidPassword |
| Test Case Description | Test Case check ability to format passwords |
| Precondition |  |
| Test Case Input | String lack of number character |
| Test Case Expected Output | False |
| Test Case Step | 1. Input Valided string 2. Valid that string 3. Create array with wish result 4. Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.3.6 |
| Test Case Name | Test ValidPassword |
| Test Case Description | Test Case check ability to format passwords |
| Precondition |  |
| Test Case Input | String less than 8 characters |
| Test Case Expected Output | False |
| Test Case Step | 1. Input Valided string 2. Valid that string 3. Create array with wish result 4. Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.3.7 |
| Test Case Name | Test ValidPassword |
| Test Case Description | Test Case check ability to format passwords |
| Precondition |  |
| Test Case Input | String more than 20 characters |
| Test Case Expected Output | False |
| Test Case Step | 1. Input Valided string 2. Valid that string 3. Create array with wish result 4. Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.4.1 |
| Test Case Name | Test UserBL |
| Test Case Description | Test Case check ability receive and direct data of UserBL |
| Precondition |  |
| Test Case Input | Correct username,password |
| Test Case Expected Output | True |
| Test Case Step | 1. Receive username and password from client, then send them to UserDAL 2. Create array string receive result from UserDAL 3. Create string wish result 4. Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.4.2 |
| Test Case Name | Test UserBL |
| Test Case Description | Test Case check ability receive and direct data of UserBL |
| Precondition |  |
| Test Case Input | Correct username,wrong password |
| Test Case Expected Output | False |
| Test Case Step | 1. Receive username and password from client, then send them to UserDAL 2. Create array string receive result from UserDAL 3. Create string wish result 4. Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.5.1 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Get Order, Return Order |
| Precondition |  |
| Test Case Input | OrderID |
| Test Case Expected Output | True |
| Test Case Step | * Create an order to get the results returned * Create another order and set its value * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.5.2 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | No order received, Return null |
| Precondition |  |
| Test Case Input | OrderID |
| Test Case Expected Output | Null |
| Test Case Step | * Create an order to get the results returned * Create another order * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.6.1 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Create order correct information, return true |
| Precondition |  |
| Test Case Input | Correct information |
| Test Case Expected Output | True |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.6.2 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | NameCustomer is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value(NameCustomer set its null) * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.6.3 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | Address is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value(Address set its null) * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.6.4 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | PhoneNumber is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value(PhoneNumber set its incorrect) * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.6.5 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | Indentity is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value(Indentity set its incorrect) * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.6.6 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | ProductName is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value(product 1,2:ProductName is incorrect value) * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.6.7 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | ProductName is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value(product 2:ProductName is incorrect value) * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.6.8 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | Quantity is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value(product 1:Quantity is incorrect value) * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.6.9 |
| Test Case Name | OrderBLTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | Price is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value(product 1:Quantity is incorrect value) * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.7.1 |
| Test Case Name | Test Insert productDAL |
| Test Case Description | Create new information product, return true |
| Precondition |  |
| Test Case Input | Correct information |
| Test Case Expected Output | True |
| Test Case Step | * Create productDAL * Create product and set its value * Create results returned and set its value * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.7.2 |
| Test Case Name | Test Insert productDAL |
| Test Case Description | Create new information product, return false |
| Precondition |  |
| Test Case Input | ProductName is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(productname incorrect value) * Create results returned and set its value * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.7.3 |
| Test Case Name | Test Insert productDAL |
| Test Case Description | Create new information product, return false |
| Precondition |  |
| Test Case Input | Price is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Price incorrect value) * Create results returned and set its value * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.7.4 |
| Test Case Name | Test Insert productDAL |
| Test Case Description | Create new information product, return false |
| Precondition |  |
| Test Case Input | Category is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Category incorrect value) * Create results returned and set its value * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.8.1 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Create order correct information, return true |
| Precondition |  |
| Test Case Input | Correct information |
| Test Case Expected Output | True |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.8.2 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | NameCustomer is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value(NameCustomer set its null) * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.8.3 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | Address is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value(Address set its null) * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.8.4 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | PhoneNumber is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value(PhoneNumber set its incorrect) * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.8.5 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | Indentity is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value(Indentity set its incorrect) * Create product 1,2 and set its value * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.8.6 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | ProductName is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value(product 1,2:ProductName is incorrect value) * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.8.7 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | ProductName is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value(product 2:ProductName is incorrect value) * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.8.8 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | Quantity is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value(product 1:Quantity is incorrect value) * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.8.9 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Create order false, return false |
| Precondition |  |
| Test Case Input | Price is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customer and set its value * Create product 1,2 and set its value(product 1:Quantity is incorrect value) * Create arratList product and add products * Create orderBL:  1. Get returned results 2. Set its value  * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.9.1 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | Get Order, Return Order |
| Precondition |  |
| Test Case Input | OrderID |
| Test Case Expected Output | True |
| Test Case Step | * Create an order to get the results returned * Create another order and set its value * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.9.2 |
| Test Case Name | OrderDALTestGetByID |
| Test Case Description | No order received, Return null |
| Precondition |  |
| Test Case Input | OrderID |
| Test Case Expected Output | Null |
| Test Case Step | * Create an order to get the results returned * Create another order * Compare the 2 with each other |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.10.1 |
| Test Case Name | ProductBLTestInsert |
| Test Case Description | Insert product, return true |
| Precondition |  |
| Test Case Input | Correct information |
| Test Case Expected Output | true |
| Test Case Step | * Create productBL * Create product and set its value * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.10.2 |
| Test Case Name | ProductBLTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Same name, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.10.3 |
| Test Case Name | ProductBLTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Price is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(Price is incorrect value) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.10.4 |
| Test Case Name | ProductBLTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Category is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(Category is incorrect) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.10.5 |
| Test Case Name | ProductBLTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Size is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(Size is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.10.6 |
| Test Case Name | ProductBLTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Color is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(Color is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.10.7 |
| Test Case Name | ProductBLTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | TimeWarranty is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(TimeWarranty is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.11.1 |
| Test Case Name | ProductDALTestInsert |
| Test Case Description | Insert product, return true |
| Precondition |  |
| Test Case Input | Correct information |
| Test Case Expected Output | true |
| Test Case Step | * Create productDAL * Create product and set its value * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.11.2 |
| Test Case Name | ProductDALTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Same name, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.11.3 |
| Test Case Name | ProductDALTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Price is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Price is incorrect value) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.11.4 |
| Test Case Name | ProductDALTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Category is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Category is incorrect) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.11.5 |
| Test Case Name | ProductDALTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Size is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Size is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.11.6 |
| Test Case Name | ProductDALTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | Color is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Color is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.11.7 |
| Test Case Name | ProductDALTestInsert |
| Test Case Description | Insert product, return false |
| Precondition |  |
| Test Case Input | TimeWarranty is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(TimeWarranty is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.12.1 |
| Test Case Name | ProductDALTestUpdate |
| Test Case Description | Update product, return true |
| Precondition |  |
| Test Case Input | Correct information |
| Test Case Expected Output | true |
| Test Case Step | * Create productDAL * Create product and set its value * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.12.2 |
| Test Case Name | ProductDALTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Same name, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.12.3 |
| Test Case Name | ProductDALTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Price is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Price is incorrect value) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.12.4 |
| Test Case Name | ProductDALTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Category is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Category is incorrect) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.12.5 |
| Test Case Name | ProductDALTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Size is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Size is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.12.6 |
| Test Case Name | ProductDALTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Color is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(Color is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.12.7 |
| Test Case Name | ProductDALTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | TimeWarranty is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productDAL * Create product and set its value(TimeWarranty is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.13.1 |
| Test Case Name | ProductBLTestUpdate |
| Test Case Description | Update product, return true |
| Precondition |  |
| Test Case Input | Correct information |
| Test Case Expected Output | true |
| Test Case Step | * Create productBL * Create product and set its value * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.13.2 |
| Test Case Name | ProductBLTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Same name, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.13.3 |
| Test Case Name | ProductBLTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Price is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(Price is incorrect value) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.13.4 |
| Test Case Name | ProductBLTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Category is incorrect, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(Category is incorrect) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.13.5 |
| Test Case Name | ProductBLTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Size is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(Size is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.13.6 |
| Test Case Name | ProductBLTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | Color is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(Color is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.13.7 |
| Test Case Name | ProductBLTestUpdate |
| Test Case Description | Update product, return false |
| Precondition |  |
| Test Case Input | TimeWarranty is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and set its value(TimeWarranty is null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.14.1 |
| Test Case Name | OrderDALTestUpdateCustomer |
| Test Case Description | Update customer , return true |
| Precondition |  |
| Test Case Input | Correct information |
| Test Case Expected Output | True |
| Test Case Step | * Create customerDAL * Set customer value * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.14.2 |
| Test Case Name | OrderDALTestUpdateCustomer |
| Test Case Description | Update customer , return false |
| Precondition |  |
| Test Case Input | CustomerName is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customerDAL * Set customer value(CustomerName is Null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.14.3 |
| Test Case Name | OrderDALTestUpdateCustomer |
| Test Case Description | Update customer , return false |
| Precondition |  |
| Test Case Input | Address is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customerDAL * Set customer value(Address is Null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.14.4 |
| Test Case Name | OrderDALTestUpdateCustomer |
| Test Case Description | Update customer , return false |
| Precondition |  |
| Test Case Input | PhoneNumber is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customerDAL * Set customer value(PhoneNumber is Null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.14.5 |
| Test Case Name | OrderDALTestUpdateCustomer |
| Test Case Description | Update customer , return false |
| Precondition |  |
| Test Case Input | IdentityCard is null, remaining correct |
| Test Case Expected Output | False |
| Test Case Step | * Create customerDAL * Set customer value(IdentityCard is Null) * Create result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.15.1 |
| Test Case Name | OrderDALTestUpdateProducts |
| Test Case Description | Update products , return true |
| Precondition |  |
| Test Case Input | Correct information |
| Test Case Expected Output | True |
| Test Case Step | * Create customerDAL * Create orderID and set its value * Create pruduct 1,2 and set customer value * Create ArrayList products and set its value * Create order and set its value * Create orderDetailDAL, result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.15.2 |
| Test Case Name | OrderDALTestUpdateProducts |
| Test Case Description | Update products , return false |
| Precondition |  |
| Test Case Input | OrderID incorrect |
| Test Case Expected Output | False |
| Test Case Step | * Create customerDAL * Create orderID and set its value(orderID incorrect) * Create pruduct 1,2 and set customer value * Create ArrayList products and set its value * Create order and set its value * Create orderDetailDAL, result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.15.3 |
| Test Case Name | OrderDALTestUpdateProducts |
| Test Case Description | Update products , return false |
| Precondition |  |
| Test Case Input | ProductName incorrect |
| Test Case Expected Output | False |
| Test Case Step | * Create customerDAL * Create orderID and set its value * Create pruduct 1,2 and set customer value(product 1:ProductName incorrect) * Create ArrayList products and set its value * Create order and set its value * Create orderDetailDAL, result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.15.4 |
| Test Case Name | OrderDALTestUpdateProducts |
| Test Case Description | Update products , return false |
| Precondition |  |
| Test Case Input | Quantity incorrect |
| Test Case Expected Output | False |
| Test Case Step | * Create customerDAL * Create orderID and set its value * Create pruduct 1,2 and set customer value(Quantity incorrect) * Create ArrayList products and set its value * Create order and set its value * Create orderDetailDAL, result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.15.5 |
| Test Case Name | OrderDALTestUpdateProducts |
| Test Case Description | Update products , return false |
| Precondition |  |
| Test Case Input | Price incorrect |
| Test Case Expected Output | False |
| Test Case Step | * Create customerDAL * Create orderID and set its value * Create pruduct 1,2 and set customer value(Price incorrect) * Create ArrayList products and set its value * Create order and set its value * Create orderDetailDAL, result returned and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.16.1 |
| Test Case Name | ProductBLTestGetByName |
| Test Case Description | Find the product and return true |
| Precondition |  |
| Test Case Input | Information product |
| Test Case Expected Output | True |
| Test Case Step | * Create productBL * Create product and get producName * Create product1 and set its value * Compare them |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.16.2 |
| Test Case Name | ProductBLTestGetByName |
| Test Case Description | Can not find the product and return false |
| Precondition |  |
| Test Case Input | ProductName incorrect |
| Test Case Expected Output | False |
| Test Case Step | * Create productBL * Create product and get producName * Return null |
| Default Value Prevering |  |

|  |  |
| --- | --- |
| Test Case Number | 1.17.1 |
| Test Case Name | ProductDALTestGetByName |
| Test Case Description | Find the product and return true |
| Precondition |  |
| Test Case Input | Information product |
| Test Case Expected Output | True |
| Test Case Step | * Create producDAL * Create product and get producName * Create product1 and set its value * Compare them |
| Default Value Prevering |  |

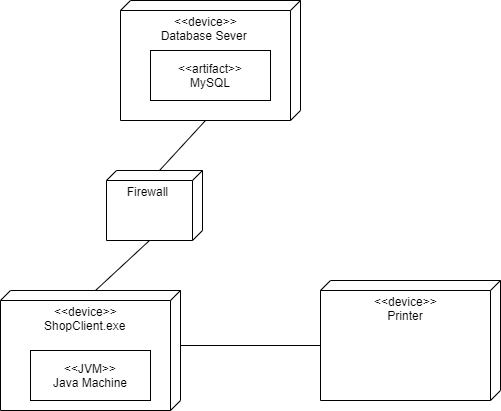
|  |  |
| --- | --- |
| Test Case Number | 1.17.2 |
| Test Case Name | ProductDALTestGetByName |
| Test Case Description | Can not find the product and return false |
| Precondition |  |
| Test Case Input | ProductName incorrect |
| Test Case Expected Output | False |
| Test Case Step | * Create producDAL * Create product and get producName * Return null |
| Default Value Prevering |  |

V. Assign work to each team member

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Group 5 | **Moto Store** | | | | | |
| No | Task name | Description | Start Date | End Date | Member | Self assessment |
| 1. | Draw ERD | Draw ERDSQL,  ERDNoSQL | 3/8/20 | 8/8/20 | Lân | Done |
| 2. | Draw Use Case | 5 use case | 3/8/20 | 8/8/20 | Được | Done |
| 3. | Draw Class Diagram | Design details split layer  Mapping to the database | 3/8/20 | 8/8/20 | Được | Slow |
| 4. | Write Document | Write by Word | 3/8/20 | 8/8/20 | Lân,Được | Finished in the end |
| 5. | Draw Activity Diagrams | 5 activity diagrams | 3/8/20 | 8/8/20 | Lân | Slow |
| 6. | Draw Deployment diagram | Deployment diagram | 3/8/20 | 8/8/20 | Lân | Slow |
| 7. | Draw Sequence Diagrams | 5 sequence diagrams | 3/8/20 | 8/8/20 | Được | Very Slow |
| 8. | Draw UI design | Design display for 5 use case, maybe 1 or more blueprints for each use case | 3/8/20 | 8/8/20 | Lân: Design Login function, Insert Product function, Update Product function. Được: Design Create Order function, Update Order function. | Done |

VI. Installation Instructions

1. Deployment Diagram



1. Installation steps

* Database install
* Server install
* Application install

Appendix