

**PROJECT REPORT**

# Clothing Shop

|  |  |
| --- | --- |
| **Semester:** | Programming Fundamentals |
| **Class:** | PF1121 |
| **Group:** | Group 11 |
| **Instructor** | Nguyen Xuan Sinh |
| **Team members:** | Cu Duy Toan  Mai Thi Hong Minh |

# Index

**Project Name** 1

**Index** 2

**I. Project introduction** 4

**II. Analyze System Requirements** 5

1. use case 6
   1. use case 6
   2. login description 7
   3. view clothes detail description 8
   4. Create order description 9
   5. payment description 10
2. activity diagram 11
   1. login 11
   2. create order 12
   3. payment 13

**III. Design Details** 14

1. UI Design 14
   1. login 14
   2. main menu 14
   3. show all list clothes 15
   4. show list clothes by category 15
   5. create invoice 16
   6. payment 18
2. code design (class diagram) 21
3. sequence diagram 22
   1. login 22
   2. choice clothes 23
   3. create order 24
4. database design 25
   1. entity relationship diagram 25
   2. database design details 26

**IV. Test** 30

1. login test 31
   1. login test 1 31
   2. login test 2 32
   3. login test 3 33
   4. login test 4 34
   5. login test 5 35
   6. login test 6 36
   7. login test 7 36
   8. login test 8 37
2. create order test 38
   1. create order test 1 38
   2. create order test 2 38
   3. create order test 3 39
   4. create order test 4 39
   5. create order test 5 40
3. Payment test 40
   1. payment test 1 40
   2. payment test 2 41
   3. payment test 3 41
   4. payment test 4 42
   5. payment test 5 42
   6. payment test 6 43
   7. payment test 7 43
   8. payment test 8 44
4. new customer test 45
   1. new customer test 1 45
   2. new customer test 2 46

**V. Task Assign (to each team member)** 47

**VI. Installation Instructions** 47

1. development diagram 48
2. installation steps 48
3. application 49
4. link download 49

**Appendix** 49

# Project introduction

This system will provide some basic features including:

There are 2 main features include create order, confirm payment or cancel order which include additional features such as show list clothes, view clothes details, show clothes list by category, show order.

1. Proposed System

* Create a program to Cashier create order
* Create a program to Cashier confirm payment or cancel order

1. The scope of the project to be applied

* Make for actor is Cashier

1. System Name

* Order management system

1. Deployment Environment

* Windows

1. Development Tools

* Visual Studio Code
* Draw.io Diagram
* MySQL Workbench 8.0 CE

1. Customer Requirements

* Cashier login
* Create order
* Show list clothes
* Choice category
* Show list clothes by category
* View order detail
* Update quantity of clothes in order
* Modify clothing information in the order
* Enter customer phone
* Enter payment method
* Confirm order or cancel order

# Analyze System Requirements

This system was created to help clothes shop manage orders more easily. It can help the cashier to have more information about the clothes sold in the shop, that information can include price, size, color, material so that the cashier can give more specific advice to the buyer according to the needs of each person. The main function of this system for the cashier is that it helps create order according to the buyer’s request, allows them to view the information of unpaid orders and confirm successful payment if enough money has been received from the buyer, cancel the order when the buyer change their mind.

1. Use Case
   1. Use case

A black screen with white circles

Description automatically generated

1.2 . Login description

|  |  |
| --- | --- |
| Use Case Name | Login |
| Use Case ID | UC\_001 |
| Description | Used to access program |
| Actor | Cashier |
| Organizational Benefits | This feature was created to determine if a user is a shop cashier |
| Triggers | Login will be automatically activated as soon as we start the program |
| Preconditions | Must enter the correct username and password, the database must be connected, accounts must be made available |
| Postconditions | Success: Cashier have logged into the system, the main menu of the program must be displayed  Fail: Cashier haven’t logged into the system |
| Main Course | 1. Input username and password  2. Verify login  3. Show main menu |
| Alternate Courses | AC 1: User enters wrong username, password  1. Display message error  2. Re-enter request  AC 2: Username and password exists  1. Show main menu |
| Exceptions | EX 1: Input Username/ Password  1. Display “Username or password incorrect, please re-enter” and ask to re-enter until correct  2. Request to re-enter username and password  EX 2: Disconnect to database  1. Step 1: Display “Error! Can’t connect to database”  2. Request to recheck internet connection |

1.3. View clothes details description

|  |  |
| --- | --- |
| Use Case Name | View clothes details |
| Use Case ID | UC\_002 |
| Description | Used to view clothes details |
| Actor | Cashier |
| Organizational Benefits | You can view all clothes infomation |
| Triggers | Login to the program by the cashier’s account and you can use this feature |
| Preconditions | Must have at least one item previously created |
| Postconditions | Success: Show all item information  Fail: Not have item information in database |
| Main Course | 1. Choose item you want to view item information  2. Show item information |
| Alternate Courses | AC: Not have item information in database  1. Display message “Not have information”  2. Requires pressing Tab key to go back |
| Exceptions | EX: Disconnect to database  1. Request to recheck internet connection  2. Request program restart |

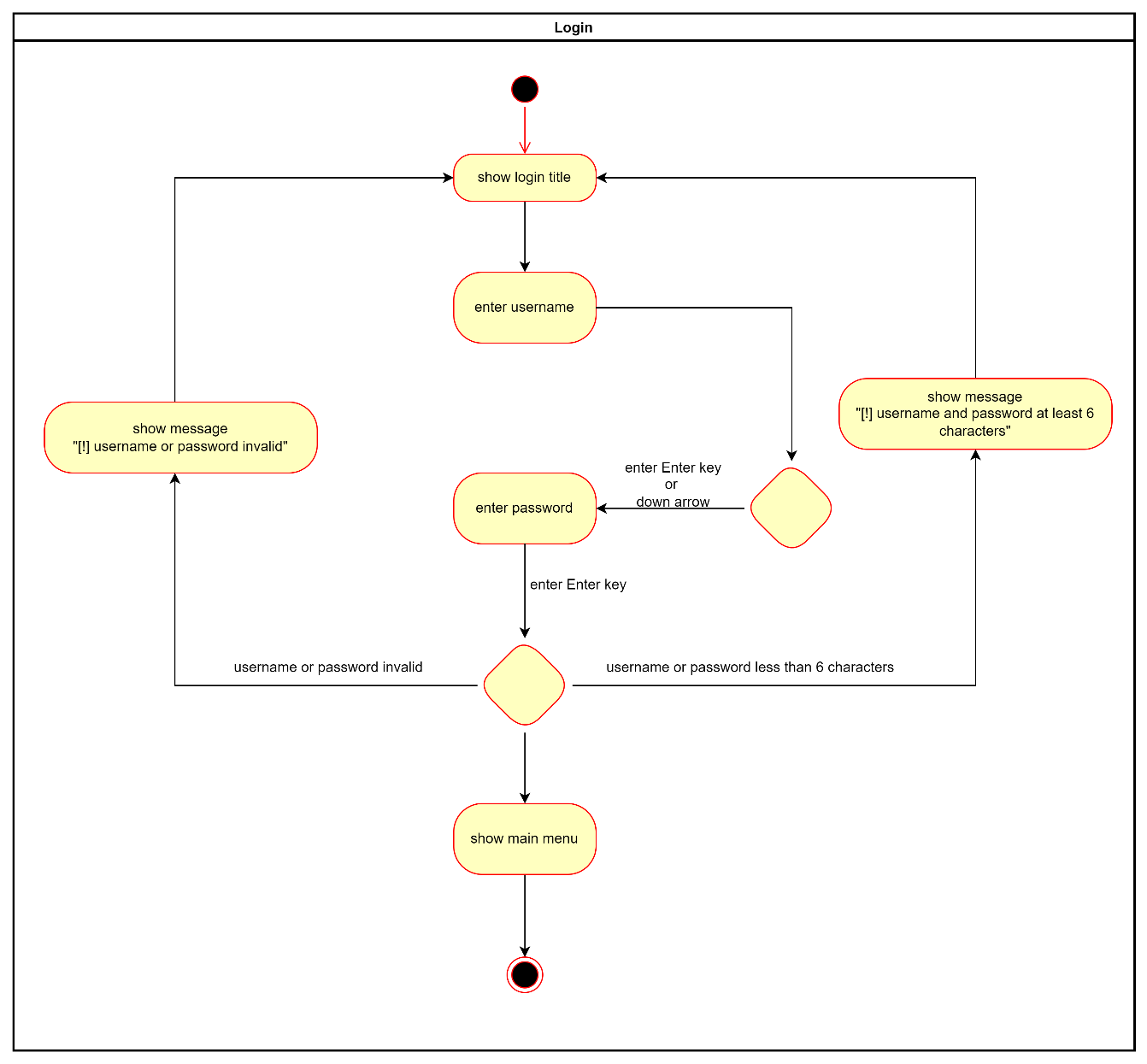
1.4. Create order description

|  |  |
| --- | --- |
| Use Case Name | Create order |
| Use Case ID | UC\_003 |
| Description | Used to create order |
| Actor | Cashier |
| Organizational Benefits | You can create one or more orders according to customer’s request |
| Triggers | Login to the program by the cashier’s account and you can use this feature |
| Preconditions | Must have at least one item previously created |
| Postconditions | Success: Create order complete  Fail: Create order not complete |
| Main Course | 1. Input customer information  2. Choose item and input quantity you want to add to order  3. Create order complete |
| Alternate Courses | AC: User input phone number by letters  1. Display message “Only input number”  2. Re-enter request |
| Exceptions | EX: Disconnect to database  1. Request to recheck internet connection  2. Request program restart |

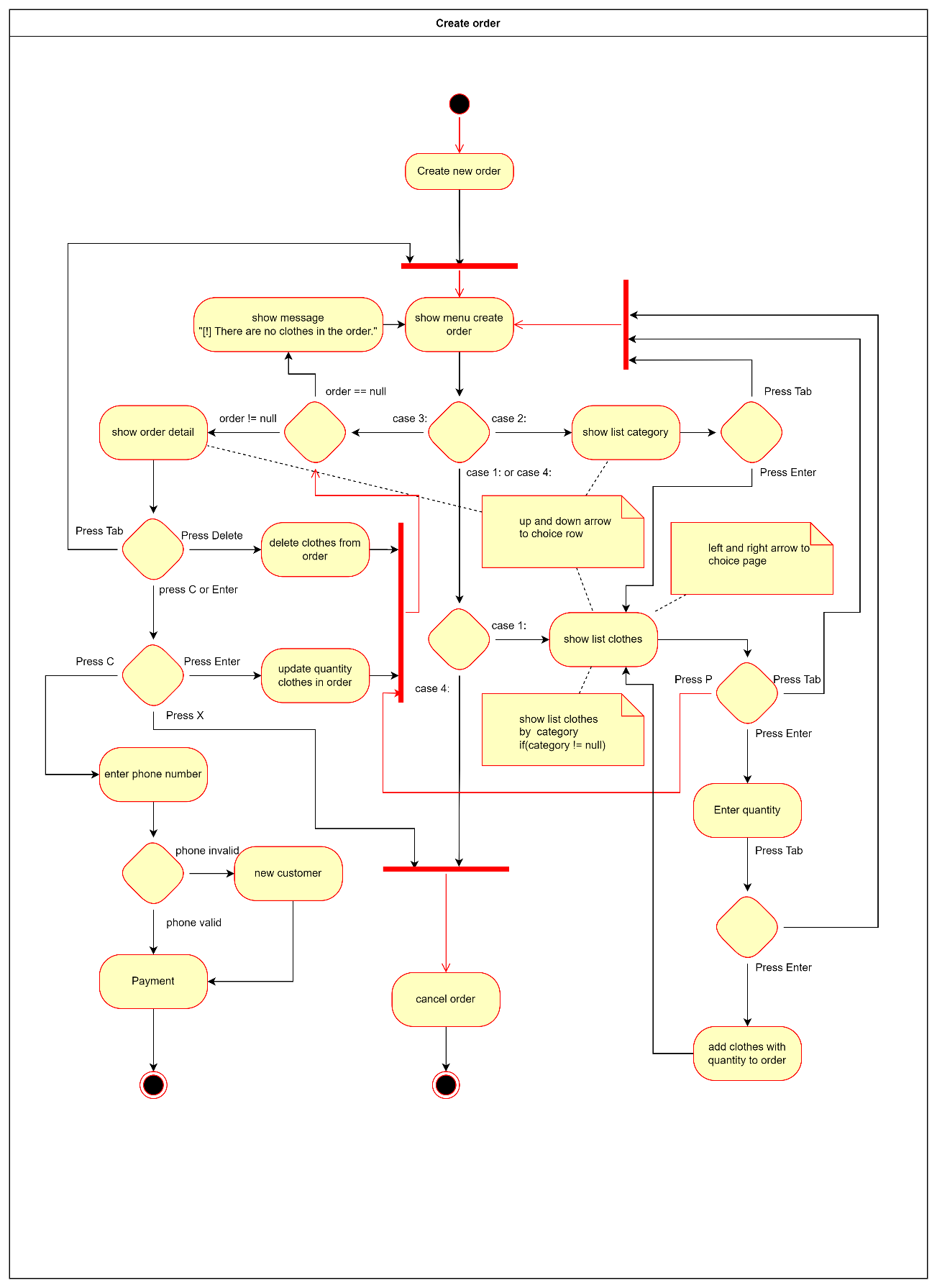
1.5. Payment description

|  |  |
| --- | --- |
| Use Case Name | Payment |
| Use Case ID | UC\_004 |
| Description | Used to confirm payment or cancel payment |
| Actor | Cashier |
| Organizational Benefits | Allow cashier to confirm paid orders or cancel orders |
| Triggers | Login to the program by the cashier’s account and you can use this feature |
| Preconditions | Must have at least one order unpaid |
| Postconditions | Success: Confirm payment, cancel order complete or payment pause  Fail: Confirm payment or cancel order not complete |
| Main Course | 1. Press “Show order”  2. Choose to payment, cancel order or exit |
| Alternate Courses | AC 1: Confirm payment  1. Press “C” to confirm payment  2. Export invoice  AC 2: Cancel order  1. Press “X” to cancel order  2. Display msg “Cancel order complete!”  AC 3: Back  1. Press Tab to go back  2. Payment pause |
| Exceptions | EX: Disconnect to database  1. Request to recheck internet connection  2. Request program restart |

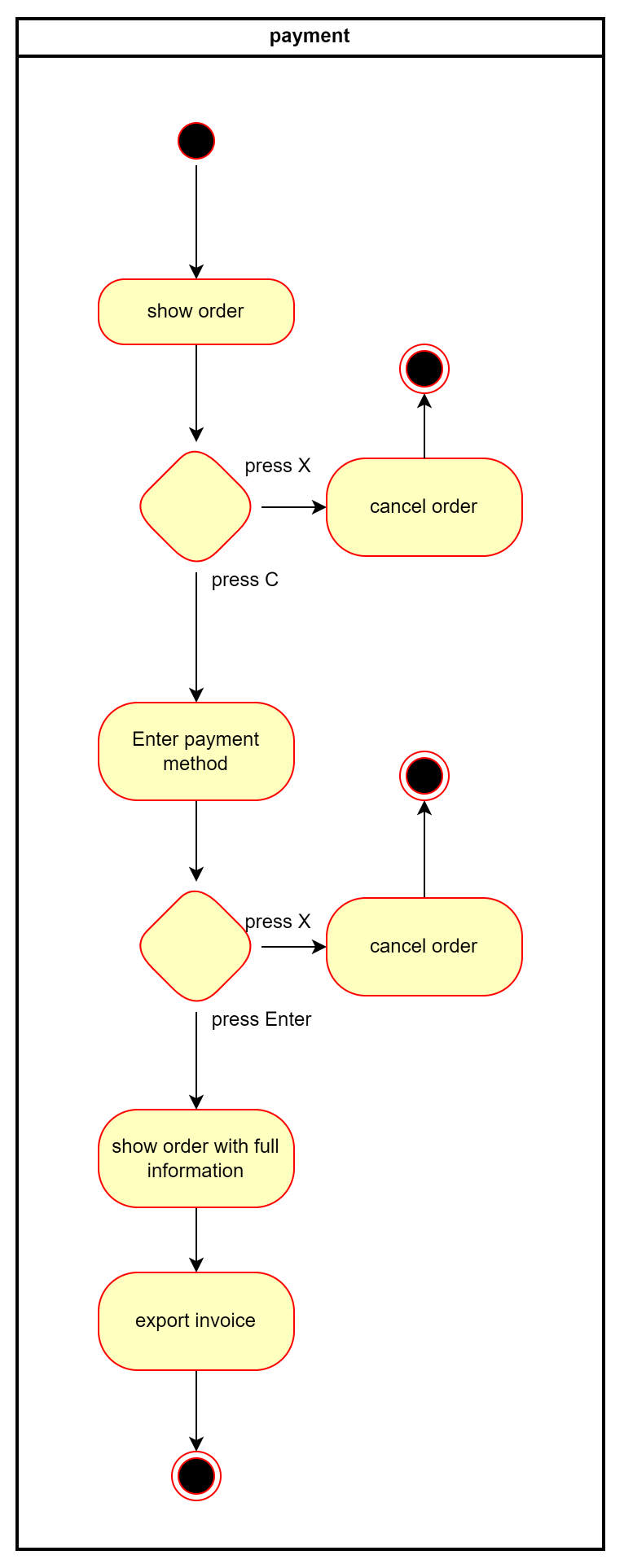
1. Activity Diagram:

2.1. Login

2.2. Create order



2.3. Payment



# Design Details

1. UI Design

1.1. Login

A screen shot of a computer

Description automatically generated

1.2. Main menu

A screen shot of a computer

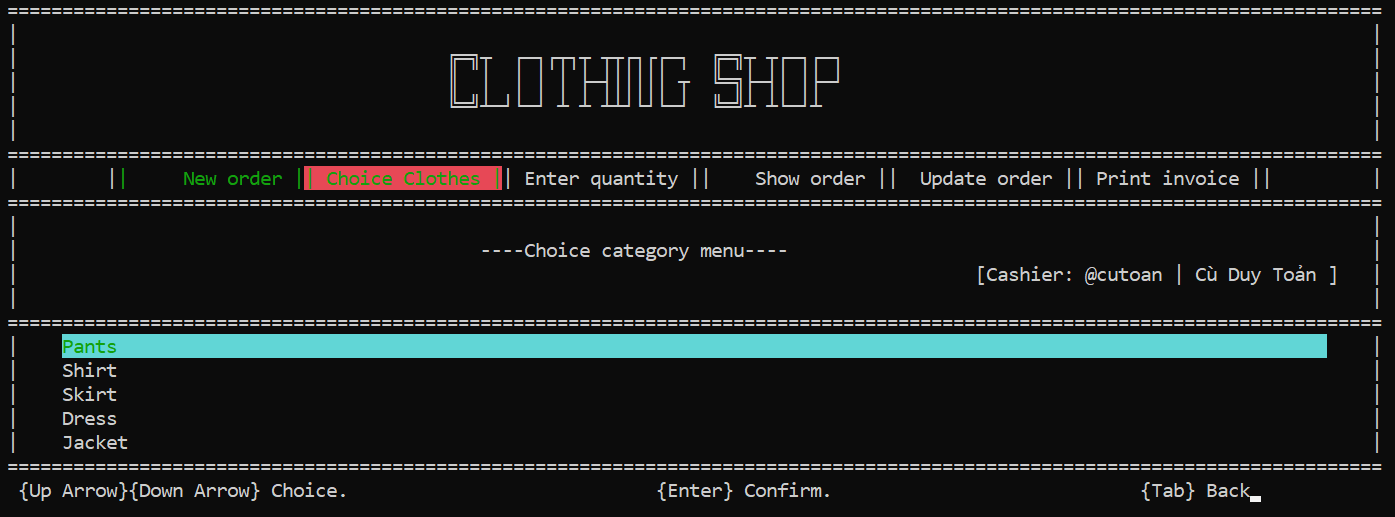
Description automatically generated

* 1. Show all list item

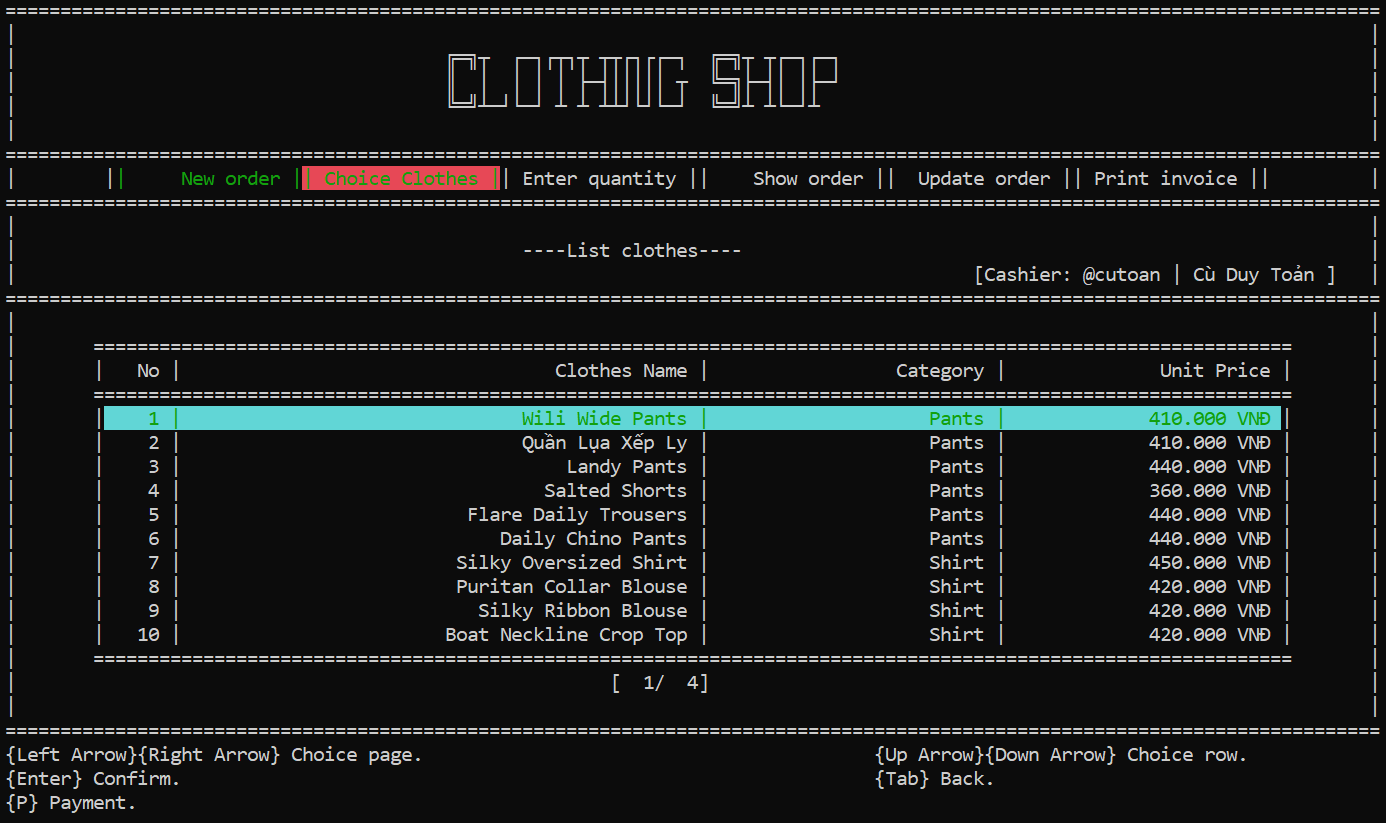
A screen shot of a computer

Description automatically generated

1.4. Show list item by category



1.5. Create order







1.6. Payment

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated



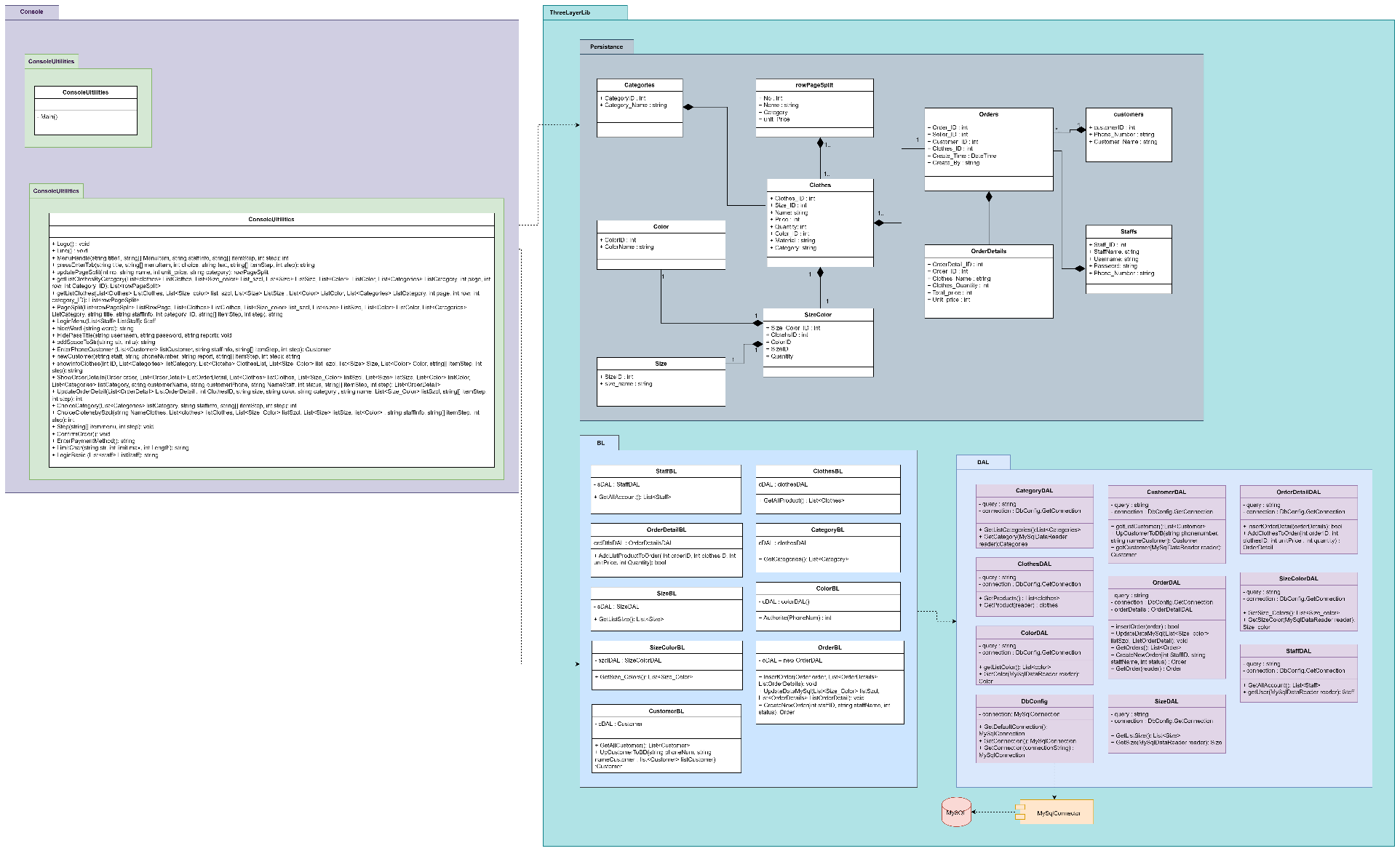




A screen shot of a computer screen

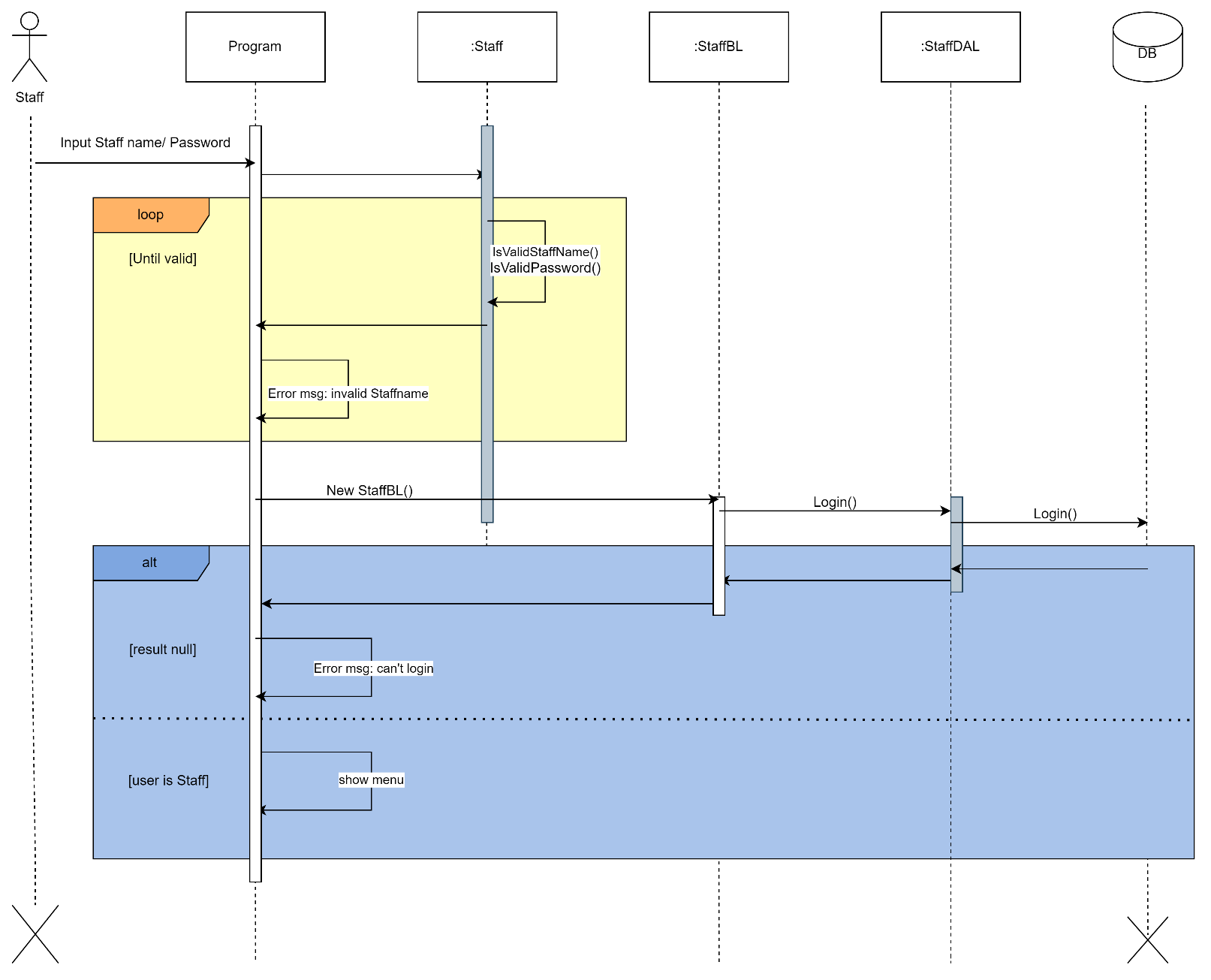
Description automatically generated

1. Code Design (Class Diagram)

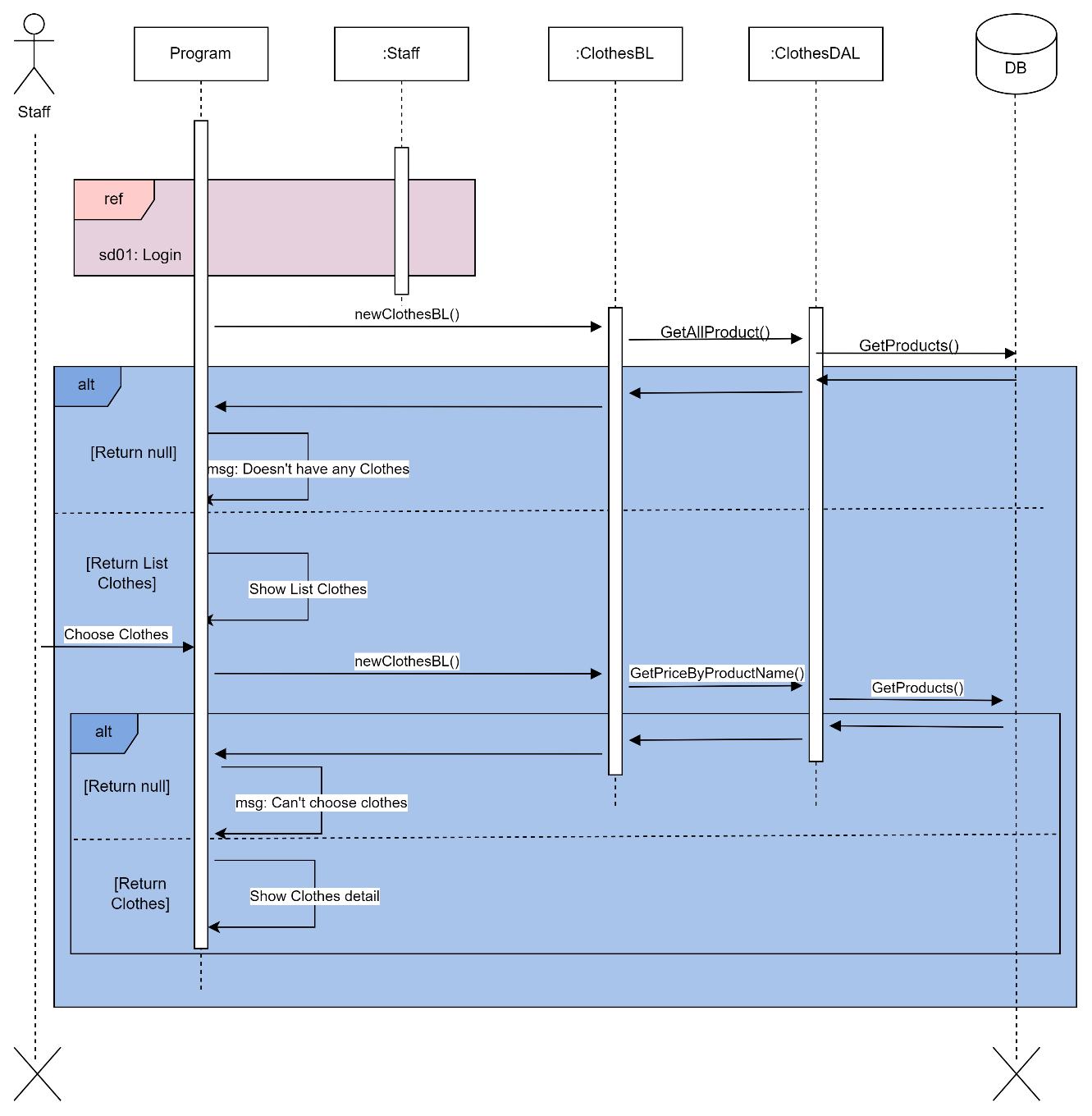


1. Sequence Diagram

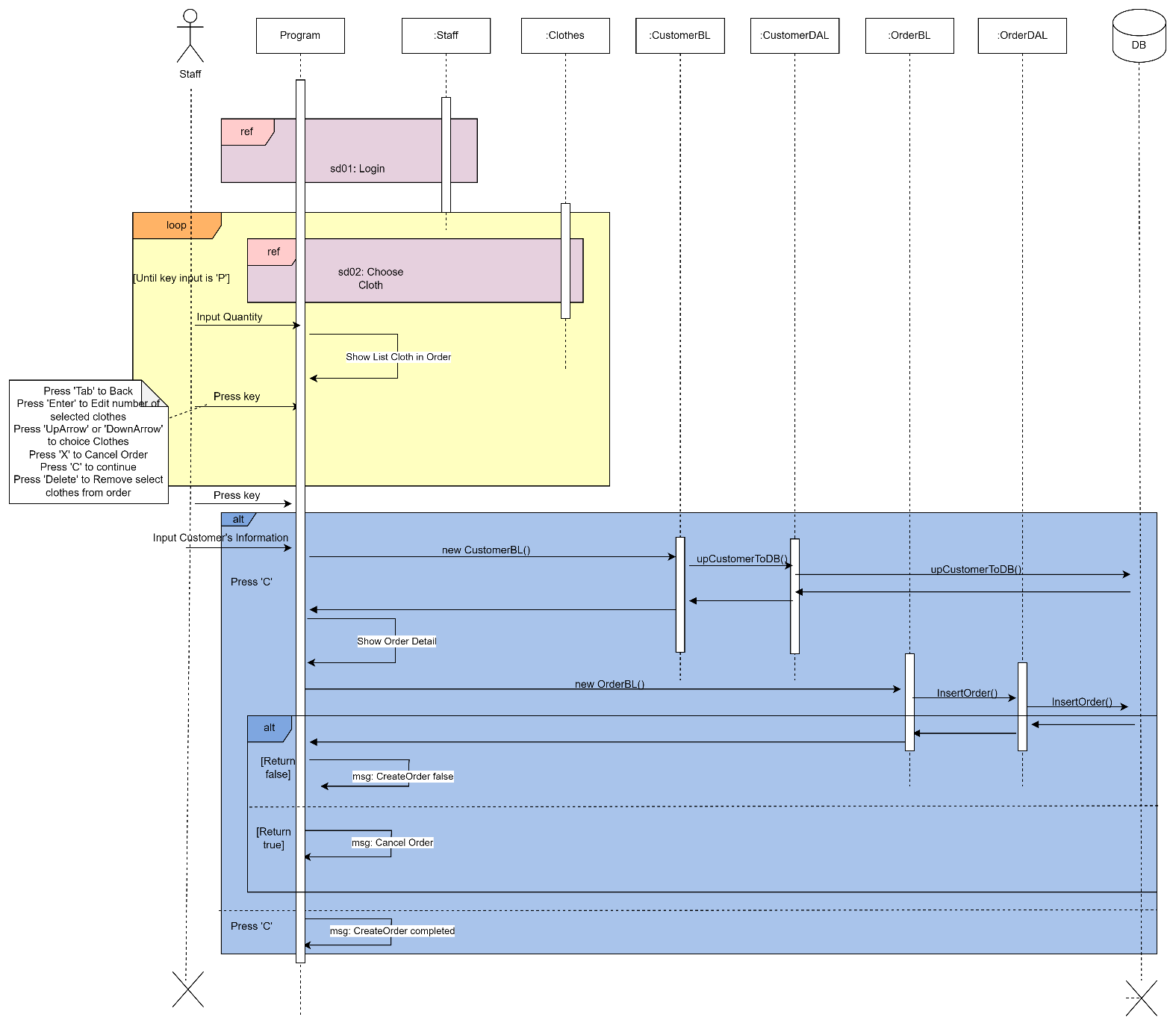
3.1. Login



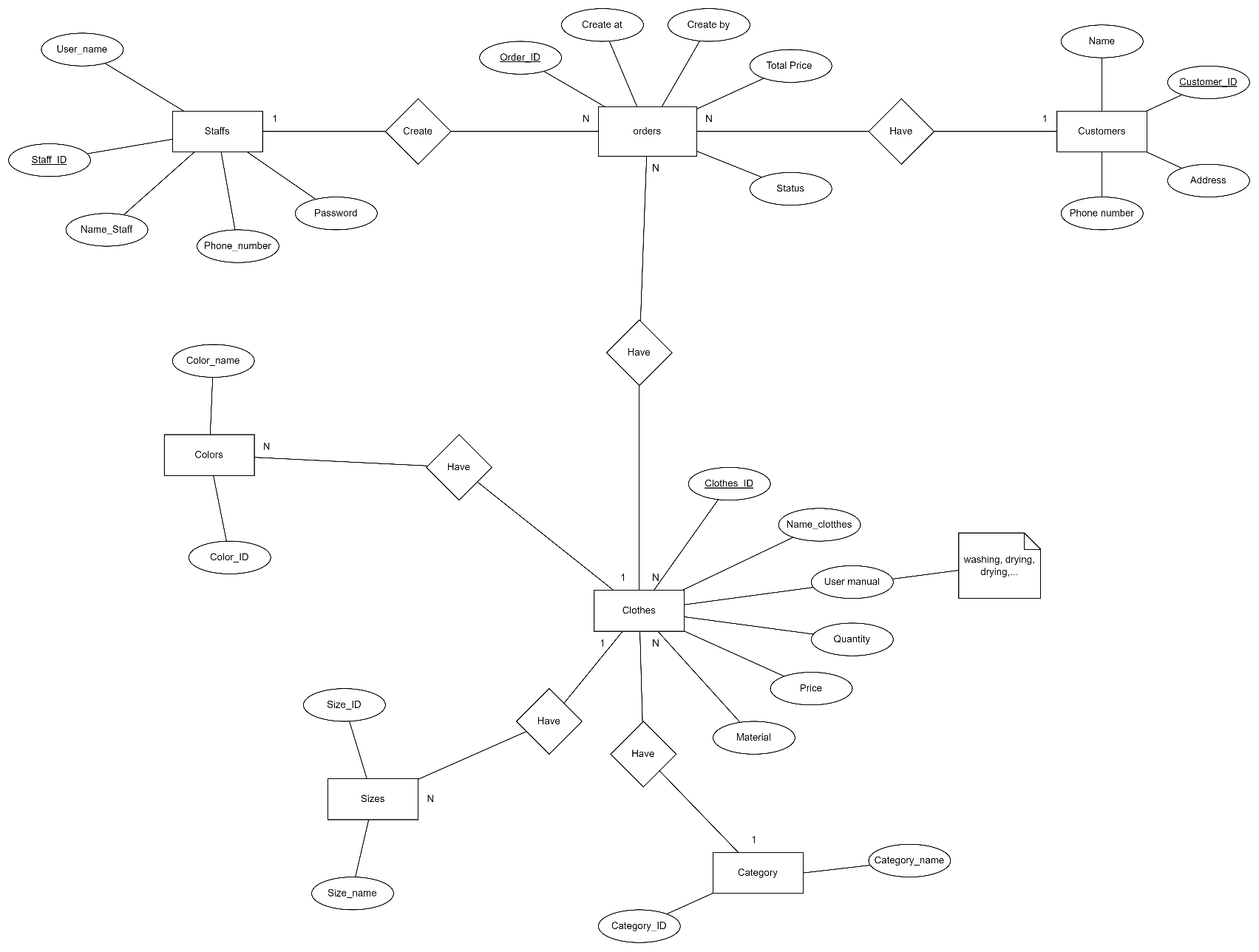
3.2. Choice clothes



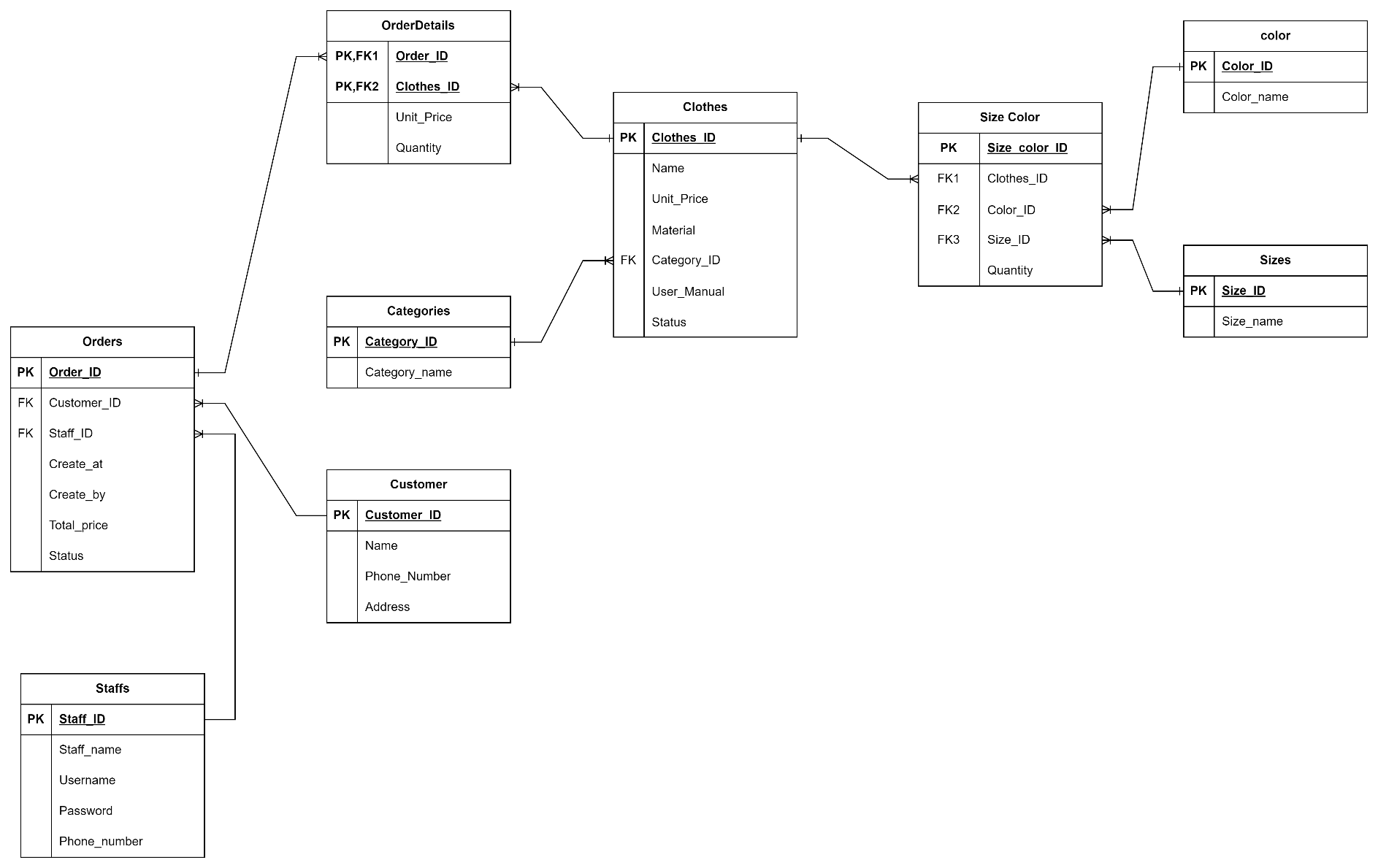
3.3. Create order



1. Database Design
   1. Entity Relationship Diagram



* 1. Database Design Details



|  |  |  |  |
| --- | --- | --- | --- |
| Staffs | | | |
| Column Name | Data Type | Constraints | Description |
| Staff\_ID | int | PRIMARY KEY, AUTOINCREMENT |  |
| Username | varchar(50) | NOT NULL |  |
| Password | varchar(50) | NOT NULL |  |
| Staff\_name | varchar(50) | NOT NULL | Staff’s name |

|  |  |  |  |
| --- | --- | --- | --- |
| Customers | | | |
| Column Name | Data Type | Constraints | Description |
| Customer\_ID | int | PRIMARY KEY, AUTOINCREMENT |  |
| Name | varchar(50) | NOT NULL | Customer’s name |
| Address | varchar(100) |  | Customer’s address |
| Phone number | varchar(10) | NOT NULL, UNIQUE | Customer’s phone number |

|  |  |  |  |
| --- | --- | --- | --- |
| Categories | | | |
| Column Name | Data Type | Constraints | Description |
| Categories\_ID | int | PRIMARY KEY, AUTOINCREMENT |  |
| Categories\_name | varchar(50) | NOT NULL | Clothes’ category |

|  |  |  |  |
| --- | --- | --- | --- |
| Orders | | | |
| Column Name | Data Type | Constraints | Description |
| Order\_ID | int | PRIMARY KEY, AUTOINCREMENT |  |
| Staff\_ID | int | NOT NULL |  |
| Customer\_ID | int | NOT NULL |  |
| Create\_at | datetime | NOT NULL, DEFAULT CURRENT DATE | Order creation date |
| Create\_by | Varchar(50) | NOT NULL | Staff’s name |
| Total\_price | decimal | NOT NULL | Invoice’s unit price |
| Status | int | NOT NULL | 1: Unpaid  2: Paid  3: Cancel |

|  |  |  |  |
| --- | --- | --- | --- |
| Orders\_details | | | |
| Column Name | Data Type | Constraints | Description |
| Order\_ID | int | PRIMARY KEY, FOREIGN KEY |  |
| Clothes\_ID | int | PRIMARY KEY, FOREIGN KEY |  |
| Total\_price | decimal | NOT NULL | Clothes’s unit price |
| Quantity | int | NOT NULL | Clothes’s quantity |

|  |  |  |  |
| --- | --- | --- | --- |
| Size color | | | |
| Column Name | Data Type | Constraints | Description |
| Size\_color\_ID | int | PRIMARY KEY, AUTOINCREMENT |  |
| Clothes\_ID | int | FOREIGN KEY |  |
| Size\_ID | int | FOREIGN KEY |  |
| Color\_ID | int | FOREIGN KEY |  |
| Quantity | int | NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Color | | | |
| Column Name | Data Type | Constraints | Description |
| Color\_ID | int | PRIMARY KEY, AUTOINCREMENT |  |
| Color\_name | Varchar(50) | NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Sizes | | | |
| Column Name | Data Type | Constraints | Description |
| Size\_ID | int | PRIMARY KEY, AUTOINCREMENT |  |
| Size\_name | Varchar(50) | NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Clothes | | | |
| Column Name | Data Type | Constraints | Description |
| Clothes\_ID | int | PRIMARY KEY, AUTOINCREMENT |  |
| Clothes\_name | Varchar(50) | NOT NULL | Clothes’s name |
| Unit\_price | decimal | NOT NULL, DEFAULT 0 | Clothes’s price |
| Material | varchar(50) | NOT NULL | Clothes’s material |
| Category\_ID | int | FOREIGN KEY | Clothes’s category |
| User\_manual | Varchar(50) |  |  |
| Quantity | int | NOT NULL, DEFAULT 0 | Clothes’s quantity |

# Test

1. Login test

1.1. Login test 1

|  |  |
| --- | --- |
| Test Case Number | 001 |
| Test Case Name | LoginTest() |
| Test Case Description | This test case checks username and password correct or not |
| Preconditions | Username and password must exists in database |
| Test Case Input | “cutoan”, “toan2004”  “hongminh”, “minh2004”  “admin”, “admin@clothesShop” |
| Test Case Expected Output | True |
| Test Case Steps | New Staff  New StaffDAL  Call Login method  Compare output with expected output |
| Default Value Preverving |  |

1.2. Login test 2

|  |  |
| --- | --- |
| Test Case Number | 002 |
| Test Case Name | LoginTest () |
| Test Case Description | This test case checks username and password correct or not |
| Preconditions | Username and password not necessarily exists in database |
| Test Case Input | “18tamtrinhvtc”, “1234567890”  “vietnamfc”, “hoangsatruongsa”  “abcdef”, “123456” |
| Test Case Expected Output | Display msg “[!]{username} or {password} incorrect.” |
| Test Case Steps | New Staff  New StaffDAL  Call Login method  Compare output with expected output |
| Default Value Preverving |  |

1.3. Login test 3

|  |  |
| --- | --- |
| Test Case Number | 003 |
| Test Case Name | LoginTest () |
| Test Case Description | This test case checks if entering too many characters affects the predefined texture of the app. |
| Preconditions | Username and password not necessarily exists in database |
| Test Case Input | Username : 18tamtrinhvtca0123456789  Password : vtcastudent0123456789 |
| Test Case Expected Output | User name : [vtca0123456789]  Password : [\*\*\*\*\*\*\*\*\*\*\*\*\*] |
| Test Case Steps | New Staff  New StaffDAL  Call Login method  Compare output with expected output |
| Default Value Preverving |  |

1.4. Login test 4

|  |  |
| --- | --- |
| Test Case Number | 004 |
| Test Case Name | LoginTest () |
| Test Case Description | The test case check if in case username or password input less than 6 characters is accepted |
| Preconditions | username or pass word must be numbers or letters and less than 6 character |
| Test Case Input | Username : abcde  Password : 123 |
| Test Case Expected Output | Display msg: “[!] username and password at least 6 characters.” |
| Test Case Steps | New Staff  New StaffDAL  Call Login method  Compare output with expected output |
| Default Value Preverving |  |

1.5. Login test 5

|  |  |
| --- | --- |
| Test Case Number | 005 |
| Test Case Name | LoginTest () |
| Test Case Description | The test case checks if special characters like !@#$... are accepted |
| Preconditions | username and pass word not necessarily exists in database |
| Test Case Input | Username : !@#$%^&\*()-=  Password : {}[]{;’\/.,\_+ |
| Test Case Expected Output | Username : [!@#$%^&\*()-=]  Password: [\*\*\*\*\*\*\*\*\*\*\*\*\*] |
| Test Case Steps | New Staff  New StaffDAL  Call Login method  Compare output with expected output |
| Default Value Preverving |  |

1.6. Login test 6

|  |  |
| --- | --- |
| Test Case Number | 006 |
| Test Case Name | LoginTest () |
| Test Case Description | The test case check if the spacebar is pressed many times when the number of characters of the username or password is 0 there is an error |
| Preconditions | username and password without characters |
| Test Case Input |  |
| Test Case Expected Output | False |
| Test Case Steps | New Staff  New StaffDAL  Call Login method  Compare output with expected output |
| Default Value Preverving |  |

1.7. Login test 7

|  |  |
| --- | --- |
| Test Case Number | 007 |
| Test Case Name | LoginTest () |
| Test Case Description | Username is correct but password is incorrect. Can I log in? |
| Preconditions | pass word not necessarily exists in database |
| Test Case Input |  |
| Test Case Expected Output | False |
| Test Case Steps | New Staff  New StaffDAL  Call Login method  Compare output with expected output |
| Default Value Preverving |  |

1.8. Login test 8

|  |  |
| --- | --- |
| Test Case Number | 008 |
| Test Case Name | LoginTest () |
| Test Case Description | test case check when changing line to enter during login phase is it possible |
| Preconditions | username and pass word not necessarily exists in database |
| Test Case Input | Enter “abc”  Press down arrow  Enter “123”  Press up arrow  Enter “d” |
| Test Case Expected Output | True  Username : [abcd ]  Password : [\*\*\* ] |
| Test Case Steps | New Staff  New StaffDAL  Call Login method  Compare output with expected output |
| Default Value Preverving |  |

2. Create order test

2.1. Create order test 1

|  |  |
| --- | --- |
| Test Case Number | 001 |
| Test Case Name | CreateOrderTest1() |
| Test Case Description | The test case checks what happens when too many clothes are entered in the database |
| Preconditions | Logged in |
| Test Case Input | Quantity : 999 |
| Test Case Expected Output | Show msg “[!] The order quantity exceeds the quantity of the clothes.” |
| Test Case Steps | Choose item and input quantity, |
| Default Value Preverving |  |

2.2. Create order test 2

|  |  |
| --- | --- |
| Test Case Number | 001 |
| Test Case Name | CreateOrderTest1() |
| Test Case Description | test case check what happens when too many products are imported |
| Preconditions | Logged in |
| Test Case Input | Quantity : 9999999 |
| Test Case Expected Output | Show msg “  [!] The order quantity exceeds the quantity of the clothes.  [!] Max value!” |
| Test Case Steps | Choose item and input quantity |
| Default Value Preverving |  |

2.3. Create order test 3

|  |  |
| --- | --- |
| Test Case Number | 003 |
| Test Case Name | CreateOrderTest3() |
| Test Case Description | The test case checks what happens when a non-numeric character is pressed at the quantity input stage |
| Preconditions | Logged in |
| Test Case Input | Quantity : abcd |
| Test Case Expected Output | Display msg “[!] Only enter number” |
| Test Case Steps | Choose item and input quantity “2”, choose Show order and press “C” to confirm order  Input customer’s phone number by letters |
| Default Value Preverving |  |

2.4. Create order test 4

|  |  |
| --- | --- |
| Test Case Number | 004 |
| Test Case Name | CreateOrderTest4() |
| Test Case Description | The test case check if it is possible to get list of clothes from category |
| Preconditions | Logged in |
| Test Case Input |  |
| Test Case Expected Output | True |
| Test Case Steps | Enter category  Call method showlistclothes() |
| Default Value Preverving |  |

2.4. Create order test 5

|  |  |
| --- | --- |
| Test Case Number | 005 |
| Test Case Name | CreateOrderTest5() |
| Test Case Description | The test case check that in case of out of stock (quantity = 0) but still want to include in the order, what will happen |
| Preconditions | Logged in |
| Test Case Input | “1” “2” “3” “Enter” “123456” |
| Test Case Expected Output | Show message “[!] Out of stock.” |
| Test Case Steps | Call method showInfoClothes() |
| Default Value Preverving |  |

3. Payment test

3.1. Payment test 1

|  |  |
| --- | --- |
| Test Case Number | 001 |
| Test Case Name | PaymentTest() |
| Test Case Description | The test case checks what happens when there are no items in order yet |
| Preconditions | No items in order yet |
| Test Case Input |  |
| Test Case Expected Output | Display msg “[!] There are no clothes in the order.” |
| Test Case Steps | New Order  Call function showOrderDetail |
| Default Value Preverving |  |

3.2. Payment test 2

|  |  |
| --- | --- |
| Test Case Number | 002 |
| Test Case Name | PaymentTest() |
| Test Case Description | The test case checks what happens when you enter more than the maximum number of clothes |
| Preconditions |  |
| Test Case Input | “999”, “200”, “300” |
| Test Case Expected Output | Display msg “[!] The order quantity exceeds the quantity of the clothes..” |
| Test Case Steps | Call function UpdateOrderDetail  Enter quantity |
| Default Value Preverving |  |

3.3. Payment test 3

|  |  |
| --- | --- |
| Test Case Number | 003 |
| Test Case Name | PaymentTest() |
| Test Case Description | The test case checks what happens when you enter more than the maximum number of clothes |
| Preconditions |  |
| Test Case Input | “9999”, “2000”, “3000”, “1000” |
| Test Case Expected Output | Display msg “  [!] Max value.  [!] The order quantity exceeds the quantity of the clothes.” |
| Test Case Steps | Call function UpdateOrderDetail  Enter quantity |
| Default Value Preverving |  |

3.4. Payment test 4

|  |  |
| --- | --- |
| Test Case Number | 004 |
| Test Case Name | PaymentTest() |
| Test Case Description | The test case checks if it is possible to select clothes with a quantity that exceeds the maximum number of clothes |
| Preconditions | Enter the quantity |
| Test Case Input | “9999”, “2000”, “3000”, “1000” |
| Test Case Expected Output | False |
| Test Case Steps | Call function UpdateOrderDetail  Enter quantity |
| Default Value Preverving |  |

3.5. Payment test 5

|  |  |
| --- | --- |
| Test Case Number | 004 |
| Test Case Name | PaymentTest() |
| Test Case Description | check if it is possible to enter a string shorter than 10 characters |
| Preconditions | Enter phone customer |
| Test Case Input | “12345” |
| Test Case Expected Output | False |
| Test Case Steps | Call function enterPhoneCustomer  Enter phone number |
| Default Value Preverving |  |

3.5. Payment test 5

|  |  |
| --- | --- |
| Test Case Number | 005 |
| Test Case Name | PaymentTest() |
| Test Case Description | check if a string longer than 10 characters can be entered |
| Preconditions | Enter phone customer |
| Test Case Input | “01234567890” |
| Test Case Expected Output | Show msg: “[?] Are you sure you want to use the phone number [0123456789]?” |
| Test Case Steps | Call function enterPhoneCustomer  Enter phone number |
| Default Value Preverving |  |

3.6. Payment test 6

|  |  |
| --- | --- |
| Test Case Number | 006 |
| Test Case Name | PaymentTest() |
| Test Case Description | check if it is possible to enter a string of characters other than numbers in the phone customer |
| Preconditions | Enter phone customer |
| Test Case Input | “abcdefghik”, “@!#$%^&\*()” |
| Test Case Expected Output | False |
| Test Case Steps | Call function enterPhoneCustomer  Enter characters |
| Default Value Preverving |  |

3.7. Payment test 7

|  |  |
| --- | --- |
| Test Case Number | 007 |
| Test Case Name | PaymentTest() |
| Test Case Description | Check what happens in case the customer is not in the database |
| Preconditions | The customer's phone number is not in the database |
| Test Case Input | “0349808890” “0378324503” |
| Test Case Expected Output | Show msg : “[!]Phone number not found  Add new customers?  [Enter] Confirm [Tab] Return” |
| Test Case Steps | Call function enterPhoneCustomer  Enter phone number |
| Default Value Preverving |  |

3.8. Payment test 8

|  |  |
| --- | --- |
| Test Case Number | 008 |
| Test Case Name | PaymentTest() |
| Test Case Description | check if entering too many characters during the Enter payment method breaks the display of the program |
| Preconditions | Enter payment method |
| Test Case Input | Payment method : qwertyuiopasdfghjklzxcvbnm |
| Test Case Expected Output | Payment method : [uiopasdfghjklzxcvbnm ] |
| Test Case Steps | Call function enterPayMentMethod  Enter word |
| Default Value Preverving |  |

1. New customer test
   1. New customer test 1

|  |  |
| --- | --- |
| Test Case Number | 001 |
| Test Case Name | NewCustomerTest() |
| Test Case Description | Check what happens in case the customer is not in the database |
| Preconditions | The customer's phone number is not in the database |
| Test Case Input | “0349808890” “0378324503” |
| Test Case Expected Output | Show msg : “[!]Phone number not found  Add new customers?  [Enter] Confirm [Tab] Return” |
| Test Case Steps | Call function enterPhoneCustomer  Call function newCustomer  Enter phone number  Press enter if there is no data of the phone number in the database |
| Default Value Preverving |  |

* 1. New customer test 2

|  |  |
| --- | --- |
| Test Case Number | 002 |
| Test Case Name | NewCustomerTest() |
| Test Case Description | Check what happens in case the customer is not in the database |
| Preconditions | The customer's phone number is not in the database |
| Test Case Input | Enter name customer “newcustomer 1” |
| Test Case Expected Output | True |
| Test Case Steps | Call function newCustomer  Enter phone name customer |
| Default Value Preverving |  |

# Task Assign (to each team member)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Group Name | Project Name | | | | | |
| No | Task name | Description | Start Date | End Date | Member | Self assessment |
| 1 | Draw diagram |  | 9 Jun | 10 Aug | Toan & Minh |  |
| 2 | Code |  | 10 Jul | 28 Aug | Toan |  |
| 3 | Write report |  | 1 Aug | 10 Aug | Minh |  |
| 4 | Write report |  | 12 Aug | 31 Aug | Toan & Minh |  |
| 5 | Draw diagram |  | 12 Aug | 31 Aug | Toan & Minh |  |
| 6 | Test |  | 20 Jul | 31 Aug | Minh |  |

# Installation Instructions

* Deployment Diagram

A diagram of a server

Description automatically generated

* Installation steps
  + Download MySQL workbench
  + Download console app
  + Unzip files
  + Install MySQL workbench
  + Open MySQL workbench
  + Open file CreateDataMySQL in folder MySQL of console app
  + Run file sql
  + Open app console
  + Login with default account or create new account in MySQL
* Applications

MySQL : used for editing data in a console application's database and also as a data storage engine for a console application. is a necessary tool for the application dashboard to function

* Link download

MySQL : <https://dev.mysql.com/downloads/workbench/>

Console app : <https://drive.google.com/file/d/1szl9YqAelvg_3oND3CREgHw88W9ncb-T/view?usp=drive_link>

# Appendix

Terms and abbreviations

1. Abbreviations :

* msg or Msg : message

References

1. Reference 1: <https://github.com/sinhnx-dev/cshap-examples/tree/main/three-layer-model/ThreeLayerWithMySql>
2. Reference 2: <https://howkteam.vn/course/lap-trinh-phan-mem-quan-ly-quan-cafe-voi-c-winform-24>

Some other issues

1. Limitations :

* Don't know how to take advantage of the convenience of the library.
* Programming thinking still has many shortcomings.
* Diagram drawing skills (use case, activity, sequence diagram, ...) are not yet proficient.
* Teamwork skills are not good.

1. Experience:

* Gain more experience in writing a few simple programs.
* Learn some interesting points about programming.
* Understand the value of organizing and dividing work.
* Know the need to separate functions and clean up the code.