

**PROJECT REPORT**

# Milk Tea Mini Shop Management

|  |  |
| --- | --- |
| **Semester:** | Programming Fundamentals |
| **Class:** | PF08 |
| **Group:** | Group 2 |
| **Instructor** | Nguyễn Xuân Sinh |
| **Team members:** | NDE 19055 – Nông Hồng Cương  NDE 19024 – Nguyễn Long Nhật |

Ha Noi, September 2020

# Index

Milk Tea Mini Shop Management

Index 2

I. Project introduction 3

II. Analyze System Requirements 4

III. Design Details 17

IV. Test 25

V. Assign work to each team member 31

VI. Installation Instructions 33

Appendix 35

# Project introduction

User login to system and performs the functions such as: add, update drink and order. Finally, the user prints the invoice for the customer

1. Proposed System

Help user manage our milk tea shop more conveniently.

1. The scope of the project to be applied

Apply in many milk tea mini shops.

1. System Name

Milk Tea Mini Shop Management

1. Deployment Environmet

* Hardware: Computers: PCs, Laptops; Servers.
* Software:

+ Computers: Window, MacOS, Linux,...

+Servers: MySQL Server,...

1. Development Tools

- Visual Studio Code

- IntelliJ IDEA

- Apache NetBeans IDE 12.0

- MySQL Workbench 8.0 CE

1. Customer Requirements

Build an app with some feautures such as: Login, add and update drink; create, update orders and print out the invoice.

# Analyze System Requirements

1. Use Case
   1. Use case 1: Login

|  |  |
| --- | --- |
| Use Case Name | Login |
| Use Case ID | UC01\_Login |
| Description | Staff/Manager must log in to the system. |
| Actor | -Staff  -Manager |
| Organizational Benefits | Authenticate user |
| Frequency of Use | Often |
| Triggers | Staff/Manager want to use the system. |
| Preconditions | Staff/manager must have account and password to log in. |
| Postconditions | Show features of the staff/manager. |
| Main Course | 1. User input account and password. 2. Valid account/password.(see AC1) 3. Show features of user:    1. If user is Manager, display Manager’Menu .   3.2 If user in Staff, display Staff’s Menu. |
| Alternate Courses |  |
| Exceptions | EX1. Valid Username:   * 1. Maximum 20 characters.   2. Minimun 8 characters.   3. Must contain letter, at least 1 number and 1 upper letter.   EX2. Valid Password  2.1. Minimum password length 8 characters.  2.2. Must contain letter, at least 1 number and 1 upper letter.  EX3. Incorrect password  3.1. Return user to Main Course Step 1  EX4. Account is not existed  4.1.Return user to Main Course Step1 |

1.2. Use case 2: Insert Drink

|  |  |
| --- | --- |
| Use Case Name | Insert Drink |
| Use Case ID | UC02\_Insert\_Drink |
| Description | Add new drink to database. |
| Actor | Manager |
| Organizational Benefits | Add new drink to database. |
| Frequency of Use | Often |
| Triggers | Select “Insert Drink” |
| Preconditions | User is the manager logged in to the system. |
| Postconditions | Return tre previous menu. |
| Main Course | 1. Input drink information. 2. Save drink to database. 3. System prompt that drink is save successfully. 4. Return the Manager’s Menu. |
| Alternate Courses | AC1.1: Input drink information   * 1. Code   2. Name   3. Unit price |
| Exceptions | EX1: Product existed(code existed)  1.1. System prompt: code is existed  EX2: Valid unit price  2.1.Unit price > 0  2.2. System prompt: unit price must be greater than 0.  EX3: Insert fails   * 1. System prompt insert fails   3.2 Return to Main Course step 1. |

1.3. Use case 3: Update drink

|  |  |
| --- | --- |
| Use Case Name | Update Drink |
| Use Case ID | UC03\_Update\_Drink |
| Description | Update drink’s information existed in database. |
| Actor | Manager |
| Organizational Benefits | Update change of drink information. |
| Frequency of Use | Sometimes |
| Triggers | Select “Update Drink” |
| Preconditions | User is the manager logged in to the system. |
| Postconditions | Return the previous menu. |
| Main Course | 1. Input drink Code 2. Show drink’s information 3. Show update menu. 4. Input new properties 5. Update drink’s information. 6. System prompt that drink is update successfully. 7. Return Manager’s menu. |
| Alternate Courses |  |
| Exceptions | EX1.1: Product has Existed   1. Return user to Main courses step 1   EX4.1: Valid unit price   1. Unit price > 0.   If unit price <= 0, prompt that unit price must be greater than 0.  EX3.1: Update fails   1. System prompt insert fails 2. Return to Main Course step 1. |

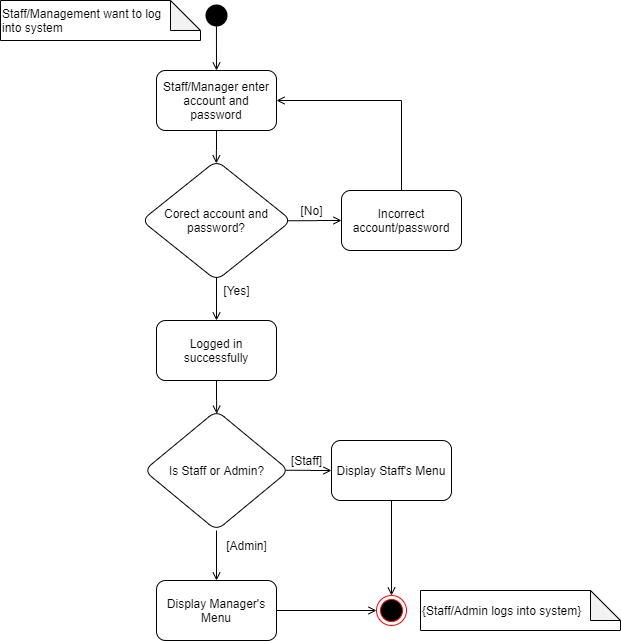
1.4 Use case 4: Create Order

|  |  |
| --- | --- |
| Use Case Name | Create Order |
| Use Case ID | UC04\_Create\_Order |
| Description | Create new order. |
| Actor | Staff |
| Organizational Benefits | Easy to manage sales and invoices. |
| Frequency of Use | Often |
| Triggers | Staff selects “Create order” in Staff’s menu. |
| Preconditions | Staff logged in to the system. |
| Postconditions | Print invoice & return the previous menu. |
| Main Course | * 1. Input drink code.   2. Input quantity, percantage of ice and sugar.   3. Save order to database.   4. System prompt that order is saved successfully.   5. Show order information.   6. Print invoice and return the previous menu. |
| Alternate Courses | AC1.1. Check drink code   1. Code is existed in order.  * Quantity = quantity + new quantity.  1. Code is not existed in order.  * Add drink into order. |
| Exceptions | EX1.1. Drink code is not existed   1. System prompt that drink code is not existed.   EX2.1 Valid quantity   1. Quantity >0.   If quantity <= 0, prompt that unit quantity be greater than 0.  EX3.1 Save order fails   1. System prompt that order is not saved successfully. 2. Return to Main Course Step 1. |

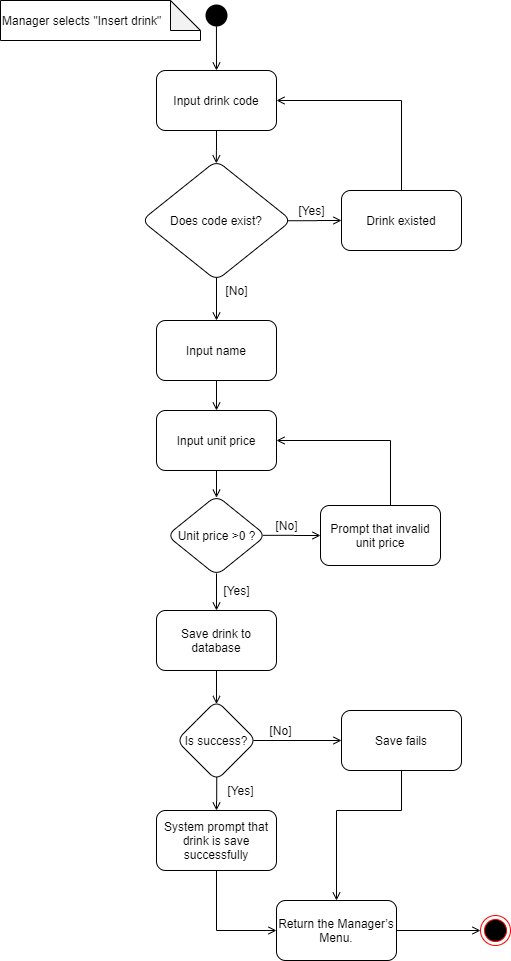
1.5 Use case 5: Update Order

|  |  |
| --- | --- |
| Use Case Name | Update Order |
| Use Case ID | UC05\_Update\_Order |
| Description | Update drink’s information in order. |
| Actor | Staff |
| Organizational Benefits | Update drink’s information in order. |
| Frequency of Use | Sometimes |
| Triggers | Staff selects “Update order” in Staff’s menu. |
| Preconditions | Staff logged in to the system. |
| Postconditions | Print updated invoice & return the previous menu. |
| Main Course | 1. Input invoice ID. 2. Show invoice details. 3. Select row of drink to update.   4. Input new properties: quantity, percantage of ice and sugar.   1. Save change to database. 2. System prompt that order is saved successfully.   7. Show invoice information. |
| Alternate Courses |  |
| Exceptions | EX1.1 Invoice ID doesn’t exist.   1. System prompt that invoice ID is not existed.   EX3.1 No row is selected.   1. System prompt that No row is selected. 2. Return to Main Course Step 3.   EX4.1 Valid quantity   1. Quantity >0. 2. If quantity <= 0, prompt that quantity must be greater than 0.   EX5.1 Save order fails   1. System prompt that order is not saved successfully. 2. Return to Main Course Step 1. |

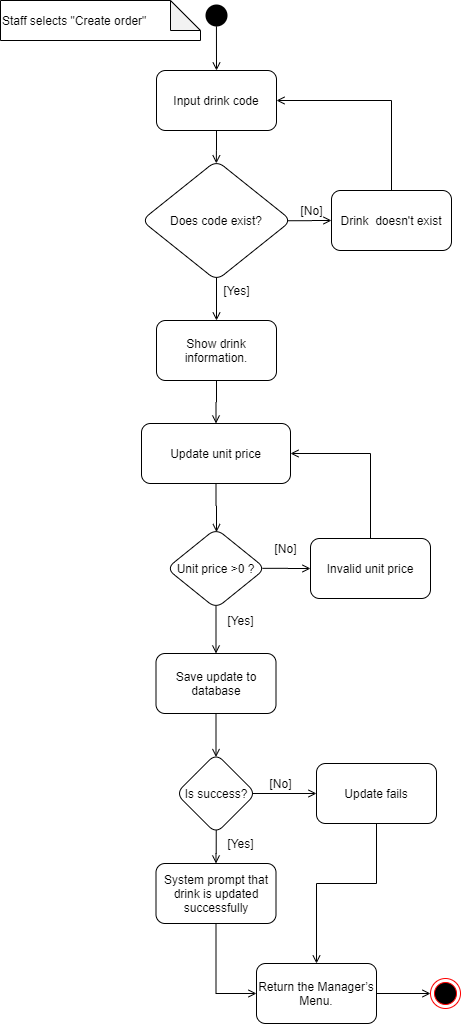
1. Activity Diagram:
   1. Login



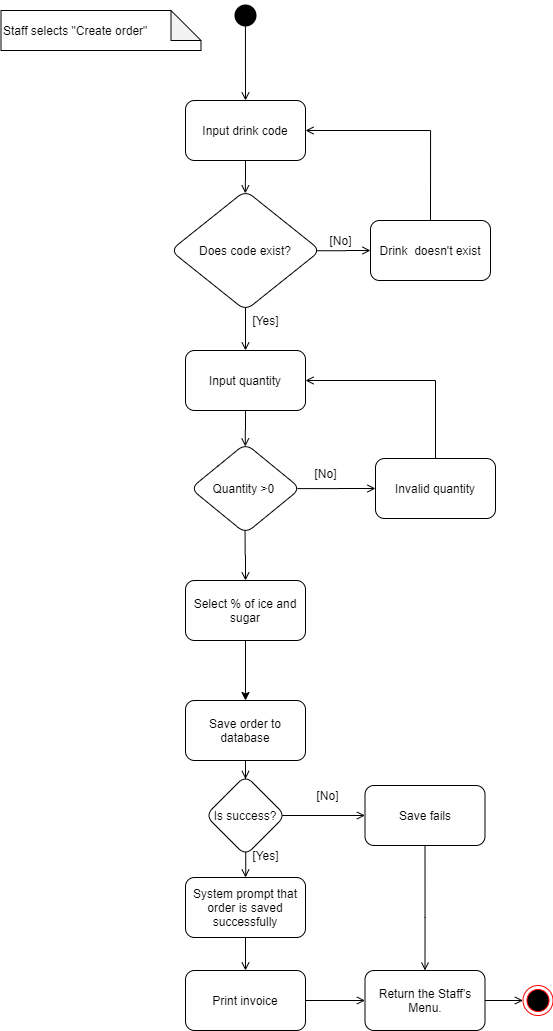
* 1. Insert drink



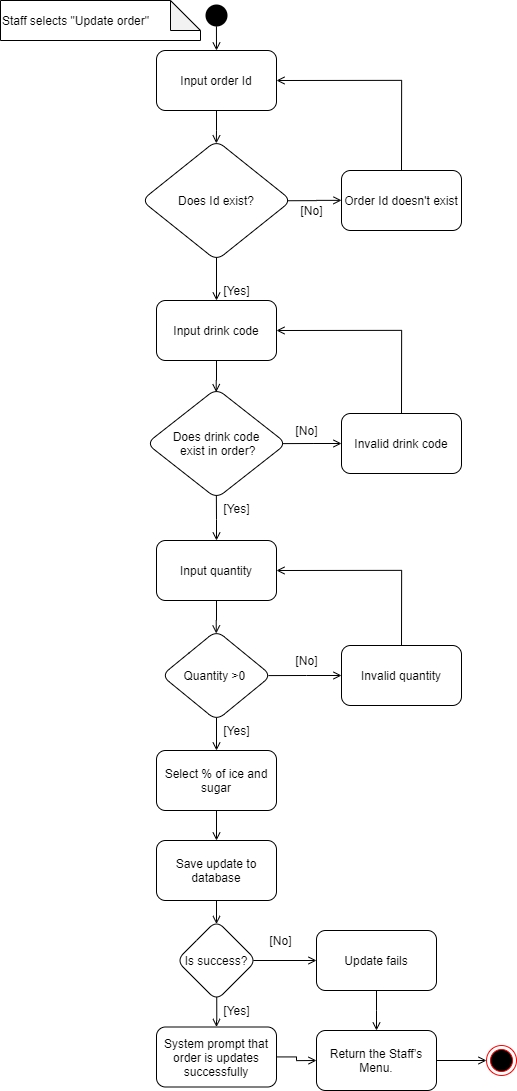
* 1. Update drink

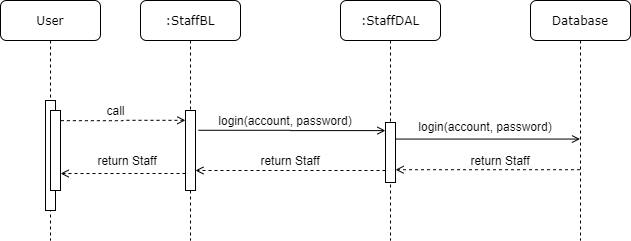


* 1. Creat Order

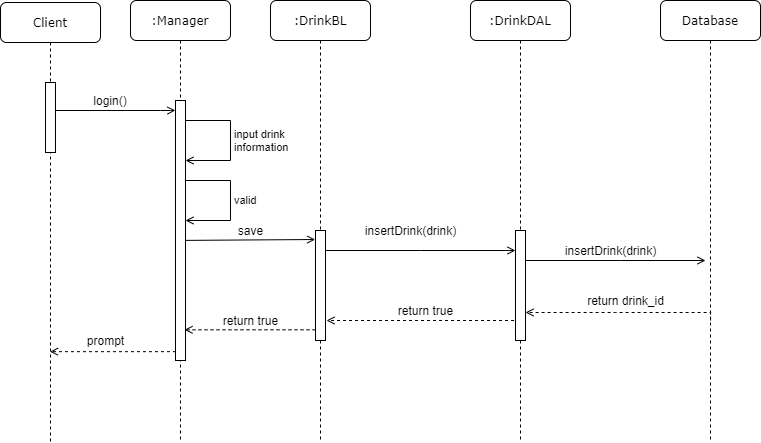


* 1. Update order

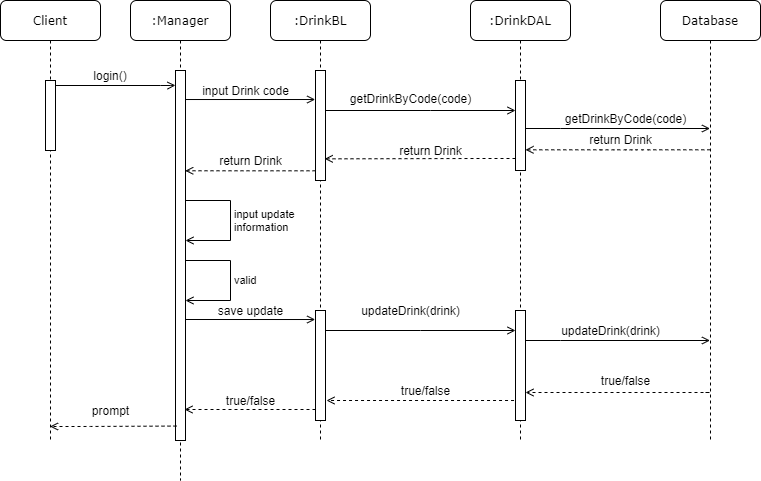


1. Sequence Diagram
   1. Login

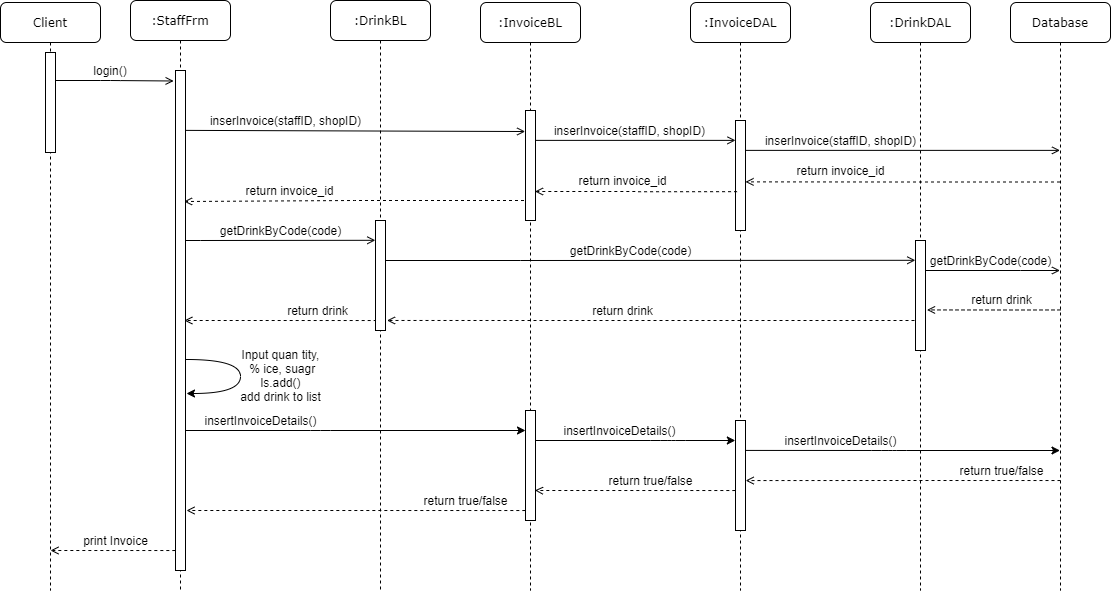
3.2 Insert drink



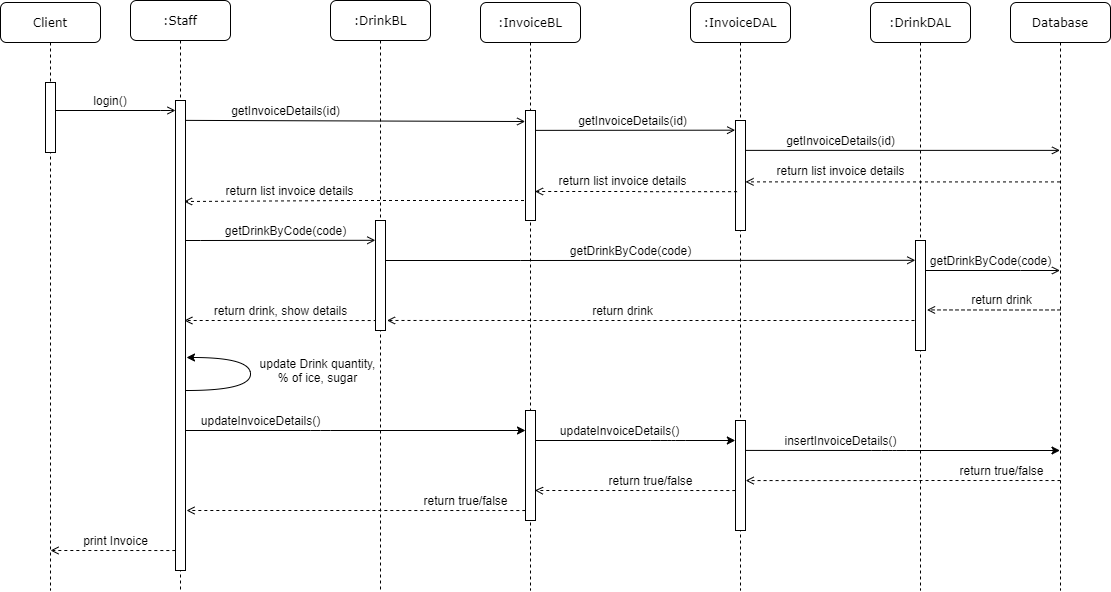
3.3 Update Drink



3.4 Insert Invoice

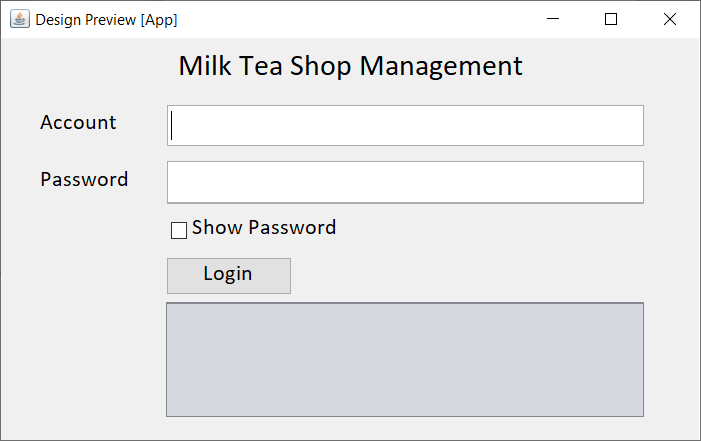


3.5 Update Invoice

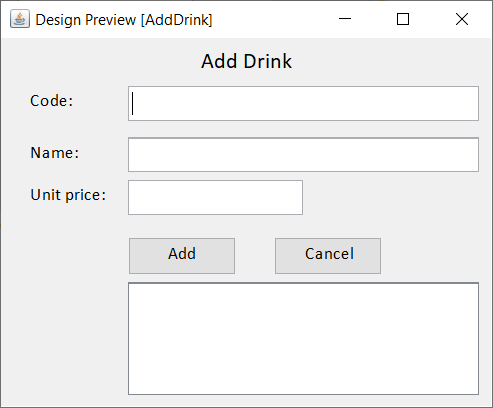


# Design Details

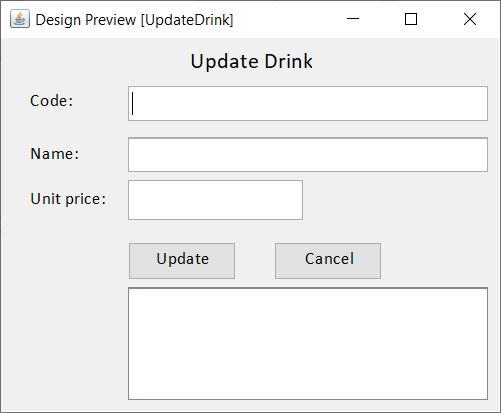
1. UI Design
   1. Login UI



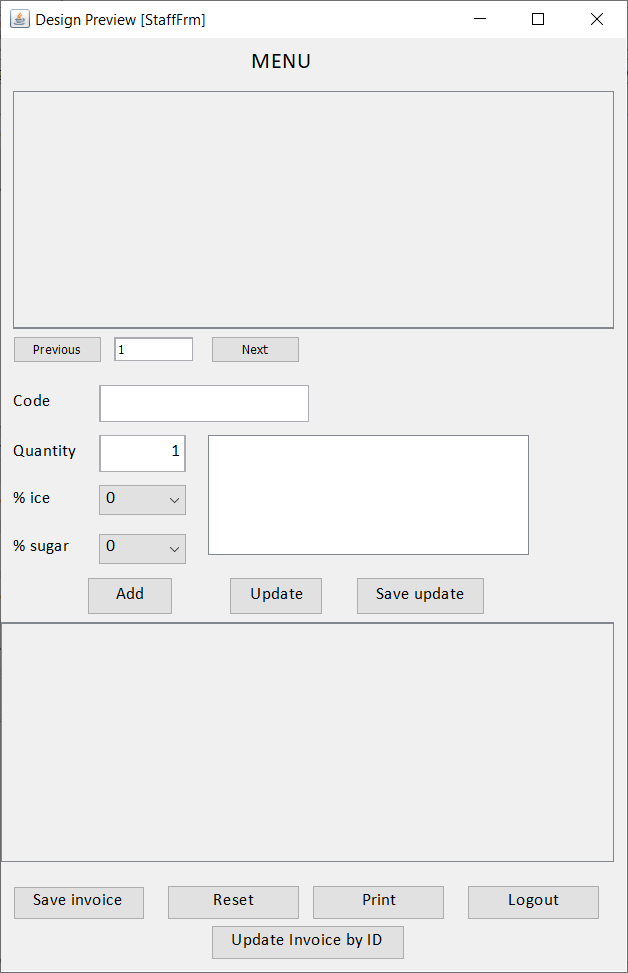
* 1. Add Drink UI



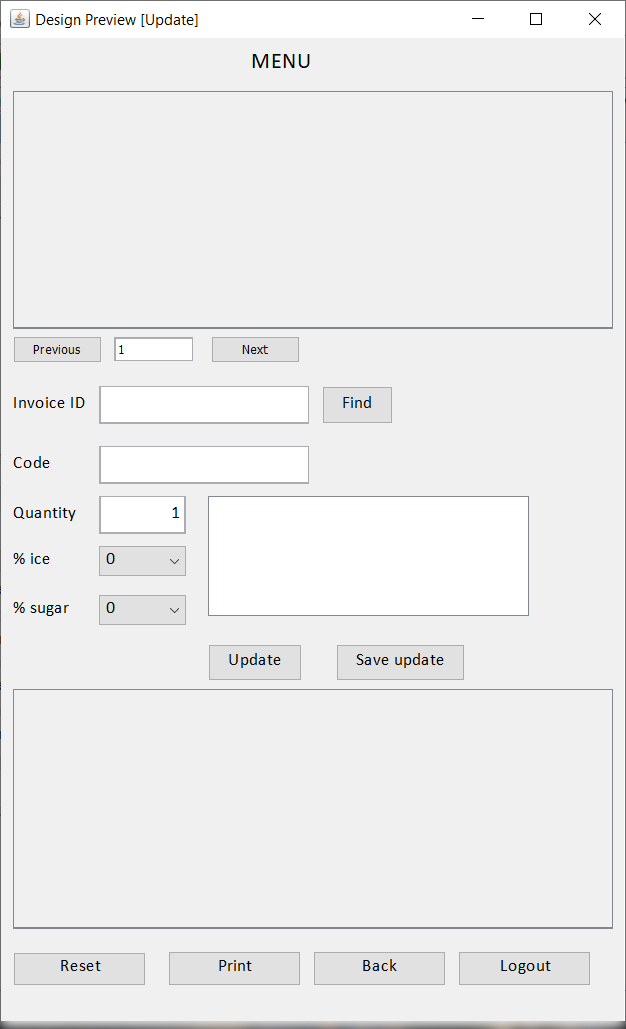
* 1. Update Drink UI

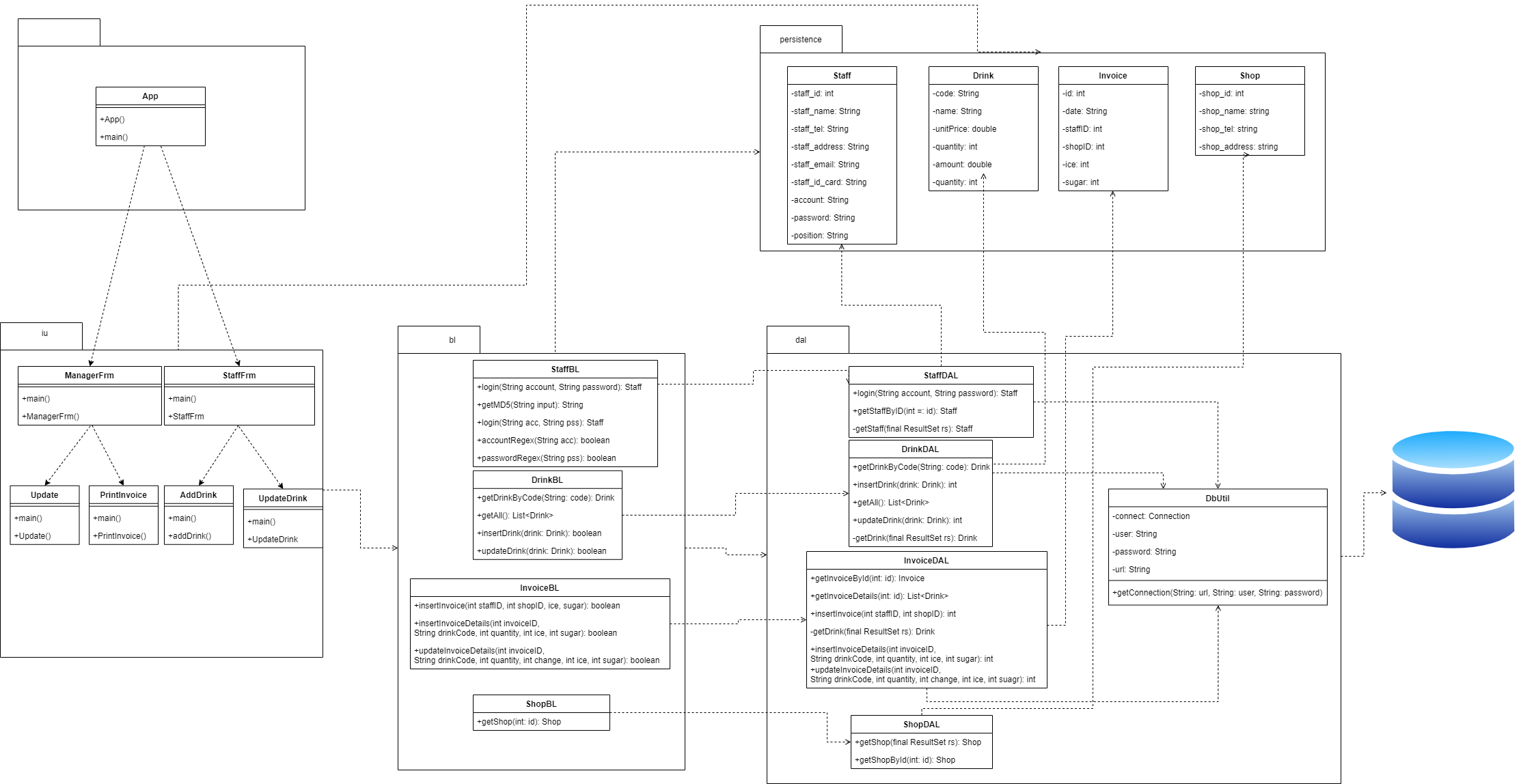


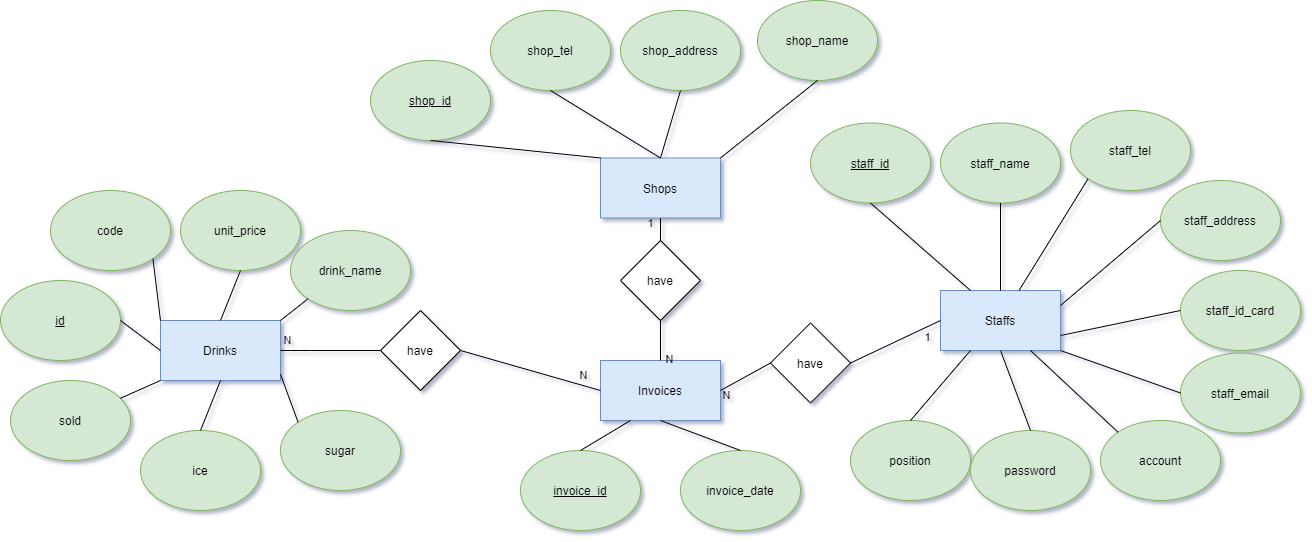
* 1. Create Invoice



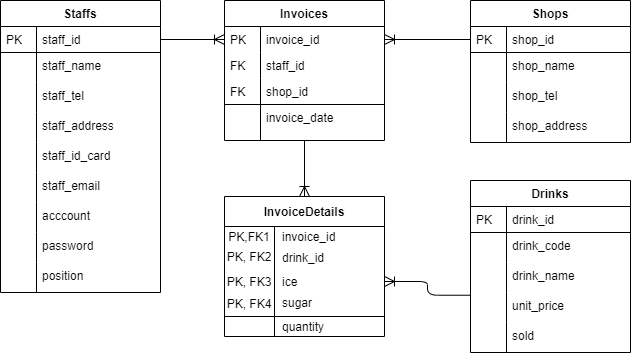
* 1. Update Invoice



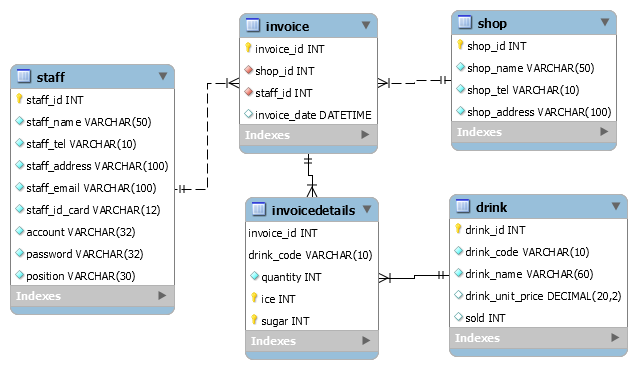
1. Code Design (Class Diagram)
2. Database Design
   1. Entity Relationship Diagram



Or



* 1. Database Design Details



|  |  |  |  |
| --- | --- | --- | --- |
| Staff | | | |
| Column Name | Data Type | Constraints | Description |
| staff\_id | int | Primary key, Auto increment, not null |  |
| staff\_name | varchar(50) | Not null | User’s full name |
| staff\_tel | varchar(10) | Not null | User’s phone number |
| staff\_address | varchar(100) | Not null | User’s address |
| staff\_email | varchar(100) | Not null | User’s email |
| staff\_id\_card | varchar(12) | Not null | User’s id card number |
| account | varchar(45) | Not null | User’s account |
| password | varchar(45) | Not null | User’s password |
| position | varchar(100) | Not null | User’s position |

|  |  |  |  |
| --- | --- | --- | --- |
| Drink | | | |
| Column Name | Data Type | Constraints | Description |
| drink\_id | int | Primary key, Auto increment,not null | Drink’s id |
| drink\_code | varchar(10) | Unique, not null | Drink’s code |
| drink\_category | varchar(45) | Not null | Drink’s category |
| drink\_name | varchar(100) | Not null | Drink’s name |
| unit\_price | decimal(20,2) | Not null, default 0 | Drink’s unit price |
| sold | int | Not null, default 0 | Drink’s sold |

|  |  |  |  |
| --- | --- | --- | --- |
| Shop | | | |
| Column Name | Data Type | Constraints | Description |
| shop\_id | int | Primary Key, Auto increment, Not null |  |
| shop\_name | varchar(45) | Not null | Shop’s name |
| shop\_tel | varchar(10) | Not null | Shop’s phone number |
| shop\_name | varchar(100) | Not null | Shop’s address |

|  |  |  |  |
| --- | --- | --- | --- |
| Invoice | | | |
| Column Name | Data Type | Constraints | Description |
| invoice\_id | int | Primary key, auto increment |  |
| staff\_id | int | Foreign Key, not null | Staff’s Id |
| shop\_id | int | Foreign Key, not null | Shop’s Id |
| invoice\_date | date | Not null | Invoice date |

|  |  |  |  |
| --- | --- | --- | --- |
| InvoiceDetails | | | |
| Column Name | Data Type | Constraints | Description |
| invoice\_id | int | Foreign key, not null | Invoice’s Id |
| drink\_id | int | Foreign key, not null | Drink’s Id |
| Ice | int | Foreign key, not null, default 0 | Drink’ percentage of ice |
| Sugar | int | Foreign key, not null, default 0 | Drink’ percentage of sugar |
| quantity | int | Not null, default 0 | Quantity of drink |

# Test

|  |  |
| --- | --- |
| Test Case Number | 1 |
| Test Case Name | getMD5Test1 |
| Test Case Description | Encrypt correctlly a string into MD5 string. |
| Preconditions |  |
| Test Case Input | “Nhatnl3004” |
| Test Case Expected Output | df7ac565f8308523e45b3b528513d8bd |
| Test Case Steps | 1. Call getMD5(“Nhatnl3004”). 2. Create expected output. 3. Compare result. |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 2 |
| Test Case Name | getMD5Test1 |
| Test Case Description | Encrypt correctlly a string into MD5 string. |
| Preconditions |  |
| Test Case Input | “Nhatnl19024” |
| Test Case Expected Output | 80be2a761fe62ab3f023e3f77dc9b8dd |
| Test Case Steps | 1. Call getMD5(“Nhatnl19024”). 2. Create expected output. 3. Compare result. |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 3 |
| Test Case Name | getMD5Test1 |
| Test Case Description | Encrypt correctlly a string into MD5 string. |
| Preconditions |  |
| Test Case Input | “cuongDeeptry1” |
| Test Case Expected Output | 38ed25b6ec4c911283f7da1626e9cead |
| Test Case Steps | 1. Call getMD5(“cuongDeeptry1”). 2. Create expected output. 3. Compare result. |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 4 |
| Test Case Name | getMD5Test1 |
| Test Case Description | Encrypt correctlly a string into MD5 string. |
| Preconditions |  |
| Test Case Input | “cuongDeeptry2” |
| Test Case Expected Output | 7a98924465e161165a7fa692b11649f1 |
| Test Case Steps | 1. Call getMD5(“cuongDeeptry2”). 2. Create expected output. 3. Compare result. |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 5 |
| Test Case Name | regexAccount1 |
| Test Case Description | Test account format. |
| Preconditions |  |
| Test Case Input | “Nhatnl3004” |
| Test Case Expected Output | True |
| Test Case Steps | 1. Call accountRegex(“Nhatnl3004”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 6 |
| Test Case Name | regexAccount2 |
| Test Case Description | Test account format. |
| Preconditions |  |
| Test Case Input | “cuongDeeptry1” |
| Test Case Expected Output | True |
| Test Case Steps | 1. Call accountRegex(“cuongDeeptry1”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 7 |
| Test Case Name | regexAccount3 |
| Test Case Description | Test account format. |
| Preconditions |  |
| Test Case Input | “Nhatnl” |
| Test Case Expected Output | False |
| Test Case Steps | 1. Call accountRegex(“Nhatnl”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 8 |
| Test Case Name | regexAccount4 |
| Test Case Description | Test account format. |
| Preconditions |  |
| Test Case Input | “98765431” |
| Test Case Expected Output | False |
| Test Case Steps | 1. Call accountRegex(“98765431”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 9 |
| Test Case Name | regexAccount5 |
| Test Case Description | Test account format. |
| Preconditions |  |
| Test Case Input | “Nhatnl99999999999999999999” |
| Test Case Expected Output | False |
| Test Case Steps | 1. Call accountRegex(“Nhatnl99999999999999999999”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 10 |
| Test Case Name | regexPassword1 |
| Test Case Description | Test password format. |
| Preconditions |  |
| Test Case Input | “Nhatnl19024” |
| Test Case Expected Output | True |
| Test Case Steps | 1. Call passwordRegex(“Nhatnl19024”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 11 |
| Test Case Name | regexPassword2 |
| Test Case Description | Test password format. |
| Preconditions |  |
| Test Case Input | “cuongDeeptry2” |
| Test Case Expected Output | True |
| Test Case Steps | 1. Call passwordRegex(“cuongDeeptry2”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 12 |
| Test Case Name | regexPassword3 |
| Test Case Description | Test password format. |
| Preconditions |  |
| Test Case Input | “123456” |
| Test Case Expected Output | False |
| Test Case Steps | 1. Call passwordRegex(“123456”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 13 |
| Test Case Name | regexPassword1 |
| Test Case Description | Test password format. |
| Preconditions |  |
| Test Case Input | “qwert” |
| Test Case Expected Output | False |
| Test Case Steps | 1. Call passwordRegex(“qwert”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 14 |
| Test Case Name | regexPassword1 |
| Test Case Description | Test password format. |
| Preconditions |  |
| Test Case Input | “Nhatnl.3040” |
| Test Case Expected Output | False |
| Test Case Steps | 1. Call passwordRegex(“Nhatnl.3040”) |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 15 |
| Test Case Name | loginTest |
| Test Case Description | Test login funtion. |
| Preconditions |  |
| Test Case Input | “Nhatnl3004”, “Nhatnl19024” |
| Test Case Expected Output | True |
| Test Case Steps | 1. Call login(“Nhatnl3004”, “Nhatnl19024”). 2. Compare result & actual. |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 15 |
| Test Case Name | getShopTest1 |
| Test Case Description | Test getShop funtion. |
| Preconditions |  |
| Test Case Input | 1 |
| Test Case Expected Output | “VTC Milk Tea”, “0837427412”, “18 Tam Trinh Hai Ba Trung Ha Noi” |
| Test Case Steps | 1. Call getShop(1). 2. Compare result & actual. |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 16 |
| Test Case Name | getShopTest2 |
| Test Case Description | Test getShop funtion. |
| Preconditions |  |
| Test Case Input | 0 |
| Test Case Expected Output | null |
| Test Case Steps | 1. Call getShop(1). 2. Compare result & actual. |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 17 |
| Test Case Name | getByCodeTest1 |
| Test Case Description | Test get drink by code funtion. |
| Preconditions |  |
| Test Case Input | “TOPPING1” |
| Test Case Expected Output | “TOPPING1”, “Pearl”, 0 , 5000 |
| Test Case Steps | 1. Call getByCode(“TOPPING1). 2. Compare result & actual. |
| Default Value Preverving |  |

|  |  |
| --- | --- |
| Test Case Number | 18 |
| Test Case Name | insertDinkTest1 |
| Test Case Description | Test insert drink funtion. |
| Preconditions |  |
| Test Case Input | “TOPPING1”, “Pearl”, 0 , 5000 |
| Test Case Expected Output | “TOPPING1”, “Pearl”, 0 , 5000 |
| Test Case Steps | 1. Call insertDrink(). 2. Compare result & actual. |
| Default Value Preverving |  |

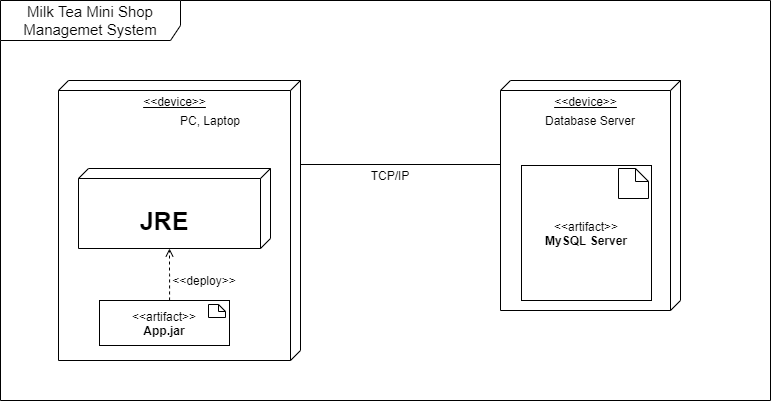
|  |  |
| --- | --- |
| Test Case Number | 19 |
| Test Case Name | updateDinkTest1 |
| Test Case Description | Test update drink funtion. |
| Preconditions |  |
| Test Case Input | 1000 |
| Test Case Expected Output | “TOPPING1”, “Pearl”, 0 , 1000 |
| Test Case Steps | 1. Call updateDrink(). 2. Compare result & actual. |
| Default Value Preverving |  |

# Assign work to each team member

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Group 2 | Milk Tea Mini Shop Management | | | | | |
| No | Task name | Description | Start Date | End Date | Member | Self assessment |
| 1 | System design | System model archtecture | 25/07/2020 |  | Nguyễn Long Nhật |  |
| 2 | Use case design | Design system | 30/07/2020 |  | Nông Hồng Cương, Nguyễn Long Nhật |  |
| 3 | Entity relationships design | Identify the ralationship between entities | 29/07/2020 | 30/07/2020 | Nông Hồng Cương |  |
| 4 | Design database | Create Database | 01/08/2020 | 07/08/2020 | Nông Hồng Cương, Nguyễn Long Nhật |  |
| 5 | Use case descriptions | Describe usecases | 01/08/2020 | 08/08/2020 | Nông Hồng Cương, Nguyễn Long Nhật |  |
| 6 | Design activity diagram | Design the system's task processing flowchart for each use case | 01/08/2020 | 08/08/2020 | Nông Hồng Cương, Nguyễn Long Nhật |  |
| 7 | Design sequence diagram | Design sequence diagrams for the system's workflow for each use case | 01/08/2020 | 28/08/2020 | Nông Hồng Cương, Nguyễn Long Nhật |  |
| 8 | Design class diagram | Design sequence diagrams for the system | 05/08/2020 | 30/08/2020 | Nông Hồng Cương, Nguyễn Long Nhật | Cương: persistance, bl.  Nhật: ui, dal |
| 9 | Coding | Write code base on class diagram | 15/08/2020 | 01/09/2020 | Nông Hồng Cương, Nguyễn Long Nhật |  |
| 10 | Update code | Fix errors, update features | 25/08/2020 | 07/09/2020 | Nguyễn Long Nhật |  |
| 11 | Deployment digram | Draw deployment digram | 01/08/2020 | 01/09/2020 | Nông Hồng Cương |  |
| 12 | Complete document |  | 01/09/2020 |  | Nông Hồng Cương |  |
| 13 | Complete slide |  | 01/09/2020 |  | Nông Hồng Cương |  |
| 14 | Update code | Code UI with Java Fx | 07/09/2020 | 18/09/2020 | Nguyễn Long Nhật |  |
| 15 | Update database | Change properties | 07/09/2020 | 18/09/20202 | Nông Hồng Cương |  |
| 16 | Complete document |  | 15/09/2020 |  | Nông Hồng Cương |  |
| 17 | Complete slide |  | 17/09/2020 |  | Nguyễn Long Nhật |  |

# Installation Instructions

1. Deployment Diagram



* 1. Install and set path JRE

[https://docs.oracle.com/goldengate/1212/gg-winux/GDRAD/java.htm#BGBFJHAB](https://docs.oracle.com/goldengate/1212/gg-winux/GDRAD/java.htm%23BGBFJHAB)

* 1. Install MySQL Server

<https://dev.mysql.com/doc/refman/8.0/en/installing.html>

1. Installation steps
2. Download .zip file from this link: <https://drive.google.com/drive/folders/1cgveLR8448_pglbgDSgfjjNBxkqmoUlV?usp=sharing>
3. Extract this file to C:\
4. Database Install

* Open cmd, type path: cd C:\
* Type: mysql –u root –p <CreateTable.sql
* And mysql –u root –p <InsertData.sql

1. Application install

* Open cmd, type path: cd C:\
* Type java –jar MilkTea-1.0-SNAPSHOT.jar

# Appendix

Some other issues

Results: build successfully an app.

Experiences: gain more experiences about Java, Mysql, Team-work,....

Problems: having many problems while we work together...