Fast Food And Ice-Cream Shop

CS162- Final Project Requirement Document



Project Supervisor

Mr. Samyan Qayyum Wahla

Project Members (Group Id Here)

|  |  |
| --- | --- |
| Member 1 Name: Muneeb-ur-Rehman | Member 1 Registration No:2019- CS-133 |
| Member 2 Name: Zain Hassan | Member 2 Registration No:2020-CS-134 |

Department of Computer Science

University of Engineering and Technology, Lahore

Pakistan

Table of Contents

[Project Description: 3](#_Toc71624246)

[Project Features: 4](#_Toc71624247)

[Technology Stack: 5](#_Toc71624248)

[Project Actors: 6](#_Toc71624249)

[Use Cases: 7](#_Toc71624250)

[Use Case 1(Write the name of the use case): 7](#_Toc71624251)

[User Interface Details 8](#_Toc71624252)

[Classes: 9](#_Toc71624253)

[Object Oriented Features: 10](#_Toc71624254)

[Composition: 10](#_Toc71624255)

[Inheritance: 10](#_Toc71624256)

[Multiple Inheritance: 10](#_Toc71624257)

[Multi-Level Inheritance: 10](#_Toc71624258)

[Polymorphism: 10](#_Toc71624259)

[Collections: 11](#_Toc71624260)

[Exceptions: 12](#_Toc71624261)

[Data Storage: 13](#_Toc71624262)

[Email Sending: 14](#_Toc71624263)

[Project Plan 15](#_Toc71624264)

# Project Description:

In this Project we will manage the Fast Food And Ice-Cream Shop.

**Admin:**

This program is used by admin for managing Stores, product and for creating order and for Paying bills. There is no need to maintain the record manually. It can maintain the record of Product, Store, Categories, and etc. so that it will be easy to access any time 24\*7. This project will perform all the CRUD operations like Add, Delete, Update and Read functions. With this system the tasks would be performed in less amount of time and more efficiently.  In which at first there will be login for the admin. Admin can login to his account by adding valid Id and Password. Admin will be able to see the feedback enter by customer. Admin will be able to leave the section by logging out using the password set for him.

The admin can add, delete, edit the order on the demand of customer.

**Customer:**

Customer will login the customer section to buy the different products. Customer can view all the products entered by the admin. He can get discount offer if he will buy things greater than Rs1500. Discount will be 20 percent of the total bill. The customer can choose the food he want to eat and can replace the item with other. Customer can cancel the order and give a new one.

After login there will be a menu bar in which two types of menu will be displayed:

* **Fast Food**
* **Ice Cream**

Admin will select the category after taking order from the customer.

Further, there will be sub-categories in each of the menu. In Fast Food there will be sub-categories:

* **Shawarma**
* **Burgers**
* **Pizza**

And in Ice Cream there will be sub-category of:

* **Cup Ice Cream**
* **Cone**
* **Family Pack**
* **Kulfi**
* **Stick Ice Cream**

Prices of each food will be applied when the customer will order the specific items. There will be a special discount on the order over Rs 1500. There will be specific service charges in the bill of each customer. Drinks will also be available on the customer’s demand. **Fast Food** will contain Burger (Egg, Chicken, Zinger), Shawarma (Chicken, Zinger), Pizza (Chicken Tikka, Malai Boti, Veg, Pepperoni, BBQ, Hawaiian). Desi Food will contain Malai Boti, Karahi, Mutton, Beef, Biryani etc.

Ice Cream will be available in Mango, Banana, Blueberry, Kulfa, Vanilla and Chocolate flavours.

There will be three types of services:

* **Dine In**
* **Take Away**
* **Home Delivery**

Menu displayed will contain all types of fast food and ice creams. Customer can choose the food he / she wants to order the items will display on the screen with the specified price and the bill will pop out. After ordering the food he wants to pay the bill, he will press the specific number and then the bill receipt will be shown to him and then he will be asked to give the amount on the bill.

* In this project the admin can add, delete and update the menu in menu bar. He can add more item on the customer reviews. On the other hand, the customer can add, delete items in his/her order.

**Detail:**

At first there will be a login page admin and customer both can login at that page. After logging in as admin, admin can add, delete and update items in the menu. He/ She can view the feedback given by the customers. And logging in as customer, customer can view menu, buy products, pay bill ang give the feedback. Customer can also give suggestions to admin on the feedback.

# Project Features:

In this section, write down the project features clearly. These are the features that will be available to the customer of your project directly.

**Admin**

* **Add items in the Menu**
* **Delete items from the Menu**
* **Update items in the Menu**

**Customer**

* **Add items in the Order**
* **Delete items in the Order**
* **View Menu**
* **Buy Products**
* **Pay Bill**

# Technology Stack:

|  |  |
| --- | --- |
| Language(C#/Java) | Java |
| Platform(Web/Desktop) | Desktop |
| Frontend Technology(Simple HTML/Bootstrap/any other library) | Html, Bootstrap, JavaScript, C++ |
| IDEs | Visual Studio 2019, Notepad, StarUML, Pencil Tool |

# Project Actors:

**Admin:**

**Admin** can login to his account by adding valid Id and Password. Admin will be able to see the feedback enter by customer. Admin will be able to leave the section by logging out using the password set for him.

The admin can add, delete, edit the order on the demand of customer.

**Customer:**

**Customer** will login the customer section to buy the different products. Customer can view all the products entered by the admin.The customer can choose the food he want to eat and can replace the item with other. Customer can cancel the order and give a new one.

# Use Cases:

All the use cases should be written in the following format. The name of use should be start with a word e.g. Add Student. It should not be as Student/ Student Add. Add Student is separate use case and Edit Student is separate use case:

## Use Case 1(Write the name of the use case):

|  |  |
| --- | --- |
| Use Case ID | U01 |
| Name | Add item in Menu |
| Actor | Admin |
| Description | In this case admin can add the items in the menu. |
| Layout in pencil tool |  |
| Validators | nameValid(string Name);  priceValid(int price); |

**Use Case 2:**

|  |  |
| --- | --- |
| Use Case ID | U02 |
| Name | Delete item from Menu |
| Actor | Admin |
| Description | In this case admin can delete the added items from the menu. |
| Layout in pencil tool |  |
| Validators | nameValid(string Name); |

**Use Case 3:**

|  |  |
| --- | --- |
| Use Case ID | U03 |
| Name | Update item in Menu |
| Actor | Admin |
| Description | In this case admin can update the added items in the menu. |
| Layout in pencil tool |  |
| Validators | nameValid(string Name);  priceValid(int price); |

**Use Case 4:**

|  |  |
| --- | --- |
| Use Case ID | U04 |
| Name | Add item in Order |
| Actor | Customer |
| Description | In this case customer can add the items in the order. |
| Layout in pencil tool |  |
| Validators | nameValid(string Name);  numberValid(int num); |

**Use Case 5:**

|  |  |
| --- | --- |
| Use Case ID | U05 |
| Name | Delete item from Order |
| Actor | Customer |
| Description | In this case customer can delete the added items from the order. |
| Layout in pencil tool |  |
| Validators | nameValid(string Name); |

**Use Case 6:**

|  |  |
| --- | --- |
| Use Case ID | U06 |
| Name | View Menu |
| Actor | Customer |
| Description | In this case customer can view the menu. |
| Layout in pencil tool |  |
| Validators | catValid(string Cat); |

**Use Case 7:**

|  |  |
| --- | --- |
| Use Case ID | U07 |
| Name | Buy Products |
| Actor | Customer |
| Description | In this case customer can buy/order from the menu. |
| Layout in pencil tool |  |
| Validators | catValid(string Cat);  numberValid(int num); |

**Use Case 8:**

|  |  |
| --- | --- |
| Use Case ID | U08 |
| Name | Pay Bill |
| Actor | Customer |
| Description | In this case customer can pay bill of the order he has given. |
| Layout in pencil tool |  |
| Validators | numberValid(int num); |

**Use Case 9:**

|  |  |
| --- | --- |
| Use Case ID | U09 |
| Name | Login as Admin |
| Actor | Admin |
| Description | In this case admin can login to their account by adding specific email and password. |
| Layout in pencil tool |  |
| Validators | emailValid(String email);  passValid(String a, int b); |

**Use Case 10:**

|  |  |
| --- | --- |
| Use Case ID | U10 |
| Name | Login as Customer |
| Actor | Customer |
| Description | In this case customer can login to their account by adding specific email and password. |
| Layout in pencil tool |  |
| Validators | emailValid(String email);  passValid(String a, int b); |

# User Interface Details

In this section, fill the table for summary that which use case will have the required component. Inside each box, write the counts for each component. If component is not used, write zero.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Use Case Id | TextBox | DropDown | Password Box | Label | Date Field | Buttons | AutoComplete | Radio Button | CheckBox | Menu | Text Area | ProgressBar |
| U01 | 2 | 0 | 0 | 3 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| U02 | 1 | 0 | 0 | 2 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| U03 | 4 | 0 | 0 | 5 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| U04 | 2 | 1 | 0 | 3 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| U05 | 1 | 0 | 0 | 2 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| U06 | 0 | 2 | 0 | 3 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| U07 | 0 | 3 | 0 | 4 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| U08 | 1 | 0 | 0 | 3 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| U09 | 2 | 0 | 0 | 3 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| U10 | 2 | 0 | 0 | 3 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |

# Classes:

In this section, we do not require detailed design diagram. But identify the tentative classes with the requirement that you should have at least 10 domain classes and 5 software classes, 3 abstract classes, 2 singelton classes. Fill the following table for details. Note that class name should follow naming conventions

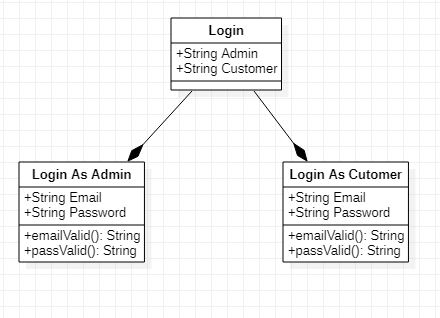
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class Name** | **Software/ Domain** | **Is Abstract (Yes/No)** | **Is Singleton (Yes/No)** | **Is the class will has parametrized constructor(Yes/No)** |
| Login | Software | Yes | No | No |
| Login as Admin | Software | No | No | Yes |
| Login as Customer | Software | No | No | Yes |
| Admin | Domain | No | No | Yes |
| Add | Domain | No | No | Yes |
| Delete | Domain | No | Yes | Yes |
| Update | Domain | No | No | Yes |
| Customer | Domain | No | No | Yes |
| Add | Domain | No | No | Yes |
| Delete | Domain | No | Yes | Yes |
| Menu | Domain | No | Yes | Yes |
| Products Buy | Domain | No | No | Yes |
| Billing | Software | Yes | Yes | Yes |
| Discount | Domain | No | No | Yes |
| Exit | Software | Yes | No | No |

# Object Oriented Features:

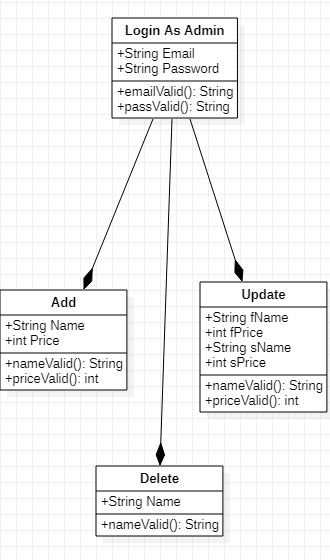
## Composition:

In this section, Identify the at least three solid examples where composition can be perform. Add UML diagram of each example.

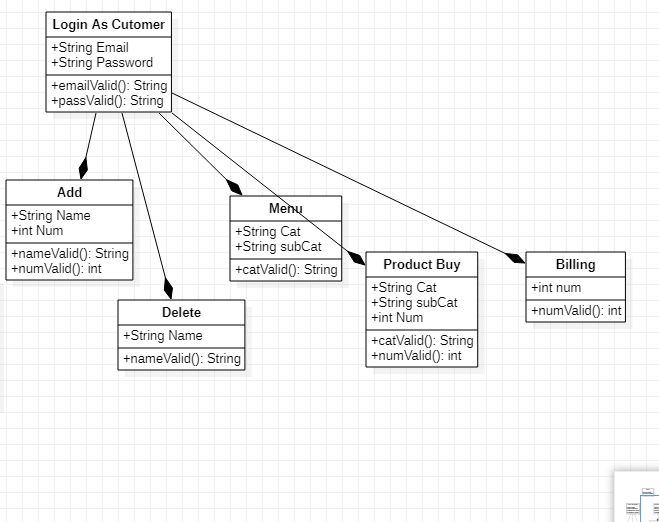
**UML 1:**

****

**UML 2:**

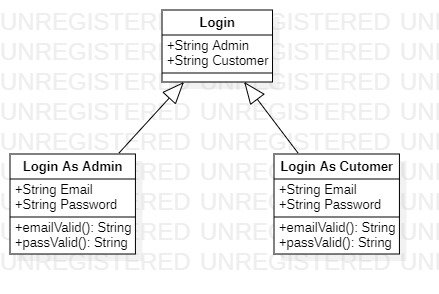
****

**UML 3:**

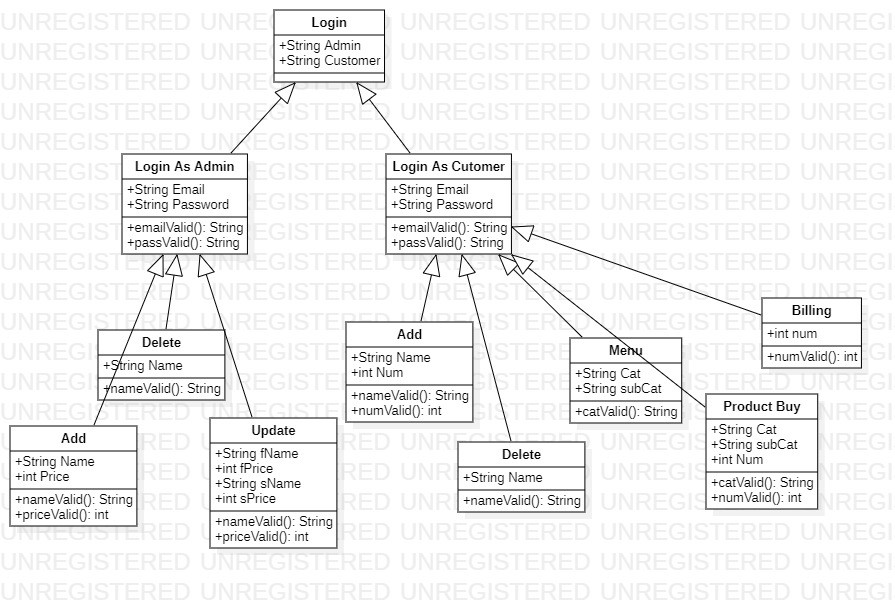
****

## Inheritance:

**UML 1:**

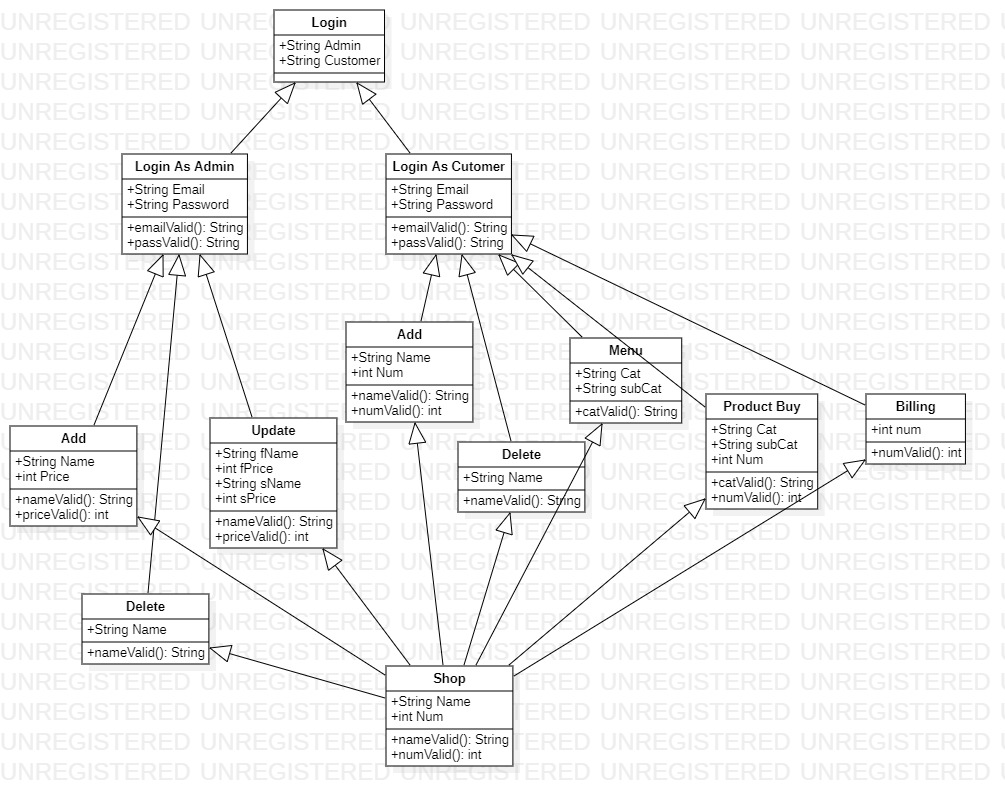
****

**UML 2:**

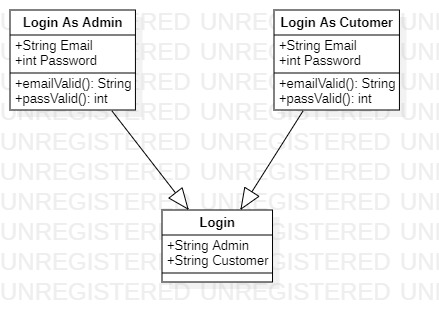


## Multiple Inheritance:

**UML 1:**

****

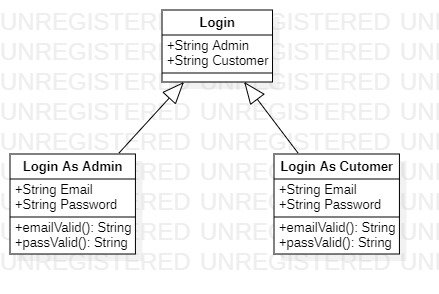
**UML 2:**



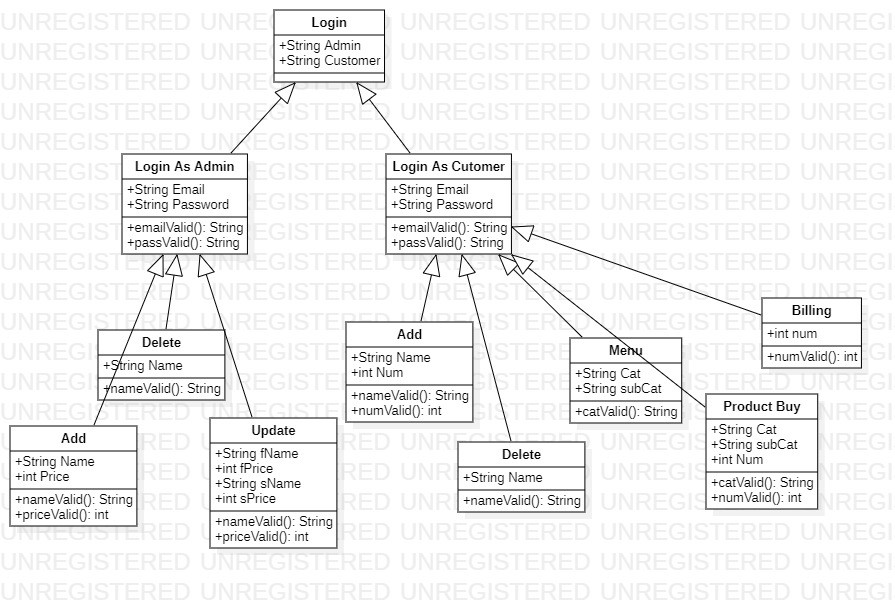
## Multi-Level Inheritance:

Two examples with UML diagrams of multi-level inheritance.

**UML 1:**

****

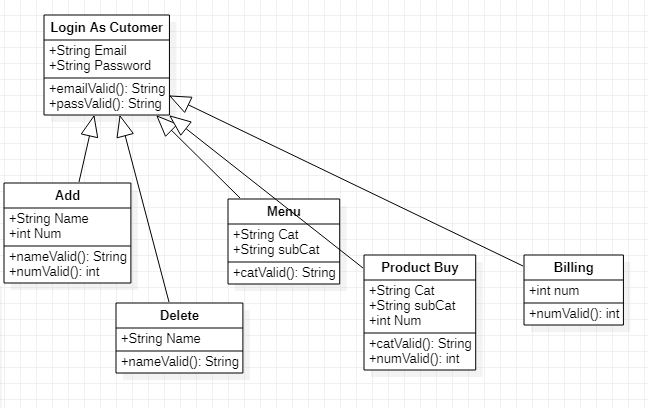
**UML 2:**



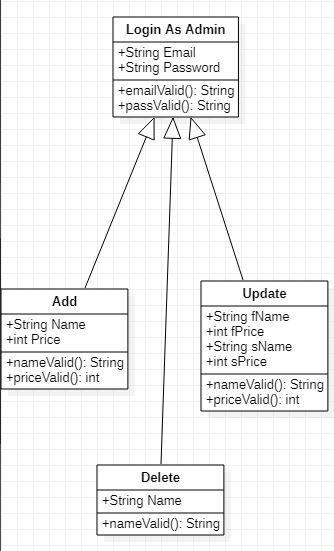
## Polymorphism:

At least 3 examples with UML diagram for polymorphism.

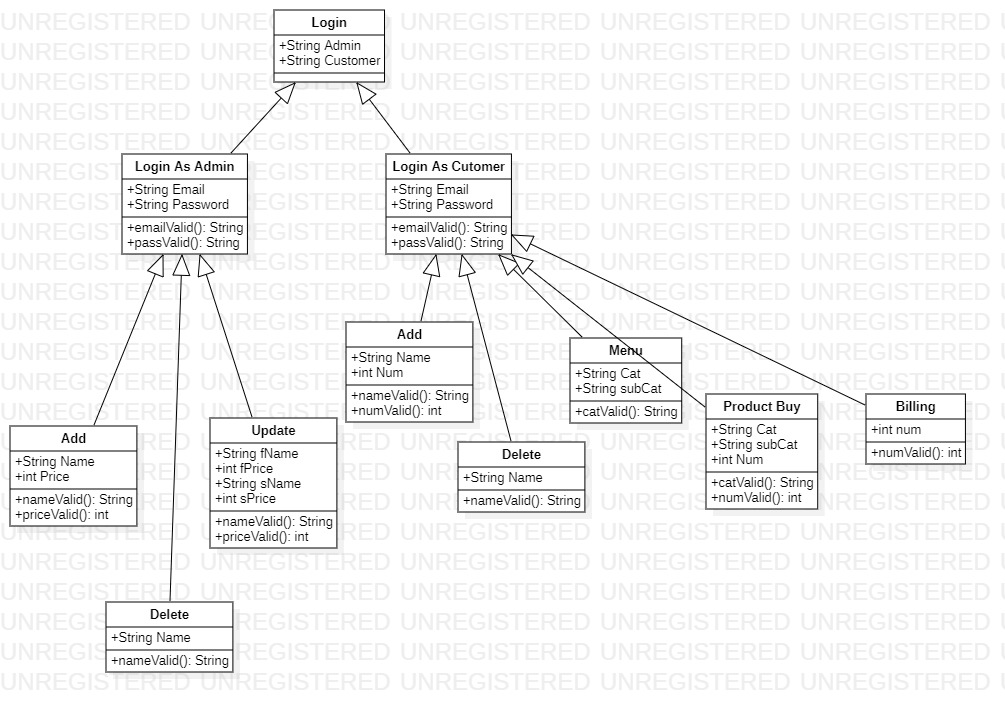
**UML 1:**

****

**UML 2:**

****

**UML 3:**

****

# Collections:

In this section, describe how and where you will use the following collections and why you are forced to use these collections.

ArrayList, LinkedList, Queue, Stack, HashSet and TreeSet

**Array List:**

Array list will be used to add the bills paid by the customers.

**LinkedList:**

**…..**

**Queue:**

Queue will be used to take the orders of the customer on first come first serve basis.

Bill Payment.

**Stack:**

Not Available…

**HashSet:**

Orders will be taken in HashSet List.

**TreeSet:**

Orders will be controlled by TreeSet.

# Exceptions:

In this sections, identify at high level which type of exceptions you can face in your code and what are the solutions. Add more rows in the table as per requirements.

*Exceptions will come out after we code and run it so at this time there is no run time or compile time exception.*

|  |  |  |
| --- | --- | --- |
| Type of Exception | Why this exception will occur | How you will handle the exception |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Data Storage:

In this sections, describe the five files with their format from where you will read or store data. In case of database table, write down the names of columns for tables. At least 5 files/ tables are required.

* **Bill.txt**
* **Order.txt**
* **FeedBack.txt**
* **Menu.txt**
* **File.txt (Combine storing)**

# Email Sending:

In this section, describe the points where you will be required to send the email from the code. Additionally, write down the sample subject and email content.

At login time email will be provided by the admin and customer.

# Project Plan

This section should include the implementation plan and work division among the members. All the estimated dates should be before June 15, 2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Id** | **Use Case Name** | **Member Name** | **Estimated Completion Date** |
| 1 | Add | Zain Hassan | 12-06-2021 |
| 2 | Delete | Zain Hassan | 12-06-2021 |
| 3 | Update | Zain Hassan | 12-06-2021 |
| 4 | Add (Customer) | Muneeb-ur-Rehman | 12-06-2021 |
| 5 | Delete (Customer) | Muneeb-ur-Rehman | 12-06-2021 |
| 6 | Menu | Muneeb-ur-Rehman | 12-06-2021 |
| 7 | Products Buy | Muneeb-ur-Rehman | 12-06-2021 |
| 8 | Billing | Muneeb-ur-Rehman | 12-06-2021 |
| 9 | Login As Admin | Zain Hassan | 12-06-2021 |
| 10 | Login As Customer | Zain Hassan | 12-06-2021 |