**SOFTWARE REQUIREMENTS SPECIFICATION**

**FOR**

**ONLINE FOOD ORDERING SYSTEM**

**Prepared by: Bushra (63759)**

**Submitted to: Aleenah Khan**

**Date: 22th November, 2020**

Contents

[**Introduction** 3](#_Toc56973501)

[Purpose 3](#_Toc56973503)

[Software Scope 3](#_Toc56973504)

[**Overall Description** 3](#_Toc56973505)

[Software Perspective 3](#_Toc56973506)

[Software Functions 3](#_Toc56973507)

[**Manager Module** 3](#_Toc56973508)

[**Customer Module** 3](#_Toc56973509)

[**Flow of Project** 4](#_Toc56973510)

[**Flow of Event** 4](#_Toc56973511)

[USECASE DIAGRAM 9](#_Toc56973512)

[ACTIVITY DIAGRAM 10](#_Toc56973513)

[Requirements for Project 11](#_Toc56973514)

[Software 11](#_Toc56973515)

[Hardware 11](#_Toc56973516)

[Project Layout 11](#_Toc56973517)

[Realization of Classes 14](#_Toc56973518)

[**Sequence Diagrams** 17](#_Toc56973519)

[**Class Diagram** 25](#_Toc56973520)

[**Connection Class** 26](#_Toc56973521)

[**Working Output** 27](#_Toc56973522)

# **Introduction**

## Purpose

This project is serving as a food ordering system based on desktop application. This software will help the Restaurants to manage their orders supplying the customers with online order functionality. This will lead to quick and easy management.

## Software Scope

The objective of this software is to improve the efficiency of food ordering, time management, accuracy and simplify the ordering system from both ends i.e. User and Admin/Customer and Manager. Customer can view the available items and will have a visual confirmation of their orders correctly.

This software will result in efficient way of online food ordering system with proper management of data.

# **Overall Description**

## Software Perspective

This Software can facilitate to any restaurant for online food ordering. This will simplify the process of order for customer as well as for restaurant. When the customer visits an online and interactive platform will have menu with all options. Once the food is ordered its details will have entered in database and this all is going to be managed by the application with minimal delay.

Thus the software helps the restaurant end to handle all process/orders without any confusion. Moreover, there will be a software admin who is going to manage the stock and can also view and manage customers end though possessing all rights for the software. Playbill for every order is maintained and feedback can also be entered.

## Software Functions

### **Manager Module**

* Register
* Login
* Insert update delete search (items)
* View Customers
* View Receipts
* Managing Stock

### **Customer Module**

* Register
* Login
* Order Placement
* View Order
* Pay Bill

# **Flow of Project**

This project consists of two actors i.e. Manager/Admin and Customer. Firstly, both are allowed to use LOGIN use case with valid credentials if they are registered to have a access to the system.

The admin can perform operations like insert, update, delete, and as well as search. Admin can also view customers and order details.

Furthermore, Customer is able to view menu and as well as search for the food item and can place order. If customer wants to cancel the order he/she can simply do it by clicking the cancel order the status for order will be changed to canceled. Admin can also cancel the order if it is out of stock or currently not offering. Customer is able to give feedback whereas admin can view feedback of customer and in the end they can logout.

# **Flow of Event**

**LOGIN**

**Main Flow**

This is the process by which the users logged in to the system. Thus the users refer to both actors i.e. Customer and Admin of Online Food Ordering System. First the users will register themselves by those credentials they are validated to logged in.

**Sub Flow**

When the application is started this use case will be used

Interface for Login will be displayed

Enter the valid credentials and logged in to the system.

**Alternate Flow**

If in the sub flow actor is not registered, then will proceed to registration and if invalid credentials so repeat the process of sub flow.

**PLACE ORDER**

**Main Flow**

After successfully logged in customer can now place the order.

**Sub Flow**

A form of Place Order is displayed to the customer

Customer can choose the order from the menu

Customer can also search for the menu

Customer can select multiple items

Customer then click to order.

**Alternate Flow**

If customer ordered for a specific item and amount is huge, customer will not be served the order.

If order is out of stock can be cancelled

**VIEW ORDER**

**Main Flow**

Once customer ordered he can then choose to view the order.

**Sub Flow**

The form for view will display details for order

The form will have order ID, customer ID, contact number, delivery details (address), item name, quantity, and total amount.

**Alternate Flow**

None.

**PAY BILL**

**Main Flow**

Once Customer ordered he can no switch to the pay bill use case.

**Sub Flow**

In the form of pay bill customer has a button when clicked will make his/her payment.

**Alternate Flow**

Customer unable to pay due to un sufficient amount of balance.

**VIEW RECEIPT**

**Main Flow**Customer can see detail of order in the receipt.

Admin can see the receipt of the order

**Sub Flow**

Order Id, Customer Id, Item name, Quantity and Total amount will be displayed in receipt form

**Alternate Flow**

The system will have displayed error message when customer Id and Order Id not exits for particular order.

**CANCEL ORDER**

**Main Flow**

The process by which a customer and admin can cancel an order is defined in this Use Case. The customer wants to cancel the order can cancel it any time and same goes for system admin.

**Sub Flow**

The Use Case starts when the customer selects Cancel Order

The system will display the cancel order message for the cancel order

**Alternate Flow**

No orders for this customer

In system order ID not found

In system customer ID not found

If the customer wants to cancel particular order and order date is equivalent to current date the order will be cancelled.

**MANAGE ITEM**

**Main Flow**

When system admin wants to perform some basic operations on its stock then this use case starts. Insert Item, Update Item, Search Item, and Delete Item are included in operations.

If the activity selected is **INSERT ITEM**, the S-1: Insert Item sub flow is performed.

If the activity selected is **UPDATE ITEM**, the S-2: Update Item sub flow is performed.

If the activity selected is **DELETE PRODUCT**, the S-3: Delete Product sub flow is performed.

If admin selected **SEARCH ITEM** activity the S-4: Search Item sub flow is performed.

A form containing four buttons for four different operations is displayed in system. The particular operation is performed by clicking on desired button by the admin.

**Sub Flow**

**Insert Item**

The information about items will be inserted by clicking on insert button i.e. item name, item quantity, item price etc. If the item will insert successfully a message “Successfully Inserted” will display on the screen

**Update Item**

Information about items will be updated by clicking on update button on the system

Initially he will select the item name and then he will update the information. When he will complete the steps, system will update the product information and use case will begin again. If the item will update successfully a message “Successfully Updated” will display on the screen

**Delete Item**

The system will show the list of available items to the administrator from record then administrator will select the product he wants to delete from record. A message “Successfully Deleted” will display on the screen if the item deleted successfully.

**Search Item**

When admin will have to search for a particular item from record the system will display a form on which Search Item button will be visible. The admin will click on it and the desired item will be displayed. A “message successfully” searched will be display on the screen if the item will be searched successfully.

**Alternate Flow**

Searching an item the system will display error message if the form has error in inserting item, updating item, deleting particular item

**VIEW CUSTOMER**

**Main Flow**

In the VIEW CUSTOMER screen system admin can view all the customers.

**Sub Flow**

A screen will show the details about customer including Customer ID.

**Alternate Flow**

Error message will display if there is no customer in the system

**GIVE FEEDBACK**

**Main Flow**

Customer can provide feedback by using this use case

**Sub Flow**

System will display the given feedback form to the customer.

The customer will type in a message on the "Feedback" dialog.

The customer will submit the feedback

**Alternate Flow**

Instead of sending, it can be cancelled.

**VIEW FEEDBACK**

**Main Flow**

The admin will evaluate the feedbacks given by the customers using this use case.

**Sub Flow**

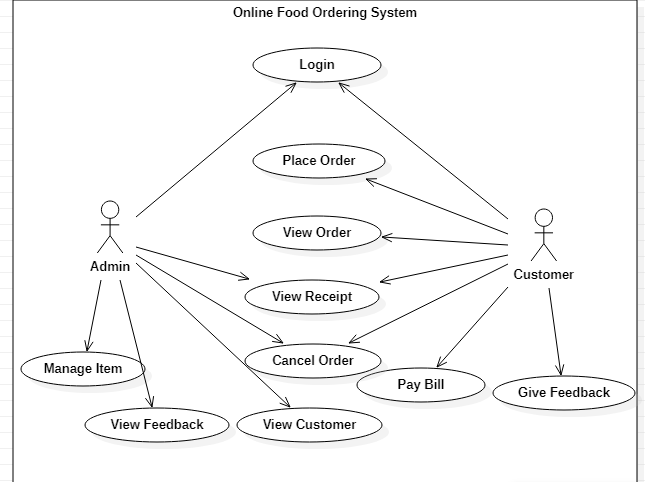
The admin opens the "Process Feedbacks" dialog.

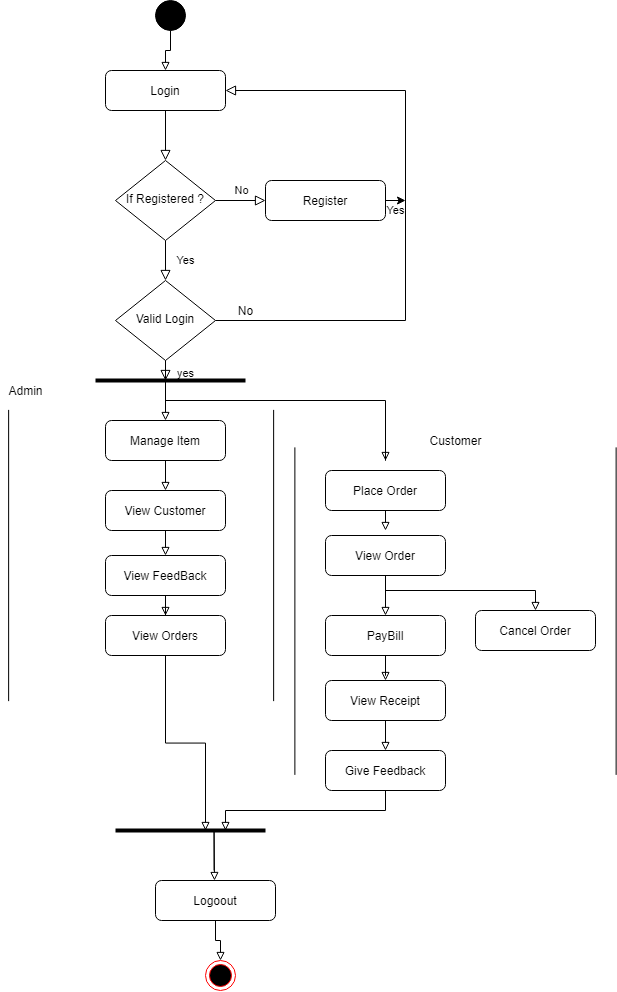
Read Feedbacks

**Alternate Flow**

None

# USECASE DIAGRAM

****

ACTIVITY DIAGRAM  
  


# Requirements for Project

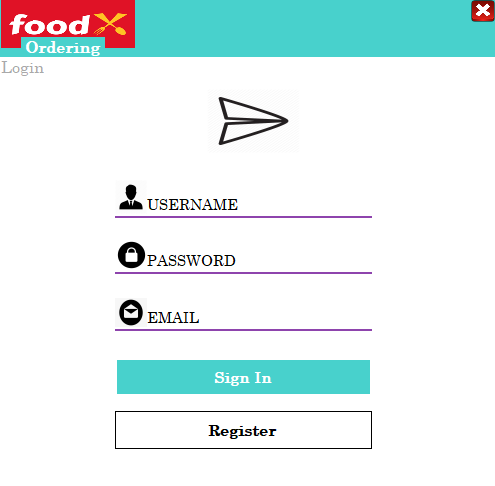
## Software

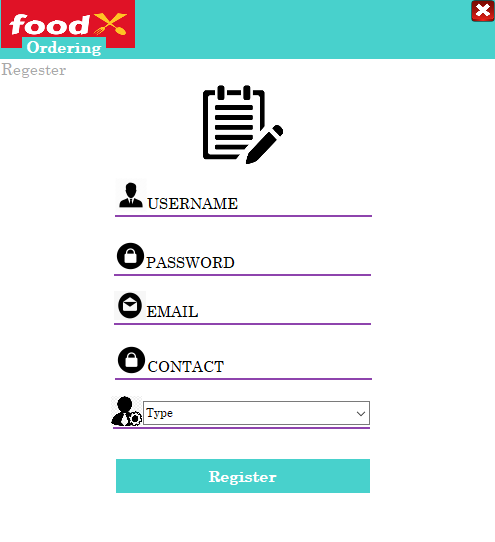
The software required to run this system is Visual Studio, Language is C# and database will be maintained in SQL.

## Hardware

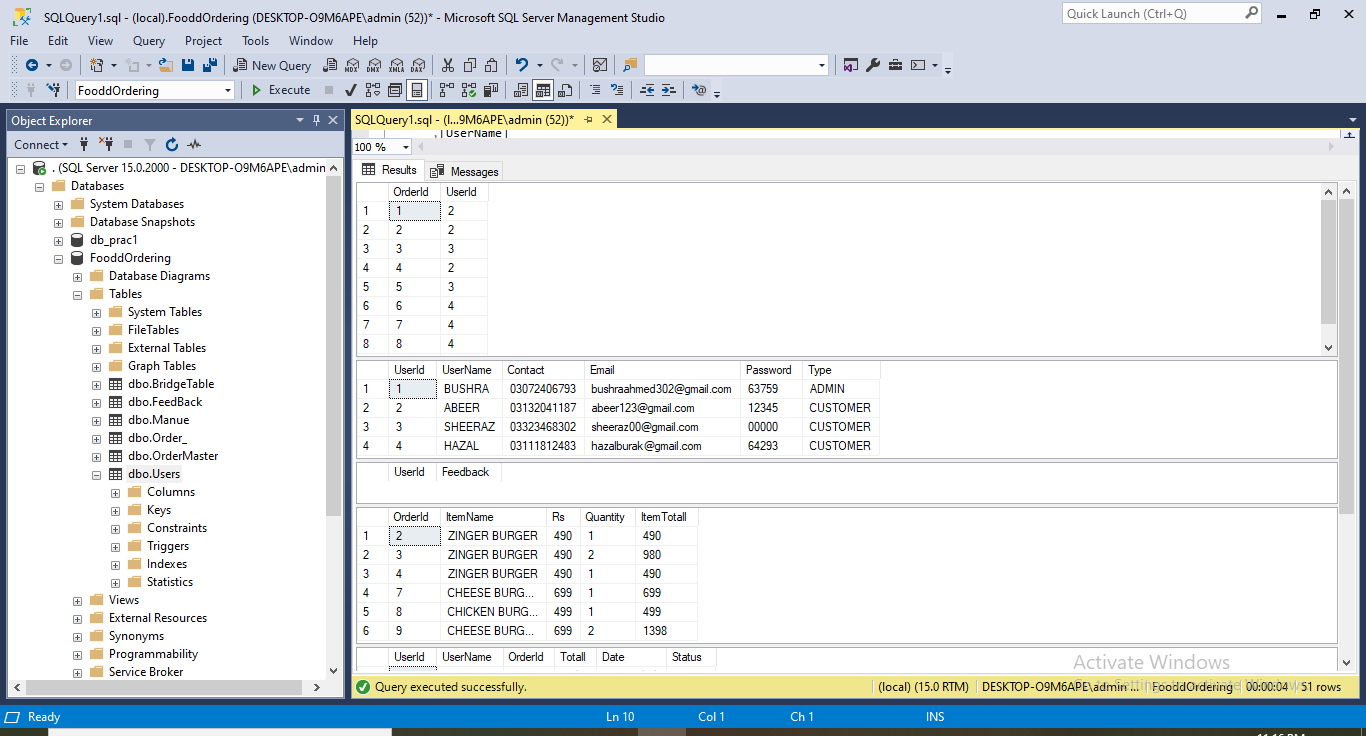
Since the project is based on software so there is not any requirement of hardware.

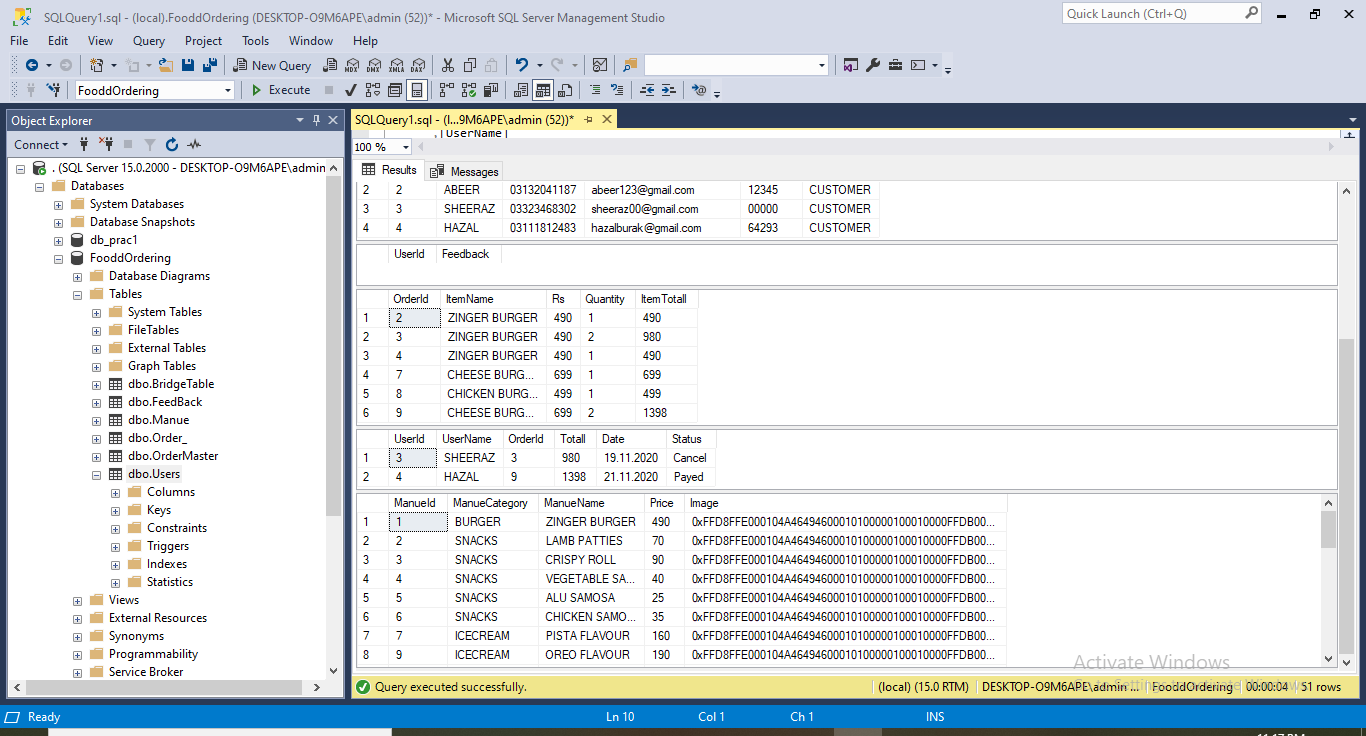
# Project Layout





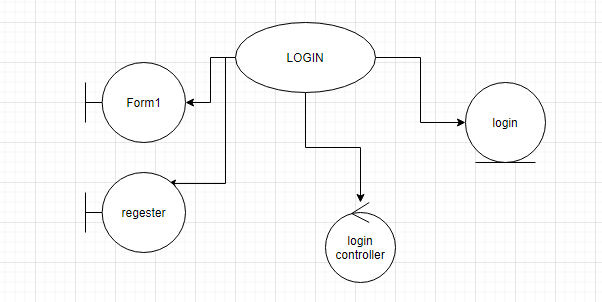
OTHER SCREENSHOTS ATTACHED IN WORKING OUTPUT



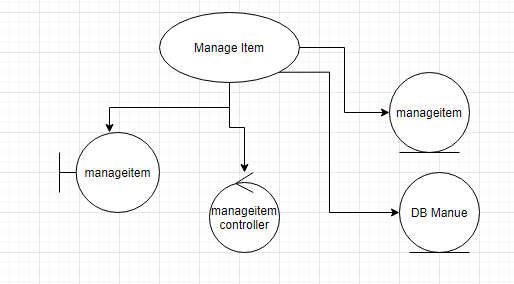


# Realization of Classes

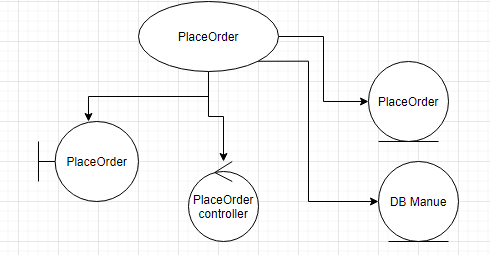
**Login**



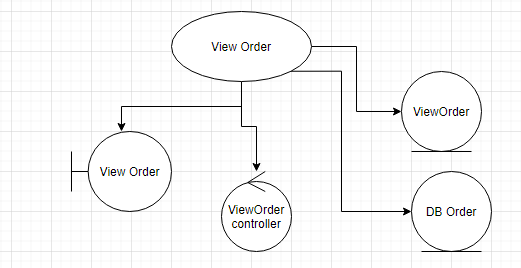
**Manage Item**



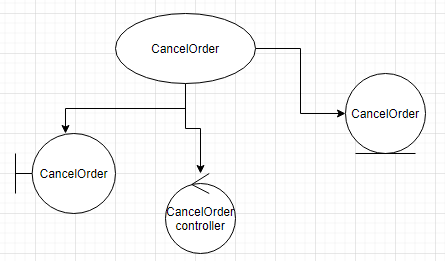
**Place Order**



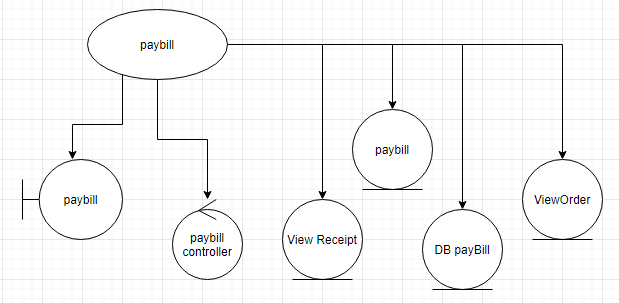
**View Order**



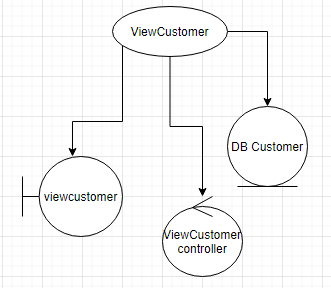
**Cancel Order**



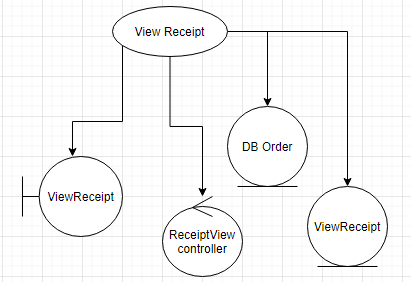
**Pay Bill**



**View Customer**

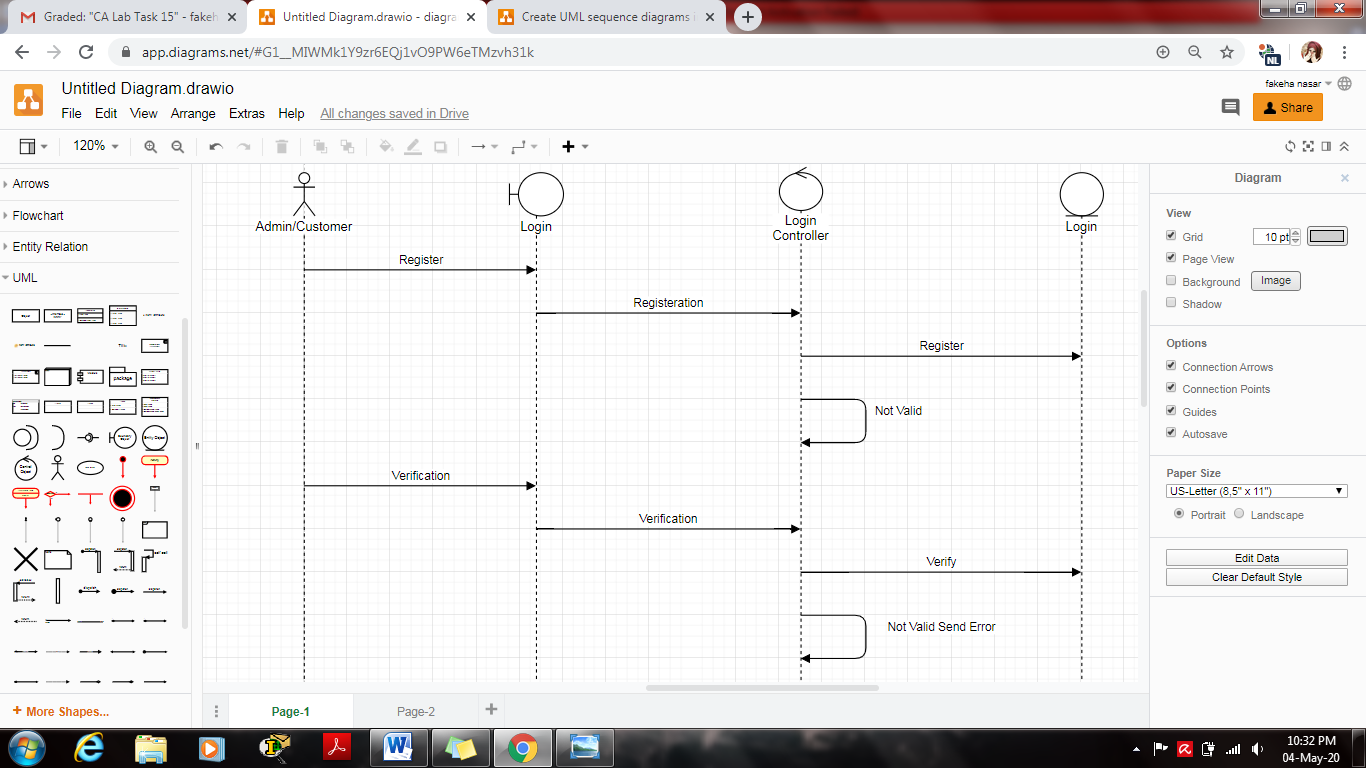


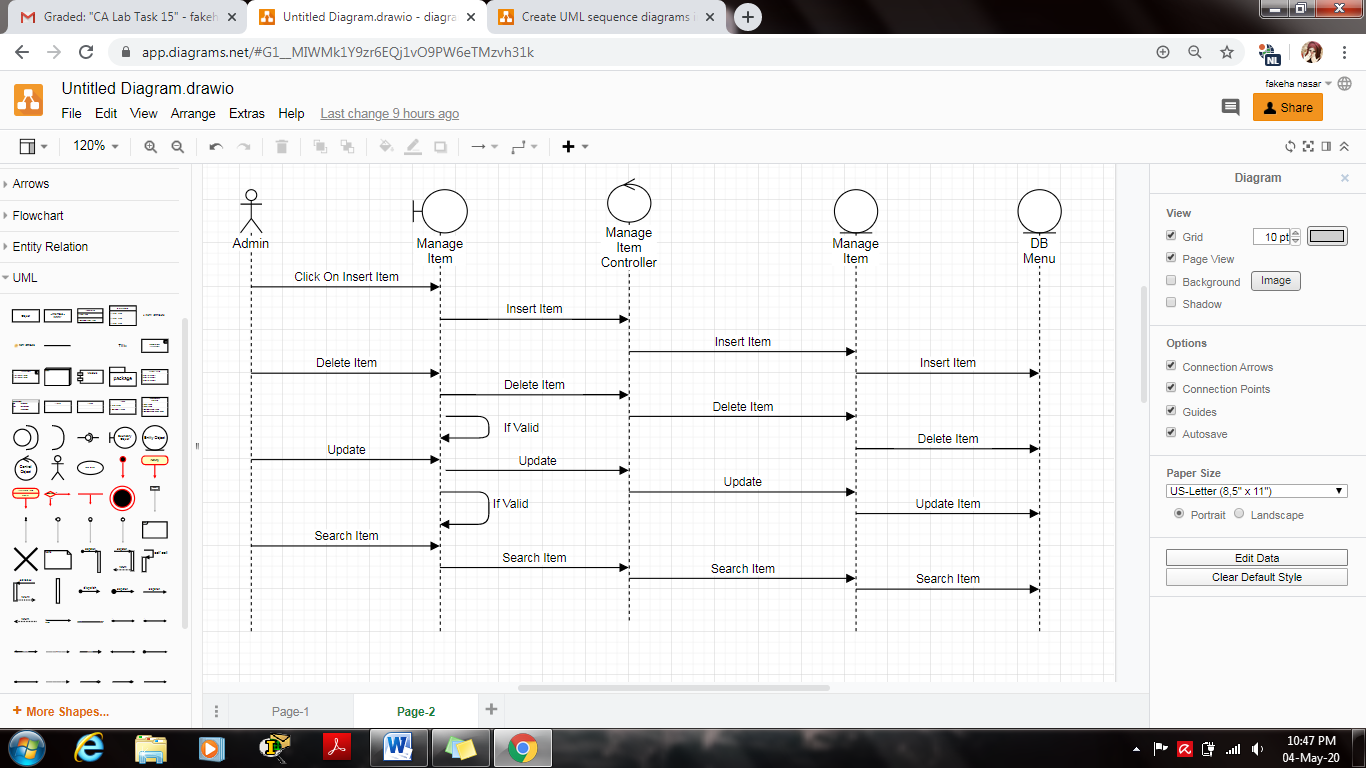
**View Receipt**

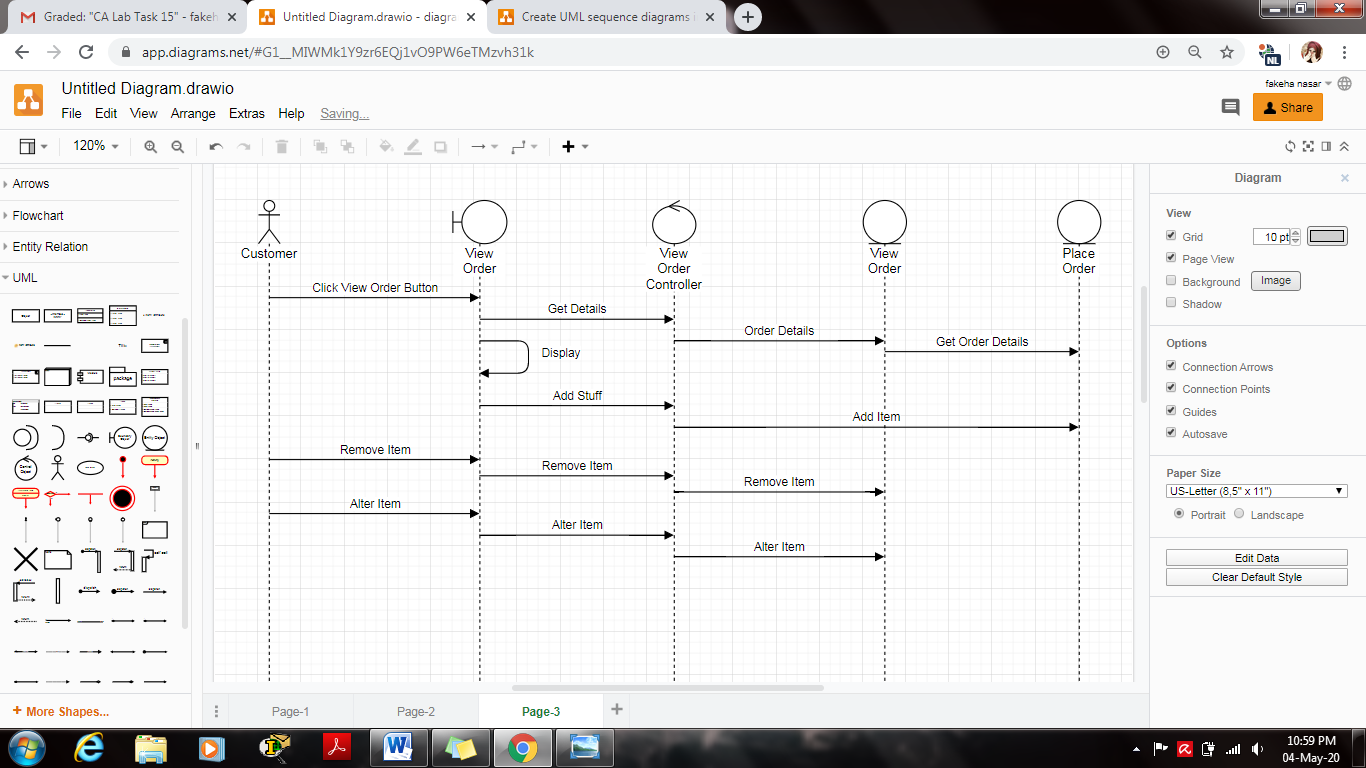


# **Sequence Diagrams**

**Login**

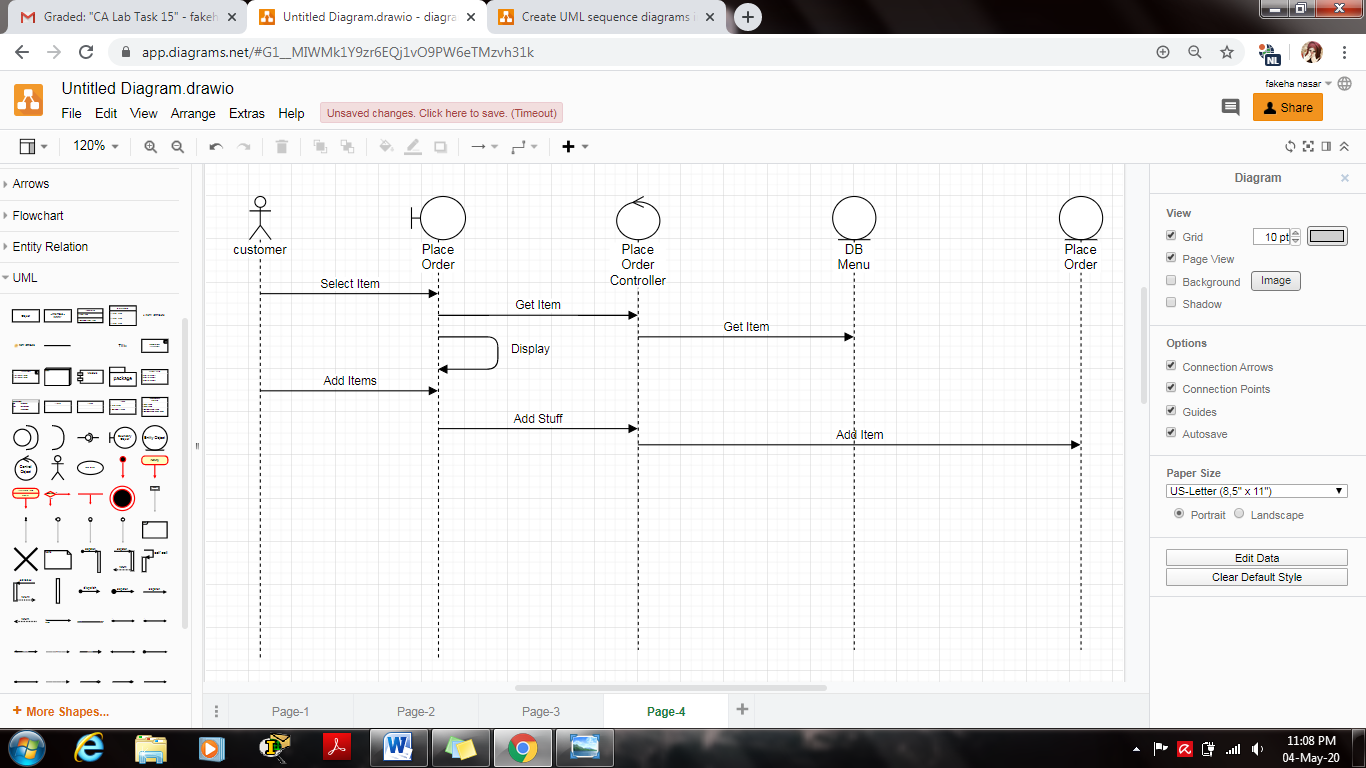


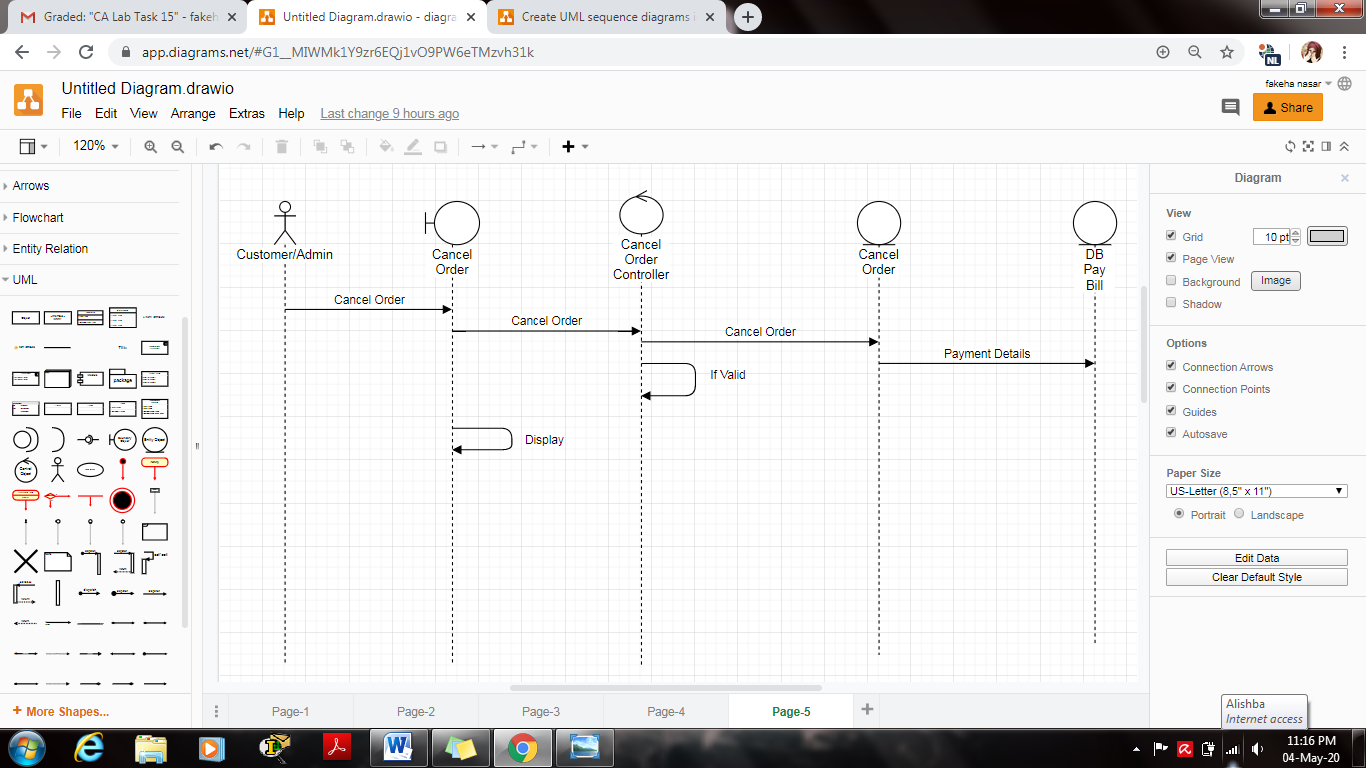
**Manage Item**

****

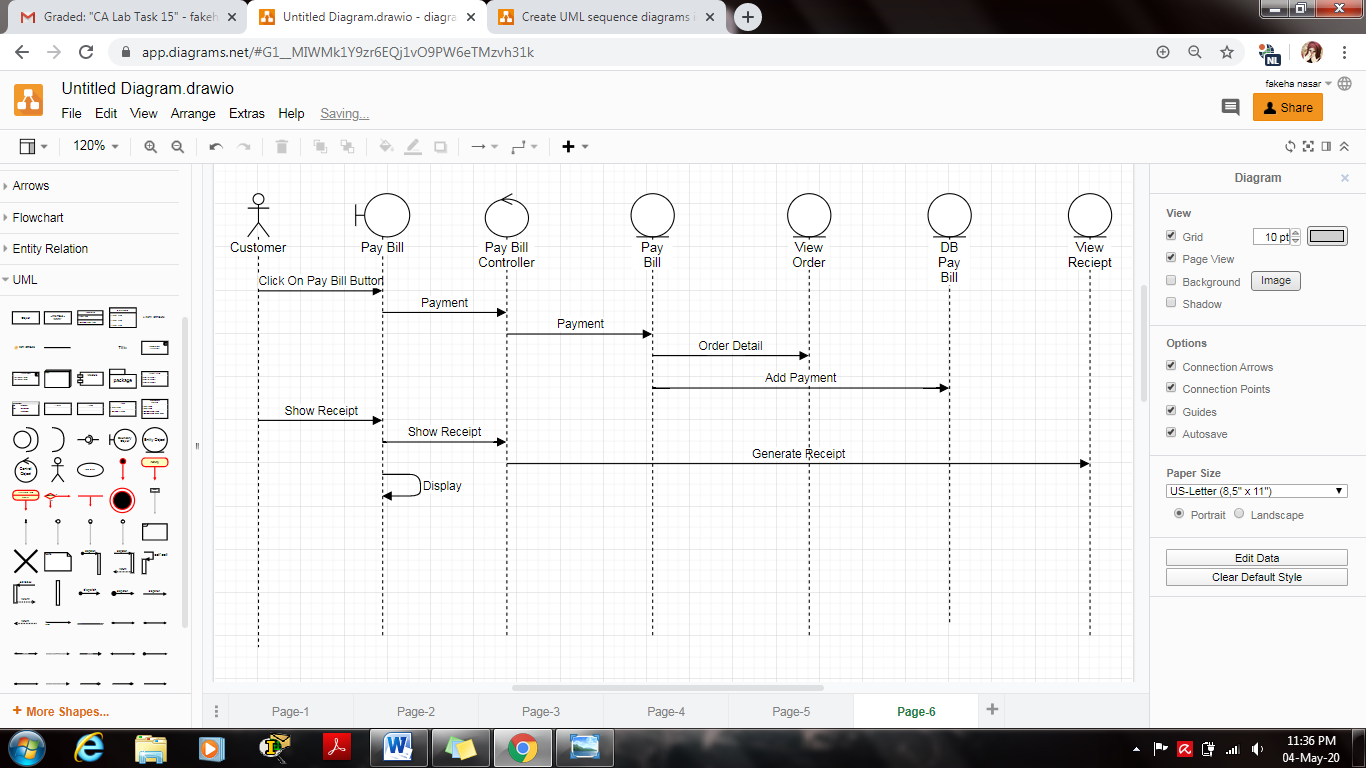
**View Order**

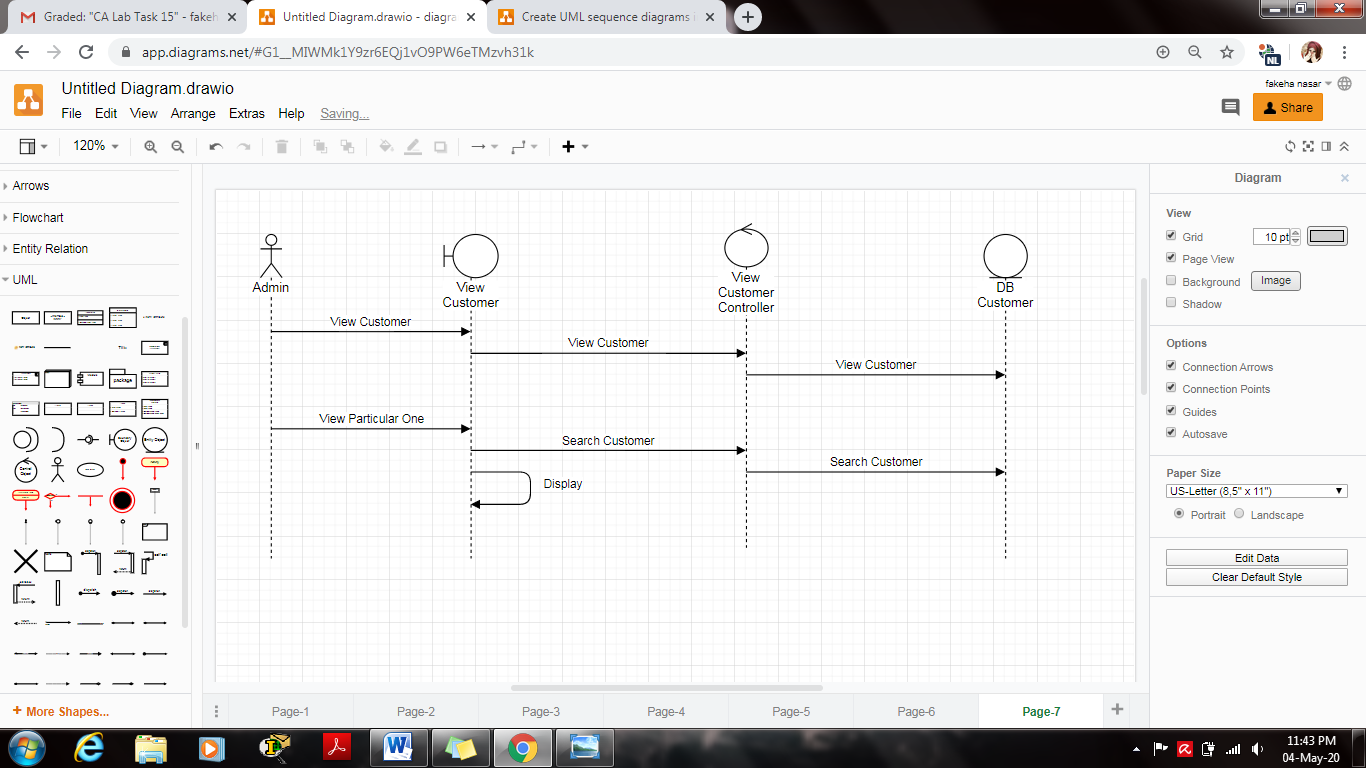
**Place Order**

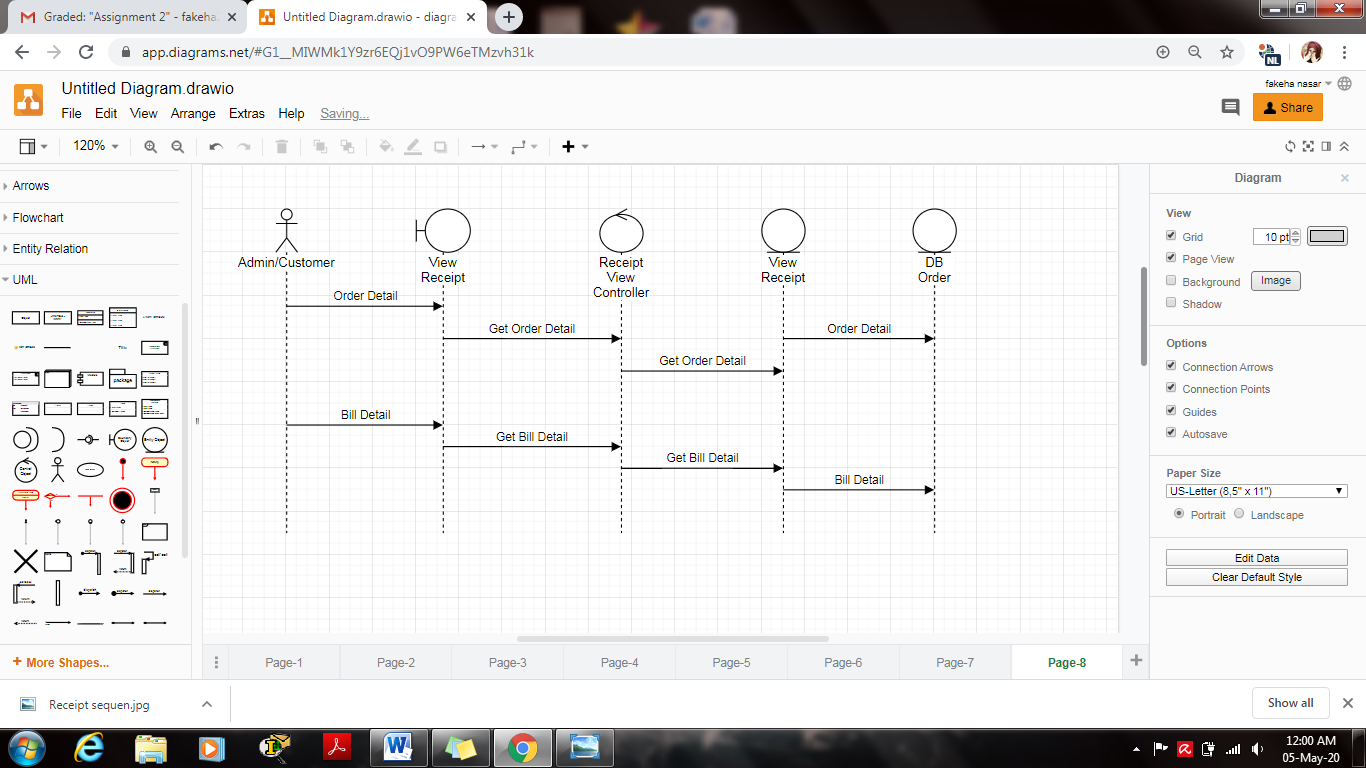


**Cancel Order**

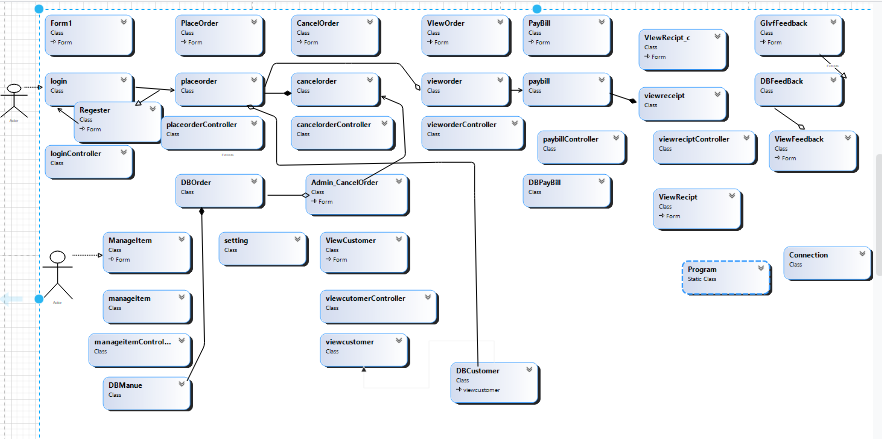
**Pay Bill**

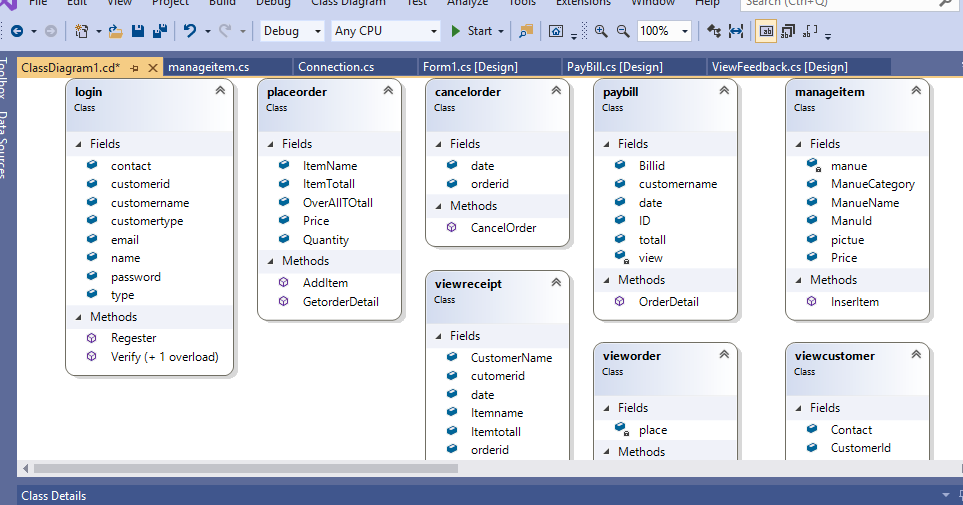


**View Customer**

**View Receipt**

# **Class Diagram**

  
  
1. place order view order forms a dependency  
2. place order and cancel order forms a composition  
3. place order and paybill is association whereas paybill and view receipt forms a dependency



# **Connection Class**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FoodOrdering\_ooad\_

{

class Connection

{

static SqlConnection sc;

public static SqlConnection get()

{

if (sc == null)

{

sc = new SqlConnection();

sc.ConnectionString = @"initial catalog=FooddOrdering;integrated security=true;data source=.";

sc.Open();

}

return sc;

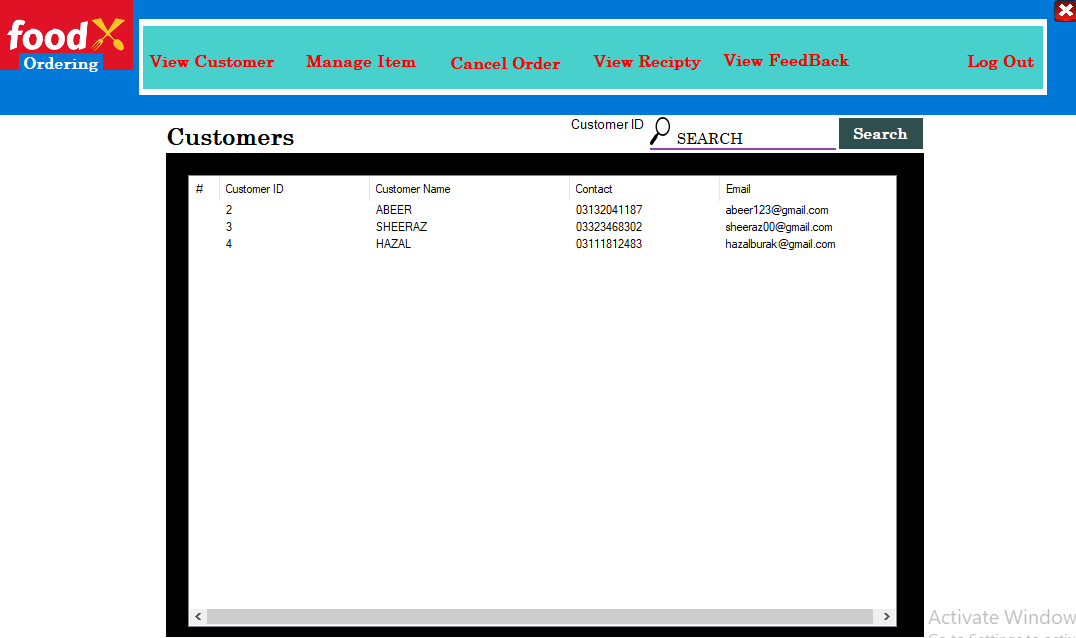
}

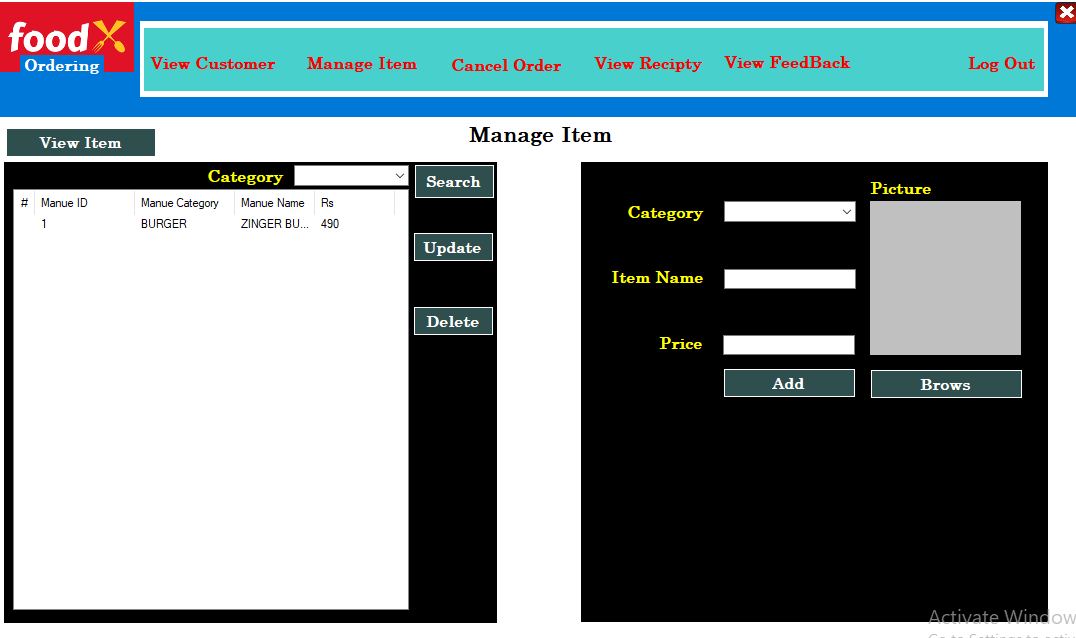
}

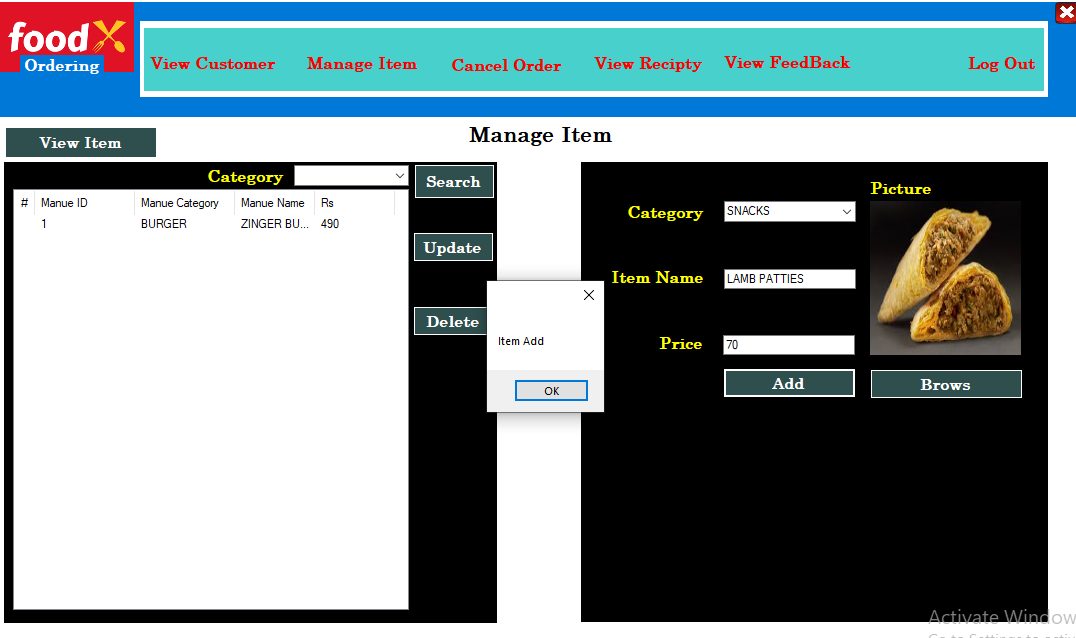
}

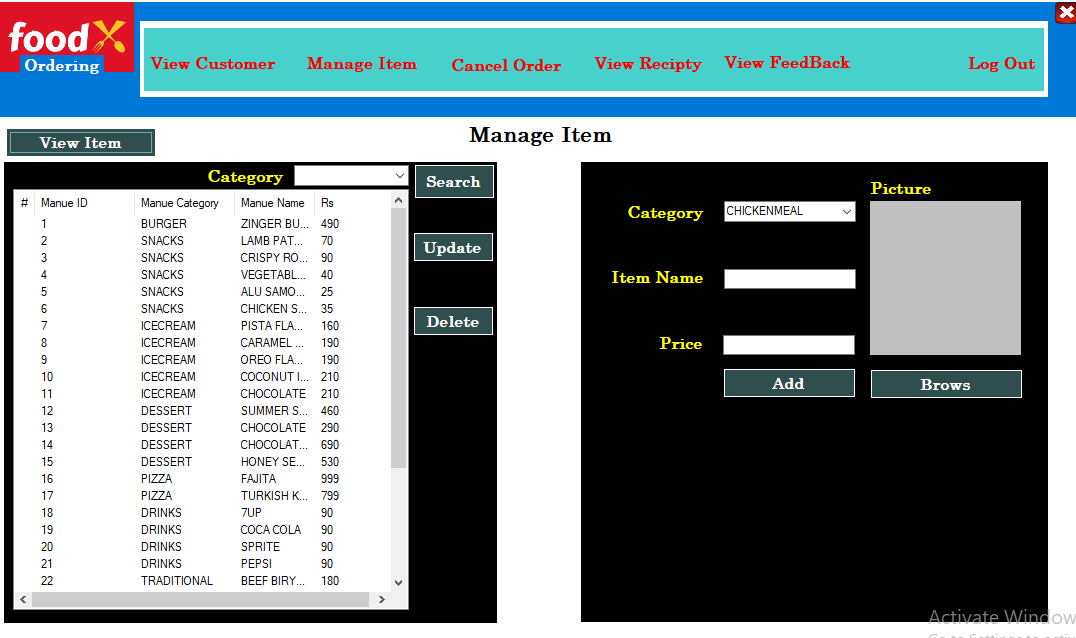
# **Working Output**

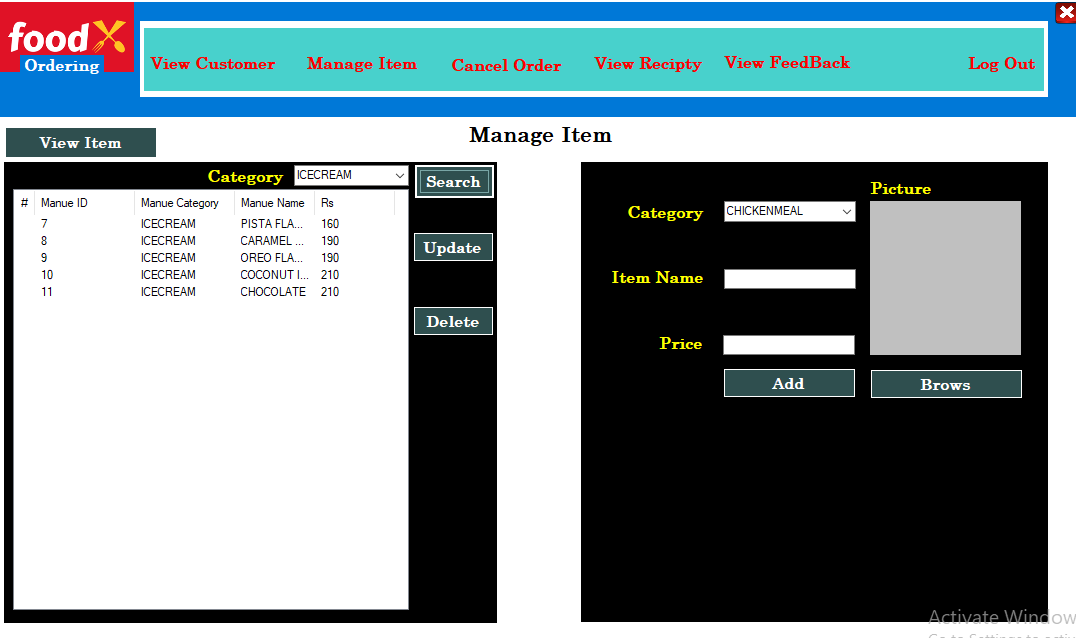
Admin End:

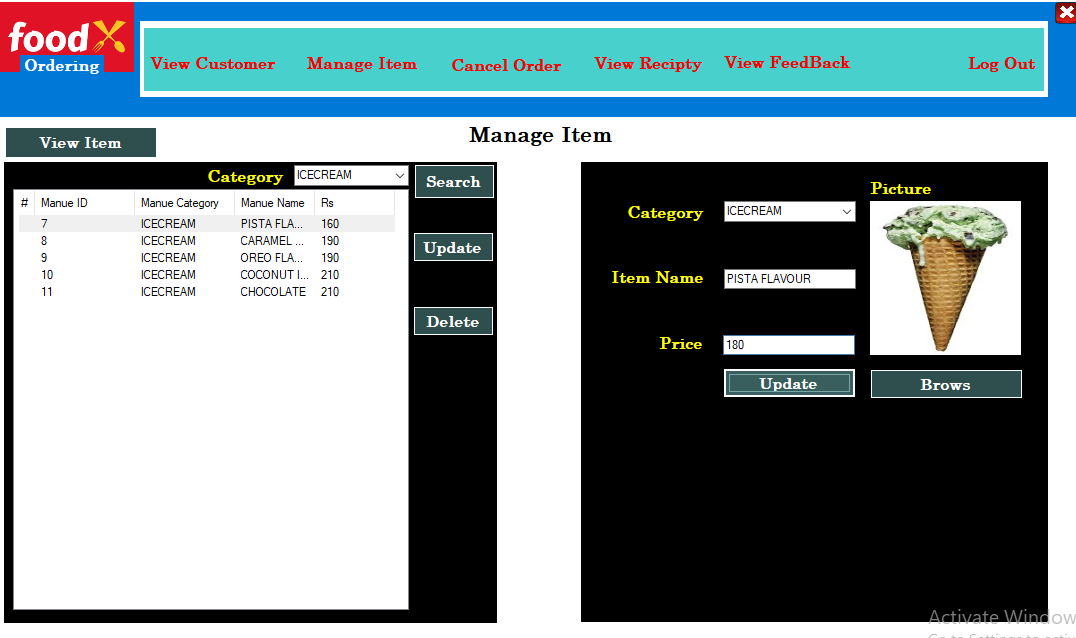


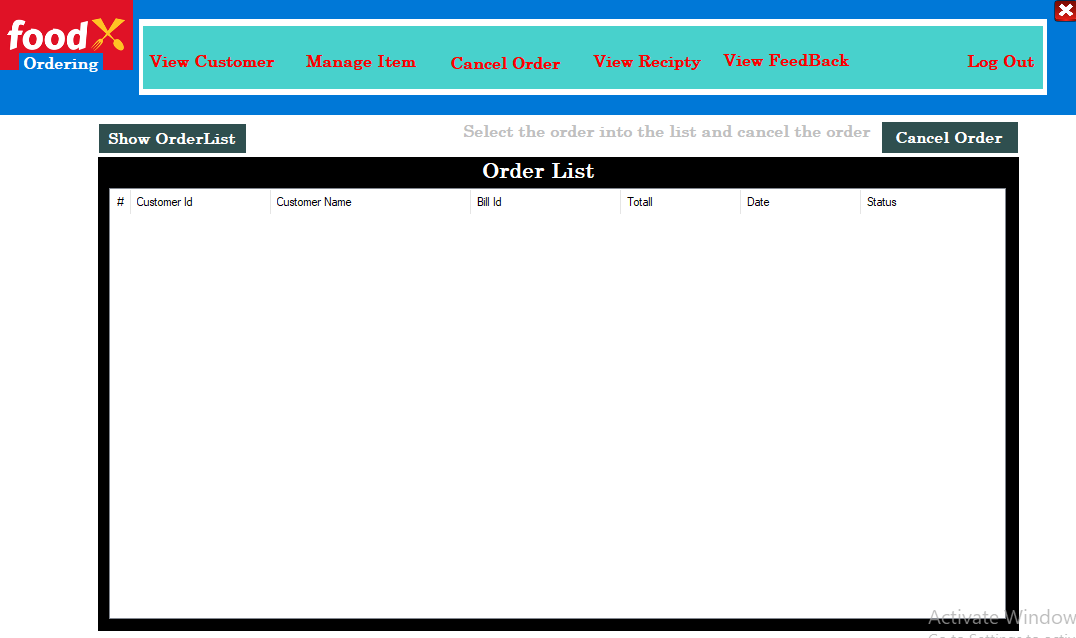


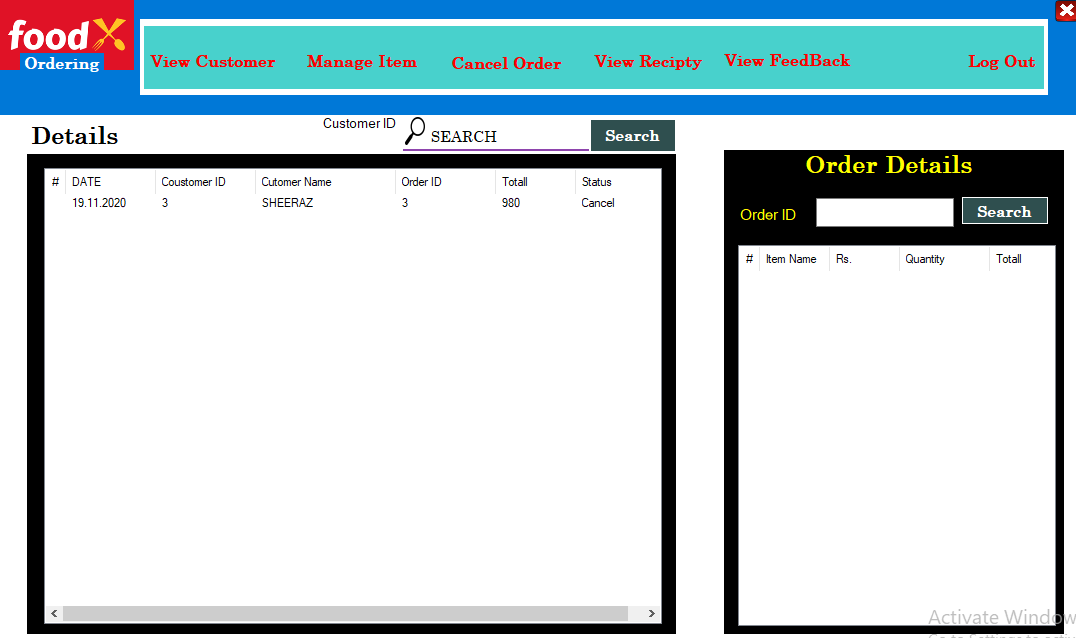


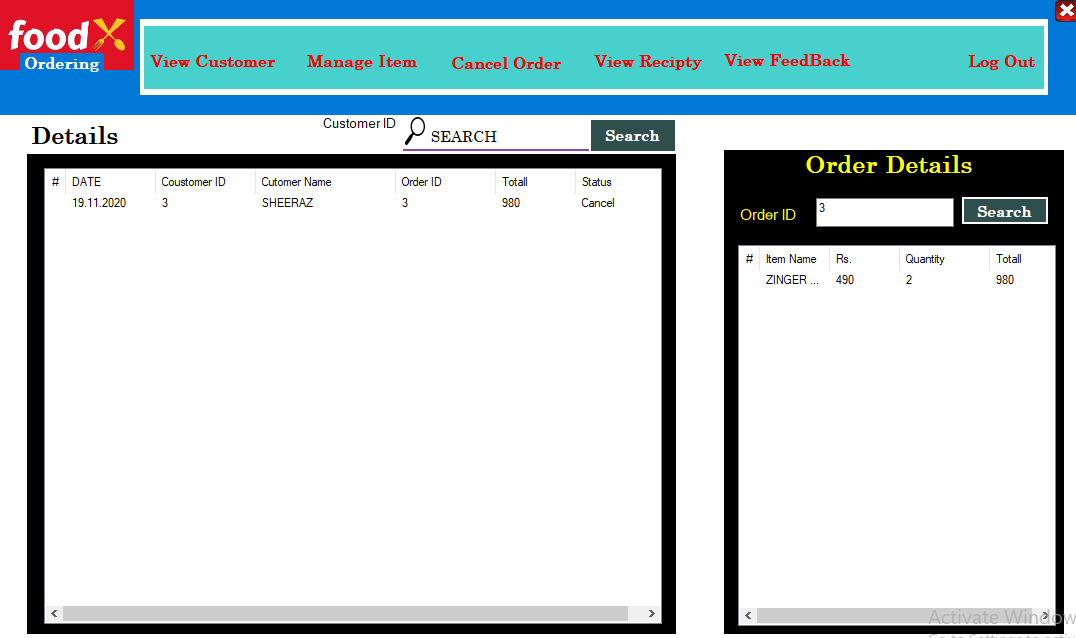


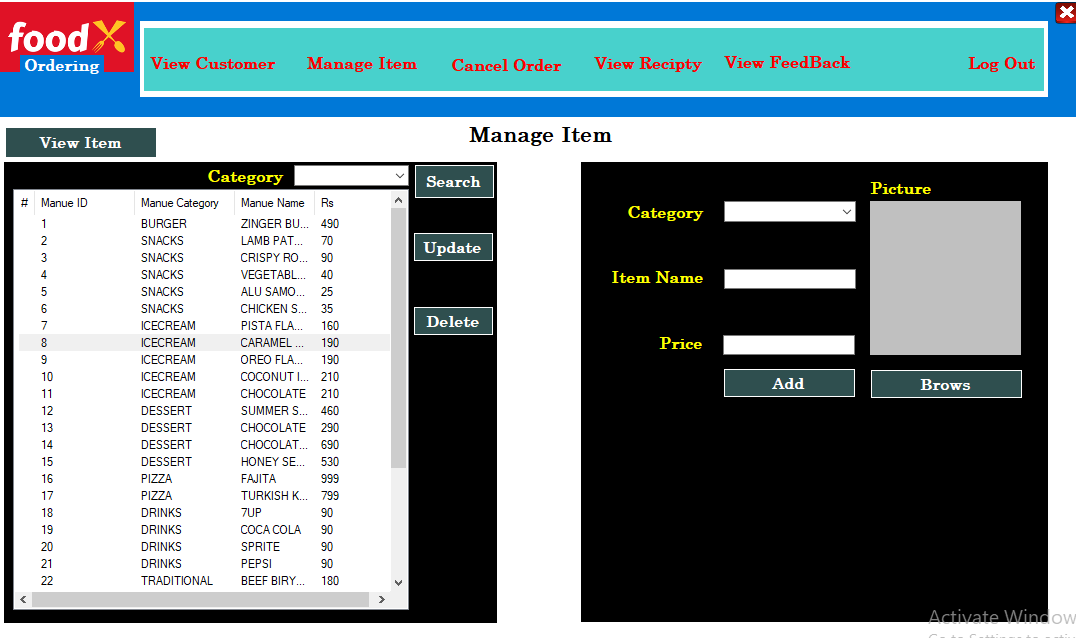


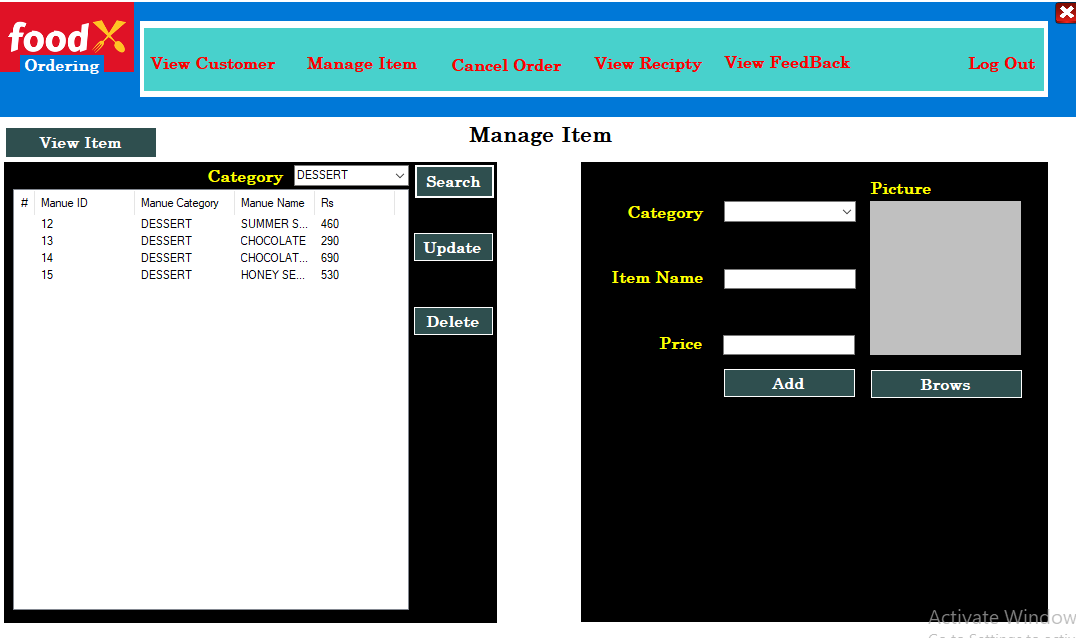


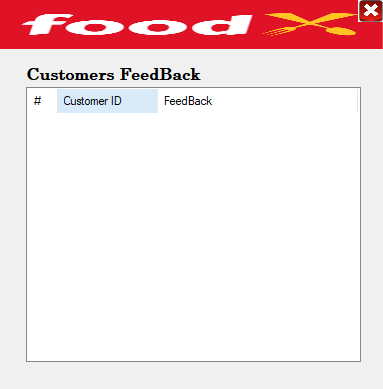




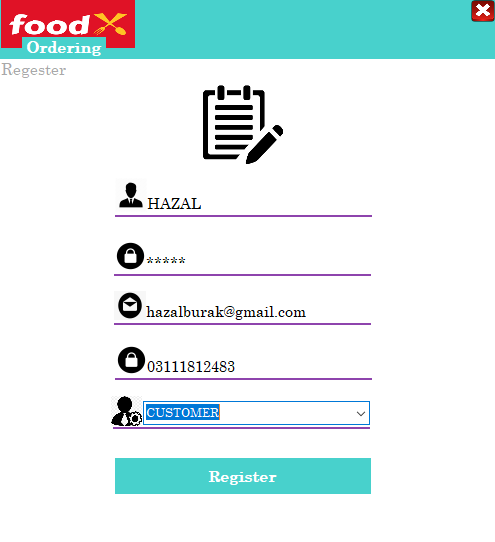


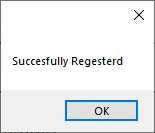


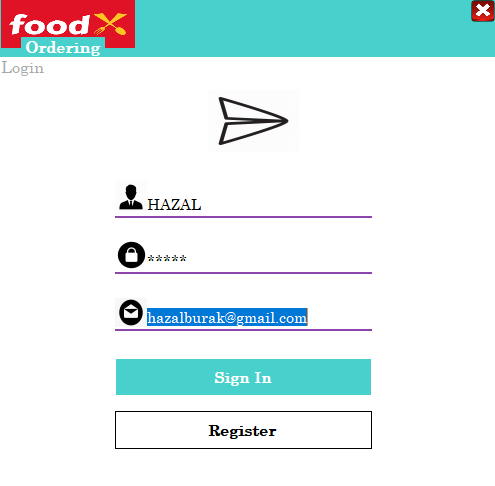




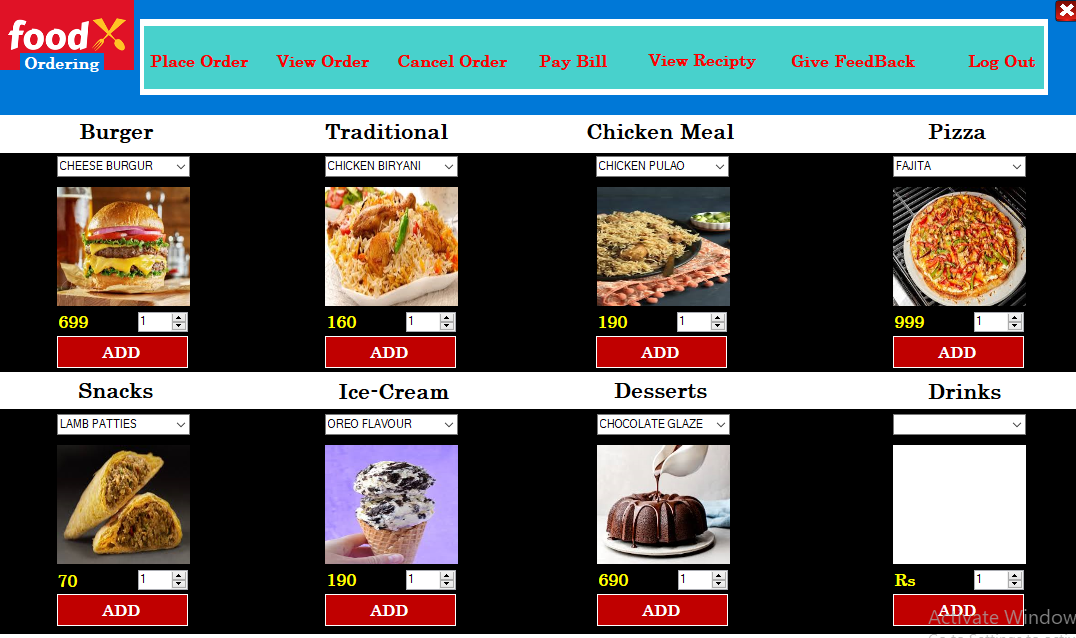
Customer End:

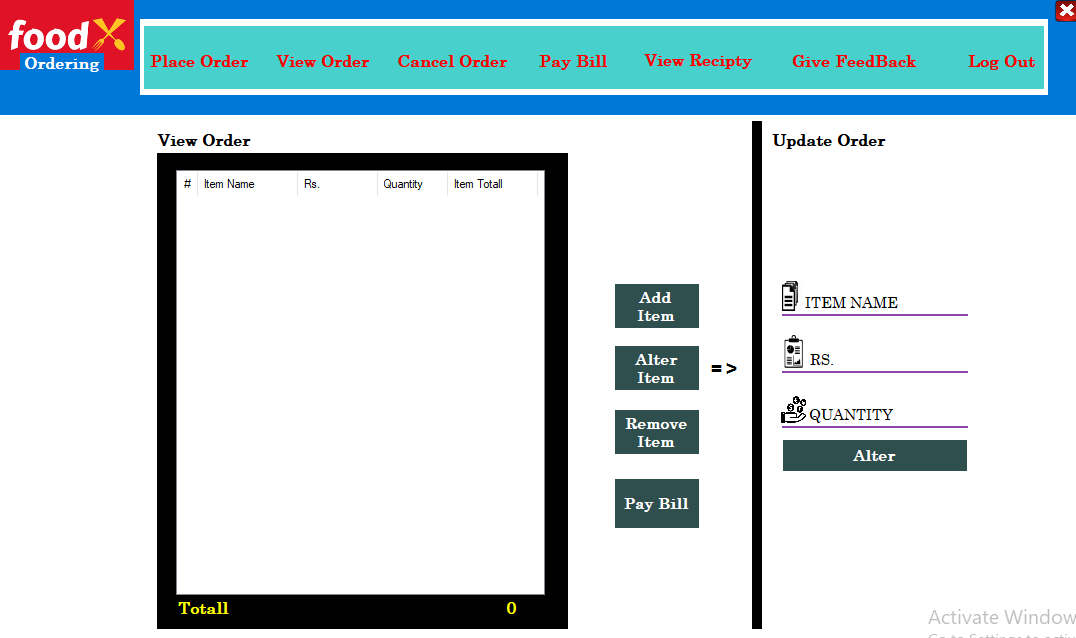


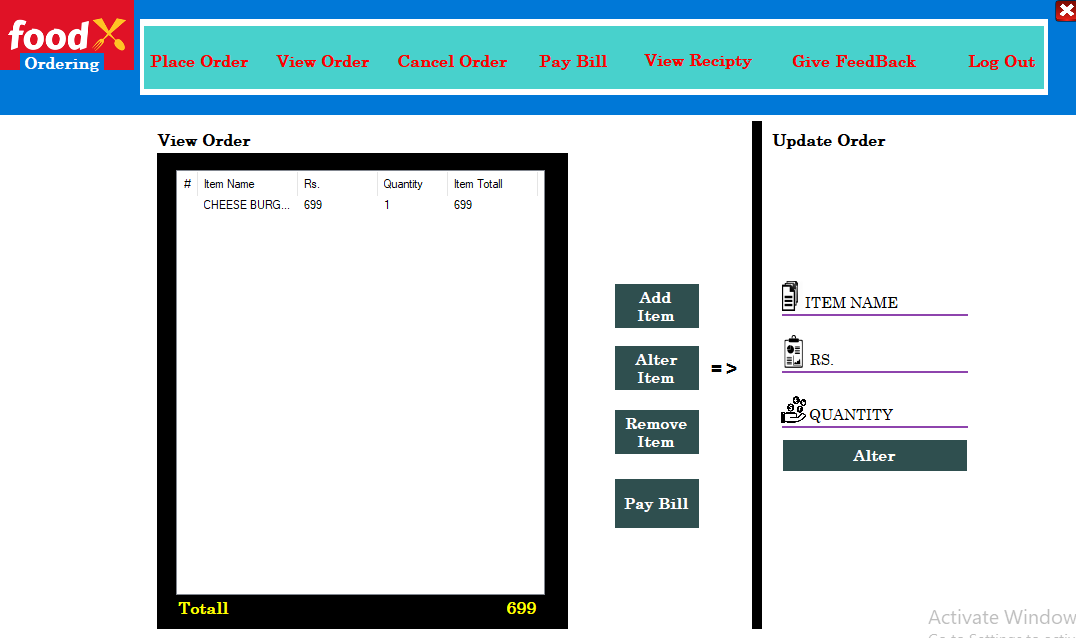


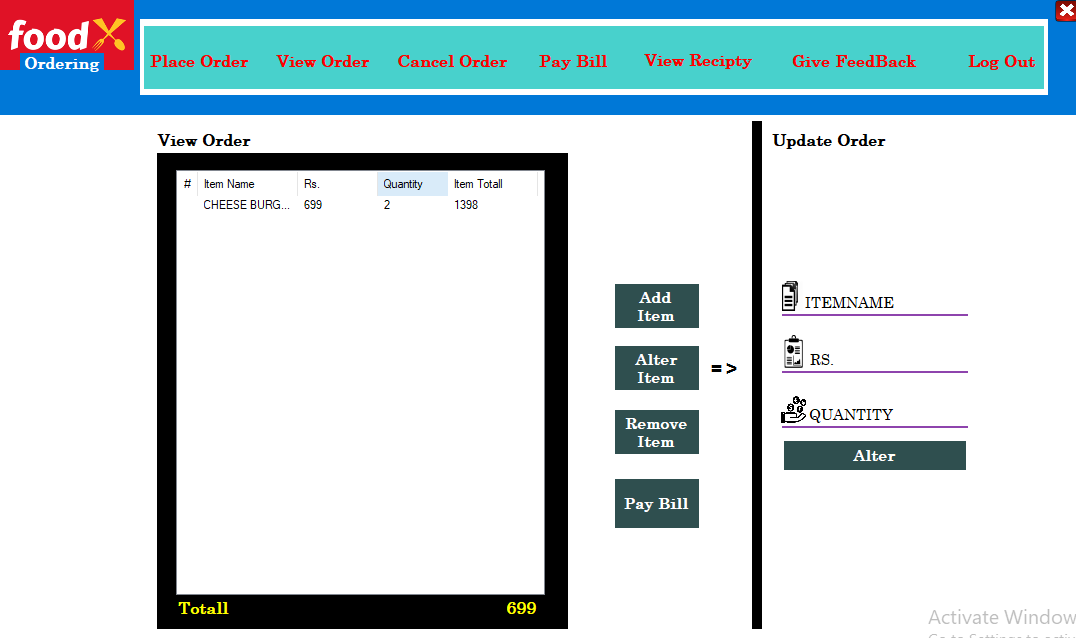


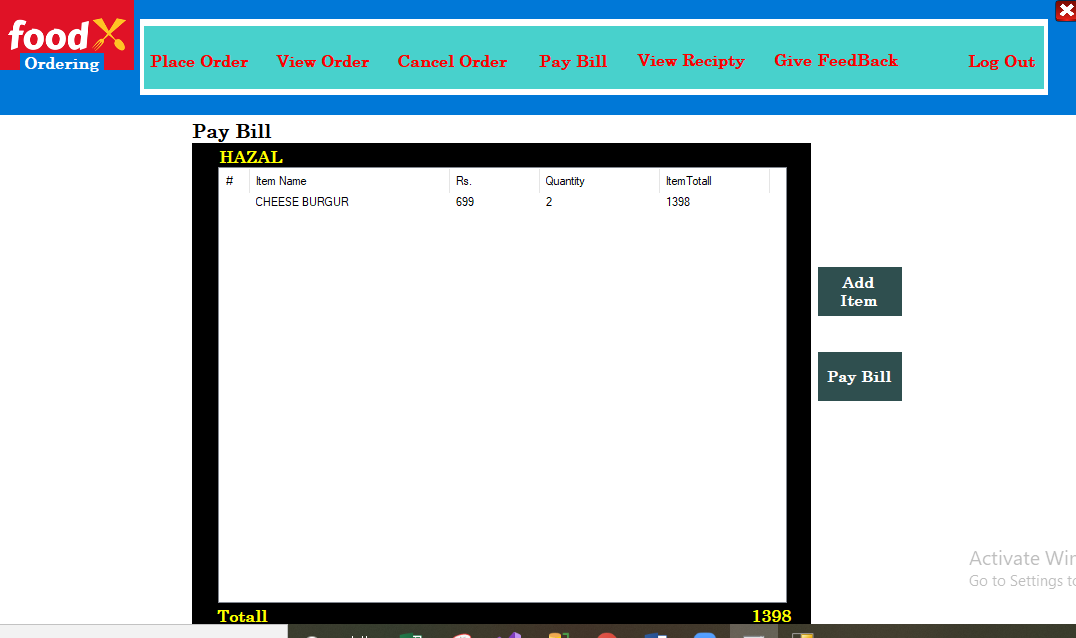


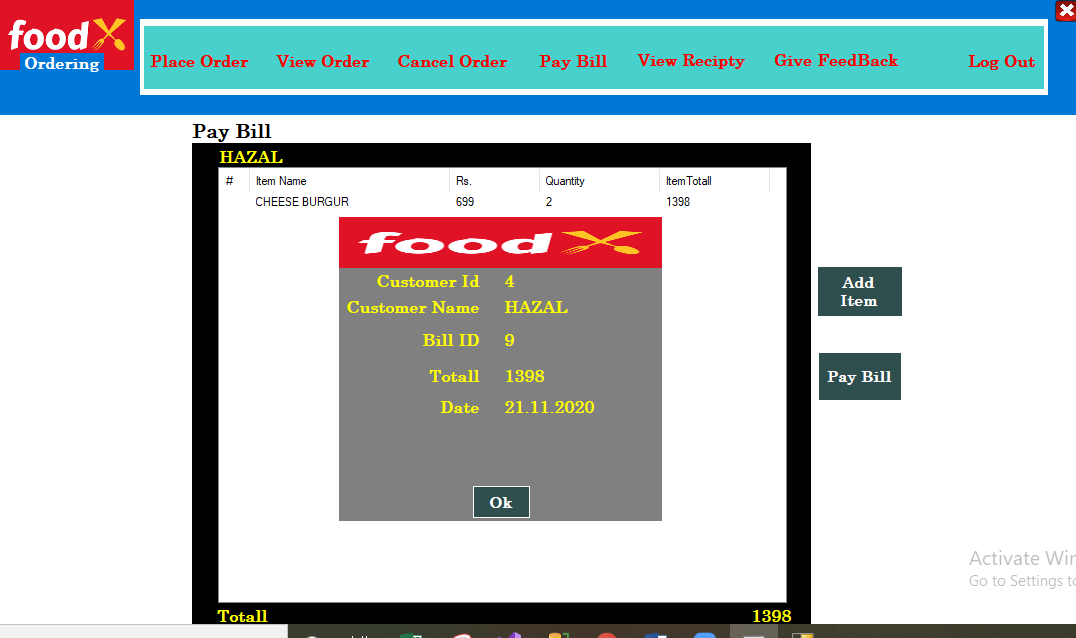












ADMIN:

