**Foodies Cuisine**



**Session 2023 - 2027**

**Submitted by:**

Muhammad Fahad Registration No:2023-CS-124

**Supervised by:**

Sir Laeeq Khan

**Course:**

OOP

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Contents

[ About Foodies Cuisine: 3](#_Toc11269)

[ Why Foodies Cuisine: 3](#_Toc10197)

[ User Types on Foodies Cuisine: 3](#_Toc13839)

[ Admin: 3](#_Toc26579)

[ Customer: 3](#_Toc22749)

[ Functional Requirements of Foodies Cuisine: 4](#_Toc24493)

[ Wireframes of Foodies Cuisine: 4](#_Toc13992)

[Main Menu: 5](#_Toc17383)

[Sign Up Menu: 5](#_Toc10310)

[Admin Menu: 6](#_Toc20162)

[Menu Card: 7](#_Toc14516)

[Add Dish: 7](#_Toc6682)

[Remove Dish: 8](#_Toc12774)

[Update Dish: 8](#_Toc10610)

[Employee Info: 9](#_Toc27950)

[Add Employee: 9](#_Toc11595)

[Remove Employee: 10](#_Toc27572)

[Update Employee: 10](#_Toc4654)

[Update Menu: 11](#_Toc8147)

[Update Name: 11](#_Toc16663)

[Update Contact Number: 12](#_Toc22116)

[Sales: 12](#_Toc25632)

[Customer Menu: 13](#_Toc26417)

[Make Order: 13](#_Toc22943)

[Bill Payment: 14](#_Toc20005)

[ Code: 14](#_Toc4369)

[Dish.BL: 14](#_Toc6913)

[Dish.DL(With DataBase): 15](#_Toc10379)

[Dish.DL(With FileHandling): 18](#_Toc21391)

[Dish.UI: 21](#_Toc5769)

[ CRC: 23](#_Toc1206)

# About Foodies Cuisine:

Welcome to Foodies Cuisine, it’s major usage to manage all the financial issues in the restaurant. It enables user to view all information about restaurant and easy to perform desired action. Owner can deal with three different branches at a time and keep check and balance of the restaurant.

## Why Foodies Cuisine:

Foodies Cuisine application provides different types of functionalities to users and an easy way to manage account. It analysis the data to check out profits and loss of each branch separately. Moreover, it also gives a graphical representation of ratio of profits of the branches.

# User Types on Foodies Cuisine:

This application provides four types of users with different functionalities for particular branches. Overall, all branches have same functionalities for same type of user. The hierarchy and functionality of user types is as under:

## Admin:

admin is same for all the branches of the restaurant. He is the only one to add manager for particular branch. Admin has many functionalities to keep an eye of his restaurant. He can view list of employs and also of customers. He can check expenditures and sales of the branch and thus, profit or loss. He can further manage his profile.

## Customer:

Customers can easily visit the restaurant through this application. He can has all information about menu card or prices on this application. HE can also get premier or student Memberships. Based on the membership, customer would get discounts on their orders. He can also reserve tables in the restaurant of different seats. He can view list of prices of reservations given by manager. Customer can also give feedbacks related to the particular branch to manager for betterment of the services.

# Functional Requirements of Foodies Cuisine:

Some of the functional requirements expected from Foodies Cuisine are as under:

|  |  |  |
| --- | --- | --- |
| ***User Type*** | ***Required Function to be Performed*** | ***Result of Action Performed*** |
| ***Admin*** | Menu Card | Show all dishes |
| Add Dish | Add New dish |
| Remove Dish | Remove dish |
| Update Dish | Update price of dish |
| Employ Info | List of Employees |
| Add Employee | Add New Employee |
| Remove Employee | Remove Employee |
| Upload Employee | Update Name or Contact Number of employee |
| Sales | Shows List of dishes ordered |
| Log out | Returns to the main menu. |
| ***Customer*** | Order | Give Order |
| Order History | Check previuos Orders. |
| Edit Profile | Can Edit its username and Password. |
| Feedback | Can give Feedback. |
| Log Out | Returns to main menu |

# Wireframes of Foodies Cuisine:

The following is the wireframe of Foodies Cuisine displayed in

Command Line Interface:

## Main Menu:

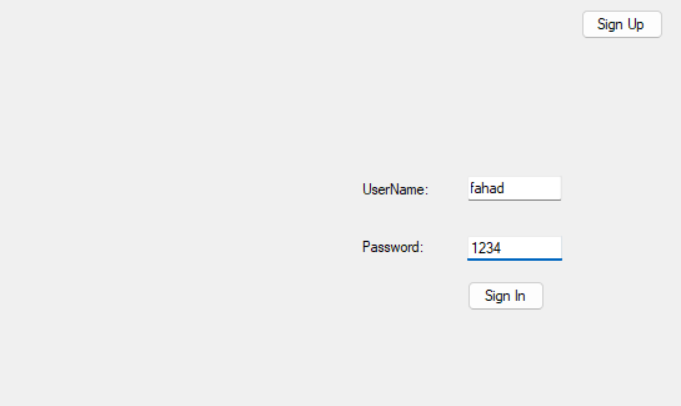


Figure 1MainMenu

## Sign Up Menu:

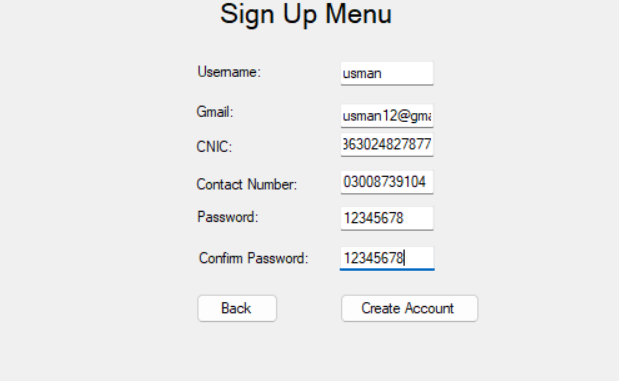


Figure 2Sign Up Menu

## Admin Menu:

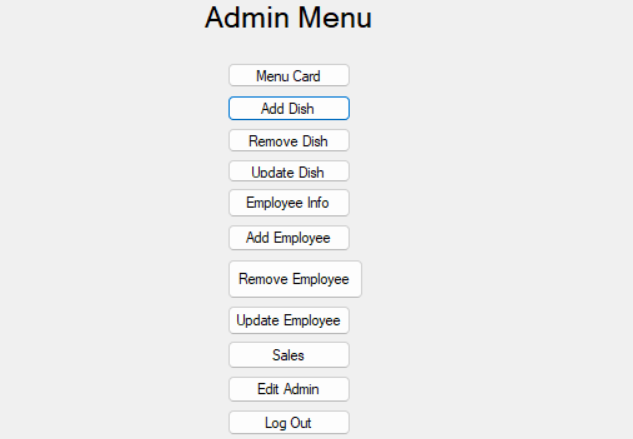


Figure 3Admin Menu

## Menu Card:

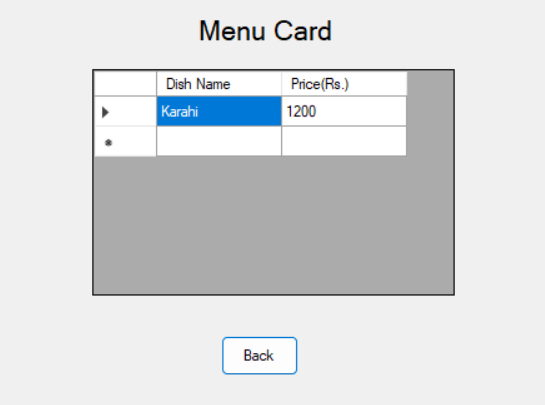


Figure 4MenuCard

## Add Dish:

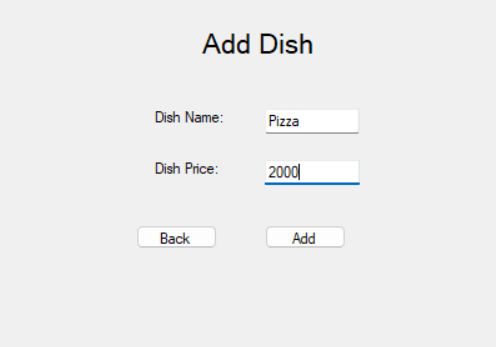


Figure 5Add Dish

## Remove Dish:

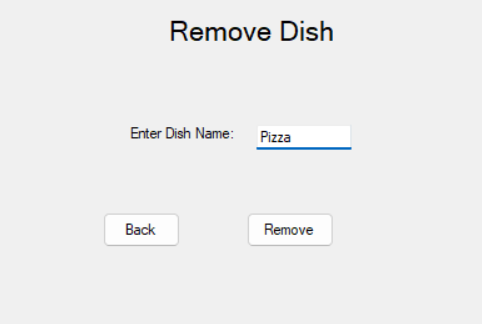


Figure 7RemoveDish

## Update Dish:

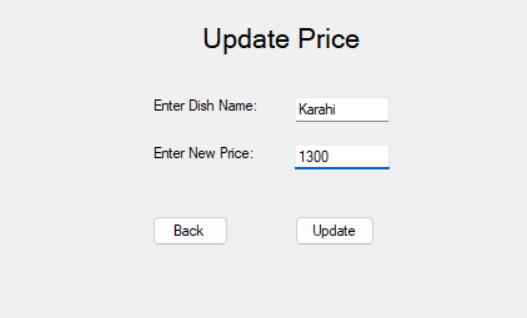


Figure 7 Update Dish

## Employee Info:

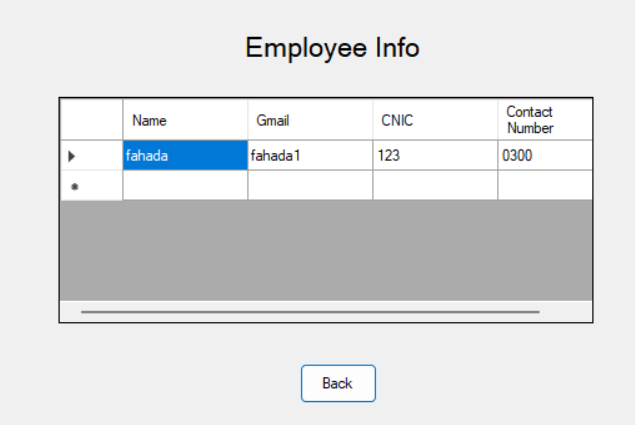


Figure 8 Employee Info

## Add Employee:

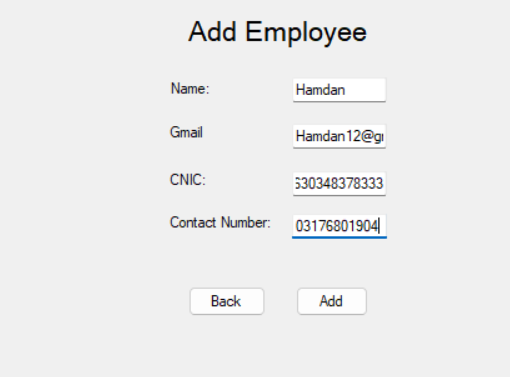


Figure 9 AddEmployee

## Remove Employee:

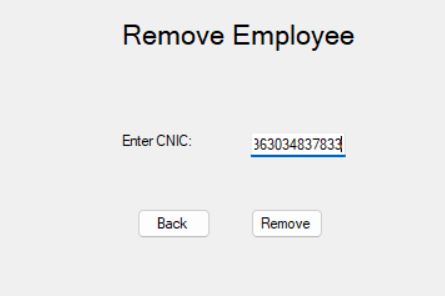


Figure 10 Remove Employee

## Update Employee:

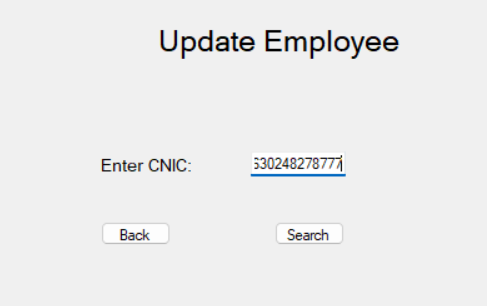


Figure 11 Update Employee

## Update Menu:

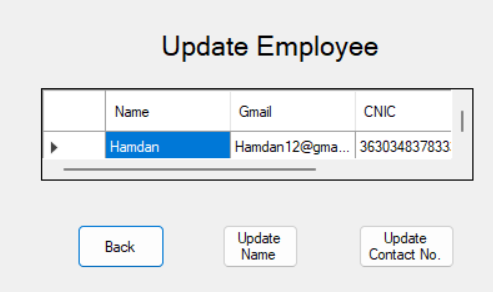


Figure 12 Update Menu

## Update Name:

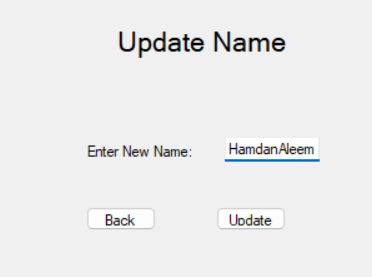


Figure 13 Update Name

## Update Contact Number:

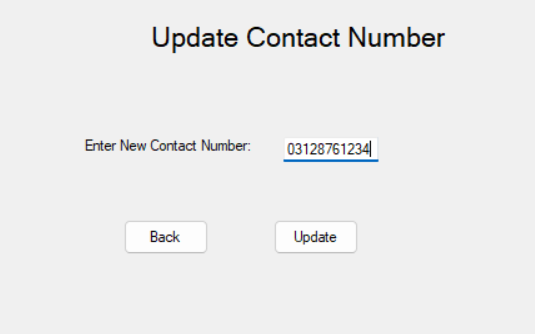


Figure 14 Update Name

## Sales:

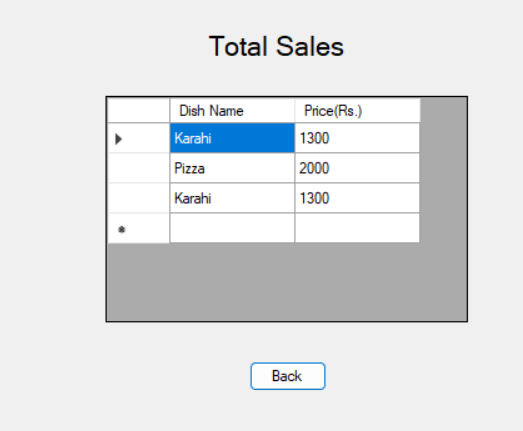


Figure 14Sales

## Customer Menu:

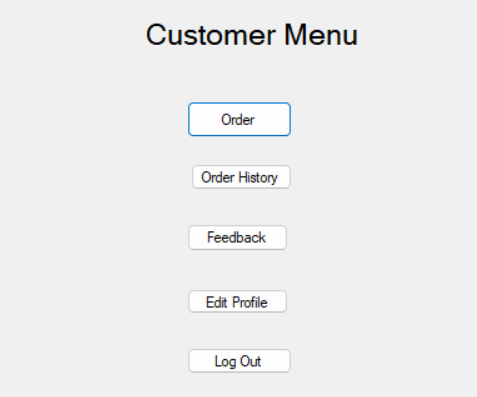


Figure 15CustomerMenu

## Make Order:

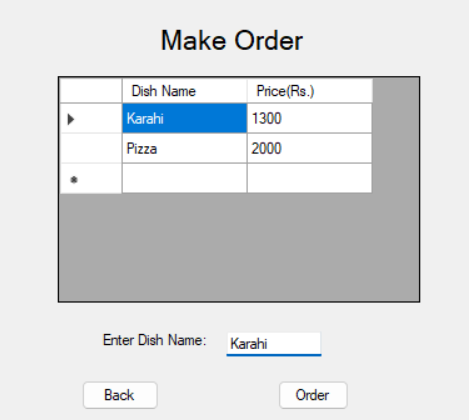


Figure 16 Make Order

## Bill Payment:

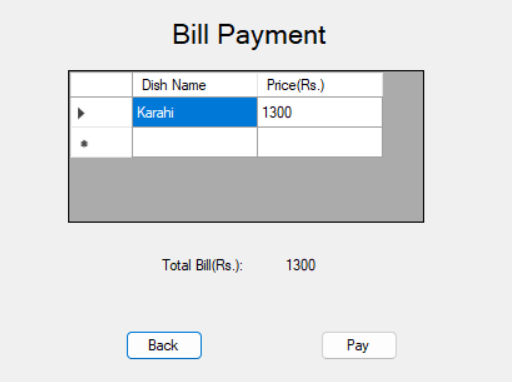


Figure 17BillPayent

# Code:

# Dish.BL:

public class Dish

{

private string Name;

private double Price;

public Dish() { }

public Dish(string Name) { this.Name = Name; }

public Dish (string Name,double Price)

{

this.Name = Name;

this.Price = Price;

}

public void SetName(string Name)

{

this.Name = Name;

}

public string GetName()

{

return this.Name;

}

public void SetPrice(double Price)

{

this.Price = Price;

}

public double GetPrice()

{

return this.Price;

}

}

# Dish.DL(With DataBase):

public class DishDL: IDishDL

{

public string ConnectionString = Utility.GetConnectionString();

public DishDL(string ConnectionString)

{

this.ConnectionString = ConnectionString;

}

public bool AddDish(Dish dish)

{

string query = String.Format("Insert into Dishlist(Name,Price)" + " values('{0}','{1}')",dish.GetName(), dish.GetPrice());

SqlConnection sqlConnection = new SqlConnection(ConnectionString);

sqlConnection.Open();

SqlCommand sql = new SqlCommand(query, sqlConnection);

int rows = sql.ExecuteNonQuery();

sqlConnection.Close();

if (rows > 0)

{

return true;

}

return false;

}

public List<Dish> GetDishList()

{

List<Dish> dishes =new List<Dish> ();

string query = "Select \* from Dishlist ";

SqlConnection sqlConnection = new SqlConnection(ConnectionString);

sqlConnection.Open();

SqlCommand sqlCommand = new SqlCommand(query, sqlConnection);

SqlDataReader reader = sqlCommand.ExecuteReader();

while (reader.Read())

{

Dish dish = new Dish(Convert.ToString(reader["Name"]), Convert.ToInt64(reader["Price"]));

dishes.Add(dish);

}

reader.Close();

sqlConnection.Close();

return dishes;

}

public bool CheckDish(Dish dish)

{

List<Dish> dishs = GetDishList();

foreach (Dish dishes in dishs)

{

if (dishes.GetName() == dish.GetName())

{

return false;

}

}

return true;

}

public bool RemoveDish(Dish dish)

{

string query = String.Format("Delete from Dishlist where Name ='{0}'", dish.GetName());

SqlConnection sqlConnection = new SqlConnection(ConnectionString);

sqlConnection.Open();

SqlCommand sql = new SqlCommand(query, sqlConnection);

int rows = sql.ExecuteNonQuery();

sqlConnection.Close();

if (rows > 0)

{

return true;

}

return false;

}

public bool UpdateDish(Dish dish)

{

string query = String.Format("Update Dishlist SET Price = '{1}' WHERE Name = '{0}'", dish.GetName(),dish.GetPrice());

SqlConnection sqlConnection = new SqlConnection(ConnectionString);

sqlConnection.Open();

SqlCommand sql = new SqlCommand(query, sqlConnection);

int rows = sql.ExecuteNonQuery();

sqlConnection.Close();

if (rows > 0)

{

return true;

}

return false;

}

}

# Dish.DL(With FileHandling):

public class DishFH: IDishDL

{

private static List<Dish> dishes = new List<Dish>();

private string Path;

public DishFH(string Path)

{

this.Path = Path;

ReadDishes(Path);

}

public bool AddDish(Dish dish)

{

dishes.Add(dish);

StreamWriter streamWriter = new StreamWriter(Path, false);

foreach (Dish dish1 in dishes)

{

streamWriter.WriteLine(dish1.GetName() + "," + dish1.GetPrice());

streamWriter.Flush();

}

streamWriter.Close();

return true;

}

public List<Dish> GetDishList()

{

return dishes;

}

private bool ReadDishes(string Path)

{

StreamReader streamReader = new StreamReader(Path);

if (File.Exists(Path))

{

string record;

while ((record = streamReader.ReadLine()) != null)

{

string Name = GetField(record, 1);

string Price = GetField(record, 2);

Dish dish = new Dish(Name, double.Parse(Price));

dishes.Add(dish);

}

streamReader.Close();

}

return true;

}

public bool RemoveDish(Dish dish)

{

foreach (Dish dish1 in dishes)

{

if (dish.GetName() == dish1.GetName())

{

dishes.Remove(dish1);

StreamWriter streamWriter = new StreamWriter(Path, false);

streamWriter.WriteLine(dish1.GetName() + "," + dish1.GetPrice());

streamWriter.Flush();

streamWriter.Close();

return true;

}

}

return false;

}

public bool CheckDish(Dish dish)

{

foreach (Dish dishes1 in dishes)

{

if (dishes1.GetName() == dish.GetName())

{

return false;

}

}

return true;

}

public bool UpdateDish(Dish dish)

{

foreach(Dish dish1 in dishes)

{

if (dish.GetName() == dish1.GetName())

{

dish1.SetPrice(dish.GetPrice());

}

}

StreamWriter streamWriter = new StreamWriter(Path, false);

foreach (Dish dish1 in dishes)

{

streamWriter.WriteLine(dish1.GetName() + "," + dish1.GetPrice());

streamWriter.Flush();

}

streamWriter.Close();

return true;

}

private string GetField(string record, int field)

{

int commaCount = 1;

string result = "";

for (int x = 0; x < record.Count(); x++)

{

if (record[x] == ',')

{

commaCount++;

}

else if (commaCount == field)

{

result += record[x];

}

}

return result;

}

}

# Dish.UI:

internal class DishUI

{

public static void PrintDishs(List<Dish> dishs )

{

Console.WriteLine("Dish Name Prices(Rs.)");

Console.WriteLine();

foreach (Dish dish in dishs)

{

Console.WriteLine(dish.GetName()+ " "+dish.GetPrice() );

}

Console.WriteLine();

ConsoleUtility.ContinueStatement();

}

public static Dish AddDishUI()

{

Console.Write("Enter Dish Name: ");

string Name=Console.ReadLine();

Console.Write("Enter Dish Price: ");

double Price=double.Parse(Console.ReadLine());

ConsoleUtility.ContinueStatement();

Dish dish = new Dish(Name, Price);

return dish;

}

public static Dish RemoveDishUI()

{

Console.Write("Enter Dish Name: ");

string Name=Console.ReadLine();

ConsoleUtility.ContinueStatement();

Dish dish=new Dish(Name);

return dish;

}

public static Dish UpdateDishUI()

{

Console.Write("Enter Dish Name: ");

string Name= Console.ReadLine();

Console.Write("Enter New Price: ");

double Price = double.Parse(Console.ReadLine());

ConsoleUtility.ContinueStatement();

Dish dish = new Dish(Name,Price);

return dish;

}

}

# CRC:

