



Islington college
(इस्लिंग्टन कलेज)

Module Code & Module Title

CS6004NI - Application Development

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2020-21 Autumn

Student Name: Salina Budhathoki

London Met ID: NP01CP4A180167

College ID: 18029966

Assignment Due Date: 22th January 2021

Assignment Submission Date: 22th January 2021

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Contents

1.	Introduction.....	1
1.1.	Overview of assignment.....	1
1.2.	Tools used.....	1
2.	Background	2
2.1.	User Manual.....	2
2.2.	Software Architecture Diagram	8
2.3.	Class Diagram.....	9
2.4.	Class Description	10
2.5.	Method Description	15
2.6.	Data Structure and Algorithm Description	18
3.	Testing.....	21
4.	Conclusion.....	34
5.	Bibliography.....	35
6.	References	36
7.	Appendix.....	37

Table of figure

Figure 1: The working program (login form)	2
Figure 2: Report Form	3
Figure 3: Customer Feedback form.....	3
Figure 4: Filling details of Customer	4
Figure 5: Submitting the feedback form.....	4
Figure 6: Graph form without data.....	5
Figure 7: Graph View of selected data	5
Figure 8: Add of Criteria Parking	6
Figure 9: Criteria Name added in feedback form.....	6
Figure 10: Logout	7
Figure 11: Main page	7
Figure 12: System Architecture	8
Figure 13: Class Diagram of Rating System	9
Figure 14: Form1; Login.....	10
Figure 15: Form2; Feedback	11
Figure 16: Form3; Dashboard	12
Figure 17: Form4; Graph.....	13
Figure 18: Form5; Details.....	14
Figure 19: Bubble Sort Algorithm	19
Figure 20: Flow Chart.....	20
Figure 21: Home Page of system	21
Figure 22: Submitting incorrect username & password	22
Figure 23: Submitting only password	23
Figure 24: Submitting password only	24
Figure 25: Customer Data is Added	25
Figure 26: Submitting string type in contact	26
Figure 27: Report View.....	27
Figure 28: Graph of Order Accuracy	28
Figure 29: Graph of Food Quality	29
Figure 30: Graph of Staff Friendliness.....	29
Figure 31: Criteria Added	30
Figure 32: Criteria Added in feedback form.....	31
Figure 33: Clicking the logout	32
Figure 34: After clicking logout	33

Table of table

Table 1: Method Description of Login Form	15
Table 2: Method Description of Feedback Form	15
Table 3: Method Description of Dashboard Form.....	16
Table 4: Method Description of Graph Form	16
Table 5: Method Description of Details Form	17
Table 6: Testing 1.....	21
Table 7: Testing 2.....	22
Table 8: Testing 3.....	23
Table 9: Testing 4.....	25
Table 10: Testing 5.....	26
Table 11: Testing 6.....	27
Table 12: Testing 7.....	28
Table 13: Testing 8.....	30
Table 14: Testing 9.....	31
Table 15: Testing 10.....	32

Acknowledgement

Firstly, I would like to express my sincere gratitude towards the module leader of the following module 'Application Development', Mr. Dhurba Sen and Tutor of the Module, Mr. Sushil Sapkota for their continuous support towards my study, research and motivating me overcome the problems that I faced while doing this project. Their guidance was very crucial in the completion of this assignment and it would not have been complete by the deadline without their help.

Besides my supervisors, I would like to thank my college for providing study resource and materials which has been very helpful for our academic performance and also would like to convey my gratitude for the teachers and other staff members of our college for their constant evaluation and support.

Abstract

This report explains the use of C#, Visual Studio and .NET Framework to create a desktop application “Customer Rating System”. The application input the information and review of customer and displays them in a table inside the form and also generates a weekly report based on the course.

This report also describes the use of a sorting algorithm “Bubble Sort” with its algorithm and flowchart. The report is divided into six sections: Introduction: where the program overview is written and tools used are explained; Background: where the user manual along with the class diagram, description, method description and algorithm description are done; Testing: where the program functions are tested with details; Conclusion: where the personal review is written; Reference & Bibliography: where the research links and sources are shown; and Appendix: where the functional program's code is written.

1. Introduction

1.1. Overview of assignment

This project was assigned to us by our module leader for the module application development. It is an individual assignment which accounts for 30% of our total module grades. The primary task is to develop a desktop application which will be used to review customer's feedback and generate report. This application keeps track of the customer's feedback report which includes: Customer Name, Contact info, email, along with some functionalities that will be discussed in the report. The program is developed to meet the needs of restaurant as well as the requirement of this coursework. The program also lets the user sort the details according to date.

1.2. Tools used

The assessment entailed the use C#, Visual Studio and .NET framework for the development of the desktop application.

I. C# (Programming Language)

C# is a general object-oriented programming (OOP) language for networking and Web development. As a common language infrastructure (CLI) language, C# is defined. It is a simple, modern, general-purpose programming language oriented towards objects generated by Microsoft and running on the .NET Platform. It is used to build web, desktop and mobile app software, along with game creation and so on. ([techopedia, 2020](#))

II. Visual Studio (IDE)

Visual Studio is an integrated development platform created by Microsoft and is commonly used for the development of both web and mobile computer programs and applications. Its features and other built-in instruments make it very reliable for development. ([Microsoft , 2021](#))

III. .NET (Framework)

.NET is an open source, free, cross-platform developer platform for the development of several different types of applications. This software framework was developed by Microsoft and runs on Microsoft Windows in particular. For various languages, such as Visual Basic, C# and C++, it supports an object-oriented programming model. Its language interoperability feature allows several languages to be accepted and code written in other languages can be used. ([Microsoft, 2021](#))

2. Background

2.1. User Manual

Customer Feedback System is a desktop application that keeps record of the review given by the customer. Upon launching the application, a window application will appear as shown in fig 1. The instruction guide shown below will help with the better understanding about the application.

a. About

The image below represents my application which consists of text fields, picture box, checkbox, charts and buttons with report as per the requirement of coursework.

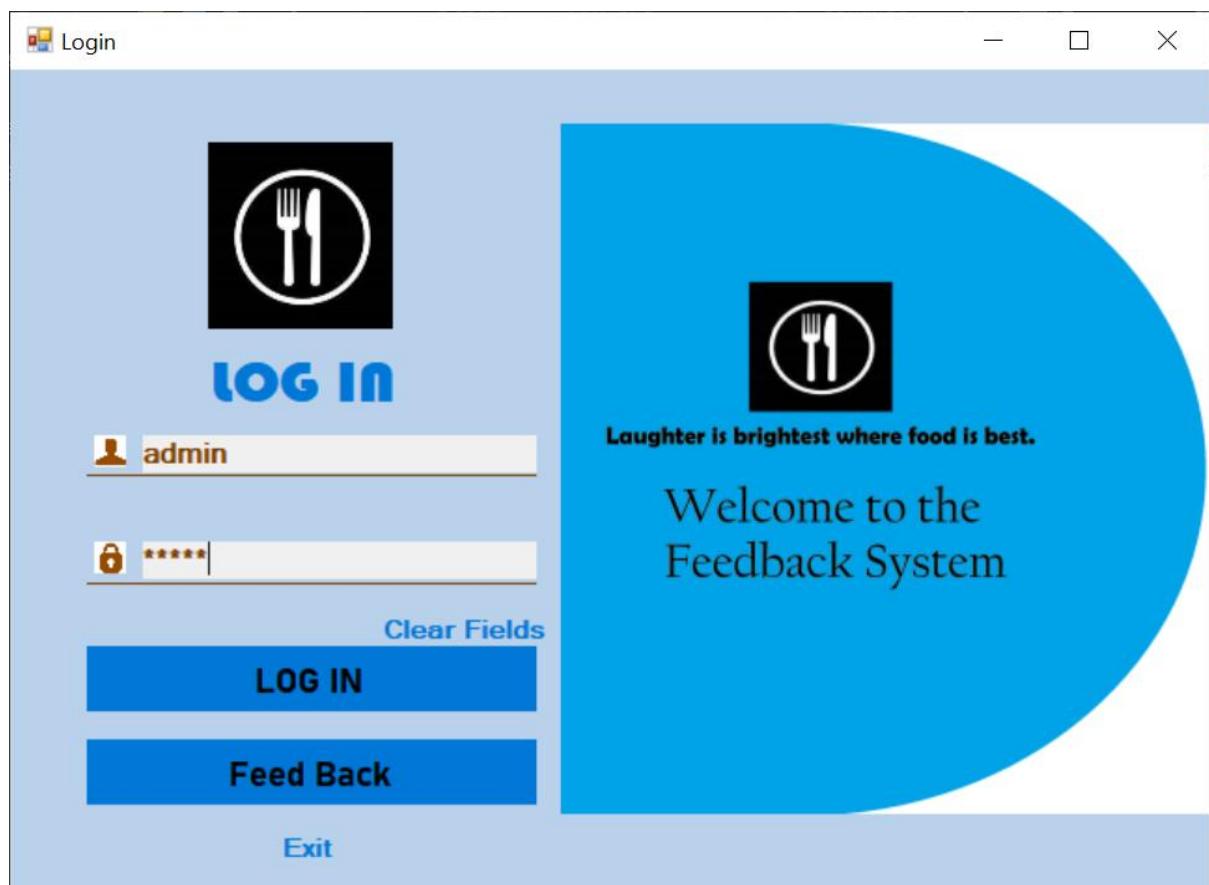


Figure 1: The working program (login form)

The screenshot shows a software interface with a blue sidebar on the left containing menu items: Dashboard, Report, Graph, Add Details, and Log Out. The main area is titled "View Report" and contains a large, empty gray rectangular box. In the top right corner of this box is a blue button labeled "View Report". The window has standard minimize, maximize, and close buttons at the top right.

Figure 2: Report Form

The screenshot shows a feedback form titled "Send us your feedback !". It features a "Welcome" logo and a "Get In Touch !" section with input fields for Customer Name, Contact Number, Email, and Suggestion. To the right is a table for rating various criteria. The table has columns for CriteriaName (Food Quality, Staff Friendliness, Cleanliness, Order Accuracy, Restaurant Ambia, Value For Money, Parking) and rows for rating levels (Excellent, Good, Average, Dissatisfied). At the bottom are "Submit" and "Return" buttons.

CriteriaName	Excellent	Good	Average	Dissatisfied
Food Quality	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Staff Friendliness	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Cleanliness	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Order Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Restaurant Ambia	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Value For Money	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Parking	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 3: Customer Feedback form

b. Input of Customer feedback form

For filling the form with appropriate data and click Submit button.

CriteriaName	Excellent	Good	Average	Dissatisfied
Food Quality	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Staff Friendliness	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Cleanliness	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Order Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Restaurant Ambia	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Value For Money	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 4: Filling details of Customer

	Name	Phone No	Email	Message	Date	Time	FoodQuality
▶	Shrijana Kc	9847823456	shrijanakc@gmail...	It was good	22/01/2021	10:16	3
▶	Bidhan Pant	9851234578	bidhanpant88@g...	It was a good ex...	22/01/2021	10:26	1

Figure 5: Submitting the feedback form

Once clicked, a message will appear on the screen: "Data Added" and the data will be added in the admin panel of View Report table grid.

c. Selecting the Graph

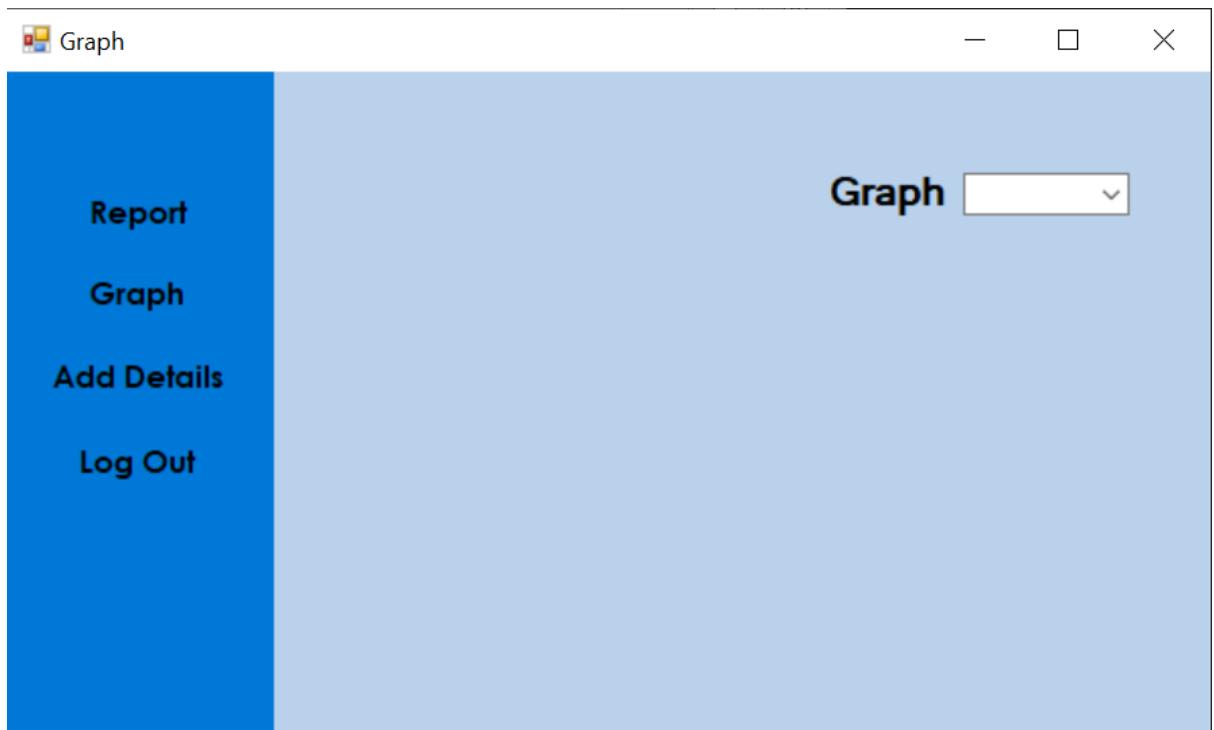


Figure 6: Graph form without data

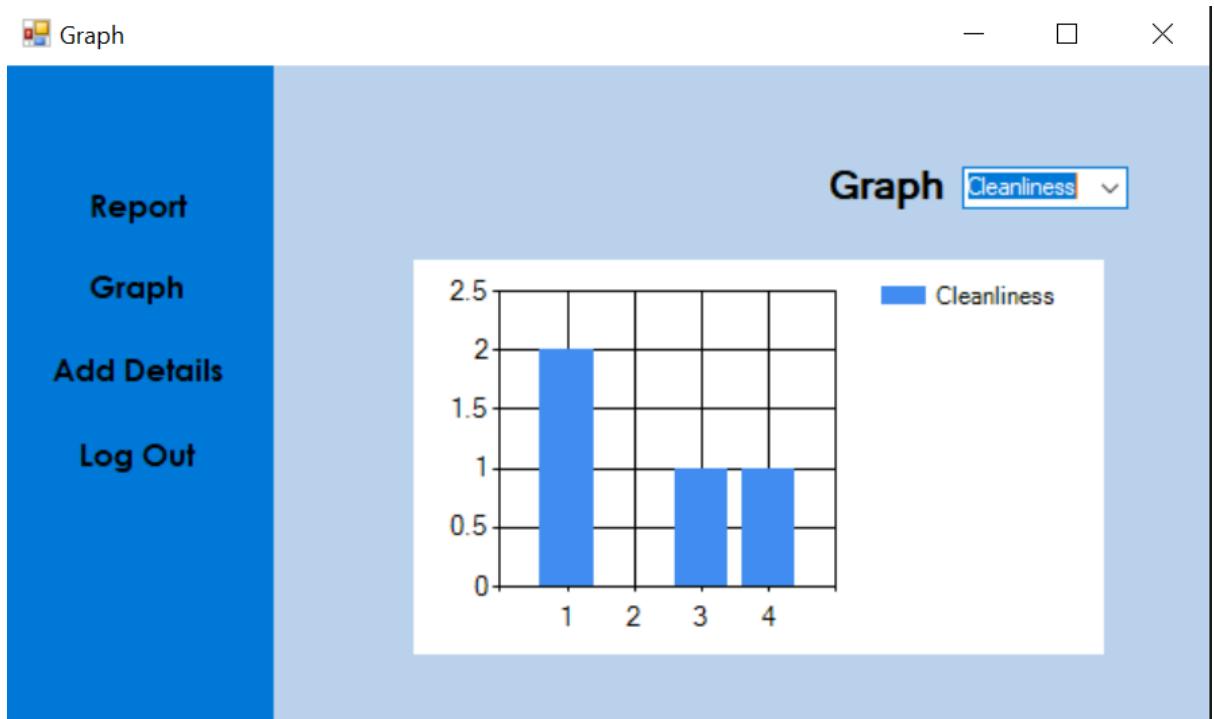


Figure 7: Graph View of selected data

Upon selecting the combo box the graph will appear according to the criteria selected.

d. Adding Criteria Name



Figure 8: Add of Criteria Parking

The criteria name is added successfully.

CriteriaName	Excellent	Good	Average	Dissatisfied
Food Quality	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Staff Friendliness	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Cleanliness	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Order Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Restaurant Ambia	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Value For Money	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Parking	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 9: Criteria Name added in feedback form

The criteria name filled in admin panel displays in feedback form of the application.

e. Logout



Figure 10: Logout

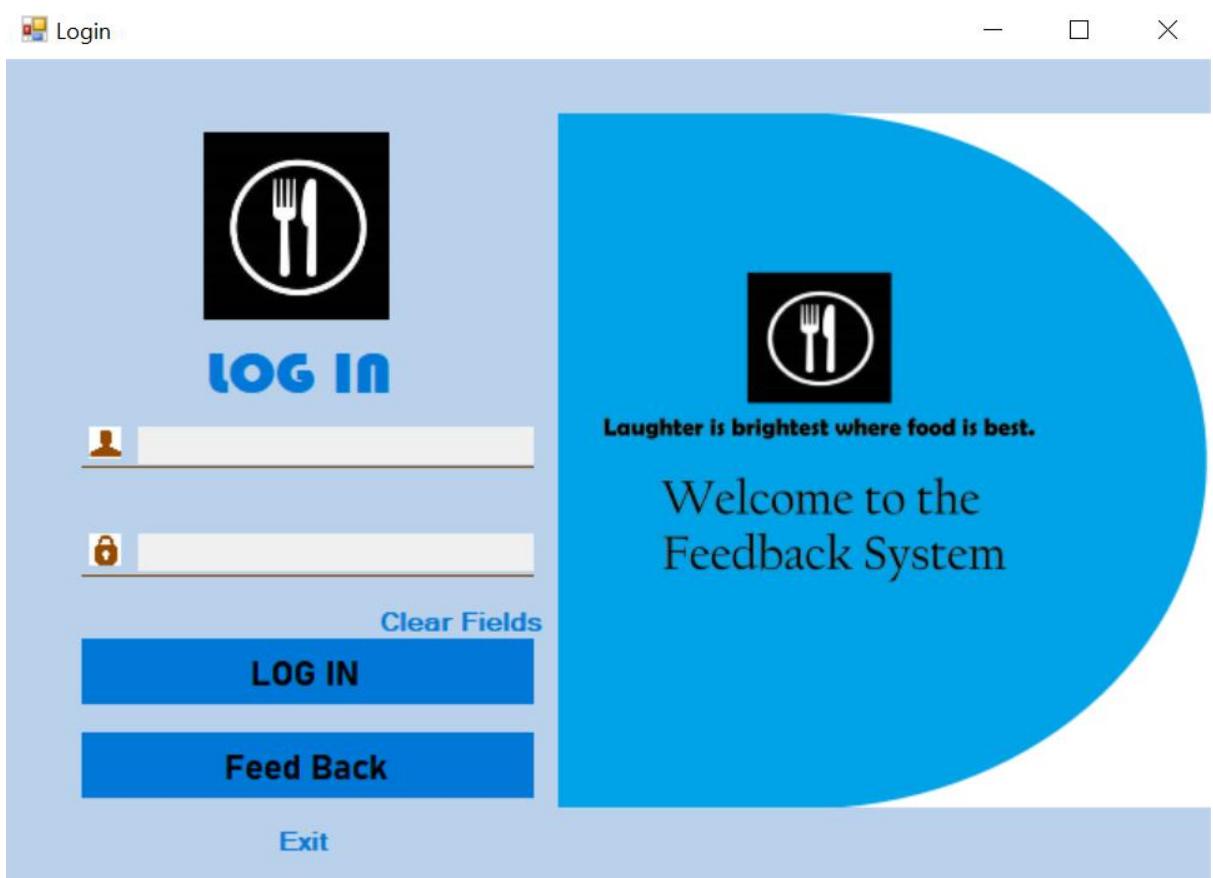


Figure 11: Main page

After clicking the logout button, it redirects to home page of the application.

2.2. Software Architecture Diagram

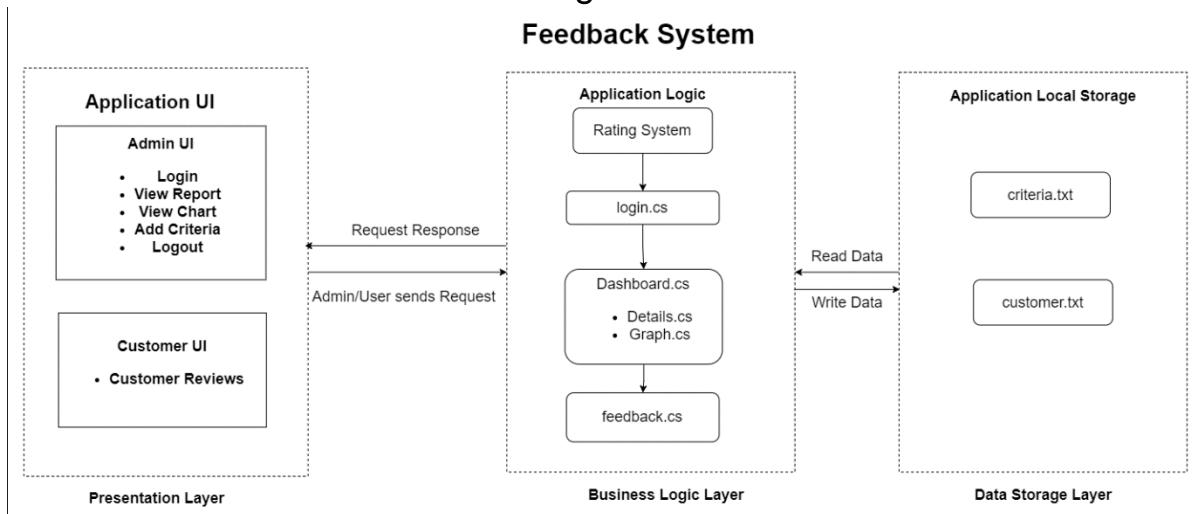


Figure 12: System Architecture

2.3. Class Diagram

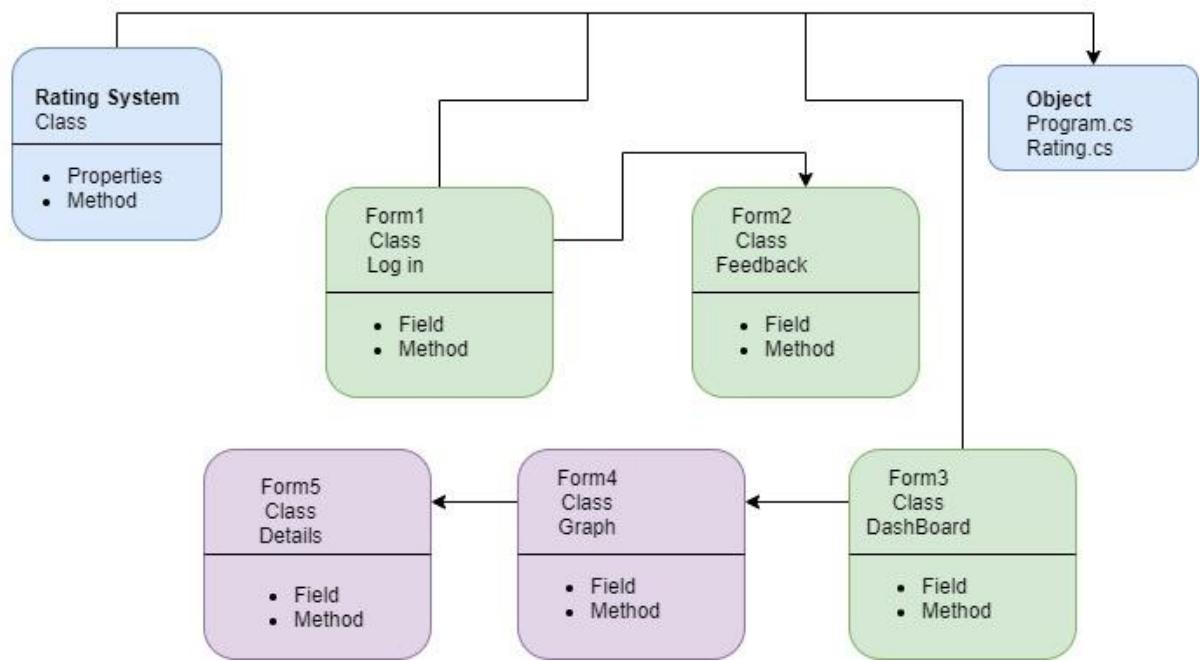


Figure 13: Class Diagram of Rating System

2.4. Class Description

Form1: Class

Login	
Fields	
	pictureBox1 pictureBox2 pictureBox2 lblClear panel1 panel2 txtUsername txtPassword btnLogin btnfeedback label1 label2
Methods	Login() lblClear_Click() btnLogin_Click() btnfeedback_Click() Label2_Click() txtPassword_TextChanged()

Figure 14: Form1; Login

Form2: Feedback

Feeback	
Fields	pictureBox1 label1 label2 label3 label4 label5 label6 panel1 txtEmail txtSuggestion txtCustomerName txtContactNumber gridFeedback btnSubmit btnReturn
Methods	Feedback() LoadCsvData() Feedback_Load() btnSubmit_Click() btnReturn_Click() gridFeedback_CellContentClick() addRecord() addRecord1()
Class	CriteriaDetail

Figure 15: Form2; Feedback

Form3: Dashboard

Dashboard	
Fields	
Methods	
	btnReport btnGraph btnAddDetails btnLogout btnViewReport btnSort txtViewReport gridReport label1 panel1
	Dashboard() btnReport_Click() btnAddDetails_Click() btnLogout_Click() btnGraph_Click() BindData()

Figure 16: Form3; Dashboard

Form4: Graph

Graph	
Fields	
Methods	
	btnReport btnGraph btnLogout btnAddDetails panel1 panel2 label1 label2 comboBoxGraph lblGraphTitle chartValueForMoney chartRestaurantAmbiance chartOrderAccuracy chartStaffFriendliness ChartFoodQuality chartCleanliness
	Graph() btnReport_Click() btnGraph_Click() btnAddDetails_Click() btnLogout_Click() comboBoxGraph_SelectedIndexChanged() Graph_Load()

Figure 17: Form4; Graph

Form5: Details

Details	
Fields	
Methods	
	btnReport btnClear btnGraph btnAddDetails btnLogout btnAddCriteria label1 label2 panel1 txtCriteria
	Details() AddRecord() btnGraph_Click() btnAddDetails_Click() btnLogout_Click() btnAddCriteria_Click() btnClear_Click() btnReport_Click()

Figure 18: Form5; Details

2.5. Method Description

Login Form

Method	Description
btnLogin_Click()	Get the data from the textbox and enter the dashboard on clicks. Also validates the data.
lblClear_Click()	Clear the value in the textbox when clicked.
Label2_Click()	Exit the application.
btnfeedback_Click()	Shows the feedback page by calling it.
txtPassword_TextChanged()	Make password hidden while typing.

Table 1: Method Description of Login Form

Feedback Form

Method	Description
CriteriaName()	To get and set data.
LoadCsvData()	Add the data one by one from the csv file to List.
Feedback_Load()	Display the data from the list to the gridFeedback in the table form.
btnSubmit_Click()	Takes the data from the feedback form and table then send the data to addRecord and addRecord1 method.
addRecord()	Adds the textbox value received from btnSubmit_Click method to customer.csv file.
addRecord1()	Adds the table rating data to the customer.csv file.
btnReturn_Click()	Display login page by calling its constructor.
gridFeedback_CellContentClick()	Allows to tick only one check box in a row of gridFeedback.

Table 2: Method Description of Feedback Form

Dashboard Form

Method	Description
btnReport_Click()	Open File Dialog box, set the file path in the textfield and sends the textfield data to BindData method.
BindData()	Create an object of datatable, Gets data using loop and display table details.
btnGraph_Click()	Shows or take the admin to Graph page by calling its constructor.
btnAddDetails_Click()	Shows or take the admin to Details page by calling its constructor.
btnLogout_Click()	Log outs and take you to login page by calling its constructor.

Table 3: Method Description of Dashboard Form

Graph Form

Method	Description
btnReport_Click()	Takes the admin or user to report by calling its constructor.
Graph_load()	Makes all the chart hidden by using Hide() method.
btnGraph_Click()	Display or take the admin to Graph page by calling its method.
btnAddDetails_Click()	Display or take the admin to Details page by calling its constructor.
btnLogout_Click()	Log outs and take you to login page by calling its constructor.
comboBoxGraph_SelectedIndexChanged()	As per the selected index of the combox index number gets the details of selected criteria from the csv file with the help of index value in it and display the chart.

Table 4: Method Description of Graph Form

Details Form

Method	Description
btnGraph_Click()	Shows or take the admin to Graph page by calling its constructor.
btnReport_Click()	Takes the admin or user to report by calling its constructor.
btnAddDetails_Click()	Display or take the admin to Details page by calling its constructor.
btnLogout_Click()	Log outs and take you to login page by calling its constructor.
btnAddCriteria_Click ()	Takes the text field data send it to AddRecord method through parameter.
AddRecord()	Take the data and file path and add criteria to new line in criteria.csv file.
btnClear_Click()	Clears the text field value by calling Clear() method.

Table 5: Method Description of Details Form

2.6. Data Structure and Algorithm Description

2.6.1. Data Structure

Data structures are a reflection of the logical relationships that exist between various data elements. Data structures, in other words, describe a way to organize all data items that take into consideration not only the elements stored, but also their relationship with each other. To explain the way data is stored, the term Data Structures is used. ([educationtrick, 2021](#))

The data types that were used in this project were:

- String: The String data type stores any kind of data so it is used to store name, suggestion, email, and others.
- Bool: It is a data type that has one of two possible values.
- Int: The integer data type is used to store the number or id.
- Double: The Double data type provides the largest and smallest possible magnitudes for a number.
- DateTime: The DateTime data type is used to store the date and time values such as the Date of filling form.

2.6.2. Algorithm used

Bubble sort is a simple sorting algorithm that works by repeatedly stepping through lists that need to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. This passing procedure is repeated until no swaps are required, indicating that the list is sorted. Bubble sort gets its name because smaller elements bubble toward the top of the list. This algorithm is not suitable for large data sets as its average. Bubble sort is also referred to as sinking sort or comparison sort. ([Frank Donald, 2018](#))

Bubble sort is an algorithm that compares the adjacent elements and swaps their positions if they are not in the intended order. The order can be ascending or descending. Starting from the first index, compare the first and the second elements. If the first element is greater than the second element, they are swapped. Now, compare the second and the third elements. Swap them if they are not in order. The above process goes on until the last element.

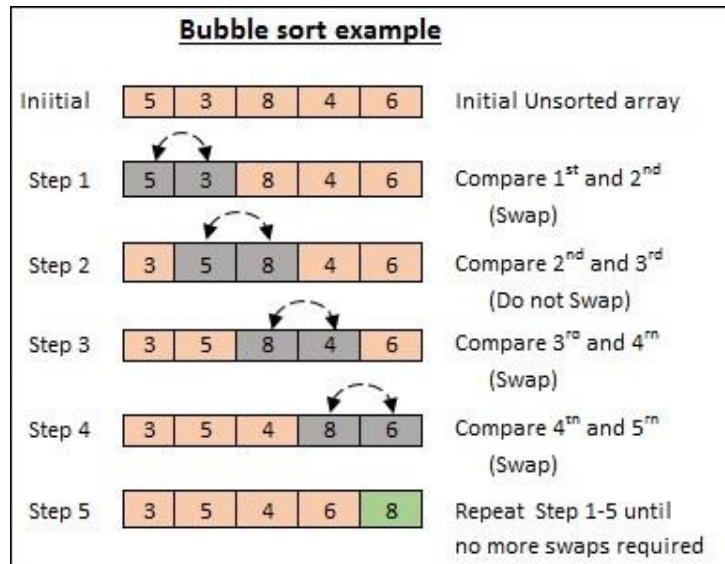


Figure 19: Bubble Sort Algorithm

2.6.3. Algorithm

Step 1: Bubble Sort(Array : list of sortable items)

Step 2: N= Array. Length

Set Flag := True

Step 3: Repeat Steps from 3 to 5 for I = 1 to N-1 while Flag == true

Set Flag := False

Set i:=0

Step 4: Repeat while i<N-1 [Executing pass]

(a) If Array[i+1]>Array[i], then:

Step 5: Swap Array[i] and Array[i+1] Set Flag:= True

(b) Set i :=i+1

Step 6: End bubble sort

2.6.4. Flowchart

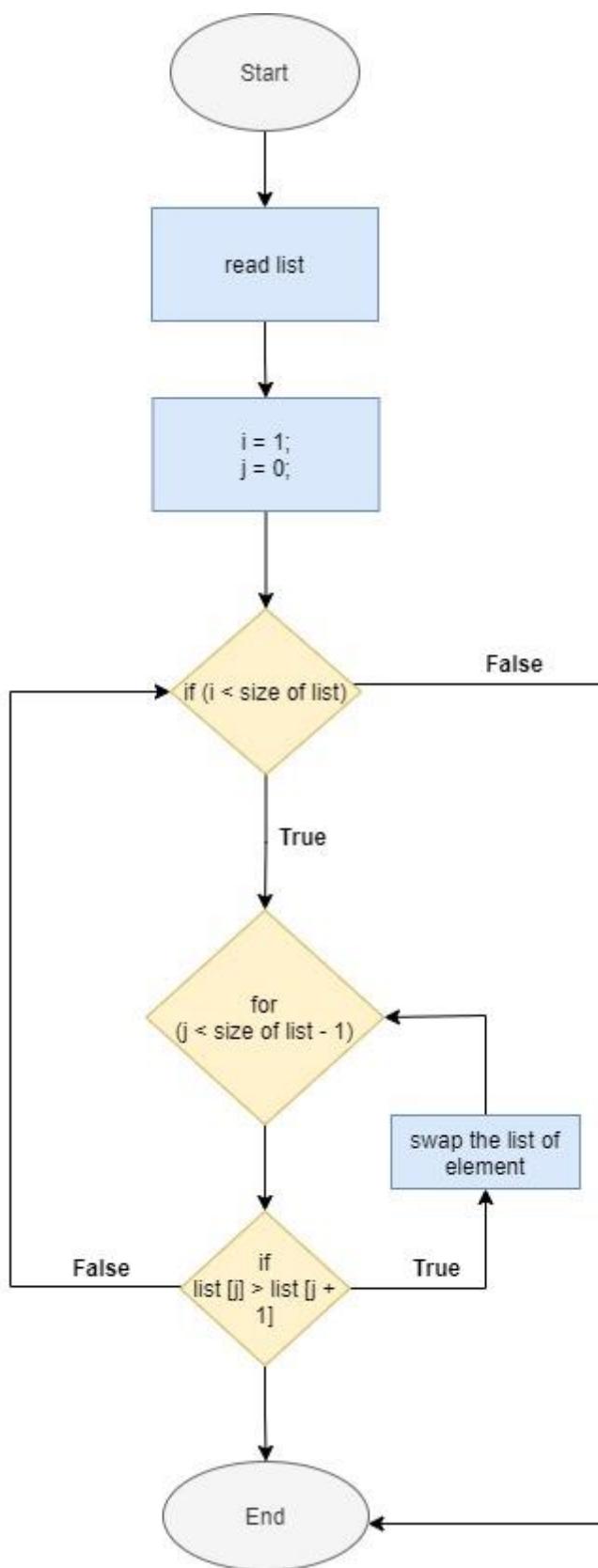


Figure 20: Flow Chart

3. Testing

Testing 1: Launching the Application

Action	Launch the application.
Expected output	Successful loading of the desktop application with appropriate GUI interface.
Actual result	Desktop application is loaded with required interface.
Conclusion	Successful.

Table 6: Testing 1

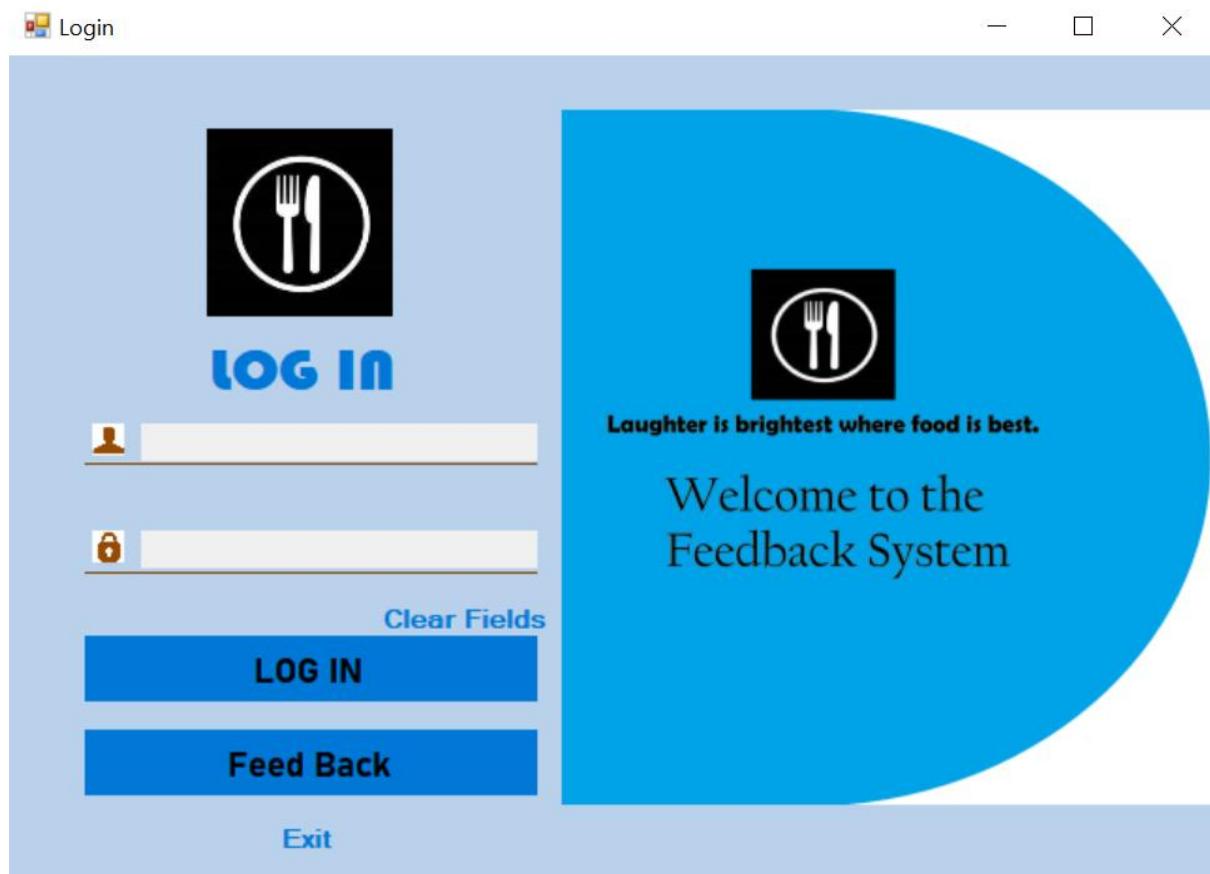


Figure 21: Home Page of system

Testing 2: Submitting inappropriate username and password

Action	Submitting the form with inappropriate user login and password.
Expected output	Error message to be displayed.
Actual result	An error message is displayed: “The user name or password you entered is incorrect, try again”.
Conclusion	Successful.

Table 7: Testing 2



Figure 22: Submitting incorrect username & password

Testing 3:

Action	Submitting the form without username and password.
Expected output	Error message to be displayed.
Actual result	An error message is displayed: "Enter Username or Enter Password".
Conclusion	Successful.

Table 8: Testing 3

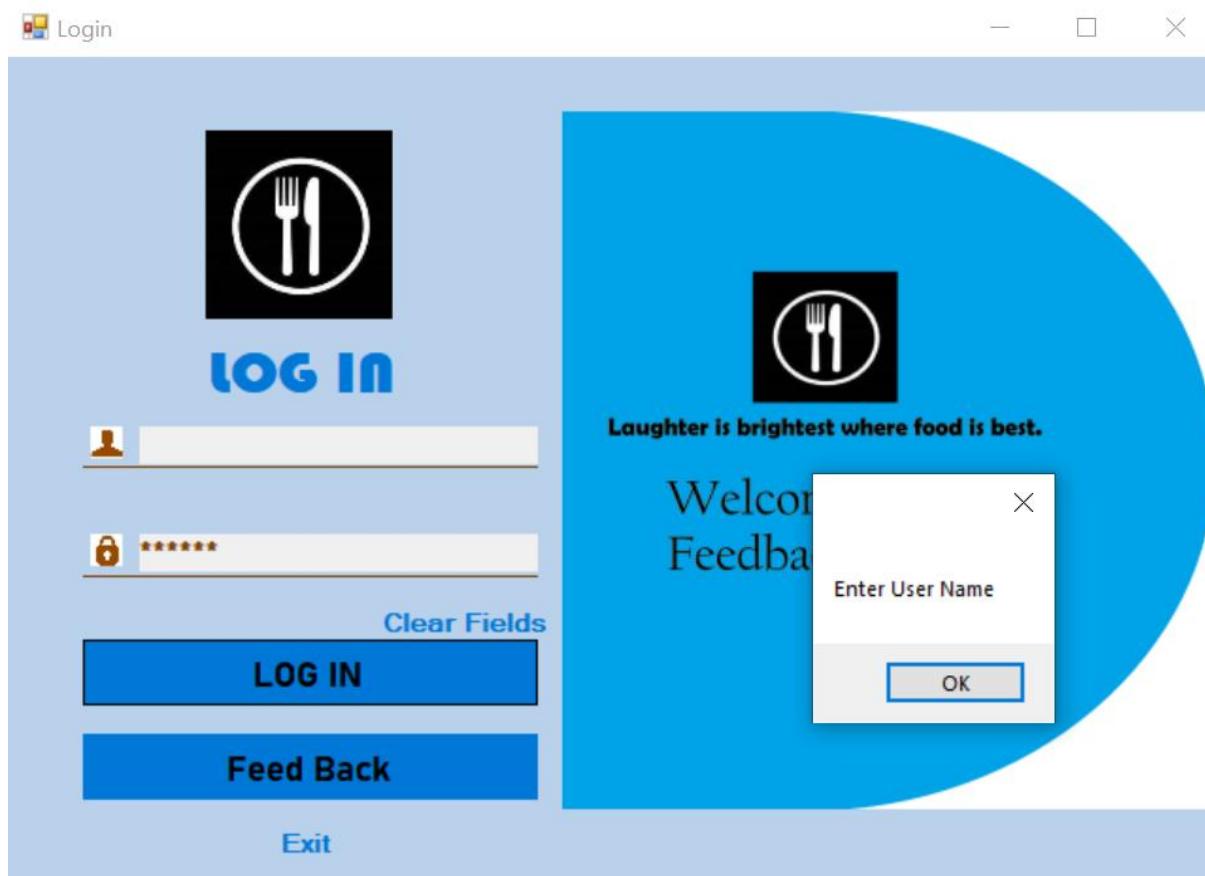


Figure 23: Submitting only password

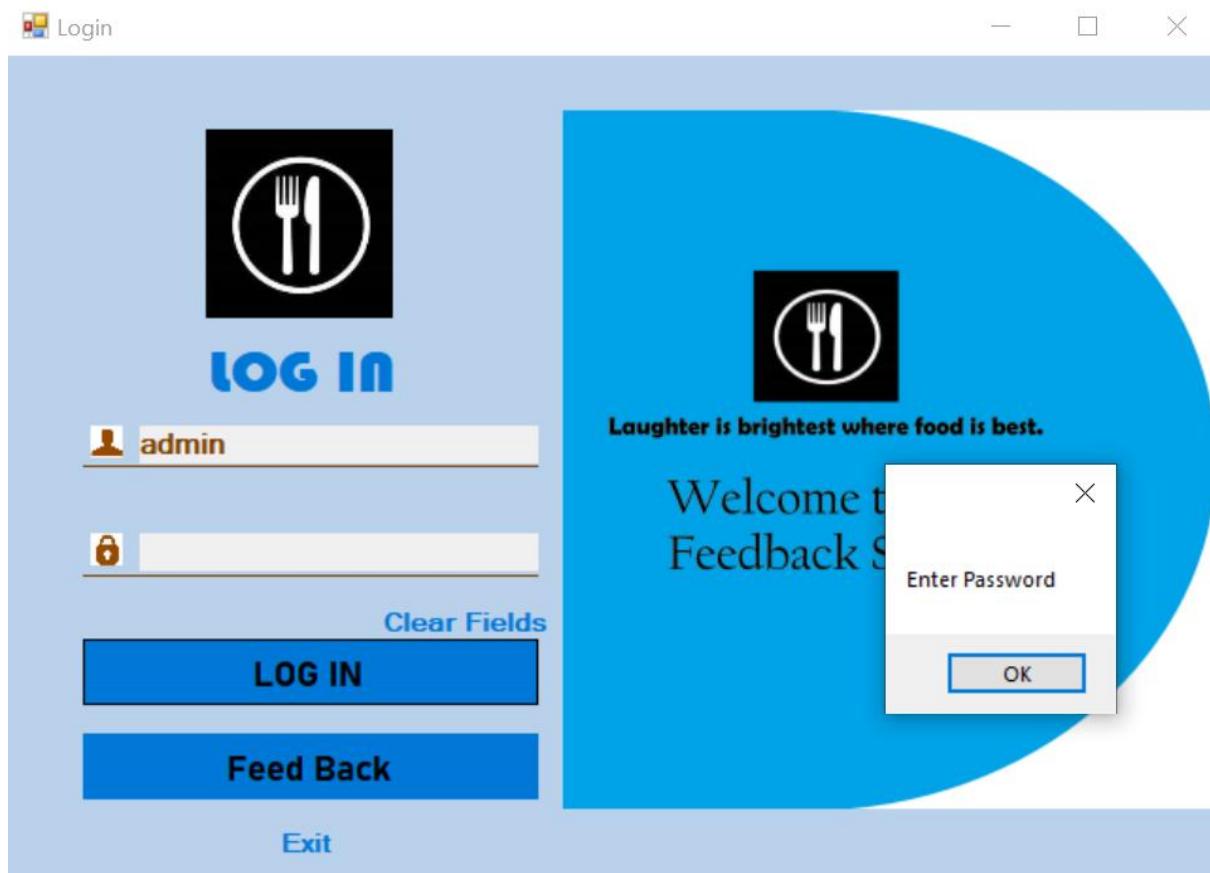


Figure 24: Submitting password only

Testing 4: Submitting valid form for customer feedback

Action	Submit valid form for customer feedback
Expected output	Customer data to be inserted in the table grid with an appropriate alert message.
Actual result	A message is displayed: “Successful Data added” with the data being displayed at the report of admin.
Conclusion	Successful.

Table 9: Testing 4

The screenshot shows a web-based application for customer feedback. At the top, there's a logo with the word "Welcome" and a sun icon, followed by the text "Get In Touch!". Below this, there are input fields for "Customer Name" (Salina Chettri), "Contact Number" (9847821689), "Email" (b.salina71@gmail.com), and a "Suggestion" text area containing the text "It was nice experience". To the right, there's a large table titled "Send us your feedback!" with columns for CriteriaName (Food Quality, Staff Friendliness, Cleanliness, Order Accuracy, Restaurant Ambia, Value For Money, Parking) and rows for rating levels (Excellent, Good, Average, Dissatisfied). A modal dialog box is overlaid on the table, displaying the message "Successful" and "Data Added" with a checked checkbox, and an "OK" button. At the bottom of the page are "Submit" and "Return" buttons.

Figure 25: Customer Data is added

Testing 5: Submitting string type in contact number.

Action	Submit string type in contact number.
Expected output	Error message to be displayed.
Actual result	An error message is displayed: “Contact Number should be of type int”.
Conclusion	Successful.

Table 10: Testing 5

The screenshot shows a web-based feedback form titled "Send us your feedback!". The form includes fields for Customer Name (Shreeta Mool), Contact Number (salina), Email, and Suggestion. A large "Get In Touch!" button is visible. On the right, there is a grid for rating service criteria: Food Quality (Excellent checked), Good (unchecked), Average (unchecked), Dissatisfied (unchecked). Below the grid is a table with columns for CriteriaName and ratings from Excellent to Dissatisfied. A modal dialog box is overlaid on the page, displaying the error message "Contact Number should be of type int" with an "OK" button.

Figure 26: Submitting string type in contact

Testing 6: Viewing Report

Action	Viewing the report from admin side in table.
Expected output	The feedbacks given by customer should be displayed in grid view.
Actual result	The feedback is shown successfully in the table.
Conclusion	Successful.

Table 11: Testing 6

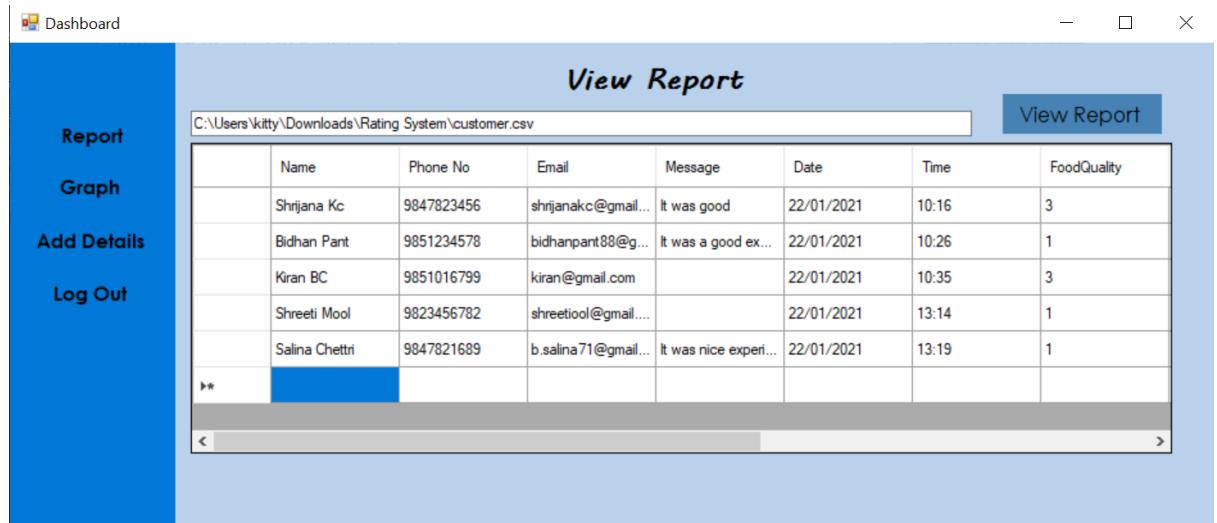


Figure 27: Report View

Testing 7: Displaying graph

Action	Click checkbox and generate the graph of feedbacks.
Expected output	Graph of criteria should be displayed.
Actual result	Graph of each criteria is displayed.
Conclusion	Successful.

Table 12: Testing 7

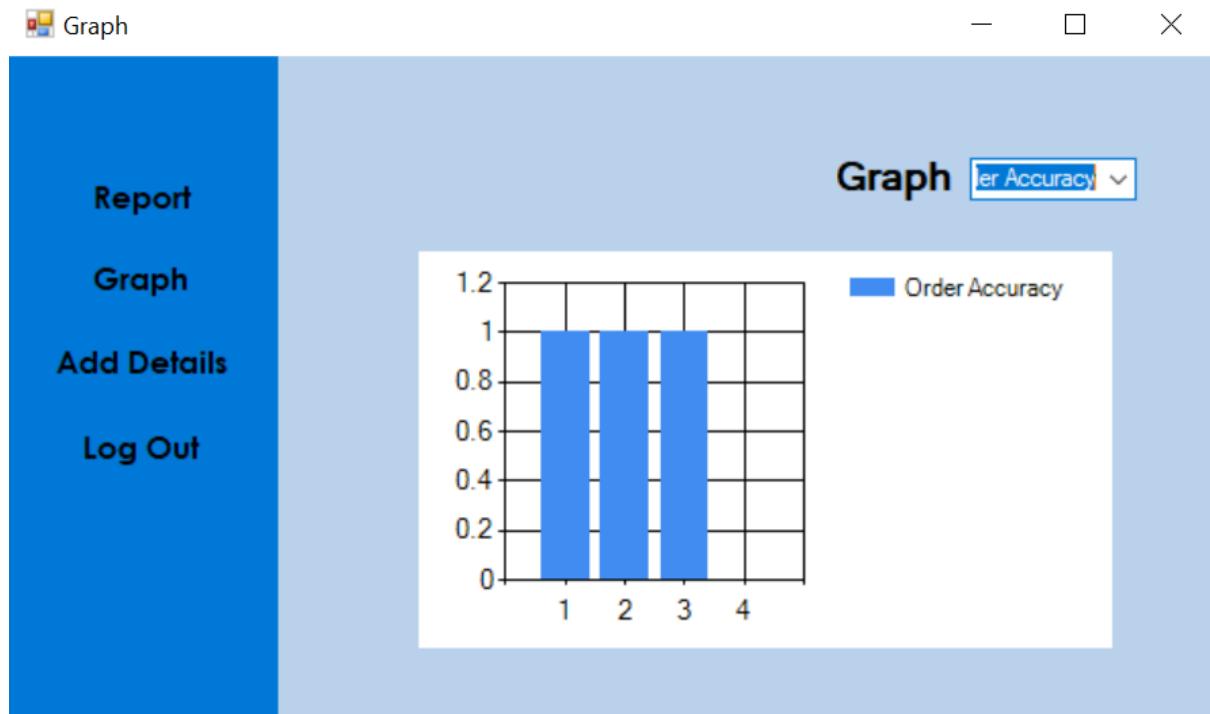


Figure 28: Graph of Order Accuracy

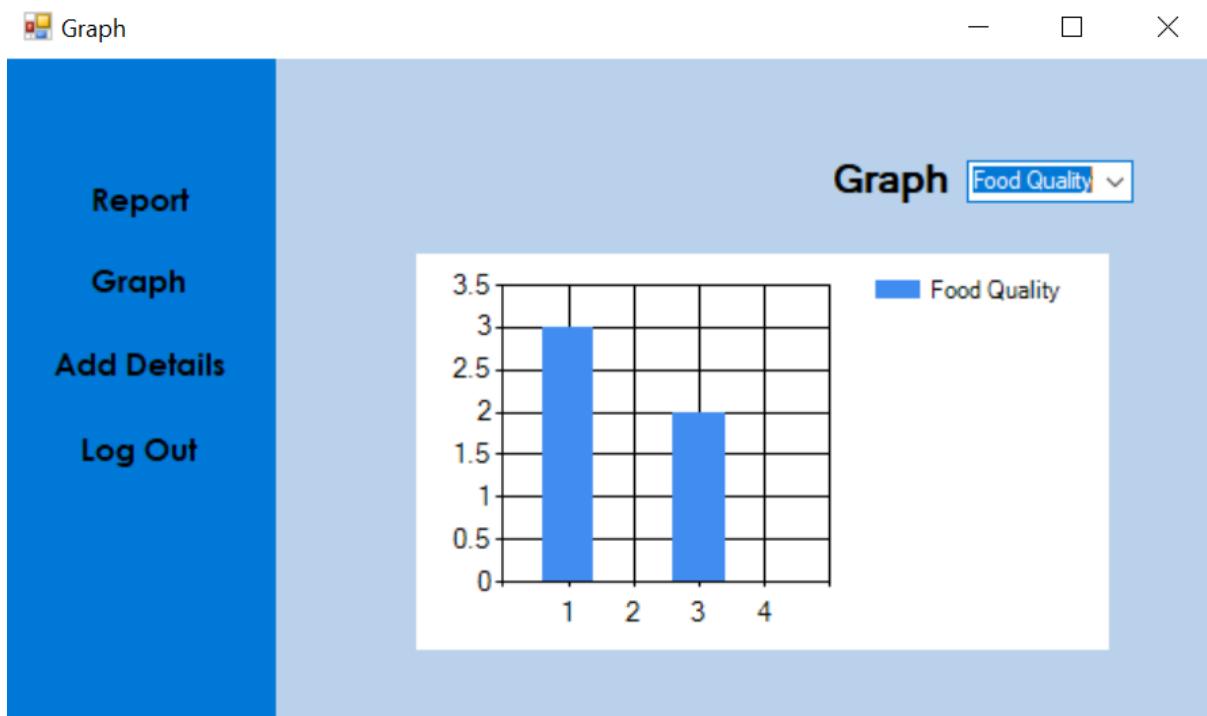


Figure 29: Graph of Food Quality

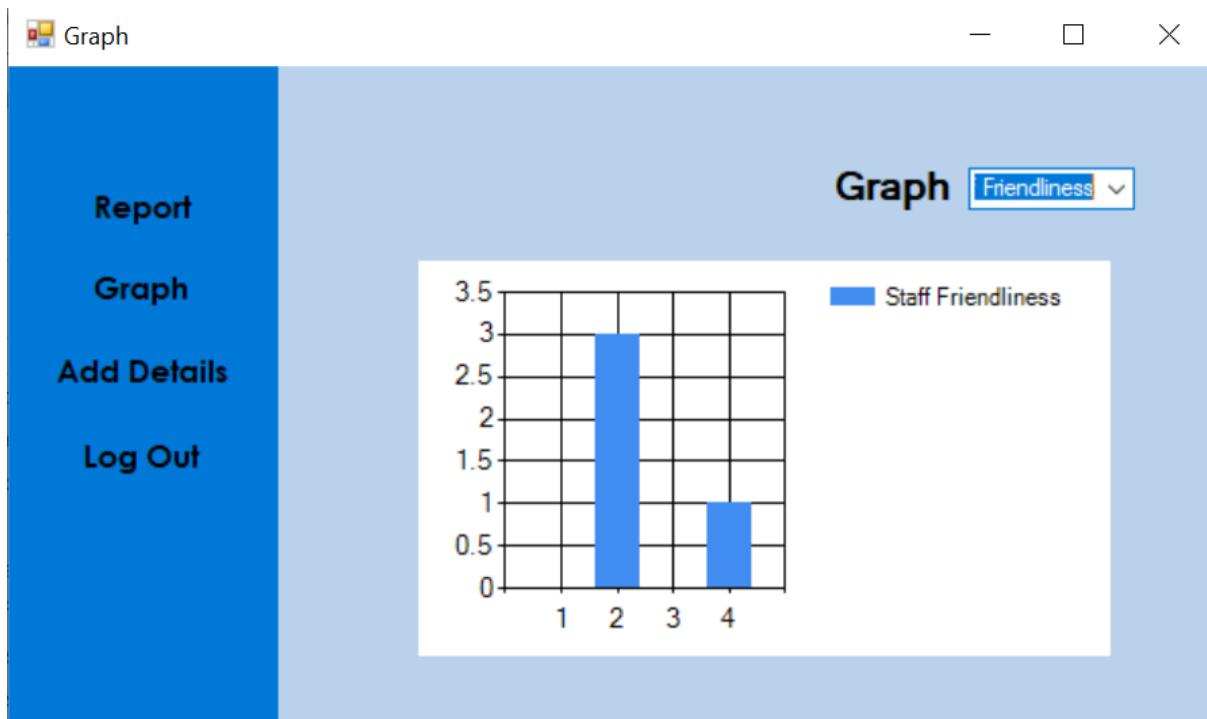


Figure 30: Graph of Staff Friendliness

Testing 8: Add of Criteria

Action	Adding new detail in criteria of feedback form.
Expected output	The live music will added in the list of criteria.
Actual result	The live music is added in the feedback form
Conclusion	Successful.

Table 13: Testing 8

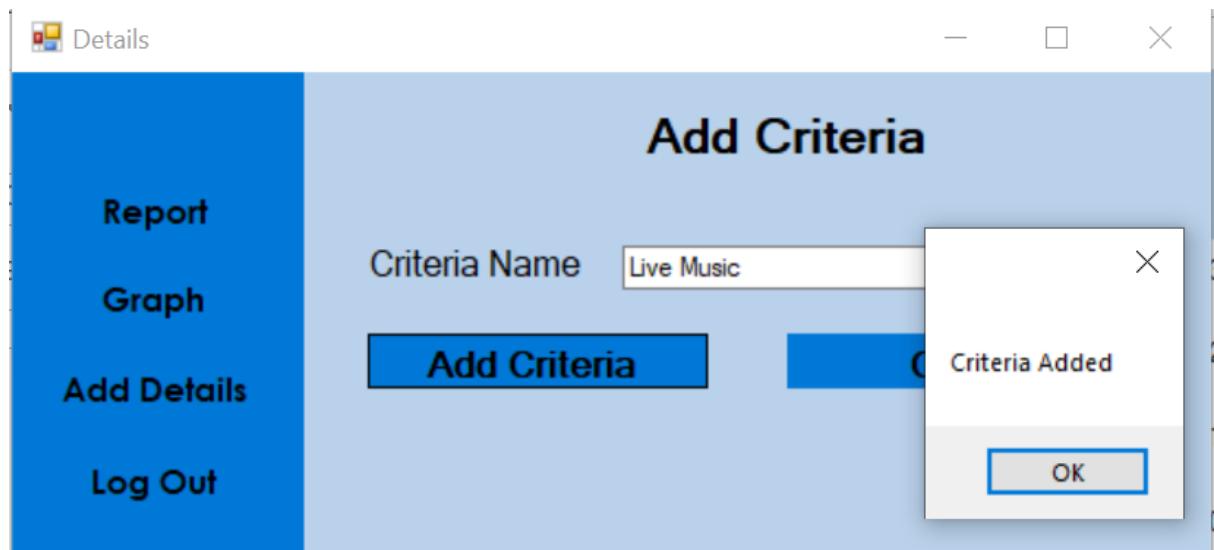


Figure 31: Criteria Added

Testing 9: Showing the added criteria details in feedback form

Action	Showing the added criteria details in feedback form.
Expected output	The live music will be added in the list of criteria and it will be shown in the feedback form.
Actual result	The live music is shown in feedback form.
Conclusion	Successful.

Table 14: Testing 9

The screenshot shows a feedback form titled "Send us your feedback !". On the left, there are input fields for "Customer Name", "Contact Number", "Email", and a "Suggestion" text area. On the right, there is a grid for rating criteria. The grid has columns for "Excellent", "Good", "Average", and "Dissatisfied". The rows include "Food Quality", "Staff Friendliness", "Cleanliness", "Order Accuracy", "Restaurant Ambia", "Value For Money", "Parking", and "Live Music". The "Live Music" row is highlighted with a blue background. In the "Excellent" column for "Live Music", there is a checked checkbox. In the "Good" column, there is also a checked checkbox. The other two columns ("Average" and "Dissatisfied") have empty checkboxes. At the bottom of the grid are "Submit" and "Return" buttons.

CriteriaName	Excellent	Good	Average	Dissatisfied
Food Quality	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Staff Friendliness	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Cleanliness	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Order Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Restaurant Ambia	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Value For Money	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Parking	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Live Music	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 32: Criteria Added in feedback form

Testing 10: Logout

Action	Logging out and closing the system
Expected output	Clicking the logout the system will take to login form and after clicking exit the system will be closed.
Actual result	Clicking the logout the system shows login form and after clicking exit the system is closed.
Conclusion	Successful.

Table 15: Testing 10

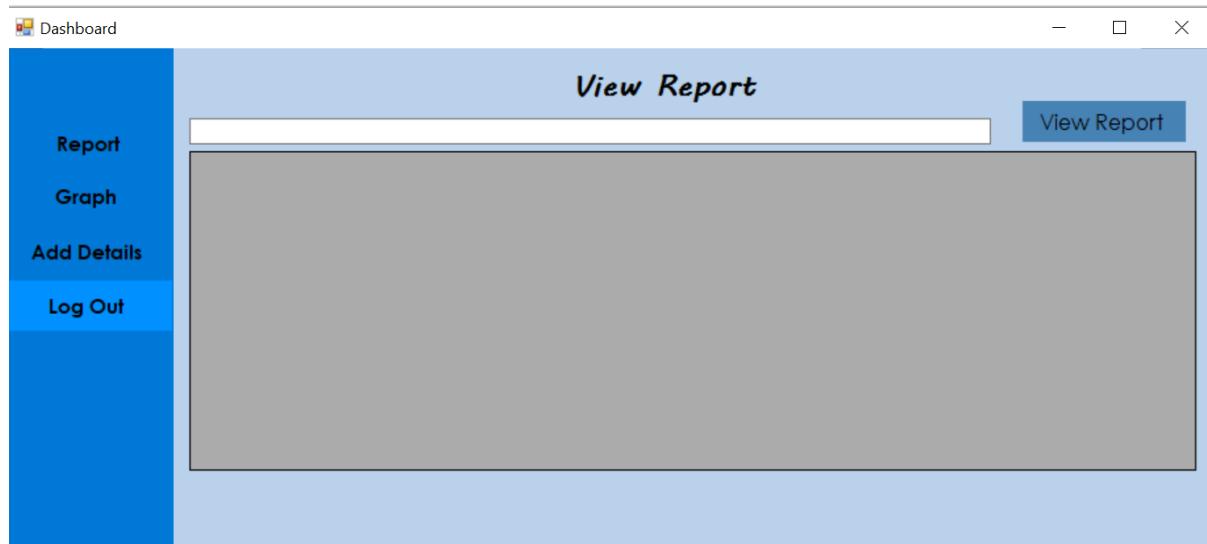


Figure 33: Clicking the logout

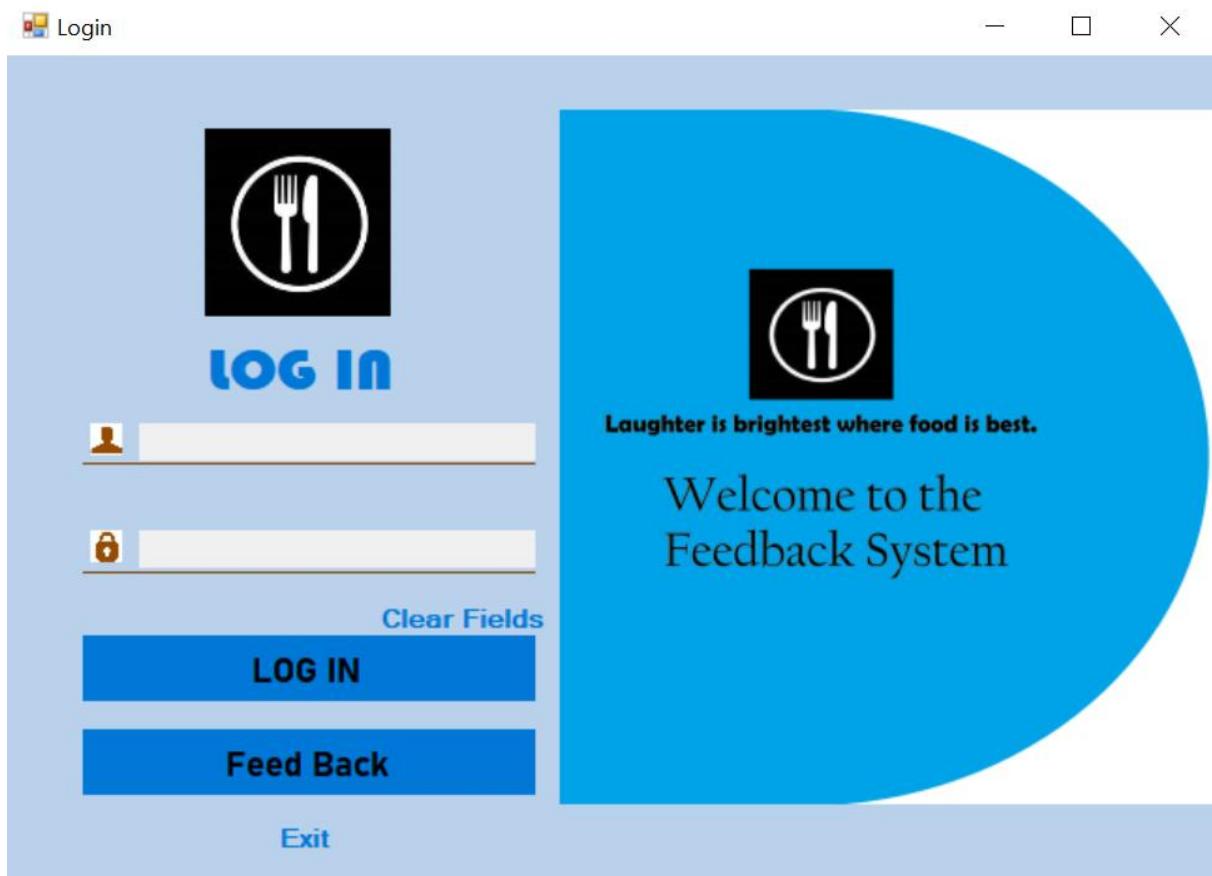


Figure 34: After clicking logout

4. Conclusion

I acquired knowledge of the C# language, Visual Studio and the .NET framework after completing this project. Learning these components gave me ideas about how to build a desktop application and how to use the combination of .NET and C# to run the program. I also learned about the features that Visual Studio gives the developer, and the debugging feature is one of its most complementary features. The debugging features of Visual Studio are extremely useful when it comes to recommending and highlighting errors. Its user-friendly tools for developing GUI are also very helpful in a short period of time for developing applications.

5. Bibliography

- educationtrick, 2021. *Data Structures & Algorithms | What Is Data Structures? | Education Trick.* [Online] Available at: <https://www.educationtrick.com/what-is-data-structures/> [Accessed 22 January 2021].
- Frank Donald, 2018. *Bubble sort Algorithm Explained - Gadgetronicx.* [Online] Available at: <https://www.gadgetronicx.com/bubble-sort-explained-code/> [Accessed 20 January 2021].
- Microsoft , 2021. *Visual Studio 2019 IDE - Programming Software for Windows.* [Online] Available at: <https://visualstudio.microsoft.com/vs/> [Accessed 20 JJanuary 2021].
- Microsoft, 2021. *What is .NET? An open-source developer platform..* [Online] Available at: <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet#:~:text=.NET.%20is%20a%20free,%20cross-platform,%20open%20source%20developer,build%20for%20web,%20mobile,%20desktop,%20games,%20and%20IoT.> [Accessed 20 January 2021].
- techopedia, 2020. *What is C# (C Sharp)? - Definition from Techopedia.* [Online] Available at: <https://www.techopedia.com/definition/26272/c-sharp> [Accessed 20 JJanuary 2021].

6. References

- educationtrick, 2021. *Data Structures & Algorithms | What Is Data Structures? | EducationTrick.* [Online] Available at: <https://www.educationtrick.com/what-is-data-structures/> [Accessed 22 January 2021].
- Frank Donald, 2018. *Bubble sort Algorithm Explained - Gadgetronicx.* [Online] Available at: <https://www.gadgetronicx.com/bubble-sort-explained-code/> [Accessed 20 January 2021].
- Microsoft , 2021. *Visual Studio 2019 IDE - Programming Software for Windows.* [Online] Available at: <https://visualstudio.microsoft.com/vs/> [Accessed 20 JJanuary 2021].
- Microsoft, 2021. *What is .NET? An open-source developer platform..* [Online] Available at: <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet#:~:text=.NET.%20is%20a%20free,%20cross-platform,%20open%20source%20developer,build%20for%20web,%20mobile,%20desktop,%20games,%20and%20IoT.> [Accessed 20 January 2021].
- techopedia, 2020. *What is C# (C Sharp)? - Definition from Techopedia.* [Online] Available at: <https://www.techopedia.com/definition/26272/c-sharp> [Accessed 20 JJanuary 2021].

7. Appendix

1. Login

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar says "Rating System". The main window displays the "Login.cs" code file. The code is as follows:

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace Rating_System
12 {
13     public partial class Login : Form
14     {
15         public Login()
16         {
17             InitializeComponent();
18         }
19
20         private void label1_Click(object sender, EventArgs e)
21         {
22             //null validation
23             if (string.IsNullOrEmpty(txtUsername.Text))
24             {
25                 MessageBox.Show("Enter User Name");
26                 txtPassword.Clear();
27                 txtUsername.Focus();
28             }
29             else
30             {
31                 if (string.IsNullOrEmpty(txtPassword.Text))
32                 {
33                     MessageBox.Show("Enter Password");
34                     txtPassword.Clear();
35                 }
36             }
37         }
38
39         private void btnLogin_Click(object sender, EventArgs e)
40         {
41             //null validation
42             if (string.IsNullOrEmpty(txtPassword.Text))
43             {
44                 MessageBox.Show("Enter Password");
45                 txtPassword.Focus();
46             }
47             else
48             {
49                 if (txtUsername.Text == "admin" & txtPassword.Text == "admin")
50                 {
51                     new Dashboard().Show();
52                     this.Hide();
53                 }
54                 else
55                 {
56                     MessageBox.Show("The user name or password you entered is incorrect, try again");
57                     txtUsername.Clear();
58                     txtPassword.Clear();
59                     txtUsername.Focus();
60                 }
61             }
62         }
63
64         private void lblClear_Click(object sender, EventArgs e)
65         {
66             txtUsername.Clear();
67             txtPassword.Clear();
68             txtUsername.Focus();
69         }
70
71         private void label2_Click(object sender, EventArgs e)
72         {
73             Application.Exit();
74         }
75
76         private void btFeedback_Click(object sender, EventArgs e)
77         {
78             new Feedback().Show();
79             this.Hide();
80         }
81     }
82 }

```

The Solution Explorer on the right shows the project structure for "Rating System" with files like Program.cs, Rating.cs, Graph.cs, Graph.cs [Design], Login.cs, Login.cs [Design], and Login.cs [Design]. The Properties and Solution Explorer tabs are also visible.

This screenshot continues the "Login.cs" code from the previous one. The code handles the button click event for the "label2_Click" method, which exits the application. It also includes the "btFeedback_Click" method which shows a feedback form and hides the current login form.

```

81     private void btFeedback_Click(object sender, EventArgs e)
82     {
83         new Feedback().Show();
84         this.Hide();
85     }
86 }

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Rating_System
{
    public partial class Login : Form
    {
        private void txtUsername_TextChanged(object sender, EventArgs e)
        {
            txtPassword.Clear();
            txtUsername.Focus();
        }

        private void txtPassword_TextChanged(object sender, EventArgs e)
        {
            txtUsername.Clear();
            txtPassword.Clear();
            txtUsername.Focus();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            txtUsername.Clear();
            txtPassword.Clear();
            txtUsername.Focus();
        }

        private void label2_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void btnFeedback_Click(object sender, EventArgs e)
        {
            new Feedback().Show();
            this.Hide();
        }

        private void txtPassword_TextChanged(object sender, EventArgs e)
        {
            txtPassword.PasswordChar = '*';
        }
    }
}

```

2. Feedback

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Rating_System
{
    public partial class Feedback : Form
    {
        public string CriteriaFile = "#..\..\criteria.csv";
        public string CustomerFile = "#..\..\customer.csv";

        public Feedback()
        {
            InitializeComponent();
        }

        public class CriteriaDetail
        {
            public string CriteriaName { get; set; }
        }

        public List<CriteriaDetail> LoadCsvData(string CsvFile)
        {
            var query = from l in File.ReadAllLines(CsvFile)
                       let data = l.Split(',')
                       select new CriteriaDetail
                       {
                           //l[0] = data[0],
                           CriteriaName = data[0]
                       };
            return query.ToList();
        }

        private void Feedback_Load(object sender, EventArgs e)
        {
        }
    }
}

```

The screenshot shows the Visual Studio IDE interface with the following details:

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search (Ctrl+Q).
- Solution Explorer:** Shows the solution 'Rating System' with files: Properties, References, img, App.config, Dashboard.cs, Details.cs, Feedback.cs (selected), Graph.cs, Login.cs, Program.cs, and Rating.cs.
- Toolbox:** Shows the 'General' group with a note: "There are no usable controls in this group. Drag an item onto this text to add it to the toolbox."
- Code Editor:** Displays the Feedback.cs code for the Rating System. The code handles loading data from a CSV file, validating contact numbers, and processing feedback data based on criteria values (Excellent, Good, Average, Poor, Dissatisfied).
- Status Bar:** Shows Ln: 208, Ch: 1, SPC, CRLF, Properties, Solution Expl..., Git Changes.
- Taskbar:** Shows the Windows Start button, a search bar with 'Type here to search', and pinned icons for Mail, Photos, OneDrive, Microsoft Edge, Store, and File Explorer.

This screenshot is identical to the one above, showing the Visual Studio interface with the Feedback.cs code for the Rating System project. The code is identical to the one shown in the first screenshot, handling the conversion of grid feedback rows into criteria values (Excellent, Good, Average, Poor, Dissatisfied) based on their row indices.

```

113     }
114     catch (InvalidCastException ex)
115     {
116         MessageBox.Show(" " + ex);
117     }
118     if (l == 0)
119     {
120         addRecord(txtCustomerName.Text, txtContactNumber.Text, txtEmail.Text, txtSuggestion.Text, date, time, CriteriaValue,
121     }
122     else
123     {
124         addRecord1(CriteriaValue, CustomerFile);
125     }
126 }
127 using (System.IO.StreamWriter file = new System.IO.StreamWriter(Customerfile, true))
128 {
129     file.WriteLine();
130 }
131 catch (Exception)
132 {
133     MessageBox.Show("Contact Number should be of type int");
134     txtContactNumber.Clear();
135     txtContactNumber.Focus();
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }

// method to add the feedback to customer file

public void addRecord(string name, string contact, string email, string message, string date, string time, string feedback, string filepath)
{
    try
    {
        using (System.IO.StreamWriter file = new System.IO.StreamWriter(@filepath, true))
        {
            //file.WriteLine();
            file.WriteLine(name + "," + contact + "," + email + "," + message + "," + date + "," + time + "," + feedback + ",");
        }
        MessageBox.Show("Data Added", "Successful");
    }
    catch (Exception ex)
    {
        throw new ApplicationException("This program is not doing well", ex);
    }
}

// method to add the feedback to customer file

public void addRecord1(string feedback, string filepath)
{
    try
    {
        using (System.IO.StreamWriter file = new System.IO.StreamWriter(@filepath, true))
        {
            file.WriteLine(feedback + ",");
        }
    }
    catch (Exception ex)
    {
        throw new ApplicationException("this program is not doing well", ex);
    }
}

// reference
private void btnReturn_Click(object sender, EventArgs e)
{
    new Login().Show();
    this.Hide();
}

```

```

146 }

// method to add the feedback to customer file

public void addRecord(string name, string contact, string email, string message, string date, string time, string feedback, string filepath)
{
    try
    {
        using (System.IO.StreamWriter file = new System.IO.StreamWriter(@filepath, true))
        {
            //file.WriteLine();
            file.WriteLine(name + "," + contact + "," + email + "," + message + "," + date + "," + time + "," + feedback + ",");
        }
        MessageBox.Show("Data Added", "Successful");
    }
    catch (Exception ex)
    {
        throw new ApplicationException("This program is not doing well", ex);
    }
}

// method to add the feedback to customer file

public void addRecord1(string feedback, string filepath)
{
    try
    {
        using (System.IO.StreamWriter file = new System.IO.StreamWriter(@filepath, true))
        {
            file.WriteLine(feedback + ",");
        }
    }
    catch (Exception ex)
    {
        throw new ApplicationException("this program is not doing well", ex);
    }
}

// reference
private void btnReturn_Click(object sender, EventArgs e)
{
    new Login().Show();
    this.Hide();
}

```

```

177     }
178 
179     private void btnReturn_Click(object sender, EventArgs e)
180     {
181         new Login().Show();
182         this.Hide();
183     }
184 
185     private void gridFeedback_CellContentClick(object sender, DataGridViewCellEventArgs e)
186     {
187         var loop = false;
188         if (loop == false)
189         {
190             loop = true;
191 
192             if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
193             {
194                 for (int i = 0; i < 3; i++)
195                 {
196                     if (i != e.ColumnIndex)
197                     {
198                         gridFeedback.Rows[(e.RowIndex)].Cells[i].Value = false;
199                     }
200                 }
201             }
202             gridFeedback.EndEdit();
203         }
204     }
205 }
206 }
207 }
208 }
209 }
210 }
211 }

```

3. Dashboard

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.IO;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 using System.Windows.Forms;
11 
12 namespace Rating_System
13 {
14     public partial class Dashboard : Form
15     {
16         public string CustomerFile = @"..\..\customer.csv";
17         public Dashboard()
18         {
19             InitializeComponent();
20         }
21 
22         private void btnReport_Click(object sender, EventArgs e)
23         {
24             OpenFileDialog dig = new OpenFileDialog();
25             dig.ShowDialog();
26             txtViewReport.Text = dig.FileName;
27             BindData(txtViewReport.Text);
28         }
29 
30         private void BindData(string filePath)
31         {
32             DataTable dt = new DataTable();
33             string[] lines = System.IO.File.ReadAllLines(filePath);
34             if (lines.Length > 0)
35             {
36                 string firstline = lines[0];
37                 string[] headerLabels = firstline.Split(',');
38                 foreach (string headerWord in headerLabels)
39                 {
40                     dt.Columns.Add(new DataColumn(headerWord));
41                 }
42             }
43         }
44     }
45 }

```

```

    string firstline = lines[0];
    string[] headerLabels = firstline.Split(',');
    foreach (string headerWord in headerLabels)
    {
        dt.Columns.Add(new DataColumn(headerWord));
    }

    for (int i = 1; i < lines.Length; i++)
    {
        string[] dataWords = lines[i].Split(',');
        DataRow dr = dt.NewRow();
        foreach (string headerWord in headerLabels)
        {
            dr[headerWord] = dataWords[columnIndex++];
        }
        dt.Rows.Add(dr);
    }

    if (dt.Rows.Count > 0)
    {
        gridReport.DataSource = dt;
    }
}

private void panel1_Paint(object sender, PaintEventArgs e)
{
    new Graph().Show();
    this.Hide();
}

private void btnGraph_Click(object sender, EventArgs e)
{
    new Graph().Show();
    this.Hide();
}

private void btnAddDetails_Click(object sender, EventArgs e)
{
    new Details().Show();
    this.Hide();
}

private void btnLogout_Click(object sender, EventArgs e)
{
    new Login().Show();
    this.Hide();
}

private void Dashboard_Load(object sender, EventArgs e)
{
}

```

```

    string firstline = lines[0];
    string[] headerLabels = firstline.Split(',');
    foreach (string headerWord in headerLabels)
    {
        dt.Columns.Add(new DataColumn(headerWord));
    }

    for (int i = 1; i < lines.Length; i++)
    {
        string[] dataWords = lines[i].Split(',');
        DataRow dr = dt.NewRow();
        foreach (string headerWord in headerLabels)
        {
            dr[headerWord] = dataWords[columnIndex++];
        }
        dt.Rows.Add(dr);
    }

    if (dt.Rows.Count > 0)
    {
        gridReport.DataSource = dt;
    }
}

private void panel1_Paint(object sender, PaintEventArgs e)
{
    new Graph().Show();
    this.Hide();
}

private void btnGraph_Click(object sender, EventArgs e)
{
    new Graph().Show();
    this.Hide();
}

private void btnAddDetails_Click(object sender, EventArgs e)
{
    new Details().Show();
    this.Hide();
}

private void btnLogout_Click(object sender, EventArgs e)
{
    new Login().Show();
    this.Hide();
}

private void Dashboard_Load(object sender, EventArgs e)
{
}

```

4. Details

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and Search (Ctrl+Q). The toolbar below has icons for New, Open, Save, Print, Find, Replace, Cut, Copy, Paste, Select All, Undo, Redo, and others. The main window displays the Details.cs code for the Rating System. The code defines a partial class Details that inherits from Form. It includes methods for handling button clicks (btnGraph_Click, btnReport_Click, btnLogout_Click, btnAddDetails_Click) and loading data (Details_Load). The code also contains logic for displaying different forms (Graph, Dashboard, Login) based on button interactions. The Solution Explorer on the right shows the project structure with files like App.config, Details.cs, Feedback.cs, Graph.cs, Login.cs, Program.cs, and Rating.cs.

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Drawing;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Windows.Forms;
9
10
11  namespace Rating_System
12  {
13      public partial class Details : Form
14      {
15          public string CriteriaFile = @"..\..\criteria.csv";
16
17          public Details()
18          {
19              InitializeComponent();
20          }
21
22          private void btnGraph_Click(object sender, EventArgs e)
23          {
24              new Graph().Show();
25              this.Hide();
26          }
27
28          private void btnReport_Click(object sender, EventArgs e)
29          {
30              new Dashboard().Show();
31              this.Hide();
32          }
33
34          private void btnLogout_Click(object sender, EventArgs e)
35          {
36              new Login().Show();
37              this.Hide();
38          }
39
40          private void btnAddDetails_Click(object sender, EventArgs e)
41          {
42              new Details().Show();
43              this.Hide();
44          }
45
46          private void Details_Load(object sender, EventArgs e)
47          {
48          }
49
50
51          private void btnAddCriteria_Click(object sender, EventArgs e)
52          {
53              string criteriaName = txtCriteria.Text;
54              AddRecord(criteriaName, CriteriaFile);
55          }
56
57          public void AddRecord(string criteria, string filepath)
58          {
59              try
60              {
61                  using (System.IO.StreamWriter file = new System.IO.StreamWriter(@filepath, true))
62                  {
63                      file.WriteLine();
64                      file.Write(criteria + "\n"); // add criteria to new line
65                  }
66                  MessageBox.Show("Criteria Added");
67              }
68              catch (Exception ex)
69              {
70                  throw new ApplicationException("This program is not doing well", ex);
71              }
72          }
73
74          private void btnClear_Click(object sender, EventArgs e)
75          {
76              txtCriteria.Clear();
77          }
78      }
79  }

```

This screenshot shows the same Microsoft Visual Studio interface as the previous one, but with a different section of the Details.cs code visible. The code now includes the implementation of the AddRecord method, which uses a StreamWriter to append new criteria names to a CSV file. It also shows the btnClear_Click event handler, which clears the text in the txtCriteria TextBox. The Solution Explorer on the right remains the same, showing the project structure.

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Drawing;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Windows.Forms;
9
10
11  namespace Rating_System
12  {
13      public partial class Details : Form
14      {
15          public string CriteriaFile = @"..\..\criteria.csv";
16
17          public Details()
18          {
19              InitializeComponent();
20          }
21
22          private void btnGraph_Click(object sender, EventArgs e)
23          {
24              new Graph().Show();
25              this.Hide();
26          }
27
28          private void btnReport_Click(object sender, EventArgs e)
29          {
30              new Dashboard().Show();
31              this.Hide();
32          }
33
34          private void btnLogout_Click(object sender, EventArgs e)
35          {
36              new Login().Show();
37              this.Hide();
38          }
39
40          private void btnAddDetails_Click(object sender, EventArgs e)
41          {
42              new Details().Show();
43              this.Hide();
44          }
45
46          private void Details_Load(object sender, EventArgs e)
47          {
48          }
49
50
51          private void btnAddCriteria_Click(object sender, EventArgs e)
52          {
53              string criteriaName = txtCriteria.Text;
54              AddRecord(criteriaName, CriteriaFile);
55          }
56
57          public void AddRecord(string criteria, string filepath)
58          {
59              try
60              {
61                  using (System.IO.StreamWriter file = new System.IO.StreamWriter(@filepath, true))
62                  {
63                      file.WriteLine();
64                      file.Write(criteria + "\n"); // add criteria to new line
65                  }
66                  MessageBox.Show("Criteria Added");
67              }
68              catch (Exception ex)
69              {
70                  throw new ApplicationException("This program is not doing well", ex);
71              }
72          }
73
74          private void btnClear_Click(object sender, EventArgs e)
75          {
76              txtCriteria.Clear();
77          }
78      }
79  }

```

5. Graph

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "Rating System". The main window displays the "Graph.cs [Design]" tab, which contains C# code for a Windows Form application. The code includes event handlers for button clicks (btReport_Click, btClose_Click, btLogout_Click, btAddDetails_Click) and a chart selection event (comboBoxGraph_SelectedIndexChanged). The code uses namespaces like System, System.Collections.Generic, System.ComponentModel, System.Data, System.Drawing, System.Linq, System.Text, System.Threading.Tasks, and System.Windows.Forms. The Solution Explorer on the right shows the project structure with files like Details.cs, Feedback.cs, Dashboard.cs, Graph.cs, Login.cs, and Program.cs.

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Rating_System
{
    public partial class Graph : Form
    {
        public Graph()
        {
            InitializeComponent();
        }

        private void btReport_Click(object sender, EventArgs e)
        {
            new Dashboard().Show();
            this.Hide();
        }

        private void btClose_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void btLogout_Click(object sender, EventArgs e)
        {
            new Login().Show();
            this.Hide();
        }

        private void btAddDetails_Click(object sender, EventArgs e)
        {
            new Details().Show();
        }
    }
}

```

This screenshot is identical to the one above, showing the Microsoft Visual Studio interface with the "Graph.cs [Design]" tab selected. The code in Graph.cs is identical, including the event handlers for button clicks and the chart selection event. The Solution Explorer shows the same project structure with files like Details.cs, Feedback.cs, Dashboard.cs, Graph.cs, Login.cs, and Program.cs.

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Rating_System
{
    public partial class Graph : Form
    {
        public Graph()
        {
            InitializeComponent();
        }

        private void btAddDetails_Click(object sender, EventArgs e)
        {
            new Details().Show();
            this.Hide();
        }

        private void btGraph_Click(object sender, EventArgs e)
        {
            new Graph().Show();
            this.Hide();
        }

        private void Graph_Load(object sender, EventArgs e)
        {
            FoodQualityChart.Hide();
            StaffFriendlinessChart.Hide();
            CleanlinessChart.Hide();
            OrderAccuracyChart.Hide();
            RestaurantBalanceChart.Hide();
            ValueForMoneyChart.Hide();
        }

        string[] raw_text = System.IO.File.ReadAllLines(@"..\..\customer.csv");
        string[] data_col = null;
        bool foodqualitycount = true;
        bool stafffriendlinesscount = true;
        bool cleanlinesscount = true;
        bool orderaccuracycount = true;
        bool restaurantbalancecount = true;
        bool valueformoneycount = true;

        private void comboBoxGraph_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBoxGraph.SelectedIndex == 0)
            {
                FoodQualityChart.Show();
                StaffFriendlinessChart.Hide();
                CleanlinessChart.Hide();
                OrderAccuracyChart.Hide();
            }
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Windows.DataVisualization.Charting;

namespace Rating_System
{
    public partial class Graph : Window
    {
        FoodQualityChart.Show();
        StaffFriendlinessChart.Hide();
        CleanlinessChart.Hide();
        CustomerSatisfactionChart.Hide();
        RectangleQualityChart.Hide();
        ValueForMoneyChart.Hide();

        if (FoodQualityCount == true)
        {
            int foodQuality1 = 0;
            int foodQuality2 = 0;
            int foodQuality3 = 0;
            int foodQuality4 = 0;

            foreach (string text_line in raw_text)
            {
                data_col = text_line.Split(',');
                for (int i = 6; i < 8; i++)
                {
                    if (data_col[i] == "1")
                    {
                        foodQuality1++;
                    }
                    else if (data_col[i] == "2")
                    {
                        foodQuality2++;
                    }
                    else if (data_col[i] == "3")
                    {
                        foodQuality3++;
                    }
                    else if (data_col[i] == "4")
                    {
                        foodQuality4++;
                    }
                }
            }

            FoodQualityChart.Series["Food Quality"].Points.Add("1", #FoodQuality1);
            FoodQualityChart.Series["Food Quality"].Points.Add("2", #FoodQuality2);
            FoodQualityChart.Series["Food Quality"].Points.Add("3", #FoodQuality3);
            FoodQualityChart.Series["Food Quality"].Points.Add("4", #FoodQuality4);

            foodQualityCount = false;
        }

        if (comboBoxGraph.SelectedIndex == 1)
        {
            StaffFriendlinessChart.Show();
            FoodQualityChart.Hide();

            if (StaffFriendlinessCount == true)
            {
                int StaffFriendliness1 = 0;
                int StaffFriendliness2 = 0;
                int StaffFriendliness3 = 0;
                int StaffFriendliness4 = 0;

                foreach (string text_line in raw_text)
                {
                    data_col = text_line.Split(',');
                    for (int i = 7; i < 9; i++)
                    {
                        if (data_col[i] == "1")
                        {
                            StaffFriendliness1++;
                        }
                        else if (data_col[i] == "2")
                        {
                            StaffFriendliness2++;
                        }
                        else if (data_col[i] == "3")
                        {
                            StaffFriendliness3++;
                        }
                        else if (data_col[i] == "4")
                        {
                            StaffFriendliness4++;
                        }
                    }
                }

                StaffFriendlinessChart.Series["Staff Friendliness"].Points.Add("1", #StaffFriendliness1);
                StaffFriendlinessChart.Series["Staff Friendliness"].Points.Add("2", #StaffFriendliness2);
                StaffFriendlinessChart.Series["Staff Friendliness"].Points.Add("3", #StaffFriendliness3);
                StaffFriendlinessChart.Series["Staff Friendliness"].Points.Add("4", #StaffFriendliness4);
            }
        }
    }
}

```


The screenshot displays two separate instances of Microsoft Visual Studio, both showing the 'Rating System' solution in the Solution Explorer. The top instance shows the 'Graph.cs' file with the following code:

```
if (data_col[1] == "1")
{
    OrderAccuracy1++;
}
else if (data_col[1] == "2")
{
    OrderAccuracy2++;
}
else if (data_col[1] == "3")
{
    OrderAccuracy3++;
}
else if (data_col[1] == "4")
{
    OrderAccuracy4++;
}

OrderAccuracyChart.Series["Order Accuracy"].Points.AddXY("1", #OrderAccuracy1);
OrderAccuracyChart.Series["Order Accuracy"].Points.AddXY("2", #OrderAccuracy2);
OrderAccuracyChart.Series["Order Accuracy"].Points.AddXY("3", #OrderAccuracy3);
OrderAccuracyChart.Series["Order Accuracy"].Points.AddXY("4", #OrderAccuracy4);
OrderAccuracyCount = false;
}

if (comboBoxGraph.SelectedIndex == 4)
{
    RestaurantAmbianceChart.Show();
    FoodQualityChart.Hide();
    StaffFriendlinessChart.Hide();
    CleanlinessChart.Hide();
    OrderAccuracyChart.Hide();

    if (RestaurantAmbianceCount == true)
    {
        int RestaurantAmbiance1 = 0;
        int RestaurantAmbiance2 = 0;
        int RestaurantAmbiance3 = 0;
        int RestaurantAmbiance4 = 0;

        foreach (string text_line in raw_text)
        {
            data_col = text_line.Split(',');
            for (int i = 10; i < 10; i++)
            {

                if (data_col[i] == "1")
                {
                    RestaurantAmbiance1++;
                }
                else if (data_col[i] == "2")
                {
                    RestaurantAmbiance2++;
                }
                else if (data_col[i] == "3")
                {
                    RestaurantAmbiance3++;
                }
                else if (data_col[i] == "4")
                {
                    RestaurantAmbiance4++;
                }
            }

            RestaurantAmbianceChart.Series["Restaurant Ambiance"].Points.AddXY("1", #RestaurantAmbiance1);
            RestaurantAmbianceChart.Series["Restaurant Ambiance"].Points.AddXY("2", #RestaurantAmbiance2);
            RestaurantAmbianceChart.Series["Restaurant Ambiance"].Points.AddXY("3", #RestaurantAmbiance3);
            RestaurantAmbianceChart.Series["Restaurant Ambiance"].Points.AddXY("4", #RestaurantAmbiance4);
            RestaurantAmbianceCount = false;
        }
    }
}
```

The bottom instance shows the same code with a 'TextLine' method added to the 'Rating_System.Graph' class:

```
FoodQualityChart.Hide();
StaffFriendlinessChart.Hide();
CleanlinessChart.Hide();
OrderAccuracyChart.Hide();

if (RestaurantAmbianceCount == true)
{
    int RestaurantAmbiance1 = 0;
    int RestaurantAmbiance2 = 0;
    int RestaurantAmbiance3 = 0;
    int RestaurantAmbiance4 = 0;

    foreach (string text_line in raw_text)
    {
        data_col = text_line.Split(',');
        for (int i = 10; i < 10; i++)
        {

            if (data_col[i] == "1")
            {
                RestaurantAmbiance1++;
            }
            else if (data_col[i] == "2")
            {
                RestaurantAmbiance2++;
            }
            else if (data_col[i] == "3")
            {
                RestaurantAmbiance3++;
            }
            else if (data_col[i] == "4")
            {
                RestaurantAmbiance4++;
            }
        }

        RestaurantAmbianceChart.Series["Restaurant Ambiance"].Points.AddXY("1", #RestaurantAmbiance1);
        RestaurantAmbianceChart.Series["Restaurant Ambiance"].Points.AddXY("2", #RestaurantAmbiance2);
        RestaurantAmbianceChart.Series["Restaurant Ambiance"].Points.AddXY("3", #RestaurantAmbiance3);
        RestaurantAmbianceChart.Series["Restaurant Ambiance"].Points.AddXY("4", #RestaurantAmbiance4);
        RestaurantAmbianceCount = false;
    }
}
```

```

1 // This is a sample C# script showing how to read a CSV file and plot it into a chart.
2 // The CSV file has columns: Name, Phone No, Email, Message, Date, Time, FoodQuality, Staff Frien Cleanlines Order Acc Restauran Value for Money
3 // The chart will have 6 series: FoodQuality, Staff Friendliness, Cleanliness, Order Accuracy, Restaurant Ambience, and Value for Money.
4
5 using System;
6 using System.IO;
7 using System.Linq;
8 using System.Collections.Generic;
9 using System.Windows.Forms.DataVisualization.Charting;
10
11 namespace Rating_System
12 {
13     public class Rating_System_Graph
14     {
15         public void ReadCSV(string filePath)
16         {
17             string rawText = File.ReadAllText(filePath);
18
19             foreach (string textLine in rawText)
20             {
21                 string[] data_col = textLine.Split(',');
22
23                 int valueForMoneyCount = 0;
24
25                 if (data_col[6] == "1")
26                 {
27                     int valueForMoney1 = 0;
28                     int valueForMoney2 = 0;
29                     int valueForMoney3 = 0;
30                     int valueForMoney4 = 0;
31
32                     if (valueForMoneyCount == true)
33                     {
34                         int valueForMoney1 = 0;
35                         int valueForMoney2 = 0;
36                         int valueForMoney3 = 0;
37                         int valueForMoney4 = 0;
38
39                         foreach (string text_line in rawText)
40                         {
41                             data_col = text_line.Split(',');
42
43                             for (int i = 0; i < data_col.Length; i++)
44                             {
45
46                                 if (data_col[i] == "1")
47                                 {
48                                     valueForMoney1++;
49
50                                 }
51                                 else if (data_col[i] == "2")
52                                 {
53                                     valueForMoney2++;
54
55                                 }
56                                 else if (data_col[i] == "3")
57                                 {
58                                     valueForMoney3++;
59
60                                 }
61                                 else if (data_col[i] == "4")
62                                 {
63                                     valueForMoney4++;
64
65                                 }
66
67                             valueForMoneyChart.Series["Value for Money"].Points.AddXY("1", valueForMoney1);
68                             valueForMoneyChart.Series["Value for Money"].Points.AddXY("2", valueForMoney2);
69                             valueForMoneyChart.Series["Value for Money"].Points.AddXY("3", valueForMoney3);
70                             valueForMoneyChart.Series["Value for Money"].Points.AddXY("4", valueForMoney4);
71                             valueForMoneyCount = false;
72
73                         }
74
75                     }
76
77                 }
78
79             }
80
81         }
82
83     }
84
85 }
86
87 
```

6. Customer.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Name	Phone No	Email	Message	Date	Time	FoodQuality	Staff Frien	Cleanlines	Order Acc	Restauran	Value for Money						
2	Shrijana K	9847823456	shrijanak	It was goo	22/01/2021	10:16	3	2	1	2	1	2						
3	Bidhan Pa	9851234578	bidhanpar	It was a g	22/01/2021	10:26	1	2	1	3	1	2						
4	Kiran BC	9851016799	kiran@gmail.com		22/01/2021	10:35	3	4	3	1	3	2						
5																		
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		
17																		
18																		
19																		
20																		
21																		
22																		
23																		

7. Criteria.csv

The screenshot shows a Microsoft Excel spreadsheet titled "criteria - Microsoft Excel". The "HOME" tab is selected. The data is listed in column A from row 1 to row 7:

	Food Quality
1	Food Quality
2	Staff Friendliness
3	Cleanliness
4	Order Accuracy
5	Restaurant Ambiance
6	Value For Money
7	Parking

The status bar at the bottom shows the date as 22/01/2021 and the time as 13:00.