

Table of Contents

1. Introduction	1
2. Background	2
2.1 User Manual	2
i. Login form	2
ii. Feedback form	3
iii. Staff Login form	4
iv. General form	5
v. Report form	6
vi. Import form	7
vii. Admin form	8
3. Development	10
3.1 System Architecture	10
3.2 Class diagram	11
3.3 Description of Classes' properties and methods	12
3.4 System Algorithm	20
3.4.1 Login Form	20
3.4.2 Staff Login Form	20
3.4.3 Feedback Form	21
3.4.4 General Form	22
3.4.5 Import Form	22
3.4.6 Report Form	23
3.4.7 Admin Form	24
3.5 Flowchart	26
3.5.1 Login form	26
3.5.2 Staff Login form	27
3.5.3 Feedback form	28
3.5.4 General form	29
3.5.5 Import form	30
3.5.6 Report form	31
3.5.7 Admin form	32
3.6 Use case diagram	33
3.7 Data structures and algorithms used	34
4. Reflection	35
References	36
Appendix	37

Table of Figure

Figure 1: Login form.....	2
Figure 2: Feedback form	3
Figure 3: Clicking the submit button.....	3
Figure 4: Staff Login form	4
Figure 5: General form.....	5
Figure 6: Report form.....	6
Figure 7: Displaying table and chart.....	6
Figure 8: Import form	7
Figure 9: Importing the csv file	7
Figure 10: Admin form	8
Figure 11: Adding the new criteria	8
Figure 12: Deleting the entered criteria.....	8
Figure 13: Updating the criteria.....	9
Figure 14: System Architecture.....	10
Figure 15: Class diagram.....	11
Figure 16: Flowchart of Login form	26
Figure 17: Flowchart of Staff Login form	27
Figure 18: Flowchart of Feedback form	28
Figure 19: Flowchart of General form	29
Figure 20: Flowchart of Import form	30
Figure 21: Flowchart of Report form	31
Figure 22: Flowchart of Admin form	32
Figure 23: Use case diagram.....	33

Table of Tables

Table 1: Login class	12
Table 2: LoginAG class.....	13
Table 3: CustomerFeedback class.....	15
Table 4: Admin class.....	16
Table 5: General class	17
Table 6: Report class	18
Table 7: ImportBulkData class	19

1. Introduction

In this modern generation, technology plays an important role in our daily life. People are more dependent on technologies as it makes our work faster and easier. Hence, computer based system are preferred more than the paper based system. With the help of computer based system, records can be stored and accessed easily.

In this coursework, our task was to create a feedback/rating system in Visual studio using C# programming language. A feedback system is an application or a software which helps to manage what customers are saying such as online reviews, ratings, comments etc. (Bassig, 2020). This application must allow the admin to input criteria to give feedback and to rate the services. The application should also allow the customers to input the feedback/rating point where Excellent is 5 points, Good is 3 points, Average is 2 points and Dissatisfied is 1 point. This application is developed to keep track of the feedbacks given by the customers.

In my system, there are three users i.e. customer, admin and general. The customer manually provides the feedback/rating data using system. The admin can add, delete and update criteria. The general can display the rating of the restaurant services given by the customer and chart showing the total rating of the restaurant services.

2. Background

2.1 User Manual

The detailed instructions to run the system along with the screenshots is shown below.

Firstly, open the “cw1” file in the Visual studio and run the program.

i. Login form

At first, login form displays on the screen. If the user is customer, then click on the “Customer” button.

If the user is admin or general, then click on the “Staff” button.

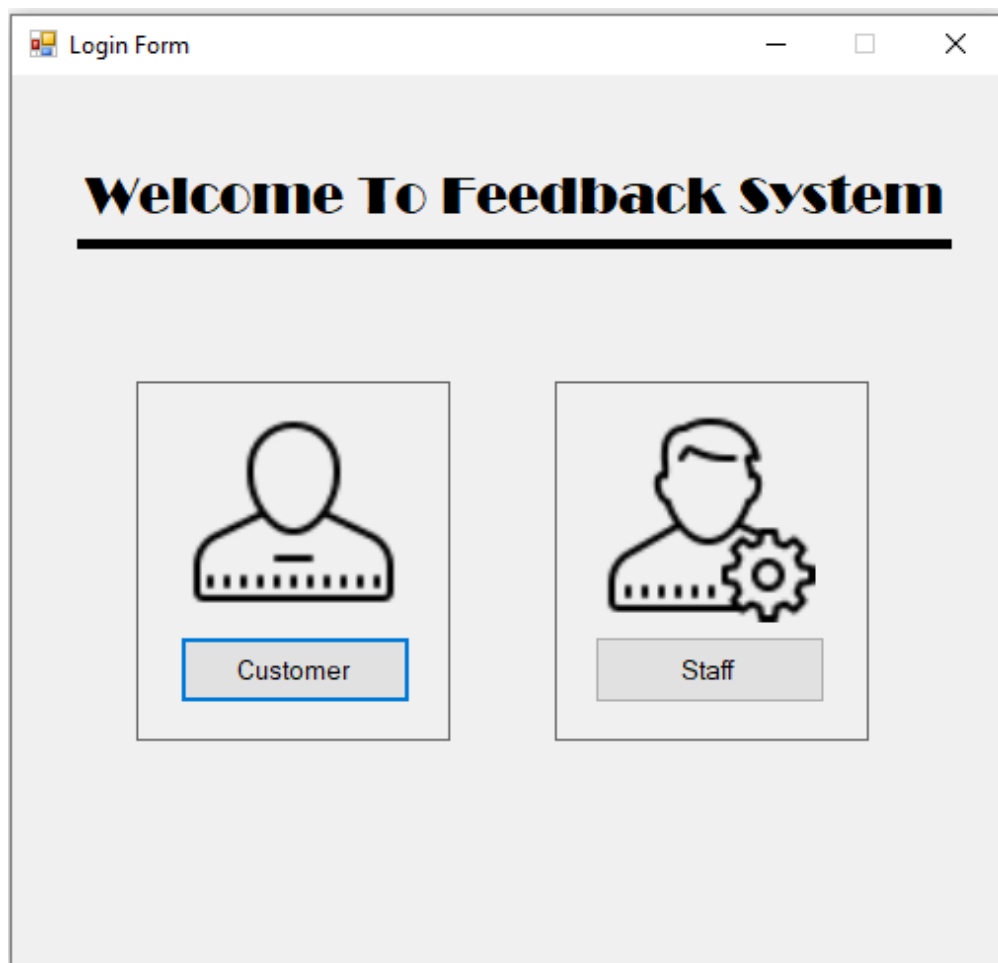


Figure 1: Login form

ii. Feedback form

When “Customer” button is clicked, customer feedback form is displayed on the screen where the customer gives their rating/feedback. If the user wants to go back to the login form, click on the “Exit” button which is located at the top right corner of the screen. Click the “Clear” button to clear the input text fields. If the user wants to retrieve data, click on the “Retrieve” button. Real time and date are also displayed on the screen.

Customer Feedback Form

Date : 23-01-2021
Time : 10-04-14 PM

Service Name
Service 1

Customer Name
Contact

Email

Overall feedback for service

Retrieve

Criteria	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 Ambiance	<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"/>

Submit Clear

Figure 2: Feedback form

After entering all the details, click on the “Submit” button.

If the users want to stay anonymous, they can leave the details like customer name, contact, email and overall feedback empty.

Customer Feedback Form

Date : 23-01-2021
Time : 10-04-14 PM

Service Name
Service 1

Customer Name
Rosnee

Contact
9837348212

Email
rosnee@gmail.com

Overall feedback for service
Excellent

Retrieve

Criteria	Excellent	Good	Average	Dissatisfied
Food Quality	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Staff Friendliness	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Cleanliness	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Order Accuracy	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Restaurant Ambiance	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Value for Money	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Submit Clear

Figure 3: Clicking the submit button

iii. Staff Login form

When “Staff” button is clicked, staff login form displays on the screen. If the user wants to go back to the login form, click on the “Back” button.

- If the user is Admin, username is “admin” and password is “123”.
- If the user is General, username is “general” and password is “123”.

If the user wants to save the username and password, check the “Remember me” box.

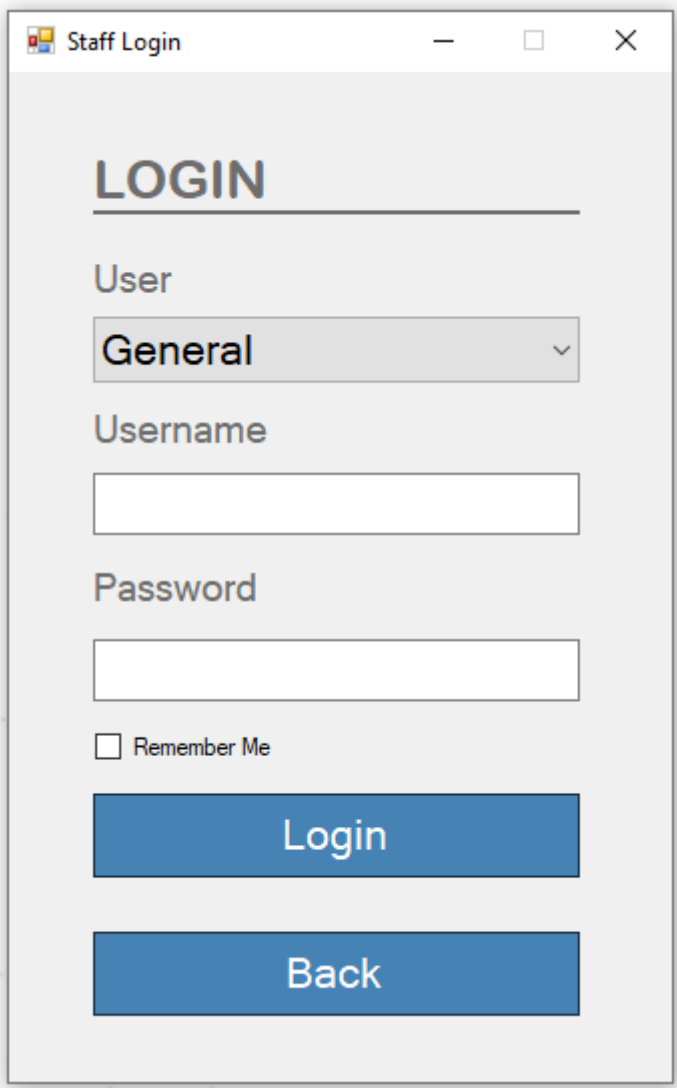
The image shows a screenshot of a web application window titled "Staff Login". The window has a light gray background and a white title bar with standard minimize, maximize, and close buttons. The main content area is also light gray. At the top, the word "LOGIN" is displayed in a large, bold, dark blue font, underlined. Below this, there are three input fields: a dropdown menu labeled "User" with "General" selected, a text input field labeled "Username", and another text input field labeled "Password". Below the password field is a checkbox labeled "Remember Me". At the bottom of the form are two large, blue rectangular buttons with white text: "Login" and "Back".

Figure 4: Staff Login form

iv. General form

If the user is “General”, General form displays on the screen. If the user wants to go back to the staff login form, click the “Back” button. If the user wants to display the report, click on the “Report” button. If the user wants to import the customer feedback/rating data from a text file (.csv), click the “Import” button.

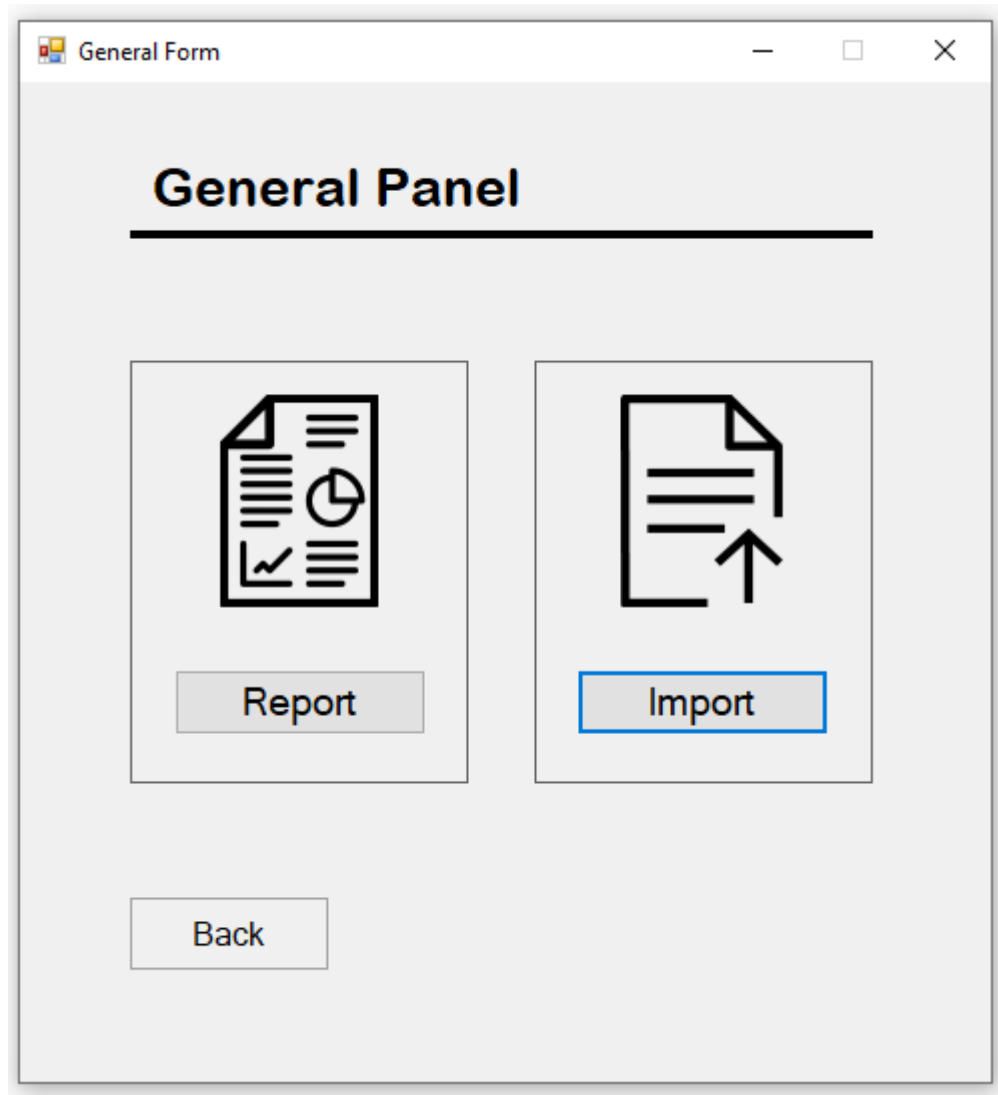


Figure 5: General form

v. Report form

When the “Report” button is clicked, the Report form displays on the screen. If the user wants to go back to the General form, enter the click the “Exit” menu.

Figure 6: Report form

To open the .csv file, click on the file menu and then click on the “Open” menu. Open dialog box displays on the screen. Then select the file that you want to display. If the file is valid, data of the csv file displays on grid view table. If the file is invalid, error message displays on the screen. To view pie chart, click on the “Chart” button.

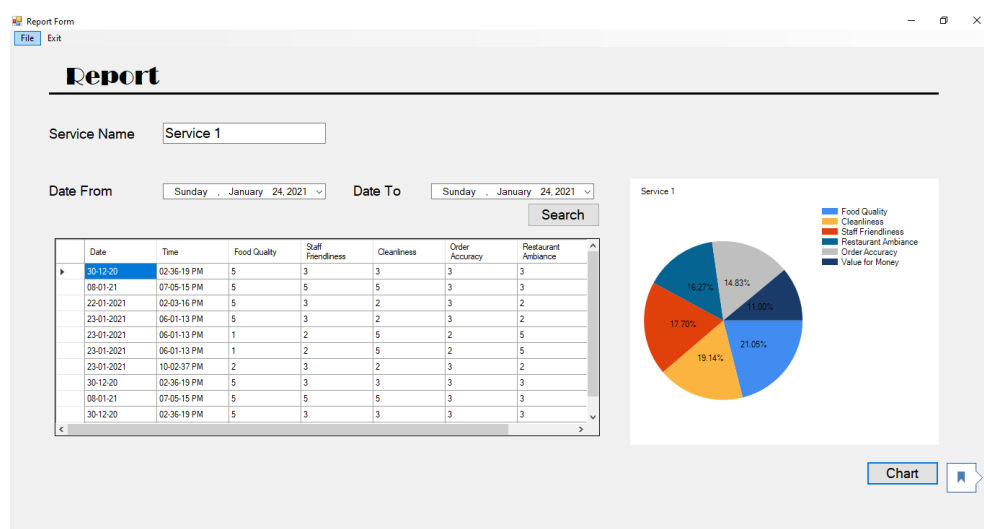


Figure 7: Displaying table and chart

vi. Import form

When the “Import” button is clicked, the Import form displays on the screen. If the user wants to go back to the General form, enter the click the “Exit” button. Click the “Clear” button to clear the input text fields.

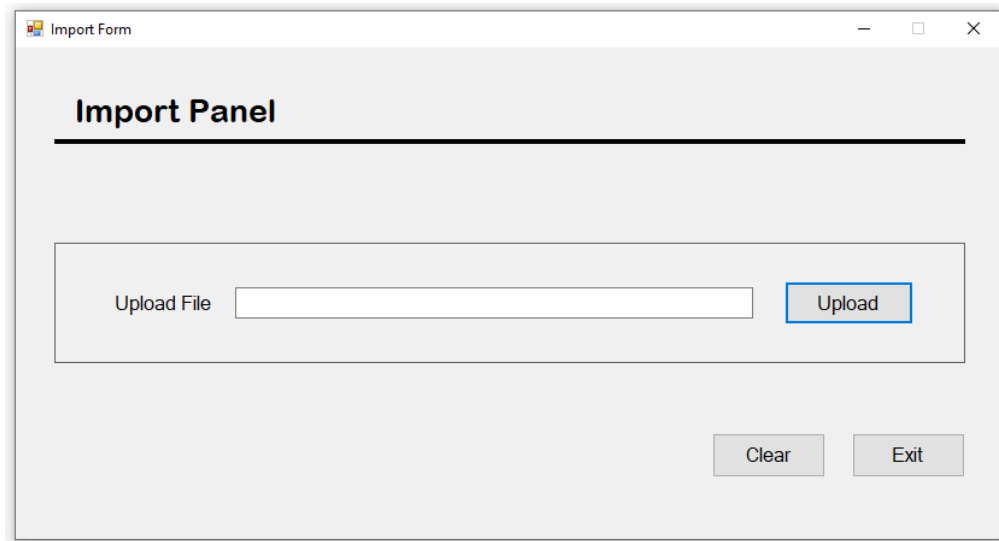
The screenshot shows a window titled "Import Form" with a standard Windows title bar (minimize, maximize, close buttons). The main content area has a header "Import Panel" followed by a horizontal line. Below this is a form container with a label "Upload File" next to a text input field. To the right of the input field is a blue "Upload" button. At the bottom right of the window are two buttons: "Clear" and "Exit".

Figure 8: Import form

To import the file, click on the “Upload” button. Open dialog box displays on the screen. Then select the file that you want to import. If the file is valid, customer feedback/rating data from the selected csv file is imported. If the file is invalid, error message displays on the screen.

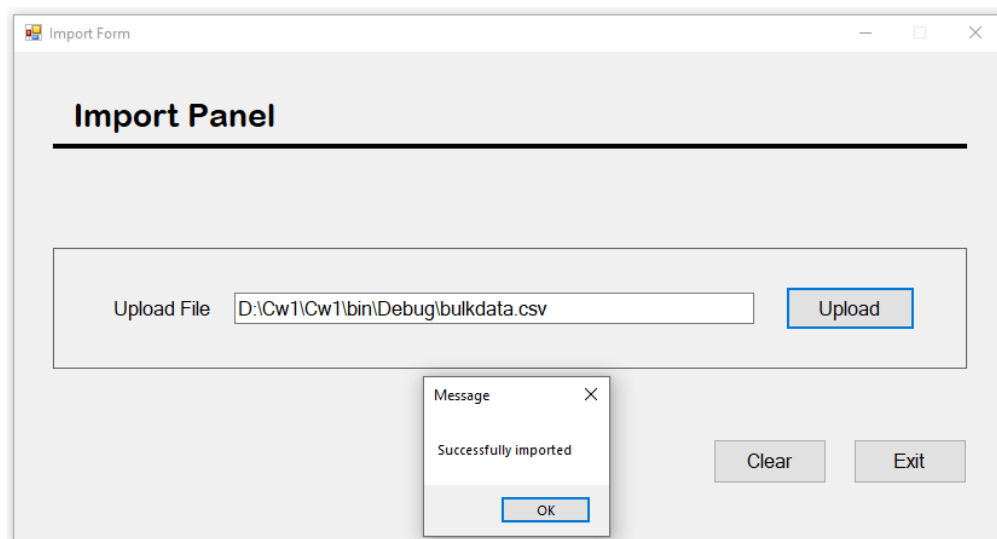
This screenshot shows the same "Import Form" window as Figure 8, but with the text input field containing the file path "D:\Cw1\Cw1\bin\Debug\bulkdata.csv". A small "Message" dialog box is open in the foreground, displaying the text "Successfully imported" and an "OK" button. The "Upload" button is still visible to the right of the input field, and the "Clear" and "Exit" buttons are at the bottom right.

Figure 9: Importing the csv file

vii. Admin form

If the user is “Admin”, Admin form displays on the screen. If the user wants to go back to the login form, enter the “Exit” button. Click the “Clear” button to clear the input text fields.

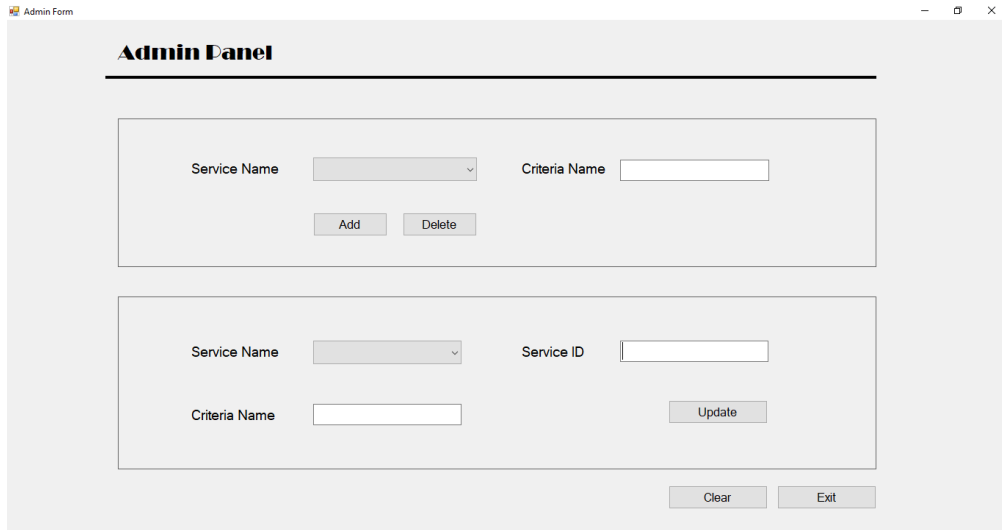
The screenshot shows a web application window titled "Admin Form". Inside, there's a section titled "Admin Panel". Below this, there are two main form areas. The first area contains a "Service Name" dropdown menu and a "Criteria Name" text input field, with "Add" and "Delete" buttons below them. The second area contains a "Service Name" dropdown menu, a "Service ID" text input field, and a "Criteria Name" text input field, with an "Update" button below them. At the bottom of the form, there are "Clear" and "Exit" buttons.

Figure 10: Admin form

Select the service name and enter the criteria name.

- Add: To add new criteria.

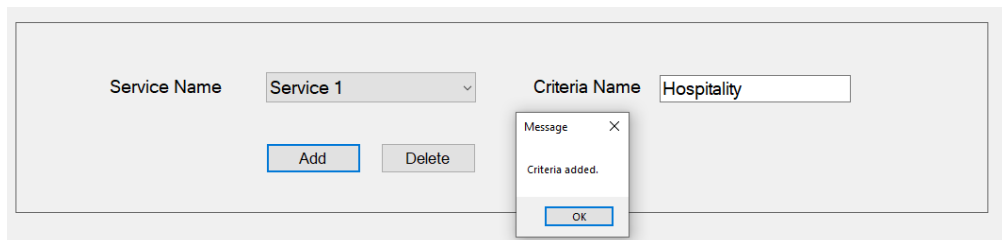
This screenshot shows the "Add" button highlighted in blue. A small "Message" dialog box is open in the center, displaying the text "Criteria added." with an "OK" button.

Figure 11: Adding the new criteria

- Delete: To delete the entered criteria.

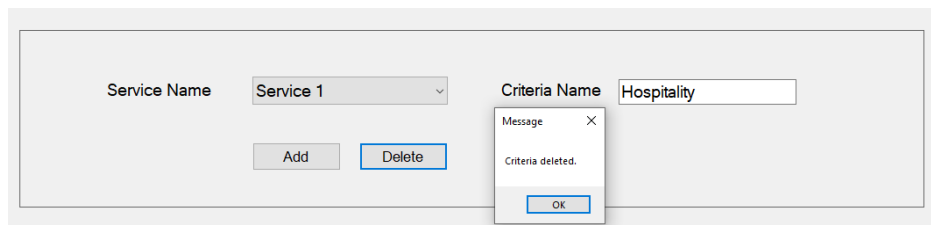
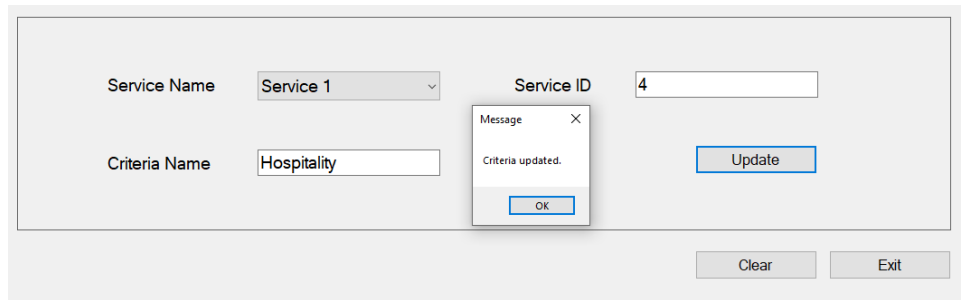
This screenshot shows the "Delete" button highlighted in blue. A small "Message" dialog box is open in the center, displaying the text "Criteria deleted." with an "OK" button.

Figure 12: Deleting the entered criteria

Select the service name, enter the criteria id and criteria name.

- Update: To update the existing criteria to entered criteria.



The screenshot shows a web application interface for updating criteria. It features a form with two rows: 'Service Name' with a dropdown menu showing 'Service 1', and 'Criteria Name' with a text input field containing 'Hospitality'. To the right, there is a 'Service ID' label and a text input field containing '4'. Below the 'Criteria Name' field is an 'Update' button. A modal dialog box titled 'Message' is open in the center, displaying the text 'Criteria updated.' and an 'OK' button. At the bottom right of the form, there are 'Clear' and 'Exit' buttons.

Figure 13: Updating the criteria

3. Development

3.1 System Architecture

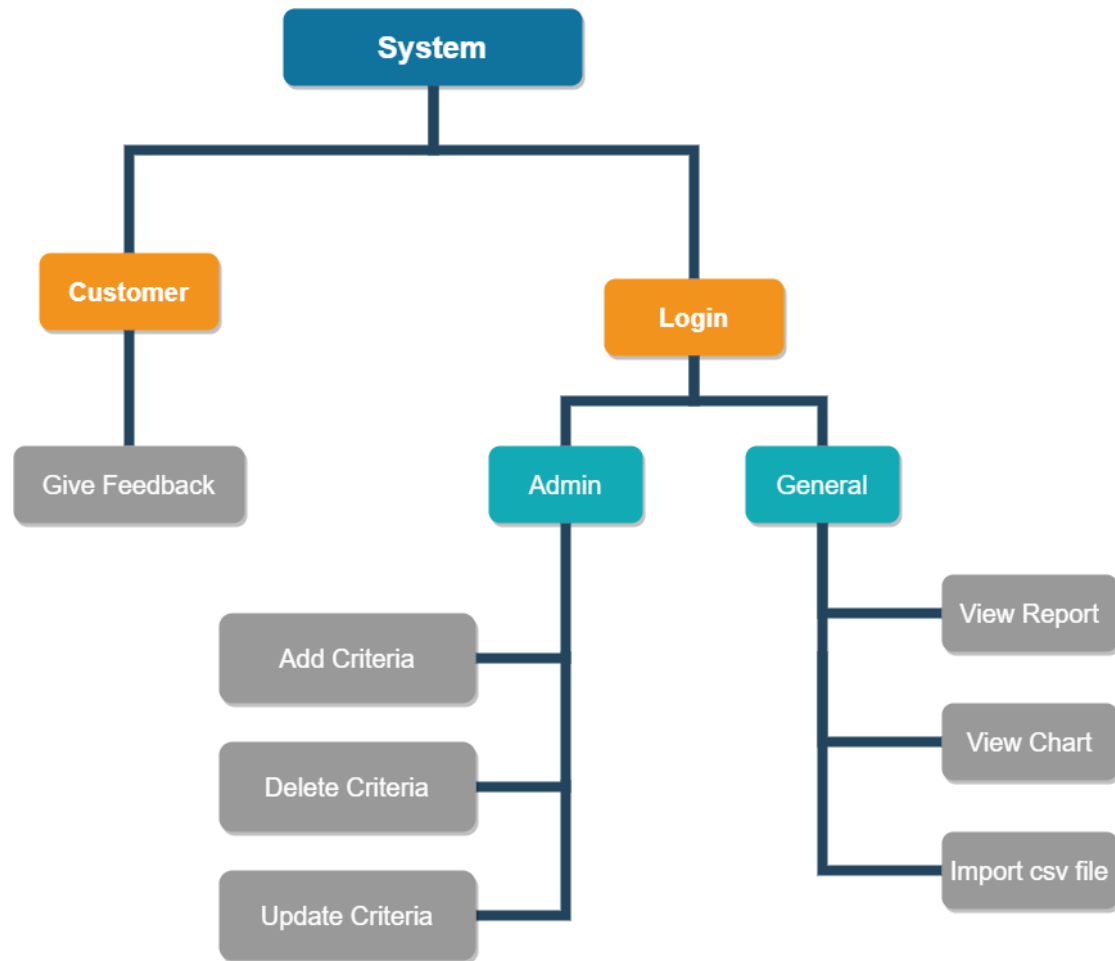


Figure 14: System Architecture

The above figure is the system architecture of “Customer Feedback System”. The system is divided into three different actors i.e. Customer, Admin and General. Among the actors, admin and general have to login in to access the system. Customers can directly access the feedback form page to provide feedback. The functionalities like adding, deleting and updating criteria is done by Admin while viewing report and chart is done by general.

3.2 Class diagram

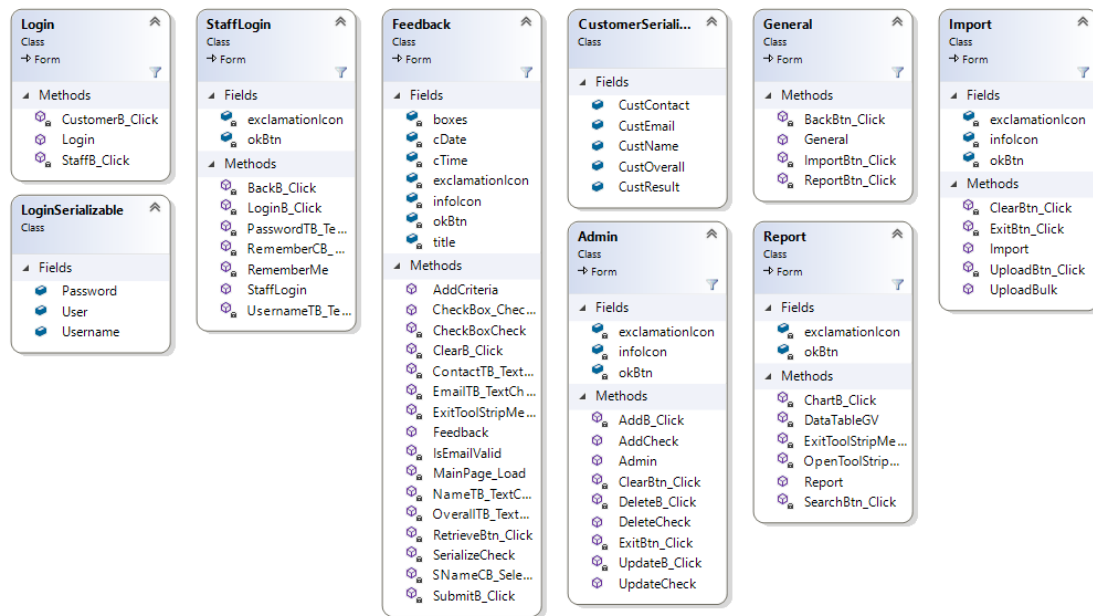


Figure 15: Class diagram

3.3 Description of Classes' properties and methods

Classes	Event Methods	Description
Login	StaffB_Click()	This method runs when the "Staff" button is clicked. This method hides the current form i.e. Login and redirects the user to the Staff Login form.
	CustomerB_Click()	This method runs when the "Customer" button is clicked. This method hides the current form i.e. Login and redirects the user to the Feedback form.

Table 1: Login class

Classes	Methods	Description
Staff Login	RememberMe()	This method runs when there is changes in the textboxes or the "Remember me" checkbox is checked or unchecked. When the user checks the checkbox, selected user type and entered username and password are stored in a csv file (loginData.csv) using serialization.
	Event Methods	Description
	LoginB_Click()	This method runs when the "Login" button is clicked. The user selects the user type and enters username and password. If the inputs are correct and the user type is General, the user is directed to the General form. If the user type is Admin, the user is directed to the Admin form.
	BackB_Click()	This method runs when the "Back" button is clicked. This method hides the

		current form i.e. Staff Login and redirects the user to the Login form.
	RememberCB_CheckedChanged()	This method runs when state of the checkbox changes.
	UsernameTB_Text Changed	This method runs when the text of the “username” textbox changes.
	PasswordTB_Text Changed	This method changes the characters of the text written in the text field to “*” sign.

Table 2: LoginAG class

Classes	Method	Description
Feedback	AddCriteria()	When this method is called, the checkboxes and labels (criteria name) displays on the layout table panel.
	IsEmailValid()	This method runs when the “Submit” button is clicked. If the email address is invalid, it returns false. If the email is valid, it returns true.
	CheckBoxCheck()	This method runs when the “Submit” button is clicked. It checks the state of the checkboxes.
	SerializeCheck()	This method runs when all the textboxes of this form are not empty. This method stores the input data into the csv file using serialization.
	Event Methods	Description
	SNameCB_Selected IndexChanged()	This method runs when the selected item of the combo box changes. The items are “Service 1” and “Service 2”. When the selected item is “Service 1”,

		the path is “criteriaservice1.csv”. When the selected item is “Service 2”, the “path” is “criteriaservice2.csv”. Then “AddCriteria” method with “path” as the parameter is called.
	CheckBox_Checked Changed()	This method runs when the check boxes’ state is checked or unchecked.
	NameTB_Text Changed()	This method runs when the text of the “name” textbox changes.
	EmailTB_Text Changed()	This method runs when the text of the “email” textbox changes.
	OverallTB_Text Changed()	This method runs when the text of the “overall” textbox changes.
	ContactTB_Text Changed()	This method runs when the text of the “contact” textbox changes.
	SubmitB_Click()	This method runs when the “Submit” button is clicked. After clicking the button, if the checkboxes is not checked, error message displays on the screen. If checked, the details are stored in the .csv file.
	ExitToolStripMenu Item_Click()	This method runs when the “Exit” menu item is clicked. After clicking this item, dialog box displays on the screen. If the user clicks “Yes” button, this form i.e. Feedback hides and the user is directed to the Login form.

	ClearB_Click()	This method runs when the “Clear” button is clicked. This method clears the textbox and unchecks the checkboxes.
	RetrieveBtn_Click()	This method runs when the “Retrieve” button is clicked. This method retrieves the data from the csv file to the textboxes using serialization.

Table 3: CustomerFeedback class

Classes	Methods	Description
Admin	AddCheck()	This method runs when the “Add” button is clicked. When service name is not selected and the textbox is empty, this method returns false value else it returns true value.
	DeleteCheck()	This method runs when the “Delete” button is clicked. When service name is not selected and the textbox is empty, this method returns false value else it returns true value.
	UpdateCheck()	This method runs when the “Update” button is clicked. When service name is not selected and the textboxes is empty, this method returns false value else it returns true value.
	Event Methods	Description
	AddB_Click()	This method runs when the “Add” button is clicked. The user selects the service name and enters the criteria name. If the criteria already exists, error message

		displays on the screen. If not, criteria is added in the csv file of criteria.
	DeleteB_Click()	This method runs when the “Delete” button is clicked. The user selects the service name and enters the criteria name. If the criteria exists, it is deleted. If not, error message displays on the screen.
	UpdateB_Click()	This method runs when the “Update” button is clicked. The user selects the service name and enters the criteria id and criteria name. If the criteria id exists, the criteria name respective to the id is updated to the entered criteria name. If not, error message displays on the screen.
	ExitBtn_Click()	This method runs when the “Exit” button is clicked. After clicking this button, dialog box displays on the screen. If the user clicks “Yes” button, this form i.e. Admin hides and the user is directed to the Staff Login form.
	ClearBtn_Click()	This method runs when the “Clear” button is clicked. This method clears the textboxes and unselects the combo boxes.

Table 4: Admin class

Classes	Event Methods	Description
General	ReportBtn_Click()	This method runs when the “Report” button is clicked. This method hides the current form i.e. General and redirects the user to the Report form.
	ImportBtn_Click()	This method runs when the “Import” button is clicked. This method hides the current form i.e. General and redirects the user to the Import form.
	BackBtn_Click()	This method runs when the “Back” button is clicked. This method hides the current form i.e. General and redirects the user to the Staff Login form.

Table 5: General class

Classes	Methods	Description
Report	DataTableGV()	This method runs when the user opens a valid csv file. This method displays the data of the csv file in the data grid view.
	Event Methods	Description
	OpenToolStripMenu Item_Click()	This method runs when the “Open” menu item is clicked. After clicking this item, open dialog box displays on the screen. When the user clicks on the valid csv file, the path of the file is stored in a variable. Then, “DataTableGV” method with path of the selected file as the parameter is called.
	ExitToolStripMenu Item_Click()	This method runs when the “Exit” menu item is clicked. After clicking this item, dialog box displays on the screen. If the user clicks “Yes” button, this form i.e. Report hides and the user is directed to the General form.
	ChartB_Click()	This method runs when the “Chart” button is clicked. This method displays the pie chart using the data from the csv file.
	SearchBtn_Click()	This methods sorts the data according to the date that is picked by the user.

Table 6: Report class

Classes	Methods	Description
Import	UploadBulk()	This method runs when the user opens a valid bulk csv file. This method exports the data from the bulk csv file to the .csv file of the customer details.
	Event Methods	Description
	UploadBtn_Click()	This method runs when the “Upload” button is clicked. After clicking this button, open dialog box displays on the screen. When the user clicks on the valid csv file, the path of the file is stored in a variable. Then, “UploadBulk” method with path of the selected file as the parameter is called.
	ExitBtn_Click()	This method runs when the “Exit” button is clicked. After clicking this button, dialog box displays on the screen. If the user clicks “Yes” button, this form i.e. Import hides and the user is directed to the General form.
	ClearBtn_Click()	This method runs when the “Clear” button is clicked. This method clears the textbox.

Table 7: ImportBulkData class

3.4 System Algorithm

3.4.1 Login Form

Step 1: Start

Step 2: Load Login Form.

Step 3: Click the “Customer” button or “Staff” button.

Step 4: If the “Customer” button is clicked, the user is redirected to the “Feedback form”.

Go to step 6.

Step 5: If the “Staff” button is clicked, the user is redirected to the “Staff login form”.

Step 6: End

3.4.2 Staff Login Form

Step 1: Start

Step 2: Load Staff login form.

Step 3: Select the user i.e. General or Admin.

Step 4: Enter the username and password.

Step 5: Check whether the username and password is correct or not.

If yes, go to step 6.

If no, go to step 4.

Step 6: If the user is general, the user is directed to the “General form”.

Go to step 8.

Step 7: If the user is admin, the user is directed to the “Admin form”.

Go to step 8.

Step 8: End

3.4.3 Feedback Form

Step 1: Start

Step 2: Load Feedback form.

Step 3: Enter the customer details and check the ratings.

Step 4: Click the “Submit” button.

Step 5: Check whether the email is valid or not.

 If yes, go to step 6.

 If no, go to step 3.

Step 6: Check whether the checkboxes are checked or not.

 If yes, go to step 7.

 If no, go to step 3.

Step 7: Write the input data in the csv file.

Step 8: End

3.4.4 General Form

Step 1: Start

Step 2: Load General form.

Step 3: Click the “Report” button or “Import” button.

Step 4: If the “Report” button is clicked, the user is redirected to the “Report form”.

Go to step 6.

Step 5: If the “Import” button is clicked, the user is redirected to the “Import form”.

Go to step 6.

Step 6: End

3.4.5 Import Form

Step 1: Start

Step 2: Load Import form.

Step 3: Click “Import” button and select .csv file.

Step 4: Check whether the file is valid or not.

If yes, go to step 5.

If no, go to step 3.

Step 5: Imports the data from the selected csv file to the customer feedback csv file.

Step 6: End

3.4.6 Report Form

Step 1: Start

Step 2: Load Report form.

Step 3: Click the file menu and then open menu. Select the file that you want to open.

Step 4: Check whether the file is valid or not.

 If yes, go to step 5.

 If no, go to step 3.

Step 5: Display the csv file data in the table.

Step 6: View pie chart?

 If yes, go to step 7.

 If no, go to step 8.

Step 7: Click the “Chart” button to display the pie chart.

Step 8: End

3.4.7 Admin Form

Step 1: Start

Step 2: Load Admin form.

Step 3: Select the service name

Step 4: Enter the criteria name.

Step 5: Add criteria?

 If yes, go to step 6.

 If no, go to step 9.

Step 6: Click the “Add” button.

Step 7: Check whether the criteria name already exists or not.

 If yes, go to step 4.

 If no, go to step 8.

Step 8: Criteria is added.

Step 9: Delete criteria?

 If yes, go to step 10.

 If no, go to step 14.

Step 10: Click the “Delete” button.

Step 11: Check whether the criteria name exists or not.

 If yes, go to step 12.

 If no, go to step 4.

Step 12: Criteria is deleted.

Step 13: Enter criteria id.

Step 14: Update criteria?

 If yes, go to step 15.

 If no, go to step 19.

Step 15: Click the “Update” button.

Step 16: Check whether the criteria id exists or not.

 If yes, go to step 17.

 If no, go to step 13.

Step 17: Check whether the criteria name already exists or not.

 If yes, go to step 4.

 If no, go to step 18.

Step 18: Criteria is updated.

Step 19: End

3.5 Flowchart

3.5.1 Login form

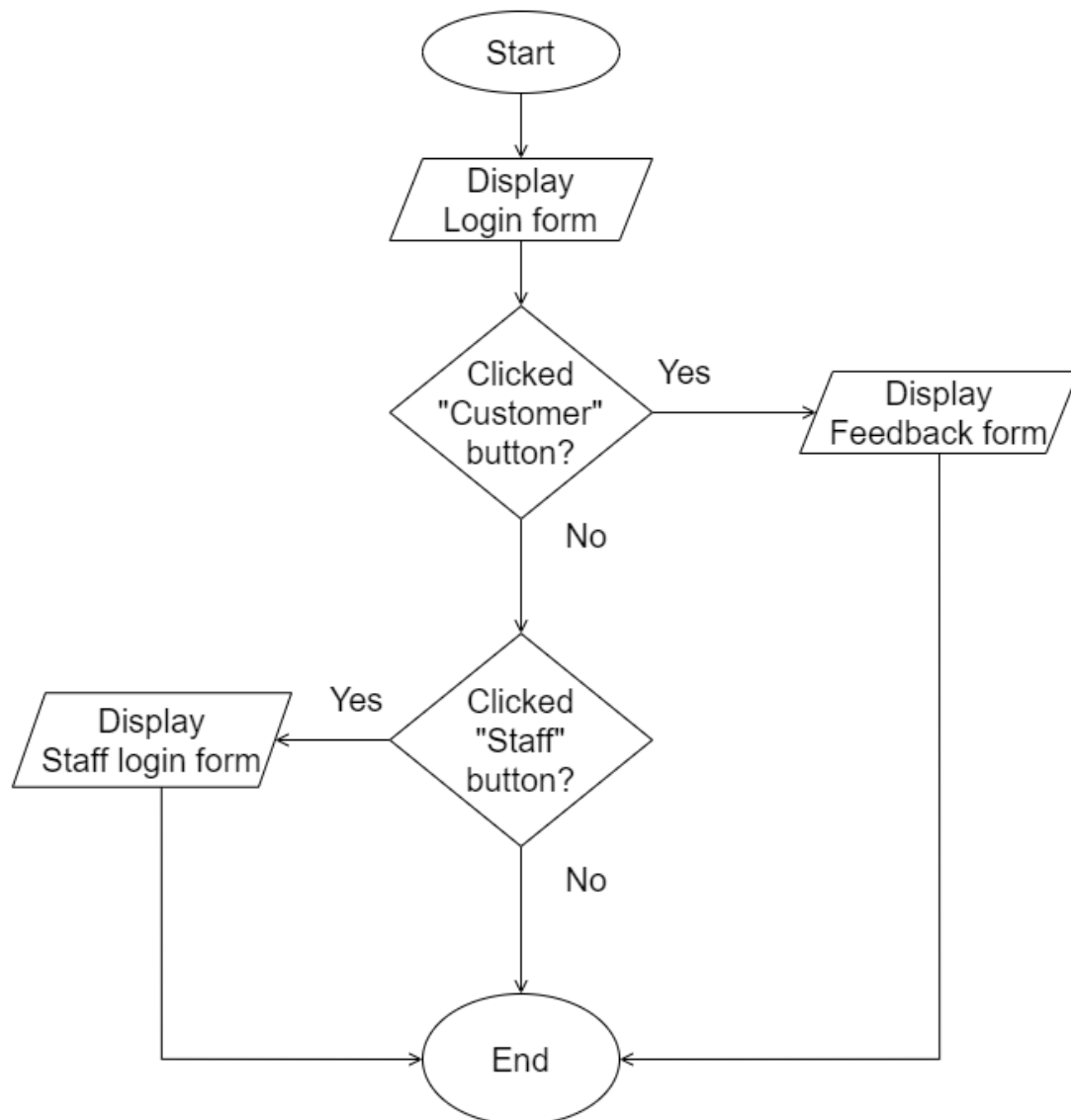
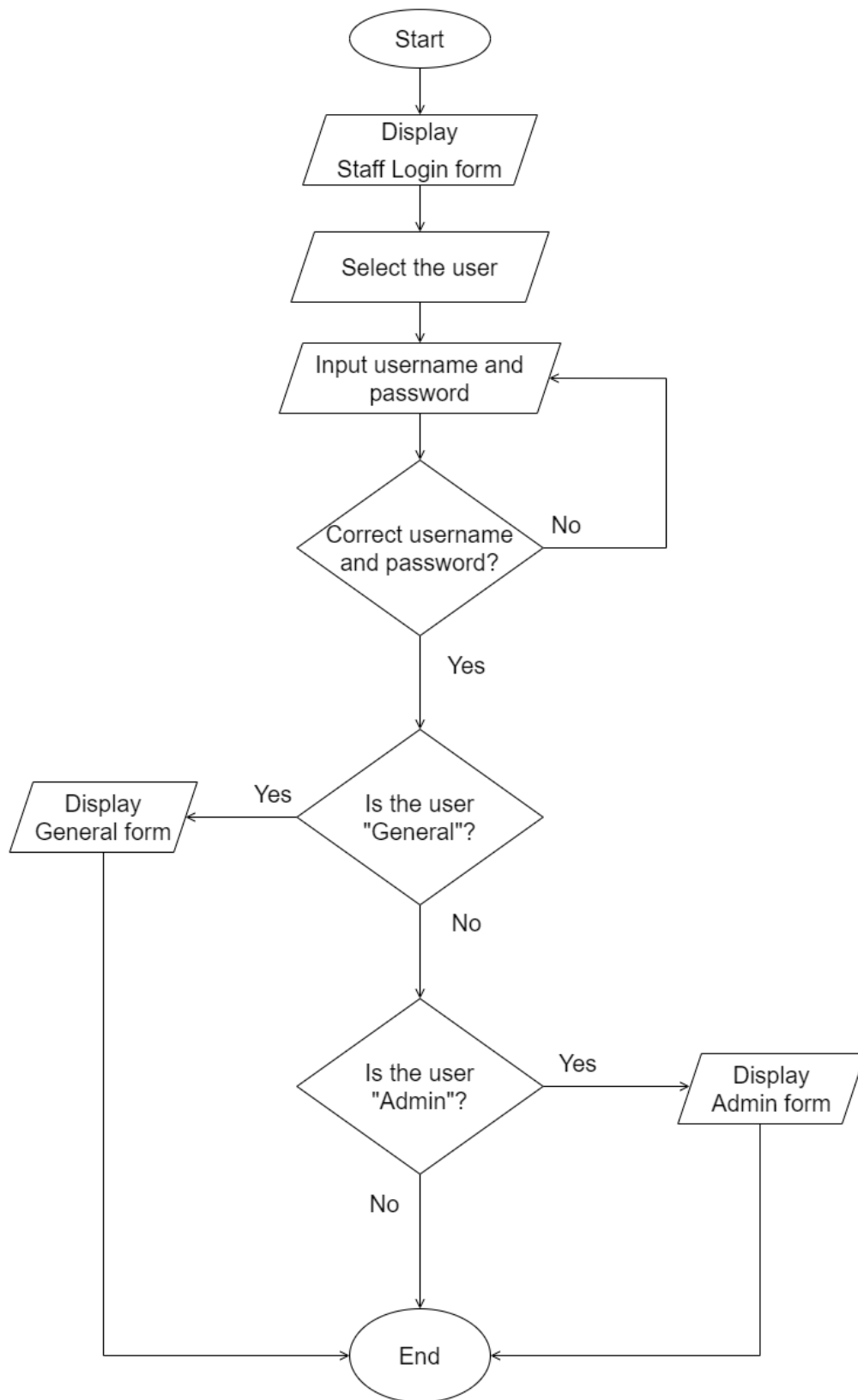
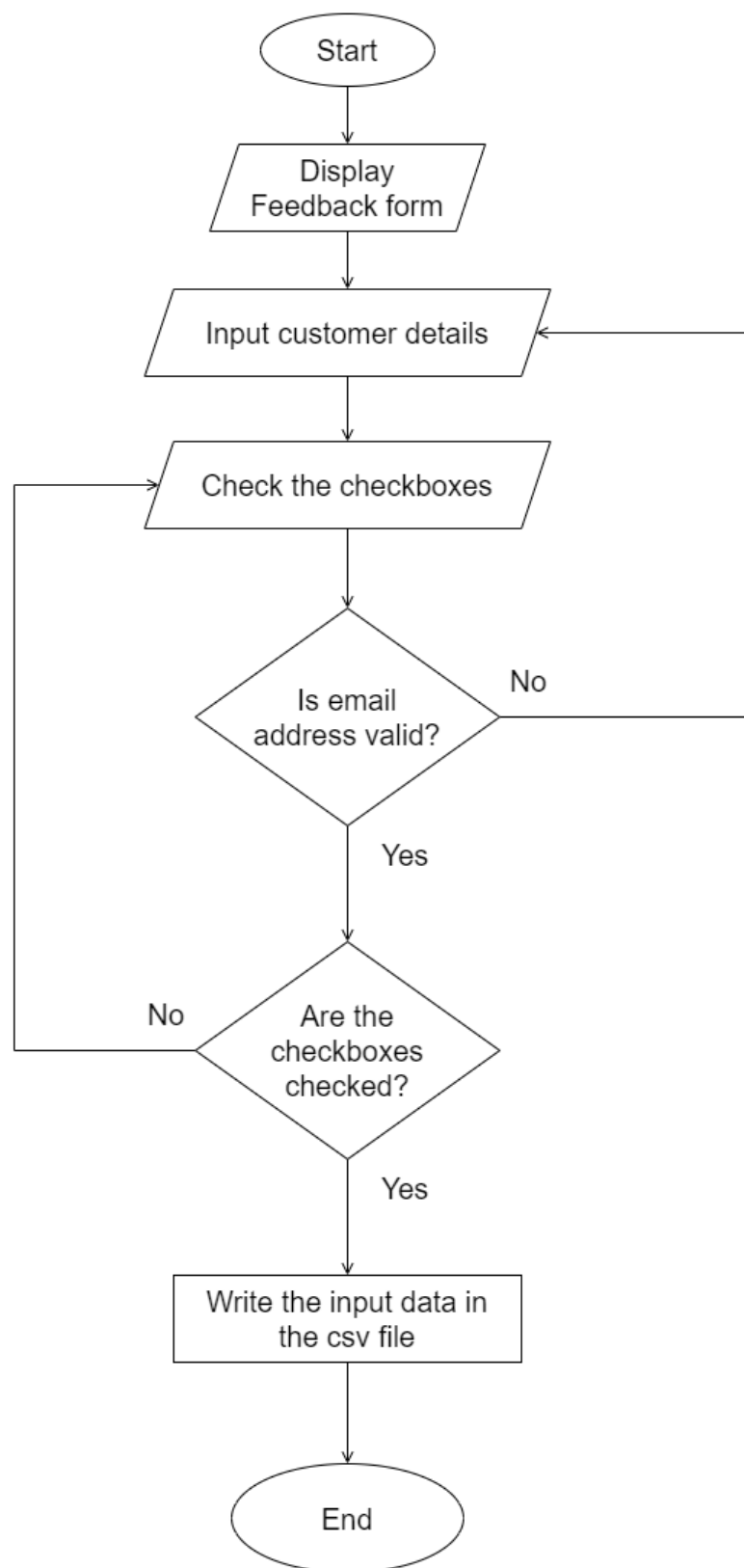


Figure 16: Flowchart of Login form

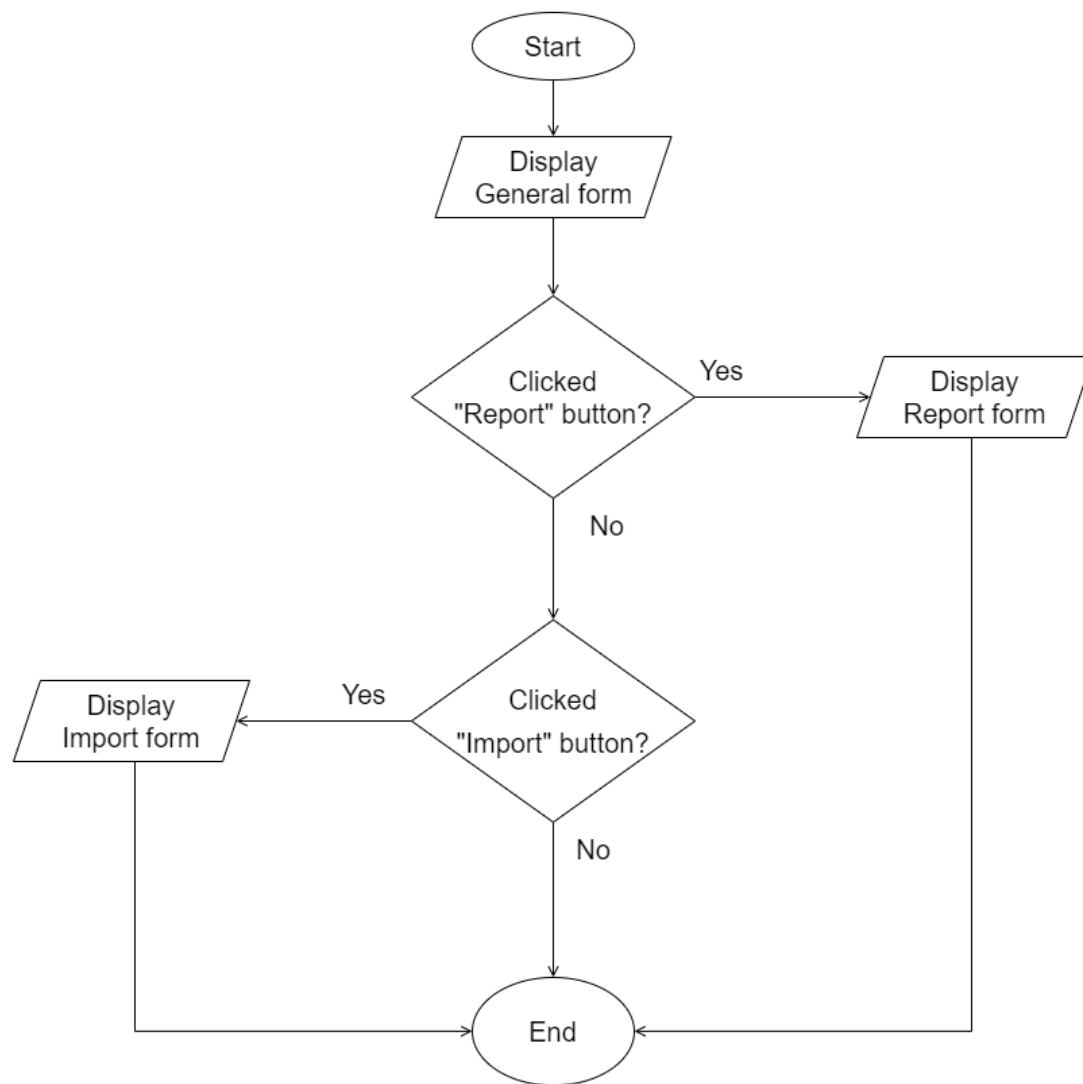
3.5.2 Staff Login form

*Figure 17: Flowchart of Staff Login form*

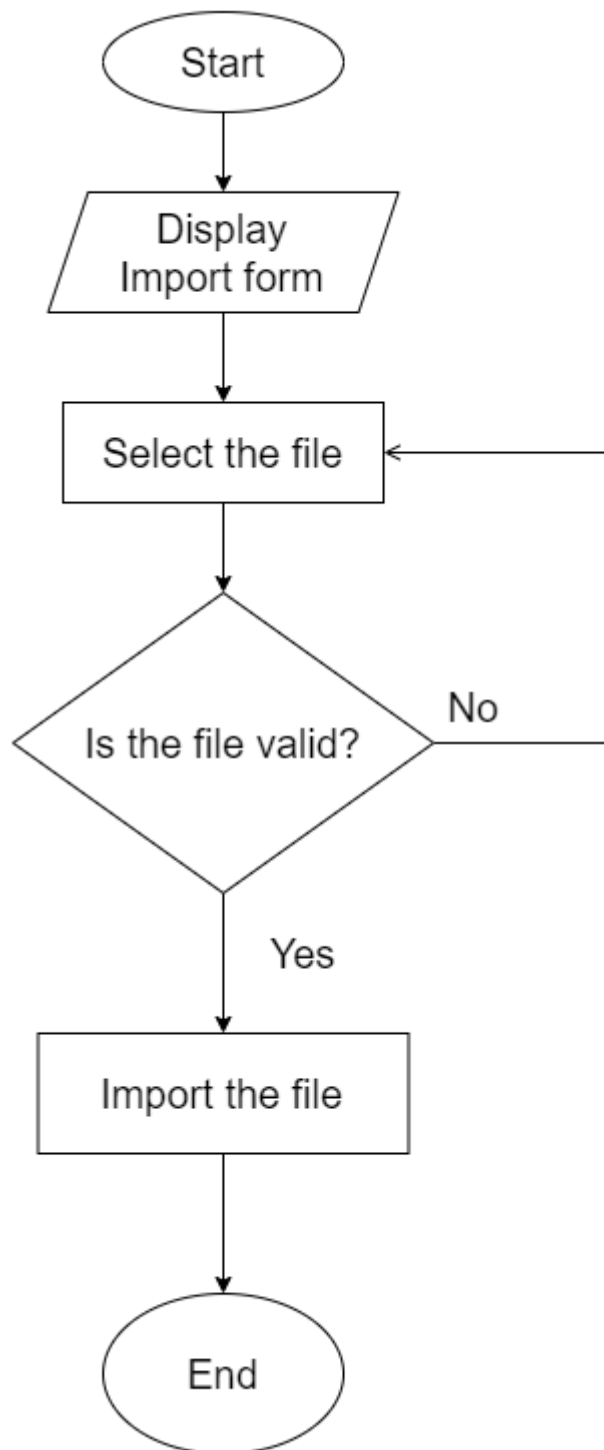
3.5.3 Feedback form

*Figure 18: Flowchart of Feedback form*

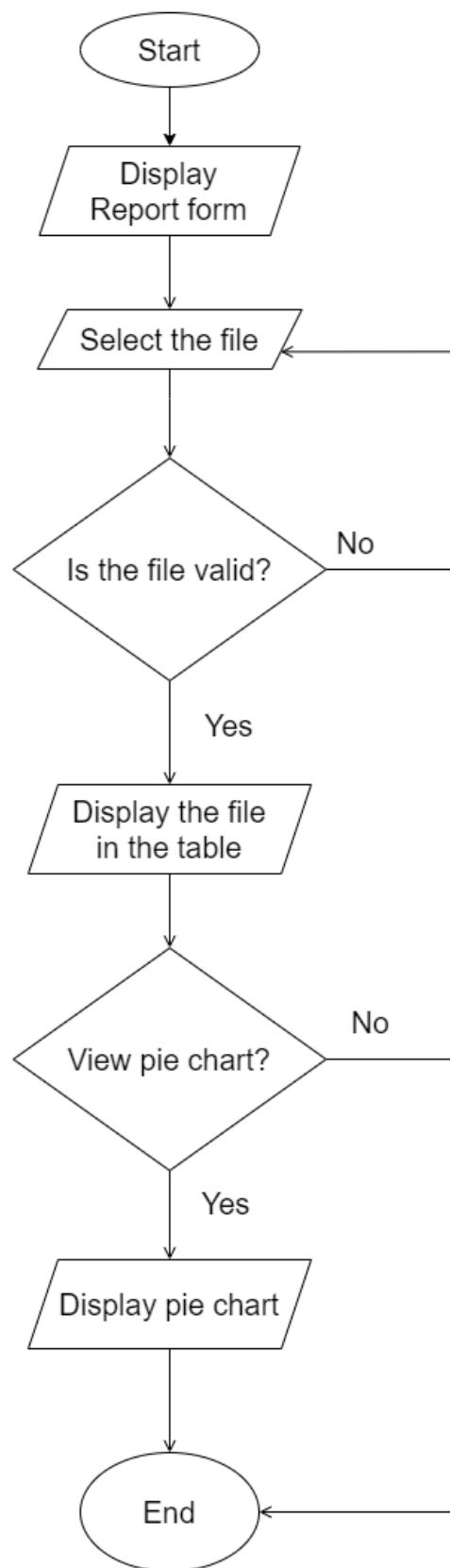
3.5.4 General form

*Figure 19: Flowchart of General form*

3.5.5 Import form

*Figure 20: Flowchart of Import form*

3.5.6 Report form

*Figure 21: Flowchart of Report form*

3.5.7 Admin form

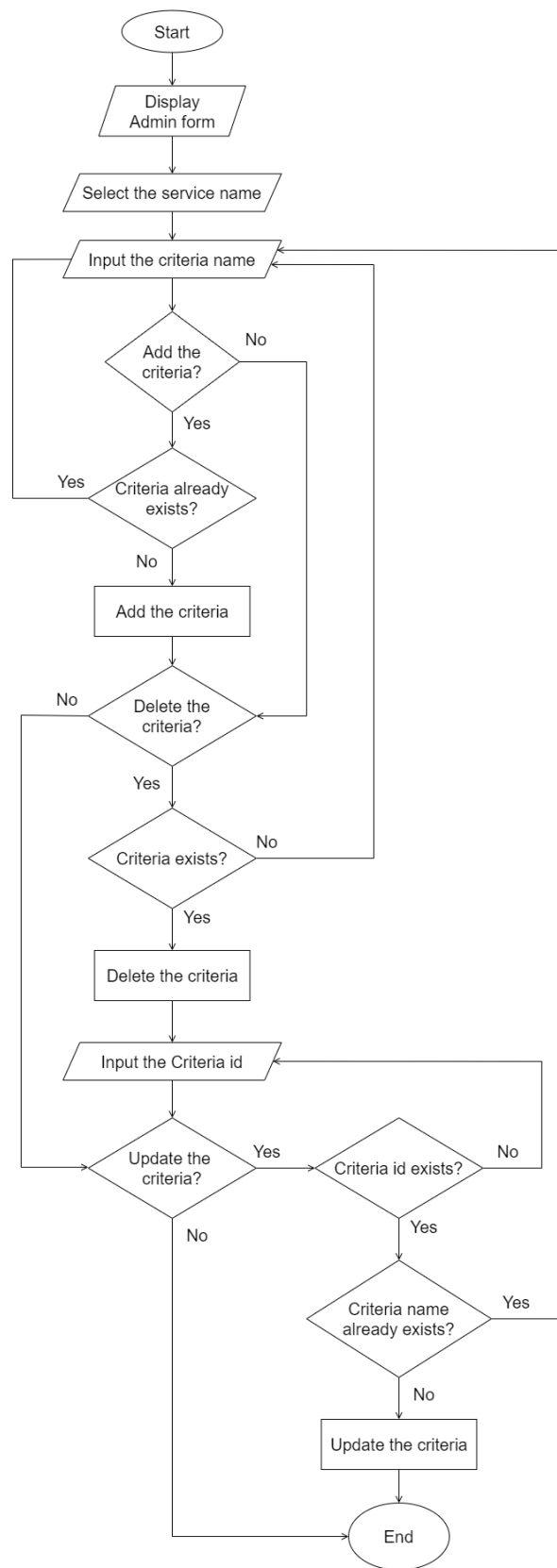
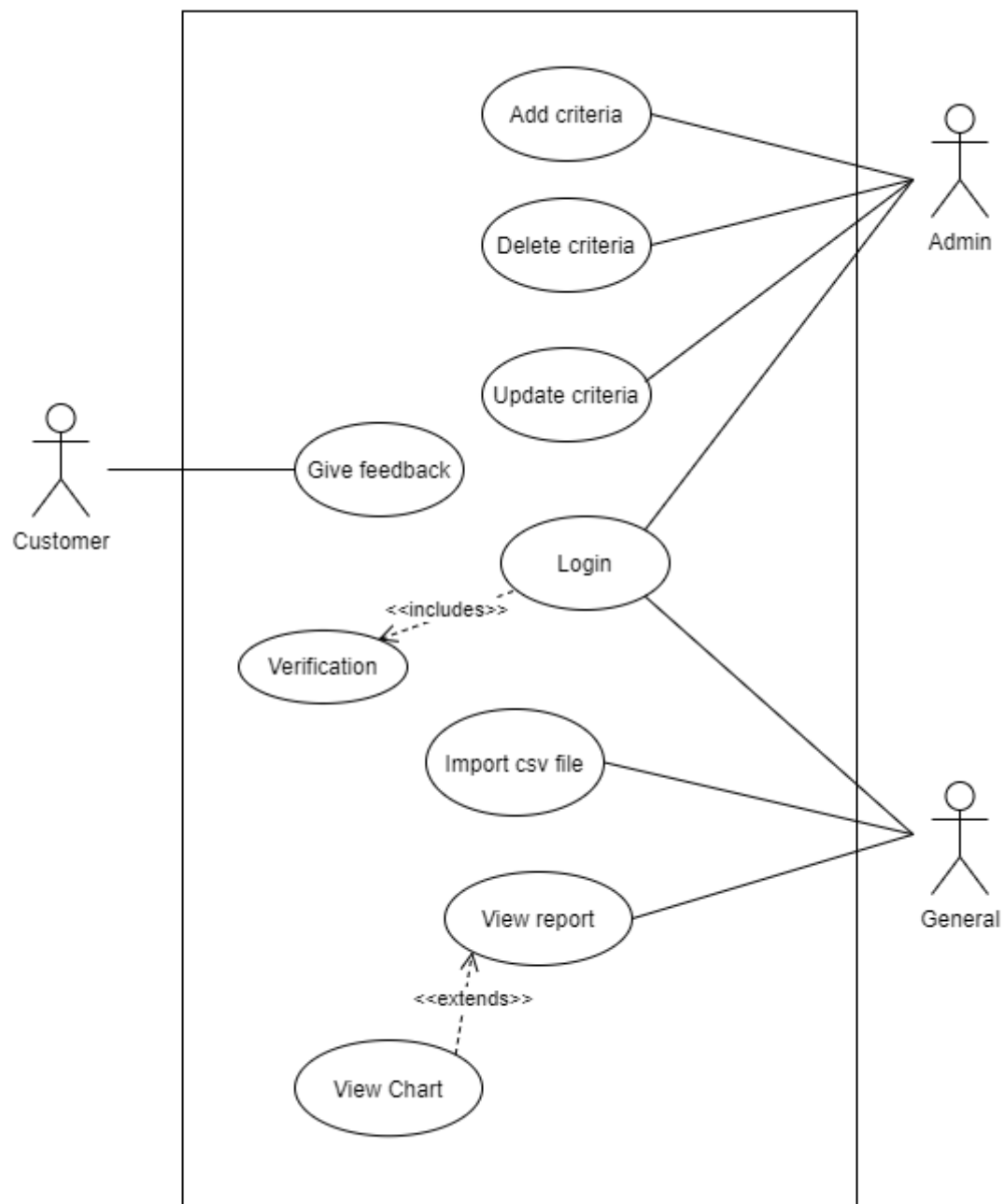


Figure 22: Flowchart of Admin form

3.6 Use case diagram

*Figure 23: Use case diagram*

3.7 Data structures and algorithms used

i. Data structure

Data structure is a particular way of organizing, processing, retrieving and storing data in a computer so that it can be used effectively. The data structures are of two type i.e. Linear Data Structure and Non-Linear Data Structure. Linear Data Structure are Arrays, List, Stack and Queue while Non-Linear Data Structure are Trees and Graphs (Rouse, 2019). In this project, I used Linear Data Structure i.e. Array and List.

1. Array

An array is a homogenous collection of items stored at adjoining memory locations (Rouse, 2019). These memory locations are called elements of that array. Using the array data structure, we can store one or multiple items having the same data-type.

In this project, I have used array to store the data while reading and writing the csv file.

2. List

List is a generic collection which is used to store the elements in the form of a list. It provides similar functionality like ArrayList (Geeks for Geeks, 2019).

In this project, I have used List to store data. I have also used it to store the rating of the criteria.

ii. Algorithm

In this project, I have used Bubble sort algorithm. Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm. It works by repeatedly stepping through the list. It compares adjacent pairs and swaps them if they are in the wrong order. This passing procedure is repeated until no swaps are required, indicating that the list is sorted (Technopedia, n.d.).

In this project, after the percentage of individual criteria are calculated, it is stored in a list which is then sorted in decreasing order. Then the sorted data is displayed in the chart.

4. Reflection

The development part was done in Visual Studio using C# programming language. After using Visual Studio, I could tell that Visual Studio is one of the best IDE that is developed till now. It was easy to use, write code and navigate between classes. One of the reasons why I really liked Visual studio is that it supports IntelliSense feature. While writing object name for classes, it gives meaningful suggestions i.e. quite similar to the class name.

Since I knew how to use Visual Studio and had some basic knowledge of C# language, it wasn't hard to start the project. But some of the tasks assigned in the coursework was quite hard for me. One of our task was to save and retrieve the objects using serialization. Since the serialization was a new topic for me, I had difficulty at first. But after doing lots of research related to this topic, I cleared my doubts. Using csv file to read and write data was also new concept for me.

Although the task were tough and required nights of hard work, the task was completed and submitted on given time.

References

Bassig, M., 2020. *reviewtrackers*. [Online]

Available at: <https://www.reviewtrackers.com/blog/customer-feedback-system/#:~:text=A%20customer%20feedback%20system%20%E2%80%94%20which,to%20improving%20overall%20customer%20experience>.

[Accessed 2020 December 26].

Geeks for Geeks, 2019. *Geeks for Geeks*. [Online]

Available at: <https://www.geeksforgeeks.org/list-implementation-in-c-sharp/>

[Accessed 24 January 2021].

Rouse, M., 2019. *SearchSqlServer*. [Online]

Available at: <searchsqlserver.techtarget.com/definition/data-structure>

[Accessed 15 January 2020].

Singh, 2020. *Techgeekbuzz*. [Online]

Available at: <https://www.techgeekbuzz.com/types-of-data-structure/>

[Accessed 15 January 2021].

Technopedia, n.d. *Technopedia*. [Online]

Available at: <https://www.techopedia.com/definition/3757/bubble-sort>

[Accessed 19 January 2021].

Appendix

1. Login.cs

```
using System;
using System.Windows.Forms;
namespace Cw1
{
    public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();
        }
        private void StaffB_Click(object sender, EventArgs e)
        {
            StaffLogin loginAG = new StaffLogin();
            loginAG.Show();
            Hide();
        }
        private void CustomerB_Click(object sender, EventArgs e)
        {
            Feedback customerFeedback = new Feedback();
            customerFeedback.Show();
            Hide();
        }
    }
}
```


2. StaffLogin.cs

```
using System;
using System.IO;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using System.Windows.Forms;
namespace Cw1
{
    public partial class StaffLogin : Form
    {
        MessageBoxButtons okBtn = MessageBoxButtons.OK;
        MessageBoxIcon exclamationIcon = MessageBoxIcon.Exclamation;
        public StaffLogin()
        {
            InitializeComponent();
            userCB.SelectedIndex = 0;
            string username;
            string password;
            int comboBoxIndex;
            try
            {
                IFormatter formatter = new BinaryFormatter();
                using (var stream = File.Open("loginData.csv", FileMode.Open,
                    FileAccess.Read))
                {
                    LoginSerializable loginSerializable =
                    (LoginSerializable)formatter.Deserialize(stream);
                    if (loginSerializable.User.Equals("General"))
                    {
                        comboBoxIndex = 0;
                    }
                    else
                    {
                        comboBoxIndex = 1;
                    }
                }
            }
            catch { }
        }
    }
}
```

```
        }
        username = loginSerializable.Username;
        password = loginSerializable.Password;
    }
    if (username != "" || password != "")
    {
        if (comboBoxIndex == 0)
        {
            userCB.SelectedIndex = 0;
        }
        else
        {
            userCB.SelectedIndex = 1;
        }
        usernameTB.Text = username;
        passwordTB.Text = password;
        rememberCB.Checked = true;
    }
}
catch
{
}
}

private void LoginB_Click(object sender, EventArgs e)
{
    string aUsername = "admin";
    string aPassword = "123";
    string gUsername = "general";
    string gPassword = "123";
    string username = usernameTB.Text;
    string password = passwordTB.Text;
    if (username.Equals("") && password.Equals(""))
```

```
{
    MessageBox.Show("Please fill the fields.", "Invalid", okBtn,
exclamationIcon);
}
else
{
    if (userCB.SelectedItem.Equals("Admin"))
    {
        if (username.Equals(aUsername) &&
password.Equals(aPassword))
        {
            Admin adminView = new Admin();
            adminView.Show();
            Hide();
        }
        else if (username == aUsername && password != aPassword)
        {
            MessageBox.Show("Password is incorrect.", "Invalid",
okBtn, exclamationIcon);
        }
        else
        {
            MessageBox.Show("Invalid field.", "Invalid", okBtn,
exclamationIcon);
        }
    }
    else if (userCB.SelectedItem.Equals("General"))
    {
        if (username.Equals(gUsername) &&
password.Equals(gPassword))
        {
            General general = new General();
            general.Show();
            Hide();
        }
    }
}
```

```
        else if (username == gUsername && password != gPassword)
        {
            MessageBox.Show("Password is incorrect.", "Invalid",
okBtn, exclamationIcon);
        }else
        {
            MessageBox.Show("Invalid field.", "Invalid", okBtn,
exclamationIcon);
        }
    }
}

private void BackB_Click(object sender, EventArgs e)
{
    Login login = new Login();
    login.Show();
    Hide();
}

private void RememberCB_CheckedChanged(object sender,
EventArgs e)
{
    RememberMe();
}

private void UsernameTB_TextChanged(object sender, EventArgs e)
{
    if (usernameTB.Text != "" && passwordTB.Text != "")
    {
        RememberMe();
    }
}

private void PasswordTB_TextChanged(object sender, EventArgs e)
{
    passwordTB.PasswordChar = '*';
}
```

```
        if (usernameTB.Text != "" && passwordTB.Text != "")
        {
            RememberMe();
        }
    }
    private void RememberMe()
    {
        LoginSerializable loginSerializable = new LoginSerializable();
        if (rememberCB.Checked == true)
        {
            loginSerializable.User = userCB.Text;
            loginSerializable.Username = usernameTB.Text;
            loginSerializable.Password = passwordTB.Text;
        }
        else
        {
            loginSerializable.User = "";
            loginSerializable.Username = "";
            loginSerializable.Password = "";
        }
        IFormatter formatter = new BinaryFormatter();
        Stream stream = new FileStream("loginData.csv",
        FileMode.Create, FileAccess.Write);
        formatter.Serialize(stream, loginSerializable);
        stream.Close();
    }
}
```

3. LoginSerializable.cs

```
using System;
namespace Cw1
{
    [Serializable]
    class LoginSerializable
    {
        public String User;
        public String Username;
        public String Password;
    }
}
```

4. Feedback.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.IO;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Net.Mail;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
namespace Cw1
{
    public partial class Feedback : Form
    {
        string cDate, cTime;
        string title = "Invalid";
        MessageBoxButtons okBtn = MessageBoxButtons.OK;
        MessageBoxIcon exclamationIcon = MessageBoxIcon.Exclamation;
        MessageBoxIcon infoIcon = MessageBoxIcon.Information;
        CheckBox[] boxes;
```

```
private void MainPage_Load(object sender, EventArgs e)
{
}
public Feedback()
{
    InitializeComponent();
    currentD.Text = DateTime.Now.ToString("dd/MM/yyyy");
    currentT.Text = DateTime.Now.ToString("hh/mm/ss tt");
    cDate = currentD.Text;
    cTime = currentT.Text;
    sNameCB.SelectedIndex = 0;
}
private void SNameCB_SelectedIndexChanged(object sender,
EventArgs e)
{
    criteriaS1.Controls.Clear();
    string criteriaPath;
    if (sNameCB.SelectedItem.Equals("Service 1"))
    {
        criteriaPath = "criteriaservice1.csv";
        AddCriteria(criteriaPath);
    }
    else if (sNameCB.SelectedItem.Equals("Service 2"))
    {
        criteriaPath = "criteriaservice2.csv";
        AddCriteria(criteriaPath);
    }
}
public void AddCriteria(string path)
{
    StreamReader streamReader = new StreamReader(path);
    string read = streamReader.ReadLine();
```

```
if (read != null)
{
    String[] criteria = read.Split(',');
    int criteriaCount = criteria.Length;
    int totalCheckBox = criteriaCount * 4;
    boxes = new CheckBox[totalCheckBox];
    int row = 0;
    for (int i = 0; i < criteriaCount; i++)
    {
        criteriaS1.RowStyles.Add(new RowStyle(SizeType.Percent,
Height = 100 ));
        Label label = new Label()
        {
            AutoSize = false,
            Anchor = AnchorStyles.Top | AnchorStyles.Bottom |
AnchorStyles.Left | AnchorStyles.Right,
            Font = new Font("Microsoft Sans Serif", 12),
            TextAlign = ContentAlignment.MiddleCenter,
            Text = criteria[i]
        };
        criteriaS1.Controls.Add(label, 0, row);
        row++;
    }
    for (int a = 0; a < totalCheckBox; a++)
    {
        CheckBox checkBox = new CheckBox();
        checkBox.Name = "checkbox" + a.ToString();
        checkBox.Anchor = AnchorStyles.Top | AnchorStyles.Bottom |
AnchorStyles.Left | AnchorStyles.Right;
        checkBox.CheckAlign = ContentAlignment.MiddleCenter;
        checkBox.CheckedChanged += new
EventHandler(CheckBox_CheckedChanged);
        boxes[a] = checkBox;
```



```
        criteriaS1.Controls.Add(boxes[a]);
    }
    submitB.Enabled = true;
    clearB.Enabled = true;
}
else
{
    MessageBox.Show("No criteria found.", title, okBtn,
exclamationIcon);
}
}
public void CheckBox_CheckedChanged(Object sender,EventArgs e)
{
    int boxesCount = boxes.Length;
    CheckBox selectedCheckBox = (CheckBox)sender;
    for (int i = 0; i < boxesCount; i += 4)
    {
        if (selectedCheckBox.Equals(boxes[i]))
        {
            boxes[i + 1].Checked = false;
            boxes[i + 2].Checked = false;
            boxes[i + 3].Checked = false;
        }
        else if (selectedCheckBox.Equals(boxes[i + 1]))
        {
            boxes[i].Checked = false;
            boxes[i + 2].Checked = false;
            boxes[i + 3].Checked = false;
        }
        else if (selectedCheckBox.Equals(boxes[i + 2]))
        {
            boxes[i].Checked = false;
            boxes[i + 1].Checked = false;
            boxes[i + 3].Checked = false;
        }
    }
}
```

```
    }
    else if (selectedCheckBox.Equals(boxes[i + 3]))
    {
        boxes[i].Checked = false;
        boxes[i + 1].Checked = false;
        boxes[i + 2].Checked = false;
    }
}

bool check = CheckBoxCheck();
if (nameTB.Text != "" && contactTB.Text != "" && emailTB.Text !=
"" && overallTB.Text != "" && check.Equals(true))
{
    SerializeCheck();
}
}

private bool CheckBoxCheck()
{
    bool check = false;
    int boxesCount = boxes.Length;
    for (int i = 0; i < boxesCount; i += 4)
    {
        if (boxes[i].Checked == false && boxes[i + 1].Checked == false
&& boxes[i + 2].Checked == false && boxes[i + 3].Checked == false)
        {
            check = false;
            break;
        }
        else
        {
            check = true;
        }
    }
    return check;
}
```

```
private void ContactTB_TextChanged(object sender, EventArgs e)
{
    if (Regex.IsMatch(contactTB.Text, "[^0-9]"))
    {
        MessageBox.Show("Please enter only numbers.", title, okBtn,
exclamationIcon);
        contactTB.Clear();
    }
    bool check = CheckBoxCheck();
    if (nameTB.Text != "" && contactTB.Text != "" && emailTB.Text !=
"" && overallTB.Text != "" && check.Equals(true))
    {
        SerializeCheck();
    }
}

private void SubmitB_Click(object sender, EventArgs e)
{
    string custName = nameTB.Text;
    string contact = contactTB.Text;
    string email = emailTB.Text;
    string overall = overallTB.Text;
    bool emailCheck = IsEmailValid(email);
    if (custName.Equals(""))
    {
        custName = "Anonymous";
    }
    if (contact.Equals(""))
    {
        contact = "Anonymous";
    }
    if (email.Equals(""))
    {
        email = "Anonymous";
    }
}
```

```
if (overall.Equals(""))
{
    overall = "Nothing";
}
if (emailCheck.Equals(true))
{
    bool check = CheckBoxCheck();
    if (check.Equals(true))
    {
        List<string> rating = new List<string>();
        int boxCount = boxes.Length;
        for (int i = 0; i < boxCount; i += 4)
        {
            if (boxes[i].Checked == true)
            {
                rating.Add("5");
            }
            else if (boxes[i + 1].Checked == true)
            {
                rating.Add("3");
            }
            else if (boxes[i + 2].Checked == true)
            {
                rating.Add("2");
            }
            else if (boxes[i + 3].Checked == true)
            {
                rating.Add("1");
            }
        }
        int ratingCount = rating.Count;
        string[] finalRating = new string[ratingCount];
        for (int i = 0; i < ratingCount; i++)
```

```
{
    finalRating[i] = rating[i];
}
string result = string.Join(",", finalRating);
if (sNameCB.Text.Equals("Service 1"))
{
    string serviceName = sNameCB.Text;
    string path = "customerdetails1.csv";
    try
    {
        StreamWriter streamWriter = new StreamWriter(path,
true);

        using (streamWriter)
        {
            streamWriter.WriteLine(serviceName + "," + custName
+ "," + contact + "," + email + "," + overall + "," + cDate + "," + cTime + "," +
result);

            MessageBox.Show("Thank you for the feedback.",
"Message", okBtn, infoIcon);
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Error while writing the data.", title,
okBtn, exclamationIcon);
    }
}
else
{
    String serviceName = sNameCB.Text;
    String path = "customerdetails2.csv";
    try
    {
```

```
        StreamWriter streamWriter = new StreamWriter(path,
true);

        using (streamWriter)
        {
            streamWriter.WriteLine(serviceName + "," + custName
+ "," + contact + "," + email + "," + overall + "," + cDate + "," + cTime + "," +
result);

            MessageBox.Show("Thank you for the feedback.",
"Message", okBtn, infoIcon);
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Error while writing the data.", title,
okBtn, exclamationIcon);
    }
}
else
{
    MessageBox.Show("Please check the check boxes.", title,
okBtn, exclamationIcon);
}
else
{
    MessageBox.Show("Please enter valid email address.", title,
okBtn, exclamationIcon);
}
}
private bool IsEmailValid(string email)
{
    try
    {
```

```
        if (email.Equals(""))
        {
            return true;
        }
        else
        {
            var eAddress = new MailAddress(email);
            return true;
        }
    }
    catch
    {
        return false;
    }
}

private void ExitToolStripMenuItem_Click(object sender, EventArgs e)
{
    string exitTitle = "Close Window";
    string message = "Do you want to close this window?";
    MessageBoxButtons buttons = MessageBoxButtons.YesNo;
    DialogResult result = MessageBox.Show(message, exitTitle,
buttons);
    if (result == DialogResult.Yes)
    {
        Login login = new Login();
        login.Show();
        Hide();
    }
}

private void ClearB_Click(object sender, EventArgs e)
{
    try
    {
        nameTB.Clear();
    }
}
```

```
        contactTB.Clear();
        emailTB.Clear();
        overallTB.Clear();
        currentD.Text = DateTime.Now.ToString("dd/MM/yyyy");
        currentT.Text = DateTime.Now.ToString("hh/mm/ss tt");
        cDate = currentD.Text;
        cTime = currentT.Text;
        int bCount = boxes.Length;
        for (int i = 0; i < bCount; i++)
        {
            boxes[i].Checked = false;
        }
    }
    catch
    {
        MessageBox.Show("Fields are already empty.", "Message",
okBtn, infoIcon);
    }
}

private void NameTB_TextChanged(object sender, EventArgs e)
{
    bool check = CheckBoxCheck();
    if (nameTB.Text != "" && contactTB.Text != "" && emailTB.Text !=
"" && overallTB.Text != "" && check.Equals(true))
    {
        SerializeCheck();
    }
}

private void EmailTB_TextChanged(object sender, EventArgs e)
{
    bool check = CheckBoxCheck();
    if (nameTB.Text != "" && contactTB.Text != "" && emailTB.Text !=
"" && overallTB.Text != "" && check.Equals(true))
    {
```



```
        SerializeCheck();
    }
}

private void OverallTB_TextChanged(object sender, EventArgs e)
{
    bool check = CheckBoxCheck();
    if (nameTB.Text != "" && contactTB.Text != "" && emailTB.Text !=
"" && overallTB.Text != "" && check.Equals(true))
    {
        SerializeCheck();
    }
}

private void RetrieveBtn_Click(object sender, EventArgs e)
{
    try
    {
        IFormatter formatter = new BinaryFormatter();
        using (var stream = File.Open("serializedCustData.csv",
FileMode.Open, FileAccess.Read))
        {
            CustomerSerializable customer =
(CustomerSerializable)formatter.Deserialize(stream);
            nameTB.Text = customer.CustName;
            contactTB.Text = customer.CustContact;
            emailTB.Text = customer.CustEmail;
            overallTB.Text = customer.CustOverall;
            string[] rating = customer.CustResult.Split(',');
            int ratingCount = rating.Length;
            int a = 0;
            for (int i = 0; i < ratingCount; i++)
            {
                if (rating[i].Equals("5"))
                {
                    boxes[a].Checked = true;
                }
            }
        }
    }
}
```

```
        }
        else if (rating[i].Equals("3"))
        {
            boxes[a + 1].Checked = true;
        }
        else if (rating[i].Equals("2"))
        {
            boxes[a + 2].Checked = true;
        }
        else if (rating[i].Equals("1"))
        {
            boxes[a + 3].Checked = true;
        }
        a += 4;
    }
}
}
catch
{
    MessageBox.Show("Data was not saved", "Invalid", okBtn,
exclamationIcon);
}
}
private void SerializeCheck()
{
    try
    {
        CustomerSerializable customer = new CustomerSerializable();
        List<string> rating = new List<string>();
        int boxesCount = boxes.Length;
        for (int i = 0; i < boxesCount; i += 4)
        {
            if (boxes[i].Checked == true)
            {
```

```
        rating.Add("5");
    }
    else if (boxes[i + 1].Checked == true)
    {
        rating.Add("3");
    }
    else if (boxes[i + 2].Checked == true)
    {
        rating.Add("2");
    }
    else if (boxes[i + 3].Checked == true)
    {
        rating.Add("1");
    }
    else
    {
        rating.Add("0");
    }
}
int ratingCount = rating.Count;
string[] finalRating = new string[ratingCount];
for (int i = 0; i < ratingCount; i++)
{
    finalRating[i] = rating[i];
}
string result = string.Join(",", finalRating);
customer.CustName = nameTB.Text;
customer.CustContact = contactTB.Text;
customer.CustEmail = emailTB.Text;
customer.CustOverall = overallTB.Text;
customer.CustResult = result;
IFormatter formatter = new BinaryFormatter();
Stream stream = new FileStream("serializedCustData.csv",
    FileMode.Create, FileAccess.Write);
```

```
        formatter.Serialize(stream, customer);
        stream.Close();
    }
    catch
    {
    }
}
}
```

5. CustomerSerializable.cs

```
using System;
namespace Cw1
{
    [Serializable]
    class CustomerSerializable
    {
        public String CustName;
        public String CustContact;
        public String CustEmail;
        public String CustOverall;
        public String CustResult;
    }
}
```

6. Admin.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;
namespace Cw1
{
    public partial class Admin : Form
    {
        MessageBoxButtons okBtn = MessageBoxButtons.OK;
```

```
MessageBoxIcon exclamationIcon = MessageBoxIcon.Exclamation;
MessageBoxIcon infoIcon = MessageBoxIcon.Information;
public Admin()
{
    InitializeComponent();
}
private void AddB_Click(object sender, EventArgs e)
{
    string serviceName = serviceNameCB.Text;
    string criteria = criteriaNameTB.Text;
    bool check = AddCheck(serviceName, criteria);
    if (check.Equals(true))
    {
        if (serviceName.Equals("Service 1"))
        {
            string pathname = "criteriaservice1.csv";
            string[] criteriaArray = null;
            try
            {
                StreamReader streamReader = new
StreamReader(pathname);
                string read = null;
                while ((read = streamReader.ReadLine()) != null)
                {
                    criteriaArray = read.Split(',');
                }
                streamReader.Close();
            }
            catch
            {
                MessageBox.Show("Error while reading the data.", "Invalid",
okBtn, exclamationIcon);
            }
        }
    }
}
```

```
List<string> criteriaList = new List<string>();
for (int i = 0; i < criteriaArray.Length; i++)
{
    criteriaList.Add(criteriaArray[i].ToLower());
}
if (criteriaList.Contains(criteria.ToLower()))
{
    MessageBox.Show("Criteria already exists.", "Invalid",
okBtn, exclamationIcon);
}
else
{
    string result = string.Join(",", criteriaArray);
    try
    {
        StreamWriter streamWriter = new
StreamWriter(pathname);
        using (streamWriter)
        {
            streamWriter.WriteLine(result + "," + criteria);
            MessageBox.Show("Criteria added.", "Message",
okBtn, infoIcon);
        }
    }
    catch
    {
        MessageBox.Show("Error while writing the data.",
"Invalid", okBtn, exclamationIcon);
    }
}
else if (serviceName.Equals("Service 2"))
{
    string pathname = "criteriaservice2.csv";
```

```
string[] criteriaArray = null;
try
{
    StreamReader streamReader = new
StreamReader(pathname);
    string read = null;
    while ((read = streamReader.ReadLine()) != null)
    {
        criteriaArray = read.Split(',');
    }
    streamReader.Close();
}
catch
{
    MessageBox.Show("Error while reading the data.",
"Invalid", okBtn, exclamationIcon);
}
List<string> criteriaList = new List<string>();
for (int i = 0; i < criteriaArray.Length; i++)
{
    criteriaList.Add(criteriaArray[i].ToLower());
}
if (criteriaList.Contains(criteria.ToLower()))
{
    MessageBox.Show("Criteria already exists.", "Invalid",
okBtn, exclamationIcon);
}
else
{
    string result = string.Join(", ", criteriaArray);
    try
    {
        StreamWriter streamWriter = new
StreamWriter(pathname);
```

```
        using (StreamWriter)
        {
            streamWriter.WriteLine(result + "," + criteria);

            MessageBox.Show("Criteria added.", "Message",
okBtn, infoIcon);
        }
    }
    catch
    {
        MessageBox.Show("Error while writing the data.",
"Invalid", okBtn, exclamationIcon);
    }
}
}

public bool AddCheck(string serviceName, string criteria)
{
    if (serviceName.Equals("") && criteria.Equals(""))
    {
        MessageBox.Show("Please fill all the fields.", "Invalid");
    }
    else if (serviceName.Equals(""))
    {
        MessageBox.Show("Please select the service name.", "Invalid");
    }
    else if (criteria.Equals(""))
    {
        MessageBox.Show("Please enter the criteria.", "Invalid");
    }
    else
    {
        return true;
    }
}
```



```
    }
    return false;
}
private void UpdateB_Click(object sender, EventArgs e)
{
    string serviceId = idTB.Text;
    string criteriaName = criteriaUTB.Text;
    string serviceName = serviceUCB.Text;
    bool check = UpdateCheck(serviceId, criteriaName, serviceName);
    if (check.Equals(true))
    {
        int id = int.Parse(serviceId);
        if (serviceName.Equals("Service 1"))
        {
            string path = "criteriaservice1.csv";
            string[] criteriaArray = null;
            try
            {
                StreamReader streamReader = new StreamReader(path);
                string read = null;
                while ((read = streamReader.ReadLine()) != null)
                {
                    criteriaArray = read.Split(',');
                }
                streamReader.Close();
            }
            catch
            {
                MessageBox.Show("Error while reading the data.",
"Invalid", okBtn, exclamationIcon);
            }
            int count = criteriaArray.Length;
            if (count >= id)
```

```
{
    List<string> criteriaList = new List<string>();
    for (int i = 0; i < count; i++)
    {
        criteriaList.Add(criteriaArray[i].ToLower());
    }
    if (criteriaList.Contains(criteriaName.ToLower()))
    {
        MessageBox.Show("Criteria already exists.", "Invalid",
okBtn, exclamationIcon);
    }
    else
    {
        criteriaArray[id - 1] = criteriaName;
        string result = string.Join(",", criteriaArray);
        try
        {
            StreamWriter streamWriter = new StreamWriter(path);
            using (streamWriter)
            {
                streamWriter.WriteLine(result);
                MessageBox.Show("Criteria updated.", "Message");
            }
        }
        catch
        {
            MessageBox.Show("Error while writing the data.",
"Invalid");
        }
    }
}
else
{
```

```
        MessageBox.Show("Please enter valid Id number.",
    "Invalid");
    }
}
else if (serviceName.Equals("Service 2"))
{
    string path = "criteriaservice2.csv";
    string[] criteriaArray = null;
    try
    {
        StreamReader streamReader = new StreamReader(path);
        string read = null;
        while ((read = streamReader.ReadLine()) != null)
        {
            criteriaArray = read.Split(',');
        }
        streamReader.Close();
    }
    catch
    {
        MessageBox.Show("Error while reading the data.",
    "Invalid", okBtn, exclamationIcon);
    }
    int count = criteriaArray.Length;
    if (count >= id)
    {
        List<string> criteriaList = new List<string>();
        for (int i = 0; i < count; i++)
        {
            criteriaList.Add(criteriaArray[i].ToLower());
        }
        if (criteriaList.Contains(criteriaName.ToLower()))
        {
```

```
        MessageBox.Show("Criteria already exists.", "Invalid",
        okBtn, exclamationIcon);
    }
    else
    {
        criteriaArray[id - 1] = criteriaName;
        string result = string.Join(", ", criteriaArray);
        try
        {
            StreamWriter streamWriter = new StreamWriter(path);
            using (streamWriter)
            {
                streamWriter.WriteLine(result);
                MessageBox.Show("Criteria updated.", "Message");
            }
        }
        catch
        {
            MessageBox.Show("Error while writing the data.",
            "Invalid");
        }
    }
}
else
{
    MessageBox.Show("Please enter valid Id number.",
    "Invalid");
}
}
}
```

```
public bool UpdateCheck(String SId, String criteriaName, String
serviceName)
{
    if (SId.Equals("") && serviceName.Equals("") &&
criteriaName.Equals(""))
    {
        MessageBox.Show("Please fill all the fields.", "Invalid");
    }
    else if (SId.Equals(""))
    {
        MessageBox.Show("Please enter the id number.", "Invalid");
    }
    else if (serviceName.Equals(""))
    {
        MessageBox.Show("Please select the service name.", "Invalid");
    }
    else if (criteriaName.Equals(""))
    {
        MessageBox.Show("Please enter the criteria.", "Invalid");
    }
    else
    {
        return true;
    }
    return false;
}

private void DeleteB_Click(object sender, EventArgs e)
{
    string criteriaName = criteriaNameTB.Text;
    string serviceName = serviceNameCB.Text;
    bool check = DeleteCheck(criteriaName, serviceName);
    if (check.Equals(true))
    {
        if (serviceName.Equals("Service 1"))
```

```
{
    List<string> newCriteria = new List<string>();
    string path = "criteriaservice1.csv";
    try
    {
        StreamReader streamReader = new StreamReader(path);
        string read = null;
        while ((read = streamReader.ReadLine()) != null)
        {
            string[] criteriaArray = read.Split(',');
            int count = criteriaArray.Length;
            List<string> criteriaList = new List<string>();
            for (int i = 0; i < count; i++)
            {
                criteriaList.Add(criteriaArray[i].ToLower());
            }
            if (criteriaList.Contains(criteriaName.ToLower()))
            {
                for (int i = 0; i < count; i++)
                {
                    if (criteriaArray[i] != criteriaName)
                    {
                        //Adding the values from array to the list
                        newCriteria.Add(criteriaArray[i]);
                    }
                }
            }
            else
            {
                MessageBox.Show("Criteria not found. Please check
the spelling.", "Invalid", okBtn, exclamationIcon);
            }
        }
        streamReader.Close();
    }
}
```

```
    }
    catch
    {
        MessageBox.Show("Error while reading the data.",
"Invalid", okBtn, exclamationIcon);
    }
    int newCriteriaCount = newCriteria.Count;
    if (newCriteriaCount != 0)
    {
        string[] finalCriteria = new string[newCriteriaCount];
        for (int i = 0; i < newCriteriaCount; i++)
        {
            finalCriteria[i] = newCriteria[i];
        }
        string result = string.Join(",", finalCriteria);
        try
        {
            StreamWriter streamWriter = new StreamWriter(path);
            using (streamWriter)
            {
                streamWriter.WriteLine(result);
                MessageBox.Show("Criteria deleted.", "Message",
okBtn, infoIcon);
            }
        }
        catch
        {
            MessageBox.Show("Error while reading the data.",
"Invalid", okBtn, exclamationIcon);
        }
    }
    else if (serviceName.Equals("Service 2"))
    {
```

```
List<string> newCriteria = new List<string>();
string path = "criteriaservice2.csv";
try
{
    StreamReader streamReader = new StreamReader(path);
    string read = null;
    while ((read = streamReader.ReadLine()) != null)
    {
        string[] criteriaArray = read.Split(',');
        int count = criteriaArray.Length;
        List<string> criteriaList = new List<string>();
        for (int i = 0; i < count; i++)
        {
            criteriaList.Add(criteriaArray[i].ToLower());
        }
        if (criteriaList.Contains(criteriaName.ToLower()))
        {
            for (int i = 0; i < count; i++)
            {
                if (criteriaArray[i] != criteriaName)
                {
                    newCriteria.Add(criteriaArray[i]);
                }
            }
        }
        else
        {
            MessageBox.Show("Criteria not found. Please check
the spelling.", "Invalid", okBtn, exclamationIcon);
        }
    }
    streamReader.Close();
}
```



```
        catch
        {
            MessageBox.Show("Error while reading the data.",
"Invalid", okBtn, exclamationIcon);
        }
        int newCriteriaCount = newCriteria.Count;
        if (newCriteriaCount != 0)
        {
            string[] finalCriteria = new string[newCriteriaCount];
            for (int i = 0; i < newCriteriaCount; i++)
            {
                finalCriteria[i] = newCriteria[i];
            }
            string result = string.Join(", ", finalCriteria);
            try
            {
                StreamWriter streamWriter = new StreamWriter(path);
                using (streamWriter)
                {
                    streamWriter.WriteLine(result);
                    MessageBox.Show("Criteria deleted.", "Message",
okBtn, infoIcon);
                }
            }
            catch
            {
                MessageBox.Show("Error while reading the data.",
"Invalid", okBtn, exclamationIcon);
            }
        }
    }
}

public bool DeleteCheck(String criteriaName, String serviceName)
```

```
{
    if (serviceName.Equals("") && criteriaName.Equals(""))
    {
        MessageBox.Show("Please fill all the fields.", "Invalid");
    }
    else if (serviceName.Equals(""))
    {
        MessageBox.Show("Please select the service name.", "Invalid");
    }
    else if (criteriaName.Equals(""))
    {
        MessageBox.Show("Please enter the criteria.", "Invalid");
    }
    else
    {
        return true;
    }
    return false;
}

private void ExitBtn_Click(object sender, EventArgs e)
{
    string title = "Close Window";
    string message = "Do you want to close this window?";
    MessageBoxButtons buttons = MessageBoxButtons.YesNo;
    DialogResult result = MessageBox.Show(message, title, buttons);
    if (result == DialogResult.Yes)
    {
        StaffLogin loginAG = new StaffLogin();
        loginAG.Show();
        Hide();
    }
}
```

```
private void ClearBtn_Click(object sender, EventArgs e)
{
    serviceNameCB.SelectedItem = null;
    serviceUCB.SelectedItem = null;
    criteriaNameTB.Clear();
    idTB.Clear();
    criteriaUTB.Clear();
}
}
```

7. General.cs

```
using System;
using System.Windows.Forms;
namespace Cw1
{
    public partial class General : Form
    {
        public General()
        {
            InitializeComponent();
        }
        private void ReportBtn_Click(object sender, EventArgs e)
        {
            Report report = new Report();
            report.Show();
            Hide();
        }
        private void BackBtn_Click(object sender, EventArgs e)
        {
            StaffLogin loginAG = new StaffLogin();
            loginAG.Show();
            Hide();
        }
    }
}
```

```
private void ImportBtn_Click(object sender, EventArgs e)
{
    Import importBulkData = new Import();
    importBulkData.Show();
    Hide();
}
}
```

8. Report.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Windows.Forms;
using System.IO;
using System.Windows.Forms.DataVisualization.Charting;
using System.Drawing;
namespace Cw1
{
    public partial class Report : Form
    {
        MessageBoxButtons okBtn = MessageBoxButtons.OK;
        MessageBoxIcon exclamationIcon = MessageBoxIcon.Exclamation;
        public Report()
        {
            InitializeComponent();
            chartB.Enabled = false;
        }
        private void OpenToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            openFileDialog.ShowDialog();
            string pathName = openFileDialog.FileName;
            string filename = Path.GetFileName(pathName);
```

```
string criteriaPath= null;
if (pathName != "")
{
    if (filename.Equals("customerdetails1.csv"))
    {
        criteriaPath = "criteriaservice1.csv";
    }
    else if (filename.Equals("customerdetails2.csv"))
    {
        criteriaPath = "criteriaservice2.csv";
    }
    chart1.Series.Clear();
    chart1.Titles[0].Text = "";
    DataTableGV(pathName, criteriaPath);
}
}

private void DataTableGV(string detailsPath, string criteria_Path)
{
    try
    {
        DataTable data = new DataTable();
        string[] custDetails = File.ReadAllLines(detailsPath);
        string[] criteriaArray = File.ReadAllLines(criteria_Path);
        string criteria = criteriaArray[0];
        string[] criteriaSplit = criteria.Split(',');
        List<string> allCriteria = new List<string>();
        allCriteria.Add("Date");
        allCriteria.Add("Time");
        for(int i =0; i<criteriaSplit.Length; i++)
        {
            allCriteria.Add(criteriaSplit[i]);
        }
        List<string> newCustData = new List<string>();
```

```
if (custDetails.Length > 0)
{
    foreach (string header in allCriteria)
    {
        data.Columns.Add(new DataColumn(header));
    }
    for (int i = 0; i < custDetails.Length; i++)
    {
        string[] customerData = custDetails[i].Split(',');
        int custDataCount = customerData.Length;
        for (int j = 0; j < custDataCount; j++)
        {
            newCustData.Add(customerData[j]);
        }
        DataRow dataRow = data.NewRow();
        int columnIndex = 5;
        foreach (string criteriaData in allCriteria)
        {
            try
            {
                dataRow[criteriaData] =
customerData[columnIndex++];
            }
            catch
            {
                dataRow[criteriaData] = 0;
                columnIndex++;
            }
        }
        //Adding the rows in the table
        data.Rows.Add(dataRow);
    }
}
if (data.Rows.Count > 0)
```

```
        {
            tableGV.DataSource = data;
        }
        else
        {
            MessageBox.Show("The csv file is empty", "Invalid", okBtn,
exclamationIcon);
        }
        string service = newCustData[0];
        serviceNameTB.Text = service;
        chartB.Enabled = true;
    }
    catch (Exception)
    {
        MessageBox.Show("File cannot be opened. Please choose csv
file.", "Invalid", okBtn, exclamationIcon);
    }
}
private void ChartB_Click(object sender, EventArgs e)
{
    try{
        chart1.Series.Clear();
        string seriesname = "Series1";
        int colCount = tableGV.Columns.Count;
        int rowCount = tableGV.Rows.Count;
        List<string> headerNames = new List<string>();
        List<int> totalValues = new List<int>();
        for (int i = 2; i < colCount; i++)
        {
            headerNames.Add(tableGV.Columns[i].HeaderText);
            int individualTotal = 0;
            for (int j = 0; j < rowCount; j++)
            {
                try
```

```

        {
            individualTotal +=
Convert.ToInt16(tableGV.Rows[j].Cells[i].Value);
        }
        catch
        {
            individualTotal += 0;
        }
    }
    totalValues.Add(individualTotal);
}
decimal totalsum = 0;
for (int i = 0; i < colCount-2; i++)
{
    totalsum += totalValues[i];
}
chart1.Series.Add(seriesname);
chart1.Series[seriesname].ChartType =
SeriesChartType.Pie;
chart1.Series[seriesname].Points.DataBindY(totalValues);
List<decimal> percentList = new List<decimal>();
for (int i = 0; i < headerNames.Count; i++)
{
    decimal percent = (totalValues[i] / totalsum) * 100;
    decimal finalPercent = decimal.Round(percent, 2,
MidpointRounding.AwayFromZero);
    percentList.Add(finalPercent);
}
int count = percentList.Count;
decimal temp;
string criteriaTemp;
for (int i = 0; i < count; i++)
{
    for (int j = 0; j < count - 1; j++)

```



```

        {
            if (percentList[j] < percentList[j + 1])
            {
                temp = percentList[j];
                percentList[j] = percentList[j + 1];
                percentList[j + 1] = temp;
                criteriaTemp = headerNames[j];
                headerNames[j] = headerNames[j + 1];
                headerNames[j + 1] = criteriaTemp;
            }
        }
    }
    for (int i = 0; i < headerNames.Count; i++)
    {
        chart1.Series[seriesname].Points[i].Label = percentList[i] +
        @"%";
        chart1.Series[seriesname].Points[i].LegendText =
        headerNames[i];
    }
    chart1.Titles[0].Text = serviceNameTB.Text;
    chart1.Titles[0].Alignment = ContentAlignment.TopLeft;
}catch
{
    MessageBox.Show("No data in the table", "Invalid", okBtn,
    exclamationIcon);
}
private void ExitToolStripMenuItem_Click(object sender, EventArgs
e)
{
    string title = "Close Window";
    string message = "Do you want to close this window?";
    MessageBoxButtons buttons = MessageBoxButtons.YesNo;
    DialogResult result = MessageBox.Show(message, title, buttons);

```

```
        if (result == DialogResult.Yes)
        {
            General general = new General();
            general.Show();
            Hide();
        }
    }

    private void SearchBtn_Click(object sender, EventArgs e)
    {
        try
        {
            if (DateTime.Compare(dateFromP.Value.Date,
dateToP.Value.Date) > 0)
            {
                MessageBox.Show("Please select valid date");
                dateFromP.Focus();
                return;
            }
            DataTable dataTable = (DataTable)tableGV.DataSource;
            DataView dataView = new DataView();
            dataView = dataTable.DefaultView;
            dataView.RowFilter = String.Format("Date >= '{0:dd / MM /
yyyy}' AND Date <= '{1:dd / MM / yyyy}'", dateFromP.Value,
dateToP.Value);
        }
        catch
        {
            MessageBox.Show("Please open file first.", "Invalid", okBtn,
exclamationIcon);
        }
    }
}
```


9. Import.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;
namespace Cw1
{
    public partial class Import : Form
    {
        MessageBoxButtons okBtn = MessageBoxButtons.OK;
        MessageBoxIcon exclamationIcon = MessageBoxIcon.Exclamation;
        MessageBoxIcon infoIcon = MessageBoxIcon.Information;
        public Import()
        {
            InitializeComponent();
        }
        private void UploadBtn_Click(object sender, EventArgs e)
        {
            openFileDialog.ShowDialog();
            string pathName = openFileDialog.FileName;
            string filename = Path.GetFileName(pathName);
            string fileExtension = Path.GetExtension(pathName);
            if (filename.Equals("customerdetails1.csv"))
            {
                MessageBox.Show("Please select other file.", "Invalid", okBtn,
exclamationIcon);
            }
            else if (filename.Equals("customerdetails2.csv"))
            {
                MessageBox.Show("Please select other file.", "Invalid", okBtn,
exclamationIcon);
            }
            else if (fileExtension.Equals(".csv"))
            {

```

```
        UploadBulk(pathName);
    }
    else
    {
        MessageBox.Show("Please select other file.", "Invalid", okBtn,
exclamationIcon);
    }
}
public void UploadBulk(string bulkFilePath)
{
    try
    {
        string[] bulkLines = File.ReadAllLines(bulkFilePath);
        string[] serviceLine = bulkLines[1].Split(',');
        if (serviceLine[0].Equals("Service 1") ||
serviceLine[0].Equals("Service 2"))
        {
            if (bulkLines.Length > 0)
            {
                for (int i = 0; i < bulkLines.Length - 1; i++)
                {
                    List<string> dataS1 = new List<string>();
                    List<string> dataS2 = new List<string>();
                    string[] bulkData = bulkLines[i + 1].Split(',');
                    int bulkDataCount = bulkData.Length;
                    if (bulkData[0].Equals("Service 1"))
                    {
                        for (int a = 0; a < bulkDataCount; a++)
                        {
                            dataS1.Add(bulkData[a]);
                        }

                        string[] finalDataS1 = new String[bulkDataCount];
                        for (int s = 0; s < bulkDataCount; s++)
```

```
{
    finalDataS1[s] = dataS1[s];
}
string result = string.Join(",", finalDataS1);
try
{
    string pathName = "customerdetails1.csv";
    StreamWriter streamWriter = new
StreamWriter(pathName, true);
    using (streamWriter)
    {
        streamWriter.WriteLine(result);
    }
    streamWriter.Close();
}
catch (Exception)
{
    MessageBox.Show("Error while writing the data.",
"Invalid", okBtn, exclamationIcon);
}
}
else
{
    for (int a = 0; a < bulkDataCount; a++)
    {
        dataS2.Add(bulkData[a]);
    }
    string[] finalDataS2 = new String[bulkDataCount];
    for (int s = 0; s < bulkDataCount; s++)
    {
        finalDataS2[s] = dataS2[s];
    }
    string result = string.Join(",", finalDataS2);
    try
```

```
        {
            string pathName = "customerdetails2.csv";
            StreamWriter streamWriter = new
StreamWriter(pathName, true);
            using (streamWriter)
            {
                streamWriter.WriteLine(result);
            }
            streamWriter.Close();
        }
        catch (Exception)
        {
            MessageBox.Show("Error while writing the data.",
"Invalid", okBtn, exclamationIcon);
        }
    }
    uploadTB.Text = bulkFilePath;
    MessageBox.Show("Successfully imported", "Message",
okBtn, infoIcon);
}
else
{
    MessageBox.Show("Invalid file. Please check the file again.",
"Invalid", okBtn, exclamationIcon);
}
}
catch
{
    MessageBox.Show("Error while importing.", "Invalid", okBtn,
exclamationIcon);
}
}
```

```
private void ClearBtn_Click(object sender, EventArgs e)
{
    uploadTB.Text = "";
}
private void ExitBtn_Click(object sender, EventArgs e)
{
    string title = "Close Window";
    string message = "Do you want to close this window?";
    MessageBoxButtons buttons = MessageBoxButtons.YesNo;
    DialogResult result = MessageBox.Show(message, title, buttons);
    if (result == DialogResult.Yes)
    {
        General general = new General();
        general.Show();
        Hide();
    }
}
}
```