

# Informatics College Pokhara



## Application Development

### CS6004NP

#### Coursework 1

**Submitted By:**

Student Name: Kushal Gurung

London Met ID: 18029022

Group: L3C3

Date: 24-Jan-2021

**Submitted To:**

Mr. Sachin Subedi

Module Leader

Application Development

## Table of Contents

1.	Introduction .....	1
1.1	Problem Statement .....	1
1.2	Project as a solution .....	2
2.	Background.....	3
2.1	Explanation of Features .....	4
2.1.1	Customer Rating and Feedback Dashboard .....	4
2.1.2	Admin Login .....	5
2.1.3	Admin Dashboard .....	6
2.1.4	Add Criteria .....	6
2.1.5	Customer Report .....	7
2.1.6	Overall Report .....	8
2.1.7	Bulk Data Entry .....	8
2.1.8	Mean Rating Bar Graph .....	9
2.1.9	Overall Rating Pie Chart.....	9
3.	Development.....	10
3.1	System Architecture.....	10
3.2	Use Case diagram of the system .....	11
3.3	Flowchart .....	12
3.3.1	Flowchart of Customer Rating and Feedback Dashboard .....	12
3.3.2	Flowchart of Admin Login.....	13
3.3.3	Flowchart of Add Criteria.....	14
3.3.4	Flowchart of View Report .....	15
3.3.5	Flowchart of Bulk Data Entry .....	16
3.3.6	Flowchart of Graph.....	17
3.4	Entity Relationship Diagram of the system .....	18
3.5	Description of Classes and Methods .....	19
3.5.1	CustomerUI .....	19
3.5.2	AdminDashboard .....	20
3.5.3	CriteriaAdd .....	21
3.5.4	ReportOption.....	21
3.5.5	CustomerReport.....	21
3.5.6	OverallReport .....	22
3.5.7	Bulk_Data_Entry .....	23
3.5.8	Columns.....	23

3.5.9	GraphOption.....	24
3.5.10	BarGraph .....	24
3.5.11	PieChart.....	25
3.5.12	LoginForm.....	25
3.5.13	Program .....	25
3.5.14	DataHandler.....	26
4.	Flowchart and Algorithm of mean rating bar graph .....	27
5.	Flowchart and Algorithm of overall rating pie chart .....	28
6.	Data structure .....	29
7.	User Manual .....	30
8.	Conclusion .....	37
9.	Bibliography .....	39
10.	Appendix.....	40

## Table of Figures

Figure 1: Customer Rating and Feedback Dashboard.....	4
Figure 2: Admin Login UI .....	5
Figure 3: Admin Dashboard .....	6
Figure 4: Add Criteria Form .....	7
Figure 5: Customer Report .....	7
Figure 6: Overall Report.....	8
Figure 7: Bulk Data Entry UI .....	8
Figure 8: Mean Rating Bar Graph .....	9
Figure 9: Overall Rating Pie Chart .....	9
Figure 10: System Architecture .....	10
Figure 11: Use Case diagram of the system.....	11
Figure 12: Flowchart of Customer Rating and Feedback Dashboard .....	12
Figure 13: Flowchart of Admin Login .....	13
Figure 14: Flowchart of Add Criteria .....	14
Figure 15: Flowchart of View Report.....	15
Figure 16: Flowchart of Bulk Data Entry .....	16
Figure 17: Flowchart of Graph .....	17
Figure 18: Entity Relationship Diagram of the system .....	18
Figure 19: Flowchart of mean rating bar graph .....	27
Figure 20: Flowchart of overall rating pie chart .....	28
Figure 21: Customer Rating and Feedback Dashboard.....	30
Figure 22: Admin Login.....	31
Figure 23: Admin Dashboard .....	32
Figure 24: Add Criteria Form .....	32
Figure 25: Report Option Form .....	33
Figure 26: Customer Report .....	33
Figure 27: Overall Report.....	34
Figure 28: Bulk Data Entry UI .....	34
Figure 29: Graph Option Form .....	35
Figure 30: Bar Graph of mean rating .....	35
Figure 31: Pie Chart of overall report.....	36

## **1. Introduction**

This is the first individual coursework of our Application Development module. It is based on development of criteria rating and feedback system. The system was developed using Windows form and C# programming language in Visual Studio 2019 Integrated Development Environment (IDE). Let's go more deeper into the system's mechanism. The system was built in such a way that when customers rate certain criteria of the Restaurant service, all of their details such as their name, contact number, address, email address, etc along with their rating values will be saved in a separate xml file, however customers have the option to not provide their details while rating the system.

Similarly, the system has an admin dashboard. To access the admin dashboard, user will have to enter admin login details. Admin can manage various things in the system. He/she can add more criteria of the Restaurant service, view Customer report and Overall report. Customer report will have the records of customer's details along with their ratings, whereas Overall report will display the records of each criteria and how many ratings has each criteria got for four different categories, which are Excellent, Good, Average, and Dissatisfied. Overall rating's table also has an overall column, where it shows the total rating of each criteria. Likewise, admin can also import bulk entry of customer's rating. Lastly, admin can view the pie chart and bar graph of the reports.

### **1.1 Problem Statement**

Talking about the current context of Nepal, small number of organizations use software for rating and feedback system. However, most of the organizations do not use software for collecting customer's ratings and feedbacks. They are still dependent on old manual system. We all know how difficult it is to manage data in the manual system, because each day there will be hundreds of new hard copies which will be difficult to manage. In case, if we want to get the overall record of certain criteria of a service using manual system, it will take lots of time, effort and even after that there is no guarantee that the output will come accurate.

## **1.2 Project as a solution**

As we are now aware of the fact that how challenging it is to save, retrieve and maintain data when we use manual system. It may not be that much difficult for a small restaurant, but for bigger restaurants and bigger organizations, it is almost impossible to use manual system. Therefore, the rating and feedback system that I have built in this project will be able to save the time, provide accurate overall report, maintain data easily, save new data and retrieve data easily. It also has the feature of importing bulk entry, if in case the admin wants to load bulk of data from a CSV file and save it into the system. Hence, this system will be able to solve all the problems that I have stated above.

## 2. Background

Rating and Feedback system is a very important system in today's world, more specifically in a business sector. They play an important role to grow business for any organization, whether that may be for a restaurant, five star hotels or an e-commerce site. However, it is not just business sector that this system provides benefit to. Let's take an example of one of the biggest social media application "Facebook". We all give ratings and feedback in different features of Facebook. The most common features are voice calls, and video calls. We give certain ratings after each audio or voice calls. Also, we give our feedback on Facebook when we see some inappropriate posts. Now, Facebook utilizes those ratings and feedbacks and try to make their application more effective, which will eventually make their system even more advanced. Therefore, rating and feedback system is very important in modern days.

The features of Rating and Feedback System are listed below:

- ❖ Customer Rating and Feedback Dashboard
- ❖ Admin Login
- ❖ Admin Dashboard
- ❖ Add Criteria
- ❖ Customer Report
- ❖ Overall Report
- ❖ Bulk Data Entry
- ❖ Mean Rating Bar Graph
- ❖ Overall Rating Pie Chart

## 2.1 Explanation of Features

The features of Rating and Feedback system are explained along with their images below:

### 2.1.1 Customer Rating and Feedback Dashboard

Customer Rating and Feedback Dashboard is the first User Interface (UI) that will be displayed when running the application. The main purpose of this UI is to allow customers to provide rating and feedback to different criteria of the Restaurant service. When users give ratings to all the criteria and clicks on “SUBMIT” button, an XML file named as **RatingsData.xml** and an XML Schema file named as **RatingsSchema.xml** will be created and saved in the project’s bin folder. The XML Schema file was created to define the structure and the element tags required for the XML file, whereas the XML file was created to store the submitted rating and feedback data (Chand, 2017).

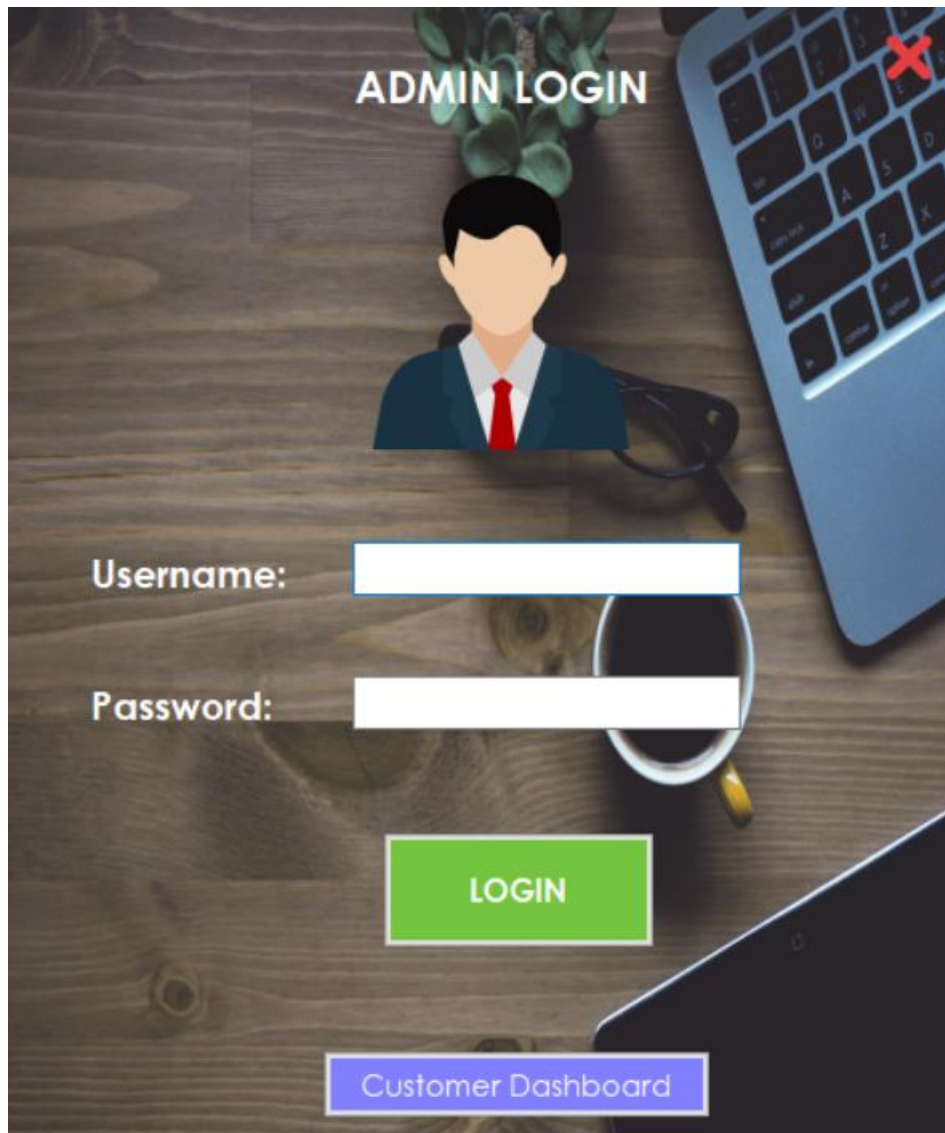
SN	Criteria	Description	Excellent	Good	Average	Dissatisfied
1	Food Order	Food Ordering criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Waiter Service	Waiter Service criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Food Quality	Food Quality criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	Greeting	Greeting criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	Reception	Reception criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	Hospitality	Hospitality criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7	Environment	Environment criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8	WiFi	WiFi criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 1: Customer Rating and Feedback Dashboard



### 2.1.2 Admin Login

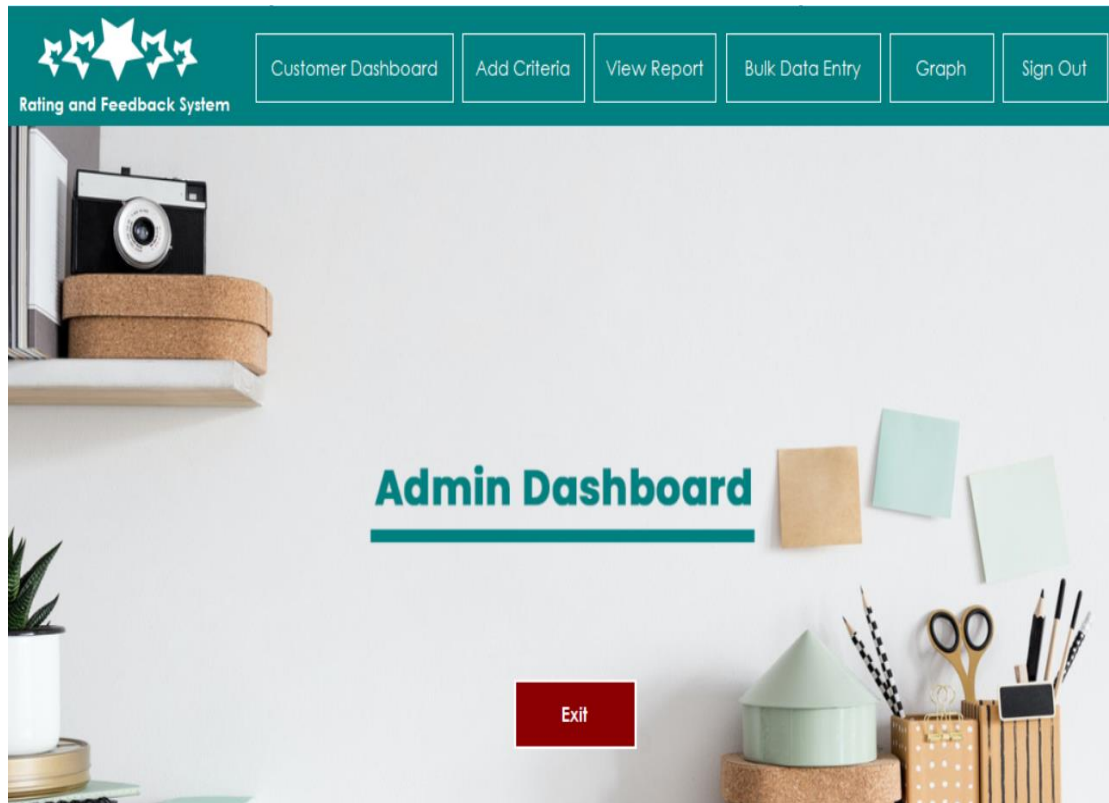
In the Admin Login UI, user will have to enter their username and password to login into the system as admin. The username is set as “Kushal” and the password is set as “Gurung”. When correct login details are entered and “LOGIN” button is clicked, Admin Dashboard will be displayed.



*Figure 2: Admin Login UI*

### 2.1.3 Admin Dashboard

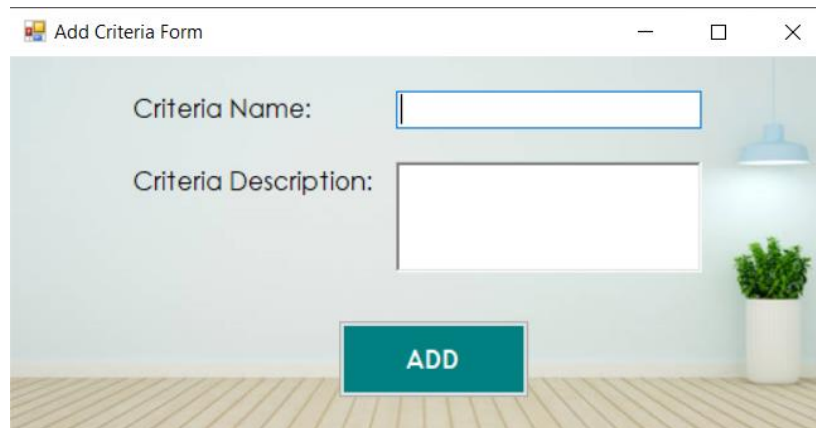
Admin Dashboard is the UI which will help the admin to use different features of the system. From this UI, admin can click on “Add Criteria” button to add new criteria, “View Report” button to view customer report and overall report, “Bulk Data Entry” button to enter bulk of data, “Graph” button to view bar graph of mean rating and pie chart of overall rating, “Sign Out” button to sign out of admin dashboard, and lastly “Exit” button to close the application.



*Figure 3: Admin Dashboard*

### 2.1.4 Add Criteria

This UI will help the admin to add new criteria of the Restaurant service. After “ADD” button is clicked, the system will first read previous criteria data from **CriteriaData.xml** XML file and **CriteriaSchema.xml** XML Schema file, and then finally add the new criteria in the same xml file.



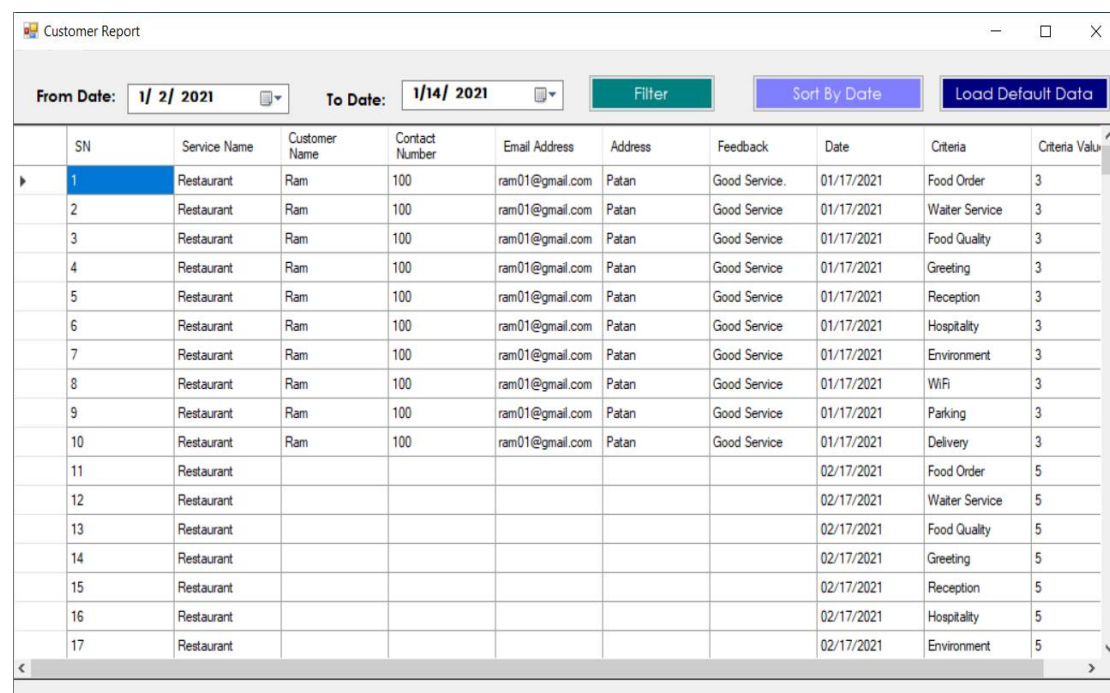
The image shows a window titled "Add Criteria Form". It contains two input fields: "Criteria Name:" with a single-line text box, and "Criteria Description:" with a multi-line text box. Below these fields is a green button labeled "ADD". The background of the form is a light blue wall with a white floor, a small potted plant, and a blue pendant light.

Figure 4: Add Criteria Form

### 2.1.5 Customer Report

Customer report will display the customer data and the ratings that they have provided in the Customer Rating and Feedback Dashboard. Admin can view the report of particular dates by clicking on "Filter" button after choosing a starting date and an end date. The data that we see in the DataGridView are read from **RatingsData.xml** XML file and **RatingsSchema.xml** XML Schema file. This UI will also allow the admin to sort data by date.

When we click on "Filter" or "Sort By Date" button, the data in DataGridView will change. Hence, "Load Default Data" button will display the original default data in the DataGridView while admin clicks on it.



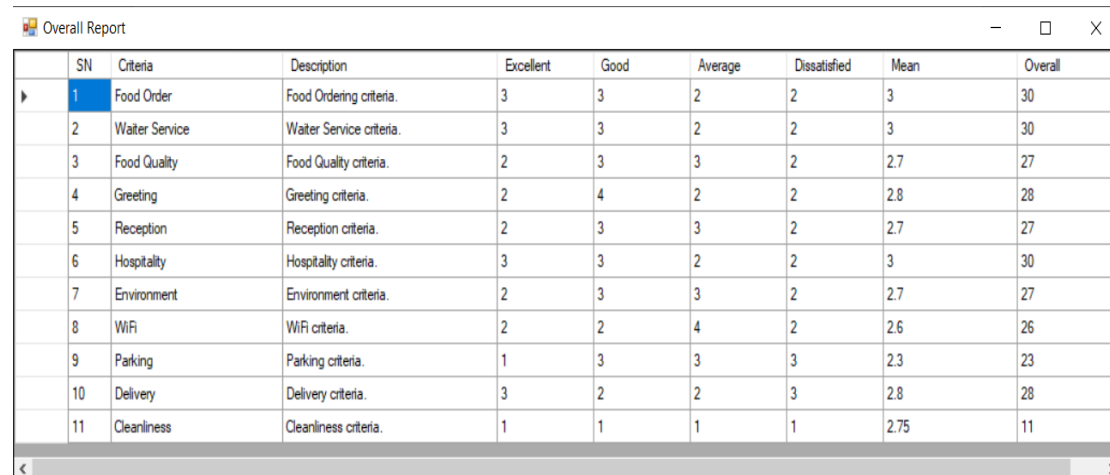
The image shows a window titled "Customer Report". It has a header section with "From Date:" (1/ 2/ 2021), "To Date:" (1/14/ 2021), and three buttons: "Filter", "Sort By Date", and "Load Default Data". Below the header is a DataGridView table with 11 columns: SN, Service Name, Customer Name, Contact Number, Email Address, Address, Feedback, Date, Criteria, and Criteria Value. The table contains 17 rows of data, with the first row highlighted in blue.

SN	Service Name	Customer Name	Contact Number	Email Address	Address	Feedback	Date	Criteria	Criteria Value
1	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service.	01/17/2021	Food Order	3
2	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Waiter Service	3
3	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Food Quality	3
4	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Greeting	3
5	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Reception	3
6	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Hospitality	3
7	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Environment	3
8	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	WiFi	3
9	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Parking	3
10	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Delivery	3
11	Restaurant						02/17/2021	Food Order	5
12	Restaurant						02/17/2021	Waiter Service	5
13	Restaurant						02/17/2021	Food Quality	5
14	Restaurant						02/17/2021	Greeting	5
15	Restaurant						02/17/2021	Reception	5
16	Restaurant						02/17/2021	Hospitality	5
17	Restaurant						02/17/2021	Environment	5

Figure 5: Customer Report

### 2.1.6 Overall Report

This is the overall report which shows how many customers have gave Excellent, Good, Average, Dissatisfied rating for each criteria. The last two columns in the DataGridView shows mean rating and overall rating.

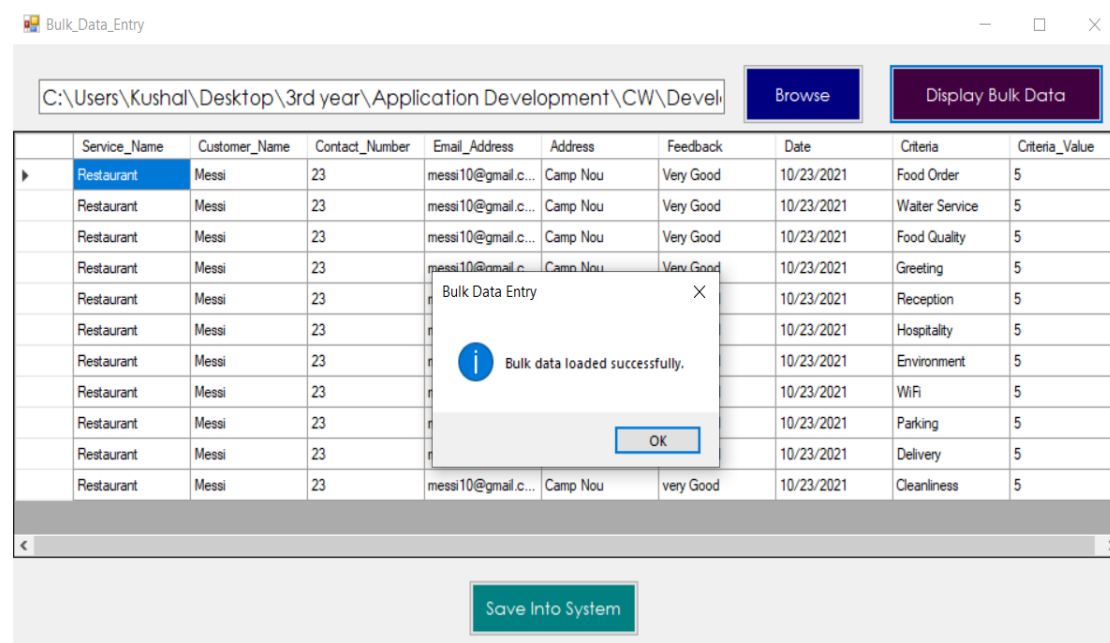


SN	Criteria	Description	Excellent	Good	Average	Dissatisfied	Mean	Overall
1	Food Order	Food Ordering criteria.	3	3	2	2	3	30
2	Waiter Service	Waiter Service criteria.	3	3	2	2	3	30
3	Food Quality	Food Quality criteria.	2	3	3	2	2.7	27
4	Greeting	Greeting criteria.	2	4	2	2	2.8	28
5	Reception	Reception criteria.	2	3	3	2	2.7	27
6	Hospitality	Hospitality criteria.	3	3	2	2	3	30
7	Environment	Environment criteria.	2	3	3	2	2.7	27
8	WiFi	WiFi criteria.	2	2	4	2	2.6	26
9	Parking	Parking criteria.	1	3	3	3	2.3	23
10	Delivery	Delivery criteria.	3	2	2	3	2.8	28
11	Cleanliness	Cleanliness criteria.	1	1	1	1	2.75	11

Figure 6: Overall Report

### 2.1.7 Bulk Data Entry

Bulk Data Entry feature will allow the admin to import bulk data from a CSV file. However, if admin mistakenly selects a wrong file, a message box will appear to inform the admin to select the correct file. After importing bulk data from correct file, when admin clicks on “Save Into System” button, the displayed data will be saved in Ratings XML file.



Service_Name	Customer_Name	Contact_Number	Email_Address	Address	Feedback	Date	Criteria	Criteria_Value
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Food Order	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Waiter Service	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Food Quality	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Greeting	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Reception	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Hospitality	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Environment	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	WiFi	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Parking	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Delivery	5
Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Cleanliness	5

Figure 7: Bulk Data Entry UI

### 2.1.8 Mean Rating Bar Graph

Mean Rating Bar Graph shows the mean rating of each criteria. The data for mean rating is read from **ReportSchema.xml** XML Schema file and **ReportData.xml** XML file (thedeveloperblog, n.d.).

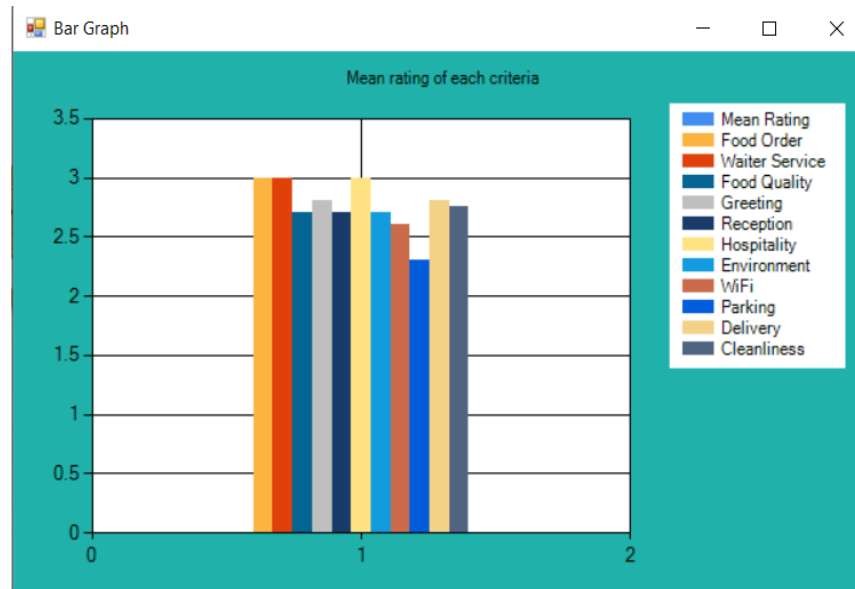


Figure 8: Mean Rating Bar Graph

### 2.1.9 Overall Rating Pie Chart

Overall Rating Pie Chart shows the overall rating of each criteria. The data for overall rating is read from same files as for mean rating, which are **ReportSchema.xml** XML Schema file and **ReportData.xml** XML file (Nayak, 2012).

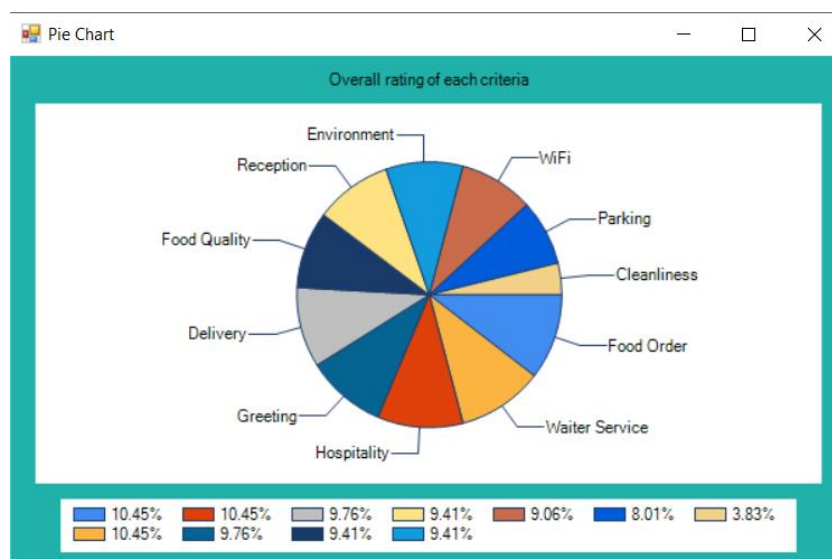
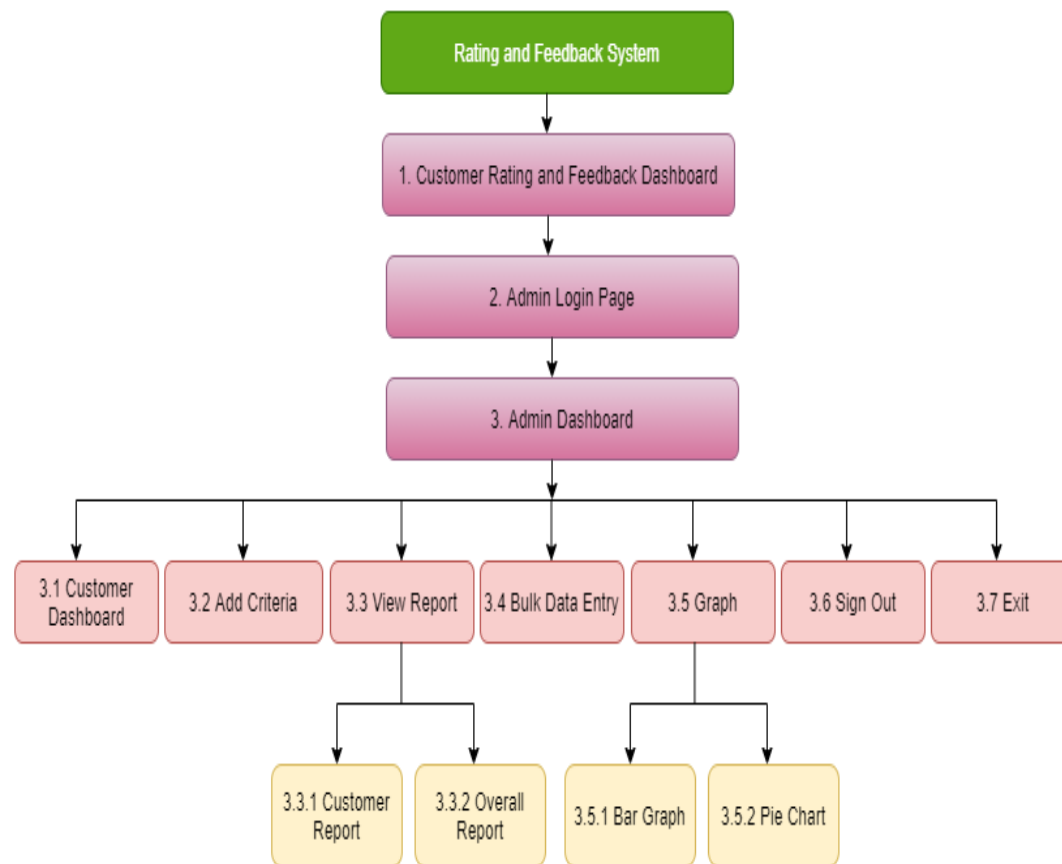


Figure 9: Overall Rating Pie Chart

### 3. Development

#### 3.1 System Architecture



*Figure 10: System Architecture*

Above diagram is the architecture of the Rating and Feedback system. When the system is started, the first UI that we will see is 1. Customer Rating and Feedback Dashboard. From this UI, we can visit 2. Admin Login Page. From login page, we have to provide correct login details to get access to 3. Admin Dashboard. From 3. Admin Dashboard, we can get access to 3.1 Customer Dashboard, 3.2 Add Criteria, 3.3 View Report, 3.4 Bulk Data Entry, 3.5 Graph, 3.6 Sign Out, and 3.7 Exit. From 3.3 View Report, we can view 3.3.1 Customer Report and 3.3.2 Overall Report. Finally, from 3.5 Graph, we can view 3.5.1 Bar Graph of mean rating and 3.5.2 Pie Chart of overall rating.

### 3.2 Use Case diagram of the system

Use Case diagram of the system is shown in the image below:

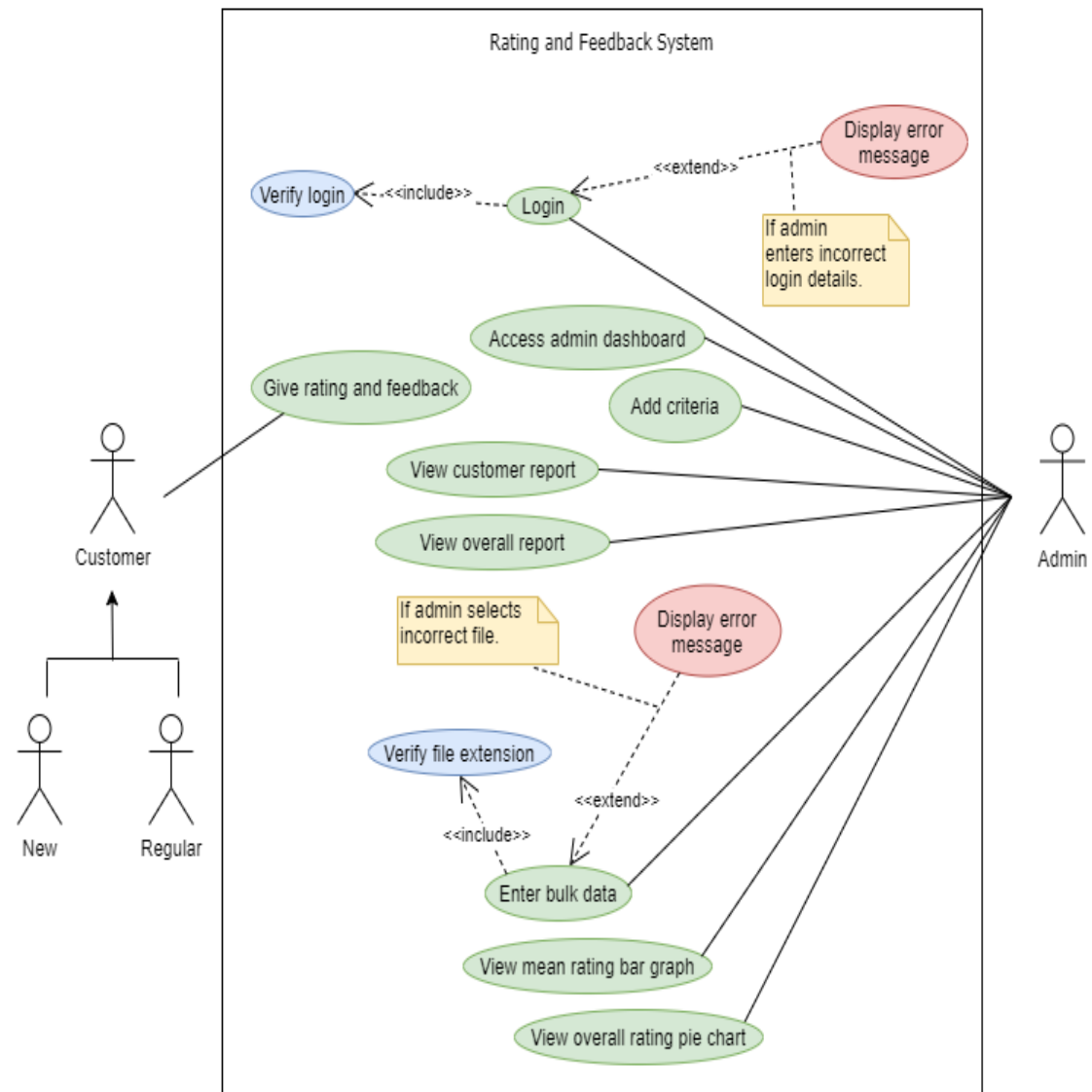


Figure 11: Use Case diagram of the system

### 3.3 Flowchart

Flowchart of the system's features are shown in the images below:

#### 3.3.1 Flowchart of Customer Rating and Feedback Dashboard

This is the flowchart of Customer Rating and Feedback Dashboard when "SUBMIT" button is clicked.

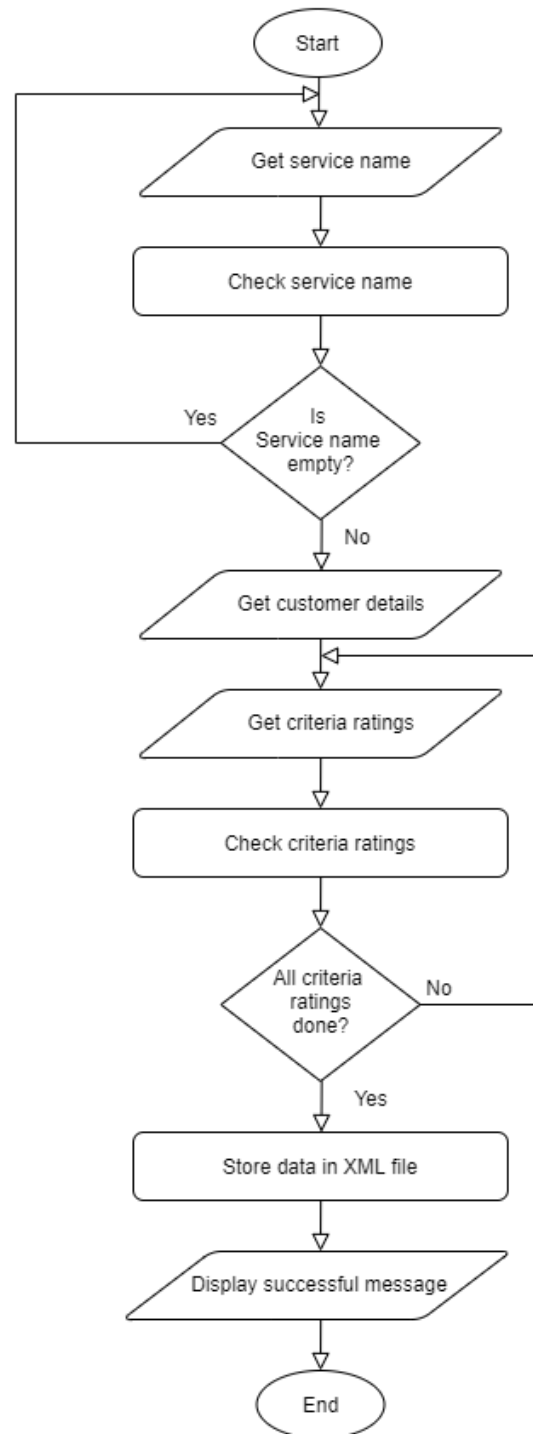


Figure 12: Flowchart of Customer Rating and Feedback Dashboard



### 3.3.2 Flowchart of Admin Login

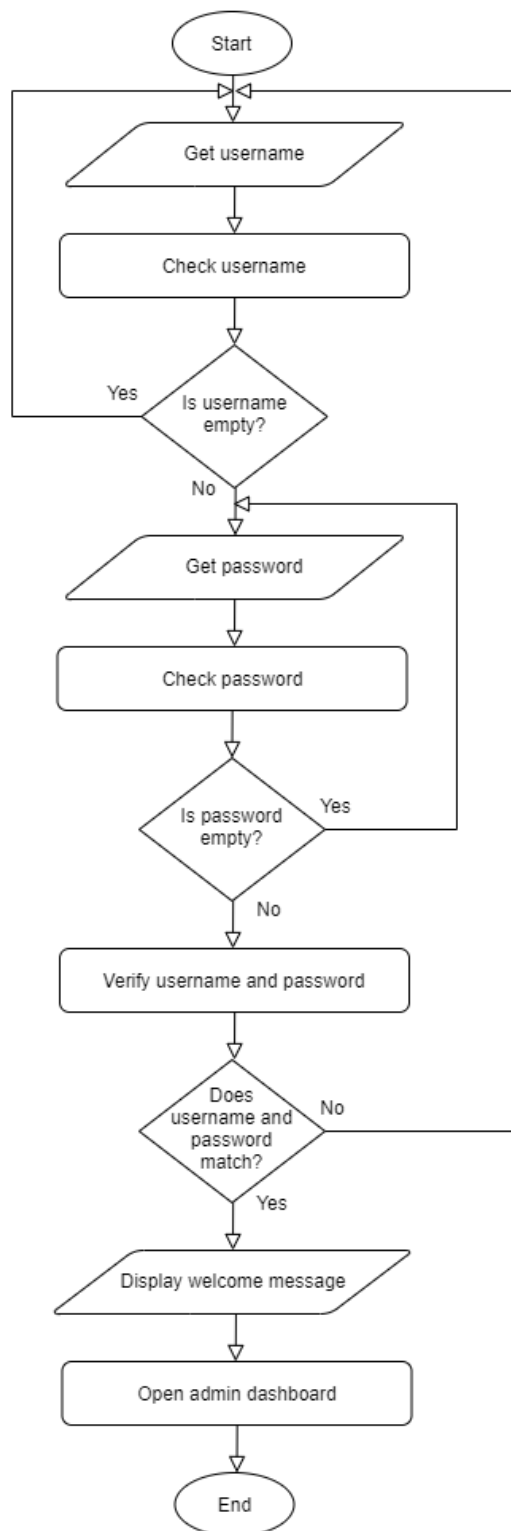


Figure 13: Flowchart of Admin Login

### 3.3.3 Flowchart of Add Criteria

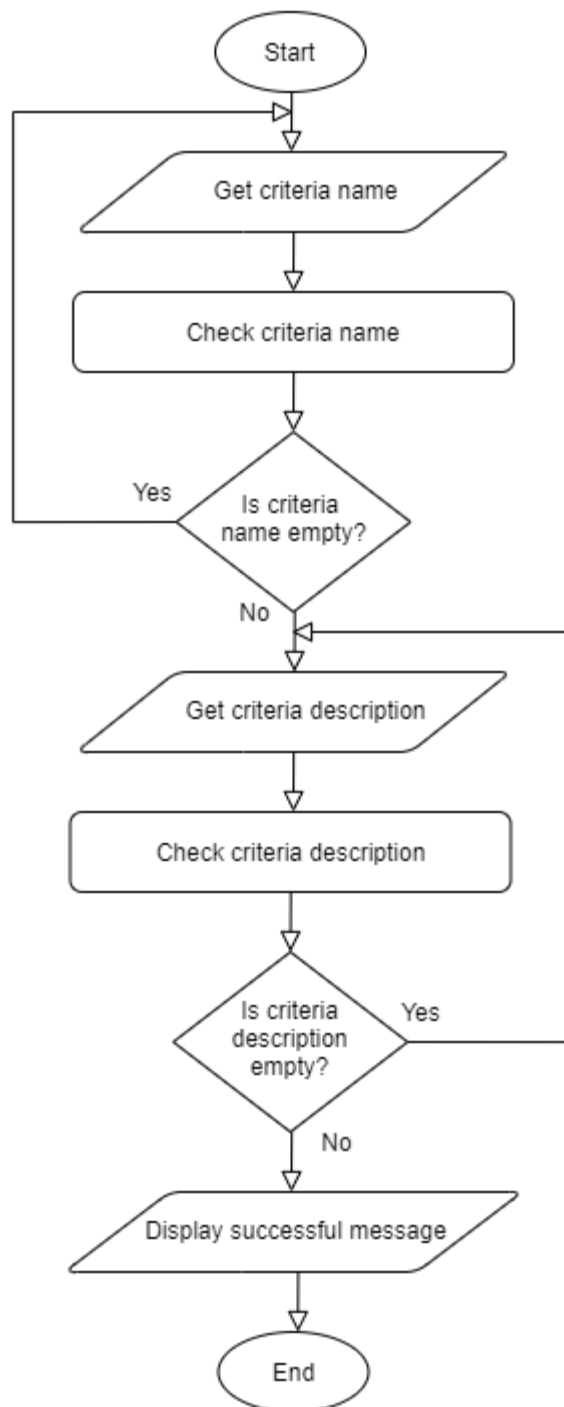


Figure 14: Flowchart of Add Criteria

### 3.3.4 Flowchart of View Report

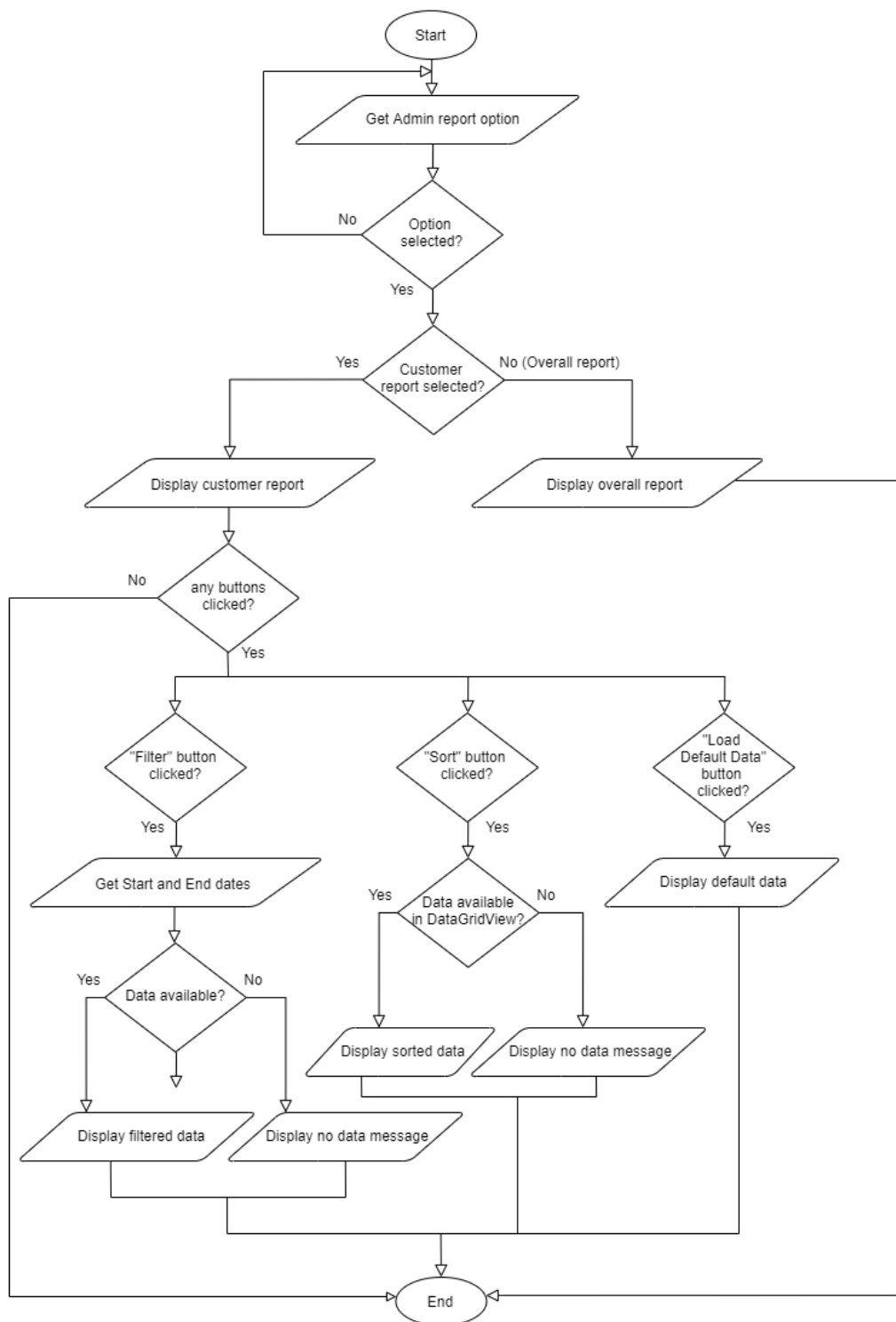


Figure 15: Flowchart of View Report

### 3.3.5 Flowchart of Bulk Data Entry

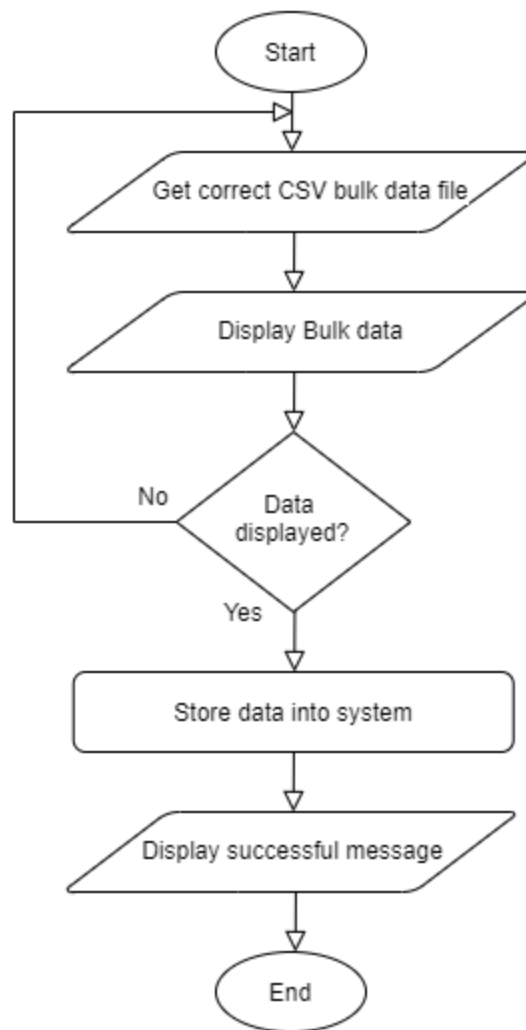


Figure 16: Flowchart of Bulk Data Entry

### 3.3.6 Flowchart of Graph

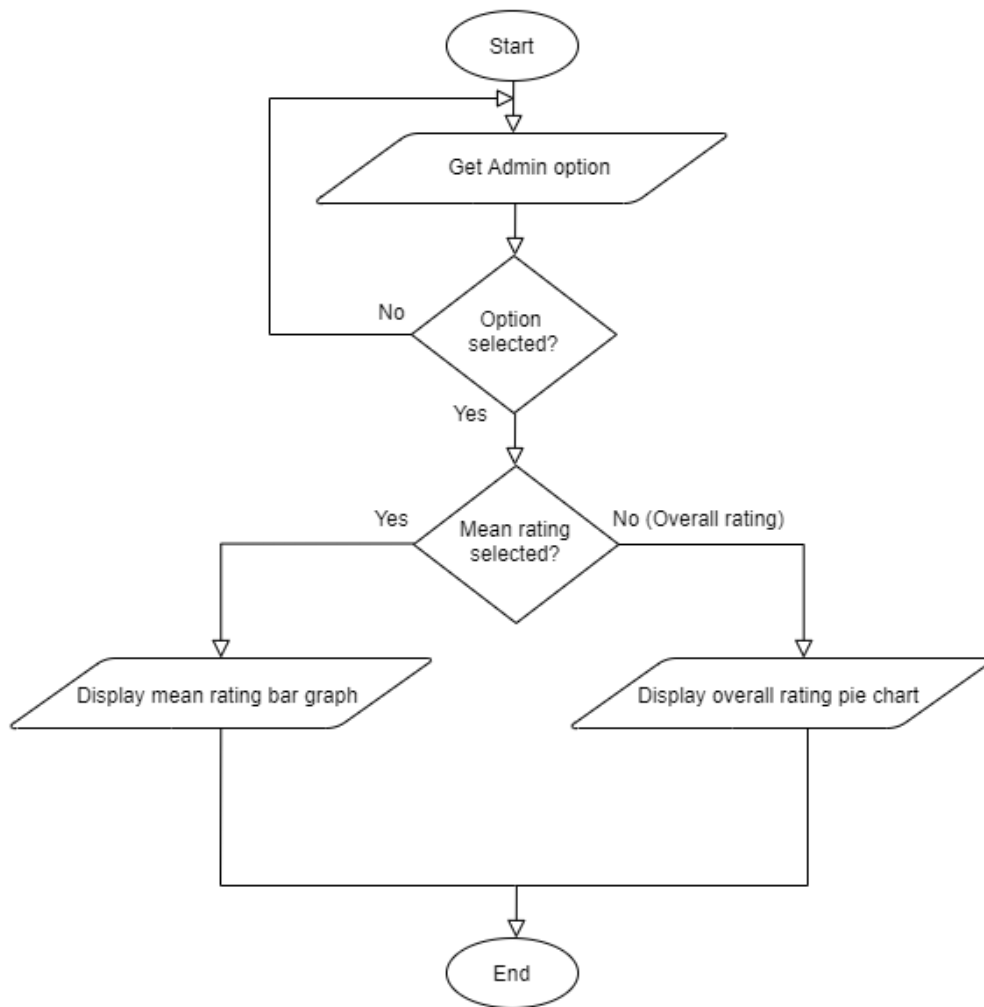


Figure 17: Flowchart of Graph

### 3.4 Entity Relationship Diagram of the system

Entity Relationship Diagram of the system is shown in the image below:

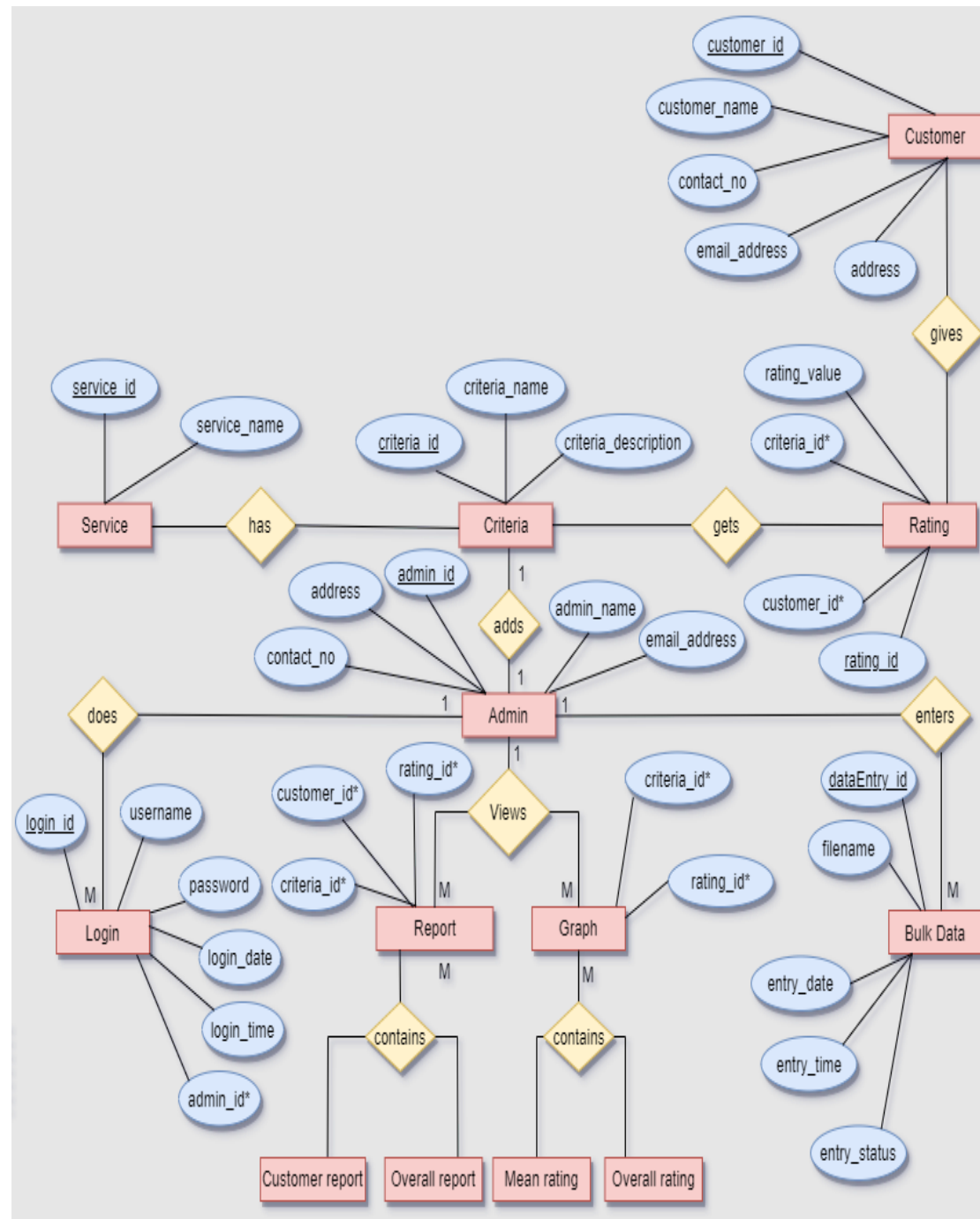


Figure 18: Entity Relationship Diagram of the system

### 3.5 Description of Classes and Methods

#### 3.5.1 CustomerUI

CustomerUI class is responsible for displaying the Customer Rating and Feedback Dashboard and allowing customers to give rating to different criterias of the Restaurant Service. The global property used in this class is “thread” which is the object of Thread pre-defined class. It is used in the project by using “System.Threading” namespace. When “Login as Admin” button is clicked, Customer UI will be closed and thread will open the admin login page. Similarly, the methods used in this class are:

- ❖ **dataGridView1\_CellContentClick:** dataGridView1\_CellContentClick method helps customer to select only one rating option for one criteria, i.e. either Excellent, Good, Average or Dissatisfied. It means that we can only select on one rating checkbox in a row.
- ❖ **submitButton\_Click:** The main role of submitButton\_Click method is to submit the customer rating and save it in an XML file. While clicking on “SUBMIT” button, “Restaurant” Service Name combo-box must be selected, all the criterias’ rating should be done in order to submit the rating, else the system will show exception message through MessageBox. Then, it will call “submitRatingsData” method to complete the process.
- ❖ **getCriteriaData:** This method retrieves criteria data from **CriteriaSchema.xml** XML Schema file and **CriteriaData.xml** XML file. Then, those criteria details are added in the DataGridView of the rating dashboard.
- ❖ **submitRatingsData:** This method calls another method, “AddSampleDataForRating” to get new customer rating. After getting rating values, they are written in **RatingsData.xml** XML file and **RatingsSchema.xml** XML Schema file.
- ❖ **AddSampleDataForRatings:** This method saves all the rating data from Ratings XML and XML Schema file in Report table of dataset. Then, after customer gives rating, the new data are added in the same Report table of dataset. This method also defines the value of four rating column, such as 5 for Excellent, 3 for Good, 2 for Average, and 1 for Dissatisfied.

- ❖ **pictureBox1\_Click:** This method closes the system.
- ❖ **loginAsAdminButton\_Click:** This method closes the rating dashboard and calls “OpenLoginForm” method.
- ❖ **OpenLoginForm:** This method opens the admin login page.

### 3.5.2 AdminDashboard

AdminDashboard class is responsible for displaying the admin dashboard. It includes several buttons in it which allows admin to control the system. The global property used in this class is “thread” similar to CustomerUI class. When “Customer Dashboard” button is clicked, admin dashboard will be closed and thread will start customer rating and feedback dashboard. Likewise, when “Sign Out” button is clicked, admin dashboard will be closed and admin login page will be opened by thread. The methods used in this class are:

- ❖ **addCriteriaBtn\_Click:** This method creates an object of CriteriaAdd class to display Criteria Add form.
- ❖ **viewReportBtn\_Click:** This method creates an object of ReportOption class to display report option form.
- ❖ **graphButton\_Click:** This method creates an object of GraphOption class to display graph option form.
- ❖ **exitButton\_Click:** This method closes the application.
- ❖ **signOutButton\_Click:** This method closes admin dashboard and calls “OpenLoginForm” method.
- ❖ **OpenLoginForm:** This method opens admin login form.
- ❖ **bulkDataEntryBtn\_Click:** This method creates an object of Bulk\_Data\_Entry class to display Bulk Data Entry form.
- ❖ **customerDashboardButton\_Click:** This method closes admin dashboard and calls “OpenCustomerUI” method.
- ❖ **OpenCustomerUI:** This method opens Customer Rating and Feedback Dashboard.



### 3.5.3 CriteriaAdd

CriteriaAdd class helps the admin to add a new criteria of the Restaurant service along with its description. The methods used in CriteriaAdd class are:

- ❖ criteriaAddButton\_Click: This method checks if there are any exception in the Criteria Add form, if there are no exception, it calls “SaveDataToFile” method.
- ❖ SaveDatatoFile: This method gets all the criteria along with the new criteria from “AddSampleData” method and writes them on criteria xml and xml schema files.
- ❖ AddSampleData: This method first saves all the criteria from criteria xml and xml schema file in Criteria table of dataset. Then, after admin adds a new criteria, it is added in the same Criteria table of dataset.

### 3.5.4 ReportOption

ReportOption class is responsible for displaying a form and requesting admin to choose either “Customer Report” or “Overall Report”. If “Customer Report” is selected, customer report will be displayed, and if “Overall Report” is selected, overall report will be displayed. The methods used in this class are:

- ❖ customerReport\_Click: This method creates an object of CutomerReport class to display customer report.
- ❖ overallReportButton\_Click: This method creates an object of OverallReport class to display overall report.
- ❖ pictureBox1\_Click: this method closes the report option form.

### 3.5.5 CustomerReport

CustomerReport class helps the admin to view customer’s details and the ratings that they have given for each criteria. In this class, two global properties are used: “dataTable” which is an object of DataTable class, and “dataView” which is an object of DataView class. The property dataTable is used to store the all data of ratings xml files, whereas dataView is used to display the data of dataTable in the customer report’s DataGridView. The methods used in this class are:

- ❖ **GetCustomerReport:** This method creates an object of DataTable class. In data table, the columns are created and all the ratings values are added from our ratings xml and schema files. Then, a data view object is created and the data table containing the ratings value are passed as an argument in it. Finally, the data source of “customerReportTable” DataGridView is initialized with the data view.
- ❖ **CustomerReport\_Load:** This method just calls the “GetCustomerReport” method.
- ❖ **filterButton\_Click:** This method takes start and end date from two date time pickers. Then, it filters the customer report that lies between these selected dates.
- ❖ **sortByDateButton\_Click:** This method sorts the customer report by date in ascending form.
- ❖ **loadDefaultDataButton\_Click:** This method loads the default customer report because data in DataGridView can change while clicking on “Filter” or “Sort” button.

### 3.5.6 OverallReport

OverallReport class displays the overall report of all the criteria. The methods used in this class are:

- ❖ **getReportData:** This method reads the data of criteria and ratings from criteria and ratings xml and their schema files respectively. Then, it will count the number of customers that have given ratings for Excellent, Good, Average and Dissatisfied options. Similarly, if there is null rating for a certain criteria, the system will give 0 count to it. Criteria can get null value only when admin have just added new criteria, and no customer has given any rating to it yet. Then, it will call “AddSampleDataForReport” method to get overall report data from Report table of data set. After getting overall data, it will be stored in **ReportSchema.xml** XML schema file and **ReportData.xml** XML file.
- ❖ **AddSampleDataForReport:** This method will store all the data of “reportTable” DataGridView in Report table of dataset.

### 3.5.7 Bulk\_Data\_Entry

Bulk\_Data\_Entry class is responsible for importing rating bulk data from a csv file, and saving the imported data to the system's ratings file. The methods used in this class are:

- ❖ browseButton\_Click: This method opens a file dialog box for admin to choose a file. When a file is chosen, its location will be displayed on the text box.
- ❖ LoadCSV: This method reads the data of bulk csv file from the file location of text box. Then, it gives data to each method of Columns class and returns data in the form of list.
- ❖ loadDataButton\_Click: This method displays the data in DataGridView that are returned from "LoadCSV" method. It also provides file location to LoadCSV method, which is basically the text of text box.
- ❖ saveButton\_Click: This method will call "AddSampleDataForRatings" method to get all the ratings data including the new data imported from bulk file. Then, it will be written in ratings xml and schema files.
- ❖ AddSampleDataForRatings: First, this method will store data from ratings xml file in Ratings table of a dataset, then add new data in the that table from the data of DataGridView.

### 3.5.8 Columns

Columns class is used to provide column names in the DataGridView when bulk data are displayed for importing. The methods used in Columns class are:

- ❖ Service\_Name: This method will be the column of 0<sup>th</sup> index of DataGridView while displaying bulk file.
- ❖ Customer\_Name: This method will be the column of 1<sup>st</sup> index of DataGridView while displaying bulk file.
- ❖ Contact\_Number: This method will be the column of 2<sup>nd</sup> index of DataGridView while displaying bulk file.
- ❖ Email\_Address: This method will be the column of 3<sup>rd</sup> index of DataGridView while displaying bulk file.
- ❖ Address: This method will be the column of 4<sup>th</sup> index of DataGridView while displaying bulk file.

- ❖ Feedback: This method will be the column of 5<sup>th</sup> index of DataGridView while displaying bulk file.
- ❖ Date: This method will be the column of 6<sup>th</sup> index of DataGridView while displaying bulk file.
- ❖ Criteria: This method will be the column of 7<sup>th</sup> index of DataGridView while displaying bulk file.
- ❖ Criteria\_Value: This method will be the column of 8<sup>th</sup> index of DataGridView while displaying bulk file.

### 3.5.9 GraphOption

GraphOption class is responsible for displaying a form and requesting admin to choose either “Mean Rating” or “Overall Rating”. If “Mean Rating” is selected, bar graph will be displayed, and if “Overall Rating” is selected, pie chart will be displayed. The methods used in this class are:

- ❖ meanRatingButton\_Click: This method creates an object of BarGraph class to display mean rating’s bar graph.
- ❖ overallRatingsButton\_Click: This method creates an object of PieChart class to display overall rating’s pie chart.
- ❖ pictureBox1\_Click: This method closes the graph option form.

### 3.5.10 BarGraph

BarGraph is the class that displays a bar graph of each criteria’s mean rating. The only method used in this class is:

- ❖ DisplayBarGraph: This method will first read report data from report xml and schema files. Then, two ArrayList objects will be declared. One will store criteria’s name and another will store the mean rating’s value from report xml and schema files. Finally, bar graph of mean rating will be generated using the data of these two ArrayList objects.

### 3.5.11 PieChart

PieChart class is responsible for displaying pie chart of overall report of all the criteria. The only method used in this class is:

- ❖ PieChart: This method will first read report data from report xml and schema files. Then, two ArrayList objects will be declared. One will store criteria's name and another will store the overall rating's value from report xml and schema files. Finally, pie chart of overall rating will be generated using the data of these two ArrayLists objects.

### 3.5.12 LoginForm

LoginForm class displays the login page for admin to login into the admin dashboard. The global property used in this class is "thread", an object of Thread class. When "Customer Dashboard" button is clicked, admin login page will be closed and thread will start the customer rating and feedback dashboard. The methods used in this class are:

- ❖ loginButton\_Click: When correct username and password are entered, this method creates an object of AdminDashboard class to display admin dashboard. If incorrect login details are entered, message box will show exception messages.
- ❖ pictureBox1\_Click: This method will close the application.
- ❖ Button1\_Click\_1: This method will close the admin login page, and call "OpenCustomerUI" method.
- ❖ OpenCustomerUI: This method will display customer rating and feedback dashboard.

### 3.5.13 Program

Program class is the main class of the system. It is used to display customer rating and feedback dashboard. The only method used in this class is:

- ❖ Main: This method will display customer rating and feedback dashboard.

### 3.5.14 DataHandler

DataHandler class is used for creating Criteria, Ratings and Report tables. The methods used in this class are:

- ❖ CreateDataSet: This method returns a dataset which will contain Criteria, Ratings and Report tables.
- ❖ CreateCriteriaTable: This method returns “Criteria” data table.
- ❖ CreateRatingsTable: This method returns “Ratings” data table.
- ❖ CreateReportTable: This method returns “Report” data table.

#### 4. Flowchart and Algorithm of mean rating bar graph

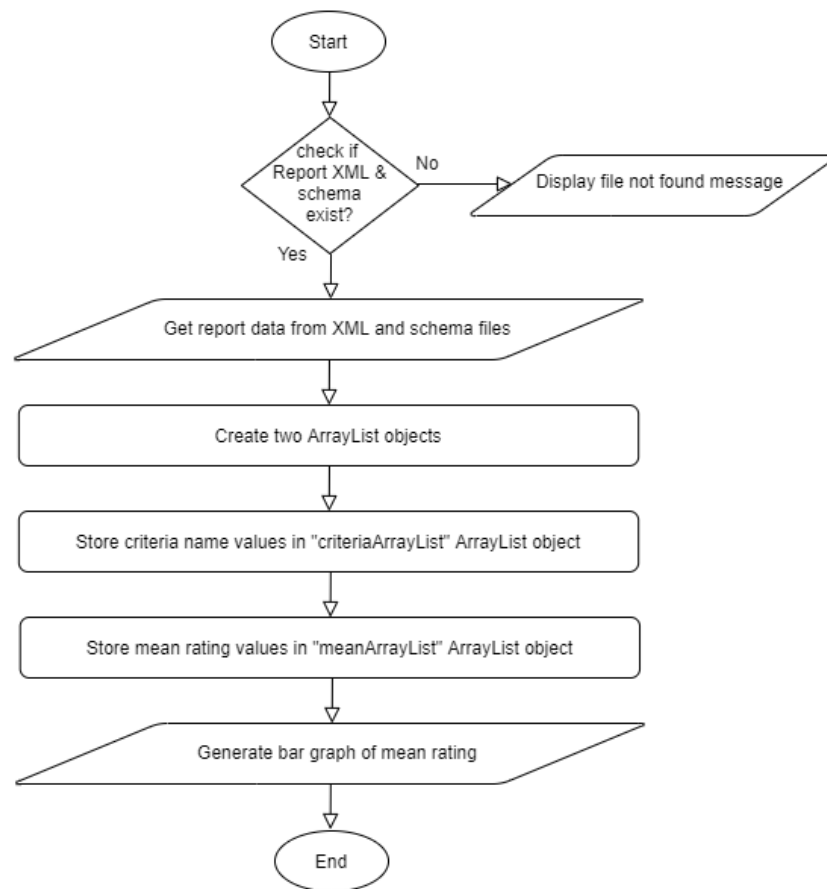


Figure 19: Flowchart of mean rating bar graph

Algorithm:

STEP 1: Start

STEP 2: Check if Report XML & schema file are available in bin folder.

STEP 3: If "Yes", Go to STEP 4. If "No", Display file not found message.

STEP 4: Get Report Data from XML and schema files

STEP 5: Create two ArrayList objects named as "criteriaArrayList" and "meanArrayList"

STEP 6: Save criteria names in "criteriaArrayList" ArrayList object

STEP 7: Save mean rating values in "meanArrayList" ArrayList object

STEP 8: Generate bar graph of each criteria's mean rating

STEP 9: End

## 5. Flowchart and Algorithm of overall rating pie chart

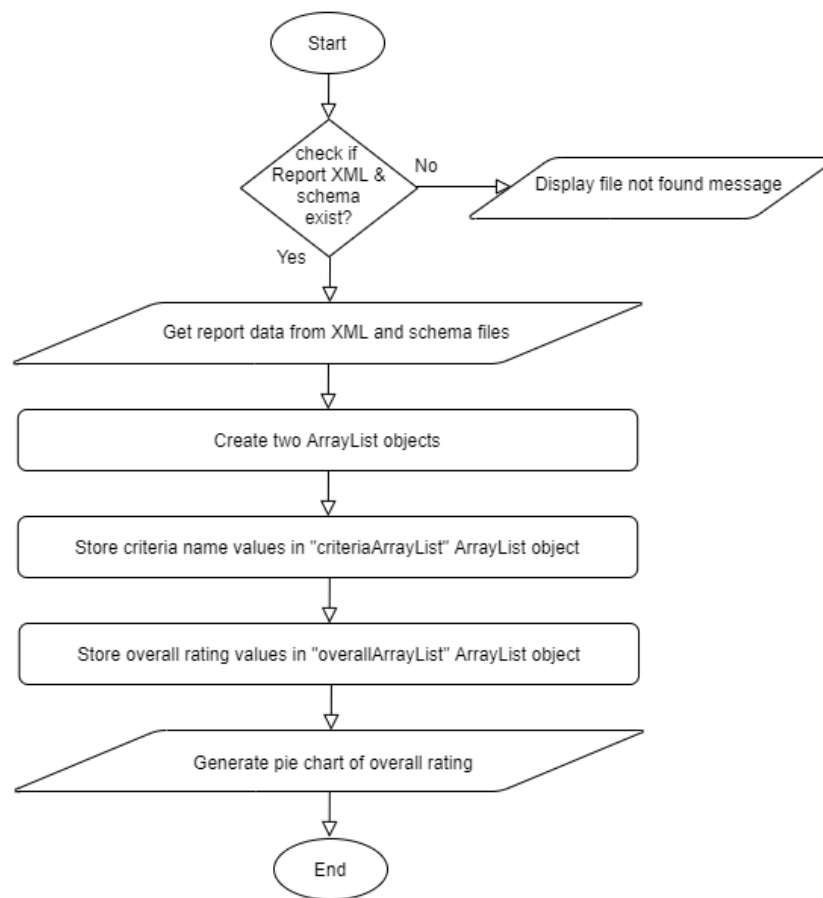


Figure 20: Flowchart of overall rating pie chart

Algorithm:

STEP 1: Start

STEP 2: Check if Report XML & schema file are available in bin folder.

STEP 3: If "Yes", Go to STEP 4. If "No", Display file not found message.

STEP 4: Get Report Data from XML and schema files

STEP 5: Create two ArrayList objects named as "criteriaArrayList" and "overallArrayList"

STEP 6: Save criteria names in "criteriaArrayList" ArrayList object

STEP 7: Save overall rating values in "overallArrayList" ArrayList object

STEP 8: Generate pie chart of each criteria's overall rating

STEP 9: End



## 6. Data structure

While developing the system, I have used List and ArrayList data structures. Regarding List data structure, it is used in Bulk\_Data\_Entry class to store the data of each column of the csv file for importing bulk data (geeksforgeeks, 2019). Similarly, I have used ArrayList data structure in BarGraph and PieChart class. In BarGraph class, first I have created two objects of ArrayList which are named as “criteriaArrayList” and “meanArrayList”. Then, Report data are imported from Report xml and schema files and stored in Report table of dataset. Then, criteria names from Report table are added in “criteriaArrayList” object, and mean rating values are added in “meanArrayList” object. After the report data are read from Report table, a bar graph of mean rating is generated using data of these two ArrayList objects.

Similarly, in PieChart class, the same process is applied as that of BarGraph class. Two ArrayList objects “criteriaArrayList” and “overallArrayList” are created where criteriaArrayList will store the names of criteria, however overallArrayList will store the data of overall ratings. Finally, pie chart of overall report is generated from these two ArrayList objects (Kashif, 2018; geeksforgeeks, 2019).

## 7. User Manual

Here is the user manual of Rating and Feedback System.

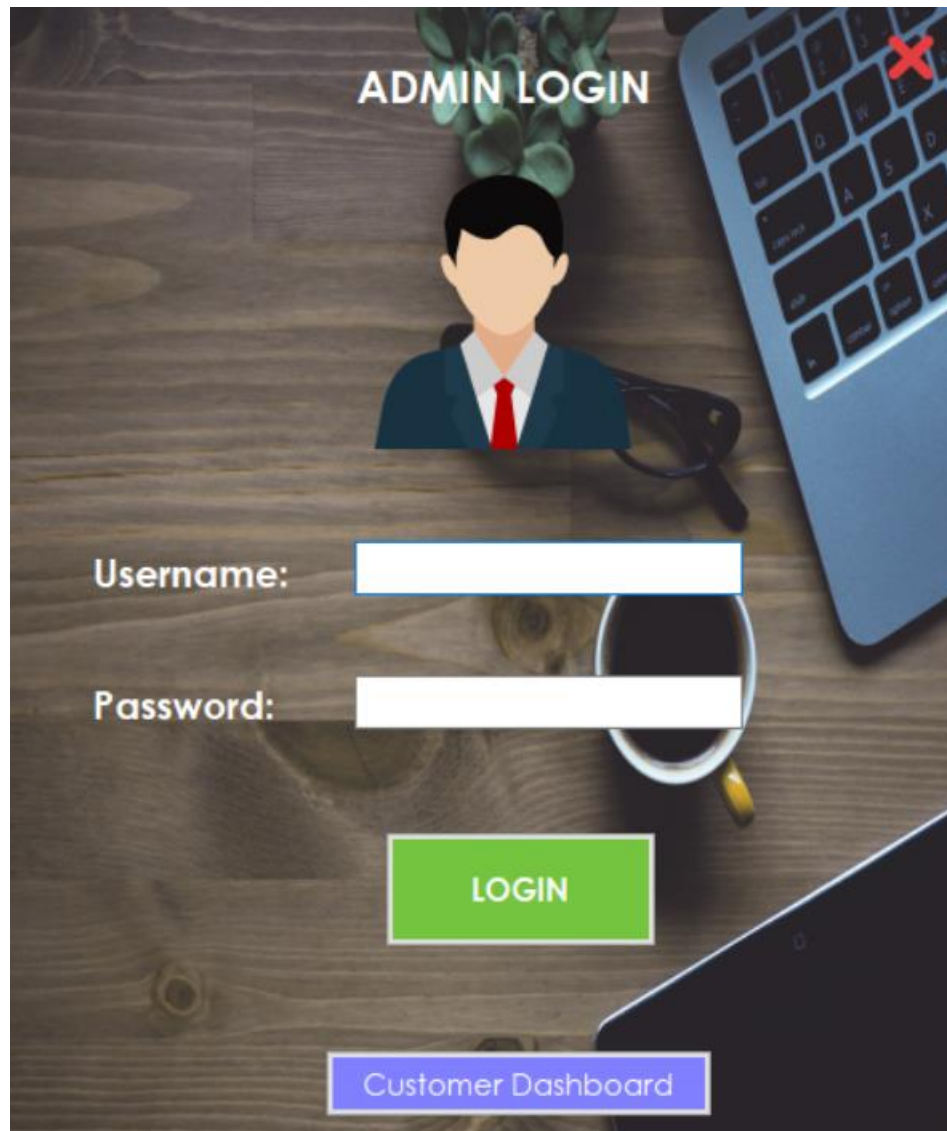
1) As we know, the system's main objective is rating and feedback system. So, when the system is started, Customer Rating and Feedback Dashboard will be the first page to be displayed as shown in the image below:

SN	Criteria	Description	Excellent	Good	Average	Dissatisfied
1	Food Order	Food Ordering criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Waiter Service	Waiter Service criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Food Quality	Food Quality criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	Greeting	Greeting criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	Reception	Reception criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	Hospitality	Hospitality criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7	Environment	Environment criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8	WiFi	WiFi criteria.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 21: Customer Rating and Feedback Dashboard

Here, Customer will have to select “Restaurant” service from the combo box at the top. It is not mandatory for customer to provide their name, contact number, email address, address, and feedback. Hence, these fields can be left empty. However, customer have to give rating to each criteria by clicking on the checkbox. When “SUBMIT” button is clicked, rating will be completed.

2) There is a “Login as Admin” button at the bottom right of the Customer Rating and Feedback Dashboard. When it is clicked, admin login page will be displayed as shown in the image below:



*Figure 22: Admin Login*

On this login page, admin will have to enter their correct login details. Correct username is “Kushal” and password is “Gurung”. After admin enters correct login details, he can login into the admin dashboard by clicking on “LOGIN” button. However, if admin wants to go back to previous Customer Rating and Feedback dashboard, he/she can click on “Customer Dashboard” button.

**3)** After “LOGIN” button is clicked, Admin Dashboard will be displayed as shown in the image below. Admin dashboard is the main UI for the system’s admin.

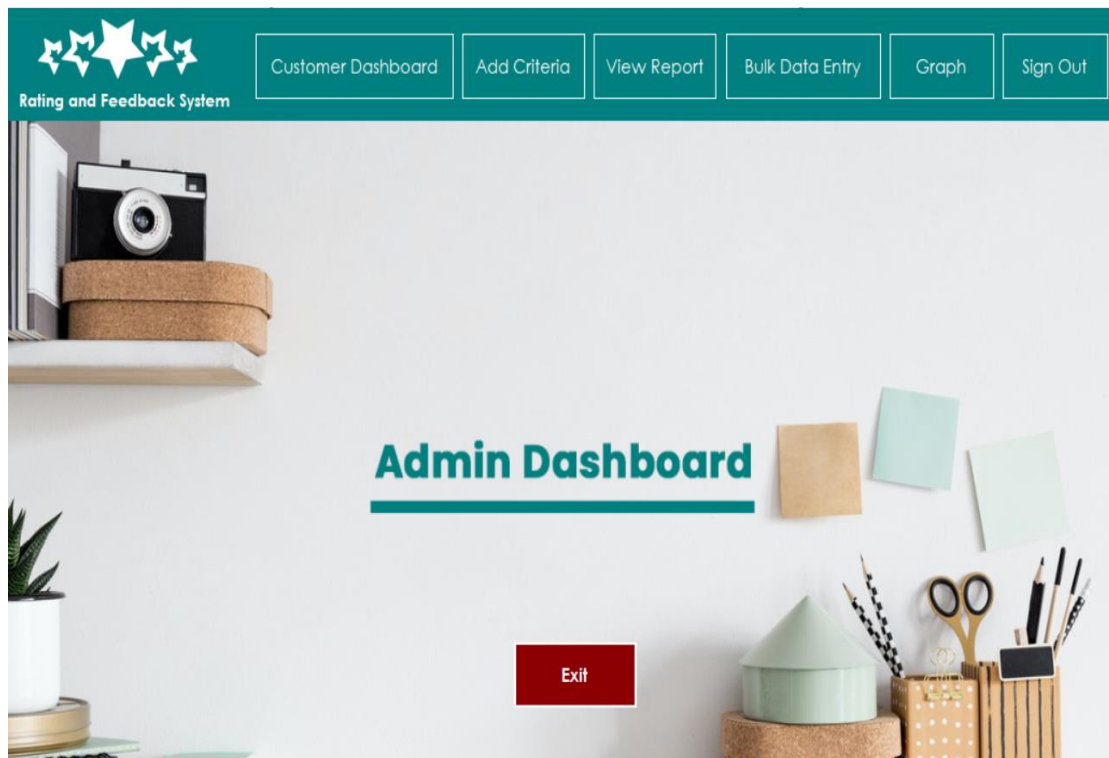


Figure 23: Admin Dashboard

4) When “Customer Dashboard” button is clicked in admin dashboard, admin will be taken to Customer Rating and Feedback Dashboard. The image of customer dashboard is already shown above in user manual number: 1).

5) When “Add Criteria” button is clicked in admin dashboard, a criteria addition form will be displayed as shown in the image below:

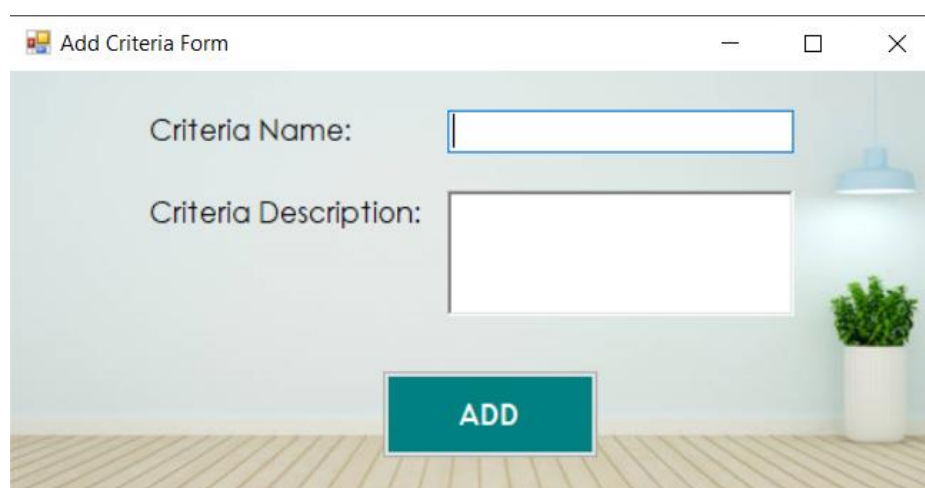
The image shows a window titled 'Add Criteria Form'. It contains two input fields: 'Criteria Name:' with a single-line text box, and 'Criteria Description:' with a multi-line text box. At the bottom center, there is a teal button labeled 'ADD'. The background of the form is a light blue wall with a potted plant and a lamp.

Figure 24: Add Criteria Form

Here, admin will have to enter the name of new criteria along with its description, then click on “ADD” button.

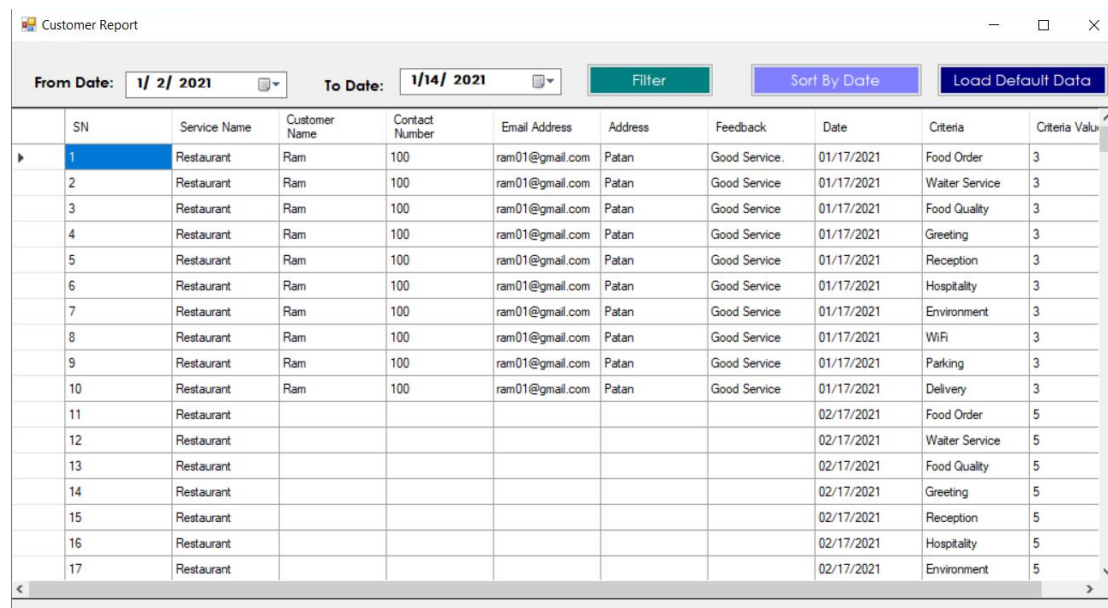
6) When “View Report” button is clicked in admin dashboard, a form will open and ask the admin to choose either customer report or overall report as shown in the image below:



A teal-colored modal window titled "Choose One:" with a close button in the top right corner. It contains two options: "1. Customer Report" and "2. Overall Report", each enclosed in a light blue rectangular button.

Figure 25: Report Option Form

6.1) If admin clicks on “Customer Report” button, the following form will open:



A web application window titled "Customer Report" with a table of customer feedback data. The table has columns for SN, Service Name, Customer Name, Contact Number, Email Address, Address, Feedback, Date, Criteria, and Criteria Value. The first 10 rows show data for a restaurant named "Ram" with a contact number of 100, all with a feedback of "Good Service" and a criteria value of 3. The next 7 rows show data for a restaurant with an empty name and contact number, with feedback of "Good Service" and criteria values of 5.

SN	Service Name	Customer Name	Contact Number	Email Address	Address	Feedback	Date	Criteria	Criteria Value
1	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service.	01/17/2021	Food Order	3
2	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Waiter Service	3
3	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Food Quality	3
4	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Greeting	3
5	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Reception	3
6	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Hospitality	3
7	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Environment	3
8	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	WiFi	3
9	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Parking	3
10	Restaurant	Ram	100	ram01@gmail.com	Patan	Good Service	01/17/2021	Delivery	3
11	Restaurant						02/17/2021	Food Order	5
12	Restaurant						02/17/2021	Waiter Service	5
13	Restaurant						02/17/2021	Food Quality	5
14	Restaurant						02/17/2021	Greeting	5
15	Restaurant						02/17/2021	Reception	5
16	Restaurant						02/17/2021	Hospitality	5
17	Restaurant						02/17/2021	Environment	5

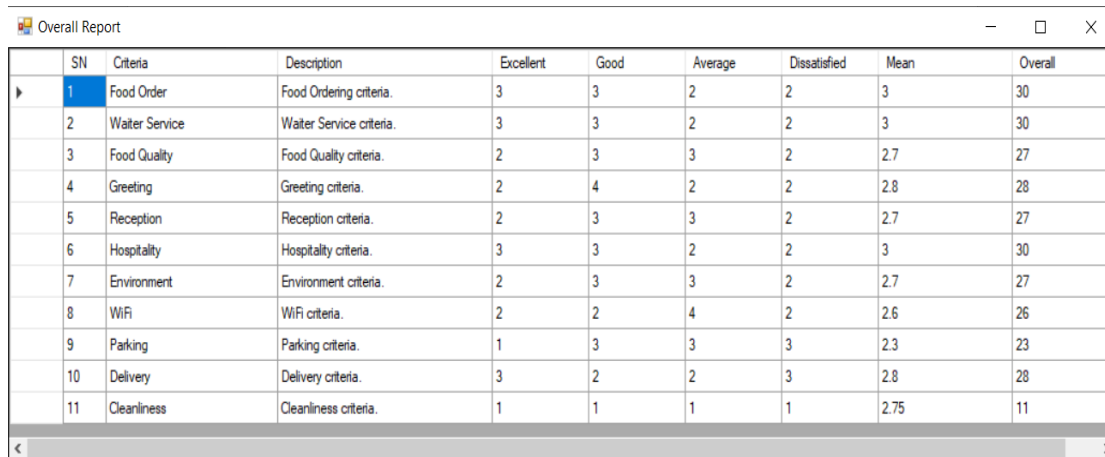
Figure 26: Customer Report

There are three buttons in this form:

- ❖ Admin can view reports of specific dates by selecting the start and end dates and then clicking on “Filter” button.
- ❖ Admin can sort the data by dates in ascending order by clicking on “Sort By Date” button.

- ❖ When we click on “Filter” or “Sort By Date” button, the data in DataGridView will be changed. So, if admin wants to view the original default data, he/she can click on “Load Default Data” button.

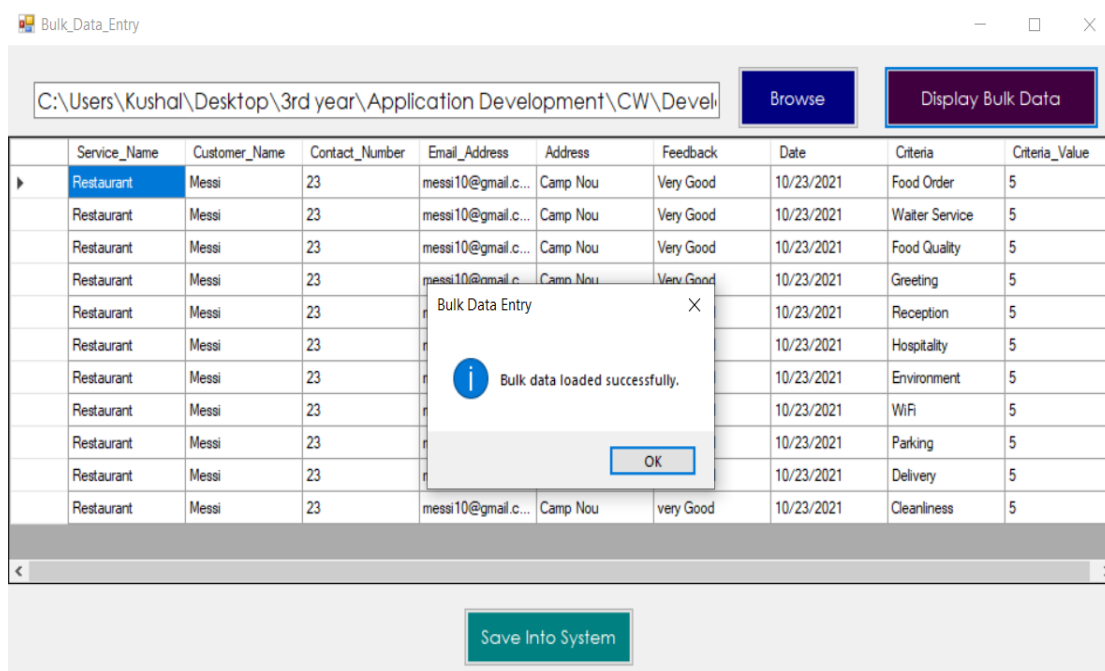
**6.2)** If admin clicks on “Overall Report” button, the following form will open:



	SN	Criteria	Description	Excellent	Good	Average	Dissatisfied	Mean	Overall
▶	1	Food Order	Food Ordering criteria.	3	3	2	2	3	30
	2	Waiter Service	Waiter Service criteria.	3	3	2	2	3	30
	3	Food Quality	Food Quality criteria.	2	3	3	2	2.7	27
	4	Greeting	Greeting criteria.	2	4	2	2	2.8	28
	5	Reception	Reception criteria.	2	3	3	2	2.7	27
	6	Hospitality	Hospitality criteria.	3	3	2	2	3	30
	7	Environment	Environment criteria.	2	3	3	2	2.7	27
	8	WiFi	WiFi criteria.	2	2	4	2	2.6	26
	9	Parking	Parking criteria.	1	3	3	3	2.3	23
	10	Delivery	Delivery criteria.	3	2	2	3	2.8	28
	11	Cleanliness	Cleanliness criteria.	1	1	1	1	2.75	11

*Figure 27: Overall Report*

**7)** When “Bulk Data Entry” button is clicked in admin dashboard, the following UI will be displayed. A sample CSV file named “Bulk\_Data” is kept in bin folder of the project for bulk data entry.



Bulk\_Data\_Entry

C:\Users\Kushal\Desktop\3rd year\Application Development\CW\Devel... Browse Display Bulk Data

	Service_Name	Customer_Name	Contact_Number	Email_Address	Address	Feedback	Date	Criteria	Criteria_Value
▶	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Food Order	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Waiter Service	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Food Quality	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Greeting	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Reception	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Hospitality	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Environment	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	WiFi	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Parking	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	Very Good	10/23/2021	Delivery	5
	Restaurant	Messi	23	messi10@gmail.c...	Camp Nou	very Good	10/23/2021	Cleanliness	5

Save Into System

Bulk Data Entry

Bulk data loaded successfully.

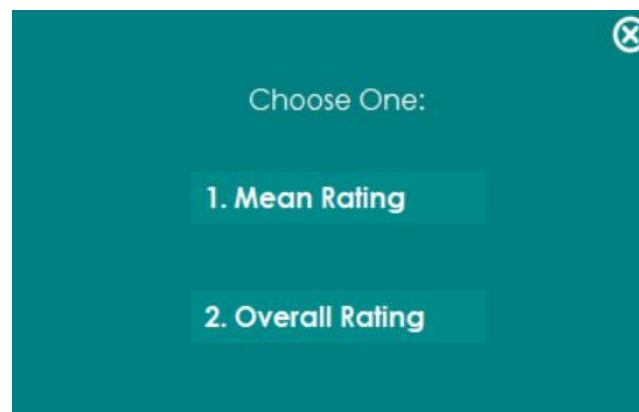
OK

*Figure 28: Bulk Data Entry UI*

Here, admin have to select a bulk csv file by clicking on “Browse” button. After correct file is selected, admin should click on “Display Bulk Data” to confirm if

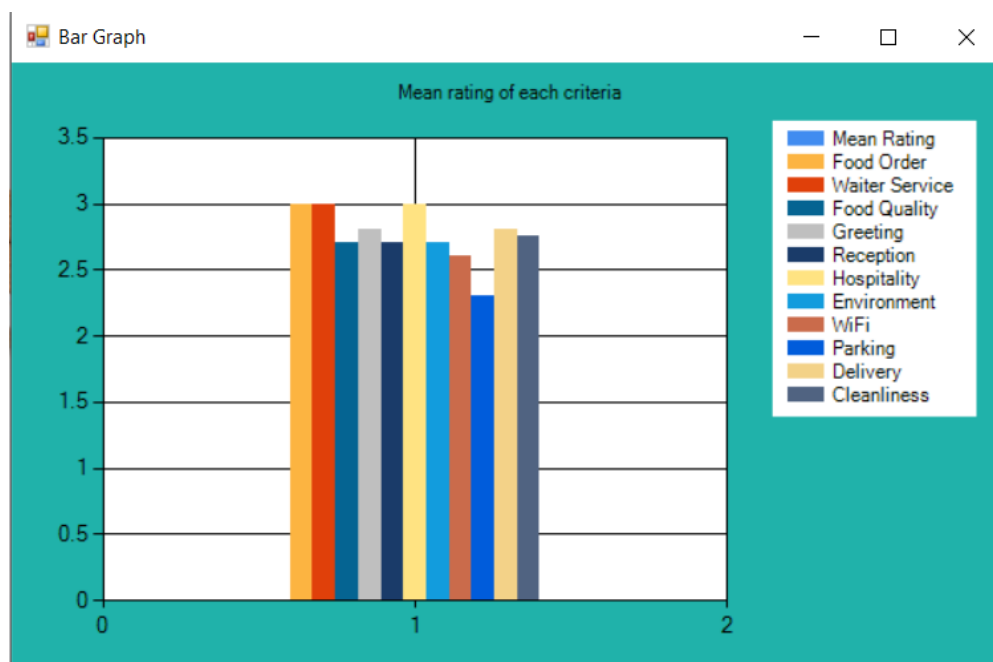
the bulk data are imported or not. If data are imported successfully, the data will be displayed in the DataGridView, else a message box will pop up and request the admin to select the correct file. After the bulk data are displayed in DataGridView, admin can save the bulk data by clicking on “Save Into System” button.

**8)** When “Graph” button is clicked in admin dashboard, a form will open and ask the admin to choose either mean rating or overall rating as shown in the image below:



*Figure 29: Graph Option Form*

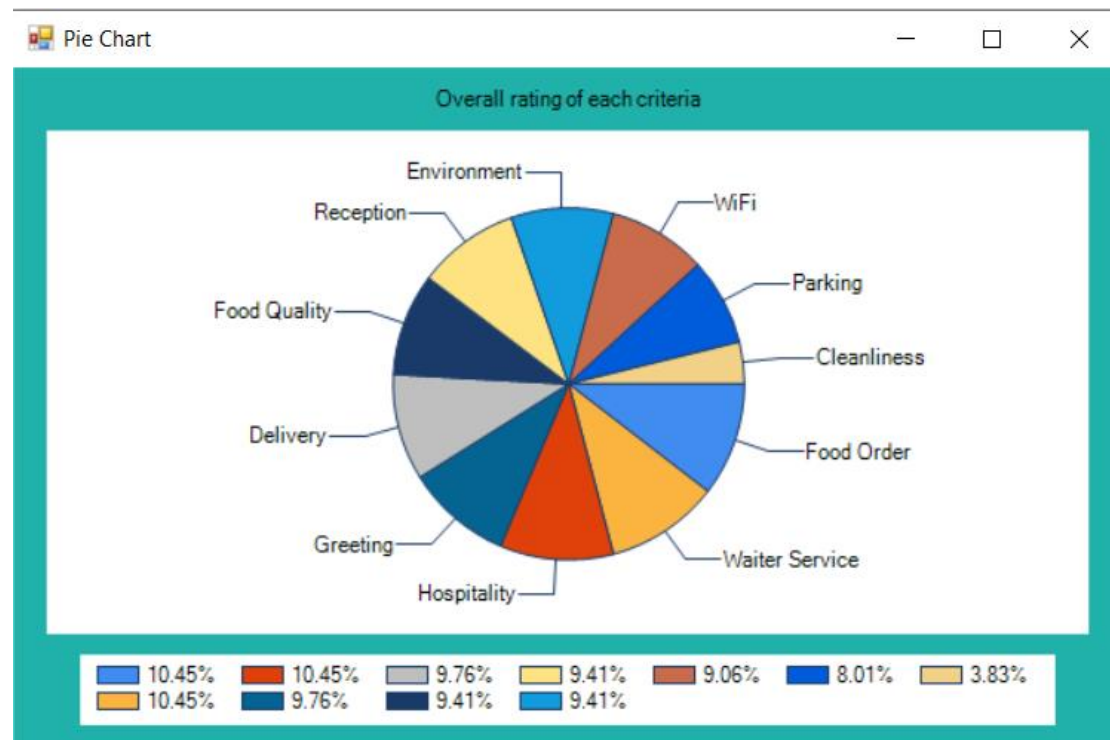
**8.1)** If admin clicks on “Mean Rating” button, the mean rating bar graph will be displayed as shown in the image below:



*Figure 30: Bar Graph of mean rating*



**8.2)** If admin clicks on “Overall Rating” button, the overall rating pie chart will be displayed as shown in the image below:



*Figure 31: Pie Chart of overall report*

**9)** If “Sign Out” button is clicked in admin dashboard, admin will be guided to Admin Login UI. The image of Admin Login UI is already shown above in user manual number: **2)**.

**10)** Finally, when “Exit” button is clicked in admin dashboard, the system will be closed.



## 8. Conclusion

Finally, the first coursework of Application Development module is completed. It was a great experience and I have gained lots of C# programming knowledge while developing Rating and Feedback System. While developing the system, I have learned how to combine different classes together and share data between them. Similarly, I have learned how to create XML and XML Schema files programmatically and store the data of the system in it. Adding on, I learned how to read the data of XML and XML Schema files, and display them on the DataGridView of the system. Likewise, I have learned how to import bulk data from a CSV file, generate bar graph, pie chart, filter customer report according to start and end dates, add new criteria into the system, and so on.

Similarly, I have become much more familiar with the tools of Visual Studio IDE such as: labels, textbox, checkbox, button, datetimepicker, DataGridView, pictured button, etc. I have also learned how to customize tools in properties option. The feature that I liked about Visual Studio is the toggle breakpoint. I found it very important throughout the development course of the project. While importing data from XML and XML Schema files, it was very necessary to check whether data were being retrieved from the xml files. Similarly, it was very helpful to display correct values in the DataGridView. Moreover, while writing the values of DataGridView into the xml files, it was very important to use toggle breakpoint to make sure that non-empty and correct data were being written. Hence, these are reasons why I liked toggle breakpoint feature of Visual Studio IDE.

During the development of the system, I faced various problems in different parts of the project. Some of them were very tough to sort out. I faced problem while giving values to Excellent, Good, Average, and Dissatisfied ratings from checkbox of the DataGridView in customer rating and feedback dashboard. Similarly, filtering customer report from start and end dates, importing and saving bulk data files, reading rating data and overwriting new rating data in it were also the issues that I had faced during the development of the system. To overcome these issues, I did lots of research on the parts where I was getting error, and I also used toggle breakpoint feature to check the data flow. Also, review of lecture slides, discussions with friends and suggestions from module

leader played an important role to overcome these issues and develop customer rating and feedback system with all the requirements of the coursework.

## 9. Bibliography

Chand, M., 2017. *c-sharpcorner*. [Online]  
Available at: <https://www.c-sharpcorner.com/article/reading-and-writing-xml-in-C-Sharp/>

[Accessed 27 December 2020].

geeksforgeeks, 2019. *geeksforgeeks*. [Online]  
Available at: <https://www.geeksforgeeks.org/arraylist-in-c-sharp/>  
[Accessed 06 January 2021].

geeksforgeeks, 2019. *geeksforgeeks*. [Online]  
Available at: <https://www.geeksforgeeks.org/list-implementation-in-c-sharp/>  
[Accessed 04 January 2021].

Kashif, M., 2018. *linkedin*. [Online]  
Available at: <https://www.linkedin.com/pulse/c-data-structure-array-arraylist-list-muhammad-kashif>  
[Accessed 05 January 2021].

Nayak, G., 2012. *c-sharpcorner*. [Online]  
Available at: <https://www.c-sharpcorner.com/uploadfile/9f4ff8/draw-a-pie-chart-in-C-Sharp/>  
[Accessed 12 January 2021].

thedeveloperblog, n.d. *thedeveloperblog*. [Online]  
Available at: <https://thedeveloperblog.com/chart>  
[Accessed 11 January 2021].

## 10. Appendix

### 1. AdminDashboard.cs

```

using System;
using System.Windows.Forms;
using System.Threading;

namespace Development
{
    public partial class AdminDashboard : Form
    {
        Thread thread;
        public AdminDashboard()
        {
            InitializeComponent();

            // Add Criteria button
            private void addCriterionBtn_Click(object sender, EventArgs e)
            {
                CriteriaAdd criteriaAdd = new CriteriaAdd();
                criteriaAdd.Show();
            }

            // View Report button
            private void viewReportBtn_Click(object sender, EventArgs e)
            {
                ReportOption reportOption = new ReportOption();
                reportOption.Show();
            }

            // Graph button
            private void graphButton_Click(object sender, EventArgs e)
            {
                GraphOption graphOption = new GraphOption();
                graphOption.Show();
            }

            // Exit button
            private void exitButton_Click(object sender, EventArgs e)
            {
                Application.Exit();
            }

            // Sign Out button
            private void signOutButton_Click(object sender, EventArgs e)
            {
                MessageBox.Show("You are successfully signed out.", "Rating and
Feedback System", MessageBoxButtons.OK, MessageBoxIcon.Information);

                // Closing the current running form, and calling OpenLoginForm
method.
                this.Close();
                thread = new Thread(OpenLoginForm);
                thread.SetApartmentState(ApartmentState.STA);
                thread.Start();
            }

            // Opens Admin Login UI
            public static void OpenLoginForm()

```

```

    {
        // It will run the LoginForm.
        Application.Run(new LoginForm());
    }

    // Opens Bulk Data Entry form
    private void bulkDataEntryBtn_Click(object sender, EventArgs e)
    {
        Bulk_Data_Entry bulk_Data_Entry = new Bulk_Data_Entry();
        bulk_Data_Entry.Show();
    }

    // Closes the current running form, and calls OpenCustomerUI method.
    private void customerDashboardButton_Click(object sender, EventArgs e)
    {
        this.Close();
        thread = new Thread(OpenCustomerUI);
        thread.SetApartmentState(ApartmentState.STA);
        thread.Start();
    }

    // Opens Customer Rating and Feedback Dashboard
    public static void OpenCustomerUI()
    {
        Application.Run(new CustomerUI());
    }

    private void panel1_Paint(object sender, PaintEventArgs e)
    {
    }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {
    }

    private void label1_Click(object sender, EventArgs e)
    {
    }
}
}

```

## 2. BarGraph.cs

```

using System;
using System.Data;
using System.Windows.Forms;
using System.Collections;
using System.Windows.Forms.DataVisualization.Charting;

namespace Development
{
    public partial class BarGraph : Form
    {
        public BarGraph()
        {
            InitializeComponent();
            DisplayBarGraph();
        }
    }
}

```

```

    }

    public void DisplayBarGraph()
    {
        try
        {
            ArrayList criteriaArrayList = new ArrayList();
            ArrayList meanArrayList = new ArrayList();

            //reading data from xml and schema files
            DataSet dsRead = new DataSet();
            dsRead.ReadXmlSchema(@"../ReportSchema.xml");
            dsRead.ReadXml(@"../ReportData.xml");

            if (dsRead.Tables["Report"] != null)
            {
                for (int i = 0; i < dsRead.Tables["Report"].Rows.Count;
i++)
                {
                    // Criteria Name

                    criteriaArrayList.Add(dsRead.Tables["Report"].Rows[i][1].ToString());

                    // Mean Rating

                    meanArrayList.Add(dsRead.Tables["Report"].Rows[i][7].ToString());
                }

                // Setting title of the bar graph
                this.chart2.Titles.Add("Mean rating of each criteria");

                // Setting Color Palette for the bar graph
                this.chart2.Palette = ChartColorPalette.BrightPastel;

                // Add Series
                for (int i = 0; i < dsRead.Tables["Report"].Rows.Count; i++)
                {
                    // Series represents the data points and series attributes
                    to store
                    // Adding series attributes
                    Series series =
                    this.chart2.Series.Add(criteriaArrayList[i].ToString());

                    // Adding data point
                    series.Points.Add(Convert.ToDouble(meanArrayList[i]));
                }
            }

            catch (System.IO.FileNotFoundException ex)
            {
                Console.WriteLine(ex);
                MessageBox.Show("The report file for mean rating cannot be
found in the saved directory. Please verify it once.", "Error!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void BarGraph_Load(object sender, EventArgs e)
        {

```

```

    }

    private void chart2_Click(object sender, EventArgs e)
    {
    }
}

```

### 3. Bulk\_Data\_Entry.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows.Forms;
using System.IO;
using Handler;

namespace Development
{
    public partial class Bulk_Data_Entry : Form
    {
        public Bulk_Data_Entry()
        {
            InitializeComponent();
        }

        private void loadDataButton_Click(object sender, EventArgs e)
        {
            if (fileLocationTextBox.Text == "")
            {
                MessageBox.Show("Please select a bulk file.", "Bulk Data Entry", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                try
                {
                    // file location will be passed into LoadCSV method to get data of csv file for displaying them in data grid view
                    bulkEntryDataGridView.DataSource = LoadCSV(fileLocationTextBox.Text);
                    MessageBox.Show("Bulk data loaded successfully.", "Bulk Data Entry", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception ex)
                {
                    Console.WriteLine(ex);
                    MessageBox.Show("Please select the correct bulk file.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }

        // returns list of data of csv file
        public List<Columns> LoadCSV(string csvFile)
        {
            var query = from l in File.ReadAllLines(csvFile)

```

```

        let data = l.Split(',')
        select new Columns
        {
            Service_Name = data[0],
            Customer_Name = data[1],
            Contact_Number = data[2],
            Email_Address = data[3],
            Address = data[4],
            Feedback = data[5],
            Date = data[6],
            Criteria = data[7],
            Criteria_Value = data[8],
        };
    }

    return query.ToList();
}

private void browseButton_Click(object sender, EventArgs e)
{
    OpenFileDialog dialog = new OpenFileDialog();
    dialog.ShowDialog();
    fileLocationTextBox.Text = dialog.FileName;
}

private void saveButton_Click(object sender, EventArgs e)
{
    try
    {
        if (fileLocationTextBox.Text == "")
        {
            MessageBox.Show("Please select a bulk file.", "Bulk Data Entry", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        else if (fileLocationTextBox.Text != "" && bulkEntryDataGridView.DataSource == null)
        {
            MessageBox.Show("Please display and check the bulk data before saving.", "Bulk Data Entry", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        else if (bulkEntryDataGridView.DataSource == null)
        {
            MessageBox.Show("Please select the correct bulk file.", "Bulk Data Entry", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        else
        {
            DataHandler dh = new DataHandler();
            DataSet ds = new DataSet();
            ds = dh.CreateDataSet();

            AddSampleDataForRatings(ds);
            ds.WriteXmlSchema(@"../RatingsSchema.xml");
            ds.WriteXml(@"../RatingsData.xml");
            MessageBox.Show("Bulk data are successfully saved into the system.", "Bulk Data Entry", MessageBoxButtons.OK, MessageBoxIcon.Information);
            fileLocationTextBox.Text = "";
            bulkEntryDataGridView.DataSource = null;
        }
    }
}

```



```

    }
}

catch (Exception ex)
{
    Console.WriteLine(ex);
    MessageBox.Show("Please select the correct file.", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}

}

void AddSampleDataForRatings(DataSet dataSet)
{
    try
    {
        DataSet dsRead = new DataSet();
        dsRead.ReadXmlSchema(@"../RatingsSchema.xml");
        dsRead.ReadXml(@"../RatingsData.xml");

        DataRow dr;

        // If the data in Ratings table is not null, then only it's
        statement will be executed.
        if (dsRead.Tables["Ratings"] != null)
        {
            // All the data from previous xml file will be added in the
            new xml file.
            for (int i = 0; i < dsRead.Tables["Ratings"].Rows.Count;
i++)
            {
                dr = dataSet.Tables["Ratings"].NewRow();

                dr["Service Name"] =
dsRead.Tables["Ratings"].Rows[i]["Service Name"].ToString();
                dr["Customer Name"] =
dsRead.Tables["Ratings"].Rows[i]["Customer Name"].ToString();
                dr["Contact Number"] =
dsRead.Tables["Ratings"].Rows[i]["Contact Number"].ToString();
                dr["Email Address"] =
dsRead.Tables["Ratings"].Rows[i]["Email Address"].ToString();
                dr["Address"] =
dsRead.Tables["Ratings"].Rows[i]["Address"].ToString();
                dr["Feedback"] =
dsRead.Tables["Ratings"].Rows[i]["Feedback"].ToString();
                dr["Date"] =
dsRead.Tables["Ratings"].Rows[i]["Date"].ToString();

                dr["Criteria"] =
dsRead.Tables["Ratings"].Rows[i]["Criteria"].ToString();
                dr["Criteria Value"] =
dsRead.Tables["Ratings"].Rows[i]["Criteria Value"].ToString();
                dataSet.Tables["Ratings"].Rows.Add(dr);
            }
        }

        // All the previous data are read. Now, this for loop will add
        all the new data of bulkEntryDataGridView into the Rating table.
        for (int i = 0; i < bulkEntryDataGridView.Rows.Count; i++)
        {
            dr = dataSet.Tables["Ratings"].NewRow();

```

```

        dr["Service Name"] =
bulkEntryDataGridView.Rows[i].Cells[0].Value.ToString();
        dr["Customer Name"] =
bulkEntryDataGridView.Rows[i].Cells[1].Value.ToString();
        dr["Contact Number"] =
bulkEntryDataGridView.Rows[i].Cells[2].Value.ToString();
        dr["Email Address"] =
bulkEntryDataGridView.Rows[i].Cells[3].Value.ToString();
        dr["Address"] =
bulkEntryDataGridView.Rows[i].Cells[4].Value.ToString();
        dr["Feedback"] =
bulkEntryDataGridView.Rows[i].Cells[5].Value.ToString();
        dr["Date"] =
bulkEntryDataGridView.Rows[i].Cells[6].Value.ToString();

        dr["Criteria"] =
bulkEntryDataGridView.Rows[i].Cells[7].Value.ToString();
        dr["Criteria Value"] =
bulkEntryDataGridView.Rows[i].Cells[8].Value.ToString();
        dataSet.Tables["Ratings"].Rows.Add(dr);
    }
}

catch (System.IO.FileNotFoundException ex)
{
    Console.WriteLine(ex);
    MessageBox.Show("The ratings file cannot be found in the saved
directory. Please verify it once.", "Error!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}

}

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

private void bulkEntryDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
}

}

public class Columns
{
    public string Service_Name { get; set; }

    public string Customer_Name { get; set; }

    public string Contact_Number { get; set; }

    public string Email_Address { get; set; }

    public string Address { get; set; }

    public string Feedback { get; set; }

    public string Date { get; set; }
}

```

```

        public string Criteria { get; set; }

        public string Criteria_Value { get; set; }
    }
}

```

#### 4. CriteriaAdd.cs

```

using Handler;
using System;
using System.Data;
using System.Windows.Forms;

namespace Development
{
    public partial class CriteriaAdd : Form
    {
        public CriteriaAdd()
        {
            InitializeComponent();
        }

        void SaveDataToFile()
        {
            DataHandler dh = new DataHandler();

            DataSet ds = new DataSet();

            ds = dh.CreateDataSet();
            AddSampleData(ds);
            ds.WriteXml(@"../CriteriaData.xml");
            ds.WriteXmlSchema(@"../CriteriaSchema.xml");
        }

        private void AddSampleData(DataSet dataSet)
        {
            try
            {
                DataSet dsRead = new DataSet();
                dsRead.ReadXml(@"../CriteriaData.xml");
                dsRead.ReadXmlSchema(@"../CriteriaSchema.xml");

                DataRow dr;

                // In for loop, rows will be added from our XML file.
                if (dsRead.Tables["Criteria"] != null)
                {
                    for (int i = 0; i < dsRead.Tables["Criteria"].Rows.Count;
i++)
                    {
                        dr = dataSet.Tables["Criteria"].NewRow();
                        dr["Criteria Name"] =
dsRead.Tables["Criteria"].Rows[i]["Criteria Name"].ToString();
                        dr["Description"] =
dsRead.Tables["Criteria"].Rows[i]["Description"].ToString();

                        dataSet.Tables["Criteria"].Rows.Add(dr);
                    }
                }
            }
        }
    }
}

```

```

        //Here we will add rows based on what user enters.
        string criteriaName = "";
        string criteriaDescription = "";

        criteriaName = criteriaNameTextBox.Text;
        criteriaDescription = descriptionTextBox.Text;

        var dr1 = dataSet.Tables["Criteria"].NewRow();
        dr1["Criteria Name"] = criteriaName;
        dr1["Description"] = criteriaDescription;

        dataSet.Tables["Criteria"].Rows.Add(dr1);
    }

    catch (System.IO.FileNotFoundException ex)
    {
        Console.WriteLine(ex);
        MessageBox.Show("The criteria file cannot be found in the saved
directory. Please verify it once.", "Error!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

private void criteriaAddButton_Click_1(object sender, EventArgs e)
{
    if (criteriaNameTextBox.Text == "" )
    {
        MessageBox.Show("Please provide criteria's name.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    else if (descriptionTextBox.Text == "")
    {
        MessageBox.Show("Please provide criteria's description.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    else
    {
        SaveDataToFile();
        MessageBox.Show("New criteria added successfully.", "Criteria
Addition", MessageBoxButtons.OK, MessageBoxIcon.Information);
        criteriaNameTextBox.Text = "";
        descriptionTextBox.Text = "";
    }
}

private void label1_Click(object sender, EventArgs e)
{
}

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

```

## 5. CustomerReport.cs

```
using System;
using System.Data;
using System.Windows.Forms;
using System.ComponentModel;

namespace Development
{
    public partial class CustomerReport : Form
    {
        public CustomerReport()
        {
            InitializeComponent();

            DataView dataView;
            DataTable dataTable;

            public void GetCustomerReport()
            {
                try
                {
                    // Adding columns in data table
                    dataTable = new DataTable();
                    dataTable.Columns.Add("SN");
                    dataTable.Columns.Add("Service Name");
                    dataTable.Columns.Add("Customer Name");
                    dataTable.Columns.Add("Contact Number");
                    dataTable.Columns.Add("Email Address");
                    dataTable.Columns.Add("Address");
                    dataTable.Columns.Add("Feedback");
                    dataTable.Columns.Add("Date");
                    dataTable.Columns.Add("Criteria");
                    dataTable.Columns.Add("Criteria Value");

                    DataSet dsRead = new DataSet();
                    dsRead.ReadXmlSchema(@"../RatingsSchema.xml");
                    dsRead.ReadXml(@"../RatingsData.xml");

                    for (int i = 0; i < dsRead.Tables["Ratings"].Rows.Count; i++)
                    {
                        // Adding value in data table
                        dataTable.Rows.Add(
                            i + 1,
                            dsRead.Tables["Ratings"].Rows[i][1].ToString(),
                            dsRead.Tables["Ratings"].Rows[i][2].ToString(),
                            dsRead.Tables["Ratings"].Rows[i][3].ToString(),
                            dsRead.Tables["Ratings"].Rows[i][4].ToString(),
                            dsRead.Tables["Ratings"].Rows[i][5].ToString(),
                            dsRead.Tables["Ratings"].Rows[i][6].ToString(),
                            dsRead.Tables["Ratings"].Rows[i][7].ToString(),
                            dsRead.Tables["Ratings"].Rows[i][8].ToString(),
                            dsRead.Tables["Ratings"].Rows[i][9].ToString());
                    }

                    dataView = new DataView(dataTable);

                    // Displaying data in data grid view
                    customerReportTable.DataSource = dataView;
                }
            }
        }
    }
}
```

```

        catch (System.IO.FileNotFoundException ex)
        {
            Console.WriteLine(ex);
            MessageBox.Show("The ratings file cannot be found in the saved
directory. Please verify it once.", "Error!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void CustomerReport_Load(object sender, EventArgs e)
    {
        GetCustomerReport();
    }

    private void filterButton_Click(object sender, EventArgs e)
    {
        DateTime startDate = start_dateTimePicker.Value.Date;
        DateTime endDate = end_dateTimePicker.Value.Date;
        dataView.RowFilter = String.Format("Date >= #{0:MM/dd/yyyy}#
AND Date <= #{1:MM/dd/yyyy}#", startDate, endDate);
        customerReportTable.DataSource = dataView;

        if (customerReportTable.Rows.Count == 1)
        {
            MessageBox.Show("No customer report available from date: " +
startDate.ToString("MM/dd/yyyy") + " to " + endDate.ToString("MM/dd/yyyy") +
".", "Customer Report", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show("Here are the customer report from date: " +
startDate.ToString("MM/dd/yyyy") + " to " + endDate.ToString("MM/dd/yyyy") +
".", "Customer Report", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }

    private void sortByDateButton_Click(object sender, EventArgs e)
    {
        if (customerReportTable.Rows.Count == 1)
        {
            MessageBox.Show("There are no data to sort.", "Data Sorting",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            this.customerReportTable.Sort(customerReportTable.Columns["Date"],
ListSortDirection.Ascending);
            MessageBox.Show("Reports are successfully sorted according to
date.", "Customer Report", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }

    private void loadDefaultDataButton_Click(object sender, EventArgs e)
    {
        GetCustomerReport();
        MessageBox.Show("Default data are loaded successfully.", "Customer
Report", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```

```

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void customerReportTable_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }
    }
}

```

## 6. CustomerUI.cs

```

using System;
using System.Data;
using System.Windows.Forms;
using Handler;
using System.Threading;

namespace Development
{
    public partial class CustomerUI : Form
    {
        Thread thread;
        public CustomerUI()
        {
            InitializeComponent();
            getCriteriaData();
        }

        //When one checkbox is selected in a row, another will be unselected
        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            DataGridView gridView = (DataGridView)sender;

            for (int i = 3; i<=6; i++)
            {
                DataGridViewCheckBoxCell checkBoxCell =
(DataGridViewCheckBoxCell)gridView.Rows[e.RowIndex].Cells[i];
                if( i != e.ColumnIndex)
                {
                    checkBoxCell.Value = checkBoxCell.FalseValue;
                }
            }
        }

        // Submit button
        private void submitButton_Click(object sender, EventArgs e)
        {
            try
            {
                bool isEmptyColumn = true;

                if (isEmptyColumn == true)
                {
                    // If all of the checkboxes of a row are unselected,
                    isEmptyColumn will be set to false.
                }
            }
            catch { }
        }
    }
}

```

```

        for (int i = 0; i < ratingTable.Rows.Count; i++)
        {
            if (ratingTable.Rows[i].Cells[3].Value == null &&
ratingTable.Rows[i].Cells[4].Value == null &&
ratingTable.Rows[i].Cells[5].Value == null &&
ratingTable.Rows[i].Cells[6].Value == null)
            {
                isColumnEmpty = false;
            }
        }

        if (serviceNameComboBox.SelectedItem == null)
        {
            MessageBox.Show("Please select the service.", "Error!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        else if (isColumnEmpty == false)
        {
            MessageBox.Show("Please give ratings to all the criteria.",
"Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        else
        {
            submitRatingsData();
            MessageBox.Show("Thank you for your rating and feedback.",
"Rating and Feedback System", MessageBoxButtons.OK,
MessageBoxIcon.Information);

            // After rating is submitted, textboxes will be set to ""
            customerNameTextBox.Text = "";
            contactNumberTextBox.Text = "";
            emailAddressTextBox.Text = "";
            addressTextBox.Text = "";
            feedbackTextBox.Text = "";

            // After the ratings are submitted, all the checkbox values
will be set to null.
            for (int i = 0; i < ratingTable.Rows.Count; i++)
            {
                if (ratingTable.Rows[i].Cells[3].Value != null)
                {
                    ratingTable.Rows[i].Cells[3].Value = null;
                }

                if (ratingTable.Rows[i].Cells[4].Value != null)
                {
                    ratingTable.Rows[i].Cells[4].Value = null;
                }

                if (ratingTable.Rows[i].Cells[5].Value != null)
                {
                    ratingTable.Rows[i].Cells[5].Value = null;
                }

                if (ratingTable.Rows[i].Cells[6].Value != null)
                {
                    ratingTable.Rows[i].Cells[6].Value = null;
                }
            }
        }
    }
}

```



```

        // End of for loop
    }

    // End of else statement
}

catch (Exception ex)
{
    MessageBox.Show(ex + "", "Error!", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}

}

// This method will retrieve criteria data from xml file.
void getCriteriaData()
{
    //reading data from dataset

    try
    {
        DataSet dsRead = new DataSet();
        dsRead.ReadXmlSchema(@"../CriteriaSchema.xml");
        dsRead.ReadXml(@"../CriteriaData.xml");

        if (dsRead.Tables["Criteria"] != null)
        {
            for (int i = 0; i < dsRead.Tables["Criteria"].Rows.Count;
i++)
            {
                ratingTable.Rows.Add();
                ratingTable.Rows[i].Cells["snColumn"].Value = i + 1;
                ratingTable.Rows[i].Cells["criteriaColumn"].Value =
dsRead.Tables["Criteria"].Rows[i][1].ToString();
                ratingTable.Rows[i].Cells["descriptionColumn"].Value =
dsRead.Tables["Criteria"].Rows[i][2].ToString();
            }
        }

        catch (System.IO.FileNotFoundException ex)
        {
            Console.WriteLine(ex);
            MessageBox.Show("The criteria file cannot be found in the saved
directory. Please verify it once.", "Error!", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        }

    }

    // This method will be called when submit button is clicked.
    void submitRatingsData()
    {
        // Writing data into XML file
        DataHandler dh = new DataHandler();
        DataSet ds = new DataSet();
        ds = dh.CreateDataSet();

        AddSampleDataForRatings(ds);
    }
}

```

```

        ds.WriteXmlSchema(@"../RatingsSchema.xml");
        ds.WriteXml(@"../RatingsData.xml");
    }
    void AddSampleDataForRatings(DataSet dataSet)
    {
        try
        {
            DataSet dsRead = new DataSet();
            dsRead.ReadXmlSchema(@"../RatingsSchema.xml");
            dsRead.ReadXml(@"../RatingsData.xml");

            DataRow dr;

            // If the data in Ratings table is not null, then only it's
            statement will be executed.
            if (dsRead.Tables["Ratings"] != null)
            {
                // All the data from previous xml file will be added in the
                new xml file.
                for (int i = 0; i < dsRead.Tables["Ratings"].Rows.Count;
                i++)
                {
                    dr = dataSet.Tables["Ratings"].NewRow();

                    dr["Service Name"] =
                    dsRead.Tables["Ratings"].Rows[i]["Service Name"].ToString();
                    dr["Customer Name"] =
                    dsRead.Tables["Ratings"].Rows[i]["Customer Name"].ToString();
                    dr["Contact Number"] =
                    dsRead.Tables["Ratings"].Rows[i]["Contact Number"].ToString();
                    dr["Email Address"] =
                    dsRead.Tables["Ratings"].Rows[i]["Email Address"].ToString();
                    dr["Address"] =
                    dsRead.Tables["Ratings"].Rows[i]["Address"].ToString();
                    dr["Feedback"] =
                    dsRead.Tables["Ratings"].Rows[i]["Feedback"].ToString();
                    dr["Date"] =
                    dsRead.Tables["Ratings"].Rows[i]["Date"].ToString();

                    dr["Criteria"] =
                    dsRead.Tables["Ratings"].Rows[i]["Criteria"].ToString();
                    dr["Criteria Value"] =
                    dsRead.Tables["Ratings"].Rows[i]["Criteria Value"].ToString();
                    dataSet.Tables["Ratings"].Rows.Add(dr);
                }
            }

            //for loop is run from the criteria that we read from criteria
            xml file.
            for (int i = 0; i < ratingTable.Rows.Count; i++)
            {
                //criteria variable will store the criteria name from
                ratingTable DataGridView
                string criteria;
                criteria =
                ratingTable.Rows[i].Cells["criteriaColumn"].Value.ToString();

                string CriteriaValue = "0";

                string Excellent;
                string Good;
                string Average;
            }
        }
        catch { }
    }
}

```

```

        string Dissatisfied;

        //
        if (ratingTable.Rows[i].Cells[3].Value != null)
        {
            Excellent =
ratingTable.Rows[i].Cells[3].Value.ToString();
            if (Excellent == "True")
            {
                CriteriaValue = "5";
            }
        }

        if (ratingTable.Rows[i].Cells[4].Value != null)
        {
            Good = ratingTable.Rows[i].Cells[4].Value.ToString();
            if (Good == "True")
            {
                CriteriaValue = "3";
            }
        }

        if (ratingTable.Rows[i].Cells[5].Value != null)
        {
            Average =
ratingTable.Rows[i].Cells[5].Value.ToString();
            if (Average == "True")
            {
                CriteriaValue = "2";
            }
        }

        if (ratingTable.Rows[i].Cells[6].Value != null)
        {
            Dissatisfied =
ratingTable.Rows[i].Cells[6].Value.ToString();
            if (Dissatisfied == "True")
            {
                CriteriaValue = "1";
            }
        }

        // Adding new row in the Ratings table.
        dr = dataSet.Tables["Ratings"].NewRow();

        dr["Service Name"] =
serviceNameComboBox.SelectedItem.ToString();
        dr["Customer Name"] = customerNameTextBox.Text;
        dr["Contact Number"] = contactNumberTextBox.Text;
        dr["Email Address"] = emailAddressTextBox.Text;
        dr["Address"] = addressTextBox.Text;
        dr["Feedback"] = feedbackTextBox.Text;
        dr["Date"] = dateTimePicker.Value.ToString("MM/dd/yyyy");

        dr["Criteria"] = criteria;
        dr["Criteria Value"] = CriteriaValue;
        dataSet.Tables["Ratings"].Rows.Add(dr);

        // End of for loop
    }
}

```

```
        catch (System.IO.FileNotFoundException ex)
        {
            Console.WriteLine(ex);
            MessageBox.Show("The ratings file cannot be found in the saved
directory. Please verify it once.", "Error!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }

        catch (Exception ex)
        {
            Console.WriteLine(ex);
            MessageBox.Show(ex + "", "Error!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void pictureBox1_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void loginAsAdminButton_Click(object sender, EventArgs e)
    {
        // Closing the current running form, and calling OpenLoginForm
method.
        this.Close();
        thread = new Thread(OpenLoginForm);
        thread.SetApartmentState(ApartmentState.STA);
        thread.Start();
    }

    public static void OpenLoginForm()
    {
        // It will run the new form, i.e. LoginForm.
        Application.Run(new LoginForm());
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
    }

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

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

    private void label1_Click(object sender, EventArgs e)
    {
    }

    private void addressLabel_Click(object sender, EventArgs e)
    {
    }
}
```

```

        private void CustomerUI_Load(object sender, EventArgs e)
        {

        }

    }
}

```

## 7. DataHandler.cs

```

using System;
using System.Data;

namespace Handler
{
    public class DataHandler
    {
        public DataSet CreateDataSet()
        {
            var ds = new DataSet();
            ds.Tables.Add(CreateCriteriaTable());
            ds.Tables.Add(CreateRatingsTable());
            ds.Tables.Add(CreateReportTable());
            return ds;
        }

        private DataTable CreateCriteriaTable()
        {
            var dt = new DataTable("Criteria");
            DataColumn dataColumn = new DataColumn("ID", typeof(int));
            dataColumn.AutoIncrement = true;
            dataColumn.AutoIncrementSeed = 1;
            dataColumn.AutoIncrementStep = 1;

            dt.Columns.Add(dataColumn);
            dt.Columns.Add("Criteria Name", typeof(string));
            dt.Columns.Add("Description", typeof(string));

            dt.PrimaryKey = new DataColumn[]
            {
                dt.Columns["ID"]
            };
            return dt;
        }

        private DataTable CreateRatingsTable()
        {
            var dt = new DataTable("Ratings");
            DataColumn dataColumn = new DataColumn("ID", typeof(int));
            dataColumn.AutoIncrement = true;
            dataColumn.AutoIncrementSeed = 1;
            dataColumn.AutoIncrementStep = 1;
            dt.Columns.Add(dataColumn);

            dt.Columns.Add("Service Name", typeof(string));
            dt.Columns.Add("Customer Name", typeof(string));
            dt.Columns.Add("Contact Number", typeof(string));
            dt.Columns.Add("Email Address", typeof(string));
            dt.Columns.Add("Address", typeof(string));
            dt.Columns.Add("Feedback", typeof(string));
            dt.Columns.Add("Date", typeof(string));
        }
    }
}

```

```

        dt.Columns.Add("Criteria", typeof(string));
        dt.Columns.Add("Criteria Value", typeof(string));
        dt.PrimaryKey = new DataColumn[]
        {
            dt.Columns["ID"]
        };

        return dt;
    }

    private DataTable CreateReportTable()
    {
        var dt = new DataTable("Report");
        DataColumn dataColumn = new DataColumn("ID", typeof(int));
        dataColumn.AutoIncrement = true;
        dataColumn.AutoIncrementSeed = 1;
        dataColumn.AutoIncrementStep = 1;
        dt.Columns.Add(dataColumn);

        dt.Columns.Add("Criteria", typeof(string));
        dt.Columns.Add("Description", typeof(string));
        dt.Columns.Add("Excellent", typeof(string));
        dt.Columns.Add("Good", typeof(string));
        dt.Columns.Add("Average", typeof(string));
        dt.Columns.Add("Dissatisfied", typeof(string));
        dt.Columns.Add("Mean", typeof(string));
        dt.Columns.Add("Overall", typeof(string));

        dt.PrimaryKey = new DataColumn[]
        {
            dt.Columns["ID"]
        };

        return dt;
    }
}

```

## 8. GraphOption.cs

```

using System;
using System.Windows.Forms;

namespace Development
{
    public partial class GraphOption : Form
    {
        public GraphOption()
        {
            InitializeComponent();
        }

        private void meanRatingButton_Click(object sender, EventArgs e)
        {
            BarGraph barGraph = new BarGraph();
            barGraph.Show();
        }

        private void overallRatingButton_Click(object sender, EventArgs e)
        {
            PieChart pieChart = new PieChart();
        }
    }
}

```

```

        pieChart.Show();
    }

    private void pictureBox1_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void reportOptionPanel_Paint(object sender, PaintEventArgs e)
    {
    }

    }
}

```

## 9. LoginForm.cs

```

using System;
using System.Windows.Forms;
using System.Threading;

namespace Development
{
    public partial class LoginForm : Form
    {
        Thread thread;
        public LoginForm()
        {
            InitializeComponent();
        }

        private void loginButton_Click(object sender, EventArgs e)
        {
            // Defining username and password.
            string username = "Kushal";
            string password = "Gurung";

            if (usernameTextBox.Text == username && passwordTextBox.Text ==
password)
            {
                MessageBox.Show("Welcome Admin, " + username+".", "Rating and
Feedback System", MessageBoxButtons.OK, MessageBoxIcon.Information);
                this.Hide();
                AdminDashboard adminDashboard = new AdminDashboard();
                adminDashboard.Show();
            }
            else
            {
                if (usernameTextBox.Text == "" && passwordTextBox.Text == "")
                {
                    MessageBox.Show("Please Enter Username and Password.",
"Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }

                else if (usernameTextBox.Text == "")
                {
                    MessageBox.Show("Please Enter Your Username.", "Error!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }

                else if (passwordTextBox.Text == "")

```

```

        {
            MessageBox.Show("Please Enter Your Password.", "Error!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            MessageBox.Show("Username and Password doesn't match.",
"Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
            usernameTextBox.Text = "";
            usernameTextBox.Focus();
            passwordTextBox.Text = "";
        }
    }
}

private void pictureBox1_Click(object sender, EventArgs e)
{
    Application.Exit();
}

// Customer Dashboard button
private void button1_Click_1(object sender, EventArgs e)
{
    // Closing the current running form, and calling OpenCustomerUI
method.
    this.Close();
    thread = new Thread(OpenCustomerUI);
    thread.SetApartmentState(ApartmentState.STA);
    thread.Start();
}

public static void OpenCustomerUI()
{
    // It will run the new form, i.e. CustomerUI.
    Application.Run(new CustomerUI());
}

private void label1_Click(object sender, EventArgs e)
{
}

private void label1_Click_1(object sender, EventArgs e)
{
}

private void label2_Click(object sender, EventArgs e)
{
}

private void button1_Click(object sender, EventArgs e)
{
}

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

```



```
    }
}
```

## 10. OverallReport.cs

```
using System;
using System.Data;
using System.Windows.Forms;
using Handler;

namespace Development
{
    public partial class OverallReport : Form
    {
        public OverallReport()
        {
            InitializeComponent();
            getReportData();
        }

        void getReportData()
        {
            try
            {
                // Reading data from dataset
                DataSet dsRead = new DataSet();

                // For criteria
                dsRead.ReadXmlSchema(@"../CriteriaSchema.xml");
                dsRead.ReadXml(@"../CriteriaData.xml");

                // For Ratings
                dsRead.ReadXmlSchema(@"../RatingsSchema.xml");
                dsRead.ReadXml(@"../RatingsData.xml");

                if (dsRead.Tables["Criteria"] != null &&
dsRead.Tables["Ratings"] != null)
                {
                    for (int i = 0; i < dsRead.Tables["Criteria"].Rows.Count;
i++)
                    {
                        reportTable.Rows.Add();
                        reportTable.Rows[i].Cells["snColumn"].Value = i + 1;
                        reportTable.Rows[i].Cells["criteriaColumn"].Value =
dsRead.Tables["Criteria"].Rows[i][1].ToString();
                        reportTable.Rows[i].Cells["descriptionColumn"].Value =
dsRead.Tables["Criteria"].Rows[i][2].ToString();

                        int excellentRating = 0;
                        int goodRating = 0;
                        int averageRating = 0;
                        int dissatisfiedRating = 0;

                        for (int j = 0; j <
dsRead.Tables["Ratings"].Rows.Count; j++)
                        {
                            // If the data in Criteria column of Ratings table
                            is equal to report table's Criteria column's data, then only the statement will
                            be executed.

                            if (dsRead.Tables["Ratings"].Rows[j][8].ToString()
== reportTable.Rows[i].Cells["criteriaColumn"].Value.ToString())
```

```

        {
            if
(dsRead.Tables["Ratings"].Rows[j][9].ToString() == "5")
            {

reportTable.Rows[i].Cells["excellentColumn"].Value =
Convert.ToString(Convert.ToInt32(reportTable.Rows[i].Cells["excellentColumn"].V
alue) + 1);

                excellentRating = excellentRating + 1;
            }

            else if
(dsRead.Tables["Ratings"].Rows[j][9].ToString() == "3")
            {

reportTable.Rows[i].Cells["goodColumn"].Value =
Convert.ToString(Convert.ToInt32(reportTable.Rows[i].Cells["goodColumn"].Value)
+ 1);

                goodRating = goodRating + 1;
            }

            else if
(dsRead.Tables["Ratings"].Rows[j][9].ToString() == "2")
            {

reportTable.Rows[i].Cells["averageColumn"].Value =
Convert.ToString(Convert.ToInt32(reportTable.Rows[i].Cells["averageColumn"].Val
ue) + 1);

                averageRating = averageRating + 1;
            }

            else if
(dsRead.Tables["Ratings"].Rows[j][9].ToString() == "1")
            {

reportTable.Rows[i].Cells["dissatisfiedColumn"].Value =
Convert.ToString(Convert.ToInt32(reportTable.Rows[i].Cells["dissatisfiedColumn"
].Value) + 1);

                dissatisfiedRating = dissatisfiedRating +
1;
            }

        }

        // End of nested for loop
    }

    // End of main for loop
}

decimal excellent = 0;
decimal good = 0;
decimal average = 0;
decimal dissatisfied = 0;

for (int i = 0; i < reportTable.Rows.Count; i++)
{
    if (reportTable.Rows[i].Cells["excellentColumn"].Value ==
null)
    {
        reportTable.Rows[i].Cells["excellentColumn"].Value = 0;
    }
}

```

```

    }

    if (reportTable.Rows[i].Cells["goodColumn"].Value == null)
    {
        reportTable.Rows[i].Cells["goodColumn"].Value = 0;
    }

    if (reportTable.Rows[i].Cells["averageColumn"].Value ==
null)
    {
        reportTable.Rows[i].Cells["averageColumn"].Value = 0;
    }

    if (reportTable.Rows[i].Cells["dissatisfiedColumn"].Value
== null)
    {
        reportTable.Rows[i].Cells["dissatisfiedColumn"].Value =
0;
    }

    if (reportTable.Rows[i].Cells["meanColumn"].Value == null)
    {
        reportTable.Rows[i].Cells["meanColumn"].Value = 0;
    }

    if (reportTable.Rows[i].Cells["overallColumn"].Value ==
null)
    {
        reportTable.Rows[i].Cells["overallColumn"].Value = 0;
    }

    excellent =
Convert.ToInt32(reportTable.Rows[i].Cells["excellentColumn"].Value.ToString());
    good =
Convert.ToInt32(reportTable.Rows[i].Cells["goodColumn"].Value.ToString());
    average =
Convert.ToInt32(reportTable.Rows[i].Cells["averageColumn"].Value.ToString());
    dissatisfied =
Convert.ToInt32(reportTable.Rows[i].Cells["dissatisfiedColumn"].Value.ToString(
));

    try
    {
        if(excellent == 0 && good == 0 && average == 0 &&
dissatisfied == 0)
        {
            // If newly added criteria don't have any ratings
            (,i.e. 0 ratings), we cannot get mean rating, because we will get undefined
            value while dividing by zero.
            // Hence, we will set mean ratings value to 0 by
            ourselves.
            reportTable.Rows[i].Cells["meanColumn"].Value = 0;
        }
        else
        {
            // We use String.Format() to get two decimal place
            value for our Mean Column
            reportTable.Rows[i].Cells["meanColumn"].Value =
String.Format("{0:}.##}", ((excellent * 5) + (good * 3) + (average * 2) +
(dissatisfied * 1)) / (excellent + good + average + dissatisfied));
        }
    }

```

```

        // Overall Column
        reportTable.Rows[i].Cells["overallColumn"].Value =
        ((excellent * 5) + (good * 3) + (average * 2) + (dissatisfied * 1));

    }

    catch (Exception ex)
    {
        Console.WriteLine(ex);
        MessageBox.Show(ex + "", "Error!",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

// For creating Report file
DataHandler dh = new DataHandler();
DataSet ds = new DataSet();
ds = dh.CreateDataSet();

AddSampleDataForReport(ds);
ds.WriteXmlSchema(@"../ReportSchema.xml");
ds.WriteXml(@"../ReportData.xml");
}

catch (System.IO.FileNotFoundException ex)
{
    Console.WriteLine(ex);
    MessageBox.Show("Required files cannot be found in the saved
    directory. Please verify it once.", "Error!", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}

catch (Exception ex)
{
    Console.WriteLine(ex);
    MessageBox.Show(ex + "", "Error!", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}
}

// This method will now add all the overall data row in the dataset
void AddSampleDataForReport(DataSet dataSet)
{
    // Creating a new DataRow
    DataRow dr;

    for (int i = 0; i < reportTable.Rows.Count; i++)
    {
        // Adding new row in the Report table.
        dr = dataSet.Tables["Report"].NewRow();

        dr["Criteria"] =
        reportTable.Rows[i].Cells["criteriaColumn"].Value.ToString();
        dr["Description"] =
        reportTable.Rows[i].Cells["descriptionColumn"].Value.ToString();
        dr["Excellent"] =
        reportTable.Rows[i].Cells["excellentColumn"].Value.ToString();
        dr["Good"] =
        reportTable.Rows[i].Cells["goodColumn"].Value.ToString();
        dr["Average"] =
        reportTable.Rows[i].Cells["averageColumn"].Value.ToString();
    }
}

```

```

                dr["Dissatisfied"] =
reportTable.Rows[i].Cells["dissatisfiedColumn"].Value.ToString();
                dr["Mean"] =
reportTable.Rows[i].Cells["meanColumn"].Value.ToString();
                dr["Overall"] =
reportTable.Rows[i].Cells["overallColumn"].Value.ToString();

                dataSet.Tables["Report"].Rows.Add(dr);
            }
        }

        private void Report_Load(object sender, EventArgs e)
        {

        }

        private void reportTable_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

    }
}

```

## 11.PieChart.cs

```

using System;
using System.Data;
using System.Windows.Forms.DataVisualization.Charting;
using System.Drawing;
using System.Windows.Forms;

using System.Collections;

namespace Development
{
    public partial class PieChart : Form
    {

        public PieChart()
        {

            InitializeComponent();
            DisplayPieChart();
        }

        public void DisplayPieChart()
        {

            try
            {
                ArrayList criteriaArrayList = new ArrayList();
                ArrayList overallArrayList = new ArrayList();

                // Reading data from dataset
                DataSet dsRead = new DataSet();
                dsRead.ReadXmlSchema(@"../ReportSchema.xml");
                dsRead.ReadXml(@"../ReportData.xml");

                for (int i = 0; i < dsRead.Tables["Report"].Rows.Count; i++)
                {

```

```

        // Criteria Name
criteriaArrayList.Add(dsRead.Tables["Report"].Rows[i][1].ToString());

        // Overall Rating
overallArrayList.Add(dsRead.Tables["Report"].Rows[i][8].ToString());

chart2.Series[0].Points.AddXY(criteriaArrayList[i].ToString(),
Convert.ToDouble(overallArrayList[i]));
    }

    this.chart2.Series[0]["PieLabelStyle"] = "Outside";

    // Set border width so that labels are shown on the outside
    this.chart2.Series[0].BorderWidth = 1;
    this.chart2.Series[0].BorderColor = Color.FromArgb(26, 59,
105);

    // Add a legend to the chart and dock it to the bottom-center
    this.chart2.Legends[0].Enabled = true;
    this.chart2.Legends[0].Docking = Docking.Bottom;
    this.chart2.Legends[0].Alignment = StringAlignment.Center;

    // Set the legend to display pie chart values as percentages
    // Again, the P2 indicates a precision of 2 decimals
    this.chart2.Series[0].LegendText = "#PERCENT{P2}";

    // By sorting the data points, they show up in proper ascending
order in the legend
    this.chart2.DataManipulator.Sort(PointSortOrder.Descending,
chart2.Series[0]);

    //chart2.Series[0].ChartType = SeriesChartType.Column;
    chart2.Series[0].ChartType = SeriesChartType.Pie;

    chart2.Titles.Add("Overall rating of each criteria");
}

catch (System.IO.FileNotFoundException ex)
{
    Console.WriteLine(ex);
    MessageBox.Show("The Report file cannot be found in the saved
directory. Please verify it once.", "Error!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}

}

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

private void chart2_Click(object sender, EventArgs e)
{
}

}
}

```

## 12. Program.cs

```
using System;
using System.Windows.Forms;

namespace Development
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new AdminDashboard());
        }
    }
}
```

## 13. ReportOption.cs

```
using System;
using System.Windows.Forms;

namespace Development
{
    public partial class ReportOption : Form
    {
        public ReportOption()
        {
            InitializeComponent();
        }

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

        private void panel1_Paint(object sender, PaintEventArgs e)
        {
        }

        private void customerReport_Click(object sender, EventArgs e)
        {
            CustomerReport customerReport = new CustomerReport();
            customerReport.Show();
        }

        private void overallReportButton_Click(object sender, EventArgs e)
        {
            OverallReport overallReport = new OverallReport();
            overallReport.Show();
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

```
private void reportOptionPanel_Paint(object sender, PaintEventArgs e)
{
}

private void label1_Click(object sender, EventArgs e)
{
}
}
```