**Table of Content**

1. [Introduction 1](#_bookmark0)
2. [Detailed Instruction to run the program 2](#_bookmark1)
3. [System Architecture 9](#_bookmark14)
4. [Class Diagram 10](#_bookmark16)
5. [Classes and their purpose 12](#_bookmark20)
6. [Method Description 14](#_bookmark26)
7. [Data Structure and Algorithm 19](#_bookmark32)
   1. [Data Structure 19](#_bookmark33)
   2. [Algorithm 21](#_bookmark37)
8. [Reflection of the learning experience 22](#_bookmark40)
9. [Conclusion 23](#_bookmark41)
   1. [Evaluation of work 23](#_bookmark42)
   2. [Learning Outcomes 23](#_bookmark43)
   3. [Difficulties encountered and it's overcome 24](#_bookmark44)
10. [References 25](#_bookmark45)
11. [Appendix 26](#_bookmark46)

**Table of Figures**

[Figure 1: Main UI of the system (Login Form) 2](#_bookmark2)

[Figure 2: Login of Staff 3](#_bookmark3)

[Figure 3: Staff Dashboard 3](#_bookmark4)

[Figure 4: Screenshot of view ticket rates 4](#_bookmark5)

[Figure 5: Submit entry (Customers details) 4](#_bookmark6)

[Figure 6: Screenshot of view customer’s details 5](#_bookmark7)

[Figure 7: Screenshot of view daily report (staff dashboard) 6](#_bookmark8)

[Figure 8: Screenshot of view weekly report (Staff dashboard) 6](#_bookmark9)

[Figure 9: Screenshot of view weekly graph (Staff dashboard) 7](#_bookmark10)

[Figure 10: Login of Admin 7](#_bookmark11)

[Figure 11: Admin dashboard 8](#_bookmark12)

[Figure 12: Screenshot of adding ticket 8](#_bookmark13)

[Figure 13: System Architecture design 9](#_bookmark15)

[Figure 14: Class diagram of admin dashboard class and staff dashboard class 10](#_bookmark17)

[Figure 15: Class diagram of login class and daily report class 11](#_bookmark18)

[Figure 16: Class diagram of the weekly report class 11](#_bookmark19)

[Figure 17: Screenshot of array data structure used 19](#_bookmark34)

[Figure 18: Screenshot of string data structure used 20](#_bookmark35)

[Figure 19: Screenshot of hash table data structure used 20](#_bookmark36)

[Figure 20: Screenshot of list data structure used 21](#_bookmark38)

[Figure 21: Screenshot of bubble sort algorithm used 21](#_bookmark39)

**Table of Table**

[Table 1: purpose of Login class 12](#_bookmark21)

[Table 2: Purpose of Staff Dashboard class 12](#_bookmark22)

[Table 3: Purpose of Admin Dashboard class 13](#_bookmark23)

[Table 4: Purpose of Daily report class 13](#_bookmark24)

[Table 5: Purpose of Weekly report class 13](#_bookmark25)

[Table 6: Method description of Login.cs class 14](#_bookmark27)

[Table 7: Method description of StaffDashboard.cs class 16](#_bookmark28)

[Table 8: Method description of AdminDashboard.cs class 18](#_bookmark29)

[Table 9: Method description of WeeklyReport.cs class 18](#_bookmark30)

[Table 10: Method description of DailyReport.cs class 18](#_bookmark31)

# Introduction

The purpose of this report is to describe the design and implementation of a C# desktop application for a recreation center to manage visitor records. For the development of the desktop application, this Visual Studio Dot Net framework was used. It is not a database or web-based application. C# is a modern, object-oriented programming language developed by Microsoft under the leadership of Anders Hejlsberg and his team as part of the .Net initiative, and it has been approved by the European Computer Manufacturers Association (ECMA) (GeeksforGeeks, 2022) . It enables developers to create a wide range of secure and robust .NET applications. Dot Net Framework is a software development platform for creating applications that run on the Windows platform. The Dot Net framework is made up of developer tools, programming languages, and libraries that can be used to create desktop and web- based applications. (Barbara Thompson, 2021)

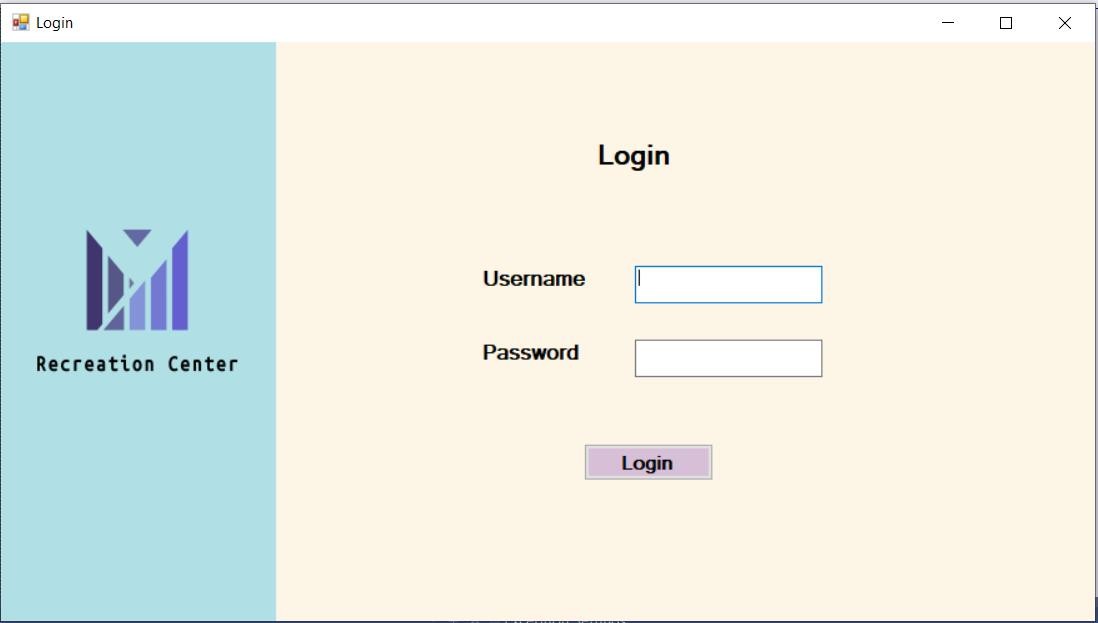
The Recreation Center is currently facing a problem using a paper-based manual system to keep records of tickets and visitors. So to solve this problem the application is designed to keep the trace of each visitor’s details with the time duration that they spend inside the recreation center. The application also keeps the ticket rate based on different age groups and the number of visitors. The recreation center opens for visitors from 10:00 AM to 6:00 PM and has different ticket options based on age (Child and adult), based on the group (Group of 5, Group of 10), and time duration (1hrs, 2hrs …, unlimited). The proposed application has the following features:

* + Generate and display two different reports, a daily report by listing the total number of visitors with different visitor categories set by the employee in the ticket option. The other weekly report includes the number of visitors on each day and is sorted by the total number of visitors and total earning in each day.
  + Display weekly chart showing each day’s total number of visitors and total earning.
  + To import ticket price rate data and visitors in-out time from a . CSV file.
  + Save and retrieve the ticket details and visitors in-out time.

# Detailed Instruction to run the program

The recreation center system is the system related to records of tickets and visitors. The system of the recreation center is simple and easy to use but the instruction will help the user to run the system efficiently and effectively. So the user manual or the detailed instruction to run the program is explained below:

#### Main UI of the system (Login Form)



*Figure 1: Main UI of the system (Login Form)*

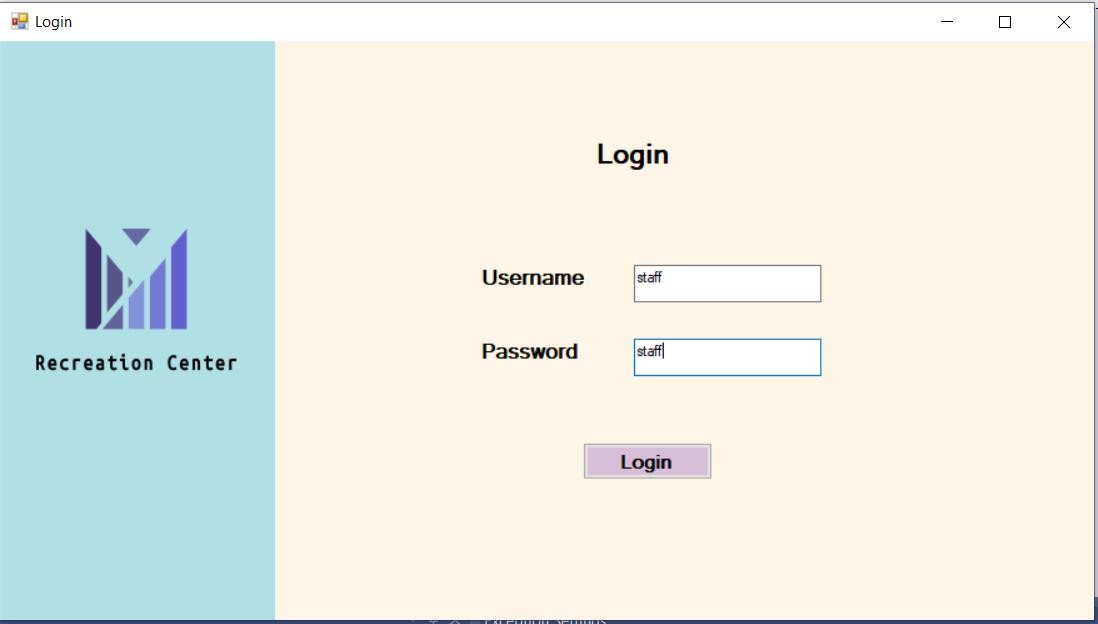
This is the main form, which is a login form that loads when the system is launched. This is the primary form through which staff and administrators can access their respective dashboards. Staff and Admin can both login from this page because the username and password for both are specified:

Username for Admin: admin Username for Staff: staff

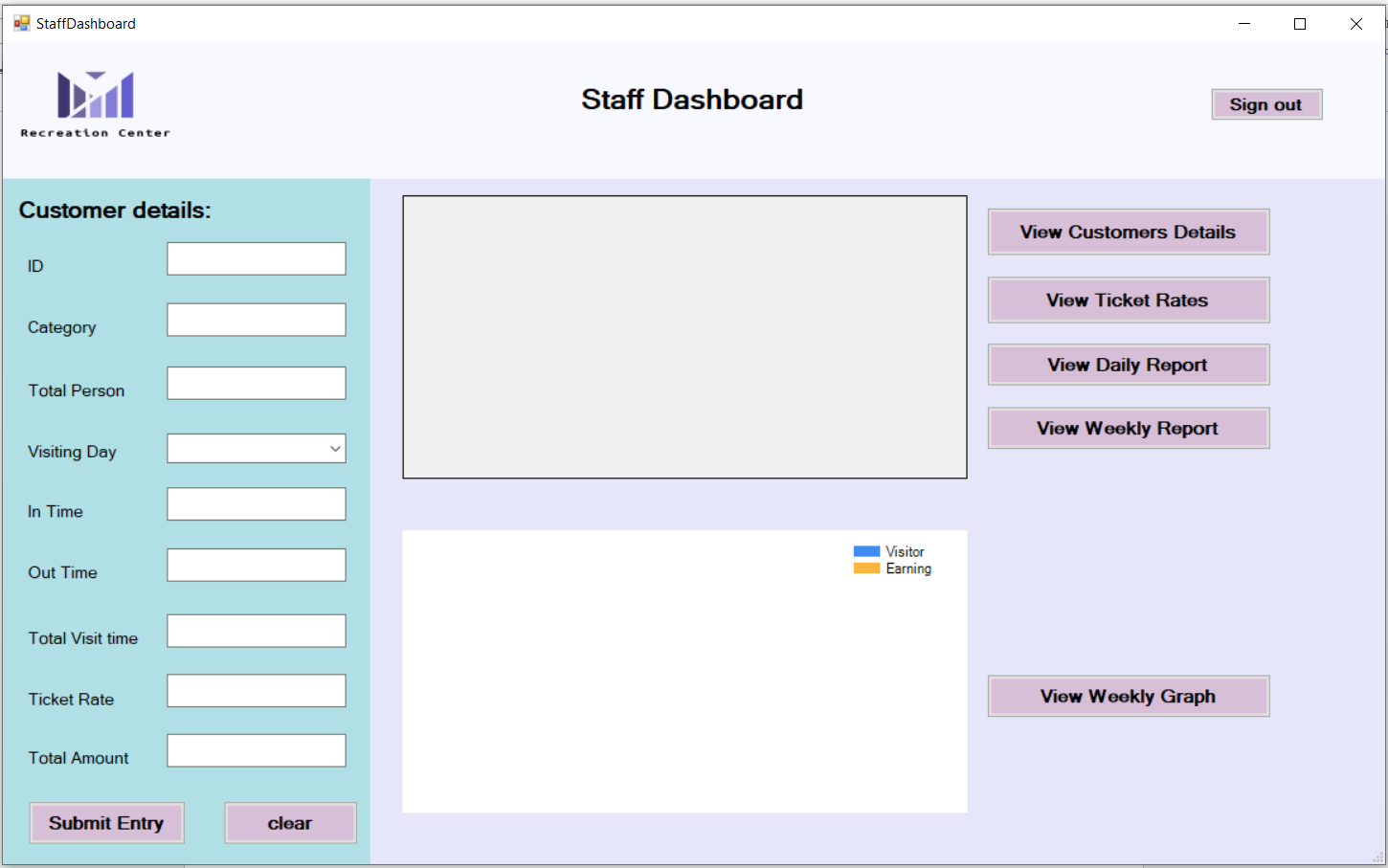
Password for Admin: admin Password for Staff: staff

The specified username and the password should be used to login into the system.

#### Staff Dashboard



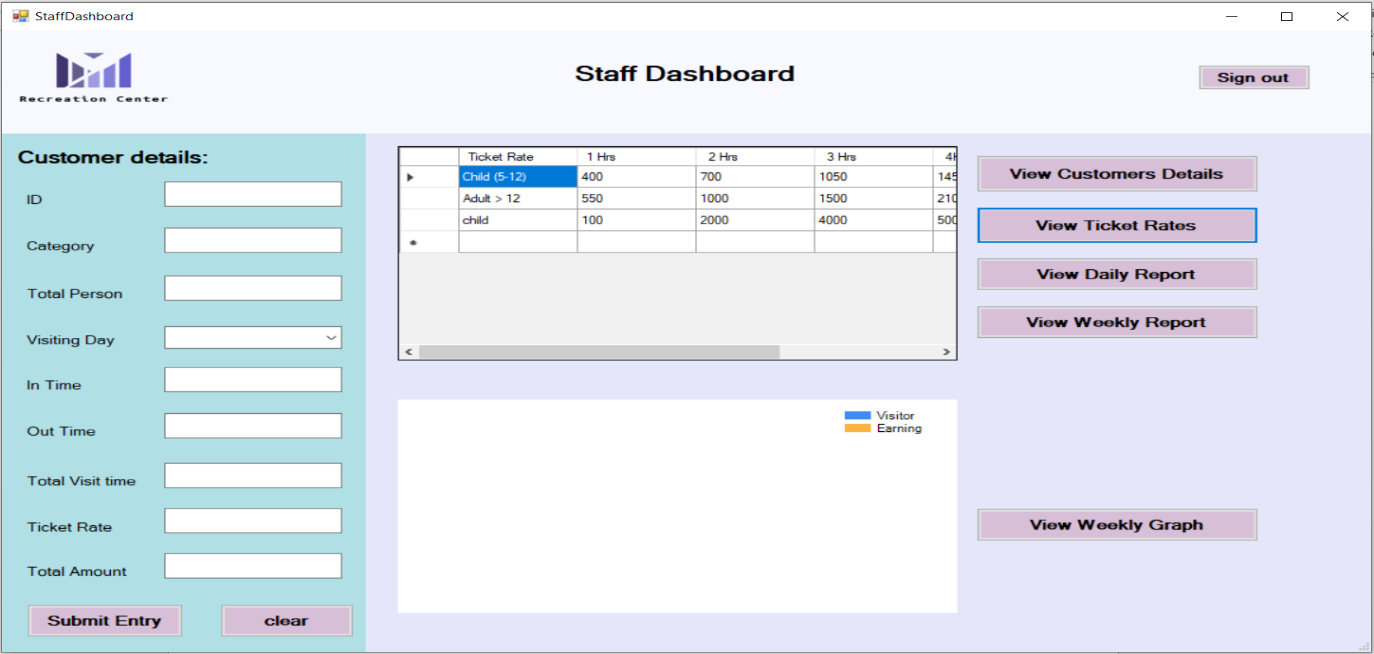
*Figure 2: Login of Staff*



*Figure 3: Staff Dashboard*

When the Staff enters the specified username and password, they will be presented with the staff dashboard, as shown in figure 3, where they can add customer’s details, view customer’s details, view ticket rate, generate daily and weekly reports, and view weekly graph. When staff clicks the sign-out button, they are redirected to the login form.

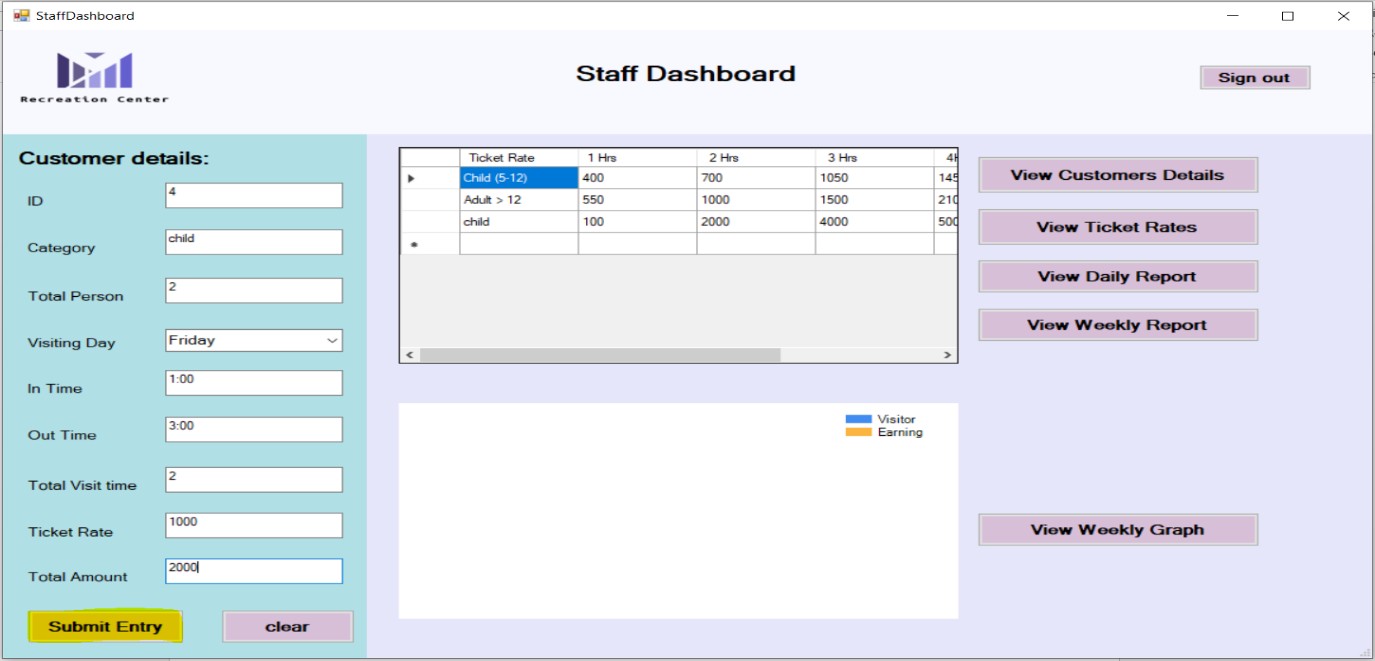
#### View Ticket Rate



*Figure 4: Screenshot of view ticket rates*

When employees first log in to the staff dashboard, they view the ticket rates that the admin has entered. The ticket rates are specified according to the hours and category. This makes it easier for staff to enter the ticket price for customers according to their category.

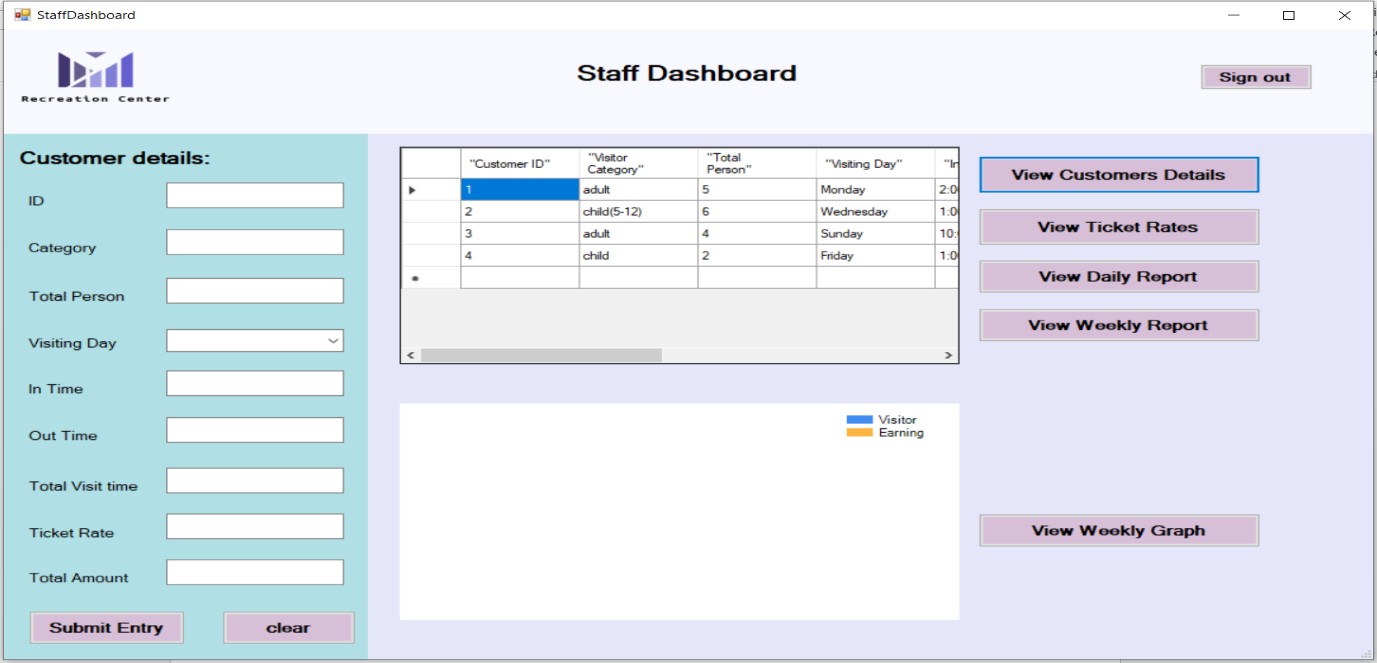
#### Submit Entry (Customers details)



*Figure 5: Submit entry (Customers details)*

The staff enters all the details of the customers. The ticket rate is according to the rate set by the admin that is viewed in the grid. The ticket rates are fixed and determined by the admin. The ticket rate is calculated in the customer details panel based on the category and total visit time. And the total amount is calculated based on the total number of people. When the submit entry button is pressed the customers' details will be recorded.

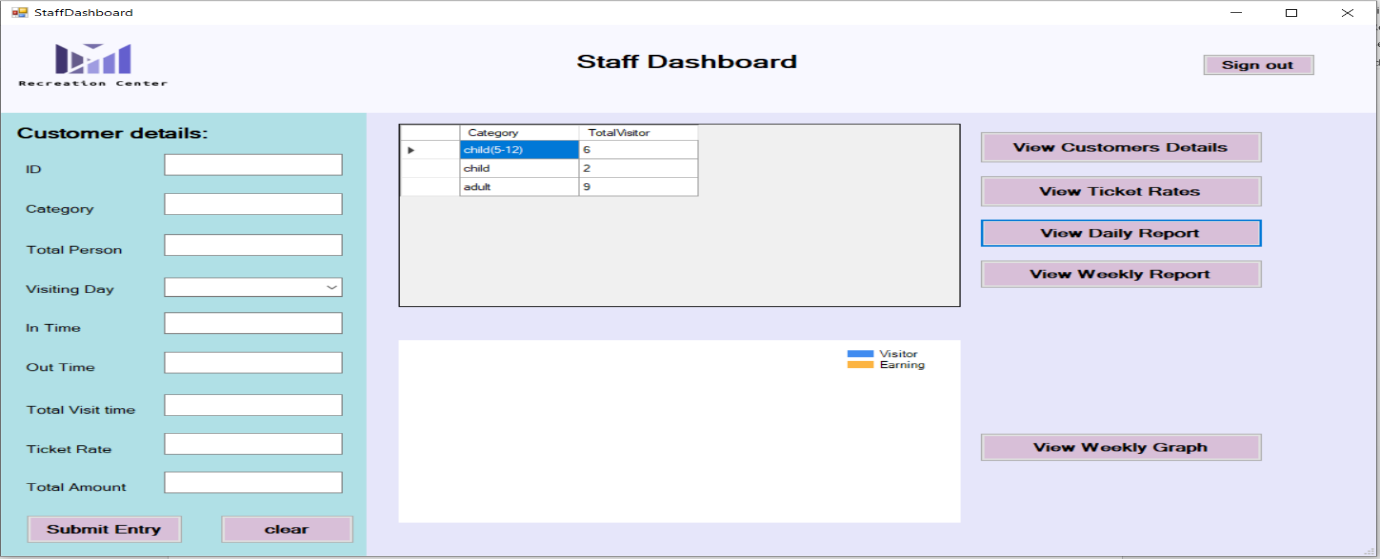
#### View Customer details



*Figure 6: Screenshot of view customer’s details*

The details of the customers recorded by the staff are saved in a CSV file, which is retrieved in the grid view. When customers press the view customer details button, the recorded details of the customer are presented in the grid as shown in above figure 6.

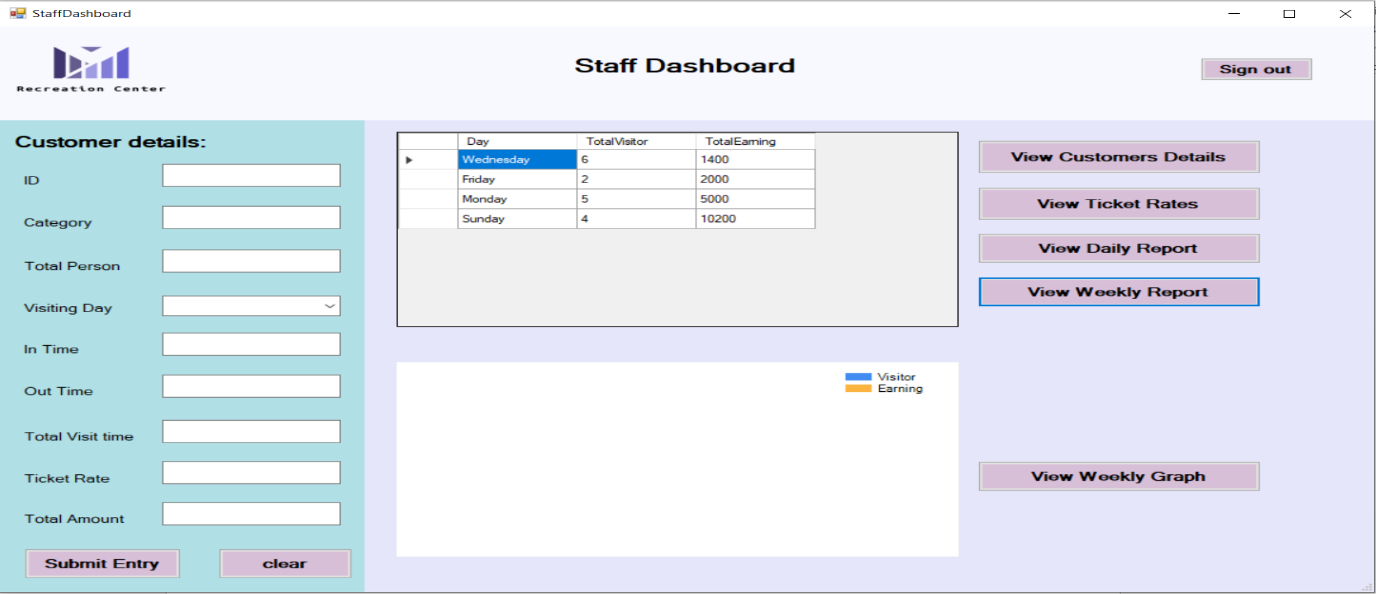
#### View Daily Report



*Figure 7: Screenshot of view daily report (staff dashboard)*

The staff generates the daily report, which is displayed in the grid and the admin can also view it in the admin dashboard. The category and total visitors are included in the daily reports. The daily report provides an overview of how many visitors of a specific category have arrived.

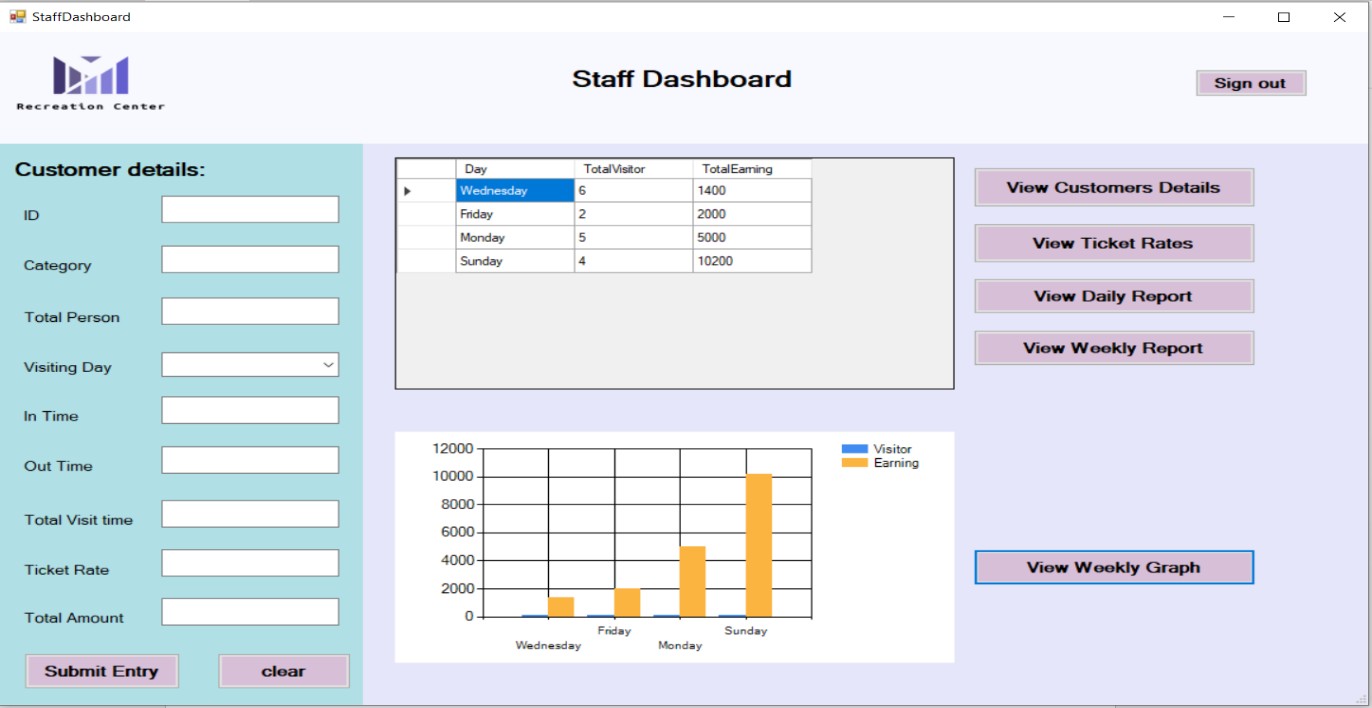
#### View Weekly Report



*Figure 8: Screenshot of view weekly report (Staff dashboard)*

The staff generates the weekly report, which is displayed in the grid and the admin can also view it in the admin dashboard. The visiting day, total visitors and the total earning is included in the weekly report. The weekly report provides an overview of the total earnings based on the weekdays and the number of visitors on a certain day.

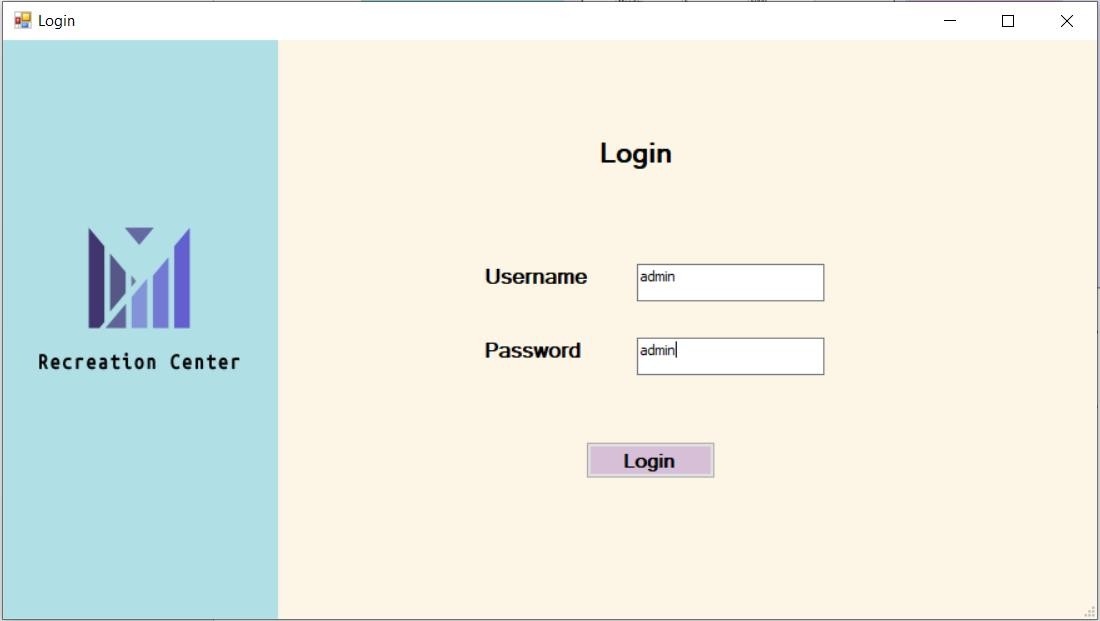
#### View Weekly Chart



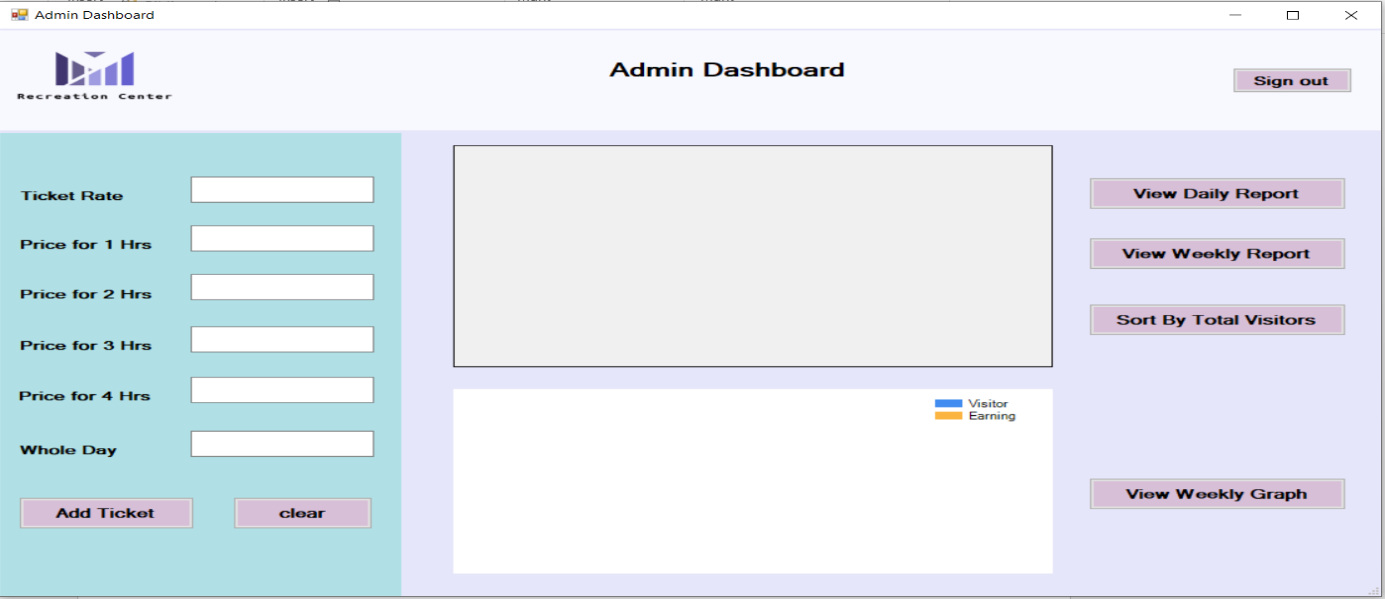
*Figure 9: Screenshot of view weekly graph (Staff dashboard)*

As the staff generates the weekly report and displays it in the grid then the staff can view the weekly graph. The admin can also view it in the admin dashboard. The weekly graph is a graphical representation of the weekly report. The yellow bar in figure 9 represents total earnings, while the blue bar represents total visitors.

#### Admin Login



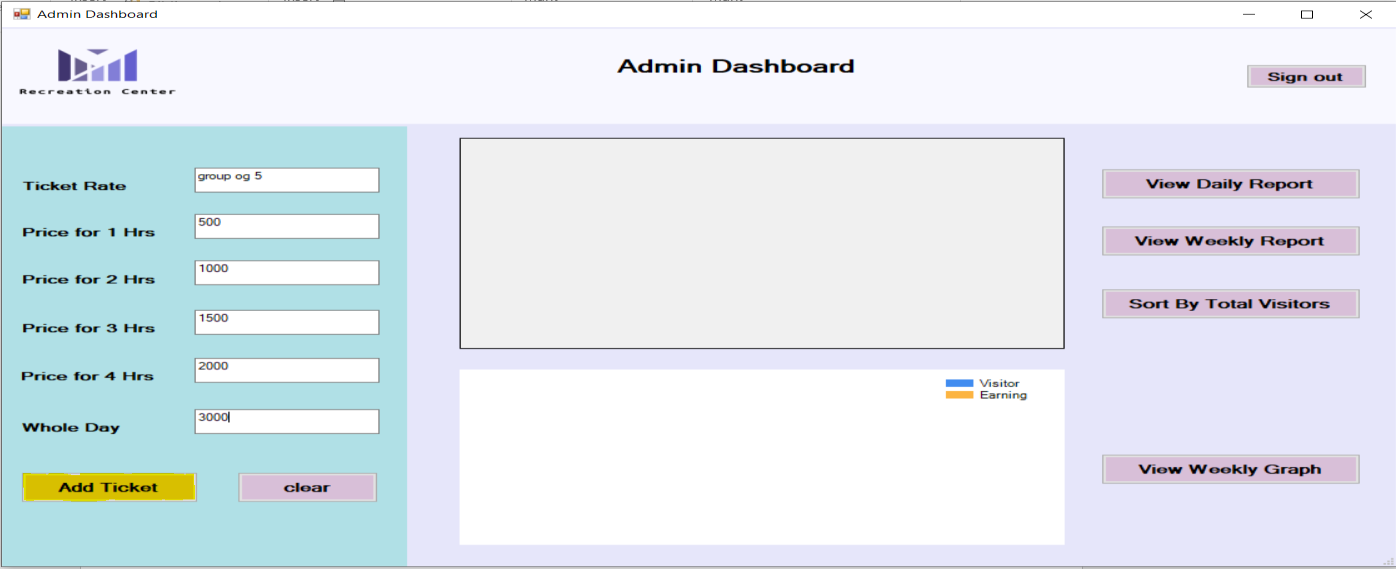
*Figure 10: Login of Admin*



*Figure 11: Admin dashboard*

When the Admin enters the specified username and password, they will be presented with the admin dashboard, as shown in figure 11, where admin can add ticket rate according to the hours or whole day, view daily and weekly reports, and view weekly graph. When an admin clicks the sign-out button, they are redirected to the login form.

#### Add Ticket

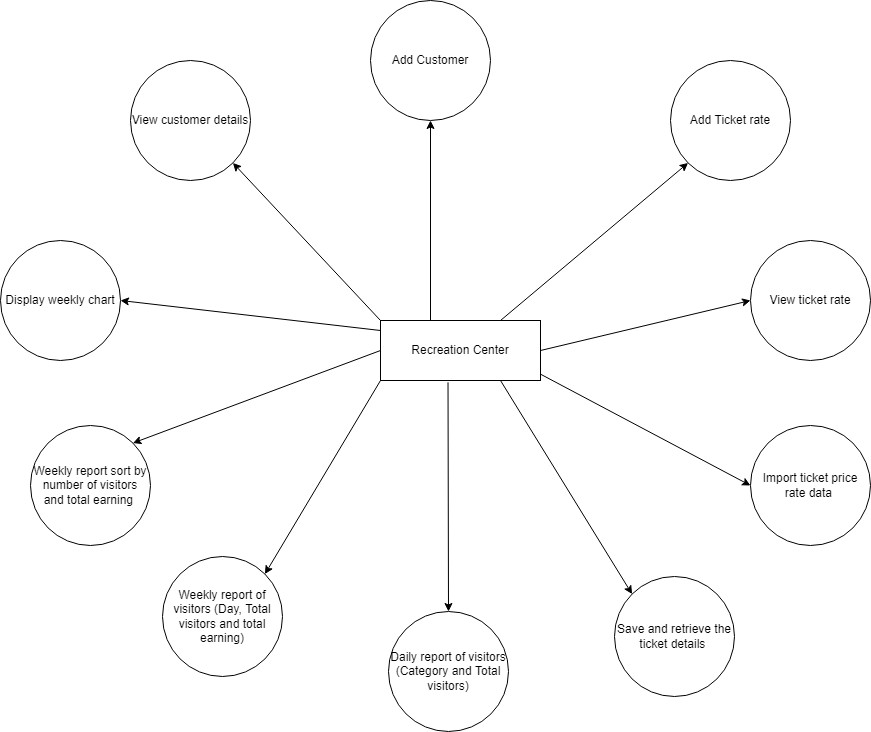


*Figure 12: Screenshot of adding ticket*

The admin determines the ticket rate, which is visible in the staff dashboard. After entering all of the ticket's values, the add ticket button is pressed, and the ticket rate is saved. Similarly, admin can view the daily and weekly reports and weekly graph in the same way that staff can

# System Architecture

System Architecture is defined as a conceptual model that describes the structural design of software that automates work. It is usually a series of diagrams that illustrate services, components, layers, and interactions. It is architecture description language that helps to describe the entire system architecture. (GeeksforGeeks, 2022)



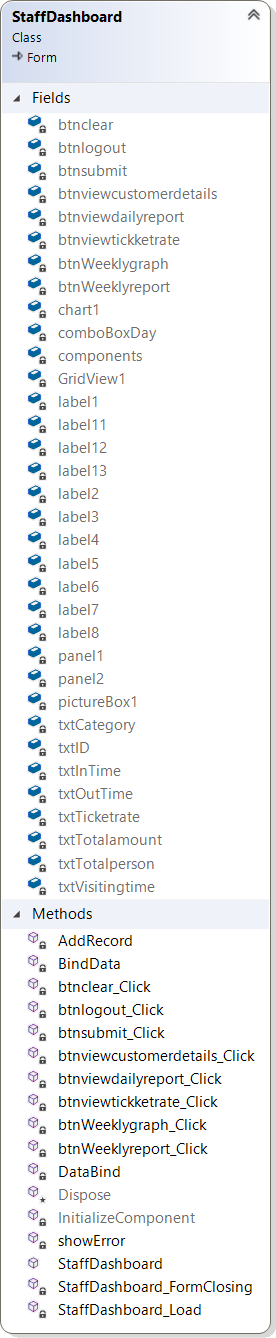
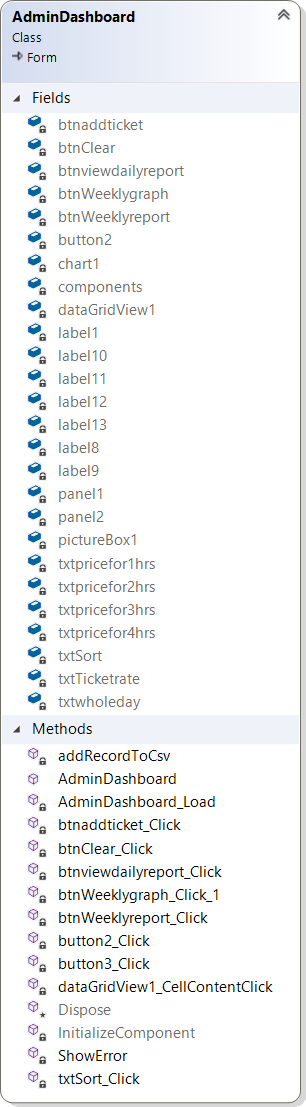
*Figure 13: System Architecture design*

Figure 13 depicts a conceptual view of the developed program recreation center. It depicts the system's interaction with the admin and the staff, in which the staff can add customers and view their details, generate weekly and daily reports, and import the ticket rate set by the admin. The admin can view the daily and weekly reports, as well as sort the weekly report by the number of visitors. They are also able to view the weekly graph. The staff can enter customer information based on the category.

# Class Diagram

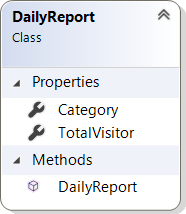
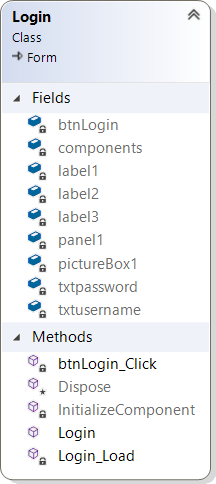
A static view of an application or a diagram that describes the different types of relationship that exists among them and the types of objects in a system is called a class diagram. It can run with almost all Object-Oriented Methods. Before studying the actual code we can overview how the application is structured with the help of a class diagram. (Guru99, 2021)

* 1. **Class diagram of AdminDashboard class 4.2. Class diagram of StaffDashborad class**



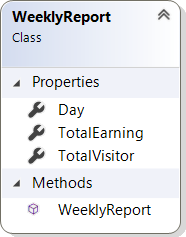
*Figure 14: Class diagram of admin dashboard class and staff dashboard class*

* 1. **Class diagram of Login class 4.3. Class diagram of Daily report**



*Figure 15: Class diagram of login class and daily report class*

#### 4.4 Class diagram of weekly report



*Figure 16: Class diagram of the weekly report class*

# Classes and their purpose

The application is made up of three window forms in total (Login, Admin dashboard, and Staff dashboard). There are a total of eight classes, each with its own purpose, which is described below.

#### Purpose of Login class

|  |  |  |
| --- | --- | --- |
| **Class** | **IDE/Self made** | **Purpose** |
| Login.cs | IDE | The purpose of this class is to allow  admin and staff to login into their respective dashboard |
| Login.cs[Design] | Self-made | It is a login form designed to take the username and password of staff and  admin to login |

*Table 1: purpose of Login class*

#### Purpose of Staff Dashboard class

|  |  |  |
| --- | --- | --- |
| **Class** | **IDE/Self made** | **Purpose** |
| StaffDashboard.cs | IDE | The purpose of this class is to submit the customer details to create the ticket for visitors. The staff can view the tick rate in the grid that is specified by the admin.  To generate weekly and daily reports.  To view the customers' details. |
| StaffDashboard.cs[Design] | Self-made | It includes buttons for submitting customer entries, viewing ticket rates, viewing customer details, and generating weekly and daily reports and a sign-out button that redirects to  the login page. |

*Table 2: Purpose of Staff Dashboard class*

#### Purpose of Admin Dashboard class

|  |  |  |
| --- | --- | --- |
| **Class** | **IDE/Self made** | **Purpose** |
| AdminDashboard.cs | IDE | The purpose of this class is to add the ticket rate that can be imported to the staff dashboard to enter customer details.  To view the daily report generated by the staff (category and total visitors). To view the weekly report (visiting day, total visitors, and total earning). Weekly report sort by the number of  visitors. |
| AdminDashboard.cs[Design] | Self-made | It includes buttons for adding the ticket rates, viewing the weekly and daily report, sorting by total visitors, and a sign-out button that redirects to  the login page. |

*Table 3: Purpose of Admin Dashboard class*

#### Purpose of Daily Report class

|  |  |  |
| --- | --- | --- |
| **Class** | **IDE/Self made** | **Purpose** |
| DailyReport.cs | Self-made | This class was designed to function  as a datatype, providing total visitors and visitors category for its object. |

*Table 4: Purpose of Daily report class*

#### Purpose of Weekly Report class

|  |  |  |
| --- | --- | --- |
| **Class** | **IDE/Self made** | **Purpose** |
| WeeklyReport.cs | Self-made | This class was designed to work as datatype, providing method to get visisting day, total visitiors and the  total earning for its object. |

*Table 5: Purpose of Weekly report class*

# Method Description

First of all, the method is a section of a program that contains a code or a set of instructions. So, method description is the process that describes or defines every method that is used in a program. It makes it easy for us to understand every method of a program.

#### Method description of Login.cs class

|  |  |  |
| --- | --- | --- |
| S.N | Method Name | Description |
| 1. | private void btnLogin\_Click(object sender, EventArgs e) | This method specifies the user name and password for the staff and admin. The else condition is used if the username and password are both staffs, in which case the staff dashboard is opened. If the username and password are both admins, the admin dashboard is displayed. If the text field is left empty, an error message is displayed, and if the username and password are both incorrect, an error message is  displayed. |

*Table 6: Method description of Login.cs class*

#### Method description of StaffDashboard.cs class

|  |  |  |
| --- | --- | --- |
| **SN** | **Method Name** | **Description** |
| 1. | private void showError(string key) | This method is used to display a  message box with the specified error message. |
| 2. | public StaffDashboard() | This method consists of the initializecomponemt method, which is created and managed automatically by the Windows Forms designer and also defines  everything seen on the form. |
| 3. | private void StaffDashboard\_FormClosing(object sender, FormClosingEventArgs e) | It consists of the exit method, which notifies all message pumps that they must be terminated, and the close method, which closes all application windows once the message has  been processed. |
| 4. | private void btnsubmit\_Click(object sender, EventArgs e) | This method saves the customer details and consists of the validation  for each text field. |
| 5. | private void AddRecord() | This method takes each textfield value as a parameter and passes it through the addRecord method. And displays the message box containing the confirmation  message. |
| 6. | private void btnlogout\_Click(object  sender, EventArgs e) | This method redirects the admin to  the login form again. |

|  |  |  |
| --- | --- | --- |
| 7. | private void btnviewdailyreport\_Click(object sender, EventArgs e) | This method is used to retrieve data from the visitors details .CSV file and add it to a list by creating object of daily report. And then pass it on to  the grid datasource. |
| 8. | private void btnWeeklyreport\_Click(object sender, EventArgs e) | This method is used to retrieve the data from visitors details.CSV file and store it in a list by creating object weekly report. And then pass it on to  the grid datasource. |
| 9. | private void  btnWeeklygraph\_Click(object sender, EventArgs e) | This method is used to transfer the  value from the grid view to the chart and display it in the graph section. |
| 10. | private void btnclear\_Click(object  sender, EventArgs e) | This methods clears all the textfield  and the combo box. |
| 11. | private void  btnviewcustomerdetails\_Click(object sender, EventArgs e) | This method pass the value of the customer |
| 12. | private void btnviewtickketrate\_Click(object sender, EventArgs e) | This method search whether the file exists or not and if the file is found it shows the customers details on the  datagrid |

*Table 7: Method description of StaffDashboard.cs class*

#### Method description of AdminDashboard.cs class

|  |  |  |
| --- | --- | --- |
| **S.N** | **Method Name** | **Description** |
| **1.** | public AdminDashboard() | This method consists of the initializecomponemt method, which is created and managed automatically by the Windows Forms designer and also defines  everything seen on the form. |
| **2.** | private void ShowError(string key) | This method is used to display a  message box with the specified error message. |
| **3.** | private void btnlogout\_Click(object  sender, EventArgs e) | This method redirects the admin to  the login form again. |
| **4.** | private void btnaddticket\_Click(object sender, EventArgs e) | This method is used to extract the ticket price from the text file and validate it before passing it as a parameter to the addRecordTocsv  function. |
| **5.** | private void addRecordToCsv() | This method takes each textfield value as a parameter and passes it through the addRecord method. And displays the message box containing the confirmation  message. |
| **6.** | private void btnClear\_Click(object  sender, EventArgs e) | This method is used to clear all the  textfield. |
| **7.** | private void btnviewdailyreport\_Click(object sender, EventArgs e) | This method is used to retrieve data from the visitors details .CSV file and add it to a list by creating object of daily report. And then pass it on to  the grid datasource. |

|  |  |  |
| --- | --- | --- |
| **8.** | private void txtSort\_Click(object sender, EventArgs e) | This method is used to retrieve data from the visitors details. CSV file into a list, sort the list using the bubble sort algorithm, and pass the value to  the grid view. |
| **9.** | private void btnWeeklyreport\_Click(object sender, EventArgs e) | This method is used to retrieve the data from visitors details.CSV file and store it in a list by creating object weekly report. And then pass it on to  the grid datasource. |
| **10.** | private void  btnWeeklygraph\_Click\_1(object sender, EventArgs e) | This method is used to transfer the  value from the grid view to the chart and display it in the graph section. |

*Table 8: Method description of AdminDashboard.cs class*

#### Method description of WeeklyReport.cs class

|  |  |  |
| --- | --- | --- |
| S.N | Method Name | Description |
| 1. | public WeeklyReport(string day, string totalVisitor, string  totalEarning) | It Initialize the visiting day, total visitors and the total earning of the  WeeklyReport class. |

*Table 9: Method description of WeeklyReport.cs class*

#### Method description of DailyReport.cs class

|  |  |  |
| --- | --- | --- |
| S.N | Method Name | Description |
| 1. | public DailyReport(string  category, string totalVisitor) | It Initialize the category and the total  visitors of the DailyReport class. |

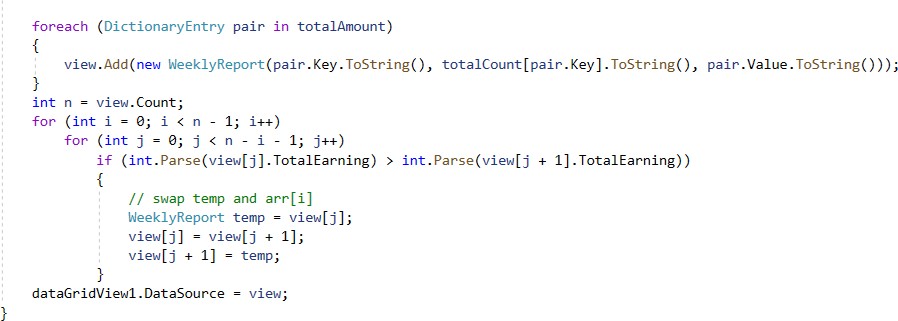
*Table 10: Method description of DailyReport.cs class*

# Data Structure and Algorithm

## Data Structure

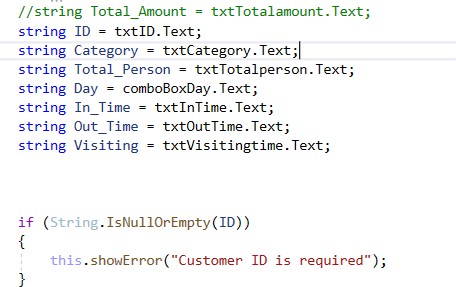
The term "data structure" refers to a specialized format for organizing, processing, retrieving, and storing data. They serve as the foundation for more sophisticated applications, and there are several basic and advanced types of data structures, all of which are designed to arrange data for a specific purpose. It enables users to easily access and work with the data they require inappropriate ways. It logically connects data elements and enables effective data use, persistence, and sharing. (Sarah Lewis, David Loshin, 2022)The following data structures were used in the development of this system:

* **Array:** A collection of items stored in contiguous memory locations is referred to as an array. Items of the same type are stored together so that the position of each element can be easily calculated or retrieved by an index, and the length can be fixed or flexible. This simplifies the calculation of each element's position by simply adding an offset to a base value.



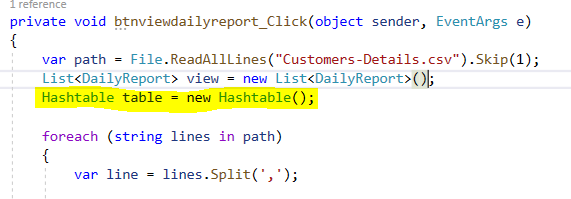
*Figure 17: Screenshot of array data structure used*

* **String:** A string is a data structure that consists of an array of characters followed by a stop code. It is defined as a character array. A "0" value is typically managed using the length field, which is an integer value.



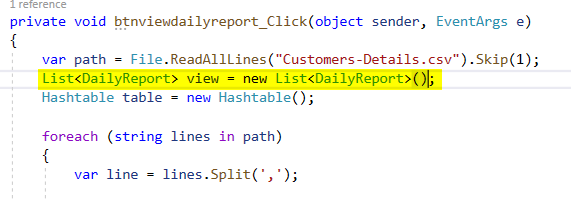
*Figure 18: Screenshot of string data structure used*

* **Hash table:** A hash table, also known as a hash map, is a data structure that stores a collection of items in an associative array that maps keys to values. It converts an index into an array of buckets containing the desired data item using a hash function.



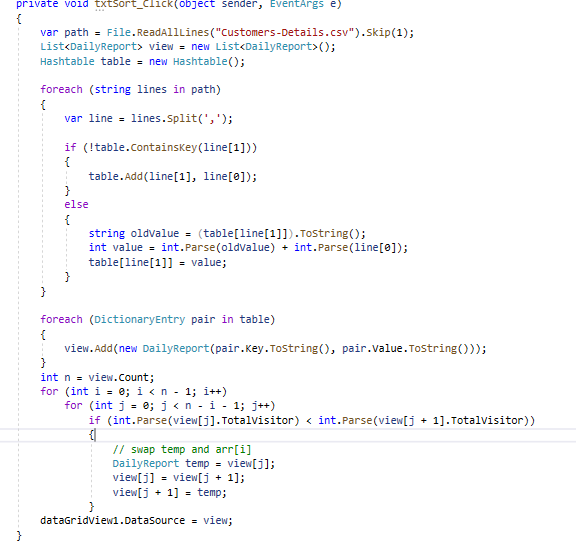
*Figure 19: Screenshot of hash table data structure used*

* **List:** List is defined as a generic collection that is used to store elements or objects in the form of a list, and it is defined as follows. It accepts null as a valid reference type value and allows duplicate elements. The elements in the list are not sorted by default, and they are accessed using the zero index.



## Algorithm

*Figure 20: Screenshot of list data structure used*

To sort the daily report by the total number of visitors, the bubble sort algorithm is used. The bubble sort algorithm is a simple comparison-based sorting algorithm that compares each pair of adjacent elements and swaps them if they are not in order. The Bubble Sort works by iterating through an array from first to the last element, comparing each pair of elements, and switching their positions as needed. This procedure is repeated as many times as required until the array is sorted. (C#Corner, 2022)

*Figure 21: Screenshot of bubble sort algorithm used*

# Reflection of the learning experience

The Recreation Center system was built with C# as the programming language and Visual Studio as the IDE. The entire program was new because it was my first time working on such a project. In the earlier stages, I found it difficult to become acquainted with both Visual Studio and the programming language. However, with more practice and time, the problem was eventually eradicated. The lecture slides and the tutorials projects were very beneficial to overcome many problems. From little knowledge at the beginning of the project to more knowledge at the end, the overall experience was good. During the development of this system, I discovered that Visual Studio was the user-interactive IDE that provided every aspect required during the system's development. It was simple to use and had an appealing user interface, with the drag and drop feature being the standout feature. It also provides an interface for desktop and console applications, window form applications, and many other features on the C#.net framework. Following the drag and drop, it automatically creates the methods on the other side. I consider Visual Studio to be one of the best IDEs I've used to date.

This was the first C# project that I was working on. But after some time I found it quite similar to java. The syntax, event-handling method, and the flow were similar which helped me a lot to learn C#. The data structure and the algorithm were already studied, so I did not face many problems there. The entire project was a fantastic learning experience for me because I learned so much from it. However, completing this project was not an easy task. Throughout the coding process, I encountered numerous difficulties. Nevertheless, in the end, I learned a lot of new things that I didn't know before. I'd learned how to deal with errors and exceptions.

Through a lot of errors, trials, effort, and research the project was completed and the overall experience was good.

# Conclusion

This is the first coursework of this module that accounts for 30% of the module grade which focuses on developing a system for the recreation center. At the end of this project, a fully-functional desktop application was created.

## Evaluation of work

It was a difficult challenge to complete this coursework. Several issues arise during the mission's completion. Several mistakes were made during the design process. Despite the challenges, the project was finished on time. The primary goal of this project was to develop a window-based application that assists a recreation center in managing visitor records. So a lot of things had to be done to finish this module. It is not only about creating documentation. Along with documentation, the development phase was also required. The project began with the coding portion. During the development phase, three window forms (Login, Admin dashboard, and Staff dashboard) were created. Staff and admin can use the login form to access their respective dashboards. The staff enters the customer's information and generates the ticket, which can be viewed in the grid. The ticket rate specified by the admin is also visible to the staff. The weekly and daily reports are created by the staff. In addition, the admin can enter the ticket rate and view the weekly and daily reports generated by the staff. The admin can also see the weekly graph. And the documentation section includes a brief description of the system build, a user manual with detailed instructions on how to run the program, a class diagram with all of the fields and methods, a table with all of the classes and their purposes, and a method description for each method used in the development of this project, data structure, and the algorithm were described.

## Learning Outcomes

At the end of this project, I learned a lot of new things that I didn't know before. During the system's development, I was able to gain extensive knowledge of the C# programming language as well as the.Net framework. Now, I have a clear vision of developing a desktop-based application. Once this system is completed, I will be able to create an effective and ready-to-use desktop on my own. I learned how to handle

errors and exceptions, as well as the various data structures. I now have a clear picture of the bubble sort algorithm because it was used.

## Difficulties encountered and it's overcome

Completing this project was a difficult task because severe issues arise throughout the development phase. After all, it was my first C# project. Several mistakes were made during the design process. Completing the overall project provided practical experience with the C# programming language. At first, I had a lot of problems, so I consulted the teacher about it, researched it on some reliable websites, and completed the project. The lecture slides were also extremely useful in completing this project. Yes, the assigned coursework was difficult, but it was completed on time and efficiently.

I found that working on this project provided me with a wealth of information that will come in handy in the near future.

# References

Barbara Thompson, 2021. *What is .NET Framework? Explain Architecture & Components.* [Online]

Available at: https:/[/www](http://www.guru99.com/net-framework.html).[guru99.com/net-framework.html](http://www.guru99.com/net-framework.html) [Accessed 03 01 2022].

C#Corner, 2022. *Bubble Sort In C#.* [Online]

Available at: https:/[/www.c](http://www.c-sharpcorner.com/UploadFile/3d39b4/bubble-sort-in-C-Sharp/)-[sharpcorner.com/UploadFile/3d39b4/bubble-sort-in-C-Sharp/](http://www.c-sharpcorner.com/UploadFile/3d39b4/bubble-sort-in-C-Sharp/) [Accessed 03 01 2022].

GeeksforGeeks, 2022. *Difference between System Architecture and Software Architecture - GeeksforGeeks.* [Online]

Available at: https:/[/www](http://www.geeksforgeeks.org/difference-between-system-architecture-).[geeksforgeeks.org/difference-between-system-architecture-](http://www.geeksforgeeks.org/difference-between-system-architecture-) and-software-architecture/

[Accessed 03 01 2022].

GeeksforGeeks, 2022. *Introduction to C# - GeeksforGeeks.* [Online] Available at: https:/[/www](http://www.geeksforgeeks.org/introduction-to-c-sharp/).[geeksforgeeks.org/introduction-to-c-sharp/](http://www.geeksforgeeks.org/introduction-to-c-sharp/) [Accessed 03 01 2022].

Guru99, 2021. *UML Class Diagram Tutorial: Abstract Class with Examples.* [Online]

Available at: https:/[/www](http://www.guru99.com/uml-class-diagram.html).[guru99.com/uml-class-diagram.html](http://www.guru99.com/uml-class-diagram.html) [Accessed 03 01 2022].

Sarah Lewis, David Loshin, 2022. *TechTarget.* [Online]

Available at: https://searchsqlserver.techtarget.com/definition/data-structure [Accessed 03 01 2022].

# Appendix

#### Code of Login.cs

using System;

using System.Collections.Generic; using System.ComponentModel; using System.Data;

using System.Drawing; using System.Linq; using System.Text;

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

namespace CourseWork1

{

public partial class Login : Form

{

public Login()

{

InitializeComponent();

}

private void btnLogin\_Click(object sender, EventArgs e)

{

string username = txtusername.Text; string password = txtpassword.Text;

// validation of the text field

if (String.IsNullOrEmpty(username) || String.IsNullOrEmpty(password))

{

MessageBox.Show("Empty textfield found. Please! insert username and

password");

}

else if (username == "staff" && password == "staff") // username and password for staff

{

this.Hide();

StaffDashboard sd = new StaffDashboard(); sd.ShowDialog();

}

else if (username == "admin" && password == "admin") //Username and password for admin

{

}

else

{

}

}

this.Hide();

AdminDashboard ad = new AdminDashboard(); ad.ShowDialog();

MessageBox.Show("Invalid Username or Password"); //Message box

private void Login\_Load(object sender, EventArgs e)

{

}

}

}

#### Code of AdminDashboard.cs

using System;

using System.Collections.Generic; using System.Collections;

using System.ComponentModel; using System.Data;

using System.Drawing; using System.Linq; using System.IO; using System.Text;

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

namespace CourseWork1

{

public partial class AdminDashboard : Form

{

public AdminDashboard()

{

InitializeComponent();

}

private void ShowError(string key)

{

MessageBox.Show(key + "Price Is Not Valid Number Format price is not ");

}

private void button3\_Click(object sender, EventArgs e)

{

MessageBox.Show("Weekly Report");

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void btnaddticket\_Click(object sender, EventArgs e)

{

string TicketRate = (txtTicketrate.Text); double price1Hr = 0.00;

double price2Hr = 0.00; double price3Hr = 0.00; double price4Hr = 0.00; double priceUHr = 0.00;

object[] TicketData = new object[6];

//validation of the text field

if (string.IsNullOrEmpty(TicketRate))

{

MessageBox.Show("Please, Insert the ticket rate");

}

else if (!double.TryParse(txtpricefor1hrs.Text, out price1Hr))

{

this.ShowError("1 Hrs");

}

else if (!double.TryParse(txtpricefor2hrs.Text, out price2Hr))

{

this.ShowError("2 Hrs");

}

else if (!double.TryParse(txtpricefor3hrs.Text, out price3Hr))

{

this.ShowError("3 Hrs");

}

else if (!double.TryParse(txtpricefor4hrs.Text, out price4Hr))

{

this.ShowError("4 Hrs");

}

else if (!double.TryParse(txtwholeday.Text, out priceUHr))

{

}

else

{

this.ShowError("Whole Day");

this.addRecordToCsv(TicketRate, price1Hr, price2Hr, price3Hr,

price4Hr, priceUHr); // adding record to the csv

}

}

// method to show the data from csv to the grid

private void addRecordToCsv(string rate, double price1hr, double price2hr, double price3hr, double price4hr, double priceuhr)

{

string filePath = "Ticket-Rate.csv";

try

{

if (!File.Exists(filePath))

{

string clientHeader = $"\"Ticket Rate\",\"1 Hrs\",\"2 Hrs\",\"3 Hrs\",\"4 Hrs\",\"Whole Day\"{Environment.NewLine}";

File.WriteAllText(filePath, clientHeader);

}

using (System.IO.StreamWriter file = new System.IO.StreamWriter(filePath, true))

{

file.WriteLine(rate + "," + price1hr + "," + price2hr + "," + price3hr + "," + price4hr + "," + priceuhr);

txtTicketrate.Text = ""; txtpricefor1hrs.Text = ""; txtpricefor2hrs.Text = ""; txtpricefor3hrs.Text = ""; txtpricefor4hrs.Text = ""; txtwholeday.Text = "";

MessageBox.Show("Ticket rate is added");

}

}

catch (Exception ex)

{

throw new ApplicationException("WRONG WRONG WRONG", ex); // Exception

}

}

// Clears the tesxt field

private void btnClear\_Click(object sender, EventArgs e)

{

txtTicketrate.Text = ""; txtpricefor1hrs.Text = ""; txtpricefor2hrs.Text = ""; txtpricefor3hrs.Text = ""; txtpricefor4hrs.Text = ""; txtwholeday.Text = "";

}

// retrieve data from the visitors details .CSV file

// add it to a list by creating object of daily report

private void btnviewdailyreport\_Click(object sender, EventArgs e)

{

var path = File.ReadAllLines("Visitors-Details.csv").Skip(1); List<DailyReport> view = new List<DailyReport>();

Hashtable table = new Hashtable();

foreach (string lines in path)

{

var line = lines.Split(',');

if (!table.ContainsKey(line[1]))

{

table.Add(line[1], line[2]);

}

else

{

string oldValue = (table[line[1]]).ToString();

int value = int.Parse(oldValue) + int.Parse(line[2]); table[line[1]] = value;

}

}

foreach (DictionaryEntry pair in table)

{

view.Add(new DailyReport(pair.Key.ToString(), pair.Value.ToString()));

}

dataGridView1.DataSource = view; //pass it on to the grid datasource.

}

// Retrives the data from the visitors details. CSV file into a list private void txtSort\_Click(object sender, EventArgs e)

{

var path = File.ReadAllLines("Visitors-Details.csv").Skip(1); List<DailyReport> view = new List<DailyReport>();

Hashtable table = new Hashtable();

foreach (string lines in path)

{

var line = lines.Split(',');

if (!table.ContainsKey(line[1]))

{

table.Add(line[1], line[0]);

}

else

{

string oldValue = (table[line[1]]).ToString();

int value = int.Parse(oldValue) + int.Parse(line[0]); table[line[1]] = value;

}

}

foreach (DictionaryEntry pair in table)

{

view.Add(new DailyReport(pair.Key.ToString(), pair.Value.ToString()));

}

// sort the list using the bubble sort algorithm, and pass the value to the grid view.

int n = view.Count;

for (int i = 0; i < n - 1; i++)

for (int j = 0; j < n - i - 1; j++)

if (int.Parse(view[j].TotalVisitor) < int.Parse(view[j +

1].TotalVisitor))

{

// swapping the temp and arr[i] DailyReport temp = view[j]; view[j] = view[j + 1];

view[j + 1] = temp;

}

dataGridView1.DataSource = view;

}

private void btnWeeklyreport\_Click(object sender, EventArgs e)

{

// Retrives the data from Visitors-Details.csv file

var path = File.ReadAllLines("Visitors-Details.csv").Skip(1); List<WeeklyReport> view = new List<WeeklyReport>();

Hashtable totalCount = new Hashtable(); Hashtable totalAmount = new Hashtable();

foreach (string lines in path)

{

var line = lines.Split(',');

if (!totalCount.ContainsKey(line[3]) &&

!totalAmount.ContainsKey(line[3]))

{

totalCount.Add(line[3], line[2]); totalAmount.Add(line[3], line[8]);

}

else

{

string oldCount = totalCount[line[3]].ToString(); string oldAmount = totalAmount[line[3]].ToString();

int newCount = int.Parse(oldCount) + int.Parse(line[2]); int newAmount = int.Parse(oldAmount) + int.Parse(line[8]);

totalCount[line[3]] = newCount; totalAmount[line[3]] = newAmount;

}

}

foreach (DictionaryEntry pair in totalAmount)

{

view.Add(new WeeklyReport(pair.Key.ToString(), totalCount[pair.Key].ToString(), pair.Value.ToString()));

}

int n = view.Count;

for (int i = 0; i < n - 1; i++)

for (int j = 0; j < n - i - 1; j++)

if (int.Parse(view[j].TotalEarning) > int.Parse(view[j +

1].TotalEarning))

{

// swap temp and arr[i] WeeklyReport temp = view[j]; view[j] = view[j + 1]; view[j + 1] = temp;

}

dataGridView1.DataSource = view;

}

private void AdminDashboard\_Load(object sender, EventArgs e)

{

}

// Transfesr the value from the grid view to the chart and display it in the graph section

private void btnWeeklygraph\_Click\_1(object sender, EventArgs e)

{

int rowCount = dataGridView1.RowCount; string day; double c1 = 0, c2 = 0; for (int i = 0; i < rowCount; i++)

{

day = (dataGridView1.Rows[i].Cells[0].Value).ToString();

c1 = Convert.ToDouble(dataGridView1.Rows[i].Cells[1].Value); c2 = Convert.ToDouble(dataGridView1.Rows[i].Cells[2].Value);

chart1.Series["Visitor"].Points.AddXY(day, c1); chart1.Series["Earning"].Points.Add(c2);

}

}

// Logout from the admin dashboard

private void btnlogout\_Click(object sender, EventArgs e)

{

this.Hide();

Login l = new Login(); l.ShowDialog();

}

}

}

#### Code of StaffDashboard.cs

using System;

using System.Collections.Generic; using System.Collections;

using System.ComponentModel; using System.Data;

using System.IO;

using System.Drawing; using System.Linq; using System.Text;

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

namespace CourseWork1

{

public partial class StaffDashboard : Form

{

public StaffDashboard()

{

InitializeComponent();

}

private void StaffDashboard\_FormClosing(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

private void showError(string key)

{

MessageBox.Show(key);

}

private void StaffDashboard\_Load(object sender, EventArgs e)

{

}

private void btnsubmit\_Click(object sender, EventArgs e)

{

int i;

int Total\_Amount = 0; int Ticket\_Rate = 0;

//string Total\_Amount = txtTotalamount.Text; string ID = txtID.Text;

string Category = txtCategory.Text;

string Total\_Person = txtTotalperson.Text; string Day = comboBoxDay.Text;

string In\_Time = txtInTime.Text; string Out\_Time = txtOutTime.Text; string Visiting = txtVisitingtime.Text;

//Validation of text field

if (String.IsNullOrEmpty(ID))

{

this.showError("Customer ID is required");

}

else if (!int.TryParse(txtID.Text, out i))

{

MessageBox.Show("ID consists of int value only"); txtID.Focus();

}

else if (String.IsNullOrEmpty(Category))

{

this.showError("Category of customer is Required");

}

else if (String.IsNullOrEmpty(Total\_Person))

{

this.showError("Total number of customer is Required");

}

else if (!int.TryParse(txtTotalperson.Text, out i))

{

MessageBox.Show("Total person consists of int value only"); txtID.Focus();

}

else if (comboBoxDay.Text == "")

{

MessageBox.Show("please enter day");

}

else if (String.IsNullOrEmpty(In\_Time))

{

this.showError("In time of the customer is required");

}

else if (String.IsNullOrEmpty(Out\_Time))

{

this.showError("Out time of the customer is required");

}

else if (String.IsNullOrEmpty(Visiting))

{

this.showError("Visiting Time of customer is Required");

}

else if (!int.TryParse(txtTicketrate.Text, out Ticket\_Rate))

{

this.showError("Ticket Rate is Required or is invalid number Format");

}

else if (!int.TryParse(txtTotalamount.Text, out Total\_Amount))

{

this.showError("Total Price is Required or is invalid number Format");

}

else

{

this.AddRecord(ID, Category, Total\_Person, Day, In\_Time, Out\_Time, Visiting, Ticket\_Rate, Total\_Amount); // adding record to the csv

}

}

// method to show the data from csv to the grid

private void AddRecord(string ID, string visitorCategory, string Totalperson, string Day, string InTime, string OutTime, string visithour, int TicketRate, int TotalPrice)

{

string filePath = "Visitors-Details.csv";

try

{

if (!File.Exists(filePath))

{

string clientHeader = $"\"Customer ID\",\"Visitor Category\",\"Total Person\",\"Visiting Day\",\"In Time\",\"Out Time\",\"Total Visit Time\",\"Ticket Rate\",\"Total Amount\"{Environment.NewLine}";

File.WriteAllText(filePath, clientHeader);

}

using (System.IO.StreamWriter file = new System.IO.StreamWriter(filePath, true))

{

file.WriteLine(ID + "," + visitorCategory + "," + Totalperson + "," + Day + "," + InTime + ","

+ OutTime + "," + visithour + "," + TicketRate + "," + TotalPrice + ","); txtID.Text = "";

txtTotalperson.Text = ""; txtCategory.Text = ""; comboBoxDay.Text = ""; txtInTime.Text = ""; txtOutTime.Text = ""; txtVisitingtime.Text = ""; txtTicketrate.Text = ""; txtTotalamount.Text = "";

MessageBox.Show("Ticket is generated");

}

}

catch (Exception ex)

{

throw new ApplicationException("WRONG WRONG WRONG", ex); //Exception

}

}

// Logout from the admin dashboard

private void btnlogout\_Click(object sender, EventArgs e)

{

this.Hide();

Login login = new Login(); login.ShowDialog();

}

// retrieve data from the visitors details .CSV file

// add it to a list by creating object of daily report

private void btnviewdailyreport\_Click(object sender, EventArgs e)

{

var path = File.ReadAllLines("Visitors-Details.csv").Skip(1); List<DailyReport> view = new List<DailyReport>(); Hashtable table = new Hashtable();

foreach (string lines in path)

{

var line = lines.Split(',');

if (!table.ContainsKey(line[1]))

{

table.Add(line[1], line[2]);

}

else

{

string oldValue = (table[line[1]]).ToString();

int value = int.Parse(oldValue) + int.Parse(line[2]); table[line[1]] = value;

}

}

foreach (DictionaryEntry pair in table)

{

view.Add(new DailyReport(pair.Key.ToString(), pair.Value.ToString()));

}

GridView1.DataSource = view;

}

private void btnWeeklyreport\_Click(object sender, EventArgs e)

{

// Retrives the data from Visitors-Details.csv file

var path = File.ReadAllLines("Visitors-Details.csv").Skip(1); List<WeeklyReport> view = new List<WeeklyReport>(); Hashtable totalCount = new Hashtable();

Hashtable totalAmount = new Hashtable();

foreach (string lines in path)

{

var line = lines.Split(',');

if (!totalCount.ContainsKey(line[3]) && !totalAmount.ContainsKey(line[3]))

{

totalCount.Add(line[3], line[2]); totalAmount.Add(line[3], line[8]);

}

else

{

string oldCount = totalCount[line[3]].ToString(); string oldAmount = totalAmount[line[3]].ToString();

int newCount = int.Parse(oldCount) + int.Parse(line[2]);

int newAmount = int.Parse(oldAmount) + int.Parse(line[8]);

totalCount[line[3]] = newCount; totalAmount[line[3]] = newAmount;

}

}

foreach (DictionaryEntry pair in totalAmount)

{

view.Add(new WeeklyReport(pair.Key.ToString(), totalCount[pair.Key].ToString(), pair.Value.ToString()));

}

int n = view.Count;

for (int i = 0; i < n - 1; i++)

for (int j = 0; j < n - i - 1; j++)

if (int.Parse(view[j].TotalEarning) > int.Parse(view[j + 1].TotalEarning))

{

// swap temp and arr[i] WeeklyReport temp = view[j]; view[j] = view[j + 1];

view[j + 1] = temp;

}

GridView1.DataSource = view;

}

// Transfesr the value from the grid view to the chart and display it in the graph section private void btnWeeklygraph\_Click(object sender, EventArgs e)

{

int rowCount = GridView1.RowCount; string day; double c1 = 0, c2 = 0;

for (int i = 0; i < rowCount; i++)

{

day = (GridView1.Rows[i].Cells[0].Value).ToString();

c1 = Convert.ToDouble(GridView1.Rows[i].Cells[1].Value); c2 = Convert.ToDouble(GridView1.Rows[i].Cells[2].Value);

chart1.Series["Visitor"].Points.AddXY(day, c1); chart1.Series["Earning"].Points.Add(c2);

}

}

private void btnclear\_Click(object sender, EventArgs e)

{

txtID.Text = ""; txtTotalperson.Text = ""; txtCategory.Text = ""; comboBoxDay.SelectedItem = null; txtInTime.Text = "";

txtOutTime.Text = ""; txtVisitingtime.Text = ""; txtTotalamount.Text = ""; txtTicketrate.Text = "";

}

private void btnviewcustomerdetails\_Click(object sender, EventArgs e)

{

BindData("Visitors-Details.csv");

}

private void DataBind(string filePath)

{

DataTable dtable = new DataTable(); string[] lines = File.ReadAllLines(filePath); if (lines.Length > 0)

{

//first line to create header string firstLine = lines[0];

string[] headerLabels = firstLine.Split(','); foreach (string headerWord in headerLabels)

{

dtable.Columns.Add(new DataColumn(headerWord));

}

//For Data

for (int i = 1; i < lines.Length; i++)

{

string[] dataWords = lines[i].Split(','); DataRow dr = dtable.NewRow();

int columnIndex = 0;

foreach (string headerWord in headerLabels)

{

dr[headerWord] = dataWords[columnIndex++];

}

dtable.Rows.Add(dr);

}

}

if (dtable.Rows.Count > 0)

{

GridView1.DataSource = dtable;

}

}

private void btnviewtickketrate\_Click(object sender, EventArgs e)

{

BindData("Ticket-Rate.csv");

}

private void BindData(string filePath)

{

DataTable dtable = new DataTable(); string[] lines = File.ReadAllLines(filePath); if (lines.Length > 0)

{

//first line to create header string firstLine = lines[0];

string[] headerLabels = firstLine.Split(','); foreach (string headerWord in headerLabels)

{

dtable.Columns.Add(new DataColumn(headerWord));

}

//For Data

for (int i = 1; i < lines.Length; i++)

{

string[] dataWords = lines[i].Split(','); DataRow dr = dtable.NewRow();

int columnIndex = 0;

foreach (string headerWord in headerLabels)

{

dr[headerWord] = dataWords[columnIndex++];

}

dtable.Rows.Add(dr);

}

}

if (dtable.Rows.Count > 0)

{

GridView1.DataSource = dtable;

}

}

}

}