LONDON METROPOLITAN UNIVERSITY

ITAHARI INTERNATIONAL COLLEGE

**Module Code & Module Title**

**CS6004NT Application Development**

**Assessment Weightage & Type**

**30% Individual Coursework**

**Semester**

**2020-21 Autumn**

**Student Name: Susan Shrestha**

**London Met ID: 19033540**

**College ID: NP05CP4S200039**

**Assignment Due Date: 03 January 2022**

**Assignment Submission Date: 03 January 2022**

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

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the module tutor of Application Development Mr. Rabi Rouniyar for guiding and motivating me throughout the coursework. Without his kind support and proper guidance, this report would not have been completed perfectly.

At last but not least, I would like to thank Itahari International college **(IIC)** for allowing me to access required resources to prepare this report, conduct research for development of this coursework and for providing me an opportunity to study this course which includes this wonderful module.

## Contents

## Table of Figures



## Table of Tables

# 1. Introduction

A GUI desktop application has been designed and developed in Visual Studio 2019 using C# programming and Windows Forms Application for a recreation center to manage the visitor's records as well as ticket rate for various ticket options.

The application has two different user type access i.e. admin and staff that is used for login in which admin is allowed with full functionality of the software while staff does not get access to set ticket rate. After login, the application preloads an existing record file (if available) containing visitor's details including date, entry and exit time and calculating number of hours they spent. Main functionality such as sorting the records in grid view by name or date, easy navigation to tabs/section, import/save .csv file from local directory, unload data from grid view, log out, add new record, edit selected record, delete selected record, retrieve ticket details for staff, clear form, discount on holidays, generate daily and weekly report chart showing each day's total number of visitors and total earnings, CSV file read/write and admin control for adding or editing ticket option are implemented in the developed application. As well as proper validation of input fields, exception handling, Interface design and usability of the system is given major priority.

During the development part, the programming style is mainly focused and a standard is maintained for proper naming conventions, useful comments, sensible naming while defining variables, classes, properties and methods.

# 2. User Manual

### Instruction

- Run the application using .exe file available in Debug folder of Project folder or using Visual Studio. Also, install the Poppins available in the folder for better user interface.

### Getting Started

After running the application, a login windows form will appear as shown below. For now, only two users for two different user type is available respectively.

| Usertype (Role) | Username | Password |
|---|---|---|
| **Admin** | admin_susan | admin |
| **Staff** | staff_susan | staff |

*Table 1: User Credentials*

### Login

- Enter the login details as given above for the intended user role.
- Select the **usertype** (role).
- Click on **Login** button.

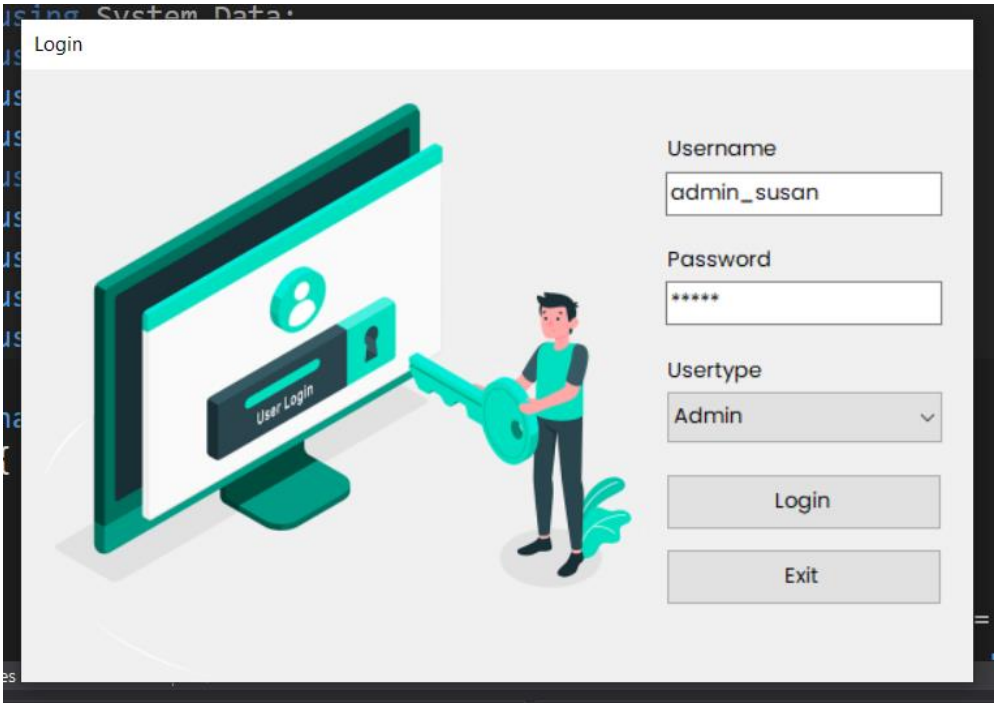



*Figure 1: Login*

**Staff Dashboard**

Log in with staff role redirects to staff dashboard where staff features such as tracking visitor's records and generating daily and weekly reports are available to user. Screenshot of staff dashboard can be seen below. Here, no data will be available in grid view as no files have been saved.

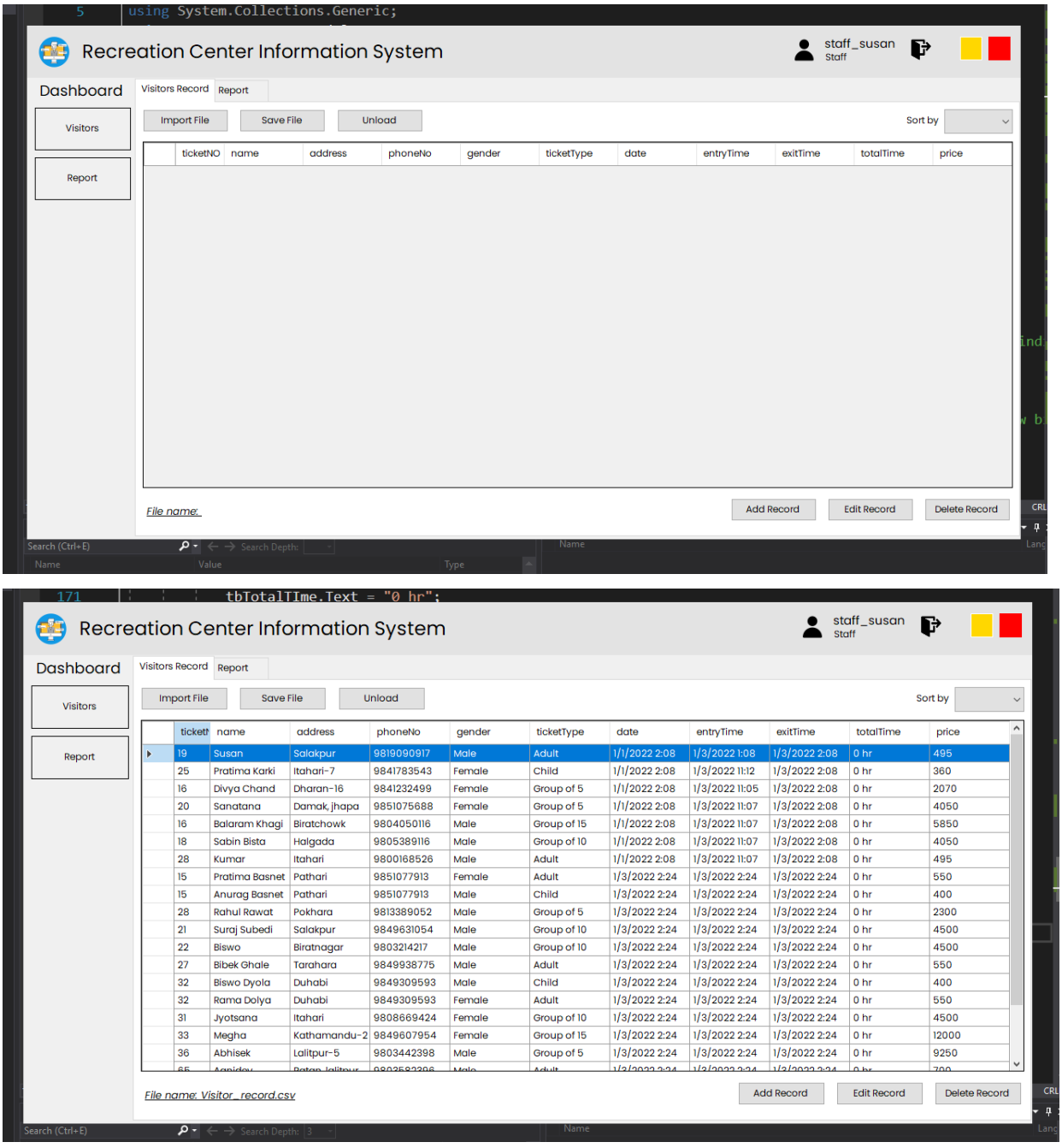



*Figure 2: Staff Dashboard*

**Import file/record**

- Click on Import File button
- Select a valid .csv file.
- Visitor's records will be loaded to the grid view.

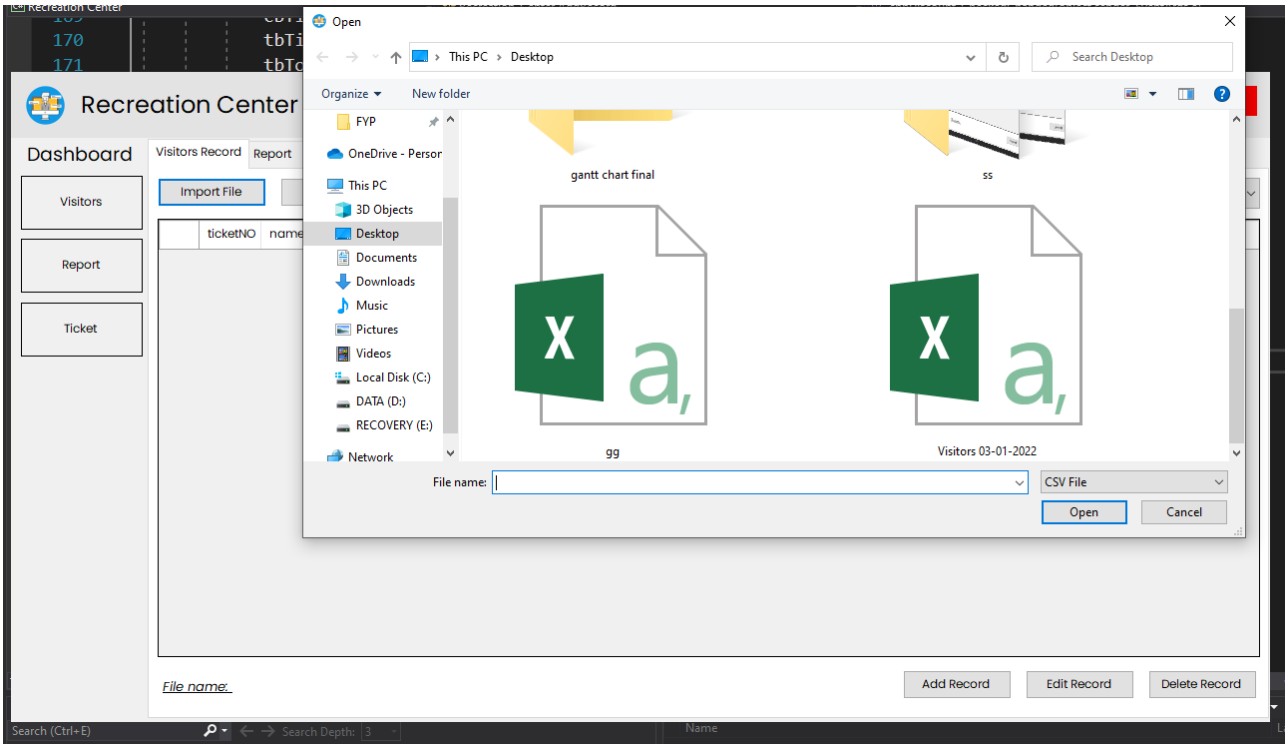



*Figure 3: Import File*

**Save file/record**

- Click on Save File button
- Select location and name the file (optional).
- Click on Save. Records will be saved as new file.

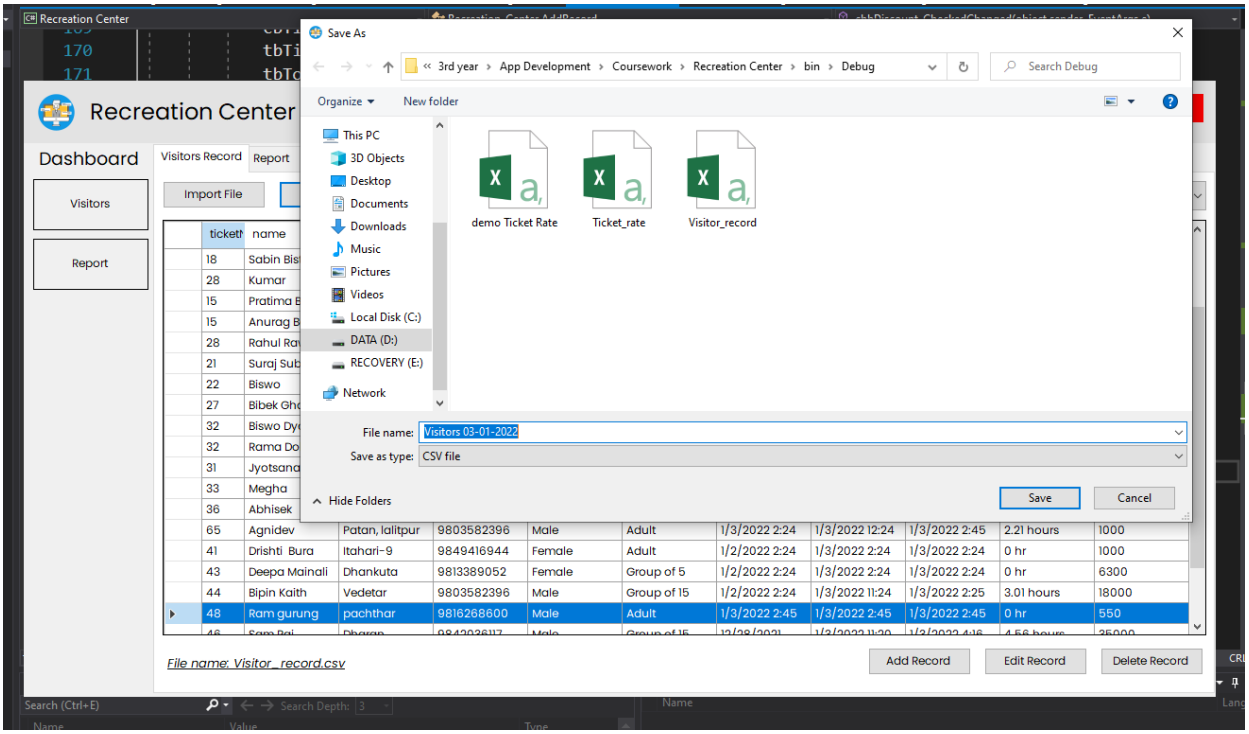



*Figure 4: Save File*

## Unload data from grid view

- Click on **Unload** button.
- Records will be cleared from grid view and new file/record can be imported.

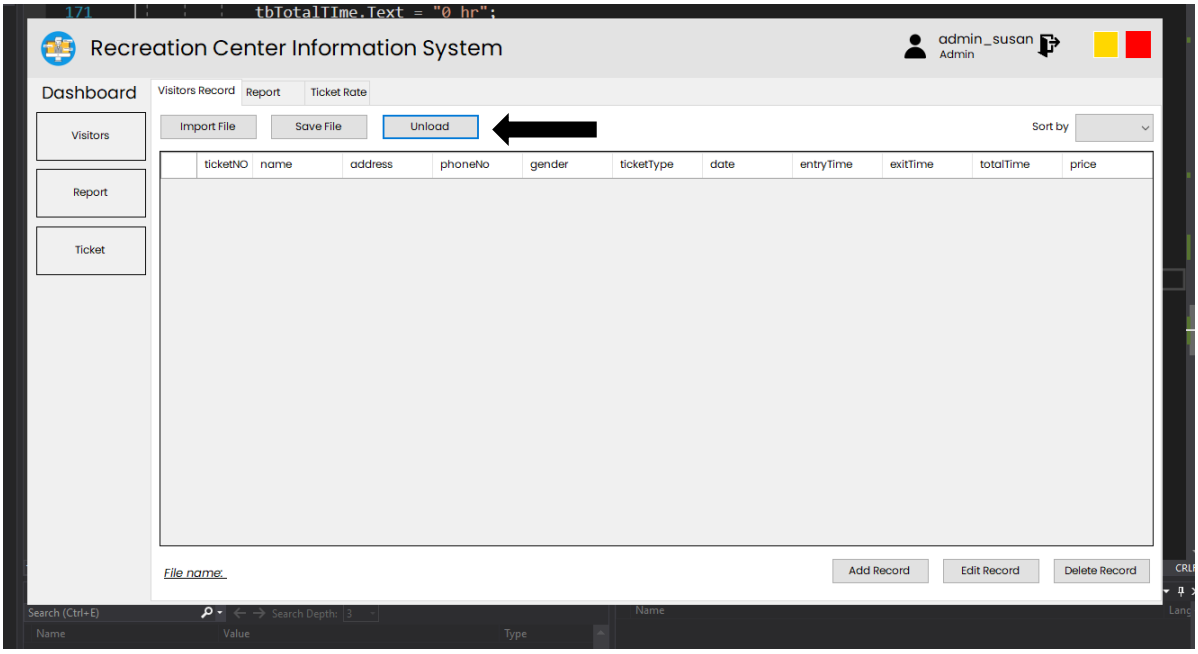



*Figure 5: Unload Data*

## Sort visitor's records

- Navigate to Sort by combo box (dropdown).
- Select the sorting category i.e. Name or Date.
- Records will be sorted and displayed in the grid view.

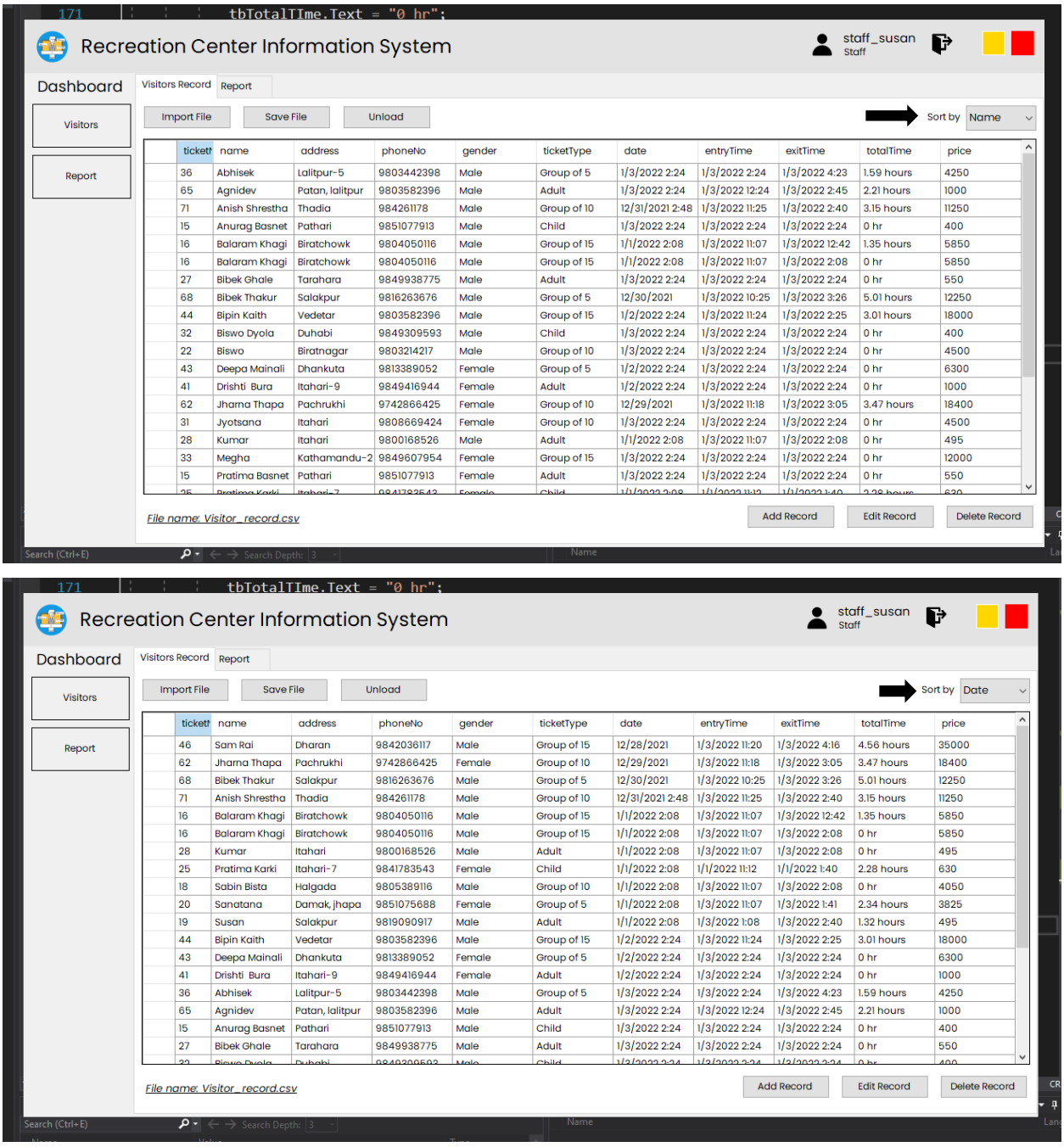






*Figure 6: Sort visitor record by name or date*

**Add new visitor's record**

- Click on Add Record button.
- Fill the required fields denoted by (*) asterisk.
- Check Holiday Discount on holidays to get automatic discount price.
- Ticket price rate list will be available for staff to see present price rate or staff can unload and import a specific ticket price rate file provided by admin.
- After entering the required details, click on Add Record.

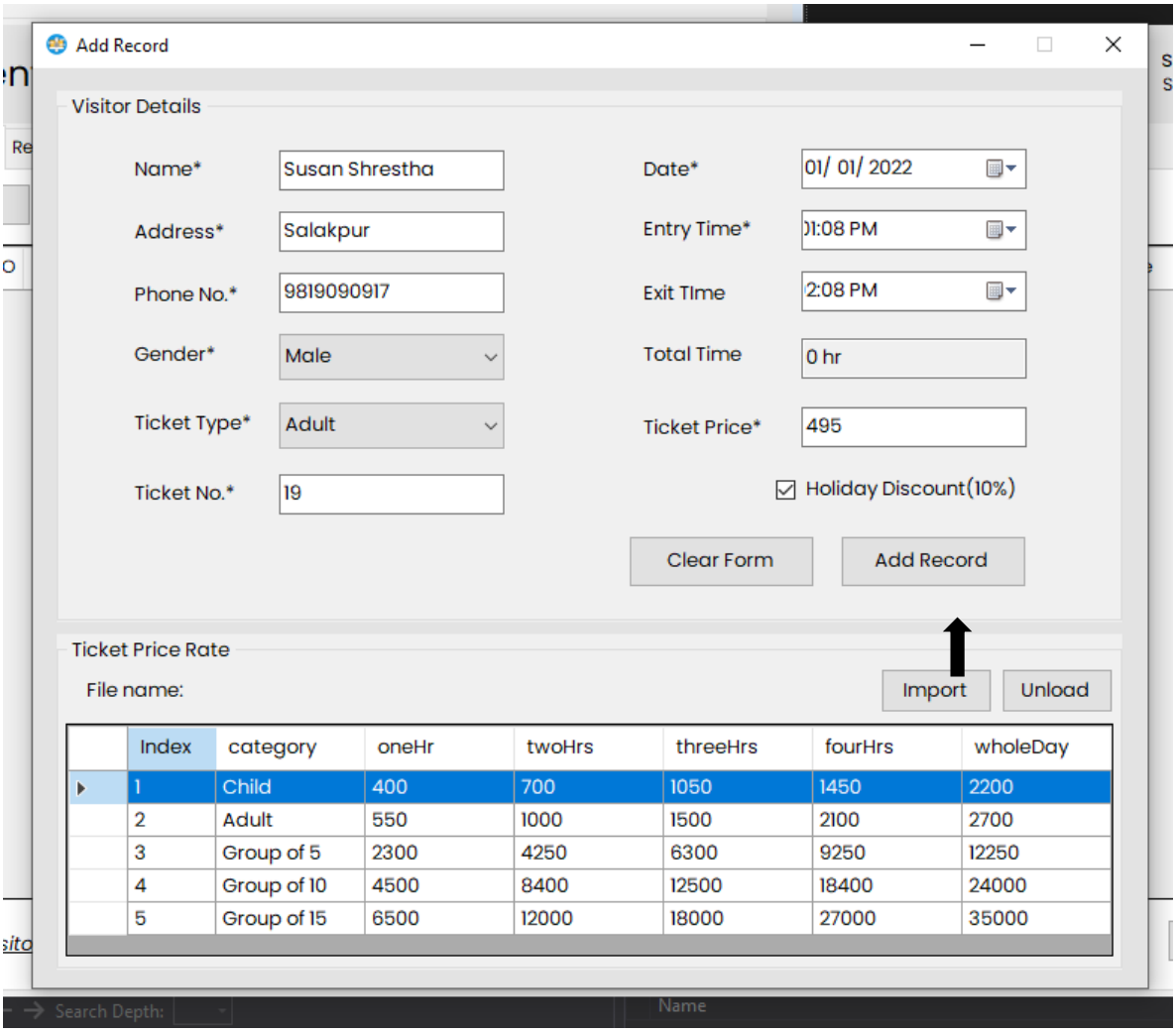



*Figure 7: Add new record*

**Edit visitor's record**

- Select the row from grid view indicated by an arrow.
- Click on Edit Record button.
- Change the exit time of the visitor in order to calculate total time automatically.
- Enter the price according to the hours spent.
- Click on Edit Record button.

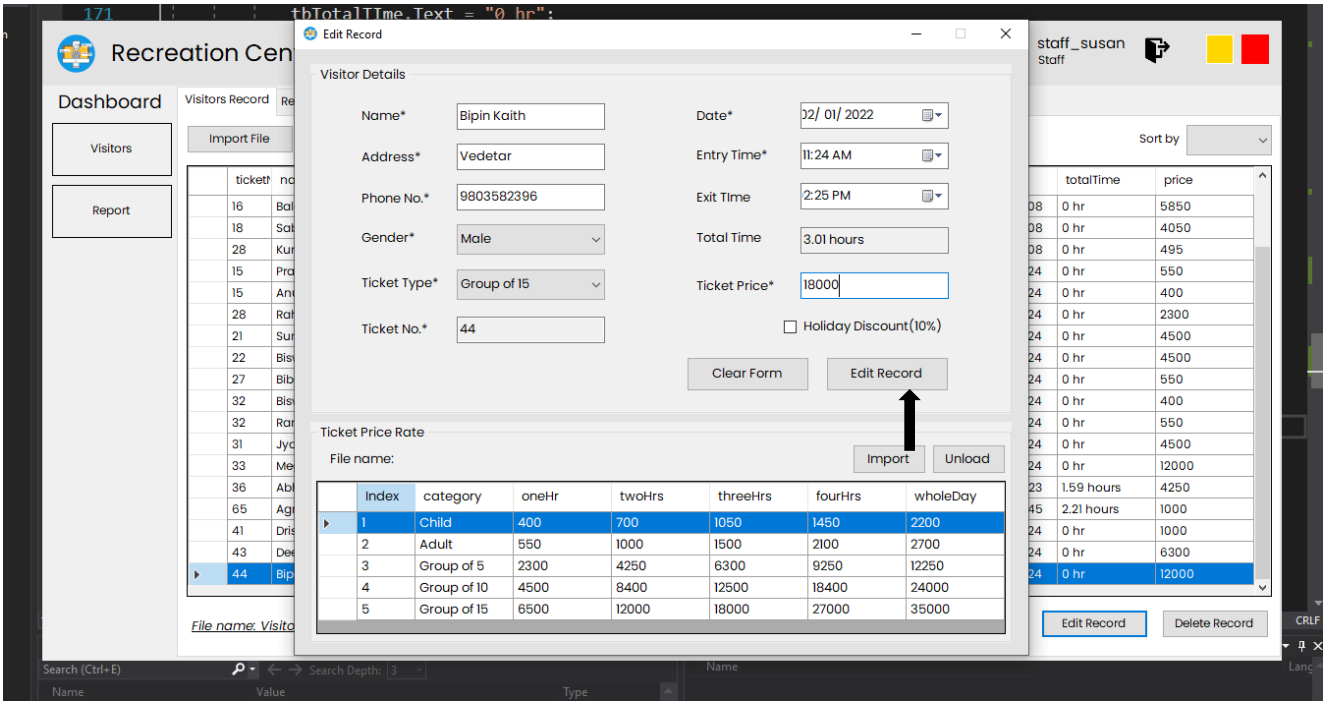
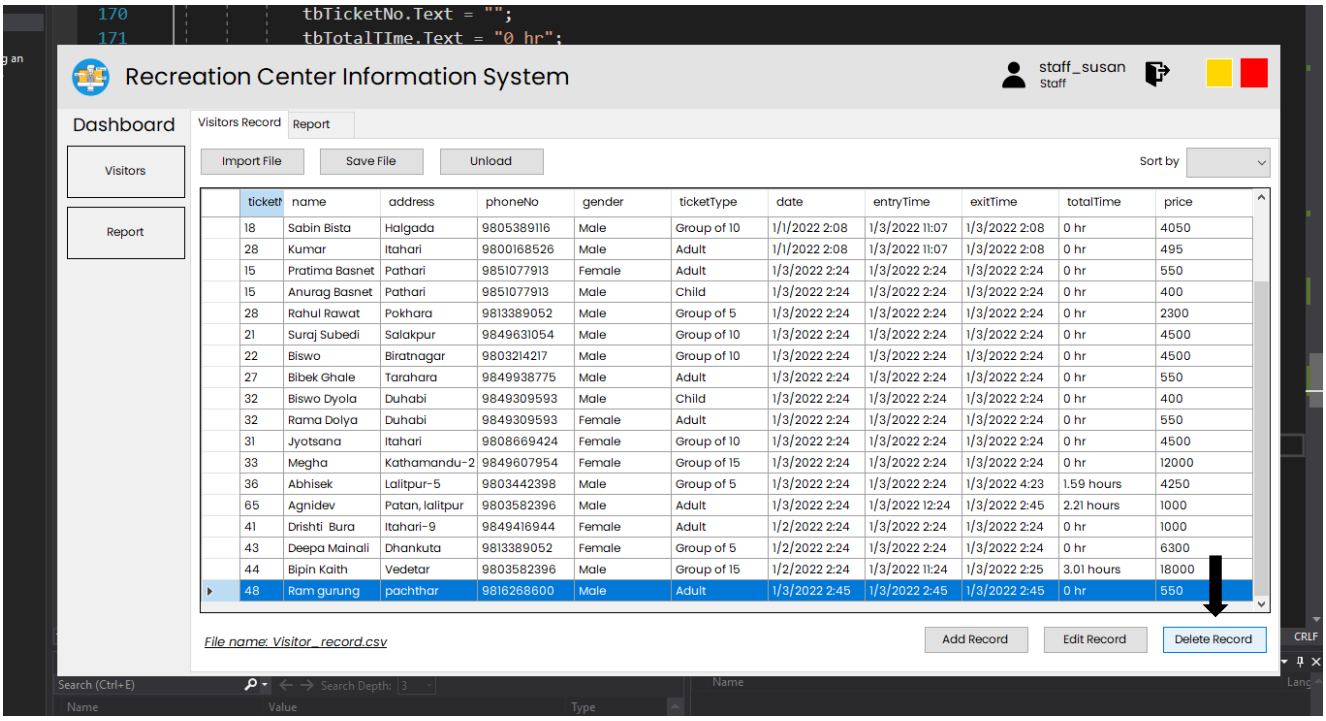



*Figure 8: Edit record*

## Delete visitor's record

- Select the row from grid view indicated by an arrow.
- Click on Delete Record button.




| ticketN | name | address | phoneNo | gender | ticketType | date | entryTime | exitTime | totalTime | price |
|---|---|---|---|---|---|---|---|---|---|---|
| 18 | Sabin Bista | Halgada | 9805389116 | Male | Group of 10 | 1/1/2022 2:08 | 1/3/2022 11:07 | 1/3/2022 2:08 | 0 hr | 4050 |
| 28 | Kumar | Itahari | 9800168526 | Male | Adult | 1/1/2022 2:08 | 1/3/2022 11:07 | 1/3/2022 2:08 | 0 hr | 495 |
| 15 | Pratima Basnet | Pathari | 9851077913 | Female | Adult | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 550 |
| 15 | Anurag Basnet | Pathari | 9851077913 | Male | Child | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 400 |
| 28 | Rahul Rawat | Pokhara | 9813389052 | Male | Group of 5 | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 2300 |
| 21 | Suraj Subedi | Salakpur | 9849631054 | Male | Group of 10 | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 4500 |
| 22 | Biswo | Biratnagar | 9803214217 | Male | Group of 10 | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 4500 |
| 27 | Bibek Ghale | Tarahara | 9849938775 | Male | Adult | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 550 |
| 32 | Biswo Dyola | Duhabi | 9849309593 | Male | Child | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 400 |
| 32 | Rama Dolya | Duhabi | 9849309593 | Female | Adult | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 550 |
| 31 | Jyotsana | Itahari | 9808669424 | Female | Group of 10 | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 4500 |
| 33 | Megha | Kathamandu-2 | 9849607954 | Female | Group of 15 | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 12000 |
| 36 | Abhisek | Lalitpur-5 | 9803442398 | Male | Group of 5 | 1/3/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 4:23 | 1.59 hours | 4250 |
| 65 | Agnidev | Patan, lalitpur | 9803582396 | Male | Adult | 1/3/2022 2:24 | 1/3/2022 12:24 | 1/3/2022 2:45 | 2.21 hours | 1000 |
| 41 | Drishti Bura | Itahari-9 | 9849416944 | Female | Adult | 1/2/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 1000 |
| 43 | Deepa Mainali | Dhankuta | 9813389052 | Female | Group of 5 | 1/2/2022 2:24 | 1/3/2022 2:24 | 1/3/2022 2:24 | 0 hr | 6300 |
| 44 | Bipin Kaith | Vedetar | 9803582396 | Male | Group of 15 | 1/2/2022 2:24 | 1/3/2022 11:24 | 1/3/2022 2:25 | 3.01 hours | 18000 |
| 48 | Ram gurung | pachthar | 9816268600 | Male | Adult | 1/3/2022 2:45 | 1/3/2022 2:45 | 1/3/2022 2:45 | 0 hr | 550 |

*Figure 9: Delete record*

## Generating report

- Navigate to report tab using side panel or tab bar.
- Switch between Daily Report or Weekly Report using tab bar control.
- Click on Generate Chart button.

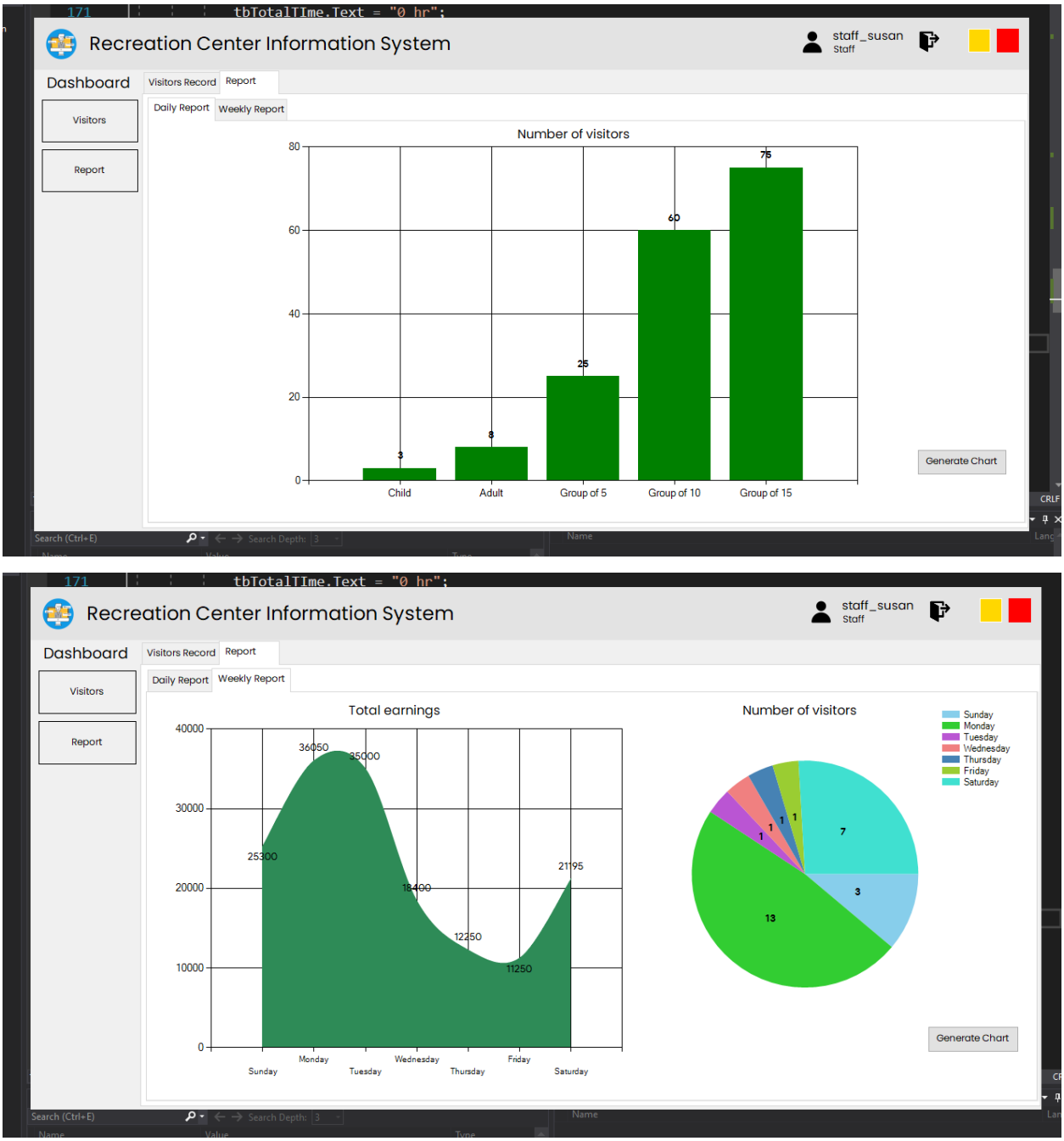
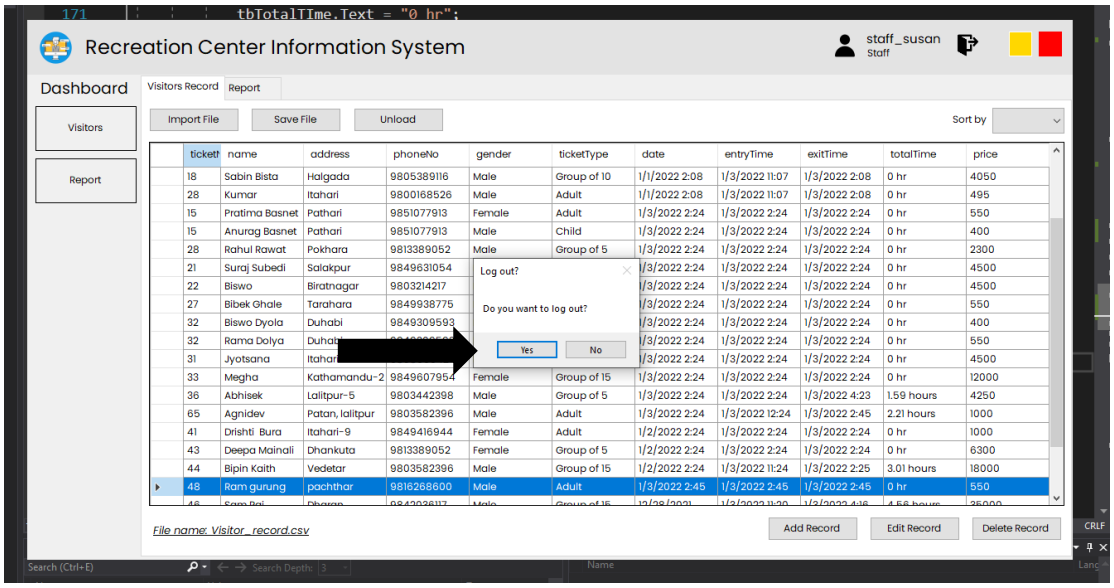



*Figure 10: Generate daily/weekly report chart*

## Log out

- Click on Log Out icon on the top right of the window form.
- Click on Yes to log out the user.




## Admin Dashboard

After log in as an admin, the application redirects user to the admin dashboard where all features including admin functionality to track (add/edit/import/save) ticket record is given. Expect tracking the ticket rate, all the functionalities are same as staff user role.

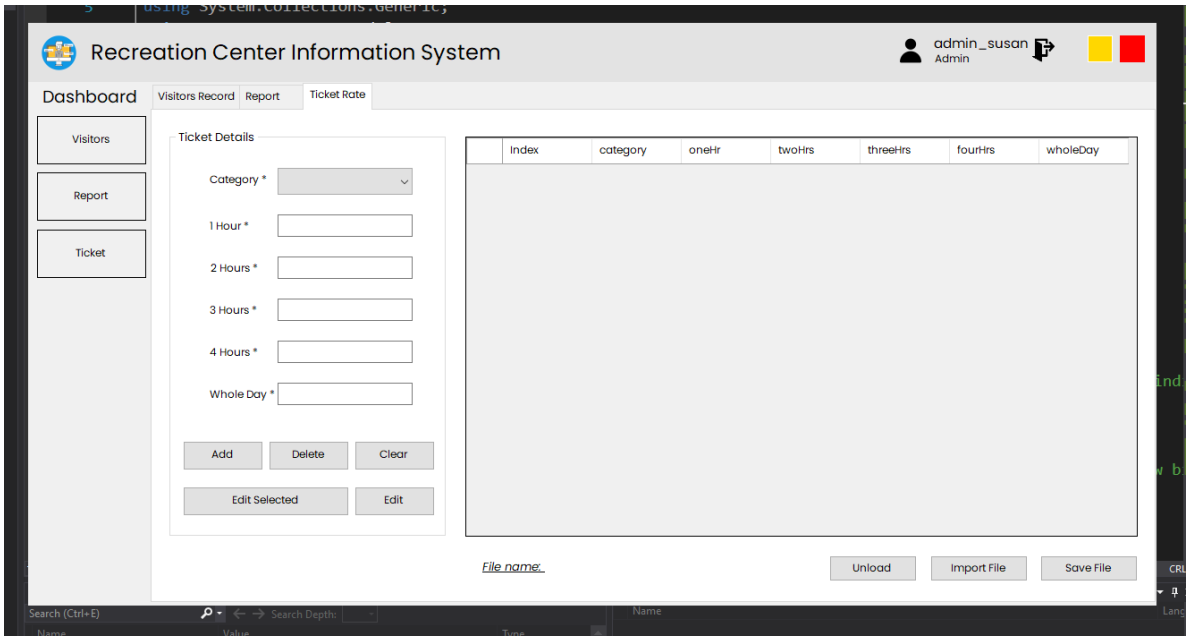



*Figure 11: Admin Dashboard*

**Add ticket option**

- Navigate to Ticket Rate section using side panel or tab bar.
- Select the ticket category and enter price rate according to hour/s.
- Click on Add button.

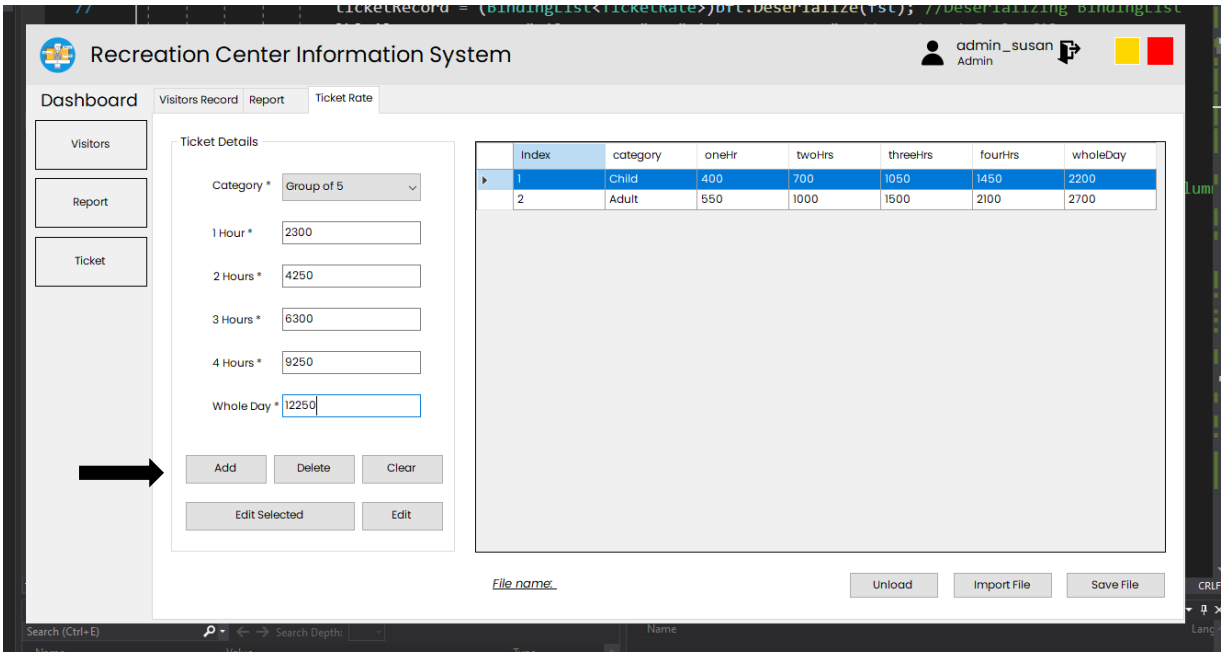



*Figure 12: Add ticket rate list*

**Delete ticket option**

- Select the row to delete from grid view indicated by an arrow.
- Click on Delete.

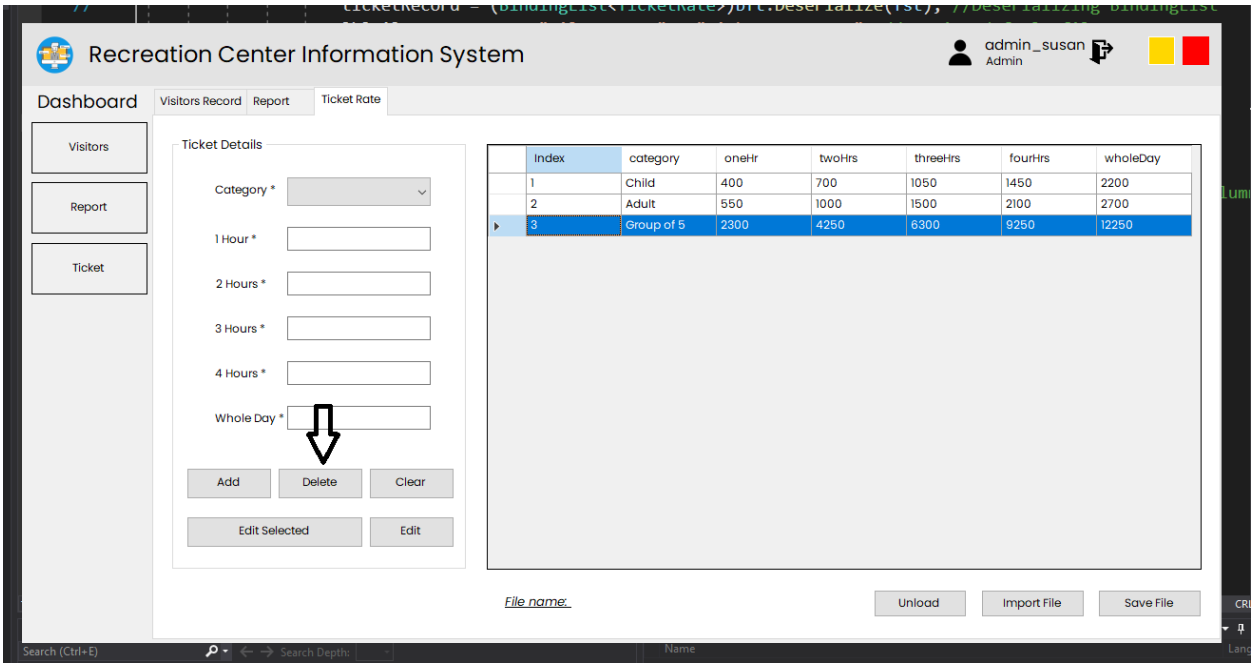



*Figure 13: Delete ticket rate*

**Edit ticket option**

- Select the row to edit from grid view indicated by an arrow.
- Click on Edit Selected button.
- Enter new input where required.
- Click on Edit button.

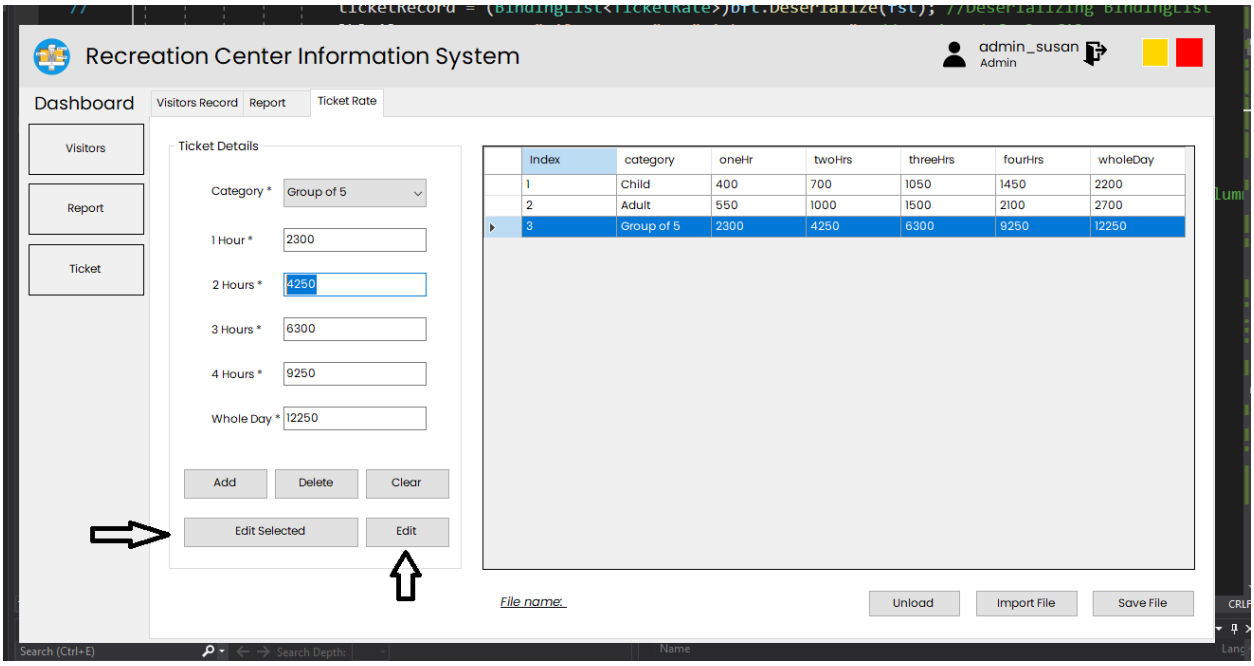



*Figure 14: Edit ticket rate list*

**Hidden features**

- Grid view state (records) are automatically saved as a default file using **serialization**.
- Automatically loads visitor/ticket records if previously saved file is available using **deserialization**.
- Displays user login details (username and user role) on the top bar of application.
- Feature access according to user role and switch between tabs using side panel directly.
- Automatically calculates time spent on editing exit time.
- Displays imported file name on user interface.

# 3. Solution Design

## 3.1. Data Structure

### 3.1.1. Binding List

BindingList is a generic list type that also supports binding. While a generic list may be bound to, BindingList gives you more control over list items, such as whether they can be changed, deleted, or added. BindingList additionally exposes events that are triggered when the list is altered (Telerik.com, 2021).

To bind the GridView components to a generic list in the application, I used Binding List that gave me more control over list items, including the ability to modify, remove, or add items, as well as generate report charts.

### 3.1.2. Array

An array is a collection of similar-typed variables with a common name. Each data item is referred to as an array element. The elements' data types can be any legal data type, such as char, int, float, and so on, and they are kept in a single location. The number of elements in the array is specified by the length of the array (GeeksforGeeks, 2021).

For sorting the visitors' record, I utilized the bubble sort algorithm. The bubble sort compares nearby array elements repeatedly. The first and second elements are compared, and if they are out of order, they are swapped until all of the elements are sorted.

## 3.2. Algorithm

### 3.2.1. Bubble sort

Bubble sort is a sorting algorithm in which two adjacent elements are compared and swapped until they are no longer in the desired order. Each iteration moves each member of the array closer to the end, similar to how air bubbles rise to the surface of water. As a result, it's known as a bubble sort (programiz.com, 2021).

The complexity of Bubble sort, on the other hand, is the same in both the best and worst cases. The effectiveness of the bubble sort algorithm is determined by the number of comparisons between items and the number of exchanges between elements.

#### 3.2.1.1. Working of Bubble sort

**1. First Iteration (Compare and Swap)**

- Compare the first and second items starting with the first index.
- The first and second elements are switched if the first is bigger than the second.
- Compare the second and third items now. If they aren't in the right sequence, swap them.
- The procedure continues until the last piece is added.

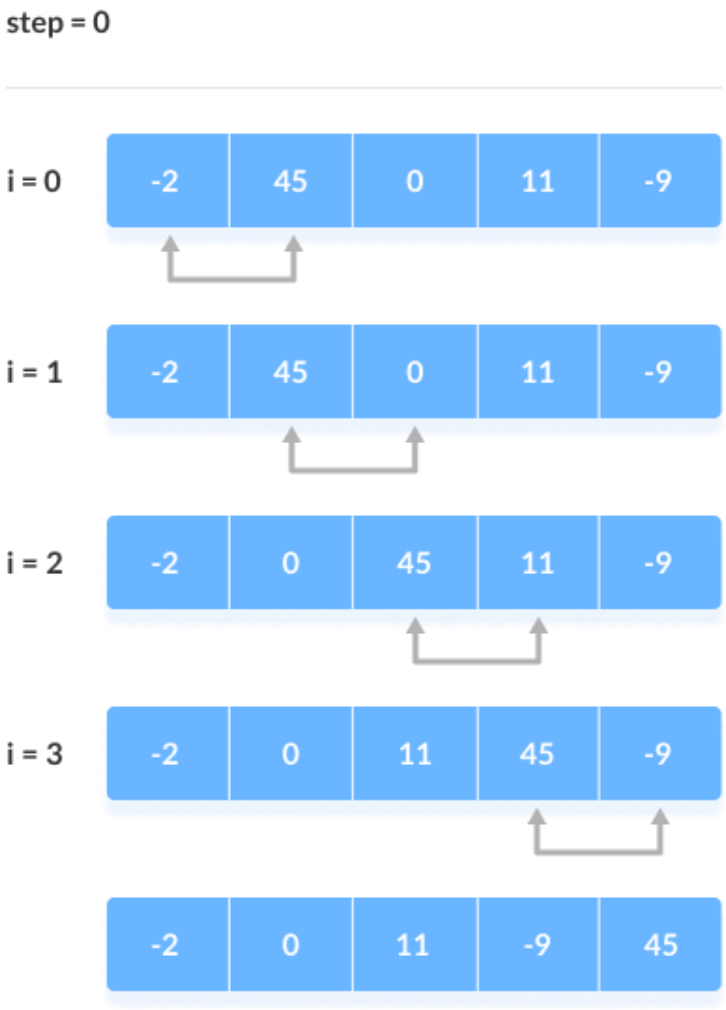



*Figure 15: Bubble sort step 0*

2. **Remaining Iteration**
    - The same process as first iteration goes on for the remaining iterations.
    - The biggest element among the unsorted items is placed at the conclusion of each iteration.

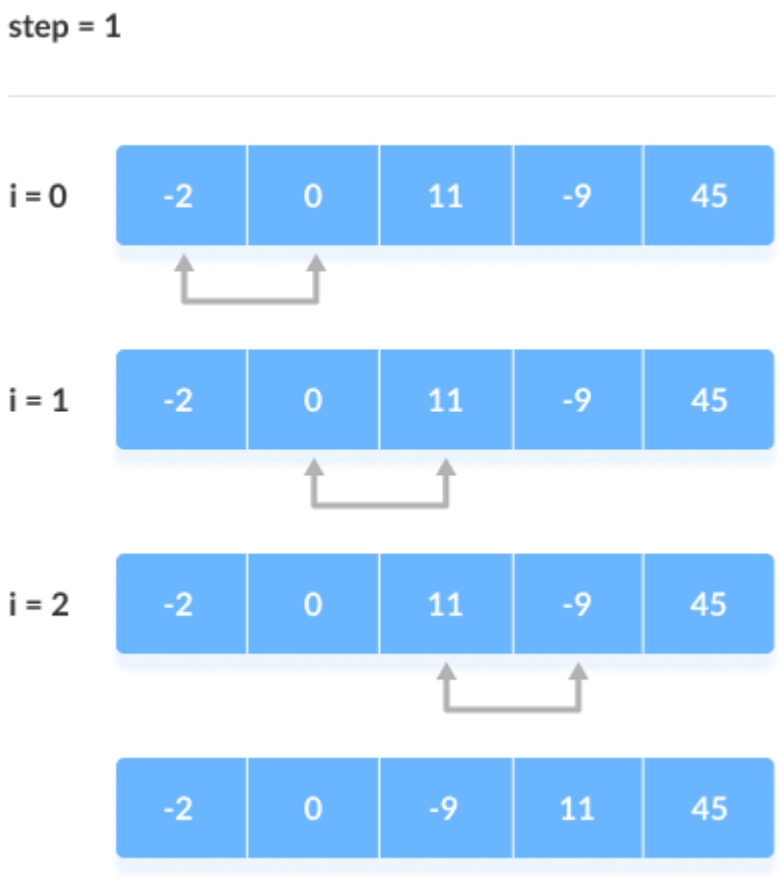



*Figure 16: Bubble sort step 1*

- The comparison takes place up to the final unsorted element in each iteration.

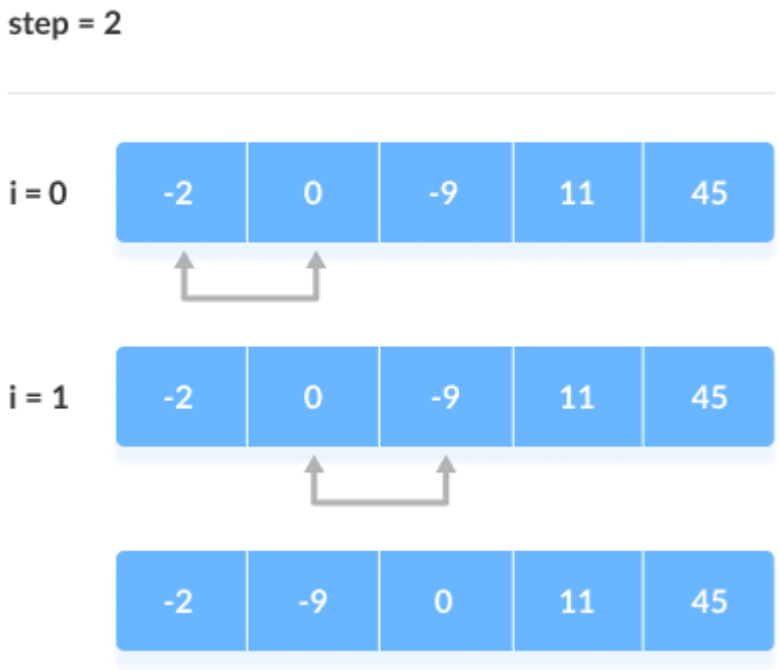



*Figure 17: Bubble step 2*

- When all of the unsorted items have been put in their right placements, the array is said to be sorted.

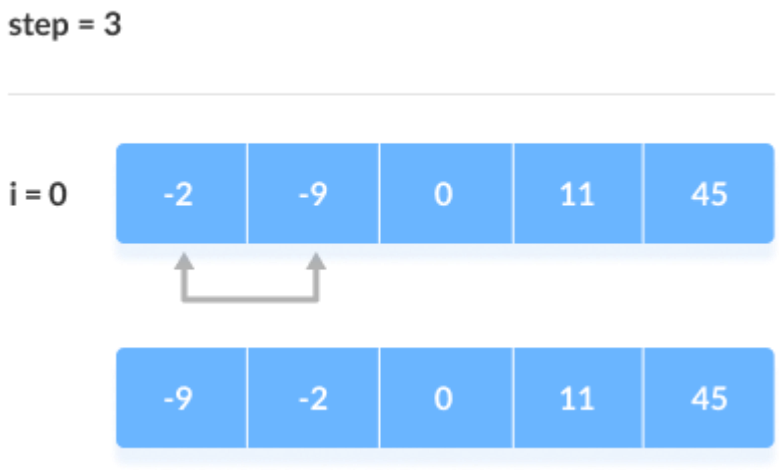



*Figure 18: Bubble sort step 3*

### 3.2.1.2. Flowchart of Bubble sort

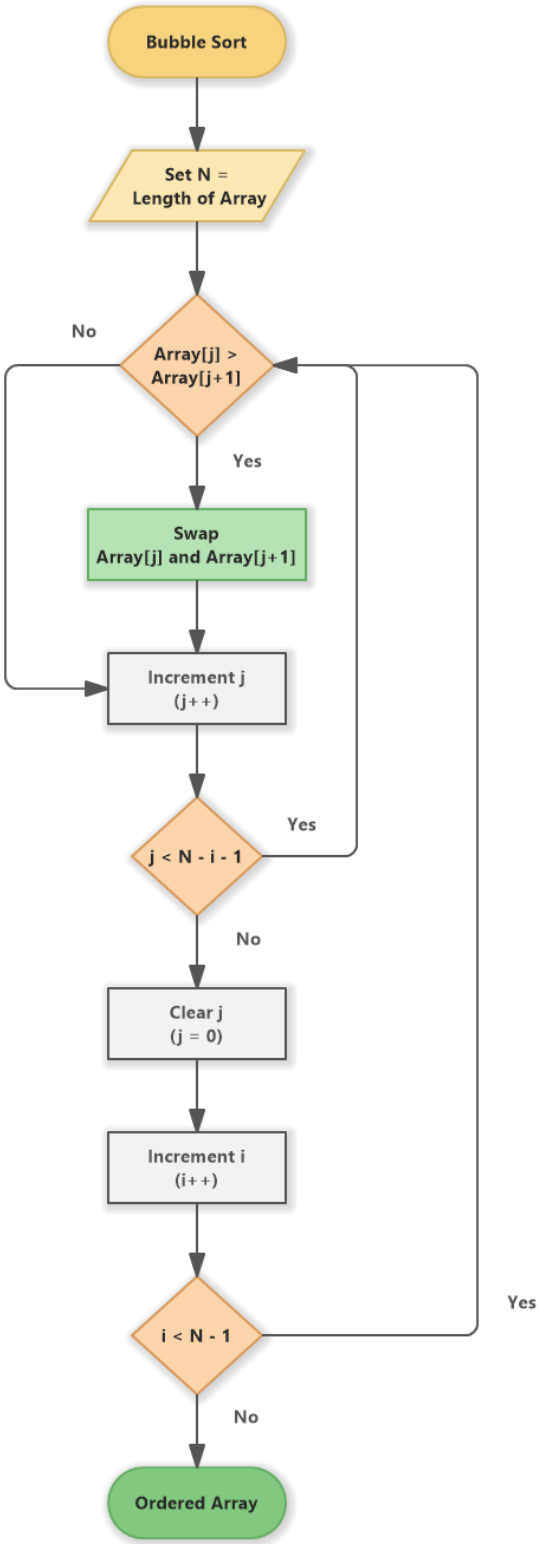



*Figure 19: Flow chart of Bubble sort*

The above flowchart describes the bubble sort algorithm more clearly. Here, an array is taken and its length is set as **N**. After that, if the two adjacent values of an array is compared as (Array[j] > Array[j+1]) and if the condition is satisfied then those values are swapped and if the condition is false then the value of j is incremented and next two indexed values are compared. After that, when the condition is matched and value of j is incremented, it checks for another condition (j < N – i - 1) if the condition is satisfied, it loops to the first comparison condition to order the elements by swapping and if the condition does not match value of j is cleared to 0 and value of i is incremented and the loop is continued until (i < N – 1).

# 4. Software Architecture

## 4.1. Class Diagram

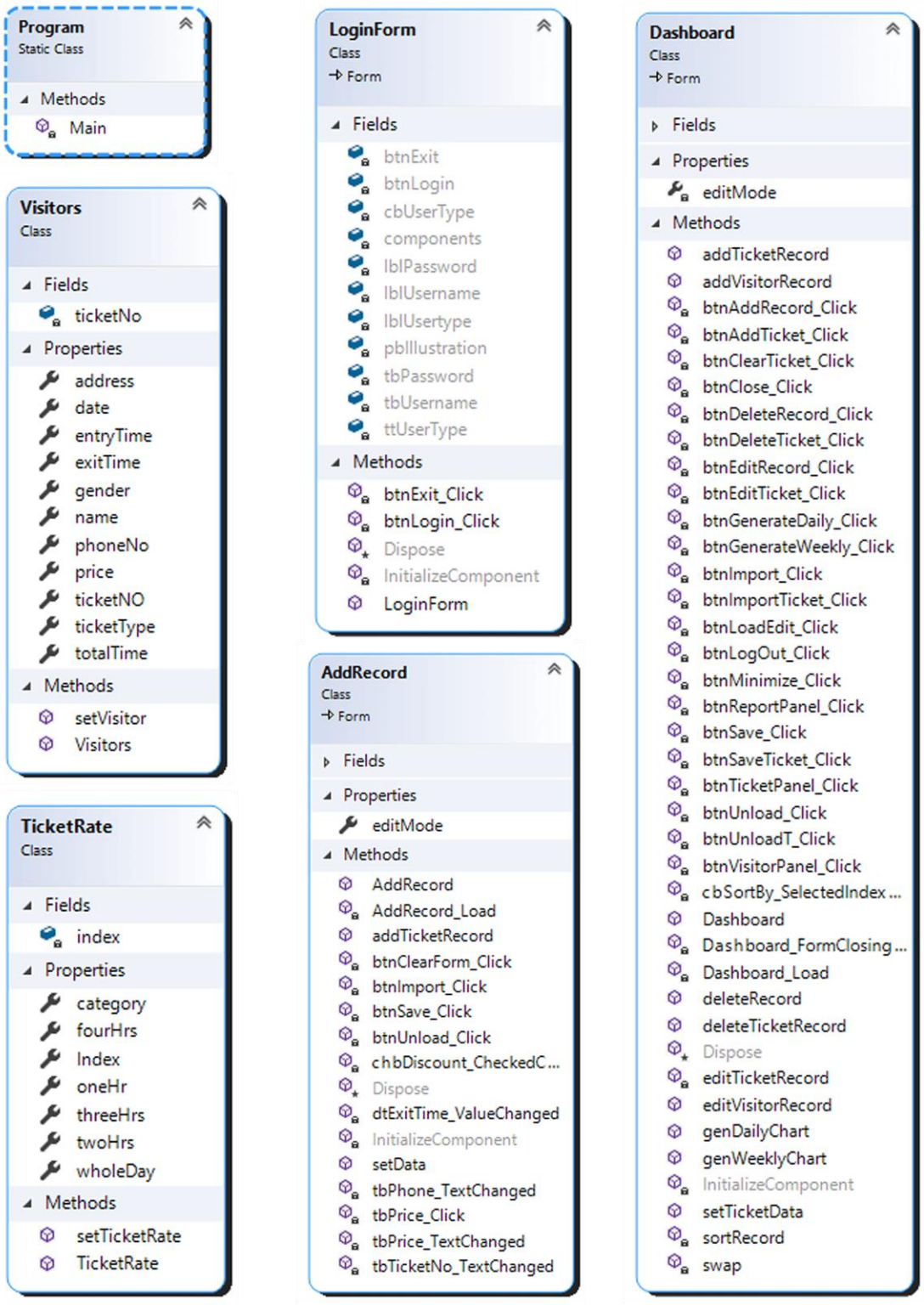



*Figure 20: Class Diagram*

## 4.2. Class, Properties and Method Description

### 4.2.1. Class: LoginForm.cs

This class consists of login fields in UI where as various validation functions are written in the class. This class is used to design and implement login functionality in the application.

### 4.2.2. Class: Visitors.cs

This class is self-written class that has various visitors' details written as properties that are used to record visitors' details as an instance of visitor's object. This class has methods such as setVisitor() and constructor as Visitors() that accepts arguments as paraments from user input.

### 4.2.3. Class: TicketRate.cs

This class is self-written class that has Ticket rate details written as properties that are used to record visitors' details as an instance of visitor's object. This class has methods such as setTicketRate() and constructor as TicketRate() that accepts arguments as paraments from user input.

### 4.2.4. Class: Dashboard.cs

This class consist of various fields, properties and methods as shown in above class diagram to implement required functionalities in the application. The class has designer and code sub classes where design class consist of design component code whereas code class consist of various methods and properties.

### 4.2.5. Class: AddRecord.cs

This class consist of various fields, properties and methods as shown in above class diagram to implement add record functionality. This class is used to add new record to the visitor records file.

## 5. Reflection

The C# programming language is a pleasure to learn and use. It is totally written in the C and C++ programming languages. There is no type conversion, hence there is no risk of data loss. That is why, in addition to the C# language, the developer can write safe code. Types that are both null-capable and non-null-capable are supported. C# is a portable executable file. As I learned Java before c# in my academic course, the 'class' record in Java can accommodate any type of training, whereas the language can accommodate any type of training. One class is the most useful in the language. The C# language's instructions are categorized. Namespaces are used to group classes in the Java language, whereas Packages are used to organize classes in the Java language. C# is a programming language. The variables of primitive record types are stronger in c#, but inside. The variables of primitive facts types are less strong in the Java language.

The C# language has properties and indexers, but the Java language does not. In C# programming, we can easily access Windows API features and COM components, however in Java programming, this is fairly difficult. Operator Overloading, structures, and pre-processor directives are all supported by the C# language, but not by Java. Namespaces are not related with directories in C# programming, but directory names are directly associated with programs in Java programming (fyber, 2019).

## 6. Conclusion

A technical report has been prepared with the development process of the entire application. This report consist of a section for User Manual that is prepared for new users to understand the application and the operation steps and Solution Design section where logical solution to implement various functions in the application is concisely described. Also, a detailed section for Software Architecture is included in this report that consist of a class diagram, a detailed description of classes, properties and method. Lastly, the reflection of own experience of using C# and Visual Studio for development task is included.

## 7. References

fyber. (2019) *What Is C# Language, Advantages & Features Of C# Language* [Online]. Available at: https://codexoxo.com/advantages-c-sharp-language/ [Accessed 2 January 2022].

GeeksforGeeks. (2021) *C# | Arrays* [Online]. Available at: https://www.geeksforgeeks.org/c-sharp-arrays/ [Accessed 2 January 2022].

programiz.com. (2021) [Online]. Available at: https://www.programiz.com/dsa/bubble-sort [Accessed 2 January 2022].

Telerik.com. (2021) *Binding to BindingList* [Website]. Available at: https://docs.telerik.com/devtools/winforms/controls/gridview/populating-with-data/binding-to-bindinglist [Accessed 2 January 2022].

# 8. Appendix

## Appendix-A: Source Code

### Class: program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

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

### Class: LoginForm.cs

```
//Written by: Susan Shrestha
//Date: 29 December, 2021
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 Recreation_Center
{
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
        }

        private void btnLogin_Click(object sender, EventArgs e)
        {
            //storing windows form values in variables
            string username = tbUsername.Text;
            string password = tbPassword.Text;
```

```
            string userType = Convert.ToString(cbUserType.SelectedItem);

            if (username == "" && password == "") //checking empty fields
            {
                MessageBox.Show(
                    "Please enter the username and password.",
                    "Invalid input");
            }
            else if (cbUserType.SelectedItem == default) //validating user type
            {
                MessageBox.Show(
                    "Please select user type.",
                    "Invalid user credentials.");
            }
            else if (username == "admin_susan" && password == "admin" && userType ==
"Admin") //validating user credentials
            {
                Dashboard dashboard = new Dashboard(username, userType); //passing
username and usertype values to dashboard form
                this.Hide();
                dashboard.Show();
            }
            else if (username == "staff_susan" && password == "staff" && userType ==
"Staff") //validating user credentials
            {
                Dashboard dashboard = new Dashboard(username, userType);
                this.Hide();
                dashboard.Show();
            }
            else
            {
                MessageBox.Show("Please enter the correct login details.", "Invalid
login");

            }
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.Application.Exit(); //closing application
        }
    }
}
```

**Class: Dashboard.cs**

//Written by: Susan Shrestha

//Date: 29 December, 2021

```
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;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Windows.Forms.DataVisualization.Charting;

namespace Recreation_Center
{
    public partial class Dashboard : Form
    {
        //for visitor records
        public static BindingList<Visitors> visitorRecord = new BindingList<Visitors>();
//initializing new binding list with the type of Visitor
        private BindingList<string> alreadyOpenedVisitor = new BindingList<string>();

        //for ticket rate
        public static BindingList<TicketRate> ticketRecord = new
BindingList<TicketRate>(); //initializing new binding list with the type of Visitor
        private BindingList<string> alreadyOpenedTicket = new BindingList<string>();

        string username;
        string usertType;
        public Dashboard(string getUsername, string getUserType)
        {
            InitializeComponent();
            //retriving username and usertype values from LoginForm
            this.username = getUsername;
            this.usertType = getUserType;
```

```
    }

    private void Dashboard_Load(object sender, EventArgs e) //After Login
    {
        //setting defaults on form load event
        lblUsername.Text = username;
        lblUserType.Text = usertType;
        tbIndex.Visible = false;

        try {
            if (File.Exists("Visitor_record.csv")) //checking existing visitor record file to load in the GridView
            {
                FileStream fs = new FileStream("Visitor_record.csv", FileMode.Open);
                BinaryFormatter bf = new BinaryFormatter();
                visitorRecord = (BindingList<Visitors>)bf.Deserialize(fs); //Deserializing BindingList
                lblFileName.Text = "File name: " + "Visitor_record.csv"; //setting default filename
                fs.Close();
            }
        }
        catch(Exception ex) { MessageBox.Show(ex.Message); } //excepting handling for possible errors

        //granting access to admin and user
        if (lblUserType.Text == "Admin")
        {
            btnTicketPanel.Visible = true;
        }
        else
```

```
        {
            btnTicketPanel.Visible = false;
            mainPanel.TabPages.Remove(ticket);

        }

        //loading ticket rate record in datagridview
        try
        {
            if (File.Exists("Ticket_rate.csv")) //checking existing file to load in the GridView
            {
                FileStream fst = new FileStream("Ticket_rate.csv", FileMode.Open);
                BinaryFormatter bft = new BinaryFormatter();
                ticketRecord = (BindingList<TicketRate>)bft.Deserialize(fst); //Deserializing
BindingList
                lblFileNameT.Text = "File name: " + "Ticket_rate.csv"; //setting default
filename
                fst.Close();
            }
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }

        dgvVisitors.DataSource = visitorRecord;
        dgvVisitors.Columns["ticketNo"].AutoSizeMode                                =
DataGridViewAutoSizeColumnMode.Fill; //gridView column size fill

        dgvTicketRate.DataSource = ticketRecord; //setting GridView datasource with
ticketrate' record
        //dgvTicketRate.Columns["Index"].AutoSizeMode                               =
DataGridViewAutoSizeColumnMode.Fill;
    }
```

```
//Switching between tabs with button control
private void btnTicketPanel_Click(object sender, EventArgs e)
{
    mainPanel.SelectedTab = mainPanel.TabPages[2];
}
private void btnReportPanel_Click(object sender, EventArgs e)
{
    mainPanel.SelectedTab = mainPanel.TabPages[1];
}
private void btnVisitorPanel_Click(object sender, EventArgs e)
{
    mainPanel.SelectedTab = mainPanel.TabPages[0];
}

//closing the program
private void btnClose_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Do you want to Exit?", "Exit?", MessageBoxButtons.YesNo);
    if (result == DialogResult.Yes)
        System.Windows.Forms.Application.Exit();
}

//minimize window (form)
private void btnMinimize_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Minimized;
}

//asking user to log out when clicked
```

```
    private void btnLogOut_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("Do you want to log out?", "Log out?", MessageBoxButtons.YesNo);
        if(result == DialogResult.Yes)
        {
            this.Hide();
            LoginForm login = new LoginForm();
            login.Show();
        }
    }

    private void btnImport_Click(object sender, EventArgs e) //importing file from local folder
    {
        OpenFileDialog ofd = new OpenFileDialog();
        ofd.Filter = "CSV File|*.csv"; //filtering files
        if (ofd.ShowDialog() == DialogResult.OK)
        {
            if (!(alreadyOpenedVisitor.Contains(ofd.FileName)))
            {
                string[] lines = File.ReadAllLines(ofd.FileName); //retriving csv file in lines to lines array
                // filename for validation
                alreadyOpenedVisitor.Add(ofd.FileName);
                string[] items;
                bool firstline = true;

                lines.Skip<string>(1); //skipping header line

                foreach (string line in lines) //retriving each lines from lines array
```

```
            {
                items = line.Split(','); //spliting each items
                if (firstline == true)
                {
                    firstline = false;
                }
                else
                {
                    try
                    {
                        Visitors visitor = new Visitors(Convert.ToInt32(items[0]), items[1], items[2], Convert.ToDouble(items[3]), items[4], items[5], Convert.ToDateTime(items[6]), Convert.ToDateTime(items[7]), Convert.ToDateTime(items[8]), items[9], Convert.ToInt32(items[10]));
                        addVisitorRecord(visitor); //passing instance of an object to add visitor record
                    }
                    catch(FormatException)
                    {
                        MessageBox.Show("Please import a valid CSV file.", "Invalid file import");
                        break;
                    }

                }

            }
            lblFileName.Text = "File name: " + ofd.SafeFileName; //setting file name
        }
        else
        {
```

```
                MessageBox.Show("ERROR", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
    }

    public void addVisitorRecord(Visitors e)
    {
        visitorRecord.Add(e); //adding record to list
        dgvVisitors.DataSource = visitorRecord; //setting datasource for datagridview
    }

    //Opens AddRecord form to add new record
    private void btnAddRecord_Click(object sender, EventArgs e)
    {
        if(Application.OpenForms["AddRecord"] == null)
        {
            AddRecord ar = new AddRecord(); //creating object for AddRecord form
            ar.Show(); //opening AddRecord form using object instance
        }
        else
        {
            Application.OpenForms["AddRecord"].BringToFront();
        }
    }

    //saving file in local directory
    private void btnSave_Click(object sender, EventArgs e)
    {
        SaveFileDialog save = new SaveFileDialog();
        save.Filter = "CSV file|*.csv;";
```

```
        save.DefaultExt = ".csv";
        save.FileName = "Visitors " + DateTime.Now.ToString("dd-MM-yyyy");
        if (save.ShowDialog() == DialogResult.OK)
        {
            StreamWriter swr = new StreamWriter(save.FileName); //creating streamWriter object
            StringBuilder sb = new StringBuilder();
            sb.AppendLine("Ticket No., Name, Address, Phone, Gender, Ticket Type, Date, Entry Time, Exit Time, Total Time, Price");
            foreach (Visitors visitor in visitorRecord)
            {
                //appending rows using StringBuilder
                sb.AppendLine(visitor.ticketNO.ToString() + "," +
                    visitor.name + "," +
                    visitor.address + "," +
                    visitor.phoneNo.ToString() + "," +
                    visitor.gender + "," +
                    visitor.ticketType + "," +
                    visitor.date.ToString() + "," +
                    visitor.entryTime.ToString() + "," +
                    visitor.exitTime.ToString() + "," +
                    visitor.totalTime + "," +
                    visitor.price.ToString());
            }
            swr.Write(sb.ToString()); //writing to csb file
            swr.Close();
            MessageBox.Show("Visitor records saved successfully.","Saved",MessageBoxButtons.OK);
        }
    }
```

```
public void deleteRecord(int ticketNo)
{
    foreach (Visitors e in visitorRecord)
    {
        if (e.ticketNO == ticketNo) //selecting key as ticketNo to delete selected row from record
        {
            visitorRecord.Remove(e); //removing from binding list
            dgvVisitors.Refresh();
            return;
        }
    }
}

private void btnDeleteRecord_Click(object sender, EventArgs e)
{
    try
    {
        deleteRecord(Convert.ToInt32(dgvVisitors.SelectedRows[0].Cells[0].Value)); //for deleting the selected/active row
    }
    catch (ArgumentOutOfRangeException) { MessageBox.Show("No rows selected.","Invalid delete"); }
}

public void editVisitorRecord(int ticketNo, String name, String address, Double phoneNo, String gender, String ticketType, DateTime date, DateTime entryTime, DateTime exitTime, String totalTime, int price)
{
    foreach (Visitors e in visitorRecord)
    {
```

```
            if (e.ticketNO == ticketNo) //selected ticketNo from selected row to edit record
            {
                e.setVisitor(ticketNo, name, address, phoneNo, gender, ticketType, date, entryTime, exitTime, totalTime, price);
                dgvVisitors.Refresh();
                return;
            }
        }
    }

    private void btnEditRecord_Click(object sender, EventArgs e)
    {
        try
        {
            if (Application.OpenForms["AddRecord"] == null)
            {
                AddRecord ar = new AddRecord();
                ar.editMode = true;
                ar.Text = "Edit Record";
                ar.setData(Convert.ToInt32(dgvVisitors.SelectedRows[0].Cells[0].Value),
                    dgvVisitors.SelectedRows[0].Cells[1].Value.ToString(),
                    dgvVisitors.SelectedRows[0].Cells[2].Value.ToString(),
                    Convert.ToDouble(dgvVisitors.SelectedRows[0].Cells[3].Value),
                    dgvVisitors.SelectedRows[0].Cells[4].Value.ToString(),
                    dgvVisitors.SelectedRows[0].Cells[5].Value.ToString(),
                    Convert.ToDateTime(dgvVisitors.SelectedRows[0].Cells[6].Value),
                    Convert.ToDateTime(dgvVisitors.SelectedRows[0].Cells[7].Value),
                    Convert.ToDateTime(dgvVisitors.SelectedRows[0].Cells[8].Value),
                    dgvVisitors.SelectedRows[0].Cells[9].Value.ToString(),
                    Convert.ToInt32(dgvVisitors.SelectedRows[0].Cells[10].Value)
                );
```

```
                ar.Show();
            }
            else
            {
                ((AddRecord)Application.OpenForms["AddRecord"]).editMode = true;
                Application.OpenForms["AddRecord"].BringToFront();
            }
        }
        catch (ArgumentOutOfRangeException) { MessageBox.Show("No rows selected.", "Invalid action"); }

    }

    //Report section coding: Chart generation
    private void btnGenerateDaily_Click(object sender, EventArgs e)
    {
        dailyChart.Series["Number of visitors"].Points.Clear();
        genDailyChart(visitorRecord);
    }

    //daily chart
    public void genDailyChart(BindingList<Visitors> record)
    {
        int childCount = 0;
        int adultCount = 0;
        int groupOf5Count = 0;
        int groupOf10Count = 0;
        int groupOf15Count = 0;
        string child = "Child";
        string adult = "Adult";
        string groupOf5 = "Group of 5";
```

```
string groupOf10 = "Group of 10";
string groupOf15 = "Group of 15";

foreach (Visitors vis in record)
{
    if (vis.ticketType.ToUpper().Contains(child.ToUpper()))
    {
        childCount++;
    }
    else if (vis.ticketType.ToUpper().Contains(adult.ToUpper()))
    {
        adultCount++;
    }
    else if (vis.ticketType.ToUpper().Contains(groupOf5.ToUpper()))
    {
        groupOf5Count += 5;
    }
    else if (vis.ticketType.ToUpper().Contains(groupOf10.ToUpper()))
    {
        groupOf10Count += 10;
    }
    else if(vis.ticketType.ToUpper().Contains(groupOf15.ToUpper()))
    {
        groupOf15Count += 15;
    }
}

dailyChart.Series[0].Points.AddXY("Child", childCount);
dailyChart.Series[0].Points.AddXY("Adult", adultCount);
dailyChart.Series[0].Points.AddXY("Group of 5", groupOf5Count);
dailyChart.Series[0].Points.AddXY("Group of 10", groupOf10Count);
```

```
        dailyChart.Series[0].Points.AddXY("Group of 15", groupOf15Count);
        dailyChart.Series[0].IsValueShownAsLabel = true;
        dailyChart.Series[0].IsVisibleInLegend = true;
        //dailyChart.Series[0].ChartType = SeriesChartType.Pie;
    }

    private void btnGenerateWeekly_Click(object sender, EventArgs e)
    {
        weeklyChartVisitor.Series["Visitors"].Points.Clear();
        weeklyChartEarning.Series["Earnings"].Points.Clear();
        genWeeklyChart(visitorRecord);
    }

    //weekly chart
    public void genWeeklyChart(BindingList<Visitors> record)
    {
        //visitor variables
        int sundayCount = 0;
        int mondayCount = 0;
        int tuesdayCount = 0;
        int wednesdayCount = 0;
        int thursdayCount = 0;
        int fridayCount = 0;
        int saturdayCount = 0;

        //earning variables
        double sundayEarning = 0;
        double mondayEarning = 0;
        double tuesdayEarning = 0;
        double wednesdayEarning = 0;
        double thursdayEarning = 0;
```

```
        double fridayEarning = 0;
        double saturdayEarning = 0;

        string sunday = "Sunday";
        string monday = "Monday";
        string tuesday = "Tuesday";
        string wednesday = "Wednesday";
        string thursday = "Thursday";
        string friday = "Friday";
        string saturday = "Saturday";

        foreach (Visitors vis in record)
        {
            if (vis.date.DayOfWeek.ToString().Contains(sunday)) //checking condition for visitor record appearing in sunday
            {
                sundayCount++; //incrementing visitor count if visitor record appears in sunday
                sundayEarning += vis.price; //sum of earnings for sunday
            }
            else if (vis.date.DayOfWeek.ToString().Contains(monday))
            {
                mondayCount++;
                mondayEarning += vis.price;
            }
            else if (vis.date.DayOfWeek.ToString().Contains(tuesday))
            {
                tuesdayCount++;
                tuesdayEarning += vis.price;
            }
            else if (vis.date.DayOfWeek.ToString().Contains(wednesday))
```

```
        {
            wednesdayCount++;
            wednesdayEarning += vis.price;
        }
        else if (vis.date.DayOfWeek.ToString().Contains(thursday))
        {
            thursdayCount++;
            thursdayEarning += vis.price;
        }
        else if (vis.date.DayOfWeek.ToString().Contains(friday))
        {
            fridayCount++;
            fridayEarning += vis.price;
        }
        else if (vis.date.DayOfWeek.ToString().Contains(saturday))
        {
            saturdayCount++;
            saturdayEarning += vis.price;
        }
    }

    weeklyChartVisitor.Series[0].Points.AddXY("Sunday", sundayCount);
    weeklyChartVisitor.Series[0].Points.AddXY("Monday", mondayCount);
    weeklyChartVisitor.Series[0].Points.AddXY("Tuesday", tuesdayCount);
    weeklyChartVisitor.Series[0].Points.AddXY("Wednesday", wednesdayCount);
    weeklyChartVisitor.Series[0].Points.AddXY("Thursday", thursdayCount);
    weeklyChartVisitor.Series[0].Points.AddXY("Friday", fridayCount);
    weeklyChartVisitor.Series[0].Points.AddXY("Saturday", saturdayCount);

    weeklyChartVisitor.Series[0].IsValueShownAsLabel = true;
    weeklyChartVisitor.Series[0].IsVisibleInLegend = true;
```

```
        weeklyChartVisitor.Series[0].ChartType = SeriesChartType.Pie;

        weeklyChartEarning.Series[0].Points.AddXY("Sunday", sundayEarning);
        weeklyChartEarning.Series[0].Points.AddXY("Monday", mondayEarning);
        weeklyChartEarning.Series[0].Points.AddXY("Tuesday", tuesdayEarning);
        weeklyChartEarning.Series[0].Points.AddXY("Wednesday", wednesdayEarning);
        weeklyChartEarning.Series[0].Points.AddXY("Thursday", thursdayEarning);
        weeklyChartEarning.Series[0].Points.AddXY("Friday", fridayEarning);
        weeklyChartEarning.Series[0].Points.AddXY("Saturday", saturdayEarning);

        weeklyChartEarning.Series[0].IsValueShownAsLabel = true;
        weeklyChartEarning.Series[0].IsVisibleInLegend = true;
        weeklyChartEarning.Series[0].ChartType = SeriesChartType.SplineArea;
    }

    //clearing form
    private void btnClearTicket_Click(object sender, EventArgs e)
    {
        cbCategory.Text = default;
        tb1Hr.Text = "";
        tb2Hrs.Text = "";
        tb3Hrs.Text = "";
        tb4Hrs.Text = "";
        tbWholeDay.Text = "";
    }

    //unloading dataGridView
    private void btnUnload_Click(object sender, EventArgs e)
    {
        lblFileName.Text = "File name: ";
        dgvVisitors.Rows.Clear();
```

```
            dgvVisitors.Refresh();
        }

        //TicketRate code starts here
        public int index = 1;
        private bool editMode { get; set; }
        private void btnAddTicket_Click(object sender, EventArgs e)
        {
            if (cbCategory.Text == "" || tb1Hr.Text == "" || tb2Hrs.Text == "" || tb3Hrs.Text ==
"" || tb4Hrs.Text == "" || tbWholeDay.Text == "")
            {
                MessageBox.Show("Please fill the required (*) feilds.", "Invalid Input",
MessageBoxButtons.OK);
            }
            else
            {
                try
                {
                    TicketRate ticket = new TicketRate(
                        index++, //incrementing index value
                        cbCategory.Text,
                        Convert.ToInt32(tb1Hr.Text),
                        Convert.ToInt32(tb2Hrs.Text),
                        Convert.ToInt32(tb3Hrs.Text),
                        Convert.ToInt32(tb4Hrs.Text),
                        Convert.ToInt32(tbWholeDay.Text));
                    addTicketRecord(ticket);
                    cbCategory.Text = default;
                    tb1Hr.Text = "";
                    tb2Hrs.Text = "";
                    tb3Hrs.Text = "";
```

```
            tb4Hrs.Text = "";
            tbWholeDay.Text = "";
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Error");
        }
    }
}

private void btnLoadEdit_Click(object sender, EventArgs e)
{
    try
    {
        setTicketData( //parsing and passing data from gridview selected row
            Convert.ToInt32(dgvTicketRate.SelectedRows[0].Cells[0].Value),
            dgvTicketRate.SelectedRows[0].Cells[1].Value.ToString(),
            Convert.ToInt32(dgvTicketRate.SelectedRows[0].Cells[2].Value),
            Convert.ToInt32(dgvTicketRate.SelectedRows[0].Cells[3].Value),
            Convert.ToInt32(dgvTicketRate.SelectedRows[0].Cells[4].Value),
            Convert.ToInt32(dgvTicketRate.SelectedRows[0].Cells[5].Value),
            Convert.ToInt32(dgvTicketRate.SelectedRows[0].Cells[6].Value)
        );

    }
    catch (Exception) { MessageBox.Show("No rows selected.", "Error"); }
}

//sets ticket data to form fields for edit
public void setTicketData(int index, String category, int oneHr, int twoHrs, int threeHrs, int fourHrs, int wholeday)
```

```
    {
        tbIndex.Text = index.ToString();
        cbCategory.Text = category;
        tb1Hr.Text = oneHr.ToString();
        tb2Hrs.Text = twoHrs.ToString();
        tb3Hrs.Text = threeHrs.ToString();
        tb4Hrs.Text = fourHrs.ToString();
        tbWholeDay.Text = wholeday.ToString();
    }

    private void btnEditTicket_Click(object sender, EventArgs e)
    {
        try
        {
            editTicketRecord( //parsing and passing form data to edit record
                Convert.ToInt32(tbIndex.Text),
                cbCategory.Text,
                Convert.ToInt32(tb1Hr.Text),
                Convert.ToInt32(tb2Hrs.Text),
                Convert.ToInt32(tb3Hrs.Text),
                Convert.ToInt32(tb4Hrs.Text),
                Convert.ToInt32(tbWholeDay.Text)
            );

        }
        catch (Exception)
        {
            MessageBox.Show("No rows selected.", "Error");
        }

    }
```

```
    private void btnDeleteTicket_Click(object sender, EventArgs e)
    {
        try
        {

deleteTicketRecord(Convert.ToInt32(dgvTicketRate.SelectedRows[0].Cells[0].Value));

        }
        catch (ArgumentOutOfRangeException) { MessageBox.Show("No rows selected.", "Invalid delete"); }
    }

    public void deleteTicketRecord(int index)
    {
        foreach (TicketRate e in ticketRecord)
        {
            if (e.Index == index) //deleting selected row using index value
            {
                ticketRecord.Remove(e);
                dgvTicketRate.Refresh();
                return;
            }
        }
    }

    //importing record from file using OpenFileDialog
    private void btnImportTicket_Click(object sender, EventArgs e)
    {
        OpenFileDialog ofd = new OpenFileDialog();
```

```
        ofd.Filter = "CSV File|*.csv";
        if (ofd.ShowDialog() == DialogResult.OK)
        {
            if (!(alreadyOpenedTicket.Contains(ofd.FileName)))
            {
                string[] lines = File.ReadAllLines(ofd.FileName);
                // filename for validation
                alreadyOpenedTicket.Add(ofd.FileName);
                string[] items;
                bool firstline = true;

                lines.Skip<string>(1);

                foreach (string line in lines)
                {
                    items = line.Split(',');
                    if (firstline == true)
                    {
                        firstline = false;
                    }
                    else
                    {
                        try
                        {
                            TicketRate ticket = new TicketRate(Convert.ToInt32(items[0]), items[1], Convert.ToInt32(items[2]), Convert.ToInt32(items[3]), Convert.ToInt32(items[4]), Convert.ToInt32(items[5]), Convert.ToInt32(items[6]));
                            addTicketRecord(ticket);
                            dgvTicketRate.Columns["Index"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
                            lblFileNameT.Text = "File name: " + ofd.SafeFileName;
```

```
                    }
                    catch (Exception)
                    {
                        MessageBox.Show("Please import a valid CSV file.", "Invalid file import", MessageBoxButtons.OK);
                        break;
                    }

                }

            }
        }
        else
        {
            MessageBox.Show("ERROR", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

        }
    }
}

public void addTicketRecord(TicketRate e)
{
    ticketRecord.Add(e);
    dgvTicketRate.DataSource = ticketRecord;
}

private void editTicketRecord(int index, String category, int oneHr, int twoHrs, int threeHrs, int fourHrs, int wholeDay)
{
    foreach (TicketRate e in ticketRecord)
```

```
        {
            if (e.Index == index) //editing selected index row
            {
                e.setTicketRate(index, category, oneHr, twoHrs, threeHrs, fourHrs,
wholeDay);
                dgvTicketRate.Refresh();
                return;
            }
        }
    }

    //saving ticket record file in local directory using saveFileDialog
    private void btnSaveTicket_Click(object sender, EventArgs e)
    {
        SaveFileDialog save = new SaveFileDialog();
        save.Filter = "CSV file|*.csv;";
        save.DefaultExt = ".csv";
        save.FileName = "Ticket Rate " + DateTime.Now.ToString("dd-MM-yyyy");
        if (save.ShowDialog() == DialogResult.OK)
        {
            StreamWriter swr = new StreamWriter(save.FileName);
            StringBuilder sb = new StringBuilder();
            sb.AppendLine("Category, 1Hr, 2Hrs, 3Hrs, 4Hrs, Whole Day");
            foreach (TicketRate ticket in ticketRecord)
            {
                sb.AppendLine(index + "," + ticket.category + "," +
                    ticket.oneHr.ToString() + "," +
                    ticket.twoHrs.ToString() + "," +
                    ticket.threeHrs.ToString() + "," +
                    ticket.fourHrs.ToString() + "," +
                    ticket.wholeDay.ToString());
```

```
            }

            swr.Write(sb.ToString());
            swr.Close();
            MessageBox.Show("Ticket rate saved successfully.", "Saved",
MessageBoxButtons.OK);

        }
    }

    //unloading dataGridView
    private void btnUnloadT_Click(object sender, EventArgs e)
    {
        lblFileNameT.Text = "File name: ";
        dgvTicketRate.Rows.Clear();
        dgvTicketRate.Refresh();
    }

    private void Dashboard_FormClosing_1(object sender, FormClosingEventArgs e)
    {
        //saving object state and file using serialization
        FileStream fs = new FileStream("Visitor_record.csv", FileMode.Create);
        BinaryFormatter bf = new BinaryFormatter();
        bf.Serialize(fs, visitorRecord);
        fs.Close();

        FileStream fst = new FileStream("Ticket_rate.csv", FileMode.Create);
        BinaryFormatter bft = new BinaryFormatter();
        bft.Serialize(fst, ticketRecord);
        fst.Close();
    }
```

```
private void cbSortBy_SelectedIndexChanged(object sender, EventArgs e)
{
    sortRecord();
}

//Bubble sort implementation for sorting records
private void sortRecord()
{

    for (int i = 0; i < visitorRecord.Count - 1; i++)
        for (int j = 0; j < visitorRecord.Count - 1; j++)
        {
            switch (cbSortBy.SelectedIndex)
            {

                case 0:
                    if (visitorRecord[j].name.CompareTo(visitorRecord[j + 1].name) > 0)
                    {
                        swap(j, j + 1);
                    }
                    break;

                case 1:
                    if (visitorRecord[j].date.CompareTo(visitorRecord[j + 1].date) > 0)
                    {
                        swap(j, j + 1);
                    }
                    break;
            }
```

```
            }
        dgvVisitors.DataSource = visitorRecord;
    }

    private void swap(int i, int j) //swap for sorting
    {
        Visitors temp;
        temp = visitorRecord[i];
        visitorRecord[i] = visitorRecord[j];
        visitorRecord[j] = temp;
    }

    //ticket price input validation using regex pattern
    private void tb1Hr_TextChanged(object sender, EventArgs e)
    {
        if (System.Text.RegularExpressions.Regex.IsMatch(tb1Hr.Text, "[^0-9]"))
        {
            MessageBox.Show("Please enter valid ticket price.", "Invalid input",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            tb1Hr.Text = tb1Hr.Text.Remove(tb1Hr.Text.Length - 1);
        }
    }

    private void tb2Hrs_TextChanged(object sender, EventArgs e)
    {
        if (System.Text.RegularExpressions.Regex.IsMatch(tb2Hrs.Text, "[^0-9]"))
        {
            MessageBox.Show("Please enter valid ticket price.", "Invalid input",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            tb2Hrs.Text = tb2Hrs.Text.Remove(tb2Hrs.Text.Length - 1);
        }
```

```
    }

    private void tb3Hrs_TextChanged(object sender, EventArgs e)
    {
        if (System.Text.RegularExpressions.Regex.IsMatch(tb3Hrs.Text, "[^0-9]"))
        {
            MessageBox.Show("Please enter valid ticket price.", "Invalid input",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            tb3Hrs.Text = tb3Hrs.Text.Remove(tb3Hrs.Text.Length - 1);
        }
    }

    private void tb4Hrs_TextChanged(object sender, EventArgs e)
    {
        if (System.Text.RegularExpressions.Regex.IsMatch(tb4Hrs.Text, "[^0-9]"))
        {
            MessageBox.Show("Please enter valid ticket price.", "Invalid input",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            tb4Hrs.Text = tb4Hrs.Text.Remove(tb4Hrs.Text.Length - 1);
        }
    }

    private void tbWholeDay_TextChanged(object sender, EventArgs e)
    {
        if (System.Text.RegularExpressions.Regex.IsMatch(tbWholeDay.Text, "[^0-9]"))
        {
            MessageBox.Show("Please enter valid ticket price.", "Invalid input",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            tbWholeDay.Text = tbWholeDay.Text.Remove(tbWholeDay.Text.Length - 1);
        }
    }
```

```
    }
}

[Serializable]
public class Visitors  //visitor class for visitor records
{
    //defining Visitor class attributes with getter and setter methods.
    private int ticketNo;
    public int ticketNO
    {
        get {
            return ticketNo;
        }
        set {
            ticketNo = value;
        }
    }
    public String name { get; set; }
    public String address { get; set; }
    public Double phoneNo { get; set; }
    public String gender { get; set; }
    public String ticketType { get; set; }
    public DateTime date { get; set; }
    public DateTime entryTime { get; set; }
    public DateTime exitTime { get; set; }
    public String totalTime { get; set; }
    public int price { get; set; }

    //Constructor for setting visitor's details
```

```
    public Visitors(int ticketNo, String name, String address, Double phoneNo, String gender, String ticketType, DateTime date, DateTime entryTime, DateTime exitTime, String totalTime, int price)
    {
        setVisitor(ticketNo, name, address, phoneNo, gender, ticketType, date, entryTime, exitTime, totalTime, price);
    }

    //defining setVisitor method to set data in defined attributes.
    public void setVisitor(int ticketNo, String name, String address, Double phoneNo, String gender, String ticketType, DateTime date, DateTime entryTime, DateTime exitTime, String totalTime, int price)
    {
        this.ticketNo = ticketNo;
        this.name = name;
        this.address = address;
        this.phoneNo = phoneNo;
        this.gender = gender;
        this.ticketType = ticketType;
        this.date = date;
        this.entryTime = entryTime;
        this.exitTime = exitTime;
        this.totalTime = totalTime;
        this.price = price;
    }

}

[Serializable]
public class TicketRate  //TicketRate class for Tickets Price
{
```

```
    //defining TicketRate class attributes with getter and setter methods.
    private int index;
    public int Index
    {
        get
        {
            return index;
        }
        set
        {
            index = value;
        }
    }
    public String category { get; set; }
    public int oneHr { get; set; }
    public int twoHrs { get; set; }
    public int threeHrs { get; set; }
    public int fourHrs { get; set; }
    public int wholeDay { get; set; }

    //Constructor for setting TicketRate details
    public TicketRate(int index, String category, int oneHr, int twoHrs, int threeHrs, int fourHrs, int wholeDay)
    {
        setTicketRate(index, category, oneHr, twoHrs, threeHrs, fourHrs, wholeDay);
    }

    //defining setVisitor method to set data in defined attributes.
    public void setTicketRate(int index, String category, int oneHr, int twoHrs, int threeHrs, int fourHrs, int wholeDay)
    {
```

```
        this.index = index;
        this.category = category;
        this.oneHr = oneHr;
        this.twoHrs = twoHrs;
        this.threeHrs = threeHrs;
        this.fourHrs = fourHrs;
        this.wholeDay = wholeDay;
    }

}
```

**Class: AddRecord.cs**

```
//Written by: Susan Shrestha
//Date: 29 December, 2021
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Formatters.Binary;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Recreation_Center
{
    public partial class AddRecord : Form
    {
```

```
    public static BindingList<TicketRate> ticketRecord = new BindingList<TicketRate>(); //initializing new binding list with the type of Visitor
    private BindingList<string> alreadyOpenedTicket = new BindingList<string>();

    public bool editMode { get; set; }

    public Double totalTime = 0;
    public int totalAmount = 0;

    public AddRecord()
    {
        InitializeComponent();
    }

    private void btnSave_Click(object sender, EventArgs e)
    {
        if (tbName.Text == "" || tbAddress.Text == "" || tbPhone.Text == "" || cbGender.Text == "" || cbTicketType.Text == "" || tbTicketNo.Text == "" || datePicker.Text == "" || dtEntryTime.Text == "" || tbPrice.Text == "")
        {
            MessageBox.Show("Please fill the required (*) feilds.", "Invalid Input", MessageBoxButtons.OK);
        }
        else
        {
            try
            {
                if(!editMode)
                {
                    Visitors visitor = new Visitors(Convert.ToInt32(tbTicketNo.Text),
                        tbName.Text,
```

```
                    tbAddress.Text,
                    Convert.ToDouble(tbPhone.Text),
                    cbGender.Text,
                    cbTicketType.Text,
                    datePicker.Value,
                    dtEntryTime.Value,
                    dtExitTime.Value,
                    tbTotalTIme.Text,
                    Convert.ToInt32(tbPrice.Text));

((Dashboard)Application.OpenForms["Dashboard"]).addVisitorRecord(visitor);
                tbTicketNo.Text = "";
                tbName.Text = "";
                tbAddress.Text = "";
                tbPhone.Text = "";
                cbGender.Text = "";
                cbTicketType.Text = "";
                tbPrice.Text = "";
            }
            else
            {
                ((Dashboard)Application.OpenForms["Dashboard"]).editVisitorRecord(
                    Convert.ToInt32(tbTicketNo.Text),
                    tbName.Text,
                    tbAddress.Text,
                    Convert.ToDouble(tbPhone.Text),
                    cbGender.Text, cbTicketType.Text,
                    datePicker.Value,
                    dtEntryTime.Value,
                    dtExitTime.Value,
                    tbTotalTIme.Text,
```

```
                    Convert.ToInt32(tbPrice.Text)
                );
            }
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message,"Error");
        }
    }
}
```

public void setData(int ticketNo, String name, String address, Double phoneNo, String gender, String ticketType, DateTime date, DateTime entryTime, DateTime exitTime, String totalTime, int price)

```
{
    tbTicketNo.Text = ticketNo.ToString();
    tbTicketNo.ReadOnly = true;
    tbName.Text = name;
    tbAddress.Text = address;
    tbPhone.Text = phoneNo.ToString();
    cbGender.Text = gender;
    cbTicketType.Text = ticketType;
    datePicker.Value = date;
    dtEntryTime.Value = entryTime;
    dtExitTime.Value = exitTime;
    tbTotalTIme.Text = totalTime;
    tbPrice.Text = price.ToString();
}

private void AddRecord_Load(object sender, EventArgs e)
{
```

```
        try
        {
            if (File.Exists("Ticket_rate.csv")) //checking existing file to load in the GridView
            {
                FileStream fst = new FileStream("Ticket_rate.csv", FileMode.Open);
                BinaryFormatter bft = new BinaryFormatter();
                ticketRecord = (BindingList<TicketRate>)bft.Deserialize(fst); //Deserializing
BindingList
                fst.Close();
            }
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }

        dgvTicketList.DataSource = ticketRecord;
        dgvTicketList.Columns["Index"].AutoSizeMode                                    =
DataGridViewAutoSizeColumnMode.Fill;

        if (!editMode)
        {
            btnSave.Text = "Add Record";
            dtEntryTime.Text = DateTime.Now.ToString("hh:mm tt");

        }
        else
        {
            btnSave.Text = "Edit Record";
            dtExitTime.Text = DateTime.Now.ToString("hh:mm tt");

        }
    }
```

```
//validation using regex pattern for number input only
private void tbPhone_TextChanged(object sender, EventArgs e)
{
   if (System.Text.RegularExpressions.Regex.IsMatch(tbPhone.Text, "[^0-9]"))
   {
      MessageBox.Show("Please enter valid phone number.", "Invalid input", MessageBoxButtons.OK, MessageBoxIcon.Error);
      tbPhone.Text = tbPhone.Text.Remove(tbPhone.Text.Length - 1);
   }
}

private void tbTicketNo_TextChanged(object sender, EventArgs e)
{
   if (System.Text.RegularExpressions.Regex.IsMatch(tbTicketNo.Text, "[^0-9]"))
   {
      MessageBox.Show("Please enter valid ticket number.", "Invalid input", MessageBoxButtons.OK, MessageBoxIcon.Error);
      tbTicketNo.Text = tbTicketNo.Text.Remove(tbTicketNo.Text.Length - 1);
   }
}

private void tbPrice_TextChanged(object sender, EventArgs e)
{
   if (System.Text.RegularExpressions.Regex.IsMatch(tbPrice.Text, "[^0-9]"))
   {
      MessageBox.Show("Please enter valid price amount.", "Invalid input", MessageBoxButtons.OK, MessageBoxIcon.Error);
      tbPrice.Text = tbPrice.Text.Remove(tbPrice.Text.Length - 1);
   }
}
```

```
private void btnClearForm_Click(object sender, EventArgs e)
{
    tbName.Text = "";
    tbAddress.Text = "";
    tbPhone.Text = "";
    cbGender.Text = default;
    cbTicketType.Text = default;
    tbTicketNo.Text = "";
    tbTotalTIme.Text = "0 hr";
    tbPrice.Text = "";
}


private void chbDiscount_CheckedChanged(object sender, EventArgs e)
{
    try
    {
        int getPrice = Convert.ToInt32(tbPrice.Text);
        int discount = Convert.ToInt32(0.1 * getPrice);
        if (chbDiscount.Checked)
        {
            int discountedPrice = getPrice - discount;
            tbPrice.Text = discountedPrice.ToString();
        }
    }
    catch (Exception) { MessageBox.Show("Format Error", "Error."); }

}

private void tbPrice_Click(object sender, EventArgs e)
```

```
{
    chbDiscount.Checked = false;

}

private void dtExitTime_ValueChanged(object sender, EventArgs e)
{
    TimeSpan duration = dtExitTime.Value - dtEntryTime.Value;
    tbTotalTIme.Text = duration.ToString(@"h\.mm") + " hours";
}

private void btnUnload_Click(object sender, EventArgs e)
{
    lblFileName.Text = "File name: ";
    dgvTicketList.Rows.Clear();
    dgvTicketList.Refresh();
}

private void btnImport_Click(object sender, EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    ofd.Filter = "CSV File|*.csv";
    if (ofd.ShowDialog() == DialogResult.OK)
    {
        if (!(alreadyOpenedTicket.Contains(ofd.FileName)))
        {

            string[] lines = File.ReadAllLines(ofd.FileName);
            // filename for validation
            alreadyOpenedTicket.Add(ofd.FileName);
            string[] items;
```

```
            bool firstline = true;

            lines.Skip<string>(1);

            foreach (string line in lines)
            {
                items = line.Split(',');
                if (firstline == true)
                {
                    firstline = false;
                }
                else
                {
                    try
                    {
                        TicketRate ticket = new TicketRate(Convert.ToInt32(items[0]), items[1], Convert.ToInt32(items[2]), Convert.ToInt32(items[3]), Convert.ToInt32(items[4]), Convert.ToInt32(items[5]), Convert.ToInt32(items[6]));
                        lblFileName.Text = "File name: " + ofd.SafeFileName; //file name of imported file
                        addTicketRecord(ticket);
                        dgvTicketList.Columns["category"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
                    }
                    catch (Exception)
                    {
                        MessageBox.Show("Please import a valid CSV file.", "Invalid file import", MessageBoxButtons.OK);
                        break;
                    }
```

```
                }

            }
        }
        else
        {
            MessageBox.Show("ERROR", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);

        }
    }
}

public void addTicketRecord(TicketRate e)
{
    ticketRecord.Add(e);
    dgvTicketList.DataSource = ticketRecord;
}
}
}
```