

**MS806 – Business Application Programming**

**Student Name: Jayakarthi Boovendran**

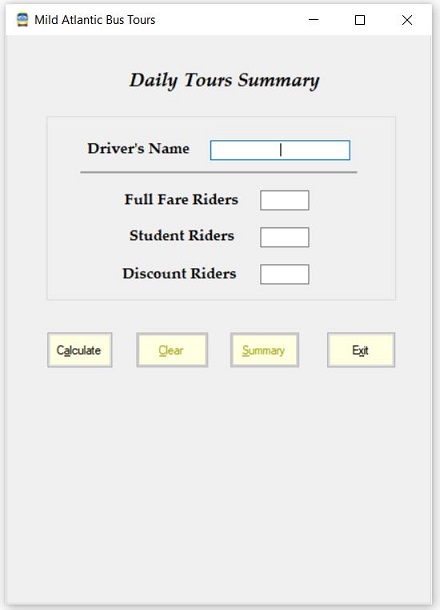
**NUI Galway ID Number: 19230487**

**Assignment:** 2

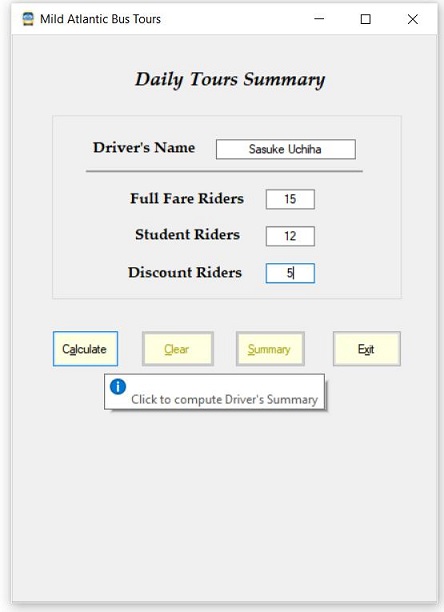
**Date of Submission:** **03/10/2019**

**Assignment II: Mild Atlantic Bus Tours**

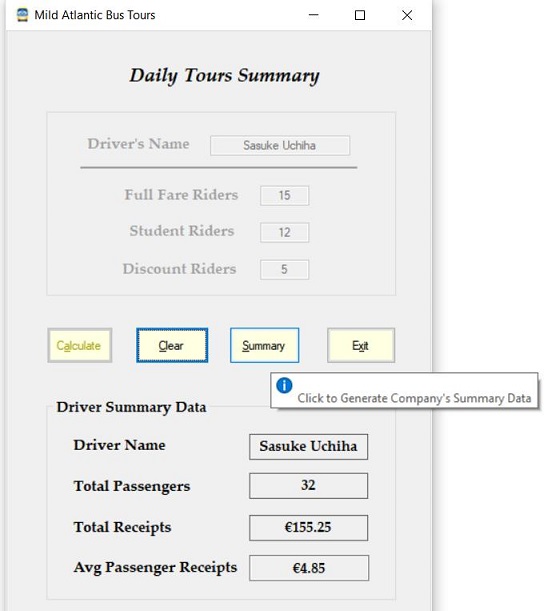
**Screenshots of Application at Runtime:**



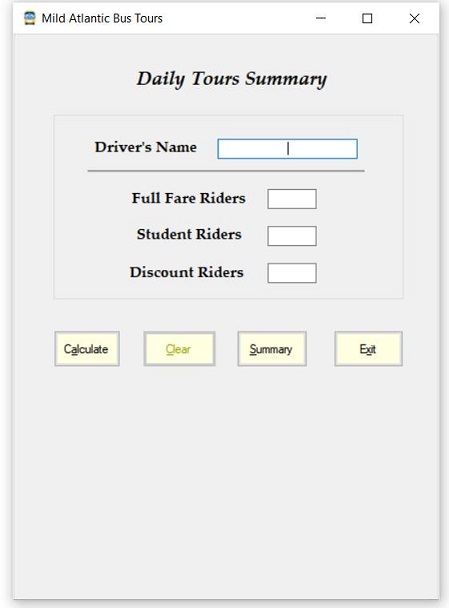
**Fig 1: On Form Load**



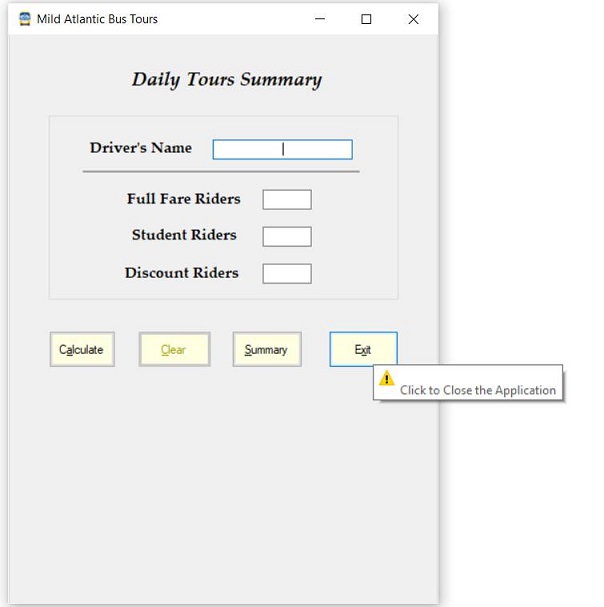
**Fig 2: Form with Driver 1’s Data**



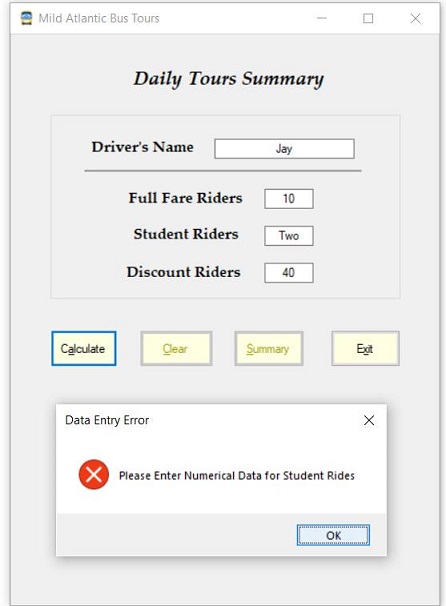
**Fig 3: After ‘Calculate’ button is clicked**



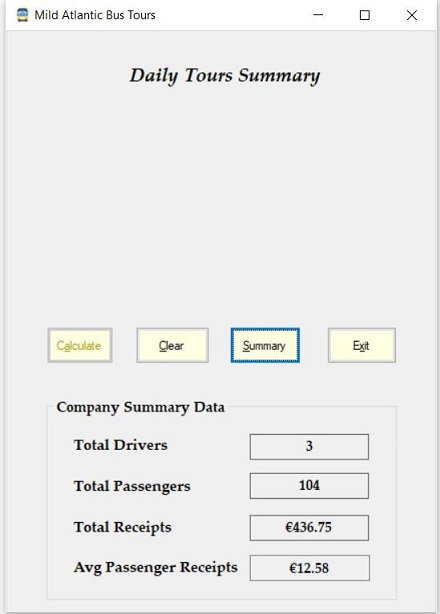
**Fig 4: After ‘Clear’ button is clicked**



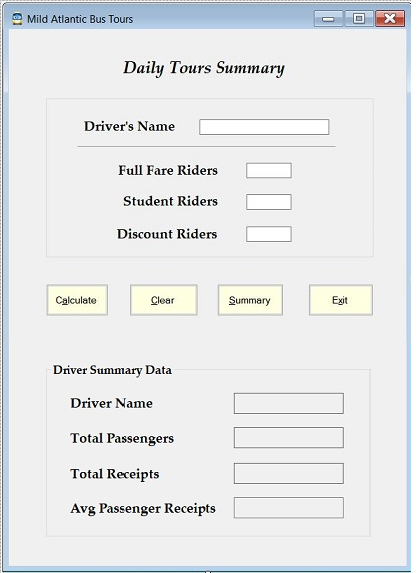
**Fig 5: ‘Exit’ Button and its tooltip**



**Fig 6: Error Handling**

****

**Fig 7: After ‘Summary’ Button is clicked**



**Fig 8: Application Tab Order**

**Application Source Code:**

/\*

## MS806 - Business Application Programming

## Assignment No : 2

## Name : Jayakarthi Boovendran | ## StudentID : 19230487

## Date of Submission : October 03, 2019

## Assignment Description : To Build a Window Form for 'Mild Atlantic Bus Tours' to Keep track on the Ride Information of Drivers and to compute overall Company's Summary and Summary of individual Drivers. ## The Form takes in Driver's info like Name, No.of.Full Fare Rides,No.of.Student Rides and No.of.Discounted Rides as input,and associated Receipts are computed and reported as Output. \*/

using System;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using System.Collections.Generic;

namespace MildAtlanticBusTour

{

public partial class AtlanticBusToursForm : Form

{

// Initializations

const string EUR = "€";

const double FullRideFare = 5.50, StudentFare = 4.50, DiscountedFare = 3.75;

ConsolidatedTour ConsolidatedTourInfo = new ConsolidatedTour();

// Initializes Form Controls

// Loads the Initial Form Page

public AtlanticBusToursForm()

{

InitializeComponent();

InitializeAtlanticBusPortal();

}

#region Form Initialization

// Initializes Form Page to its Pristine State

public void InitializeAtlanticBusPortal()

{

RideInfoGroupBox.Visible = true;

RideInfoGroupBox.Enabled = true;

DriverNameInput.Text = string.Empty;

FullFareRidesInput.Text = string.Empty;

StudentRidesInput.Text = string.Empty;

DiscountRidesInput.Text = string.Empty;

CalculateButton.Visible = true;

CalculateButton.Enabled = true;

ClearButton.Visible = true;

ClearButton.Enabled = false;

// Atleast a Driver's Calculated Data is required to Enable the 'Summary'

SummaryButton.Visible = true;

if (ConsolidatedTourInfo.ConsolidatedTourSummary.Count <= 0)

{

SummaryButton.Enabled = false;

}

else

{

SummaryButton.Enabled = true;

}

ExitButton.Visible = true;

SummaryGroupBox.Visible = false;

DriverNameInput.Focus();

}

// Displays Driver / Company Summary Section

public void ShowSummarySection()

{

RideInfoGroupBox.Visible = true;

CalculateButton.Visible = true;

ClearButton.Visible = true;

ClearButton.Enabled = true;

SummaryButton.Visible = true;

ExitButton.Visible = true;

SummaryGroupBox.Visible = true;

SummaryGroupBox.Text = "Driver Summary Data";

SummaryInfoDriverNameLabel.Text = "Driver Name";

}

#endregion

#region User Input Validation

// Validates User Inputs

// Exception Handling is implemented to handle Type-casting Errors at run-time

public bool InputValidation()

{

string FieldName = string.Empty;

int TempFullFareRides, TempStudentRides, TempDiscountedRides;

try

{

FieldName = "Full Fare Rides";

TempFullFareRides = int.Parse(FullFareRidesInput.Text);

FieldName = "Student Rides";

TempStudentRides = int.Parse(StudentRidesInput.Text);

FieldName = "Discounted Rides";

TempDiscountedRides = int.Parse(DiscountRidesInput.Text);

FieldName = string.Empty;

return true;

}

catch (Exception)

{

//Data Validation Error Message is Displayed along with the Field Name

ShowValidationError(FieldName);

return false;

}

}

// Displays Data Validation Error Message

public void ShowValidationError(string FieldName)

{

MessageBox.Show("Please Enter Numerical Data for " + FieldName + "", "Data Entry Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

#endregion

#region Calculate

// Computes Summary of Rides of an Individual Driver

// Adds the Driver's Summary to the Company's Summary

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

{

try

{

if (InputValidation())

{

// Computes and Saves Calculated fields of a Ride

SaveRideInfo();

Tour Ride = new Tour();

Ride = ConsolidatedTourInfo.ConsolidatedTourSummary.Last();

SummaryInfoDriverName.Text = Ride.DriverName;

SummaryInfoTotalPassengers.Text = Ride.TotalPassengers.ToString();

SummaryInfoTotalReceipts.Text = EUR + Ride.TotalReceipts.ToString("f2");

SummaryInfoAvgPassengerReceipts.Text = EUR + Ride.AvgPassengerReceipts.ToString("f2");

SummaryInfoDriverNameLabel.Text = "Driver Name";

ShowSummarySection();

RideInfoGroupBox.Enabled = false;

CalculateButton.Enabled = false;

if (ConsolidatedTourInfo.ConsolidatedTourSummary.Count > 0) {

SummaryButton.Enabled = true;

}

else {

SummaryButton.Enabled = false;

}

}

}

catch (Exception)

{

// On Exception Clears the Form, and sets to its initial state

InitializeAtlanticBusPortal();

}

}

// Saves the computed values of Individual Ride

public void SaveRideInfo()

{

Tour RideInfo = new Tour();

RideInfo.DriverName = DriverNameInput.Text;

RideInfo.FullFareRides = Convert.ToInt32(FullFareRidesInput.Text);

RideInfo.StudentRides = Convert.ToInt32(StudentRidesInput.Text);

RideInfo.DiscountRides = Convert.ToInt32(DiscountRidesInput.Text);

RideInfo.TotalPassengers = RideInfo.FullFareRides + RideInfo.StudentRides + RideInfo.DiscountRides;

RideInfo.TotalReceipts = Math.Round(((RideInfo.FullFareRides \* FullRideFare) + (RideInfo.StudentRides \* StudentFare) + (RideInfo.DiscountRides \* DiscountedFare)), 2);

RideInfo.AvgPassengerReceipts = Math.Round((RideInfo.TotalReceipts / RideInfo.TotalPassengers), 2);

// Add Ride Info to Company's Summary

SaveTourInfo(RideInfo);

}

// Adds consolidated Info of all the Rides

public void SaveTourInfo(Tour RideInfo)

{

ConsolidatedTourInfo.ConsolidatedTourSummary.Add(RideInfo);

}

#endregion

#region Summary

// Generates Company's Summary from the list of Driver's Summary

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

{

GenerateTourSummary();

ShowSummarySection();

SummaryInfoDriverNameLabel.Text = "Total Drivers";

SummaryGroupBox.Text = "Company Summary Data";

RideInfoGroupBox.Visible = false;

CalculateButton.Enabled = false;

}

// Computes Company's Summary from the list of Driver Summaries

// Made use of Linq to Compute Total Drivers, Total Passengers, Total Receipts and Avg Passenger Receipt

public void GenerateTourSummary()

{

try

{

SummaryInfoDriverName.Text = ConsolidatedTourInfo.ConsolidatedTourSummary.Select(x => x.DriverName).Distinct().Count().ToString();

SummaryInfoTotalPassengers.Text = ConsolidatedTourInfo.ConsolidatedTourSummary.Select(x => x.TotalPassengers).Sum().ToString();

SummaryInfoTotalReceipts.Text = EUR + Math.Round(Convert.ToDouble(ConsolidatedTourInfo.ConsolidatedTourSummary.Select(x => x.TotalReceipts).Sum()), 2).ToString("f2");

SummaryInfoAvgPassengerReceipts.Text = EUR + Math.Round(Convert.ToDouble(ConsolidatedTourInfo.ConsolidatedTourSummary.Select(x => x.AvgPassengerReceipts).Sum()), 2).ToString("f2");

}

catch (Exception)

{

InitializeAtlanticBusPortal();

}

}

#endregion

#region Exit

// Exits the Application

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

{

Close();

}

#endregion

#region Clear

// Clears User Data and sets the Form back to its Pristine state

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

{

InitializeAtlanticBusPortal();

}

#endregion

}

#region Models

// Model to store Individual Ride Summary of a Driver

public class Tour

{

public string DriverName { get; set; }

public int FullFareRides { get; set; }

public int StudentRides { get; set; }

public int DiscountRides { get; set; }

public int TotalPassengers { get; set; }

public double TotalReceipts { get; set; }

public double AvgPassengerReceipts { get; set; }

}

// Model to hold Consolidated Summary of All the Rides

public class ConsolidatedTour

{

public IList<Tour> ConsolidatedTourSummary = new List<Tour>();

}

#endregion

}