/\*

## MS806 - Business Application Programming

## Assignment No : 5

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

## Date of Submission : December 01, 2019

## Assignment Description : To Build a Window Form for 'Team Builder Ltd' to enable users to book for events at different locations for several places .

## The Form takes in user's info like Event Name, Event Location, Meal Plan as input,

and associated Reports are computed and saved in a file. \*/

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using Newtonsoft.Json;

using System.Configuration;

using System.IO;

using System.Diagnostics;

namespace TeamBuilderExtended

{

public partial class TeamBuilderV2 : Form

{

MetaEvent MetaData = new MetaEvent();

BookSummary Summary = new BookSummary();

public TeamBuilderV2()

{

InitializeComponent();

//Loads HomePage

InitializeHomePage();

}

#region InitializeHomePage

public void InitializeHomePage()

{

HomePageGroupBox.Visible = true;

//Loads MetaDate from App.Config and File Storage onto Modals

GetMetaData();

//Loads EventInfo, LocationInfo and MealInfo DataGrids with Apppropriate Metadata

LoadEventsGridData();

LoadLocationGridData();

LoadMealInfoGrid();

//Loads BookingInfo Grid

InitializeBookingInfoGrid();

EventInfoGridView.ClearSelection();

LocationInfoGridView.ClearSelection();

MealPlanGridView.ClearSelection();

}

#endregion

#region GetMetaData

public void GetMetaData()

{

string FileName = string.Empty;

try

{

// Gets MetaData from App.Config

MetaData.EventData = JsonConvert.DeserializeObject<IList<Event>>(ConfigurationManager.AppSettings.Get("EventInfo").ToString());

MetaData.EventLocationData = JsonConvert.DeserializeObject<IList<EventLocation>>(ConfigurationManager.AppSettings.Get("LocationInfo").ToString());

MetaData.MealData = JsonConvert.DeserializeObject<IList<MealInfo>>(ConfigurationManager.AppSettings.Get("MealInfo").ToString());

MetaData.PrincingInfo = JsonConvert.DeserializeObject<IList<PricingInfo>>(ConfigurationManager.AppSettings.Get("PricingInfo").ToString());

//Gets Closing Stock Info from File

FileName = ConfigurationManager.AppSettings.Get("StockFileName").ToString();

if (!File.Exists(FileName))

{

MetaData.StockData = JsonConvert.DeserializeObject<IList<StockInfo>>(ConfigurationManager.AppSettings.Get("StockInfo").ToString());

}

else

{

// Loads Stock Info from App.Config if StockData.txt is not available

// Suits when the application is running for the first time and there is no previous closing Stock data available

foreach (string data in File.ReadAllLines(FileName))

{

MetaData.StockData = JsonConvert.DeserializeObject<IList<StockInfo>>(data);

break;

}

}

}

catch

{

MessageBox.Show("Error Loading MetaData, Contact IT Team!", "Temporary Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#region Load Grid Data

#region Load Event Data

public void LoadEventsGridData()

{

// Loads Event Details like Event\_Name, No. of Event Days, Event\_Registration\_Fee into the DataGrid

string GridColumnHeader = string.Empty;

try

{

DataTable dataTable = new DataTable();

GridColumnHeader = ConfigurationManager.AppSettings.Get("EventInfoGridHeader").ToString();

foreach (string Column in GridColumnHeader.Split('|'))

{

dataTable.Columns.Add(Column, typeof(string));

}

foreach (Event \_event in MetaData.EventData)

{

dataTable.Rows.Add(\_event.EventID.ToString(), \_event.EventName, \_event.Days.ToString(), "€ " + \_event.RegistrationFee.ToString("f2"));

}

EventInfoGridView.DataSource = dataTable;

EventInfoGridView.Rows[0].Selected = true;

}

catch (Exception)

{

MessageBox.Show("Error Loading EventInfo !", "System Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#region Load Location Data

public void LoadLocationGridData()

{

// Loads Location info like Location\_Name and Lodging\_Fee into the Grid

string LocationInfoGridHeader = string.Empty;

int \_days = 0;

try

{

DataTable dataTable = new DataTable();

LocationInfoGridHeader = ConfigurationManager.AppSettings.Get("LocationInfoGridHeader").ToString();

foreach (string column in LocationInfoGridHeader.Split('|'))

{

dataTable.Columns.Add(column, typeof(string));

}

\_days = Convert.ToInt32(EventInfoGridView.SelectedRows[0].Cells[2].Value);

foreach (EventLocation location in MetaData.EventLocationData)

{

dataTable.Rows.Add(location.LocationID.ToString(), location.Location, "€ " + location.LodgingFee.ToString("f2"), "€ " + (location.LodgingFee \* \_days).ToString("f2"));

}

LocationInfoGridView.DataSource = dataTable;

LocationInfoGridView.ClearSelection();

}

catch (Exception)

{

MessageBox.Show("Error Loading Event Location Info !", "System Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#region Load Meal Info

public void LoadMealInfoGrid()

{

int EventID = 0;

string MealInfoGridHeader = string.Empty;

try

{

DataTable datatable = new DataTable();

MealInfoGridHeader = ConfigurationManager.AppSettings.Get("MealInfoGridHeader").ToString();

foreach (string column in MealInfoGridHeader.Split('|'))

{

datatable.Columns.Add(column, typeof(string));

}

EventID = Convert.ToInt32(EventInfoGridView.SelectedRows[0].Cells[0].Value);

datatable.Rows.Add("1", "Full Board", "€ " + MetaData.MealData.Where(x => x.EventID == EventID).Select(x => x.Full).FirstOrDefault().ToString());

datatable.Rows.Add("2", "Half Board", "€ " + MetaData.MealData.Where(x => x.EventID == EventID).Select(x => x.Half).FirstOrDefault().ToString());

datatable.Rows.Add("3", "Breakfast", "€ " + MetaData.MealData.Where(x => x.EventID == EventID).Select(x => x.Breakfast).FirstOrDefault().ToString());

datatable.Rows.Add("4", "No Meal", "€ " + MetaData.MealData.Where(x => x.EventID == EventID).Select(x => x.NoMeal).FirstOrDefault().ToString());

MealPlanGridView.DataSource = datatable;

MealPlanGridView.ClearSelection();

}

catch (Exception)

{

MessageBox.Show("Error Loading Meal Info !", "System Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#region Load Booking Info

public void InitializeBookingInfoGrid()

{

//Initializes booking Info Grid

string BookingInfoGridHeader = string.Empty;

DataTable dataTable = new DataTable();

BookingInfoGridHeader = ConfigurationManager.AppSettings.Get("BookingInfoGridHeader").ToString();

foreach (string column in BookingInfoGridHeader.Split('|'))

{

dataTable.Columns.Add(column, typeof(string));

}

BookingInfoGridView.DataSource = dataTable;

}

#endregion

#endregion

#region Grid Event Handling

// Loads appropriate MetaData on Change of Event Selection

private void EventInfoGridSelectionChanged(object sender, EventArgs e)

{

if (EventInfoGridView.SelectedRows.Count != 0)

{

LoadLocationGridData();

LoadMealInfoGrid();

InitializeBookingInfo();

}

}

// Loads appropriate MetaData on Change of Location

private void LocationInfoGridSelectionChanged(object sender, EventArgs e)

{

if (LocationInfoGridView.SelectedRows.Count != 0)

{

InitializeBookingInfo();

}

}

// Loads appropriate MetaData on Change of Meal Plan

private void MealPlanGridViewSelectionChanged(object sender, EventArgs e)

{

if (MealPlanGridView.SelectedRows.Count > 0)

{

InitializeBookingInfo();

}

}

#endregion

#region Clear Selection

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

{

//Clears All Selections

EventInfoGridView.ClearSelection();

LocationInfoGridView.ClearSelection();

MealPlanGridView.ClearSelection();

BookingInfoGridView.DataSource = new DataTable();

}

#endregion

#region InputValidation

public bool InputValidation()

{

//Validates if Event,Location and Meal Plan are selected

//Note : Multiple Selections are disables in the all the DataGrids

if (EventInfoGridView.SelectedRows.Count == 0)

{

MessageBox.Show("Please select an Event!", "Input Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

if (LocationInfoGridView.SelectedRows.Count == 0)

{

MessageBox.Show("Please select a Location!", "Input Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

if (MealPlanGridView.SelectedRows.Count == 0)

{

MessageBox.Show("Please select a Meal Plan!", "Input Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

return true;

}

#endregion

#region Load Booking Info Grid

public void InitializeBookingInfo()

{

BookingInfoGridView.Visible = true;

string BookingInfoGridHeader = string.Empty;

string EventName = string.Empty;

int EventID = 0, defaultbookingCount = 1;

string Availability = string.Empty;

int LocationID = 0;

decimal CostPerBooking = 0m;

int ColumnCount = 0; string MealPlan = string.Empty;

try

{

DataTable dataTable = new DataTable();

BookingInfoGridHeader = ConfigurationManager.AppSettings.Get("BookingInfoGridHeader").ToString();

foreach (string column in BookingInfoGridHeader.Split('|'))

{

dataTable.Columns.Add(column, typeof(string));

}

// If Only Event is selected

if (EventInfoGridView.SelectedRows.Count != 0 && LocationInfoGridView.SelectedRows.Count == 0 && MealPlanGridView.SelectedRows.Count == 0)

{

EventName = EventInfoGridView.SelectedRows[0].Cells[1].Value.ToString();

EventID = Convert.ToInt32(EventInfoGridView.SelectedRows[0].Cells[0].Value);

CostPerBooking = Math.Round(Convert.ToDecimal(MetaData.EventData.Where(x => x.EventID == EventID).Select(x => x.RegistrationFee).FirstOrDefault()), 2);

dataTable.Rows.Add(EventName, '-', '-', "€ " + CostPerBooking, '-', "€ " + CostPerBooking);

}

else if (EventInfoGridView.SelectedRows.Count != 0 && LocationInfoGridView.SelectedRows.Count != 0 && MealPlanGridView.SelectedRows.Count == 0)

{

//If only Event and Location is selected

EventName = EventInfoGridView.SelectedRows[0].Cells[1].Value.ToString();

EventID = Convert.ToInt32(EventInfoGridView.SelectedRows[0].Cells[0].Value);

foreach (DataGridViewRow row in LocationInfoGridView.SelectedRows)

{

LocationID = Convert.ToInt32(row.Cells[0].Value);

Availability = MetaData.StockData.Where(x => x.EventID == EventID).Select(x => x).FirstOrDefault().AvailabilityInfo.Where(y => y.LocationID == LocationID).Select(y => y.Availability).FirstOrDefault().ToString();

CostPerBooking = Math.Round((Convert.ToDecimal(MetaData.EventData.Where(x => x.EventID == EventID).Select(x => x.RegistrationFee).FirstOrDefault()) + Convert.ToDecimal(row.Cells[3].Value.ToString().Replace("€", string.Empty).Trim())), 2);

// Availability Check

if (Convert.ToInt32(Availability) == 0)

{

MessageBox.Show("No Places available for '" + EventName + "' @" + row.Cells[1].Value.ToString() + "'", "Transaction Aborted!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

defaultbookingCount = 1;

dataTable.Rows.Add(EventName, row.Cells[1].Value, Availability, "€ " + CostPerBooking, defaultbookingCount, "€ " + CostPerBooking);

}

EventName = string.Empty;

}

}

else if (EventInfoGridView.SelectedRows.Count != 0 && LocationInfoGridView.SelectedRows.Count != 0 && MealPlanGridView.SelectedRows.Count != 0)

{

//If Event, Location and Meal Plan is selected

EventName = EventInfoGridView.SelectedRows[0].Cells[1].Value.ToString();

EventID = Convert.ToInt32(EventInfoGridView.SelectedRows[0].Cells[0].Value);

MealPlan = MealPlanGridView.SelectedRows[0].Cells[1].Value.ToString();

foreach (DataGridViewRow row in LocationInfoGridView.SelectedRows)

{

LocationID = Convert.ToInt32(row.Cells[0].Value);

Availability = MetaData.StockData.Where(x => x.EventID == EventID).Select(x => x).FirstOrDefault().AvailabilityInfo.Where(y => y.LocationID == LocationID).Select(y => y.Availability).FirstOrDefault().ToString();

CostPerBooking = Math.Round((Convert.ToDecimal(MetaData.EventData.Where(x => x.EventID == EventID).Select(x => x.RegistrationFee).FirstOrDefault()) + Convert.ToDecimal(row.Cells[3].Value.ToString().Replace("€", string.Empty).Trim()) + Convert.ToDecimal(MealPlanGridView.SelectedRows[0].Cells[2].Value.ToString().Replace("€", string.Empty).Trim())), 2);

//Availability Check

if (Convert.ToInt32(Availability) == 0)

{

MessageBox.Show("No Places available for '" + EventName + "' @" + row.Cells[1].Value.ToString() + "'", "Transaction Aborted!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

defaultbookingCount = 1;

dataTable.Rows.Add(EventName, row.Cells[1].Value, Availability, "€ " + CostPerBooking, defaultbookingCount, "€ " + CostPerBooking);

}

EventName = string.Empty;

}

}

//Block Grid Cells other than 'Quantity' from Editing

BookingInfoGridView.DataSource = dataTable;

ColumnCount = BookingInfoGridHeader.Split('|').Count();

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

{

if (i != 4)

{

BookingInfoGridView.Columns[i].ReadOnly = true;

}

}

BookingInfoGridView.ClearSelection();

BookingInfoGridView.Rows[0].Cells[4].Selected = true;

}

catch (Exception)

{

MessageBox.Show("Error Loading Booking Info", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#region DataGrid Error Styles

//Highlights Error Cells in Red

public void ApplyErrorStyles(DataGridViewCell cell)

{

DataGridViewCellStyle ErrorCellStyle = new System.Windows.Forms.DataGridViewCellStyle();

ErrorCellStyle.BackColor = System.Drawing.Color.OrangeRed;

cell.Style = ErrorCellStyle;

}

// Clears Error Styles

public void ClearErrorStyles(DataGridViewCell cell)

{

DataGridViewCellStyle ErrorCellStyle = new System.Windows.Forms.DataGridViewCellStyle();

ErrorCellStyle.BackColor = System.Drawing.Color.White;

cell.Style = ErrorCellStyle;

}

#endregion

#region Total Booking Fee Calculation

public void ComputeTotalBookingFee(DataGridViewCell cell)

{

int BookingCount = 0;

decimal BookingFee = 0m;

try

{

BookingCount = Convert.ToInt32(cell.EditedFormattedValue);

BookingFee = Convert.ToDecimal(BookingInfoGridView.Rows[cell.RowIndex].Cells[3].Value.ToString().Replace("€", string.Empty).Trim());

// Booking Fee= Quanity\* Fee per booking;

BookingInfoGridView.Rows[cell.RowIndex].Cells[5].Value = "€ " + Math.Round(BookingCount \* BookingFee, 2).ToString();

}

catch (Exception)

{

MessageBox.Show("Error Computing Total Booking Fee", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#region Confirm Order

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

{

// Prompts user with Order details

string \_data = string.Empty;

try

{

if (Summary.Summary.Count == 0)

{

MessageBox.Show("No Transactions Selected!", "No Transactions Found!", MessageBoxButtons.OK, MessageBoxIcon.Error);

RunningTotalTextBox.Text = "-";

}

else

{

foreach (BookingInfo booking in Summary.Summary)

{

\_data = \_data + "TransactionID : " + booking.TransactionID + " | TransactionDate : " + booking.TransactionDate + "\n" + "Event Name: " + booking.EventName + "\n" + "Location: " + booking.LocationName + "\n" + "MealPlan: " + booking.MealPlan + "\n" + "Quantity: " + booking.Quantity + "\n" + "Total Booking Cost: " + booking.OrderPrice + "\n\n";

}

MessageBox.Show(\_data, "Order Confimed!", MessageBoxButtons.OK, MessageBoxIcon.Information);

//Computes running total of all orders

RunningTotalTextBox.Text = "€ " + (Summary.Summary.Select(x => x.OrderPrice).Sum()).ToString("f2");

}

}

catch (Exception ex)

{

MessageBox.Show("Error in Completing Order!, Error:" + ex.Message, "Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#region Quantity Cell Validation

private void BookingInfoGridView\_CellLeave(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 4)

{

DataGridViewCell cell = BookingInfoGridView.Rows[e.RowIndex].Cells[e.ColumnIndex];

AddToBookingButton.Enabled = false;

if (ValidateBookingInfo(cell))

{

//Disables 'Add to Booking' on Validation Errors

AddToBookingButton.Enabled = true;

}

}

}

#region Quanity Validation

public bool ValidateBookingInfo(DataGridViewCell cell)

{

int tempBookingCount = 0;

int AvailableCount = 0;

try

{

tempBookingCount = int.Parse(cell.EditedFormattedValue.ToString());

AvailableCount = Convert.ToInt32(BookingInfoGridView.Rows[cell.RowIndex].Cells[2].Value.ToString());

//If Availability is 0

if (AvailableCount == 0)

{

MessageBox.Show("No Places Available for the Selected Event and Location!", "Order Can't be placed", MessageBoxButtons.OK, MessageBoxIcon.Error);

InitializeBookingInfoGrid();

EventInfoGridView.ClearSelection();

LocationInfoGridView.ClearSelection();

MealPlanGridView.ClearSelection();

return false;

}

//If Quanity is negative or zero

if (tempBookingCount <= 0 && AvailableCount > 0)

{

MessageBox.Show("Invalid Quantity", "Input Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

ApplyErrorStyles(cell);

return false;

}

//If Quantity exceeds availability

if (tempBookingCount > AvailableCount)

{

MessageBox.Show("Value exceeds Availability. Availability: " + AvailableCount.ToString() + " | Please refer 'Availabilty' near the location", "Input Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

cell.Value = AvailableCount.ToString();

BookingInfoGridView.RefreshEdit();

ClearErrorStyles(cell);

return true;

}

}

catch (Exception)

{

MessageBox.Show("Enter Numeric Data for No.of.Bookings", "Input Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

ApplyErrorStyles(cell);

return false;

}

ClearErrorStyles(cell);

//Computes total booking fee

ComputeTotalBookingFee(cell);

return true;

}

#endregion

#endregion

#region Add to Booking

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

{

try

{// Adds booking Info to Order Summary

BookingInfo booking = new BookingInfo();

if (InputValidation())

{

foreach (DataGridViewRow row in BookingInfoGridView.Rows)

{

booking.TransactionID = Convert.ToInt64(DateTime.Now.ToString().Replace("-", string.Empty).Replace(":", string.Empty).Replace(" ", string.Empty).Trim());

booking.TransactionDate = DateTime.Now;

booking.EventName = row.Cells[0].Value.ToString();

booking.LocationName = row.Cells[1].Value.ToString();

booking.MealPlan = MealPlanGridView.SelectedRows[0].Cells[1].Value.ToString();

booking.Quantity = Convert.ToInt32(row.Cells[4].Value);

booking.OrderPrice = Convert.ToDecimal(row.Cells[5].Value.ToString().Replace("€", string.Empty));

DialogResult dialogResult = MessageBox.Show("Confirm your Booking for '" + booking.EventName + "' @ '" + booking.LocationName + "' with Mealplan - '" + booking.MealPlan + "'" + "\n" + "Quantity : " + booking.Quantity + "\n" + "Total Booking Cost : " + booking.OrderPrice, "Booking Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

if (dialogResult == DialogResult.Yes)

{

// Updates Availability of Places

MetaData.StockData.Where(x => x.EventName == booking.EventName).Select(x => x).FirstOrDefault().AvailabilityInfo.Where(y => y.Location == booking.LocationName).Select(y => y).FirstOrDefault().Availability -= booking.Quantity;

Summary.Summary.Add(booking);

EventInfoGridView.ClearSelection();

LocationInfoGridView.ClearSelection();

MealPlanGridView.ClearSelection();

InitializeBookingInfoGrid();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Error Adding to Booking. Error:" + ex.Message, "Transaction Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#region Reset

//Sets the form to its pristine state

//Clears any pending previous bookings (if click before 'Save & Exit')

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

{

DialogResult \_result = MessageBox.Show("Unsaved Transactions will be aborted. Do you want to proceed?", "Confirm Reset", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (\_result == DialogResult.Yes)

{

Reset();

}

}

public void Reset()

{

Summary.Summary.Clear();

//Restores Availability of Non-saved Transactions

GetMetaData();

EventInfoGridView.ClearSelection();

LocationInfoGridView.ClearSelection();

MealPlanGridView.ClearSelection();

InitializeBookingInfoGrid();

RunningTotalTextBox.Text = string.Empty;

}

#endregion

#region Save and Exit

//Saves Transactional Data and Stock Data

//Closes the Application

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

{

WriteTransactionDataToFile();

WriteStockDataToFile();

Close();

}

#region WriteTransactionDataToFile

public void WriteTransactionDataToFile()

{

// Saves Order's Transactional Data in a Text file

string FileName = string.Empty;

string Data = string.Empty;

try

{

//Unique File Name for Every Transactional Day - Ex: TransactionData\_2019121.txt

FileName = ConfigurationManager.AppSettings.Get("TransactionFileName").ToString();

FileName = FileName.Replace("{Date}", DateTime.Now.Year.ToString() + DateTime.Now.Month.ToString() + DateTime.Now.Day.ToString());

StreamWriter streamWriter;

if (!File.Exists(FileName))

{

var \_file = File.CreateText(FileName);

\_file.Dispose();

}

streamWriter = File.AppendText(FileName);

// Transation is written onto the file

foreach (BookingInfo booking in Summary.Summary)

{

Data = "| Transaction ID : " + booking.TransactionID.ToString() + "\n" + "| Transaction Date : " + booking.TransactionDate.ToLongTimeString() + "\n|\n" + "| Event Name : " + booking.EventName + "\n" + "| Location : " + booking.LocationName + "\n" + "| Meal Plan : " + booking.MealPlan + "\n|\n" + "| Quantity : " + booking.Quantity + "\n" + "| Total Price : " + "€ " + booking.OrderPrice.ToString();

streamWriter.WriteLine("|--------------------------------------------|");

streamWriter.WriteLine(Data);

streamWriter.WriteLine("|--------------------------------------------|\n\n");

}

streamWriter.Close();

MessageBox.Show("Order Details Saved!", "Transaction Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception)

{

MessageBox.Show("Error Saving Transactions!", "Transaction Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#region WriteStockDataToFile

public void WriteStockDataToFile()

{

string StockFileName = string.Empty;

string Data = string.Empty;

try

{

// Writes Closing Stock Data on to StockData.txt file

StockFileName = ConfigurationManager.AppSettings.Get("StockFileName").ToString();

StreamWriter streamWriter;

File.Delete(StockFileName);

if (!File.Exists(StockFileName))

{

var \_file = File.CreateText(StockFileName);

\_file.Dispose();

}

streamWriter = File.AppendText(StockFileName);

// Stock Data is written in the format of a JSON

// This JSON string is fetched and Mapped to MetaData.StockData instance on form load

Data = JsonConvert.SerializeObject(MetaData.StockData).ToString();

streamWriter.WriteLine(Data);

streamWriter.Close();

MessageBox.Show("Stock Data Updated!", "Transaction Successful", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("Error updating Stock Data!, Error: " + ex.Message, "Transaction Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#endregion

#endregion

#region Generate Management Report

//Generates Management Report

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

{

string ManagerReportFileName = string.Empty; string ManagementReportTemplate = string.Empty;

int EventID = 0, Availability = 0; string \_data = string.Empty; string \_replaceData = string.Empty;

try

{

DialogResult \_result = DialogResult.Yes;

if (Summary.Summary.Count > 0)

{

\_result = MessageBox.Show("There are pending Orders to be Saved to file! Generating Management Report will abort pending Transactions. Do you want to proceed?", "Pending Orders!", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

}

if (\_result == DialogResult.Yes)

{

//Pending Transactions are cleared

//Restores Closing Stock Data of pending Transactions- For Generating Management Report

Reset();

ManagerReportFileName = ConfigurationManager.AppSettings.Get("ManagerReportFileName").ToString();

//Creates a new instance of Management Report each time

File.Delete(ManagerReportFileName);

if (!File.Exists(ManagerReportFileName))

{

var \_file = File.CreateText(ManagerReportFileName);

\_file.Dispose();

}

StreamWriter StreamWriter;

StreamWriter = File.AppendText(ManagerReportFileName);

//Fetches ManagementReport Template

ManagementReportTemplate = ConfigurationManager.AppSettings.Get("ManagementReportTemplate").ToString();

string[] TemplateData = File.ReadAllLines(ManagementReportTemplate);

string[] ReportData = new string[TemplateData.Length];

for (int index = 0; index < TemplateData.Length; index++)

{

if (index == 2)

{

// Updates 'Last Updated On' in the report

ReportData[index] = TemplateData[index].ToString().Replace("{Date}", DateTime.Now.ToString());

}

else if (index == 7 || index == 9 || index == 11 || index == 13 || index == 15 || index == 17 || index == 19 || index == 21 || index == 23 || index == 25)

{

ReportData[index] = TemplateData[index];

// EvventID is assigned based on the index position of the Event in the ManagementTemplate.txt

switch (index)

{

case 7: EventID = 1; break;

case 9: EventID = 2; break;

case 11: EventID = 3; break;

case 13: EventID = 4; break;

case 15: EventID = 5; break;

case 17: EventID = 6; break;

case 19: EventID = 7; break;

case 21: EventID = 8; break;

case 23: EventID = 9; break;

case 25: EventID = 10; break;

}

// Availability is added

for (int i = 1; i <= MetaData.EventLocationData.Count; i++)

{

Availability = MetaData.StockData.Where(x => x.EventID == EventID).Select(x => x).FirstOrDefault().AvailabilityInfo.Where(y => y.LocationID == i).Select(y => y.Availability).FirstOrDefault();

// To maintain alignmnet - a space is added if the Availability is a single digit number

if (Availability < 10)

{

\_data = " " + Availability.ToString();

}

else

{

\_data = Availability.ToString();

}

//Values '01','02','03','04' and '05' in the template are replaced with Availability of Cork,Dublin,Galway,Belmullet and Belfast

\_replaceData = "0" + i;

ReportData[index] = ReportData[index].ToString().Replace(\_replaceData, \_data);

}

}

else

{

ReportData[index] = TemplateData[index];

}

}

//The Updated Template is written on the file - ManagementReport.txt

foreach (string data in ReportData)

{

StreamWriter.WriteLine(data);

}

StreamWriter.Close();

MessageBox.Show("Management Report Generated! Please Check the file - ManagementReport.txt", "Report Generated!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception ex)

{

MessageBox.Show("Error Generation Management Report! Error :" + ex.Message, "Error Generating Report!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

#endregion

}

#region Modals

// Event Modal

public class Event

{

public int EventID { get; set; }

public string EventName { get; set; }

public int Days { get; set; }

public decimal RegistrationFee { get; set; }

}

//Location Modal

public class EventLocation

{

public int LocationID { get; set; }

public string Location { get; set; }

public decimal LodgingFee { get; set; }

public decimal TotalLodgingFee { get; set; }

}

//MealPlan Model

public class MealInfo

{

public int EventID { get; set; }

public decimal Full { get; set; }

public decimal Half { get; set; }

public decimal Breakfast { get; set; }

public decimal NoMeal { get; set; }

}

#region Stock Info

// Modal for holding available stiock data

public class AvailabilityInfo

{

public int LocationID { get; set; }

public string Location { get; set; }

public int Availability { get; set; }

}

public class StockInfo

{

public int EventID { get; set; }

public string EventName { get; set; }

public IList<AvailabilityInfo> AvailabilityInfo = new List<AvailabilityInfo>();

}

#endregion

#region Pricing Info

// Holds Pricing info of Events

public class LocationInfo

{

public int LocationID { get; set; }

public string Location { get; set; }

public decimal Price { get; set; }

}

public class PricingInfo

{

public int EventID { get; set; }

public string EventName { get; set; }

public IList<LocationInfo> LocationInfo = new List<LocationInfo>();

}

#endregion

public class MetaEvent

{

public IList<Event> EventData = new List<Event>();

public IList<EventLocation> EventLocationData = new List<EventLocation>();

public IList<MealInfo> MealData = new List<MealInfo>();

public IList<StockInfo> StockData = new List<StockInfo>();

public IList<PricingInfo> PrincingInfo = new List<PricingInfo>();

}

//Holds individual booking Info

public class BookingInfo

{

public long TransactionID { get; set; }

public DateTime TransactionDate { get; set; }

public string EventName { get; set; }

public string LocationName { get; set; }

public string MealPlan { get; set; }

public int Quantity { get; set; }

public decimal OrderPrice { get; set; } = 0.0m;

}

//Holds order info

public class BookSummary

{

public IList<BookingInfo> Summary = new List<BookingInfo>();

}

#endregion

}

**App.Config**

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

<appSettings>

<!--JSONS-->

<add key="EventInfo" value='[{"EventID":"1","EventName":"Murder Mystery Weekend","Days":"2","RegistrationFee":"600"},{"EventID":"2","EventName":"CSI Weekend","Days":"3","RegistrationFee":"1000"},{"EventID":"3","EventName":"The Great Outdoors","Days":"4","RegistrationFee":"1500"},{"EventID":"4","EventName":"The Chase","Days":"6","RegistrationFee":"1800"},{"EventID":"5","EventName":"Digital Refresh","Days":"2","RegistrationFee":"599"},{"EventID":"6","EventName":"Action Photography","Days":"5","RegistrationFee":"999"},{"EventID":"7","EventName":"Team Ryder Cup","Days":"3","RegistrationFee":"619"},{"EventID":"8","EventName":"Abseiling","Days":"2","RegistrationFee":"499"},{"EventID":"9","EventName":"War Games","Days":"6","RegistrationFee":"1999"},{"EventID":"10","EventName":"Find Walley","Days":"5","RegistrationFee":"799"}]'/>

<add key="LocationInfo" value='[{"LocationID":"1","Location":"Cork","LodgingFee":"250"},{"LocationID":"2","Location":"Dublin","LodgingFee":"165"},{"LocationID":"3","Location":"Galway","LodgingFee":"225"},{"LocationID":"4","Location":"Belmullet","LodgingFee":"305"},{"LocationID":"5","Location":"Belfast","LodgingFee":"95"}]'></add>

<add key='MealInfo' value='[{"EventID":"1","Full":"99","Half":"75","Breakfast":"24","NoMeal":"0"},{"EventID":"2","Full":"149","Half":"113","Breakfast":"36","NoMeal":"0"},{"EventID":"3","Full":"198","Half":"150","Breakfast":"48","NoMeal":"0"},{"EventID":"4","Full":"297","Half":"225","Breakfast":"72","NoMeal":"0"},{"EventID":"5","Full":"99","Half":"75","Breakfast":"24","NoMeal":"0"},{"EventID":"6","Full":"248","Half":"188","Breakfast":"60","NoMeal":"0"},{"EventID":"7","Full":"149","Half":"113","Breakfast":"36","NoMeal":"0"},{"EventID":"8","Full":"99","Half":"75","Breakfast":"24","NoMeal":"0"},{"EventID":"9","Full":"297","Half":"225","Breakfast":"72","NoMeal":"0"},{"EventID":"10","Full":"248","Half":"188","Breakfast":"60","NoMeal":"0"}]'/>

<add key='StockInfo' value='[{"EventID":"1","EventName":"Murder Mystery Weekend","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"35"},{"LocationID":"2","Location":"Dublin","Availability":"67"},{"LocationID":"3","Location":"Galway","Availability":"12"},{"LocationID":"4","Location":"Belmullet","Availability":"77"},{"LocationID":"5","Location":"Belfast","Availability":"32"}]},{"EventID":"2","EventName":"CSI Weekend","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"28"},{"LocationID":"2","Location":"Dublin","Availability":"3"},{"LocationID":"3","Location":"Galway","Availability":"34"},{"LocationID":"4","Location":"Belmullet","Availability":"12"},{"LocationID":"5","Location":"Belfast","Availability":"7"}]},{"EventID":"3","EventName":"The Great Outdoors","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"23"},{"LocationID":"2","Location":"Dublin","Availability":"2"},{"LocationID":"3","Location":"Galway","Availability":"6"},{"LocationID":"4","Location":"Belmullet","Availability":"4"},{"LocationID":"5","Location":"Belfast","Availability":"3"}]},{"EventID":"4","EventName":"The Chase","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"12"},{"LocationID":"2","Location":"Dublin","Availability":"6"},{"LocationID":"3","Location":"Galway","Availability":"9"},{"LocationID":"4","Location":"Belmullet","Availability":"4"},{"LocationID":"5","Location":"Belfast","Availability":"6"}]},{"EventID":"5","EventName":"Digital Refresh","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"2"},{"LocationID":"2","Location":"Dublin","Availability":"0"},{"LocationID":"3","Location":"Galway","Availability":"7"},{"LocationID":"4","Location":"Belmullet","Availability":"5"},{"LocationID":"5","Location":"Belfast","Availability":"8"}]},{"EventID":"6","EventName":"Action Photography","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"1"},{"LocationID":"2","Location":"Dublin","Availability":"8"},{"LocationID":"3","Location":"Galway","Availability":"4"},{"LocationID":"4","Location":"Belmullet","Availability":"7"},{"LocationID":"5","Location":"Belfast","Availability":"4"}]},{"EventID":"7","EventName":"Team Ryder Cup","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"16"},{"LocationID":"2","Location":"Dublin","Availability":"24"},{"LocationID":"3","Location":"Galway","Availability":"40"},{"LocationID":"4","Location":"Belmullet","Availability":"4"},{"LocationID":"5","Location":"Belfast","Availability":"12"}]},{"EventID":"8","EventName":"Abseiling","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"3"},{"LocationID":"2","Location":"Dublin","Availability":"7"},{"LocationID":"3","Location":"Galway","Availability":"45"},{"LocationID":"4","Location":"Belmullet","Availability":"3"},{"LocationID":"5","Location":"Belfast","Availability":"3"}]},{"EventID":"9","EventName":"War Games","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"45"},{"LocationID":"2","Location":"Dublin","Availability":"12"},{"LocationID":"3","Location":"Galway","Availability":"56"},{"LocationID":"4","Location":"Belmullet","Availability":"12"},{"LocationID":"5","Location":"Belfast","Availability":"23"}]},{"EventID":"10","EventName":"Find Walley","AvailabilityInfo":[{"LocationID":"1","Location":"Cork","Availability":"0"},{"LocationID":"2","Location":"Dublin","Availability":"0"},{"LocationID":"3","Location":"Galway","Availability":"3"},{"LocationID":"4","Location":"Belmullet","Availability":"0"},{"LocationID":"5","Location":"Belfast","Availability":"0"}]}]'/>

<add key='PricingInfo' value='[{"EventID":"1","EventName":"Murder Mystery Weekend","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"1100"},{"LocationID":"2","Location":"Dublin","Price":"930"},{"LocationID":"3","Location":"Galway","Price":"1050"},{"LocationID":"4","Location":"Belmullet","Price":"1210"},{"LocationID":"5","Location":"Belfast","Price":"790"}]},{"EventID":"2","EventName":"CSI Weekend","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"1750"},{"LocationID":"2","Location":"Dublin","Price":"1495"},{"LocationID":"3","Location":"Galway","Price":"1675"},{"LocationID":"4","Location":"Belmullet","Price":"1915"},{"LocationID":"5","Location":"Belfast","Price":"1285"}]},{"EventID":"3","EventName":"The Great Outdoor","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"2500"},{"LocationID":"2","Location":"Dublin","Price":"2160"},{"LocationID":"3","Location":"Galway","Price":"2400"},{"LocationID":"4","Location":"Belmullet","Price":"2720"},{"LocationID":"5","Location":"Belfast","Price":"1880"}]},{"EventID":"4","EventName":"The Chase","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"3300"},{"LocationID":"2","Location":"Dublin","Price":"2790"},{"LocationID":"3","Location":"Galway","Price":"3150"},{"LocationID":"4","Location":"Belmullet","Price":"3630"},{"LocationID":"5","Location":"Belfast","Price":"2730"}]},{"EventID":"5","EventName":"Digital Refresh","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"1099"},{"LocationID":"2","Location":"Dublin","Price":"929"},{"LocationID":"3","Location":"Galway","Price":"1049"},{"LocationID":"4","Location":"Belmullet","Price":"1209"},{"LocationID":"5","Location":"Belfast","Price":"789"}]},{"EventID":"6","EventName":"Action Photography","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"2249"},{"LocationID":"2","Location":"Dublin","Price":"1824"},{"LocationID":"3","Location":"Galway","Price":"2124"},{"LocationID":"4","Location":"Belmullet","Price":"2524"},{"LocationID":"5","Location":"Belfast","Price":"1474"}]},{"EventID":"7","EventName":"Team Ryder Cup","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"1369"},{"LocationID":"2","Location":"Dublin","Price":"1114"},{"LocationID":"3","Location":"Galway","Price":"1294"},{"LocationID":"4","Location":"Belmullet","Price":"1534"},{"LocationID":"5","Location":"Belfast","Price":"904"}]},{"EventID":"8","EventName":"Abseiling","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"999"},{"LocationID":"2","Location":"Dublin","Price":"829"},{"LocationID":"3","Location":"Galway","Price":"949"},{"LocationID":"4","Location":"Belmullet","Price":"1109"},{"LocationID":"5","Location":"Belfast","Price":"689"}]},{"EventID":"9","EventName":"War Games","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"3499"},{"LocationID":"2","Location":"Dublin","Price":"2989"},{"LocationID":"3","Location":"Galway","Price":"3349"},{"LocationID":"4","Location":"Belmullet","Price":"3829"},{"LocationID":"5","Location":"Belfast","Price":"2569"}]},{"EventID":"10","EventName":"Find Wally","LocationInfo":[{"LocationID":"1","Location":"Cork","Price":"2049"},{"LocationID":"2","Location":"Dublin","Price":"1624"},{"LocationID":"3","Location":"Galway","Price":"1924"},{"LocationID":"4","Location":"Belmullet","Price":"2324"},{"LocationID":"5","Location":"Belfast","Price":"1274"}]}]'/>

<!--Grid Headers-->

<add key='EventInfoGridHeader' value='ID|Event Name|Days|Registration Fee'/>

<add key='LocationInfoGridHeader' value='ID|Location|Lodging Fee Per Day|Total Lodging Fee'/>

<add key='MealInfoGridHeader' value='ID|Meal Type|Price'/>

<add key='BookingInfoGridHeader' value='Event Name|Location|Availability|Cost Per booking|No.of.Bookings|Total Booking Cost'/>

<add key='OrderSummaryGridHeader' value='Transaction Number|Transaction Date|Event Name|Location|Meal Plan|Quantity|Order Price'/>

<!--File Names-->

<add key='StockFileName' value='StockData.txt'/>

<add key='TransactionFileName' value='TransactionData\_{Date}.txt'/>

<add key='ManagerReportFileName' value='ManagementReport.txt'/>

<add key='ManagementReportTemplate' value='ManagementReportTemplate.txt'/>

</appSettings>

</configuration>