**Assignment 6 (Second Part)**

The second part of this assignment is to take the new code and finish it. Here is what needs to be done for the rest of the assignment. The first part of assignment 6 requirements still apply.

Once a flight is selected, all seats that are taken by passengers need to be turned red to indicate a taken seat. Next the user may choose to enter a new passenger and assign them a seat, select an existing passenger and change their seat, or delete a passenger from the flight.

To add a passenger the user will select the flight, and then click the “Add Passenger” button. This will bring up another form where the passenger’s information (First and Last Name) may be entered. If the user clicks the save button, the new passenger is inserted into the database. Once the information is saved, the passenger’s name will show up in the list of passengers combo box. From there the user must select a seat for the passenger (rest of the form is disabled) by clicking an empty seat on the diagram, which will update the seat with the “Selected Passengers” seat color (green).

To delete a passenger the user will select the flight, and then select the passenger. The passenger may be selected from the list of passengers or by clicking a taken seat. The selected passenger’s seat will show up in the Selected Passenger’s seat color (green). From there the “Delete Passenger” button may be pressed to delete the user from the list of passengers and turn the seat color back to an empty seat (blue).

To change a passenger’s seat the user will select the flight, and then select the passenger. The passenger may be selected from the list of passengers or by clicking a taken seat. The selected passenger’s seat will show up in the Selected Passenger’s seat color (green). From there the “Change Seat” button may be pressed (rest of the form is disabled), which will allow the user to click on an empty seat (blue) to assign it to the passenger. The passenger’s new seat will then show up in the Selected Passengers seat color (green).

On the plane seating diagram the color blue will mean that a seat is empty, the color red will mean that a seat is taken, and the color green will be the current selected passenger’s seat.

Please use help files provided for Assignment 6 that are contained in the “Assignment 6 Help” folder. There is an example of a completed Assignment 6 that may be ran to see how a full operational program should behave.

All business logic must be in separate classes and not behind the UI. All top level methods need to handle exceptions, and all other methods need to throw the exceptions. All methods and each attribute needs an XML comment.

For the flight and passenger combo box, they must be filled up with flights and passengers by binding a collection to the source of the combo box. This means that classes of type “FlightData” and “PassengerData” will have to be used. These classes will just hold the data for each, then added to a collection, like a List, then returned and bound to the combo box, to display the data.

**NOTE**: Microsoft Access should be used to store the data.

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Assignment6AirlineReservation

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

clsDataAccess clsData;

wndAddPassenger wndAddPass;

public MainWindow()

{

try

{

InitializeComponent();

Application.Current.ShutdownMode = ShutdownMode.OnMainWindowClose;

DataSet ds = new DataSet();

//Should probably not have SQL statements behind the UI

string sSQL = "SELECT Flight\_ID, Flight\_Number, Aircraft\_Type FROM FLIGHT";

int iRet = 0;

clsData = new clsDataAccess();

//This should probably be in a new class. Would be nice if this new class

//returned a list of Flight objects that was then bound to the combo box

//Also should show the flight number and aircraft type together

ds = clsData.ExecuteSQLStatement(sSQL, ref iRet);

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

{

cbChooseFlight.Items.Add(ds.Tables[0].Rows[i][0]);

}

}

catch (Exception ex)

{

HandleError(MethodInfo.GetCurrentMethod().DeclaringType.Name,

MethodInfo.GetCurrentMethod().Name, ex.Message);

}

}

private void cbChooseFlight\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

try

{

string selection = cbChooseFlight.SelectedItem.ToString();

cbChoosePassenger.IsEnabled = true;

gPassengerCommands.IsEnabled = true;

DataSet ds = new DataSet();

int iRet = 0;

if (selection == "1")

{

CanvasA380.Visibility = Visibility.Hidden;

Canvas767.Visibility = Visibility.Visible;

}

else

{

Canvas767.Visibility = Visibility.Hidden;

CanvasA380.Visibility = Visibility.Visible;

}

//I think this should be in a new class to hold SQL statments

string sSQL = "SELECT Passenger.Passenger\_ID, First\_Name, Last\_Name, FPL.Seat\_Number " +

"FROM Passenger, Flight\_Passenger\_Link FPL " +

"WHERE Passenger.Passenger\_ID = FPL.Passenger\_ID AND " +

"Flight\_ID = " + cbChooseFlight.SelectedItem.ToString();//If the cbChooseFlight was bound to a list of Flights, the selected object would have the flight ID

//Probably put in a new class

ds = clsData.ExecuteSQLStatement(sSQL, ref iRet);

cbChoosePassenger.Items.Clear();

//Would be nice if code from another class executed the SQL above, added each passenger into a Passenger object, then into a list of Passengers to be returned and bound to the combo box

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

{

cbChoosePassenger.Items.Add(ds.Tables[0].Rows[i][1] + " " + ds.Tables[0].Rows[i][2]);

}

}

catch (Exception ex)

{

HandleError(MethodInfo.GetCurrentMethod().DeclaringType.Name,

MethodInfo.GetCurrentMethod().Name, ex.Message);

}

}

private void cmdAddPassenger\_Click(object sender, RoutedEventArgs e)

{

try

{

wndAddPass = new wndAddPassenger();

wndAddPass.ShowDialog();

}

catch (Exception ex)

{

HandleError(MethodInfo.GetCurrentMethod().DeclaringType.Name,

MethodInfo.GetCurrentMethod().Name, ex.Message);

}

}

private void HandleError(string sClass, string sMethod, string sMessage)

{

try

{

MessageBox.Show(sClass + "." + sMethod + " -> " + sMessage);

}

catch (System.Exception ex)

{

System.IO.File.AppendAllText(@"C:\Error.txt", Environment.NewLine + "HandleError Exception: " + ex.Message);

}

}

}

}