**Class diagram**

**C:\Users\Andre Moazed\Downloads\Assessment 2 - 40216327 - Class Diagram.png**

**ER diagram**

**C:\Users\Andre Moazed\Downloads\Assessment 2 - 40216327 - ER diagram.pngs**

**Database\_facade:**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: A facade class that is used by all GUI classes to execute mathods

\* This class makes use of both the Cur\_customer\_Singelton and the Cur\_booking\_Singelton

\*/

using System;

using System.Collections.Generic;

using System.Linq;

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;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using MySql.Data.MySqlClient;

using System.Text.RegularExpressions;

namespace SD2CW2

{

public class DatabaseFacade //all methods within this class will be public in order for them to be called in other classes

{

/\*

\* The next line will be used to connect to the database that will be used in the program

\* It is a SQL connection string that connects to the localhost server (the server on this laptop)

\* It will automatically use the user id root, standard for when dealing with localhost servers

\* The password has been set to password in the server and so is automatically inputed by the SQL connection string

\* The database to be used is called 40216327 (my matriculation number) and the ER diagram for this database can be found in the included pdf

\*/

private static string SQLConnect = "server=localhost;user id=root;password=password;database=40216327;persistsecurityinfo=False";

private MySqlConnection con = new MySqlConnection(SQLConnect); //variable to create a new connection with the SQLconnect connection string

private MySqlCommand cmd; //a variavle that will be used for SQL commands

private MySqlDataReader sdr; //the data reader variable

private string sql; //A new string, sql, that will be used to contain sql queries or sql non-queries

public DatabaseFacade()

//constructor method

{

cmd = new MySqlCommand(sql, con); //constructor will create a new instance of an sql command that will take in a string for queries/non-queries and that will use the connection string.

}

public bool OpenCon()

//opening the database connection method

{

try //error handeling to ensure the connection is opened

{

con.Open(); //open the connection to the database

return true; //If the connection was sucessfully opened then set the boolean value to true

}

catch (MySqlException ex)

{

MessageBox.Show(ex.Message); //show the exception message in a message box

return false; //set the value to false

}

}

public bool CloseCon()

//closing the connection to the database method

{

try

{

con.Close(); //attempt to close the connection

return true; //if connection is closed set value to true

}

catch (MySqlException ex)

{

MessageBox.Show(ex.Message); //Show the exeption message is the connection fails to close

return false; //the boolean value is set to false to indicate an unsucesessful close

}

}

public MySqlCommand InitSqlCommand(string sql)

//Method that will allow the sql string to be used

{

cmd = new MySqlCommand(sql, con); //setting the cmd (command) to include the sql string and the connection to the database

return cmd;

}

public void End()

//Method to dispose of any database conenction as this cannot be done by the c# garabge collector

{

if (con != null && con.State == ConnectionState.Closed)

//If the connection has a value and the state is unclosed then do the following:

{

con.Dispose(); //dispose of the connection

}

}

public Customer\_class Select\_Cust(int cust\_ref)

{

//cust\_ref = Int32.Parse(txtBox\_select\_cust.Text); //sets the value of variable cust ref to what is in the select text box

bool custExists = false;

sql = "SELECT \* FROM customer WHERE cust\_ref = " + cust\_ref + ";"; //Sql query to set the value of the sql variable in the database\_facade

cmd.CommandText = sql; //set the cmd command text to the sql statement

sdr = cmd.ExecuteReader(); // set the data reader equal tot the execute reader method

Customer\_class cust = new Customer\_class();

while (sdr.Read())

{

if (sdr.GetString(0) == cust\_ref.ToString()) //if the value of the first column of the database table is equal to the customer reference

{

custExists = true;

cust.Ref = Int32.Parse(sdr.GetString(0)); //Get the value of the first column from the database table

cust.LastName = sdr.GetString(1); //Get the value of the second column from the database table

cust.Firstname = sdr.GetString(2);

cust.Address = sdr.GetString(3);

}

}

if (custExists == false) //if the customer does not exist

{

MessageBox.Show("Customer does not exist. Try a diffrent customer reference");

}

else

{

Cur\_customer\_Singleton cur\_cust = Cur\_customer\_Singleton.Instance(cust);

Customer cust\_Win = new Customer(); //create a new customer window

//cust\_Win.Owner = this;

cust\_Win.Show();

/\*

\* The next lines will set the value of the text boxes to the previously save values

\* This will make the text boxes fill in with their correct values

\*/

cust\_Win.txtBox\_cust\_ref.Text = cust.Ref.ToString();

cust\_Win.txtBox\_cust\_last.Text = cust.LastName;

cust\_Win.txtBox\_cust\_first.Text = cust.Firstname;

cust\_Win.txtBox\_cust\_address.Text = cust.Address;

}

sdr.Close(); //closes the data reader - prevents leaks

return cust;

}

public void Save\_cust(string lastname, string firstname, string address)

/\*This method will take in a non-query and insert values into the database table

\* It will take in the values from the text boxes in the customer window

\* This is used for updating a customer and therfore will take in customer lastname, firstname, address and the customer reference

\*/

{

/\*

\* the next line of code is used by the database and is a simple INSERT statement that will insert the values taken in by the method (the

\* values in the textboxes from the customer window) and place them into the sql non-query

\*/

sql = @"INSERT INTO customer VALUES(

cust\_ref,'" + lastname + "','" + firstname + "','" + address + "');";

cmd.CommandText = sql;

cmd.ExecuteNonQuery(); //once again this is a non-query and therefore mus be executed as such

}

public int Place\_cust\_ref()

{

int result = 0;

sql = "SELECT MAX(cust\_ref) FROM customer;"; //ovverides the sql cariable contents

cmd.CommandText = sql; //set command text equal to the sql variable

sdr = cmd.ExecuteReader();

while (sdr.Read())

{

result = Int32.Parse(sdr.GetString(0)); //set variable equal to the first column of the table

}

sdr.Close();

//creating a new instance of the customer singelton

Customer\_class cust = Select\_Cust(result);

Cur\_customer\_Singleton cust\_sin = Cur\_customer\_Singleton.Instance(cust);

return result;

}

public void Update\_cust(int cust\_ref, string lastname, string firstname, string address)

/\*

\* this method will be called by the customer window class and it will be used to update the MySql database given that the customer reference

\* is existing. The UPDATE allows for changing of data, the cust\_ref is used to determine which row will be changed

\* the row to be changed will be that of the row containing the customer reference number

\*/

{

sql = @"UPDATE customer SET "

+ "cust\_lastname='" + lastname + "',"

+ "cust\_firstname='" + firstname + "',"

+ "cust\_address='" + address + "'"

+ "WHERE cust\_ref=" + cust\_ref + ";";

cmd.CommandText = sql;

cmd.ExecuteNonQuery(); //Non-query being executed

}

public void Del\_cust(int cust\_ref)

/\*

\* Method to delete rows from a database entity (table), this delete method will remove a row from the

\* customer table and will remove the row with the corresponding customer reference (essentially deleting all records of the customer

\* form the system)

\*/

{

sql = "DELETE FROM customer WHERE cust\_ref='" + cust\_ref + "';";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

}

public Booking\_class Select\_booking(int booking\_ref, int cust\_ref)

//Method to select the booking, since each booking has a customer associated with it the customer reference must also be used

{

Cur\_customer\_Singleton cur\_cust = Cur\_customer\_Singleton.Instance();

sql = "SELECT \* FROM booking WHERE booking\_ref=" + booking\_ref + " AND cust\_ref=" + cust\_ref + ";"; //Query to find the correct booking given the correct customer reference

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

bool bookExists = false;

Booking\_class book = new Booking\_class();

while(sdr.Read()) //while the data reader is reading through the database

{

if(sdr.GetString(0) == booking\_ref.ToString())

/\*

\* This if statement will ensure that the booking reference in the database will be equal to the booking reference in the c#

\* The booking class will have the data from the database saved to the corresponding variable in the booking\_class

\* Ref = first column of the database (i.e the booking reference) etc.

\*/

{

bookExists = true;

book.Ref = Int32.Parse(sdr.GetString(0));

book.Arrivaldate = sdr.GetString(1);

book.DepDate = sdr.GetString(2);

}

}

sdr.Close(); //data reader is no longer in use, therfore close it

if(bookExists == false)

//Error checking ensuring that the booking reference exists.

{

MessageBox.Show("Booking does not exist given the selected reference number.");

}

else

{

Cur\_booking\_Singleton cur\_booking = Cur\_booking\_Singleton.Instance(book);

Booking bookingWin = new Booking(); //creating a new booking window

bookingWin.Show(); //open the booking window

bookingWin.txtBox\_booking\_ref.Text = book.Ref.ToString(); //set the text box value, booking ref, to the booking reference

bookingWin.datepicker\_arrival.Text = book.Arrivaldate; //The arrival date is already a string and therefore does not need to be converted into a string

bookingWin.datepicker\_dep.Text = book.DepDate;

bookingWin.btn\_invoice.IsEnabled = true;

bookingWin.btn\_extras.IsEnabled = true;

bookingWin.btn\_view\_guests.IsEnabled = true;

}

return book;

}

public void Save\_booking(string arrival\_date, string dep\_date, int cust\_ref)

{

/\*

\* sql non-query that inserts values of into the database, the insert will not need to take in a booking reference as this is auto incremented

\* in the database itself. The arrival date and departure dates need to be inserted as strings as the date time conversions between c# and MySql

\* do not work fully.

\*/

sql = @"INSERT INTO booking VALUES(

booking\_ref,'" + arrival\_date + "','" + dep\_date + "'," + cust\_ref + ");";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

}

public int Place\_booking\_ref()

/\*

\* This method will place the newly generated booking refernce into the booking reference text box

\*/

{

int result = 0; //since the result of the query needs to be returned and is going to be an int the result varible is set as an int

sql = "SELECT MAX(booking\_ref) FROM booking;"; //This selects the most recently created booking reference from the database

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

while (sdr.Read())

{

result = Int32.Parse(sdr.GetString(0));

}

sdr.Close();

return result; //returns the result

}

public void Update\_booking(string arrival\_date, string dep\_date, int booking\_ref)

/\*

\* Another update method, however this one is used for the booking class

\* It uses the same prinicples as all other update methods, taking in the set of values that need to be updated in the

\* databse and used an execute non-query command, where the unique identifier of the non-query will be the booking\_ref

\*/

{

sql = @"UPDATE booking SET " +

"arrival\_date='" + arrival\_date +

"',departure\_date='" + dep\_date +

"'WHERE booking\_ref=" + booking\_ref + ";";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

}

public void Del\_booking(int booking\_ref)

/\*

\* Removes the row of data from the booking table where the booking reference is equal to the booking\_ref

\*/

{

sql = "DELETE FROM booking WHERE booking\_ref=" + booking\_ref + ";";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

}

public List<string> Guest\_exist(int booking\_ref)

/\*

\* This method will be used for error checking/handling.

\* If the method returns

\*/

{

sql = "SELECT pass\_num FROM guest WHERE booking\_ref=" + booking\_ref + ";";

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

List<string> pass\_numbers = new List<string>();

while(sdr.Read())

{

pass\_numbers.Add(sdr.GetString(0));

}

sdr.Close();

return pass\_numbers;

}

public List<string> Extras\_exist(int booking\_ref)

/\*

\* Used to check if there are extras that exist in relation to a booking reference

\*/

{

sql = "SELECT extra\_ref FROM extras WHERE booking\_ref=" + booking\_ref + ";";

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

List<string> extra\_exists = new List<string>(); //creating a new list

while (sdr.Read())

{

extra\_exists.Add(sdr.GetString(0)); //adding the value of the first column into the extras list, to be used later witha count

}

sdr.Close();

return extra\_exists; //returns the extra list

}

public string Pass\_num\_exists(string pass\_num)

//same concept as the extras existing method, however uses strings ass opposed to ints as passport numbers are always strings

{

sql = "SELECT pass\_num FROM guest WHERE pass\_num='" + pass\_num + "';";

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

string found\_pass\_num = "";

while(sdr.Read())

{

found\_pass\_num = sdr.GetString(0);

}

sdr.Close();

if (found\_pass\_num == "")

{

pass\_num = "";

return pass\_num;

}

else

{

return pass\_num;

}

}

public void Load\_guests(int booking\_ref)

/\*

\* This view method will be slightly different as it will need to load multiple different text boxes from multiple different rows

\* in the database, within the same method. In order to achieve this a list will be used to iterate through all the four possible guests

\*/

{

Cur\_booking\_Singleton curbooking = Cur\_booking\_Singleton.Instance(); //create a new instance of booking, as only one should be used at one time

sql = "SELECT \* FROM guest WHERE booking\_ref=" + booking\_ref + ";"; //SELECT statement tha searches for the guests with the correct booking reference

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

List<guest> guestList = new List<guest>(); //creation of the list

while(sdr.Read())

{

if(sdr.GetString(5) == booking\_ref.ToString()) //if the value of the 5th column in the database table is equal to the booking ref then

/\*

\* Same as previous reads however this time the identifier is the foreign key booking\_reference as there may be multiple guests that need to be loaded in

\*/

{

guest guest = new guest();

guest.Pass\_num = sdr.GetString(0);

guest.LastName = sdr.GetString(1);

guest.Firstname = sdr.GetString(2);

guest.DoB = sdr.GetString(3);

guest.Dietreq = sdr.GetString(4);

guest.Booking\_ref = booking\_ref;

guestList.Add(guest); //adding all the values from the previous lines into the list, naming each group of values "guest"

}

}

sdr.Close();

Guests guestsWin = new Guests(); //creating a new guest window

guestsWin.Show();

guestsWin.txtBox\_booking\_ref.Text = booking\_ref.ToString(); //inserting the booking reference into the booking refernce text box in the guest window

for(int i = 0; i < guestList.Count(); i++)

/\*

\* This for loop is used to iterate through the possible data that can be loaded into the guest window

\* It will only be the size of the count of the guest list. Therefore the data wont be loaded in if the data for the guest does not exist.

\*/

{

if(i == 0)

//for the first item in the list set the follwoing text boxes to the values in the list

{

guestsWin.txtBox\_g1\_pass\_num.Text = guestList[i].Pass\_num; //i.e set the passs number text box to be equal to the passpot number of guest 1

guestsWin.txtBox\_g1\_last.Text = guestList[i].LastName;

guestsWin.txtBox\_g1\_first.Text = guestList[i].Firstname;

guestsWin.datepicker\_g1\_DoB.Text = guestList[i].DoB;

guestsWin.txtBox\_g1\_diet\_req.Text = guestList[i].Dietreq;

}

if(i == 1)

{

guestsWin.txtBox\_g2\_pass\_num.Text = guestList[i].Pass\_num;

guestsWin.txtBox\_g2\_last.Text = guestList[i].LastName;

guestsWin.txtBox\_g2\_first.Text = guestList[i].Firstname;

guestsWin.datepicker\_g2\_DoB.Text = guestList[i].DoB;

guestsWin.txtBox\_g2\_diet\_req.Text = guestList[i].Dietreq;

}

if(i == 2)

{

guestsWin.txtBox\_g3\_pass\_num.Text = guestList[i].Pass\_num;

guestsWin.txtBox\_g3\_last.Text = guestList[i].LastName;

guestsWin.txtBox\_g3\_first.Text = guestList[i].Firstname;

guestsWin.datepicker\_g3\_DoB.Text = guestList[i].DoB;

guestsWin.txtBox\_g3\_diet\_req.Text = guestList[i].Dietreq;

}

if(i == 3)

{

guestsWin.txtBox\_g4\_pass\_num.Text = guestList[i].Pass\_num;

guestsWin.txtBox\_g4\_last.Text = guestList[i].LastName;

guestsWin.txtBox\_g4\_first.Text = guestList[i].Firstname;

guestsWin.datepicker\_g4\_DoB.Text = guestList[i].DoB;

guestsWin.txtBox\_g4\_diet\_req.Text = guestList[i].Dietreq;

}

}

}

public void Save\_guest(string pass\_num, string lastname, string firstname, string DoB, string diet\_req, int booking\_ref)

/\*

\* Guest save method that will use INSERT to input new values into the guest table in the MySql database

\*/

{

sql = @"INSERT INTO guest VALUES('" + pass\_num + "','"

+ lastname + "','"

+ firstname + "','"

+ DoB + "','"

+ diet\_req + "',"

+ booking\_ref + ");";

cmd.CommandText = sql;

cmd.ExecuteNonQuery(); //Execute a non-query, as INSERT isn't a query this is used rather than ExecuteReader()

}

public void Update\_guest(string pass\_num, string lastname, string firstname, string DoB, string diet\_req, int booking\_ref)

/\*

\* Update method that will take in strings and integers to be inputed into the database in order to update the database

\* This is done via a UPDATE statemtment. The statement uses variables as this prevent both sql injection from occuring and also cleans up

\* the code in the program, allowing this method to be easily used in any class

\*/

{

sql = @"UPDATE guest SET " +

"pass\_num='" + pass\_num +

"',guest\_lastname='" + lastname +

"',guest\_firstname='" + firstname +

"',guest\_DoB='" + DoB +

"',guest\_diet\_req='" + diet\_req +

"' WHERE pass\_num='" + pass\_num + "' AND booking\_ref=" + booking\_ref +";";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

}

public void Del\_guest(string pass\_num)

//delete method that searches the databse for the guest with the corresponding passport number

{

if (pass\_num != "")

{

sql = "DELETE FROM guest WHERE pass\_num='" + pass\_num + "';";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

}

else

{

MessageBox.Show("This customer does not exist or hasn't been saved and therefore cannot be deleted");

}

}

public extras Load\_extras(int booking\_ref)

//once again loading in the extras via the booking class

{

sql = "SELECT \* FROM extras WHERE booking\_ref='" + booking\_ref + "';";

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

extras extra = new extras();

while(sdr.Read())

{

if(booking\_ref == Int32.Parse(sdr.GetString(9)))

{

extra.Ref = Int32.Parse(sdr.GetString(0));

extra.Breakfast\_price = Double.Parse(sdr.GetString(1));

extra.Breakfast\_num = Int32.Parse(sdr.GetString(2));

extra.Evening\_meal\_price = Double.Parse(sdr.GetString(3));

extra.Evening\_meal\_num = Int32.Parse(sdr.GetString(4));

extra.Car\_hire\_price = Double.Parse(sdr.GetString(5));

extra.Car\_hire\_start = sdr.GetString(6);

extra.Car\_hire\_end = sdr.GetString(7);

extra.Car\_hire\_name = sdr.GetString(8);

extra.BookingRef = Int32.Parse(sdr.GetString(9));

}

}

sdr.Close();

Extras extraWin = new Extras();

extraWin.Show();

extraWin.txtBox\_extras\_ref.Text = extra.Ref.ToString();

extraWin.txtBox\_booking\_ref.Text = booking\_ref.ToString();

extraWin.txtBox\_breakfast.Text = extra.Breakfast\_num.ToString();

extraWin.txtBox\_evening\_meal.Text = extra.Evening\_meal\_num.ToString();

extraWin.datepicker\_start.Text = extra.Car\_hire\_start;

extraWin.datepicker\_end.Text = extra.Car\_hire\_end;

extraWin.txtBox\_car\_hire\_name.Text = extra.Car\_hire\_name;

//the next lines will change the value in the datepicker to be clear rather than the databse default of "01/01/0001"

if(extraWin.datepicker\_start.Text == "01/01/0001")

{

extraWin.datepicker\_start.Text = "";

}

if(extraWin.datepicker\_end.Text == "01/01/0001")

{

extraWin.datepicker\_end.Text = "";

}

return extra;

}

public void Save\_extras(int booking\_ref, int breakfasts, int eveningmeals, string carhire\_start, string carhire\_end, string carhire\_name)

{

sql = @"INSERT INTO extras VALUES(extra\_ref,5.00," + breakfasts + ",15.00,"

+ eveningmeals + ",50.00,'" + carhire\_start + "','" + carhire\_end + "','"

+ carhire\_name + "'," + booking\_ref + ");";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

}

public int Place\_extras\_ref()

{

int result = 0;

sql = "SELECT MAX(extra\_ref) FROM extras;";

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

while (sdr.Read())

{

result = Int32.Parse(sdr.GetString(0));

}

sdr.Close();

return result;

}

public void Update\_extras(int booking\_ref, int breakfasts, int eveningmeals, string carhire\_start, string carhire\_end, string carhire\_name)

{

sql = @"UPDATE extras SET "

+ "breakfast\_num=" + breakfasts + ",evening\_meal\_num=" + eveningmeals

+ ",car\_hire\_start='" + carhire\_start + "',car\_hire\_end='" + carhire\_end + "'"

+ ",car\_hire\_name='" + carhire\_name + "' WHERE booking\_ref=" + booking\_ref + ";";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

}

public void Del\_extras(int extra\_ref)

{

if (extra\_ref.ToString() != "")

{

sql = "DELETE FROM extras WHERE extra\_ref='" + extra\_ref + "';";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

}

else

{

MessageBox.Show("This extra does not exist or hasn't been saved and therefore cannot be deleted");

}

}

public double Extras\_invoice(int booking\_ref)

/\*

\* Method with a query that will return the total of all extras that refer to a certain booking

\* The calculation is done via sql an statement that will take in the booking reference as its variable

\* this allows for faster calcualtion than doing it in the c# program, this is because the program will not need

\* to continously fetch information from the database

\*/

{

string result = "";

sql = "SELECT SUM(((COALESCE(breakfast\_num,0))\*5.00)+"

+ "((COALESCE(evening\_meal\_num,0))\*15.00)+"

+ "((COALESCE(DATEDIFF(car\_hire\_end,car\_hire\_start),0)))\*50.00)"

+ " FROM extras WHERE booking\_ref=" + booking\_ref + ";";

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

while(sdr.Read())

{

result = sdr.GetString(0);

}

sdr.Close();

return Convert.ToDouble(result);

}

public double Rooms\_invoice(int booking\_ref)

/\*

\* Method with a query that will return the total of the price for each guest multiplied by the cost for that specific guest

\* This price refers to a certain booking and the age of the guest

\* The calculation is done via sql an statement that will take in the booking reference as its variable

\* this allows for faster calcualtion than doing it in the c# program, this is because the program will not need

\* to continously fetch information from the database

\*/

{

string result = "";

sql = "SELECT SUM(CASE WHEN(YEAR(CURDATE())-YEAR(guest\_DoB)"

+ "-(DATE\_FORMAT(CURDATE(),'%m%d'<DATE\_FORMAT(guest\_DoB,'%m%d'))"

+ ">= 18)) THEN (50\*(DATEDIFF(departure\_date, arrival\_date)))"

+ " ELSE (30\*(DATEDIFF(departure\_date, arrival\_date))) END)"

+ " FROM booking JOIN guest ON booking.booking\_ref=guest.booking\_ref"

+ " WHERE booking.booking\_ref=" + booking\_ref + ";";

cmd.CommandText = sql;

sdr = cmd.ExecuteReader();

while(sdr.Read())

{

result = sdr.GetString(0);

}

sdr.Close();

return Convert.ToDouble(result);

}

}

}

**Cur\_customer\_Singelton**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: used to only allow one value of the current customer to be stored

\* used by the databse facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SD2CW2

{

public class Cur\_customer\_Singleton

//This class will be used to reduce the number of customers that can be created on the heap

//Since only one customer can be viewd and changed at one time there is no need to allow there

//to exist more than one value of customer on the memory heap

{

private static Cur\_customer\_Singleton instance; //create an instance of the class

Customer\_class cust; //calling the customer class and setting it to cust

public static Cur\_customer\_Singleton Instance()

//Used for the creation of an instance of this class

{

if (instance == null)

//the following if statemnts will check if the instance already exist

//if the instance doesn't exist then a new instance is created

{

instance = new Cur\_customer\_Singleton();

return instance;

}

else

//otehrwise the current existing instance will be returned

{

return instance;

}

}

public static Cur\_customer\_Singleton Instance(Customer\_class cust)

//Instance of customer where the customer data exists

//a new instance of customer will be made, calling the previous method

//if the customer instance is already non-existant

{

if (instance == null)

{

instance = new Cur\_customer\_Singleton();

instance.cust = cust;

return instance;

}

else

//otherwise the existing instance of customer will be called

{

instance.cust = cust;

return instance;

}

}

public int GetRefNum()

//method to return the reference number of the customer, only one is required

//at any given time, therefore it can exist in this singleton pattern class

{

return cust.Ref;

}

}

}

**Cur\_booking\_Singelton**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: used to only allow one value of the current booking to be stored

\* used by the databse facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SD2CW2

{

public class Cur\_booking\_Singleton

//This class will be used to reduce the number of bookings that can be created on the heap

//Since only one booking can be viewd and changed at one time there is no need to allow there

//to exist more than one value of booking on the memory heap

{

private static Cur\_booking\_Singleton instance;

Booking\_class book; //calling the booking class and setting it to book

public string Bookingref { get; set; }

public static Cur\_booking\_Singleton Instance()

{

if (instance == null)

{

instance = new Cur\_booking\_Singleton(); //creating a new instance of the class if there isn't one already

return instance;

}

else

{

return instance; //else the instance will be used, only one location in memory used

}

}

public static Cur\_booking\_Singleton Instance(Booking\_class book)

{

if (instance == null)

{

instance = new Cur\_booking\_Singleton(); //using an instance of the booking class - this is where the variables were declared

instance.book = book;

return instance;

}

else

{

instance.book = book;

return instance;

}

}

public int GetRefNum()

{

return book.Ref;

}

}

}

**Booking\_class**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: Booking class that is used by the database facade and singelton in order to store values

\* used by the booking singelton class and database facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SD2CW2

{

public class Booking\_class

{

private int reference;

private string arrivaldate;

private string depedate;

private int num\_of\_guests;

private int custref;

public int Ref

{

get { return reference; }

set { reference = value; }

}

public string Arrivaldate

{

get { return arrivaldate; }

set { arrivaldate = value; }

}

public string DepDate

{

get { return depedate; }

set { depedate = value; }

}

public int Num\_of\_guests

{

get { return num\_of\_guests; }

set { num\_of\_guests = value; }

}

public int CustRef

{

get { return custref; }

set { custref = value; }

}

}

}

**Customer\_class**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: customer class that is used by the database facade and singelton in order to store values

\* used by the customer singelton class and database facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SD2CW2

{

public class Customer\_class

{

private string lastname;

private string firstname;

private int reference;

private string address;

public string LastName

{

get { return lastname; }

set { lastname = value; }

}

public string Firstname

{

get { return firstname; }

set { firstname = value; }

}

public int Ref

{

get { return reference; }

set { reference = value; }

}

public string Address

{

get { return address; }

set { address = value; }

}

}

}

**Extras**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: class used to declare variables

\* used by the database facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SD2CW2

{

public class extras

{

public int Ref { get; set; }

public double Breakfast\_price { get; set; }

public int Breakfast\_num { get; set; }

public double Evening\_meal\_price { get; set; }

public int Evening\_meal\_num { get; set; }

public double Car\_hire\_price { get; set; }

public string Car\_hire\_start { get; set; }

public string Car\_hire\_end { get; set; }

public string Car\_hire\_name { get; set; }

public int BookingRef { get; set; }

}

}

**Guest**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: class used to declare variables

\* used by the database facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SD2CW2

{

class guest

{

private string lastname;

private string firstname;

private string pass\_num;

private string dob;

private string dietreq;

private int booking\_ref;

public string LastName

{

get { return lastname; }

set { lastname = value; }

}

public string Firstname

{

get { return firstname; }

set { firstname = value; }

}

public string Pass\_num

{

get { return pass\_num; }

set { pass\_num= value; }

}

public string DoB

{

get { return dob; }

set { dob = value; }

}

public string Dietreq

{

get { return dietreq; }

set { dietreq = value; }

}

public int Booking\_ref

{

get { return booking\_ref; }

set { booking\_ref = value; }

}

}

}

**GUI classes:**

**Main Window:**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: GUI class to create new customers or select existing customers

\* Uses the database facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

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;

using MySql.Data.MySqlClient;

using System.Text.RegularExpressions;

namespace SD2CW2

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

DatabaseFacade dbcon = new DatabaseFacade(); //new instance of database facade so that its methods can be used within Main

public MainWindow()

{

InitializeComponent();

dbcon.OpenCon(); //call open connection method from database facade class

}

private void btn\_select\_cust\_Click(object sender, RoutedEventArgs e)

//Button to select a customer and open a new customer window

{

if (txtBox\_select\_cust.Text == "")

{

MessageBox.Show("Please enter a customer reference");

}

else

{

dbcon.Select\_Cust(Int32.Parse(txtBox\_select\_cust.Text));

}

}

private void btn\_new\_cust\_Click(object sender, RoutedEventArgs e)

{

new Customer().Show(); //open a new customer window

}

private void IntegerValidationInput(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

}

}

**Customer**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: GUI class that will link to the booking window, uses the database facade to store information

\* Makes use of the database facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

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.Shapes;

using System.Text.RegularExpressions;

namespace SD2CW2

{

/// <summary>

/// Interaction logic for Customer.xaml

/// </summary>

public partial class Customer : Window

{

DatabaseFacade dbcon = new DatabaseFacade();

public Customer()

{

InitializeComponent();

dbcon.OpenCon();

}

private void btn\_cust\_save\_Click(object sender, RoutedEventArgs e)

{

if(txtBox\_cust\_first.Text == "") //validation to ensure names and addresses are inputed before saving

{

MessageBox.Show("Please insert a first name");

}

else if(txtBox\_cust\_last.Text == "")

{

MessageBox.Show("Please insert a last name");

}

else if(txtBox\_cust\_address.Text == "")

{

MessageBox.Show("Please insert an address");

}

else

{

if (txtBox\_cust\_ref.Text == "") //for new customers it will use the save method from the database facade

{

dbcon.Save\_cust(txtBox\_cust\_last.Text, txtBox\_cust\_first.Text, txtBox\_cust\_address.Text);

txtBox\_cust\_ref.Text = dbcon.Place\_cust\_ref().ToString();

this.Close();

}

if(txtBox\_cust\_ref.Text != "") //for existing customers an update will be called

{

dbcon.Update\_cust(Int32.Parse(txtBox\_cust\_ref.Text), txtBox\_cust\_last.Text, txtBox\_cust\_first.Text, txtBox\_cust\_address.Text);

}

}

}

private void btn\_cust\_del\_Click(object sender, RoutedEventArgs e)

//A delete that will remove the data from the MySql database

{

if (txtBox\_cust\_ref.Text == "")

{

MessageBox.Show("Customer must exist before it can be deleted");

}

else

{

dbcon.Del\_cust(Int32.Parse(txtBox\_cust\_ref.Text));

}

this.Close();

MessageBox.Show("The customer has been removed.");

}

private void btn\_new\_booking\_Click(object sender, RoutedEventArgs e)

//simple opening of a new GUI window for the booking

{

if(txtBox\_cust\_ref.Text != "") //If there exists a value in the cust\_ref text box

{

new Booking().Show(); //open the booking window

}

else

{

MessageBox.Show("The customer must be saved before a new booking can be created.");

}

}

private void btn\_select\_booking\_Click(object sender, RoutedEventArgs e)

//opens the booking window inserting the correct data that is retrieved from the database via the database facade

{

if (txtBox\_select\_booking.Text == "")

{

MessageBox.Show("Please insert a booking reference");

}

else if(txtBox\_cust\_ref.Text == "")

{

MessageBox.Show("Please save the customer first, then add a booking");

}

else

{

dbcon.Select\_booking(Int32.Parse(txtBox\_select\_booking.Text), Int32.Parse(txtBox\_cust\_ref.Text));

}

}

private void IntegerValidationInput(object sender, TextCompositionEventArgs e)

//Regex used to disallow strings to be inputed into the search bar

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

}

}

**Booking**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: GUI class that uses the database facade to retrive information

\* Uses the customer singelton design pattern as well as the database facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

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.Shapes;

namespace SD2CW2

{

/// <summary>

/// Interaction logic for booking.xaml

/// </summary>

public partial class Booking : Window

{

DatabaseFacade dbcon = new DatabaseFacade();

private string cust\_ref { get; set; }

public Booking()

{

InitializeComponent();

Cur\_customer\_Singleton cur\_cust = Cur\_customer\_Singleton.Instance();

this.cust\_ref = cur\_cust.GetRefNum().ToString();

dbcon.OpenCon();

}

private void btn\_save\_booking\_Click(object sender, RoutedEventArgs e)

/\*call the method form the database facade

\* In this the date picker needs to be formated to convert it to a date time and then into a string with the format

\* yyyy-MM-dd - M is capitilized as this refers to months rather than minutes

\* this is because the c# and MySQL database types don't mix well and as such ocasionally requires conversions

\*/

{

if (datepicker\_arrival.Text == "")

{

MessageBox.Show("Please enter an arrival date");

}

else if (datepicker\_dep.Text == "")

{

MessageBox.Show("Please enter a departure date");

}

else if(Convert.ToDateTime(datepicker\_dep.Text) < Convert.ToDateTime(datepicker\_arrival.Text))

{

MessageBox.Show("Departure date must be set to be after the arrival date");

}

else

{

if (txtBox\_booking\_ref.Text == "")

//Method call from the database facade using input from text boxes

{

dbcon.Save\_booking(Convert.ToDateTime(datepicker\_arrival.Text).ToString("yyyy-MM-dd"), Convert.ToDateTime(datepicker\_dep.Text).ToString("yyyy-MM-dd"), Int32.Parse(cust\_ref));

txtBox\_booking\_ref.Text = dbcon.Place\_booking\_ref().ToString();

MessageBox.Show("Booking sucesfully saved");

}

else

{

dbcon.Update\_booking(Convert.ToDateTime(datepicker\_arrival.Text).ToString("yyyy-MM-dd"), Convert.ToDateTime(datepicker\_dep.Text).ToString("yyyy-MM-dd"), Int32.Parse(txtBox\_booking\_ref.Text));

MessageBox.Show("Booking sucesfully saved");

}

}

btn\_invoice.IsEnabled = true; //enable buttons when the booking has been saved

btn\_view\_guests.IsEnabled = true;

btn\_extras.IsEnabled = true;

}

private void bt\_del\_booking\_Click(object sender, RoutedEventArgs e)

//button that calls the method from the database facade

{

if (txtBox\_booking\_ref.Text == "")

{

MessageBox.Show("Fuck you Dennis");

}

else

{

dbcon.Del\_booking(Int32.Parse(txtBox\_booking\_ref.Text));

}

this.Close();

MessageBox.Show("Booking has been removed"); //message box that confirms deleted booking

}

private void btn\_view\_guests\_Click(object sender, RoutedEventArgs e)

//method is called from the database facade when the button is clicked

{

Guests guestWin = new Guests();

guestWin.txtBox\_booking\_ref.Text = txtBox\_booking\_ref.Text;

guestWin.Show();

}

private void btn\_extras\_Click(object sender, RoutedEventArgs e)

//Show the extras window, containing the booking reference

{

Extras extrasWin = new Extras();

extrasWin.txtBox\_booking\_ref.Text = txtBox\_booking\_ref.Text;

extrasWin.Show();

}

private void btn\_invoice\_Click(object sender, RoutedEventArgs e)

{

double extras\_total = 0;

if(dbcon.Extras\_exist(Int32.Parse(txtBox\_booking\_ref.Text)).Count == 0) //used for error checking - if the extras dont exist set the defaul to 0

{

extras\_total = 0;

}

else

{

extras\_total = dbcon.Extras\_invoice(Int32.Parse(txtBox\_booking\_ref.Text));

}

double rooms\_total = 0;

if(dbcon.Guest\_exist(Int32.Parse(txtBox\_booking\_ref.Text)).Count == 0)

{

rooms\_total = 0;

}

else

{

rooms\_total = dbcon.Rooms\_invoice(Int32.Parse(txtBox\_booking\_ref.Text));

}

double total = extras\_total + rooms\_total;

if(rooms\_total == 0) //Won't allow a booking invoice without saving some guests - prevents sql database errors

{

MessageBox.Show("There are no guests in this booking, therefore the invoice cannot exist");

}

else

{

Invoice invoiceWin = new Invoice();

invoiceWin.Show();

invoiceWin.txtBox\_booking\_ref.Text = txtBox\_booking\_ref.Text; //basic invoice, small breakdown of the invoice

invoiceWin.txtBox\_extras\_total.Text = extras\_total.ToString();

invoiceWin.txtBox\_room\_total.Text = rooms\_total.ToString();

invoiceWin.txtBox\_total.Text = total.ToString();

}

}

}

}

**Extras**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: GUI class that will display information about extras as well as be used to update the database

\* Uses the database facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

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.Shapes;

using System.Text.RegularExpressions;

namespace SD2CW2

{

/// <summary>

/// Interaction logic for Extras.xaml

/// </summary>

public partial class Extras : Window

{

DatabaseFacade dbcon = new DatabaseFacade();

public Extras()

{

InitializeComponent();

dbcon.OpenCon();

}

private void btn\_load\_extras\_Click(object sender, RoutedEventArgs e)

/\*

\* A load button made more sense in this scenario as it allows for more efficient less messy code

\* As well as a lower risk of errors occuring via null values

\* As opposed to loading the data in on

\*/

{

dbcon.Load\_extras(Int32.Parse(txtBox\_booking\_ref.Text));

this.Close();

}

private void IntegerValidationInput(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

private void btn\_extra\_save\_Click(object sender, RoutedEventArgs e)

{

int breakfast = 0; //setting standard values for variables so that errors don't break the system

int evening\_meal = 0;

string start\_date = "0001-01-01";

string end\_date = "0001-01-01";

string car\_name = "";

if(txtBox\_breakfast.Text != "")

{

breakfast = Int32.Parse(txtBox\_breakfast.Text); //will use information in the textbox for the database

}

else

{

breakfast = 0;

}

if(txtBox\_evening\_meal.Text != "")

{

evening\_meal = Int32.Parse(txtBox\_evening\_meal.Text); //will use information in the textbox for the database

}

else

{

evening\_meal = 0;

}

if(datepicker\_start.Text != "")

{

start\_date = Convert.ToDateTime(datepicker\_start.Text).ToString("yyyy-MM-dd"); //will use information in the textbox for the database

}

else

{

start\_date = "0001-01-01";

}

if(datepicker\_end.Text != "")

{

end\_date = Convert.ToDateTime(datepicker\_end.Text).ToString("yyyy-MM-dd"); //will use information in the textbox for the database

}

else

{

end\_date = "0001-01-01";

}

if(txtBox\_car\_hire\_name.Text != "")

{

car\_name = txtBox\_car\_hire\_name.Text; //will use information in the textbox for the database

}

else

{

car\_name = "";

}

//all the other values that are set if there is no information in the text boxes that will allow the invoice to show

if(dbcon.Extras\_exist(Int32.Parse(txtBox\_booking\_ref.Text)).Count == 0)

/\*

\* If the textbox containing the extra reference number is empty then the save method should run

\* this takes in the booking reference as an int, the number of breakfasts/evening meals as an int

\* as well as the start and end dates of the car hire, including the name that the car is registered under

\* once again the date time conversion needs to be done becaus eof how c# interacts with MySql

\* the place\_extras\_ref method will place the newly created extras reference in the extras reference textbox for identification purposes.

\*/

{

dbcon.Save\_extras(Int32.Parse(txtBox\_booking\_ref.Text), breakfast, evening\_meal, start\_date, end\_date, car\_name);

txtBox\_extras\_ref.Text = dbcon.Place\_extras\_ref().ToString(); //sets the extras\_ref text box equal to the result from the DatabaseFaceade method

}

else

/\*

\* If there exists a extras reference then the update\_extras method will be run, this is like the save method but takes in the

\* extras reference in order to properly update the correct row in the database

\*/

{

dbcon.Update\_extras(Int32.Parse(txtBox\_booking\_ref.Text), breakfast, evening\_meal, start\_date, end\_date, car\_name);

}

MessageBox.Show("The extras has been sucessfully saved");

btn\_load\_extras.IsEnabled = true;

}

private void btn\_extra\_delete\_Click(object sender, RoutedEventArgs e)

/\*

\* This method will run the method in the databse facade to delete the row with the corresponding extra reference

\* hence the need to include the value in the extras reference

\*/

{

if (txtBox\_extras\_ref.Text == "")

{

MessageBox.Show("Extra must be saved/exist before it can be deleted");

}

else

{

dbcon.Del\_extras(Int32.Parse(txtBox\_extras\_ref.Text));

}

MessageBox.Show("The extras have been deleted");

}

}

}

**Guests**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: GUI class to save guests that relate to a booking

\* Uses the database facade

\*/

using System;

using System.Collections.Generic;

using System.Linq;

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.Shapes;

namespace SD2CW2

{

/// <summary>

/// Interaction logic for Guests.xaml

/// </summary>

public partial class Guests : Window

{

DatabaseFacade dbcon = new DatabaseFacade();

public Guests()

{

InitializeComponent();

dbcon.OpenCon();

}

private void btn\_load\_guests\_Click(object sender, RoutedEventArgs e)

//Calling methods from the database

{

if (dbcon.Guest\_exist(Int32.Parse(txtBox\_booking\_ref.Text)) != null) //check to ensure guest exists

{

dbcon.Load\_guests(Int32.Parse(txtBox\_booking\_ref.Text)); //if exists then load guests

}

else

{

MessageBox.Show("No guests exist in this booking.");

}

this.Close();

}

private void btn\_g1\_save\_Click(object sender, RoutedEventArgs e)

{

if (txtBox\_g1\_pass\_num.Text == "") //validation for the guest textboxes - doesn't allow a guest to be saved without proper string inputs

{

MessageBox.Show("Please insert a passport number for Guest 1");

}

else if(txtBox\_g1\_first.Text == "")

{

MessageBox.Show("Please insert a surname for Guest 1");

}

else if (txtBox\_g1\_last.Text == "")

{

MessageBox.Show("Please insert a first name for Guest 1");

}

else if (datepicker\_g1\_DoB.Text == "")

{

MessageBox.Show("Please insert a date of birth for Guest 1");

}

else if(txtBox\_g1\_diet\_req.Text == "")

{

MessageBox.Show("Please insert dietary requierments for Guest 1 (If none please enter 'None')");

}

else if (txtBox\_g1\_pass\_num.Text == txtBox\_g3\_pass\_num.Text || txtBox\_g1\_pass\_num.Text == txtBox\_g2\_pass\_num.Text || txtBox\_g1\_pass\_num.Text == txtBox\_g4\_pass\_num.Text)

{

MessageBox.Show("Two passport numbers are the same"); //preventing two customers from having the same passport number

}

else

{

if (txtBox\_g1\_pass\_num.Text != dbcon.Pass\_num\_exists(txtBox\_g1\_pass\_num.Text))

{

dbcon.Save\_guest(txtBox\_g1\_pass\_num.Text, txtBox\_g1\_last.Text, txtBox\_g1\_first.Text, Convert.ToDateTime(datepicker\_g1\_DoB.Text).ToString("yyy-MM-dd"), txtBox\_g1\_diet\_req.Text, Int32.Parse(txtBox\_booking\_ref.Text));

MessageBox.Show("Guest 1 saved");

}

else

{

dbcon.Update\_guest(txtBox\_g1\_pass\_num.Text, txtBox\_g1\_last.Text, txtBox\_g1\_first.Text, Convert.ToDateTime(datepicker\_g1\_DoB.Text).ToString("yyy-MM-dd"), txtBox\_g1\_diet\_req.Text, Int32.Parse(txtBox\_booking\_ref.Text));

MessageBox.Show("Guest 1 saved");

}

}

}

private void btn\_g2\_save\_Click(object sender, RoutedEventArgs e)

{

if (txtBox\_g2\_pass\_num.Text == "")

{

MessageBox.Show("Please insert a passport number for Guest 2");

}

else if (txtBox\_g2\_first.Text == "")

{

MessageBox.Show("Please insert a surname for Guest 2");

}

else if (txtBox\_g2\_last.Text == "")

{

MessageBox.Show("Please insert a first name for Guest 2");

}

else if (datepicker\_g2\_DoB.Text == "")

{

MessageBox.Show("Please insert a date of birth for Guest 2");

}

else if (txtBox\_g2\_diet\_req.Text == "")

{

MessageBox.Show("Please insert dietary requierments for Guest 2 (If none please enter 'None')");

}

else if (txtBox\_g2\_pass\_num.Text == txtBox\_g3\_pass\_num.Text || txtBox\_g2\_pass\_num.Text == txtBox\_g4\_pass\_num.Text || txtBox\_g2\_pass\_num.Text == txtBox\_g1\_pass\_num.Text)

{

MessageBox.Show("Two passport numbers are the same");

}

else

{

if (txtBox\_g2\_pass\_num.Text != dbcon.Pass\_num\_exists(txtBox\_g2\_pass\_num.Text))

{

dbcon.Save\_guest(txtBox\_g2\_pass\_num.Text, txtBox\_g2\_last.Text, txtBox\_g2\_first.Text, Convert.ToDateTime(datepicker\_g2\_DoB.Text).ToString("yyy-MM-dd"), txtBox\_g2\_diet\_req.Text, Int32.Parse(txtBox\_booking\_ref.Text));

MessageBox.Show("Guest 2 saved");

}

else

{

dbcon.Update\_guest(txtBox\_g2\_pass\_num.Text, txtBox\_g2\_last.Text, txtBox\_g2\_first.Text, Convert.ToDateTime(datepicker\_g2\_DoB.Text).ToString("yyy-MM-dd"), txtBox\_g2\_diet\_req.Text, Int32.Parse(txtBox\_booking\_ref.Text));

MessageBox.Show("Guest 2 saved");

}

}

}

private void btn\_g3\_save\_Click(object sender, RoutedEventArgs e)

{

if (txtBox\_g3\_pass\_num.Text == "")

{

MessageBox.Show("Please insert a passport number for Guest 3");

}

else if (txtBox\_g3\_first.Text == "")

{

MessageBox.Show("Please insert a surname for Guest 3");

}

else if (txtBox\_g3\_last.Text == "")

{

MessageBox.Show("Please insert a first name for Guest 3");

}

else if (datepicker\_g3\_DoB.Text == "")

{

MessageBox.Show("Please insert a date of birth for Guest 3");

}

else if (txtBox\_g3\_diet\_req.Text == "")

{

MessageBox.Show("Please insert dietary requierments for Guest 3 (If none please enter 'None')");

}

else if (txtBox\_g3\_pass\_num.Text == txtBox\_g4\_pass\_num.Text || txtBox\_g3\_pass\_num.Text == txtBox\_g2\_pass\_num.Text || txtBox\_g3\_pass\_num.Text == txtBox\_g1\_pass\_num.Text)

{

MessageBox.Show("Two passport numbers are the same");

}

else

{

if (txtBox\_g3\_pass\_num.Text != dbcon.Pass\_num\_exists(txtBox\_g3\_pass\_num.Text))

{

dbcon.Save\_guest(txtBox\_g3\_pass\_num.Text, txtBox\_g3\_last.Text, txtBox\_g3\_first.Text, Convert.ToDateTime(datepicker\_g3\_DoB.Text).ToString("yyy-MM-dd"), txtBox\_g3\_diet\_req.Text, Int32.Parse(txtBox\_booking\_ref.Text));

MessageBox.Show("Guest 3 saved");

}

else

{

dbcon.Update\_guest(txtBox\_g3\_pass\_num.Text, txtBox\_g3\_last.Text, txtBox\_g3\_first.Text, Convert.ToDateTime(datepicker\_g3\_DoB.Text).ToString("yyy-MM-dd"), txtBox\_g3\_diet\_req.Text, Int32.Parse(txtBox\_booking\_ref.Text));

MessageBox.Show("Guest 3 saved");

}

}

}

private void btn\_g4\_save\_Click(object sender, RoutedEventArgs e)

{

if (txtBox\_g4\_pass\_num.Text == "")

{

MessageBox.Show("Please insert a passport number for Guest 4");

}

else if (txtBox\_g4\_first.Text == "")

{

MessageBox.Show("Please insert a surname for Guest 4");

}

else if (txtBox\_g4\_last.Text == "")

{

MessageBox.Show("Please insert a first name for Guest 4");

}

else if (datepicker\_g4\_DoB.Text == "")

{

MessageBox.Show("Please insert a date of birth for Guest 4");

}

else if (txtBox\_g4\_diet\_req.Text == "")

{

MessageBox.Show("Please insert dietary requierments for Guest 4 (If none please enter 'None')");

}

else if (txtBox\_g4\_pass\_num.Text == txtBox\_g3\_pass\_num.Text || txtBox\_g4\_pass\_num.Text == txtBox\_g2\_pass\_num.Text || txtBox\_g4\_pass\_num.Text == txtBox\_g1\_pass\_num.Text)

{

MessageBox.Show("Two passport numbers in two seperate guests are the same.");

}

else

{

if (txtBox\_g4\_pass\_num.Text != dbcon.Pass\_num\_exists(txtBox\_g4\_pass\_num.Text))

{

dbcon.Save\_guest(txtBox\_g4\_pass\_num.Text, txtBox\_g4\_last.Text, txtBox\_g4\_first.Text, Convert.ToDateTime(datepicker\_g4\_DoB.Text).ToString("yyy-MM-dd"), txtBox\_g4\_diet\_req.Text, Int32.Parse(txtBox\_booking\_ref.Text));

MessageBox.Show("Guest 4 saved");

}

else

{

dbcon.Update\_guest(txtBox\_g4\_pass\_num.Text, txtBox\_g4\_last.Text, txtBox\_g4\_first.Text, Convert.ToDateTime(datepicker\_g4\_DoB.Text).ToString("yyy-MM-dd"), txtBox\_g4\_diet\_req.Text, Int32.Parse(txtBox\_booking\_ref.Text));

MessageBox.Show("Guest 4 saved");

}

}

}

private void btn\_g1\_delete\_Click(object sender, RoutedEventArgs e)

//delete method for guest 1

{

if (txtBox\_g1\_pass\_num.Text == "")

{

MessageBox.Show("First create/save Guest 1 before deleting");

}

else

{

dbcon.Del\_guest(txtBox\_g1\_pass\_num.Text);

}

MessageBox.Show("Guest 1 removed");

}

private void btn\_g2\_delete\_Click(object sender, RoutedEventArgs e)

{

if (txtBox\_g1\_pass\_num.Text == "")

{

MessageBox.Show("First create/save Guest 2 before deleting");

}

else

{

dbcon.Del\_guest(txtBox\_g2\_pass\_num.Text);

}

MessageBox.Show("Guest 2 removed");

}

private void btn\_g3\_delete\_Click(object sender, RoutedEventArgs e)

{

if (txtBox\_g1\_pass\_num.Text == "")

{

MessageBox.Show("First create/save Guest 3 before deleting");

}

else

{

dbcon.Del\_guest(txtBox\_g3\_pass\_num.Text);

}

MessageBox.Show("Guest 3 removed");

}

private void btn\_g4\_delete\_Click(object sender, RoutedEventArgs e)

{

if (txtBox\_g1\_pass\_num.Text == "")

{

MessageBox.Show("First create/save Guest 4 before deleting");

}

else

{

dbcon.Del\_guest(txtBox\_g4\_pass\_num.Text);

}

MessageBox.Show("Guest 4 removed");

}

}

}

**Invoice (there isn’t anything in here)**

/\*

\* Author: Andre Moazed Matricualtion number: 40216327

\* Class description: GUI class with information only being loaded into it

\*

\*/

using System;

using System.Collections.Generic;

using System.Linq;

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.Shapes;

namespace SD2CW2

{

/// <summary>

/// Interaction logic for Invoice.xaml

/// </summary>

public partial class Invoice : Window

{

public Invoice()

{

InitializeComponent();

}

}

}