***TAWANDA MADZIYA GRADE 12B***

***IT PAT @2020***



**Contents**

**Scenario pg3**

**User requirements pg4**

**Navigation pg5**

**Data Dictionary pg6-7**

**GUI pg7-10**

**Database Design pg10-12**

**IPO pg12-14**

***Scenario and Scope***

***Alegence airlines*** is an airline company founded by John Cane who is an American entrepreneur. Alegence airline started with only a single aeroplane in 1980 and has since then continued to grow year by year and at this current moment there is a fleet of over 300 aeroplanes and as we know, with the growth of the company there is also a need for more staff and now the airline has hired 3 000 staff members and 20 administrators over the years.

The airline company has had problems with the way that they are conducting their business and how the are unable to do most things that modern airline companies are able to do. Keeping a clear record of customer details(ID, Name and Surname, Age, Email, Phone number, address) has been hard because the company uses normal counter books which can be easily misplaced. Keeping track of the number of passengers on a single aeroplane has proven to be a challenge. Being able to differentiate between customers with some of the same details has been a challenge. Attendants have been changing customer details in-order to steal money from **Alegence airlines**

The only possible solution is to start relying on a more technological approach and by using a digital system, there will be security and accuracy in terms of the details of all customers and the chances of errors occurring will be far less thus a decrease in money that has been wasted it the past as compensation to customers.

The system will be able to keep track of the number of seats that are available on a flight. The system will be able to obtain and store customer details. The system will create a unique ID to identify a customer incase someone with the same Name and surname appears. The system will activate only when correct passwords and usernames are entered by authorized persons. The system will allow the changing of certain details ONLY when authorized by administrators. The system will be able to generate a unique ticket number for the customer.

Class Diagram:

A CLASS and OBJECT will be used when a customer is added to the database. This is to ensure simplicity and convenience and much more effectiveness for the system. The class will be called “TCustomer” and object will be called “objCustomer”.

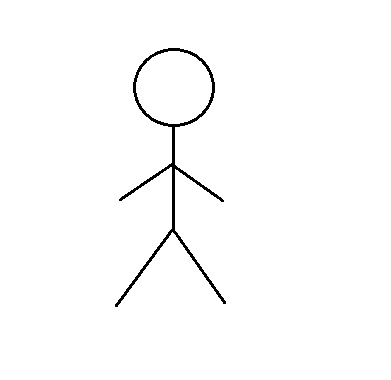
|  |
| --- |
| TCustomer |
| Attributes:   * fName * fSurname * fEmail * fDestination * fCellNumber * fDOD * fPrice |
| Methods:  + constructor>> Create(sName, sSurname, sEmail, sID, sCellNumber: string)  + getName: string  + getSurname: string  + getEmail: string;  + getID: string;  + getCellNumber:string;  + getDateOfDeparture:string;  +getPrice:real; |

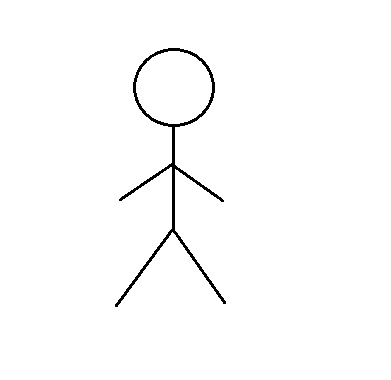
**USER REQUIREMENTS**

**Airline Bookings**

<Include>

<Extend>





<Include>

Customer

Attendants

<Extend>

<Extend>

**Flow Diagram**

Open main form

Enter password and userID

Open welcome form

Open form to add new customer

Open form to edit customer details

Open form to remove a customer

Enter customer ID

Enter customer name

Enter customer name

Locate customer

Enter customer surname

Enter customer Surname

Select details that need correction

Enter customer email address

Enter customer ID

Enter customer home address

Click edit customer button

Click Remove button

Enter customer cell number

Enter customer ID

Select flight destination

Select departure date

Enter number of seats desired by customer

Click add customer button

Show suitable message

Exit form

Data Dictionary

Access

TblAttendents

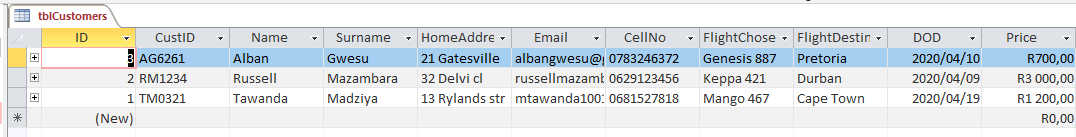
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Attribute Name | Required | Type | Length | Default Values | Notes | | AttendentID | Yes | String | 5 | Primary key | Used to uniquely identify the user. | | DOD | Yes | Date/Time |  |  | Date of Departure of customer flight | | CustID | Yes | String | 25 | (Foreign key) | To uniquely identify the customer | | FlightDestination | No | String | 25 |  | Flight chosen links with system to keep record of available seats | | Price | Yes | Currency |  |  | Can be used as a comparison to other flight prices | |

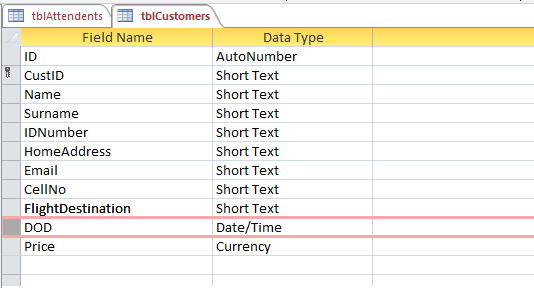
TblCustomers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Required | Type | Length | Default Values | Notes |
| CustomerID | Yes | String | 25 | (primary key) | To uniquely identify the customer |
| Name | Yes | String | 25 | n/a | First Name of customer only |
| Surname | Yes | String | 25 | n/a | Surname of customer |
| Address | Yes | String | 50 | n/a | Where the Customer lives |
| Destination | Yes | String | 250 | n/a | Where customer is going to |
| Price | Yes | Number |  | n/a | Price of flight |
| DOD | Yes | Date/Time |  | n/a | Date and time of departure |
| Email | Yes | String | 25 |  | Used as a form of informing customer about changes or any additional information |

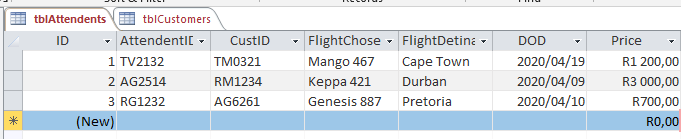
Database Design

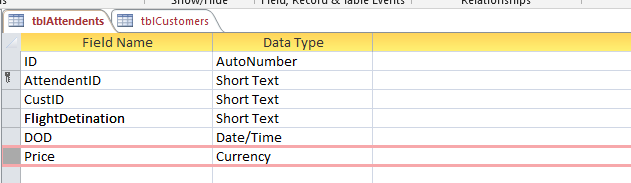
TblCustomers



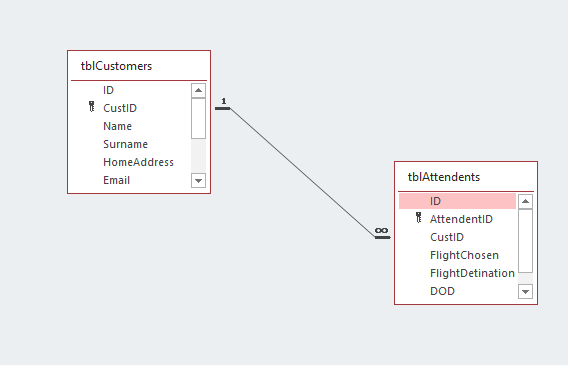


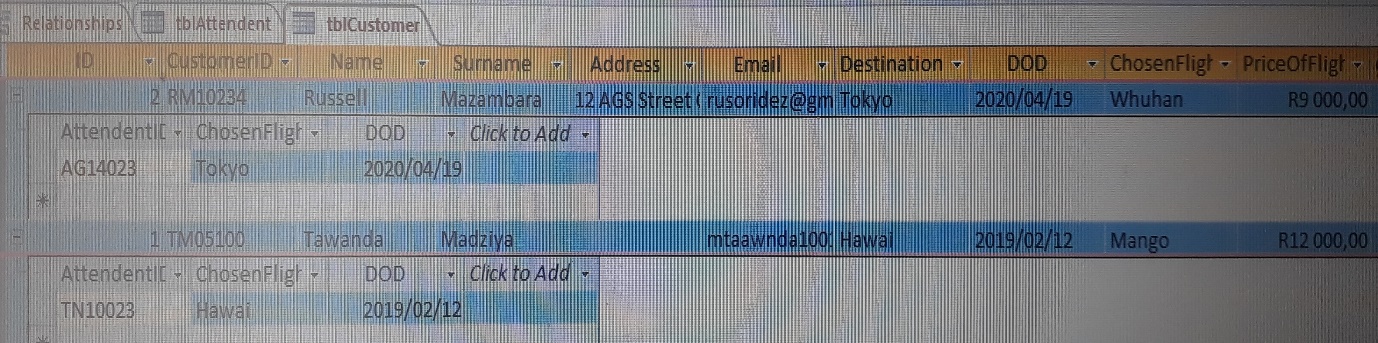
TblAttendents





Relationships

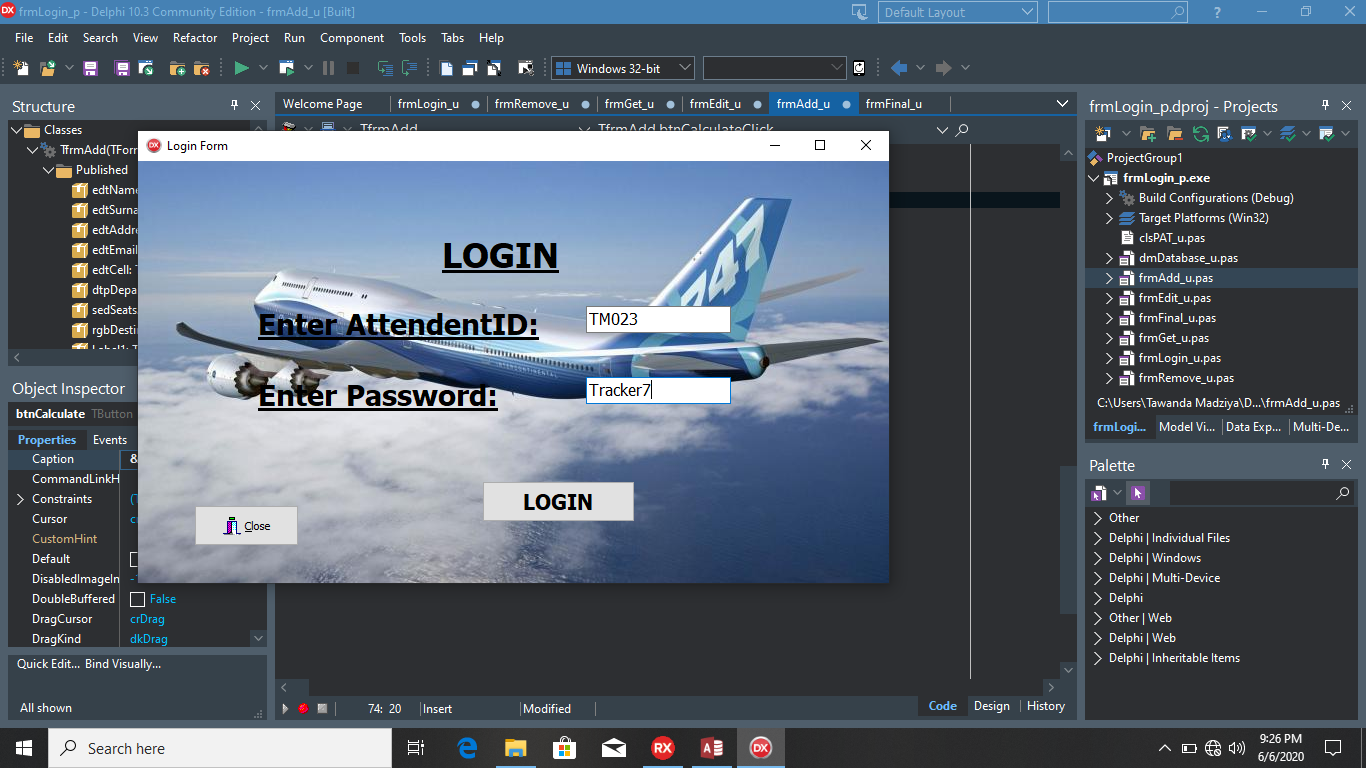




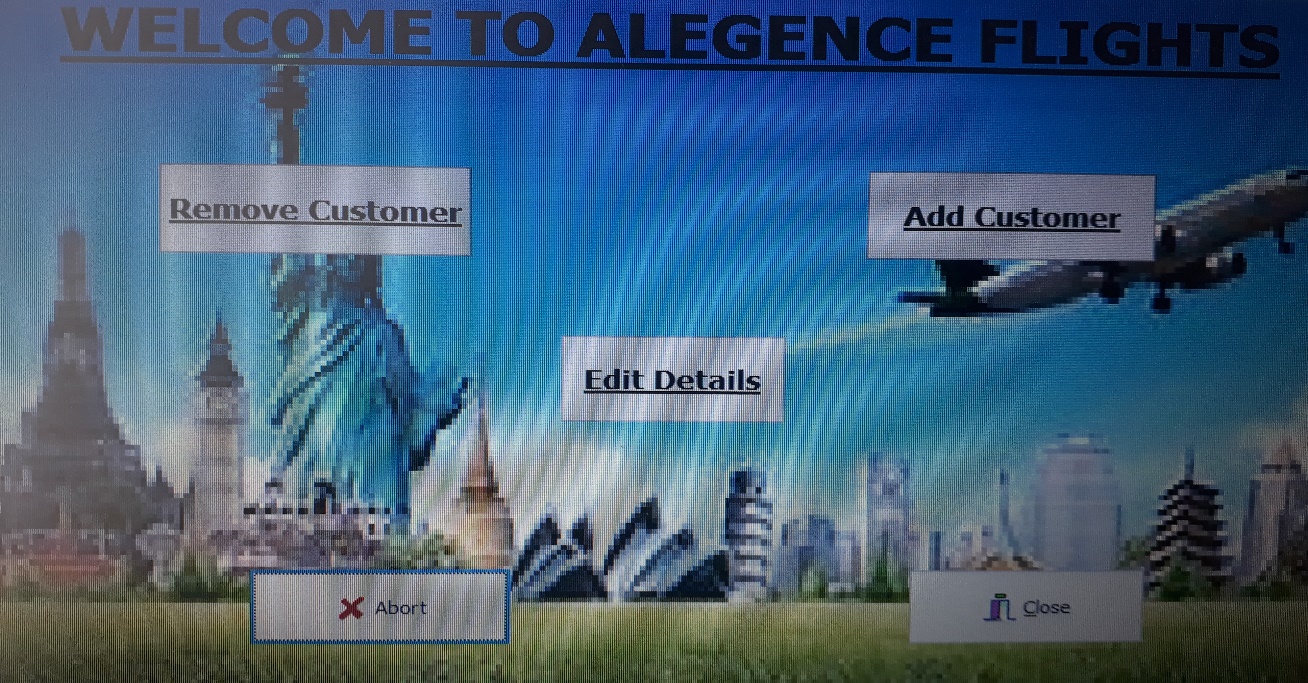
The above view of the database design shows a clear highlight on the fields that will be used and also helps to show a clear understanding on the relevancy of the fields being used. All relationships between tables are shown and the Primary keys in both tables will be the CustID with will also be seen as a foreign key in the opposing table.

In both tables it is shown in the last image how the different fields will be relying on each other and how they will make the system more endurable and more manageable to the user in the way that each detail requires specific characters and security is one of the key features in the database.

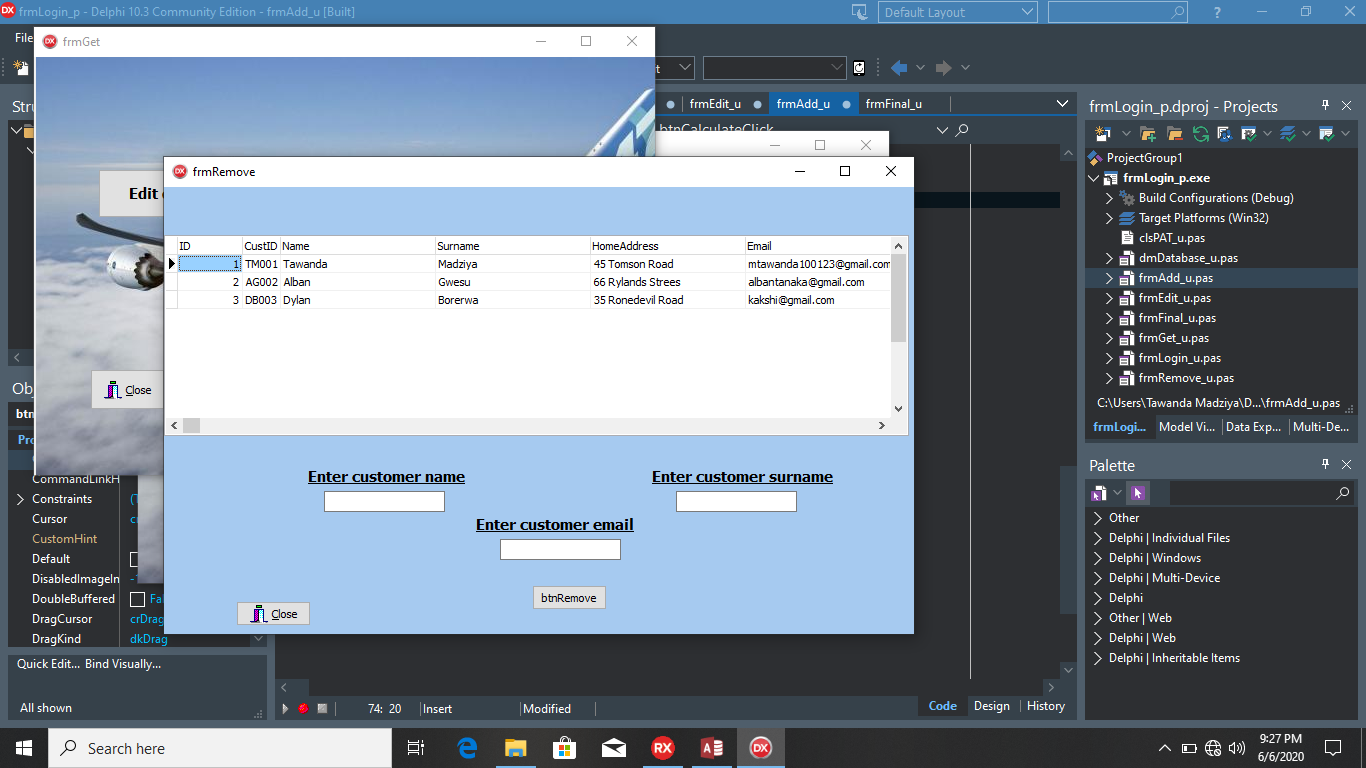
G.U.I



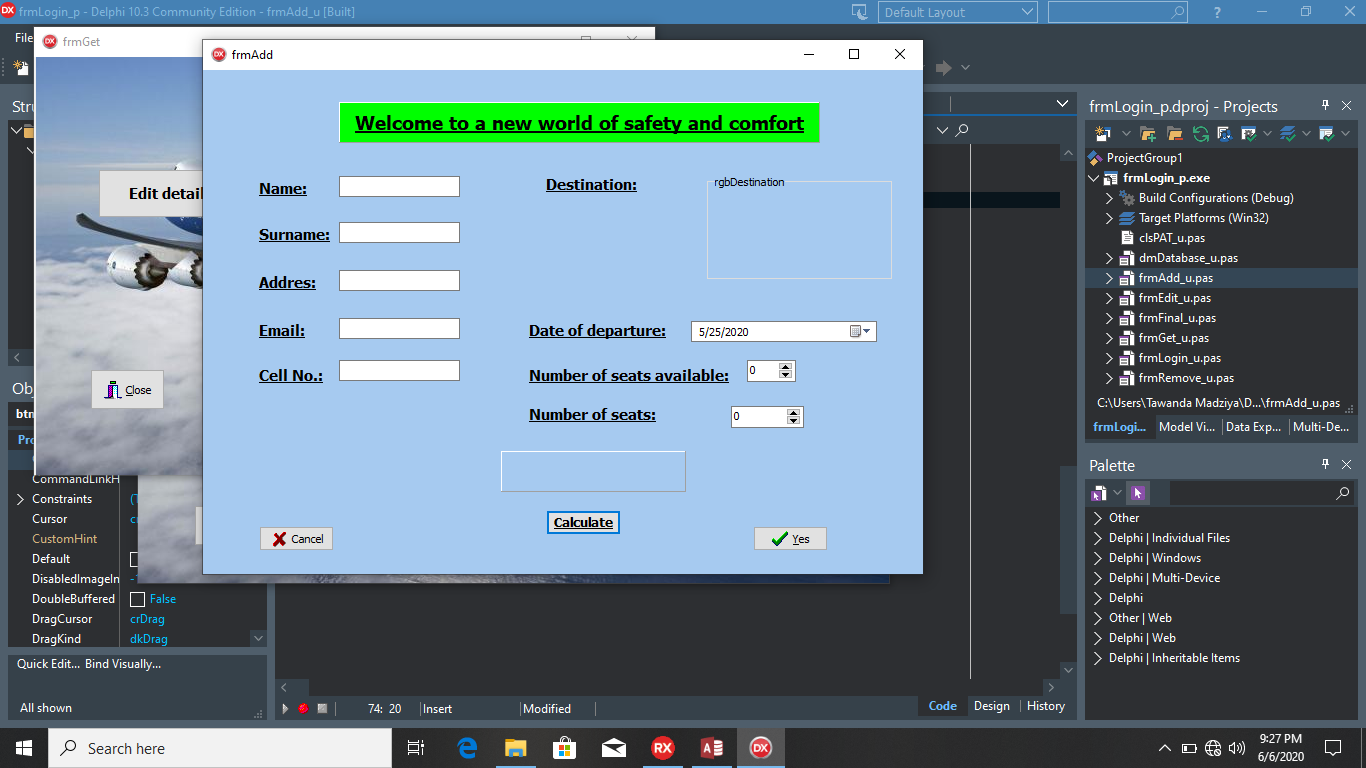
1] The system starts off with a form[frmLogin], whereby the user[Admin] is asked for his/her AdminID and password which has already been issued to the admins and attendants and if either of those is incorrect then a suitable message will appear and the user will have to put in the corrects details again.



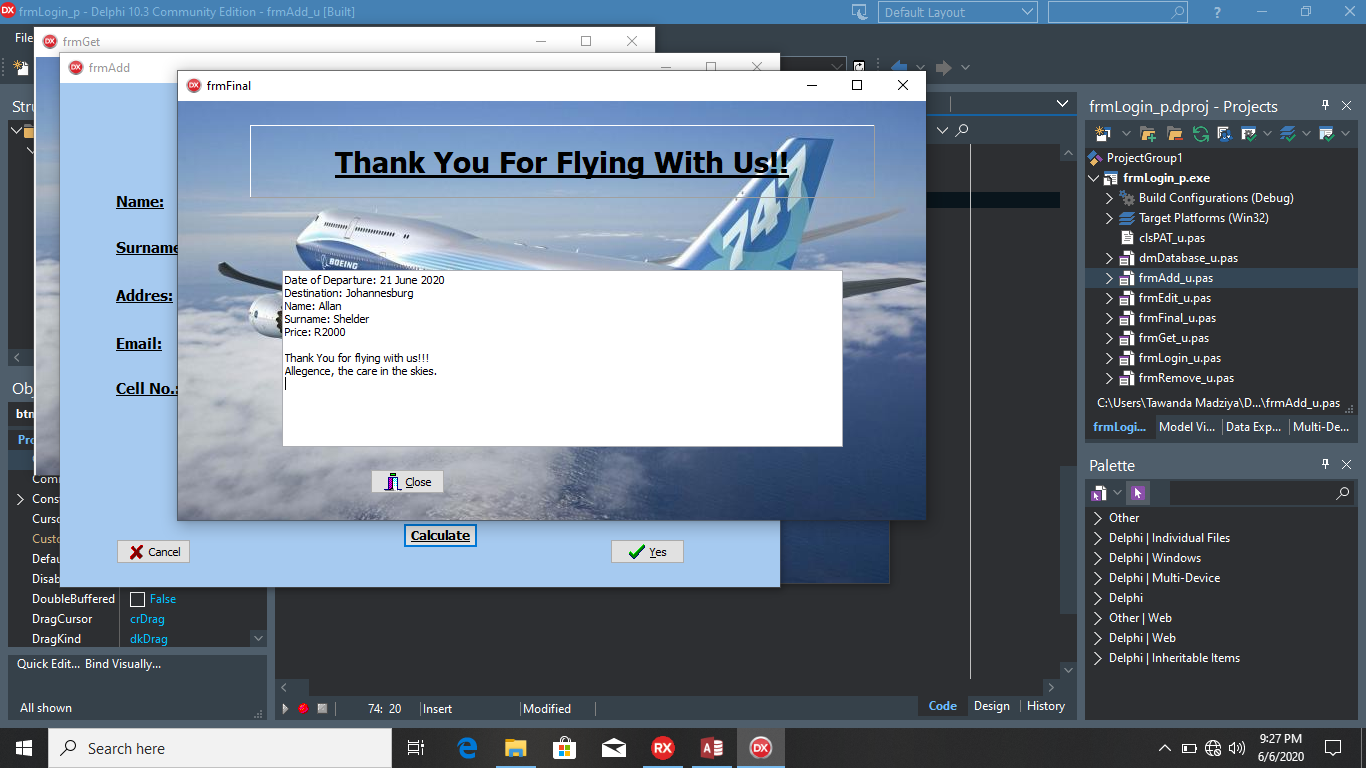
2] if all details entered in the first image are corrects the this will be the next form to appear and on this form[frmWelcome], the user will be allowed to either choose to change customer details or to add another customer.



3] in this form this is where the customer will be allowed to make changes such as removing a specific customer after searching for the customer by entering their name, surname and email address since that is the most unique details that identify individual customers besides their ID numbers.



4] The form[frmAdd] will be used to enter the customer’s, name, surname, address, email, phone number, destination, date of departure and the desired number of seats. All information will be saved to database.



5) This form[frmFinal] will be used to display the boarding ticket for the customer and also the necessary information that will be physically asked when the customer boards their assigned flight.

IPO METHOD

**(FORMAT, DATA TYPE, Validation, Processing, OUTPUT )**

**­Format**

The system I created only requires the date to be formatted . The date of departure and arrival format- yyyy/mm/dd

**­Data types**

Data types will differ according to the value needed. Data types will be, strings, integers, datetime and number in accordance to their uses in the system.

**­Validations**

Multiple Validations will take place throughout the system such as presence check, range check, data types check. E.g. In Algorithm

Begin

If CellNum>10 OR CellNum<10

Showmessgae(“Incorrect cell number”);

edtCellNum.Text:=” “;

End.

**­Processes**

The main processing that will be done by the system will be, creating a unique Customer ID{CustID}. The Customer ID will most likely be seen under that name of a single customer for as long as the system is operational so as long as more customers with the same initials comes to book flights then the Customer ID will remain unique to only one customer. The Customer ID will be made by generating a random number and taking the first initials of the customer’s name and surname.

**­ Outputs**

The outputs of the system will be seen using a message or even through sound depending on the preference of the user or admin who will be using the system for the benefit of the company but a ticket for the customer will be printed showing the time and date of departure as well as the time and date of arrival.

**IPO**

**INTERFACE 1 - Customer**

**INPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INPUT** | **COMPONENT** | **DATA TYPE** | **SOURCE** | **FORMAT** |
| Name | TEdit | String | Keyboard Entry | Text |
| Surname | TEdit | String | Keyboard Entry | Text |
| ID | TEdit | String | Keyboard Entry | Text |
| Contact Number | TEdit | String | Keyboard Entry | Text |
| Email | TEdit | String | Keyboard Entry | Text |
| Address | TEdit | String | Keyboard Entry | Text |
| Select Destination | TRadioGroup | String | Mouse Selection | Text |
| Select date of departure | DateTimePicker | String | Mouse Selection | Text |
| Select desired number of seats | SpinEdit | Integer | Keyboard Entry  Selection | Value |

**VALIDATION**

|  |  |  |
| --- | --- | --- |
| **WHAT WILL BE VALIDATED** | **ALGORITHM** | **MESSAGE** |
| Check if ID fits all character requirements | var  iID:integer;  begin  iID:=Length(edtID.Text);  if (iID>13) OR (iID<13) then  MessageDlg('ID number is incorrect please try again!',mtError,[mbOK],0);  edtID.SetFocus;  end; | ‘ID number is incorrect please try again’ |
| Check if contact number fits all character requirements | var  iNumber:integer;  begin  iNumber:=Length(edtCellNo.Text);  if (iNumber>10) OR (iNumber<10) then  MessageDlg('Phone number is incorrect please try again!',mtError,[mbOK],0);  edtCellNo.SetFocus;  end; | ‘Phone number is incorrect please try again’ |
| **Check if Name is entered** | var  sName:string;  begin  sName:=edtName.Text;  if sName='' then  MessageDlg('Please enter Name!',mtError,[mbOK],0);  edtName.SetFocus;  end; | 'Incorrect PIN' |
| Check if Surname is entered | var  sSurname:string;  begin  sSurname:=edtISurname.Text;  if sSurname='' then  MessageDlg('Please enter Surname!',mtError,[mbOK],0);  edtSurname.SetFocus;  end; | 'Please fill in required fields' |
| Check if Address is entered | var  sAddress:string;  begin  sAddress:=edtIAddress.Text;  if sAddress='' then  MessageDlg('Please enter Address!',mtError,[mbOK],0);  edtIAddress.SetFocus;  end; | 'Incorrect ID' |
| Check if Email is entered | var  sEmail:string;  begin  sEmail:=edtEmail.Text;  if sEmail='' then  MessageDlg('Please enter Email!',mtError,[mbOK],0);  edtEmail.SetFocus;  end; | 'Please fill in all required fields' |
| Check if destination is chosen | begin  if rgbDestination.ItemIndex=-1 then  messagedlg('Please choose a destination',mtWarning,[mbOK],0);  end; | ‘Please choose a destination’ |

**PROCESSING**

|  |  |
| --- | --- |
| **WHAT WILL BE PROCESSED** | **ALGORITHMS** |
| Creating a Unique CustomerID(CustID) | var  sCustID:string;  iRandom:integer;  begin  Randomize;  sCustID:=Copy(edtName.Text,1,1) ;  sCustID:=sCustID+Copy(edtSurname.Text,1,1);  sCustID:=UpperCase(sCustID);  iRandom:=RandomRange(1000,9900);  sCustID:=sCustID+IntToStr(iRandom);  end; | |
| The information the guest entered will be stored in a text file which can be accessed by the proper authority | AssignFile(tCustomer, 'Customer.txt');  Append(tCustomer);    Writeln(tCustomer, sName);  Writeln(tCustomer, sSurname);  Writeln(tCustomer, sID);  Writeln(tCustomer, stEmail);  Writeln(tCustomer, sPhoneNumber);  Writeln(tCustomer, sAddress);  Writeln(tCustomer, sDateOfDeparture);  Writeln(tCustomer, sDestination);  Writeln(tCustomer, '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');  CloseFile(tCustomer); | |
| Music for comfort of customer | begin  MediaPlayer1.Open;  end; | |

**OUTPUT**

|  |  |  |
| --- | --- | --- |
| **WHAT WILL BE DISPLAYED** | **FORMAT** | **COMPONENT** |
| **TABSHEET 7** | | |
| Boarding ticket of passenger appears with minor yet needed details | Memticket.lines.add(‘Alegence Airlines’+#13sName+’ ’+sSurname+#13+’Date of departure: ’+sDOD+#13+’Flight Destination: ‘+sDestination+#13+’Unique boarding ID: ‘+sCustID+#13+’Thank you for choosing Alegence Airlines’); | Memo |
| Information of business that is required by customer will be made available | redInfo.lines.add(‘***Alegence airlines*** is an airline company founded by John Cane who is an American entrepreneur. Alegence airline started with only a single aeroplane in 1980 and has since then continued to grow year by year and at this current moment there is a fleet of over 300 aeroplanes and as we know, with the growth of the company there is also a need for more staff and now the airline has hired 3 000 staff members and 20 administrators over the years.  ’); | RichEdit |

**INTERFACE 2 - Attendants**

**INPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INPUT** | **COMPONENT** | **DATA TYPE** | **SOURCE** | **FORMAT** |
| **TABSHEET 1** | | | | |
| AttendentID | TEdit | String | Keyboard Entry | Text |
| Password | TEdit | String | Keyboard Entry | Text |
| CustID | Input Box | String | Keyboard Entry | Text |
| Flight destination | Input Box | String | Keyboard Entry | Text |
| Date of departure | Input Box | String | Keyboard Entry | Text |
| Price | TEdit | String | Keyboard Entry | Text |

**VALIDATION**

|  |  |  |
| --- | --- | --- |
| **WHAT WILL BE VALIDATED** | **ALGORITHM** | **MESSAGE** |
| **TABSHEET 1** | | |
| Checks whether password is correct | if edtPassword.Text<>’Alpha Travel’ then  begin  messagedlg(‘Password is incorrect please re-enter the corrects password’,mtError,[mbOK],0);  edtPassword.SetFocus;  end  else  frmWelcome.visible:=True;  end; | ‘Password is incorrect please re-enter the corrects password’ |
| **Checks whether Username meets there suitable requirements** | if edtSAttendentID.Text<>’AT1232 then  begin  messagedlg(‘AttendentID is incorrect please re-enter the corrects password’,mtError,[mbOK],0);  edtAttendentID.SetFocus;  end  else  frmWelcome.visible:=True;  end; | ‘Password is incorrect please re-enter the corrects password’ |

**PROCESSING**

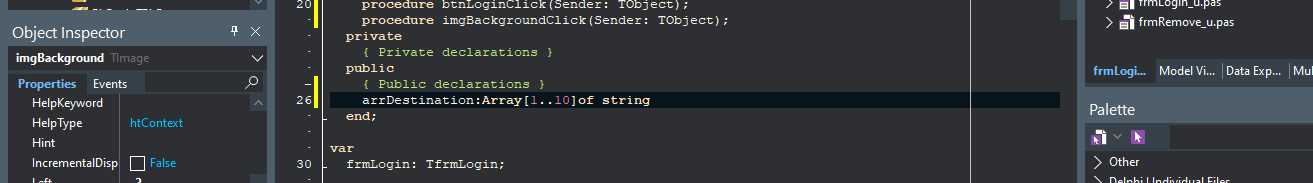
|  |  |
| --- | --- |
| **WHAT WILL BE PROCESSED** | **ALGORITHMS** |
| **TABSHEET 3** | | |
| The attendants can sort the customers according to their number of flight bookings | dmDatabase.DataModule1.tblCustomer.Sort := 'CustID ASC'; | |
| Attendants can find a specific customer by just typing their name address | sEmail := edtLocate.Text;  if dmDatabase.DataModule1.tblCustomer.Locate('Email', sEmail, []) = true  then  begin  ShowMessage(dmDatabase.DataModule1.tblCustomer[‘Name']  + ' booked flight to ‘+dmDatabase.DataModule1.tblCustomer['FlightDestination']);  end  else  begin  MessageDlg('Customer does not exist',mtError,[mbOK],0); | |
| Attendants might make mistakes so they are therefor able to change customer details | iCorrection := StrToInt(InputBox(‘CustID’,  'Enter a Booking Code you want to change', ''));  sField := InputBox('Field To Change',  'Enter name of field that requires change', '');  if sField = 'Name' then  sNameEdit := InputBox('Name', 'Enter new name’, '')  else if sField = 'Surname' then  sSurnameEdit := InputBox('Surname',  'Enter new surname', '')  else if sField = 'Email' then  sEmailEdit := InputBox('Email’,  'Enter ne Email', '')  else if sField = 'ID' then  sIDEdit := InputBox('Age', 'Enter new ID number', '')  else if sField= 'Address' then  sAddressEdit := InputBox('Address',  'Enter new address’, '')  else if sField = 'CellNo' then  sCellNoEdit := InputBox('Contact Number',  'Enter new contact number ', '')  with dmDatabase.DataModule1 do  begin  if ADOTableGuests.Locate('Booking\_Code', iBookingCodeEdit, []) = true then  begin  ADOTableGuests.Edit;  ADOTableGuests['Name'] := sNameEdit;  ADOTableGuests['Name'] := sSurnameEdit;  ADOTableGuests['ID'] := sIDEdit;  ADOTableGuests['Address'] := sAddressEdit;  ADOTableGuests[‘CellNo’] := sCellNoEdit;  ADOTableGuests.Post;  end; | |
| Attendants are able to locate customers and their information who have booked flights in the past | AssignFile(tCustomer, 'Customer.txt');  try  Reset(tCustomer);  except  ShowMessage('File not found');  Exit  end;  while NOT Eof(tCustomer) do  begin  Readln(tCustomer, sLine);  redOutput.Lines.Add(sLine);  end;  CloseFile(tCustomer); | |

**OUTPUT**

|  |  |  |
| --- | --- | --- |
| **WHAT WILL BE DISPLAYED** | **FORMAT** | **COMPONENT** |
| **TABSHEET 4** | | |
| Attendants will be able to access the text file holding all the customer details that have booked flights | AssignFile(tCustomer, 'Customer.txt');  try  Reset(tCustomer);  except  ShowMessage('File not found');  Exit  end;  while NOT Eof(tCustomer) do  begin  Readln(tCustomer, sLine);  redGuests.Lines.Add(sLine);  end;  CloseFile(tCustomer); | RichEdit |
| Attendants will be able to access the text file containing flight tickets | AssignFile(tTicket, 'Ticket.txt');  try  Reset(tTicket);  except  ShowMessage('File not found');  Exit  end;  while NOT Eof(tTicket) do  begin  Readln(tTicket, sLine);  redPayments.Lines.Add(sLine);  end;  CloseFile(tExtras); | RichEdit |

**ARRAY**

An array, called “ArrDestination”, will be used in the program to store the names of the available destinations where the flights are authorized to go, for example, Cape Town, Durban, Pretoria, etc. This will be easier and safer because only specific persons can change the flight destinations and there will be no errors if the destinations are already inserted other than letting a person type the destination. The array will be populated when the program is initiated.



**TextFile**

A textfile, called Attendents.txt, will be used as a secondary security measure to keep track of the different employees that make changes or add new customers to the system and this will ensure full security and transparency to the company. The textfile will be composed of the employee’s UserName and the employee’s full name, surname, date of login and time of login will be kept and the existence of the textfile will only be know by the management and owners of the company.

