CG2.3: Program Documentation

CG2.3/1 – Data structures and variables

1). Data dictionary of variables and constants and fields in tables

Form	Subroutine where used	Name	Type	Purpose	Page # of code
frmAddCustomer	cmdAddCustomer	rsCustomer	Recordset Variable	Links the add customer form to the customer's table	7
frmAddPeformance	cmdAdd Performance	PerformanceID	Integer variable	Stores the performance ID for the 'Add performance' form	8
	CreateNew Performance	rsPerformance	Recordset Variable	Links the add performance form to the performance table	9
	GetPerformanceID	sqlGetID	String Variable	SQL Select statement returning the performance ID for the performance which was just created	9
	CreateNewBooking	sqlAdd	String Variable	SQL Insert statement, inserting seats into the bookings table	10
frmAddUser	cmdAddUsers_Click	rsUser	Recordset Variable	Links the adduser form to the security table so that a user can be created	11
	cmdDeleteUser_Click	lngID	Long variable	Holds the user ID	12
		SQLdeleteuser	String variable	SQL Delete statement deleteing records in a table	12
frmAvailableSeats	cmbPerformance_Click	PerformanceID	Integer variable	Stores the performance ID so that a user can view available seats for particular performance	14
	GetAvailableSeats	sqlseats	String Variable	SQL Select statement returning values from booking and seat tables	16
	Form_Load	sqlPerformance	String Variable	SQL Select statement to return values from the performance table	13
frmBookings	Form_Load	sqlCustomer	String Variable	SQL Select statement to return values from the customer table	17
	cmdAddBookings_Click	Oitem	Variant Variable	Gets a single item from the listbox	17
		bookingID	String Variable	Stores the bookingID	17
		selectedBooking IDs	String Variable	Holds the total number of booking id'd selected	17
		count	Integer variable	Used to check if it is the last seat	17
		PerformanceID	String variable	Holds the performanceID	17
		CustomerID	Integer variable	Holds the customerID	17
		totalItemsSelect ed	Integer variable	Holds the total number of seats selected	17

Form	Subroutine where used	Name	Type	Purpose	Page # of code
frmBookings	cmdAddBookings_Click	isSeatDisabled	Boolean Variable	Checks to see if there is a disabled seat	18
	UpdateBooking	rsUpdateBookings	Recordset Variable	Links the bookings form to the bookings table	20
	ShowAvailableSeats	PerformanceID	Integer variable	Holds the performance ID	21
		row	String variable	Holds the row to filter the listbox for seats on the bookings table	21
		sqlAvailable	String variable	SQL Select statement returning values from the booking and seat tables	21
	GetTotalBooking	Oitem	Variant variable	Holds the count for how many rows have been booked	22
		Price	Double variable	Holds the price for the seats	22
		totalbooking	Double variable	Totals the prices in the price variable	22
	cmdAddBookings_Click	whereStr	String variable	Passed as a filter to the report	17
frmCustomer	Form_Load	sqlCustomer	String variable	SQL Select statement returning values from the customer table	23
	cmdDelete_Click	lngID	Long Variable	Stores the selected customer's ID	24
		sqlDelete	String variable	SQL Delete statement deleting records in the customer table	24
	cmdUpdate_Click	CustomerID	Integer variable	Holds the customer ID to pass on to the update form	25
frmLogin	cmdLogin_Click	rsUser	Recordset Variable	Links the login form to the security table	25
frmRefund	Form_Load (frmRefund)	sqlCustomerID	String variable	SQL select statement to access the customer table and return some values	28
	cmbCustomer_Click	CustomerID	Integer variable	Holds a customer ID	28
		sqlBookingsdetail	String variable	SQL select statement to access the bookings and seats table and return some given values	28
	cmdRefund_Click	count	Integer variable	Is a running count of the bookings	29
		CustomerID	String variable	Holds the customer ID	29
		Oitem	Variant Variable	Looks at how many bookings are selected	29
		seatID	String variable	Holds the Seat ID	29
		totalRefund	Double variable	Holds the total refund price	29

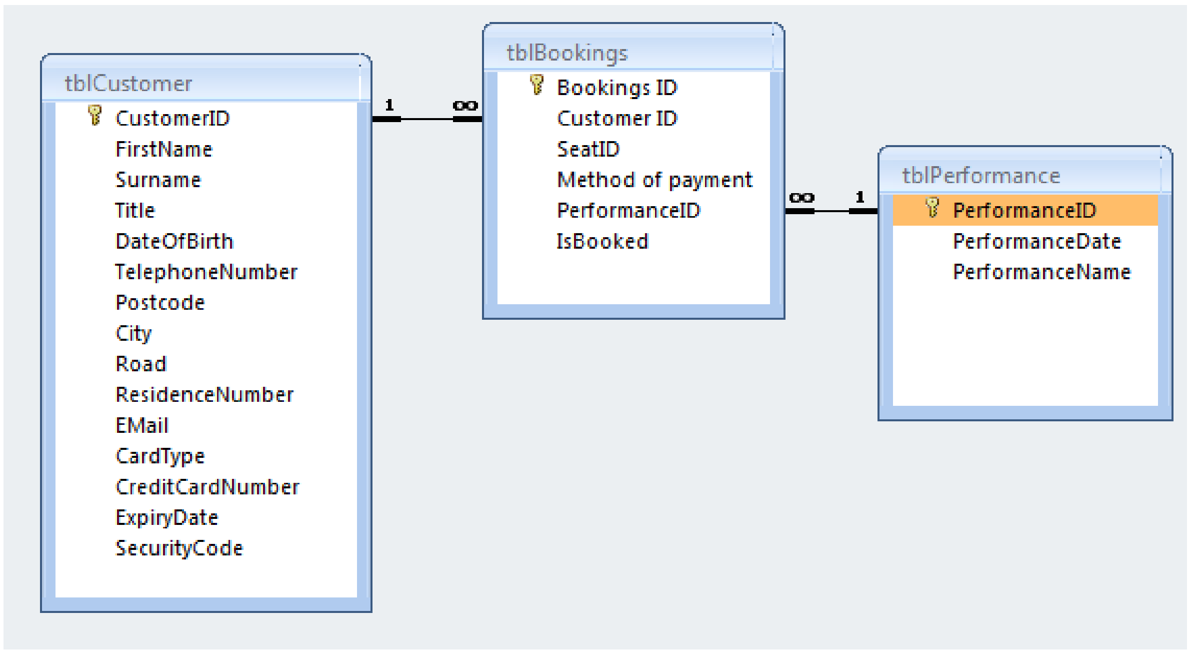
Form	**Subroutine where used**	**Name**	**Type**	**Purpose**	**Page # of code**
frmUpdate	Form_Load (frmUpdate)	Customer_ID	String variable	Holds the customerID	32
		Pstring	Variant Variable	Builds an array of arguments passed to the form	32
		sqlID	String variable	SQL SELECT statement getting the customerID	32
		rsCustomer	Recordset Variable	Links the update form to the customers form	32
	cmdUpdateCustomer_Click	Customer_ID	String variable	Holds the customerID	34
		Pstring	Variant Variable	Builds an array of arguments passed to the form	34
		sqlUpdate	String variable	SQL Update statement to update the customer details	34
frmViewBookings	Form_Load (frmViewBookings)	sqlCustomerID	String variable	SQL SELECT statement getting the customer ID	35
	cmbCustomer_Click	CustomerID	Integer variable	Holds the customer ID	36
		sqlCustomerDetail	String variable	SQL SELECT statement getting values from the customer table	36
		sqlBookingsdetail	String variable	SQL SELECT statement getting values from the Bookings table	36
frmLogin	cmdLogin_Click	rsUser	Recordset Variable	Links the login form to the security table	25
frmMainMenu	Form_Load	rsLevel	Recordset Variable	Links the Main Menu form to the security table	26
	cmdLogOut_Click	answer	Integer Variable	Stores the answer to the messagebox	27
rptReceipt	Report_Load	Customer_ID	String variable	Holds the customer ID	37
		Pstring	Variant Variable	Builds an array of arguments passed to the form	37
		sqlReceipt	String variable	SQL Update statement to update the customer details	37
		rsCustomer	Recordset Variable	Links the receipt report to the customers form	37

Tables Data Dictionary

Tables where used	Name	Type	Purpose
tblCustomer	CustomerID	Integer	Primary Key
	FirstName	String	Stores Customer’s first name
	Surname	String	Stores Customer’s Surname
	Title	String	Stores Customer’s Title
	DateOfBirth	Date	Stores Customer’s Date Of Birth

	TelephoneNumber	String	Stores Customer's Telephone Number
	Postcode	String	Stores Customer's Address (Postcode)
	City	String	Stores Customer's Address (City)
	Road	String	Stores Customer's Address (Road)
	ResidenceNumber	String	Stores Customer's Address (Residence Number)
	EMail	String	Stores Customer's email
	CardType	String	Stores Customer's Card Type
	CreditCardNumber	Integer	Stores Customer's Card Number
	ExpiryDate	Date	Stores Customer's Card Expiry Date
	SecurityCode	String	Stores Customer's Card Security Code
tblBookings	Bookings ID	Integer	Primary Key
	Customer ID	Integer	Foreign Key
	SeatID	Integer	foreign Key
	Method of payment	String	Stores the method of payment
	PerformanceID	Integer	foreign Key
	IsBooked	Boolean	Shows if the seat is booked
tblPerformance	PerformanceID	Integer	Primary Key
	PerformanceDate	Date	Stores the Performance Date
	PerformanceName	String	Stores the Performance name
tblSeats	SeatID	Integer	Primary Key
	Row	String	Row Reference
	Seats	Integer	Seat Reference
	Disabled	Boolean	Disabled Reference
	Price	Single	Seat Price
tblSecurity	UserID	Integer	Primary Key
	Username	String	Stores User's username
	Password	String	Stores User's Password
	Level	String	Stores User's access Level

2). Entity Relationship Diagram



CG2.3/1 – User Interface

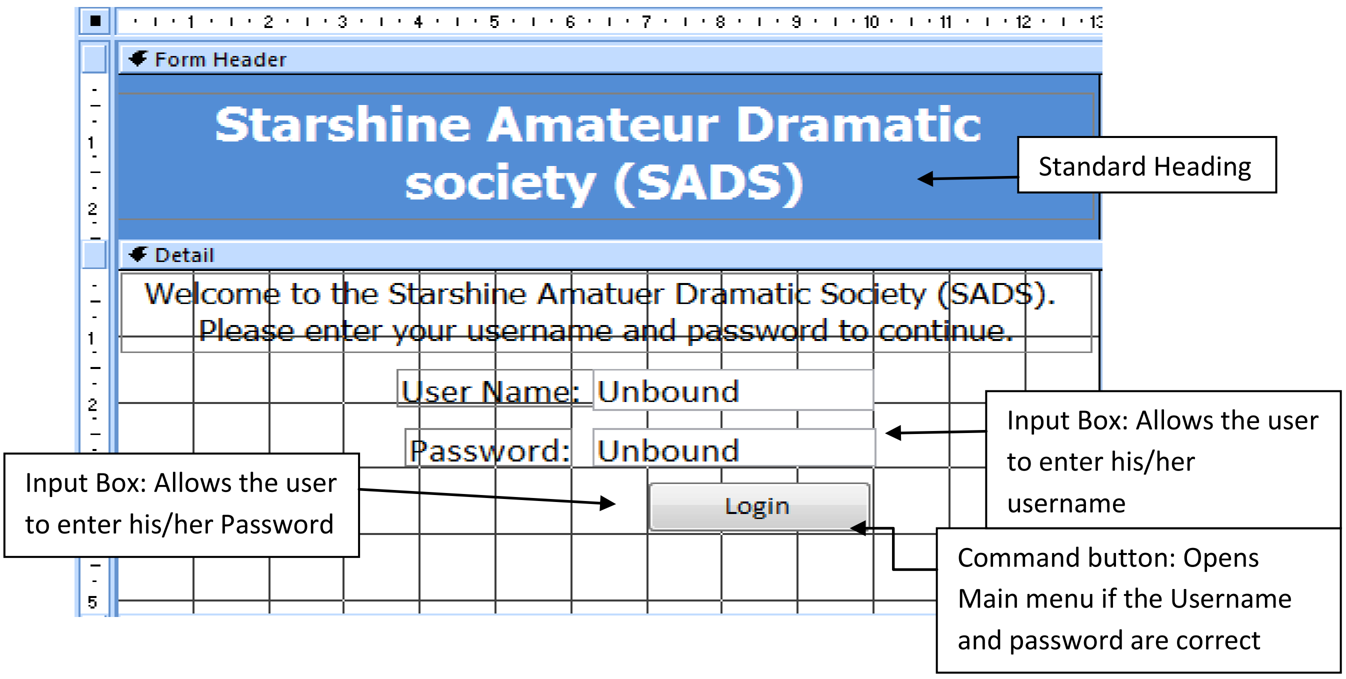
For screen shots of data entry forms and reports please see section CG2.2

I want to use data entry forms and reports because I do not want to give the user access to the raw tables. This is due to many factors; the users might be new and thus might not be trusted as much: It's very easy to change data to a raw table. My user interface is suitable for the user because it's very simple and easy to use; all the buttons are fairly big and easy to see.

All of my forms and reports have a consistent layout (you can look at my menus for the consistent layout). For my house style have I chosen a simple colour 'blue' for my header and footer and have written the form title on the header and the name of the society on the footer.

The user does not have to remember any special key stroke therefore making the database easier to use. The database runs quickly on most forms but there is a 2-4 second delay whilst opening the available seats form. After the login screen there is a main menu form which the user can navigate around but if the user clicks on the 'customer' button a menu appears asking the user what they want to do with a customer (add, update, delete). Apart from this instance there is no other menu apart from these 2 menus.

Login Form

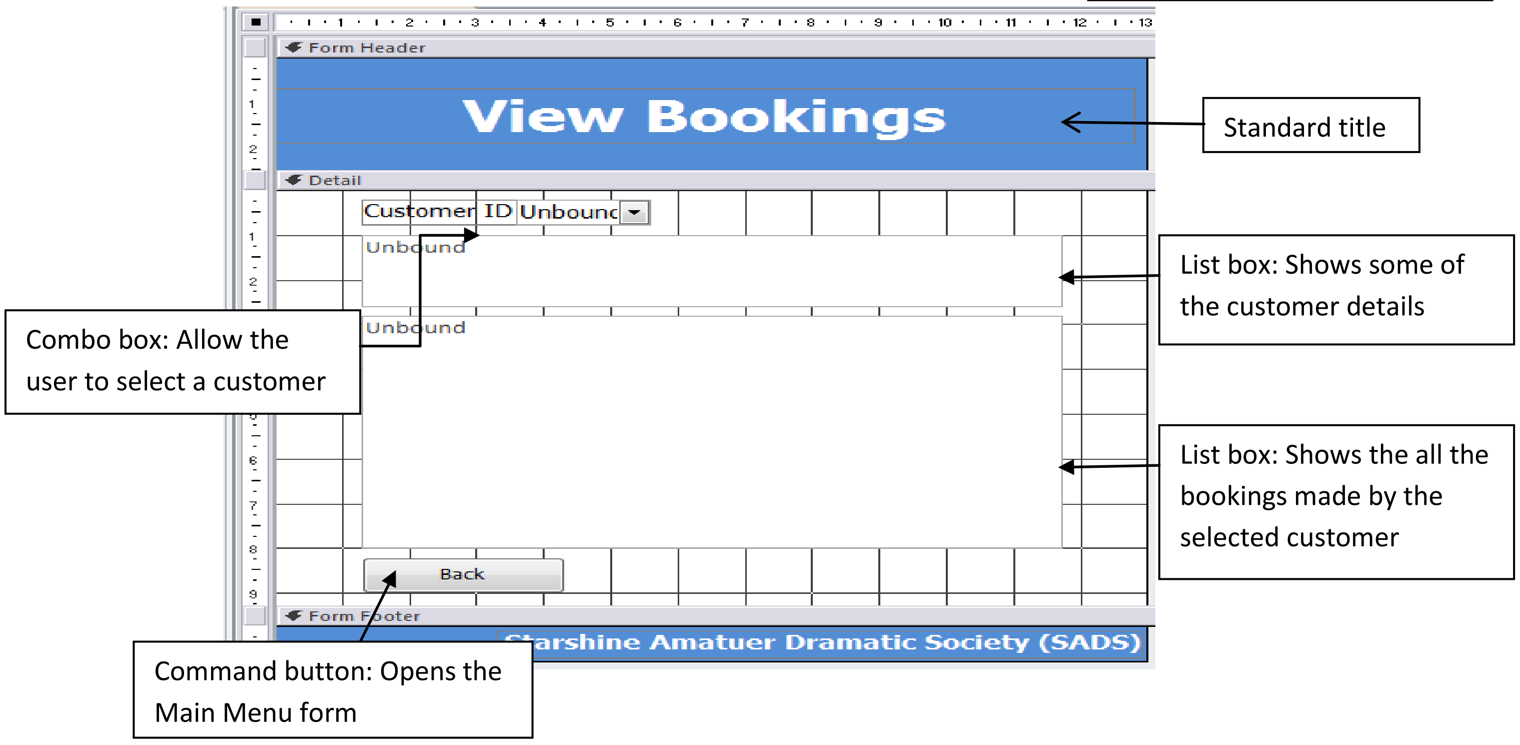

Form Header
Detail
User Name:
Password:
Login
Standard Heading
Input Box: Allows the user to enter his/her username
Input Box: Allows the user to enter his/her Password
Command button: Opens Main menu if the Username and password are correct

Starshine Amateur Dramatic society (SADS)

Welcome to the Starshine Amatuer Dramatic Society (SADS).
Please enter your username and password to continue.

User Name: Unbound

Password: Unbound


Form Header
Detail
Customer ID
Back
Form Footer
Standard title
List box: Shows some of the customer details
Combo box: Allow the user to select a customer
List box: Shows the all the bookings made by the selected customer
Command button: Opens the Main Menu form

View Bookings

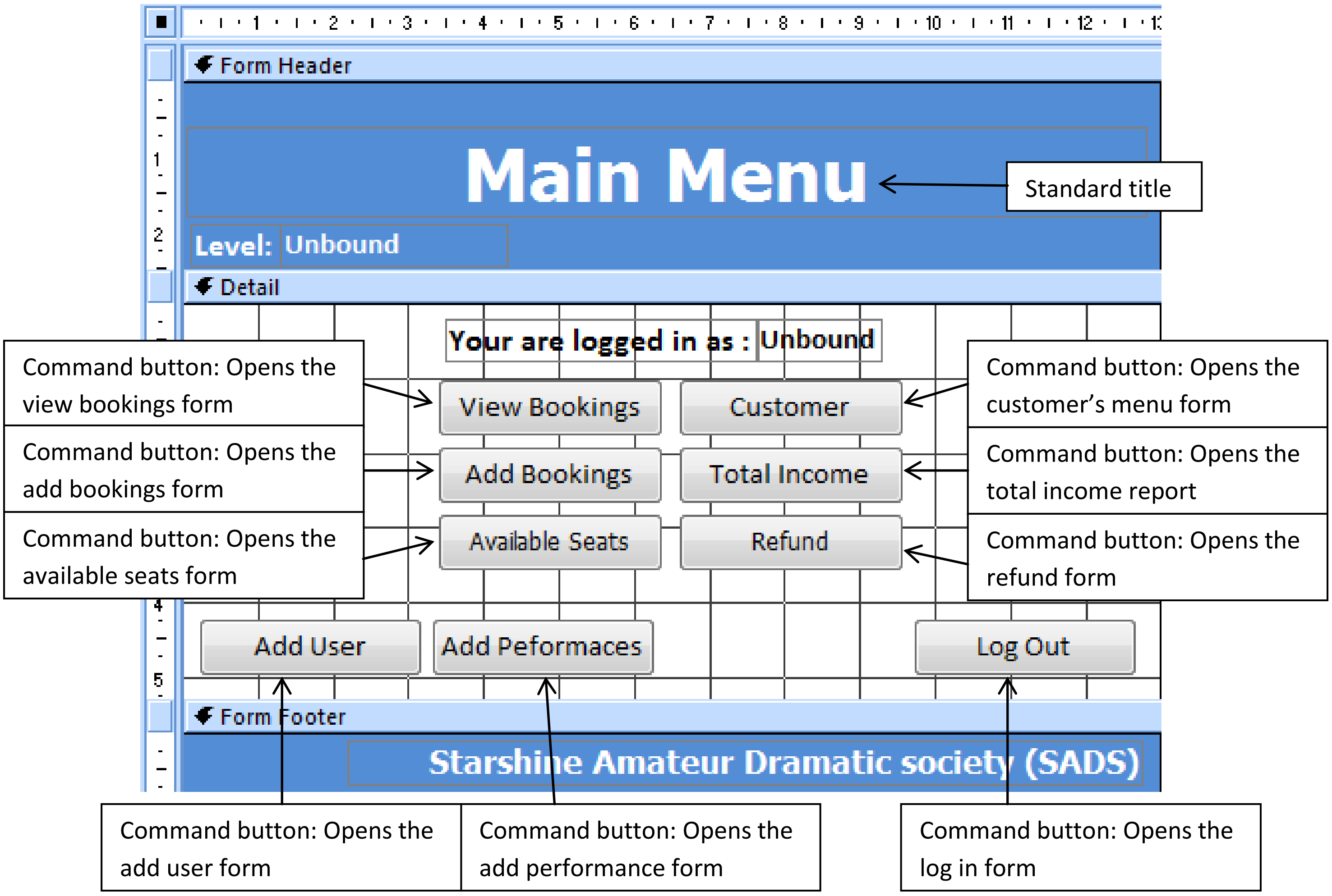
Customer ID Unbound

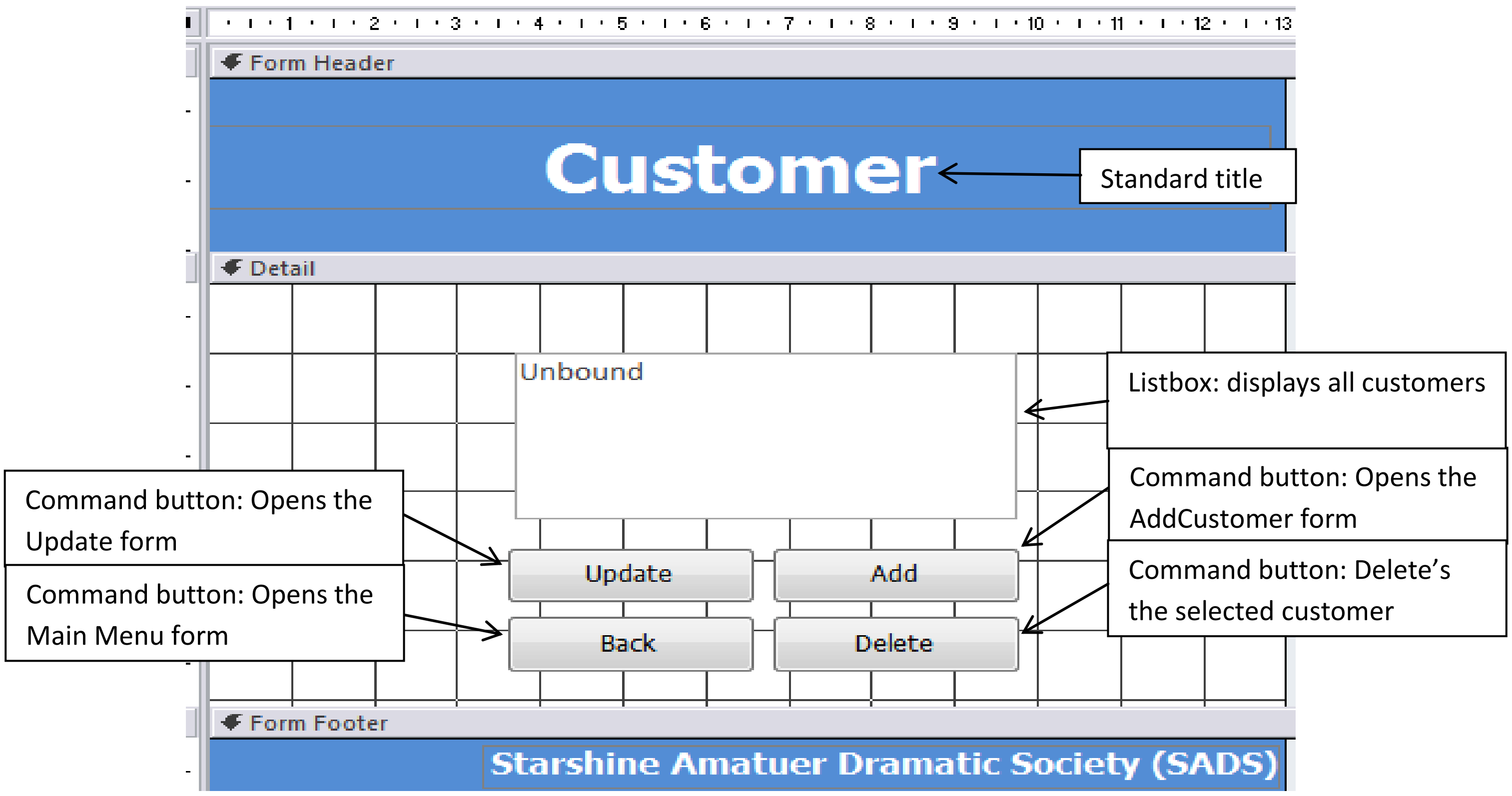
Unbound

Unbound

Starshine Amatuer Dramatic Society (SADS)

Main Menu


Form Header
Main Menu
Standard title
Level: Unbound
Detail
Your are logged in as : Unbound
View Bookings
Customer
Add Bookings
Total Income
Available Seats
Refund
Add User
Add Peformaces
Log Out
Form Footer
Starshine Amateur Dramatic society (SADS)
Command button: Opens the view bookings form
Command button: Opens the add bookings form
Command button: Opens the available seats form
Command button: Opens the customer's menu form
Command button: Opens the total income report
Command button: Opens the refund form
Command button: Opens the add user form
Command button: Opens the add performance form
Command button: Opens the log in form


Form Header
Customer
Standard title
Detail
Unbound
Listbox: displays all customers
Command button: Opens the AddCustomer form
Command button: Delete's the selected customer
Command button: Opens the Update form
Command button: Opens the Main Menu form
Update
Add
Back
Delete
Form Footer
Starshine Amatuer Dramatic Society (SADS)

CG2.3/1 – Annotated Listings

frmAddCustomer

Subroutine: cmdAdd_Click()
Description: This command is called when the "Add" button is clicked. It allows the user to add a new customer.
Objective’s met: Allow customer details to be input and stored

----
```
Private Sub cmdAdd_Click()
    'Declare a recordset
    Dim rsCustomer As Recordset

    'Set the recordset to access the cutsomer's table
    Set rsCustomer = CurrentDb.OpenRecordset("tblCustomer", dbOpenDynaset)

    'Set the recordset to add a new record to the customer's table
    rsCustomer.AddNew

    'Assign form data (input boxes)to relevant recorset fields
    rsCustomer!Title = Me.txtTitle
    rsCustomer!FirstName = Me.txtFirstName
    rsCustomer!Surname = Me.txtSurname
    rsCustomer!DateOfBirth = Me.txtDate
    rsCustomer!TelephoneNumber = Me.txtNumber
    rsCustomer!EMail = Me.txtEmail
    rsCustomer!ResidenceNumber = Me.txtRnumber
    rsCustomer!Road = Me.txtRoad
    rsCustomer!City = Me.txtCity
    rsCustomer!Postcode = Me.txtPostcode
    rsCustomer!CardType = Me.txtCtype
    rsCustomer!CreditCardNumber = Me.txtCnumber
    rsCustomer!ExpiryDate = Me.txtExpirydate
    rsCustomer!SecurityCode = Me.txtCode
    'Set the recordset to update the new cutsomer's details
    rsCustomer.Update

    'Close the recordset
    rsCustomer.Close

    MsgBox ("A new customer has been added!")

    DoCmd.Close
    DoCmd.OpenForm "frmMainMenu"
End Sub
```

---

frmAddPerformance

Subroutine: cmdAdd_Click()
Description: Allows the User to add a new Peformnace.
Objective's met: Allows multiple performances to be created

```
Private Sub cmdAdd_Click()
    Dim PerformanceID As Integer

    ' Call AddPerformance to add a new performance row into the database.
    Call CreateNewPerformance

    ' Get the PeformanceID just added.
    PerformanceID = GetPerformanceID()

    CreateNewBooking (PerformanceID)

    DoCmd.Close
    DoCmd.OpenForm "frmMainMenu"
End Sub
```

Subroutine: CreateNewPerformance
Description: Creates a new row to "tblPerformance" table.
Objective's met: Allows multiple performances to be created

```
Private Sub CreateNewPerformance()
    'Declare a recordset
    Dim rsPerformance As Recordset

    'Set the recordset to add a new record to the Performance table
    Set rsPerformance = CurrentDb.OpenRecordset("tblPerformance", dbOpenDynaset)

    'Assign form data (input boxes)to relevant recorset fields
    rsPerformance.AddNew
    rsPerformance!PerformanceDate = Me.txtPerformanceDate
    rsPerformance!PerformanceName = Me.txtPerformanceName

    'Update the new performance details
    rsPerformance.Update
    rsPerformance.Close
End Sub
```

Subroutine: GetPerformanceID
Description: Gets the PeformanceID just added. This PeformanceID is used to update the performanceID column in tblBookings.
Objective's met: Allows multiple performances to be created

```
Private Function GetPerformanceID() As Integer
    Dim sqlGetID As String
    Dim rsPerformance As Recordset

    ' SQL Select statement to get the maximum PerfancementID, i.e. Get PerformanceID just saved.
    sqlGetID = "SELECT MAX(PerformanceID) AS NextPerformanceID FROM [tblPerformance]"
    Set rsPerformance = CurrentDb.OpenRecordset(strSQL)

    ' Return NextPerformanceID
    GetPerformanceID = rsPerformance!NextPerformanceID
End Function
```

Subroutine: CreateNewBooking
Description: Adds rows to an existing column to " tblBookings " table.
Objective's met: Allows multiple performances to be created

```
Private Sub CreateNewBooking(ByVal PerformanceID)
    Dim sqlAdd As String

    ' Selects data from the bookings table
    sqlAdd = "INSERT INTO tblBookings ( SeatID, PerformanceID ) SELECT tblSeats.SeatID, " & PerformanceID & " FROM 
tblSeats;"

    ' Executes the SQL statement
    DoCmd.RunSQL sqlAdd

    MsgBox ("New performance has been booked!")
End Sub
```

----

frmAddUser

Subroutine: cmdAddUsers_Click()
Description: Adds a new column to the security table ("tblSecurity").
Objective's met: Allows Users to be created with access levels

----

```
Private Sub cmdAddUsers_Click()
    ' Declare Variable
    Dim rsUser As Recordset

    ' Connect the rsUser Recordset to the tblSale
    Set rsUser = CurrentDb.OpenRecordset("tblSecurity", dbOpenDynaset)

    ' Set the rsUser recordset into the mode to add a new record
    rsUser.AddNew

    ' Transfer the contents of all the objects on the form to the respective column in the recordset
    rsUser!Username = Me.Username
    rsUser!Password = Me.Password
    rsUser!Level = Me.Level

    ' If the 're-enter' password field is not equal to the 'password' field then clear the input boxes and display
      an error message
    If Me.Password <> Me.txtrepass Then
        MsgBox ("Passwords do not match!")
        Me.Password.Value = Null
        Me.txtrepass.Value = Null
    Else

       ' Update the new user details
        rsUser.Update

       ' close the recorset
        rsUser.Close

        DoCmd.Close
        DoCmd.OpenForm "frmMainMenu"
    End If
End Sub
```

Subroutine: cmdDeleteUsers_Click()
Description: Delete's a user.
Objective's met: Allows Users to be deleted

```
Private Sub cmdDeleteUsers_Click()
    Dim lngID As Long
    Dim SQLdeleteuser As String

    'If no record is selected then show a message and leave the subroutine.
    If IsNull(lstUser) Then
        MsgBox ("Please select a user")
        Exit Sub
    End If
    ' Get the selected record's ID value
    lngID = lstUser.Value

    ' Create a SQL statement
    SQLdeleteuser = "DELETE * FROM [tblSecurity] WHERE UserID = " & lngID

    ' Run it against the database
    CurrentDb.Execute strSQL

    ' Refresh the list control
    lstUser.Requery
End Sub
```

----

frmAvailableSeats

Subroutine: Form_Load
Description: Clears the contents of all the listboxes.
Objective's met: Display all available seats for a selected performance

----

```
Private Sub Form_Load()
    Dim sqlPeformance As String

    ' SQL statement selecting the performance ID from the performance table
    sqlPeformance = "SELECT PerformanceID FROM tblPerformance"

    ' Make the RowSource of 'cmbPerformance' equal the SQL statement just made
    cmbPerformance.RowSource = sqlPeformance

    ' Refresh the combo box control
    cmbPerformance.Requery

    ' Set the RowSource of all the listboxes to nothing
    lstSeats1.RowSource = ""
    lstSeats2.RowSource = ""
    lstSeats3.RowSource = ""
    lstSeats4.RowSource = ""
    lstSeats5.RowSource = ""
    lstSeats6.RowSource = ""
    lstSeats7.RowSource = ""
    lstSeats8.RowSource = ""
    lstSeats9.RowSource = ""
    lstSeats10.RowSource = ""
    lstSeats11.RowSource = ""
    lstSeats12.RowSource = ""
    lstSeats13.RowSource = ""
    lstSeats14.RowSource = ""
    lstSeats15.RowSource = ""
    lstSeats16.RowSource = ""
    lstSeats17.RowSource = ""
    lstSeats18.RowSource = ""
    lstSeats19.RowSource = ""
    lstSeats20.RowSource = ""
End Sub
```

Subroutine: cmbPerformance_Click()

Description: Store the selected PerformanceID. Set the row source of all the listboxes to display the available seats for the performance selected in cmbPerformance.

Objective's met: Display all available seats for a selected performance

----

```
Private Sub cmbPerformance_Click()
    Dim PerformanceID As Integer

    PerformanceID = cmbPerformance.Value

    ' Set the seat numbers for a given performance ID.
    lstSeats1.RowSource = GetAvailableSeats(1, PerformanceID)
    lstSeats1.Requery

    lstSeats2.RowSource = GetAvailableSeats(2, PerformanceID)
    lstSeats2.Requery

    lstSeats3.RowSource = GetAvailableSeats(3, PerformanceID)
    lstSeats3.Requery

    lstSeats4.RowSource = GetAvailableSeats(4, PerformanceID)
    lstSeats4.Requery

    lstSeats5.RowSource = GetAvailableSeats(5, PerformanceID)
    lstSeats5.Requery

    lstSeats6.RowSource = GetAvailableSeats(6, PerformanceID)
    lstSeats6.Requery

    lstSeats7.RowSource = GetAvailableSeats(7, PerformanceID)
    lstSeats7.Requery

    lstSeats8.RowSource = GetAvailableSeats(8, PerformanceID)
```

```
lstSeats8.Requery

lstSeats9.RowSource = GetAvailableSeats(9, PerformanceID)
lstSeats9.Requery

lstSeats10.RowSource = GetAvailableSeats(10, PerformanceID)
lstSeats10.Requery

lstSeats11.RowSource = GetAvailableSeats(11, PerformanceID)
lstSeats11.Requery

lstSeats12.RowSource = GetAvailableSeats(12, PerformanceID)
lstSeats12.Requery

lstSeats13.RowSource = GetAvailableSeats(13, PerformanceID)
lstSeats13.Requery

lstSeats14.RowSource = GetAvailableSeats(14, PerformanceID)
lstSeats14.Requery

lstSeats15.RowSource = GetAvailableSeats(15, PerformanceID)
lstSeats15.Requery

lstSeats16.RowSource = GetAvailableSeats(16, PerformanceID)
lstSeats16.Requery

lstSeats17.RowSource = GetAvailableSeats(17, PerformanceID)
lstSeats17.Requery

lstSeats18.RowSource = GetAvailableSeats(18, PerformanceID)
lstSeats18.Requery

lstSeats19.RowSource = GetAvailableSeats(19, PerformanceID)
lstSeats19.Requery
```

```
    lstSeats20.RowSource = GetAvailableSeats(20, PerformanceID)
    lstSeats20.Requery
End Sub
```

Subroutine: GetAvailableSeats
Description:
Objective's met: Display all available seats for a selected performance

----

```
Private Function GetAvailableSeats(ByVal seatNumber As Integer, ByVal PerformanceID As Integer)

    Dim sqlSeats As String

    ' SQL Select statement returning values from the booking and seat table's

    sqlSeats = "SELECT tblBookings.SeatID, tblSeats.Row , tblSeats.Seats AS Seats, tblBookings.IsBooked AS IsBooked,
iif(tblBookings.IsBooked = 0, 'Yes', 'No') As " & seatNumber & " FROM tblBookings INNER JOIN tblSeats ON
tblBookings.SeatID = tblSeats.SeatID WHERE tblSeats.Seats = " & seatNumber & " AND PerformanceID = " & PerformanceID


    GetAvailableSeats = sqlSeats

End Function
```

----

frmBookings

Subroutine: Form_Load
Description: Query for a combo box to select the customerID
Objective's met: Allows the user to add multiple bookings for a customer, Enable seats to be booked for the correct day

----

```
Private Sub Form_Load()
    Dim sqlCustomer As String

    ' SQL Select statement to return customer id, title and name from the customer table.
    sqlCustomer = "SELECT CustomerID, Title + ' ' + FirstName + ' ' + Surname AS FullName FROM tblCustomer"

    txtCustomerID.RowSource = sqlCustomer
    txtCustomerID.Requery

    ' Clear all seats.
    lstBookingSeats.RowSource = ""

    txtPerformanceID.AllowValueListEdits = True
    cmbRow.AllowValueListEdits = True
End Sub
```

Subroutine: cmdAddBookings_Click()
Description: Initialise the form
Objective's met: Allows the user to add multiple bookings for a customer, Enable seats to be booked for the correct day
----

```
Private Sub cmdAddBookings_Click()
    Dim Oitem As Variant
    Dim bookingID As String
    Dim selectedBookingIDs As String
    Dim count As Integer
    Dim PerformanceID As String
    Dim CustomerID As Integer
    Dim totalItemsSelected As Integer
    Dim whereStr As String
    Dim totalbookingPrice As Double
```

```vb
Dim isSeatDisabled As Boolean

' Store the selected performance ID
PerformanceID = txtPerformanceID.Value

' Initialise the string which is later used to store the selected booking IDs separated by comma.
selectedBookingIDs = ""

' Get the total number of seats booked.
totalItemsSelected = lstBookingSeats.ItemsSelected.count

' If there are seats selected then
If totalItemsSelected <> 0 Then
    ' Initialise the counter. This counter is used to check if it is the last seat. if it is the last seat then
      there is no need to append the comma (,) because it is not needed in the WHERE statement.
    count = 0

    ' Loop through the seats to be booked and update the status in the database table.
    For Each Oitem In lstBookingSeats.ItemsSelected
        ' Convert from string to Integer because the items in the listbox are stored as strings.
        bookingID = CInt(lstBookingSeats.Column(0, Oitem))
        count = count + 1 ' increment the counter

        ' Build a string of selected booking IDs. This string is used in the SQL WHERE statement.
        selectedBookingIDs = selectedBookingIDs & bookingID

        ' If it is not the last selected item then append a comma to the string.
        If (count < totalItemsSelected) Then
            selectedBookingIDs = selectedBookingIDs & ","
        End If

        ' Convert from string to Boolean because the items in the
        ' listbox are stored as strings.
        isSeatDisabled = CBool(lstBookingSeats.Column(5, Oitem))
```

```
            ' If there is a disabled seat selected then
            If (isSeatDisabled = True) Then
                ' Display a message box asking the user whether they want to coninue
                If MsgBox("Warning,You have selected a diabled seat. Do you want to continue?", vbYesNo) =vbYes Then
                    Call BookSeat(bookingID) ' Book a seat.
                Else
                    Exit Sub  'Do not book any seat.
                End If
            Else
                Call BookSeat(bookingID) ' Book a seat.
            End If
Next Oitem

        ' Get the total number of seats booked and display in the message box.
        totalbookingPrice = GetTotalBookingPrice()

        MsgBox ("Total Booking Price: £" & totalbookingPrice)

        ' Build the WHERE statement which is passed to Receipt report as an argument.
        whereStr = "((tblBookings.[Bookings ID] In (" & selectedBookingIDs & ")) AND (tblBookings.IsBooked=True))"

        ' The customer ID is passed to the report because we only want to view the report of a single customer.
        CustomerID = txtCustomerID.Value

        DoCmd.Close

        ' Generates and opens the report in print preview. The 'WhereCondition' acts as a filter to report.
        DoCmd.OpenReport ReportName:="rptReceipt", View:=acViewPreview, WhereCondition:=whereStr,
        OpenArgs:=CustomerID
    Else
        MsgBox "Nothing was selected from the list", vbInformation
        Exit Sub  'Nothing was selected
    End If
End Sub
-----------------------------------------------------------------------------------------------------------------------
```

Subroutine: UpdateBooking
Description: Finds the row in tblBookings table given the bookingID and sets IsBooked to true to indicate that a seat has been booked
Objective's met: Allows the user to add multiple bookings for a customer, Enable seats to be booked for the correct day

```
-----------------------------------------------------------------------------------------------------------------------

Private Sub UpdateBooking(ByVal bookingID)
    ' declare a recordset
    Dim rsUpdateBookings As Recordset

    ' Set the recordset to access the bookings table
    Set rsUpdateBookings = CurrentDb.OpenRecordset("tblBookings", dbOpenDynaset)

    rsUpdateBookings.FindFirst "[Bookings ID] = " & bookingID

    ' edit the recorset
    rsUpdateBookings.Edit

    ' tranfer the contents of the input boxes to the corresponding fields in the recordset
    rsUpdateBookings![Method of payment] = Me.txtMethodofpayment
    rsUpdateBookings![Customer ID] = Me.txtCustomerID
    rsUpdateBookings!IsBooked = True ' indicate a seat has been booked.

    ' Update the recorset
    rsUpdateBookings.Update

    ' close the recordset
    rsUpdateBookings.Close
End Sub

-----------------------------------------------------------------------------------------------------------------------
```

Subroutine: ShowAvailableSeats
Description: Shows available seat in BookingSeats listbox. This method is called only when PerformanceID or Row comboboxes is selected.
Objective's met: Allows the user to add multiple bookings for a customer, Enable seats to be booked for the correct day. Ensure that seats are not double booked

```
-----------------------------------------------------------------------------------------------------------------

Private Sub ShowAvailableSeats()
    Dim PerformanceID As Integer
    Dim row As String
    Dim sqlAvailable As String

    ' If the performance ID and row are both not null.
    If (Not IsNull(txtPerformanceID.Value) And Not IsNull(cmbRow.Value)) Then
        ' Get the performance ID.
        PerformanceID = txtPerformanceID.Value

        ' Get the row.
        row = cmbRow.Value

        ' Build the SQL string for available seats. This statement also check if the seat is not already booked
          because we don't want to book the same seat twice or over-book it. i.e. Prevents double bookings
        sqlAvailable = "SELECT tblBookings.[Bookings ID], tblBookings.SeatID, tblSeats.Row & ' ' & tblSeats.Seats AS
Seats, tblBookings.PerformanceID, tblBookings.IsBooked,tblSeats.Disabled, tblSeats.Price FROM tblBookings INNER JOIN
tblSeats ON tblBookings.SeatID=tblSeats.SeatID WHERE tblBookings.PerformanceID=" & PerformanceID & " AND
tblSeats.Row='" & row & "' AND tblBookings.IsBooked=False"

        ' Initialise the BookingSeat listbox.
        lstBookingSeats.ColumnHeads = True
        lstBookingSeats.ColumnCount = 7
        lstBookingSeats.RowSourceType = "Table/Query"

        ' provide the data to BookingSeats listbox.
        lstBookingSeats.RowSource = sqlAvailable
        lstBookingSeats.Requery
    Else
        ' Clears the BookingSeats listbox.
        lstBookingSeats.RowSource = ""
    End If
End Sub
```

Subroutine: GetTotalBooking

Description: Gets the total price of items selected in BookingSeats listbox, i.e. calculates the total price of seats being booked.

Objective's met: Allows the user to add multiple bookings for a customer, Enable seats to be booked for the correct day

----

```
Private Function GetTotalBookingPrice()
    Dim Oitem As Variant
    Dim Price As Double
    Dim totalbookingPrice As Double

    ' Example of Error handling in VBA ( I used this to find the error in the code because I could not find the
      error until I used this method)
    On Error GoTo err_handler

    ' Initialise the total booking. We want to calculate the total of seat booked.
    totalbookingPrice = 0

    ' Loop through the seats you want to book and calculate the total price.
    For Each Oitem In lstBookingSeats.ItemsSelected
        Price = CDbl(lstBookingSeats.Column(6, Oitem))
        totalbookingPrice = totalbookingPrice + Price
    Next Oitem

    GetTotalBookingPrice = totalbookingPrice
```

```
endit:
   Exit Function

' An error has occurred
err_handler:
   MsgBox "Error: " & Err.Description & Chr(13) & "Function: GetTotalBookingPrice()"
   Resume endit
End Function
```

----

frmCustomer

Subroutine: Form_Load

Description: Displays some of the customer details in a listbox

Objective's met: Allows the user to add, delete or update customers

----

```
Private Sub Form_Load()
    Dim sqlCustomer As String
    ' SQL statement which selects the customerID, first name and last name from the customer's table
    sqlCustomer = "SELECT CustomerID, FirstName + ' ' + Surname AS FullName FROM tblCustomer"

    ' Populate a list box
    lstCustomer.ColumnHeads = False
    lstCustomer.ColumnCount = 2
    lstCustomer.RowSourceType = "Table/Query"
    lstCustomer.RowSource = strSQL
    lstCustomer.Requery
End Sub
```

Subroutine: cmdDelete_Click()

Description: Deletes the selected customer
Objective's met: Allows the user to delete customers

```
Private Sub cmdDelete_Click()
    Dim lngID As Long
    Dim sqlDelete As String

    'If no record is selected then show a message and leave the subroutine.
    If IsNull(lstCustomer) Then
        MsgBox ("Please select a customer")
        Exit Sub
    End If

    ' Get the selected record's ID value
    lngID = lstCustomer.Value

    ' Create a SQL statement
    sqlDelete = "DELETE * FROM [tblCustomer] WHERE CustomerID = " & lngID

    ' Run it against the database
    CurrentDb.Execute strSQL

    'Refresh the list control
    lstCustomer.Requery
End Sub
```

Subroutine: cmdUpdate_Click()
Description: Update the selected customer
Objective's met: Allows the user to update customer details

```
Private Sub cmdUpdate_Click()
    Dim CustomerID As String

    ' If a customer is selected show the Update form.
    If lstCustomer.Value <> 0 Then
        CustomerID = lstCustomer.Value
        DoCmd.Close

        ' Open the Update form and pass the CustomerID as an argument.
        DoCmd.OpenForm "frmUpdate", OpenArgs:=CustomerID
    Else
        MsgBox ("Please select a Customer")
    End If
End Sub
```

---

frmLogin

Subroutine: cmdLogin_Click()
Description: Update the selected customer
Objective's met: Allows the user to update customer details

```
Private Sub cmdLogin_Click()
    ' Declare a recordset
    Dim rsUser As Recordset

    ' Set the recordset to access the security table
    Set rsUser = CurrentDb.OpenRecordset("tblSecurity", dbOpenDynaset)

    ' Check to see if the entered values are equal to the values in the table
    Do While Not rsUser.EOF
        If Me.txtUserName = rsUser!Username Then
            If Me.txtPassword = rsUser!Password Then
```

```
                pubLevel = rsUser!Level ' Store the entered values in a global variable
                pubUsername = rsUser!Username

                DoCmd.Close
                DoCmd.OpenForm "frmMainMenu"
            Else
                MsgBox "Incorrect Password", , "Wrong Password"
            End If

            Exit Sub
        End If

        ' Move to the next record in the recordset
        rsUser.MoveNext
    Loop

    MsgBox "Username or password not found!", , "User not found"
End Sub
```
----

frmMainMenu

Subroutine: Form_Load
Description: Displays the user's access level and useraname
Objective's met: NA

```
Private Sub Form_Load()
    ' Declare a recordset
    Dim rsLevel As Recordset

    Me.txtStatus = pubLevel
    Me.txtUser = pubUsername
    ' Set the recordset to access the security table
    Set rsLevel = CurrentDb.OpenRecordset("tblSecurity", dbOpenDynaset)

    ' if the user is a admin the enable all buttons else disable cmdAddPerformances and cmdAddUser
    If publevel = "Admin" Then
        cmdAddPerformances.Enabled = True
        cmdAddUser.Enabled = True
    End If
End Sub
```

Subroutine: cmdLogOut_Click()
Description: Allows the user to log out of the system
Objective's met: NA

```
Private Sub cmdLogOut_Click()
    Dim answer As Integer

    ' Check to see if the use really want to log out
    answer = MsgBox(" Are you sure ", vbYesNo)

    If answer = vbYes Then
      DoCmd.Close
      DoCmd.OpenForm "frmLogin"
    End If
End Sub
```
----

frmRefund

Subroutine: Form_Load
Description: Displays the name of the customer in a comb box
Objective's met: Allows the user to refund bookings

```
Private Sub Form_Load()
    Dim sqlCustomerID As String

    ' Create an SQL statement which selects the CustomerID, FirstName and Surname from the customer’s table
    sqlCustomerID = "SELECT CustomerID, FirstName + ' ' + Surname AS FullName FROM tblCustomer"

    ' Set the row source of cmbCustomer to the SQL statement
    cmbCustomer.RowSource = sqlCustomerID
    cmbCustomer.Requery

    ' Clear the Booking listbox.
    lstBookings.RowSource = ""
End Sub
```

Subroutine: cmbCustomer_Click
Description: populates the listbox with the booking details of a customer
Objective’s met: Allows the user to refund bookings

```
Private Sub cmbCustomer_Click()
    Dim CustomerID As Integer
    Dim sqlBookingsdetail As String

    ' Get the customerID and store it in 'CustomerID'
    CustomerID = cmbCustomer.Value

    ' if there is a customer selected then create an SQL statement which return the bookings information for the
    ' customer selected
    If Not IsNull(CustomerID) Then
        sqlBookingsdetail = "SELECT tblBookings.SeatID, tblSeats.Row &' '& tblSeats.Seats AS Seats,
tblBookings.[Bookings ID], tblBookings.[Method of payment], tblBookings.PerformanceID, tblSeats.Price FROM
tblBookings INNER JOIN tblSeats ON tblBookings.SeatID = tblSeats.SeatID WHERE [Customer ID] IS NOT NULL AND
[Customer ID] = " & CustomerID
```

```
        ' Display the booking deatils in a listbox
        lstBookings.RowSource = sqlBookingsdetail
        lstBookings.Requery
    End If
End Sub
```

Subroutine: cmdRefund_Click
Description: Allows the user to refund bookings
Objective's met: Allows the user to refund bookings

```
Private Sub cmdRefund_Click()
    Dim count As Integer
    Dim CustomerID As String
    Dim Oitem As Variant
    Dim seatID As String
    Dim totalRefund As Double

    ' Set the count to 0
    count = 0

    ' Store the selected customerID
    CustomerID = cmbCustomer.Value

    ' Set the total refund to 0
    totalRefund = 0

    ' if there is a bookings selected then
    If lstBookings.ItemsSelected.count <> 0 Then

        ' store the SeatID for each of the bookings that are selected
        For Each Oitem In lstBookings.ItemsSelected
```

```
            seatID = lstBookings.ItemData(Oitem)

            ' Add 1 to the count
            count = count + 1

            Call UpdateBooking(CustomerID, seatID)
        Next Oitem

        totalRefund = GetTotalRefund()
        MsgBox ("Total Refund: £" & totalRefund)

        DoCmd.Close acForm, Me.name
        DoCmd.OpenForm "frmMainMenu"
    Else
        MsgBox "Nothing was selected from the list", vbInformation
        Exit Sub  'Nothing was selected
    End If
End Sub
```

Subroutine: GetTotalRefund
Description: Calculates the total price of the refund
Objective's met: Allows the user to refund bookings

```
Private Function GetTotalRefund()
    Dim Oitem As Variant
    Dim Price As Double
    Dim totalRefund As Double

    'Set total refund to 0
    totalRefund = 0

    ' If a booking is selected then store the price of the booking in 'Price'
    If lstBookings.ItemsSelected.count <> 0 Then
        For Each Oitem In lstBookings.ItemsSelected
            Price = CDbl(lstBookings.Column(5, Oitem))
            totalRefund = totalRefund + Price
        Next Oitem

        GetTotalRefund = totalRefund
    Else
        MsgBox "Nothing was selected from the list", vbInformation
        GetTotalRefund = 0
    End If
End Function
```

Subroutine: UpdateBooking
Description: Removes the booking information of the selected booking in the bookings table
Objective's met: Allows the user to refund bookings

```
Private Sub UpdateBooking(ByVal CustomerID, ByVal seatID)
    ' Declare a recordset
    Dim rsUpdateBookings As Recordset

    ' Set the recordset to access the bookings table
    Set rsUpdateBookings = CurrentDb.OpenRecordset("tblBookings", dbOpenDynaset)

    ' Find the first record in which the customerID is equal to the selected customerID and the seatID is equal to
    ' the selected SeatID
    rsUpdateBookings.FindFirst "[Customer ID] = " & CustomerID & " AND SeatID = " & seatID

    ' Edit the recordset
    rsUpdateBookings.Edit

    ' Clear the fields mentioned below
    rsUpdateBookings![Method of payment] = Null
    rsUpdateBookings![Customer ID] = Null
    rsUpdateBookings!IsBooked = False

    ' Update the recordset
    rsUpdateBookings.Update

    ' close the recordset
    rsUpdateBookings.Close
End Sub
```

----

frmUpdate

Subroutine: Form_Load
Description: Initilises the "Update" form. Selects the customer information from the "tblCustomer" table and display on the form which can be updated.
Objective's met: Allows the user to update a customer

```
Private Sub Form_Load()
    Dim Customer_ID As String
    Dim Pstring As Variant
    Dim sqlID As String

    ' Declare a recordset
    Dim rsCustomer As Recordset

    ' A customer is selected in the "frmCustomer" form. The Customer_ID selected is passed
    ' from frmCustomer to this form as an argument. The arguments are passed as a line of string
    ' and are separated by "|". Hence, the String Split function is used to convert the line of
    ' string into the string of array or Variant. The Customer_ID is the first argument stored in
    ' the 1st position or index 0.
    If Len(Me.OpenArgs) > 0 Then ' Check if there are any arguments
       Pstring = Split(Me.OpenArgs, "|")
```

```
        Customer_ID = Pstring(0)
    End If

    ' Set the SQL statemet to get all the customer information based on a selected customer ID
    sqlID = "SELECT * FROM tblCustomer WHERE CustomerID = " & Customer_ID

    ' Set the recordset to access the SQL statement
    Set rsCustomer = CurrentDb.OpenRecordset(sqlID, dbOpenDynaset)

    ' Display all the selected customer information
    Me.txtTitle = rsCustomer!Title
    Me.txtFirstName = rsCustomer!FirstName
    Me.txtSurname = rsCustomer!Surname
    Me.txtDate = rsCustomer!DateOfBirth
    Me.txtNumber = rsCustomer!TelephoneNumber
    Me.txtEmail = rsCustomer!EMail
    Me.txtRnumber = rsCustomer!ResidenceNumber
    Me.txtRoad = rsCustomer!Road
    Me.txtCity = rsCustomer!City
    Me.txtPostcode = rsCustomer!Postcode
    Me.txtCtype = rsCustomer!CardType
    Me.txtCnumber = rsCustomer!CreditCardNumber
    Me.txtExpirydate = rsCustomer!ExpiryDate
    Me.txtCode = rsCustomer!SecurityCode
End Sub
```

Subroutine: cmdUpdateCustomer_Click
Description: Updates the customer information.
Objective's met: Allows the user to update a customer

```
Private Sub cmdUpdateCustomer_Click()
```

```
Dim Customer_ID As String
Dim Pstring As Variant
Dim sqlUpdate As String

' Parse the CustomerID argument.
If Len(Me.OpenArgs) > 0 Then
   Pstring = Split(Me.OpenArgs, "|")
   Customer_ID = Pstring(0)
End Sub

' We can use either a VBA RecordSet Edit() method or use SQL UPDATE statement to update row(s) in the table
sqlUpdate = "UPDATE tblCustomer " & _
            "SET Title = '" & Me.txtTitle & "'," & _
            "FirstName = '" & Me.txtFirstName & "'," & _
            "Surname = '" & Me.txtSurname & "'," & _
            "DateOfBirth = '" & Me.txtDate & "'," & _
            "TelephoneNumber = '" & Me.txtNumber & "'," & _
            "EMail = '" & Me.txtEmail & "'," & _
            "ResidenceNumber = '" & Me.txtRnumber & "'," & _
            "Road = '" & Me.txtRoad & "'," & _
            "City = '" & Me.txtCity & "'," & _
            "Postcode = '" & Me.txtPostcode & "'," & _
            "CardType = '" & Me.txtCtype & "'," & _
            "CreditCardNumber = '" & Me.txtCnumber & "'," & _
            "ExpiryDate = '" & Me.txtExpirydate & "'," & _
            "SecurityCode = '" & Me.txtCode & "' " & _
            "WHERE CustomerID = " & Customer_ID

DoCmd.RunSQL sqlUpdate

MsgBox ("Customer has been updated!")

DoCmd.Close
DoCmd.OpenForm "frmCustomer"
```

```
End Sub
```

----

frmViewBookings

Subroutine: Form_Load
Description: Initialise the form. Fill the Customer combobox and clear the "Customer" and "Bookings" listboxes.
Objective's met: Allows the user to browse bookings made by a specific customer.

```
Private Sub Form_Load()
    Dim sqlCustomerID As String

    ' Select all CustomerID from the tblCustomer table.
    sqlCustomerID = "SELECT CustomerID FROM tblCustomer"

    ' Load CustomerID into "Customer" combobox from tblCustomer.
    cmbCustomer.RowSource = sqlCustomerID
    cmbCustomer.Requery

    ' Clear the "Customer" list.
    lstCustomer.RowSource = ""
    lstCustomer.Requery

    ' Clear the "Bookings" list.
    lstBookings.RowSource = ""
    lstBookings.Requery
End Sub
```

Subroutine: cmbCustomer_Click
Description: Initialise the form. Fill the Customer combobox and clear the "Customer" and "Bookings" listboxes.
Objective's met: Allows the user to browse bookings made by a specific customer.

```
Private Sub cmbCustomer_Click()
    Dim CustomerID As Integer
    Dim sqlCustomerDetail As String
    Dim sqlBookingsdetail As String

    CustomerID = cmbCustomer.Value

    ' Select info from the tblCustomer table.
    sqlCustomerDetail = "SELECT Title, FirstName AS [First Name], Surname, TelephoneNumber AS [Telephone Number], 
Postcode FROM tblCustomer WHERE CustomerID = " & CustomerID

    ' Select info from the tblBookings table.
    sqlBookingsdetail = "SELECT tblBookings.SeatID, tblSeats.Row &' '& tblSeats.Seats AS Seats, 
tblBookings.[Bookings ID], tblBookings.[Method of payment], tblBookings.PerformanceID FROM tblBookings INNER JOIN 
tblSeats ON tblBookings.SeatID = tblSeats.SeatID WHERE [Customer ID] IS NOT NULL AND [Customer ID] = " & CustomerID

    If Not IsNull(CustomerID) Then
        ' Load data into "Customer" listbox from tblCustomer.
        lstCustomer.RowSource = sqlCustomerDetail
        lstCustomer.Requery

        ' Load data into "Bookings" listbox from the "tblbooking" table.
        lstBookings.RowSource = sqlBookingsdetail
        lstBookings.Requery
    End If
End Sub
```
----

rptReceipt

Subroutine: Report_Load

Description: This report has be stored with all the booked seats but it is being filtered to only show the seats just being booked form the bookings form

Objective's met: This enables seats to be booked for the correct day because if the seats aren't booked for the correct the day the customer can ask to get a refund and change the day he wants to go on. This also provides booking information

---

```
Private Sub Report_Load()
  Dim Customer_ID As String
  Dim Pstring As Variant
  Dim sqlReceipt As String

  ' delcare a recordset
  Dim rsCustomer As Recordset

  ' A customer is selected in the "frmCustomer" form. The Customer_ID selected is passed from frmCustomer to this form as an argument. The argruments are passed as a line of string and are separated by "|". Hence, the String Split function is used to convert the line of string into the string of array or Variant. The Customer_ID is the first argument stored in the 1st position or index 0.
  If Len(Me.OpenArgs) > 0 Then
    Pstring = Split(Me.OpenArgs, "|")
    Customer_ID = Pstring(0)
  End If

  ' Set the SQL Select statemet to get all the customer information based on a selected customer ID
  sqlReceipt  = "SELECT * FROM tblCustomer WHERE CustomerID = " & Customer_ID

  ' Set the recordset to access the SQL statement
  Set rsCustomer = CurrentDb.OpenRecordset(sqlReceipt , dbOpenDynaset)

  'display the data in text boxes
  Me.txtFullName = rsCustomer!Title & ". " & rsCustomer!FirstName & " " & rsCustomer!Surname
  Me.txtRoad = rsCustomer!ResidenceNumber & " " & rsCustomer!Road
```

```
    Me.txtCity = rsCustomer!City
    Me.txtPosctcode = rsCustomer!Postcode
End Sub
```