1. INTRODUCTION

The Airline Reservations System (ARS) was one of the earliest changes to improve efficiency. ARS eventually evolved into the Computer Reservations System (CRS), and then into Global Distribution System (GDS). The airline industry created the first GDS in the 1960s as a way to keep track of flight schedules, availability, and prices. Although accused of being "dinosaurs" due to their use of legacy system technology, GDSs were actually among the first e-commerce companies in the world facilitating B-2-B electronic commerce as early as the mid 1970s, when SABRE (owned by American Airline) and Apollo (United) began installing their propriety internal reservations systems in travel agencies. Prior to this, travel agents spent an inordinate amount of time manually entering reservations. The airlines realized that by automating the reservation process for travel agents, they could make the travel agents more productive and essentially turn into an extension of the airline's sales force. It is these original, legacy GDSs that today provide the backbone to the Internet travel distribution system.

There are currently four major GDS systems:

1. Amadeus
2. Sabre
3. Galileo
4. Worldspan

PROBLEM DEFINITION

In 21st century the world has become a global village where every thing is available in a single click of mouse button. Aviation sector is one of fastest mode of travel available with us, both at domestic and international level. To maintain such a large system is a hectic job. The present system is very time consuming and inefficient.

The definition of our problem lies in manual system and **a** fully automated system.

Manual system : The system is more prone to errors and some times it encounters various problems which are unstructured.

Technical system : With the advent of latest technology if we do not update our system then our business will suffer massive losses financially. The technical system (we have proposed) contains the tools of latest trend i.e. computers printers, fax etc. The systems with this technology are very fast, accurate, user-friendly and reliable.

Need of Airlines system

1. A few factors that direct us to develop a new system are given below -:

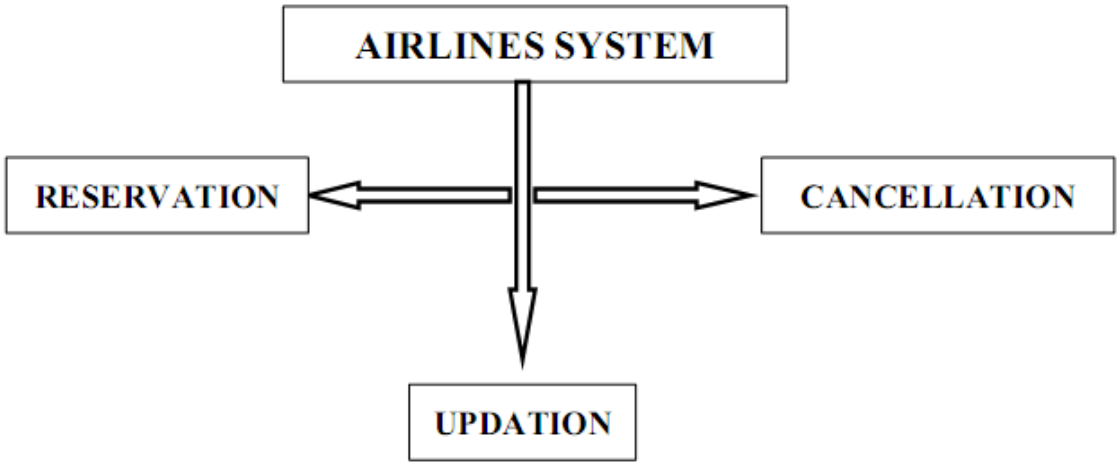
   1) Faster System

   2) Accuracy

   3) Reliability

   4) Informative

   5) Reservations.



2. OBJECTIVES

- To develop a system to management of airlines, this will perform all the functions with a click of mouse button's
- To develop a system that has good management of data along with integrity and minimizing redundancy.
- To develop a system that will be user friendly in all possible ways.
- To provide better customer support for passengers.

3. TOOLS & PLATFORM USED

We have a wide range of options of languages. From these options we can choose appropriate platform/ tools and languages for development of the project. Some of these are as follows:-

Programming Languages:- In programming language we have C, C++, C#, Microsoft Access, Microsoft Visual Basic, and Oracle PL/SQL etc.

Relational Database: - Oracle, IBM DB2, SQL Server, MS Access and FoxPro etc.

SOFTWARE REQUIREMENTS:

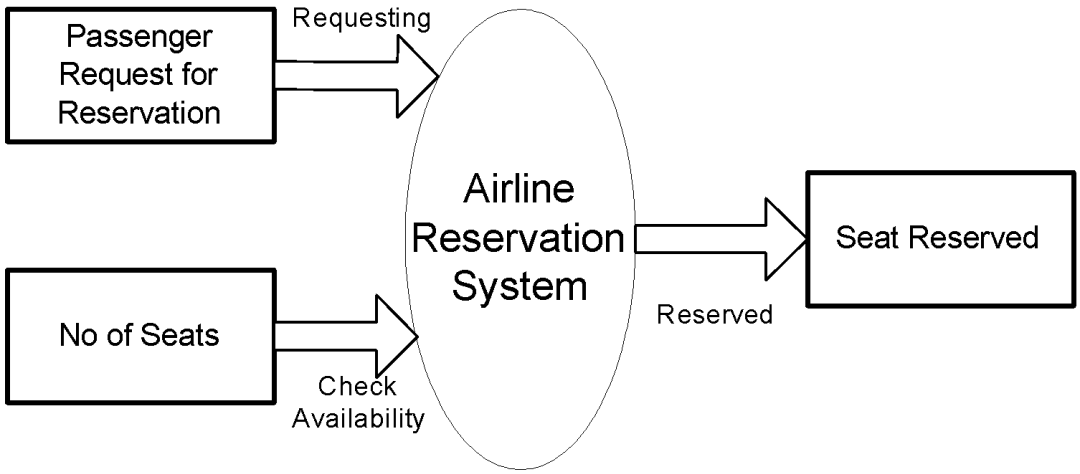
Operating system	:	Windows 2000 or later
Front End	:	Visual Basic
Back End	:	SQL Server 2000

HARDWARE SPECIFICATIONS

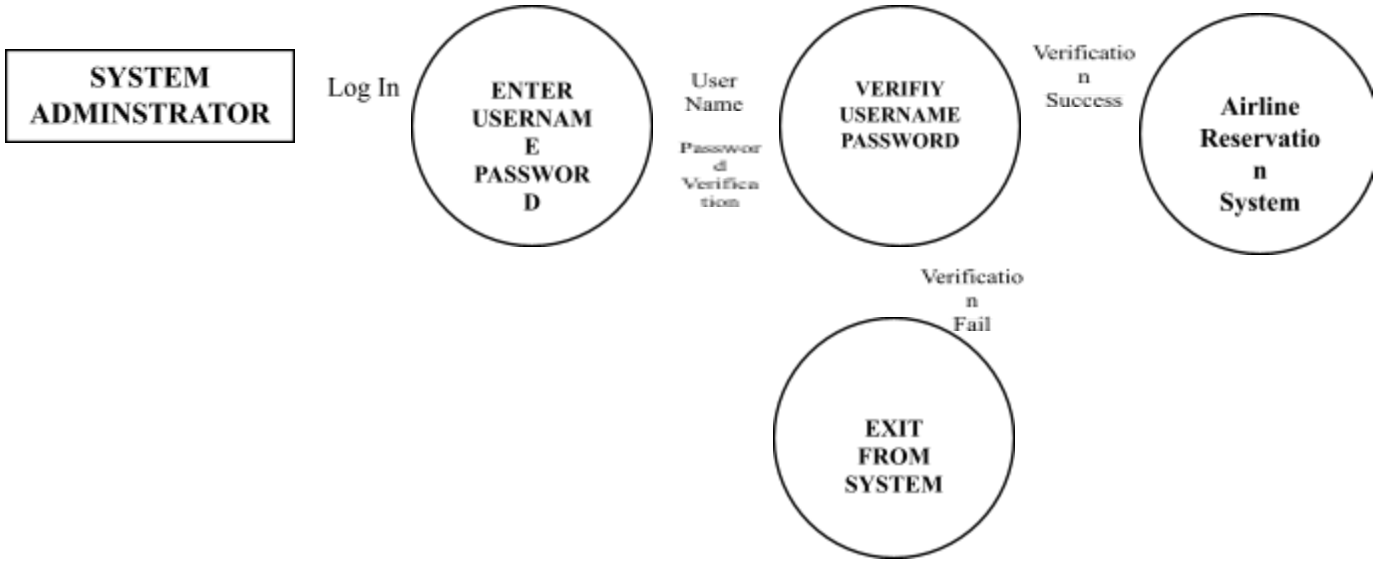
Processor	:	Intel Pentium or more
Ram	:	128 MB or more
Cache	:	512 KB
Hard disk	:	16 GB hard disk recommended

4. ANALYSIS

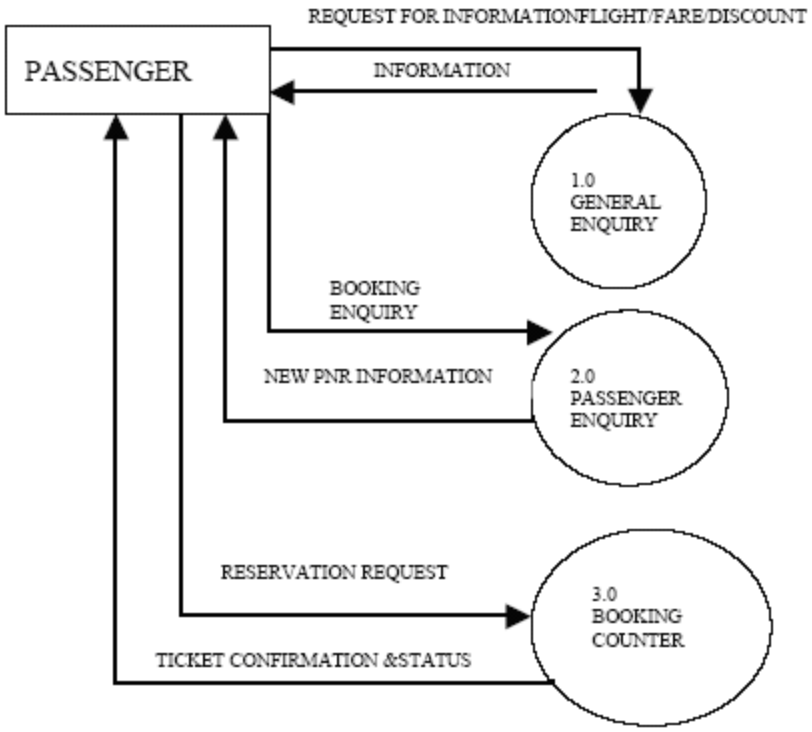
DFD for Airline Reservation System



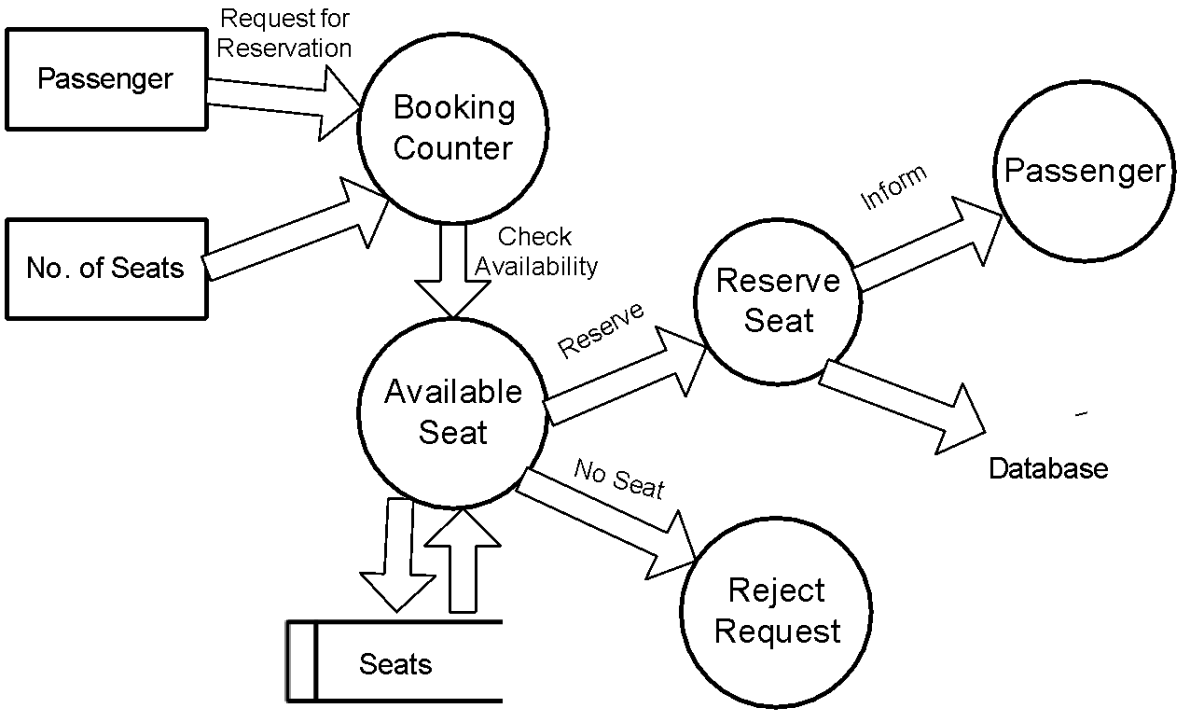
First Level of Data Flow Diagram for System Login



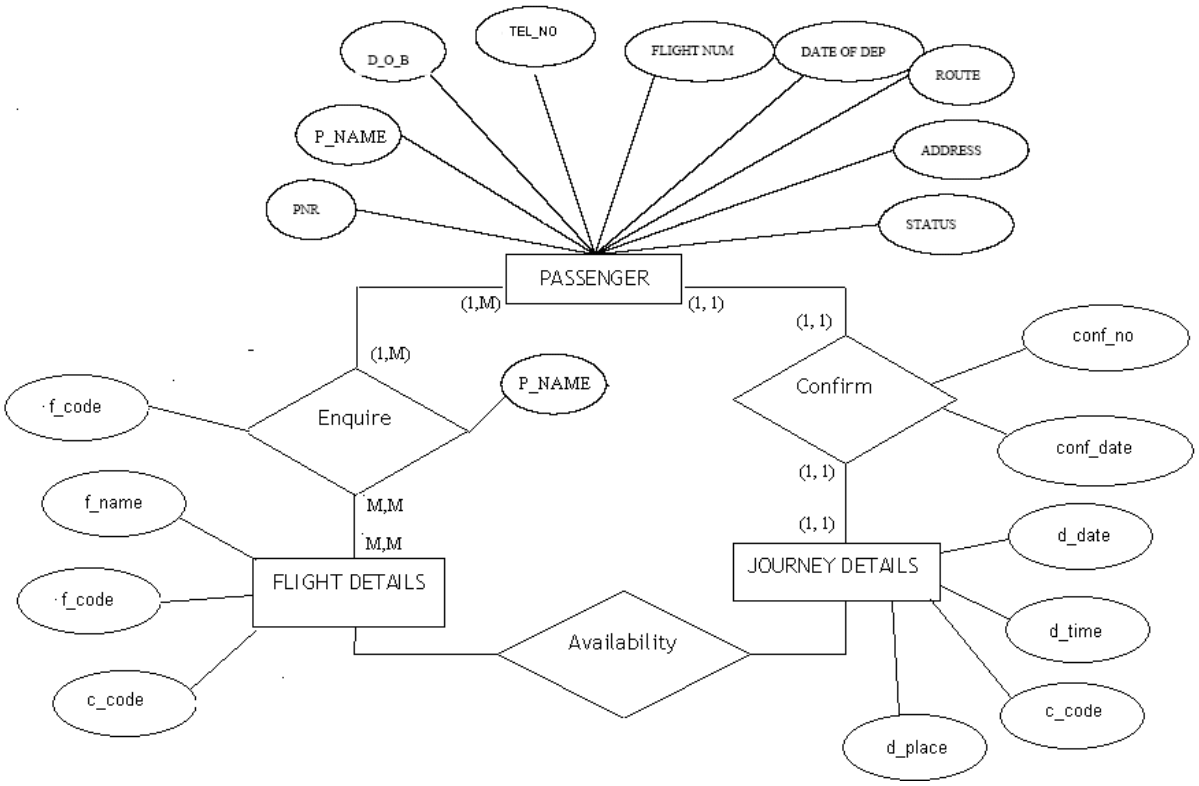
Second Level of Data Flow Diagram for General Inquiry System



Third Level DATA FLOW DIAGRAM OF BOOKING SECTION

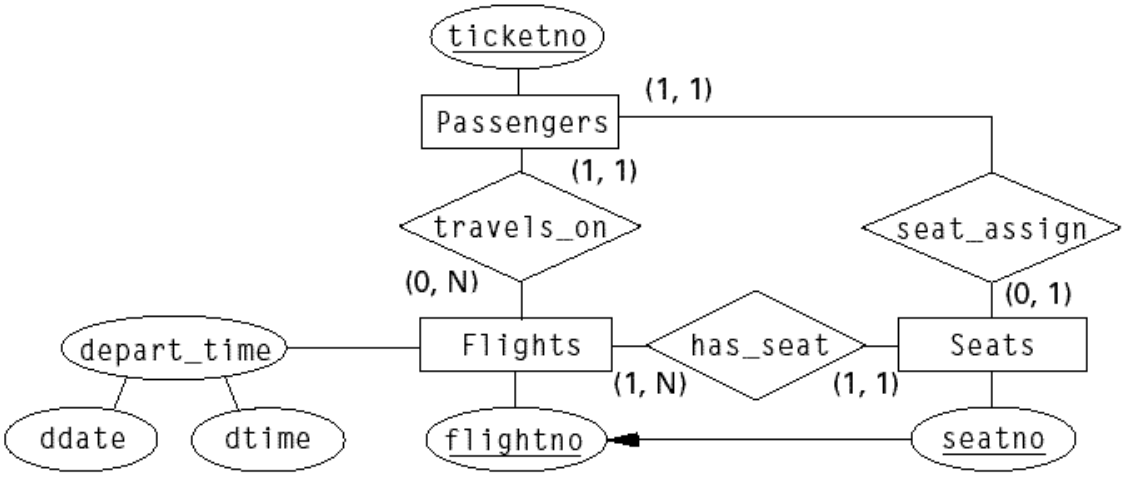


E-R Diagram



E-R Diagram for Airline Reservation System

ER-DIAGRAM FOR PASSENGER



5. SOFTWARE ENGINEERING APPROACH

The field of software engineering is related to the development software in systematic manner unlike simple programs which can be developed in isolation and there may not be any systematic approach being followed. As there is large difference between programming and software engineering. As it provide models that lead to the production of well documented software in a manner that is predictable. For a mature process, it should be possible to determine in advance how much time and effort will be required to produce the final product. To develop successful software I have to follow some models, which act as guidelines.

The model I have used is **Waterfall Model or Classic Life Cycle**. In this model first of all the existed system is observed. Then customer requirements are taken in consideration then planning, modeling, construction and finally deployment.

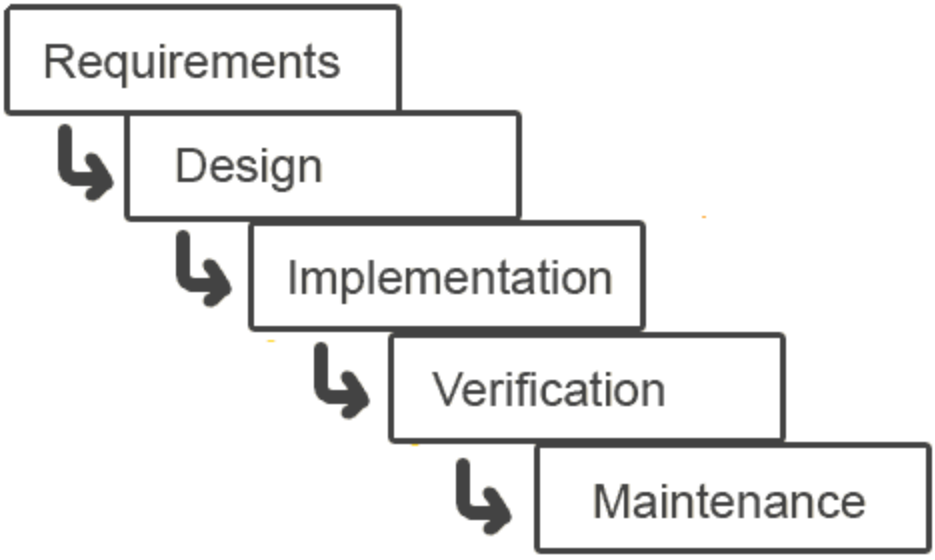


Fig.1. Waterfall Model

6. SYSTEM DESIGN

Introduction

Analysis collects a great deal of unstructured data through interviews, questionnaires, on-site observations, and procedural manuals and like. It is required to organize and convert the data through system flowcharts, data flow diagrams, structured English, decision tables and the like which support future development of the system.

The Data flow diagrams and various processing logic techniques show how, where, and when data are used or changed in an information system, but these techniques do not show the definition, structure and relationships within the data.

It is a way to focus on functions rather than the physical implementation. This is analogous to the architect's blueprint as a starting point for system design. The design is a solution, a "how to" approach, compared to analysis, a "what is" orientation.

System design is a highly creative process. This system design process is also referred as data modeling. The most common formatted used the E-R notation explains the characteristics and structure of data independent of how the data may be stored in computer memories.

The process of system design can be divided into three stages. They are:

- Structure design (already discussed)
- Database design

- Interface design

As we know that system design is a solution to "How to approach to the creation of new system". It provides the understudying and procedural details necessary for implementing the system. The steps involved during system design were as follow: -

LOGICAL AND PHYSICAL DESIGN

The current physical system was thoroughly reviewed from point of view how the data flow, what are file contents, its volumes and frequency etc.

After this input, output specifications security & control specification were prepared. It was also decided that how physical information will flow through the system and a physical design walkthrough.

OUTPUT DESIGN

The format of outputs is designed in such a way that it is simple to read and interpret In the present output we have clearly labeled title it contains date and time and all the fields are clearly mentioned (labeled).

INPUT DESIGN

.Input should be as simple as possible. It is design to reduce possibility of incorrect data being enter and the need of system user are considered with this view of mind several human factor is evaluated.

SCREEN DESIGN

The screen design for inputting the inputs were also panned as the format of inputs.

7. DATABASE DESIGN

Table: *login*

FIELDS	DATATYPE	DESCRIPTION	CONSTRAINT
Username	varchar (50)	User Name	Primary Key
Password	varchar (50)	User Password	

Table: *tblCity*

FIELDS	DATATYPE	DESCRIPTION	CONSTRAINT
CityCode	int (4)	Store City Code	Primary Key
CityName	varchar (50)	Store City Name	

Table: *tblClass*

FIELDS	DATATYPE	DESCRIPTION	CONSTRAINT
ClassCode	int (4)	Store Class Code	Primary Key
ClassName	varchar (50)	Store Class Type	
FixedFare	int (4)	Store Fixed Fare of Class	

Table: *tblFair*

FIELDS	DATATYPE	DESCRIPTION	CONSTRAINT
FlightCode	int (4)	Store Flight Code	Primary Key
ClassName	varchar (50)	Store Class Type Name	
FixedFair	int (4)	Store Fixed Price Per Seat	
FlightName	varchar (50)	Store Flight Name	
CityFrom	varchar (50)	Store Arrival City Name	
CityTo	varchar (50)	Store Destination City Name	
NormalFair	int (4)	Store Normal Fair	
BaggageFairPerKg	varchar (50)	Store Luggage Fair	
TotalFair	varchar (50)	Store Total Fire of Flight	

Table: *tbleFlight*

FIELDS	DATATYPE	DESCRIPTION	CONSTRAINT
Flight_Code	int (4)	Store Flight Code	Primary Key
Flight_Name	varchar (50)	Store Flight Name	
Flight_Origin	varchar (50)	Store Flight Departure City	
Flight_Destination	varchar (50)	Store Flight Destination Name	
Flight_Arr_Time	varchar (50)	Store Flight Arrival Time	
Flight_Dep_Time	varchar (50)	Store Flight Departure Time	
No_Seats	int (4)	Store Number of Seats	
Fare	int (4)	Store Fare Price	

Table: *tblPass*

FIELDS	DATATYPE	DESCRIPTION	CONSTRAINT
Flight_Code	int (4)	Store Flight Code	Primary Key
Flight_Name	varchar (50)	Store Flight Name	
Origin	varchar (50)	Store Flight Departure City Nam	
Destination	varchar (50)	Store Flight Destination Name	
PassengerName	varchar (50)	Store Passenger Name	
SeatNo	int (4)	Store Seat No	
Fare	int (4)	Store Faire	
Date_of_Res	varchar (50)	Store Date of Reservation	
Ticket_No	varchar (50)	Store Ticket No	

8. PROGRAM CODE & INPUT-OUTPUT SCREEN

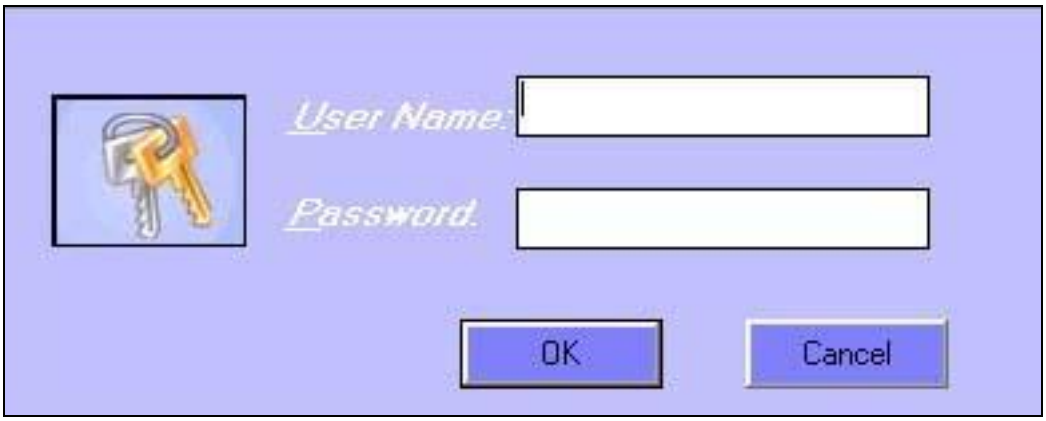
Module1

```
Public cmk As String

Public sqk As String

Public rsFlight As Recordset

Public rsCity As Recordset

Public rsPassanger As Recordset

Public rsLogin As Recordset

Public rs4 As Recordset

Public rs5 As Recordset

Public con As Connection
```

```
Public Sub DatabaseConnection()

Set rsFlight = New Recordset

Set rsCity = New Recordset

Set rsFare = New Recordset

Set rsPassanger = New Recordset

Set rsLogin = New Recordset

Set rs5 = New Recordset

Set con = New Connection
```

```
cmk = "Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=RajBCA"

With con
.ConnectionString = cmk
.Open
End With
con.CursorLocation = adUseClient
rsLogin.Open "select * from Login", con, adOpenDynamic, adLockOptimistic

End Sub
```

Login Form




```
Dim num As Integer
    Dim u, p As String
    Option Explicit

Public LoginSucceeded As Boolean
```

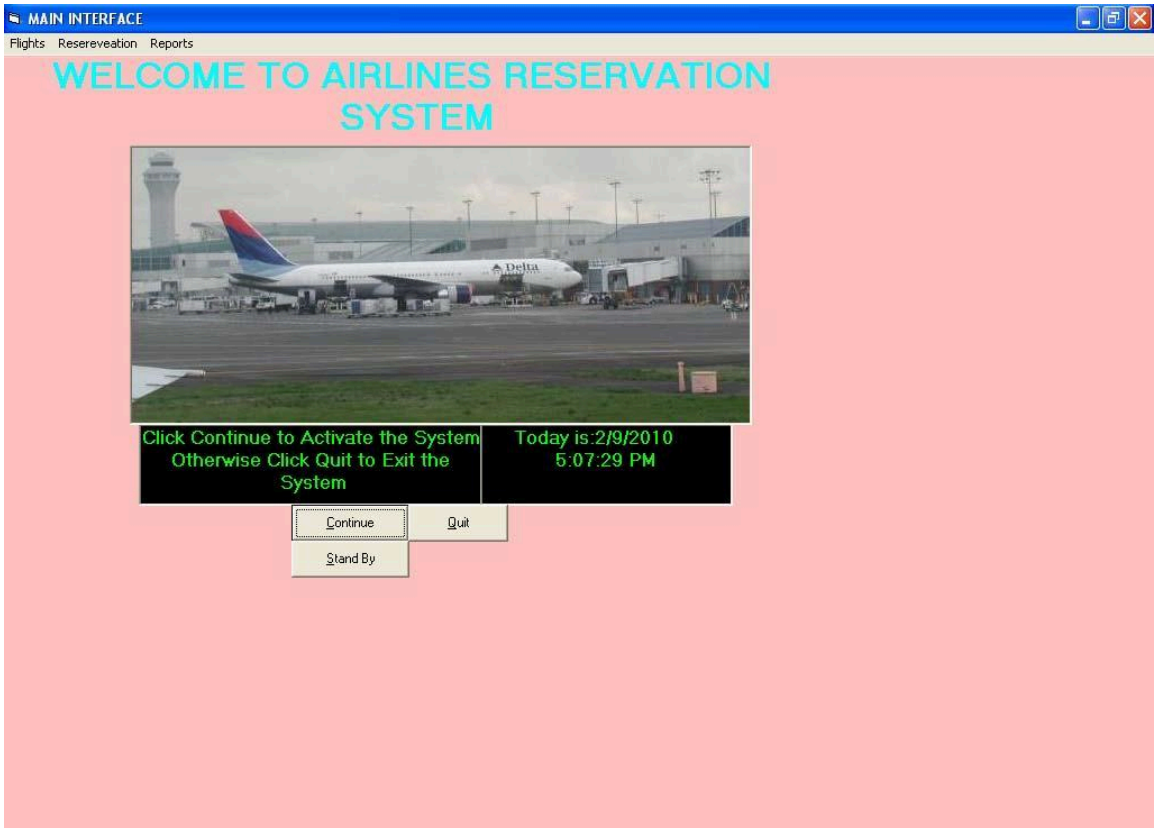
```
Private Sub cmdCancel_Click()
        End
End Sub
```

```
Private Sub cmdOK_Click()
        With rsLogin
        While Not rsLogin.EOF
        u = .Fields(0)
        p = .Fields(1)
        num = .Fields(2)
        If txtUName.Text = u And txtPass.Text = p Then
                If num = 1 Then
                        MsgBox "A c c e s s  G r a n t e d   " & Time, vbOKOnly,
                       "Authentication"

                         FrmMain.Show

                        Unload Me

                        txtPass.Text = ""
```

```
                End If
                .MoveNext

                Exit Sub
                Exit Sub
                Else
                .MoveNext
        End If
        Wend
        End With
                If txtUName.Text <> u Or txtPass.Text <> p Then
                MsgBox "Please Enter User name & Password", vbOKOnly + vbCritical,
                "Log In Error!"
                txtUName.Text = ""
                txtPass.Text = ""
                txtUName.SetFocus
                rsLogin.MoveFirst
        End If
        Exit Sub
End Sub
```

```
Private Sub Form_Load()
        DatabaseConnection
End Sub
```

Main Form




```
Dim i As Integer

        Private Sub Command1_Click()

        mnuFlights.Enabled = True

        mnuReservation.Enabled = True

        mnuReports.Enabled = True


End Sub
```

```
Private Sub Command2_Click()
        Dim ans As Integer
        ans = MsgBox("Are You Sure to Leave the System", vbYesNo + vbInformation,
        "Quit")
                If ans = 6 Then
                        End
                Else
                        End If
End Sub
```

```
Private Sub Command3_Click()
        mnuFlights.Enabled = False
        mnuReservation.Enabled = False
        mnuReports.Enabled = False
End Sub
```

```
Private Sub Form_Load()
        mnuFlights.Enabled = False
        mnuReservation.Enabled = False
        mnuReports.Enabled = False
        i = 10
End Sub
```

```
Private Sub mnuAddFlights_Click()
        frmAddFlights.Show
End Sub
```

```
Private Sub mnuCancelReservation_Click()
        FrmCancelRes.Show
End Sub
```

```
Private Sub mnuDeleteFlight_Click()
        frmDeleteFlight.Show
End Sub
```

```
Private Sub mnuDisplayAllFlight_Click()
        frmDisplayFlightDetails.Show
End Sub
```

```
Private Sub mnuEditFlights_Click()
        frmEditFlight.Show
End Sub
```

```
Private Sub mnuExit_Click()
        Unload Me
        End
End Sub
```

```
Private Sub mnuIssueTicket_Click()
        frmTicket.Show

End Sub
```

```
Private Sub mnuMarkeRes_Click()
        frmReservation.Show

End Sub
```

```
Private Sub mnuModifyReser_Click()
        frmModifyReservation.Show
End Sub
```

```
Private Sub mnuPassangerlist_Click()
        FrmPassangerList.Show
```

```
End Sub
```

```
Private Sub Timer1_Timer()
        Label3.Caption = "Today is:" & Date & "     " & Time
End Sub
```

Add Flights Form


```
Private Sub cmdAdd_Click()
        txtCode.Text = ""
        txtName.Text = ""
        cmbOrg.Text = ""
        cmbDest.Text = ""
        txtDepTime.Text = ""
        txtSeats.Text = ""
         txtFare.Text = ""

        txtArrTime.Text = Time
        cmdAdd.Enabled = False
```

```
        cmdSave.Enabled = True


End Sub
```

```
Private Sub cmdExit_Click()
        Unload Me
        FrmMain.Show


End Sub
```

```
Private Sub cmdSave_Click()
        If txtCode.Text = "" Or txtName.Text = "" Or cmbOrg.Text = "" Or cmbDest.Text
        = "" Or txtArrTime.Text = "" Or txtDepTime.Text = "" Or txtSeats.Text = "" Then
                MsgBox "Can'nt Leave any Field Blank", vbOKCancel + vbCritical,
                "Error"
        Else
        If (cmbOrg.Text = cmbDest.Text) Then
                MsgBox "No Flights can be added to Same Cities", vbOKCancel +
                vbCritical, "Error"
        Else
                con.Execute ("insert into TblFlight values(" & txtCode.Text & ",'" &
                txtName.Text & "','" & cmbOrg.Text & "','" & cmbDest.Text & "','" &
```

```
            txtArrTime.Text & "','" & txtDepTime.Text & "', " & txtSeats.Text & ", "
            & txtFare.Text & ") ")
            MsgBox "Record Saved", vbOKCancel + vbInformation, "Saved"
        End If
            cmdSave.Enabled = False
            cmdAdd.Enabled = True
        End If
End Sub
```

```
Private Sub Form_Load()
        DatabaseConnection
        txtArrTime.Text = Time
        Set rsCity = con.Execute("select CityName from TblCity")
        Do Until rsCity.EOF
        cmbOrg.AddItem rsCity(0)
        rsCity.MoveNext
        Loop
        rsCity.MoveFirst
        Do Until rsCity.EOF
        cmbDest.AddItem rsCity(0)
        rsCity.MoveNext
        Loop
```

```
        'cmdSave.Enabled = False

End Sub
```

Edit Flights Form


```
Private Sub cmdEdit_Click()
        Set rsFlight = con.Execute("update  TblFlight set Flight_Name='" & txtName.Text
        & "',Flight_Orgin='" & cmbOrg.Text & "',Flight_Destination='" & cmbDest.Text &
        "',Flight_Arr_Time='" & txtArrTime.Text & "',Flight_Dep_Time='" &
        txtDepTime.Text & "',No_of_seats=" & txtSeats.Text & ",Fare=" & txtFare.Text &
        " where Flight_Name='" & cmdFlightName.Text & " '")
        cmdEdit.Enabled = False
        cmdSave.Enabled = True

End Sub
```

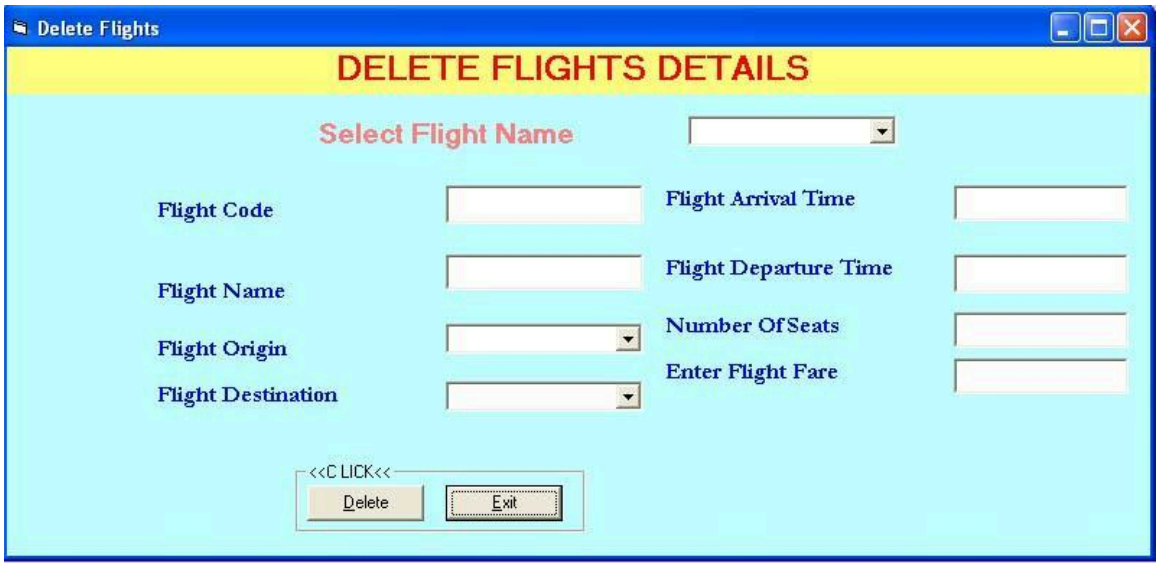
```
Private Sub cmdExit_Click()
        Unload Me
        FrmMain.Show

End Sub
```

```
Private Sub cmdFlightName_Click()
        Set rsFlight = con.Execute("select *from TblFlight where Flight_Name='" &
        cmdFlightName.Text & " '")
        txtCode.Text = rsFlight(0)
        txtName.Text = rsFlight(1)
        cmbOrg.Text = rsFlight(2)
        cmbDest.Text = rsFlight(3)
        txtArrTime.Text = rsFlight(4)
        txtDepTime.Text = rsFlight(5)
        txtSeats.Text = rsFlight(6)
        txtFare.Text = rsFlight(7)

End Sub
```

```
Private Sub cmdSave_Click()
        MsgBox "Record Updated", vbOKCancel + vbInformation, "Updated"
        cmdSave.Enabled = False
        cmdEdit.Enabled = True

End Sub
```

```
Private Sub Form_Load()
        DatabaseConnection
        Set rsFlight = con.Execute("select *from TblFlight")
        Do Until rsFlight.EOF
        cmdFlightName.AddItem rsFlight(1)
        rsFlight.MoveNext
        Loop
        cmdSave.Enabled = False

End Sub
```

Delete Flights Form



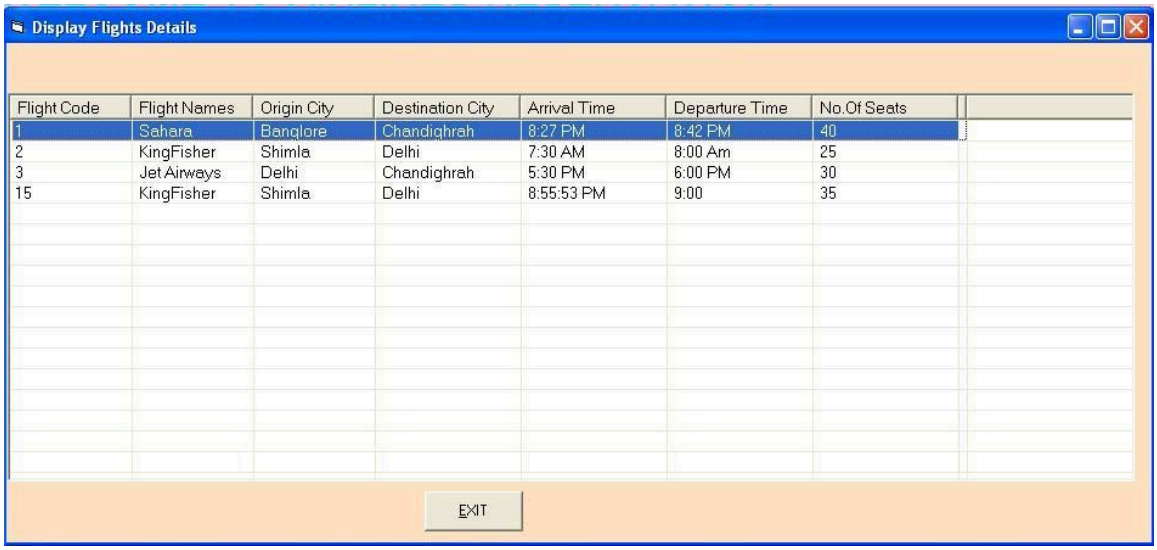

```
Private Sub cmdDelete_Click()
        Dim ans As Integer
        ans = MsgBox("Do You really want to delete the flight details", vbYesNo +
        vbInformation, "Confirm Delete")
        If ans = vbYes Then
                con.Execute ("delete from TblFlight where  Flight_Name='" &
                cmdFlightName.Text & " ' And Flight_Code=" & txtCode.Text & " ")
                MsgBox "Record Deleted", vbOKCancel + vbInformation, "Deleted"
        End If
End Sub
```

```
Private Sub cmdExit_Click()
        Unload Me
        FrmMain.Show

End Sub
```

```
Private Sub cmdFlightName_Click()
        Set rsFlight = con.Execute("select *from TblFlight where Flight_Name='" &
        cmdFlightName.Text & " ' ")

        txtCode.Text = rsFlight(0)
        txtName.Text = rsFlight(1)
        cmbOrg.Text = rsFlight(2)
        cmbDest.Text = rsFlight(3)
        txtArrTime.Text = rsFlight(4)
        txtDepTime.Text = rsFlight(5)
        txtSeats.Text = rsFlight(6)
        txtFare.Text = rsFlight(7)

End Sub
```

```
Private Sub Form_Load()
        DatabaseConnection
        Set rsFlight = con.Execute("select *from TblFlight")
        Do Until rsFlight.EOF
        cmdFlightName.AddItem rsFlight(1)
        rsFlight.MoveNext
        Loop

End Sub
```

Display Flight Details




```
Private Sub Command1_Click()
        Unload Me
        FrmMain.Show

End Sub
```

```
Private Sub Form_Load()
        DatabaseConnection
        Set rsFlight = con.Execute("select *from TblFlight")
        fill_list

End Sub
```

```
Public Sub fill_list()
        StudList.ListItems.Clear
        If rsFlight.RecordCount = 0 Then Exit Sub

        While Not rsFlight.EOF

        Set lst = StudList.ListItems.Add(, , rsFlight(0))
           For eX = 1 To 7
           lst.SubItems(eX) = rsFlight(eX)
           Next eX
        rsFlight.MoveNext
        Wend
End Sub
```

Make Your Reservation Form


```
Private Sub cmdExit_Click()
        Unload Me
        FrmMain.Show
End Sub
```

```
Private Sub cmdFlightName_Click()
        Set rsFlight = con.Execute("select *from TblFlight where Flight_Name='" &
        cmdFlightName.Text & " '")
        'Set rsFare = con.Execute("select *from TblFare where FlightName='" &
        cmdFlightName.Text & "' and flightCode=" & txtCode.Text & "")
```

```
        txtCode.Text = rsFlight(0)
        cmbOrg.Text = rsFlight(2)
        cmbDest.Text = rsFlight(3)
        txtArrTime.Text = rsFlight(4)
        txtDepTime.Text = rsFlight(5)
        txtSeats.Text = rsFlight(6)
        txtFare.Text = rsFlight(7)
End Sub
```

```
Private Sub cmdSave_Click()
        If txtName.Text = "" Or txtYourSeat.Text = "" Then
                MsgBox "Provide All Information", vbOKCancel + vbInformation,
                "Error"
        Else
                lblTicket.Visible = True
                Label13.Visible = True
                lblTicket.Caption = cmdFlightName.Text & txtYourSeat.Text &
                Label14.Caption
                con.Execute ("insert into TblPas values(" & txtCode.Text & ",'" &
                cmdFlightName.Text & "','" & cmbOrg.Text & "','" & cmbDest.Text &
```

```
"'," & txtName.Text & "'," & txtYourSeat.Text & ", " & txtFare.Text & ",'"
& txtDate.Text & "','" & lblTicket.Caption & "' ) ")

MsgBox "Record Saved", vbOKCancel + vbInformation, "Saved"
End If
End Sub
```

```
Private Sub Command1_Click()
txtCode.Text = ""
txtName.Text = ""
cmbOrg.Text = ""
cmbDest.Text = ""
txtFare.Text = ""
txtYourSeat.Text = ""
txtDepTime.Text = ""
txtArrTime.Text = ""
txtSeats.Text = ""
Label13.Visible = False
lblTicket.Visible = False
lblTicket.Caption = ""
End Sub
```

```
Private Sub Form_Load()
        DatabaseConnection
        Set rsFlight = con.Execute("select *from TblFlight")
        Do Until rsFlight.EOF
                cmdFlightName.AddItem rsFlight(1)
                rsFlight.MoveNext
        Loop
        txtDate.Text = Date
End Sub
```

```
Private Sub Timer1_Timer()
        Label14.Caption = Time
End Sub
```

Cancel Reservation Form


```
Private Sub cmbTickeNo_Click()
        Set rsPassanger = con.Execute("select *from TblPas where TickertNumber='" &
        cmbTickeNo.Text & " ' ")
        txtName.Text = rsPassanger(4)
        txtFlightName.Text = rsPassanger(1)

End Sub
```

```
Private Sub cmdDelete_Click()
        Dim ans As Integer
        ans = MsgBox("Do You really want to Cancel", vbYesNo + vbInformation,
        "Confirm Delete")

        If ans = vbYes Then
                con.Execute ("delete from TblPas where  TickertNumber='" &
                cmbTickeNo.Text & " ' And PassangerName='" & txtName.Text & "' ")
                MsgBox "Record Cancaled", vbOKCancel + vbInformation, "Deleted"
        End If

End Sub
```

```
Private Sub cmdExit_Click()
        Unload Me
        FrmMain.Show

End Sub
```

```
Private Sub Form_Load()
        DatabaseConnection
        Set rsPassanger = con.Execute("select *from TblPas")

        Do Until rsPassanger.EOF
                cmbTickeNo.AddItem rsPassanger(8)
                rsPassanger.MoveNext
        Loop
End Sub
```

Modify Your Reservation Form


```
Private Sub cmbTickeNo_Click()
        Set rsPassanger = con.Execute("select *from TblPas where TickertNumber='" &
        cmbTickeNo.Text & " ' ")
        txtName.Text = rsPassanger(4)

        txtYourSeat.Text = rsPassanger(5)
        txtDate.Text = rsPassanger(7)
        txtCode.Text = rsPassanger(0)
        cmdUpdate.Enabled = True
End Sub
```

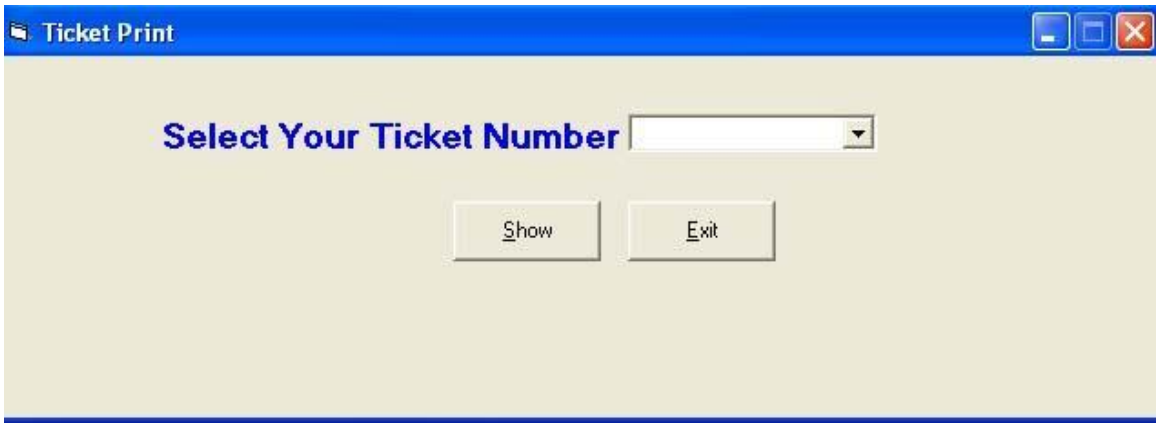
```
Private Sub cmdExit_Click()
        Unload Me
        FrmMain.Show

End Sub
```

```
Private Sub cmdSave_Click()
        cmdUpdate.Enabled = True
        MsgBox "Record Updated"
        cmdSave.Enabled = False

End Sub
```

```
Private Sub cmdUpdate_Click()
        Set rsFlight = con.Execute("update  TblPas set PassangerName='" &
        txtName.Text & "',SeatNo=" & txtYourSeat.Text & ",DateOfRes='" &
        txtDate.Text & "' where TickertNumber='" & cmbTickeNo.Text & "'")
        cmdUpdate.Enabled = False
        cmdSave.Enabled = True

End Sub
```

```
Private Sub Form_Load()
        DatabaseConnection
        Set rsPassanger = con.Execute("select *from TblPas")
        Do Until rsPassanger.EOF
                cmbTickeNo.AddItem rsPassanger(8)
                rsPassanger.MoveNext
        Loop

End Sub
```

Ticket Print Form




```
Private Sub Command1_Click()
Dim id As Integer
With DataEnvironment1

   If con.Provider = "SQLOLEDB.1" Then
        .Commands(1).CommandType = adCmdText
        .Commands(1).CommandText = "SELECT * FROM TblPas where
TickertNumber = '" & cmbTickeNo.Text & "'"
        .Commands(1).Execute

      DtrTicket.Show
```

```
        If .rsCommand1.State = 1 Then
          .rsCommand1.Close
        End If
    Else
        id = code

        .Commands(1).CommandType = adCmdText
                .Commands(1).CommandText = "select * " & "from receipt where
receipt_no = " & id
        .Commands(1).Execute
       DtrTicket.Show
          .rsCommand1.Close

       End If
  End With
End Sub
```

```
Private Sub Command2_Click()
          Unload Me
          FrmMain.Show
```

End Sub

```
Private Sub Form_Load()
        DatabaseConnection
        Set rsPassanger = con.Execute("select *from TblPas")
        Do Until rsPassanger.EOF
                cmbTickeNo.AddItem rsPassanger(8)
                rsPassanger.MoveNext
        Loop

End Sub
```

Passenger Details Form


```
Private Sub Command1_Click()
With DataEnvironment1

   If con.Provider = "SQLOLEDB.1" Then

         .Commands(1).CommandType = adCmdText
         .Commands(1).CommandText = "SELECT * FROM TblPas where FlightName =
'" & cmbTickeNo.Text & "'"
         .Commands(1).Execute
       DataReport1.Show
        If .rsCommand1.State = 1 Then
         .rsCommand1.Close
        End If
   Else
```

```
        id = code

          .Commands(1).CommandType = adCmdText
                    .Commands(1).CommandText = "select * " & "from receipt where
receipt_no = " & id
          .Commands(1).Execute

       DataReport1.Show

          .rsCommand1.Close

       End If

  End With

End Sub
```

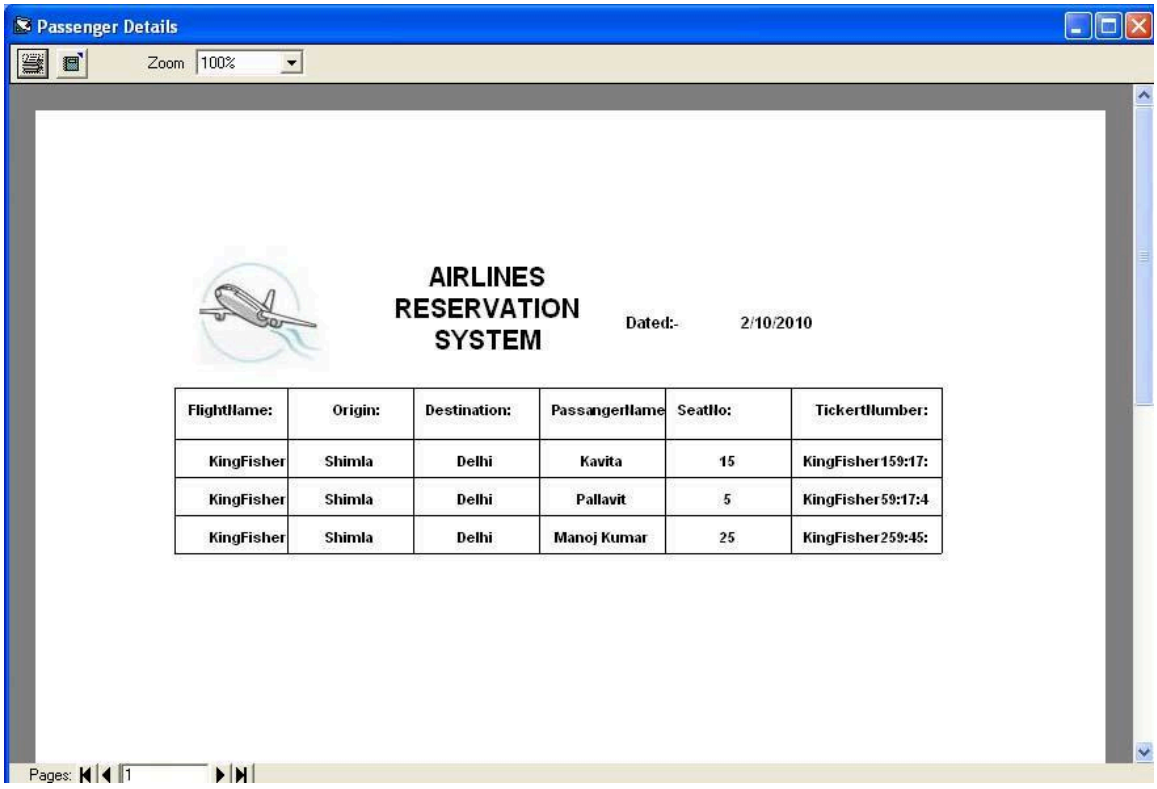
```
Private Sub Command2_Click()
          Unload Me
          FrmMain.Show
```

```
End Sub
```

```
Private Sub Form_Load()
        DatabaseConnection
        Set rsPassanger = con.Execute("select *from TblPas")
        Do Until rsPassanger.EOF
                cmbTickeNo.AddItem rsPassanger(1)
                rsPassanger.MoveNext
        For i = 0 To cmbTickeNo.ListCount - 1
                For j = i + 1 To cmbTickeNo.ListCount - 1
                        If cmbTickeNo.List(i) = cmbTickeNo.List(j) Then
                                cmbTickeNo.RemoveItem j
                        End If
                        Next j
                        Next i
        Loop
End Sub
```

Ticket Print

Passenger List Details Print



FlightName:	Origin:	Destination:	PassangerName	SeatNo:	TickertNumber:
KingFisher	Shimla	Delhi	Kavita	15	KingFisher159:17:
KingFisher	Shimla	Delhi	Pallavit	5	KingFisher59:17:4
KingFisher	Shimla	Delhi	Manoj Kumar	25	KingFisher259:45:

9. TESTING

Software Testing

Software testing is a process of *verifying* and *validating* that a software application or program. Software testing

1. Meets the business and technical requirements that guided its design and development, and
2. Works as expected.

Software testing also identifies important *defects*, flaws, or errors in the application code that must be fixed. The modifier "important" in the previous sentence is, well, important because defects must be categorized by severity.

During test planning we decide what an important defect is by reviewing the requirements and design documents with an eye towards answering the question "Important to whom?" Generally speaking, an important defect is one that from the customer's perspective affects the usability or functionality of the application. Using colors for a traffic lighting scheme in a desktop dashboard may be a no-brainer during requirements definition and easily implemented during development but in fact may not be entirely workable if during testing we discover that the primary business sponsor is color blind. Suddenly, it becomes an important defect. (About 8% of men and .4% of women have some form of color blindness.)

The quality assurance aspect of software development—documenting the degree to which the developers followed corporate standard processes or best practices—is not addressed in this paper because assuring quality is not a responsibility of the testing team. The testing team cannot improve quality; they can only measure it, although it can be argued that doing things like designing tests before coding begins will improve quality because the coders can then use that information while thinking about their designs and during coding and debugging.

Software testing has three main purposes: verification, validation, and defect finding.

- The *verification* process confirms that the software meets its technical specifications. A "specification" is a description of a function in terms of a measurable output value given a specific input value under specific preconditions. A simple specification may be along the line of "a SQL query retrieving data for a single account against the multi-month account-summary table must return these eight fields <list> ordered by month within 3 seconds of submission."
- The *validation* process confirms that the software meets the business requirements. A simple example of a business requirement is "After choosing a branch office name, information about the branch's customer account managers will appear in a new window. The window will present manager identification and summary information about each manager's customer base:

<list of data elements>." Other requirements provide details on how the data will be summarized, formatted and displayed.

- A *defect* is a variance between the expected and actual result. The defect's ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

Testing methods

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

Black box testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

Specification-based testing: Specification-based testing aims to test the functionality of software according to the applicable requirements. Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case.

Specification-based testing is necessary, but it is insufficient to guard against certain risks.

Advantages and disadvantages: The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code *must* have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers do not. *But,* on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed. As a result, there are situations when (1) a tester writes many test cases to check something that could have been tested by only one test case, and/or (2) some parts of the back-end are not tested at all.

Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

White box testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

Types of white box testing

- API testing (application programming interface) - Testing of the application using Public and Private APIs

- Code coverage - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
- Fault injection methods - improving the coverage of a test by introducing faults to test code paths
- Mutation testing methods
- Static testing - White box testing includes all static testing

A sample testing cycle

Although variations exist between organizations, there is a typical cycle for testing:

- **Requirements analysis**: Testing should begin in the requirements phase of the software development life cycle. During the design phase, testers work with developers in determining what aspects of a design are testable and with what parameters those tests work.
- **Test planning**: Test strategy, test plan, tested creation. Since many activities will be carried out during testing, a plan is needed.
- **Test development**: Test procedures, test scenarios, test cases, test datasets, test scripts to use in testing software.
- **Test execution**: Testers execute the software based on the plans and tests and report any errors found to the development team.

- **Test reporting**: Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.
- **Test result analysis**: Or Defect Analysis, is done by the development team usually along with the client, in order to decide what defects should be treated, fixed, rejected (i.e. found software working properly) or deferred to be dealt with later.
- **Defect Retesting**: Once a defect has been dealt with by the development team, it is retested by the testing team.
- **Regression testing**: It is common to have a small test program built of a subset of tests, for each integration of new, modified, or fixed software, in order to ensure that the latest delivery has not ruined anything, and that the software product as a whole is still working correctly.
- **Test Closure**: Once the test meets the exit criteria, the activities such as capturing the key outputs, lessons learned, results, logs, documents related to the project are archived and used as a reference for future projects.

TESTING SNAPSHOTS

(When Unauthorized user try to access the system)

Error	Description	Requirement
Please Enter User name & Password	Error occurred due to authorization failure	Correct Username & Password

(when some required fields in form not filled)

Error	**Description**	**Requirement**
Can't Leave any Field Blank	Error occurred due to Incomplete Information	Fill all required fields

(When user try to add flight from same city to same city)

Error	Description	Requirement
No Flights can be added to same cities	Error occurred due to same flight origin and destination	Enter different cities in both and origin and destination

10. FUTURE SCOPE

- This system can be updated as online system.
- Multi-user Interface can be added to this system.
- As a Aviaries prepared for future growth it determine, it would replace open skies airline reservation system.

BIBLIOGRAPHY

Websites

- http://www.google.com
- http://www.microsoft.com
- http://www.codeproject.com
- http://www.msdn.com.
- http://www.vb123.com
- http://www.vbcode.com
- http://www.sqltuner.com

Books

- Mastering Visual Basic 6 (Paperback)
- Visual Basic Black Book (Paperback)
- SQL Bible, 2nd Edition (Paperback)
- Database Development in Visual Basic