P-1496

SEAMLESS INTERCONNECTION BETWEEN LEGACY SYSTEMS USING WEB SERVICES

A PROJECT REPORT

Submitted by

NAME	REG.NO
D.G.ARJUN	71201104006
R.RAJESH KANNAN	71201104038
VISWA N. MOHAMMED AZARUDDIN	71201104071

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY : CHENNAI 600025

APRIL 2005

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"SEAMLESS INTERCONNECTION BETWEEN LEGACY SYSTEM USING WEB SERVICES"** is the bonafide work of **D.G.ARJUN (71201104006), R.RAJESH KANNAN (71201104038) and VISWA N. MOHAMMED AZARUDDIN (71201104071),** who carried out the project work under my supervision.

SIGNATURE

Dr. S. Thangasamy

HEAD OF THE DEPARTMENT

Department of
Computer Science and Engg,
Kumaraguru College of Technology,
Chinnavedampatti P.O.,
Coimbatore - 641006

SIGNATURE

Ms.V.S.Akshaya M.E.,

SUPERVISOR

Lecturer

Department of
Computer Science and Engg,
Kumaraguru College of Technology,
Chinnavedampatti P.O.,
Coimbatore - 641006

Submitted for the Viva-voce Examination held on 20th April 2005

Internal Examiner

External Examiner

ANNA UNIVERSITY : CHENNAI 600025

EVALUATION CERTIFICATE

College : **KUMARAGURU COLLEGE OF TECHNOLOGY**

Branch : **COMPUTER SCIENCE AND ENGINEERING**

Semester: **EIGHT (08)**

S.No	Name of the Student	Title of the Project	Name of Supervisor
1	D.G.ARJUN	Seamless Interconnection between Legacy systems using Web Services	Ms.V.S.Akshaya, Lecturer
2	R.RAJESH KANNAN	Seamless Interconnection between Legacy systems using Web Services	Ms.V.S.Akshaya, Lecturer
3	VISWA N. MOHAMMED AZARUDDIN	Seamless Interconnection between Legacy systems using Web Services	Ms.V.S.Akshaya, Lecturer

The report of the project work submitted by the above students in partial fulfillment for the award of BACHELOR OF ENGINEERING degree in COMPUTER SCIENCE AND ENGINEERING of Anna University were evaluated and confirmed to be the report of the work done by the above students and then evaluated.

(INTERNAL EXAMINER) **(EXTERNAL EXAMINER)**

DECLARATION

We hereby declare that the project entitled **"SEAMLESS INTERCONNECTIONS BETWEEN LEGACY SYTEMS USING WEB SERVICES"**, is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any other institution, for fulfillment of the requirement of the course study.

This report is submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University, Chennai.

Place: Coimbatore
Date : 20.04.2005

(D.G.Arjun)

(R.Rajesh Kannan)

(Viswa N. Mohammed Azaruddin)

ACKNOWLEDGEMENTS

We would like to express our gratitude to the Principal, Dr. K. K. Padmanabhan for helping us to complete this project successfully.

We would express our sincere thanks to our Head of the Department Dr.S.Thangasamy for motivating and inspiring us during the course of this project.

We have immense pleasure in expressing our heartfelt thanks to our guide, Ms.V.S.Akshaya, Lecturer, for her constant advice and support during the project. We are grateful to her for her guidance.

We would like to thank our project coordinator, Mrs.P.Devaki, Assistant Professor, for her support during the course of our project.

We would like to express our sincere thanks to all the members of the faculty of the Department of Computer Science & Engineering for their support.

We would like to express our gratitude to Mr. SakthiVel Rajamanickam, Senior Systems Analyst, Hewlett Packard, for encouraging us to take up a project of this nature.

We thank many of our patient fellow students for listening about the problems we were tackling and helping us understand them more clearly by

asking the right questions. We wish to single out the following people, in particular (in alphabetical order): Raghavendran and Santhosh Kumar.

We thank all those who have been involved directly or indirectly in our project.

ABSTRACT

We present a method for accessing legacy systems from modern day systems without making changes to either the legacy system or the modern day system. Here, we build a web service that is capable of extracting the various parameters from one format and converting it to the other format so that the home system understands the format. We make use of wrapper classes and adapters which decide the nature of the legacy and modern systems. The base concept of the project is to make changes to the web service rather than the systems, in order to make the systems inter-operable. Every year, millions of dollars are wasted in developing changes in legacy systems so as to create a compatibility to modern systems. The project is expected to alleviate the problem.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

UDDI - Universal Description, Discovery and Integration

SOAP - Simple Object Access Protocol

XML - Extensible Markup Language

WSDL - Web Service Definition Language

CRUD - Create Read Update Delete

1.INTRODUCTION

As your large enterprise grows, your IT challenges grow with it. Virtually all enterprises have legacy applications and databases, and they want to continue to use them while adding or migrating to a new set of applications that utilize the Internet, e-commerce, the extranet, and other new technologies. New platforms foster new technologies and capabilities. Rewriting legacy applications running on archaic systems to connect them to this new functionality is usually expensive and time-consuming.

In addition, as organizations expand and merge, their IT platform deployments often become fragmented. For example, different departments within an organization may use different applications and persistence mechanisms to access the same data. As a result, some data, such as customer information, may exist in multiple locations. The problems this can cause range from lack of efficiency (having to enter the same data multiple times) to inconsistency in data that is stored in different locations. Conflicting data stored in different locations results in higher operations costs, reduced customer satisfaction, and other negative impacts to an organization's bottom line. Having a unified view of all mission-critical data is beyond the capabilities of such a fragmented system.

This problem is not new. Connecting to the legacy applications saves the time and expense of having to migrate the legacy applications, and it provides a mechanism for tying together fragmented IT platforms. For years, IT departments have created point-to-point mechanisms, and middleware

developers have written millions of lines of code for achieving interoperability of data sources and applications.

1.1 EXISTING SYSTEM AND ITS LIMITATIONS

Typical solutions use a combination of rewriting legacy code to run on more modern systems or installing so-called "middleware" applications to patch together disparate systems into a cohesive whole. There are significant downsides to both of these approaches. Rewriting legacy systems can amount to reinventing these systems from scratch. This can be extremely costly and sometimes result in solutions that are less reliable and cost-effective than the original. Patching together disparate solutions with middleware creates multiple points of failure and an overall system architecture so complex that increased maintenance costs can eliminate any benefit that the integration may have provided.

In the type of solution shown in Figure 1.1, each application requires custom code to access the legacy application. Any changes in the interface exposed by the legacy application would require modifying each of the Client applications individually, multiplying the development effort required. If multiple programming languages exist within your Client applications, then development expertise in those languages will also be required. Some older applications use languages that have quite cryptic networking APIs, further hampering interface efforts. Because of scenarios like this, in the past, most attempts have resulted in point-to-point solutions that did not scale

and were difficult to maintain. However, the widespread adoption of XML standards and Web services has made effective interfaces and integration more attainable.

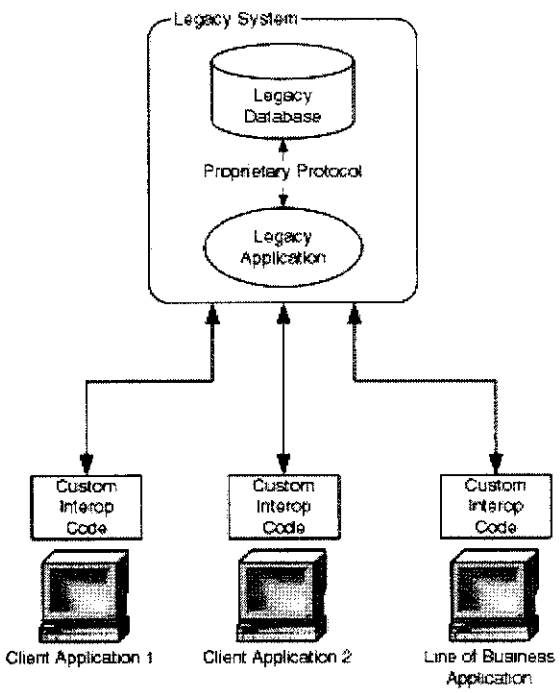


Figure 1.1 – Existing System Architecture

1.2 PROPOSED SYSTEM AND ITS ADVANTAGES

The proposed system incorporates the usage of web services to remove the dependability of the system on the existing architectures. The web service alone can be modified based upon the requirement of the legacy system (flat file) or the front end. The Web Service makes use of wrapper classes to extract the particular parameters. These parameters are then packaged into XML for platform independent transport of parameters. SOAP is used along the path. The XML parameters are extracted by the front end(Aircraft booking system) and the changes are displayed. In the end a seamless integration between both the legacy system and the front end with both system not needing any change in the existing code.

Development of Web services can be focused on the business logic within the applications rather than on the tedious details of formatting, and of transporting data. Some of the benefits include:

- **Web services**—Web services and supporting technologies including SOAP are quickly becoming the standard building blocks for modern distributed application design.
- **HTTP**—The ubiquitous presence of HTTP makes it the obvious choice for a standardized communication layer through which components of a distributed application can interact. An HTTPS channel should be used for point-to-point secure communications.
- **XML**—XML provides a platform-neutral format for data transfer and provides validation for content and structure

- **Secure Sockets Layer (SSL)**—SSL provides data security using industry standard encryption protocols.

2 PROPOSED LINE OF ATTACK

Instead of rewriting legacy applications or customizing them with middleware to connect to other applications one by one, this solution entails creating a "facade" for the legacy application. Other applications are easily "plugged into" this facade. By modeling a legacy application into its basic functions of create, read, update, and delete (CRUD), and then exposing these functions through the HTTP communications protocol and an XML-based SOAP interface, the Web service facade solution allows other applications to access legacy data by making use of common Web services through standardized protocols. A consistent and simple interface with consolidated security makes for easier integration. This allows developers to focus on business logic rather than replicating existing functionality.

In the proposed solution, new client applications can interact directly with the Web service without making use of the Client Helper. Existing client applications can utilize the Client Helper to minimize the changes required for them to access Web services. Since the Client Helper is a .NET class, its use of the Web service is seamless and easy to implement. Also, accessing it is easy, and the skills required to do so would transfer to other development efforts within the Enterprise. Another primary advantage of this solution is that if a change occurs in the interface used to access the legacy application, only the Adapter would require modification. Isolating change and providing a simple common interface for all of your Client applications provides a major

productivity benefit in the application interface development efforts. The possible limitation of this architecture is that such a simplified form of access may not support all of the functionality of more complex legacy applications.

3 ARCHITECTURE OF THE SYSTEM

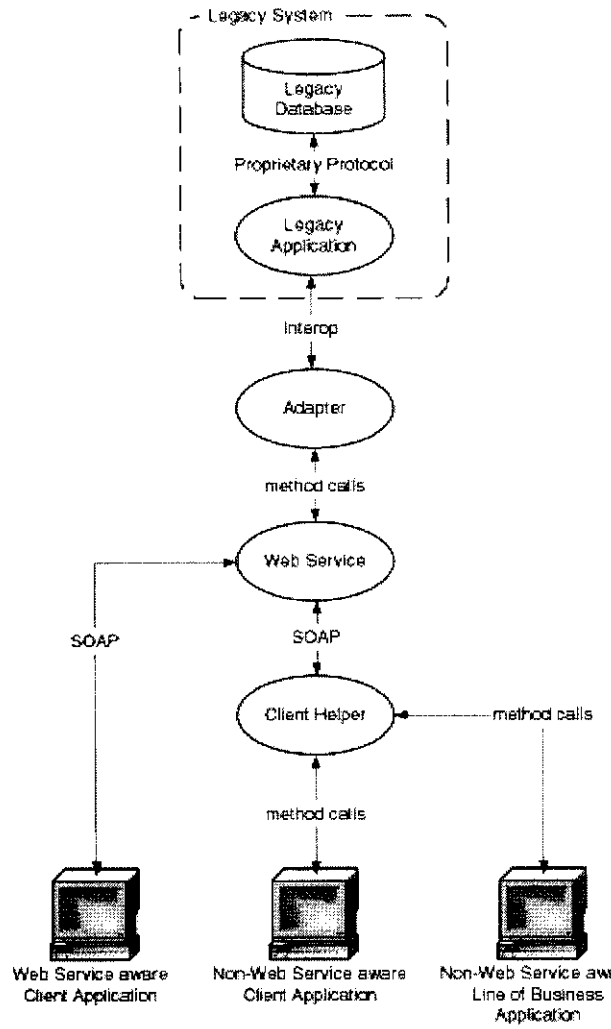


Figure 3.1 – Proposed System Architecture

4 SYSTEM DESIGN

The system is composed of three phases namely

1. Development of legacy system
2. Development of front end
3. Development of web service

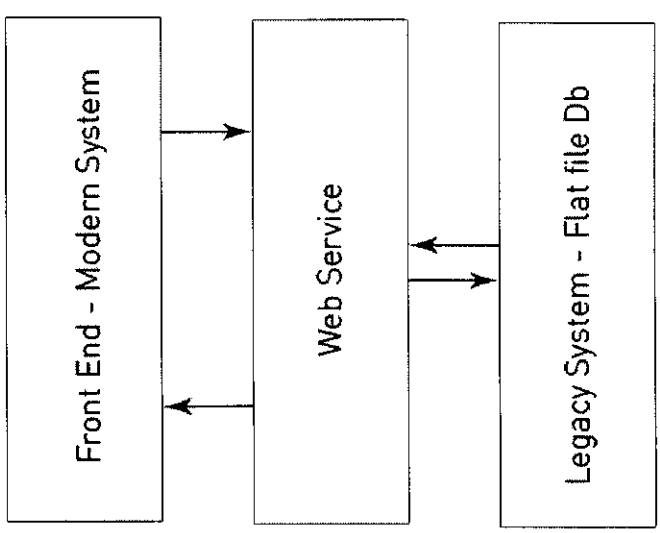


Figure 4.1: Interactions between various elements of the system

LIBRARY

1. Development of Legacy System

This phase involves the study and development of flat file database as a legacy database. It also involves the development of the various classes that create, update or delete the various entries of the flat file database. This is the equivalent of the legacy system the aircraft service provider. The various details of the number of free tickets and the corresponding flight number are stored based upon the date of transport. The date of transport is based as the name of each legacy database(flat file database). Some of the important methods in the legacy system invocation are:

1	createDate()
2	createOrModifyEntry()

Table 4.1 Important methods in legacy application

2. Development of Front End

The front end incorporates various security capabilities. It also proves as a front end to the web service. The front end is connected to the front end database which contains the various details about customers, flight details of the various flights etc. It also incorporates the user's GUI for the various interfaces to reserve, cancel and view issued tickets.

The important functions used are summarized below:

1. Welcome
2. Check Ticket
3. Reservation
4. Login
5. Issued Tickets

List 4.2 Important functions in front end

3. Development of Web Service

The Web Service is basically composed of components that are used to access the legacy system. The basic components of the web service are two fold. They are to read from or write into the file from the legacy database and to wrap it into the XML which transports it across to the front end which extracts it after the required transaction is performed. The web service acts as the intermediate between both these systems and forms the more intrinsic part of the project. An individual GUI is further developed for the CheckBook Web Service so as the independently portray the dual functionality of the web service.

The Web Service helps in the implementation of the reservation and the cancellation functionalities that form the major components of the front end. These are in turn altered in the back end of the legacy system as when changes in the front end are requested.

The important methods in the Web Service are summarized below:

S	
1	readFile()
2	writeFile()
3	getFileByName()

Table 4.3 Important methods in the Web Service

Programming Environment

Hardware Requirements:

1. Processor: Intel Pentium 4 Processor
2. Processor Speed: 1.8 GHz
3. Hard Disk: 40 GB IBM HDD
4. Memory: 256 MB SD RAM
5. Operating System: Windows 98, Windows XP
6. Backup Devices: 3 ½ inch floppy drive, Maxell Floppy Disk, Samsung CD Writer Drive, Sony Optical Rewritable Disk (4x)

Software Requirements:

1. Visual Studio .NET 2003
2. Flat File Database
3. Visual Studio 7
4. .NET Framework 1.1
5. SQL Server 7

6 DETAILED DESIGN

This section deals with a detailed explanation of the various classes used by this application.

6.1 Devlopment of Legacy System

6.1.1 Purpose

The Legacy System has the following purposes:

1. Existing flight details in the legacy system
2. Details that can be manually inserted into the legacy database from the legacy front end.
3. Exhibiting sample flat file legacy system.

6.1.2 Members

The legacy system is basically composed of the legacy database and the front end of the legacy system. The basic legacy database is a flat file database. It is composed of files which represent dates of the date of travel. These correspond to the parent company's flight details for that one particular date. This legacy system is generally not accessible by the front end of the modern databases. The legacy application may be used to create a new date or edit and change the details of the existing date.

6.1.3 Methods

The methods within this phase are summarized in Table 5.3

Method	Purpose
createDate()	Used to create a date file for the particular date of travel. This creates a text file based on the specified date that is passed as a parameter to the createDate() method.
createOrModifyEntry()	Used to create or edit details of a text file which has the details of the aircraft travel which corresponds to the date specified in the file. The various flight numbers are entered here along with the number of free tickets that may be available. The details are either added or modified or deleted based on the number of tickets availble in the particular set of aircrafts on the specified date.

Table 6.4 Methods in Legacy Application

6.2 Development of Front End :

6.2.1 Purpose

The Front end serves the following purposes:

1. To provide the GUI for the modern application
2. To enable checking, reserving, cancelling of tickets
3. To enable viewing of issued tickets

6.2.2 Functions

The functions within this phase are summarized in Table 5.3

Method	Purpose
Welcome	To provide the welcome screen wherein the user can select from the given list of options to either check availibility of tickets, reserve tickets, cancel tickets or view issued tickets.
Check Ticket	To check if the number of tickets are availible in the given aircraft on the specified day. This in turn invokes the web service which performs the given operation.
Reservation	To reserver the number of tickets, if availible in the given aircraft on the

	specified day. This in turn invokes the web service which performs the given operation.
Login	To authenicate the user by checking if he/she is authorised. This process is carried out by cross checking with the password that is associated with the user in the front end database that is used to store the user details along with the user's name and password.
Issued Tickets	To display the tickets that have already been booked and issued.

Table 6.4 Functions of Front End

6.3 Development of the Web Service

6.3.1 Purpose

The Web Service serves the following purposes:

1. To interact with the legacy system and extract required parameters.
2. To transport via XML the parameters that should be passed on to the front end.
3. To query the legacy system with the queries that are posed by the front end.

6.3.3 Methods

The methods within this class are summarized in Table 6.6

Method	Description
readFile()	Reads the flat file present in the legacy database in the legacy system and extracts the required information and wraps it and formulates the XML code which transports the data to the front end independently. This data is then further extracted by the Front end which passes this onto the front end.
writeFile()	Writes into the flat file present in the legacy database in the legacy system after getting the information from the front end and posts the returned information and wraps it and formulates the XML code which transports the data to the front end independently. This data is then further extracted by the Front end which passes this onto the front end.
getFileByName()	This a minor method which extracts the name of the flat file based on the

	date specified in the front end. The basic functionality is to perform a format conversion to the required format in which the flat file can be opened.

Table 6.6 Methods of CheckBook Web Service

7 FUTURE ENHANCEMENTS

The application has been built with minimum functionalities, with the intention of demonstrating the working of the concept of web services. In the future, enhancements can be made to the User Interface. The following features can be included to the User Interface:

1. Enable the user to view the actual back end database in a separate window, make changes to the actual back end legacy database directly.
2. Enable the user to have a variety of legacy databases as back ends (viz. different flight modules).
3. Improvising security to all levels and not restrict it only to the user login and ssl authentications alone.
4. Multiple user handling in front end along with concurrency issues.

8 CONCLUSION

The implementation of seam-less interconnection between legacy systems using web services has proven to be successful. The approach proves to be efficient for simple and semi-complex databases and applications. The interface has been made simple enough so that the user can work easily with the application.

The project was a very challenging one to work with. This project presented a lot of hidden difficulties in terms of logic as well as implementation. This project proved to be mentally stimulating and intellectually satisfying. It gave a proper insight into the difficulties in the field of middle ware development. It also threw light in the most recent research currently being pursued in the same field.

On the whole, the project was very exciting and proved to be truly rewarding working on it.

9 APPENDIX

9.1 SAMPLE CODE

CheckTicket.aspx.vb:

```
#Region " Imports "
Imports Microsoft.ApplicationBlocks.Data
Imports System.Data.SqlClient
Imports System.Text
Imports System.Web.Services

#End Region

Public Class CheckTicket
    Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()

    End Sub
    Protected WithEvents Calendar1 As
System.Web.UI.WebControls.Calendar
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
    Protected WithEvents DateTravel As
System.Web.UI.WebControls.TextBox
```

```
    Protected WithEvents LblMessage As
System.Web.UI.WebControls.Label
    Protected WithEvents AirLine As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents ALInfoGrid As
System.Web.UI.WebControls.DataGrid
    Protected WithEvents ALInfoGrid1 As
System.Web.UI.WebControls.DataGrid
    Protected WithEvents Reserve_Btn As
System.Web.UI.WebControls.Button
    Protected WithEvents Reserve_ As System.Web.UI.WebControls.Button
    Protected WithEvents NoOfTickets As
System.Web.UI.WebControls.TextBox


    'NOTE: The following placeholder declaration is required by the Web
Form Designer.
    Private designerPlaceholderDeclaration As System.Object

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        InitializeComponent()
    End Sub
    Protected WithEvents LblMessage1 As
System.Web.UI.WebControls.Label
```

```
#End Region

    Public AgentName As String
    Dim Message As String
    Dim myConnection As New SqlConnection
    Dim Ds As New DataSet
    Dim Dv As DataView
    Dim WebSr As New TicketBook.Service1


    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        If Session("Validated") = "True" Then

            AgentName = Session("AgentNAme")

            myConnection.ConnectionString =
(ConfigurationSettings.AppSettings("ConnectionString"))
            myConnection.Open()

            If IsPostBack Then
                If Request.Form("CallCheck") = "True" Then
                    Message = WebSr.ReadFile(Request.Form("FlightNo"),
DateTravel.Text().Trim)
                    If IsNumeric(Message) Then
                        If Message = 0 Then
                            Message = "No seats available."
```

```
                Else
                    Message = Message & " Seats(s) available.<a
href=javascript:Reservation_sub();>Reserve</a>"
                End If
              End If
            End If
            If Request.Form("CallReserve") = "True" Then
              If NoOfTickets.Text <> "" And NoOfTickets.Text <= 0 Then
                Message = "No Of Tickets is Invalid"
              Else
                Message = WebSr.WriteFile(Request.Form("FlightNo"),
DateTravel.Text().Trim, NoOfTickets.Text(), "")
                If Message = "Done" Then
                  Message = " Seats(s) Blocked. <a
href=BookTickets.aspx?S=" & NoOfTickets.Text() & "&F=" &
Request.Form("FlightNo") & "&D=" & DateTravel.Text().Trim & ">Issue
Tickets </a>"
                End If
              End If
            End If
            If Request.Form("CallCancel") = "True" Then
              If NoOfTickets.Text <> "" And NoOfTickets.Text <= 0 Then
                Message = "No Of Tickets is Invalid"
              Else
                Message = WebSr.WriteFile(Request.Form("FlightNo"),
DateTravel.Text().Trim, NoOfTickets.Text(), "True")
                If Message = "Done" Then
```

```
                    Message = " Seats(s) Canceled."
                End If
            End If
        End If
        If AirLine.SelectedValue <> "" Then
            getFlightDetail(AirLine.SelectedValue)
        End If
    Else
        DateTravel.Text = Date.Today.Month & "-" & Date.Today.Day &
"-" & Date.Today.Year
        getAirlinedata()
    End If
    LblMessage1.Text = Message
    myConnection = Nothing
  Else
    Response.Redirect("Login.aspx")
  End If
End Sub

Private Sub getAirlinedata()

  Dim LiList As ListItem

  Dim cmd As New SqlCommand("SELECT * FROM Airlines ",
myConnection)
  Dim Dr As SqlDataReader = cmd.ExecuteReader()
```

```
        AirLine.Items.Add("")
        Do While Dr.Read()
            LiList = New ListItem
            LiList.Value = Dr.Item("airlineID").ToString()
            LiList.Text = Dr.Item("airlineName").ToString()
            AirLine.Items.Add(LiList)
        Loop

        Dr.Close()

    End Sub
    Private Sub getFlightDetail(ByVal ALID As Integer)

        Dim StrSql As String
        StrSql = "select FlightNo,FlightName, C1.City_Name as From_city,
C2.City_Name as To_City, FlightType from flightInfo, Cities C1, cities C2 "
        StrSql = StrSql & "where airlineID = " & ALID & " And "
        StrSql = StrSql & "C1.City_ID = From_city and C2.city_ID = To_city"

        Dim SQlAdp As New SqlDataAdapter(StrSql, myConnection)
        SQlAdp.Fill(Ds)
        If Ds.Tables(0).Rows.Count > 0 Then
            ALInfoGrid.DataSource = Ds
            ALInfoGrid.DataBind()
            ALInfoGrid.Visible = True
            Dv = New DataView(Ds.Tables(0))
        Else
```

```
            ALInfoGrid.DataSource = Nothing
            ALInfoGrid.Visible = False
            Message = "No Details available for the selected flight."
        End If
        Ds = Nothing
        SQlAdp = Nothing
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Calendar1.Visible = True
    End Sub

    Private Sub Calendar1_SelectionChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Calendar1.SelectionChanged
        Dim trDate As Date
        trDate = Calendar1.SelectedDate
        DateTravel.Text = trDate.Month & "-" & trDate.Day & "-" &
trDate.Year
        Calendar1.Visible = False
    End Sub
#Region "DataGrid Functionality"

    Public Sub FormatColumn(ByVal objSender As Object, ByVal objArgs
As DataGridItemEventArgs)
```

```
    Dim objItem As DataGridItem = CType(objArgs.Item, DataGridItem)
    Dim objItemType As ListItemType = CType(objArgs.Item.ItemType,
ListItemType)
    Dim tbiStyle As TableItemStyle = New TableItemStyle    ' used for
customized HeaderStyles

    If objItemType = ListItemType.AlternatingItem Or objItemType =
ListItemType.Item Then
        Dim objRowView As DataRowView =
CType(objArgs.Item.DataItem, DataRowView)
        Dim FLNo As String = objRowView("FlightNo")

        Dim objLabel As Label =
CType(objArgs.Item.FindControl("lblcheckTicket"), Label)
        objLabel.Visible = True
        If Request.QueryString("Reserve") = "True" Then
          objLabel.Text = "<a href=javascript:frm_sub_res('" & FLNo &
"');>Reserve</a>" & " " & "<a href=javascript:Cancel_Reserve_sub('" &
FLNo & "');>Cancel</a>"
        Else
          objLabel.Text = "<a href=javascript:frm_sub_chk('" & FLNo &
"');>Check</a>"
        End If
    End If
    objItem = Nothing
    objItemType = Nothing
    tbiStyle = Nothing
```

```
  End Sub


  Public Sub SortGrid(ByVal source As Object, ByVal e As
System.Web.UI.WebControls.DataGridSortCommandEventArgs)
    Dim strSORT As String

    strSORT = e.SortExpression.ToString
    Dv.Sort = strSORT
    ALInfoGrid.DataSource = Dv
    ALInfoGrid.DataBind()

  End Sub
#End Region

End Class
```

BookTickets.aspx.vb:

```
#Region " Imports "
Imports Microsoft.ApplicationBlocks.Data
Imports System.Data.SqlClient
Imports System.Text
Imports System.Web.Services
#End Region

Public Class BookTickets
  Inherits System.Web.UI.Page
```

```
#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()

    End Sub
    Protected WithEvents Res_AirLine As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Res_DateTravel As
System.Web.UI.WebControls.TextBox
    Protected WithEvents Res_NoOfTickets As
System.Web.UI.WebControls.TextBox
    Protected WithEvents Res_LblMessage1 As
System.Web.UI.WebControls.Label
    Protected WithEvents Res_Calendar1 As
System.Web.UI.WebControls.Calendar
    Protected WithEvents Res_ALInfoGrid As
System.Web.UI.WebControls.DataGrid
    Protected WithEvents Pass1 As
System.Web.UI.WebControls.DropDownList

    'NOTE: The following placeholder declaration is required by the Web
Form Designer.
    Private designerPlaceholderDeclaration As System.Object
```

```
    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        InitializeComponent()
    End Sub

#End Region
    Public AgentName As String
    Public I As Integer
    Public PrintTkt As String
    Public TicketCnt As Integer
    Public FlightNo As String
    Public FlightDate As String
    Dim myConnection As New SqlConnection
    Dim Ds As New DataSet
    Dim Dv As DataView
    Dim SQL As String
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim J As Integer ' loop counter
        Dim ErrorOccured As Boolean
        If Session("Validated") = "True" Then
            AgentName = Session("AgentNAme")
            myConnection.ConnectionString =
(ConfigurationSettings.AppSettings("ConnectionString"))
            myConnection.Open()
            If IsPostBack Then
```

```
            For J = 1 To Request.Form("TktCnt")
                If InsertRecord(Request.Form("FName" & J),
Request.Form("LName" & J), Request.Form("Addr1" & J),
Request.Form("Addr2" & J), Request.Form("City" & J),
Request.Form("Pin" & J), Request.Form("FNo"), Request.Form("FDt")) =
"Error" Then
                    ErrorOccured = True
                End If
            Next
            If ErrorOccured = False Then
                PrintTkt = "true"
            End If
        End If
        myConnection = Nothing
    Else
        Response.Redirect("Login.aspx")
    End If
  End Sub

  Private Function InsertRecord(ByVal FName As String, ByVal LName
As String, ByVal Addr1 As String, ByVal Addr2 As String, ByVal City As
String, ByVal Pin As String, ByVal FNo As String, ByVal FDt As String)
As String
    Dim StrSql As String
    Dim Result As String
    Dim SQlCmd As New SqlCommand
    InsertRecord = ""
```

```
        StrSql = "insert into Passengers values ('" & FName & "','" & LName &
"','" & Addr1 & "','" & Addr2 & "','" & City & "','" & Pin & "')"
        SQlCmd.Connection = myConnection
        SQlCmd.CommandType = CommandType.Text
        SQlCmd.CommandText = StrSql
        Result = SQlCmd.ExecuteNonQuery()

        SQlCmd.CommandText = "select @@IDentity"
        Result = SQlCmd.ExecuteScalar()
        If Result > 0 Then
            StrSql = "insert into Ticket_Booking values (" & Result & "," & FNo
& ",'" & FDt & "','1','0','Confirmed')"
            SQlCmd.CommandText = StrSql
            Result = SQlCmd.ExecuteNonQuery()
            If Result < 0 Then
                InsertRecord = "Error"
            End If
        End If
    End Function
End Class
```

CheckBook.asmx.vb-Web Service:

```
Imports System.Web.Services
Imports System.IO
Imports System.Text
```

```
<System.Web.Services.WebService(Namespace :=
"http://tempuri.org/WebService1/Service1")> _
Public Class Service1
    Inherits System.Web.Services.WebService

#Region " Web Services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web Services Designer.
        InitializeComponent()

        '
    End Sub

    'Required by the Web Services Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Web Services
Designer
    'It can be modified using the Web Services Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        components = New System.ComponentModel.Container()
    End Sub
```

```
    Protected Overloads Overrides Sub Dispose(ByVal disposing As
Boolean)
        'CODEGEN: This procedure is required by the Web Services Designer
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region

    'Reads a Text File
    <WebMethod()> _
    Public Function ReadFile(ByVal FlightCode As String, ByVal FLDate As
Date) As String
        Dim objStreamReader As StreamReader
        Dim strLine As String
        Dim strNewLine As String
        Dim FL_Avail As String
        Dim FileName As String

        FileName = getFileName(FLDate)
        If FileName = "BadDate" Then
            FL_Avail = "Enter a valid date."
            Return FL_Avail
```

```
End If

If File.Exists(FileName) Then ' check file exist
   objStreamReader = New StreamReader(FileName) 'open the file

   'Read the first line of text.
   strNewLine = "Start"
   strLine = ""
   FL_Avail = ""

   'Continue to read until you reach the end of the file.
   Do While Not strNewLine Is Nothing
      strNewLine = ""
      strNewLine = objStreamReader.ReadLine
      'do the string operation
      If strNewLine <> "" Then
         If strNewLine.Substring(0, 2).Trim.ToUpper =
FlightCode.Trim.ToUpper Then
            FL_Avail = strNewLine.Substring(3, 3).Trim
            objStreamReader.Close()
            Return FL_Avail
         End If
      End If
   Loop

   'Close the file.
   objStreamReader.Close()
```

```
        If FL_Avail.Trim() = "" Then
            FL_Avail = "Flight not found"
        End If
    Else 'file doesnt exist
        FL_Avail = "Reservation not started"
    End If


    Return FL_Avail
End Function
' Updates a textFile
<WebMethod()> _
Public Function WriteFile(ByVal FlightCode As String, ByVal FLDate
As Date, ByVal TicketsNeeded As String, ByVal Cancel As String) As
String
    Dim objStreamWriter As StreamWriter
    Dim objstreamReader As StreamReader
    Dim Temp_file As String
    Dim Real_File As String
    Dim strNewLine As String
    Dim TicketAvail As String  ' Old Availability
    Dim NewAvail As String     ' New  Availability

    Real_File = getFileName(FLDate)
    If Real_File = "BadDate" Then
        WriteFile = "Enter a valid date."
        Return WriteFile
```

```
End If

TicketAvail = ReadFile(FlightCode, FLDate)
If Cancel = "True" Then
   NewAvail = Int(TicketAvail) + Int(TicketsNeeded)
Else
   If IsNumeric(TicketAvail) Then
      If TicketAvail - TicketsNeeded < 0 Then
         WriteFile = "Requested number of seat(s) not available."
         Return WriteFile
      Else
         NewAvail = Int(TicketAvail) - Int(TicketsNeeded)
      End If
   Else
      WriteFile = "Check seat availablity."
      Return WriteFile
   End If
End If

If Len(NewAvail) <= 0 Then
   NewAvail = "000"
ElseIf Len(NewAvail) = 1 Then
   NewAvail = "00" & NewAvail
ElseIf Len(NewAvail) = 2 Then
   NewAvail = "0" & NewAvail
ElseIf Len(NewAvail) > 3 Then
   NewAvail = Right(NewAvail, 3)
```

```
        End If

        Temp_file = Replace(Real_File, ".txt", "_Temp.txt")
        If File.Exists(Temp_file) Then 'check file exist
            WriteFile = "Reservation in progress, Try again later."
            Return WriteFile
        End If

        If File.Exists(Real_File) Then 'check file exist
            objStreamWriter = New StreamWriter(Temp_file) 'open the file to
write
            objstreamReader = New StreamReader(Real_File) 'open the file to
read

            'Read the first line of text.
            strNewLine = "Start"

            'Continue to read until you reach the end of the file.
            Do While Not strNewLine Is Nothing
                strNewLine = ""
                strNewLine = objstreamReader.ReadLine()
                If strNewLine <> "" Then
                    If strNewLine.Substring(0, 2).Trim.ToUpper =
FlightCode.Trim.ToUpper Then
                        strNewLine = strNewLine.Substring(0, 3) & NewAvail
                    End If
                    objStreamWriter.WriteLine(strNewLine)
```

```
            End If
        Loop

        'Close the files, update the real file, delete the temp file.
        objstreamReader.Close()
        objStreamWriter.Close()
        File.Delete(Real_File)
        File.Copy(Temp_file, Real_File)
        File.Delete(Temp_file)
        WriteFile = "Done"
      End If

      Return WriteFile
    End Function
    ' Makes the filename with the supplies date
    Private Function getFileName(ByVal FL_Date As Date) As String
      getFileName = "C:\Reservations\"
      If IsDate(FL_Date) Then 'check date
        getFileName = getFileName & FL_Date.Month() & "_" &
FL_Date.Day() & "_" & FL_Date.Year() & ".txt"
      Else ' not a Good Date
        getFileName = "BadDate"
        Return getFileName
      End If
    End Function
End Class
```

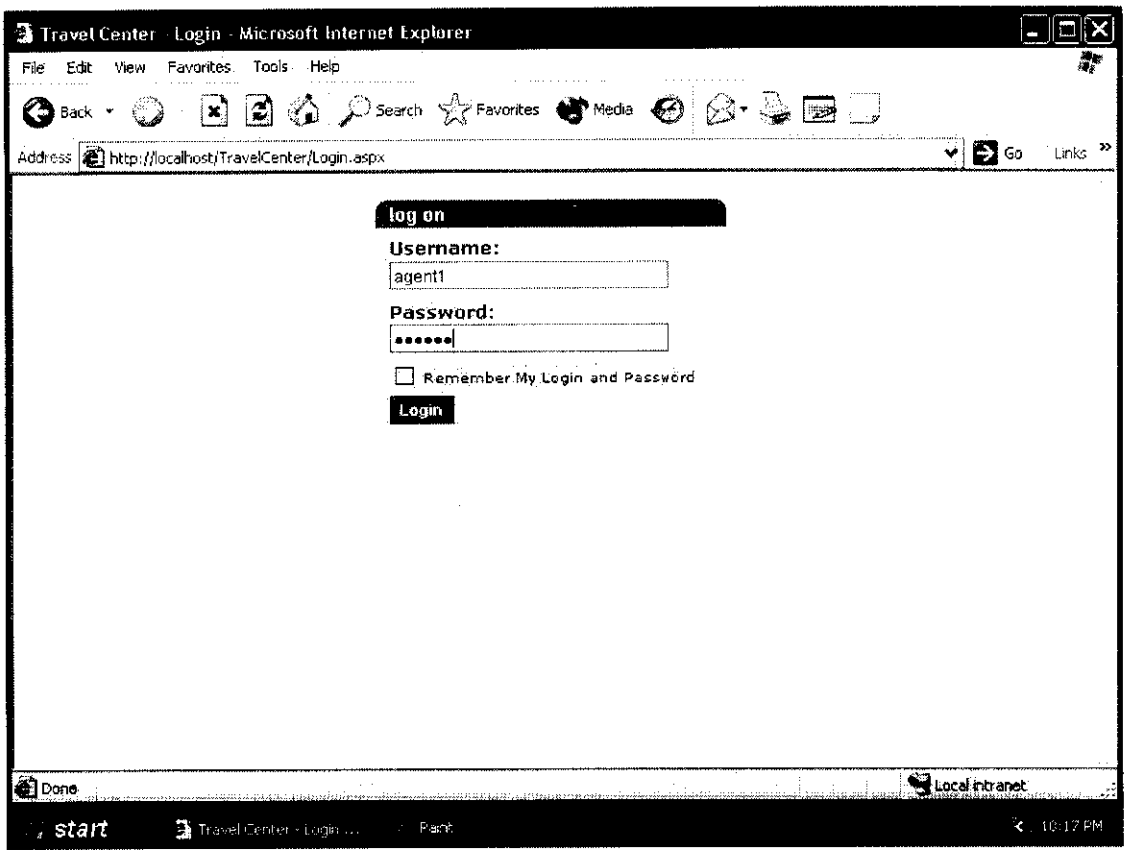
Welcome.aspx.vb:

```
Public Class welcome
   Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

   'This call is required by the Web Form Designer.
   <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()

   End Sub

   'NOTE: The following placeholder declaration is required by the Web
Form Designer.
   'Do not delete or move it.
   Private designerPlaceholderDeclaration As System.Object

   Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
      'CODEGEN: This method call is required by the Web Form Designer
      'Do not modify it using the code editor.
      InitializeComponent()
   End Sub

#End Region
   Public AgentName As String
   Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```

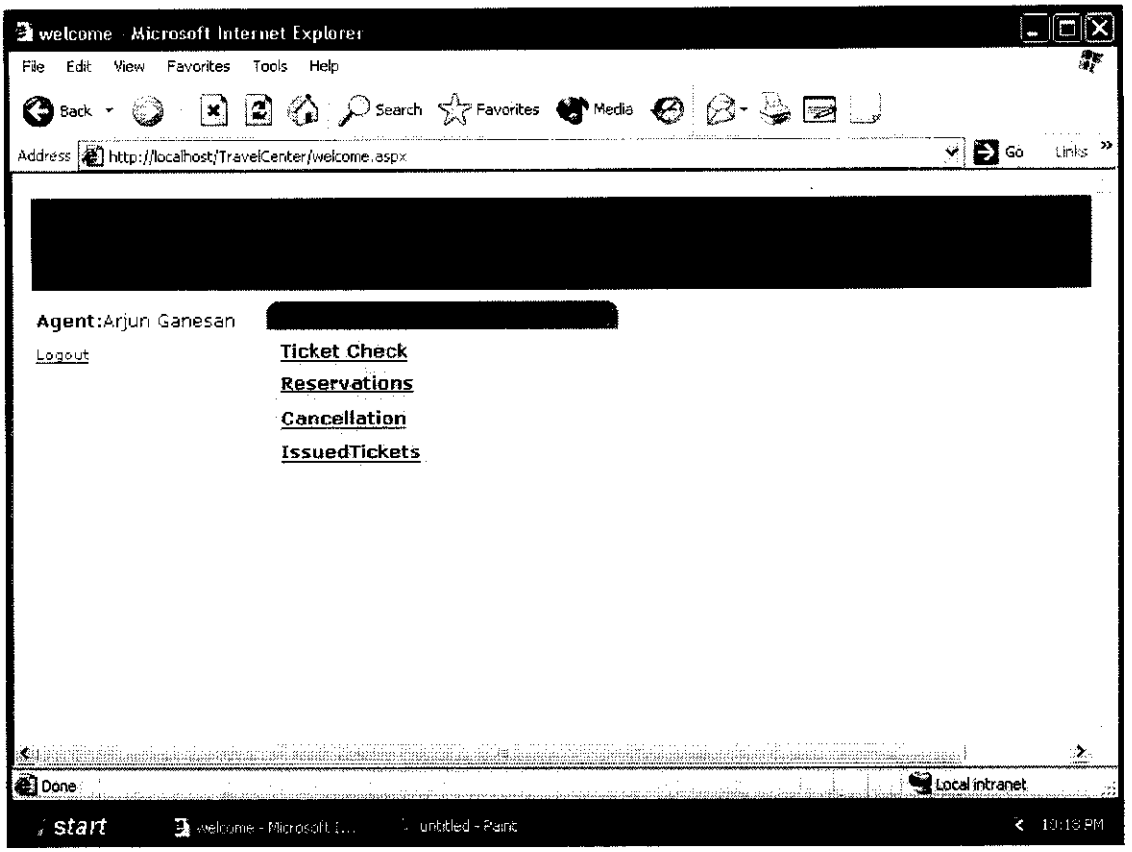
```
        If Session("Validated") = "True" Then
            AgentName = Session("AgentNAme")
        Else
            Response.Redirect("Login.aspx")
        End If
    End Sub

End Class
```

9.2 RESULTS

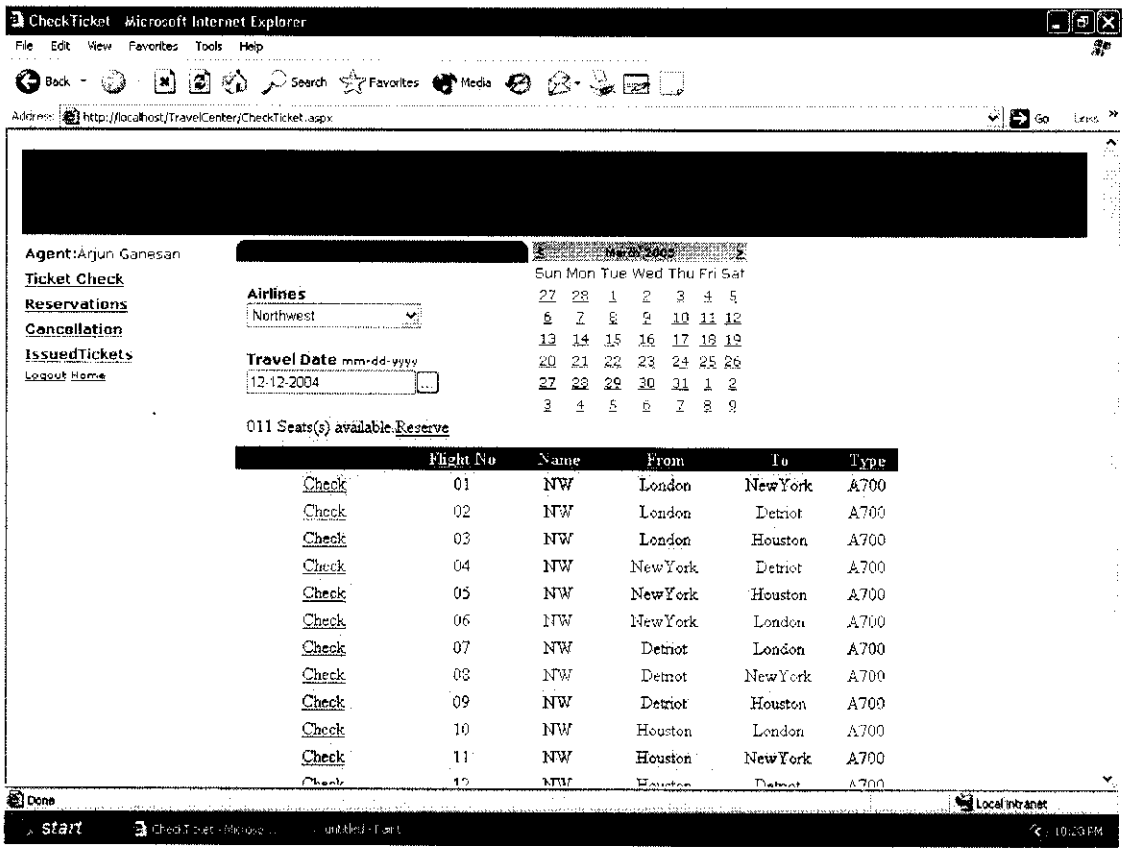
Login Screen:



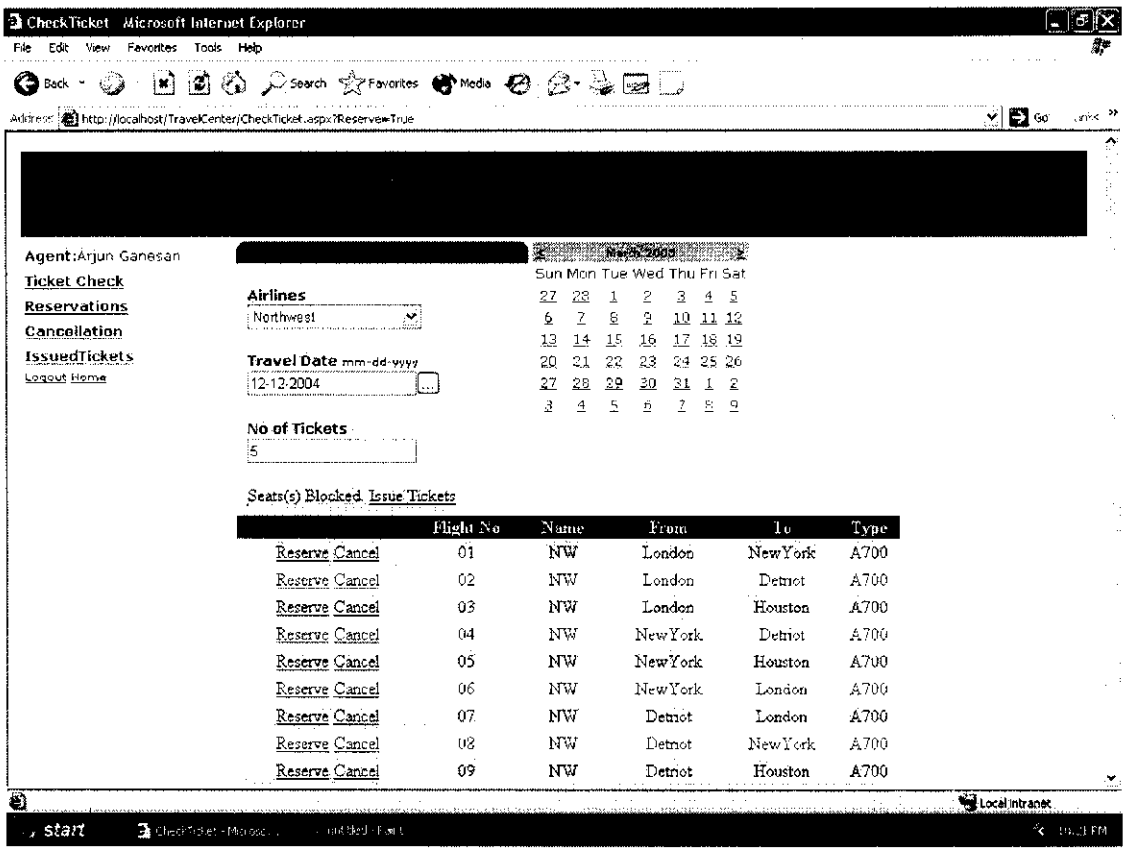
Welcome Screen :



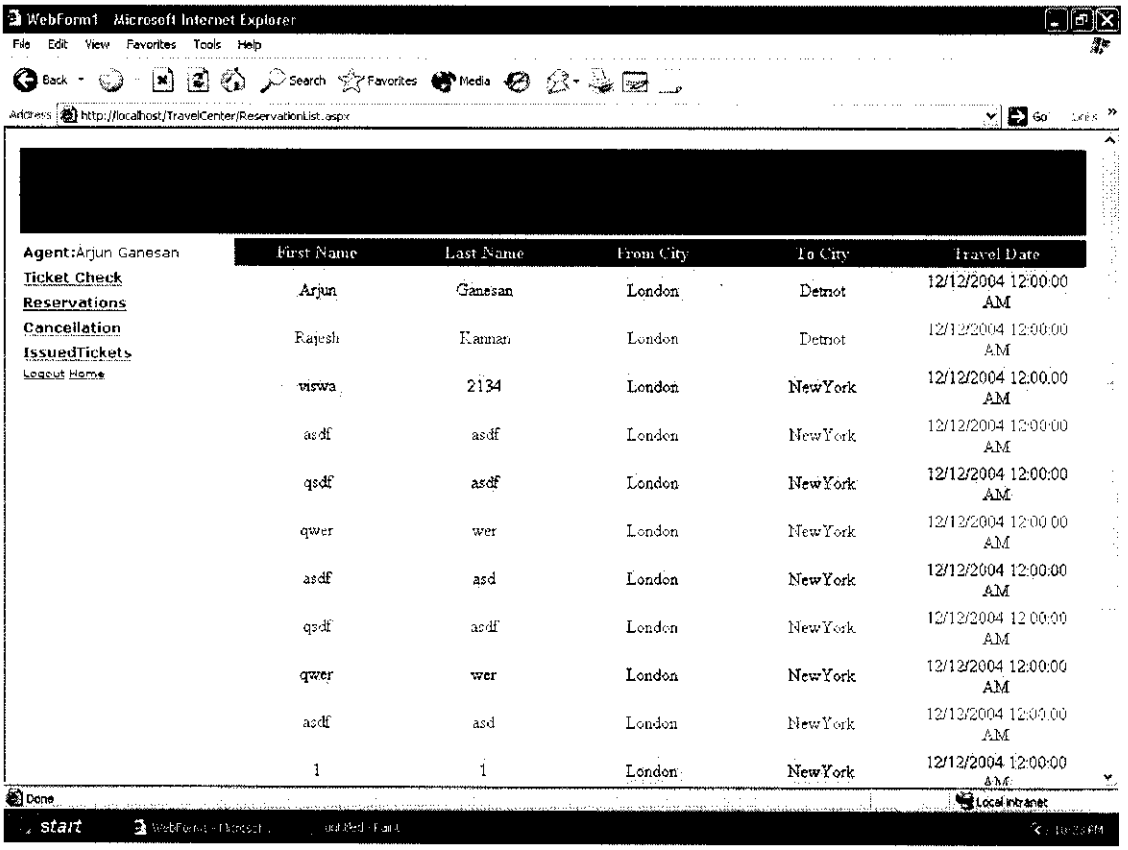
Check Ticket Instance :



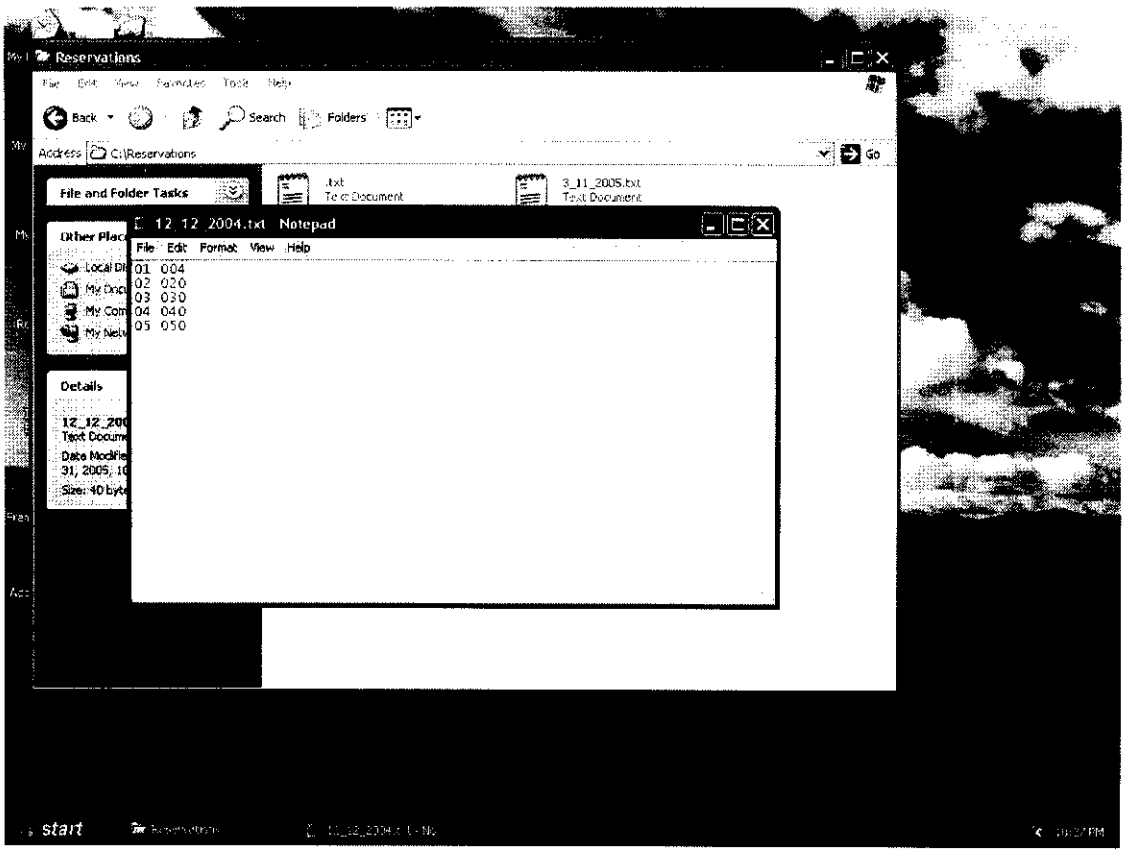
Reservation Instance :



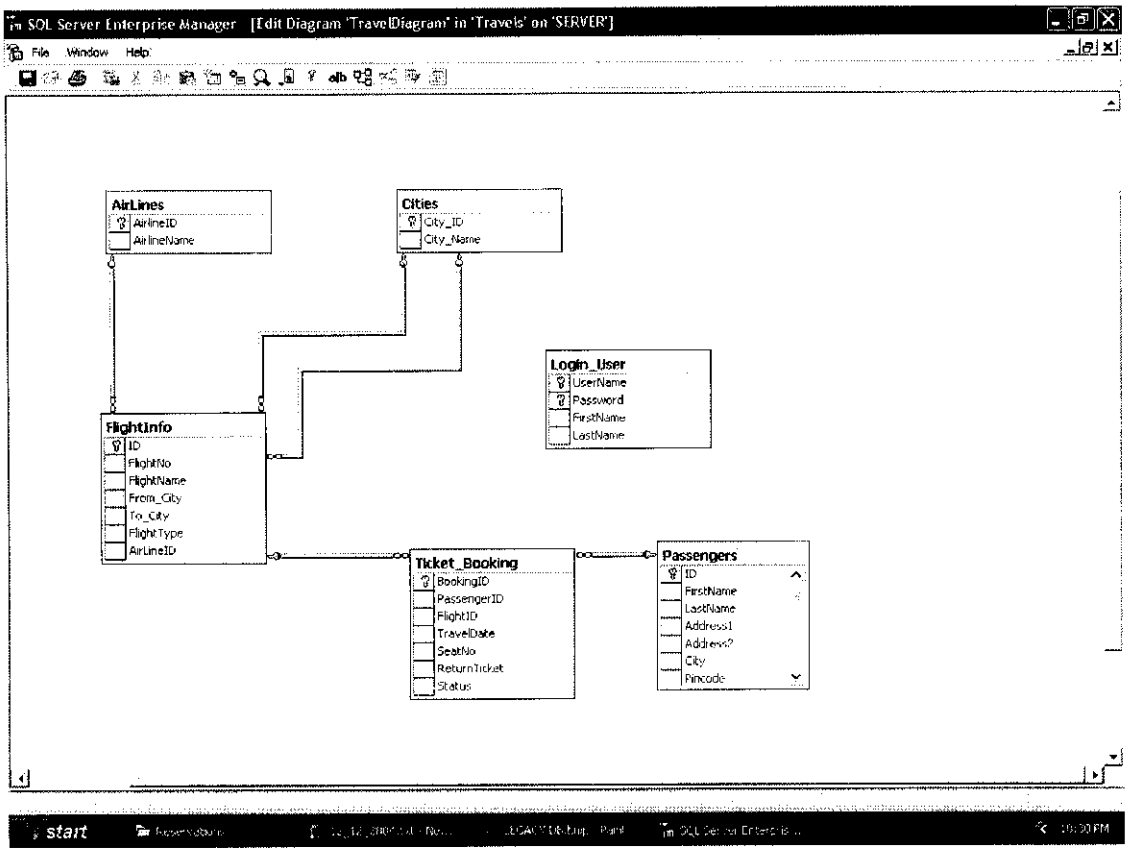
Issued Tickets :



Legacy Database :



Front End Database :



10 REFERENCES

1. CHRIS PAYNE (2002) 'Teach Yourself ASP.NET in 21 days', *4th Edition, SAMS publication.*

2. Harvey M. Deitel, 'Web Services: A Technical Introduction' , *4th edition, Addison Wellesley.*

3. DUNCAN MACKENZIE (2003) 'Yourself ASP.NET in 21 days', *SAMS publication.*

4. ETHAN CERAMI (2003) 'Web Services Essentials', *Oreilley.*

5. DAN BOX, 'Teach Yourself ADO.NET in 21 days', *SAMS publication.*