**PROJECT REPORT**

On

**“ULTIMATIX DIGITOX SYSTEM”**

Designed and Developed

By

**MR. RUCHIT MEHTA**

Sincere Thanks from,

(RUCHIT MEHTA)

# INDEX

|  |  |  |
| --- | --- | --- |
| **Sr.**  **No.** | **Contents** | **Page No.** |
| 1. | Project Title | 1 |
|  | Introduction | 4 |
|  | Scope of the project | 4 |
| 2. | Analysis | 5 |
| 3. | Design Phase | 6 |
|  | Detailed Life Cycle of project(Logical design) | 7 |
|  | Class Diagram | 8 |
|  | Flow Chart | 9 |
| 4. | Coding phase |  |
|  | Source Code | 10 |
| 5. | Testing Phase | 47 |
| 6. | Snapshots | 51 |
| 7. | Time Tracking Enhancements | 58 |
| 8. | References | 59 |

## INTRODUCTION

A ultimatix digitox system that acts as biometrics does. With real-time logging, this system makes it possible to record the time-in and time-out of a user's attendance with ease. It provides a user-friendly UI and shortcut keys and for fast processing of records and logging. It also has a picture for each user to ensure the user's identity when being logged.

The Employee Attendance Record System is a simple project which allows to track and store the employees' time log for a specific company. This project has two sides of user pages, one is the Administrator page which is where all the data and functionalities of the project will be found and be operated except for logging the time records of the employee and the one is the client page which where employees log their daily time records..

### SCOPE

* This app is good for school, college and office where in they maintain a huge record for attendance, as it save time and hardawre functionality.
* It doesnt require any machine base hardware Any number of user can connect to this platform

### MODULES

* Registered User (Admin)
* Visitors (New user)

### HARDWARE SPECIFICATIONS

* Operating System: Windows 7/XP/8/10
* RAM: 2GB or higher
* Processor: 2GHz or optimal
* Disk Space: Minimum 1GB

### SOFTWARE SPECIFICATIONS

✔ Microsft Visual Studio 2010 or higher version.

✔ Microsft Access Database Engine 2010 / ADO.Net.

## ANALYSIS

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to every minute detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

### MODULE FUNCTIONS

USER SIDE INCLUDES :

-User id

-Password

-Time in

-Time out

-Clear function

ADMIN SIDE INCLUDES :

-Admin login

-Admin id

-Password

-Shortcut key to login

### REQUIREMENT ANALYSIS

* Refines project goals into defined functions and operation of intended application.

A key question is :

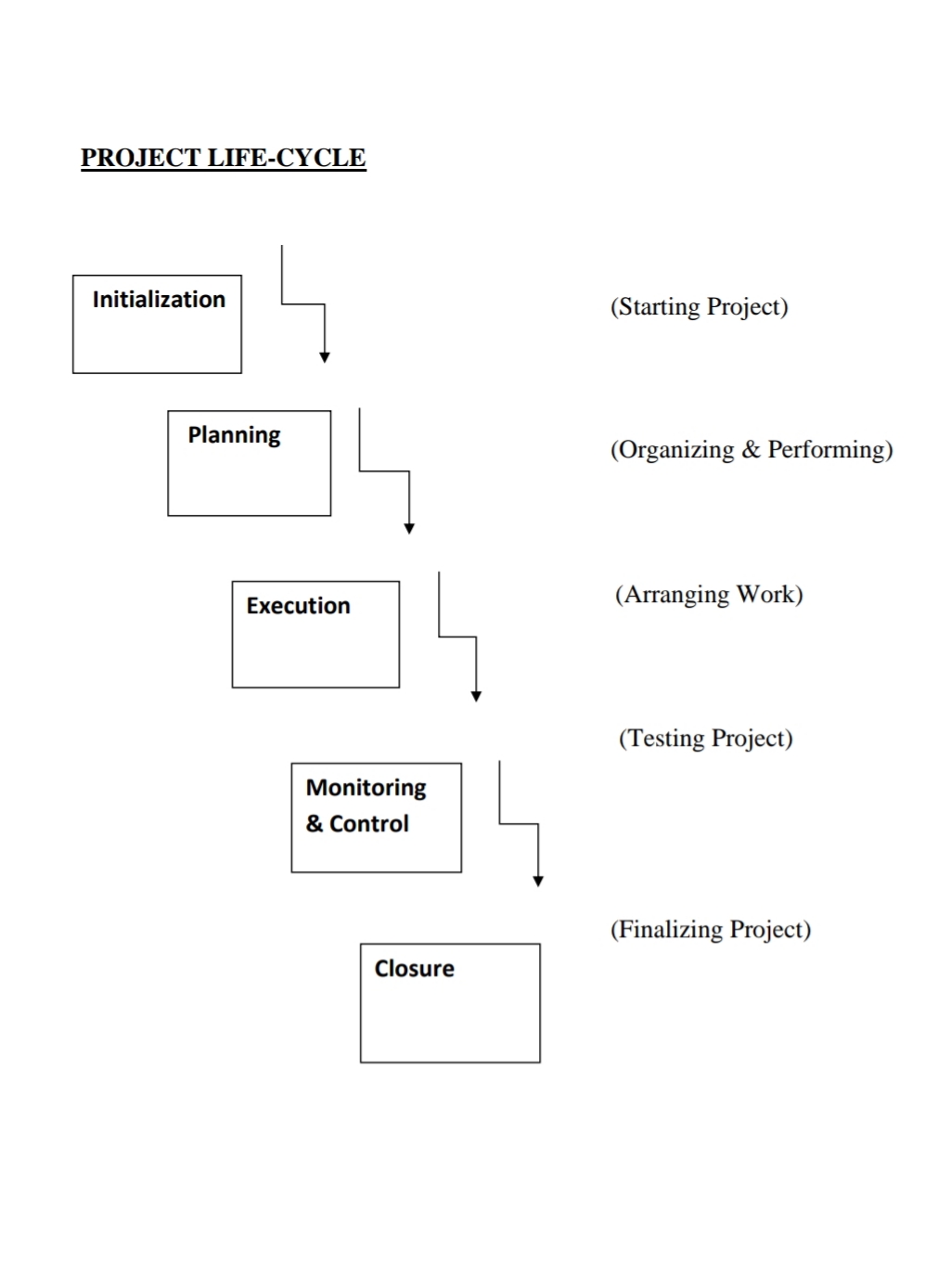
* 'What is needed for project and how the system will achieve its goal?'
* Once the analysis is complete, the developer has a firm understanding of what is to be done.
* The goal of the requirement specification phase is to produce the Software Requirement Specification (SRS) document

TOOLS AND TECHNOLOGY USED

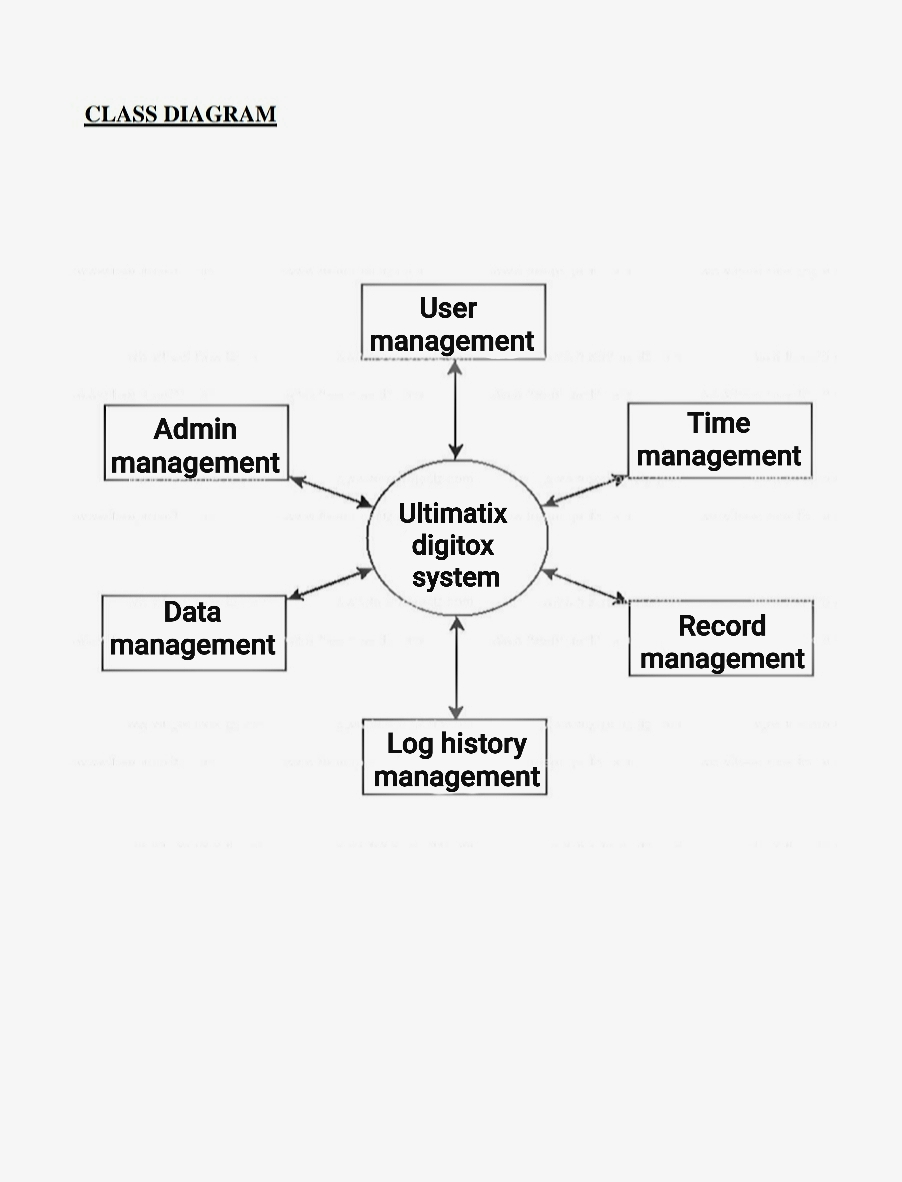
* Microsft Visual Studio 2010 or higher version.
* Microsft Access Database Engine / ADO.Net
* Any Desktop computer meeting the minimum system requirements

## DESIGN PHASE

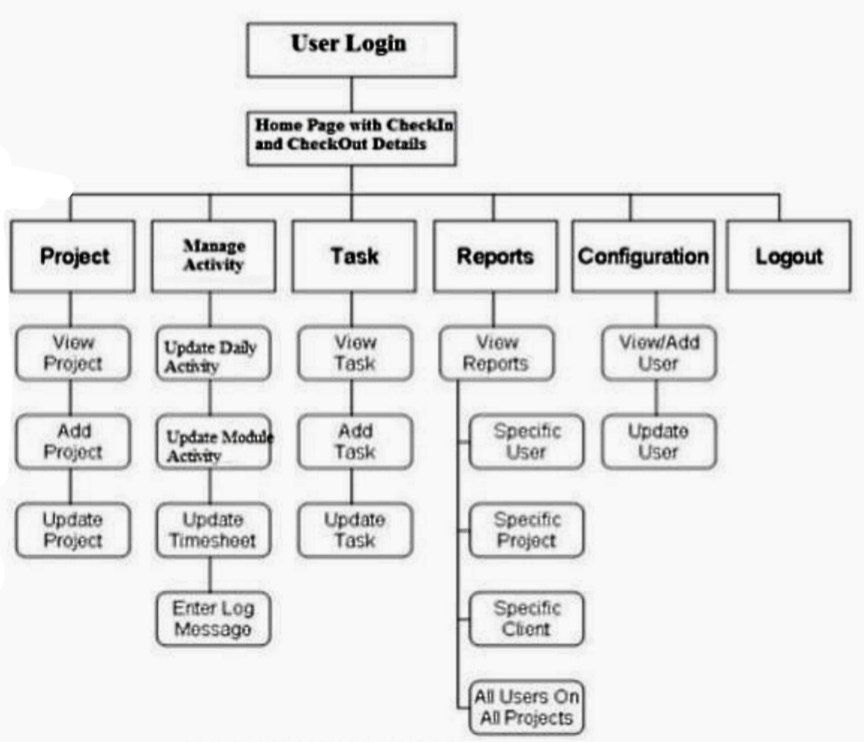
**PROJECT LIFE-CYCLE**



**CLASS DIAGRAM**



**FLOW CHART**



**How to Use**

First, the system admin must populate the user list along with some necessary information. When creating a new user, the password will be "password" by default and this can be updated afterward. In order for the user to log their time record, they just have to enter their user ID and password and the system itself will detect if the user has already a "time-in " record for the day, the system will show the "time out button" otherwise the "time in button" will be shown.

**SOURCE CODE**

**BackEnd.vb**

Imports System.Data

Imports System.Data.OleDb

Imports System.Math

Imports Microsoft.Office.Interop

Public Class Back\_End

#Region "FORM CLOSE and MINIMIZE BUTTON"

Private Sub PictureBox1\_MouseHover(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnclose.MouseHover

Btnclose.BackColor = Color.Gray

End Sub

Private Sub Btnclose\_MouseLeave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnclose.MouseLeave

Btnclose.BackColor = Color.Black

End Sub

Private Sub Btnmin\_MouseHover(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnmin.MouseHover

Btnmin.BackColor = Color.Gray

End Sub

Private Sub Btnmin\_MouseLeave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnmin.MouseLeave

Btnmin.BackColor = Color.Black

End Sub

Private Sub Btnclose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnclose.Click

Me.Close()

Front\_End.MdiParent = Main

Front\_End.Show()

End Sub

Private Sub Btnmin\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnmin.Click

Me.WindowState = FormWindowState.Minimized

End Sub

#End Region

#Region "Buttons Up"

Private Sub btnshutdown\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnshutdown.Click

End

End Sub

Private Sub btnlogoff\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnlogoff.Click

Me.Close()

Front\_End.MdiParent = Main

Front\_End.Show()

End Sub

Private Sub btnadmin\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnadmin.Click

frmadmin.ShowDialog()

End Sub

#End Region

#Region "Shortcuts"

Private Sub toollogoff\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles toollogoff.Click

btnlogoff\_Click(sender, e)

End Sub

Private Sub toolshutdown\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles toolshutdown.Click

btnshutdown\_Click(sender, e)

End Sub

Private Sub ToolUserlist\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ToolUserlist.Click

MainTab.SelectedTab = TabItem1

End Sub

Private Sub ToolUserform\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles ToolUserform.Click

MainTab.SelectedTab = TabItem2

End Sub

Private Sub ToolLogsHistory\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles ToolLogsHistory.Click

MainTab.SelectedTab = TabItem3

End Sub

Private Sub ToolAbout\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles ToolAbout.Click

MainTab.SelectedTab = TabItem4

End Sub

#End Region

#Region "Declaration"

'Public Conn As OleDbConnection

'TAB 1 Declaration

Dim constring As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" & Application.StartupPath & "\Dbase\UTIS.mdb;Persist Security Info=False"

Dim eDS0 As DataSet = New DataSet

Dim eDA0 As OleDbDataAdapter = New OleDbDataAdapter

Dim eCB0 As OleDb.OleDbCommandBuilder

Dim eDR0 As DataRow

Public iCountRecord As Integer

Public iCurrentRecord As Integer

Dim X As Integer

'TAB 2 Declaration

Public sConn0 As OleDbConnection

'TAB 3 Declaration

Dim dDS As DataSet = New DataSet

Dim dDA As OleDbDataAdapter = New OleDbDataAdapter

Dim dCB As OleDb.OleDbCommandBuilder

Dim dDR As DataRow

#End Region

Private Sub Back\_End\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

'TAB 1 LoAD

Load\_User()

Load\_Logs("SELECT \* FROM tblDTR order by date\_timein DESC")

'TAB 2 LOAD

If Me.dtgUser.SelectedRows.Count > 0 Then

Load\_Userform()

Read\_Only\_Controls()

btnnewuser.Text = "New"

btnedituser.Text = "Edit"

btnchangepass.Enabled = False

End If

'TAB 3 LOAD

MonthCalendar.TodayDate = Now

MonthCalendar.MaxDate = Now

End Sub

#Region "Tab 1 - User Procedure"

Private Sub Load\_User()

Dim conn As New OleDbConnection

Dim cmd As New OleDbCommand

Dim da As New OleDbDataAdapter

Dim dt As New DataTable

Dim sSQL As String = String.Empty

'try catch block is used to catch the error

Try

'get connection string declared in the Module1.vb and assing it to conn variable

conn = New OleDbConnection(constring)

conn.Open()

cmd.Connection = conn

cmd.CommandType = CommandType.Text

sSQL = "SELECT \* FROM tbluser"

cmd.CommandText = sSQL

da.SelectCommand = cmd

da.Fill(dt)

Me.dtgUser.DataSource = dt

If dt.Rows.Count = 0 Then

MsgBox("No record found!")

End If

Me.lbluser.Text = "Number of Users: " & dtgUser.RowCount

Catch ex As Exception

MsgBox(ErrorToString)

Finally

conn.Close()

End Try

End Sub

Function SearchUserid()

Dim da As OleDbDataAdapter

Dim dsetorder As New DataSet

Try

Using connstring As New OleDbConnection(constring)

If txtsearchuser.Text <> "" Then

da = New OleDbDataAdapter("SELECT \* " & \_

"from tbluser " & \_

"where user\_idno like '" & txtsearchuser.Text & "%' ", connstring)

dsetorder = New DataSet

da.Fill(dsetorder, "tbluser")

dtgUser.DataSource = dsetorder.Tables("tbluser").DefaultView

Me.lbluser.Text = "Number of Users: " & dtgUser.RowCount

Else

Call Load\_User()

End If

End Using

Catch ex As Exception

End Try

Return True

End Function

Function SearchUsername()

Dim da As OleDbDataAdapter

Dim dsetorder As New DataSet

Try

Using connstring As New OleDbConnection(constring)

If txtsearchuser.Text <> "" Then

da = New OleDbDataAdapter("SELECT \* " & \_

"from tbluser " & \_

"where user\_fname like '" & txtsearchuser.Text & "%' " & \_

"OR user\_lname like '" & txtsearchuser.Text & "%' " & \_

"OR user\_mname like '" & txtsearchuser.Text & "%' ", connstring)

dsetorder = New DataSet

da.Fill(dsetorder, "tbluser")

dtgUser.DataSource = dsetorder.Tables("tbluser").DefaultView

Me.lbluser.Text = "Number of Users: " & dtgUser.RowCount

Else

Call Load\_User()

End If

End Using

Catch ex As Exception

End Try

Return True

End Function

Function SearchUseraddr()

Dim da As OleDbDataAdapter

Dim dsetorder As New DataSet

Try

Using connstring As New OleDbConnection(constring)

If txtsearchuser.Text <> "" Then

da = New OleDbDataAdapter("SELECT \* " & \_

"from tbluser " & \_

"where user\_addr like '" & txtsearchuser.Text & "%' ", connstring)

dsetorder = New DataSet

da.Fill(dsetorder, "tbluser")

dtgUser.DataSource = dsetorder.Tables("tbluser").DefaultView

Me.lbluser.Text = "Number of Users: " & dtgUser.RowCount

Else

Call Load\_User()

End If

End Using

Catch ex As Exception

End Try

Return True

End Function

Function SearchUserpos()

Dim da As OleDbDataAdapter

Dim dsetorder As New DataSet

Try

Using connstring As New OleDbConnection(constring)

If txtsearchuser.Text <> "" Then

da = New OleDbDataAdapter("SELECT \* " & \_

"from tbluser " & \_

"where user\_pos like '" & txtsearchuser.Text & "%' ", connstring)

dsetorder = New DataSet

da.Fill(dsetorder, "tbluser")

dtgUser.DataSource = dsetorder.Tables("tbluser").DefaultView

Me.lbluser.Text = "Number of Users: " & dtgUser.RowCount

Else

Call Load\_User()

End If

End Using

Catch ex As Exception

End Try

Return True

End Function

#End Region

#Region "Tab 2 - User Form Procedure"

Private Sub Load\_Userform()

Dim conn As New OleDbConnection

Dim cmd As New OleDbCommand

Dim dr As OleDbDataReader

Dim arrImage() As Byte

Dim myMS As New IO.MemoryStream

Try

conn = New OleDbConnection(constring)

conn.Open()

cmd.Connection = conn

cmd.CommandType = CommandType.Text

cmd.CommandText = "select \* from tblUser where user\_idno = '" & Me.dtgUser.Item(0, Me.dtgUser.CurrentRow.Index).Value & "'"

dr = cmd.ExecuteReader

If dr.HasRows Then

While dr.Read

Me.txtuseridno.Tag = IIf(Not IsDBNull(dr("user\_idno")), dr("user\_idno"), "")

Me.txtuseridno.Text = IIf(Not IsDBNull(dr("user\_idno")), dr("user\_idno"), "")

Me.lblpass.Text = IIf(Not IsDBNull(dr("user\_pass")), dr("user\_pass"), "")

Me.txtfname.Text = IIf(Not IsDBNull(dr("user\_fname")), dr("user\_fname"), "")

Me.txtmname.Text = IIf(Not IsDBNull(dr("user\_mname")), dr("user\_mname"), "")

Me.txtlname.Text = IIf(Not IsDBNull(dr("user\_lname")), dr("user\_lname"), "")

If Not IsDBNull(dr("user\_img")) Then

arrImage = dr("user\_img")

For Each ar As Byte In arrImage

myMS.WriteByte(ar)

Next

Me.Userpic.Image = System.Drawing.Image.FromStream(myMS)

Else

Userpic.Image = Login.My.Resources.admin

End If

Me.txtaddr.Text = IIf(Not IsDBNull(dr("user\_addr")), dr("user\_addr"), "")

Me.txtdob.Value = IIf(Not IsDBNull(dr("user\_dob")), dr("user\_dob"), "")

Me.txtage.Text = IIf(Not IsDBNull(dr("user\_age")), dr("user\_age"), "")

Me.txtpos.Text = IIf(Not IsDBNull(dr("user\_pos")), dr("user\_pos"), "")

If dr("user\_stat") = "perm" Then

Me.txtstat.Text = "Permanent"

ElseIf dr("user\_stat") = "cont" Then

Me.txtstat.Text = "Contractual"

Else

Me.txtstat.Text = IIf(Not IsDBNull(dr("user\_stat")), dr("user\_stat"), "")

End If

End While

End If

Catch ex As Exception

MsgBox(ErrorToString)

Finally

conn.Close()

End Try

End Sub

Private Sub Clear\_userform()

txtuseridno.Clear()

txtfname.Clear()

txtmname.Clear()

txtlname.Clear()

txtaddr.Clear()

txtdob.Value = Date.Now

txtage.Clear()

txtpos.Clear()

txtstat.Text = ""

End Sub

Private Sub Read\_Only\_Controls()

txtuseridno.ReadOnly = True

txtfname.ReadOnly = True

txtmname.ReadOnly = True

txtlname.ReadOnly = True

txtaddr.ReadOnly = True

txtdob.Enabled = False

txtpos.ReadOnly = True

txtstat.Enabled = False

Userpic.Enabled = False

End Sub

Private Sub Read\_Controls()

txtuseridno.ReadOnly = False

txtfname.ReadOnly = False

txtmname.ReadOnly = False

txtlname.ReadOnly = False

txtaddr.ReadOnly = False

txtdob.Enabled = True

txtpos.ReadOnly = False

txtstat.Enabled = True

Userpic.Enabled = True

End Sub

Private Sub Delete\_User()

Dim conn As New OleDbConnection

Dim cmd As New OleDbCommand

Dim sSQL As String = String.Empty

Try

'get connection string declared in the Module1.vb and assing it to conn variable

conn = New OleDbConnection(constring)

conn.Open()

cmd.Connection = conn

cmd.CommandType = CommandType.Text

cmd.CommandText = "delete from tbluser where user\_idno = @id"

cmd.Parameters.Add("@id", OleDbType.Numeric).Value = Me.txtuseridno.Tag

cmd.ExecuteNonQuery()

Catch ex As Exception

MsgBox(ErrorToString)

Finally

conn.Close()

End Try

End Sub

Private Sub Save\_User()

Dim conn As New OleDbConnection

Dim cmd As New OleDbCommand

Dim sSQL As String = String.Empty

Dim arrImage() As Byte

Dim myMs As New IO.MemoryStream

Dim bSaveImage As Boolean = False

Dim strImg As String = String.Empty

Try

If Not IsNothing(Me.Userpic.Image) Then

Me.Userpic.Image.Save(myMs, Me.Userpic.Image.RawFormat)

arrImage = myMs.GetBuffer

Else

arrImage = Nothing

End If

'get connection string declared in the Module1.vb and assing it to conn variable

conn = New OleDbConnection(constring)

conn.Open()

cmd.Connection = conn

cmd.CommandType = CommandType.Text

'I just use the textbox tag property to idetify if the data is new or existing.

If Me.txtuseridno.Tag = "" Then

sSQL = "INSERT INTO tbluser (user\_pass, user\_fname, user\_lname, user\_mname, user\_addr, user\_dob, user\_age, user\_pos, user\_stat, user\_img, user\_idno)"

sSQL = sSQL & " VALUES('password', @user\_fname, @user\_lname, @user\_mname, @user\_addr, @user\_dob, @user\_age, @user\_pos, @user\_stat, @user\_img, @id)"

cmd.CommandText = sSQL

Else

sSQL = "UPDATE tbluser set user\_fname = @user\_fname, user\_lname = @user\_lname, user\_mname = @user\_mname, user\_addr = @user\_addr, user\_dob = @user\_dob, user\_age = @user\_age, user\_pos = @user\_pos, user\_stat = @user\_stat, user\_img = @user\_img where user\_idno = @id"

cmd.CommandText = sSQL

End If

cmd.Parameters.Add("@user\_fname", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtfname.Text)) > 0, Me.txtfname.Text, DBNull.Value)

cmd.Parameters.Add("@user\_lname", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtlname.Text)) > 0, Me.txtlname.Text, DBNull.Value)

cmd.Parameters.Add("@user\_mname", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtmname.Text)) > 0, Me.txtmname.Text, DBNull.Value)

cmd.Parameters.Add("@user\_addr", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtaddr.Text)) > 0, Me.txtaddr.Text, DBNull.Value)

cmd.Parameters.Add("@user\_dob", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtdob.Text)) > 0, Me.txtdob.Text, DBNull.Value)

cmd.Parameters.Add("@user\_age", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtage.Text)) > 0, Me.txtage.Text, DBNull.Value)

cmd.Parameters.Add("@user\_pos", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtpos.Text)) > 0, Me.txtpos.Text, DBNull.Value)

cmd.Parameters.Add("@user\_stat", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtstat.Text)) > 0, Me.txtstat.Text, DBNull.Value)

cmd.Parameters.Add("@user\_img", OleDbType.Binary).Value = IIf(Not IsNothing(arrImage), arrImage, DBNull.Value)

cmd.Parameters.Add("@id", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtuseridno.Text)) > 0, Me.txtuseridno.Text, DBNull.Value)

cmd.ExecuteNonQuery()

'If the record is new then we have to get its ID so that we can edit it rightaway after the insertion.

If Me.txtuseridno.Tag = "" Then

cmd.CommandText = "Select @@Identity"

'Set textbox tag property with the ID of new record

Me.txtuseridno.Tag = cmd.ExecuteScalar()

End If

Catch ex As Exception

MsgBox(ErrorToString)

Finally

conn.Close()

Load\_User()

End Try

End Sub

Private Sub Save\_Pass()

Dim conn As New OleDbConnection

Dim cmd As New OleDbCommand

Dim sSQL As String = String.Empty

Dim strImg As String = String.Empty

'Try

'get connection string declared in the Module1.vb and assing it to conn variable

conn = New OleDbConnection(constring)

conn.Open()

cmd.Connection = conn

cmd.CommandType = CommandType.Text

sSQL = "UPDATE tbluser set user\_pass = @user\_pass where user\_idno = @id"

cmd.CommandText = sSQL

cmd.Parameters.Add("@user\_pass", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtnewpass.Text)) > 0, Me.txtnewpass.Text, DBNull.Value)

cmd.Parameters.Add("@id", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtuseridno.Text)) > 0, Me.txtuseridno.Text, DBNull.Value)

cmd.ExecuteNonQuery()

'If the record is new then we have to get its ID so that we can edit it rightaway after the insertion.

If Me.txtuseridno.Tag = "" Then

cmd.CommandText = "Select @@Identity"

'Set textbox tag property with the ID of new record

Me.txtuseridno.Tag = cmd.ExecuteScalar()

End If

'Catch ex As Exception

' MsgBox(ErrorToString)

'Finally

conn.Close()

Load\_User()

'End Try

End Sub

#End Region

#Region "Tab 3 - Logs Procedure"

Private Function Load\_Logs(ByVal sSQL As String)

Dim conn As New OleDbConnection

Dim cmd As New OleDbCommand

Dim da As New OleDbDataAdapter

Dim dt As New DataTable

'try catch block is used to catch the error

Try

'get connection string declared in the Module1.vb and assing it to conn variable

conn = New OleDbConnection(constring)

conn.Open()

cmd.Connection = conn

cmd.CommandType = CommandType.Text

cmd.CommandText = sSQL

da.SelectCommand = cmd

da.Fill(dt)

Me.dtglogs.DataSource = dt

If dt.Rows.Count = 0 Then

End If

Me.lbllogs.Text = "Number of logs: " & dtglogs.RowCount

Catch ex As Exception

MsgBox(ErrorToString)

Finally

conn.Close()

End Try

End Function

Sub FilterLogs()

If RdAlldate.Checked Then

Load\_Logs("SELECT \* FROM tblDTR order by date\_timein DESC")

MonthCalendar.Enabled = False

txtuseridlog.Enabled = False

ElseIf RdDateSelected.Checked Then

Load\_Logs("SELECT \* FROM tblDTR WHERE date\_timein>=#" & MonthCalendar.SelectionStart & "# AND date\_timein<=#" & MonthCalendar.SelectionEnd & "#" & " order by date\_timein DESC")

MonthCalendar.Enabled = True

txtuseridlog.Enabled = False

ElseIf RdUserID.Checked Then

txtuseridlog.Focus()

Load\_Logs("SELECT \* FROM tblDTR WHERE user\_idno='" & txtuseridlog.Text & "' order by date\_timein DESC")

MonthCalendar.Enabled = False

txtuseridlog.Enabled = True

End If

End Sub

#End Region

#Region "Tab 1 - User Events"

Private Sub txtsearchuser\_TextChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles txtsearchuser.TextChanged

SearchUsername()

If txtsearchbyuser.Text = "ID" Then

SearchUserid()

ElseIf txtsearchbyuser.Text = "Name" Then

SearchUsername()

ElseIf txtsearchbyuser.Text = "Address" Then

SearchUseraddr()

ElseIf txtsearchbyuser.Text = "Position" Then

SearchUserpos()

End If

End Sub

Private Sub dtgUser\_CellClick(ByVal sender As Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles dtgUser.CellClick

If Me.dtgUser.SelectedRows.Count > 0 Then

Load\_Userform()

Read\_Only\_Controls()

btnnewuser.Text = "New"

btnedituser.Text = "Edit"

btnchangepass.Enabled = False

End If

End Sub

Private Sub dtgUser\_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs) Handles dtgUser.DoubleClick

MainTab.SelectedTab = TabItem2

btnedituser\_Click(sender, e)

End Sub

#End Region

#Region "Tab 2 - Userform Events"

Private Sub btnchangepass\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnchangepass.Click

GrpPassword.Enabled = True

End Sub

Private Sub btnnewuser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnnewuser.Click

'BUTTONS ENABLED

If btnnewuser.Text = "New" Then

Read\_Controls()

Clear\_userform()

btnnewuser.Text = "Save"

btnedituser.Text = "Cancel"

btnchangepass.Enabled = False

Me.txtuseridno.Tag = ""

Me.Userpic.Image = Login.My.Resources.admin

ElseIf btnnewuser.Text = "Save" Then

Dim res As New DialogResult

res = MsgBox("Do you want to save data?", MsgBoxStyle.YesNo, "Save")

If res = DialogResult.Yes Then

If Len(Trim(Me.txtuseridno.Text)) > 0 And Len(Trim(Me.txtfname.Text)) > 0 And Len(Trim(Me.txtmname.Text)) > 0 And Len(Trim(Me.txtlname.Text)) > 0 And Len(Trim(Me.txtaddr.Text)) > 0 And Len(Trim(Me.txtage.Text)) > 0 And Len(Trim(Me.txtpos.Text)) > 0 And Len(Trim(Me.txtstat.Text)) > 0 Then

Read\_Only\_Controls()

btnnewuser.Text = "New"

btnedituser.Text = "Edit"

btnchangepass.Enabled = False

Save\_User()

MessageBox.Show("New User has been Created. Password: password", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information)

Else

MsgBox("All Fields are required.")

End If

End If

ElseIf btnnewuser.Text = "Update" Then

Dim res As New DialogResult

res = MsgBox("Do you want to update data?", MsgBoxStyle.YesNo, "Update")

If res = DialogResult.Yes Then

If Len(Trim(Me.txtuseridno.Text)) > 0 And Len(Trim(Me.txtfname.Text)) > 0 And Len(Trim(Me.txtmname.Text)) > 0 And Len(Trim(Me.txtlname.Text)) > 0 And Len(Trim(Me.txtaddr.Text)) > 0 And Len(Trim(Me.txtage.Text)) > 0 And Len(Trim(Me.txtpos.Text)) > 0 And Len(Trim(Me.txtstat.Text)) > 0 Then

Read\_Only\_Controls()

btnnewuser.Text = "New"

btnedituser.Text = "Edit"

btnchangepass.Enabled = False

Save\_User()

Else

MsgBox("All Fields are required")

End If

End If

End If

End Sub

Private Sub btnedituser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnedituser.Click

If btnedituser.Text = "Cancel" Then

btnnewuser.Text = "New"

btnedituser.Text = "Edit"

btnchangepass.Enabled = False

Read\_Only\_Controls()

txtuseridno.Enabled = True

If Me.dtgUser.SelectedRows.Count > 0 Then

Load\_Userform()

End If

ElseIf btnedituser.Text = "Edit" Then

btnnewuser.Text = "Update"

btnedituser.Text = "Cancel"

btnchangepass.Enabled = True

Read\_Controls()

txtuseridno.Enabled = False

If Me.dtgUser.SelectedRows.Count > 0 Then

Load\_Userform()

End If

End If

End Sub

Private Sub btndeleteuser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btndeleteuser.Click

Dim res As New DialogResult

res = MsgBox("Do you want to delete data?", MsgBoxStyle.YesNo, "Delete")

If res = DialogResult.Yes Then

Delete\_User()

Clear\_userform()

Load\_User()

End If

End Sub

Private Sub txtdob\_ValueChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles txtdob.ValueChanged

Dim iDOB As Integer

iDOB = CInt(DateDiff(DateInterval.Year, txtdob.Value, Now) / 4)

txtage.Text = Floor((DateDiff(DateInterval.Day, txtdob.Value, Now) - iDOB) / 365)

End Sub

Private Sub Userpic\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Userpic.Click

OpenFileDialog1.FileName = ""

OpenFileDialog1.Filter = "JPEG files (\*.jpg)|\*.jpg|GIF files (\*.gif)|\*.gif|All files (\*.\*)|\*.\*"

OpenFileDialog1.ShowDialog()

Me.lblPath.Text = OpenFileDialog1.FileName

Me.Userpic.ImageLocation = Me.lblPath.Text

End Sub

Private Sub btnpasscancel\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnpasscancel.Click

GrpPassword.Enabled = False

txtoldpass.Clear()

txtnewpass.Clear()

txtconfirmpass.Clear()

End Sub

Private Sub btnpasschange\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnpasschange.Click

If txtoldpass.Text <> "" And txtoldpass.Text = lblpass.Text Then

If txtnewpass.Text <> "" And txtconfirmpass.Text <> "" And txtnewpass.Text = txtconfirmpass.Text Then

Save\_Pass()

GrpPassword.Enabled = False

txtoldpass.Clear()

txtnewpass.Clear()

txtconfirmpass.Clear()

Else

MessageBox.Show("The new passwords entered is not identical.", "Unidentical Password", MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

Else

MessageBox.Show("The old password entered is Invalid.", "Invalid Password", MessageBoxButtons.OK, MessageBoxIcon.Error)

txtoldpass.Focus()

End If

End Sub

#End Region

#Region "Tab 3 - Logs Events"

Private Sub MonthCalendar\_DateChanged(ByVal sender As Object, ByVal e As System.Windows.Forms.DateRangeEventArgs) Handles MonthCalendar.DateChanged

If RdDateSelected.Checked Then

Load\_Logs("SELECT \* FROM tblDTR WHERE date\_timein>=#" & MonthCalendar.SelectionStart & "# AND date\_timein<=#" & MonthCalendar.SelectionEnd & "#")

End If

End Sub

Private Sub RdAlldate\_CheckedChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles RdAlldate.CheckedChanged

FilterLogs()

End Sub

Private Sub RdDateSelected\_CheckedChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles RdDateSelected.CheckedChanged

FilterLogs()

End Sub

Private Sub txtuseridlog\_TextChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles txtuseridlog.TextChanged

Load\_Logs("SELECT \* FROM tblDTR WHERE user\_idno='" & txtuseridlog.Text & "' order by date\_timein DESC")

End Sub

Private Sub btnexport\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnexport.Click

If dtglogs.RowCount = Nothing Then

MessageBox.Show("Sorry nothing to export into excel sheet.." & vbCrLf & "Please retrieve data in datagridview", "", MessageBoxButtons.OK, MessageBoxIcon.Error)

Exit Sub

End If

Dim rowsTotal, colsTotal As Short

Dim I, j, iC As Short

System.Windows.Forms.Cursor.Current = System.Windows.Forms.Cursors.WaitCursor

Dim xlApp As New Excel.Application

Try

Dim excelBook As Excel.Workbook = xlApp.Workbooks.Add

Dim excelWorksheet As Excel.Worksheet = CType(excelBook.Worksheets(1), Excel.Worksheet)

xlApp.Visible = True

rowsTotal = dtglogs.RowCount

colsTotal = dtglogs.Columns.Count - 1

With excelWorksheet

.Cells.Select()

.Cells.Delete()

For iC = 0 To colsTotal

.Cells(1, iC + 1).Value = dtglogs.Columns(iC).HeaderText

Next

For I = 0 To rowsTotal - 1

For j = 0 To colsTotal

.Cells(I + 2, j + 1).value = dtglogs.Rows(I).Cells(j).Value

Next j

Next I

.Rows("1:1").Font.FontStyle = "Bold"

.Rows("1:1").Font.Size = 12

.Cells.Columns.AutoFit()

.Cells.Select()

.Cells.EntireColumn.AutoFit()

.Cells(1, 1).Select()

End With

Catch ex As Exception

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Finally

'RELEASE ALLOACTED RESOURCES

System.Windows.Forms.Cursor.Current = System.Windows.Forms.Cursors.Default

xlApp = Nothing

End Try

End Sub

Private Sub btndeletealllog\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btndeletealllog.Click

Try

If dtglogs.Rows.Count > 0 Then

If MessageBox.Show("Are you sure want to delete all logs?", "CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Information) = Windows.Forms.DialogResult.Yes Then

Using conn As New OleDbConnection(constring)

conn.Open()

Dim command As New OleDbCommand("delete \* from tblDTR", conn)

command.ExecuteNonQuery()

command.Dispose()

conn.Close()

End Using

Load\_Logs("SELECT \* FROM tblDTR order by date\_timein DESC")

End If

End If

Catch ex As Exception

MessageBox.Show(ex.Message, "ERROR12", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

#End Region

#Region "Context Menu"

Private Sub Toolreload\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Toolreload.Click

Load\_User()

End Sub

Private Sub ContextNewuser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ContextNewuser.Click

MainTab.SelectedTab = TabItem2

btnnewuser\_Click(sender, e)

End Sub

Private Sub ContextEditUser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ContextEditUser.Click

MainTab.SelectedTab = TabItem2

btnedituser\_Click(sender, e)

End Sub

Private Sub ContextDeleteUser\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ContextDeleteUser.Click

btndeleteuser\_Click(sender, e)

End Sub

Private Sub ToolPanelScale\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ToolPanelScale.Click

If SplitContainer1.Panel1Collapsed = False Then

SplitContainer1.Panel1Collapsed = True

ToolPanelScale.Enabled = False

Toolsplit.Enabled = True

End If

End Sub

Private Sub Toolsplit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Toolsplit.Click

If SplitContainer1.Panel1Collapsed = True Then

SplitContainer1.Panel1Collapsed = False

ToolPanelScale.Enabled = True

Toolsplit.Enabled = False

End If

End Sub

Private Sub Label24\_Click(sender As Object, e As EventArgs) Handles Label24.Click

End Sub

Private Sub Label11\_Click(sender As Object, e As EventArgs) Handles Label11.Click

End Sub

Private Sub dtgUser\_CellContentClick(sender As Object, e As DataGridViewCellEventArgs) Handles dtgUser.CellContentClick

End Sub

Private Sub CloudHeader3\_Load(sender As Object, e As EventArgs) Handles CloudHeader3.Load

End Sub

Private Sub Label2\_Click(sender As Object, e As EventArgs) Handles Label2.Click

End Sub

#End Region

End Class

**Front-End.vb**

Imports System.Data

Imports System.Data.OleDb

Public Class Front\_End

#Region "FORM CLOSE and MINIMIZE BUTTON"

Private Sub PictureBox1\_MouseHover(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnclose.MouseHover

Btnclose.BackColor = Color.Gray

End Sub

Private Sub Btnclose\_MouseLeave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnclose.MouseLeave

Btnclose.BackColor = Color.Black

End Sub

Private Sub Btnmin\_MouseHover(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnmin.MouseHover

Btnmin.BackColor = Color.Gray

End Sub

Private Sub Btnmin\_MouseLeave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnmin.MouseLeave

Btnmin.BackColor = Color.Black

End Sub

Private Sub Btnclose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnclose.Click

End

End Sub

Private Sub Btnmin\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnmin.Click

Me.WindowState = FormWindowState.Minimized

End Sub

#End Region

#Region "Variable Declaration"

Public Conn As OleDbConnection

Dim eDS As DataSet = New DataSet

Dim eDA As OleDbDataAdapter = New OleDbDataAdapter

Dim eDR As DataRow

Dim dDS As DataSet = New DataSet

Dim dDA As OleDbDataAdapter = New OleDbDataAdapter

Dim dDR As DataRow

'Public bExitApplication As Boolean

#End Region

#Region "Procedures "

Private Sub Check\_Info()

Dim arrImage() As Byte

Dim myMS As New IO.MemoryStream

Conn.Open()

eDA.SelectCommand = New OleDbCommand("SELECT user\_fname,user\_lname,user\_mname, user\_img FROM tblUser WHERE user\_idno='" & txtuserid.Text.ToString & "' and user\_pass='" & txtpass.Text & "'", Conn)

eDS.Clear()

eDA.Fill(eDS)

If eDS.Tables(0).Rows.Count > 0 Then

eDR = eDS.Tables(0).Rows(0)

txtusername.Text = eDR("user\_lname") & ", " & eDR("user\_fname") & " " & eDR("user\_mname")

If Not IsDBNull(eDR("user\_img")) Then

arrImage = eDR("user\_img")

For Each ar As Byte In arrImage

myMS.WriteByte(ar)

Next

Me.loginimage.Image = System.Drawing.Image.FromStream(myMS)

Else

loginimage.Image = Login.My.Resources.admin

End If

dDA.SelectCommand = New OleDbCommand("SELECT \* FROM tblDTR WHERE user\_idno='" & txtuserid.Text & "' AND date\_timein=#" & Format(Now, "MM/d/yyyy") & "# AND time\_timeout IS NULL", Conn)

dDS.Clear()

dDA.Fill(dDS)

If dDS.Tables(0).Rows.Count > 0 Then

dDR = dDS.Tables(0).Rows(0)

btntimein.Enabled = True

btntimein.Text = "Time Out"

txttimein.Text = dDR("time\_timein")

Else

btntimein.Enabled = True

btntimein.Text = "Time In"

End If

dDR = Nothing

dDS.Dispose()

dDA.Dispose()

Else

txtusername.Clear()

txttimein.Clear()

txttimeout.Clear()

btntimein.Enabled = False

End If

eDR = Nothing

eDS.Dispose()

eDA.Dispose()

Conn.Close()

End Sub

Private Sub ReturnFocus(ByVal sender As Object, ByVal e As System.EventArgs) Handles txtusername.GotFocus, txttimein.GotFocus, txttimeout.GotFocus

txtuserid.Focus()

End Sub

Private Sub TextboxChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtuserid.TextChanged, txtpass.TextChanged

If (txtuserid.Text <> "" And txtpass.Text <> "") Then

Check\_Info()

End If

End Sub

Private Sub LoginGotFocus(ByVal sender As Object, ByVal e As System.EventArgs) Handles txtuserid.GotFocus, txtpass.GotFocus

txtusername.Clear()

txttimein.Clear()

txttimeout.Clear()

loginimage.Image = Login.My.Resources.admin

End Sub

#End Region

Private Sub Front\_End\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Conn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & Application.StartupPath & "\Dbase\UTIS.mdb;Persist Security Info=False")

txtuserid.Focus()

End Sub

Private Sub btnclear\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnclear.Click

txtuserid.Clear()

txtpass.Clear()

txtusername.Clear()

txttimein.Clear()

txttimeout.Clear()

btntimein.Text = "Time In"

btntimein.Enabled = False

txtuserid.Focus()

loginimage.Image = Login.My.Resources.admin

End Sub

Private Sub btntimein\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btntimein.Click

' Time-in and Time-out button

Dim strSQL As String

If txtuserid.Text = "" Or txtpass.Text = "" Then

MessageBox.Show("Please Enter your User Data", "Enter User Data", MessageBoxButtons.OK, MessageBoxIcon.Information)

Else

Conn.Open()

If btntimein.Text = "Time In" Then

dDR = dDS.Tables(0).NewRow()

txttimein.Text = Format(Now, "h:mm:ss tt")

strSQL = "INSERT INTO tblDTR (user\_idno, date\_timein, time\_timein) VALUES ('" & txtuserid.Text & "', #" & Format(Now, "MM/d/yyyy") & "#, #" & txttimein.Text & "#)"

btntimein.Text = "Time Out"

Else

txttimeout.Text = Format(Now, "h:mm:ss tt")

strSQL = "UPDATE tblDTR SET time\_timeout=#" & txttimeout.Text & "# WHERE user\_idno='" & txtuserid.Text & "' AND date\_timein=#" & Format(Now, "MM/d/yyyy") & "# and time\_timein=#" & txttimein.Text & "#"

btntimein.Text = "Time In"

End If

Dim dCmd As OleDbCommand = New OleDbCommand(strSQL, Conn)

dCmd.ExecuteNonQuery()

dCmd.Dispose()

dDR = Nothing

dDS.Dispose()

dDA.Dispose()

Conn.Close()

btntimein.Enabled = False

txtuserid.Clear()

txtpass.Clear()

End If

End Sub

Private Sub toolclear\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles toolclear.Click

btnclear\_Click(sender, e)

End Sub

Private Sub btnadminlogin\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnadminlogin.Click

Me.Close()

frmadminlogin.MdiParent = Main

frmadminlogin.Show()

End Sub

Private Sub toolAdmin\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles toolAdmin.Click

btnadminlogin\_Click(sender, e)

End Sub

End Class

**Main.vb**

Option Explicit On

Public Class Main

Private Sub Main\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Front\_End.MdiParent = Me

Front\_End.StartPosition = FormStartPosition.CenterScreen

Front\_End.Show()

End Sub

End Class

**frmadmin.vb**

Imports System.Data.OleDb

Public Class frmadmin

Dim constring As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" & Application.StartupPath & "\Dbase\UTIS.mdb;Persist Security Info=False"

Private Sub btnclose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnclose.Click

Me.Close()

End Sub

Private Sub Load\_Admin()

Dim conn As New OleDbConnection

Dim cmd As New OleDbCommand

Dim dr As OleDbDataReader

Dim myMS As New IO.MemoryStream

Try

conn = New OleDbConnection(constring)

conn.Open()

cmd.Connection = conn

cmd.CommandType = CommandType.Text

cmd.CommandText = "select \* from tblSysLogin"

dr = cmd.ExecuteReader

If dr.HasRows Then

While dr.Read

Me.txtusername.Text = IIf(Not IsDBNull(dr("usr\_name")), dr("usr\_name"), "")

Me.txtpass.Text = IIf(Not IsDBNull(dr("usr\_pass")), dr("usr\_pass"), "")

End While

End If

Catch ex As Exception

MsgBox(ErrorToString)

Finally

conn.Close()

End Try

End Sub

Private Sub Save\_Admin()

Dim conn As New OleDbConnection

Dim cmd As New OleDbCommand

Dim sSQL As String = String.Empty

Dim strImg As String = String.Empty

Try

'get connection string declared in the Module1.vb and assing it to conn variable

conn = New OleDbConnection(constring)

conn.Open()

cmd.Connection = conn

cmd.CommandType = CommandType.Text

sSQL = "UPDATE tblSysLogin set usr\_pass = @usr\_pass, usr\_name = @usr\_name"

cmd.CommandText = sSQL

cmd.Parameters.Add("@usr\_pass", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtpass.Text)) > 0, Me.txtpass.Text, DBNull.Value)

cmd.Parameters.Add("@usr\_name", OleDbType.VarChar).Value = IIf(Len(Trim(Me.txtusername.Text)) > 0, Me.txtusername.Text, DBNull.Value)

cmd.ExecuteNonQuery()

Catch ex As Exception

MsgBox(ErrorToString)

Finally

conn.Close()

End Try

End Sub

Private Sub frmadmin\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Load\_Admin()

End Sub

Private Sub btnsave\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnsave.Click

Save\_Admin()

Me.Close()

End Sub

End Class

**frmadminlogin.vb**

Imports System.Data

Imports System.Data.OleDb

Public Class frmadminlogin

#Region "FORM CLOSE and MINIMIZE BUTTON"

Private Sub PictureBox1\_MouseHover(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnclose.MouseHover

Btnclose.BackColor = Color.Gray

End Sub

Private Sub Btnclose\_MouseLeave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnclose.MouseLeave

Btnclose.BackColor = Color.Black

End Sub

Private Sub Btnmin\_MouseHover(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnmin.MouseHover

Btnmin.BackColor = Color.Gray

End Sub

Private Sub Btnmin\_MouseLeave(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnmin.MouseLeave

Btnmin.BackColor = Color.Black

End Sub

Private Sub Btnclose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnclose.Click

Me.Close()

Front\_End.MdiParent = Main

Front\_End.Show()

End Sub

Private Sub Btnmin\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnmin.Click

Me.WindowState = FormWindowState.Minimized

End Sub

#End Region

#Region "Variable Declaration"

Dim Conn As OleDbConnection

Dim lDS As DataSet = New DataSet

Dim lDA As OleDbDataAdapter = New OleDbDataAdapter

Dim lDR As DataRow

#End Region

Private Sub btncancel\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btncancel.Click

Me.Close()

Front\_End.MdiParent = Main

Front\_End.Show()

End Sub

Private Sub frmadminlogin\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Conn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & Application.StartupPath & "\Dbase\UTIS.mdb;Persist Security Info=False")

End Sub

Private Sub btnlogin\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnlogin.Click

Conn.Open()

lDA.SelectCommand = New OleDbCommand("SELECT \* FROM tblSysLogin WHERE usr\_name='" & txtusername.Text.ToString & "' and usr\_pass='" & txtpass.Text & "'", Conn)

lDS.Clear()

lDA.Fill(lDS)

If lDS.Tables(0).Rows.Count > 0 Then

Me.Close()

Back\_End.MdiParent = Main

Back\_End.Show()

Else

MsgBox("Sorry, you do NOT have Administrative permission. Please contact the System Administrator.", MsgBoxStyle.Exclamation + MsgBoxStyle.OkOnly, "Access Denied")

txtusername.Focus()

End If

lDR = Nothing

lDS.Dispose()

lDA.Dispose()

Conn.Close()

End Sub

Private Sub toolcancel\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles toolcancel.Click

btncancel\_Click(sender, e)

End Sub

End Class

## TESTING PHASE

Software Testing Software Testing is the process of executing software in a controlled manner. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses teniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Testing is a set of activity that can be planned in advance and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system.

There are three ways to test program.

* For correctness
* For implementation efficiency
* For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do.

### Test Plan

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

* Unit testing
* Integration Testing
* Data validation Testing
* Output Testing

### UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is whitebox oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

2 Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

### INTEGRATION TESTING

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop.

### SYSTEM TESTING

This is the final step in testing. In this the entire system is tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

### ACCEPTANCE TESTING

The system considered is tested for user acceptance; here it should satisfy the firm’s need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points

⮚ Input Screen Design

⮚ Output Screen Designs

⮚ Message to guide the user

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

### SYSTEM MAINTAINANCE

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner.

System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding

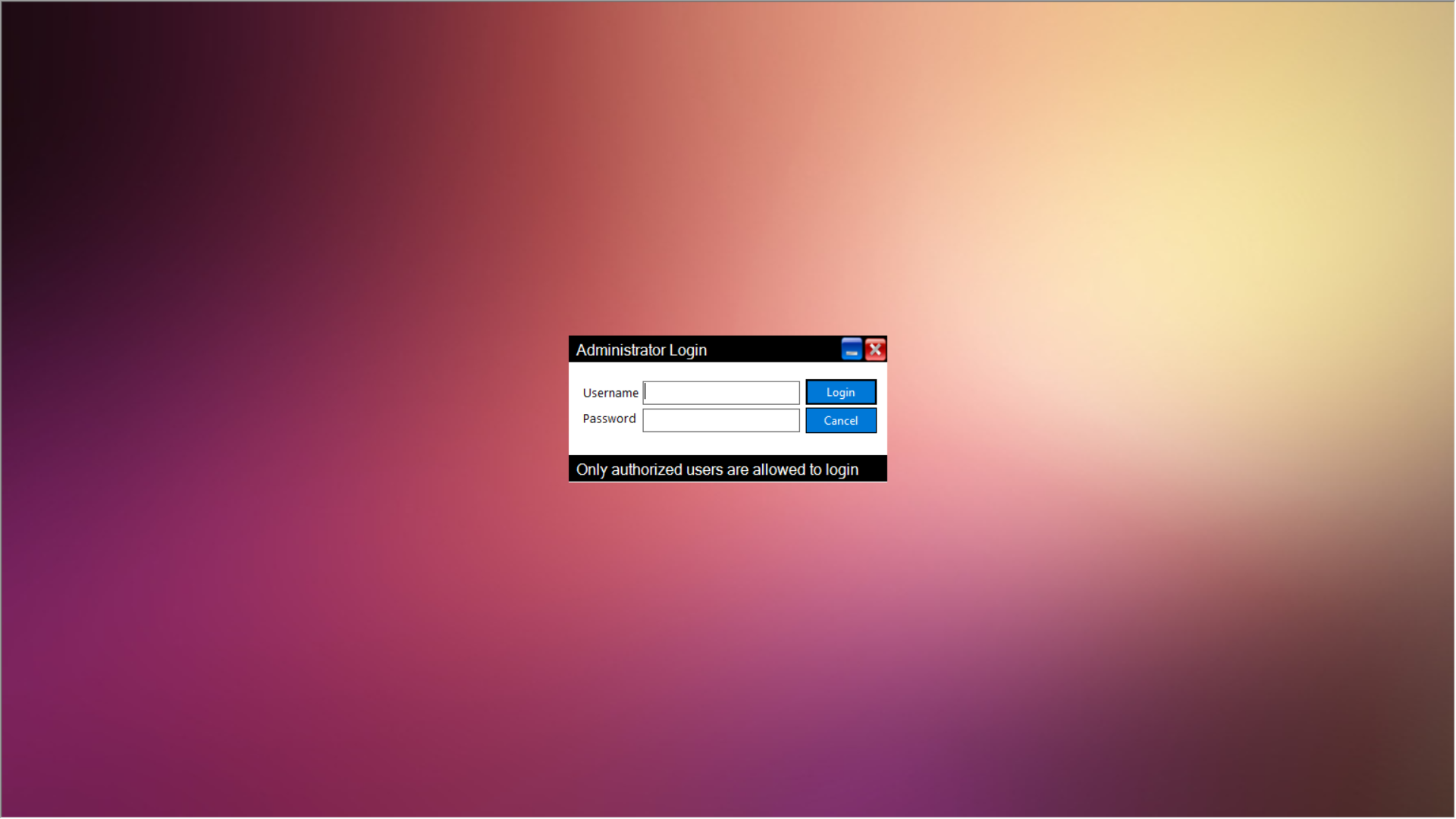
Mistakes". Maintenance may be defined by describing four activities that are undertaken after a program is released for use.

## SNAPSHOTS

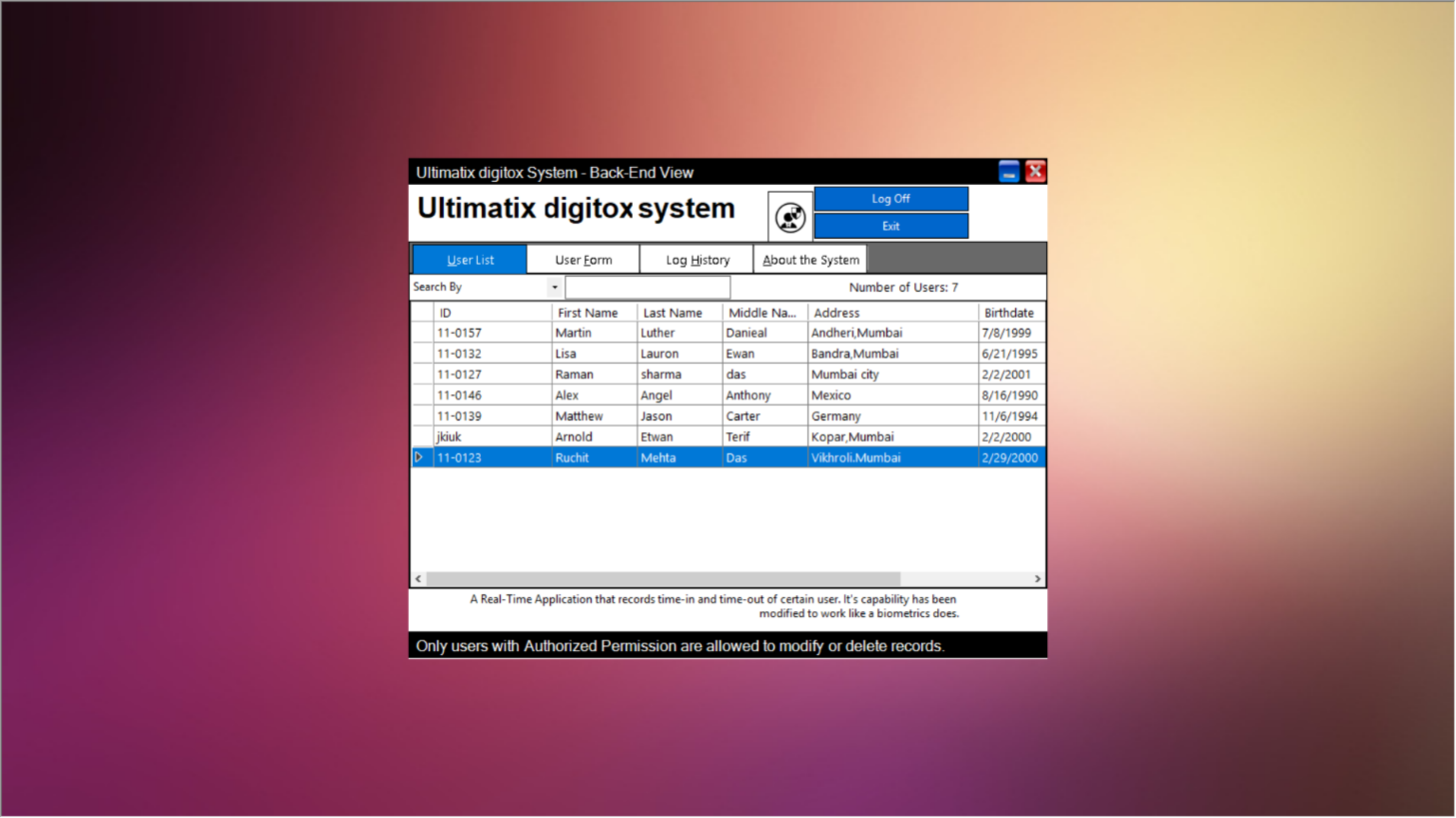
Front end view-



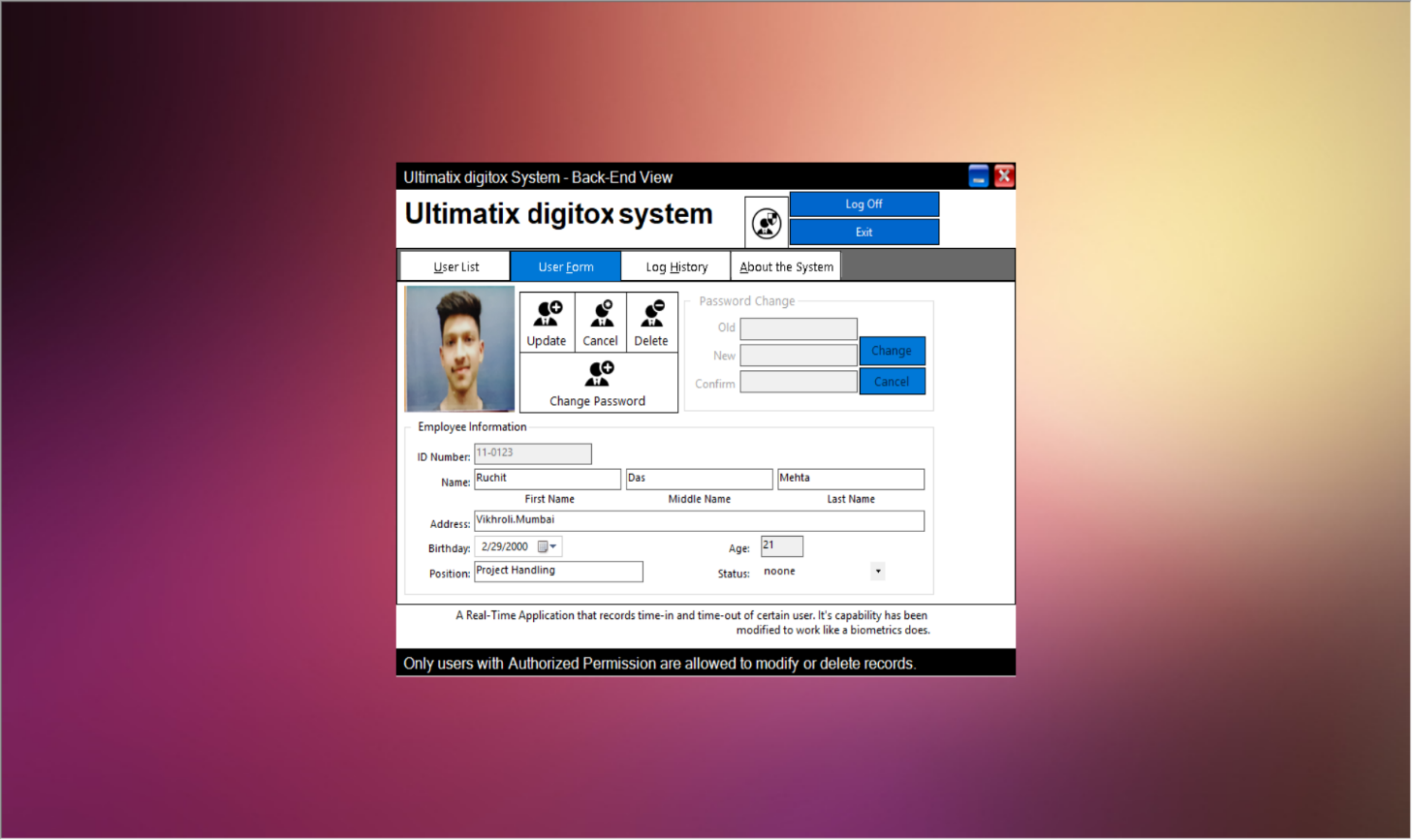
Administration Login-



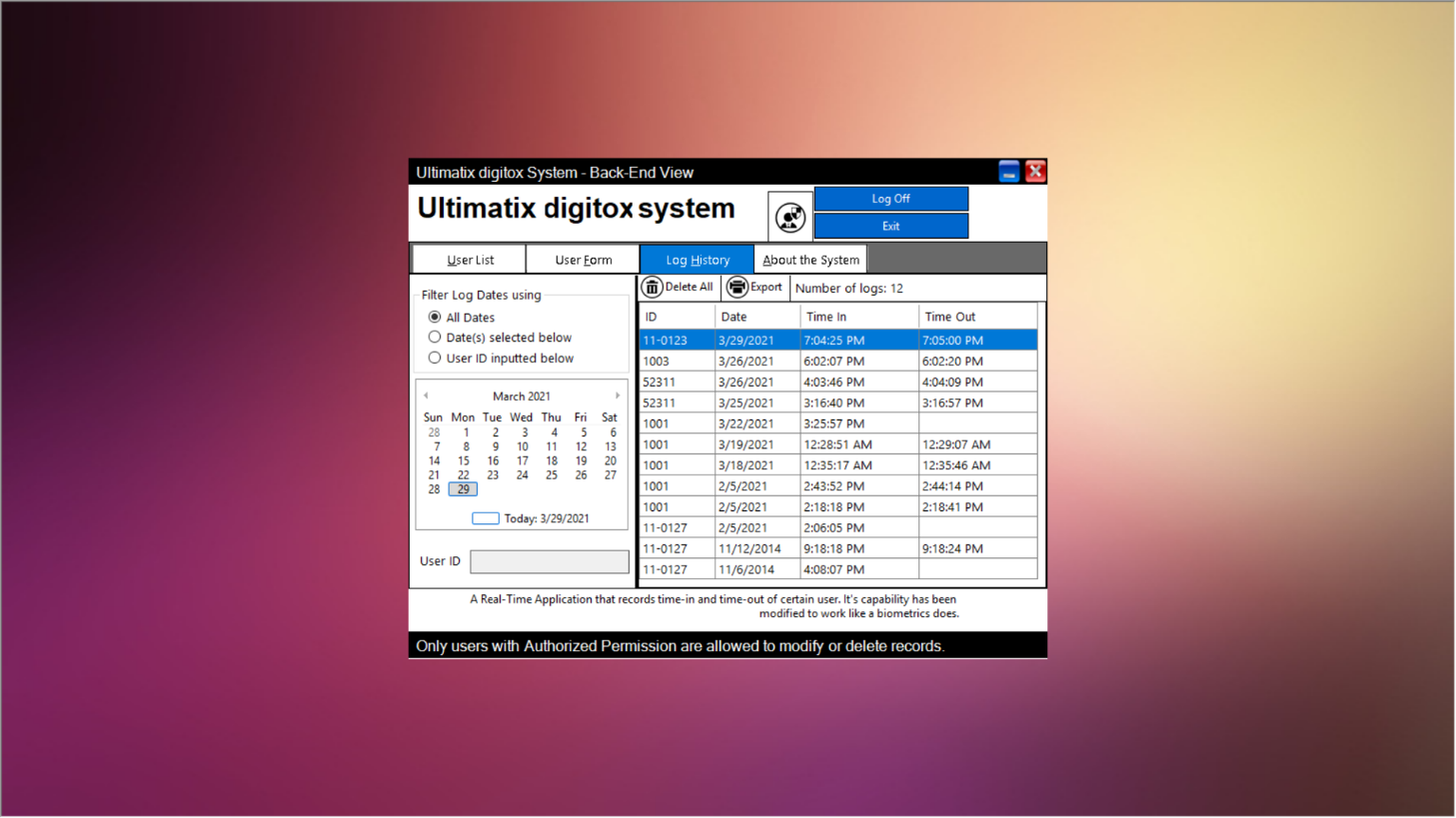
User list-



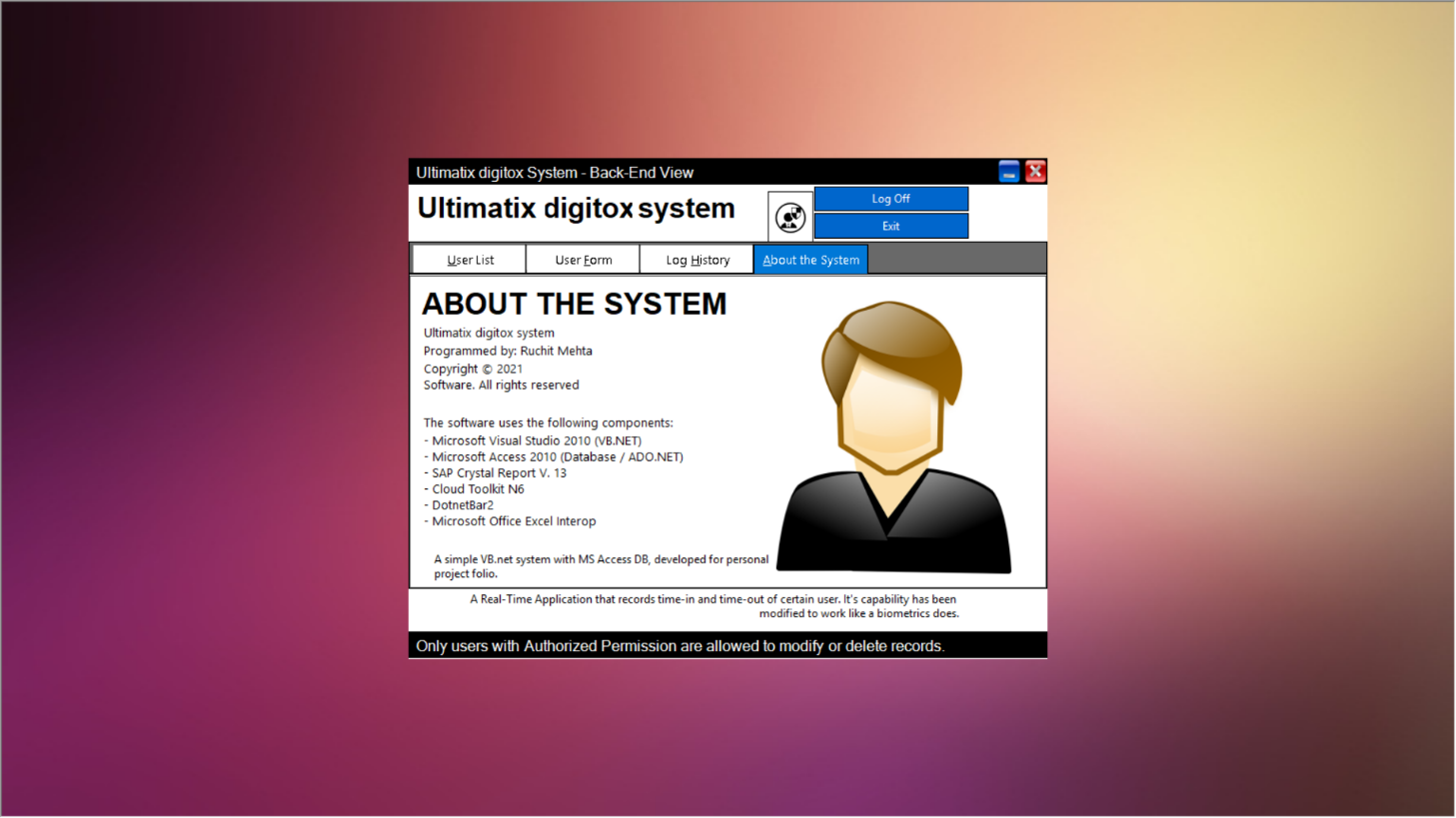
User form-



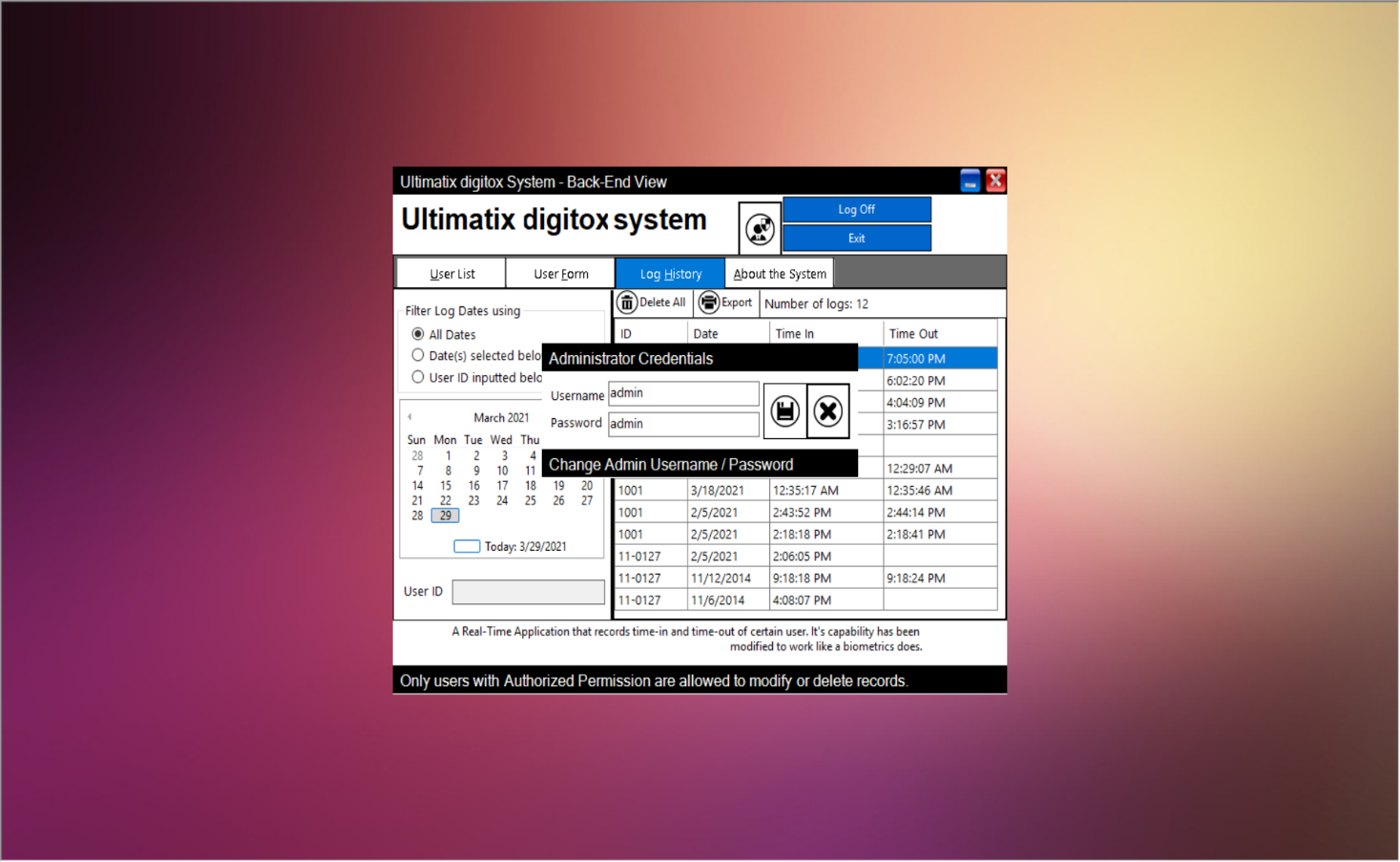
Log history



About the system-



Administrator credential-



## Time Tracking Enhancements

## 

Time tracking is the process of tracking one’s time in hours so that they can get

paid at the end of their payment cycle.

As a manager or boss, you’ve probably said the age-old phrase, “Don’t forget to log your hours this week.” Trying to get everyone to log their time can sometimes feel like herding rogue sheep. Additionally, doing manual timesheets can bring up issues of dishonesty, excessive overtime, and even time theft.

This is why is time tracking is so important. There are several benefits to time tracking and a number of industries this time tracking system can be used some of which are:

1. **Keep track of contracts**-

Keeping an accurate record of hours worked ensures you’re only paying for what’s actually done, and also enables you to track the performance of different contractors so you can allocate work to the right people.

1. **Provide transparency-**

Team members can log hours automatically and give transparency into their work, which can help keep team members accountable.

1. **Improve project schedules**-

Your time tracking software will tell you how many hours each task on a job takes. If you see consistent patterns in certain project types / stages, perhaps you need to adjust your schedule to a more realistic timeframe.

1. **Review team performance-**

For the members of your team to see how their performance stacks up, and can help you both identify strengths to emphasize and weaknesses to work on.

1. **Accurately report hours-**

Hours worked by your team are required for reporting and compliance reasons. This can be a huge task at the end of each month or quarter to compile and send off.

By accurate report hours,you will be able to quickly run a report and pull out the information you need.

## REFERENCES

**ONLINE REFERENCES:**

* **https://www.geeksforgeeks.org**
* **https://www.google.com**
* **https://www.guru99.com**
* **https://www.geeksforgeeks.org**
* **https://www.youtube.com**