LEAVE MANAGEMENT SYSTEM

SYNOPSIS

SYNOPSIS

Title: Leave Management System

OBJECTIVE:

Leave Management System project in VB.Net is a windows application specially designed to manage and store the details of employees as well as manage and store the leave taken by employees of an organization. We designed this software in Vb.Net programming language. The objective and scope of Leave Management is to maintain the leaves of employees and thereby recording the leaves for future use.

The primary purpose of Leave Management System is to search for employee details, manage their leaves accordingly and thereby store or record the details. The organization should also maintain the records on all the employees and the project team. The employees thus will be able to request a Leave easily and efficiently. To be able to fewer paper works and faster leave applying with lesser time to be spent and finally to evaluate the acceptability of the proposed system.

In this project, we are using VB.Net 2019 and Bunifu Framework to maintain the front end of the project and the back end is implemented on Remote SQL Server (WAMP) 8.0.31 Remote SQL Server software is used to design the backend mainly because we can create relational table employee database system easily; using SQL Server and VB.Net software , provide a good graphical user interface to the user of the system.

List of Modules:

Login Module:

Check the login details. This module is used to login into the system.

It is used by the administrator or the HR personnel of the organization. This module is used to login into the system. It is helpful for the administrator or the HR personnel of the organization to log into the Leave Management System.

Employee Module:

An Employee module generally consist of two types of information about an employee. The first type is Personal Information , which consist of various personal information of an employee. The various personal information are: Employee ID, Employee Name , Employee Sex, Employee Address, Employee DOB, Employee civil status, Employee Contact Number. Whereas the second type is work information. The work information is composed of daily rate of the employee, position of the employee, department in which the employee is working on, pay mode, date hired and work status .

Thus, all together this module is use to add the details of an employee in an organization.

Find Employee Module:

This module is used to modify the details of an employee that are already existing or to modify the details by adding new details of an employee. It can also be used to delete the details of an employee from the system.

Leave of Absence Module:

This module consists of two segments namely – Add Leave of Absence and History. The first segment is used for applying leave. It includes employee details like Employee Id, Employee position in the organization, Employee salary and the department in which the employee is working on. Leave applied for is generally specifies the reason for which the employee is taking the leave. It includes type of leave like with or without pay, leave starting date, leave end date and the time.

The second segment shows the leave details of all the employees in an organization. It includes the active leaves i.e., the which are currently on a leave, as well as the details of leave taken by the employees previously.

Settings Module:

The settings module consists of two segments namely: Position, Department. Position is used by the administrator to set, update and delete the position of employee inside a particular department within an organization. It is helpful for the HR Personnel also to manage different employee details.

Department Segment on the other hand is used to set, update and delete the department the employee is currently working on.

Manage Users Module:

This module is used to manage the type of users who have the access to the system. This module includes the user id, name, username, password and type of user (Administrator or HR personnel). Through this module we can add user who can access the system.

Log out Module:

This module is mainly going back to the login page and to log out from the system.

REQUIREMENTS SPECIFICATION

Hardware Requirements:

- Processor : 32 bit, Pentium-4 or higher
- RAM : 4 GB RAM or higher
- Hard disk : 10 GB (Minimum)
- Monitor : 1024x768(Resolution)
- Keyboard
- Mouse
- Printer

Software requirements:

- Operating System : Windows 10
- Front End : VB.Net
  : Bunifu Framework
- Back End : Remote SQL Server

INTRODUCTION

INTRODUCTION

An organization always need to monitor the performance of its numerous employees. This kind of organization must have an information system that can do all of that in a convenient manner. In one year, an employee can commit several leaves of absences. So, the organizations can use the manual system, where the employee process leaves by filling up a form and let his/her manager or supervisor approve it. This type of system is much more time consuming.

Thus, Leave Management System can be used in such case , which will aim to have a better system. Here, administrator would get to track who and how many employees are on leave and are available. This Leave Management System will bring to the betterment for future use to the company and its employees. It will be easy for the organization to provide information about an employee or queries about leaves of every employee whether it may be concerned about leave balances, approval of leave on a certain date and etc. This system is an stand alone system for managing leaves related info f employees and approval of leaves, cancellation, etc. are elements of the system. There are features like storing employee details, editing employee details, managing users of the system, finding the leave details of an employee, applying for leave of absence, looking for past details.

LEAVE MANAGEMENT SYSTEM

Leave Tracking is the fundamental requirement of a leave management system because inability to track leaves properly can result in staffing shortages, workload build- ups and business.

Some of the major challenges that an organization can tackle with this system are :

- Reduce the loads of paper work
- Understanding the employee leave pattern

Nowadays companies avoids paperwork and manual computing. So, in current situation Leave Management System is the ideal system to manage the leaves as well as details of employees inside an organization.

Objective:

The main objective of the leave management project is to manage the details of the leaves taken by the employees of an organization.

It manages all the information about the leave like leave types, date of leave, number of days leaves taken and the date of joining. This project is totally built of the administrator or the HR to help to keep track of the employees who have taken leave. The purpose of this project is to build an application program to reduce manual work for managing leave, Employee details and leave types (paid or unpaid). It tracks all the information of the employees and the leaves.

Moving Data

Moving Data:

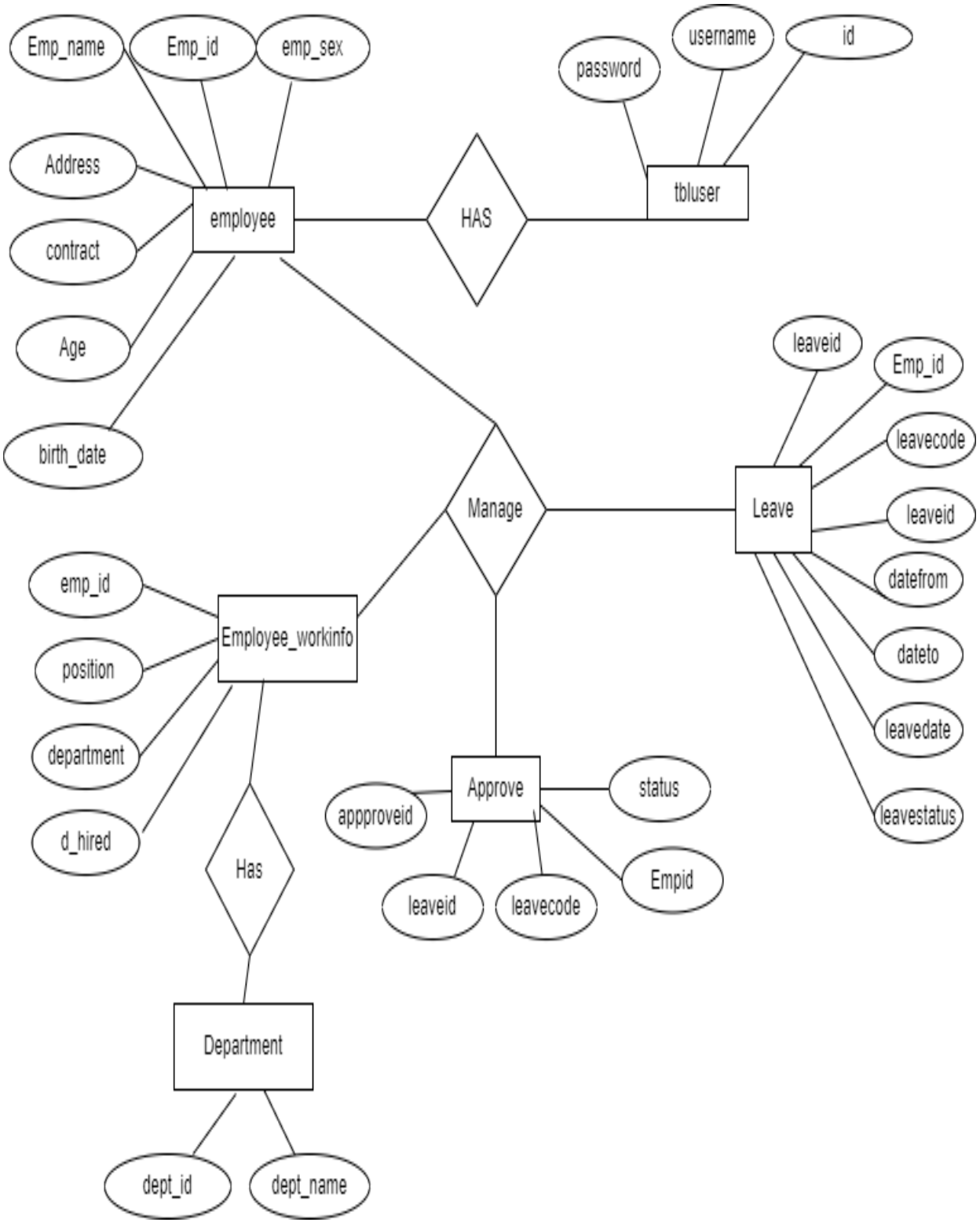
Recovery

Recovery:

Restore:

E-R Diagram

E-R Diagram:


Emp_name
Emp_id
emp_sex
Address
contract
Age
birth_date
employee
HAS
password
username
id
tbluser
Manage
leaveid
Emp_id
leavecode
leaveid
datefrom
dateto
leavedate
leavestatus
Leave
emp_id
position
department
d_hired
Employee_workinfo
Has
Department
dept_id
dept_name
Approve
appproveid
status
Empid
leaveid
leavecode

Tool Description

Visual Basic 2019:

Visual Basic Express 2019 is the version of Visual Basic used to develop this application. Visual Basic Express 2019 is almost similar to Visual Basic Express 2015, but it has added many new features. The most distinct difference is that Visual Basic Express 2012 no more comes as a standalone program, it is now integrated with other Microsoft Programming languages C# and C++ in a package called Visual Studio 2019. Further, Visual Studio Express 2019 now come in five editions, they are:

- Visual Studio Express 2019 for Web
- Visual Studio Express 2019 for Windows 8
- Visual Studio Express 2019 for Windows Desktop
- Visual Studio Express 2019 for Windows Phone
- Visual Studio Team Foundation Server Express 2019

Like Visual Basic Express 2015, Visual Basic Express 2013 is also a full-fledged Object-Oriented Programming (OOP) Language, so it has caught up with other OOP languages such as C++, Java, C# and others.

Visual Basic 2019 Data Types:

Visual Basic 2019 classifies the information mentioned above into two major data types, they are the numeric data types and the non-numeric data types.

Numeric Data Types:

Numeric data types are types of data that consist of numbers, which can be computed mathematically with various standard operators such as add, minus, multiply, divide and so on. Examples of numeric data types are your examination marks, your height, your weight, the number of students in a class, share values, price of goods, monthly bills, fees and etc. In Visual Basic 2019, numeric data are divided into 7 types, depending on the range of values they can store. Calculations that only involve round figures or data that don't need precision can use Integer or Long integer in the computation. Programs that require high precision calculation need to use Single and Double decision data types, they are also called floating point numbers. For currency calculation, you can use the currency data types. Lastly, if even more precision is requires to perform calculations that involve a many decimal points, we can use the decimal data types

Non-numeric Data Types:

Nonnumeric data types are data that cannot be manipulated mathematically using standard arithmetic operators. The non-numeric data comprises text or string data types, the Date data types, the Boolean data types that store only two values (true or false), Object data type and Variant data type.

Visual Studio 2019 Project Page:

The New Project Page comprises three templates, Visual Basic, Visual C# and Visual C++. We shall select Visual Basic. Visual

Basic 2019 offers you four types of projects that you can create. As we are going to learn to create windows Applications, we will select Windows Forms Application.

At the bottom of this dialog box, you can change the default project name WindowsApplication1 to some other name you like, for example, MyFirstProgram. After you have renamed the project, click OK to continue.

Controls in Visual Basic 2019 are objects that can be placed on the form to perform various tasks. We can use them to create all kinds of Windows applications. The diagram below shows the toolbox that contains the controls of Visual Basic 2019. They are categorized into Common Controls, Containers, Menus, Toolbars, Data, Components, Printings and Dialogs. At the moment, we will focus on the common controls. Some of the most frequently used common controls are Button, Label, ComboBox, ListBox, PictureBox, Text Box etc.

To insert a control into your form in Visual Basic 2019 IDE, you just need to drag the control from the toolbox and drop it into the form. You can reposition and resize it as you like.

The Control Properties in Visual Basic 2019:

Before writing an event procedure for a control in Visual Basic 2019 to response to a user's input or action, you have to set certain properties for the control to determine its appearance and how it will work with the event procedure. You can set the properties of the controls in the properties window of Visual Basic 2019 IDE at design time or at run time.

BUNIFU FRAMEWORK:

Bunifu Framework UI Tools provides an easy way to craft stunning desktop apps UI in less time. With Bunifu Framework UI tools, we will get all the tools to maximize our creativity, make our UI design more stronger. Bunifu Framework helps developers to create stunning software interfaces and user experience. It supports Windows Forms for C# and VB.NET.

FEATURES:

- Simplicity:

  It is very simple to use. Simply drag and drop the controls in the forms.

- Performance:

  Many design framework are heavy on the system. But Bunifu UI WinForms are lightweight and easy to use.

- Controls:

  Bunifu Framework UI controls are easy to customize and gives us power to create stunning UI designs.

Controls	**Use**
Picture Box	Used to insert image into a form of any shape like circular, sharp corner or round corner.
Image Button	use add interactivity to images.
Gradient button	Use to add gradient to element form at design time or runtime.

Button	Use to add button which generally used as an interactive element to submit data.
Dock	Use to add docking(moving) ability to out forms
Text Box	Use to give input by the user
Toolstrip	Use to add multiple child forms a parent form
Snack Box	Use to display event-based pop up message

HOW TO USE BUNIFU IN VISUAL BASIC 2019 :

- ***STEP 1:*** Create a new VB.NET forms project.
- ***STEP 2:*** Right Click on the toolbox and click choose items.
- ***STEP 3:*** Click Browse .
- ***STEP 4:*** Select Bunifu UI dll File.
- ***STEP 5:*** Drag Drop Bunifu Controls from toolbox.
- ***STEP 6:*** Open the program file.
- ***STEP 7:*** Paste the provided license token .
- ***STEP 8:*** Run the program.

Microsoft SQL Server 2014

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Its primary query languages are T-SQL and ANSI SQL.

SQL Server 2014 was released on April 2014 and it has started becoming favorite among professionals. Any new product comes from Microsoft the first thing I personally ask myself, is it worth to jump in?. Is it worth to spend customer's hard earned money to get in to that product?. The way to assess the same is dividing the product features in to "revolution" and "evolution". "Revolution" means it's completely a new thing while "evolution" means there was something already and it has been improvised.

Features:

1. **Always On Availability Groups**

This feature takes database mirroring to a whole new level. With always On, users will be able to fail over multiple databases in groups instead of individually. Also, secondary copies will be readable, and can be used for database backups. The big win is that your DR environment no longer needs to sit idle.

2. **Windows Server Core Support**

If you don't know what Windows Server Core is, you may want to come up to speed before Windows 8 (MS is making a push back to the command line for server products). Core is the GUI-less version of Windows that uses DOS and PowerShell for user interaction. It has a much lower footprint (50% less memory and disk space utilization), requires fewer patches, and is more secure than the full install. Starting with SQL 2014, it is supported for SQL Server.

3. **Column store Indexes**

This a cool new feature that is completely unique to SQL Server. They are special type of read-only index designed to be use with Data Warehouse queries. Basically, data is grouped and stored in a flat, compressed column index, greatly reducing I/O and memory utilization on large queries.

4. **User-Defined Server Roles**

DBAs have always had the ability to create custom database role, but never server wide. For example, if the DBA wanted to give a development team read/write access to every database on a shared server, traditionally the only ways to do it were either manually, or using undocumented procedures. Neither of which were good solutions. Now, the DBA can create a role, which has read/write access on every DB on the server, or any other custom server wide role.

5. Enhanced Auditing Features

Audit is now available in all editions of SQL Server. Additionally, users can define custom audit specifications to write custom events into the audit log. New filtering features give greater flexibility in choosing which events to write to the log.

6. BI Semantic Model

This is replacing the Analysis Services Unified Dimensional Model (or cubes most people referred to them). It's a hybrid model that allows one data model will support all BI experiences in SQL Server. Additionally, this will allow for some really neat text info graphics

7. Sequence Objects

For those folks who have worked with Oracle, this has been a long requested feature. A sequence is just an object that is a counter --

a good example of it's use would be to increment values in a table, based a trigger. SQL has always had similar functionality with identity columns, but now this is a discrete object.

8. Enhanced PowerShell Support –

Windows and SQL Server admins should definitely start brushing up on their PowerShell scripting skills. Microsoft is driving a lot of development effort into instrumenting all of their server-based products with PowerShell. SQL 2014 gave DBAs some exposure to it, but there are many more in cmdlets in SQL 2014.

9. Distributed Replay

Once again this is answer to a feature that Oracle released (Real Application Testing). However, and in my opinion where the real value proposition of SQL Server is, in Oracle it is a (very expensive) cost option to Enterprise Edition. With SQL, when you buy your licenses for Enterprise Edition, you get everything. Distributed replay allows you to capture a workload on a production server, and replay it on another machine. This way changes in underlying schemas, support packs, or hardware changes can be tested under production conditions.

10. Power View

You may have heard of this under the name "Project Crescent" it is a fairly powerful self-service BI toolkit that allows users to create mash ups of BI reports from all over the Enterprise.

11. SQL Azure Enhancements

These don't really go directly with the release of SQL 2014, but Microsoft is making some key enhancements to SQL Azure. Reporting Services for Azure will be available, along with backup to the Windows Azure data store, which is a huge enhancement. The maximum size of an Azure database is now up to 150G. Also Azure data sync allows a better hybrid model of cloud and on premise solutions

12. Big Data Support

The PASS (Professional Association for SQL Server) conference last year, Microsoft announced a partnership with Haddon provider Cloud era. One part of this involves MS releasing a ODBC driver for SQL Server that will run on a Linux platform. Additionally, Microsoft is building connectors for Haddon, which is an extremely popular NoSQL platform. With this announcement, Microsoft has made a clear move into this very rapidly growing space.

SQL 2014 is a big step forward for Microsoft -- the company is positioning itself to be a leader in availability and in the growing area of big data. As a database professional, I look forward to using SQL 2014 to bring new solutions to my clients.

System Analysis

Feasibility Study

Feasibility study is the measure of how beneficial or practical in development of an information system will be to an organization. The Feasibility analysis is a cross life cycle activity and should be continuously performed throughout the life cycle of the system. Feasibility study let developer foresee the future of the project usefullness.The study of the feasibility is done on the following factors. They are:

Operational feasibility:

By automating the Leave Management System the administrator or the HR will feel better than the manual. User will get a quick service by reducing the manual recording. Also the Administrator or the HR will feel comfortable by reduction of their work. Recording the error will be reduced and it is very easy to handle a very large database. Losing of the recording will be avoided.

Considering all the following factors we can conclude that all the users and the end users will be satisfied.

Technical feasibility:

The technical feasibility can be evaluated from technical point of view. The assessment of this feasibility must be based on an outline design of the system requirements in terms of input, output, programs and procedures.

For the system design and the development of the system, several software products have been accommodated.

Database design – Remote SQL Server

Interface Design-VISUAL BASIC 2019

This software has enough efficiency in producing the system. Therefore, the product is technically feasible.

Schedule Feasibility:

The duration of the time required for the project has been planned appropriately and it is the same duration as expected by the customer. Therefore, the project can be delivered to the customer within the expected time duration, satisfying the customer.

Hence, the project is feasible in scheduling.

Economic Feasibility:

According to the resources available and the project scheduling process it is estimated that the expenses allocated for the software to be developed by the customer is sufficient enough.

Hence, the economical factor has been considered feasible.

Behavioral Feasibility:

This includes the following questions:

- Is there sufficient support for the user?
- Will the proposed system cause harm?

This project developed will be beneficial because it satisfies its objectives when developed and installed. All behavioral as been

considered carefully and conclude that the project is behaviorally feasible.

Data Flow Diagram

Data Flow Diagram:

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD)

The idea behind the explosion of a process into more process is that understanding at one level of details is exploded into greater detailed at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

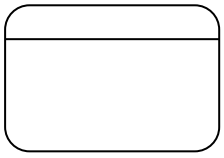
Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical form, this lead to modular design.

A DFD is known as a “bubble chart” has the purpose of clarifying system requirements and identifying major transformation they will become program in system design. So it is the starting point of the design to lowest level of details. A DFD consists of series of bubbles joined by data flows in the system.

DFD Symbols:

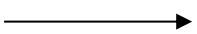
In DFD, there are four Symbols

1. A square-defines a source or destination system data
2. An arrow identified data flow. Its is the pipeline through which the information flow
3. A circle or a bubble represents a process that transforms
4. Incoming data flow into outgoing data flows
5. An open rectangle is a data source, data at rest or a temporary of data

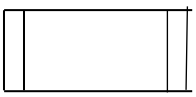


Process that transforms data flow

Source or destination of the data

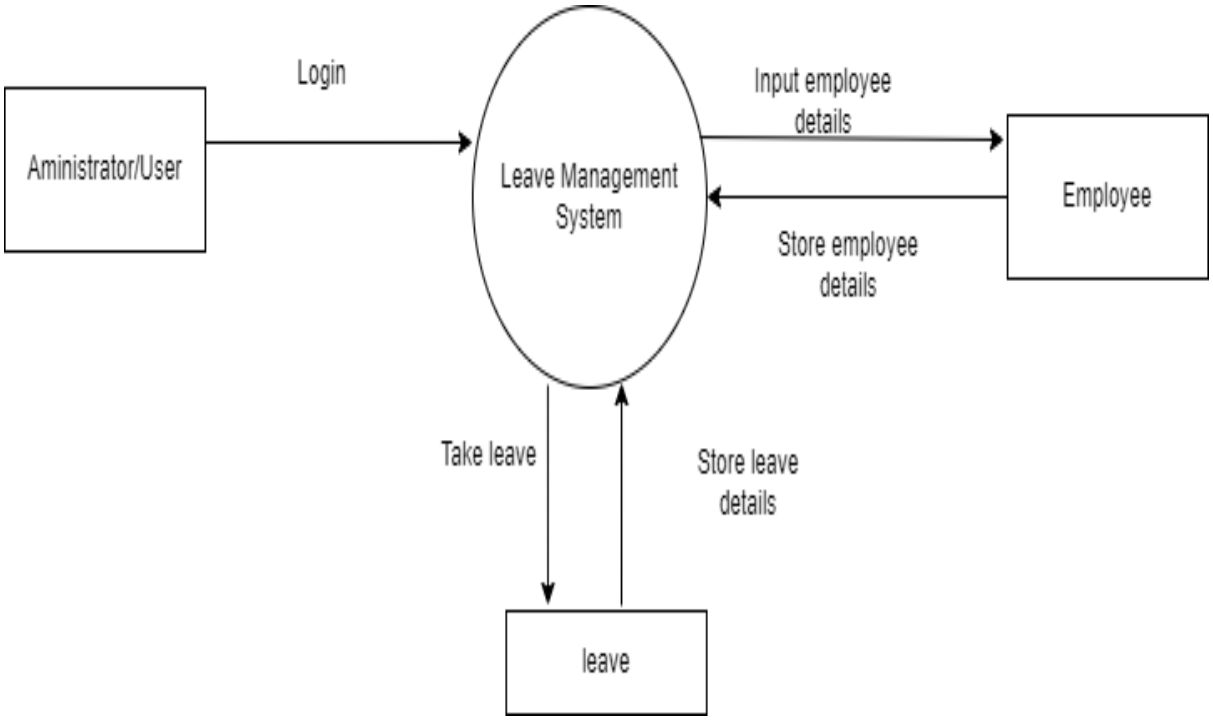


Data flow

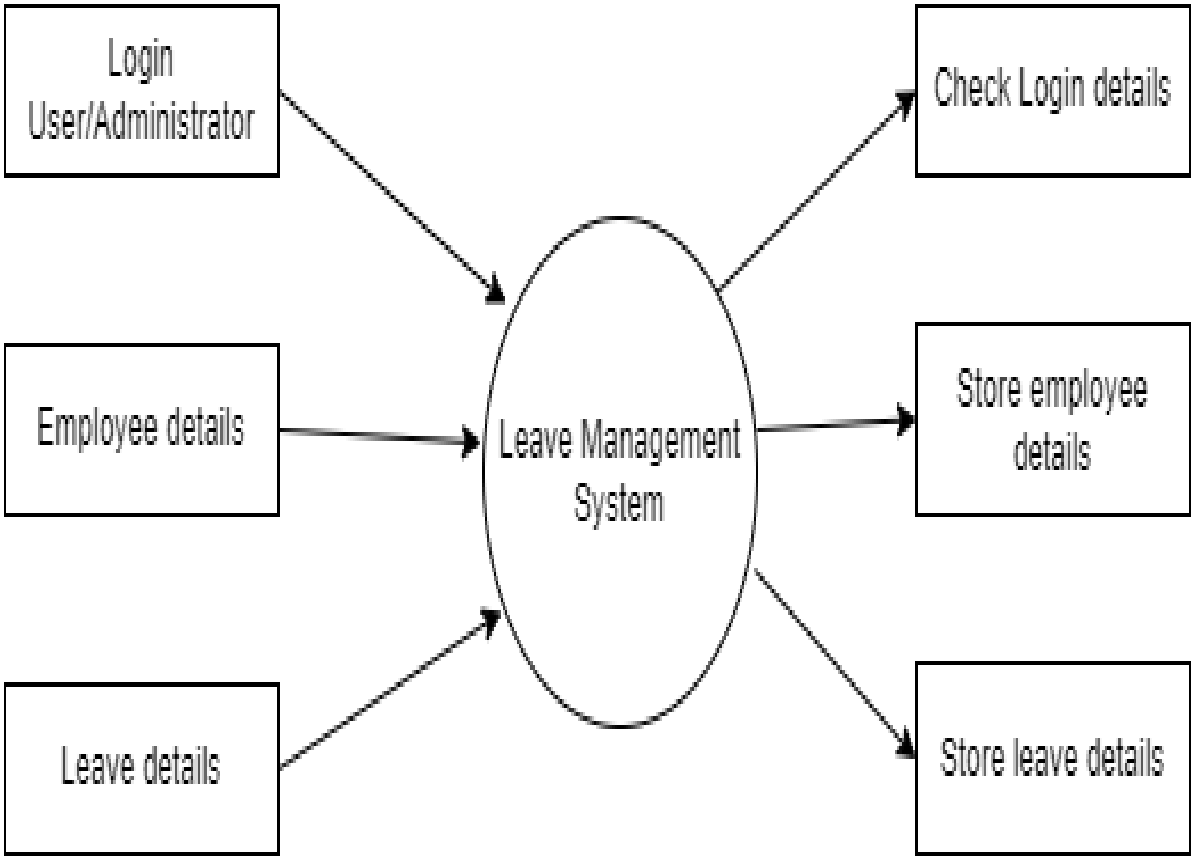


Process for data store.

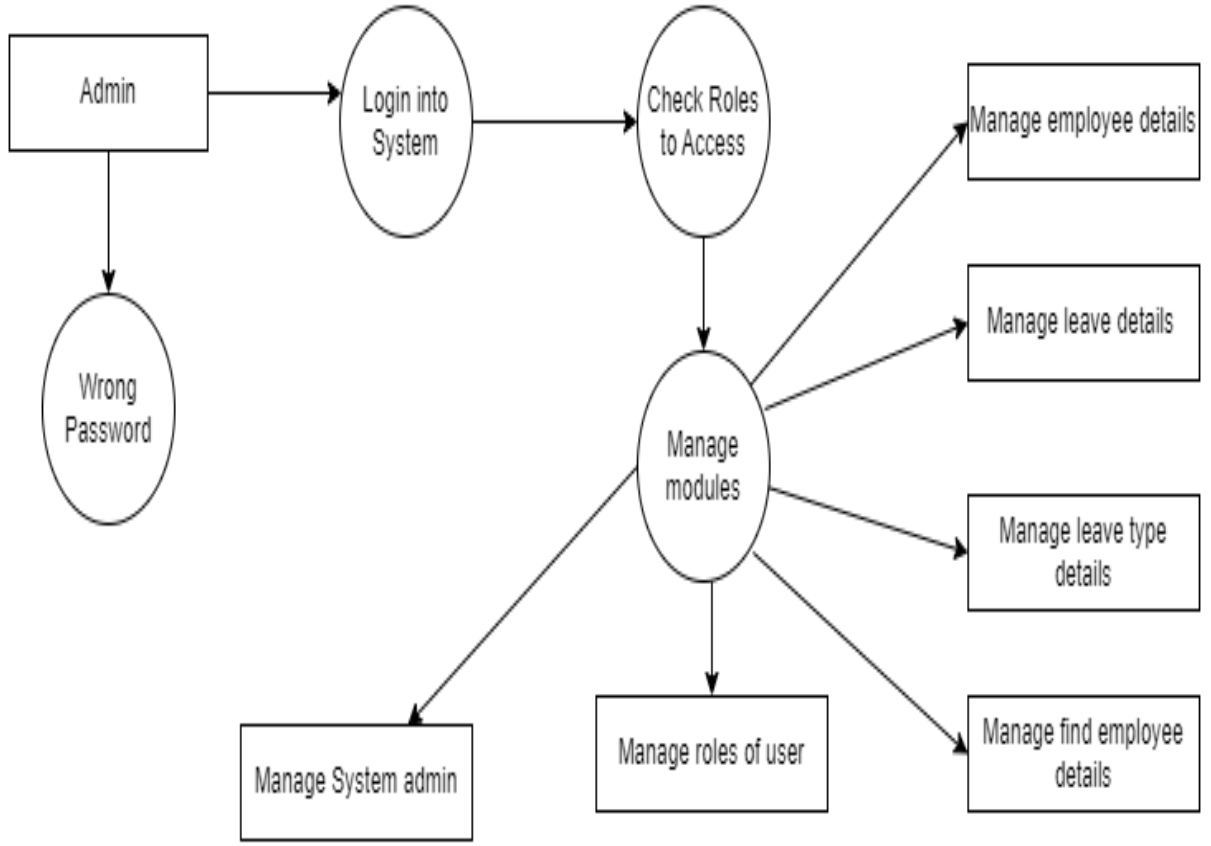
Context Level DFD:



LEVEL 1 DFD:

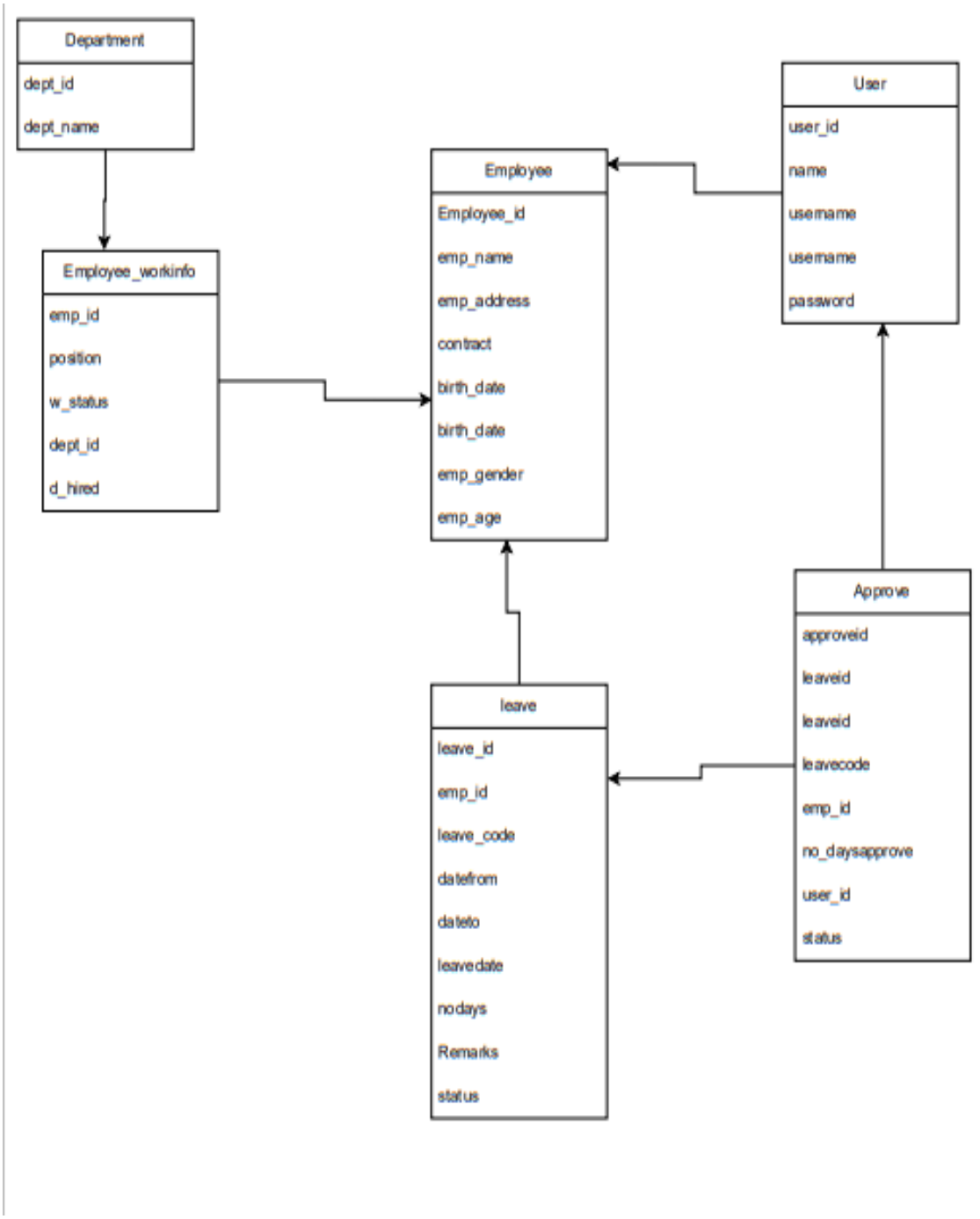

Login
User/Administrator
Employee details
Leave details
Leave Management System
Check Login details
Store employee details
Store leave details

LEVEL 2 DFD:


Admin
Login into System
Check Roles to Access
Manage employee details
Wrong Password
Manage leave details
Manage modules
Manage leave type details
Manage System admin
Manage roles of user
Manage find employee details

SCHEMA

SCHEMA:


Department
dept_id
dept_name
Employee_workinfo
emp_id
position
w_status
dept_id
d_hired
Employee
Employee_id
emp_name
emp_address
contract
birth_date
birth_date
emp_gender
emp_age
User
user_id
name
username
username
password
Approve
approveid
leaveid
leaveid
leavecode
emp_id
no_daysapprove
user_id
status
leave
leave_id
emp_id
leave_code
datefrom
dateto
leavedate
nodays
Remarks
status

SOURCE CODE

Login Module:

UI code:

```
Public Class LoginForm1

    ' TODO: Insert code to perform custom authentication using the
provided username and password
    ' The custom principal can then be attached to the current
thread's principal as follows:
    '     My.User.CurrentPrincipal = CustomPrincipal
    ' where CustomPrincipal is the IPrincipal implementation used to
perform authentication.
    ' Subsequently, My.User will return identity information
encapsulated in the CustomPrincipal object
    ' such as the username, display name, etc.


    Private Sub BunifuThinButton21_Click(sender As Object, e As
EventArgs) Handles BunifuThinButton21.Click
        login(UsernameTextBox.Text, PasswordTextBox.Text)
    End Sub

    Private Sub BunifuThinButton22_Click(sender As Object, e As
EventArgs)
        Application.Exit()
    End Sub

    Private Sub Label1_Click(sender As Object, e As EventArgs)
Handles Label1.Click
        Application.Exit()
    End Sub
End Class
```

Source Code:

```
Imports MySql.Data.MySqlClient
Module user
    Public con As MySqlConnection = mysqldb()
    Public USERID As Integer = 0
    Public Sub login(ByVal username As Object, ByVal pass As Object)
        Try

            con.Open()
            reloadtxt("SELECT * FROM tbluser WHERE username= '" &
username & "' and pass = sha1('" & pass & "')")
```

```
            If dt.Rows.Count > 0 Then
                USERID = dt.Rows(0).Item("user_id")
                If dt.Rows(0).Item("type") = "Administrator" Then
                    MsgBox("Welcome " & dt.Rows(0).Item("type"))
                    With Form1
                        .Show()
                        '.Text = "Today's user : " &
dt.Rows(0).Item("name") & "/" & dt.Rows(0).Item("type")
                    End With
                    LoginForm1.Hide()

                ElseIf dt.Rows(0).Item("type") = "HR Personnel" Then
                    MsgBox("Welcome " & dt.Rows(0).Item("type"))
                    With Form1
                        .Show()
                    End With
                    LoginForm1.Hide()
                End If

            Else
                MsgBox("Account doest not exits!",
MsgBoxStyle.Information)
            End If
        Catch ex As Exception
            Console.WriteLine(ex)
        End Try
        con.Close()
        da.Dispose()
    End Sub
    Public Sub append(ByVal sql As String, ByVal field As String,
ByVal txt As Object)
        reloadtxt(sql)
        Try
            Dim r As DataRow
            txt.AutoCompleteCustomSource.Clear()
            For Each r In dt.Rows

txt.AutoCompleteCustomSource.Add(r.Item(field).ToString)
            Next
        Catch ex As Exception
            Console.WriteLine(ex)
        End Try


    End Sub
End Module
```

Connection Code:

```
Imports MySql.Data.MySqlClient
Module connection
    Public Function mysqldb() As MySqlConnection
        Return New MySqlConnection("server=localhost;user
id=root;password=;database=db_leave;sslMode=none")
    End Function
    Public con As MySqlConnection = mysqldb()
End Module
```

```
Imports MySql.Data.MySqlClient
Module crud
    Public con As MySqlConnection = mysqldb()
    Public cmd As New MySqlCommand
    Public da As New MySqlDataAdapter
    Public dt As New DataTable
    Public ds As New DataSet
    Public sql As String
    Public result As String
    Public add As String
    Public edit As String
#Region "old crud"
    Public Sub save_or_update(ByVal sql As String, ByVal add As
String, ByVal edit As String)
        con.Open()
        With cmd
            .Connection = con
            .CommandText = sql
        End With
        dt = New DataTable
        da = New MySqlDataAdapter(sql, con)
        da.Fill(dt)
        con.Close()
        If dt.Rows.Count > 0 Then

            con.Open()
            With cmd
                .Connection = con
                .CommandText = edit
                result = cmd.ExecuteNonQuery

            End With
            con.Close()
        Else
```

```
            con.Open()
            With cmd
                .Connection = con
                .CommandText = add
                result = cmd.ExecuteNonQuery

            End With
        End If
        con.Close()
    End Sub

    Public Sub createNoMsg(ByVal sql As String)
        Try
            con.Open()
            With cmd
                .Connection = con
                .CommandText = sql
                cmd.ExecuteNonQuery()

            End With
            con.Close()
        Catch ex As Exception
            MsgBox(ex.Message & "createNoMsg")
        End Try

    End Sub
    Public Sub create(ByVal sql As String, ByVal msgsuccess As
String)
        Try
            con.Open()
            With cmd
                .Connection = con
                .CommandText = sql
                result = cmd.ExecuteNonQuery
                If result = 0 Then
                    'MsgBox(msgsuccess & " is failed to save in the
database ", MsgBoxStyle.Information)
                    MsgBox("This action cannot be performed.",
MsgBoxStyle.Information)
                Else
                    MsgBox(msgsuccess & " has been save to the
database")
                End If
            End With

        Catch ex As Exception
            MsgBox(ex.Message & " create")
        End Try
        con.Close()
```

```
    End Sub
    Public Sub reloadDtg(ByVal sql As String, ByVal dtg As
DataGridView)
        Try
            con.Open()
            With cmd
                .Connection = con
                .CommandText = sql
            End With
            dt = New DataTable
            da = New MySqlDataAdapter(sql, con)
            da.Fill(dt)
            dtg.DataSource = dt
        Catch ex As Exception
            MsgBox(ex.Message & "reloadDtg")
        End Try

        con.Close()
        da.Dispose()
    End Sub
    Public Sub reloadtxt(ByVal sql As String)
        Try
            con.Open()
            With cmd
                .Connection = con
                .CommandText = sql
            End With
            dt = New DataTable
            da = New MySqlDataAdapter(sql, con)
            da.Fill(dt)

        Catch ex As Exception
            MsgBox(ex.Message & "reloadtxt")
        End Try

        con.Close()
        da.Dispose()
    End Sub
    Public Sub updates(ByVal sql As String, ByVal msgsuccess As
String)
        Try
            con.Open()
            cmd = New MySqlCommand
            With cmd
                .Connection = con
                .CommandText = sql
                result = cmd.ExecuteNonQuery
                If result = 0 Then
```

```
                    ' MsgBox(msgsuccess & " is failed to updated in
the database.", MsgBoxStyle.Information)
                    MsgBox("This action cannot be performed.",
MsgBoxStyle.Information)
                Else
                    MsgBox(msgsuccess & " has been updated in the
database.")
                End If
            End With
            con.Close()
        Catch ex As Exception
            MsgBox(ex.Message & "updates")
        End Try

    End Sub
    Public Sub deletes(ByVal sql As String, ByVal msgsuccess As
String)
        Try
            con.Open()
            With cmd
                .Connection = con
                .CommandText = sql
            End With
            'If MessageBox.Show("Do you want to delete this
rocord?", "Delete" _
            '                       , MessageBoxButtons.YesNo,
MessageBoxIcon.Information) _
            '                       = Windows.Forms.DialogResult.Yes
Then
            result = cmd.ExecuteNonQuery
            If result = 0 Then
                ' MsgBox(msgsuccess & " is failed to delete in the
database.")
                MsgBox("This action cannot be performed.",
MsgBoxStyle.Information)
            Else
                MsgBox(msgsuccess & " has been deleted in the
database.")
            End If
            'End If
            con.Close()
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

#End Region
End Module
```

Home Module:

UI Code:

```
Module functions
    Public Sub cleartext(ByVal obj As Object)
        For Each ctrl As Control In obj.Controls
            If ctrl.GetType Is GetType(TextBox) Then
                ctrl.Text = Nothing
            End If
        Next
        For Each ctrl As Control In obj.Controls
            If ctrl.GetType Is GetType(RichTextBox) Then
                ctrl.Text = Nothing
            End If
        Next
        For Each ctrl As Control In obj.Controls
            If ctrl.GetType Is GetType(ComboBox) Then
                ctrl.Text = "----Select-----"
            End If
        Next
        For Each ctrl As Control In obj.Controls
            If ctrl.GetType Is GetType(DateTimePicker) Then
                ctrl.Text = Now
            End If
        Next

        For Each ctrl As Control In obj.Controls
            If ctrl.GetType Is GetType(RadioButton) Then
                ctrl.Controls.Clear()
            End If
        Next
    End Sub
    Public checkh As New DataGridViewCheckBoxColumn
    Public ckBox As New CheckBox()
    'Public Sub addchk(ByVal dtg As DataGridView)

    '    Try
    '        ckBox = New CheckBox()
    '        checkh = New DataGridViewCheckBoxColumn
    '        With dtg
    '            .Columns.Insert(0, checkh)
    '            .Columns(0).Width = 20
    '            .Columns(1).Width = 20
    '            .Columns(1).DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter
    '        End With
```

```
    '        'Get the column header cell bounds
    '        Dim rect = dtg.GetCellDisplayRectangle(0, -1, True)
    '        ckBox.Size = New Size(20, 20)
    '        'Change the location of the CheckBox to make it stay on
the header
    '        ckBox.Location = rect.Location

    '        AddHandler ckBox.CheckedChanged, AddressOf
ckBox_CheckedChanged

    '        'Add the CheckBox into the DataGridView
    '        dtg.Controls.Add(ckBox)
    '    Catch ex As Exception
    '        MsgBox(ex.Message)
    '    End Try

    'End Sub
    'Public Sub ckBox_CheckedChanged(ByVal sender As Object, ByVal e
As EventArgs)
    '    Try
    '        For Each row As DataGridViewRow In
frmLeaveAbsence.dtgemplist.Rows
    '            If ckBox.CheckState = CheckState.Checked Then
    '                row.Cells(0).Value = True
    '            Else
    '                row.Cells(0).Value = False
    '            End If
    '        Next
    '    Catch ex As Exception
    '        MsgBox(ex.Message)
    '    End Try
    'End Sub

    Public Sub closeChildForm()
        For Each frm As Form In Form1.MdiChildren
            frm.Close()
        Next
    End Sub
    Public Sub showForm(frm As Form)
        With frm
            .MdiParent = Form1
            .Show()

        End With
    End Sub

End Module
```

Source Code:

```
Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.Shared
Public Class Form1
    Private Sub btnLogout_Click(sender As Object, e As EventArgs)
Handles btnLogout.Click
        Me.Close()
        LoginForm1.UsernameTextBox.Clear()
        LoginForm1.PasswordTextBox.Clear()
        LoginForm1.UsernameTextBox.Focus()
        LoginForm1.Show()

    End Sub
    Public Sub addContent(frm As UserControl)

        Try
            pnlContainer.Controls.Clear()
            Dim f As New UserControl()
            f = frm
            pnlContainer.Controls.Add(f)
            f.Dock = DockStyle.Fill
            f.Visible = False
            f.BringToFront()
            animate1.ShowSync(f)
        Catch ex As Exception

        End Try

    End Sub
    Private Sub btnEmployee_Click(sender As Object, e As EventArgs)
Handles btnEmployee.Click
        addContent(FrmEmployee1)
    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        addContent(FrmHome1)
        BunifuFlatButton1.selected = True
    End Sub

    Private Sub BunifuFlatButton1_Click(sender As Object, e As
EventArgs) Handles BunifuFlatButton1.Click
        addContent(FrmHome1)
    End Sub
```

```
    Private Sub BunifuFlatButton3_Click(sender As Object, e As
EventArgs) Handles BunifuFlatButton3.Click
        addContent(FrmFindEmployee1)
    End Sub

    Private Sub BunifuFlatButton4_Click(sender As Object, e As
EventArgs) Handles BunifuFlatButton4.Click
        addContent(FrmAddLeave1)
    End Sub

    Private Sub BunifuFlatButton5_Click(sender As Object, e As
EventArgs) Handles BunifuFlatButton5.Click
        addContent(FrmSetting1)
    End Sub

    Private Sub BunifuFlatButton6_Click(sender As Object, e As
EventArgs) Handles BunifuFlatButton6.Click
        addContent(FrmUser1)
    End Sub

    Private Sub BunifuFlatButton7_Click(sender As Object, e As
EventArgs) Handles BunifuFlatButton7.Click
        addContent(FrmReport1)
    End Sub

    Private Sub BunifuImageButton1_Click(sender As Object, e As
EventArgs) Handles BunifuImageButton1.Click
        Process.Start("")
    End Sub
End Class
```

Employee Module:

```
Public Class frmEmployee

    Public empid As String = ""
    Private Sub frmEmployee_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        fillcbo("SELECT * FROM `tblsettings` WHERE
`FORTHE`='Position'", txtposition)
        cleartext(GroupBox9)
        cleartext(GroupBox10)
        fillcbo("SELECT `DEPARTMENT` FROM `tbldepartment` ",
cbodeaprtment)

        Try

            sql = "SELECT * FROM `employee` e, `employee_workinfo`
ew " _
                & " WHERE e.`EMPID`=ew.`EMPID` AND e.EMPID ='" &
txtcode.Text & "'"
            reloadtxt(sql)

            If dt.Rows.Count > 0 Then



                txtdrate.Text = dt.Rows(0).Item("d_rate")
                txtpmethod.Text = dt.Rows(0).Item("p_method")
                txtposition.Text = dt.Rows(0).Item("position")
                dtpdhired.Value = dt.Rows(0).Item("d_hired")

                txtfname.Text = dt.Rows(0).Item("emp_fname")
                txtlname.Text = dt.Rows(0).Item("emp_lname")
                txtmname.Text = dt.Rows(0).Item("emp_mname")
                txtaddress.Text = dt.Rows(0).Item("address")
                txtcontact.Text = dt.Rows(0).Item("contact")
                txtstatus.Text = dt.Rows(0).Item("status")
                dtpdbirth.Value = dt.Rows(0).Item("birth_date")
                txtbplace.Text = dt.Rows(0).Item("birth_place")
                If dt.Rows(0).Item("emp_sex") = "MALE" Then
                    rdomale.Checked = True
                Else
```

```
                    rdofemale.Checked = True
                End If
                txtemerg.Text = dt.Rows(0).Item("emerg_contct")
                cbodeaprtment.Text = dt.Rows(0).Item("DEPARTMENT")
                cbowtype.Text = dt.Rows(0).Item("w_type")
                'Else
                '    cleartext(GroupBox10)
                '    cleartext(GroupBox9)
            Else

                loadautonumber("employee", txtcode)
            End If
            'aloadautonumber("employee", txtcode)
        Catch ex As Exception
            Console.WriteLine(ex)
        End Try

        'If empid = "" Then
        '    loadautonumber("employee", txtcode)
        'Else
        '    txtcode.Text = empid
        'End If
    End Sub

    Private Sub btnSave_Click(sender As Object, e As EventArgs)
Handles btnSave.Click
        Try
            'loadautonumber("employee", txtcode)
            For Each ctrl As Control In GroupBox9.Controls
                If ctrl.GetType Is GetType(TextBox) Then
                    If ctrl.Text = "" Then
                        MsgBox("One of the box is empty. It needed
to be filled up.", MsgBoxStyle.Exclamation)
                        Return
                    End If
                End If
                If ctrl.GetType Is GetType(ComboBox) Then
                    If ctrl.Text = "----Select-----" Then
                        MsgBox("You have to set the correct
information.", MsgBoxStyle.Exclamation)
                        Return
                    End If
                End If
            Next

            For Each ctrl As Control In GroupBox10.Controls
                If ctrl.GetType Is GetType(ComboBox) Then
                    If ctrl.Text = "----Select-----" Then
```

```
                        MsgBox("You have to set the correct
information.", MsgBoxStyle.Exclamation)
                        Return
                    End If
                End If

                If ctrl.GetType Is GetType(TextBox) Then
                    If ctrl.Text = "" Then
                        MsgBox("One of the box is empty. It needs to
be filled up.", MsgBoxStyle.Exclamation)
                        Return
                    End If
                End If
            Next

            Dim bdate As Integer =
Math.Round(DateDiff(DateInterval.DayOfYear, dtpdbirth.Value, Now) /
12 / 31)
            If bdate < 18 Then
                MsgBox("Invalid birth of date.",
MsgBoxStyle.Exclamation)
                Exit Sub
            End If


            '-------------------------------------------------------
--
            Dim rdo As String = ""
            sql = "SELECT * FROM `employee` e, `employee_workinfo`
ew " _
                & " WHERE e.`EMPID`=ew.`EMPID` AND e.EMPID ='" &
txtcode.Text & "'"
            reloadtxt(sql)
            If dt.Rows.Count > 0 Then
                '-------------------update
                If rdomale.Checked = True Then
                    rdo = "MALE"
                Else
                    rdo = "FEMALE"
                End If

                sql = "UPDATE `employee_workinfo` SET  `d_rate`='" &
txtdrate.Text _
                                    & "', `p_method`='" &
txtpmethod.Text & "', `position`='" & txtposition.Text _
                                    & "', `d_hired`='" &
Format(dtpdhired.Value, "yyyy-MM-dd") &
                                    "', `DEPARTMENT`='" &
cbodeaprtment.Text &
```

```
                                "',`w_type`='" & cbowtype.Text & "'
WHERE `EMPID`='" & txtcode.Text & "'"
                createNoMsg(sql)

                sql = "UPDATE `employee` SET  `emp_fname`='" &
txtfname.Text _
                & "', `emp_lname`='" & txtlname.Text & "',
`emp_mname`='" & txtmname.Text _
                & "', `address`='" & txtaddress.Text & "',
`contact`='" & txtcontact.Text & "', `status`='" & txtstatus.Text _
                & "', `birth_date`='" & Format(dtpdbirth.Value,
"yyyy-MM-dd") & "', `birth_place`='" & txtbplace.Text & "',
`emp_sex`='" & rdo _
                & "' , `emerg_contct`='" & txtemerg.Text _
                & "'  WHERE `EMPID`='" & txtcode.Text & "'"
                updates(sql, txtlname.Text)



                '--------------end update
            Else
                '------------------insert
                If rdomale.Checked = True Then
                    rdo = "MALE"
                Else
                    rdo = "FEMALE"
                End If
                sql = "INSERT INTO `employee_workinfo` (`EMPID`,
`d_rate`, `p_method`, `position`,   `d_hired`,DEPARTMENT,w_type)" _
                        & " VALUES ('" & txtcode.Text & "','" &
txtdrate.Text & "','" & txtpmethod.Text & "','" & txtposition.Text _
                        & "', '" & Format(dtpdhired.Value, "yyyy-MM-
dd") & "','" & cbodeaprtment.Text & "','" & cbowtype.Text & "')"
                createNoMsg(sql)
                '-----------------------------------------------------
--------------------------
                sql = "INSERT INTO `employee` (`EMPID`, `emp_fname`,
`emp_lname`, `emp_mname`" _
                & ", `address`, `contact`, `status`, `birth_date`,
`birth_place`, `emp_sex`" _
                & ", `emerg_contct`,`REMAININGLEAVE`,`DEFAULTLEAVE`)
VALUES ('" & txtcode.Text & "','" & txtfname.Text & "','" &
txtlname.Text _
                & "','" & txtmname.Text & "','" & txtaddress.Text &
"','" & txtcontact.Text & "','" & txtstatus.Text _
                & "','" & Format(dtpdbirth.Value, "yyyy-MM-dd") &
"','" & txtbplace.Text & "','" & rdo & "','" & txtemerg.Text &
"',30,30)"
                create(sql, txtfname.Text & " " & txtlname.Text)
```

```
                ''-----------------------------------------------
                sql = "INSERT INTO `tblleaveinfo`  (`EMPID`,
`LEAVEDAYS`, `REASONS`)" _
                & " VALUES ('" & txtcode.Text & "',15,'SICK'),('" &
txtcode.Text & "',15,'Vacation')"
                createNoMsg(sql)
                '-----------------------------------------------
                updateautonumber("employee")

                cleartext(GroupBox9)
                cleartext(GroupBox10)
                loadautonumber("employee", txtcode)
                emptitle.Text = "Add New Employee"
                '---------------end insert
            End If




        Catch ex As Exception
            Console.WriteLine(ex)
        End Try
    End Sub

    Private Sub btnNew_Click(sender As Object, e As EventArgs)
Handles btnNew.Click
        cleartext(GroupBox9)
        cleartext(GroupBox10)
        loadautonumber("employee", txtcode)
        emptitle.Text = "Add New Employee"
    End Sub

    Private Sub txtcode_TextChanged(sender As Object, e As
EventArgs) Handles txtcode.TextChanged
        Try

            sql = "SELECT * FROM `employee` e, `employee_workinfo`
ew " _
                & " WHERE e.`EMPID`=ew.`EMPID` AND e.EMPID ='" &
txtcode.Text & "'"
            reloadtxt(sql)

            If dt.Rows.Count > 0 Then




                txtdrate.Text = dt.Rows(0).Item("d_rate")
```

```
                txtpmethod.Text = dt.Rows(0).Item("p_method")
                txtposition.Text = dt.Rows(0).Item("position")
                dtpdhired.Value = dt.Rows(0).Item("d_hired")

                txtfname.Text = dt.Rows(0).Item("emp_fname")
                txtlname.Text = dt.Rows(0).Item("emp_lname")
                txtmname.Text = dt.Rows(0).Item("emp_mname")
                txtaddress.Text = dt.Rows(0).Item("address")
                txtcontact.Text = dt.Rows(0).Item("contact")
                txtstatus.Text = dt.Rows(0).Item("status")
                dtpdbirth.Value = dt.Rows(0).Item("birth_date")
                txtbplace.Text = dt.Rows(0).Item("birth_place")
                If dt.Rows(0).Item("emp_sex") = "MALE" Then
                    rdomale.Checked = True
                Else
                    rdofemale.Checked = True
                End If
                txtemerg.Text = dt.Rows(0).Item("emerg_contct")
                cbodeaprtment.Text = dt.Rows(0).Item("DEPARTMENT")
                cbowtype.Text = dt.Rows(0).Item("w_type")
                'Else
                '    cleartext(GroupBox10)
                '    cleartext(GroupBox9)

            End If
            'aloadautonumber("employee", txtcode)
        Catch ex As Exception
            Console.WriteLine(ex)
        End Try
    End Sub
End Class
```

Find Employee Module:

```
Public Class frmFindEmployee
    Private Sub frmFindEmployee_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
        sql = "SELECT `EMPID` AS 'Employee Id',`emp_fname` as 'First
Name', `emp_lname` as 'Last Name',`emp_mname` AS 'Middle Name'" _
       & ",round( ((DATEDIFF( NOW( ) ,  `birth_date` ) /12) /31))
AS 'Age', `emp_sex` AS 'Gender', `status` AS 'Status', `address` AS
'Address', `contact` AS 'Contact Number'  FROM `employee`"
        reloadDtg(sql, dtgemplist)
    End Sub

    Private Sub txtempsearch_TextChanged(sender As Object, e As
EventArgs) Handles txtempsearch.TextChanged
        sql = "SELECT `EMPID` AS 'Employee Id',`emp_fname` as 'First
Name', `emp_lname` as 'Last Name',`emp_mname` AS 'Middle Name'" _
     & ", round( ((DATEDIFF( NOW( ) ,  `birth_date` ) /12) /31))  AS
'Age', `emp_sex` AS 'Gender', `status` AS 'Status', `address` AS
'ADDRESS'" _
     & ", `contact` AS 'CONTACT'  FROM `employee` WHERE `EMPID` LIKE
'%" & txtempsearch.Text & "%'" _
     & " OR emp_fname LIKE '%" & txtempsearch.Text & "%' OR
emp_lname LIKE '%" & txtempsearch.Text & "%'"
        reloadDtg(sql, dtgemplist)
    End Sub

    Private Sub btnPrintAll_Click(sender As Object, e As EventArgs)
        sql = "SELECT  e.`EMPID`, concat( `emp_fname`,' ',
`emp_lname`,' ', `emp_mname`) as 'Name' ,`emp_sex`,(`d_rate` * 14)
as 'TwooWeeksSalary', `position`, `DEPARTMENT` FROM `employee` e
,`employee_workinfo` we  WHERE e.`EMPID`=we.`EMPID`"
        reports(sql, "allemployees",
frmprint_emp.CrystalReportViewer1)
        frmprint_emp.ShowDialog()
    End Sub

    Private Sub btnPrintEmp_Click(sender As Object, e As EventArgs)
        sql = "SELECT  e.`EMPID`, `emp_fname`, `emp_lname`,
`emp_mname`,`address`, `contact`, `status`, `emp_sex`, round(
((DATEDIFF( NOW( ) ,  `birth_date` ) /12) /31))  AS 'Age',`d_rate`,
`position`, `d_hired`,(`d_rate` * 15) as 'Salary', `DEPARTMENT`,
`REMAININGLEAVE`, `DEFAULTLEAVE`  ,w_type" &
            " FROM  `employee` e,  `employee_workinfo` w " &
```

```
            " WHERE e.EMPID = w.EMPID AND e.EMPID='" &
dtgemplist.CurrentRow.Cells(0).Value & "'"
        reports(sql, "selectedemployee",
frmprint_emp.CrystalReportViewer1)
        frmprint_emp.ShowDialog()
    End Sub

    Private Sub btnEdit_Click(sender As Object, e As EventArgs)
Handles btnEdit.Click
        Try
            Form1.addContent(Form1.FrmEmployee1)
            Form1.FrmEmployee1.emptitle.Text = "Update Employee"
            Form1.FrmEmployee1.txtcode.Text =
dtgemplist.CurrentRow.Cells(0).Value
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub btnDelete_Click(sender As Object, e As EventArgs)
Handles btnDelete.Click
        sql = "DELETE FROM employee WHERE EMPID = '" &
dtgemplist.CurrentRow.Cells(0).Value & "'"
        createNoMsg(sql)
        sql = "DELETE FROM employee_workinfo WHERE EMPID = '" &
dtgemplist.CurrentRow.Cells(0).Value & "'"
        deletes(sql, dtgemplist.CurrentRow.Cells(1).Value)

        sql = "SELECT `EMPID` AS 'Employee Id',`emp_fname` as 'First
Name', `emp_lname` as 'Last Name',`emp_mname` AS 'Middle Name'" _
        & ",round( ((DATEDIFF( NOW( ) ,  `birth_date` ) /12) /31))
AS 'Age', `emp_sex` AS 'Gender', `status` AS 'Status', `address` AS
'Address', `contact` AS 'Contact Number'  FROM `employee`"
        reloadDtg(sql, dtgemplist)
    End Sub
End Class
```

Leave Of Absence Module:

```
Public Class frmAddLeave
    Private Sub btnFind_Click(sender As Object, e As EventArgs) Handles btnFind.Click
        With frmviewEmployee
            .ShowDialog()
        End With
    End Sub

    Private Sub txtEmployeeId_TextChanged(sender As Object, e As EventArgs) Handles
txtEmployeeId.TextChanged
        Try

            sql = "SELECT `d_rate`, `position`, `DEPARTMENT` FROM `employee` e
,`employee_workinfo` ew WHERE e.`EMPID`=ew.`EMPID` AND e.`EMPID`='" &
txtEmployeeId.Text & "'"
            reloadtxt(sql)
            If dt.Rows.Count > 0 Then
                With dt.Rows(0)
                    txtposition.Text = .Item("position")
                    txtsalary.Text = .Item("d_rate")
                    txtdepartment.Text = .Item("DEPARTMENT")
                End With
                sql = "SELECT * FROM `employee_workinfo` WHERE `EMPID` ='" &
txtemid.Text & "'"
                reloadtxt(sql)

                If dt.Rows(0).Item("w_type") = "Regular" Then
                    rdowithPay.Enabled = True
```

```
            Else
                rdowithPay.Enabled = False
            End If


        Else
            txtEmployeeId.Clear()
            txtposition.Clear()
            txtsalary.Clear()
            txtdepartment.Clear()
        End If


    Catch ex As Exception
        '  MsgBox(ex.Message)
    End Try
End Sub

Private Sub btnSave_Click(sender As Object, e As EventArgs) Handles btnSave.Click
    Try


        'For Each grp As Control In Me.Controls
        '    If TypeOf grp Is GroupBox Then
        '        For Each ctrl As Control In grp.Controls
        '            If TypeOf ctrl Is TextBox Then
        '                If ctrl.Text = "" Then
        '                    MessageBox.Show("Please put information in " & ctrl.Tag, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        '                    Exit Sub
        '                End If
        '            End If
        '            If TypeOf ctrl Is RichTextBox Then
        '                If ctrl.Text = "" Then
```

```
        '                MessageBox.Show("Please put information in " & ctrl.Tag, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        '                Exit Sub
        '              End If
        '           End If

        '       Next
        '    End If
        'Next
        If rdoAcidentOnDuty.Checked = False And rdoPaternity.Checked = False And
rdoMaternity.Checked = False _
         And rdoVacation.Checked = False And rdoFuneral.Checked = False And
rdoSick.Checked = False Then
            MessageBox.Show("Please choose your leave applied for.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If

        "---------------------------------
        Dim rdoleaveformat As String = ""
        Dim rdoleaveapplied As String = ""

        If rdoWithoutPay.Checked = True Then
            rdoleaveformat = "Without Pay"
        ElseIf rdowithPay.Checked = True Then
            rdoleaveformat = "With Pay"
        End If

        ' -------------------------
        If rdoSick.Checked = True Then
            rdoleaveapplied = "Sick"
        ElseIf rdoVacation.Checked = True Then
```

```
    rdoleaveapplied = "Vacation"
ElseIf rdoFuneral.Checked = True Then
    rdoleaveapplied = "Funeral"
ElseIf rdoPaternity.Checked = True Then
    rdoleaveapplied = "Paternity"
ElseIf rdoMaternity.Checked = True Then
    rdoleaveapplied = "Maternity"
ElseIf rdoAcidentOnDuty.Checked = True Then
    rdoleaveapplied = "AccidentOnDuty"
End If
"--------------------------------------------------
Dim day As Double
Dim numdays As Integer
Dim numtime As Integer
numtime = DateDiff(DateInterval.Hour, dtpTimeFrom.Value, dtpTimeTo.Value)
' MsgBox(numtime)
numdays = DateDiff(DateInterval.Day, dtpdatestart.Value, dtpenddate.Value)
'MsgBox(numdays)
If numdays = 0 Then
    If numtime >= 0 Then
        day = 0.5
    ElseIf numtime = 12 Then
        day = 1
    End If
Else
    day = numdays
End If

MsgBox(day)
```

```
        "----------------------------------------

        sql = "INSERT INTO `leave` (`EMPID`,`LEAVECODE`,`LEAVEFORMAT`,
`LEAVEAPPLIED`, `DATEFROM`, `DATETO`,`LEAVEDATE`,`LEAVEENDDATE`,
`NODAYS`, `REASON`,  `DAYOFFSCHEDULE`,`REMARKS`,`APPLIED`,`STATUS`) "
_
        & "VALUES ('" & txtEmployeeId.Text & "','" & lblcode.Text & "','" &
rdoleaveformat & "','" & rdoleaveapplied & "','" & Format(dtpTimeFrom.Value, "yyyy-MM-
dd hh:mm:ss tt") &
        "','" & Format(dtpTimeTo.Value, "yyyy-MM-dd hh:mm:ss tt") & "','" &
Format(dtpdatestart.Value, "yyyy-MM-dd hh:mm:ss tt") & "','" & Format(dtpenddate.Value,
"yyyy-MM-dd hh:mm:ss tt") & "'," & day &
        ",'" & txtreasons.Text & "','" & Format(dtpNotfallWeekened.Value, "yyyy-MM-dd
hh:mm:ss tt") & "','Approved',1,'Approved')"
        create(sql, "New Leave")



        "----------------------------------------

        sql = "UPDATE `employee` set `ONLEAVE`=1 ,`REMAININGLEAVE`
=`REMAININGLEAVE`- " & day & " WHERE `EMPID`='" & txtEmployeeId.Text & "'"
        createNoMsg(sql)
        '----------------------------------------
        updateautonumber("applicationcode")
        Call btn_new_Click(sender, e)
      Catch ex As Exception
        MsgBox(ex.Message)
      End Try
    End Sub

    Private Sub btn_new_Click(sender As Object, e As EventArgs) Handles btn_new.Click
      cleartext(GroupBox1)
      cleartext(GroupBox2)
      cleartext(GroupBox3)
      cleartext(GroupBox4)
      cleartext(GroupBox5)
```

```
        cleartext(Me)


        'For Each rdo As Control In GroupBox2.Controls
        '    If TypeOf rdo Is RadioButton Then
        '        rdo.Enabled = False
        '    End If
        '    If TypeOf rdo Is RadioButton Then
        '        TryCast(rdo, RadioButton).Checked = False
        '    End If
        'Next



        'rdoWithoutPay.Checked = False
        'rdowithPay.Checked = False


        sql = "SELECT  e.`EMPID` as 'Employee Id', concat( `emp_fname`,' ', `emp_lname`) as
'Name',`LEAVEFORMAT` as 'Status', `LEAVEAPPLIED` as 'Applied Leave',
TIME(`DATEFROM`) as 'From', TIME(`DATETO`) as 'To' ,DATE(`LEAVEDATE`) as
'Start of Date Leave', DATE(LEAVEENDDATE) as 'End of Date Leave',`NODAYS` as 'No.
Days', `REASON` as 'Reasons'  FROM `employee` e,`leave` l  WHERE
e.`EMPID`=l.`EMPID`"
        reloadDtg(sql, dtgapprovedlist)
    End Sub


    Private Sub dtpenddate_ValueChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles dtpenddate.ValueChanged, dtpdatestart.ValueChanged
        Try


            txtnoDays.Text = DateDiff(DateInterval.Day, dtpdatestart.Value, dtpenddate.Value)


            Dim numdays As Integer
```

```
        numdays = DateDiff(DateInterval.Day, dtpdatestart.Value, dtpenddate.Value)


        If numdays > 0 Then
            dtpTimeFrom.Enabled = False
            dtpTimeTo.Enabled = False
        Else
            dtpTimeFrom.Enabled = True
            dtpTimeTo.Enabled = True
        End If
    Catch ex As Exception


    End Try
End Sub

Private Sub AddLeave(sender As Object, e As EventArgs) Handles MyBase.Load
     Call btn_new_Click(sender, e)
    Try
        'For Each rdo As Control In GroupBox2.Controls
        '    If TypeOf rdo Is RadioButton Then
        '        rdo.Enabled = False
        '    End If
        'Next
        '-------------------------------------
        dtpTimeFrom.Format = DateTimePickerFormat.Time
        dtpTimeFrom.ShowUpDown = True
        '---------------------------------
        dtpTimeTo.Format = DateTimePickerFormat.Time
        dtpTimeTo.ShowUpDown = True
        '---------------------------------
        loadautonumber("applicationcode", lblcode)
```

```
            '----------------------------------------
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub txtemid_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtemid.KeyPress,
txtnoDays.KeyPress

        '97 - 122 = Ascii codes for simple letters
        '65 - 90  = Ascii codes for capital letters
        '48 - 57  = Ascii codes for numbers

        If Asc(e.KeyChar) <> 8 Then
            If Asc(e.KeyChar) < 48 Or Asc(e.KeyChar) > 57 Then
                e.Handled = True
            End If
        End If
    End Sub

    Private Sub txtapprovesearch_TextChanged(sender As Object, e As EventArgs) Handles
txtapprovesearch.TextChanged
        Try
            '---------------------------------approved list.
            sql = "SELECT  e.`EMPID` as 'Employee Id', concat( `emp_fname`,' ', `emp_lname`)
as 'Name',`LEAVEFORMAT` as 'Status', `LEAVEAPPLIED` as 'Applied Leave',
TIME(`DATEFROM`) as 'From', TIME(`DATETO`) as 'To' , DATE(`LEAVEDATE`) as
'Start of Date Leave',`LEAVEENDDATE` as 'End of Date Leave', `NODAYS` as 'No. Days',
`REASON` as 'Reasons'   FROM `employee` e,`leave` l  WHERE e.`EMPID`=l.`EMPID` "
&
        " AND (e.`EMPID` LIKE '%" & txtapprovesearch.Text & "%' OR concat(
`emp_fname`,' ', `emp_lname`) LIKE '%" & txtapprovesearch.Text & "%')"
            reloadDtg(sql, dtgapprovedlist)
```

```
            'sql = "SELECT LEAVECODE, e.`EMPID` as 'Employee Id', concat( `emp_fname`,'
', `emp_lname`) as 'Name',`LEAVEFORMAT` as 'Status', `LEAVEAPPLIED` as 'Applied
Leave', TIME(`DATEFROM`) as 'From', TIME(`DATETO`) as 'To' ,
DATE(`LEAVEDATE`) as 'Date of Leave', `NODAYS` as 'No. Days', `REASON` as
'Reasons', `DAYOFFSCHEDULE` as 'Dayoff Schedule'   FROM `employee` e,`leave` l
WHERE   AND e.`EMPID`=l.`EMPID`" & _

            "' AND (e.`EMPID` LIKE '%" & txtapprovesearch.Text & "%' OR concat(
`emp_fname`,' ', `emp_lname`) LIKE '%" & txtapprovesearch.Text & "%')"

            'reloadDtg(sql, dtgapprovedlist)

            'dtgapprovedlist.Columns(0).Visible = False

        Catch ex As Exception

            MsgBox(ex.Message)

        End Try

    End Sub

End Class
```

Settings Module:

```
Imports MySql.Data.MySqlClient
Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.Shared
Module selects
    Public con As MySqlConnection = mysqldb()
    'procedure of your autoappend and autosugest
    Public Sub autocompletetxt(ByVal sql As String, ByVal txt As
TextBox)
        Try
            dt = New DataTable
            'OPENING THE CONNECTION
            con.Open()
            'HOLDS THE DATA TO BE EXECUTED
            With cmd
                .Connection = con
                .CommandText = sql
            End With
            'FILLING THE DATA IN THE DATATABLE
            da.SelectCommand = cmd
            da.Fill(dt)
            'SET A VARIABLE AS A ROW OF DATA IN THE DATATABLE
            Dim r As DataRow
            'CLEARING THE AUTOCOMPLETE SOURCE OF THE TEXTBOX
            txt.AutoCompleteCustomSource.Clear()
            'LOOPING THE ROW OF DATA IN THE DATATABLE
            For Each r In dt.Rows
                'ADDING THE DATA IN THE AUTO COMPLETE SOURCE OF THE
TEXTBOX
                txt.AutoCompleteCustomSource.Add(r.Item(0).ToString)
            Next
            ''''''''''''''''''''''''
        Catch ex As Exception
            Console.WriteLine(ex)
        End Try
        'CLOSING THE CONNECTION
        con.Close()
        da.Dispose()
    End Sub
    Public Sub auto_sugestAll(ByVal sql As String, ByVal txt As
Object)
        With cmd
            .Connection = con
            .CommandText = sql
        End With
        dt = New DataTable
        da = New MySqlDataAdapter(sql, con)
        da.Fill(dt)
```

```
        Dim r As DataRow
        txt.AutoCompleteCustomSource.Clear()
        For Each r In dt.Rows
            txt.AutoCompleteCustomSource.Add(r.Item(0).ToString)
            txt.AutoCompleteCustomSource.Add(r.Item(1).ToString)
            txt.AutoCompleteCustomSource.Add(r.Item(2).ToString)
        Next
    End Sub
    Public Sub fillcbo(ByVal sql As String, ByVal cbo As ComboBox)
        Try
            dt = New DataTable
            'OPENING THE CONNECTION
            con.Open()
            'HOLDS THE DATA TO BE EXECUTED
            With cmd
                .Connection = con
                .CommandText = sql
            End With
            'FILLING THE DATA IN THE DATATABLE
            da.SelectCommand = cmd
            da.Fill(dt)
            With cbo
                .DataSource = dt
                .DisplayMember = "DESCRIPTION"
                .DisplayMember = "DEPARTMENT"
            End With
        Catch ex As Exception
            Console.WriteLine(ex)
        End Try
        'CLOSING THE CONNECTION
        con.Close()
        da.Dispose()
    End Sub

    Public Sub loadautonumber(ByVal desc As String, ByVal txt As
Object)
        Try
            sql = "SELECT concat(`STRT`, `END`) FROM `tblautonumber`
WHERE `DESCRIPTION`= '" & desc & "'"
            reloadtxt(sql)
            txt.Text = dt.Rows(0).Item(0)
        Catch ex As Exception
            Console.WriteLine(ex)
        End Try

    End Sub
    Public Sub updateautonumber(ByVal desc As String)
        Try
```

```
            sql = "UPDATE `tblautonumber` SET
`END`=`END`+`INCREMENT` WHERE `DESCRIPTION`='" & desc & "'"
            createNoMsg(sql)
        Catch ex As Exception
            Console.WriteLine(ex)
        End Try
    End Sub
#Region "Report"
    Public Sub reports(ByVal sql As String, ByVal rptname As String,
ByVal crystalRpt As Object)
        Try
            con.Open()

            Dim reportname As String
            With cmd
                .Connection = con
                .CommandText = sql
            End With
            ds = New DataSet
            da = New MySqlDataAdapter(sql, con)
            da.Fill(ds)
            reportname = rptname
            Dim reportdoc As New
CrystalDecisions.CrystalReports.Engine.ReportDocument
            Dim strReportPath As String
            strReportPath = Application.StartupPath('C:\Program
Files (x86)\SAP BusinessObjects\Crystal Reports for .NET Framework
4.0\Common\SAP BusinessObjects Enterprise XI 4.0\win64_x64') &
"\reports\" & reportname & ".rpt"
            With reportdoc
                .Load(strReportPath)
                .SetDataSource(ds.Tables(0))
            End With
            With crystalRpt
                ' .ShowRefreshButton = False
                .ShowCloseButton = False
                .ShowGroupTreeButton = False

                .ReportSource = reportdoc
            End With
        Catch ex As Exception
            MsgBox(ex.Message & "No Crystal Reports have been
Installed")
        End Try
        con.Close()
        da.Dispose()
    End Sub
#End Region
```

```
End Module


Public Class frmSetting
    Dim positionID As Integer = 0
    Dim departmentid As Integer = 0
    Private Sub frmSetting_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        Try
            sql = "SELECT ID,`DESCRIPTION` as Position FROM
`tblsettings` WHERE `FORTHE`='Position'"
            reloadDtg(sql, dtglistposition)
            txtposition.Clear()
            dtglistposition.Columns(0).Visible = False

            sql = "SELECT ID,`DEPARTMENT` FROM `tbldepartment` "
            reloadDtg(sql, dtgdeptlist)
            txtdepartment.Clear()
            dtgdeptlist.Columns(0).Visible = False

            btn_delete_dept.Enabled = False
            btn_Delete_Position.Enabled = False
            btn_update_dept.Enabled = False
            btn_update_Position.Enabled = False
            btn_save_dept.Enabled = True
            btn_save_Position.Enabled = True

            dtgdeptlist.DefaultCellStyle.Font = New Font("arial", 8)
            dtglistposition.DefaultCellStyle.Font = New
Font("arial", 8)
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try

    End Sub

    Private Sub btn_save_Position_Click(sender As Object, e As
EventArgs) Handles btn_save_Position.Click
        Try
            sql = "INSERT INTO `tblsettings` (`DESCRIPTION`,
`FORTHE`) VALUES ('" & txtposition.Text & "','Position')"
            create(sql, "New Position")
            Call frmSetting_Load(sender, e)
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub btn_update_Position_Click(sender As Object, e As
EventArgs) Handles btn_update_Position.Click
```

```
        sql = "UPDATE `tblsettings` SET `DESCRIPTION`='" &
txtposition.Text & "' WHERE  `ID`=" & positionID
        updates(sql, "Position")
        Call frmSetting_Load(sender, e)
    End Sub

    Private Sub btn_Delete_Position_Click(sender As Object, e As
EventArgs) Handles btn_Delete_Position.Click
        sql = "DELETE FROM `tblsettings` WHERE `ID`=" &
dtglistposition.CurrentRow.Cells(0).Value
        deletes(sql, "Position")
        Call frmSetting_Load(sender, e)
    End Sub

    Private Sub btn_save_dept_Click(sender As Object, e As
EventArgs) Handles btn_save_dept.Click
        sql = "INSERT INTO `tbldepartment` (`DEPARTMENT`) VALUES ('"
& txtdepartment.Text & "')"
        create(sql, "New Department")
        Call frmSetting_Load(sender, e)
    End Sub

    Private Sub btn_update_dept_Click(sender As Object, e As
EventArgs) Handles btn_update_dept.Click
        sql = "UPDATE `tbldepartment`  SET `DEPARTMENT`='" &
txtdepartment.Text & "' WHERE `ID`=" & departmentid
        updates(sql, "Department")
        Call frmSetting_Load(sender, e)
    End Sub

    Private Sub btn_delete_dept_Click(sender As Object, e As
EventArgs) Handles btn_delete_dept.Click
        sql = "DELETE FROM `tbldepartment` WHERE `ID`='" &
dtgdeptlist.CurrentRow.Cells(0).Value & "'"
        deletes(sql, dtgdeptlist.CurrentRow.Cells(1).Value)
        Call frmSetting_Load(sender, e)
    End Sub

    Private Sub dtgdeptlist_CellClick(sender As Object, e As
DataGridViewCellEventArgs) Handles dtgdeptlist.CellClick
        Try
            departmentid = dtgdeptlist.CurrentRow.Cells(0).Value
            txtdepartment.Text =
dtgdeptlist.CurrentRow.Cells(1).Value

            btn_delete_dept.Enabled = True
            btn_update_dept.Enabled = True
            btn_save_dept.Enabled = False
        Catch ex As Exception
```

```
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub dtglistposition_CellClick(sender As Object, e As
DataGridViewCellEventArgs) Handles dtglistposition.CellClick
        Try
            positionID = dtglistposition.CurrentRow.Cells(0).Value
            txtposition.Text =
dtglistposition.CurrentRow.Cells(1).Value


            btn_Delete_Position.Enabled = True
            btn_update_Position.Enabled = True
            btn_save_Position.Enabled = False
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub
End Class
```

Manage Users Module:

```
Public Class frmUser
    Private Sub chkShowpass_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
chkShowpass.CheckedChanged
        Try
            If chkShowpass.CheckState = CheckState.Checked Then
                txtpass.UseSystemPasswordChar = False
            Else
                txtpass.UseSystemPasswordChar = True
            End If
        Catch ex As Exception

        End Try
    End Sub

    Private Sub frmUser_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        Try

            sql = "SELECT user_id as Id,name as Name,username as
Username, type as Type FROM  tbluser  "
            reloadDtg(sql, dtglist)

            cleartext(Me)
            loadautonumber("user", txtId)

            txtId.Enabled = False

        Catch ex As Exception
            Me.Text = ex.Message
        End Try
    End Sub

    Private Sub btnAdd_Click(sender As Object, e As EventArgs)
Handles btnAdd.Click
        Try

            For Each ctrl As Control In Me.Controls
                If ctrl.GetType Is GetType(TextBox) Then
                    If ctrl.Text = "" Then
                        MsgBox("One of the box is empty. It needed
to be filled up.", MsgBoxStyle.Exclamation)
                        Return
                    End If
```

```
                End If
                If ctrl.GetType Is GetType(ComboBox) Then
                    If ctrl.Text = "----Select-----" Then
                        MsgBox("You have to set the correct
information.", MsgBoxStyle.Exclamation)
                        Return
                    End If
                End If
            Next
            If txtpass.Text <> txtrpass.Text Then
                MsgBox("Password does not match.",
MsgBoxStyle.Exclamation)
                Return
            End If


            sql = "SELECT * FROM tbluser WHERE user_id='" &
txtId.Text & "'"
            reloadtxt(sql)
            If dt.Rows.Count > 0 Then
                sql = "UPDATE tbluser SET name='" & txtname.Text &
"' ,username='" & txtusername.Text & "',pass=sha1('" & txtpass.Text
& "'),type='" & cboType.Text & "'  WHERE user_id ='" & txtId.Text &
"'"
                updates(sql, txtname.Text)
            Else
                sql = "INSERT INTO tbluser
(user_id,name,username,pass,type)" &
             " VALUES (" & txtId.Text & ",'" & txtname.Text & "','"
& txtusername.Text & "',sha1('" & txtpass.Text & "'),'" &
cboType.Text & "')"
                create(sql, txtname.Text)
                updateautonumber("user")
            End If


            cleartext(Me)
            loadautonumber("user", txtId)
            sql = "SELECT user_id as Id,name as Name,username as
Username, type as Type FROM  tbluser  "
            reloadDtg(sql, dtglist)

        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub btnUpdate_Click(sender As Object, e As EventArgs)
Handles btnUpdate.Click
```

```
        Try

            txtId.Text = dtglist.CurrentRow.Cells(0).Value
            sql = "SELECT * FROM tbluser WHERE user_id ='" &
txtId.Text & "'"
            reloadtxt(sql)

            With dt.Rows(0)
                txtname.Text = .Item("name")
                txtusername.Text = .Item("username")
                'txtpass.Text = .Item("pass")
                'cboType.Text = .Item("type")
                cboType.Text = .Item("type")
            End With
        Catch ex As Exception
            Me.Text = ex.Message
        End Try
    End Sub

    Private Sub btnDelete_Click(sender As Object, e As EventArgs)
Handles btnDelete.Click
        Try
            sql = "DELETE FROM tbluser WHERE user_id ='" &
dtglist.CurrentRow.Cells(0).Value & "'"
            deletes(sql, dtglist.CurrentRow.Cells(2).Value)

            sql = "SELECT user_id as Id,name as Name,username as
Username, type as Type FROM  tbluser  "
            reloadDtg(sql, dtglist)

            cleartext(Me)
            loadautonumber("user", txtId)
        Catch ex As Exception
            Me.Text = ex.Message
        End Try
    End Sub
End Class
```

Testing

TESTING

SYSTEM TESTING

TESTING

Testing goes through the various stages, during testing the program to be tested has to be executed with a set of test cases, and ha the output of the program for the test case is evaluated to determine if the program is performing as expected. Due to its approach dynamic testing only ascertains the presence of error in the program. The exact nature of error is not usually decided by testing. Testing form is the first in determining error in the program.

Once the programs are tested individually then the system as a whole needs to be tested. During testing, the system is used experimentally to ensure that the software does not fail i.e. it will run according to its specification. The programs executed to check for any syntax or logical error. The error is corrected and test is made to determine whether the program is doing what it is supposed to do.

Various types of testing

Unit testing

Each component of the system is tested individually. The programmer does the testing. Testing is restrictive in nature i.e. programmer should

try to test all individual conditions and see if the program breaks under any circumstance.

System testing

This is an integrated form of testing, which focuses on functionality and interface between units and team in a controlled environment does it.

Acceptance Testing

This is system testing done by the user of the application the only emphasis is functionally testing as the user is not aware of the technical aspect of the system. The testing is also done in a controlled environment with logging o all error based on the error found in the system, the user has to accept or reject the system.

Module Testing

This is an optional form of testing, which is done only for large system, which has a large number of modules.

Security Testing

Security testing will be done as a specialized form of testing if there is a high risk exposure in that area. If the risk exposure is not very high, then it can be done as part of the system testing. Typically, security testing would involve trying to break in to the system, trying to execute transactions not allowed to person; to access areas on disk were the user is not allowed.

Testing is vital t the success of the system. If it on. This done successfully, this shows that the parts of the system are working correctly and all the goals are achieved.

IMPLEMENTATION

Implementation is used here to mean the process of converting a new or revised system design int operational one; conversion is one aspect of implementation. The other aspect is post implementation review and software and maintenance.

There are three types f implementation:

1. Implementation of a computer system
2. Implementation of a new computer system
3. Implementation of a modified application.

MAINTAINANCE

After the system has successfully been implemented maintenance activity may require continuous involvement of the developers. Provision must be made for environmental changes, which may affect either the computer, or other parts of the computer based system: such activity is normally called maintenance. It includes both the improvement of the system functions and the correction of faults that arise during the operation of a system.

Maintenance activity may require the continuing involvement of a large proportion of computer department resources. Maintenance works may arise due to two reasons:

1. Errors that creep up during normal running
2. Request for changes by the service providers. As part of the normal running of the system when errors are found.

This maintenance work will help to ensure that the system works smoothly as predicted in the open environment. Whenever a maintenance work arises, the work has to be properly carried out with proper documentation. This is

to avoid any form of changes in the structure of the system.

For every maintenance work an amendment notification is to be issued. This notification will have required changes and also authenticated. On the receipt of the amendment notification the amendment

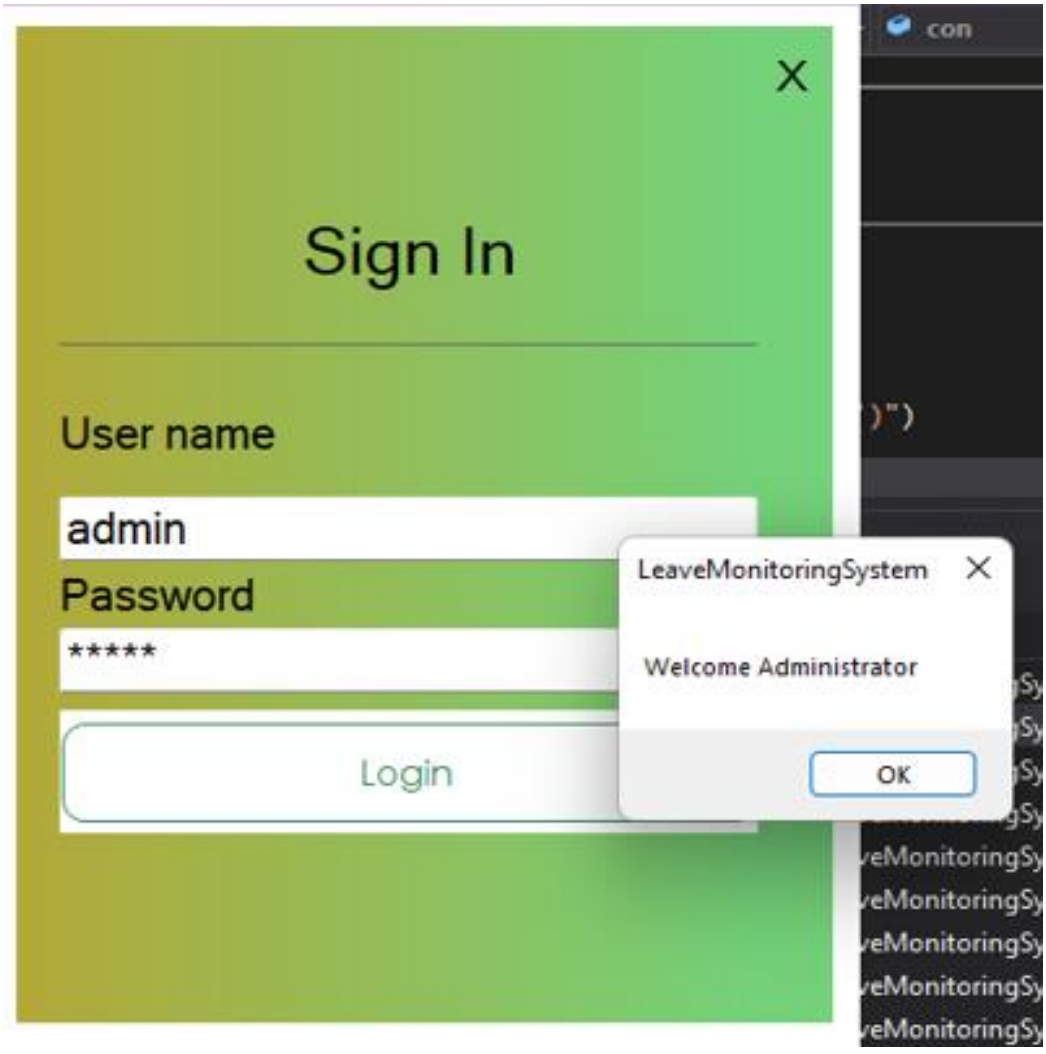
Log is prepared which records these courses of action that has been planned to be taken. It also records the estimated and the actual completion of each activity.

Sample Test Cases:

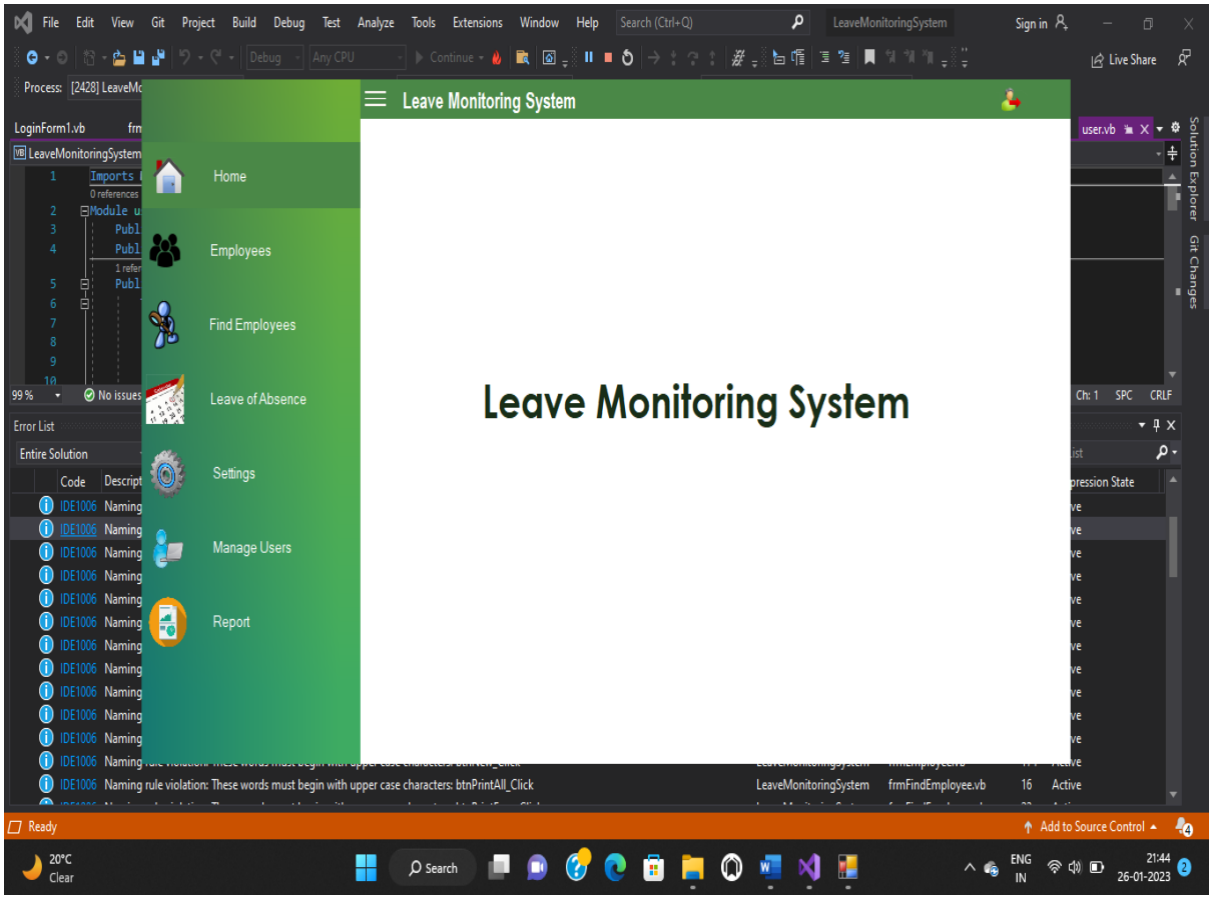
- Invalid Login credentials are tested and appropriate error messages are displayed.
- The data capture part is thoroughly tested for entry of correct data in acceptable type and size.

Appropriate interfaces are created to link various modules.

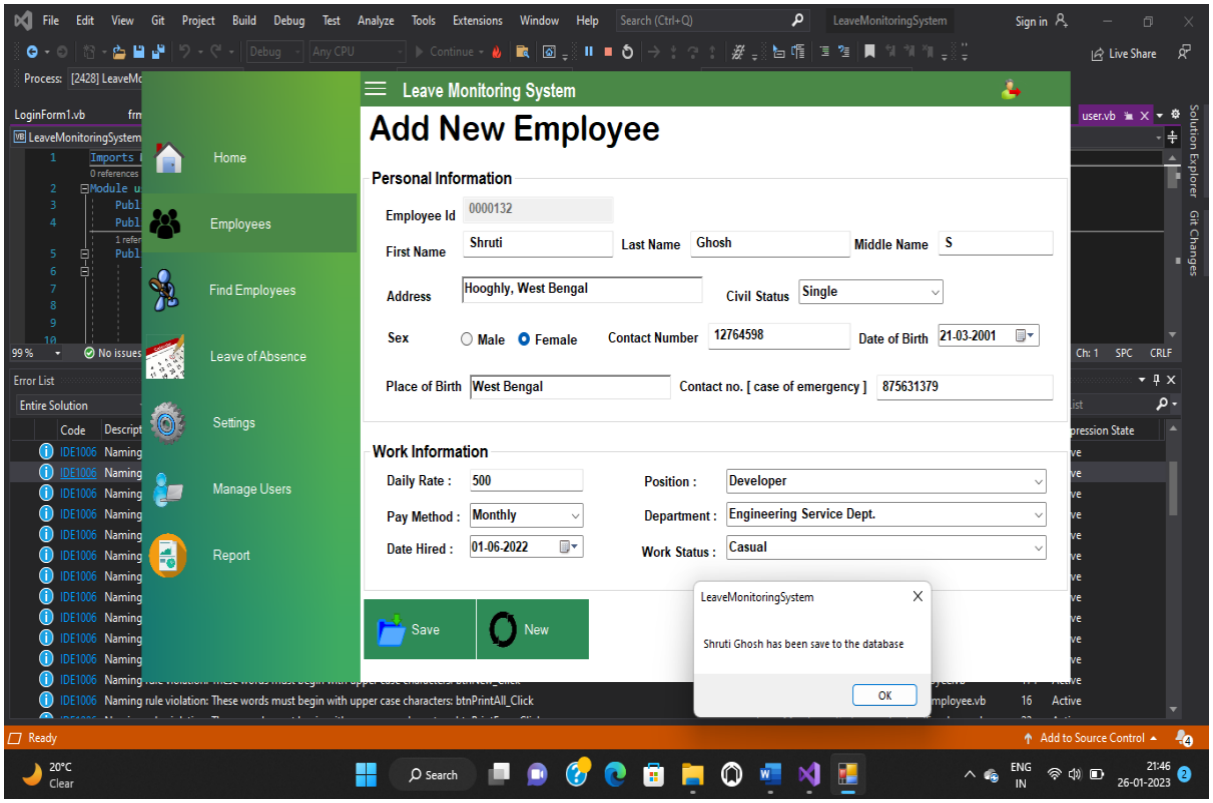
Snapshots


Sign In
User name
admin
Password
*****
Login
LeaveMonitoringSystem
Welcome Administrator
OK

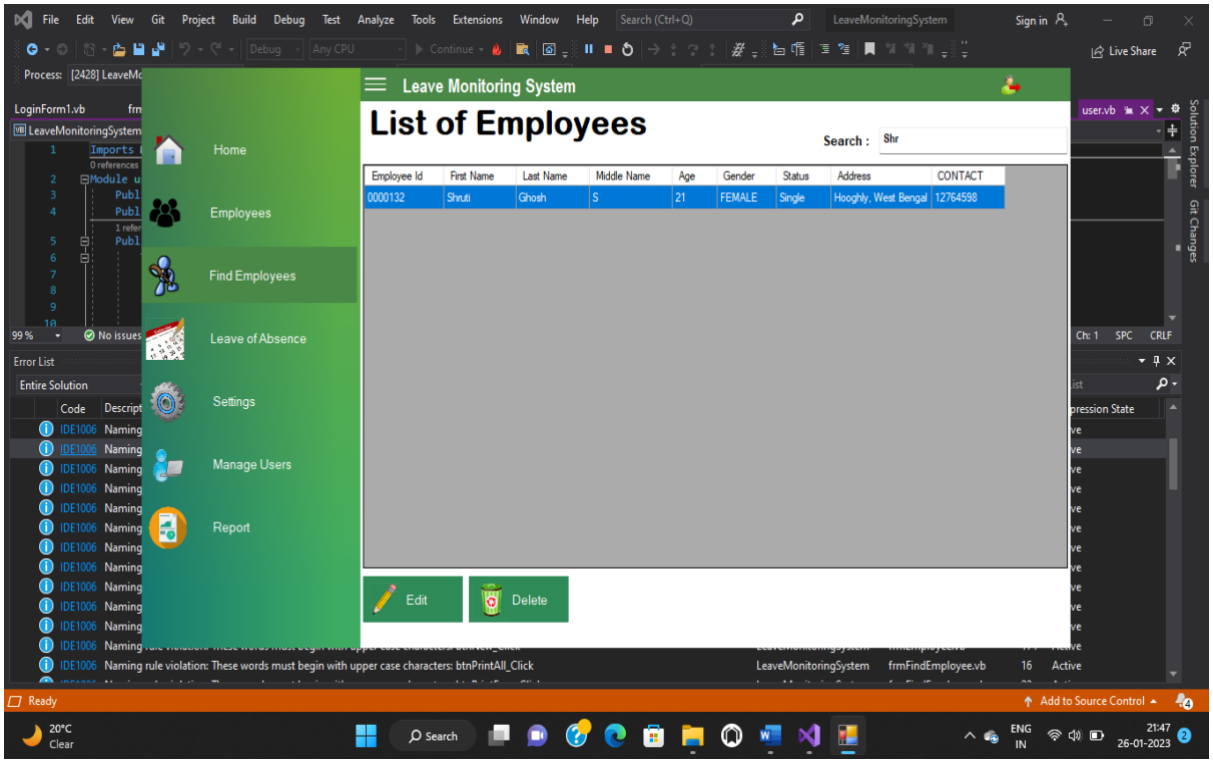
LEAVE MANAGEMENT SYSTEM



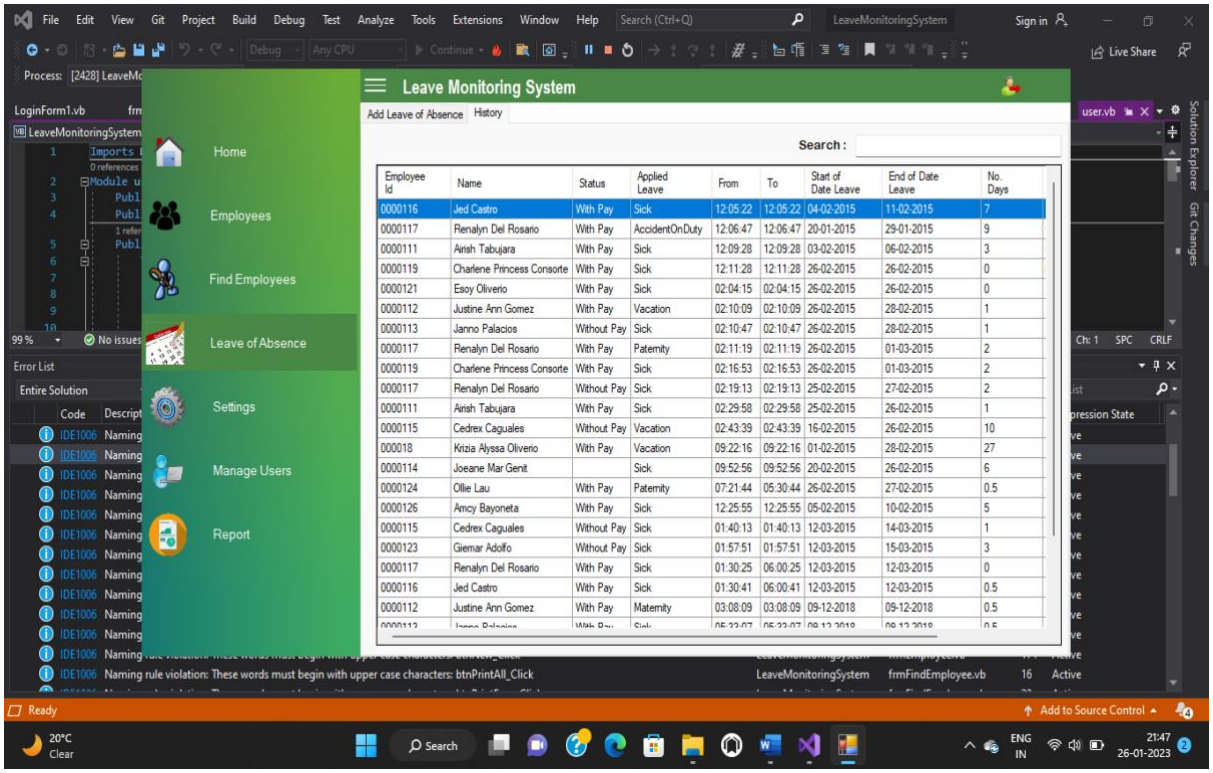
LEAVE MANAGEMENT SYSTEM



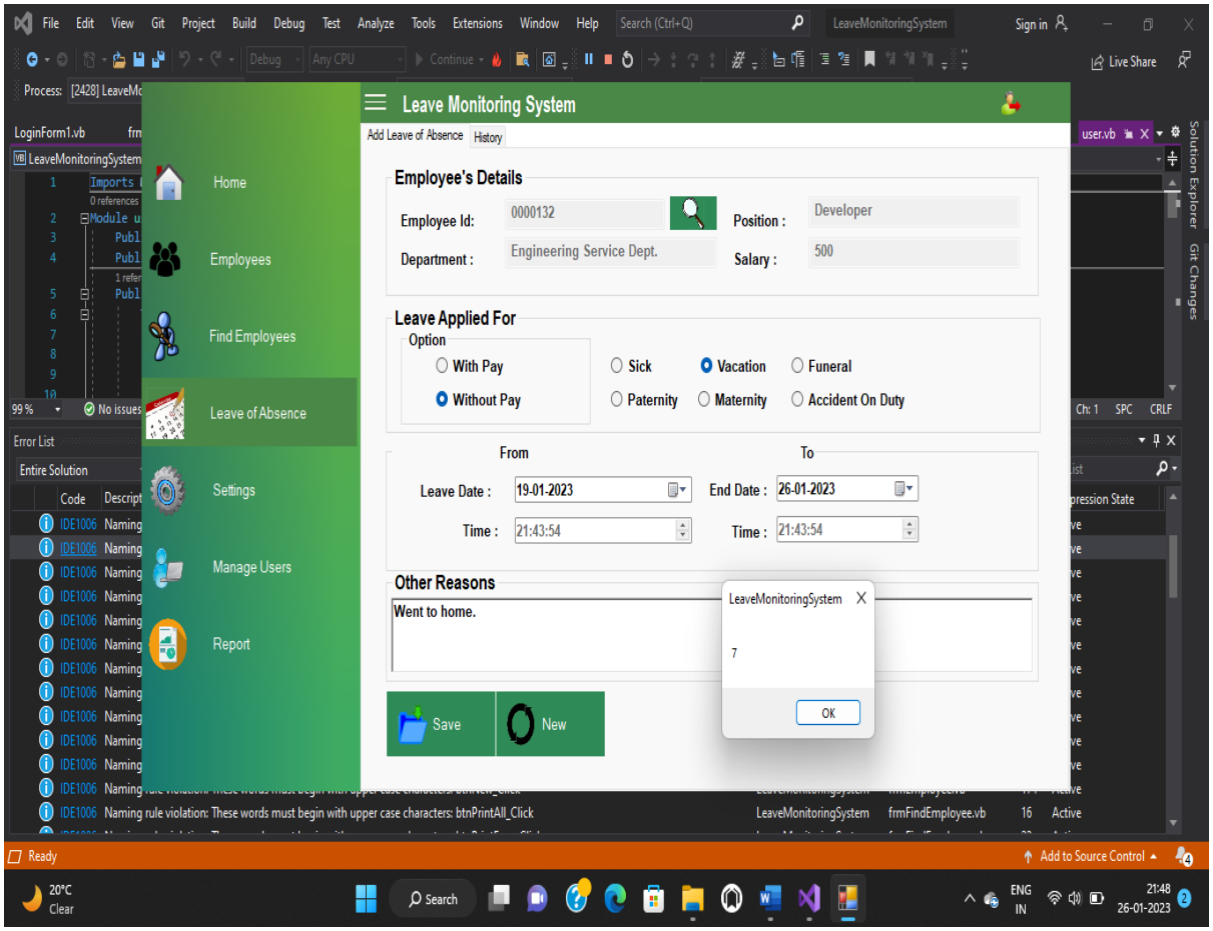
LEAVE MANAGEMENT SYSTEM



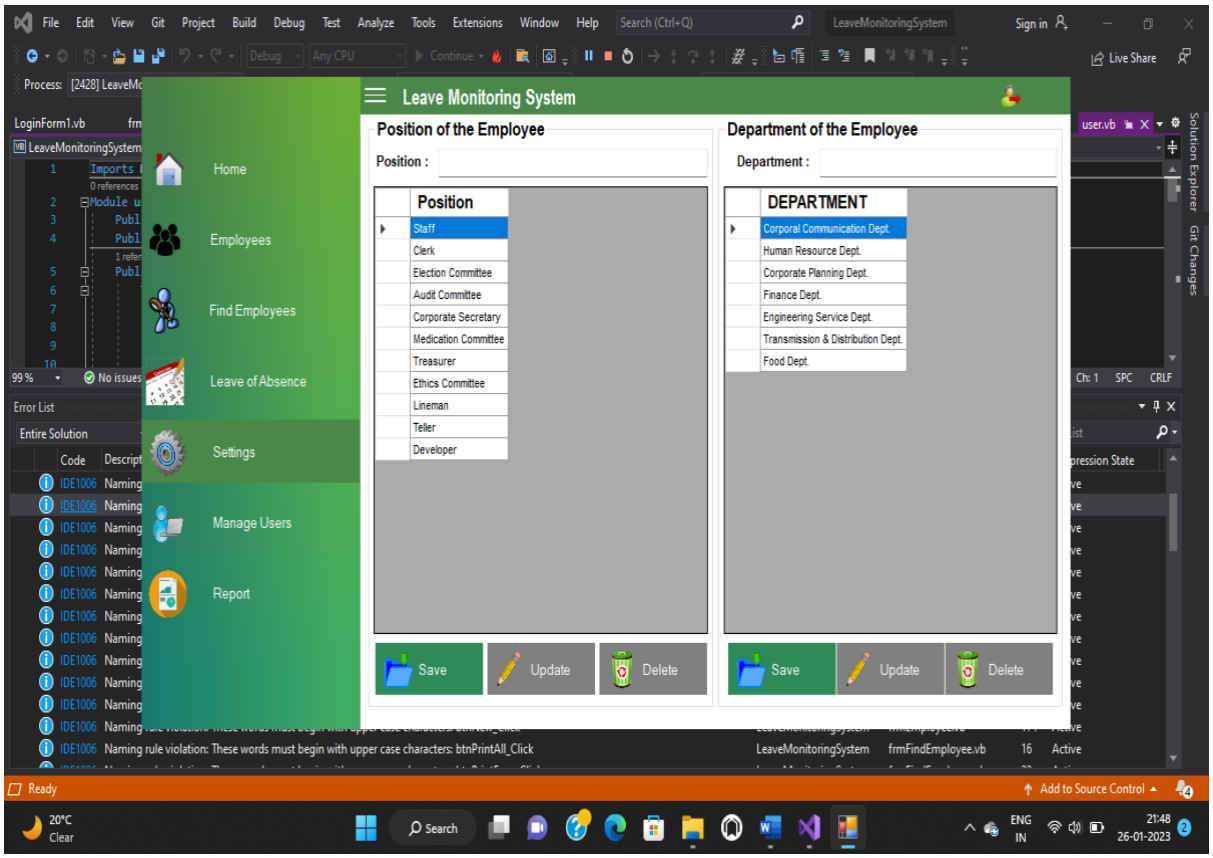
LEAVE MANAGEMENT SYSTEM



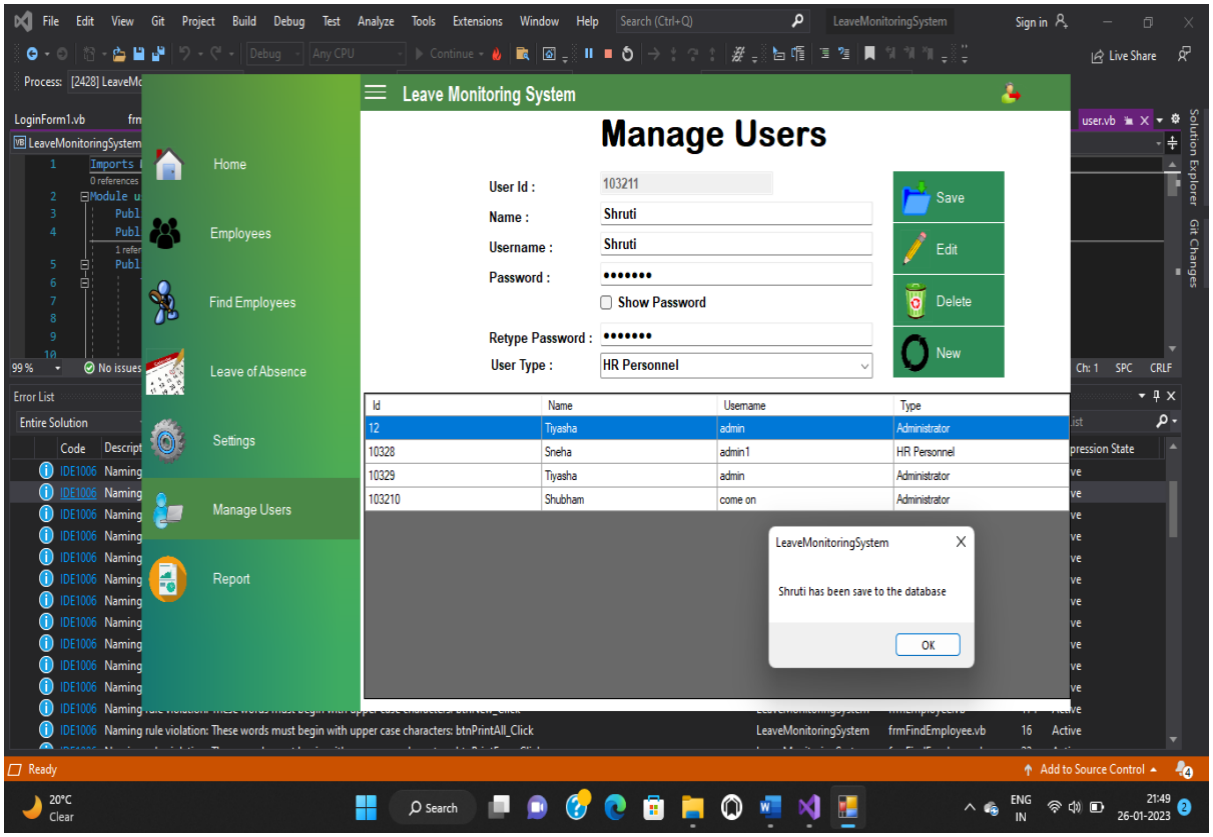
LEAVE MANAGEMENT SYSTEM



LEAVE MANAGEMENT SYSTEM



LEAVE MANAGEMENT SYSTEM



Future Enhancements

FUTURE ENHANCEMENTS

As per the organizations needs new modules can be integrated without any modification to the existing application and we can add other types of departments and positions of the organization's employees when new departments get started. We can also provide mobile facility to improve the satisfaction of customers.

MERITS

- → This project gives detailed overview of the organizations employee and the leaves taken by them.
- → Increase in efficiency of storage of employee details and their leaves within organization.
- → It improves accuracy of the work of operators.
- → It provides safety and security.
- → We have provided user-friendly interface to help the user to operate the system.
- → Data will be stored in the database easily without any errors.

Conclusion

CONCLUSION

This project helps to maintain the all the details of the employees and the leaves taken by them. We hope that this project can be the most useful in all organizations. In an organization the administrator and the HR can easily maintain the employee details. It is easy to keep track of the number of days leave taken by an employee and whether it is paid or not.

In our project titled “Leave Management System” using the Remote SQL Server as a backend and VB.Net 2019,Bunifu Framework as front end we have tried to provide many options like Login forms, View forms, Employee forms and Leave Form and History Form, which are helpful to the operators to give better service to the users.

Since the implementation of a database application by us was related to leave management, we could know how real world constraints have to be dealt with and how, any problems that arose could be solved.

Bibliography

BIBLIOGRAPHY

- Beginning SQL Server
  - Denise Gosnell
- .NET FRAME WORK
  - Matthew A. Stocker & Steven J. Stein
- Bunifu Framework
  - Bunifu Technologies Limited
- MSDN CDs
- Visual basic .NET bible
- http://www.tutorialspoint.com/vb.net/