Chapter 6

Implementation

6.1 Introduction

Last chapter discussed with Analysis and Design Stage includes database design and User Interface Design. This chapter implements the source code of the system, which last chapter designed. Only their important points are highlighted here.

6.2 Show Alarms

Show Alarms is the most important and attracting implementation with colorful data grid interface. There are five main functions running with this data grid once it opened.

1. Hide Not-Commissioned Sites
2. Maintain colour information according with Alarm Code.
3. Insert colour information with alarm duration
4. Insert Tool tip text with alarm history records
5. when site failure with power alarm insert failure reason with “suspect power failure and battery drain”

6.2.1 Hide Not-Commissioned Sites

This is a Do loop which runs with increasing row number, end condition is

```
Loop Until GridView.Item(Colnum,Rownum).RowIndex > GridView.RowCount- 2
```

Following code shows how to implement this feature on the .Grid view

```
Do
     SiteHistory = GridView.Item(Colnum, Rownum).Value
     HistoryNotCommissioned(SiteHistory)
     If ComStatus = True Then
          GridView.Rows(Rownum).Visible = False
     End If
     ComStatus = False
     Rownum = Rownum + 1
Loop Until GridView.Item(Colnum,Rownum).RowIndex > GridView.RowCount- 2
```

6.2.2 Maintain colour information according with Alarm Code

This code extracts colour information for its Alarm Code from class “AlarmCodeColour” , and mixed RGB colour values using “FromArgb” method and show on the Grid cell.

Following code shows how system receives background and font colours.

```
GridView.Item(Colnum + 3, Rownum).Style.BackColor = Color.FromArgb(Rclr,
Gclr, Bclr)
                GridView.Item(Colnum + 3, Rownum).Style.ForeColor =
Color.FromArgb(Rclrfont, Gclrfont, Bclrfont)
```

6.2.3 Insert colour information with alarm duration

```
Dtime = GridView.Item(Coldte, Rownum).Value
Durtime = Now - Dtime
Durday = Durtime.TotalDays
Durhour = Durtime.TotalHours
```

Above code shows how system calculates colour duration from current and alarm time difference. If difference <1h = green colour

Difference <2h = orange colour

Difference <3h = red colour

Else Difference >3days = no colour is the accepted colour format.

6.2.4 Insert Tool tip text with alarm history records

Here tool tip text adding and modification happen in three times each time the usable code is,

```
GridView.Item(Colnum + 3, Rownum).ToolTipText = AlmNme & " - " & ClrPrty
```

First: every record has Tool tip text with priority level.

Second: if site failure (AlarmCode=8196) Tool tip text replace as

```
= "B.Bank Duration -" & BBankDrn & "hours, Failure case:" & AlmHstry
```

Third: when site failure (8196) with power alarm (3862) for same Site Name, Tool tip text of alarm 3862 modify as,

```
= "B.Bank Duration -" & BBankDrn & "hours, Failure case:suspect Power
Failure and battery drain"
```

6.3 Maintain Alarm Codes

6.3.1 Using Stored procedures

This module functionality is alarm code maintenance with 3 main functions. [6]

1. Add
2. Modify
3. Delete

With implementation it is very important in design how to add colours to the databases. Finally it end at wonderful design with more flexible interface, I used horizontal Scroll bars and its “value change” event initiates each colour value to “Display” label and generate label colour. Each 3 Horizontal bar values stored into database as RGB colour values with integer data type. If ColourID already exists in database, the system accepts “Alarm” table and “Priority” table only. The stored procedure for this case is as follows,

```
ALTER PROCEDURE dbo.AlarmColour_InsertOnlyCode
(
    @AlarmCode      int,
    @AlarmName      varchar(50),
    @ColourID       varchar(50),
    @P_ID           int
    )
AS
INSERT INTO Alarm
   (AlarmCode,AlarmName, ColourID, P_ID)
   VALUES(@AlarmCode,@AlarmName, @ColourID, @P_ID)
UPDATE Alarm
SET ColourID=@ColourID
WHERE  AlarmCode=@AlarmCode
```

Figure 6.1 – Stored procedure for insert only AlarmCode not Colour details

If none of tables exist this new adding record in all 3 database tables.

Add 1. Colour table

2. Priority table

And finally Alarm table accepts the new record. The stored procedure for this case is as follows,

```
ALTER PROCEDURE dbo.AlarmColour_Insert
(
    @AlarmCode      int,
    @AlarmName      varchar(50),
    @ColourID       varchar(50),
    @P_ID           int,
    @RColour        int,
    @GColour        int,
    @BColour        int,
    @RColourfont    int,
    @GColourfont    int,
    @BColourfont    int
    )
AS
INSERT INTO Colour
   (ColourID, Rcolour, Gcolour, Bcolour, Rcolourfont, Gcolourfont, Bcolourfont)
   VALUES(@ColourID, @Rcolour, @Gcolour, @Bcolour, @Rcolourfont, @Gcolourfont, @Bcolourfont)
INSERT INTO Alarm
   (AlarmCode,AlarmName, ColourID, P_ID)
   VALUES(@AlarmCode,@AlarmName, @ColourID, @P_ID)
```

Figure 6.2 – Stored procedure for insert both AlarmCode and Colour details.

6.3.2 Integer validation

Alarm Code only accepts numeric values. Then it is important to define proper rules and messages to implement this area. Following code shows how to implement numeric check in Alarm monitoring system at deletion.

```
If IsNumeric(AlarmCode.Text) Then
Dim response As MsgBoxResult
response = MsgBox("Are you Sure, You want to Delete", MsgBoxStyle.YesNo, "Alarm Delete")
        If response = MsgBoxResult.Yes Then
        Succesful = CdColr.DeleteAlarmRecord(AlarmCode.Text)
        CdColr.CloseConnection()
                If Succesful = True Then
                        LabelMsg.Text = "Alarm Code '" & AlarmCode.Text & "' deleted
                successfully"
                Else
                        LabelMsg.Text = "'" & AlarmCode.Text & "' not existing, Alarm Code
                deletion failed"
                End If
                Cleartxt()
        Else
                LabelMsg.Text = "Alarm Code deletion aborted"
        End If
Else
        LabelMsg.Text = "Please Enter numeric value for the Alarm Code"
End If
```

6.3.3 Priority Level

The priority level select with combo box and priority already added to the drop down list of the combo. They are,

Minor, Critical and Major

Following code shows how to implement priority in VB.net .Priority ID (P_ID) data type is integer, values are 1, 2 and 3 only.

```
If Prty = "Minor" Then
    objCommand.Parameters.Add("@P_ID", SqlDbType.Int, 20).Value = 1
ElseIf Prty = "Critical" Then
    objCommand.Parameters.Add("@P_ID", SqlDbType.Int, 20).Value = 2
Else
    objCommand.Parameters.Add("@P_ID", SqlDbType.Int, 20).Value = 3
End If
```

6.3.4 Insert Record

Following code shows how to use above stored procedures in practical programming

1. Given method search whether “ColourID” exists in “Colour” table first,

```
objDataReader1 = selectColourID(objCommand, "ColourRecordIDQuery", ClrID)
If objDataReader1.HasRows Then
    CloseConnection()
```

2. If exists then search whether “AlarmCode” exists in “Alarm” table,

```
    OpenConnection()
    objDataReader1 = selectCode(objCommand, "ColourRecordQuery", AlmCde)
    If objDataReader1.HasRows Then
        MsgBox("Alarm Already exists")
        CloseConnection()
        Return False
```

3. If exists then give message box `("Alarm Already exists")` and exit.

```
    Else
        CloseConnection()
```

4. Else insert “AlarmCode” to “Alarm” table with existing “ColourID”.

```
        OpenConnection()
        objCommand = New SqlCommand("AlarmColour_InsertOnlyCode", objConnection)
        objCommand.CommandType = Data.CommandType.StoredProcedure

        intRowsAffected = objCommand.ExecuteNonQuery()
        Return True
    End If
Else
    CloseConnection()
```

5. If “ColourID” not exists then search whether “AlarmCode” exists in “Alarm” table,

```
    OpenConnection()
    objDataReader1 = selectCode(objCommand, "ColourRecordQuery", AlmCde)
    If objDataReader1.HasRows Then
        MsgBox("Alarm Already exists")
        CloseConnection()
        Return False
```

6. If exists then insert Colour records to “Colour” table, and modify “Alarm” table row with new “ColourID”

```
    Else
        CloseConnection()

        OpenConnection()
        objCommand = New SqlCommand("AlarmColour_Insert", objConnection)
        objCommand.CommandType = Data.CommandType.StoredProcedure
```

7. If Both “ColourID” and “AlarmCode” not exists then, insert both records.

```
        intRowsAffected = objCommand.ExecuteNonQuery()
        Return True
    End If
End If
```

6.4 Maintain Commissioned Site's Alarm History and Not-Commissioned Sites

6.4.1 Maintain two GUIs with same Table and column Status

Commissioned Site's Alarm History and Not-Commissioned Site both records are added to the same table with status different. Table 6.1 shows the priority level of both.

GUI form	Status
Commissioned Site's Alarm History	0
Not-Commissioned Site	1

Table 6.1 – Priority level bit

Data type of the Status is "bit" values 1 or 0.

When inserting or updating the database this value is given as "True" or "False"

Following code shows how to implement above function.

```
If txtSiteName.Text <> "" Then
  Succesful = dbMgr.AddCommissionedRecord(txtSiteName.Text, Double.Parse(0), " ", "True")
  dbMgr.CloseConnection()
  If Succesful = True Then
     LabelMsg.Text = "Site Name added successfully"
  Else
     LabelMsg.Text = "Site Name addition Failed"
  End If
  Cleartxt()
Else
  LabelMsg.Text = "Sorry! Required fields cannot be empty"
End If
```


Above implementation is for Not-Commissioned Site.

When it received to "ClassDBManager" class it stores above record to the database by using following code,

```
objCommand.Parameters.Add("@Status", SqlDbType.Bit, 5).Value = CType(Sts, Boolean)
```

6.4.2 Add existing Site Name

System can identify how given Site name status and what are the messages should be given, as example if given "Site Name" status is "Not-Commissioned" or "Commissioned" and give error message according to the status. Following code shows how to implement such situation.

```
        OpenConnection()
            objDataReader1 = selectSiteName(objCommand, "AlarmDetail_SelectAllforAdd",
SiteName)
            If objDataReader1.HasRows Then
                objDataReader1.Read()
                ComStatus = objDataReader1.Item("Status")
                If ComStatus = "True" Then
                    MsgBox("Site already exists in Not-Commissioned Table")
                Else
                    MsgBox("Site already exists in Commissioned Table")
                End If
                CloseConnection()
                Return False
            Els
```

Figure 6.3 – Error message

Figure 6.3 shows how error message given when Not-Commissioned site exists.

Additionally Commissioned and Not-Commissioned GUI come with extra button which use to change the status of Site from Commissioned to Notcommissioned or Notcommissioned to Commissioned.

6.4.3 Delete not existing Site Name

When you are in Commissioned form delete button do not support on deletion of Not-Commissioned Site Details. Following code shows how to implement this feature.

```
            OpenConnection()
           objDataReader1 = selectSiteName(objCommand, "AlarmDetail_SelectAllforAdd",
SiteName)
           If objDataReader1.HasRows Then
               objDataReader1.Read()
               ComSts = objDataReader1.Item("Status")
               CloseConnection()
               If ComSts = Sts Then
                   OpenConnection()
                   objCommand = New SqlCommand("AlarmDetail_Delete", objConnection)
               Else
                   Return False
               End If
           Else
              Return False
           End If
```


6.5 Search Alarms

Search option of the input box support for both Text and integer values, when text input system count as Search Site name.

It input value is integer system search “Alarm Code” on same input box (Figure 6.4)

Figure 6.4 – Search

Following code shows how to implement this feature,

```
If openConnection() Then
            If IsNumeric(Name) Then
                objCommand.CommandText = "AlarmRecord_AlarmCodeQuery"
                objCommand.CommandType = CommandType.StoredProcedure
                objCommand.Connection = objConnection
                objCommand.Parameters.Add("@AlarmCode",
SqlDbType.VarChar, 20).Value = Name
            Else
                objCommand.CommandText = "AlarmRecord_SiteNameQuery"
                objCommand.CommandType = CommandType.StoredProcedure
                objCommand.Connection = objConnection
```

```
                objCommand.Parameters.Add("@SiteName",
SqlDbType.VarChar, 20).Value = Name
            End If
```

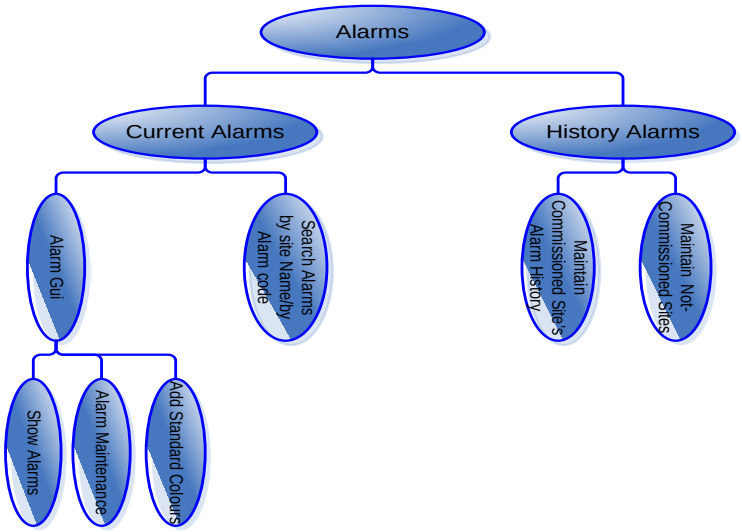

Figure 6.5 – Coding maintenance guide

6.6 Coding maintenance guide

Sub system of architectural design	Class
Show Alarms	`CodeColourTooltipDurationColour()`
Alarm Maintenance - search	`SearchAlarmRecord()`
Alarm Maintenance - Add	`InputAlarmCodeandColour()`
Alarm Maintenance – Modify	`ModifyAlarmCodeandColour()`
Alarm Maintenance – Delete	`DeleteAlarmCodeandColour()`
Standard Colours	`ModifyAlarmCodeandColour()`
Search Alarms by Site name/ by Alarm Code	`PopulateList()`
Commissioned site's Alarm History Maintenance- search	`SearchRecord()`
Commissioned site's Alarm History Maintenance- Add	`InputSiteNameandDetails()`
Commissioned site's Alarm History Maintenance– Modify	`ModifySiteNameandDetails()`
Commissioned site's Alarm History Maintenance– Delete	`DeleteSiteNameandDetails()`
Commissioned site's Alarm History Maintenance– Ntcom.	`SetNotCommissioned()`
Not- Commissioned Sites Maintenance- search	`SearchRecord()`
Not- Commissioned Sites Maintenance- Add	`InputSiteName()`
Not- Commissioned Sites Maintenance– Modify	`ModifySiteName()`
Not- Commissioned Sites Maintenance– Delete	`DeleteSiteName()`
Not- Commissioned Sites Maintenance– Commissioned	`SetCommissioned()`

Table 6.2 – Coding maintenance guide

6.7 Summary

This chapter discussed with implementation in each module with their important code snippets. Next chapter is for Evaluation & Testing all the codes using black box test. This test based on Activity diagram input and output interfaces. Test cases and test results are bundled to tables, gives clear picture on implementation chapter.