Project-CIS232\_TL

[**Project 2A: Maintenance Products**](#_heading=h.30j0zll) **3**

[**The main form and the Product Maintenance form**](#_heading=h.6g23qseiem8s) **3**

[**Project items**](#_heading=h.1fob9te) **3**

[**Operation of the main form**](#_heading=h.3znysh7) **4**

[**The main form items**](#_heading=h.2et92p0) **4**

[**Operation of the Product Maintenance form**](#_heading=h.tyjcwt) **4**

[**Code**](#_heading=h.nzhe5xjyazyv) **4**

[**Project 2B: Customer Maintenance**](#_heading=h.w77eulfsg06i) **6**

[**The Customer Maintenance form**](#_heading=h.sy4d7fuzv2ll) **6**

[**Project items**](#_heading=h.hytlecbxo4zx) **6**

[**Operation of the Customer Maintenance form**](#_heading=h.9vabyvy1j2fw) **6**

[**Code**](#_heading=h.onpkym2818nm) **7**

[**Project 2C: Incidents by Customers**](#_heading=h.66lccek6g8ys) **10**

[**The Incidents by Product form**](#_heading=h.29d4h9qg9w6s) **10**

[**The Customer form**](#_heading=h.j78a2lqwgn46) **10**

[**Project items**](#_heading=h.iuatqb2s4h04) **10**

[**Operation of the Incidents by Product form**](#_heading=h.ampp9bgvk0o) **11**

[**Operation of the Customer form**](#_heading=h.1ob3tzk3lzgg) **11**

[**Code (frmProductIncidents)**](#_heading=h.sfiuo66vfu91) **11**

[**Code (frmCustomer)**](#_heading=h.meqvuoy2yftr) **12**

[**Project 3A: Open Incidents**](#_heading=h.wu8wtjqfffvx) **13**

[**The Open Incident form**](#_heading=h.n961s3bk0hz3) **13**

[**SportPro project item**](#_heading=h.pnuzu37bn4ur) **13**

[**TechSupportData project items**](#_heading=h.z84opr85w004) **13**

[**Operation of the Open Incidents form**](#_heading=h.pmy9eeocyw65) **14**

[**Code (frmOpenIncidents)**](#_heading=h.hiv9fr8p2bgy) **14**

[**Code (Incident)**](#_heading=h.vs770g2jaby7) **15**

[**Code (TechSupportDB)**](#_heading=h.jpbbkvqhd39b) **17**

[**Code (IncidentDB)**](#_heading=h.x7p6s8xb04q1) **17**

[**Code (CustomerDB)**](#_heading=h.x05rgs13axi9) **18**

[**Code (TechnicanDB)**](#_heading=h.a6dfoe3syrs3) **19**

[**Project 3B: Create an incident**](#_heading=h.hy02711e6kvc) **20**

[**The Create Incident form**](#_heading=h.9lvkm36d4kwm) **20**

[**SportsPro project items**](#_heading=h.uhd0vrxi9rds) **20**

[**TechSupportData project items**](#_heading=h.w250axv64isi) **20**

[**Operation**](#_heading=h.j1viq2fh9l1t) **21**

[**Code (frmCreateIncident)**](#_heading=h.8nd2osalrvm1) **21**

[**Code (Validator)**](#_heading=h.kdc1r8ja9xfl) **23**

[**Code (CustomerDB // GetCustomerList Function)**](#_heading=h.w619siqz06qr) **23**

[**Code (ProductDB)**](#_heading=h.gwi0d4jhvp48) **24**

[**Code (IncidentDB // AddIncident Sub)**](#_heading=h.bkgjaar95vv) **25**

[**Code (RegistrationDB)**](#_heading=h.upli8fvht69h) **25**

[**Project 3C: Update an incident**](#_heading=h.tl1w5p6zfuz) **27**

[**The Update Incident form**](#_heading=h.ik6zlhspp0d2) **27**

[**SportsPro Project items**](#_heading=h.85bex3dgqaew) **27**

[**TechSupportData items**](#_heading=h.dlwi6n9ry2uj) **27**

[**Operation**](#_heading=h.e3qy5zxqwxhb) **28**

[**Code (frmUpdateIncident)**](#_heading=h.wvvg4af9ngq8) **29**

[**Code (Validator // IsInt32)**](#_heading=h.ftueh3i8ju3z) **31**

[**Code (Incident // ProductName)**](#_heading=h.mvvalfurb9zf) **31**

[**Code (IncidentDB // GetIncident, UpdateIncident, CloseIncident)**](#_heading=h.mfa8ku5zaia6) **31**

[**Code (ProductDB // GetProductName)**](#_heading=h.52g6bqd8kr1e) **34**

[**Project 3D: Display open incidents by technician**](#_heading=h.drg0tz292c8) **35**

[**The Open Incidents by Technicians form**](#_heading=h.npfm25s2cx8q) **35**

[**SportsPro project item**](#_heading=h.x9ypqw44qmk9) **35**

[**TechSupportData project items**](#_heading=h.k49xysinjxxk) **35**

[**Operation**](#_heading=h.o0m60dbq9kvx) **36**

[**Code (frmTechnicianIncident)**](#_heading=h.6kwayph706tf) **36**

[**Code (Technician)**](#_heading=h.sa0hjcshruv2) **37**

[**Code (TechnicianDB // GetTechnicianList, GetTechnician)**](#_heading=h.kxy5zbrmewwo) **38**

[**Code (IncidentDB // GetOpenTechnicianIncidents)**](#_heading=h.x5f9woshui1) **40**

[**Project 3E: Maintain product registrations**](#_heading=h.2n43ptpymx2s) **41**

[**The Maintain Product Registrations form**](#_heading=h.b3r0peaysbhi) **41**

[**SportsPro project items**](#_heading=h.v3ti1z9bkk38) **41**

[**TechSupportData project items**](#_heading=h.fbrkwlavq7nf) **41**

[**Operation**](#_heading=h.axevxy2gtzk8) **42**

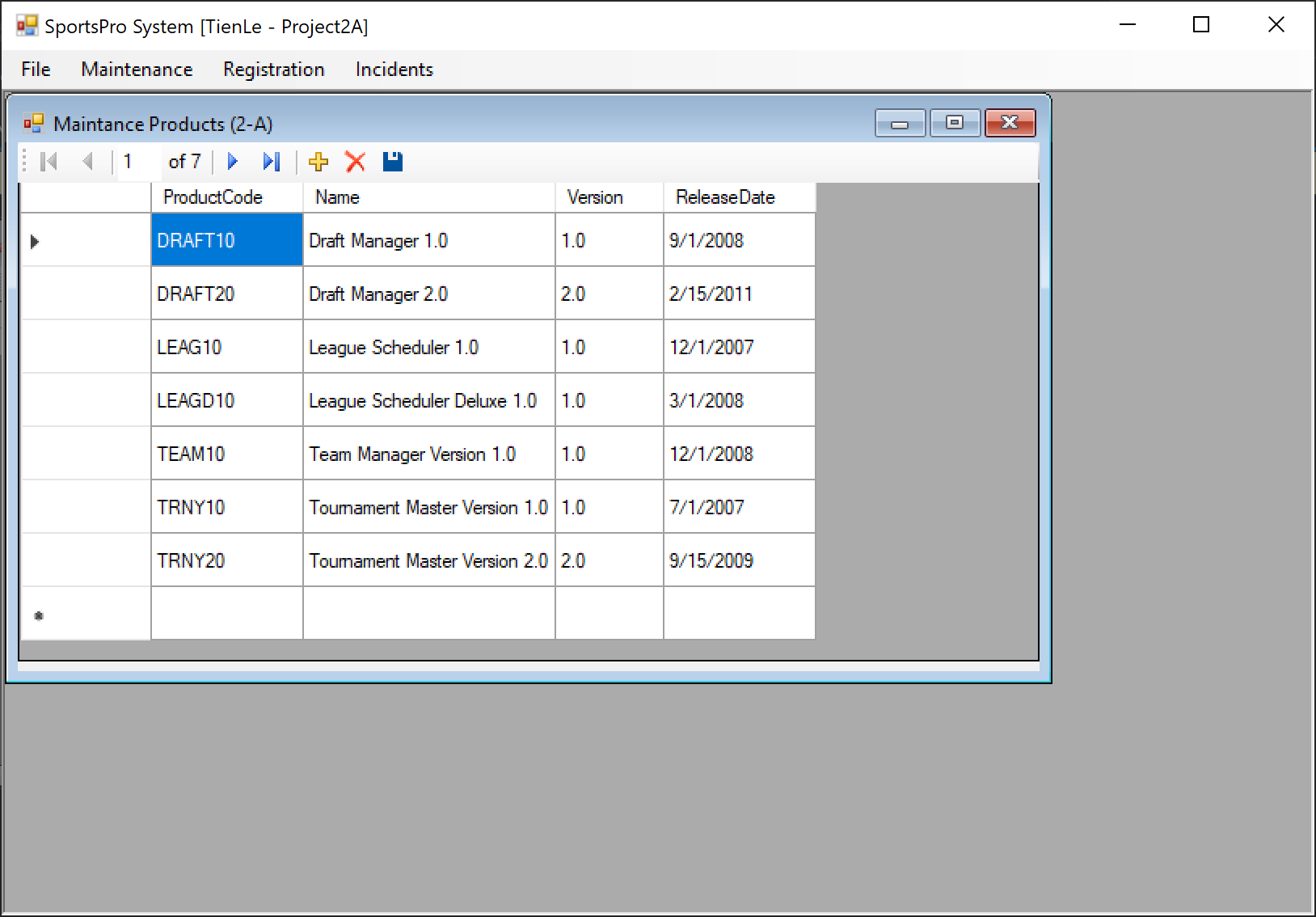
[**Code (frmMaintainRegistrations)**](#_heading=h.p7wo5s4xhqwx) **43**

[**Code (Registration)**](#_heading=h.wffdwn3upl4x) **44**

[**Code (RegistrationDB // AddRegistration)**](#_heading=h.m3nm0lcc3had) **45**

# Project 2A: Maintenance Products

## The main form and the Product Maintenance form



## Project items

|  |  |
| --- | --- |
| Name | Description |
| frmMain | The main form for the SportsPro application. The IsMdiContainer property for this form is set to True so it can contain MDI child forms. To provide access to the child forms, the main form includes a menu |
| frmProductMaintenance | A child form that lets the user add, update, and delete rows in the Products table. |
| TechSupportDataSet2A | A typed dataset with a single table named Products. |

## Operation of the main form

* The main form is the MDI parent form for the SportsPro application. The user can choose commands from this form’s menu to display the child forms of the application.
* To exit the application, the user can choose the File → Exit command or click the main form’s Close button

## The main form items

* The main form includes a status bar that displays my name: Tien Le
* The menu control for the main form should include the following menus

|  |  |
| --- | --- |
| Name | Description |
| File | Export Incidents (5-B)  Exit |
| Maintenance | Maintain Products (2-A)  Maintain Customers (2-B)  Maintain Registrations (3-E)  Maintain Technicians (5-G) |
| Registration | Display Customers by Product (5-F) |
| Incidents | Create Incident (3-B)  Update Incident (3-C)  Display Open Incidents (3-A)  Display Open Incidents by Technician (3-D)  Display Incidents by Product (2-C)  Create Mailing List (5-C)  Display Incidents by Product and Technician (5-D)  Display Incidents by Customer (5-E) |

## Operation of the Product Maintenance form

* The Product Maintenance form should be displayed when the user chooses the Maintenance → Maintain Products command from the menu on the main form.
* The user can use the navigator control and the DataGridView control to display, add, modify, or delete rows in the Products table.
* To close the Product Maintenance form, the user can click the form’s Close button

## Code

Imports System.Data.OleDb

Public Class frmProductMaintenance

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

Me.Validate()

'Me.ProductsBindingSource.EndEdit()

'Me.TableAdapterManager.UpdateAll(Me.TechSupportDataSet2A)

Try

Me.ProductsBindingSource.EndEdit()

Me.TableAdapterManager.UpdateAll(Me.TechSupportDataSet2A)

Catch ex As DBConcurrencyException

MessageBox.Show("A concurency error occurred. " &

"One or more rows were not updated or deleted.", "Concurrency Exception")

Me.ProductsTableAdapter.Fill(Me.TechSupportDataSet2A.Products)

Catch ex As DataException

MessageBox.Show(ex.Message, ex.GetType.ToString)

ProductsBindingSource.CancelEdit()

Catch ex As OleDbException

'ex.Message & vbCrLf & "You are being re-directed back to the Main Window", ex.GetType.ToString

MessageBox.Show("OLEDB Error: " & ex.Message & vbCrLf, ex.GetType.ToString)

End Try

End Sub

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

'TODO: This line of code loads data into the 'TechSupportDataSet2A.Products' table. You can move, or remove it, as needed.

Me.ProductsTableAdapter.Fill(Me.TechSupportDataSet2A.Products)

End Sub

Private Sub ProductsDataGridView\_DataError(ByVal sender As Object, ByVal e As System.Windows.Forms.DataGridViewDataErrorEventArgs) Handles ProductsDataGridView.DataError

Dim row As Integer = e.RowIndex + 1

Dim errorMessage As String = "A data error has occurred. " & vbCrLf &

"Row" & row & vbCrLf & "Error: " & e.Exception.Message

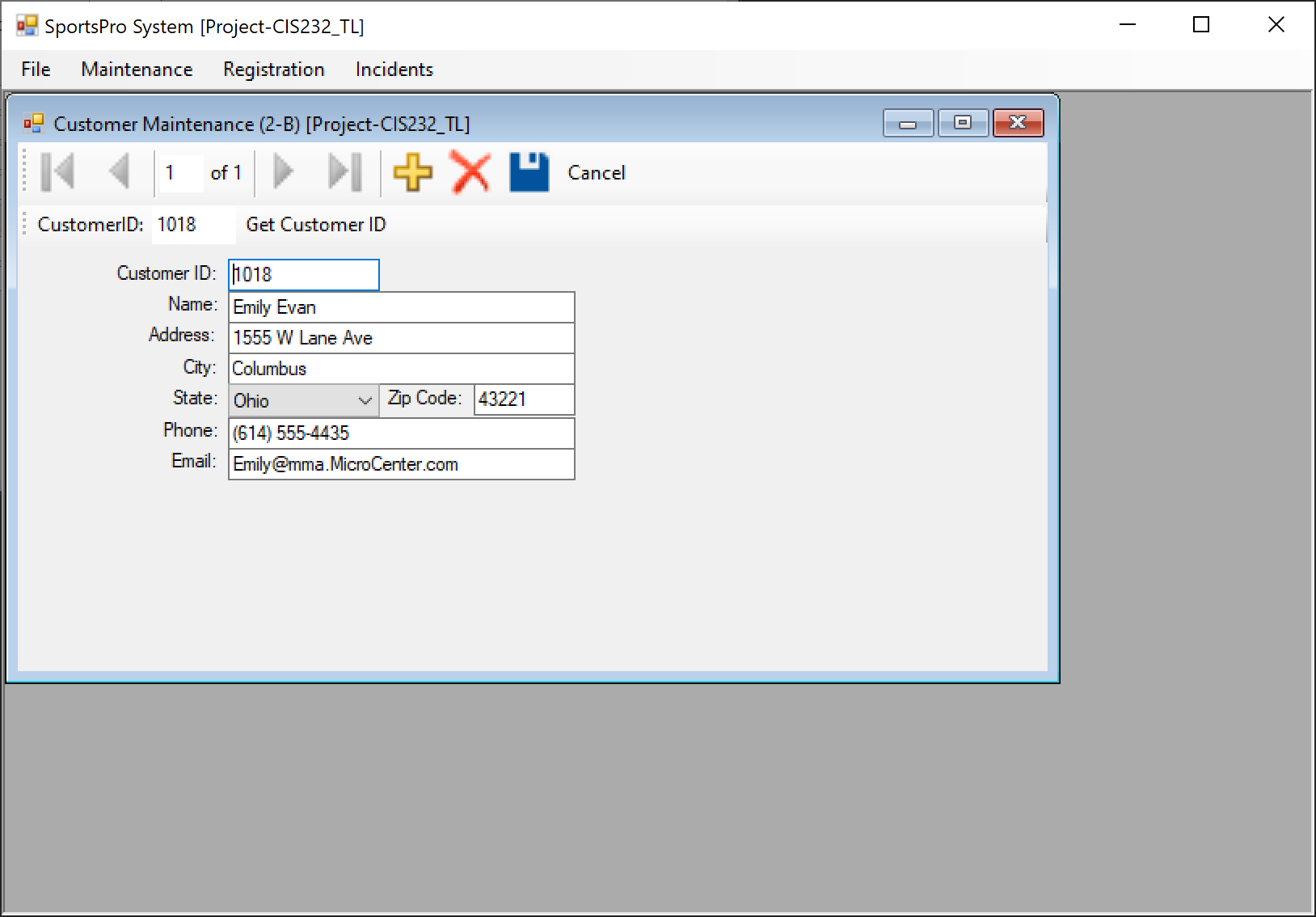
MessageBox.Show(errorMessage, "Data Error")

End Sub

End Class

# Project 2B: Customer Maintenance

## The Customer Maintenance form



## Project items

|  |  |
| --- | --- |
| Name | Description |
| frmCustomerMaintenance | A form that lets the user add, update, or delete customer rows. |
| Main | A module that contains a procedure for formatting a zip code. |
| TechSupportDataSet2B | A typed dataset with two tables named Customers and States |

## Operation of the Customer Maintenance form

* The Customer Maintenance form should be displayed when the user chooses the Maintenance → Maintain Customers command from the menu on the main form.
* The user can use the binding navigator control and the text boxes and combo box to display, add, modify, and delete a customer. The user can also enter a customer ID in the second toolbar and then click the Get Customer button to display the data for a customer.

## Code

Public Class frmCustomerMaintenance

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

'Me.Validate()

'Me.CustomersBindingSource.EndEdit()

'Me.TableAdapterManager.UpdateAll(Me.TechSupportDataSet2B)

'save changes

If IsValidData() Then

Try

Me.Validate()

Me.CustomersBindingSource.EndEdit()

Me.TableAdapterManager.UpdateAll(Me.TechSupportDataSet2B)

Catch er As FormatException

MessageBox.Show(er.Message, er.GetType.ToString)

Me.CustomersBindingSource.CancelEdit()

Catch er As OleDb.OleDbException

MessageBox.Show(er.Message, er.GetType.ToString)

Me.CustomersBindingSource.CancelEdit()

Catch er As Exception

MessageBox.Show(er.Message, er.GetType.ToString)

Me.CustomersBindingSource.CancelEdit()

End Try

End If

End Sub

Private Function IsValidData() As Boolean

If CustomersBindingSource.Count > 0 Then

Return IsPresent(CustomerIDTextBox, "CustomerID") AndAlso

IsPresent(NameTextBox, "Name") AndAlso

IsPresent(AddressTextBox, "Address") AndAlso

IsPresent(CityTextBox, "City") AndAlso

IsPresent(StateComboBox, "State") AndAlso

IsPresent(ZipCodeTextBox, "Zip Code") AndAlso

IsPresent(PhoneTextBox, "Phone Number") AndAlso

IsPresent(EmailTextBox, "Email Address")

Else

Return True

End If

End Function

Private Function IsPresent(ByVal control As Control,

ByVal name As String) As Boolean

If control.GetType.ToString = "System.Windows.Forms.TextBox" Then

Dim textBox As TextBox = CType(control, TextBox)

If textBox.Text = "" Then

MessageBox.Show(name & " is a required field.", "Entry Error")

textBox.Select()

Return False

Else

Return True

End If

ElseIf control.GetType.ToString = "System.Windows.Forms.ComboBox" Then

Dim comboBox As ComboBox = CType(control, ComboBox)

If comboBox.SelectedIndex = -1 Then

MessageBox.Show(name & " is a required field.", "Entry Error")

comboBox.Select()

Return False

Else

Return True

End If

End If

Return False

End Function

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

Dim b As Binding = ZipCodeTextBox.DataBindings("Text")

AddHandler b.Format, AddressOf frmMain.FormatZipCode

AddHandler b.Parse, AddressOf frmMain.UnformatZipCode

'TODO: This line of code loads data into the 'TechSupportDataSet2B.States' table. You can move, or remove it, as needed.

Me.StatesTableAdapter.Fill(Me.TechSupportDataSet2B.States)

'TODO: This line of code loads data into the 'TechSupportDataSet2B.Customers' table. You can move, or remove it, as needed.

Me.CustomersTableAdapter.Fill(Me.TechSupportDataSet2B.Customers)

End Sub

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

'parameterized query to search by CustomerID

Try

'The following 2 lines dont work

'Dim CustomerID As Integer = Convert.ToInt32(CustomerIDTextBox.Text)

'Me.CustomersTableAdapter.FillBy(Me.TechSupportDataSet2A.Customers, CustomerID)

'The following line is the default one generated by Visual Studio

Me.CustomersTableAdapter.FillByCustomerID(Me.TechSupportDataSet2B.Customers, CType(CustomerIDToolStripTextBox.Text, Integer))

Catch er As FormatException

MessageBox.Show("Customer ID must be an integer", "Entry Error")

CustomerIDToolStripTextBox.Text = ""

Catch er As System.Exception

MessageBox.Show(er.Message, er.GetType.ToString)

CustomerIDToolStripTextBox.Text = ""

End Try

End Sub

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

'Cancel button

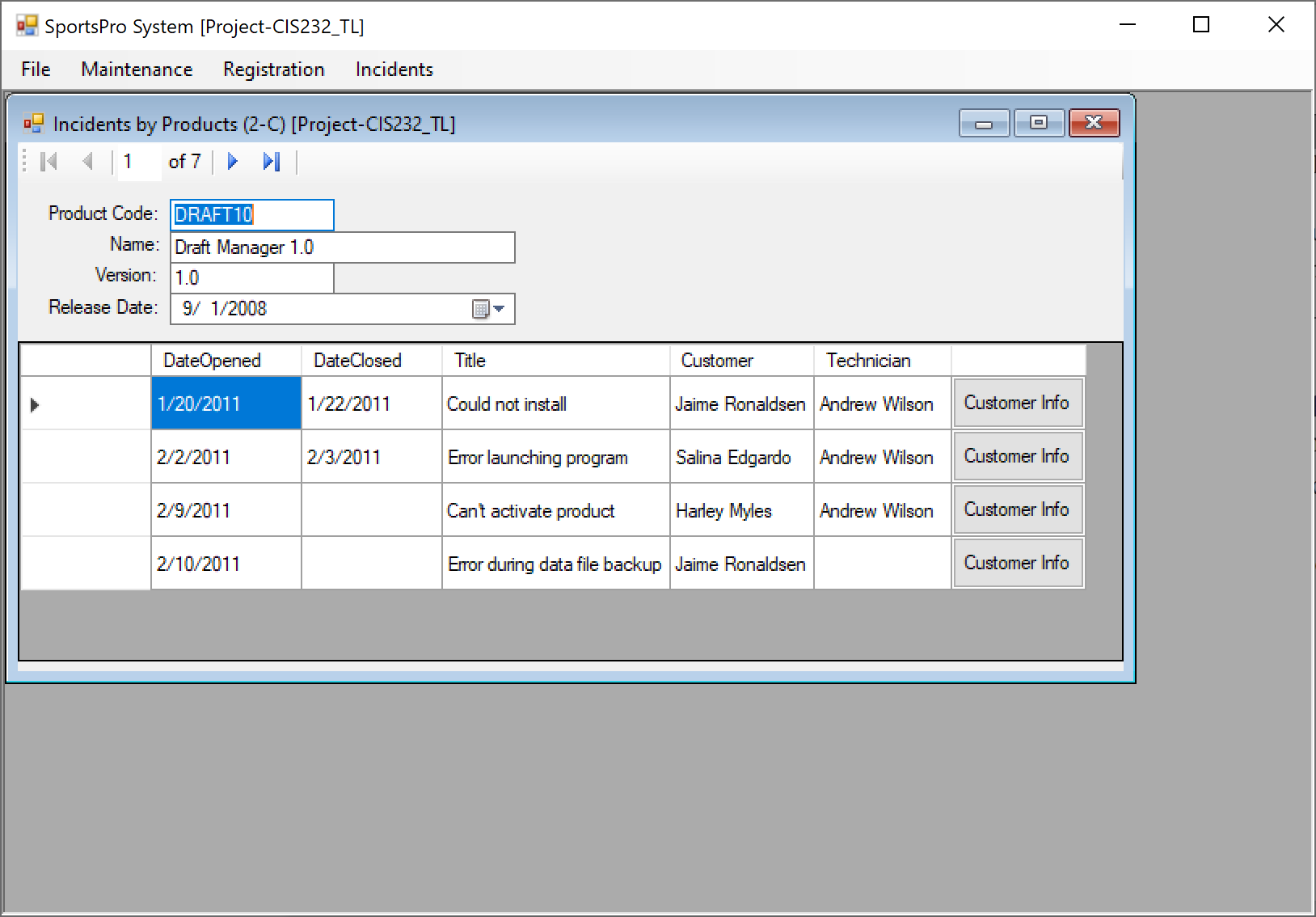
CustomersBindingSource.CancelEdit()

End Sub

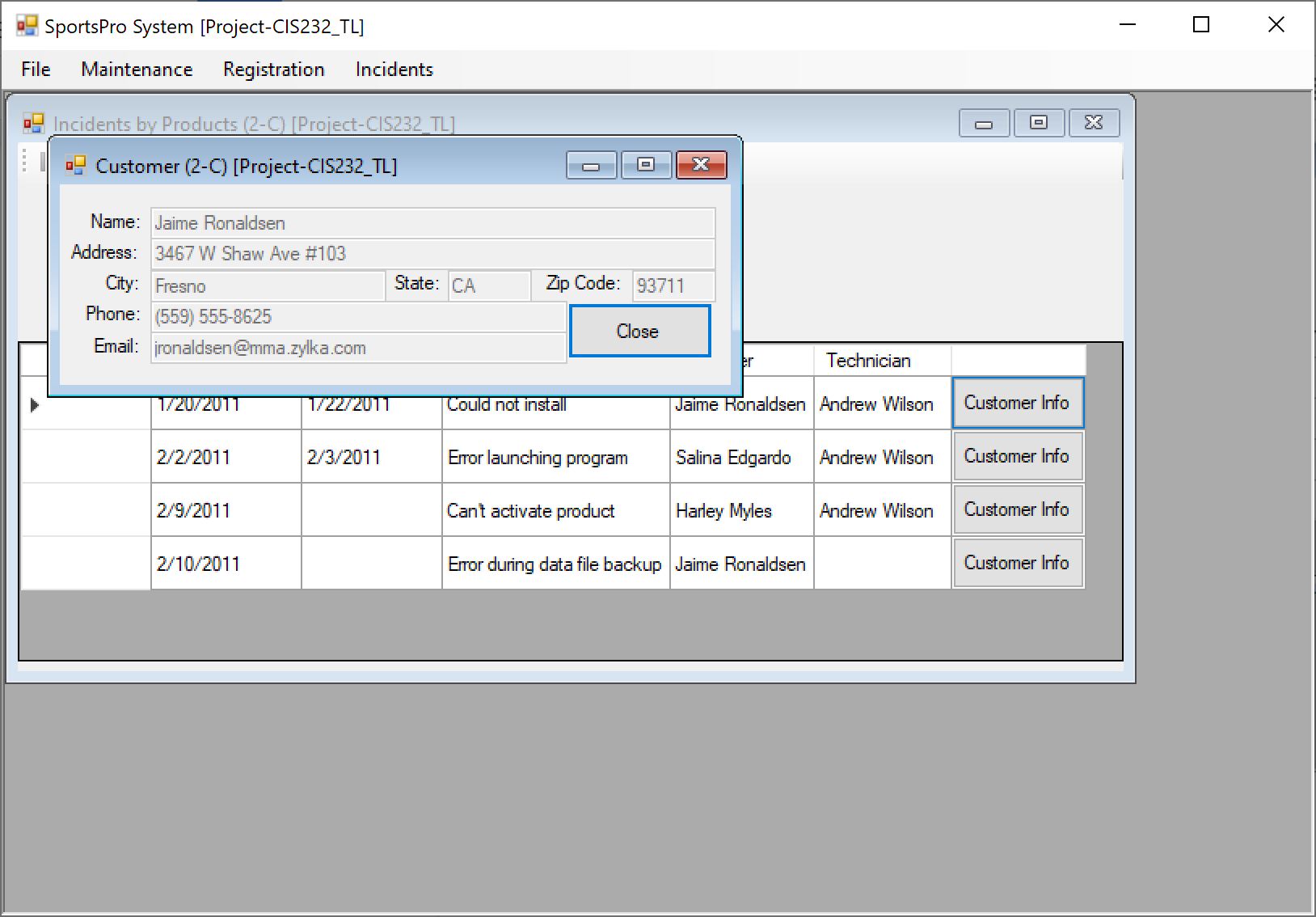
End Class

# Project 2C: Incidents by Customers

## The Incidents by Product form



## The Customer form



## Project items

|  |  |
| --- | --- |
| Name | Description |
| frmProductIncident | A Master/Detail form that displays the data for a product and lists the incidents for that product. |
| frmCustomer | A form that displays the data for a customer. |
| sTechSupportDataSet2C | A typed dataset with three tables named Products, Incidents, and Customers. |

## Operation of the Incidents by Product form

* The Incidents by Product form should be displayed when the user chooses the Incidents → Display Incidents by Product command from the menu on the main form.
* The user can use the binding navigator control to scroll through the products and display the product and incident data.
* To display the customer information for an incident, the user clicks the Customer Info button for that incident.

## Operation of the Customer form

* The Customer form should be displayed when the user clicks a Customer Info button on the Incidents by Product form.
* After reviewing the customer data, the user can click the form’s Close button to close the form and return to the Incidents by Product form

## Code (frmProductIncidents)

Public Class frmProductIncidents

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

'TODO: This line of code loads data into the 'TechSupportDataSet2C.Incidents' table. You can move, or remove it, as needed.

'Me.IncidentsTableAdapter.Fill(Me.TechSupportDataSet2C.Incidents)

'TODO: This line of code loads data into the 'TechSupportDataSet2C.Products' table. You can move, or remove it, as needed.

'Me.ProductsTableAdapter.Fill(Me.TechSupportDataSet2C.Products)

Try

Me.IncidentsTableAdapter.Fill(Me.TechSupportDataSet2C.Incidents)

Me.ProductsTableAdapter.Fill(Me.TechSupportDataSet2C.Products)

Catch er As Exception

MessageBox.Show(er.Message, er.GetType.ToString)

End Try

End Sub

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

'event handler to get customer id and display customer info in child form

If e.ColumnIndex = 5 Then

Dim rowIndex As Integer = e.RowIndex

Dim rowType As DataGridViewRow = IncidentsDataGridView.Rows(rowIndex)

Dim rowCell As DataGridViewCell = rowType.Cells(3)

Dim customerNameParameter As String = rowCell.Value

Dim customerIDParameter As Integer = Me.IncidentsTableAdapter.GetDataByName(customerNameParameter)

Try

'Type Casting will not work

'Dim customerInstance As Customer = CType(row.DataBoundItem, Customer)

'open instance of child form

Dim newMDIChild As New frmCustomer

newMDIChild.customerID = customerIDParameter

newMDIChild.MdiParent = frmMain

newMDIChild.Show()

Catch er As Exception

MessageBox.Show(er.Message, er.GetType.ToString)

End Try

End If

End Sub

End Class

## Code (frmCustomer)

Public Class frmCustomer

Public customerID As Integer

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

'fill the datatable with the global variable

Try

Me.CustomersTableAdapter.Fill(Me.TechSupportDataSet2C.Customers, customerID)

Catch ex As System.Exception

System.Windows.Forms.MessageBox.Show(ex.Message)

End Try

'code that wires procedure to format event (p. 103)

Dim b As Binding = ZipCodeTextBox.DataBindings("Text")

AddHandler b.Format, AddressOf frmMain.FormatZipCode

AddHandler b.Parse, AddressOf frmMain.UnformatZipCode

End Sub

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

'close button

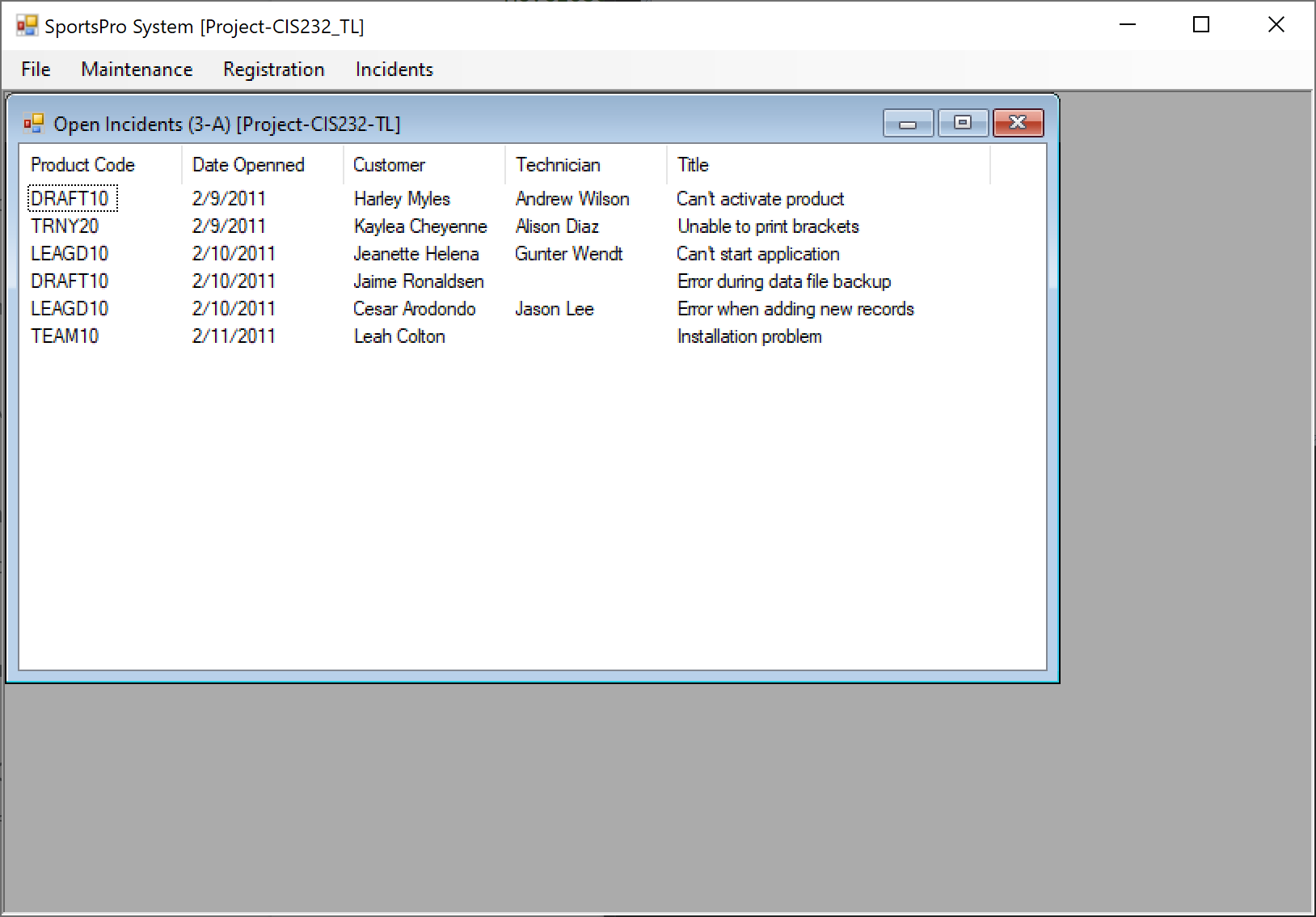
Me.Close()

End Sub

End Class

# Project 3A: Open Incidents

## The Open Incident form



## 

## SportPro project item

|  |  |
| --- | --- |
| Name | Description |
| frmOpenIncidents | A form that displays open incidents in a ListView control. |

## TechSupportData project items

|  |  |
| --- | --- |
| Name | Description |
| Incident | A business class that represents a single incident. |
| TechSupportDB | A database class that contains a method that returns a connection object for the TechSupport database. |
| IncidentDB | A database class that contains methods for working with the Incidents table in the TechSupport database. |
| CustomerDB | A database class that contains methods for working with the Customers table in the TechSupport database. |
| TechnicianDB | A database class that contains methods for working with the Technicians table in the TechSupport database |

## Operation of the Open Incidents form

* The Open Incidents form should be displayed when the user chooses the Incidents → Display Open Incidents command from the menu on the main form.

## Code (frmOpenIncidents)

Imports TechSupportData

Public Class frmOpenIncidents

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

Dim listIncident As List(Of Incident)

Try

listIncident = IncidentDB.GetOpenIncidents()

If listIncident.Count > 0 Then

Dim incident As Incident

For i As Integer = 0 To listIncident.Count - 1

incident = listIncident(i)

lvIncidents.Items.Add(incident.ProductCode)

lvIncidents.Items(i).SubItems.Add(CDate(incident.DateOpened).ToShortDateString)

lvIncidents.Items(i).SubItems.Add(incident.CustomerName)

lvIncidents.Items(i).SubItems.Add(incident.TechnicianName)

lvIncidents.Items(i).SubItems.Add(incident.Title)

Next

Else

MessageBox.Show("There are currently no open incidents.", "Notice")

Me.Close()

End If

Catch ex As Exception

MessageBox.Show(ex.Message, ex.GetType.ToString)

Me.Close()

End Try

End Sub

End Class

## Code (Incident)

Public Class Incident

Private m\_IncidentID As Integer

Private m\_CustomerID As Integer

Private m\_ProductCode As String

Private m\_TechID As Nullable(Of Integer)

Private m\_DateOpened As Date

Private m\_DateClosed As Nullable(Of Date)

Private m\_Title As String

Private m\_Description As String

Public Property IncidentID() As Integer

Get

Return m\_IncidentID

End Get

Set(ByVal value As Integer)

m\_IncidentID = value

End Set

End Property

Public Property CustomerID() As Integer

Get

Return m\_CustomerID

End Get

Set(ByVal value As Integer)

m\_CustomerID = value

End Set

End Property

Public Property ProductCode() As String

Get

Return m\_ProductCode

End Get

Set(ByVal value As String)

m\_ProductCode = value

End Set

End Property

Public Property TechID() As Nullable(Of Integer)

Get

If m\_TechID.HasValue() Then

Return m\_TechID

Else

Return Nothing

End If

End Get

Set(ByVal value As Nullable(Of Integer))

m\_TechID = value

End Set

End Property

Public Property DateOpened() As Date

Get

Return m\_DateOpened

End Get

Set(ByVal value As Date)

m\_DateOpened = value

End Set

End Property

Public Property DateClosed() As Nullable(Of Date)

Get

If m\_DateClosed.HasValue Then

Return m\_DateClosed

Else

Return Nothing

End If

End Get

Set(ByVal value As Nullable(Of Date))

m\_DateClosed = value

End Set

End Property

Public Property Title() As String

Get

Return m\_Title

End Get

Set(ByVal value As String)

m\_Title = value

End Set

End Property

Public Property Description() As String

Get

Return m\_Description

End Get

Set(ByVal value As String)

m\_Description = value

End Set

End Property

Public ReadOnly Property CustomerName() As String

Get

Return CustomerDB.GetCustomerName(CustomerID)

End Get

End Property

Public ReadOnly Property TechnicianName() As String

Get

Return TechnicianDB.GetTechnicianName(TechID)

End Get

End Property

End Class

## Code (TechSupportDB)

Imports System.Data.OleDb

Public Class TechSupportDB

Public Shared Function GetConnection() As OleDbConnection

Dim connectionString As String = "Provider = Microsoft.Jet.OLEDB.4.0;Data Source=C:\Bob\TechSupport.mdb"

Return New OleDbConnection(connectionString)

End Function

End Class

## Code (IncidentDB)

Imports System.Data.OleDb

Public Class IncidentDB

Public Shared Function GetOpenIncidents() As List(Of Incident)

Dim openIncidents As New List(Of Incident)

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectstatement As String = "SELECT CustomerID, ProductCode, TechID, DateOpened, Title " &

"FROM Incidents " &

"WHERE DateClosed Is NULL "

Dim selectCommand As New OleDbCommand(selectstatement, connection)

Try

connection.Open()

Dim reader As OleDbDataReader = selectCommand.ExecuteReader()

Dim incident As Incident

Do While reader.Read

incident = New Incident

incident.CustomerID = CInt(reader("CustomerID"))

incident.ProductCode = reader("ProductCode").ToString

incident.DateOpened = CDate(reader("DateOpened"))

incident.Title = reader("Title").ToString

If IsDBNull(reader("TechID")) Then

incident.TechID = Nothing

Else

incident.TechID = CInt(reader("TechID"))

End If

openIncidents.Add(incident)

Loop

reader.Close()

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

End Try

Return openIncidents

End Function

End Class

## Code (CustomerDB)

Imports System.Data.OleDb

Public Class CustomerDB

Public Shared Function GetCustomerName(ByVal customerID As Integer) As String

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectCommand As New OleDbCommand()

Dim customerName As String

selectCommand.Connection = connection

selectCommand.CommandText =

"SELECT Name " &

"FROM Customers " &

"WHERE CustomerID = " & customerID

Try

connection.Open()

customerName = selectCommand.ExecuteScalar.ToString

connection.Close()

Catch ex As Exception

Throw ex

End Try

Return customerName

End Function

End Class

## Code (TechnicanDB)

Imports System.Data.OleDb

Public Class TechnicianDB

Public Shared Function GetTechnicianName(ByVal techID As Integer) As String

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectCommand As New OleDbCommand()

Dim technicianName As String

Dim objTechName As Object

selectCommand.Connection = connection

selectCommand.CommandText =

"SELECT Name " &

"FROM Technicians " &

"WHERE TechID = " & techID

Try

connection.Open()

objTechName = selectCommand.ExecuteScalar()

If objTechName Is Nothing Then

technicianName = ""

Else

technicianName = objTechName.ToString

End If

connection.Close()

Catch ex As Exception

Throw ex

End Try

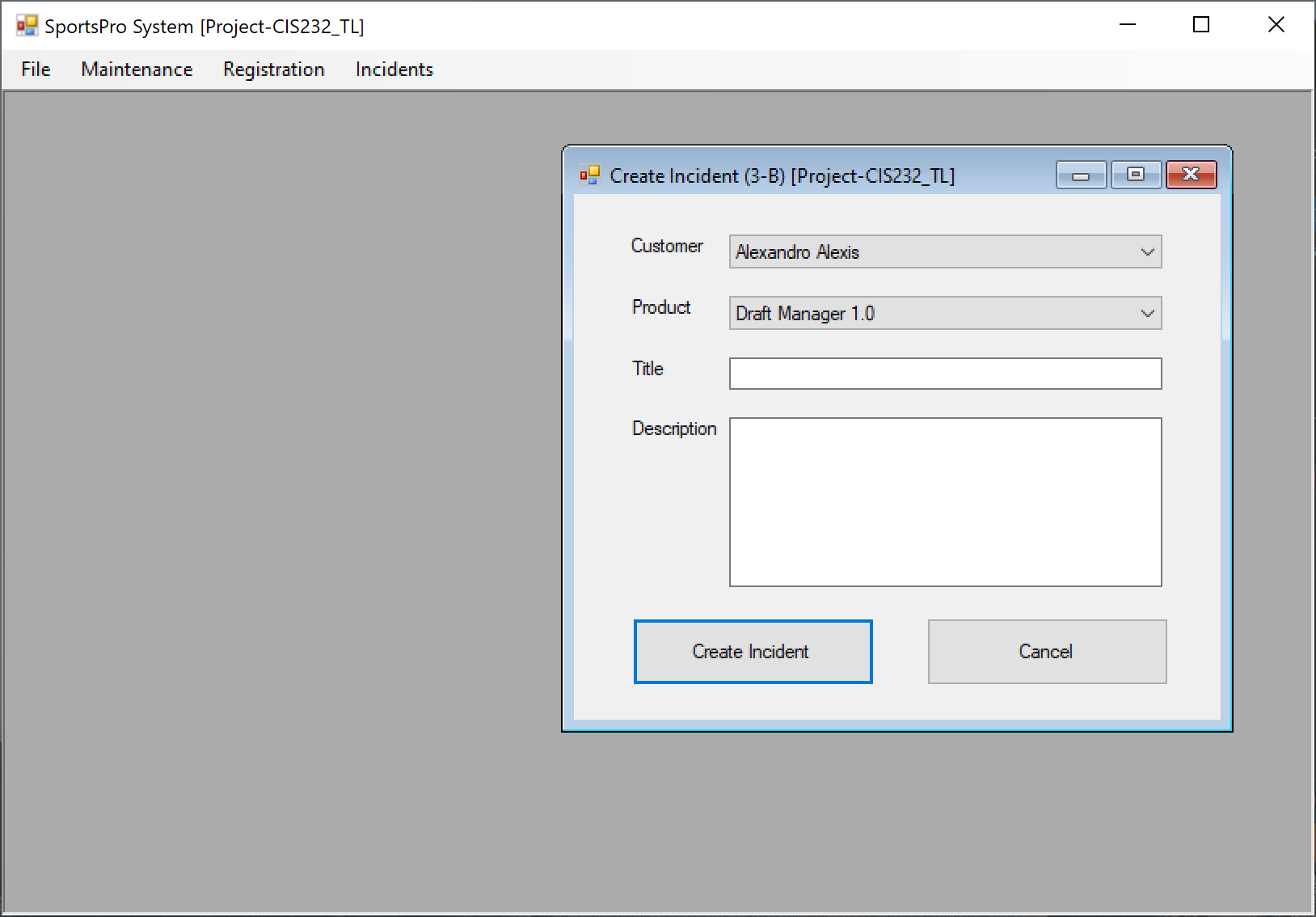
Return technicianName

End Function

End Class

# Project 3B: Create an incident

## The Create Incident form



## SportsPro project items

|  |  |
| --- | --- |
| Name | Description |
| frmCreateIncident | A form that lets the user add a new incident to the Incidents table. |
| Validator | A class that contains generic data validation methods. |

## TechSupportData project items

|  |  |
| --- | --- |
| Name | Description |
| Incident | A business class that represents a single incident. |
| Customer | A business class that represents a single customer. |
| Product | A business class that represents a single product. |
| TechSupportDB | A database class that contains a method that returns a connection object for the TechSupport database. |
| IncidentDB | A database class that contains methods for working with the Incidents table in the TechSupport database. |
| CustomerDB | A database class that contains methods for working with the Customers table in the TechSupport database. |
| ProductDB | A database class that contains methods for working with the Products table in the TechSupport database. |
| RegistrationDB | A database class that contains methods for working with the Registrations table in the TechSupport database. |

## Operation

* The Create Incident form should be displayed when the user chooses the Incidents → Create Incident command from the menu on the main form.
* To create an incident, the user selects the customer and product from the combo boxes, enters a title and description, and clicks the Create Incident button. If the incident is accepted, a confirmation message is displayed and the form is closed.
* To close the form without creating an incident, the user clicks the Cancel button

## Code (frmCreateIncident)

Imports TechSupportData

Public Class frmCreateIncident

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

Me.Close()

End Sub

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

'Check if title and desription is filled. If not, display error message and return to form

'Then, check if customer is associated with product

'Then, set value to incident table

'Display successful then close the form

Try

If Validator.IsPresent(txtTitle, "Title") AndAlso Validator.IsPresent(txtDesc, "Description") Then

If RegistrationDB.ProductRegistered(cboCustomer.SelectedValue, cboProduct.SelectedValue) Then

Dim incident As New Incident

incident.CustomerID = CInt(cboCustomer.SelectedValue)

incident.ProductCode = cboProduct.SelectedValue

incident.Title = txtTitle.Text

incident.Description = txtDesc.Text

Try

IncidentDB.AddIncident(incident)

MessageBox.Show("Successfully added incident", "Confirmation")

Me.Close()

Catch ex As Exception

MessageBox.Show(ex.Message, ex.GetType.ToString)

End Try

Else

MessageBox.Show("Customer and Product does not match", "Error")

End If

End If

Catch ex As Exception

MessageBox.Show(ex.Message, ex.GetType.ToString)

End Try

End Sub

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

'Load combo boxes

Me.LoadComboBoxes()

End Sub

Private Sub LoadComboBoxes()

Try

Dim customerList As List(Of Customer)

customerList = CustomerDB.GetCustomerList

cboCustomer.DataSource = customerList

cboCustomer.DisplayMember = "Name"

cboCustomer.ValueMember = "CustomerID"

Dim productList As List(Of Product)

productList = ProductDB.GetProductList

cboProduct.DataSource = productList

cboProduct.DisplayMember = "Name"

cboProduct.ValueMember = "ProductCode"

Catch ex As Exception

MessageBox.Show(ex.Message, ex.GetType.ToString)

End Try

End Sub

End Class

## Code (Validator)

Public Class Validator

Public Shared Function IsPresent(ByVal textBox As TextBox, ByVal name As String) As Boolean

'Returns a Boolean value that indicates if a text box contains a value. If not, an error message Is displayed And the focus Is set to the text box.

Dim check As Boolean = False

If textBox.Text <> "" Then

check = True

Else

MessageBox.Show(name + " is missing", "Error")

End If

Return check

End Function

End Class

## Code (CustomerDB // GetCustomerList Function)

Public Shared Function GetCustomerList() As List(Of Customer)

Dim customerList As New List(Of Customer)

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectStatement As String =

"SELECT CustomerID, Name " &

"FROM Customers " &

"ORDER BY Name"

Dim selectCommand As New OleDbCommand(selectStatement, connection)

Try

connection.Open()

Dim reader As OleDbDataReader = selectCommand.ExecuteReader()

Dim customer As Customer

Do While reader.Read

customer = New Customer

customer.CustomerID = CInt(reader("CustomerID"))

customer.Name = reader("Name").ToString

customerList.Add(customer)

Loop

reader.Close()

Catch ex As oledbException

Throw ex

Finally

connection.Close()

End Try

Return customerList

End Function

## Code (ProductDB)

Public Class ProductDB

Public Shared Function GetProductList() As List(Of Product)

Dim productList As New List(Of Product)

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectStatement As String =

"SELECT ProductCode, Name " &

"FROM Products " &

"ORDER BY Name"

Dim selectCommand As New OleDbCommand(selectStatement, connection)

Try

connection.Open()

Dim reader As OleDbDataReader = selectCommand.ExecuteReader()

Dim product As Product

Do While reader.Read

product = New Product

product.ProductCode = reader("ProductCode").ToString

product.Name = reader("Name").ToString

productList.Add(product)

Loop

reader.Close()

Catch ex As oledbException

Throw ex

Finally

connection.Close()

End Try

Return productList

End Function

End Class

## Code (IncidentDB // AddIncident Sub)

Public Shared Sub AddIncident(ByVal incident As Incident)

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectstatement As String = "INSERT INTO Incidents " &

"(CustomerID, ProductCode, DateOpened, Title, Description) " &

"VALUES (@CustomerID, @ProductCode, @DateOpened, @Title, @Description) "

Dim selectCommand As New OleDbCommand(selectstatement, connection)

selectCommand.Parameters.AddWithValue("@CustomerID", incident.CustomerID)

selectCommand.Parameters.AddWithValue("@ProductCode", incident.ProductCode)

selectCommand.Parameters.AddWithValue("@DateOpened", CDate(DateTime.Today))

selectCommand.Parameters.AddWithValue("@Title", incident.Title)

selectCommand.Parameters.AddWithValue("@Description", incident.Description)

Try

connection.Open()

selectCommand.ExecuteNonQuery()

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

End Try

End Sub

## Code (RegistrationDB)

Imports System.Data.OleDb

Public Class RegistrationDB

Public Shared Function ProductRegistered(ByVal customerID As Integer, ByVal productCode As String) As Boolean

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectStatement As String =

"SELECT Count(\*) " &

"FROM Registrations " &

"WHERE CustomerID = @CustomerID " &

"AND ProductCode = @ProductCode"

Dim selectCommand As New OleDbCommand(selectStatement, connection)

Dim check As Boolean = False

selectCommand.Parameters.AddWithValue("@CustomerID", customerID)

selectCommand.Parameters.AddWithValue("@ProductCode", productCode)

Try

connection.Open()

Dim length = CInt(selectCommand.ExecuteScalar)

If length > 0 Then

check = True

End If

connection.Close()

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

End Try

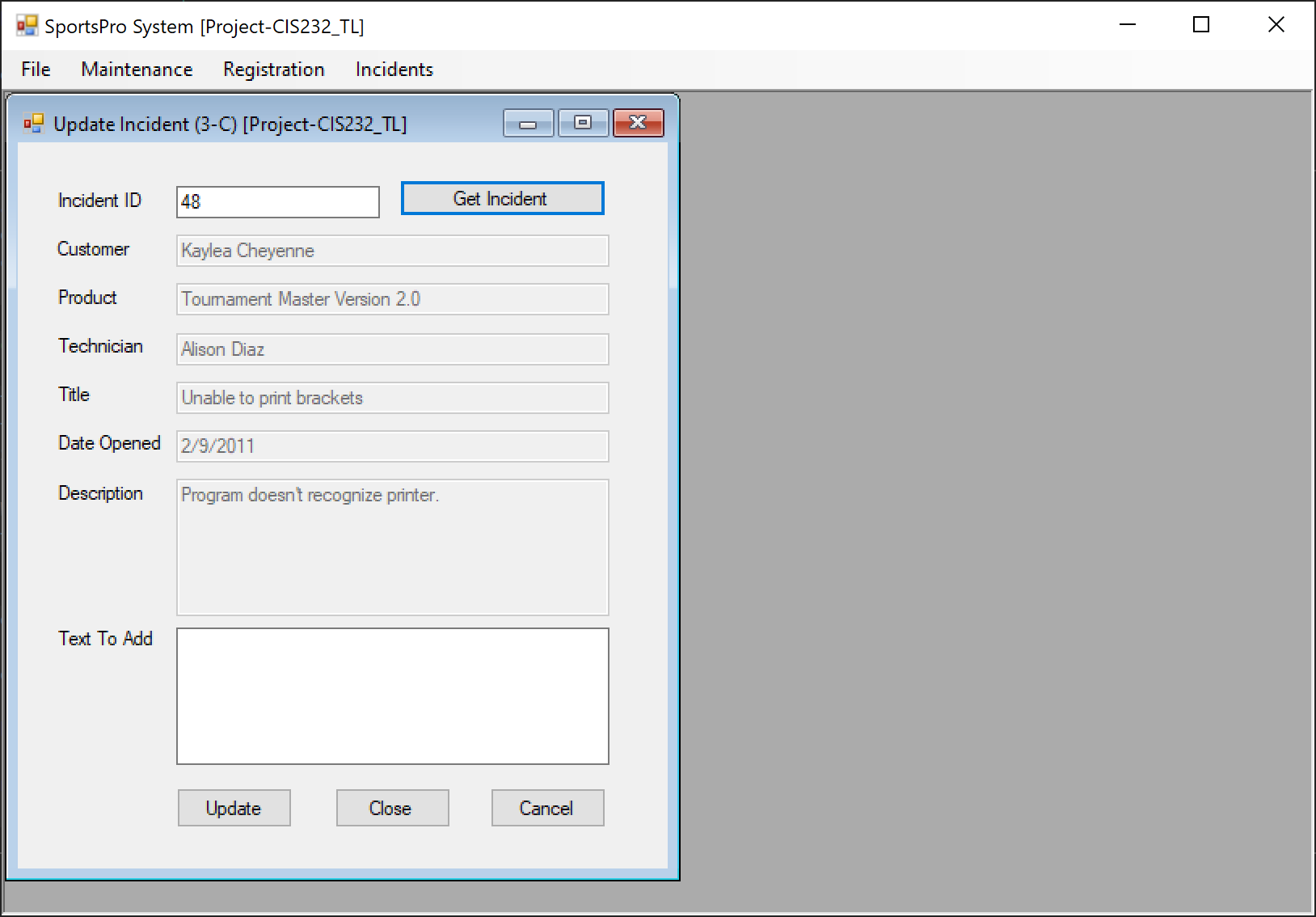
Return check

End Function

End Class

# Project 3C: Update an incident

## The Update Incident form



## SportsPro Project items

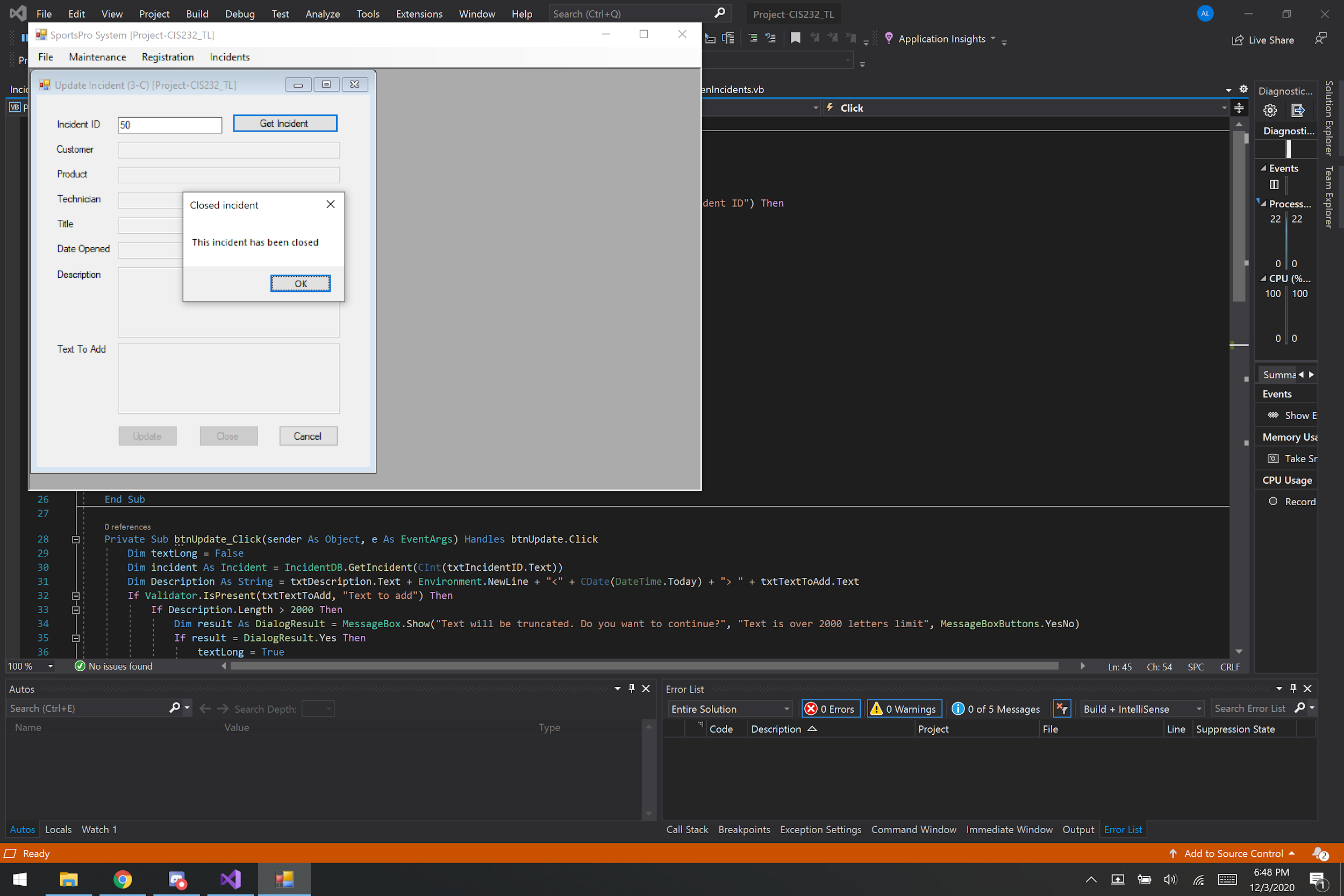
|  |  |
| --- | --- |
| Name | Description |
| frmUpdateIncident | A form that lets the user update or close an incident |
| Validator | A class that contains generic data validation methods. |

## 

## TechSupportData items

|  |  |
| --- | --- |
| Name | Description |
| Incident | A business class that represents a single incident. |
| TechSupportDB | A database class that contains a method that returns a connection object for the TechSupport database. |
| IncidentDB | A database class that contains methods for working with the Incidents table in the TechSupport database |
| CustomerDB | A database class that contains methods for working with the Customers table in the TechSupport database. |
| TechnicianDB | A database class that contains methods for working with the Technicians table in the TechSupport database. |
| ProductDB | A database class that contains methods for working with the Products table in the TechSupport database |

## Operation

* The Update Incident form should be displayed when the user chooses the Incidents → Update Incident command from the menu on the main form.
* To update or close an incident, the user must first retrieve the incident by entering the incident ID in the Incident ID text box and clicking the Get Incident button. The application then displays the information for the incident.
* To update an incident, the user enters text to be added to the Description column in the Text to add text box and clicks the Update button.
* To close an incident, the user clicks the Close Incident button.
* *The form will not open an incident that is already closed.*
* To close the form without updating or closing an incident, the user clicks the Cancel button.

## Code (frmUpdateIncident)

Imports TechSupportData

Public Class frmUpdateIncident

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

Dim incident As Incident

If Validator.IsPresent(txtIncidentID, "Incident ID") AndAlso Validator.IsInt32(txtIncidentID, "Incident ID") Then

Dim incidentID As Integer = CInt(txtIncidentID.Text)

incident = IncidentDB.GetIncident(incidentID)

If incident Is Nothing Then

MessageBox.Show("There are no incidents with this ID", "Unknown ID")

ElseIf Not IncidentDB.UpdateIncident(incident, incident.Description) Then

MessageBox.Show("This incident had been closed", "Closed incident")

Else

txtIncidentID.Text = incident.IncidentID

txtCustomer.Text = incident.CustomerName

txtProduct.Text = incident.ProductName

txtTechnician.Text = incident.TechnicianName

txtTitle.Text = incident.Title

txtDateOpened.Text = incident.DateOpened

txtDescription.Text = incident.Description

txtTextToAdd.Enabled = True

btnClose.Enabled = True

btnUpdate.Enabled = True

End If

End If

End Sub

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

Dim textLong = False

Dim incident As Incident = IncidentDB.GetIncident(CInt(txtIncidentID.Text))

Dim Description As String = txtDescription.Text + Environment.NewLine + "<" + CDate(DateTime.Today) + "> " + txtTextToAdd.Text

If Validator.IsPresent(txtTextToAdd, "Text to add") Then

If Description.Length > 2000 Then

Dim result As DialogResult = MessageBox.Show("Text will be truncated. Do you want to continue?", "Text is over 2000 letters limit", MessageBoxButtons.YesNo)

If result = DialogResult.Yes Then

textLong = True

Description.Substring(0, 2000)

End If

End If

If Description.Length < 2000 OrElse textLong = True Then

If IncidentDB.UpdateIncident(incident, Description) Then

txtDescription.Text = Description

txtTextToAdd.Text = ""

Else

MessageBox.Show("The incident has been closed", "Unable to add description")

txtTextToAdd.Text = ""

End If

End If

End If

End Sub

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

Dim incident As Incident = IncidentDB.GetIncident(CInt(txtIncidentID.Text))

Dim result As DialogResult = MessageBox.Show("This will close the incident. Do you want to continue?", "Close incident?", MessageBoxButtons.YesNo)

If result = DialogResult.Yes Then

If IncidentDB.CloseIncident(incident) Then

MessageBox.Show("Incident has been closed", "Notice")

Me.Close()

Else

MessageBox.Show("Another user has updated this incident", "Unable to close incident")

End If

End If

End Sub

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

Me.Close()

End Sub

End Class

## Code (Validator // IsInt32)

Public Shared Function IsInt32(ByVal textBox As TextBox, ByVal name As String) As Boolean

'Returns Boolean Check if Int32

Try

Convert.ToInt32(textBox.Text)

Return True

Catch ex As FormatException

MessageBox.Show(name + " must be an integer value", "Error")

textBox.Select()

textBox.SelectAll()

Return False

End Try

End Function

## Code (Incident // ProductName)

Public ReadOnly Property ProductName() As String

Get

Return ProductDB.GetProductName(ProductCode)

End Get

End Property

## Code (IncidentDB // GetIncident, UpdateIncident, CloseIncident)

**Public Shared Function GetIncident(ByVal incidentID As Integer) As Incident**

Dim incident As New Incident

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectStatement As String =

"SELECT IncidentID, CustomerID, ProductCode, TechID, " &

"DateOpened, DateClosed, Title, Description " &

"FROM Incidents " &

"WHERE incidentID = @IncidentID "

Dim selectCommand As New OleDbCommand(selectStatement, connection)

selectCommand.Parameters.AddWithValue("@IncidentID", incidentID)

Try

connection.Open()

Dim reader As OleDbDataReader = selectCommand.ExecuteReader(CommandBehavior.SingleRow)

If reader.Read Then

incident.IncidentID = CInt(reader("IncidentID"))

incident.CustomerID = CInt(reader("CustomerID"))

incident.ProductCode = reader("ProductCode").ToString

incident.TechID = CInt(reader("TechID"))

incident.DateOpened = CDate(reader("DateOpened"))

'incident.DateClosed = CDate(reader("DateClosed"))

incident.Title = reader("Title").ToString

incident.Description = reader("Description").ToString

Else

incident = Nothing

End If

reader.Close()

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

End Try

Return incident

End Function

**Public Shared Function UpdateIncident(ByVal incident As Incident, ByVal description As String) As Boolean**

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectStatement As String =

"UPDATE Incidents " &

"SET Description = @NewDescription " &

"WHERE IncidentID = @IncidentID " &

"AND Description = @Description " &

"AND DateClosed IS NULL"

Dim selectCommand As New OleDbCommand(selectStatement, connection)

selectCommand.Parameters.AddWithValue("@NewDescription", description)

selectCommand.Parameters.AddWithValue("@IncidentID", incident.IncidentID)

selectCommand.Parameters.AddWithValue("@Description", incident.Description)

Try

connection.Open()

Dim count As Integer = selectCommand.ExecuteNonQuery

If count > 0 Then

Return True

Else

Return False

End If

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

End Try

End Function

**Public Shared Function CloseIncident(ByVal incident As Incident) As Boolean**

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectStatement As String =

"UPDATE Incidents " &

"SET DateClosed = @DateClosed " &

"WHERE IncidentID = @IncidentID " &

"AND Description = @Description " &

"AND DateClosed IS NULL"

Dim selectCommand As New OleDbCommand(selectStatement, connection)

selectCommand.Parameters.AddWithValue("@DateClosed", CDate(DateTime.Today))

selectCommand.Parameters.AddWithValue("@IncidentID", incident.IncidentID)

selectCommand.Parameters.AddWithValue("@Description", incident.Description)

Try

connection.Open()

Dim count As Integer = selectCommand.ExecuteNonQuery

If count > 0 Then

Return True

Else

Return False

End If

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

End Try

End Function

## Code (ProductDB // GetProductName)

Public Shared Function GetProductName(ByVal productCode As String) As String

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectCommand As New OleDbCommand()

Dim productName As String

'Dim objReturn As Object

selectCommand.Connection = connection

selectCommand.CommandText =

"SELECT Name " &

"FROM Products " &

"WHERE ProductCode = @productCode"

selectCommand.Parameters.AddWithValue("@productCode", productCode)

Try

connection.Open()

productName = selectCommand.ExecuteScalar.ToString

connection.Close()

Catch ex As Exception

Throw ex

End Try

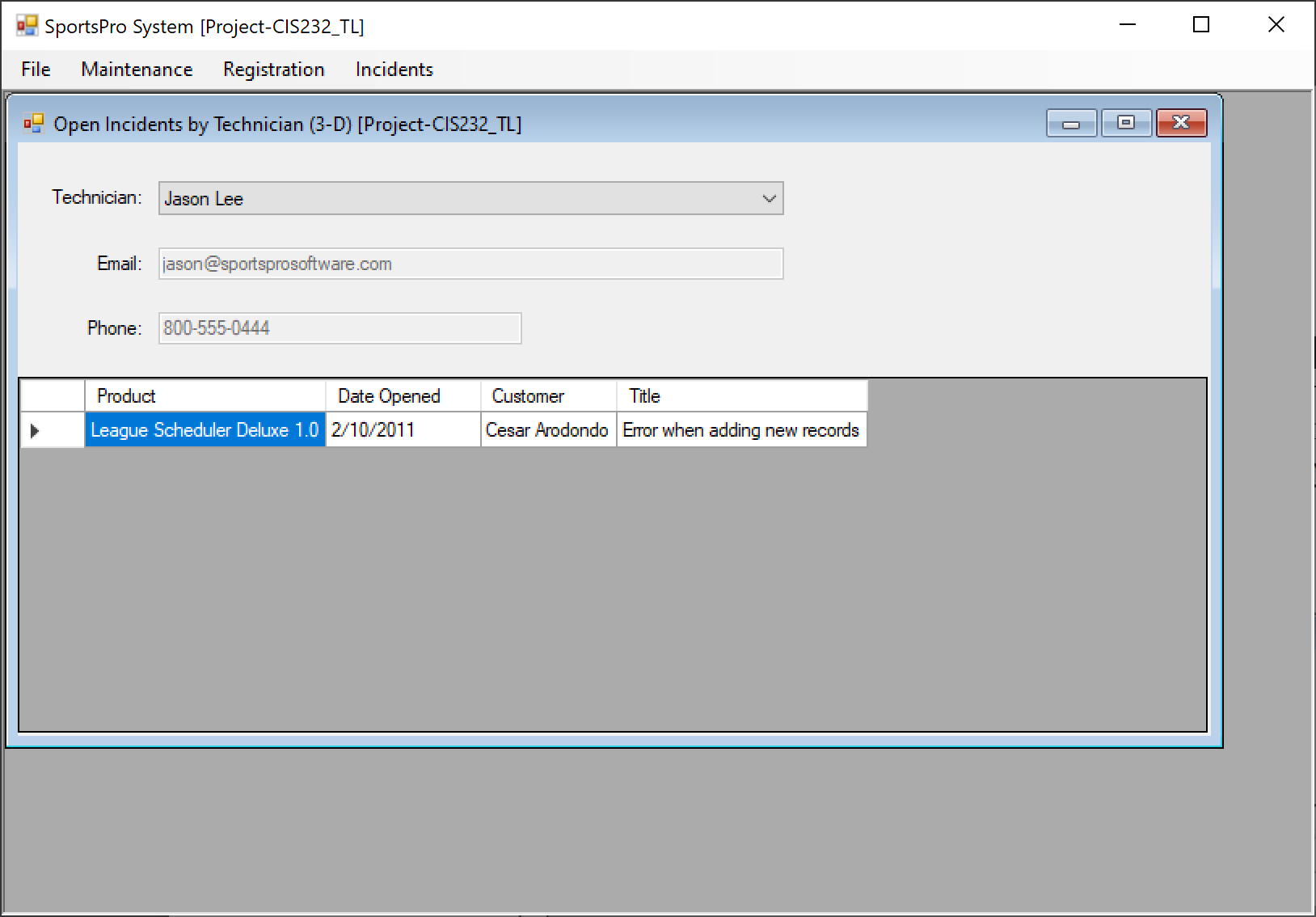
Return productName

End Function

# 

# Project 3D: Display open incidents by technician

## The Open Incidents by Technicians form



## SportsPro project item

|  |  |
| --- | --- |
| Name | Description |
| frmTechnicianIncidents | A form that lets the user display the open incidents for a technician. |

## TechSupportData project items

|  |  |
| --- | --- |
| Name | Description |
| Technician | A business class that represents a single technician. |
| Incident | A business class that represents a single incident. |
| TechSupportDB | A database class that contains a method that returns a connection object for the TechSupport database. |
| TechnicianDB | A database class that contains methods for working with the Technicians table in the TechSupport database. |
| IncidentDB | A database class that contains methods for working with the Incidents table in the TechSupport database. |
| ProductDB | A database class that contains methods for working with the Products table in the TechSupport database. |
| CustomerDB | A database class that contains methods for working with the Customers table in the TechSupport database. |

## Operation

* The Open Incidents by Technician form should be displayed when the user chooses the Incidents → Display Open Incidents by Technician command from the menu on the main form.
* To display the open incidents for a technician, the user selects the technician from the combo box. In addition to the incidents, contact information for the selected technician is displayed on the form

## Code (frmTechnicianIncident)

Imports TechSupportData

Public Class frmTechnicianIncidents

Private tech As Technician

Private techList As List(Of Technician)

Private incidentList As List(Of Incident)

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

Me.GetTechList()

Me.GetIncidentList()

End Sub

Private Sub GetTechList()

Try

techList = TechnicianDB.GetTechnicianList

cboTech.DataSource = techList

Catch ex As Exception

MessageBox.Show(ex.Message, ex.GetType.ToString)

End Try

End Sub

Private Sub GetIncidentList()

Dim techID As Integer = CInt(cboTech.SelectedValue)

Try

tech = TechnicianDB.GetTechnician(techID)

TechnicianBindingSource.Clear()

TechnicianBindingSource.Add(tech)

incidentList = IncidentDB.GetOpenTechnicianIncidents(techID)

IncidentDataGridView.DataSource = incidentList

Catch ex As Exception

MessageBox.Show(ex.Message, ex.GetType.ToString)

End Try

End Sub

Private Sub cboTech\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cboTech.SelectedIndexChanged

Me.GetIncidentList()

End Sub

End Class

## Code (Technician)

Public Class Technician

Private m\_TechID As Integer

Private m\_Name As String

Private m\_Email As String

Private m\_Phone As String

Public Property TechID() As Integer

Get

Return m\_TechID

End Get

Set(ByVal value As Integer)

m\_TechID = value

End Set

End Property

Public Property Name() As String

Get

Return m\_Name

End Get

Set(ByVal value As String)

m\_Name = value

End Set

End Property

Public Property Email() As String

Get

Return m\_Email

End Get

Set(ByVal value As String)

m\_Email = value

End Set

End Property

Public Property Phone() As String

Get

Return m\_Phone

End Get

Set(ByVal value As String)

m\_Phone = value

End Set

End Property

End Class

## Code (TechnicianDB // GetTechnicianList, GetTechnician)

**Public Shared Function GetTechnicianList() As List(Of Technician**)

Dim technicianList As New List(Of Technician)

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectstatement As String =

"SELECT TechID, Name " &

"FROM Technicians " &

"ORDER BY Name "

Dim selectCommand As New OleDbCommand(selectstatement, connection)

Try

connection.Open()

Dim reader As OleDbDataReader = selectCommand.ExecuteReader()

Dim technician As Technician

Do While reader.Read

technician = New Technician

technician.TechID = CInt(reader("TechID"))

technician.Name = reader("Name").ToString

'technician.Phone = reader("Phone").ToString

'technician.Email = reader("Email").ToString

technicianList.Add(technician)

Loop

reader.Close()

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

End Try

Return technicianList

End Function

**Public Shared Function GetTechnician(ByVal techID As Integer) As Technician**

Dim technician As New Technician

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectstatement As String =

"SELECT TechID, Name, Email, Phone " &

"FROM Technicians " &

"WHERE TechID = @TechID "

Dim selectCommand As New OleDbCommand(selectstatement, connection)

selectCommand.Parameters.AddWithValue("@TechID", techID)

Try

connection.Open()

Dim reader As OleDbDataReader = selectCommand.ExecuteReader(CommandBehavior.SingleRow)

If reader.Read Then

technician = New Technician

technician.TechID = CInt(reader("TechID"))

technician.Name = reader("Name").ToString

technician.Phone = reader("Phone").ToString

technician.Email = reader("Email").ToString

Else

technician = Nothing

End If

reader.Close()

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

End Try

Return technician

End Function

## Code (IncidentDB // GetOpenTechnicianIncidents)

Public Shared Function GetOpenTechnicianIncidents(ByVal techID As Integer) As List(Of Incident)

Dim openTechnicianIncidents As New List(Of Incident)

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectstatement As String =

"SELECT CustomerID, ProductCode, DateOpened, Title, Description " &

"FROM Incidents " &

"WHERE TechID = @TechID " &

"AND DateClosed IS NULL "

Dim selectCommand As New OleDbCommand(selectstatement, connection)

selectCommand.Parameters.AddWithValue("@TechID", techID)

Try

connection.Open()

Dim reader As OleDbDataReader = selectCommand.ExecuteReader()

Dim incident As Incident

Do While reader.Read

incident = New Incident

incident.CustomerID = CInt(reader("CustomerID"))

incident.ProductCode = reader("ProductCode").ToString

incident.DateOpened = CDate(reader("DateOpened"))

incident.Title = reader("Title").ToString

incident.Description = reader("Description").ToString

openTechnicianIncidents.Add(incident)

Loop

reader.Close()

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

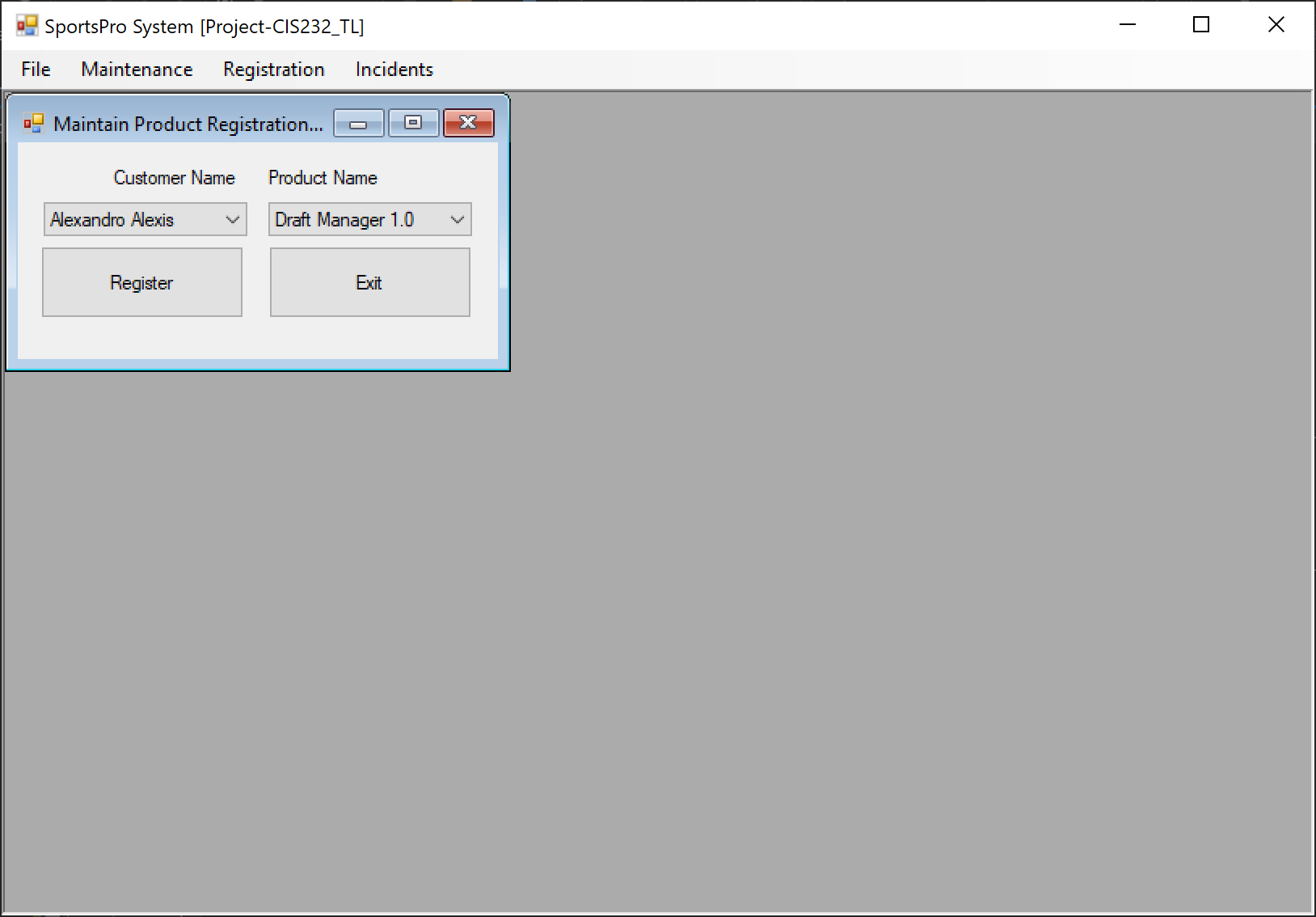
End Try

Return openTechnicianIncidents

End Function

# Project 3E: Maintain product registrations

## The Maintain Product Registrations form



## SportsPro project items

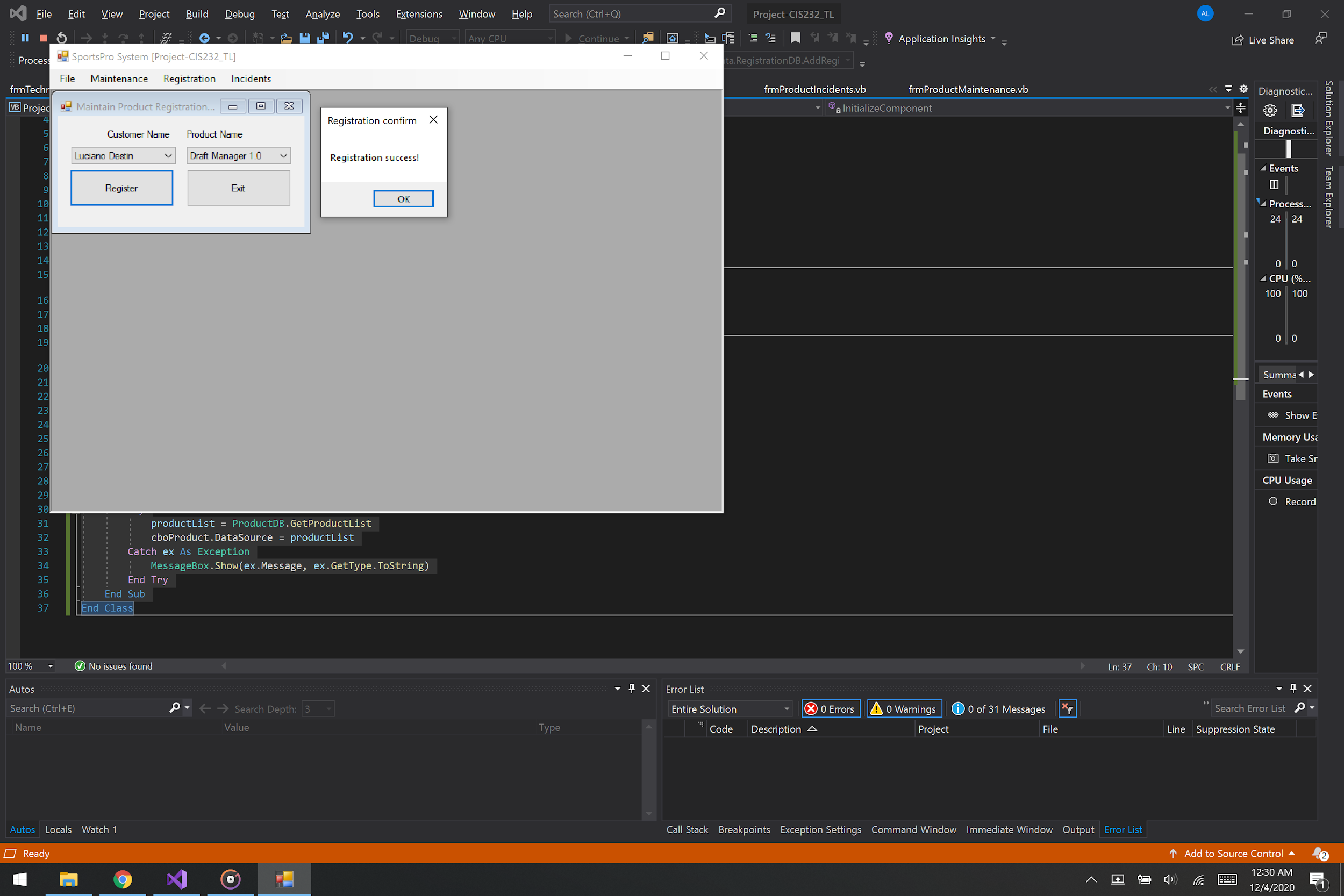
|  |  |
| --- | --- |
| Name | Description |
| frmTechnicianIncidents | A form that lets the user display the open incidents for a technician. |
| Validator | A class that contains generic data validation methods. |

## TechSupportData project items

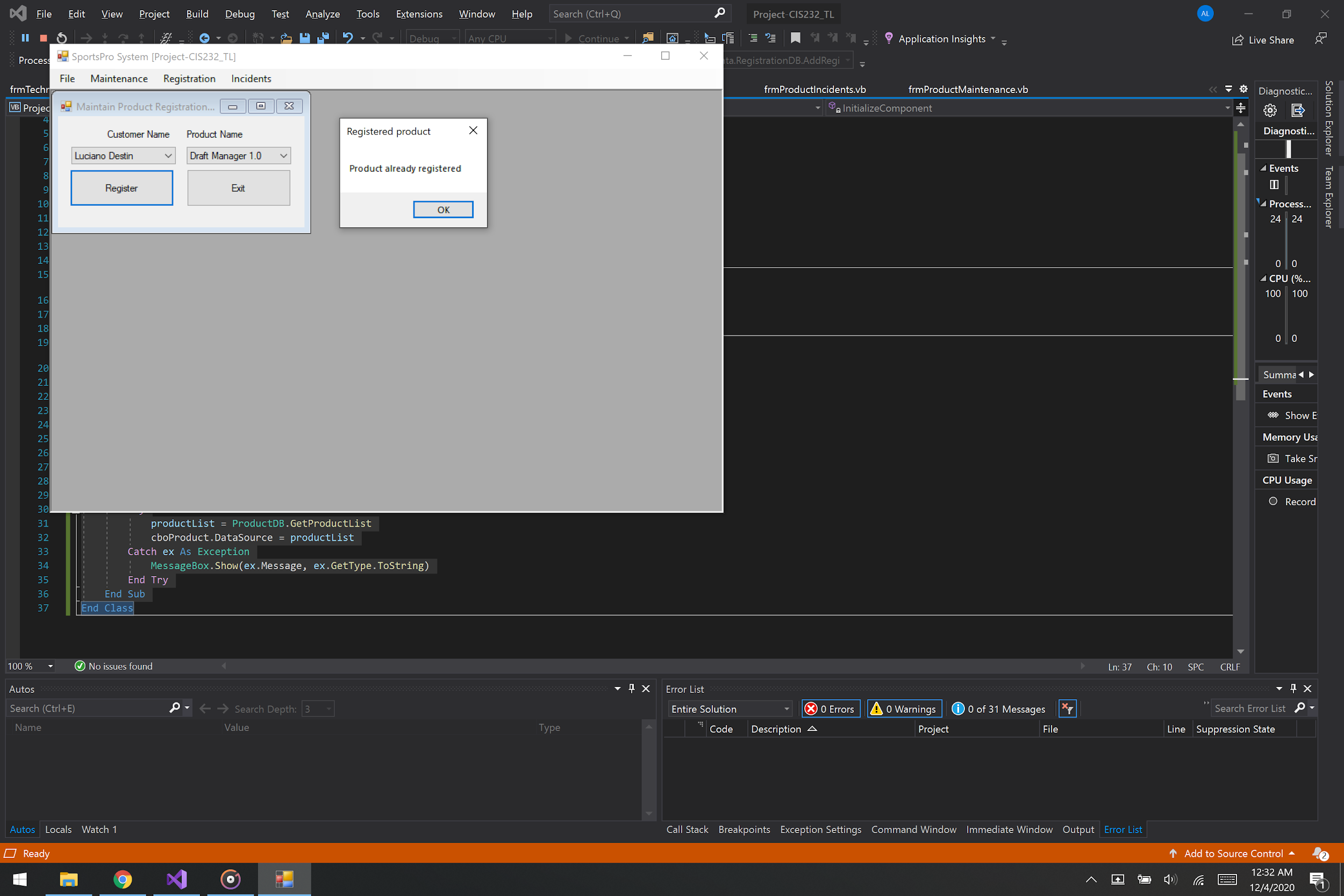
|  |  |
| --- | --- |
| Name | Description |
| Registration | A business class that represents a single registration. |
| Customer | A business class that represents a single customer. |
| Product | A business class that represents a single product. |
| TechSupportDB | A database class that contains a method that returns a connection object for the TechSupport database. |
| CustomerDB | A database class that contains methods for working with the Customers table in the TechSupport database. |
| ProductDB | A database class that contains methods for working with the Products table in the TechSupport database. |
| RegistrationDB | A database class that contains methods for working with the Registrations table in the TechSupport database. |

## Operation

* The Maintain Registrations form should be displayed when the user chooses the Maintenance → Maintain Registrations command from the menu on the main form.



* To create a registration, the user selects the customer and product from the combo boxes, and clicks the Register button. If the incident is accepted, a confirmation message is displayed. If the registration already exists, an information message is displayed.



* To close the form without creating a registration, or after entering registrations, the user clicks the Exit button

## Code (frmMaintainRegistrations)

Imports TechSupportData

Public Class frmMaintainRegistrations

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

If RegistrationDB.ProductRegistered(cboCustomer.SelectedValue, cboProduct.SelectedValue) Then

MessageBox.Show("Product already registered", "Registered product")

Else

Dim registration As New Registration

registration.CustomerID = cboCustomer.SelectedValue

registration.ProductCode = cboProduct.SelectedValue

registration.DateOpened = CDate(DateTime.Today)

RegistrationDB.AddRegistration(registration)

MessageBox.Show("Registration success!", "Registration confirm")

End If

End Sub

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

Me.Close()

End Sub

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

Dim customerList As List(Of Customer)

Dim productList As List(Of Product)

Try

customerList = CustomerDB.GetCustomerList

cboCustomer.DataSource = customerList

Catch ex As Exception

MessageBox.Show(ex.Message, ex.GetType.ToString)

End Try

Try

productList = ProductDB.GetProductList

cboProduct.DataSource = productList

Catch ex As Exception

MessageBox.Show(ex.Message, ex.GetType.ToString)

End Try

End Sub

End Class

## Code (Registration)

Public Class Registration

Private m\_CustomerID As Integer

Private m\_ProductCode As String

Private m\_RegistrationDate As Date

Public Property CustomerID() As Integer

Get

Return m\_CustomerID

End Get

Set(ByVal value As Integer)

m\_CustomerID = value

End Set

End Property

Public Property ProductCode() As String

Get

Return m\_ProductCode

End Get

Set(ByVal value As String)

m\_ProductCode = value

End Set

End Property

Public Property DateOpened() As Date

Get

Return m\_RegistrationDate

End Get

Set(ByVal value As Date)

m\_RegistrationDate = value

End Set

End Property

End Class

## Code (RegistrationDB // AddRegistration)

Public Shared Sub AddRegistration(ByVal registration As Registration)

Dim connection As OleDbConnection = TechSupportDB.GetConnection

Dim selectStatement As String =

"INSERT INTO Registrations " &

"(CustomerID, ProductCode, RegistrationDate) " &

"VALUES(@CustomerID, @ProductCode, @RegistrationDate) "

Dim selectCommand As New OleDbCommand(selectStatement, connection)

selectCommand.Parameters.AddWithValue("@CustomerID", registration.CustomerID)

selectCommand.Parameters.AddWithValue("@ProductCode", registration.ProductCode)

selectCommand.Parameters.AddWithValue("@RegistrationDate", registration.DateOpened)

Try

connection.Open()

selectCommand.ExecuteNonQuery()

Catch ex As OleDbException

Throw ex

Finally

connection.Close()

End Try

End Sub