2021

Project 3-E_YK

YAW KOOSONO

Course: CIS232-40 .NET Programming II

Instructor: Bob Desilets Spring 2021 [v2 02/17/21]

Contents

Program specifications

This software display form to register product with customer. User need to select customer and product to register. The process will be success if it's not a duplicate registration.

It is an offline desktop software which needs OS WIN 2007 and higher also need OLEDB driver installed to run.

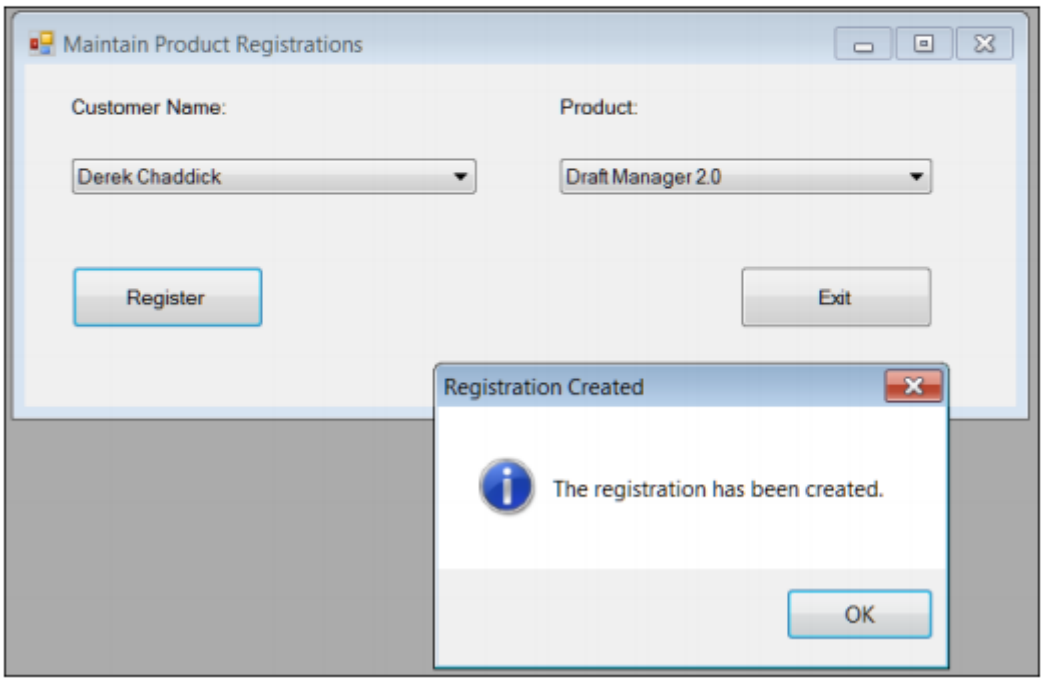
User need to store database file (TechSupport.mdb) into C:\Bob location of same PC where the software has installed.

Hardware: Computer/Laptop, Mouse, Keyboard

Project 3-E: Maintain product registrations

For this project, you'll enhance the SportsPro application by adding a form that lets the user add a row to the Registrations table. To do that, you'll write code to create the command and parameter objects needed to insert a new incident into the table. (*Required reading: chapters 6 and 7.*)

The Maintain Product Registrations form



SportsPro project items

Name	Description
frmMaintainRegistrations	A form that lets the user add a new registration to the Registrations table.
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.

Specifications

1. To get the lists of customers and products that the combo boxes are bound to, use commands and data readers.
2. When the user tries to create a new registration, the program should check that this registration doesn't already exist in the database. If it does, the program should display an error message.
3. To determine if a product is registered to a customer, use a command with a parameterized query that returns a count of the number of rows for the specified customer and product.
4. To add a registration to the Registrations table, use a command with parameters whose values are set to the properties of the Registration object.

The design of the Registration class

The private fields that store the property values

```
Private m_CustomerID As Integer
Private m_ProductCode As String
Private m_RegistrationDate As Date
```

The CustomerID property

```
Public Property CustomerID() As Integer
```

Gets and sets the customer ID for the incident.

The ProductCode property

```
Public Property ProductCode() As String
```

Gets and sets the product code for the incident.

The RegistrationDate property

```
Public Property RegistrationDate() As Date
```

Gets and sets the date the incident was created.

The design of the Customer class

The private fields that store the property values

```
Private m_CustomerID As Integer
Private m_Name As String
```

The CustomerID property

```
Public Property CustomerID() As Integer
```

Gets and sets the ID for the customer.

The Name property

```
Public Property Name() As String
```

Gets and sets the name for the customer.

The design of the Product class

The private fields that store the property values

```
Private m_ProductCode As String
Private m_Name As String
```

The ProductCode property

```
Public ProductCode As String
```

Gets and sets the product code that uniquely identifies the product.

The Name property

```
Public Name As String
```

Gets and sets the name for the product.

The design of the TechSupportDB class

The GetConnection method

```
Public Shared Function GetConnection() As SqlConnection
```

Returns a SqlConnection object that establishes a connection to the TechSupport database.

The design of the CustomerDB class

The GetCustomerList method

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

Returns a List(Of Customer) object that contains one item for each customer in the Customers table.

The design of the ProductDB class

The GetProductList method

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

Returns a List(Of Product) object that contains one item for each product in the Products table.

The design of the RegistrationDB class

The ProductRegistered method

```
Public Shared Function ProductRegistered(
```

```
    ByVal customerID As Integer, ByVal productCode As String) _
    As Boolean
```

Returns a Boolean value that indicates if a product is registered to a customer.

The AddRegistration method

```
Public Shared Sub AddRegistration(ByVal registration As Registration)
```

Adds a row to the Registrations table.

SQL statements

Select statement to get the customer data

```
SELECT CustomerID, Name
FROM Customers
ORDER BY Name
```

Select statement to get the product data

```
SELECT ProductCode, Name
FROM Products
ORDER BY Name
```

Select statement to determine if a customer has registered a product

```
SELECT Count(*)
FROM Registrations
WHERE CustomerID = @CustomerID
  AND ProductCode = @ProductCode
```

Insert statement to insert a registration

```
INSERT Registrations
   (CustomerID, ProductCode, RegistrationDate)
```

VALUES (@CustomerID, @ProductCode, @RegistrationDate

A walk through of the application

Main form display when user run the application. All sub forms can navigate through main form.

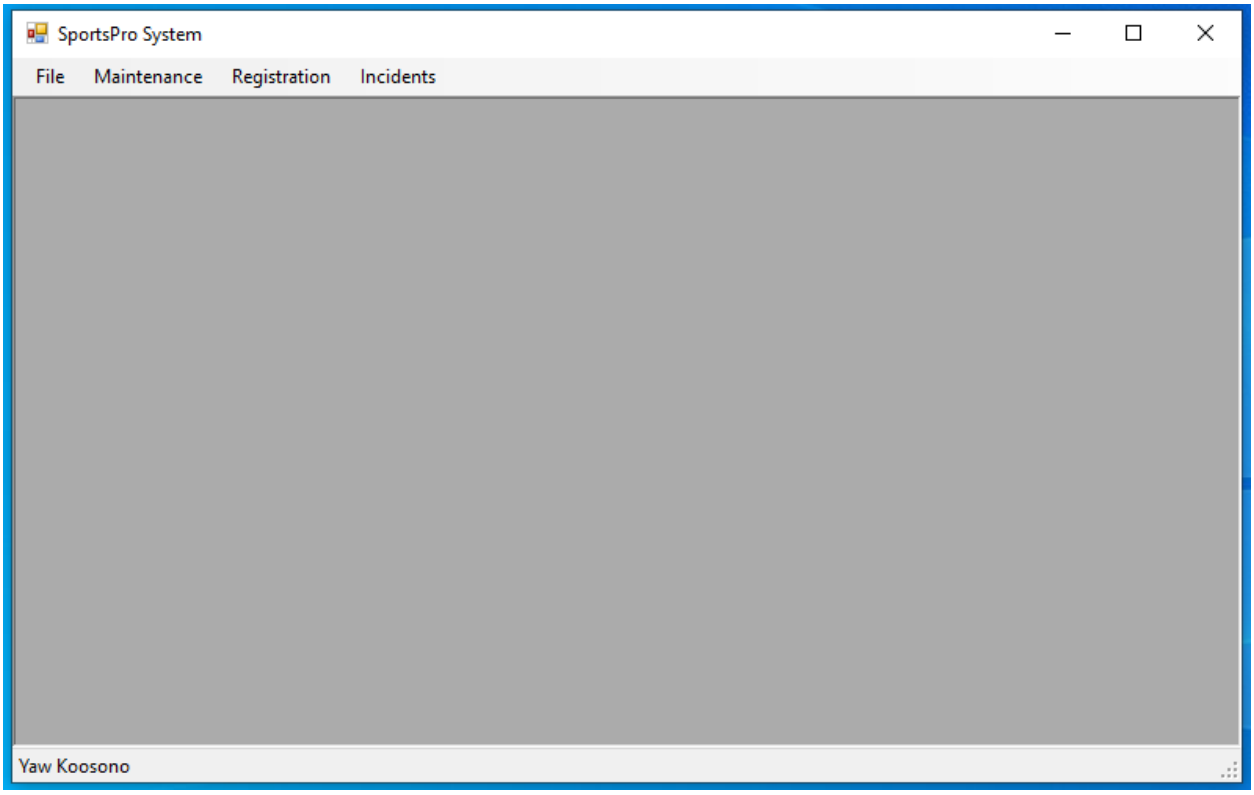


Fig-1: Main form when program start

Displaying the navigation menu for Maintain registrations form.

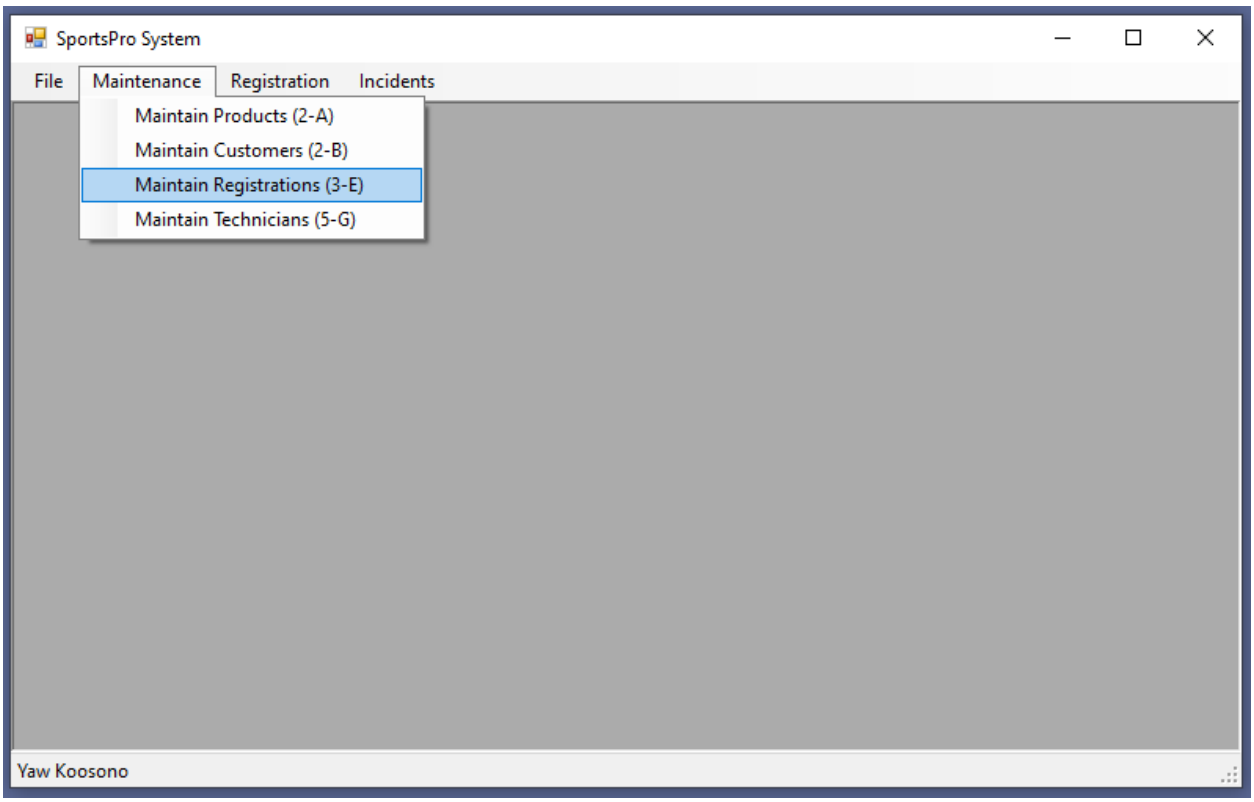


Fig-2: Main form with Maintain Registrations navigation

Maintain registrations form when load.

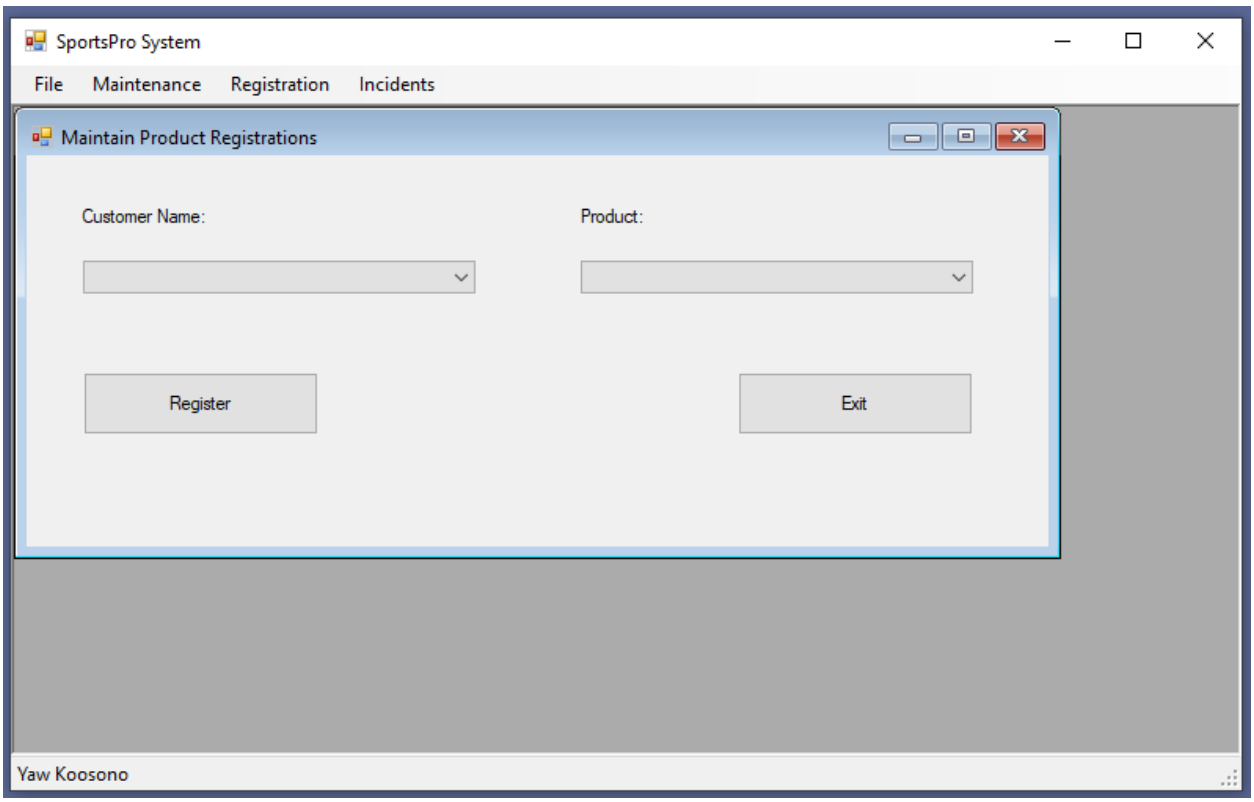


Fig-3: Maintain Registrations form

User need to select a customer before click the ‘Register’ button.

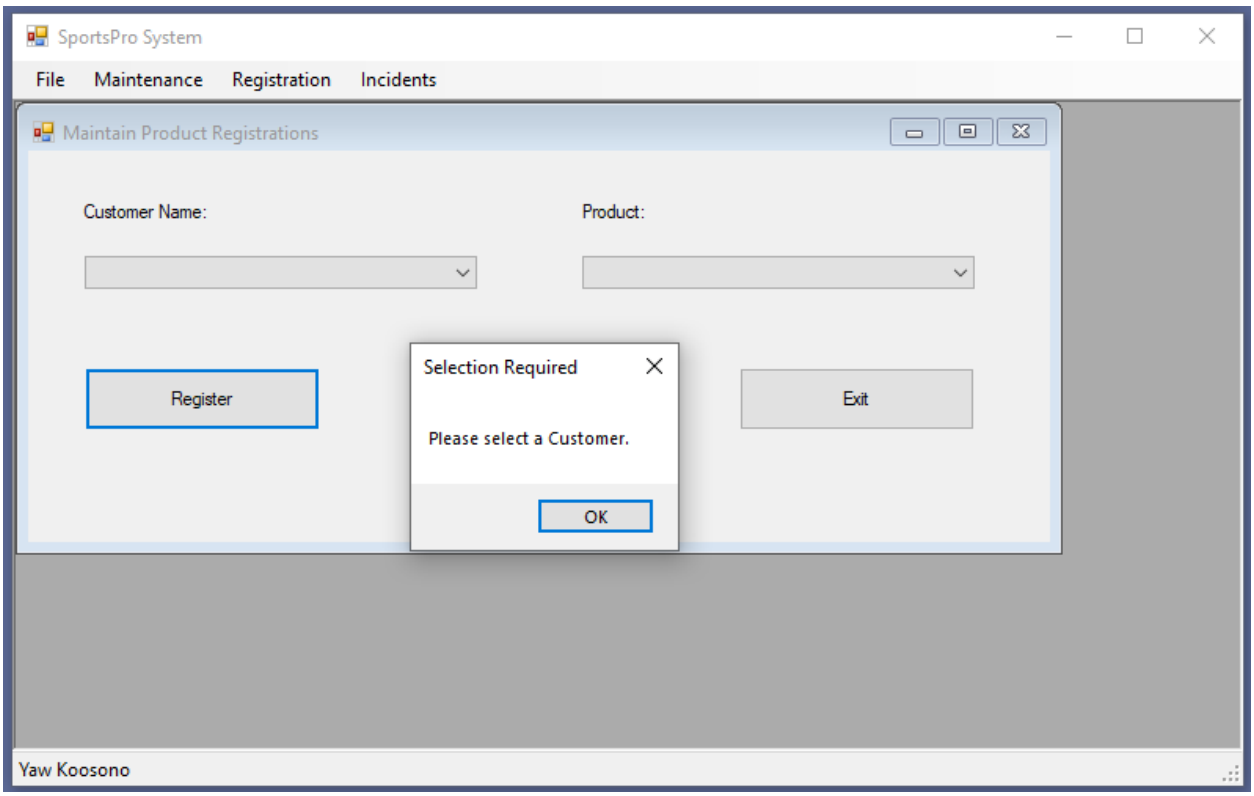


Fig-4: Maintain Registrations form – Customer name required validation

User need to select a product before click the ‘Register’ button.

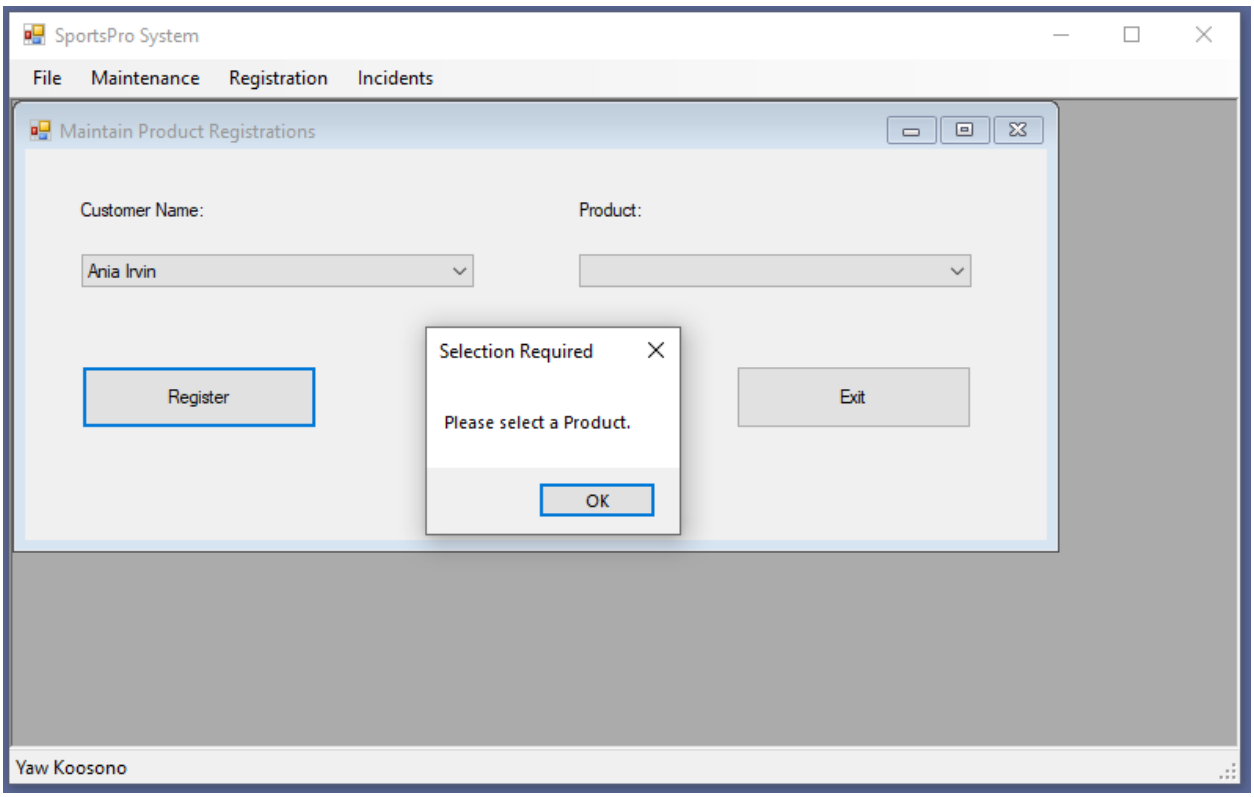


Fig-5: Maintain Registrations form – Product required validation

Displaying success message after registration done.

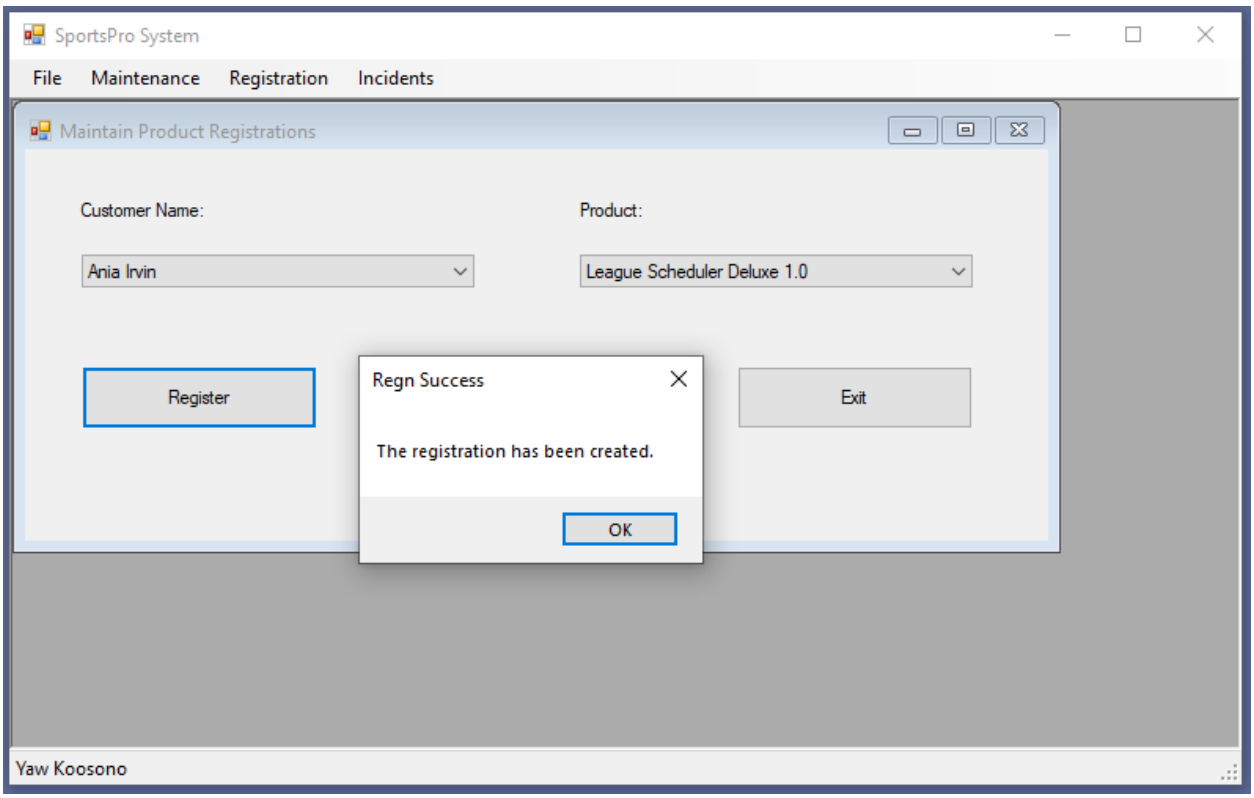


Fig-6: Maintain Registrations form – Successful Registration

Selected customer already has registered with selected product. Duplicate registration not allowed.

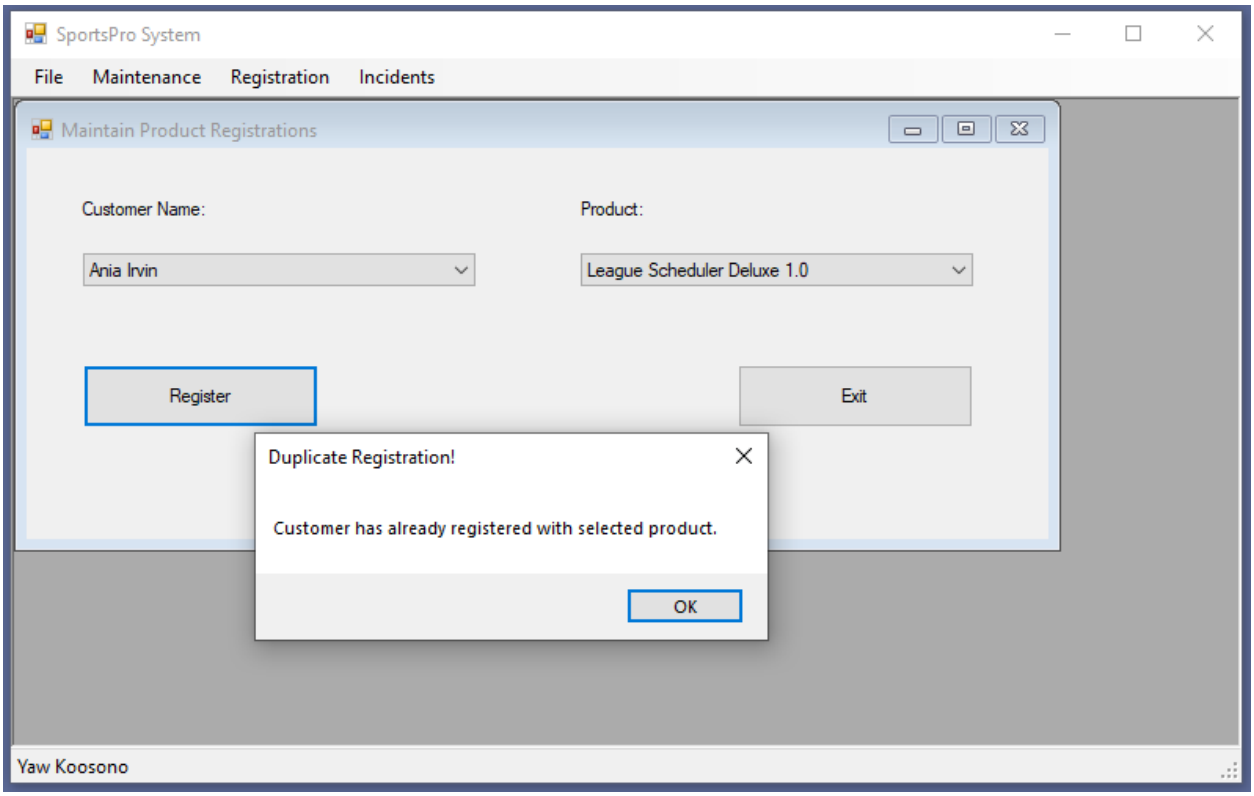


Fig-7: Maintain Registrations form – Duplicate registration validation

Source Code

Classes of TechSupportData class library

Code of TechSupportDB Class (for partial 3E)

```
Imports System.Data.OLEDB
Public Class TechSupportDB
    Public Shared Function GetConnection() As OleDbConnection
        Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Bob\TechSupport.mdb;Persist Security Info=True"

        Return New OleDbConnection(connString)
    End Function
End Class
```

Code of Technician Class

```
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 of Incident Class

```
Imports System.Data.OLEDB
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 Sub New()

    End Sub

    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(value As String)
            m_ProductCode = value
        End Set
    End Property

    Public Property TechID() As Nullable(Of Integer)
        Get
            If m_TechID.HasValue Then
                Return CInt(m_TechID)
            Else
                Return Nothing
            End If
```

```
    End Get
    Set(value As Nullable(Of Integer))
        m_TechID = value
    End Set
End Property

Public Property DateOpened() As Date
    Get
        Return m_DateOpened
    End Get
    Set(value As Date)
        m_DateOpened = value
    End Set
End Property

Public Property DateClosed() As Nullable(Of Date)
    Get
        If m_DateClosed.HasValue Then
            Return CDate(m_DateClosed)
        Else
            Return Nothing
        End If
    End Get
    Set(value As Nullable(Of Date))
        m_DateClosed = value
    End Set
End Property

Public Property Title() As String
    Get
        Return m_Title
    End Get
    Set(value As String)
        m_Title = value
    End Set
End Property

Public Property Description() As String
    Get
        Return m_Description
    End Get
    Set(value As String)
        m_Description = value
    End Set
End Property

Public ReadOnly Property CustomerName() As String
    Get
        Dim name As String = ""

        If m_CustomerID <> 0 Then
            Try
                name = CustomerDB.GetCustomerName(m_CustomerID)
            Catch ex As Exception
                Throw ex
            End Try
        End If
        Return name
```

```
        End Get
    End Property
    Public ReadOnly Property TechName() As String
        Get
            Dim name As String = ""

            If m_TechID.HasValue And m_TechID <> 0 Then
                Try
                    name = TechnicianDB.GetTechnicianName(CInt(m_TechID))
                Catch ex As Exception
                    Throw ex
                End Try
            End If
            Return name
        End Get
    End Property

    Public ReadOnly Property ProductName() As String
        Get
            Dim name As String = ""

            If m_ProductCode <> "" Then
                Try
                    name = ProductDB.GetProductName(m_ProductCode)
                Catch ex As Exception
                    Throw ex
                End Try
            End If
            Return name
        End Get
    End Property
End Class
```

Code of Customer Class (for partial 3E)

```
Public Class Customer
    Private m_CustomerID As Integer
    Private m_Name As String
    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 Name() As String
        Get
            Return m_Name
        End Get
        Set(ByVal value As String)
            m_Name = value
        End Set
    End Property
End Class
```

Code of Product Class (for partial 3E)

```
Public Class Product
    Private m_ProductCode As String
    Private m_Name As String
    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 Name() As String
        Get
            Return m_Name
        End Get
        Set(ByVal value As String)
            m_Name = value
        End Set
    End Property
End Class
```

Code of Registration Class (for 3E)

```
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 RegistrationDate() As Date
        Get
            Return m_RegistrationDate
```

```
        End Get
        Set(ByVal value As Date)
            m_RegistrationDate = value
        End Set
    End Property
End Class
```

Code of TechnicianDB Class

```
Imports System.Data.OleDb
Public Class TechnicianDB
    Public Shared Function GetTechnicianName(ByVal p_TechID As Integer) As String
        Dim name As String
        Dim conn As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery As String = "SELECT Name FROM Technicians WHERE TechID = " &
p_TechID
        Dim selectCmd As New OleDbCommand(selectQuery, conn)

        Try
            conn.Open()
            name = selectCmd.ExecuteScalar.ToString
        Catch ex As Exception
            Throw ex
        Finally
            conn.Close()
        End Try
        Return name
    End Function

    Public Shared Function GetTechnicianList() As List(Of Technician)
        Dim technicianList As New List(Of Technician)
        Dim conn As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery As String = "SELECT TechID, Name From Technicians Order By Name"
        Dim selectCmd As New OleDbCommand(selectQuery, conn)

        Try
            conn.Open()
            Dim reader As OleDbDataReader = selectCmd.ExecuteReader
            Dim technician As Technician

            Do While reader.Read
                technician = New Technician
                technician.TechID = CInt(reader("TechID"))
                technician.Name = reader("Name").ToString
                technicianList.Add(technician)
            Loop
            reader.Close()
        Catch ex As Exception
            Throw ex
        Finally
            conn.Close()
        End Try
        Return technicianList
    End Function

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

```
        Dim technician As Technician = Nothing
        Dim conn As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery As String = "SELECT TechID, Name, Email, Phone FROM Technicians
WHERE TechID = @TechID"
        Dim selectCmd As New OleDbCommand(selectQuery, conn)
        selectCmd.Parameters.AddWithValue("@TechID", techID)
        Try
            conn.Open()
            Dim reader As OleDbDataReader = selectCmd.ExecuteReader

            If reader.Read Then
                Technician = New Technician
                Technician.TechID = CInt(reader("TechID"))
                technician.Name = reader("Name").ToString
                technician.Email = reader("Email").ToString
                technician.Phone = reader("Phone").ToString
            End If
            reader.Close()
        Catch ex As Exception
            Throw ex
        Finally
            conn.Close()
        End Try

        Return technician
    End Function
End Class
```

Code of IncidentDB Class

```
Imports System.Data.OleDb
Public Class IncidentDB
    Public Shared Function GetOpenIncidents() As List(Of Incident)
        Dim incidentList As New List(Of Incident)
        Dim connection As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery = "SELECT CustomerID, ProductCode, TechID, DateOpened, Title " &
                          "FROM Incidents " &
                          "WHERE DateClosed IS NULL"
        Dim selectCmd As New OleDbCommand(selectQuery, connection)

        Try
            connection.Open()
            Dim reader As OleDbDataReader = selectCmd.ExecuteReader
            Dim incident As Incident

            Do While reader.Read
                incident = New Incident
                incident.CustomerID = CInt(reader("CustomerID"))
                incident.ProductCode = reader("ProductCode").ToString

                If IsDBNull(reader("TechID")) Then
                    incident.TechID = Nothing
                Else
                    incident.TechID = CInt(reader("TechID"))
```

```
                End If
                incident.DateOpened = CDate(reader("DateOpened"))
                incident.Title = reader("Title").ToString
                incidentList.Add(incident)
            Loop
            reader.Close()
        Catch ex As Exception
            Throw ex
        Finally
            connection.Close()
        End Try
        Return incidentList
    End Function

    Public Shared Sub AddIncident(ByVal p_Incident As Incident)
        Dim connection As OleDbConnection = TechSupportDB.GetConnection

        Dim insertQuery = "INSERT INTO Incidents " &
                          "(CustomerID, ProductCode, DateOpened, Title, Description) "
&
                          "VALUES (@CustomerID, @ProductCode, @DateOpened, @Title,
@Description)"

        Dim insertCmd As New OleDbCommand(insertQuery, connection)

        Try
            insertCmd.Parameters.AddWithValue("@CustomerID", p_Incident.CustomerID)
            insertCmd.Parameters.AddWithValue("@ProductCode", p_Incident.ProductCode)
            insertCmd.Parameters.AddWithValue("@DateOpened", CDate(DateTime.Today))
            insertCmd.Parameters.AddWithValue("@Title", p_Incident.Title)
            insertCmd.Parameters.AddWithValue("@Description", p_Incident.Description)
            connection.Open()
            insertCmd.ExecuteNonQuery()
        Catch ex As Exception
            Throw ex
        Finally
            connection.Close()
        End Try
    End Sub

    Public Shared Function GetIncident(ByVal p_IncidentID As Integer) As Incident
        Dim incident As Incident = Nothing
        Dim connection As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery = "SELECT IncidentID, CustomerID, ProductCode, TechID, " &
                          "DateOpened, DateClosed, Title, Description " &
                          "From Incidents Where IncidentID = @IncidentID"
        Dim selectCmd As New OleDbCommand(selectQuery, connection)
        selectCmd.Parameters.AddWithValue("@IncidentID", p_IncidentID)

        Try
            connection.Open()
            Dim reader As OleDbDataReader = selectCmd.ExecuteReader

            If reader.Read Then
                incident = New Incident
                incident.IncidentID = CInt(reader("IncidentID"))
                incident.CustomerID = CInt(reader("CustomerID"))
```

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

            If IsDBNull(reader("TechID")) Then
                incident.TechID = Nothing
            Else
                incident.TechID = CInt(reader("TechID"))
            End If
            incident.DateOpened = CDate(reader("DateOpened"))
            If IsDBNull(reader("DateClosed")) Then
                incident.DateClosed = Nothing
            Else
                incident.DateClosed = CDate(reader("DateClosed"))
            End If
            incident.Title = reader("Title").ToString
            incident.Description = reader("Description").ToString
        End If

        reader.Close()
    Catch ex As Exception
        Throw ex
    Finally
        connection.Close()
    End Try
    Return incident
End Function

Public Shared Function UpdateIncident(ByVal p_Incident As Incident,
                                      ByVal p_Description As String) As Boolean

    Dim isUpdated As Boolean = False
    Dim connection As OleDbConnection = TechSupportDB.GetConnection

    Dim insertQuery = "UPDATE Incidents SET Description = @NewDescription " &
                      "WHERE IncidentID = @IncidentID " &
                      "And Description = @Description " &
                      "And DateClosed Is NULL"


    Dim insertCmd As New OleDbCommand(insertQuery, connection)

    Try
        insertCmd.Parameters.AddWithValue("@NewDescription", p_Description)
        insertCmd.Parameters.AddWithValue("@IncidentID", p_Incident.IncidentID)
        insertCmd.Parameters.AddWithValue("@Description", p_Incident.Description)

        connection.Open()
        If insertCmd.ExecuteNonQuery() > 0 Then
            isUpdated = True
        End If
    Catch ex As Exception
        Throw ex
    Finally
        connection.Close()
    End Try
    Return isUpdated
End Function

Public Shared Function CloseIncident(ByVal p_Incident As Incident) As Boolean
```

```
        Dim isUpdated As Boolean = False
        Dim connection As OleDbConnection = TechSupportDB.GetConnection

        Dim insertQuery = "UPDATE Incidents SET DateClosed = @DateClosed " &
                          "WHERE IncidentID = @IncidentID " &
                          "And Description = @Description " &
                          "And DateClosed Is NULL"


        Dim insertCmd As New OleDbCommand(insertQuery, connection)

        Try
            insertCmd.Parameters.AddWithValue("@DateClosed", CDate(DateTime.Today))
            insertCmd.Parameters.AddWithValue("@IncidentID", p_Incident.IncidentID)
            insertCmd.Parameters.AddWithValue("@Description", p_Incident.Description)

            connection.Open()
            If insertCmd.ExecuteNonQuery() > 0 Then
                isUpdated = True
            End If
        Catch ex As Exception
            Throw ex
        Finally
            connection.Close()
        End Try
        Return isUpdated
    End Function

    Public Shared Function GetOpenTechnicianIncidents(ByVal techID As Integer) As List(Of
Incident)
        Dim incidentList As New List(Of Incident)
        Dim connection As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery = "SELECT CustomerID, ProductCode, TechID, DateOpened, Title " &
                          "FROM Incidents " &
                          "WHERE TechID = @TechID AND DateClosed IS NULL"
        Dim selectCmd As New OleDbCommand(selectQuery, connection)
        selectCmd.Parameters.AddWithValue("@TechID", techID)

        Try
            connection.Open()
            Dim reader As OleDbDataReader = selectCmd.ExecuteReader
            Dim incident As Incident

            Do While reader.Read
                incident = New Incident
                incident.CustomerID = CInt(reader("CustomerID"))
                incident.ProductCode = reader("ProductCode").ToString
                incident.TechID = CInt(reader("TechID"))
                incident.DateOpened = CDate(reader("DateOpened"))
                incident.Title = reader("Title").ToString
                incidentList.Add(incident)
            Loop
            reader.Close()
        Catch ex As Exception
            Throw ex
        Finally
```

```
            connection.Close()
        End Try
        Return incidentList
    End Function
End Class
```

Code of CustomerDB Class (for partial 3E)

```
Imports System.Data.OleDb
Public Class CustomerDB
    Public Shared Function GetCustomerName(ByVal p_CustomerID As Integer) As String
        Dim name As String
        Dim conn As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery As String = "SELECT Name FROM Customers WHERE CustomerID = " &
p_CustomerID
        Dim selectCmd As New OleDbCommand(selectQuery, conn)

        Try
            conn.Open()
            name = selectCmd.ExecuteScalar.ToString
        Catch ex As Exception
            Throw ex
        Finally
            conn.Close()
        End Try
        Return name
    End Function

    Public Shared Function GetCustomerList() As List(Of Customer)
        Dim customerList As New List(Of Customer)
        Dim conn As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery As String = "SELECT CustomerID, Name FROM Customers ORDER BY
Name"
        Dim selectCmd As New OleDbCommand(selectQuery, conn)

        Try
            conn.Open()
            Dim reader As OleDbDataReader = selectCmd.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 Exception
            Throw ex
        Finally
            conn.Close()
        End Try
```

```
        Return customerList
    End Function
End Class
```

Code of ProductDB Class (for partial 3E)

```
Imports System.Data.OleDb

Public Class ProductDB
    Public Shared Function GetProductName(ByVal p_ProductCode As String) As String
        Dim name As String
        Dim conn As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery As String = "SELECT Name FROM Products WHERE ProductCode =
@ProductCode"
        Dim selectCmd As New OleDbCommand(selectQuery, conn)
        selectCmd.Parameters.AddWithValue("@ProductCode", p_ProductCode)

        Try
            conn.Open()
            name = selectCmd.ExecuteScalar.ToString
        Catch ex As Exception
            Throw ex
        Finally
            conn.Close()
        End Try
        Return name
    End Function

    Public Shared Function GetProductList() As List(Of Product)
        Dim productList As New List(Of Product)
        Dim conn As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery As String = "SELECT ProductCode, Name FROM Products ORDER BY
Name"
        Dim selectCmd As New OleDbCommand(selectQuery, conn)

        Try
            conn.Open()
            Dim reader As OleDbDataReader = selectCmd.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 Exception
```

```
            Throw ex
        Finally
            conn.Close()
        End Try
        Return productList
    End Function
End Class
```

Code of RegistrationDB Class

```
Imports System.Data.OleDb

Public Class RegistrationDB

    Public Shared Function ProductRegistered(ByVal p_CustomerID As Integer,
                      ByVal p_ProductCode As String) As Boolean

        Dim isRegistered As Boolean = False
        Dim connection As OleDbConnection = TechSupportDB.GetConnection

        Dim selectQuery As String = "SELECT Count(*) FROM Registrations " &
            "WHERE CustomerID = @CustomerID AND ProductCode = @ProductCode"

        Dim selectCmd As New OleDbCommand(selectQuery, connection)

        Try
            selectCmd.Parameters.AddWithValue("@CustomerID", p_CustomerID)
            selectCmd.Parameters.AddWithValue("@ProductCode", p_ProductCode)
            connection.Open()

            Dim count As Integer = selectCmd.ExecuteScalar

            If count > 0 Then
                isRegistered = True
            End If

        Catch ex As Exception
            Throw ex
        Finally
            connection.Close()
        End Try

        Return isRegistered
    End Function

    Public Shared Sub AddRegistration(ByVal registration As Registration)
        Dim connection As OleDbConnection = TechSupportDB.GetConnection

        Dim insertQuery = "INSERT INTO Registrations " &
                          "(CustomerID, ProductCode, RegistrationDate) " &
                          "VALUES (@CustomerID, @ProductCode, @RegistrationDate)"

        Dim insertCmd As New OleDbCommand(insertQuery, connection)

        Try
```

```
            insertCmd.Parameters.AddWithValue("@CustomerID", registration.CustomerID)
            insertCmd.Parameters.AddWithValue("@ProductCode", registration.ProductCode)
            insertCmd.Parameters.AddWithValue("@RegistrationDate",
registration.RegistrationDate)
            connection.Open()
            insertCmd.ExecuteNonQuery()
        Catch ex As Exception
            Throw ex
        Finally
            connection.Close()
        End Try
    End Sub
End Class
```

Classes of SportsPro project

Code of frmMaintainRegistrations_YK Class

```
Imports TechSupportData
Public Class frmMaintainRegistrations_YK
    Private Sub frmMaintainRegistrations_YK_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        Try
            Dim customerList As List(Of Customer)
            Dim productList As List(Of Product)

            customerList = CustomerDB.GetCustomerList
            customerComboBox.DataSource = customerList
            customerComboBox.DisplayMember = "Name"
            customerComboBox.ValueMember = "CustomerID"

            productList = ProductDB.GetProductList
            productComboBox.DataSource = productList
            productComboBox.DisplayMember = "Name"
            productComboBox.ValueMember = "ProductCode"

            ResetControls()

        Catch ex As Exception
            MessageBox.Show(ex.Message, ex.GetType.ToString)
        End Try
    End Sub

    Private Sub registerButton_Click(sender As Object, e As EventArgs) Handles
registerButton.Click
        Try
            If Validator.IsSelected(customerComboBox, "Customer") AndAlso
               Validator.IsSelected(productComboBox, "Product") Then

                Dim custID As Integer = customerComboBox.SelectedValue
                Dim productCode As String = productComboBox.SelectedValue

                If Validator.HasRegistered(custID, productCode) Then
                    MessageBox.Show("Customer has already registered with selected
product.",
```

```
                                "Duplicate Registration!")
                Else
                    Dim regn As New Registration
                    regn.CustomerID = custID
                    regn.ProductCode = productCode
                    regn.RegistrationDate = Date.Today

                    RegistrationDB.AddRegistration(regn)

                    MessageBox.Show("The registration has been created.", "Regn Success")
                End If

                ResetControls()
            End If
        Catch ex As Exception
            MessageBox.Show(ex.Message, ex.GetType.ToString)
        End Try
    End Sub

    Private Sub exitButton_Click(sender As Object, e As EventArgs) Handles
exitButton.Click
        Me.Close()
    End Sub

    Private Sub ResetControls()
        customerComboBox.SelectedIndex = -1
        productComboBox.SelectedIndex = -1
    End Sub
End Class
```

Code of Validator Class

```
Imports TechSupportData
Public Class Validator
    Public Shared Function IsPresent(ByVal textBox As TextBox, ByVal name As String) As
Boolean

        If String.IsNullOrEmpty(textBox.Text) Then
            MessageBox.Show(name & " can't be empty.", "Input Required")
            textBox.Focus()
            Return False
        Else
            Return True
        End If
    End Function

    Public Shared Function IsSelected(ByVal comboBox As ComboBox, ByVal name As String)
As Boolean

        If comboBox.SelectedIndex < 0 Then
            MessageBox.Show($"Please select a {name}.", "Selection Required")
            comboBox.Focus()
            Return False
        Else
            Return True
        End If
```

```
    End Function

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

        If Integer.TryParse(textBox.Text, id) Then
            Return True
        Else
            MessageBox.Show(name & " must be an integer.", "Invalid Input")
            textBox.Focus()
            textBox.SelectAll()
            Return False
        End If

        Return True
    End Function

    Public Shared Function HasRegistered(ByVal customerID As Integer,
                                         ByVal productCode As String)
        Return RegistrationDB.ProductRegistered(customerID, productCode)
    End Function
End Class
```

Added code of frmMain YK Class only for Project 3-A

```
Private Sub MaintainRegistrations3EToolStripMenuItem_Click(sender As Object, e As
EventArgs) Handles MaintainRegistrations3EToolStripMenuItem.Click
        Dim frmMaintainRegn As New frmMaintainRegistrations_YK

        frmMaintainRegn.MdiParent = Me
        frmMaintainRegn.Show()
End Sub
```