SE Lab 2 contents

I. Introduction to Connecting Windows Forms (C#) to MSSQL Using Entity Framework

Connecting a Windows Forms application to a Microsoft SQL Server (MSSQL) database is a common requirement for many desktop applications. Entity Framework (EF) is an Object-Relational Mapper (ORM) that enables .NET developers to work with a database using .NET objects, eliminating the need for most of the data-access code that developers usually need to write.

In this introduction, we will cover the basics of connecting a Windows Forms application to an MSSQL database using Entity Framework. We will use the Entity Framework wizard to generate the necessary classes and context, and we will also discuss how to configure the connection string.

1.1. Steps to Connect Windows Forms to MSSQL Using Entity Framework

1.1.1. Set Up Your MSSQL Database:

- Ensure you have an MSSQL database set up and accessible. You can use SQL Server Management Studio (SSMS) to create and manage your database.
- Create a sample database and table for this exercise.

1.1.2. Create a New Windows Forms Project:

- Open Visual Studio and create a new Windows Forms App (.NET Framework) project.
- Name your project and choose a location to save it.

1.1.3. Install Entity Framework:

- Right-click on your project in the Solution Explorer and select Manage NuGet Packages.
- Search for EntityFramework and install it.

1.1.4. Use the Entity Framework Wizard:

- Right-click on your project in the Solution Explorer, select Add > New Item.
- Choose Data from the left menu and select ADO.NET Entity Data Model. Name it and click Add.
- In the Entity Data Model Wizard, select EF Designer from database and click Next.
- Choose your data connection or create a new connection to your MSSQL database. This will automatically generate the connection string in your App.config file.
- Select the database objects (tables, views, stored procedures) you want to include in your model and click Finish.

1.1.5. Configure the Connection String:

- The connection string is automatically added to your App.config file when you create the Entity Data Model. It looks something like this:
- <connectionStrings>
- <add name="YourEntities" connectionString="metadata=res://*/YourModel.csdl|res://*/YourModel.ssdl|res://*/YourModel.msl;provider=System.Data.SqlClient;provider connection string="data source=YourServer;initial catalog=YourDatabase;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"" providerName="System.Data.EntityClient" />
- </connectionStrings>
- Ensure the connection string points to your MSSQL server and database.

1.1.6. Using the Generated Context and Classes:

- The Entity Framework wizard generates a context class and entity classes based on your database schema.
- You can use these classes to interact with your database. For example, to retrieve data from a table:

```
using (var context = new YourEntities())
{
  var data = context.YourTable.ToList();
  // Bind data to a control, e.g., DataGridView
  dataGridView1.DataSource = data;
}
```

1.1.7. Summary

By following these steps, you can easily connect your Windows Forms application to an MSSQL database using Entity Framework. The Entity Framework wizard simplifies the process by generating the necessary classes and context, while the connection string in the App.config file ensures your application can communicate with the database. This approach allows you to focus on building your application's functionality without worrying about the complexities of database access code.

1.2. Steps to Use ConfigurationManager to Read Connection String

1.2.1. Add Reference to System.Configuration:

- By default, the System.Configuration namespace is not included in your project. You need to add a reference to it.
- Right-click on your project in the Solution Explorer and select Add > Reference.

- In the Reference Manager, search for System.Configuration and check the box next to it. Click OK.

1.2.2. Define the Connection String in App.config:

- Open the App.config file in your project.
- Add the connection string within the <connectionStrings> section.

```
<configuration>
 <connectionStrings>
    <add name="MyDatabaseConnectionString" connectionString="data
  source=YourServer;initial catalog=YourDatabase;integrated security=True;"
  providerName="System.Data.SqlClient" />
   </connectionStrings>
</configuration>
```

1.2.3. Read the Connection String Using ConfigurationManager:

- In your code, use the ConfigurationManager class to read the connection string from the App.config file.
- Make sure to include the System.Configuration namespace at the top of your code file.

```
using System;
using System.Configuration;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace WindowsFormsApp
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
```

```
        {
            // Read the connection string from App.config
            string connectionString =
ConfigurationManager.ConnectionStrings["MyDatabaseConnectionString"].ConnectionString;

            // Use the connection string to connect to the database
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                try
                {
                    connection.Open();
                    MessageBox.Show("Connection successful!");
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Connection failed: " + ex.Message);
                }
            }
        }
    }
}
```

1.2.4. Explanation

1. **Adding Reference to System.Configuration:**
    - This step ensures that your project can use the ConfigurationManager class, which is part of the System.Configuration namespace.
2. **Defining the Connection String in App.config:**
    - The App.config file is used to store configuration settings for your application. The <connectionStrings> section is specifically for storing database connection strings.
    - The name attribute is a unique identifier for the connection string, and the connectionString attribute contains the actual connection details.
3. **Reading the Connection String Using ConfigurationManager:**

- The ConfigurationManager.ConnectionStrings property returns a collection of connection strings defined in the App.config file.
- You can access a specific connection string by its name and use it to create a database connection.
- In the example code, the connection string is used to create a SqlConnection object, which is then opened to test the connection.

By following these steps, you can easily read and use connection strings from the App.config file in your Windows Forms application. This approach centralizes your configuration settings, making your code cleaner and easier to maintain.

II. In Class Activity

1. Exercise 1: Login Window Form

the exercises for creating a login Windows Form, setting up a data table tblUsers with sample data, and checking the login form.

1.1. Step 1: Setting Up the Database and tblUsers Table

Objective: Create a database and a table to store user credentials.

Steps:

1. **Create a Database:**
   - Open SQL Server Management Studio (SSMS).
   - Create a new database named Lab2DB.
2. **Create tblUsers Table:**
   - In the Lab2DB database, create a table named tblUsers with the following columns:

```
CREATE TABLE tblUsers (
  UserID INT PRIMARY KEY IDENTITY(1,1),
  Username NVARCHAR(50) NOT NULL,
  Password NVARCHAR(50) NOT NULL
);
```

3. **Insert Sample Data:**
   - Insert sample user data into the tblUsers table:

```
INSERT INTO tblUsers (Username, Password) VALUES ('user1', 'password1');
INSERT INTO tblUsers (Username, Password) VALUES ('user2', 'password2');
```

1.2. Step 2: Creating the Login Windows Form

Objective: Design a login form with fields for username and password.

Steps:

1. **Create a New Windows Forms Project:**
    - Open Visual Studio and create a new Windows Forms App (.NET Framework) project.
    - Name your project **LoginApp**.
2. **Design the Login Form:**
    - Drag two **Label** controls, two **TextBox** controls, and a **Button** control from the Toolbox to the form.
    - Set the properties of the controls as follows:
        - Label1: Text = "lblUsername"
        - Label2: Text = "lblPassword"
        - TextBox1: Name = "txtUsername"
        - TextBox2: Name = "txtPassword", PasswordChar = '*'
        - Button1: Text = "Login", Name = "btnLogin"
        - Button2: Text = "Cancel", Name = "btnCancel"
3. **Add a Label for Copyright:**
    - Add a **Label** control to the form with the text: "Developed by [Your Full Name]".
    - Adjust the location of the label as needed.

1.3. Step 3: Connecting to the Database

Objective: Set up the connection to the database using ConfigurationManager.

Steps:

1. **Add Reference to System.Configuration:**
    - Right-click on your project in the Solution Explorer and select Add > Reference.
    - In the Reference Manager, search for System.Configuration and check the box next to it. Click OK.
2. **Define the Connection String in App.config:**
    - Open the **App.config** file in your project.
    - Add the connection string within the <connectionStrings> section:

```
<configuration>
 <connectionStrings>
  <add name="LoginDBConnectionString" connectionString="data
source=YourServer;initial catalog=LoginDB;integrated security=True;"
providerName="System.Data.SqlClient" />
 </connectionStrings>
</configuration>
```

1.4. Step 4: Implementing the Login Logic

Objective: Write the code to check the login credentials against the database.

Steps:

1. **Add the Login Logic:**
   - Double-click the Login button to create the click event handler.
   - Write the following code to check the credentials:

```
using System;
using System.Configuration;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace LoginApp
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }

    private void btnLogin_Click(object sender, EventArgs e)
    {
      string connectionString =
ConfigurationManager.ConnectionStrings["LoginDBConnectionString"].ConnectionString;
      using (SqlConnection connection = new SqlConnection(connectionString))
```

```
        {
            try
            {
                connection.Open();
                string query = "SELECT COUNT(1) FROM tblUsers WHERE Username=@Username
AND Password=@Password";
                SqlCommand cmd = new SqlCommand(query, connection);
                cmd.Parameters.AddWithValue("@Username", txtUsername.Text);
                cmd.Parameters.AddWithValue("@Password", txtPassword.Text);
                int count = Convert.ToInt32(cmd.ExecuteScalar());

                if (count == 1)
                {
                    MessageBox.Show("Login successful!");
                }
                else
                {
                    MessageBox.Show("Login failed. Please check your username and password.");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }
        }
    }
  }
}
```

1.5. Step 5: Testing the Login Form

Objective: Run the application and test the login functionality.

Steps:

1. **Run the Application:**
    - Press F5 or click the Start button to run the application.
    - Enter the sample username and password (e.g., user1 and password1) and click the Login button.
2. **Take a Screenshot:**
    - After successfully logging in, take a screenshot of the form showing the login success message.
    - Include the screenshot in your summary report.

2. Exercise 2: Creating a Windows Form with DataGridView for CRUD Operations

Objective: Create a Windows Form with a DataGridView to perform CRUD (Create, Read, Update, Delete) operations on a tblStudents table.

Students Windows Form

Steps:

2.1. Step 1: Setting Up the Database and tblStudents Table

1. **Create a Database:**
    - Open SQL Server Management Studio (SSMS).
    - Create a new database named SchoolDB.
2. **Create tblStudents Table:**
    - In the SchoolDB database, create a table named tblStudents with the following columns:

```
CREATE TABLE tblStudents (
  StudentID INT PRIMARY KEY IDENTITY(1,1),
  StudentName NVARCHAR(100) NOT NULL,
  DateOfBirth DATE NOT NULL,
  City NVARCHAR(50) NOT NULL,
  Age INT NOT NULL,
  YearOfEnroll INT NOT NULL,
  Major NVARCHAR(50),
  GPA DECIMAL(3, 2)
);
```

3. **Insert Sample Data:**

    - Insert sample student data into the tblStudents table:

    INSERT INTO tblStudents (StudentName, DateOfBirth, City, Age, YearOfEnroll, Major, GPA) VALUES

    ('Trung Pham', '2000-01-15', 'HCMC', 24, 2018, 'Computer Science', 3.5),

    ('Jane Smith', '1999-05-22', 'Los Angeles', 25, 2017, 'Mathematics', 3.8),

    ('Alice Johnson', '2001-03-10', 'Chicago', 23, 2019, 'Physics', 3.6),

    ('Bob Brown', '2002-07-30', 'Houston', 22, 2020, 'Chemistry', 3.7);

2.2. Step 2: Creating the Windows Forms Project

1. **Create a New Windows Forms Project:**

    - Open Visual Studio and create a new Windows Forms App (.NET Framework) project.
    - Name your project **StudentManagementApp.**

2. **Design the Form:**

    - Drag a **DataGridView** control from the Toolbox to the form.
    - Drag four **Button** controls from the Toolbox to the form and set their properties as follows:
        - Button1: Text = "Add", Name = "btnAdd"
        - Button2: Text = "Update", Name = "btnUpdate"
        - Button3: Text = "Delete", Name = "btnDelete"
        - Button4: Text = "Refresh", Name = "btnRefresh"

3. **Add a Label for Copyright:**

    - Add a **Label** control to the form with the text: "Developed by [Your Full Name]".
    - Adjust the location of the label as needed.

2.3. Step 3: Connecting to the Database

1. **Add Reference to System.Configuration:**

    - Right-click on your project in the Solution Explorer and select Add > Reference.
    - In the Reference Manager, search for System.Configuration and check the box next to it. Click OK.

2. **Define the Connection String in App.config:**

    - Open the App.config file in your project.

- Add the connection string within the <connectionStrings> section:

```
<configuration>
 <connectionStrings>
  <add name="SchoolDBConnectionString" connectionString="data source=YourServer;initial catalog=SchoolDB;integrated security=True;" providerName="System.Data.SqlClient" />
 </connectionStrings>
</configuration>
```

2.4. Step 4: Implementing CRUD Operations

1. **Add the CRUD Logic:**
    - Double-click the buttons to create the click event handlers.
    - Write the following code to implement the CRUD operations:

```
using System;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
namespace StudentManagementApp
{
  public partial class Form1 : Form
  {
    private string connectionString;

    public Form1()
    {
      InitializeComponent();
      connectionString = ConfigurationManager.ConnectionStrings["SchoolDBConnectionString"].ConnectionString;
    }

    private void Form1_Load(object sender, EventArgs e)
    {
```

```
    LoadData();
}

private void LoadData()
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM tblStudents", connection);
        DataTable table = new DataTable();
        adapter.Fill(table);
        dataGridView1.DataSource = table;
    }
}

private void btnAdd_Click(object sender, EventArgs e)
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
            SqlCommand cmd = new SqlCommand("INSERT INTO tblStudents (StudentName,
        DateOfBirth, City, Age, YearOfEnroll, Major, GPA) VALUES (@StudentName, @DateOfBirth,
        @City, @Age, @YearOfEnroll, @Major, @GPA)", connection);
        cmd.Parameters.AddWithValue("@StudentName", "New Student");
        cmd.Parameters.AddWithValue("@DateOfBirth", DateTime.Now);
        cmd.Parameters.AddWithValue("@City", "City");
        cmd.Parameters.AddWithValue("@Age", 20);
        cmd.Parameters.AddWithValue("@YearOfEnroll", 2021);
        cmd.Parameters.AddWithValue("@Major", "Major");
        cmd.Parameters.AddWithValue("@GPA", 4.0);
        cmd.ExecuteNonQuery();
        LoadData();
    }
```

```
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            if (dataGridView1.SelectedRows.Count > 0)
            {
                int studentID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["StudentID"].Value);
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    SqlCommand cmd = new SqlCommand("UPDATE tblStudents SET
    StudentName=@StudentName, DateOfBirth=@DateOfBirth, City=@City, Age=@Age,
    YearOfEnroll=@YearOfEnroll, Major=@Major, GPA=@GPA WHERE StudentID=@StudentID",
    connection);
                    cmd.Parameters.AddWithValue("@StudentID", studentID);
                    cmd.Parameters.AddWithValue("@StudentName", "Updated Student");
                    cmd.Parameters.AddWithValue("@DateOfBirth", DateTime.Now);
                    cmd.Parameters.AddWithValue("@City", "Updated City");
                    cmd.Parameters.AddWithValue("@Age", 21);
                    cmd.Parameters.AddWithValue("@YearOfEnroll", 2022);
                    cmd.Parameters.AddWithValue("@Major", "Updated Major");
                    cmd.Parameters.AddWithValue("@GPA", 3.9);
                    cmd.ExecuteNonQuery();
                    LoadData();
                }
            }
    }

    private void btnDelete_Click(object sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count > 0)
        {
```

```
            int studentID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["StudentID"].Value);
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                SqlCommand cmd = new SqlCommand("DELETE FROM tblStudents WHERE
StudentID=@StudentID", connection);
                cmd.Parameters.AddWithValue("@StudentID", studentID);
                cmd.ExecuteNonQuery();
                LoadData();
            }
        }
    }

    private void btnRefresh_Click(object sender, EventArgs e)
    {
        LoadData();
    }
  }
}
```

2.5. Step 5: Testing the Users Form

Objective: Run the application and test the login functionality.

Steps:

1. **Run the Application:**
    - Press F5 or click the Start button to run the application.
    - Enter the sample username and password, test CRUD.
2. **Take a Screenshot:**
    - After successfully logging in, take a screenshot of the form showing the login success message.
    - Include the screenshot in your summary report.

III. HomeWorks

Implementing a Student Management Windows Form Application

Objective: Create a Windows Form Application with a login form that navigates to a main form containing a MenuStrip for navigating to different forms with CRUD.

Step 1: Setting Up the Database

1. **Create the Database and Tables:**

    - Follow the SQL scripts provided earlier to create the SchoolDB database and the tables: tblStudents, tblCourses, tblStudentCourses, tblInstructors, and tblDepartments.

i. tblCourses:

- This table will store information about courses.
- Relationship: Many-to-Many with tblStudents through tblStudentCourses.

```
CREATE TABLE tblCourses (
    CourseID INT PRIMARY KEY IDENTITY(1,1),
    CourseName NVARCHAR(100) NOT NULL,
    Credits INT NOT NULL,
    DepartmentID INT NOT NULL,
    InstructorID INT NOT NULL
);

INSERT INTO tblCourses (CourseName, Credits, DepartmentID,
InstructorID) VALUES
('Introduction to Computer Science', 3, 1, 1),
('Calculus I', 4, 2, 2),
('Physics I', 4, 3, 3),
('Organic Chemistry', 4, 4, 4);
```

ii. tblStudentCourses:

- This table will store the relationship between students and courses.
- Relationship: Many-to-Many between tblStudents and tblCourses.

```
CREATE TABLE tblStudentCourses (
    StudentID INT NOT NULL,
    CourseID INT NOT NULL,
    PRIMARY KEY (StudentID, CourseID),
    FOREIGN KEY (StudentID) REFERENCES tblStudents(StudentID),
    FOREIGN KEY (CourseID) REFERENCES tblCourses(CourseID)
);

INSERT INTO tblStudentCourses (StudentID, CourseID) VALUES
(1, 1),
```

```
(1, 2),
(2, 3),
(3, 4),
(4, 1),
(4, 3);
```

iii. tblInstructors:

- This table will store information about instructors.
- Relationship: One-to-Many with tblCourses.

```
CREATE TABLE tblInstructors (
    InstructorID INT PRIMARY KEY IDENTITY(1,1),
    InstructorName NVARCHAR(100) NOT NULL,
    DepartmentID INT NOT NULL
);

INSERT INTO tblInstructors (InstructorName, DepartmentID) VALUES
('Dr. John Smith', 1),
('Dr. Jane Doe', 2),
('Dr. Alice Brown', 3),
('Dr. Bob White', 4);
```

iv. tblDepartments:

- This table will store information about departments.
- Relationship: One-to-Many with tblCourses.

```
CREATE TABLE tblDepartments (
    DepartmentID INT PRIMARY KEY IDENTITY(1,1),
    DepartmentName NVARCHAR(100) NOT NULL
);

INSERT INTO tblDepartments (DepartmentName) VALUES
('Computer Science'),
('Mathematics'),
('Physics'),
('Chemistry');
```

Relationships

- **tblCourses** to **tblDepartments**: One-to-Many relationship (One department can offer many courses).
- **tblCourses** to **tblInstructors**: One-to-Many relationship (One instructor can teach many courses).

- **tblStudentCourses** to **tblStudents**: Many-to-One relationship (Many student-course enrollments can belong to one student).
- **tblStudentCourses** to **tblCourses**: Many-to-One relationship (Many student-course enrollments can belong to one course).

```
-- Establish relationships
ALTER TABLE tblCourses
ADD CONSTRAINT FK_Courses_Departments
FOREIGN KEY (DepartmentID) REFERENCES tblDepartments(DepartmentID);

ALTER TABLE tblCourses
ADD CONSTRAINT FK_Courses_Instructors
FOREIGN KEY (InstructorID) REFERENCES tblInstructors(InstructorID);


ALTER TABLE tblStudentCourses
ADD CONSTRAINT FK_StudentCourses_Courses
FOREIGN KEY (CourseID) REFERENCES tblCourses(CourseID);
```

Step 2: Creating the Windows Forms Project

1. **Create a New Windows Forms Project:**
    - Open Visual Studio and create a new Windows Forms App (.NET Framework) project.
    - Name your project StudentManagementApp.
2. **Add Reference to System.Configuration:**
    - Right-click on your project in the Solution Explorer and select Add > Reference.
    - In the Reference Manager, search for System.Configuration and check the box next to it. Click OK.
3. **Define the Connection String in App.config:**
    - Open the App.config file in your project.
    - Add the connection string within the <connectionStrings> section:
    - <configuration>
    - <connectionStrings>
    - <add name="SchoolDBConnectionString" connectionString="data source=YourServer;initial catalog=SchoolDB;integrated security=True;" providerName="System.Data.SqlClient" />
    - </connectionStrings>
    - </configuration>

Step 3: Implementing the Login Logic

1. **Design the Login Form:**
    - Add two **Label** controls, two **TextBox** controls, and a **Button** control to the form.
    - Set the properties of the controls as follows:
        - Label1: Text = "Username"
        - Label2: Text = "Password"
        - TextBox1: Name = "txtUsername"
        - TextBox2: Name = "txtPassword", PasswordChar = '*'
        - Button1: Text = "Login", Name = "btnLogin"
        - Button1: Text = "Cancel", Name = "btnCancel"
2. **Add a Label for Copyright:**
    - Add a **Label** control to the form with the text: "Developed by [Your Full Name]".
    - Adjust the location of the label as needed.
3. **Implement the Login Logic:**
    - Double-click the Login button to create the click event handler.
    - Write the following code to check the credentials:

```
using System;
using System.Configuration;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace StudentManagementApp
{
  public partial class LoginForm : Form
  {
    public LoginForm()
    {
      InitializeComponent();
    }
```

```
    private void btnLogin_Click(object sender, EventArgs e)
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["SchoolDBConnectionString"].ConnectionString;
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            try
            {
                connection.Open();
                string query = "SELECT COUNT(1) FROM tblUsers WHERE Username=@Username
AND Password=@Password";
                SqlCommand cmd = new SqlCommand(query, connection);
                cmd.Parameters.AddWithValue("@Username", txtUsername.Text);
                cmd.Parameters.AddWithValue("@Password", txtPassword.Text);
                int count = Convert.ToInt32(cmd.ExecuteScalar());

                if (count == 1)
                {
                    this.Hide();
                    MainForm mainForm = new MainForm();
                    mainForm.Show();
                }
                else
                {
                    MessageBox.Show("Login failed. Please check your username and password.");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }
        }
```

```
    }
  }
}
```

Step 4: Designing the Main Form

1. **Create the Main Form:**
    - Add a new Windows Form to the project and name it MainForm.
2. **Design the Main Form:**
    - Drag a **MenuStrip** control from the Toolbox to the form.
    - Add menu items for navigating to different forms:
        - MenuStrip: Items = "File", "Students", "Courses", "Instructors", "Departments"
        - Under "File", add "Exit"
        - Under "Students", add "View Students"
        - Under "Courses", add "View Courses"
        - Under "Instructors", add "View Instructors"
        - Under "Departments", add "View Departments"
3. **Add Event Handlers for Menu Items:**
    - Double-click each menu item to create the click event handlers.
    - Write the following code to handle navigation:

```
using System;
using System.Windows.Forms;

namespace StudentManagementApp
{
  public partial class MainForm : Form
  {
    public MainForm()
    {
      InitializeComponent();
    }
```

```
        private void viewStudentsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            StudentsForm studentsForm = new StudentsForm();
            studentsForm.Show();
        }

        private void viewCoursesToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CoursesForm coursesForm = new CoursesForm();
            coursesForm.Show();
        }

        private void viewInstructorsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            InstructorsForm instructorsForm = new InstructorsForm();
            instructorsForm.Show();
        }

        private void viewDepartmentsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            DepartmentsForm departmentsForm = new DepartmentsForm();
            departmentsForm.Show();
        }

        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Step 5: Creating Forms for Students, Courses, Instructors, and Departments

1. **Create Forms:**
    - Add new Windows Forms to the project and name them StudentsForm, CoursesForm, InstructorsForm, and DepartmentsForm.
2. **Design Each Form:**
    - Drag a **DataGridView** control from the Toolbox to each form.
    - Drag four **Button** controls from the Toolbox to each form and set their properties as follows:
        - Button1: Text = "Add", Name = "btnAdd"
        - Button2: Text = "Update", Name = "btnUpdate"
        - Button3: Text = "Delete", Name = "btnDelete"
        - Button4: Text = "Refresh", Name = "btnRefresh"
3. **Implement CRUD Operations for Each Form:**
    - Follow the CRUD implementation steps from the previous exercise for each form, adjusting the SQL queries and parameters as needed for each table (tblStudents, tblCourses, tblInstructors, tblDepartments).

Step 6: Designing the Sub-Form (Student Enrollments)

1. **Create the Sub-Form:**
    - Add a new Windows Form to the project and name it **StudentEnrollmentsForm**.
2. **Design the Sub-Form:**
    - Drag a **ComboBox** control from the Toolbox to the form for selecting courses.
    - Drag a **DataGridView** control from the Toolbox to the form for displaying student enrollments.
    - Drag four **Button** controls from the Toolbox to the form and set their properties as follows:
        - Button1: Text = "Add", Name = "btnAdd"
        - Button2: Text = "Update", Name = "btnUpdate"
        - Button3: Text = "Delete", Name = "btnDelete"
        - Button4: Text = "Refresh", Name = "btnRefresh"
3. **Implementing the Sub-Form Logic**

- **Create a Stored Procedure to Get Courses: In SQL Server Management Studio (SSMS), create a stored procedure to get the list of courses:**

```
CREATE PROCEDURE GetCourses
AS
BEGIN
    SELECT CourseID, CourseName FROM tblCourses;
END
```

- **Implement the Logic for Student Enrollments:**
    - **LoadCourses Method:**
        - This method loads the list of courses into the ComboBox using the stored procedure GetCourses.
        - The ComboBox displays the course names and uses the course IDs as values.
    - **LoadData Method:**
        - This method loads the student enrollments for the selected course into the DataGridView.
        - It uses the selected course ID from the ComboBox to filter the enrollments.
    - **btnAdd_Click Method:**
        - This method adds a new student enrollment for the selected course.
        - Replace the hardcoded student ID (1) with the actual student ID as needed.
    - **btnUpdate_Click Method:**
        - This method updates the course for the selected student enrollment.
        - It uses the selected row in the DataGridView to get the student ID and the selected course ID from the ComboBox.
    - **btnDelete_Click Method:**
        - This method deletes the selected student enrollment.
        - It uses the selected row in the DataGridView to get the student ID.
    - **btnRefresh_Click Method:**
        - This method refreshes the data in the DataGridView by calling the LoadData method.
    - **comboBoxCourses_SelectedIndexChanged Method:**
        - This method reloads the data in the DataGridView when the selected course in the ComboBox changes.
    -
    - Double-click the buttons to create the click event handlers.
- Write the following code to implement the logic for the `StudentEnrollmentsForm`:

```
using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;
```

```
using System.Windows.Forms;


namespace CourseManagementApp
{
    public partial class StudentEnrollmentsForm : Form
    {
        private string connectionString;
        public StudentEnrollmentsForm()
        {
            InitializeComponent();
            connectionString =
ConfigurationManager.ConnectionStrings["SchoolDBConnectionString"].ConnectionString;
        }


        private void StudentEnrollmentsForm_Load(object sender, EventArgs e)
        {
            LoadCourses();
            LoadData();
        }


        private void LoadCourses()
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                SqlCommand cmd = new SqlCommand("GetCourses", connection);
```

```
            cmd.CommandType = CommandType.StoredProcedure;
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataTable table = new DataTable();
            adapter.Fill(table);
            comboBoxCourses.DataSource = table;
            comboBoxCourses.DisplayMember = "CourseName";
            comboBoxCourses.ValueMember = "CourseID";
        }
    }

    private void LoadData()
    {
        if (comboBoxCourses.SelectedValue != null)
        {
            int courseID = Convert.ToInt32(comboBoxCourses.SelectedValue);
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
tblStudentCourses WHERE CourseID=@CourseID", connection);
                adapter.SelectCommand.Parameters.AddWithValue("@CourseID", courseID);
                DataTable table = new DataTable();
                adapter.Fill(table);
                dataGridView1.DataSource = table;
            }
        }
```

```
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (comboBoxCourses.SelectedValue != null)
            {
                int courseID = Convert.ToInt32(comboBoxCourses.SelectedValue);
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    SqlCommand cmd = new SqlCommand("INSERT INTO tblStudentCourses
(StudentID, CourseID) VALUES (@StudentID, @CourseID)", connection);
                    cmd.Parameters.AddWithValue("@StudentID", 1); // Replace with actual student
ID
                    cmd.Parameters.AddWithValue("@CourseID", courseID);
                    cmd.ExecuteNonQuery();
                    LoadData();
                }
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            if (dataGridView1.SelectedRows.Count > 0 && comboBoxCourses.SelectedValue !=
null)
            {
```

```
            int studentID =
Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["StudentID"].Value);

            int courseID = Convert.ToInt32(comboBoxCourses.SelectedValue);

            using (SqlConnection connection = new SqlConnection(connectionString))

            {

                connection.Open();

                SqlCommand cmd = new SqlCommand("UPDATE tblStudentCourses SET
CourseID=@CourseID WHERE StudentID=@StudentID", connection);

                cmd.Parameters.AddWithValue("@StudentID", studentID);

                cmd.Parameters.AddWithValue("@CourseID", courseID);

                cmd.ExecuteNonQuery();

                LoadData();

            }

        }

    }


    private void btnDelete_Click(object sender, EventArgs e)

    {

        if (dataGridView1.SelectedRows.Count > 0)

        {

            int studentID =
Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["StudentID"].Value);

            using (SqlConnection connection = new SqlConnection(connectionString))

            {

                connection.Open();
```

```
                SqlCommand cmd = new SqlCommand("DELETE FROM tblStudentCourses
WHERE StudentID=@StudentID", connection);

                cmd.Parameters.AddWithValue("@StudentID", studentID);

                cmd.ExecuteNonQuery();

                LoadData();

            }

        }

    }


    private void btnRefresh_Click(object sender, EventArgs e)

    {

        LoadData();

    }


    private void comboBoxCourses_SelectedIndexChanged(object sender, EventArgs e)

    {

        LoadData();

    }

  }

}
```

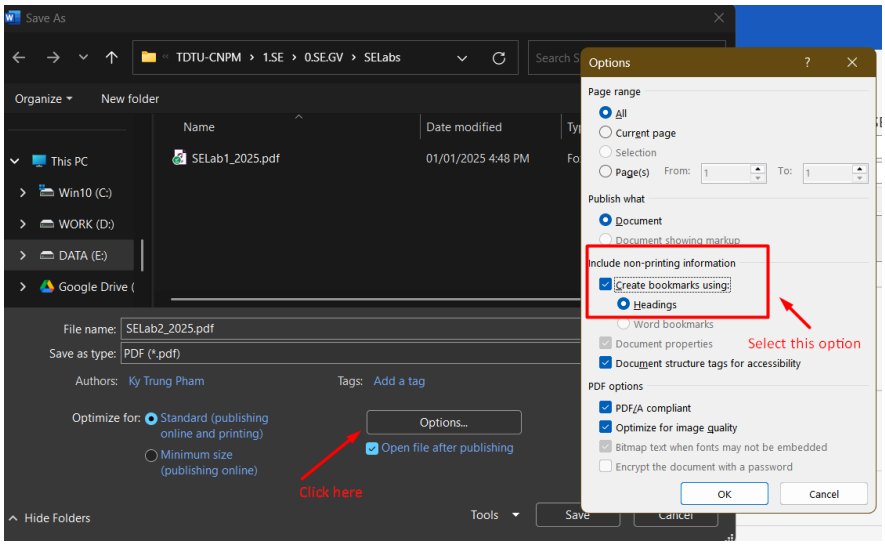
IV. Lab 2 Report Submission

- **Introduction:** Briefly describe the purpose of the exercises.
- **Exercises:** List each exercise with a brief description of what you did and learned.
- **Screenshots:** Include the screenshots of the output for each exercise.
- **Conclusion:** Summarize your overall learning experience and any challenges you faced.

By following these exercises, you will learn how to create a login form, connect to an MSSQL database, and validate user credentials using Entity Framework and ConfigurationManager.

Submit 2 files on eLearning:

- StudentID_StudentName.zip (contain Windows Form Project & .sql file)
- StudentID_StudentName.pdf (require to create bookmarks)



Notes: There may be a typo/mistake during creating this document. Please correct it by yourself.

Enjoy ☺

Email: tg_phamthaikytrung@tdtu.edu.vn