SE Lab 3 Contents

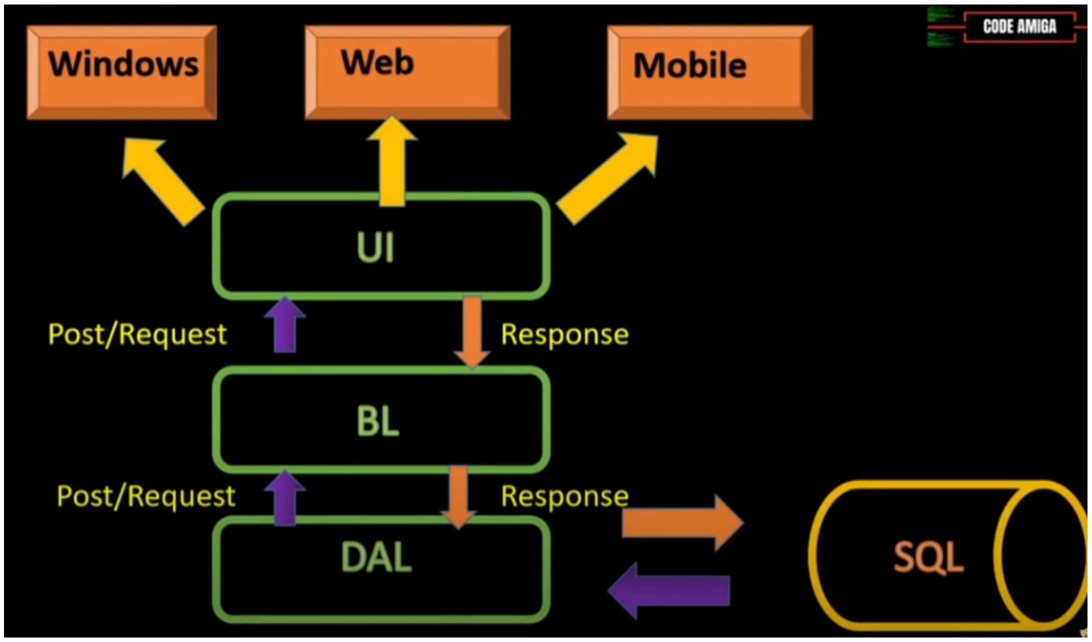


Image Source: https://www.youtube.com/watch?v=P4E9vHYsyAU

I. Creating a 3-Tier Architecture in C# .NET

Involves separating your application into three distinct layers: the Presentation Layer, the Business Logic Layer, and the Data Access Layer. Here's a step-by-step guide to help you get started:

1. Presentation Layer

This layer is responsible for the user interface and user interaction. **In a .NET application, this could be an ASP.NET MVC project or a Windows Forms/WPF application.**

- **Create a new project**: Start by creating a new ASP.NET MVC or Windows Forms/WPF project.
- **Design the UI**: Add views, controllers, or forms to handle user input and display data.

2. Business Logic Layer (BLL)

This layer contains the core functionality and business rules of your application.

- **Create a Class Library**: Add a new Class Library project to your solution for the Business Logic Layer.
- **Implement Business Logic**: Create classes and methods to handle the business rules and operations. For example, you might have a CustomerService class with methods like AddCustomer, GetCustomer, etc.

3. Data Access Layer (DAL)

This layer is responsible for interacting with the database.

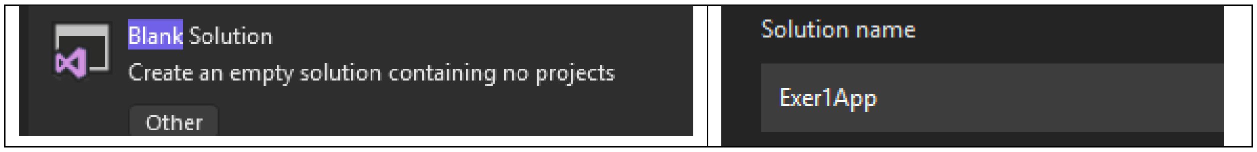
- **Create another Class Library**: Add a new Class Library project for the Data Access Layer.
- **Implement Data Access**: Use Entity Framework or ADO.NET to interact with the database. Create repository classes to handle CRUD operations. For example, you might have a CustomerRepository class with methods like AddCustomer, GetCustomerById, etc.

II. Class Activity

Exercise 1: Setting up 3-Tier Architecture with WinformUI

Step 1: Set Up the Solution

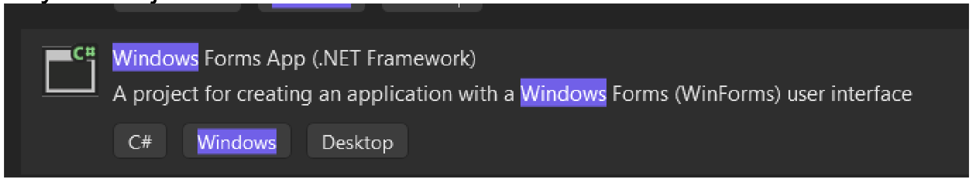
1. **Open Visual Studio** and create a new solution:
    - Go to File > New > Project.
    - Select Blank Solution and name it Exer1App.


Blank Solution
Create an empty solution containing no projects
Other

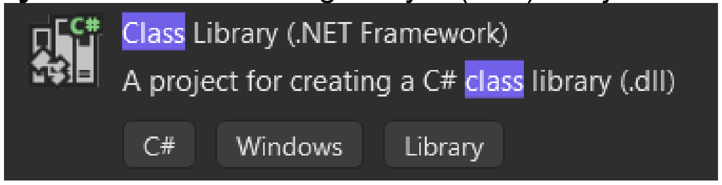
Solution name
Exer1App

2. **Add three projects** to the solution:

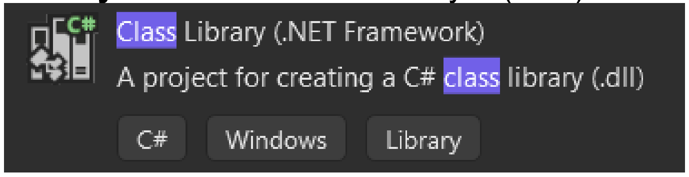
- **Windows Forms Application(.Net Framework)** for the Presentation Layer. Project name: WinFormUI


Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

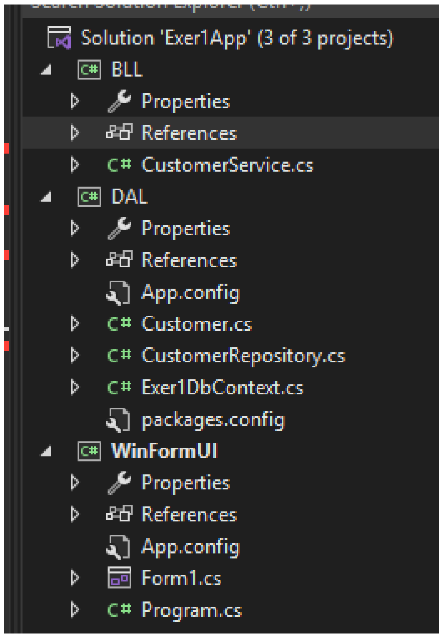
- **Class Library** for the Business Logic Layer (BLL). Project name: BLL


Class Library (.NET Framework)
A project for creating a C# class library (.dll)
C# Windows Library

- **Class Library** for the Data Access Layer (DAL).


Class Library (.NET Framework)
A project for creating a C# class library (.dll)
C# Windows Library

Solution Explorer



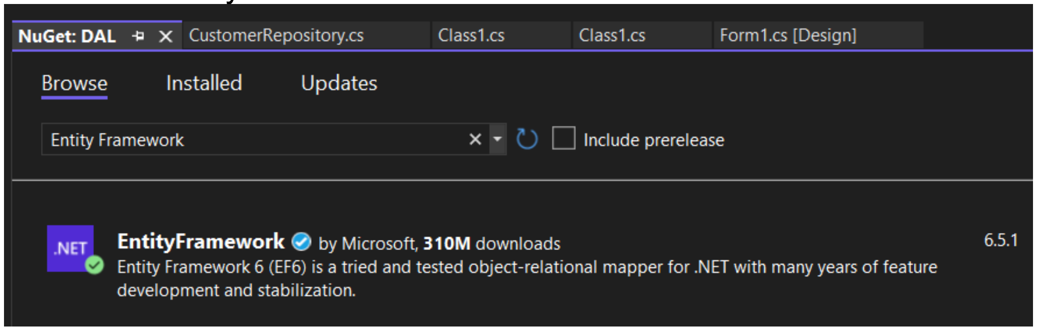
```
Solution 'Exer1App' (3 of 3 projects)
  BLL
    Properties
    References
    CustomerService.cs
  DAL
    Properties
    References
    App.config
    Customer.cs
    CustomerRepository.cs
    Exer1DbContext.cs
    packages.config
  WinFormUI
    Properties
    References
    App.config
    Form1.cs
    Program.cs
```

Step 2: Create the Data Access Layer (DAL)

1. **Add a new Class Library project** named DAL:

- Right-click on the solution in Solution Explorer.
- Select Add > New Project.
- Choose Class Library (.NET Framework) and name it DAL.

2. **Add a class** named CustomerRepository to handle database operations:
    - Right-click on the DAL project.
    - Select Add > Class and name it CustomerRepository.cs.
3. **Install Entity Framework** via NuGet Package Manager:
    - Right-click on the DAL project.
    - Select Manage NuGet Packages.
    - Search for EntityFramework and install it.



Entity Framework

EntityFramework by Microsoft, **310M** downloads
Entity Framework 6 (EF6) is a tried and tested object-relational mapper for .NET with many years of feature development and stabilization.

Example Code for CustomerRepository

// Customer.cs (Model)

```
public class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
}
```

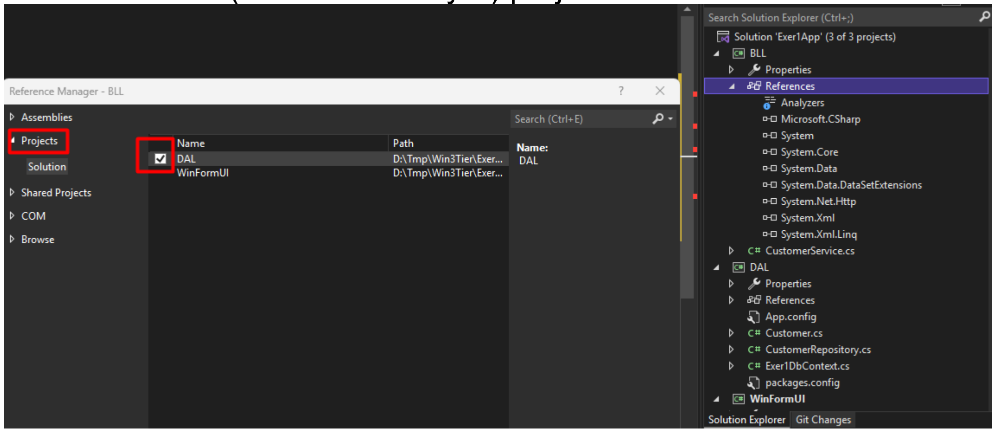
// CustomerRepository.cs

```
public class CustomerRepository
{
    private readonly Exer1DbContext _context;
    public CustomerRepository()
    {
        _context = new Exer1DbContext();
    }
    public void AddCustomer(Customer customer)
    {
        _context.Customers.Add(customer);
        _context.SaveChanges();
    }
    public Customer GetCustomerById(int id)
    {
        return _context.Customers.FirstOrDefault(c => c.CustomerID == id);
    }
    public List<Customer> GetAllCustomers()
    {
        return _context.Customers.ToList();
    }
    public void UpdateCustomer(Customer customer)
    {
        var existingCustomer = _context.Customers.FirstOrDefault(c => c.CustomerID ==
customer.CustomerID);
        if (existingCustomer != null)
        {
            existingCustomer.Name = customer.Name;
            existingCustomer.Email = customer.Email;
            _context.SaveChanges();
        }
    }
    public void DeleteCustomer(int id)
    {
        var customer = _context.Customers.FirstOrDefault(c => c.Id == id);
        if (customer != null)
        {
            _context.Customers.Remove(customer);
            _context.SaveChanges();
        }
    }
```

```
            }
```

Step 3: Create the Business Logic Layer (BLL)

1. **Add a new Class Library project** named BLL (BusinessLogicLayer):
    - Right-click on the solution in Solution Explorer.
    - Select Add > New Project.
    - Choose Class Library (.NET Framework) and name it BusinessLogicLayer.
2. **Add a class** named CustomerService to handle business logic:
    - Right-click on the BusinessLogicLayer project.
    - Select Add > Class and name it CustomerService.cs.
3. **Reference the Data Access Layer** project in the BLL(Business Logic Layer) project:
    - Right-click on the BusinessLogicLayer project.
    - Select Add > Reference.
    - Check the DAL (DataAccessLayer) project and click OK.



	Name	Path
✓	DAL	D:\Tmp\Win3Tier\Exer...
	WinFormUI	D:\Tmp\Win3Tier\Exer...

Name:
DAL

```
Solution 'Exer1App' (3 of 3 projects)
  BLL
    Properties
    References
      Analyzers
      Microsoft.CSharp
      System
      System.Core
      System.Data
      System.Data.DataSetExtensions
      System.Net.Http
      System.Xml
      System.Xml.Linq
    CustomerService.cs
  DAL
    Properties
    References
    App.config
    Customer.cs
    CustomerRepository.cs
    Exer1DbContext.cs
    packages.config
  WinFormUI
```

Right click on DAL and select Build
Example Code for CustomerService

```
// CustomerService.cs
        public class CustomerService
        {
            private readonly CustomerRepository _customerRepository;

            public CustomerService()
            {
```

```
            _customerRepository = new CustomerRepository();
        }

        public void AddCustomer(Customer customer)
        {
            // Business logic before adding customer
            _customerRepository.AddCustomer(customer);
        }

        public Customer GetCustomer(int id)
        {
            // Business logic before retrieving customer
            return _customerRepository.GetCustomerById(id);
        }
    public List<Customer> GetAllCustomers()
        {
            return _customerRepository.GetAllCustomers();
        }

        public void UpdateCustomer(Customer customer)
        {
            // Business logic before updating customer
            _customerRepository.UpdateCustomer(customer);
        }

        public void DeleteCustomer(int id)
        {
            // Business logic before deleting customer
            _customerRepository.DeleteCustomer(id);
        }
    }
```

Step 4: Create the WinformUI (Presentation Layer)

1. **Add a new Windows Forms Application project** named PresentationLayer:
    - Right-click on the solution in Solution Explorer.
    - Select Add > New Project.
    - Choose Windows Forms App (.NET Framework) and name it WindowFormUI (or PresentationLayer) .
2. **Reference the Business Logic Layer** project in the Presentation Layer project:
    - Right-click on the **WinformUI** (PresentationLayer) project.
    - Select Add > Reference.
    - Check the **BLL** (BusinessLogicLayer) project and click OK.
    - **Install Entity Framework** via NuGet Package Manager:
        - Right-click on the DAL project.
        - Select Manage NuGet Packages.
        - Search for EntityFramework and install it.
3. **Design the UI**: Add forms and controls to handle user input and display data:
    - Open Form1.cs and design the form with text boxes, labels, and buttons for adding and retrieving customers.
      (Add a new form or rename **Form1.cs** to **CustomerForm.cs**)
    - **Design the Form**
        1. **Open CustomerForm.cs** in the designer view.
        2. **Add controls** to the form:
            - **Labels**: For "Name", "Email", and "Customer ID".
            - **TextBoxes**: For entering "Name", "Email", and "Customer ID".
            - **Buttons**: For "Add Customer" and "Get Customer".
    - **Set Up the Controls**
        1. **Drag and drop** the controls from the Toolbox onto the form.
        2. **Set the properties** of the controls:
            - **Labels**: Set the Text property to "Name", "Email", and "Customer ID".
            - **TextBoxes**: Name them txtName, txtEmail, and txtCustomerId.

- **Buttons**: Name them btnAddCustomer and btnGetCustomer, and set the Text property to "Add Customer" and "Get Customer".
- May add datagridview to show all data from Customer table

- **Add Event Handlers**
  1. **Double-click** on the "Add Customer" button to create an event handler for the Click event.
  2. **Double-click** on the "Get Customer" button to create an event handler for the Click event.
  3. Double-click on the Customer Form to create CustomerForm_Load event

- **Implement the Code**

1. **Open `CustomerForm.cs`** in code view.
2. **Add the necessary using directives**:

```
using BLL; //using BusinessLogicLayer;
using DAL; //using DataAccessLayer;
```

3. **Add a private field** for the `CustomerService`:

```
private readonly CustomerService _customerService;

public CustomerForm()
{
    InitializeComponent();
    _customerService = new CustomerService();
}
```

4. **Implement the `btnAddCustomer_Click` event handler**:

```
private void btnAddCustomer_Click(object sender,
EventArgs e)
{
    var customer = new Customer
    {
        Name = txtName.Text,
        Email = txtEmail.Text
    };
    _customerService.AddCustomer(customer);
    MessageBox.Show("Customer added successfully!");
}
```

5. **Implement the `btnGetCustomer_Click` event handler**:

```
private void btnGetCustomer_Click(object sender,
EventArgs e)
{
    int customerId = int.Parse(txtCustomerId.Text);
    var customer =
_customerService.GetCustomer(customerId);
    if (customer != null)
    {
        txtName.Text = customer.Name;
        txtEmail.Text = customer.Email;
    }
    else
    {
        MessageBox.Show("Customer not found!");
    }
}
```

Code to get data to datagridview

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

{

    LoadCustomers();

}

private void LoadCustomers()

{

    var customers = _customerService.GetAllCustomers();

    dataGridView1.DataSource=customers;

}
```

6. **Set `CustomerForm` as the startup form**:

- Open `Program.cs` in the `PresentationLayer` project.
- Modify the `Main` method to start `CustomerForm`:

```
Application.Run(new CustomerForm());
```

Step 5: Configuration for Database Connection to MSSQL

Make sure to configure the connection string in your App.config (under WinformUI project) or Web.config file for the Exer1DbContext.cs to connect to your database.

```
        <connectionStrings>

                <add name="MyConn"

                        connectionString="Data Source=(localdb)\MSSQL2019;Initial
Catalog=LabDB;Integrated Security=True"

                        providerName="System.Data.SqlClient" />

        </connectionStrings>
```

You should store the Exer1DbContext.cs file in the DAL (Data Access Layer) project. Here's how you can do it:

1. **Add the Exer1DbContext.cs file** to the **DAL** (DataAccessLayer) project:
    - Right-click on the **DAL**(DataAccessLayer) project in Solution Explorer.
    - Select Add > Class (*or rename Class1.cs*)
    - Name the class **Exer1DbContext.cs.**
2. **Add the following code** to the Exer1DbContext.cs file:

```
using System.Data.Entity;

public class Exer1DbContext: DbContext

{

   public DbSet<Customer> Customers { get; set; }

   public Exer1DbContext () : base("name=MyConn")

   {

   }

}
```

This will ensure that your **Exer1DbContext** class is part of the **DAL**(Data Access Layer), where it can manage the database connection and entity sets.

Step 6: Create table Customer with MSSQL

- Create database name : LabDB, create table Customer

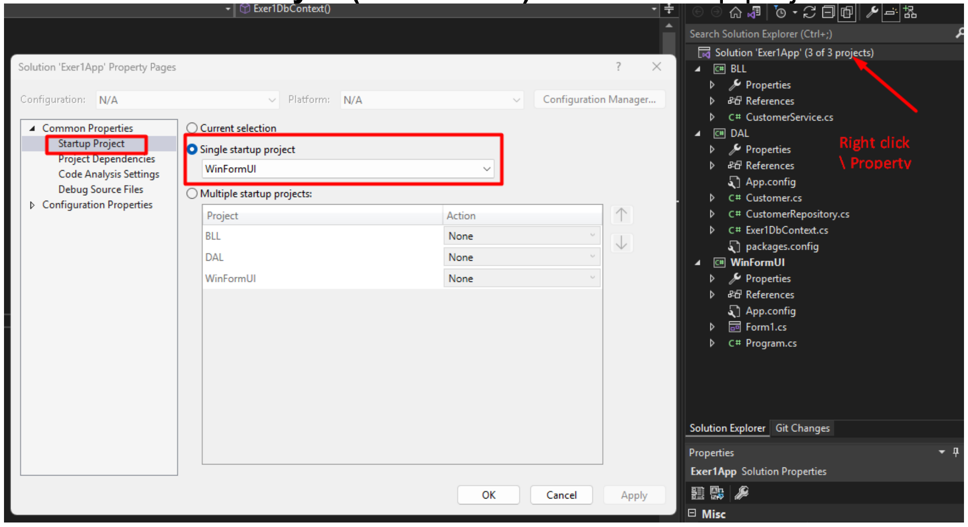
```
CREATE TABLE Customers (
   Id INT PRIMARY KEY IDENTITY(1,1),
   Name NVARCHAR(100),
    Email NVARCHAR(100)
);
```

Sample data into the Customers table:

```
INSERT INTO Customers (Name, Email) VALUES ('John Doe', 'john.doe@tdtu.com');
INSERT INTO Customers (Name, Email) VALUES ('Jane Smith', 'jane.smith@tdtu.com');
INSERT INTO Customers (Name, Email) VALUES ('Trung Pham', 'trung.pham@tdtu.com');
INSERT INTO Customers (Name, Email) VALUES ('Emily Davis', 'emily.davis@tdtu.com');
INSERT INTO Customers (Name, Email) VALUES ('Thai Pham', 'thai.pham@etdtu.com');
```

Step 7: Run the Application

1. **Build the solution** to ensure all projects compile successfully:
    - Go to Build > Build Solution.
2. **Run the Windows Forms application** and test the functionality:
    - Set **PresentationLayer (WinformUI)** as the startup project.



Solution 'Exer1App' Property Pages

Configuration: N/A — Platform: N/A — Configuration Manager...

- Common Properties
  - Startup Project
  - Project Dependencies
  - Code Analysis Settings
  - Debug Source Files
- Configuration Properties

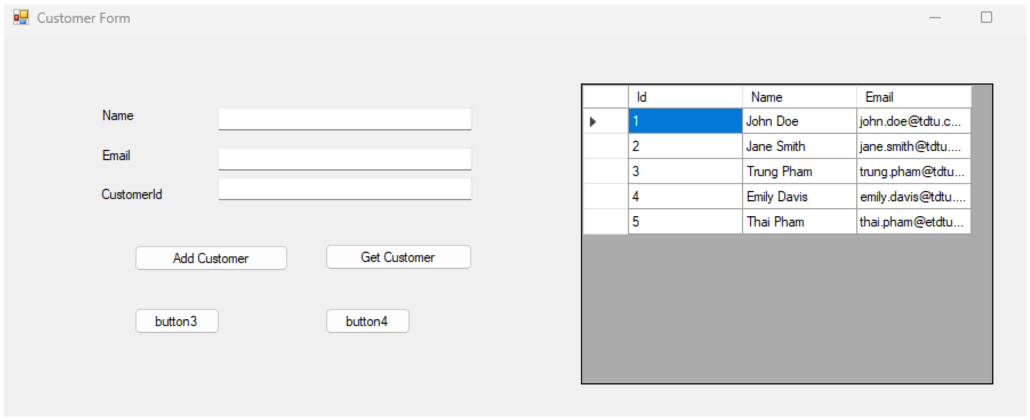
○ Current selection
● Single startup project: WinFormUI
○ Multiple startup projects:

Project	Action
BLL	None
DAL	None
WinFormUI	None

OK — Cancel — Apply

Solution 'Exer1App' (3 of 3 projects)
- BLL
  - Properties
  - References
  - CustomerService.cs
- DAL
  - Properties
  - References
  - App.config
  - Customer.cs
  - CustomerRepository.cs
  - Exer1DbContext.cs
  - packages.config
- WinFormUI
  - Properties
  - References
  - App.config
  - Form1.cs
  - Program.cs

    - Press F5 to run the application.



Customer Form

Name:
Email:
CustomerId:

Add Customer — Get Customer — button3 — button4

Id	Name	Email
1	John Doe	john.doe@tdtu.c...
2	Jane Smith	jane.smith@tdtu....
3	Trung Pham	trung.pham@tdtu...
4	Emily Davis	emily.davis@tdtu....
5	Thai Pham	thai.pham@etdtu...

This should give you a basic Windows Forms application using a 3-Tier Architecture. You can expand on this by adding more features, implementing dependency injection, and improving error handling.

Exercise 2: Login Form with 3-Tier Architecture

- Change Login form in Lab2 to 3-Tier Architecture
- Should create a new project with setting up from scratch like Exer1

Exercise 3: Login Form to Main form with 3-Tier Architecture

- Use Exer2, after logging in successfully then go to Main Form.
- On Main form, add menu strip with link to Customer form (reuse Exer 1)
- Also add Product Form, Supplier Form

III. Homework

-Use the same requirements from **Lab2 (School Management System)** homework but you should apply 3-Tier Architecture.

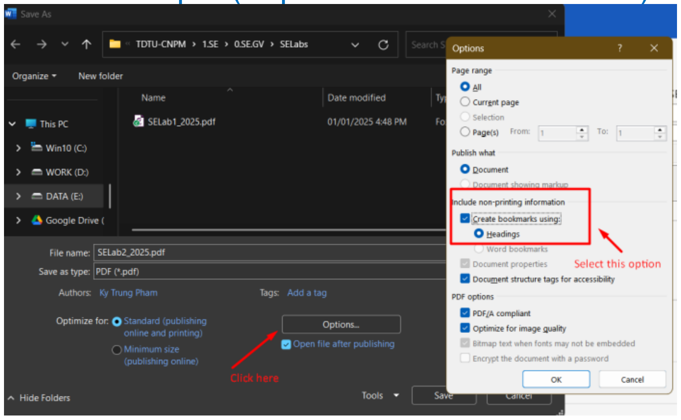
IV. Lab 3 Report Submission

- **Introduction:** Briefly describe the purpose of the exercises.
- **Exercises:** List each exercise with a brief description of what you did and learned for Class Activity included the screenshots of running App output.
- **Homework:** Describe what you do with including the screenshots of the output for homework exercise.
- **Conclusion:** Summarize your overall learning experience and any challenges you faced.

By following these exercises, you will learn how to create a 3-Tier Architecture Application, connect to an MSSQL database

Submit 2 files on eLearning:

- StudentID_StudentName.zip (contain Windows Form Project & .sql file) for both Class Activity and Homework.
- StudentID_StudentName.pdf (require to create bookmarks)



SELab1_2025.pdf 01/01/2025 4:48 PM

File name: SELab2_2025.pdf
Save as type: PDF (*.pdf)
Authors: Ky Trung Pham
Tags: Add a tag
Optimize for: [✓ Standard (publishing online and printing)] Minimum size (publishing online)
[✓ Open file after publishing]

Page range: [✓ All] Current page Selection Page(s) From: 1 To: 1
Publish what: [✓ Document] Document showing markup
Include non-printing information
[✓ Create bookmarks using:] [✓ Headings] Word bookmarks
[✓ Document properties]
[✓ Document structure tags for accessibility]
PDF options
[✓ PDF/A compliant]
[✓ Optimize for image quality]
[✓ Bitmap text when fonts may not be embedded]
Encrypt the document with a password

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

Enjoy ☺ Email: tg_phamthaikytrung@tdtu.edu.vn