Multitier architecture

In software engineering, multitier architecture (often referred to as n-tier architecture) or multilayered architecture is a client–server architecture in which **presentation**, **application** processing, and **data management** functions are physically separated. The most widespread use of multitier architecture is the three-tier architecture.

N-tier application architecture provides a model by which developers can create flexible and reusable applications. By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application. A three-tier architecture is typically composed of a presentation tier, a domain logic tier, and a data storage tier.

While the concepts of layer and tier are often used interchangeably, one fairly common point of view is that there is indeed a difference. This view holds that a layer is a logical structuring mechanism for the elements that make up the software solution, while a tier is a physical structuring mechanism for the system infrastructure. For example, a three-layer solution could easily be deployed on a single tier, such as a personal workstation.

Common layers

In a logical multilayered architecture for an information system with an object-oriented design, the following four are the most common:

- Presentation layer (a.k.a. UI layer, view layer, presentation tier in multitier architecture)
- Application layer (a.k.a. service layer or GRASP Controller Layer)
- Business layer (a.k.a. business logic layer (BLL), domain layer)
- Data access layer (a.k.a. persistence layer, logging, networking, and other services which are required to support a particular business layer)

Three-tier architecture

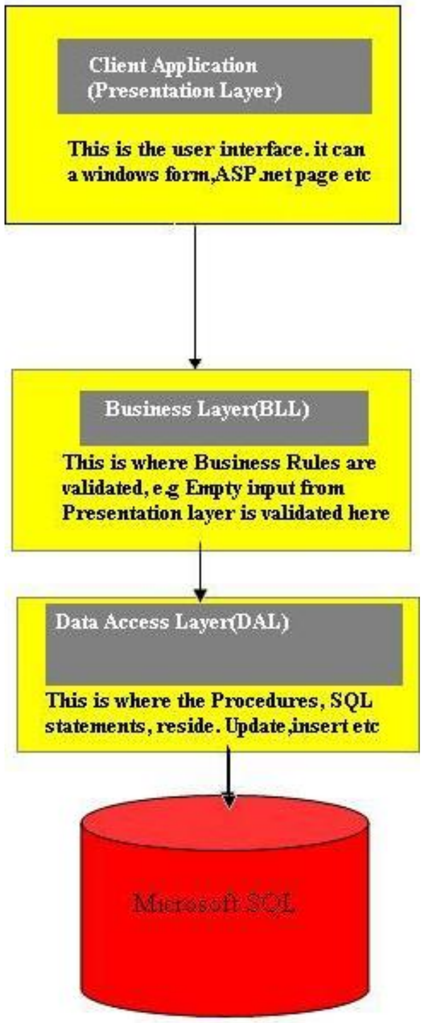
Three-tier architecture is a client–server software architecture pattern in which the user interface (presentation), functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms. It was developed by John J. Donovan in Open

Environment Corporation (OEC), a tools company he founded in Cambridge, Massachusetts.

Apart from the usual advantages of modular software with well-defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently in response to changes in requirements or technology. For example, a change of operating system in the presentation tier would only affect the user interface code.

Typically, the user interface runs on a desktop PC or workstation and uses a standard graphical user interface, functional process logic that may consist of one or more separate modules running on a workstation or application server, and an RDBMS on a database server or mainframe that contains the computer data storage logic. The middle tier may be multitiered itself (in which case the overall architecture is called an "n-tier architecture").

Three-tier architecture:


Client Application (Presentation Layer)
This is the user interface. it can a windows form,ASP.net page etc
Business Layer(BLL)
This is where Business Rules are validated, e.g Empty input from Presentation layer is validated here
Data Access Layer(DAL)
This is where the Procedures, SQL statements, reside. Update,insert etc
Microsoft SQL

- **Presentation tier**
  This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. In simple terms, it is a layer which users can access directly (such as a web page, or an operating system's GUI).
- **Application tier** (business logic, logic tier, or middle tier)
  The logical tier is pulled out from the presentation tier and, as its own layer, it controls an application’s functionality by performing detailed processing.
- **Data tier**
  The data tier includes the data persistence mechanisms (database servers, file shares, etc.) and the data access layer that encapsulates the persistence mechanisms and exposes the data. The data access layer should provide an API to the application tier that exposes methods of managing the stored data without exposing or creating dependencies on the data storage mechanisms. Avoiding dependencies on the storage

mechanisms allows for updates or changes without the application tier clients being affected by or even aware of the change. As with the separation of any tier, there are costs for implementation and often costs to performance in exchange for improved scalability and maintainability.

Benefits of separating an application into tiers

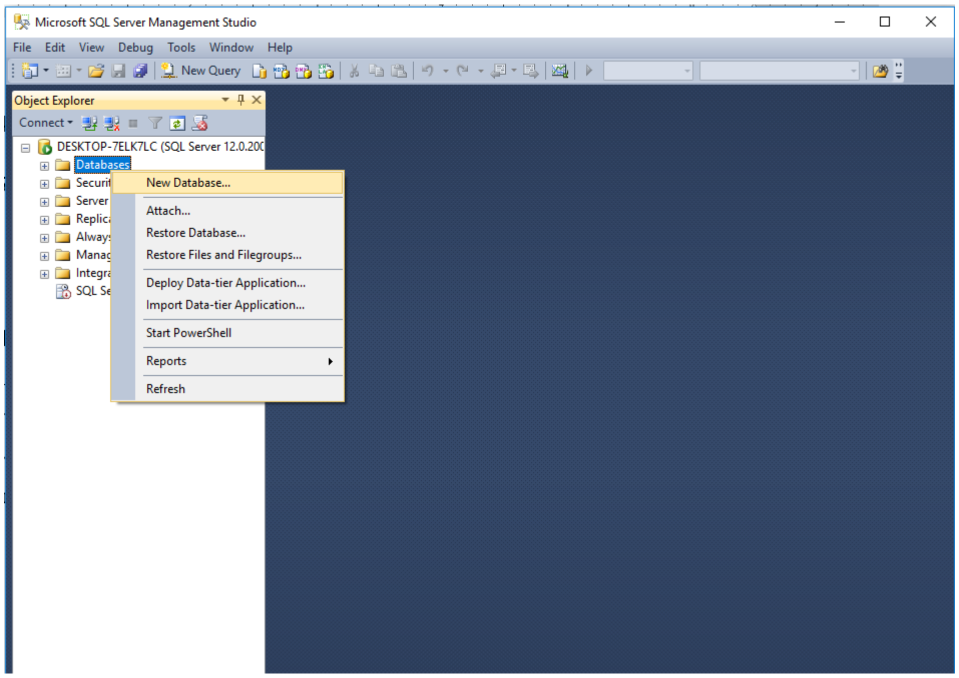
1. It gives you the ability to update the technology stack of one tier, without impacting other areas of the application.
2. It allows for different development teams to each work on their own areas of expertise. Today's developers are more likely to have deep competency in one area, like coding the front end of an application, instead of working on the full stack.
3. You are able to scale the application up and out. A separate back-end tier, for example, allows you to deploy to a variety of databases instead of being locked into one particular technology. It also allows you to scale up by adding multiple web servers.
4. It adds reliability and more independence of the underlying servers or services.
5. It provides an ease of maintenance of the code base, managing presentation code and business logic separately, so that a change to business logic, for example, does not impact the presentation layer.

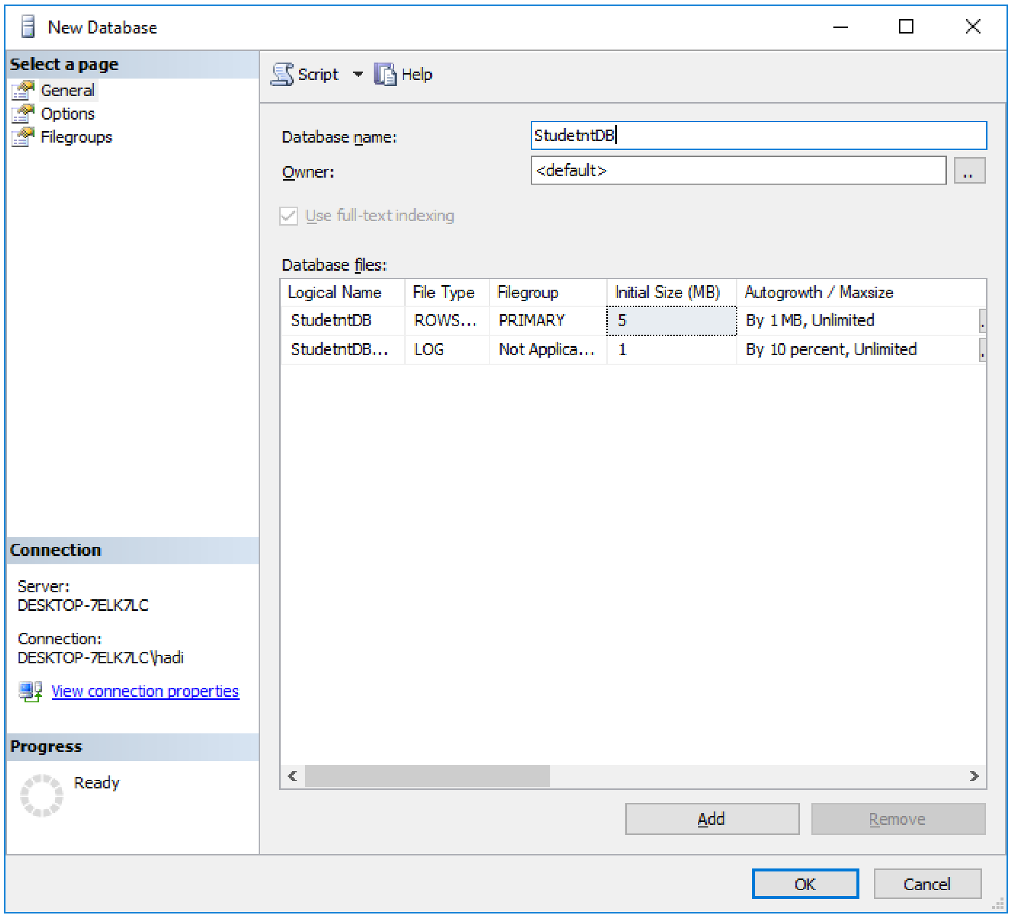
Example of Three tier architecture

We will implement university application were we will have student name and can add student and secure our application with user name and password.

Step 1. Create database

First create database in sql server


Microsoft SQL Server Management Studio
File Edit View Debug Tools Window Help
New Query
Object Explorer
Connect
DESKTOP-7ELK7LC (SQL Server 12.0.200
Databases
New Database...
Attach...
Restore Database...
Restore Files and Filegroups...
Deploy Data-tier Application...
Import Data-tier Application...
Start PowerShell
Reports
Refresh



Database name: StudetntDB

Owner: <default>

Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize
StudetntDB	ROWS...	PRIMARY	5	By 1 MB, Unlimited
StudetntDB...	LOG	Not Applica...	1	By 10 percent, Unlimited

Connection

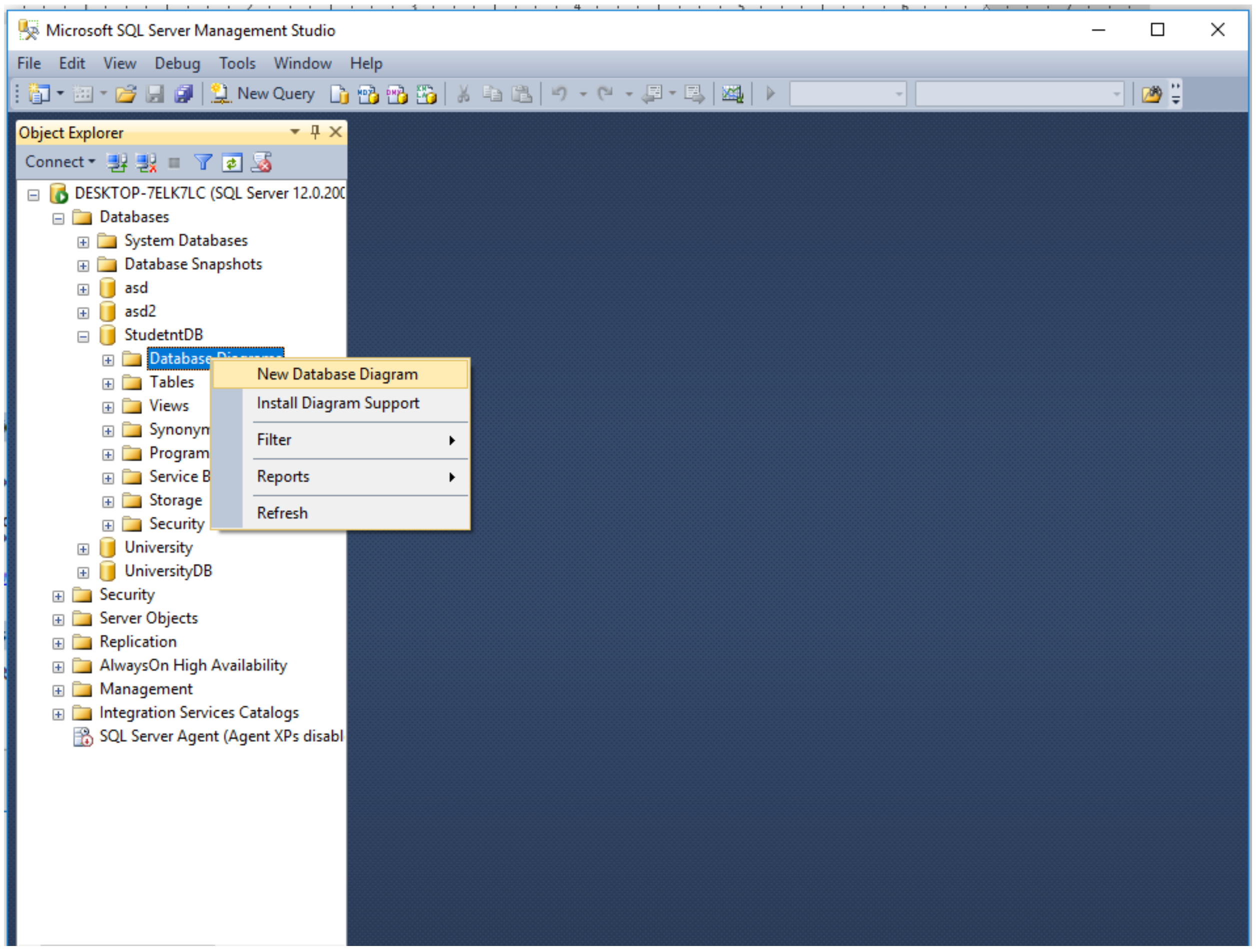
Server:
DESKTOP-7ELK7LC

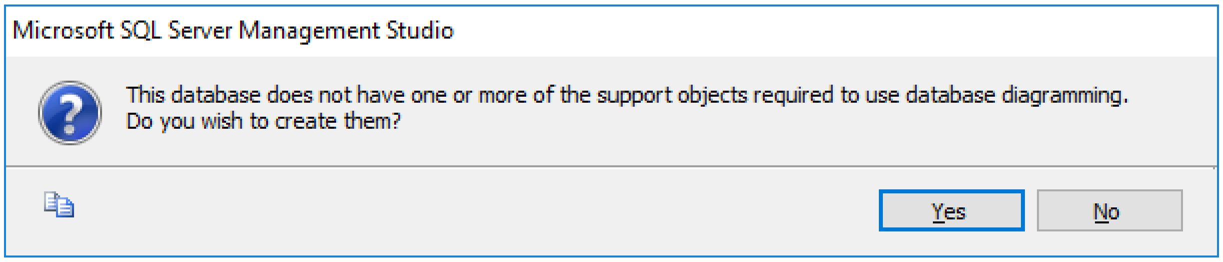
Connection:
DESKTOP-7ELK7LC\hadi

View connection properties

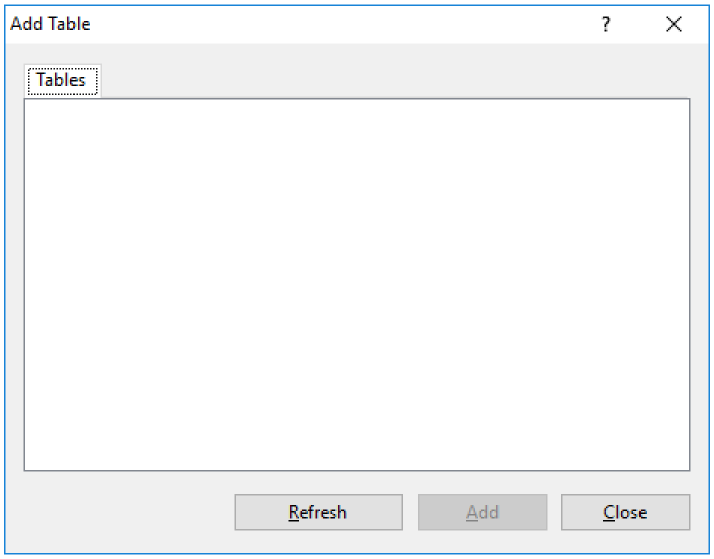
Progress

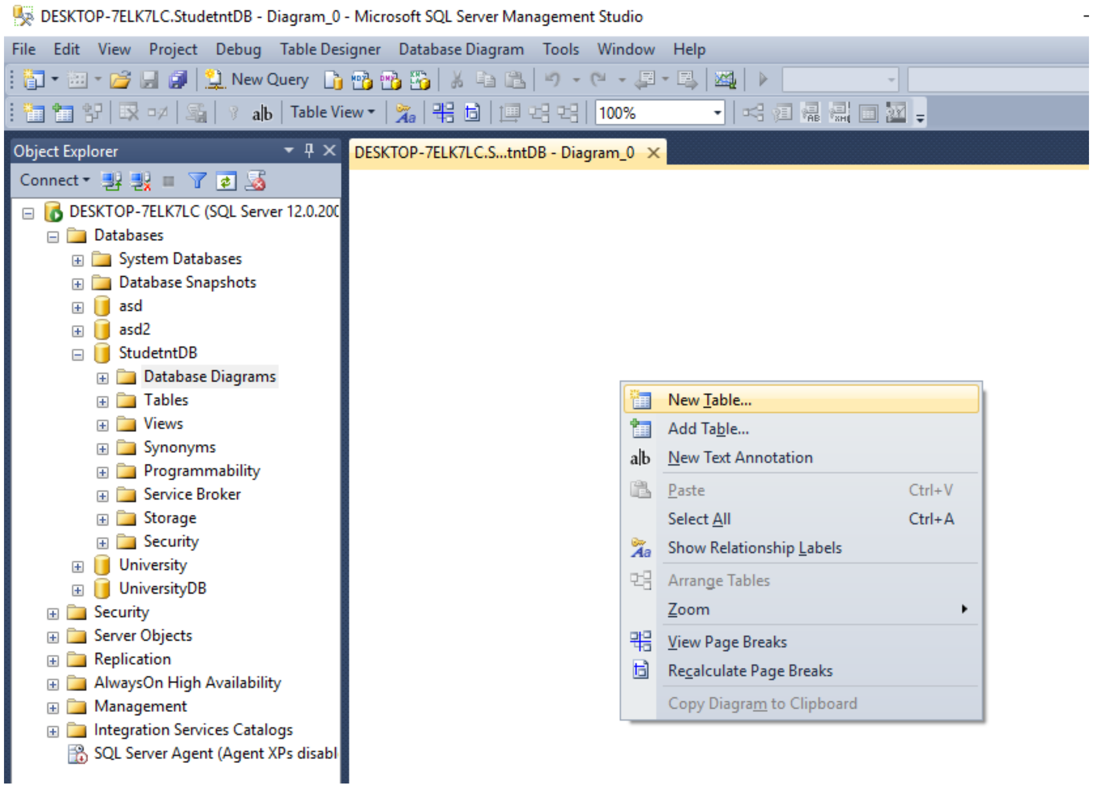
Ready

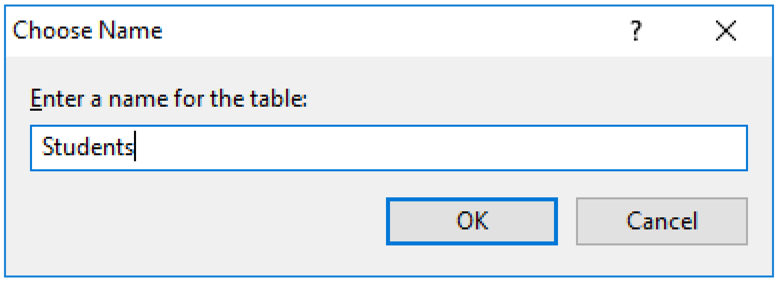


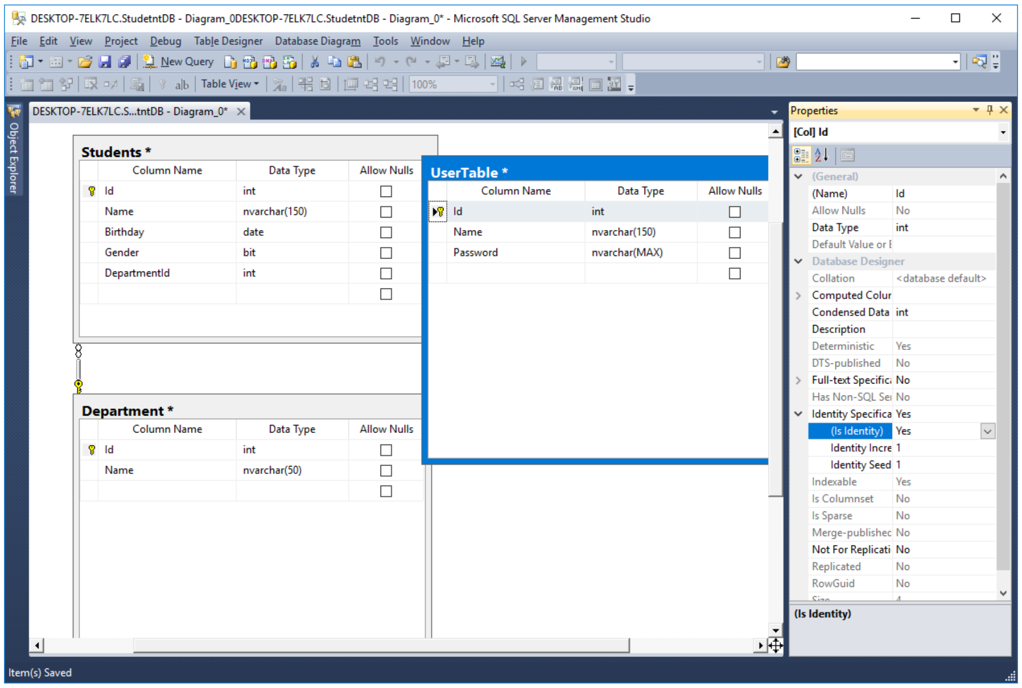

Microsoft SQL Server Management Studio

This database does not have one or more of the support objects required to use database diagramming. Do you wish to create them?


Add Table
?
Tables
Refresh
Add
Close


DESKTOP-7ELK7LC.StudetntDB - Diagram_0 - Microsoft SQL Server Management Studio
File Edit View Project Debug Table Designer Database Diagram Tools Window Help
New Query
Table View
100%
Object Explorer
Connect
DESKTOP-7ELK7LC.S...tntDB - Diagram_0
DESKTOP-7ELK7LC (SQL Server 12.0.200
Databases
System Databases
Database Snapshots
asd
asd2
StudetntDB
Database Diagrams
Tables
Views
Synonyms
Programmability
Service Broker
Storage
Security
University
UniversityDB
Security
Server Objects
Replication
AlwaysOn High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
New Table...
Add Table...
New Text Annotation
Paste Ctrl+V
Select All Ctrl+A
Show Relationship Labels
Arrange Tables
Zoom
View Page Breaks
Recalculate Page Breaks
Copy Diagram to Clipboard


Choose Name
?
Enter a name for the table:
Students
OK
Cancel


DESKTOP-7ELK7LC.StudetntDB - Diagram_0DESKTOP-7ELK7LC.StudetntDB - Diagram_0* - Microsoft SQL Server Management Studio

Students *

Column Name	Data Type	Allow Nulls
Id	int	☐
Name	nvarchar(150)	☐
Birthday	date	☐
Gender	bit	☐
DepartmentId	int	☐

UserTable *

Column Name	Data Type	Allow Nulls
Id	int	☐
Name	nvarchar(150)	☐
Password	nvarchar(MAX)	☐

Department *

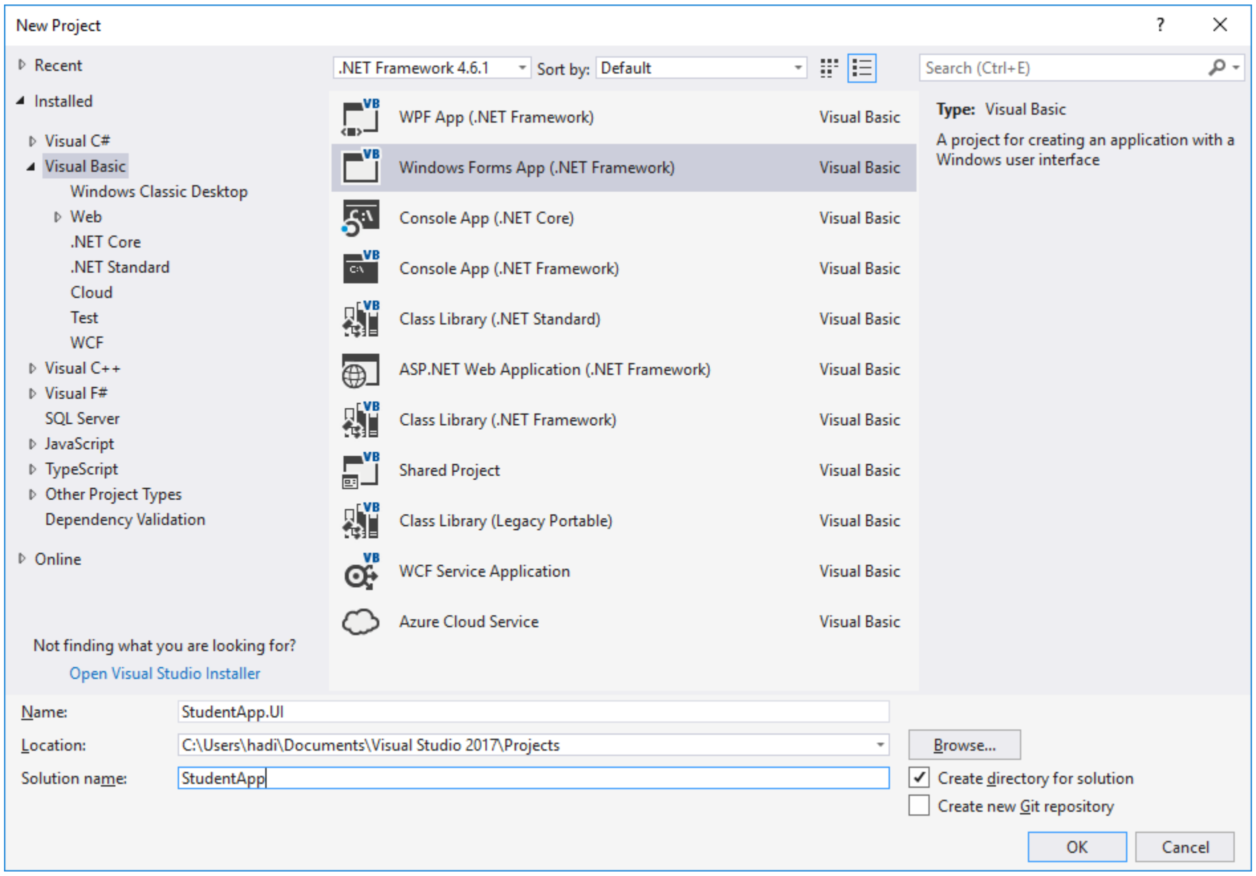
Column Name	Data Type	Allow Nulls
Id	int	☐
Name	nvarchar(50)	☐

Properties — [Col] Id

(General)	
(Name)	Id
Allow Nulls	No
Data Type	int
Default Value or B	
Database Designer	
Collation	<database default>
Computed Colur	
Condensed Data	int
Description	
Deterministic	Yes
DTS-published	No
Full-text Specifici	No
Has Non-SQL Ser	No
Identity Specifica	Yes
(Is Identity)	Yes
Identity Incre	1
Identity Seed	1
Indexable	Yes
Is Columnset	No
Is Sparse	No
Merge-publishec	No
Not For Replicati	No
Replicated	No
RowGuid	No

Item(s) Saved

Step 2. Create win form application


New Project

.NET Framework 4.6.1 — Sort by: Default

Installed > Visual Basic

WPF App (.NET Framework)	Visual Basic
Windows Forms App (.NET Framework)	Visual Basic
Console App (.NET Core)	Visual Basic
Console App (.NET Framework)	Visual Basic
Class Library (.NET Standard)	Visual Basic
ASP.NET Web Application (.NET Framework)	Visual Basic
Class Library (.NET Framework)	Visual Basic
Shared Project	Visual Basic
Class Library (Legacy Portable)	Visual Basic
WCF Service Application	Visual Basic
Azure Cloud Service	Visual Basic

Type: Visual Basic

A project for creating an application with a Windows user interface

Name: StudentApp.UI

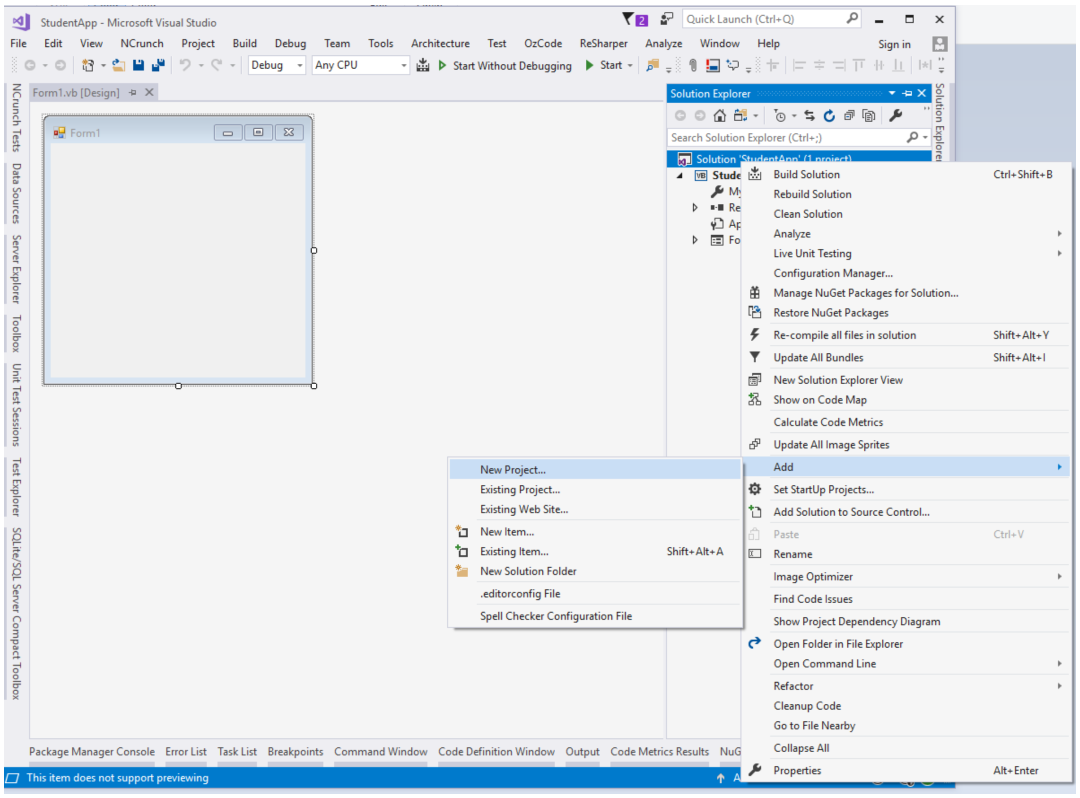
Location: C:\Users\hadi\Documents\Visual Studio 2017\Projects

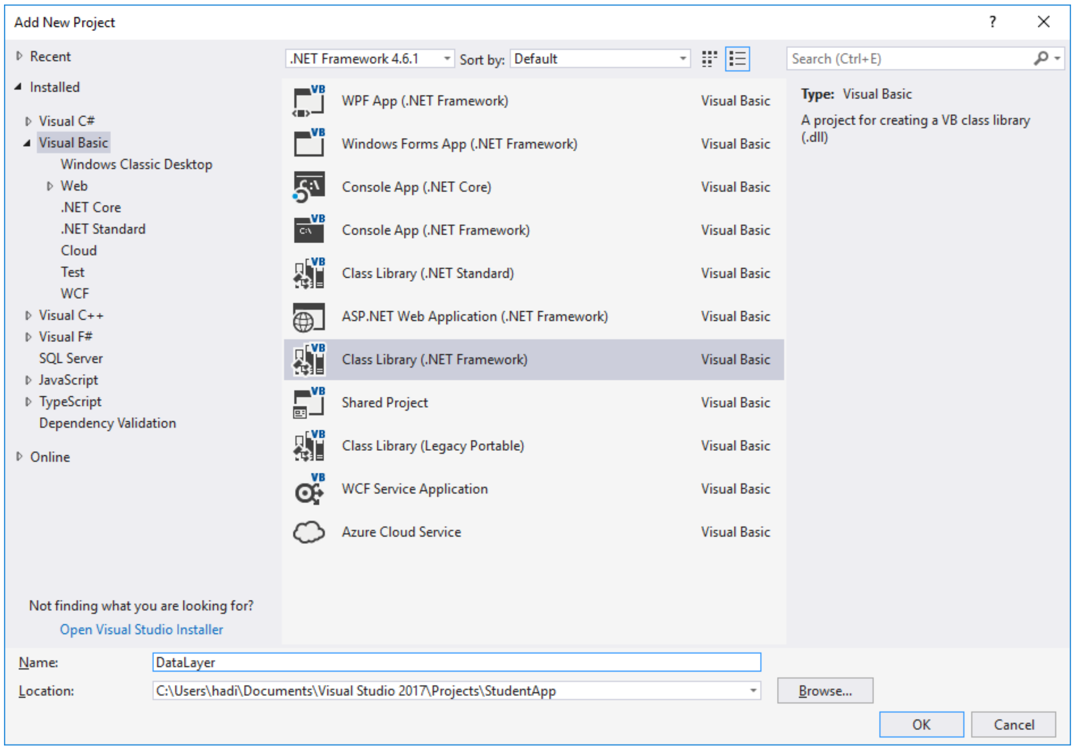
Solution name: StudentApp

☑ Create directory for solution

☐ Create new Git repository

Add new project


StudentApp - Microsoft Visual Studio
File Edit View NCrunch Project Build Debug Team Tools Architecture Test OzCode ReSharper Analyze Window Help
Solution Explorer
Solution 'StudentApp' (1 project)
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Analyze
Live Unit Testing
Configuration Manager...
Manage NuGet Packages for Solution...
Restore NuGet Packages
Re-compile all files in solution Shift+Alt+Y
Update All Bundles Shift+Alt+I
New Solution Explorer View
Show on Code Map
Calculate Code Metrics
Update All Image Sprites
Add
Set StartUp Projects...
Add Solution to Source Control...
Paste Ctrl+V
Rename
Image Optimizer
Find Code Issues
Show Project Dependency Diagram
Open Folder in File Explorer
Open Command Line
Refactor
Cleanup Code
Go to File Nearby
Collapse All
Properties Alt+Enter
New Project...
Existing Project...
Existing Web Site...
New Item...
Existing Item... Shift+Alt+A
New Solution Folder
.editorconfig File
Spell Checker Configuration File
This item does not support previewing



Add New Project

.NET Framework 4.6.1 Sort by: Default

Installed > Visual Basic

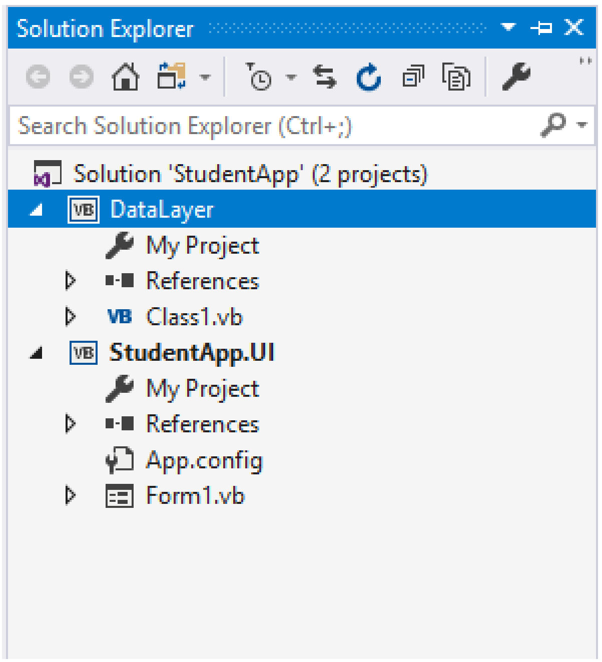
WPF App (.NET Framework)	Visual Basic
Windows Forms App (.NET Framework)	Visual Basic
Console App (.NET Core)	Visual Basic
Console App (.NET Framework)	Visual Basic
Class Library (.NET Standard)	Visual Basic
ASP.NET Web Application (.NET Framework)	Visual Basic
Class Library (.NET Framework)	Visual Basic
Shared Project	Visual Basic
Class Library (Legacy Portable)	Visual Basic
WCF Service Application	Visual Basic
Azure Cloud Service	Visual Basic

Type: Visual Basic

A project for creating a VB class library (.dll)

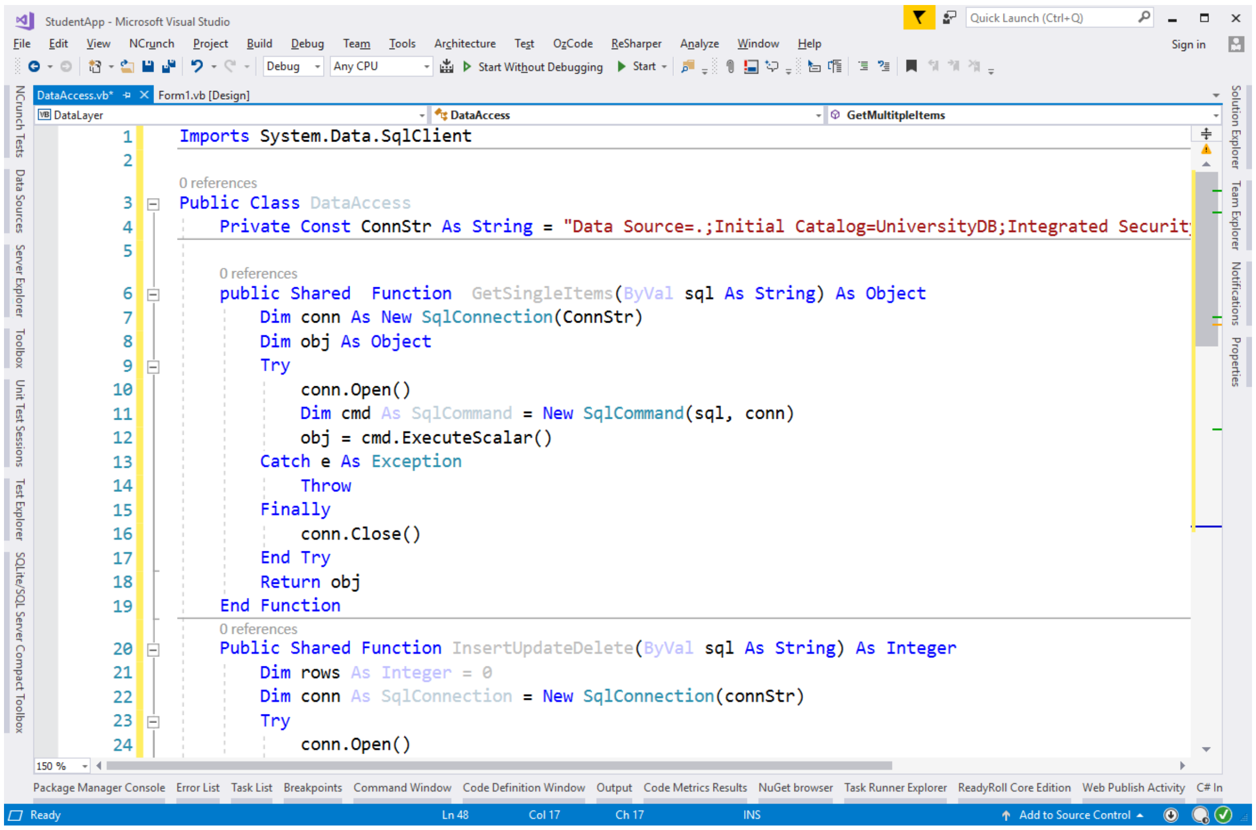
Name: DataLayer

Location: C:\Users\hadi\Documents\Visual Studio 2017\Projects\StudentApp



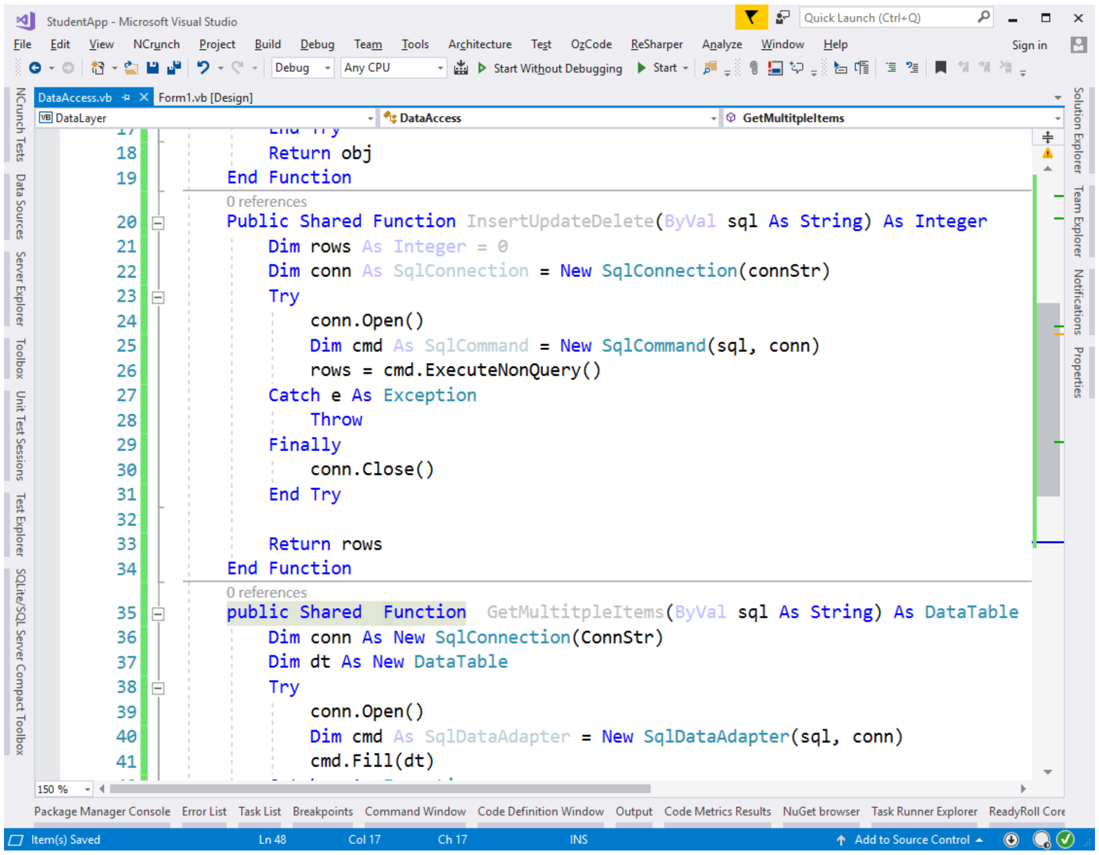
```
Solution 'StudentApp' (2 projects)
◢ DataLayer
      My Project
   ▷  References
   ▷  Class1.vb
◢ StudentApp.UI
      My Project
   ▷  References
      App.config
   ▷  Form1.vb
```

Then write the code:

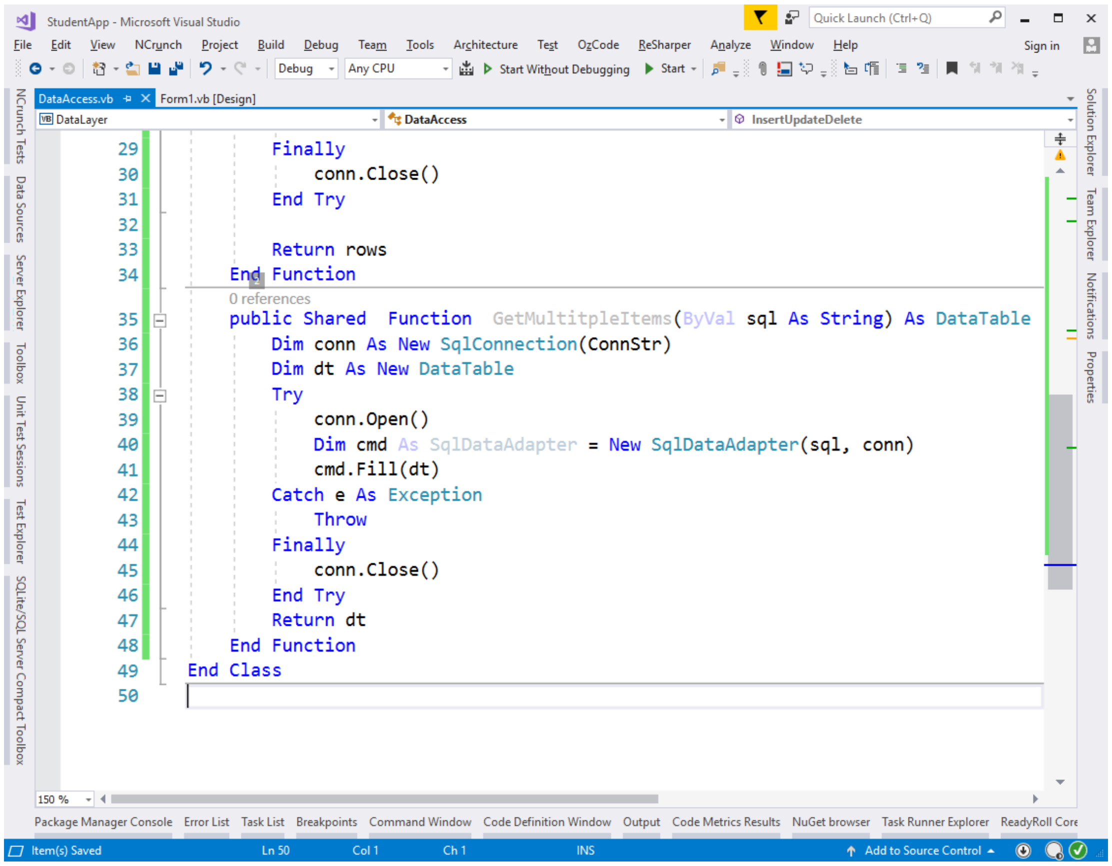


```vb
Imports System.Data.SqlClient

0 references
Public Class DataAccess
    Private Const ConnStr As String = "Data Source=.;Initial Catalog=UniversityDB;Integrated Securit

    0 references
    public Shared  Function  GetSingleItems(ByVal sql As String) As Object
        Dim conn As New SqlConnection(ConnStr)
        Dim obj As Object
        Try
            conn.Open()
            Dim cmd As SqlCommand = New SqlCommand(sql, conn)
            obj = cmd.ExecuteScalar()
        Catch e As Exception
            Throw
        Finally
            conn.Close()
        End Try
        Return obj
    End Function
    0 references
    Public Shared Function InsertUpdateDelete(ByVal sql As String) As Integer
        Dim rows As Integer = 0
        Dim conn As SqlConnection = New SqlConnection(connStr)
        Try
            conn.Open()
```

StudentApp - Microsoft Visual Studio
File Edit View NCrunch Project Build Debug Team Tools Architecture Test OzCode ReSharper Analyze Window Help
Sign in
Quick Launch (Ctrl+Q)
Debug
Any CPU
Start Without Debugging
Start
DataAccess.vb*
Form1.vb [Design]
DataLayer
DataAccess
GetMultitpleItems
150 %
Package Manager Console Error List Task List Breakpoints Command Window Code Definition Window Output Code Metrics Results NuGet browser Task Runner Explorer ReadyRoll Core Edition Web Publish Activity C# In
Ready
Ln 48
Col 17
Ch 17
INS
Add to Source Control
NCrunch Tests
Data Sources
Server Explorer
Toolbox
Unit Test Sessions
Test Explorer
SQLite/SQL Server Compact Toolbox
Solution Explorer
Team Explorer
Notifications
Properties

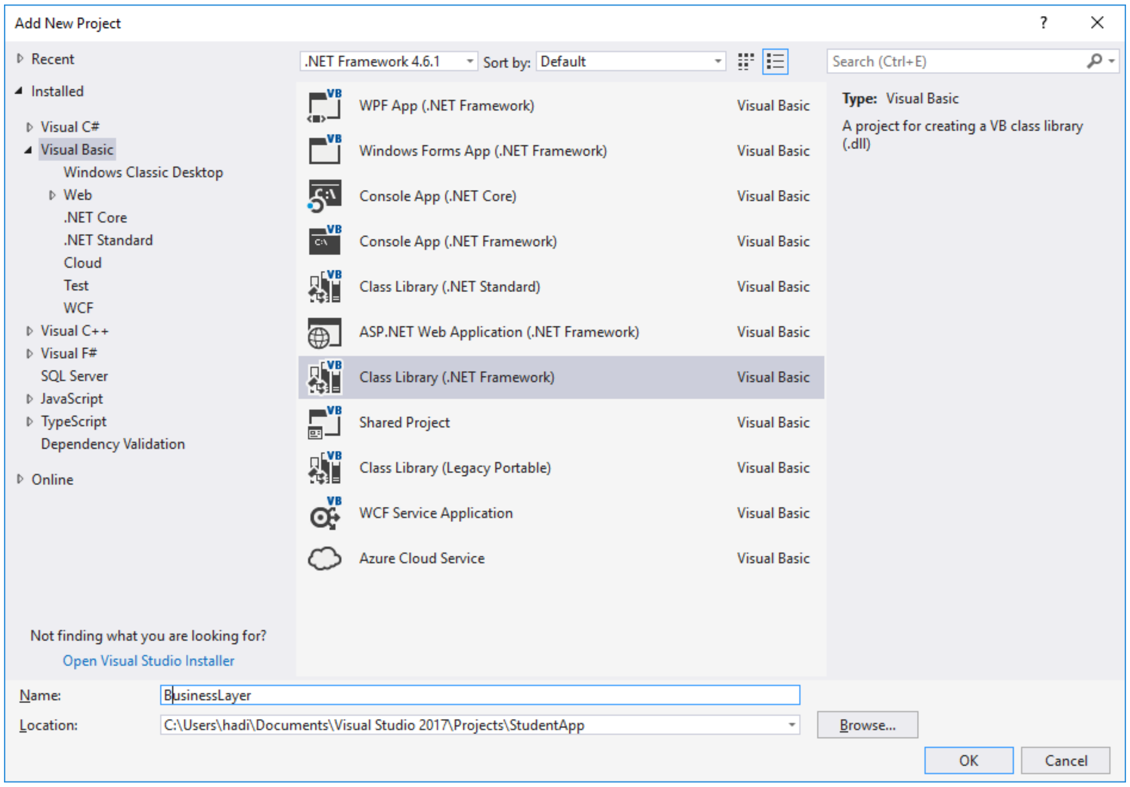


```
        Return obj
    End Function
    0 references
    Public Shared Function InsertUpdateDelete(ByVal sql As String) As Integer
        Dim rows As Integer = 0
        Dim conn As SqlConnection = New SqlConnection(connStr)
        Try
            conn.Open()
            Dim cmd As SqlCommand = New SqlCommand(sql, conn)
            rows = cmd.ExecuteNonQuery()
        Catch e As Exception
            Throw
        Finally
            conn.Close()
        End Try

        Return rows
    End Function
    0 references
    public Shared  Function  GetMultitpleItems(ByVal sql As String) As DataTable
        Dim conn As New SqlConnection(ConnStr)
        Dim dt As New DataTable
        Try
            conn.Open()
            Dim cmd As SqlDataAdapter = New SqlDataAdapter(sql, conn)
            cmd.Fill(dt)
```



```
        Finally
            conn.Close()
        End Try

        Return rows
    End Function
    0 references
    public Shared  Function  GetMultitpleItems(ByVal sql As String) As DataTable
        Dim conn As New SqlConnection(ConnStr)
        Dim dt As New DataTable
        Try
            conn.Open()
            Dim cmd As SqlDataAdapter = New SqlDataAdapter(sql, conn)
            cmd.Fill(dt)
        Catch e As Exception
            Throw
        Finally
            conn.Close()
        End Try
        Return dt
    End Function
End Class
```

Add two new Project


Add New Project

- Recent
- Installed
  - Visual C#
  - Visual Basic
    - Windows Classic Desktop
    - Web
    - .NET Core
    - .NET Standard
    - Cloud
    - Test
    - WCF
  - Visual C++
  - Visual F#
  - SQL Server
  - JavaScript
  - TypeScript
  - Dependency Validation
- Online

.NET Framework 4.6.1 — Sort by: Default — Search (Ctrl+E)

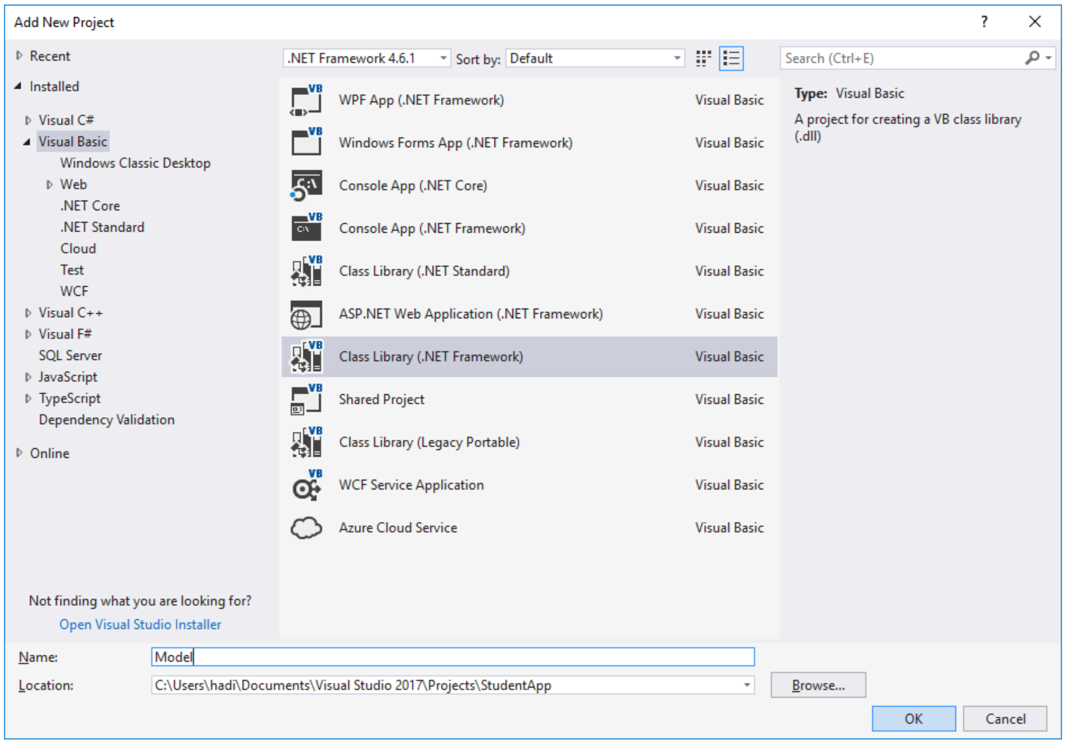
WPF App (.NET Framework)	Visual Basic
Windows Forms App (.NET Framework)	Visual Basic
Console App (.NET Core)	Visual Basic
Console App (.NET Framework)	Visual Basic
Class Library (.NET Standard)	Visual Basic
ASP.NET Web Application (.NET Framework)	Visual Basic
Class Library (.NET Framework)	Visual Basic
Shared Project	Visual Basic
Class Library (Legacy Portable)	Visual Basic
WCF Service Application	Visual Basic
Azure Cloud Service	Visual Basic

Type: Visual Basic

A project for creating a VB class library (.dll)

Not finding what you are looking for?
Open Visual Studio Installer

Name: BusinessLayer
Location: C:\Users\hadi\Documents\Visual Studio 2017\Projects\StudentApp


Add New Project

- Recent
- Installed
  - Visual C#
  - Visual Basic
    - Windows Classic Desktop
    - Web
    - .NET Core
    - .NET Standard
    - Cloud
    - Test
    - WCF
  - Visual C++
  - Visual F#
  - SQL Server
  - JavaScript
  - TypeScript
  - Dependency Validation
- Online

.NET Framework 4.6.1 — Sort by: Default — Search (Ctrl+E)

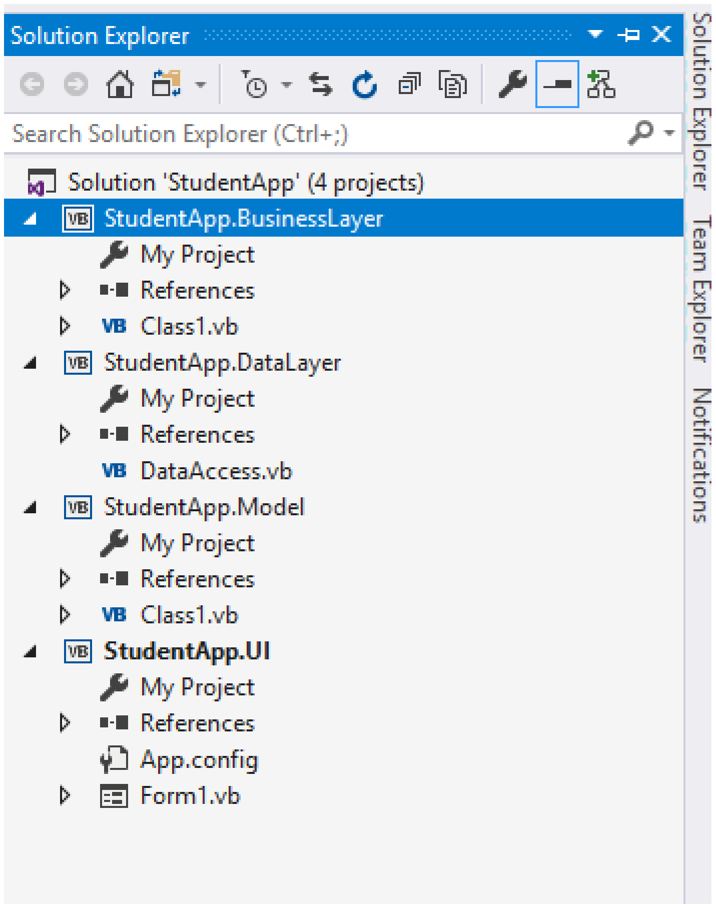
WPF App (.NET Framework)	Visual Basic
Windows Forms App (.NET Framework)	Visual Basic
Console App (.NET Core)	Visual Basic
Console App (.NET Framework)	Visual Basic
Class Library (.NET Standard)	Visual Basic
ASP.NET Web Application (.NET Framework)	Visual Basic
Class Library (.NET Framework)	Visual Basic
Shared Project	Visual Basic
Class Library (Legacy Portable)	Visual Basic
WCF Service Application	Visual Basic
Azure Cloud Service	Visual Basic

Type: Visual Basic

A project for creating a VB class library (.dll)

Not finding what you are looking for?
Open Visual Studio Installer

Name: Model
Location: C:\Users\hadi\Documents\Visual Studio 2017\Projects\StudentApp

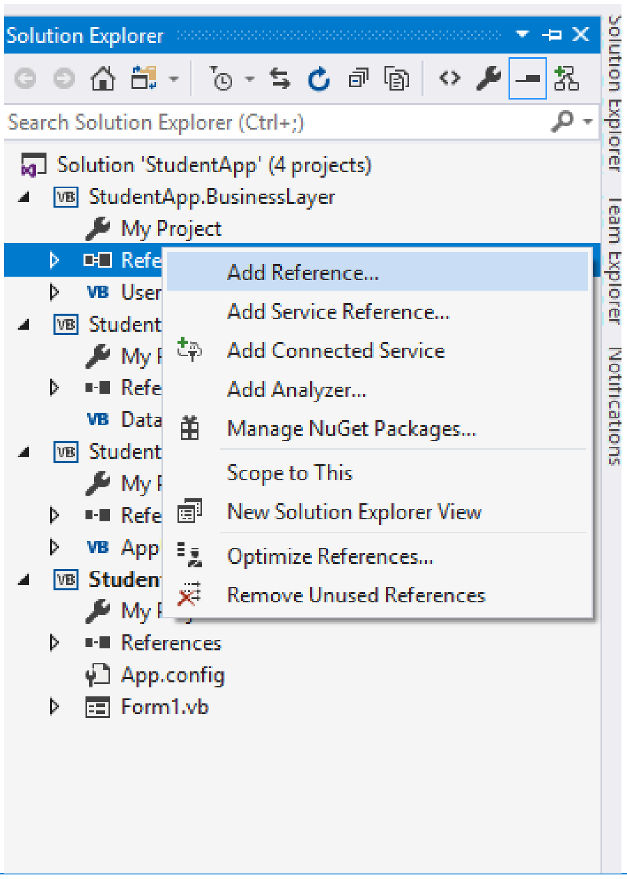


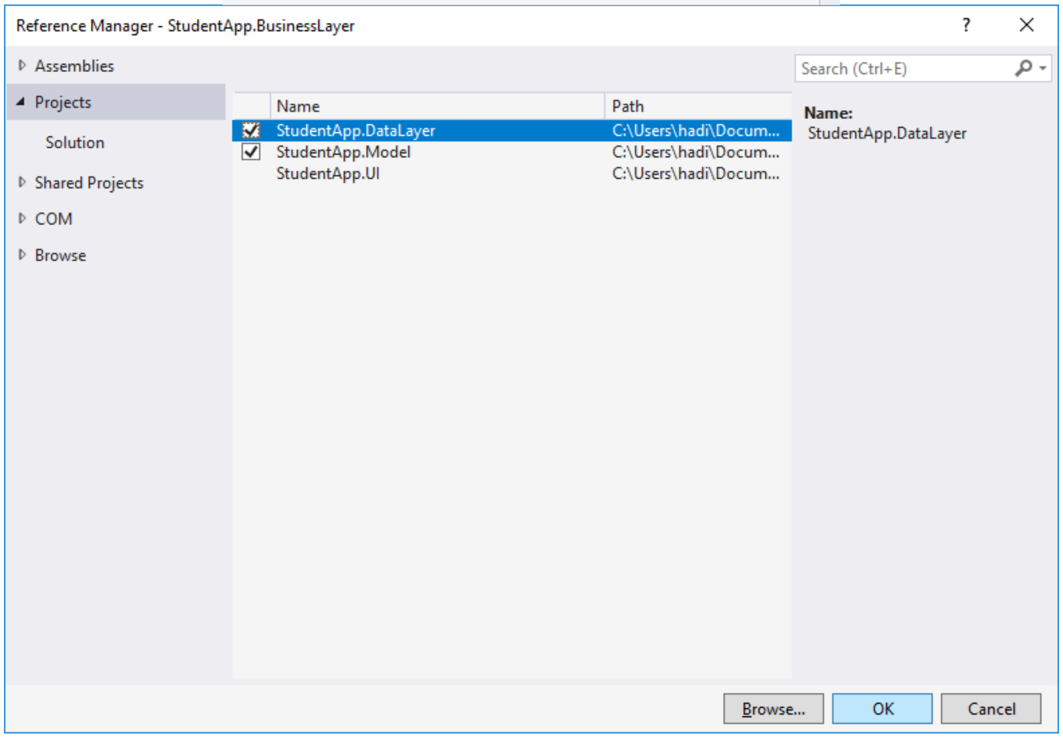
```
Solution 'StudentApp' (4 projects)
◢ StudentApp.BusinessLayer
      My Project
    ▷ References
    ▷ Class1.vb
◢ StudentApp.DataLayer
      My Project
    ▷ References
      DataAccess.vb
◢ StudentApp.Model
      My Project
    ▷ References
    ▷ Class1.vb
◢ StudentApp.UI
      My Project
    ▷ References
      App.config
    ▷ Form1.vb
```

First model we create will by user model

```
Public Class AppUser
    Public Property Id As Integer
    Public Property Name As String
    Public Property Password As String
End Class
```

Add refrence to model and datalayer from business layer

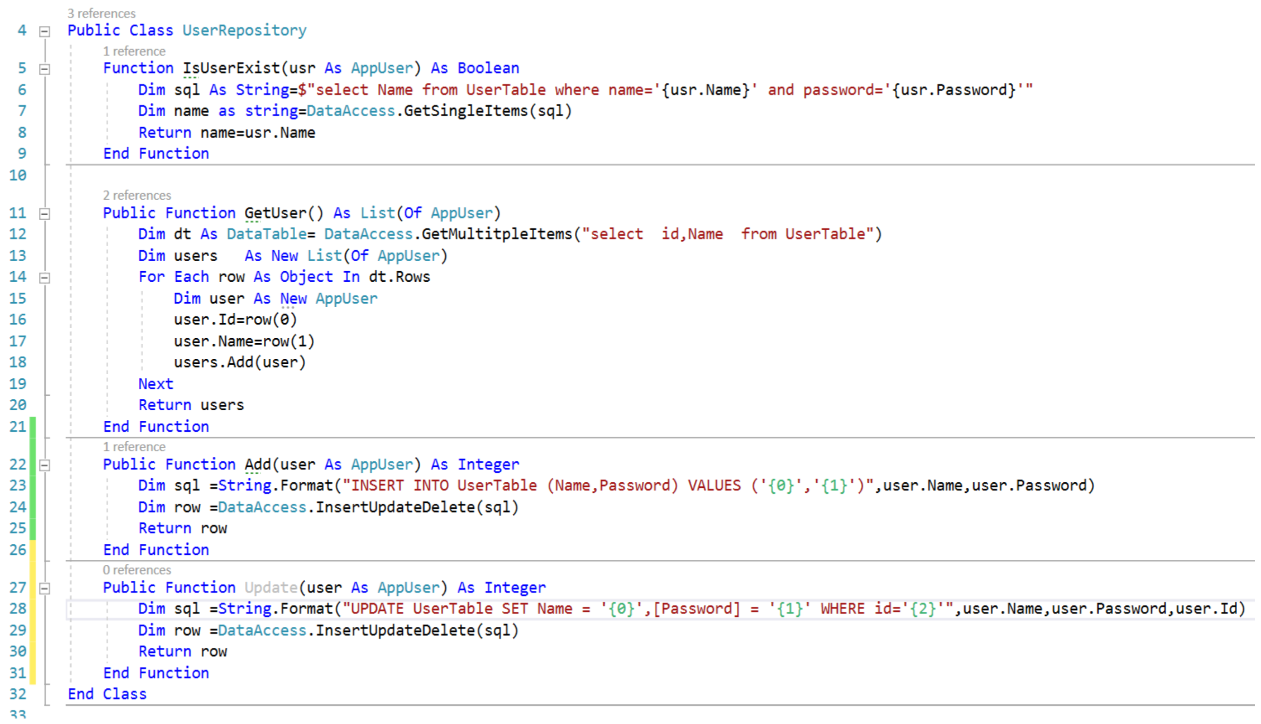




	Name	Path
✓	StudentApp.DataLayer	C:\Users\hadi\Docum...
✓	StudentApp.Model	C:\Users\hadi\Docum...
	StudentApp.UI	C:\Users\hadi\Docum...

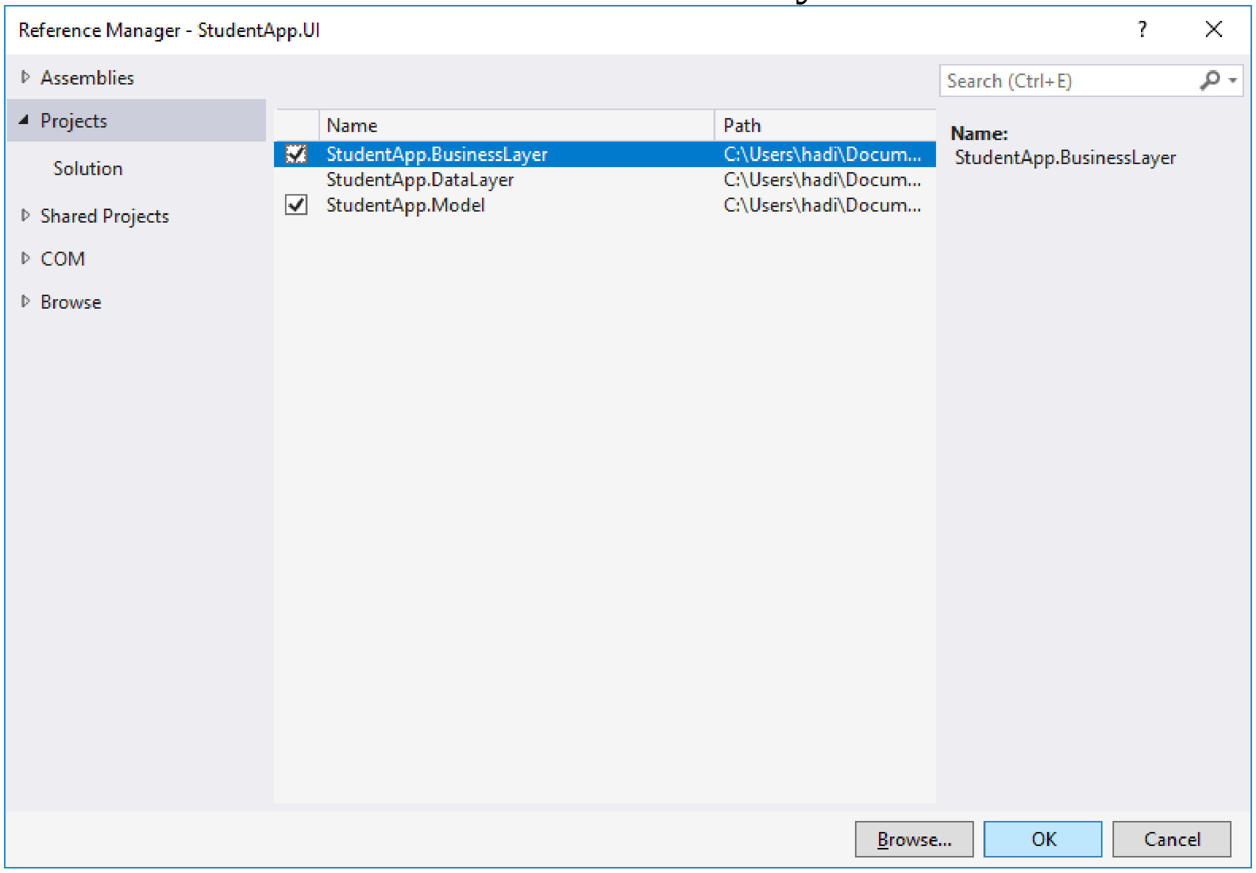
Name:
StudentApp.DataLayer

Now we add user repository in business Layer



```
3 references
Public Class UserRepository
    1 reference
    Function IsUserExist(usr As AppUser) As Boolean
        Dim sql As String=$"select Name from UserTable where name='{usr.Name}' and password='{usr.Password}'"
        Dim name as string=DataAccess.GetSingleItems(sql)
        Return name=usr.Name
    End Function

    2 references
    Public Function GetUser() As List(Of AppUser)
        Dim dt As DataTable= DataAccess.GetMultitpleItems("select  id,Name  from UserTable")
        Dim users   As New List(Of AppUser)
        For Each row As Object In dt.Rows
            Dim user As New AppUser
            user.Id=row(0)
            user.Name=row(1)
            users.Add(user)
        Next
        Return users
    End Function
    1 reference
    Public Function Add(user As AppUser) As Integer
        Dim sql =String.Format("INSERT INTO UserTable (Name,Password) VALUES ('{0}','{1}')",user.Name,user.Password)
        Dim row =DataAccess.InsertUpdateDelete(sql)
        Return row
    End Function
    0 references
    Public Function Update(user As AppUser) As Integer
        Dim sql =String.Format("UPDATE UserTable SET Name = '{0}',[Password] = '{1}' WHERE id='{2}'",user.Name,user.Password,user.Id)
        Dim row =DataAccess.InsertUpdateDelete(sql)
        Return row
    End Function
End Class
```

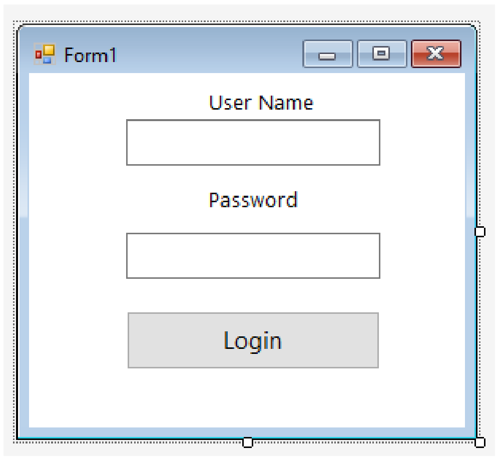
Add reference from UI to business layer



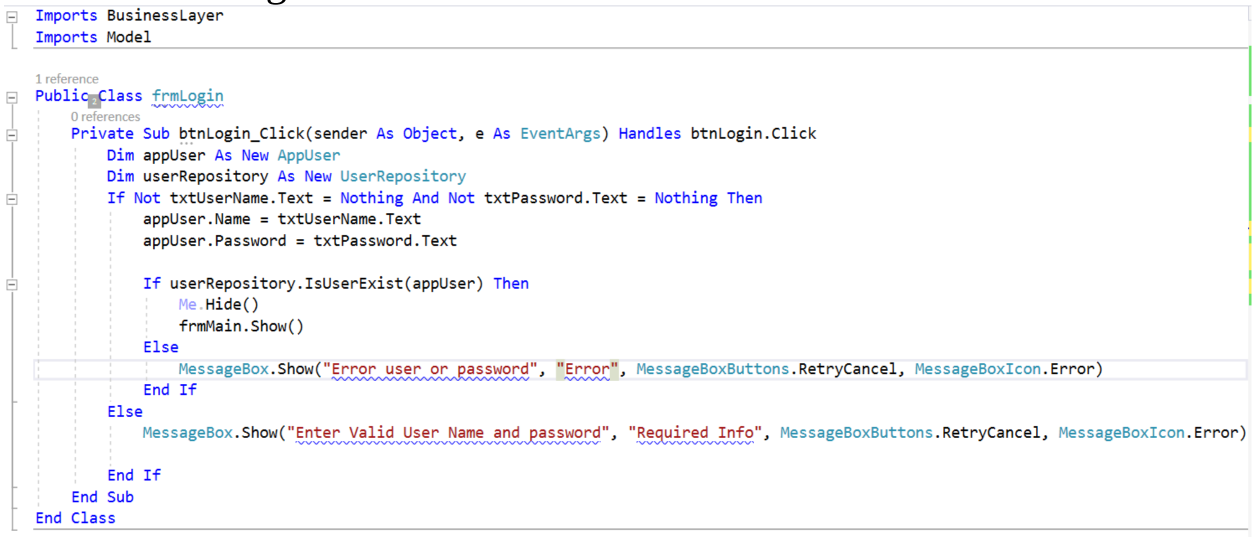
	Name	Path
✓	StudentApp.BusinessLayer	C:\Users\hadi\Docum...
	StudentApp.DataLayer	C:\Users\hadi\Docum...
✓	StudentApp.Model	C:\Users\hadi\Docum...

Name:
StudentApp.BusinessLayer

Control	Properties	Value
Form	Text Font Back Color	Form login Segoe UI, 9.75pt White
Label 1	Text	User Name
Label 2	Text	Password
Text Box	(Name)	txtUserName
Text Box	(Name)	txtPassword
Button	Name Text	btnLogin Login



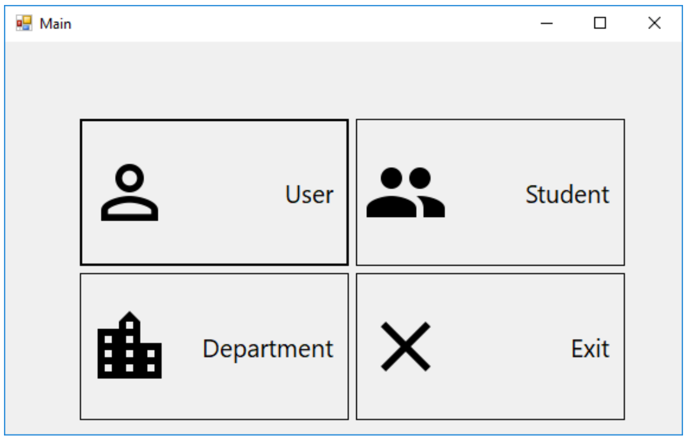
The code for login will be as follow



```
Imports BusinessLayer
Imports Model

1 reference
Public Class frmLogin
    0 references
    Private Sub btnLogin_Click(sender As Object, e As EventArgs) Handles btnLogin.Click
        Dim appUser As New AppUser
        Dim userRepository As New UserRepository
        If Not txtUserName.Text = Nothing And Not txtPassword.Text = Nothing Then
            appUser.Name = txtUserName.Text
            appUser.Password = txtPassword.Text

            If userRepository.IsUserExist(appUser) Then
                Me.Hide()
                frmMain.Show()
            Else
                MessageBox.Show("Error user or password", "Error", MessageBoxButtons.RetryCancel, MessageBoxIcon.Error)
            End If
        Else
            MessageBox.Show("Enter Valid User Name and password", "Required Info", MessageBoxButtons.RetryCancel, MessageBoxIcon.Error)

        End If
    End Sub
End Class
```

Modify frmMain by adding four button

Control	Properties	Value
Form	Text Name	Main frmMain
Button1	Name Text Image ImageAlign FlatStyle	btnUser User MiddleLeft Flat
Button2	Name Text Image ImageAlign FlatStyle	btnStudent Student MiddleLeft Flat

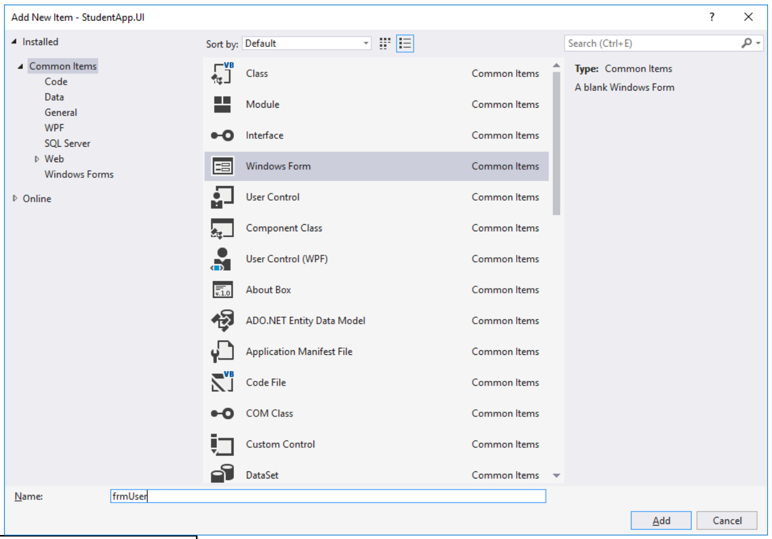


Button3	Name Text Image ImageAlign FlatStyle	btnDepartment Department MiddleLeft Flat

So the code for user button will be:

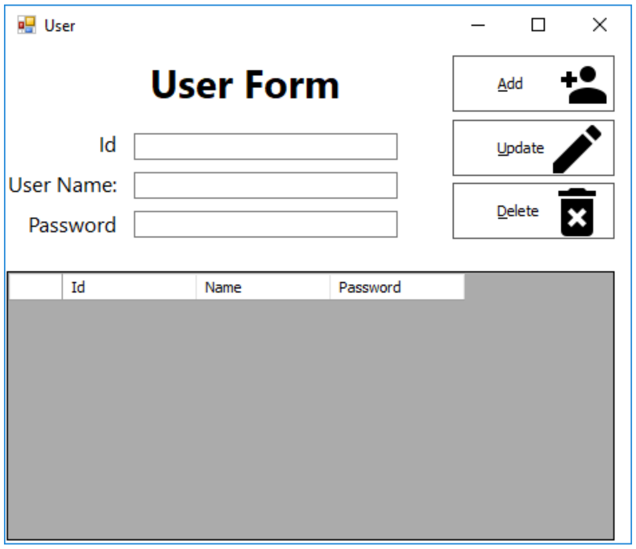
```
Private Sub btnUser_Click(sender As Object, e As EventArgs) Handles btnUser.Click
        frmUser.Show()
End Sub
```

Add new form



Name: frmUser

Control	Properties	Value
Form	Text Name	User frmUser
Label	Text	User Form
Button1	Name Text Image ImageAlign FlatStyle	btnAdd &Add MiddleLeft Flat
Button2	Name Text Image ImageAlign FlatStyle	btnUpdate &Update MiddleLeft Flat
Button3	Name Text Image ImageAlign FlatStyle	btnDelete &Delete MiddleLeft Flat
Button4	Name Text Image ImageAlign	btnExit Exit MiddleLeft



	FlatStyle	Flat
Label 1	Text Font	Id Segoe UI, 12pt
Label 2	Text Font	User Name Segoe UI, 12pt
Label 3	Text Font	Password Segoe UI, 12pt
Text Box	(Name)	txtId
Text Box	(Name)	txtUserName
Text Box	(Name)	txtPassword
DataGridView	(Name)	dgvUser

Double click on form and add code to load DataGridView

```
Private Sub frmUser_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim userRepository As New UserRepository
        dgvUser.DataSource=userRepository.GetUser()
End Sub
```

Double click on Add button and add code:

```
Private Sub btnAdd_Click(sender As Object, e As EventArgs) Handles btnAdd.Click
        Dim userRepository As New UserRepository

        If Not txtUserName.Text = Nothing and not txtPassword.Text =Nothing Then
            Dim usr As New AppUser
            usr.Name=txtUserName.Text
            usr.Password=txtPassword.Text
           If  userRepository.Add(usr) >0 Then
               MessageBox.Show("User Add Successfully")
              dgvUser.DataSource=userRepository.GetUser()

           End If
        End If
    End Sub
```

Double click on Update button and add code:

```
Private Sub btnUpdate_Click(sender As Object, e As EventArgs) Handles btnUpdate.Click
        Dim userRepository As New UserRepository

        If Not txtId.Text = Nothing and Not txtUserName.Text = Nothing and not
txtPassword.Text =Nothing Then
            Dim usr As New AppUser
            usr.Id=txtId.Text
            usr.Name=txtUserName.Text
            usr.Password=txtPassword.Text
            If  userRepository.Update(usr) >0 Then
                MessageBox.Show("User Update Successfully")
                dgvUser.DataSource=userRepository.GetUser()
            End If
        End If
    End Sub
```