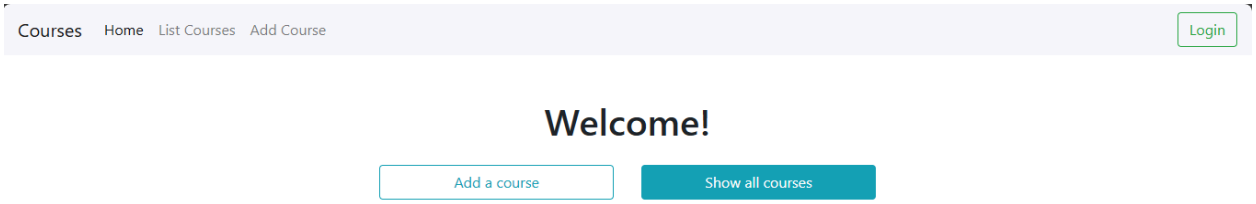
CSE 3100: Web Programming Laboratory

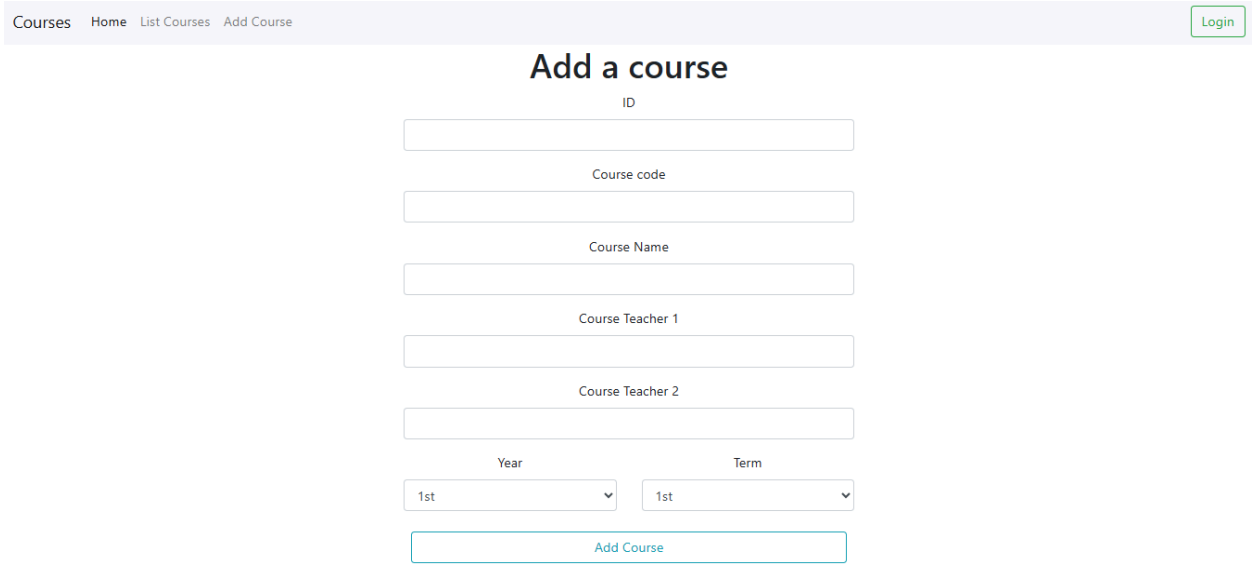
Lab 6: Database Management, Sessions, Cookies in ASP.NET

Subah Nawar
Lecturer,
Dept of CSE, KUET
Email: nawar@cse.kuet.ac.bd

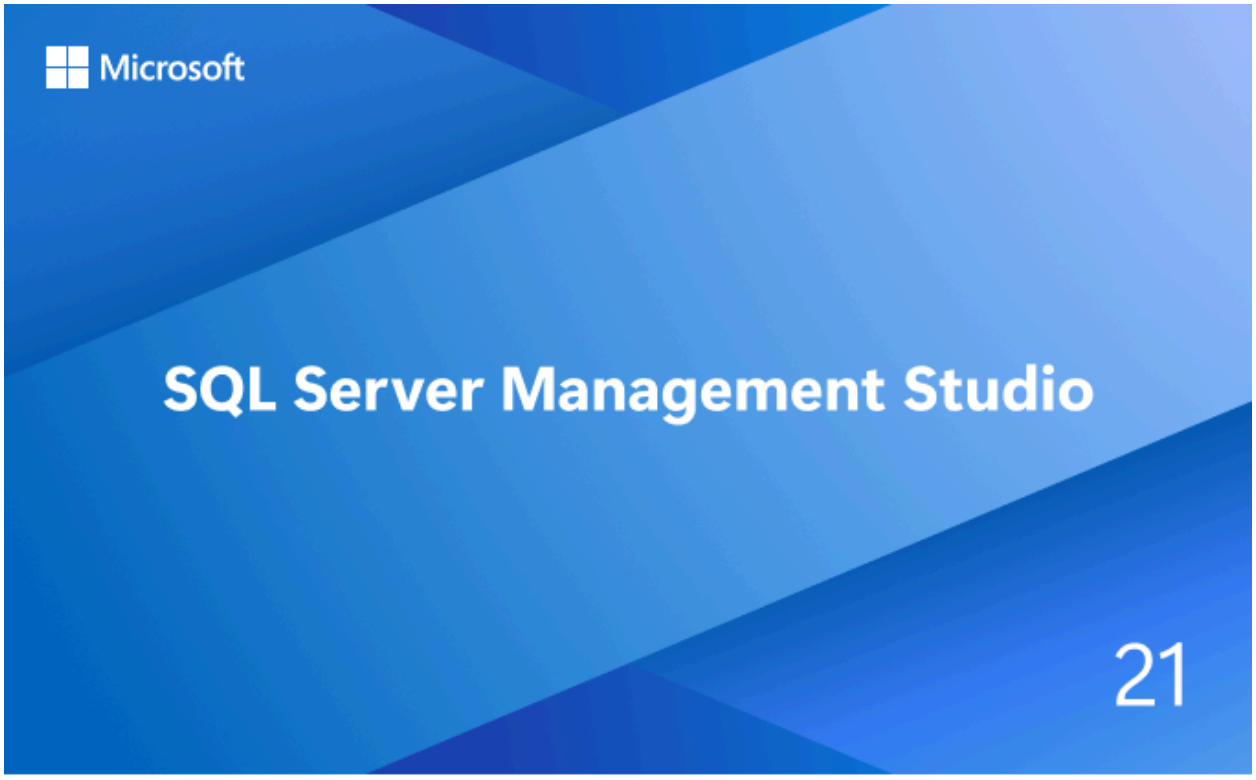
Kazi Saeed Alam
Assistant Professor,
Dept of CSE, KUET
Email: saeed.alam@cse.kuet.ac.bd

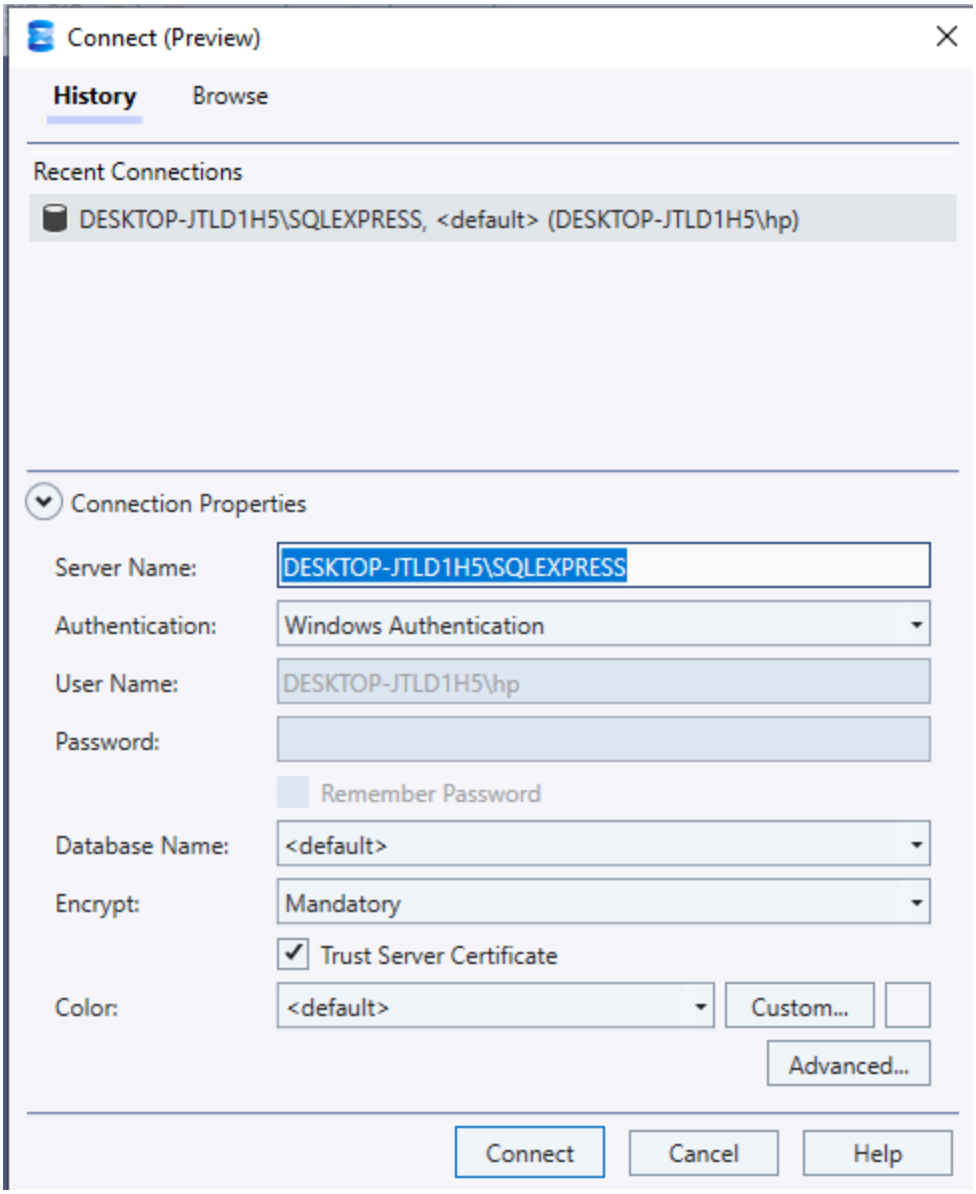
Home Page and Add a course Form

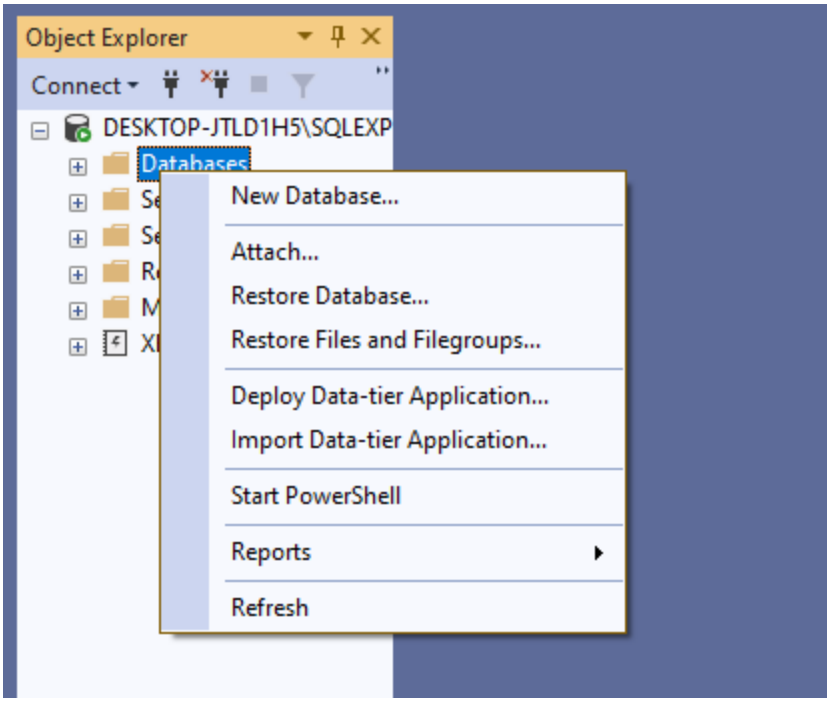


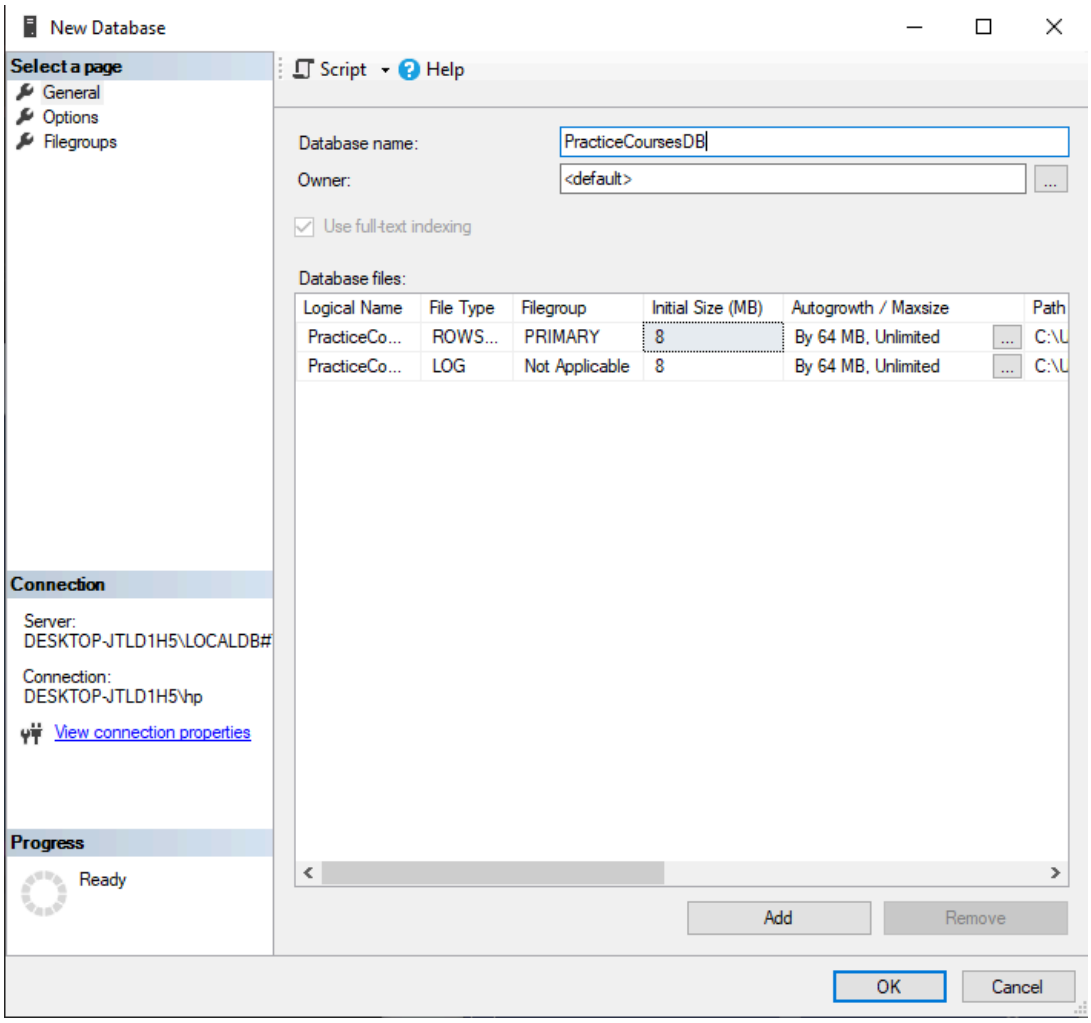


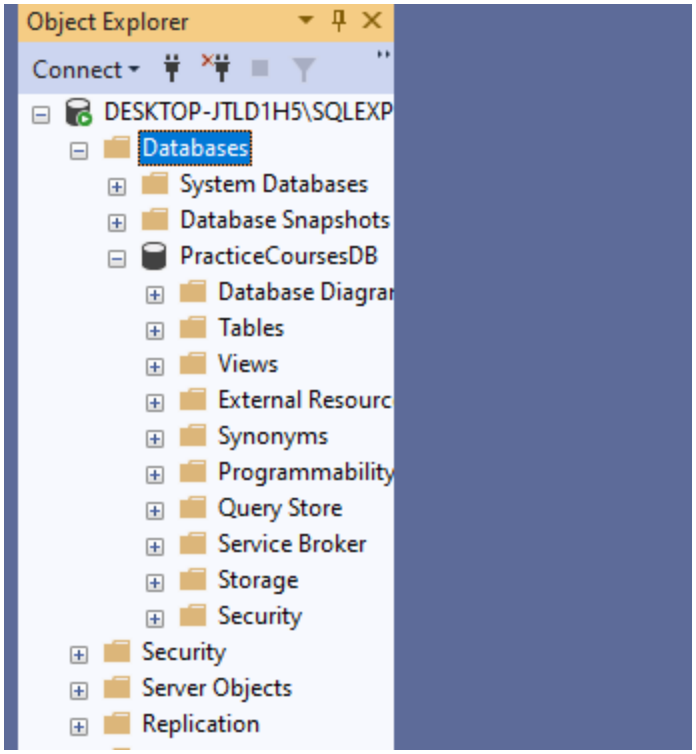
Step 1: Database Creation

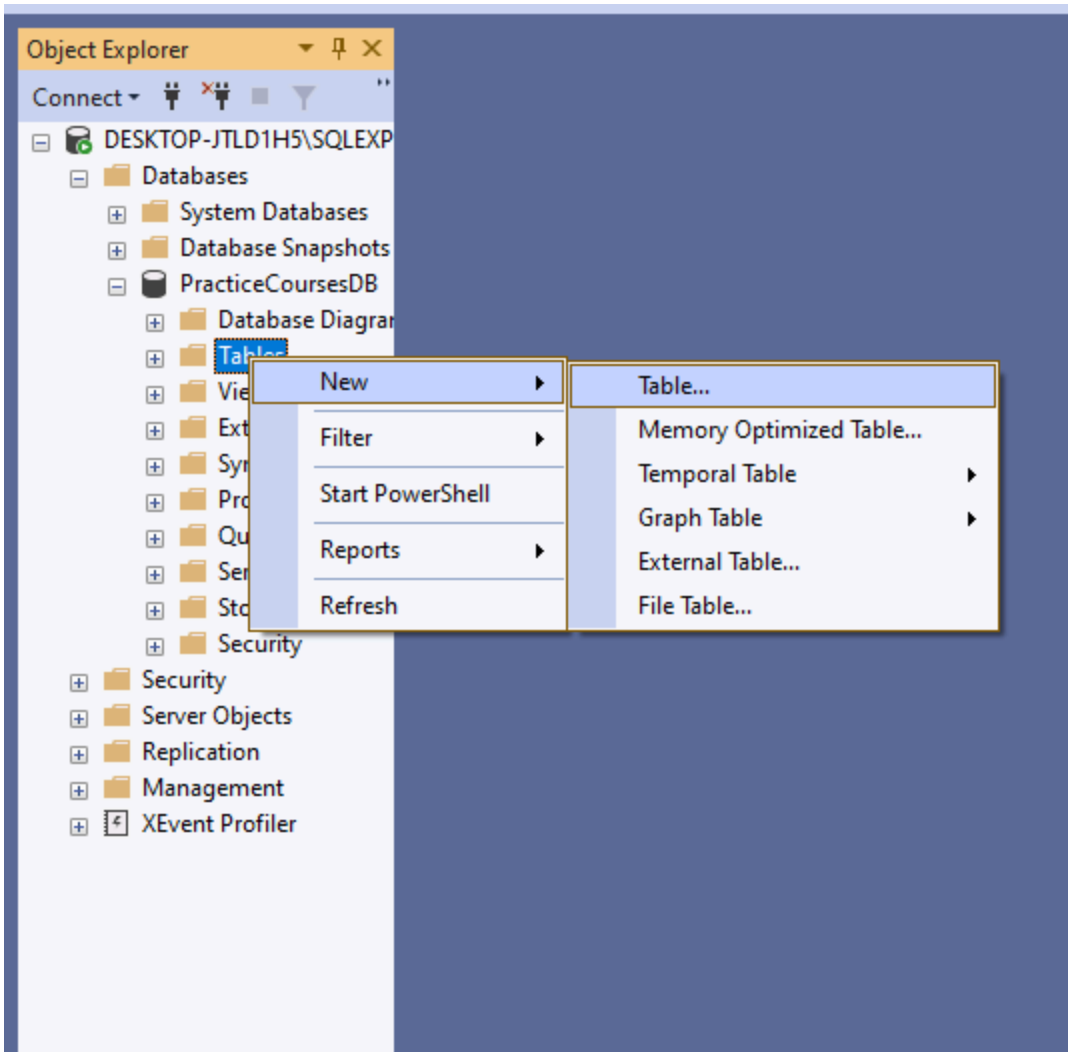

Microsoft
SQL Server Management Studio
21


Connect (Preview)
History
Browse
Recent Connections
DESKTOP-JTLD1H5\SQLEXPRESS, <default> (DESKTOP-JTLD1H5\hp)
Connection Properties
Server Name:
DESKTOP-JTLD1H5\SQLEXPRESS
Authentication:
Windows Authentication
User Name:
DESKTOP-JTLD1H5\hp
Password:
Remember Password
Database Name:
<default>
Encrypt:
Mandatory
Trust Server Certificate
Color:
<default>
Custom...
Advanced...
Connect
Cancel
Help


Object Explorer
Connect
DESKTOP-JTLD1H5\SQLEXP
Databases
New Database...
Attach...
Restore Database...
Restore Files and Filegroups...
Deploy Data-tier Application...
Import Data-tier Application...
Start PowerShell
Reports
Refresh


New Database
Select a page
General
Options
Filegroups
Script
Help
Database name:
PracticeCoursesDB
Owner:
<default>
Use full-text indexing
Database files:
Logical Name
File Type
Filegroup
Initial Size (MB)
Autogrowth / Maxsize
Path
PracticeCo...
ROWS...
PRIMARY
8
By 64 MB, Unlimited
C:\U
PracticeCo...
LOG
Not Applicable
8
By 64 MB, Unlimited
C:\U
Connection
Server:
DESKTOP-JTLD1H5\LOCALDB#
Connection:
DESKTOP-JTLD1H5\hp
View connection properties
Progress
Ready
Add
Remove
OK
Cancel


Object Explorer
Connect
DESKTOP-JTLD1H5\SQLEXP
Databases
System Databases
Database Snapshots
PracticeCoursesDB
Database Diagrar
Tables
Views
External Resourc
Synonyms
Programmability
Query Store
Service Broker
Storage
Security
Security
Server Objects
Replication

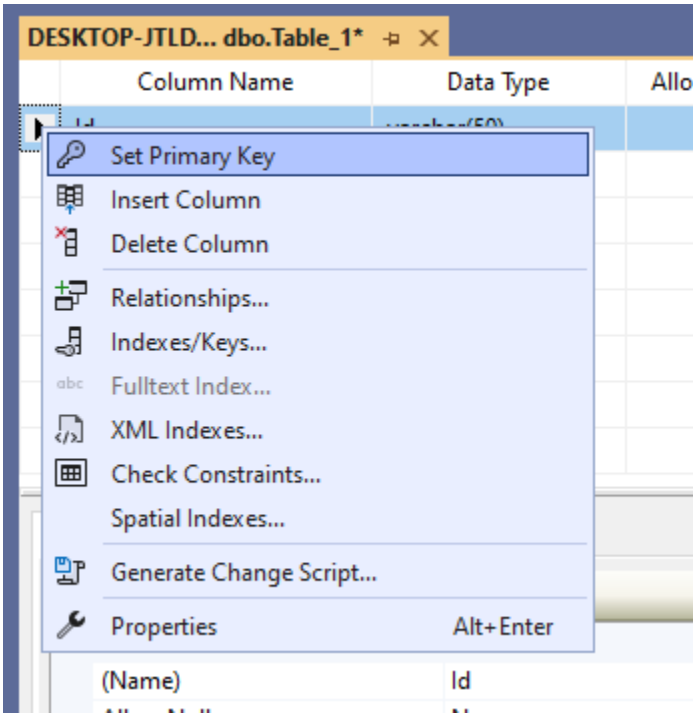

Object Explorer
Connect
DESKTOP-JTLD1H5\SQLEXP
Databases
System Databases
Database Snapshots
PracticeCoursesDB
Database Diagrar
Tables
Vie
Ext
Syn
Pro
Qu
Ser
Sto
Security
Security
Server Objects
Replication
Management
XEvent Profiler
New
Filter
Start PowerShell
Reports
Refresh
Table...
Memory Optimized Table...
Temporal Table
Graph Table
External Table...
File Table...

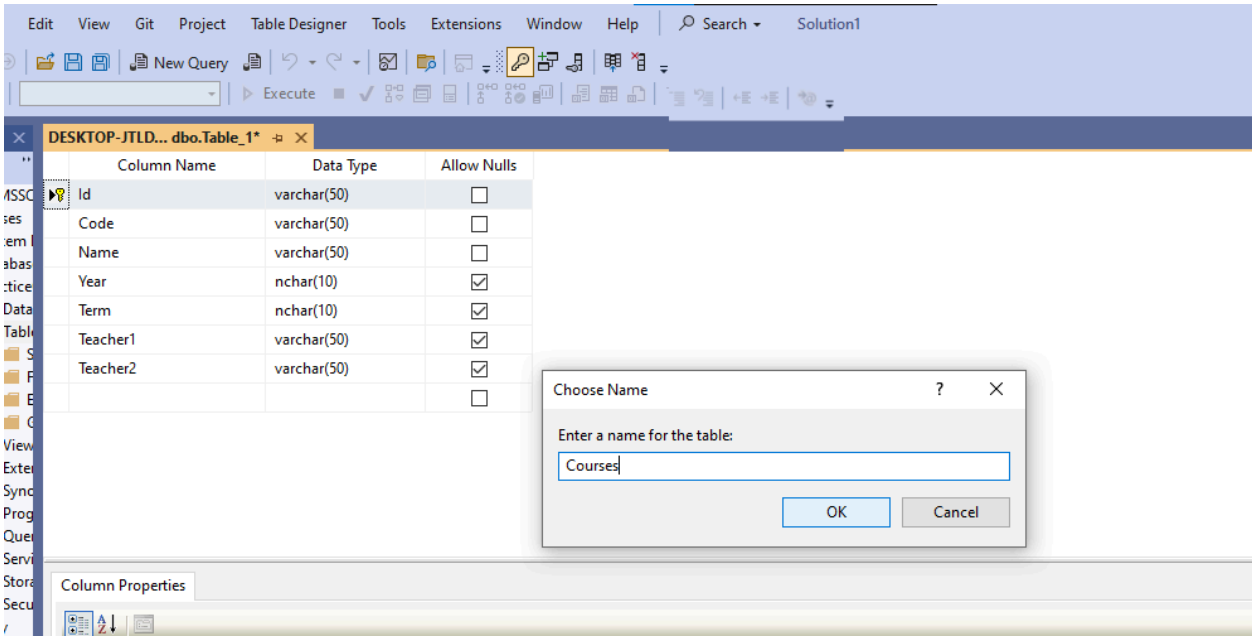
Column Name	Data Type	Allow Nulls
Id	varchar(50)	☐
Code	varchar(50)	☐
Name	varchar(50)	☐
Year	nchar(10)	☑
Term	nchar(10)	☑
Teacher1	varchar(50)	☑
Teacher2	varchar(50)	☑
		☐

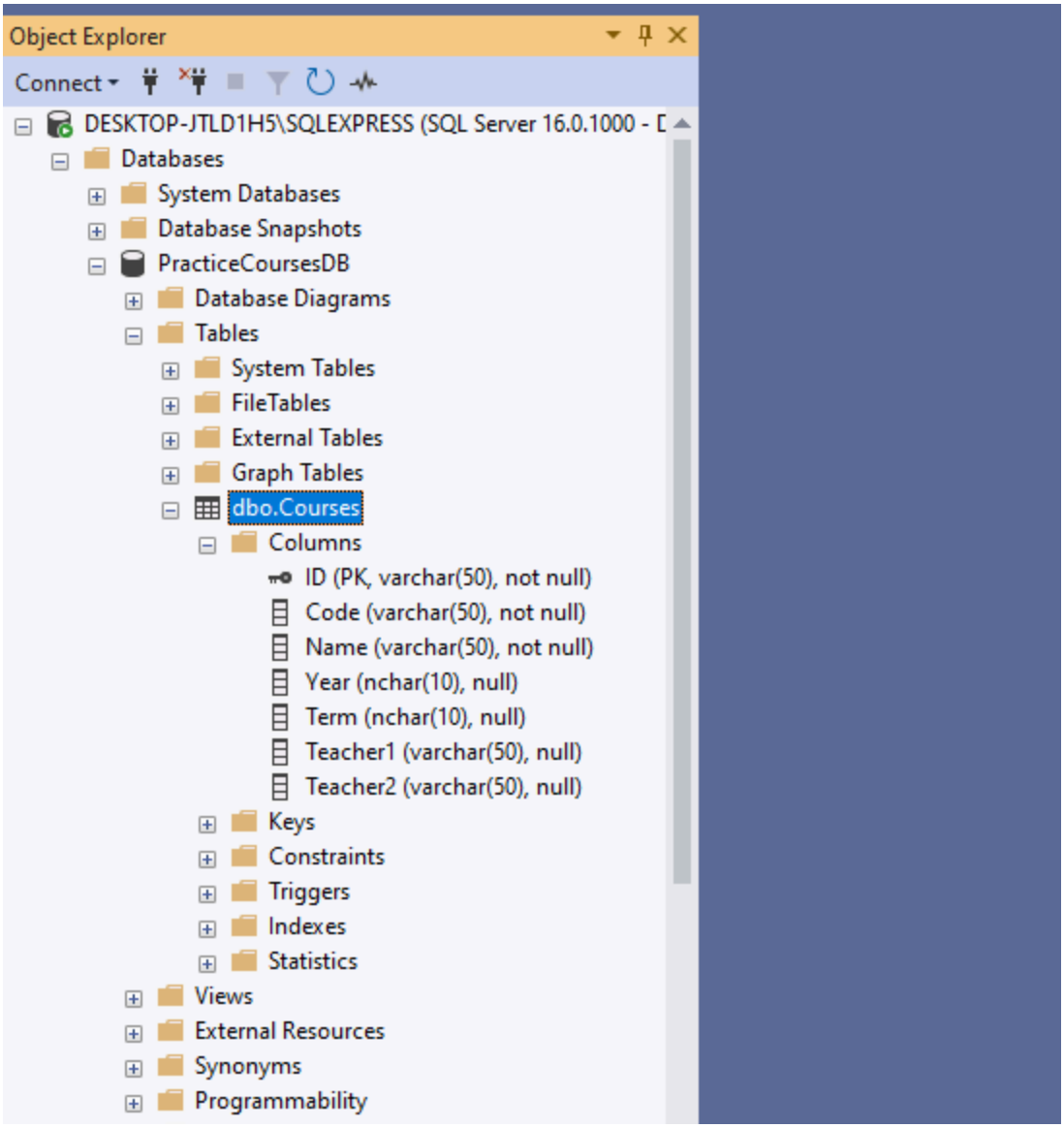
Column Properties

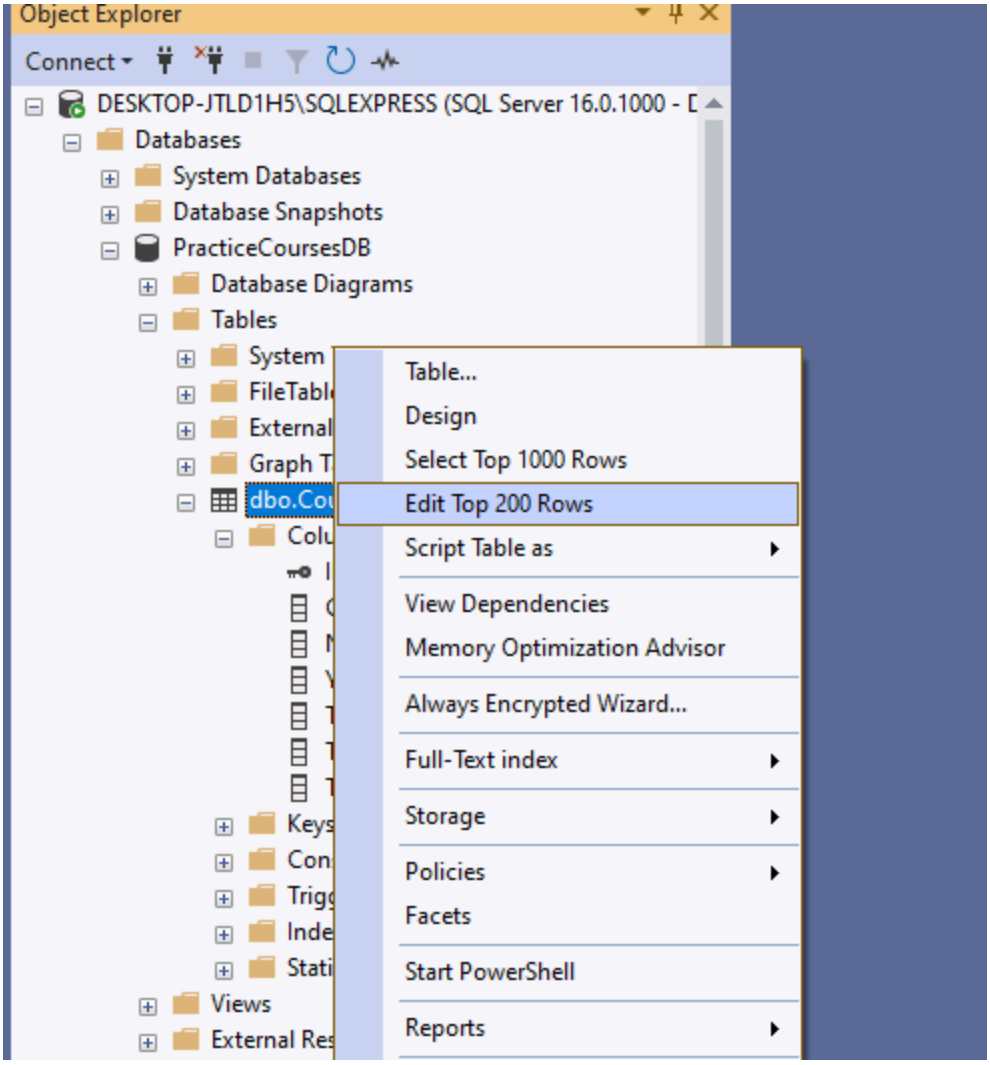
(General)	
(Name)	Id
Allow Nulls	No
Data Type	varchar
Default Value or Binding	
Length	50

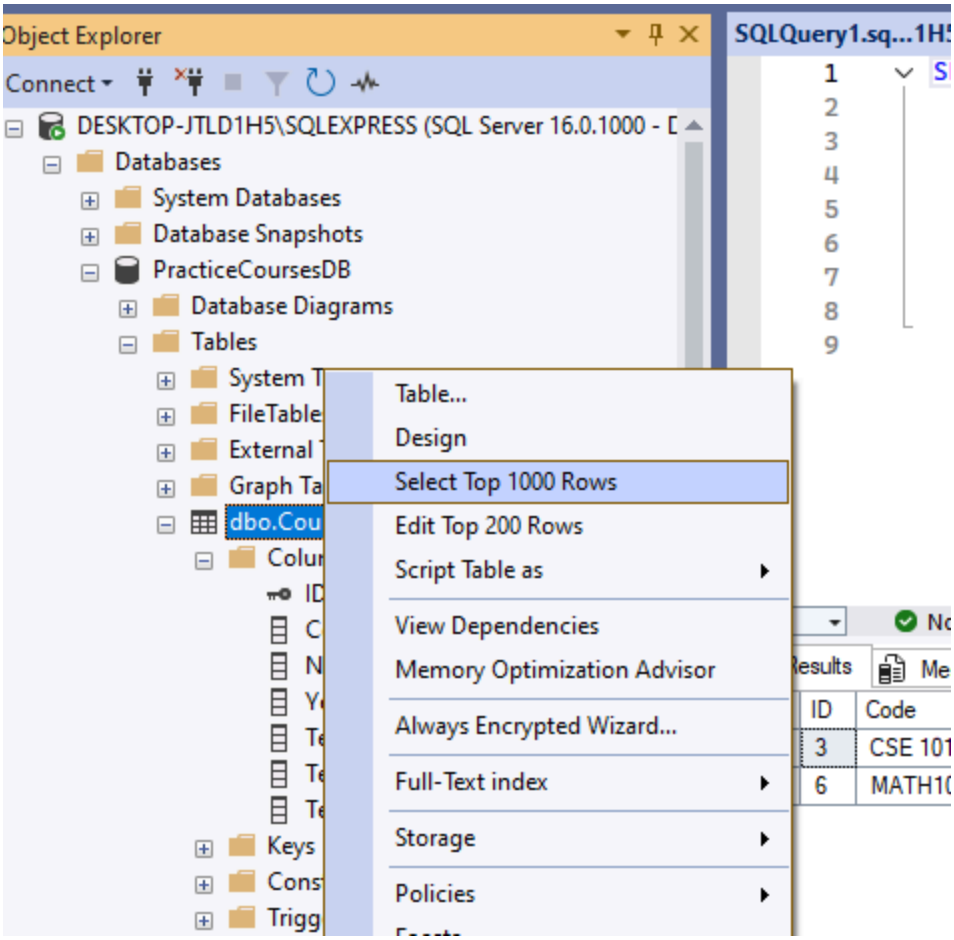
(General)




Edit View Git Project Table Designer Tools Extensions Window Help
Search
Solution1
New Query
Execute
DESKTOP-JTLD... dbo.Table_1*
Column Name
Data Type
Allow Nulls
Id varchar(50)
Code varchar(50)
Name varchar(50)
Year nchar(10)
Term nchar(10)
Teacher1 varchar(50)
Teacher2 varchar(50)
Choose Name
Enter a name for the table:
Courses
OK
Cancel
Column Properties


Object Explorer
Connect
DESKTOP-JTLD1H5\SQLEXPRESS (SQL Server 16.0.1000 - D
Databases
System Databases
Database Snapshots
PracticeCoursesDB
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Courses
Columns
ID (PK, varchar(50), not null)
Code (varchar(50), not null)
Name (varchar(50), not null)
Year (nchar(10), null)
Term (nchar(10), null)
Teacher1 (varchar(50), null)
Teacher2 (varchar(50), null)
Keys
Constraints
Triggers
Indexes
Statistics
Views
External Resources
Synonyms
Programmability


Object Explorer
Connect
DESKTOP-JTLD1H5\SQLEXPRESS (SQL Server 16.0.1000 - D
Databases
System Databases
Database Snapshots
PracticeCoursesDB
Database Diagrams
Tables
System
FileTabl
External
Graph T
dbo.Co
Colu
Keys
Con
Trigg
Inde
Stati
Views
External Res
Table...
Design
Select Top 1000 Rows
Edit Top 200 Rows
Script Table as
View Dependencies
Memory Optimization Advisor
Always Encrypted Wizard...
Full-Text index
Storage
Policies
Facets
Start PowerShell
Reports



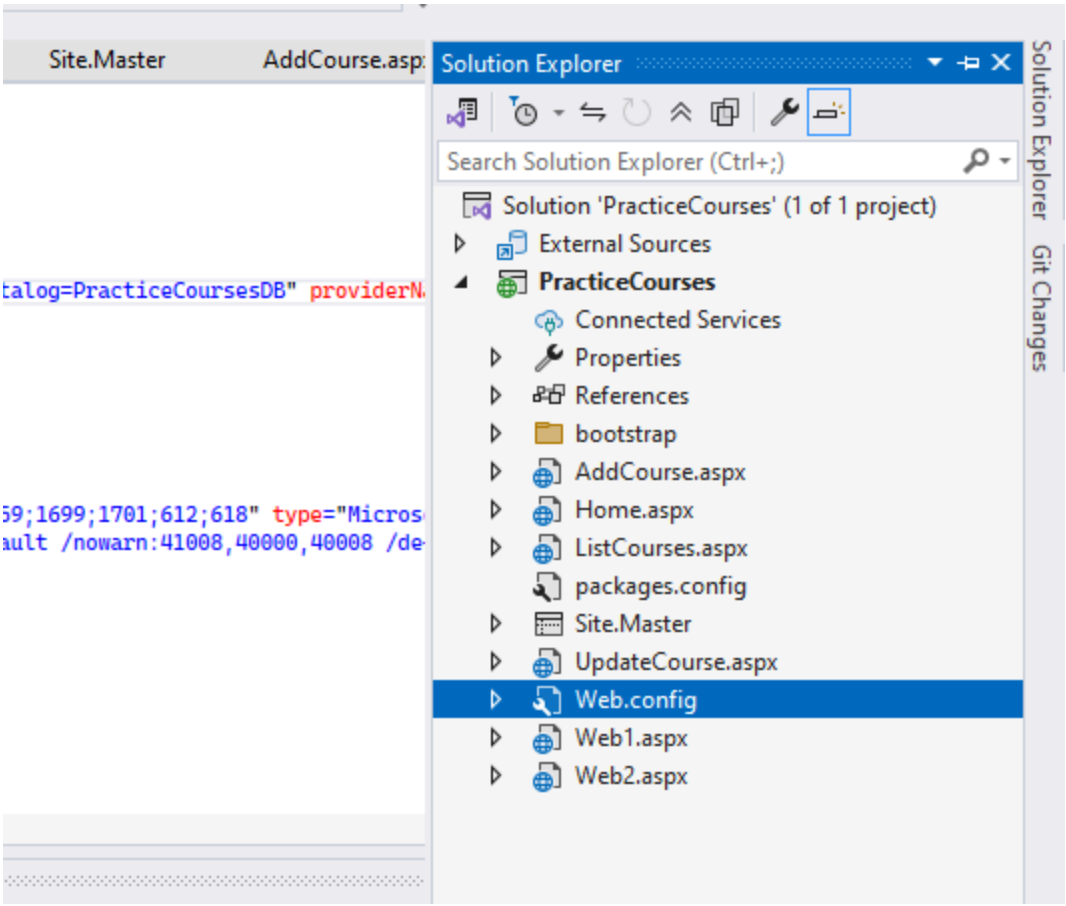
Step 2: Database Connection

After opening the web.config file in application , add sample db connection in connectionStrings section inside <configuration> </configuration> in web.config like this:

<connectionStrings>

<add name="yourconnectinstringName" connectionString="Data Source= **YourDatabaseServerName**; Integrated Security=true;Initial Catalog= **YourDatabaseName**; " providerName="System.Data.SqlClient" />

</connectionStrings>



Now we need to connect this **connection string** to our webform.aspx.cs files such as AddCourse.aspx.cs, ListCourses.cs, UpdateCourses.cs (code behind) in order to access the desired database. Now, write the code to get the connection string from web.config file in our codebehind file. Add the following namespace in codebehind file if it is not added automatically.

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace PracticeCourses
{
```

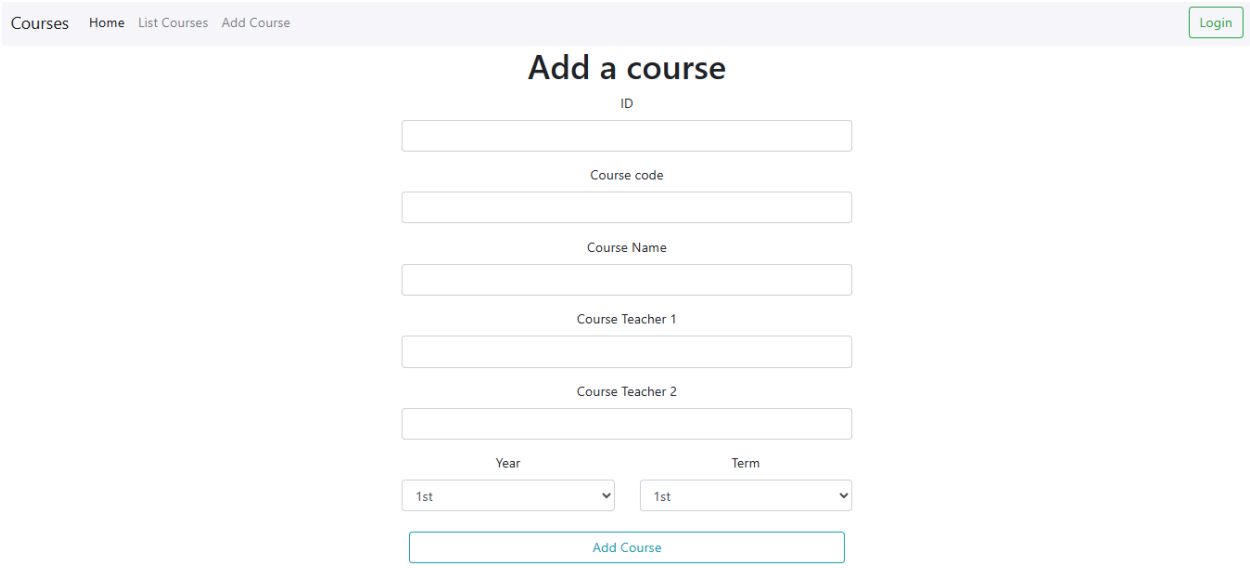
This namespace is used to get configuration section details from web.config file.

To establish the connection we will write the following codes:

```
using System;
using System.Data.SqlClient;
using System.Configuration;
public partial class _Default: System.Web.UI.Page {
  protected void Page_Load(object sender, EventArgs e) {
    //Get connection string from web.config file
    string strcon = ConfigurationManager.ConnectionStrings["dbconnection"].ConnectionString;
    //create new sqlconnection and connection to database by using connection string from web.config file
    SqlConnection con = new SqlConnection(strcon);
    con.Open();
  }
}
```

Whenever we want to execute a sql command , we will need to get this connection string and establish a connection. For details:
https://www.c-sharpcorner.com/code/3379/connection-strings-in-web-config-file-using-asp-net.aspx

Step 3: Insert Data



- Open the AddCourse.aspx file and add necessary variable names, onclick functions such as AddCourseButton. Then we need to add the c# code i.e definition for AddCourseButton function in [AddCourse.aspx.cs]() file

- Firstly Establish database connection,

```
//Get connection string from web.config file
string strcon = ConfigurationManager.ConnectionStrings["dbconnection"].ConnectionString;
//create new sqlconnection and connection to database by using connection string from web.config file
SqlConnection con = new SqlConnection(strcon);
con.Open();
```

- Write your data insert query in the following way inside the AddCourseButton function definition aFter establishing database connection:

```
// Query to insert
SqlCommand cmd = new SqlCommand("INSERT INTO Courses ([Id], [Code], [Name], " +
    "[Teacher1], [Teacher2], [Year], [Term]) VALUES(@Id, @Code, @Name, @Teacher1, @Teacher2, @Year, @Term)", con);

cmd.Parameters.AddWithValue("@Id", CourseIdTextBox.Text.Trim());
cmd.Parameters.AddWithValue("@Code", CourseCodeTextBox.Text.Trim());
cmd.Parameters.AddWithValue("@Name", CourseNameTextBox.Text.Trim());
cmd.Parameters.AddWithValue("@Teacher1", CourseTeacher1TextBox.Text.Trim());
cmd.Parameters.AddWithValue("@Teacher2", CourseTeacher2TextBox.Text.Trim());
cmd.Parameters.AddWithValue("@Year", CourseYearDropDownList.SelectedValue);
cmd.Parameters.AddWithValue("@Term", CourseTermDropDownList.SelectedValue);

cmd.ExecuteNonQuery();

// Close the connection
con.Close();
```

Here, AddWithValue maps a parameter placeholder (@Id) to the actual input value.

- You can write a response message for each successful insert operation:

```
// Success message in alerts
Response.Write("<script>alert('Course added!');</script>");
```

- Always close the DB connection after the operation using con.close()

Step 4: Read/Show Data

- Open ListCourses.aspx file , here we have used GridView and DataTable. (For details: https://www.youtube.com/watch?v=H5_6qZi7ud)
- Establish database connection in c# code file in previous manner in side Page_Load as we want to see the All Courses Table when the page loads:

```
protected void Page_Load(object sender, EventArgs e)
{
    //Get connection string from web.config file
    string strcon = ConfigurationManager.ConnectionStrings["dbconnection"].ConnectionString;
    //create new sqlconnection and connection to database by using connection string from web.config file
    SqlConnection con = new SqlConnection(strcon);
    con.Open();
    SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM COURSES", con);
    DataTable dt = new DataTable();
    da.Fill(dt);

    CoursesGridView.DataSource = dt;
    CoursesGridView.DataBind();

    con.Close();
}
```

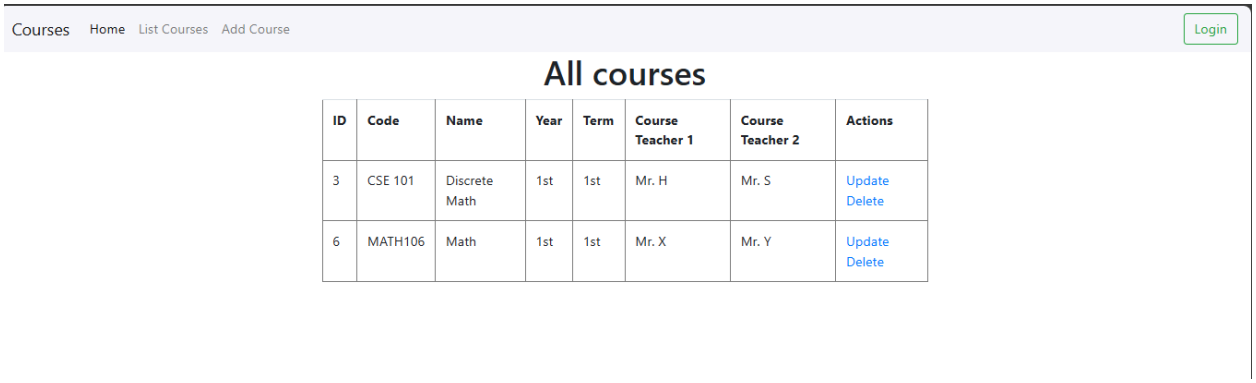
- Add following namespace in ListCourses.aspx.cs c-sharp file if it is not automatically added while coding:

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

- Now we will use SqlDataAdapter. SqlDataAdapter acts like a bridge between the database and our program. Then run the SQL query SELECT * FROM COURSES. Then close the connection after query execution.

```
0 references
protected void Page_Load(object sender, EventArgs e)
{
    //Get connection string from web.config file
    string strcon = ConfigurationManager.ConnectionStrings["dbconnection"].ConnectionString;
    //create new sqlconnection and connection to database by using connection string from web.config file
    SqlConnection con = new SqlConnection(strcon);
    con.Open();
    SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM COURSES", con);
    DataTable dt = new DataTable();
    da.Fill(dt);

    CoursesGridView.DataSource = dt;
    CoursesGridView.DataBind();

    con.Close();
}
```

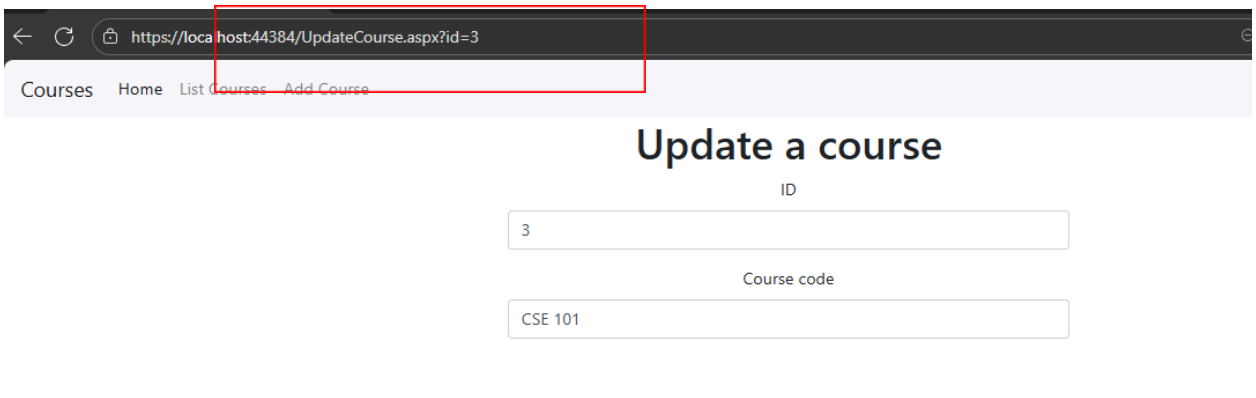
The table should have the following appearance:

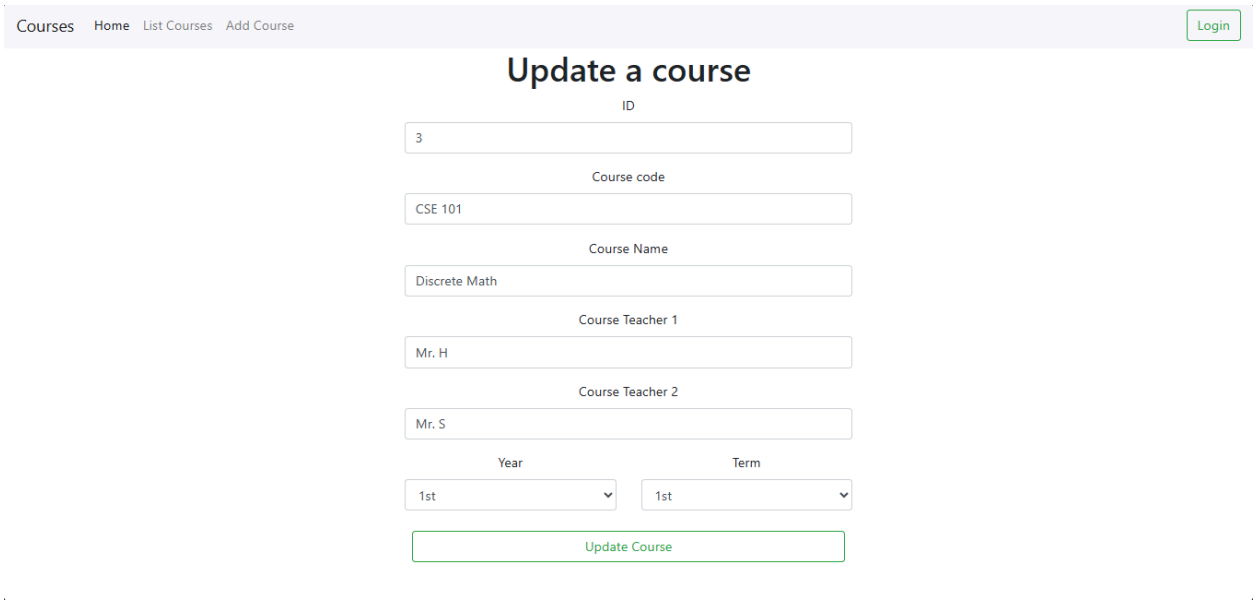


ID	Code	Name	Year	Term	Course Teacher 1	Course Teacher 2	Actions
3	CSE 101	Discrete Math	1st	1st	Mr. H	Mr. S	Update Delete
6	MATH106	Math	1st	1st	Mr. X	Mr. Y	Update Delete

Step 5: Update Data

1. When the UpdateCourse.aspx page loads, it needs to fetch a course by its `id` from the database. So we will use QueryString for sending the id as a parameter in the URL used for querystring. We will Response.direct the querystring from ListCourses.aspx to UpdateCourse.aspx. The Design of UpdateCourse.aspx is similar to AddCourse.aspx but the difference will be in the form fields. The form fields for UpdateCourse.aspx will be filled with already existing values in the database.





2. In the design of ListCourses.aspx we used GridView and ItemTemplate. In Itemtemplate we added two linkbuttons- update and delete. Now we need to add attribute OnRowCommand in the GridView to detect which row and which button has been pressed. We also need to command argument in Update linkbutton of ListCourses.aspx to send{id} as argument of querystring in order to update the desired course uniquely identified by its id:

```
<asp:GridView ID="CoursesGridView" CssClass="table" runat="server" AutoGenerateColumns="false" OnRowCommand="CoursesGridView_RowCommand">
    <Columns>
        <asp:BoundField DataField="Id" HeaderText="ID" />
        <asp:BoundField DataField="Code" HeaderText="Code" />
        <asp:BoundField DataField="Name" HeaderText="Name" />
        <asp:BoundField DataField="Year" HeaderText="Year" />
        <asp:BoundField DataField="Term" HeaderText="Term" />
        <asp:BoundField DataField="Teacher1" HeaderText="Course Teacher 1" />
        <asp:BoundField DataField="Teacher2" HeaderText="Course Teacher 2" />
        <asp:TemplateField HeaderText="Actions">
            <ItemTemplate>
                <asp:LinkButton ID="UpdateLinkButton" CommandName="upd" CommandArgument='<%#Eval("Id") %>'
                    runat="server">Update</asp:LinkButton>
                <asp:LinkButton ID="DeleteLinkButton" CommandName="del" CommandArgument='<%#Eval("Id") %>'
                    onclientclick="return confirm('Are you sure to delete?');" runat="server">Delete</asp:LinkButton>
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

```
<asp:TemplateField HeaderText="Actions">
    <ItemTemplate>
        <asp:LinkButton ID="UpdateLinkButton" CommandName="upd" CommandArgument='<%#Eval("Id") %>'
            runat="server">Update</asp:LinkButton>
        <asp:LinkButton ID="DeleteLinkButton" CommandName="del" CommandArgument='<%#Eval("Id") %>'
            onclientclick="return confirm('Are you sure to delete?');" runat="server">Delete</asp:LinkB
    </ItemTemplate>
</asp:TemplateField>
```

Now add **CoursesGridView_RowCommand Method** for Update and Delete. Remember to add the argument **GridViewCommandEventArgs**. In case of Update, we will simply redirect it to UpdateCourse.aspx with the **id of the row. This id will be read by** UpdateCourse.aspx.cs **using a Request.QueryString["id"] .**

```
protected void CoursesGridView_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName == "upd")
    {
        Response.Redirect(string.Format("~/UpdateCourse.aspx?id={0}", e.CommandArgument));
    }
    else if (e.CommandName == "del")
    {
        try
        {
            //create new sqlconnection and connection to database by using connection string from web.config file
            SqlConnection con = new SqlConnection(strcon);

            if (con.State == ConnectionState.Closed)
            {
                con.Open();
            }

            SqlCommand cmd = new SqlCommand("DELETE FROM Courses WHERE Id=@Id", con);
            cmd.Parameters.AddWithValue("@Id", e.CommandArgument.ToString().Trim());
            cmd.ExecuteNonQuery();

            // Close the connection
            con.Close();

            Response.Redirect("~/ListCourses.aspx");
        }
        catch (Exception ex)
        {
            // Error message in alerts
            Response.Write("<script>alert('Error: " + ex.Message + "');</script>");
        }
    }
}
```

3. In UpdateCourse.aspx.cs file establish connection

```
//Get connection string from web.config file
string strcon = ConfigurationManager.ConnectionStrings["dbconnection"].ConnectionString;
```

```
//create new sqlconnection and connection to database by using connection string from web.config file
SqlConnection con = new SqlConnection(strcon);
```

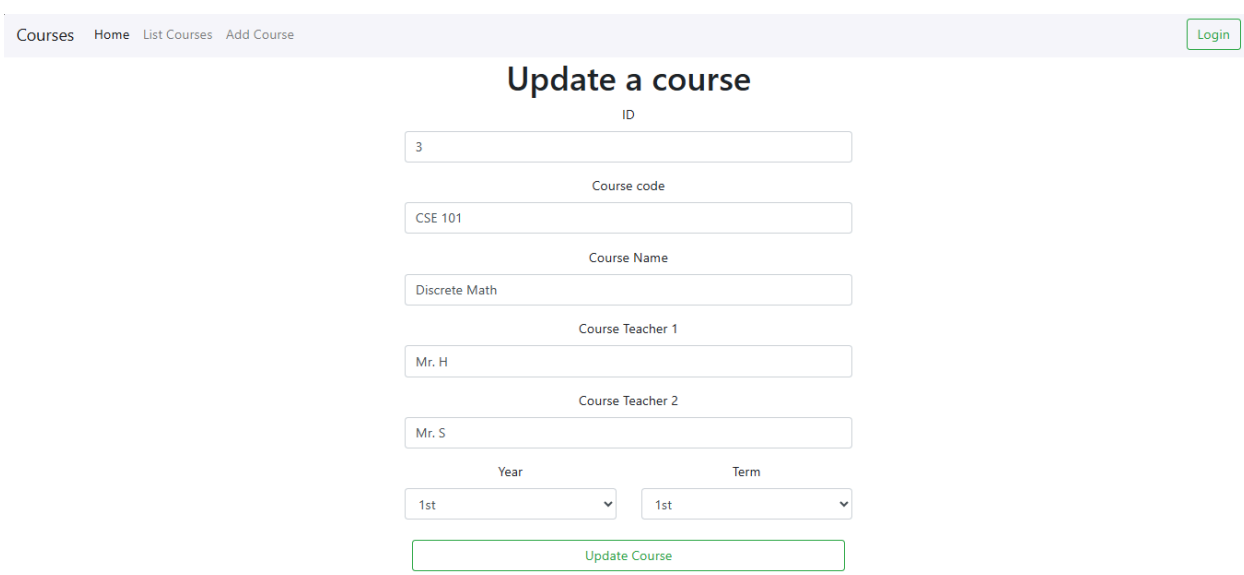
4. On Page_Load , the form will be initially filled with existing values. IsPostBack gets value or whether the page is loading for **the first time**. This code runs **only the first time** the page loads (not on button click) and gets the course `id` from the URL.
5.

```
0 references
protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        string id = Request.QueryString["id"].Trim();
        Console.WriteLine(id);
```

6. Now we will read from the database and fill the form fields. Now the user sees the current course details in textboxes/dropdowns before updating.

```
//create new sqlconnection and connection to database by using connection string from web.config file
SqlConnection con = new SqlConnection(strcon);

if (con.State == ConnectionState.Closed)
{
    con.Open();
}
```

```
SqlCommand cmd = new SqlCommand(String.Format("SELECT * FROM Courses WHERE id={0}", id), con);
SqlDataReader sdr = cmd.ExecuteReader();

if (sdr.HasRows)
{
    sdr.Read();

    CourseIdTextBox.Text = sdr.GetValue(0).ToString();
    CourseCodeTextBox.Text = sdr.GetValue(1).ToString();
    CourseNameTextBox.Text = sdr.GetValue(2).ToString();
    CourseYearDropDownList.SelectedValue = sdr.GetValue(3).ToString().Trim();
    CourseTermDropDownList.SelectedValue = sdr.GetValue(4).ToString().Trim();
    CourseTeacher1TextBox.Text = sdr.GetValue(5).ToString();
    CourseTeacher2TextBox.Text = sdr.GetValue(6).ToString();


}

// Close the connection
con.Close();
```



7. Now, we will define the Update Button (CourseAddButton_Click) function, which is similar to the AddCourse.aspx.cs

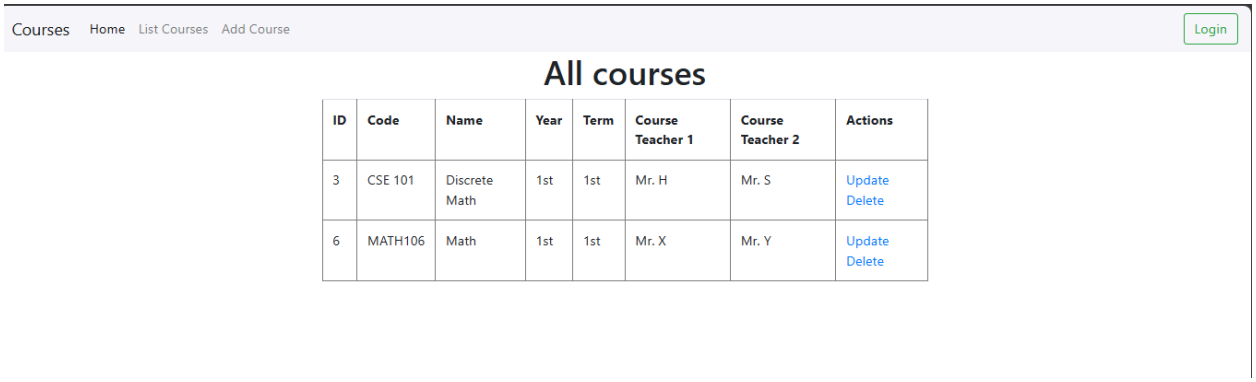
```
{
    //create new sqlconnection and connection to database by using connection string from web.config file
    SqlConnection con = new SqlConnection(strcon);

    if (con.State == ConnectionState.Closed)
    {
        con.Open();
    }

    SqlCommand cmd = new SqlCommand("UPDATE Courses SET Id=@Id, Code=@Code, Name=@Name, Teacher1=@Teacher1, " +
        "Teacher2=@Teacher2, Year=@Year, Term=@Term WHERE Id=@Id", con);

    cmd.Parameters.AddWithValue("@Id", CourseIdTextBox.Text.Trim());
    cmd.Parameters.AddWithValue("@Code", CourseCodeTextBox.Text.Trim());
    cmd.Parameters.AddWithValue("@Name", CourseNameTextBox.Text.Trim());
    cmd.Parameters.AddWithValue("@Teacher1", CourseTeacher1TextBox.Text.Trim());
    cmd.Parameters.AddWithValue("@Teacher2", CourseTeacher2TextBox.Text.Trim());
    cmd.Parameters.AddWithValue("@Year", CourseYearDropDownList.SelectedValue);
    cmd.Parameters.AddWithValue("@Term", CourseTermDropDownList.SelectedValue);

    cmd.ExecuteNonQuery();

    // Close the connection
    con.Close();

    // Success message in alerts
    Response.Write("<script>alert('Course updated!');</script>");
    // Error message in alerts
    Response.Redirect("~/ListCourses.aspx");
```

Step 6: Delete Data

1. So we will firstly need to add command argument in Delete button of ListCourses.aspx and OnRowCommand in the GridView:

```
<asp:GridView ID="CoursesGridView" CssClass="table" runat="server" AutoGenerateColumns="false" OnRowCommand="CoursesGridView_RowCommand">
    <Columns>
        <asp:BoundField DataField="Id" HeaderText="ID" />
        <asp:BoundField DataField="Code" HeaderText="Code" />
        <asp:BoundField DataField="Name" HeaderText="Name" />
        <asp:BoundField DataField="Year" HeaderText="Year" />
        <asp:BoundField DataField="Term" HeaderText="Term" />
        <asp:BoundField DataField="Teacher1" HeaderText="Course Teacher 1" />
        <asp:BoundField DataField="Teacher2" HeaderText="Course Teacher 2" />
        <asp:TemplateField HeaderText="Actions">
            <ItemTemplate>
                <asp:LinkButton ID="UpdateLinkButton" CommandName="upd" CommandArgument='<%#Eval("Id") %>'
                    runat="server">Update</asp:LinkButton>
                <asp:LinkButton ID="DeleteLinkButton" CommandName="del" CommandArgument='<%#Eval("Id") %>'
                    onclientclick="return confirm('Are you sure to delete?');" runat="server">Delete</asp:LinkButton>
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

```
<asp:TemplateField HeaderText="Actions">
    <ItemTemplate>
        <asp:LinkButton ID="UpdateLinkButton" CommandName="upd" CommandArgument='<%#Eval("Id") %>'
            runat="server">Update</asp:LinkButton>
        <asp:LinkButton ID="DeleteLinkButton" CommandName="del" CommandArgument='<%#Eval("Id") %>'
            onclientclick="return confirm('Are you sure to delete?');" runat="server">Delete</asp:LinkB
    </ItemTemplate>
</asp:TemplateField>
```



ID	Code	Name	Year	Term	Course Teacher 1	Course Teacher 2	Actions
3	CSE 101	Discrete Math	1st	1st	Mr. H	Mr. S	Update Delete
6	MATH106	Math	1st	1st	Mr. X	Mr. Y	Update Delete

2. Now, we will write the code behind delete operation in the ListCourses.aspx.cs in the method CoursGridView_RowCommand

```
protected void CoursesGridView_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName == "upd")
    {
        Response.Redirect(string.Format("~/UpdateCourse.aspx?id={0}", e.CommandArgument));
    }
    else if (e.CommandName == "del")
    {
        try
        {
            //create new sqlconnection and connection to database by using connection string from web.config file
            SqlConnection con = new SqlConnection(strcon);

            if (con.State == ConnectionState.Closed)
            {
                con.Open();
            }

            SqlCommand cmd = new SqlCommand("DELETE FROM Courses WHERE Id=@Id", con);
            cmd.Parameters.AddWithValue("@Id", e.CommandArgument.ToString().Trim());
            cmd.ExecuteNonQuery();

            // Close the connection
            con.Close();

            Response.Redirect("~/ListCourses.aspx");
        }
        catch (Exception ex)
        {
            // Error message in alerts
            Response.Write("<script>alert('Error: " + ex.Message + "');</script>");
        }
    }
}
```

The try-catch block is used to handle any exception. If you want, you can skip using it

Cookies

Enter Your Name | Email

Cookies

```
0 references
protected void Button1_Click(object sender, EventArgs e)
{
    HttpCookie cookie = new HttpCookie("UserInfo");
    cookie["Username"] = TextBox1.Text;
    cookie["email"] = TextBox2.Text;

    cookie.Expires = DateTime.Now.AddDays(7); //Persistent Cookies
    Response.Cookies.Add(cookie);
    Response.Redirect("Web2.aspx");
}
```

We will use HttpCookie to set cookies for our web forms and Request.Cookies to read them.

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie cookie = Request.Cookies["UserInfo"];
    if (cookie != null)
    {
        Label1.Text = "Username: " + cookie["Username"];
        Label2.Text = "Email: " + cookie["email"];
    }
    else
    {
        Label1.Text = "No cookie found.";
        Label2.Text = "";
    }

}
```

Sessions

```
0 references
protected void Button1_Click(object sender, EventArgs e)
{

    Session["Username"] = TextBox1.Text;
    Session["email"] = TextBox2.Text;

    |
    Response.Redirect("Web2.aspx");
}
```

H.W #1: ***Now based on the registration system, create a login page which will check username and password stored in the database and allows user to log in or not. If logged in, it will show a welcome page.***

H.W #2: ***Apply Session and Cookies functionalities in your login system. What differences can you observe for session vs cookies.***