2.1 Introduction about ADO.NET

- ADO is a Microsoft technology
- ADO stands for ActiveX Data Objects
- ADO is a Microsoft Active-X component
- ADO is automatically installed with Microsoft IIS
- ADO is a programming interface to access data in a database
- ADO.NET has the ability to separate data access mechanisms, data manipulation mechanisms and data connectivity mechanisms.
- ADO.NET is a set of classes that allow application to read and write information in database.
- ADO.NET can be used by any .NET language.
- We need to add System. Data namespace for work with ADO.NET.
- Ado.Net is a technology which works between access database Frontend Application. It is used to access database.

- **ADO.NET Object Model**

  - ADO.NET is an object-oriented set of libraries that allows you to interact with data sources.
  - The data source is a database, but it could also be a text file, an Excel spread- sheet, or an XML file.
  - There are many different types of databases available such as MicrosoftSQLSever, MicrosoftAccess,Oracle,BorlandInterbase,IBMDB2 etc.

- **Connected & Disconnected Data(Architecture)**

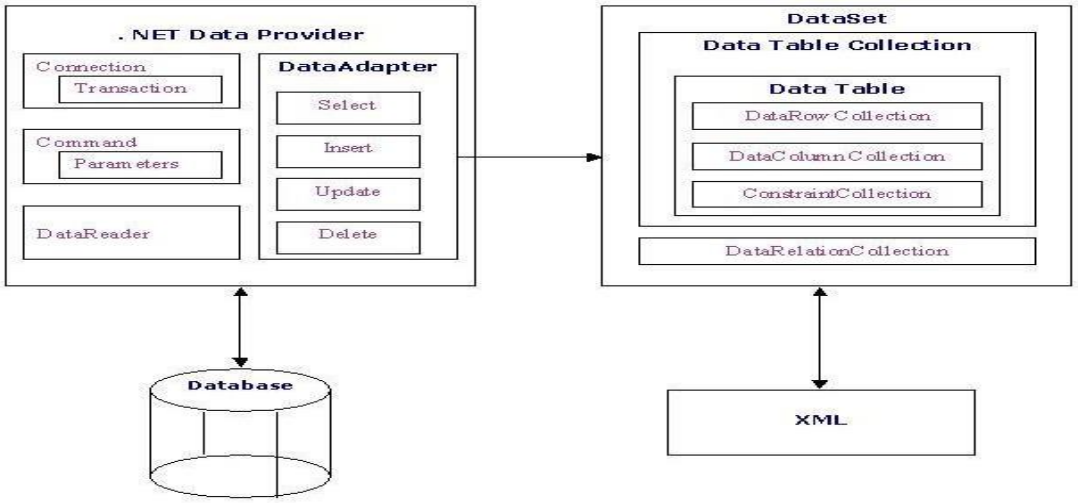
  The data access with ADO.NET consists of two parts:

  1. Data Provider
  2. DataSet

1. Data Provider

- The Data Provider is responsible for providing and maintaining the connection to the database.
- A DataProvider is a set of related components that work together to provide data in an efficient and performance driven manner.

- The .NET Framework currently comes with two DataProviders: the SQL Data



ADO .NET Data Architecture

Provider which is designed only to work with Microsoft's SQL Server 7.0 or later and the OleDb DataProvider which allows us to connect to other types of databases like Access and Oracle. Each DataProvider consists of the following component classes:

- The Connection object which provides a connection to the database
  The Command object which is used to execute a command
- The DataReader object which provides a forward-only, read only, connected recordset
- The DataAdapter object which populates a disconnected DataSet with data and performs update

Data access with ADO.NET can be summarized as follows:

- A connection object establishes the connection for the application with the database.
- The command object provides direct execution of the command to the database. If the command returns more than a single value, the command object returns a DataReader to provide the data.
- Alternatively, the DataAdapter can be used to fill the Datasetobject. The database can be updated using the command object or the DataAdapter.

Data Providers:

- It is responsible for providing and maintaining the connection to the database. We can use following data provider in Ado.Net
  - Oledb (for Access Database)
  - Sqlclient(for sqlserver Database)
  - Oracle (for oracle Database)
  - Odbc (for odbc Databse)

- **Ado.NET Objects**

ADO.NET consists of many objects that are used to work with data.

a) Connection Object
b) Command Object
c) DataAdapter Object
d) DataReader Object

a) Connection Object

- To establish connection with a database,you must have a connection object.
- The connection object helps to identify the datadase sever,the database name,user name,password,and other parameters that are required for connecting to the database.
- A connection object is used by command objects so that it will know on which database the command is executed.

Connection String – A string that specifies information about a data source and the means of connection to it is called Connection String.

```
Dim con As New SqlConnection

con.ConnectionString = "Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\jigisha\vb.netDemo_s
y6\sy6\sy6\Database1.mdf;Integrated Security=True;User
Instance=True"
```

Properties

Properties	Description
conectionString	It stores the connection string that is passed to the connection object at the time of creating its object.
Database	It stores the name of the database to which you need to connect.
State	It return the state of thae connectionEX.IsClose or IsOpen
Connection TimeOut	Gets the time to wait while trying to establish a connection before terminating the attempt and generating an error.

Methods

Methods	Description
Open	It opens the connection

Close	It closes the connection
BeginTransaction	It creates the Transaction Object.
ChangeDatabase	It creates and returns a SqlCommand object associated with the SqlConnection.
ChangeDatabase	It changes the current database for an open SqlConnection.

(b) Command Object

- It is used to retrieve a subset of data.Also invoking SQL statements insert,Update and Delete are directly require to set certain parameters on the command before executing the statement.common use of the command object
- Is to execute stored procedure and pass the appropriate parameters to the stored procedure.

Properties

Properties	Description
Connection	To set a connection object.
CommandText	It specifies the SQL string or stored procedure to be executed.
CommandType	It is used to determine how to interpret command text. Ex. CommandText is storedprocedure or Text or DirectTable.
CommandTimeOut	Gets the time to wait while trying to execute the command before terminating the attempt and generating an error.

Methods

Methods	Description
ExecuteNonQuery	It will execute the SQL statement and returns the number of rows affected by the query.
ExecuteScalar	It will execute the SQL statement which return the singleton value.
ExecuteReader	It will execute the SQL statement and returns the records in the form of DataReader.Ex.it is used to create the object of DataReader.
CreateParameter	It creates and returns a Sqlparameter object associated with the SqlCommand.
Cancel	It is used to cancel the command given for for execution.

ResetCommandTimeOut	It is used to reset Command time out property to its default value.

(C) DataReader Object

- A SqlDatReader is used to read data in the most efficient manner.You cannot use it for writing data.
- You can read forward-only and in sequential manner from SqldataReader.

Properties

Properties	Description
FieldCount	It stroes number of fields in a row.
HasRows	It specifies that the rows are selected or not for reading.
IsClosed	It specifies that DataReader is closed or not.
RecordsAffected	It returns -1 as DataReader is created on server.
Item	It gets the value of the specified column name.

Methods

Methods	Description
Read	It reads the Next Record of DataReader.
Close	It is used to Close the DataReader Connection with the database.
IsDBNull	It checks that the value of the column is Null or net.
GetSchemaTable	It returns the object of the DataTable for which the DataReader is created.
GetValues	It returns the array of the values for the row.
NextResult	It is used to nagate from one record set to another when
	more than one record sets are used in the command.

(e) DataAdapter

- It acts as a bridge between data source and in-memory data objects such as the Dataset.

Properties

Properties	Description
selectCommand	It is used to hold a Command that retrieves data from the data source.
UpdateCommand	It is used to hold a Command that updates data from the data source.
DeleteCommand	It is used to hold a Command that delete data from the data Source
InsertCommand	It is used to hold a Command that insert data from the data Source
Command and Type	It indicates CommandText property which contains SQL statement or stored procedure.If commandText property contains stroed procedure than user can set the value to CommandType.stored procedure.Default value is CommandType.Text for SQL statement.

Methods

Methods	Description
Fill	It is used to populate a dataset object with the data that the DataAdapter object retrieve from the data store using its SelectCommand.But before that we must initialize a Dataset object.
Update	It is used to update the database according to the changes that are made in the DataSet.

2. DataSet

- The dataset is a disconnected, in-memory representation of data. It can be considered as a local copy of the relevant portions of the database.

- The DataSet is continue in memory and the data in it can be manipulated and updated independent of the database.
- When the use of this DataSet is finished, changes can be made back to the central database for updating.
- The data in DataSet can be loaded from any valid data source like

Microsoft SQL server database, an Oracle database or from a Microsoft Access database.

2.4 Communications with Web Browser

ASP.NET is a server-side web development framework that allows web applications to interact with web browsers. The communication between the web browser (client) and ASP.NET web server (server) happens over the HTTP/HTTPS protocol.

This communication is request-response based:

- The browser sends a request to the server.
- The server processes the request and sends back a response.

What it Means:

ASP.NET talks to the browser using **Request** and **Response**.

- **Request** = Browser sends data to the server.
- **Response** = Server sends data back to the browser.

Key Objects Used in Communication

- ASP.NET provides built-in objects to handle communication:

Object	Description
Request	Used to receive data from the browser
Response	Used to send data to the browser
Session	Stores user-specific data across pages
Server	Provides utility methods
Application	Stores data shared among all users

Example 1:

Sending data from browser to server (Request)

HTML Form in ASP.NET WebForm:

```
<!-- Default.aspx -->
<form id="form1" runat="server">
    <asp:TextBox ID="txtName" runat="server" />
    <asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit_Click" />
    <asp:Label ID="lblResult" runat="server" />
</form>
```

Code-Behind (VB.NET)

```
' Default.aspx.vb

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub btnSubmit_Click(sender As Object, e As EventArgs)
        ' Request from browser
        Dim name As String = txtName.Text

        ' Response to browser
        lblResult.Text = "Hello, " & name
    End Sub
End Class
```

What Happens:

- User types their name and clicks the button.
- ASP.NET receives the name (**Request**).
- ASP.NET sends back "Hello, [name]" (**Response**).

2.5 Response Object in ASP.NET

What is the Response Object?

- The **Response object** is used to **send output from the server to the client's browser**.
- It belongs to the **System.Web** namespace.
- It is part of the **HttpResponse** class.

It is commonly used to:

- Write content to the browser
- Redirect to another page
- Set cookies or headers
- Set content type

Syntax:

Response.[Method or Property]

Common Uses of Response Object:

1. **Write() – Output Text to Browser**

- Used to display plain text, HTML, or dynamic values.

Example:

```
Response.Write("Welcome to ASP.NET!")
```

Output in browser:

Welcome to ASP.NET!

You can also write HTML:

Response.Write("<h2>Hello Students!</h2>")

2. **Redirect() – Redirect to Another Page**

- Moves the user to a different webpage.

Example:

```
Response.Redirect("homepage.aspx")
```

This will send the browser to homepage.aspx.

3. **ContentType – Set Type of Data**

- Tells browser what kind of data to expect (HTML, text, Excel, etc.)

Example:

```
Response.ContentType = "text/plain"

Response.Write("This is plain text content.")
```

Browser will treat it as simple text, not HTML.

4. **End() – Stop Further Processing**

- Stops the page execution immediately.

Example:

```
Response.Write("Only this will show.")
Response.End()
Response.Write("This will NOT show.")
```

5. **AddHeader() – Add Custom Headers**

- Used to add custom information to HTTP headers

Example:

```
Response.AddHeader("Refresh", "5;URL=homepage.aspx")
```

Automatically redirects to homepage.aspx after 5 seconds.

6. **Clear() – Clear Existing Output**

- Clears any content already written to the response buffer.

Example:

```
Response.Clear()
Response.Write("Only this will be sent.")
```

- **The ASP Response object is used to send output to the user from the server. Its collections, properties, and methods are described below:-**

Collections

Collection	Description
Cookies	Sets a cookie value. If the cookie does not exist, it will be created, and take the value that is specified

Properties

Property	Description
Buffer	Specifies whether to buffer the page output or not
CacheControl	Sets whether a proxy server can cache the output generated by ASP or not
Charset	Appends the name of a character-set to the content-type header in the Response object
ContentType	Sets the HTTP content type for the Response object
Expires	Sets how long (in minutes) a page will be cached on a browser before it expires
ExpiresAbsolute	Sets a date and time when a page cached on a browser will expire
IsClientConnected	Indicates if the client has disconnected from the server
Pics	Appends a value to the PICS label response header
Status	Specifies the value of the status line returned by the server

Methods

Method	Description
AddHeader	Adds a new HTTP header and a value to the HTTP response
AppendToLog	Adds a string to the end of the server log entry
BinaryWrite	Writes data directly to the output without any character conversion
Clear	Clears any buffered HTML output
End	Stops processing a script, and returns the current result
Flush	Sends buffered HTML output immediately
Redirect	Redirects the user to a different URL
Write	Writes a specified string to the output

2.7 Cookies

- Cookies is a small piece of information stored on the client machine. This file is located on client machines "C:\Document and Settings\Currently_Login user\Cookie" path.
- It is used to store user preference information like Username, Password, City, PhoneNo, etc, on client machines.
- We need to import a namespace called System.Web.HttpCookie before we use cookie.

Type of Cookies

- **Persist Cookie -** A cookie that doesn't have expired time is called a Persist Cookie
- **Non-Persist Cookie** - A cookie which has expired time is called a Non-Persist Cookie

➢ How to Create a Cookie?

- The "Response.Cookies" command is used to create cookies.

- **Note:** The Response.Cookies command must appear BEFORE the <html> tag.
- In the example below,

  we will create a cookie named "firstname" and assign the value "Alex" to it:-

```
<%
Response.Cookies("firstname")="Alex"
%>
```

- **It is also possible to assign properties to a cookie, like setting a date when the cookie should expire:-**

```
<%
Response.Cookies("firstname")="Alex"
Response.Cookies("firstname").Expires=#May 10,2012#
%>
```

➢ How to Retrieve a Cookie Value?

- The "Request.Cookies" command is used to retrieve a cookie value.
- **In the example below,**
  we retrieve the value of the cookie named "firstname" and display it on a page:

```
<%
fname=Request.Cookies("firstname")
response.write("Firstname=" & fname)
%>
```

Output: Firstname=Alex

A Cookie with Keys

- If a cookie contains a collection of multiple values, we say that the cookie has Keys.
- In the example below,

we will create a cookie collection named "user". The "user" cookie has Keys that contains information about a user:-

```
<%
Response.Cookies("user")("firstname")="John"
Response.Cookies("user")("lastname")="Smith"
Response.Cookies("user")("country")="Norway"
Response.Cookies("user")("age")="25"
%>
```

Read all Cookies

Look at the following code:

```
<%
Response.Cookies("firstname")="Alex"
Response.Cookies("user")("firstname")="John"
Response.Cookies("user")("lastname")="Smith"
Response.Cookies("user")("country")="Norway"
Response.Cookies("user")("age")="25"
%>
```

Assume that your server has sent all the cookies above to a user.

ASP.NET Cookie Practical

Step 1: Create a Web Form – CookieExample.aspx

Design (Drag & Drop in Visual Studio)

- Label → ID: lblMessage
- TextBox → ID: txtUsername
- Button → ID: btnSave → Text: **Save Cookie**
- Button → ID: btnShow → Text: **Show Cookie**
- Button → ID: btnDelete → Text: **Delete Cookie**

Step 2: Code Behind – CookieExample.aspx.vb

1. Save Cookie

```
Protected Sub btnSave_Click(sender As Object, e As EventArgs) Handles btnSave.Click

    Dim userCookie As New HttpCookie("username")

        userCookie.Value = txtUsername.Text

        userCookie.Expires = DateTime.Now.AddDays(3) ' Valid for 3 days

        Response.Cookies.Add(userCookie)

        lblMessage.Text = "Cookie Saved Successfully!"

End Sub
```

2. Show Cookie

```
Protected Sub btnShow_Click(sender As Object, e As EventArgs) Handles btnShow.Click

    If Request.Cookies("username") IsNot Nothing Then
```

```
    Dim userName As String = Request.Cookies("username").Value
    lblMessage.Text = "Welcome back, " & userName
  Else
    lblMessage.Text = "No cookie found."
  End If
  End Sub
```

3. Delete Cookie

```
 Protected Sub btnDelete_Click(sender As Object, e As EventArgs) Handles btnDelete.Click
  If Request.Cookies("username") IsNot Nothing Then
    Dim expiredCookie As New HttpCookie("username")
    expiredCookie.Expires = DateTime.Now.AddDays(-1)
    Response.Cookies.Add(expiredCookie)
    lblMessage.Text = "Cookie Deleted!"
  Else
    lblMessage.Text = "No cookie to delete."
  End If
End Sub
```

Cookie's common property

- **Domain =>** This is used to associate cookies to domain.
- **Secure** => We can enable secure cookie to set true(HTTPs).
- **Value** => We can manipulate individual cookie.
- **Values** => We can manipulate cookies with key/value pair.
- **Expires** => This is used to set expire date for the cookies.

Advantages of Cookie

- It has clear text so the user can read it.
- We can store user preference information on the client machine.

- It is an easy way to maintain.
- Fast accessing.

Disadvantages of Cookie

- If the user clears the cookie information, we can't get it back.
- No security.
- Each request will have cookie information with page.

2.7 Query String

What is a Query String?

- A Query String is a way to send data from one page to another through the URL.
- It appends values to the URL using ? and & symbols.
- The data sent via Query String is visible in the browser's address bar.
- It is mainly used for passing small, non-sensitive information like IDs, names, or page numbers.

Format of Query String:

PageName.aspx?key1=value1&key2=value2

Example URL:

Profile.aspx?name=Riya&age=20

Why Use Query String?

- To pass small data between pages (e.g., username, ID).
- It is simple and doesn't require server-side storage.

How to Use Query String in ASP.NET (VB.NET)

1. **Sending Query String (on Button Click)**

Let's say you have a button on Default.aspx:

```
Protected Sub btnSend_Click(sender As Object, e As EventArgs) Handles btnSend.Click

    Dim name As String = txtName.Text

    Dim age As String = txtAge.Text
```

```
    Response.Redirect("Profile.aspx?name=" & name & "&age=" & age)

End Sub
```

If the user enters:

- **Name: Riya**
- **Age: 20**

Then the URL becomes:

Profile.aspx?name= Riya&age=20

2 . Receiving Query String on Another Page

In Profile.aspx.vb (Page_Load):

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

    If Not IsPostBack Then

        Dim name As String = Request.QueryString("name")

        Dim age As String = Request.QueryString("age")

        lblResult.Text = "Welcome " & name & "! Your age is " & age

    End If

End Sub
```

Output:

When the page opens with URL:

Profile.aspx?name= Riya&age=20

You'll see:

Welcome Riya! Your age is 20

Advantages of Query String:

Feature	Description
Simple	Easy to implement
Visible	Data is visible in URL
No server memory	Doesn't store data on server

Limitations:

- Limited data (usually max 2048 characters)
- Not secure (data visible in URL)
- Can be modified by users

2.8 Session and State Management

- In ASP.NET, state management is the process of preserving user data (state) across multiple requests. Since HTTP is stateless, each request is treated as new.
- To maintain continuity (like logged-in users, shopping carts, etc.), ASP.NET provides state management techniques.

Types of State Management

1. Client-Side State Management

Data is stored on the client side (browser).

- ViewState
- Hidden Fields
- Cookies
- Query Strings

2. Server-Side State Management

Data is stored on the server side.

- Session State
- Application State
- Cache

Session State Management

- Session is used to store user-specific information on the server.
- Each user gets a unique Session ID.
- Data stored in a session is available across multiple pages for that user.

Example 1: Storing & Retrieving Session Values

```
' Example: ASP.NET (VB)

Partial Class _Default

    Inherits System.Web.UI.Page

    Protected Sub btnSave_Click(sender As Object, e As EventArgs) Handles btnSave.Click

        ' Store value in session

        Session("username") = txtUsername.Text

        lblMessage.Text = "Username stored in session!"

    End Sub

    Protected Sub btnGet_Click(sender As Object, e As EventArgs) Handles btnGet.Click

        ' Retrieve value from session

        If Session("username") IsNot Nothing Then

            lblMessage.Text = "Welcome, " & Session("username")

        Else

            lblMessage.Text = "Session expired or not set."

        End If

    End Sub

End Class
```

In this example:

- When user enters a name, it is stored in Session("username").
- On another page or postback, we can retrieve it.

Application State

- Shared across all users and sessions.
- Useful for storing global data.

Example 2: Application State

```
' Global.asax file

Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)

    Application("TotalVisitors") = 0

End Sub

Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)

    Application.Lock()

    Application("TotalVisitors") = CInt(Application("TotalVisitors")) + 1

    Application.UnLock()

End Sub
```

Here, Application("TotalVisitors") keeps track of how many users have visited the site.

ASP.NET Session Login with Database

Create table and insert records

```
CREATE TABLE Students ( StudentID INT PRIMARY KEY IDENTITY,

    Username NVARCHAR(50) NOT NULL,

    Password NVARCHAR(50) NOT NULL);
```

```
-- Insert sample data

INSERT INTO Students (Username, Password) VALUES ('naishal', '1234');

INSERT INTO Students (Username, Password) VALUES ('krina', 'abcd');
```

Web.config (Add Connection String)

```
<configuration>
  <connectionStrings>
    <add name="StudentDB"
        connectionString="Data Source=.\SQLEXPRESS;Initial Catalog=YourDatabaseName;Integrated Security=True"
        providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

Login.aspx (Design Page)

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Login.aspx.vb" Inherits="Login" %>

<!DOCTYPE html>

<html>

<head>

    <title>Student Login</title>

</head>

<body>

    <form id="form1" runat="server">
```

```
        <div>
            <h2>Student Login</h2>
            <asp:Label ID="lblUser" runat="server" Text="Username: "></asp:Label>
            <asp:TextBox ID="txtUsername" runat="server"></asp:TextBox>
            <br /><br />
            <asp:Label ID="lblPass" runat="server" Text="Password: "></asp:Label>
            <asp:TextBox ID="txtPassword" runat="server" TextMode="Password"></asp:TextBox>
            <br /><br />
            <asp:Button ID="btnLogin" runat="server" Text="Login" />
            <br /><br />
            <asp:Label ID="lblMessage" runat="server" ForeColor="Red"></asp:Label>
        </div>
    </form>
</body>
</html>
```

Login.aspx.vb (Code-Behind with Database Validation)

```
Imports System.Data.SqlClient

Partial Class Login
    Inherits System.Web.UI.Page
    Protected Sub btnLogin_Click(sender As Object, e As EventArgs) Handles btnLogin.Click
Dim con As New
SqlConnection(System.Configuration.ConfigurationManager.ConnectionStrings("StudentDB").ConnectionStri
ng)
```

```
        Dim cmd As New SqlCommand("SELECT * FROM Students WHERE Username=@uname AND
Password=@pwd", con)

        cmd.Parameters.AddWithValue("@uname", txtUsername.Text)

        cmd.Parameters.AddWithValue("@pwd", txtPassword.Text)

        Dim da As New SqlDataAdapter(cmd)

        Dim dt As New DataTable()

        da.Fill(dt)

        If dt.Rows.Count > 0 Then

            ' Login success

            Session("username") = dt.Rows(0)("Username").ToString()

            Response.Redirect("Welcome.aspx")

        Else

            '  Login failed

            lblMessage.Text = "Invalid Username or Password!"

        End If

    End Sub

End Class
```

Welcome.aspx (Design Page)

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Welcome.aspx.vb" Inherits="Welcome"
%>

<!DOCTYPE html>

<html>
```

```
<head>
    <title>Welcome Student</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h2>Welcome Page</h2>
            <asp:Label ID="lblWelcome" runat="server"></asp:Label>
            <br /><br />
            <asp:Button ID="btnLogout" runat="server" Text="Logout" />
        </div>
    </form>
</body>
</html>
```

Welcome.aspx.vb (Session Check & Logout)

```
Partial Class Welcome
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(sender As Object, e As EventArgs) Handles Me.Load
        If Session("username") Is Nothing Then
            Response.Redirect("Login.aspx")
```

```
        Else
            lblWelcome.Text = "Welcome, " & Session("username") & "!"
        End If
    End Sub
Protected Sub btnLogout_Click(sender As Object, e As EventArgs) Handles btnLogout.Click
        Session.Abandon()
        Response.Redirect("Login.aspx")
    End Sub
End Class
```

How this works

- Student enters login details.
- Code checks Students table in SQL Server.
- If record exists → store username in Session and redirect to Welcome.aspx.
- If not → show error message.
- On Welcome.aspx, if session exists → show welcome message. Otherwise redirect to login.
- Logout clears session.