

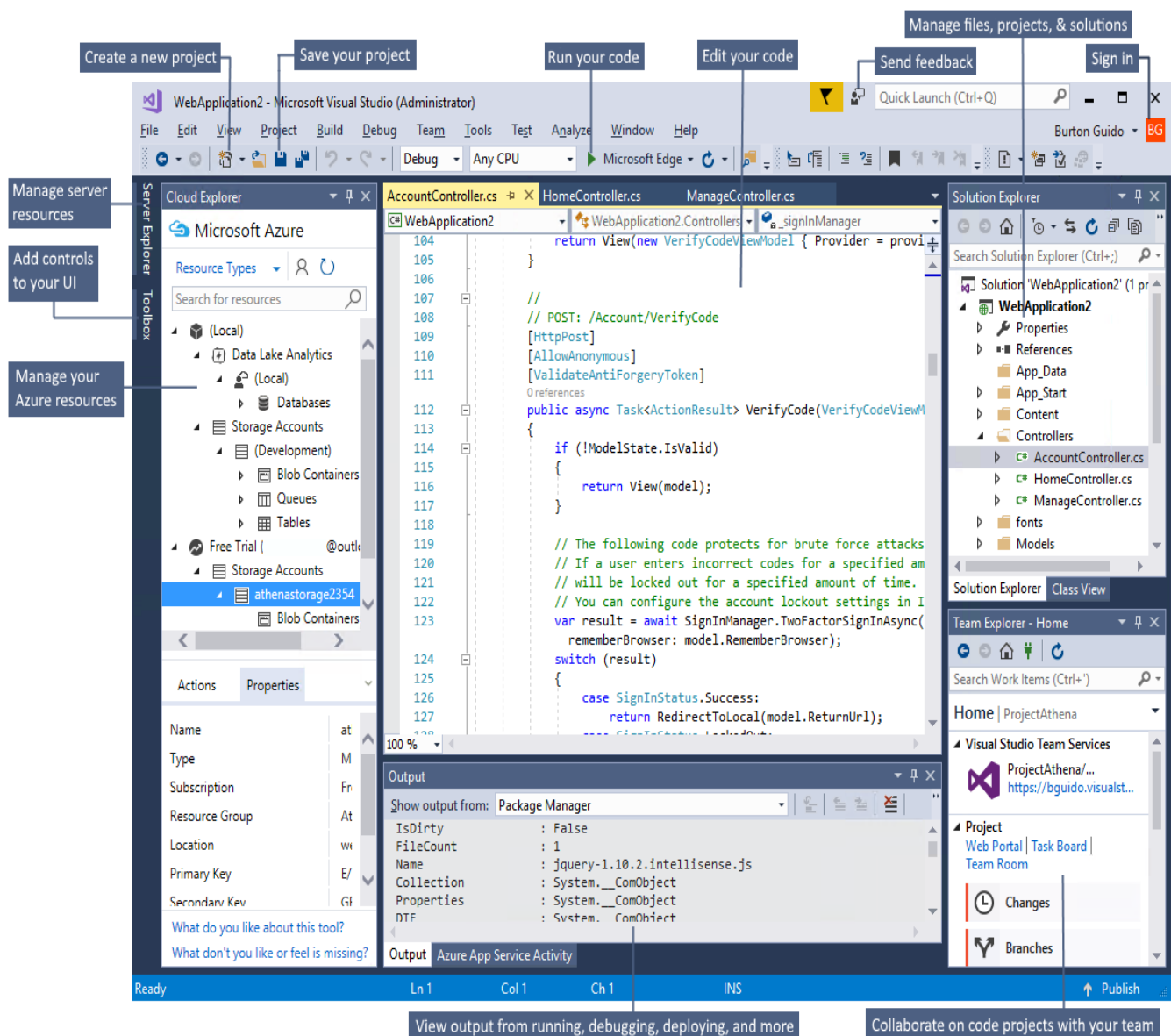
1. Introduction to Visual Studio IDE.

➤ Visual Studio IDE:

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

➤ Features:

- ❖ Mainly, it is a source code editor.
- ❖ It supports many languages.
- ❖ It has facility of debugger.
- ❖ Use version control, be agile, collaborate efficiently.



Important Features :

➤ One IDE for all .NET Projects

Visual Studio .NET IDE provides a single environment for developing all types of .NET applications.

Application's range from single windows applications to complex n-tier applications, rich web applications, Web Services, Windows Services, Mobile applications, Console Applications etc.,

➤ Option to choose from Multiple Programming Languages

You can choose the programming language of your choice to develop applications based on your expertise in that language.

The major programming languages supported by the Visual Studio are C#, VB.Net, F#, J#, C++ etc., You can also incorporate multiple programming languages in one .NET solution and edit that with the IDE.

➤ Option to choose from Multiple Frameworks

You can choose a specific framework to develop the project for your specific target platform. Also you can have multiple projects with multiple frameworks under single solution.

➤ IDE is Customizable

You can customize the IDE based on your preferences. The **Options** under **Tools** menu settings allow you to do this. With these settings you can set the IDE screen the way you want, the way the keyboard behaves and you can also filter the help files based on the language of your choice. You can select fonts, colors of your coding window.

You can also change the Text Editor properties for various languages. If you want to store your settings you can Import and Export your settings.

➤ Built-in Browser

The IDE comes with a built-in browser that helps you browse the Internet without launching another application. You can look for additional resources, online help files, source codes and much more with this built-in browser feature.

When we open VS .NET from Start->Programs->Microsoft Visual Studio .NET->Microsoft Visual Studio .NET the window that is displayed first is the Start Page which is shown below. The start Page allows us to select from the most recent projects (last four projects) with which we worked or it can be customized based on your preferences.

7. Create an C#.Net application which demonstrates the usage of properties and indexer.

Properties:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace pra7_2
{
    class propertydemo
    {
        int x;

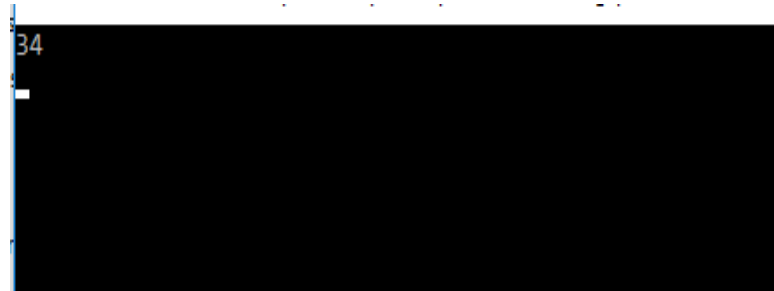
        public propertydemo(int y)
        {
            x = y;
        }

        public int Num
        {
            get { return x; }
            set { x = value; }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            propertydemo pd = new propertydemo(34);
```

```
        Console.WriteLine(pd.Num);  
        Console.ReadLine();  
    }  
}  
}
```

Output



Indexer:

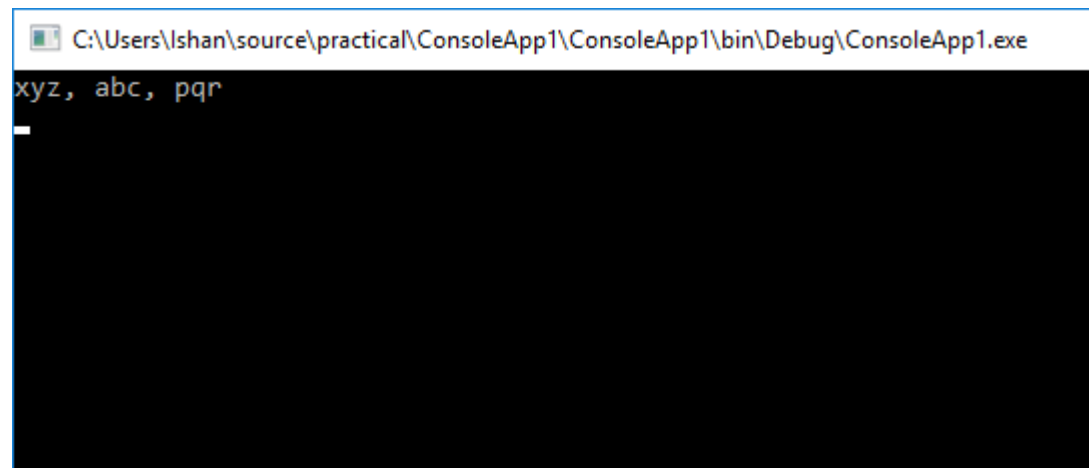
```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace ConsoleApp1
```

```
{  
    class Index  
    {  
        string[] str = new string[3];  
        public string this[int index]  
        {  
            get { return str[index]; }  
            set { str[index] = value; }  
        }  
    }  
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Index ob = new Index();
        ob[0] = "xyz";
        ob[1] = "abc";
        ob[2] = "pqr";
        Console.WriteLine("{0}, {1}, {2}", ob[0], ob[1], ob[2]);
        Console.ReadLine();
    }
}
```

Output

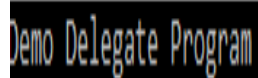
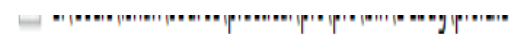


The screenshot shows a Windows command prompt window with the title bar text "C:\Users\Ishan\source\practical\ConsoleApp1\ConsoleApp1\bin\Debug\ConsoleApp1.exe". The command prompt displays the output of the program: "xyz, abc, pqr". A white cursor is visible on the line below the output.

8. Create a C#.Net application which demonstrates the usage of Delegates.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace pr8
{
    public delegate int Mydel(int x, int y);
    class sample
    {
        public static int del()
        {
            Console.WriteLine("Demo Delegate Program");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Mydel md = new Mydel(del);
            Console.ReadLine();
        }
    }
}
```

Output:



9. To study about different windows control available in .net framework 3.5.

Function	Control	Description
Data display	DataGridView control	The DataGridView control provides a customizable table for displaying data. The DataGridView class enables customization of cells, rows, columns, and borders.
Data binding and navigation	BindingSource component	Simplifies binding controls on a form to data by providing currency management, change notification, and other services.
Text editing	TextBox control	Displays text entered at design time that can be edited by users at run time, or changed programmatically.
	RichTextBox control	Enables text to be displayed with formatting in plain text or rich-text format (RTF).
	MaskedTextBox control	Constrains the format of user input
Information display (read-only)	Label control	Displays text that users cannot directly edit.
	LinkLabel control	Displays text as a Web-style link and triggers an event when the user clicks the special text. Usually the text is a link to another window or a Web site.
	StatusStrip control	Displays information about the application's current state using a framed area, usually at the bottom of a parent form.
	ProgressBar control	Displays the current progress of an operation to the user.
Web page display	WebBrowser control	Enables the user to navigate Web pages inside your form.
Selection from a list	CheckedListBox control	Displays a scrollable list of items, each accompanied by a check box.
	ComboBox control	Displays a drop-down list of items.
	DomainUpDown control	Displays a list of text items that users can scroll through with up and down buttons.
	ListBox control	Displays a list of text and graphical items (icons).
	ListView control	Displays items in one of four different views. Views include text only, text with small icons, text with large icons, and a details view.
	NumericUpDown control	Displays a list of numerals that users can scroll through with up and down buttons.
	TreeView control	Displays a hierarchical collection of node objects that can consist of text with optional

		check boxes or icons.
Graphics display	PictureBox control	Displays graphical files, such as bitmaps and icons, in a frame.
Graphics storage	ImageList control	Serves as a repository for images. ImageList controls and the images they contain can be reused from one application to the next.
Value setting	CheckBox control	Displays a check box and a label for text. Generally used to set options.
	CheckedListBox control	Displays a scrollable list of items, each accompanied by a check box.
	RadioButton control	Displays a button that can be turned on or off.
Date setting	DateTimePicker control	Displays a graphical calendar to allow users to select a date or a time.
	MonthCalendar control	Displays a graphical calendar to allow users to select a range of dates.
Dialog boxes	ColorDialog control	Displays the color picker dialog box that allows users to set the color of an interface element.
	FontDialog control	Displays a dialog box that allows users to set a font and its attributes.
	OpenFileDialog control	Displays a dialog box that allows users to navigate to and select a file.
	PrintDialog control	Displays a dialog box that allows users to select a printer and set its attributes.
	PrintPreviewDialog control	Displays a dialog box that displays how a control PrintDocument component will appear when printed.
	FolderBrowserDialog control	Displays a dialog that allows users to browse, create, and eventually select a folder
	SaveFileDialog control	Displays a dialog box that allows users to save a file.
Menu controls	MenuStrip control	Creates custom menus.
	ContextMenuStrip control	Creates custom context menus.
Commands	Button control	Starts, stops, or interrupts a process.
	LinkLabel control	Displays text as a Web-style link and triggers an event when the user clicks the special text. Usually the text is a link to another window or a Web site.
	NotifyIcon control	Displays an icon in the status notification area of the taskbar that represents an application running in the background.
	ToolStrip control	Creates toolbars that can have a Microsoft Windows XP, Microsoft Office, Microsoft

		Internet Explorer, or custom look and feel, with or without themes, and with support for overflow and run-time item reordering.
User Help	HelpProvider component	Provides pop-up or online Help for controls.
	ToolTip component	Provides a pop-up window that displays a brief description of a control's purpose when the user rests the pointer on the control.
Grouping other controls	Panel control	Groups a set of controls on an unlabeled, scrollable frame.
	GroupBox control	Groups a set of controls (such as radio buttons) on a labeled, nonscrollable frame.
	TabControl control	Provides a tabbed page for organizing and accessing grouped objects efficiently.
	SplitContainer control	Provides two panels separated by a movable bar.
	TableLayoutPanel control	Represents a panel that dynamically lays out its contents in a grid composed of rows and columns.
	FlowLayoutPanel control	Represents a panel that dynamically lays out its contents horizontally or vertically.
Audio	SoundPlayer control	Plays sound files in the .wav format. Sounds can be loaded or played asynchronously.

10. Create a C#.Net window application which demonstrates the usage of MDI form.

ParentFile:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace mdiform
{
    public partial class Parent : Form
    {
        public Parent()
        { InitializeComponent(); }
        private void newToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            mdichild1 mdi = new mdichild1();
            mdi.Show();    } }}

```

Mdichild1:

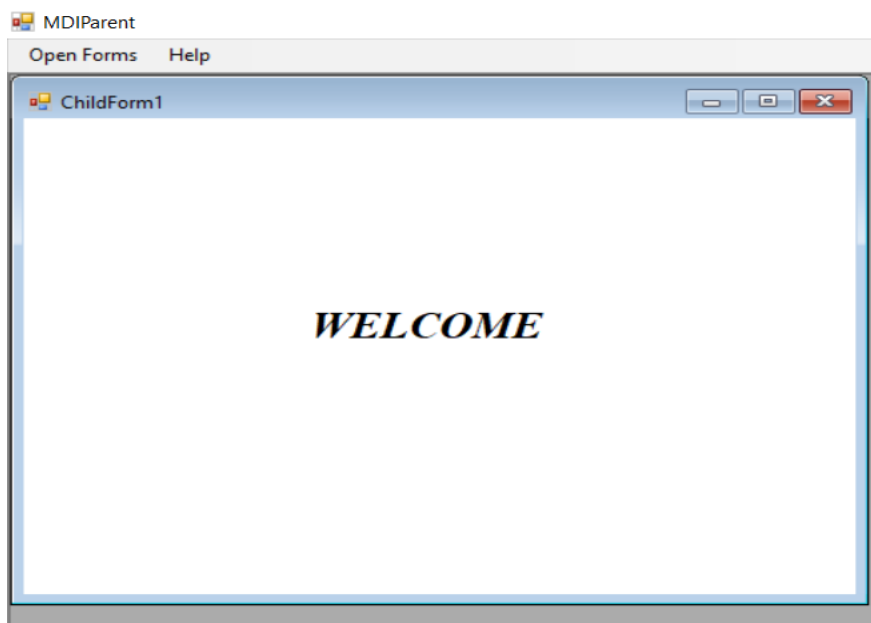
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace mdiform
{
    public partial class mdichild1 : Form
    {
        public mdichild1()
        {
            InitializeComponent();
        }
        private void newToolStripMenuItem_Click(object sender, EventArgs e)
        {
            mdichild2 mdi = new mdichild2();
            mdi.Show();
        }
    }
}

```



11. Create a window application for connection with SQL-Express.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace p11 {
    public partial class Form1 : Form {

        SqlConnection con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\jdkam\Documents\testing.m
df;Integrated Security=True;Connect Timeout=30");

        public Form1() {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e) {
            disp_data();
        }

        private void button1_Click(object sender, EventArgs e) {
            con.Open();

            SqlCommand cmd = con.CreateCommand();

            cmd.CommandType = CommandType.Text;

            cmd.CommandText = "insert into Table1
values('"+textBox1.Text+"','"+textBox2.Text+"','"+textBox3.Text+"')";

            cmd.ExecuteNonQuery();

            con.Close();

            disp_data();

            MessageBox.Show("Data inserted successfully");
        }

        public void disp_data() {
```

```

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "select * from Table1";

cmd.ExecuteNonQuery();

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

dataGridView1.DataSource = dt;

con.Close();
}

private void button3_Click(object sender, EventArgs e) {

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "delete from Table1 where name='" + textBox1.Text + "'";

cmd.ExecuteNonQuery();

con.Close();

disp_data();

MessageBox.Show("Data deleted successfully");

}

private void button2_Click(object sender, EventArgs e) {

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "update Table1 set name='"+textBox2.Text+"'where
name='"+textBox1.Text+"'";

cmd.ExecuteNonQuery();

con.Close();

disp_data();

```

```
MessageBox.Show("Data updated successfully");
```

```
}} }
```

Output:

Form1

Name: abc

City: Mumbai

Country: India

Insert Update Delete

	name	city	country
▶	abc	Mumbai	India
*			

Data inserted successfully

OK

Form1

Name: def

City: Delhi

Country: India

Insert Update Delete

	name	city	country
▶	abc	Mumbai	India
*			

Data deleted successfully

OK

Form1

Name: abc

City: pqr

Country: India

Insert Update Delete

	name	city	country
▶	pqr	Mumbai	India
*			

Data updated successfully

OK

12. To study about ASP.NET life cycle.

The ASP.NET life cycle can broadly be divided into two groups.

1. Application Life Cycle
2. Page Life Cycle

1) Application Life Cycle

- STAGE-1: - Application Start:
 - User makes a request for accessing application resource, a page. Browser sends this request to the web server.
 - During this time, there is a method called `Application_start` which is executed by the web server. Usually in this method all the global variables are set to their default values.
 - A unified pipeline receives the first request and the following events take place.
 - An object of the class `ApplicationManager` is created for this task.
 - An object of the class `HostingEnvironment` is created that provides information regarding the resources.
 - Now, top level items in the application are compiled.
- STAGE-2: - Object Creation:
 - The next stage is the creation of the `HttpContext`, `HttpRequest` & `HttpResponse` by the web server.
 - The `HttpRequest` is just the container for the `HttpRequest` and `HttpResponse` objects.
 - The `HttpRequest` object contains information about the current request, including cookies and browser information.
 - The `HttpResponse` object contains the response that is sent to the client.
- STAGE-3: - `HttpApplication` Creation:
 - An instance of the `HttpApplication` object is created and assigned to the request.
 - This object is created by the web server. It is this object that is used to process each subsequent request to the application.
- STAGE-4: - Dispose:
 - This event is called before the application instance is destroyed. During this time, one can use this method manually release any unmanaged resources.
- STAGE-5: - Application End:
 - This is the final part of the application. In this part, the application is finally unloaded from the memory.

2) ASP.NET Page Life Cycle

The page class creates a hierarchical tree of all the controls on the page.

The page life cycle phases are:

- Initialization
- Instantiation of the controls on the page
- Restoration and maintenance of the state
- Execution of the event handler codes
- Page rendering

Following are the different stages of an ASP.Net page:

- **Page request:-**
 - when ASP.Net gets a page request, it decides whether to parse and compile the page or there would be a cached version of the page; accordingly the response is sent
- **Starting of page life cycle:-**
 - at this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.
- **Page initialization:-**
 - at this stage, the controls on the page are assigned unique ID by setting the UniqueID property and themes are applied. For a new request postback data is loaded and the control properties are restored to the view-state values.
- **Page load:-**
 - at this stage, control properties are set using the view state and control state values.
- **Validation:-**
 - Validate method of the validation control is called and if it runs successfully, the IsValid property of the page is set to true.
- **PostBack event handling:-**
 - if the request is a postback (old request), the related event handler is called.
- **Page rendering:-**
 - at this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Page's Response property.
- **Unload:-**
 - the rendered page is sent to the client and page properties, such as Response and Request are unloaded and all cleanup done.

ASP.Net Page Life Cycle Events:

At each stage of the page life cycle, the page raises some events, which could be coded. An event handler is basically a function or subroutine, bound to the event, using declarative attributes like Onclick or handle.

- **PreInit:-**
 - PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback.
 - It sets the themes and master pages, creates dynamic controls and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page_PreInit handler.
- **Init:-**
 - Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page_Init handler.
- **InitComplete:-**
 - InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
- **LoadViewState:-**
 - LoadViewState event allows loading view state information into the controls.
- **LoadPostData:-**
 - during this phase, the contents of all the input fields defined with the <form> tag are processed.
- **PreLoad:-**
 - PreLoad occurs before the post back data is loaded in the controls.
 - This event can be handled by overloading the OnPreLoad method or creating a Page_PreLoad handler.
- **Load:-**
 - the Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page_Load handler.
- **LoadComplete:-**
 - the loading process is completed, control event handlers are run and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page_LoadComplete handler.
- **PreRender:-**
 - the PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.
- **PreRenderComplete:-**
 - as the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.
- **SaveStateComplete:-**
 - state of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page_Render handler.
- **UnLoad:-**
 - the UnLoad phase is the last phase of the page life cycle.
 - It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed.
 - This event can be handled by modifying the OnUnLoad method or creating a Page_UnLoad handler.

13. To study about ADO.NET architecture.

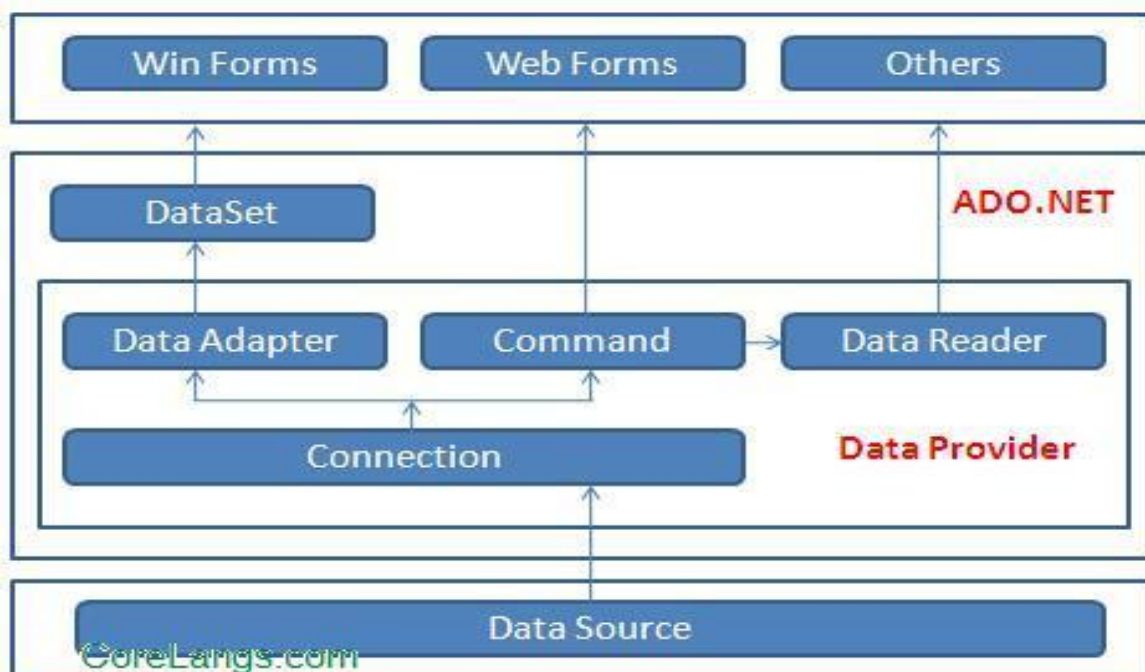
Concept:

ADO.NET is an object-oriented set of libraries that allows developer to interact with data sources. Commonly, the data source is a database, but it could also be a text file, an excel spreadsheet, or an XML file.

Explanation:

There are many different types of databases available through which data can be accessed. For Example, there is Microsoft SQL, Microsoft Access, Oracle, Borland Interface and IBM DB2, just to name a few.

ADO.NET Architecture:



➤ The ADO.NET Data Architecture

- Data Access in ADO.NET relies on two components: DataSet and Data Provider.

➤ 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. 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.

➤ Data Provider

- 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 Provider which is designed only to work the OleDb DataProvider which allows us to connect to other types of databases like Access and Oracle.
- 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

Component classes that make up the Data Providers

➤ The Connection Object

- The Connection object creates the connection to the database. Microsoft Visual Studio .NET provides two types of Connection classes: the SqlConnection object, The Connection object contains all of the information required to open a connection to the database.

➤ The Command Object

- The Command object is represented by two corresponding classes: SqlCommand and OleDbCommand. The Command objects can be used to execute stored procedures on the database, SQL commands, or return complete tables directly. Command objects provide three methods that are used to execute commands on the database:

➤ The DataReader Object

- The DataReader object provides a forward-only, read-only, connected stream recordset from a database. Unlike other components of the Data Provider, DataReader objects cannot be directly instantiated. Because only one row is in memory at a time, the DataReader provides the lowest overhead in terms of system performance but requires the exclusive use of an open Connection object for the lifetime of the DataReader.

➤ The DataAdapter Object

- The DataAdapter is the class at the core of ADO .NET's disconnected data access. It is essentially the middleman facilitating all communication between the database and a DataSet. The DataAdapter is used either to fill a DataTable or DataSet with data from the database with its Fill method. The DataAdapter provides four properties that represent database commands.

14. Create web application in ASP.NET to provide input validations using Input Valuator.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="validation.aspx.cs"
Inherits="p14.validation" %>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
```

```
</head>
```

```
<body>
```

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

```
<div>
```

```
<asp:Label ID="Label1" runat="server" Text="Name:"></asp:Label>
```

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

```
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox1"
ErrorMessage="Please enter name"></asp:RequiredFieldValidator>
```

```
<div> <asp:Label ID="Label2" runat="server" Text="Phone No."></asp:Label>
```

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
ControlToValidate="TextBox2" ErrorMessage="Please enter phone
no"></asp:RequiredFieldValidator>
```

```
<p> <asp:TextBox ID="TextBox3" runat="server" MaxLength="500"
TextMode="MultiLine"></asp:TextBox>
```

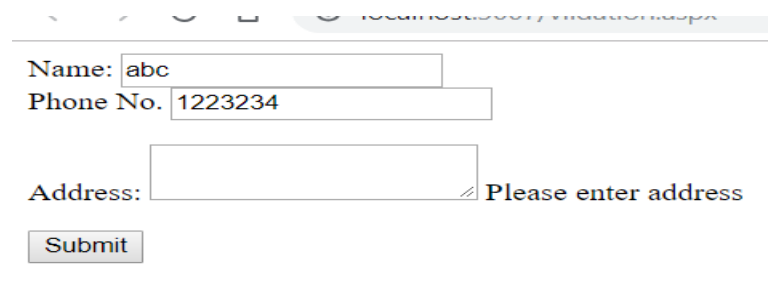
```
<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
ControlToValidate="TextBox3" ErrorMessage="Please enter
address"></asp:RequiredFieldValidator>
```

```
<asp:Button ID="Button1" runat="server" Text="Submit" />
```

```
</form>
```

```
</body>
```

```
</html>
```



Name:

Phone No.

Address:

Please enter address

15. Create a web application that will make connection with SQL Server express and perform operations of addition, updating and deletion of data on Login Form.

Login.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
namespace session_management {
    public partial class login : System.Web.UI.Page {
        SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString
"].ConnectionString);
        SqlCommand cmd;
        SqlDataAdapter adp;
        protected void Button1_Click(object sender, EventArgs e) {
            DataTable dt = new DataTable();
            cmd = new SqlCommand("select * from emp_login where Id=@ID    and
password=@password", con);
            cmd.Parameters.AddWithValue("Id", TextBox1.Text);
            cmd.Parameters.AddWithValue("password", TextBox2.Text);
            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
            adp = new SqlDataAdapter(cmd);
            adp.Fill(dt);
            if (dt.Rows.Count > 0) {
                Session["userid"] = TextBox1.Text;
                Response.Redirect("home.aspx");
            }
            else {
                Label3.Text = "invalid name or password";
            }
        }
        protected void LinkButton1_Click(object sender, EventArgs e) {
            Response.Redirect("register.aspx");
        }
    }
}
```

Register.aspx.cs:

```
using System;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Configuration; using
System.Data.SqlClient; using
System.Data;
namespace session_management
{
    public partial class register : System.Web.UI.Page
    {
        SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString
"].ConnectionString);
        SqlDataAdapter adp;
        protected void Button1_Click(object sender, EventArgs e)
        {
            if (TextBox1.Text != "" && TextBox2.Text != "" &&
TextBox3.Text != "" && TextBox4.Text != "")
            {
                SqlCommand cmd = new SqlCommand("insert into emp_login(password)
values(@pwd)", con); SqlCommand cmd1 = new SqlCommand("insert into
emp_detail(name,city,email_id) values(@name,@city,@email_id)", con);
                cmd.Parameters.AddWithValue("pwd", TextBox2.Text);
                cmd1.Parameters.AddWithValue("name", TextBox1.Text);
                cmd1.Parameters.AddWithValue("city", TextBox3.Text);
                cmd1.Parameters.AddWithValue("email_id", TextBox4.Text) ;
                cmd.ExecuteNonQuery();
                cmd1.ExecuteNonQuery();
                Response.Redirect("login.aspx");
            }
        }
    }
}
```

home.aspx.cs:

```
using System;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
```

```

namespace session_management
{
    public partial class home : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)    {
            SqlConnection con = new SqlConnection
            (ConfigurationManager.ConnectionStrings["ConnectionString"].
            ConnectionString);
            con.Open();
            DataTable dt = new DataTable();
            string Id = Session["userid"].ToString();
            SqlDataAdapter adp = new SqlDataAdapter("select * from
emp_detail where Id=@Id", con);
            adp.SelectCommand.Parameters.AddWithValue("Id",Id);
            adp.Fill(dt);
            GridView1.DataSource = dt;
            GridView1.DataBind();
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            Session.RemoveAll();
            Response.Redirect("login.aspx");
        }
    }
}

```

← → ↻ localhost:4922/register.aspx

name	<input type="text" value="sagar"/>
password	<input type="text" value="123456789"/>
city	<input type="text" value="bhavnagar"/>
email id	<input type="text" value="sagar@mail.com"/>
<input type="button" value="register"/>	

← → ↻ localhost:4922/login.aspx

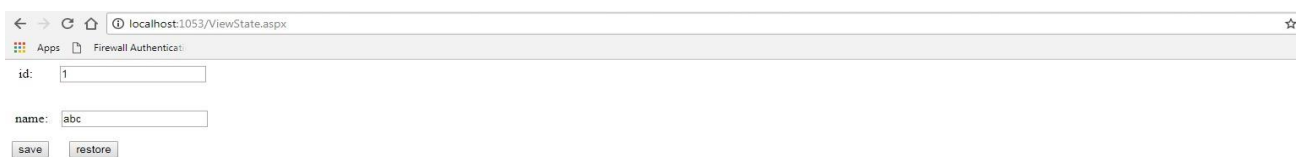
name	<input type="text" value="sagar"/>
password	<input type="text" value="123456789"/>
<input type="button" value="login"/>	
Create new account	

16. Create a web application that will use the concept of State Management

(I) View State (II) Query String (III) Cookies (IV) Session

(I) View State:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace state_maintain
{
    public partial class ViewState : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e) { }
        protected void Button1_Click(object sender, EventArgs e)
        {
            ViewState["value1"] = TextBox1.Text.ToString();
            ViewState["value2"] = TextBox2.Text.ToString();
            TextBox1.Text = TextBox2.Text = "";
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
            TextBox1.Text = ViewState["value1"].ToString();
            TextBox2.Text = ViewState["value2"].ToString();
        }
    }
}
```



(II) Query String

```
using System;
using System.Collections.Generic; using
System.Linq; using System.Web; using
System.Web.UI;
using System.Web.UI.WebControls;

namespace state_maintain {
    public partial class QueryString2 : System.Web.UI.Page {
```

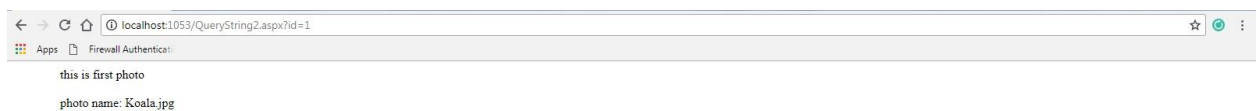


```

protected void Page_Load(object sender, EventArgs e)
{
    string id=Request.QueryString["id"];
    if (id == "1")
    {
        Label1.Text = "this is first photo";
        Label2.Text="photo name: Koala.jpg";

    }
    if (id == "2")
    {
        Label1.Text = "this is second photo";
        Label2.Text = "photo name: Jellyfish.jpg";
    }
}
}

```



(III)Cookies

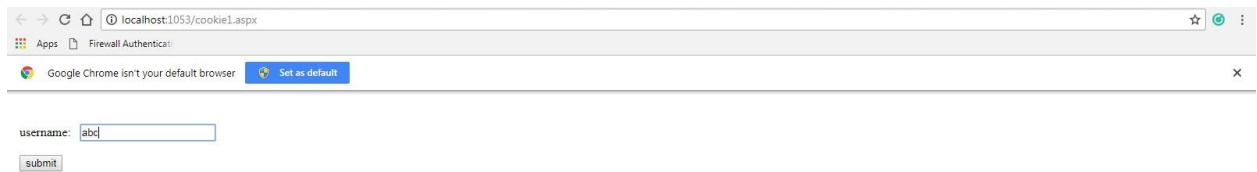
cookie1.aspx:

```

using System;
using System.Collections.Generic; using
System.Linq; using System.Web; using
System.Web.UI;
using System.Web.UI.WebControls;

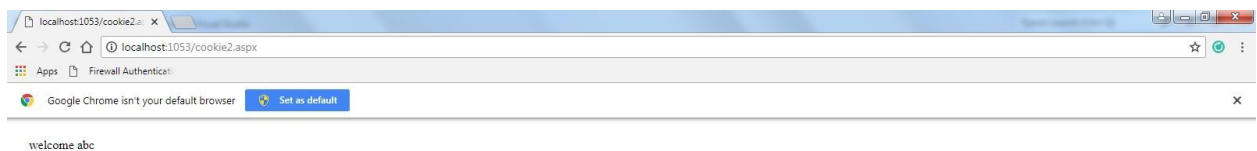
namespace state_maintain
{
    public partial class cookie1 : System.Web.UI.Page
    {
        protected void Button1_Click(object sender, EventArgs e)
        {
            HttpCookie userinfo = new HttpCookie("userinfo");
            userinfo["username"] = TextBox1.Text;          userinfo.Expires =
            DateTime.Now.AddMinutes(1);
            Response.Cookies.Add(userinfo);
            Response.Redirect("cookie2.aspx");
        }
    }
}

```



cookie2.aspx:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace state_maintain {
    public partial class cookie2 : System.Web.UI.Page {
        protected void Page_Load(object sender, EventArgs e) {
            string user_name = string.Empty;
            HttpCookie reqcookies=Request.Cookies["userinfo"];           if(reqcookies!=null)
            {
                user_name=reqcookies["username"].ToString();
            }
            Label1.Text = "welcome" + " " +user_name;
        }
    }
}
```



(IV)Session

Session1.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace session_management
{
    public partial class session1 : System.Web.UI.Page {
        protected void Page_Load(object sender, EventArgs e) { }
```

```

protected void Button1_Click(object sender, EventArgs e)
{
    var userid = TextBox1.Text;
    var pwd = TextBox2.Text;
    if (userid == "a@gmail.com" && pwd == "12345678") {
        Session["userid"] = userid;
        Response.Redirect("home1.aspx");
    }
}
}
}

```

localhost:4922/session1.aspx

user id:

password:

home.aspx.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace session_management {
    public partial class home1 : System.Web.UI.Page {
        protected void Page_Load(object sender, EventArgs e) {
            Label1.Text="welcome user:" + " " + Session["userid"];
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            Session.RemoveAll();
            Response.Redirect("session1.aspx");
        }
    }
}

```

localhost:4922/home1.aspx

welcome user: a@gmail.com