.Net History
Microsoft started development of the .NET Framework in the late 1990s, originally under the name of Next Generation Windows S
The C# programming language was developed with the .NET Framework by Anders Hejlsberg, Scott Wiltamuth, and Peter Golde
These are the versions of C# and .Net framework:
C# 1.0; released with .NET 1.0 and VS2002 (January 2002).Contained the first version CLR and base class libraries
C# 1.2 ; released with .NET 1.1 and VS2003 (April 2003). First version to call Dispose on IEnumerators which implemented IDispo
C# 2.0; released with .NET 2.0 and VS2005 (November 2005). Major new features: generics, nullable types
C# 3.0; released with .NET 3.5 and VS2008 (November 2007). Major new features: lambda expressions(=>),LINQ , implicit typing
C# 4.0; released with .NET 4 and VS2010 (April 2010). Major new features: Dynamic language run-time, parallel computing, nam
C# 5.0; released with .NET 4.5 and VS2012(August 2012).Major new features: asynchronous file operations and improved on pa
4.5.1 Preview with Visual Studio 2013 Preview. Includes performance and debugging improvements, support for automatic bindin
Overview of the .NET Framework
The .NET Framework is a technology that supports building and running the next generation of applications and XML Web service
Provide a runtime environment that simplifies software deployment and reduces the chances of version conflicts.
Enable the safe execution of code.
Use industry standards for all communication to enable integration with non-.NET code.
Provide a consistent developer experience across all types of applications in a way that is language- and platform-independent.
Provide a runtime environment that minimizes or eliminates the performance problems of scripted or interpreted languages.
Terminoly:.NET enabled Microsoft's marketing people to emphasise the "Network"-ing aspect of its technologies, and was also a r
.Net architecture

Common Language Runtime(CLR)
It is the heart of the .NET framework. it is the responsibility of the runtime to take care of the code execution of the program. Progr
Following are the responsibilities of CLR:
• Garbage Collection: - CLR automatically manages memory thus eliminating memory leaks. When objects are not referred, GC a
• Code Access Security: - CAS grants rights to program depending on the security configuration of the machine. Example the prog
• Code Verification: - This ensures proper code execution and type safety while the code runs. It ■prevents the source code to pe
Managed Code
Managed code runs inside the environment of CLR i.e. .NET runtime. In short, all IL are managed code. However, if you are using
MSIL(Microsoft Intermediate Language) or CIL(Common Intermediate Language) or IL(Intermediate Language)
All .NET source code is converted to an intermediate code known as MSIL which is interpreted by the CLR. MSIL is OS and hardw
Just-In-Time(JIT) compiler
It compiles the IL code to native executable code(.exe or .dll) that is designed for specific machine and OS.

The Framework Class Library(FCL)
The .Net Framework class library (FCL) provides the core functionality of .Net Framework architecture. The .Net Framework Class
This library is categorized into different modules and can access to Windows application, Web development, Network programmin
Common Type System(CTS)
CTS define how types are declared, used and managed in the CLR, and is also an important part of the runtime's support for cros
Establishes a framework that helps enable cross-language integration, type safety, and high-performance code execution.
Provides an object-oriented model that supports the complete implementation of many programming languages.
Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each oth
Provides a library that contains the primitive data types (such as Boolean, Byte, Char, Int32, and UInt64) used in application devel
This makes it possible for the 2 languages to communicate with each other by passing/receiving parameters to and from each ot
All types in the .NET Framework are either value types or reference types.
Common Language Specification■
CLS is a set of basic language features that .Net Languages needed to develop Applications and Services.
It is a subset of the CTS. The CLS establishes the minimum set of rules to promote language interoperability.
When there is a situation to communicate Objects written in different .Net Complaint languages , those objects must expose the fe
Microsoft has defined CLS, which are nothing but guidelines, that language should follow so that it can communicate with other .N

The Visual Studio .Net IDE
Which is the successor of Visual Studio 6. It eases the development process of the .Net applications(VC#.Net,VB.Net and more).
Keyword and syntax highlighting.
Intellisense(auto complete), which helps by automatically completing the syntax as you type a dot(.) with objects, enumerations ar
Project and solution management with solution explorer that helps to manage application consisting multiple files.
Help building user interface with simple drag and drop support.
Properties tab that you to set different properties for multiple windows and controls.
Standard debugger that allows you to debug your program using putting break points for observing run-time behavior.

Hot compiler that checks the syntax of code as type it and error notification.
Compiling and building applications.
Program Execution with or without the debugger.
Deploying .Net application over the internet or to disk.

Projects and Solutions
A project is a combination of executable and library files that make an application or module. A project information is usually place
A solution is a placeholder for different logically related projects that make some application. For example, a solution may consist
Toolbox, Properties and Class View tabs
The toolbox contains a number of common controls for windows, web and data applications like the text box etc.
The properties Tab allows you to set the properties on controls and forms without getting into code.
The Class View Tab shows all the classes that project contains along with the methods and fields in in tree hierarchy.

Advantages
Supports fully managed code.
Fully integrated IDE available.
WPF and WCF are the new way of buildign UI's and Communicating between processes and systems.
Linux and Mac support through 3rd parties (Mono).
Many languages available, both dynamic (IronPython and IronRuby) and static (C#, VB.NET, C++), both object oriented (C#, VB.N
Interoperability: Now developers of different languages can work easily together on the same project.
Disadvantages
- Multi platform support isn't available from MS and isn't available straight after installing Visual Studio■- Managed code can be sl

2.Windows Development using .NET
Structured programming
Structured programming is a logical programming method that is considered a precursor to object-oriented programming (OOP). S
Structured programming (sometimes known as modular programming) is a subset of procedural programming that enforces a logi
A defined function or set of similar functions is coded in a separate module or sub-module, which means that code can be loaded
Program flow follows a simple hierarchical model. These are sequence, selection, and repetition:
"Sequence" refers to an ordered execution of statements.
In "selection" one of a number of statements is executed depending on the state of the program. Keyword such if..then.. else.. enc
In "repetition" a statement is executed until the program reaches a certain state. Keywords such as while, for
Object Oriented Programming
Real-world entities of problem represent as an object. Objects are instances of classes. It is a technique to think real world in term
The four primary concepts of object-oriented programming are encapsulation, abstraction, inheritance, and polymorphism.

Class
Classes are special kinds of templates from which we can create objects.A class describes all the attributes of objects, as well as

Object
An object can be considered a "thing" that can perform a set of related activities.

Encapsulation: the wrapping up of data and function into a single unit(class) is called encapsulation.
Data is not accessible to the outside world and functions which are wrapped in the class can access it. Function provide interface
Abstraction:refers for representing essential features without including the background details of explanations.
Inheritance: is the process by which objects of one class acquire properties of objects of another class
Polymorphism:means one name ,multiple forms

C#
C# is an object-oriented language and fully supports the object-oriented programming concepts of inheritance, polymorphism, enc

It is similar to c++ and Java. C# code is made up of a series of statements, each of which is terminated with a semicolon.
C# is a block-structured language, meaning that all statements are part of a block of code. These blocks which are delimited with
C# code is case-sensitive.
C# is a general-purpose, object-oriented, type-safe programming language
`There were two ways of commenting C# code. (//, /*….*/ and special comment ///).
used for writing applications of any type.
#region and #endregion keywords, which defines the start and end of a region of code that can be expanded and collapsed.
is actually preprocessor directive

Using keyword allow to access the all classes of "System" namespace
Simple Types
Int, long, byte, float, char, bool, string
Variable and Function declaration
<accessibility_level><type><variable_name>;
Example
int a=2;
string myString;
<accessibilityLevel><returnType><functionName>(paramType paramName, ...)
{
Logic here
}
Example
public int Sum(int a, int b)
{
Return a+b;
}
Naming Conventions
Generally two naming conventions in .Net Framework
PascalCase and camelCase
Private viables(fields of class)name are generally in camelCase
age, firstName
Methods and public varibles name are generally inPascalCase
Above example. We can call function Sum()
Escape sequence(\)
String myString="\"myInteger\" is "; if you myString=""myInteger" is " then compiler error
\' ■single quotation mark
\n■new line
\t■ horizontal tab
\\■backslash
"C:\\MyDocument\\MyFile.doc"
@"C:\MyDocument\MyFile.doc"
String formatting
MessageBox.Show(string.Format("{0} {1}.", myString, myInteger));
String is actually template into which you insert the contents of variables. Each set of curly brackets in the string is a placeholder.
Namespace
Way of providing containers for application code. We can define sub namespace within parent namespace.
namespace CosMos
{
Code here
}
Assignment operators
var1 +=var2; var1 is assigned value that is the sum of var1 and var2. Similarly for other operators(-,*,/,%)
Flow Control
Controlling program flow(order of execution of lines of C# code)
Branching:execute code conditionally, depending on the outcome of an evaluation.
Looping: repeatedly executing the same statements.
Boolean Logic:bool types are used to store the result of a comparison.
Branching
The ternary operator
The if statement
The switch statement
Ternary operator
<test> ? <resultIfTrue> : <resultIfFlase>
Eg.string str=(myInteger<10) ? "Less than 10" : "Greater than or equal to 10";
Interrupting Loops
breake: Causes the loop to end immediately.
continue:Cauesed the current loop cycle to end immediately(execution continues with the next loop cycle)
return: Jumps out of the loop and its containinfuction.
goto: allows jumping out of a loop to a labeled position(generally not used)

Type Conversion

Declaring Arrays
<baseType>[ ] <name>;
Eg.int[] myIntArray;
string[ ] friendNames={"Ram ","Hari","Sita"}
Foreach loops
foreach(<baseType><name> in <array>)
{
// can use <name> for each element
}
Eg.
string[ ] friendNames={"Hari","Ram","Sita"} ;
foreach(stringfriendName in friendNames)
{
MessageBox.Show(friendName);
}

String Manipulation

Eg. stringmyString="A string";
char[ ] myChars=myString.ToCharArray();
string convertToLower=myString.ToLower();
int length=myString.Length;

string myString=="This is dotnet lab";
char[ ] separator={''};
string[ ] myWords=myString.Split(separator);
string takeSubString = myString.Substring(0,10);
String Interpolation
int x = 4;
■Console.Write ($"A square has {x} sides");
String.Format
we provide a composite format string followed by each of the embedded variables.
Eg. ■string composite = "Your username:{0} and password:{1} ";
■string s = string.Format (composite, "user123", "pass123");

Reference and Value parameters
Value parameter
In this you have passed a value into a variable used by the function.
 Void ShowDouble(intval)

Reference Parameter
Void ShowDouble(ref intval):unassigned variable is illegal
Void ShowDouble(out intval): unassigned variable is legal

Overloading functions
Same function name with different signature that means same function name but different in paramete type or number of paramet
//overloaded functions
int sum;
decimal dsum;

public void Add(inta,intb)
{
	sum = a+b;
 }
public void Add(decimala,decimalb)■{■	dsum = a+b;■ }
Complex Variable Types

Enumerations:Variable types that have a user-defined discrete set of possible values that can be used in a human-readable way
An enum is a special "class" that represents a group of constants (unchangeable/read-only variables).
To create an enum, use the enum keyword (instead of class or interface), and separate the enum items with a comma:
Structures:Composite variable types made up of a user-defined set of other variable types
Arrays: Types that hold multiple variables of one type, allowing index access to the individual values.
Defining Enumerations
enum typeName
{
value1,
value2,
value3
}
typeName varName=typeName.value;
eg.
enum Orientation
{
■north=1,
■south=2,
■east=3,
■west=4
}
Orientation myDirection=Orientation.north;
Defining Structs
struct<typeName>
{
■<memberDeclarations> this contain <accessibility><type><name>
}
Eg.
struct route
{
public orientation direction;
public double distance;
}
Classes and Objects
Classes and structs have members that represent their data and behavior. Those members include:
Fields
Fields are instances of objects that are considered part of a class, normally holding class data. For example, a calendar class may
public class CalendarDate
{
publicintmonth;
}
Properties
A property can provide protection for a class field to keep it from being changed without the object's knowledge.
publicclassPerson
{
//classic way of defining a property
private string _FirstName;
public string FirstName
{
get{return_FirstName;}
set{_FirstName = value;}
}
//simple/compact way of defining a property with get/set operations //internally/implicitly maintains a private variable
publicstringLastName { get; set; }

Methods
Are used to give access to functionality of object .Methods define the actions that a class can perform. Method can take paramete

Events
Events are a way of providing notifications about occurrences, such as button clicks or the successful completion of a method, to

Constructors
A constructor has the same name as the class .Class have internally default constructor and we can also define constructor.

```
public class Student
{
//Constructor without parameter:
public Student() { }
public Student(int ID) { }
}■
// object instantiation■   StudentobjStudent = newStudent();
```

By default, classes inherit from the System.Object type.

Methods type
static: accessible through class name. not the object instance.
virtual:Method may be overridden in derived class.
abstract:Method must be overridden in non-abstract derived class(permitted only on abstract class).
override: Method overrides a base class method.

Partial classes
Many developers need access to the same class, then having the class in multiple files can be beneficial. The partial keywords all
There are some rules for defining a partial class as in the following;
A partial type must have the same accessibility.
Each partial type is preceded with the "partial" keyword.
If the partial type is sealed or abstract then the entire class will be sealed and abstract.
Eg.

```
public partial class partialclassDemo■{
}
```

Static classes■A static class is declared using the "static" keyword. If the class is declared as static then the compiler never creat

```
public static class staticDemo■{■       //static fields■       public static int a=10,b=15,sum;
       //static method■       public static void Add()■       {■           sum =a+b;■       }■}
//function calling directly■           staticDemo.Add();
```

Abstract Classes■C# allows both classes and functions to be declared abstract using the abstract keyword. We can't create an in

```
   //derived class■    public class test :Employess■    {■       //abstract class method implementation■       public override voiddis
sealed class SealedClass■{■       void myfunv();■}
 public class test :SealedClass//wrong. will give compilation error■{■}
```

OOPs another property
Inheritance

```
publicclassPerson
{
//classic way of defining a property
privatestring_FirstName;
publicstringFirstName
{
get{return_FirstName;}
set{_FirstName = value;}
}
//simple/compact way of defining a property with get/set operations //internally/implicitly maintains a private variable
publicstringLastName { get; set; }

publicclassEmployee : Person //"Employee" class, now contains all properties of "Person" class
{
publicintEmployeeID { get; set; }
//the "FirstName" and "LastName" properties in "Person" classare automatically carried into this class.
}
Employee objEmployee = newEmployee();
objEmployee.EmployeeID = 100;
objEmployee.FirstName = "Ram";
objEmployee.LastName = "Shreshtha";
```

Multiple Inheritance
Multiple inheritance in .NET framework cannot be implemented with classes, It can only be implemented with interfaces.
Interface■An interface is a set of related functions that must be implemented in a derived class. Members of an interface are impl
It is used to achieve multiple inheritance which can't be achieved by class. An interface is a completely "abstract class", which can
Using interface-based design concepts provides loose coupling, component-based programming, easier maintainability, makes yo
Although there are several differences as in the following;
An Abstract class can contain some implementations but an interface can't.
An Interface can only inherit other interfaces but abstract classes can inherit from other classes and interfaces.
An Abstract class can contain constructors and destructors but an interface can't.
An Abstract class contains fields but interfaces don't.

```
interfaceIControl
{
void Paint();
}
interfaceISurface
{
void Paint();
}
classSampleClass :IControl, ISurface
{
// BothISurface.Paint and IControl.Paint call this method.
publicvoid Paint()
{
}
}
```

Polymorphism■Polymorphism is the ability to treat the various objects in the same manner. It is one of the significant benefits of in

```
public abstract class Employee■    {■        public virtual void LeaderName()■        {■        }■    }
   public class hrDepart :Employee■    {■        public override void LeaderName()■        {■            Console.WriteLine("Mr. jone");
hrDepart obj1 = new hrDepart();■itDepart obj2 = new itDepart();
 obj1.LeaderName();■ obj2.LeaderName();■Windows Form
```

We can create a userinterface by dragging and dropping controls from a toolbox to your form
Common controls
Controls for displaying information to the user, such as the Label and LinkLabel controls.
Controls for triggering events, such as the Button control.
Controls tha allow you to have the user of you application enter text, such as the TextBox control.
Controls that allow you to inform the user of the current state of the application and allow the user to change that state, such as Ra
Controls that allow you to display lists of information, such as the ListBox and ListView controls.
Controls that allow you to group other controls together, such as the Groupbox.
Menu control.
Properties
All controls have a number of properties that are used to manipulate the behavior of the control.
Events
Events generated by Windwos Form controls.These events are usually associated with the actions of the user. Example when use
Click:Occurs when control is clicked. Sometime, this event will also occur when user press the Enter key.
KeyPress: Occurs when a key is pressed while control has focus.
MouseMove: Occurs continually as the mouse travels over the control.This event will also occur when a user presses the Enter ke

What is an Exception
An exception is an error condition or unexpected behavior encountered by an executing program during runtime.
What is Exception Handling
Exception handling is an in built mechanism in .NET framework to detect and handle run time errors.
Exceptions are handled in C# using the 'try/catch/finally' statements.
Finally block is executed in every time.
Exceptions can be explicitly generated by a program by using the throw keyword.

```
try■{■// this code may cause an exception. If it does cause an exception, the execution will not continue. ■// Instead,it will jump to
thrownewException("Customise error message");■}
finally
{
■//Release resources.For example to close any streams or files that were opened in the try block.
}
```

Delegate
A delegate is a type that enables you to store references to functions. Delegates are declared much like functions, but with no fun
Eg.

```
delegate double MyDelegate(double param1, double param2);
static double Add(double param1, double param2)
{
return param1 + param2;
}
static double Subtract(double param1, double param2)
{
return param1 - param2;
}

MyDelegate md;
if(input=="M")
md=new MyDelegate(Add);
else
md=new MyDelegate(Subtract);
double d= MyDelegate(3,4)
```

Event and Event handling
Events are certain actions that happen during execution of program that the application wishes to be notified about, so it can resp

```
public delegate void MessagedHandler(string messageText);

public event MessageHandlermessageArrived;
if (messageArrived!=null)
{
messageArrived("Hello friend");
}
```

2.Adding new class,Display, stored in Display.cs

```
public class Display
{
public void DisplayMessage(string message)
{ MessageBox.Show("MessageArrived:"+message);
}
```

4. Display objDisplay=new Display();

Class Library Project
A project that contains only classes(but no entry point) is called class library. Class Library projects compile into .dl assemblies an
How to add Class Library Project
Click on Add new project and select the Class Library.
What is HTML
HTML is a language for describing web pages.
HTML stands for Hyper Text Markup Language
HTML is a markup language
A markup language is a set of markup tags
The tags describe document content
HTML documents contain HTML tags and plain text
HTML documents are also called web pages

```
<!DOCTYPE html>■<html>■<body>■<h1>My First Heading</h1>■<p>My first paragraph.</p>■</body>■</html>
```

The DOCTYPE declaration defines the document type
The text between <html> and </html> describes the web page
The text between <body> and </body> is the visible page content
The text between <h1> and </h1> is displayed as a heading
The text between and is displayed as a paragraph
JavaScript is a Scripting Language
A scripting language is a lightweight programming language.JavaScript is programming code that can be inserted into HTML page

The <script> and </script> tells where the JavaScript starts and ends.
Scripts can be in the <body> or in the <head> section of HTML, and/or in both.

```
<!DOCTYPE html>■<html>
<head>■<script>■function myFunction()■{■document.getElementById("demo").innerHTML="My First JavaScript Function";■}■<
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>■</html>
```

Scripts can also be placed in external files.External JavaScript files have the file extension .js.

```
<!DOCTYPE html>■<html>■<body>■<script src="myScript.js"></script>■</body>■</html>
```

ASP
Active Server Pages (ASP), also known as Classic ASP or ASP Classic, was Microsoft's first server-side script engine for dynamic
Web pages with the .asp file extension use ASP
VBScript
The interpreter replaces all the code in between the <% and %> tags. In the example below Response.WriteNow() dynamically re

```
<html>
<head>
<title>The current time</title>
</head>
<body>
The server's current time:<br />
<%
Response.WriteNow()
%>
</body>
</html>
```

ASP.NET
ASP.NET is a server-side Web application framework designed for Web development to produce dynamic Web pages. It was dev
ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applicat
ASP.NET Page Life-Cycle Events
PreInit
Raised after the start stage is complete and before the initialization stage begins.
Use this event for the following:
•■Check the IsPostBack property to determine whether this is the first time the page is being processed. The IsCallback and IsCr
•■Create or re-create dynamic controls.
•■Set a master page and Theme dynamically.
•■Set initialization.
If the request is a postback, the values of the controls have not yet been restored from view state. If you set a control property at t
Init
Raised after all controls have been initialized and any skin settings have been applied. The Init event of individual controls occurs
PreLoad.
Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Req
Load
The Page object calls the OnLoad method on the Page object, and then recursively does the same for each child control until the
Use the OnLoad event method to set properties in controls and to establish database connections.
PreRender
Raised after the Page object has created all controls that are required in order to render the page, including child controls of comp
Use the event to make final changes to the contents of the page or its controls before the rendering stage begins.
Render
It's now time to send the output to the browser. If you would like to make some changes to the final HTML which is going out to th
This is not an event; instead, at this stage of processing, the Page object calls this method on each control. All ASP.NET Web ser
If you create a custom control, you typically override this method to output the control's markup. However, if your custom control in
Unload
Raised for each control and then for the page. Page object is unloaded from the memory.
In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections.
For the page itself, use this event to do final cleanup work, such as closing open files and database connections, or finishing up lo

ASP.NET - Server Controls
Types of server controls:
HTML Server Controls - Traditional HTML tags
Web Server Controls - New ASP.NET tags
Validation Server Controls
ASP.NET - HTML Server Controls
HTML server controls are HTML tags understood by the server.
HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a runat="server" at
Note: All HTML server controls must be within a <form> tag with the runat="server" attribute. The runat="server" attribute indicates
<html>■<body>■<form runat="server">■Visit cosmos!■</form>■</body>■</html>
ASP.NET - Web Server Controls
Web server controls are special ASP.NET tags understood by the server.
Web server controls are also created on the server and they require a runat="server" attribute to work.
The syntax for creating a Web server control is:
<asp:control_name id="control_id" runat="server" />
Example
<asp:Button ID="Button1" runat="server" Text="Submit" onclick="Button1_Click" />
ASP.NET - Validation Server Controls
Validation Server Controls - For input validation
Validation server controls are used to validate user-input. If the user-input does not pass validation, it will display an error messag
Each validation control performs a specific type of validation (like validating against a specific value or a range of values).
By default, page validation is performed when a Button, ImageButton, or LinkButton control is clicked. You can prevent validation
The syntax for creating a Validation server control is:
<asp:control_name id="control_id" runat="server" />
Example
<asp:TextBox ID="TextBox1" runat="server" ></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ErrorMessage="invalid" ControlToValidate="TextBox1"

Introduction to ADO.NET
ADO.NET is an object-oriented set of libraries that allows you to interact with data sources. Commonly, the data source is a datab
Data Providers
ADO.NET allows us to interact with different types of data sources and different types of databases. Since different data sources e
ADO.NET provides a relatively common way to interact with data sources, but comes in different sets of libraries for each way you
Table 1. ADO.NET Data Providers are class libraries that allow a common way to interact with specific data sources or protocols.
ADO.NET Objects
The SqlConnection Object
The SqlConnection object creates a link (or connection) to a specified data source. This object must contain the necessary informa
SqlConnection conn = new SqlConnection("Data Source=server_name;Initial Catalog=database_name;Integrated Security=True"
The SqlCommand Object
The SqlCommand object uses the Connection object to execute SQL queries. These queries can be in the form of inline text, stor

■
The SqlDataReader Object
The SqlDataReader object is a simple forward-only and read-only cursor. It requires a live connection with the data source and pr
Example:

```
protected void Page_Load(object sender, EventArgs e)
{
if (!Page.IsPostBack)
{
SqlDataReaderMyReader;
SqlConnectionMyConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["ConnString"].ConnectionString);
SqlCommandMyCommand = new SqlCommand();
MyCommand.CommandText = "SELECT * FROM Student";
MyCommand.CommandType = CommandType.Text;
MyCommand.Connection = MyConnection;
MyCommand.Connection.Open();
MyReader = MyCommand.ExecuteReader(CommandBehavior.CloseConnection);
gvCustomers.DataSource = MyReader;
```

```
gvCustomers.DataBind();
MyCommand.Dispose();
MyConnection.Dispose();
}
}
```

The SqlDataAdapter Object

The SqlDataAdapter is a special class whose purpose is to bridge the gap between the disconnected DataTable objects and the p
The commonly used properties offered by the SqlDataAdapter class.

The SqlDataAdapter class also provides a method called Fill(). Calling the Fill() method automatically executes the command prov

DataSet

DataSet objects are in-memory representations of data. They contain multiple Datatable objects. The DataSet is specifically desig

DataTable

The DataTable object represents a logical table in memory. It contains rows(DataRow), columns(DataColumn), primary keys, cons
Example:

```
protected void Page_Load(object sender, EventArgs e)
{
if (!Page.IsPostBack)
{
DataSetMyDataSet = new DataSet();
SqlConnectionMyConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["ConnString"].ConnectionString);
SqlCommandMyCommand = new SqlCommand();
MyCommand.CommandText = "SELECT TOP 5 * FROM CUSTOMERS";
MyCommand.CommandType = CommandType.Text;
MyCommand.Connection = MyConnection;
SqlDataAdapterMyAdapter = new SqlDataAdapter();
MyAdapter.SelectCommand = MyCommand;
MyAdapter.Fill(MyDataSet);
gvCustomers.DataSource = MyDataSet.Tables[0];
gvCustomers.DataBind();
MyAdapter.Dispose();
MyCommand.Dispose();
MyConnection.Dispose();
}
}
```

SqlParameter

To add parameter/s in queriesusingSqlParameter class and providing it thenecessary information, such as parameter name, value

```
SqlCommandMyCommand = new SqlCommand();
SqlParameterIDParam= new SqlParameter();
IDParam.ParameterName = "@ID";
IDParam.SqlDbType = SqlDbType.Int32;
IDParam.Direction = ParameterDirection.Input;
IDParam.Value = "2";
MyCommand.Parameters.Add(IDParam);
```

State Management

Client-Based State Management Options

View State

View state is the method that the ASP.NET page framework uses to preserve page and control values between round trips
View state is used automatically by the ASP.NET page framework to persist information that must be preserved between postbac

```
ViewState["UserId"]=1
int  userId=(int)ViewState["LanguageId"]
```

Cookies

ASP.NET includes two intrinsic cookie collections. The collection accessed through the Cookies collection of HttpRequest contain
After you add a cookie by using the HttpResponse.Cookies collection, the cookie is immediately available in the HttpRequest.Coo

```
HttpCookie objcook = newHttpCookie("username", txtUserName.Text);
objcook.Expires = DateTime.Now.AddDays(7);
Response.Cookies.Add(objcook);
string username= Request.Cookies["username"].Value
```

Query Strings
A query string is information that is appended to the end of a page URL
http://www.cosmos.com/editstudent.aspx?id=100

int id= Convert.ToInt32(Request.QueryString["id"])

Server-Based State Management Options
ASP.NET session state enables you to store and retrieve values for a user as the user navigates ASP.NET pages in a Web applic
Session["USerName"] = txtUserName.Text;
string userName= Session["USerName"].ToString();

File I/O and Stream
Can manage the files on the local file system by using the frameworks System.IO namespace.
Can read from and write different data formats to memory and the local system by using various Stream classes within the framew
Can use the .Net framework to communicate with other commputers across the Internet using common protocols like HTTP and F
The DriveInfo Class
To create a DriveInfo object and display local drive information on a web page
DriveInfo drive=new DriveInfo(@"C:\")
drive.Name, drive.DriveType.ToString(), drive.DriveFormat; drive.TotalFreeSpace.ToString(), drive.TotalSize.toString()
The Directory and DirectoryInfo Classes■
To work with the file system directories.
The Directory class exposes static methods and can use to create, move and delete directories.
Example: Directory.CreateDirectory("C:\\Dotnet"), Directory.Exists("C:\\Dotnet"), Directory.Delete("C:\\Dotnet").
The DirectoryInfo represents a specific directory and you can perform many of the same actions as the Directory class on specific
DirectoryInfo directory=new DirectoryInfo(path), DirectoryInfo[] directories=directory.GetDirectories();
File and FileInfo
File: a static utility class that exposes many static methods for moving, copying and deleting files.
Methods:Copy(), Create(), Delete(), Open(), Move()
FileInfo: Represents a physical file on disk, and has methods to manipulate this file.
FileInfo aFile=new FileInfo(@"C:\test.txt")
aFile.Exists, aFile.Name, aFile.FullName, aFile.Extension, aFile.CreationTime
FileStream
Represents a file that can be written to or read from, or both. This file can be written to and read from.
FileStream aFile=new FileStream(filename, FileMode.Member);
FileStreamaFile=new FileStream(filename, FileMode.Member, FileAccess.Member);
Example 1.
System.IO.FileStream fs = new System.IO.FileStream(
Server.MapPath("TextFile.txt"),System.IO.FileMode.Open);
byte[] data = new byte[fs.Length];
fs.Read(data, 0, (int)fs.Length);
fs.Close();
this.lblResult.Text= ASCIIEncoding.Default.GetString(data);
Example 2.
System.IO.FileStreamfs = new System.IO.FileStream(Server.MapPath("TextFile.txt"),
System.IO.FileMode.Append, System.IO.FileAccess.Write,System.IO.FileShare.Read, 8, System.IO.FileOptions.None);
byte[] data = System.Text.Encoding.ASCII.GetBytes("This is an additional string");
fs.Write(data, 0, data.Length);
fs.Flush();
fs.Close();
this.lblResult.Text = ASCIIEncoding.Default.GetString(data);
StreamWriter: Writes character data to a stream and can be created by using a FileStream as base
Example
StreamWriter sw = new StreamWriter(File.Open(MapPath("TextFile.txt"),FileMode.Open));
sw.Write("This is a string");
sw.Close();
StreamReader: Reads character data from a stream and can be created by using a FileStream as base
Example
StreamReader sr = new StreamReader(File.Open(MapPath("TextFile.txt"),FileMode.Open));
string tmp = sr.ReadToEnd();

sr.Close();

Introduction to XML
XML stands for EXtensible Markup Language
XML is designed to transport and store data.
XML tags are not predefined. You can define your own tags
The Difference between XML and HTML
XML was designed to transport and store data, with focus on what data is
HTML was designed to display data, with focus on how data looks
Example
<bookstore>■ <book category="COOKING">■ <title lang="en">Everyday Italian</title>■ <author>Giada De Laurentiis</auth
Well Formed XML Documents
A "Well Formed" XML document has correct XML syntax.
The syntax rules were described in the previous chapters:
XML documents must have a root element
XML elements must have a closing tag
XML tags are case sensitive
XML elements must be properly nested
XML attribute values must be quoted
Namespace:■System.Xml
Classes:■ XmlDocument, XmlElement, XmlNode
XmlDocument objDoc = new XmlDocument();
objDoc.Load(HttpContext.Current.Server.MapPath("~/students.xml"));

Creating XML from programming(dynamic) of Student information
XmlDocument xmlDoc = newXmlDocument();
XmlElement xmlRoot = xmlDoc.CreateElement("students");
    xmlDoc.AppendChild(xmlRoot);

XmlElement xmlStudent = xmlDoc.CreateElement("student");
    xmlRoot.AppendChild(xmlStudent);

XmlElement xmlName = xmlDoc.CreateElement("Name");
    xmlName.InnerText = "Ram Shreshtha";
    xmlStudent.AppendChild(xmlName);

XmlElement xmlDOB = xmlDoc.CreateElement("DOB");
    xmlDOB.SetAttribute("date", "yyyyMMDD");
    xmlDOB.InnerText = "20010214";
    xmlStudent.AppendChild(xmlDOB);
xmlDoc.Save("~/students.xml");
Web Service
A web service is a method of communication between different applications. Or a software system designed to support interopera
The basic Web services platform is XML + HTTP.
XML provides a language which can be used between different platforms and programming languages and still express complex
Web services platform elements:
SOAP (Simple Object Access Protocol)
UDDI (Universal Description, Discovery and Integration)
WSDL (Web Services Description Language)
What is SOAP
SOAP is an XML-based protocol to let applications exchange information over HTTP. SOAP is a protocol for accessing a Web Se
What is WSDL
WSDL is an XML-based language for locating and describing Web services.
What is UDDI
UDDI is a directory service where companies can register and search for Web services.
UDDI:
is a directory for storing information about web services
is a directory of web service interfaces described by WSDL
communicates via SOAP.
is built into the Microsoft .NET platform.

Namespace: System.Web.Services
Class: WebService
Web Service file extension is .asmx
The only difference from a normal application is that this function is defined as a "WebMethod".
Use "WebMethod()" to convert the functions in your application into web services:
Example

```
public class WebService : System.Web.Services.WebService
{
[WebMethod]
publicstring HelloWorld()
   {
return"Hello World";
   }
}
```

Return from web service

```
<?xml version="1.0" encoding="utf-8"?>
<string>Hello World</string>
```

GDI+(Graphics Device Interface)
GDI+ is a technology that developers generally associate with Windows Forms applications because they use it to draw anything
We can also use GDI+ in ASP.NET Web applications whenever you want to serve up dynamic images. You can use GDI+ to crea
GDI+ is the .NET Framework wrapper assembly for Microsoft's GDI (Graphics Device Interface) technology. In simple terms, you
GDI+ resides in System.Drawing.dll assembly. namespaces are System.Drawing, System.Drawing.Imaging etc.
The Graphics Class
GDI+ uses a Graphics object as the main means to draw on a surface. The Graphics object has methods including DrawLine() or

Once you have the Graphics reference, you can call any of this class's members to draw various objects. Here are some of Graph

Graphics Objects
After creating a Graphics object, you can use it draw lines, fill shapes, draw text and so on. The major objects are:

The Pen Class
A pen draws a line of specified width and style. You always use Pen constructor to create a pen. The constructor initializes a new
Initializes a new instance of the Pen class with the specified color.
public Pen(Color);
Initializes a new instance of the Pen class with the specified Brush.
public Pen(Brush);
Initializes a new instance of the Pen class with the specified Brush and width.
public Pen(Brush, float);
Initializes a new instance of the Pen class with the specified Color and Width.
public Pen(Color, float);
Here is one example:
Pen pn = new Pen(Color.Blue); or Pen pn = new Pen(Color.Blue, 100);
Some of its most commonly used properties are:

The Color Structure
A Color structure represents an ARGB color. Here are ARGB properties of it:

The Font Class
The Font class defines a particular format for text such as font type, size, and style attributes. You use font constructor to create a
Initializes a new instance of the Font class with the specified attributes.
public Font(string, float);
Initializes a new instance of the Font class from the specified existing Font and FontStyle.
public Font(Font, FontStyle);
Where FontStyle is an enumeration and here are its members:

Here is one example:

Graphics g ;
Font font = new Font("Times New Roman", 26);
Some of its most commonly used properties are:

The Brush Class

The Brush class is an abstract base class and cannot be instantiated. We always use its derived classes to instantiate a brush ob

Here is one example:
LinearGradientBrush lBrush = new LinearGradientBrush(rect, Color.Red, Color.Yellow, LinearGradientMode.BackwardDiagonal);
OR

Brush brsh = new SolidBrush(Color.Red), 40, 40, 140, 140);
The SolidBrush class defines a brush made up of a single color. Brushes are used to fill graphics shapes such as rectangles, ellip
The TextureBrush encapsulates a Brush that uses an fills the interior of a shape with an image.
The LinearGradiantBrush encapsulates both two-color gradients and custom multi-color gradients.
The Rectangle Structure
public Rectangle(Point, Size); or public Rectangle(int, int, int, int);
The Rectangle structure is used to draw a rectangle on WinForms. Besides its constructor, the Rectangle structure has following m

Its constructor initializes a new instance of the Rectangle class. Here is the definition:

public Rectangle(Point, Size); or public Rectangle(int, int, int, int);

The Point Structure
This structure is similar to the POINT structure in C++. It represents an ordered pair of x and y coordinates that define a point in a
Here is how to instantiate a point structure:
Point pt1 = new Point(30, 30);
Point pt2 = new Point(110, 100);

Some sample Examples:

```
protected void Page_Load(object sender, EventArgs e)
{
Bitmap bm = new Bitmap(300, 150);
Graphics g = Graphics.FromImage(bm);
g.Clear(Color.White);
//g.DrawLine(Pens.Red, 10, 10, 150, 150);
//g.FillRectangle(new SolidBrush(Color.Red), 10, 10, 150, 100);
g.DrawString("Binod Thapa", new Font(FontFamily.GenericSerif,34,FontStyle.Bold), Brushes.Green, 10, 40);

// Now, we only need to send it // to the client.This save image in memory
Response.ContentType = "image/jpeg";
bm.Save(Response.OutputStream, ImageFormat.Jpeg);
Response.End();

//bm.Save(Server.MapPath("~/uploads/") + "sss.jpeg",ImageFormat.Jpeg);//save in directory

// Cleanup
g.Dispose();
oCanvas.Dispose();
}
```

Reflection
Reflection is able to find out details of an object, method, and create objects and invoke methods at runtime. The System.Reflectio

```
Public class MyClass2
{
  int answer;
  public MyClass2()
  {
    answer = 0;
  }
```

```
  public int AddNumb(intnumb1, intnumb2)
  {
    answer = numb1 + numb2;
    return answer;
  }
}
Type type1 = typeof(MyClass2);
```

Windows Service Applications

Microsoft Windows services enable us to create long-running executable applications that run in their own Windows sessions. The
We can then use the Services Control Manager to start, stop, pause, resume, and configure your service.
Using Microsoft Visual Studio or the Microsoft .NET Framework SDK, We can easily create services by creating an application tha

Threading
A thread is a sequence of instructions executed within the context of a process. MultiThreading is achieved when a program uses
Namespace: System.Threading

```
Thread.Sleep(60000);//in mimiseconds
using System;
usingSystem.Threading;

public class ThreadTest
{
   // This method that will be called when the thread is started
public void test()
   {
      //code here for logic
   }
}
Using thread
ThreadTestobj = new ThreadTest ();
Thread oThread = new Thread(new ThreadStart(obj.test));
//to start thread
oThread.Start();
// Spin for a while waiting for the started thread to become
      // alive:
while (!oThread.IsAlive);

      // Put the Main thread to sleep for 1 millisecond to allow oThread
      // to do some work:
Thread.Sleep(1);

      // Request that oThread be stopped
oThread.Abort();
```