2) Library :-

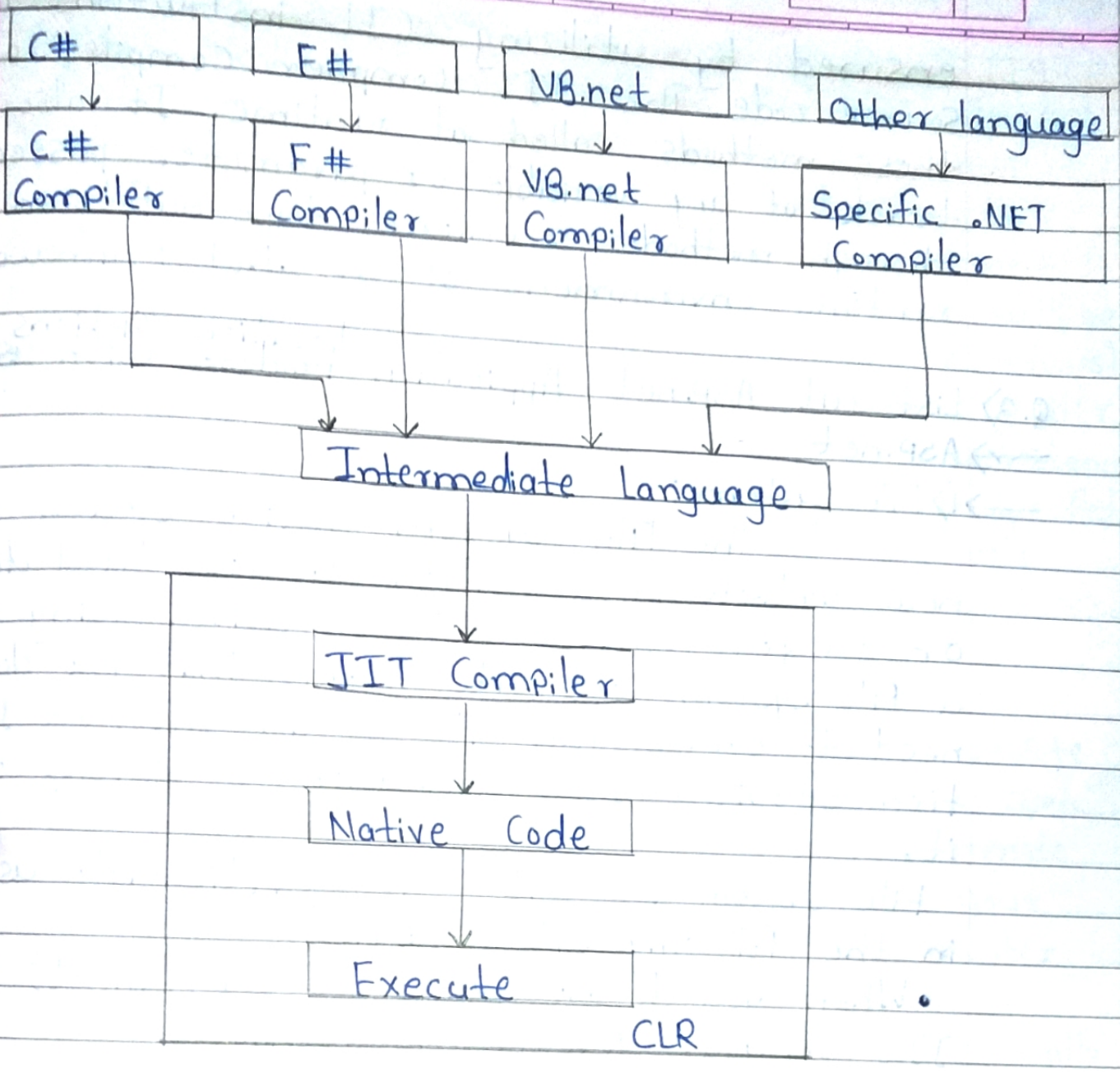
The .NET Framework includes a set of standard classlibraries. The most common library used for web application in .net is the web library. The web library has all the neccessary components used to develop .Net web based applications.

3) Common Language Runtime :-

The Common Language Infrastructure or CLI is platform .Net programs are executed on this platform. The CLR is used for performing key activities. Activities include Exception handling and Garbage Collection.

Q.2) Explain ASP.Net Page Compilation.

→ The first step is converting the source code to intermediate language (IL) by a language-specific Compiler. The second step is converting the IL to machine instructions. The main difference with the explicit Compilers is that only executed fragments of IL code are compiled into machine instructions, at runtime. The .NET framework calls this compiler the JIT (Just-In-Time) Compiler. The JIT compiler is part of the Common Language Runtime (CLR). The CLR manages the execution of all .NET applications. In addition to JIT Compilation at runtime, the CLR is also responsible for garbage Collection, type Safety & for exception handling.



Different machine configurations use different machine level instructions. As above Fig. shows the source code is compiled to exe or dll by the .NET Compiler. Common Intermediate Language (CIL) Consists of instructions that many environment supporting .NET can execute & includes metadata describing structures of both data and code. The JIT Compiler process es the CIL instruction into machine code Specific for an environment. Program portability is

ensured by utilizing CIL instructions in the Source code. The JIT Compiler compiles only those methods called at runtime. It also keeps track of any variable or parameter passed through methods & enforces type-Safety in the runtime environment of the .NET Framework.

Q.3) List out ASP.net Application ~~Folders with usage.~~ Location options for

--> ASP.net

--> 1) File System :-

The file System option creates the new website in a location on your hard drive or on your shared network drive. Using such a file system web site means that you do not need to create your site as an Internet Informa tion Services (IIS) application to develop or test it.

File System Web Sites are particularly useful in the following Situations:

- When you do not want to (or cannot) install IIS on your development computer.
- In classroom settings, where students can store files on student-specific folders on a central server.
- In a team setting, where team member can access a common website on a central server.

File System is used till the process of develop- ment, after development it is used in FTP/HTTP Server.

2) HTTP :-

An HTTP based Website is used when you

are working with a site deployed inside of IIS (either locally or on a remote server). This website may be configured as an application. A remote server running IIS will have to provide access to your web files using Front Page Server extension. In this application will be created under IIS (Internet Information Services) Server. In Http web application is stored under IIS home directory. you will not get any instance of ASP.NET Development Server. No port is specified The request will be handled at IIS port only.

3) FTP :-

FTP location provides directly facility of creating application at FTP. The FTP based WebSite is useful when you want to Connect to your web site via FTP to manage your files on a remote Server. This option is typically used when your website is hosted on a remote Server Computer & your access to the files & folders on that Server is through FTP. Selecting FTP is essentially the Same but creates the site on a ~~remote~~ Server that is accessed using the FTP (File Transfer protocol). Choosing HTTP or FTP generally requires that you have a username & password for a location on a Server provided by web host.

FTP by default is not Secure. When a user login to a Server his/her information is sent as plain text.

Q.4) List out ASP.net Application Folders with usage.

→ The asp.net application folder Contains list of Specified

folder that you can use to store specific type of files or content in an each folder.
The root folder structure is as following

1) Bin Directory
2) App_Code
3) App_GlobalResources
4) App_LocalResources
5) App_WebReferences
6) App_Data
7) App_Browsers
8) App_Themes

1) Bin Directory :
It Contains all the precompiled .Net assemblies like DLLS that the purpose of application uses.

2) App_Code Directory :
It Contains Source Code files like .CS or .Vb that are dynamically Compiled for use in your application. These Source Code files are usually Separate Components or a data access library.

3) App_GlobalResources Directory :
It Contains to Stores global resources that are accessible to every page.

4) App_LocalResources Directory :
It is serves the same purpose as app-globalresources, except these resources are accessible for their dedicated page only.

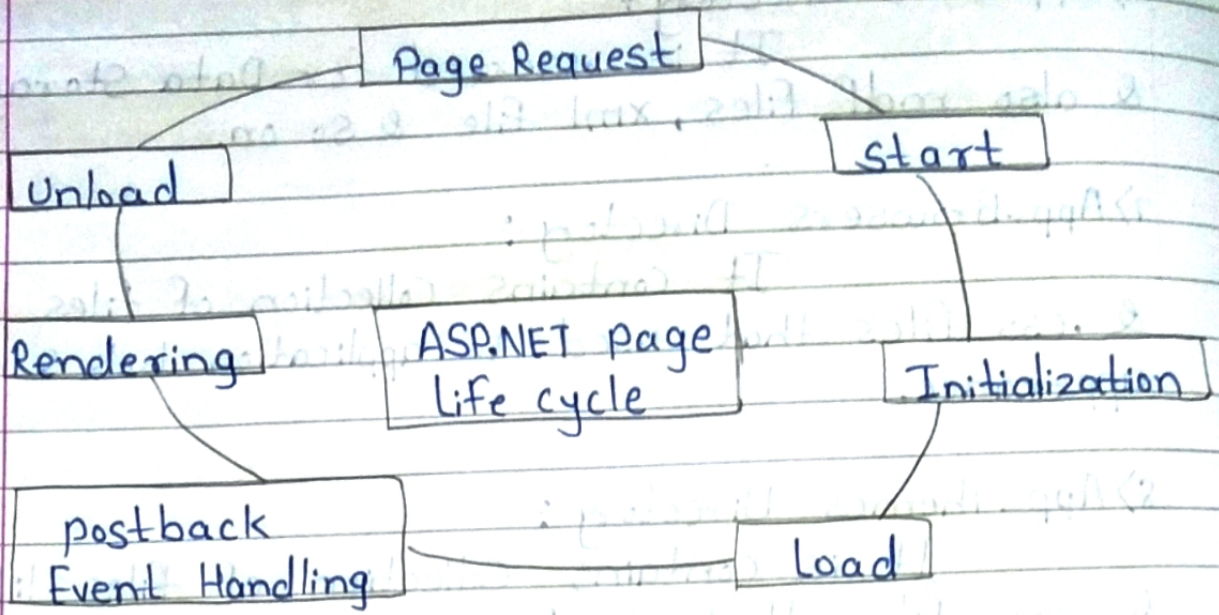
5) App_WebReferences Directory :
It is stores reference to web services that the web application uses.

6) App_Data Directory :
It is reserved for Data Storage & also mdf files, xml file & so on.

7) App_Browsers Directory :
It Contains Collection of files like .Skin & .css files that used to application look & feel apperance.

8) App_Themes Directory :
It is contains Collection of files like .Skin & .css files that used to application look & feel apperance.

Q.5) Explain ASP.net page Life Cycle.
→ When an ASP.net page runs, the page goes through a life cycle in which it performs a Series of processing steps.
These include initialization, instantiating Controls, restoring & maintaining State, running event handler Code & rendering. It is important for you to understand the page life cycle so that you can write Code at the appropriate life-cycle stage for the effect you intend.
Some parts of the life cycle occur only when a page is processed as postback. For postbacks, the page life cycle is the Same during partial-page postback (as when you use an UpdatePanel Control) as it is during a full-page postback.



1) Page Request :

The page request occurs before the page life cycle begins. When the page is requested by a user, ASP.NET determines whether the page needs to be parsed & Compiled or whether a cached version of the page can be Sent in response without running the page.

2) Start :

In the Start, page properties such as Request & Response are Set. At this Stage, the page also determines whether the request is a postback or a new request & sets the ISPost Back property. The page also Sets the UICulture property.

3) Initialization :

During page initialization, Controls on the page are available & each Controls UniqueID

property is set. A master page & themes are also applied to the page if applicable. If the current request is a postback, the Postback data has not yet been loaded & Control property values have not been restored to the values from View State.

4) Load :-

During load, if the current request is a postback, Control properties are loaded with information recovered from View State & Control State.

5) Postback event Handling :-

If the request is a postback, Control event handlers are called. After that the Validate method of all validator is called, which sets the IsValid property of individual validator Controls & of the page.

6) Rendering :-

Before rendering, View State is saved for the page & all Controls. During the rendering Stage, the page calls the Render method for each Control, providing a text writer that writes its output to the OutputStream object of the page's Response property.

7) Unload :-

The unload event is raised after the page has been fully rendered, sent to the Client & is ready to be discarded. At this point page properties such as Response & Request are unloaded & cleanup is performed.

Page No.:
Date:
YOUVA

Q.6) Explain ASP.NET page Structure.

→ There are Seven main Components of an ASP.net page Structure are as follows.

1) Directives :-

A directive Controls how the page is compiled. It is marked by tags, <%@ and %>

It can appear anywhere in a page. But, normally it is placed at the top of a page. The main types of directives are:

- Page
- Import

A page directive is used to specify the default programing languages for a page.

```
<%@ Page Language="C#" %>
```

OR

```
<%@ Language="C#" %>
```

Some namespaces are imported into an ASP.Net page by default. If you wish to use a class that is not contained in the default namespaces you must import its namespace.

```
<% Import Namespace="System.Data.SqlClient" %>
```

2) Code Declaration Blocks :-

A Code declaration block contains all the application logic for a page. It also includes declarations of global variables & functions. It must be written within the Script runat="Server" tag. The <Script> tag has two optional parameters:

1) Language
2) SRC

3) ASP.Net Controls :

ASP.net Controls can be mixed with text & Static HTML in a page. All controls must appear within a <form runat="server"> tag. Some Controls such as <Span runat="Server"> & the Label Control can appear outside this tag. You can have only one form per page in ASP.net.

4) Code Render Blocks :

If you wish to execute code within HTML, you can include the code within code render blocks. There are two types of code render blocks:

- Inline Code: It executes a statement or Series of Statements. It is marked by the characters <% and %>.
- Inline Expressions : They display the value of a Variable or method. They can be Considered as Shorthand notation for the Response.Write method. They are marked by the Characters <%= and %>.

5) Server-Side Comments :

You can add Coments in Server-side Code using the characters <%-- & --%>. The main use of these Comment blocks is to add documentation to a page.

6) Server-Side Include Directives :

You can include a file in an ASP.net page by using a Server-Side include directive. It is executed before any of the code in the page.

If the file is in the same directory or in Sub-directory of the page including the file, this directive is written as:

<!--#INCLUDE file="includedfile.aspx"-->

You can also specify the virtual path of file. To include a file located in the directory MyAspx under the wwwroot directory, you will write the directive as:

<!--#INCLUDE Virtual="/MyAspx/includedfile.aspx"-->

7) HTML Tags & literal Text:

You can build the static part of an ASP.net page using HTML tags & literal text. The HTML Content of your page is also Compiled along with the rest of the contents. The literal text has been made bold & Converted to uppercase before being rendered in the browser.

Q.7) Define Directive & State all ASP.net Directives used in web Applications.

→ Basically, pag directives are commands. These Commands are used by the compiler when the page is compiled. How to use the directives in an ASP.net page. It is not default to add a directive to an asp.net page. It is Simple to add directives to an ASP.NET page. You can write directives in the following.

<% @ [Directive] [Attributes] %>

See the directive format it starts with "<% @" & ends with "%>". The best way is

to put the directive at the top of your page. But you can put a directive anywhere in a page. One more thing, you can put more than one attribute in a single directive.

1) @ Page Directives :

It defines page specific attributes using by .net page parser & Compiler.

Syntax :

<% @ Page attribute="value" [attribute="value"] %>

e.g :

<% @ pageLanguage = "C#" AutoEventWireup="true" CodeFile = "EventTracker.aspx.cs" Inherits = "EventTracker" %>

Language - Specifies programming language used for code behind page.

AutoEventWireUp - if value is true Page_load() event will occur implicitly & if false page-load() event should occur explicitly.

CodeFile - Specifies the name for code behind pagefile.

Inherits - Specifies name of partial class used on code behind page file.

2) @ Control Directive :

It is Defines control Specifies attributes by .net page parser & Compiler. It is used only in ascx files.

Syntax :

<% @ Control attribute="value" [attribute="value".....] %>

3) @ Import Directive :

It is defines that explicitly imports a

namespace into .net application. Web page, master page user control page & global.aspx file.

Syntax:

```
<%@ Import namespace="value" %>
```

4) @Implements Directive:

It indicates that web page, master page & user control page must be implement .net interface.

Syntax:

```
<%@ Implements interface="validInterfaceName" %>
```

5) @Register Directive:

It is creates an association of two tag prefix & custom control in .net web pages.

Syntax:

```
<%@ Register tagprefix="tagprefix" namespace="name space" assembly="assembly" %>
```

6) @Assembly Directive:

Assembly is providing links to .net appln. files such as master page, user control & Global.aspx & web page. It is also help in creating all interfaces & classes.

Syntax:

```
<%@ Assembly Name="assemblyname" %>
<%@ Assembly Src="pathname" %>
```

7) @Master Directive:

It is defines attribute of a master page (.master file).

Syntax :-

<%@ Master attribute="value" [attribute="value"...] %>

8) @PreviousPageType Directive :-

Its provides strong typing against previous page.

Syntax :-

<%@ PreviousPageTypeattribute = "value" [attribute="value"... %>

9) @MasterType Directive :-

It provides reference to asp.net master page & accessed master page property.

Syntax :-

<%@ MasterType attribute="value" [attribute="value"...] %>

10) @ ~~Reference~~ OutputCache Directive :-

It is a controls output of caching policies of web page.

Syntax :-

<% OutputCache Duration="# ofseconds" Location="Any/client/Download/server/None/server & Client" Shared = True/False " %>

11) @Reference Directive

It provides user control & web page file at Virtual path.

Syntax :-

<%@ Reference Page="path to .aspx page" Control="path to .aspx file" Virtualpath = "path to file" %>

Q.9) Explain Self page posting with Ex.

→ Whenever page sends data to itself during postback is known as Self page posting.

For Example:- you have a data entry page & when you fill this page & Click on Submit button, then after saving the data, this page posts back to itself.

you will have to know the difference between postback for the first time when the new page is loaded & postback for the Second time. So postback is just posting back to the same page. Remember one thing, the postback Contains all the information Collected on the Initial page for processing.

Q.10) Explain Cross page posting with Ex.

→

```
PostBackUrl = "~/Two.aspx";
```

two.aspx

```
protected Void Page_Load (Object Sender, EventArg e)
{
if(previousPage != NULL)
{
 TextBox t1=(TextBox)previousPage.FindControl("TextBox1");
 Label2.Text = "Hello .... " + t1.Text;
 }
}
```

Q.10) Explain Cross Page Posting with Ex.

→ Cross page posting means you are posting from data to another page. This is useful when you want to post data to another page & do not want incur the overhead of reloading the current page. We can use the property PostBack Url of Button Control or redirect to another page. If you want to use the data of one page to another page without using session, object, or anything else, you can just use cross-page in your project.

```
using System;
using System.Web.UI.WebControls;

protected Void page_Load(Object Sender, EventArg e)
{
if(previousPage!=null && PreviousPage.IsCrossPagePost
        Back)
{
TextBox txtName=(TextBox)PreviousPage.FindControl(
        "textUserName");
TextBox txtlocation=(TextBox)PreviousPage.FindControl(
                        "txtLocation");
lblName.InnerText="Welcome to Default2.aspx page"+txtName;
lblLocation.InnerText="your location:"+txtlocation.Text;
}
else
{
Response.Redirect("Default.aspx");
}
}
```

Q.11) Write a note on ISPostBack property.

→

ASP.net page class has a property ISPostBack. We can check that whether the page is postback for the first time or not using ISPostBack property.

ex:-

```
pageload()
{
 if(IsPostBack==true)
 {
Response.Writeline("IsPostBack true");
 }
else
 {
Response.Writeline("IsPostBack false");
 }
}
```

Q.12) Write a note on AutoPostBack Property.

→

The AutoPostBack property is used to set or return whether or not an automatic post back occurs when the user selects an item in a list control.

If this property is set to TRUE the automatic post back is enabled, otherwise FALSE, Default is FALSE.

Syntax:-

```
<asp:SomelistControl AutoPostBack="TRUE|FALSE" runat=
```

```
"Server">

<html>
<script runat="Server">
Sub ItemChanged (Sender As Object, e As EventArgs)
    lbl1.Text = "You Selected" & rb1.SelectionItem.Text
End Sub
</script>
<body>
<form runat=Server>
<asp:RadioButtonList id="rb1" AutoPostBack="true"
 runat="Server" OnSelectedIndexChanged="ItemChanged">
   <asp:listItem Text="Item 1">
   <asp:ListItem Text="Item 2">
   <asp:listItem Text="Item 3">
   <asp:listItem Text="Item 4">
   <asp:listItem Text="Item 5">
   <asp:listItem Text="Item 6">
</asp:RadioButtonList>
<asp:Label id="lbl1" runat="Server">
</form>
</body>
</html>
```