**FULL STACK DEVELOPER**

A FULL-STACK DEVELOPER WILL HANDLE BOTH THE FRONT END(USER-SIDE) AND THE BACK END (SERVER SIDE) OF A WEBSITE OR A WEB APPLICATION, IE HE WILL HAVE GOOD KNOWLEDGE IN BOTH THE SECTION.

A FULL STACK DEVELOPER IS A COMBINATION OF

-UI DEVELOPER-designs front end or user interface of a website

-front end developer is same as UI developer but front developer mainly develops

- good knowledge in HTML5, CSS, BOOTSTRAP, JavaScript, AJAX

- .Net developer – basically for developing the web application and software’s,

- good knowledge in c#

-ASP.Net is backend and VB.Net is for front end

-NODE JS developer- Manage exchange of data between server side and user side

-he deploys and maintain a network application

-SQL developer- develops SQL database and writes application to interface with SQL database

-SQL Is for managing data in RDBMS

-Oracle developer- oracle is an RDBMS

- creates and maintain database components of an application that uses ORACLE

- based technology stack

**C#**

**C#** is a modern, type safe programming language, object oriented language

* **Object**-Oriented Language.
* Automatic Garbage Collection.
* **Cross** Platform

**ASP**.**NET**

**ASP**.**NET** is an open-source, server-side web-application **framework** designed for web development to produce dynamic web pages. ... It was developed by Microsoft to allow programmers to build dynamic web sites, applications and services. It was first released in January 2002 with version 1.0 of the

**PROGRAMMING LANGUAGES: -**

**FUNCTIONAL PROGRAMING** – Combination of variables and functions

- Used when there is more thing with less functions

- data in the functions are immutable, ie once created cannot be modified after.

- Error in any single section will affect the other sections also

- easy to understand

-more imp is for functions than data

-declarative program flow- does not check the flow of program just mind output

**OBJECT ORIENTED PROGRAMING (OOPS)**

-Program pattern that relies on Class and Object than functions and procedures

-data used here is mutable and error in any one section will not affect the others working

-java, python, c++, visual basic.Net

-more imp is for data than functions

-imperative programing- will check the steps of the program flow(how)

CLASS

- collection of objects, will have blueprint for creating objects

-provide initial values for member variables and attributes

-implementation of member functions and methods

-class have names, attributes and operations.

A **class** enables you to create your custom types by grouping variables of other types, methods

OBJECT

-instance of a class

-a real time entity, which can do a set of activities and those activities are the objects behaviour

-different classes will have different object same object can be used for calling different functions

**MAIN PILLARS OF OOPS-**

-ENCAPSULATION- process of combining data members and member functions into a single unit

ex: class.

- ABSTRACTION

– process of hiding the unwanted implementations details and showing the relevant only

- the complexity can be reducing by using abstraction

-Abstraction is done using access specifiers

=ACCESS SPECIFIERS: in .NET there is 5

1.PUBLIC – Accessible outside the class using objects

2. PRIVATE- Accessible inside the class only with member functions

3. PROTECTED – Just like private but it is also accessible in derived class also using member functions

4. Internal – visible inside the assembly and accessed using objects

5.PROTECTED INTERNAL – visible inside assembly by objects and outside assembly by member functions.

-INHERITANCE- process of deriving a new class from a base class using “ : “

- will allow us to use existing codes, which will increase the quality

-------multiple inheritance can be done only using INTERFACE in C#----------

- Private class members can be only inherited in derived class but cannot be accessed

==Method hiding: like when we inherit a class to the derived class and mention the same method mentioned in base class inside the derived class again, while referencing that method in PSVM will show the new one only.

Now method in base class will be hided and method in derived will be shown

-POLYMORPHISM – when a single message can have processed different forms in different situations

- here same method can be implemented by a class by using same name

=COMPILE TIME/OVERLOAD

-perform different tasks using same function name with different parameters

- early binding

- have nothing to do with inheritance and virtual functions

-Used when more than one function definition can be given for the same function name

-doing the same thing by taking different parameter

-we can use n number of overloading  
The rule is that overloads must be different in their signature, which means the name and the number and type of parameters

=RUNTIME /OVERRIDE

-changing functionality or behaviour of a method without changing it signature

-can override a function in base class by creating a similar function in derived class

-uses inheritance and virtual functions

- here the method will be different but then final goal will be the same

=ABSTRACT CLASS – Will contain abstract members and non-abstract members

---Abstract Method: method without any particular body

-it is only declared in the abstract class using the abstract modifiers and defined in the

Child classes using the override keyword

-we cannot create an object for an abstract class

=INTERFACE METHOD – User defined type class containing only abstract members in it

* These are implemented inside the child class by inheritance
* Any child class inheriting an abstract class have to define all the method inside the abstract class.

=CONSTRUCTORS- Used to initialise an object

-cannot be abstract

-will not have objects

- uses the class name as the function or method name