**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 developes

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

- .Net developer – basically for developing the web application and softwares,

- good knowlede 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

**PROGRAMMING LANGUAGES: -**

**FUNCTIONAL PROGRAMING** –

* Primitive that they are not mutable i.e. no assignment operation once the value is assigned after it cannot be changed
* function is primary manipulation unit
* Focus on what we are doing i.e. the path
* Here the method and its definition is kept in different places
* Step by step
* Supports abstraction over data and behaviour
* Will have pure function i.e. output is completely based on the input given

Combination of variables and functions

Used when there is more thing with less functions

variables are immutable, ie once created cannot be modified after.

Error in any single section will affect the other sections also

easy to understand

**OBJECT ORIENTED PROGRAMING (OOPS)**

Object oriented programming Is a language which uses classes and objects for its functioning, it is based on real world entities like inheritance, abstraction, polymorphism etc.

* Oops work on the concept of objects which contain data in the form of fields referred to as attributed and have code in the form of procedures referred to as methods
* Mutable
* Object is primary manipulation unit
* Here a data and its associated procedures are brought together for easy understanding
* Focus on the steps i.e. how we are doing
* Divided into objects to do specific tasks
* So if we want to do any modification we can easily identify
* Not for small programs
* Highly secure because of access specifiers
* Bottom to up approach –programmers have to create modules first
* Oops is based on the 4 key principle

1.Encapsulation:

2.abstraction: abstract and non-abstract classes—abstraction over data only

3.inheritance: multiple inheritance can occur

4.polymorphism:

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

-it is mutable and error in any one section will not affect the others working

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

CLASS

- collection of objects, will have blueprint for creating objects

-provide initial values for member variables and attributes

-implementation of member functions and methods are done here

-class have specific names, attributes and will do specific operations.

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 reduced by using abstraction

=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 “ : “ operator

- 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, it will show a compile time error

-POLYMORPHISM – when a single message can be processed in 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

=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