➢ JAVA

1. **What is Programming?**
   Programming is a way to instruct the computer to perform various tasks in a specified manner.

2. **What is Java?**
   Java is an Object – Oriented Programming language which is designed to develop all kinds of applications.
   Ex:
   Web applications.
   Desktop applications.
   Mobile applications.

3. **What are the advantages of Java?**
   a. Simple.
   b. Object – Oriented.
   c. Platform – independent.
   d. Distributed.
   e. Secure.
   f. Robust (Strong Memory Management).

4. **Disadvantages of Java?**
   a. Performance is less when compares to other languages.
   b. High memory consumption.
   c. Java is a bit costly due to its higher processing and memory requirements.

5. **What are the different technologies available in Java?**
   a. Java Standard Edition (S.E) is used to develop desktop applications.
   b. Java Enterprise Edition (E.E) is used to develop web applications.
   c. Java Micro Edition (M.E) is used to develop mobile applications.

6. **What is the Java compilation syntax in Command Prompt?**
   Javac filename.java
   Ex: javac Demo.java

7. **What is the Java execution syntax in Command Prompt?**
   After compiling, .class file will be generated with class name.
   Java .class file name.
   Ex: java Demo

8. **What is a Class in Java?**
   Class is a blue print of an Object.

9. **What is a Package in java?**
   Package is like a container which can segregate the files in the java project.

10. What are the Data Types available in Java?

Data types specify the different sizes and different type of values that can be stored in the variable. There are two types of Data Types in Java.

1. **Primitive Data types:**

   The Primitive Data Types include

   a. byte.
   b. short.
   c. int.
   d. long.
   e. float.
   f. double.
   g. char.
   h. boolean.

2. **Non-Primitive Data Types:**

   The Non-Primitive Data Types include

   a. Classes.
   b. Interfaces.
   c. And Arrays.

11. What is a Variable?

Variable is a reserved area name in the RAM at runtime.

12. What is a Method in Java?

Method is a block of code which is created for performing any action based on our requirement.

13. What is Return keyword?

If we want to return any value from the method after performing any action then we specify the return keyword.

Ex:

```
package Java;

public class Example
{

	public static int addNumbers(int a, int b)
	{
		return a+b;
	}

	public static void main(String[] args)
	{
		int result = addNumbers(3,5);
		System.out.println("Sum: "+ result);
	}
}
```

14. What are the different types of printing statements?

Ex:

```
1. System.out.println("Done");
```

```
2. System.out.print("Done");
3. System.out.printf("Done");
```

15. **What is a Constructor?**
    Constructor is a special method which is used to initialize an Object. Constructor name should be as Class name.
    There are two types of Constructors.
    1. **Default Constructor or Implicit Constructor:** Which is defined by the Compiler with zero parameterization.

    2. **User Defined Constructor or Explicit Constructor:** Which is defined by the users.

16. **What is an Object?**
    Object is an instance of a Class.

17. **Why do we create an Object?**
    To allocate a memory for a Class, we create an Object.

18. **What is a Looping Statement?**
    Looping Statements are the statements that can execute one or more statements repeatedly for several number of times.

19. **What is an Array?**
    Array is like a container which is used to store multiple values of a same data type at a time. We can store only the fixed size of elements in the Array.

    Ex 1:

```
package Java;
class EXampleArray
{
	public static void main(String[] args)
	{
		int a[]=new int[5]; //declaration and instantiation
		a[0] = 10; //initialization
		a[1] = 20;
		a[2] = 30;

		//traversing array
		for(int i=0; i<a.length; i++)
		{
			System.out.println(a[i]);
		}
	}
}
```

    Output:
    10
    20
    30

Ex 2:

```
class TestArray2
{
      public static void main(String[] args)
      {
            int a[]={1,2,3,4};

            for(int i=0; i<a.length; i++)
            {
                  System.out.println(a[i]);
            }
      }
}
```

Output:

1
2
3
4

20. What is a Comment?

Comment is a statement that is not considered by the Compiler. It can be used to provide extra information about the Variables, Methods, Classes or any.

21. What is Static?

Static is a non-access modifier and it is a keyword in Java. Static members can be accessed without creating an Object.

22. What is non-static?

The non-static members cannot be accessed directly without creating an Object.

23. Is Java Object Oriented Programming or Object Based Programming?

Object Oriented: Java is an Object-Oriented Programming language, which means one Object properties can be inherited to another Object.

Object Based: Object Based means one Object properties cannot be inherited to another Object.

24. Is Java pure 100% Object Oriented Programming Language?

No, Java is not 100% pure Object-Oriented Programming Language because of Primitive Data Types and Static keyword.

25. What is OOPS?

OOPS means Object-Oriented Programming System. That means, the data must be existed in the form of Objects. It helps the implementation of code re-usability.

26. What is Interface?

a. Interface is a one type of program.
b. It contains final variables and unimplemented methods.
c. An Interface is a blue print of a class.

d. It is like a business document which contains rules and guidelines.
e. It says that what to do but not how to do.

Ex:

```
package Java;
public interface Bank
{
      public void uSBank();
      public void ukBank();
      public void canadaBank();
}
```

27. What are the Principles of OOPS?

1. Inheritance
2. Encapsulation
3. Polymorphism
4. Abstraction

28. What is an Inheritance?

Inheritance is a process of acquiring one Class properties and behaviors to another Class from already existing class. The main concept of an Inheritance is code reusability.

29. What are the types of Inheritances?

1. **Single Inheritance:**
2. **Multi-Level Inheritance:**
3. **Hierarchical Inheritance:**
4. **Multiple Inheritance:**

   Multiple Inheritance is not possible in class level.

5. **Hybrid Inheritance:**

   Hybrid Inheritance is also not possible.

6. **Cyclic Inheritance:** Child class cannot be extended by parent class.
   1. Class ➔ Class = extends.
   2. Interface ➔ Interface = extends.
   3. Interface ➔ Class = implements.

30. What is an Encapsulation?

Encapsulation is the ability of an Object to hide its data and methods from the rest of the world.
Ex: Class

31. What is an Abstraction?

Abstraction is a process of hiding the implementation of details and showing only functionality to the users.

32. How can we achieve an Abstraction in Java?

100% abstraction can be achieved in the Interface level and some of the percentage of an abstraction can be achieved in an Abstract Class level.

33. What is Polymorphism?

Polymorphism is the ability of an Object to take on many forms.

There are two types of Polymorphisms. They are

1. **Compile-time Polymorphism or Static Polymorphism:** Compile-time Polymorphism can be achieved by using Method Overloading concept in Java.

2. **Runtime Polymorphism or Dynamic Polymorphism:** Runtime Polymorphism can be achieved by using Method Overriding concept in Java.

34. What is meant by an Access Modifier?

Access Modifiers define the scope or visibility of a Class members like Variables, Methods and Constructors.

35. List of Access Modifiers?

1. **Public:** Public members can be accessed within the Class, within the Package, within the other Package and entire the Project.

Ex:

```
package Java;
public class Modifiers
{
	public int a = 10;

	public Modifiers()
	{
		System.out.println("Constructor");
	}

	public void method()
	{
		System.out.println("Method");
	}

	public static void main(String[] args)
	{
		System.out.println("Main method");
	}
}
```

2. **Private:** Private members can be accessed within the Class only.

Ex:

```
package Java;
public class Modifiers // class should not be private
{
	private int a = 10;

	private Modifiers()
	{
		System.out.println("Constructor");
	}
```

```
	private void method()
	{
		System.out.println("Method");
	}

	public static void main(String[] args)
	{
		System.out.println("Main method");
	}
}
```

3. **Default:** Default members can be accessed within the Class and within the Package only.
   Ex:

```
package Java;
class Modifiers
{
	int a = 10;

	Modifiers()
	{
		System.out.println("Constructor");
	}

	void method()
	{
		System.out.println("Method");
	}

	public static void main(String[] args)
	{
		System.out.println("Main method");
	}
}
```

4. **Protected:** Protected members can be accessed within the Class, within the Package and with the Sub Class it may be in another Package.
   Ex:

```
package Java;
public class modifiers // Class should not be Protected
{
	protected int a = 10;

	protected modifiers()
	{
		System.out.println("Constructor");
	}

	protected void method()
	{
		System.out.println("Method");
	}
```

```
        public static void main(String[] args)
        {
              System.out.println("Main method");
        }
    }
```

36. What is an Abstract Class?

Abstract Class is a one type of Class which contains both implemented and unimplemented methods. It is defined with abstract keyword.

Ex:

```
package Java;
public abstract class Demo
{
      public abstract void method1();

      public abstract void method2();

      public void method3()
      {
            System.out.println("Implemented method");
      }

      public static void main(String[] args)
      {
            System.out.println("Done");
      }
}
```

37. When do we use Abstract class?

Whenever we are not able to fully implement the Interface then we use Abstract Class because it contains both implemented and unimplemented methods.

38. Can we create an instance (Object) for an Abstract Class?

No, we cannot create an instance for an Abstract Class because it contains both implemented and unimplemented methods.

39. Can we extend multiple Abstract Classes into a single Class?

No, it is not possible in Class level. The multiple inheritance is possible in Interface level only.

40. Can we implement the Abstract Class like an Interface?

Abstract Class is also a one type of Class. So, Abstract Class cannot be implemented it can be extended.

41. What is Method Overloading?

Method Overloading means same method names with different type of parameters like order of parameter, type of parameter, number of parameters.

Ex:

```
package Java;
public class MethodOverLoading
```

```
{
	public void method(int a, int b)
	{
		System.out.println("Int parameter");
	}

	public void method(float f1, float f2)
	{
		System.out.println("float parameter");
	}

	public static void main(String[] args)
	{
		MethodOverLoading m = new MethodOverLoading();
		m.method(10,20);
	}
}
```

Output:

Int parameter

42. What is Method Overriding?

Method Overriding means same method names with same parameters in the concept of Inheritance.

Ex:

```
public class Parent
{
	public void method(int a, int b)
	{
		System.out.println("Parent method");
	}

	public static void main(String[] args)
	{
		System.out.println("Parent class main method");
	}
}

public class Child extends Parent
{
	public void method(int a, int b)
	{
		System.out.println("child method");
	}

	public static void main(String[] args)
	{
		Child o = new Child();
		o.method(1,2);
	}
}
```

Output:
Child method

43. What is an Exception?

Exception is an event that occurs during the execution of a program that disrupt the normal flow of execution.

44. What are the different types of Exceptions?

There are two types of Exceptions.

1. **Checked Exceptions or Compile time Exceptions:** The Exceptions which are checked by the Compiler for smooth execution of the program at runtime are called Checked Exceptions.
   Ex:
   1. IO Exception.
   2. File Not Found Exception.
   3. SQL Exception.

   Note: In case of Checked Exceptions compiler checks whether we are handling the Exceptions or not. If the Programmer is not handling the Exception, then we get compiler time error.

2. **Unchecked Exceptions or Runtime Exceptions:** The Exceptions which are not checked by the Compiler for smooth execution of the program are called Unchecked Exceptions.
   Ex:
   1. Arithmetic Exception.
   2. Null Pointer Exception.
   3. String Index Out of Bounds Exception.
   4. Array Index Out of Bounds Exception.
   5. Number Format Exception.

   Note: In case of Unchecked Exceptions Compiler does not check whether we are handling the Exception or not.

45. Exceptions and its causes:

1. **Arithmetic Exception:** Arithmetic Exception is a one type of Unchecked Exception. Whenever we are trying to divisible anything by zero then we get Arithmetic Exception.
   Ex:
   ```
   System.out.println(10/0);
   ```

   Output:
   Arithmetic Exception.

2. **Null Pointer Exception:** Null Pointer Exception is a one type of Unchecked Exception. Whenever we are trying to perform any action on the Null Object then we get Null Pointer Exception.
   Ex:
   ```
   String s = null;
   int length = s.length();
   ```

   Output:
   Null Pointer Exception.

3. **String Index Out of Bounds Exception:** String Index Out of Bounds Exception is a one type of Unchecked Exception. Whenever we are trying to access a character of a String with an index which is either negative or positive than the length of the String, then we get String Index Out of Bounds Exception.
   Ex:
```
String s = "RBG";
char ch = s.charAt(10);
```
   Output:
   String Index Out of Bounds Exception.

4. **Array Index Out of Bounds Exception:** Array Index Out of Bounds Exception is a one type of Unchecked Exception. Whenever we are trying to access an array element without of range index , then we get Array Index Out of Bounds Exception.
   Ex:
```
int i[] = new int[2];
i[0] = 1;
i[1] = 2;
System.out.println(i[2]);
```
   Output:
   Array Index Out of Bounds Exception.

5. **Number Format Exception:** Number Format Exception is a one type of Unchecked Exception. Whenever we are trying to convert String into a number in case of String is not representing a number.
   Ex:
```
String s = "RBG";
int i = Integer.parseInt(s);
```
   Output:
   Number Format Exception.

46. What is an Error?
Most of the times Errors are not caused by our program. These are due to lack of system resources and Errors are not recoverable.

47. What is Stack Over Flow Error?
Stack Over Flow Error is a one type of Error in Java. Whenever we are trying to put too much of data on the Stack Memory which is limited resource. That means the function calls itself repeatedly with no termination condition.
Ex:
```
public class Stack
{
	public static void main(String[] args)
	{
		funA();
	}
```

```
        static void funA()
        {
              funB();
              System.out.println("Fun A");
        }

        static void funB()
        {
              funA();
              System.out.println("Fun B");
        }
}
```

Output:
Stack Over Flow Error.

48. What is an Exception Handling?
When the user wants to handle the Exception then the process is called Exception handling.

49. Why do we handle Exception?
Java Exception handling is very important. This is because, it helps to maintain normal flow of the program even when unexpected events occur.

50. How can we handle Exceptions?
Try and catch blocks are used to handle the Exceptions.
Try: Try block is used to enclose the code that might throw an Exception. Try block must be followed by either Catch or Finally blocks.

Catch: Catch block is used to handle the Exception by declaring the type of Exception within the parameter. The declared Exception must be parent Class Exception. The Catch block must be used after Try block only. We can use multiple catch blocks with a single Try block.
Ex:

```
try
{
     System.out.println(10/0);

}

catch (Exception e)
{
      // TODO: handle exception
}
```

51. What is Finally block?
Finally block is used to execute the important code such as closing the Data Base connections etc., It is always associated with Try Catch blocks. Finally block follows always executed whether an Exception is handled or not.
Ex:

```
try
```

```
{

}
catch (Exception e)
{
      // TODO: handle exception
}

finally
{

}
```

52. What is Throw?

Throw is a keyword in Java which is used to throw an Exception explicitly. We can throw either Checked or Unchecked Exceptions by throw keyword. It is mainly used to throw a Custom Exception.

Ex:

```
int age = 19;

if(age>=18)
{
      System.out.println("Person is eligible for vote");
}

else
{
      throw new ArithmeticException("Person is not eligible for
      vote");
}
```

Output:

Person is eligible for vote.

53. What is Throws?

Throws is a keyword in Java which gives an information to the programmer that there may occur an Exception. So, it is better for the programmer to provide Exception handling code. So, that the normal flow of the program can be maintained. It must be placed with the method signature.

Ex:

```
public class Throws
{
     public static void main(String[] args) throws Exception
    {
          Thread.sleep(3000);

     }
 }
```

54. What is the Hierarchy of Exceptions in Java?

Object ➔ Throwable ➔ Exception

Checked Exceptions:

1. File Not Found Exception.

2. SQL Exception.
3. IO Exception.
4. Runtime Exception.

Unchecked Exceptions:

1. Array Index Out of Bounds Exception.
2. String Index Out of Bounds Exception.
3. Number Format Exception.
4. Null Pointer Exception.
5. Arithmetic Exception.

Error:

1. Stack Over Flow Error.

55. What is This keyword in Java?

This is a keyword in Java which is used to refer current Class instance variable. The most common use of This keyword is to eliminate the confusion between the class attributes and parameters with the same name.

Ex:

```
class ThisKeyWord
{

	int x;

	void add(int x)
	{
		this.x = x;
	}

	public static void main(String[] args)
	{

	}
}
```

56. What is Super keyword in Java?

Super is a keyword in Java which is used to refer immediate Parent Class instance Variable, immediate Parent Class Method and immediate Parent Class Constructor. Super keyword should be as a first statement.

Ex:

```
public class Parent
{
	String color = "Parent class property";

	public void test()
	{
		System.out.println("Parent method");
	}
}

public class Child extends Parent
```

```
{
	String color = "Child class property";

	public void Test()
	{
		System.out.println("Child method");
	}

	Child()
	{
		System.out.println("Child class constructor");
	}

	public static void main(String[] args)
	{
		Child c = new Child();
		c.testColor();
	}

	void testColor()
	{
		System.out.println(color);
		super.test();
		System.out.println(super.color);
	}
}
```

Output:

Parent class constructor
Child class constructor
Child class property
Parent method
Parent class property

57. What is Final, Finally and Finalize?

1. **Final:** Final is a keyword in Java. It cannot be changed once created.
   a. **Final Variable:** If a Variable is defined as a final, the value cannot be changed throughout the entire program execution.

   Ex:

   ```
   final int x = 10;
   x = x+10; // compile time error
   ```

   b. **Final Method:** If a Method is defined as a final, the method cannot be overridden. Final Method should be in Parent Class.

   Ex:

   ```
   public class Parent
   {
   	final void test()
   	{
   ```

```
            System.out.println("Parent method");
        }
    }

    public class Child extends Parent
    {

        public void test // compile time error
        {
            System.out.println("Child method");
        }

        public static void main(String[] args)
        {
            Child c = new Child();
            c.Test();
        }
    }
```

c. **Final Class:** If a Class is defined as a final, the Class cannot be inherited.

Ex:

```
public final class Parent
{

}

public class Child extends Parent // compile time error
{

}
```

Here, Child Class cannot be created for Parent Class.

2. **Finally:** Finally block is used to execute the important code such as closing the Data Base connections etc., It is always associated with Try Catch blocks. Finally block follows always executed whether an Exception is handled or not.

Ex:

```
try
{

}

catch (Exception e)
{
    // TODO: handle exception
}

finally
{

}
```

3. **Finalize():** Finalize is a method which is always invoked by the Garbage Collector just before destroying an Object to perform cleanup activities.

   Note: Finally is meant for cleanup activities related to Try block. Whereas Finalize is meant for cleanup activities related to Objects.

58. What is the difference between JDK, JRE and JVM?

1. **JDK:** Java Development Kit provides the environment for developing and run the Java application.
   JDK = JRE + Development tools

2. **JRE:** Java Runtime Environment provides the environment to run the Java program. It is the implementation of JVM.
   JRE = JVM + Libraries

3. **JVM:** Java Virtual Machine does not have the physical exist. It has the responsible to execute the program line by line. JVM converts byte code that means output of the Compiler into machine level code.

59. Program execution:

1. JVM gets loaded.
2. Class gets loaded.
3. Static members get converted into M.L.
4. Main method gets started.

60. Explain about System.out.println();

Ex:

```
class System
{
      static PrintStream out;
}
```

- System is a Class present in java. lang package
- Out is a Static Variable present in System Class of type Print Stream.
- Println is a Method present in Print Stream Class.

61. What is a String?

String is a sequence of characters and it is a Class in Java but considered as a literal because of its unique behavior. Once we create a String Object, we cannot perform any changes in the existing Object. If we are trying to perform any changes, with those changes a new Object will be created. This non changeable nature is nothing but immutability of the String Object.

Ex:

```
String s = new String("RBG");
s.concat("Technologies");
System.out.println(s);
```

Output:

RBG

62. What is String Buffer?

String Buffer is a sequence of characters and it is a Class in Java. Once we create a String Buffer Object, we can perform any changes in the existing Object. This changeable nature is nothing but mutability of the String Buffer Object.

Ex:

```
StringBuffer sb = new StringBuffer("RBG");
sb.append("Technologies");
System.out.println(sb);
```

Output:

RBG Technologies

63. When to use String, String Buffer and String Builder?

String: If the content is fixed and does not change frequently, then we use String.

String Buffer: If the content is not fixed and keep on changing but thread safe is required then we use String Buffer.

String Builder: If the content is not fixed and keep on changing. And also thread safe is not required then we use String Builder.

64. What is the difference between String Buffer and String Builder?

String Buffer:

1. Every method present in String Buffer is synchronized.
2. At a time only one thread is allowed to operate on String Buffer Object. Hence String Buffer is thread safe.
3. Performance is low.

String Builder:

1. No method present in String Builder is synchronized.
2. At a time, multiple threads are allowed to operate on String Builder Object. Hence String Builder is not thread safe.
3. Performance is high.

65. What is Generic concept?

The main objective of generic is to provide type safety and to resolve type casting problem.

66. What is the Collections concept in Java?

a. Collections is a framework that provides an architecture to store and manipulate the group of Objects.

b. Collections can achieve all the operations that we perform on a data such as searching, sorting, inserting, manipulating and deleting.

c. Java collections framework provides many Interfaces like Set, List, Queue, Map and its classes are Array list, Vector, Linked list, Hash set, Linked hash set, Tree set, Hash map, Linked hash map and Tree map.

d. **Iterator:** Iterator is an Interface that iterates the elements. It is used to traverse the list and modify the elements. Iterator Interface has three methods. They are hash next(), next() and remove().

e. **Collection:** It is a root Interface with basic methods like add(), remove(), contains(), isEmpty(), addAll() etc.

f. **List:** List is an Interface which contains duplicates and elements will be in order. We can access any element from its index. List is more like an Array with dynamic length. The implementation Classes of the list Interface are Array list, Linked list and Vector.

Ex:

Array list:

Ex 1:

```
public class ListDemo
{
	public static void main(String[] args)
	{
		List al1 = new ArrayList<>();
		al1.add("RBG");
		al1.add("Technologies");
		al1.add(1);
		al1.add(2);
		System.out.println(al1);
	}
}
```

Output:

[RBG, Technologies, 1, 2]

Ex 2:

```
import java.util.ArrayList;
import java.util.List;
public class ListDemo
{
	public static void main(String[] args)
	{
		List<String> al2 = new ArrayList<String>();
		al2.add("RBG");
		al2.add("Technologies");
		// al2.add(1); compile time error
		System.out.println(al2);

		// for each loop to get all elements
		for(String s:al2)
		{
			System.out.println(s);
		}

		// for loop to get all elements
		for(int i = 0; i<al2.size(); i++)
```

```
                {
                        System.out.println(al2.get(i));
                }

                // adding the elements at specific index
                al2.add(1,"Hyderabad");

                // Removing particular element
                al2.remove(1);
        }
}
```

Output:
[RBG, Technologies]
RBG
Technologies
RBG
Technologies

g. What is the difference between Array list, Linked list and Vector?

1. **Array list:** Array list is size adjustable. If we want to use more than the size, size will be increased by 50%. Default size is 10. Basically, Array list is used to retrieve the values.
2. **Linked list:** Linked list is used to insert and delete the elements.
3. **Vector:** Vector is same as an Array list. Default size is 10. After completing the size, it will increase double the previous size means 100%.

h. What is Set?

Set is an Interface which does not allow the duplicates. The implementation classes of the Set Interface are Hash set, Tree set and Linked hash set.

i. What is the difference between Hash Map and Hash Table?

Hash map: Hash Map is the key and value pairs. Key should not be duplicate.
Ex:

```
HashMap<String, String> hm = new HashMap<>();
hm.put("Institute", "RBG Technologies");
hm.put("Name", "Venkata Rathnam");
// hm.put("Name", "Naidu"); key should not be duplicate (value is
going to be update)
System.out.println(hm);
System.out.println(hm.get("Name"));
```

Output:
{Institute = RBG Technologies, Name = Venkata Rathnam}
Venkata Rathnam

Hash Table: Hash Table is the key and value pairs. Key should not be null. If we give key as a null, at runtime hash code method performs and throws a Null Pointer Exception.
Ex:

```
Hashtable<Integer, String> ht = new Hashtable<>();
```

```
ht.put(1, "RBG");
// ht.put(null, "value"); key should not be null (compile time
error)
System.out.println(ht);
```

Output:
{1=RBG}

➢ MANUAL TESTING

1. **What is Software?**
   Software is a set of instructional programs to perform particular task.

2. **What is Manual testing?**
   Manual testing is a one type of software testing which test cases are executed manually by a tester without using any automation tool.

3. **What is Automation testing?**
   Automation testing is a one type of software testing which test cases are executed by a tester with using any automation tool.

4. **What is SDLC?**
   SDLC stands for Software Development Life Cycle. It is a process of developing a software to fulfill the client requirements within the specific cost and time.

5. **What are the phases involved in SDLC?**
   1. **Requirement collection:**
      In Requirement collection phase Business Analyst collects the requirement with the interaction of client. The collected information will be documented as Business Requirement Specification (BRS) or User Requirement Specification (URS) or Customer Requirement Specification (CRS).

   2. **Requirement Analysis:**
      In Requirement Analysis Phase System Analyst studies, the BRS document and prepares a detailed functionality document based on the BRS. Then the document is called Functional Requirement Specification (FRS).

   3. **Designing:**
      In Designing phase Design Architect plans the programming languages, Scripting languages and Database technologies based on the clients' requirements. Then the document is called Graphical User Interface (GUI) or Database (DB) document or Application design document.

   4. **Coding:**
      In Coding phase Application is developed by the developers based on the Design document.

   5. **Testing:**
      In Testing phase, After completing the coding, the application is given to the separate testing team to validate the software.

a. **White box testing:**
Initial stage of testing is performed by the developers like unit testing and integration testing. Those techniques come under the white box testing. Actually, white box testing is a source code level of testing by the developers.

b. **Black box testing:**
After unit and integration testing, the application is given to the separate testing team to validate the software.

c. **User Acceptance Testing (UAT):**
After completing the Black Box Testing (BBT), we conduct User Acceptance Testing (UAT) to know the client satisfaction.

6. **Release and Maintenance:**
After system testing the application is delivered to the client. It is also called as GO-LIVE and the changes can be made further upon the request of client is called maintenance.

6. **What is SDLC model?**
Depending on complexity of functionality and need of client we can use any one of the following SDLC model to develop an application.
Here, we have been using Agile methodology.

7. **What is verification?**
Verification is a process to check the correctness and completeness of the document which is prepared to develop the software. Verification is also called as a Static testing.

8. **What is Validating:**
After unit and integration testing, the application is given to the separate testing team to test the software whether it is working as expected or not. It is also called as a Dynamic testing.

9. **What are the roles and responsibilities of a tester?**
   a. **Manual tester:**
      - Understanding the requirements.
      - Writing the test cases.
      - Executing test cases.
      - Finding the defects and reporting them to the developers.
      - Participate different reviews and meetings.
      -

   b. **Automation tester:**
      - Designing, developing and executing automation scripts to test software application.
      - Collaborating with development teams to understand software requirements and identifying areas for automation.
      - Identifying and documenting software bugs and defects.
      - Maintaining and updating existing automation scripts and automation framework.
      - Creating and executing regression test suites to ensure software stability.
      - Participating in code reviews.

- Guiding and mentoring junior testers by providing support and sharing best practices.

10. **What is Smoke testing?**
    Smoke testing is used to check the critical functionalities are working fine or not. It is also called as build verification testing or build acceptance testing. The main purpose of the smoke testing is to check whether build is properly downloading and installing or not.

11. **What is Sanity testing?**
    Sanity testing is used to check the new functionalities or bugs have been fixed. It is a subset of regression testing.

12. **What is Localization testing?**
    Localization testing is used to check the localized version of the application is working fine or not.

13. **What is End to End testing?**
    End to End testing is used to check overall functionalities among the all modules are working fine or not.

14. **What are the phases involved in Software Testing Life Cycle (STLC)?**
    1. Requirement Analysis.
    2. Test planning.
    3. Test case developing.
    4. Environment setup.
    5. Text execution.
    6. Test cycle closure.

15. **What is Retesting?**
    Retesting is used to check whether the bug is fixed or not.

16. **What is Regression testing?**
    Regression testing is used to check the impact of modification on existing and working components.

17. **What is Monkey testing?**
    Monkey testing is used to check the application by providing random inputs. Its intension is to break the application to find corner defects. This testing is mainly used in Gaming applications.

18. **What is Exploratory testing?**
    Exploratory testing is used to explore the entire application when it is newer to the tester.

19. **What is Exhaustive testing?**
    Exhaustive testing is used to check the application by all possible ways.

20. **What is Optimal testing?**
    Optimal testing is used to check the application by best possible ways.

21. **What is Test case?**
    A document with expected result is called Test case.

22. **What is Test suite?**
    A collection of test cases is called Test suite.

23. **What is Test scenario?**
    Test scenario is a detailed document of the test cases which says what to be tested.

24. **What should we do after finding a bug?**
    Once we find a bug, we must try to recreate at least 3 to 4 times. Then we can be sure that it is bug. Once we confirm that it is a bug and then it is a good idea to attach supporting documents when we log. The next step is we need to log it.

25. **Explain about Bug Life Cycle?**
    - When the bug is logged, it gets **NEW** status.
    - When the bug is accepted, it gets **OPEN** status.
    - When the bug is assigned to the developers, it gets **ASSIGNED** status.
    - When the bug is being fixed, it gets **IN PROGRESS** status.
    - When the bug is fixed, it gets **FIXED** status.
    - When the bug is tested, it gets **VERIFIED** status.
    - If a decision is made to fix the bug in later release, it gets **DEFFERED** status.

➢ AGILE METHODOLOGY

1. **What is Agile?**
   Agile is a systematic approach to develop and test a software.

2. **What are the advantages of an Agile?**
   1. **Incremental:**
      In this incremental process, a software is developed as a bits and pieces.

   2. **Iterative approach:**
      In this iterative approach process, a software is developed upon already developed pieces. These are the possibilities in the Agile process.
      Note: The beauty of Agile process is, the changes can be accepted at any time in the development process.

3. **What are the disadvantages of Agile?**
   - Less documentation.
   - Team must be knowledgeable.

4. **What are the principles or manifest of Agile?**
   - Continues delivery for 1 month, 3 months or 6 months to the customers.
   - Changes can be accepted at any time in the development process.
   - Developing a software frequently for couple of weeks and months.
   - Everyone will be together that means business people, developers and testers throughout the project. Then it is accuracy because we will not miss the data.

- Face to face conversation like daily standup meeting. Will be there.
- Technical challenges will be there. So, we have to face them.

5. What are the different flavors of Agile?

1. Extreme programming (XP).
2. Feature-driven development.
3. Kanban.
4. Scrum.

6. What is Scrum?

Scrum is a kind of framework through which we will develop and test the software by following Agile process.

7. What are the Scrum terminologies?

1. Epic:

Epic is a larger requirement what client wants with less functionalities.

2. User stories:

Epic is a larger requirement. It can be divided into bits and pieces. Those bits and pieces are called User stories.

User stories are simple and short from the user perspective that means what is application and it has a simple template with simple English format.

8. Scrum team:

1. Product owner.
2. Scrum master.
3. Developers.
4. Testers.

9. What are the Roles & Responsibilities of a Product owner?

1. Product owner gathers requirement from the client.
2. This requirement is called Product backlogs.
3. Product owner designs Epics and User stories.

10. What are the Roles & Responsibilities of a Scrum master?

1. Scrum master does not know about business and flow of application.
2. They look at the process of development.
3. Scrum master makes managing the coordination between teams.

11. What are the Artefacts?

1. Product backlog:

The collection of requirements which is prepared by the Product owner.

2. Sprint backlog:

List of committed stories by dev and QA teams for specific sprint.

12. What is Sprint or Iteration?

Sprint is a period of time to complete the User stories which is decided by the Product owner and team. Usually, 2 to 4 weeks of time.

13. What are the Activities or Ceremonies in Agile?

1. Sprint Planning Meeting:

a. Sprint planning meeting is conducted with the team to define what can be delivered in the duration.
b. Team will decide that what are the stories we have to develop, test and deliver to the customer.

2. Scrum Meeting or Scrum Call or Standup Call:

a. Standup call is conducted by Scrum master every day 15 min of time.
b. If there are any blockers, if there are any requirements missing, if developer is unable to develop something, if tester is unable to test something then Scrum master will take care of it.
c. Here, we exactly discuss about what we did yesterday and what we will do today.

3. Sprint Retrospective Meeting:

a. Sprint retrospective meeting is conducted only once after completing every sprint.
b. Here, we exactly discuss about what went well? And what went wrong?
c. What are the improvements we have to do in the next Sprint.

14. What is Story Point?

1. Rough estimation of user stories is given by the dev & QA teams in the form of Fibonacci series.
2. Developer will say, we need this much of time to develop the story.
3. Tester will say, we need this much of time to write and execute the test cases.

➢ SELENIUM

1. What is Selenium?

Selenium is a one type of tool which is used to automate the web applications. It is an open-source tool.

2. Any application we should consider these points:

a. Business.
b. Entry point.
c. Functional flow.
d. Testing scope.

3.

Web component	Tag name	Mandatory Attributes	Optional Attributes
Link	a	href	id, class, name
Text fields/input box	input	type = “email”	any
Buttons	button/input	type = “submit”/type = “button”	any

Radio buttons	input	type = “radio”	any
Check box	input	type = “check box”	any
Text/label	p, div, span		any
Image	img	src	any
Drop down	select		any
Web table	table		any
I frames	iframe		any

4. **What is the Locator and what are they?**

Locator: Locator is a way to find web elements in a web page. There are 8 types of locators. They are

1. Id.
2. Name.
3. Class name
4. X path.
5. CSS selector.
6. Tag name.
7. Link text.
8. Partial text.

5. **CSS selector:**

1. Single Attribute:

Syn: Tag name [attribute name = ‘attribute value’]

2. Multiple Attribute:

Syn: Tag name [attribute name = ‘attribute value’] [attribute name = ‘attribute value’]

6. **What is X path and types of X paths?**

X path is a one type of locator which is used to identify web elements. And it is a query language for selecting nodes from an XTML document.

Types of X path:

1. **Absolute X path:** Absolute X path starts from the root element of the HTML document.
   Ex: /html/body/div [1]/div [1]/input

2. **Relative X path:** Relative X path starts from the current element of the HTML document based on the search criteria.
   Ex: // Tag name [@ attribute name = ‘attribute value’]

3. **X path operators:**
   1. = Equal

2. ! = Not Equal
3. < Less than
4. <= Less than or equal
5. > Greater than
6. >= Greater than equal

4. **X path Index:**
   (// Tag name [@ attribute name = 'attribute value']) [1]

5. **X path Functions:**
   1. **Text:**
      Ex:
      // Tag name [text () ='text']

      Note: shortcut for text () is .

   2. **Contains:**
      Ex:
      // Tag name [contains (text (), 'text')]
      // Tag name [contains (@attribute name, 'attribute value')]

   3. **Starts-with:**
      Ex:
      // Tag name [starts-with (text (), 'text')]
      // Tag name [starts-with (@attribute name, 'attribute value')]

   4. **Normalize-space:**
      Ex:
      // Tag name [normalize-space (text (), 'text')]
      // Tag name [normalize-space (@attribute name, 'attribute value')]

   5. **Last ():**
      Ex:
      (// Tag name [@ attribute name = 'attribute value']) [last ()]

   6. **Position:**
      Ex:
      (// Tag name [@ attribute name = 'attribute value']) [position () =1]

6. **X path Axes:**
   1. **Following sibling:**
      Ex:
      // Tag name [@ attribute name = 'attribute value']/following-sibling::tag name[@attribute name = 'attribute value']

   2. **Following:**
      Ex:
      // Tag name [@ attribute name = 'attribute value']/following::tag name

3. **Preceding sibling:**
   Ex:
   // Tag name [@ attribute name = 'attribute value']/preceding-sibling::tag name[@attribute name = 'attribute value']

4. **Preceding:**
   Ex:
   // Tag name [@ attribute name = 'attribute value']/preceding::tag name

5. **Child:**
   Ex:
   // Tag name [@ attribute name = 'attribute value']/child::tag name

   Note: shortcut for child is //Tag name [@attribute name = ' attribute value']/tag name

6. **Parent:**
   Ex:
   // Tag name [@ attribute name = 'attribute value']/parent::tag name

   Note: shortcut for child is //Tag name [@attribute name = ' attribute value']/.

7. **Ancestor:**
   Ex:
   // Tag name [@ attribute name = 'attribute value']/ancestor::
   // Tag name [@ attribute name = 'attribute value']/ancestor::*
   // Tag name [@ attribute name = 'attribute value']/ancestor-or-self::

8. **Descendant:**
   Ex:
   // Tag name [@ attribute name = 'attribute value']/descendant::
   // Tag name [@ attribute name = 'attribute value']/descendant::*

7. **X path Conditions:**
   1. **And:**
      Ex:
      // Tag name [@ attribute name = 'attribute value' and @ attribute name = 'attribute value']

   2. **Or:**
      Ex:
      // Tag name [@ attribute name = 'attribute value' or attribute name = 'attribute value']

   3. **Not:**
      Ex:

// Tag name [not (@ attribute name = 'attribute value')]

7. **What are the components available in Selenium?**
   Components available in Selenium are
   1. Selenium Web driver.
   2. Selenium IDE.
   3. Selenium Grid.

8. **What is Selenium Web Driver?**
   Selenium Web Driver provides a programming interface to interact with web pages.

9. **What are the advantages and disadvantages of Selenium Web Driver?**
   Advantages:
   1. Selenium Web Driver supports multiple programming languages like Java, Python, Java Script, .net etc.
   2. It supports multiple operating systems like Windows, Mac, Linux etc.
   3. And similarly, it supports multiple web browsers like Chrome, Safari, Fire fox, Opera mini, Edge etc.
   4. Moreover, it is a free of cost.

   Disadvantages:
   Selenium automates only browser. Apart from the browser it cannot take care of any other things. But in automation we need to take care of other lot of things also.
   1. Sometimes we need to perform data driven testing, like where we have to specify the data in the excel files.
   2. Sometimes we need to generate the reports also which is very important. These things are not possible by using Selenium Web Driver.
   3. It cannot automate the window-based applications. Suppose when we see something called file upload, file downloads scenarios where we should interact with windows. So, Selenium Web Driver does not support those scenarios.
   4. But still we can overcome those challenges by integrating third party tools like Apache POI for handling data driven testing, Auto IT for handling the window-based applications and Extent reports for generating reports to maintain logs.

10. **What is the Architecture of Selenium Web Driver?**
    Search context ➔ extends ➔ Web Driver, Java Script executor and Takes screen shot ➔ implements ➔ Remote Web Driver ➔ extends ➔ Chrome Driver, Fire fox, Edge Driver, Safari Driver etc.

11. **How to launch browsers in Selenium Web Driver?**
    To launch any browsers, we have two different ways.
    1. First one is we need to specify the browser driver location into the statement called System. Set Property (key, value). Here, key should be in string such as "webdriver. chrome.driver" similarly value should be also in string such as "path of the driver file".
       Ex:

```
System.setProperty("webdriver.chrome.driver","driver file path");
WebDriver driver = new ChromeDriver();
```

2. Second way is, we have something called WebDriver Manager API. Here, we can directly specify the methods which will launch browsers. So, we don't need to download drivers manually. Web Driver Manager API particularly take care of all the drivers. So, for that what we have to do is, we need to get the Web Driver Manager.
   So, we need to add another dependency called Web Driver Manager in pom.xml of Maven project. Once we have this dependency it will automatically download required jar files.
   Ex:

```
WebDriverManager.chromeDriver().setup();
WebDriver driver = new ChromeDriver();
```

12. How to open URL?

If we want to open any URL in the browsers, there is a method called driver.get("URL");

Ex:

```
driver.get("www.RBG Technologies");
```

- Driver.get method is zero parameterized.
- No return type.

13. How to capture title of the page?

Sometimes we need to capture the tittle of the web page to perform certain validation.

Ex:

```
String title = driver.getTitle();
```

- getTitle method is zero parameterized.
- Return type of the getTittle is String.

14. How to capture current URL?

If we want to capture the current URL of the application to validate whether it is properly opening the current URL or not.

Ex:

```
String currentURL = driver.getCurrentUrl();
```

- getCurrentUrl method is zero parameterized.
- Return type is String.

15. How to capture page source of the page?

Suppose we want to validate some content in the particular page, we have to capture the entire source code.

Ex:

```
String pagSource = driver.getPageSource();
```

- getPageSource method is zero parameterized.
- Return type is String.

16. How to get the text from any web element?

To get the text from any web element there is a method called getText().

Ex:

```
WebElement element = driver.findElement(By.id("email"));
String text = element.getText();
```

- getText method is zero parameterized.
- Return type is String.

17. **How to capture any Attribute value from the web element?**
    To capture any attribute value, from the web element there is a method called getAttribute("Attribute name").
    Ex:
```
WebElement element = driver.findElement(By.id("email"));
String attributevalue = element.getAttribute("Attribute name");
```
    - getAttribute method is String parameterized.
    - Return type is String.

18. **What is the difference between getText() and getAttribute()?**
    getText():
    getText() is a method which returns the inner text of the web element.

    getAttribute:
    getAttribute is a method which returns the value of any attribute.

19. **What are the conditional methods or conditional commands?**
    Basically, we need to check the status of the web element on the web page whether it is displayed, enabled and selected or not. We have three methods to check the status of the web element. They are.
    1. isSelected();
    2. isEnabled();
    3. isDislayed();

20. **How to check the web element is selected or not?**
    If we want to check whether element is selected or not to perform some validation there is a method called isSelected();.
    Ex:
```
WebElement element = driver.findElement(By.xpath("xpath value"));
boolean actual = element.isSelected();
```
    - isSelected method is zero parameterized.
    - Return type is boolean.

21. **How to check the web element is enabled or not?**
    If we want to check whether element is enabled or not to perform some validation there is a method called isEnabled();.
    Ex:
```
WebElement element = driver.findElement(By.xpath("xpath value"));
boolean actual = element.isEnabled();
```
    - isEnabled method is zero parameterized.
    - Return type is boolean.

22. **How to check the web element is displayed or not?**
    If we want to check whether element is displayed or not to perform some validation there is a method called isDislayed();.
    Ex:
```
WebElement element = driver.findElement(By.xpath("xpath value"));
```

```
boolean actual = element.isDisplayed();
```

- isDisplayed method is zero parameterized.
- Return type is boolean.

23. What are the Navigation Commands?

```
1. driver.navigate().back();
2. driver.navigate().forward();
3. driver.navigate().refresh();
4. driver.navigate().to("URL");
```

24. How to navigate to back from the web page?

If we want to come back to the previous page, it can be achieved by using navigate().back();.

Ex:

```
WebDriverManager.chromedriver().setup();
WebDriver driver = new ChromeDriver();
driver.get("https://www.Gituhub.com");
driver.get("https://www.RBGTechonologies.com");
driver.navigate().back();
```

25. How to navigate to forward?

If we want to forward to the next page, if it is opened, it can be achieved by using navigate().forward();.

Ex:

```
WebDriverManager.chromedriver().setup();
WebDriver driver = new ChromeDriver();
driver.get("https://www.Gituhub.com");
driver.get("https://www.RBGTechonologies.com");
driver.navigate().forward();
```

26. How to refresh the web page?

To refresh or reload a web page there is a method called navigate().refresh();.

Ex:

```
WebDriverManager.chromedriver().setup();
WebDriver driver = new ChromeDriver();
driver.get("https://www.RBGTechonologies.com");
driver.navigate().refresh();
```

27. What is the difference between driver.get() and driver.navigate().to()?

1. The driver.get() and driver.navigate().to() are the methods which helps us to navigate into the web sites. Basically, there is no difference between these two methods.
2. driver.get() method accepts String as a parameter and driver.navigate().to() method accepts String and URL instance as a parameter. Internally navigate().to() method invokes get() method.

Ex:

```
import java.net.URL;
public class NavigationCommands
{
      public void to(String url)
      {
            get(url);
      }
```

```
        public void to(URL url)
        {
              get(String.valueof(url));
        }
}
```

28. What is the difference between Find element and Find elements?

Find Element:

a. If Single web element is present, it returns single element.
b. If multiple web elements are present, it returns first element.
c. If element is not present, it throws no such element exception.
d. Find element return type is web element.
e. Element can be accessed directly.

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));
```

Find Elements:

a. If Single web element is present, it returns single element.
b. If multiple web elements are present, it returns all elements.
c. If element is not present, it does not throw any exception but returns zero elements.
d. Find elements return type is List<web element>.
e. Elements can be accessed through iterate.

Ex:

```
List<WebElement> element = driver.findElements(By.xpath("xpath value"));
```

29. How to enter the text?

To enter or pass the text to the input boxes there is a method called sendKeys();.

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));
element.sendKeys("RBG Technologies")
```

- sendkeys() method is String parameterized.
- No return type.

30. How to clear the data?

To clear the data from the input boxes there is a method called clear();

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));
element.sendKeys("RBG Technologies");
element.clear();
```

- clear() method is zero parameterized.
- No return type.

31. How to handle Drop down in a Selenium Web driver?

If the tag name is select, then it can be considered as a Drop down. Drop down is not a single web element because drop down itself is a one web element and inside this whatever options are available each option is a separate web element. We can select these options by using Select class there we can see three methods. They are

```
1. selectByVisibleText("RBG");
2. selectByValue("2");
3. selectByIndex(1);
```

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));
Select dropdown = new Select(element);
dropdown.selectByVisibleText("RBG");
dropdown.selectByValue("2");
dropdown.selectByIndex(1);
```

- selectByVisibleText() method is a String parameterized and no return type.
- selectByValue() method is a String parameterized (gets the value from the attribute) and no return type.
- selectByIndex() method is a int parameterized and no return type.

Sometimes we need to handle dropdown without using these three methods. Then we have a method called getOptions(); which returns all options.

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));

Select s = new Select(element);

List<WebElement> allOptions = s.getOptions();

for(WebElement option : allOptions)

{

      if(option.getText().equals("RBG")

      {

            option.click();

            break;

      }

}
```

32. How to handle Boot Strap Drop down in Selenium Web Driver?

If the element which looks like a Drop down but it does not have Select tag name then it is called Boot strap Drop down.

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));
element.click();
List<WebElement> allOptions = driver.findElements(By.id("list"));
for(WebElement option : allOptions)
```

```
{
      if(option.getText().equals("RBG"))
      {
            option.click();
            break;
      }
}
```

33. How to select specific check box?

To select specific check box there is a method called click().

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));
element.click();
```

34. How to select all check boxes?

To select all check boxes first we need to identify all elements then we go for loop statement.

Ex:

```
List<WebElement> allCheckBoxes = driver.findElements(By.xpath("xpath
value"));
int allCheckBoxesSize = allCheckBoxes.size();
for(int i = 0; i <= allCheckBoxesSize; i++)
{
      allCheckBoxes.get(i).click();
}

// for each loop
for(WebElement checkbox : allCheckBoxes)
{
      checkbox.click();
}
```

35. How to select multiple check boxes based on some text?

Ex:

```
List<WebElement> allCheckBoxes = driver.findElements(By.xpath("xpath
value"));
// for each loop
for(WebElement checkBox : allCheckBoxes)
{
      String checkBoxName  = checkBox.getAttribute("id");
      if(checkBoxName .equals("Monday")||
      checkBoxName.equals("Sunday"))
      {
            checkBox.click();
      }
}
```

36. How to select last two check boxes?

Ex:

```
List<WebElement> allCheckBoxes = driver.findElements(By.xpath("xpath
value"));
```

```
int allCheckBoxesSize = allCheckBoxes.size();
for(int i = allCheckBoxesSize-2; i<allCheckBoxesSize; i++)
{
     allCheckBoxes .get(i).click();
}
```

37. **How to select first two check boxes?**

Ex:

```
List<WebElement> allCheckBoxes = driver.findElements(By.xpath("xpath value"));
int allCheckBoxesSize = allCheckBoxes.size();
for(int i = 0; i<allCheckBoxesSize; i++)
{
     if(i<2)
     {
          allCheckBoxes .get(i).click();
     }
}
```

38. **How to handle browser windows in Selenium?**

Sometimes as a part of development new windows are opened. By default, the driver is focusing on first window. We cannot perform any actions in those windows unless we switch. In order to switch between multiple windows first we need to capture the windows ids by using getWindowHandle() and getWindowHandles().

Ex:

```
String parentWindow = driver.getWindowHandle();
Set<String> allWindows = driver.getWindowHandles();

// first method to retrieve
Iterator<String> it = allWindows.iterator();
String parentWindow2 = it.next();
String childWindow = it.next();

//Second method to retrieve
List<String> allWindowsIds = new ArrayList<>(allWindows);
String parentWindowId = allWindowsIds.get(0);
String childWindowId = allWindowsIds.get(1);
```

39. **How to switch from one window to another window?**

To switch from one window to another window there is a method called switchTo().window();.

Ex:

```
driver.switchTo().window(childWindowId);
driver.switchTo().window(parentWindowId);

//for loop
for(String windowId : allWindows)
{
     String title = driver.switchTo().window(windowId).getTitle();
     System.out.println(title);
}
```

40. **How to close specific browser window?**

Ex:

```
for(String windowId : allWindowsIds)
{
      String title = driver.switchTo().window(windowId).getTitle();
      if(title.equals("RBG"))
      {
            driver.close();
      }
}
```

41. **What is the difference between driver.quit() and driver.close()?**

quit():

quit() is a method which closes all browser windows and it terminates the driver session as well.

Ex:

```
driver.quit();
```

close():

close() is a method which closes only current browser window and it continues the driver session.

Ex:

```
driver.close();
```

42. **How to locate links in Selenium Web Driver?**

Normally when we see any web page, we can see a tag name called a. So, a represents anchor tag. Then it is considered as a link. Basically, linkText and partialLinkText locators are used to identify links.

Ex:

This is the link

43. **What is the difference between linkText and partialLinkText?**

linkText and partialLinkText are the locators which are used to identify elements in the web page.

linkText:

when we use linkText we have to pass the full text of the link which means inner text of the link.

Ex:

```
driver.findElement(By.linkText("This is the link"));
```

partialLinkText:

when we use partialLinkText we have to pass some portion of the inner text.

Ex:

```
driver.findElement(By.partialLinkText("This is"));
```

44. **How to capture all links from the web page?**

To capture all links present in the web page we need to write a common locator. Here, we have to use the method called findElements() and by using locator called tag name. All links have the common tag name that is a (anchor tag).

Ex:

```
List<WebElement> allLinks = driver.findElements(By.tagName("a"));
int allLinksSize = allLinks.size();
```

```
//for loop
for(int i = 0; i<=allLinksSize; i++)
{
      String linkText = allLinks.get(i).getText();
      System.out.println(linkText);
}
```

45. **What is broken links and how to find broken links?**

Broken links:

Broken links are something which does not have any target page. So, when we click on the link which will not take us to any target page. So, this type of links is called Broken links.

Ex:

```
List<WebElement> allLinks = driver.findElements(By.tagName("a"));
for(WebElement link : allLinks)
{
      String linkAttributeValue = link.getAttribute("href");
      // converting into URL
      URL url = new URL(linkAttributeValue);
      // opening the connection
      URLConnection urlConnection = url.openConnection();
      HttpURLConnection httpURLConnection =
      (HttpURLConnection)urlConnection;
      httpURLConnection.setConnectTimeout(5000);
      // Sending request to the server
      httpURLConnection.connect();
      int responseCode = httpURLConnection.getResponseCode();
      if(responseCode >= 400)
      {
            System.out.println(linkAttributeValue + "is a broken
link");
      }
      else
      {
            System.out.println(linkAttributeValue+ "is not a broken
link");
      }
}
```

46. **How to handle JavaScript Alerts?**

Alert:

Alert is a message or notification box that notifies the user about some information or ask for permission to perform a certain kind of operation.

a. JS Alert:

JS alert is a one type of alert which contains only button.

Ex:

```
Alert alert = driver.switchTo().alert();
String alertText = alert.getText();
System.out.println(alertText);
alert.accept();
```

b. JS Confirm Alert:

JS confirm alert is a one type of Alert which contains two button OK and CANCEL.

Ex:

```
Alert alert = driver.switchTo().alert();
String alertText = alert.getText();
System.out.println(alertText);
alert.accept();
// or
alert.dismiss();
```

c. JS Prompt Alert:

JS Prompt alert is a one type of alert which contains two buttons and input box.

Ex:

```
Alert alert = driver.switchTo().alert();
String alertText = alert.getText();
System.out.println(alertText);
alert.sendKeys("P.V.R.Naidu");
alert.accept();
// or
alert.dismiss();
```

d. Authenticated popups:

When authenticated popups come, we cannot perform any operations unless we provide the credentials or some information into the alerts.

Ex:

```
driver.get("https://username:password@RBG technologies.com");
```

This statement will login and accept the authenticated popups.

e. Permission popups:

When Permission popups come, we cannot perform any operations unless we press confirm or cancel.

Ex:

```
ChromeOptions options = new ChromeOptions();
options.addArguments("—disable-notifications");

WebDriverManager.chromedriver().setup();
WebDriver driver = new ChromeDriver(options);
driver.get("RBG Technologies");
```

47. What is the difference between frame and iframe?

Frame:

Frame is a HTML tag which is used to divide a web page into various parts.

Ex:

<frame>

Iframe:

Iframe is used to embed some other document within the current HTML document.

Ex:
<iframe>

48. How to handle iframes in Selenium?

We have the switchTo().frame() method to handle iframes.

Ex:

```
WebElement frameElement = driver.findElement(By.xpath("xpath
value"));

// value of the name attribute
driver.switchTo().frame("frame name");

// index of the frame
driver.switchTo().frame(0);

//Passing the web element
driver.switchTo().frame(frame element);

// Go back to the main page
driver.switchTo().defaultContent();
```

49. What is Synchronization problem in Automation?

Code execution and application needs to be in sync to perform any operations. If the application gets slow down for any reasons like network, heavy load etc. Then the code keeps on checking for particular web element.

50. What are the wait statements available in Selenium Web driver?

To solve synchronization problem, we have multiple wait statements available in Selenium. They are

1. **Implicit wait:**

   Implicit wait tells the web driver to wait for a certain measure of time before throwing an exception. Once this time is set, web driver waits for every element before the exception occurs. It is a global wait.

   Ex:

   ```
   driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10))
   ;
   ```

2. **Explicit wait:**

   Explicit wait tells the web driver to wait until a certain condition occurs.

   Ex:

   ```
   WebDriverWait wait = new
   WebDriverWait(driver,Duration.ofSeconds(10));
   wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath
   ("xpath value")));
   ```

3. **Fluent wait:**

   It is also a similar to web driver wait or explicit wait with more flexibility in polling time and ignoring exceptions.

   Polling time:

Suppose we say 30 sec as a maximum timeout in fluent wait and 5 sec as a polling time, what is the polling time here is, our wait will go and check for element for every 5 seconds whether element is present or not.

Ex:

```
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
        .withTimeout(Duration.ofSeconds(30))
        .pollingEvery(Duration.ofSeconds(5))
        .ignoring(NoSuchElementException);

// usage of fluent wait
WebElement element = wait.until(new
ExpectedCondition<WebElement>()
{
     public WebElement apply(WebDriver driver)
     {
          return driver.findElement(By.xpath("xpath value"));
     }
});
element.click();
```

51. How to handle Web table in Selenium WebDriver?

a. To get the rows in a Table:

Ex:

```
List<WebElement> rows =
driver.findElements(By.xpath("//table[@id =
'customers']/tbody/tr"));
int rowsCount = rows.size();
System.out.println(rowsCount);
```

b. To get the columns in a Table:

Ex:

```
List<WebElement> columns =
driver.findElements(By.xpath("//table[@id='customers']/tbody/t
r/th"));
int columnsCount = columns.size();
System.out.println(columnsCount);
```

c. Retrieve the specific row or column data:

Ex:

```
WebElement specificData =
driver.findElement(By.xpath("//table[@id='customers']/tbody/tr
[2]/td[1]"));
String text = specificData.getText();
System.out.println("specific row/column data : "+text);
```

d. Retrieve all data from the Table:

Ex:

```
// rows
```

```
List<WebElement> rows =
driver.findElements(By.xpath("//table[@id =
'customers']/tbody/tr"));
int rowsCount = rows.size();
System.out.println(rowsCount);

// columns
List<WebElement> columns =
driver.findElements(By.xpath("//table[@id='customers']/tbody/t
r/th"));
int columnsCount = columns.size();
System.out.println(columnsCount);

// all data
for(int r = 1; r<=rowsCount; r++)
{
     for(int c =1; c<=columnsCount; c++)
     {
          String data =
driver.findElement(By.xpath("//table[@id =
'customers']//tr["+r+"]/td["+c+"]")).getText();
          System.out.println("all data : "+ data);
     }
}
```

52. How to perform mouse operations in Selenium?

To perform mouse operations, we have a special class called Actions class. So, we can perform the actions like right click, double click, drag and drop and mouse over by using Actions class.

a. Right click:

Ex:

```
WebElement rightClickElement = driver.findElement(By.xpath("xpath
value"));
Actions actions = new Actions(driver);
actions.contextClick(rightClickElement).perform();
```

b. Double click:

Ex:

```
WebElement doubleClickElement =driver.findElement(By.xpath("xpath
value"));
Actions actions = new Actions(driver);
actions.doubleClick(doubleClickElement).perform();
```

c. Drag and drop:

Ex:

```
WebElement drag = driver.findElement(By.xpath("xpath value"));
WebElement drop = driver.findElement(By.xpath("xpath value"));
Actions actions = new Actions(driver);
actions.dragAndDrop(drag,drop).perform();
```

d. Mouse over:

Ex:

```
WebElement mouseOver = driver.findElement(By.xpath("xpath value"));
Actions actions = new Actions(driver);
actions.moveToElement(mouseOver).perform();
```

53. What is the difference between methods of Actions class, perform() and build.perform()?

Both perform() and build.perform() methods are the same. Internally perform() method invokes build.perform() method.

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));
Actions actions = new Actions(driver);
actions.moveToElement(element).build().perform();

//or
Action action = actions.moveToElement(element).build();
action.perform();

// Here, Action is an Interface and Actions is a Class.
```

54. How to perform key board actions in Selenium?

a. Enter:

Ex:

```
Actions actions = new Actions(driver);
actions.sendKeys(Keys.ENTER).perform();
```

b. Back space:

Ex:

```
Actions actions = new Actions(driver);
actions.sendKeys(Keys.BACK_SPACE).perform();
```

c. Escape:

Ex:

```
Actions actions = new Actions(driver);
actions.sendKeys(Keys.ESCAPE).perform();
```

d. Space:

Ex:

```
Actions actions = new Actions(driver);
actions.sendKeys(Keys.SPACE).perform();
```

e. Tab:

Ex:

```
Actions actions = new Actions(driver);
actions.sendKeys(Keys.TAB).perform();
```

f. Ctrl+A:

Ex:

```
Actions actions = new Actions(driver);
actions.keyDown(Keys.CONTROL);
```

```
actions.sendKeys("a");
actions.keyUp(Keys.CONTROL);
actions.perform();
```

g. **Ctrl+C:**

Ex:

```
Actions actions = new Actions(driver);
actions.keyDown(Keys.CONTROL);
actions.sendKeys("c");
actions.keyUp(Keys.CONTROL);
actions.perform();
```

h. **Ctrl+V:**

Ex:

```
Actions actions = new Actions(driver);
actions.keyDown(Keys.CONTROL);
actions.sendKeys("v");
actions.keyUp(Keys.CONTROL);
actions.perform();
```

55. **How to capture a screen shot of a full page?**

Sometimes we need to capture a screen shot to maintain a logs.

Ex:

```
TakesScreenshot ts = (TakesScreenshot)driver;
File sourceFile = ts.getScreenshotAs(OutputType.FILE);
File destinationFile = new File("file path\\screenshot.png");
FileUtils.copyFile(sourceFile, destinationFile);
```

56. **How to capture a screen shot of a specific section?**

To capture a specific section, we need to identify specific part of the web page.

Ex:

```
WebElement specificSection = driver.findElement(By.xpath("xpath
value"));
File src = specificSection.getScreenshotAs(OutputType.FILE);
File trg = new File("file path\\ specific part.png");
FileUtils.copyFile(src, trg);
```

57. **How to capture a screen shot of specific element?**

To capture a specific element, we need to identify the web element.

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));
File src = element.getScreenshotAs(OutputType.FILE);
File trg = new File("file path\\ specific element.png");
FileUtils.copyFile(src, trg);
```

58. **What is JavaScript executor and what are the methods that we use?**

JavaScript executor is an Interface that helps to execute the JavaScript through Selenium WebDriver. It provides executeScript() and executeAsyncScript() methods.

a. **Click:**

Ex:

```
WebElement element = driver.findElement(By.xpath("xpath value"));
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("arguments[0].click();",element);
```

b. Enter the data:

Ex:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
js.executeScript("document.getElementById('email').value =
'RBG';");
```

c. Scroll down:

Ex:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
js.executeScript("window.scrollBy(0,500);");

// or

js.executeScript("window.scrollBy(0,document.body.scrollHeight)")
;
```

d. Scroll up:

Ex:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
js.executeScript("window.scrollBy(0,-500);");

// or

js.executeScript("window.scrollBy(0,(document.body.scrollHeight))
");
```

e. High light:

Ex:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
js.executeScript("document.getElementById('email').style.border =
'5 px red solid';");
```

59. Exceptions and its causes:

a. No Such Element Exception:

No Such Element Exception is a one of the most common Exception in Selenium. It is thrown when a HTML element cannot be found due to either synchronization issue or element is in the frame or wrong value of the XPath.

b. No Such Window Exception:

Whenever WebDriver is trying to switch to an invalid window which is unavailable. Then we get No Such Window Exception.

c. No Such Frame Exception:

Whenever WebDriver is trying to switch to an invalid frame which is unavailable. Then we get No Such Frame Exception.

d. No Alert Present Exception:

Whenever WebDriver is trying to switch to an invalid Alert which is unavailable. Then we get No Alert Present Exception.

e. Time Out Exception:

Time out Exception is thrown, when a HTML element cannot be found even though we use Explicit Wait to wait for an element for some time.

f. Stale Element Reference Exception:

Stale Element Reference Exception is thrown, if the page gets changed or reloaded then DOM is rebuilt. So, previously founded elements become stale.

60. How to get the single Cell value from the Excel Sheet?

Ex:

```
public class ExcelReadAndWrite
{
      public static void main(String[] args) throws IOException
      {
            FileInputStream fln = new FileInputStream("Excel file
path");
            XSSFWorkbook book = new XSSFWorkbook(fln);
            XSSFSheet sheet = book.getSheet("Sheet name");
            XSSFRow row = sheet.getRow(2);
            XSSFCell cell = row.getCell(1);
            String cellValue = cell.getStringCellValue();
            System.out.println("Single cell value : "+cellValue);
      }
}
```

➢ TESTNG

1. What is TestNG?

TestNG is a testing framework which is inspired from Junit and NUnit but it is introducing some new functionalities that make it more powerful. Actually, TestNG is a library means collection of classes.

2. What are the Features of TestNG?

Features of TestNG:

a. Annotations.
b. Support for data driven testing with @Data Provider.
c. Support for parameters.
d. Support for parallel testing.
e. Support for grouping concept.
f. Support for cross browser testing.
g. Support for execution of particular test cases.

3. What is TestNG.xml file?

a. TestNG.xml file is a configuration file which has the execution information.

b. We can have number of .xml files inside the project.
c. The extension of the file should be .xml

4. **What is the need of TestNg.xml file?**
Actually, we cannot go into the individual classes and we cannot pick up the individual test cases to execute. But if we have the xml file, it can be used to execute the test cases whatever we want. So, that is why we need TestNG.xml file.

5. **What is the syntax for .xml file?**
Syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Suite" parallel="false">
     <test name="Test1">
          <classes>
               <class name="packageName.ClassName" />
               <class name="packageName.ClassName" />
          </classes>
     </test>

     <test name="Test2">
          <classes>
               <class name="packageName.ClassName" />
               <class name="packageName.ClassName" />
          </classes>
     </test>
</suite>
```

6. **What is meant by an Assertion?**
Assertion is an expression which verifies the actual test outcome with expected test outcome.

7. **Is the Assertion mandatory in tests?**
Yes, Assertion is mandatory. If there is no Assertion, the process of verification is not going to be done.

8. **What is Hard Assertion?**
Hard Assertion is an Assertion which throws an exception immediately upon failure of assertion.
Ex:

```
String expectedTitle = "RBG Technologies";
String actualTitle = driver.getTitle();
Assert.assertEquals(actualTitle, expectedTitle, "Error message");
```

Here, if the actual and expected get miss matched, the execution will stop immediately.

9. **What is Soft Assertion?**
Soft Assertion is an Assertion which does not throw any exception immediately upon failure of Assertion. But, occurred exception is going to be recorded.
Ex:

```
SoftAssert softAssert = new SoftAssert();
```

```
String expectedTitle = "RBG Technologies";
String actualTitle = driver.getTitle();
softAssert.assertEquals(actualTitle, expectedTitle, "Error
message");
softAssert.assertAll();
```

Note: When an Assertion gets failed, it throws an exception by calling the assertAll() method only.

10. **What is an Annotation?**
    TestNG Annotation is a piece of code which is inserted inside a program and used to control the flow of execution of test methods.

11. **What are the Annotations available in TestNG?**
    Annotations and its order:
    @Before Suite
    @Before Test
    @Before Class
    @Before Method
    @Test1
    @After Method
    @Before Method
    @Test2
    @After Method
    @After Class
    @After Test
    @After Suite

12. **What is Parameter in TestNG?**
    TestNG Parameters are the arguments that we pass to the test methods with multiple values. There are two ways through which we can pass the Parameters to the test methods.
    a. TestNG Parameters.
    b. TestNG Data providers.

13. **Why do we Disable a test case?**
    We can disable a set of test cases from getting executed. Consider a scenario where a serious bug occurs, we need to disable a test case form getting executed.

14. **How to Disable or Enable a test case from Class level?**
    From the Class level:
    We can Disable or Enable test cases from Class level.
    Ex1:

```
@Test(enabled = true)
public void test1()
{
      System.out.println("Test 1 is enabled");
}
                                                                    }
```

    Ex2:

```
@Test(enabled = false)
```

```
public void test2()
{
	System.out.println("Test 2 is disabled");
}
```

From the Suite level:
We can Disable or Enable test cases from Suite level or xml file also.
Ex:

```
<test name="RBG Technologies" enabled=false />
```

15. **What is Priority and how to define a Priority?**
    Priority:
    Priority is a Parameter inside the @Test Annotation. Whatever we define the least Priority that will be executed first.
    Ex:

```
@Test(priority = 2)
public void test1()
{
	System.out.println("Priority 2 is executed");
}

@Test(priority = 1)
public void test2()
{
	System.out.println("Priority 1 is executed");
}
```

    Output:
    Priority 1 is executed
    Priority 2 is executed

16. **Why do we need to use the Priority concept?**
    TestNG internally does not follow the test case execution orderly what we have defined. So, that is why we need to use the Priority concept in TestNG.

17. **What is the default value of Priority?**
    Default value of the Priority is Zero.
    @Test ➔ internally follows ➔ @Test(priority = 0)

18. **Can we pass negative values to the Priority?**
    Yes, we can also pass negative values to the Priority.
    Ex:

```
@Test(priority = -1)
@Test(priority = -2)
@Test(priority = -3)
```

19. **What happens when same Priority is provided to the multiple methods?**
    There is no problem. If we give same Priority to the multiple methods, TestNG again follows ASCII order between those methods.

Ex:

```
@Test(priority = 2)
public void test1()
{
	System.out.println("Test 1 is executed");
}

@Test(priority = 1)
public void test2()
{
	System.out.println("Test 2 is executed");
}

@Test(priority = 2)
public void test3()
{
	System.out.println("Test 3 is executed");
}

@Test(priority = 1)
public void test4()
{
	System.out.println("Test 4 is executed");
}
```

20. **What is Ignore Annotation?**
    @Ignore Annotation is used to ignore the test cases from getting executed whatever we want to ignore particularly.
    Note: Ignore Annotation is similar to @Test(enabled = false)

21. **How to ignore a specific test method?**
    If we want to ignore a specific test method, we have to follow as
    Ex:

```
@Ignore
@Test(enabled = true)
public void test()
{
	System.out.println("Test is ignored");
}
```

    Note: Even we use enabled = true, the test will be ignored because @Ignore annotation has the highest priority than the enabled attribute.

22. **How to ignore all test methods in a class?**
    If we want to ignore all test methods in a class, @Ignore Annotation should be placed at the Class level.
    Ex:

```
@Ignore
public class Annotations
{
```

```
	@Test
	public void test1()
	{
		System.out.println("Test 1 is ignored");
	}

	@Test
	public void test2()
	{
		System.out.println("Test 1 is ignored");
	}

}
```

23. How to ignore all test methods in a Package?

We can also ignore all test methods in a Package. We need to create a Java new file called package-info.java and in this file

Ex:

```
@Ignore
package TestNGDemo;
import org.testng.annotations.Ignore;
```

24. What is Test dependency?

Test dependency is a concept in which one test method is based on another test to run basically.

25. How to create a Test dependency on another tests?

Ex:

```
package TestNGDemo;
import org.testng.annotations.Test;
public class Dependency
{

	@Test(dependsOnMethods = {"Test2"})
	public void Test1()
	{
		System.out.println("Test1");
	}

	@Test(dependsOnMethods = {"Test3"})
	public void Test2()
	{
		System.out.println("Test2");
	}

	@Test
	public void Test3()
	{
		System.out.println("Test3");
	}
}
```

26. What happens when the Priority is set to the test methods?

When we are testing Test dependency, if we say any Priority, basically that Priority will not be considered and it will be ignored. It will consider only Test dependency concept.

Ex:

```
package TestNGDemo;
import org.testng.annotations.Test;
public class Dependency
{
	@Test(priority = 2)
	public void Test1()
	{
		System.out.println("Test1");
	}

	@Test(priority = 1, dependsOnMethods ={"Test1"})
	public void Test2()
	{
		System.out.println("Test2");
	}
}
```

Output:

Test1
Test2

27. What happens when the dependent test is failed?

Ex:

```
package TestNGDemo;
import org.testng.annotations.Test;
public class Dependency
{

	@Test
	public void test1()
	{
		System.out.println(10/0);
	}

	@Test(dependsOnMethods = {"Test1"})
	public void test2()
	{
		System.out.println("Test2");
	}

	@Test(dependsOnMethods = {"Test2"})
	public void test3()
	{
		System.out.println("Test3");
	}
```

```
}
```

Output:
// Test 2 and Test 3 are actually depended on Test 1. Here, Test 2 and Test 3 tests are going to be skipped because Test 1 got failed.

28. **Even the dependent test is failed, how to run a test method based on a failed test?**
Even the dependent test is failed, we can run the test method based on the failed test by a parameter called alwaysRun = true.
Ex:

```
package TestNGDemo;
import org.testng.annotations.Test;
public class Dependency
{

	@Test
	public void test1()
	{
		System.out.println(10/0);
	}

	@Test(dependsOnMethods = {"test1"}, alwaysRun = true)
	public void test2()
	{
		System.out.println("Test 2 is done");
	}

}
```

Output:
Test 2 is done

29. **What happen when the dependent test is ignored/skipped/deleted?**
Ex:

```
package TestNGDemo;
import org.testng.annotations.Ignore;
import org.testng.annotations.Test;
public class Dependency
{

	@Ignore
	@Test
	public void test1()
	{
		System.out.println("Ignored method");
	}

	@Test(dependsOnMethods = {"test1"})
	public void test2()
	{
```

```
		System.out.println("Test 2");
	}

	@Test(dependsOnMethods = {"test2"})
	public void test3()
	{
		System.out.println("Test 3");
	}

}
```

Output:
// Nothing will be run, skipped and failed.

30. **Even the dependent test is ignored, how to run a test method based on an ignored test method?**
Even the dependent test is ignored, we can run the test method based on the ignore test by a parameter called ignoreMissingDependencies = true.
Ex:

```
package TestNGDemo;
import org.testng.annotations.Ignore;
import org.testng.annotations.Test;
public class Dependency
{

	@Ignore
	@Test
	public void test1()
	{
		System.out.println("Ignored test");
	}

	@Test(dependsOnMethods = {"test1"}, ignoreMissingDependencies =
true)
	public void test2()
	{
		System.out.println("Test2 is done");
	}

}
```

Output:
Test 2 is done

31. **What is Grouping in TestNG?**
Groups in TestNG is a process of grouping different types of tests together into a group and running these tests by just in a single command.

32. **Why do we need Grouping?**
If we want to execute our tests as a part of some categorization then we need Grouping concept.

33. How to define Grouping?

Ex:

```
package TestNGDemo;
import org.testng.annotations.Test;
public class Grouping
{
	@Test(groups = {"smoke"})
	public void test1()
	{
		System.out.println("Test 1");
	}

	@Test(groups = {"smoke", "sanity"})
	public void test2()
	{
		System.out.println("Test 2");
	}

	@Test(groups = {"sanity"})
	public void test3()
	{
		System.out.println("Test 3");
	}

	@Test(groups = {"sanity", "regreesion"})
	public void test4()
	{
		System.out.println("Test 4");
	}

	@Test(groups = {"regression"})
	public void test5()
	{
		System.out.println("Test 5");
	}
}
```

34. Execution of Grouping from xml file?

Ex:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name=" suite ">
	<test name="Test 1>
		<groups>
			<run>
				<include name="sanity" />
			</run>
		</groups>
		<classes>
			<class name="package name. class name " />
		</classes>
```

```
        </test>
    </suite>
```

Output:

Test 2
Test 3
Test 4

Note:

If we use <exclude name = “regression”>, it will not be executed even it is along with any group.

35. How to run Test groups depend on another Test groups through xml file?

Ex:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="suite">
    <test name="Test 1>
        <groups>
            <dependences>
                <group name="sanity" depends-On="smoke" />
                <group name="regression" depends-on="sanity" />
            </dependencies>
        </groups>
        <classes>
            <class name="package name. class name " />
        </classes>
    </test>
</suite>
```

36. How to create a Test dependency on another group?

Ex:

```
package TestNGDemo;
import org.testng.annotations.Test;
public class Grouping
{

    @Test(groups ="smoke")
    public void test1()
    {
        System.out.println("Test 1");
    }

    @Test(groups ="smoke")
    public void test2()
    {
        System.out.println("Test 2");
    }

    @Test(dependsOnGroups = "smoke")
    public void test3()
```

```
	{
		System.out.println("Test 3");
	}

}
```

Output:
Test 1
Test 2
Test 3

37. What is a Data Provider?

Data Provider is like a container which passes multiple data to the methods.

38. How to use the Data Provider?

Ex:

```
package TestNGDemo;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
public class DataProviderConcept
{

	@Test(dataProvider = "loginTestData")
	public void test(String userName, String passWord)
	{
		System.out.println(userName);
		System.out.println(passWord);
	}

	@DataProvider(name = "loginTestData")
	public Object[] loginData()
	{
		Object [][] data = new Object [2][2];
		data[0][0] = "Admin";
		data[0][1] = "admin12";
		data[1][0] = "Admin";
		data[1][1] = "admin123";
		return data;
	}

}
```

Output:
Admin
Admin12
Admin
Admin123

39. What is an Invocation Count?

An Invocation Count is a one of the features available in TestNG which is used to run the same test for multiple times.

40. How to use an Invocation Count?

Ex:

```
package TestNGDemo;
import org.testng.annotations.Test;
public class InvocationCountConcept
{

	@Test(invocationCount = 3)
	public void test()
	{
		System.out.println("Test");
	}

}
```

Output:

Test
Test
Test

41. What is an Invocation Timeout?

While running a test case, some test cases take much more time than expected. In such a case, we can mark the test case as a failed test case by using an Invocation Timeout.

42. How to use an Invocation Timeout?

Ex:

```
package TestNGDemo;
import org.testng.annotations.Test;
public class InvocationTimeOutConcept
{

	@Test(invocationCount = 5, invocationTimeOut = 5000)
	public void test()
	{
		System.out.println("Test");
	}

}
```

Output:

Test 1
Test 1
Test 1

43. What is Listener?

Listener is an Interface that is listening to the events are performed by the TestNG.

44. Why do we need Listener in TestNG?

If we want to perform any actions during the execution then we need Listeners.

45. How to use Listeners?

Ex:

```
package TestNGDemo;
import org.testng.annotations.Listeners;
import org.testng.annotations.Test;
@Listeners(ITestListenerclass.class)
public class ListenersConcept
{

	@Test
	public void test1()
	{
		System.out.println("Test 1");
	}

	@Test
	public void test2()
	{
		System.out.println("Test 2");
	}

}
```

46. How to use the Listeners at Suite level?

Ex:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Listener suite ">
	<listeners>
		<listener class-name="package name. class name " />
	</listeners>
	<test name="test 1>
		<classes>
			<class name="package name. class name ">
				<methods>
					<include name="test method 1 />
					<include name="test method 2 />
				</methods>
			</class>
		</classes>
	</test>
</suite>
```

47. What is Parallel Testing?

Parallel Testing means executing the test cases along with another one. It is also known as side-by-side testing.

48. Why do we need Parallel Testing?

The purpose of Parallel Testing is, we no need to send extra time. The time is going to be saved. If we perform test cases as individually, it takes too much of time to perform.

Ex:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="suite name " Parallel="test" thread-count=2">
     <test name="test">
          <classes>
               <class name="package name. class name ">
                    <methods>
                         <include name="launch" />
                         <include name="login" />
                         <include name=verifyTitle" />
                    </methods>
               </class>
          </classes>
     </test>
</suite>
```

49. What is Cross-Browser Testing?

Cross-browser Testing is a one type of testing to verify whether an application is working fine or not across the different types of browsers.

Ex:

```
public class CrossBrowserConcept
{
     WebDriver driver;

     @BeforeMethod
     @Parameters("browser")
     public void init(String browser)
     {
          switch (browser.toLowerCase())
          {
          case "chrome":
               System.setProperty("webdriver.chrome.driver", "path
of the driver file");
               driver = new ChromeDriver();
               break;

          case "ie":
               System.setProperty("webdriver.chrome.driver", "path
of the driver file");
               driver = new InternetExplorerDriver();
               break;

          case "firefox":
               System.setProperty("webdriver.chrome.driver", "path
of the driver file");
               driver = new FirefoxDriver();
               break;
```

```
            default:
                driver = null;
                break;
            }

      }

      @Test
      public void verifyPageTitle()
      {
            driver.get("https://www.Rbg Technologies");
            Assert.assertEquals(driver.getTitle(), "Rbg
technologies");
      }

      @AfterMethod
      public void teardown() throws IOException
      {
            driver.quit();
            Runtime.getRuntime().exec("taskkill/f/IM
chromedriver.exe");
      }

}
```

Xml file:

Ex:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="suite name ">
      <test name="test1">
            <parameter name="browser" value="Chrome" />
                  <classes>
                        <class name="package name. class name " />
                  </classes>
      </test>
      <test name="test2">
            <parameter name="browser" value="IE" />
                  <Classes>
                        <class name="package name. class name " />
                  </classes>
      </test>
</suite>
```

50. What are the Reports in TestNG?

Whenever we are executing any tests, we need to track some information that means what is the status of tests execution, how much time it will take to complete the execution and if the test is failed then what is the exception actually. We can get this information by Reports.

➢ JIRA

1. **What is JIRA?**
   Jira is a bug tracking tool and also designed to handle team coordination in Agile software development as well. That means we can do all Agile related activities in Jira.

2. **Agile Activities:**
   - We can create a Project and we can add users or peoples which is done by the Administrators.
   - We can create a backlog, Epic and User stories which is done by the Product owner.
   - Creating Sprint in Jira, add users' stories to the Sprint and Sprint life cycle which is done by the Scrum master.

3. **Test Management Activities:**
   - How to create test cases in Jira with Zephys plugin.
   - Creating test cycles.
   - Executing and updating test cases in Jira.

4. **How to create a Project in Jira?**
   Go to Dash board ➔ Go to Projects ➔ Create a Project ➔ Project name ➔ Go to another Project, we have to choose the Scrum Project ➔ Click on create.
   Once it is done, it will create new Project, for this Project by default backlog page is displayed. Once the Project is created, we need to add users to this Project like Product owner, Scrum master, Developers and Testers.

5. **How to add users to the Project?**
   Go to Jira settings ➔ Click on user management ➔ under this user management there is a user's ➔ then we have to give the particular details ➔ Click on invites users.

6. **How to create an Epic in the Product backlog?**
   Go to current Project ➔ Click on current Project ➔ Backlog page is displayed ➔ Create a version number ➔ Go to Epics tab & Create an Epic name ➔ There is an issue type, click on an Epic ➔ Type Epic name ➔ Enter on create button

7. **How to add User stories in Jira?**
   Go to Project ➔ Go to backlog page ➔ Click on Epics tab ➔ enter + button in dash board or under there is a one option ➔ It will give us a one window ➔ There is an option called issue type ➔ Select story option ➔ Provide summary ➔ Provide description ➔ Assign story to the user ➔
   Enter on a create button

8. **How to create a Sprint?**
   Once the user stories are added, Sprint needs to be created.
   Backlog ➔ Click on a create Sprint button ➔ add user stories to the Sprint ➔ Click on start Sprint ➔ fill the required details (2 weeks of Sprint time) ➔ Then Sprint page is displayed or Scrum board in active Sprint.

9. **Bug Reporting:**

Go to Sprint page ➔ Click on + ➔ Issue type Bug ➔ Enter particular details ➔ Send it to the developer.
If the all operations are done, click on the complete Sprint then Sprint report will be created.

10. **How to create a Test cases?**
    Test case is also a kind of issue. Zephys plugin has to be added.
    Click on + button ➔ Issue type called Test ➔ Summary ➔ Description ➔ Create

➢ FRAMEWORK

Languages:
In our Project we are using Java language.

Type of Framework:
We are using Data driven framework by using Page Object Model with Page Factory.

POM:
As per the Page Object Model, we have to maintain a class for every web page. Each web page has a separate class and that class holds the functionality and members of that web page. Separate classes for every individual test.

Packages:
We have separate packages for Pages and Tests. All web page related classes come under the Pages package and all tests related classes come under the Tests package. Page classes are used to store element locators and these page classes methods are executed by invoking from the Test classes.

Test Base Class:
This Class is responsible for loading the configurations from Properties files, Initializing the WebDriver, Implicit Waits, Extent Reports and also to create the object of File Input Stream which is responsible for pointing towards the file from which the data should be read.

Utility Class:
The reason behind creating the Utility Class is to achieve code reusability. This Class extends the Test Base Class to inherit the properties of Test Base in TestUtil.

Properties File:
This file stores the information that remains static throughout the framework such as browser-specific information, application URL and Screenshots path etc.

Screenshots Folder:
Screenshots will be captured and stored in a separate folder and also the Screenshots of failed test cases will be added to the Extent Reports.

Test Data:
All the Test Data will be kept in an Excel sheet. We can pass the Test Data and handle Data Driven Testing by integrating Apache POI.

TestNG:
Using TestNG for Assertions, Grouping, Cross browser and Parallel Testing.

Maven:
Using Maven for build, execution and dependency purpose.

Version Control Tool:
Using Git as a repository to store our test scripts.
Jenkins:
By using Jenkins CI & CD tool, we can execute test cases on a daily basis and also on nightly basis execution on schedule.

Extent Reports:
For Reporting purpose, we are using Extent Reports. It generates beautiful HTML reports for maintaining the logs.

➢ GIT

Git is a version control tool to store any .txt files from small to very large Projects with speed and efficiency. It allows multiple Developers and Testers to work on the same code base simultaneously, keeping track of changes and versions of the code.

Git hub:
Git hub is a Remote repository hosting service. Every version must be stored in Remote servers.
Project ➔ Websites ➔ Updates 1.2 ➔ Redeploy ➔ may be bugs ➔ Revert back to previous version ➔ Older version 1.1

Elements:
Working directory ➔ **git add or git add.** ➔ Staging area ➔ **git commit** ➔ Local Repository ➔ **git push** ➔ Remote Repository

Working directory:
The place where we want to work with Git from is called Working directory. From there when we run git init command in a directory, Git initializes a new repository in that directory and creates a .git sub directory which contains all of the necessary Git metadata and also called Untracked area.

Commands:

- Repository has to be created in GitHub.
- Copying the URL from Remote Repository.
- Opening the folder, click Git bash here.
- **git nit** ➔ Initialize a local Git Repository
- . git folder will be created.
- There should be some files.
- **git remote add origin** URL of the Remote Repository ➔ adding Remote Repository.
- **git add** filename or **git add.** ➔ adding files to the Staging area.

- **git commit -m "**message" ➔ commit changes.
- **git push -u origin master** ➔ One time activity.
- **git add.**
- **git commit -m "**message**"**
- **git push**
- **git pull** ➔ Remote to local.
- **git checkout -b "**branch name" ➔ creating a new branch.
- **git add** filename
- **git commit -m "**message"
- **git push origin** branch name
- **git clone** URL of the Remote Repository ➔ creating a local copy of a Remote Repository.
- **git rm -r** filename ➔ Removing a file from Staging area.
- **git branch** branch name ➔ Creating a new branch.
- **git branch -d** branch name ➔ Deleting a branch.
- **git merge** branch name ➔ Merging a branch into active branch.
- **git merge** source branch name target branch name ➔ Merging a branch into a target branch.
- **git push origin** branch name ➔ Pushing a branch to our Remote Repository.
- **git pull origin** branch name ➔ Pulling the changes from Remote Repository.
- **git status** ➔ Checking the status.

Note: Here, whatever work is done from the branches, PR (Pull Request) will be raised. Once the team lead or owner of the Master accepts, PR will be merged into Master.

➢ JENKINS

1. **What is Jenkins?**
   Jenkins is a tool which is used to automate Build and Deployment process.

2. **Process of Build and Deployment:**
   a. Take the latest source code from the GitHub.
   b. Compile Project source code.
   c. Execute unit test cases.
   d. Perform code review using Sonar Qube.
   e. Package the application (Jar/War).
   f. Upload build artifacts in Nexus.
   g. Deploy the application in server.

3. **Application Environments:**
   A platform which is used to run our application.
   GitHub ➔ Dev ➔ QA ➔ UAT ➔ Production or live.
   Dev Environment:
   Developers will use to perform code integration testing.

   QA Environment:
   Testers will use to perform functional testing.

UAT (User Acceptance Testing) Environment:
Client-side team will perform to know the client satisfaction.

Pilot Environment:
It is also called as Pre-production Environment. It will be used to perform Performance testing.

4. **What is CI (Continuous Integration) and CD (Continuous Delivery or Continuous Deployment)?**
   Whenever we change anything in the code, that will be ready to test and deploy.
   CI (Continuous Integration):
   When code got changed, it should be ready to test.

   CD (Continuous Delivery or Continuous Deployment):
   Keep it ready to release or deploy the Project to the production.

   Note: For production deployment, we need to take client approval.

5. **Work flow:**
   a. We can attach Git, Maven, Selenium and Artifactory (final code) plugins to the Jenkins.
   b. Once developers put the code in GitHub, Jenkins pulls that code and sends to the maven for build.
   c. Once build is done, Jenkins pulls that code and sends to the Selenium for testing.
   d. Once test is done, Jenkins pulls that code and sends to the Artifactory as per requirement.
   e. We can also deploy with Jenkins.

➢ FRAMEWORK ARCHITECTURE

- **Test Framework** ➔ Data Driven, Keywork Driven
- **Maven** ➔ Build and Dependencies ➔ POM.xml
- **Project** ➔ XLSReader
- **Src/main/Java** ➔ Html Reports, Mail, Config.properties, Generic Keywords (Reusability methods) and Screenshot
- **Src/test/Java** ➔ Pages package ➔ Home page, login page, new account page. Test package ➔ Home page test extends Base Test, login page test extends Base test and new account page test extends Base test
- **Web Driver folder** ➔ Chrome driver, Edge driver, IE driver and Firefox driver etc.
- **Screenshot folder** ➔ Screenshot files
- **Extent Reports** ➔ HTML reports files
- **Test Data** ➔ Excel data files
- **Test Output** ➔ TestNG.failed.xml
- **Object Repo** ➔ Config.properties ➔ Browser, URL
- **TestNG Results** ➔ logs or information
- **TestNG.xml** ➔ Grouping, Parallel, cross browsing testing

Oops concept involved in Framework:

1. **Abstraction** ➔ Interfaces ➔ Web driver, JavaScript Executor, WebElement, Waits, Takes Screenshot and Abstract Class.

2. **Encapsulation** ➔ Page factory, Private funA by getter and setter
3. **Inheritance** ➔ Any test classes extends Base test
4. **Polymorphism** ➔ Methods overloading ➔ frame(), println(), Assertion(). Methods Overriding ➔

➢ API TESTING - REST ASSURED

1. **What is an API?**
   API (Application Programming Interface) is a one type of intermediate software which exchanges the data or information from one software to another software. Json or xml formats are used to exchange the data.

2. **What is REST?**
   Rest stands for Representational State Transfer and is an architectural style based on the Hypertext Transfer protocol (HTTP) for developing web-based applications.
   - What it basically does is, it provides a lot of several guidelines or principles and which allows us to build web services and hence they are called RESTFUL API'S.
   - These guidelines ensure that requests and responses are sent easily and efficiently between client and server using standardized HTTP methods.

3. **What are the principles REST?**
   Any API that follows all these five principles or requirements is becomes a REST API.

   1. **Client-server decoupling:** Client-server can only interact in a series of requests and responses. Only client can make requests and only servers can send responses. Apart from that there is no further communication.

   2. **Uniform Interface:** All communications between the client and server must follow the same protocol. A uniform interface simplifies integrations because every application is using the same language to request and send the data.

   3. **Stateless:** In stateless communication, the server does not store any information about past requests and responses. Each request and response contain all information needed to complete the interaction. Stateless communication reduces server load, saves memory and improve performance.

   4. **Layered system:** Layers are servers that sit between the client and API server. These additional servers perform various functions like identifying spam and improving performance.

   5. **CACHEABLE:** Cacheable server responses indicate whether the resource is cacheable or not. So that clients can cache any resource to improve performance.

4. **What are RESTFUL APIs?**
   REST APIs are also called as RESTFUL APIs. The API'S which follow REST principles.

5. **What is an API Testing?**

API testing is a testing of API's and it is an integration with the services. Like for example whether the data that we are retrieving is in the correct format or not, whether the data is correct or not, whether it has all the required keys, whether it has all the expected data or not and whether it is secured or not etc.,

6. **Share some of the advantages of API testing?**
   The reason to perform API testing is, APIs are critical infrastructure to any application. They are extremely critical for a reason that, one wrong thing in the API can bring down the entire application. It can lead to a lot of damage. So, hence we have to conduct a thorough API testing.
   a. API testing is less time consuming than the functional testing.
   b. It is cost-effective.
   c. It is language-independent and time-effective. Most of the responses that we get from the servers are planed JSON text or XML which can be read by anybody. We can perform an extensive testing without even knowledge of the back end what is happening and how it is happening.

7. **Is API considered as a software?**
   API is not a software but rather an interface to provide data exchange and functionality among different software applications.

8. **Which protocol do REST APIs use?**
   REST APIs use the HTTP protocol to communicate with the client. This allows REST APIs to be easily deployed over the internet.

9. **Which HTTP request methods are supported by REST?**
   An HTTP request method indicates which action the client wants the API to perform on a resource. Any HTTP method that we use, is trying to do a certain operation on a resource.
   They are mainly four primary requests. They are
   a. GET
   b. POST
   c. PUT – PATCH
   d. DELETE

10. **What are CRUD operations?**
    CRUD stands for Create, Read, Update, and Delete. These are the basic actions that can be performed on the database through a REST API.
    Each action corresponds to a HTTP request method. Like
    a. Create – POST
    b. Read – GET
    c. Update – PUT
    d. Delete – DELETE

11. **What is the difference between PUT, POST and PATCH?**
    PUT: PUT request is used for both creating and updating a new object in the database. If the resource is already existing, then PUT will update the resource. If not, it will create one.
    Note: But as part of RESTFUL API's always make sure that we use PUT only for updating, never for creating that is against the guidelines.

POST: POST request is used to create a new resource.

PATCH: PATCH is used to apply the partial modifications to a resource.

12. What are some drawbacks or disadvantages of REST?

- While statelessness is a benefit of REST. But sometimes it can be disadvantage too. REST does not preserve state that means it cannot go back to the previous state or go back to the next line how we have in some of the modern framework state management.
- In other words, the server does not store any information about past requests and responses. If preserving state is necessary, that responsibility falls on the client.
- Additionally, REST is less strict with its security measures than SOAP.
- It also makes REST a poor choice for sending the confidential information between servers and clients

13. What are some benefits or advantages of REST?

- REST is based around HTTP and fits within the existing infrastructure of the web. Making it easy to implement by web application.
- REST uses simple web technologies like JSON and XML, making it easy to learn.
- REST is a lightweight architecture.
- Applications build with REST are generally faster than those build with other types of API's.
- REST is easy to test in the browser with API testing tools like Postman or Katalon studio etc.,

14. State some of the public REST API's?

Some of the examples of public RESTFUL APIs are

a. Google maps API
b. Twitter API
c. Weather API
d. YouTube API

15. What is messaging in the context of REST?

- In REST, messaging refers to the back-and-forth communication between the client and API.
- An interaction always starts with the client messaging to the API with an HTTP request.
- The API processes this request, then sends back an HTTP response that gives the status of the request.

16. What is difference between API and web service?

- All web services are API's but not all APIs are web services. Because REST APIs need to follow certain principles and guidelines. Only when they follow, they become RESTful APIs otherwise they are just APIs and they are called web services.
- Web services might not contain all specifications and cannot perform all tasks that APIs would perform.
- Web services use three primary protocols for communication: SOAP, REST, and XML-RPC whereas APIs may be exposed to multiple ways.
- Web services always need a network to operate while APIs can operate over the internet.

17. What is Resource?

- Every data at the server is referred as a resource. A resource could be an HTML file, a text file, an image, a video, or an executable code file. It may be anything. When we say, we are posting any data that means we are creating a resource. When we say, we are trying to get any data, that means we are requesting a resource.
- A resource is identified with a Uniform Resource Identifier (URI) with an HTTP request method.

18. What is Uniform Resource Identifier (URI)?

- URI stands for Uniform Resource Identifier.
- In REST, a resource is identified with a Uniform Resource Identifier (URI).
- Each resource has its own unique URI which means we include in a HTTP request that allows clients to target a resource and perform action on it.
- The process of targeting a resource with its URI is called 'Addressing'.
- The format of URI is as follows.
  Ex: <protocol>://<service name>/<Resource type>/<Request 10>

19. Which languages are primarily used to request resources in REST APIs?

There are 2 common languages for representing a resource in REST APIs.

- **XML:** eXtensible markup language.
- **JSON:** JavaScript Object Notation.

20. What are idempotent methods?

- Idempotent methods are the methods that returns the same outcome irrespective of how many times the same request has been made.

21. What are the main parts of HTTP requests?

HTTP requests are sent by the client to the API. They are five main components of HTTP request in REST.

Request line: Request line indicates the intended action of the request and include.

Request method: A request method indicates the HTTP action to be performed on the resource (I.e. Get, Post, Put and Delete).

URI: A URI indicates the identification of a resource on the server.

HTTP Request Header: An HTTP request header is a key part of the HTTP request. HTTP headers are used to provide additional information about the request. Such as the user agent, format of the request body, Authorization, Content-Type and cookies etc.,

HTTP Request body: It contains data which is sent from the client to the server. This body is used when the request method involves sending data to the server to create or update a resource, such as POST, PUT, or PATCH.

22. What are the main parts of an HTTP response?

HTTP responses are sent by the server to the client after receiving an HTTP request. The main parts of HTTP response are.

a. **Status line:** The status line is the first line of the response and provides essential information about the outcome of the request. It contains some components like
   - HTTP Version: The version of the HTTP protocol used (e.g., HTTP/1.1).
   - Status Code: A three-digit number indicating the result of the request (e.g., 200, 404).
   - Status Message: A brief description of the status code (e.g., "OK", "Not Found").

b. **Headers:** Headers provide additional details and instructions (e.g., content type, length, file format of the returned resources and caching information etc.,).

c. **HTTP Response Body:** It Contains the actual data returned by the server to the client. and it is also called as Payload.

23. What is Caching?

It is a method of temporarily storing a copy of server response in a location. So that means in order to quickly retrieve it. Let us say we are making some API calls with same data and we are getting the same response. So, it makes sense that we cache them because next time we can serve from the cache not again going to the server. So, this decreases the server load and caching also makes the application run faster.

24. What is Payload?

Payload refers to the data in the body of HTTP request or response. Whatever data that we send in request or that we get from the response that becomes a payload. The payload can be XML or JSON format.

25. What is the difference between REST and SOAP?

- Soap is much stricter in protocol than the REST APIs.
- REST is not a protocol. It is an architecture style and it has some guidelines. Any API that follows those guidelines becomes a REST API.
- SOAP APIs are considered more secure than REST. Because it has its own infrastructure and it has its own processing etc.,
- REST allows caching of response. Whereas every SOAP request is a brand-new transaction and there is no caching mechanism in SOAP.
- REST allows us to encode the data in any format through XML or JSON whereas SOAP is strictly following XML format. So, that is why they are strict. If there are any errors in XML format or schema, it throws an error whereas REST will not do that.

26. What are the tools available for testing APIs?

There are many software tools available for testing APIs. Such as Postman, JMeter and Katalon studio etc.,

27. How to secure REST APIs?

a. **Use HTTPS:** Encrypts data exchanged between the client and server to ensure privacy.
b. **Authenticate:** Uses tokens like JWT to verify user identity and secure access.
c. **Authorize:** Controls user access based on predefined roles and permissions.
d. **Validate Input:** Ensures data is clean and safe to prevent security vulnerabilities.
e. **Rate Limiting:** Prevents abuse by capping the number of requests a user can make.
f. **Monitor and Log:** Keeps records of API usage to identify and address potential threats.

g. **Implement CORS:** Manages cross-origin resource sharing to control who can access the API.
h. **Regularly Update:** Keeps the system secure by applying the latest security patches and updates.

28. What is difference between API testing and Unit testing?

a. Unit testing is a white box testing and API testing is a black testing.
b. Unit testing is used to verify that each unit in isolation works as expected while API testing is used to ensure full functionality of the system.

29. What is a Pre-Request script in Postman?

Pre-Request script is a one type of script that runs before execution of a request.

30. How should we test the API security?

To test security of the API during API testing, we need to validate two components. They are.

a. Authentication: Whether the identity of the end-user is correct or not.
b. Authorization: Whether the user is allowed to access the resource.

31. Which options help to identify the type of API requests?

HTTP methods help to identify the type of API requests.

32. What are the challenges faced in API testing?

a. API chaining or sequencing API calls.
b. Testing parameter combinations.
c. Frequent schema changes.
d. Access to the data base.

33. What must be checked when performing API testing?

When performing API testing, we need to validate some components. Like

a. Accuracy of data in the response body .
b. Schema validations.
c. HTTP status codes.
d. Status messages.
e. Response time.
f. Cookies and Headers.
g. Authentication and Authorization.
h. Error handling through error messages and status codes.

34. What kind of bugs does API testing find most commonly?

a. Authentication and Authorization Issues: Problems with verifying user identity or access controls.
b. Data Validation Errors: Incorrect data formats, missing required fields, or invalid input handling.
c. Performance Bottlenecks: Slow response times or high latency under load.
d. Endpoint Misconfigurations: Incorrect or missing API endpoint paths and methods.
e. Error Handling Problems: Inconsistent error messages and status codes.
f. Integration Issues: Problems with how the API interacts with other services.
g. Security Vulnerabilities: Exposures to common security issues like SQL injection, cross-site scripting (XSS), or insecure data transmission.

35. **What are the Options in REST APIs?**
    In REST APIs, the `OPTIONS` method is used to describe available communication methods and capabilities for target a resource.

36. **What is API chaining?**
    API chaining is the process of invoking multiple API requests, where the output of one request is used as the input for the next to achieve a series of interconnected operations.

37. **How many ways can we create a request body?**
    We can create a request body in 4 ways. They are

    a. **Hash Map:**

```
// Post request by using HashMap
@Test
void testPostUsingHashMap() {

	HashMap<String, String> data = new HashMap<String,
String>();
	data.put("name", "morpheus");
	data.put("job", "leader");

	RestAssured.given()
	.contentType("application/json")
	.body(data)

	.when()
	.post("https://reqres.in/api/users")

	.then()
	.statusCode(201)
	.header("Content-Type", "application/json; charset=utf-
8")
	.log().all();

}
```

    b. **Org.json (through pom.xml):** If we add Org.json dependency in pom.xml, then we will get some additional classes and methods through which we can create request body.

```
// Post request by using Org.json
@Test
void testPostUsingJson() {

	JSONObject jsonData = new JSONObject();
	jsonData.put("name", "morpheus");
	jsonData.put("job", "leader");

	RestAssured.given().contentType("application/json")
	.body(jsonData.toString())

	.when()
	.post("https://reqres.in/api/users")
```

```
		.then()
		.statusCode(201)
		.header("Content-Type", "application/json; charset=utf-
	8")
		.log().all();

	}
```

c. **Pojo class (Plain Old Java Object):**

```
	package JSONPack;

	public class Pojo_PostRequest {

		String name;
		String job;

		public String getName() {
			return name;
		}

		public void setName(String name) {
			this.name = name;
		}

		public String getJob() {
			return job;
		}

		public void setJob(String job) {
			this.job = job;
		}

	}

	// Post request by using POJO class
	@Test
	void testPostUsingPojoClass() {

		Pojo_PostRequest pdata = new Pojo_PostRequest();
		pdata.setName("morpheus");
		pdata.setJob("leader");

		RestAssured.given()
		.contentType("application/json").body(pdata)

		.when()
		.post("https://reqres.in/api/users")

		.then()
		.statusCode(201)
		.header("Content-Type", "application/json; charset=utf-
	8")
```

```
        .log().all();

    }
```

d. **External JSON file:** We can create a data through Json file, first we need to create one Json file with the extension.json. Inside this we need to keep some data whichever data we want to post.

```
// Post request by using Json external file
@Test
void testPostUsingExternalJsonfile() throws
FileNotFoundException {

      File f = new File(".\\body.json");
      FileReader fr = new FileReader(f);
      JSONTokener jt = new JSONTokener(fr);
      JSONObject jdata = new JSONObject(jt);

       RestAssured.given()
      .contentType("application/json")
      .body(jdata.toString()

      .when()
      .post("https://reqres.in/api/users")

      .then()
      .statusCode(201).header("Content-Type",
"application/json; charset=utf-8")
      .log().all();

      System.err.println(jdata);
}
```

38. How to specify Path parameters and Query parameters?

http://reqres.in/api/users?page=2
http://reqres.in/api/users?page=2&page=3

In the URL `http://reqres.in/api/users?page=2'
Path Parameter: /api/users
Query Parameter: page=2

39. What is Path parameter?

Path parameter is a dynamic part of the URL that is used to identify a specific resource. Path parameters are specified within the braces {} in the URL path.

```
@Test
void testPathAndQueryParameters() {

      //https://reqres.in/api/users?page=2
      RestAssured.given()
      .pathParam("myPath", "users")
      .queryParam("page", 2)
```

```
        .queryParam("id", 7)

        .when()
        .get("https://reqres.in/api/{myPath}")

        .then()
        .statusCode(200)
        .log().all();
}
```

40. What is Query parameter?

Query parameter is a key and value pair sent in the URL. Query parameters are used to filter, sort, paginate or provide additional context to the resource being requested. Query parameters are added to the URL after the '?' symbol. Multiple query parameters are separated by '&' symbol.

```
@Test
void testPathAndQueryParameters() {

      //https://reqres.in/api/users?page=2
      RestAssured.given()
      .pathParam("myPath", "users")
      .queryParam("page", 2)
      .queryParam("id", 7)

      .when()
      .get("https://reqres.in/api/{myPath}")

      .then()
      .statusCode(200)
      .log().all();
}
```

41. What are Cookies and Headers?

Cookies: Sometimes we will get some cookies and headers as a part of the response. Cookies are a mechanism to store small pieces of data on the client side. They are sent by the server along with the response. Cookies are used for session management or tracking user behavior etc.,

```
public class CookiesDemo {

      @Test(priority=1)
      void testCookies(){

            RestAssured.given()

            .when()
            .get("https://www.google.com/")

            .then()
            .cookie("AEC", "AVYB7coIIzcZ--
LyCYTGFKPrFY6KMon77nIwturVg2J9tarkiOcH6Aw7wgY")
            .log().all();
```

```
	}

	@Test(priority=2)
	void getCookiesInfo(){

		Response response = RestAssured.given()

		.when()
		.get("https://www.google.com/");

		//get single Cookie information
		String cookieValue = response.getCookie("AEC");
		System.out.println("The value of cookie is :
"+cookieValue);

		//get all Cookies information
		Map<String, String> allCookiesValues =
response.getCookies();
		Set<String> allCookieNames = allCookiesValues.keySet();
		System.out.println("All Cookies names :
"+allCookiesValues.keySet());

		for(String cookieName:allCookieNames){
			String cookiesValue =
response.getCookie(cookieName);
			System.out.println(cookieName+"--"+cookiesValue);
		}
	}

}
```

Headers: Headers are the part of request and response. Headers provide metadata about the request or response. Such as information about the server, content length, caching directive etc.,

```
public class HeadersDemo {

	@Test(priority = 1)
	void testHeaders() {
		RestAssured.given()

		.when()
		.get("https://www.google.com/")

		.then()
		.header("Content-Type", "text/html; charset=ISO-8859-1")
		.and()
		.header("Content-Encoding","gzip")
		.and()
		.header("Server","gws")
		.log().all();
	}

	@Test(priority=2)
	void getAllheaders() {
		Response response = RestAssured.given()
```

```
                .when()
                .get("https://www.google.com/");

                //Get single Header information
                String headerValue = response.getHeader("Content-Type");
                System.out.println("The value of Content-Type value is :
"+headerValue);

                //get all Headers information
                Headers allHeaders = response.getHeaders();
                for(Header header:allHeaders){
                     System.out.println(header.getName()+"--
"+header.getValue());
                }

          }

     }
```

42. How to parse JSON response?

```
public class ParsingJsonResponseData {
     @Test()
     void testJsonResponse() {
          // Approach 1
          RestAssured.given()
          .contentType("ContentType.JSON")

          .when()
          .get("https://reqres.in/api/users?page=2")

          .then()
          .statusCode(200)
          .body("data[3].first_name", equalTo("Byron"))
          .log().headers();
     }

     @Test()
     void testJsonresponse2(){
          //Approach 2
          Response response = given()
          .contentType("Content-Type")

          .when()
          .get("https://reqres.in/api/users?page=2");
          Assert.assertEquals(response.getStatusCode(),200);
          Assert.assertEquals(response.getHeader("Content-
Type"),"application/json; charset=utf-8");
          String lastName =
response.jsonPath().get("data[3].last_name").toString();
          Assert.assertEquals(lastName,"Fields");
     }
```

```
	@Test()
	void testJsonResponseBodyData(){
		Response response = given()
		.contentType("Content-Type")

		.when()
		.get("https://reqres.in/api/users?page=2");

		//JSONObject class
		boolean status = false;
		JSONObject jo = new
JSONObject(response.getBody().asString());
		for(int i=0; i<jo.getJSONArray("data").length(); i++){
		String firstnames =
jo.getJSONArray("data").getJSONObject(i).get("first_name").toString
);

		System.out.println(firstnames);

		if(firstnames.equals("Lindsay")){
			status = true;
			break;
		}
	}

		Assert.assertEquals(status, true);
		//validate total
		double total = 0;
		for(int i=0; i<jo.getJSONArray("data").length(); i++){
			String ids =
jo.getJSONArray("data").getJSONObject(i).get("id").toString();
			total = total+Double.parseDouble(ids);
		}
		System.out.println("Total ids : " + total);
		//Assert.assertEquals(total, 57.0, 0.0);
	}

}
```

43. How to parse HML response?

```
public class ParsingXMLResponse {

	@Test()
	public void testXMLResponse(){
		//Approach 1
		RestAssured.given()

		.when()
		.get("http://restapi.adequate.com/api/Traveler?page=1")
```

```
                .then()
                .statusCode(200)
                .header("Content-Type", "application/xml; charset=utf-8");
                //.body("TravelerinformationResponse.page", equaTo("1"));
        }

        @Test()
        public void  testXMlResponse2(){
                //Approach 2
                Response response = RestAssured.given()

                .when()
                .get("http://restapi.adequate.com/api/Traveler?page=1");

                Assert.assertEquals(response.getStatusCode(), 200);
                Assert.assertEquals(response.header("Content-Type"),
"application/xml; charset=utf-8");
                String pageNumber =
response.xmlPath().get("TravelerinformationResponse.page").toString
);
                Assert.assertEquals(pageNumber, "1");
        }

        @Test()
        public void testXMlResponseBody(){
                //Aproach 3
                Response response = RestAssured.given()

                .when()
                .get("http://restapi.adequate.com/api/Traveler?page=1");

                XmlPath xml = new XmlPath(response.asString());
                //Total number of travelers
                List<String> travelers =
xml.getList("TravelerinformationResponse.Travelers.TravelerInformati
on");
                Assert.assertEquals(travelers.size(), 10);
                //verify traveler name is present in the response
                List<String> travelersNames =
xml.getList("TravelerinformationResponse.Travelers.TravelerInformati
on.name");
                boolean status = false;
                for(String travelerName : travelersNames){
                        if(travelerName.equals("Naidu")){
                                status = true;
                                break;
                        }
                }
                Assert.assertEquals(status, false);
        }
```

```
}
```

44. How to validate file upload and file download?

```
public class FileUpload {
	@Test()
	public void singleFileUpload(){
		File myFile = new File("File path");
		RestAssured.given()
		.multiPart("file",myFile)
		.contentType("multipart/form-data").when().post("URL")

		.then()
		.statusCode(200)
		//.body("filename", equalTo("Test1.txt"))
		.log().all();
	}

	@Test()
	public void MultipleFilesUpload(){
		File file1 = new File("File path");
		File file2 = new File("File Path");

		RestAssured.given()
		.multiPart("files",file1)
		.multiPart("files",file2)

		.when()
		.post("URL")

		.then()
		.statusCode(200)
		//.body("[0].filename", equalTo("Test1.txt"))
		//.body("[1].filename", equalTo("Test2.txt"))
		.log().all();
	}

	@Test()
	public void moreThanTwoFilesUpload(){ // May not work in all
kinds of API's
		File myFile1 = new File("File path");
		File myFile2 = new File("File path");
		File myFile3 = new File("File path");
		File[] fileArray = {myFile1, myFile2, myFile3};

		RestAssured.given()
		.multiPart("files", fileArray)

		.when()
		.body("URL")
```

```
                .then()
                .statusCode(200)
                //.body("[0].filename", equalTo("Test1.txt"))
                //.body("[1].filename", equalTo("Test2.txt"))
                .log().all();
        }

        @Test()
        public void fileDownLoad(){
                RestAssured.given()

                .when()
                .get("URL/downloadFile/Test1.txt")

                .then()
                .statusCode(200)
                .log().all();
        }

}
```

45. How to perform JSON schema validation?

JSON Schema defines the structure and constraints of JSON data, enabling validation and documentation. To perform JSON Schema validation, we can use the body method with the 'JsonSchemaValidator.matchesJsonSchemaInClasspath'.

```
package day6;
import static io.restassured.RestAssured.given;
import org.testng.annotations.Test;
import io.restassured.module.jsv.JsonSchemaValidator;

public class JSONSchemaValidation {

        @Test()
        public void jsonschemaValidation(){
                given()

                .when()
                .get("https://reqres.in/api/users?page=2")

                .then()

.assertThat().body(JsonSchemaValidator.matchesJsonSchemaInClasspath(
"json_Schema_body.json"));
        }

}
```

46. How to perform XML schema validation?

XML Schema defines the structure and constraints of XML data, enabling validation and documentation. To perform XML Schema validation, we can use the body method with the 'RestAssuredMatchers.matchesXsdInClasspath'.

```
package day6;
import static io.restassured.RestAssured.given;
import org.testng.annotations.Test;
import io.restassured.matcher.RestAssuredMatchers;

public class XMLSchemaValidation {
	@Test()
	public void xmlSchemaValidation(){
		given()

		.when()
		.get("https://reqres.in/api/users?page=2")

		.then()

.assertThat().body(RestAssuredMatchers.matchesXsdInClasspath("xml_Schema_Body.xsd"));
	}

}
```

47. **What is Serialization?**
    Serialization is the process of converting data structures (typically in a programming language) into a format suitable for transmission over HTTP, such as JSON or XML.

48. **What is De- Serialization?**
    Deserialization is the process of converting data from a serialized format (like JSON or XML) back into a data structure in a programming language.

49. **What is Authentication?**
    Authentication means, it will check whether the user is valid or not. So, whatever details that we are providing like username and password, basically they check provided details are valid or not. Types of authentications are.
    a. **Basic Authentication:** Basic Authentication is used to add authentication credentials like username and password.

```
@Test()
public void testBasicAuthentication() {
	RestAssured.given()
	.auth().basic("username", "password") // Username & Password

		.when()
		.get("URL")

		.then()
```

```
        .log().body()
        .statusCode(200);
}
```

b. **Digest Authentication:** Digest Authentication provides better security than Basic Authentication by avoiding the transmission of raw credentials.

```
@Test()
public void testBasicAuthentication() {
      RestAssured.given()
      .auth().digest("username", "password") // Username &
Password

      .when()
      .get("URL")

      .then()
      .log().body()
      .statusCode(200);
}
```

c. **Bearer Token Authentication:** Bearer Token Authentication uses a token included in the request header.

```
@Test()
public void testBearerTokenAuthentication(){
      String bearerToken = "TokenValue";

      RestAssured.given()
      .headers("Authorization", "Bearer"+bearerToken)

      .when()
      .get("URL")

      .then()
      .log().body()
      .statusCode(200);
}
```

d. **Preemptive Authentication:**

```
@Test()
public void testPreemptiveAuthentication() {
      RestAssured.given()
      .auth()
      .preemptive()
      .basic("username", "password") // Username & Password

      .when()
      .get("URL")
```

```
        .then()
        .log().body()
        .statusCode(200);

    }
```

e. **OAuth 1.0:**

```
    @Test()
    public void testOAuth1Authentication(){
        RestAssured.given()
        .auth()
        .oauth("consumerKey", "consumerSecret", "accessToken",
    "tokenSecret")

        .when()
        .get("URL")

        .then()
        .statusCode(200)
        .log().all();
    }
```

f. **OAuth 2.0:**

```
    @Test()
    public void testOAuth2Authentication(){
        RestAssured.given()
        .auth()
        .oauth2("Access token")

        .when()
        .get("URL")

        .then()
        .statusCode(200)
        .log().all();
    }
```

g. **API Key Authentication:**

```
    @Test()
    public void testAPIKeyAuthentication(){

        RestAssured.given()
        .header("API Key name", "Value")
        //.queryParam("KeyName", "Value")


        .when()
        .get("URL")
```

```
                .then()
                .log().body()
                .statusCode(200);
        }
```

50. What is Authorization?

Authorization means, the user is valid but it will check the access or permissions. So, only authenticated users can have authorization.

51. What is Faker library?

Faker library is a one type of maven dependency called JavaFaker which is used to generate fake data like first name, last name or phone number etc.,

```
@Test()
public void testGenerateDummydata(){
      Faker faker = new Faker();
      String firstName = faker.name().firstName();
      String lastName = faker.name().lastName();
      System.out.println("First name : "+firstName);
      System.out.println("Last name : "+lastName);
}
```

52. What is JSON Object, JSON Array and JSON Element?

JSON Object: JSON Object starts with one curly brace and ends with another curly brace.
JSON Array: JSON Array starts with one square bracket and ends with another square bracket.
JSON Element: JSON Element can be a JSON Object or it can be a JSON Array.

53. What is the difference between HTTP and HTTPS:

HTTP (Hypertext Transfer Protocol): HTTP is a one type protocol for transferring data over the web without encryption. It is not secured.
Ex: http://example.com/api/resource

HTTPS (Hypertext Transfer Protocol Secure): HTTPS is a one type protocol for transferring data over the web that includes encryption. It is secured than the HTTP.
Ex: https://secure.example.com/api/data

54. Important methods in API:

a. Get method is used to get the data.
b. POST method is used to insert the data.
c. PUT method is used to update the data.
d. DELETE method is used to delete the data.

55. Basic Error codes in API:

1. **200 OK:** The request is successful and the server has returned the requested data.
2. **201 Created:** When a new resource has been successfully created on the server.
3. **204 No Content:** When the request is successful, but there is no content to return. Often used for successful deletions.

4. **400 Bad Request:** When the server could not understand the request due to invalid syntax. Then the request needs to be corrected.
5. **401 Unauthorized access:** When the request requires authentication and if provided credentials are missing or incorrect.
6. **403 Forbidden:** When we are trying to access a resource that our user role does not have permission to view.
7. **404 Page not found:** When the requested resource could not be found on the server.
8. **405 Method Not Allowed:** When the HTTP method is used in the request which is not supported by the target resource.
9. **500 Internal Server Error:** When the server crashes or errors in server-side code.
10. **503 Service Unavailable:** When the server is temporarily down for maintenance.

➢ CUCUMBER FRAMEWORK

1. **What are the advantages of Cucumber?**
   a. Cucumber is an open-source tool.
   b. It represents plain text that makes it easier for non-technical users to understand the scenarios.
   c. Automation test cases which are developed by using Cucumber tool are easier to maintain and understand as well.
   d. Easy to integrate with other tools such as Selenium, Cypress etc.

2. **What kind of language is used by the Cucumber?**
   a. Gherkin is the language that is used by the Cucumber.
   b. Gherkin language uses several keywords to describe the behavior of application such as
      - Feature.
      - Scenario.
      - Scenario outline.
      - Example.
      - Background
      - @Given, @When, @ Then and @And.

3. **What is meant by Feature file?**
   a. Feature file is used to define test scenarios in a human readable format.
   b. These files have a '. feature' extension and are written in Gherkin language.
   c. Gherkin language uses several keywords to describe the behavior of application such as
      - Feature.
      - Scenario.
      - Scenario outline.
      - Example.
      - Background
      - @Given, @When, @ Then and @And.

      Note: We need to implement Step Definition methods through the feature file. When we run the feature file, it gives us unimplemented methods.

4. **What is the difference between Scenario and Scenario Outline?**
   Scenario:

Scenario is a way to pass only one set of data.

Scenario Outline:
Scenario outline is a way of parameterization of scenario to pass multiple sets of data. Scenario outline must be followed by the keyword 'examples' which is used to specify the set of values for each parameter.

5. **What are the various keywords that are used in Cucumber for writing a scenario and explain the purpose?**
   Gherkin keywords like Given, When, Then and And Gherkins are used to write a scenario.
   Given: Given is a keyword which is used to specify a precondition for the Scenario.
   When: When is a keyword which is used to specify an operation to be performed.
   Then: Then is a keyword which is used to specify the expected result of a performed action.
   And: And is a keyword which is used to join one or more statements together into a single statement.

6. **What is the use of Background keyword in Cucumber?**
   Background is a keyword which is used to group multiple given statements into a single group. This is generally used, when the same set of given statements are repeated in each scenario of the feature file.

7. **What symbol is used for parameterization in Cucumber?**
   Pipe (|) symbol is used to specify one or more parameters in a feature file.

8. **What is the purpose of Examples keyword in Cucumber?**
   Examples is a keyword which is used to specify set of values for each parameter. Scenario Outline must be followed by the keyword 'Examples'.

9. **What is meant by Step Definition file?**
   a. Each step of the feature file can be mapped to a corresponding method on the step definition file.
   b. While feature files are written in an easily understandable language such as Gherkin and Step definition files are written in programming languages such as Java, .Net and Ruby etc.

10. **What is meant by Test Runner class in Cucumber?**
    Test Runner class is used to provide a link between Feature file and Step Definition file. A Test Runner class is generally an empty class with no class definition.
    EX:

```
@RunWith(Cucumber.class)
@CucumberOptions(features = "Feature file path", glue =
"stepDefinitions")
public class TestRun
{

}
```

11. **What is the purpose of Cucumber Options tag?**
    Cucumber Options tag is used to provide a link between the Feature files and Step Definition files. Each step of the feature file can be mapped to a corresponding method on the step definition file.

Ex:

```
package testRunner;
import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
@RunWith(Cucumber.class)
@CucumberOptions(features = "Feature file path", glue =
"stepDefinitions", dryRun = true, monochrome = true, plugin =
"html: folder name", strict = true, tags= "@smoke")
public class TestRun
{

}
```

12. **What is the starting point of execution for Feature files?**
    Actually, the starting point of execution must be from Test Runner class.

13. **Should any code be written within the Test Runner class?**
    No code should be written under the Test Runner class. It should include the tags @RunWith and @CucumberOptions.

14. **What is the use of features property under Cucumber Options tag?**
    Features property is used to identify the location of the Feature files.
    Ex:
    ```
    features = "Feature file path"
    ```

15. **What is the use of glue property under Cucumber Options tag?**
    Glue property is used to identify the location of the Step Definition files.
    Ex:
    ```
    glue = "stepDefinitions"
    ```

16. **What is the use of monochrome property under Cucumber Options tag?**
    Monochrome property is used to make the console output more readable by removing unnecessary colorization.
    Ex:
    ```
    monochrome = true
    ```

17. **What is the use of plugin property under Cucumber Options tag?**
    Plugin property is used to specify different formatting options for the output reports.
    Ex:
    ```
    plugin = "html: folder name"
    ```

18. **What is the use of dryRun property under Cucumber Options tag?**
    Dry run property is used to check whether the Step Definition files have the all implemented methods or not. In case of any one of the method or steps is missing, then it will show in the console.
    Ex:
    ```
    dryRun = true
    ```

19. **What is the use of strict property under Cucumber Options tag?**

Strict property is used to fail the execution if there are undefined or pending steps.

Ex:

```
strict = true
```

20. **What is the use of tags property under Cucumber Options tag?**

    Tags are labels that we can assign to scenarios by using @ symbol. Tags are used to organize, categorize our scenarios and making it easier to run specific sets of tests.

    Ex:

    ```
    tags= "@sanity"
    tags= "@smoke" etc.,
    ```

    Note: We can mention tags in. feature files. We use these types of tags for Grouping.

21. **Hooks:**

    Hooks are used to execute the code before or after scenarios.

    - @Before.
    - @After.
    - @Before Step.
    - @After Step.

➢ SELF-INTRODUCTION

Hi, my name is (name)
I started my career as a Test executive (experience) years back with (company name). Currently I have been working as a Test Engineer.

My responsibility is to understand Business Requirement Specification (BRS) and high-level scenarios. And converting them into manual test cases and automation scripts.

Execution of test cases and reporting of defects to the developer if there are any and get them fixed. I have experience in manual and automation tests like Smoke, Sanity, Retesting and Regression testing.

As for my project automation we have been using Java with Selenium along with TestNG and Cucumber frameworks. And We are using Data driven framework by using Page Object Model with Page Factory. Using TestNG for Assertions, Grouping, Cross browser and Parallel Testing. Using Maven for build, execution and dependency purpose. Using Git as a repository to store our test scripts. For schedule execution of our test cases on nightly based and day based we are using Jenkins. For reporting purpose, we are using Extent Reports and for defect management we are using Jira.

We are working in Agile environment for systematic approach to develop, test and release the software . We have sprint cycle for 2 weeks. I am a part of 10 members out of which we are 4 testers, 4 developers, 1 product owner and 1 scrum master.

➢ ROLES AND RESPONSIBILITIES

- Designing, developing and executing automation scripts to test software application.
- Collaborating with development teams to understand software requirements and identifying areas for automation.
- Identifying and documenting software bugs and defects.
- Maintaining and updating existing automation scripts and automation framework.
- Creating and executing regression test suites to ensure software stability.
- Participating in code reviews.
- Guiding and mentoring junior testers by providing support and sharing best practices.

➢ DAY TO DAY ACTIVITIES

First thing I do login in my system. I check the active sprint in Jira. There I can see my assigned open tasks. After that I will check my mail if there is any important mail, I need to take action on it. Then we have our daily scrum meeting where we exactly discuss about what we did yesterday and what we will do today. If we have any blockers, Product owner and scrum master help us to resolve those blockers. After that I need to take the pending task and do needed action either creating test cases, Execution, Defect retesting if any and participating in meetings.