TestNG Framework:

Creating Maven Project

1. Create a Maven project in eclipse
2. Select default template as 'maven-archetype-quickstart'
3. GroupID- Organizantion
4. ArtifactID- Application
5. Package- anyName

Create an sample Selenium test involving multiple pages

Update POM.XML with the required dependencies

1. Selenium-java
2. testNG
3. io.github.bonigarcia WebDriverManager
4. com.aventstack extentreports

Page Object Modelling

1. Create a package(framework.pom) in main/java and create page classes
2. Move the sample code to their respective pages
3. Create a local variable for driver
4. Create a constructor as below

```
public Homepage(WebDriver driver)
{
	this.driver=driver;

}
```

5. Create Methods for the actions with respect to the page
6. Create PageFactory for all webElements and By for locators
7. Include below in the constructor to invoke the pageFactory
8. `PageFactory.initElements(driver, this);`
9. `Use @FindBy annotation to store WebElements`
10. `All page file inherits Base Actions class`

Base Actions Class

1. Create a BaseActions class under a new package(framework.commons) that will hold all common selenium actions
2. Create a local driver object and use a constructor in Base Actions with `this.driver=driver;`
3. You will have to add `super(driver);` in the POM class constructor to use the same driver

Generic Functions class

1. Create a GenericFunctions class under a new package(framework.commons) that will hold all generic method like propertyReader(reading values from globalParameters.properties) and getScreenhot methods

```
public String propertyReader(String key) throws IOException
```

```
	{
		Properties prop= new Properties();
		FileInputStream fis= new FileInputStream(new
File("src//main/java//framework//resources//GlobalParameter.properties"));
		prop.load(fis);
		String value=prop.getProperty(key);
		return value;
	}


	public String getScreenshot(String testName, WebDriver driver) throws
IOException
	{
		TakesScreenshot screenshotObject=(TakesScreenshot)driver;
		File src=screenshotObject.getScreenshotAs(OutputType.FILE);
		File file= new File(System.getProperty("user.dir") +"//reports//"
+testName +".png");
		FileUtils.copyFile(src, file);
		return System.getProperty("user.dir") +"//reports//" +testName
+".png";
		}
```

Driver Manager:

1. Create a Driver manager in main/java/framework.commons
2. This class will have methods to invoke browser and to launch the application
    - Keep `WebDriver driver`; as global variable
    - Create 1 method as private to invoke the browser as provided in Global Parameters and return the driver
    - Create @BeforeMethod launchApplication() method which takes in invokeBrowser() method and launches the application
    - Return the driver
    - Create @AfterMethod for driver.close() function

```
GenericFunctions generic= new GenericFunctions();
public static WebDriver driver;
private  WebDriver invokeBrowser() throws IOException
{

	String browserName=generic.propertyReader("Browser");

	if(browserName.equalsIgnoreCase("chrome"))
	{
		ChromeOptions options= new ChromeOptions();
		if(generic.propertyReader("Bypass").equalsIgnoreCase("true"))
		{
			options.setAcceptInsecureCerts(true);

		}
		WebDriverManager.chromedriver().setup();
		driver= new ChromeDriver(options);
	}
	else if(browserName.equalsIgnoreCase("edge"))
	{
		WebDriverManager.edgedriver().setup();
		driver= new EdgeDriver();
	}
```

```
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
		driver.manage().deleteAllCookies();
		return driver;
	}
	@BeforeMethod(alwaysRun = true)
	public  WebDriver launchApplication() throws IOException
	{
		invokeBrowser();
		String environment=generic.propertyReader("Environment");
		if (environment.equalsIgnoreCase("PROD"))
		{
			driver.get(generic.propertyReader("PROD_URL"));
			System.out.println(generic.propertyReader("PROD_URL"));
		}
		else if(environment.equalsIgnoreCase("STAGE"))
		{
			driver.get(generic.propertyReader("STAGE_URL"));
		}
		else if(environment.equalsIgnoreCase("QA"))
		{
			driver.get(generic.propertyReader("QA_URL"));
		}

		else if(environment.equalsIgnoreCase("DEV"))
		{
			driver.get(generic.propertyReader("DEV_URL"));
		}
		return driver;

	}

	@AfterMethod(alwaysRun = true)
	public void teardown()
	{
		driver.close();
	}
```

Test Data:

1. Create a new package under test/java (framework.testData) and declare all Testdata there as static.

```
		public static String firstName="Abilash";
		public static String lastName="Cheruvathur";
		public static String password="Test@1234";
		public static String address="1234 New";
		public static String yearofBirth="2023";
		public static String monthofBirth="October";
		public static String dateofBirth="18";
```

Listeners and Extent Reporting

1. Create a ExtentReporterNG in main/java/framework.commons
2. Add the below configurations

```
public ExtentReports extent;
```

```
    public ExtentReports getReporter() throws IOException
    {
            String path= System.getProperty("user.dir")+
propertyReader("ReportPath");
            ExtentSparkReporter reporter= new ExtentSparkReporter(path);
            reporter.config().setReportName("TestNG Hybrid Framework");
            reporter.config().setDocumentTitle("Test Automation Results");


            extent= new ExtentReports();
            extent.attachReporter(reporter);
            extent.setSystemInfo("Author", "Abilash");
            extent.setSystemInfo("Application", "XXXX");
            extent.setSystemInfo("Framework", "TestNG Hybrid Framework v1.0");

            return extent;
    }
```

3. Add TestNGlistner in test/java (framework.listeners) and implement ITestListeners
4. Add the following for OnTestStart, TestSuccess and TestFailure

```
public void onTestStart(ITestResult result) {

            try {
                  extent=extentReports.getReporter();
                  test=extent.createTest(result.getMethod().getMethodName());
            } catch (IOException e) {
                  e.printStackTrace();
            }
    }

    @Override
    public void onTestSuccess(ITestResult result) {

            test.log(Status.PASS, "Test case Passed");
    }

    @Override
    public void onTestFailure(ITestResult result) {
            test.fail(result.getThrowable());

            try {
                  driver=
(WebDriver)result.getTestClass().getRealClass().getField("driver").get(result.
getInstance());
            } catch (Exception e) {
                  e.printStackTrace();
            }

            String filePath=null;

            try {
                  filePath= getScreenshot(result.getMethod().getMethodName(),
driver);
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();

            }
```

```
            test.addScreenCaptureFromPath(filePath,
    result.getMethod().getMethodName());

        }
```

5. Add extent.flush() under onFinish() method

TestNG XML:

1. Convert the project to TestNG and create the testNG XML suite
2. Add the listerns there

```
<listeners>
        <listener class-name="framework.listeners.TestNGListerners">
        </listener>
</listeners>
```

3. Add profiles to POM.xml if we have multiple testing XML suites to be run.

```
        <profiles>
              <profile>
                    <id>SanityTest</id>
                    <build>
                          <plugins>
                                <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.21.0</version>
        <configuration>
        <suiteXmlFiles>

        <suiteXmlFile>Sanity.xml</suiteXmlFile>
        </suiteXmlFiles>
        </configuration>
                                </plugin>
                          </plugins>
                    </build>
              </profile>
```