### **Project Title: Web Application Test Automation with Selenium and TestNG**

**Objective**:  
Build an end-to-end automation framework for testing a sample web application using Java, Selenium WebDriver, and TestNG.

### **Project Deliverables:**

1. **Project Source Code**
2. **Documentation**:
   * Requirement Specification
   * Test Plan
   * Test Cases Document
   * Setup Guide
   * Automation Report Samples

### **Project Outline:**

#### **1. Project Setup**

* **Tools/Technologies**:
  + Java (Programming Language)
  + Selenium WebDriver (Automation Tool)
  + TestNG (Testing Framework)
  + Maven (Build Tool)
  + Apache POI (For Excel Integration if needed)
  + ExtentReports (For Custom Reports)
* **Environment Setup**:
  + Install Java Development Kit (JDK).
  + Install IntelliJ IDEA/Eclipse as the IDE.
  + Add required dependencies (Selenium, TestNG, etc.) to the pom.xml file for Maven.

#### **2. Web Application for Testing**

Use an open-source demo web application for automation testing. Examples:

* OrangeHRM Demo
* The Internet

#### **3. Project Workflow**

* **Module 1**: Login Page Tests  
  Automate test cases for login functionality, including:
  + Valid Login
  + Invalid Login (Wrong username/password)
  + Empty Fields Submission
* **Module 2**: Form Submission Tests  
  Automate test cases for filling out and submitting a form, including:
  + Validation of mandatory fields.
  + Successful form submission.
* **Module 3**: Dashboard/Navigation Tests  
  Validate that users are redirected to the correct pages upon clicking menu options.
* **Module 4**: Data-Driven Testing  
  Use Excel files to drive test data for login and form tests.
* **Module 5**: Reports and Logs  
  Generate detailed test execution reports using ExtentReports and maintain logs for debugging.

#### **4. Test Automation Framework**

Build a modular framework using the **Page Object Model (POM)**:

* **Base Class**: Manages WebDriver initialization and teardown.
* **Page Classes**: Represent different pages in the application, containing locators and methods for interactions.
* **Test Classes**: Contain the actual test methods.
* **Utilities**:
  + DataProvider for test data.
  + Logger for logging (using Log4j or SLF4J).
  + Reporter for generating HTML reports.

### **Documents:**

1. **Requirement Specification**
   * Define the application functionality to test and test scenarios.
2. **Test Plan**
   * Scope, objectives, tools, environment, schedule, deliverables, and risks.
3. **Test Case Document**
   * Test case ID, description, preconditions, test steps, expected results, and actual results.
4. **Setup Guide**
   * Step-by-step guide to set up the automation project in a local system.
5. **Sample Test Reports**
   * Include ExtentReports/Allure Report screenshots as examples.

### **Sample Directory Structure:**

automation-testing-project/

│

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ ├── base/ # Base class for WebDriver setup

│ │ │ ├── pages/ # Page Object classes

│ │ │ ├── utilities/ # Utility classes (e.g., DataProvider, Logger)

│ │ └── resources/

│ │ └── test-data/ # Excel files for data-driven testing

│ │

│ └── test/

│ ├── java/

│ │ ├── tests/ # Test classes

│ ├── resources/

│ └── reports/ # Test execution reports

│

├── pom.xml # Maven configuration file

├── README.md # Setup guide

└── .gitignore

### **Example Test Case (Login Test in TestNG):**

package tests;

import base.BaseTest;

import org.testng.Assert;

import org.testng.annotations.Test;

import pages.LoginPage;

public class LoginTest extends BaseTest {

@Test

public void testValidLogin() {

LoginPage loginPage = new LoginPage(driver);

loginPage.enterUsername("Admin");

loginPage.enterPassword("admin123");

loginPage.clickLoginButton();

Assert.assertTrue(loginPage.isDashboardDisplayed(), "Dashboard is not displayed");

}

@Test

public void testInvalidLogin() {

LoginPage loginPage = new LoginPage(driver);

loginPage.enterUsername("wrongUser");

loginPage.enterPassword("wrongPass");

loginPage.clickLoginButton();

Assert.assertTrue(loginPage.isErrorMessageDisplayed(), "Error message is not displayed");

}

}