**Team Guide: Test Automation Framework**

**1. Project Overview**

This is a test automation framework designed for functional and API testing. It integrates tools such as:

* Selenium WebDriver - UI automation
* TestNG - Test execution and reporting
* Rest Assured - API testing
* Postman - API request validation
* JMeter - Performance testing
* Maven - Dependency management and test execution

**The framework ensures scalability, maintainability, and reusability while following best practices.**

**2. Folder & File Structure**

| **Folder/File** | **Purpose** |
| --- | --- |
| **.git** | **Git repository tracking** |
| **.gitignore** | **Files to ignore in Git** |
| **automation-framework/** | **Main framework directory** |
| **automation-framework/logs/** | **Execution logs for debugging** |
| **automation-framework/postman/** | **Postman collections for API testing** |
| **automation-framework/reports/** | **Test reports (e.g., TestNG, Allure)** |
| **automation-framework/src/** | **Contains test scripts and framework code** |
| **automation-framework/target/** | **Compiled test results and build artifacts** |
| **automation-framework/pom.xml** | **Maven project configuration** |
| **config/** | **Configuration files (e.g., test data, environment variables)** |
| **docs/** | **Documentation (test cases, guidelines, API references)** |
| **meetings/** | **Meeting notes and discussions** |
| **README.md** | **Overview and setup instructions** |

**3. Setup & Installation**

**Prerequisites**

Ensure you have installed:

* Java 11+ (java -version)
* Maven (mvn -version)
* Node.js (if UI tests use WebDriver Manager)
* Postman & JMeter (for API & performance testing)
* IDE (IntelliJ IDEA or VS Code)

**Setup Steps**

1. Clone the Repository
2. git clone <repo-url>
3. cd automation-framework
4. Install Dependencies
5. mvn clean install
6. Run a Sample Test
7. mvn test

**4. How to Run Tests**

UI Tests (Selenium + TestNG)

mvn test -Dtest=UITestClass

API Tests (Rest Assured + Postman)

mvn test -Dtest=APITestClass

Performance Tests (JMeter)

jmeter -n -t tests.jmx -l results.jtl

Run All Tests

mvn clean test

**5. How to Write Tests**

**UI Test Example (Selenium + TestNG)**

public class LoginTest extends BaseTest {

@Test

public void testLogin() {

driver.get("https://example.com");

driver.findElement(By.id("username")).sendKeys("user");

driver.findElement(By.id("password")).sendKeys("pass");

driver.findElement(By.id("loginButton")).click();

Assert.assertTrue(driver.findElement(By.id("dashboard")).isDisplayed());

}

}

**API Test Example (Rest Assured)**

public class APITest {

@Test

public void getUsers() {

given()

.baseUri("https://api.example.com")

.when()

.get("/users")

.then()

.statusCode(200)

.body("size()", greaterThan(0));

}

}

**6. Debugging & Logs**

* All logs are saved in the logs/ folder.
* Failed test screenshots are captured in reports/.
* Use mvn test -Ddebug=true for detailed logs.

**7. Collaboration Guidelines**

**Git Workflow**

1. Create a new branch
2. git checkout -b feature-branch
3. Commit changes
4. git add .
5. git commit -m "Implemented new test"
6. Push & create a PR
7. git push origin feature-branch
8. Code review and merge

**Code Review Checklist**

**✅** Follow naming conventions  
✅ Avoid hardcoded values (use config files)  
✅ Handle waits properly in UI tests  
✅ Use assertions correctly  
✅ Ensure tests are independent

**8. Extending the Framework**

**Adding a New UI Test**

1. Create a new test class in src/test/java/ui/.
2. Extend BaseTest.
3. Implement test methods with @Test.

**Adding a New API Endpoint Test**

1. Add a test in src/test/java/api/.
2. Use Rest Assured methods (given().when().then()).
3. Validate response using assertions.

**Adding a New Report**

* Reports are stored in reports/.
* Customize TestNG reports using testng.xml.
* Use Allure Reports (mvn allure:serve).

**✅ Summary**

* Folder structure explained 📂
* Setup & installation 🛠
* Running tests 🚀
* Writing & debugging tests 🔍
* Collaboration & Git workflow 🔄
* How to extend the framework 📌

**📌 Next Steps**

* Ensure all team members follow this guide.
* Regularly update documentation with new features.
* Optimize tests for better performance and maintainability.

**Let’s build an efficient and robust test automation framework! 🚀**