**🔧 src/main/java/com.company.framework/**

**Purpose**: Contains **core framework code** used across all tests. This keeps test logic separate from reusable components like browser setup, page interactions, and utilities.

**📁 base/**

**Purpose**: Centralizes WebDriver setup, teardown, and browser configurations.

**Why?**  
Without a shared base, you'd have to duplicate browser launch logic in every test. This violates DRY (Don't Repeat Yourself).

**Example Use Case**:  
BaseTest.java may:

* Read browser type from config.properties
* Initialize WebDriver accordingly (e.g., ChromeDriver, EdgeDriver)
* Handle setup (@BeforeSuite) and teardown (@AfterSuite) tasks

*“In our framework, BaseTest ensures consistent browser setup across tests, which improves stability and reduces duplication.”*

**📁 pages/**

**Purpose**: Implements **Page Object Model (POM)**, a design pattern that abstracts web elements and actions on each page.

**Why?**  
If the UI changes (e.g., button ID changes), only one class (LoginPage.java) needs updating—not every test case.

**Example Use Case**:  
LoginPage.java might contain:

@FindBy(id = "username")

private WebElement usernameField;

public void enterUsername(String user) {

usernameField.sendKeys(user);

}

*“POM increases maintainability by separating locators and actions from test logic.”*

**📁 utils/**

**Purpose**: Stores **helper or utility methods** that are reused across multiple classes.

**Why?**  
Encapsulating common actions (like taking screenshots or waiting for elements) avoids repetition and makes debugging easier.

**Example Use Case**:  
BrowserUtils.java might include:

* waitForVisibility(WebElement element, int timeout)
* captureScreenshot(String fileName)
* scrollIntoView(WebElement element)

*“Utilities allow abstraction of repetitive logic like wait conditions, improving code readability.”*

**🧪 src/test/java/com.company.tests/**

**Purpose**: Contains all **test logic**, including step definitions, test runners, and Cucumber hooks.

**📁 stepdefinitions/**

**Purpose**: Binds **Gherkin steps** in .feature files to actual Selenium actions in Java.

**Why?**  
This layer keeps BDD feature files human-readable while ensuring those steps are executable via code.

**Example Use Case**:  
From Login.feature:

Given user is on login page

In LoginSteps.java:

@Given("user is on login page")

public void userIsOnLoginPage() {

loginPage.open();

}

*“Step Definitions link the business-readable feature steps to real browser actions using Selenium.”*

**📁 hooks/**

**Purpose**: Contains **Cucumber lifecycle hooks** for pre- and post-scenario tasks.

**Why?**  
Common setup/cleanup logic (like opening the browser, deleting cookies, or taking failure screenshots) shouldn't clutter test logic.

**Example Use Case**:  
TestHooks.java may contain:

@Before

public void setUp() {

Driver.initialize(); // Starts browser

}

@After

public void tearDown(Scenario scenario) {

if (scenario.isFailed()) {

ScreenshotUtils.capture(driver, scenario.getName());

}

Driver.quit();

}

*“Hooks improve consistency and enable things like automatic screenshot capture for failed steps.”*

**📁 runners/**

**Purpose**: Specifies how and what to run using **CucumberOptions**.

**Why?**  
Runner classes organize execution, apply tags, generate reports, and connect steps and features.

**Example Use Case**:  
TestRunner.java might include:

@CucumberOptions(

features = "src/test/resources/features",

glue = "com.company.tests.stepdefinitions",

plugin = {"pretty", "html:target/cucumber-reports"}

)

*“The runner configures what scenarios to execute and how to report the results.”*

**📂 src/test/resources/**

**Purpose**: Contains all **non-Java resources** needed for testing (like feature files, configs, and data).

**📁 features/**

**Purpose**: Stores .feature files written in **Gherkin**.

**Why?**  
Gherkin enables BDD by letting non-technical stakeholders read and contribute to test cases.

**Example**:

Feature: Login functionality

Scenario: Valid user login

Given user is on login page

When user enters valid credentials

Then user should see homepage

*“Feature files describe business logic in a readable format, aligning testers and product teams.”*

**📁 config/**

**Purpose**: Holds external configuration files like config.properties.

**Why?**  
Decouples test configuration from code, allowing easier environment switching (QA, UAT, PROD).

**Example Use Case**:  
config.properties might include:

browser=chrome

url=https://myapp.test

implicitWait=10

*“Externalizing configs allows changing environments without modifying code.”*

**📁 data/**

**Purpose**: Stores **test data files** in formats like CSV, JSON, Excel.

**Why?**  
Supports **data-driven testing** by feeding dynamic data into test scenarios.

**Example Use Case**:

* testData.json contains test users
* users.csv stores login credentials for multiple roles

*“External data promotes scalability by separating data from test scripts.”*

**📁 target/**

**Purpose**: Holds **auto-generated outputs** from test execution.

**📁 cucumber-reports/**

**Purpose**: Stores **HTML/JSON reports** created by Cucumber.

**Why?**  
Provides a detailed view of pass/fail scenarios, step duration, and attachments (like screenshots).

*“Reports provide traceability and test result visibility to all stakeholders.”*

**📁 screenshots/**

**Purpose**: Contains screenshots taken during test failures.

**Why?**  
Helps debug issues by capturing browser state at the point of failure.

*“Screenshots speed up root cause analysis when scenarios fail.”*

**📄 pom.xml**

**Purpose**: Manages **Maven dependencies and plugins**.

**Why?**  
Automates dependency management and builds for cross-environment compatibility.

**Includes**:

* Selenium, Cucumber, WebDriverManager, TestNG
* Plugins like maven-surefire-plugin, cucumber-reporting

*“Maven makes the project portable and CI/CD-ready.”*

**📄 testng.xml**

**Purpose**: Defines **TestNG suite configuration** (if integrated).

**Why?**  
Supports **parallel test execution**, grouping, and filtering tests.

*“TestNG integration adds execution control and grouping capabilities.”*

**📄 cucumber.properties**

**Purpose**: Fine-tunes **Cucumber-specific behavior**.

**Why?**  
Configures things like:

* Report formats
* Timeouts
* Step definition matching

*“This file customizes Cucumber execution beyond annotations.”*