Safety National Job Details

Description

At Safety National, we don’t just offer jobs – we build careers with purpose! Since 1942, we’ve been an industry leader, valuing integrity, teamwork, and stability while providing competitive rewards, top-tier benefits, career growth opportunities, and flexible work options that promote balance. With tuition reimbursement, wellness perks, and a strong community impact, we invest in your success—both personally and professionally. Ready to grow with us? Apply today!

Follow this link to view all of our available careers and apply: <https://www.safetynational.com/careers-page/>

**This opportunity is in the Information Technology department**  
As the company’s largest department, I.T. is responsible for supporting both ongoing operations and our next innovative project. Information technology is prioritized, allowing us to partner with colleagues to securely enable and support new capabilities every day. Utilizing a diverse set of technologies, like Guidewire, Salesforce, Java, C#/.NET, Mule, and Azure, the available career opportunities continue to grow rapidly. Our department consists of agile teams where individual contributions are recognized, and career advancement is nurtured.

**Role Description:**

Are you passionate about using automation to drive quality and efficiency in software testing? As a Senior Test Automation Engineer at Safety National, you’ll play a key role in the design, implementation, and maintenance of complex automated testing frameworks. You will combine strong manual testing capabilities with advanced programming knowledge to develop automated regression tests and reusable components. Working closely with functional test leads and manual testers, you’ll help shape testing strategies, guide junior automation engineers, and ensure thorough documentation of testing processes. You’ll also be involved in configuring Jenkins jobs, conducting peer code reviews, and participating in test automation tool evaluations. If you’re a self-motivated, detail-oriented professional with a strong grasp of Agile, DevOps, and modern programming languages, this role offers the opportunity to make a significant impact within a collaborative and forward-thinking QA team.

**Qualifications:**

**Education:**  
A bachelor’s degree in an I.T.-related field is preferred.

**Required Qualifications:**

* Must be presently authorized to work in the U.S. without a requirement for work authorization sponsorship by our company for this position now or in the future.
* This position allows for a hybrid work schedule which includes a minimum of 3 days per week in our St. Louis Corporate Office, located at 1832 Schuetz Rd. Saint Louis, MO. All employees are required to submit work from home requests and follow our Work From Home policy, which will be provided to all candidates selected to interview.
* 8 or more years of working experience in Information Services or a related field.
* Proven working experience in software development with a track record of making a difference.
* Intermediate-level programming skills in at least one of the following languages: Java, Python, C#, or JavaScript.
* Strong knowledge of automated testing software and frameworks, especially Selenium.
* Proficient with Agile, DevOps (CI/CT/CD), and associated technologies.
* Ability to prioritize and meet deadlines in a fast-paced team environment.
* Motivated, self-directed, results-oriented, and customer-focused.
* Strong attention to detail and a passion for quality.

**Preferred Qualifications:**

* Possession of one or more approved certifications within the QA/test automation area.
* Experience in Property & Casualty insurance.
* Familiarity with software development life cycles (Agile, Waterfall, Iterative).
* Knowledge of TargetProcess software.
* Familiarity with SQL, SSRS, or other reporting tools.
* Understanding of UML, process flows, use cases, and SDLC documentation practices.

Protect the confidentiality, integrity and availability of information and technology assets against unauthorized disclosure, destruction and/or alteration, in accordance with Safety National policies, standards, and procedures.

Safety National is a leading specialty insurance and reinsurance provider. Our culture is built upon relationships, which allow us to demonstrate our expertise gained through our rich 80-year history. As a wholly-owned subsidiary of Tokio Marine, Inc., we appreciate the benefits and support provided by our affiliation with one of the top 10 insurance companies in the world.

**Total Rewards That Put Employees First**

In our vision to be First with Co-Workers, compensation that includes base salary, holiday bonus, and incentive awards is only a small portion of the comprehensive total rewards package we offer. Our total rewards approach recognizes and rewards the time, talents, efforts, and results of our valued employees. Highlights of our exceptional benefits include generous health, dental, and vision coverage, health savings accounts, a 401(k)-retirement savings match and an annual profit sharing contribution. We proudly offer family forming benefits for adoption, fertility, and surrogacy, generous paid time off and paid holidays, paid parental and caregiver leave, a hybrid work environment, and company-paid life insurance and disability. To support employees in their career journeys, we provide professional growth and development opportunities in addition to employee recognition and well-being programs. Apply today to learn more.

Safety National is committed to fair, transparent pay and we strive to provide competitive, market-based compensation. In our vision to be First with Co-Workers, compensation is only one piece of the comprehensive total rewards package we offer. The target base salary range for this position is $86,000 to $112,000 . Compensation for the successful candidate will consider the candidate’s particular combination of knowledge, skills, competencies, experience and geographic location.

#LI-Hybrid

Equifax job details

A screenshot of a computer

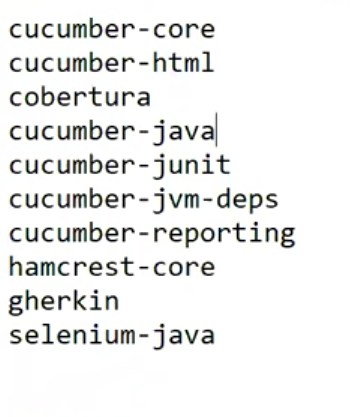
AI-generated content may be incorrect.

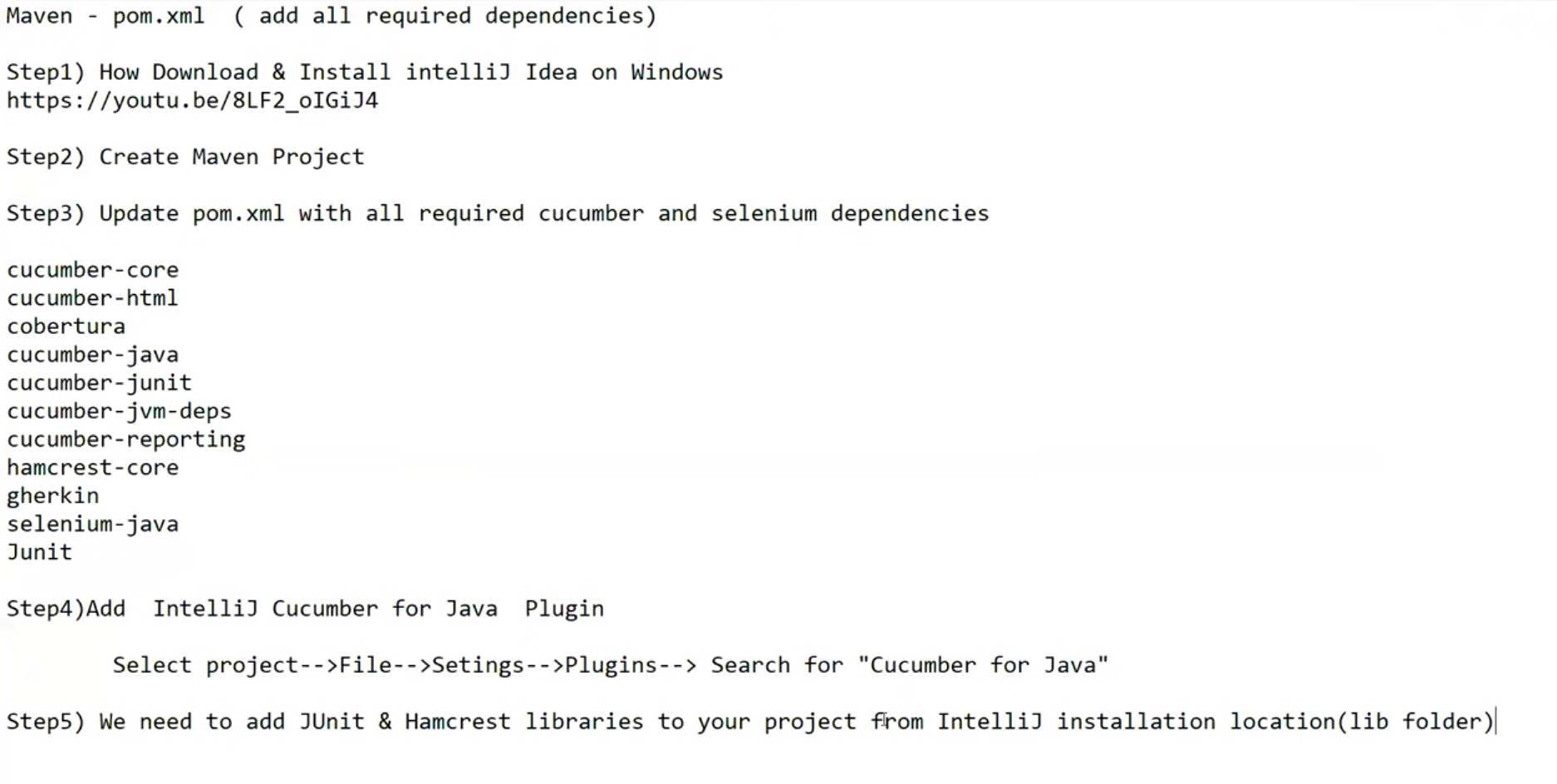
A white background with black text

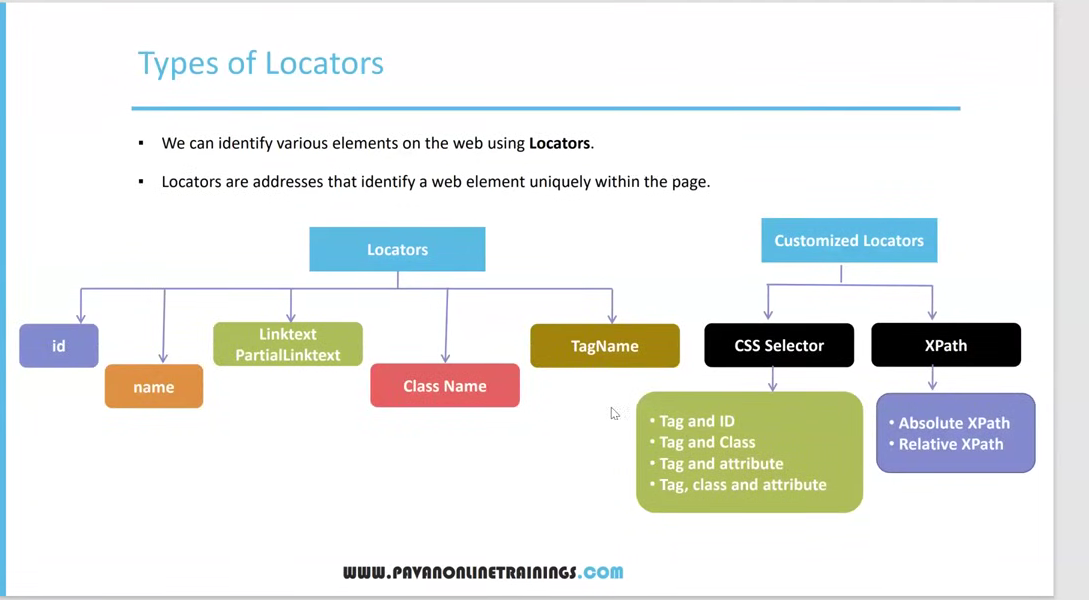
AI-generated content may be incorrect.

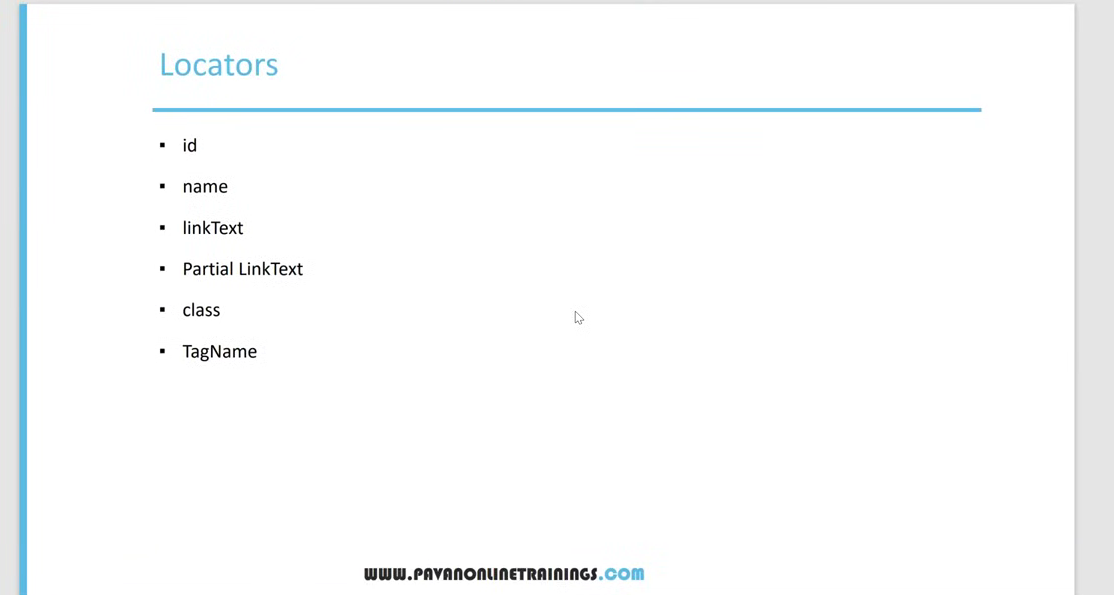
SDET QA youtuber Notes

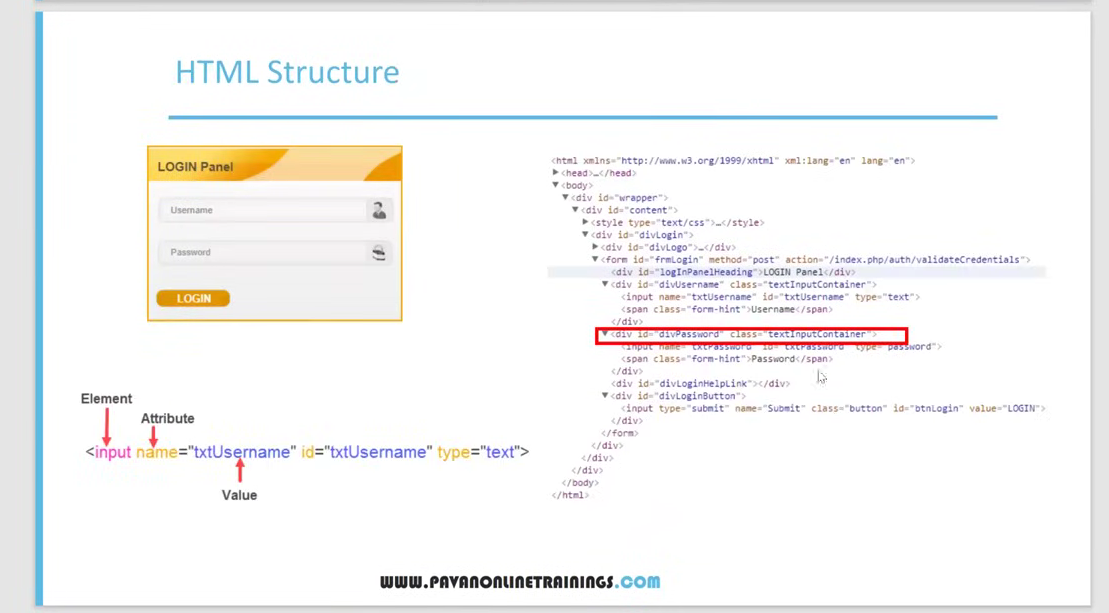
Cucumber dependencies



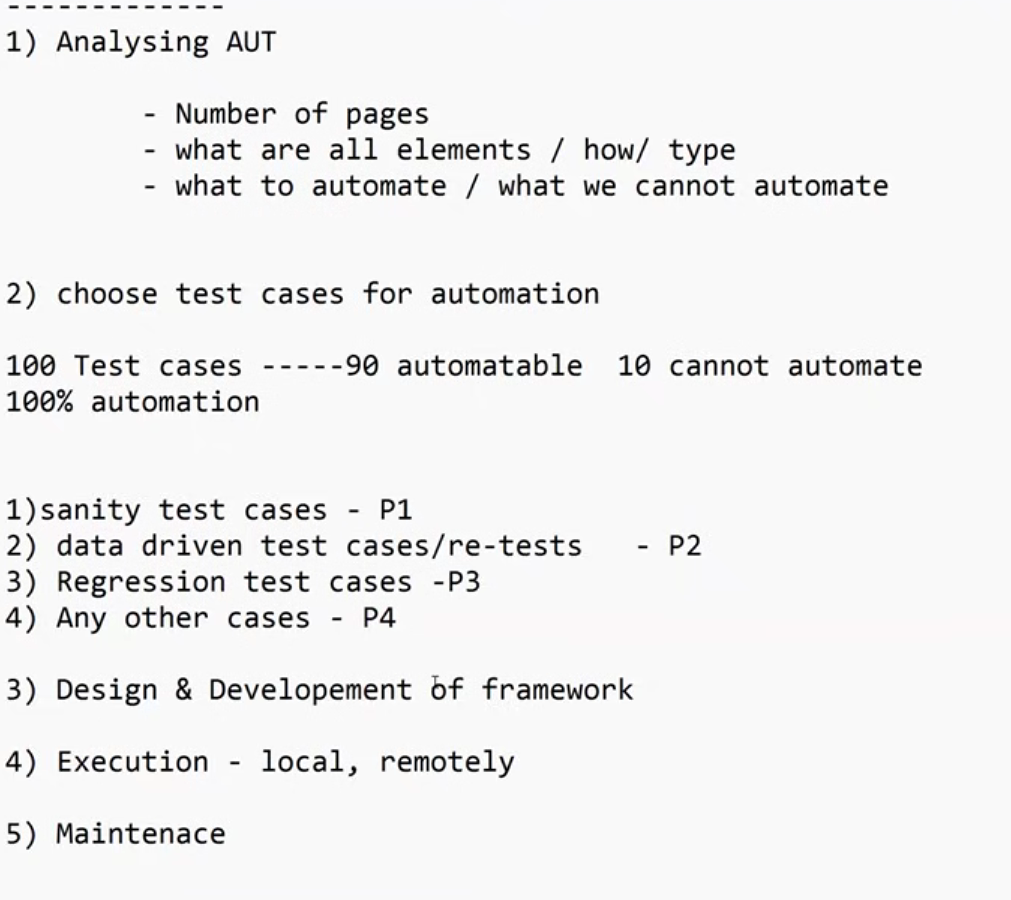


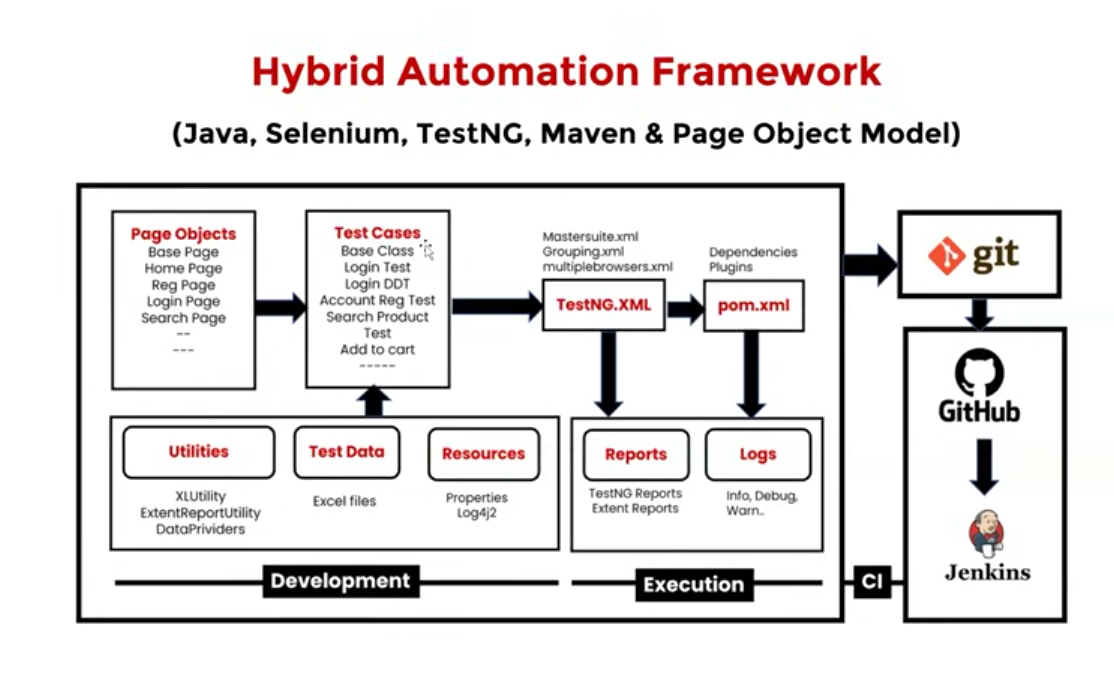












A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Dependencies for REST API

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Below is a detailed, structured explanation of a complete **Test Automation Framework** built with **Selenium, Java, TestNG, Gherkin, Maven, and Cucumber**. This covers architecture, components, best practices, and how everything works together.

**✅ 1. Overview of the Framework**

A modern automation framework using **Selenium + Java + TestNG + Cucumber (Gherkin) + Maven** typically follows a **Hybrid BDD Framework** architecture.

It combines:

* **Selenium WebDriver** → UI interactions
* **Java** → Programming language
* **Cucumber (Gherkin)** → BDD-style test scenarios
* **TestNG** → Test execution, configuration, parallel run
* **Maven** → Build management, dependency handling
* **POM (Page Object Model)** → Maintainable UI interaction layer
* **Reporting (Allure/Extent)** → Rich test reporting

**✅ 2. Project Folder Structure**

project-root

│

├── src/main/java

│ ├── base

│ │ └── BaseTest.java

│ ├── pages

│ │ └── LoginPage.java

│ ├── utils

│ │ ├── ConfigReader.java

│ │ ├── DriverFactory.java

│ │ └── TestDataUtil.java

│ └── hooks

│ └── Hooks.java

│

├── src/test/java

│ ├── stepdefinitions

│ │ └── LoginSteps.java

│ ├── runners

│ │ └── TestRunner.java

│ └── tests

│ └── ParallelTestNG.xml (optional)

│

├── src/test/resources

│ ├── features

│ │ └── login.feature

│ └── config.properties

│

├── pom.xml

└── README.md

**✅ 3. Core Components Explained**

**3.1 Page Object Model (POM)**

Each web page is represented as a Java class containing:

* Locators
* Page actions (methods)
* Reusable functions

Example:

public class LoginPage {

private WebDriver driver;

@FindBy(id="username") private WebElement username;

@FindBy(id="password") private WebElement password;

@FindBy(id="loginBtn") private WebElement loginBtn;

public LoginPage(WebDriver driver) {

this.driver = driver;

PageFactory.initElements(driver, this);

}

public void enterUsername(String user) { username.sendKeys(user); }

public void enterPassword(String pass) { password.sendKeys(pass); }

public void clickLogin() { loginBtn.click(); }

}

**3.2 Step Definitions (Cucumber + Selenium)**

Gherkin steps are connected to Java code.

public class LoginSteps {

LoginPage loginPage = new LoginPage(DriverFactory.getDriver());

@Given("user is on login page")

public void user\_is\_on\_login\_page() {

DriverFactory.getDriver().get(ConfigReader.get("url"));

}

@When("user enters {string} and {string}")

public void enter\_credentials(String user, String pass) {

loginPage.enterUsername(user);

loginPage.enterPassword(pass);

}

@And("clicks on login button")

public void click\_login() {

loginPage.clickLogin();

}

@Then("user should be logged in")

public void verify\_login() {

Assert.assertTrue(/\* some validation \*/);

}

}

**3.3 Gherkin Feature File (BDD)**

Example:

Feature: Login functionality

Scenario: Successful login

Given user is on login page

When user enters "admin" and "admin123"

And clicks on login button

Then user should be logged in

This makes tests readable for non-technical stakeholders.

**3.4 DriverFactory (Cross-Browser & Parallel Execution)**

public class DriverFactory {

private static ThreadLocal<WebDriver> driver = new ThreadLocal<>();

public static WebDriver getDriver() {

return driver.get();

}

public static void initDriver(String browser) {

if (browser.equalsIgnoreCase("chrome")) {

driver.set(new ChromeDriver());

}

// More browsers...

getDriver().manage().window().maximize();

}

public static void quitDriver() {

getDriver().quit();

driver.remove();

}

}

**3.5 Hooks (Cucumber Lifecycle)**

Executed before & after each scenario.

public class Hooks {

@Before

public void start() {

DriverFactory.initDriver(ConfigReader.get("browser"));

}

@After

public void tearDown(Scenario scenario) {

if(scenario.isFailed()) {

final byte[] screenshot =

((TakesScreenshot) DriverFactory.getDriver())

.getScreenshotAs(OutputType.BYTES);

scenario.attach(screenshot, "image/png", "failure\_screenshot");

}

DriverFactory.quitDriver();

}

}

**3.6 TestNG Runner for Cucumber**

@CucumberOptions(

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

glue = {"stepdefinitions", "hooks"},

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

)

public class TestRunner extends AbstractTestNGCucumberTests {

}

This integrates TestNG and Cucumber, allowing:

* Parallel test execution
* Test grouping
* Retry logic
* Test listeners

**3.7 Config File (Global Settings)**

config.properties:

browser=chrome

url=https://example.com

timeout=10

Used by config reader:

public class ConfigReader {

private static Properties prop;

public static String get(String key) {

if (prop == null) {

try {

FileInputStream fis = new FileInputStream("src/test/resources/config.properties");

prop = new Properties();

prop.load(fis);

} catch (Exception e) {

e.printStackTrace();

}

}

return prop.getProperty(key);

}

}

**3.8 Maven (Build & Dependency Management)**

Key dependencies inside pom.xml:

<dependencies>

<!-- Selenium -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.21.0</version>

</dependency>

<!-- Cucumber -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>7.15.0</version>

</dependency>

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>7.15.0</version>

</dependency>

<!-- TestNG -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.10.2</version>

</dependency>

</dependencies>

**✅ 4. How Everything Works Together**

| **Component** | **Role** |
| --- | --- |
| **Gherkin** | Defines test scenarios in natural language |
| **Step Definitions** | Converts Gherkin steps into Java Selenium actions |
| **POM (Pages)** | Encapsulates UI locators & actions |
| **Hooks** | Setup/teardown activities |
| **DriverFactory** | Handles driver initialization & parallel run |
| **TestNG Runner** | Executes Cucumber tests and manages suites |
| **Maven** | Manages dependencies, packaging, CI pipeline |
| **Reports** | Cucumber/Allure/Extent create HTML execution reports |

Flow:

1. User triggers **Maven** or **TestNG** execution.
2. Cucumber reads **feature files**.
3. Corresponding **step definition methods** are invoked.
4. Steps interact with **page objects** using Selenium actions.
5. Hooks manage browser lifecycle.
6. Reports are generated.

**✅ 5. Additional Best Practices**

**📌 Enhance Maintainability**

* Use **Page Factory** or **Selenium By & Actions**
* Create **Utility classes** (waits, screenshots, data readers)
* Implement **centralized exception handling**

**📌 Performance & Parallel Execution**

* Use TestNG parallel mode
* Use ThreadLocal WebDriver
* Run using Selenium Grid or cloud providers (BrowserStack/Sauce)

**📌 Reporting**

Add Allure:

allure serve target/allure-results

**📌 CI/CD Integration**

Use:

* Jenkins
* GitHub Actions
* GitLab CI

Run tests on every commit using:

mvn clean test

**🎉 Conclusion**

This framework design gives you:

✔ Maintainable & scalable architecture  
✔ Human-readable tests (BDD)  
✔ Cross-browser support  
✔ Parallel execution  
✔ Rich reporting  
✔ Easy integration with CI/CD

If you want, I can also provide:

* A downloadable **template project structure**
* Complete **pom.xml**
* Sample **feature + step + page** code
* CI/CD pipeline YAML
* Best practices guide

Would you like that?

selenium-cucumber-framework

│

├── pom.xml

│

├── src

│ ├── main

│ │ └── java

│ │ ├── base

│ │ │ └── BaseTest.java

│ │ ├── factory

│ │ │ └── DriverFactory.java

│ │ ├── pages

│ │ │ └── LoginPage.java

│ │ ├── utils

│ │ │ ├── ConfigReader.java

│ │ │ └── TestUtil.java

│ │ └── hooks

│ │ └── Hooks.java

│ │

│ └── test

│ ├── java

│ │ ├── stepdefinitions

│ │ │ └── LoginSteps.java

│ │ ├── runners

│ │ │ └── TestRunner.java

│ │ └── TestNG.xml

│ │

│ └── resources

│ ├── features

│ │ └── login.feature

│ └── config.properties

│

└── README.md