

1. Executive Summary

TestNG Framework

- **Approach:** Traditional Java-based testing
- **Annotations:** @Test, @BeforeMethod, @AfterMethod
- **Data-Driven:** @DataProvider for parameterization
- **Grouping:** groups = {smoke, regression}
- **Parallel:** Built-in via testng.xml
- **Reports:** Allure with @Epic, @Feature, @Story

Cucumber BDD Framework

- **Approach:** Behavior-Driven Development
- **Syntax:** Gherkin (Given/When/Then)
- **Data-Driven:** Scenario Outline + Examples
- **Grouping:** @tags for filtering
- **Parallel:** Via Maven Surefire plugin
- **Reports:** Allure Cucumber plugin

2. Architecture Comparison

TestNG Project Structure

```
selenium-testng-framework/  
├── src/  
│   ├── main/java/  
│   │   ├── pages/  
│   │   │   ├── BasePage.java  
│   │   │   ├── LoginPage.java  
│   │   │   ├── ProductsPage.java  
│   │   │   ├── CartPage.java  
│   │   │   └── CheckoutPage.java  
│   │   ├── factory/  
│   │   │   └── DriverFactory.java  
│   │   └── utils/  
│   │       └── ConfigReader.java  
│   └── test/java/  
│       ├── tests/  
│       │   ├── BaseTest.java  
│       │   ├── LoginTest.java  
│       │   ├── ProductsTest.java  
│       │   ├── CartTest.java  
│       │   └── CheckoutTest.java  
│       └── listeners/  
│           └── AllureListener.java
```

```
├─ testng.xml
├─ pom.xml
└─ Jenkinsfile
```

Cucumber Project Structure

```
cucumber-bdd-framework/
├─ src/
│   ├── main/java/
│   │   ├── pages/
│   │   │   ├── BasePage.java
│   │   │   ├── LoginPage.java
│   │   │   ├── ProductsPage.java
│   │   │   ├── CartPage.java
│   │   │   └── CheckoutPage.java
│   │   ├── factory/
│   │   │   └── DriverFactory.java
│   │   └── utils/
│   │       └── ConfigReader.java
│   └─ test/
│       ├── java/
│       │   ├── stepdefinitions/
│       │   │   ├── LoginSteps.java
│       │   │   ├── ProductSteps.java
│       │   │   ├── CartSteps.java
│       │   │   └── CheckoutSteps.java
│       │   ├── hooks/
│       │   │   └── Hooks.java
│       │   ├── runner/
│       │   │   └── TestRunner.java
│       └─ resources/features/
│           ├── login.feature
│           ├── products.feature
│           ├── cart.feature
│           └── checkout.feature
├─ pom.xml
└─ Jenkinsfile
```

3. Login Tests

6 test cases in this category

#1 Valid User Login

Critical

Smoke

Regression

Description

Verify that a valid user can successfully log in to the SauceDemo application using correct credentials and is redirected to the Products page.

TestNG Implementation

src/test/java/tests/LoginTest.java

```

@Story("Valid Login")
@Severity(SeverityLevel.CRITICAL)
@Test(groups = {"smoke", "regression"})
@Description("Verify valid user can login successfully")
public void testValidLogin() {
    logger.info("Starting valid login test");

    loginPage.enterUsername(config.getProperty("valid.username"));
    loginPage.enterPassword(config.getProperty("valid.password"));
    loginPage.clickLoginButton();

    Assert.assertEquals(productsPage.getPageTitle(), "Products",
        "User should be redirected to Products page");
    logger.info("Valid login test completed successfully");
}

```

Cucumber Implementation

src/test/resources/features/login.feature

```

Scenario: Valid user login
    Given I am on the SauceDemo login page
    When I enter username "standard_user"
    And I enter password "secret_sauce"
    And I click the login button
    Then I should be redirected to the products page
    And I should see "Products" as the page title

```

src/test/java/stepdefinitions/LoginSteps.java

```

    assertTrue(productsPage.isPageLoaded());
}

@Then("I should see {string} as the page title")
public void iShouldSeePageTitle(String title) {
    assertEquals(title, productsPage.getPageTitle());
}

```

Execution Flow

Browser launches via DriverFactory → Navigate to base URL from config.properties →

LoginPage.enterUsername() - locates and enters username →

LoginPage.enterPassword() - locates and enters password →

LoginPage.clickLoginButton() - clicks login button → ProductsPage.getPageTitle() - retrieves page title →

Assert equals 'Products' → Screenshot captured on success (AllureListener)

Expected Result

✓ User is logged in and redirected to Products page showing 'Products' as the page title.

#2 Invalid Credentials Error

Critical

Smoke

Regression

Negative

Description

Verify that attempting to login with invalid credentials displays an appropriate error message.

TestNG Implementation

src/test/java/tests/LoginTest.java

```
@Epic("Authentication")
@Feature("Login")
@Story("Invalid Login")
@Severity(SeverityLevel.CRITICAL)
@Test(groups = {"smoke", "regression", "negative"})
@TestDescription("Verify error message for invalid credentials")
public void testInvalidCredentials() {
    logger.info("Starting invalid credentials test");

    loginPage.enterUsername("invalid_user");
    loginPage.enterPassword("wrong_password");
    loginPage.clickLoginButton();

    String errorMessage = loginPage.getErrorMessage();
    Assert.assertTrue(errorMessage.contains("Username and password do not match"),
```

Cucumber Implementation

src/test/resources/features/login.feature

```
@smoke @regression @negative
Scenario: Invalid credentials error message
    Given I am on the SauceDemo login page
    When I enter username "invalid_user"
    And I enter password "wrong_password"
    And I click the login button
    Then I should see an error message containing "Username and password do not match"
```

src/test/java/stepdefinitions/LoginSteps.java

```
@Then("I should see an error message containing {string}")
public void iShouldSeeErrorMessage(String expectedMessage) {
    String actualError = loginPage.getErrorMessage();
    assertTrue(actualError.contains(expectedMessage),
        "Error message should contain: " + expectedMessage);
}
```

Execution Flow

Browser launches via DriverFactory → Navigate to login page → Enter invalid username →

Enter invalid password → Click login button →

LoginPage.getErrorMessage() - retrieves error text → Assert error contains expected message →

Screenshot captured

Expected Result

✓ Error message is displayed indicating 'Username and password do not match any user in this service'.

#3 Locked Out User Error

High

Regression

Negative

Description

Verify that a locked out user cannot login and receives an appropriate error message.

TestNG Implementation

src/test/java/tests/LoginTest.java

```
@Epic("Authentication")
@Feature("Login")
@Story("Locked User")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression", "negative"})
@Description("Verify locked out user cannot login")
public void testLockedOutUser() {
    logger.info("Starting locked out user test");

    loginPage.enterUsername("locked_out_user");
    loginPage.enterPassword("secret_sauce");
    loginPage.clickLoginButton();

    String errorMessage = loginPage.getErrorMessage();
    Assert.assertTrue(errorMessage.contains("locked out"),
```

Cucumber Implementation

src/test/resources/features/login.feature

```
@regression @negative
Scenario: Locked out user cannot login
    Given I am on the SauceDemo login page
    When I enter username "locked_out_user"
    And I enter password "secret_sauce"
    And I click the login button
    Then I should see an error message containing "locked out"
```

src/test/java/stepdefinitions/LoginSteps.java

```
// Uses existing step definitions
@Then("I should see an error message containing {string}")
public void iShouldSeeErrorMessage(String expectedMessage) {
    String actualError = loginPage.getErrorMessage();
    assertTrue(actualError.contains(expectedMessage));
}
```

Execution Flow

Navigate to login page → Enter 'locked_out_user' username → Enter valid password →
Click login button → Verify error message displays → Assert message contains 'locked out'

Expected Result

✓ Error message displays 'Sorry, this user has been locked out'.

#4 Empty Username Validation

Medium

Regression

Negative

Description

Verify that submitting the login form with an empty username displays a validation error.

TestNG Implementation

src/test/java/tests/LoginTest.java

```
@Epic("Authentication")
@Feature("Login")
@Story("Field Validation")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression", "negative"})
@Description("Verify empty username validation")
public void testEmptyUsername() {
    logger.info("Starting empty username validation test");

    loginPage.enterPassword("secret_sauce");
    loginPage.clickLoginButton();

    String errorMessage = loginPage.getErrorMessage();
    Assert.assertTrue(errorMessage.contains("Username is required"),
        "Error should indicate username is required");
}
```

Cucumber Implementation

src/test/resources/features/login.feature

```
@regression @negative
Scenario: Empty username validation
    Given I am on the SauceDemo login page
    When I enter password "secret_sauce"
    And I click the login button
    Then I should see an error message containing "Username is required"
```

src/test/java/stepdefinitions/LoginSteps.java

```
// Uses existing step definitions for password and login
// Error message step handles validation
```

Execution Flow

Navigate to login page → Leave username field empty → Enter password →

Click login button → Verify validation error appears →

Assert message contains 'Username is required'

Expected Result

✓ Validation error message displays 'Username is required'.

#5 Empty Password Validation

Medium

Regression

Negative

Description

Verify that submitting the login form with an empty password displays a validation error.

TestNG Implementation

src/test/java/tests/LoginTest.java

```
@Epic("Authentication")
@Feature("Login")
@Story("Field Validation")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression", "negative"})
@Description("Verify empty password validation")
public void testEmptyPassword() {
    logger.info("Starting empty password validation test");

    loginPage.enterUsername("standard_user");
    loginPage.clickLoginButton();

    String errorMessage = loginPage.getErrorMessage();
    Assert.assertTrue(errorMessage.contains("Password is required"),
        "Error should indicate password is required");
}
```

Cucumber Implementation

src/test/resources/features/login.feature

```
@regression @negative
Scenario: Empty password validation
    Given I am on the SauceDemo login page
    When I enter username "standard_user"
    And I click the login button
    Then I should see an error message containing "Password is required"
```

src/test/java/stepdefinitions/LoginSteps.java

```
// Uses existing step definitions
```

Execution Flow

Navigate to login page → Enter username → Leave password field empty →

Click login button → Verify validation error appears →

Assert message contains 'Password is required'

Expected Result

✓ Validation error message displays 'Password is required'.

#6 Performance Glitch User Login

Low

Regression

Description

Verify that the performance_glitch_user can login successfully despite delayed response times.

TestNG Implementation

src/test/java/tests/LoginTest.java

```
@Epic("Authentication")
@Feature("Login")
@Story("Performance User")
@Severity(SeverityLevel.MINOR)
@Test(groups = {"regression"})
@Test(description = "Verify performance glitch user can login")
public void testPerformanceGlitchUser() {
    logger.info("Starting performance glitch user test");

    loginPage.enterUsername("performance_glitch_user");
    loginPage.enterPassword("secret_sauce");
    loginPage.clickLoginButton();

    // Wait for slow response
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
```

Cucumber Implementation

src/test/resources/features/login.feature

```
@regression
Scenario: Performance glitch user can login
    Given I am on the SauceDemo login page
    When I enter username "performance_glitch_user"
    And I enter password "secret_sauce"
    And I click the login button
    Then I should be redirected to the products page within 10 seconds
```

src/test/java/stepdefinitions/LoginSteps.java

```
@Then("I should be redirected to the products page within {int} seconds")
public void iShouldBeRedirectedWithinSeconds(int seconds) {
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(seconds));
    wait.until(ExpectedConditions.titleContains("Products"));
    assertTrue(productsPage.isPageLoaded());
}
```

Execution Flow

Navigate to login page → Enter performance_glitch_user credentials → Click login button →
Wait up to 10 seconds for response → Verify products page loads → Assert page title

Expected Result

✓ User successfully logs in despite delayed response, products page displays.

4. Products Tests

7 test cases in this category

#7 Products Page Displayed After Login

Critical

Smoke

Regression

Description

Verify that the products page is displayed after successful login with all expected elements visible.

TestNG Implementation

src/test/java/tests/ProductsTest.java

```
@Epic("Products")
@Feature("Product Display")
@Story("Products Page Load")
@Severity(SeverityLevel.CRITICAL)
@Test(groups = {"smoke", "regression"})
@Description("Verify products page displays after login")
public void testProductsPageDisplayed() {
    logger.info("Verifying products page display");

    // Login first
    loginPage.login(config.getProperty("valid.username"),
        config.getProperty("valid.password"));

    Assert.assertTrue(productsPage.isPageLoaded(),
        "Products page should be loaded");
}
```

Cucumber Implementation

src/test/resources/features/products.feature

```
@smoke @regression
Scenario: Products page displays after login
    Given I am logged in as a standard user
    Then I should see the products page
    And the product list should be visible
    And the cart icon should be visible
```

src/test/java/stepdefinitions/ProductSteps.java

```
@Given("I am logged in as a standard user")
public void iAmLoggedInAsStandardUser() {
    driver.get(ConfigReader.getProperty("base.url"));
    loginPage = new LoginPage(driver);
    productsPage = loginPage.login("standard_user", "secret_sauce");
}
```

Execution Flow

Login with valid credentials → Wait for products page to load → Verify page title is 'Products' →
Verify product list container visible → Verify cart icon visible → Verify header elements present

Expected Result

✓ Products page displays with product list, cart icon, and all header elements visible.

#8 Verify 6 Products Listed

High

Smoke

Regression

Description

Verify that exactly 6 products are displayed on the products page.

TestNG Implementation

src/test/java/tests/ProductsTest.java

```
@Epic("Products")
@Feature("Product Display")
@Story("Product Count")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"smoke", "regression"})
@Description("Verify 6 products are listed")
public void testSixProductsListed() {
    logger.info("Verifying product count");

    loginPage.login(config.getProperty("valid.username"),
                    config.getProperty("valid.password"));

    int productCount = productsPage.getProductCount();
    Assert.assertEquals(productCount, 6,
        "There should be exactly 6 products");
}
```

Cucumber Implementation

src/test/resources/features/products.feature

```
@smoke @regression
Scenario: Verify 6 products are listed
    Given I am logged in as a standard user
    Then I should see exactly 6 products listed
```

src/test/java/stepdefinitions/ProductSteps.java

```
@Then("I should see exactly {int} products listed")
public void iShouldSeeProductCount(int expectedCount) {
    int actualCount = productsPage.getProductCount();
    assertEquals(expectedCount, actualCount);
}
```

Execution Flow

Login with valid credentials → Navigate to products page → Count all product items →

Assert count equals 6

Expected Result

✓ Exactly 6 products are displayed on the products page.

#9 Add Single Product to Cart

Critical

Smoke

Regression

Description

Verify that a single product can be added to the cart and the cart badge updates.

TestNG Implementation

src/test/java/tests/ProductsTest.java

```
@Epic("Products")
@Feature("Add to Cart")
@Story("Single Product")
@Severity(SeverityLevel.CRITICAL)
@Test(groups = {"smoke", "regression"})
@Description("Verify adding single product to cart")
public void testAddSingleProductToCart() {
    logger.info("Adding single product to cart");

    loginPage.login(config.getProperty("valid.username"),
                    config.getProperty("valid.password"));

    productsPage.addProductToCart("Sauce Labs Backpack");

    Assert.assertEquals(productsPage.getCartBadgeCount(), 1,
```

Cucumber Implementation

src/test/resources/features/products.feature

```
@smoke @regression
Scenario: Add single product to cart
    Given I am logged in as a standard user
    When I add "Sauce Labs Backpack" to the cart
    Then the cart badge should show 1 item
    And the product should show a "Remove" button
```

src/test/java/stepdefinitions/ProductSteps.java

```
@When("I add {string} to the cart")
public void iAddProductToCart(String productName) {
    productsPage.addProductToCart(productName);
}

@Then("the cart badge should show {int} item(s)")
public void cartBadgeShouldShow(int count) {
```

Execution Flow

Login with valid credentials → Navigate to products page → Click 'Add to cart' for product →
Verify button changes to 'Remove' → Verify cart badge shows '1'

Expected Result

✓ Product is added to cart, button changes to 'Remove', cart badge shows 1.

#10 Add Multiple Products to Cart

High

Regression

Description

Verify that multiple products can be added to the cart and the cart badge updates correctly.

TestNG Implementation

src/test/java/tests/ProductsTest.java

```
@Epic("Products")
@Feature("Add to Cart")
@Story("Multiple Products")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression"})
@Description("Verify adding multiple products to cart")
public void testAddMultipleProductsToCart() {
    logger.info("Adding multiple products to cart");

    loginPage.login(config.getProperty("valid.username"),
        config.getProperty("valid.password"));

    productsPage.addProductToCart("Sauce Labs Backpack");
    productsPage.addProductToCart("Sauce Labs Bike Light");
    productsPage.addProductToCart("Sauce Labs Bolt T-Shirt");
```

Cucumber Implementation

src/test/resources/features/products.feature

```
@regression
Scenario: Add multiple products to cart
    Given I am logged in as a standard user
    When I add the following products to the cart:
        | Sauce Labs Backpack |
        | Sauce Labs Bike Light |
        | Sauce Labs Bolt T-Shirt|
```

src/test/java/stepdefinitions/ProductSteps.java

```
@When("I add the following products to the cart:")
public void iAddMultipleProducts(DataTable dataTable) {
    List<String> products = dataTable.asList();
    for (String product : products) {
        productsPage.addProductToCart(product);
    }
}
```

Execution Flow

Login with valid credentials → Add first product to cart → Verify badge shows 1 →
Add second product to cart → Verify badge shows 2 → Add third product to cart →
Verify badge shows 3

Expected Result

✓ All products are added, cart badge shows correct count of 3.

#11 Remove Product from Products Page

High

Regression

Description

Verify that a product can be removed from the cart via the products page.

TestNG Implementation

src/test/java/tests/ProductsTest.java

```
@Epic("Products")
@Feature("Remove from Cart")
@Story("Remove Product")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression"})
@Test(description = "Verify removing product from products page")
public void testRemoveProductFromProductsPage() {
    logger.info("Removing product from products page");

    loginPage.login(config.getProperty("valid.username"),
        config.getProperty("valid.password"));

    productsPage.addProductToCart("Sauce Labs Backpack");
    Assert.assertEquals(productsPage.getCartBadgeCount(), 1);
}
```

Cucumber Implementation

src/test/resources/features/products.feature

```
@regression
Scenario: Remove product from products page
    Given I am logged in as a standard user
    And I have added "Sauce Labs Backpack" to the cart
    When I click the Remove button for "Sauce Labs Backpack"
    Then the cart badge should not be visible
```

src/test/java/stepdefinitions/ProductSteps.java

```
@Given("I have added {string} to the cart")
public void iHaveAddedToCart(String productName) {
    productsPage.addProductToCart(productName);
}
```

```
@When("I click the Remove button for {string}")
```

Execution Flow

Login and add product to cart → Verify cart badge shows 1 → Click 'Remove' button on product →

Verify button changes back to 'Add to cart' → Verify cart badge disappears

Expected Result

✓ Product is removed, button reverts to 'Add to cart', cart badge disappears.

#12 Sort Products by Price

Medium

Regression

Description

Verify that products can be sorted by price (low to high and high to low).

TestNG Implementation

src/test/java/tests/ProductsTest.java

```
@Epic("Products")
@Feature("Product Sorting")
@Story("Sort by Price")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression"})
@Description("Verify sorting products by price")
public void testSortProductsByPrice() {
    logger.info("Testing product sorting by price");

    loginPage.login(config.getProperty("valid.username"),
        config.getProperty("valid.password"));

    // Sort low to high
    productsPage.sortProducts("Price (low to high)");
    List<Double> pricesLowHigh = productsPage.getAllProductPrices();
```

Cucumber Implementation

src/test/resources/features/products.feature

```
@regression
Scenario: Sort products by price low to high
    Given I am logged in as a standard user
    When I sort products by "Price (low to high)"
    Then the products should be sorted by price in ascending order

Scenario: Sort products by price high to low
```

src/test/java/stepdefinitions/ProductSteps.java

```
@When("I sort products by {string}")
public void iSortProductsBy(String sortOption) {
```

```
productsPage.sortProducts(sortOption);
}

@Then("the products should be sorted by price in ascending order")
```

Execution Flow

Login with valid credentials → Click sort dropdown → Select 'Price (low to high)' →
Get all product prices → Verify prices in ascending order → Select 'Price (high to low)' →
Verify prices in descending order

Expected Result

✓ Products are correctly sorted by price in selected order.

#13 Cart Badge Count Updates

High

Regression

Description

Verify that the cart badge count updates correctly when adding and removing products.

TestNG Implementation

src/test/java/tests/ProductsTest.java

```
@Epic("Products")
@Feature("Cart Badge")
@Story("Badge Count")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression"})
@TestDescription("Verify cart badge count updates correctly")
public void testCartBadgeCountUpdates() {
    logger.info("Testing cart badge count updates");

    loginPage.login(config.getProperty("valid.username"),
                    config.getProperty("valid.password"));

    // Add products
    productsPage.addProductToCart("Sauce Labs Backpack");
    Assert.assertEquals(productsPage.getCartBadgeCount(), 1);
```

Cucumber Implementation

src/test/resources/features/products.feature

```
@regression
Scenario: Cart badge count updates correctly
    Given I am logged in as a standard user
    When I add "Sauce Labs Backpack" to the cart
    Then the cart badge should show 1 item
    When I add "Sauce Labs Bike Light" to the cart
    Then the cart badge should show 2 items
```

src/test/java/stepdefinitions/ProductSteps.java

```
// Uses existing step definitions for add, remove, and verify
```

Execution Flow

Login with valid credentials → Add first product, verify badge shows 1 →

Add second product, verify badge shows 2 → Remove first product, verify badge shows 1 →

Remove second product, verify badge disappears

Expected Result

✓ Cart badge count accurately reflects the number of items in cart.

5. Cart Tests

3 test cases in this category

#14 View Cart Items

Critical

Smoke

Regression

Description

Verify that clicking the cart icon displays all added items with correct details.

TestNG Implementation

src/test/java/tests/CartTest.java

```
@Epic("Shopping Cart")
@Feature("View Cart")
@Story("Cart Contents")
@Severity(SeverityLevel.CRITICAL)
@Test(groups = {"smoke", "regression"})
@Description("Verify viewing cart items")
public void testViewCartItems() {
    logger.info("Testing view cart items");

    loginPage.login(config.getProperty("valid.username"),
                    config.getProperty("valid.password"));

    productsPage.addProductToCart("Sauce Labs Backpack");
    productsPage.addProductToCart("Sauce Labs Bike Light");
}
```

Cucumber Implementation

src/test/resources/features/cart.feature

```
@smoke @regression
Scenario: View items in cart
    Given I am logged in as a standard user
    And I have added the following products to the cart:
```

```
| Sauce Labs Backpack |  
| Sauce Labs Bike Light |
```

src/test/java/stepdefinitions/CartSteps.java

```
@When("I click the cart icon")  
public void iClickCartIcon() {  
    cartPage = productsPage.clickCartIcon();  
}  
  
@Then("I should see {int} items in the cart")  
public void iShouldSeeItemsInCart(int count) {
```

Execution Flow

Login and add products to cart → Click cart icon in header → Verify cart page loads →
Count items in cart → Verify each product name visible → Verify product prices correct

Expected Result

✓ Cart page displays all added items with correct names, quantities, and prices.

#15 Proceed to Checkout from Cart

Critical

Smoke

Regression

Description

Verify that clicking the checkout button from cart navigates to the checkout page.

TestNG Implementation

src/test/java/tests/CartTest.java

```
@Epic("Shopping Cart")  
@Feature("Checkout")  
@Story("Proceed to Checkout")  
@Severity(SeverityLevel.CRITICAL)  
@Test(groups = {"smoke", "regression"})  
@Description("Verify proceeding to checkout from cart")  
public void testProceedToCheckout() {  
    logger.info("Testing proceed to checkout");  
  
    loginPage.login(config.getProperty("valid.username"),  
                    config.getProperty("valid.password"));  
  
    productsPage.addProductToCart("Sauce Labs Backpack");  
    CartPage cartPage = productsPage.clickCartIcon();
```

Cucumber Implementation

src/test/resources/features/cart.feature

```
@smoke @regression  
Scenario: Proceed to checkout from cart  
    Given I am logged in as a standard user
```

```
And I have added "Sauce Labs Backpack" to the cart
And I am on the cart page
When I click the checkout button
Then I should see the checkout information page
```

src/test/java/stepdefinitions/CartSteps.java

```
@Given("I am on the cart page")
public void iAmOnCartPage() {
    cartPage = productsPage.clickCartIcon();
}

@When("I click the checkout button")
public void iClickCheckoutButton() {
```

Execution Flow

Login and add product to cart → Navigate to cart page → Click 'Checkout' button →
Verify checkout info page loads → Verify input fields visible

Expected Result

✓ Checkout information page displays with first name, last name, and postal code fields.

#16 Continue Shopping from Cart

Medium

Regression

Description

Verify that clicking 'Continue Shopping' returns to the products page.

TestNG Implementation

src/test/java/tests/CartTest.java

```
@Epic("Shopping Cart")
@Feature("Navigation")
@Story("Continue Shopping")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression"})
@Description("Verify continue shopping from cart")
public void testContinueShopping() {
    logger.info("Testing continue shopping");

    loginPage.login(config.getProperty("valid.username"),
        config.getProperty("valid.password"));

    productsPage.addProductToCart("Sauce Labs Backpack");
    CartPage cartPage = productsPage.clickCartIcon();
```

Cucumber Implementation

src/test/resources/features/cart.feature

@regression

Scenario: Continue shopping from cart

Given I am logged in as a standard user

And I have added "Sauce Labs Backpack" to the cart

And I am on the cart page

When I click the continue shopping button

Then I should return to the products page

src/test/java/stepdefinitions/CartSteps.java

```
@When("I click the continue shopping button")
public void iClickContinueShopping() {
    productsPage = cartPage.clickContinueShopping();
}
```

```
@Then("I should return to the products page")
public void iShouldReturnToProductsPage() {
```

Execution Flow

Login and add product to cart → Navigate to cart page → Click 'Continue Shopping' button →

Verify products page loads → Verify cart badge still shows item count

Expected Result

✓ User returns to products page with cart items preserved.

6. Checkout Tests

4 test cases in this category

#17 Complete Checkout with Valid Info

Critical

Smoke

Regression

Description

Verify that a complete checkout flow with valid information results in order confirmation.

TestNG Implementation

src/test/java/tests/CheckoutTest.java

```
@Epic("Checkout")
@Feature("Complete Purchase")
@Story("Successful Checkout")
@Severity(SeverityLevel.CRITICAL)
@Test(groups = {"smoke", "regression"})
@Description("Verify complete checkout with valid information")
public void testCompleteCheckout() {
    logger.info("Testing complete checkout flow");

    loginPage.login(config.getProperty("valid.username"),
        config.getProperty("valid.password"));
```

```
productsPage.addProductToCart("Sauce Labs Backpack");
CartPage cartPage = productsPage.clickCartIcon();
```

Cucumber Implementation

src/test/resources/features/checkout.feature

```
@smoke @regression
Scenario: Complete checkout with valid information
  Given I am logged in as a standard user
  And I have added "Sauce Labs Backpack" to the cart
  And I am on the checkout information page
  When I enter first name "John"
  And I enter last name "Doe"
```

src/test/java/stepdefinitions/CheckoutSteps.java

```
@Given("I am on the checkout information page")
public void iAmOnCheckoutPage() {
    cartPage = productsPage.clickCartIcon();
    checkoutPage = cartPage.clickCheckout();
}

@When("I enter first name {string}")
```

Execution Flow

Login and add product to cart → Navigate to cart and click checkout →
Enter first name, last name, postal code → Click continue to overview →
Verify product in order summary → Click finish to complete order → Verify confirmation message

Expected Result

✓ Order is placed successfully and confirmation message 'Thank you for your order!' is displayed.

#18 Checkout Missing First Name Error

High

Regression

Negative

Description

Verify that attempting checkout without first name displays a validation error.

TestNG Implementation

src/test/java/tests/CheckoutTest.java

```
@Epic("Checkout")
@Feature("Form Validation")
@Story("Missing First Name")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression", "negative"})
@Description("Verify error when first name is missing")
```

```

public void testMissingFirstName() {
    logger.info("Testing missing first name validation");

    loginPage.login(config.getProperty("valid.username"),
        config.getProperty("valid.password"));

    productsPage.addProductToCart("Sauce Labs Backpack");
    CartPage cartPage = productsPage.clickCartIcon();
}

```

Cucumber Implementation

src/test/resources/features/checkout.feature

```

@regression @negative
Scenario: Checkout without first name shows error
    Given I am logged in as a standard user
    And I have added "Sauce Labs Backpack" to the cart
    And I am on the checkout information page
    When I enter last name "Doe"
    And I enter postal code "12345"

```

src/test/java/stepdefinitions/CheckoutSteps.java

```

@Then("I should see an error message containing {string}")
public void iShouldSeeCheckoutError(String expectedMessage) {
    String actualError = checkoutPage.getErrorMessage();
    assertTrue(actualError.contains(expectedMessage));
}

```

Execution Flow

Login and add product to cart → Navigate to checkout info page → Leave first name empty →
 Enter last name and postal code → Click continue → Verify error message displays

Expected Result

✓ Error message 'Error: First Name is required' is displayed.

#19 Checkout Missing Postal Code Error

High

Regression

Negative

Description

Verify that attempting checkout without postal code displays a validation error.

TestNG Implementation

src/test/java/tests/CheckoutTest.java

```

@Epic("Checkout")
@Feature("Form Validation")
@Story("Missing Postal Code")
@Severity(SeverityLevel.NORMAL)
@Test(groups = {"regression", "negative"})
@Description("Verify error when postal code is missing")

```

```

public void testMissingPostalCode() {
    logger.info("Testing missing postal code validation");

    loginPage.login(config.getProperty("valid.username"),
        config.getProperty("valid.password"));

    productsPage.addProductToCart("Sauce Labs Backpack");
    CartPage cartPage = productsPage.clickCartIcon();
    CheckoutPage checkoutPage = cartPage.clickCheckout();
}

```

Cucumber Implementation

src/test/resources/features/checkout.feature

```

@regression @negative
Scenario: Checkout without postal code shows error
    Given I am logged in as a standard user
    And I have added "Sauce Labs Backpack" to the cart
    And I am on the checkout information page
    When I enter first name "John"
    And I enter last name "Doe"

```

src/test/java/stepdefinitions/CheckoutSteps.java

```
// Uses existing error message step definition
```

Execution Flow

Login and add product to cart → Navigate to checkout info page →
 Enter first name and last name → Leave postal code empty → Click continue →
 Verify error message displays

Expected Result

✓ Error message 'Error: Postal Code is required' is displayed.

#20 Order Confirmation Message

Critical

Smoke

Regression

Description

Verify that after completing checkout, the order confirmation page displays with correct messaging.

TestNG Implementation

src/test/java/tests/CheckoutTest.java

```

@Epic("Checkout")
@Feature("Order Confirmation")
@Story("Confirmation Page")
@Severity(SeverityLevel.CRITICAL)
@Test(groups = {"smoke", "regression"})
@Description("Verify order confirmation page")

```

```

public void testOrderConfirmationPage() {
    logger.info("Testing order confirmation page");

    loginPage.login(config.getProperty("valid.username"),
        config.getProperty("valid.password"));

    productsPage.addProductToCart("Sauce Labs Backpack");
    CartPage cartPage = productsPage.clickCartIcon();
}

```

Cucumber Implementation

src/test/resources/features/checkout.feature

```

@smoke @regression
Scenario: Order confirmation page displays correctly
    Given I have completed a checkout with valid information
    Then I should see the confirmation header "Thank you for your order!"
    And I should see a confirmation message containing "Your order has been dispatched"
    And I should see the confirmation image
    And I should see a "Back Home" button

```

src/test/java/stepdefinitions/CheckoutSteps.java

```

@Given("I have completed a checkout with valid information")
public void iHaveCompletedCheckout() {
    driver.get(ConfigReader.getProperty("base.url"));
    loginPage = new LoginPage(driver);
    productsPage = loginPage.login("standard_user", "secret_sauce");
    productsPage.addProductToCart("Sauce Labs Backpack");
    cartPage = productsPage.clickCartIcon();
}

```

Execution Flow

Complete full checkout flow → Verify confirmation page loads → Verify header text →
 Verify confirmation message → Verify confirmation image visible → Verify Back Home button visible

Expected Result

✓ Order confirmation page displays with thank you message, dispatch info, image, and back home button.

7. Framework Features Tests

8 test cases in this category

#21 Screenshot on Failure

High

Framework

Description

Verify that screenshots are automatically captured when a test fails.

TestNG Implementation

src/test/java/listeners/AllureListener.java

```
public class AllureListener implements ITestListener {  
  
    @Override  
    public void onTestFailure(ITestResult result) {  
        Object testClass = result.getInstance();  
        WebDriver driver = ((BaseTest) testClass).getDriver();  
  
        if (driver instanceof TakesScreenshot) {  
            byte[] screenshot = ((TakesScreenshot) driver)  
                .getScreenshotAs(OutputType.BYTES);  
  
            Allure.addAttachment(  
                "Screenshot on Failure - " + result.getName(),  
                "image/png",  
                new ByteArrayInputStream(screenshot),  
            );  
        }  
    }  
}
```

Cucumber Implementation

N/A - Framework Configuration

```
# Screenshot on failure is handled by Hooks.java  
# No feature file needed - automatic behavior
```

src/test/java/hooks/Hooks.java

```
public class Hooks {  
    private static WebDriver driver;  
    private static final Logger logger = LoggerFactory.getLogger(Hooks.class);  
  
    @After  
    public void tearDown(Scenario scenario) {  
        if (scenario.isFailed()) {  
            // Screenshot on failure is handled by Hooks.java  
        }  
    }  
}
```

Execution Flow

Test executes and fails → Listener/Hook catches failure event →

WebDriver captures screenshot as bytes → Screenshot attached to report →

Failure logged with details

Expected Result

✓ Screenshot is automatically captured and attached to the test report when a test fails.

#22 Parallel Execution Support

High

Framework

Description

Verify that tests can be executed in parallel across multiple threads.

TestNG Implementation

testng.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="SauceDemo Test Suite" parallel="methods" thread-count="3">

  <listeners>
    <listener class-name="listeners.Allurelistener"/>
  </listeners>

  <test name="Smoke Tests">
    <groups>
      <run>
        <include name="smoke"/>
      </run>
    </groups>
    <classes>
```

Cucumber Implementation

N/A - Framework Configuration

```
# Parallel execution configured in TestRunner.java
# and Maven Surefire plugin
```

src/test/java/runner/TestRunner.java

```
@RunWith(Cucumber.class)
@cucumberOptions(
  features = "src/test/resources/features",
  glue = {"stepdefinitions", "hooks"},
  plugin = {
    "pretty",
    "html:target/cucumber-reports/cucumber.html",
```

Execution Flow

Test suite starts → Thread pool created (3 threads) → Tests distributed across threads →

Each thread gets unique WebDriver → Tests execute concurrently → Results aggregated

Expected Result

✓ Tests execute in parallel, reducing overall execution time.

#23 Cross-Browser Testing

Medium

Framework

Description

Verify that tests can be executed across different browsers (Chrome, Firefox, Edge).

TestNG Implementation

src/test/java/factory/DriverFactory.java

```
public class DriverFactory {
    private static ThreadLocal<WebDriver> driver = new ThreadLocal<>();

    public static WebDriver initDriver(String browser) {
        WebDriver webDriver;

        switch (browser.toLowerCase()) {
            case "chrome":
                WebDriverManager.chromedriver().setup();
                ChromeOptions chromeOptions = new ChromeOptions();
                chromeOptions.addArguments("--headless");
                chromeOptions.addArguments("--disable-gpu");
                webDriver = new ChromeDriver(chromeOptions);
                break;
        }
    }
}
```

Cucumber Implementation

N/A - Framework Configuration

```
# Browser selection via config.properties or environment variable
# browser=chrome|firefox|edge
```

src/test/java/hooks/Hooks.java

```
public class Hooks {
    private static WebDriver driver;

    @Before
    public void setUp() {
        String browser = System.getProperty("browser",
            ConfigReader.getProperty("browser"));
    }
}
```

Execution Flow

Read browser config from properties/env → DriverFactory creates appropriate driver →
 WebDriverManager handles driver binaries → Browser-specific options applied →
 Tests execute on selected browser → Browser closed after tests

Expected Result

✓ Tests can be executed on Chrome, Firefox, or Edge browsers via configuration.

#24 Allure Test Reports

High

Framework

Description

Verify that Allure reports are generated with test results, screenshots, and metadata.

TestNG Implementation

pom.xml (excerpt)

```

<!-- Allure Dependencies -->
<dependency>
  <groupId>io.qameta.allure</groupId>
  <artifactId>allure-testng</artifactId>
  <version>2.24.0</version>
</dependency>

<!-- Allure Maven Plugin -->
<plugin>
  <groupId>io.qameta.allure</groupId>
  <artifactId>allure-maven</artifactId>
  <version>2.12.0</version>
  <configuration>
    <reportVersion>2.24.0</reportVersion>
  </configuration>

```

Cucumber Implementation

N/A - Framework Configuration

```
# Allure reports configured in TestRunner and pom.xml
```

pom.xml (excerpt)

```

<!-- Allure Cucumber Plugin -->
<dependency>
  <groupId>io.qameta.allure</groupId>
  <artifactId>allure-cucumber7-jvm</artifactId>
  <version>2.24.0</version>
</dependency>

```

Execution Flow

Tests execute with Allure annotations → Results written to allure-results folder →

Screenshots attached on failure → Run 'mvn allure:serve' → HTML report generated and opened →

View test history and trends

Expected Result

✓ Comprehensive Allure reports generated with test results, screenshots, history, and trends.

#25 Logging Implementation

Medium

Framework

Description

Verify that comprehensive logging is implemented for test execution tracking.

TestNG Implementation

src/main/resources/log4j2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
  <Properties>
    <Property name="LOG_PATTERN">
      %d{yyyy-MM-dd HH:mm:ss} [%t] %-5level %logger{36} - %msg%n
    </Property>
  </Properties>

  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="${LOG_PATTERN}"/>
    </Console>

    <RollingFile name="RollingFile"
      fileName="logs/test-automation.log"
```

Cucumber Implementation

N/A - Framework Configuration

```
# Cucumber uses SLF4J with Logback
```

src/main/resources/logback.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property name="LOG_PATTERN"
    value="%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n"/>

  <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
```

Execution Flow

Logger initialized in test class → Info/debug/error logs recorded → Logs output to console →
 Logs written to rolling file → Files rotated by date/size → Historical logs preserved

Expected Result

✓ Comprehensive logs are generated in console and file with proper formatting and rotation.

#26 CI/CD Pipeline Integration

High

Framework

Description

Verify that the framework integrates with Jenkins CI/CD pipeline for automated execution.

TestNG Implementation

Jenkinsfile

```
pipeline {
  agent any
```

```

tools {
    maven 'Maven-3.9.4'
    jdk 'JDK-17'
}

environment {
    BROWSER = 'chrome'
}

stages {
    stage('Checkout') {

```

Cucumber Implementation

N/A - Framework Configuration

```
# Cucumber Jenkinsfile
```

Jenkinsfile

```

pipeline {
    agent any

    tools {
        maven 'Maven-3.9.4'
        jdk 'JDK-17'
    }

```

Execution Flow

Jenkins job triggered (manual/scheduled) → Checkout code from repository →

Maven compiles project → Tests execute with parameters → Reports generated and published →

Notifications sent on failure

Expected Result

✓ Tests run automatically in Jenkins with reports published and notifications configured.

#27 Data-Driven Testing

Medium

Framework

Description

Verify that tests support data-driven execution using external data sources.

TestNG Implementation

src/test/java/tests/DataDrivenTest.java

```

public class DataDrivenTest extends BaseTest {

    @DataProvider(name = "loginData")
    public Object[][] getLoginData() {

```

```

        return new Object[][] {
            {"standard_user", "secret_sauce", true, "Products"},
            {"locked_out_user", "secret_sauce", false, "locked out"},
            {"invalid_user", "wrong_pass", false, "do not match"},
            {"", "secret_sauce", false, "Username is required"},
            {"standard_user", "", false, "Password is required"}
        };
    }

    @Test(dataProvider = "loginData", groups = {"regression", "data-driven"})
    @Description("Data-driven login test")

```

Cucumber Implementation

src/test/resources/features/login.feature

```

@regression @data-driven
Scenario Outline: Login with multiple credentials
    Given I am on the SauceDemo login page
    When I enter username "<username>"
    And I enter password "<password>"
    And I click the login button
    Then I should see "<result>"

```

src/test/java/stepdefinitions/LoginSteps.java

```

@Then("I should see {string}")
public void iShouldSee(String expectedResult) {
    if (expectedResult.equals("Products")) {
        assertEquals(expectedResult, productsPage.getPageTitle());
    } else {
        assertTrue(loginPage.getErrorMessage().contains(expectedResult));
    }
}

```

Execution Flow

Test reads data from DataProvider/Examples → Test executes once per data row →

Each iteration uses different credentials → Results validated against expected outcome →

All iterations reported separately

Expected Result

✓ Tests execute multiple times with different data sets, all iterations tracked in reports.

#28 Tagged Test Execution

Medium

Framework

Description

Verify that tests can be selectively executed using tags or groups.

TestNG Implementation

testng.xml

```

<!-- Running specific groups -->
<suite name="Tagged Tests">
  <test name="Smoke Tests Only">
    <groups>
      <run>
        <include name="smoke"/>
      </run>
    </groups>
    <classes>
      <class name="tests.LoginTest"/>
      <class name="tests.ProductsTest"/>
    </classes>
  </test>

  <test name="Negative Tests Only">

```

Cucumber Implementation

src/test/resources/features/*.feature

```

# Tags in feature files
@smoke @regression
Feature: Login functionality

@smoke @critical
Scenario: Valid user login
  Given I am on the login page

```

src/test/java/runner/TestRunner.java

```

@CucumberOptions(
  features = "src/test/resources/features",
  glue = {"stepdefinitions", "hooks"},
  tags = "@smoke", // Default tag filter
  plugin = {
    "pretty",
    "io.qameta.allure.cucumber7jvm.AllureCucumber7Jvm"
  }
)

```

Execution Flow

Specify tags via config or command line → Test runner filters tests by tags →

Only matching tests execute → Other tests skipped → Report shows executed subset

Expected Result

✓ Only tests matching specified tags/groups are executed, others are skipped.

8. Framework Components

Page Objects

TestNG - BasePage.java

```

public class BasePage {
  protected WebDriver driver;
  protected WebDriverWait wait;
  protected Logger logger = LogManager.getLogger(this.getClass());
}

```

```

public BasePage(WebDriver driver) {
    this.driver = driver;
    this.wait = new WebDriverWait(driver, Duration.ofSeconds(10));
    PageFactory.initElements(driver, this);
}

protected void click(WebElement element) {
    wait.until(ExpectedConditions.elementToBeClickable(element));
    element.click();
}

protected void type(WebElement element, String text) {
    wait.until(ExpectedConditions.visibilityOf(element));
    element.clear();
    element.sendKeys(text);
}

protected String getText(WebElement element) {

```

Cucumber - BasePage.java

```

public class BasePage {
    protected WebDriver driver;
    protected WebDriverWait wait;
    protected Logger logger = LoggerFactory.getLogger(this.getClass());

    public BasePage(WebDriver driver) {
        this.driver = driver;
        this.wait = new WebDriverWait(driver, Duration.ofSeconds(10));
        PageFactory.initElements(driver, this);
    }

    protected void click(WebElement element) {
        wait.until(ExpectedConditions.elementToBeClickable(element));
        element.click();
    }

    protected void type(WebElement element, String text) {
        wait.until(ExpectedConditions.visibilityOf(element));
        element.clear();
        element.sendKeys(text);
    }

    protected String getText(WebElement element) {

```

Driver Factory

```

public class DriverFactory {
    private static ThreadLocal<WebDriver> driver = new ThreadLocal<>();

    public static WebDriver initDriver(String browser) {
        WebDriver webDriver;

        switch (browser.toLowerCase()) {
            case "chrome":
                WebDriverManager.chromedriver().setup();
                ChromeOptions options = new ChromeOptions();
                options.addArguments("--headless");
                options.addArguments("--no-sandbox");
                options.addArguments("--disable-dev-shm-usage");
                webDriver = new ChromeDriver(options);
                break;
            case "firefox":

```

```

        WebDriverManager.firefoxdriver().setup();
        webDriver = new FirefoxDriver();
        break;
    case "edge":
        WebDriverManager.edgedriver().setup();
        webDriver = new EdgeDriver();
        break;
    default:
        throw new RuntimeException("Browser not supported");
    }

    webDriver.manage().window().maximize();
    webDriver.manage().timeouts()
        .implicitlyWait(Duration.ofSeconds(10));
    driver.set(webDriver);
    return webDriver;
}

public static WebDriver getDriver() {
    return driver.get();
}

public static void quitDriver() {
    if (driver.get() != null) {
        driver.get().quit();
        driver.remove();
    }
}
}

```

9. Configuration Files

config.properties

```

# Application Configuration
base.url=https://www.saucedemo.com
valid.username=standard_user
valid.password=secret_sauce

# Browser Configuration
browser=chrome
headless=true

# Timeout Configuration
implicit.wait=10
explicit.wait=15
page.load.timeout=30

```

testng.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="SauceDemo Test Suite" parallel="methods" thread-count="3">
    <listeners>
        <listener class-name="listeners.AllureListener"/>
    </listeners>

    <test name="All Tests">
        <classes>
            <class name="tests.LoginTest"/>
            <class name="tests.ProductsTest"/>

```

```

        <class name="tests.CartTest"/>
        <class name="tests.CheckoutTest"/>
    </classes>
</test>
</suite>

```

Cucumber TestRunner

```

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features",
    glue = {"stepdefinitions", "hooks"},
    plugin = {
        "pretty",
        "html:target/cucumber-reports/cucumber.html",
        "json:target/cucumber-reports/cucumber.json",
        "io.qameta.allure.cucumber7jvm.AllureCucumber7Jvm"
    },
    monochrome = true,
    tags = "@smoke or @regression"
)
public class TestRunner {
}

```

Cucumber Hooks

```

public class Hooks {
    private static WebDriver driver;
    private static final Logger logger = LoggerFactory.getLogger(Hooks.class);

    @Before
    public void setUp(Scenario scenario) {
        logger.info("Starting scenario: " + scenario.getName());
        String browser = System.getProperty("browser",
            ConfigReader.getProperty("browser"));
        driver = DriverFactory.initDriver(browser);
        driver.get(ConfigReader.getProperty("base.url"));
    }

    @After
    public void tearDown(Scenario scenario) {
        if (scenario.isFailed()) {
            byte[] screenshot = ((TakesScreenshot) driver)
                .getScreenshotAs(OutputType.BYTES);
            scenario.attach(screenshot, "image/png", "Screenshot");
            logger.error("Scenario failed: " + scenario.getName());
        }

        if (driver != null) {
            driver.quit();
            logger.info("Browser closed");
        }
    }

    public static WebDriver getDriver() {
        return driver;
    }
}

```

SauceDemo Selenium Automation Framework Documentation
TestNG & Cucumber BDD Combined Reference
© 2025 - Generated Documentation