Table of Contents

1. BDD Framework Overview

Behavior-Driven Development Benefits

- ✓ **Living Documentation:** Feature files serve as both tests and documentation
- ✓ **Collaboration:** Business stakeholders can read and write scenarios
- ✓ **Reusable Steps:** Step definitions can be shared across scenarios
- ✓ **Data Tables:** Easy parameterization with Examples
- ✓ **Tag-Based Execution:** Run specific scenarios with @tags
- ✓ **Hooks:** @Before/@After for setup and teardown
- ✓ **Rich Reports:** Allure integration with Gherkin steps
- ✓ **IDE Support:** Syntax highlighting and step navigation

2. Project Architecture

```
cucumber-bdd-framework/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── pages/                    # Page Object classes
│   │   │   │   ├── BasePage.java         # Base page with common
methods
│   │   │   │   ├── LoginPage.java        # Login page interactions
│   │   │   │   ├── ProductsPage.java     # Products page interactions
│   │   │   │   ├── CartPage.java         # Cart page interactions
│   │   │   │   └── CheckoutPage.java     # Checkout page interactions
│   │   │   ├── factory/
│   │   │   │   └── DriverFactory.java    # WebDriver initialization
│   │   │   └── utils/
│   │   │       └── ConfigReader.java     # Configuration file reader
│   │   └── resources/
│   │       ├── config.properties         # Test configuration
│   │       └── logback.xml               # Logging configuration
│   └── test/
│       ├── java/
│       │   ├── stepdefinitions/          # Step definition classes
│       │   │   ├── LoginSteps.java       # Login step definitions
│       │   │   ├── ProductSteps.java     # Product step definitions
│       │   │   ├── CartSteps.java        # Cart step definitions
│       │   │   └── CheckoutSteps.java    # Checkout step definitions
│       │   ├── hooks/
│       │   │   └── Hooks.java            # Before/After hooks
│       │   └── runner/
│       │       └── TestRunner.java       # Cucumber test runner
│       └── resources/
│           └── features/                 # Gherkin feature files
│               ├── login.feature         # Login scenarios
│               ├── products.feature      # Products scenarios
│               ├── cart.feature          # Cart scenarios
│               └── checkout.feature      # Checkout scenarios
├── pom.xml                               # Maven dependencies
```

```
├── Jenkinsfile                    # CI/CD pipeline
└── README.md                      # Project documentation
```

3. Gherkin Syntax Guide

Basic Keywords

Feature: Describes the feature being tested
Scenario: A single test case
Scenario Outline: Data-driven scenario
Examples: Data table for Scenario Outline

Given: Precondition or initial state
When: Action performed
Then: Expected outcome
And/But: Additional steps

Example Feature File

```
@smoke @regression
Feature: Login Functionality
  As a registered user
  I want to login to SauceDemo
  So that I can shop for products

  Background:
    Given I am on the SauceDemo login page

  @critical
  Scenario: Valid user login
    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

  @negative
  Scenario Outline: Invalid login attempts
    When I enter username "<username>"
    And I enter password "<password>"
    And I click the login button
    Then I should see an error message containing "<error>"
```

```
    Examples:
      | username        | password      | error                    |
      | invalid_user    | wrong_pass    | do not match             |
      | locked_out_user | secret_sauce  | locked out               |
      |                 | secret_sauce  | Username is required     |
```

4. Hooks & Setup

Hooks.java

src/test/java/hooks/Hooks.java

```
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;
    }
}
```

TestRunner.java

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",
        "json:target/cucumber-reports/cucumber.json",
        "io.qameta.allure.cucumber7jvm.AllureCucumber7Jvm"
    },
    monochrome = true,
    tags = "@smoke or @regression"
)
public class TestRunner {
}
```

5. Login Scenarios

6 scenarios in this category

Scenario #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.

Feature File (Gherkin)

src/test/resources/features/login.feature

```
@smoke @regression
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
```

Step Definitions

src/test/java/stepdefinitions/LoginSteps.java

```
@Given("I am on the SauceDemo login page")
public void iAmOnLoginPage() {
    driver.get(ConfigReader.getProperty("base.url"));
    loginPage = new LoginPage(driver);
}

@When("I enter username {string}")
public void iEnterUsername(String username) {
    loginPage.enterUsername(username);
}

@When("I enter password {string}")
public void iEnterPassword(String password) {
    loginPage.enterPassword(password);
}

@When("I click the login button")
public void iClickLoginButton() {
    productsPage = loginPage.clickLoginButton();
}

@Then("I should be redirected to the products page")
public void iShouldBeOnProductsPage() {
    assertTrue(productsPage.isPageLoaded());
}

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

Execution Flow

1. Browser launches via DriverFactory
2. Navigate to base URL from config.properties
3. LoginPage.enterUsername() - locates and enters username
4. LoginPage.enterPassword() - locates and enters password
5. LoginPage.clickLoginButton() - clicks login button
6. ProductsPage.getPageTitle() - retrieves page title

7. Assert equals 'Products'
8. Screenshot captured on success (AllureListener)

Expected Result

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

Scenario #2 Invalid Credentials Error

Critical @smoke @regression @negative

Description

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

Feature File (Gherkin)

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"
```

Step Definitions

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

1. Browser launches via DriverFactory
2. Navigate to login page
3. Enter invalid username
4. Enter invalid password
5. Click login button
6. LoginPage.getErrorMessage() - retrieves error text
7. Assert error contains expected message
8. Screenshot captured

Expected Result

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

Scenario #3 Locked Out User Error

High @regression @negative

Description

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

Feature File (Gherkin)

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"
```

Step Definitions

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

1. Navigate to login page
2. Enter 'locked_out_user' username
3. Enter valid password
4. Click login button
5. Verify error message displays
6. Assert message contains 'locked out'

Expected Result

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

Scenario #4 Empty Username Validation

Medium @regression @negative

Description

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

Feature File (Gherkin)

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"
```

Step Definitions

src/test/java/stepdefinitions/LoginSteps.java

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

Execution Flow

1. Navigate to login page
2. Leave username field empty
3. Enter password
4. Click login button
5. Verify validation error appears
6. Assert message contains 'Username is required'

Expected Result

✓ Validation error message displays 'Username is required'.

Scenario #5 Empty Password Validation

Medium @regression @negative

Description

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

Feature File (Gherkin)

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"
```

Step Definitions

src/test/java/stepdefinitions/LoginSteps.java

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

Execution Flow

1. Navigate to login page
2. Enter username
3. Leave password field empty
4. Click login button
5. Verify validation error appears
6. Assert message contains 'Password is required'

Expected Result

✓ Validation error message displays 'Password is required'.

Scenario #6 Performance Glitch User Login

Low @regression

Description

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

Feature File (Gherkin)

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
```

Step Definitions

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

1. Navigate to login page
2. Enter performance_glitch_user credentials
3. Click login button
4. Wait up to 10 seconds for response
5. Verify products page loads
6. Assert page title

Expected Result

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

6. Products Scenarios

7 scenarios in this category

Scenario #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.

Feature File (Gherkin)

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
```

Step Definitions

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");
}

@Then("I should see the products page")
public void iShouldSeeProductsPage() {
    assertTrue(productsPage.isPageLoaded());
}

@Then("the product list should be visible")
public void productListShouldBeVisible() {
    assertTrue(productsPage.isProductListVisible());
}

@Then("the cart icon should be visible")
public void cartIconShouldBeVisible() {
    assertTrue(productsPage.isCartIconVisible());
}
```

Execution Flow

1. Login with valid credentials
2. Wait for products page to load
3. Verify page title is 'Products'
4. Verify product list container visible
5. Verify cart icon visible
6. Verify header elements present

Expected Result

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

Scenario #8 Verify 6 Products Listed

High @smoke @regression

Description

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

Feature File (Gherkin)

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
```

Step Definitions

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

1. Login with valid credentials
2. Navigate to products page
3. Count all product items
4. Assert count equals 6

Expected Result

✓ Exactly 6 products are displayed on the products page.

Scenario #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.

Feature File (Gherkin)

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
```

Step Definitions

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) {
```

```
    assertEquals(count, productsPage.getCartBadgeCount());
}

@Then("the product should show a {string} button")
public void productShouldShowButton(String buttonText) {
    assertTrue(productsPage.isButtonVisible(buttonText));
}
```

Execution Flow

1. Login with valid credentials
2. Navigate to products page
3. Click 'Add to cart' for product
4. Verify button changes to 'Remove'
5. Verify cart badge shows '1'

Expected Result

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

Scenario #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.

Feature File (Gherkin)

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|
  Then the cart badge should show 3 items
```

Step Definitions

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

1. Login with valid credentials
2. Add first product to cart
3. Verify badge shows 1
4. Add second product to cart
5. Verify badge shows 2
6. Add third product to cart
7. Verify badge shows 3

Expected Result

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

Scenario #11 Remove Product from Products Page

High @regression

Description

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

Feature File (Gherkin)

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
```

Step Definitions

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}")
public void iClickRemoveButton(String productName) {
    productsPage.removeProductFromCart(productName);
}

@Then("the cart badge should not be visible")
public void cartBadgeShouldNotBeVisible() {
    assertFalse(productsPage.isCartBadgeVisible());
}
```

Execution Flow

1. Login and add product to cart
2. Verify cart badge shows 1
3. Click 'Remove' button on product
4. Verify button changes back to 'Add to cart'
5. Verify cart badge disappears

Expected Result

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

Scenario #12 Sort Products by Price

Medium @regression

Description

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

Feature File (Gherkin)

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

Step Definitions

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")
public void productsShouldBeSortedAscending() {
    List<Double> prices = productsPage.getAllProductPrices();
    assertTrue(isSortedAscending(prices));
}

@Then("the products should be sorted by price in descending order")
public void productsShouldBeSortedDescending() {
    List<Double> prices = productsPage.getAllProductPrices();
    assertTrue(isSortedDescending(prices));
}
```

Execution Flow

1. Login with valid credentials
2. Click sort dropdown

3. Select 'Price (low to high)'
4. Get all product prices
5. Verify prices in ascending order
6. Select 'Price (high to low)'
7. Verify prices in descending order

Expected Result

✓ Products are correctly sorted by price in selected order.

Scenario #13 Cart Badge Count Updates

High @regression

Description

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

Feature File (Gherkin)

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
  When I click the Remove button for "Sauce Labs Backpack"
  Then the cart badge should show 1 item
  When I click the Remove button for "Sauce Labs Bike Light"
  Then the cart badge should not be visible
```

Step Definitions

src/test/java/stepdefinitions/ProductSteps.java

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

Execution Flow

1. Login with valid credentials
2. Add first product, verify badge shows 1
3. Add second product, verify badge shows 2
4. Remove first product, verify badge shows 1
5. Remove second product, verify badge disappears

Expected Result

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

7. Cart Scenarios

3 scenarios in this category

Scenario #14 View Cart Items

Critical @smoke @regression

Description

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

Feature File (Gherkin)

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 |
  When I click the cart icon
  Then I should see 2 items in the cart
  And I should see "Sauce Labs Backpack" in the cart
  And I should see "Sauce Labs Bike Light" in the cart
```

Step Definitions

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) {
    assertEquals(count, cartPage.getItemCount());
}

@Then("I should see {string} in the cart")
public void iShouldSeeProductInCart(String productName) {
    assertTrue(cartPage.isProductInCart(productName));
}
```

Execution Flow

1. Login and add products to cart
2. Click cart icon in header
3. Verify cart page loads
4. Count items in cart
5. Verify each product name visible
6. Verify product prices correct

Expected Result

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

Scenario #15 Proceed to Checkout from Cart

Critical @smoke @regression

Description

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

Feature File (Gherkin)

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
```

Step Definitions

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() {
    checkoutPage = cartPage.clickCheckout();
}

@Then("I should see the checkout information page")
public void iShouldSeeCheckoutPage() {
    assertTrue(checkoutPage.isCheckoutInfoPageDisplayed());
}
```

Execution Flow

1. Login and add product to cart
2. Navigate to cart page
3. Click 'Checkout' button
4. Verify checkout info page loads
5. Verify input fields visible

Expected Result

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

Scenario #16 Continue Shopping from Cart

Medium @regression

Description

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

Feature File (Gherkin)

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
  And the cart badge should still show 1 item
```

Step Definitions

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() {
    assertTrue(productsPage.isPageLoaded());
}

@Then("the cart badge should still show {int} item(s)")
public void cartBadgeShouldStillShow(int count) {
    assertEquals(count, productsPage.getCartBadgeCount());
}
```

Execution Flow

1. Login and add product to cart
2. Navigate to cart page

3. Click 'Continue Shopping' button
4. Verify products page loads
5. Verify cart badge still shows item count

Expected Result

✓ User returns to products page with cart items preserved.

8. Checkout Scenarios

4 scenarios in this category

Scenario #17 Complete Checkout with Valid Info

Critical @smoke @regression

Description

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

Feature File (Gherkin)

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"
  And I enter postal code "12345"
  And I click continue
  Then I should see the checkout overview page
  And I should see "Sauce Labs Backpack" in the order summary
  When I click finish
  Then I should see the order confirmation
  And I should see the message "Thank you for your order!"
```

Step Definitions

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}")
public void iEnterFirstName(String firstName) {
    checkoutPage.enterFirstName(firstName);
}

@When("I enter last name {string}")
public void iEnterLastName(String lastName) {
    checkoutPage.enterLastName(lastName);
}

@When("I enter postal code {string}")
public void iEnterPostalCode(String postalCode) {
    checkoutPage.enterPostalCode(postalCode);
}

@When("I click continue")
public void iClickContinue() {
    checkoutPage.clickContinue();
}

@Then("I should see the checkout overview page")
public void iShouldSeeOverviewPage() {
    assertTrue(checkoutPage.isOverviewPageDisplayed());
}

@When("I click finish")
public void iClickFinish() {
    checkoutPage.clickFinish();
}

@Then("I should see the order confirmation")
public void iShouldSeeOrderConfirmation() {
    assertTrue(checkoutPage.isOrderConfirmed());
}

@Then("I should see the message {string}")
public void iShouldSeeMessage(String message) {
    assertEquals(message, checkoutPage.getConfirmationMessage());
}
```

Execution Flow

1. Login and add product to cart
2. Navigate to cart and click checkout
3. Enter first name, last name, postal code
4. Click continue to overview
5. Verify product in order summary
6. Click finish to complete order
7. Verify confirmation message

Expected Result

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

Scenario #18 Checkout Missing First Name Error

High @regression @negative

Description

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

Feature File (Gherkin)

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"
  And I click continue
  Then I should see an error message containing "First Name is
required"
```

Step Definitions

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

1. Login and add product to cart
2. Navigate to checkout info page
3. Leave first name empty
4. Enter last name and postal code
5. Click continue
6. Verify error message displays

Expected Result

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

Scenario #19 Checkout Missing Postal Code Error

High @regression @negative

Description

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

Feature File (Gherkin)

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"
  And I click continue
  Then I should see an error message containing "Postal Code is
required"
```

Step Definitions

src/test/java/stepdefinitions/CheckoutSteps.java

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

Execution Flow

1. Login and add product to cart
2. Navigate to checkout info page
3. Enter first name and last name
4. Leave postal code empty
5. Click continue
6. Verify error message displays

Expected Result

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

Scenario #20 Order Confirmation Message

Critical @smoke @regression

Description

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

Feature File (Gherkin)

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
```

Step Definitions

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();
    checkoutPage = cartPage.clickCheckout();
    checkoutPage.enterFirstName("John");
    checkoutPage.enterLastName("Doe");
    checkoutPage.enterPostalCode("12345");
    checkoutPage.clickContinue();
    checkoutPage.clickFinish();
}

@Then("I should see the confirmation header {string}")
public void iShouldSeeConfirmationHeader(String header) {
    assertEquals(header, checkoutPage.getConfirmationHeader());
}

@Then("I should see the confirmation image")
public void iShouldSeeConfirmationImage() {
    assertTrue(checkoutPage.isConfirmationImageVisible());
}

@Then("I should see a {string} button")
public void iShouldSeeButton(String buttonText) {
    assertTrue(checkoutPage.isButtonVisible(buttonText));
}
```

Execution Flow

1. Complete full checkout flow
2. Verify confirmation page loads
3. Verify header text

4. Verify confirmation message
5. Verify confirmation image visible
6. Verify Back Home button visible

Expected Result

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

9. Framework Features Scenarios

8 scenarios in this category

Scenario #21 Screenshot on Failure

High @framework

Description

Verify that screenshots are automatically captured when a test fails.

Feature File (Gherkin)

N/A - Framework Configuration

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

Step Definitions

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()) {
```

```
            logger.error("Scenario failed: " + scenario.getName());

            if (driver instanceof TakesScreenshot) {
                byte[] screenshot = ((TakesScreenshot) driver)
                    .getScreenshotAs(OutputType.BYTES);

                scenario.attach(
                    screenshot,
                    "image/png",
                    "Screenshot on Failure"
                );

                logger.info("Screenshot attached to scenario");
            }
        }

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

Execution Flow

1. Test executes and fails
2. Listener/Hook catches failure event
3. WebDriver captures screenshot as bytes
4. Screenshot attached to report
5. Failure logged with details

Expected Result

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

Scenario #22 Parallel Execution Support

High @framework

Description

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

Feature File (Gherkin)

N/A - Framework Configuration

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

Step Definitions

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",
        "json:target/cucumber-reports/cucumber.json",
        "io.qameta.allure.cucumber7jvm.AllureCucumber7Jvm"
    },
    tags = "@smoke or @regression"
)
public class TestRunner {
    // Parallel execution via Maven Surefire:
    // <parallel>methods</parallel>
    // <useUnlimitedThreads>true</useUnlimitedThreads>
    // Or use cucumber-jvm-parallel-plugin
}
```

Execution Flow

1. Test suite starts
2. Thread pool created (3 threads)
3. Tests distributed across threads
4. Each thread gets unique WebDriver
5. Tests execute concurrently
6. Results aggregated

Expected Result

✓ Tests execute in parallel, reducing overall execution time.

Scenario #23 Cross-Browser Testing

Medium @framework

Description

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

Feature File (Gherkin)

N/A - Framework Configuration

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

Step Definitions

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"));
        driver = DriverFactory.initDriver(browser);
        logger.info("Browser initialized: " + browser);
    }

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

// Run with: mvn test -Dbrowser=firefox
```

Execution Flow

1. Read browser config from properties/env
2. DriverFactory creates appropriate driver
3. WebDriverManager handles driver binaries
4. Browser-specific options applied
5. Tests execute on selected browser
6. Browser closed after tests

Expected Result

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

Scenario #24 Allure Test Reports

High @framework

Description

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

Feature File (Gherkin)

N/A - Framework Configuration

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

Step Definitions

pom.xml (excerpt)

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

<!-- TestRunner plugin configuration -->
plugin = {
```

```
    "io.qameta.allure.cucumber7jvm.AllureCucumber7Jvm"
}

// Generate report: mvn allure:serve
// Or: mvn allure:report
```

Execution Flow

1. Tests execute with Allure annotations
2. Results written to allure-results folder
3. Screenshots attached on failure
4. Run 'mvn allure:serve'
5. HTML report generated and opened
6. View test history and trends

Expected Result

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

Scenario #25 Logging Implementation

Medium @framework

Description

Verify that comprehensive logging is implemented for test execution tracking.

Feature File (Gherkin)

N/A - Framework Configuration

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

Step Definitions

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>
            <pattern>${LOG_PATTERN}</pattern>
        </encoder>
    </appender>

    <appender name="FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>logs/cucumber-tests.log</file>
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>logs/cucumber-tests.%d{yyyy-MM-
dd}.log</fileNamePattern>
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <pattern>${LOG_PATTERN}</pattern>
        </encoder>
    </appender>

    <root level="INFO">
        <appender-ref ref="CONSOLE"/>
        <appender-ref ref="FILE"/>
    </root>
</configuration>
```

Execution Flow

1. Logger initialized in test class
2. Info/debug/error logs recorded
3. Logs output to console
4. Logs written to rolling file
5. Files rotated by date/size
6. Historical logs preserved

Expected Result

✓ Comprehensive logs are generated in console and file with proper

formatting and rotation.

Scenario #26 CI/CD Pipeline Integration

High @framework

Description

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

Feature File (Gherkin)

N/A - Framework Configuration

```
# Cucumber Jenkinsfile
```

Step Definitions

Jenkinsfile

```
pipeline {
    agent any

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

    parameters {
        choice(name: 'BROWSER', choices: ['chrome', 'firefox',
'edge'])
        choice(name: 'TAGS', choices: ['@smoke', '@regression',
'@smoke or @regression'])
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'main',
                    url: 'https://github.com/user/cucumber-bdd-
framework.git'
            }
```

```
        }

        stage('Run Tests') {
            steps {
                sh """
                    mvn test \
                    -Dbrowser=${params.BROWSER} \
                    -Dcucumber.filter.tags="${params.TAGS}"
                """
            }
        }

        stage('Generate Reports') {
            steps {
                cucumber buildStatus: 'UNSTABLE',
                         jsonReportDirectory: 'target/cucumber-
reports',
                         reportTitle: 'Cucumber Report'

                allure results: [[path: 'target/allure-results']]
            }
        }
    }
}
```

Execution Flow

1. Jenkins job triggered (manual/scheduled)
2. Checkout code from repository
3. Maven compiles project
4. Tests execute with parameters
5. Reports generated and published
6. Notifications sent on failure

Expected Result

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

Scenario #27 Data-Driven Testing

Medium @framework

Description

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

Feature File (Gherkin)

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>"

  Examples:
    | username          | password      | result                    |
    | standard_user     | secret_sauce  | Products                  |
    | locked_out_user   | secret_sauce  | locked out                |
    | invalid_user      | wrong_pass    | do not match              |
    |                   | secret_sauce  | Username is required      |
    | standard_user     |               | Password is required      |
```

Step Definitions

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

1. Test reads data from DataProvider/Examples
2. Test executes once per data row
3. Each iteration uses different credentials

4. Results validated against expected outcome
5. All iterations reported separately

Expected Result

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

Scenario #28 Tagged Test Execution

Medium @framework

Description

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

Feature File (Gherkin)

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
  ...

@regression @negative
Scenario: Invalid credentials
  Given I am on the login page
  ...

# Run specific tags:
# mvn test -Dcucumber.filter.tags="@smoke"
# mvn test -Dcucumber.filter.tags="@smoke and @regression"
# mvn test -Dcucumber.filter.tags="@regression and not @slow"
```

Step Definitions

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"
    }
)
public class TestRunner {
}

// Dynamic tag selection via system property:
// -Dcucumber.filter.tags="@smoke or @regression"
```

Execution Flow

1. Specify tags via config or command line
2. Test runner filters tests by tags
3. Only matching tests execute
4. Other tests skipped
5. Report shows executed subset

Expected Result

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

10. Step Definitions

Page Objects

src/main/java/pages/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) {
        wait.until(ExpectedConditions.visibilityOf(element));
        return element.getText();
    }
}
```

LoginPage.java

src/main/java/pages/LoginPage.java

```
public class LoginPage extends BasePage {

    @FindBy(id = "user-name")
    private WebElement usernameInput;

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

    @FindBy(id = "login-button")
    private WebElement loginButton;

    @FindBy(css = "[data-test='error']")
    private WebElement errorMessage;

    public LoginPage(WebDriver driver) {
        super(driver);
    }

    public void enterUsername(String username) {
        type(usernameInput, username);
    }

    public void enterPassword(String password) {
        type(passwordInput, password);
```

```
    }

    public ProductsPage clickLoginButton() {
        click(loginButton);
        return new ProductsPage(driver);
    }

    public ProductsPage login(String username, String password) {
        enterUsername(username);
        enterPassword(password);
        return clickLoginButton();
    }

    public String getErrorMessage() {
        return getText(errorMessage);
    }
}
```

11. Configuration

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
```

cucumber.properties

```
cucumber.publish.quiet=true
cucumber.publish.enabled=false
```

12. CI/CD Pipeline

Jenkinsfile

```
pipeline {
    agent any

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

    parameters {
        choice(name: 'BROWSER', choices: ['chrome', 'firefox', 'edge'])
        choice(name: 'TAGS', choices: ['@smoke', '@regression', '@smoke or
@regression'])
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'main',
                    url: 'https://github.com/user/cucumber-bdd-
framework.git'
            }
        }

        stage('Run Tests') {
            steps {
                sh """
                    mvn test \
                    -Dbrowser=${params.BROWSER} \
                    -Dcucumber.filter.tags="${params.TAGS}"
                """
            }
        }

        stage('Generate Reports') {
            steps {
                cucumber buildStatus: 'UNSTABLE',
                         jsonReportDirectory: 'target/cucumber-reports',
                         reportTitle: 'Cucumber Report'

                allure results: [[path: 'target/allure-results']]
            }
        }
    }
}
```

Running Tests

Command Line Execution

```
# Run all scenarios
mvn clean test

# Run by tags
mvn test -Dcucumber.filter.tags="@smoke"
mvn test -Dcucumber.filter.tags="@regression"
mvn test -Dcucumber.filter.tags="@smoke and @critical"
mvn test -Dcucumber.filter.tags="not @slow"

# Run on specific browser
mvn test -Dbrowser=firefox

# Generate Allure report
mvn allure:serve

# Generate Cucumber HTML report
# Reports available at: target/cucumber-reports/cucumber.html
```

Tag Strategies

Tag	Usage
`@smoke`	Critical path tests for quick validation
`@regression`	Full regression test suite
`@negative`	Negative/error scenario tests
`@critical`	Business-critical scenarios
`@wip`	Work in progress (skip in CI)

SauceDemo Selenium Automation Framework

Cucumber BDD Framework Documentation