Q. What is Cucumber and why is it used?

- Cucumber is a testing framework that supports Behaviour Driven Development (BDD).
- It allows writing tests in plain language which stakeholders can easily understand.
- It bridges the gap between non-technical stakeholders and developers by allowing the creation of documentation that is actually runnable as tests.

Q. Explain the structure of a cucumber Feature file.

- A Cucumber feature file contains scenarios written in Gherkin language.
- It is a high-level description of a software feature, written in a .feature file.
- Each scenario within the feature is described using Scenario: followed by steps such as Given, When, Then, And to describe the flow.

Q. What is a Scenario?

A Scenario is a single executable test case that represents a specific behavior or functionality being tested within a feature, followed steps like Given, When, and Then.

Q. What is a Scenario Outline?

A Scenario Outline is a template for scenarios that need to be executed multiple times with different sets of data, using the Examples keyword for data input.

Q. What is DataTable?

DataTable is a way in Cucumber to handle structured data input, allowing for data to be passed into steps in a tabular format to test various conditions within a scenario.

Q. What are Step Definitions in Cucumber?

- Step Definitions are the actual code that gets executed when Cucumber runs the scenarios.
- They link the steps in the feature file to the code that performs the operations described by those steps.
- They are typically placed in classes under the stepDefinitions package.

Q. What is the difference between dryRun and strict in Cucumber?

- **dryRun** = **true -:** Checks if all steps in the feature files have corresponding step definitions without actually running the tests.
- **strict** = **true -:** Fails the execution if there are any undefined or pending steps, ensuring that no step goes unimplemented.

Q. What are Cucumber Hooks and how are they used?

- Hooks are blocks of code in Cucumber that can be executed before or after each scenario by using @Before & @After.
- They are used for setup and teardown operations, such as initializing WebDriver, setting up databases, or cleaning up after test execution.

Q. How do you execute a Cucumber Test using Jenkins?

To execute a Cucumber test in Jenkins, configure a Jenkins job with build steps that run the test using Maven, and integrate Cucumber reports for result visualization.

Q. How do you generate reports in Cucumber?

We can generate report using plugins **@CucumberOptions** annotation.

For example, plugin = {"pretty", "html:target/cucumber-html-report", "json:target/cucumber.json"}

will generate both HTML and JSON reports.

We can also use the **maven-cucumber-reporting plugin** to generate detailed reports.

Q. How do you handle multiple sets of test data in Cucumber?

- Cucumber supports data-driven testing through **Scenario Outline** and
  Examples keyword.
- The **Scenario Outline** is used to run the same scenario with different sets of data specified in the **Examples table**.

Q. Explain the concept of Tags in Cucumber.

- Tags are used to group and filter scenarios. We can tag scenarios with @smoke, @regression, etc.,
- Then run tests based on these tags using the @CucumberOptions annotation, for example: tags = "@smoke".

Q. What is the role of the Page Object Model (POM) in Cucumber?

- The Page Object Model is a design pattern that helps create an object repository for web elements.
- It separates the test logic from the page-specific code, making the tests more maintainable and reusable.
- Each page of the application is represented by a class, and web elements are defined as variables.

File Structure

1. Created a SignIn.feature file under the src/test/resources/Features folder.
2. Created a LoginPage class under the src/test/java path inside the pageObjects package.
3. Created a SigninSteps class under the src/test/java path inside the stepDefination package.
4. Created a TestRunner_SignIn class under the src/test/java path inside the testRunner package.

<plugin> Usage in POM.xml file

- **maven-surefire-plugin-:** Provides a framework to run tests in Maven.
- **maven-cucumber-reporting -:** Generates HTML and JSON reports for Cucumber test executions.

Q. What is the role of the @CucumberOptions annotation?

- @CucumberOptions is used to configure Cucumber execution.
- It can specify

Cucumber Options-:

- **Features**: The path of the Features files
- **Glue**: The path of Step Definition files
- **dryRun** = true : to check the mapping is proper between "Feature file" and "Step definition file"
- **Tags**: @smoke
- **Monochorome** = true : Remove unnecessary info from console
- **Format**: To generate different type of reporting
- **Strict** = true: It will check if any step is not defined in Step Definition file

Feature file

Feature: Book Cart E-Commerce Functionality

Scenario Outline: User Registration and Login when account already exists

- **Given** User navigates to the 99BooksCart website
- **And** user opens URL "https://www.99bookscart.com/"
- **And** click on profile button

- **And** User clicks on the Sign Up button
- **And** User enters valid registration details <email> and <password>
- **And** User submits the registration form
- **Then** User should see a message

# Steps for logging in since the account already exists

- **When** User clicks on the Sign In link
- **When** User enters valid existing credentials <email> and <password>
- **Then** User clicks on the Sign In button
- **Then** user is searching for a book and add them into cart

Examples:

email	password
kumarrajatpradhan5364@gmail.com	Rajat@5364