**1.Automation experience and about framework you have worked on.**

I have accumulated two years of dedicated experience in automation testing, with a significant focus on utilizing the Behavior-Driven Development (BDD) approach through the Cucumber framework. Within this timeframe, I've actively contributed to projects where I leveraged Cucumber to bridge the communication gap between stakeholders and development teams by writing feature files in a human-readable format.

In my previous role, I primarily used Selenium to automate web application testing, and I am familiar with creating and executing test scripts using this framework.

**2.Tell brief about your roles and responsibilities in project and explain the automation framework used in project.**

* In my previous role, I served as an Automation Test Engineer, where my responsibilities included designing, implementing, and maintaining automated test suites to ensure the quality and reliability of software products. I collaborated closely with cross-functional teams to understand project requirements, translate them into test scenarios, and automate them using the Behavior-Driven Development (BDD) approach with the Cucumber framework.
* I am refining and optimizing the automation framework, leveraging Cucumber's flexibility and scalability to enhance test coverage and maintainability. By continuously assessing and improving testing processes, I contributed to the overall efficiency and effectiveness of the project, ultimately delivering high-quality software solutions that met client expectations.
* As part of this role, I authored feature files using Cucumber's Gherkin syntax. These feature files captured business requirements in a human-readable format, enabling efficient collaboration and alignment of testing efforts with project goals. Additionally, I developed step definitions and reusable components within the Cucumber framework to execute test scenarios and validate application functionality.

**3.Explain the selenium project structure in previous project. (Which Build tool used, how the repositories are maintained, how the data is providing, what kind of reports are generated, what type of dependencies are part of the project, which framework)**

In the previous Selenium project, we maintained a structured project layout to ensure scalability, maintainability, and collaboration across team members. Here's an overview of the project structure and its components:

1. Build Tool: We utilized Maven as the build automation tool for managing project dependencies, compiling source code, and executing tests. Maven's declarative approach simplified project configuration and dependency management.
2. Repository Management: The project source code was stored in a version control system such as Git. We followed GitFlow or a similar branching strategy to manage feature development, bug fixes, and releases. This allowed for efficient collaboration and version control among team members.
3. Data Provisioning: Test data was provided through various sources depending on the specific requirements of each test case. This could include static data files, database queries, or dynamic data generation methods. We ensured data integrity and privacy by implementing data masking techniques where necessary.
4. Report Generation: We leveraged reporting libraries such as Extent Reports or Allure to generate comprehensive and visually appealing test reports. These reports provided detailed insights into test execution results, including test pass/fail status, screenshots of failed tests, and execution timelines. Integration with CI/CD pipelines enabled automatic report generation and sharing.
5. Dependencies: The project dependencies included Selenium WebDriver for browser automation, along with additional libraries and frameworks based on the chosen testing framework. For example, if using TestNG or JUnit as the testing framework, dependencies for these frameworks were included. We also incorporated logging libraries, assertion libraries (such as AssertJ or Hamcrest), and utilities for handling configurations and test data.
6. Framework: The Selenium project followed a modular and scalable framework such as Page Object Model (POM) or Screenplay Pattern. This facilitated separation of concerns, improved code reusability, and enhanced maintainability. We implemented design patterns and best practices to optimize test automation code, minimize code duplication, and improve readability.

**4.Which selenium version did you used in your previous project?**

In our previous project, we utilized Selenium version 4.x. Selenium 4 brought significant enhancements and new features compared to previous versions, including improved support for modern web technologies. By leveraging Selenium 4, we were able to enhance our test automation capabilities, improve test stability, and adapt to the evolving landscape of web development.

**5.What is the latest Selenium version and new features?**

Selenium 4 was the latest stable version available. However, I don't have real-time access to current updates beyond that point.

Some of the key features introduced in Selenium 4 include:

(i) W3C WebDriver Protocol: Selenium 4 aligns more closely with the W3C WebDriver protocol, enhancing compatibility and consistency across different browser implementations.

(ii) Improved Selenium Grid: Selenium Grid has been revamped in Selenium 4 to support various deployment modes, including standalone, hub/node, and Docker-based grids. It offers better support for parallel testing and improved scalability.

(iii) Relative Locators: Selenium 4 introduced new locator strategies such as "near," "above," "below," "toLeftOf," and "toRightOf" to locate elements relative to other elements on the web page. This makes it easier to write maintainable and robust test scripts, especially for dynamic web applications.

(iv) New APIs: Selenium 4 introduced several new APIs and utility methods to simplify common tasks in test automation. For example, the Actions class has been enhanced with new methods for performing keyboard actions, handling mouse events, and executing complex interactions.

(v) DevTools Integration: Selenium 4 provides better integration with browser DevTools, allowing testers to access and leverage browser debugging features directly from their Selenium scripts. This facilitates better debugging and troubleshooting of test failures.

**6.Types of frameworks. Explain each briefly.**

In Selenium, various automation frameworks help structure and organize test code, making it more scalable, maintainable, and reusable. Here are some common types of frameworks used in Selenium automation:

(i) Data-Driven Framework:

Data-Driven Framework separates test data from test scripts, allowing tests to be executed with different input datasets.

Test data is typically stored in external files like Excel sheets, CSV files, or databases. Test scripts retrieve data from these sources to drive test execution.

It enables comprehensive test coverage and facilitates testing with various data scenarios.

(ii) Keyword-Driven Framework:

Keyword-Driven Framework abstracts test steps into keywords or actions stored in external repositories.

Test scripts call these keywords, which internally execute the corresponding test steps.

It enhances test case maintainability, readability, and reusability by promoting modularization and abstraction.

(iii) Hybrid Framework:

Hybrid Framework combines the features of multiple frameworks, such as POM, Data-Driven, and Keyword-Driven frameworks.

It offers flexibility in test design and execution, allowing testers to leverage the strengths of different frameworks based on project requirements.

Test scripts may incorporate page objects, data-driven techniques, and keyword-driven components as needed.

(iv) Behavior-Driven Development (BDD) Framework:

BDD Framework focuses on collaborative development and communication between stakeholders, developers, and testers.

It uses a human-readable language like Gherkin to define test scenarios in feature files, which are then automated using Selenium.

BDD promotes a shared understanding of project requirements and facilitates the creation of executable specifications.

(v) TestNG Framework:

TestNG is a testing framework for Java that supports various testing methodologies, including unit, functional, and integration testing.

It offers features like annotations, assertions, parameterization, grouping, and parallel execution, enhancing test flexibility and scalability.

TestNG integrates seamlessly with Selenium WebDriver, making it a popular choice for Selenium automation in the Java ecosystem.