**Manual Testing:**

Example of a defect with High Severity and Low priority?

Difference between Test plan & Test Strategy?

What is usability testing?

Explain STLC? Elaborate in each step?

In which phase you will design test case?#

What is test data?

Difference between positive and negative test data? Epi path testing

Difference between smoke and sanity testing?

High level testing for a webpage?

Test coverage?

Positive and negative scenarios for google map?

What all things u have to review in your junior’s test cases?

Parameters determined to review the test cases

How will u handle the situation if Scenario fails in a DEMO?

How will you keep updating yourself with new technologies?

**Automation Testing:**

1. Tell me about the Automation framework u worked?

**1. Overview:**

"I developed a comprehensive test automation framework for our web application using Java and Selenium. The framework was designed to integrate various tools and technologies to ensure efficient, reliable, and scalable testing."

**2. Framework Structure and Design:**

* **Technology Stack**: "The framework was built using Java as the programming language and Selenium WebDriver for browser automation."
* **Testing Framework**: "We used both JUnit and TestNG for writing and organizing our test cases, benefiting from JUnit's simplicity and TestNG's advanced features like parallel execution and data-driven testing."

**3. Key Components:**

* **Maven**: "Maven was used for project management and build automation. We managed dependencies, such as Selenium and TestNG, through the pom.xml file, ensuring consistency across different environments."
* **Page Object Model (POM)**: "We followed the Page Object Model to create reusable page classes that encapsulate web elements and actions, promoting code reuse and maintainability."

**4. Continuous Integration/Continuous Deployment (CI/CD):**

* **Jenkins**: "We integrated our framework with Jenkins for CI/CD. Jenkins was configured to pull the latest code from GitHub, trigger builds, execute tests, and generate reports automatically upon every commit."
* **GitHub**: "GitHub was used for version control, enabling collaboration among team members and tracking changes. Our Jenkins jobs were set up to monitor the GitHub repository for changes."

**5. API Testing:**

* **Tools Used**: "For API testing, we used RestAssured along with Postman for manual testing. Automated API tests were written in Java using RestAssured and integrated into the same framework."
* **Integration**: "API tests were also executed through Maven and Jenkins, ensuring comprehensive test coverage that included both UI and API layers."

**6. Example Workflow:**

"Here's a typical workflow in our framework:

1. **Development**: Write test scripts in Java, using Selenium for UI tests and RestAssured for API tests.
2. **Management**: Use Maven to manage dependencies and run tests locally.
3. **Version Control**: Commit code changes to GitHub.
4. **CI/CD**: Jenkins detects changes in GitHub, pulls the latest code, executes the tests, and generates reports.
5. **Reporting**: Test results are published in Jenkins, and notifications are sent out for any failures."

**7. Achievements and Impact:**

* **Efficiency**: "The framework significantly reduced manual testing efforts and ensured quicker feedback on code changes."
* **Quality**: "Improved the overall quality of our releases by catching bugs early in the development cycle."
* **Scalability**: "The modular design and CI/CD integration allowed us to easily scale our testing efforts as the project grew."
* Can u upload a file as a part of automation, is it possible via selenium ?

Yes, it is possible to upload a file as part of automation using Selenium. Typically, this involves locating the file input element and sending the file path to it. For example:

WebElement uploadElement = driver.findElement(By.id("upload"));

String filePath = "/path/to/your/file.txt";

uploadElement.sendKeys(filePath);

This method works well for standard file input elements. For more complex scenarios involving custom components or native file dialogs, additional tools like AutoIt or the Robot class can be used to handle the upload process."

(his Ans: click method, Action and Robot class)

1. Write a sample code for page factory class?

**1. Explain Page Factory:**

"Page Factory is a way to initialize web elements in a Page Object Model (POM) framework. It helps in creating an object repository for web elements. Page Factory initializes every web element with a reference to a corresponding element on the actual web page using @FindBy annotations."

**2. Provide the Sample Code:**

"Here’s a sample code for a Page Factory class in a login page scenario:"

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class LoginPage {

WebDriver driver;

// Constructor to initialize elements

public LoginPage(WebDriver driver) {

this.driver = driver;

PageFactory.initElements(driver, this);

}

// WebElements using @FindBy annotation

@FindBy(id = "username")

WebElement usernameField;

@FindBy(id = "password")

WebElement passwordField;

@FindBy(id = "loginButton")

WebElement loginButton;

// Actions on the web elements

public void setUsername(String username) {

usernameField.sendKeys(username);

}

public void setPassword(String password) {

passwordField.sendKeys(password);

}

public void clickLoginButton() {

loginButton.click();

}

// Login method that combines all actions

public void login(String username, String password) {

setUsername(username);

setPassword(password);

clickLoginButton();

}

}

**3. Explain the Code:**

* **Constructor**: "The constructor initializes the web elements using PageFactory.initElements(driver, this);. This is necessary for Page Factory to work."
* **WebElements**: "The @FindBy annotation is used to locate the elements on the page. This helps in keeping the object repository clean and easy to maintain."
* **Actions**: "Methods like setUsername(), setPassword(), and clickLoginButton() perform actions on the web elements."
* **Login Method**: "The login() method is a combination of actions to perform the login operation in one call, making the test scripts more readable."

**4. Example Usage in a Test Case:**

"Here’s how you can use this LoginPage class in your test case:"

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class LoginTest {

WebDriver driver;

LoginPage loginPage;

@BeforeClass

public void setUp() {

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

driver = new ChromeDriver();

driver.get("https://example.com/login");

loginPage = new LoginPage(driver);

}

@Test

public void testValidLogin() {

loginPage.login("username", "password");

// Add assertions to verify successful login

}

@AfterClass

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

}

Write a sample code for Page Action Class?

**1. Explain Page Action Class:**

"A Page Action Class in Selenium is used to define actions or methods that interact with the elements defined in the Page Factory class. This helps in separating the test logic from the web element definitions, making the code more modular and maintainable."

**2. Provide the Sample Code:**

"Here’s a sample code for a Page Action Class that works with the LoginPage class we discussed earlier:"

**LoginPage.java (Page Factory Class):**

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class LoginPage {

WebDriver driver;

// Constructor to initialize elements

public LoginPage(WebDriver driver) {

this.driver = driver;

PageFactory.initElements(driver, this);

}

// WebElements using @FindBy annotation

@FindBy(id = "username")

WebElement usernameField;

@FindBy(id = "password")

WebElement passwordField;

@FindBy(id = "loginButton")

WebElement loginButton;

}

**LoginActions.java (Page Action Class):**

public class LoginActions {

WebDriver driver;

LoginPage loginPage;

// Constructor to initialize the LoginPage object

public LoginActions(WebDriver driver) {

this.driver = driver;

loginPage = new LoginPage(driver);

}

// Method to set the username

public void enterUsername(String username) {

loginPage.usernameField.sendKeys(username);

}

// Method to set the password

public void enterPassword(String password) {

loginPage.passwordField.sendKeys(password);

}

// Method to click the login button

public void clickLoginButton() {

loginPage.loginButton.click();

}

// Method to perform the login action

public void login(String username, String password) {

enterUsername(username);

enterPassword(password);

clickLoginButton();

}

}

**3. Explain the Code:**

* **Constructor**: "The constructor initializes the LoginPage object, which is necessary to access the web elements defined in the Page Factory class."
* **Methods**: "The enterUsername(), enterPassword(), and clickLoginButton() methods interact with the corresponding web elements. The login() method combines these actions to perform a complete login operation."

**4. Example Usage in a Test Case:**

"Here’s how you can use the LoginActions class in your test case:"

**LoginTest.java (Test Case):**

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class LoginTest {

WebDriver driver;

LoginActions loginActions;

@BeforeClass

public void setUp() {

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

driver = new ChromeDriver();

driver.get("https://example.com/login");

loginActions = new LoginActions(driver);

}

@Test

public void testValidLogin() {

loginActions.login("username", "password");

// Add assertions to verify successful login

}

@AfterClass

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

}

"A Page Action Class in Selenium is used to define actions that interact with web elements defined in a Page Factory class. Here’s a sample Page Action Class for a login page, which includes methods to set the username, password, and click the login button, as well as a method to perform the login action. Additionally, I’ve provided an example of how to use this class in a test case to perform a login operation. This approach helps in keeping the test logic separate from the element definitions, making the code more modular and maintainable."

* What is Sikuli?

Sikuli is an open-source automation tool that uses image recognition to automate tasks by interacting with GUI components based on screenshots. It works by matching images of UI elements to automate interactions, making it highly versatile for automating tasks across different platforms, including web and desktop applications. Sikuli scripts are typically written in Python or Jython. It is particularly useful for automating interactions with legacy systems, complex UI components, and applications where traditional automation tools like Selenium cannot be used effectively.

* How do u resolve conflicts in GIT?

**1. Acknowledge the Importance of Conflict Resolution:**

"Conflict resolution is a critical aspect of using Git, especially when working in a team environment where multiple people may be working on the same codebase."

**2. Identify the Conflict:**

"When a conflict occurs, Git will alert you during the merge process, and it will mark the conflicted files with conflict markers. You can identify these conflicts using Git commands or your IDE."

**3. Steps to Resolve Conflicts:**

**a. Fetch and Pull the Latest Changes:**

"First, I fetch and pull the latest changes from the remote repository to ensure I have the most recent updates."

git fetch origin

git pull origin branch\_name

**b. Identify the Conflicted Files:**

"Git will list the files with conflicts. I use git status to see which files are in conflict."

git status

**c. Open and Edit the Conflicted Files:**

"I open the conflicted files using my preferred text editor or IDE. Conflicted areas are marked with conflict markers (<<<<<<<, =======, >>>>>>>). I carefully review the changes from both branches."

**d. Resolve the Conflicts:**

"I manually edit the conflicting sections to choose the correct code or combine the changes in a meaningful way. After resolving the conflicts, I remove the conflict markers."

**e. Mark the Conflicts as Resolved:**

"Once the conflicts are resolved, I add the resolved files to the staging area."

git add resolved\_file

**f. Commit the Changes:**

"After staging the resolved files, I commit the changes to complete the merge process."

git commit -m "Resolved merge conflicts"

**4. Verify the Resolution:**

"I ensure that the code compiles and runs correctly after resolving conflicts. I run tests to verify that everything works as expected."

**5. Push the Changes:**

"Finally, I push the resolved changes to the remote repository."

git push origin branch\_name

**6. Use Tools to Assist in Conflict Resolution:**

"In addition to manual conflict resolution, I sometimes use tools like GitKraken, SourceTree, or the built-in merge conflict resolution tools in IDEs like IntelliJ or VS Code, which provide a visual interface to help resolve conflicts more efficiently."

**7. Communicate with Team Members:**

"If the conflict is complex or unclear, I communicate with my team members to understand their changes and reach a consensus on the best resolution."

**Putting It All Together:**

"Conflict resolution is an essential part of using Git. When a conflict arises, I follow a systematic process to resolve it. I start by fetching and pulling the latest changes, identifying the conflicted files using git status, and then manually editing the files to resolve conflicts. After marking the conflicts as resolved, I commit and push the changes. I also use visual tools when necessary and communicate with team members to ensure the best resolution. Ensuring that the code works correctly after resolving conflicts is crucial, so I always run tests to verify the resolution.

Difference between Action and Actions in selenium?

**Action vs. Actions**

**1. Action Class:**

* **Purpose**: Represents a single user interaction action.
* **Usage**: Rarely used directly by users. It's more common to use the Actions class which provides a more user-friendly way to create complex user interactions.
* **Example**: If you want to encapsulate a specific action like a click or a key press, it would be represented as an Action object.

import org.openqa.selenium.interactions.Action;

import org.openqa.selenium.interactions.Actions;

// Example

Action clickAction = new Actions(driver).click().build();

clickAction.perform();

**2. Actions Class:**

* **Purpose**: Provides a way to build and perform a series of user interactions. It's a builder class used to construct complex actions sequences.
* **Usage**: Commonly used to create sequences of actions like drag-and-drop, multiple key presses, etc.
* **Example**: You can chain multiple actions together to create a complex user interaction.

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Actions;

// Example

Actions actions = new Actions(driver);

actions.moveToElement(element).click().build().perform();

**Summary:**

* **Action**: Represents a single user interaction. Typically, you'll work with it indirectly through the Actions class.
* **Actions**: A builder class used to create a sequence of user interactions. It is commonly used for more complex actions.

In practice, you will mostly use the Actions class to build and execute action sequences, while the Action class is more of an underlying mechanism.

1. What is TestNG?
2. How to automate a button and validate the colour display in selenium?
3. What is Jenkins?
4. Write a Java program to print all the duplicate characters in a String in automation?

**Scenario based/General:**

1. How many people are there in your team?

"In our team, we have 8 members. This includes:

* **2 Developers**: They are responsible for writing and maintaining the application code.
* **1 Business Analyst**: Gathers and defines requirements from stakeholders.
* **1 Product Owner**: Prioritizes the backlog and ensures the team is working on the most valuable tasks.
* **2 Testers**: Focus on manual and automated testing to ensure the quality of the product.
* **1 DevOps Engineer**: Manages our CI/CD pipelines and infrastructure.
* **1 Project Manager**: Oversees the project timelines and coordinates among team members.

My role in the team is as a [Your Role], where I primarily [briefly describe your responsibilities and contributions, such as developing test cases, automating tests, integrating with CI/CD pipelines, etc.]."

1. Do you automate tests everyday?

While I don't necessarily automate new test cases every single day, automation is an integral part of my daily activities. I spend my time writing new scripts, maintaining existing ones, and ensuring our automation suite runs effectively. The specific tasks I focus on can vary depending on the project's current needs, but overall, automation is a constant and essential aspect of my role.

1. 