Scenario Based Test Automation Interview Questions

A script you wrote was working fine yesterday but is failing today. How do you troubleshoot?

- Check for Environment Changes: Ensure there have been no changes in the testing environment, such as browser updates, driver updates, or changes in the test data.

- Analyze Logs: Review the logs to identify where the failure occurred. Logs often provide clues about what went wrong.

- Inspect the Application: Check if there have been any updates or changes to the application itself, which could affect the locators or the workflow.

- Check Locators: Verify if any locators have changed or if elements have become dynamic. Use tools like the browser's developer tools to revalidate the locators.

- Debug the Code: Use breakpoints and debug the script to step through the code and identify the point of failure.

If a Test fails, what will be your next step?

- Re-run the Test: Sometimes, tests fail due to temporary issues. Re-running can help confirm if the failure is consistent.

- Analyze the Failure: Review the failure message and logs to understand the cause.

- Check for Flakiness: Determine if the test is flaky and fails intermittently. If so, investigate why.

- Fix the Issue: Based on the analysis, fix the identified issue in the script or report a defect if it's an application issue.

- Update the Test Case: If the application has changed, update the test case and the corresponding script.

If the application has minor changes, what would be your approach to modifying the Automation scripts?

- Impact Analysis: Identify which scripts are affected by the changes in the application.

- Update Locators: Modify the locators if elements have been moved or renamed.

- Refactor Code: If needed, refactor the code to handle new workflows or additional validations.

- Re-run Tests: After updating the scripts, run the tests to ensure they work correctly with the new changes.

How would you automate login functionality for a website?

- Locate Elements: Identify the username, password fields, and login button using locators (e.g., ID, XPath).

- Write the Script: Use Selenium WebDriver to interact with these elements.

- Example Code:

```
WebDriver driver = new ChromeDriver();
driver.get("http://example.com/login");
driver.findElement(By.id("username")).sendKeys("yourUsername");
driver.findElement(By.id("password")).sendKeys("yourPassword");
driver.findElement(By.id("loginButton")).click();
```

- Validate: Add assertions to verify successful login, such as checking for the presence of a logout button or a welcome message.

How would you automate a Test scenario where you need to check if an email is sent after a user registration?

- Register User: Automate the user registration process.

- Access Email: Use an email API or web interface to access the inbox.

- Validate Email: Verify the presence of the registration email.

- Example Code:

```
// Example with JavaMail API to check email
Properties properties = new Properties();
properties.put("mail.store.protocol", "imaps");
Session emailSession = Session.getDefaultInstance(properties);
Store store = emailSession.getStore("imaps");
```

```
store.connect("imap.gmail.com", "yourEmail@gmail.com", "yourPassword");

Folder inbox = store.getFolder("INBOX");
inbox.open(Folder.READ_ONLY);

Message[] messages = inbox.getMessages();
for (Message message : messages) {
    if (message.getSubject().contains("Registration Confirmation")) {
        // Email found
        System.out.println("Registration email received.");
    }
}
```

If there is a scenario that takes a long time to execute, would you prefer Manual Testing or Automation Testing? Why?

- Prefer Automation: For long-running scenarios, automation is preferred as it can run unattended, saving time and resources.
- Efficiency: Automation scripts can be executed multiple times without additional effort.
- Reliability: Automated tests reduce the risk of human error and provide consistent results.

How would you automate a scenario where you need to validate the contents of a downloaded file after clicking a button on a webpage?

- Trigger Download: Automate the click action to download the file.
- Wait for Download: Ensure the file is downloaded completely, possibly by checking the file system.
- Read File: Use appropriate libraries to read the file content.
- Example Code:

```
// Example with Apache POI for an Excel file
File file = new File("/path/to/downloaded/file.xlsx");
FileInputStream fis = new FileInputStream(file);
Workbook workbook = new XSSFWorkbook(fis);
Sheet sheet = workbook.getSheetAt(0);
Row row = sheet.getRow(0);
Cell cell = row.getCell(0);
String cellContent = cell.getStringCellValue();
assertEquals("Expected Content", cellContent);
```

How would you automate a scenario where you need to verify a specific color, font, and position of an element on a webpage?

- Locate Element: Identify the element using locators.
- Retrieve CSS Properties: Use Selenium to get the CSS properties.
- Example Code:

```
WebElement element = driver.findElement(By.id("elementId"));
String color = element.getCssValue("color");
String font = element.getCssValue("font-family");
Point position = element.getLocation();

assertEquals("expectedColor", color);
assertEquals("expectedFont", font);
assertEquals(new Point(expectedX, expectedY), position);
```

How would you handle pop-up windows or alert boxes in your Automation script?

- Switch to Alert: Use Selenium's `switchTo` method to handle alerts.

- Example Code:

```
Alert alert = driver.switchTo().alert();
String alertText = alert.getText();
alert.accept(); // or alert.dismiss();
```

- Handle Windows: Use window handles to switch between different browser windows.
- Example Code:

```
String mainWindowHandle = driver.getWindowHandle();
for (String handle : driver.getWindowHandles()) {
    driver.switchTo().window(handle);
}
driver.switchTo().window(mainWindowHandle);
```

How would you automate a scenario where you need to verify if a user is able to scroll down a webpage until the footer section is visible?

- Scroll Down: Use JavaScript to scroll the page.
- Example Code:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("window.scrollTo(0, document.body.scrollHeight);");
```

- Check Footer Visibility: Verify if the footer is displayed.
- Example Code:

```
WebElement footer = driver.findElement(By.id("footer"));
assertTrue(footer.isDisplayed());
```

You've been asked to automate a legacy application. What is your approach?

- Understand the Application: Study the application's workflow and technology stack.

- Identify Challenges: Determine potential challenges like lack of modern web elements or non-standard interfaces.

- Choose Tools: Select appropriate tools that can handle the legacy technologies.

- Incremental Automation: Start with small, high-impact areas and gradually expand the automation coverage.

You are asked to automate a functionality that is not yet fully developed. How do you handle this?

- Mock Data: Use mock data and services to simulate the incomplete functionality.

- Design Flexible Scripts: Write scripts that can be easily modified as the functionality evolves.

- Collaboration: Work closely with developers to understand the expected behavior and changes.

Your Automation scripts are running slowly. How can you improve the speed?

- Optimize Locators: Use efficient locators like IDs instead of complex XPaths.

- Reduce Waits: Minimize the use of `Thread.sleep` and use explicit waits where necessary.

- Parallel Execution: Run tests in parallel using tools like TestNG or JUnit.

- Headless Browsers: Use headless browsers like Chrome or Firefox to speed up execution.

Your Automation script is failing due to a change in the application. How do you handle this?

- Update Locators: Modify the locators to reflect the changes in the application.

- Refactor Code: Adjust the script to accommodate new workflows or element properties.

- Run Regression Tests: After making updates, run regression tests to ensure no other scripts are affected.

- Version Control: Use version control to track changes and rollback if necessary.