Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

CAPGEMINI AUTOMATION INTERVIEW QUESTIONS: (4+ YOE)

1. Describe how you would set up a Jenkins pipeline for test automation. Explain stages for building, testing, and deployment.

Ans: To set up a Jenkins pipeline for test automation, navigate to Jenkins & create a new pipeline job.
In the pipeline configuration, define stages such as 'Build', 'Test', & 'Deploy'.
In 'Build' stage, specify commands to compile the code and generate artifacts.
In 'Test' stage, execute test scripts & tools like Selenium or JUnit to run automated tests.
Finally, in the 'Deploy' stage, deploy the tested artifacts to the desired environment using tools like Maven or Docker.

2. How would you integrate Jenkins with a version control system like Git? Give an example of configuring Jenkins to automatically trigger a build on new commits.

Ans: To integrate Jenkins with Git, install the Git plugin in Jenkins and configure the repository URL.
In the Jenkins job configuration, set up a webhook or polling mechanism to automatically trigger builds on new commits.
For example, specify the Git repository URL in the Jenkins job configuration and enable the 'Poll SCM' option to trigger builds on new commits.

3. Demonstrate how you would create parameterized builds in Jenkins for test automation. For instance, configure a build to accept input for selecting specific test suites or environments.

Ans:
In Jenkins, create a parameterized job for test automation by adding parameters like 'TestSuite' or 'Environment'.
In the job configuration, define input parameters using the 'This build is parameterized' option.
For instance, create a 'Choice parameter' for selecting the test suite and an 'String parameter' for specifying the environment.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

4. Explain how to configure Jenkins to send email notifications after a build is complete. Give an example to setup email notifications for test automation builds.

Ans:
To configure email notifications in Jenkins, navigate to the job configuration and select the 'Add post-build action' option.
Then, choose the 'Editable Email Notification' option and specify the recipients' email addresses, configure the email subject and content templates according to the build status and results.

CAPGEMINI AUTOMATION INTERVIEW QUESTIONS: (3+ YOE)

5. What are the reasons automated scripts worked one night prior but not working today?

- System Updates: If there were updates or changes to the operating system, dependencies, or other software the script relies on, it could cause unexpected behaviour.
- Changes in Data or Environment: If the script interacts with external systems or databases, changes in the data or environment it operates in could cause it to fail. This could include changes in API responses, database schema changes, or network configurations.
- Permissions or Access: Changes in permissions or access rights to certain files, directories, or resources that the script needs to operate could cause it to fail.
- Timing or Scheduling: If the script is scheduled to run at specific times, there may have been changes to the scheduling system or conflicts with other tasks running at the same time.
- Dependency Issues: If the script relies on external libraries or dependencies, updates or changes to those dependencies could cause compatibility issues.
- Errors or Bugs: There might be errors or bugs in the script itself that went unnoticed during previous runs but are causing failures now.
- Configuration Changes: Changes in configuration files or settings that the script depends on could lead to unexpected behaviour.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

6. What is difference between get and navigate method?

The get () and navigate () methods in Selenium WebDriver are used to navigate to URLs, but they have slightly different purposes and functionality:

get () Method:

- The get() method is used to navigate to a new URL by opening it in the current browser window.
- It replaces the current page with the new URL.
- It's typically used to open the initial page or to navigate to a specific URL directly.
- Usage: driver.get(url)

navigate () Method:

- The navigate () method provides additional navigation functionalities beyond just loading a new URL.
- It allows for more granular control over the browser's navigation history.
- It can be used to navigate backward, forward, or to refresh the current page.

Usage:

- driver. navigate (). back (): Navigates to the previous page in the browsing history.
- driver. navigate ().forward (): Navigates to the next page in the browsing history, if available.
- driver. navigate (). refresh (): Refreshes the current page.

In summary, while both methods are used to navigate to URLs, get () is primarily for opening new URLs in the current window, while navigate () offers additional functionality for managing the browser's history and refreshing the page.

7. What is difference between close and quit method?

In Selenium WebDriver, both close () and quit () are methods used to close the browser window. However, they differ in their behaviour:

close () Method:

- The close () method is used to close the current browser window or tab that the WebDriver instance is currently controlling.
- If there is only one window open, close () will close the entire browser.
- If there are multiple windows open, calling close () will close the current window/tab but leave other windows open.
- It's useful when you want to close a specific window but keep the WebDriver session running for further operations.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Usage: driver.close()

quit () Method:

- The quit () method is used to exit the WebDriver session completely, closing all browser windows/tabs that the WebDriver instance is currently controlling.
- It terminates the WebDriver session and releases all associated resources, including browser processes.
- It's recommended to use quit () when you are done with the WebDriver session to ensure proper cleanup and resource management.

Usage: driver.quit()

In summary, close () is used to close the current browser window/tab, while quit () is used to close all browser windows/tabs and terminate the WebDriver session. It's good practice to use quit () to ensure proper clean up, especially in automated testing scenarios where you want to release resources efficiently.

8. How will you switch to 3rd child window in selenium java?

```
import java.util.Set;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;


public class WindowSwitchingExample {

main method

    // Open a URL that opens multiple windows/tabs

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

    // Click on a link/button that opens a new window (do this as many times as needed to open multiple windows)

    WebElement link = driver.findElement(By.linkText("Link Text"));

    link.click();
```

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

```
    // Get handles of all open windows
    Set<String> allWindows = driver.getWindowHandles();
    // Switch to the third window (index starts from 0)
    String[] windowHandles = allWindows.toArray(new String[0]);
    String thirdWindowHandle = windowHandles[2];
    driver.switchTo().window(thirdWindowHandle);
    // Now, you are switched to the third child window and can interact with its elements
    // For example:
    System.out.println(driver.getCurrentUrl());
    // After performing operations in the third window, you might want to switch back to the main window:
    // driver.switchTo().window(mainWindowHandle);
  }
}
```

9. How & where do you use collections in your Automation framework?

1. Storing WebElements: Collections like Array List or HashMap are often used to store WebElements located during the test execution. For example, a HashMap can store WebElement objects with keys representing their unique identifiers.

Code in Java

```
Map<String, WebElement> elementMap = new HashMap<>();
elementMap.put("usernameField", driver.findElement(By.id("username")));
```

2. **Managing Test Data**: Collections are used to store and manage test data, such as test inputs, expected outputs, or configuration parameters. This can include lists of test cases, test data sets, or configuration settings.

```
List<String> testData = new ArrayList<>();
```

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

```
testData.add("user1");

testData.add("password1");
```

3. **Handling Dynamic Data**: Collections are used to handle dynamic data encountered during test execution, such as dynamically generated IDs or lists of search results. This data can be stored in lists or maps for later use.

```
List<String> searchResults = new ArrayList<>();

for (WebElement result : searchResultsElements) {

   searchResults.add(result.getText());

}
```

4. Storing Test Results: Collections are used to store test results, including pass/fail statuses, error messages, or screenshots captured during test execution. This data can be stored in lists, maps, or custom objects.

```
Map<String, String> testResults = new LinkedHashMap<>();

testResults.put("Test1", "Pass");

testResults.put("Test2", "Fail - Element not found");
```

5. **Browser Navigation History**: Collections like Stack or LinkedList can be used to manage the browser's navigation history, allowing for easy navigation back and forth between pages during test execution.

```
Stack<String> navigationHistory = new Stack<>();

navigationHistory.push(driver.getCurrentUrl());
```

10. If ID keep changing id = ab123 and next id=ab234 and next id=ab456 then how will you find locator?

If part of the ID remains consistent, you can use partial matching in CSS selectors or XPath expressions to target the element. This is particularly useful if a portion of the ID remains constant.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

WebElement element = driver.findElement(By.cssSelector("[id^='partialID']")); // Starts with

Craft XPath expressions that dynamically select elements based on their changing attributes. You can use functions like contains(), starts-with(), or ends-with() to create flexible XPath expressions.

WebElement element = driver.findElement(By.xpath("//*[contains(@id, 'partialID')]"));

11. Which exceptions you frequently faced while automation?

- NoSuchElementException: This exception occurs when the WebDriver is unable to locate an element on the web page using the specified locator. It can happen if the element does not exist, is not visible, or if the locator used is incorrect.
- TimeoutException: This exception occurs when a command takes longer than the timeout specified for that command. For example, it might occur if WebDriver is unable to find an element within the specified timeout period.
- StaleElementReferenceException: This exception occurs when an element is no longer attached to the DOM (Document Object Model), usually because it has been refreshed or modified since it was located. This can happen if the page content changes dynamically.
- ElementNotInteractableException: This exception occurs when an element is present in the DOM but is not in a state where it can be interacted with. For example, it might occur if the element is disabled, hidden, or overlapped by another element.
- InvalidSelectorException: This exception occurs when the WebDriver encounters an invalid or incorrect selector syntax in the locator used to find an element.
- WebDriverException: This is a general exception that indicates an unexpected error has occurred in the WebDriver. It can be caused by various factors such as browser issues, network problems, or issues with the WebDriver itself.
- ElementClickInterceptedException: This exception occurs when the element is not clickable at the point where the click action is performed. This can happen if another element is covering the target element or if the element's position has changed.
- UnhandledAlertException: This exception occurs when WebDriver encounters an unexpected alert dialog on the web page that it cannot handle. It typically happens when trying to interact with an element that triggers an alert.

12. Write code for capture Screenshot.

- Use the TakesScreenshot interface to cast the WebDriver instance and capture the screenshot using getScreenshotAs(OutputType.FILE).
- Save the screenshot to a file using FileUtils.copyFile().

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

- Close the browser using driver.quit().

```
import org.apache.commons.io.FileUtils;
      // Capture screenshot and save to a file
      try {
         File screenshotFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
         FileUtils.copyFile(screenshotFile, new File("screenshot.png"));
         System.out.println("Screenshot captured successfully!");
      } catch (IOException e) {
         System.out.println("Failed to capture screenshot: " + e.getMessage());
      }
      // Close the browser
      driver.quit();
   }
}
```

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

13. How to read excel file in selenium?

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class ReadExcelFile {
    public static void main(String[] args) {
        try {
            // Specify the path to the Excel file
            FileInputStream file = new FileInputStream(new File("path_to_excel_file.xlsx"));
            // Create a workbook instance for the Excel file
            XSSFWorkbook workbook = new XSSFWorkbook(file);
            // Get the first sheet in the workbook
            XSSFSheet sheet = workbook.getSheetAt(0);
            // Iterate through each row in the sheet
            for (Row row : sheet) {
                // Iterate through each cell in the row
                for (Cell cell : row) {
                    // Read the cell value and print it
                    System.out.print(cell.getStringCellValue() + "\t");
                }
```

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

```
            System.out.println(); // Move to the next line after printing values of all cells in the row
        }
        // Close the workbook
        workbook.close();
    } catch (IOException e) { e.printStackTrace()}}
```

14. How do you handle AJAX calls in selenium?

Handling AJAX calls in Selenium involves ensuring that the WebDriver waits for the AJAX requests to complete before interacting with the elements that are affected by those requests. You can achieve this using explicit waits, or JavaScriptExecutor. Here's how you can handle AJAX calls using each method:

Implicit Waits:

```
WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("https://example.com");
```

Explicit Waits:

```
WebDriver driver = new ChromeDriver();
WebDriverWait wait = new WebDriverWait(driver, 10);
driver.get("https://example.com");
wait.until(ExpectedConditions.presenceOfElementLocated(By.id("elementId")));
```

Javascript Executor

```
WebDriver driver = new ChromeDriver();
driver.get("https://example.com");
JavascriptExecutor js = (JavascriptExecutor)driver;
// Wait for jQuery AJAX to complete
js.executeScript("return jQuery.active == 0");
// or wait for generic AJAX calls
js.executeScript("return document.readyState").equals("complete");
```

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

15. How do you handle test failures and errors in TestNG?

Answer: TestNG provides various listeners to handle test failures and errors. For example, the IInvokedMethodListener interface allows you to implement custom logic when a test method fails or encounters an error.

16: How do you run specific test methods or groups using the TestNG XML file?

Answer: In the TestNG XML file, you can define <test> tags and specify the test methods or groups to be executed using the <classes> or <methods> tags. You can also include or exclude specific test methods or groups.

17: How do you run TestNG tests from the command line?

Answer: You can run TestNG tests from the command line using the java command with the TestNG JAR file and the testng.xml file as arguments. For example: java -cp "testng.jar" org.testng.TestNG testng.xml

18: How do you configure TestNG to run tests in a specific order?

Answer: By default, TestNG runs test methods in no particular order. However, you can specify the execution order using the preserve-order attribute in the <test> tag in the TestNG XML file.

19. How do you handle test data setup and teardown in TestNG?

Answer: TestNG provides annotations like @BeforeClass, @AfterClass, @BeforeSuite, and @AfterSuite to handle test data setup and teardown. These annotations allow you to run setup and teardown methods before and after test classes or test suites.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

ADP, S&P Global Automation and Agile Interview Questions:

20.Difference between Selenium and Cypress in-terms of architecture and which one do you recommend according to your opinions

Selenium Architecture:

- Selenium is a suite of tools, with Selenium WebDriver being the most commonly used component for browser automation.
- It operates by directly controlling the browser using browser-specific drivers (e.g., ChromeDriver, GeckoDriver) through a client-server architecture.
- Selenium WebDriver communicates with the browser using a browser-specific protocol (e.g., JSONWireProtocol for most browsers, Chrome DevTools Protocol for Chrome).

Cypress Architecture:

- Cypress is a JavaScript-based end-to-end testing framework built specifically for modern web applications.
- It operates within the same browser as the application under test, using a combination of JavaScript and browser APIs to control the application and make assertions.
- Cypress runs within the same JavaScript event loop as the application, enabling direct access to the DOM and real-time debugging.

Recommendation:

- Choosing between Selenium and Cypress depends on various factors, including the nature of your project, your team's expertise, and your testing requirements:

- Selenium: Recommended for projects requiring cross-browser testing, support for multiple programming languages, or testing across different domains and technologies. It's a versatile tool suitable for a wide range of web testing scenarios.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

- Cypress: Recommended for projects with modern web applications built using JavaScript frameworks like React, Angular, or Vue.js. Cypress offers a more streamlined and developer-friendly experience, with features tailored specifically for modern web development workflows.

21.Difference between monolithic and microservices architecture

Monolithic and microservices architectures are two different approaches to designing and implementing software systems. Here's a comparison between the two:

Monolithic Architecture:

Overview:

- In a monolithic architecture, the entire application is designed as a single, self-contained unit.
- All components of the application, including the user interface, business logic, and data access layers, are tightly integrated and deployed as a single unit.
- Monolithic applications typically have a single codebase and are developed, tested, and deployed as a single entity.

Characteristics:

- Tight coupling: Components are tightly integrated, making it challenging to modify or scale individual parts of the application independently.
- Single deployment unit: The entire application is deployed as a single unit, requiring downtime for updates or changes.
- Limited scalability: Scaling the application involves scaling the entire monolith, which may not be efficient for all components.
- Technology stack: Typically, monolithic applications use a single technology stack for all components.

Advantages:

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

- Simplicity: Monolithic architectures are often simpler to develop, deploy, and manage compared to microservices.
- Easier debugging: With all components running within the same process, debugging and troubleshooting can be easier.

Disadvantages:

- Lack of flexibility: Changes to one part of the application may require rebuilding and redeploying the entire monolith, leading to slower release cycles.
- Scalability challenges: Scaling individual components independently can be challenging, leading to inefficient resource usage.
- Limited technology choices: Since the entire application uses the same technology stack, it may not be easy to adopt new technologies or languages.

Microservices Architecture:

Overview:

- In a microservices architecture, the application is decomposed into a collection of loosely coupled, independently deployable services.
- Each service is responsible for a specific business capability and can be developed, deployed, and scaled independently.
- Services communicate with each other through well-defined APIs, often using lightweight protocols such as HTTP or messaging queues.

Characteristics:

- Loose coupling: Services are loosely coupled, allowing them to be developed, deployed, and scaled independently.
- Independent deployment: Each service can be deployed independently, enabling faster release cycles and reduced downtime.
- Polyglot architecture: Services can be implemented using different programming languages, frameworks, or databases based on their specific requirements.
- Scalability: Individual services can be scaled independently based on demand, leading to more efficient resource utilization.

Advantages:

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

- Scalability: Microservices architecture allows for better scalability by scaling individual services independently based on demand.
- Flexibility: Services can be developed, deployed, and scaled independently, enabling faster innovation and shorter release cycles.
- Technology diversity: Each service can use the most appropriate technology stack for its requirements, allowing for greater flexibility and innovation.

Disadvantages:

- Complexity: Microservices architectures can introduce complexity in terms of managing inter-service communication, deployment, and monitoring.
- Distributed systems challenges: Distributed systems introduce challenges such as network latency, data consistency, and service discovery, which need to be addressed.
- Operational overhead: Managing a large number of services and their dependencies can increase operational overhead compared to monolithic architectures.

Recommendation:

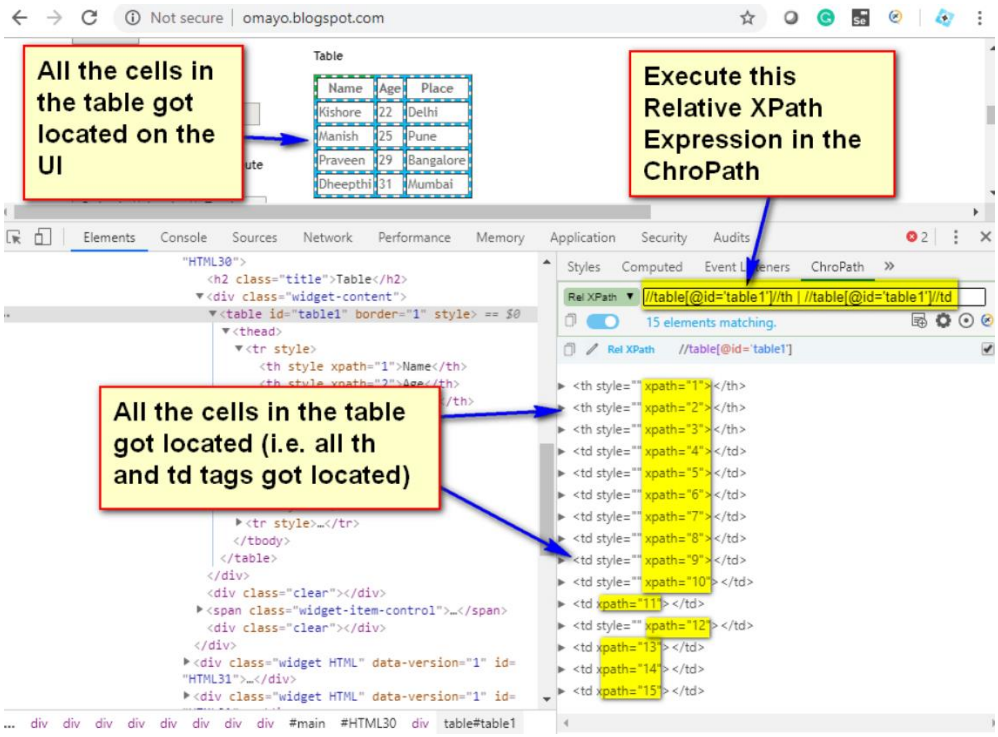
- The choice between monolithic and microservices architectures depends on factors such as the complexity of the application, scalability requirements, team expertise, and organizational goals:
- Monolithic architecture may be suitable for smaller applications with simpler requirements, where simplicity and ease of development are more important than scalability and flexibility.
- Microservices architecture is recommended for larger, complex applications with evolving requirements, where scalability, flexibility, and the ability to innovate quickly are critical.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

22. Find xpath of a particular table cell



Note: Q23 to Q30 are cross question type which subscriber got

23. Let's start with the framework you've worked on. Could you tell me about it?

You: Certainly. I've primarily worked on a modular-driven automation framework using Java, TestNG, and Selenium WebDriver. The aim was to create a scalable, maintainable, and easily understandable framework for both technical and non-technical team members.
*You can add more details about the framework.

24. Interviewer: Interesting. In such frameworks, challenges are inevitable. Can you share a specific challenge you faced and how you overcame it?

You: One notable challenge was dealing with dynamic elements, which frequently changed their attributes. To tackle this, we implemented dynamic waits using WebDriverWait along with ExpectedConditions to wait for elements until they stabilized.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

25. Interviewer: Handling dynamic elements is crucial indeed. Apart from Selenium's built-in methods, how did you handle dynamic variables?

You: Besides relying on Selenium's methods, we incorporated custom logic. This involved using regular expressions, string manipulation, and dynamic XPath/CSS selectors to locate elements based on changing attributes or text.

26. Interviewer: Moving on to browser tabs, how did you handle multiple tabs in Selenium?

You: For handling multiple tabs, we utilized WebDriver's window handles. By using getWindowHandles() to get handles of all open windows, we could then switch between tabs using the switchTo() method.

27. Interviewer: Good. Now, which interface or class does getWindowHandles() belong to?

You: getWindowHandles() belongs to the WebDriver interface in Selenium.

28. Interviewer: There are multiple products in a web table, like on Amazon. How would you approach selecting one product and adding it to the cart?

You: First, we'd locate the web table containing products using its unique identifier. Then, by iterating through the rows and columns, we'd identify the desired product based on specific criteria such as product name or ID.

29. Interviewer: Great approach. Lastly, could you explain the difference between findElement and findElements methods in Selenium?

You: findElement -> Returns matching web element found in the DOM
It will throw exception when it is not able to find/locate the webelement
and findElements -> Returns all the List of elements found that matches with the given selector in the DOM
It wont throw exception rather it will return empty list.

30. Interviewer: And what about the StaleElementReferenceException?

You: A StaleElementReferenceException occurs when the element is no longer attached to the DOM, often due to a page refresh or navigation. For instance, if an element is located and stored, but the page changes, attempting to interact with the stored element will

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

trigger this exception. To overcome it, we typically re-locate the element or refresh the page before interacting with it again.

31. Difference between git pull and git fetch

git pull:

Behaviour: git pull is a combination of two operations: git fetch followed by git merge. It retrieves changes from the remote repository and integrates them into the current branch.

Usage:

git pull [<remote> [<branch>]]

Effect: It fetches the changes from the remote repository and automatically merges them into the current branch. If there are no conflicts, the changes are merged automatically. If conflicts occur, manual intervention may be required to resolve them.

Potential Risks: Since git pull automatically merges changes into the current branch, there is a risk of inadvertently introducing conflicts or unwanted changes if the local branch and remote branch have diverged significantly.

git fetch:

Behaviour: git fetch retrieves changes from the remote repository and stores them in the local repository without merging them into any branch.

Usage:

git fetch [<remote> [<refspec>…]]

Effect: It updates the remote tracking branches in the local repository to reflect the changes on the remote repository. However, it does not merge these changes into the current branch.

Advantages: git fetch provides a safer way to inspect changes fetched from the remote repository before merging them into the local branch. It allows you to review changes and resolve conflicts manually if necessary.

Common Usage: After running git fetch, you can inspect the changes using commands such as git log, git diff, or gitk, and then decide whether to merge them using git merge or git rebase.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Recommendation:

Use git pull when: You want to quickly fetch changes from the remote repository and automatically merge them into the current branch. This is suitable for routine updates where you trust the remote repository and want to synchronize your local branch with it.

Use git fetch when: You want to fetch changes from the remote repository without automatically merging them into the current branch. This is useful when you want to review the changes, inspect them, or resolve conflicts manually before merging them into your branch. It provides a safer approach, especially when working with diverged branches or collaborating with others.

32. What is definition of done in Agile

The "Definition of Done" (DoD) in Agile is a set of criteria or checklist items that a product increment must meet for it to be considered complete and ready for release. It serves as a shared understanding within the Agile team of what it means for work to be done and ensures that each increment delivered is of high quality and meets the acceptance criteria. The DoD is typically defined collaboratively by the Agile team, including developers, testers, product owners, and other stakeholders.

- The specific items included in the Definition of Done may vary depending on the project, team, and organization, but it often includes aspects such as:
- Functional requirements: All user stories or features are implemented according to the agreed-upon acceptance criteria and meet the specified functionality requirements.
- Quality assurance: The product increment has been thoroughly tested, including unit tests, integration tests, and acceptance tests, and meets the team's quality standards.
- Documentation: Any necessary documentation, such as user manuals, technical documentation, or release notes, is completed and up-to-date.
- Code review: The code has been reviewed by peers or through a formal code review process to ensure it adheres to coding standards, is maintainable, and does not introduce technical debt.
- Performance and scalability: The product increment meet performance and scalability requirements, and any performance-related issues have been addressed.
- User acceptance: The product increment has been demonstrated to stakeholders or end-users, and any feedback has been addressed.
- Integration and deployment: The increment has been integrated with the main codebase, and any necessary deployment steps have been completed to make it available for use.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

- Regulatory compliance: If applicable, the product increment complies with relevant regulations, standards, or industry best practices.
- The Definition of Done should be reviewed and updated regularly to reflect changes in the project or team's requirements, processes, or quality standards. It helps ensure transparency, alignment, and accountability within the Agile team and promotes the delivery of high-quality, value-added increments to customers or stakeholders.

33. Do you think Agile model is really helpful in sprint basis?

- Yes, Agile methodology, particularly when implemented using sprint-based iterations, can be highly beneficial for many software development projects. Here are several reasons why Agile is helpful in sprint basis:
- Iterative and Incremental Development: Agile emphasizes delivering working software in small, incremental releases. Sprints provide short, time-boxed iterations where the team can focus on delivering a potentially shippable product increment. This iterative approach allows for continuous feedback and adaptation throughout the development process.
- Flexibility and Adaptability: Agile methodologies, including Scrum and Kanban, are designed to be flexible and adaptive to changing requirements and priorities. Sprints provide a framework for regularly reassessing priorities, adapting to feedback, and making course corrections as needed.
- Transparency and Visibility: Sprints promote transparency by making progress visible through regular sprint planning, daily stand-up meetings, and sprint reviews. This transparency helps stakeholders understand the status of the project, identify any potential issues early, and make informed decisions.
- Risk Reduction: By breaking down the project into smaller, manageable chunks, Agile helps mitigate risk by allowing the team to identify and address issues early in the development process. Sprints provide frequent opportunities for validation and feedback, reducing the risk of delivering a product that does not meet customer expectations.

34. What are different assertions used in Cypress?

1. .should('exist'): Asserts that the element exists in the DOM.

   Code: ***cy.get('button').should('exist');***

2. should('not.exist'): Asserts that the element does not exist in the DOM.

   Code: ***cy.get('#nonexistent-element').should('not.exist');***

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

3. .should('be.visible'): Asserts that the element is visible.

Code: ***cy.get('.visible-element').should('be.visible');***

4. .should('not.be.visible'): Asserts that the element is not visible.

Code: ***cy.get('.hidden-element').should('not.be.visible');***

5. .should('be.enabled'): Asserts that the element is enabled (applicable to input fields, buttons, etc.).

Code: ***cy.get('input').should('be.enabled');***

1. .should('be.disabled'): Asserts that the element is disabled.

Code: ***cy.get('button').should('be.disabled');***

7.should('have.value', expectedValue): Asserts that the input field has the expected value.

Code: ***cy.get('#input-field').should('have.value', 'expected value');***

8.should('contain', expectedText): Asserts that the element contains the expected text.

Code: ***cy.get('div').should('contain', 'expected text');***

9 .should('have.attr', attributeName, expectedValue): Asserts that the element has the expected attribute with the specified value.

Code: ***cy.get('a').should('have.attr', 'href', '/home');***

10 .should('have.css', propertyName, expectedValue): Asserts that the element has the expected CSS property with the specified value.

Code: ***cy.get('button').should('have.css', 'background-color', 'rgb(255, 0, 0)');***

11. .should('match', regex): Asserts that the element's text matches the provided regular expression.

Code: ***cy.get('span').should('match', /pattern/);***

35. Explain cypress folder structure in detail

In Cypress, the folder structure is organized in a specific way to facilitate the creation and execution of tests. Here's a detailed explanation of the typical folder structure in a Cypress project:

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

cypress.json:

This is the main configuration file for Cypress. It contains settings that control how Cypress runs, such as the base URL, browser options, test retries, and more.

cypress/fixtures/:

This folder is used to store static fixture data that can be used in tests. Fixture files can contain JSON, CSV, or any other format of test data that your tests may need.

cypress/integration/:

- This is where you'll place your test files. Cypress automatically scans this folder for test files to execute.
- Test files are typically written using the Mocha test framework syntax and should have the .spec.js or .test.js file extension.
- You can organize your tests into subfolders within the integration folder to keep them organized.

cypress/plugins/index.js:

This file is used to extend Cypress's functionality through plugins. Plugins can be used to perform tasks such as customizing test runs, intercepting network requests, or integrating with other tools.

cypress/support/:

- This folder contains support files that Cypress loads before running your tests. These files can be used to define custom commands, set up test data, or configure Cypress behavior.
- commands.js: You can define custom commands here to encapsulate common actions or assertions that you use across multiple tests.
- index.js: This file is used to import and configure any additional Cypress plugins or libraries that you want to use in your tests.

cypress/screenshots/ and cypress/videos/:

- These folders are automatically created by Cypress to store screenshots and videos captured during test runs.
- Screenshots are saved here when tests fail, providing visual feedback on the state of the application at the time of failure.
- Videos capture the entire test execution process and can be useful for debugging and analyzing test failures.

node_modules/ and package.json:

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

- These folders and files are part of the Node.js ecosystem and are used to manage project dependencies.
- package.json contains metadata about the project and a list of dependencies required by Cypress and other packages used in the project.
- node_modules contains the installed packages and their dependencies.

cypress.env.json (Optional):

- This file is used to define environment variables that can be accessed in your tests.
- Environment variables can be useful for configuring tests to run against different environments (e.g., development, staging, production) or for storing sensitive data such as API keys or credentials.
- This folder structure provides a clear organization for your Cypress tests and support files, making it easy to write, manage, and execute tests for your web applications.

Broadridge Company (Onsite round) for 4+ YOE

36. Explain the agile process you followed

Answer: Here's an overview of the Agile process:

Project Initiation:

The Agile process begins with project initiation, where the project goals, scope, and requirements are defined. This may involve creating a project vision, identifying key stakeholders, and establishing the initial backlog of work.

Iteration Planning:

Agile projects are organized into iterations, often called sprints, which are time-boxed periods (typically 1-4 weeks) during which a set of work is completed. Before each sprint, the team conducts an iteration planning meeting to select and prioritize the items from the backlog to be completed during the sprint.

Daily Stand-up Meetings:

Throughout the sprint, the team holds daily stand-up meetings, also known as daily scrums, to review progress, discuss any obstacles or issues, and plan the day's work. These meetings are typically short (15 minutes or less) and help keep the team aligned and focused.

Development and Testing:

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

During the sprint, the development team works on implementing the features and functionality identified in the sprint backlog. Developers write code, testers write and execute tests, and designers create user interfaces. Continuous integration and automated testing are often used to ensure that changes are integrated and tested frequently.

Demo and Review:

At the end of each sprint, the team holds a sprint review meeting to demonstrate the completed work to stakeholders and gather feedback. This allows stakeholders to see the progress made during the sprint and provide input on any changes or adjustments needed.

Retrospective:

Following the sprint review, the team holds a retrospective meeting to reflect on the sprint process and identify areas for improvement. This includes discussing what went well, what didn't go well, and what actions can be taken to improve in the next sprint.

Backlog Refinement:

Throughout the project, the product backlog is continuously refined and updated based on feedback from stakeholders, changes in priorities, and new insights gained during development. This ensures that the backlog remains relevant and reflects the current state of the project.

Continuous Delivery:

Agile teams aim to deliver working software frequently, typically at the end of each sprint. This allows stakeholders to see tangible progress, provide feedback, and make adjustments as needed. Continuous integration and deployment practices are often used to automate the process of building, testing, and deploying software.

37. If TL asking for a report and you are working on high priority task, what will be your response?

Answer: **Assess the Urgency and Importance:** I would first assess the urgency and importance of the report requested by the TL. If the report is critical for decision-making or has a tight deadline, it may need immediate attention.

- **Communicate Priorities:** I would communicate my current workload and priorities to the TL, explaining the importance and urgency of the high-priority task I am working on. It's important to provide transparency and context to help the TL understand the situation.
- **Negotiate and Offer Alternatives:** I would negotiate with the TL to see if there is flexibility in the deadline for the report or if there are alternative solutions that could

meet their needs. For example, I might suggest providing an interim update or summary instead of the full report, or delegating the task to another team member if feasible.

38. Sequence of git commands to resolve merge conflict

Here's the typical sequence of Git commands to resolve a merge conflict:

1. **Identify the Conflict:** When you attempt to merge branches and Git encounters conflicting changes, it will notify you about the conflict. You'll see messages indicating which files have conflicts.
2. **Open the Conflicted File(s):** Use a text editor or integrated development environment (IDE) to open the conflicted file(s). Inside the file, you'll see markers indicating the conflicting sections, typically surrounded by <<<<<<<, =======, and >>>>>>>.
3. **Resolve the Conflict:** Manually edit the conflicted file(s) to resolve the conflicting changes. You'll need to decide which changes to keep, modify, or remove. Remove the conflict markers (<<<<<<<, =======, >>>>>>>) and make any necessary adjustments to reconcile the conflicting changes.
4. **Stage the Changes:** After resolving the conflict, stage the modified file(s) using the git add command. This marks the conflicts as resolved and prepares the file(s) for the next commit.

git add <conflicted_file>

5. **Complete the Merge:** Once all conflicts have been resolved and staged, complete the merge using the git commit command. This creates a new commit that records the merged changes and resolves the conflict.

git commit -m "Merge conflict resolution"

6. **Review the Commit Message:** Review the commit message to ensure it accurately describes the resolution of the merge conflict. Make any necessary adjustments to the commit message before finalizing the commit.
7. **Verify the Merge:** After committing the resolution, verify that the merge was successful by reviewing the repository's history and checking for any remaining conflicts.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

8. **Push the Changes (if applicable):** If you resolved the conflict on a branch that is tracked remotely, push the changes to the remote repository to update it with the resolved merge.

git push origin <branch_name>

Following these steps will help you effectively resolve merge conflicts in Git and ensure that your project's history remains clean and accurate. It's important to carefully review the changes and communicate with your team to ensure that conflicts are resolved correctly and any necessary adjustments are made to the codebase.

39. If few test cases failed and you have release next day, what will be course of action?

Answer: If some test cases have failed and there is a release scheduled for the next day, it's essential to take prompt action to address the failures while ensuring that the release remains on track. Here's a suggested course of action:

Identify the Root Cause: Investigate the reasons for the test failures to understand the root cause. Determine whether the failures are due to changes in the application code, environmental factors, data dependencies, or other issues.

Prioritize Fixes: Prioritize the test cases that have failed based on their impact on the application's functionality, criticality, and likelihood of causing issues in the release.

Focus on fixing high-priority test failures first to ensure that critical functionality is working as expected.

Fix the Failures: Once you've identified the root cause of the failures, work on fixing them promptly. This may involve debugging the code, updating test scripts, adjusting test data, or addressing environmental issues.

Collaborate with developers, testers, and other team members as needed to resolve the failures efficiently.

Re-run Tests: After fixing the failed test cases, re-run the affected tests to verify that the issues have been addressed and that the tests now pass successfully.

It's important to ensure that the fixes have not introduced regressions or caused other unintended side effects.

Review and Validate: Review the fixes and validate the changes to ensure that they address the root cause of the test failures effectively.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Conduct thorough testing, including regression testing, to verify that the application behaves as expected after the fixes have been applied.

Communicate Status: Keep stakeholders, including project managers, product owners, and other team members, informed about the status of the test fixes and the impact on the release schedule.

Provide regular updates on the progress of resolving the test failures and any adjustments to the release plan.

Mitigation Strategies: If it's not possible to fix all test failures before the release deadline, consider implementing mitigation strategies to minimize risk.

This may involve temporarily disabling or skipping affected tests, providing workarounds for known issues, or adjusting the release scope if necessary.

Release Decision: Based on the progress of resolving the test failures and the overall stability of the application, make an informed decision about whether to proceed with the release as scheduled or to postpone it.

40. Under what situation will you contact your team lead?

Answer. **Urgent Issues or Emergencies:** If you encounter urgent issues or emergencies that require immediate attention and resolution, such as critical system failures, security breaches, or data loss, contacting your team lead is necessary to escalate the situation and coordinate a response.

Major Roadblocks or Impediments: When you encounter major roadblocks or impediments that hinder progress on your tasks or prevent you from meeting project deadlines, it's important to inform your team lead promptly. They can provide support, allocate resources, or help remove obstacles to keep the project on track.

Need for Guidance or Clarification: If you have questions, uncertainties, or require clarification on project requirements, priorities, or expectations, reaching out to your team lead is beneficial. They can provide guidance, offer insights, or provide context to help you make informed decisions and proceed with your work effectively.

Conflict Resolution: In situations where conflicts arise within the team, disagreements occur, or interpersonal issues emerge, involving your team lead can help facilitate resolution, mediate discussions, and foster a collaborative and harmonious work environment.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Performance or Progress Updates:

Providing regular updates on your work progress, accomplishments, and challenges to your team lead helps maintain transparency, enables alignment with project goals, and allows for proactive management of expectations and resources.

Need for Resources or Support:

If you require additional resources, tools, or support to accomplish your tasks effectively, informing your team lead is essential. They can allocate resources, provide assistance, or escalate requests as needed to ensure that you have the necessary support to succeed.

Feedback or Suggestions:

Offering feedback, suggestions, or insights to your team lead about process improvements, workflow optimizations, or areas for enhancement can contribute to the overall success of the project and foster a culture of continuous improvement and innovation.

Compliance or Ethical Concerns:

If you encounter situations that raise compliance issues, ethical concerns, or potential violations of company policies or regulations, notifying your team lead is crucial to address the issue promptly, mitigate risks, and uphold organizational integrity and values.

41. What is the reporting structure you use in your framework?

Answer: We are using extent reports and its configured within a BaseTest or MainTest class file using which whenever a test is executed or failed or skipped can be captured. We have setup a ThreadPool for the extent reports object which even helps in case of parallel execution.

We are also recording screenshots and logs if required within our child Test classes. This one of the easiest approaches to implement and scale the reporting within our frameworks. I have also implemented TestNG listeners known ITestListener to trigger the reports but it is less customisable as compared to the above way!

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Nalashaa Solutions Interview Questions (4+ YOE)

First round:

42. Authorization vs Authentication

Answer: Authorization and authentication are two crucial concepts in the field of security, particularly in the context of access control and user identity verification:

Authentication:

- Authentication is the process of verifying the identity of a user or system entity attempting to access a resource or service.
- It typically involves presenting credentials, such as a username and password, biometric data (e.g., fingerprint, facial recognition), digital certificates, or token-based authentication (e.g., one-time passwords).
- The primary goal of authentication is to ensure that the entity claiming a particular identity is indeed who they say they are.
- Once authenticated, the user is granted access to the system or application based on their level of authorization.

Authorization:

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

- Authorization, also known as access control, is the process of determining what actions or resources a user or system entity is permitted to access or perform after successful authentication.
- It involves defining access rights, permissions, and privileges based on the identity and attributes of the authenticated user or entity.
- Authorization mechanisms enforce policies that govern who can access specific resources, what actions they can perform, and under what conditions.
- Authorization controls may be based on various factors, including user roles, groups, attributes, or contextual information.

45. There is an application with payment option, what are the different types of testing you will perform. Explain different scenarios

Testing an application with a payment option requires thorough validation to ensure security, functionality, and user experience meet expected standards. Here are several types of testing that should be performed:

Functional Testing: Payment Process Testing: Verify that users can successfully initiate payments, enter payment details, select payment methods, and complete transactions without errors.

Error Handling: Test how the application responds to various error scenarios during payment processing, such as invalid card details, network issues, or server errors.

Integration Testing: Ensure seamless integration between the application and payment gateways, APIs, or third-party services involved in payment processing.

Security Testing: Payment Security: Assess the security measures implemented to protect sensitive payment information, including encryption, secure connections (HTTPS), and compliance with Payment Card Industry Data Security Standard (PCI DSS) requirements.

Authentication and Authorization: Test the authentication and authorization mechanisms to prevent unauthorized access to payment-related functionalities and data.

Usability Testing: User Interface (UI) Usability: Evaluate the clarity, intuitiveness, and accessibility of the payment interface, including the layout of payment forms, error messages, and feedback provided to users during the payment process.

Cross-Device Compatibility: Ensure that the payment functionality works seamlessly across different devices (desktops, tablets, smartphones) and screen sizes.

Performance Testing:

Load Testing: Assess how the application handles concurrent users and transaction loads during peak periods, ensuring that it remains responsive and stable under heavy traffic.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Response Time: Measure the time taken for payment transactions to process, including authorization, verification, and confirmation, to ensure optimal performance.

Regression Testing:

Payment Flow Regression: Validate that recent changes or updates to the application have not introduced regressions in the payment process, ensuring that existing payment functionalities remain unaffected.

Compatibility Testing:

Browser Compatibility: Verify that the payment functionality works correctly across different web browsers (e.g., Chrome, Firefox, Safari, Edge) and browser versions.

Operating System Compatibility: Ensure that the payment process functions properly on various operating systems (e.g., Windows, macOS, iOS, Android).

Accessibility Testing:

Accessibility Compliance: Evaluate whether the payment interface complies with accessibility standards (e.g., WCAG) to ensure that users with disabilities can navigate and complete payments effectively.

46. What is difference between Regression & Retesting , Regression testing vs Retesting

Answer: Regression testing and retesting are both important aspects of software testing, but they serve different purposes:

Regression Testing:

- Regression testing is performed to ensure that recent code changes or modifications in a software application have not adversely affected existing features.
- It involves re-executing test cases that cover the unchanged parts of the software, as well as testing the modified or added features to ensure that no new defects have been introduced and that existing functionality remains intact.
- Regression testing helps in maintaining the stability and reliability of the software by catching any unexpected side effects of code changes.

Retesting:

- Retesting, on the other hand, is specifically focused on verifying that defects identified in earlier testing cycles have been successfully fixed.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

- When a defect is found during initial testing, the development team fixes it, and then the tester verifies that particular defect fix by retesting the affected functionality.
- Retesting ensures that the reported issues have been resolved and that the affected part of the software now functions correctly.

47. How to perform parallel testing in TestNG with proper syntax?

Answer. Parallel testing in TestNG can be achieved by setting the parallel attribute to one of its values like tests, methods, or classes in the <test> tag. Here's an example syntax:

xml

<suite name="Parallel Test Suite" parallel="tests">

<test name="Test 1"> <!-- Test configurations and classes --> </test>

<test name="Test 2"> <!-- Test configurations and classes --> </test> </suite>

48. How do you validate the test report in TestNG?

Answer. TestNG generates detailed test reports in HTML format by default. To validate the test report, we can look for elements like pass/fail indicators, test execution time, and any error or exception messages in the report.

49. How to handle multiple window handles?

Answer. In Selenium, to handle multiple window handles, we can use the getWindowHandles() method to get a set of window handles. Then, we can switch between windows using methods like switchTo().window(handle).

50. Difference between throw encrypted exception and throw, Throwable?

Answer: throw is used to manually throw an exception in Java, while Throwable is the superclass of all errors and exceptions in Java. There isn't a concept of "encrypted exception" in Java, so I believe it might be a typo.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

51.For what purpose. getTagName() is used for in Selenium?

Answer: In Selenium, the. getTagName() method is used to retrieve the tag name of a WebElement. For example, if you have a WebElement representing a element, .getTagName() would return "div".

52. How to optimize verification of 50 links present in a web page?

Answer: We can optimize verification of 50 links by using efficient locators and looping through the links to verify their presence. Also, we can use parallel execution or threading to speed up the verification process.

53. Why do we use Git Stash?

Answer: Git Stash is used to temporarily store changes that are not ready to be committed. It's helpful when you need to switch branches or pull changes from a remote repository without committing your current changes.

54. How to resolve conflicts in a pull from the master branch?

Answer: To resolve conflicts in a pull from the master branch, we can use Git's merge or rebase functionality. We need to manually resolve conflicts in the files marked as conflicted by Git, then add the resolved files and commit the changes.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Solitera Software Interview Questions (3+ YOE)

55. What is error seeding?

Answer: Error seeding, also known as fault seeding, is a software testing technique used to intentionally inject defects, errors, or faults into a program's code or data to evaluate the effectiveness of the testing process. The purpose of error seeding is to assess the thoroughness and efficiency of testing procedures by measuring the ability of testers to detect and identify seeded errors.

Here's how error seeding typically works:

Identifying Potential Errors: Before testing begins, the development team or testers identify potential areas within the software where errors or defects may occur. These could be based on past experience, common software vulnerabilities, or specific requirements of the system.

Injecting Seeded Errors: Once potential error locations are identified, intentional defects or faults are introduced into the software code, data, or documentation. These seeded errors are strategically placed to simulate real-world issues that users might encounter.

Conducting Testing: Testers then execute various test cases, including functional, non-functional, and edge cases, to determine if they can identify and report the seeded errors. The goal is to assess the effectiveness of the testing process in detecting the injected faults.

Analysing Results: After testing is complete, the results are analysed to determine how many of the seeded errors were detected and reported by the testing team. This analysis provides insights into the thoroughness of testing efforts and helps identify areas for improvement in the testing process.

56. Difference in Bug-Error-Defect-Failure

Bug:

- A bug is a general term used to describe any flaw, mistake, or fault in a software application that causes it to behave unexpectedly or produce incorrect results.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

- Bugs can occur at any stage of the software development lifecycle, from design and coding to testing and deployment.
- When a bug is discovered, it typically needs to be reported, analysed, and fixed by developers to ensure the software functions correctly.

Error:

- An error refers to a mistake made by a human during the development or maintenance of software. It can be a misunderstanding of requirements, a typo in code, or a logic error in the implementation.
- Errors are unintentional and can lead to the introduction of defects or bugs in the software if they are not identified and corrected during the development process.

Defect:

- A defect is a variation between the expected and actual behaviour of a software application. It arises when there is an error in the code or design that causes the software to not perform as intended.
- Defects are typically discovered during testing, either by manual inspection or through automated testing tools.

Failure:

- A failure occurs when a software application does not meet its specified requirements or user expectations while running in a particular environment.
- Failures are observable and measurable instances of incorrect behaviour or malfunctioning in the software.

Experience Level = 2 to 7 years

Important list of questions which are tricky when asked during an interview. Understand the concept and then formulate your own answers!

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

57. What is MR?

Ans: Modification Request - when client/users ask for modification of an existing feature.

58. Why do we need negative testing?

Ans: If a user performs actions which are not as per a expected user flow - at that time we require Negative testing.

Negative Testing - tests a feature which its not supposed to do.

Eg: In a login page it expects - Username and Password for login process - But you as a tester Enter Email ID in username field - Then should it work?

59. What is End-to-End testing?

Ans: System testing can be also called as E2E testing, Now in this we basically test whole application by prioritizing the positive flows of an application and then negative flows.

60. How to we execute the following?

1) Regression Testing

2) System Integration Testing (when 2 or more different systems are integrated, conducted when required)

3) Sanity Testing

4) Smoke Testing

5) User acceptance testing

Ans: 4-3-1-2-5

61. What is Shift Right Testing?

Ans: When developers deploy build in QA/Staging/Pre-Prod env and then testers perform there testing.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

62. What is Shift Left Testing?

Ans: When testers take part in testing of newly developed features which are yet to be deployed to QA/Staging/Pre-Prod/Prod env.

63. What is Recovery Testing?

Ans: This is executed to tested how quickly system can recover if goes into crash/failure.

Who will perform this? Usually Devops.

64. What is A/B testing?

Ans: A means version-1 and B means version-2.

Then these versions are released into different market regions and tested by different end-users.

The version which has higher ratings are released world-wide.

Eg: Usually done when features are region specific.

65. What is Crowd-Source-Testing?

Ans: A completely unknown pool of testing resources test your application, you can judge the quality of your product on the basis of number of bugs reported.

66. What is Out-Source-Testing?

Ans: A dedicated team is present to handle your testing needs we can say it’s a third party which is unknown to you, test your application or product with a fresh set of mind.

Can also be called in some cases as Beta testing.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

67. Why do we need API testing?

Ans. API testing helps ensure that the APIs function correctly, integrate seamlessly with other components, handle errors gracefully, and deliver data securely and efficiently.

68. What are the common types of API tests?

Ans. Common types of API tests include:

Unit Testing: Testing individual API methods/functions.

Functional Testing: Verifying the functionality of API endpoints.

Load Testing: Assessing API performance under expected load.

Security Testing: Checking API security mechanisms.

Integration Testing: Testing how APIs interact with other components.

End-to-End Testing: Testing entire workflows involving multiple APIs.

69. How do you perform API testing using Postman?

Ans. Using Postman, you can create requests for various HTTP methods (GET, POST, PUT, DELETE, etc.) to interact with APIs. You can then analyse the responses, set up automated tests using scripts (e.g., JavaScript), and organize requests into collections for better management.

70. Explain the components of a Postman request.

Ans. The main components of a Postman request include:

Request URL: The endpoint URL to which the request is sent.

HTTP Method: The type of request (GET, POST, PUT, DELETE, etc.).

Headers: Additional information sent with the request, such as authorization tokens or content type.

Body: Data sent with the request, often used in POST and PUT requests.

Parameters: Optional query parameters included in the URL.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

71. How do you handle authentication in Postman?

Ans. Postman supports various authentication methods such as Basic Auth, OAuth 2.0, and API keys. You can configure authentication in the request settings or use environment variables to store sensitive information securely.

72. What are Postman collections?

Ans. Postman collections are groups of requests that can be organized together for easier management and sharing. Collections can include requests, scripts, and pre-request scripts, allowing testers to automate and streamline API testing workflows.

73. How do you write tests in Postman?

Ans. Tests in Postman are written using JavaScript within the Postman interface. You can write tests in the Tests tab of a request, where you have access to the response body and headers. Common assertions include checking status codes, response body content, and response times.

74. How do you handle data-driven testing in Postman?

Ans. Data-driven testing in Postman can be achieved using CSV or JSON files as data sources. You can import data files into Postman and use variables to dynamically substitute values in requests. Postman's collection runner can then execute requests with different data sets.

75. What are the common challenges in API testing, and how does SoapUI address them?

Ans. Common challenges in API testing include handling authentication, managing test data, validating responses, and ensuring compatibility across different environments. SoapUI provides solutions like built-in support for various authentication methods, data-driven testing capabilities, extensive assertion options, and environment management features.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

76. Explain the difference between REST and SOAP APIs.

Ans. REST (Representational State Transfer) and SOAP (Simple Object Access Protocol) are two different architectural styles for building APIs.

REST is based on standard HTTP methods like GET, POST, PUT, and DELETE and uses JSON or XML for data interchange. It's known for its simplicity, scalability, and statelessness.

SOAP, on the other hand, is a protocol that uses XML for message formatting and typically relies on HTTP or SMTP for transport. It's more rigid and standardized, with built-in support for features like security and transactions.

77. How do you handle security testing in SoapUI?

Ans. SoapUI supports various security testing functionalities, including SSL encryption, WS-Security, OAuth, and basic authentication. Testers can configure security settings within SoapUI projects and execute security tests to verify that APIs are secure against common vulnerabilities.

78. What are the typical steps involved in API testing using SoapUI?

Ans. The typical steps in API testing using SoapUI include:

Creating a new SoapUI project and importing API definitions (WSDL, WADL, Swagger, etc.).

Designing test cases by creating requests, specifying parameters, and setting up assertions.

Executing test cases individually or as part of test suites.

Analysing test results and debugging failures.

Generating detailed reports for documentation and analysis.

79. How do you handle data-driven testing in SoapUI?

Ans. SoapUI supports data-driven testing by allowing testers to import data from external sources like Excel, CSV, or databases. Testers can use data sources to parameterize requests and iterate over multiple data sets, enabling comprehensive testing with varying input values.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

80. How do you integrate SoapUI tests into continuous integration/continuous deployment (CI/CD) pipelines?

Ans. SoapUI provides command-line execution capabilities, allowing testers to run tests from CI/CD tools like Jenkins or Azure DevOps. Testers can configure SoapUI tests as part of automated build pipelines, ensuring that API tests are executed automatically with each code change.

81. Can you explain the difference between mocking and virtualization in SoapUI?

Ans. Mocking in SoapUI involves creating simulated responses for APIs that may not yet be developed or available. It allows testers to simulate API behaviour and test client applications independently. Virtualization, on the other hand, involves creating virtual services that mimic the behaviour of real services, enabling parallel development and testing.

82. How do you ensure API test coverage in SoapUI?

Ans. Test coverage in SoapUI can be ensured by designing comprehensive test suites that cover different scenarios and edge cases. Testers can use techniques like equivalence partitioning, boundary value analysis, and exploratory testing to identify critical test cases and ensure thorough coverage of API functionality.

83. What are Postman collections, and how do you organize and manage them effectively?

Ans. Postman collections are groups of requests that can be organized together for easier management and sharing. Testers can create collections to group related requests, organize requests hierarchically using folders, and share collections with team members for collaboration.

84. How do you write automated tests in Postman, and what are some common assertions you use?

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Ans. Tests in Postman are written using JavaScript within the Postman interface. Testers can write scripts to automate tests and assertions to validate API responses. Common assertions include checking status codes, response body content, response times, headers, and schema validation.

85. Can you explain how to handle file uploads in Postman?

Ans. Postman supports file uploads using the form-data or binary format in request bodies. Testers can add a file parameter to the request body and specify the file to be uploaded. Postman also allows testers to generate code snippets for various programming languages to handle file uploads programmatically.

86. How do you integrate Postman tests into continuous integration/continuous deployment (CI/CD) pipelines?

Ans. Postman provides command-line execution capabilities, allowing testers to run tests from CI/CD tools like Jenkins, Travis CI, or Azure DevOps. Testers can export Postman collections as JSON files and use Newman (Postman's command-line runner) to execute tests as part of automated build pipelines.

87. What are some best practices for API testing with Postman?

Ans. Some best practices for API testing with Postman include:

Organizing requests into collections and folders for better management.

Using environment variables and data files for parameterization.

Writing clear and concise test scripts with meaningful assertions.

Regularly updating and maintaining tests as APIs evolve.

Sharing collections and collaborating with team members for better coverage and efficiency.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

88. How do you handle API versioning in Postman tests?

Ans. API versioning can be handled in Postman tests by creating separate collections or environments for different API versions. Testers can update requests in the collection to use the appropriate version endpoints and variables, ensuring compatibility and consistency in testing.

89. Which protocol does Rest Assure use?

Ans. Rest Assured is a Java-based library for testing RESTful APIs. It does not have its own protocol, but rather it works with HTTP-based protocols such as HTTP and HTTPS, which are the underlying protocols for RESTful web services.

90. Which methods does Rest Assure support?

Ans. Rest Assured supports various HTTP methods (GET, POST, PUT, DELETE, etc.), authentication mechanisms, request and response specifications, and assertion capabilities.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

HCL TECHNOLOGIES AUTOMATION INTERVIEW QUESTIONS: (4+ YOE)

91. Difference between endpoint and query parameters.

Ans. Endpoints and query parameters are both used in web APIs to specify and retrieve specific data or perform actions, but they serve different purposes and are used in different parts of a URL.

Endpoint:

- An endpoint is the specific URL (Uniform Resource Locator) where an API or web service can be accessed.
- It represents a resource or a collection of resources in the API.
- Endpoints are used to perform operations on resources such as retrieving, creating, updating, or deleting data.
- Example: https://api.example.com/users, where /users is the endpoint that represents a collection of user resources.

Query Parameters:

- Query parameters are additional parameters added to the end of a URL that modify the data returned from the endpoint.
- They are used to filter, sort, or paginate the data returned by the endpoint.
- Query parameters consist of key-value pairs separated by an ampersand (&) and are appended to the URL after a question mark (?).

Example: https://api.example.com/users?status=active&sort=asc, where status and sort are query parameters used to filter and sort the list of users.

92. How to pass more than one parameter for a request?

Ans. To pass more than one parameter for a request, you can use query parameters, request body, or a combination of both, depending on the HTTP method and the requirements of the API you are working with. Here's how you can pass multiple parameters using different approaches:

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

1. **Query Parameters:**

- Query parameters are appended to the end of the URL and are separated by an ampersand (&).
- Example URL with query parameters: https://api.example.com/resource?param1=value1&param2=value2
- You can pass multiple parameters by adding additional key-value pairs to the URL.

2. Request Body (POST, PUT, PATCH):

- For HTTP methods that support request bodies like POST, PUT, and PATCH, you can pass parameters in the body of the request.
- Parameters can be sent in various formats such as JSON, XML, or form-urlencoded.

Example JSON request body

```
{
 "param1": "value1",
 "param2": "value2"
}
```

3. Path Parameters:

- Path parameters are part of the URL path itself and are used to specify variable parts of the URL.
- They are typically used to identify a specific resource within a hierarchical structure.
- Example URL with path parameters: https://api.example.com/resource/{param1}/{param2}
- Path parameters are extracted from the URL path by the server.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

93. What tool do you use for documenting APIs?

- Swagger: Swagger is one of the most widely used tools for API documentation. It allows you to describe your API using the Specification (formerly Swagger Specification), which provides a standard way to define RESTful APIs. Swagger generates interactive API documentation, client SDKs, and server stubs based on your API specification.

94. What is Sprint Grooming?

Ans. Sprint grooming, also known as backlog grooming or refinement, is an essential ceremony in Agile methodologies like Scrum. It involves preparing the product backlog for upcoming sprints by reviewing, refining, and prioritizing user stories and tasks.

95. How to handle multiple browser windows?

Ans.

- Use the getWindowHandles() method to retrieve handles of all open browser windows.
- Use the switchTo().window() method to switch between different browser windows by providing the window handle.
- Use the close() method to close the current window or the quit() method to close all browser windows and end the WebDriver session.
- Once you've switched to a specific window, you can perform actions on elements within that window.

Code: WebElement element = driver.findElement(By.id("elementId"));

element.click();

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Complete Code:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.Set;
public class MultipleWindowsExample {
    public static void main(String[] args) {
        // Set system property for ChromeDriver
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
        // Initialize WebDriver
        WebDriver driver = new ChromeDriver();
        // Navigate to the webpage with multiple windows
        driver.get("https://www.example.com");
        // Get window handles
        Set<String> windowHandles = driver.getWindowHandles();
        // Iterate through each window handle
        for (String windowHandle : windowHandles) {
            // Switch to the window
            driver.switchTo().window(windowHandle);
            // Perform actions on the current window
            WebElement element = driver.findElement(By.id("elementId"));
            element.click();
        }
        // Close the browser
        driver.quit();
    }}
```

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

96. What are different exceptions you faced in Selenium?

Ans.

NoSuchElementException:

- Occurs when an element could not be found using the specified locator strategy (e.g., ID, XPath, CSS selector).
- Possible reasons include incorrect locator, element not yet loaded, or element not present on the page.

TimeoutException:

- Occurs when an operation (e.g., element search, page load) takes longer than the specified timeout period.
- Possible reasons include slow network connection, heavy page load, or incorrect timeout configuration.

StaleElementReferenceException:

- Occurs when an element reference becomes "stale" or "detached" from the DOM.
- Commonly encountered after a page refresh, navigation, or DOM modification that causes the element to be re-rendered.

ElementNotVisibleException:

- Occurs when an element is present in the DOM but not visible on the page.
- Commonly encountered with hidden elements or elements obscured by other elements.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

ElementNotInteractableException:

- Occurs when an element is present and visible but not interactable (e.g., input field disabled, button not clickable).
- Commonly encountered with disabled or read-only elements.

InvalidSelectorException:

- Occurs when an invalid or malformed locator strategy is used to find elements.
- Possible reasons include syntax errors, unsupported selector types, or incorrect usage of locators.

WebDriverException:

- Represents a generic exception related to WebDriver operations.
- Can be thrown for various reasons such as session timeout, unexpected browser behavior, or internal WebDriver errors.

NoSuchWindowException:

- Occurs when attempting to switch to a window that does not exist or has been closed.
- Commonly encountered when interacting with multiple browser windows or tabs.

UnhandledAlertException:

- Occurs when an unexpected alert dialog (e.g., JavaScript alert, confirmation, prompt) is encountered.
- Commonly encountered when interacting with JavaScript alerts or confirmation dialogs.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

UnexpectedTagNameException:

- Occurs when a WebElement is used with a tag name that does not match the expected tag name (e.g., trying to use a div element as if it were an input element).

97. How can we avoid duplicating the same URL in API testing?

Ans. To avoid duplicating the same URL in API testing, you can follow these best practices:

Centralize URL Definitions:

- Define URLs in a central location, such as a configuration file or constants file, rather than hardcoding them directly into test scripts.
- This allows you to easily update URLs in one place if they change, reducing the risk of duplication and inconsistency.

Use Environment-specific Configuration:

- Use environment-specific configuration files (e.g., development, testing, production) to store URLs for different environments.
- This ensures that test scripts can be executed in different environments without modifying the URLs manually.

URL Concatenation:

- If your API endpoints share a common base URL, concatenate the base URL with endpoint paths in your test scripts.
- This prevents duplication of the base URL and makes it easier to manage **changes to the base URL.**

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

URL Builders:

- Use URL builder functions or methods to construct URLs dynamically based on input parameters or configuration settings.
- This allows you to generate URLs programmatically and avoid duplicating static URL parts across multiple test scripts.

Parameterization:

- If your API endpoints require dynamic parameters (e.g., query parameters, path parameters), parameterize the URLs in your test scripts.
- Use placeholders or variables for parameter values that can be replaced at runtime, reducing the need for duplicate URLs with different parameter values.

Reuse URLs in Test Data:

- If you have multiple test cases that use the same URL, consider defining the URL in test data files (e.g., JSON, CSV) and referencing it from test scripts.
- This promotes reusability and reduces duplication of URLs across test cases.

Automated Checks for URL Consistency:

- Implement automated checks or linting rules to ensure URL consistency across test scripts.
- Use static code analysis tools or custom scripts to identify and flag duplicate or inconsistent URLs.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

98.What is status code 403? When do you get it?

Ans. HTTP status code 403 is the standard response for a server refusing to fulfil a request from a client because the client does not have the necessary permissions to access the requested resource. It is often referred to as "Forbidden".

Here are some common scenarios where a server might return a 403-status code:

Insufficient Permissions:

- The server recognizes the user's identity, but the user does not have sufficient permissions to access the requested resource.
- For example, attempting to access a protected file or directory without proper authentication or authorization.

Authentication Failure:

- The server requires authentication, but the provided credentials are invalid or missing.
- For example, accessing a restricted area of a website without providing the correct username and password.

IP Address Blocking:

- The server may block access to certain resources based on the client's IP address.
- For example, if the server detects suspicious activity or abusive behavior from a particular IP address, it may block access to prevent further unauthorized actions.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Rate Limiting:

- The server may impose rate limits on certain endpoints to prevent abuse or excessive usage.
- For example, if a client makes too many requests to a particular API endpoint within a short period, the server may respond with a 403-status code to indicate that the request has been throttled.

99. What is difference between Authentication vs Authorization?

Authentication:

- Authentication is the process of verifying the identity of a user or entity.
- It ensures that the user is who they claim to be before granting access to a system or resource.
- Authentication mechanisms typically involve presenting credentials, such as a username/password combination, biometric data, digital certificates, or API tokens.
- The primary goal of authentication is to establish trust in the identity of the user or entity accessing the system.

Authorization:

- Authorization is the process of determining what actions or resources a user or entity is allowed to access or perform.
- Once a user's identity has been authenticated, authorization determines the specific permissions or privileges associated with that identity.
- Authorization mechanisms typically involve defining roles, permissions, and access control rules that govern access to resources or functionality.
- The primary goal of authorization is to enforce access control policies and ensure that users can only access the resources and perform the actions that they are authorized to do.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

100. What is the background keyword in Cucumber?

Ans.

The Background keyword is used to define a set of common steps that are shared by both scenarios in the feature file.

- The steps defined under the Background section will be executed before each scenario in the feature file.
- In this case, the Background section sets up the initial state by navigating to the login page and ensuring that the user is logged out before each scenario.

101. Suppose you find a bug in production. how would you make sure that the same bug is not introduced again ?

Ans:

- Add uncaught functionality to regression test cases.
- If you have Automated Regression Suite, then write a new Script which validates above functionality

102. What do you do when your developer denies that what you filed IS A BUG ?

Ans : Provide Business Documentation reference to support why the existing functionality is not as per design.

• Involve Product Owner / Business Analyst for Discussion.

IF Bug is not reproducible then

• Provide Screenshots of the Bug, Give Timestamp on when you reproduced this so that Developer can check in Application Logs.

• Provide Test Data you have used for replicating issue

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

103. What has been one of your greatest challenges while doing regression testing?

Ans:

• Test Data issue

• Improper selection of regression test cases might skip a major regression defect to be found

104. What are the drawbacks of the Agile implementation/ methodology that you faced?

Ans:

- Sprints are usually very deadline constrained.
- Documentation is not the priority
- Frequent change in requirements

105. What is your approach when you have a high priority release to be delivered in a very short time?

Ans:

• Run Automation Suites

• Run Unit tests.

• Manual testing on high level Priority Business test cases

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

106. Can we use a background in one feature file for another feature file?

No, in Cucumber, the Background section defined in one feature file cannot be directly reused in another feature file. Each feature file in Cucumber is intended to be self-contained and independent of others. However, you can achieve similar behaviour by using reusable step definitions or scenario outlines.

Here are two approaches you can use to achieve reuse across multiple feature files:

Reusable Step Definitions:

- Define common steps in step definition files that can be reused across multiple feature files.
- Create separate step definition files for common actions or setup steps and include them in your feature files using the Given, When, Then, And, or But keywords.
- By organizing your step definitions effectively, you can achieve similar behaviour to a Background section shared across multiple feature files.

Scenario Outlines:

- Use scenario outlines to define parametrized scenarios that can be reused with different sets of input data.
- Define the common steps within each scenario outline, and provide different input data using Examples tables.
- While scenario outlines are typically used for data-driven testing, you can also use them to achieve common setup steps across multiple scenarios.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Example of Scenario Outline used in code

```
# Feature1.feature

Feature: Login functionality


  Scenario Outline: Valid login with different users
    Given the user navigates to the login page
    And the user is logged out
    When the user enters "<username>" and "<password>"
    Then the user should be logged in successfully


    Examples:
      | username | password |
      | user1    | pass1    |
      | user2    | pass2    |
```

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

107. How will you choose language for automating web-based application using selenium?

Ans. When choosing a programming language for automating web-based applications using Selenium, consider the following factors:

Familiarity and Expertise:

- Choose a programming language that you and your team are familiar with and proficient in.
- If your team has expertise in a particular language, it will be easier to write and maintain automation scripts.

Community Support:

- Consider the size and activity of the community around the programming language.
- A larger community usually means more resources, libraries, frameworks, and online forums for support and collaboration.

Ecosystem and Tooling:

- Evaluate the ecosystem and tooling available for the programming language.
- Look for frameworks, libraries, and IDE support that can enhance your automation efforts.

Integration with Testing Frameworks:

- Ensure that the programming language has good integration with popular testing frameworks, such as TestNG, JUnit, NUnit, or PyTest.
- Compatibility with testing frameworks simplifies test execution, reporting, and integration with CI/CD pipelines.

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Cross-Platform Support:

- Consider whether the programming language and its associated libraries support cross-platform development.
- If your team works across different operating systems (Windows, macOS, Linux), choose a language that offers good cross-platform support.

Performance and Speed:

- Evaluate the performance and execution speed of the programming language and its associated libraries.
- Choose a language that offers good performance for web automation tasks, especially for large test suites or complex scenarios.

Maintainability and Scalability:

- Consider the maintainability and scalability of automation scripts written in the programming language.
- Choose a language that supports good coding practices, modular design, and easy refactoring to ensure long-term maintainability.

Corporate Policies and Preferences:

- Consider any corporate policies, preferences, or standards regarding programming languages and technology stacks.
- Ensure that the chosen language aligns with the organization's guidelines and requirements.

108. What are the different snippets you used in postman for testing APIs?

In Postman, you can use a variety of snippets to write tests for your APIs. These snippets are pre-written code snippets that you can insert into your requests to perform various types of tests. Here are some common snippets you can use for testing APIs in Postman:

Top mate link for Mock Interviews: https://topmate.io/rd_automation_learning/723081

YouTube Link: https://www.youtube.com/channel/UC_0lWsDQPbATkXWdZTS8sfQ

Instagram Link: https://www.instagram.com/rdautomationlearning/

Status Code Check: Asserts that the response status code matches the expected value.

Response Body Check:

Asserts that the response body contains a specific string or value.

JSON Response Validation:

Asserts that the response body is a valid JSON and contains specific JSON properties and values.

Header Check:

Asserts that the response contains a specific header with the expected value.

Performance Testing:

Measures response time and asserts that it meets certain criteria.

Environment Variables:

Uses environment variables in tests for dynamic values.

Data-Driven Testing:

Runs the same test with different data sets.