61. Introduction about yourself Hi I am Madhusudhan , I am a QA l Automation Engineer with 5.2 years of experience. I have 3 years of experience in Test Automation and Rest with Manual Testing I have experience in UI Automation using Selenium with Java with a very good understanding of the Framework with TestNG annotations, assertions,Extent Reports,logs. Working with GitHub & Jenkins. I have working experience in Insurance,HealthCare Domains. My last project was Aviva Insurance application for UK client for which I was involved in TestAutomation

1a.including details about your project involvement.(Roles & Responsibilities)

In my last project I worked as Sr Automation Engineer.

My major responsibilities included:

a.Creation of POM Pages

where we would find elements with different locator techniques

Writing application specific methods for the page

b.Creation of Automation Scripts

c.Execution of the Individual Scripts,fixing issues.

d.Execution of the Framework with different Suites

f.Analyzing reports,fixing issues,re-execution of the Framework

g.logging defects

h.Building application specific methods

i.Participated in all the Sprint activities including Daily standup meeting,Sprint plan....

2. Explain about your project

Provide a clear, concise, and structured response that highlights the project goals,

your role, the tools and technologies used, your testing strategies, and the impact of your

work.

### Tips for Presenting Your Project

* Be Specific: Provide concrete examples and specific details about your contributions and the technologies used.
* Be Honest: If there were challenges, explain them honestly but focus on how you overcame them.
* Showcase Your Skills: Highlight any skills or tools that are relevant to the position you're applying for.
* Keep It Concise: While being thorough, ensure your explanation is concise and to the point to keep the interviewer engaged.
* Practice: Rehearse your explanation beforehand to ensure you can present it smoothly and confidently.

3. Describe the framework you have worked with.

What framework

TDD Approach

HybridFramework- Data Driven,Modular driven & Function driven

Key Components - It as Maven Project - Java (Programming Language)

- Selenium (Automation Tool)

TestNG for its annotations & Assertions

Extent Reports as a tool for Reporting

Log4j for logging

Data providers for supporting data driven testing which read the Data from Excel sheet and

runs the tests with different sets of data

Framework structure:

Generic Libraries

Application Specific Libraries

POM Pages

Modular - TestScripts

Framework Flow :We normally execute the framework through Jenkins with Scheduled execution

by executing the batch file/TestSuite.xml

and we can also execute it manually for local run

Execution can be done by anybody even without the knowledge of the Framework

It needs only configuration of the Excel File(Controller.xlsx) and double clicking the Batch file.

4.How do you handle dynamic elements in Selenium WebDriver?

5.What are some common exceptions in Selenium WebDriver and how do you handle them?

6.Explain the difference between implicit wait, explicit wait, and fluent wait in Selenium WebDriver.

7. How do you perform mouse hover actions in Selenium WebDriver?

8.How do you handle frames in Selenium WebDriver?

9.What is Page Object Model (POM) in Selenium WebDriver?

10. What are the main features of Java?

11.Explain the concept of the JVM, JRE, and JDK?

12.What is the difference between a JDK and a JRE?

13.What are the four main principles of Object-Oriented Programming?

14.Explain the concepts of inheritance, polymorphism, encapsulation, and abstraction?

15.How do you handle multiple windows in Selenium WebDriver?

16.What is a WebElement in Selenium?

17. Explain the difference between `final`, `finally`, and `finalize` in Java.

18.What is method overloading and method overriding?

19.What is an interface, and how is it different from an abstract class?

20.What are the different types of variables in Java?

21.Explain the difference between primitive and non-primitive data types?

22.How does Java handle memory management and garbage collection?

23.Explain the different control flow statements (if, switch, for, while, do-while)?

24.How does the switch statement work in Java?

25.What is the Collections Framework in Java?

26.Explain the difference between List, Set, and Map interfaces?

27.How does a HashMap work internally?

28.What is an exception in Java?

29.Explain the difference between checked and unchecked exceptions?

30.What are try, catch, finally, throw, and throws in exception handling?

18.Explain the difference between driver.get() and driver.navigate().to() methods in Selenium WebDriver.

20.What are checked and unchecked exceptions in Java? Give examples.

21. Describe the concept of object serialization in Java.

22.What is Selenium Grid and how does it work?

23.How do you take screenshots in Selenium WebDriver?

24.How do you handle alerts and pop-ups in Selenium WebDriver?

25.How do you handle multiple windows in Selenium WebDriver?

26.What are the principles of OOP? Explain with examples.

27.Discuss the differences between abstract classes and interfaces. When would you use one over the other?

28.How does Java support multiple inheritance through interfaces?

29.Explain the difference between `ArrayList` and `LinkedList`. When would you use one over the other?

30. Discuss the advantages of `HashMap` over `Hashtable`.

31.What is the `Comparator` interface used for in Java? Provide an example of its usage.

32.What is the `volatile` keyword used for in Java? How does it differ from `synchronized`?

33.Explain the `Executor` framework in Java. How would you submit tasks for execution using `ExecutorService`?

34.Discuss the `synchronized` keyword and its significance in concurrent programming.

35.Have you used any version control systems (e.g., Git) in your projects? Describe your experience.

36.What build tools have you used with Java projects? Discuss the advantages of tools like Maven or Gradle.

37.Discuss the Singleton pattern. How would you implement it in Java?

1. How do you review someone else's test script? 2. Where do you store test data, and how do you maintain it? 3. How do code changes affect your test scripts? 4. How do you handle OTP and captcha in Selenium? 5. How do you handle alert pop-ups that appear intermittently? 6. What are you doing in TY (assuming TY is a company or project)? 7. Can you explain a negative end-to-end scenario on Flipkart? 8. What approach do you take when a developer rejects a defect you've reported? 9. How do you perform regression testing, and how often do you do it? 10. How many threads do you run your scripts in parallel? 11. How do you share your jar files with colleagues? 12. Are you involved in backend testing? 13. Where do you see yourself in three years? 14. What is cURL? and how is it used? 15. What is the difference between authentication and authorization? 16. What are the new features in the latest version of Selenium? 17. Which Java version are you using, and what are the new features in the latest version? 18. What is the critical feature added to the new version of Selenium? 19. How do you handle new requirements that affect existing automation scripts? 20. What is the difference between regression testing and sanity testing? 21. Can you explain Agile ceremonies with respect to automation? 22. Have you performed sanity testing, and how? 23. Which JDK version are you using? 24. How do you estimate the time required to automate 30 test cases? 25. How do you ensure that all test cases are executed? 26. How do you handle failed test scripts? 27. How do you prioritize test cases? 28. How do you switch to the last tab when there are multiple tabs and you don't know the exact number? 29. How do you get test cases for automation? 30. How do you select test cases for automation? 31. Have you done non-functional testing in your project? 32. How do you prepare a regression suite? 33. Can you explain a test automation plan? 34. How do you synchronize an ArrayList? 35. Which environment have you worked in? 36. What is a testing environment? 37. Can you use a different design pattern instead of POM? 38. Can you explain your daily routine? 39. Can you take a screenshot of a headless script/tool? 40. How do you run the same test cases in different browsers without using a suite XML file? 41. How would you approach developing scripts for 100 test cases? 42. If you had to automate 100 test cases with a release deadline of tomorrow, how would you approach it? 43. What is the difference between smoke and sanity testing? 44. What code have you written in the BeforeTest and BeforeClass methods? 45. Which browsers have you used for compatibility testing in parallel execution? 46. Who assigns test cases to you? 47. Are developers involved in sprint planning meetings? 48. Have you used custom exceptions in your framework? 49. Can you change the polling period of implicit wait? 50. Have you done database testing, and how? 51. Can you explain the release process? 52. Why is automation testing important? 53. If I gave you an application, how would you determine which technology was used? 54. How can you identify if an API is a REST API? 55. If you set implicit wait to 10 minutes and explicit wait to 5 minutes for an element, how long would it wait for the element to be clickable? 56. Which API request would you use for login? 57. How would you estimate the time required to automate 25 test cases? 58. How do you prioritize test cases? 59. If the target folder is blank in the POM.xml file, will the script run? 60. If a script runs fine in Eclipse but fails in Jenkins, what would you do? 61. How would you click an element that is only visible when you mouse hover over it, without using the mouse hover action? 62. How often do you execute builds in Jenkins, and how do you handle failed scripts? 63. How often do you execute your test scripts? 64. Which environments have you worked in? 65. Can you perform automation testing in a production environment? 66. What is the maximum number of test scripts you have run? 67. Can you explain how you have used data providers? 68. How would you select a dynamic element in a dropdown, like the 5th element? 69. If you had 100 methods and wanted to execute only 10, how would you approach it? 70. How do you prioritize test cases? 71. How long does it take for Zenkin to execute all test scripts? 72. If the login button is not working, what steps would you take to debug before reporting a defect to the develope

1.Explain your roles and responsibilities

2.Explain Automation Life Cycle

3.Which Framework YOU have Used?

4.Explain Your Framework

5.What are the things u stored in PageFactory ? Why ?

6.Explain TestNG

7.What are the annotations U have Used ?

8.What is Constructor

9.Where have used a constructor in Selenium?

10.oops concepts

11.Login code

12.jdbc connection

13.Framework explain

14.Abstract and interface

15.Rtm

16.Test design technique

17.Regression testing

18.how to handle alert

19.how to take screenshot

21.tell me about Project .What are the challenge face during project

22.what is the difference between RC and webdriver

23.what is the framework to explain it.

24.why we use the wait statement.

25.method overriding and overloading with small program

26.difference between lists,set and maps

27.reverse a string programs

28.missing element in given array program

29.how to handle dynamic elements

30.tell me the end to end scenarios in your project

31.integration scenarios in your project.

32 What are the advantages of using Selenium WebDriver for automated testing?

33.What is an API?

34.Explain the difference between SOAP and RESTful APIs.

35.What is JSON? How is it related to APIs?

36.What are HTTP methods? List commonly used methods and their purposes.

37.What is HTTP status code 200? What does it signify?

38.Explain the term "endpoint" in the context of APIs.

39.Differentiate between PUT and POST methods in HTTP.

40.What is the purpose of the HTTP PATCH method?

41.What are the advantages of using JSON over XML in APIs?

42.What are the characteristics of REST architecture?

43.Explain the principles of RESTful web services.

44.What is HATEOAS (Hypermedia as the Engine of Application State)?

45.How does RESTful API handle sessions?

45.What are the common data formats used in RESTful APIs?

46.What are URI templates in RESTful APIs?

47.Explain the concept of OAuth. How does it work?

48.What is API key authentication? When is it used?

49.How do you handle authentication in RESTful APIs?

50.What is the difference between authentication and authorization?

51.What are the different types of testing for APIs?

52.How do you test RESTful APIs?

53.What is the role of tools like Postman and curl in API testing?

54.What are the challenges of testing APIs compared to GUI testing?

55.What are the best practices for designing a good API?

56.Explain the concept of versioning in APIs. Why is it important?

57.How do you handle pagination in API responses?

58.What is rate limiting? Why is it important in API design?

59.What HTTP headers are commonly used in API responses?

60.How do you secure an API?

61.What are some common security vulnerabilities in APIs?

62.Explain the concept of CORS (Cross-Origin Resource Sharing).

63.How do you handle errors in API responses?

64.What are some common HTTP status codes used for error handling?

65.What is API monitoring? Why is it important?

66.How do you improve the performance of an API?

67.What is caching in APIs? How does it help in improving performance?

68.Explain the concept of microservices. How does it relate to APIs?

69.What is GraphQL? How is it different from RESTful APIs?

70.How do you handle asynchronous operations in RESTful APIs?

71.Explain the concept of Webhooks in the context of APIs.

72.Have you used any API testing tools or libraries? Which ones?

73.What are the benefits of using Swagger or OpenAPI for API documentation?

74.Describe a situation where you had to integrate multiple APIs. How did you approach it?

75.How would you design an API for a real-time chat application?

76.What is your approach to documenting APIs?

78.How do you handle backward compatibility when updating APIs?

79.Explain the concept of idempotency in APIs. Why is it important?

80.How do you handle versioning of APIs? What are the different approaches?

81.How do you ensure the security of sensitive data transmitted via APIs?

82.What are the key factors to consider when choosing between RESTful and GraphQL APIs for a project?

83.What is method overloading and method overriding?84.What is an interface, and how is it different from an abstract class?

85.What are the different types of variables in Java?

86.Explain the difference between primitive and non-primitive data types?

87.How does Java handle memory management and garbage collection?

88.Explain the different control flow statements (if, switch, for, while, do-while)?

89.How does the switch statement work in Java?

90.What is the Collections Framework in Java?

91.Explain the difference between List, Set, and Map interfaces?

92.How does a HashMap work internally?

93.What is an exception in Java?

94.Explain the difference between checked and unchecked exceptions?

95.What are try, catch, finally, throw, and throws in exception handling?

selenium Question

96.How do you handle dynamic elements in Selenium WebDriver?

97.What are some common exceptions in Selenium WebDriver and how do you handle them?

98.Explain the difference between implicit wait, explicit wait, and fluent wait in Selenium WebDriver.

99. How do you perform mouse hover actions in Selenium WebDriver?

100.How do you handle frames in Selenium WebDriver?

101.What is Page Object Model (POM) in Selenium WebDriver?

102. What are the main features of Java?

103.Explain the concept of the JVM, JRE, and JDK?

104.What is the difference between a JDK and a JRE?

105.What are the four main principles of Object-Oriented Programming?

106.Explain the concepts of inheritance, polymorphism, encapsulation, and abstraction?

107.How do you handle multiple windows in Selenium WebDriver?

108.What is a WebElement in Selenium?

109.What strategies do you use to maintain Selenium WebDriver test scripts?

1.Explain complete Automation testing process

2.what is pom class

3.write a Restassured script for post request with header and path parameter and validate wrt statuscode

4.maven

5.arr={8,1,4,8,6,9,6,2}

O/p 8

print first duplicate value

6.have you worked on performance testing

7.status codes

8.jenkins

9.have you worked on database testing

10.how do you did Api testing using Restassured

11.how do you handle payload in api testing

12.which one you prefer to handle payload in api

13.what are the different types of Authorization you know

14.what type Authorization you perform

15.what type of framework you used in your project

[16:07, 08/08/2024] SachinB2B: Introduction

How you have done API testing

How did you perform CRUD operations in Api

Difference between authorisation and authentication

Which authentication you have used

Have you generated bearer token??

How you were getting bearer token from developer??

Were you hardcoding bearer token in the test script?? If no how you were maintaining generic bearer token for all the scripts

Difference between put and patch

Difference between put and post

Can you do post using put

Tell me a scenario where you have used put and delete http method in your project

Interviewer - Dhananjay

1. Introduction

2. How you have done API testing

3. How did you perform CRUD operations in Api

4. Difference between authorisation and authentication

5. Which authentication you have used

6. Have you generated bearer token??

7. How you were getting bearer token from developer??

8. Were you hardcoding bearer token in the test script?? If no how you were maintaining generic bearer token for all the scripts

9. Difference between put and patch

10. Difference between put and post

11. Can you do post using put

12. Tell me a scenario where you have used put and delete http method in your project

13. ⁠Java program, write in notepad and debug and compile using Java compiler

I/P — I Am String

O/P— g ni rtSmAI

14. How many members were there in your team

15. Are you ready to work for manual for some period of time and then continue automation?

16. ⁠Api automation framework

17. ⁠How you used to validate response body

18. ⁠Have you got status code as 401 and 403? If so explain the scenario

19. ⁠Which scrum methodology you have used??

20. ⁠Which are the meetings you have attended

Manual Testing interview Question

1.What is Manual Testing?

2.Why is Manual Testing important?

3.What are the different types of Manual Testing?

4.What is a Test Case?

5.What is a Test Plan?

6.What is Exploratory Testing?

7.What is Regression Testing?

8.What is the difference between Verification and Validation?

9.What is Smoke Testing?

10.What is Sanity Testing?

11.What is a Defect Life Cycle?

12.What is the difference between a defect, bug, and error?

13.What is Severity and Priority in the context of a defect?

14.What are the different types of test documents?

15.What is a Traceability Matrix?

16.What are some best practices for manual testing?

17.What tools are commonly used in Manual Testing?

18.How do you prioritize test cases?

19.How do you handle incomplete requirements during testing?

API Question:

1.What is an API?

2.What are the different types of APIs?

3.What is the REST API?

4.What is the SOAP API?

5.Explain the difference between REST and SOAP?

6.What are the common HTTP methods used in RESTful APIs?

7.When would you use a POST method over a GET method?

8.What is the difference between PUT and PATCH?

9.What is the purpose of the OPTIONS HTTP method?

10.What are the principles of RESTful API design?

11.What is an endpoint in REST API?

12.What are the best practices for designing RESTful APIs?

13.What is statelessness in REST APIs, and why is it important?

14.How do you handle versioning in REST APIs?

15.What is OAuth, and how does it work?

16.How do you document an API?

17.What tools do you use for testing APIs?

### **1. Manual Testing**

1. What is Manual Testing?
2. Why is Manual Testing important?
3. What are the different types of Manual Testing?
4. What is a Test Case?
5. What is a Test Plan?
6. What is Exploratory Testing?
7. What is Regression Testing?
8. What is the difference between Verification and Validation?
9. What is Smoke Testing?
10. What is Sanity Testing?
11. What is a Defect Life Cycle?
12. What is the difference between a defect, bug, and error?
13. What is a Severity and Priority in the context of a defect?
14. What are the different types of test documents?
15. What is a Traceability Matrix?
16. What are some best practices for manual testing?
17. What tools are commonly used in Manual Testing?
18. How do you prioritize test cases?
19. How do you handle incomplete requirements during testing?

### **82. Java Concepts**

1. What are the main features of Java?
2. Explain the concept of the JVM, JRE, and JDK?
3. What is the difference between a JDK and a JRE?
4. What are the four main principles of Object-Oriented Programming?
5. Explain the concepts of inheritance, polymorphism, encapsulation, and abstraction?
6. Explain the difference between final, finally, and finalize in Java.
7. What is method overloading and method overriding?
8. What is an interface, and how is it different from an abstract class?
9. What are the different types of variables in Java?
10. Explain the difference between primitive and non-primitive data types?
11. How does Java handle memory management and garbage collection?
12. Explain the different control flow statements (if, switch, for, while, do-while)?
13. How does the switch statement work in Java?
14. What is the Collections Framework in Java?
15. Explain the difference between List, Set, and Map interfaces?
16. How does a HashMap work internally?
17. What is an exception in Java?
18. Explain the difference between checked and unchecked exceptions?
19. What are try, catch, finally, throw, and throws in exception handling?
20. Describe the concept of object serialization in Java.
21. Discuss the differences between abstract classes and interfaces. When would you use one over the other?
22. How does Java support multiple inheritance through interfaces?
23. Explain the difference between ArrayList and LinkedList. When would you use one over the other?
24. Discuss the advantages of HashMap over Hashtable.
25. What is the Comparator interface used for in Java? Provide an example of its usage.
26. What is the volatile keyword used for in Java? How does it differ from synchronized?
27. Explain the Executor framework in Java. How would you submit tasks for execution using ExecutorService?
28. Discuss the synchronized keyword and its significance in concurrent programming.
29. What build tools have you used with Java projects? Discuss the advantages of tools like Maven or Gradle.
30. Discuss the Singleton pattern. How would you implement it in Java?

### **3. Java Programs**

1. Write a program for method overriding and overloading.
2. Write a program to reverse a string.
3. Write a program to find the missing element in a given array.

### 

### 

### **4. Selenium**

1. How do you handle dynamic elements in Selenium WebDriver?
2. What are some common exceptions in Selenium WebDriver and how do you handle them?
3. Explain the difference between implicit wait, explicit wait, and fluent wait in Selenium WebDriver.
4. How do you perform mouse hover actions in Selenium WebDriver?
5. How do you handle frames in Selenium WebDriver?
6. What is a Page Object Model (POM) in Selenium WebDriver?
7. How do you handle multiple windows in Selenium WebDriver?
8. What is a WebElement in Selenium?
9. Explain the difference between driver.get() and driver.navigate().to() methods in Selenium WebDriver.
10. What are the advantages of using Selenium WebDriver for automated testing?
11. How do you handle alerts and pop-ups in Selenium WebDriver?
12. How do you take screenshots in Selenium WebDriver?
13. What is Selenium Grid and how does it work?
14. What strategies do you use to maintain Selenium WebDriver test scripts?
15. What is the difference between RC and WebDriver?
16. Why do we use the wait statement?

### **5. API Postman**

1. What is Postman?
2. How do you test RESTful APIs with Post
3. How do you debug API requests in Postman?
4. What is the difference between Global variables and Environment variables in Postman?
5. How do you handle authentication in Postman?
6. Explain the Collection Runner in Postman?
7. How do you write tests in Postman?
8. What is an Environment in Postman?
9. How do you run multiple requests in Postman?
10. What are Collections in Postman?
11. Explain the main features of Postman?
12. How does Postman integrate with CI/CD pipelines?
13. What is Newman in Postman?

### **6. Git & GitHub**

1. What is the difference between Git & GitHub
2. How do you add multiple files to the staging area?
3. How do you add all files to the staging area?
4. How do you commit the changes and where will it go?
5. How do you create a branch ?
6. How do you switch to a branch?
7. How do you create & switch to a branch?
8. How did you work on your project using Git/GitHub?
9. How do you merge a branch?
10. What is the difference between pull & fetch?
11. How do you undo the changes in the working directory?

git checkout -- <file>

1. How do you unstage a file?

git reset <file>

1. How do you add changes or modify the latest commit?

git commit --amend

**Example**

1. git commit --amend -m "Updated commit message"
2. git add <file>

git commit --amend

1. How do you change a specific commit?

git commit --amend

1. What was changed since I last committed?

ans:Running the plain git diff command without *any* parameters can be pretty helpful: it will show you all of your local changes since you last committed.

1. What changes did I make to a specific file?

ans: You'll want to see only the changes in a certain file. You can simply add the path of a file as an option: git diff index.html

1. What changes did I really add to the staging area ?

By adding the --staged (or alternatively: --cached) option, Git will show which local changes you have already added to Staging Area, via "git add":

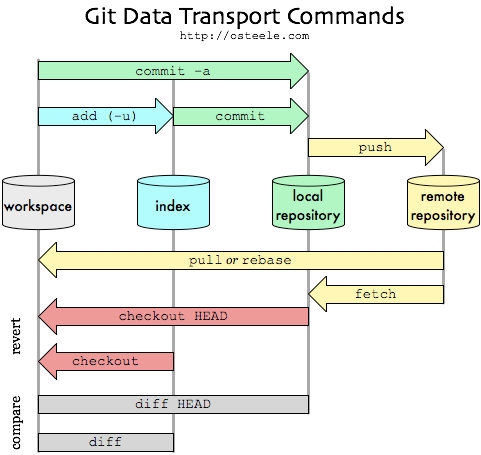
*# Staged changes in a certain file…*

$ git diff --staged index.html

*# Staged changes in all local files...*

$ git diff --staged

For your reference



Refer Below Link For Selenium and Java Questions

[Selenium and Java interview Questions](https://docs.google.com/document/d/1vyTXIlpYLJVK9UeQ6AqwMivam0PR8t0S6wKd-k8QXIw/edit)

**Jenkins Questions**

1. What is jenkins?
2. What are the different ways to execute a job?
3. What is scheduling?
4. What is a cron job?
5. What is cron expression?
6. What is poll SCM?
7. Steps to create a job in jenkins?
8. What is a pipeline job in jenkins?

1.Self introduction

2.Hashmap sorting

import java.util.\*;

public class HashMapSorting {

public static void main(String[] args) {

// Creating a HashMap

HashMap<String, Integer> map = new HashMap<>();

map.put("Banana", 3);

map.put("Apple", 2);

map.put("Orange", 5);

map.put("Mango", 1);

// Sorting by keys

List<Map.Entry<String, Integer>> keySortedList = new ArrayList<>(map.entrySet());

keySortedList.sort(Map.Entry.comparingByKey());

System.out.println("Sorted by keys:");

for (Map.Entry<String, Integer> entry : keySortedList) {

System.out.println(entry.getKey() + " = " + entry.getValue());

}

// Sorting by values

List<Map.Entry<String, Integer>> valueSortedList = new ArrayList<>(map.entrySet());

valueSortedList.sort(Map.Entry.comparingByValue());

System.out.println("Sorted by values:");

for (Map.Entry<String, Integer> entry : valueSortedList) {

System.out.println(entry.getKey() + " = " + entry.getValue());

}

}

}

3.Merge two lists without duplicates

and expected to use Arrays.asList() method .

import java.util.\*;

public class MergeListsWithoutDuplicates {

public static void main(String[] args) {

// Sample input lists

List<String> list1 = Arrays.*asList*("Apple", "Banana", "Orange");

List<String> list2 = Arrays.*asList*("Banana", "Grapes", "Apple", "Pineapple");

// Merging lists without duplicates

List<String> mergedList = *mergeListsWithoutDuplicates*(list1, list2);

// Printing the merged list

System.*out*.println("Merged List without Duplicates: " + mergedList);

}

public static <T> List<T> mergeListsWithoutDuplicates(List<T> list1, List<T> list2) {

Set<T> set = new LinkedHashSet<>(list1);

set.addAll(list2);

return new ArrayList<>(set);

}

}

4. When it is only one day left and you have more test cases to execute, what will be your approach

When you have only one day left and more test cases to execute, your approach should be:

1. **Prioritize Test Cases**: Focus on the most critical test cases that cover the core functionality and the areas with the highest risk.
2. **Risk-Based Testing**: Identify the areas of the application that are most likely to fail or have the highest impact if they fail.
3. **Use Automation**: If you have automated tests, run them to cover as much ground as possible in a short time.
4. **Time Management**: Allocate specific time slots for different sets of test cases to ensure maximum coverage.
5. **Collaboration**: Work closely with your team to divide the workload. If possible, get additional resources to help execute the test cases.

5. When you get critical defect today and your production is in evening, what will be your action.

If a critical defect is found on the day of production:

1. **Immediate Reporting**: Report the defect to the development and management teams immediately.
2. **Assessment**: Assess the impact of the defect. Determine if it affects critical functionality and if there is a workaround.
3. **Decision Making**: Work with the stakeholders to decide whether to delay the release or proceed with a known issue.
4. **Fix and Verify**: If a fix is possible, ensure it is implemented, tested, and verified before the production deployment.
5. **Communication**: Keep all relevant parties informed about the defect, its impact, and the chosen course of action.

6. How will you log defects.

When logging defects, you should include the following information:

1. **Defect ID**: A unique identifier for the defect.
2. **Title**: A short, descriptive title of the defect.
3. **Description**: A detailed description of the defect, including steps to reproduce it, expected results, and actual results.
4. **Severity and Priority**: The severity and priority of the defect.
5. **Environment**: The environment in which the defect was found (e.g., OS, browser, version).
6. **Attachments**: Screenshots, logs, or any other relevant attachments that help understand the defect.
7. **Assignee**: The person responsible for fixing the defect.
8. **Status**: The current status of the defect (e.g., New, In Progress, Fixed, Closed).

Defect ID: 12345

Title: Login page throws a 500 error when valid credentials are provided

Description:

- Steps to reproduce:

1. Navigate to the login page.

2. Enter valid credentials.

3. Click the login button.

- Expected result: User should be logged in and redirected to the dashboard.

- Actual result: The page throws a 500 error.

Severity: Critical

Priority: High

Environment:

- OS: Windows 10

- Browser: Chrome 89.0

Attachments:

- Screenshot of the error page

- Server logs

Assignee: John Doe

Status: New

7. Difference between list and hashmap, and what will they do on duplicates.

#### **List**

A List is an ordered collection of elements. It allows duplicate elements and maintains the insertion order. Common implementations include ArrayList, LinkedList, and Vector.

* **Allows Duplicates**: Yes, a List can contain duplicate elements.
* **Order**: Maintains the order of elements as they are inserted.
* **Access**: Elements can be accessed by their index.

**Example**:

#### **List**

A List is an ordered collection of elements. It allows duplicate elements and maintains the insertion order. Common implementations include ArrayList, LinkedList, and Vector.

* **Allows Duplicates**: Yes, a List can contain duplicate elements.
* **Order**: Maintains the order of elements as they are inserted.
* **Access**: Elements can be accessed by their index.

**Example**:

In this example, the List will contain two "Apple" entries because duplicates are allowed.

#### **HashMap**

A HashMap is an implementation of the Map interface. It stores key-value pairs and does not allow duplicate keys. However, it allows duplicate values. The order of the elements is not guaranteed.

* **Allows Duplicate Keys**: No, a HashMap does not allow duplicate keys. If a duplicate key is added, the new value replaces the old value.
* **Allows Duplicate Values**: Yes, a HashMap can contain duplicate values.
* **Order**: Does not maintain any order.

**Example**:

Map<String, String> map = new HashMap<>();

map.put("1", "Apple");

map.put("2", "Banana");

map.put("1", "Orange"); // Duplicate key

System.out.println(map); // Output: {1=Orange, 2=Banana}

In this example, the "Apple" value is replaced by "Orange" because the key "1" is already present.

8. When will you perform regression testing.

**Regression Testing** is performed to ensure that recent changes in the code have not adversely affected the existing functionality of the software. It is typically performed in the following scenarios:

1. **After Bug Fixes**: To verify that the bug fixes have not introduced new issues.
2. **After New Features**: To ensure that newly added features do not interfere with existing functionality.
3. **After Code Refactoring**: To confirm that the refactored code does not break any existing functionality.
4. **After Integration**: When integrating new modules or components, to ensure the integration does not affect the existing system.
5. **Regular Intervals**: As part of continuous integration/continuous deployment (CI/CD) pipelines, to ensure stability over time.

9. What are the locators you have used.

The common locators are:

1. **ID**: By.id("elementId")
2. **Name**: By.name("elementName")
3. **Class Name**: By.className("elementClass")
4. **Tag Name**: By.tagName("tagName")
5. **Link Text**: By.linkText("linkText")
6. **Partial Link Text**: By.partialLinkText("partialLinkText")
7. **CSS Selector**: By.cssSelector("cssSelector")
8. **XPath**: By.xpath("xpathExpression")

10. How will you come to know that there is error in data’s while passing API through postman.

**Status Codes**: Check the HTTP status codes returned by the API. Common error codes include:

* 400: Bad Request
* 401: Unauthorized
* 403: Forbidden
* 404: Not Found
* 500: Internal Server Error

**Response Body**: Inspect the response body for error messages or validation errors.

* API responses often include error details in the response body in case of failure.

Self introduction

Project explaination

Roles and responsibilities

Wat s ur roles and responsibilities in test yantra

**Explain defect life cycle**

New -> Assigned -> Open -> Fixed -> Retest -> Verified -> Closed

| | |

| V |

| Reopened |

V

Deferred

|

V

Rejected

The defect life cycle, also known as the bug life cycle, is a series of states that a defect or bug goes through from its identification to its resolution and closure. Understanding the defect life cycle is crucial for effective bug tracking and management in software development. Here's an explanation of the defect life cycle, detailing each stage:

### **1. New**

When a defect is first discovered, it is logged into the defect tracking system and assigned a status of "New." At this point, the defect is yet to be reviewed or assigned.

### **2. Assigned**

The defect is reviewed by the project manager or lead, and then assigned to a developer for investigation and fixing. The status is changed to "Assigned."

### **3. Open**

The developer starts analyzing and working on the defect. The status is changed to "Open." The developer will attempt to reproduce the defect to understand its nature and cause.

### **4. Fixed**

After the developer has made the necessary code changes to fix the defect, they mark it as "Fixed." The defect is now ready to be retested by the testing team.

### **5. Retest**

The tester retests the defect to verify that the fix works correctly and that it does not introduce new issues. The status is changed to "Retest."

### **6. Verified**

If the tester confirms that the defect is fixed and no longer exists, the status is changed to "Verified." This means the fix has been confirmed and the defect is considered resolved.

### **7. Reopened**

If the defect is still present or the fix is not adequate, the tester reopens the defect. The status is changed back to "Reopened," and it is reassigned to the developer for further investigation and fixing.

### **8. Closed**

Once the defect has been fixed and verified successfully, the tester changes the status to "Closed." This indicates that the defect has been resolved and no further action is required.

### **9. Deferred**

Sometimes a defect is not critical or significant enough to be fixed immediately and may be postponed to a future release. In this case, the status is changed to "Deferred."

### **10. Rejected**

If the defect is not valid or cannot be reproduced, the status is changed to "Rejected." This typically happens if the defect is due to user error, a misunderstanding, or is not actually a defect.

### **Importance of the Defect Life Cycle**

1. **Tracking and Management**: The defect life cycle helps in tracking and managing defects efficiently, ensuring that they are resolved in a systematic manner.
2. **Transparency**: It provides transparency in the defect management process, allowing all stakeholders to see the status and progress of each defect.
3. **Quality Assurance**: It ensures that defects are properly addressed, contributing to the overall quality and reliability of the software.
4. **Accountability**: It assigns responsibility at each stage, ensuring that defects are promptly and effectively handled by the appropriate team members.

By understanding and following the defect life cycle, teams can improve their defect management processes, leading to higher software quality and more efficient project execution.

How many test cases u have executed and how many were regression

How ur release goes wat do u do in the entire release

Program to Validate the password- should have atleast one uppercase, one lower case,one no,no space ,no concurrent characters and size should be between 8 to 16

Tekion 2nd round

1.self intro

2.challenges

3.critical defect

4.contributions in your framework

5.how do you seggregate test cases

6.critical feature in your framework

7.have you faced any challenges while automating test cases and how did you resolve it

8.have you supported any of your team members when they were facing difficulties

9.have you worked on manual testing