**Q1-Introduce yourself:**

As you know, I’m **Vishal Vishwakarma**, with over 14 years of experience in the healthcare insurance domain, including 9 years in the IT industry. For the last 7 years, I’ve been working on automation testing and currently working as a **Lead QA Tester at Wipro**.

Throughout my career, I’ve been actively involved in the complete QA lifecycle — from requirements analysis, test planning, test case design, execution, reporting, and post-deployment validation. I ensure that every test meet standards of accuracy.

From a technical perspective, I specialize in **Java and Object-Oriented Programming**, which has helped me create reusable and scalable automation frameworks. I’ve developed cross-browser automation test scripts using Selenium **WebDriver**, applying both Page Factory and Hybrid frameworks for better maintainability and reusability.

I use **Maven** for managing project dependencies in **POM.xml** file, and build processes, which integrates seamlessly **with Jenkins for CI/CD**. I regularly schedule automated executions and manage test pipelines across multiple browsers using Jenkins jobs.

I store test data in **Excel** or sometimes **JSON** files, and access them in test cases using custom reader utility methods. For global configuration data like browser type, I use a properties file.

For test reporting, I use **TestNG** along with **Extent Reports** to generate interactive reports for client-facing dashboards. Internally, we rely on **Log4j** to capture detailed execution logs for debugging and tracking.

I also use **Git and GitHub** for version control, manage branching, pull requests, and team collaboration.

On the process side, I work in an **Agile** environment where I actively participate in all Agile ceremonies including **Sprint planning, Daily stand-ups, Sprint Reviews, Retrospectives, and Backlog refinement meetings**. I also participate in **Defect Triage** meetings to prioritize bugs based on severity, impact, and business timelines. This helps align QA efforts closely with development and product goals, ensuring high-quality and timely deliveries.

**Challenges: #mailosaur #links**

**Q2-What is Abstract Class, uses and benefits:**

**Data abstraction** is the process of hiding certain details and showing only essential information to the user.

**Abstraction** can be achieved by abstract classes.

The abstract keyword is a non-access modifier, used for classes and methods:

**Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

**Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

An abstract class can have both abstract and regular methods

**Q3- What is Page Object Model or Page Factory? How it is used in Selenium?**

**Page Object Model (POM)/PageFactory is a design pattern used in Selenium automation to improve code structure, readability, and maintainability.**

**Each web page** of the application is represented by a **separate Java class**

The **elements (locators)** and the **actions (methods)** for that page are written inside that class

**Example:**

\* Login Page → LoginPage.java

\* Home Page → HomePage.java

\* Dashboard Page → DashboardPage.java

**Then Web Elements identified and actions methods are build:**

\* Web elements (using **@FindBy or By**)

\* Methods to interact with those elements

Once all the page classes are created, they are used in the actual test cases to avoid code duplication. This makes the test cases cleaner and easier to understand. It also provides the advantage of maintaining or updating the core logic in the page classes without modifying the test cases themselves.

**Q4-What is Maven and POM.xml file:**

**Maven** is a widely used open-source build and project management tool for Java projects. It streamlines tasks like compiling, testing, packaging, deploying, and managing dependencies. At the core of Maven is the pom.xml file (Project Object Model), which holds all the essential project configuration — including dependencies, plugins, goals, and build lifecycle instructions needed to build and manage the Java project efficiently.

**Q5-Challenges in Selenium Automation:**

Selenium automation comes with several challenges. One major issue is handling dynamic web element, where elements load or change after the page has initially rendered. This can cause scripts to fail if the elements aren't present or have changed when Selenium tries to interact with them. Browser compatibility is another concern, as different browsers may render pages differently, and updates can break existing tests. Selenium also has limitations—it is focused solely on web applications and cannot handle videos, barcodes, images, or captchas effectively. Scaling tests across multiple browsers and environments can be difficult without additional tools or infrastructure. Test flakiness is also common due to timing issues, network delays, or unpredictable behaviors, resulting in false positives or negatives. Handling browser alerts, pop-ups, or JavaScript dialogs can be tricky as well. Additionally, running a large number of tests in parallel can put a strain on system resources, affecting overall performance.

**Q6-What is Unit/Integration/Regression/Sanity/Smoke Testing?**

**Unit**- Unit testing is a software testing method where individual units or components of a code, typically functions or methods, are tested in isolation to verify their functionality. It's a crucial part of the software development process, helping identify bugs early and enhance code quality.

**Integration**- Integration testing in software development verifies how different components or modules of a system work together after they have been individually tested. It focuses on identifying and resolving issues that arise from the interaction between these components, ensuring they communicate and collaborate correctly.

**Regression**- It is full volume or on full database testing when Integration testing is completed. It is the final testing to see if the build is error free and ready for prod load.

**Sanity**- It is the subset of regression testing and is performed whenever a new build is created. It is performed when we have limited time, so a narrow and shallow testing is done to check if crucial functions are working properly.

**Smoke**- It is done immediately when a defect is fixed, it is mainly to focus on the main issue to check if the build is stable or not.

**Q7-What is class and object in java:**

The class represents a group of objects having similar properties and behavior, or in other words, we can say that a class is a blueprint for objects, while an object is an instance of a class.

**Q8-What are methods/functions in java:**

A method is a block of code which only runs when it is called. Methods are used to perform certain actions, and they are also known as functions. We can pass the data or parameters to process the testing if needed. Methods are with/without parameters both.

**Q9-What are collections in java:**

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Map) and classes (ArrayList, LinkedList, HashSet, LinkedHashSet, HashMap)

**Q10-What is public/private keyword in java:**

Public: The public keyword in Java is an access modifier used to define the access level of classes, methods, and variables. When a member is declared as public , it can be accessed from any other class, regardless of the package it belongs to.

Private: The private modifier specifies that the member can only be accessed in its own class. The protected modifier specifies that the member can only be accessed within its own package.

**Q11-What is static in java:**

A static class in Java is a class that cannot be instantiated. That is, we cannot create objects of a static class. We can only access its members using the class name itself. In other words, static means it belongs to a type itself rather than an instance of that type.

**Q12-What is void in java:**

The void keyword in Java is used to specify that a method does not return any value. It is a return type that indicates the method performs an action but does not produce a result that can be used elsewhere in the code.

**Q13-Explain 4 principles of java:**

Encapsulation in Java is a technique used to protect an object’s data from unauthorized access and modification by using private fields and providing public getter and setter methods. It allows data hiding and ensures that data is only accessed or modified in a controlled manner.

Inheritance is a fundamental concept in object-oriented programming that allows a new class to inherit properties and behaviors (fields and methods) from an existing class. This promotes code reusability and establishes a hierarchical relationship between classes.

Polymorphism in Java is a core concept of object-oriented programming (OOP) that allows objects to be treated as instances of their parent class. It facilitates flexibility and the ability to define methods in multiple forms. Polymorphism is primarily achieved through method overriding and method overloading.

Abstraction in Java refers to hiding the implementation details of a code and exposing only the necessary information to the user.

**Q14- What is different between setSpeed() and Thread.sleep();**

The main difference between them is that: setSpeed sets a speed that will apply a delay time before every Selenium operation. thread. sleep() will set up wait only for once when called.

**Q15-What is difference between assert and verify:**

If the assert condition is not met, test case execution will be aborted. The remaining tests are skipped, and the test case is marked as failed. These are hard assertions and throw the exception and derail the execution.

Verify or Soft Asserts will report the errors at the end of the test. Tests will not be aborted and will continue to run until the last test is executed, even if assert conditions are not met.

**Q16-What are setters and getters in java:**

Getters and setters are used to protect the data, particularly when creating classes. For each instance variable, a getter method returns its value while a setter method sets or updates its value. Getters and setters are also known as accessors and mutators, respectively.

**Q17-What is overriding in java:**

Overriding in Java occurs when a subclass or child class implements a method that is already defined in the superclass. When a subclass provides its own version of a method that is already defined in its superclass, we call it method overriding. The subclass method must match the parent class method’s name, parameters, and return type.

**Q18-What is overloading in java:**

Method Overloading allows us to define multiple methods with the same name but different parameters within a class. This difference can be in the number of parameters, the types of parameters, or the order of those parameters. Method is also called Static Polymorphism.

**Q19- what is difference between final, finally, and finalize:**

1. **final**: The “final” is the keyword that can be used for restricting any change in variables, methods, and classes. Final method can be inherited, but cannot be overridden or executed. It can be inherited but ensures that the method’s logic remains unchanged in child classes. This is because any critical business logic codes are implemented and in testing we do not want to mess by overriding it.
2. **finally**: The “finally block” is used in exception handling to ensure that a certain piece of code is always executed whether an exception occurs or not.
3. **finalize**: finalize is a method of the object class, used for cleanup before garbage collection.

**Q20-What is thread in java:**

A thread is an execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently. Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority.

**Q21-What is thread safe in java:**

Thread safety in Java refers to the property of a program or system where it can handle multiple threads concurrently without causing data corruption or unexpected behavior.

**Q22-What is the difference between == and .equals() in Java?**

In Java, == compares object references (memory locations), while .equals() compares actual content or values. Use == to check if two references point to the same object, and .equals() to check if two objects have the same data.

**Follow-up: How does this apply to comparing Strings in test cases?**

In test cases, use .equals() to compare strings, as == checks memory reference, not content or values. This ensures accurate value comparison.

**Q23- What are access modifiers in Java? Explain their scope:**

Access modifiers control the visibility of classes, methods, and variables in Java. There are four types:

1. Private: Accessible only within the same class.

2. Default (no modifier): Accessible within the same package.

3. Protected: Accessible within the same package and subclasses.

4. Public: Accessible from anywhere in the program.

**Q24- What is an Interface and how is it different from an Abstract Class?**

In Java, an interface defines a set of abstract methods that a class must implement, often to represent shared behavior across unrelated classes. Unlike abstract classes, interfaces (until Java 8) had no method implementations, but now can include default and static methods. A class can implement multiple interfaces but extend only one abstract class. Interfaces only hold constants(**public,static,final**), while abstract classes can have variables, constructors, and both abstract and concrete methods. Use abstract classes for shared base logic and interfaces for defining common capabilities.

**Q25- How do you handle exceptions in Java:**

Exception handling in Java is done using try, catch, and finally blocks:

1. **try**: Contains the code that may cause an exception.

2. **catch**: Handles the exception.

3. **finally**: Executes regardless of whether an exception occurs or not.

**Using throw and throws also exception are handled:**

**throw** is used to manually throw an exception

**throws** is used in method declaration to indicate that the method may throw an exception

**How does try/catch works? And is multiple try catch block possible?**

**>Java tries to execute the code inside the try block.**

**>If an exception occurs, it jumps to the catch block.**

**>If no exception occurs, the catch block is skipped.**

**>You can use multiple catch blocks to handle different types of exceptions.**

**> Optionally, you can add a finally block, which always runs — even if an exception occurs.**

**Q26- What is a constructor in Java:**

A constructor is a special method used to initialize objects. It has the same name as the class and does not have a return type.

**Q27- Java Collections is a framework that provides data structures like lists, sets, and maps.**

Commonly used collections in automation testing:

1. List (ArrayList, LinkedList) – Storing test data dynamically.

2. Set (HashSet) – Storing unique test data.

3. Map (HashMap) – Storing key-value pairs (e.g., configuration settings).

**Q28- What is the role of ‘synchronization’ in multi threading ?**

Synchronization prevents multiple threads from accessing the same resource simultaneously, avoiding race conditions. One can use this by putting keyword “synchronized”, or give the example of Singleton design pattern.

**Q28- What are benefits of Singleton design pattern ?**

The singleton design pattern offers several benefits, primarily by ensuring a class has only one instance and providing a global access point to it. This can lead to resource efficiency, simplified management of shared resources, and consistency across the application.

**Q28 - What is Log4j?**  
It is a **logging framework** for **Java** used to record **runtime information** for **debugging**, **auditing**, or **monitoring**.

**Q - What are the main components of Log4j?**  
**A**: **Logger**, **Appender**, **Layout**, **Configuration File**

**Q - What are logging levels in Log4j?**  
**A**: **TRACE**, **DEBUG**, **INFO**, **WARN**, **ERROR**, **FATAL**

**Q - What is the difference between Logger and RootLogger?**  
**A**: **Logger** is specific to a **class/package**; **RootLogger** is the **default logger** if no specific one is defined.

**Q - How is Log4j configured?**  
**A**: Via **XML**, **properties**, or **JSON configuration files**.

**Q - What is an Appender in Log4j? Name a few types.**  
**A**: An **Appender** defines **where the logs are written**, like:

**ConsoleAppender**

**FileAppender**

**RollingFileAppender**

**SocketAppender**

**Q - What is the use of PatternLayout?**  
**A**: **Formats** the **log message output**. Supports **custom patterns** like **%d**, **%p**, **%c**, **%m**, etc.

**Q - What do MaxFileSize and MaxBackupIndex do in Log4j 1.x?**  
**A:MaxFileSize**: Defines **rollover size**.

**MaxBackupIndex**: Number of **backup log files** to keep.

**Q - How does Log4j 2 differ from Log4j?**  
**A:Async logging support**

**Better performance**

**Plugin-based architecture**

**No static methods**

**Support for Java 8 lambdas**

**Q - What is Async Logging in Log4j2? Why is it used?**  
**A**: **Improves performance** by **decoupling logging** from the **application thread** using **AsyncAppender** or **AsyncLogger**.

**Q - How does RollingFileAppender work?**  
**A**: **Rolls over the log file** once a **limit** (like **size** or **date**) is reached. Can keep a specified number of **backup files**.

**Q - Can you configure Log4j programmatically?**  
**A**: **Yes**. We can use **Java APIs** to configure **loggers**, **appenders**, and **layouts** at **runtime**.

**Q - How do you handle logging in a multi-threaded environment?**  
**A**: Use **asynchronous logging (AsyncLogger)** or ensure **thread-safe appenders**.

**Q - How do you avoid performance issues with too much logging?**  
**A:**Use **INFO** or **WARN** levels in **production**

Use **DEBUG** or **TRACE** in **development/testing**

Use **async loggers** and **rolling policies**

**Q - What would you do if log files grow too large?**  
**A**:Implement **RollingFileAppender** to manage file size

Use **TimeBasedRollingPolicy**, which rolls over log files based on **time intervals** (e.g., **daily**, **hourly**, or **weekly**)

Helps in **organizing logs** and managing **file sizes** over time

**Q - What happens if the logging configuration file is missing?**  
**A**: **Log4j** will fallback to a **default configuration** (console output with **ERROR level**)

**Q29- What is a git repository?**

A repository is a file structure where git stores all the project-based files. Git can either stores the files on the local or the remote repository.

**Q30- What does git clone do?**

The command creates a copy (or clone) of an existing git repository. Generally, it is used to get a copy of the remote repository to the local repository.

**Q31- What does the command git config do?**

The git config command is a convenient way to set configuration options for defining the behavior of the repository, user information and preferences.

##How to configure User Name and User Email in GitBash/Terminal

git config --global user.name "Your Name", then enter

git config --global user.name - use this command to see if configured or not

git config --global user.email "Your Email Id", then enter

git config --global user.email - use this command to see if configured or not

**Q32- Can you explain head in terms of git and also tell the number of heads that can be present in a repository?**

• A head is a reference to the last commit object of a branch.

• For every repository, there will always be a default head referred to as “master” or now “main” (as per GitHub) but there is no restriction to the count of heads available. In other words, it can have any number of heads.

• Uses below:

To go or checkout to 1 commit before the latest commit, we use git checkout HEAD~1

- To uncommit the last 3 commits without losing the changes, we first run git reset HEAD~3. Then we can see the changes made in the last 3 commits and then update it manually and commit it finally.

- In order to uncommit the last 3 commits and also remove the changes, we can run the command: git reset --hard HEAD~3. This command will completely remove all the changes.

**Q33- What is a conflict or merge conflict?**

Conflict appears when:

1. When two separate branches have changes to the same line in a file

2. A file is deleted in one branch but has been modified in the other.

These conflicts have to be solved manually after discussion with the team as git will not be able to predict what and whose changes have to be given preference.

**Q34- What does git status command do?**

‘git status’ command is used for showing the difference between the working directory and the index, also keep track of the tracked and non-tracked changes.

**Q35- Define “Index”.**

Before making commits to the changes done, the developer is given provision to format and review the files and make innovations to them. All these are done in the common area which is known as ‘Index’ or ‘Staging Area’. It provides an opportunity for the lead to evaluate changes before committing them.

**Q36- What does git add command do?**

• This command adds files and changes to existing directory.

• You can add all changes at once using ‘git add . ‘

• You can add files one by one specifically using ‘git add <file\_name> ‘

• You can add contents of a particular folder by using git ‘add /<folder\_name>/ ‘

**Q37- What is a version control system ?**

A Version Control System keeps track of the contributions of the developers working as a team on the projects. They maintain the history of code changes done and with project progression, it gives an upper hand to the developers to introduce new code, fixes bugs, and run tests with confidence that their previously working copy could be restored at any moment in case things go wrong.

**Q38- How would you recover a branch that has already pushed changes in the central repository but has been accidentally deleted from every team member’s local machines?**

We can recover this by checking out the latest commit of this branch in the reflog and then checking it out as a new branch.

**Q39- Can you tell something about git reflog?**

This command tracks every single change made in the repository references (that can be branches or tags) and also maintains the branches/tags log history that was either created locally or checked out. Reference logs such as the commit snapshot of when the branch was created or cloned, checked-out, renamed, or any commits made on the branch are maintained by Git and listed by the ‘reflog’ command.

• This recovery of the branch is only possible when the branch was either created locally or checked-out from a remote repository in your local repository for Git to store its reference history logs.

• This command should be executed in the repository that had the lost branch.

**Q40- What consists of a commit object?**

A commit object consists of the following components:

• A set of files that represents the state of a project at a given point in time.

• Reference to parent commit objects.

• A 40 character string termed as SHA-1 name uniquely identifies the commit object.

**Q41- What is a detached HEAD and what causes this and how to avoid this?**

Detached HEAD indicates that the currently checked-out repository is not a local branch. This can be caused by the following scenarios:

• When a branch is a read-only branch and we try to create a commit to that branch, then the commits can be termed as “free-floating” commits not connected to any branch. They would be in a detached state.

• When we checkout a tag or a specific commit and then we try to perform a new commit, then again the commits would not be connected to any branch. When we now try to checkout a branch, these new commits would be automatically placed at the top.

In order to ensure that detached state doesn't happen, =instead of checking out commit/tag, we can create a branch emanating from that commit and then we can switch to that newly created branch.

**Q42- What do the git diff and git status commands do?**

git diff git status

This shows the changes between commits, working trees, etc. This shows the difference between the working directory and index that is essential in understanding git in depth.

**Q43- How will you create a git repository?**

• Create a folder for the project and then run ‘git init’

• Doing this will create a .git file in the project folder which indicates that the repository has been created.

**Q44- Tell me something about git stash?**

Git stash can be used in cases where we need to switch in between branches and at the same time not wanting to lose edits in the current branch. Running the git stash command basically pushes the current working directory state and index to the stack for future use and thereby providing a clean working directory for other tasks.

**Q45- How to delete a branch.**

‘git branch -d ‘branch-name’ (One should exit from the branch before deleting, and switch to main/other branch first)

**Q46- Can you tell the difference between Git and GitHub?**

**Git:** This is a distributed version control system installed on local machines which allow developers to keep track of commit histories and supports collaborative work.

**GitHub:**This is a cloud-based source code repository developed by using git.

**Q47- How will you resolve conflict in Git?**

• Conflicts occur whenever there are multiple people working on the same file across multiple branches. In such cases, git won't be able to resolve it automatically as it is not capable of deciding what changes has to get the preference.

• Following are the steps are done in order to resolve git conflicts:

1. Identify the files that have conflicts.

2. Discuss with members who have worked on the file and ensure that the required changes are done in the file.

3. Add these files to the staged section by using the git add command.

4. Commit these changes using the git commit command.

5. Finally, push the changes to the branch using the git.

**Q48- What command helps us know the list of branches merged to master?**

• git branch --merged helps to get the list of the branches that have been merged into the current branch.

• Note: git branch --no-merged lists the branches that have not been merged to the current branch.

**Basic commands of Git:**

##HTTPS means you are associated with credential manager. Its always better for SSH.

cd /c/Users/visha/.ssh (How to move to any folder)

cd /c/Users/visha/eclipse-workspace/openEnrollment

cat id\_ed25519.pub (to check its keys and if private or public)

$ ssh-keygen -R github.com (Fixed - The authenticity of host 'github.com (20.207.73.82)' can't be established.

ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.

This key is not known by any other names.)

pwd (Check the current folder exact path)

git init (To initialize the Git repository, run from project folder)

git remote add origin git@github.com:vishal20karma/OpenEnrollment.git (Laptop Connecting GitHub, can be found under)

git add . (Adds all the files)

git status (Check the project status)

git commit -m "initial commit" (How to commit with comments)

git push -u origin main (How to push the code to GitHub)

git clone git@github.com:vishal20karma/OpenEnrollment.git (How developer clones the project, this url can be copied from repository created in GitHub under code, then same cloned project is imported by developer/tester through Eclipse>File>Import>Maven>Existing maven project>Next>Browse>folder where project cloned>open)

git branch (How to check the branch)

git pull (How to pull the code from GitHub to local machine/laptop)

git branch b1 (how to create a branch, can name anything)

git checkout b1 (How to switch to any branch)

git branch -d b1 (How to delete any branch, one should exit from the branch before deleting, and switch to main/other branch)

git checkout -b b1 (How to create a branch and directly switch to it within one line of command)

Code changes in Eclipse can be seen for that branch only where the control(\*) will be, if on main branch then only main branch codes will appear and vice versa

git branch -m 'old branch' 'new branch' (How to rename the branch, also control should be in master to rename the branch)

**Q49. What is Jenkins?**

Jenkins is an open-source automation server that can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

**Q50. Tell me something about Continuous Integration, Continuous Delivery, and Continuous Deployment?**

**Continuous Integration:** A software development process where the changes made to software are integrated into the main code as and when a patch is ready so that the software will be always ready to be - built, tested, deployed, monitored - continuously.

**Continuous Delivery:** This is a Software Development Process where the continuously integrated (CI) changes will be tested & deployed continuously into a specific environment, generally through a manual release process, after all the quality checks are successful

**Continuous Deployment:** A Software Development practice where the continuously integrated (CI) changes are deployed automatically into the target environment after all the quality checks are successful.

• CI/CD – At project level in Jenkins, ‘Configure’ > under ‘Triggers’ and check ‘GitHub Webhook trigger for GITScm polling?’

• ‘Manage Jenkins’ > under ‘GitHub’ we have ‘Advance’ then copy the WebHook URL under ‘Override Hook URL’, then go to GitHub, Go to Project Level ‘Settings’, click ‘WebHook’ given on the left side, click ‘Add WebHook’ and paste the URL copied from Jenkins in the ‘Payload URL’. Save it and, so whenever any new code will be pushed to the project repository, Jenkins will automatically trigger the build. WebHook is a connection to communicate between GitHub and Jenkins.

**Q51. How can we stop a scheduled job from being executed temporarily?**

Disable the job from the job details page to temporarily stop all scheduled executions & other factors/events from triggering the job and enable it back to resume the job schedules/triggers. If a job is not required permanently, we can delete the job from the jobs list view page.

**Q52. What is a Jenkins Pipeline?**

The pipeline is a special type of Jenkins job - simply a sequence of steps controlled by a defined logic - which Orchestrates long-running activities that can span across multiple build agents. It is suitable for building and organizing complex activities that cannot be easily achieved using a freestyle job.

**Q53. How do you store credentials in Jenkins securely?**

Credentials can be stored securely in Jenkins using the Credentials plugin, which stores different types of credentials like - Username with a password, SSH username with the private key, AWS Credentials, Jenkins Build Token, Secret Files.

**Q54. What are the ways to trigger a Jenkins Job/Pipeline?**

There are many ways we can trigger a job in Jenkins. Some of the common ways are as below -

• Trigger an API (POST) request to the target job URL with the required data.

• Trigger it manually from the Jenkins web application.

• Trigger it using Jenkins CLI from the master/slave nodes.

• Time-based Scheduled Triggers like a cron job.

• Event-based Triggers like SCM Actions (Git Commit, Pull Requests), WebHooks, etc.

• Upstream/Downstream triggers by other Jenkins jobs.

**Q55. What is Jenkins Build Cause?**

Build Cause is a text attribute that represents what made a job's build to be triggered, say it could be a Jenkins User (from UI), Timer for Scheduled jobs, Upstream jobs for a job which was triggered by upstream job, etc. This is mainly used to identify the nature of the builds - be it nightly, manual, automated, etc.

**Q56. How Jenkins knows when to execute a Scheduled job/pipeline and how it is triggered?**

Jenkins master will have the cron (periodically build job) entries set up for the jobs as per the scheduled Job's configurations. As and when the time for a particular job comes, it commands agents (based on the configuration of the job) to execute the job with required configurations.

**Q57. What is the Jenkins User Content service?**

Jenkins has a mechanism known as "User Content", where administrators can place files inside the $JENKINS\_HOME/userContent folder and these files are served from yourhost/jenkins/userContent.

**Q56. Setup one Jenkins project?**

Click ‘New Item’, ‘Name’ the project, add the description > click ‘git’ and give ‘git repository URL’ > Click ‘add credentials’ > ‘Secret Text’, take that from GitHub ‘Setting’, under ‘Developer Settings’, if ‘Token’ is not there then we can generate a new. Same will ‘Secret Text’ needs to update in Jenkins. Click ‘GitHub hook trigger for GITScm polling’, then update ‘Goals and options’ : test -P"$Profile" -DBrowser="$browserName"

**Q57. Schedule triggers:**

• \* \* \* \* \* (min, hour, date, month, weekDay)

**Q59. What is your testing strategy or approach for web portals ?**

**1.Understand Requirements**

\* Gather and review functional specs, UI/UX designs, and user stories.

\* Identify the core modules and user roles (admin, client, user etc.)

\* Clarify business logic and expected user flows.

**2.Creating the Test Plan**

\* Define scope (what will and won’t be tested).

\* Identify test objectives and features to be tested.

**3.Define the types of testing you will do (see below).**

\* Plan test environments, browsers, and devices likes mobiles.

\* Create a testing schedule and assign roles.

**4.Types of Testing to Perform**

**A.Functional Testing**

\* Verify all UI elements, links, buttons, forms, menus, checkboxes, radio button, dropdowns, field boxes

\* Test user login/signups.

**B.UI/UX Testing**

\* Check design alignment, fonts, colors, and responsiveness.

\* Validate error messages, navigation consistency, and user feedback.

\* Perform accessibility testing like screen reader, mouse and keyboard actions.

**C.Cross-Browser and Cross-Device Testing**

\* Test on major browsers: Chrome, Firefox, Edge

\* Test on desktops, mobiles, ipads etc

**E.Boundary Testing**

\* Range testing, maximum dependents add, date range testing

\* Validate password policies and session timeouts.

**F.Integration Testing**

**Test integration with ULR links, third-party sign-ons**

**G.Database Testing**

Check if data is stored, updated, and deleted correctly.

**4.Test Case Design**

\* Write clear, reusable test cases for each functionality.

\* Along with positive, add negative scenarios also.

**5.Identify the testing tools which is needed, like we have in-house tools like catalyst(A), Prueba(M) or selenium**

**6.Test Execution**

\* Execute test cases manually or with automation tools.

\* Log defects with detailed reproduction steps and screenshots.

**7.Track defect life cycle (New → Assigned → Fixed → Retested → Closed).**

**8.Use CI/CD tools (e.g., Jenkins, GitHub Actions) for regular test runs.**

**7.Regression Testing**

\* Re-test existing functionality after every change or release.

\* Ensure new code hasn’t broken old features.

**8.Reporting & Feedback**

\* Share daily/weekly test reports and bug summaries.

**9.Tracking tools like Jira etch can be used.**

**10.User Acceptance Testing (UAT)**

\* Conduct testing with real users or clients.

**11.Post-Go-Live Monitoring**

\* Monitor portal performance and error logs and reporting to developers.

**Q60.What is Hybrid framework and its challenges:**

A Hybrid Framework in Selenium is a combination of two or more automation frameworks like:

Data Driven, Pagefactory, Page Object Model, TestNG

Currently using Pagefactory design pattern and the components which we are using are:

TestNG, Page classes with pagefactory, log4j reporting for debug/references, properties files we are using for global parameters, for data we have json/excel file integration, for client reporting we are using Extent Reports

**Q61. What is Scrum, its components and stages ?**

Scrum is an Agile framework used for developing, delivering, and maintaining complex projects, especially in software development. Scrum is a lightweight, iterative process where work is divided into small chunks called Sprints.

It encourages continuous feedback, quick adaptability to change, and frequent delivery of working software.

**Key Components of Scrum**

**1. Scrum Roles (Who’s involved?)**

Product Owner: Defines the features and prioritizes the Product Backlog.

Scrum Master: Facilitates the Scrum process, removes blockers, ensures Scrum is followed.

Development Team: Cross-functional team that does the actual work (design, development, testing, etc.).

**Event Description for the Development/Testing team**

\* Sprint: A fixed time-box (usually 1–4 weeks) where a set of work is completed and made ready for review.

\* Sprint Planning: Meeting at the start of the Sprint where the team decides what they will work on.

\* Daily Scrum (Stand-up): 15-minute daily meeting to inspect progress and plan for the next 24 hours.

\* Sprint Review: Held at the end of the Sprint to demonstrate the work done and get feedback.

\* Sprint Retrospective : After the Review, the team reflects on the process and identifies improvements.

**Scrum Project Lifecycle (Stages)**

Here’s how a Scrum-based project typically flows:

\* Product Backlog Creation

\* Product Owner gathers and prioritizes requirements.

\* Sprint Planning

\* Selects a subset of backlog items to work on in the Sprint.

\* Sprint Execution

\* Team works on tasks; Daily Scrum meetings are held.

\* Sprint Review

\* Team show ‘Prove It’ or demo session about the progress.

\* Sprint Retrospective

\* Team discusses what went well, what didn’t, and how to improve.

\* Next Sprint Starts

\* Repeat the cycle.

**Q62.What is ‘Use Case’ in software testing ?**

In software testing, a use case refers to a detailed description of how a user interacts with the software to achieve a specific goal. It helps testers understand how the system should behave during the testing flow in real-world scenarios.

**Q63.What is a Defect Triage meeting?**

\*A newly created bug is analyzed on the flow of the test case. A quick debug is done to check if that is real bug or its manual process. If it’s a manual process error, then it is immediately cancelled or deferred.

\*If that is debugged as correct error in the flow, then we need to hit the RCA.

\*Once RCA is done, then we identify how many accounts have been impacted.

\*Identify what is Severity and Priority of the bug.

\*If it’s on Executive accounts or most accounts then it will always be top priority.

\*If it’s on payroll deduction side issue, then it will be critical severity to fix it.

\*Bug is immediately assigned to the development team to fix it with a due date.

\*Testing child task for the same bug will be assigned to a tester for Smoke testing.

\*Keep tracking the bug status in every Defect Triage meeting.

Defect = "Something is not working as per the requirement."

Bug = "There’s a flaw in the code causing that defect."

**Q64.When to Entry in and Exit from the test plan?**

**Entry:**

\*Test Plan and Test Cases are reviewed and approved.

\* Test environment is set up and stable.

\* Test data has been prepared.

\* Build is deployed.

\* All dependencies are resolved, like integrations of code etc

**Exit:**

\*100% test case execution

\*All major defects fixed and closed

\*Smoke testing completed

\*Regression testing completed

\*Test summary reports approved by QA Manager

**Q65. How do you define quality in a software project?**

Quality in a software project should meet the standards of high accuracy and should meets or exceeds customer expectations, while also being reliable, usable, maintainable, and secured

Quality is not just about being bug-free – it’s about delivering the right functionality, with a great user experience, that performs well under real-world scenarios.

In practical terms, I ensure quality by focusing on three key pillars:

**\*Process Quality** – Having clear requirements, well-defined acceptance criteria, traceable test cases, and adherence to development/testing best practices.

**\*Product Quality** – Ensuring the software behaves as expected (functional), is intuitive (UX), performs well (non-functional), and has minimal critical bugs.

**\*Customer Satisfaction** – At the end of the day, if the software solves the user's problem effectively and delightfully, that’s true quality.

**Q66. How do you stay updated with current QA trends and tools?**

As a QA professional, I believe staying updated with evolving tools and trends is essential to remain effective and competitive. Here's how I stay current:

**1. Online Learning Platforms:**

I regularly take courses on platforms like Udemy, new tools, automation frameworks, and best practices in testing (e.g., Cypress, Playwright, performance testing tools).

**2. Community & Forums:**

I engage in discussions on LinkedIn groups, Stack Overflow. These platforms provide practical solutions and exposed all the solution which industries of projects have.

**Q67. How do you plan and estimate testing efforts for a project?**

I thoroughly review the business requirements, user stories, or functional specs.

I collaborate with BAs, developers, and product owners to clarify assumptions and understand acceptance criteria.

I break the testing work into major activities:

\*Test case design

\*Test data preparation

\*Test environment setup

\*Test execution

\*Regression testing

\*Defect retesting and triage meetings

Estimating time for each testing task and summing them.

Referring to similar past projects for realistic estimation.

Team-based estimation: Conducting estimation sessions with QA engineers using techniques like Planning.

Resource skill levels and Risk areas and potential rework.

**Q68. How do you ensure full test coverage?**

Ensuring full test coverage is about validating that all functional and non-functional requirements have been tested thoroughly, so nothing critical slips through to production

\*I create and maintain an **RTM (Requirements Traceability Matrix)** to map each requirement or user story to test cases.

\*This helps confirm that every requirement has at least one corresponding test and that gaps are visible.

\*RTM also helps with impact analysis when requirements change.

\*I apply boundary value analysis, and decision tables to design comprehensive test cases.

\*Integration and Regression should have thoroughly executed.

\*I track testing reports.

\*Test case execution blueprints through Jira tool helps a lot to track the status.

\*I conduct test case reviews with developers and BAs to uncover any missing flows.

\*Regular retrospectives help the team identify missed coverage in past releases and improve continuously. Internal prove it sessions are done with QA Managers and Developers and are open to feedback.

**Q69. How do you handle scope changes in the middle of a test cycle?**

Scope changes during a test cycle are common, especially in Agile environments.

**\*Analyze the Change Request** — is it a new feature, a change to an existing one, or a bug fix with wide impact?

\*I check where we are in the test cycle and how this change may affect timelines, existing test cases, areas/modules etc.

\*Do existing test cases need to change?, Will it require rework, re-testing, or regression?

\*I also assess test data, environment readiness, and resource availability.

\*Based on the impact, I update test estimates and priorities.

\*I communicate the impact clearly to Project Managers, Product Owners, and Dev Leads

**Q70. How do you allocate tasks within your QA team?**

I allocate tasks based on a combination of skillset, experience, workload balance.

\*Evaluate Team Members’ Strengths

\*I match tasks to individual skills and interests:

**UI testing** → those strong in visual validation.

**API testing** → those comfortable with tools like Postman or REST Assured.

**Automation** → based on tool expertise.

I also consider learning opportunities, giving newer team members simpler tasks to grow confidence.

**Q71. How do you mentor junior testers?**

\*I begin by helping new testers understand the domain, product features, tools, and processes we follow.

\*I walk them through real test cases, our test management tool like jira and explain the SDLC/STLC lifecycle we use.

\*I define short-term learning goals

\*I often pair juniors with experienced testers for shadow testing, exploratory testing, or to observe test case reviews.

**Q72. How do you handle underperforming team members?**

\*I schedule a confidential discussion to understand what’s going on:

\*Are there personal challenges, lack of clarity, or skill gaps?

\*I listen carefully and avoid being judgmental. Often, there are valid reasons.

\*Set Clear and Improvement Goals

\*Based on the root cause, I create a short improvement plan:

\*I also offer support — extra training, pairing with a senior, or mentorship.

\*Monitor Progress and Provide Ongoing Feedback

\*I give positive reinforcement when they improve, and honest feedback when gaps remain.

**Q73. How do you handle critical defects found just before a release?**

\*I immediately validate whether the defect is Consistent and reproducible.

\*If related to payments, deductions then its high priority defect.

\*If it’s ambiguous, I involve Dev or BA to confirm severity and impact.

\*What modules or users are affected?

\*Notify Stakeholders Immediately

\*I escalate the issue to Dev, Product Owner, Project Manager, and Release Manager — with a clear summary of the bug.

\*If the team decides to fix before release, I ensure Dev prioritizes it.

\*Allocate testers for quick retest + regression on impacted areas

\*Release this if client allows an off cycle or plan for next iteration.

\*After the release, I conduct a root cause analysis (RCA):

\*Why wasn’t the defect caught earlier?

\*A proper/constructive feedback should be given to the tester.

**Q74. How do you ensure defect leakage is minimized in production?**

\*I ensure comprehensive test cases for:

\*Functional, UI, regression, integration, and end-to-end scenarios.

\*I use traceability matrices to verify that all requirements have corresponding test cases.

**Q75. How do you decide when a build is ready for release?**

\*I make sure the build has met exit criteria defined in the test plan.

\*All high-severity bugs are fixed

\*100% of planned test cases are executed.

\*Prove it meetings already conducted.

\*Sign-Off given by the client.

**Q76. How do you communicate risk to stakeholders?**

\*Identify the risk type, it is a bug, test delay or project delay.

\*Assess the risk impact on high, medium or low on the basis of severity.

\*Explain how it affects users, deadlines, or payroll etc

\*Avoid technical terms with non-technical stakeholders

\*Offer mitigation strategies like delay release, partial rollout, important ones to release first

\*Present risks early and let stakeholders help decide next steps

\*Update regularly to all the stakeholders.

\*Document everything and future actions.

**Q77. Describe a challenging QA project and how you managed it.**

\*Give Links challenge example

**Q78. How do you handle disagreements within your team or with developers?**

\*Listen actively and stay calm

\*Understand everyone’s perspective before responding

\*Focus on the issue, not the person

\*Address on facts and data, not on assumptions.

\*Create a safe space where team members can voice concerns freely

\*Use one-on-one or team meetings to discuss issues openly

\*Identify shared goals (e.g., product quality, timely delivery)

\*Emphasize collaboration rather than conflict

\*Let evidence drive resolution, not opinions, we can set examples from the past, how we together delivered the project.

\*Bring in project managers if disagreement persists.

\*Summarize decisions and action items to avoid confusion

\*Check back to ensure resolution is effective

**Q79. Why do you want to be a QA Lead here?**

Give company history + opportunity and skill sets.

**Q80. What would your first 30/60/90 days look like in this role?**

**First 30 Days: Learning & Observing**

\*Understand the company’s products, domain, and QA processes

\*Meet with key stakeholders: developers, product owners, project managers, and QA team members

\*Review existing test plans, automation suites, defect reports, and release cycles

\*Identify immediate challenges, gaps, and provide your inputs too.

\*Familiarize with tools, environments, and workflows used by the team

**Next 30 Days (31-60): Planning & Improving**

\*Collaborate with the team to define or refine QA strategies and standards

\*Start addressing high-priority issues found in initial review (e.g., flaky tests, coverage gaps)

\*Implement or enhance automation frameworks and CI/CD integration where needed

\*Facilitate knowledge-sharing sessions or training for team skill development

**Final 30 Days (61-90): Leading & Optimizing**

\*Lead test planning for upcoming releases and ensure robust risk-based testing

\*Establish regular defect triage meetings and quality gates with cross-functional teams

\*Mentor and guide junior testers to improve overall team performance

\*Work with product and development teams to align on continuous improvement feedback loops

\*Work/discuss and plan with onshore/client team for upcoming projects

**Q81.What is your biggest achievement as a QA professional?**

My biggest achievement is learning automation testing. I started in manual testing, but because of project needs and resource crunch, I was moved to the automation team. It was very difficult at first — I honestly didn’t understand anything. I worked closely with my team members and learned Java, Selenium scripts, what they do, and how to connect them with projects. I didn’t get any formal training; I learned everything while working on a live project.

Many times, clients asked me about the testing strategies for the project. At first, I found it hard to answer and had to ask my lead for help. But now, I feel confident and comfortable in this role. This is my biggest achievement of parallel learning, working and delivering.

**Q82.What is STLC?**

STLC (Software Testing Life Cycle) is a systematic process that defines various stages of testing to ensure software quality.

**STLC is Iterative**

\*It often runs in parallel with SDLC (Software Development Life Cycle)

\*In Agile, STLC happens within each sprint

**Requirement Analysis**

\*Understand and analyze the requirements from a testing point of view

\*Identify testable requirements and raise any doubts or gaps

**Test Planning**

\*Define the test strategy, scope, objectives, effort estimation, and resource planning

\*Prepare a Test Plan document

**Test Case Design**

\*Write detailed test cases, test scenarios, and test data

\*Ensure coverage for functional, non-functional, and edge cases

**Test Environment Setup**

\*Prepare the hardware, software, and network configuration where testing will be done

\*May include setting up databases, servers, and test tools

**Test Execution**

\*Execute the test cases as per the test plan

**Log defects**

\*If actual results don’t match expected results

\*Defect Reporting & Tracking

\*Report bugs using a defect management tool (like JIRA)

\*Retest and verify once issues are fixed

**Test Closure**

\*Evaluate test completion criteria

\*Prepare Test Summary Report and lesson learned.

**Q83.What is difference between UI and UX?**

UI (User Interface) focuses on the visual elements and interactive components of a product, like buttons, icons, and layouts, while UX (User Experience) focuses on the overall user journey and how the user interacts with and feels about the product. Essentially, UI is about "how it looks" and UX is about "how it feels in the flow".

**Q84.What is User Stories/story points?**

User stories are short, informal descriptions of a feature from the end-user's perspective, acting as the smallest unit of work in the framework.

**Q85.What is Risk Based Testing/or during resource shortage:**

Risk-based testing (RBT) is a software testing approach that prioritizes testing efforts based on the potential risks associated with different features or components of an application. It focuses on identifying and assessing risks, then focuses testing resources on areas where failures would have the highest impact or are most likely to occur. This helps to optimize testing efforts and ensure that the most critical parts of the software are thoroughly tested, even when resources are limited.

Key Concepts of RBT:

**• Risk Identification:**

Identifying potential risks that could lead to defects, failures, or security breaches.

**• Risk Assessment:**

Analyzing the identified risks to determine their likelihood of occurring and their potential impact on the system.

**• Prioritization:**

Prioritizing testing efforts based on the assessed risks, focusing on the areas with the highest risk.

**• Resource Allocation:**

Allocating resources (time, effort, budget) to the areas that require the most testing.

Benefits of RBT:

**• Optimized Testing:**

Focus on the most critical areas, saving time and resources.

**• Improved Quality:**

Ensures that the most important aspects of the software are thoroughly tested, reducing the risk of defects.

**• Cost Reduction:**

By focusing on the highest-risk areas, RBT can help to reduce overall testing costs.

**Q86.What is ROI?**

It helps organizations determine whether the time, money, and resources spent on testing are delivering valuable results.

**Key Points:**

• **Time savings**: Automation can significantly reduce the time it takes to execute tests, especially for repetitive tasks.

• **Reduced manual effort**: Automated tests can free up QA engineers to focus on more complex tasks.

• **Improved test coverage**: Automation can cover more test cases, leading to better defect detection.

• **Faster feedback**: Automated tests can provide faster feedback on code changes, leading to quicker bug fixes.

• **Reduced costs**: Automation can lead to significant cost savings in terms of labor, resources, and time.

• **Increased product quality**: More thorough testing can lead to higher-quality software.

**Q87.What approach you will follow for Automating 100 test cases:**

It requires strategy, prioritization, and tool efficiency to manage time and complexity.

Split into categories:

| **Priority** | **Test Case Type** | **Reason** |
| --- | --- | --- |
| **High** | Smoke, Sanity, Core Flows | Must run in **every release** |
| **Medium** | Common Regression | Covers **frequently used features** |
| **Low** | Rare or Edge Cases | Can be **automated later** or remain **manual** |

**Chose the right automation tool and framework– Selenium/Api/Postman/Hybrid**

**Estimate Time and Create a Plan**

• Simple TC = 1 hr

• Medium = 2 hrs

• Complex = 3+ hrs

• 100 cases = ~150–200 hours

**Features to include:**

• Reusable functions (login, navigation, common actions)

• Config-driven data (via JSON/CSV)

• Test report generation

• Cross-browser capability (if needed)

**Q88.What are best practices of coding:**

**1-Code Readability:**

• **Meaningful Naming:**

Use descriptive names for variables, functions, classes, and other identifiers to clearly convey their purpose.

• **Consistent Formatting:**

Use indentation, spaces, and line breaks consistently throughout the code to improve readability.

• **Comments:**

Explain the "why" behind complex logic, not just the "what" the code does.

• **Avoid Deep Nesting:**

Break down complex functions into simpler ones to reduce nesting levels and improve readability.

**2-Code Organization and Structure:**

**• Modular Design:**

Structure code into reusable modules or functions to promote code reuse and maintainability.

• **Single Responsibility Principle:**

Each function or class should have a single, well-defined purpose.

• **Don't Repeat Yourself (DRY):**

Avoid duplicating code and instead, create reusable components or functions.

• **Clear Structure:**

Organize code logically with consistent folder structures and file naming conventions.

**3-Error Handling and Testing:**

• **Robust Error Handling:**

Implement error handling mechanisms (e.g., try-catch blocks) to gracefully handle exceptions and unexpected input.

**• Thorough Testing**:

Write unit tests, integration tests, and other tests to ensure code functionality and quality.

**• Test-Driven Development (TDD):**

Write tests before writing the code, which can lead to more focused and robust development.

**4-Version Control and Collaboration:**

**• Use Version Control Systems:**

Utilize tools like Git to track changes, collaborate with others, and revert to previous versions if needed.

**• Collaborative Workflows:**

Utilize branching, merging, and pull requests to facilitate team collaboration.

**• Maintain a Clean Commit History:**

Write meaningful commit messages and avoid adding unrelated changes to a single commit.

**5-Optimization and Performance:**

• Efficient Data Processing:

Avoid unnecessary loops and iterations, and use techniques like memorization or caching to optimize data processing.

**Q89-What is the architecture of Selenium framework?**

[ Test Data Files (Excel/CSV/JSON) ]

↓

[ Data Provider / Reader ]

↓

[Test Cases] ---> [ Test Scripts Layer (TestNG/JUnit) ] ---> [ Reports ]

↓

[ Page Object Model (POM) Layer ]

↓

[ Selenium WebDriver Layer ]

↓

[ Browsers: Chrome, Firefox, Edge, etc. ]

**Q90-What is the default polling in Selenium waits?**

500 milliseconds

**Q91-Can we put customized polling into Explicit wait?**

Yes - WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

wait.pollingEvery(Duration.ofMillis(200)); // Poll every 200ms

wait.until(ExpectedConditions.elementToBeClickable(By.id("submit")));

**Q92-Why need of FluentWait when already have Explicit Wait?**

Explicit Wait uses a default polling interval of 500 milliseconds. On the other hand, FluentWait allows you to define a custom polling frequency, giving you better control over how often Selenium checks for a condition. Unlike Explicit Wait, which handles only basic timeouts and limited exceptions, FluentWait can be configured to ignore specific exceptions (like NoSuchElementException or StaleElementReferenceException) during the wait period. This makes FluentWait especially useful for handling flaky test cases, as it retries the condition without immediately failing the script.

**Q93-What is JIT (Just In Time) in Java ?**

Just-In-Time (JIT) Compilation improves program speed by compiling code during runtime, only when needed. First, Java code is compiled into bytecode. Then, the JIT compiler turns frequently used bytecode into machine code while the program runs. This machine code is cached, so next time it runs faster.

**Q94-What is Java Virtual Machine ?**

The Java Virtual Machine (JVM) is a program that runs Java code. When you write Java code and compile it, it becomes bytecode (a .class file). The JVM takes this bytecode and turns it into machine code that your computer can understand and run. JVM lets you write code once and run it anywhere — on Windows, Linux, or Mac — as long as a JVM is installed. It also takes care of things like memory management (garbage collection), error handling, multithreading, and security.

**Q95-What is difference between JVM and JIT?**

JVM is a runtime engine that runs Java bytecode, while JIT is a part of JVM that compiles bytecode into machine code. JVM Loads, verifies, and executes Java code whereas JIT Speeds up code execution by compiling parts during runtime.

**Q96-What are ceremonies in Agile ?**

They ensure communication, collaboration, and continuous improvement throughout the development cycle.

**1. Sprint Planning:** The team meets at the start of the sprint to decide what tasks to work on, based on priority and capacity. The Product Owner shares the top items, and the team commits to them.

**2. Daily Stand-up:** A quick 15-30 minute daily meeting where team members share what they did yesterday, what they plan to do today, and if they have any blockers.

**3. Sprint Review:** At the end of the sprint, the team shows the completed work to stakeholders and gets feedback.

**4. Sprint Retrospective:** After the review, the team discusses what went well, what didn’t, and how to improve in the next sprint.

**5. Backlog Refinement (optional but helpful):** The team and Product Owner review and update the backlog to keep it clear, estimated, and ready for future sprints.

**Q97-What is the difference between ArrayList and LinkedList?**

In Java, ArrayList and LinkedList are both used to store collections, but they differ in how they manage data. ArrayList uses a dynamic array, making it faster for accessing elements by index. In contrast, LinkedList uses a doubly linked list(insertion/deletion from both ends), which makes it better for frequent insertions and deletions, especially in the middle or beginning of the list. While ArrayList offers better performance for read-heavy operations, LinkedList is more efficient when the focus is on adding or removing elements frequently.

Follow-up: When would you use one over the other in test data handling?

Use ArrayList when you need fast access to test data by index, like reading or looping through values. Use LinkedList when you frequently add or remove test data, especially at the start or middle of the list.

**Q98-How do you handle NullPointerException in Java?**

A NullPointerException occurs when you try to use a variable that hasn’t been initialized or its null. We can initialize or we can use:

if (myObject != null) {

myObject.doSomething();

}

or - We can use Try Catch to handle it.

Have you faced this during test execution? How did you debug it?

When the error happens, Java tells you exactly where it occurred, that is line number error.

**Q99-How does HashMap work internally in Java?**

In Java, a HashMap stores data as key-value pairs and uses hashing for fast lookup. When you add a key, Java calls its hashCode() method to get a hash, which is used to find an index in an internal bucket array. The key-value pair is stored at that index. If multiple keys map to the same index (a collision), the entries are stored using a LinkedList or Red-Black Tree. When retrieving a value, Java recalculates the hash, goes to the correct bucket, and uses .equals() to find the exact key.

Follow-up: How do you use it in your test framework?

Give the example of Json Files which is stored like a HashMap or Data Providers.

**Q100-What is the difference between throw and throws in Java?**

In Java, throw is used to manually create and trigger an exception during program execution. For example, you might write throw new IllegalArgumentException("Invalid input"); to raise an error when a condition isn't met. On the other hand, throws is used in a method declaration to indicate that the method might cause an exception, and it tells the caller to handle it. For instance, a method like public void readFile() throws IOException declares that it may throw an IOException while running.

**Q101-How do you use exception handling in Selenium tests?**

You can wrap Selenium actions in try-catch to handle exceptions like NoSuchElementException, TimeoutException, etc. Finally can be used to:

finally {

driver.quit(); // Ensures browser closes

}

Or

finally {

System.out.println("Test completed. Cleaning up...");

}

**Q102-What is the difference between List, Set, and Map interfaces?**

In Java, List, Set, and Map are core interfaces with different uses. A List allows duplicates and maintains insertion order with indexed access. A Set ensures unique elements and may not keep order. A Map stores key-value pairs where keys are unique, and values can repeat. While List and Set store elements, Map is used for fast lookup using keys. Example – List (ArrayList), Set(HashSet), Map(HashMap).

Where have you used these in your automation projects?

List– This is generally when we have list of WebElements identified and we need to traverse through all.

Set– This is generally used in when we have multiple windows to handle. The we can have “driver.getWindowHandles()”, this has return code Set<String>.

Map– This is generally when we have multiple rounds of data to be used for same kind of test cases, for example- we need to check login credentials multiple times. Then this can be used under dataProviders.

**Q103-How do you create and use Enum in Automation?**

In **Java**, an **enum** (short for enumeration) is a **special data type** used to **define a fixed set of constant values**.

Give example of multiple drop downs and using the Enum in switch.

public enum dropDownType {

byVisibleText, byIndex, byValue

}

**Q104-What is Bubble Sort in java?**

Bubble Sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The larger elements "bubble" to the top (end) of the list with each pass. It continues these passes until no more swaps are needed, indicating the list is sorted.

**Q105-What is Xpath and its different types?**

**XPath(XML Path)** is a way to find elements on a web page by navigating through the page’s HTML structure. In **Selenium**, it is mainly used when basic locators like **id, name, or className** don’t work well or are not reliable. XPath can move through the page both forward (to child elements) and backward (to parent or ancestor elements), which makes it useful for finding elements that are dynamic or hard to locate.

There are two types of Xpaths :

**1-Absolute**- Which navigates from parent to child using tag nodes.

**2-Relative**- Begins from **anywhere** in the DOM, using // to search through the structure.

**Q106- Explain how annotations are used in Selenium from the point of view of a manual tester.**

In Selenium, **annotations** are special @ tags used before methods or classes to guide the test framework (like TestNG or JUnit) on how to run tests. They control the test flow—what runs **before**, **during**, and **after** a test.

If you're moving from manual to automation testing, understanding annotations is important because it helps you see how automated tests are structured. You'll learn what parts of the test run first and what runs last, as well as how setup and cleanup steps are handled automatically. This knowledge allows you to review or even create test scripts in a logical order, without needing deep coding expertise.

**Q107-What is difference between .get() and .navigate() ?**

**.get()** and **.navigate()** both methods open the webpage, but **.navigate()** provides additional controls like **.back(), .forward() an .refresh();**

**Q108-How to handle multiple windows?**

**driver.getWindowHandles()** can be used to get all open window handles, and the **Iterator** class can help loop through them to manage user flow. A while loop can be used to go through each child window, and you can switch back to the parent window using **driver.getWindowHandle().**

**Q109- What are Mouse Actions in Selenium?**

**Mouse actions** in Selenium are used to simulate **user interactions** like:

Hovering, Clicking, Right Click, Double Click, Drag and Drop, Move to Element etc. It is performed through Actions class.

**Q110- What are different locators are there in Selenium?**

By.id, By.name, By.tagName, By.className, By.linkText, By.partialLinkText, By.xpath, By.cssSelector

**Q111- What are exceptions in Java and how they are handled ? Example of Exceptions:**

In Java, an **exception** is an **unexpected event or error** that occurs during the execution of a program. It **disrupts the normal flow** and results fail. We can use Try/Catch block to handle this situation.

**NullPointerException**- to **access or use something that doesn't exist** (it's null) — like calling a method, accessing a variable, or using .length on a null object.

**ArithmeticException-** is a **runtime exception** in Java that occurs when an **illegal arithmetic operation** is performed — most commonly, **division by zero**.

**ArrayIndexOutOfBoundsException** - occurs when you try to access an index that is outside the valid range — such as accessing an index that is greater than the size of an array or ArrayList.

**Q112- What keyboard actions do selenium support and how to use them?**

One most common keyboard action is sendKeys(“Selenium”); , keys.CONTROL, keys.ENTER, keys.TAB, keys.BACK\_SPACE

We can use **Actions** class to perform various Keyboard actions:

**Example: “Contro+A”**

Actions actions = new Actions(driver);

actions.click(input)

.keyDown(Keys.CONTROL)

.sendKeys("a")

.keyUp(Keys.CONTROL)

.perform();

I used **Keys.chord(Keys.CONTROL, Keys.ENTER)** and stored it in a string variable. This combination opens a link in a new tab. So, during iteration when multiple links need to be opened in new tabs, you can send this variable to each link, and it will open accordingly.

**Q113- Does Selenium support API testing? How to integrate ?**

**No, Selenium does not directly support API testing.**  
Selenium is a browser automation tool used for **UI testing** of web applications — it works only with **HTML DOM** through a browser.

While Selenium itself **can't perform API calls**, you can **combine Selenium with other tools/libraries** in your test framework to test APIs alongside UI.

**Q114- What is Test Plan and what components it should include?**

A **test plan** is a **formal document** that outlines the **strategy, scope, objectives, resources, schedule, and activities** required for testing a software application. It acts as a **roadmap** to ensure testing is planned, structured, and effective. A test plan defines **what needs to be tested**, **how it will be tested**, and **who will perform the testing**. It sets clear goals and timelines and ensures that the testing process aligns with overall business requirements.

**Key Components of a Test Plan:**

**Test Plan ID / Title** – Unique name.

**Introduction** – Overview of the project and test purpose and who is owner/tester.

**Scope of Testing** – What will and won’t be tested.

**Objectives** – Goals of testing, also if it mapped to RTM (Requirement Traceability Matrix)

**Test Strategy** – Manual or automation approach, types of testing.

**Test Environment** – Required tools and software.

**Test Deliverables** – Test cases, bug reports, summary reports, etc.

**Entry and Exit Criteria** – When to start and stop testing.

**Schedule / Timeline** – Estimated testing duration.

**Risks and Mitigation** – Possible risks and how to handle them.

**Tools** – Tools used for test execution and tracking.

**Approvals** – Stakeholders who approve the plan.

**Q115- What is cross browser testing?**

**Cross Browser Testing** is the process of verifying that your **web application works correctly across different web browsers, browser versions, and operating systems**.

Different browsers (like **Chrome, Firefox, Safari, Edge**) and devices can render HTML, CSS, and JavaScript **differently**. Cross browser ensures testing looks consistent, functions properly and provides good user experience.

**Q116- What is Grid in selenium?**

**Selenium Grid** is a part of the Selenium Suite that allows you to run **tests in parallel** across **multiple machines, browsers, and operating systems**.

It is used for **distributed test execution** — especially helpful for **cross-browser** and **cross-platform** testing. This can be good option when we have limited resources and time to deliver the testing results.

**Q117- What is WebDriverManager, dependency of 5.1?**

It has feature to launch browser without System.SetProperty();

**Q118- Difference between Long and Short in Java?**

A **short** uses 16 bits of memory, an **int** uses 32 bits, and a **long** uses 64 bits.

**Q119- What is Object class in Java?**

Object class is the parent class of all the classes in Java by default. Any class created and has methods then this class is parent.

**Q120- What is Super class and Sub class in Java?**

The derived class from where it has been derived is called Super class and the derived class is Sub class or child class.

**Q121- Can main method be overridden in Java?**

No, the main() method cannot be overridden because it is a static method. In Java, static methods belong to the class itself rather than to individual instances. Since method overriding only applies to instance methods, static methods like main() are not eligible for overriding.

**Q122- What is Wrapper function in Java?**

A wrapper function is a block of code organized into a separate method to perform a specific operation. It serves as a utility that can be reused whenever needed. This method may or may not include a return type, depending on its purpose.

**Q123- Why strings are immutable?**

In Java, Strings are immutable, meaning they can't be changed once created.  
This is important because strings often store sensitive data like usernames or URLs.  
If they were mutable, attackers could modify them after creation, causing security risks.  
Java also uses a String pool to save memory by reusing string values.  
If strings were mutable, changing one reference could affect others sharing the same value.

**Example:**

**String s = "Java";**

**s.concat(" Programming");**

**System.out.println(s); // Output: Java**

**s = s.concat(" Programming");**

**System.out.println(s); // Now it prints "Java Programming"**

**Q124- Difference between StringBuilder() and StringBuffer() ?**

Both **StringBuilder and StringBuffer** are mutable classes used for operations like appending, reversing, and concatenating strings. The key difference is that StringBuffer is **thread-safe**, while StringBuilder is not. Use case of StringBuilder can be single **thread** and for StringBuffer it can be **multi-thread**.

**Q125- What is Zero sprint ?**

**Zero Sprint** (also called **Sprint 0**) is the **preparation phase** that occurs **before the first official sprint** begins in Agile development. It is the assessment of upcoming and various sprint phases, how to plan and achieve the goals.

**Q126- What is return type of dataProviders?**

Object [][]

**Q127- What is difference between Exceptions and Errors?**

Exceptions are thrown by a program when something goes wrong during execution. They can be handled using try and catch blocks to prevent the program from crashing. Exceptions can be either checked or unchecked, and are usually recoverable.

Errors, on the other hand, are serious issues returned by the runtime environment, such as system failures or lack of system resource. They are unchecked and generally not recoverable, as they are not detected by the compiler.

**Example:**

**Exception** – NoSuchElementException

**Error** – OutOfMemoryError: OutOfMemoryError is a **runtime error** in Java that occurs when the **Java Virtual Machine (JVM) runs out of memory** and cannot allocate space for new objects. **Too many objects** are created without being released. **Large files** or data structures are loaded into memory.

**Q128- How Objects are released in Java to regain space?**

In Java, memory management is handled automatically by the **Garbage Collector (GC)**. However, you can help the GC **identify unused objects** by following best practices to release them.

**Q129- What is Java heap?**

The **Java Heap** is the **area of memory** used by the Java Virtual Machine (**JVM**) to **store objects** created during the execution of a program.

**Q130- What is testNG pluggin in Jenkins?**

**TestNG Results**

**Q131-What is Surefire Pluggin?**

It is responsible for running the test when testNG is used.

**Q132- What is difference between Test Case and Test Scenario ?**

A **test case** is a set of specific actions designed to verify the functionality or features of an application, including detailed steps on how to perform the test.  
A **test scenario** refers to a high-level description of what to test, focusing on end-to-end behavior without including detailed test steps.

**Q133- What is difference between System Testing and End to End Testing?**

System testing is done to check out on the whole system requirement specifications or functional requirement specifications. Example- To check if user can see the application.

End to End Testing is done at each level like Unit testing, Integration, Regression. It is to check if all the testing flow is error free in various environments. Example- To check if user can register on the application.

**Q134- What is default method in Java?**

A **default method** is a method in a **Java interface** that has a **method body** — introduced in **Java 8** to allow interfaces to have some **predefined behavior**. Helps add new methods to interfaces **without breaking** existing implementations. It can only be defined in interfaces.

**Q135- What is difference between Interface and Abstract class?**

**Interface:**

1-Has abstract and non abstract methods

2-Supports multiple inheritance. (Java does not support multiple inheritance, but with interfaces it allows multiple inheritance.)

3-Has Static and Final variables only

4-Interface can be implemented by ‘implements’ keyword

**Abstract Class:**

1-Has abstract, default and static methods.

2-No support of multiple inheritance. (Java does not support multiple inheritances because it can cause ambiguity.)

3-It has final/non-final/static/non-static variables.

4-Abstract classes can be inherited by ‘extends’ keyword.