**What agile ceremonies have you been a part of and what are your specific contributions to those ceremonies as an automation tester?**

SPRINT PLANNING:

The purpose of sprint planning is to agree on a goal for the next sprint and pick up a list of prioritized backlog items (according to the product owner) that the team can commit to achieve. The commitment can be in the form of story points, story sizes etc.

Sprint planning kicks off every sprint before starting the sprint. As a QA we show our confidence that we should be able to complete the testing for the items brought into sprint or committed stories and deliver value to the customer.

**Backlog Refinement:**

 is when the product owner and rest of the team review items on the backlog to ensure the backlog items contains appropriate AC and material and are prioritized according to business and are ready to be pulled in upcoming sprint. **From a Testing perspective** we as a tester go through the Acceptance Criteria , Mocks , Wire frames , Functionality of the user story whatever related information is available for better understanding and ask desired questions accordingly so we can give an estimate in how much amount of time we can test the user story (keeping in mind the complexity, time and unknowns that can come in our way )Also with this information we can write precise test cases and we test it accordingly with proper understanding of the story.

**Sprint Demo:**

In demo the work done in that specific sprint is demonstrated to product and stake holders. Also, how many points completed and what value is being delivered to customers when the deployment is done.

As QA you can explain the feature that has been developed currently how it behaves, functions and how the end user will use it so that the product and stake holders can understand it in a better way to know what value is being delivered to the customers

**Retrospective Meeting:** (Everybody is involved)

What went right and what went wrong in that specific sprint and what can we do to improve things/process. Also, how many points you completed and delivered to product.

**Daily Stand Up:** What we did yesterday what are we planning to do today any roadblocks/barriers/impediments which is not allowing us to fulfill our commitment

**DAILY ROUTINE**

**1. Day starts with attending the daily stand-up where i will update my team about what i did yesterday and my plans for today and if there is any blockage so i will highlight it.**

**2. After that my usual work would be to start with defect tracking, how many defects are fixed. Address high priority issues. Get more details from developers or my test lead for testing the modules if i don't know much about the changes they did.**

**3. Downloading or triggering the latest build**

**4. Reporting and discussion of new defects and assigning to respective developers.**

**5. Looking up stories that are available for testing or writing test cases.**

**6. My daily routine also varies day to day Depending upon which day of sprint it is. For instance, the routine varies on the day i had my grooming/refinement, planning, demo, retro or any other meetings in which my presence is required like bug triage or depositions.**

**What Attributes should be there in a good Story?**

**• Precise explanation of goal of the story**

**• Independent Acceptance Criteria that move towards achieving the story goal**

**• Any dependencies**

**• Wireframes or workflow diagram**

**• Api service details, if any**

**• Link to Epic**

**• Accessibility or analytics requirements if any**

**What testing’s generally you have done and at what phases?**

It all depends on which phase of development you are. If you are near to code freeze (Regression, Sanity and End to End Testing, Monkey Testing) But normally in sprints we do Functional Testing or UI Testing. Before DEMO if story is linked to any previous story so we will perform END to END testing or Sanity Testing.

**What are your thoughts or best practices on regression testing when on an agile team? How does automation fit in?**

Answer- This is extremely important with Agile development methodologies where software is developed incrementally and there is a constant potential that adding new features could break existing ones.

**What are your thoughts on regression testing? how may you approach regression testing?**

It depends upon the scope of Regression. Normally we check with the existing features in the start of the sprint and will perform regression for those features because the burden is less. Once the new stories have been developed and ready to test, we will conduct a progression on them individually first and perform regression with existing features so that we can verify that features are working fine. If a detailed regression is required against any feature set or code change than we might have to put some extra effort on a prioritized Test case basis, because at the end of the day it always gives you the assurance that features are working properly as per requirements.

**What are the different types of waits available in WebDriver?**

Implicit Wait 2. Explicit Wait

**Implicit Wait:** Implicit waits are used to provide a default waiting time (say 10 seconds) between each consecutive test step/command across the entire test script.

driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS) ;

**Explicit Wait**: Explicit waits are used to halt the execution till the time a particular condition is met, or the maximum time has elapsed. Unlike Implicit waits, explicit waits are applied for a particular instance only.

wait. until(ExpectedConditions.visibilityOfElementLocated(By.xpath

**API TESTING**:

We need to validate API response, the data we are receiving as per restful services or sending to server from client side are as per requirements or not.

**ENVIRONMENT:**

There are two types of environments, global and local. They define the scope of the variable to use it in the requests. Most commonly the variable is defined inside the environment for the same purpose. The most common variable we use is url because url is used in every request and changing it can be very time consuming. When we create an environment inside Postman, we can change the value of the key value pairs and the changes are reflected in our requests. An environment just provides boundaries to variables.

AES (Advanced Encryption Standard)

1. In the "Body" tab, add the data you want to encrypt.
2. In the "Headers" tab, add an Authorization header with a bearer token that will be used to authenticate the request.

**Collection in Postman?**

A collection in Postman can be imagined like a folder in your system. A collection is the grouping of requests, preferably of the similar types. It can be compared like the folder inside your system which has one type of files

.

**PARAMETER**: they are the inputs you need to pass along with API address to get a successful response.

**HEADERS**: headers are also mandatory parameters which are required to get proper response. You have to pass it along with your API’S.

**API METHODS:**

GET- If you want to fetch some data from server you will apply the GET method.

POST: If you want to submit some details to server then you will apply POST method.

PUT: If you want to modify an existing data you will apply PUT.

DELETE: If you want to remove a particular data then you will apply DELETE.

**Can you explain the inner workings of the JavaScript event loop? How does it handle asynchronous code execution?**

**Ans:** The JavaScript event loop is the mechanism that enables asynchronous code execution in JavaScript. It continuously checks the message queue for new messages and processes them one by one. Each message can be a task, a microtask, or a callback function.

The event loop operates on a single thread, but it can handle multiple tasks concurrently by breaking them down into small chunks of work and processing them in turn. This allows long-running tasks to be executed without blocking the main thread.

When a task is added to the message queue, it is first processed by the call stack. The call stack is a LIFO (last in, first out) data structure that tracks the current execution context of the JavaScript engine. When the call stack is empty, the event loop takes the next message from the message queue and processes it.

Microtasks are processed before other types of messages in the event queue. This means that Promise callbacks and other microtasks are executed before any other messages, ensuring that they have higher priority and are executed as soon as possible.

In summary, the JavaScript event loop is responsible for handling the processing of messages in the message queue, and it does so by continuously checking for new messages and processing them one by one, using a single thread of execution.

**Can you describe how garbage collection works in JavaScript? What are some techniques you can use to optimize memory usage in your automated tests?**

**Mark and sweep algorithem**

**Ans:** The JavaScript event loop operates on a single thread, which means that all JavaScript code is executed in a single call stack. This can create issues when executing long-running tasks that can block the main thread, leading to a poor user experience.

To address this issue, the event loop uses an approach called "asynchronous concurrency", which involves breaking long-running tasks into smaller chunks of work and processing them in turn. This allows the event loop to handle multiple tasks concurrently, without blocking the main thread.

The event loop achieves this by using asynchronous APIs, such as setTimeout, setInterval, and Promise, to delegate long-running tasks to the browser or operating system. When the task is completed, the event loop adds a message to the message queue, which is then processed in turn by the event loop.

This approach allows the event loop to handle long-running tasks without blocking the main thread, ensuring that the user interface remains responsive and that the user experience is not affected. It is a key feature of the JavaScript language that enables modern web applications to provide a fast and responsive user experience.

**Can you explain how you would use the Reflect API in JavaScript to dynamically modify object behavior? What are some use cases for using the Reflect API in your test automation code?**

**Ans:** The Reflect API in JavaScript provides a set of built-in methods that allow you to dynamically modify object behavior. These methods are designed to be used as functions that mirror the behavior of language constructs, such as new, delete, typeof, and instanceof.

One of the benefits of using the Reflect API in your test automation code is that it allows you to write more flexible and dynamic code. For example, you can use the Reflect.construct() method to dynamically create new instances of objects based on their constructors, or use the Reflect.get() and Reflect.set() methods to dynamically read and modify object properties.

Another use case for using the Reflect API in your test automation code is to provide a more secure and robust codebase. The Reflect API provides methods that mirror built-in language constructs, but with additional security features. For example, the Reflect.defineProperty() method provides a way to define properties that cannot be deleted or modified.

In summary, the Reflect API in JavaScript provides a set of built-in methods that allow you to dynamically modify object behavior. By using the Reflect API in your test automation code, you can write more flexible and dynamic code, as well as provide a more secure and robust codebase.

The Reflect API is a set of JavaScript built-in functions that allow you to intercept and modify object behavior at runtime. It provides a way to manipulate an object's properties and methods in a dynamic and flexible way.

Use cases for using the Reflect API in test automation code could include:

* Modifying the behavior of objects to simulate different scenarios during testing.
* Intercepting and logging API requests and responses.
* Intercepting and modifying browser events and behavior during automated testing of web applications.

**Can you explain how you would use the Promisify function in JavaScript to convert callback-based APIs into Promise-based APIs? What are some benefits of using Promisify in your automated tests?**

**Ans:** In JavaScript, many APIs use a callback-based approach to handle asynchronous operations. However, this can make it difficult to work with these APIs in a Promise-based environment, which is more modern and easier to use. The promisify function in JavaScript can help to solve this problem.

The promisify function is a utility function that takes a function that uses a callback-based API and returns a new function that uses a Promise-based API

In summary, the promisify function in JavaScript can be a valuable tool for converting callback-based APIs to Promise-based APIs. By doing so, you can simplify your code, improve error handling, and work more easily with modern JavaScript syntax.

The **Promisify** function in JavaScript is a utility function that allows you to convert callback-based APIs into Promise-based APIs. This can make your code cleaner and easier to read, and it can simplify error handling and flow control in asynchronous operations.

**Why and when Do We need Webdriver.io Instead of Selenium Webdriver**

WebDriverIO can do the same things that Selenium can, but also comes with lots of integrations with popular test automation tools and plugins. It is built for testing modern web applications written in React, Vue, Angular, Svelte, or other front-end frameworks built in NodeJS.

**At What stage of testing We need to use Sauce Lab?**

Sauce Labs is a full-scale testing platform that supports every stage of your software development and operations lifecycle to ensure complete digital confidence every time a user hits your website or app, regardless of the device, browser or operating system from which it is being accessed.

**What time zone does Sauce Labs use, and How to change it?**

Sauce Labs servers run **UTC,** Can be Changed Via  [Setting a time zone for Sauce Labs tests via Protractor](https://www.binarysludge.com/2014/08/19/setting-a-time-zone-for-sauce-labs-tests-via-protractor/" \l ":~:text=Sauce%20Labs%20servers%20run%20UTC.)