**Vijay Kumar Reddy**

* DOB – 10 AUG 1991
* GC ON 09 JUNE 2022
* 2012 June B. Tech
* 2012 July – 2015 Jan (CGI) – 2.5 Years
* Got Married in 2015 Feb
* Came to USA on 2015 March on H4 Visa
* 2015 June to 2016 Feb (H4 EAD Process)
* 2016 March to 2019 April (Progressive\_H4 EAD) – 3 Years
* 2019 May to 2020 Nov (Johnson & Johnso\_H4 EAD) – 1.5 Years
* 2020 Dec to 2022 Jan (Target\_GCEAD) – 1 year
* 2022 Feb to Till Now (Huntington\_GC) – 2 years.

Having 10 years of working experience as Senior QA Automation and Manual Engineer. I've been deeply involved in creating and enhancing automated testing solutions across various sectors, including **Banking**, E-**commerce**, **Insurance**, and **Healthcare.** I have experience working on **API Testing, UI Testing, Mobile Testing, Database Testing, and ETL testing.**

Certainly! Here are the top 10 interview questions and answers for the QA-Automation Lead position with a focus on Selenium automation, POS, BDD, Unix, SQL, API testing, and retail experience:

**1. Question: Can you elaborate on your experience in designing and building Selenium Test Automation Frameworks, especially in the context of your previous roles?**

Answer: In my previous roles, I have led the design and implementation of Selenium Test Automation Frameworks to ensure efficient and maintainable automated test suites. This involved creating modular and scalable frameworks that catered to the specific needs of the projects, facilitating easy maintenance and extension of test scripts.

**2. Question: How do you approach test automation for Point of Sale (POS) systems within the retail domain?**

Answer: When automating testing for POS systems in the retail domain, I focus on validating critical transactions, payment processing, and inventory synchronization. I leverage Selenium and Java to create test scenarios that cover various POS functionalities and ensure a seamless integration with the E-commerce platform.

**3. Question: Explain your experience with TDD (Test-Driven Development) and BDD (Behavior-Driven Development), particularly with Cucumber frameworks.**

Answer: I have hands-on experience implementing both TDD and BDD practices. In TDD, I write tests before the actual code to drive the development process. With BDD, I collaborate with stakeholders to define behavior in a readable format using tools like Cucumber. This ensures alignment between technical and non-technical team members.

4. Question: How do you ensure effective communication within your QA team and with other stakeholders, especially in an enterprise client environment?

Answer: Communication is critical, and I prioritize regular team meetings, status updates, and collaboration sessions. In an enterprise client setting, clear communication channels are established to ensure that project goals, timelines, and any challenges are effectively communicated and addressed.

5. Question: Could you share an example of a challenging situation you faced while leading QA automation for an enterprise client and how you handled it?

Answer: In a project for an enterprise client, we encountered a complex integration issue. I facilitated collaboration among team members, conducted a thorough root cause analysis, and implemented a resolution plan. Regular communication with the client kept them informed, and we successfully delivered the project on time.

6. Question: How do you integrate API testing into your overall test automation strategy, and what tools do you prefer for API testing?

Answer: API testing is an integral part of my automation strategy. I use tools like RestAssured for Java to validate API functionalities. API tests are designed to cover various scenarios, including data validation, error handling, and performance testing.

7. Question: Can you discuss your experience in utilizing Unix for testing purposes, and how it contributes to your automation efforts?

Answer: I have extensive experience working with Unix in testing environments. Unix commands are employed for tasks such as file manipulation, running scripts, and managing environments. This proficiency enhances automation efforts by allowing seamless integration into continuous integration pipelines.

8. Question: How do you keep yourself updated with the latest trends and advancements in the QA and automation field?

Answer: Staying updated is crucial, and I regularly participate in webinars, conferences, and online forums. Additionally, I subscribe to industry blogs and newsletters, engage in online communities, and experiment with new tools and technologies to stay ahead in the rapidly evolving QA landscape.

9. Question: In your experience, how have you implemented various testing methodologies to suit the specific needs of different projects?

Answer: Each project may have unique requirements, and I adapt testing methodologies accordingly. Whether it's Waterfall, Agile, or a hybrid approach, I ensure that the chosen methodology aligns with the project's goals. Flexibility is key, and I leverage my experience to implement the most effective testing approach.

10. Question: What considerations do you take into account when selecting test automation tools for a project, and how do you ensure the scalability of automated test suites?

Answer: When selecting automation tools, I consider factors such as the project's technology stack, team expertise, and the scope of automation. I prioritize tools that support scalability, allowing the automated test suites to adapt to changes in the application and accommodate future enhancements. The design of modular and maintainable frameworks contributes to scalability.

Cross-Functional Collaboration:

**Question:** **Can you share an example from your experience where you successfully collaborated with cross-functional Agile teams, including developers and business users, to deliver high-quality software**?

**Answer:** In my previous role, I was an integral part of an Agile team where we collaborated closely on a complex project. We held regular sprint ceremonies, participated in daily stand-ups, and engaged in continuous communication to align testing efforts with development goals.

Understanding of Agile Quality Assurance:

**Question:** **How do you demonstrate a thorough understanding of the software development process and all aspects of Agile Quality Assurance?**

**Answer**: I ensure Agile principles are ingrained in our testing processes, adapting to changes, and focusing on delivering value in each sprint. I actively participate in Agile ceremonies, incorporating feedback loops to enhance the overall quality assurance strategy.

Effective & Comprehensive Testing:

**Question:** **Discuss your approach to delivering effective and comprehensive testing within the timeframe of Agile sprints. How do you balance thorough testing with tight sprint timelines?**

**Answer:** I prioritize testing activities based on user stories and features, leveraging automation frameworks where applicable. Regular communication with the team allows us to identify critical areas for testing, ensuring the highest quality software is delivered within sprint timelines.

Automation Frameworks and Tools:

**Question:** **Can you elaborate on your experience with automation frameworks and tools? How have you routinely edited and executed automated test scripts to enhance testing efficiency?**

**Answer:** I have hands-on experience with automation frameworks like Selenium and tools such as JUnit. Regularly updating and executing automated scripts not only accelerates regression testing but also contributes to the overall efficiency of our testing processes.

Defect Tracking and Root Cause Analysis:

**Question:** **How do you manage defect tracking, and what steps do you take to perform root cause analysis? Can you provide an example where your defect analysis led to process improvement?**

**Answer:** I use robust defect tracking tools like Jira, ensuring timely identification and resolution of issues. Root cause analysis involves collaboration with the team to pinpoint underlying issues. In a previous project, this analysis led to a streamlined defect resolution process.

Risk-Based Testing Approach:

**Question**: Explain how you apply a Risk-Based Testing approach to identify the appropriate level of testing for different initiatives. Can you share an instance where this approach proved effective?

**Answer**: I assess project risks to prioritize testing efforts. In one project, by focusing on high-risk areas first, we were able to address critical issues early in the development cycle, minimizing potential risks and ensuring a higher quality outcome.

Proactive Collaboration:

**Question**: **How do you proactively work with different roles on the project team to provide support and ensure overall quality and success of initiatives?**

**Answer:** I maintain open communication channels with developers, business analysts, and other team members. This proactive collaboration involves addressing issues promptly, sharing insights during sprint planning, and fostering a collaborative environment that values quality.

Continuous Learning and Expertise:

**Question:** **How do you independently and proactively maintain expertise in a broad range of testing methodologies and strategies?**

**Answer:** I stay informed about industry trends, participate in relevant training, and actively engage in communities. This continuous learning approach ensures that my skills are up to date, enabling me to contribute effectively to diverse testing projects.

Designing Clear Tests:

**Question:** **Can you explain how you design tests that clearly convey business rules and code, ensuring both business users and developers understand their intentions?**

**Answer**: I focus on creating clear and concise test cases aligned with user stories. Using a behavior-driven development (BDD) approach, I ensure that tests are easily understandable by both technical and non-technical stakeholders, facilitating collaboration.

UAT Collaboration with Business Partners:

**Question**: **How do you work with Business partners during User Acceptance Testing (UAT)? Can you share an example of a project where your collaboration with business users was crucial to project success?**

**Answer:** I actively collaborate with Business partners during UAT by involving them in the testing process. In a past project, this collaboration ensured that UAT scenarios accurately reflected real-world usage, leading to a successful project outcome.

Certainly! Here are the top 10 interview questions along with possible answers based on the provided job description:

API Testing Tools:

**Question: Can you name and briefly describe some API testing tools you are familiar with, and share an example of when you used them in a previous role?**

**Answer:** I have experience with tools like Postman, Rest Assured, and SoapUI. In my previous role, I extensively used Postman to create and execute API tests, allowing for efficient testing and validation of endpoints.

Multisystem Experience:

**Question: How have you managed testing across multiple systems in your previous experience? Can you provide an example of a complex testing scenario involving integration with multiple systems?**

**Answer**: In my previous role, I worked on a project that required integration with external payment gateways and internal databases. I designed and executed tests that ensured seamless data flow and accurate transactions across the integrated systems.

Experience with Selenium:

**Question: Discuss your experience with Selenium. What types of testing scenarios have you automated using Selenium, and how do you ensure robust test scripts?**

**Answer:** I have extensive experience with Selenium for web automation, covering functional testing scenarios, regression testing, and cross-browser testing. I ensure robustness by following best practices, like using unique identifiers and implementing explicit waits for dynamic elements.

Basic Automation Knowledge:

**Question: When we mention "basic automation knowledge," what specific tools or frameworks come to your mind, and how comfortable are you with implementing basic automation scripts?**

**Answer:** Basic automation knowledge often includes tools like WebDriver for browser automation and JUnit/TestNG for test execution. I am comfortable creating and executing basic scripts using these tools to automate repetitive and time-consuming manual tests.

Cross-Functional Collaboration:

**Question: How do you collaborate with developers, product managers, and other cross-functional team members to understand testing requirements and ensure comprehensive test coverage?**

**Answer:** I actively participate in sprint planning meetings to understand upcoming features. Regular communication with developers and product managers ensures alignment on testing priorities, and I collaborate to create test cases that cover all relevant scenarios.

Testing Strategies for API:

**Question: Describe your testing strategies for API testing. How do you ensure API functionalities are thoroughly tested for accuracy and reliability?**

**Answer:** My API testing strategy involves creating test cases for different HTTP methods, handling various response codes, and validating data integrity. I also focus on edge cases, error scenarios, and performance aspects to ensure comprehensive coverage.

Challenges in Multisystem Testing:

**Question: In your experience, what challenges have you faced when testing across multiple systems, and how did you overcome them?**

**Answer:** Challenges often include data consistency, communication between systems, and dependency management. To overcome these, I establish clear communication channels, use mock services, and carefully plan test data to simulate real-world scenarios.

Integration Testing Approach:

**Question: Can you outline your approach to integration testing, especially when dealing with diverse systems? How do you ensure seamless integration and data flow?**

**Answer:** My approach involves testing the interaction between different components to ensure they work together as expected. I create test scenarios covering various integration points and validate the flow of data to identify and address any issues.

Selenium Best Practices:

**Question: What are some best practices you follow when using Selenium for test automation? How do you handle dynamic elements and ensure the maintainability of your test scripts?**

**Answer:** I follow best practices such as using unique identifiers, implementing proper waits for dynamic elements, and structuring my tests using the Page Object Model. This ensures maintainability and reduces the likelihood of test failures.

Adaptability to Automation Frameworks:

**Question: How adaptable are you to different automation frameworks? Can you share an example of a project where you had to quickly adapt to a new automation framework, and how did you approach the learning curve?**

**Answer:** I am adaptable to various automation frameworks, having previously transitioned from a keyword-driven framework to a behavior-driven framework in a short timeframe. I approach learning curves by leveraging documentation, online resources, and collaborating with experienced team members to quickly become proficient in new frameworks.

**TOP 10**

Certainly! Here are the top 10 interview questions and answers for a QA Automation Engineer based on the provided job description:

Understanding of Agile QA:

**Question:** **How do you demonstrate a thorough understanding of the software development process and all aspects of Agile Quality Assurance?**

**Answer**: I actively participate in cross-functional Agile teams, collaborating with testers, developers, and business users throughout the development lifecycle. I ensure effective and comprehensive testing within Agile sprints, aligning with Agile principles.

Automation Frameworks and Tools:

**Question: Can you elaborate on your experience with automation frameworks and tools? How do you routinely edit and execute automated test scripts using tools like Cucumber, FitNesse, TestNG, or Selenium?**

**Answer**: I have hands-on experience with automation frameworks like **Cucumber** and **Selenium.** I regularly edit and execute test scripts to achieve efficient and reliable automated testing, contributing to the speed and accuracy of our testing processes.

Defect Management and Root Cause Analysis:

**Question: How do you manage defect tracking and perform root cause analysis? Can you provide an example of a situation where your defect analysis led to process improvement?**

**Answer:** I use tools like **Jira** for defect tracking and perform root cause analysis to identify the underlying issues. In a previous project, my defect analysis prompted process improvements, enhancing the overall quality of deliverables.

Analytical and Review Skills:

**Question: How do you analyze and review data, test results, and documents for consistency and accuracy?**

**Answer:** I meticulously review test results, ensuring they align with expected outcomes. I analyze data and documentation for consistency, accuracy, and completeness, maintaining a high standard of quality in our testing artifacts.

Collaboration and Support:

**Question: How do you assist QA team members in utilizing test resources effectively? Can you provide an example of how you support other QA team members in testing activities and processes?**

**Answer:** I actively collaborate with team members, ensuring resources are used efficiently. For instance, I've supported colleagues by sharing expertise on test automation techniques and troubleshooting challenges, fostering a collaborative testing environment.

Risk-Based Testing Approach:

**Question: How do you ensure the appropriate level of testing through a Risk-Based Testing approach? Can you share an example of how you applied this approach to identify critical test initiatives?**

**Answer:** I apply **a Risk-Based Testing approach** by assessing project risks and prioritizing testing efforts accordingly. For instance, in a time-constrained project, I focused on testing critical functionalities first, ensuring that the most impactful risks were addressed promptly.

Test Strategy and Documentation:

**Question: Can you elaborate on your experience in creating test strategy and test case documents? How do you design tests that convey business rules and code for both business users and developers to understand?**

**Answer:** **I design tests with clarity on business rules**, ensuring that test documentation is easily understandable by both business users and developers. My test strategies encompass comprehensive test plans, test cases, and acceptance testing, aligning with project objectives.

UAT Collaboration with Business Partners:

**Question: How do you collaborate with Business partners during User Acceptance Testing (UAT)? Can you share an example of a project where your collaboration with business users was crucial to project success?**

**Answer:** I actively collaborate with Business partners during UAT by creating UAT plans and involving them in the testing process. In a previous project, our collaboration ensured that UAT scenarios accurately reflected real-world usage, leading to a successful project outcome.

Continuous Learning and QA Standards:

**Question: How do you independently maintain expertise in various testing methodologies and strategies? Additionally, how have you contributed to the implementation and improvement of QA standards, processes, and procedures?**

**Answer:** I stay updated on industry trends, attend training sessions, and actively participate in relevant communities. In past roles, I contributed to the implementation and improvement of QA standards by sharing best practices and leading initiatives to enhance testing processes.

SQL Knowledge and Test Data Setup:

**Question: Explain your ability to write and understand basic SQL queries for data verification and test data setup. Can you share an example of how SQL queries were integral to your testing activities?**

**Answer:** I have a strong grasp of SQL, allowing me to write queries for data verification and set up test data. In a previous project, I used SQL queries to validate data integrity during regression testing, ensuring the reliability of our testing processes.

1. **How do you select Test cases for Automation Testing**

* Repetitive tests that run for multiple builds.
* Tests that tend to cause human error.
* Tests that require multiple data sets.
* Frequently used functionality that introduces high risk conditions.
* Tests that are impossible to perform manually.
* Tests that run on several different hardware or software platforms and configurations.
* Tests that take a lot of effort and time when manual testing.

Remember that not all test cases may be suitable for automation. It's essential to perform a cost-benefit analysis and continually evaluate and update your automation suite based on the evolving needs of the project.

1. **How do you select Test cases for Regression Testing**

1. Select test cases where there are recent code changes or functional changes

2. Select test cases that map to the business requirements

3. Select test cases in the areas with frequent bugs/defects

4. Select test cases for the areas which are visible to the user

5. Select all integration test cases

6. Select all complex test cases

7. Select test cases based on priorities

8. Select test cases based on criticality and impact of bug fixes

9. Select a sample of successful and failed test cases

1. **Selenium WebDriver architecture**

**Test Script:**

Code written in Java, Python, etc., to automate browser interactions.

**WebDriver API:**

Selenium's interface providing methods to control browsers and interact with web elements.

**Browser Drivers:**

Specific drivers (e.g., Chrome Driver, Gecko Driver) enabling communication between WebDriver and browsers.

**JSON Wire Protocol:**

Defines a RESTful communication protocol translating commands between WebDriver and browser drivers.

**Browser:**

Web browser where tests run, controlled by WebDriver through the browser driver.

**The flow of automation using Selenium WebDriver typically follows these steps:**

1.Your test script (written in a programming language) uses the Selenium **WebDriver API** to send commands.

2.These commands are translated into the **JSON Wire Protocol format.**

3.The **browser driver** (specific to the browser being used) receives the HTTP requests containing the commands.

4.The browser driver translates these requests into **actions** that the browser can understand and execute.

5.The browser performs the actions on the **actual web page**.

6.The results of these actions are sent back through the browser driver to your **test script.**

7.This architecture allows **Selenium WebDriver** to support multiple browsers and programming languages, making it a versatile tool for automating web applications. 8.It provides a standardized way to interact with browsers and ensures consistency across different browser-driver combinations.

**4.Here's a simplified explanation of testing Android and iOS using Appium:**

**1.Install Appium:**

Install Appium on your computer. It's a tool that helps automate testing for mobile apps.

**2. Setup Dependencies:**

Make sure you have the necessary software installed, like **Node.js**, **Java** (for Android), and **Xcode** (for iOS).

**3.Start Appium Server:**

Run the Appium server. It's like a traffic cop that helps communication between your **computer and the mobile devices**.

**4. Define Device Settings:**

Tell Appium about the mobile device you want to test (name, version, etc.). These are called "**desired capabilities."**

**5.Write Test Scripts:**

Use a programming language (like Java or Python) to write simple scripts that describe what actions the app should take during testing.

**6.Identify App Elements**:

Specify how to find and interact with elements in your app, like buttons or text fields.

**7.Run Your Scripts:**

Run your scripts. Appium translates your commands into actions on the mobile device, like tapping buttons or entering text.

**8.Check Results:**

See if your app behaves as expected. Appium provides logs and reports to help you understand what happened during the tests.

**9.Repeat for iOS:**

If you want to test on an iOS device, you follow a similar process, but you might **need a Mac and Xcode installed.**

**10.Make Testing Easier:**

Use tools like **Appium Inspector** to visually identify app elements, making it easier to write your scripts.

**11.Automate Everything (Optional):**

If you like, you can set up things, so your tests run automatically whenever you make changes to your app.

**In short, Appium helps you automate testing for both Android and iOS by allowing you to write scripts that simulate user interactions on mobile devices, helping ensure your app works as expected.**

Selenium WebDriver provides a range of actions to interact with web elements and automate browser tasks. Here's a list of common actions you can perform using Selenium WebDriver:

**Navigation:**

get (String url): Navigate to a specified URL.

navigate ().to (String url): **Navigate to a specified URL.**

navigate (). refresh (): **Refresh the current page.**

navigate (). back (): **Navigate back to the previous page.**

navigate ().forward (): **Navigate forward to the next page.**

**Interacting with Web Elements:**

find Element (By locator): Find a single web element based on the given locator strategy.

find Elements (By locator): Find multiple web elements based on the given locator strategy.

**send Keys** (Char Sequence... keysToSend): Simulate typing into an input field.

click (): Click on a web element.

clear (): Clear the content of an input field.

getAttribute (String attribute): Get the value of a specified attribute of the element.

getText (): Get the visible text of the element.

is Selected (): Check if the element is selected.

is Enabled (): Check if the element is enabled.

is Displayed(): Check if the element is displayed.

**Dropdowns:**

selectByVisibleText (String text): Select an option by its visible text from a dropdown.

selectByValue (String value): Select an option by its attribute value from a dropdown.

selectByIndex (int index): Select an option by its index from a dropdown.

deselect All(): Deselect all selected options in a multi-select dropdown.

**Checkboxes and Radio Buttons:**

click (): Click on a checkbox or radio button to toggle its state.

**Alerts and Pop-ups:**

switchTo().alert(): Switch to an alert.

accept(): Accept the alert.

dismiss (): Dismiss the alert.

sendKeys (String keysToSend): Send keys to an alert.

**Frames and Windows:**

switchTo ().frame(String frameId): Switch to a frame by its ID.

switchTo().defaultContent(): Switch back to the default content.

switchTo ().window(String windowHandle): Switch to a specific browser window.

**Mouse and Keyboard Actions:**

Actions actions = new Actions(driver): Initialize Actions class.

actions.moveToElement(WebElement target): Move the mouse to a specified element.

actions.click(), actions.doubleClick(), actions.contextClick(): Perform mouse actions.

actions.keyDown(Keys key), actions.keyUp(Keys key): Simulate keyboard key presses.

**Wait Commands:**

implicitlyWait(long time, TimeUnit unit): Set the implicit wait time.

Thread.sleep(long millis): Pause execution for a specified time.

Explicit waits using WebDriverWait for waiting conditions.

These are some of the common actions you can perform using Selenium WebDriver.

The actual usage of these actions depends on the specific requirements of your test.

scenarios.

**Appium in android or IOS**

Certainly! Here are simplified versions of some common actions in Appium for both Android and iOS:

**Launch App:**

Android: driver.findElementByAndroidUIAutomator("text('App Name')").click();

iOS: driver.findElementByAccessibilityId("App Name").click();

**Click on an Element:**

driver.findElement(By.id("elementId")).click();

**Type into a Text Field:**

driver.findElement(By.id("elementId")).sendKeys("Text to Type");

**Clear Text from an Element:**

driver.findElement(By.id("elementId")).clear();

**Get Text from an Element:**

String text = driver.findElement(By.id("elementId")).getText();

**Assert Element Presence:**

assertTrue(driver.findElement(By.id("elementId")).isDisplayed());

**Dropdown** - Select Option by Value:

new Select(driver.findElement(By.id("dropdownId"))).selectByValue("optionValue");

Switch to WebView Context (Hybrid Apps):

driver.context("WEBVIEW\_packageName");

Switch Back to Native Context:

driver.context("NATIVE\_APP");

Swipe Left:

TouchAction action = new TouchAction(driver);

action.press(startX, startY).moveTo(endX, endY).release().perform();

Scroll to an Element:

Android: driver.findElementByAndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).scrollIntoView(text(\"ElementText\"));");

iOS: driver.findElementByIosNsPredicate("name == 'ElementText'").click();

Take Screenshot:

**File screenshot** = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

Hide Keyboard:

driver.hideKeyboard();

Execute JavaScript:

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].setAttribute('value', 'new value')", element);

Perform Tap:

TouchAction action = new TouchAction(driver);

action.tap(x, y).perform();

Remember to adapt these based on your specific app and testing needs. Mobile automation may involve additional considerations, so refer to the Appium documentation for more details.

Playwright:

* Playwright is a framework for Web Testing and Automation. It allows testing Chromium, Firefox and WebKit with a single API. Playwright is built to enable cross-browser web automation that is ever-green, capable, reliable, and fast. Headless execution is supported for all browsers on all platforms.
* The primary goal of Playwright is improving automated UI testing.
* Playwright comes with e integration. For example, the Playwright has Docker images, allowing you to run tests quickly in an isolated and controlled environment.
* They also support your existing JavaScript test runners, like Jest/Jasmine, AVA, and Mocha, which is helpful if you are porting from an existing code base.
* The framework supports cross-browser development on Chromium, Web Kit, and Firefox – including Chrome, Edge, Firefox, Opera, and Safari.
* Cross-platform execution is supported on Windows, Linux, and macOS.
* Testing cross-language, including JavaScript, TypeScript, Python, Java, and .NET – choose the environment that suits you while still covering all areas and formats.
* Auto-wait, smart assertions that retry until an element is found, and test data tracing – keep track of logs and videos easily.
* Built with modern architecture and no restrictions, the application allows you to interact with multi-page, multi-tab websites like a real user and easily handles frames and browser events.
* As the Playwright framework is aligned with the modern browser's architecture, it doesn't have the in-process test runner limitations.
* Playwright delivers full test isolation with no overhead. It creates a browser context for each test which only takes a handful of milliseconds.

Cypress:

* I would choose Cypress for its simplicity and fast execution, especially for non-Angular applications.
* Data-driven testing in Cypress involves iterating through test data sets.
* It's beneficial for testing different scenarios without duplicating test code. Data-driven tests enhance test coverage and make scripts more maintainable.
* In my previous role, I extensively used Cypress for end-to-end testing due to its ease of use and robust capabilities. I also have experience with Protractor for Angular applications, Zest for API testing, Enzyme for React component testing, and Mocha for JavaScript unit testing.

**Roles and Responsibilities: QA Automation Engineer - Banking Project**

Project Overview:

I played a key role in the development and quality assurance of a robust mobile banking application, leveraging Xcode and Swift to ensure the delivery of a secure and seamless financial experience.

\*\*1. **iOS Automation Development with Xcode and Swift:**

Designed and implemented end-to-end automated test scripts using XCode’s testing framework and Swift scripting.

Ensured comprehensive coverage of critical banking functionalities, including account management, fund transfers, and transaction history.

\*\*2. **Security Testing:**

Implemented Swift scripts for security testing, addressing vulnerabilities and ensuring compliance with stringent banking security standards.

Collaborated closely with security experts to conduct regular security assessments and address emerging threats.

\*\*3. **Transaction and Data Integrity Testing:**

Conducted rigorous testing to guarantee the accuracy and integrity of financial transactions and sensitive user data.

Utilized Swift-based test scenarios to simulate various transaction scenarios, ensuring the reliability of financial processes.

\*\*4. **Continuous Integration and Deployment (CI/CD):**

Integrated automated tests into the CI/CD pipeline, ensuring swift and reliable validation of code changes.

Implemented automated deployment verification to ensure consistency between testing and production environments.

\*\*5. **Collaboration with Cross-Functional Teams:**

Worked closely with developers, product managers, and business analysts to understand banking requirements and translate them into effective test cases.

Facilitated communication between QA and development teams to address issues promptly and ensure the timely delivery of high-quality features.

\*\*6. **Regression Testing and Release Validation:**

Conducted thorough regression testing to identify and address potential issues in new releases.

Validated each release to guarantee the stability and functionality of the mobile banking app, collaborating with development teams to resolve any identified issues.

Achievements:

Successfully automated critical banking functionalities, resulting in a 30% reduction in testing cycles.

Contributed to a seamless and secure user experience, receiving positive feedback from users and stakeholders.

Maintained a high level of quality assurance, ensuring the mobile banking app consistently met regulatory requirements.

What Drives Me:

I am motivated by the opportunity to contribute my Xcode and Swift expertise to deliver secure, reliable, and user-friendly mobile banking applications. Let's connect and discuss how my experience can bring value to your team or project.

**Test lead roles responsibilities in the project**

* Create, review, and maintain the Test Plan outlining the scope, strategy, resources, schedule, and deliverables for testing.
* Allocate tasks among the testing team members based on their skill sets and strengths.
* Mentor and provide guidance to the testing team.
* Conduct regular team meetings to discuss progress, risks, and challenges.
* Oversee the creation and execution of detailed test cases, scripts, and scenarios.
* Oversee the **defect tracking process**.
* Prioritize defects in coordination with developers and **product managers.**
* Regularly update stakeholders, including developers, product managers, and senior management, about the testing progress, risks, and results.
* Participate in meetings with cross-functional teams to provide insights from the testing perspective.

Create end-of-cycle reports summarizing testing efforts, results, and metrics.

**automation roles in the project**

* Write, design, and execute automated tests by creating scripts that run testing functions automatically.
* Maximize test coverage for the most critical features of the system.
* Determine the priority for test scenarios and create execution plans to implement these Scenarios.
* Write documentation for automated processes including test plans, test procedures, and test Cases.
* Build test automation framework and maintain the automation framework.
* Log and document bugs in the tracking system.
* Grouping Test cases and generating detailed Test reports using **TestNG.**
* Analyzing Test Results and Reporting Defects.
* Besides writing automated tests, we do exploratory testing when needed.

**Banking Project**

• database testing,

• functional testing,

• security

• user acceptance testing.

• In the banking application domain, you should check the login, registration, security, performance, transactions, and general banking services features.

**Challenges faced during Banking Project**

QA Automation in a banking project comes with its set of challenges, often due to the complexity, security requirements, and regulatory standards associated with financial applications. Here are some common challenges faced during a banking project in QA Automation:

**Security Compliance:**

Challenge: Ensuring compliance with strict security standards and protocols poses a significant challenge. Security testing must cover encryption, authentication, authorization, and protection against vulnerabilities.

Approach: Implement comprehensive security testing using tools and frameworks to identify and address potential vulnerabilities. Collaborate closely with security experts to stay updated on emerging threats.

Complex Business Rules:

Challenge: Banking applications often involve intricate business rules related to transactions, interest calculations, and compliance. Validating these rules thoroughly in an automated manner can be challenging.

Approach: Develop detailed test scenarios that cover various business rule combinations. Leverage automation to execute and validate these scenarios, ensuring adherence to business requirements.

Regulatory Compliance:

Challenge: Adhering to regulatory standards and compliance requirements is critical in the banking sector. Ensuring that the application complies with laws and regulations adds complexity to testing.

Approach: Stay updated on regulatory changes and incorporate them into test scenarios. Collaborate with regulatory experts and conduct regular audits to ensure compliance.

Integration Testing:

Challenge: Banking systems often comprise multiple interconnected modules and external services. Testing the integration points thoroughly to ensure seamless communication is a complex task.

Approach: Develop robust test cases for integration points, conduct end-to-end testing, and simulate real-world scenarios. Leverage tools for API testing and ensure interoperability between different components.

Data Privacy and Confidentiality:

Challenge: Protecting sensitive customer data and ensuring data privacy are paramount in banking applications. Handling and masking confidential information during testing can be challenging.

Approach: Implement data anonymization and masking techniques during testing

## Steps to Improve the Banking Application Test Results

• Start by thoroughly understanding the project requirements by collaborating with the stakeholders and project managers.

• Develop a comprehensive test plan that outlines test objectives, scope, resources, schedules, and risk assessment.

• Establish a robust strategy for managing test data. Use realistic, anonymized, and diverse data sets that mimic real-world scenarios to validate various banking transactions effectively.

• Implement test automation to increase test coverage and efficiency. Automate repetitive test cases for banking application functionalities, including regression testing, to reduce manual effort and accelerate testing cycles.

• Prioritize security testing to identify vulnerabilities and protect sensitive customer data. Conduct penetration testing, vulnerability assessments, and code reviews to fortify the application’s security.

• Continuously analyze test results and incorporate feedback to refine test cases for banking applications.

## Test Phases in Testing Banking Applications

* **Requirement Analysis**: It is done by business analyst; requirements for a particular banking application are gathered and documented.
* **Requirement Review**: Quality analysts, business analysts, and development leads are involved in this task. The requirement gathering document is reviewed at this stage, and cross-checked to ensure that it does not affect the workflow.
* **Business Requirements Documentation**: Business requirements documents are prepared by quality analysts in which all reviewed business requirements are covered.
* **Database Testing**: It is the most important part of bank application testing. This testing is done to ensure data integrity, data loading, data migration, stored procedures, and functions validation, rules testing, etc.
* **Integration Testing**: Under Integration all components that are developed are integrated and validated.
* **Functional Testing**: The usual software testing activities like Test Case preparation, test case review and test case execution is done during this phase.
* **Security Testing**: It ensures that the software does not have any security flaws. During test preparation, QA team needs to include both negative as well as positive test scenarios to break into the system and report it before any unauthorized individual access it. While to prevent from hacking, the bank should also implement a multi-layer of access validation like a one-time password.
* **Usability Testing**: It ensures that differently able people should be able to use the system as normal user. For example, ATM with hearing and Braille facility for disabled.
* **User Acceptance Testing**: It is the final stage of testing done by the end users to ensure the compliance of the application with the real-world scenario.

INTRO

With over 10 years of dedicated experience as a Software QA Automation Engineer, I bring expertise across a spectrum of testing methodologies and domains.

**Banking:**

**Security Testing:**

Ensure the robustness of security measures in online banking platforms to protect sensitive financial information.

**Performance Testing:**

Evaluate the responsiveness and scalability of banking applications, especially during peak transaction periods.

**Compliance Testing**:

Validate that the banking software adheres to industry regulations and compliance standards such as PCI-DSS.

E-commerce:

**Usability Testing:**

Assess the user-friendliness and overall user experience of the e-commerce platform, including navigation and checkout processes.

**Load Testing:**

Evaluate how the e-commerce platform handles varying levels of user traffic to prevent downtime during high-demand periods.

**Cross-Browser Testing**:

Ensure consistent functionality and appearance across different web browsers to provide a seamless shopping experience.

Insurance:

**Integration Testing:**

Validate the seamless integration of insurance applications with external systems such as payment gateways and regulatory databases.

**Data Accuracy Testing:**

Verify the accuracy of data processed by insurance applications, especially critical for policy information and claims processing.

**Regulatory Compliance Testing:** Ensure that the insurance software complies with industry-specific regulations and legal requirements.

Healthcare:

HIPAA Compliance Testing:

Validate that healthcare applications adhere to the Health Insurance Portability and Accountability Act (HIPAA) regulations for patient data protection.

Interoperability Testing:

Confirm the seamless exchange of healthcare data between different systems and devices to enhance patient care coordination.

Clinical Validation Testing:

Ensure that healthcare software accurately reflects clinical procedures and guidelines for safe and effective patient care.

These testing types are tailored to the specific needs and challenges of each domain, contributing to the overall quality and reliability of software applications in Banking, E-commerce, Insurance, and Healthcare.

**automation roles in the project**

* Write, design, and execute automated tests by creating scripts that run testing functions automatically.
* Maximize test coverage for the most critical features of the system.
* Determine the priority for test scenarios and create execution plans to implement these Scenarios.
* Write documentation for automated processes including test plans, test procedures, and test Cases.
* Build test automation framework and maintain the automation framework.
* Log and document bugs in the tracking system.
* Grouping Test cases and generating detailed Test reports using TestNG.
* Analyzing Test Results and Reporting Defects.
* Besides writing automated tests, we do exploratory testing when needed.

**Test lead roles responsibilities in the project**

* Create, review, and maintain the Test Plan outlining the scope, strategy, resources, schedule, and deliverables for testing.
* Allocate tasks among the testing team members based on their skill sets and strengths.
* Mentor and provide guidance to the testing team.
* Conduct regular team meetings to discuss progress, risks, and challenges.
* Oversee the creation and execution of detailed test cases, scripts, and scenarios.
* Oversee the defect tracking process.
* Prioritize defects in coordination with developers and product managers.
* Regularly update stakeholders, including developers, product managers, and senior management, about the testing progress, risks, and results.
* Participate in meetings with cross-functional teams to provide insights from the testing perspective.
* Review the test artifacts, including test cases, scripts, and strategies, to ensure they meet quality standards.
* Create end-of-cycle reports summarizing testing efforts, results, and metrics.

**Test case best practices:**

We use Zephyr for test case management, and I had good practices in writing test cases.

* Creating test cases simple and Transparent.
* Create test case with End User in Mind.
* Avoid writing test case Repetition.
* Do not assume, Ensure 100% coverage. (Requirement Traceability Matrix)
* Creating Unique ID for each test case (Easy tracking)
* Implementing testing techniques – Boundary Value analysis, Equivalence Partition
* And peer Reviewing the test cases.

**How does a typical day looks like for QA engineer ?**

* First thing in the morning is to review emails for any critical issues reported overnight or messages from offshore teams.
* update jira board to reflect the current status.
* participate in daily stand-up meetings (if following Agile) to discuss what was done the previous day, what's planned for today, and any blockers.
* Look for any issues in automation builds especially if there were code integrations or merges the previous day.
* Test new features, changes, or bug fixes manually based on test cases.
* If any issues are found during testing, log them in the defect tracking system with detailed steps, expected vs. actual results, and any relevant screenshots or logs.
* Check resolved issues to confirm they're fixed and close them in the tracking system.
* send a daily status report highlighting progress, issues found, and other critical metrics.
* Make a to-do list for tomorrow.

**Sprint Planning**

**I** am part of a 14-member team that includes Project Manager, Business Analyst, Product Owner, developers, and testers. Our sprint cycle is 2 weeks long and we follow the agile methodology.

We use JIRA for Task Management tool. In the sprint planning meeting, I reviewed the user stories and acceptance criteria with the BA and the rest of the team. We estimate the size of each task and assign them to the appropriate team members. I write test cases based on the acceptance criteria and the expected behavior of the software. During the sprint, I collaborate with the developers and other testers through GitHub, where we push and review our code. I also attend daily stand-up meetings to report my progress and challenges. I report and track defects found during testing using Jira platform. I participate in defect triage meetings **(Defect review board)** three times a week to prioritize and assign defects to developers.

Before the demo session, I perform regression testing to ensure that all the features are working as expected. I use Jenkins to run my automated test cases and generate reports. I attended the demo session where we presented our work to the PO in the UAT environment. I provide feedback and suggestions for improvement.

After the demo session, I attended the **retrospective meeting** where we discussed what went well and what can be improved in the next sprint. I also prepare for the production release by ensuring that all the defects are resolved, and all the test cases are passed.

**CI/CD Pipeline:**

* When Developers push the code to the **GITHUB** Repository. We use **Jenkins** as a Maven Build tool which will build the jar files.
* Jenkins polls with the GITHUB and invokes for any new code changes. where it does Smoke testing, Unit testing, Integration testing, as a tester I do regression testing.
* Jenkins will automate and execute test cases.
* We deploy code to the QA and UAT environment.

**Loan IQ**

Loan IQ is a popular loan origination and servicing platform used in the financial industry. In the context of Quality Assurance (QA), here are key aspects related to Loan IQ:

1. **Understanding Loan IQ Functionality:**

QA professionals working with Loan IQ should have a comprehensive understanding of the platform's functionalities, including loan origination, servicing, and related financial processes.

2. **Test Case Design for Loan Scenarios:**

QA engineers need to design test cases that cover various loan scenarios, ensuring that Loan IQ handles different types of loans, interest rates, payment schedules, and other financial parameters accurately.

3. **Integration Testing:**

Perform integration testing to validate the seamless integration of Loan IQ with other systems and applications within the financial ecosystem. This includes testing interfaces with core banking systems, CRMs, and other financial tools.

4. **Regression Testing:**

Implement rigorous regression testing to ensure that new updates or changes to Loan IQ do not adversely impact existing functionalities. This is critical to maintaining the stability and reliability of the loan processing system.

5. **Data Management:**

Develop strategies for effective data management in Loan IQ testing, considering various data scenarios, data migration, and the accuracy of financial data processing.

6. Performance Testing:

Conduct performance testing to assess the scalability and responsiveness of Loan IQ under different load conditions. This is crucial for ensuring that the platform can handle a high volume of transactions efficiently.

7. Security Testing:

Implement security testing measures to evaluate the robustness of Loan IQ in safeguarding sensitive financial data. This involves testing access controls, encryption, and compliance with financial industry security standards.

8. Automation of Loan Processes:

Leverage automation tools and frameworks to streamline the testing of Loan IQ processes. Automation can enhance testing efficiency, especially for repetitive and complex scenarios.

9. User Acceptance Testing (UAT):

Collaborate with business users to conduct User Acceptance Testing, ensuring that Loan IQ meets the business requirements and user expectations. UAT is crucial for validating that the system aligns with real-world usage.

10. Documentation and Reporting:

Maintain comprehensive documentation of test cases, test plans, and test results for Loan IQ. Provide detailed reports to stakeholders, highlighting testing outcomes, identified issues, and recommendations for improvements.

11. Continuous Learning and Updates:

Stay informed about updates and new features in Loan IQ. Continuous learning is essential for QA professionals to adapt their testing strategies to evolving functionalities and industry standards.

12. Collaboration with Development Teams:

Work closely with development teams to understand system changes, updates, and enhancements. Collaboration ensures that testing efforts align with the development lifecycle and business goals.

QA professionals in the financial industry, particularly those dealing with Loan IQ, play a crucial role in ensuring the accuracy, reliability, and security of loan origination and servicing processes. Their testing efforts contribute to the overall efficiency of financial institutions in managing loan portfolios and delivering high-quality services to clients.

**Dynatrace:**

* Dynatrace helps identify performance issues by capturing detailed transaction traces and analyzing application performance.

**Datadog:**

* Data Dog is a monitoring tool that provides real-time insights into application and system performance. In performance testing, Data Dog helps track key metrics, analyze trends, and identify bottlenecks by monitoring server resources, application response times, and overall system health**.**
* Data Dog provides real-time visibility into system and application metrics, enabling performance testers to pinpoint issues such as high resource utilization, slow response times, or increased error rates. This information aids in troubleshooting and optimizing system performance**.**
* **in a recent project, Data Dog highlighted a spike in database response times during load testing. By analyzing Data Dog metrics, we identified inefficient queries, optimized indexes, and improved database configurations, resulting in enhanced overall application performance.**

**Load Runner:**

* Load Runner is a **performance testing** tool that simulates user traffic on applications, measuring system response and identifying **performance** bottlenecks. It includes features like Virtual User Generator (VuGen), Controller, and Analysis to script, control, and analyze performance tests.
* In a previous project, **Load Runner** identified a bottleneck in database response times during peak load. We optimized the SQL queries, implemented caching strategies, and fine-tuned database configurations. The result was a significant improvement in overall application performance.
* **Load Runner's** Analysis tool is used to interpret results. Metrics like **response times, throughput**, and error rates are analyzed. Performance issues can be identified by examining transaction breakdowns, identifying high-resource utilization, and correlating these findings with system metrics.
* Error handling in **Load Runner** involves defining rules to identify errors and specifying how to handle them. This includes **logging**, **retrying**, or **aborting transactions** based on predefined criteria. Detailed reports generated by Load Runner help in analyzing errors and their impact on performance.

**FAQ in Interviews**

### 1. **Types of Classes in Java**

1. Final Class
2. Static Class
3. Abstract Class
4. Concrete Class
5. POJO Class
6. Singleton Class
7. Inner Class

## 2. **Java Packages & API**

**A package in Java is used to group related classes. Think of it as a folder in a file directory. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:**

* **Built-in Packages (packages from the Java API)**
* **User-defined Packages (create your own packages)**

**QA**

**3.Day-to-day activities in QA (Quality Assurance) typically involve a variety of tasks aimed at ensuring the quality and reliability of software products. Here are some common activities:**

* **Test Planning: Reviewing project requirements, user stories, and acceptance criteria to develop test plans and strategies for upcoming features or releases.**
* **Test Case Design: Creating detailed test cases and test scenarios based on requirements, ensuring comprehensive coverage of functional and non-functional aspects of the software.**
* **Test Execution: Running test cases manually or using automated testing tools to verify the functionality of the software, identify defects, and ensure adherence to quality standards.**
* **Defect Management: Documenting, prioritizing, and tracking defects found during testing using defect tracking tools. Collaborating with development teams to investigate, resolve, and retest defects.**
* **Regression Testing: Conducting regression testing to ensure that new features or code changes have not introduced any unintended side effects or regressions in existing functionality.**
* **Automated Testing: Developing, maintaining, and executing automated test scripts using various automation tools and frameworks to increase testing efficiency and coverage.**
* **Performance Testing: Conducting performance testing to evaluate the speed, scalability, and stability of the software under different load conditions. Analyzing performance metrics and identifying areas for optimization.**
* **Continuous Integration/Continuous Deployment (CI/CD) Integration: Integrating testing processes into CI/CD pipelines to automate testing and ensure rapid and reliable delivery of software updates.**
* **Collaboration: Communicating and collaborating with cross-functional teams, including developers, product owners, and business stakeholders, to clarify requirements, address issues, and ensure alignment on quality goals.**
* **Documentation: Creating and maintaining test documentation, including test plans, test cases, test reports, and other relevant artifacts to support traceability and compliance requirements.**
* **Quality Metrics and Reporting: Tracking and analyzing quality metrics, such as defect density, test coverage, and pass/fail rates, to assess the effectiveness of testing efforts and identify areas for improvement.**
* **Continuous Learning: Staying updated on industry best practices, emerging technologies, and new testing tools and techniques through training, conferences, and self-study to enhance skills and expertise in QA.**

**List only the Properties in SoapUI**

**Global Properties:** Variables defined at the project level and accessible by any test case or request within the project.

**Test Case Properties**: Variables specific to a particular test case and accessible only within that test case.

**Property Expansion**: Syntax used to reference properties within various fields and settings, such as ${#TestCase#Property} for test case properties and ${#Project#Property} for global properties.

**Property Transfer:** Feature allowing data transfer between requests by extracting data from one response and storing it in a property for use in subsequent requests.

**Dynamic Values:** Properties used to dynamically generate values within test cases or requests, such as unique identifiers, timestamps, or random values.

**Environment-Specific Configuration:** Properties used to define environment-specific settings, such as endpoints, credentials, or timeouts, facilitating easy switching between different environments.

**DataDog**

Datadog is a cloud-based monitoring and analytics platform that helps organizations gain visibility into the performance of their applications, infrastructure, and cloud services. Here's some key information about Datadog:

Features: Datadog offers a wide range of features including:

Real-time metrics: Collect and visualize metrics from servers, applications, databases, and other sources in real-time.

Distributed tracing: Trace requests across distributed systems to identify performance bottlenecks and troubleshoot issues.

Log management: Collect, search, and analyze logs from applications and infrastructure to gain insights and troubleshoot problems.

APM (Application Performance Monitoring): Monitor the performance of applications and services, including response times, error rates, and throughput.

Infrastructure monitoring: Track the health and performance of servers, containers, databases, and cloud services.

Synthetic monitoring: Monitor the availability and performance of web applications and APIs from multiple locations around the world.

Security monitoring: Detect and alert on security threats and vulnerabilities in real-time.

Collaboration and alerting: Collaborate with teams and set up alerts based on predefined thresholds or custom conditions.

Integration: Datadog supports integration with a wide range of technologies and platforms including cloud providers (AWS, Azure, GCP), container orchestration platforms (Kubernetes, Docker), databases (MySQL, PostgreSQL, MongoDB), messaging systems (Kafka, RabbitMQ), and more.

Ease of Use: Datadog offers a user-friendly interface with customizable dashboards, visualizations, and alerting capabilities. It provides out-of-the-box integrations and easy-to-use APIs for custom integrations and automation.

Scalability: Datadog is designed to scale with your organization's needs, supporting millions of metrics per second and petabytes of data storage.

Cost: Datadog offers flexible pricing plans based on the volume of data ingested and the features required. It provides free trials and transparent pricing to help organizations choose the right plan for their needs.

Overall, Datadog is a powerful platform for monitoring and optimizing the performance of modern applications and infrastructure, providing comprehensive visibility and actionable insights to help organizations deliver better digital experiences to their users.

4. To execute Cucumber tests, follow these steps:

Write Feature Files: Create feature files using Gherkin syntax to define test scenarios in a human-readable format.

Implement Step Definitions: Write step definitions in your programming language (e.g., Java, Ruby) to map each step in the feature files to automation code.

Run Tests: Use the Cucumber command-line interface (CLI) or a build automation tool (e.g., Maven, Gradle) to run the tests. Execute the command with the appropriate parameters to specify the location of feature files and step definition files.

View Results: After execution, Cucumber generates test reports in various formats (e.g., HTML, JSON) displaying the test results, including passed, failed, or pending scenarios.

**cucumber <path\_to\_feature\_files>**