QA Interview Preparation Guide

PART 1: OPENING QUESTIONS

"Tell Me About Yourself"

"Thank you. I'm a dedicated Software Development student at SAIT, graduating in less than two months, with a strong background in both Electronics Engineering and professional QA experience.

My career began in technical support, where I developed sharp troubleshooting skills, which I then successfully transitioned into a Software QA Engineer role at Convoso, a cloud-based SaaS company. There, I've been responsible for manual and regression testing, creating test plans, and collaborating with developers to ensure high-quality software releases.

I'm now looking to bring my unique blend of SaaS domain knowledge, proactive QA skills, and growing automation expertise to a permanent role where I can contribute to a team's success long-term."

"Can you walk us through your resume and explain why you're interested in this QA & Manual Tester role?"

"Certainly. My career path has been a focused journey from technical support into software quality assurance. I started in Level 2 technical support roles at companies like Emerson and Ascent Solutions, where I was on the front lines troubleshooting complex technical issues for customers. This is where I developed a deep sense of customer advocacy and a keen eye for identifying root causes.

This naturally led me to my role as a Software QA Engineer at Convoso, where I transitioned from finding bugs after a release to preventing them before they reach the customer. I have hands-on experience with the full QA lifecycle—collaborating with developers on requirements, creating test plans and cases in tools like TestRail, and executing various testing methodologies like regression, smoke, and sanity testing.

I'm interested in this role because it perfectly aligns with my experience in SaaS applications and my passion for ensuring software reliability. Your focus on manual and regression testing directly matches my work at Convoso, which is also a SaaS platform. Furthermore, I'm excited by the opportunity to eventually contribute to automation and DevOps initiatives, as I have foundational skills in Python, Selenium, and Cypress that I'm eager to build upon."

PART 2: TECHNICAL EXPERIENCE QUESTIONS

"The job requires comprehensive manual and regression testing. Can you describe your experience in this area?"

"Absolutely. At Convoso, manual and regression testing were core to my daily responsibilities. For every release, I was responsible for executing a full regression suite to ensure that new features didn't break existing functionality. This involved testing across various workflows and user scenarios.

I also created and maintained detailed test plans and test cases based on product requirements and user stories. For example, when a new feature like a specific workflow automation rule was developed, I designed test cases to validate its behavior under different conditions, ensuring it integrated seamlessly with the existing platform. This rigorous process was critical for maintaining the stability of a customer-facing SaaS platform."

"Can you tell us about your experience creating test plans and test cases, and perhaps a process you implemented?"

"Absolutely. When I was promoted to QA Engineer at Convoso, I identified a significant gap: the automation team was struggling because there were no detailed, structured test cases to base their scripts on. The existing documentation was vague or non-existent.

I took the initiative to lead a project to fix this. I created and implemented a standardized process for test case creation:

1. **Template Standardization**: I first created a unified test case template in TestRail that included clear sections for Pre-conditions, Step-by-Step Actions, Expected Results, and Actual Results. This brought immediate consistency.
2. **Requirement Mapping**: I mandated that every user story in Jira must have at least one linked test case in TestRail. This created a clear, auditable traceability matrix from requirement to validation.
3. **Peer Review & Sign-off**: I introduced a two-step review process. First, a peer QA would review the test cases for clarity and coverage. Then, the developer assigned to the feature would review them to ensure the test accurately reflected the intended functionality before signing off.

For example, for a new feature, I didn't just write vague test descriptions. I wrote specific cases like: 'Verify the button is enabled only when valid data is entered,' and 'Verify a successful action triggers the expected system response and logs the event properly.'

This process was a success. My boss was very happy because it drastically reduced ambiguity, accelerated the automation team's script-writing speed, and improved the overall quality of our releases."

"This role requires someone who is organized, autonomous, and a strong communicator. Can you provide an example that demonstrates these qualities?"

"Certainly. At Convoso, I worked closely with the Product Team to review user stories and acceptance criteria before development began. This required me to be highly organized and proactive in seeking clarity to ensure my test plans were accurate.

In terms of autonomy, I was often tasked with owning the QA for specific features from start to finish. This involved designing the test strategy, executing the tests, logging defects with detailed steps and evidence in Jira, and following through with developers until resolution—all with minimal supervision. My background in technical support has also honed my written and verbal communication, whether it's documenting a complex bug for a developer or explaining a technical issue to a non-technical stakeholder."

"Tell us about your experience with managing and logging incident reports."

"I have extensive experience using Jira for the entire incident management lifecycle. As a QA Engineer, logging clear, actionable bug reports was a daily task. I ensured every ticket included a descriptive summary, detailed reproduction steps, expected vs. actual results, environment details, and attachments like screenshots or log files.

Beyond just logging, I also managed these reports. I prioritized bugs based on severity and impact, tracked them through to resolution, and verified fixes. My involvement extended further—I also assisted in planning our quarterly tasks for the automation team by helping break down large testing initiatives into manageable Jira tasks and epics, ensuring our long-term goals were aligned and trackable."

PART 3: AGILE & METHODOLOGY QUESTIONS

"Can you describe your experience with Agile software development and testing methodologies?"

"Yes, I have both academic and professional experience with Agile. Professionally, my entire workflow at Convoso was built around Agile Scrum. We worked in two-week sprints, which involved daily stand-ups, sprint planning, backlog grooming, and retrospective meetings.

In my role, I was deeply integrated into this cycle. During sprint planning, I provided testability feedback on user stories. During the sprint, I executed tests continuously as features were developed, not just at the end. This continuous testing is a core Agile testing principle.

Furthermore, I led my capstone project at SAIT as the Project Manager, where I applied Agile methodologies directly. I broke down the project into epics and user stories in ClickUp, facilitated weekly sprint meetings, and ensured testing was not a final phase but an ongoing activity parallel to development. This hands-on leadership experience solidified my understanding of how QA is not a separate phase in Agile but an integral part of the development flow."

"What is the role of a QA engineer in an Agile team?"

"In an Agile team, a QA engineer is not a gatekeeper at the end of a sprint but an integrated partner throughout the development process. My role at Convoso involved:

- **Sprint Planning**: Participating in planning meetings to understand new user stories and provide feedback on testability and potential edge cases.
- **Continuous Testing**: Testing features as soon as they are developed, not waiting for the end of the sprint.
- **Collaboration**: Working closely with developers daily to discuss requirements, clarify behavior, and verify fixes. This is the 'collaborated with developers to understand requirements' point on my resume.
- **Feedback Loop**: Providing rapid feedback on quality, which allows for quicker iterations and a higher-quality product release."

PART 4: TESTING CONCEPTS & DEFINITIONS

"What is the difference between Manual Testing and Automation Testing?"

"Manual testing is the process where a QA engineer personally executes test cases without using any automation tools. It's essential for exploratory testing, usability assessment, and testing features that are newly developed or frequently changing. For example, at Convoso, I performed manual testing when a new reporting feature was first built to check its look, feel, and basic workflow.

Automation testing, on the other hand, uses scripts and tools to execute tests automatically. It's ideal for regression testing, load testing, and repeating lengthy test cases. While my role at Convoso was primarily manual, I have used Cypress and Selenium to automate repetitive login and navigation steps, which is a foundation I plan to build upon. Manual testing requires human intuition, while automation provides speed and repeatability."

"Can you explain what Regression Testing is and provide an example?"

"Regression testing is the process of testing existing software functionality to ensure that new changes, like features, enhancements, or bug fixes, haven't adversely affected the existing, working parts of the application.

Example from my experience: At Convoso, when the development team released an update to a core feature, my regression test suite would include not just testing the new functionality, but also re-testing related features. I would verify that:

- User authentication and session management still worked flawlessly
- Related workflow features still functioned correctly with the new changes
- The user interface and data displays had not been broken

This comprehensive approach ensured that a fix in one area didn't create a bug in another, maintaining overall platform stability."

"What is Smoke Testing and Sanity Testing? How are they different?"

"Smoke and sanity testing are both types of shallow, focused testing, but they are used at different stages.

Smoke Testing is done immediately after a new build is received to ensure the most critical functions of the application are working and the build is stable enough for further testing. It's like turning on a device to see if it smokes. For a SaaS app like Convoso's, my smoke test would be: 'Can a user successfully log in, access their dashboard, and perform a basic action?' If any of these fail, I would reject the build for immediate fixes.

Sanity Testing is performed after a specific bug fix or a minor change. It's a narrow, deep test to verify that the particular fix worked and that the related functionality is sane. For instance, if a bug was fixed where a specific automation rule wasn't triggering, my sanity test would focus only on that specific scenario to confirm it's now working, without retesting the entire application suite."

"What are the key components of a good bug report?"

"A good bug report is clear, concise, and contains all the information a developer needs to understand, reproduce, and fix the issue. The key components are:

1. **Descriptive Title**: A summary that instantly conveys the problem
2. **Detailed Steps to Reproduce**: A step-by-step guide that allows the developer to see the bug themselves
3. **Expected vs. Actual Result**: What should have happened versus what actually happened
4. **Environment Details**: OS, Browser, App Version, etc. (e.g., 'Tested on Chrome 119, Windows 11')
5. **Evidence**: Screenshots, screen recordings, or log files
6. **Severity and Priority**: How bad the bug is and how urgently it needs to be fixed

Example from my work: In Jira at Convoso, I once reported: 'Title: Workflow fails to execute when triggered by users with specific role permissions.' The steps detailed the exact user roles and the trigger conditions. I included a screenshot of the error message and a log snippet. This level of detail helped the developer identify a permission validation issue in the workflow logic quickly."

PART 5: PRACTICAL TESTING SCENARIOS

"What is your approach to testing a completely new feature?"

"My approach is structured and thorough:

1. **Requirement Analysis**: I start by deeply understanding the feature requirements and user stories by collaborating with the Product Team, as I did at Convoso.
2. **Test Plan & Case Design**: I then design a test plan outlining the scope, strategy, and resources. I write detailed test cases covering positive, negative, and edge-case scenarios.
3. **Environment Setup**: I ensure the test environment is configured correctly to mimic production.
4. **Execution**: I execute the test cases, starting with positive 'happy path' tests, then moving to negative testing (entering invalid data), and finally edge cases.
5. **Exploratory Testing**: Beyond the scripted cases, I spend time exploring the feature

intuitively, trying to break it in ways a real user might.

6. **Regression Testing**: Once the feature is stable, I run a regression suite to ensure it hasn't impacted existing functionality.
7. **Reporting & Closure**: I log any defects found and retest them once fixed, finally signing off on the feature for release."

"How would you test a login functionality?"

"I would test the login functionality very systematically:

Positive Testing:

- Valid username and password → Successful login

Negative Testing:

- Invalid username, valid password → Error message
- Valid username, invalid password → Error message
- Both fields empty → Error message

Security Testing:

- Check if the password is masked (shown as dots)
- Verify there is a rate limit or lockout after multiple failed attempts
- Check if the session expires after a period of inactivity
- Ensure the login occurs over a secure (HTTPS) connection

Usability & UI Testing:

- Are the error messages clear and user-friendly?
- Is there a 'Forgot Password' link, and does it work?
- Is the layout responsive on different browsers and devices?

Database Testing:

- When a user logs in successfully, does the application correctly record the login time in the database?"

PART 6: TOOLS & TECHNICAL SKILLS

"Do you have any experience with the tools mentioned, like Jenkins, MySQL, or Linux?"

"Yes, I have experience with each of these:

MySQL: I've used it extensively in my projects, such as the Rental Management System, where I designed the database schema and wrote queries to manage data. I'm also proficient with SQL Server Management Studio and PostgreSQL.

Linux: My foundational knowledge of Linux and command-line operations was gained during my Software Development diploma program at SAIT, where it was a part of the curriculum and used in various project environments.

Jenkins: I have familiarity with Jenkins as a CI/CD tool from my studies and have used it in a learning context to understand automated build and deployment pipelines. I am eager to gain more hands-on experience in a professional setting."

"Can you describe your experience with relational databases like MySQL?"

"Yes, I have both academic and practical experience. I learned fundamental SQL and database design in my Software Development diploma at SAIT. I'm currently applying and expanding this knowledge in my capstone project, where I use PostgreSQL as the primary database.

To interact with it, I'm using modern tools and ORMs. I'm using Supabase as a backend-as-a-service and have also worked with Convex. Furthermore, I'm currently learning Prisma as my ORM to write efficient, type-safe database queries

from my application. This gives me a well-rounded understanding of how data is structured, accessed, and managed in a real-world application."

"Do you have any experience with Jenkins?"

"Yes, I have practical experience with Jenkins in a CI/CD context. At Convoso, I use Jenkins to run automated regression suites, especially in preparation for a release.

I am responsible for triggering these jobs, monitoring their execution, and analyzing the test results reports they generate. If a test fails in the pipeline, I can trace it back to the specific code change and environment, which is crucial for maintaining stability. I understand its role in automating the build and test process to ensure quality is continuously integrated."

PART 7: AUTOMATION EXPERIENCE

"The description mentions a willingness to support automation in the future. What is your exposure to test automation or scripting?"

"I have a solid foundational exposure to automation that I'm very motivated to expand. I've completed a Cypress certification on Udemy and have practical knowledge of Selenium. While my professional work at Convoso was primarily manual, I understand the principles and value of automation.

Furthermore, I've used Python in several of my SAIT projects, such as the Library Management Application, for back-end logic and file handling. I'm confident in my ability to read and understand Python scripts, which is a plus for this role as mentioned. I see this position as a perfect opportunity to deepen my automation skills by initially supporting and eventually contributing to your automation initiatives."

"The job description mentions automated testing as a plus. What is your experience there?"

"I am very proactive about growing my automation skills. At Convoso, we have a culture of continuous learning, and I dedicate at least two hours per week to it. Recognizing the team's need, I took a Cypress course on Udemy for two months to build a solid foundation.

This initiative paid off and I successfully transitioned from the manual testing team to contribute to the automation team. I'm currently working on a significant project where I am authoring and maintaining automated test scripts. This hands-on experience, combined with my manual testing background, allows me to write effective, reliable automation."

PART 8: HR & SOFT SKILLS QUESTIONS

"Tell me about a time when you had a conflict with a developer or team member. How did you handle it?"

"At Convoso, a developer marked my bug report as 'Working as Intended,' but I believed it was a genuine issue where users could save incomplete workflows that would fail later. Instead of escalating, I scheduled a one-on-one to understand their perspective and gathered evidence showing how this would create customer support issues. We had a productive discussion where I reframed it as a usability concern rather than a technical bug. Together, we agreed to add validation warnings instead of blocking the save entirely, which was better than either initial solution. This taught me that conflicts often come from different perspectives, and curiosity leads to better outcomes than being right."

"Describe a time when you had to meet a tight deadline. How did you prioritize your work?"

"At Convoso, we had a major feature that needed to launch for a key client within two weeks while also handling ongoing regression testing. I did a quick risk assessment to identify which existing features were most likely to be impacted, then had an honest conversation with my manager about what I could realistically test and what trade-offs we were making. I time-blocked my mornings for the complex new feature when I'm most focused, and afternoons for regression work. Because I had previously created detailed test cases in TestRail, I could execute regression tests efficiently without planning from scratch. I provided daily status updates so everyone knew our progress. We met the deadline with zero critical bugs, and I learned that transparent communication and strategic prioritization are essential under pressure."

"Tell me about a time you made a mistake. How did you handle it?"

"Early at Convoso, I approved a feature for release but missed testing an edge case where special characters in account names caused display errors in the reporting dashboard. A customer discovered it in production, which was embarrassing. I immediately took ownership without making excuses and informed my manager and the dev team. I analyzed why I missed it and realized my test cases focused on happy path scenarios but didn't systematically test special character handling. I worked with the developer to quickly verify and release the fix, then updated our test case template to include mandatory input validation testing for all user-input fields. I shared this learning in our retrospective so the whole team could benefit. My manager appreciated my accountability and proactive approach to preventing similar issues. This made me a more thorough tester and taught me that mistakes are growth opportunities when handled with transparency."

"How do you handle a situation where you're given unclear or incomplete requirements?"

"This happens often, and I've developed a proactive approach. At Convoso, we once received a user story that simply said 'As an admin, I want to bulk update user statuses' with no acceptance criteria or details. Instead of making assumptions or waiting, I documented specific questions like what's the maximum number of users, should there be a confirmation dialog, what happens if updates fail for some users, and should there be an audit log. I scheduled a clarification session with the Product Owner before the sprint started and proposed concrete user scenarios to make the abstract requirement tangible. Together, we updated the story with clear acceptance criteria before development began. This prevented rework, saved development time, and resulted in a better feature. I believe it's a QA engineer's responsibility to ask the hard questions early, not just find problems late."

"Describe a time when you had to say 'no' or push back on something."

"At Convoso, as we approached a release deadline, the product team asked if we could skip regression testing on a 'low-risk' module to save time. I understood the pressure but had to push back professionally. I showed them examples from our bug history where unchanged modules broke due to shared dependencies or database changes from other features. Instead of a flat refusal, I proposed running a smoke test on that module covering only critical paths, which would take two hours instead of a full day. I clearly explained that untested code could lead to production issues and more time spent on hotfixes than we'd save now, and I offered to stay late to complete the critical regression tests. The team appreciated my perspective and agreed to the compromise. We actually found a critical bug that would have broken a key workflow for customers. This experience reinforced that saying no isn't about being difficult—it's about being a responsible advocate for quality and helping the team understand risks."

"How do you stay motivated when doing repetitive tasks like regression testing?"

"I stay motivated by focusing on impact and continuous improvement. I remind myself that every test protects our customers—I once caught a critical billing calculation bug during routine regression testing at Convoso that would have overcharged customers, which justified all the repetitive work. Even when executing the same test cases, I'm always looking for ways to improve them or identify edge cases I haven't considered. I also set personal goals to gamify the work and approach testing with variety by trying different browsers, user roles, or data sets. My repetitive testing actually made me an expert in our product, which makes me valuable for troubleshooting. Additionally, recognizing that some regression work should be automated motivated me to get my Cypress certification and transition to the automation team at Convoso. I took initiative to learn automation specifically to make repetitive tasks more efficient for everyone."

"Tell me about a time you received critical feedback. How did you respond?"

"During my first few months at Convoso, my manager told me my bug reports were too verbose and took developers longer to understand because I included too much background before getting to the reproduction steps. Initially I felt defensive because I thought I was being thorough, but I asked her to show me specific examples so I could see it from the developer's perspective. I studied bug reports from senior QA engineers and noticed they used a clean format with steps right at the top. I created a personal template that forced me to be concise and put reproduction steps first, then results, with any additional context at the bottom. A few weeks later I asked a developer if my reports had improved and he confirmed they were much easier to parse. When I later led the test case standardization project, I incorporated this feedback into the templates for the whole team. This taught me that feedback is a gift and being defensive only prevents growth, so now I actively seek feedback in my one-on-ones."

"How do you handle stress or pressure in your work?"

"I handle stress by staying organized, communicating proactively, and maintaining perspective. Last year at Convoso, a major production bug affecting a high-value customer was reported on a Friday afternoon, and everyone was looking to QA to verify the fix quickly. I resisted the urge to rush and carefully reproduced the issue first, which actually helped the developer identify the root cause faster. I gave the team realistic time estimates for testing the fix and regression testing related features, prioritized ruthlessly to identify which tests were absolutely critical before releasing the hotfix, and took a five-minute break before final verification to prevent careless mistakes. I reminded myself that while the situation was urgent, panicking wouldn't help and a rushed fix that breaks something else would make things worse. We resolved the issue that evening with no new bugs introduced, and the experience reinforced that stress is manageable when you have a systematic approach and don't let urgency compromise quality."

"What do you do when you disagree with a decision made by management or the product team?"

"I believe in respectful advocacy while ultimately supporting team decisions once they're made. At Convoso, the product team once decided to release a feature with some known minor bugs that I had documented. I disagreed, but I made my case clearly by documenting the issues with specific examples of how they could impact users and stating my recommendation to delay by one sprint. I then listened to their reasoning and asked questions to understand their perspective—they explained that a key client urgently requested this feature and delaying could risk contract renewal. Once I understood the full business context they were weighing, I accepted their decision and

actively supported the release by creating a list of known issues for the support team and documenting workarounds for customers. I tracked the known issues in production, and as it turned out, customers didn't report most of them as problems, which taught me that my assessment of risk isn't always aligned with real-world impact. This experience showed me that while my job is to advocate for quality, I need to trust that leadership has information and pressures I may not fully understand."

"How do you continue learning and staying current with QA practices and tools?"

"I'm very intentional about continuous learning. At Convoso, we dedicate at least two hours per week to learning, which I take seriously—I completed a two-month Udemy Cypress certification during these sessions that directly led to my transition to the automation team. I don't just consume content; I apply it immediately. I'm currently learning Prisma ORM in my SAIT capstone project for practical database interaction experience. I actively participate in code reviews and pair testing sessions with senior team members, asking questions when I see new approaches. I follow QA blogs, testing newsletters, and occasionally watch conference talks on YouTube. I'm completing my Software Development diploma at SAIT, graduating in less than two months, which has given me strong foundations in programming and databases. The test case standardization process I led at Convoso came from learning about documentation best practices and bringing those ideas back to my team. I believe the QA field constantly evolves with automation, AI-assisted testing, and shift-left practices, so staying current isn't optional—it's essential to being effective."

PART 9: QUESTIONS TO ASK THE INTERVIEWER

When they ask "Do you have any questions for us?", use these:

1. **"Can you describe the typical structure of a sprint or release cycle, and how the QA team integrates with the development and DevOps teams throughout that process?"**
2. **"The job description mentions future automation readiness. Could you share what the current vision or roadmap for test automation looks like, and what the first steps for this role would be in that journey?"**
3. **"What are the biggest challenges the team is currently facing regarding product quality or testing, and how could this role help overcome them?"**

PART 9: CLOSING QUESTIONS

"Why should we hire you?"

"You should hire me because I offer proven SaaS QA experience and can start contributing from day one. I don't just find bugs; I improve processes, and I already have a foundation in automation, making me a strong long-term asset."

"When can you start?" / Addressing Graduation Timeline

"I graduate in less than two months. I can start immediately, working 24 hours per week, and seamlessly transition to full-time upon graduation. This allows me to contribute valuable work now without delay."

QUICK REFERENCE: Key Definitions

Difference Between Regression, Smoke, and Sanity Testing

Smoke Testing: "A quick check to verify the most critical features work and the build is stable. It's the 'sanity check' for a new build. Example: After a new deployment, I verify that a user can log in and access core features."

Sanity Testing: "A narrow, deep test on a specific bug fix or feature to ensure that one particular change works as expected. Example: After fixing a specific bug, I test only that specific functionality to confirm it's now working."

Regression Testing: "A comprehensive test of the entire application to ensure that new changes haven't broken any existing functionality. Example: Before a major release, I run a full suite of tests on all major features and user workflows."

In short:

- **Smoke**: "Is the build healthy enough to test?"
- **Sanity**: "Did the specific fix work?"
- **Regression**: "Did the new change break anything that was already working?"