# Senior QC Engineer – Interview Questions & Answers

## 1. Steps in the Quality Control Process & Alignment with Business Requirements

Steps:

1. **Define Quality Standards** – Work with stakeholders to set measurable KPIs based on user expectations and business goals.
2. **Plan Test Strategies** – Design a test plan that includes functional, performance, security, and usability testing.
3. **Test Execution** – Conduct manual and automated testing, covering unit, integration, and system testing.
4. **Defect Reporting & Triage** – Prioritize issues based on impact, ensuring critical defects are resolved first.
5. **Validation & Acceptance Testing** – Engage business owners for UAT to confirm product readiness.
6. **Continuous Improvement** – Implement root cause analysis and retrospective meetings to refine processes

- Alignment:  
 - Review requirements with business analysts/product owners.  
 - Create traceability matrix (RTM) to Map test cases and bugs directly to business requirements to ensure complete coverage.  
 - Participate in backlog grooming and sprint planning.  
- Risk-Based Testing – Prioritize test scenarios based on business impact and critical functionalities.  
-Always double check burnout chart with deadline to avoid overwilling.

## 2. Black-box vs. White-box Testing

### Black-Box Testing:

#### - Focuses on functional behavior.

#### - Testers do not see the internal code.

#### - Used for UI, integration, and system testing.

#### - Example: Validating login functionality with different credentials.

is preferred for **validating business workflows**

### White-Box Testing:

#### - Focuses on internal code and logic.

#### - Testers need programming knowledge.

#### - Used for unit testing, security testing, and code review.

#### - Example: Verifying edge cases in an API response by testing different conditions

#### - Cost less as detect earlier

## 3. Agile Methodologies & QA Integration

### - Attend daily standups and sprint Meetings. -Analysis & write test cases from user stories. -Make sure to meet estimations and conduct Sprint spike. - Shift-left testing – Start from planning meeting and changes made to DB. - Integrate automated tests into Automation framework. -Daily Stand-ups to check blockers and critical or high defects. -Root-Cause analysis for sudden issues to make sure that no ghosting or unnormal issues happen while stage or pre-production.

## 4. Risk-Based Testing & Test Prioritization

### - Focus on high-impact, high-likelihood features like Core functionalities, payment processing and Cron jobs. - Steps: Identify risks → Assess impact/likelihood → Prioritize testing accordingly. High-risk –> as blockers - Medium-risk -> Performance & UI validations. - - Low-risk -> Minor UI inconsistencies, Image quality, Colors, Formats or Typo.

## 5. Essential Components of a Test Case

### - Includes: ID, Title, Description, Preconditions, Steps, Expected Result, Actual Result, Priority, Status, Environment, Execution date, Tags, Attachements. TC001: Validate that active user able to valid login Preconditions: User has a valid , Active account. Steps: 1. Navigate to the login page. 2. Enter valid username and password. Valid\_Username/Email – Valid\_password 3. Click "Login". Expected Result: - User is redirected to the dashboard. -Toast message logged in successfully. Actual Result: - Pass (Evidence).

## 6. Ensuring Test Coverage

### - Use RTM (requirement traceability matrix). TCs status pass/Fail rate Bugs Severity, Status and areas - Tools: Jira, Azure, TFS. - Combine manual and automation coverage metrics.

## 7. Automation Tools & Tool Selection

### - Tools: Selenium (Java), Cypress (JavaScript), Playwright (TypeScript), Postman, JMeter. - Criteria: - App type, skillset. - Project Needs – UI testing vs API testing. - Language Support – Matching team skills. - Integration – CI/CD compatibility. - Performance – Execution speed.

## 8. Version Control in QA

### - Use Git with best practices: - Feature branches -> feature/test-automation - Code reviews - Tag stable releases - Sync with application codebase.

## 9. Importance of Test Metrics

### - Metrics improve visibility and quality. - Critical: Pass rate, defect density, test coverage, defect leakage, execution progress.

## 10. Measuring Testing Success

### - Indicators: - Test completion rate - Zero critical defects - Low leakage – through pre-prod regression - High coverage

## 11. Example of a Challenging Defect

### - Issue: user encounter multiple payment while no evidence for that. -Checked the logs, transactions, Database records and API request through postman but issue still exist only on Stage server, while not exist on testing server. -while checking 3rd part service that store customer credit cards and transactions -> show that the customer is billed twice every time. -Solution: Have a meeting with Senior developer aside with Team lead , and went through every line of the code and make comments and through enrtupts to check where’s the issue -Finally we found that another payment function outside payment process array {}, where not logged or saved in DB, thanks Allah didn’t went live.

## 12. Root Cause Analysis of Recurring Defect

### Steps: 1. Reproduce issue with different user types and account statuses 2. Separate API and FE / Mobile testing each alone. 2. Analyze logs/ Database records 3. Trace to affected code 4. Collaborate with team

## 13. Handling Team Conflicts on Priorities

### - Listen to all sides. - Use data to and requirements to check severity. - Discuss impact and time frame to complete rood cause. - Make sure that we are on same boat and have 1 single goal to deliver the best product with the best quality and ensure client satisfaction.

## 14. Communicating Test Results to Non-Technical Stakeholders

### - Use summaries, visual dashboards charts, and business terms. - User Numbers, effort and time for completion.