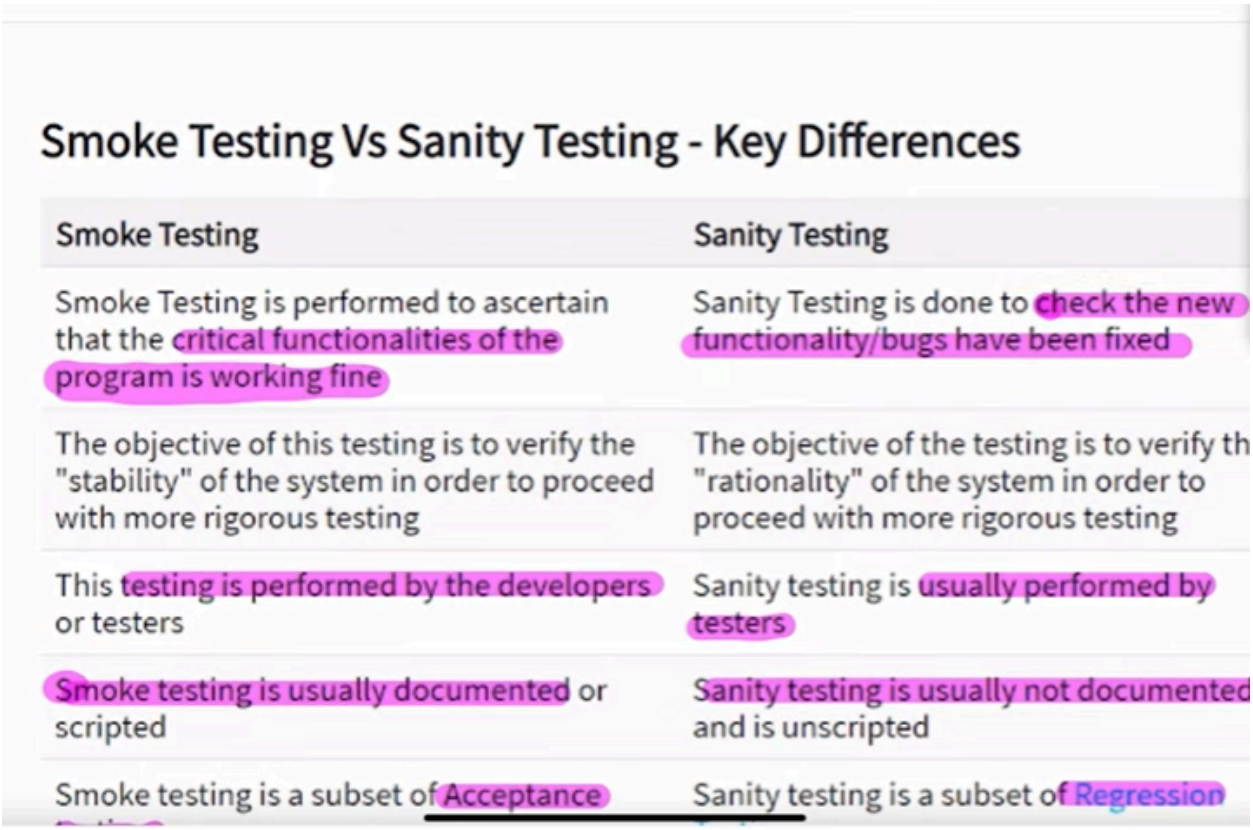
1. What to do if you find a production bug

    - Start analysis → add bug in jira → RCA process : Root cause analysis →inform stakeholders
2. What to do when developer say this is not a bug
    - Analyze and research within requirements
    - Pin point the requirement and bug , explain about the missing area and impact of the bug
    - Add proofs (screenshot,video)

3. SANITY VS SMOKE TESTING



Smoke Testing Vs Sanity Testing - Key Differences

Smoke Testing	Sanity Testing
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality/bugs have been fixed
The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify th "rationality" of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Acceptance	Sanity testing is a subset of Regression

4. What do you understand by software testing?

Software testing is a validation process that confirms that a system works as per the business requirements. It qualifies a system on various aspects such as usability, accuracy, completeness, efficiency, etc.

5. What is the purpose of the end-to-end testing?

End-to-end testing is a testing strategy to execute tests that cover every possible flow of an application from its start to finish. The objective of performing end-to-end tests is to discover software dependencies and to assert that the correct input is getting passed between various software modules and sub-systems.

6. How do you prioritize test cases when you have limited time for testing?

We can prioritize test cases when we have limited time for testing in several scenario, including:

1. Identify critical functionalities: Focus on testing essential software features critical for user satisfaction and core operations.
2. Assess risk impact: Prioritize test cases based on their potential impact on software stability, user experience, and business objectives.
3. Consider frequency of use: Test functionalities frequently used to ensure thorough evaluation of critical aspects.
4. Focus on high risk areas: Allocate more time in testing areas to defects, implemented features, or complex functionalities.
5. Engage with stakeholders: Talk to project participants to coordinate testing strategies with business objectives and confirm that priorities match the requirements.

7. How do you ensure adequate test coverage in your testing process?

To ensure comprehensive test coverage in your testing process, consider the following strategies:

1. Thorough Requirement Analysis: Understand project requirements to identify all functionalities and scenarios requiring testing.
2. Risk-Focused Testing: Arrange testing according to the consequences and chances of failure for each feature.
3. Thorough Test Scenario Development: Develop test scenarios that address negative and boundary conditions.
4. Exploratory Testing: Perform tests to uncover issues.
5. Code Coverage Evaluation: Employ tools to track code execution during testing, guaranteeing that all code pathways are assessed.
6. Traceability Matrix: Map test cases to requirements for tracking and ensuring all requirements are tested.
7. Continuous Improvement: Regularly update test cases to reflect changes, ensuring evolving coverage over time.

8. If a product is in the production stage and one of its modules gets updated, then is it necessary to perform regression testing?

Yes, it is necessary to perform regression testing when a module of a product in the production stage gets updated. Regression testing helps ensure that the changes made to the updated module do not have unintended effects on other modules or the overall

functionality of the product. By retesting the previously working functionalities, it helps identify any potential issues or regressions caused by the module update. This testing process helps maintain the quality and stability of the product throughout its lifecycle.

These were some basic manual testing interview questions. In the following section, we will present some Intermediate manual testing interview questions.

9. How will you overcome the challenges faced due to the unavailability of proper documentation for testing?

If standard documents like the system requirement specification or feature description document are not available, then QA may have to rely on the following references, if available.

- Screenshots
- A previous version of the application
- Wireframes

Another reliable way is to have discussions with the developer and the business analyst. It helps in solving the doubts, and it opens a channel for bringing clarity on the requirements. Also, the emails exchanged could be useful as a testing reference.

Smoke testing is yet another option that would help verify the main functionality of the application. It would reveal some very basic bugs in the application. If none of these work, then we can just test the application from our previous experiences

10. Is there any difference between retesting and regression testing?

Differences between retesting and regression testing are as follows:

- We perform retesting to verify the defect fixes. But, regression testing assures that the bug fix does not break other parts of the application.
- Regression test cases verify the functionality of some or all modules.
- Regression testing ensures the re-execution of passed test cases. Whereas, retesting involves the execution of test cases that are in a failed state.
- Retesting has a higher priority over regression. But in some cases, both get executed in parallel.

11. What are the different types of functional testing?

Functional testing covers the following types of validation techniques:

- Unit testing
- Smoke testing
- UAT
- Sanity testing
- Interface testing
- Integration testing
- System testing
- Regression testing

12. What are functional test cases and non-functional test cases?

- Functional Test Cases: Functional test cases are designed to evaluate the functionality of a software system or application. These test cases focus on verifying whether the system performs its intended functions correctly and meets the specified functional requirements. Functional test cases typically involve validating inputs, testing different scenarios, and verifying expected outputs.
- Non-Functional Test Cases: Non-functional test cases, on the other hand, assess the non-functional aspects of a software system or application. These test cases evaluate performance, usability, reliability, security, scalability, and compatibility. Non-functional testing cases ensure the system meets the required quality standards and provides a satisfactory user experience.

13. What do you understand about STLC?

Software Testing Life Cycle (STLC) is like a roadmap that guides how we test software, starting with planning and ending with making sure everything works as intended.

Software testing life cycle (STLC) proposes the test execution in a planned and systematic manner. In the STLC model, many activities occur to improve the quality of the product.

The STLC model lays down the following steps:

1. Requirement Analysis
2. Test Planning
3. Test Case Development
4. Environment Setup
5. Test Execution
6. Test Cycle Closure

14. How do severity and priority relate to each other?

Severity: It represents the gravity/depth of a bug. It describes the application point of view.

Priority: It specifies which bug should get fixed first. It defines the user's point of view.

15. What is the purpose of a traceability matrix, and how do you create and maintain one?

The traceability matrix aligns requirements with test cases, ensuring thorough test coverage. To create and maintain it, identify project artifacts, establish traceability links, and update it continuously. Use the matrix for reporting and analysis to track test coverage effectively.

16.Can you describe the defect life cycle and the different stages involved in defect management?

The defect life cycle outlines stages in defect management:

- Identification: The defect is logged.
- Assignment: A team member analyzes the defect.
- Open: The defect is confirmed.
- In Progress:The developer fixes the defect.
- Fixed: The defect is resolved.
- Retesting:The defect is ready for testing.
- Reopened: If necessary, the defect is revisited.
- Closed: Verified defect closure.

17. Describe the process you follow for creating and executing test cases in a manual testing environment.

Creating and executing test cases in a manual testing environment involves several key steps:

1. Requirement Analysis: Thoroughly understand project requirements.
2. Test Planning: Develop a comprehensive test plan.
3. Test Case Design: Create detailed test cases for various scenarios.
4. Test Data Preparation: Gather or create the necessary test data.
5. Test Environment Setup: Ensure the readiness of the testing environment.
6. Test Execution: Meticulously execute test cases and record results.
7. Defect Reporting: Document encountered defects with clear descriptions.
8. Defect Tracking: Monitor progress in defect resolution.
9. Regression Testing: Ensure changes don’t introduce new defects.
10. Test Closure: Evaluate results and provide comprehensive summary reports.

18. What do you mean by defect detection percentage in software testing?

Defect detection percentage (DDP) is a type of testing metric. It indicates the effectiveness of a testing process by measuring the ratio of defects discovered before the release and reported after the release by customers.

For example, let's say, the QA has detected 70 defects during the testing cycle and the customer reported 20 more after the release. Then, DDP would be: 70/(70 + 20) = 72.1%

19. On the basis of which factors would you consider choosing automated testing over manual testing?

Choosing automation testing over manual testing depends on the following factors:

1. Tests require periodic execution.
2. Tests include repetitive steps.
3. Tests execute in a standard runtime environment.
4. Automation is expected to take less time.
5. Automation is increasing reusability.
6. Automation reports are available for every execution.
7. Small releases like service packs include a minor bug fix. In such cases, executing the regression test is sufficient for validation.

In such cases, executing the regression test is sufficient for validation.

20. How do you prioritize test cases when you have limited time for testing?

When it comes to prioritizing tests, I employ techniques such as risk-based testing and impact analysis. This entails giving priority to test cases based on their criticality to the applications functionality, frequency of usage, and potential consequences of failure. By addressing high-risk areas, I ensure that essential functionalities receive testing within the allocated time frame. This approach optimizes my testing efforts. Enhances the quality of the software product.

21.Can you explain the difference between smoke testing and sanity testing?

Smoke Testing:

Smoke testing is a form of software evaluation that aims to validate whether critical features of an application are working properly. It is usually carried out in the development phase immediately after deploying a build to ensure that vital functions are operational and that the application is sufficiently stable for testing.

The primary focus of smoke testing lies in identifying any issues that could impede the testing process.

Sanity Testing:

Sanity testing, a type of regression testing, focuses on assessing the functions or sections of the software after changes have been implemented. Its goal is to confirm that recent updates or repairs have not adversely affected the features of the software.

Compared to smoke testing, sanity testing has scope. Specifically targets areas affected by recent changes.

22.How do you ensure adequate test coverage in your testing process?

Several strategies used to ensure adequate test coverage are:

1. Requirement Coverage: Ensure test cases cover all specified requirements and functionalities.
2. Risk Prioritization: Focus testing on high-risk areas or critical functionalities.
3. Boundary and Edge Cases: Test inputs at boundaries and edge conditions for accurate behavior.
4. Code Coverage Analysis: Measure the code exercise by the test suite to identify gaps.
5. Regular Reviews and Feedback: Review and enhance test cases periodically for improved coverage.

23.Is there any difference between bug leakage and bug release?

Bug leakage: Bug leakage is when the bug is discovered by the end user/customer and missed by the testing team. It is a defect that exists in the application and not detected by the tester, which is eventually found by the customer/end user.

Bug release: A bug release is when a particular version of the software is released with a set of known bug(s). These bugs are usually of low severity/priority. It is done when a

software company can afford the existence of bugs in the released software but not the time/cost for fixing it in that particular version.

24. How do you ensure thorough test coverage in a manual testing scenario, particularly when dealing with complex software functionalities?

To achieve test coverage in a testing scenario particularly when dealing with complex software functionalities there are several strategies that can be employed:

1. Understanding the Requirements; It is crucial to comprehend the project requirements in order to identify functionalities and potential edge cases that necessitate testing.
2. Creating a Comprehensive Test Plan; Developing a test plan is essential. This plan should outline objectives, scope, available resources and timelines. Test scenarios should be prioritized based on their criticality and complexity.
3. Designing Detailed Test Cases; The creation of test cases is paramount. These test cases should encompass scenarios such, as negative, boundary and edge cases.
4. Conducting Exploratory Testing: This approach helps us discover any defects that might have slipped through unnoticed otherwise.
5. Implementing Risk Based Testing; Prioritizing testing efforts based on risk assessment is essential for resource allocation. Critical functionalities and areas prone to defects should receive attention in terms of time and resources in order to ensure coverage.

25. How would you test the user registration process to ensure it functions correctly? What validations and verifications would you perform?

To ensure the user registration process works flawlessly, consider the following steps:

- Setup: Ensure the registration feature is accessible and operational.
- Positive Tests:
    1. Valid Registration: Verify the successful registration with the correct details.
    2. Username Uniqueness: Confirm usernames are unique.
    3. Password Strength: Validate password complexity requirements for security.
    4. Email Validation: Ensure correct email formatting and validation.
    5. Confirmation Email: Verify users receive confirmation emails post-registration.
- Negative Tests:
    1. Invalid Email Format: Test registration with incorrectly formatted emails.
    2. Existing Username: Attempt registration with an already used username.
    3. Weak Password: Test with passwords that do not meet complexity criteria.
    4. Empty Fields: Validate mandatory fields and corresponding error messages for empty inputs.
- Edge Cases:

1. Long Username/Password: Test with unusually long inputs for system stability.
2. Special Characters: Validate registration with special characters in inputs.
3. Session Timeout: Ensure registration can be completed after session timeouts.

- Performance:
    1. Load Testing: Assess registration performance under different loads.
    2. Response Time: Measure response times for prompt feedback.
- Security:
    1. Data Encryption: Ensure sensitive data encryption during storage.
    2. SQL Injection: Test for protection against SQL injection attacks.
- Cross Device Testing:
    1. Browser Compatibility: Validate across browsers for consistency.
    2. Device Compatibility: Test on various devices for responsiveness.