Manual Testing Interview Question

Interview Question for Manual testing

1. Question:

What is the difference between Verification and Validation?

Answer:

- **Verification**:
  - Ensures the product is being built correctly according to requirements and design specifications.
  - Focuses on static testing (reviews, walkthroughs, inspections).
  - Example: Reviewing requirement documents, design documents, or code.
- **Validation**:
  - Ensures the product meets the user's needs and works as intended in a real-world scenario.
  - Focuses on dynamic testing (actual testing of the application).
  - Example: Executing test cases, performing functional testing, and checking if the software behaves as expected.

Key Difference: Verification is about "building the product right," while validation is about "building the right product."

---

2. Question:

What are the different levels of testing?

Answer:

1. **Unit Testing**:
   - Focuses on testing individual components or modules of the application.
   - Performed by developers.
2. **Integration Testing**:
   - Verifies the interaction between integrated modules or components.
   - Example: Testing APIs or communication between modules.
3. **System Testing**:
   - Tests the entire application as a whole to ensure it meets the specified requirements.
   - Performed by testers.
4. **Acceptance Testing**:
   - Ensures the application meets business requirements and is ready for deployment.
   - Performed by end-users or clients.

---

3. Question:

What is the difference between Smoke Testing and Sanity Testing?

Answer:

- **Smoke Testing**:
  - Performed to ensure that the basic functionalities of the application are working after a new build.
  - Broad but shallow testing.
  - Example: Verifying that the application launches and main features are accessible.
- **Sanity Testing**:
  - Performed to ensure that specific functionalities or bug fixes are working as expected.
  - Narrow but deep testing.
  - Example: Retesting a login feature after a bug fix to ensure it works correctly.

Key Difference: Smoke testing is a high-level check of the overall system, while sanity testing focuses on specific areas of functionality.

---

4. Question:

How do you prioritize test cases when time is limited?

Answer:

1. **Identify Critical Areas**:
   - Focus on functionalities that are critical to the application, such as login, payments, or data integrity.
2. **Risk-Based Testing**:
   - Prioritize tests in areas prone to defects or those that have undergone recent changes.
3. **Business Impact**:
   - Execute test cases that cover features with the highest business value or end-user impact.
4. **Frequent Use Scenarios**:
   - Test features or workflows that are used frequently by users.
5. **Regression Testing**:
   - Ensure that previously working features are not broken by recent changes.

---

5. Question:

What is the difference between Priority and Severity in defect management?

Answer:

- **Severity**:
  - Refers to the impact of the defect on the functionality or system.
  - Assigned by testers.
  - Levels: Critical, Major, Minor, Trivial.
  - Example: A critical defect causes a system crash, while a minor defect might be a UI misalignment.
- **Priority**:
  - Refers to the urgency of fixing the defect.
  - Assigned by developers or project managers.
  - Levels: High, Medium, Low.
  - Example: A defect on the home page might have high priority even if it's minor.

Key Difference: Severity is about the technical impact, while priority is about the business impact and urgency.

---

6. Question:

What is the purpose of a Traceability Matrix?

Answer:

- A Traceability Matrix is a document that maps test cases to requirements to ensure 100% test coverage.
- **Purpose**:
  1. Verify that all requirements are covered by test cases.
  2. Identify gaps in testing.
  3. Trace defects back to specific requirements.

Example:

Requirement ID	Requirement Description	Test Case ID	Test Case Description	Status
R1	User Login Functionality	TC1	Test Login with valid credentials	Pass

---

7. Question:

What is exploratory testing, and when would you use it?

Answer:

- **Exploratory Testing**: A testing approach where testers actively explore the application without predefined test cases.
- **Purpose**:
    1. Discover hidden defects.
    2. Validate usability and design issues.
    3. Test areas not covered by scripted tests.
- **When to Use**:
    1. When requirements are incomplete or unclear.
    2. During early stages of testing to identify major flaws.
    3. For testing complex workflows or edge cases.

---

8. Question:

How do you ensure a defect report is effective?

Answer:

1. **Clear Title**:
    - Use a descriptive title summarizing the defect.
2. **Detailed Steps to Reproduce**:
    - Provide precise steps, test data, and environment details.
3. **Expected vs. Actual Result**:
    - Clearly state what was expected and what actually happened.
4. **Attachments**:
    - Include screenshots, logs, or videos to provide evidence.
5. **Severity and Priority**:
    - Assign appropriate levels based on impact and urgency.
6. **Environment Information**:
    - Specify browser, device, or OS details where the defect was observed.

---

9. Question:

How would you test a login page?

Answer:
Testing a login page involves a combination of functional, security, and UI testing. Key test cases include:

1. **Functional Testing**:
    - Valid credentials: Verify login with correct username and password.
    - Invalid credentials: Check error messages for wrong username/password.
    - Blank fields: Ensure validation messages appear when fields are left empty.
2. **Boundary Value Analysis**:
    - Test username and password with minimum and maximum character limits.
3. **Negative Testing**:

    - Test with SQL injections, special characters, or script tags.
    - Check for error messages when submitting without input.
4. **Security Testing**:
    - Verify password encryption and secure data transmission (HTTPS).
    - Ensure the application prevents brute-force attacks by locking accounts after multiple failed attempts.
5. **Usability Testing**:
    - Validate field alignment, placeholder text, and ease of navigation.
6. **Cross-Browser Testing**:
    - Verify the login functionality on different browsers and devices.

---

10. Question:

How do you ensure complete test coverage?

Answer:

1. **Understand Requirements**:
    - Thoroughly analyze requirements and create a traceability matrix to map them to test cases.
2. **Categorize Test Scenarios**:
    - Cover all functional, non-functional, edge cases, and integration points.
3. **Include Positive and Negative Tests**:
    - Test normal workflows as well as scenarios with invalid or unexpected inputs.
4. **Use Equivalence Partitioning and Boundary Value Analysis**:
    - Divide test inputs into valid and invalid partitions.
5. **Perform Exploratory Testing**:
    - Execute unscripted tests to identify gaps in predefined test cases.
6. **Review by Peers**:
    - Conduct peer reviews of test cases to ensure no requirements are missed.

---

11. Question:

What is regression testing, and how do you decide what to include?

Answer:

- **Regression Testing**: Ensures that recent changes or fixes do not break existing functionality.
- **Inclusions**:
    1. Test cases for modules impacted by code changes.
    2. Core features of the application.
    3. High-priority defects fixed in recent builds.
    4. Tests for integrations with external systems.

Approach:

1. Use a risk-based strategy to focus on critical workflows.
2. Maintain a regression suite and update it as features evolve.
3. Automate repetitive regression tests to save time.

---

12. Question:

How do you handle incomplete or ambiguous requirements?

Answer:

1. **Clarify with Stakeholders**:
    - Collaborate with business analysts, product owners, or clients to get clarity.
2. **Refer to Similar Features**:
    - Use knowledge from similar modules or past projects as a reference.
3. **Document Assumptions**:
    - Clearly outline assumptions about expected behavior and share them for approval.
4. **Create a Risk-Based Plan**:
    - Focus on critical functionalities and perform exploratory testing for the unclear areas.
5. **Communicate Effectively**:
    - Keep all stakeholders informed about the challenges and testing strategy.

---

13. Question:

What is the difference between Retesting and Regression Testing?

Answer:

- **Retesting**:
    - Verifies that a specific defect is fixed.
    - Focuses on failed test cases.
    - Performed in the same environment with the same inputs.
- **Regression Testing**:
    - Ensures recent changes do not impact existing functionality.
    - Focuses on both fixed and unaffected areas.
    - Covers a broader scope of test cases.

Key Difference: Retesting confirms defect fixes, while regression testing checks for unintended side effects.

---

14. Question:

What is usability testing, and what aspects do you check?

Answer:

- **Usability Testing**: Evaluates how user-friendly and intuitive an application is.
- **Aspects to Check**:
    1. Navigation: Ensure menus, links, and buttons are easy to find and use.
    2. Content Clarity: Verify labels, error messages, and instructions are clear.
    3. Consistency: Check uniformity in font styles, colors, and layout.
    4. Accessibility: Ensure the application is usable for differently-abled users (e.g., screen readers, keyboard navigation).
    5. Performance: Validate response times for user actions.

---

15. Question:

How would you test a payment gateway?

Answer:

1. **Functional Testing**:
    - Verify successful transactions using valid card details.
    - Test failed transactions with invalid, expired, or insufficient funds cards.
2. **Security Testing**:
    - Ensure sensitive data (card number, CVV) is encrypted.
    - Validate secure communication via HTTPS.
3. **Boundary Value Testing**:
    - Test amount limits (minimum and maximum transaction values).
4. **Integration Testing**:
    - Verify integration with third-party payment processors like PayPal or Stripe.
5. **Negative Testing**:
    - Test transaction interruptions (e.g., network failure, session timeout).
6. **Performance Testing**:
    - Simulate high transaction loads to ensure stability under peak usage.

---

16. Question:

What is exploratory testing, and how do you approach it?

Answer:

- **Exploratory Testing**: An unscripted approach to uncover hidden defects by exploring the application.

Approach:

1. Define a charter or focus area (e.g., testing login functionality or user profile).
2. Start with basic functionality, then explore edge cases.
3. Take notes on any unusual or unexpected behavior.
4. Prioritize areas with high complexity or recent changes.
5. Use tools (e.g., session recorders) to document findings.