Manual Testing

Basic and Conceptual Questions

1. What is Software Testing, and why is it important?

- **Definition**: Software Testing is the process of identifying errors, gaps, or missing requirements in a software application by comparing the actual outcomes with expected outcomes.
- **Why it's important**:
  - Ensures software quality.
  - Prevents bugs and errors in the production environment.
  - Enhances user satisfaction and confidence.
  - Saves cost and time by detecting issues early.
- **Example**: Testing an e-commerce website to ensure that the "Add to Cart" feature works correctly under different scenarios.

---

2. Explain the different levels of testing.

- **Unit Testing**: Tests individual components or modules of code.
  - *Example*: Testing a "calculateDiscount()" function to ensure it returns the correct discount value.
- **Integration Testing**: Verifies how multiple modules interact.
  - *Example*: Testing the integration of the login page with the dashboard.
- **System Testing**: Validates the entire system end-to-end.
  - *Example*: Checking an online shopping app's workflow—from product search to payment processing.
- **Acceptance Testing**: Ensures the system meets business requirements and is ready for delivery.
  - *Example*: User Acceptance Testing (UAT) performed by clients to verify that the product behaves as expected.

---

3. What is the difference between verification and validation?

- **Verification**: Ensures the product is built correctly by following processes (Are we building the product right?).
  - *Example*: Reviewing requirement documents and code.

- **Validation**: Ensures the right product is built according to customer needs (Are we building the right product?).
    - *Example*: Performing tests to check whether a "Search" button retrieves accurate results.

---

4. What are the different types of testing?

- **Functional Testing**: Validates that the application behaves as expected based on requirements.
    - *Example*: Testing login functionality.
- **Non-functional Testing**: Focuses on performance, security, usability, and other non-functional aspects.
    - *Example*: Testing how fast a website loads under heavy traffic.
- **Regression Testing**: Ensures new changes don't affect existing features.
    - *Example*: After fixing a bug, re-testing all related areas.

---

5. What is the SDLC and STLC? Explain their phases.

SDLC (Software Development Life Cycle)

SDLC is a systematic process used to develop high-quality software. It outlines a framework for planning, developing, testing, and maintaining software applications.

Phases of SDLC:

1. **Requirement Analysis**:
    - Collect and document functional and non-functional requirements.
    - Tools: Interviews, surveys, and requirement specification documents.
2. **Planning**:
    - Create project plans, timelines, and resource allocation.
    - Outcome: Project charter, risk assessment.
3. **Design**:
    - High-level (HLD) and low-level designs (LLD) are created.
    - Tools: UML diagrams, flowcharts.
4. **Development**:
    - Actual coding takes place based on the design.
    - Tools: Programming languages, IDEs.
5. **Testing**:
    - Verify that the software works as intended and meets requirements.
    - Tools: Selenium, JUnit, etc.

6. **Deployment**:
    - Release the software to the production environment.
    - Techniques: Continuous Deployment (CI/CD pipelines).
7. **Maintenance**:
    - Fix bugs, update software, and ensure scalability.
    - Involves corrective, adaptive, and preventive maintenance.

STLC (Software Testing Life Cycle)

STLC is a sequence of steps followed to ensure that the software product meets quality standards. It focuses on testing the software during various stages of SDLC.

Phases of STLC:

1. **Requirement Analysis**:
    - Understand and analyze testable requirements.
    - Output: Requirement Traceability Matrix (RTM).
2. **Test Planning**:
    - Define the scope, resources, schedule, and strategy for testing.
    - Output: Test Plan document.
3. **Test Case Design**:
    - Write test cases and prepare test data.
    - Tools: Test case management tools like TestRail.
4. **Environment Setup**:
    - Prepare hardware and software environments to execute tests.
    - Includes configuring servers, networks, and databases.
5. **Test Execution**:
    - Execute test cases and report defects.
    - Tools: TestNG, JIRA for defect tracking.
6. **Test Closure**:
    - Analyze test metrics, lessons learned, and prepare final reports.
    - Output: Test Summary Report.

Key Differences Between SDLC and STLC

Aspect	SDLC	STLC
Focus	Overall software development	Testing and quality assurance
Process Start	Begins with requirement gathering	Begins after requirements are analyzed

Participants	Developers, designers, project managers	Testers and QA team
Output	Fully developed software	Certified and quality-checked software

---

6. Define test case and test scenario. How are they different?

- **Test Case**: A set of detailed instructions to validate a specific functionality.
    - *Example*: "Enter valid credentials and click Login; expect redirection to the dashboard."
- **Test Scenario**: High-level overview of what needs to be tested.
    - *Example*: "Verify login functionality."
- **Difference**: Test scenarios are broader and outline what to test, while test cases explain how to test.

---

7. What are the key components of a test plan?

- **Components**:
    1. Test Objectives.
    2. Scope of Testing.
    3. Test Schedule.
    4. Test Resources.
    5. Entry and Exit Criteria.
    6. Risks and Mitigation Plan.

Test Design and Execution

1. How do you write a good test case?

- Include:
    - Test Case ID.
    - Test Description.
    - Preconditions.
    - Test Steps.
    - Expected Results.
    - Actual Results.
    - Status (Pass/Fail).

Example:

Test Case ID	Test Description	Steps	Expected Result	Actual Result	Status
TC001	Verify login	Enter valid credentials	Redirect to Dashboard	Redirected	Pass

---

2. Explain Boundary Value Analysis (BVA) and Equivalence Partitioning (EP).

- **BVA**: Tests values at boundaries.
  - *Example*: For a field accepting age between 18–60, test 17, 18, 60, and 61.
- **EP**: Divides inputs into equivalent groups.
  - *Example*: Age field with valid (18–60) and invalid (<18, >60) partitions.

---

3. What is the difference between positive and negative testing? Provide examples.

- **Positive Testing**: Tests valid inputs.
  - *Example*: Entering correct username and password on a login form.
- **Negative Testing**: Tests invalid inputs.
  - *Example*: Leaving fields empty or entering special characters in the password field.

---

4. What do you understand by exploratory testing?

- Testing without predefined test cases to discover issues dynamically.
  - *Example*: Randomly navigating through an app to find crashes or inconsistencies.

---

5. How do you prioritize test cases?

- Based on:
  - Business Impact.
  - Critical Functionality.
  - Risk of Failure.

---

6. What steps do you follow if you find a defect?

1. Reproduce the defect.
2. Document it with screenshots, logs, and steps.
3. Classify severity and priority.
4. Report it in a defect tracking tool (e.g., JIRA).

Real-time Scenarios

1. How would you test a login page?

- **Positive Tests**:
  - Valid username and password.
  - Remember Me functionality.
- **Negative Tests**:
  - Empty fields.
  - Invalid credentials.
  - SQL injection attacks.

---

2. If you are given incomplete requirements, how would you proceed with testing?

1. Use exploratory testing.
2. Collaborate with stakeholders.
3. Document assumptions and get them validated.

Documentation and Tools

1. What is the Requirements Traceability Matrix (RTM)?

- **Definition**: A document that maps requirements to test cases.
  - Ensures all requirements are tested.

Example:

Requirement ID	Requirement Description	Test Case ID

R001	Login with valid credentials	TC001

---

2. Which test management tools have you used?

- Examples: TestRail, Zephyr, HP ALM.

Soft Skills and Best Practices

1. How do you handle conflicts with developers over a defect?

- Provide evidence (logs, screenshots).
- Use data to justify your argument.

2. How do you manage tight deadlines?

- Prioritize high-risk areas.
- Communicate challenges early.

3. What steps do you take to improve your testing skills?

- Follow industry trends.
- Practice on real-world projects.
- Learn new tools (e.g., Selenium, JMeter).