**1. Software Testing Life Cycle (STLC)**

The **Software Testing Life Cycle (STLC)** is a systematic process that defines all the testing activities required to ensure software quality.  
It runs parallel to the Software Development Life Cycle (SDLC) but focuses entirely on **verification and validation**.

**Phases of STLC**

1. **Requirement Analysis**
   * Understand the requirements from BRD/SRS.
   * Identify what is testable and what is not.
   * Detect ambiguities or missing requirements.
   * **Output:** Requirement Traceability Matrix (RTM).
2. **Test Planning**
   * Define scope, objectives, resources, schedule, and risk mitigation.
   * Select tools and assign responsibilities.
   * Estimate effort and cost.
   * **Output:** Test Plan Document.
3. **Test Case Development**
   * Write detailed test cases for functional and non-functional testing.
   * Develop automation scripts if feasible.
   * Prepare test data (valid, invalid, boundary cases).
   * Review and optimize test cases.
   * **Output:** Test Cases, Test Scripts, Test Data.
4. **Test Environment Setup**
   * Set up hardware, software, network, and test servers.
   * Prepare environments similar to production.
   * Use stubs/drivers if needed.
   * **Output:** Test Environment Ready.
5. **Test Execution**
   * Execute test cases manually or via automation.
   * Compare actual vs expected results.
   * Log defects in defect tracking tools (JIRA, Bugzilla).
   * Perform regression testing for bug fixes.
   * **Output:** Executed Test Cases, Defect Reports.
6. **Test Cycle Closure**
   * Check whether all planned tests are executed.
   * Collect metrics: defect density, test coverage, pass/fail rate.
   * Prepare Test Summary Report.
   * Document lessons learned and process improvements.
   * **Output:** Test Closure Report, Metrics, Sign-off.

**2. Types / Levels of Testing**

Testing is performed at multiple levels to ensure the software works correctly at all layers.

**2.1 Unit Testing**

* Tests individual components or functions.
* Usually automated and performed by developers.
* Example: A function in a banking app that calculates interest.

**2.2 Integration Testing**

* Ensures modules work together correctly.
* Types: Top-Down, Bottom-Up, Big Bang, Hybrid.
* Example: Testing fund transfer where account and transaction modules interact.

**2.3 System Testing**

* Validates the complete integrated application against requirements.
* Performed in an environment similar to production.
* Example: End-to-end testing of login → transfer → logout in a banking app.

**2.4 Acceptance Testing**

* Determines if the system meets business requirements.
* Types: Alpha (internal team), Beta (end-users).
* Example: Customers test the banking app before release.

**2.5 Regression Testing**

* Ensures that new changes do not break existing functionality.
* Often automated with regression suites.
* Example: After fixing a bug in fund transfer, check login and account balance still work.

**3. Test Planning and Strategy**

**Test Planning** defines how testing will be performed, and **Test Strategy** defines the approach and objectives.

**Key Elements:**

* Objectives and scope of testing.
* Resource allocation: team, tools, environments.
* Time, effort, and cost estimation.
* Risk analysis and mitigation.
* **Output:** Test Plan Document.

**4. Test Automation Tools and Frameworks**

Automation reduces manual effort, improves accuracy, and increases coverage.

**Popular Tools:**

* Selenium – Web application automation.
* JUnit/TestNG – Unit testing in Java.
* Pytest – Python testing framework.
* JMeter – Performance/load testing.
* Appium – Mobile automation.

**Types of Automation Frameworks:**

* Linear (Record & Playback)
* Modular Framework
* Data-Driven Framework
* Keyword-Driven Framework
* Hybrid Framework

**5. Bug Life Cycle and Management**

The **Bug Life Cycle** (Defect Life Cycle) tracks the status of a defect from discovery to closure.

**Typical Stages:**

1. New – Defect reported.
2. Assigned – Assigned to developer.
3. Open – Developer starts working.
4. Fixed – Developer fixes defect.
5. Retest – Tester re-tests.
6. Verified – Tester confirms fix.
7. Reopen – If issue persists.
8. Closed – Issue resolved.
9. Deferred / Rejected – Not a defect or postponed.

**Bug Management Tools:**

* JIRA
* Bugzilla
* Mantis
* HP ALM / Quality Center