Software Testing

---

1. Overview of Software Testing

What is Software Testing?

Software Testing is the process of evaluating a software application to detect differences between given input and expected output. It ensures that the software system meets requirements and is **bug-free**, reliable, and performs well.

Goal: To identify errors, gaps, or missing requirements.

Why Testing is Important?

- Ensures **software quality**.
- Prevents **costly failures** post-release.
- Maintains **customer trust and satisfaction**.
- Ensures **compliance** with standards and regulations.

Software Testing Lifecycle (STLC):

1. **Requirement Analysis**
2. **Test Planning**
3. **Test Case Design**
4. **Test Environment Setup**
5. **Test Execution**
6. **Defect Reporting & Tracking**
7. **Test Closure**

Live Scenario Example:

A **banking application** needs to transfer funds. Testing ensures:

- Valid accounts transfer correctly.
- Invalid accounts raise errors.
- Correct amount is deducted/credited.
- No crash during high usage.

---

2. Levels of Software Testing

There are **4 main levels** of testing:

a. Unit Testing

- Tests **individual components** or functions.
- Done by **developers**.
- Tools: JUnit, NUnit, pytest.

Example: Test a login() function with correct/incorrect credentials.

---

b. Integration Testing

- Tests **interactions between modules**.
- Done by developers/testers.
- Types: Top-down, Bottom-up, Big Bang.

Example: Check if payment module integrates correctly with user and order module.

---

c. System Testing

- Tests the **entire system as a whole**.
- Done by QA team.
- Includes functional & non-functional testing.

Example: Test an e-commerce platform from product search to checkout.

---

d. Acceptance Testing

- Validates system meets **business requirements**.
- Done by **client or end users**.
- Types: Alpha, Beta testing.

Example: Client tests final version of school management app before going live.

---

3. Software Testing Techniques

a. Black Box Testing

- Focuses on input/output without knowing internal code.
- Tester checks functionality.

Example: Enter values into a form and verify correct submission.

---

b. White Box Testing

- Involves **internal logic** and structure of code.
- Done by developers.

Example: Check loops, conditions, and logic paths in `calculateTax()`.

---

c. Grey Box Testing

- Combination of black & white box.
- Tester has partial knowledge of code.

Example: Test APIs with some knowledge of database schema.

---

d. Static Testing

- Review or inspection without executing code.
- Early stage testing.

Example: Peer review of code or requirements document.

---

e. Dynamic Testing

- Testing involves execution of the code.

Example: Test the working of a registration form in runtime.

---

4. Types of Testing

Functional Testing:

- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing (UAT)
- Regression Testing
- Smoke & Sanity Testing

Non-Functional Testing:

- Performance Testing (Load, Stress)
- Security Testing
- Usability Testing
- Compatibility Testing
- Scalability Testing
- Accessibility Testing

Live Scenarios for Testing Types:

Type	Scenario
Unit	Test if `applyCoupon()` gives correct discount.
Integration	Check if order placed reflects in invoice.
System	Test entire ride-booking process in Uber app.
UAT	Client verifies leave management system features.
Regression	Re-test login after password module update.
Performance	Check response time under 1000 users.
Security	Check SQL injection on login form.
Usability	Are forms easy to use for non-technical users?

---

5. Test Planning and Design

What is Test Planning?

A **test plan** outlines:

- Scope of testing
- Objectives
- Resources (people, tools)
- Schedule
- Deliverables

Components of Test Plan:

1. **Test Objectives**
2. **Testing Scope (In-Scope/Out-of-Scope)**
3. **Test Strategy**
4. **Resource Planning**
5. **Risk Management**
6. **Entry/Exit Criteria**
7. **Deliverables**

Test Design:

- Define **test scenarios**
- Write **test cases**
- Prepare **test data**
- Define **expected results**

Example: Test Case Template

Test Case ID	Description	Steps	Expected Result	Status
TC001	Login Functionality	1. Open Login Page 2. Enter valid credentials 3. Click Login	User is redirected to dashboard	Pass

---

6. Test Execution

Steps Involved:

1. Execute test cases.
2. Compare **actual vs expected** results.
3. Log pass/fail status.
4. Report any **defects**.
5. Re-execute after fixes (retesting).
6. Perform **regression testing**.

Example:

Scenario: Student Registration Page

- Input valid data: ✔ Pass
- Input invalid email: Fail → Bug ID: BUG_123
- Input blank form: ✔ Error message shown → Pass

---

7. Defect Management

What is a Defect?

A **defect** is a variance between expected and actual result. Also known as **bug, error, or issue**.

Defect Life Cycle:

1. **New** → 2. **Assigned** → 3. **Open** → 4. **Fixed** → 5. **Retest** → 6. **Closed** OR **Reopen**

Defect Report Format:

Field	Description
Defect ID	Unique ID
Summary	Short title
Description	Detailed steps to reproduce
Severity	Critical, Major, Minor
Priority	High, Medium, Low
Status	New, Open, Fixed, Closed
Assigned To	Developer name
Attachments	Screenshots/logs

Example Defect:

- **ID**: BUG_2025
- **Title**: Amount field accepts alphabets
- **Steps**: Go to payment → Enter "abc" in amount
- **Expected**: Only numbers allowed
- **Actual**: Form submitted
- **Severity**: Major
- **Priority**: High
- **Assigned to**: Dev001

---

8. Tools Used in Software Testing

Category	Tools
Test Management	TestRail, Zephyr
Bug Tracking	Jira, Bugzilla, Mantis
Automation	Selenium, Cypress, Playwright
Performance	JMeter, LoadRunner
CI/CD	Jenkins, GitHub Actions

---

9. Conclusion & Best Practices

- Understand business goals clearly.
- Write **clear, reusable test cases**.
- Log detailed defects with evidence.
- Automate where repeatable.
- Always do **regression testing** after changes.
- Communicate with developers efficiently.

Live Scenario:

Project: Online Learning Platform

- **Login Module**: Unit Tested
- **Course Enroll**: Integration Tested
- **Video Streaming & Quizzes**: System Tested
- **Client Reviews Final UAT**
- All test cases passed. 3 minor bugs fixed during retest.