# Software Testing & Quality Assurance – Detailed Syllabus with Explanations

## Unit 1: Introduction to Software Testing and Quality Assurance

### *Verification vs Validation*

Verification ensures the product is built correctly (process■based), while validation ensures the right product is built (output■based).

### *Importance of Software Testing in SDLC*

Testing ensures reliability, performance, security, and usability. It helps find defects early and prevents failures.

### *Overview of Software Quality Assurance (SQA)*

SQA focuses on maintaining standards, audits, reviews, and ensuring processes follow quality guidelines.

### *Software Testing Principles*

Examples: Testing shows presence of defects, exhaustive testing is impossible, defects cluster together, pesticide paradox.

### *Myths & Misconceptions*

E.g., ‘Testing guarantees no defects’, ‘Only testers are responsible for quality’.

### *Types of Software Testing: Manual vs Automated*

Manual testing involves human execution; automation uses tools/scripts to speed up repetitive tasks.

### *Defect Life Cycle*

Stages: New → Assigned → Open → Fixed → Retest → Verified → Closed.

### *Test Documentation*

Includes Test Plans, Test Cases, Traceability Matrix, Bug Reports.

# Unit 2: Software Testing Life Cycle (STLC) & Testing Techniques

### *STLC Phases*

Requirement analysis, Test planning, Test design, Environment setup, Test execution, Test closure.

### *Levels of Testing*

Unit testing, Integration testing, System testing, Acceptance testing.

### *Functional Testing*

Tests system as per functional requirements such as boundary value, equivalence partitioning, decision tables, BDD.

### *Non■Functional Testing*

Performance, usability, reliability, compatibility, scalability.

### *Black Box Testing Techniques*

Equivalence partitioning, boundary value analysis, state transition testing.

### *White Box Testing Techniques*

Statement coverage, branch coverage, path testing.

### *Behavior■Driven Development (BDD)*

Using Gherkin syntax: Given–When–Then to define test scenarios.

# Unit 3: Test Automation and Performance Testing

### *Benefits & Challenges of Automation*

Benefits: Faster execution, reusability, accuracy. Challenges: High initial cost, tool expertise needed.

### *Automation Tools: Selenium, JUnit, TestNG*

Tools used for scripting, running tests, generating reports.

### *Writing Automated Test Scripts*

Includes selecting locators, creating reusable functions, assertions, test suites.

### *CI/CD in Testing*

Continuous Integration automates code building/testing; Continuous Deployment automates release pipelines.

### *Performance Testing Concepts*

Load testing, stress testing, spike testing, endurance testing.

### *Performance Tools*

JMeter (open source), LoadRunner (enterprise), Gatling (DevOps■friendly).

# Unit 4: DevOps, CI/CD, AI in Testing, Security

### *Testing in DevOps*

Continuous testing & integration ensures faster delivery with quality.

### *Docker & Kubernetes in Testing*

Used to create isolated test environments & scale test executions.

### *AI & ML in Testing*

Predictive analytics, defect prediction, self■healing locators, intelligent test case generation.

### *Self■Healing Automation*

Uses AI to update broken test locators automatically.

### *Security Testing*

Includes checking for vulnerabilities, data leaks, insecure authentication mechanisms.

### *Ethical Hacking*

White■hat techniques used to find system vulnerabilities.

### *Penetration Testing*

Simulated attack to detect exploit weaknesses.

### *Vulnerability Assessment*

Systematic identification of risks using scanning tools.

# Unit 5: Case Studies & Capstone

### *Real■World Case Studies*

Applying testing methodologies on real software—web apps, mobile apps, enterprise apps.

### *Designing a Test Plan*

Identify objectives, scope, strategy, environment needs, risks.

### *Developing Test Cases & Scenarios*

Derive structured test cases aligned with requirements.

### *Executing Test Plans*

Perform testing, log defects, track metrics.

### *Optimizing Test Strategies*

Use automation, risk■based testing, prioritization techniques.

### *Presentation & Documentation*

Create reports, dashboards, defect analysis summaries.