Software Testing - Concise Answers

Q1: Explain the differences between System Testing and Integration Testing.

System Testing tests the entire system as a whole, ensuring it meets requirements. Integration Testing checks data flow and communication between modules. Example: System testing a banking app's full workflow; integration testing login and transaction modules together.

Q2: Describe the TopDown and BottomUp integration testing approaches.

TopDown tests top modules first, using stubs for lower modules. BottomUp starts with lower modules, using drivers. TopDown finds UI issues early; BottomUp finds utility bugs early.

Q3: What is Scenario Testing? How does it differ from traditional functional testing?

Scenario Testing involves end-to-end real-world situations. Functional testing tests isolated features. Example: Booking a flight from search to payment (scenario), vs. testing only the payment module (functional).

Q4: Define NonFunctional Testing. Discuss its importance with examples.

NonFunctional Testing checks system attributes like performance, usability, and scalability. Examples: Stress Testing (extreme load), Scalability Testing (growing users). It's vital for system reliability and user experience.

Q5: Explain the concept of Acceptance Testing.

Acceptance Testing ensures software meets business requirements. Acceptance criteria include expected results, conditions of satisfaction, and constraints. These guide test case design to validate customer needs.

Q6: Discuss the role of Beta Testing.

Beta Testing is performed by real users before release. It provides feedback on usability, bugs, and performance in real environments. Benefits include reduced risk, improved UX, and customer involvement.

Q7: What is Deployment Testing?

Deployment Testing verifies software runs correctly in the production environment. Activities: install checks, environment validation, rollback procedures. It ensures smooth, error-free deployment.

Software Testing - Concise Answers

Q8: Compare Functional Testing vs Design/Architecture Verification.

Functional Testing validates specific features. Design Verification checks system structure and interactions. Both are essential: one ensures correct behavior; the other ensures robust structure.

Q9: Explain Bidirectional Integration Testing.

It combines TopDown and BottomUp. Middle modules are tested with both stubs and drivers. Advantage: detects issues in both upper and lower layers concurrently.

Q10: Discuss test case selection in Acceptance Testing.

Selecting relevant test cases ensures user requirements are validated. Effective selection improves software quality, reduces risk, and ensures customer satisfaction.

Software Testing - Concise Answers

UNIT 5 - Concise Answers

Q1: Explain the Regression Testing Process.

Regression Testing verifies new changes don't break existing features. Steps: identify test cases, prioritize, execute, compare results, log defects, report status.

Q2: Discuss Execution Trace in regression testing.

Execution Trace logs code paths during testing. It helps identify affected areas after changes, aiding in selecting effective regression tests.

Q3: Define Dynamic Slicing in regression testing.

Dynamic Slicing isolates code impacting a specific output. It improves regression test effectiveness by focusing on affected code segments.

Q4: Describe tools for regression testing and selection factors.

Tools: Selenium, JUnit, TestNG. Factors: compatibility, language support, reporting, CI integration, ease of use, cost, and support.

Q5: What is Ad hoc Testing?

Informal testing without predefined cases. Focuses on intuition and experience. Useful for quick checks but lacks repeatability of formal methods.

Q6: Explain Pair Testing.

Two testers collaborate on the same system-one tests, one observes. Enhances idea sharing, finds more bugs, and improves test quality.

Q7: Describe Exploratory Testing.

Testers learn, design, and execute simultaneously. Complements scripted testing. Useful in early stages or unstable systems.

Software Testing - Concise Answers

Q8: What is Iterative Testing?

Performed in cycles after each development phase. Helps catch defects early and aligns with Agile. Unlike traditional testing, it's continuous.

Q9: Explain Defect Seeding.

Intentionally inserting bugs to evaluate testing effectiveness. Helps measure defect detection rate and improve QA processes.

Q10: Key components of Test Planning in Automation Testing.

Define scope, objectives, tools, environment, schedule, and resources. Ensures clear strategy and smooth execution.

Q11: Describe Test Execution in Automation Testing.

Running automated test scripts. Challenges: flaky tests, environment issues, synchronization errors, tool limitations.

Q12: Importance of Reporting in Automation Testing.

Provides visibility, tracks progress, aids decision-making. Metrics: test status, execution time, pass/fail rate, defect linkage.

Q13: Scope of automation and suitable test types.

Best for regression, data-driven, performance, and smoke tests. These are repetitive and benefit from speed and accuracy.

Q14: Design considerations for automation frameworks.

Include modularity, reusability, data handling, integration, logging. Good design ensures maintainability and scalability.

Q15: Generic requirements for a test tool framework.

Should support OOP, object recognition, integration, scripting, and cross-platform. Affects tool choice for object-oriented systems.