Advanced Software Testing Questions

# Introduction to Software Testing

* Q: If a system passes all test cases but still fails in the real world, what might have gone wrong in the testing process?

A: Tests were not aligned with real user behavior; lacked exploratory or negative scenario testing.

* Q: Can a software be considered 100% defect-free after testing? Why or why not?

A: No, exhaustive testing is impossible. Testing only reduces risk, it doesn't eliminate it.

# Principles of Software Testing

* Q: Why is the principle 'Absence of errors is a fallacy' important when delivering software to the client?

A: Because fixing all bugs doesn’t guarantee the software meets user needs or is usable.

* Q: How does early testing reduce cost, and what principle supports this idea?

A: It follows the 'Early Testing' principle — catching defects early prevents expensive late-stage fixes.

# Quality, QA, QC

* Q: If two teams test the same application but one does QA while the other does QC, how might their responsibilities differ?

A: QA is process-oriented and focuses on preventing defects; QC is product-oriented and detects defects.

* Q: A company has excellent QA processes but still releases buggy software. What could be the issue?

A: QA is about setting the right processes, but poor implementation or lack of QC can still allow bugs.

# Five Perspectives of Quality

* Q: How would the 'value-based' view of quality differ from the 'transcendental' view in a real-world scenario?

A: Value-based: judged by customer needs; Transcendental: an inherent excellence, hard to define.

* Q: If a software looks good, performs well but doesn’t meet user requirements, which quality perspective failed?

A: User-based or Product-based may be good, but Value-based has failed.

# SDLC and SDLC Models

* Q: How would choosing the wrong SDLC model affect project outcomes, even if testing is done thoroughly?

A: A wrong model can lead to unclear requirements, scope creep, or delays.

* Q: Why might Agile be unsuitable for certain critical or regulatory projects?

A: Agile is flexible, but less documentation and fast-paced changes may not suit high-risk domains.

# STLC (Software Testing Life Cycle)

* Q: What happens if the 'test planning' phase is skipped in STLC?

A: Leads to poor resource allocation, unclear scope, delays, and unmeasurable test success.

* Q: Why is traceability matrix created during STLC and not SDLC?

A: To ensure all requirements are covered by test cases — it's a testing responsibility.

# Levels of Testing

* Q: Which testing level would likely catch a bug in data transfer between modules, and why?

A: Integration testing — verifies interfaces and data flow between components.

* Q: How is system testing different from acceptance testing even if both test the whole application?

A: System testing: internal QA team; Acceptance testing: end-users or clients.

# Black, White, Grey Box Testing

* Q: Which testing technique would be most suitable to test a banking app's internal logic, and why?

A: White-box — gives visibility into internal logic like calculations, branching.

* Q: If you are given no documentation or code but only the app, which testing method would you apply and what are the limitations?

A: Black-box testing — test UI, inputs/outputs, but internal bugs may go unnoticed.

# Test Cases

* Q: If two testers write different test cases for the same feature, does it mean one is wrong?

A: Not necessarily — test cases can vary based on perspective, creativity, and test goals.

* Q: Why might writing too many detailed test cases become counterproductive?

A: Hard to maintain, time-consuming to execute, and may reduce flexibility for exploratory testing.