ISTQB Remaining Topics

1. Non-Functional Testing

Non-functional testing focuses on **how well** the system performs rather than **what it does**. It checks quality attributes of a system.

Types of Non-Functional Testing

- **Performance Testing** – Ensures the system meets response time, throughput, and scalability requirements.
  Example: Website should load within 3 seconds for 1000 users.
- **Load Testing** – Evaluates system behavior under expected workload. *Example:* Testing e-commerce site on peak shopping days.
- **Stress Testing** – Pushes the system beyond normal limits to find breaking points. *Example:* Increase user load until server crashes.
- **Scalability Testing** – Determines whether the system can handle growth (more users/data).
- **Security Testing** – Ensures protection against unauthorized access and vulnerabilities. *Example:* Penetration testing, SQL injection testing.
- **Usability Testing** – Evaluates user-friendliness and accessibility.
- **Reliability Testing** – Measures stability over time. Uses metrics like MTBF (Mean Time Between Failures).
- **Portability Testing** – Checks whether the system runs on different platforms, OS, or devices.
- **Maintainability Testing** – Evaluates how easily code can be modified, fixed, or upgraded.
- **Compliance Testing** – Ensures software follows industry standards, regulations, or laws (e.g., GDPR).

---

2. Test Management

Test management ensures testing activities are properly organized and controlled.

Test Organization

- **Roles in Testing:**
  - Test Manager: Responsible for planning, monitoring, and controlling testing.
  - Tester: Designs, executes, and reports tests.
  - Developer: Fixes defects and supports unit testing.
  - Business Analyst: Provides requirement clarification.
- **Levels of Independence:**
  - Developers testing their own code (low independence).
  - Independent test team within project (medium independence).

    - External independent test team or certification body (high independence).

Test Planning

Defines what, how, who, and when testing will be done.
Key elements include:

- Objectives
- Scope
- Test approach
- Resources and environment
- Entry and exit criteria
- Schedule and milestones

Monitoring and Control

- **Monitoring:** Collecting test metrics such as % executed, % passed, defects found, test duration.
- **Control:** Adjusting the plan when actual results deviate from expectations.

Configuration Management

Ensures test artifacts like plans, cases, data, and environments are version controlled. Tools like Git or Jira can be used.

Risk-Based Testing

Focuses testing effort on high-risk areas.
Formula: **Risk = Likelihood × Impact**
Example: Payment module is high risk → tested more thoroughly than low-risk features.

Incident / Defect Management

- **Defect Lifecycle:** New → Open → Fixed → Retest → Closed (or Reopen).
- **Defect Report typically includes:**
    - ID and summary
    - Severity and priority
    - Steps to reproduce
    - Expected vs actual results
    - Environment information
    - Status and history

---

3. Test Tools

Test tools help automate, manage, or support parts of the testing process.

Categories of Test Tools

1. **Test Management Tools**
   - Manage test cases, execution, reporting.
   - Examples: TestRail, TestLink, Zephyr.
2. **Defect Tracking Tools**
   - Log and manage bugs.
   - Examples: Jira, Bugzilla, MantisBT.
3. **Static Analysis Tools**
   - Analyze code without executing.
   - Examples: SonarQube, ESLint.
4. **Test Execution Tools**
   - Run automated scripts.
   - Examples: Selenium, Cypress, JUnit.
5. **Coverage Tools**
   - Measure code coverage (statements, branches).
   - Examples: JaCoCo, Cobertura.
6. **Performance Testing Tools**
   - Load & stress testing.
   - Examples: JMeter, LoadRunner.
7. **CI/CD Tools** (support testing automation in pipelines)
   - Examples: Jenkins, GitHub Actions, GitLab CI.

Benefits of Test Tools

- Faster and consistent execution
- Reusable test assets
- Improved coverage
- Better reporting and traceability
- Efficient for regression testing

Risks of Test Tools

- High cost (tools, training, maintenance)
- Steep learning curve
- Over-reliance on tools (manual testing still needed)
- Tools may not fit every context
- Risk of false confidence if misused