Qualitative or Quantitative Testing Diploma Project Requirements

Scope and Goals

The goal of this document is to provide students with a clear overview of the expectations and evaluation criteria for diploma projects focused on Qualitative or Quantitative Software Testing.

The document defines:

- Minimum requirements – mandatory criteria to obtain a passing grade (grade = 5).
- Maximum requirements – advanced criteria to achieve the highest grade (grade = 10).
- Major errors and shortcomings – typical critical issues resulting in penalties.

Minimum Requirements (for a passing grade)

1. The project must demonstrate the use of a structured testing workflow. A clear definition of whether the project focuses on qualitative testing, quantitative testing, or a combination of both has to be included.
2. There must be a clearly defined test scope, testing goals, objectives, and entry/exit criteria. These elements must be justified with respect to system complexity and associated risks.
3. The project must contain a requirements analysis or feature breakdown identifying testable functionality, user flows, or system components. This analysis has to serve as the basis for the testing process.
4. A set of manual test cases, checklists, or scenarios must be present. Each artifact must include preconditions, steps, expected results, and prioritization (e.g., Smoke, Critical, etc.; High, Medium, etc.)
5. For quantitatively oriented projects, several key testing metrics (e.g., pass rate, defect density, defect leakage, execution time, coverage metrics) must be defined and measured.
6. For qualitatively oriented projects, documented findings on usability issues, UX problems, unclear flows, or inconsistencies must be included.
7. The project must provide a structured test execution report with identified defects, environment details, retesting results, and references to supporting artifacts.
8. At least one professional testing tool (e.g., Postman, JMeter, TestLink, Jira, Azure DevOps, etc.) must be used, and its configuration and purpose must be documented. Tool usage must be relevant to the tested system and contribute to the overall testing workflow.
9. The entire project must be stored in a Git repository with meaningful commit history, along with documentation that must allow full reproduction of the environment and test execution process.
10. A concise technical report must be prepared, summarizing the tested system, testing strategy, executed work, collected metrics (if applicable), and final conclusions.

Maximum Requirements (for the highest grade)

1. A comprehensive testing strategy combining functional, non-functional, usability, exploratory, or regression testing must be included. The chosen approach must be justified using risk analysis or project constraints.
2. Advanced test design techniques (equivalence partitioning, boundary value analysis, decision tables, state transition diagrams, cause-effect graphs) must be applied, with a demonstration of how they improved coverage or reduced redundancy.
3. Test prioritization or risk-based testing must be used, with clearly presented reasoning and measurable impact.
4. For quantitative testing:
    a. Testing metrics (e.g., traceability coverage, trends, defect distribution, performance indicators) must be collected, visualized, and interpreted.
    b. Metric collection should be automated through scripts or built-in reporting tools.
    c. Comparisons between multiple test cycles with data-driven conclusions must be included.
5. For qualitative testing:
    a. Structured exploratory testing sessions with session charters, timelines, and documented findings must be conducted and included.
    b. Heuristic evaluation methods or cognitive walkthroughs must be applied.
    c. UX issues must be supported with appropriate evidence (screenshots, annotations, guideline comparisons).
6. Where applicable, automated checks (UI, API, load, static analysis) must be present, even if only partially implemented, with proper architecture (e.g., page-object pattern, reusable components).
7. CI/CD pipelines must be used to automatically execute tests and produce logs, reports, and other artifacts.
8. Advanced visualizations (charts, defect trends, coverage graphs, performance curves) must be included, with a consolidated dashboard or structured summary where appropriate.
9. The project must contain comparisons of different testing methods, tools, or approaches and must provide justified conclusions along with recommendations for future improvements.

Major Errors and Shortcomings Affecting Evaluation

Category	Example of issue	Penalty
Test design	Missing test cases, unclear steps, inconsistent expected results	-1 point
Documentation	Missing reports or coverage data, incomplete execution results	-1 point
Functional logic	Tests not aligned with system behavior or requirements	-2 points

Tools & methodology	Incorrect or outdated tool usage	-1 point
Analysis quality	Missing interpretation or conclusions	-1 point
Testing completeness	Significant system areas not tested without justification	-2 points
Code quality (automation)	Non-modular, fragile, or poorly structured automation	-1 point
Evidence	Missing screenshots, logs, metrics, or unverifiable results	-1 point