Testing Best Practices

IGNITE BRILLIANCE

Contents

Test Best Practices & QA Governance Guide

A Comprehensive Framework Covering Manual, Automation, Performance, Integration & System Testing

1. Purpose of this Document

This guide establishes unified test practices across all stages of software quality assurance—manual testing, automation, performance, system, and integration testing. It aligns with **CMMI**, **ISO/IEC 27001**, **12207**, and other quality frameworks, while remaining practically grounded for teams using tools like **Selenium**, **Cucumber**, **Rest Assured**, **Appium**, and **JMeter**.

---

2. Who Should Use This

This document is intended for:

- QA Engineers (Manual & Automation)
- Test Leads and Managers
- Performance Engineers
- DevOps/SRE Teams
- Developers responsible for writing/maintaining tests
- Security and Compliance Auditors

---

3. Manual Testing Practices

Approach:

- Test cases are derived from SRS, user stories, and acceptance criteria.
- Use checklists and exploratory testing for edge case validation.

Best Practices:

- Use traceability matrix (RTM) to ensure coverage.
- Maintain reusable and modular test cases.
- Follow structured bug reporting using tools like Jira.
- Conduct test case reviews as part of QA process.

Artifacts:

- Test Plan, Test Cases (Excel or TestRail), RTM, Defect Log, Daily Status Reports.

---

4. Automation Testing Practices

Approach:
Use a hybrid framework built with:

- **Selenium + Java + Maven + Cucumber (BDD)**
- Modular folder structure (Base, Utils, Pages, Steps, Hooks, Enums, etc.)
- Screenshots and logs integrated with **Extent Reports**

Best Practices:

- Follow BDD style for clear, business-readable scenarios.
- Reuse page object elements across multiple tests.
- Parameterize test data and use environment-specific config files.
- Include screenshot and log per step.
- Integrate into CI/CD (Bitbucket Pipelines, GitHub Actions).
- Tag-based execution for smoke, regression, sanity.

Artifacts:

- Feature files (.feature), Step Definitions, Runner class, Extent Reports, Logs, Screenshots, CI Pipeline logs.

---

5. API Testing Practices

Approach:

- Framework built using **Rest Assured + Java + Cucumber + Maven**.
- Organized layers: Base, Config, Endpoints, Payload, StepDefinitions, Hooks.

Best Practices:

- Use POJO models and reusable request specs.
- Validate response codes, schema, headers, and values.

- Include negative test cases (auth failures, invalid input).
- Integrate with CI tools for nightly API regression.

Artifacts:

- Swagger/OpenAPI specs, JSON schema validators, Feature files, Test data files, Reports.

---

6. Mobile Testing Practices

Approach:

- Automation with **Appium + Java + Cucumber**, using real devices and emulators.
- Supports both Android and iOS apps.

Best Practices:

- Use Appium Inspector to locate elements accurately.
- Isolate environment-specific settings (APK/IPA, credentials).
- Enable screenshot logging per step.
- Test across screen resolutions, OS versions, and devices.

Artifacts:

- Appium config files, Device capabilities JSON, Feature files, Execution logs, Screenshot report.

---

7. Performance Testing Practices

Approach:

- Use **Apache JMeter** for performance and load testing.

Best Practices:

- Simulate realistic user loads for APIs and UI flows.
- Test during staging, pre-prod, and post-release monitoring.
- Parameterize input, simulate ramp-up, think time, and varied load patterns.
- Run with assertions for response time SLAs.

- Integrate JMeter with CI/CD for regular load tests.

Artifacts:

- JMX test plans, HTML reports, CSV output logs, CI results archive.

---

8. System Testing (Security-Centric)

Approach: System testing at Array-World is led by penetration testing teams, validating real-world threat scenarios.

Steps:

1. Conduct penetration testing for auth, input validation, session, and APIs.
2. Document vulnerabilities with severity and PoC.
3. Log defects in Jira and assign for remediation.
4. Verify resolved issues and iterate till all are closed.
5. Final approval by Project Manager post remediation of high/critical issues.

Artifacts:

- Vulnerability report (mobile & web)
- Jira tickets with traceability
- Pull requests and secure coding notes
- Remediation checklist
- Testing sign-off form

---

9. Integration Testing

Approach:

- Performed post-unit testing, focusing on validating interfaces between systems, modules, services.

Best Practices:

- Write integration tests as part of test suites using APIs and UI flows.
- Use test doubles/stubs for downstream dependencies.

- Log all interface validation and error conditions.
- Automate critical integration flows using Rest Assured and Selenium.

Artifacts:

- Integration Test Plan, Interface Control Documents (ICDs), Test logs, Service mocks.

---

10. Governance & Compliance

Aligned with: CMMI, ISO/IEC 27001, 12207
Controls:

- Version control for all test artifacts.
- Test review and sign-off checkpoints.
- Defect lifecycle documentation.
- Access-controlled environments.
- Retain reports, logs, and evidence for audit readiness.

---

11. CI/CD and Reporting

CI/CD Tools: Bitbucket Pipelines, GitHub Actions
Automation Inclusions:

- Maven test triggers
- JUnit/TestNG & Extent Report generation
- JMeter CLI execution
- Slack/Email alerts for test outcomes

Reports:

- Execution summary
- Screenshots per step
- HTML/Cucumber/Extent Reports
- Performance dashboards

12. Review & Continuous Improvement

- Monthly QA retrospectives and feedback sessions
- Automation code reviews
- Static code analysis using SonarQube
- Performance test trends reviewed quarterly
- Compliance readiness check every 6 months