Coding Standards for Automation Projects

# Executive Summary

This document defines coding standards and best practices for automation projects.  
It ensures code is readable, maintainable, reusable, and collaborative.  
Helps reduce flaky tests, improve quality, and make automation a first-class citizen in development.

Table of Contents will be auto-generated in Confluence based on headings.

# Introduction

* These coding standards apply to all automation projects.
* Ensures consistent structure, maintainability, and long-term value from test automation.

# 1. Naming Conventions

* Classes: PascalCase → LoginPageTest
* Methods: camelCase with action verbs → calculateTotal()
* Variables: camelCase → totalAmount
* Constants: UPPER\_CASE → MAX\_LIMIT
* Packages: lowercase only → com.project.automation
* Test methods: descriptive → shouldLoginWithValidCredentials()

# 2. Comments & Documentation

* Write Javadoc for classes, methods, complex logic
* Inline comments only for non-obvious code
* Avoid redundant comments
* Comment why, not just what

# 3. Error Handling

* Always use try-catch with proper logging
* Avoid generic catch blocks
* Never suppress exceptions silently
* Provide meaningful error messages

# 4. Code Structure

* One class per file; name must match class
* Modular folder/package structure (tests, utilities, pages, data, config)
* Avoid magic numbers/strings → use constants/enums
* Keep methods short (≤50 lines)

# 5. Constants & Configurations

* Use dedicated Constants/config files
* Never hardcode values; use configs or environment variables
* Maintain environment-specific configurations (dev, QA, staging, prod)

# 6. Collections & Generics

* Use correct collection types (List, Set, Map)
* Always use generics → List<String>
* Prefer interfaces: List<String> list = new ArrayList<>()

# 7. Functions & Methods

* Follow Single Responsibility Principle
* Keep parameter lists short
* Use DTOs or method overloading instead of long parameter lists
* Avoid global/static variables; prefer dependency injection

# 8. Code Optimization

* Avoid unnecessary object creation
* Use StringBuilder in loops
* Prefer enhanced for-loops and streams
* Lazy initialization for heavy objects

# 9. Logging Standards

* Use consistent logging framework (SLF4J/Log4j)
* Logging levels: INFO, DEBUG, ERROR
* Never log sensitive data

# 10. Test Automation Best Practices

* Tests must be independent and repeatable
* Follow Page Object Model (POM) or Screenplay pattern
* Avoid hard waits; use explicit waits
* Externalize test data (CSV, JSON, DB, DataProvider)
* Descriptive test names and grouping (@Smoke, @Regression)
* Ensure tests are idempotent
* Fix flaky tests promptly
* Use assertions with messages: assertEquals(actual, expected, 'Login failed')

# 11. Collaboration & Code Quality

* Use Git for version control: main, develop, feature/\*
* Always raise Pull Requests
* Enforce peer code reviews
* Use static analysis (SonarLint, Checkstyle, PMD)
* Integrate linting & static analysis in CI/CD
* Keep documentation updated

# 12. Test Reporting

* Use reporting tools (Allure, Extent)
* Reports should include: Test case name & ID, Execution time, Logs & screenshots on failure
* Store execution history for trend analysis

# 13. Continuous Improvement

* Review coding standards quarterly
* Team feedback sessions
* Adapt practices to new tools/frameworks
* Encourage knowledge sharing

# Common Problems in Test Automation

* Automation seen as less critical
* Poor coding practices → unreadable tests
* Skipped reviews/documentation
* Lack of ownership beyond QA
* Fragile/flaky tests (UI sync, data issues)
* Focus on quantity over quality
* Misunderstanding tools as 'magic solutions'

# How To Address These

* Raise awareness of test code quality impact
* Conduct training on automation design
* Enforce coding standards & reviews
* Allocate time for peer reviews in sprints
* Encourage cross-role ownership (Dev + QA + Product)
* Automate code checks in CI/CD
* Groom test backlog regularly
* Encourage continuous learning and upskilling