

Inazuma.co Software Development Standards Policy

Version: 1.0

Prepared By: Technology & Compliance Team

services, middleware, and backend systems developed or maintained by Inazuma.co.

- All environments including development, staging, and production.

1. Purpose

The Software Development Standards Policy ensures consistency, security, maintainability, and quality in software development practices at Inazuma.co. This policy sets the standards and best practices for software design, coding, documentation, version control, testing, and deployment. It also supports organizational goals by reducing technical debt, facilitating easier onboarding of new developers, and ensuring compliance with legal and contractual obligations.

2. Scope

This policy applies to:

- All internal and external software development projects.
- Full-time, part-time, contractual, freelance, and outsourced development teams.
- All platforms, mobile and web applications, internal tools, APIs,

3. Coding Standards

All developers must follow language-specific and general development standards to ensure code quality, performance, readability, and reusability.

3.1 Language-Specific Standards

Lan gua ge	Standards & Tools	Notes
Java Scri pt	Airbnb Style Guide, ESLint	Use TypeScript wherever possible
Pyth on	PEP8, Flake8, Black	Emphasize readability and testability
Java	Google Java Style Guide	Use JUnit for testing
HTM L/C SS	W3C Standards	Use responsive and accessible design patterns

SQL	ANSI SQL, SQLLint	Use parameterized queries to prevent SQLi
-----	-------------------	---

Branch	Purpose
main/master	Stable production code
develop	Integration of new features
feature/	Development of individual features
hotfix/	Emergency fixes to production
release/	Final preparation before pushing to main

3.2 General Guidelines

- No hardcoding of secrets or credentials.
- Avoid code duplication and large functions (maintain modularity).
- All functions must be commented and documented using appropriate docstring formats.
- Use consistent indentation (spaces over tabs).
- Variable and function names must follow camelCase or snake_case as per language conventions.

4.3 Commit Protocols

- Use descriptive and meaningful commit messages.
- Commit early and often but not unnecessary changes.

INAZUMA.CO
Transforming Brand Experiences

- Follow the Conventional Commits standard (e.g., `feat:`, `fix:`, `chore:`).

4. Version Control

4.1 Tooling

- Git is the mandatory VCS.
- All code must be hosted on Inazuma's GitHub/GitLab repository under protected branches.

4.2 Branching Strategy

5. Secure Development Practices

Security must be embedded throughout the software development lifecycle.

5.1 Security Principles

- Adopt the principle of least privilege.
- Secure third-party integrations with tokenized access.
- Perform threat modeling for all new applications.

5.2 Secure Coding Techniques

Area	Control Measures
Input Validation	Allow-list inputs, avoid <code>eval()</code> functions
Credential Management	Use Hashicorp Vault or AWS Secrets Manager
Data Protection	TLS/HTTPS, AES-256 encryption
Error Handling	Avoid exposing stack traces to users

6.2 Code Review Checklist

- Business logic validation
- Readability and documentation
- Compliance with security practices
- Testing coverage and results

6.3 Tools

- GitHub Pull Requests
- Code review platforms like Review Board or Gerrit (if applicable)

7. Documentation Standards

7.1 Mandatory Documents Per Project

- `README.md`: Setup, usage, and deployment instructions.
- `CONTRIBUTING.md`: Guidelines for external/internal contributors.
- Architecture Overview: High-level diagrams, component interactions.
- API Documentation: Swagger/OpenAPI with live sandbox

6. Code Review Process

6.1 Mandatory Reviews

- No code can be merged into the main branch without at least one peer review.
- High-risk modules (auth, payments, etc.) require reviews from senior or security team members.

links.

7.2 Inline Documentation

- Use consistent inline comments.
- Every method must have a docstring.
- Document edge cases and known limitations clearly.

8.1 Testing Metrics

- Code coverage should be maintained at >80%.
- Regression issues must not exceed 5% during releases.

9. Deployment Standards

8. Testing Requirements

Comprehensive testing is non-negotiable. Automated testing must be integrated within the CI/CD pipeline.

9.1 Continuous Integration/Continuous Deployment

- All projects must implement automated build/test/deploy pipelines.
- Use Docker containers where applicable.

Test Type	Tools	Responsibility
Unit Testing	Jest, PyTest, JUnit	Developers
Integration Testing	Postman, REST Assured	QA/Test Engineers
UI Testing	Selenium, Cypress	QA/Test Engineers
Security Testing	OWASP ZAP, Burp Suite	Security Team
Performance Testing	JMeter, Locust	DevOps/QA

9.2 Staging Environment

- Mandatory UAT (User Acceptance Testing) before production push.
- Staging must mirror production for configuration and data anonymization.

10. Logging and Monitoring

10.1 Logging

- Use structured logging formats (e.g., JSON).
- Log critical operations (logins, payment attempts, permission changes).
- Do not log passwords, tokens, or PII.

10.2 Monitoring

- Integrate Prometheus, ELK Stack, and Grafana dashboards.
- Set up alerts for error thresholds, service outages, and resource spikes.

11. Compliance and Audits

11.1 Compliance Checkpoints

- GDPR, ISO 27001, SOC2 considerations must be applied.

- Access logs and change logs must be preserved for 6 months.

11.2 Audit Table

Audit Type	Frequency	Conducted By
Security Audit	Quarterly	Internal Security Team
Code Quality	Bi-annually	Tech Leads + QA
Access Review	Annually	Compliance + IT Admin

12. Enforcement

12.1 Violations

- First violation: Written warning and retraining.
- Repeated violations: Removal of deployment access or disciplinary action.

12.2 Escalation

- Violations affecting production or user data will be escalated directly to the CTO and Compliance Committee.

13. Training and Awareness

13.1 Mandatory Training Topics

- Secure Coding (OWASP Top 10)
- Git and Branching Strategies
- Internal Tools Training (CI/CD, Vault, Docker)
- Secure API Development

- Introduction of new technology or architecture.
- Changes in regulatory/compliance environment.
- Recurrent audit findings.

All revisions will be communicated via internal email and published on the developer portal.

13.2 Training Calendar

Session	Frequency	Audience
Secure Coding Bootcamp	Bi-Annually	All Developers
Code Review Masterclass	Quarterly	Senior Developers
Tool Onboarding	Monthly	New Joiners

14. Review and Updates

The policy will be reviewed annually or upon: