Inazuma.co Software Development Lifecycle (SDLC) Policy

Version: 1.0

Reviewed By: IT Governance Committee

---

1. Objective

The purpose of this Software Development Lifecycle (SDLC) Policy is to establish a comprehensive, standardized, and auditable process for software development, deployment, and maintenance at Inazuma.co. This ensures the delivery of high-quality, secure, and reliable digital products that align with business objectives, enhance customer experience, reduce development risks, and comply with industry standards.

Key objectives include:

- Define consistent SDLC phases and deliverables.
- Encourage collaboration across cross-functional teams.
- Promote adherence to security and quality standards.
- Mitigate risks associated with software development.
- Support scalability and innovation within engineering practices.

---

2. Scope

This policy applies to:

- All software and system development projects managed or sponsored by Inazuma.co.
- All internal applications, third-party software, open-source integrations, mobile and web applications, APIs, microservices, cloud-native applications, and data systems.
- All employees, consultants, interns, contractors, and third-party vendors involved in software development or maintenance activities.
- Both Waterfall and Agile development environments.

This policy is binding at all organizational levels, including product development, DevOps, security, testing, and support teams.

---

3. SDLC Phases (Detailed)

3.1 Planning Phase

Activities:

- Business case development.

- Cost-benefit analysis and ROI estimation.
- Stakeholder alignment.
- Defining project charter.

Key Deliverables: Project Plan, Feasibility Report, Risk Register, Preliminary Resource Plan

3.2 Requirements Analysis Phase

- Conduct workshops, interviews, and surveys with stakeholders.
- Prioritize and validate requirements with the business team.
- Develop a Requirements Traceability Matrix (RTM).
- Perform requirement sign-off.

Key Deliverables: SRS (Software Requirements Specification), RTM, Use Case Documents, Approval Log

3.3 Design Phase

Activities:

- Technical architecture design
- Component and module-level design
- UI/UX wireframes and mockups
- Threat modeling and secure design principles

Key Deliverables: System Architecture Diagrams, Database Design (ERD), Mockups, Design Review Checklist

3.4 Development Phase

- Follow secure coding practices and standards (e.g., OWASP, CERT)
- Implement automated unit tests.
- Use Git for version control and establish branching strategies (e.g., GitFlow).
- Conduct daily stand-ups and sprint reviews.
- Perform static code analysis.

Development Standards Table

Standard	Tool/Process	Enforcement Level
Version Control	Git, Bitbucket	Mandatory
Code Review	Pull requests + Peer Review	Mandatory
Secure Coding	OWASP Top 10, SAST Tools	Mandatory
CI/CD Integration	Jenkins, GitHub Actions	Recommended

Documentation	In-line comments, Confluence	Mandatory

3.5 Testing Phase

Testing must be integrated into every development cycle. Types include:

- Smoke testing
- Regression testing
- Performance testing
- Security testing

Key Deliverables: Test Strategy Document, Test Cases, Test Reports, Defect Logs

Testing Metrics Table

Metric	Target
Test Coverage	> 90% code coverage
Defect Leakage	< 5% post-UAT defects
Mean Time to Detect	< 24 hours
Mean Time to Resolve	< 72 hours

3.6 Deployment Phase

Deployment activities:

- Prepare release notes and deployment plan.
- Validate against staging.
- Execute cutover plan.
- Use blue-green or canary deployment strategies when needed.

Deployment Checklist Includes:

- Data backup confirmation
- Rollback readiness
- Downtime announcement
- Post-deployment validation

3.7 Maintenance Phase

- Issue tracking via Jira or similar tools
- Monitoring via application performance tools (e.g., Datadog, Prometheus)
- Incident management
- Bug fixes and patch rollouts

SLAs for Maintenance Activities

Type	Resolution SLA
Critical Bug	< 4 hours
High Priority	< 24 hours

Medium Priority	< 48 hours
Low Priority	< 5 business days

4. Roles and Responsibilities (Expanded)

Role	Responsibility
Product Manager	Prioritize backlog, gather requirements, approve scope changes
Software Architect	Design scalable systems, validate infrastructure needs
Developers	Write clean, secure, maintainable code
QA Engineers	Develop test cases, automate tests, manage UATs
DevOps Team	Manage CI/CD, ensure availability, automate infra provisioning
Security Analyst	Perform risk assessments, review designs, validate data privacy
Project Manager	Track timelines, manage risks, drive SDLC adherence

5. Development Methodologies

Inazuma.co supports both Agile and DevOps-driven SDLC methodologies:

- **Agile**: Sprint planning, retrospectives, backlog grooming, continuous feedback.
- **DevOps**: Continuous Integration, Continuous Delivery, automated testing, and environment provisioning.
- **Shift-Left** testing and security adoption at early phases.

6. Compliance and Standards

All software must adhere to:

- ISO/IEC 27001: Information Security
- OWASP Secure Coding Guidelines
- GDPR, HIPAA, or PCI-DSS based on project requirements
- Internal policies including InfoSec and Privacy

Audits will be conducted bi-annually.

7. Documentation Requirements

Document	Owner	Update Cycle
Software Requirement Specification	Product Manager	Per Release
Architecture Design Document	Tech Lead	Once Per Project
Test Strategy Document	QA Lead	Per Major Release
Deployment Playbook	DevOps Lead	Each Production Deployment
User Documentation	Tech Writer	Post Feature Completion

All documents must be stored in Confluence and linked via Jira stories.

---

8. Risk Management

- Maintain project-specific risk registers.
- Classify risks into operational, security, compliance, and performance.
- Assess each risk based on likelihood and impact.
- Create mitigation strategies and assign ownership.

Sample Risk Table

Risk Type	Description	Impact	Mitigation Plan
Security	SQL Injection in login module	High	Use parameterized queries
Operational	Key dev unavailable during sprint	Medium	Cross-train team members
Performance	DB latency under load	High	Optimize queries, caching

---

9. Training and Awareness

- **Onboarding**: Secure coding, agile practices, and SDLC overview.
- **Monthly Sessions**: Deep dives into topics like threat modeling, CI/CD tools.
- **Annual Certification**: Internal SDLC and Secure Dev training.

Training Calendar Example

Month	Topic	Target Group
January	Threat Modeling & STRIDE	Architects, Developers
April	CI/CD Pipeline Optimization	DevOps, QA
August	Agile Best Practices	Product & PMs
November	Secure Code Review Techniques	Developers, Tech Leads

---

10. Policy Review and Revisions

- Annual review by IT Governance Committee.
- Feedback gathered from project retrospectives.
- Updates approved and version-controlled.
- Distributed to all development teams, stakeholders, and documented in Confluence.

Acknowledgment Form

I acknowledge that I have read, understood, and agreed to comply with the SDLC Policy of Inazuma.co.

For questions or clarifications, contact the Software Governance Office at **sdlc-policy@inazuma.co**.