Software Development Policy

IronHub Inc

August 2026

Contents

Table 1: Control satisfaction

Standard	Controls Satisfied
TSC	CC8.1

Table 2: Document history

Date	Comment
Jun 1 2022	Initial document
Feb 6 2025	Added Secure Coding Policy, Web Security Guidelines

1 Purpose

The purpose of this policy is to define the framework, processes, and best practices for software development within IronHub. This ensures that software development is efficient, secure, and aligned with business objectives. The policy applies to all employees, contractors, and third parties involved in software development at IronHub.

2 Scope

This policy applies to all software development projects within IronHub, including enterprise-wide systems, applications, and infrastructure, whether developed in-house or by external vendors. It covers the full Software Development Life Cycle (SDLC) and adheres to Agile and Scrum methodologies.

3 References

a. Risk Assessment and Management Policy

4 Development Methodology

IronHub follows an **Agile** approach using the **Scrum framework**, allowing for iterative development, continuous feedback, and rapid adaptation to business needs.

4.1 Agile Principles in IronHub

- Deliver working software frequently, with short iteration cycles.
- Welcome changes in requirements, even late in development.
- Promote collaboration between developers, product owners, and stakeholders.
- Maintain continuous attention to technical excellence and security.
- Conduct retrospectives to improve processes continuously.

4.2 Scrum Framework

IronHub's software development process follows **Scrum**, consisting of:

- **Product Backlog**: Maintained and prioritized by the Product Owner.
- **Sprint Planning**: Team selects and commits to backlog items for a sprint.

- **Daily Stand-ups**: Quick daily meetings to sync progress and resolve blockers.
- **Sprint Review**: Demonstration of completed features to stakeholders.
- **Sprint Retrospective**: Lessons learned and improvements for the next sprint.

5 Software Development Lifecycle (SDLC) at IronHub

IronHub's SDLC follows an iterative, Agile approach with the following phases:

5.1 Requirements Analysis

- Collaborate with stakeholders to gather business and technical requirements.
- Perform **risk assessments**, including security, compliance, and business impact.
- Define acceptance criteria for deliverables.
- Prioritize user stories and maintain a **well-groomed product backlog**.

5.2 Architecture & Design

- Define the system architecture, security controls, and design standards.
- Create technical specifications and conduct architecture walkthroughs.
- Ensure segregation of duties between development, testing, and production environments.
- Follow **secure coding practices** as per OWASP and industry standards.

5.3 Development

- Implement features in incremental iterations.
- All code changes to production branches require review and approval by **at least one other developer** before merging. The reviewer must not be the same person who authored the change. Security-sensitive changes (authentication, authorization, cryptography, data handling) require review by a developer with security expertise or the Security team.
- Ensure code is version-controlled and changes are documented.
- **Secrets Management:**

  - **All keys, secrets, and credentials (including third-party API keys) must NOT be stored in source code**.
  - Instead, they must be managed securely using environment variables, secrets managers (e.g., AWS Secrets Manager, HashiCorp Vault), or encrypted configuration files.
  - Periodic **secrets scanning tools** (e.g., GitHub Secret Scanning, TruffleHog, Gitleaks) must be used to prevent secret leakage.
  - Any leaked secrets must be **revoked and rotated immediately**.
- **Production Data Restrictions:**
  - **Production data must never be copied, transferred, or used in test, development, or staging environments.**
  - **Only synthetic, anonymized, or masked data may be used in non-production environments.**
  - Any use of actual production data outside the production environment is a **strict violation** and must be reported immediately.

5.4 Testing

- Conduct **automated and manual testing**, including unit, integration, and regression tests.
- Perform **security testing**, including static and dynamic code analysis.
- **Dependency and Supply Chain Scanning:** All third-party libraries and dependencies must be scanned for known vulnerabilities using automated tooling (e.g., Dependabot, Snyk, OWASP Dependency-Check) integrated into the CI/CD pipeline. Critical and high-severity dependency vulnerabilities must be remediated before deployment to production.
- Validate performance, usability, and business logic before deployment.

5.5 Deployment & Release Management

- Use **CI/CD pipelines** for automated builds, testing, and deployments.
- Conduct a **security and compliance review** before deployment.
- Implement **blue-green deployments** or **feature toggles** to reduce risks.

5.6 Operations & Maintenance

- Continuously monitor system performance and security.
- Apply security patches, updates, and vulnerability fixes within **30 days** or immediately for critical risks.

5.7 Decommissioning

- Ensure **data sanitization** and compliance with **data retention policies** before system retirement.

6 Security & Compliance Controls

- **Code Reviews:** Mandatory for all commits, including security-focused reviews.
- **Access Controls:** Developers have **least privilege** access to production systems.
- **Penetration Testing:** Conducted before major releases and annually for security compliance.

6.1 Secure Coding Policy

- Follow **OWASP Secure Coding Practices**.
- Use **input validation** and **output encoding** to prevent injection attacks.
- Avoid **hardcoded credentials** in source code.
- **Enforce authentication & authorization controls** for APIs and internal endpoints.

6.2 Web Security Guidelines

- Implement **secure cookies**, CSRF protection, and session management best practices.
- Disable **HTTP TRACE** method in production.
- Ensure all user inputs are **sanitized** to prevent **XSS attacks**.
- Use **Content Security Policy (CSP)** headers to prevent **clickjacking**.
- Encrypt sensitive information before storing it in the database.

6.3 Cryptography Standard

- **Custom cryptographic algorithms are NOT allowed**.
- All encryption methods must use **NIST-approved cryptographic algorithms**.
- **Secrets and encryption keys must be rotated regularly**.
- **All sensitive data must be encrypted at rest and in transit**.

7 Change Management & Governance

- All **production changes** must go through a **change control process**.
- Emergency fixes follow a **fast-track approval** but require post-implementation review.

8 Enforcement & Compliance

- **Failure to comply with this policy may result in disciplinary action, up to termination.**
- **Security violations must be reported to the Security & Compliance Team immediately.**
- Regular audits will be conducted to ensure compliance.

This policy ensures that IronHub's software development aligns with **business goals, security best practices, and Agile principles** while delivering high-quality products efficiently.