# Synthetic Users — Secure Software Development Lifecycle (SDLC) Policy

## Secure Software Development Lifecycle (SDLC) Policy

**Version:** 1.7

**Effective Date:** November 2025

**Last Updated:** March 25, 2026

**Document Owner:** CTO — Artur Ventura

**Review Frequency:** Annually or upon significant architectural or security change

**Classification:** Internal – Confidential

## Change History

| Version | Date | Author | Changes |
|---|---|---|---|
| 1.6 | November 2025 | CTO / Security Lead | Prior release |
| 1.7 | March 25, 2026 | Artur Ventura, CTO | Added explicit CWE/SANS Top 25 reference to development and testing standards (Section 3.3, 3.4). Added AI/GenAI scope section (Section 6) cross-referencing the SDLC AI/GenAI Addendum. Updated per JPMC SCA CRA 13.1.1. |

## 1. Purpose

This policy defines the Secure Software Development Lifecycle (SDLC) used by Synthetic Users to ensure that all software is designed, developed, tested, deployed, and maintained in a secure and controlled manner. The objective is to protect customer data, reduce security risk, and align with applicable security and privacy frameworks, including **SOC 2**, **GDPR**, and **ISO 27001** principles.

## 2. Scope

This policy applies to:

- All Synthetic Users software, services, APIs, and infrastructure
- All employees, contractors, and third parties involved in software design, development, testing, deployment, and maintenance

## 3. SDLC Stages and Security Controls

### 3.1 Planning and Requirements

- Security, privacy, and compliance requirements are defined for all new features and material changes.
- Risk assessments and threat modeling are performed for new systems, integrations, or significant architectural changes.
- Open-source and third-party components are evaluated for known vulnerabilities and license compliance prior to adoption.

## 3.2 Design

- Security and privacy impact assessments (PIA/DPIA), where applicable, are conducted during design.
- Architecture is reviewed to ensure appropriate:
  - Encryption of data in transit and at rest
  - Access control and least-privilege principles
  - Logical data segregation
- Authentication and authorization requirements are validated, including support for **SSO**, **RBAC**, and **MFA**.

---

## 3.3 Development

- Engineers follow secure coding standards aligned with **OWASP Top 10** and the **CWE/SANS Top 25 Most Dangerous Software Weaknesses**.
- All code changes require peer review through **Git-based pull request (PR) workflows**.
- Automated static analysis (SAST) is enforced using **GitHub Advanced Security / CodeQL**, configured to report CWE identifiers alongside findings. Findings mapping to CWE/SANS Top 25 entries are automatically elevated in severity.
- Dependency scanning is enforced using **Dependabot**. Critical and high-severity dependency vulnerabilities block deployment until remediated.
- Hardcoded secrets are prohibited. Secrets are managed via **Render environment variables** or approved password managers (e.g. **1Password**).
- CI/CD pipelines enforce reviewed merges and controlled deployments.

---

## 3.4 Testing

- Automated unit, integration, and regression tests are executed as part of CI/CD pipelines.
- Dynamic application security testing (DAST) and vulnerability scanning are performed in non-production environments. DAST findings are triaged using **OWASP Top 10** as the primary taxonomy and **CWE/SANS Top 25** identifiers for engineering-level remediation tracking.

- Penetration test reports are required to map findings to CWE identifiers where applicable.
- Production data is not used in testing; anonymized or synthetic data is required.
- Deployments are blocked if critical or high-risk vulnerabilities are detected and not remediated.

---

## 3.5 Deployment

- Deployments are performed through automated CI/CD pipelines (e.g. GitHub Actions, Render deploy hooks).
- Environments are segregated (development, staging, production).
- Infrastructure is managed using **Infrastructure as Code (IaC)** and version-controlled repositories.
- All deployments are logged and auditable.

---

## 3.6 Monitoring and Maintenance

- Continuous monitoring for vulnerabilities and dependency updates is maintained.
- Operating systems and libraries are patched on a regular schedule or sooner for critical vulnerabilities.
- Periodic third-party penetration testing is conducted.
- Incident response, rollback, and recovery procedures are documented and tested.
- Change management activities are tracked through GitHub and internal documentation systems.

---

## 3.7 Training and Documentation

- Engineers receive periodic training on secure coding, data protection, and security best practices.
- Developer standards, SOPs, and architectural guidance are maintained internally.
- Evidence such as code reviews, deployment logs, and security findings is retained for audit and compliance purposes.

---

## 4. Supporting Tools and Technologies

Synthetic Users leverages the following tools to support the SDLC:

- **GitHub** (version control, PR reviews, CI/CD integration)
- **GitHub Advanced Security** (code scanning)
- **Dependabot** (dependency vulnerability monitoring)
- **AWS** (core infrastructure, compute, storage, database, networking)
- **Render** (application hosting and configuration management)
- **1Password** (credential management)
- **Monitoring and logging tools** for observability and alerting
- **Automated testing frameworks** for security and regression testing

## 5. Review and Compliance

This policy is reviewed at least annually by the CTO and Security Lead. Updates are approved through the change management process and communicated to relevant personnel. Compliance with this policy is mandatory and subject to periodic review and audit.

## 6. AI/GenAI Scope

Synthetic Users' platform includes AI/GenAI capabilities (multi-provider LLM orchestration, Persona Engine, RAG pipeline). These capabilities are within the scope of this SDLC policy for all standard controls (access, secrets management, code review, CI/CD, testing).

AI/GenAI-specific lifecycle requirements — including model validation, AI/GenAI risk assessments, retraining governance, adversarial testing strategies, and OWASP Top 10

for LLM Applications alignment — are defined in the SDLC AI/GenAI Addendum, which is a binding supplement to this policy.

All engineers working on AI/GenAI features are required to comply with both this policy and the Addendum.

---

## 7. Related Documents

- SDLC AI/GenAI Addendum — Binding supplement covering AI/GenAI model lifecycle, CWE/SANS Top 25 alignment, and AI-specific testing (CRA 13.1.1)
- AI/GenAI Algorithm Design Document — System architecture and data flow for AI/GenAI features
- Change Management Policy — Process for managing changes to systems and infrastructure
- Incident Response Plan — Response procedures including AI/GenAI-specific incidents
- Third-Party Risk Management Policy — Vendor risk framework covering AI/GenAI model providers

---