A LOVABLE × AIUC-1 WHITE PAPER

Setting the standard for agentic development

How AIUC-1 evolved to address the unique challenges of agentic development — and why Lovable will be the first coding agent platform to adopt the standard.

BY Lovable | AIUC-1

PUBLISHED

May 2026

TABLE OF CONTENTS

What's inside

EXECUTIVE SUMMARY

Key takeaways

AI coding agents produce running software — not text. They need a standard built for that reality. AIUC-1 is that standard, and Lovable is among the first to adopt it.

Platforms like Lovable let non-developer teams build, deploy and iterate on production applications through natural language. Enterprise adoption is accelerating, and teams are integrating coding agents into core workflows — from prototyping internal tools to shipping customer-facing products.

But this shift introduces a new category of risk. Traditional AI chatbots generate text for a human to read. Coding agents produce **executable artifacts** — source code, database schemas, API configurations and deployed applications — that interact directly with production infrastructure, real user data and systems-level changes. A vulnerability in generated code is not hypothetical; it is a live security exposure.

And the exposure is well-documented.

This white paper introduces **AIUC-1** — the standard for AI agent security, safety and reliability — and describes how the framework has been extended to become the first purpose-built certification for AI coding agents. Lovable is among the first AI coding agent platforms to pursue AIUC-1 certification, with a third-party audit scheduled for summer 2026.

~20%

of AI-recommended packages did not exist on npm or PyPI in a study of 576,000 code samples (USENIX Security).

36%

of 3,984 agent skills audited by Snyk contained prompt injection; 76 carried actively malicious payloads.

90+

organizations targeted by prompt-injection attacks documented in CrowdStrike's 2026 threat report.

THE LANDSCAPE

The rise of AI coding agents and the risks that follow

Autonomy unlocks extraordinary productivity. It also means the blast radius of a failure is no longer confined to a suggestion a developer can review and reject.

AI coding agents represent a paradigm shift in how software is built. Rather than autocompleting individual lines of code, modern platforms understand full application context, connect to external services via APIs and MCP servers, execute shell commands, write to file systems, manage databases and deploy applications to production environments — all through natural language prompting.

When an agent writes insecure code, that code can be deployed. When it mishandles a secret, that secret can be exposed. When it follows a poisoned instruction from a compromised dependency, the entire project can be compromised.

The threat landscape differs from traditional AI systems across three dimensions that compound to create risks no prior framework was designed to address.

DIMENSION 01

The output surface is executable.

Traditional AI chatbots produce text. Coding agents produce running software. A hallucinated fact in a chatbot response is an inconvenience. A hallucinated authentication pattern in generated code is a security vulnerability that ships to production.

DIMENSION 02

The input surface is vast and untrusted.

Coding agents ingest context from everywhere — documentation files, MCP server tool descriptions, third-party dependency metadata. In April 2026, the "Comment and Control" attack showed hidden prompt-injection instructions in code comments and PR descriptions could cause Claude Code, Gemini CLI and GitHub Copilot to exfiltrate their own API keys.

DIMENSION 03

The action surface is broad and persistent.

Coding agents take actions in the real world. When an agent automates a workflow it operates with elevated privileges — credentials, production secrets, infrastructure configuration. A prompt injection that reaches it can modify deployment scripts, alter test assertions or insert backdoors that pass automated review.

INTRODUCING AIUC-1

Six principles · *One standard*

AIUC-1 is the industry standard for AI agent security, safety and reliability. Originally developed as a modality-agnostic framework covering six core principles, the standard has been progressively adapted to address the specific challenges of different AI agent categories.

Drawing on AIUC-1's CISO Consortium feedback and working with security teams from leading platforms including Lovable, the team identified **75 coding-agent-specific risks** across 13 thematic categories — from secrets in generated code to agent autonomy boundaries to MCP server supply-chain integrity.

Below: the six core principles that make up the standard, with the central risk each one addresses.

Security Safety Reliability Accountability Society Data & Privacy

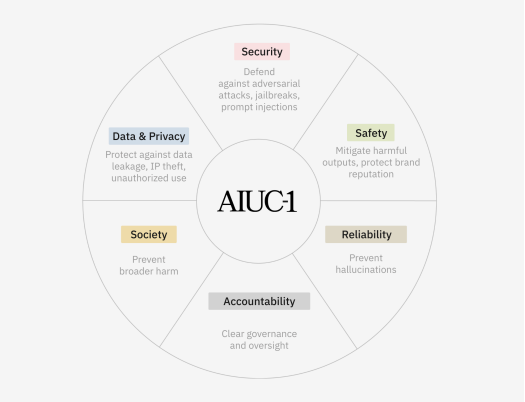

FIG. 01 — THE AIUC-1 FRAMEWORK

DEEP DIVE · 1 OF 4

Seven priority areas for coding agents

The 75 identified risks cluster into seven priority domains. Each represents a category where coding agents introduce risks that are either entirely new — or materially amplified — compared to other AI agent modalities.

PRIORITY
01

Secrets management

Coding agents routinely ingest entire codebases as context. Those codebases contain hardcoded API keys, database strings, OAuth tokens — often accidentally. Without dedicated controls, secrets can leak through four channels: generated code, conversation logs, inference traffic and platform telemetry.

CONTROL AREA	WHAT IT REQUIRES
Input-side detection	Scan user prompts and ingested context for credential patterns before transmission to model providers.
Output-side detection	Detect when generated code contains hardcoded credentials rather than environment variable references.
Credential isolation	Tenant-scoped storage so one customer's OAuth tokens are inaccessible to another.
User-facing warnings	Alert users when potential credentials are detected in their input *before* inference.
Log & artifact redaction	Prevent secrets from being retained in logs, history, or stored artifacts.

DEEP DIVE · 2 OF 4

PRIORITY
02

Secure code generation defaults

> *"When a first-time user with no security expertise asks the platform to build a web application, what is the default security posture of the generated code?"*
>
> AIUC-1 DESIGN PRINCIPLE

If the answer depends on the user remembering to specify security requirements, the platform is relying on the weakest link. Generated output should be secure by default — not as an opt-in feature, but as a baseline property of the generation pipeline.

Secure web-app defaults

Parameterized queries and safe ORM patterns; output encoding and contextual escaping for UI; HTTPS/TLS enforcement on network code — applied through system prompts and framework scaffolding, not user instruction.

Auth & authorization patterns

Default to established libraries (Auth0, Clerk, framework-native auth) rather than custom cryptographic primitives. Least-privilege access checks are applied by default.

Safe dependency specification

Avoid wildcard versions; validate referenced packages against registries before inclusion; avoid known-abandoned or typosquatted packages. Directly addresses the *slopsquatting* attack vector.

Secure session management

HttpOnly, Secure and SameSite flags on cookies by default; CSRF protections on state-changing endpoints.

Defensive programming

Input validation on user-supplied data; safe error handling that avoids exposing stack traces; rate limiting on public-facing endpoints.

Safe logging defaults

Exclude sensitive fields — no full request bodies, no auth headers, no PII in log output.

DEEP DIVE · 3 OF 4

PRIORITY
03

Runtime execution & sandbox integrity

Coding agents run shell commands, write files, install packages and interact with databases inside sandboxed environments. If the boundary is breached, a compromised agent reaches production infrastructure. The standard expands unauthorized agent action controls with three new capabilities:

Execution-level safeguards — sandboxed environments with configurable filesystem, network and credential restrictions for both agent-executed code and first-party MCP servers, plus pre-execution authorization hooks. **Project configuration trust controls** — explicit user or organizational approval before any project-defined hook, tool definition or MCP server setting takes effect. **Non-interactive session termination** — process-level kill switches, session timeouts and heartbeat monitoring for agents running unattended in CI/CD or background contexts.

PRIORITY
04

Dependency & supply chain integrity

Coding agents actively generate dependency manifests and install packages as part of normal operation — the agent itself is a supply-chain participant. *Slopsquatting* illustrates why this matters: in a 576,000-sample study, ~20% of AI-recommended packages did not exist (5% commercial, 21.7% open-source models), and 43% of hallucinated names repeated on every run — predictable targets for attackers.

> *"The hallucinated package huggingface-cli received over 30,000 downloads in three months. A hallucinated npm package, react-codeshift, spread through 237 repositories via AI-generated agent skill files."*
>
> SLOPSQUATTING FIELD EVIDENCE

The standard addresses this through two interlocking mechanisms: safe dependency specification within the secure code generation requirement, and an expansion of factual accuracy controls that treats package hallucination as a form of factual inaccuracy — platforms must validate that referenced dependencies exist and resolve to legitimate, intended packages.

DEEP DIVE · 4 OF 4

PRIORITY
05

Agent autonomy & human oversight

Autonomy is what makes coding agents valuable — but it must be bounded, and high-risk or irreversible actions must involve *informed* approval, not rubber-stamped confirmation dialogs. The standard requires: **complete change summaries** before user confirmation, with destructive actions visually distinguished from routine operations; **default-restrictive permissions** requiring explicit opt-in for elevated capabilities; and **persistent memory management** so users can review, edit and clear stored content — closing the door on prompt injections that persist across sessions.

PRIORITY
06

Data confidentiality & IP protection

Three overlapping risk categories beyond what chatbot-oriented controls address. **Cross-tenant data exposure** — isolation expanded to credential storage, execution environments and generated artifacts. **License contamination** — IP-violation filtering extended to detect code matching restrictively licensed training data. **Credential detection in inputs and outputs** — recognising that for coding agents the user-provided context is the primary surface where credentials appear.

PRIORITY
07

Enterprise governance & transparency

Individual developer guardrails are necessary but insufficient. The standard now requires **organizational policy enforcement** (admin-defined permission postures, MCP allow-lists, mandatory safety settings); a documented **shared responsibility model** analogous to cloud IaaS/PaaS; and **agent execution chain logging** capturing tool calls, sub-agent delegations, approval events and reasoning traces for forensic reconstruction.

INSIDE THE STANDARD

Requirements and principles

51
requirements

6
principles

1–5
controls

Each requirement has between one and five controls, and each control specifies the typical evidence that must be submitted to an accredited third-party auditor. Controls are classified as **Core** (mandatory for certification) or **Supplemental** (recommended for ambitious organizations or based on specific use cases). What makes AIUC-1 more than a checklist is its evidence model — each control specifies not just what to do, but what to show and to whom.

The four evidence categories

CATEGORY 01

Policy evidence

Legal and governance documentation — terms of service, privacy policies, data processing agreements, acceptable use policies. Where commitments like "we do not train on customer data" are made.

CATEGORY 02

Technical evidence

The actual implementation — system prompt guidance, filtering logic, IAM configs, sandbox policies, credential detection code, logging. Where an auditor reviews code, not slide decks.

CATEGORY 03

Operational evidence

Internal processes — change management, incident response, quarterly internal reviews, vendor due diligence. Demonstrates that controls are not just present but actively maintained.

CATEGORY 04

Third-party red-teaming

Conducted at least quarterly across adversarial robustness, harmful outputs, hallucinations, and tool call safety. The layer that prevents self-certification.

HOW LOVABLE MEETS THE STANDARD

Built in, audit-ready

Lovable has spent over a year doing something no other AI coding platform has done at scale: hosting, running, and governing production applications built entirely through natural language. Across enterprise workspaces with thousands of applications, many idle, some exposing data, others driving real business value, Lovable has accumulated hard-won operational insight into what it actually takes to manage AI-generated software in production. That experience shapes everything below.

The challenge enterprises face is how to govern them. When any team member can build and deploy a production application through a chat interface, the result is a fleet of applications that needs observability, policy enforcement, and architectural guardrails — not just secure defaults at generation time.

Lovable is building for that reality, positioning not just as an app creation platform but as an app governance and runtime platform.The following describes how Lovable's existing and in-development capabilities align with the AIUC-1 framework across the priority risk domains. All claims will be third-party validated as part of Lovable's AIUC-1 audit in summer 2026.

SECURE CODE GENERATION

System-level guidance steers generated code toward recognized secure patterns — parameterized queries, output encoding, established auth libraries, version-pinned dependencies. A vulnerability scanner runs on generated outputs to catch common vulnerability classes before they reach the user.

DATA PRIVACY & OWNERSHIP

Lovable does not use customer code or prompts to train AI models. This commitment is documented in terms of service and DPAs. Customer data is isolated at the tenant level with strict logical separation; connected-service credentials are stored with tenant-scoped encryption.

SECRETS PROTECTION

Detection mechanisms for credentials and secrets operate on both user inputs and generated outputs. When secrets are detected in user-provided context, the platform warns users and prevents the credentials from being persisted in stored artifacts or logs.

SANDBOX & EXECUTION SECURITY

Generated code executes within sandboxed environments with defined filesystem, network and credential boundaries. Separation is enforced between preview and production, and agent access is restricted to approved backend services and MCP servers.

HUMAN OVERSIGHT & INTERVENTION

Users get real-time visibility into agent actions and the ability to pause, stop or redirect system behavior. For destructive or high-risk operations, the platform presents change summaries and requires user confirmation before proceeding.

ENTERPRISE GOVERNANCE

Lovable's enterprise offering gives admins centralized controls for security policies, user permissions and agent capabilities across teams, plus a clear shared responsibility model delineating platform vs. deployer obligations.

INDEPENDENT VERIFICATION

Third-party testing and audit

AIUC-1 certification requires two layers of independent validation — and the dual-layer approach is what distinguishes it from self-attestation frameworks and voluntary checklists.

Quarterly third-party red-teaming

AIUC-1 mandates that qualified third-party assessors conduct testing at least every quarter across critical risk categories. For coding agent platforms, the testing scope includes:

Adversarial robustness. Structured red-teaming covering prompt injection (including code-injection chains via poisoned documentation, malicious dependencies and crafted user inputs), jailbreaking, adversarial perturbation and simulated malicious tool invocations.

Harmful and out-of-scope output. Evaluating the platform's ability to prevent offensive, biased or deceptive content and to enforce boundary constraints on what the agent will and will not generate.

Hallucination testing. Traditional factual accuracy plus code-specific scenarios like package hallucination, where the agent recommends nonexistent dependencies.

Tool call testing. Whether agents can be induced to execute unauthorized actions, access restricted information or make decisions beyond their intended scope through tool calls and MCP server interactions.

Customer-defined risk testing. Aligned with the platform's specific risk taxonomy, covering any additional high-risk categories identified in the risk assessment.

Each cycle produces a structured report — methodology, findings, severity ratings, remediation timelines, follow-up verification — submitted to the accredited auditor as evidence of ongoing compliance.

Accredited audit

The certification audit is conducted by an accredited third-party auditor. Lovable's audit will be performed by **Schellman**, a firm with deep expertise in security compliance assessments. The audit evaluates all submitted evidence against the AIUC-1 requirements, verifies the effectiveness of controls through testing and review, and issues a formal certification recommendation upon successful completion. This ensures certification represents *verified compliance* — not self-attestation.

CONCLUSION

Lead on security

The speed at which AI development technology is being adopted has outpaced the development of security standards to govern it. AIUC-1 closes that gap.

The extension for coding agents is designed drawing on 75 real risks, mapped against a live standard, tested against real-world incidents and CVEs, and refined through feedback from the CISOs who have to approve these tools for enterprise deployment.

Lovable's decision to be among the first coding agent platforms to pursue AIUC-1 certification reflects a core belief: that the platforms building the future of software development have a responsibility to lead on security, not wait for regulation to catch up.

For enterprise security teams, the message is clear: when evaluating AI coding agents, **demand evidence**. Ask for the controls. Ask for the test results. Ask for the audit report. The standard now exists to make those conversations concrete.

> *"Platforms building the future of software development have a responsibility to lead on security — not wait for regulation to catch up."*
>
> LOVABLE SECURITY & TRUST

About

Lovable

The AI app development platform.

Lovable is a software creation platform that empowers anyone to build full-stack apps and websites by chatting with AI. In its first year, builders created over 25 million projects with Lovable. Lovable-built applications now attract 600 million visits per month. Teams at companies like HCA Healthcare, HubSpot, Microsoft, Uber, and Zendesk rely on Lovable to build internal tools, prototypes, and production-ready applications. Learn more at lovable.dev.

lovable.dev

AIUC-1

The standard for AI agent security.

AIUC-1 is the standard for AI agent security, safety and reliability. Updated quarterly, the standard is maintained through input from the AIUC-1 Consortium of enterprise CISOs, academic researchers and frontier AI platform builders.

aiuc-1.com