SiteGuard

AI Development SDLC

Contractor compliance prototype - contracts, insurance, reports, and workspace configuration

Prototype position

This document explains the software development lifecycle for the current SiteGuard app. The application is a working prototype intended for structured feedback from non-developer end users, then broader pilot testing, then security and larger development-team vetting before final release.

Prepared for

Prototype review, tester onboarding, and stakeholder planning

Development model

AI-assisted prototyping with Codex, GitHub source control, and GitHub Actions build automation

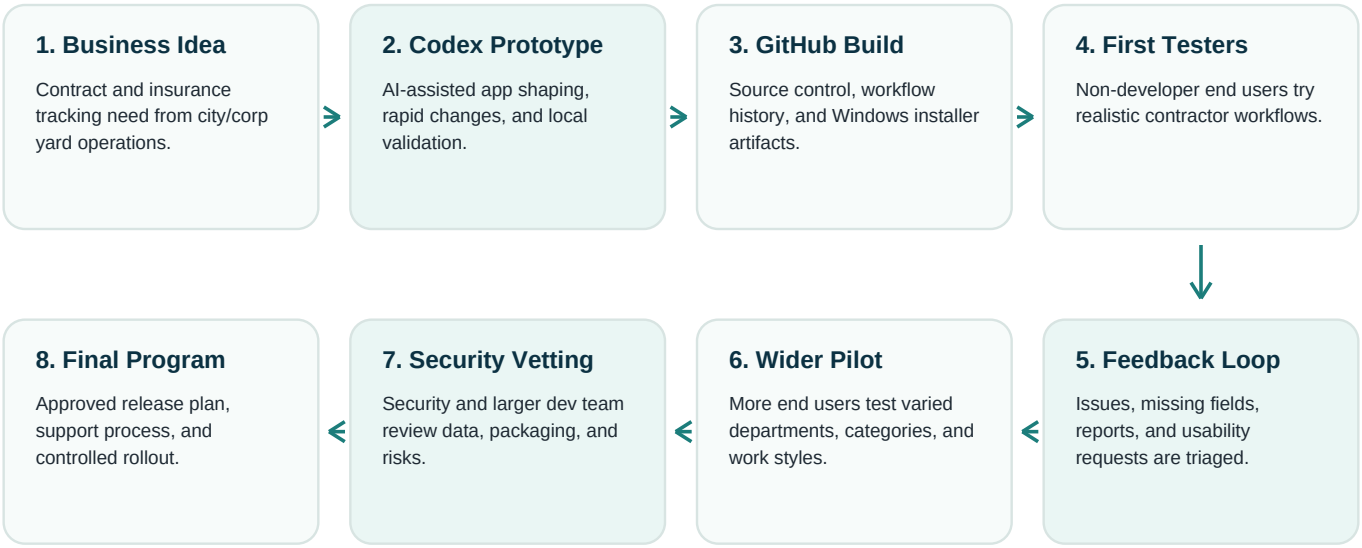
Document date: 2026-07-14

Lifecycle Overview

How this prototype moves from idea to validated program

AI-Assisted Prototype SDLC Graphic

The app remains a working prototype until security and broader engineering review are complete.



Iteration rule Feedback from tester phases can return to Phase 2 for Codex-assisted changes, then move through validation and rebuild again.

Gate A
Prototype ready for first testers

Gate B
Feedback incorporated for wider pilot

Gate C
Security and dev vetting complete

Gate D
Final program release decision

Important status note

SiteGuard can be demonstrated and tested as a working application now. It should still be labeled as a prototype until security review, vulnerability assessment, and broader engineering review are complete.

SDLC Principles

The operating rules for this AI-assisted prototype

Purpose of this SDLC

This lifecycle is designed for a practical city management or corporate yard compliance app. The goal is to move quickly enough to learn from real users while preserving traceability, review checkpoints, and a clear boundary between prototype testing and production release.

Core principles

- Build around real workflows: contractor onboarding, contract dates, insurance dates, document status, reporting, and workspace categories.
- Use Codex for rapid implementation, review, documentation, and iteration, while keeping human ownership of product direction and approval gates.
- Use GitHub as the source-of-truth for code history, issue tracking, release packaging, and automated Windows builds through GitHub Actions.
- Collect end-user feedback before overbuilding; testers should validate whether the process replaces or improves the Excel workflow.
- Keep security boundaries explicit: no production-sensitive data should be required during early testing unless the organization approves that data use.
- Treat each testing wave as evidence gathering: what works, what confuses users, what fields are missing, and what risks need attention.

Roles in the prototype SDLC

Role	Responsibility
Product sponsor / PM-CTO	Owns the app vision, prioritizes feedback, decides when the prototype is ready for each testing gate.
Codex-assisted builder	Implements changes, runs local checks, updates help content, and prepares builds or GitHub workflow updates.
First testers	Non-developer potential end users who test realistic day-to-day workflows and explain what would make the app useful.
Wider pilot users	Broader group that validates department differences, report needs, category changes, and reliability under varied use.
Security and dev reviewers	Review data protection, packaging, dependencies, file handling, permissions, and vulnerability risks before finalization.

Detailed Phases: Build To First Testing

What happens before the first non-developer testers receive the app

Phase	Purpose	Exit Evidence
1. Idea and workflow discovery	Capture the business need: replace or improve spreadsheet tracking for contractor compliance, contracts, insurance, due dates, overdue flags, and configurable categories.	A plain-language scope statement, starter category list, key users, sample workflows, and known constraints.
2. Prototype architecture	Shape a local desktop app that can save/open project files, protect workspaces with passwords, export reports, and allow flexible workspace layouts.	Working local app structure, project file format, app icon, help guide, and initial UI workflows.
3. AI-assisted implementation	Use Codex to rapidly implement screens, save logic, exports, drag/resize workspace panels, help content, tests, and package scripts.	Code changes in Git, local validation results, and a build that can be launched on macOS for testing.
4. GitHub and build automation	Push the source to GitHub and use GitHub Actions to build Windows installers. Maintain history so changes can be traced and rebuilt.	Repository on GitHub, Windows build workflow, downloadable NSIS EXE and MSI artifacts, and a repeatable build path.

Gate A - Ready for first testers

The prototype is ready for the first non-developer testers when the app launches, a new project can be created, Save/Open/Save As work, reports export, help content explains the workflow, and the Windows installer can be produced from GitHub Actions.

Detailed Phases: Testing To Finalization

How feedback becomes a vetted program

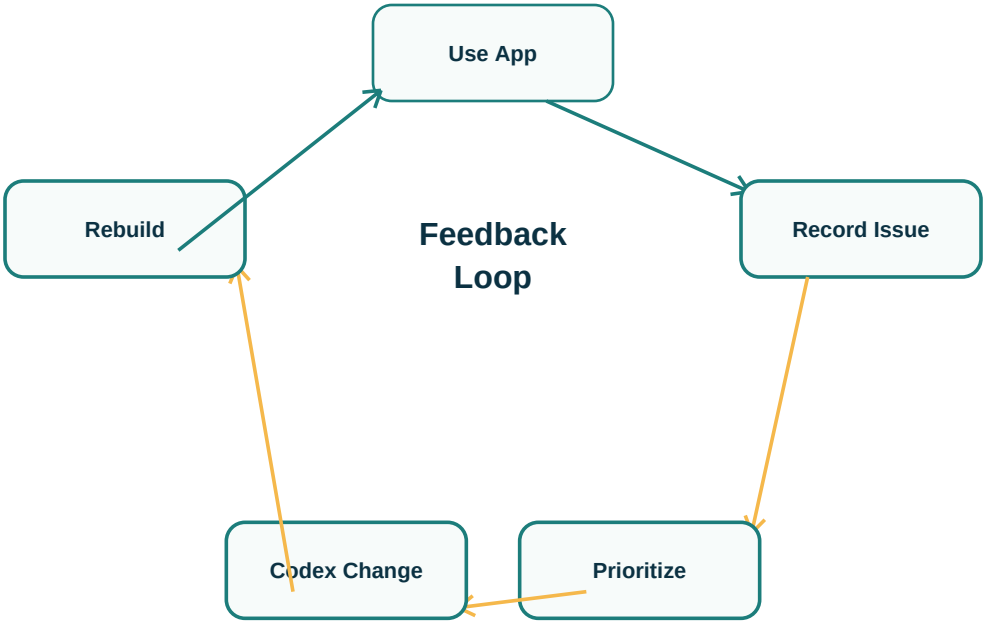
Phase	Purpose	Exit Evidence
5. First end-user testing	Give the app to a small group of potential end users who are not developers. Ask them to perform realistic contract intake, review, reporting, and workspace setup tasks.	Tester feedback notes, usability issues, missing fields, confusing labels, workflow gaps, and prioritized change requests.
6. Codex iteration cycle	Bring feedback back into Codex. Convert observations into targeted app changes, update help content, run tests, rebuild, and push updates to GitHub.	Updated prototype build, documented changes, passing checks, and an explanation of what changed for testers.
7. Wider pilot testing	Offer the improved prototype to a larger group of end users across more departments, job types, and reporting needs to test flexibility and reliability.	Pilot feedback, report expectations, category/workspace templates, performance notes, and final product requirements.
8. Security and final release review	Before calling it final, security and a larger dev team review vulnerabilities, dependencies, file protection, encryption, installer behavior, update process, and support needs.	Security findings resolved or accepted, dev review complete, release checklist approved, and a final rollout decision.

Prototype boundary

The program may be functionally usable during pilot testing, but it should not be treated as a finalized production product until the security and larger development-team review is complete. This protects the organization and keeps expectations clear.

Tester Feedback Workflow

How non-developer feedback should be collected and converted into app changes



Recommended tester questions

- Could you replace the current Excel process with this? If not, what exact step is missing?
- Which fields, categories, contract stages, or reports need different names for your department?
- Were overdue and upcoming-expiration flags clear enough to act on?
- Could you create, save, reopen, and protect a project without help?
- Which actions felt slow, confusing, risky, or too hidden?
- What would you need before trusting this app for real operational work?

Feedback categories to track

Workflow gap | Missing data field | Report/export need | UI wording | Bug or reliability issue | Security/privacy concern | Training/help content | Nice-to-have enhancement

Testing data rule

Use sample data, sanitized records, or data approved for prototype testing. Do not require sensitive production data until the security review confirms how the app should handle it.

Artifacts, Checks, And Release Gates

What should exist before the prototype becomes final

Key artifacts

- GitHub repository with source code, commit history, and Windows build workflow.
- Downloadable Windows installer artifacts from GitHub Actions: NSIS setup EXE and MSI package.
- Help guide explaining project creation, save/open, password protection, master panel setup, dashboards, reports, and exports.
- Tester feedback log with decisions: accepted, deferred, rejected, needs clarification, or security review required.
- Release notes for each tester build so users know what changed and what to retest.
- Security review notes covering project file protection, local cache behavior, dependency risks, packaging, permissions, and update strategy.

Minimum checks before wider pilot

- App installs and launches on a clean Windows test machine.
- Create, Save, Save As, Open, Close Project, and password-protected workspace flows work consistently.
- Reports export to Excel and PDF with overdue and upcoming-expiration information visible.
- Dashboard panels can be docked, undocked, resized, reordered, closed, reopened, and reset.
- Help guide matches the current UI and uses non-technical language.
- No demo/sample data remains in new clean projects unless intentionally included as a sample workspace.

Finalization gate

- Product sponsor confirms the workflow solves the real operational problem.
- Wider pilot users confirm the app works across the main departments and category styles.
- Security review confirms acceptable risk or required fixes are completed.
- Larger dev team review confirms maintainability, dependency choices, packaging, and support plan.
- A final version number, release notes, installer, and support contact path are approved.

Plain-language summary

This SDLC keeps the speed of AI-assisted prototyping while adding enough structure for responsible adoption: learn from real users, iterate quickly, validate with broader testers, and only finalize after security and engineering review.