



# CodeGuard – AI-Powered Code Reviewer and Quality Assistant

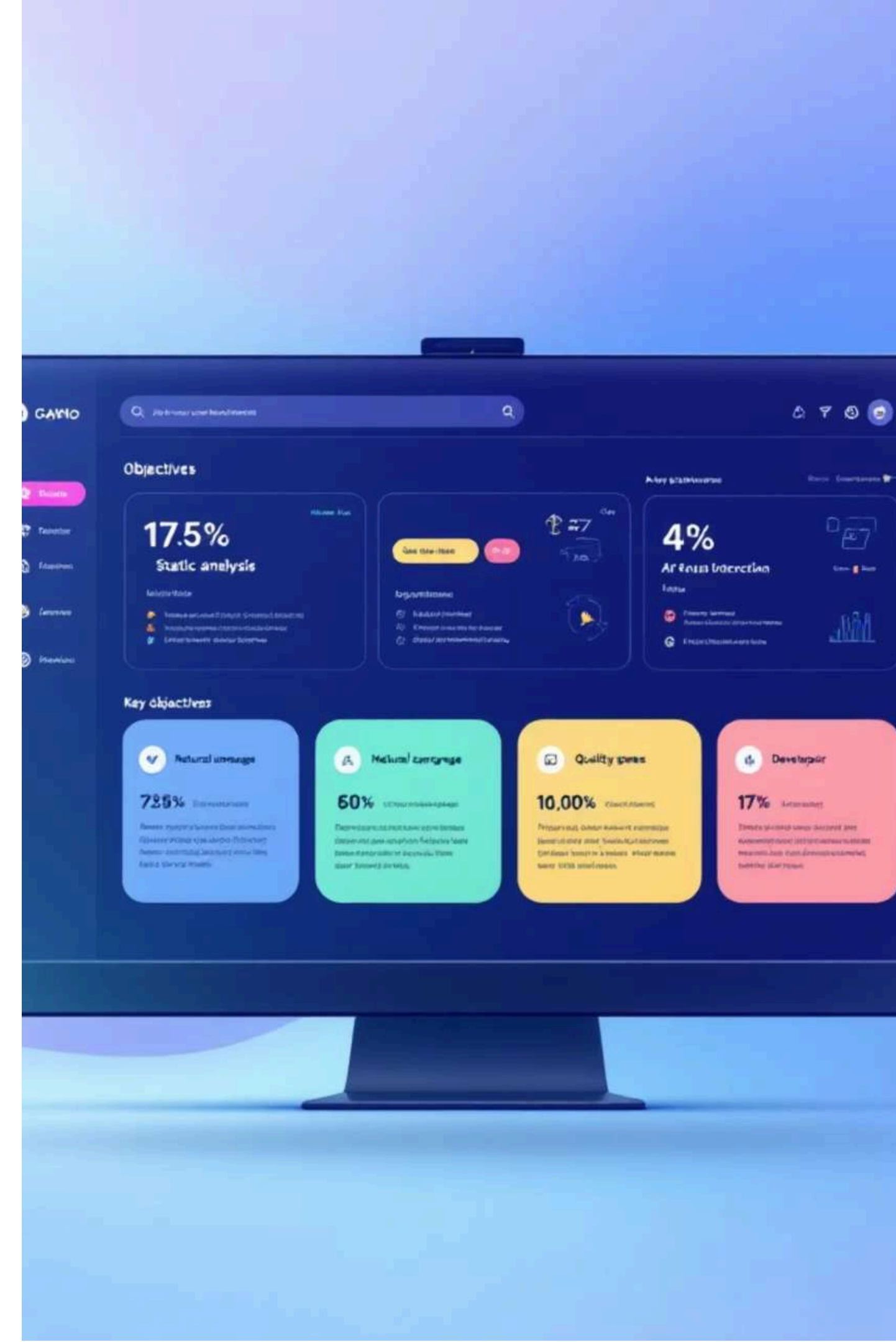


# PROBLEM STATEMENT

Code reviews are essential for maintaining software quality, but traditional manual reviews are often time-consuming, inconsistent, and prone to human oversight. Developers struggle to keep documentation, readability, and coding standards intact across large projects, while existing static analysis tools provide only technical checks without contextual, human-like feedback. This leads to reduced maintainability, slower onboarding, and quality drift over time. To address these challenges, there is a need for an AI-assisted solution that can automatically detect issues, explain them in natural language, and suggest actionable fixes, thereby improving efficiency, consistency, and developer understanding.

# OBJECTIVES

- 1 Automate code review using static analysis and AST parsing
- 2 Leverage Ollama Phi-3 model to provide natural-language feedback and auto-fixes
- 3 Ensure quality gates with Git pre-commit hooks and CI/CD integration
- 4 Provide developer interaction via CLI commands and Streamlit dashboard



# PROJECT OVERVIEW



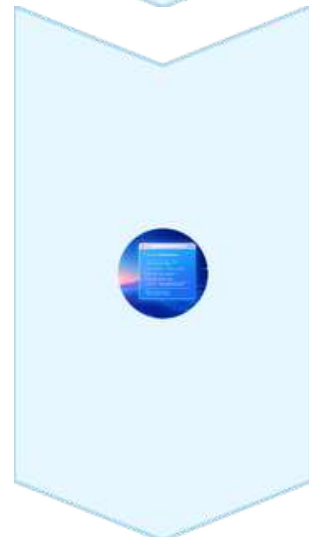
## Deliverables

AI-assisted code review tool for Python projects combining static analysis and Ollama Phi-3 feedback to deliver actionable suggestions.



## Developer interfaces

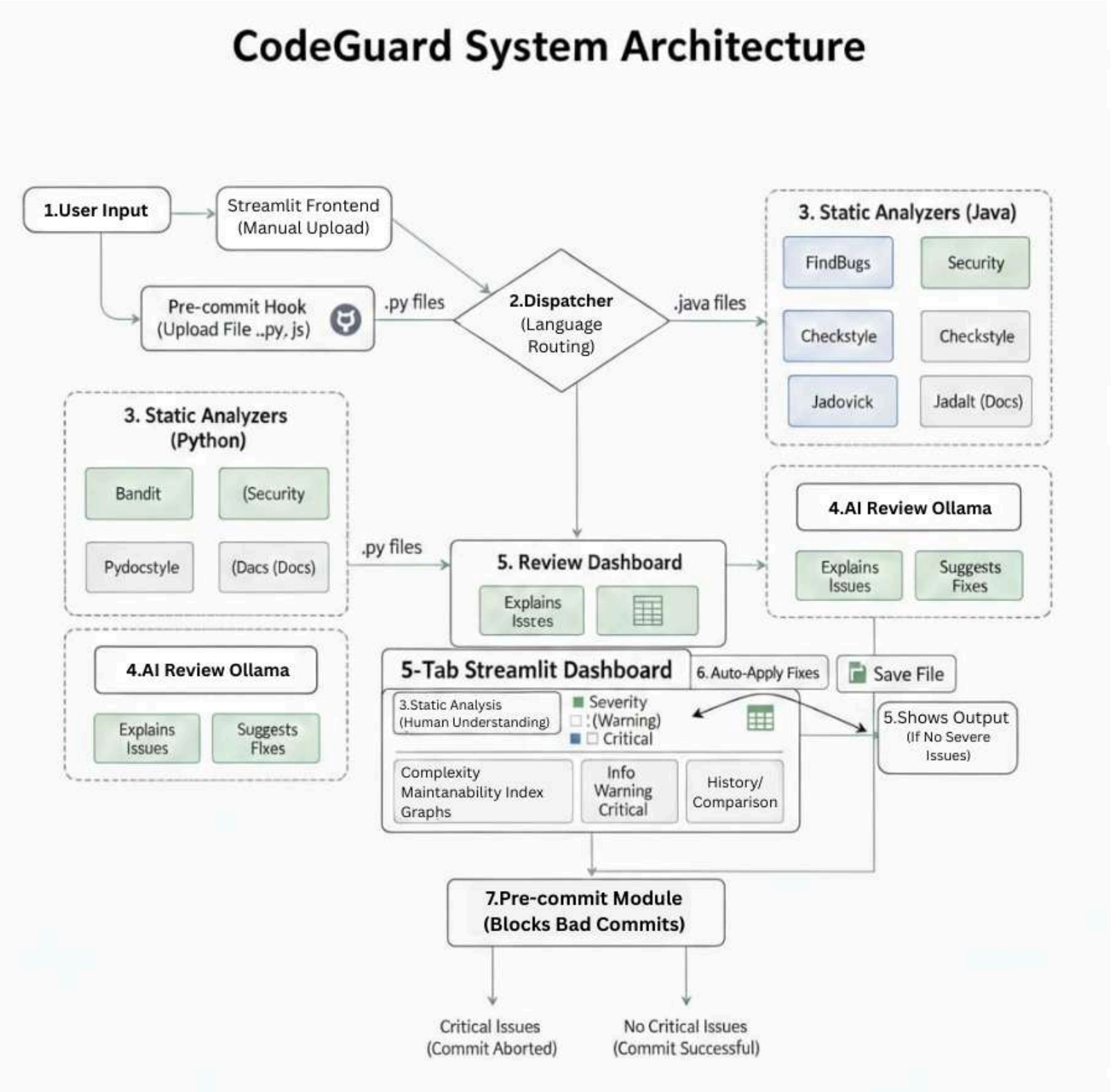
CLI commands for scanning, reviewing, applying fixes, reporting, plus a Streamlit dashboard for interactive visualization.



## Configurability

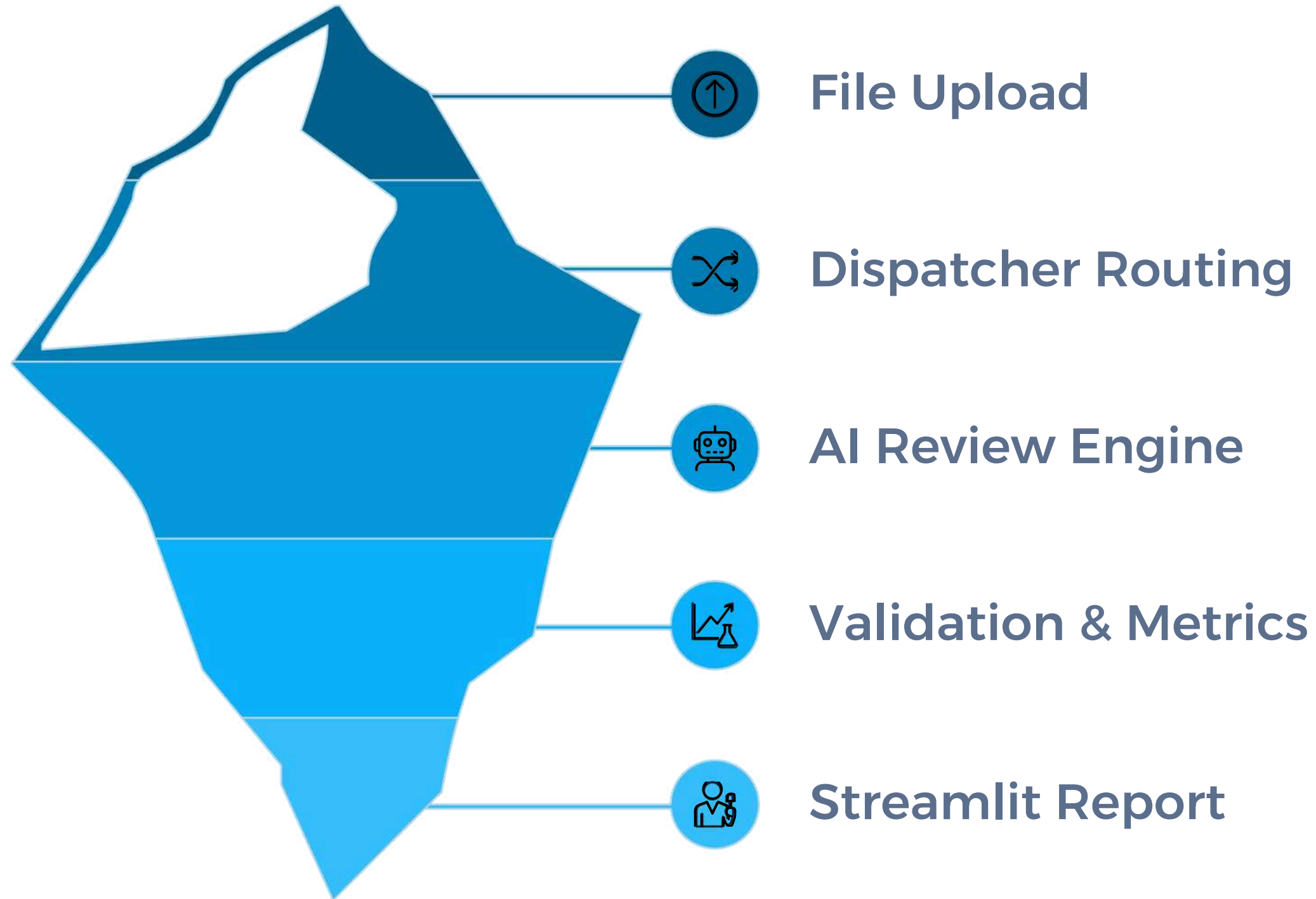
Seamless integration with Git workflows (pre-commit hooks, CI/CD pipelines), configurable rules (PEP8, custom checks, severity thresholds), and outputs including docstring coverage, maintainability index, and complexity metrics..

# SYSTEM ARCHITECTURE



- **Input Layer:** Code files uploaded via Streamlit frontend or Git pre-commit hooks.
- **Processing Layer:** Dispatcher routes files to language-specific static analysers (Python: Bandit, Pydocstyle; Java: FindBugs, Checkstyle, etc.) and AI review engine for explanations and fix suggestions.
- **Output Layer:** Streamlit dashboard visualizes issues (severity, complexity, maintainability), supports interactive review, auto-apply fixes, and enforces quality gates through pre-commit/CI pipelines.

# WORKFLOW



Each run: parse AST, run deterministic checks, enrich findings with LLM explanations, propose patch, run safety validation and present an actionable report to the developer.

## AST Parser 🏠

### Module 1: Code Parsing & Analysis

#### AST Parser ➤

```
{ Functions; function /;  
1 { Classes  
2   classerfalls;  
3   Functions  
18 Imports  
46 Imports:  
16 Long Functions  
16. Missing Type Hints ✓}
```

Static  
Analysis

## Static Analysis

Code Smell

Code Smells  
Moderate attention

🚩 Code Smells  
● Long Functions or  
Missing Type Hints

Extractive: Constituted: ➤

<ix:

```
Functions functions;  
Classes, Missing Functions;  
Eart. Long functions  
Missin type Hints;
```

# MODULE 1 : CODE PARSING & ANALYSIS

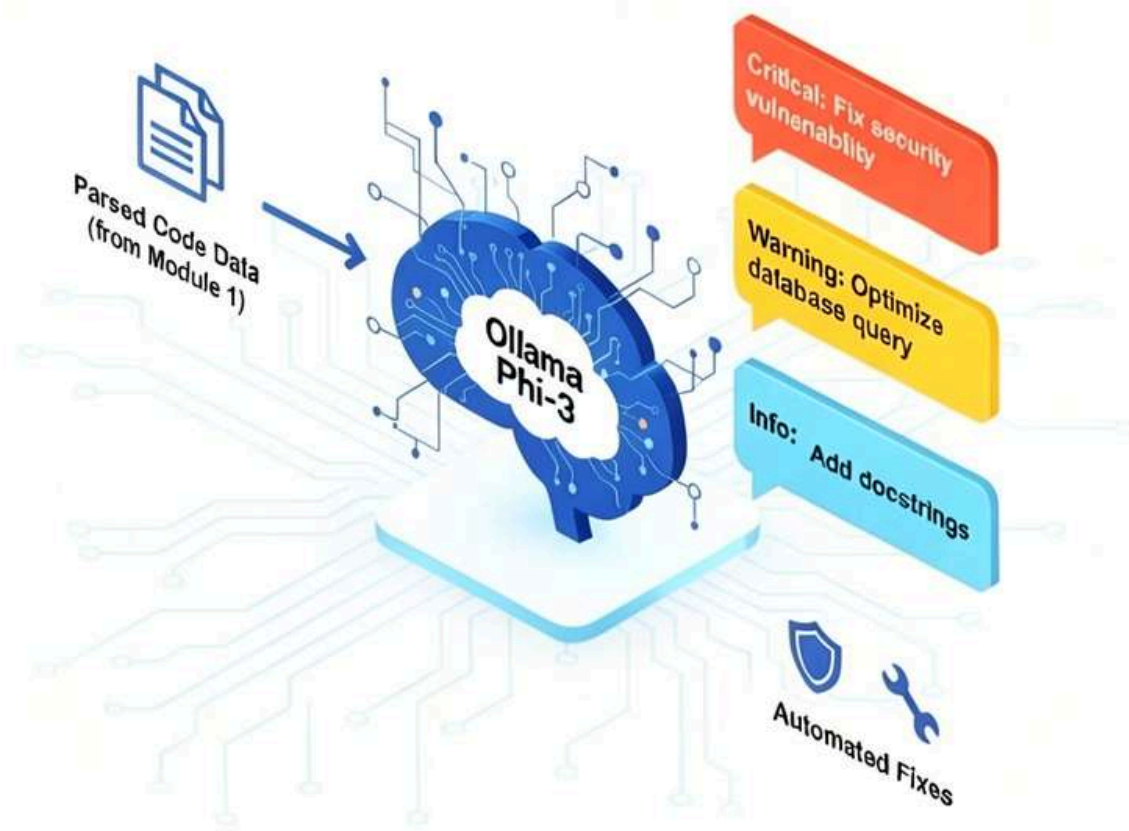
AST-driven analysis for Python and pluggable detectors for other languages.  
Detects complexity, style, security, and documentation issues.

- High cyclomatic complexity
- Long functions, missing docstrings & type hints
- Naming violations, hardcoded secrets
- Unsafe eval/exec patterns

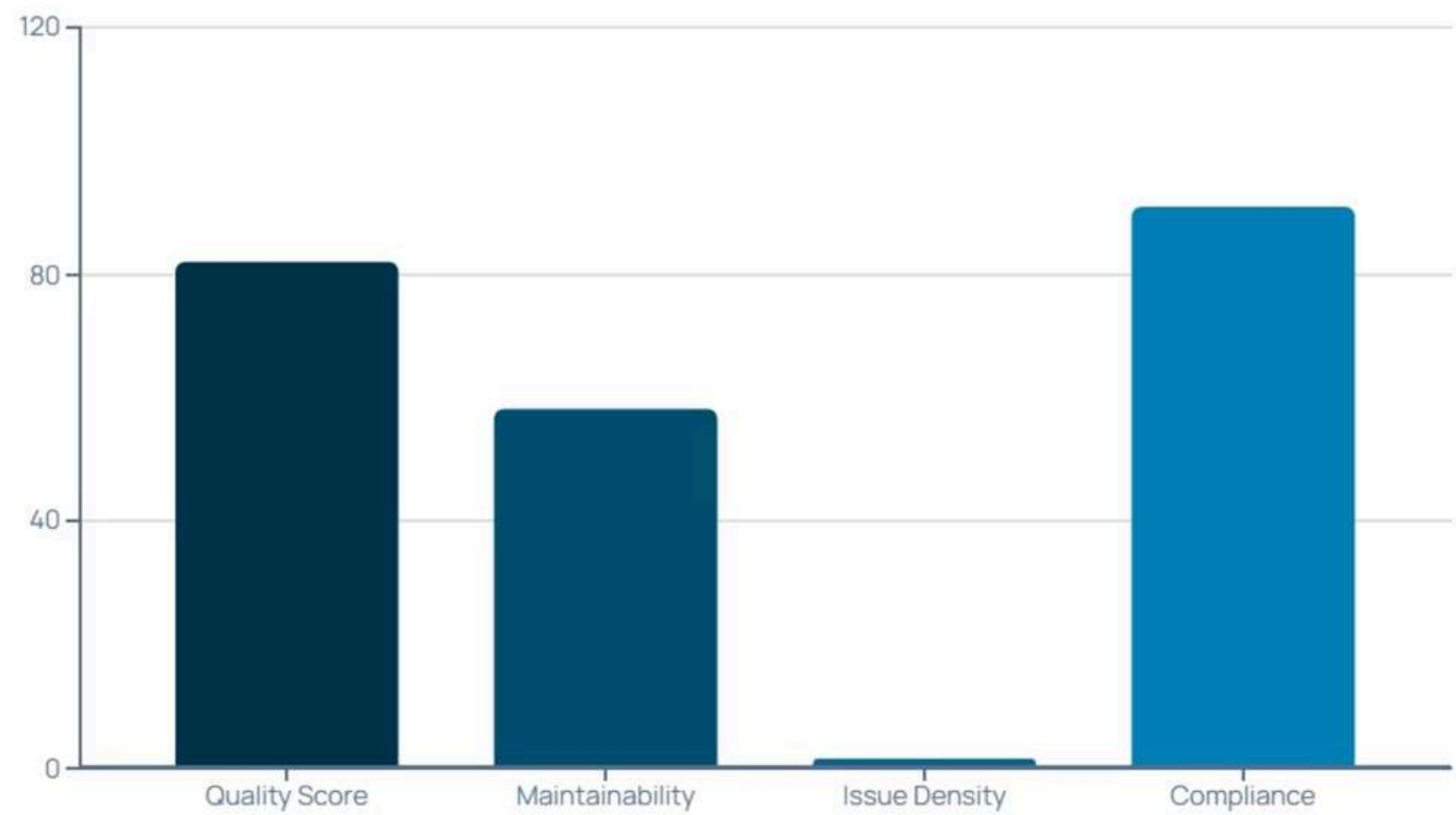
Dispatcher enables language-specific analyzers and unified issue format.

## MODULE 2 : AI REVIEW ENGINE

- Uses Ollama Phi-3 model with prompt templates to generate human-like code reviews.
- Classifies findings by severity: Info, Warning, Critical.
- Explains issues in natural language for easy understanding.
- Automatically corrects simple issues like naming, docstrings, and spacing.
- Compatible with CLI, Streamlit UI, and Git workflow.



# MODULE 3 : VALIDATION & METRICS



- Tracks key indicators: Quality Score, Maintainability Index, Issue Density, Severity breakdown.
- Enforces quality gates with thresholds (Score  $\geq 70$ , MI  $\geq 50$ ).
- Identifies best and worst files for targeted improvements.
- Uses severity-weighted scoring to prioritize issues.
- Generates exportable reports (CSV/HTML; planned JSON/ZIP) for easy sharing.

# MODULE 4 – CLI & CONFIGURATION

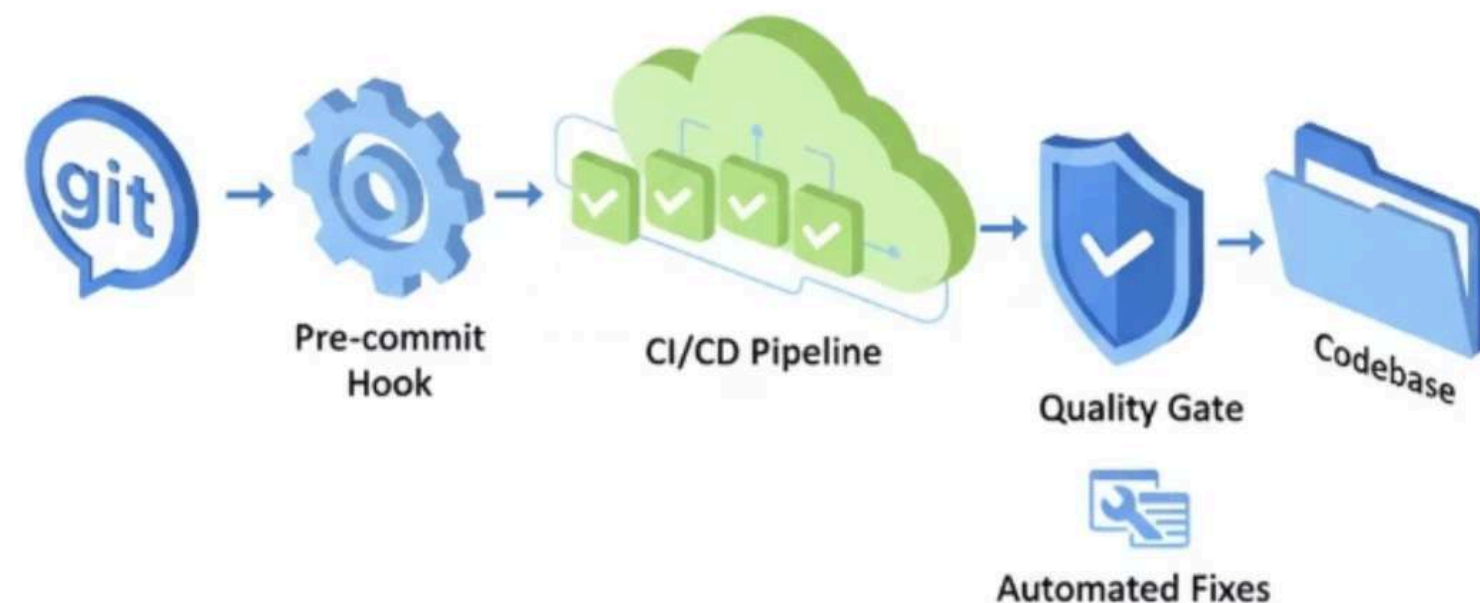
## Command Set

- scan — initiates static analysis of code.
- review — triggers AI-powered explanations and feedback.
- apply — automatically applies suggested code fixes.
- report — generates comprehensive review artifacts.
- diff — displays proposed code changes in detail.

## Capabilities

- Configurable rules via pyproject.toml (PEP8, custom checks, severity thresholds, excluded paths)
- Flexible severity control: block commits on critical issues, warn on lower-priority findings
- Supports exclusions for specific files or directories
- Seamless Git integration for smooth developer workflows

# Module 5 : VCS & CI Integration

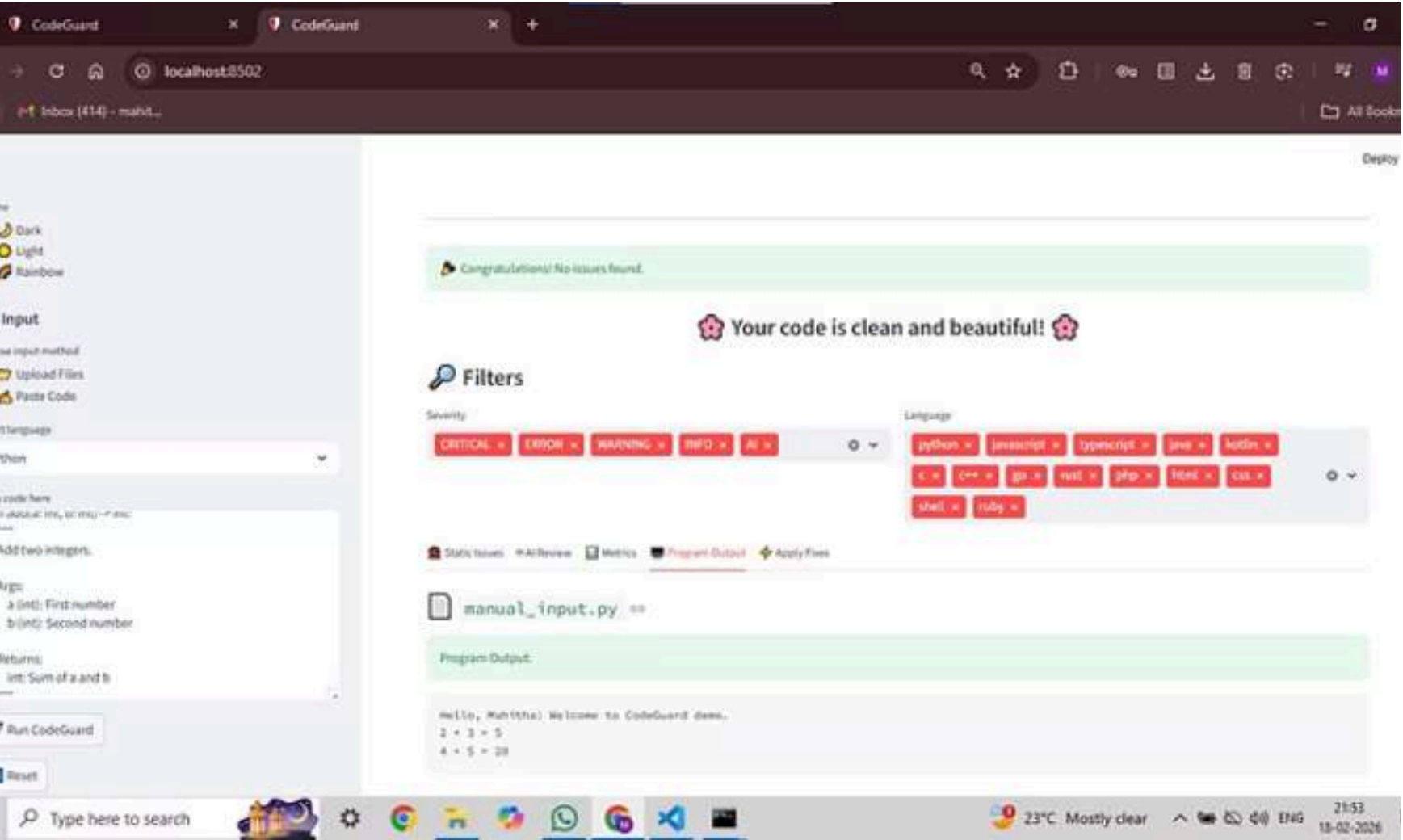


- Pre-commit hook reviews staged files before commit; CI/CD templates enforce quality gates in pipelines
- Fast execution (<5s on staged changes) ensures smooth developer workflow without friction
- Coverage thresholds (e.g.,  $\geq 90\%$ ) block builds if standards are not met
- Seamless integration with GitHub/GitLab workflows for consistent code quality.

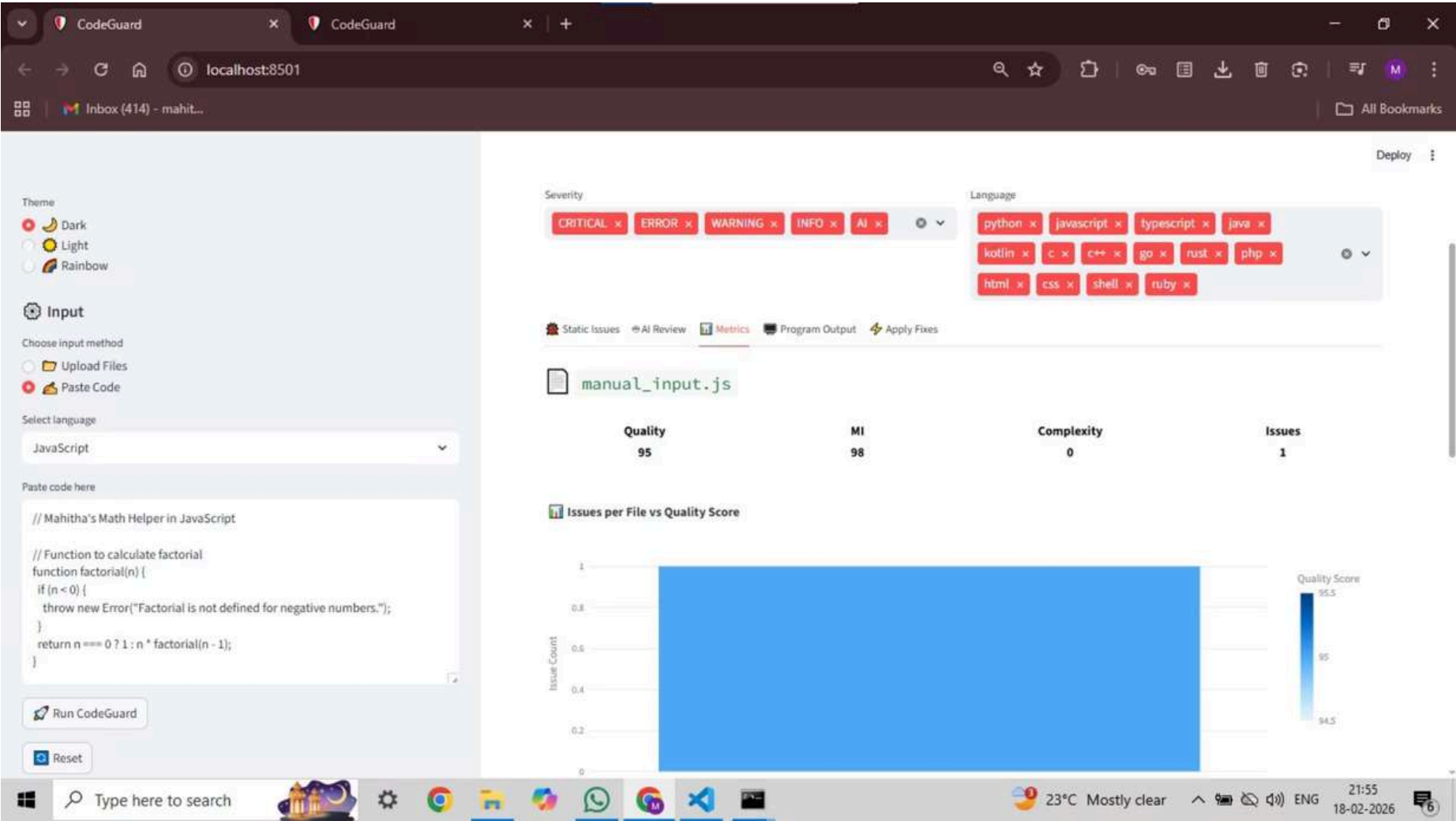
# MODULE 6 : STREAMLIT WEB UI

- Interactive dashboard to visualize issues and fixes
- Side-by-side diff view with AI suggestions for easy comparison
- Preview, accept, or reject fixes directly in the interface
- Enhanced usability with filters, search, and tooltips (planned features)
- Fast performance (<5s review speed on staged changes)

# PERFORMANCE & RESULTS

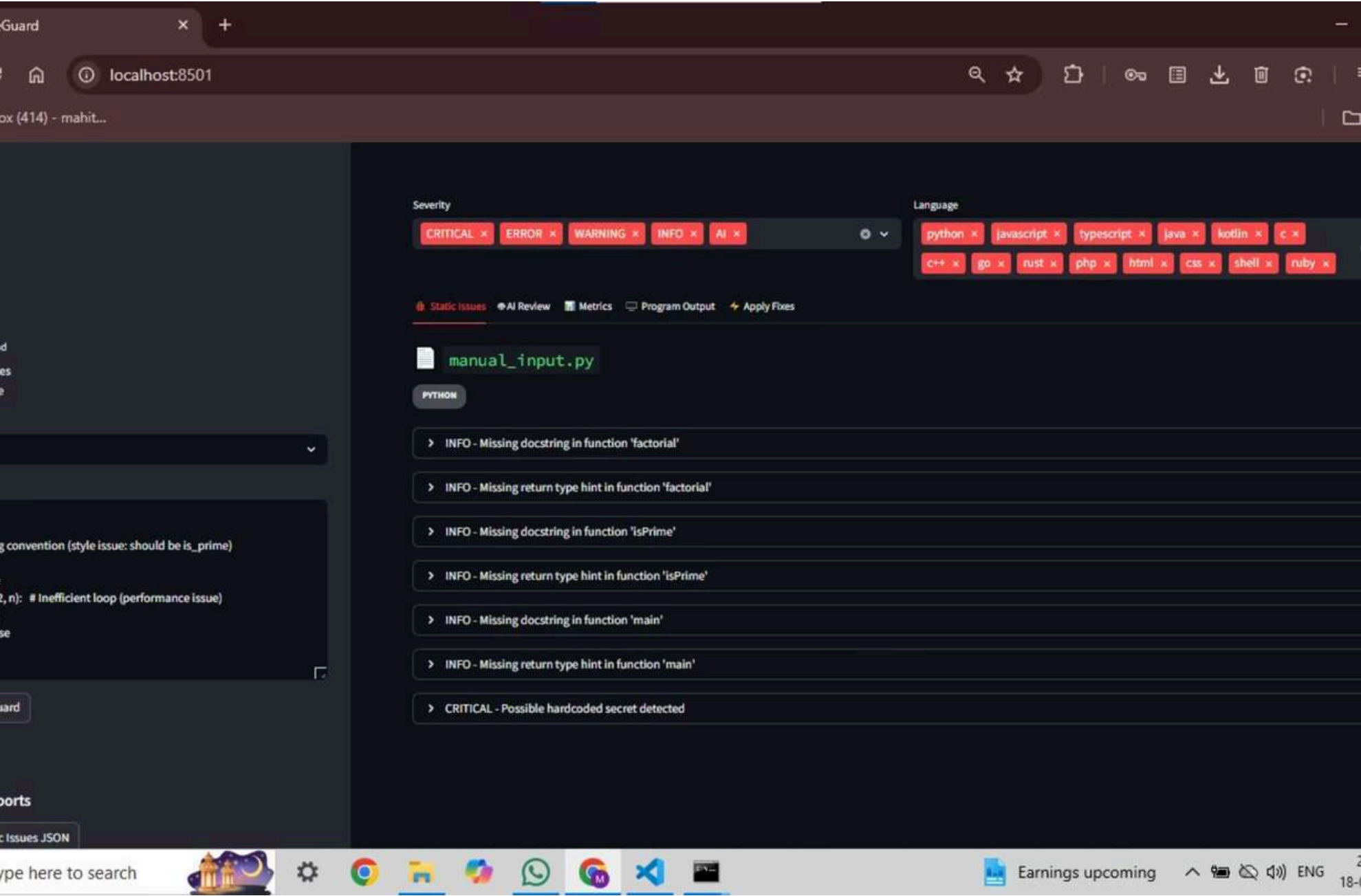


CodeGuard confirms no issues — clean, documented, and standards-compliant code.

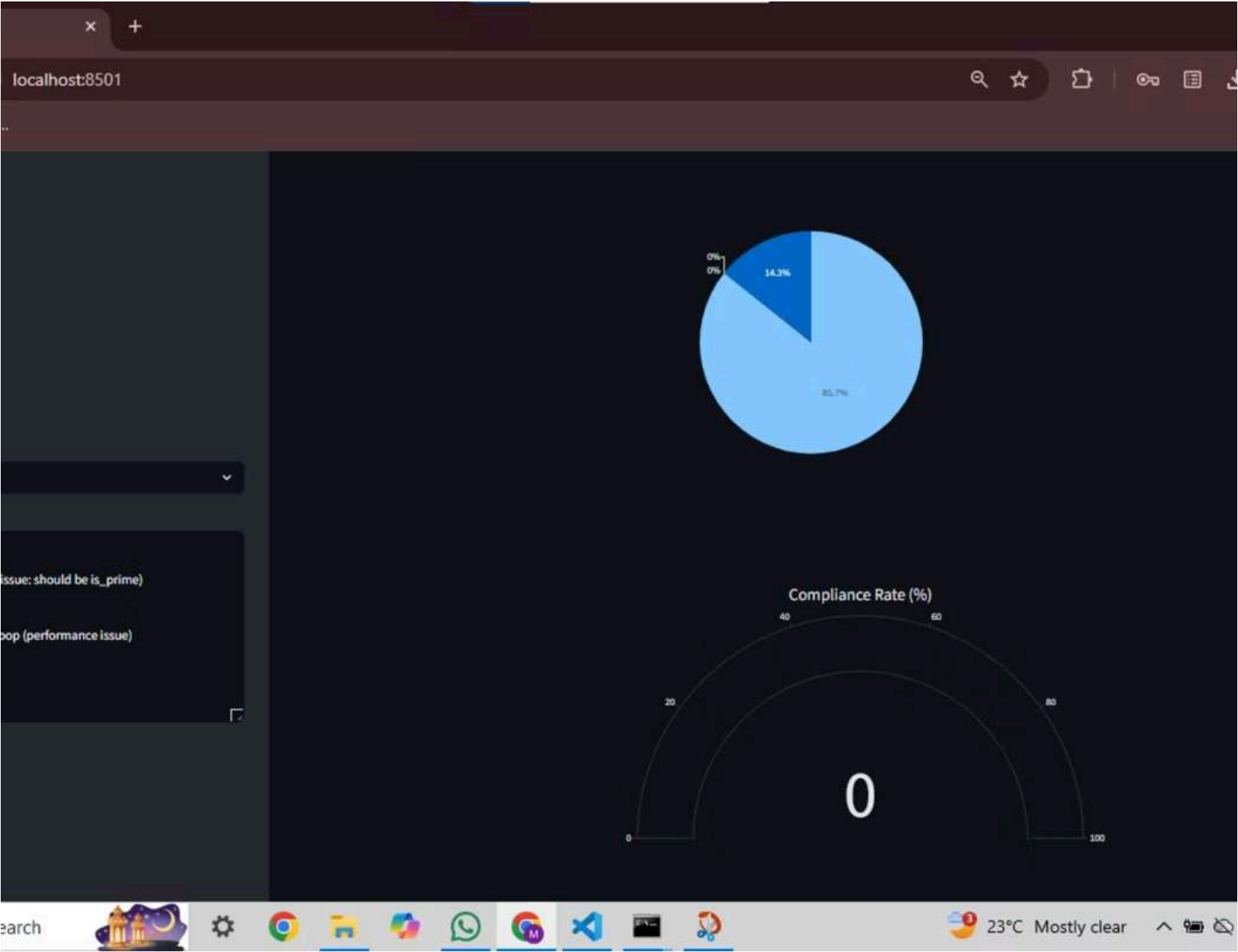


Metrics dashboard highlights Quality Score, Maintainability Index, Complexity, and issue count with clear visualizations for per-file analysis.

# PERFORMANCE & RESULTS

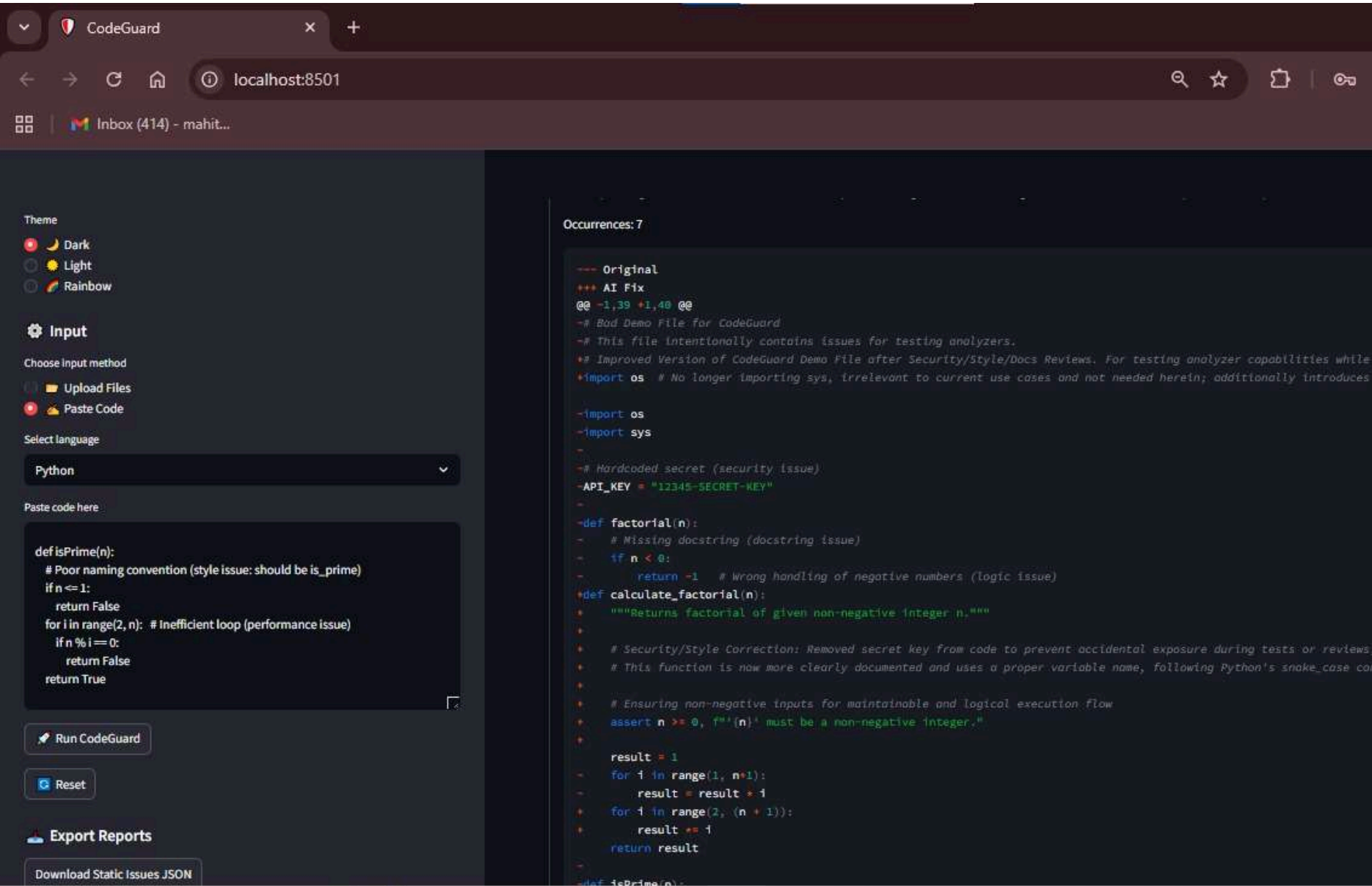
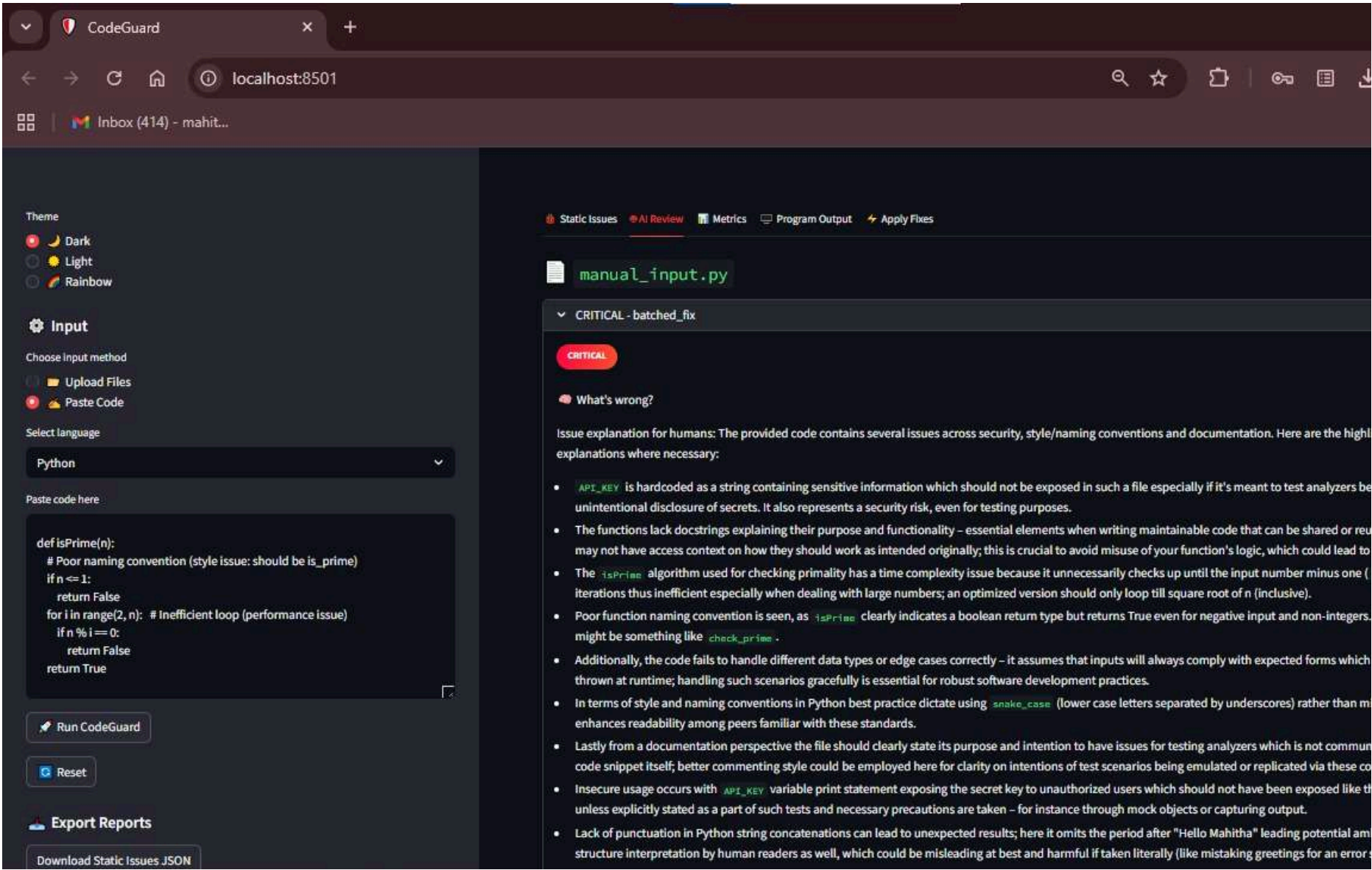


Static analysis highlights missing docstrings, type hints, and critical security risks, enabling developers to address style and documentation gaps alongside severe issues.



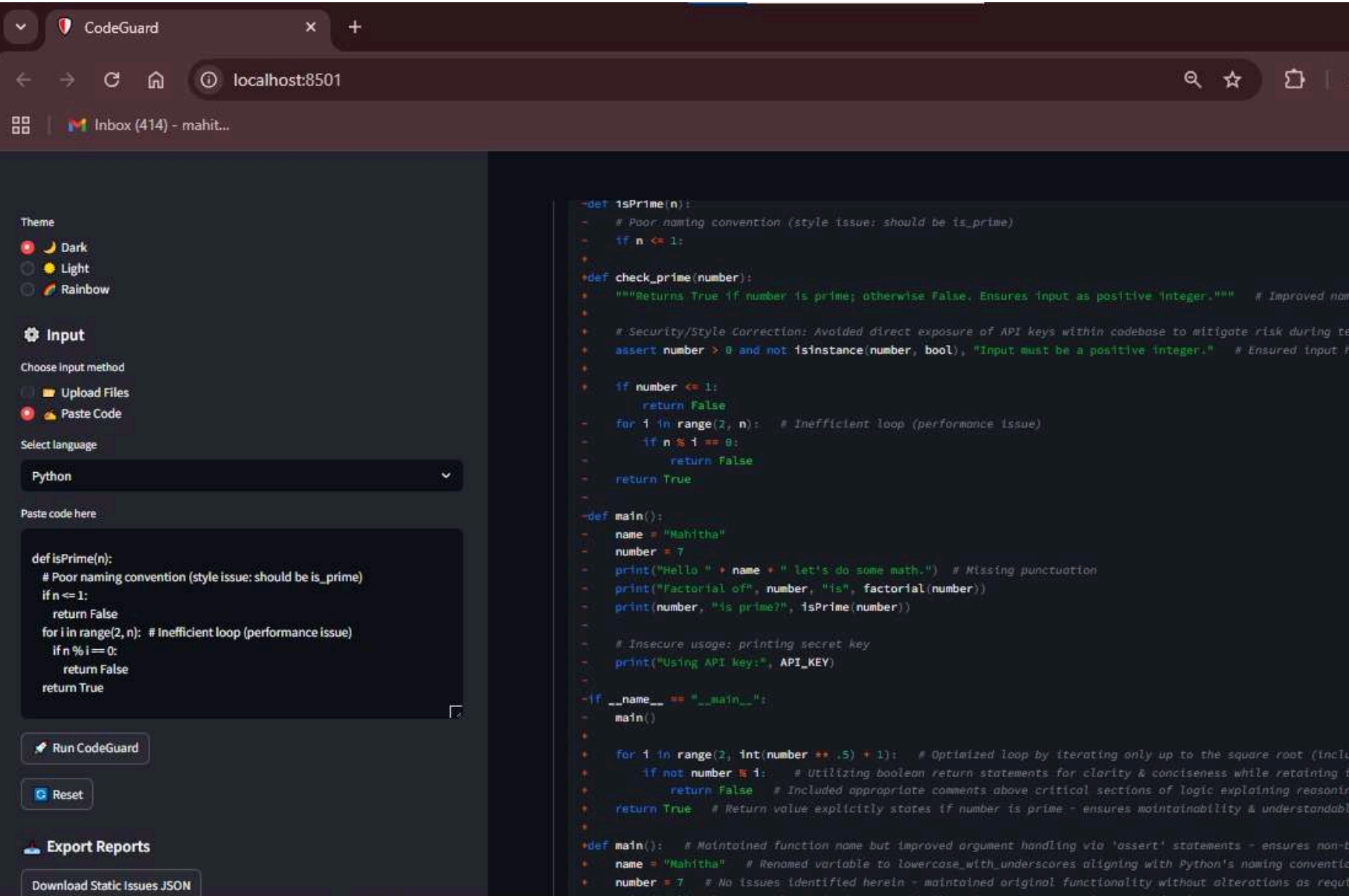
Compliance dashboard visualizes issue severity distribution with pie chart and gauge, highlighting adherence gaps in coding standards.

# PERFORMANCE & RESULTS

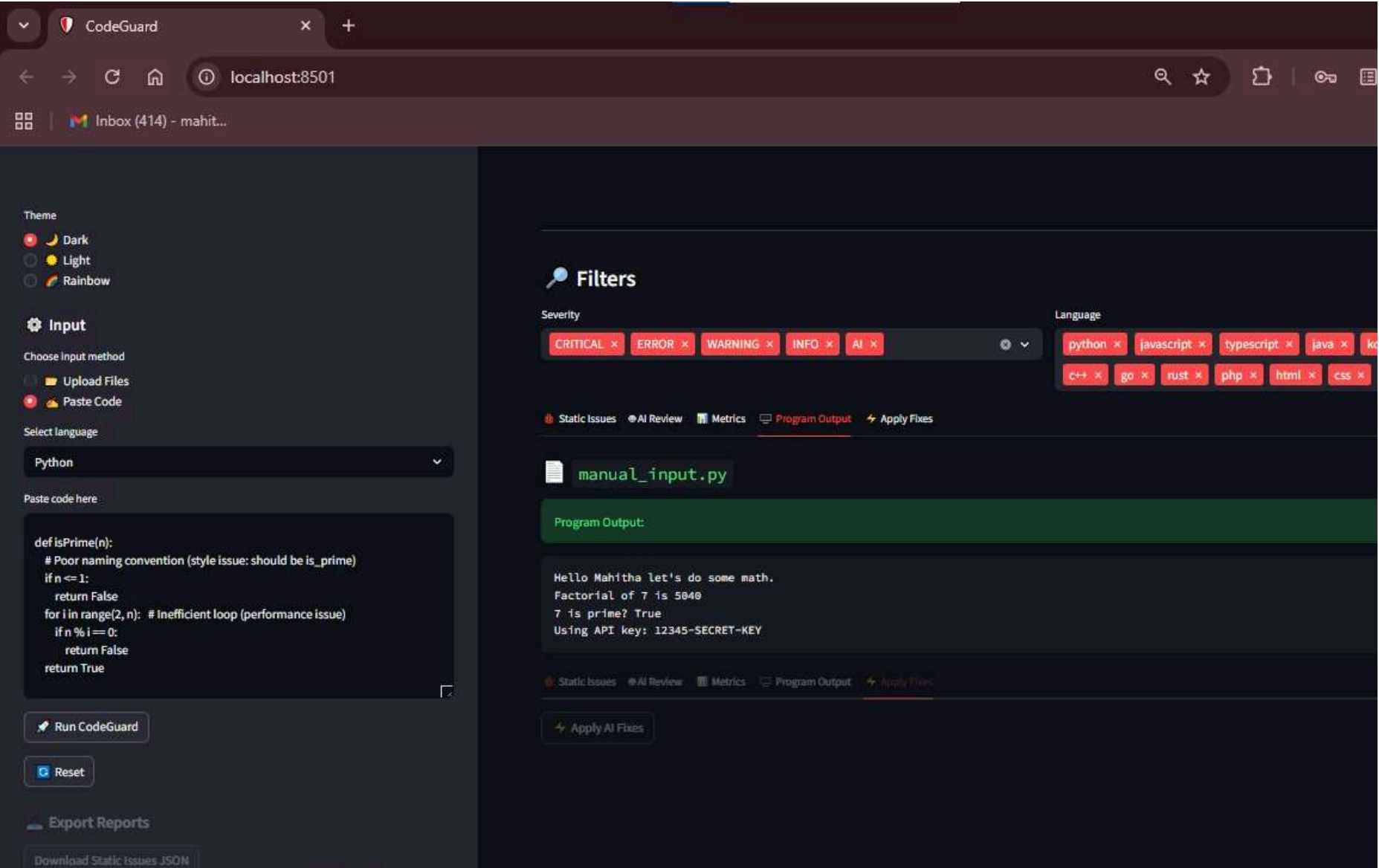


AI-assisted fixes remove hardcoded secrets, improve naming conventions, and enhance documentation, producing secure and maintainable code.”

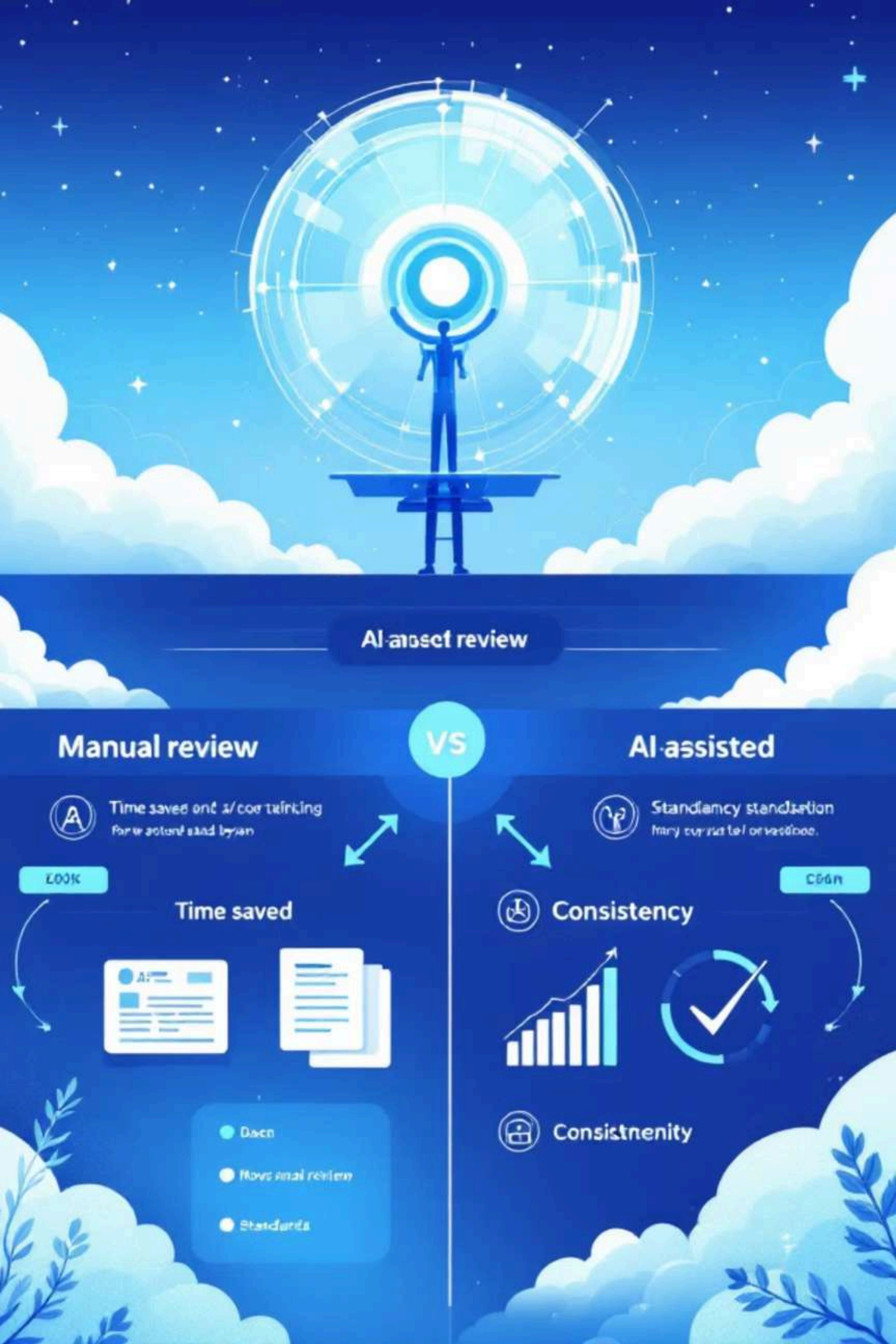
# PERFORMANCE & RESULTS



AI-assisted fixes remove hardcoded secrets, improve naming conventions, and enhance documentation, producing secure and maintainable code.”



Execution output reveals program results alongside static analysis filters, demonstrating detection of style, performance, and security issues across multiple languages.



# COMPARISON: MANUAL VS AI-ASSISTED REVIEW



## Turnaround Time

Manual: hours–days. CodeGuard: minutes with immediate feedback and gating.



## Consistency

Manual results vary by reviewer. AI enforces deterministic rules and documented rationale.



## Quality Assurance

Automated severity ranking reduces human oversight errors and prevents regressions via CI gates.



## Scalability

AI-assisted scans scale across repositories and languages with plugin analyzers and LLM prompting.

# FUTURE ENHANCEMENTS

## Multi-language support

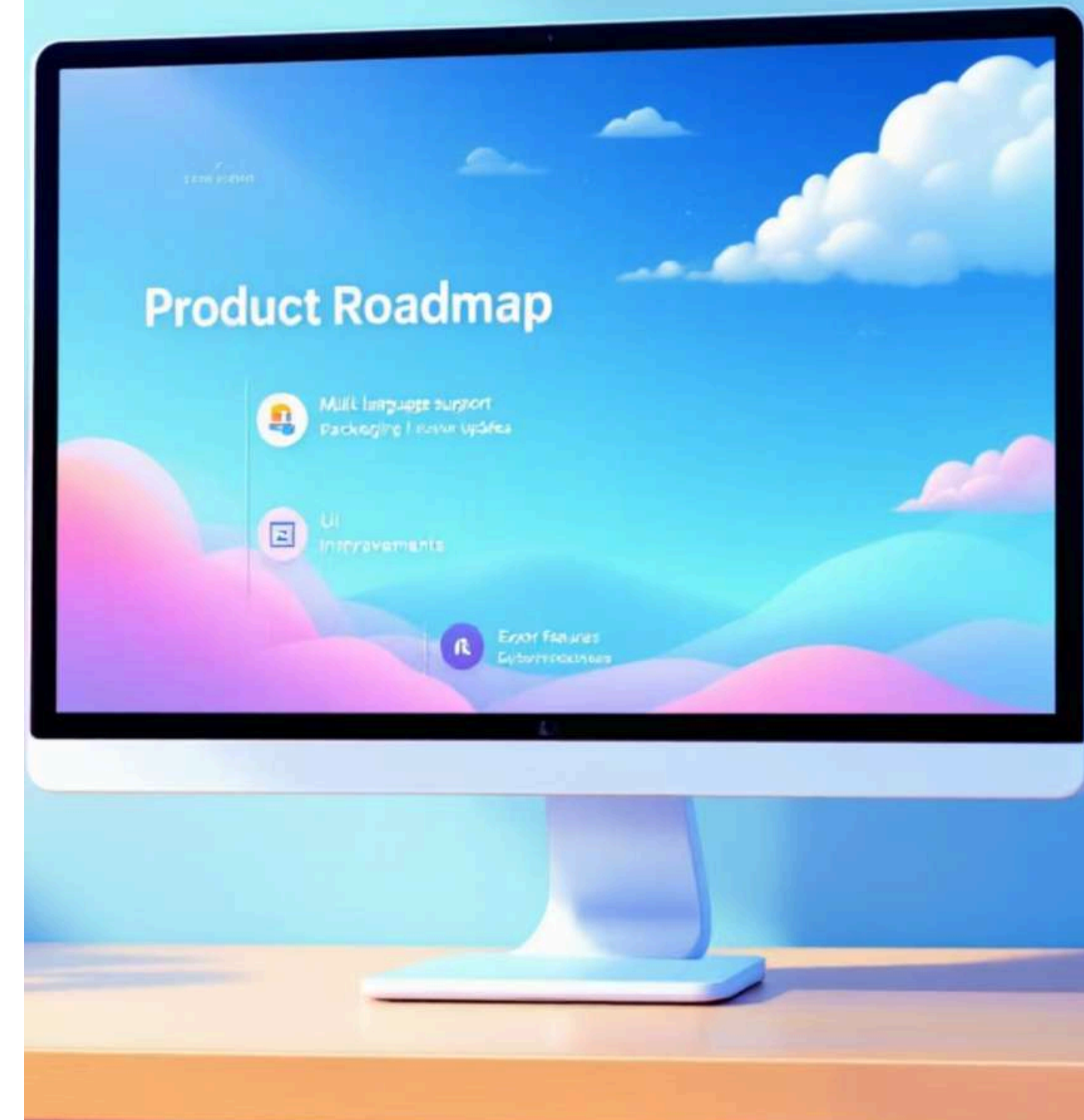
Extending language-specific environments for fixes across Java, C++, and other multi- languages.

## Advanced Streamlit UI

Filters, search, tooltips, audit trails and collaborative review modes.

## Packaging & Exports

Publish as pip package, add CSV/HTML report exports and CI integrations for enterprise pipelines.



# CONCLUSION

CodeGuard automates code quality by accelerating reviews, ensuring consistency, and detecting security and maintainability issues at an early stage. With seamless integration into Git workflows, it enforces quality gates without disrupting developer productivity. The system is production-ready, robust, and designed with extensibility in mind, making it well-suited for future enhancements and broader language support.

# Thank You

CodeGuard — Reliable, Secure, Future-Ready.

**Submitted By: TEAM 1**

**Mahitha Chippada**

**Sahasra Doddi**

**Lakshmi Priya Vuyyuru**

**Rishitha Rallapalli**

**Jyotsnamathi J**

**Thottempudi Venkat Sainadh**

**Disha R Gowda**