

AI-Powered Code Reviewer & Quality Analyzer



Problem Statement & Objectives

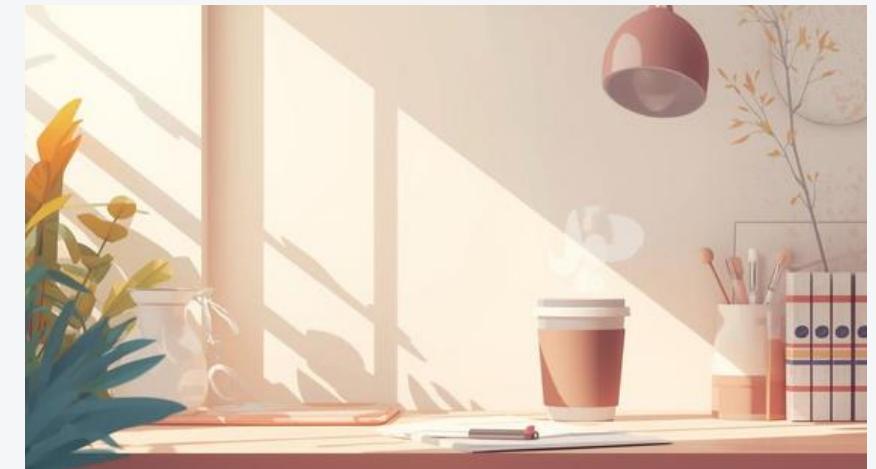
- Manual code reviews are time-consuming processes that require significant effort, leading to inconsistencies in quality and delaying project timelines as reviewers may have differing standards and approaches.
- Poor or missing documentation reduces **code readability, maintainability, and overall developer productivity**.
- Early-stage projects often lack **automated quality validation**, leading to technical debt.
- Existing tools focus mainly on **syntax and linting**, not on **docstring quality**.



Core Challenge:

Ensuring consistent, high-quality documentation and documentation standards across large codebases at scale.

Key Project Objectives for AI Assistant



Automate

Streamline documentation tasks for efficiency and accuracy.

Improve

Enhance readability and consistency across codebases effectively.

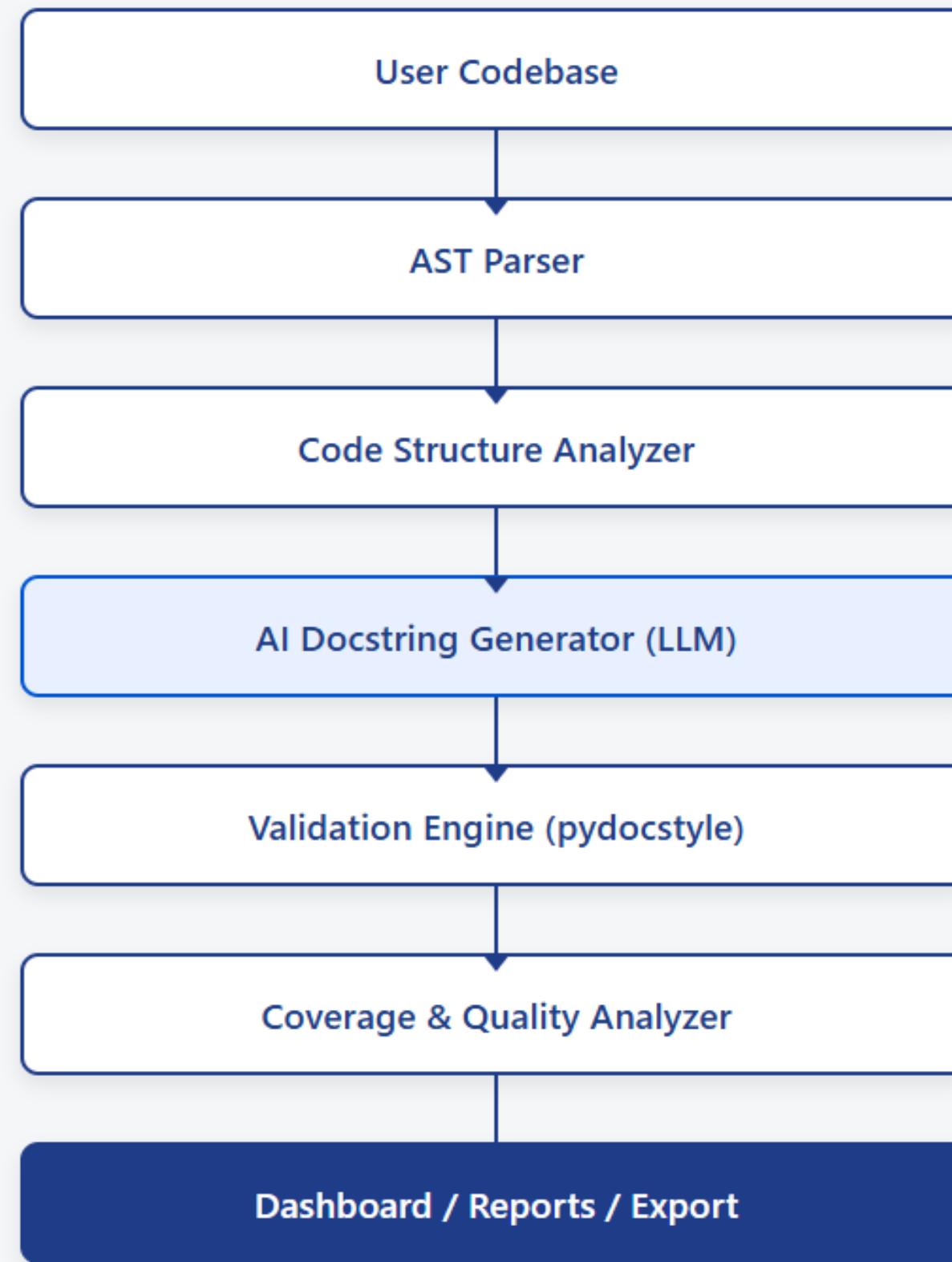
Enforce

Ensure adherence to PEP 257 documentation standards consistently.

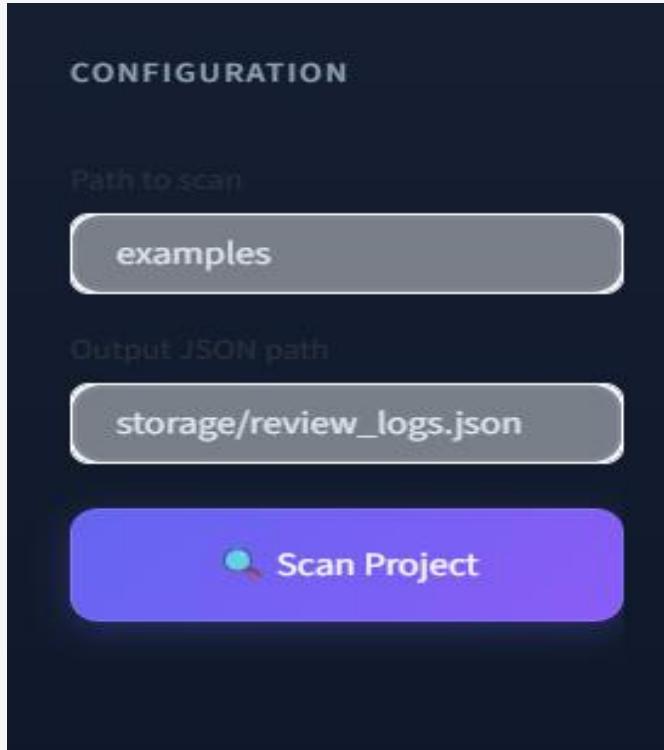
Reduce

Minimize manual review time and enhance productivity.

System Architecture Overview

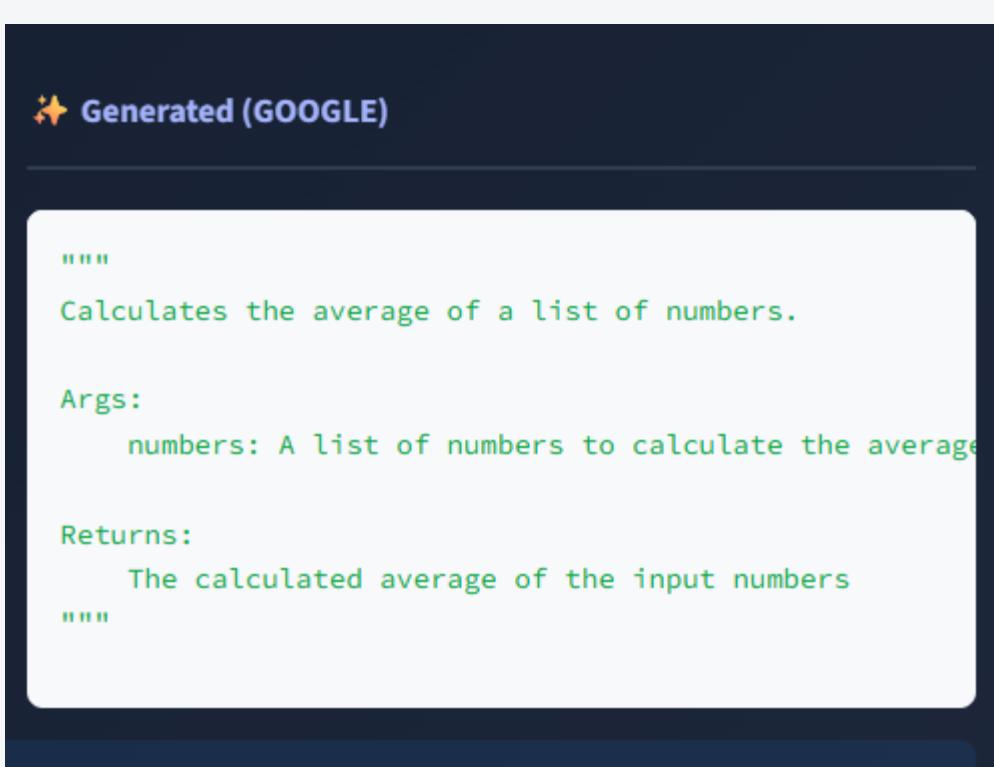


Core Modules Overview



Code Parsing

Extracts code structure for analysis and validation.



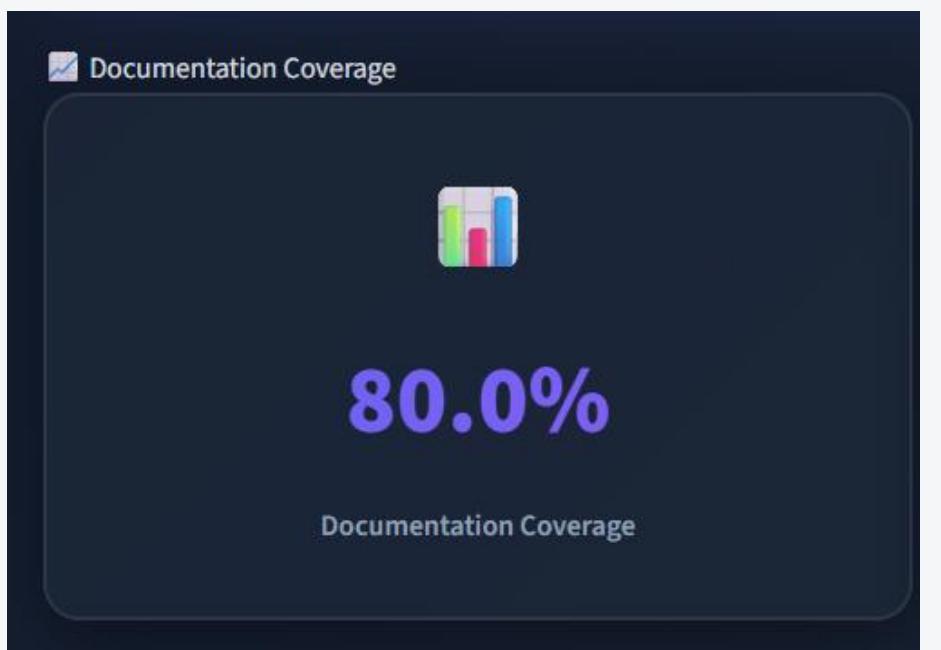
Documentation

Generates compliant docstrings from the code analysis.



Validation

Enforces compliance with established documentation standards.



Coverage

Measures and ensures Documentation completeness and quality.

Dashboard

Advanced tools for code analysis and management

Dashboard Navigation

Filters Search Tests Export Help

Filter Functions by Docstring Status

All Functions Has Docstring Missing Docstring

Dashboard

Visualizes metrics and offers insights for developers.

A1	X	✓	fx	File
1	File	Function	Has Docst	Complexity
2	sample_a.calculate_	No		3
3	sample_a.add	Yes		1
4	sample_a.Processor	Yes		3
5	sample_b.generator	Yes		2
6	sample_b.raises_ex	Yes		2
7				
8				

Export

Outputs data in formats for integration into workflows.

Technology Stack Overview

Programming Language

Python

Code Parsing

AST (Abstract Syntax Tree)

AI Engine

LLM (Groq Integration)

Validation

pydocstyle (PEP 257)

UI Framework

Streamlit

Testing

pytest

Reporting

JSON, CSV Export

Deployment Ready

CLI + Web Dashboard

Future Integration

CI/CD, Git Hooks

Platform Walkthrough

The platform provides a centralized dashboard to analyze documentation coverage, review AI-generated docstrings, validate standards compliance, and monitor quality metrics. Developers can search and filter code elements, accept or reject AI suggestions, and track validation results through an intuitive navigation workflow.

The screenshot displays the AI-Powered Code Reviewer dashboard. At the top center, there is a purple header with the text "AI-Powered Code Reviewer" and "Intelligent documentation analysis and generation". To the left, a sidebar titled "AI Code Reviewer" contains a "Close Menu" button and a "Home" button highlighted in purple. Below the sidebar, the "NAVIGATION" section lists "Documentation Coverage", "Metrics", and "Validation". The main content area features three cards: "Documentation Coverage" (80.0%), "Total Functions" (5), and "Documented" (4).

Category	Value
Documentation Coverage	80.0%
Total Functions	5
Documented	4

Automated Testing Overview

- Comprehensive automated testing was conducted using **pytest**
- Tests cover all major functional components of the system
- Ensures correctness, stability, and production readiness



Results & Impact

Key Benefits Achieved by the System



Faster Development

Automated code review and documentation generation significantly reduce manual effort, enabling developers to focus on feature development and faster delivery.



Higher Code Quality

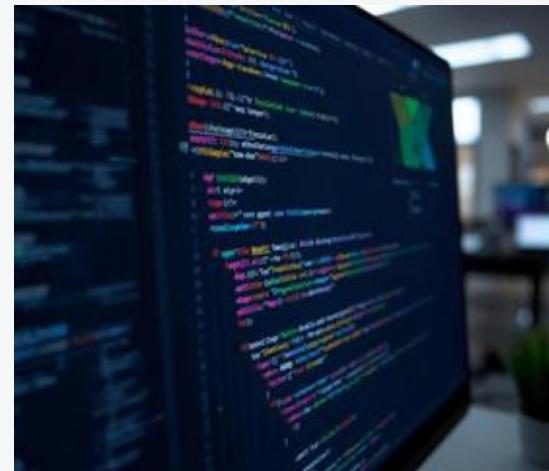
Enforcing documentation standards and providing actionable insights improves code readability, maintainability, and long-term project quality.



Consistency & Standard Compliance

Ensures PEP 257-compliant, uniform docstring styles across the codebase, reducing inconsistencies and simplifying future maintenance.

Future Enhancements for Our AI Assistant



Code Improvements

Suggestions to enhance code quality and efficiency.

Auto-Fixing

Automatically resolve documentation problems to save time.

Git Integration

Seamlessly integrate with Git for continuous improvement.

Collaboration

Enhance teamwork capabilities for project development.

Packaging

Make the tool easily installable for developers.

Advanced Analytics

Monitor trends and metrics for continuous improvement.

Thank You

-Hema Latha Thota