AI-Powered Code Reviewer & Quality Assistant

Automated Code Quality Analysis using AI & Static Analysis

KARISHMA B

Project Statement: Elevating Code Quality

Modern software projects evolve at a rapid pace, frequently lacking consistent review quality or standardised coding practices. Manual code reviews are inherently time-consuming, subjective, and heavily dependent on the individual expertise of the reviewer.

> This project introduces an AI-Powered Code Reviewer & Quality Assistant designed to automatically analyse Python code.

Key Areas of Analysis:

- Code quality and adherence to best practices
- Documentation consistency and completeness
- Code complexity and maintainability scores
- Identification of potential bugs and anti-patterns

By integrating static analysis, Large Language Models (LLMs), and Git workflows, this tool provides actionable, human-like feedback via a Command Line Interface (CLI) and an optional Streamlit User Interface.

Motivation & Business Need

1

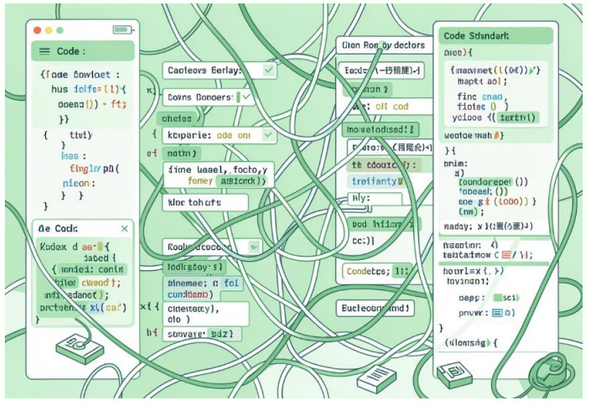
Scaling Challenges

Manual code reviews do not scale effectively for large or rapidly expanding codebases, leading to bottlenecks and delays.

2

Inconsistent Standards

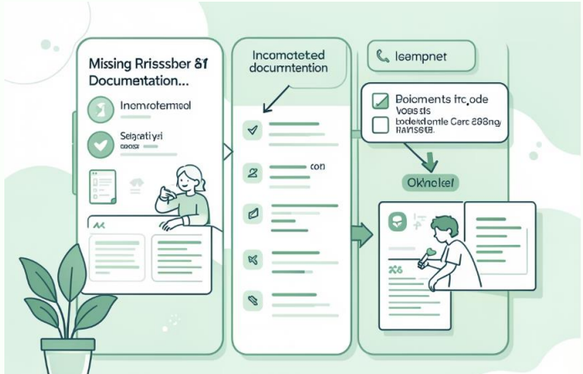
Inconsistent enforcement of PEP standards and coding guidelines across different team members or projects.



3

Documentation Gaps

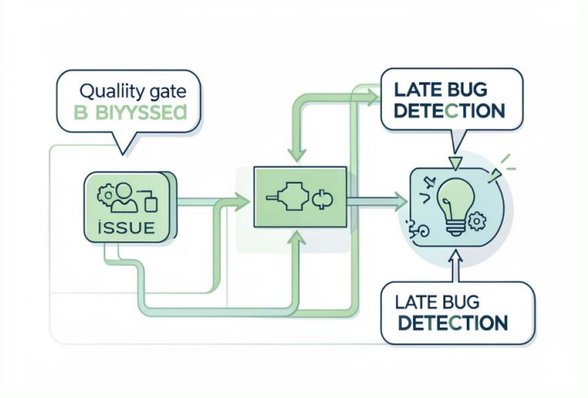
Developers often skip crucial documentation due to tight deadlines or time constraints, impacting future maintainability.



4

Quality Lag

CI pipelines typically catch syntax errors but often miss deeper code quality and design issues until later stages.



Expected Outcomes & Deliverables

AI-Assisted CLI Tool

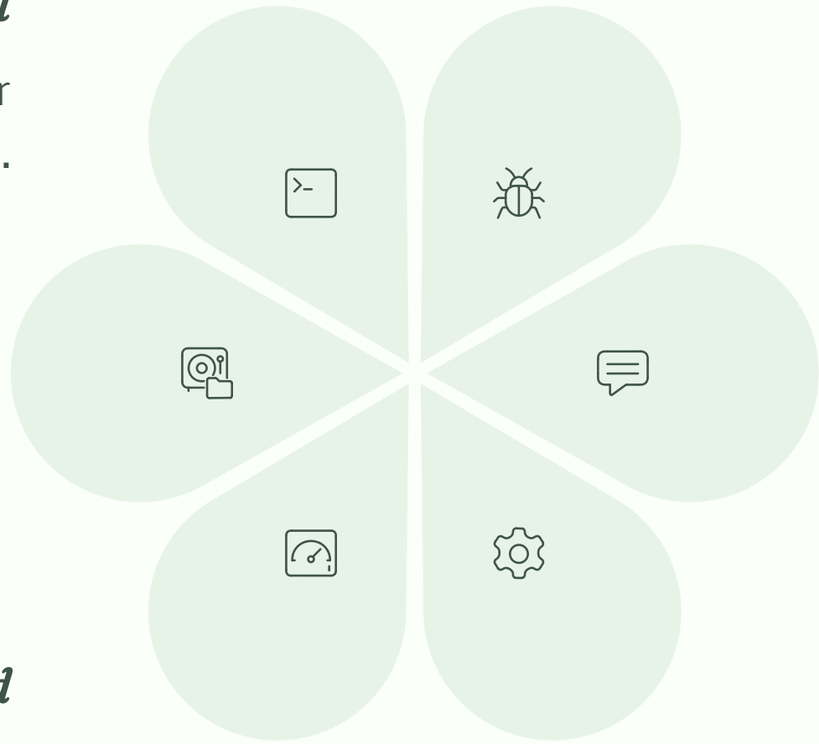
A powerful command-line interface for Python code reviews.

Git & CI/CD Integration

Seamless integration with Git pre-commit hooks and CI/CD pipelines for automated enforcement.

Optional Streamlit Dashboard

Interactive UI for review, approval, and visualisation of findings.



Comprehensive Issue Detection

Identifies unused imports, high cyclomatic complexity, missing docstrings, and poor naming conventions.

LLM-Powered Suggestions

Natural-language suggestions and explanations for identified issues.

Configurable Rules

Supports PEP-8, PEP-257, and custom coding standards.

Milestone Overview

The AI-Powered Code Reviewer & Quality Assistant is structured into six key, interconnected modules:

1. *Code Parsing & Analysis*
2. *AI Review Engine*
3. *Validation & Metrics*
4. *CLI & Configuration*
5. *VCS & CI Integration*
6. *Review Web UI*

Milestone 1: Code Parsing & Static Analysis

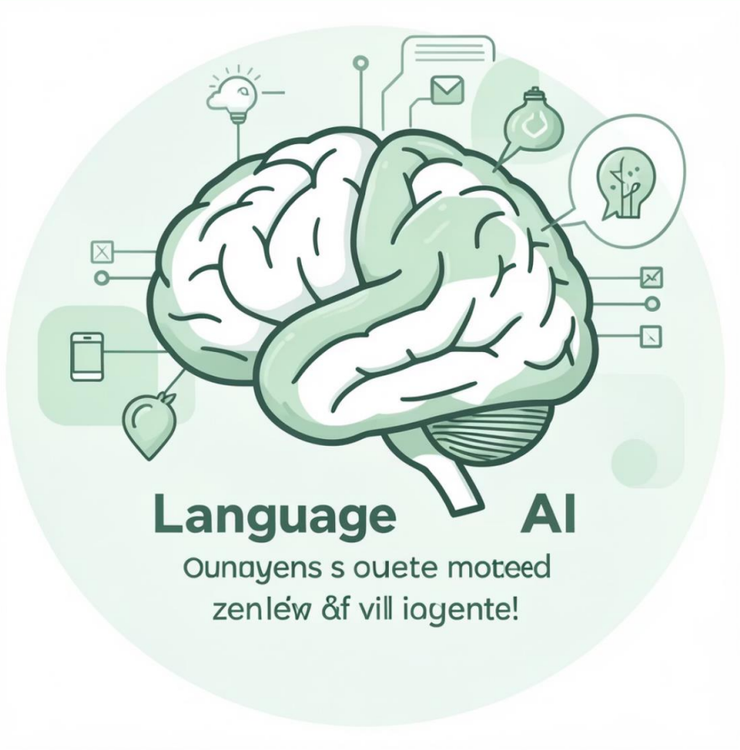
This foundational module is responsible for dissecting the Python source code to understand its structure and properties.

Key Functions:

- **AST Parsing:** Utilises the Abstract Syntax Tree (AST) to parse Python source files, enabling a deep structural understanding of the code.
- **Information Extraction:** Extracts critical elements such as imports, classes, functions, methods, arguments, and return types.
- **Metric Computation:** Computes essential code quality metrics like Cyclomatic Complexity and Maintainability Index.
- **Code Smell Detection:** Identifies common code smells, including long functions, deep nesting, missing docstrings, and missing type hints.

By performing a thorough static analysis, this module lays the groundwork for intelligent review by the AI engine.

Milestone 2: AI Review Engine



The core intelligence of the system, this module leverages Large Language Models (LLMs) to provide human-like feedback.

How it Works:

- **LLM-based Prompt Templates:** Employs sophisticated prompt templates to guide the LLM in generating relevant and contextual review comments.
- **Human-like Feedback:** Generates clear, concise, and actionable review comments with detailed explanations for identified issues.
- **Severity Categorisation:** Automatically categorises issues into severity levels: ℹ️ Info, ⚠️ Warning, and 🔴 Critical.
- **Suggested Fixes:** Offers intelligent suggestions for improvements related to docstrings, naming conventions, and code formatting.
- **Preview-Only Workflow:** All suggestions are provided in a preview-only mode, ensuring a safe and non-intrusive workflow until approved by the developer.

This module transforms raw static analysis findings into intelligent, actionable recommendations.

Milestone 3: Validation & Quality Metrics

PEP-257

Docstring Validation

Ensures compliance with PEP-257 docstring conventions using `pydocstyle`.

15+

Maintainability Index

A crucial metric indicating how easy the code is to modify and understand.

75%

Documentation Coverage

Measures the percentage of code elements that are adequately documented.

Cyclomatic

Complexity

Quantifies the number of independent paths through the code, indicating testability.

Reporting Capabilities:

- **Granular Scores:** Provides quality scores at both per-file and project-level.
- **Exportable Formats:** Generates reports in JSON, CSV, and CI-friendly formats for seamless integration.

Milestone 4: CLI & Configuration

Command-Line Interface (CLI) Features:

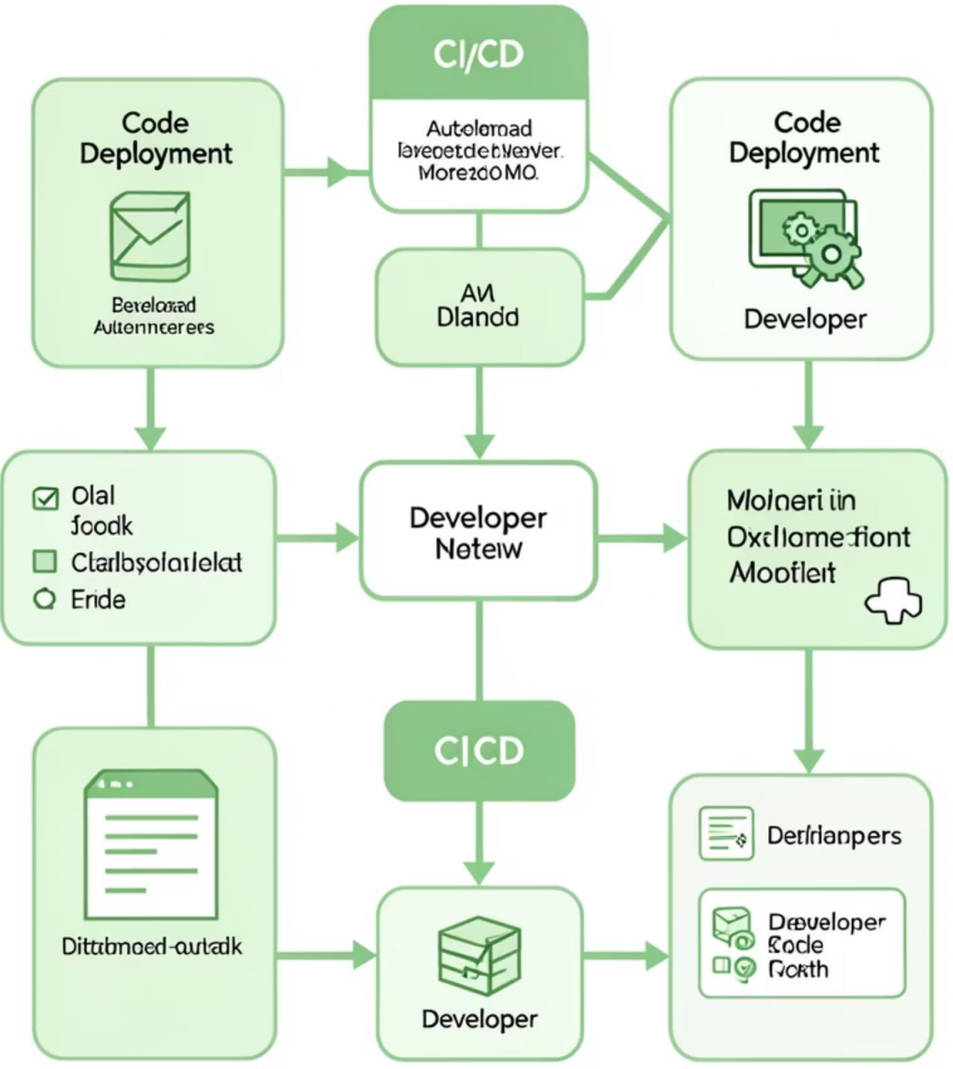
- `scan`: Initiates a comprehensive scan of the codebase.
- `review`: Generates and displays AI-powered review comments.
- `apply`: Applies approved suggestions (if enabled).
- `diff`: Shows a clear diff view of proposed changes.
- `report`: Generates detailed quality reports.

Flexible Configuration Options:

The tool allows for extensive customisation through a `pyproject.toml` file, ensuring it adapts to diverse project requirements.

- **Severity Thresholds:** Define acceptable levels for warnings and critical issues.
- **Excluded Paths:** Specify files or directories to be ignored during scans.
- **Style Preferences:** Configure preferences for code style, naming conventions, and documentation formats.

Designed for optimal automation and scripting within development workflows.



Git & CI/CD Integration

Seamlessly integrates into existing development pipelines to enforce quality standards proactively.

Git Pre-commit Hook

Automatically reviews staged files before each commit, ensuring early detection of issues.

CI/CD Pipelines

Integrates with platforms like GitHub Actions and GitLab CI to block builds that fall below defined quality thresholds.

Quality Gates

Enforces quality gates, preventing low-quality code from being merged into the main codebase.

Consistent Standards

Ensures consistent adherence to coding and quality standards across all development teams.

THANK YOU