AI-Powered-Code-Reviewer & Quality Assistant

Amal P V
Batch 10
Infosys Springboard

Introduction

The Problem: Modern codebases evolve at a rapid pace, which often leads to inconsistent review quality and a lack of standardized practices.

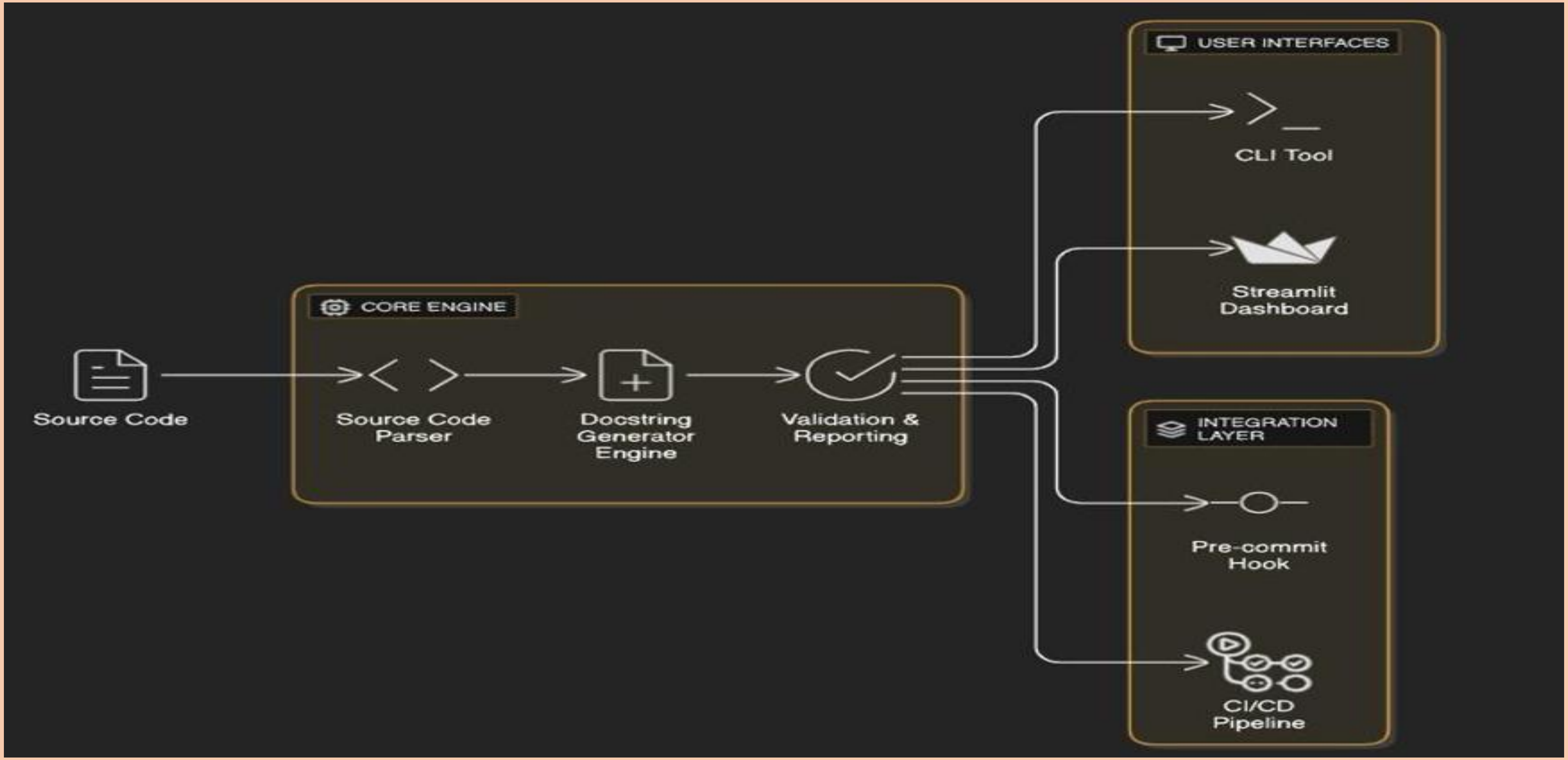
Manual Limitations: Traditional manual code reviews are extremely time-intensive and their effectiveness depends heavily on the specific expertise of the reviewer.

The Solution: This project introduces an AI-assisted tool designed to automatically analyze Python code for style, performance, and potential bugs.

Objectives

- Extract code structure and check if the functions have docstrings.
- Track documentation coverage.
- Generate missing docstrings following PEP 257 for the following docstring formats – NumPy, Google, reStructuredText (reST).
- Accept/Reject docstrings from the streamlit UI.
- Validate the efficiency for each module and store the reports in json file.

System Architecture



Technology Stack

Programming Language - Python 3.9+

Libraries & Frameworks

- **Streamlit:** Interactive web-based user interface and dashboard
- **LangChain:** LLM orchestration and integration framework
- **Groq API (LangChain-Groq)**: High-performance LLM inference
- **Python AST Module:** Abstract Syntax Tree parsing for code analysis
- **Radon**: Code complexity and maintainability metrics
- **Pydocstyle**: Docstring convention checker
- **Pytest**: Testing framework with JSON report generation
- **Pandas**: Data manipulation and analysis

AI/ML Technologies

- **Large Language Models (LLMs)**: Groq-powered transformer models for code understanding and generation
- **Natural Language Processing**: Text generation and analysis
- **Vector Embeddings**: Semantic understanding of code context (via LangChain)

Milestone 1

Focus: Parsing & Baseline Generation

- Implemented AST-based extraction of functions.
- Functions obtained are checked for existing docstrings.
- Coverage percentage based on existing docstring is displayed on the streamlit UI.
- Bar charts are employed to visually convey the same.

Milestone 2

Focus: Docstring Style Support & Validation

- Implemented docstring generation for NumPy, Google, and reST formats by predefining it in the code.
- Checks for violations of PEP 257 constraints.
- Enhanced the UI.

Milestone 3

Focus: Integration of Generated Docstring with the code

- LLM model (Llama 3.1 8b) integration and docstring generation.
- Added “Accept & Apply” button that directly affects the source file.
- On click, it adds docstring to the respective function.

Milestone 4

Focus: Dashboard View & Testing

- Addition of advanced search and filter options.
- Test results of different modules displayed in bar graphs.
- Incorporated tips for end users on how to operate the system.

Transform Code Quality: Automated Review & Documentation

Reduce Manual Effort

- Automates code review process, saving hours of manual analysis.
- Generates professional docstrings automatically.
- Eliminates repetitive documentation tasks.

Improve Code Consistency

- Ensures uniform docstring formatting across projects.
- Maintains coding standards automatically.
- Enforces best practices consistently.

Enhance Code Quality

- LLM-powered intelligent code analysis and recommendations.
- Real-time code complexity metrics (Cyclomatic Complexity, Maintainability Index).
- Identifies potential improvements and vulnerabilities.

Better Documentation & Onboarding

- Auto-generated comprehensive docstrings (Google, NumPy, reStructuredText formats).
- Improved knowledge transfer for new team members.
- Professional, exportable reports for compliance.

Thank You