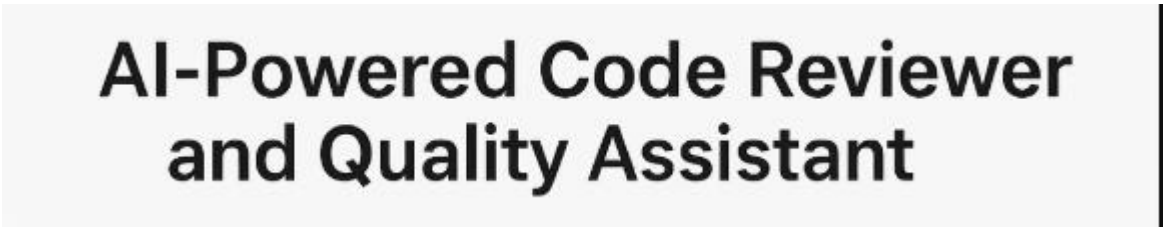
1. **Title**

AI-Powered Code Reviewer and Quality Assistant



2. Project Statement

Modern codebases evolve rapidly, often without consistent review quality or standardized practices. Manual code reviews are time-intensive and depend heavily on reviewer expertise. This project builds an **AI-assisted code review tool** that automatically analyzes Python code for style, performance, and potential bugs. It leverages static analysis, large language models (LLMs), and code embeddings to provide **actionable feedback** in pull requests or CI pipelines. Developers can interact through a **CLI** or an optional **Streamlit UI**, reviewing, accepting, or ignoring suggestions.

By integrating into Git workflows, the tool helps maintain high-quality code standards while reducing reviewer workload.

---

3. Outcomes

- A CLI tool for AI-assisted code reviews on Python projects.
- Identifies potential issues: unused imports, complexity spikes, missing tests, poor variable naming, etc.
- Suggests improvements with natural-language explanations.
- Configurable linting and rule sets (PEP8, custom checks).
- Optional Streamlit dashboard for reviewing AI suggestions.
- Integration with Git pre-commit and CI/CD to enforce quality gates.

---

4. Modules to be Implemented

1. Code Parsing & Analysis

- Parse Python files via ast and static analysis.
- Extract structure: imports, classes, functions, complexity, dependencies.
- Detect smells: long functions, deeply nested loops, missing type hints.

2. AI Review Engine

- Use LLM-based prompt templates to generate human-like code feedback.

- Rank findings by severity (info, warning, critical).
- Optionally auto-fix simple issues (naming, docstrings, spacing).

3. Validation & Metrics

- Evaluate code quality scores per file and overall.
- Track maintainability index, complexity metrics, and coverage hints.
- Export reports (CSV/HTML).

4. CLI & Configuration

- Commands: scan, review, apply, report, diff.
- Configurable rules in pyproject.toml.
- Supports severity thresholds and excluded paths.

5. VCS & CI Integration

- Git pre-commit hook to auto-review staged files.
- CI templates for GitHub/GitLab enforcing quality gates.

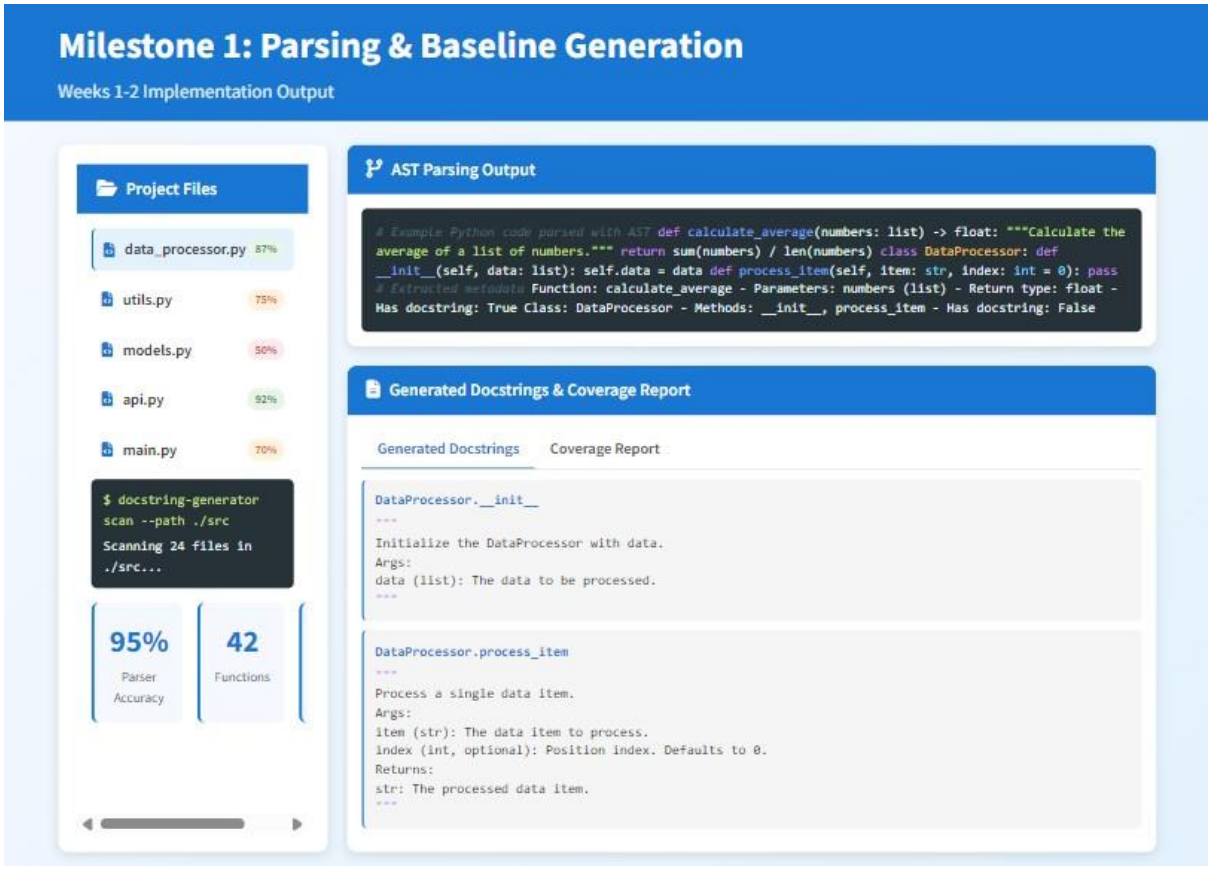
6. Review Web UI (Optional)

- Streamlit interface for visualizing issues and fixes.
- Side-by-side diff view with AI suggestions.

---

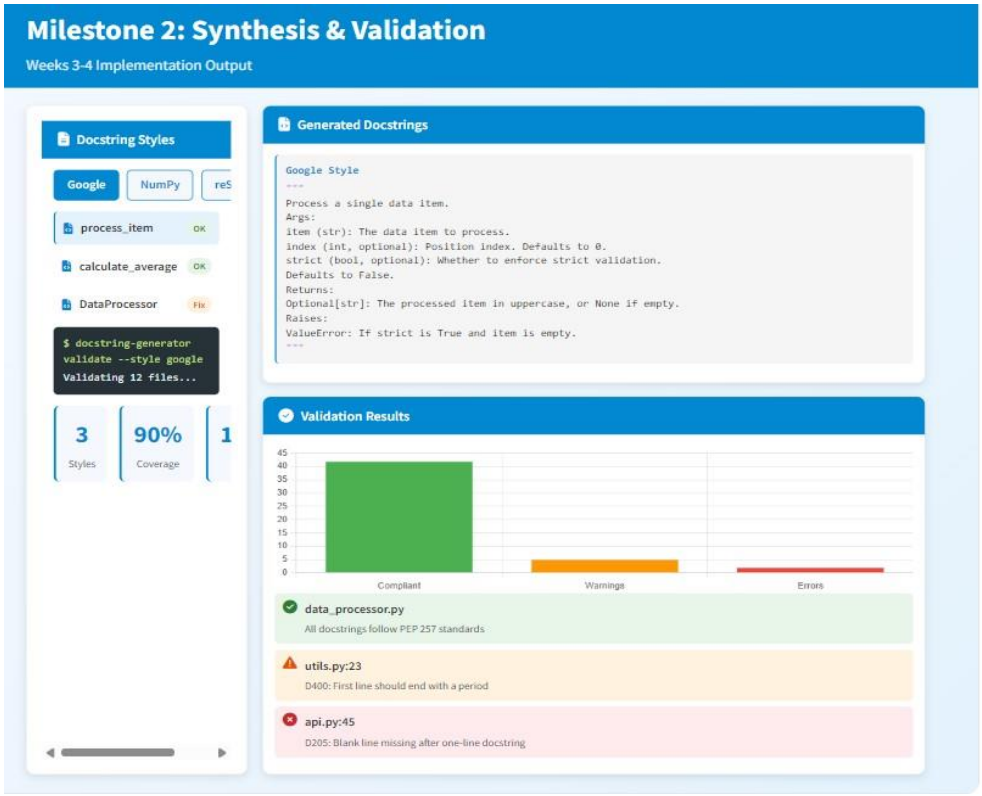
5. Week-wise Module Implementation & High-Level Requirements Milestone 1 (Weeks 1–2) – Parsing & Baseline Generation

- Implement AST-based extractor for functions, classes, and modules.
- Generate baseline docstrings in Google style.
- Produce initial docstring coverage report.



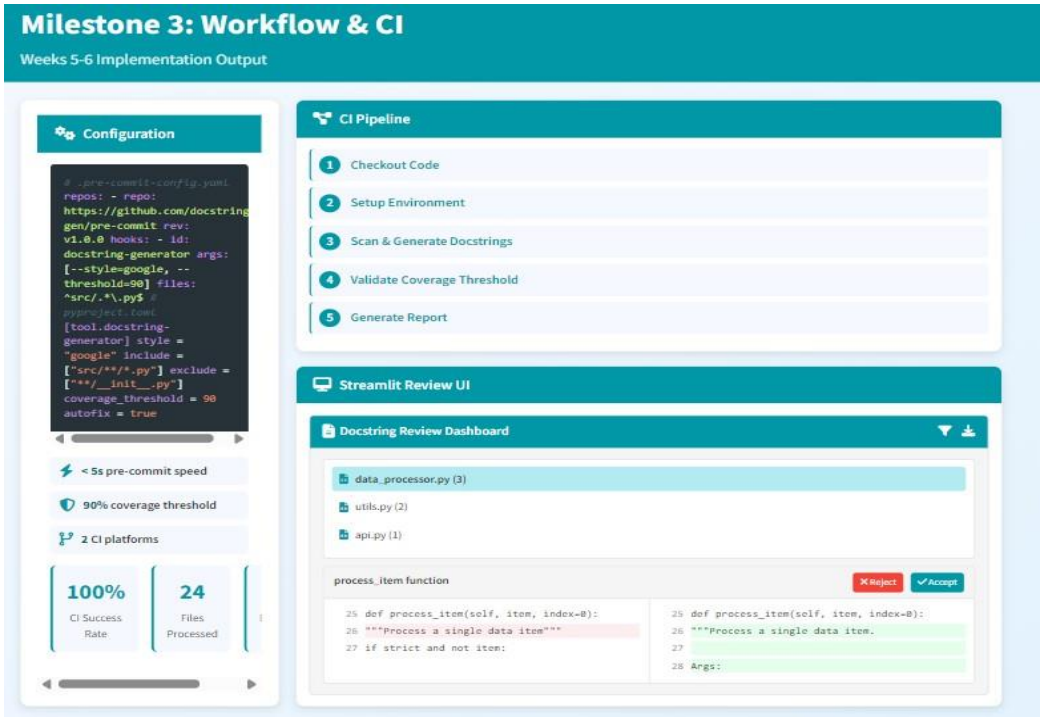
Milestone 2 (Weeks 3–4) – Synthesis & Validation

- Add support for **NumPy and reST styles**.
- Improve generation with Raises/Yields/Attributes sections.
- Integrate **pydocstyle checks** and coverage reporting.



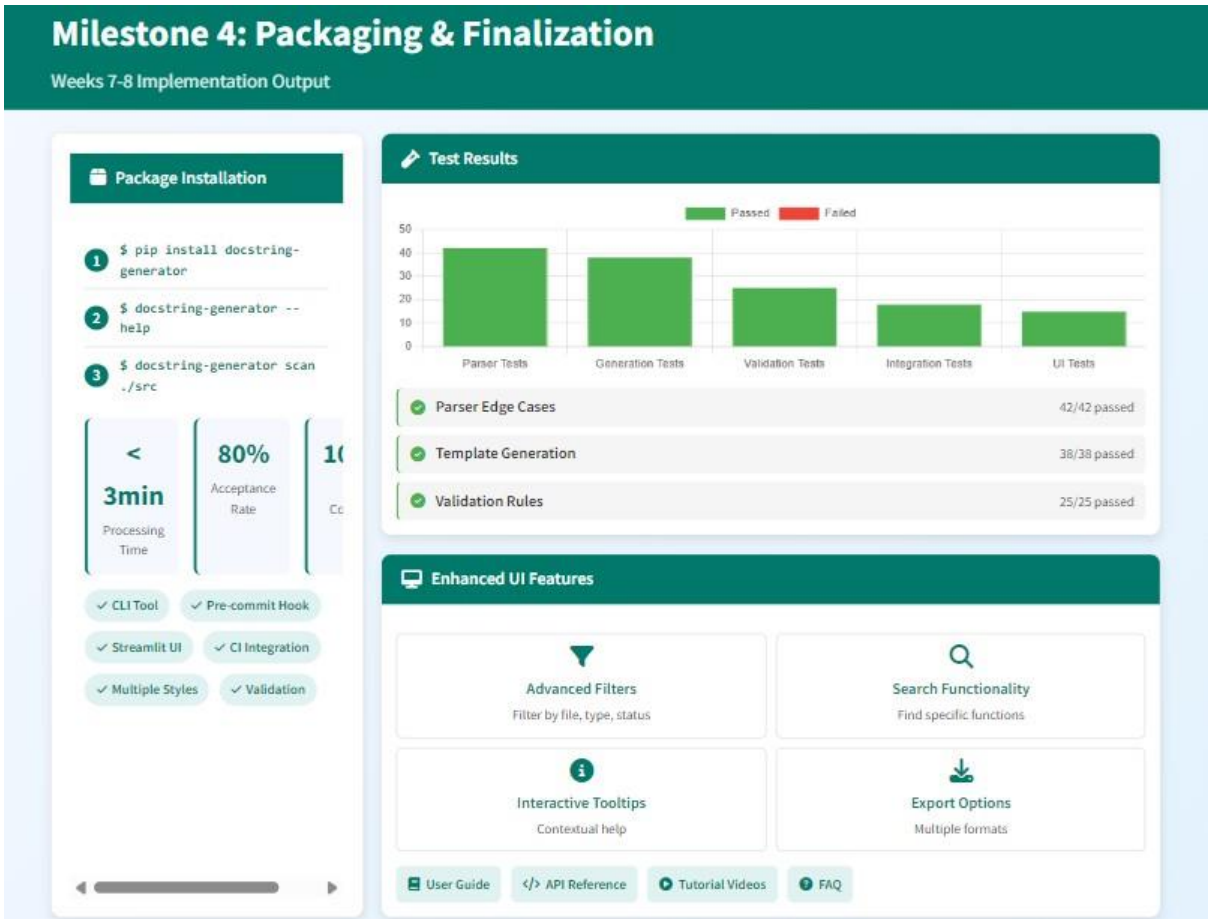
Milestone 3 (Weeks 5–6) – Workflow & CI

- Add **pre-commit hook** and **CI workflow** with coverage enforcement.
- Support configuration via pyproject.toml.
- Build Streamlit **review UI prototype**.



Milestone 4 (Weeks 7–8) – Packaging & Finalization

- Package tool as a **pip-installable library**.
- Add robust tests for edge cases.
- Improve Streamlit UI (filters, search, tooltips).
- Publish documentation and usage guides.



---

6. Evaluation Criteria

Milestone 1 (Week 2)

- Parser extracts ≥95% of functions/classes without errors.
- Baseline docstrings generated for all missing cases.
- Coverage report generated successfully.

Milestone 2 (Week 4)

- Generated docstrings conform to **PEP 257**.
- Support for at least 3 docstring styles.
- Coverage ≥90% for target sample repositories.

Milestone 3 (Week 6)

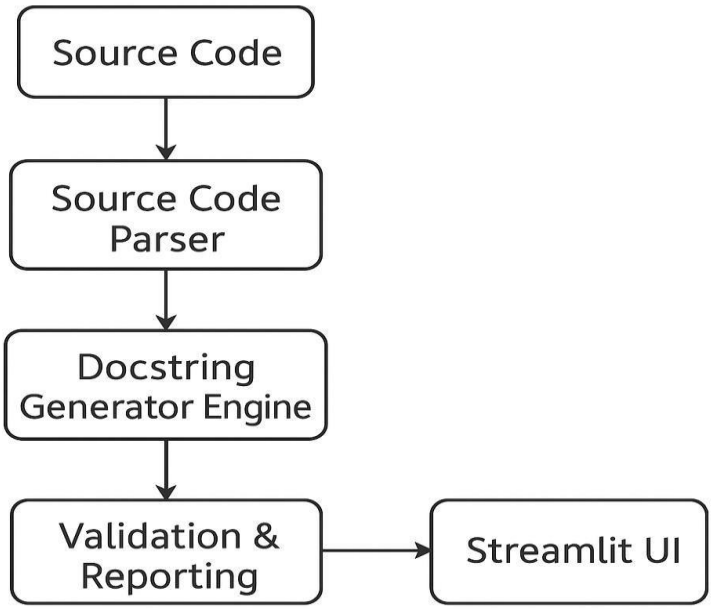
- CI pipeline blocks builds under configured coverage threshold.
- Pre-commit runs in <5s on staged changes.
- Streamlit UI supports previewing and accepting/rejecting docstrings.

Milestone 4 (Week 8)

- Tool installable via pip with documentation.
- Works end-to-end on medium projects (<2k functions) in <3 minutes.
- Developers report ≥80% acceptance rate of generated docstrings.

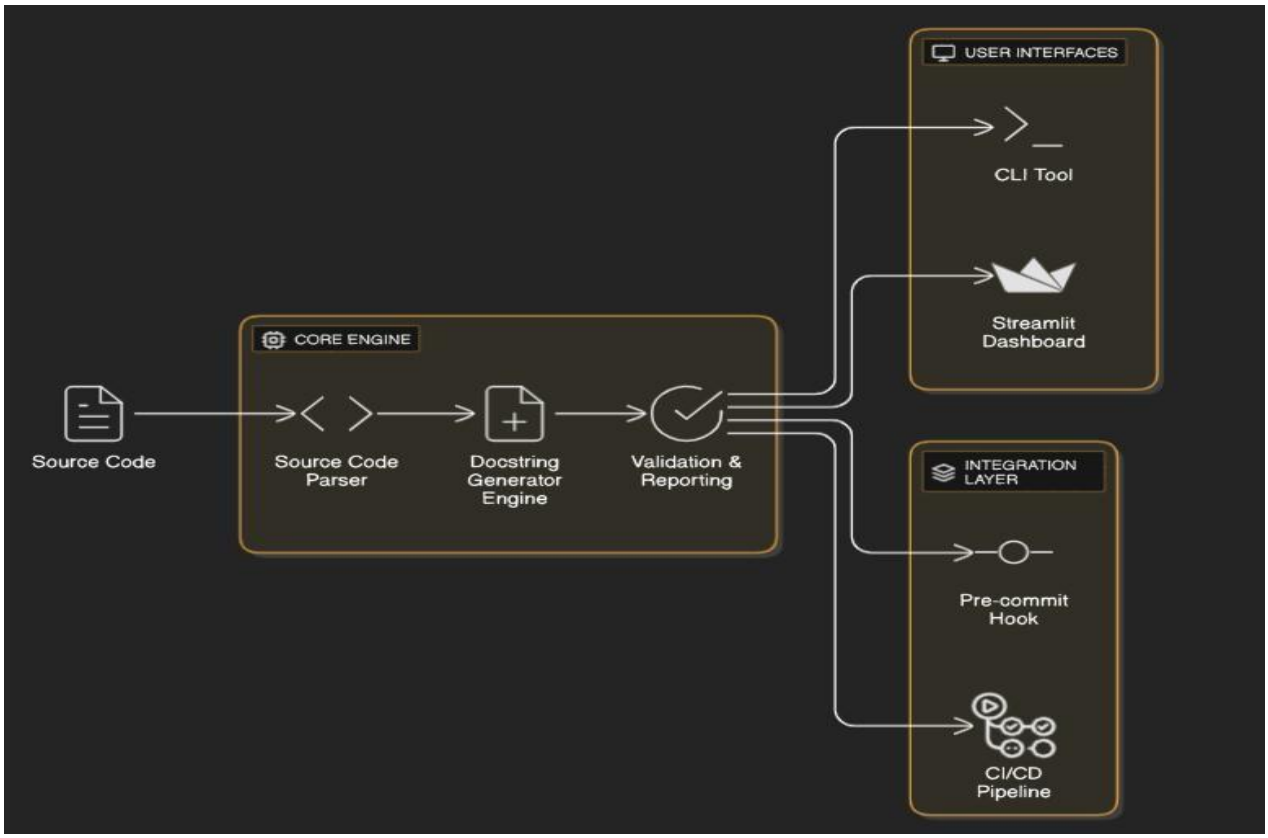
---

7. Workflow Diagram



---

8. Architecture Diagram



9. Database Schema