Project Report: Pybase AI — AI Code Reviewer Tool

Author: Rohit Bhavsar
Date: July 21, 2025

Abstract

This report presents the development of Pybase-AI, an intelligent Python code reviewer that integrates AI-based suggestions with conventional static code analysis tools. The application enhances the code review process by providing feedback on syntax, complexity, maintainability, and potential security vulnerabilities. Built with Python and Streamlit, it leverages open-source tools (flake8, black, radon, bandit) for local analysis and Google's Gemini API for context-aware suggestions and unit test generation. The project was completed as part of the Elevate Labs Internship and is deployed publicly to demonstrate real-world usability.

---

1. Introduction

1.1 Problem Statement

Manual code review can be time-consuming, inconsistent, and limited in scope—especially for beginners or fast-paced teams. Tools like linters and formatters help maintain standards but lack contextual intelligence. Developers often struggle to detect deeper maintainability issues or receive feedback that explains *why* a problem exists.

1.2 Project Objectives

The primary objectives of this project were to:

1. Provide an intuitive UI for reviewing Python code via upload or pasting.
2. Analyze code with widely adopted tools like flake8, radon, and bandit.
3. Format code using black and visualize complexity metrics.
4. Integrate AI-based inline feedback and auto-generated unit tests using Gemini API.
5. Enable developers to download a full analysis report for future reference.

1.3 Scope

The first version of Pybase-AI supports static code analysis, AI-powered suggestions, and test generation for any Python script. It allows code upload or paste, formats the code, provides insights, and offers downloadable reports—all through a web-based UI.

---

2. System Architecture and Design

2.1 Core Components

- **Streamlit** (UI): For developing the interactive web interface quickly with Python.
- **flake8**: Linting tool for identifying style guide violations.
- **black**: Opinionated formatter for ensuring consistent code layout.
- **radon**: Tool to calculate cyclomatic complexity and maintainability metrics.
- **bandit**: Scans code for common security vulnerabilities.
- **Gemini API**: Google's generative AI model used for contextual suggestions and test case generation.

2.2 Data Flow

The application follows a simple, linear data flow:

1. **User Input**: User uploads or pastes Python code.
2. **Pre-processing**: black reformats the code; code is stored in session state.
3. **Static Analysis**:
    - flake8 for linting
    - radon for complexity
    - bandit for security checks
4. **AI Features** (Optional):
    - **Inline suggestions** via Gemini
    - **Unit test generation**
5. **Report Generation**: Combines all feedback into a downloadable .txt file.
6. **Display**: Results shown across six tabbed sections in Streamlit.

---

3. Features Implemented (Tier 1)

- Paste or upload Python code
- Run flake8, radon, bandit, and black
- Display cyclomatic complexity in bar charts
- Generate full-text reports
- AI-powered inline comments (Gemini)
- AI-based unit test generation
- Download report as .txt

---

4. Deployment

4.1 Deployment Strategy

The app is deployed on Streamlit Community Cloud. To resolve dependency parsing issues (e.g., incorrect requirements.txt format), the package list was cleaned and updated.

A secret token-based approach was used to integrate Gemini API via secrets.toml.

4.2 Live Demo

Live Demo: Pybase - AI Code Reviewer

---

5. Future Roadmap

- **Tier 2:**
    - Integrate local LLMs using Ollama for offline suggestions
    - Add support for codebase-wide reviews and file navigation
- **Tier 3:**
    - Maintain coverage dashboard with test-case status
    - Enable GitHub repo analysis directly via URL
    - Schedule review reminders or batch reviews

---

6. Conclusion

Pybase-AI automates Python code review using industry-standard tools combined with Google's Gemini AI model. The project is a step toward smarter developer tooling, reducing review time and increasing code quality even for beginner developers. Its extensible architecture makes it suitable for both personal learning and production team environments.