AI – CODE – REVIEWER – PROJECT– REPORT

I. INTRODUCTION.

Software development is not just about writing code that complies, but also ensuring that the code is clean, efficient, readable, maintainable and secure(bug free). Traditionally, this requires manual reviews by a teammates or mentors, which can be time–consuming and error-prone.
Our project, was built to solve this problem. It's a multi-language automated review platform that combines the best of both worlds:

- Classic liniting tools to catch syntax and style issues.
- Complexity analysis to measure how hard the code is to read and maintain.
- AI – Powered Review for deeper insight where tools fall short for Java and Go.
- And finally, the ability to generate polished PDF reports so results can be shared or archived.

With a FastAPI backend and Streamlit frontend, the project is both practical for developers and simple enough for student or beginners to use. And the project is successfully developed and deployed in Render.

II. ABSTRACT.

The main idea was to create an automated code review assistant. Instead of juggling different tools for different languages, users can just paste their code, select the language, and get:

- Immediate feedback on errors, formatting and best practices.
- An AI-generated summary of strengths and weaknesses.
- An optional PDF report for records or assignments.

The tools supports Python, JavaScript, C++, Java, and GO. It has been deployed on Render, so it's accessible online without setup. This makes it useful for developers, students preparing projects, or even organizations looking for lightweight review automation.

III. TOOLS USED.

- Backend (FastAPI): Handles requests, runs linters, calls Ai review.
- Frontend(Steamlit): Provides a simple, clean dashboard for users.
- Language tools:
    1. Python – pylint, radon.
    2. JavaScript – eslint.
    3. C++ – cpplint, lizard.
    4. Java & GO – reviewed using AI feedback.
- Other Libraries: requests, reportlab (for PDF), dotenv.
- Deployment: Render (Cloud Hosting), GitHub (Version Control).

IV. STEPS IN BUILDING THE PROJECT.

Planning – We first mapped out what developers usually expect in a code review: error checking, complexity insight, AI-style feedback, and reports.

Backend Development – We built a FastAPI service with three main endpoints:

- analyze → runs linters or AI review.
- report → generates a PDF review report.

For each language, we wrote dedicated modules so the system is modular and easy to extend.

Frontend Development – Using Streamlit, we built a one-page app where users can either paste code or upload a file. We styled it with **custom CSS** so it feels modern — not like a raw prototype.

Testing – We ran both small snippets (like unused variables) and larger code files to make sure the tool could handle different cases. We checked that it gives useful, actionable feedback instead of just plain errors.

Deployment – Finally, we deployed the backend and frontend to Render so the tool can run from anywhere, and linked it to GitHub for easy updates.

V. CONCLUSION.

The AI-Code-Reviewer project shows how a mix of automation and AI can make code reviews faster and more accessible.

- It supports multiple languages.
- It produces both technical analysis and natural-language suggestions.
- It generates professional reports at the click of a button.

This project has immediate practical use for students, solo developers, and teams who want quick reviews without setting up multiple tools. In the future, it could be extended to:

1. Integrate directly into GitHub/GitLab workflows,
2. Add real-time collaboration,
3. Support even more languages and security scanning.

In short, AI-Code-Reviewer is a step toward smarter, more efficient software development — where developers spend less time on repetitive checks and more time writing great code.