1. Introduction

This project focuses on the automated review of student Python programs to assist in identifying common programming issues. Instead of executing the code, the system analyzes the structural components of the program. It helps in detecting syntax errors, coding style problems, and areas where improvements can be made. The goal of the project is to reduce manual effort in code evaluation and provide quick feedback to students. This approach is especially useful in academic environments.

2. Analysis Approach

The system follows a static code analysis approach to evaluate Python programs. Static analysis inspects the source code without running it, which makes the process safe and efficient. By examining the structure and syntax of the program, the system can identify errors and poor coding practices early. This method avoids security risks associated with executing untrusted code. It also ensures faster analysis suitable for large numbers of submissions.

3. Use of Abstract Syntax Tree (AST)

Abstract Syntax Tree (AST) is used to represent Python source code in a structured tree format. Each node in the tree corresponds to a programming construct such as functions, variables, loops, and import statements. By traversing the AST, the system can understand how the code is written and organized. This helps in detecting issues like unused imports, long functions, and missing definitions. AST-based analysis allows accurate inspection without executing the program.

4. Future AI Extension

In future enhancements, artificial intelligence models can be integrated to improve feedback quality. AI language models can convert detected issues into detailed and human-readable explanations. They can also suggest optimized code patterns and best practices. This will make the feedback more interactive and helpful for students. The AI extension can further enhance learning by providing personalized suggestions.