Software Metrics Analyzer

Submitted By-

Nafiz Mahmud Fardin

BSSE-1528

Supervised by-

Dr.Mohammad Shoyaib

Professor

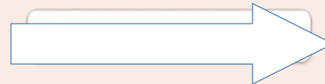
IIT,DU

25-3-2025

Introduction

Brief Overview

The **Software Metrics Analyzer** is a tool designed to evaluate the quality and complexity of a software codebase. It calculates various software metrics, such as Code Size Metrics, Cyclomatic Complexity, Halstead Metrics, and more. The project provides statistical insights to help developers understand the maintainability, efficiency, and complexity of their code.



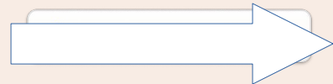
Objective

To **automate** the process of analyzing software metrics.

To **help developers and researchers** assess software quality and maintainability.

To **provide visualization** of different complexity factors for better interpretation.

To **enhance software engineering practices** by offering data-driven insights.



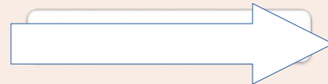
Project Background and Motivation

Importance of Software Metrics

Software metrics are crucial for evaluating code quality, identifying potential issues, and ensuring maintainability. They provide quantitative measures to assess complexity, size, and other important characteristics of software.

Motivation

This project aims to address the challenges in existing approaches by providing an efficient and comprehensive tool for software metrics analysis. It seeks to automate the process and offer valuable insights into code quality.



System Architecture Overview

1

Architecture Overview

The system architecture involves several components working together to analyze code and generate metrics. These components include file analyzing, metric calculators, and a visualization module.

2

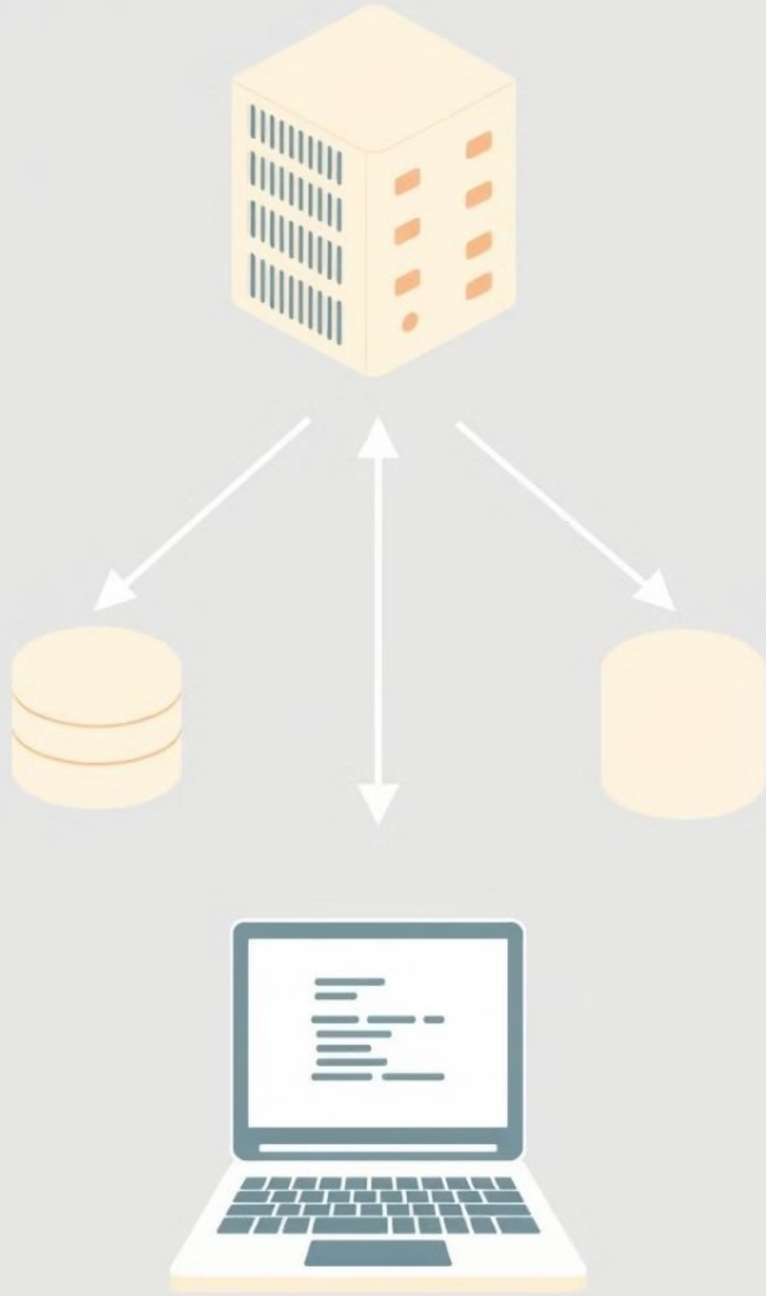
Components and Their Roles

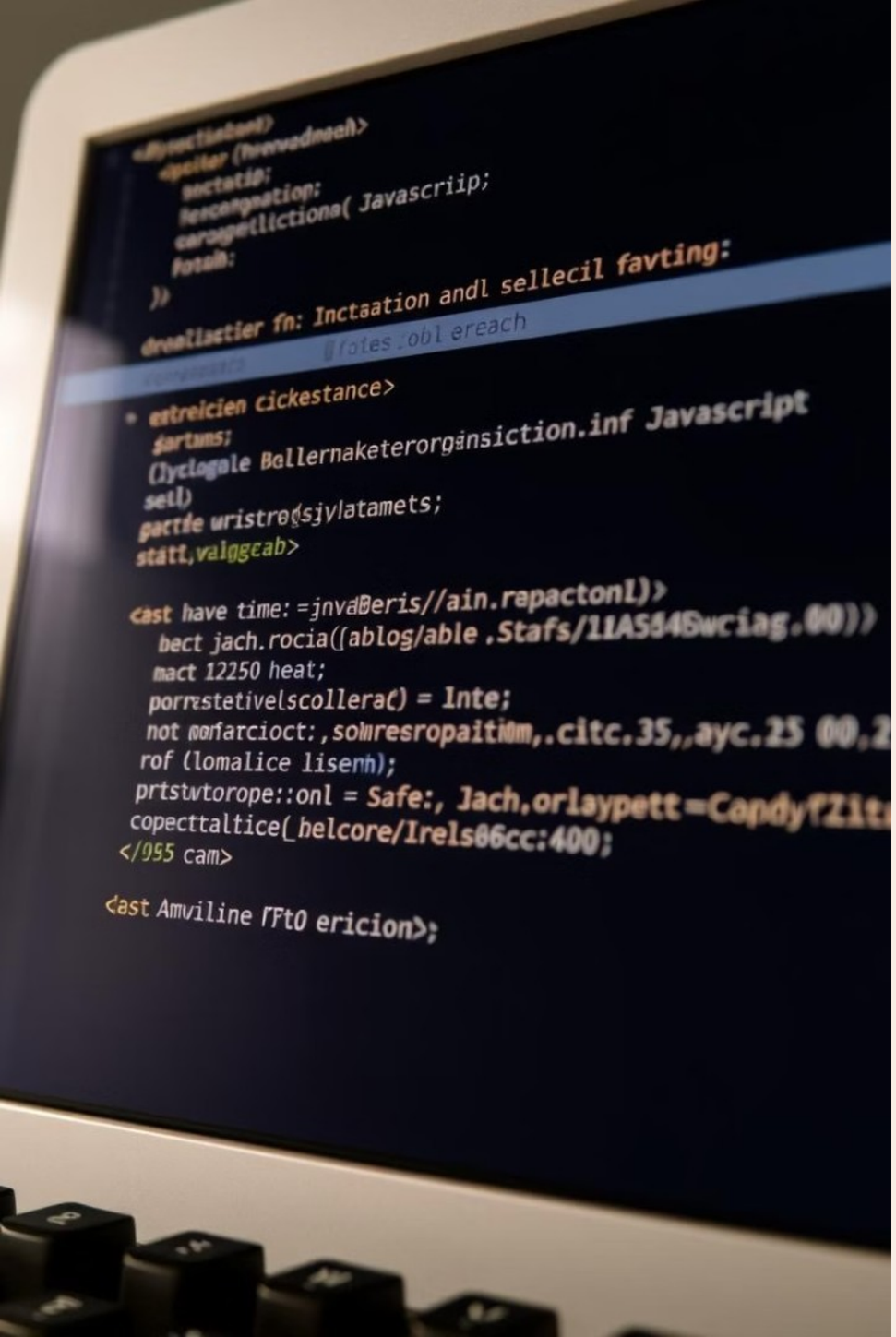
Each component plays a specific role in the analysis process. The analyzer extracts code structures, the metric calculators compute relevant metrics, and the visualization module presents the data.

3

Data Flow

The data flow in the system involves analyzing the code, calculating metrics, and presenting the results through a user-friendly interface. This process ensures accurate and efficient analysis.





Implementation Details

Code Size Metrics

Implementation of code size metrics involves calculating lines of code, number of functions, and other size-related measures to assess the overall scale of the software.

Cyclomatic Complexity

Cyclomatic complexity is calculated to determine the number of independent paths in the code, providing insights into its complexity and testability.

Halstead Metrics

Halstead metrics are computed to evaluate the effort and difficulty of writing the code, based on the number of operators and operands used.

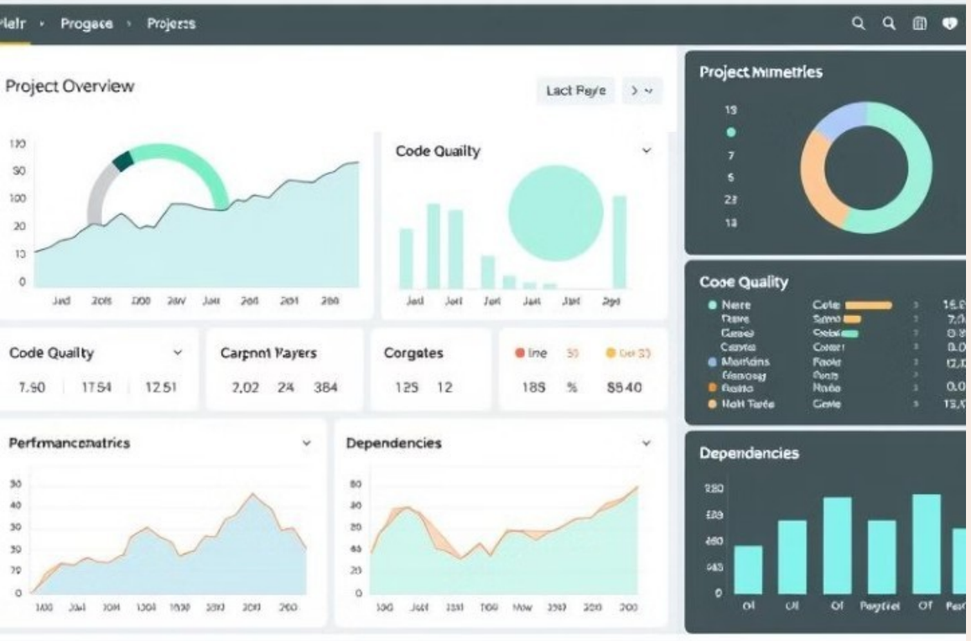
Testing Details

Unit Testing

Unit tests were conducted to ensure each component of the system functions correctly. These tests validate the accuracy of metric calculations and the reliability of the parser.

Integration Testing

Integration tests were performed to verify the interaction between different components. These tests ensure the system works seamlessly as a whole.



User Interface Features

Dark Mode

The user interface includes a dark mode option for improved readability and reduced eye strain in low-light environments.

Exit Button

An exit button is provided for easy termination of the application, enhancing user convenience.

Save as PDF

A save as PDF button allows users to export the analysis results for documentation and sharing purposes.

Challenges Faced

Parsing and Analyzing Code

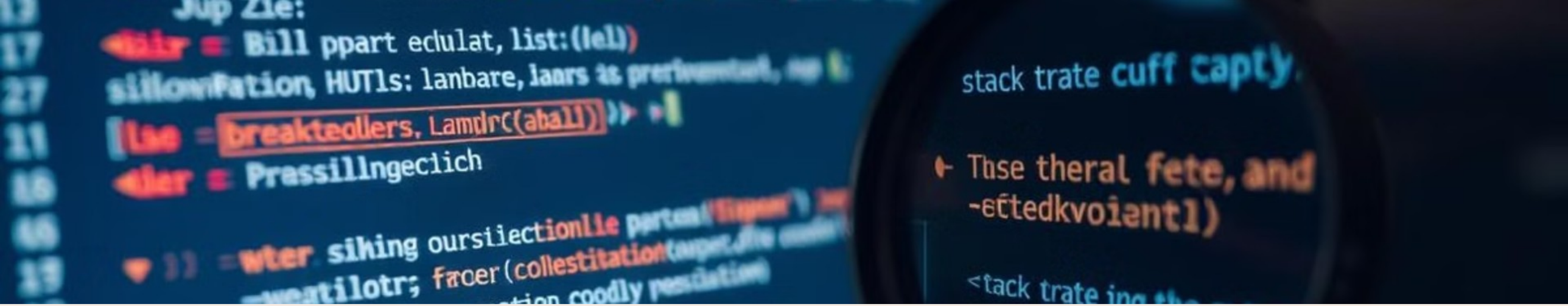
Parsing and analyzing different code structures presented a significant challenge due to the variety of programming languages and coding styles.

Efficient Metric Calculation

Efficiently calculating software metrics required optimization techniques to handle large codebases without compromising performance.

Handling Large Codebases

Handling large codebases and multiple files demanded robust memory management and efficient data processing strategies.



More Challenges

Graph Visualization

Integrating graph visualization with metrics data required careful design to ensure the information was presented clearly and effectively.

Cross-Platform Compatibility

Ensuring cross-platform compatibility and managing the build system with CMake added complexity to the development process.

Debugging and Validating

Debugging and validating metric accuracy was crucial to ensure the reliability of the analysis results.

Future Enhancements

1

Multiple Languages

Support for multiple programming languages to broaden the tool's applicability.

2

Real-Time Analysis

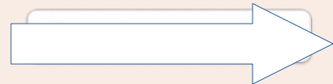
Real-time code analysis and IDE integration for immediate feedback.

3

Advanced Metrics

More advanced code quality metrics to provide deeper insights.

2.



Conclusion

The Software Metrics Analyzer project successfully delivers a tool for assessing code quality and complexity. It addresses the need for automated metrics analysis and provides valuable insights into software development.

Despite the challenges faced, the project achieved its objectives and lays the groundwork for future enhancements. The tool's user-friendly interface and comprehensive metric calculations make it a valuable asset for software developers.

Thank You

Thank you for your time and attention. I appreciate the opportunity to present my Project

Github Link: https://github.com/nafizfardin28/SPL-1