International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Complify: Smart Online Compiler with AI-Based Error Explanation and Optimization

Under the guidance of
Dr. Sachidanand Joshi

Co-ordinated By
Dr Varsha Jadhav

Sharat Kubasad, Sanjay Kulkarni, Rizwan Rikati, Varsha Muktenahalli, Keerti Shanbagh.

SDM COLLEGE OF ENGINEERING AND TECHNOLOGY DHARWAD

Department of Information Science and Engineering

***Abstract*—The complexity in the software development increasing day by day, And it created many challenges. Traditional compilers focuses on Translating high level coding language to machine instruction but they will not give perfect feedback for the user. So this paper presents Complify a smart compiler which integrates artificial intelligence in the compiler. The compiler provide human readable error explanation and optimization code. By combining normal compiler to artificial intelligence , This project improve the usability and also helps in debugging the errors easily.**

***Index Terms*—Smart Compiler, Artificial Intelligence, NLP, Optimization, Cloud Computing**

I. Introduction

Compiler plays an important role in the software development by converting high-level programming languages into executable machine code. The compilers efficiently make lexical analysis, syntax checking, semantic validation and code generation. But there is lack of user - friendly feedback mechanism, For that we built a compiler called **COMPLIFY** that helps the user to understand the code and improve the quality of the code.

In the real-world scenarios, developers face many errors while coding which are difficult to understand and for beginners it became even more complex. So as the complexity increases the need of intelligent tools also increases to support learning, debugging and performance improvement.

Complify is built to solve these challenges by integrating artificial intelligence into the compiler. It not only translates the code but also acts as intelligent assistant model. This will help both experience and beginners for the overall development

II. Problem Statement

The rapid development of the software has several limitation in the compilers. The errors are technical and difficult to understand the user should rely on other resources For the debugging. The conventional compiler are static and do not adapt user behavior and historical data

The major limitation is absence of performance prediction. The compiler which are existed do not give personalized feedback and learning support. For these challenges the intelligent compiler is required that can provide explanation of the error and also for the optimization of the code and predictive analysis.

III. Objectives

The objective of this project is to develop an artificial intelligence integrated compiler for debugging the errors easily and also for optimizing the code. The system makes the error human readable which enables users to understand the error and resolve it effectively

Complify also focuses on the optimizing the code to improve the performance and also predict the run time and memory usage before execution. The system is designed in such a way that it can adapt the learning capabilities, allowing it to improve based on the user interaction.

IV. Literature Survey

Modern advancements in the artificial intelligence have new approaches to optimize the compiler and error handling. The techniques like reinforcement learning used to discover efficient algorithms, the graph neural networks applied to improve the compiler optimization.

For generating human readable explanation for code errors the natural language processing has also been explored. However the system which are existed are either focus on optimization or assistance and lack a approach that integrates both the functionalities. Traditional compilers such as GCC and Clang provide efficient compilation but do not offer adaptive learning or intelligent feedback,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

V. System Architecture

For scalability and efficient processing the Complify system follows a multi structured architecture. The presentation layer provide interface for the user to input the code and view the result. The compiler layer performs lexical, syntax, and semantic analysis, along with code generation.

The AI layer is used to improve the system by giving natural language errors, explanation, code optimization suggestion and performance prediction. The cloud layer helps to store the data enables distributed compilation and synchronization. This layered design ensures modularity and seamless communication between system components.

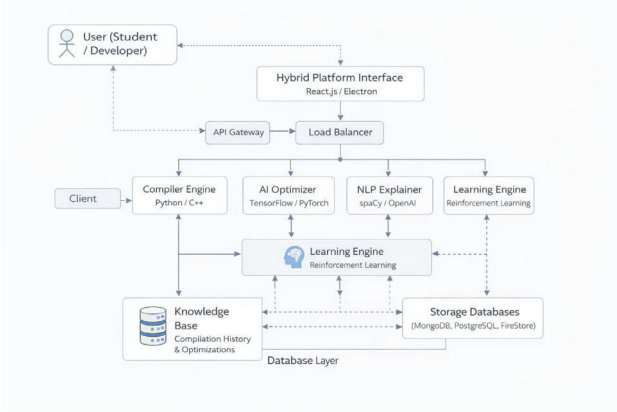

Fig. 1. System Architecture of Complify

VI. Methodology

The system works with a structure workflow that integrates AI based enhancements and traditional compiler. So the code undergoes lexical analysis in that the code is broken into tokens using deterministic finite automata. To validate the structure of the code it follows syntax analysis using LL(1) parsing technique.

For ensure logical correctness semantic analysis is done, after that the AI will generate the natural language for the errors and also suggest the optimization in the code .to predict the performance metrices like runtime and memory usage machine learning models are used. Cloud integration helps to distributed processing and storage for system improvement.

VII. Modules

Many functional modules work together to provide intelligent compilation in this system. The error module converts technical error to simple language and compiler module handles the lexical, syntax and semantic analysis. The learning module continuously improve the system by user interaction and historical data.

The optimization module helps to analyze the pattern of the code and suggest the improvement in the code for making system more efficient.

VIII. Design

Design of the Complify system focuses on both structural organization and relationships. The ER diagram shows the relationship between the entities like users, source code, error logs and optimization data, explains how date flows with in the system.

The structure of the system is defined by class diagram and also show the interaction between compiler, optimizer and AI modules. This design check the separation of responsibilities, improving maintainability and scalability.

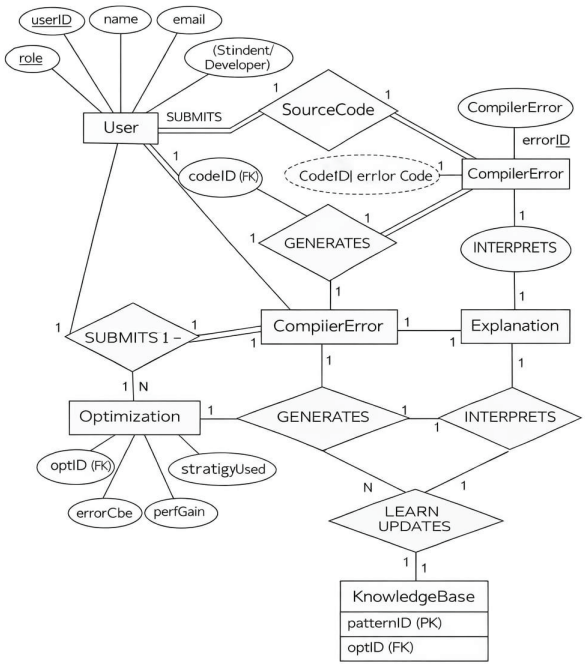

Fig. 2. ER Diagram of Complify

IX. Implementation

The implementation of the system is done using python for developing the compiler and for AI functionalities machine learning and natural language framework is used.for data storage and synchronization we used firebase as cloud service, the communication between the modules is handled by REST API's. This check the efficient execution and scalability.

X. Results and Discussion

The Complify compiler is checked in various condition, to evaluate the performance and usability.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

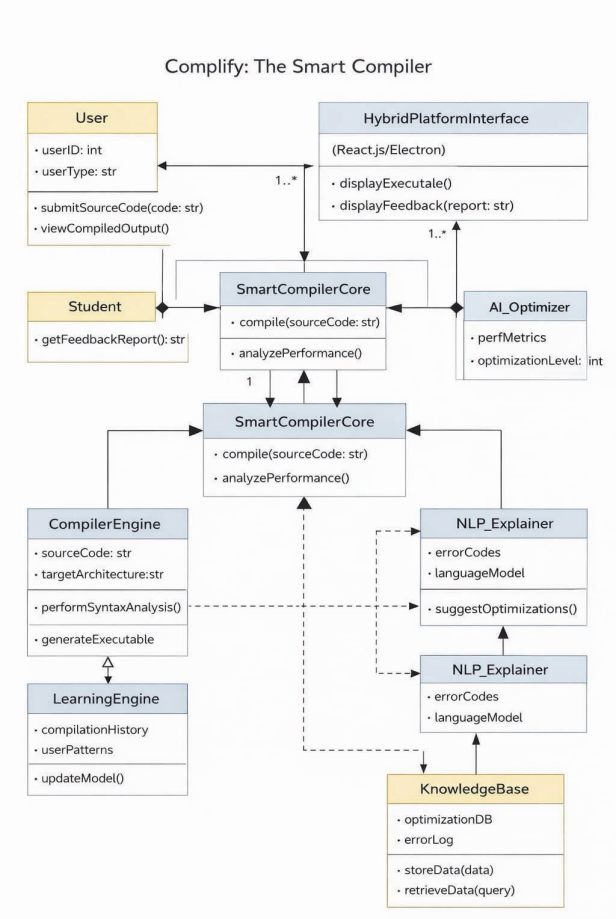

Fig. 3. Class Diagram of Complify

It improves error understanding by giving clear and human readable explanation. The optimization module helps to improve the efficiency of the code, the performance prediction module is used to identify the potential bottlenecks.

The Complify compiler reduces the debugging time and enhance user productivity compared to other compilers

XI. Future Scope

The Complify system can be enhanced for multiple program and more AI based optimization techniques. It can be integrated with modern IDE's which improve usability and collaboration features support team based development. Continuous enhancing the machine learning models will improve the accuracy and performance

XII. Conclusion

The Complify project shows an intelligent approach for designing the compiler by integrating the AI to the traditional compilation technique . The system helps to improve optimization, debugging and performance analysis. It is a helpful tool for the developers, students and researchers and also have potential to evolve into a next generation smart compiler platform.

References

[1] C. Li et al., "Accelerated Auto-Tuning of GPU Kernels for Tensor Programs," 2024.
[2] M. Ganai et al., "Target-Independent XLA Optimization using Reinforcement Learning," 2023.
[3] D. Mankowitz et al., "AlphaDev: Reinforcement Learning for Algorithm Discovery," 2023.
[4] C. Zhang, "Self-Supervised Learning for Compiler Optimization," 2023.
[5] J. Henning et al., "Graph Neural Networks for Compiler Optimization," 2022.
[6] Y. Li et al., "Natural Language Feedback for Code Generation," 2022.
[7] C. Cummins et al., "CompilerGym: A Benchmark Toolkit for Machine Learning in Compilers," 2021.
[8] Y. Wang et al., "CodeT5: Identifier-Aware Pretrained Encoder-Decoder for Code Understanding," 2021.
[9] R. Baghdadi et al., "A Deep Learning Based Cost Model for Automatic Code Optimization," 2021.
[10] C. Lattner et al., "MLIR: A Compiler Infrastructure for the End of Moore's Law," 2020.
[11] T. Chen et al., "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning," 2018.
[12] J. Ragan-Kelley et al., "Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation," 2013.