UNIFIED COMPILER FOR LANGUAGE AGNOSTIC DEVELOPMENT

Project Reference No.: *48S_BE_5982*

College : A.P.S. College of Engineering, Bengaluru
Branch : Computer Science and Engineering
Guide(s) : Prof. Puneeth R
Student(s) : Mr. Abhinav Naik
Mr. Rahul Sharma
Mr. Sayan Chatterjee

Keywords:

Unified Compiler, Language-Agnostic Development, Polyglot Programming, Multi-language Support, LLM-powered, Platform-Independent Execution.

Introduction:

A Unified Compiler for Language-Agnostic Development is a unified interface for polyglot programming enabling compilation and execution of code written in different programming languages. It supports polyglot programming, allowing seamless development across languages. The system eliminates the need to switch environments for different languages. Developers can write, view, and run multi-language code through a single interface. The frontend is built using JavaScript, offering an interactive and responsive UI. The backend is powered by C++, managing compiler invocations and system-level execution. A key feature is the integration of a Large Language Model (LLM). The LLM provides intelligent code suggestions, completions, and language conversions. It enhances the user experience and speeds up development across multiple languages. Based on the selected or inferred language, code is routed to the correct compiler. The system ensures consistent execution and unified output formatting. This project simplifies multi-language workflows and promotes language-agnostic development.

Objectives:

- **Ease of Use:** A user-friendly interface that eliminates the need for manual configuration, making writing, compiling, and executing code seamless.

- **Educational Utility:** A tool for students to experiment with multiple languages, focusing on core concepts without toolchain setup distractions.
- **Industry Applications:** Support for polyglot development in professional settings, minimizing the effort required to manage multiple environments.
- **Streamlined Debugging:** Intelligent code suggestions and feedback to help quickly identify and fix syntax, logic, and runtime issues.
- **Automation:** Automating tasks like compilation, dependency resolution, and environment setup to simplify workflows.
- **Innovation**: Cutting-edge features like code suggestions, cross-language translation, and advanced debugging to push the boundaries of software development.

Methodology:

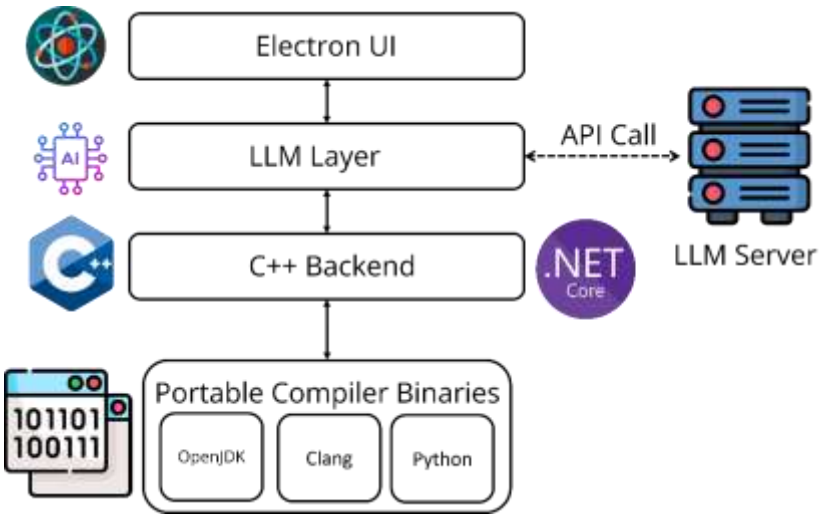

Figure 1: Architecture of Unified Compiler for Language Agnostic Development

1. Electron provides a cross-platform desktop UI for the IDE.
2. AI/ML integration enables code suggestions, auto-completion, and real-time error detection.
3. Supports multiple languages, including C++, for polyglot development.
4. Binary translation and backend execution handle compilation and simulation efficiently.
5. .NET Core backend for developing binaries that manages logic, code execution, and communication with the editor.
6. Large Language Model accessed using API calls, reducing local load.
7. Modular and scalable architecture supports updates and enterprise use.

8. Automation simplifies compilation, testing, and environment setup.
9. Unified experience for writing, debugging, and running code.

Result and Conclusion:

In conclusion, the unified compiler streamlines multi-language development by offering an intuitive interface and AI-powered features that automate routine tasks, significantly reducing the setup time, and minimizing the errors. Its scalable and modular architecture facilitates integration within existing workflows, enabling teams to deliver high-quality software more efficiently and consistently.

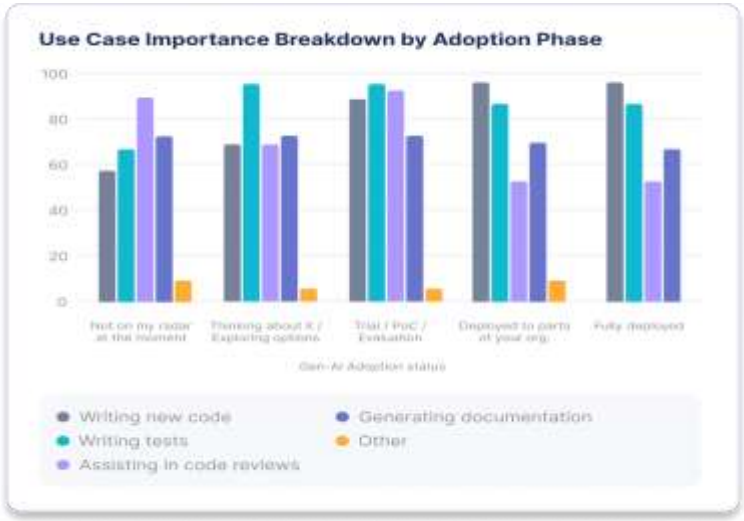

Figure 2: Adoption of Gen AI

Project Outcomes and Industry Relevance:

- **Unified Interface:** Single interface for coding, compiling, and running across multiple languages.
- **Automatic Language Detection:** AI-driven identification of the programming language and automatic compiler/interpreter selection.
- **Integrated Dependency Management:** Automatic resolution and installation of required dependencies for each language.
- **Enhanced Error Handling:** Consistent and detailed error messages across languages with built-in debugging tools.
- **Modular Architecture:** Backend designed for easy addition of new languages via a plugin system.

Software Development and IT:

- **Description:** Streamlines multi-language software development workflows by simplifying the management of diverse tools and compilers.
- **Applications:** Benefits IDEs, DevOps pipelines, and multi-language software/web development projects.

Future Scope:

The future scope of this project includes:

1. **Cloud-based compilation** for faster processing and scalability.
2. **Collaborative coding features** for team-based development.
3. **Version control integration** for seamless project management.
4. **Automated documentation** to assist in code understanding and maintenance.
5. **Support for more languages and frameworks** to widen applicability.
6. **Enhanced UI/UX improvements** for a more intuitive user experience.