2025 Girl Hackathon Ideathon Round: Solution Submission

Project Name: IntelliSense

Participant Name: Riya Chand

Participant Email ID: riyachand025@gmail.com

Participant GOC ID: 179581340031

ReadMe File Link: https://github.com/riyachand025/intelligent_ide/blob/main/README.md

Brief Summary

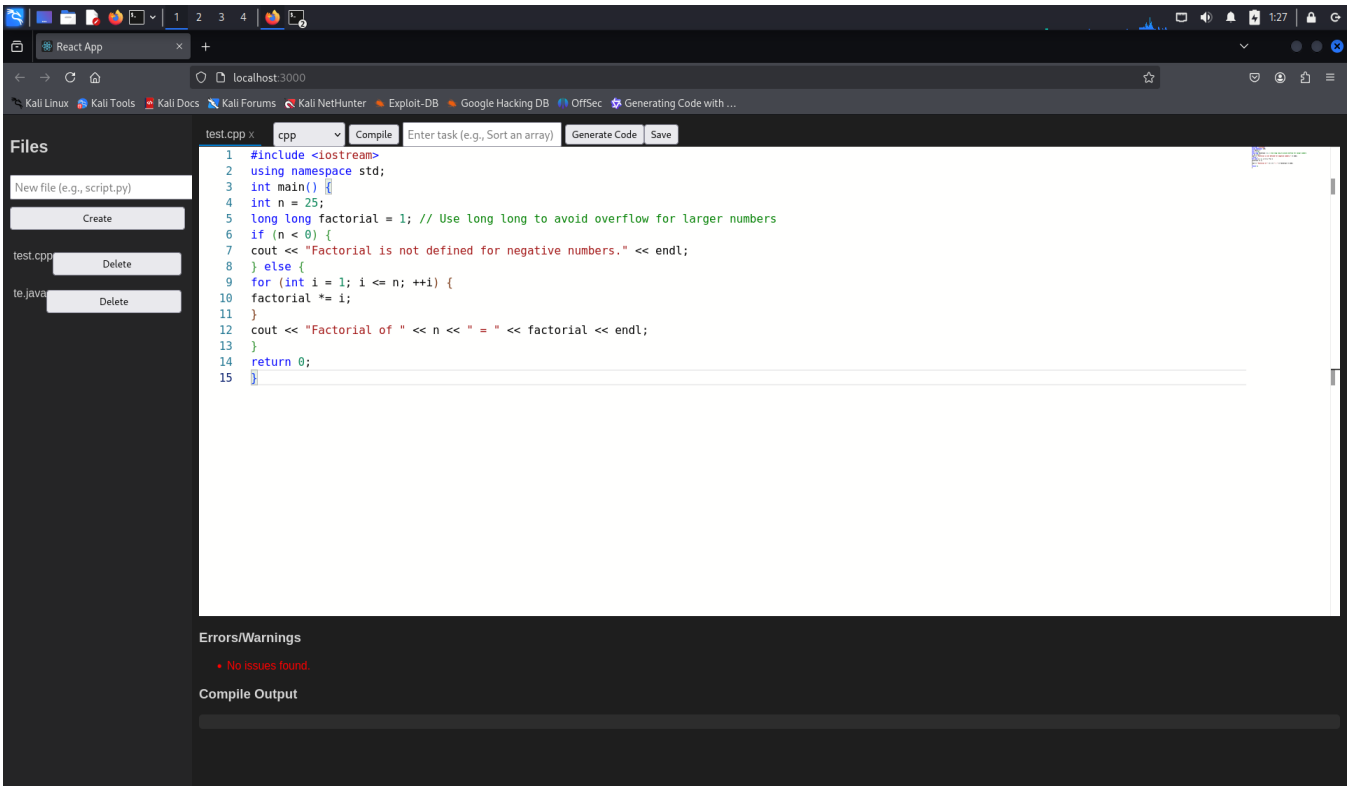
This AI-powered web app is designed to make coding faster, easier, and more efficient by automating key tasks like code generation, real-time debugging, autocompletion, compilation, and file management. It supports multiple programming languages, including Python, Java, and C++, allowing developers to write cleaner and more secure code with intelligent suggestions and NLP-based code generation. Built using Flask and integrated with Google's Gemini AI, the app provides real-time syntax feedback through Flask-SocketIO, helping users catch and fix errors as they code. It also seamlessly connects with cloud platforms, making it scalable, accessible, and ideal for collaborative development.
Whether you're an individual developer, a coding student, or part of a small team, this all-in-one coding assistant boosts productivity by reducing errors, optimizing workflows, and simplifying complex coding tasks. With its real-time debugging capabilities and AI-driven insights, it enhances security, improves code quality, and helps users focus more on building great software rather than fixing tedious bugs.

Problem Statement

Developer Productivity: **Intelligent IDE** - automated generation of code, tests with advanced debugging capabilities (automagically suggest bug fixes) leading on to largely automated continuous build and integration processes.

Design Idea and Approach

- **Technologies Used**: The backend runs on Flask, with Flask-SocketIO enabling real-time communication. Google's AI helps with code generation and debugging, while tools like Pylint and AST analyze Python code. The subprocess module handles code compilation.
- **Core Features**: It includes custom-built functions for generating code, debugging, autocompleting, compiling, and managing files. A WebSocket-based syntax checker ensures real-time error detection.
- **Development Tools**: The backend is built in Python, using libraries like tokenize, ast, and pylint for code analysis, and subprocess for running compiled code.
- **Scalability**: It can handle small-to-medium code snippets (up to 10MB) with around 100-500 queries per second on a single server. The final scalability depends on server resources and the Gemini AI API's request limits (~60 requests per minute per key).
- **Deployment Plan**: The first version runs on a local server (localhost:5000). Later, it will be moved to cloud platforms like Google Cloud Platform with load balancing for better scalability.
- **Security & Privacy**: Right now, user files are stored locally without encryption, and the API key is hardcoded, which is a security risk. Future updates will include authentication and encrypted storage for better security.



This approach combines generative AI to understand and write code while using static analysis tools to catch errors and ensure code quality.

Impact

This project tackles the time-consuming and error-prone nature of software development, a challenge faced by students and professionals. This may often lead to frustration and inefficiency. Therefore, by automating key aspects of coding, this tool can potentially reduce development time by 20-30%, helping individuals and teams work more efficiently.
The project is backed by research on AI-powered coding tools, which have shown significant improvements in productivity and error reduction. The rise of solutions like GitHub Copilot demonstrates the growing demand for AI-driven code assistance. This tool builds on those ideas while integrating real-time debugging, autocompletion, and multi-language support to address common pain points more effectively.
For real-world deployment, the application will first launch as a local prototype, followed by cloud-based hosting (GCP) for broader accessibility. A public API and web app will make it available to a wide audience, from beginners learning to code to professionals handling complex projects.

Expected Outcomes:

- **Faster coding**: Automating repetitive tasks allows developers to focus on problem-solving.
- **Fewer bugs**: Real-time syntax checking and AI-assisted debugging improve code quality.
- **Better learning**: Beginners benefit from AI-generated suggestions and explanations.
- **Scalability**: Cloud integration ensures the tool can handle multiple users efficiently.

Feasibility

The plan is realistic and already in action, with a working prototype ready. Since users input their own code, there's no need for massive datasets—though open-source code from platforms like GitHub could help refine AI-generated suggestions.
Having strong expertise in Python, Flask, and AI integration, makes implementation smooth. Collaborations with language experts (for Rust, Go, etc.) and cloud providers could help scale the project.
Some challenges include API rate limits and dependencies on compilers like GCC and Rustc, which might require careful handling. However, the overall approach ensures efficient and practical execution without the need to train a separate LLM.

Use of AI

AI is the heart of this project, powered by Gemini 2.0 Flash to help with code generation, debugging, autocompletion, and syntax checking. It understands natural language, making coding simpler and more intuitive. It spots errors in real time, helping developers fix issues quickly and write cleaner code. Autocompletion and syntax checking minimize mistakes and speed up development. With smart suggestions based on context, the AI makes coding more accurate and efficient.

Alternatives Considered

- **Non-AI Approach:** Manual rule-based code generation and static analysis—less flexible and slower to adapt to new languages.
- **Other AI Models:** OpenAI GPT or GitHub Copilot APIs—costlier and less integrated with Flask workflows.
- **Client-Side Execution:** Moving compilation to the browser (e.g., via WebAssembly)—faster but limited by browser constraints and security risks.

The chosen design balances server-side control, AI power, and multi-language support.

References and Appendices

- **References**:
  - Flask Documentation (flask.palletsprojects.com)
  - Google Generative AI SDK (ai.google.dev)
  - Pylint Documentation (pylint.pycqa.org)
- **Demos**: Current code runs locally at http://localhost:5000 with endpoints like /generate-code and /compile.