# CodeClash: LLM Code Generator and Comparator

Palveet Kaur Saluja
University of Illinois, Urbana-Champaign
Champaign, United States
psaluja2@illinois.edu

Rudrik Sanmukhlal Patel
University of Illinois, Urbana-Champaign
Champaign, United States
rudrikp2@illinois.edu

## Abstract

CodeClash aims to revolutionize the coding experience by creating a versatile tool that streamlines code generation and comparison. Currently, users face the cumbersome task of navigating multiple platforms to obtain and manipulate code, often with limited editing capabilities. Our solution not only saves time but also enhances accessibility and efficiency. Through seamless integration with advanced language models like ChatGPT and Gemini, users can effortlessly articulate coding challenges in natural language, receiving instant, editable code outputs. One of the key features of our tool is its ability to facilitate side-by-side comparisons between code generated by different models. This empowers users to evaluate the efficacy and accuracy of solutions, fostering a deeper understanding of coding principles.

***Keywords:*** Large Language Models, ChatGPT, Gemini
In . ACM, New York, NY, USA, 5 pages.

## 1 Introduction

In today's fast-paced digital landscape, the demand for efficient and accessible coding solutions continues to soar. Recognizing this need, we present a web-based application poised to transform the code generation and execution process through the utilization of Language Large Models (LLMs). With the advent of advanced LLMs such as GPT-3.5/4 and Gemini, our platform harnesses the power of natural language processing to streamline the coding experience like never before. Our application boasts a myriad of capabilities designed to empower users at every stage of the coding journey. From seamlessly generating code across more than 30+ programming languages to facilitating real-time editing and execution, our platform sets a new standard for coding efficiency and accessibility.

One of the standout features of our application lies in its ability to provide users with a comprehensive comparison of outputs from various LLMs. By offering side-by-side analyses of solutions generated by different models, including GPT-3.5/4 and Gemini, users gain invaluable insights into the strengths and nuances of each approach. This unique selling point not only enhances the decision-making process but also fosters a deeper understanding of coding principles.

In this report, we delve into the intricacies of our code generation and execution platform, exploring its purpose, capabilities, and unique selling points. From the underlying architecture to practical implementation details, we provide a comprehensive overview of our application's functionality. Furthermore, we showcase real-world evaluation results and offer detailed instructions on how to leverage our platform to its fullest potential. By presenting this innovative web-based solution, we aim to revolutionize the coding landscape and empower users to unleash their full potential in the digital age.

## 2 Motivation

In today's digital landscape, the proliferation of Language Large Models (LLMs) has undoubtedly revolutionized the way we approach coding tasks. However, despite the remarkable advancements in natural language processing, a significant gap remains in the accessibility and usability of these platforms. The crux of the issue lies in the inherent limitations of current LLM platforms, which often lack the functionality to facilitate editable or comparable code generation directly within their interfaces.

This glaring problem underscores the pressing need for a transformative solution that simplifies the code generation process while empowering users to seamlessly edit and compare code outputs. At its core, our motivation stems from a deep understanding of the user needs within the coding community. Coders at all levels, from novice learners to seasoned professionals, crave a tool that not only accelerates the code generation process but also fosters a deeper understanding of coding principles through comparative analysis.

The significance of our proposed solution, CodeClash, cannot be overstated. By offering a user-friendly platform that bridges the gap between code generation and comparison, we aim to revolutionize the coding experience for users worldwide. CodeClash not only saves invaluable time by streamlining the code generation process but also enhances the learning and development journey for coders of all skill levels.

Currently, users are burdened with the cumbersome task of navigating multiple LLM platforms to obtain code solutions, with limited opportunities for editing or comparison. Our tool eliminates this inefficiency by consolidating code generation, editing, and comparison functionalities within a single, intuitive interface. By providing editable and comparable code outputs directly within our platform, CodeClash offers a game-changing solution that empowers users to iterate, refine, and optimize their code with unprecedented ease.

Moreover, CodeClash goes beyond mere convenience—it serves as a catalyst for innovation and collaboration within the coding community. The ability to analyze and compare outputs from different LLMs, such as ChatGPT and Gemini, opens up new avenues for research and experimentation, shedding light on the efficacy and accuracy of various models for specific coding tasks.

## 3 Intended Users

CodeClash is a versatile platform designed to cater to a diverse audience of aspiring coders and seasoned professionals alike. Whether you're just starting out on your coding journey or looking to refine your existing skills, CodeClash offers an intuitive and accessible interface that simplifies the code generation process. For beginners, CodeClash provides a supportive learning environment, with editable code outputs and side-by-side comparisons to facilitate experimentation and understanding. Experienced coders benefit from the platform's efficiency and productivity enhancements, leveraging advanced Language Large Models (LLMs) like ChatGPT and Gemini to streamline their workflow and optimize code performance.

## 4 Architecture

CodeClash's architecture is designed to facilitate seamless user interaction, robust evaluation metrics, and efficient code execution, ensuring a comprehensive and streamlined coding experience. The interface provides users with intuitive input fields for describing coding problems, specifying user inputs, and defining expected outputs. This ensures accurate code generation tailored to the user's requirements.

CodeClash incorporates runtime metrics such as execution time, memory usage, and success status to provide users with valuable insights into the performance of their code solutions. For example, the platform may display a message like "Code compiled and executed in 11ms, using 1024 kb of memory, with a status of success," offering users a detailed assessment of their code's efficiency and functionality. CodeClash utilizes React.js and TailwindCSS to create a responsive and visually appealing user interface. The backend operations of CodeClash are powered by Python and Flask, offering a lightweight and flexible framework for server-side logic. APIs from OpenAI and Google are integrated into the backend architecture to leverage the capabilities of advanced Language Large Models (LLMs) for code generation and comparison.

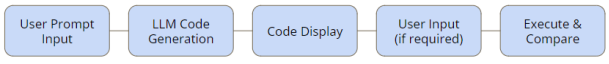


**Figure 1.** CodeClash Workflow

CodeClash utilizes the Judge0 API, supported by RapidAPI, for code compilation and execution. This API infrastructure ensures reliable and efficient execution of code across multiple programming languages, enabling users to test and validate their solutions with ease. To enrich the coding and editing experience, CodeClash incorporates the Monaco Editor, a feature-rich code editor from Visual Studio Code (VSCode). The Monaco Editor provides users with essential features such as syntax highlighting, code completion, and real-time error checking, empowering them to write and refine their code with confidence.

## 5 Approach and Implementation

Our approach to developing CodeClash is anchored in the integration frameworks to provide users with a seamless and intuitive coding experience. Central to our platform's functionality is the utilization of powerful APIs from OpenAI and Google, tapping into the capabilities of Language Large Models (LLMs) to facilitate code generation and comparison. Leveraging Flask on the backend, a lightweight and flexible web framework for Python, we orchestrate the core functionality of the platform, efficiently handling user requests, managing data processing tasks, and communicating with external APIs to ensure a responsive user experience. This backend architecture forms the bedrock of CodeClash, enabling the smooth integration of frontend components and backend logic.

On the frontend, we harness React for building user interfaces, with Tailwind CSS to craft a sleek and modern UI design. React's component-based architecture fosters modularity and reusability across different sections of the application, while Tailwind CSS's utility-first approach enables rapid prototyping and iteration on the visual design. In addition to core code generation and comparison features, CodeClash offers comprehensive support for code compilation in over 30+ programming languages. This is made possible through the integration of Monaco, the feature-rich code editor from Visual Studio Code (VSCode). By embedding Monaco within our platform, users benefit from a familiar and powerful environment for editing and refining their code solutions, enhancing usability and facilitating efficient code editing and debugging. Furthermore, in the user interface, we provide Prompt window where users enters the problem they need to code with details such as question description, input and output constraints, and other parameters to create strong and detailed prompts.

## 6 How to run the software

To effectively utilize the CodeClash software, follow these step-by-step instructions:

1. Begin by cloning the CodeClash GitHub repository from the following link: https://github.com/Palveet/CS510-CodeClash. This will provide you with the necessary source

code and files to set up the application locally on your machine.

2. Navigate to the cloned repository directory and execute npm install in your terminal. This command installs all required React libraries and dependencies necessary for the frontend of the application.

3. Next, install Flask, CORS, Google Generative AI, and OpenAI by executing pip install flask cors google.generative_ai openai in your terminal. This step ensures that the backend server and API integrations are properly set up.

4. In the /backend/app.py file, add your API keys for OpenAI and Gemini. These keys are essential for accessing the respective APIs and enabling code generation functionality within the application.

5. Create a .env file in the root directory of the project to store the RapidAPI host and key. Add the following variables to the .env file:

REACT_APP_RAPID_API_HOST=YOUR_HOST_URL
REACT_APP_RAPID_API_KEY=YOUR_SECRET_KEY
REACT_APP_RAPID_API_URL=YOUR_SUBMISSIONS_URL

Replace YOUR_HOST_URL, YOUR_SECRET_KEY, and YOUR_SUBMISSIONS_URL with your RapidAPI host URL, secret key, and submissions URL respectively.

6. To start the React frontend, execute npm run start in your terminal. This command initiates the development server and launches the CodeClash application in your default web browser.

7. Similarly, start the Flask backend server by executing flask –app app –debug run in your terminal. This command runs the Flask application in debug mode, ensuring any errors or issues are promptly identified and addressed.

8. Once both the frontend and backend servers are up and running, you can begin using the CodeClash tool by entering prompts in the designated text area. Input your coding challenges or descriptions, and let the application's advanced language models generate and compare code solutions for you.

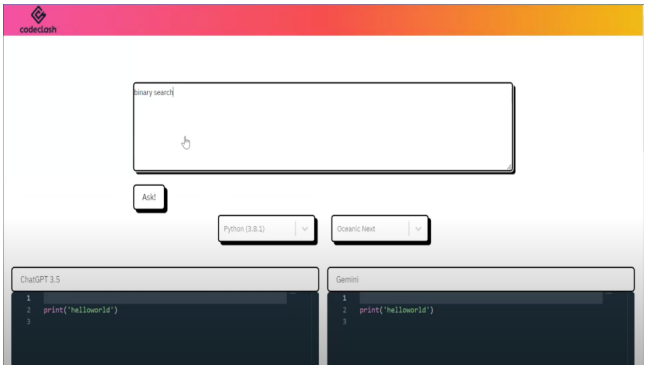


**Figure 2.** Prompt window for adding prompts for the LLMs

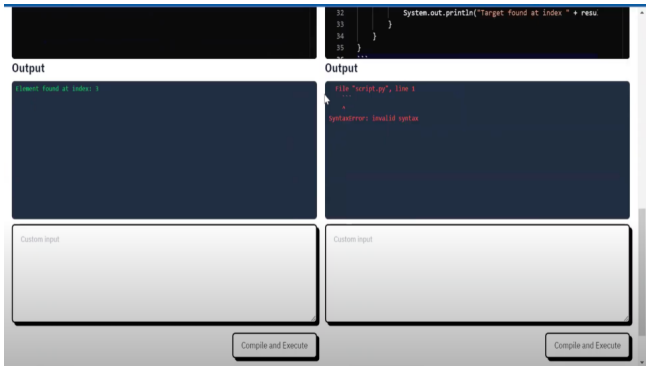


**Figure 3.** Output window with input window and execution button and details


**Figure 4.** Language Options

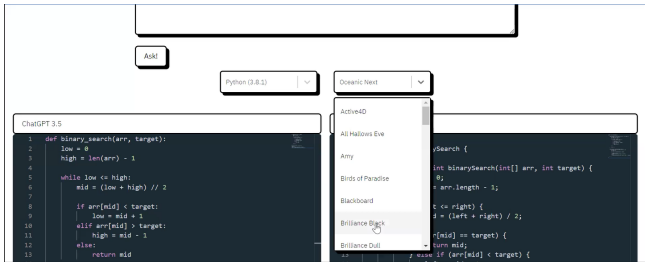


**Figure 5.** Theme Options

## 7 Evaluation

In our comprehensive evaluation of the CodeClash tool, we rigorously tested its capabilities across 25 diverse programming problems spanning a wide range of difficulties. For this assessment, we leveraged two advanced language models: GPT and Gemini. GPT demonstrated exceptional prowess, delivering correct solutions for an impressive 22 out of the 25 problems. Remarkably, all 22 of GPT's solutions exhibited flawless syntax, adhering to the strictest coding standards and best practices.

While Gemini's logical reasoning was sound in 20 of the solutions, it encountered syntax errors or lacked proper example usage in 11 of those cases. This highlights a potential area for improvement in Gemini's output formatting and code documentation.

However, it is noteworthy that both GPT and Gemini excelled at solving common, easier problems, providing correct logical approaches more often than not. Where GPT truly shone was in tackling the more challenging problems, outperforming Gemini in terms of solution accuracy and efficiency.

Furthermore, when explicitly prompted to generate optimized code, GPT's solutions exhibited a higher degree of optimization and efficiency compared to Gemini's output, reflecting its ability to tailor its responses to specific user requirements.


**Figure 6.** Sample User Prompt in python


**Figure 7.** Python Generated Code

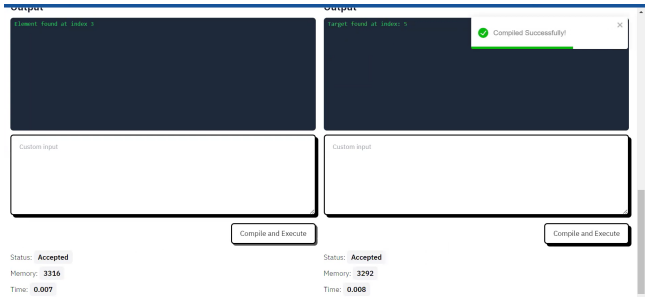


**Figure 8.** Evaluation of python Generated Code

## 8 Group Member Contribution

In the development of CodeClash, our team, comprised of Palveet (psaluja2) and Rudrik (rudrikp2), collaborated closely to ensure the success of the project. While Palveet led the frontend development, utilizing React.js and TailwindCSS to craft an intuitive user interface, Rudrik spearheaded the backend development, leveraging Python and Flask to implement core functionality and integrate APIs from OpenAI and Google. Despite our distinct roles, we worked together seamlessly, sharing insights, troubleshooting issues, and making architectural decisions collectively. Our collaborative efforts extended beyond individual responsibilities, enabling us to deliver CodeClash—a user-friendly coding platform that streamlines code generation and comparison, empowering users to enhance their coding experience efficiently.

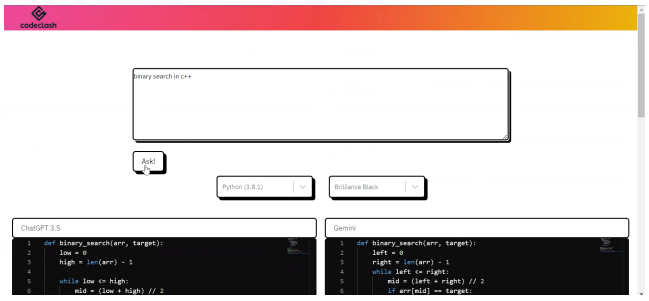


**Figure 9.** Sample User prompt in c++

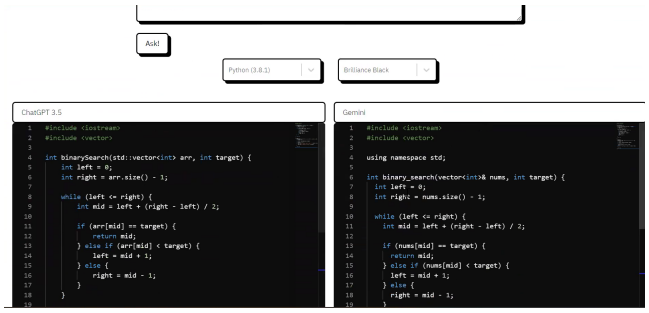


**Figure 10.** C++ generated code

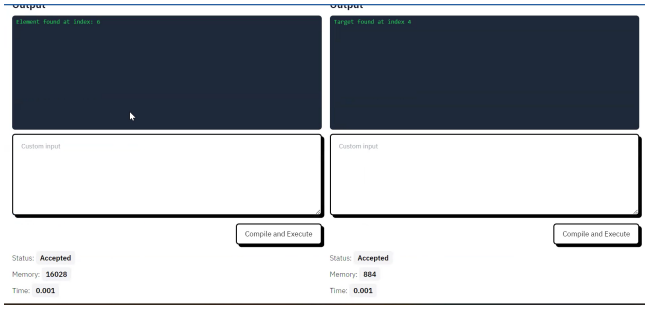


**Figure 11.** Evaluation of C++ generated code

## 9 Deployment

We opted not to deploy CodeClash due to the potential costs associated with using the OpenAI API. As the API may incur charges for usage, we prioritize transparency and cost-consciousness in our deployment decisions. Instead, we encourage users to follow the instructions outlined in the "How to Run the Software" section of this report to set up and run the software locally on their machines. By running CodeClash locally, users can enjoy the full functionality of the application without incurring any additional expenses, ensuring accessibility and affordability for all.

## 10 Future Work and Improvements

To further enhance the capabilities of CodeClash, several avenues for future work and improvement have been identified. Firstly, the integration of additional Language Large Models (LLMs) such as Claude, Llama3, and Mistral presents an exciting opportunity to expand the range of code generation options available to users. By leveraging a cloud-hosted LLM service like Groqcloud, we can tap into a diverse array of models to provide users with even more accurate and varied code solutions.

In terms of the development roadmap, several key features are planned for implementation in future iterations of CodeClash. This includes the introduction of user accounts, allowing users to personalize their experience and access enhanced functionality such as saving prompts and generated code snippets. Additionally, the integration of code sharing features and interactive pair programming capabilities will foster collaboration and knowledge sharing among users, enabling real-time collaboration on coding projects.

One significant enhancement is the creation of a dataset comprising user prompts and generated code snippets. By allowing users to save their prompts and generated code, CodeClash can serve as a valuable resource for research and analysis in the field of natural language processing and code generation.

In terms of user experience, several enhancements are planned to improve usability and convenience. This includes the implementation of a login and registration module for personalized experiences, custom user dashboards similar to platforms like CodePen, and features such as bookmarking favorite code snippets and sharing code snippets over the internet. Additionally, the introduction of collaborative coding capabilities using Socket Programming will enable users to collaborate on coding projects in real-time, enhancing productivity and fostering teamwork.

Despite these exciting prospects for future development, it's important to acknowledge the limitations of CodeClash in its current state. These include restrictions imposed by Judge0's basic plan, which limits the number of requests per day, as well as the cost associated with using the OpenAI API and the anticipated charges for Gemini's services in the future. However, by addressing these limitations and continuing to innovate and improve, CodeClash has the potential to become an indispensable tool for coders at all levels.

## 11 Conclusion

In conclusion, the development of CodeClash represents a significant step forward in streamlining the code generation and comparison process using Language Large Models (LLMs). By leveraging advanced technologies and frameworks, our project offers a user-friendly platform that empowers coders of all skill levels to efficiently generate, edit, and compare code solutions across multiple programming languages. While our focus has been on local deployment to ensure accessibility and cost-effectiveness, future iterations of CodeClash hold promise for even greater functionality and usability enhancements, including the integration of additional LLMs, collaborative coding features, and personalized user experiences. Despite current limitations such as API usage restrictions and potential costs, CodeClash stands as a testament to the power of collaboration, innovation, and user-centric design in driving forward the coding landscape. As we continue to refine and expand upon our project, we remain committed to empowering coders worldwide to unlock their full potential and thrive in the ever-evolving world of programming.

## 12 Inspiration and credit

The inception of CodeClash was fueled by inspiration drawn from innovative projects in the realm of program synthesis using Language Large Models (LLMs). In particular, the Codescope project served as a beacon of inspiration, showcasing the transformative potential of LLMs in automating code generation tasks. Building upon the pioneering work of projects like Codescope, we embarked on our journey to create CodeClash—a platform that democratizes access to advanced coding solutions and fosters a culture of experimentation and learning within the coding community.

Additionally, we extend our gratitude to Mannu Arora for his contributions to the field of user interface design. Drawing upon his expertise and insights, we adopted elements of his UI design to enhance the usability and visual appeal of CodeClash. https://github.com/manuarora700/react-code-editor