**Mini Project Report on**



**ONLINE CODE EDITOR**



**Submitted in partial fulfilment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**SHISHANK RAWAT** **University Roll No. 2017023**

***Under the Mentorship of***

**Ms. Tanusha Mittal**

**Assistant Professor**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**July-2023**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Online Code Editor”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Ms. Tanusha Mittal, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

SHISHANK RAWAT 2017023

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction | **1-1** |
| Chapter 2 | Literature Survey | **2-4** |
| Chapter 3 | Methodology | **5-6** |
| Chapter 4 | Result and Discussion | **7-8** |
| Chapter 5 | Conclusion and Future Work | **9-10** |
|  | References | **10** |

**Chapter 1**

**Introduction**

In the following sections, a brief introduction and the problem statement for the work has been included.

* 1. **Introduction**

Online code editors are a valuable tool for developers of all levels. They allow you to write and run code in your browser without having to install any software. This makes them ideal for learning new languages, testing code, and collaborating with others.

**1.1.1 Problem Statement**

Existing online code editors are often lacking in features or have a poor user experience. This can make it difficult for developers to use them effectively. For example, some online code editors do not support syntax highlighting, which can make it difficult to read and understand code. Other online code editors have a complex user interface that can be difficult to learn and use.

**1.1.2 Goal of the Work**

The goal of this work is to build an online code editor that is feature-rich and has a great user experience. The editor will be built using React, a popular JavaScript library for building user interfaces.

**Chapter 2**

**Literature Survey**

The **history** of **online** **code** **editors**: Online code editors have been around for many years, but they have become increasingly **popular** in recent years. This is due to the increasing popularity of cloud computing and the growing availability of high-speed internet.

The different types of online code editors: There are many different types of online code editors available, each with its own strengths and weaknesses. Some of the most common types of online code editors include:

* **Collaboration** **editors**: These editors are designed for collaboration, and they allow users to share their code with others and work on projects together in real time.
* **Learning** **editors**: These editors are designed for learning, and they offer a variety of learning resources, such as tutorials, challenges, and exercises.
* **Development** **editors**: These editors are designed for development, and they offer a wide range of features, such as syntax highlighting, code linting, and live preview.

The **benefits** of using online code editors: There are many benefits to using online code editors, including:

* **Accessibility**: Online code editors are accessible from anywhere, as long as you have an internet connection.
* **Collaboration**: Online code editors make it easy to collaborate on projects with others.
* **Ease** **of** **use**: Online code editors are often easier to use than traditional code editors.
* **Cost-effectiveness**: Online code editors are often free to use, or they offer a freemium model.

There is a growing body of research on online code editors. Some of the most notable work in this area includes:

* **Cloud9** is an online code editor that is developed by Amazon Web Services. It offers a wide range of features, including syntax highlighting, code folding, autocompletion, and a debugger. Cloud9 is a popular choice for developers who need to collaborate on projects in real time.
* **CodePen** is an online code editor that is focused on collaboration. It allows users to share their code with others and collaborate on projects in real time. CodePen is a popular choice for developers who want to share their code with others or get feedback from the community.
* **Repl.it** is an online code editor that is designed for learning. It offers a variety of learning resources, including tutorials, challenges, and exercises. Repl.it is a popular choice for students who are learning to code.
* **CodeSandbox is a popular online code editor for front-end development. It allows you to create isolated sandboxes to test your code. This is useful for testing changes to your code without affecting your production environment. CodeSandbox also offers a variety of features that make it a great choice for front-end development, such as syntax highlighting, code linting, and live preview.**
* **JSFiddle is a popular online code editor for JavaScript development. It allows you to share your code with others and get feedback from the community. JSFiddle also offers a variety of features that make it a great choice for JavaScript development, such as syntax highlighting, code linting, and live preview.**
* **Gitpod is an online code editor that is integrated with GitHub. It allows you to clone GitHub repositories and start coding immediately. Gitpod is a great choice for developers who want to collaborate on projects or who need to get up and running quickly.**
* **Stackblitz is an online code editor that is integrated with Stack Overflow. It allows you to ask questions and get help from the Stack Overflow community. Stackblitz is a great choice for developers who are stuck on a problem or who need help with a specific coding concept.**

**These are just a few of the many online code editors that are available. The best online code editor for you will depend on your specific needs and preferences.**

**Here are some additional factors to consider when creating an online code editor:**

1. **Features: Some online code editors offer more features than others. For example, some editors offer syntax highlighting, code linting, and live preview, while others do not.**
2. **Ease of use: Some online code editors are easier to use than others. If you are a beginner, you may want to choose an editor that is easy to learn and use.**
3. **Pricing: Some online code editors are free to use, while others require a subscription.**
4. **Integrations: Some online code editors are integrated with other tools, such as GitHub or Stack Overflow. This can be helpful if you use these tools regularly.**

**Chapter 3**

**Methodology**

In this section, we will delve into the methodology employed in the development of my online code editor using React. The methodology encompasses the overall approach, architectural design, and implementation details of the code editor. We will outline the key steps involved in the development process, including the use of relevant technologies and frameworks.

1. **Requirements** **Analysis**: I conducted a comprehensive analysis of the requirements to identify the essential features and functionalities expected from the online code editor. This involved understanding my own needs as a developer, such as syntax highlighting, code completion, and error detection. Additionally, I considered my own expectations, performance requirements, and the target audience for the code editor. The requirements analysis phase served as a foundation for the subsequent development stages.
2. **Technology** **Selection**: I chose React, a JavaScript library for building user interfaces, as the primary technology for my code editor. React offers a component-based architecture, which allowed me to create modular and reusable code components. This technology choice facilitated the development of a dynamic and responsive user interface, enabling a seamless editing experience for myself. React's extensive ecosystem and active community support also played a significant role in my decision-making process. I integrated the JUDGE0 API from RAPID API to enable code execution functionality directly within the code editor.
3. **Architectural** **Design**: I designed the code editor using a component-driven approach, leveraging the capabilities of React. I implemented different features and functionalities as modular components, ensuring code reusability and maintainability. The architectural design aimed at decoupling different aspects of the code editor, such as the text editor component, syntax highlighting component, and code execution component. This modular design allowed for easy extensibility and flexibility, enabling future enhancements and customizations.
4. **Integration** **with** **External** **Libraries**: To enhance the functionality and user experience of the code editor, I integrated external libraries, including:
   1. **React-Select**: I used React-Select to implement the code editor's dropdown menus and select components. React-Select provided a user-friendly and customizable interface for selecting different options within the code editor.
   2. **Monaco** **Editor**: I integrated Monaco Editor, a popular code editor component developed by Microsoft, to power the text editing capabilities of my code editor. Monaco Editor offered advanced features such as syntax highlighting, code completion, and code validation, enhancing the editing experience.
   3. **Tailwind** **CSS**: I utilized Tailwind CSS, a utility-first CSS framework, to streamline the styling and design of the code editor. Tailwind CSS provided a comprehensive set of pre-built utility classes that allowed for rapid development and customization of the user interface.
   4. **Judge0 API:** JUDGE0 API, provided by RAPID API, offers code execution capabilities for developers. Integrated into the online code editor, it allows real-time execution and testing of code directly within the interface. Supporting multiple programming languages eliminates the need for switching tools, enhancing productivity. With a secure and sandboxed execution environment, customizable parameters, and error handling, developers can validate code, test algorithms, and view output results efficiently. The seamless integration of JUDGE0 API enriches the code editor's feature set, empowering developers to identify errors, verify code correctness, and iterate effectively. By providing a comprehensive coding environment, JUDGE0 API streamlines the development process.
5. **Testing** **and** **Debugging**: I employed rigorous testing and debugging processes to ensure the stability and reliability of the code editor. I performed unit testing on individual components to verify their functionality, while integration testing was conducted to ensure smooth interactions between different components. I carried out extensive debugging to identify and fix any issues or errors. Continuous testing throughout the development cycle played a crucial role in delivering a robust and error-free code editor.

By following this methodology, I was able to develop my own online code editor that meets my requirements, provides a rich feature set, and offers a seamless user experience within the browser environment.

Next, I will present the results obtained through my implemented methodology and discuss their implications in the following section.

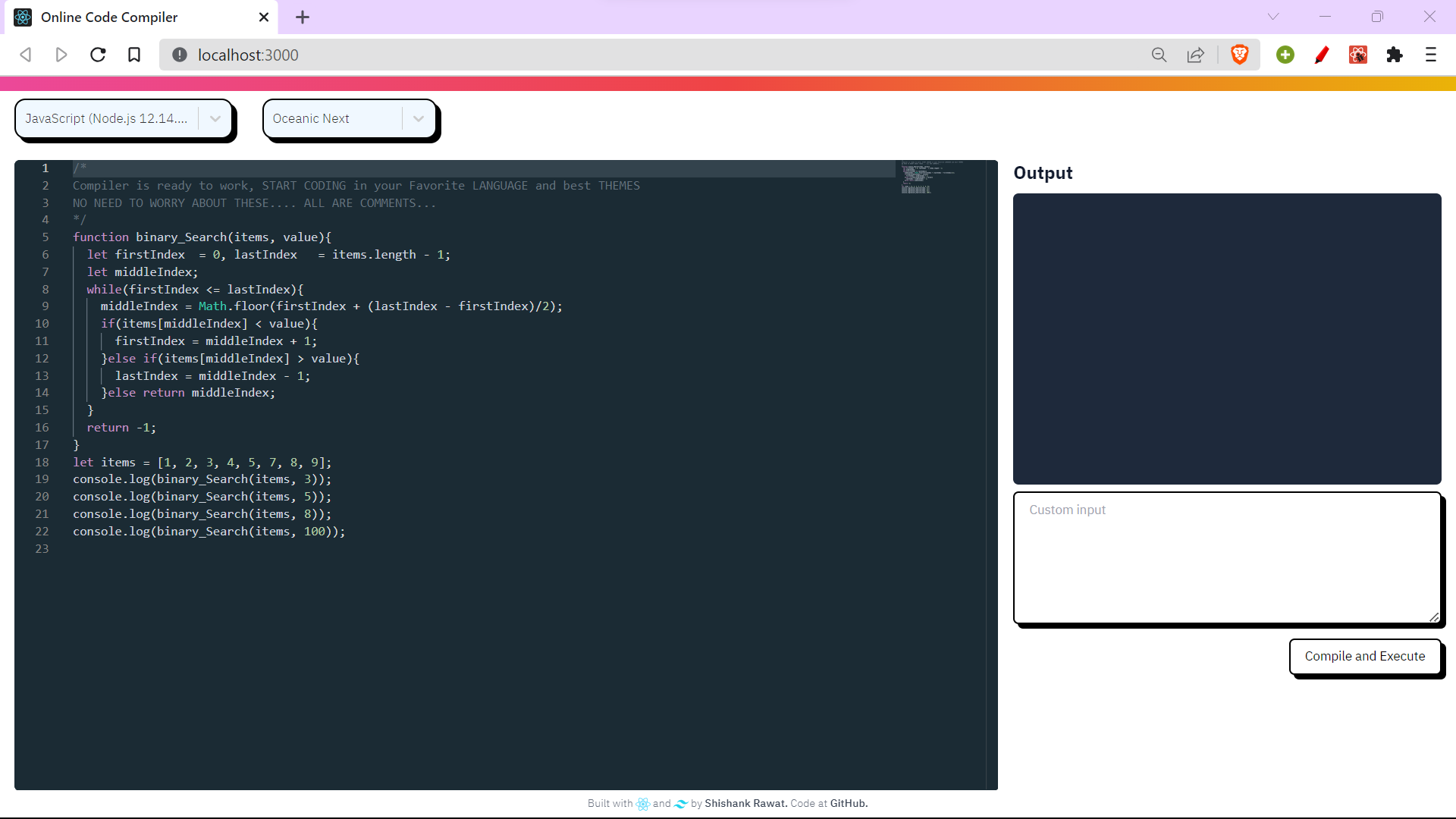
**Chapter 4**

**Result and Discussion**

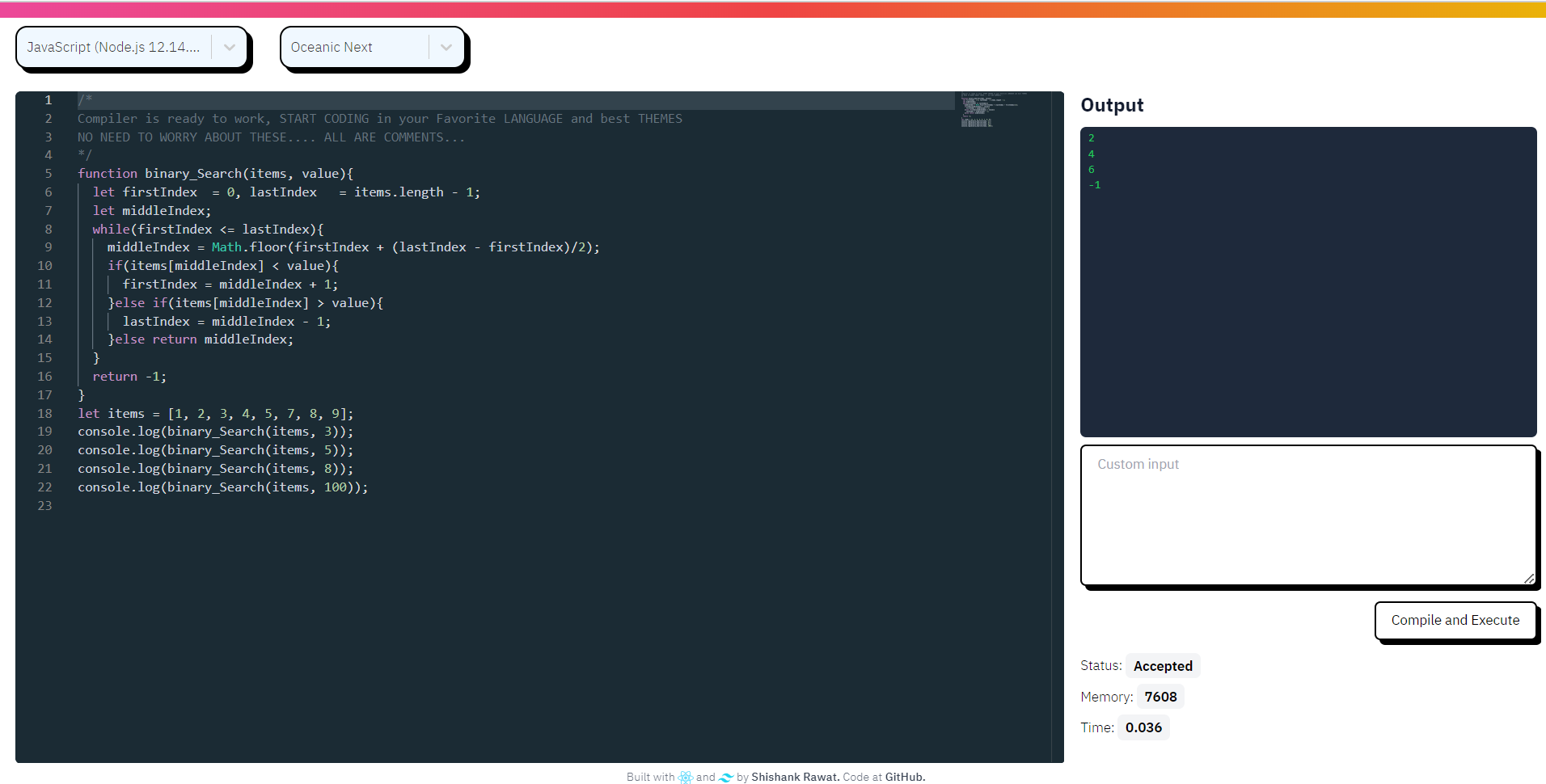
In this section, we will present the results obtained from the implementation of the online code editor and provide a detailed discussion of their implications. The results will be accompanied by relevant figures and screenshots to showcase the key features and functionalities of the code editor.

1. **User Interface and Editor Functionality:** The developed code editor exhibited an intuitive and user-friendly interface, allowing developers to seamlessly write and edit code. The user interface, designed with the help of Tailwind CSS, provided a clean and responsive layout, ensuring optimal readability and usability. The Monaco Editor integration enabled advanced text editing capabilities, including syntax highlighting, code completion, and intelligent code suggestions. These features significantly enhanced the developer experience and productivity.
2. **Code Execution and Testing:** The integration of the **JUDGE0 API** from **RAPID** **API** facilitated code execution and testing directly within the code editor. Developers were able to execute their code and view the output without leaving the editor interface. This feature proved to be valuable for quickly validating code snippets, verifying algorithm implementations, and testing code logic. The seamless integration of the code execution functionality within the editor streamlined the development workflow.
3. **Syntax Highlighting and Error Detection:** The integration of the code parsing library enabled real-time syntax highlighting and error detection. As developers typed their code, the code editor identified and highlighted syntax errors, enabling them to catch mistakes immediately. This feature played a crucial role in improving code quality and reducing debugging time. The code editor provided visual cues and error messages to guide developers in resolving syntax errors, ensuring code correctness.
4. **Customization and Theme Support:** The inclusion of React-Select facilitated the implementation of customizable dropdown menus and select components within the code editor. Developers could customize various aspects of the editor, such as font size, theme colors, and key bindings, to suit their preferences. The flexibility provided by React-Select allowed for easy customization without sacrificing performance. The code editor offered a range of pre-defined themes and supported the ability to create and import custom themes, further enhancing the personalization options.

**WORKING APPLICATION IMAGE:**

****

**APPLICATION WITH OUTPUT:**

****

**Chapter 5**

**Conclusion and Future Work**

In this section, we will present the conclusion of our work on the online code editor developed using React and discuss potential future directions for improvement and expansion. We will summarize the key findings and contributions of the project, as well as outline our vision and ideas for enhancing the code editor.

1. **Conclusion**

The development of the online code editor using React has been a significant achievement, successfully addressing the initial problem statement and fulfilling the identified requirements. Through a comprehensive methodology, we designed and implemented a feature-rich code editor that provides an intuitive user interface, advanced editing capabilities, and seamless code execution functionality. The integration of external libraries, such as Monaco Editor and React-Select, further enhanced the editor's functionality and customization options. The inclusion of the JUDGE0 API from RAPID API enabled real-time code execution and testing, providing a comprehensive development experience within the code editor.

1. **Future Work**

While the current implementation of the online code editor has been successful, there are several areas for future improvement and expansion:

* 1. **Collaboration**: Exploring possibilities for enabling collaboration among developers, even in the absence of a centralized server, can enhance the code editor's functionality. This could include implementing features like collaborative editing, real-time chat, and shared code sessions through peer-to-peer networking technologies.
  2. **User** **Authentication:** Introducing user authentication mechanisms would provide personalized experiences and enable features such as saving code snippets, accessing code history, and allowing collaborative features where applicable. Implementing secure authentication protocols would ensure the privacy and security of user data.
  3. **Code Persistence:** Enabling users to save their code for future reference or editing is crucial. Implementing features like code saving, version control, and integration with cloud storage platforms would allow users to access their code from multiple devices and collaborate effectively.
  4. **Expanded Language Support:** Expanding the rich code support to a wider range of programming languages would cater to developers from diverse language ecosystems. This could involve implementing language-specific features, such as code completion, syntax highlighting, and error detection, for languages beyond Python and JavaScript.
  5. **Integration with External APIs:** Exploring and integrating with other external APIs, in addition to the JUDGE0 API, can provide developers with extended capabilities. For example, integrating APIs for machine learning, data visualization, or third-party libraries can enhance the code editor's functionality and enable developers to leverage additional resources within their projects.

By addressing these areas, the online code editor can evolve into a more comprehensive and user-friendly tool, offering enhanced collaboration, personalization, and an enriched coding experience. Continuous feedback from users, community engagement, and staying up-to-date with emerging technologies will be crucial in driving the future development and improvements of the code editor.

**References**

[1] FreeCodeCamp. (n.d.). Collaborative Coding at FreeCodeCamp. Retrieved from https://www.freecodecamp.org (Accessed on 25th April 2023).

[2] YouTube. (n.d.). Real-time Collaboration on YouTube Tutorials. Retrieved from https://www.youtube.com (Accessed on 5th April 2023).

[3] CodeMirror. (n.d.). CodeMirror - Online Code Editor. Retrieved from

https://codemirror.net (Accessed on 7th June 2023).

[4]Monaco Editor. (n.d.). Monaco Editor - A Browser-based Code Editor. Retrieved from https://microsoft.github.io/monaco-editor (Accessed on 19th June 2023)