**PROJECT REPORT DOCUMENTATION**

* **Problem Statement**

Creating a website for DSA learners to learn, practice and give contest. It is a platform which lets you write code in your favorite programming language and run that code on the same platform and make it easy to use.

* **Hardware And Software**

The software which is been used in this product is VS Code.

The hardware which is been used in this product is 12th Gen Intel Core i5-1235U | Speed: 10 Cores (2xP-core 1.3/ 4.4GHz, 8xE-core 0.9 / 3.3GHz) | 12 Threads | 12MB Cache, Operating System: Pre-Loaded Windows 11

* **Technology Used**
* [React.js](https://reactjs.org/) – For the front-end
* CSS- For styling
* [Tailwind CSS](https://tailwindcss.com/) – For compiler
* [Judge0](https://judge0.com/) – For compiling and executing our code.
* [Rapid API](https://rapidapi.com/) – For Compiler
* [Monaco Editor](https://www.npmjs.com/package/monaco-editor) – The code editor that powers the project
* **Project Description**

A compiler is a special program that translates a programming language's source code into machine code, bytecode or another programming language.

1. Lexical analysis. The compiler splits the source code into lexemes, which are individual code fragments that represent specific patterns in the code. The lexemes are then tokenized in preparation for syntax and semantic analyses.
2. Syntax analysis. The compiler verifies that the code's syntax is correct, based on the rules for the source language. This process is also referred to as parsing. During this step, the compiler typically creates abstract syntax trees that represent the logical structures of specific code elements.
3. Semantic analysis. The compiler verifies the validity of the code's logic. This step goes beyond syntax analysis by validating the code's accuracy. For example, the semantic analysis might check whether variables have been assigned the right types or have been properly declared.
4. IR code generation. After the code passes through all three analysis phases, the compiler generates an intermediate representation (IR) of the source code. The IR code makes it easier to translate the source code into a different format.
5. The compiler optimizes the IR code in preparation for the final code generation. The type and extent of optimization depends on the compiler. Some compilers let users configure the degree of optimization.
6. Output code generation. The compiler generates the final output code, using the optimized IR cod

* **Scope**

1. Bring best questions for the users.
2. It gives you a solution for the code.
3. 40+ languages have been used in this project to make compiler more easy and fast for users.
4. The user experience is best and interface is also fast.
5. The user can see relevant details of the code that was executed.
6. Once the user is done writing their code, they can compile their code and see the output / results in the output window.
7. It can compile code on a web app with standard input and output with support to over 40 programming languages

* **Challenges Faced**

1. The main challenge was to render our website using API.
2. Although I am not a react student then also i have made a system which works on react.

* **Conclusion**

Fully functional web app is developed that help users increase the DSA knowledge.