IDEATE AND IMPLEMENT A SYSTEM TO ENHANCE THE QUALITY OF EDUATION IN EMERGING TECHNOLOGY (ONLINE COMPIER)

Sumitted By-

Aboli Wankhade(AM21004) **Tanushree Sarode(AM21032)**

Vaishnavi Rahamatkar(AM21014) **Sakshi Mantri(AM21038)**

Abstract—
In the dynamic landscape of Emerging Technologies, rapid prototyping and hands-on coding experiences are pivotal for students' learning journeys. To bolster this educational approach, we introduce an Online Compiler, a web based platform designed to empower learners and educators alike in exploring ,practicing, and mastering programming languages and concepts.
The Online Compiler serves as a virtual coding environment, providing a seamless and accessible space for users to write,compile, and execute code directly from their web browsers. With a user-friendly interface, it caters to diverse learning styles, from beginners taking their first steps in coding to advanced users refining their algorithms. Key features include real-time syntax highlighting, error detection, and instant feedback on code execution, fostering a conducive learning environment. Additionally, the platform supports a wide array of programming languages, ensuring versatility in curriculum integration and accommodating various emerging technology disciplines. Incorporating collaborative tools, the Online Compiler enables students to work on projects together, enhancing teamwork and problem-solving skills crucial in the tech industry. Furthermore, educators can create customized coding exercises, quizzes, and assignments, tailoring the learning experience to meetspecific educational objectives. Through this Online Compiler, we aim to democratize access to quality programming education, transcending geographical barriers and resource limitations. By providing a robust platform for hands-on learning, we envision a future where students, regardless of their backgrounds, can confidently navigate the complexities of Emerging Technologies and contribute meaningfully to the digital era.

Scope-

The system aims to develop a web-based application that offers a collaborative editor, compiler, and execution environment for programming languages. The system will be accessible to users without the need to download or install any software on their local machines. The collaborative editor will support multiple programming languages and provide various features. The compiler will be integrated with the editor and provide real-time feedback to users on any syntax errors or other issues. Additionally, the system will include an execution environment for users to run and test their code. Overall, the system seeks to provide a seamless and accessible programming environment for users of all levels, making it easier for them to collaborate and work on code together.

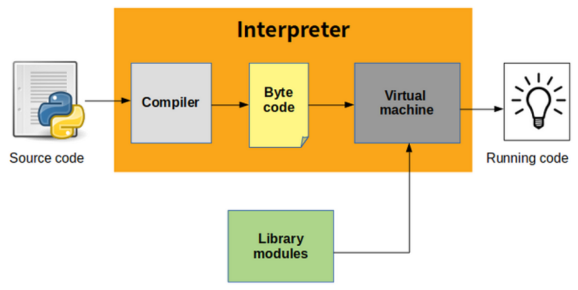
Introduction

The rapid evolution of Emerging Technologies has ushered in a new era of possibilities, promising groundbreaking innovations across various industries. From artificial intelligence to block chain, the landscape is vibrant with opportunities, calling for a workforce equipped with advanced programming skills and a deep understanding of these cutting-edge concepts.In the realm of education, keeping pace with this technological surge poses a significant challenge. Traditional classroom settings often struggle to provide the hands-on,real-time coding experiences essential for mastering these complex technologies. Recognizing this gap, we embark on a mission to revolutionize the educational journey through the development of an Online Compiler. The Online Compiler represents more than just a virtual coding environment; it stands as a gateway to immersive learning in Emerging Technologies. This platform aims to empower students and educators alike by offering a dynamic, interactive space where theoretical knowledge seamlessly transitions into practical application. With the Online Compiler, learners can delve into a diverse array of programming languages, experiment with algorithms, and witness the immediate impact of their code execution. Real-time syntax highlighting and error detection features provide invaluable guidance, transforming every line of code into a learning opportunity.Educators, too, find a powerful ally in the Online Compiler. Through this platform, they can craft tailored coding exercises, quizzes, and assignments that align with the curriculum's emerging technology focus. Moreover, collaborative tools foster teamwork among students, nurturing the collaborative problem-solving skills crucial for success in the tech-driven world.The significance of this Online Compiler extends beyond its technical capabilities. It represents a democratization of access to quality programming education, breaking down geographical barriers and leveling the playing field for aspiring technologists worldwide. Whether in bustling urban centers or remote rural communities, learners now have a passport to the frontier of Emerging Technologies.

Compiler

A compiler is a special program that translates a programming language's source code into machine code, bytecode or another programming language. The source code is typically written in a high-level, human-readable language such as Java or C++.
 A programmer writes the source code in a code editor or an integrated development environment (IDE) that includes an editor, saving the source code to one or more text files. A compiler that supports the source programming language reads the files, analyzes the code, and translates it into a format suitable for the target platform.
Compilers that translate source code to machine code target specific operating systems and computer architectures. This type of output is sometimes referred to as object code (which is not related to object-oriented programming). The outputted machine code is made up entirely of binary bits -- 1s and 0s -- so it can be read and executed by the processors on the target computers.
Some compilers can translate source code to bytecode instead machine code. Bytecode, which was first introduced in the Java programming language, is an intermediate language that can be executed on any system platform running a Java virtual machine (JVM) or bytecode interpreter. The JVM or interpreter converts the bytecode into instructions that can be executed by the hardware processor. A JVM also makes it possible for the bytecode to be recompiled by a just-in-time compiler.
Some compilers can translate source code into another high-level programming language, rather than machine code or bytecode. This type of compiler might be referred to as a transpiler, transcompiler, source-to-source translator or it might go by another name. For example, a developer might use a transpiler to convert COBOL to Java.

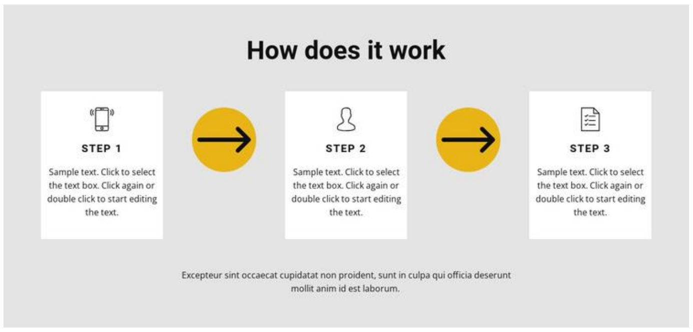
How Compiler Works?



Language

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.
Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

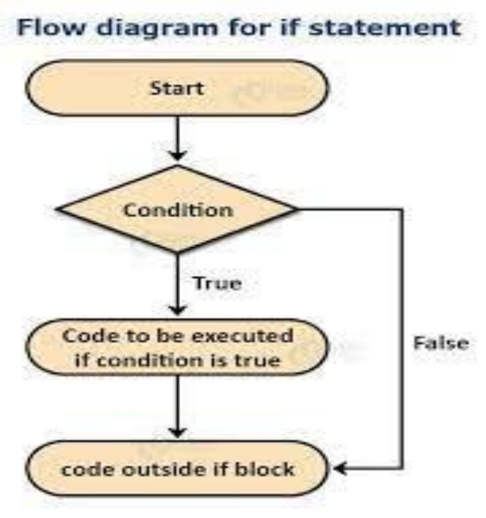


Sample text. Click to select the text box. Click again or double click to start editing the text.

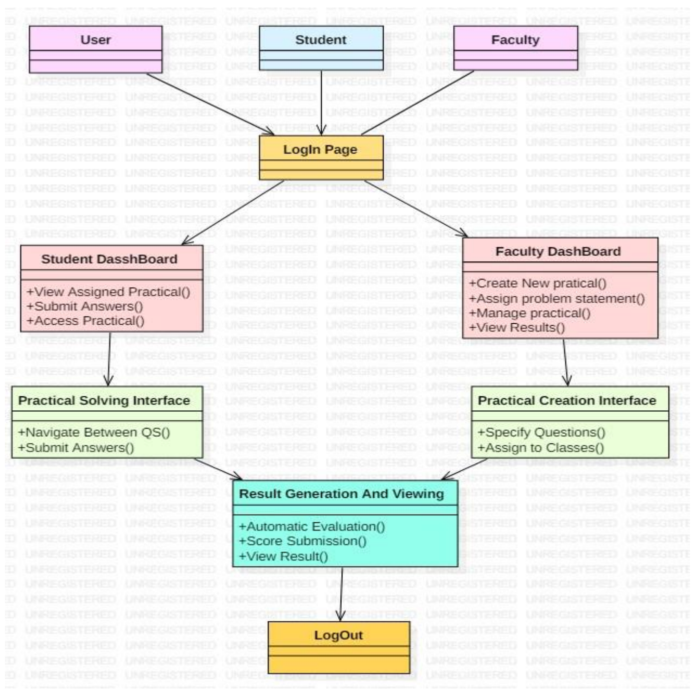
Sample text. Click to select the text box. Click again or double click to start editing the text.

Sample text. Click to select the text box. Click again or double click to start editing the text.

Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.


Flow diagram for if statement
Start
Condition
True
False
Code to be executed if condition is true
code outside if block

Methodology



Login Page:

• Faculty and students log in using their respective credentials.
• Separate login interfaces for faculty and students.

Faculty Dashboard:

• Faculty can create new tests, manage existing tests, and view test results.
• Interface for assigning tests to specific students or groups of
students.

Test Creation:

• Faculty can create tests by specifying questions, options, and
correct answers.
• Ability to set time limits, assign test to specific classes or groups,and customize test parameters.

Student Dashboard:
• Students can view assigned tests and access them for solving.
• Clear interface displaying test instructions and questions.

Test Solving:
• Students can attempt tests within the specified time limit.
• Interface should allow for easy navigation between questions and submission of answers.

Automatic Result Generation:
• System automatically evaluates student responses against correct answers.
• Generates test scores and provides immediate feedback to students upon completion.

Score Submission:

• After completing the test, students can submit their answers for evaluation.
• Scores are automatically recorded and submitted to the faculty.

Result Viewing:

• Faculty can view test results for each student, including scores and detailed breakdown of answers.
• Ability to export results for record-keeping or further analysis.

Characteristics of Online Python Compiler
1. Accessibility: They are accessible via web browsers, making them convenient for users who don't have Python installed locally or need to code on devices where installation isn't feasible.
2. Code Execution: They allow users to write Python code directly in the browser and execute it to see the output.

3. Syntax Highlighting: They often provide syntax highlighting, making it easier for users to identify elements of their code such as keywords, strings, and comments.

4. Output Display: They display the output of the executed code, enabling users to verify the correctness of their programs.

5. Support for Multiple Python Versions: Some online compilers support different versions of Python, allowing users to choose the version that best fits their needs.

6. Error Handling: They provide error messages and traceback information to help users debug their code when errors occur.

7. Code Sharing: Many platforms allow users to share their code with others by providing a unique URL or embedding options.

8. Integration with IDEs: Some online compilers integrate with Integrated Development Environments (IDEs) or text editors, offering additional features such as auto-completion and code suggestions.

9. Security: They often implement measures to ensure the security of the user's code and prevent malicious

activities.

10. Community and Support: Some platforms offer community forums or support channels where users can ask questions, seek help, and share knowledge with others.

Implementation of Online Python Compiler-

1. User Interface: Design a user-friendly interface where users can input their Python code and interact with the compiler.

2. Code Execution Environment: Set up an environment to execute Python code safely and securely. This often involves using sandboxing techniques to prevent malicious code from affecting the server.

3. Parsing and Compilation: Parse the user's input code and compile it into a format that can be executed by the Python interpreter.

4. Execution: Execute the compiled code in a controlled environment to prevent security risks and ensure stability.

5. Output Handling: Capture the output of the executed code and display it to the user in a readable format.

6. Error Handling: Implement mechanisms to handle errors gracefully, providing informative error messages to users when their code contains syntax or runtime errors.

7. Support for Python Versions: Depending on the requirements, support multiple versions of Python by configuring the execution environment accordingly.

8. Security Measures: Implement security measures to protect against code injection attacks and other security vulnerabilities.

9. Optimization: Optimize the compiler's performance to ensure fast and responsive execution of code, even under heavy user load.

10. Integration and Deployment: Integrate the compiler into a web server environment and deploy it to a hosting provider to make it accessible to users over the internet.

Advantages-

1. Interactive Learning Environment:Create a user-friendly interface that allows students to write, compile, and execute code directly within the browser. This provides hands-on experience and immediate feedback.
2. Support for Multiple Languages: Include support for various programming languages commonly used in emerging technologies such as Python, JavaScript, Java, C++, etc. This ensures versatility and accommodates a wide range of courses.
3. Real-time Collaboration: Enable collaborative coding sessions where multiple users can work on the same code simultaneously. This fosters teamwork and facilitates peer learning.
4. Auto-Grading and Feedback: Implement automatic code evaluation to provide instant feedback on the correctness and efficiency of the code written by students. This helps them identify and correct errors more efficiently.
5. Integration with Learning Management Systems (LMS): Integrate the online compiler with existing learning management systems used by educational institutions. This streamlines the workflow for both students and instructors by centralizing course materials and assignments.
6. Code Visualization and Debugging Tools: Incorporate tools for visualizing code execution and debugging, such as step-by-step execution, variable inspection, and error highlighting. This enhances understanding and problem-solving skills.

Future Work

To continue enhancing our system, future work includes improving the scalability and performance of our live code share feature, enhancing the accuracy and relevance of the error suggestion API, and expanding support for specialized programming languages and frameworks. We also plan to gather user feedback through testing and continually improve the user experience while addressing any issues or limitations.

Acknowlegement

1. Online Compiler Integration: Partner with reputable online compilers specialized in emerging technologies like Python, Java, or machine learning. Integrate these compilers into educational platforms used by students and researchers.
2. Usage Tracking: Implement a tracking mechanism to record when students or researchers use the online compilers for their projects or experiments. This could involve logging compiler usage data such as frequency, duration, and the specific technologies used.
3. Quality Assessment: Develop criteria to assess the quality of code written using the online compilers. This could include factors like efficiency, correctness, readability, and adherence to best practices in the respective technology domain.
4. Feedback Loop: Provide feedback to users based on the quality assessment, highlighting areas for improvement and suggesting resources or additional learning materials to enhance their understanding of the technology.
5. Acknowledgement System: Introduce a citation mechanism within the research paper submission process. When researchers use the online compilers for code implementation in their papers, they would be required to acknowledge the compiler platform used, providing a

citation or reference link.

6. Recognition and Rewards: Establish a recognition system to incentivize students and researchers who consistently produce high-quality code using the online compilers. This could include certificates, badges, or even scholarships for outstanding contributions.

7. Community Engagement: Foster a community around the online compiler platform where users can share their projects, collaborate on code development, and provide peer feedback to further improve the quality of education and research in emerging technologies.

By implementing such a system, you can not only enhance the quality of education in emerging technologies but also encourage the responsible use of online compilers in academic research, fostering a culture of transparency and acknowledgment within the academic community.

Conclusion-

The implementation of our system to enhance the quality of education in emerging technology holds significant promise for bridging the gap between traditional education and the rapidly evolving demands of the digital age. By leveraging interactive online platforms, personalized learning experiences, real-world applications, and ongoing professional development for educators, we aim to empower students with the necessary skills and knowledge to thrive in a dynamic technological landscape. Through continuous evaluation and refinement, we aspire to adapt our system to meet the evolving needs of learners and educators alike, ensuring that our educational endeavors remain at the forefront of innovation and excellence. With a steadfast commitment to collaboration, adaptability, and student-centered learning, we believe our system will serve as a catalyst for transformative change in education, empowering individuals and communities to unlock their full potential in the digital era.

References-

1. Aho, A. V., & Ullman, J. D. (1977). Principles of Compiler Desig Addison-Wesley.
2. Appel, A. W. (1998). Modern Compiler Implementation in Python.Cambridge University Press.
3. Arnold, K. S., & Gosling, J. (1996). The Python programming language Addison-Wesley.
4. Baumann, R. (2014). Developing web-based IDEs. Journal of We Engineering, 13(2&3), 91-113.
5. Bodik, R., & Gupta, R. (2003). Web-based program analysis tools. I Proceedings of the 2003 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation (pp. 3-9)
6. Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A Burrows, M., ... & Gruber, R. E. (2006). Bigtable: A distributed storage system for structured data. ACM Transactions on Computer System(TOCS), 26(2), 4.
7. Cooper, K. D., & Torczon, L. (2011). Engineering a compiler. Elsevier.
8. Deitel, P., Deitel, H., & Deitel, A. (2017). Java: How to Program (Ear Objects). Pearson.
9. Ferrante, J., Ottenstein, K. J., & Warren, J. D. (1987). The program dependence graph and its use in optimization. ACM Transactions Programming Languages and Systems (TOPLAS), 9(3), 319-349.
10. Fisher, D., Grabski, T., & Walia, G. (2015). Online code editors: they affect coding behaviors and learning outcomes? Journal Information Systems Education, 26(4), 287-296.
11. Fraser, C. W. (1992). A Retargetable C Compiler: Design and Implementation. Addison-Wesley.
12. Henglein, F., & Mossin, C. (1997). Web-based programming in a nontrivial domain. In Proceedings of the 1997 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation (pp. 88-97).
13. Lam, M. S., & Wilson, R. P. (1995). Limits of control flow on parallelism. ACM SIGPLAN Notices, 30(6), 46-57.
14. Palsberg, J., & Schwartzbach, M. I. (1991). Object-oriented type inference. ACM Transactions on Programming Languages and Systems(TOPLAS), 13(2), 237-268.
15. Rauschmayer, A. (2014). Exploring ES6: Upgrade to the next version of JavaScript. " O'Reilly Media, Inc.".
16. Seidl, M., & Zuleger, F. (2001). Web-based programming exercises: A didactic perspective. In Proceedings Seventh Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS'00) (pp. 122-128)