Grenze International Journal of Engineering and Technology, June Issue

Compatible Code Editor

Megha Kadam[1], Amol Deshmukh[2], Shubhankar Badmanji[3], Pavan Gokak[4] and Chetan Sanap[5]
[1-5]Dept of Computer Science, Marathwada Mitramandal's Institute of Technology (MMIT), Maharashtra, Pune

***Abstract*— In today's digital age, software development has become an integral part of various industries and applications. Integrated Development Environments (IDEs) play a pivotal role in facilitating and enhancing the coding experience by providing programmers with tools for code editing, debugging, and project management. In this paper we mentioned the hypothesis of how to create a Lightweight Cross-Platform desktop based integrated Development Environment using the Code Mirror library, a versatile and customizable JavaScript library for code editing. In this paper we write a different research paper's summary which supporting us in developing Lightweight IDE with different ways and features. Furthermore, the Paper discusses the technologies utilized in this project and proposed applications of the IDE across different domains. Moreover, the paper outlines the architecture flow of lightweight IDE. Our project addresses the need for a user-friendly, Cross-Platform desktop based IDE that caters to the diverse requirements of software developers. The Code Mirror library serves as the foundation for our IDE, enabling features such as syntax highlighting, code folding, auto-completion, and theme customization. Users will be able to write, edit, and debug code efficiently, regardless of their choice of programming language and aesthetic preferences.**

I. INTRODUCTION

A. IDE

An Integrated Development Environment (IDE) is a software application designed to offer a wide range of tools and functionalities to aid programmers and developers in the creation, testing, and debugging of code for software development. Common components of an IDE comprise a code editor, a compiler or interpreter, debugging tools, and various utilities, all aimed at enhancing and simplifying the software development process. IDEs are designed to enhance productivity and efficiency in software development by providing a centralized environment for writing, testing, and managing code projects.

1 Key IDE components

a. Code Editor

The central part of an IDE, the code editor allows developers to write, edit, and view the source code of their projects. It often provides syntax highlighting, code completion, and indentation features to enhance code readability and writing efficiency. Code editors are language-specific and offer tools tailored to the programming language being used.

b. Compiler/Interpreter

Many IDEs include a built-in compiler or interpreter that can convert source code into executable programs or run scripts in the case of interpreted languages. This component helps developers catch syntax errors and run their code within the same environment, facilitating the debugging process.

Grenze ID: 01.GIJET.10.2.53

c. Debugger

A distributed network closely resembles a decentralized network as it operates without the need for centralized organization. It is commonly employed to characterize networks that are spread across different geographic locations. The specific processes through which a network reaches decisions and achieves consensus are contingent upon the network's consensus mechanism. These networks exhibit high speed, scalability, improved transparency, and exceptional tolerance.

Code Mirror Libraries (Open source): Code Mirror is a versatile and popular JavaScript library for creating text editors in web applications. It provides a rich set of features for code editing, including syntax highlighting, auto completion, bracket matching, and more.

II. Related Works

A number of proposals have been found for integrated development environment in the literatures. In [7], the literature survey explores P. S. Newman's proposal for an Integrated Development Environment (IDE), focusing on the integration of a Unified Command Language (UCL) and an extended repository. The UCL ensures consistency in both developmental and production settings, enhancing efficiency and collaboration. The extended repository facilitates better data organization, access control, and automated installation processes. Integrating these aspects streamlines software development and deployment, leading to more robust and error-resistant systems. Further research may delve into practical implementation and broader impacts on software development. In [8] this survey explores IDEs designed for program verification, focusing on features like Non-Linear Editing, Multithreading, Dependency Analysis, Caching, Information Showcase, and Integrated Debugging. These features aim to enhance efficiency, agility, and user experience in program verification by enabling flexible code editing, efficient multithreading, reduced wait times through caching, hover text for code elements, error highlighting, debugging with code mirror libraries, and interactive issue resolution. Overall, these IDE features significantly contribute to improving code comprehension, verification, and development efficiency. In [9] the literature survey explores advanced development environments in software engineering, covering program verification IDEs and integrated environments for rule-based expert systems. It discusses innovative features such as non-linear editing, multi-threading, dependency analysis, and caching mechanisms in program verification IDEs. Additionally, it highlights user interaction capabilities and integrated debugging. For rule-based expert systems, it emphasizes the integration of multiple programming languages and database management systems. The survey underscores challenges and suggests potential areas for future research to advance software development practices and expert system construction. In [10**]**, Software development is a complex professional activity that demands a diverse set of skills and qualities from developers. These encompass technical proficiency for coding and interpersonal skills for effective collaboration with peers and stakeholders. The creation of the paragraph discusses the concept of developer experience (DX) in software development, emphasizing the importance of factors such as efficiency, flexibility, informativeness, and intuitiveness in an integrated development environment (IDE). Qualitative findings from a survey study with 45 developers highlight their perceptions of a specific cross-platform IDE, focusing on qualities desired and areas for improvement. Developers prioritize efficiency and flexibility while seeking better integration with external tools and enhanced stability. The study aims to further analyse quantitative data on flow state, intrinsic motivation, and UX to deepen understanding of DX and its facilitators. Further research is needed to explore the relationship between DX and achieving a state of flow for developers' productivity and satisfaction.

III. Proposed System Architecture

The architecture of the integrated development environment (IDE) begins with the initialization of its components and settings upon application start. Users engage with the IDE through a user friendly workspace interface, which serves as a central hub for creating and manipulating code. A crucial aspect of the IDE is its robust file management system, which empowers users to efficiently organize, access, and manipulate project files. This includes mechanisms for opening, saving, and handling files seamlessly within the IDE environment. Once files are opened, the system parses them to understand their structure, content, and language type, preparing them for further processing and analysis. To enhance the coding experience, the architecture integrates a regular expression (regex) Code Mirror extension, enabling advanced text searching and manipulation within the code. Moreover, the system intelligently categorizes routes and workflows based on the programming languages used. This division allows for optimized processing and validation workflows, distinguishing between compiled languages and mark up

languages. Such categorization ensures efficient handling and validation of code, contributing to a smoother development experience for users.

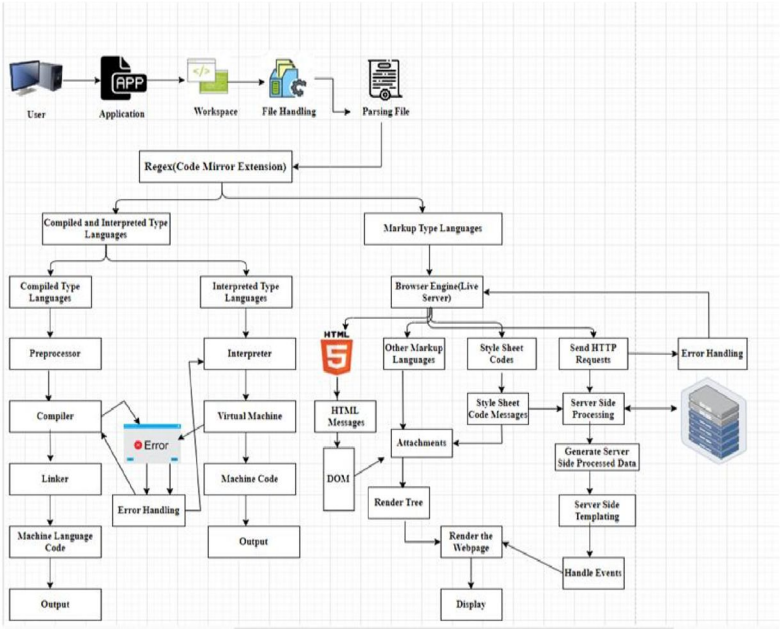

Fig.1 System Architecture

For compiled languages, such as C++ or C#, the architecture progresses through several stages. It begins with the processor, which passes the source code to the compiler for translation. The compiler then converts the source code into machine-readable instructions, preparing it for execution. Following this, the linker comes into play, responsible for incorporating any external libraries and generating a complete machine language code. This generated code is then ready for execution on the target system, resulting in various forms of output, such as the desired application or results. In case of compilation errors, the flow is redirected to the error handling mechanism, allowing developers to rectify issues before restarting the compilation process.

Conversely, for interpreted languages like Python or Java, the workflow diverges. It begins with the interpreter, which directly reads and executes the source code line by line. In the case of Java, the Java Virtual Machine (JVM) manages the execution of bytecode. Some interpreted languages may not have an explicit machine code generation step, as the interpreter or virtual machine directly executes the code. The output of code execution results in desired behaviour or output. Similar to compiled languages, error handling processes are in place to address any issues that may arise during interpretation or execution.

For markup type languages such as HTML, the workflow follows a different path. It starts with the browser engine, responsible for rendering and displaying the HTML content received from the server. In case of errors during loading or rendering, the browser may resend the request to the server for more reliable retrieval. On the server side, data processing often involves interactions with a database to fetch or manipulate information. Server-side templating allows for dynamic insertion of processed data into HTML templates to generate the final HTML page. JavaScript handles events like user interactions on the webpage. Finally, the browser engine renders the webpage, combining HTML content with CSS styling and JavaScript interactions, ultimately displaying the final rendered webpage to the user.

IV. Flow Chart Diagram

The flowchart illustrates a simplified workflow of IDE (Integrated Development Environment). For a desktop application, here's a breakdown of this general workflow:

1. The user interacts with the application, providing input like clicks or text. This input is then parsed by the application to understand what the user wants to do. There might be a step to check for syntax errors, especially if the user is entering code. If errors are found, the user would need to fix them before proceeding.
2. For applications that involve code execution (like a code editor), the parsed code may be compiled into a machine-readable format. This compiled code can then be run by the computer's processor. There might be another

check for errors during compilation. If errors are found, the user would need to debug the code to fix them and then recompile it.

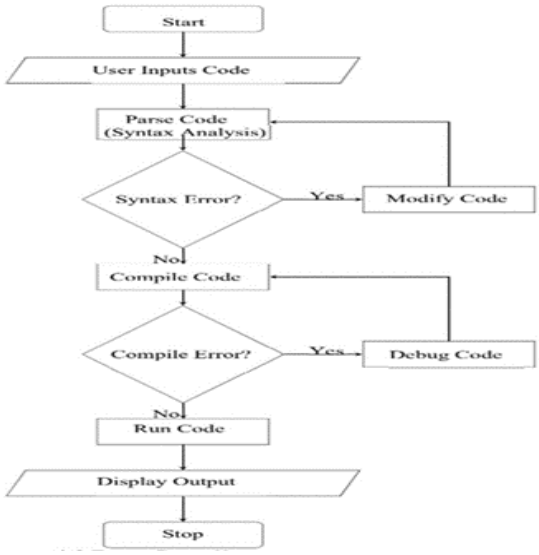

Fig.2 Flow chart Diagram

3. Once the code is ready to run (or if compilation isn't needed), the application executes the code and displays the results back to the user. This could be anything from text output to visualizations or interactive elements. Throughout this process, the application can also handle user interactions like mouse clicks or key presses.

V. METHODOLOGY

The main intention of this research is to make a light weight IDE which process the programs and software in light weight-based IDE environment. We propose some methodology for developing an IDE which are follows.

Integration of Code Mirror Library

Begin by integrating the Code Mirror library into the code editor project. Ensure that the library is included properly and configured to fulfil the entire execution process, including syntax highlighting, code editing features, and any other necessary functionalities. Customize the Code Mirror library as needed to fit the requirements of the code editor project.

Implementation of Project Explorer using Binary Tree

Design and implement a binary tree data structure to represent the project explorer and its subfolders. Develop algorithms to traverse the binary tree and display the project structure in the code editor interface. Implement functionalities to add, remove, and manage project files and folders within the binary tree data structure. Implement functionalities to add, remove, and manage project files and folders within the binary tree data structure.

Socket Programming for Connection Establishment

Utilize socket programming to establish communication between the local host (where the code editor is running) and end-user devices over the IP protocol and network interface. Implement socket server and client components to facilitate communication. Define protocols and message formats for exchanging data and commands between the code editor and connected clients.

Database Management with Relational DBMS for CRUD Operations

Choose a relational database management system (RDBMS) suitable for the project requirements (e.g., MySQL). Design the database schema to store information related to user accounts, projects, files, and any other relevant data. Implement CRUD (Create, Read, Update, and Delete) operations using SQL queries or ORM (Object-Relational Mapping) frameworks to interact with the database. Integrate database operations into the code editor to enable functionalities such as saving and loading projects, managing user preferences, and storing session data.

VI. Technology Used

Electron.js Framework

Electron.js serves as the foundation for cross-platform compatibility, allowing the IDE to run on Windows, macOS, and Linux.

Node.js and npm (Node Package Manager)

Node.js and npm (Node Package Manager) play a significant role in the development ecosystem, with Node.js serving as a runtime environment. IT is used for server-side functionalities, and npm (Node Package Manager) is employed for managing project dependencies.

React.js:

The user interface is built using React.js, a JavaScript library for building responsive user interfaces.

Tailwind CSS

Tailwind CSS is used for styling the user interface, allowing for efficient and customizable design.

Localtunnel

Localtunnel enables you to make your localhost accessible to the world, simplifying testing and sharing. There's no need to deal with DNS or deploy your changes just for others to test them. It's particularly useful for working with browser testing tools such as Browserling or external API callback services like Twilio that necessitate a public URL for callbacks.

Code-Mirror Library

The project relies on the Code-Mirror library for core code editing features, and updates or Changes to this library may impact the project's functionality.

ADVANTAGES

- *Efficient Coding*:
  Integrated Development Environments (IDEs) encompass the methodology of composing and overseeing code with the objective of optimizing productivity, reducing errors, and enhancing the efficiency of the entire development process.
- *Enhanced Productivity:*
  Integrated Development Environment (IDE) refers to the ability of the IDE to streamline and optimize the software development process, allowing developers to work more efficiently and effectively.
- *Integrated Tools:*
  Integrated Development Environment (IDE) refers to the combination of various software tools and features within a single environment to streamline and enhance the software development process.
- *Cross-Platform Development:*
  The IDE that can run on multiple platforms or operating systems, such as Windows, macOS, Linux.
- *Extensibility:*
  IDE to be easily extended or customized to accommodate new functionalities, features, or programming languages. It implies the ease with which developers can enhance or modify the IDE according to their specific needs or the evolving requirements of their projects**.**
- *Customization:*
  An IDE is a software application that provides comprehensive tools and features to facilitate the development of software. Customization empowers developers to adapt the IDE according to their coding style, project needs, and overall comfort, enhancing productivity and efficiency.

VII. Proposed System Applications

Web Development

IDEs are widely used for web development, providing features for HTML, CSS, and JavaScript coding, as well as tools for web server integration and debugging.

Game Development

This integrated development environment (IDE) provides support for 2D game development, for example, using Pygame.

Data Science and Analysis

IDEs with data science extensions, such as Jupyter Notebook, are essential for data analysis, machine learning, and scientific computing.

Teaching and Learning

Small-scale IDEs are often used in educational settings to introduce programming concepts to beginners. They provide a less overwhelming environment for students who are just starting to learn how to code.

Code Editing and Review

Developers may use small IDEs for code editing and review tasks. They can open code files, make quick edits, and review changes without launching a larger IDE.

Lightweight Development Environments

In resource constrained environments, such as on older computers or in cloud-based coding platforms, small IDEs are preferred due to their minimal system requirements.

VIII. Conclusion And Future Scope

In conclusion, our small-scale Integrated Development Environment (IDE) project has provided a lightweight and efficient coding environment for a range of development tasks. While it may have limitations in handling large-scale projects or extensive tool integration, it excels in teaching and learning, scripting, and rapid prototyping. The IDE's simplicity and low resource requirements make it accessible to beginners and suitable for tasks like automation, code editing, and text processing.

- Firstly, we aim to enhance its scalability and performance to support larger projects and improve compatibility with a wider range of development tools and kits.
- Additionally, we plan to introduce features for collaboration and code review to facilitate teamwork. Integration with version control systems will also be a priority.
- Furthermore, we intend to expand the IDE's capabilities for specific applications, such as web development, data analysis, and embedded systems development. We will keep refining the user interface, adding features for code navigation, and offering better support for different programming languages.

References

[1] Enikeev AI, Burnashev RA, Vakhitov GZ. Software tools and techniques for the expert systems building, Fourth International Congress on Information and Communication Technology: Singapore: Springer. 2020, pp,191-199. http://dx.doi.org/10.1007/978-981150637-6_16.

[2] Burnashev RA, Gubaydullin AV, AI. Enikeev's work titled “Construction of specialized Computer-Aided Software Engineering (CASE) tools for the development of expert systems” was published in 2020, http://dx.doi.org/10.1007/978-3-319-77703-0_59.

[3] Gabdrahmanov RT, Hussein AH, Burnashev RA, Enikeev AI. Formulation of the task of constructing an expert system for the diagnosis of leukemia. 2019 Third World Conference on Smart Trends in Systems Security and Sustainability (WorldS4): IEEE, 2019, pp. 1-5, http://dx.doi.org/10.1109/WorldS4.2019.8903934.

[4] Burnashev RA, Amer I, Enikeev AI. Expert system building tools based on dynamically updated knowledge. J Phys.: Conf Ser 1352(1): IOP Publishing. 2019:1. C. 012008, http://dx.doi.org/10.1088/1742-6596/1352/1/012008

[5] Muller HA, Norman RJ, Slonim J. 1996. Computer aided software engineering: New York: Springer,https://www.springerprofessional.de/specialized-case-tools-for-the development-of-expert-systems/15563852

[6] Hayes-Roth F, Waterman DA, Lenat DB. 1983. Building expert system, https://doi.org/10.1017/S026357470000202

[7] P. S. Newman IBM Scientific Center, Los Angeles, CA, Towards an integrated developmentenvironment,https://ieeexplore.ieee.org/abstract/document/5387858/authors #authors

[8] K. Leino, Valentin Wüstholz The Dafny Integrated Development Environment,2014, https://arxiv.org/pdf/1404.6602.pdf

[9] R.A. Burnashev; I.A. Enikeev; A.I. Enikeev Design and Implementation of Integrated Development Environment for Building Rule-Based Expert Systems,2020, https://ieeexplore.ieee.org/document/9271143/authors#authors

[10] Software Developers as Users: Developer Experience of a Cross-Platform Integrated Development Environment, Published:02 December 2015, https://doi.org/10.1007/978-3319-26844-6_40