PyEdit Pro: The Ultimate Advanced Text Editor in Python - Empowering Text Editing Experience

Rahul Roshan G
Dept. of Computer Science and Engineering
PES University
Bangalore, India
rahulroshanganesh2002@gmail.com

Rohan C
Dept. of Computer Science and Engineering
PES University
Bangalore, India
chandrashekar.rohans@gmail.com

Rohit Roshan
Dept. of Computer Science and Engineering
PES University
Bangalore, India
rohitroshan2002@gmail.com

S M Sutharsan Raj
Dept. of Computer Science and Engineering
PES University
Bangalore, India
sutharsanraj2001@gmail.com

Dr. Sapna Vikram Mewundi
Dept. of Computer Science and Engineering
PES University
Bangalore, India
sapnavm@pes.edu

***Abstract*— This research article aims to delve into the creation of an innovative text editor using Python. Text editing tools have gained significance in various domains, making it crucial to explore the design principles, features, and implementation techniques behind a robust editor. The paper will cover essential aspects such as user interface design, text manipulation, syntax highlighting, plugin support, and other advanced features. It also offers insights for developers interested in Python coding. Furthermore, it will address the development challenges and suggest potential avenues for future research in this field.**

Keywords*— *text-editor, formatting, style, code-editor, tts, python, GUI, extensibility, visual-impaired, compatibility

I. INTRODUCTION

Text editors are crucial in software development and various fields like web development, automation, data analysis, and more. They ensure text is readable and provide essential editing functions such as insertion, deletion, copy-paste. These tools enhance coding by offering features like syntax highlighting, auto-completion, code navigation, and debugging. Popular ones include Visual Studio Code, Sublime Text, Atom, and Vim, widely used in web development for HTML, CSS, and JavaScript. They simplify scripting and automation tasks in Python, Ruby, PowerShell, Bash, and others. Advanced text editors aim to prevent syntax errors. Collaborative features in some editors allow multiple users to work on the same document simultaneously, aiding remote teams and promoting real-time collaboration, boosting productivity and knowledge sharing.

II. METHODOLOGY

A. Architectural Design

Involved selecting appropriate user interface design patterns, determining the text manipulation capabilities, and deciding on the implementation of syntax highlighting, plugin support, code editors and other advanced functionalities. Adopting an iterative software development process, which included frequent testing, feedback, and refinement of the text editor. This allowed for the identification and resolution of any issues or bugs that arose during the development process. Develop the software using agile manifesto and the related principles

Programming Language and Framework Selection: The text editor was developed using the Python programming language due to its flexibility, ease of use, and availability of relevant libraries and modules. Both built-in Python functionality and external modules were employed to implement desired features. The tkinter module was used to showcase the rich GUI framework of the text editor.

Platform Independence: Ensuring platform independence by developing the text editor to be compatible with multiple operating systems, including Windows, Linux, and Mac. This involved considering the specific dependencies and requirements of each platform during the development process.

979-8-3503-0470-1/23/$31.00 ©2023 IEEE

Application and Testing: Deploying the completed text editor as an executable file making it suitable for local storage. Conducting thorough testing such as unit testing, integration testing, etc… to evaluate the functionality, performance, and usability of the text editor, and making necessary adjustments and improvements based on the test results. Figure 1 gives an ER Diagram for the basic text editor where the user can operate all these available operations like creating opening and editing the file.

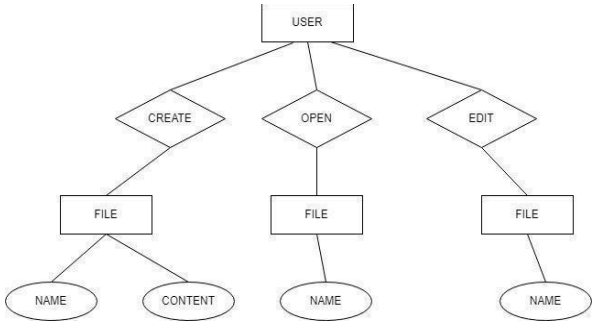

Fig. 1. ER Diagram for the basic text editor where the user can operate all editing the file.

B. Overall Description of the text editor software:

1) Overview:

The presented text editor software will help people to type in a GUI view and gives easy access to use various functionalities. The software aims to be primarily divided as shown in the Figure 2. This text-editor is very simple for users who are not even familiar with the usage of computer systems. Since this is menu driven, users will have a lot of options confronting them. The functionalities by this software are very professional such as highlighting text, spell-check, and many more in addition to naïve texting. It provides instructions similar to those of a traditional text editor, but with a few modulations of code changes, it can also edit non-textual data using the same user interface.

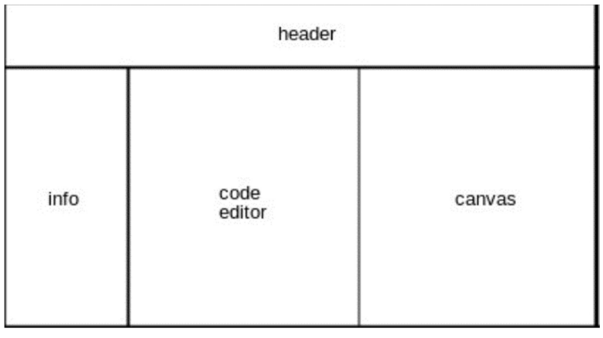

Fig. 2. Basic aim of the editor

2) Product Feature:

This product itself speaks for its many functionalities such as File operations to save, open, etc., Cursor motion and highlighting, Cut and paste, Search and replace, Customization and coding framework. Th same is indicate in Figure 3.

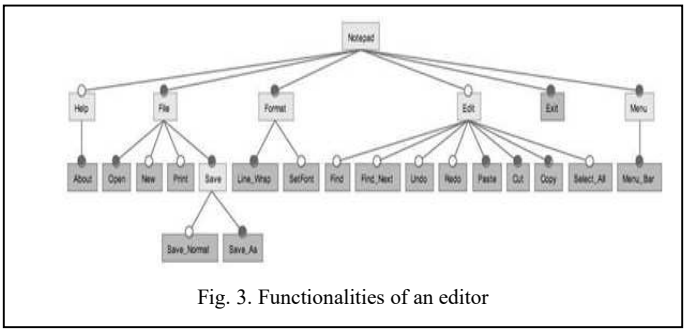

Fig. 3. Functionalities of an editor

3) User Case, Classes and Characteristics:

To present what these use cases are for, their overall roles in the project, the actors related to them and the relations between them are briefly explained in the table 1 below.

The system will support these user privileges: -

- Developer
- Naïve user
- Illustrator
- Future software developer to add more functionalities to this editor.
- Product owner to decide the future capabilities and investments to this product.
- Testers, for rigidity.

TABLE I: USE CASES AND THEIR DESCRIPTIONS

Use Case Name	***Description***
Develop	Develop a custom DGP algorithm in the environment. IDE offers a robust text editor for this purpose.
Step-by-step Debug	Detect the logical errors the custom algorithm may have by classic step by-step debugging technique
Visual Debug	Detect the logical errors the custom algorithm may have by visual debugging technique
Compare algorithm	Compare the behavior of the custom algorithm with a known, already. implemented one
Use DGP API	Develop a custom DGP algorithm in the environment by an offered, easy-to-use API
Customize IDE	Change predefined settings for the IDE (through preferences) for the suitable working environment

4) Operating Environment:

The software is designed to work on any version of Windows, Linux (kernel 2.7 and above) and Mac platform. This software enables different views for different varieties of users. Like, example, and user who simply types can use the regular views, for developers, a coding view, etc. Therefore, the design is implemented such that it is flexible.

Some of the constraints are: -

- Editor should be a use-once environment. That is, only user preferences are held in the file system therefore only the changes in preferences are remembered between two successive runs.
- Editor should not let access to calls for all legacy libraries through its lifetime.
- Editor should have one scene at a time.
- Editor should always be initialized to an empty scene.
- Editor should not depend on the operating system.

5) Assumptions and Dependencies:

The user is familiar with normal typing with the text editor and knows the basic functionalities and settings of any text editor. As mentioned in the operating environment it demands decent hardware specifications.

a) User Interfaces:

The text enable provides the following UI features:

Find and Replace, Cut, Copy and Paste, Ability to handle UTF-8 encoded text, Text Formatting, Undo and redo, line wrap, comment formatting, syntax highlighting and Source Code Editor.

- Hardware Interfaces: The editor is compatible with Personal Computers (PCs), Laptop Notebooks, and systems supporting Heroku.
- Software Interfaces: Built using Python, the editor uses .config files for customization. It utilizes Python inbuilt and external modules for development, with deployment on Heroku.
- Communications Interfaces: The app is deployable on local systems or cloud-based platforms via Heroku.

This text editor offers efficient text editing features, allowing users to create and save files with customized content. It also supports opening and editing existing files. The user-friendly interface enables easy text navigation and manipulation, including copy, cut, and paste functions. Text formatting options such as styles, fonts, sizes, and colors are available to enhance visual appearance, providing a comprehensive tool for productive and seamless text editing.

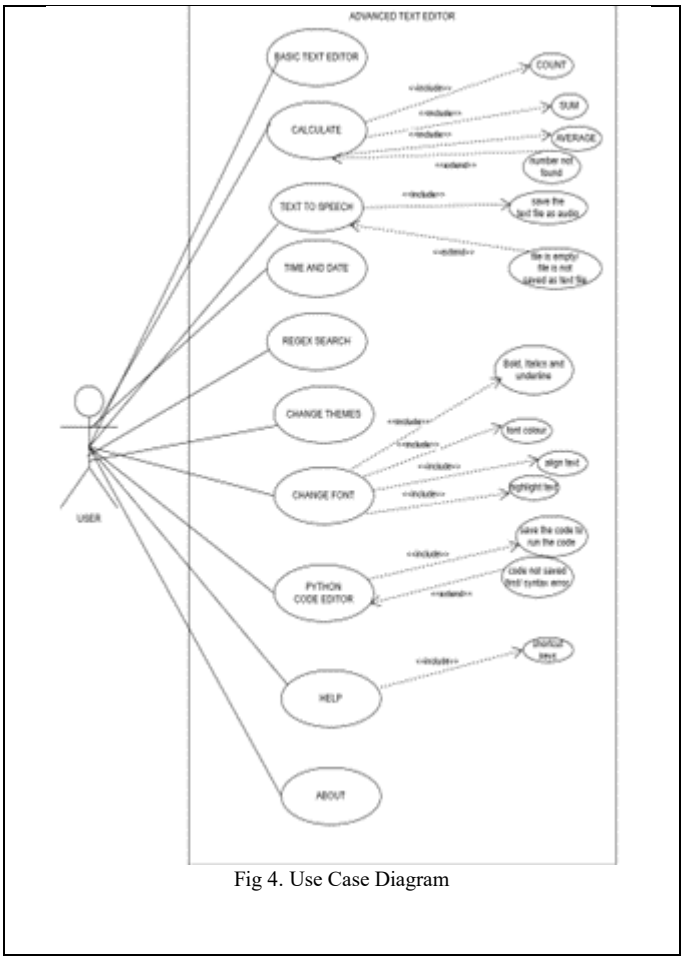

Fig 4. Use Case Diagram

6) Use Case Diagram:

Here user can use the text editor as a basic text editor like creating a new file, opening a new file, saving a file and exit the application. The basic feature also includes like find a word, find and replace, copy, paste and cut. This is advanced text editor where user can do way beyond, he thinks like regex search where user can enter a pattern to search. User can calculate the numbers (or digits) that are present in the text editor and can give average, sum and count (of number of times a digit present in the text editor). User will be able to print the present time and date. User can change the font size, color, and highlight text. This text editor can invoke the code editor where user can run the code. User can save the text in other format as pdf and audio file using google TTS. If user is naive, he can take help from help tab which can redirect to our GitHub page he can readme file or else he can view shortcut keys. Following Figure 4 shows a Use case diagram.

This use case diagram is used to how the system (advanced text editor) and user interacts. This can be used as the

manual to the user to see what the user can do with the system. This use case diagram describes the high-level functions and scope of advanced text editor. The system is defined as advanced text editor and actor mainly focused on visually impaired or normal user in this use case diagram.

III. WORKFLOW

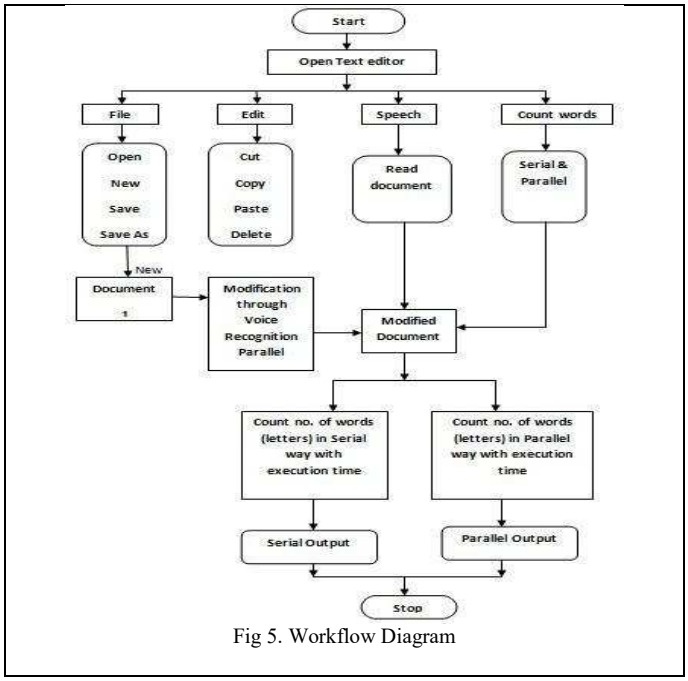

Fig 5. Workflow Diagram

The text editor can operate as basic text editor as well as advanced text editor. It can open a file, create a new file, save a file. It can also edit the text in text editor like cut, copy, paste the copied text and delete the text. This text editor can also can read the text that is written in the text editor this is used to visually impaired people. This can also be used to count the number of digits present between the text in text editor. Calculate the sum and average of the number. This are the vital features of this text editor. There are more advanced feature present in this text editor like regex search, font changing and highlighting and more.

IV. SYSTEM FEATURES AND INTERFACE

1) Find and Replace: Allows you to find all similar words and replace it with required word,: Add author names horizontally, moving to a third row if needed for more than 8 authors. (Refer Fig. 6)

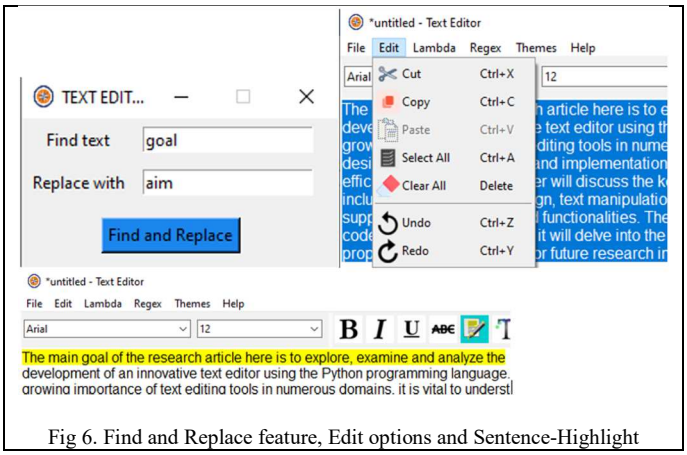

Fig 6. Find and Replace feature, Edit options and Sentence-Highlight

2) Cut, Copy and Paste: Allows the user to cut, copy and paste a word , text in the editor. To change the default, adjust the template as follows. (Refer Fig. 6)

3) Ability to handle UTF-8 encoded text: Large text can be stored in UTF-8 encoding and can be brought back to string when needed

4) Text Formatting: Allows you to change style, size and color of text.

5) Undo and redo: Allows you to undo and redo on the editor

6) Sentence highlight: This user allows user to highlight text. (Refer Fig. 6)

7) Comment formatting: Enable comments in the editor

8) Source Code Editor:

Fig 7. Code Editor

9) Saving text: Generating a paper soft copy of a text editor-created document

10) Regex search: A type of sophisticated search that prioritizes patterns over individual terms and phrases.

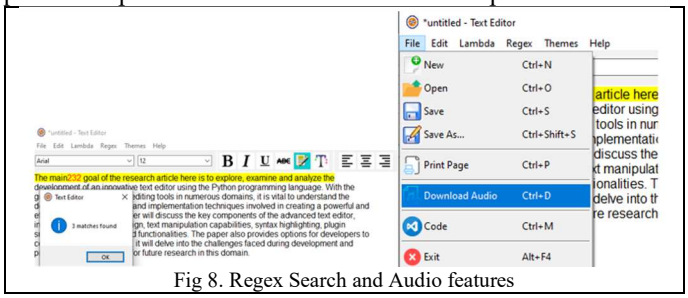

Fig 8. Regex Search and Audio features

11) Intro audio and saving the audio files: As you can see in Fig 8., this feature will be helpful for visual impaired people. It speaks out when the text editor launches. It has the audio feature which can save the text into speech audio in mp3 format which can be played at any time.

V. STIMULUS/RESPONSE SEQUENCES

A user can be able to click on the various buttons, highlight a text, open a menu, scroll the screen, hover on some menus, etc.

VI. PROJECT SCOPE

The software developed helps people mainly in: -

- Writing texts and including advanced features. – (Future ideas: flow-charts and clip arts).
- Writing codes and highlighting syntax. Providing different views and settings depending on the user.
- Option to save the files in the format they like.

This tool helps users in various ways, like preparing a list, or helps for documentaries. Apart from that it also helps users to perform many utilities like searching, find and replace, text-highlighting, etc.

VII. RESULTS

Fig 9. Text editor features and Themes

- The text editor shown is now able to use all the "high" features with good GUI tkinter python. Now this text editor can be used by everyone. As you can see Fig 9., we can change themes of text editor

- As seen in Figure 7, Code editor with saving option and opening a new file which is opened from a text editor. This tab shows that this editor enables the creation of fully-fledged programs and that it is capable of carrying out all the standard functions of a text editor as well as sophisticated compiler operations.

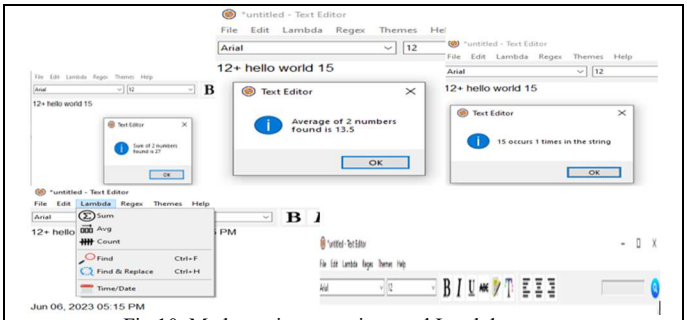

Fig 10. Mathematics operations and Lambda menu.

- Text editor with sum, average and count of number in text editor and time-date respectively.

As you can see in Fig. 10, we can change font style and search bar.

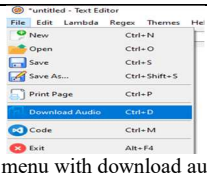

Fig 11. File menu with download audio features.

- Download audio and basic file features.

VIII. PERFORMANCE

The following Table shows the sample execution time in both serial and parallel way along with its total number of lines of code, and other features.:

TABLE II: SERIAL EXECUTION TIME TO OPEN THE APPLICATION

	2 cores	4 cores	4 cores	8cores
Minimum	2.0996 s	0.4028 s	0.1121s	0.1853s
Average	2.5627 s	0.4150 s	0.1188s	0.4351s

TABLE III: PARALLEL EXECUTION TIME TO OPEN THE APPLICATION

	2 cores	4 cores	4cores	8cores
Minimum	7.42 s	7.90s	6.11s	6.26s
Average	7.48 s	7.93s	6.16s	6.34s

TABLE IV: SAVING/DOWNLOADING AUDIO FILE.

97 words	9.0719 seconds
194 words	14.8728 seconds
388 words	29.229 seconds

TABLE V: TESTING BUILT-IN CODE EDITOR COMPILING TIME

one line "print" code	0.195 seconds
one for loop "print" code	0.182 seconds
two for loop "print" code	0.179 seconds
three for loop "print" code	0.165 seconds

IX. NOVELTY

Here, we are comparing our project with other research paper projects, similar to what they have accomplished.

- Accessibility: Tailored for visually impaired users with audio notifications and text-to-audio conversion.
- Versatile: Offers extensive features, from text editing to complex operations.
- Customizable: Provides themes and customization options for a personalized experience.
- Integration: Integrates with external tools and supports Python for versatility.
- Open-Source: Open for community contributions and collaborations.
- Text Processing: Offers advanced text processing

capabilities. Here are few features incorporated.

SN	Research paper	Their Methodology	Our Methodology
1.	Terminal Based Text Editor by Sumitkumar Shrivastav and Harsh Attri	Built a terminal based text editor using data structures and ncurses. (With search and replace features)	Our tkinter-based text editor prioritizes accessibility, offering audio notifications and text-to-audio conversion for the visually impaired. It provides robust text editing features, including search, replace, copy, paste, and undo/redo, alongside functionalities like printing, audio saving, and basic calculations. Offers various themes too for UI enhancement.
2.	Blocks and Text Integration in a Language-Based Editor for a Domain-Specific Language.	Integration of block-based syntax (feature like save, load, search, toggle, themes, auto conversion) and test-based syntax (feature like syntax highlighting, indentation, code folding, auto-completion, find/replace and bracket matching.) in language-based editor	Same as above. But this research paper has high feature. We have mostly had all the features except code folding, auto-completion and bracket matching (this is our future ideas to improve)
3.	Multimodal Input Interface Using Multicore	Get input through multi-ways (typing, speech recognition) in a Text pad, and performing multiple actions (like cut, copy, delete, paste by speech recognition.)	Same as above. But in our project, we take input as text and text editor guides visually impaired people by audio.
4.	NEWRITER: A Text Editor for Boosting Scientific Paper Writing.	A neural network-based approach to address scientific text writing assistance.	Same as above.

X. CONCLUSION

In summary, text editors are versatile tools that significantly impact software development, web development, scripting, data analysis, technical writing, system administration, and collaborative work. Their features and flexibility make them essential for professionals across various fields and it will help the visually impaired people. The usage of a text editor is facilitated for those with visual impairments. When compared to other text editors within the Python ecosystem, PyEdit Pro shines brightly due to its array of advanced features and user-friendly interface. Its rich toolset and the ability to extend its functionality position it as a formidable contender against well-established text editors such as Sublime Text and Atom. The efficient memory management and resource optimization further contribute to its appeal among developers.

XI. REFERENCES

[1] R, Balaji & Babu, M.. (2011). Multimodal Input Interface Using Multicore. International Journal of Computer Information System. 2. 67-71.

[2] E. Altuncu, B. K. Bilgehan, Y. S. Kartal, S. Kızılgüneş, M. S. Nikoo and H. Oğuztüzün, "Blocks and text integration in a language-based editor for a domain-specific language," 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 2017, pp. 1084-1089, doi: 10.1109/UBMK.2017.8093484.

[3] Christopher W. Fraser. 1980. A generalized text editor. Commun. ACM 23, 3 (March 1980), 154–158. https://doi.org/10.1145/358826.358834

[4] S. Shrivastav and H. Attri, "Terminal Based Text Editor," 2022 3rd International Conference for Emerging Technology (INCET), Belgaum, India, 2022, pp. 1-4, doi: 10.1109/INCET54531.2022.9824597.

[5] Joao Ribeiro Bezerra, Lu´ıs Fabr´ıcio Wanderley Goes & Wladmir Cardoso Brandao (2022). NEWRITER: A Text Editor for Boosting Scientific Paper Writing.

[6] Jiseong Gu and Geehyuk Lee. 2019. “In-Place Ink Text Editor: A Notepad-like Simple Text Editor with Direct Writing.” In Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces (ISS '19). Association for Computing Machinery, New York, NY, USA, 327–329. https://doi.org/10.1145/3343055.3360744.