Report On

TEXT EDITOR

Submitted in partial fulfillment of the requirements of the Skill Based Language (SBL) Course project in

Semester IV of Second Year Computer Engineering

by

Yash Chavan (Roll No. 29)

Kamal Chhotaray (Roll No. 30)

Krisha Chikka (Roll No. 31)

Supervisor

Prof. Sneha Mhatre

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**

****

**(2023-24)**

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**

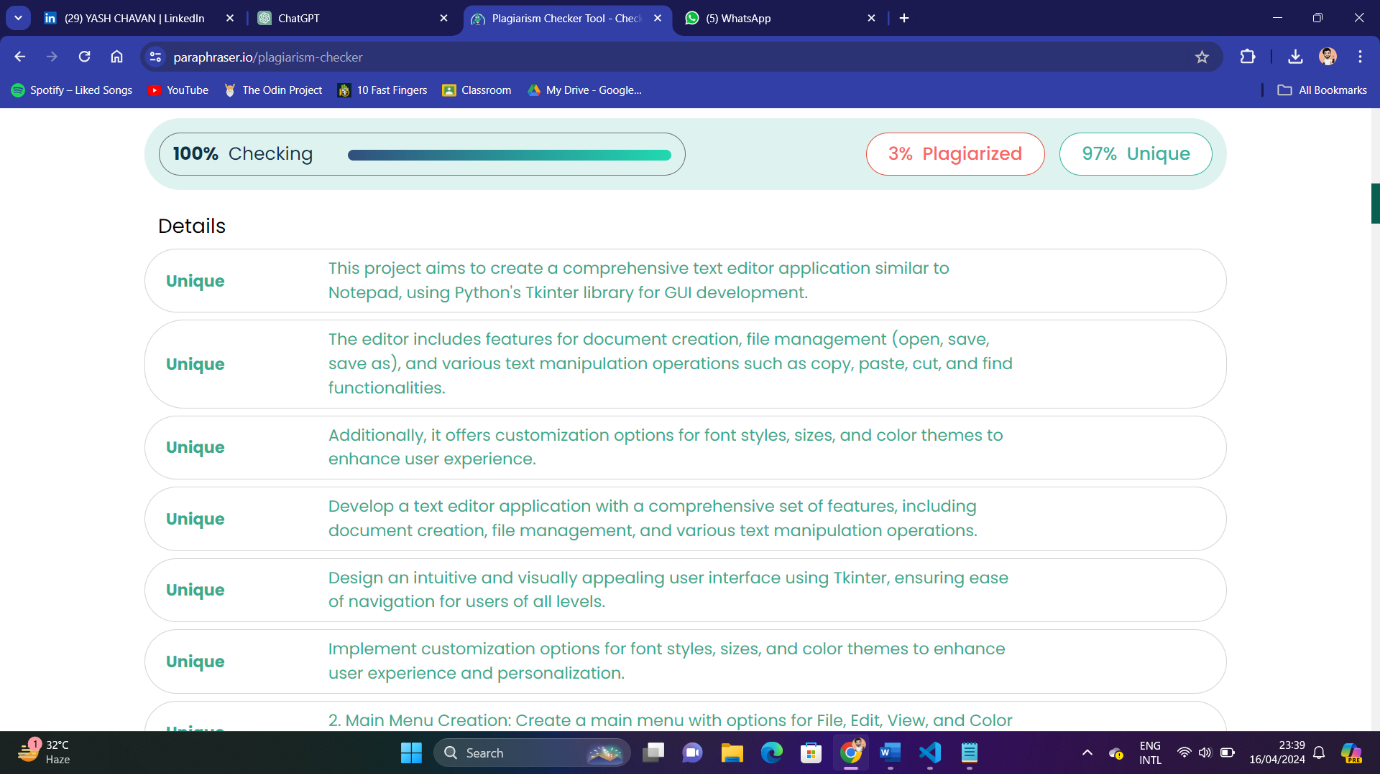
**CERTIFICATE**

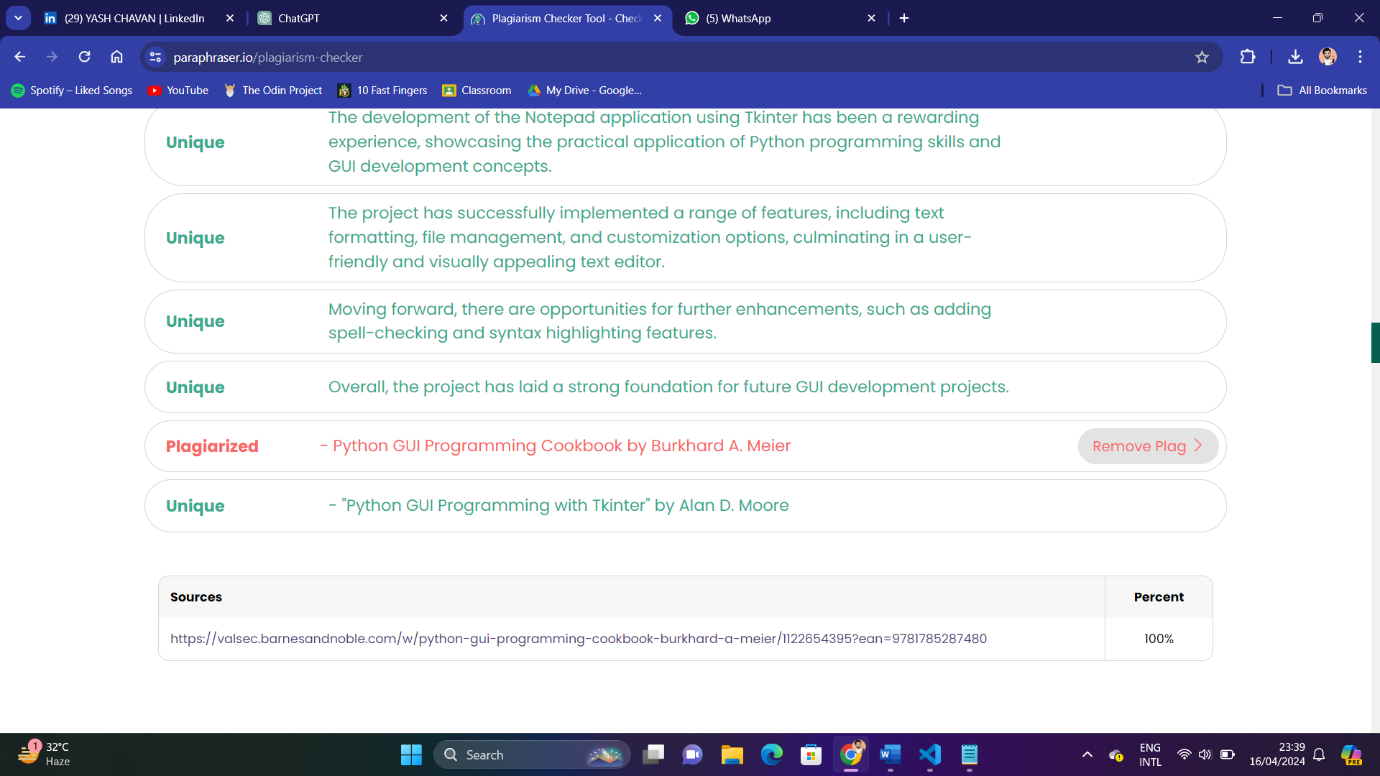
This is to certify that the course project entitled “TEXT EDITOR” is a bonafide work of "Yash Chavan (Roll No. 29), Kamal Chhotaray (Roll No. 30), Krisha Chikka (Roll No. 31)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

**Supervisor**

|  |  |  |
| --- | --- | --- |
| Prof. Sneha Mhatre |  |  |
|  |  |  |
| Dr Megha Trivedi  Head of Department |  | Dr. H.V. Vankudre  Principal |

**PLAGIARISM REPORT:**

****

****

**Abstract:**

This project “TEXT EDITOR” endeavors to create a comprehensive text editor application akin to the ubiquitous Notepad, leveraging Python's Tkinter library for graphical user interface (GUI) development. The text editor is designed to offer a plethora of functionalities essential for text manipulation, including document creation, opening existing files, and saving documents.

Furthermore, it encompasses a range of text editing tools such as copy, paste, cut, and find functionalities. To enhance user experience, the editor also facilitates customization options, allowing users to select from various color themes, modify font styles and sizes, and toggle the visibility of toolbars and status bars. By amalgamating GUI design principles with robust text manipulation features, this project serves as an instructive endeavor in constructing a versatile and user-friendly text editor application in Python.

**Contents**

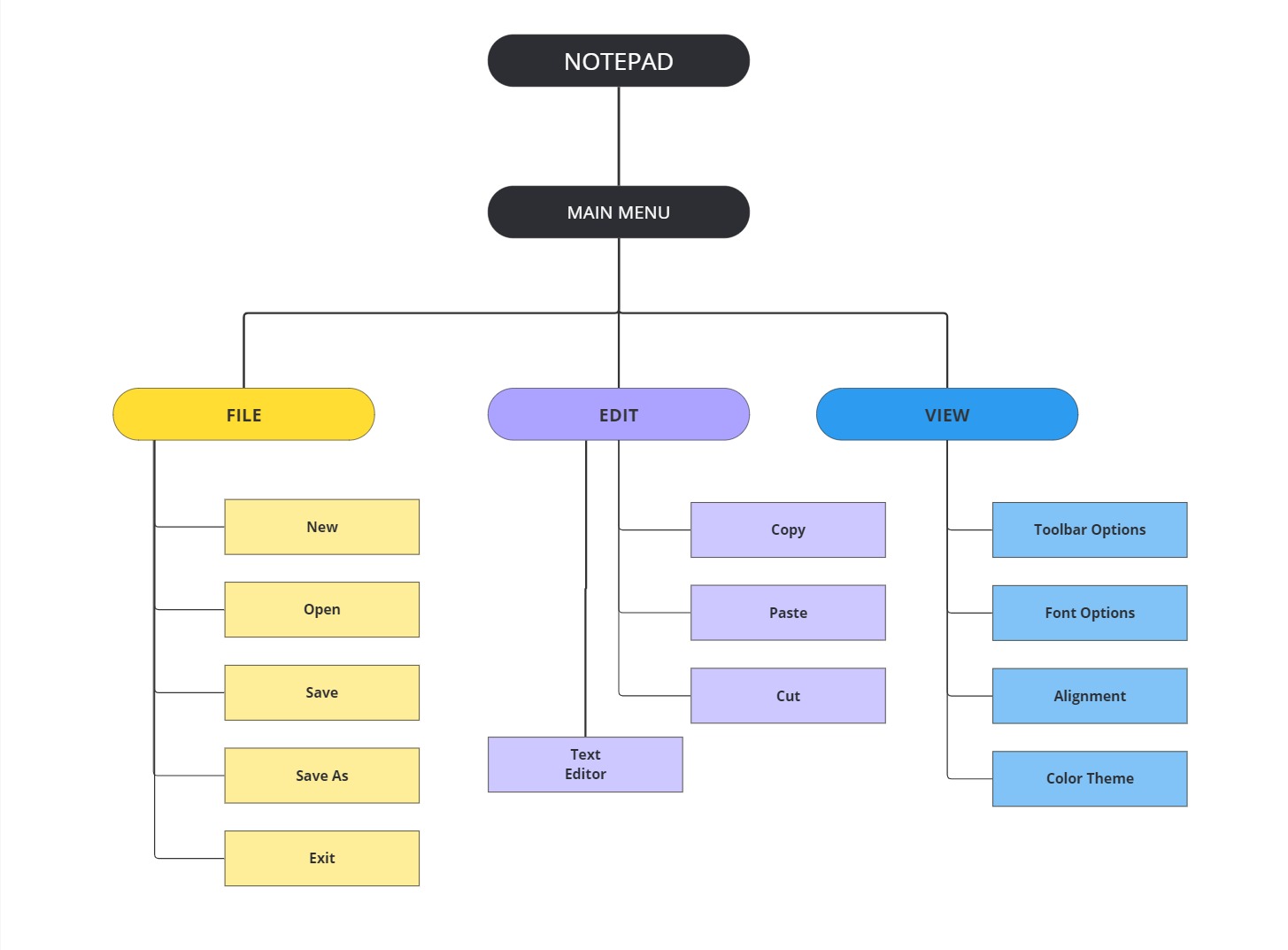
|  |  |
| --- | --- |
| 1) Problem Statement | 6 |
| 2.1) Block Diagram | 7 |
| 2.2) Description | 7-8 |
| 2.3) Working | 8-10 |
| 3) Module Description | 11-12 |
| 4) Hardware and Software Requirements | 13 |
| 5) Code  6.1) Results  6.2) Conclusion  7) References | 14-15  16  17  18 |

**Problem Statement:**

* To develop a text editor application with a comprehensive set of features, including document creation, file management (open, save, save as), and various text manipulation operations such as copy, paste, cut, and find functionalities.
* To design an intuitive and visually appealing user interface using Tkinter library, ensuring ease of navigation and accessibility for users of all levels. Implement customization options for font styles, sizes, and color themes to enhance user experience and personalization.

**Block Diagram, description and Working:**

* **Block Diagram:**

****

* **Description:**

The Notepad application is a lightweight text editor designed to provide users with a simple yet feature-rich platform for creating, editing, and managing text documents. Developed using the Python programming language and the Tkinter library for GUI development, this application offers a user-friendly interface and a range of functionalities tailored to meet the needs of both casual users and professionals.

Key Features:

1. *Intuitive User Interface:*

The application boasts an intuitive and visually appealing interface, allowing users to navigate seamlessly and focus on their writing tasks without distractions.

1. *Comprehensive Text Editing:*

Users can perform a wide range of text editing operations, including font selection, size adjustment, bold, italic, and underline formatting, making it easy to customize the appearance of their documents.

1. *Efficient File Management:*

With support for essential file management operations such as creating new documents, opening existing files, saving changes, and saving documents under different filenames, users can efficiently organize and manage their text files.

1. *Powerful Editing Tools:*

The application includes essential editing tools such as copy, paste, cut, clear all, and find/replace functionalities, enabling users to manipulate text with ease and precision.

1. *Customization Options:*

Users have the flexibility to customize their editing environment by selecting from various color themes, adjusting font styles and sizes, and aligning text according to their preferences.

1. *Real-time Status Updates:*

The status bar at the bottom of the interface provides real-time updates on word and character counts, giving users instant feedback on the length and structure of their documents.

1. *User-Friendly Shortcuts:*

Keyboard shortcuts for common operations enhance productivity and streamline the user experience, allowing users to perform tasks quickly and efficiently.

* **Working:**

*1. Initialization:*

- The application is initialized with the Tkinter library.

- The main window is created with specific dimensions and title.

*2. Main Menu Creation:*

- The main menu is created with options for File, Edit, View, and Color Theme.

*3. Toolbar Creation:*

- A toolbar is added to the top of the window with buttons for various text formatting and editing options.

*4. Text Editor Creation:*

- A text editor widget is added to the main window, allowing users to input and edit text.

- A scrollbar is attached to the text editor for vertical scrolling.

*5. Font Selection:*

- Dropdown menus for selecting font family and font size are added to the toolbar.

- Users can select their preferred font family and size from these dropdown menus.

*6. Text Formatting Buttons:*

- Buttons for bold, italic, underline, font color, and text alignment are added to the toolbar.

- Users can apply these formatting styles to the selected text in the editor.

*7. Status Bar Creation:*

- A status bar is added to the bottom of the window to display information about the text, such as word count and character count.

*8. File Operations:*

- Options for creating new files, opening existing files, saving files, and saving files with different names are added to the File menu.

- Keyboard shortcuts are assigned to these operations for easy access.

*9. Edit Operations:*

- Options for copying, pasting, cutting, and clearing text are added to the Edit menu.

- Keyboard shortcuts are assigned to these operations for convenience.

*10. Find and Replace Functionality:*

- The Find option is added to the Edit menu, allowing users to search for specific words or phrases in the text.

- Users can also replace occurrences of a word or phrase with another word or phrase.

*11. View Options:*

- Checkable options are added to the View menu for showing/hiding the toolbar and status bar.

*12. Color Theme Selection:*

- Options for selecting different color themes for the text editor background and text color are added to the Color Theme menu.

- Users can choose their preferred color theme from the available options.

*13. Keyboard Shortcuts:*

- Keyboard shortcuts are bound to various menu options and operations for quick access and usability.

*14. Executable Creation:*

- Using the cx\_Freeze library, the script is converted into an executable file that can be run on Windows platforms.

*15. Event Handling:*

- Functions are defined to handle events such as text modifications, menu selections, and button clicks.

- These functions implement the desired functionality based on the user's actions.

*16. Main Loop:*

- The main loop of the application is started, allowing it to listen for user interactions and update the interface accordingly.

- The application remains responsive and functional until the user closes it.

This workflow outlines the steps involved in creating and using the text editor application, from initialization to execution and interaction.

**Module Description:**

1. User Interface Module:

* Description: This module handles the graphical user interface (GUI) components of the text editor.
* Functionality: It creates and manages windows, menus, toolbars, buttons, and other UI elements using the Tkinter library.
* Responsibilities: Displaying the text editor window, handling user input events, and updating the interface based on user actions.

1. Text Editing Module:

* Description: This module manages the text editing functionality of the text editor.
* Functionality: It provides operations for inserting, deleting, selecting, and modifying text within the editor.
* Responsibilities: Handling text input, cursor movement, selection highlighting, and undo/redo functionality.

1. File Operations Module:

* Description: This module handles file-related operations such as opening, saving, and closing files.
* Functionality: It interfaces with the operating system to read from and write to files on disk.
* Responsibilities: Implementing file dialogs, reading/writing file content, managing file paths, and handling file-related errors.

1. Formatting Module:

* Description: This module provides functionality for formatting text, including font selection, size adjustment, and style changes.
* Functionality: It allows users to customize the appearance of text by changing font families, sizes, styles (e.g., bold, italic), and colors.
* Responsibilities: Applying user-selected formatting options to the text editor’s content and updating the display accordingly.

1. Search and Replace Module:

* Description: This module facilitates searching for specific words or phrases within the text and replacing them with alternative text.
* Functionality: It implements search algorithms to find occurrences of a given query and offers options for replacing or highlighting matches.
* Responsibilities: Processing search queries, highlighting search results, and performing text replacements as requested by the user.

1. Status Bar Module:

* Description: This module manages the status bar displayed at the bottom of the text editor window.
* Functionality: It provides information about the current state of the text editor, such as word count, character count, and editing mode.
* Responsibilities: Updating status bar contents in response to text editing events, file operations, or user interactions.

1. Color Theme Module:

* Description: This module allows users to customize the color theme of the text editor interface.
* Functionality: It provides predefined color schemes and options for selecting background and text colors.
* Responsibilities: Applying the chosen color theme to the various UI components of the text editor, including the text area, menus, and toolbars.

1. Executable Creation Module:

* Description: This module handles the conversion of the Python script into a standalone executable file.
* Functionality: It uses the cx\_Freeze library to freeze the Python script and package it with the necessary dependencies.
* Responsibilities: Generating an executable file that users can run on Windows platforms without requiring Python or additional installations.

**Brief description of software and hardware used and its programming:**

Software Required:

1. Operating System:

Compatible with Windows, macOS, and Linux.

1. Python Runtime:

Python 3.x interpreter installed on the system.

1. Tkinter Library:

Included with standard Python installations.

1. Additional Libraries:

* os:

The os module provides functions for interacting with the operating system, such as working with files, directories, and environment variables.

* sys:

The sys module provides access to system-specific parameters and functions, such as command-line arguments and exit codes.

* cx\_Freeze:

This library is used for creating standalone executable files from Python scripts, allowing users to run the text editor application without requiring Python or additional installations.

* tkinter.font
* tkinter.colorchooser
* tkinter.filedialog
* tkinter.messagebox
* tkinter.scrolledtext
* tkinter.ttk

Hardware Required:

1. Personal Computers: Standard personal computers or laptops were used by developers for coding and testing.

**Code :**

* **Notepad.py**

import tkinter as tk

from tkinter import ttk, font, colorchooser, filedialog, messagebox

import os

main\_application = tk.Tk()

main\_application.geometry('1200x800')

main\_application.title('Notepad')

def new\_file(event=None):

pass

def open\_file(event=None):

pass

def save\_file(event=None):

pass

def save\_as(event=None):

pass

def exit\_func(event=None):

pass

def find\_func(event=None):

pass

def change\_theme():

pass

main\_menu = tk.Menu()

file = tk.Menu(main\_menu, tearoff=False)

file.add\_command(label='New', command=new\_file)

file.add\_command(label='Open', command=open\_file)

file.add\_command(label='Save', command=save\_file)

file.add\_command(label='Save As', command=save\_as)

file.add\_command(label='Exit', command=exit\_func)

main\_menu.add\_cascade(label='File', menu=file)

edit = tk.Menu(main\_menu, tearoff=False)

edit.add\_command(label='Copy')

edit.add\_command(label='Paste')

edit.add\_command(label='Cut')

edit.add\_command(label='Clear All')

edit.add\_command(label='Find', command=find\_func)

main\_menu.add\_cascade(label='Edit', menu=edit)

view = tk.Menu(main\_menu, tearoff=False)

view.add\_checkbutton(label='Tool Bar')

view.add\_checkbutton(label='Status Bar')

main\_menu.add\_cascade(label='View', menu=view)

color\_theme = tk.Menu(main\_menu, tearoff=False)

color\_theme.add\_radiobutton(label='Light Default', command=change\_theme)

color\_theme.add\_radiobutton(label='Dark', command=change\_theme)

main\_menu.add\_cascade(label='Color Theme', menu=color\_theme)

main\_application.config(menu=main\_menu)

main\_application.mainloop()

* **setup.py**

import cx\_Freeze

import sys

import os

base = None

if sys.platform == 'win32':

base = "Win32GUI"

os.environ['TCL\_LIBRARY'] = r"C:\Users\MAUT\AppData\Local\Programs\Python\Python37\tcl\tcl8.6"

os.environ['TK\_LIBRARY'] = r"C:\Users\MAUT\AppData\Local\Programs\Python\Python37\tcl\tk8.6"

executables = [cx\_Freeze.Executable("Notepad.py", base=base, icon="mainicon.ico")]

cx\_Freeze.setup(

name = " Notepad ",

options = {"build\_exe": {"packages":["tkinter","os"], "include\_files":["mainicon.ico",'tcl86t.dll','tk86t.dll', 'icons2']}},

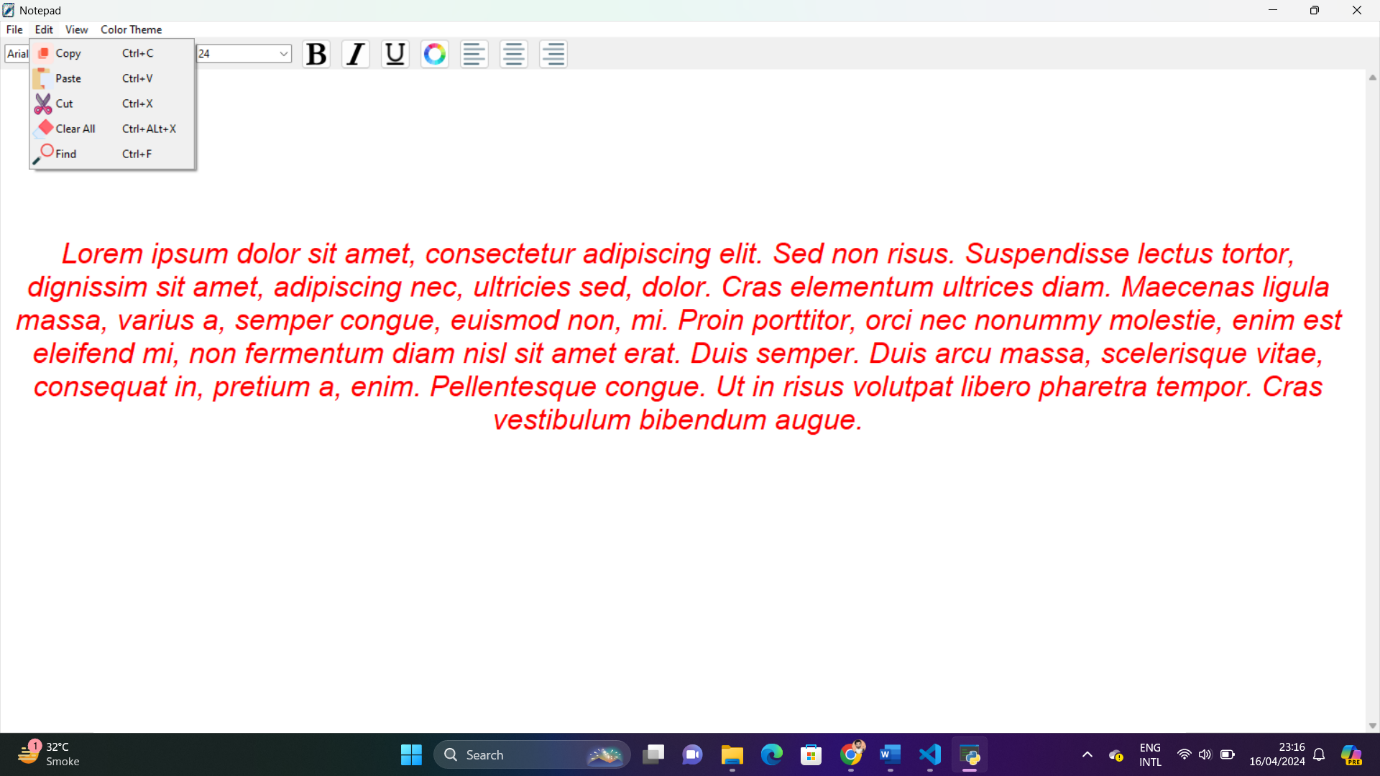
version = "0.01",

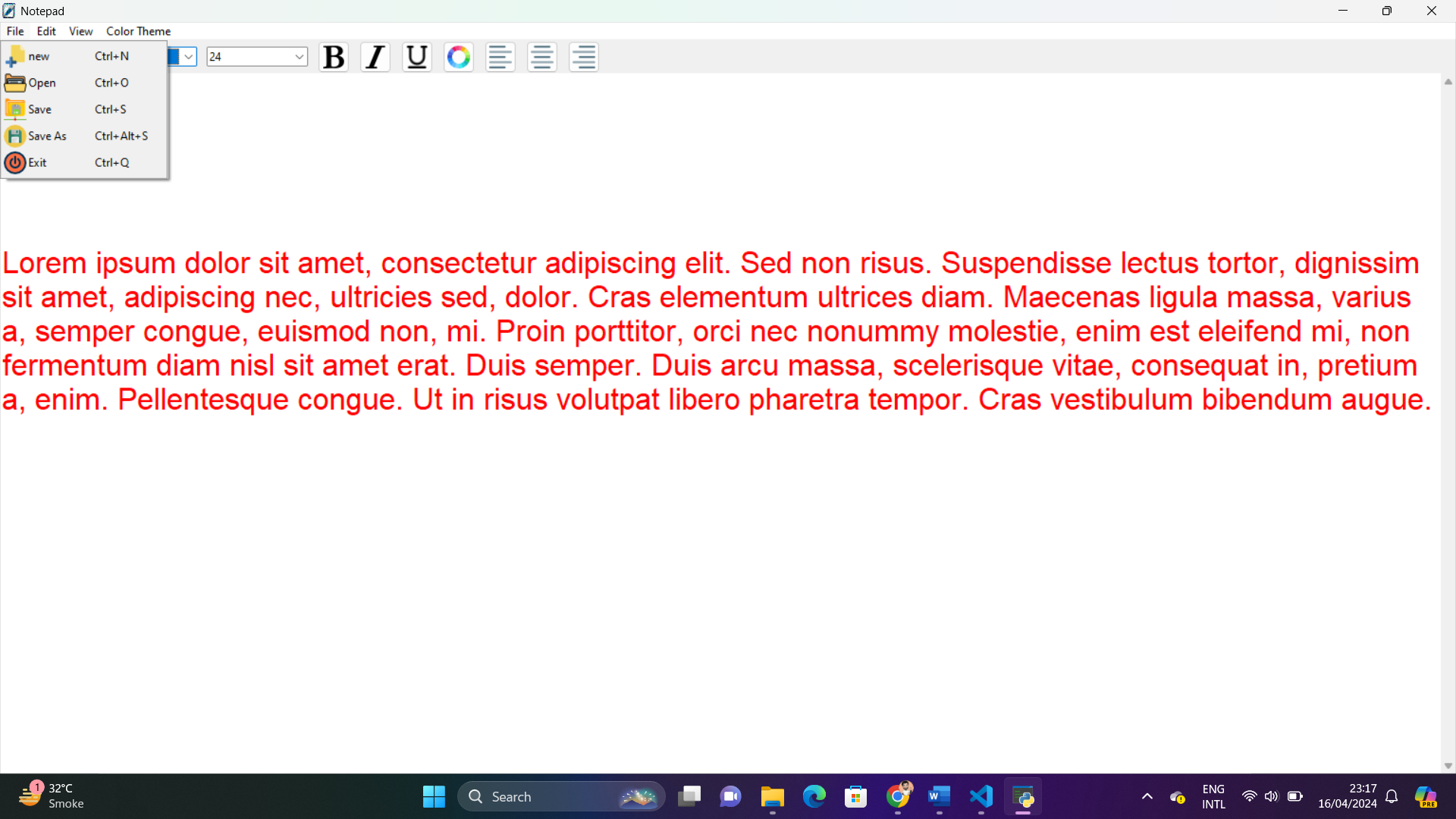
description = "Tkinter Application",

executables = executables

)

**Results:**

****



**Conclusion:**

In conclusion, the development of the Notepad application using Tkinter has been a rewarding journey that has allowed for the practical application of various concepts learned throughout the course. Through this project, we have achieved the following objectives:

* GUI Development: We successfully designed and implemented a user-friendly graphical interface using Tkinter, allowing users to interact with the application seamlessly.
* Functionality: The Notepad application encompasses a wide range of functionalities, including text formatting options (such as font selection, size adjustment, bold, italic, and underline), file management operations (such as new file creation, opening, saving, and saving as), and editing capabilities (such as copy, paste, cut, and find/replace).
* Customization: Users can personalize their experience by selecting from various color themes and toggling the visibility of the toolbar and status bar according to their preferences.
* User Experience: Emphasis has been placed on providing a smooth and intuitive user experience, with features like real-time word and character count in the status bar, keyboard shortcuts for common operations, and a clean and visually appealing interface.
* Learning Outcomes: This project has not only allowed for the practical application of Python programming skills but has also deepened the understanding of GUI development concepts and the integration of third-party libraries like cx\_Freeze for executable creation.

Moving forward, there are opportunities for further enhancements and refinements, such as incorporating additional features like spell-checking, syntax highlighting, and support for different file formats. Overall, the development of the Notepad application has been an enriching experience, showcasing the capabilities of Tkinter and laying the foundation for future projects in GUI development.

**References:**

1. Python Documentation: <https://docs.python.org/3/library/index.html>

- Official Python documentation provides comprehensive information on the Python programming language and its standard libraries, including Tkinter.

2. Tkinter Documentation: <https://docs.python.org/3/library/tkinter.html>

- The official Tkinter documentation offers detailed guidance on using Tkinter for GUI development in Python.

3. Stack Overflow: <https://stackoverflow.com/>

- Stack Overflow is a valuable resource for troubleshooting programming issues, finding solutions to common problems, and seeking advice from the developer community.

4. Real Python: <https://realpython.com/>

- Real Python provides tutorials, articles, and resources for Python developers at all skill levels, including topics related to GUI development with Tkinter.

5. Python GUI Programming Cookbook by Burkhard A. Meier - This book offers practical recipes and examples for building graphical user interfaces with Python and Tkinter, making it a useful reference for GUI development projects.

6. "Python GUI Programming with Tkinter" by Alan D. Moore - This book provides a comprehensive guide to creating GUI applications using Tkinter in Python, covering topics such as widget creation, event handling, and layout management.

7. TkDocs: <https://tkdocs.com/>

- TkDocs offers tutorials and documentation for Tkinter, covering fundamental concepts and advanced techniques for building GUI applications with Tkinter.