B.TECH. (CSE)

V SEMESTER

UE20303 –SOFTWARE ENGINEERING

PROJECT REPORT

ON

SIMPLE TEXT EDITOR

SUBMITTED BY

NAME	SRN
1) **Rashmi K R**	**PES2UG20CS266**
2) **Reshmi Pradeep**	**PES2UG20C270**
3) **Rimzim Sanghvi**	**PES2UG20CS273**

Github Repository: https://github.com/reshmipradeep/SE-Project

August – Nov 2022
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
BENGALURU – 560100, KARNATAKA, INDIA

TABLE OF CONTENTS

PROPOSAL OF THE PROJECT

Text editor is a simple application. It helps users to open any text file, write, edit and format texts and save it in a file. We are making this editor using python and it's libraries like Tkinter. Tkinter is a standard python interface to the Tk GUI toolkit.

A text editor makes it easier to type texts, specially ones in different program languages, so many special features are included like autocompletion for parenthesis, brackets (square and curly), single quotes, and double quotes; auto indentation for code blocks and brackets (when creating lists or creating functions); syntax highlighting for different languages (Python, C, C++, JavaScript, Java, HTML, Sql, CSS) and the ability to compile and run your code from the editor.

To make it attractive and easier to read, different themes and color schemes that the user can pick from will be added along with customization options like font, text color, etc.

SOFTWARE REQUIREMENTS SPECIFICATION

Version 1.1 approved

Prepared by:

Rashmi KR PES2UG20CS266

Reshmi Pradeep PES2UG20CS270

Rimzim Sanghvi PES2UG20CS273

PES UNIVERSITY, Bangalore

04/09/2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Reshmi Pradeep	3/11/2022	Added the Test case IDs to RTM	Version 1.1

Introduction

Purpose

The purpose of this document is to build a simple text editor. It helps users to open any text file, write, edit and format texts and save it in a file, which summarizes its basic scope.

Intended Audience

This project is a prototype for an editor and it is restricted within the college premises. This project is useful for the anyone who wishes to do any type of programming or coding for any

purpose with ease. Readers can include developers, users, and testers.

Product Scope

Text editor is a simple application used to write, edit and format texts and save it in a file. This editor is to be implemented using python and it's libraries like Tkinter.
A text editor makes it easier to type texts, especially ones in different program languages, so many special features and the ability to compile and run your code from the editor.
To make it attractive and easier to read, different themes and color schemes that can be customized are also included. Above all, we will try to provide the user with an easy, customizable and user-friendly editor to help code with ease

References

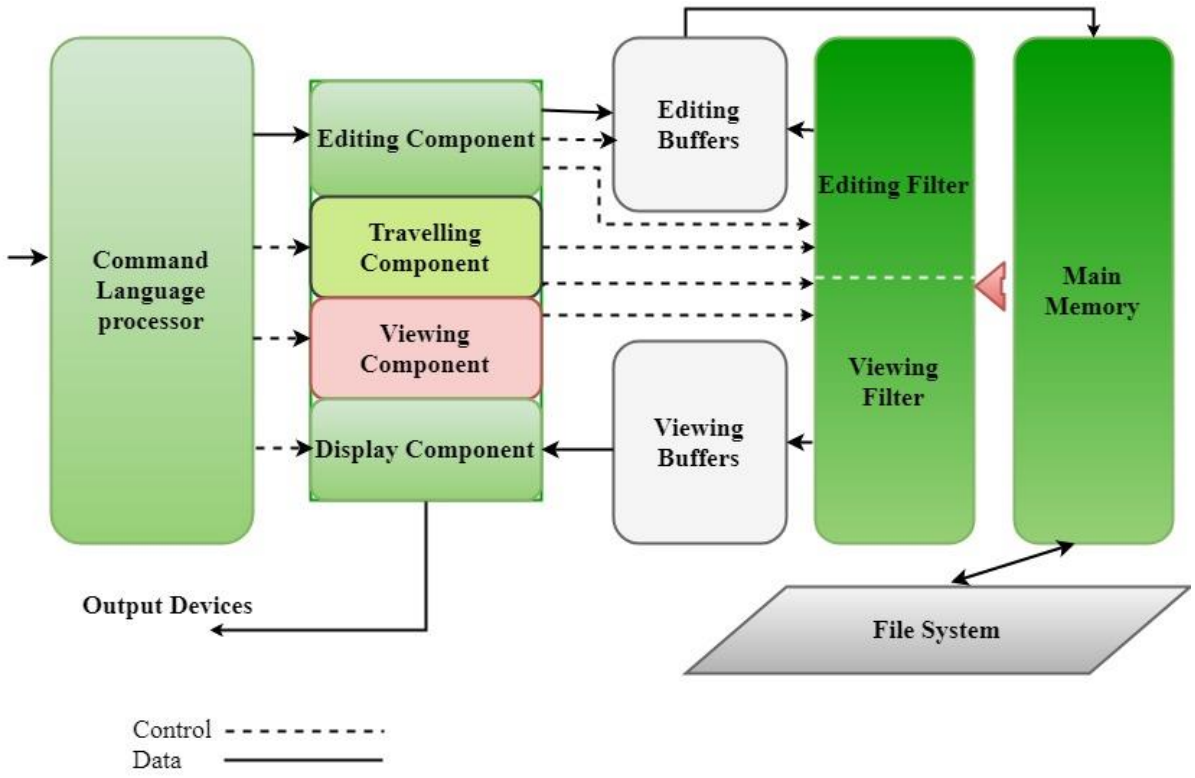
https://www.mattduck.com/build-your-own-text-editor.html

https://www.geeksforgeeks.org/editors-types-system-programming/

http://www.tezu.ernet.in/~utpal/course_mat/ss_editor.html

Overall Description

Product Perspective



Product Functions

The major functions of text editor are:

- Creation of a file
- Open file and edit its contents
- Save file
- Close file
- Cut, copy and paste the text
- Choose custom themes

The functions coupled with a few more tentative functions make up the text editor as shown in the following entity-relationship diagram.

User Classes and Characteristics

The main user for this product will the any person acting as the user and using the editor to open, modify and save files with data, in an easier and organized method.
The system engineer will have access to installing and dealing with issues related to the editor.

Operating Environment

Operating environment for simple text editor is as listed below. distributed database

- Operating system: Windows.
- Platform: python

Design and Implementation Constraints

The file system cannot be largely kept track of as no DBMS is being used. The editor is not launchable from the terminal and free to run scripts from any platform due the differences in the syntax and less generalisation.

Assumptions and Dependencies

Since a lot of pre-defined libraries are to be used in this product, including Tkinter from Python for GUI, it could have certain unexpected changes during the process of making the editor.

External Interface Requirements

User Interfaces

The screen layout would be similar to that of any editor, with text area in the middle and a small task bar and menu bar. It would have buttons for most functions like save, new file, quit, run, etc. All the typing is done in the text area. Tkinter is used for all the GUI which makes the entire text editor and it's appearance on the screen.

Software Interfaces

Software interfaces used for the text editor:

- Front-end software: Python 3.10 (Tkinter library for GUI)
- Back-end software: Python 3.10
- Operating system: Windows operating system for its best support and user-friendliness.
- More libraries to be used (TBD)

Communications Interfaces

No communication functions or interfaces used.

Analysis Models

USE CASE
From the problem description, we can see that the system has 2 actors.
Regular user
System Manager
To define system's functionalities, we can view the system as a collection of following cases:

- File handling
- Editing
- Formatting
- Programming

System Features

File Handling

Description and Priority

User can create and modify any file they can open in the text editor. The priority is high for this feature as it is required so that further features can be put into place.

Stimulus/Response Sequences

File handling is necessary to manage all the files. It comes into use when a user opens, edits or deletes a particular file through the text editor. This then accesses both the view and edit filters along with memory access.

Functional Requirements

All implemented using python.
REQ-1: Create file - gives error when user tries to name it the same as an existing file name and is handled by asking user to change file name or replace it.

REQ-2: Open file – error when the mentioned file name doesn't exist
REQ-3: Save file
REQ-4: Close file – asks for saving changes made before closing

Editing

Description and Priority

User can perform many editing operations like cut, copy, paste:
Cut - Select the portion or whole of the text you want to cut and then press cut.
Copy - Select the portion or whole of the text you want to copy and then press copy
Paste - Press the paste button at the place you want to paste the copied text.

Its priority is after the highest priorities as the text editor can still work without these features

Stimulus/Response Sequences

Editing comes into use when a user opens a file and selects some text to modify it. This again opens up the edit filter with access to the memory.

Functional Requirements

REQ-1: Cut text command
REQ-2: Copy text command
REQ-3: Paste text command
REQ-4: Find and Replace

Shortcut keys for the commands would also be greatly helpful and are implemented.

Formatting

Description and Priority

User can use formatting to work with the text in the text editor. They can customize the font to their liking, as well as the syntax highlighting and the overall theme of the editor from some preloaded themes. This has the lowest priority as it's just to make the editor look presentable and is related to the gui.

Stimulus/Response Sequences

Formatting is when the user edits the font features or picks a theme from the available themes.

Functional Requirements

REQ-1: Choose different themes
REQ-2: Indenting and unindenting
REQ-3: Font color and bold, italic, underline

Programming

Description and Priority

Since text editors are most commonly used for programming and coding, some helpful features for coding are included. This includes autocompleting brackets, syntax highlighting and some other functionalities. This has low priority as its just to make the editor easier to use.

Stimulus/Response Sequences

These come into use when a user is coding using text editor. They could also be useful for normal note-taking or texts as well. The user selects some text and then performs the functions on it. It calls the editing filter and memory access too.

Functional Requirements

REQ-1: Autocomplete brackets
REQ-2: Syntax Highlighting
REQ-3: Run current file

Other Non-functional Requirements

Performance Requirements

- Pleasing layout to look at
- Save files to desirable location
- Help/ guide to use the editor
- Should be fast and shouldn't take up much space

Timing relationship for real time system (deadline of tasks)

- week 1: create, open and close files
- week 2: edit files including copy and paste
- week 3: formatting files
- week 4: compiling files

Performance requirements for each function:
(a)reliability of each function
-which includes availability of files
(b)maintainability of files
-scalability of text editor:
-overall easy access of files

Safety Requirements
- files could be lost: this could be prevented by saving the files
- files can be edited and other functions are done only by users
- files can be damaged while compiling

Safety certifications show their commitment to safety and risk management, skills for managing safety of yourself and others
ex: CSP (certified safety professional) certificate obtained after completing ASP certificate.

Security Requirements

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

Software Quality Attributes

Qualities important for both user and customers:

- reusability: each file can be used any number of times for viewing or editing
- flexibility: easy compilation of the files
- robustness: all the functions should be done quick
- adaptability
- availability: space for saving files

All these qualities help the user to easily use all the functions

Business Rules

The user should be able to create as many new Untitled text files as they wish, the application should allow the user to open a file from any drive or network onto the desktop.
After editing a file either on the Source code desktop, the user should be able to save it.
Keeping in mind the same context, the user should have the option of saving a selected file as another new file.
In the event of editing text, the application should support cut, copy and paste operations

Other Requirements

Other non-functionality requirements:

Good response time
Efficiency
Effectiveness
Interface is user friendly
Easy to use
No special downloads necessary
Simple to install

Appendix A: Glossary

- *REQ – Requirement*
- *DBMS – Database Management System*
- *TBD – To Be Determined*
- *Appendix B: Field Layouts*

Appendix B: Field Layouts

Information required to create or open a file

Field	Length	Data Type	Description	Is Mandatory
File name	256	String	Name of the file	Y
File type	-	String	Type of the file	Y

Appendix C: Requirement Traceability Matrix

Sl. No	Requirement ID	Brief Description of Requirement	Architecture Reference	Design Reference	Code File Reference	Test Case ID	System Test Case ID
1.	Req-01	Creating a new file	Document Processing	Digitizer V5	texteditor.py	UT-01	UT-01

			Architecture - IBM				
2.	Req-02	Opening an existing file	Document Processing Architecture - IBM	Digitizer V5	texteditor.py	UT-03	UT-03
3.	Req-03	Saving a file	Document Processing Architecture - IBM	Digitizer V5	texteditor.py	UT-06	UT-06
4.	Req-04	Closing a file	Document Processing Architecture - IBM	Digitizer V5	texteditor.py	UT-10	UT-10
5.	Req-05	Cut (Ctrl+X) text	None	None	Context.py	UT-11	UT-11
6.	Req-06	Paste (Ctrl+V) text	None	None	Context.py	UT-12	UT-12
7.	Req-07	Copy (Ctrl+C) text	None	None	Context.py	UT-14	UT-14
8.	Req-08	Find and Replace	None	None	Find.py	UT-16	UT-16
9.	Req-09	Themes for editor	None	None	Syntax_and_themes.py	UT-21	UT-21
10.	Req-10	Auto-indentation of text	None	None	Context.py	UT-23	UT-23
11.	Req-11	Font formatting	None	None	Context.py	UT-25	Ut-25
12.	Req-12	Auto-completion of brackets	None	None	Texteditor.py	UT-28	UT-28
13.	Req-13	Syntax highlighting for program files	None	None	Syntax_highlighting.py	UT-30	UT-30
14.	Req-14	Running a program file from editor	None	None	Loaders.py	UT-31	UT-31

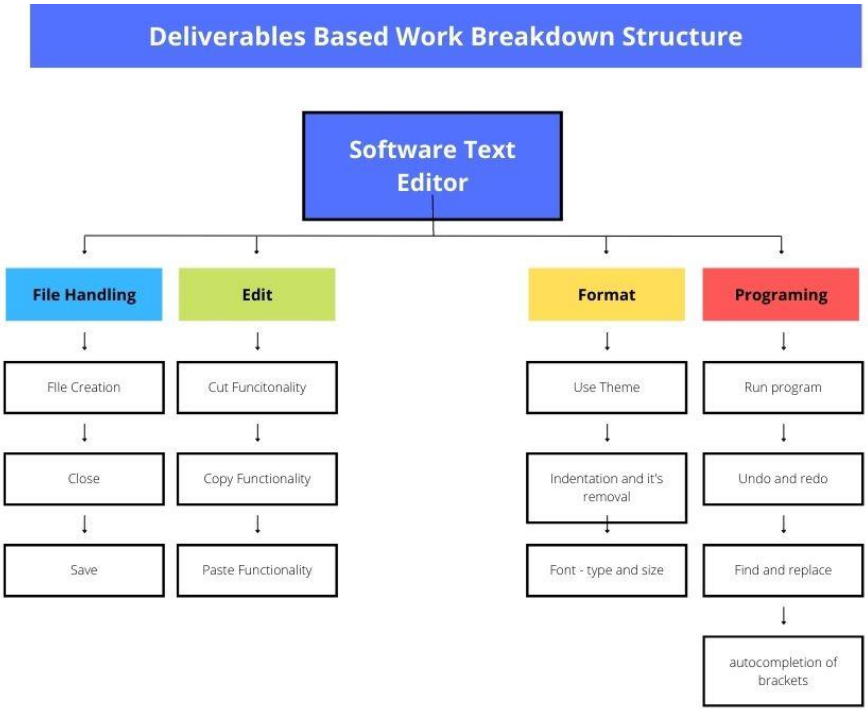
PROJECT PLAN

1. The lifecycle followed for the execution of your project and justify why you have chosen the model.
   Since the software being implemented is small and the there are no major changes in the requirements, we will be using the waterfall models. It accommodates small and less complex projects and has a testing phase at the very end. The tools and techniques used are also not subject to change.

2. The tools used throughout the lifecycle like planning tool, design tool, version control, development tool, bug tracking and testing
   tool.
   Planning tool, Bug tracking tool: Jira software - this is a work management, planning, bug tracking tool for all kind of use cases.
   Design tool: Draw.io, Canva
   Version control: Github
   Development Tools: Python

3. All the deliverables and categorise them as reuse/build components.
   File Handling Module: This is a build component as it consists of the functions that help user create a file, close and save a file.
   Edit Module: This is a build component as it consists of the cut, copy and paste commands and functions to help ease of using the text editor.
   Format Module: This is a re-use component it consists of functions that help a user to choose different themes and indenting and font types and size.
   Program Module: This is a build component as it consists of commands that allow the user to run the text as a program and run it. It specifically consists of: Undo and redo commands, autocomplete brackets, run current file, find and replace functions.

4. WBS for the functionalities in detail
   A deliverable based WBS is shown below



5. A rough estimate of effort required to accomplish each task in terms of person months.

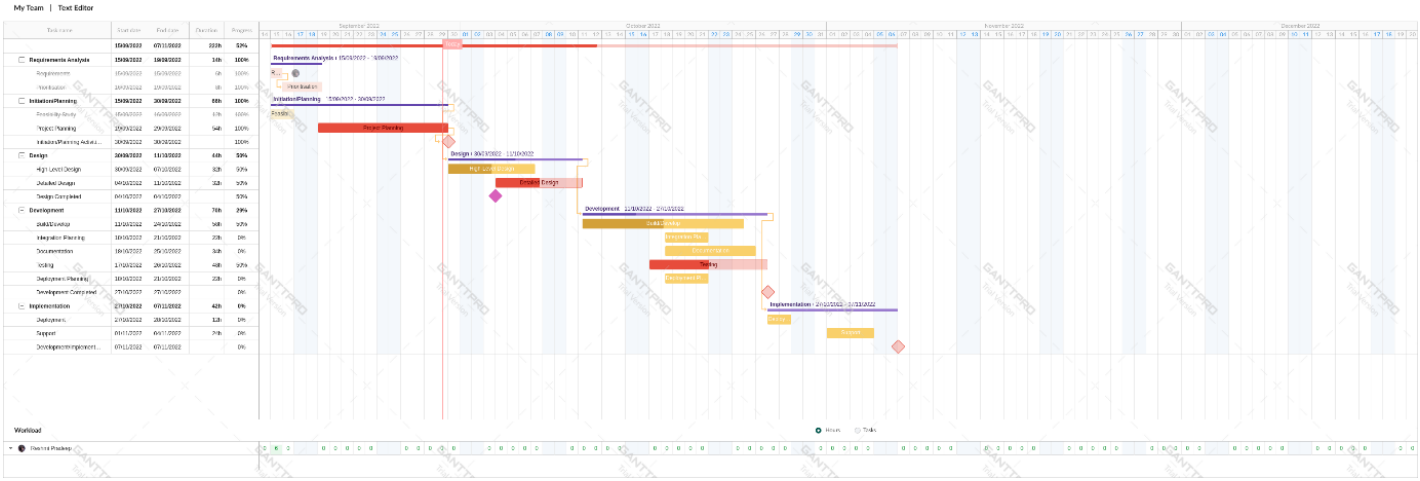
   With respect to the Constructive Cost Model (COCOMO), schedule and cost are related as

   P= KLOC / E
   where, P=Productivity, KLOC = Kilo lines of code (the estimated size) and E= Effort (the total effort in person months)

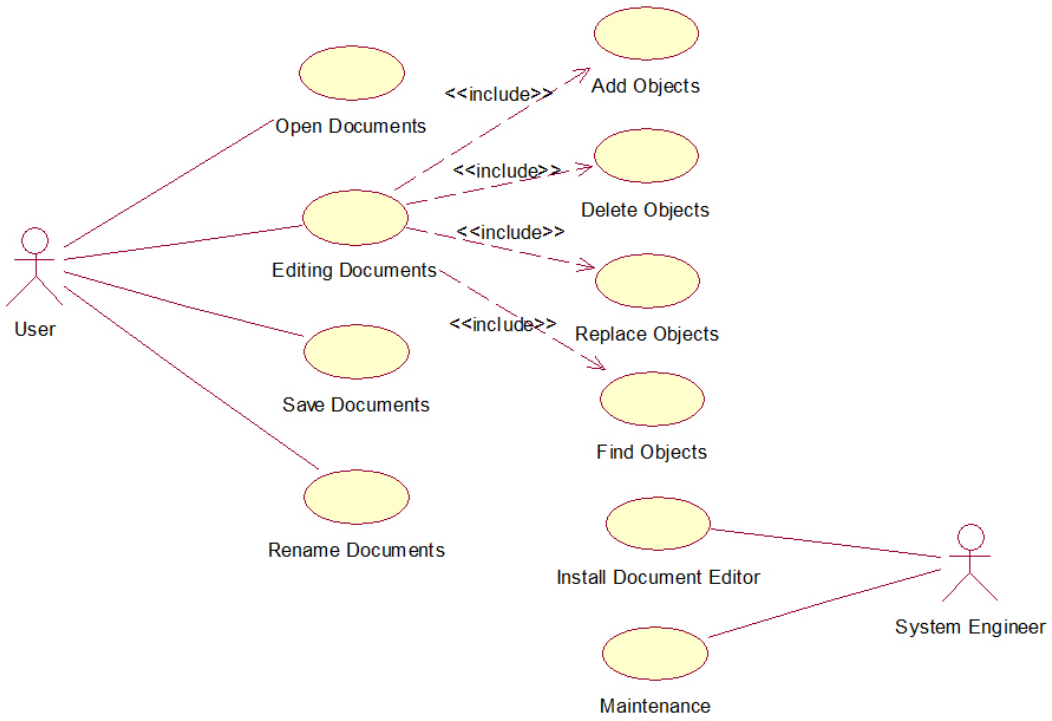
   Since this is a rough estimate, consider the value of KLOC to be 40.
   Considering this to be organic,
   Effort = a1 * KLOC * a2 (PM)
   Where, a1=2.4 and a2=1.05
   Therefore, Effort = (2.4)(40)(1.05)PM = 100.8 PM

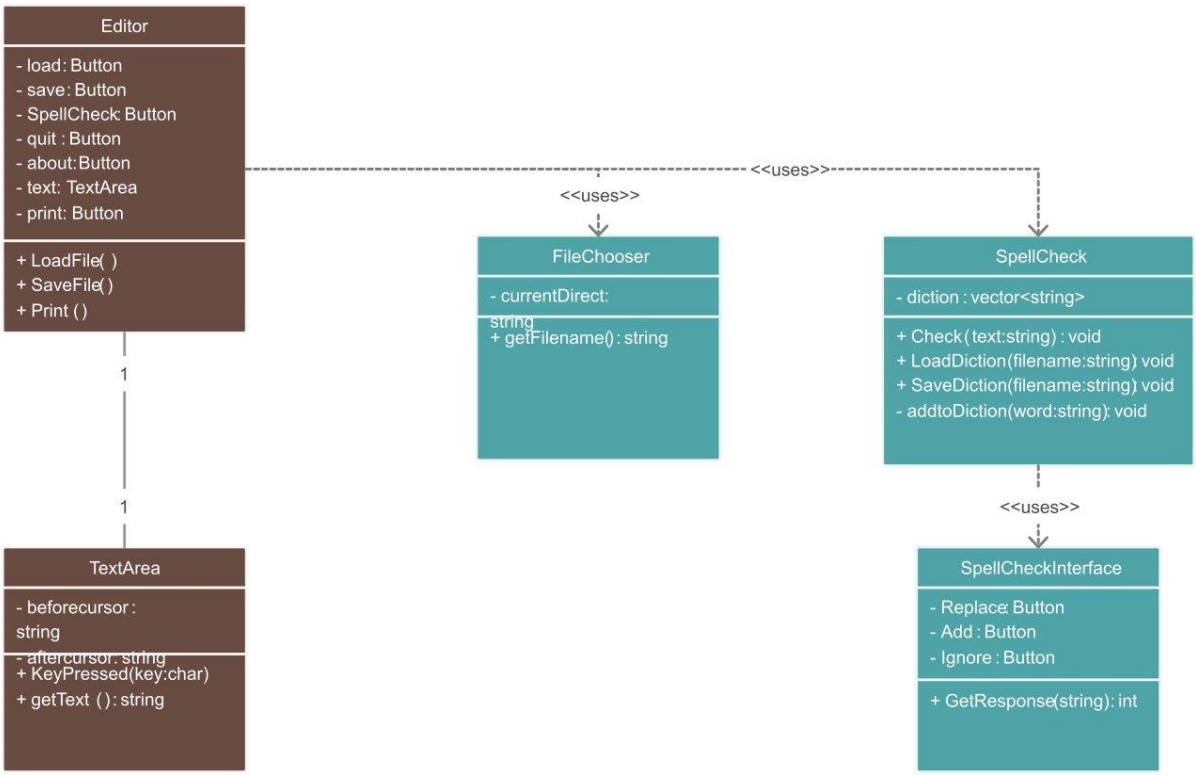
   Productivity, P = KLOC / Effort
   Therefore, P = 40 / 100.8 = 396.8 LOC/PM

6. Create the Gantt Chart for scheduling using any tool.



DESIGN DIAGRAMS





TEST CASES

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
UT-01	File Handling	To test creating a new file	Open the text editor. Text file called File1 should not exist.	1: Click on File in the menu bar 2: Click on New 3: Enter a valid file name and type 4: Click on Save	File name: File1 File type: Text File	New text file should be created and opened with name File1	New text file should created and opened with name File1	Pass
UT-02	File Handling	To test creating a new file	Open the text editor. A text file named File1 should already be created.	1: Click on File in the menu bar 2: Click on New 3: Enter a file name that already exists 4: Click on Save	File name: File1 File type: Text File	Error message box that displays "File1.txt already exists. Do you want to replace it?"	Error message box that displays "File1.txt already exists. Do you want to replace it?"	Pass
UT-03	File Handling	To test opening a file	Open the text editor. A text file named File1 should already be created.	1: Click on File in the menu bar 2: Click on Open 3: Enter a file name that already exists 4: Click on Open	File name: File1 File type: Text File	Opens the text file called File1	Opens the text file called File1	Pass
UT-04	File Handling	To test opening a file	Open the text editor. A text file named File2 should not exist.	1: Click on File in the menu bar 2: Click on Open 3: Enter a file name that doesn't exist 4: Click on Open	File name: File2 File type: Text File	Error message box that displays "File2.txt not found. Check the file name and try again"	Error message box that displays "File2.txt not found. Check the file name and try again"	Pass
UT-05	File Handling	To test opening a file	Open the text editor. An image called Pic.png should exist.	1: Click on File in the menu bar 2: Click on Open 3: Enter a file name with a type that is not supported on the text editor 4: Click on Open	File name: Pic File type: png	Error message box that displays "This file type is not supported"	Error message box that displays "This file type is not supported"	Pass
UT-06	File Handling	To test saving a file	Open the text editor. Text file called File1 should not exist.	1: Click on File in the menu bar 2: Click on Save 3: Enter a valid file name and type 4: Click on Save	File name: File1 File type: Text File	Text file should be saved with name File1	Text file should be saved with name File1	Pass
UT-07	File Handling	To test saving a file	Open the text editor. Text file called File1 should exist.	1: Click on File in the menu bar 2: Click on Save 3: Enter a valid file name and type	File name: File1 File type: Text File	Error message box that displays "File1.txt already exists. Do	Error message box that displays "File1.txt already exists. Do	Pass

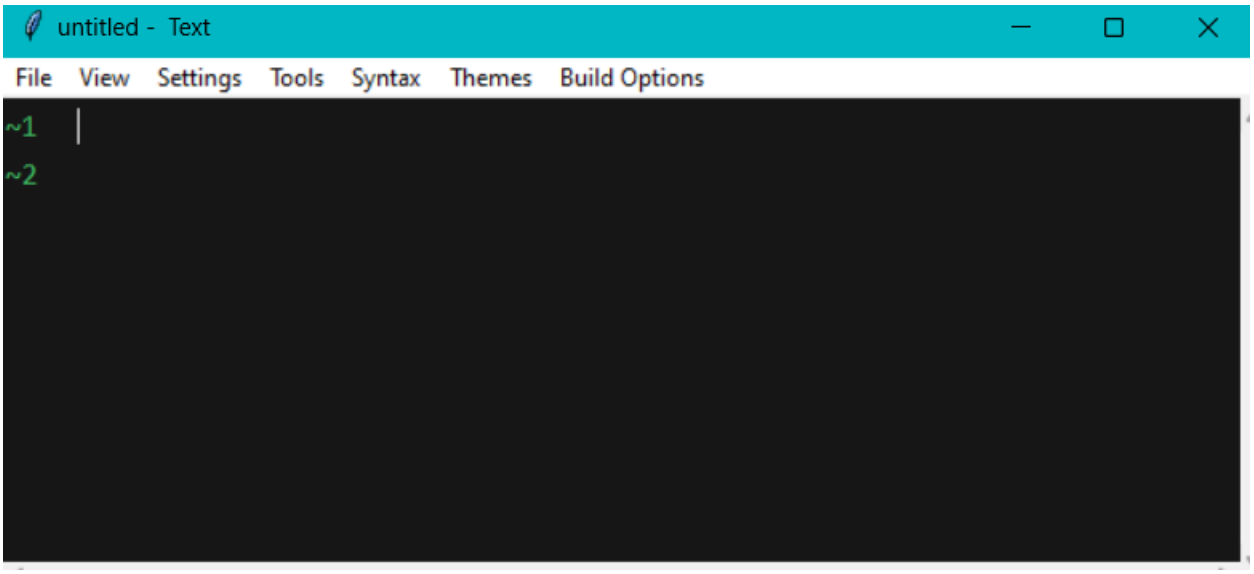
				4: Click on Save		you want to replace it?"	you want to replace it?"	
UT-08	File Handling	To test saving a file	Open the text editor. Image file called File1 should not exist.	1: Click on File in the menu bar 2: Click on Save 3: : Enter a file name with a type that is not supported on the text editor 4: Click on Save	File name: File1 File type: png	Error message box that displays "This file type is not supported"	Error message box that displays "This file type is not supported"	Pass
UT-09	File Handling	To test saving changes to a file	Open the text editor. Image file called File1 should already be saved and open.	1: Click on File in the menu bar 2: Click on Save	None	"Changes saved" in status bar	"Changes saved" in status bar	Pass
UT-10	File Handling	To test closing a file	Open the text editor. Have a file open	1: Click on File in the menu bar 2: Click on Exit	None	Message box that displays "Save the changes before closing?"	Message box that displays "Save the changes before closing?"	Pass
UT-11	Editing	To test editing a file	Open the text editor. Have a file open with text in it.	1: Select some text. 2: Ctrl + X	Text: This a cut testing Selected text: This is	Text: A cut testing	Text: A cut testing	Pass
UT-12	Editing	To test editing a file	Open the text editor. Have a file open with text in it.	1: Select some of the text. 2: Ctrl + X 3: Ctrl + V at the end of the text	Text: This a cut and paste testing. Selected text: This is	Text: A cut and paste testing.This is	Text: A cut and paste testing.This is	Pass
UT-13	Editing	To test editing a file	Open the text editor. Have a file open with text in it.	1: Don't select any text. 2: Ctrl + X 3: Ctrl + V at the end of the text	Text: This a cut testing. Selected text:	Text remains the same	Text remains the same	Pass
UT-14	Editing	To test editing a file	Open the text editor. Have a file open with text in it.	1: Select some of the text. 2: Ctrl + C	Text: This a copy testing. Selected text: This is	Text remains the same	Text remains the same	Pass
UT-15	Editing	To test editing a file	Open the text editor. Have a file open with text in it.	1: Select some of the text. 2: Ctrl + C 3: Ctrl + V at the end of the text	Text: This a copy testing. Selected text: This is	Text: This is a copy testing.This is	Text: This is a cut testing.This is	Pass
UT-16	Editing	To test find and replace	Open the text editor. Have a file open with text in it.	1: Go to Tools on the menu bar. 2: Select Find and Replace from drop down list	None	Opens find and replace window with text boxes for find and replace	Opens find and replace window with text boxes for find and replace	Pass

UT-17	Editing	To test find and replace	Open the text editor. Have a file open with text in it.	1: Go to Tools on the menu bar. 2: Select Find and Replace from drop down list 3: Type some text in the Find text box 4: Click on Search	File Text: This is find and replace testing Find Text Box Text: testing	Selects "testing" in the File	Selects "testing" in the File	Pass
UT-18	Editing	To test editing a file	Open the text editor. Have a file open with text in it.	1: Go to Tools on the menu bar. 2: Select Find and Replace from drop down list 3: Type some text that is not in the file in the Find text box 4: Click on Search	File Text: This is find and replace testing Find Text: searching	Error box that displays "No more results. Search from the beginning?"	Error box that displays "No more results. Search from the beginning?"	Pass
UT-19	Editing	To test editing a file	Open the text editor. Have a file open with text in it.	1: Go to Tools on the menu bar. 2: Select Find and Replace from drop down list 3: Type some text in the Find text box 4: Click on Search 5: Type some text in Replace text box 6: Click on Replace	File Text: This is find and replace testing Find Text: testing Replace Text: functioning	File Text: This is find and replace functioning	File Text: This is find and replace functioning	Pass
UT-20	Editing	To test editing a file	Open the text editor. Have a file open with text in it.	1: Go to Tools on the menu bar. 2: Select Find and Replace from drop down list 5: Don't type anything in Replace text box 6: Click on Replace	File Text: This is find and replace testing Find Text Box Text: searching Replace Text:	File Text: This is find and replace	File Text: This is find and replace	Pass
UT-21	Formatting	To test changing themes	Open the text editor. Have a file open with text in it.	1: Go to Themes on the menu bar. 2: Select a theme from the drop-down list.	Theme: desert	The theme changes from default to the desert theme	The theme changes from default to the desert theme	Pass
UT-22	Formatting	To test changing themes	Open the text editor. Have the desert theme applied on	1: Go to Themes on the menu bar. 2: Select a theme that is already applied from the drop-down list.	Theme: desert	The theme remains the same.	The theme remains the same.	Pass

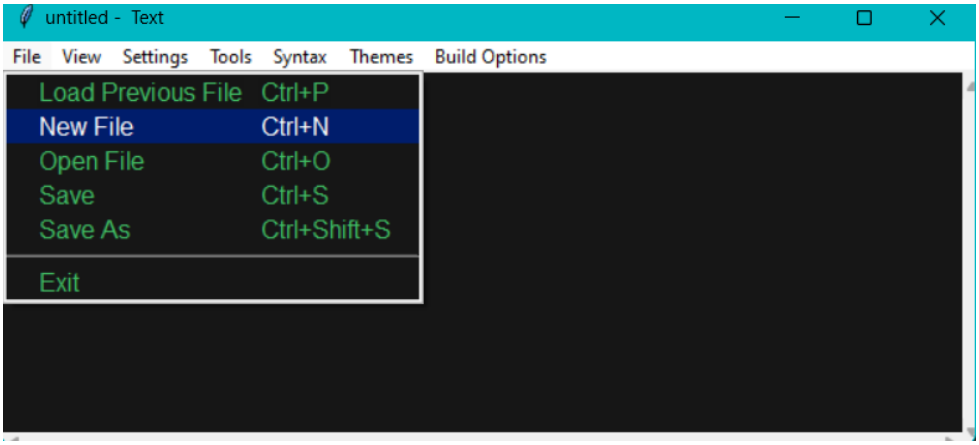
			the file that is open.					
UT-23	Formatting	To test indentation of text	Open the text editor. Open file that contains text.	1: Go to the beginning of a line and press tab. 2: Go to the end of the same line and press enter.	None	One tab space is automatically applied to the next line	One tab space is automatically applied to the next line	Pass
UT-24	Formatting	To test indentation of text	Open the text editor. Open file that contains text.	1: Go to the beginning of an indented line and press backspace. 2: Go to the end of the same line and press enter.	None	One tab space indentation is automatically removed from the next line	One tab space indentation is automatically removed from the next line	Pass
UT-25	Formatting	To test text formatting	Open the text editor. Have a file open with text in it.	1: Select some text. 1: Go to Tools on the menu bar. 2: Select Color selector from drop down list 3: Pick a color 4: Click apply	Text: This is test sample Color: Green	Display “This is test sample” in Green	Display “This is test sample” in Green	Pass
UT-26	Formatting	To test text formatting	Open the text editor. Have a file open with text in it.	1: Select some text 2: Press Ctrl+B	Text: This is test sample	Display “This is test sample” in bold formatting	Display “This is test sample” in bold formatting	Pass
UT-27	Formatting	To test text formatting	Open the text editor. Have a file open.	1: Don’t select any text 2: Press Ctrl+B	Text: This is test sample	No changes made to text	No changes made to text	Pass
UT-28	Programmi-ng	To test auto-completion	Open the text editor. Have a program file open.	1: Type a left bracket of any type	Text: {	adds }	adds }	Pass
UT-29	Programmi-ng	To test auto-completion	Open the text editor. Have a program file open.	1: Type a single or double quote	Text: “	Adds second “	Adds second “	Pass
UT-30	Programmi-ng	To test syntax highlightin g	Open the text editor. Have a program file open with text in it.	1: Go to Syntax on the menu bar. 2: Select the corresponding file type from drop-down list.	File Type : Python Syntax: Python	The python file syntax highlight is applied to the program	The python file syntax highlight is applied to the program	Pass
UT-31	Programmi-ng	To test running a program	Open the text editor. Have a program file open.	1: Go to Build Options on the menu bar. 2: Select Run from the drop-down list	File Type: Python Program: To print hello	Prints “Hello” in a cmd window	Prints “Hello” in a cmd window	Pass

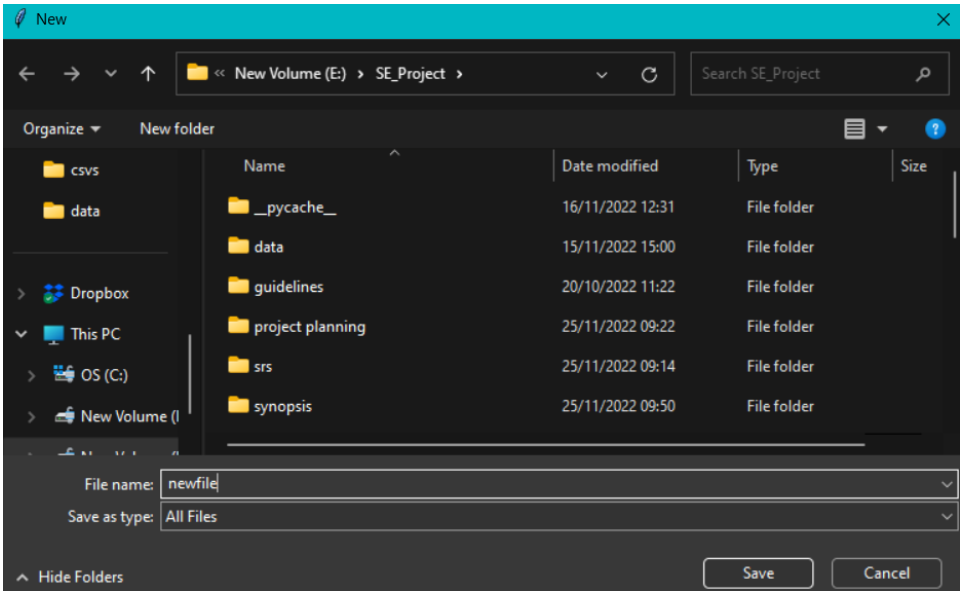
SCREENSHOTS OF OUTPUT

Text Editor:

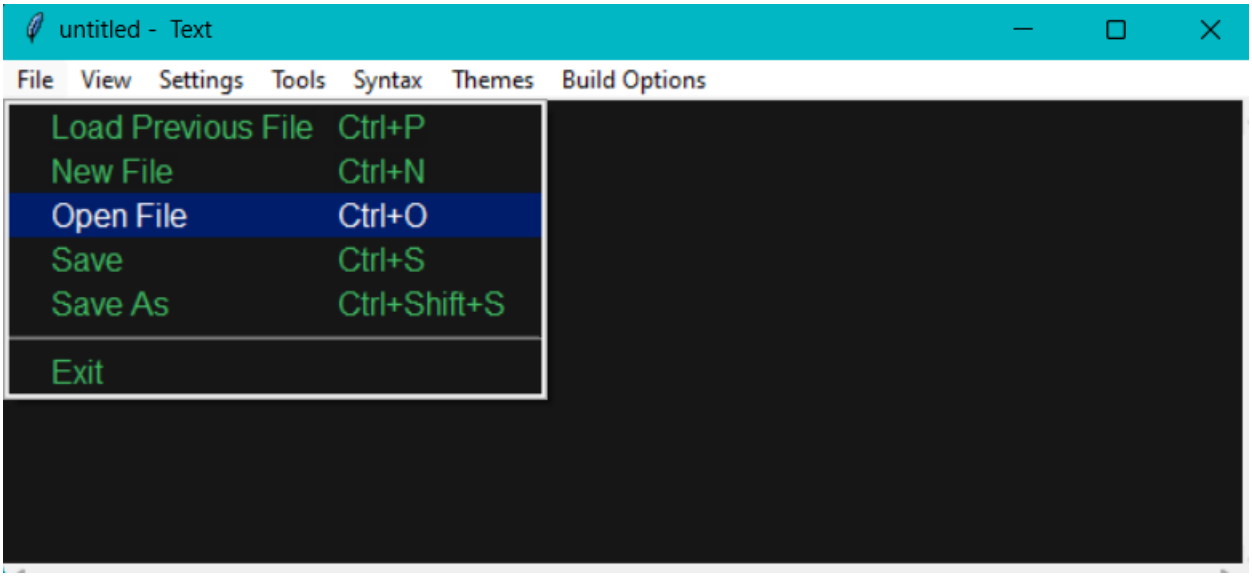


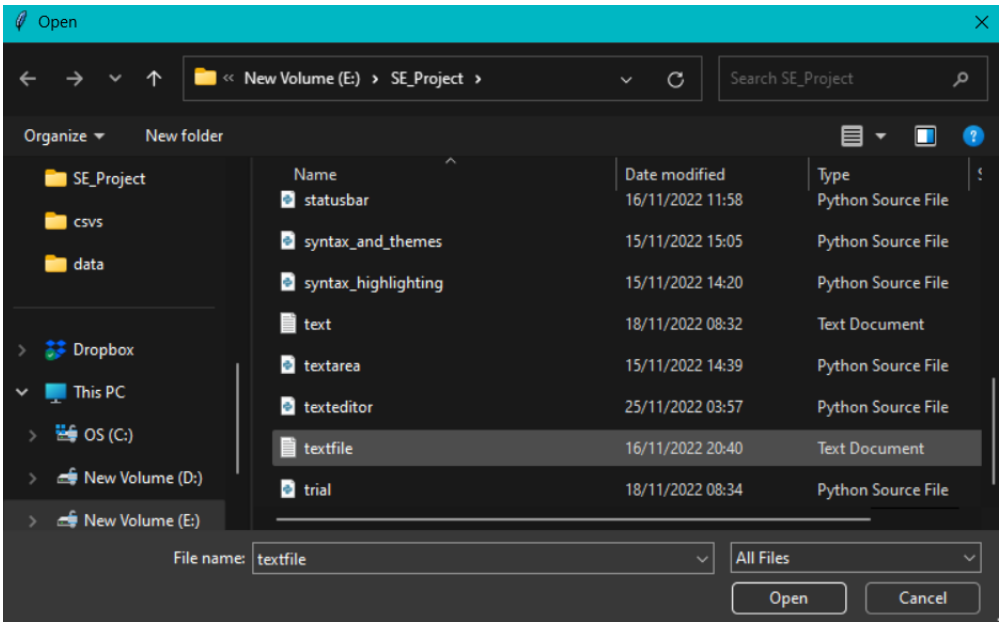
Creating a new file:

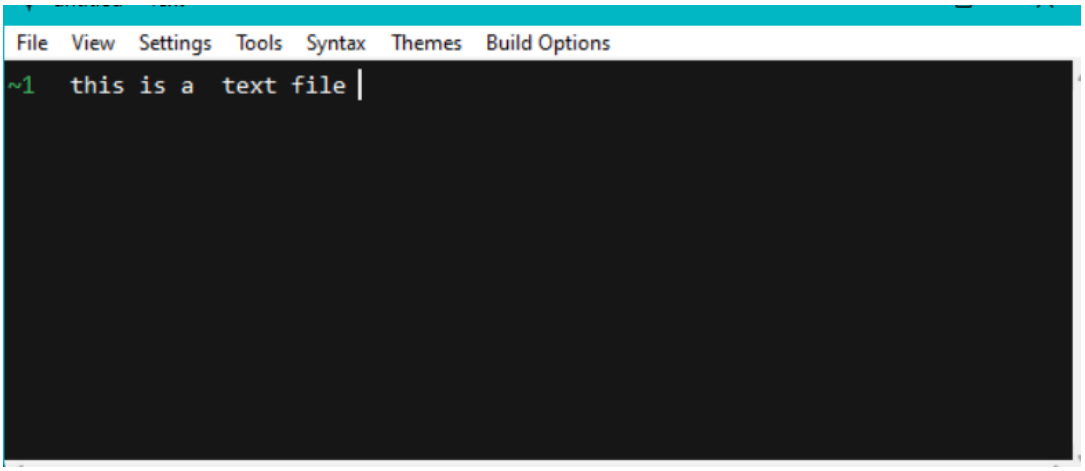




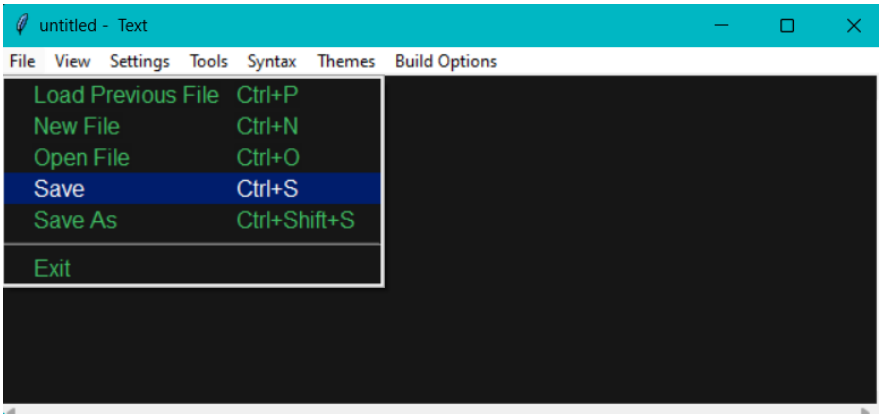
Opening a file:

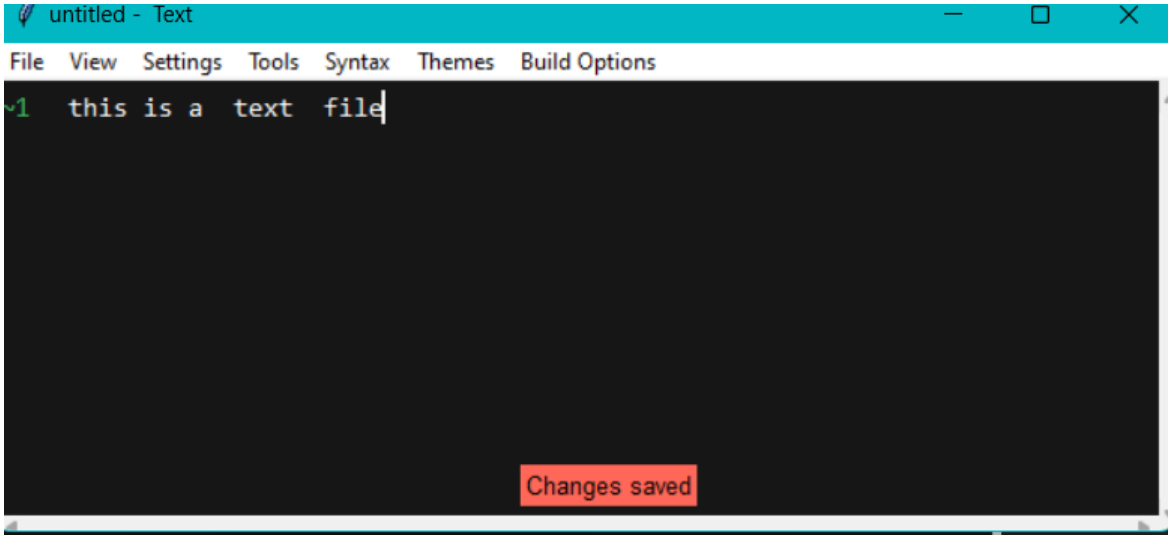




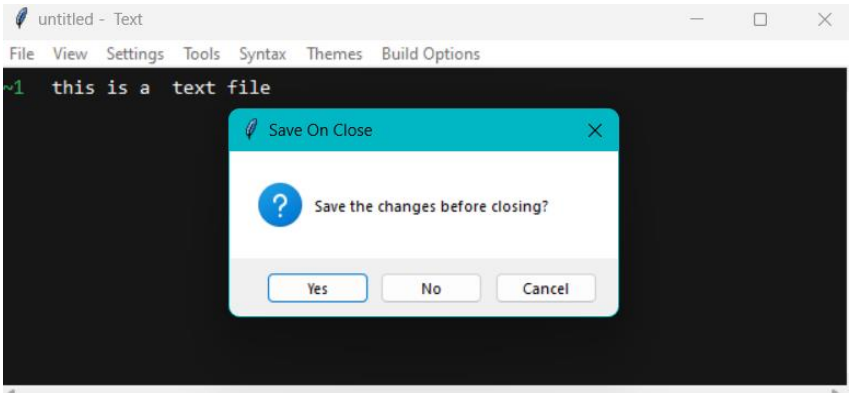


Saving a file:

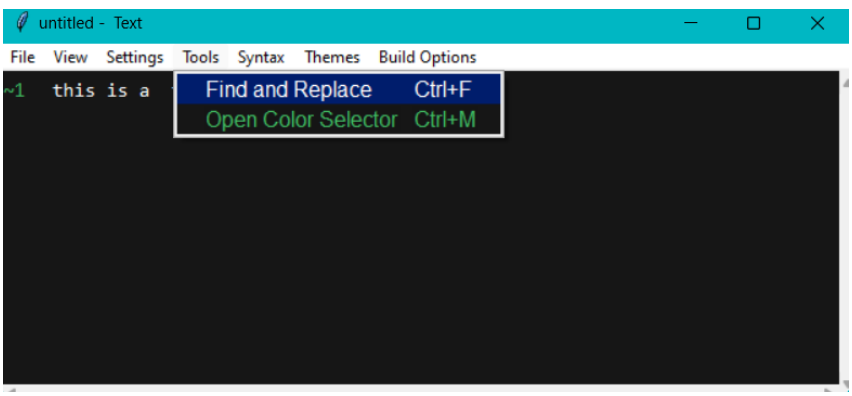


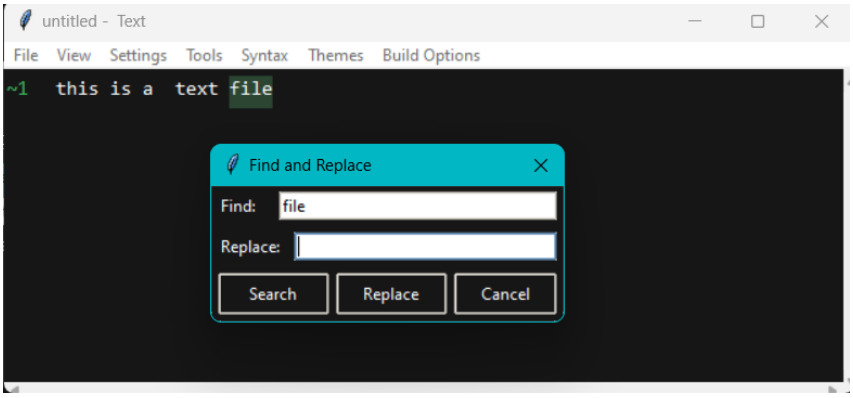


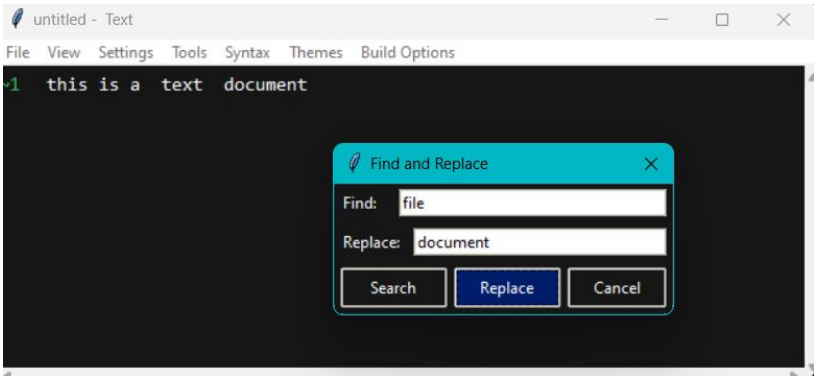
Closing a file:



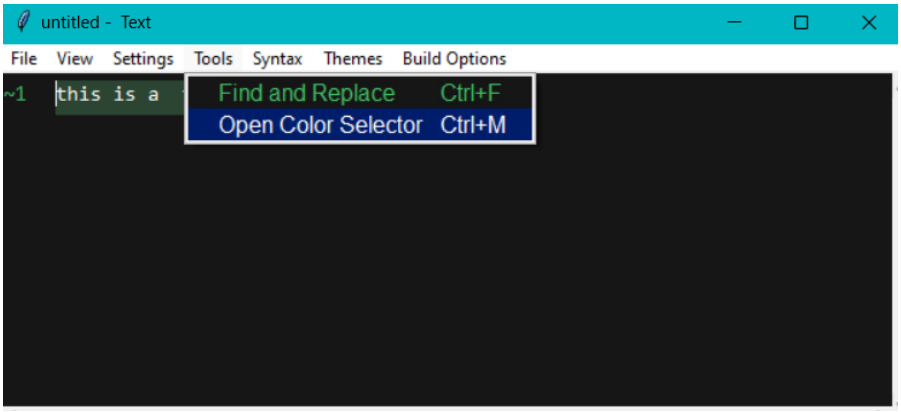
Find and Replace:

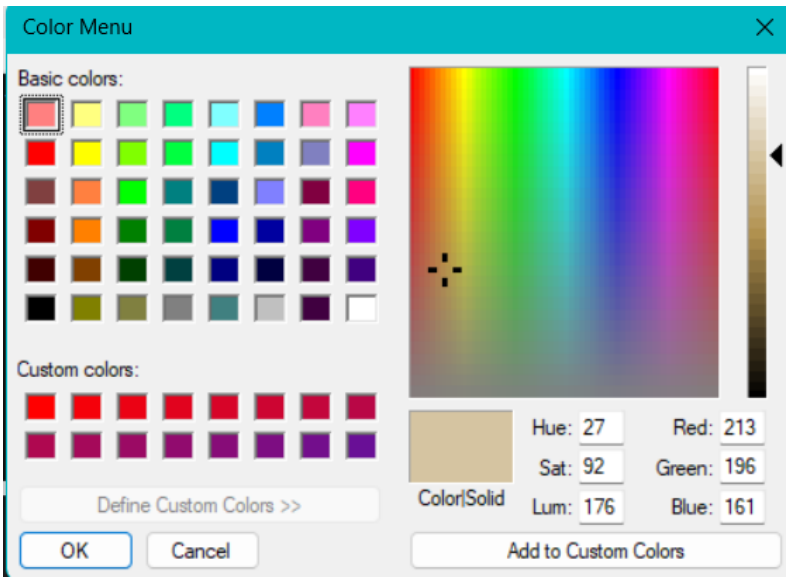




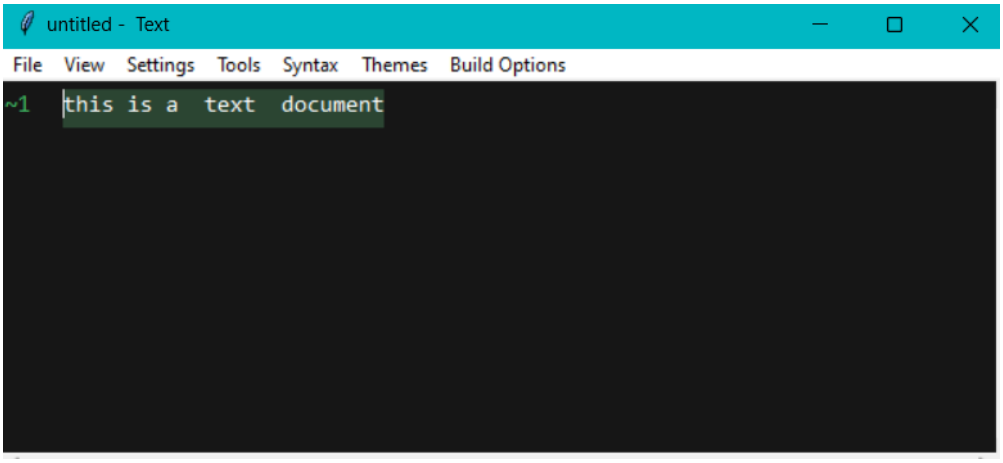


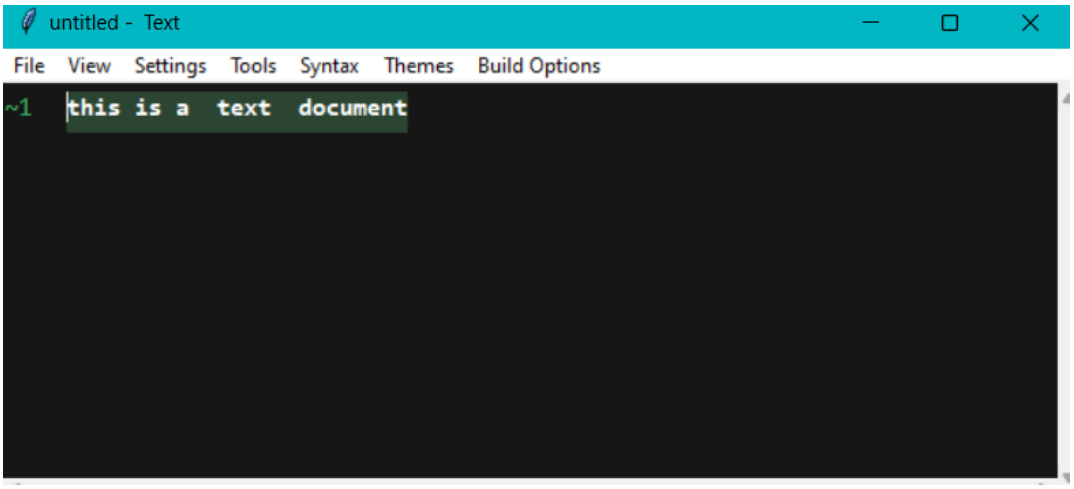
Changing font color:



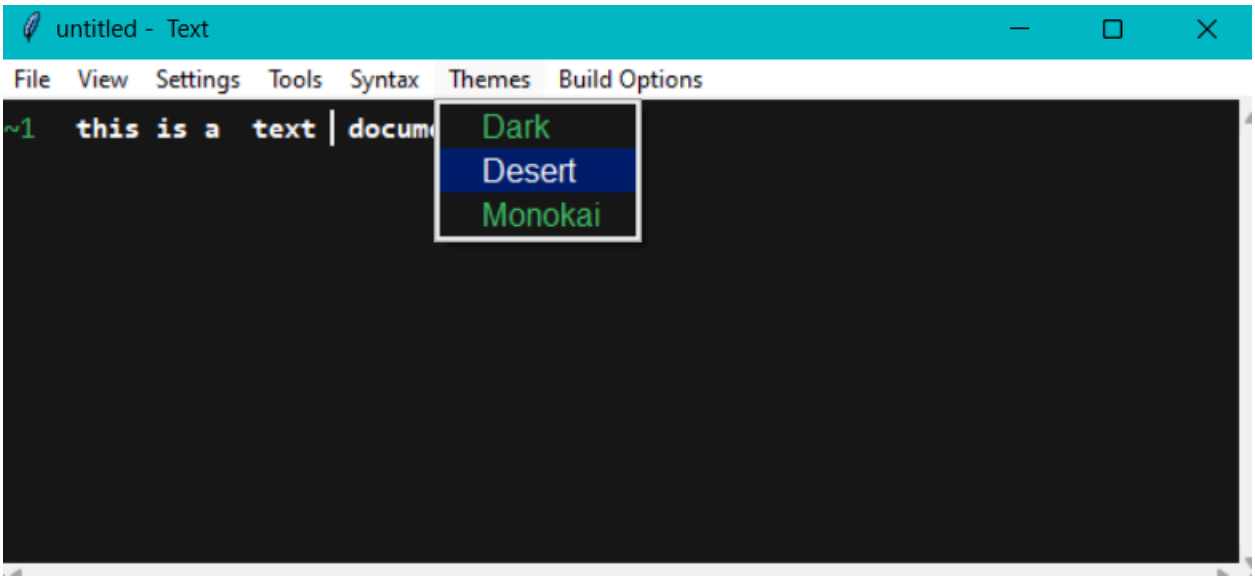


Formatting:

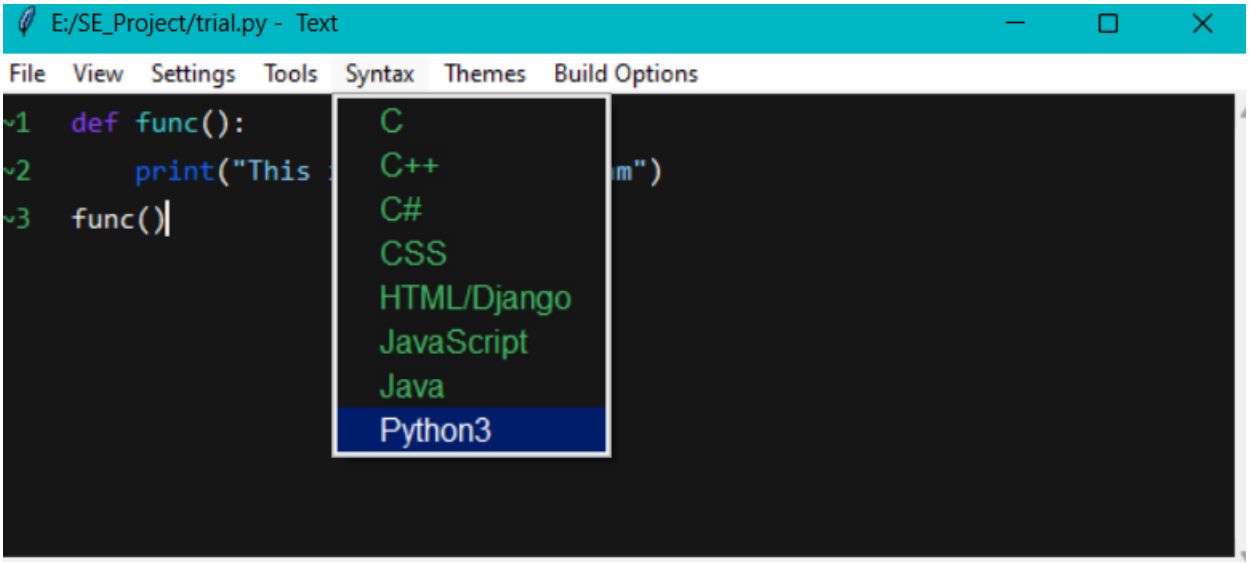




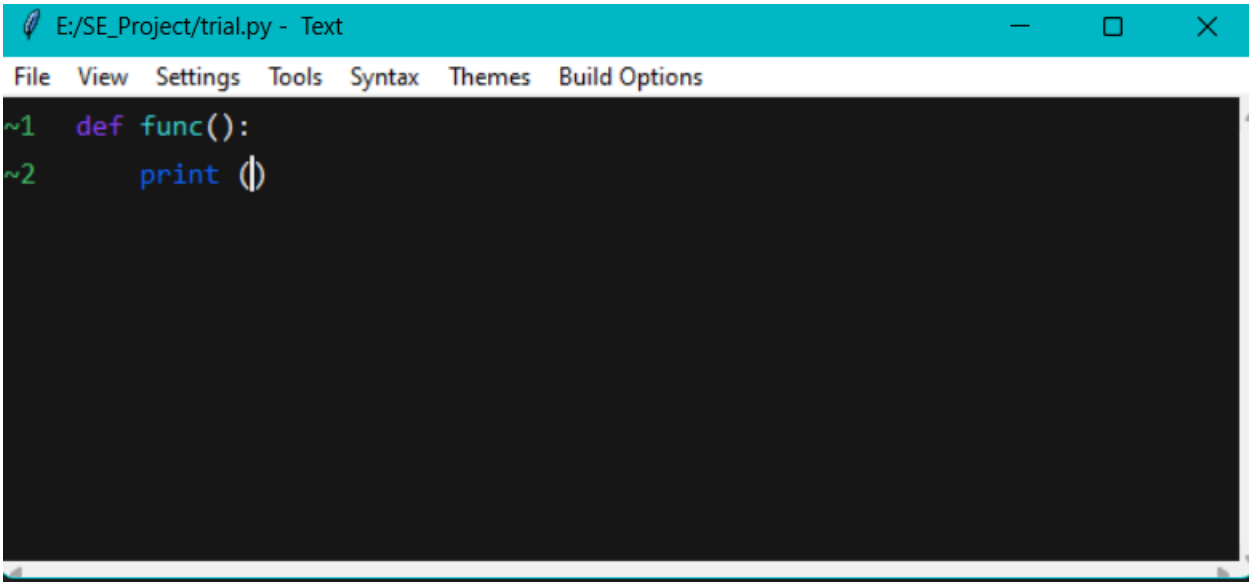
Changing themes:

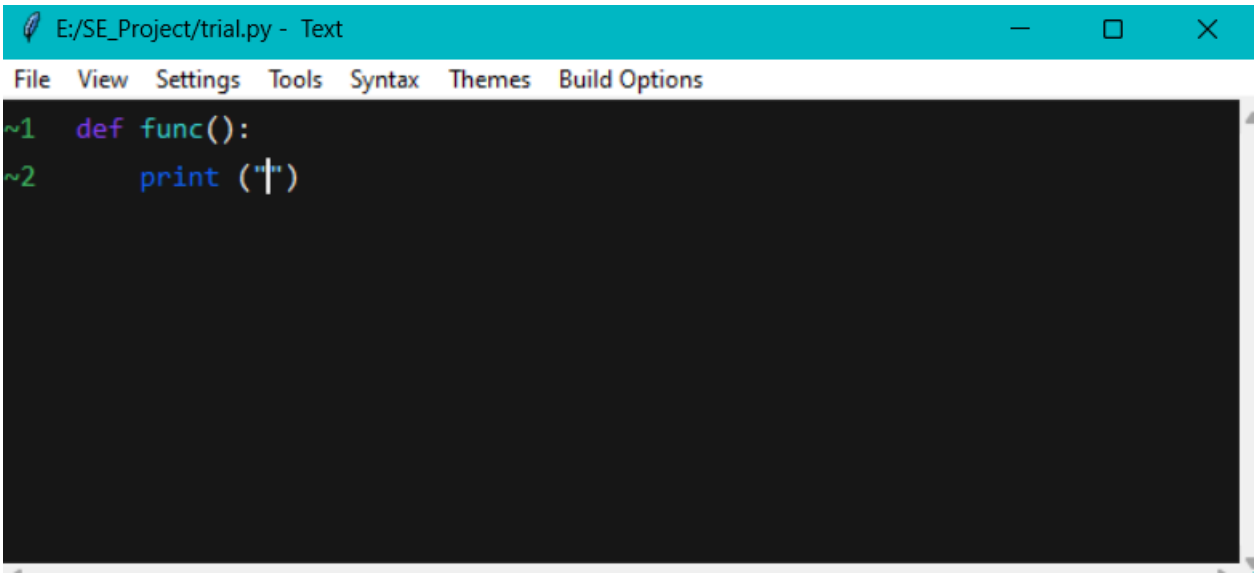


Syntax Highlighting:



Auto-Complete for brackets and quotes:





Running a program: