CmpSci 4500

Intro to Software Profession

Small Group Project 3 (SG3)

Design Document

Group Number: 5

Group Members: Abi Baniya, Kayla Gaynor, Elisa Reyes, Rashmika Srivastava, Vincent Stech, DeJa Thompson

December 2, 2025

Revision Date	**Revised By**	**Description of Change**	**File(s) Affected**	**Commit ID**
12/02/2025	Rashmika	Initial commit	README.md	a0037d7
12/03/2025	Kayla	Added files via upload	SG2.py	1657909
12/03/2025	Kayla	Added files via upload	CONCORDANCE.txt	39ca42c
12/03/2025	Kayla	Clean up	SG2.py	ac77402
12/07/2025	Rashmika	Initial SG3 project upload	SG3.py	2a2e7a6
12/09/2025	Kayla	Uploaded DFD level 0 & level 1	SG3_DFD_level_0.drawio.png, SG3_DFD_level_1.drawio.png	4d77dae, 55afc29
12/09/2025	Kayla	Uploaded static call graph	SG3_Static_Call_Graph.drawio.png	bebb521

Table of Contents

1.0. Introduction

SG3 is a Python program that provides a graphical interface for processing text files, searching for words, and generating structured outputs. It expands on the functionality of previous projects by allowing users to open multiple files, analyze their contents, and create concordance and summary lists through an easy-to-use Tkinter GUI.

2.0. General Description

This program accepts up to ten .txt files selected by the user through the GUI. After a file is opened, the program extracts all legal words using regular expressions and stores them for later analysis. The user may search for a specific word across all open files, and the results are displayed in the output panel. When the concordance option is selected, the program re-reads all open files and generates a complete concordance showing each word and every position it appears in. SG3 also creates three additional summary lists: the top ten most frequent words, the words appearing in all files, and the words appearing in only one file. All results are shown in the GUI and written to external text files for reference.

3.0. Interface Requirements

Desktop, laptop, or workstation capable of running Python and displaying a graphical interface is required. The system must support Tkinter window rendering, mouse input, and basic keyboard entry for user commands.

4.0. Performance Requirements

Any computer capable of running Python 3 and executing Tkinter applications is sufficient. The program performs lightweight text processing and can operate efficiently as long as the machine supports standard file I/O and GUI event handling.

5.0. Design Description

SG3 is implemented using modular functions and a class-based structure to improve clarity, reuse, and reliability. The program separates GUI components from text-processing logic, with the SG3App class managing user interaction while helper functions handle file parsing, word extraction, and concordance creation. A short main function initializes the graphical window and starts the Tkinter event loop. All development and testing were performed in Python using the Thonny IDE.

5.1. Functions / Main Logic

main() - Creates the Tkinter root window, constructs an SG3App object, and starts the GUI event loop. This is the entry point for the program.

SG3App._init_(root) - Stores the root window, sets the title and size of the main window, initializes the filenames and word_arrays lists, and calls build_layout() to create the GUI.

build_layout() - Builds the main interface. It creates the left menu with the buttons for each operation (open file, search word, build concordance, close file, quit) and a scrollable text box on the right for displaying the results. Each button is connected to its corresponding method.

display(text) - Clears the output text box and writes the given text. All user-visible results (status messages, search results, concordance, and extra lists) are shown through this function.

open_file() - Handles file selection and initial processing. It enforces the maximum of ten open files, checks for .txt extension and duplicates, reads the selected file, calls extract_words() to get all legal words, stores the filename and its word list, and reports the file name and total word count via display().

extract_words(text) - Uses a regular expression to remove hyphen line breaks and extract all legal words from a block of text. It returns a list of words for later analysis.

search_word() - Starts the “Find a Word” operation. It creates a small popup window where the user enters a search word and defines the inner function run_search() that performs the actual search.

run_search() (inner) - Validates the user’s word against the legal-word regex rule. It then counts how many times the word appears in each open file’s word array, builds a simple report with file names and counts, and calls display() to show the results before closing the popup.

build_concordance_gui() - Begins the concordance operation. It opens a window listing all open files and defines the inner function run_concordance() that creates the concordance and extra lists.

run_concordance() (inner) - Validates that a file is selected, then calls build_concordance() and format_concordance() to create the concordance for all open files. It writes the formatted concordance to CONCORDANCE.TXT, calls generate_extra_lists() to create summary lists, and sends a combined report (including confirmation messages, concordance, and extra lists) to display().

build_concordance(filenames, word_arrays) - Reopens each file and scans every line and word position. It builds a dictionary mapping each lowercase word to a list of locations (file number, line number, and position in line). This structure is used for both the concordance display and the extra lists.

format_concordance(concordance) - Converts the concordance dictionary into a multi-line string. For each word, it lists all recorded locations in a readable format that can be saved to a file or shown in the GUI.

generate_extra_lists(filenames, word_arrays, concordance) - Builds a statistics table for each word (total occurrences and the set of files it appears in). It then calls generate_top_ten(), generate_words_in_all_files(), and generate_words_in_one_file() to produce three text sections, writes them to ExtraLists.txt, and returns the combined text.

generate_top_ten(word_stats) - Selects up to ten words with the highest total counts and formats them into a small table showing the word, its total count, and how many files it appears in.

generate_words_in_all_files(word_stats, file_count) - Finds all words that appear in every open file and returns them as a simple alphabetical list. If no such words exist, it returns a message stating that.

generate_words_in_one_file(word_stats) - Finds words that appear in exactly one file and lists them along with the file number where they appear. If there are no such words, it returns a message stating that.

close_file() - Allows the user to close one of the currently open files. It shows a list of filenames and the inner function run_close() removes the selected filename and its word array from the internal lists and displays a confirmation message.

run_close() (inner) - Performs the actual removal of the chosen file's data from filenames with word_arrays, then calls display() with a "Closed file" message and closes the popup window.

quit_app() - Closes the main window by destroying the Tkinter root object, which ends the program.

5.2. Pseudocode

```
MAIN

        Create main GUI window

        Create SG3App object using window

        Start GUI event loop

END MAIN

CLASS SG3App

        INIT(root)

                Store root window

                Set window title and size

                Initialize empty lists:

                        Filenames, word_arrays

                Call BUILD_LAYOUT

        END INIT

BUILD LAYOUT
```

```
    Create main frame with:
        Left menu panel
        Right scrollable text output box
    Add buttons to left menu:
        “Open Text File” -> OPEN_FILE
        “Find a Word” -> SEARCH_WORD
        “Build Concordance” -> BUILD_CONCORDANCE_GUI
        “Close a File” -> CLOSE_FILE
        “Quit” -> QUIT_APP
END BUILD_LAYOUT
DISPLAY(text)
    Clear output box
    Insert text into output box
END DISPLAY
OPEN_FILE
    If 10 files already open -> show error and return
    Ask user to choose a .txt file
    Validate extension and that it is not already open
    Read entire file into a string
    Words <- EXTRACT_WORDS(text)
    Append filename to filenames list
    Append words to word_arrays list
    Show message with filename and total word count using DISPLAY
```

```
END OPEN_FILE
SEARCH_WORD
    If no files open -> show error and return
    Open small window asking user to enter a legal word
    Inner function RUN_SEARCH:
        Read word, validate it with regex rule
        For each (filename, words_array) in open files
            Count occurrences of word
            Build one line of output with filename and count
        Send combined results to DISPLAY
        Close search window
END SEARCH_WORD
BUILD_CONCORDANCE_GUI
    If no files open -> show error and return
    Open window listing all open filenames
    Inner function RUN_CONCORDANCE:
        Ensure a file is selected (for spec compliance)
        Concordance <- BUILD_CONCORDANCE(filenames, word_arrays)
        Formatted <- FORMAT_CONCORDANCE(concordance)
        Write formatted to “CONCORDANCE.TXT”
        Extra_text <- GENERATE_EXTRA_LISTS(filenames, word_arrays,
concordance)
        Show confirmation + formatted concordance + extra lists using DISPLAY
        Close selection window
```

```
END BUILD_CONCORDANCE_GUI
CLOSE_FILE
    If no files open -> show error and return
    Open window listing all open filenames
    Inner function RUN_CLOSE:
        Get selected filename
        Remove that filename and its word array from lists
        Show “Closed file” message using DISPLAY
        Close window
END CLOSE_FILE
QUIT_APP
    Destroy root window and exit program
END QUIT_APP
FUNCTION EXTRACT_WORDS(text)
    Clean hyphen-split lines
    Return list of all “legal” words found by regex
END FUNCTION
FUNCTION BUILD_CONCORDANCE(filenames, word_arrays)
    For each file in filenames
        Reopen file, read all lines
        For each line and each word position
            Convert word to lowercase
            Record (file_number, line_number, position) in concordance[word]
        Return concordance dictionary
END FUNCTION
FUNCTION GENERATE_EXTRA_LISTS(filenames, word_arrays, concordance)
    Build word_stats: for each word, total count and set of files
    Create:
        TOP TEN frequent words
        Words appearing in ALL files
        Words appearing in ONLY ONE file
    Combine these sections into one text,
    Write it to "ExtraLists.txt",
    And return the text
END FUNCTION
```

5.3 Pseudocode Revision History

Version	Date	Editor	Version Description
1.0	12/09/2025	Kayla	Initial pseudocode
1.1	12/09/2025	Kayla	Updated pseudocode