Project Documentation: Multi-threaded File Searcher

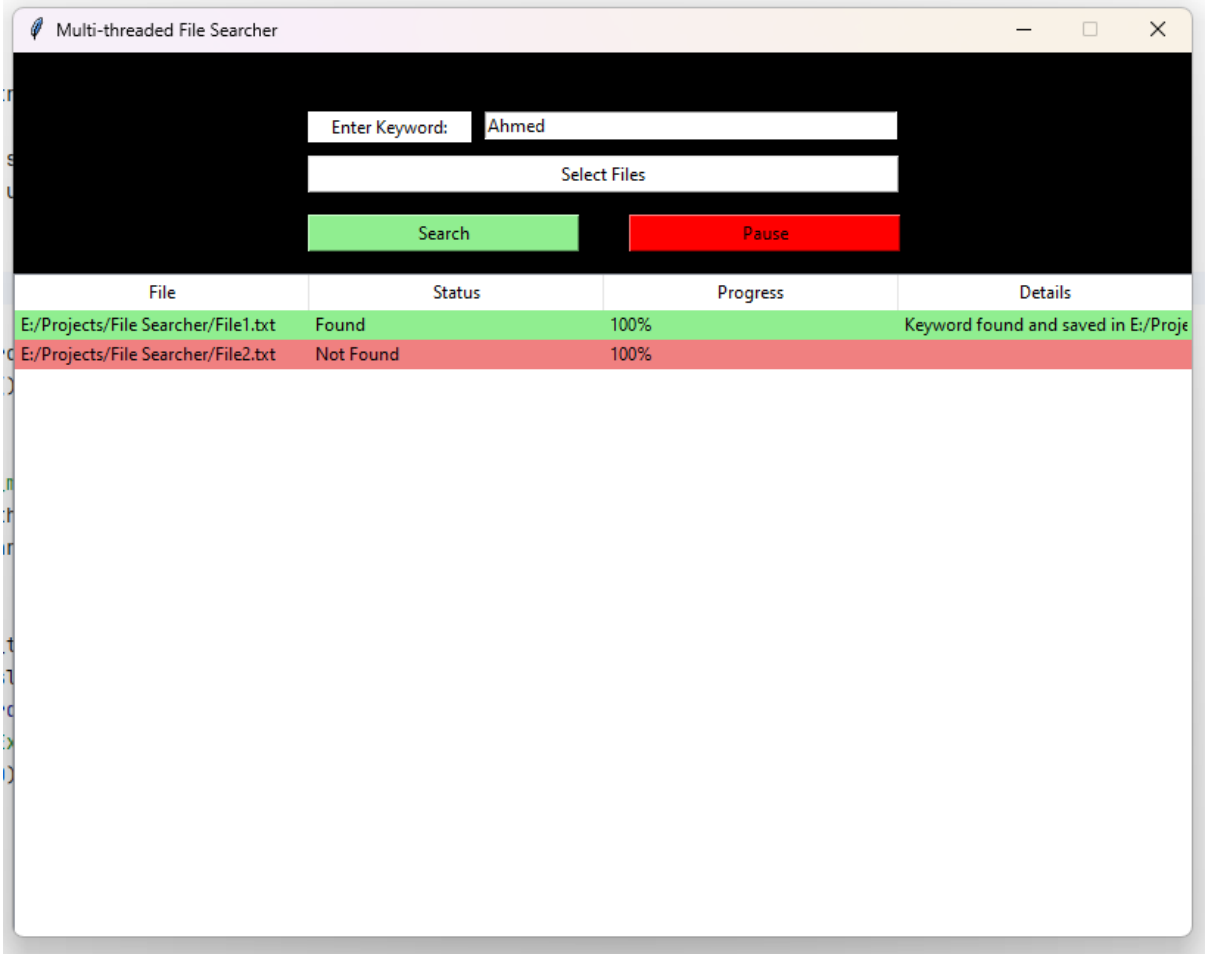
Team Members

- Ahmed Bassem Kamal
- Ahmed Khaled Gomaa
- Ahmed Gamal Abd Elhay
- Ahmed Khaled Nasr
- Ahmed Ashraf Ahmed
- Ahmed Ramadan Rajab

Project Overview

This project implements a multi-threaded file search application that allows users to search for a specific keyword across multiple files. Each file is processed by an individual thread, ensuring efficient and concurrent searching. The graphical user interface (GUI) is implemented using Tkinter and runs on a separate thread to maintain responsiveness. The application provides features such as:

- Real-time progress tracking for each file.
- Pausing and resuming the search operation.
- Opening files directly from the interface.
- Highlighting found keywords in an HTML format.



Main Components

1. FileHandler Class

The FileHandler class encapsulates all file handling and searching logic.

Attributes:

- **files**: List of files selected by the user.
- **threads**: List of threads processing each file.
- **pause_toggle**: Boolean to track pause/resume state.
- **keyword**: Keyword to be searched.

Methods:

- **set_files (**files**)**: Sets the list of files for processing.
- **toggle_pause ()**: Toggles the pause_toggle state.
- **start_search (**keyword, progress_callback**)**: Initializes threads for each file and starts the search.
- **search_file (**index, file, progress_callback**)**: Searches for the keyword in a file and creates an HTML file with highlighted results on the target keyword if found.
- **open_selected_file (**event**)**: Opens the selected file directly from the GUI.

2. FileSearcherApp Class

The FileSearcherApp class manages the GUI and user interactions.

Attributes:

- root: Main Tkinter window.
- file_handler: Instance of FileHandler to manage file operations.
- files: Stores the list of selected files.

GUI Components:

- Frames for organizing the layout.
- Input field for entering the keyword.
- for selecting files, starting the search, and pausing/resuming.
- A tree view for displaying the search progress and results.

Methods:

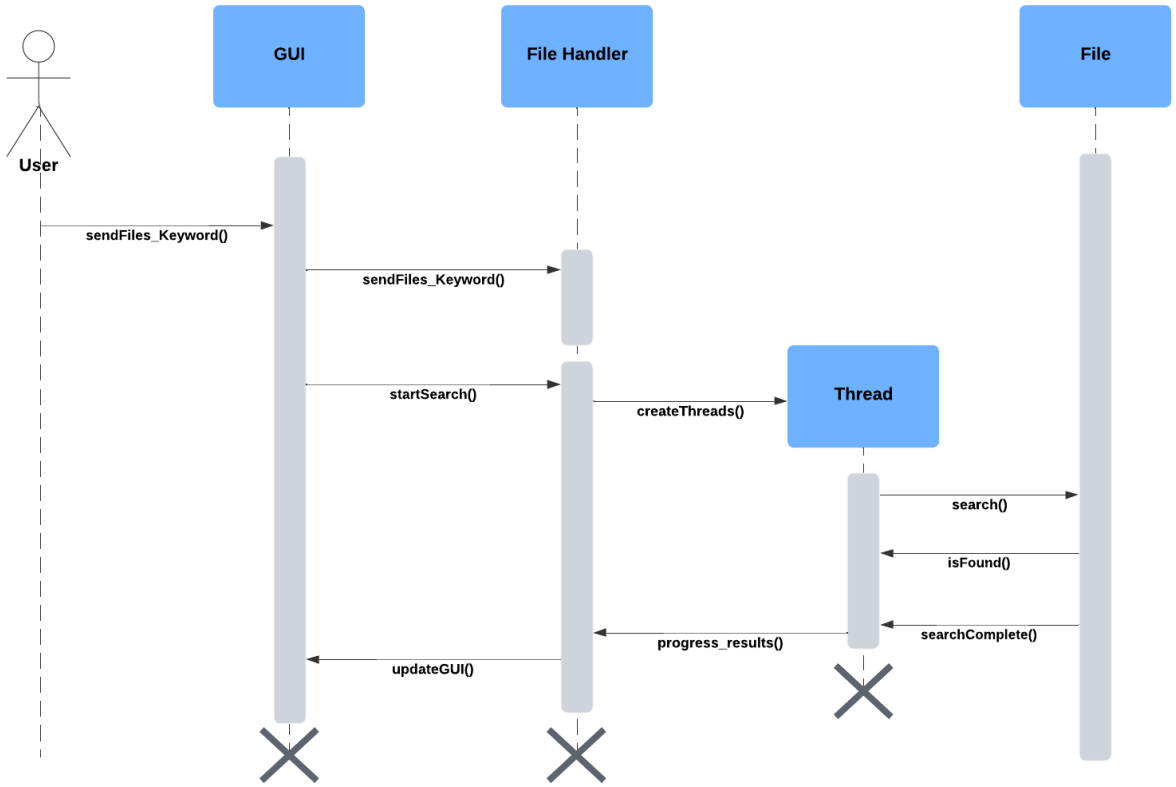
- **init_components ()**: Initializes and places all GUI components.
- **select_files ()**: Opens a file dialog to select files and updates the tree view.
- **start_search ()**: Starts the search process after validating user input.
- **pause_resume ()**: Toggles the search state between paused and resumed.
- **update_progress (**index, status, progress, details, color**)**: Updates the tree view with the progress and status of each file.

3. Main Functionality

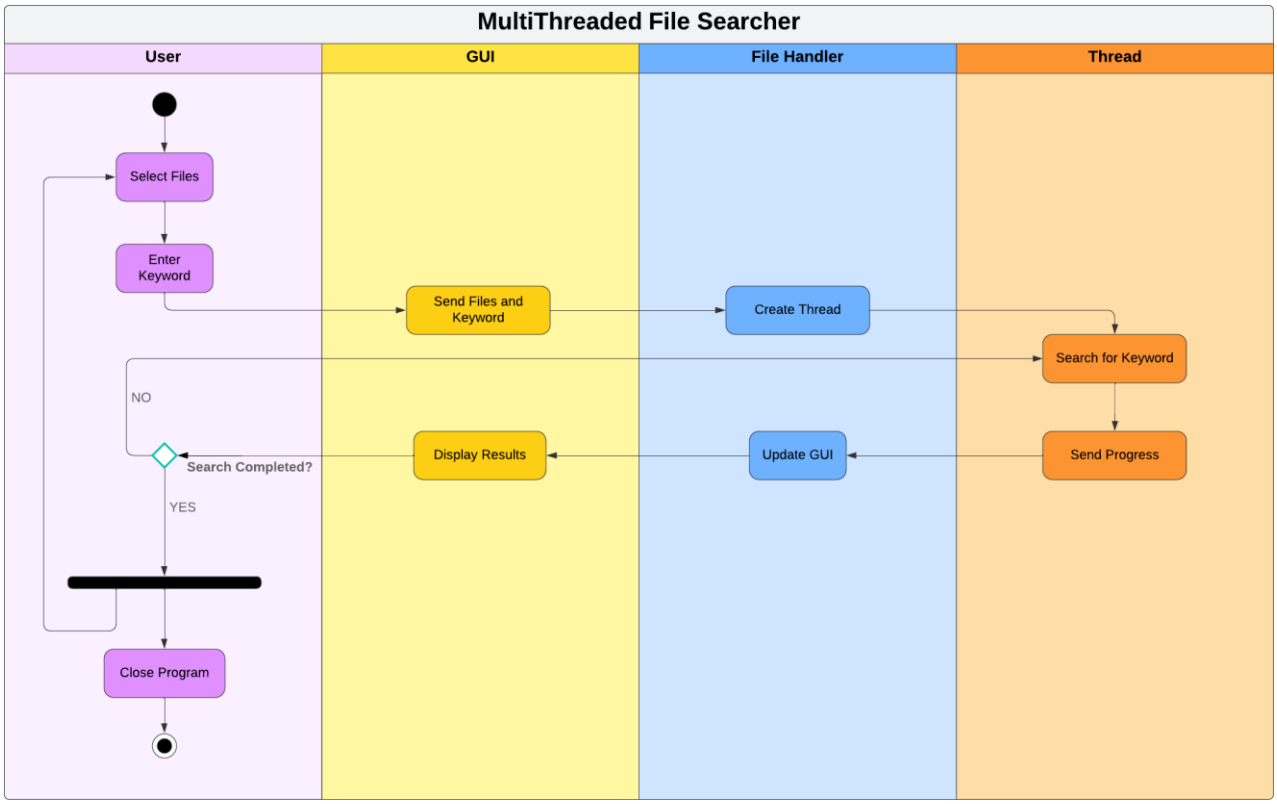
Runs the GUI in a separate thread using threading.Thread.
Continuously monitors the GUI thread to keep the application responsive.

Diagrams

1. Sequence Diagram



2. Activity Diagram



Features

1. **Multi-threaded Search**: Each file is processed in a separate thread, improving performance and ensuring efficient use of system resources.
2. **GUI Responsiveness**: The GUI runs in a separate thread, preventing UI freezing during the file search process.
3. **Keyword Highlighting**: If the keyword is found in a file, the application generates an HTML file with the keyword highlighted for easy identification.
4. **Pause/Resume Functionality**: Users can pause and resume the search process at any time, providing flexibility during long searches.
5. **File Access**: Files can be accessed directly from the interface by double-clicking on the file entry in the tree view.

How It Works

1. **Selecting Files**: Users click the Select Files button to open a file dialog and choose files for processing.
2. **Entering a Keyword**: The keyword is entered into the text field.
3. **Starting the Search**: Clicking the Start Search button initiates the search process. Progress is displayed in the tree view.
4. **Pausing/Resuming**: The Pause button toggles the search state, allowing users to pause and resume the operation.
5. **Viewing Results**: The tree view displays the status and progress for each file. Double-clicking on a file opens it directly. If the keyword is found, an HTML file with highlighted results is created and saved.

Dependencies

Python Standard Library:

- os: For file operations.
- threading: For multi-threading.
- time: For managing pauses and delays.
- tkinter: For GUI implementation.

No External Libraries.

Code Structure

1. **FileHandler Class**: Handles all core operations related to file searching and highlighting.
2. **FileSearcherApp Class**: Defines the GUI layout and manages user interactions.
3. **Main Execution Block**: Launches the GUI in a separate thread and manages the application's lifecycle.

Usage Instructions

1. Run the script: python filename.py
2. Select files using the Select Files button.
3. Enter a keyword in the input field.
4. Click Start Search to begin.
5. Use Pause/Resume to control the search process.
6. View results in the tree view and double-click to open files.
7. Check the generated HTML files for highlighted keywords if found.

Conclusion

The Multi-threaded File Searcher is a robust and user-friendly tool designed to simplify keyword searching across multiple files. With its multi-threaded approach and responsive GUI, it provides a seamless experience for users managing large-scale text file searches.