**NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY**

**MILITARY COLLEGE OF SIGNALS**

****

**INFORMATION RETRIEVAL**

**(CS-424)**

**ASSIGNMENT # 01**

**Submitted by:** **MUHAMMAD AHMAD SULTAN**

**CMS ID**: **408709**

**R**ANK: **NC**

**C**OURSE: **BESE-28**

**S**ECTION: **C**

**Submitted to:** **DR. NAUMAN ALI KHAN**

***D****ated:*10-02-2025

**Assignment # 01**

Boolean Search Implementation

Efficient Boolean Search System: Implementation and GUI Development

## Introduction

Boolean search is a powerful technique used in information retrieval to refine and enhance search results using logical operators such as AND, OR, and NOT. This report documents the implementation of a Boolean search system using Python. The system tokenizes text, removes stopwords, creates an inverted index, and allows users to perform Boolean searches. Additionally, a GUI is developed to facilitate user interaction with the search functionality.

## Objectives

* To implement a Boolean search system using Python.
* To process textual data through tokenization and stopword removal.
* To construct an inverted index for efficient retrieval.
* To implement Boolean search operations (AND, OR, NOT).
* To develop a user-friendly GUI for conducting searches.
* To provide a structured and documented codebase.

## GitHub Repository

The complete source code for this project is available on GitHub: [GitHub Repository - BoolPySearch](https://github.com/ahmadsultan03/bool-py-Search)

## Methodology

The implementation follows a systematic approach:

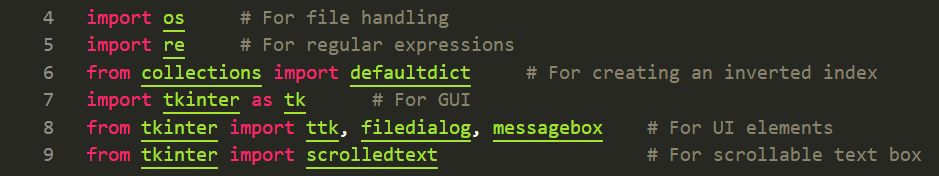
1. **Data Preprocessing**: Tokenization, stopword removal, and text normalization.
2. **Indexing**: Creating an inverted index for efficient search operations.
3. **Boolean Search Logic**: Implementing AND, OR, and NOT operations.
4. **GUI Development**: Building a user-friendly interface with Tkinter.
5. **Testing & Optimization**: Ensuring accurate and efficient search performance.

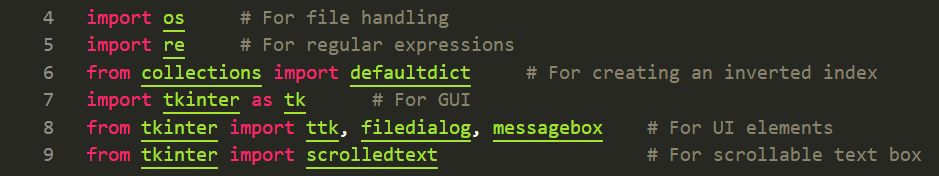
## Code Explanation

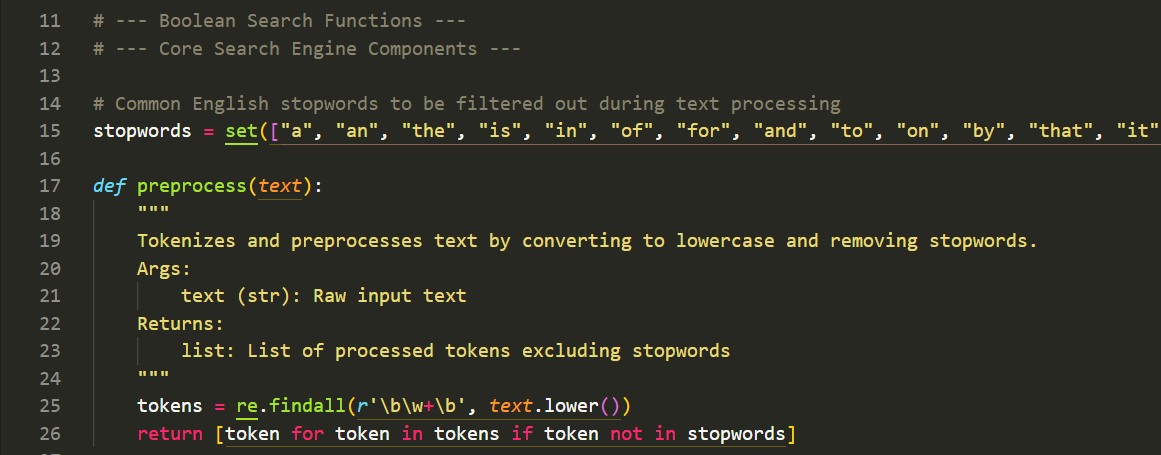
The implementation consists of multiple components, including data preprocessing, indexing, search functions, and a GUI. Below is a detailed explanation of each module with code snippets.

### Data Preprocessing

1. **Tokenization:** Text is split into individual words (tokens).
2. **Stopword Removal:** Common words (e.g., "is", "the", "and") are filtered out to improve search efficiency.
3. **Lowercasing:** Ensures case-insensitive matching.

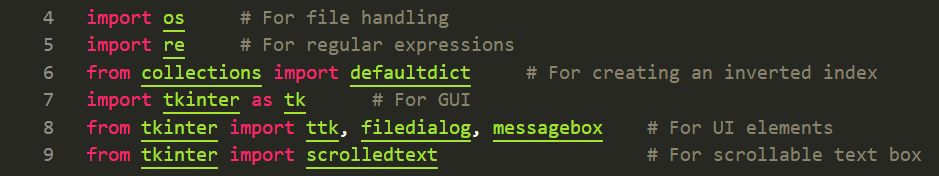


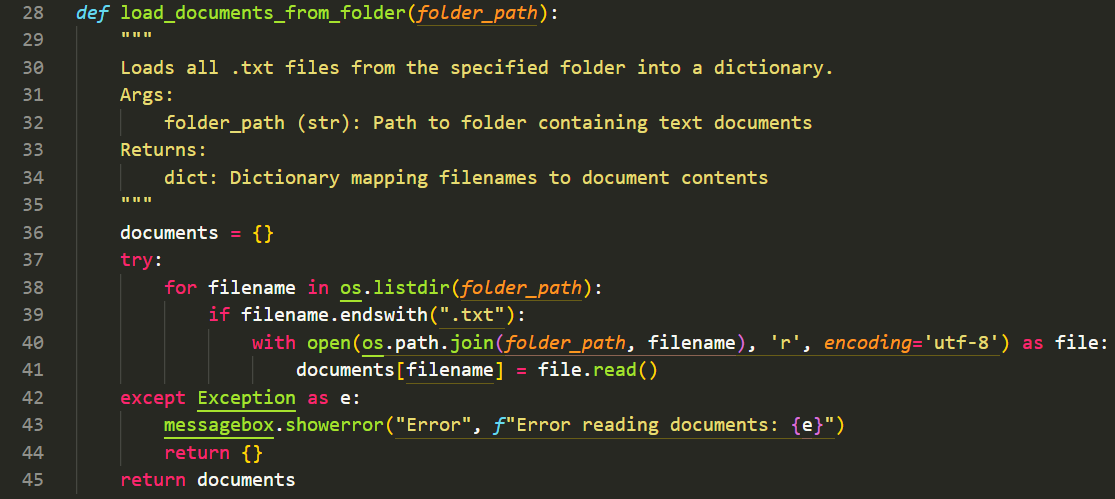




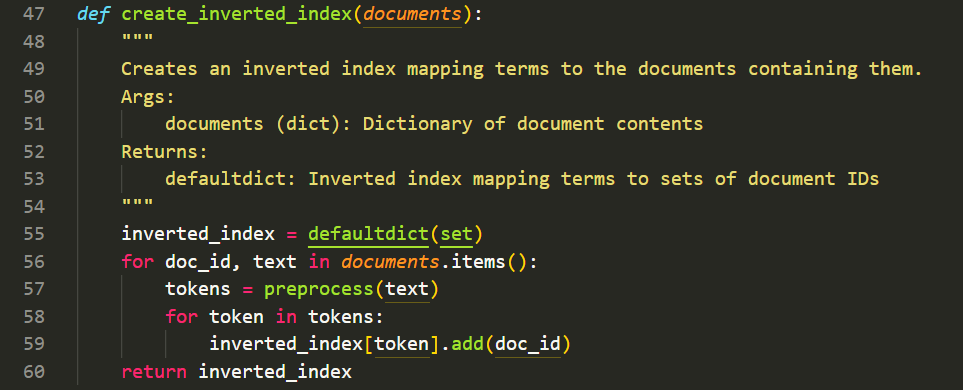
### Indexing

**Loading Documents:** Text files are loaded from a specified directory.



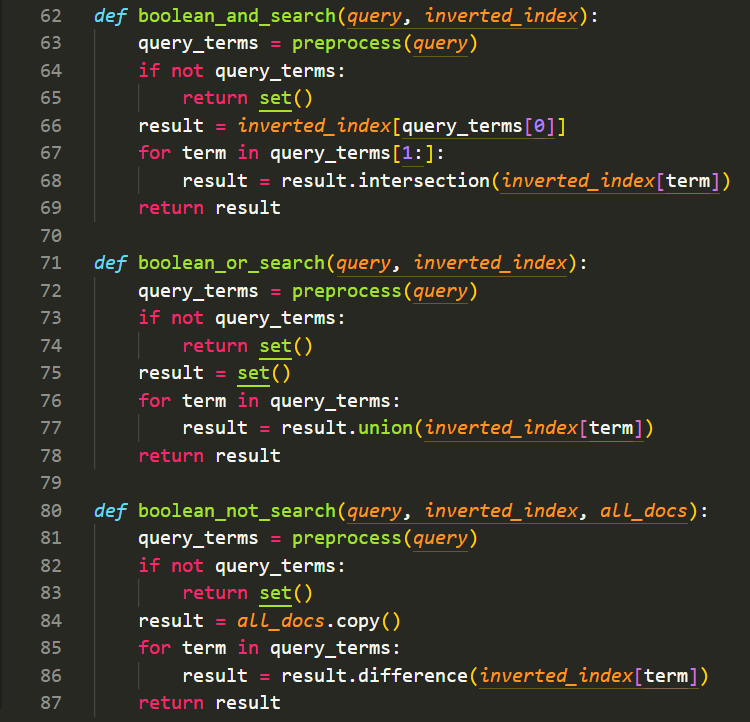
****

**Creating an Inverted Index:** A dictionary is built where each word maps to a set of document IDs containing that word.

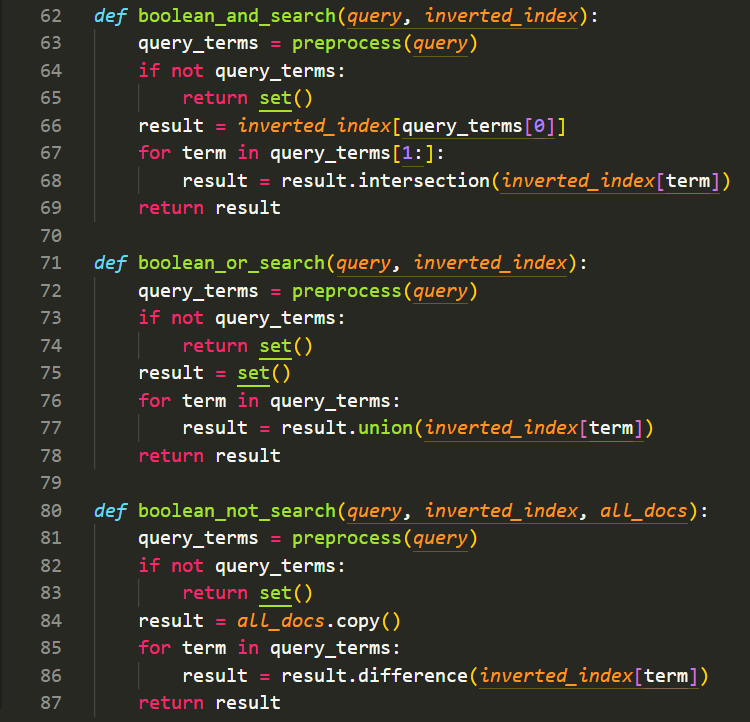


### Boolean Search Implementation

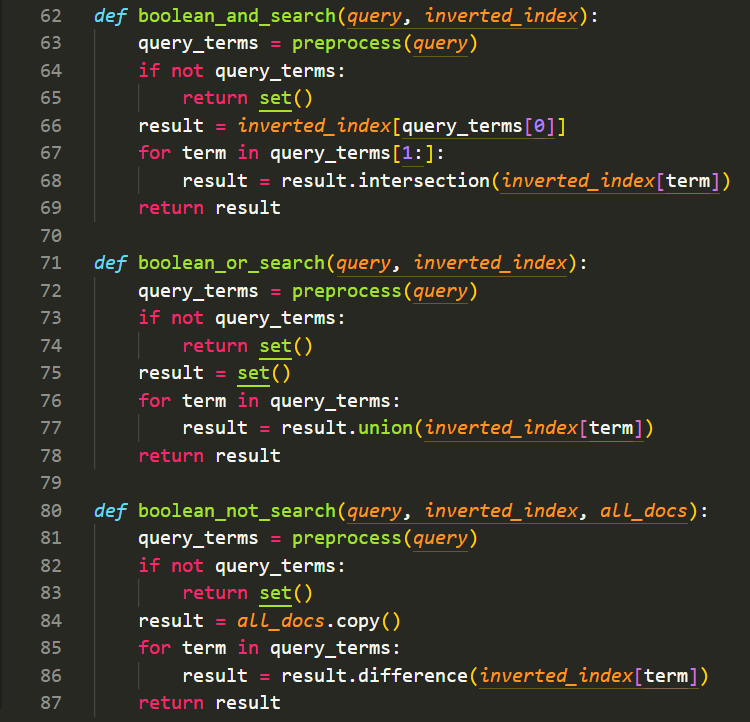
1. **AND Search:** Retrieves documents containing all query terms.



1. **OR Search:** Retrieves documents containing at least one of the query terms.



1. **NOT Search:** Excludes documents containing the query terms.

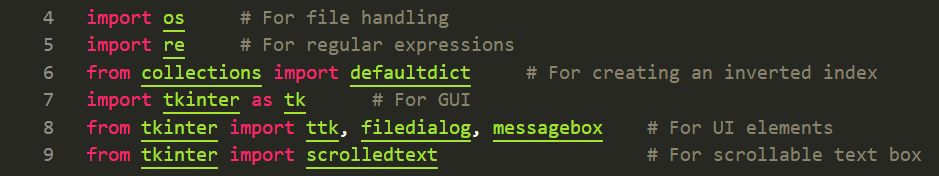


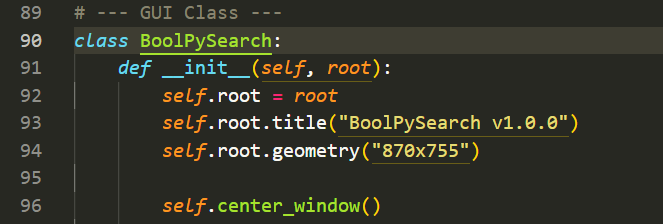
### GUI Implementation

A GUI is developed using Tkinter for enhanced usability. Key features include:

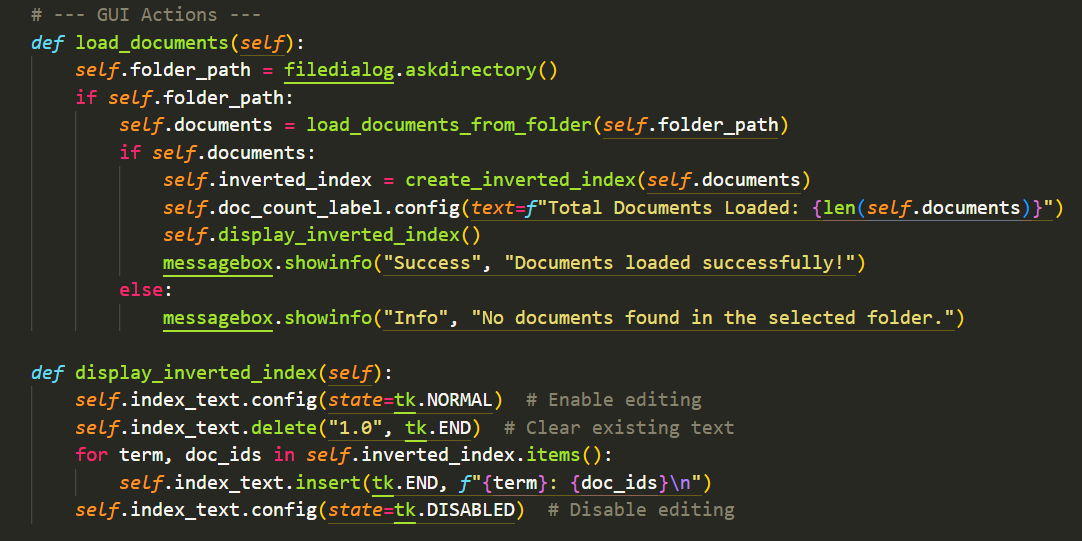
* **Load Documents:** Allows users to select a folder containing text files.
* **Search Box:** Users enter queries and select a Boolean search type.
* **Results Display:** Shows the retrieved documents and their contents.
* **Styling:** Uses a dark theme with enhanced UI elements for readability.

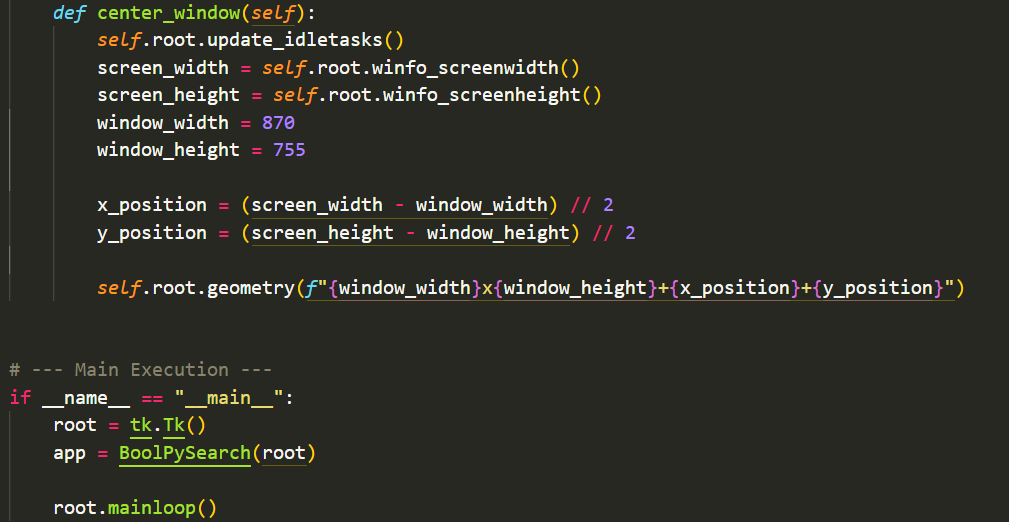
#### GUI Code Snippet







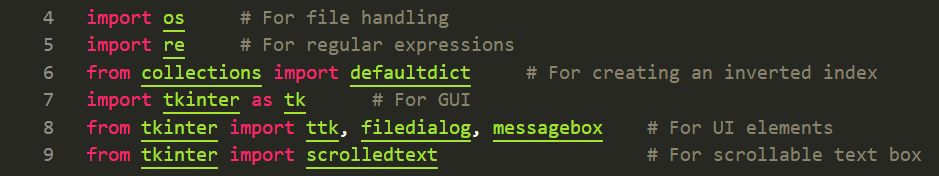




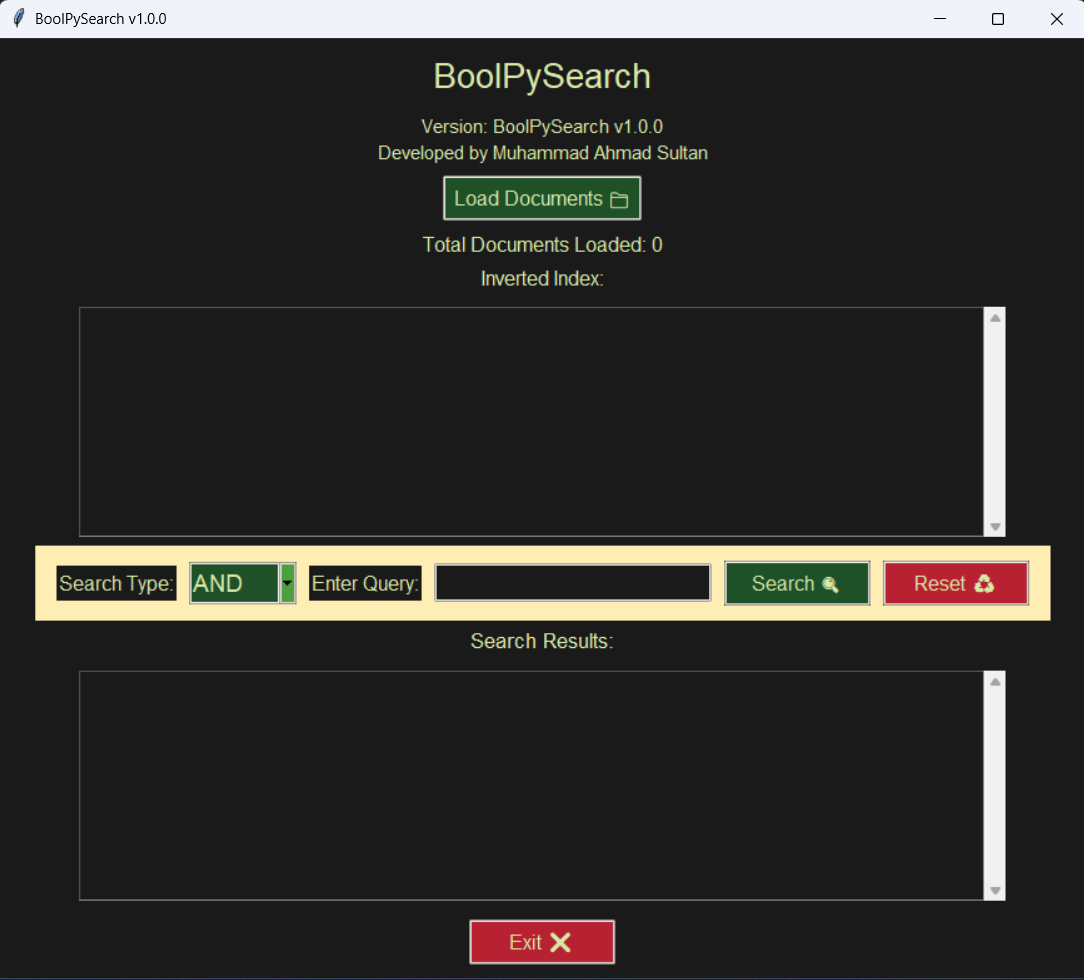
## Tools and Technologies Used

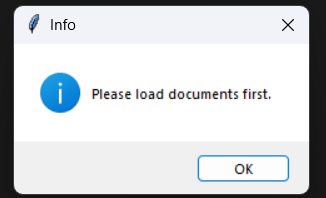
* **Python**: Core programming language.
* **Tkinter**: GUI development.
* **Regular Expressions (re module)**: Tokenization and text processing.
* **Collections (defaultdict)**: Efficiently managing the inverted index.

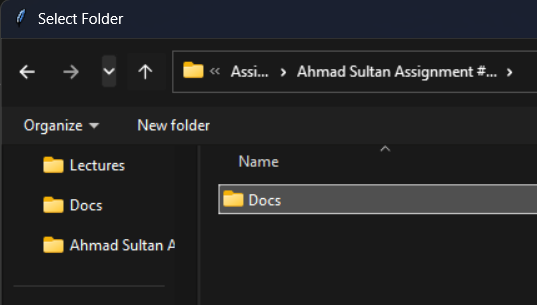


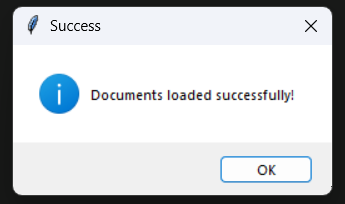


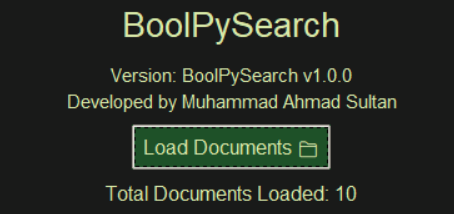
## Output & Display:

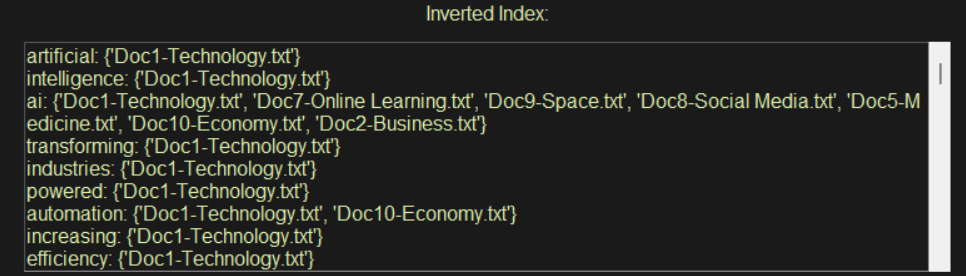


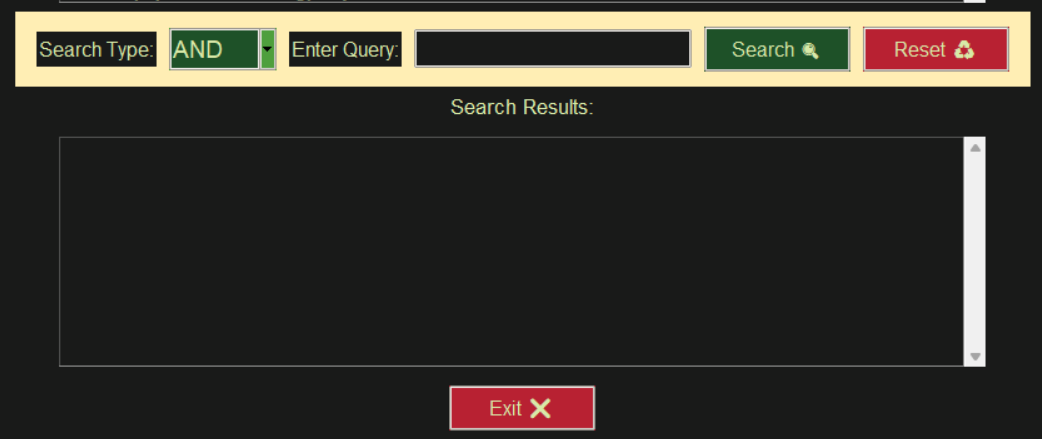


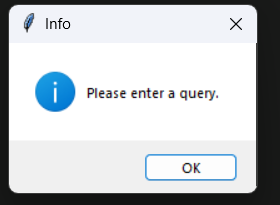


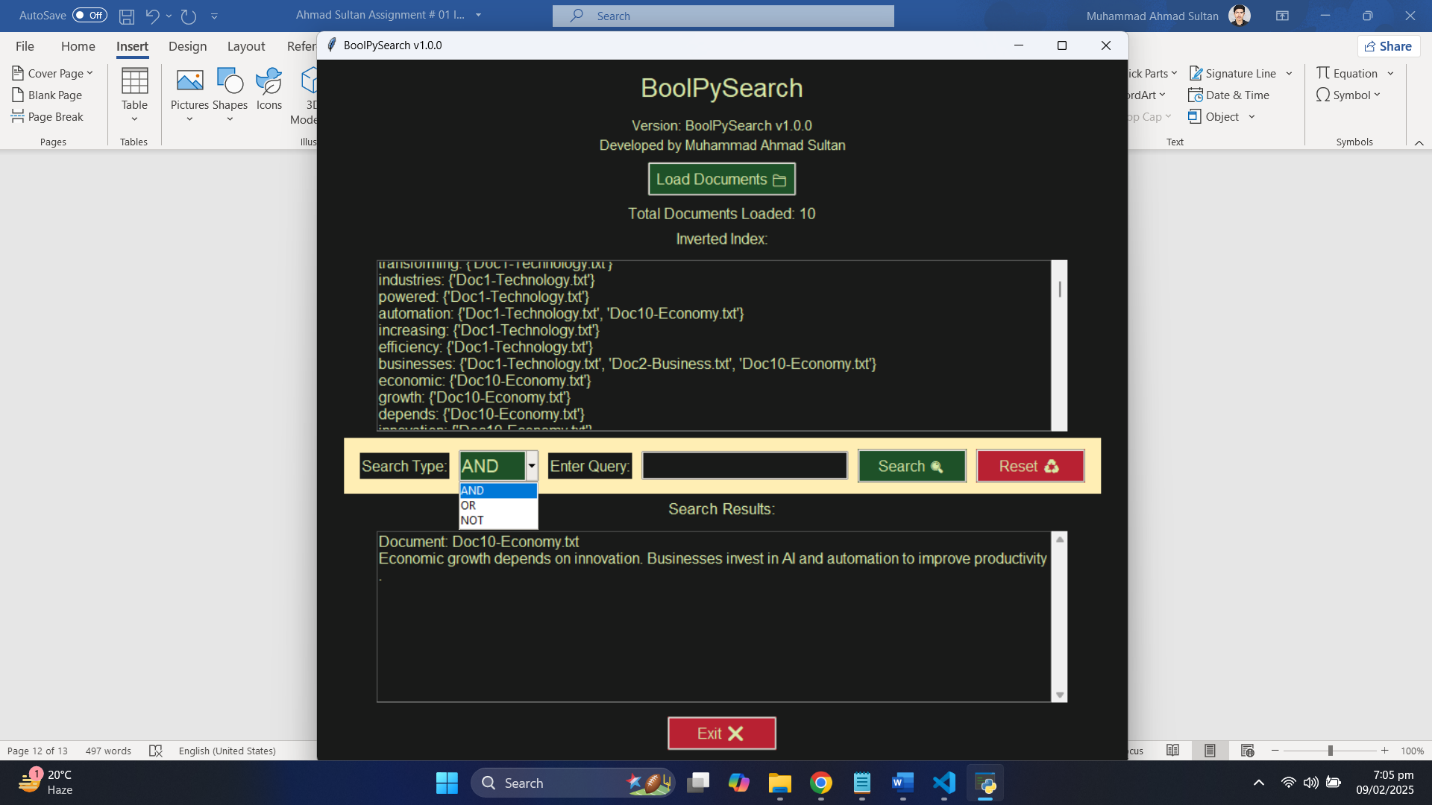


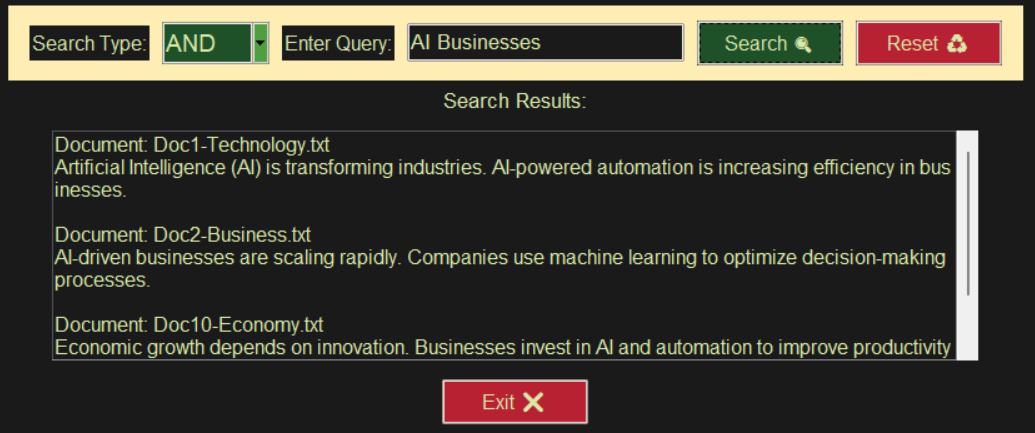


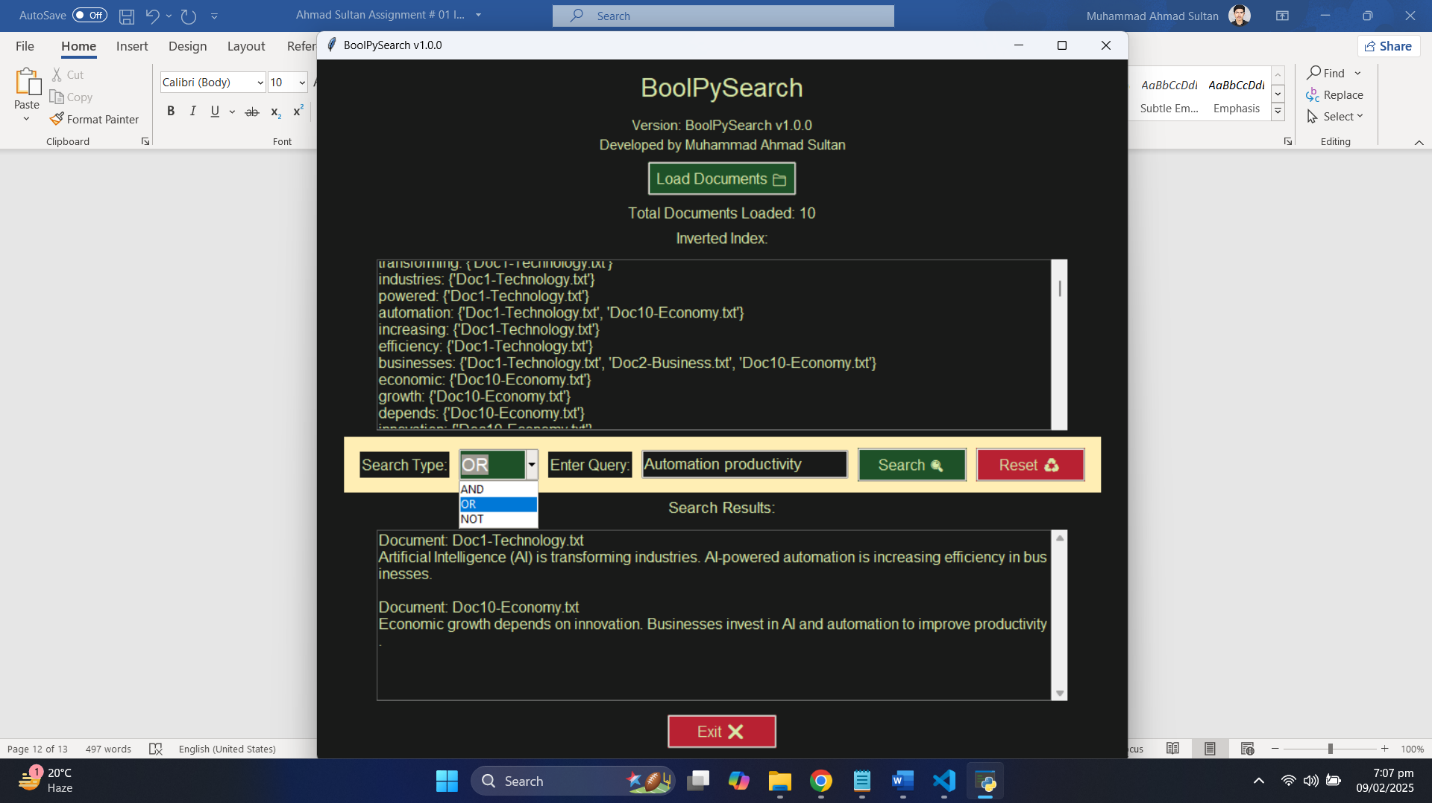




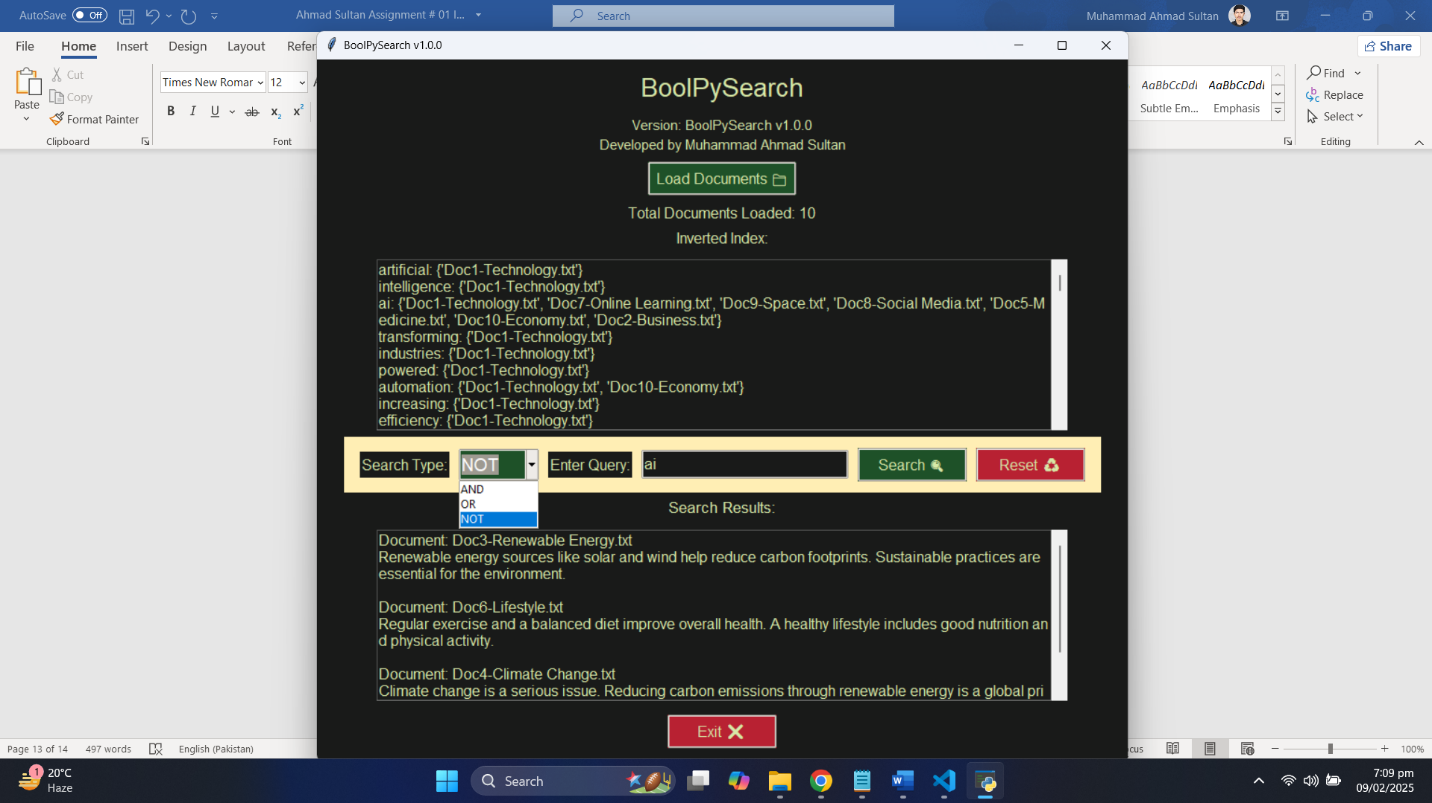


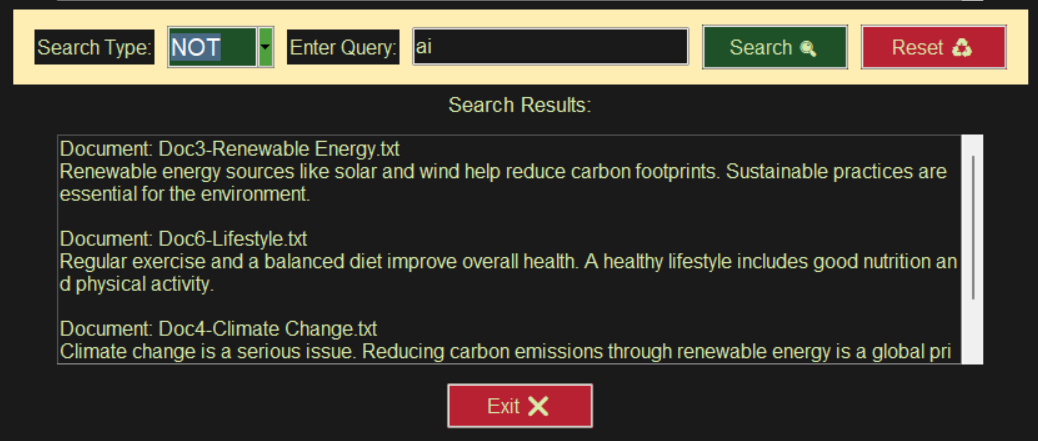


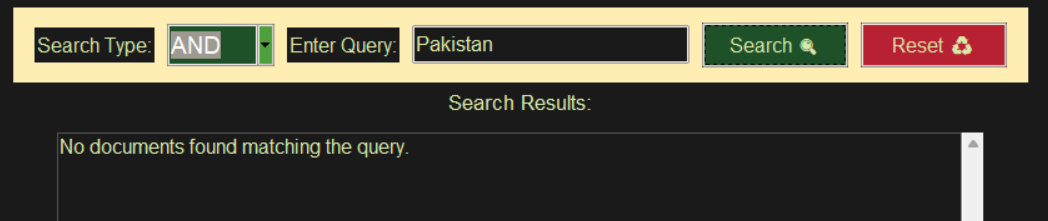


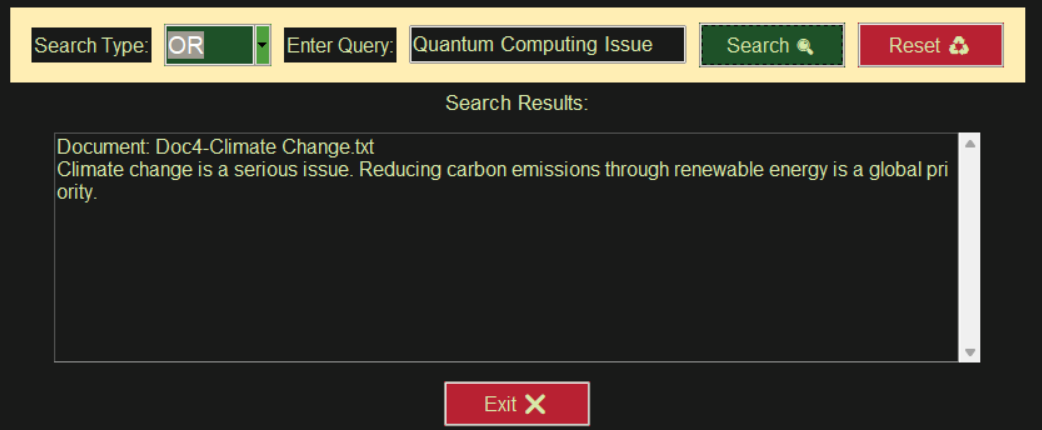




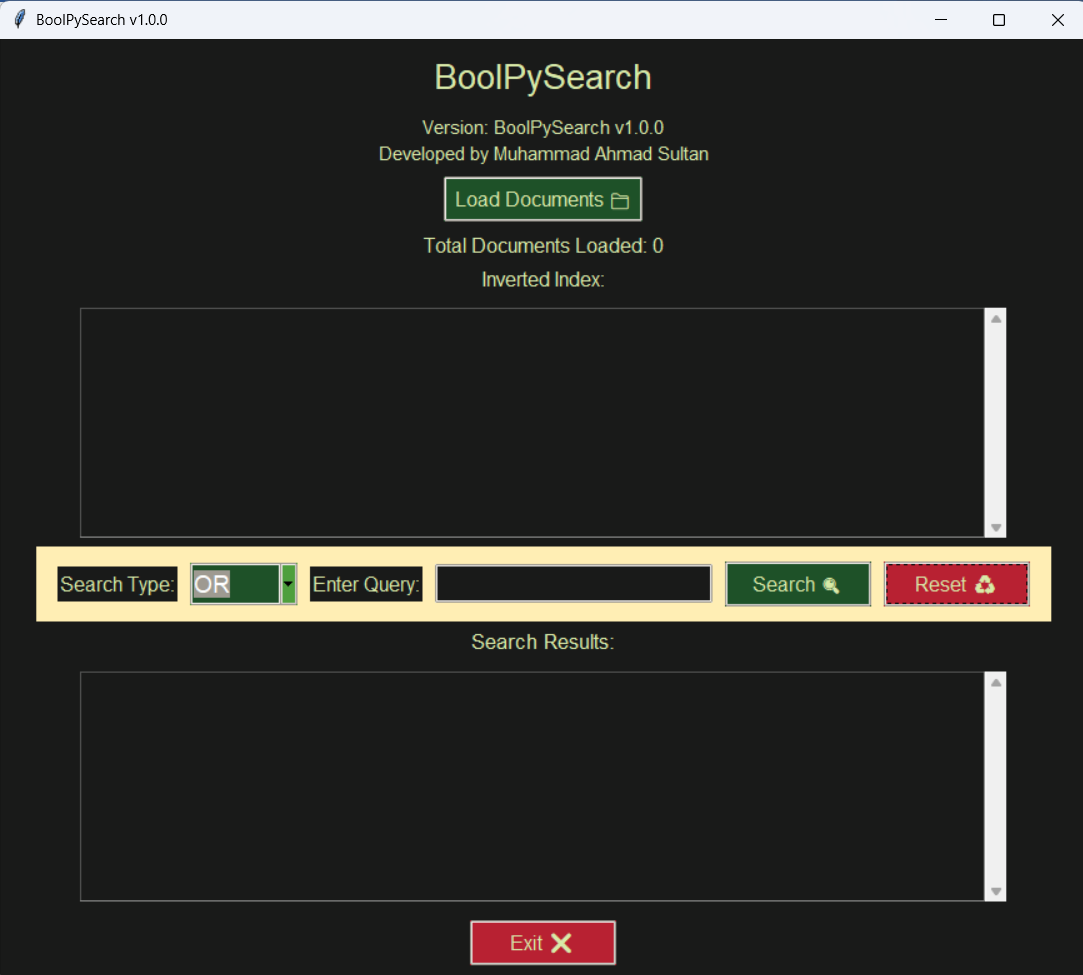












## Conclusion

In a nutshell, this project successfully implements a Boolean search system with an interactive GUI. It efficiently processes text, creates an inverted index, and retrieves documents based on Boolean logic. The combination of structured indexing and user-friendly interaction makes it a useful tool for text-based information retrieval.

**THE END**