NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY

MILITARY COLLEGE OF SIGNALS

INFORMATION RETRIEVAL

(CS-424)

ASSIGNMENT # 01

Submitted by:	**MUHAMMAD AHMAD SULTAN**
CMS ID:	**408709**
RANK:	**NC**
COURSE:	**BESE-28**
SECTION:	**C**

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

Dated: **10-02-2025**

Assignment # 01

BoolPySearch

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

I. To implement a Boolean search system using Python.

II. To process textual data through tokenization and stopword removal.

III. To construct an inverted index for efficient retrieval.

IV. To implement Boolean search operations (AND, OR, NOT).

V. To develop a user-friendly GUI for conducting searches.

VI. To provide a structured and documented codebase.

▪ GitHub Repository

The complete source code for this project is available on GitHub: GitHub Repository - BoolPySearch

▪ Methodology

The implementation follows a systematic approach:

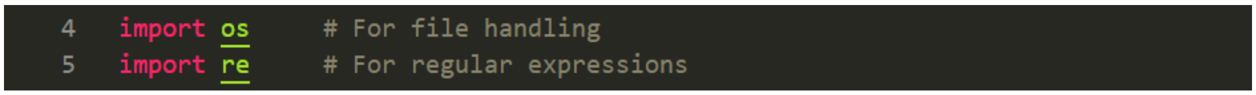
I. **Data Preprocessing**: Tokenization, stopword removal, and text normalization.
II. **Indexing**: Creating an inverted index for efficient search operations.
III. **Boolean Search Logic**: Implementing AND, OR, and NOT operations.
IV. **GUI Development**: Building a user-friendly interface with Tkinter.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

▪ Code Explanation

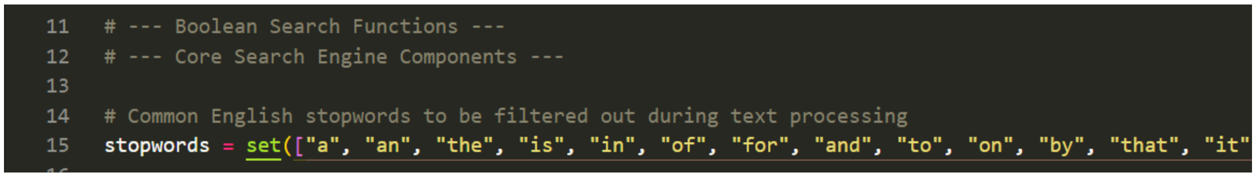
The implementation consists of multiple components, including data preprocessing, indexing, search functions, and a GUI. Below is a detailed explanation of each module with code snippets.

Data Preprocessing

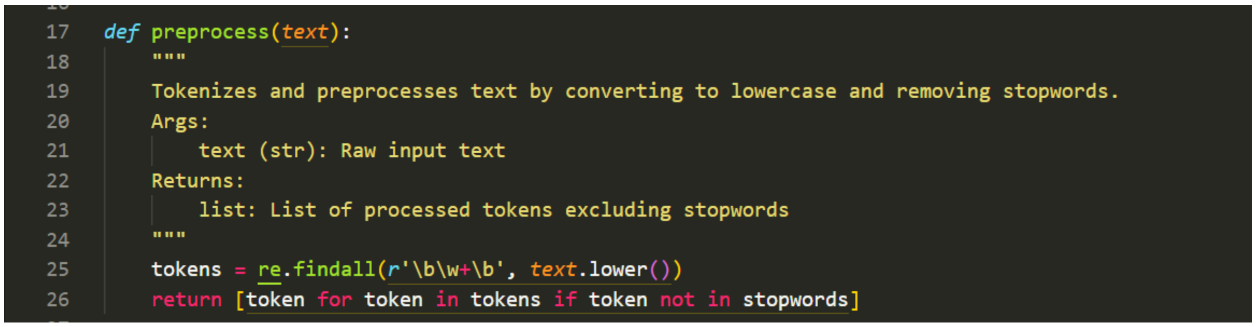
I. **Tokenization:** Text is split into individual words (tokens).
II. **Stopword Removal:** Common words (e.g., "is", "the", "and") are filtered out to improve search efficiency.
III. **Lowercasing:** Ensures case-insensitive matching.



```python
import os      # For file handling
import re      # For regular expressions
```

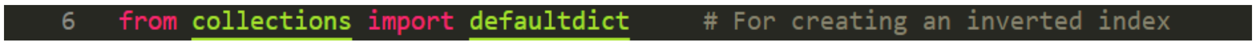


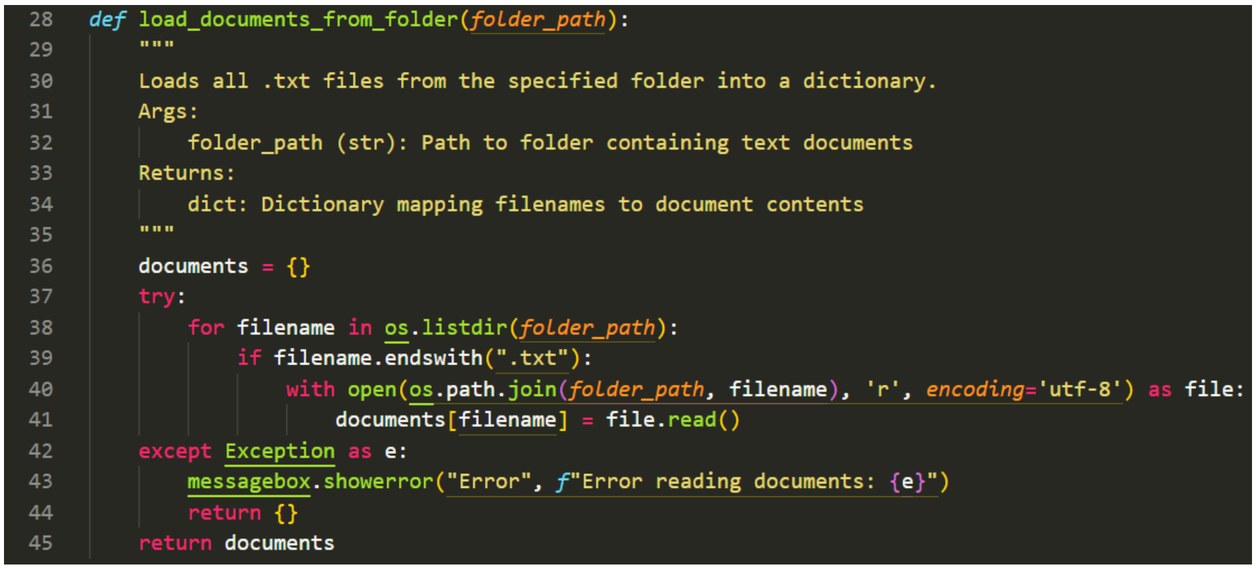
```python
# --- Boolean Search Functions ---
# --- Core Search Engine Components ---

# Common English stopwords to be filtered out during text processing
stopwords = set(["a", "an", "the", "is", "in", "of", "for", "and", "to", "on", "by", "that", "it"
```


```python
def preprocess(text):
    """
    Tokenizes and preprocesses text by converting to lowercase and removing stopwords.
    Args:
        text (str): Raw input text
    Returns:
        list: List of processed tokens excluding stopwords
    """
    tokens = re.findall(r'\b\w+\b', text.lower())
    return [token for token in tokens if token not in stopwords]
```

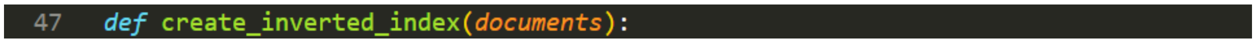
Indexing

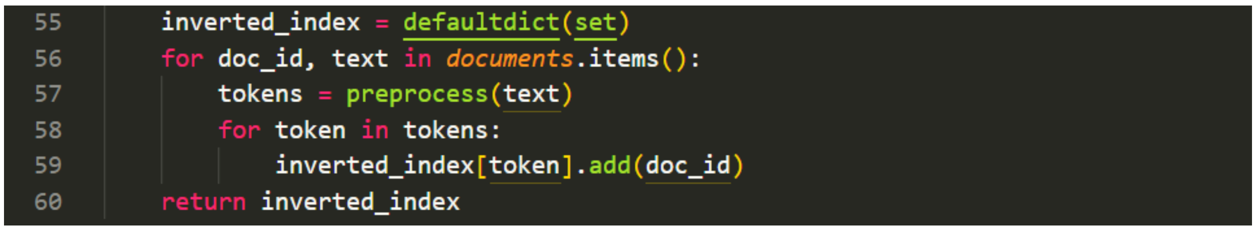
I. **Loading Documents:** Text files are loaded from a specified directory.


```python
from collections import defaultdict      # For creating an inverted index
```


```python
def load_documents_from_folder(folder_path):
    """
    Loads all .txt files from the specified folder into a dictionary.
    Args:
        folder_path (str): Path to folder containing text documents
    Returns:
        dict: Dictionary mapping filenames to document contents
    """
    documents = {}
    try:
        for filename in os.listdir(folder_path):
            if filename.endswith(".txt"):
                with open(os.path.join(folder_path, filename), 'r', encoding='utf-8') as file:
                    documents[filename] = file.read()
    except Exception as e:
        messagebox.showerror("Error", f"Error reading documents: {e}")
        return {}
    return documents
```

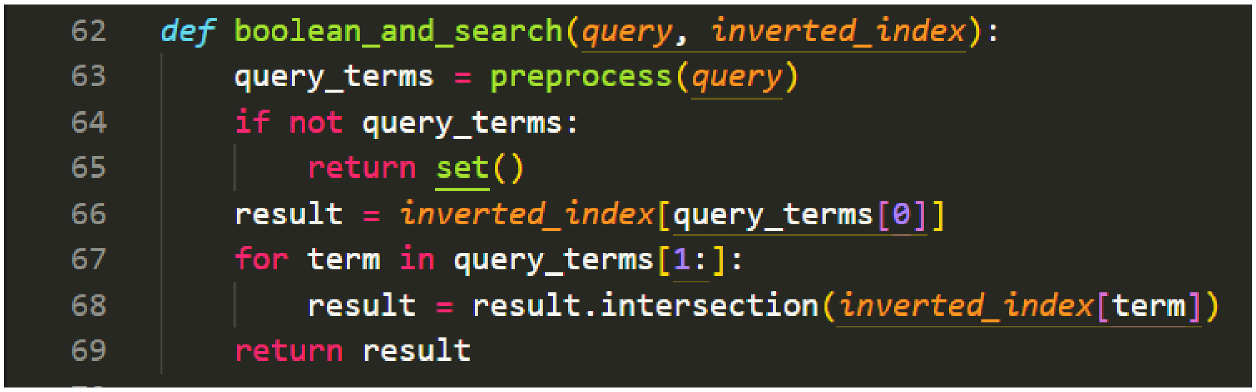
II. **Creating an Inverted Index:** A dictionary is built where each word maps to a set of document IDs containing that word.


```python
def create_inverted_index(documents):
```


```python
    inverted_index = defaultdict(set)
    for doc_id, text in documents.items():
        tokens = preprocess(text)
        for token in tokens:
            inverted_index[token].add(doc_id)
    return inverted_index
```

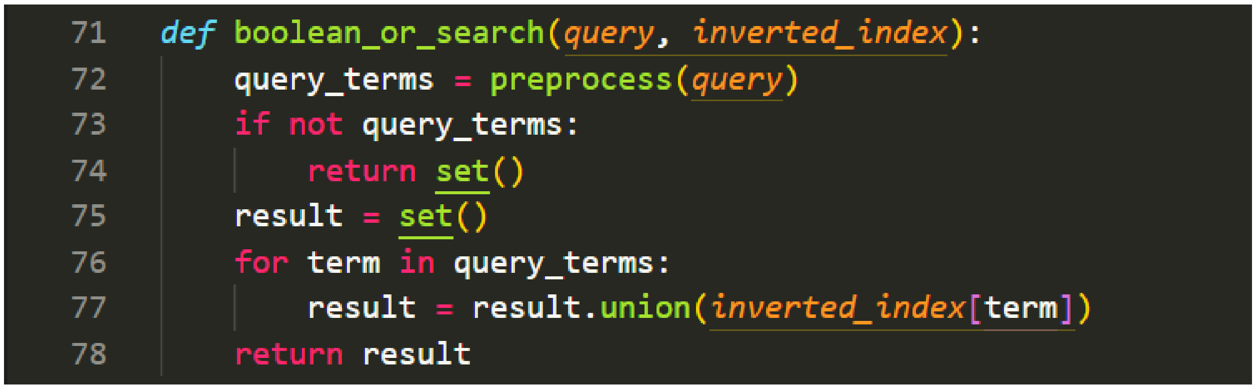
Boolean Search Implementation

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



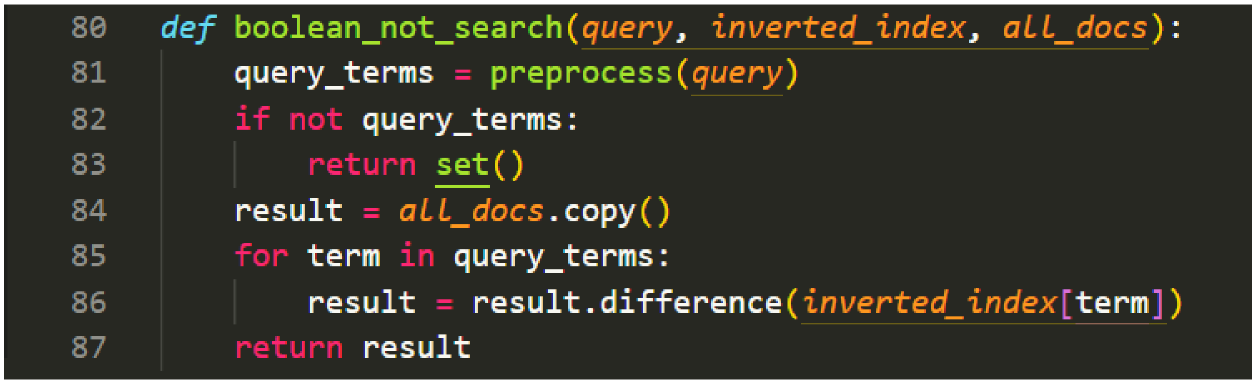
```python
def boolean_and_search(query, inverted_index):
    query_terms = preprocess(query)
    if not query_terms:
        return set()
    result = inverted_index[query_terms[0]]
    for term in query_terms[1:]:
        result = result.intersection(inverted_index[term])
    return result
```

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



```python
def boolean_or_search(query, inverted_index):
    query_terms = preprocess(query)
    if not query_terms:
        return set()
    result = set()
    for term in query_terms:
        result = result.union(inverted_index[term])
    return result
```

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



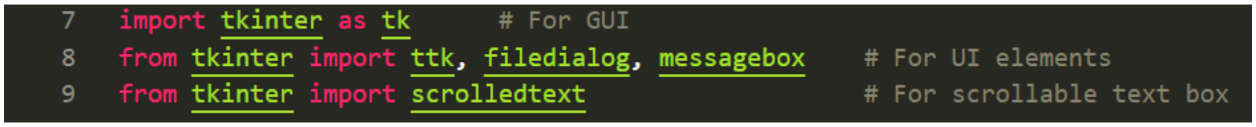
```python
def boolean_not_search(query, inverted_index, all_docs):
    query_terms = preprocess(query)
    if not query_terms:
        return set()
    result = all_docs.copy()
    for term in query_terms:
        result = result.difference(inverted_index[term])
    return result
```

GUI Implementation

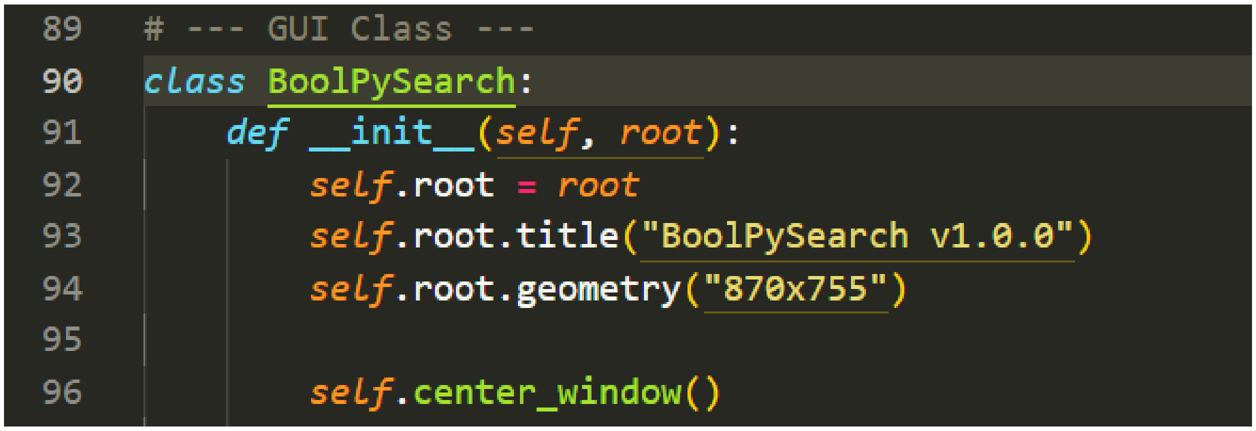
A GUI is developed using Tkinter for enhanced usability. Key features include:

- **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.

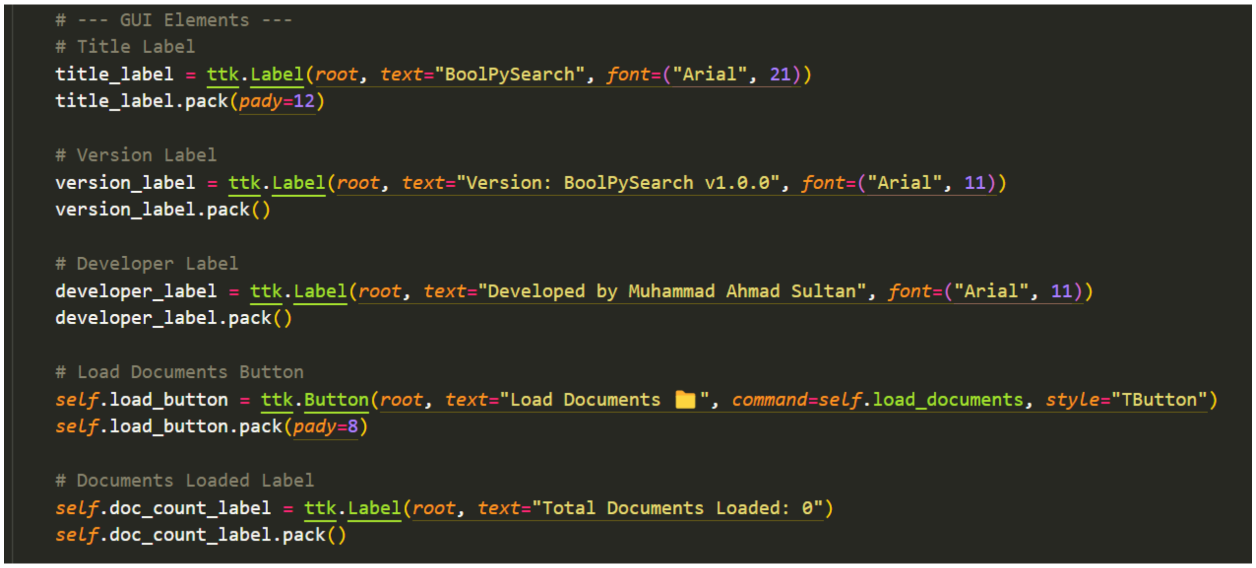
GUI Code



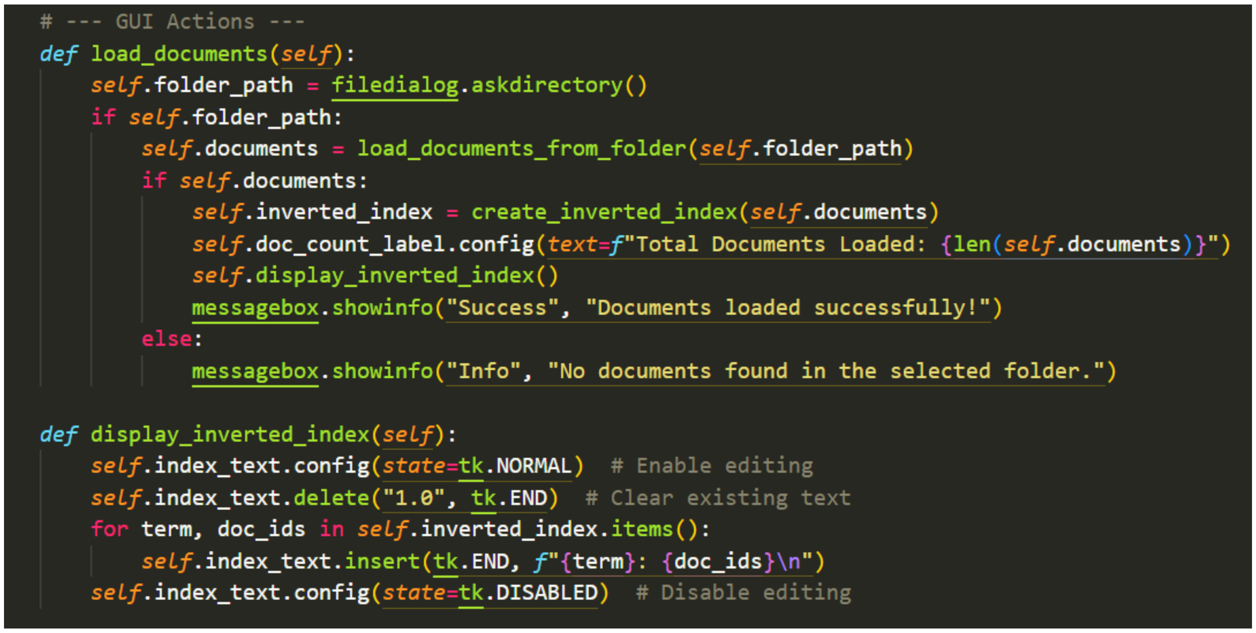
```python
import tkinter as tk        # For GUI
from tkinter import ttk, filedialog, messagebox     # For UI elements
from tkinter import scrolledtext                    # For scrollable text box
```



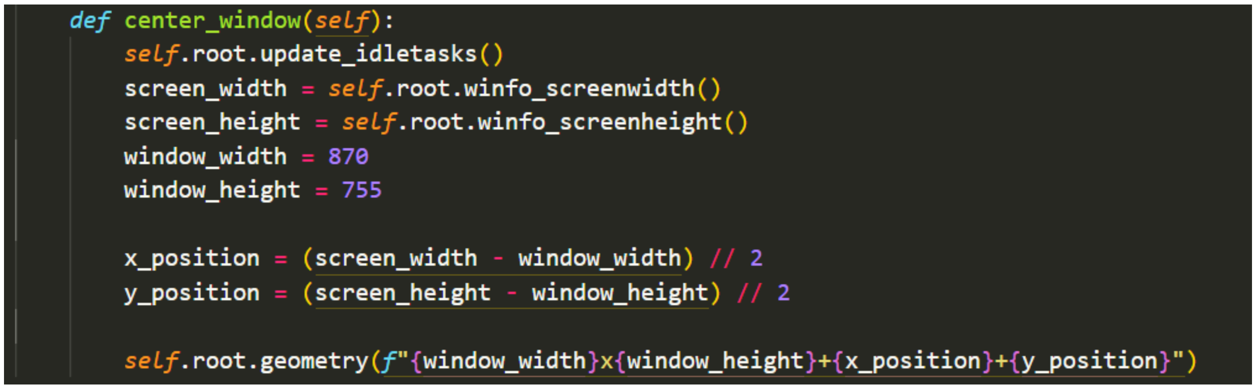
```python
# --- GUI Class ---
class BoolPySearch:
    def __init__(self, root):
        self.root = root
        self.root.title("BoolPySearch v1.0.0")
        self.root.geometry("870x755")

        self.center_window()
```



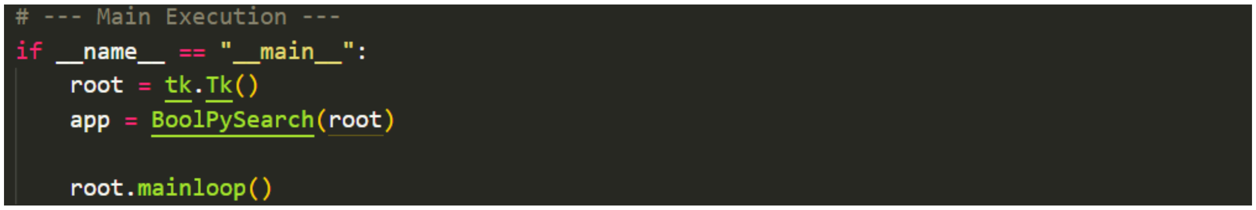
```python
# --- GUI Elements ---
# Title Label
title_label = ttk.Label(root, text="BoolPySearch", font=("Arial", 21))
title_label.pack(pady=12)

# Version Label
version_label = ttk.Label(root, text="Version: BoolPySearch v1.0.0", font=("Arial", 11))
version_label.pack()

# Developer Label
developer_label = ttk.Label(root, text="Developed by Muhammad Ahmad Sultan", font=("Arial", 11))
developer_label.pack()

# Load Documents Button
self.load_button = ttk.Button(root, text="Load Documents 📁", command=self.load_documents, style="TButton")
self.load_button.pack(pady=8)

# Documents Loaded Label
self.doc_count_label = ttk.Label(root, text="Total Documents Loaded: 0")
self.doc_count_label.pack()
```



```python
# --- GUI Actions ---
def load_documents(self):
    self.folder_path = filedialog.askdirectory()
    if self.folder_path:
        self.documents = load_documents_from_folder(self.folder_path)
        if self.documents:
            self.inverted_index = create_inverted_index(self.documents)
            self.doc_count_label.config(text=f"Total Documents Loaded: {len(self.documents)}")
            self.display_inverted_index()
            messagebox.showinfo("Success", "Documents loaded successfully!")
        else:
            messagebox.showinfo("Info", "No documents found in the selected folder.")

def display_inverted_index(self):
    self.index_text.config(state=tk.NORMAL)  # Enable editing
    self.index_text.delete("1.0", tk.END)  # Clear existing text
    for term, doc_ids in self.inverted_index.items():
        self.index_text.insert(tk.END, f"{term}: {doc_ids}\n")
    self.index_text.config(state=tk.DISABLED)  # Disable editing
```



```python
    def center_window(self):
        self.root.update_idletasks()
        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()
        window_width = 870
        window_height = 755

        x_position = (screen_width - window_width) // 2
        y_position = (screen_height - window_height) // 2

        self.root.geometry(f"{window_width}x{window_height}+{x_position}+{y_position}")
```



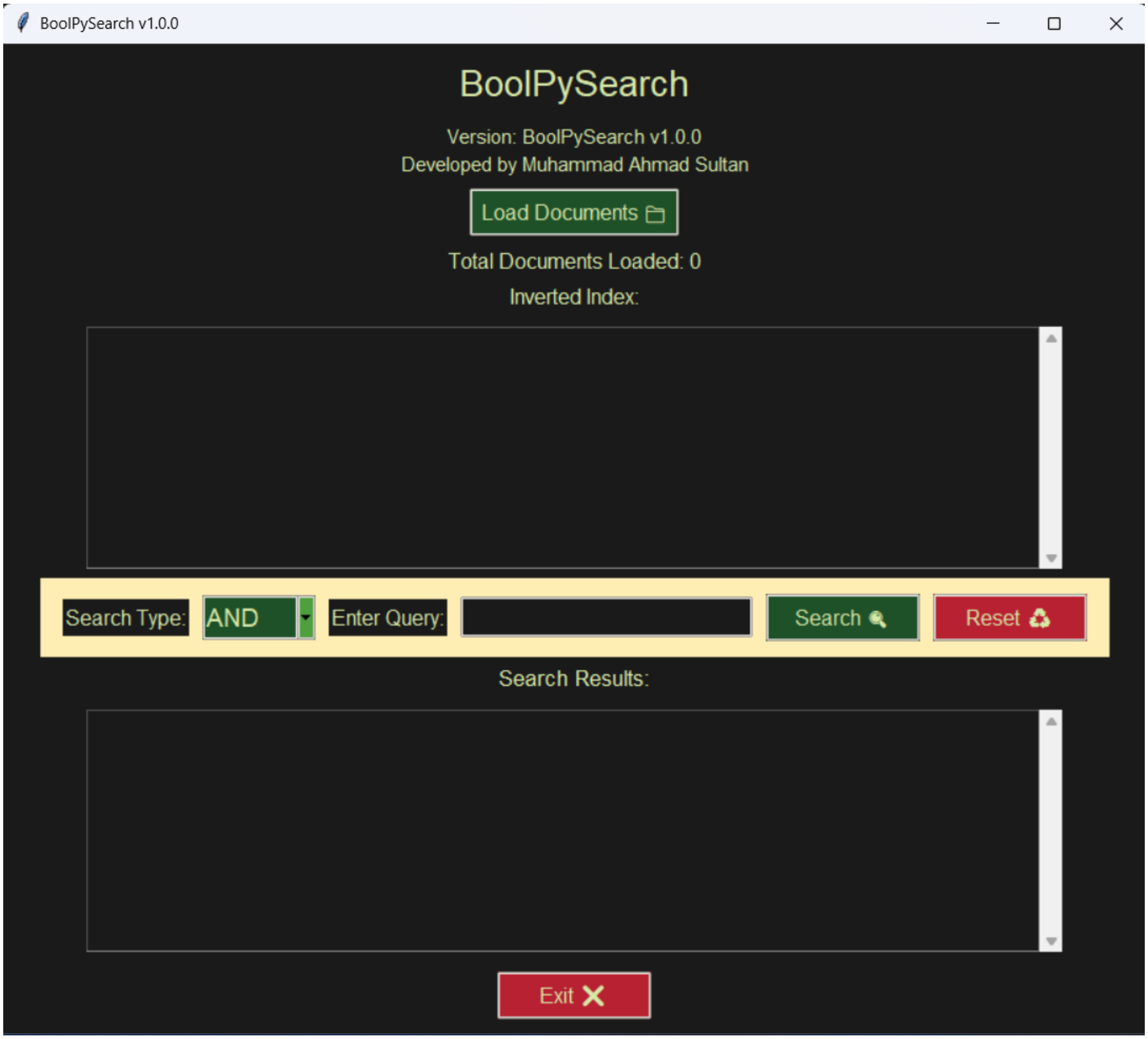
```python
# --- Main Execution ---
if __name__ == "__main__":
    root = tk.Tk()
    app = BoolPySearch(root)

    root.mainloop()
```

▪ 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.

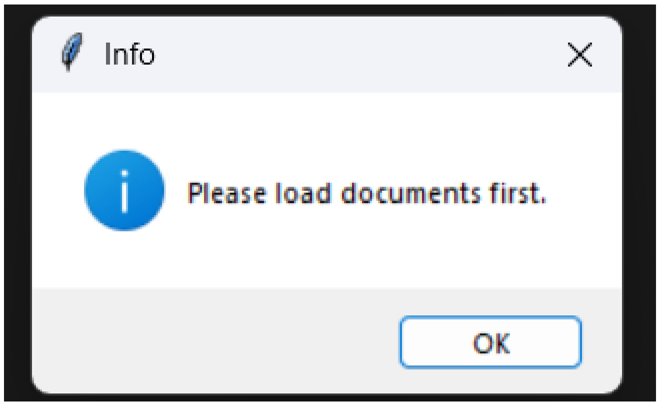
```python
import os       # For file handling
import re       # For regular expressions
from collections import defaultdict     # For creating an inverted index
import tkinter as tk        # For GUI
from tkinter import ttk, filedialog, messagebox     # For UI elements
from tkinter import scrolledtext                    # For scrollable text box
```

- **Output & Display:**

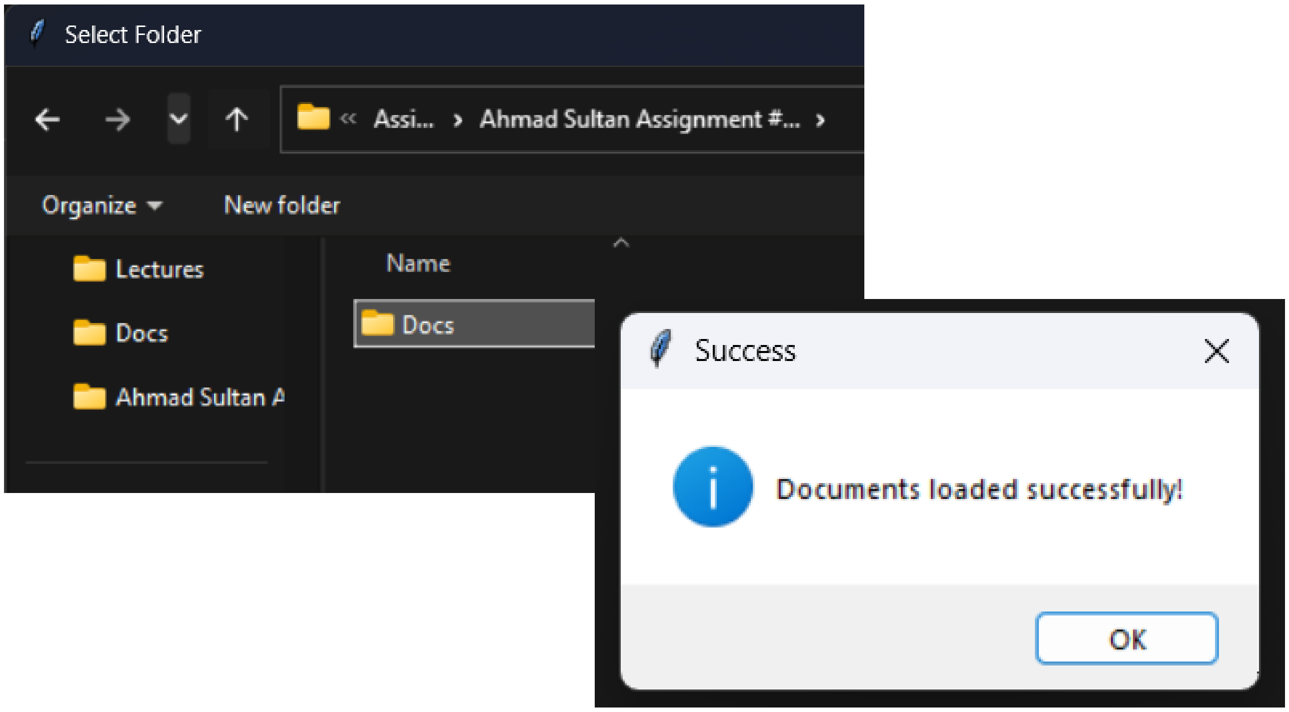

BoolPySearch v1.0.0

BoolPySearch

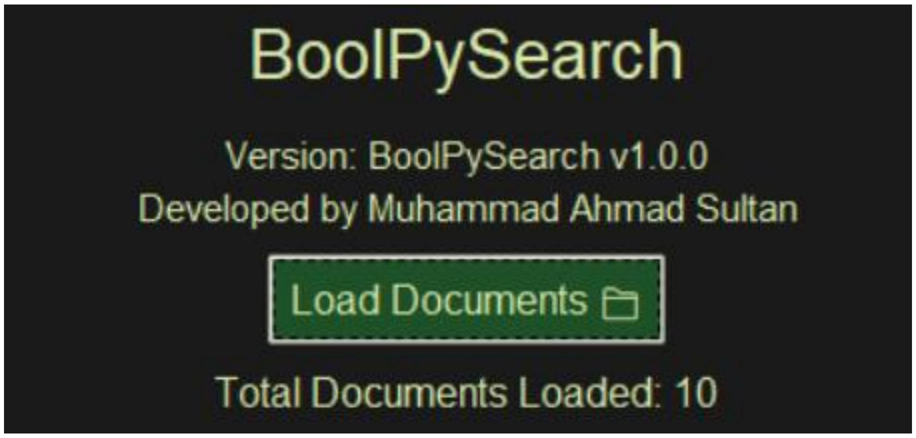
Version: BoolPySearch v1.0.0
Developed by Muhammad Ahmad Sultan



Please load documents first.



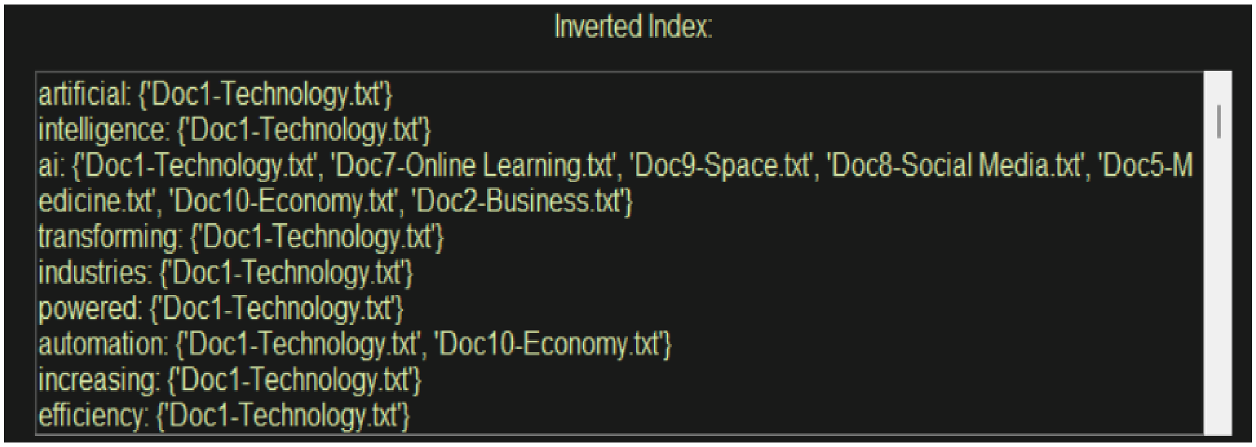
Documents loaded successfully!


BoolPySearch

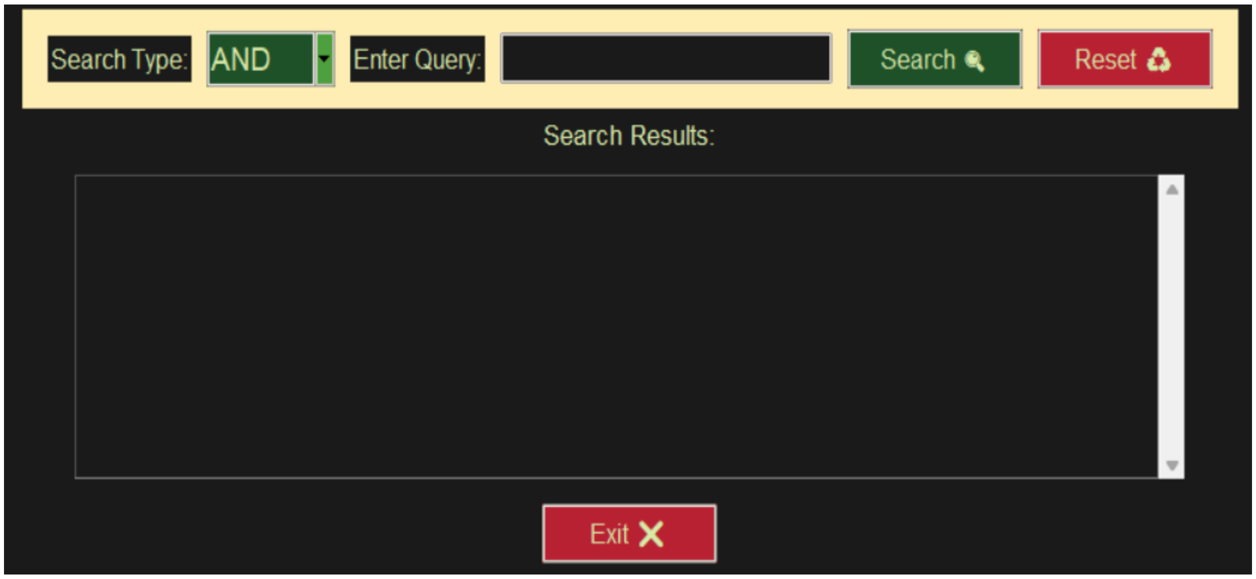
Version: BoolPySearch v1.0.0

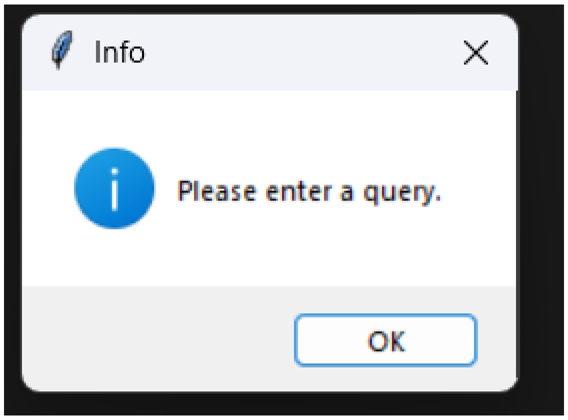
Developed by Muhammad Ahmad Sultan

Total Documents Loaded: 10

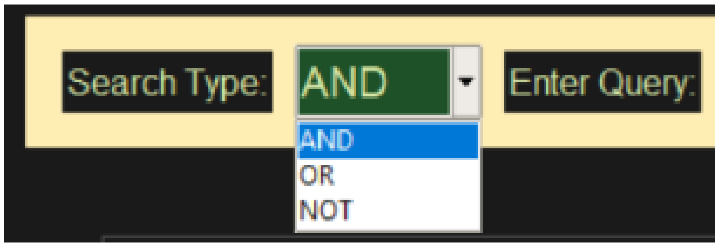

Inverted Index:

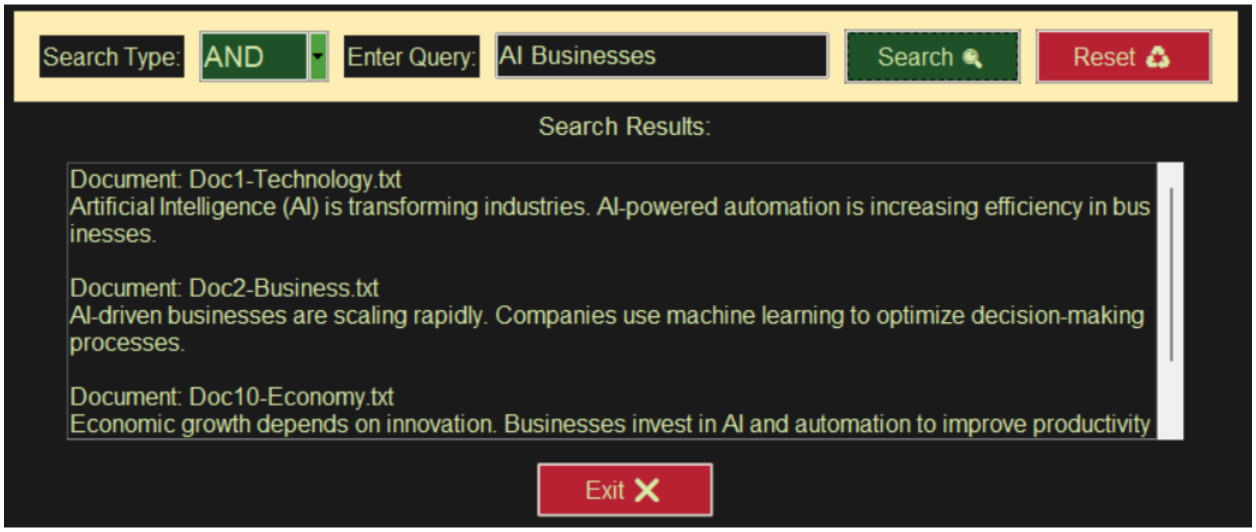
```
artificial: {'Doc1-Technology.txt'}
intelligence: {'Doc1-Technology.txt'}
ai: {'Doc1-Technology.txt', 'Doc7-Online Learning.txt', 'Doc9-Space.txt', 'Doc8-Social Media.txt', 'Doc5-M
edicine.txt', 'Doc10-Economy.txt', 'Doc2-Business.txt'}
transforming: {'Doc1-Technology.txt'}
industries: {'Doc1-Technology.txt'}
powered: {'Doc1-Technology.txt'}
automation: {'Doc1-Technology.txt', 'Doc10-Economy.txt'}
increasing: {'Doc1-Technology.txt'}
efficiency: {'Doc1-Technology.txt'}
```





Please enter a query.




Search Type: AND

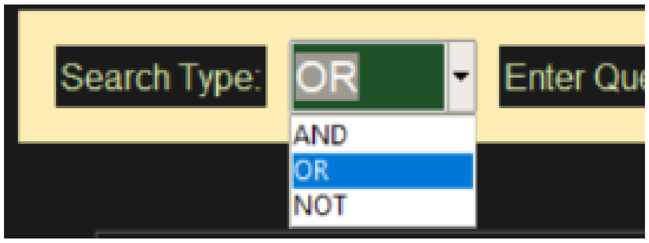
Enter Query: AI Businesses

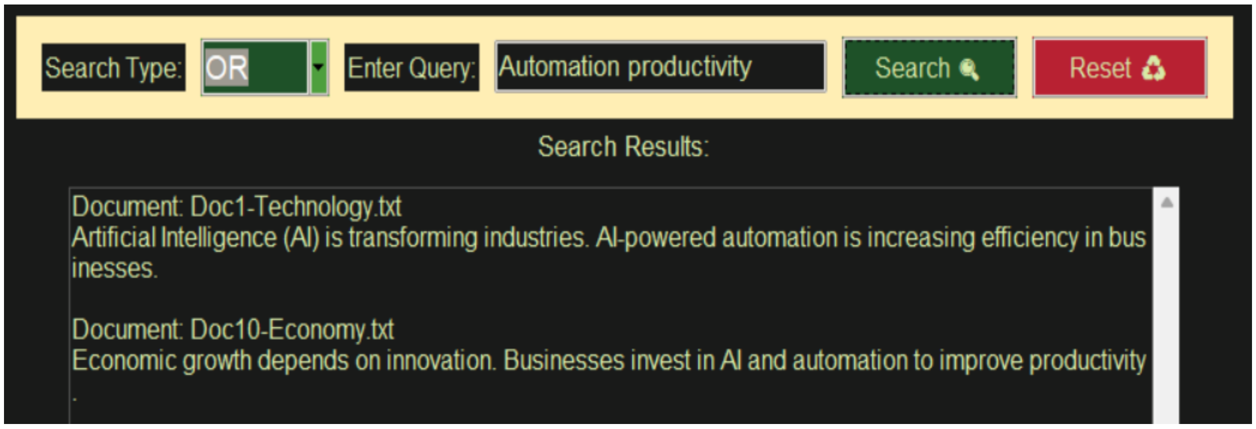
Search Results:

Document: Doc1-Technology.txt
Artificial Intelligence (AI) is transforming industries. AI-powered automation is increasing efficiency in bus
inesses.

Document: Doc2-Business.txt
AI-driven businesses are scaling rapidly. Companies use machine learning to optimize decision-making
processes.

Document: Doc10-Economy.txt
Economic growth depends on innovation. Businesses invest in AI and automation to improve productivity



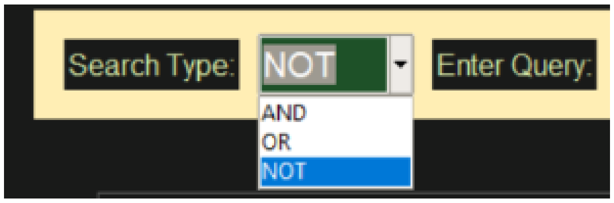

Search Type: OR

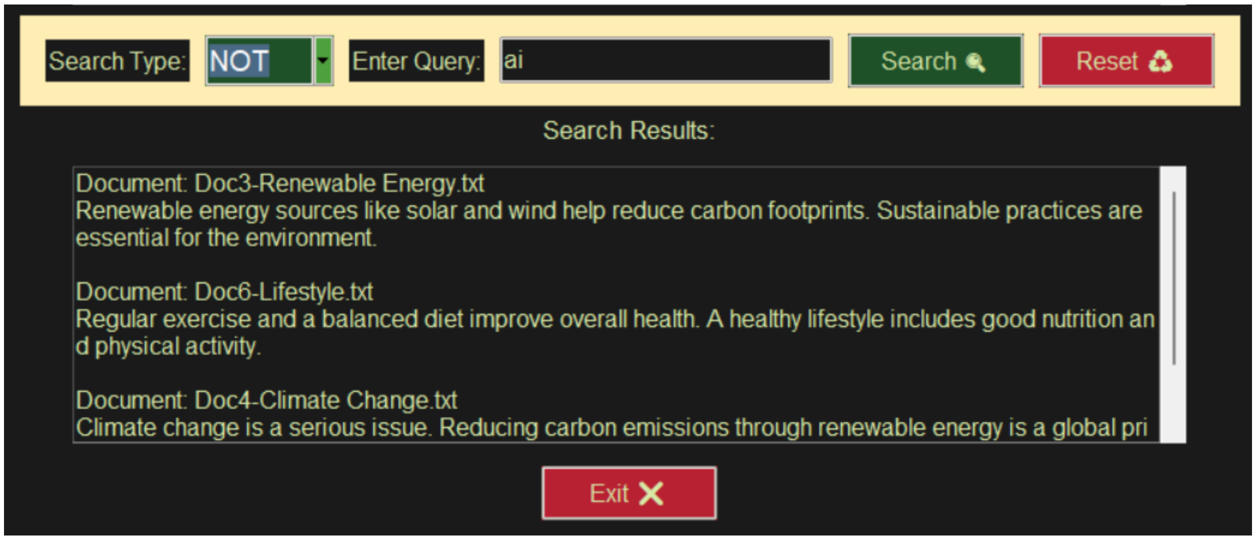
Enter Query: Automation productivity

Search Results:

Document: Doc1-Technology.txt
Artificial Intelligence (AI) is transforming industries. AI-powered automation is increasing efficiency in bus
inesses.

Document: Doc10-Economy.txt
Economic growth depends on innovation. Businesses invest in AI and automation to improve productivity
.



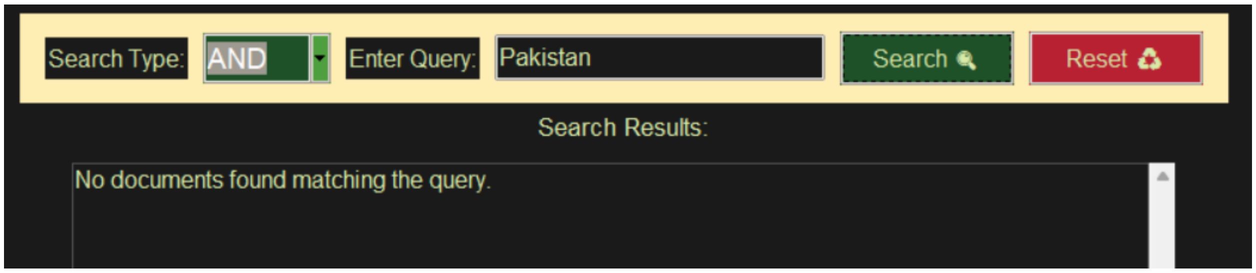


ai

Document: Doc3-Renewable Energy.txt
Renewable energy sources like solar and wind help reduce carbon footprints. Sustainable practices are essential for the environment.

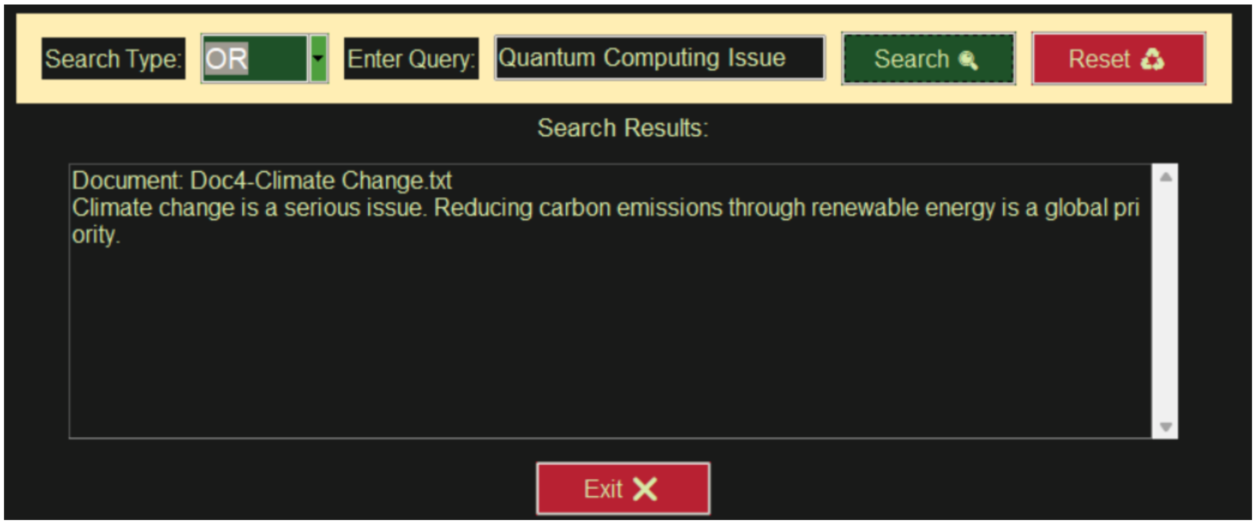
Document: Doc6-Lifestyle.txt
Regular exercise and a balanced diet improve overall health. A healthy lifestyle includes good nutrition an d physical activity.

Document: Doc4-Climate Change.txt
Climate change is a serious issue. Reducing carbon emissions through renewable energy is a global pri



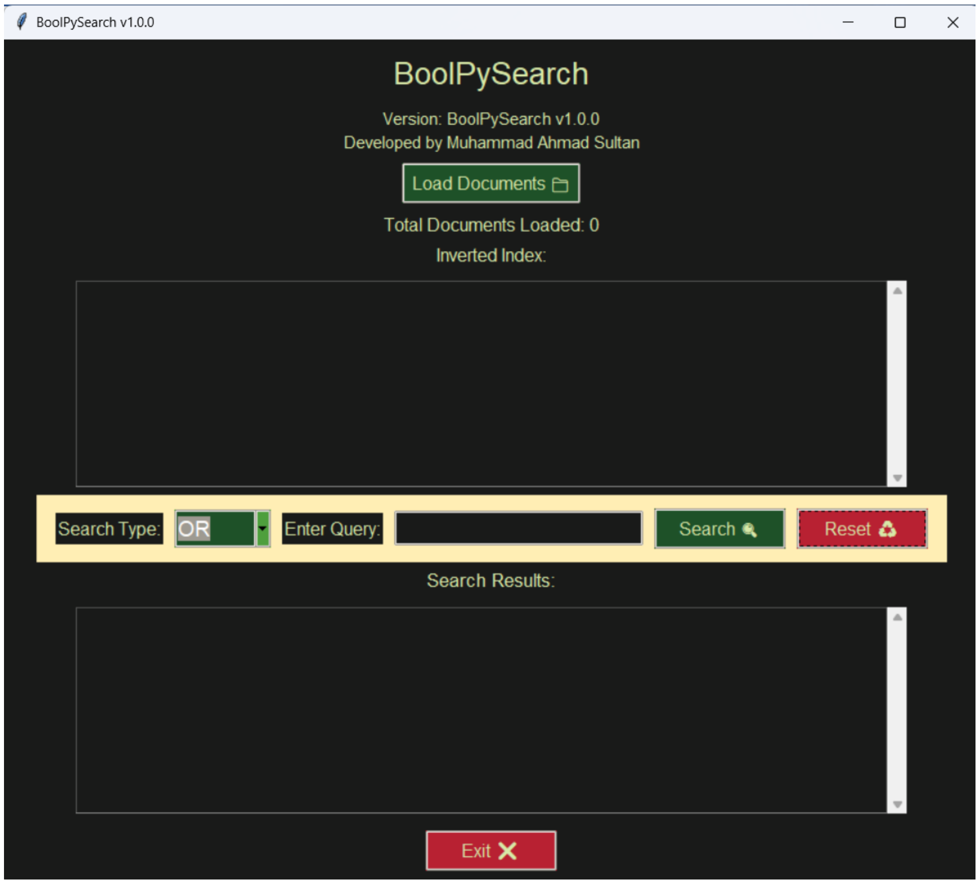
Pakistan

No documents found matching the query.



Quantum Computing Issue

Document: Doc4-Climate Change.txt
Climate change is a serious issue. Reducing carbon emissions through renewable energy is a global pri ority.



Version: BoolPySearch v1.0.0
Developed by Muhammad Ahmad Sultan

Total Documents Loaded: 0

▪ 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