

Session: 2021 – 2025

**Submitted by:**

Abdul Mateen 2021-CS-190

**Supervised to:**

Dr. Khaldoon Syed Khurshid

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

# Report

This document provides a comprehensive overview of the Django-based search application, which includes functionality for keyword matching and TF-IDF ranking of textual documents. The codebase is modular and divided into functions for handling document reading, keyword matching, and TF-IDF ranking.

## Overview

The application allows users to search through textual documents stored in a directory and ranks the documents based on the search query. Two search methods are supported:

1. **Keyword Matching**: Matches the query words directly with the words in the documents.
2. **TF-IDF Scoring**: Ranks documents based on term frequency-inverse document frequency.

## Key Components

## 1. Search View

**File:** views.py

The search\_view function handles the user's search request, retrieves documents, performs the appropriate search operation, and renders the results in a Django template.

**Function Logic:**

* **Input:**
  + query (Search query string)
  + type (Search type: content or tf-idf)
* **Steps:**
  1. Check if the HTTP request is POST.
  2. Extract query and type from the POST request.
  3. Read documents using read\_documents().
  4. Perform the search using:
     + keyword\_matching(query, documents) for keyword-based search.
     + calculate\_tf\_idf(query, documents) for TF-IDF-based ranking.
  5. Render the results to the template ranking/search.html.

### 2. Document Handling

**File:** utils.py

**read\_documents()**

Reads all .txt files from the documents directory and returns a dictionary where keys are filenames and values are file content.

### 3. Keyword Matching

**Function:** keyword\_matching(query, documents)

Ranks documents based on the count of query words found in each document.

**Logic:**

* Convert the query and document content to lowercase.
* Split the content into words.
* Count the occurrences of query words in each document.
* Sort documents by the count in descending order.

### 4. TF-IDF Ranking

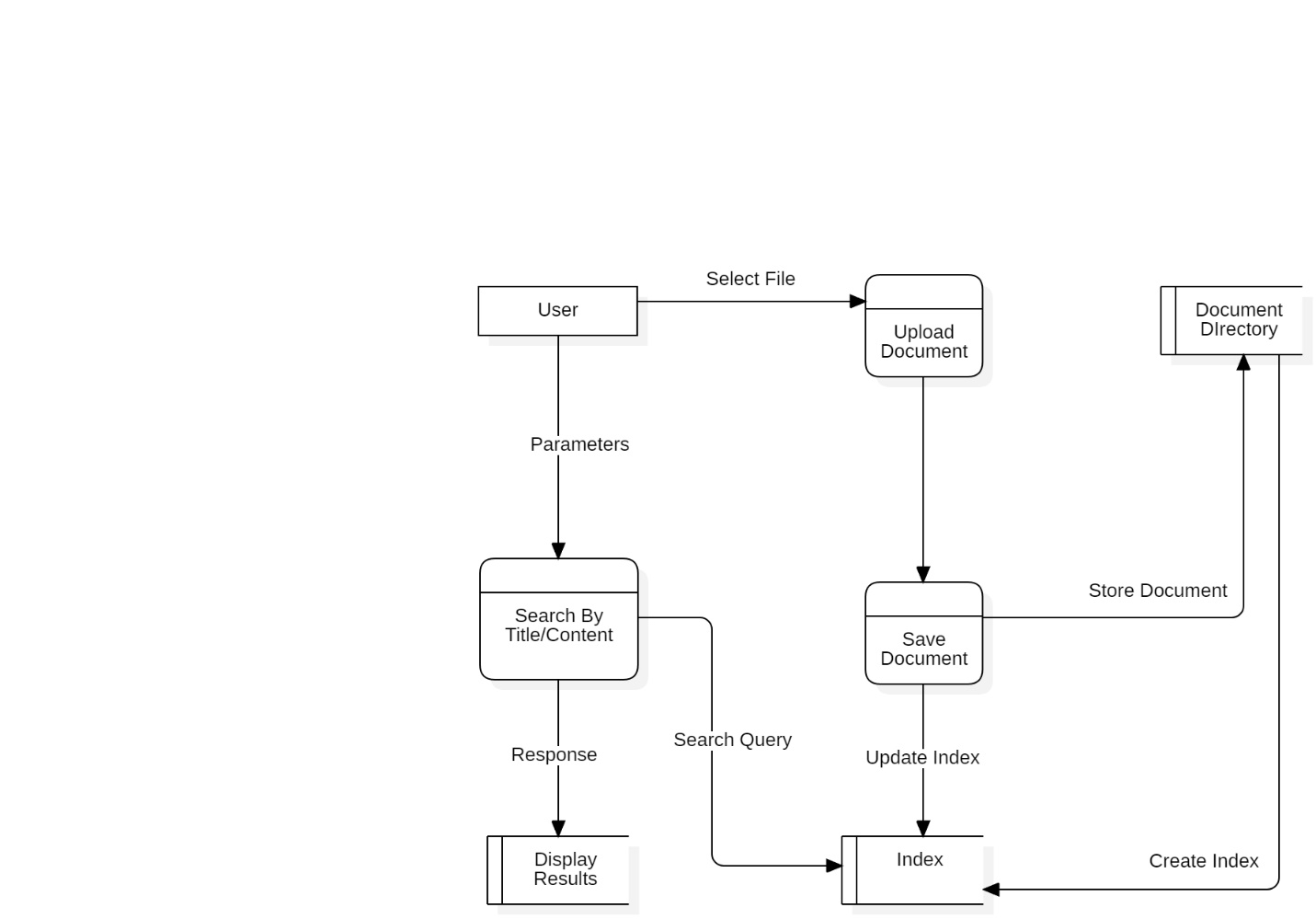
**Function:** calculate\_tf\_idf(query, documents)

Ranks documents using the TF-IDF (Term Frequency-Inverse Document Frequency) scoring method.

**Logic:**

1. Tokenize the query into words.
2. Calculate the term frequency (TF) for each word in a document.
3. Compute the inverse document frequency (IDF) for each word.
4. Calculate the TF-IDF score for the query words in each document.
5. Sort documents by their total TF-IDF score in descending order.

## **Data Flow Diagram**



## **Summary**

This application is a simple yet effective solution for ranking documents based on user queries. It uses both basic keyword matching and the more advanced TF-IDF method to ensure relevant results. The modular design ensures easy maintainability and extensibility.