Credit-Based Document Scanning System

T. Gnana kartheek

9381195913

1. Introduction

The Credit-Based Document Scanning System is a web application built using Django that allows users to scan and match documents efficiently. The system operates on a credit-based model, where users need sufficient credits to perform document scanning. It includes user authentication, credit management, and AI-powered document processing.

The system supports two user roles and utilizes the OpenAI API for advanced text similarity analysis and document comparison. By leveraging AI-driven analysis, the platform ensures higher accuracy in detecting similar content. The system is designed to be user-friendly, secure, and scalable, making it suitable for automated document processing.

Additionally, it provides a detailed scan history for users to track their document comparisons. The platform is optimized for efficient performance, ensuring fast processing times even for large documents.

2. System Requirements

2.1 Hardware Requirements

- Processor: Intel Core i5 or higher / AMD Ryzen 5 or higher
- RAM: Minimum 8GB (16GB recommended for better performance)
- Storage: Minimum 20GB of free space (SSD recommended)
- Operating System: Windows 10/11, macOS, or Linux (Ubuntu 20.04 or later)

2.2 Software Requirements

- Programming Language: Python 3.10+
- Framework: Django 4.x
- Frontend: HTML, CSS, JavaScript (Bootstrap)
- Database: SQL 13+
- AI & NLP: OpenAI API, Scikit-learn (for TF-IDF & Cosine Similarity)

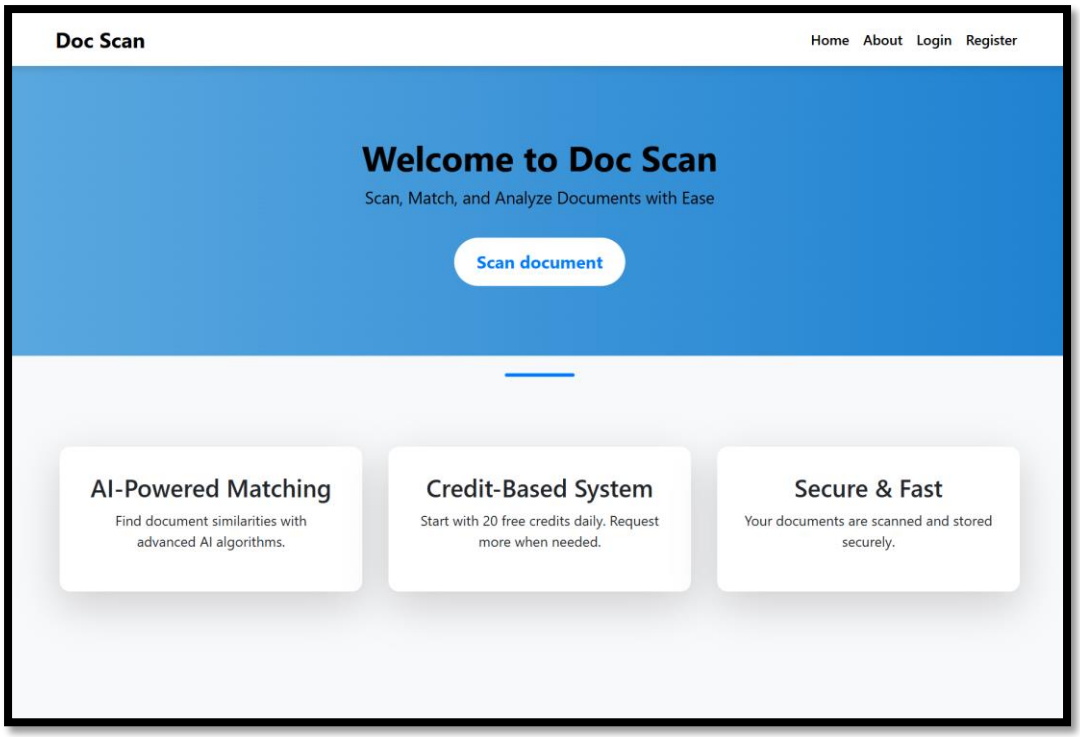
3. Features & Functionalities

3.1 Home Page & Navigation

User-Friendly Dashboard: Displays a welcoming message with quick access to key features

Navigation Menu: Includes links to Login, Registration, Profile, About, Home.

Credit-Based System & AI Features: The home page also provides information about the credit-based system for document scanning and highlights the AI-powered similarity detection feature for enhanced accuracy.



3.2 User Authentication & Role Management

User Registration & Login:

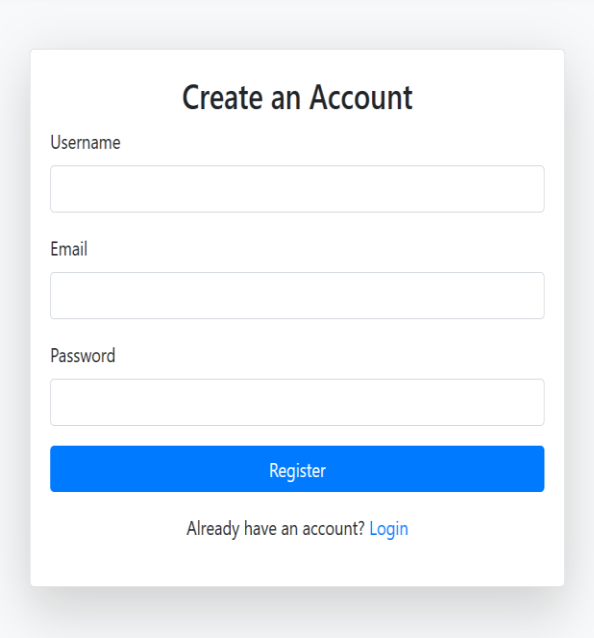
Users can sign up with their username, email, and password.

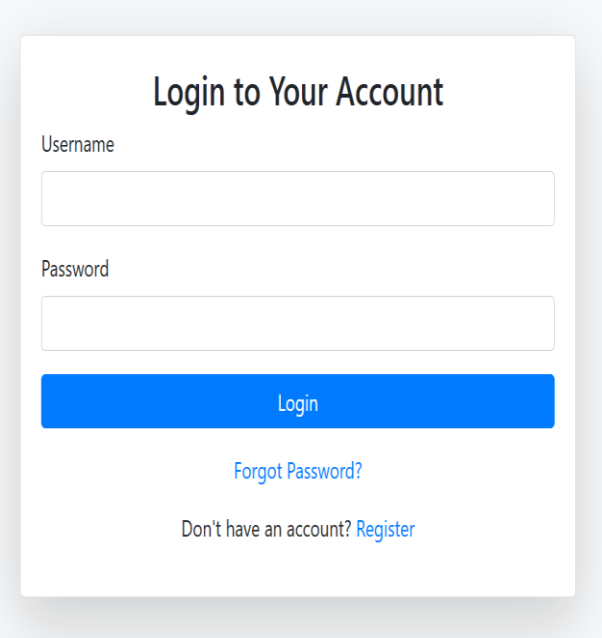
Login system with session-based authentication

Two user roles:

Standard User: Can scan documents using available credits.

Admin: Can upload documents, approve or deny credits, and analyze system usage.





3.3 Credit-Based Document Scanning System

Credit Management:

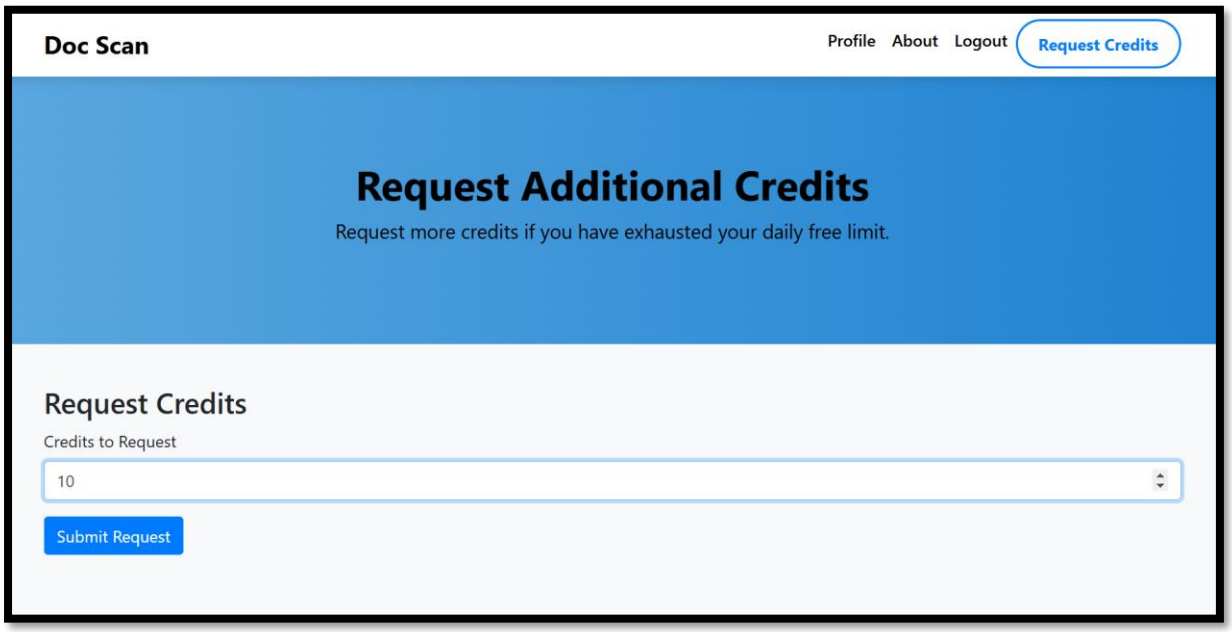
Each user gets 20 free credits per day, automatically reset at midnight.

Each document scan deducts **1 credit** from the user's balance.

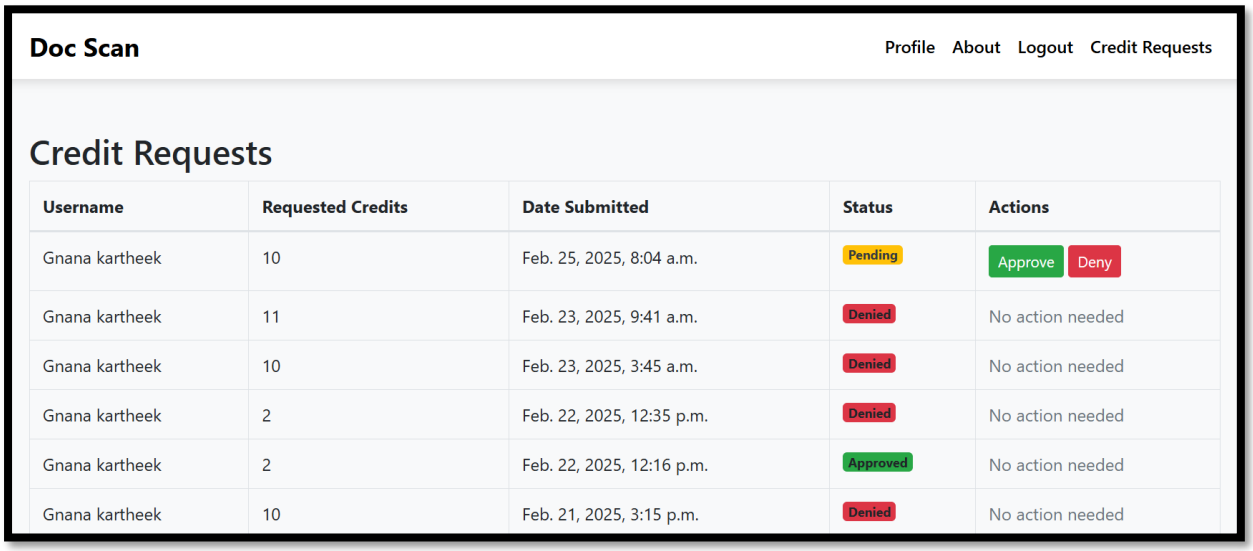
Users can send a request to the admin for more credits.

Admin can approve or deny user credit requests.

User Request for credits:



Admin Credit Request Action:



3.4 Document Scanning & Matching

Document Upload: Users can upload plain text documents for scanning.

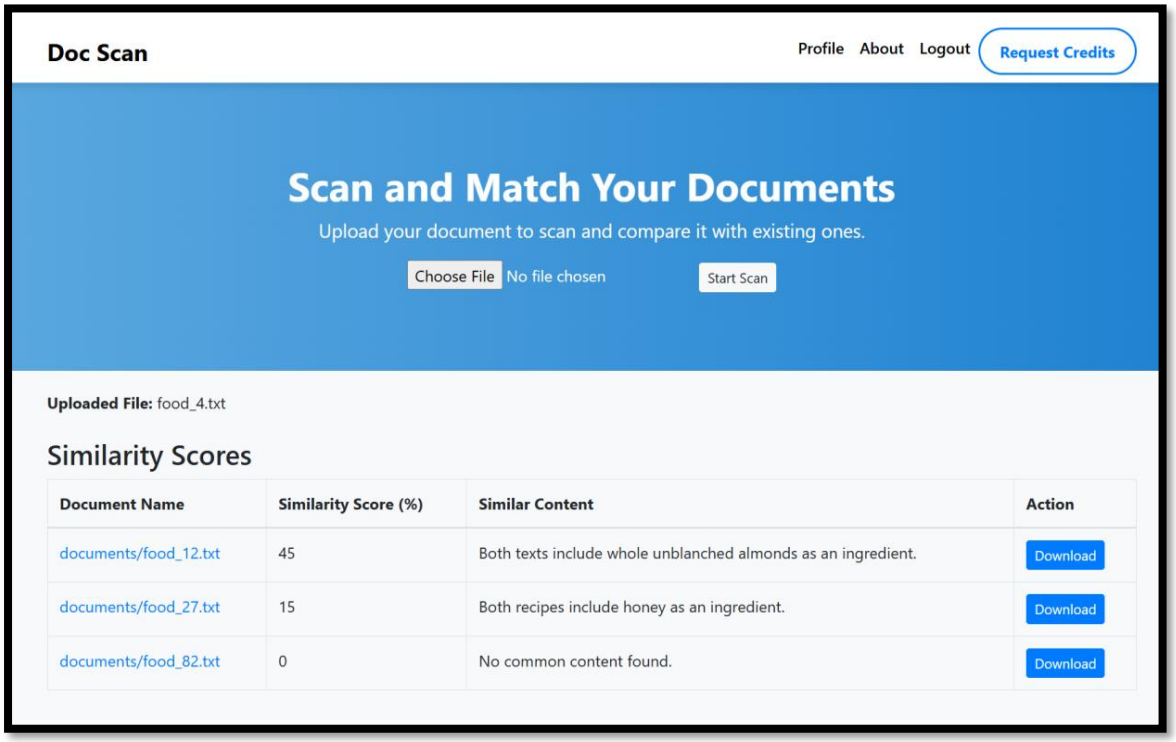
Text Matching Algorithm: Initially, similarity is calculated using **TF-IDF and Cosine Similarity**, which focus on word frequency and vector-space comparisons. (Traditional Approach)

AI-Powered Matching: The system calls the OpenAI API to extract a meaningful, one-line summary of common content between two documents.

It ensures that the extracted content is concise and contextually relevant.

I have utilized OpenAI's GPT-4o-mini, a lightweight yet powerful AI model optimized for efficient and context-aware text analysis.

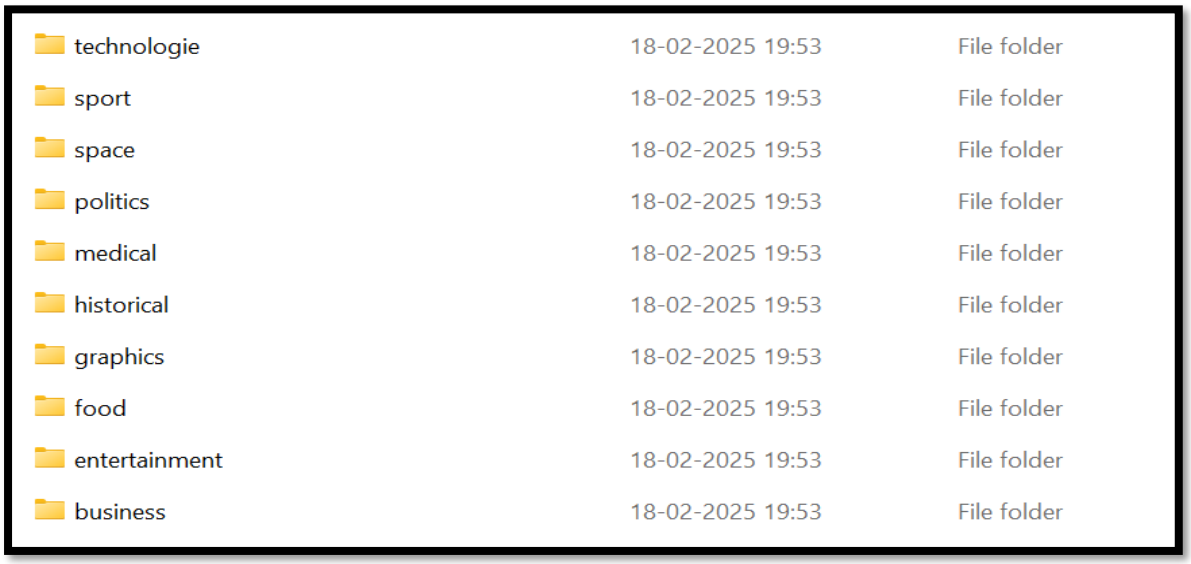
User Scan's the document:



Admin upload the documents:

I have Utilized 1000 text file date set for this project which contains10 folder and in each folder there are 100 text files.

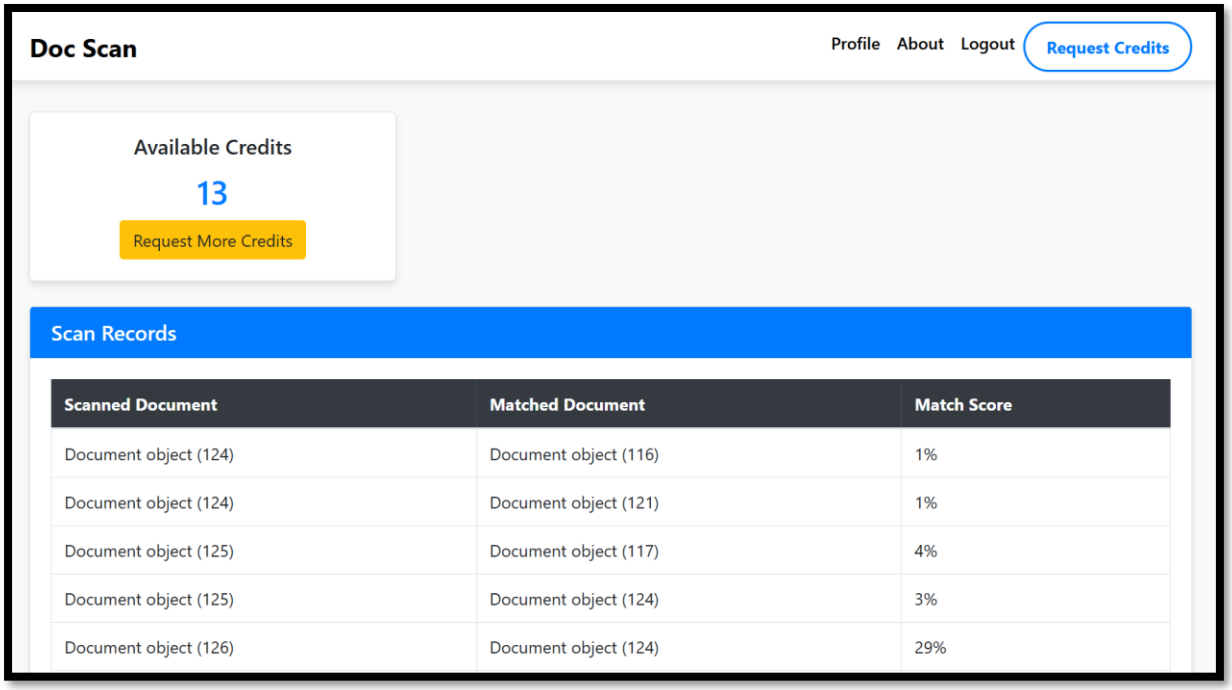
Dataset:



technologie	18-02-2025 19:53	File folder
sport	18-02-2025 19:53	File folder
space	18-02-2025 19:53	File folder
politics	18-02-2025 19:53	File folder
medical	18-02-2025 19:53	File folder
historical	18-02-2025 19:53	File folder
graphics	18-02-2025 19:53	File folder
food	18-02-2025 19:53	File folder
entertainment	18-02-2025 19:53	File folder
business	18-02-2025 19:53	File folder

3.5 User's Profile & History

1. **User profile**: Users can view their credit balance and past document scans and past credit request history.



Scanned Document	Matched Document	Match Score
Document object (124)	Document object (116)	1%
Document object (124)	Document object (121)	1%
Document object (125)	Document object (117)	4%
Document object (125)	Document object (124)	3%
Document object (126)	Document object (124)	29%

Credit Requests

Requested Credits	Status
10	Approved
5	Approved
10	Approved
10	Approved

2. **Admin profile:** Admin can manage past uploaded documents and users past credit request history.

Uploaded Documents

Document	Uploaded On	Action
documents/food_82.txt	Feb. 19, 2025, 10:43 a.m.	Delete
documents/food_83.txt	Feb. 19, 2025, 10:43 a.m.	Delete
documents/sport_93.txt	Feb. 19, 2025, 10:44 a.m.	Delete
documents/sport_92.txt	Feb. 19, 2025, 10:44 a.m.	Delete
documents/sport_97.txt	Feb. 19, 2025, 10:45 a.m.	Delete

Credit Requests

User	Requested Credits	Status	Action
Gnana kartheek	10		Processed
Gnana kartheek	5		Processed
Gnana kartheek	10		Processed
Gnana kartheek	10		Processed

3.6 Security & Access Control

a. **User Authentication:** The system uses Django's built-in authentication mechanism, which is session-based.
User passwords are never stored in plain text, Django automatically hashes passwords using a secure hashing algorithm before storing them in the database.

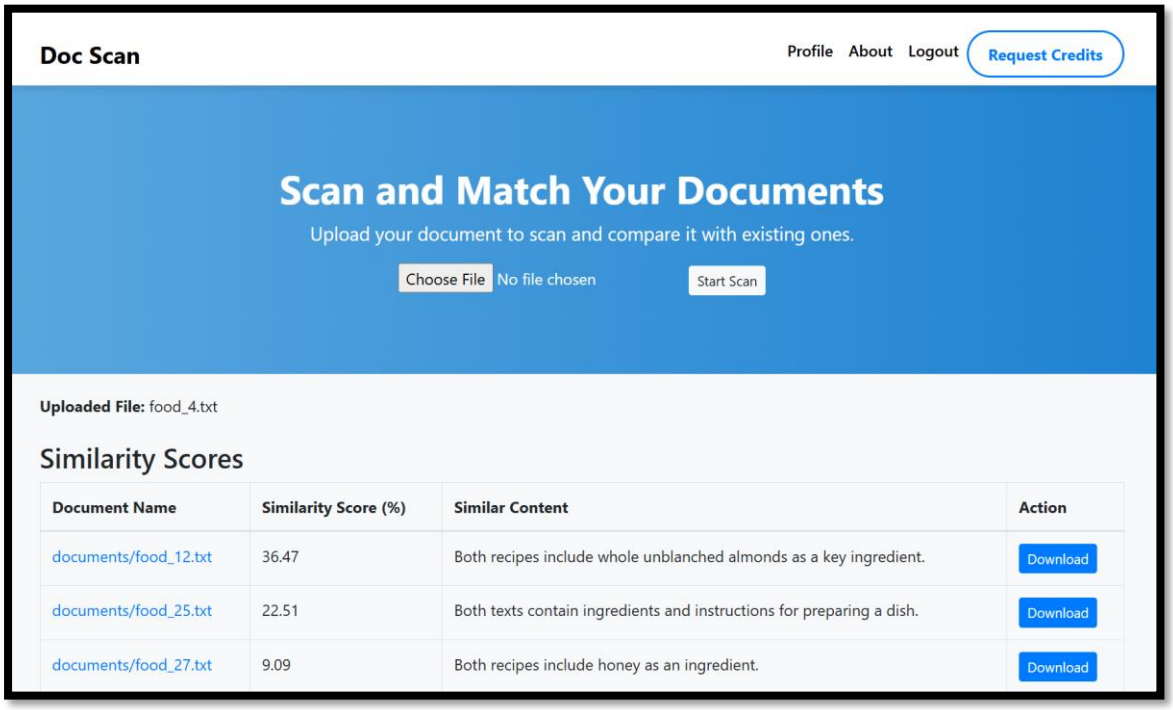
b. **Access Control:** @login_required decorator is applied to key views, ensuring that only authenticated users can upload and scan documents.
Only authenticated users with available credits can scan documents. If a user has insufficient credits, they are prevented from proceeding.

4. Performance Evaluation

I implemented two versions of the document similarity detection process and evaluated their effectiveness by scanning **food_4.txt** against a set of stored documents.

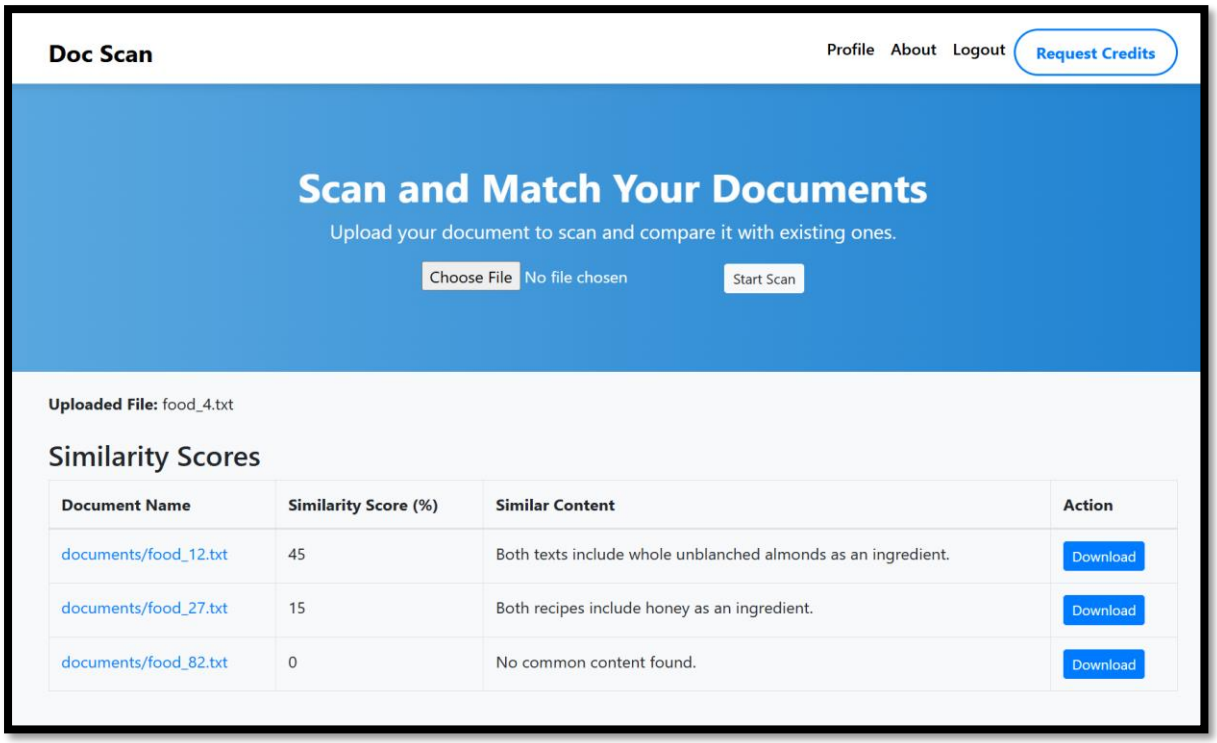
Version 1: TF-IDF & Cosine Similarity + OpenAI

- In this approach, I first used TF-IDF (Term Frequency-Inverse Document Frequency) and cosine similarity to compute similarity scores between the uploaded document and stored documents.
- Based on the top 3 most similar documents, I then leveraged the OpenAI API to extract a short, meaningful common content between the uploaded document and the top matches.
- **Results**: food_4.txt achieved a 36% similarity score with food_12.txt using TF-IDF & Cosine Similarity.



Version 2: OpenAI API for Direct Similarity Calculation

- Instead of using TF-IDF and cosine similarity, I directly utilized the OpenAI API to compute the similarity score and extract common content between the uploaded document and each stored document.
- The OpenAI model provided a more context-aware similarity analysis, capturing deeper semantic relationships beyond just word frequency and vector space representation.
- **Results**: Using OpenAI API directly, food_4.txt achieved a 45% similarity score with food_12.txt, showing a notable improvement over TF-IDF.



5. Conclusion

The Credit-Based Document Scanning System successfully integrates AI-powered document analysis with a structured credit-based access model, offering a secure, efficient, and user-friendly solution for document similarity detection. By leveraging the OpenAI API, the system enhances accuracy in identifying similar content, surpassing traditional text-matching techniques like TF-IDF and Cosine Similarity.

With features such as user authentication, credit management, scan history tracking the platform provides a seamless experience for users. The scalability and performance optimizations ensure that the system can handle large document comparisons effectively.

This project demonstrates the potential of AI-driven automation in document processing, making it a valuable tool for applications in plagiarism detection, legal document comparison, content validation, and research analysis.

Looking ahead, the system has significant potential for growth, with opportunities to integrate real-time scanning, multi-language support, and advanced AI models for even deeper text analysis. Future enhancements could include automated summarization, document classification, and integration with cloud-based services to expand its capabilities and improve user experience.