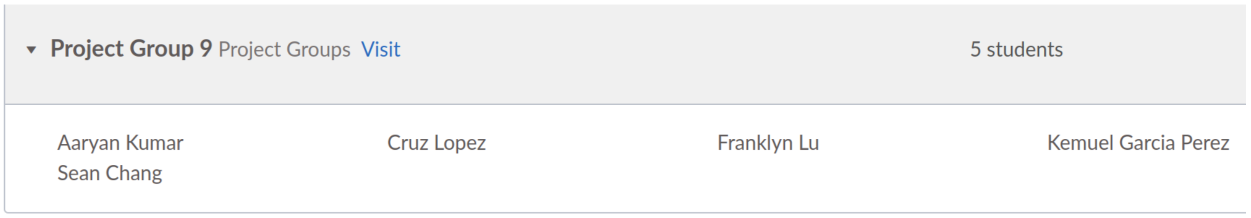
Group 9 Project Part B Report



Project Group 9 Project Groups Visit 5 students

Aaryan Kumar
Sean Chang
Cruz Lopez
Franklyn Lu
Kemuel Garcia Perez

Excluding Sean Chang

Collaboration Details:

Aaryan Kumar:

- Modified and developed searching, indexing, and display scripts for the search engine, and added many features like limiting comment numbers and shortening result snippets during display, improving search performance and quality.
- Helped develop and integrate the Flask-based front end, connecting the user interface with backend search functionality and the PyLucene indexing system to support real-time query processing. This allows users to use the search engine without interacting much with the scripts/code through a simple web interface.
- Helped create and work on the project's final report, detailing features and data indexing techniques/searching algorithms.
- Coordinated team meetings to track progress and align development efforts. Scheduled and led check-ins to ensure everyone was on pace and blockers were addressed early.

Cruz Lopez:

- Helped coordinate team meetings to keep up with team progress and ensure the team was not having any sort of unaddressed issues.
- Initialized the flask_demo.py file to integrate everything with Flask and help create the search engine
- Integrated the Flask front end for both input.html and output.html to include the search bar and display the search results.
- Designed the front end for both input/output files to have a cleaner look and look more designed for Reddit search by using HTML and CSS elements.
- Tested ran port servers to ensure that everything is properly connected to make the search engine run properly and efficiently.

Franklyn Lu:

- Helped initialize the Flask front end, specifically focusing on integrating the user interface with the backend search logic and PyLucene index, enabling result presentation.

- Debugged and improved the indexing scripts for both Reddit JSON and HTML sources. This included fixing parsing errors, improving field extraction, and ensuring compatibility with PyLucene's analyzers.
- Helped write and organize the final project report, including an architecture overview. Ensured clarity and completeness for technical and non-technical readers.
- Actively engaged with the TA by asking detailed questions throughout the project to clarify requirements, resolve technical uncertainties, and ensure the team stayed aligned with the assignment expectations.

Ivan Li:

- Implemented the fuzzy boosting ranking method within the Pylucene search framework, enabling an alternative to BM25 scoring for improved query flexibility and semantic matching.
- Worked on the HTML user interface logic to support dynamic ranking method selection, allowing users to toggle between BM25 and fuzzy boosting when submitting queries. This involved updating both the front-end form and the backend routing logic to ensure consistent result rendering

Kemuel Garcia Perez:

- Modified the crawling file to enhance compatibility with the Bolt server environment by removing sensitive information storage.
- Executed web crawling for multiple Reddit communities, including r/redditserials, r/prorevenge, and r/confession subreddits, ensuring diverse content coverage for the search engine dataset.
- Coordinated the transfer of locally crawled data to the shared server infrastructure, enabling centralized data access and facilitating team collaboration for processing and indexing.
- Established the foundational search and indexing framework, serving as the base architecture for subsequent feature development, including implementing multiple page ranking algorithms and Flask-based web interface integration.

Overview:

The primary goal was to build an end-to-end search engine pipeline using PyLucene to index and retrieve Reddit posts and/or HTML content. The system includes both an efficient backend indexing process and a lightweight, user-friendly web-based search interface. This allows users to enter search queries and receive ranked results in real-time based on relevance and timestamp.

Architecture:

Our system uses Python with PyLucene for indexing and querying, and Flask to serve a web-based search interface. The architecture consists of two main components:

1. Indexing Engine: A Python script that processes and indexes documents (Reddit JSON or HTML files) into a Lucene index. Each document includes relevant metadata fields such as title, body, timestamp, and username.

2. Web Interface: A Flask-based front end that includes a query textbox and a results display section. The search is performed on the Lucene index and returns the top 10 most relevant results based on a custom ranking function.

Indexing Strategy:

- For HTML files, we parse:
    - title: extracted from <title> tag
    - body: extracted from HTML content (stripped of tags)
    - creation date: extracted from metadata if available
- For Reddit JSON, we parse:
    - body: main text content of the post
    - username: Reddit user handle
    - timestamp: used in ranking
    - score/upvotes: used optionally in ranking (if available)

All fields are indexed using PyLucene's Document and Field classes with appropriate analyzers (StandardAnalyzer for English). For Reddit content, we combine BM25 similarity (default in PyLucene) with a time-decay function to boost more recent results. We also included a toggle option for users to select the Fuzzy Search method.

Search Algorithm and Scoring Methods:

Our system uses PyLucene, which implements the Lucene search library in Java and exposes it to Python. By default, Lucene uses the BM25 similarity algorithm, a state-of-the-art ranking function in information retrieval that balances term frequency and document frequency to compute relevance scores. For Reddit data, we enhanced the ranking by incorporating a time-based decay function to prioritize more recent posts while preserving relevance. This scoring method mimics Reddit's natural emphasis on recent and relevant content, ensuring that users see results that are both timely and informative. For HTML files, we relied solely on BM25 scoring since timestamps are often missing or inconsistent. Fields such as title, body, and optional date (if available) were tokenized and indexed using Lucene's StandardAnalyzer, which performs lowercasing, stop word removal, and token normalization. All search queries are parsed into Lucene query syntax and executed via the IndexSearcher, returning the top 10 documents ranked by their computed score.

In addition to BM25 scoring, we implemented a fuzzy boosting ranking method to improve retrieval performance for queries with minor misspellings or semantic variations. This method enhances Lucene's fuzzy matching capability by assigning boosted scores to documents that contain approximate matches to the query terms, rather than exact ones. It is particularly effective when users enter noisy or imprecise inputs, such as typos or word inflections.

Challenges and Limitations:

- PyLucene setup complexity: Installing and configuring PyLucene is non-trivial, especially across different operating systems.

- Limited metadata: Some Reddit posts lack consistent timestamps or usernames, which limits the quality of indexing.

- HTML parsing errors: Improper or malformed HTML tags occasionally led to parsing issues.

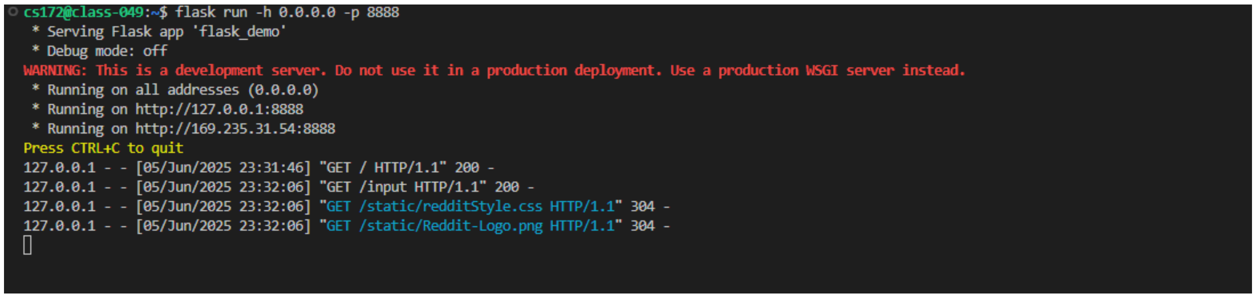
- Scalability: The current setup works well for small to medium datasets. For larger datasets, memory usage and indexing time can be significant.

Instructions to deploy the system:

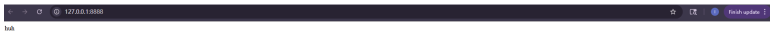
1. Open VS Code
2. Make sure you at the correct directory: cs172@class-049~
3. Build index: python3 build_index.py

4. Run the code:
    a. export FLASK_APP=flask_demo
    b. Flask run -h 0.0.0.0 -p 8888
    c. Open the external webpage, and you should see a “huh” on the screen
    d. Replace the URL with this: http://localhost:8888/input

Screenshots:

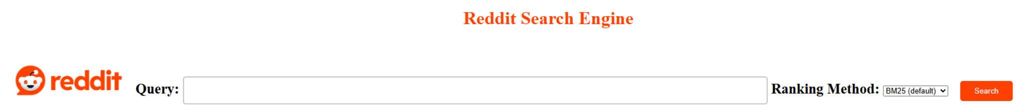


```
cs172@class-049:~$ flask run -h 0.0.0.0 -p 8888
 * Serving Flask app 'flask_demo'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8888
 * Running on http://169.235.31.54:8888
Press CTRL+C to quit
127.0.0.1 - - [05/Jun/2025 23:31:46] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2025 23:32:06] "GET /input HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2025 23:32:06] "GET /static/redditStyle.css HTTP/1.1" 304 -
127.0.0.1 - - [05/Jun/2025 23:32:06] "GET /static/Reddit-Logo.png HTTP/1.1" 304 -
```



127.0.0.1:8888

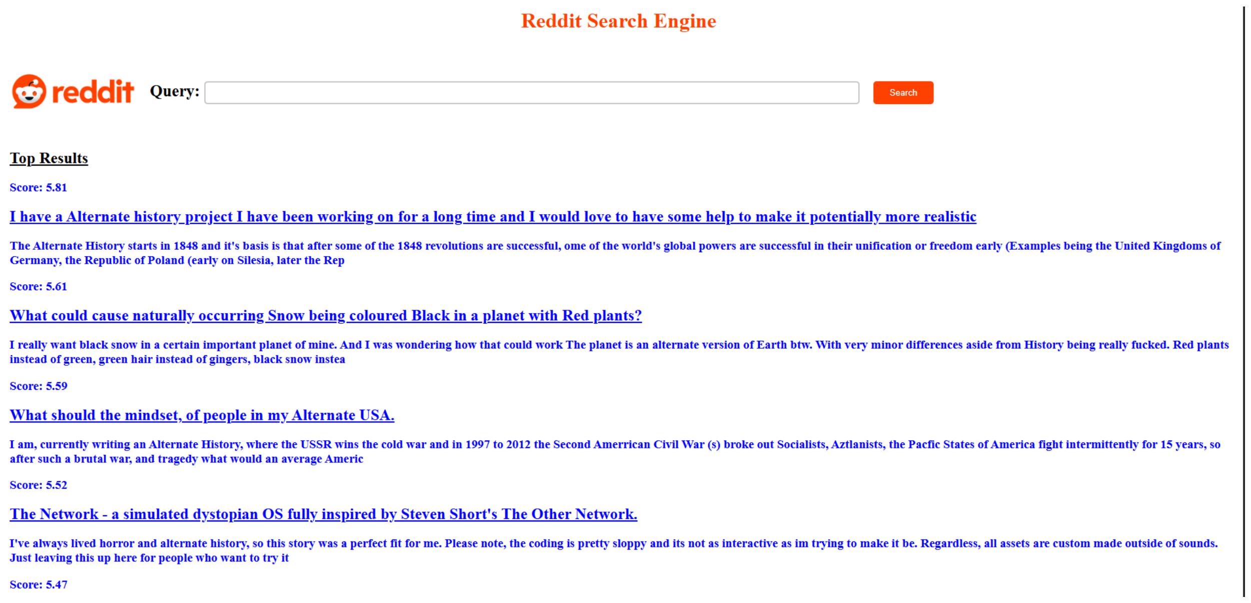
huh



Reddit Search Engine

Query:

Ranking Method: BM25 (default)



Top Results

Score: 5.81

[I have a Alternate history project I have been working on for a long time and I would love to have some help to make it potentially more realistic]

The Alternate History starts in 1848 and it's basis is that after some of the 1848 revolutions are successful, ome of the world's global powers are successful in their unification or freedom early (Examples being the United Kingdoms of Germany, the Republic of Poland (early on Silesia, later the Rep

Score: 5.61

What could cause naturally occurring Snow being coloured Black in a planet with Red plants?

I really want black snow in a certain important planet of mine. And I was wondering how that could work The planet is an alternate version of Earth btw. With very minor differences aside from History being really fucked. Red plants instead of green, green hair instead of gingers, black snow instea

Score: 5.59

What should the mindset, of people in my Alternate USA.

I am, currently writing an Alternate History, where the USSR wins the cold war and in 1997 to 2012 the Second Amerrican Civil War (s) broke out Socialists, Aztlanists, the Pacfic States of America fight intermittently for 15 years, so after such a brutal war, and tragedy what would an average Americ

Score: 5.52

The Network - a simulated dystopian OS fully inspired by Steven Short's The Other Network.

I've always lived horror and alternate history, so this story was a perfect fit for me. Please note, the coding is pretty sloppy and its not as interactive as im trying to make it be. Regardless, all assets are custom made outside of sounds. Just leaving this up here for people who want to try it

Score: 5.47

Link to Video Demo:

https://www.youtube.com/watch?v=IH360s8eP5k&t=4s