INNOMATICS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

“Enhancing Search Engine Relevance for Video Subtitles”

Submitted By:
Team ID: T211050
1. Pandey Abhishek Nath Roy - IN1240134
2. Srinivasarao Kesana – IN1240318

OBJECTIVE

The goal of this project is to create a sophisticated search engine algorithm that can efficiently retrieve subtitles based on user queries. The algorithm will place a special emphasis on the content of the subtitles. To achieve this, we will leverage natural language processing (NLP) and machine learning (ML) techniques, which will ultimately enhance the accuracy and relevance of the search results delivered to the user.

- ❑ Develop an advanced search engine algorithm that retrieves subtitles based on user queries, with a particular emphasis on subtitle content.
- ❑ Leverage natural language processing (NLP) and machine learning (ML) techniques to improve the accuracy and relevance of search results.

INTRODUCTION

In today's information age, search engines are bridges connecting users to the vast and constantly expanding sea of digital content. Google, a leader in search innovation, is dedicated to providing seamless and informative search experiences. This project tackles the challenge of improving search relevance for video content by utilizing video subtitles. By doing so, we aim to unlock video information and make it more accessible to a broader audience.

TYPES OF SEARCH ENGINES

- **Keyword-based Search Engine:** Matches exact words in your query to indexed documents. Simpler, but struggles with synonyms or different phrasings. It will match our search query to web pages that contains the same keywords, without considering the deeper meaning or context of those words.

- **Semantic-based Search Engine:** Understands the meaning and context of your search. Delivers relevant results even if phrased differently. It is far more efficient and provides better results in comparison to "***keyword-based search engines***".

WORKFLOW

There are two important steps that must be followed to implement search engine. They are as follows:

- **Ingesting the Documents**
  1. Data Sampling
  2. Data Preprocessing
  3. Document Chunking
  4. Text Vectorization
  5. Storing Embeddings in Chromadb
- **Retrieving the Documents**
  1. Take the user's search query and preprocess the query (if needed).
  2. Create query embedding.
  3. Using cosine distance, calculate the similarity score b/w embeddings of documents and user query.
  4. Returns the most relevant candidate documents as per user's search query.

INGESTING THE DOCUMENTS

1. Data Sampling

We started with a ***"eng_subtitle_database.db"*** file. First of all, we used sqlite3 to connect to the database and extract the dataset. After that we have to load the dataset as a pandas dataframe to for further processing. Though, the subtitle_content available is encoded with ***"latin-1"*** encoding, we have decoded it first. The first issue, we encountered was to sample the data because there were around 82k subtitle available, we have just taken a 30% of the actual available content.

INGESTING THE DOCUMENTS

2. Data Preprocessing

We created a special function that performs several tasks on the text data. This includes:

- ✓ Removes timestamps that track when the data was collected.
- ✓ Converts all the text to lowercase for consistency.
- ✓ Eliminates common words (stopwords) that don't hold much meaning.
- ✓ Gets rid of anything that isn't actual text (like symbols or punctuation).

Additionally, due to system configuration limitations we are have trouble with the dataset, so we have stored the cleaned dataset after data processing as parquet file for faster and better memory optimization. Parquet files are memory efficient and faster than csv to load and retrieve the data.

INGESTING THE DOCUMENTS

3. Document Chunking

- ✓ When dealing with massive amounts of text data, a technique called "embeddings" can be used to represent the text in a way that machines can understand. However, this process can sometimes lose important information, especially with very long documents.
- ✓ To tackle information loss in embedding large text chunks, a technique called "chunking" is employed.
- ✓ By splitting documents into smaller, manageable pieces, we can create more accurate text embeddings as machines can better understand the content.
- ✓ This approach is like chopping large documents into bite-sized pieces for better comprehension.

INGESTING THE DOCUMENTS

4. Text Vectorization

To bridge the communication gap between humans and machines, we leverage a technique called "text vectorization." This process essentially translates words into numerical representations machines can understand and analyze. Different techniques offer varying levels of sophistication for this conversion:

- ✓ **BOW/TF-IDF:** This method creates a basic fingerprint by counting word occurrences. It excels at identifying exact matches, making it suitable for keyword-based search engines where finding specific terms is paramount.
- ✓ **BERT Embeddings (utilized in this project):** This more advanced approach delves deeper, considering not just word frequency but also meaning and context. BERT excels in semantic search, where grasping the overall intent of a query is crucial. By employing BERT embeddings, we empower the search engine to understand the nuances of human language, leading to more relevant and insightful video content retrieval.

INGESTING THE DOCUMENTS

5. Storing Embeddings in Chromadb

Chromadb is a special kind of database designed specifically for handling these vector representations.

Chromadb isn't just a storage locker for our embeddings. It also allows us to:

- ✓ **Enrich Embeddings:** Add metadata like timestamps or categories for more informative data.
- ✓ **Targeted Retrieval:** Search embeddings based on metadata, just like filtering library books.
- ✓ **Speed vs. Persistence:** Choose in-memory storage for blazing-fast access (but temporary) or hard drive storage for permanence (slower access). This flexibility optimizes for your needs.
- ✓ We can filter and search through the vast collection of embeddings based on their associated metadata, enabling a more targeted and efficient retrieval process.

RETRIEVING THE RESULTS

- **Retrieving the Documents**

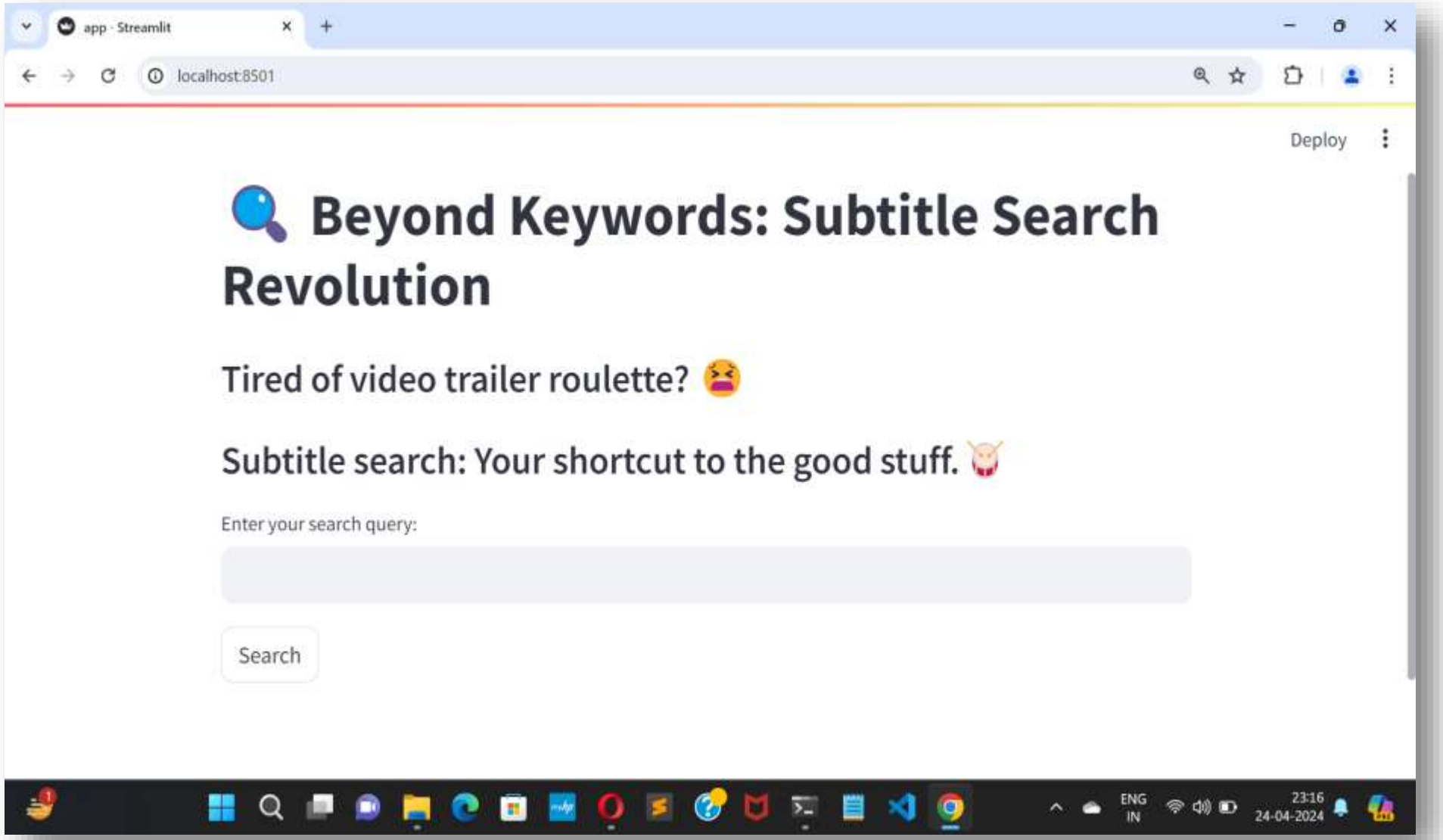
  1. Take the user's search query and preprocess the query (if needed).

  2. Create query embedding.

  3. Using cosine distance, calculate the similarity score b/w embeddings of documents and user query.

  4. Returns the most relevant candidate documents as per user’s search query.

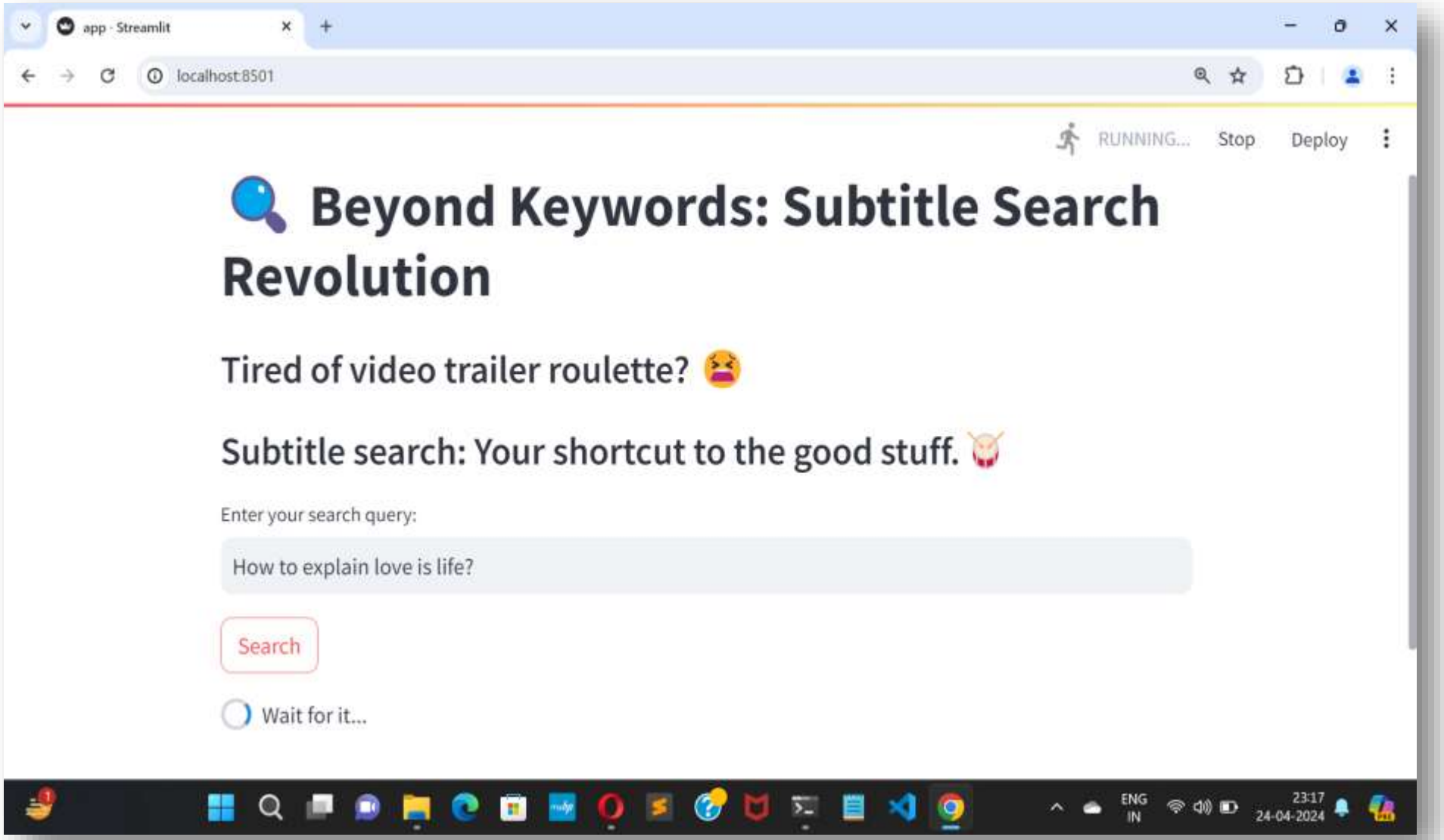
RESULTS (SCREENSHOTS)



Homepage

INNOMATICS
RESEARCH LABS

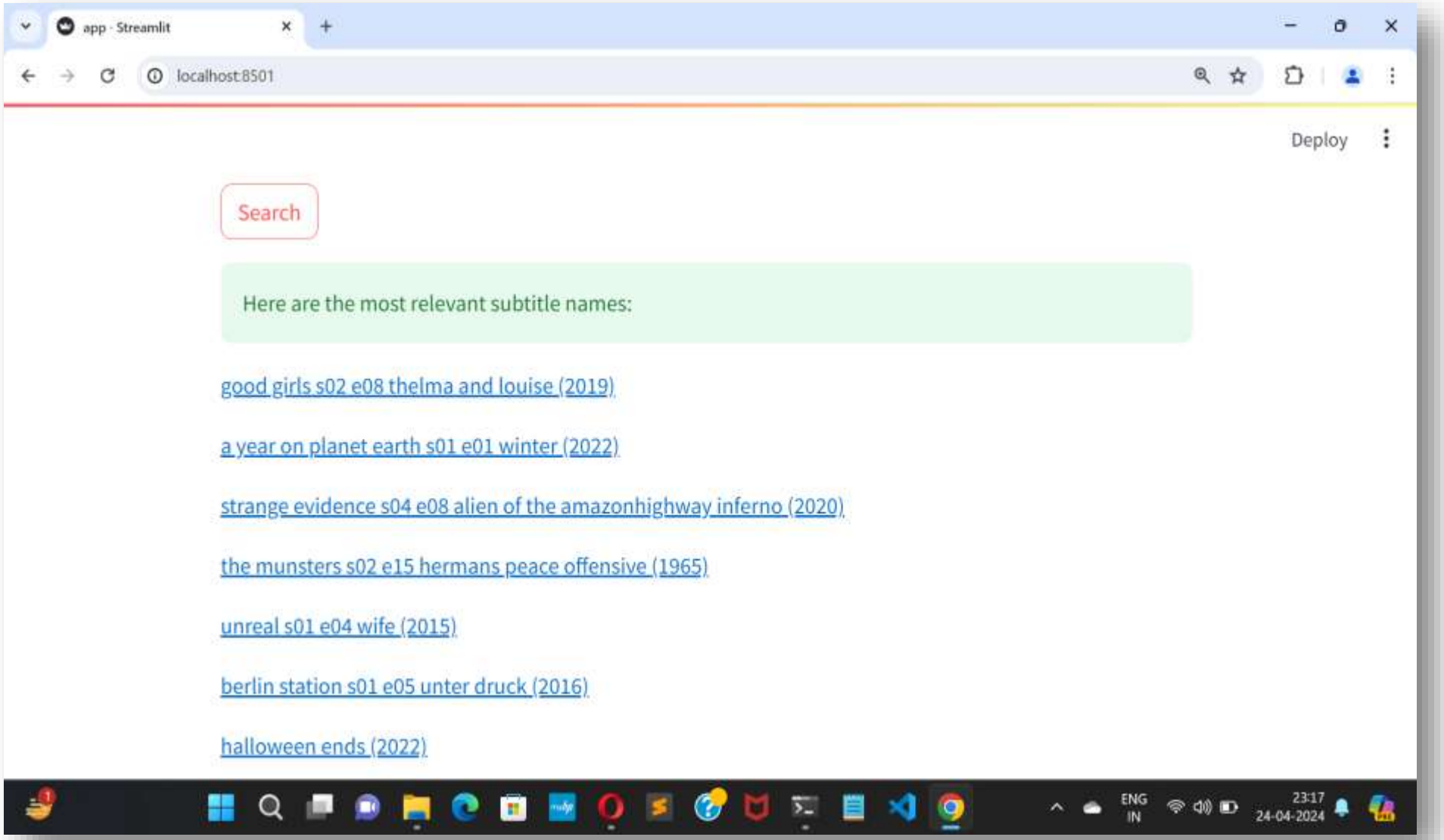
RESULTS (SCREENSHOTS)



Querying

INNOMATICS RESEARCH LABS

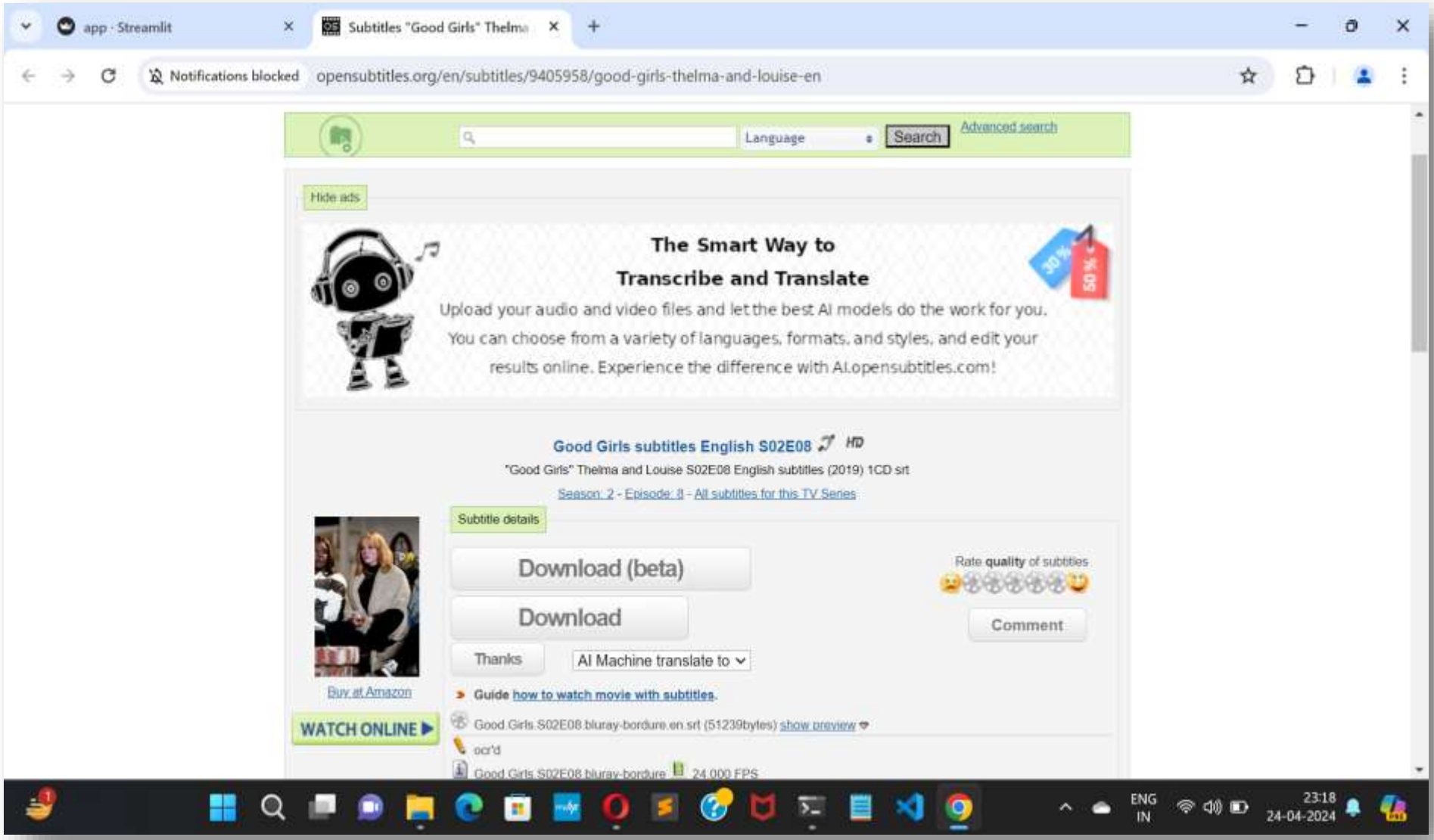
RESULTS (SCREENSHOTS)



Search Results

INNOMATICS RESEARCH LABS

RESULTS (SCREENSHOTS)



Subtitle Page

INNOMATICS
RESEARCH LABS

SEARCH ENGINE DEMO & CODE

- **Search Engine Demonstration**

  To see the demonstration, please go to the youtube link below :

  https://youtu.be/CPh3WonOLPw?si=UiSxvTMmd_BPFj7t

- **Github**

  Check out the code at Github. Follow the url : https://github.com/vjabhi000985/Subtitle_Search_Engine/

CONCLUSION

By leveraging the power of natural language processing and semantic search, this approach has the potential to revolutionize video search. By focusing on the rich tapestry of information within subtitles, we can unlock a new level of accuracy and relevance in search results. This not only enhances the user experience for everyone, but also empowers a wider audience to access the wealth of knowledge housed within video content. This project represents a significant step forward, paving the way for an even more inclusive and informative future of video search.

THANK YOU

INNOMATICS RESEARCH LABS