TACTICAL TRENDS DETAILED REPORT

GOLD PROBLEM STATEMENT:

Develop an innovative AI based news aggregator application customized explicitly for strategic & geopolitical significance (Defence centric).

OBJECTIVE:

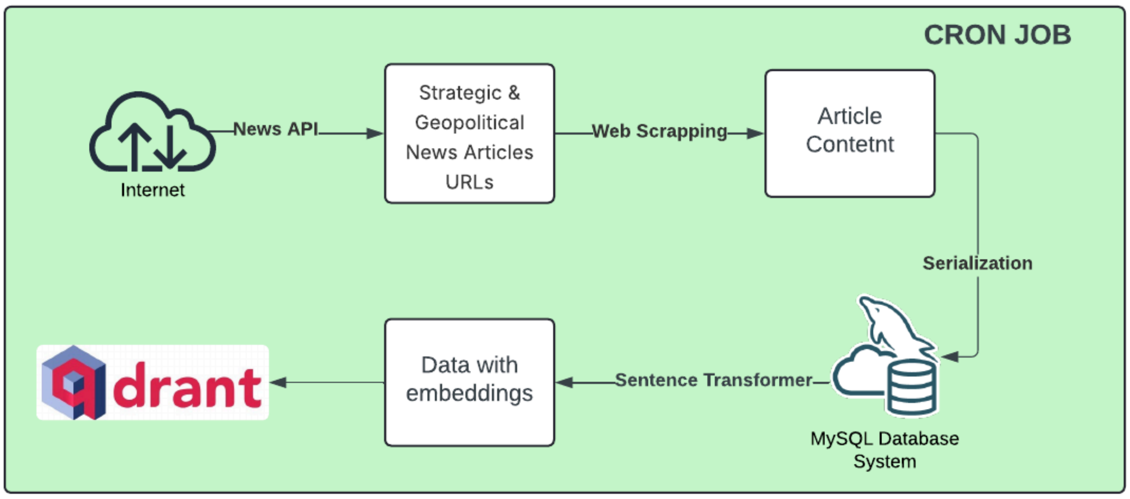
Tactical Trends curates strategic and defense-related content, providing users with personalized, relevant news from domestic and international sources. Utilizing machine learning and NLP, the platform delivers concise summaries, insightful analysis, and tailored recommendations based on user preferences, ensuring an enhanced and focused news consumption experience.

Contents:

1. Features

News Aggregation Process:

1. **Fetching News via News API**:
   A **cron job** is set up to automatically fetch news articles from the **News API** at regular intervals. The API retrieves articles based on specific filters like keywords, categories, or dates to ensure the fetched content is relevant to the application's focus (e.g defence, geopolitical and strategic etc.).
2. **Storing Fetched News in MySQL**:
   The fetched news articles are then stored in a **MySQL** database. Each article is saved with relevant details, such as the title, URL, content, publication date, and source. This allows for efficient querying and further processing.
3. **Vectorizing Articles for Search**:
   After storing the news in MySQL, the content of each article is **vectorized**—converted into numerical representations (vectors) using a suitable embedding model. This step is crucial for enabling advanced searches, such as similarity searches, and facilitating content recommendations.
4. **Storing Vectors in Qdrant**:
   Once vectorized, the article representations are stored in the **Qdrant vector database**. This enables fast and efficient retrieval of news articles based on semantic similarity, enhancing the search and recommendation capabilities of the application.



Keyword Search

1. **User Input**:
   The user submits a search query containing one or more keywords.
2. **Keyword Extraction**:
   The query is split into individual keywords.
3. **Similarity Search**:
   Each keyword is checked for similarity in the vector database (e.g., Qdrant) to find contextually similar terms, even if the exact word isn't present.
4. **Logging User Activity**:
   If a similar keyword is found (with a high similarity score), the system updates an existing record in the SQL database, which tracks how often that keyword has been searched and logs the latest search time.
   If no similar keyword is found, the keyword is added to both the SQL database (for logging) and the vector database (for future searches).
5. **Article Search**:
   The system searches for articles related to the keywords by performing a similarity search in the vector database.
6. **Fetching Articles**:
   If similar articles are found, they are retrieved from the MySQL database, ordered by publication date, and returned to the user.
7. **Handling No Results**:
   If no matching articles are found, the user is informed that no relevant articles are available.
8. **Fallback to Default List**:
   If no search query is provided, the system displays all articles by default.

Content-Based Recommendations:

1. **Fetching Top User Reviews**:
   The system fetches the top 5 articles the user has reviewed with a rating of 3 or higher and a positive sentiment. These articles are sorted by highest rating and most recent review.
2. **Fetching Top User Search Keyword**:
   It also fetches the top 5 most frequently searched keywords by the user
3. **Processing Top Reviews (60% weight)**:
   For each of the top-reviewed articles, the system fetches up to 4 similar articles from the vector database using similarity search . Each of these similar articles is assigned a 60% weight in the recommendation counter.
4. **Processing Top Searches (40% weight)**:
   For each of the top-searched keywords, the system finds up to 4 related articles using similarity search. These articles are added to the counter with a 40% weight.

5. **Sorting Articles by Weight**:

The article IDs are sorted based on their weighted occurrence in descending order, meaning the articles with the highest combined weight (from reviews and searches) will appear first.

6. **Excluding Already Reviewed Articles**:Articles that the user has already reviewed are excluded from the final recommendations to avoid redundant content.

Sentiment Analysis:

1. **Sentiment Analysis Pipeline**:
   The function initializes a sentiment-analysis pipeline using the HuggingFace transformers library. The model used for sentiment analysis is **distilbert-base-uncased-finetuned-sst-2-english model.**
2. **Content Splitting**:
   The full article content is split into smaller chunks, with each chunk having a maximum length of 512 tokens to ensure it can be processed by the model, as most transformer-based models have input size limitations.
3. **Sentiment Analysis on Chunks**:
   Each chunk of the article content is analyzed separately using the initialized sentiment analyzer. The results for each chunk are stored in a list.
4. **Combining Results**:
   The sentiment results from all chunks are combined. These results can either be aggregated or returned as a list, depending on the desired output format.

Summarization:

1. **Summarization Pipeline Initialization**:
   The function initializes a summarization pipeline using the Hugging Face Transformers library. The model used for summarization is the **sshleifer/distilbart-cnn-12-6 model**.
2. **Content Splitting:**
   The full content is split into smaller chunks, with each chunk having a maximum length of 1024 tokens to ensure it can be processed by the model, as many transformer-based models have input size limitations.
3. **Summarization on Chunks:**
   Each chunk of the content is summarized separately using the initialized summarization pipeline. The summaries for each chunk are collected in a list.
4. **Combining Summaries**:
   The summaries from all chunks are combined into a final summary string, which is returned as the output of the function.

Word Cloud Generation:

1. **Article Content Retrieval:**
   Retrieve the article content
2. **Word Cloud Generation:**
   A word cloud is generated using the WordCloud class from the wordcloud library. The word cloud is configured with a width of 800 pixels, a height of 400 pixels, and a white background. The word cloud is created by processing the full_content of the article.

User Feedback:

1. **User Feedback Collection**:
   Users provide feedback on articles in two formats:
   Rating: A numeric score on a scale of 1 to 5.
   Text Review: An optional written review providing qualitative insights about the article.
2. **Sentiment Analysis**:
   The feedback, particularly the text review, is processed through a sentiment analysis pipeline.
   Utilizes a pre-trained sentiment analysis model (e.g., distilbert-base-uncased-finetuned-sst-2-english) to evaluate the sentiment of the provided text.
   The model categorizes the sentiment as positive, negative, or neutral, generating a sentiment score.
3. **Feedback Integration**:
   The collected ratings and sentiment scores are integrated into the recommendation system:
   Positive Feedback: High ratings (e.g., 4 or 5) and positive sentiment scores enhance the visibility of similar articles in future recommendations.
   Negative Feedback: Low ratings (e.g., 1 or 2) and negative sentiment scores can decrease the visibility of poorly rated articles or suggest alternative content to the user.
4. **Recommendation System Enhancement**:
   The feedback is used to adjust the underlying algorithms of the recommendation system:
   Weight Adjustments: Positive ratings and sentiment can increase the weights assigned to certain articles or keywords, while negative feedback can decrease their relevance in future recommendations.
   Dynamic Learning: The system continually learns from user interactions, refining the recommendations over time based on accumulated feedback.

2. Additional Features

Fake News Detection:

- **Fake News Detection Pipeline**:
  The function initializes a fake new detection pipeline using hugging face transformers
- **Content Splitting**:
  The full article content is split into smaller chunks, with each chunk having a maximum length of 512 tokens to ensure it can be processed by the model, as most transformer-based models have input size limitations.
- **NewsAnalysis on Chunks**:
  Each chunk of the article content is analyzed separately using the initialized sentiment analyzer. The results for each chunk are stored in a list.
- **Combining Results**:
  The sentiment results from all chunks are combined. These results can either be aggregated or returned as a list, depending on the desired output format.

Bookmarking System:

1. Users can bookmark articles they like
2. Access: Saved articles appear in a "Bookmarks" tab within the user's profile.
3. Management: Users can view, organize, and remove their saved articles in this tab.
4. Easy retrieval: Clicking a bookmark takes the user directly to the saved article.

Chrome Extension:

1. **Article Summary**:

   The extension extracts key points and generates a concise summary of the article.

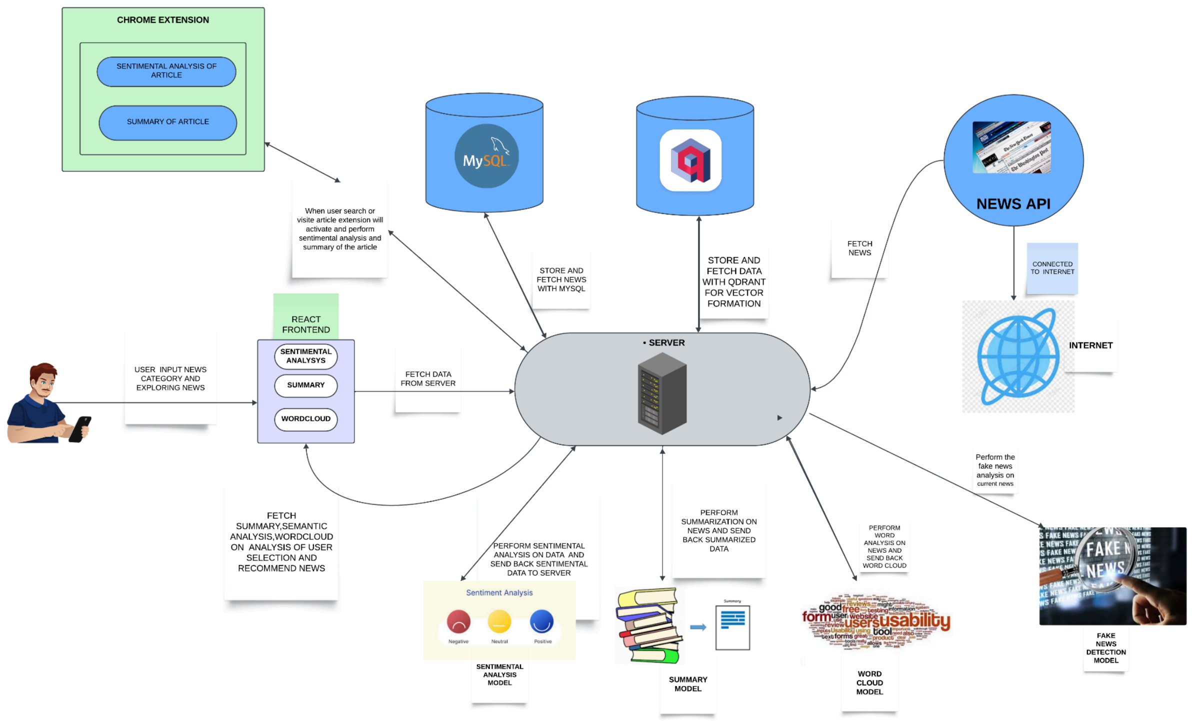
   Saves time by eliminating the need to read the entire article.

2. **Sentiment Analysis**:

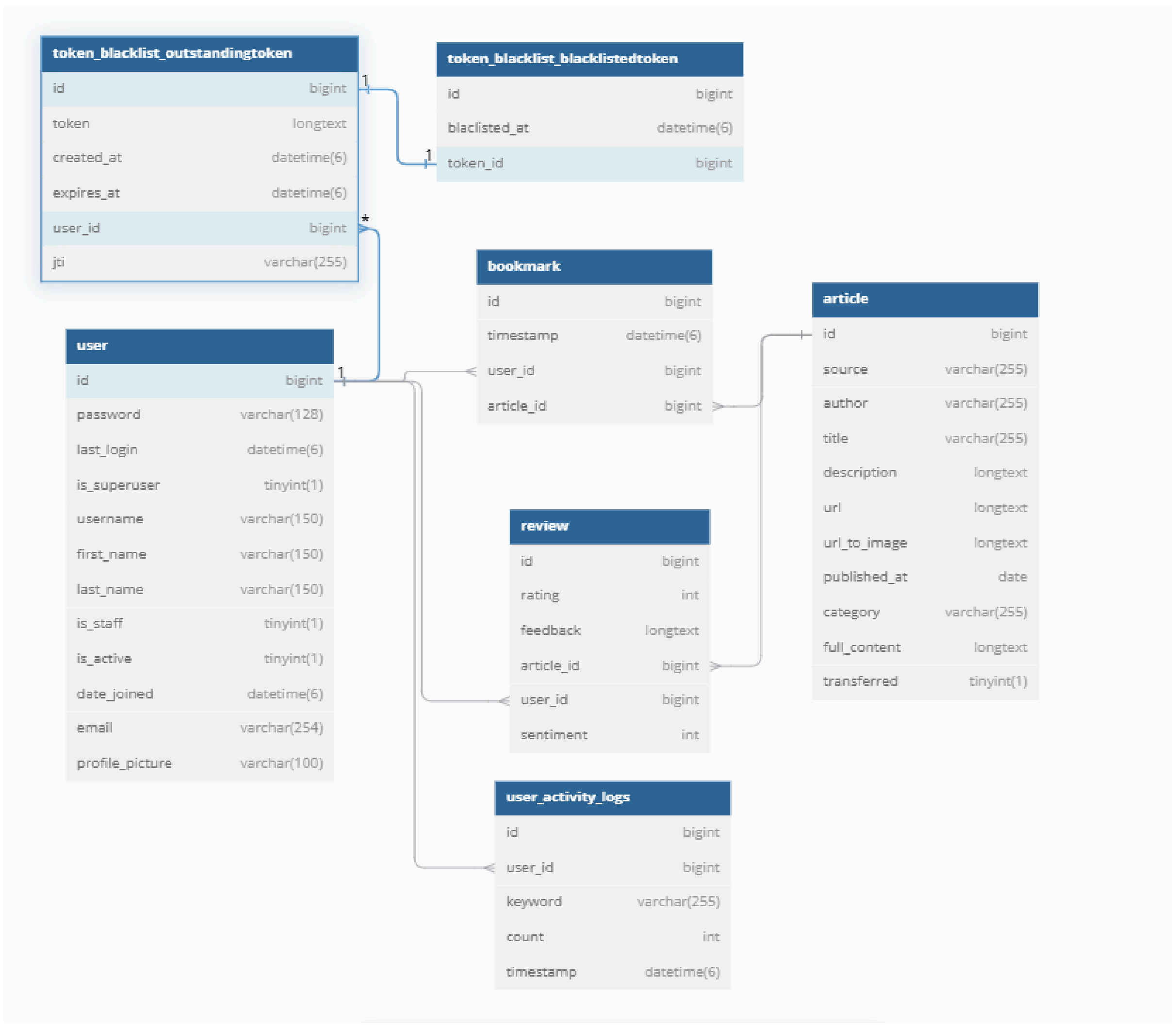
   Analyzes the article's tone (positive, negative, or neutral).

   Helps users understand the overall sentiment conveyed by the content at a glance

3.System Architecture


CHROME EXTENSION
SENTIMENTAL ANALYSIS OF ARTICLE
SUMMARY OF ARTICLE
When user search or visite article extension will activate and perform sentimental analysis and summary of the article
MySQL
STORE AND FETCH NEWS WITH MYSQL
STORE AND FETCH DATA WITH QDRANT FOR VECTOR FORMATION
FETCH NEWS
NEWS API
CONNECTED TO INTERNET
INTERNET
REACT FRONTEND
SENTIMENTAL ANALYSYS
SUMMARY
WORDCLOUD
USER INPUT NEWS CATEGORY AND EXPLORING NEWS
FETCH DATA FROM SERVER
• SERVER
FETCH SUMMARY,SEMANTIC ANALYSIS,WORDCLOUD ON ANALYSIS OF USER SELECTION AND RECOMMEND NEWS
PERFORM SENTIMENTAL ANALYSIS ON DATA AND SEND BACK SENTIMENTAL DATA TO SERVER
PERFORM SUMMARIZATION ON NEWS AND SEND BACK SUMMARIZED DATA
PERFORM WORD ANALYSIS ON NEWS AND SEND BACK WORD CLOUD
Perform the fake news analysis on current news
Sentiment Analysis
Negative
Neutral
Positive
SENTIMENTAL ANALYSIS MODEL
Summary
SUMMARY MODEL
WORD CLOUD MODEL
FAKE NEWS
FAKE NEWS DETECTION MODEL

4.Database Design



5. Links

Frontend:-
https://github.com/Virucodes/COEP-MINDSPARK-FRONTEND.git

Backend:-
https://github.com/swarup-2004/MindSpark-Hacathon.git

Chrome-Extension:-
https://github.com/Virucodes/News-Aggregation-Chrome-Extension.git