TWITTER DATABASE AND SEARCH APPLICATION

GROUP 17	
Srinivasaraghavan Sundar	ss4805
Udit Goel	ug42
Vijay Sivakumar	vs851
Vidyoth Sateesh	vs863
Github URL	srinivassundar98/DBMS_Project (github.com)
Github User Names -	https://github.com/srinivassundar98 https://github.com/goeludit/ https://github.com/VijaySivak https://github.com/Vidyoth7

TABLE OF CONTENTS

ABSTRACT

This report details the creation of a triune-database architecture for efficient management and querying of Twitter data, incorporating snowflake, mysql and mongodb stores to optimize storage for user and tweet information. A Python-based data ingestion system and kafka simulates real-time streaming, while an intelligent indexing enhances the data retrieval. The search application built using streamlit offers comprehensive query options with advanced drill-down features. A multilayered caching strategy used for the search application improves the performance of the system. Also, Performance evaluations via test queries demonstrate the system's efficiency, and the report addresses the impact of architectural decisions on response times and data relevance.

INTRODUCTION

The advent of social media has resulted in an explosive growth of data generation, with platforms like Twitter leading the charge. Twitter's user base generates a staggering volume of content each second, offering invaluable insights into public sentiment, market dynamics, and unfolding events. However, this wealth of data presents significant challenges in terms of storage, processing, and retrieval. Traditional data management solutions are often inadequate for handling the variety, velocity, and volume of Twitter data.

This report underscores the essential requirement for effective and scalable architectural solutions capable of storing, processing, and analyzing Twitter data streams in real-time. It investigates cutting-edge methodologies designed to handle the extensive quantities of data and support intelligent querying and retrieval processes that yield timely, actionable insights within a search application. The convergence of sophisticated data processing frameworks and advanced search functionalities is proposed to ensure a comprehensive solution adept at managing the complexity and expansiveness of Twitter data flows.

The discourse extends to the integration of distributed storage systems with refined data processing pipelines. This examination seeks to determine the orchestration necessary to enhance both performance and dependability. The ambition is to establish a formidable architecture that promotes the swift intake of high-volume Twitter data, implements effective storage strategies for expedited access and analysis, and introduces search applications that facilitate pertinent and user-friendly data navigation.

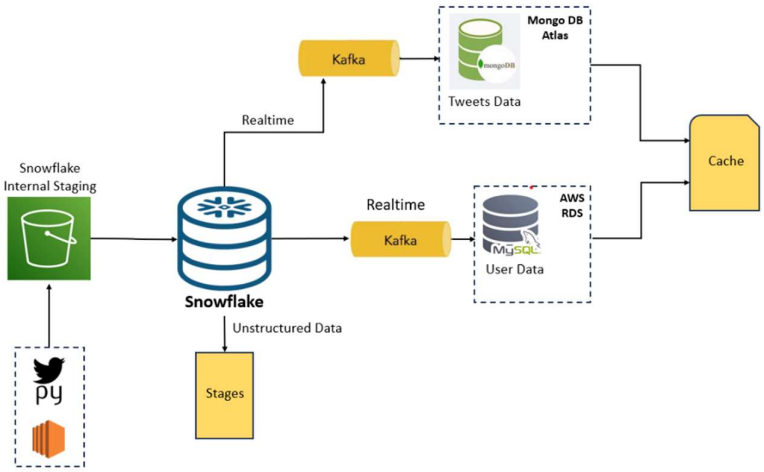
Employing innovative big data technologies and search algorithms, the proposed solution endeavors to reconcile the exigencies of rapid processing with the analytical depth required for thorough data examination. The architecture is not only tailored for the efficient capture and storage of Twitter data but also for distilling significant patterns, tracking sentiment evolution, and promptly responding to the derived insights. Additionally, the integration of machine learning models such as sentiment analysis, emotion detection, and topic modeling enrich the understanding of the data corpus.

DATASET

The core of the dataset is the tweet itself, which relates to topics around Corona. This information is paired with extensive metadata about the user who posted it, including their profile, follower statistics, and account details, providing context about the tweeter's social influence and credibility. The dataset also includes engagement metrics such as the number of retweets, replies, and favorites, which help gauge the tweet's reach and audience interaction. Additionally, it contains geographical data pinpointing where the tweet was posted, which can be vital for geospatial analyses of social media trends. This dataset is valuable for a variety of applications, including sentiment analysis, trend tracking, and demographic studies, offering insights into public perception and social media dynamics during the specified period.

METHODOLOGY

DATA STORAGE ARCHITECTURE

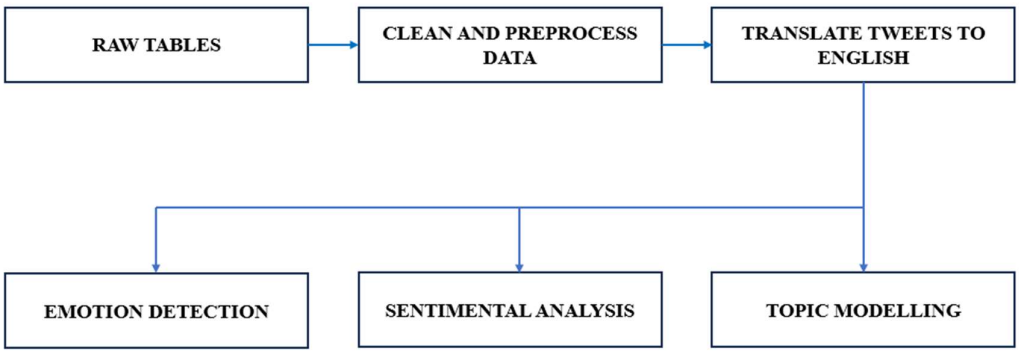


DATA INGESTION PROCESS

A Python script is utilized to process the Twitter dataset. This script is scheduled to run periodically using CronJob, that is new data is ingested into the system hourly. The Python script processes the data and outputs it as JSON files. The data is being converted into JSON, format suitable for transmission or storage that supports hierarchical structures. These JSON files are then sent to Snowflake's internal staging area. Staging areas are temporary storage spaces where data can be held before it is loaded into the data warehouse. Snowflake is a cloud-based data platform that offers data warehousing capabilities. Within Snowflake, the data is stored in a staging table designed for semi-structured data. Semi-structured data is

a form of data that does not conform to the tabular structure associated with relational databases but has some organizational properties that make it easier to analyse than unstructured data. Finally, the data from the staging table is moved to raw tables. Raw tables in a data warehouse are used for the initial storage of ingested data before it undergoes further processing or is moved to other tables for specific analytic purposes.

DATA TRANSFORMATION PIPELINE



The Tweets, Retweets and User Tables are the raw tables that are obtained from the previously discussed step in snowflake. This is the data from the source and contains unprocessed data in a structured format. This data may be inconsistent, contain errors, or missing data that are not immediately suitable for analysis.

Clean/Preprocess the Data: The next step involves cleaning and preprocessing the data. This is a crucial step in any data pipeline and it involves removing or correcting inaccuracies, dealing with missing values, and performing other data quality improvements to ensure the data is in a usable state for analysis.

Translate Tweets to English: Since the raw data includes tweets and retweets that are in various languages, this step involves translating them into English using the translator imported from googletrans module. This standardization is important for consistent analysis, especially when employing natural language processing techniques.

After the data has been cleaned and standardized, it moves on to the machine learning pipeline, which consists of:

- **Topic Modelling**: This analysis aims to identify the underlying themes or topics within the corpus of tweets. Topic modelling is an NLP task that discovers abstract topics within text, helping to understand the main subjects discussed across the tweets.
- **Sentiment Analysis**: Here, the pipeline includes an analysis to discern the sentiment behind each tweet—whether the emotions expressed are positive, negative, or neutral. This involves using pre-trained models to classify sentiment based on the text.

- **Emotion Detection**: This step goes beyond sentiment analysis by identifying specific emotions conveyed in the tweets, such as joy, anger, or sadness. This can add depth to the analysis by providing insights into the emotional responses associated with different topics or events.

SNOWFLAKE EGRESS

The methodology for data egress encompasses the migration or exportation of datasets from Snowflake to designated storage solutions, specifically MySQL and MongoDB databases. This process utilizes Amazon EC2 instances as intermediate agents, leveraging their scalable compute capacity in the cloud to facilitate a secure data transfer.

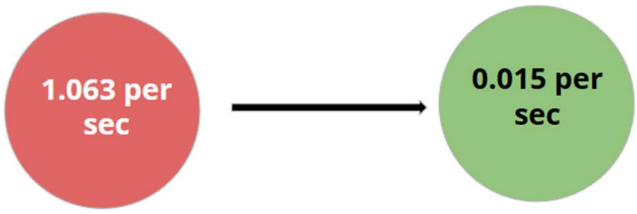
The strategic plan includes the integration of a Kafka pipeline, an advanced distributed event streaming platform renowned for its ability to manage substantial data volumes. The implementation of Kafka is anticipated to substantially augment the real-time data processing capabilities of the current architecture—a critical component for maintaining up-to-the-minute data flows.

Upon the successful implementation of this system, user-related information is earmarked for storage in the MySQL database, while tweets and retweets are allocated to MongoDB. Each database is hosted on its respective cloud service—AWS RDS for MySQL and MongoDB Atlas for MongoDB—ensuring reliable and scalable cloud support.

Complementing the storage architecture is a caching mechanism paired with the search application. This addition is engineered to optimize data retrieval speeds, effectively bolstering the performance and responsiveness of the system.

INDEXING

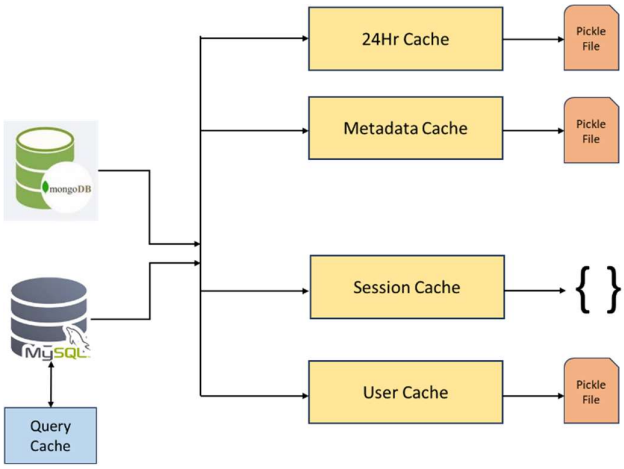
A query optimization on the MySQL is done using indexing for enhancing user data retrieval. The results are queried almost 70 times faster post using indexing.



The below image represents the query performance after indexing which is highlighted in red.

26	16:19:37	SELECT * FROM TWEET_STORE_FINAL.USER_DETAILS where ID='1144669339599200256'	1 row(s) returned	1.063 sec / 0.000 sec
27	16:19:51	CREATE INDEX idx_userid ON TWEET_STORE_FINAL.USER_DETAILS(id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.438 sec
28	16:20:12	select * FROM TWEET_STORE_FINAL.USER_DETAILS where ID='1144669339599200256'	1 row(s) returned	0.015 sec / 0.000 sec

CACHING



In the system implemented, caching plays a crucial role in enhancing performance by allowing for the rapid retrieval of frequently accessed data, effectively reducing the need to repeatedly read from disk. To cater to the diverse needs of data access and retrieval, a multi-faceted caching approach has been implemented that includes session caching, a 24-hour result cache, and summary-statistic cache.

Session caching is designed to be compact, with a typical size ranging from 5 to 10MB. This cache is ephemeral, existing only for the duration of a user session, and is cleared once the session is terminated. An LRU eviction policy is employed to manage the cache's limited capacity, where the least recently accessed items are discarded first. This mechanism ensures that active session data is retrieved swiftly and is stored using a dictionary format within the cache for efficient access and modification.

The **24-hour result cache**, as indicated by its name, stores items with a TTL of 24 hours. After this period, the data is deemed expired and is eligible for automated purging. The removal of outdated items is handled by a dedicated method, which cleanses the cache of stale entries. Should the cache reach its maximum capacity, it adheres to the LRU policy to make room for new data, prioritizing the retention of recently accessed items.

For data that sees frequent consumption, such as the most active users or the highest number of retweets,

we implement **summary-statistic caching**. The unique aspect of this caching strategy is its persistent nature, achieved by serializing JSON objects into a pickle file. This persistence ensures that there is no loss of cache state between program executions, thereby providing quick access to popular data without the need to regenerate or refetch it. The use of `OrderedDict` is common across all our caching techniques, maintaining the order of data insertion and facilitating the LRU eviction policy by keeping track of which items to discard first as new data comes in.

Together, these three caching strategies form a comprehensive system that delivers fast, reliable data access tailored to the specific temporal and operational requirements of our application, thereby ensuring optimal performance with minimal delays. It is aimed to enhance the caching mechanism by establishing a user cache that anticipates and preloads pertinent data based on insights obtained from user activity logs from the application.

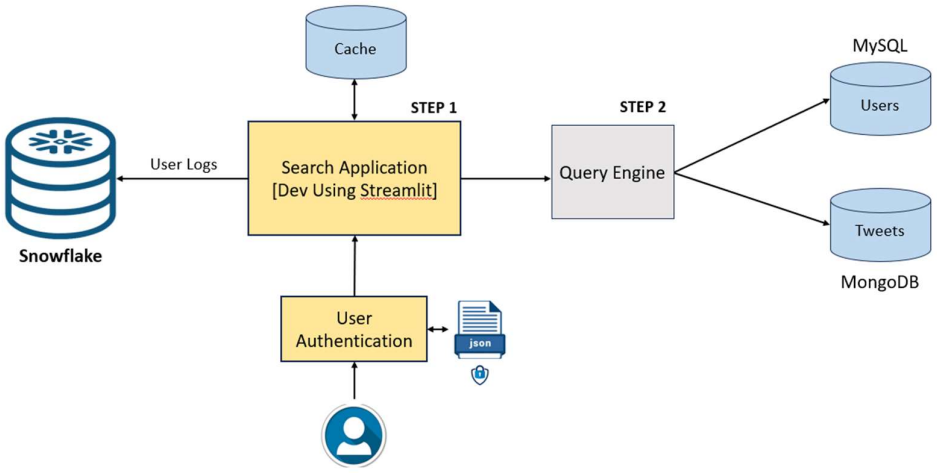
Below described is the query optimization in SQL and MongoDB through indexing and caching.

SL no.	Query	Optimization Method	Without optimization	With optimization
1.	SELECT SCREEN_NAME,followers_count FROM TWEET_STORE_FINAL.USER_DETAILS ORDER BY followers_count DESC LIMIT 5;	Caching	0.338 seconds	0.00012 seconds
2.	SELECT SCREEN_NAME,friends_count FROM TWEET_STORE_FINAL.USER_DETAILS ORDER BY friends_count DESC LIMIT 5;	Caching	0.154 seconds	0.00052 seconds
3.	SELECT SCREEN_NAME,listed_count FROM TWEET_STORE_FINAL.USER_DETAILS ORDER BY listed_count DESC LIMIT 5;	Caching	0.131 seconds	0.00054 seconds
4.	SELECT SCREEN_NAME,favourites_count FROM TWEET_STORE_FINAL.USER_DETAILS ORDER BY favourites_count DESC LIMIT 5;	Caching	0.116 seconds	0.001 seconds

5.	SELECT SCREEN_NAME,statuses_count FROM TWEET_STORE_FINAL.USER_DETAILS ORDER BY statuses_count DESC LIMIT 5;	Caching	0.100 seconds	0.0008 seconds
6.	SELECT LOCATION,COUNT(LOCATION) as COUNT FROM TWEET_STORE_FINAL.USER_DETAILS GROUP BY LOCATION ORDER BY COUNT DESC LIMIT 10;	Caching	0.211 seconds	0.0003 seconds
7.	select DISTINCT(SCREEN_NAME) AS USER_NAME,LENGTH(ENTITIES_USER_MENTIONS) - LENGTH(REPLACE(ENTITIES_USER_MENTIONS, ',', '')) AS USERS_MENTIONED, LENGTH(ENTITIES_HASHTAGS) - LENGTH(REPLACE(ENTITIES_HASHTAGS, ',', '')) AS HASHTAGS_MENTIONED FROM TWEET_STORE_FINAL.USER_DETAILS WHERE CREATED_AT>='Fri Apr 01 01:09:58 +0000 2016' AND CREATED_AT<='Wed Sep 30 23:44:06 +0000 2009' ORDER BY USERS_MENTIONED DESC, HASHTAGS_MENTIONED DESC;	Caching	1.758 seconds	0.003 seconds
8.	SELECT * FROM TWEET_STORE_FINAL.USER_DETAILS WHERE SCREEN_NAME='Kenkendall19';	Caching	0.220 seconds	0.003 seconds
9.	SELECT SCREEN_NAME,ENTITIES_HASHTAGS FROM TWEET_STORE_FINAL.USER_DETAILS;	Caching	0.990 seconds	0.001 seconds
10.	SELECT * from TWEETS_STORE_FINALUSER_DETAILS where ID = ‘1144669339599200256’ CREATE INDEX idx_userid on TWEET_STORE_FINALUSER_DETAILS	Indexing	1.063 seconds	0.015 seconds

	(id)			
11.	client.dbmsproject98_final.TWEETS.count_documents({"USER_ID": user_id})	Caching	0.54 seconds	0.004 seconds
12.	client.dbmsproject98_final.RETWEETS.find({'USER_ID': user_id}, {'ORIGINAL_TWEET_ENTITIES_USER_MENTIONS': 1})	Caching	0.75 seconds	0.006 seconds

SEARCH APPLICATION DESIGN

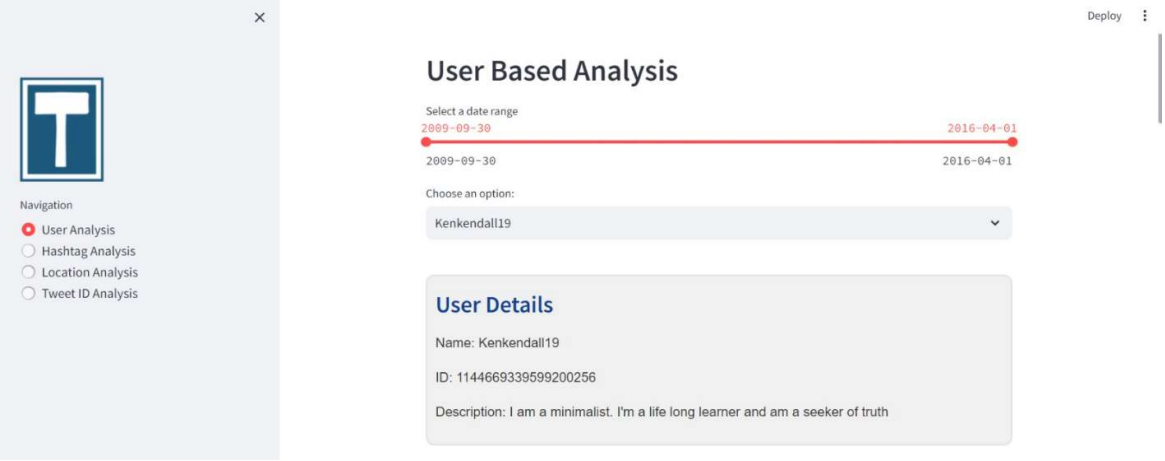


The search application is constructed utilizing Streamlit, an open-source application framework. The initial stage of user interaction with the application is underpinned by a robust user authentication system. This system secures the application by requiring users to log in with valid credentials, which are meticulously verified to prevent unauthorized access.

Data retrieval, a critical functionality within the application, is executed in two distinct phases. Primarily, the application seeks to retrieve the required data from a strategically implemented cache. This caching mechanism substantially accelerates data access, thereby curtailing the frequency of database queries. This not only enhances performance by diminishing load times but also elevates the overall user experience. When data requests are fulfilled from the cache, the frontend operations proceed without delay.

In instances where the cache does not contain the requested data, the backend comes into play, powered by a query engine that processes the application's search queries. This engine is integral to the application, equipped with business logic capable of deciphering user queries and procuring the appropriate data. The user-related information is stored within a MySQL database, while tweet and retweet data are housed in MongoDB. The query engine effectively liaises with both databases, procuring data as necessitated by user-initiated searches or specific commands.

Additionally, Snowflake is employed to maintain user logs that chronicle the spectrum of user activities and interactions with the application. Collectively, the components of the search application function in concert to deliver a streamlined and intuitive user interface, maintaining high standards of data management and security protocols.



The search application interface is tailored for the twitter data analysis. The application allows for a variety of search and analysis types. Also, the navigation panel on the left provides a simple and intuitive method for switching between different types of analysis. Different search tabs are -

- User Analysis: Users can search for specific Twitter users to gather detailed analytics. The 'User Details' demonstrates that once a user is selected, details such as username, ID, and their self-description are displayed.

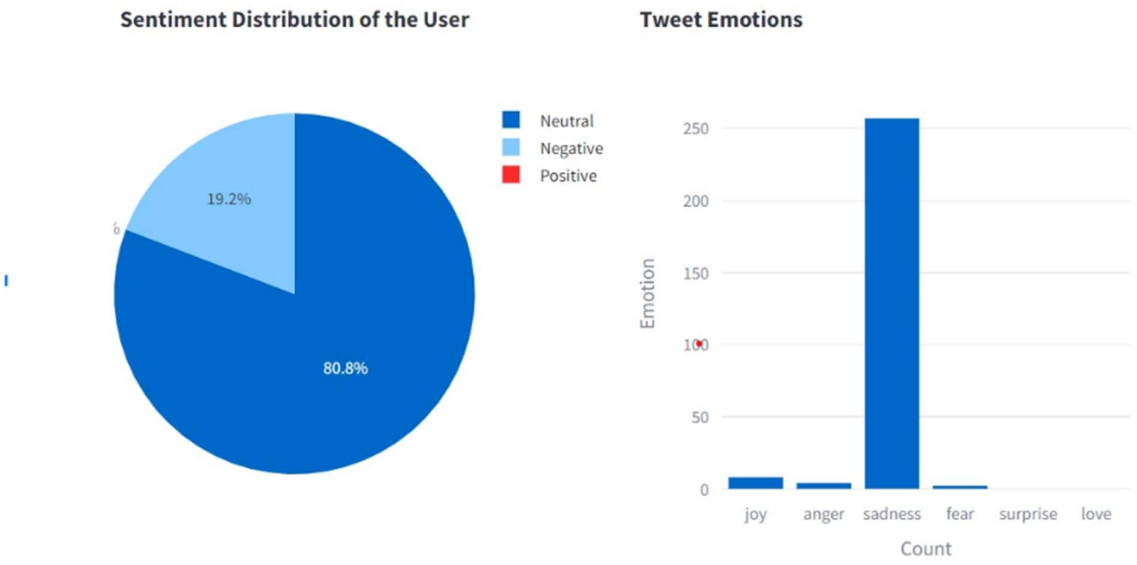
- Hashtag Analysis: This feature enables users to search and track the usage and popularity of specific hashtags over time, which can be critical for understanding trends and the spread of information or campaigns on social media.

- Location Analysis: This function allows users to search and analyse tweets based on location data, which can be useful for geo-targeted marketing, understanding regional discourse, and more.

- Tweet ID Analysis: By analysing specific tweet IDs, users can drill down into individual tweets to understand engagement, reach, and the tweet's content for closer examination.

Another drill down feature is that while selecting either a user or tweet id or hashtag is that one can select a date range, providing the ability to analyse activity within specified timeframes.

The drill-down capability is noticed in all the 4 panels. For instance, selecting a user allows for drilling down into their tweet history (most popular tweets), retweet patterns (Top 10 retweets), sentiment of the tweets, user interaction over time, and other engagement metrics.



CONCLUSION

The development of a comprehensive Twitter Data Search application has been successfully completed from the ground up. This robust tool is equipped with sophisticated drill-down features that enhance data analysis capabilities. It includes a suite of visual analytic options, such as assessments of the top 10 most retweeted posts, analyses of the most popular tweets, and evaluations of sentiments and emotions associated with the tweets. Additionally, the application conducts thematic examinations to identify prevalent topics, thereby facilitating a deeper understanding of content trends and the propagation patterns of information across social media platforms. Location analysis is another integral feature of the application, providing strategic insights beneficial for geo-targeted marketing and in-depth comprehension of regional communications.

In concluding the evaluation of our project, we acknowledge certain limitations that are present within the current framework. The system faces complexity in maintenance and monitoring, potentially affecting the ease of managing its operations over time. Moreover, it does not possess real-time data processing capabilities. The project also currently operates without a robust authentication system, an aspect that could be fortified to ensure heightened security and access management. Additionally, the absence of multithreading for cache optimization is noted as a limitation, which, if addressed, could significantly improve the performance of our caching strategy. Each of these identified limitations provides a clear direction for future enhancements and underscores our commitment to continuous improvement.

Looking ahead, the application is poised for further enhancements, notably the incorporation of Kafka to enable real-time data streaming. Moreover, an improved caching strategy is in consideration, intending to utilize historical user logs to refine system performance significantly. Also, an implementation of a better authentication system along with multithreading capabilities for the 4 caching strategies is planned. These advancements are aimed at fortifying the application's efficiency and its capacity to deliver immediate, data-driven insights.

REFERENCES

[1] srinivassundar98/DBMS_Project (github.com) – Github Link of the project code uploaded.

[2] – Github Link of the project code uploaded.

[3] Kharde, Vishal & Sonawane, Sheetal. (2016). Sentiment Analysis of Twitter Data: A Survey of Techniques. International Journal of Computer Applications. 139. 5-15. 10.5120/ijca2016908625.

[4] Rasul, Hakar & Jumaa, Alaa. (2022). Real-Time Twitter Data Analysis: A Survey. UHD Journal of Science and Technology. 6. 147-155. 10.21928/uhdjst.v6n2y2022.pp147-155.

[5] Maesaroh, S., Gunawan, H., Lestari, A., Tsaurie, M.S.A., & Fauji, M. (2022). Query Optimization in MySQL Database Using Index. International Journal of Cyber and IT Service Management (IJCITSM), 2(2), 104-110 retrieved from https://iiastjournal.org/ijcitsm/index.php/IJCITSM/article/view/84

[6] A Qualitative Study of Application-Level Caching | IEEE Journals & Magazine | IEEE Xplore- Mertz, Jhonny & Nunes, Ingrid. (2020). A Qualitative Study of Application-level Caching.

[7] Connect to Snowflake | Snowflake Documentation

[8] MongoDB Documentation

[9] MySQL :: MySQL Documentation

[10] Bogdanowicz A, Guan C (2022) Dynamic topic modeling of twitter data during the COVID-19 pandemic. PLoS ONE 17(5): e0268669. https://doi.org/10.1371/journal.pone.0268669

[11] [1901.08458] Emotion Detection and Analysis on Social Media (arxiv.org) Gaind, Bharat & Syal, Varun & Padgalwar, Sneha. (2019). Emotion Detection and Analysis on Social Media.

APPENDICES

The successful execution of the Twitter Data Search application project was a testament to the collaborative efforts of a dedicated team, each member bringing specialized expertise to the table. Srinivasaraghavan spearheaded the development of the search application, laying the groundwork for its functionality and also extended his contribution to the creation of intricate visualizations that enhance the analytical features of the tool. The critical task of loading data into Snowflake and architecting the transformation pipeline was jointly managed by Srinivasaraghavan and Vidyoth, ensuring a seamless flow of data through the system. Vidyoth's role was further emphasized in the integration of live datasets into the S3 Bucket from Twitter's API, establishing a foundation for real-time data analysis. Udit was instrumental in channelling this information into MongoDB and MySQL, focusing on robust database storage solutions. The optimization of database storage, a pivotal aspect for performance, was a collaborative endeavour tackled by both Udit and Vijay, maximizing efficiency and data retrieval speeds. Vijay's expertise was further demonstrated in the caching implementation, significantly contributing to the system's responsiveness and user experience.