



## **Twitter Search Application - Team 08**

**[Github Link](#)**

### **Course: 16:954:694:01 - Data Management for Advanced Data Science Applications**

#### **Team Members:**

Prashanth Aripirala ([Prashanth1998-18](#))

Anirudh Gaur ([Anirudh1308](#))

Rohit Macherla ([RohitMacherla3](#))

Rishik Shekar Salver ([rishik6561](#))

**Department of Statistics & Data Science**

**Rutgers University**

**New Brunswick, NJ**

Supervised by

Prof. Ajita John

**Note: Presentation to Cohort is completed (Cohort -3)**

## 1. Introduction

Twitter is one of the most popular social media platforms in the world, with millions of active users posting tweets on a daily basis. The Twitter search application harnesses the power of Twitter's underlying vast database to provide users with a simple yet powerful tool to quickly and easily find the most relevant information on any topic. This version of the Twitter search application allows users to search for a tweet based on username, hashtag, or part of a tweet with further drill-downs such as tweets related to a particular search available for each search case. This project aims to explore the design of various data storages and retrieval schemas, usage of various database technologies and the appropriate division of datasets for efficient data storage and. Additionally, the project includes the development of a caching model to facilitate faster data retrieval. The report will delve into the dataset and its storage, the design of the search application, the caching mechanism, the search queries implemented and present the results.

## 2. Dataset & Exploratory Analysis (By Anirudh Gaur)

It is always a good idea to get an idea of the dataset provided. Both the corona-out-2 and corona-out-3 were in JSON format. Upon performing a quick analysis it was found out that approximately 134k documents were present in the files. Some more insightful information found during data exploration and analysis was that all the tweets belonged to 2 different dates which lie during April 2020 (Covid period). Upon doing a sentiment analysis of all the tweets' text, the net sentiment can out to be 0.02 which depicts an overall negative skewness. The number of unique users in the dataset was approximately 108K and the distribution of the type of tweet can be seen in Figure below.

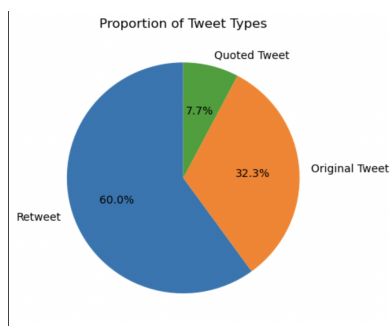


Fig 2.1: Distribution of tweets

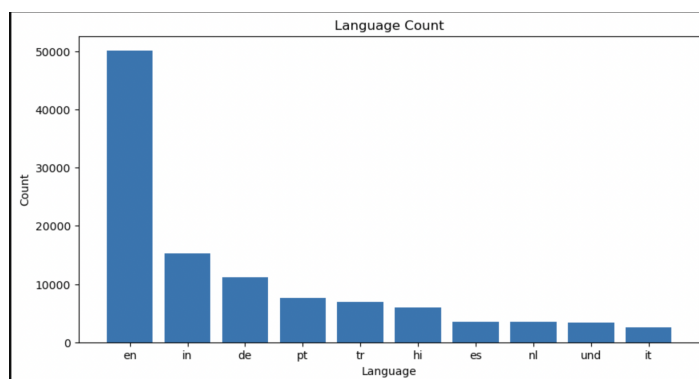


Fig 2.2: Top 10 languages of tweets

Since each document in the dataset contained information regarding both the user and the tweet, it was decided that the fields related to users and fields related to tweets would be kept in a separate datastore. Analysis regarding the top 10 most common languages used in tweets can be seen in Figure 2.2 above. For more analysis and visualizations of the dataset, the GitHub repository can be visited which has a file Data\_Analysis.ipynb containing the EDA.

### 3. System Architecture

System Architecture is a critical aspect that needs to be defined, at least tentatively, before commencing the development of any application, and was made sure to prioritize this step in our project. The system architecture establishes the overall flow of the application, including the key components and their coordination within the bigger picture. Given the various designs and functionalities that could be implemented using our dataset, it was chosen to prioritize the functionalities that would be most relevant to a typical Twitter user. After careful consideration and several iterations, Figure 3.1 below illustrates our final architecture and its key functionalities.

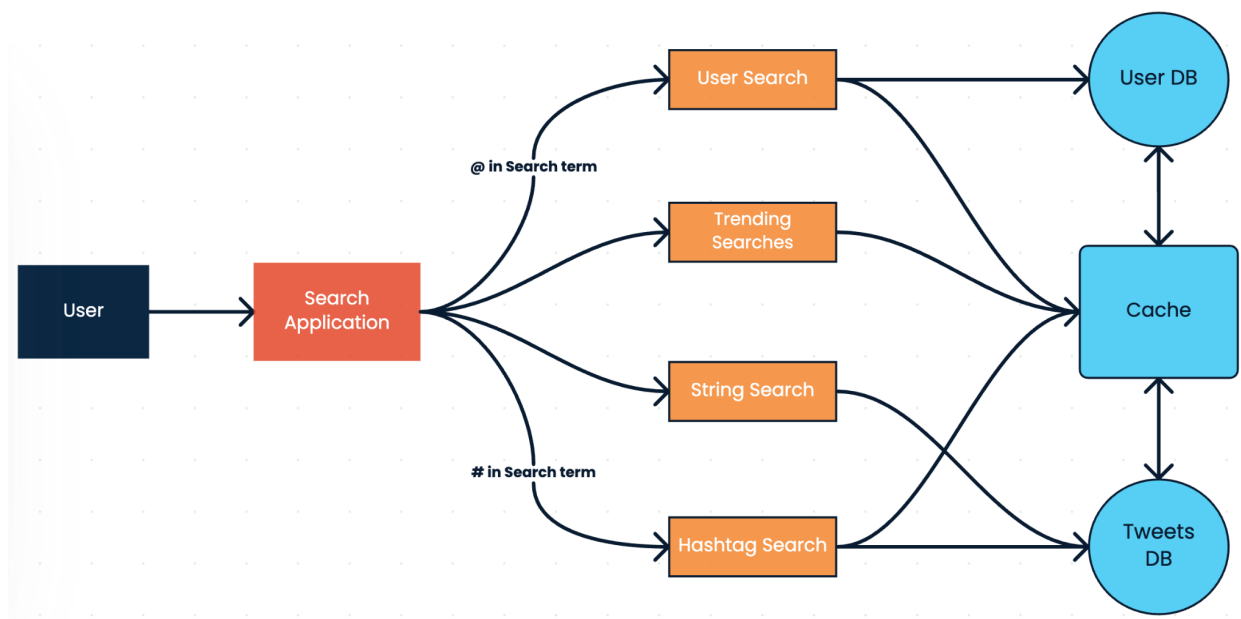


Fig 3.1 System Architecture

The Search Application's user interface, built using FLASK, a Python web framework, serves as the primary means for users to interact with the application and route between multiple URLs for various types of searches. Three core functionalities were implemented: searching by User, Hashtag, and Search String/Term, as well as providing users with access to get a quick look at the Trending Searches (Users, Hashtags, Tweets).

To ensure efficient search performance, a caching system with a Python dictionary as the primary data structure was built. When a user initiates a search(apart from the string/text search), the system first checks the cache for any matching results. Only if no match is found, the system will retrieve the results from the database. While displaying them to the user, simultaneously they are also stored in the cache for future retrieval. This approach significantly reduces the time and cost of fetching results from the database, ultimately resulting in faster and more efficient search performance.

Both files were processed to extract the User information and the tweet information. User Information was stored in a relational database MySQL because of many reasons which include that it is Open-Source, Scalable, and has a huge community support. Since users' database is not write-intensive unlike tweets data, a relational database would be an appropriate solution. The entire Twitter application runs around its tweets, reading and writing millions of them every single day. A traditional database cannot serve this purpose, so it was evident that a non-relational database with high performance handling the intensive read/writes, and flexibility in terms of schema paired with scalability, was required. MongoDB was the solution that also has huge community support online. A detailed walkthrough of each of the above-discussed components would be discussed in the further part of this report.

#### **4. Data Storage**

Data is a key aspect of every application because without the data any application would be meaningless. Hence storage and retrieval of data are significant components of an application, especially in applications that have a huge user base and millions of active users reading or writing. As discussed in the previous section, User and tweet data extracted are stored in two different database systems based on their usage.

#### 4.1. User Data Storage (By Prashanth Aripirala)

In this section, the design strategy and how the user data was extracted and transferred using Python to a MySQL database will be discussed. Because it is safe, dependable, and capable of handling both flexibility and scalability in resource management, MySQL is regarded by professionals in the industry as a relational database tool of the highest caliber. The necessary libraries, including "mysql.connector" and "json," were set up and loaded before the extraction and loading of tweets data into a MySQL database. Once these libraries were set up, a connection to the MySQL server was made, and a cursor was made to start building the database. Following the creation of the users table within the database with a number of fields, including but not limited to ID, name, location, and verification status, the connection was made using the credentials to the database. To extract and load the user data into the database, two functions were defined. The corona-out-3 and corona-out-2 files were passed to the **loadData()** function which reads line by line to extract each JSON object. Here when considered a tweet, which can be a source tweet or a retweet, or a quoted tweet. In any case, the user details of the author who created it has to be stored. So, the user part of the tweet was extracted and passed to the second function **UserInsert()**. This function extracts the ID from the user object passed, then checks if the user already exists in the database. If the user is not present in the database, the corresponding values were inserted into the database. In the case of a retweet/quoted tweet, the JSON object had a "retweeted status"/"quoted status" key containing the details of the source tweet to which the current tweet is a retweet/quoted tweet, the user object from the "retweeted status"/"quoted status" was extracted and passed to the UserInsert() function to perform a similar operation as discussed above. Handling duplicates is utterly important because there is a high possibility of users having many tweets/retweets/quoted tweets. To handle this efficiently, instead of reading all the user details at once before checking whether or not the user details already exist, the user details were extracted from the JSON only after confirming that the user details are not present in the database, which improved the overall time consumed to load the data into

the table. After processing both the given files and extracting the user information as discussed in the Users Data Load section, 108k unique users were identified and 8 different attributes were stored as shown in the figure below.

Column	Type	Default Value	Nullable	Character Set
◆ id	varchar(255)		NO	utf8mb4
◆ name	varchar(255)		YES	utf8mb4
◆ screen_name	varchar(255)		YES	utf8mb4
◆ verified	tinyint(1)		YES	
◆ followers_count	bigint		YES	
◆ friends_count	bigint		YES	
◆ created_at	datetime		YES	
◆ location	varchar(255)		YES	utf8mb4
◆ tweets_count	bigint		YES	
◆ Description	text		YES	utf8mb4

Fig 4.1.1: Attributes in the Users Table

After loading the data, and with the architecture and functionalities in mind, Indexing was performed on two columns, as shown in the below figure. The Primary index on the ID column is by default created by MySQL and the index on the name column was created to support our searches more efficiently. Both the indices are of the type Binary Tree.



Indexes in Table				
Visible	Key	Type	Uniq...	Columns
<input checked="" type="checkbox"/>	 PRIMARY	BTREE	YES	id
<input checked="" type="checkbox"/>	 idx_users_name	BTREE	NO	name

Fig 4.1.2: Indices in Users Table

4.2.   **Tweets Data Storage (By Rohit Macherla)**

Given the evident JSON structure of the tweets, it makes sense to store the tweets' data in a non-relational database such as MongoDB. MongoDB drastically

decreases the complexity of tweet storage, search, and recall, doing so without the requirement for a tweet parser. It is most suited for storing tweets due to its dynamic structure because not every tweet will have the same field. In order to load the data into MongoDB using python, the essentials are installing MongoDB client and necessary python packages such as pymongo. First, using pymongo, connection was established to the database and then using different function that perform specific tasks such as **insert\_tweet()** - inserts a tweet into the database, **find\_tweet()** - check if the tweet if it already exists in the database to avoid duplicate entries, **retweet\_update()** - updates the retweets count every time a retweet encountered for a tweet, **process\_tweets()** - reads the input json files line by line, check for duplicates and inserts both tweets and retweets, both tweets and retweets were processed and stored in the database. About 134000 unique tweets and retweets were extracted as shown in Figure 4.2.1 The data was loaded in such a way that every retweet has a field to capture the source tweet thereby creating a link for any drill-through features of the search application (Figure 4.2.2). After the data load, two indexes(Fig.4.2.3) were created on the “User\_Id” and “Text” fields for the purpose of better query performance and the reason for this is that most of the queries in the search application relied on fetching details based on this id and on joining and combine the results from sql database.

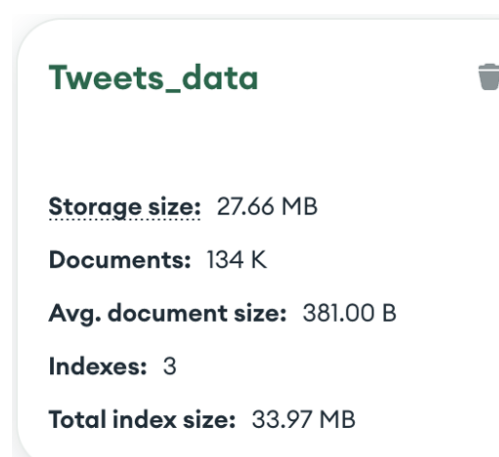


Fig 4.2.1: MongoDB Summary

```

_id: ObjectId('64471bda65ef7286884c347a')
created_at: "2020-04-12 18:27:26"
Tweet_Id: "1249403771676815361"
Text: "RT @ani_royal007: India Mai Corona Virus Ka Bhukar...!! https://t.co/kZ..."
▶ Hashtag: Array
User_Id: "2341644176"
User_Name: "Abhisht"
Source_tweet_Id: "1248907321947783170"
Retweet_Count: 0
Likes_Count: 0

```

Fig 4.2.2: MongoDB Schema for tweets

▼ Text_text	TEXT ⓘ	28.8 MB	10 (since Wed Apr 26 2023)
_FTS (TEXT) _FTSX ↑			
▼ User_Id_1	REGULAR ⓘ	2.3 MB	35 (since Wed Apr 26 2023)
USER_ID ↑			
▼ _id_	REGULAR ⓘ	2.9 MB	11 (since Mon Apr 24 2023)
_ID ↑			

Fig 4.2.3: Indices in MongoDB

## 5. Search Implementation

Searches form the core part of the application, in this section, the implementation and working of the search functionalities will be discussed.

### 5.1. User Search (By Prashanth Aripirala)

Searching for users and their information is a critical aspect of our application. When a user begins a search, they enter the "@" symbol before their desired search term, indicating they are looking for other users. The application then utilizes a search function to query the database and display the search results to the user. To optimize performance, the search function first checks for any matching results in the cache and retrieves them if they exist. If no matching results are found, the function retrieves the results from the database and stores them in the cache for future searches.

Along with displaying user information, the search function also retrieves and stores the recent three tweets of each user from the Tweets database in MongoDB.



This feature enables users to view the most recent tweets of their selected authors, who are sorted by `followers_count` and `verified_status`. Once the search results are displayed, users have the option to select the desired author(s) whose tweets they want to see, by entering a number from 1 to 5. The application then retrieves the corresponding tweets from the cache and displays them to the user, providing a seamless and efficient user experience.

## **5.2. String Search (By Anirudh Gaur)**

Searching for part of a tweet is another essential task that the application performs. For this search query, indexing was done on the text field of each tweet and a text index was created. For the query to get more relevant results, standalone stop words from the search (like is, an the, etc.) were removed. In this application, if '#' or '@' are not entered at the start of a search query, it will automatically search for part of a tweet and display the 5 most recent tweets matching the search query. The tweets have been sorted by the most recent one and a tweet has been given more preference compared to a retweet. Search by string data was not stored in cache as there is no use in storing search by a particular part of a tweet in cache due to the randomness which is linked to this search query. In the results section, the most relevant tweets based on a particular string search will be shown.

## **5.3. Hashtag Search (By Rohit Macherla)**

The final search feature of the application is searching by Hashtags. Users can enter the '#' symbol followed by the desired hashtag to get the results. When a search is made, the application runs a query in the background to hit MongoDB and displays the top 5 hashtags that are not only the exact matches but also nearby searches and displays the top 5 hashtags ordered by the number of times they appeared in the database. As part of the drill-through, users can select for which of the displayed hashtags he/she wants to get the tweets, and then the top 3 tweets are displayed by the relevance of recently created. Getting the tweets as part of the drill-through is done in the background simultaneously either from cache or

from the database while the user is looking at the initial results so as to provide the results immediately for a better user experience.

## **5.4. Trending Searches**

### **Trending Users: (By Prashanth Aripirala)**

Our application offers a "Trending Users" feature that provides users with a quick look at the most popular users based on their tweets\_count and number of followers. To access this feature, users can simply click a button on the home page. When the button is clicked, the application calls a function that retrieves the results. To improve performance and reduce the load on the database, the function first checks whether the results are already present in the cache. If they are, the results are retrieved from the cache and displayed to the user. If the results are not present in the cache, the function fetches them from the database and stores them in the cache for future use. Once the results are retrieved, they are displayed on the UI for the user to view, providing a convenient and efficient way to stay up-to-date with the most popular users on the platform.

### **Trending Tweets: (By Rohit Macherla)**

Similar to the "Trending Users" feature, this feature can also be accessed with a single click which gives information about the most popular tweets based on the compound field involving retweets and likes count. Every tweet was given a composite score based on the retweets and likes the tweet got. 60% weightage was given to the retweets and 40% to the likes as retweets are the major indicator of engagement on Twitter and contribute directly to the spread of information. The top 10 trending tweets were obtained based on this composite score and the respective username, retweet count, likes count, created date and time, the text of the tweet and hashtags the tweet contained were displayed in the UI order by the highest composite score to the least.

### **Trending Hashtags: (By Rohit Macherla)**

This feature can also be accessed with a single click from the user interface which provides information about the top 10 most used hashtags. To calculate the trending hashtags, every hashtag was given a count based on the total number of times the particular hashtag appeared across all the tweets. The hashtags were then ordered in descending order and the top 10 were displayed in the UI along with their total count.

## **6. Caching (By Rishik Salver and Rohit)**

Caching is a technique used to speed up data access by storing frequently accessed data in a temporary storage location called cache, so that the data can be quickly accessed the next time it is needed, instead of querying the database every time. This helps to reduce the load on the server, decrease response time, and improve the overall user experience.

### **6.1. Cache Implementation:**

In this project, a Python dictionary was used as a cache to store user information and their tweets. The cache class has been implemented to provide efficient caching mechanisms, and it offers various methods for managing the cache. Here are the methods of the class and their functions:

- **\_\_init\_\_() method:** This method initializes the Cache object with a maximum size, eviction strategy, checkpoint interval, and time-to-live (TTL) for cached items. The maximum number of objects that can be held in the cache is determined by the `max_size` parameter, which for this project is set to 15000, or around 10 MB of RAM.

The `evict_strategy` parameter specifies the eviction strategy which is used to remove items when the cache is full. In this project, the eviction strategy is set to "least\_accessed". This means that when the cache is full, the item with the lowest access counts will be removed first. Hence, frequently used items are kept in the

cache while rarely used items are deleted. Every time a cached item is accessed, its access count, which is kept in the `access_count` attribute, is increased by 1.

The `checkpoint_interval` parameter specifies the time interval between checkpoints for saving the cache to disk. It is set at 300 seconds or 5 minutes. This implies that every five minutes, the cache will be saved to disk.

The `ttl` parameter specifies the time-to-live for cached items. For our use case, it is set to `None`. This means that cached objects won't lose their value and will be retained there until the cache fills to its maximum capacity or they are manually removed.

The `cache` attribute is a dictionary that stores the cached items, with the cache item keys acting as the keys and the cache item value and timestamp acting as the values. The least-accessed entries are removed from the cache when it reaches its maximum size to make room for new ones.

- **`load_from_checkpoint()` method:** This method is used to load the cache from a file. The method accepts a filename as an input and uses the pickle module to load the cache and `access_count` dictionaries from the file. The appropriate characteristics of the Cache object are then given the cache and `access_count` dictionaries. In the event of any failures, this method proves to be useful for restoring the cache to a previous state.
- **`save_to_checkpoint()` method:** This method is used to save the cache to a file. The cache and `access_count` dictionaries are saved to the file using the pickle module using this method, which accepts a filename as input. To make sure that the cache is saved to a file at regular intervals, this method is called every 1 hour as this is our `checkpoint_interval`.
- **`get()` method:** The `get` method takes a key as an argument and returns the value associated with that key in the cache. If the key is not present in the cache, `None` is returned. The key's access count is incremented. If the cache reaches its

maximum capacity, the item with the least access count is evicted, and the key and its access count are removed from the cache.

- **put() method:** The put method takes a key-value pair as an argument and adds it to the cache. The method checks if the cache has reached its maximum size. If that's the case, the item with the least access count is evicted. The key-value pair is added to the cache, with the access count for the key set to zero. If the checkpoint interval has passed, the cache is saved to disk.

## 6.2. Caching in Search Application:

The cache class mentioned above plays a vital role in improving the search functionality for both search by username and search by hashtag discussed in the above section. Whenever a search query is made, the application first checks whether the required data is present in the cache. If the data is already present in the cache, it directly returns the results to the user using the **get()** method. However, if the data is not present in the cache, the application queries the database and saves the results to the cache using the **put()** method for faster retrieval in the future thereby reducing the response time and the load on the database. Based on the checkpoint time filed, cache is stored to a checkpoint file using the **save\_to\_checkpoint()** as per the mentioned checkpoint interval and can be restored back using the **load\_from\_checkpoint()** when the system is restarted making it resilient in case of sudden failures. Apart from all of these features, the cache also checks the access counts field before putting items into the cache to remove some items according to the eviction strategy. Therefore, using caching mechanisms this way not only boosts the system's performance but also helps in optimizing the use of system resources, making it more efficient.

## 7. Search Application Design (By Anirudh Gaur)

User experience is one of the most important aspects, especially for a search application based to simulate functionalities of a social media app like Twitter. In this version of the search application, CSS styling and HTML was used to build the user interface. Establishing a connection between the front end and the backend was done using the Python Flask library and a

virtual environment was created by Anaconda to use the Flask library. Multiple pages were created using HTML which are stored in a separate template folder. Below figure 7.1 shows the home page of the twitter search application.

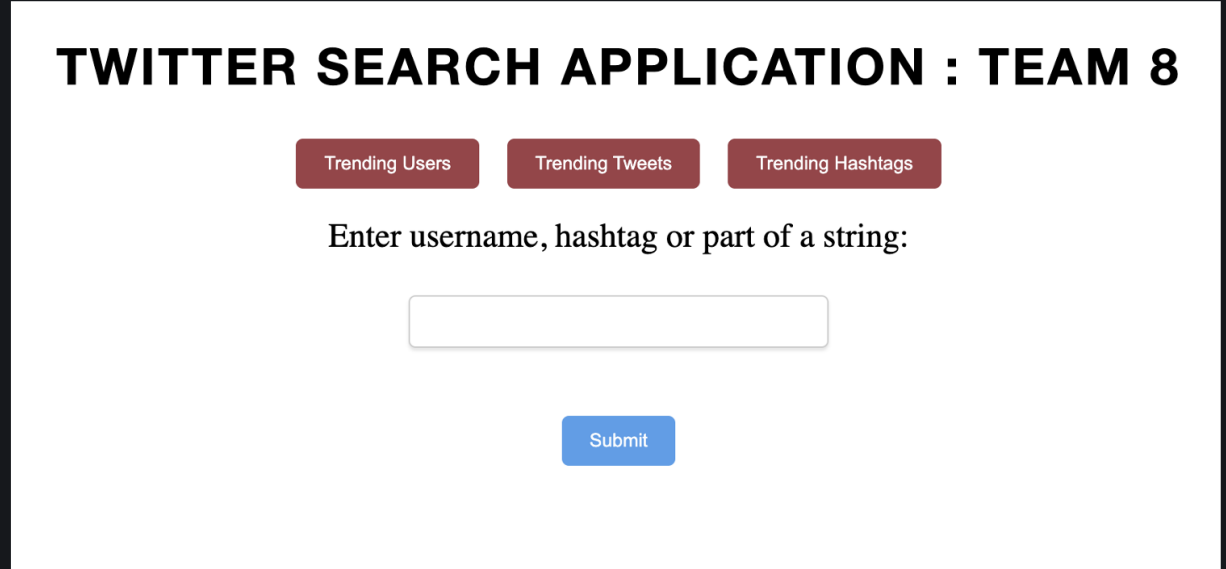


Figure 7.1: User interface home page

As visible on the home page of UI, there are 3 buttons for trending users, tweets and hashtags and also an input field for search query along with a submit button. These buttons are interacting with the backend using GET/POST methods and are being sent to function in the app.py file which helps in rerouting the requests made by users to another HTML page based on the type of the search performed.

All the HTML pages have been stored in a folder called template from where they are being accessed and rendered by our python app routing Flask functions. For each query, there are further drill-downs available which are discussed in the respective sections above. These drill-downs vary from displaying tweets related to a particular hashtag or displaying tweets related to a particular username. For users returned using trending users or search by username, there are hyperlinks available in the Handle field of the users table which is available in the below figure 7.2.

## User Search Results

#	Name	Handle	Verified	Followers	Tweets	Description	Location	Creation Date
1	Rohit Kansal	<a href="#">kansalrohit69</a>	✗	118385	0	Civil Servant IAS; Pr Secy Planning J&K LSE, IIM Kolkata; RTs not endorsement	Jammu, Jammu And Kashmir	2016-05-13 13:43:06
2	Rohitt Jaiswal	<a href="#">rohitjsw01</a>	✓	38921	1	Film Critic   Trade Analyst   Film Marketing Consultant, For Campaign collaborations & film marketing related query mail me 📧 cinematicfreedom@gmail.com	Kolkata, India	2014-04-28 03:47:09
3	Pinky Rajpurohit (ABP News)	<a href="#">Madrassan_Pinky</a>	✗	23026	1	Marwari chori brought up in Madras. South India Correspondent with @ABPNews. Focus @DefenceMinIndia @ISRO @DRDO_India and Politics. Ex @NewsNationTV @ANI	Bengaluru, India	2012-10-02 06:24:54
4	Dr.Sanjeev Rajpurohit	<a href="#">DrSanjeevRajp4</a>	✗	12545	5	Assistant Director (Research) J.R.Nagar Rajasthan Vidyapeeth University, Spokesperson, District Congress, Udaipur. RT# Endorsement Tweets=ByLifeExperiences	Udaipur, India	2015-08-14 16:09:43
5	Mersal Rohit ツ	<a href="#">Mersal_Rohit</a>	✗	8586	1	VIJAY IMy Silence Is My Attitude IPhotography♥ Crazy on editing🔪மத்தவர்கள் வேதனை படுத்தற ஒரு சின்ன ஸ்தைல் கூட தப்பு தான்! Enjoy Every Moment @ImagixCreations💙	Vijay's ❤️	2017-06-21 17:20:01

Enter the user number whose tweets you want to be displayed:

Submit

Fig 7.2: Username search having user handle as hyperlink

## 8. Results

**8.1. User Searching:** Below is the results page after searching for the term “@rohit”. All the fields that are deemed necessary are displayed on the UI. Below the results, the text box takes input 1 to 5 asking which author’s tweets the user wants to see. For example, if tweets of “Dr.Sanjeev Rajpurohit” are to be seen, 4 is entered as displayed in the Figure Below.

User Search Results								
#	Name	Handle	Verified	Followers	Tweets	Description	Location	Creation Date
1	Rohit Kansal	<a href="#">kansalrohit69</a>	✗	118385	0	Civil Servant IAS; Pr Secy Planning J&K LSE, IIM Kolkata; RTs not endorsement	Jammu, Jammu And Kashmir	2016-05-13 13:43:06
2	Rohitt Jaiswal	<a href="#">rohitjsw01</a>	✓	38921	1	Film Critic   Trade Analyst   Film Marketing Consultant, For Campaign collaborations & film marketing related query mail me 📧 cinematicfreedom@gmail.com	Kolkata, India	2014-04-28 03:47:09
3	Pinky Rajpurohit (ABP News)	<a href="#">Madrassan_Pinky</a>	✗	23026	1	Marwari chori brought up in Madras. South India Correspondent with @ABPNews. Focus @DefenceMinIndia @ISRO @DRDO_India and Politics. Ex @NewsNationTV @ANI	Bengaluru, India	2012-10-02 06:24:54
4	Dr.Sanjeev Rajpurohit	<a href="#">DrSanjeevRajp4</a>	✗	12545	5	Assistant Director (Research) J.R.Nagar Rajasthan Vidyapeeth University, Spokesperson, District Congress, Udaipur. RT# Endorsement Tweets=ByLifeExperiences	Udaipur, India	2015-08-14 16:09:43
5	Mersal Rohit ツ	<a href="#">Mersal_Rohit</a>	✗	8586	1	VIJAY IMy Silence Is My Attitude IPhotography♥ Crazy on editing🔪மத்தவர்கள் வேதனை படுத்தற ஒரு சின்ன ஸ்தைல் கூட தப்பு தான்! Enjoy Every Moment @ImagixCreations💙	Vijay's ❤️	2017-06-21 17:20:01

Enter the user number whose tweets you want to be displayed:

Submit

Fig 8.1.1: User Search Results

Then the user is redirected to the page which consists of the recent three tweets of that author and some details about those tweets as shown in the Figure below.

### User Tweets

User: Dr.Sanjeev Rajpurohit

#	Date	Text	Hashtags	Retweets	Likes
1	2020-04-25 14:19:43	RT @ashokgehlot51: A separate ward has been set up at the Mahila Chikitsalaya, Sanganeri Gate, #Jaipur for pregnant women coming from #coro...	['Jaipur']	0	0
2	2020-04-25 13:08:30	@RahulGandhi : Problem is you are building your #CentralVista project at same time. So you not sort of doing just t... <a href="https://t.co/7IjWN4ale3">https://t.co/7IjWN4ale3</a>	['CentralVista']	1	0
3	2020-04-25 12:23:19	RT @The_ManishSood: Want to know what we've done wrong in this country under modi govt about Corona pandemic.. Watch this video explaining...	-	0	0

Fig 8.1.2: Tweets of the selected user

**8.2. Hashtag Searching:** Below is the results page after searching for the term “#corona”. Results are displayed by the number of tweets they appear in as shown below in Figure below.

### Top 5 Hashtags Matching Your Search for "#corona"

Hashtag	Tweets Count
Corona	5920
corona	1928
coronavirus	825
COVID19	572
Covid_19	503

Enter the hashtag related to which the relevant tweets you want to be displayed:

Submit

Fig 8.2.1: Top 5 Hashtags related to search term

From the above results, users can then select the hashtag for which tweets are to be displayed by typing the hashtag in the search box below the results and clicking submit. If the user enters “Covid\_19” as the hashtag, the top 3 recent tweets containing that particular hashtag are shown as seen in Figure below.



### Tweets of #Covid\_19

Username	Retweets	Likes	Created At	Text	Hashtags
HARPAL SINGH	0	0	2020-04-25 14:48:25	#Covid_19 #Who_Is_Real_SadGuru #Why_Need_TrueWorship the benefits of true Bhakti which availed by Mirabai / Guru... <a href="https://t.co/EPSazvXEOK">https://t.co/EPSazvXEOK</a>	['Covid_19', 'Who_Is_Real_SadGuru', 'Why_Need_TrueWorship']
SWR Sport	0	0	2020-04-25 14:48:24	Traurige Nachrichten aus Mannheim. @svw07 #corona #Covid_19 <a href="https://t.co/6JRZRimIZG">https://t.co/6JRZRimIZG</a>	['corona', 'Covid_19']
Divyanshu Nidhi	0	0	2020-04-25 14:48:22	RT @pranitasubhash: Gratitude 🙏 Preparation and packing of meals #Corona #Covid_19 <a href="https://t.co/LvIUlracSy">https://t.co/LvIUlracSy</a>	['Corona', 'Covid_19']

Fig 8.2.2: Top 3 recent tweets of a hashtag

- 8.3. String Searching:** Below is the result for searching tweets relevant to the string given as the search query. The string given by the user is “death on” and the top 5 tweets containing this string as a part of the tweet are displayed on the UI.

### Tweets relevant to "death on"

User	Retweets	Likes	Created At	Text	Hashtags
Ahmed Siddique	0	0	2020-04-25 14:47:47	<a href="https://t.co/owsm135LeQ">https://t.co/owsm135LeQ</a> Watch corona virus patient death on youtube 🙏 #SaturdayThoughts #KIMJONGUNDEAD... <a href="https://t.co/guycuZmG">https://t.co/guycuZmG</a>	['SaturdayThoughts', 'KIMJONGUNDEAD']
Krishnendu Ghosh	0	0	2020-04-25 14:33:27	Latest news of Bengal, 57 Death in Corona but death only for Corona is 18, I can't understand what it means stand for. #CoronaPoliticsHurts	['CoronaPoliticsHurts']
Michelle Meno	0	0	2020-04-25 12:35:46	Truth traveling for work everyday.China to Italy.. Maybe we should count all the good people of Italy's death on Ch... <a href="https://t.co/ms9wrSMwo0">https://t.co/ms9wrSMwo0</a>	[]
nat🍷	0	0	2020-04-12 18:29:17	maybe it's just me , but I ain't ever wish death on nobody	[]
juwelz v	0	0	2020-04-12 18:27:25	RT @nuffsaidny: wishing death on people is weirdo behavior.	[]

Figure 8.3: String search results

### 8.4. Trending Searches:

**Trending Users:** When the user clicks the Trending Users button on the Home Page, they are redirected to the page as shown in the Figure below, all the necessary information which are deemed necessary are fetched from the Users Database and displayed.

## Trending Users

Name	Handle	Verified	Followers	Following	Location	Tweets	Bio
Barack Obama	<a href="#">BarackObama</a>	Yes	116518121	607194	Washington, DC	1	Dad, husband, President, citizen.
Donald J. Trump	<a href="#">realDonaldTrump</a>	Yes	78467254	46	Washington, DC	0	45th President of the United States of America 🇺🇸
CNN Breaking News	<a href="#">cnnbrk</a>	Yes	57529057	120	Everywhere	0	Breaking news from CNN Digital. Now 56M strong. Check @cnn for all things CNN, breaking and more. Download the app for custom alerts: <a href="http://cnn.com/apps">http://cnn.com/apps</a>
Narendra Modi	<a href="#">narendramodi</a>	Yes	55781248	2364	India	6	Prime Minister of India
Shakira	<a href="#">shakira</a>	Yes	52250613	212	Barranquilla	0	🎵 ME GUSTA Shakira & Anuel AA Nuevo Sencillo / New Single
CNN	<a href="#">CNN</a>	Yes	47567385	1106	None	0	It's our job to #GoThere & tell the most difficult stories. Join us! For more breaking news updates follow @CNNBRK & Download our app <a href="http://cnn.com/apps">http://cnn.com/apps</a>
The New York Times	<a href="#">nytimes</a>	Yes	46359985	904	New York City	1	News tips? Share them here: <a href="http://nyti.ms/2FVHq9v">http://nyti.ms/2FVHq9v</a>
BBC Breaking News	<a href="#">BBCBreaking</a>	Yes	43014510	3	London, UK	0	Breaking news alerts and updates from the BBC. For news, features, analysis follow @BBCWorld (international) or @BBCNews (UK). Latest sport news @BBCSport.
Amitabh Bachchan	<a href="#">SrBachchan</a>	Yes	41596464	1833	Mumbai, India	1	"तुमने हमें पूज पूज कर पत्थर कर डाला ; ये जो हमपर जुमले कसते हैं हमें ज़िंदा तो समझते हैं" ~ हरिवंश राय बच्चन
Salman Khan	<a href="#">BeingSalmanKhan</a>	Yes	40094611	26	MUMBAI	0	Film actor, artist, painter, humanitarian

Fig 8.4.1: Trending Users

**Trending Tweets:** When the user clicks on the Trending Tweets button, they are redirected to the page shown below that displays the top 10 tweets based on the composite score which is discussed above.

## Trending Tweets

User	Retweets	Likes	Created At	Text	Hashtags
Quirinale	798	9524	2020-04-25 09:23:42	#25Aprile, il Presidente #Mattarella si è voluto recare all' #AltaredellaPatria dove ha deposto una corona d'alloro... <a href="https://t.co/ev4b4Dl9Df">https://t.co/ev4b4Dl9Df</a>	['25Aprile', 'Mattarella', 'AltaredellaPatria']
dr. Shela Putri Sundawa	784	5498	2020-04-25 05:24:38	Ngomongin teori konspirasi corona sama orang yg percaya kalo bumi itu datar, I'm not a smart people, but please dud... <a href="https://t.co/aCIWBZg41m">https://t.co/aCIWBZg41m</a>	[]
Quirinale	654	4062	2020-04-25 09:56:54	#25Aprile, nel 75° anniversario della #Liberazione il Presidente #Mattarella ha deposto una corona all' Altare della... <a href="https://t.co/vDcVcwEMQy">https://t.co/vDcVcwEMQy</a>	['25Aprile', 'Liberazione', 'Mattarella']
Brit Hume	1474	1831	2020-04-25 11:27:12	And where is the evidence that Covid 19 is easily spread outdoors? <a href="https://t.co/YPcJXU1uqw">https://t.co/YPcJXU1uqw</a>	[]
🇷🇺 Only In Russia	586	1352	2020-04-25 11:26:44	Corona disinfecting in Russia. <a href="https://t.co/AEF5ccDmM3">https://t.co/AEF5ccDmM3</a>	[]
🇧🇷 🇧🇷 🇧🇷	445	1532	2020-04-23 19:58:13	Quando eu ligo a televisão e fica falando só de corona vírus	[]
Ben Wikler	986	0	2020-04-25 12:51:03	Milwaukee's health commissioner has now tied 40 coronavirus infections to the April 7 election. <a href="https://t.co/fGIsLKzTkm">https://t.co/fGIsLKzTkm</a>	[]
Funda	732	0	2020-04-25 12:43:26	Gözün çıksın corona 🤔 Ülkece asabii olduk 🤔 muhtemel psikoloji ektedir 🤔🤔🤔🤔🤔 #PideAlmayaDiyeÇıkıp <a href="https://t.co/KoGSbVAMxZ">https://t.co/KoGSbVAMxZ</a>	['PideAlmayaDiyeÇıkıp']
Laura Ingraham	500	0	2020-04-25 13:53:37	A MUST READ...Coronavirus Restrictions: Government Bears the Burden of Proof Before Denying Freedoms   National Rev... <a href="https://t.co/RcaAK9nwDs">https://t.co/RcaAK9nwDs</a>	[]
Gu Ru Thalaiva	422	0	2020-04-25 13:01:20	Thalapathy fans from Sivakasi Helped the Poor Family who are affected by this corona Crisis ! They have supplied th... <a href="https://t.co/ozoG6d9Ax8">https://t.co/ozoG6d9Ax8</a>	[]

Fig 8.4.2: Trending Tweets

**Trending Hashtags:** Similar to the other trending searches, when the user clicks on the ‘Trending Hashtags’ button, they are redirected to the page as shown below which displays the top 10 most used hashtags across all the tweets and their counts.

## Trending Hashtags

Hashtag	Tweets Count
Corona	5920
corona	1928
Mattarella	1516
25Aprile	1476
Covid_19	1119
COVID19	1003
coronavirus	825
AltaredellaPatria	806
PideAlmayaDiyeÇıkıp	777
Liberazione	700

Fig 8.4.3: Trending Hashtags

### 8.5. Caching:

Caching was implemented to improve the user experience through better application performance. So, let’s try to compare the performance of data retrieval from the database and from cache for different searches. This comparison shows how caching can improve the overall performance of the system by reducing the time required to access frequently accessed data.

#### 8.5.1. Search by @username:

The retrieval times from the database are shown below when the search term is ‘@jimmy’

```
*****
Search term @jimmy
Time for getting user info: 0.09813242599999938
Time for getting tweets info: 0.22909174800000187
```

Fig 8.5.1.1: Time to extract user and tweet information from the database

The retrieval times from cache are shown below when the search term is '@jimmy'

```
*****
Search term @jimmy
Time for getting user info: 0.000228437999999350267
Time for getting tweets info: 0.00048390599999994788
```

Fig 8.5.1.2: Time to extract user and tweet information from cache

## 8.5.2. Search by #hashtag

The retrieval time from the database is shown below when the search term is '#run'

```
*****
Search Term #run
Time for getting tweets info: 0.3470667460000003
```

Fig 8.5.2.1: Time to extract tweet information from the database

The retrieval time from cache is shown below when the search term is '#run'

```
*****
Search Term #run
Time for getting tweets info: 0.00062963099999953564
```

Fig 8.5.2.2: Time to extract tweet information from cache

As it can be observed from the query timings, fetching the information from cache is more than 500 times faster than fetching from the database, reiterating the fact that cache is a very powerful tool which decreases the response time, reduces the load on the database and improves the user experience greatly.

## **9. Conclusion**

This project was an insightful exploration of data ingestion into multiple technologies. The project focuses on ingesting data from a JSON file into two different databases, MySQL and MongoDB. In addition, a search application was built using Python Flask, which demonstrates the ability to retrieve data from these databases based on user input search terms.

Furthermore, the project highlights the significance of caching using Python dictionaries and indexing in improving query performance. By caching search results and indexing tables, query timings are greatly reduced, resulting in a more efficient and effective system. Overall, this project offered a fantastic opportunity to learn how to integrate a variety of tools and technologies and showcases the importance of database design and optimization, providing valuable insights into the use of multiple technologies and tools for improving database system performance.