

**PROJECT REPORT**

**Twitter Search Application**

**“Chirp Search”**

**Prepared by**

Vishnuram Jatin Bangaru(vb440)

Ankeeta Priyam(ap2213)

Noopur Singh(ns1482)

Ganesh Raj Kyatham(grk62)

Table of Contents

**[INTRODUCTION](#_Toc132218008)** [3](#_Toc132218008)

**[PROJECT ARCHITECTURE](#_Toc132218009)** [4](#_Toc132218009)

**[PROCESSING TWEETS FOR STORING IN DATASTORES](#_Toc132218010)** [6](#_Toc132218010)

**[SEARCH APPLICATION DESIGN](#_Toc132218011)** [7](#_Toc132218011)

**[RESULTS](#_Toc132218012)** [8](#_Toc132218012)

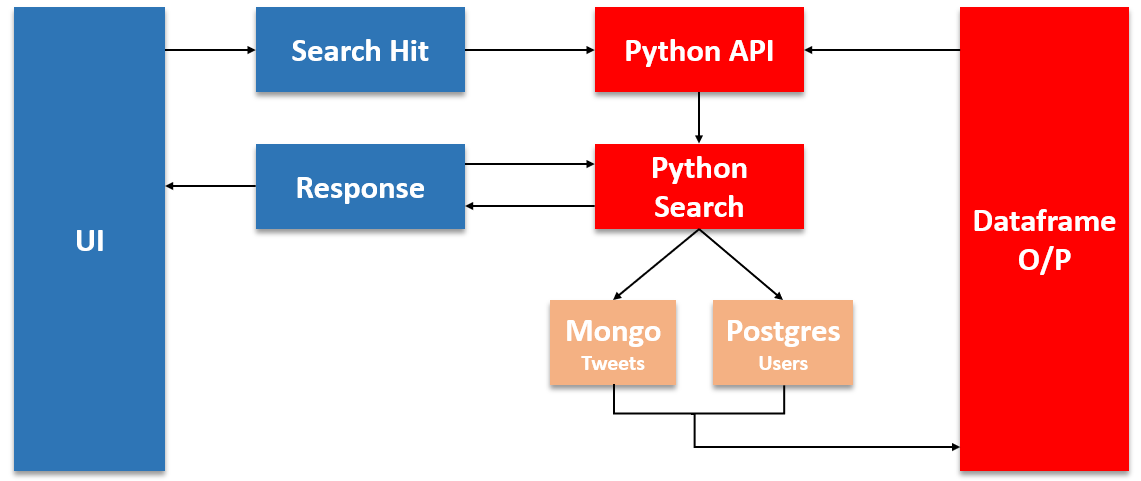
**[REFERENCES](#_Toc132218013)** [10](#_Toc132218013)

# **Introduction**

Twitter is a popular social media platform that enables users to share short messages, known as tweets, with a wide audience. With over 330 million monthly active users, Twitter has become a critical communication tool for individuals, businesses, and organizations worldwide. The platform's real-time nature and broad reach makes it a valuable source of information for breaking news, trending topics, and public opinion. Overall, Twitter plays a significant role in shaping public discourse and connecting people from diverse backgrounds and perspectives.

As part of the MSc Data Science program at Rutgers University, State University of New Jersey, course 954:694:01, the project is started that involves developing a search application utilizing Twitter data. The objective is to design and store the information in multiple data stores to enable rapid access via Python. Additionally, the project involves implementing a caching system for the top search results to minimize the need for repeated visits to the data stores.

# **Project Architecture**



*Fig. 1*

The project architecture displayed above depicts the flow of information. Firstly, the UI receives input from the user and sends it to the Python API.

The Python API then directs the control to the search function in the Python source code. Within the search function, a query string is formulated based on the input received from the user, and it is sent to the mongo dB context, where the Tweets table contains the relevant tweets.

The output of this search is returned to the Search Function, which generates an output Dataframe that includes the tweet, username, and the number of retweets for that tweet. The Python API then receives this result and sends it back to the UI, which displays the response to the user.

If the user clicks on the username or number of tweets, the Python API is called again, and depending on the selection, either the Users table in PostgresSQL (for user click) or the Tweets table in MongoDB (for tweet click) is queried. If the user clicks on the username, the details of the user are returned. If the user clicks on the retweet count, the retweeted tweets are returned.

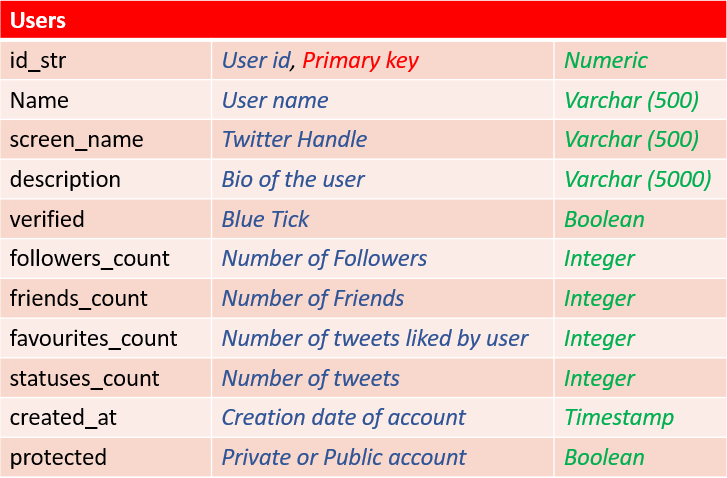
**Persisted Data Model and DataStores**

The project incorporates two different types of databases - Relational and Non-Relational. For Relational database management, PostgresSQL is employed, and for Non-Relational database management, MongoDB is utilized.

The project have connected both of these databases to python and performed insert, delete, query and other basic operations.

The tweet dataset is distributed between these databases. User level details are stored in the Users table in PostgresSQL, while tweet-related details are stored in the Tweets table in MongoDB.

The Users table's architecture and schema are displayed in the figure 2.

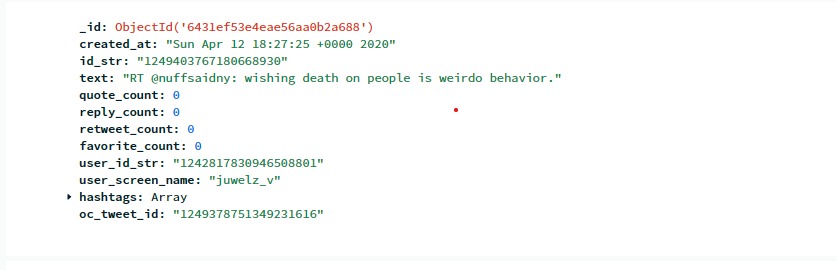


*Fig. 2*

PostgreSQL is a popular open-source relational database management system that uses SQL to store and manage data in tables.

The schema is designed such that the data is stored and accessed efficiently.Each tweet is processed and user detail is extracted from each and inserted one by one using the command “Insert Into” in the Users table.

The Tweets table's architecture and schema are illustrated in the figure 3.



*Fig. 3*

MongoDB is a popular NoSQL database that stores data as JSON-like documents in collections. To insert the tweets, we have made use of dictionary collection. Formulated the required data into key-value pair and inserted into the table.

# **SEARCH APPLICATION DESIGN**

The project have allowed for basic searches where a user can search based on user\_name, a tweet string or hashtag. The search would retrieve information about the tweet made, when it was made, the retweet\_count, the user details(no\_of\_followers, user\_name).

The project expands by providing drill downs by clicking on the retweet\_count, user\_name to further display details of the retweet or the other tweet details made by the user.

Suppose the search is based on a string, we would clean the string and trigger a python function which would retrieve the tweet details using the find operator and regex operator from MongoDB. We would then retrieve the user details from the User Table using the foreign-key alike user\_id\_str and merge to get the final output. This is a bare bound approach which we are working for the search queries.

4. We extracted the id of the original tweet which as been retweeted and stored it as an attribute(oc\_tweet\_id) in the current tweet. The Original tweet is inserted into the database as a tweet object. It can be accessed by querying using the oc\_tweet\_id attribute.

7. We are planning on using a number of factors to order the search results. The factors include retweet\_count, Quote\_count, reply\_count.

8. This is our plan to split the work:

ganesh raj - non relational database, search application ui, query processing

vishnu - query processing, cacheing

noopur- python api , cacheing

ankeeta - relational database , database connections and joins

# **RESULTS**

Show the results of each type of query.

Timings of your test search queries (make sure you are hitting cached and non cached data)

**CONCLUSIONS**

# **REFERENCES**