# DESIGN DOCUMENT - TWITTER implementation using MongoDB and PYTHON

## GENERAL OVERVIEW OF YOUR SYSTEM WITH A SMALL USER GUIDE

The Twitter Program (project2.py) allows users to a range of options of viewing tweets and users through an integrated service built through MongoDB and Python. The options include:

1) ***Search Tweets***: Allows for multiple keywords present within the content of the tweet, displays tweet details, and lets the main user select tweets to see more.
2) ***Search Users***: Allows for keyword search within the displayname or location of the user, displays user details, and lets the main user select users to see more.
3) ***List Top Tweets***: Main user can find a (user-requested) number of maximum tweets based on either retweets, likes, or quotes (which is also user-selected).
4) ***List Top Users***: Main users can find a (user-requested) number of maximum users based on highest follower count.
5) ***Write Tweets***: Main user (291user) can compose a tweet.
6) ***Exit***: Finally, when the user wishes to exit the program, this allows them to do so.

## DETAILED DESIGN OF OUR SOFTWARE:

Our software currently uses a variety of different functions to prevent redundancy of code along with several imported modules and globalized user defined variables.

### Imported Modules

1) sys:
2) MongoClient: Imported from 'pymongo' to make the connection to the MongoDB.
3) ObjectId: Imported from 'bson' to retrieve the ObjectID of tweets.
4) os: To be able to use os.system('clear') which is to clear the outputs displayed at the moment.
5) datetime: Used to retrieve current date and time to write a tweet into the MongoDB.
6) time: Used for time.sleep(5) to provide smooth user interaction.
7) pprint: To prettily print the dictionary outputs.
8) json: Used in the load-json.py to read the JSON file using json.loads()

### Global Variables

1) db: Connects to the 291db.
2) tweets: A collection that is created in the 291db, loaded in the load-json.py and accessed in the proejct2.py.

## Notes

- ➢ Users are almost always prompted to try inputting again if invalid input is entered.
- ➢ **os.system('clear')** and **time.sleep(5)** have been used to clear the output screen and pause accordingly to imitate website flow from one page to another.

## User Defined Functions

1) main(): Presents the menu for the five important functionalities of the program

2) search_tweets(): Searches for tweets with one or more keywords present in tweet content and printing certain details if found or a specific message if not found. Offers to continuously search for tweets with other keywords or to go to the main menu.

3) tweets_keyword_validation(): Confirming that the keyword(s) has at least one character, and if there are multiple keywords, split based on space and return a list of the split keywords.

4) select_tweet(oids): Accepts the list of all the displayed tweet IDs and allows the user to select a tweet ID to view from that list. Confirms that the selected tweet ID (S.No) is present in the oids before printing.

5) search_users(): Searches for user and pulls only the necessary parameter fields to print. Offers to select a user to view more details.

6) condition_to_user(list_of_id): Accepts argument which is a list of user IDs that are being displayed. Also allows the user to select a user and view more details about them.

7) select_user(temp_user_id): Same argument as the above function. This function checks that the inputted number of records to view is always smaller than or equal to the total number of users that can be displayed. It then prints all the details or the record neatly.

8) user_keyword_validation(): Checks whether a user inputted only one keyword to search for users. Otherwise keeps prompting the user to try again.

9) list_top_tweets(): Prompts the user to enter number of records they wish to view and the basis for ranking before calling for top_tweets(x,user_value).

10) top_tweets(x,user_value): 'x' value is the basis for ranking and user_value is the number of records user wishes to see displayed passed as arguments. Prints all the top tweets neatly.

11) list_top_users(): Takes user ID as a dictionary key and all the respective user details as a list of values for its value. When asked to display top users, sorts the dictionary and neatly prints the necessary parameters after calling for validate_user_input(max_value). Also offers to select a user by calling for condition_to_user(list_of_id).

12) validate_user_input(): Validates that the entered user ID (S.No) to search for is not a negative value. Returns the validated value to properly search for.

13) write_tweet(): Composes a tweet by accepting user input and sets the values for MongoDB after retrieving the current date and time.

## YOUR TESTING STRATEGY

We tested our code by running the program after every stage and trying to break it down to find problems. We worked on the MongoDB queries on the lab machines through ssh. Some of the errors we caught and solved are:

- ➢ ***Incorrect query outputs***: Through rigorous testing, we'd find edge cases resulting in incorrect outputs. For e.g., we had an error where our 'any' was retrieving both 'anybody' and 'any body' whereas it had to print only 'any body'. We fixed the query.
- ➢ ***Invalid inputs***: We validated user inputs in every function. We accepted user inputs and printed warning statements when they input an incorrect value, prompting the user to try again.
- ➢ ***Incorrect breaks in program flow***: Instead of using a while loop as in our first mini project, we chose to do a simple main menu function where break was easily implemented instead of having to trace the program flow and break out of every function.
- ➢ ***Manual testing***: Testing by running the program repeatedly, and seeing if a bug rises through our varied inputs. We confirm database updation using the MongoDB and JSON files through lab machine.

## GROUP WORK BREAKDOWN STRATEGY

Firstly, we coordinated through a VS Code extension called Live Share, which works similarly to Google Documents, to edit the source code simultaneously, alongside the MongoDB and Python extensions. Secondly, the work breakdown was decided as we worked together on this project. For example, we split work where Yaatheshini did the queries and python code for the elements related to tweets and design document while Vishal worked on the queries and python code for the elements related to users and composing tweets. We divided the work evenly through various contributions between ourselves. We spent roughly two days working on the project individually and together.