Design Document

# System Overview

This system is implemented in Python as a CLI application, with SQLite as the database, and a simulation of a social media platform. Users can interact with the system through commands to perform tasks such as posting tweets, replying to tweets, retweeting, viewing tweets, and managing hashtags. All data is stored in an SQLite database, and users can search tweets by keywords and view them sorted by date and time. The goal of this project is to provide a simple, CLI-driven tweet management system that simulates some basic functionalities of a social media platform . After login, they can:

Post tweets

Search for tweets or users View followers

Logout

# Files and Modules

The program is split into several modules, each of which handles a distinct functionality:

**main.py**: Manages user login, user registration, system functions, and program flow.

**follower\_utils.py**: Handles the display of followers, including detailed information about each follower.

**compose\_tweet.py**: Allows the user to compose and post tweets. **tweet\_search.py**: Handles the functionality for searching tweets. **search\_users.py**: Allows searching for users based on input criteria.

# Database Design

The system uses SQLite to store the following tables:

users(usr, name, email, phone, pwd) follows(flwer, flwee, start\_date) lists(owner\_id, lname) include(owner\_id, lname, tid)

tweets(tid, writer\_id, text, tdate, ttime, replyto\_tid) retweets(tid, retweeter\_id, writer\_id, spam, rdate) hashtag\_mentions(tid,term)

-- users

CREATE TABLE users

(

usr INT PRIMARY KEY,

name TEXT NOT NULL,

email TEXT NOT NULL,

phone INT NOT NULL,

pwd TEXT NOT NULL

);

-- follows

CREATE TABLE follows

(

flwer INT REFERENCES users ON DELETE CASCADE,

flwee INT REFERENCES users ON DELETE CASCADE,

start\_date DATE NOT NULL,

PRIMARY KEY (flwer, flwee)

);

-- tweets

CREATE TABLE tweets

(

tid INT PRIMARY KEY,

writer\_id INT NOT NULL REFERENCES users ON DELETE CASCADE,

text TEXT NOT NULL,

tdate DATE NOT NULL,

ttime TIME NOT NULL,

replyto\_tid INT REFERENCES tweets ON DELETE CASCADE

);

-- hashtag\_mentions

CREATE TABLE hashtag\_mentions

(

tid INT NOT NULL REFERENCES tweets ON DELETE CASCADE,

term TEXT NOT NULL,

PRIMARY KEY (tid, term)

);

-- list

CREATE TABLE lists

(

owner\_id INT REFERENCES users ON DELETE CASCADE,

lname TEXT,

PRIMARY KEY (owner\_id, lname)

);

-- include

CREATE TABLE include

(

owner\_id INT ,

lname TEXT,

tid INT NOT NULL REFERENCES tweets ON DELETE CASCADE,

PRIMARY KEY (owner\_id, lname, tid),

FOREIGN KEY (owner\_id, lname) REFERENCES lists ON DELETE CASCADE

);

-- retweets

CREATE TABLE retweets

(

tid INT NOT NULL REFERENCES tweets ON DELETE CASCADE,

retweeter\_id INT NOT NULL REFERENCES users ON DELETE CASCADE,

writer\_id INT NOT NULL REFERENCES users ON DELETE CASCADE,

spam INT DEFAULT 0,

rdate DATE,

PRIMARY KEY (tid, retweeter\_id)

);

# Module Details:

**follower\_utils.py Module:**

The **followe\_utils.py** module contains the following functions:

1. **showFollowers(user\_id, cursor)**:

This function shows a paginated list of followers for the given user.

It checks if there are more followers to display, offering the option to load more or view the details of a specific follower.

1. **getFollowerList(offset=0, limit=5)**:

Retrieves a list of followers for the current user using SQL queries.

Supports pagination with an offset and limit, displaying 5 followers at a time.

1. **showFollowerDetails(follower\_id)**:

Displays detailed information about a specific follower, such as their tweets and contact information.

Allows interaction with the follower, including viewing more tweets or following/unfollowing them.

# Group Work Breakdown Strategy

The work is divided among the team members as follows:

**Luke Thomas: Task:**

**Estimated Time:**

**Yuheng Li: follower\_utils.py Module**

**Task**: Were responsible for developing the

followers\_utils

module, which manages

the followers and followee relationships within the application. This includes implementing the functionality to follow and unfollow users, checking the follower status of a user, and providing necessary methods to retrieve lists of followers and the users a person is following.

**Estimated Time**: 10 hours

**Anant Gupta gurbaaz:**

**Task:**

**Estimated Time:**

**Gurbaaz Gill: Task:**

**Estimated Time:**

# Coordination Method:

The project uses Git for version control to manage code contributions. Team meeting was held once to discuss progress and resolve any issues.