DESIGN DOCUMENT

I. System Overview & User Guide:

This project is a Twitter-like social media platform implemented in Python 3.12 with an SQLite database. Main features are listed below.

a. Main Features:

- User Authentication: Secure login and sign-up system with password masking.
- Tweeting System: Find tweets using keywords or hashtags, show statistics about the tweet (no: of retweets, no: of replies), compose a reply to tweet, retweet, add a tweet to a selected favorite list.
- User Interaction: Retrieve tweets of a selected user, display statistics (no: of tweets, no: of followers, no: following), see up 3 most recent tweets, more tweets of the user etc.
- Compose Tweets: Compose Tweets and maintain hashtag table.
- List all followers: View a list of all their followers, select a follower to see statistics (no: of tweets, followed users, followers, and up to three recent tweets, with the option to follow them or view more tweets)
- Favorite Lists: Users can organize tweets into personal favorite list.

b. User Guide:

- Starting the Application:
  - Run the script with: python3 MiniProject.py <database_filename>
  - If no database file is provided, an error will occur.
- Login & Signup:
  - New users can register by providing name, email, phone no: & password.
  - Existing users log in with their user ID and password.
- Using the Platform:
  - Creates a platform for operations defined under Main Features.
- Navigation:
  - The system provides menu-driven navigation.
  - Users enter choices to interact with features.
  - Pagination is used for large search results.

This system provides an intuitive interface for social media-like interactions with structured data storage in SQLite.

II. Detailed design of your software:

This structured design enables a streamlined social media experience, integrating user authentication, tweet management, search functionalities, and follower interactions efficiently.

a. Functions and their Functionality:

i. Database Management:

- get_database_connection(): Establishes a connection to the SQLite database using a filename passed via command-line arguments. Returns a connection object.
- Set_Database(): Ensures database integrity by enabling foreign key constraints and creating necessary tables, including users, follows, tweets, retweets, lists, include, and hashtag_mentions.

ii. User Authentication:

- login_screen(): Displays options for login, sign-up, or exit.
- Calls login() or sign_up()accordingly.

- login(user_id, password): Authenticates users against stored credentials in the users table.
- sign_up(): Collects user information, validates inputs, assigns a user ID, and stores the new user in the database.
- masked_input(prompt="Enter your password: "): Hides password input
- is_valid_email(email): Validates email format using regex.
- is_valid_phone(phone): Ensures phone contains only digits.

iii. User Menu and Navigation:
- user_menu(user_id): Provides options for searching tweets/users, composing tweets, listing followers, and managing favorite list
- view_followed_tweets(user_id, offset, limit): Displays tweets and retweets from followed users with pagination.

iv. Tweet Management:
- compose_tweet(user_id, replyto_tid): Allows users to post tweets or replies while extracting and validating hashtags.
- retweet (user_id, tid): Inserts a record in retweets, linking the retweeter with the original tweet.
- add_to_favorites(user_id, tid): Enables users to save tweets into their personal lists.

v. Searching and Viewing Tweets:
- search_tweets(user_id): Retrieves tweets based on keyword/hashtag matches, supporting exact and substring searches (Case insensitive)
- show_tweet_details(user_id, tid): Displays tweet statistics (replies, retweets) and offers options for replying, retweeting, or adding to favorites.

vi. User Search and Interaction:
- search_users(user_id): Searches for users by name and displays results with pagination (Case insensitive).
- display_user_details(selected_user_id, current_user_id): Shows a user's profile, including tweet count, followers, and recent tweets, with options to follow or view more tweets.
- view_all_tweets(user_id): Lists all tweets posted by a specific user(pagination).

vii. Managing Followers:
- list_followers(user_id): Displays a list of followers, allowing selection for more details(pagination)
- display_follower_details(user_id, follower_id): Shows details of a follower, offering options to follow or view their tweets.

viii. Favorite List:
- list_favorite_lists(user_id): Displays the user's favorite lists and the tweets stored within them.

ix. Program Execution:
- Main Execution (__name__ == "__main__"):
- Initializes the databas
- launches login_screen() to handle user interaction.

b. Main Flowchart:

Please refer to Appendix A for only the main basic flow of code.

III. Testing Strategy:

We created a data dump and tested the code for potential SQL injection vulnerabilities. The listed cases were tested, along with additional scenarios evaluated on an ad-hoc basis.

Scenario	Test Cases	Expected Outcome
Database Setup	Run Set_Database() with an empty database	Tables should be created successfully without errors
User Signup - Valid Case	Sign up with name="John", email="john@example.com", phone="1234567890", password="pass123"	User should be registered, and a new user ID assigned
User Signup - Invalid Email	email="johnexample.com" or email=John@lkl.com/ phone="abcd1234"	Signup should fail with an "Invalid email/phone Ni" message else user ID should be incremented to next
User Login - Valid Credentials	Correct user_id and password	Login should be successful, and user should access the menu
User Login - Incorrect Password/ Non-existent User	Wrong password for a registered user Login with an ID that doesn't exist	Login should fail with an "Invalid user ID or password" message Should display "Invalid user ID"
Tweet Posting - Valid Tweet/Reply to tweets	Post a tweet "Hello World! #Python"/Reply to Tweets	Tweet should be stored in tweets and hashtags in hashtag_mentions/Update reply
Tweet Posting - Empty Tweet	Submit an empty tweet	Should display "Tweet cannot be empty"
Tweet Posting - Duplicate Hashtags	"Learning Python #Code #code"	Should reject tweet due to duplicate hashtags
Search Tweets - Valid Query	Search for "Python" Seach for #Party	Should return tweets containing "Python" or #Python Should return #Party and not #PartyVibes and not Party
Search for Users/Tweets	Search for "JoH", "Joh", "#PARTY", "#party" are all the same respectively	Should return users with character that matches and are insensitive
Search Tweets - No Matches	Search for "NonExistentTopic"	Should return "No tweets found"
Follow User - Valid Case	Follow another user by ID	Should add a record in follows table
Follow User - Already Following	Follow the same user twice	Should prevent duplicate entries
Retweet - Valid Case	Retweet an existing tweet	Row to be added in retweets table
Retweet - Duplicate Retweet	Retweet the same tweet twice	Should display "You have already retweeted this tweet"
List followers	Also tested various cases on favorite list	Made sure follows are updated
Logout	Select the logout from the menu	Should return to the login screen

IV. Work Break down Strategy:

Work Module	Assigned Member	Function involved	Time Spent	Progress Made	Coordination
Project Setup/Github loading and commits	Vikasinisenthil/ Saachi07	NA	-	Complete	Google Shared document. Chats, emails, zoom meetings, Communication , Github, online meetings
UI Development and planning	All	NA	1 day	Complete	
Test Database creation	All	NA	Ad Hoc	Complete	
Functionality	Saachi07	Search_users List favorite List	3 -4 – Ave hs/wk	Complete	
Functionality	vikasinisenthil	compose_tweet list_followers	3 -4 – Ave hs/wk	Complete	
Functionality	eshaankrishna	Login and sign_up search_tweets	3 -4 – Ave hs/wk	Complete	
Functionality/ Database connectivity	Gskakar	Logout and exits get_database_connection(): Set_Database():	3 -4 – Ave hs/wk	Complete	
Reports	All	NA	Ad Hoc	Complete	
Refinement	All	NA	Ad Hoc	Complete	
Testing and debugging	All	NA	Ad Hoc	Complete	

V. Project Specification:

To the best of our ability, we have strictly adhered to the project specifications. Our design document reflects this by avoiding any additional coding beyond what was required, with the exception of phone number validation. As per the ITU E.164 standard, the valid range for phone numbers has been defined as 7 to 15 digits.

Appendix A