Quacker

Omar Mahmoud, Nasif Qadri, Yousef Moussa

Design Document - CMPUT 291 Mini Project 1

2024-11-14

1 Overview and User Guide

Running the Application

1. **Compile the Application:**
   Use the provided *Makefile* to compile the code by running:

   make

2. **Start the Application:**
   Execute the application with a database filename:

   build/quacker <database_filename>

   Example:

   build/quacker test/test.db

3. **Interacting with the Application:**
   Upon launch, the application displays the main start page, providing options for interacting with Quacker .

2 Software Design

The application consists of the following key components:

Quacker Class

The *Quacker* class is the main entry point of the application and manages user interactions. It provides functionality such as:

- **User Authentication:** Handles login and signup using methods *loginPage()* and *signupPage()*.
- **Feed Management:** Displays the user feed using the *mainPage()* function, and interacts with the *Pond* class for data.
- **Posting Quacks:** Allows users to post and reply to Quacks using the *postingPage()* and *replyPage()* methods.
- **Search Functionality:** Facilitates searching for users and Quacks using *searchUsersPage()* and *searchQuacksPage()*.

Pond Class

The *Pond* class handles all database interactions and encapsulates the data logic. It includes:

- **User Management:** Provides methods for adding users (*addUser()*) and validating login credentials (*checkLogin()*).
- **Quack Management:** Handles operations like adding new Quacks (*addQuack()*), replies (*addReply()*), and managing hashtags.
- **Feed Retrieval:** Generates personalized user feeds (*getFeed()*), combining tweets and requacks in chronological order.
- **Search Operations:** Performs database queries for searching users (*searchForUsers()*) and Quacks (*searchForQuacks()*).

Component Interactions

The application follows a structured flow of data between the *Quacker* and *Pond* classes:

1. The user interfaces with the *Quacker* class, which calls the relevant *Pond* methods for database interactions.
2. The *Pond* class processes SQL queries to retrieve, insert, or update data in the database.
3. Results from the *Pond* class are used to render user-friendly views in the *Quacker* class.

3 Testing Strategy

In testing this program the Python script *populate_db.py* was used repopulate the testing database with new unqiue data improving the manually typed tests ran on the project and increasing the number of tested edge cases. To generate simply run:

python3 test/populate_db.py

The script should then return the data of a randomly generated user. This is to make it easier to locate a valid login to use when entering the program. Example:

Random User Data:
User ID: 53
Name: Deborah Scott
Email: jasminchang@example.com
Phone: 4567624481
Password: E1AjR1ib*t
SUCCESS! Database populated with random test data.

The test script also allows you to easily change parameters regarding the generated data. These can be located in the defintion of *populate_db()* and are self-explanitory as seen below:

```
def populate_db(db_name, user_count=100, tweet_count=500, list_count=200, follow_count
    =300):
```

When it came to testing the application, this was done manually due to the page centric format of our user interface. The felxibilty of testing manually allowed our team to catch unique edge cases that may have passed over an automated system.

4 Group Work Breakdown

We give all credit to Allah SWT, all success comes from Him alone.