CMPUT 291 – Mini-Project 1

Created By:
Raamish Naveed
Weichen Qiu
Solomon Song

General Overview and User Guide

Our program is a command line application that works just like a forum. It allows people to create accounts, login, post questions, post answers, vote on posts, search for posts, etc. A program like this has a lot of benefits. Firstly, a platform like this encourages discussion. This is the main objective. As we dive deeper, we see that collaboration and communication can be improved as well. With older platforms such as emails being slow and unproductive, forums can be one way to improve communication.

When the user starts the program, they can either login or create an account. After a successful login, an ordinary user will be presented with a list of options. They can either post a question, search for posts, answer a question, and vote on a post. If the user is a privileged user, then they will be presented with additional options that include marking an answer as the accepted answer, giving badges to users, adding tags to posts, and editing the title/body of posts. We have implemented some validation features to ensure that the input is correct. However, it is **highly recommended** that the user double checks their inputs and selects the correct instructions on the menus to ensure a smooth run.

Software Design

Our program runs on python and implements the SQLite 3 library. We have created multiple files to reduce redundancy. Our program is linear and procedural. Our program flows from main.py to either User.py or Privileged.py. These are the files needed to run the program:

- main.py
- User.py
- Privileged.py

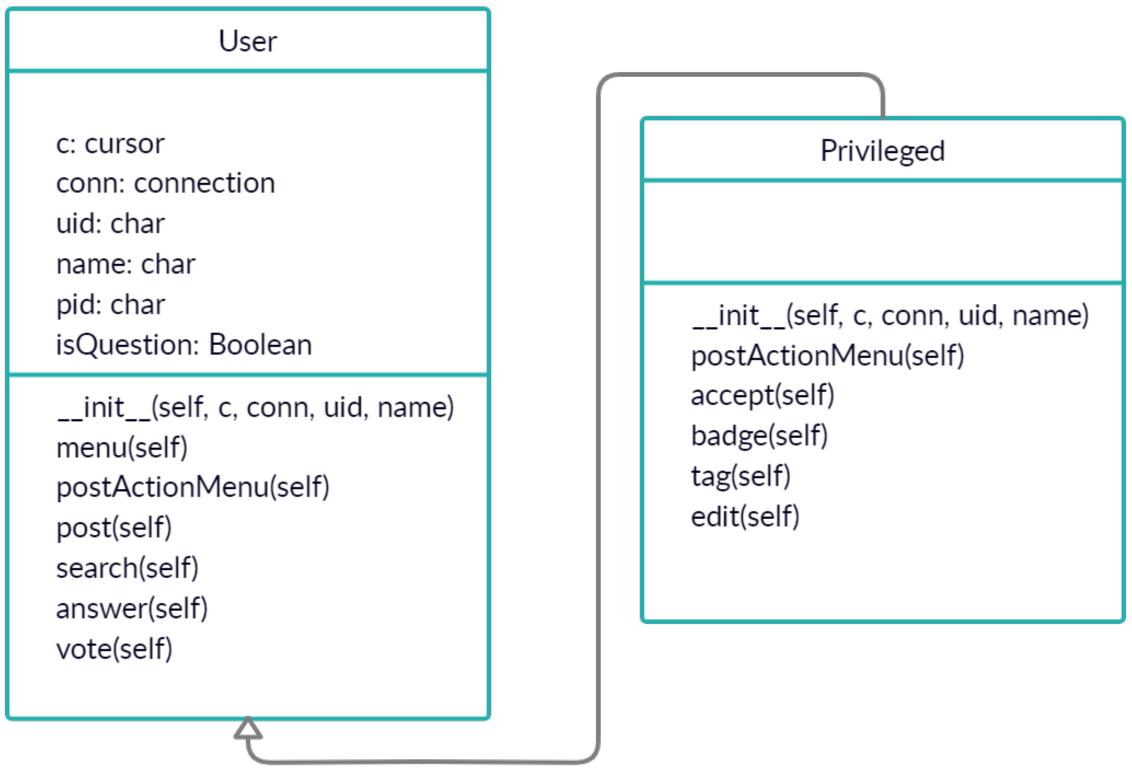
main.py: This is the starting point of this application. It first asks the user to select a database that will be used and then presents the user with 2 options of either logging in or registering a new account. Once a user is successfully logged in, a new User or Privileged object is initialized with user properties and the respective menu function is then called.

User.py: This file has functions that implement the requirements of our program when an ordinary user is logged in. The user is provided with a menu that displays a list of possible actions such as posting a question, searching for posts, posting an answer, voting on posts.

Privileged.py: This file has functions that implement the requirements of our program when a privileged user is logged in. The privileged user is provided with a menu that displays a list of possible actions such as the ones listed in User.py. Additional functions are also included such as marking an answer as the accepted answer, giving badges to users, adding tags to posts, and editing the title/body of posts.

To avoid SQL injections, we decided not to use Python string operators to pass variables into the SQL query. Instead, we used named/positional parameters in execute(). As we learnt, this automatically escapes special characters in variables.

We approached the design using object-oriented programming because it allowed more structure but more importantly, it allowed Privileged users to inherit shared methods from Users. Here's a diagram that visualizes this:



Testing Strategy

We followed a sequential approach and tested our program manually step by step. Initially we created a bunch of ordinary and privileged users. Next, we logged into an ordinary user and posted a bunch of random questions to create some test data in our database. About 20 questions were developed for an ordinary user. We were also constantly checking our database to see if the data was actively being added to it or not. Next, we tested the search function to make sure that it works. We played around with our options and kept discovering bugs and fixed them as we went along. After that, we selected a post and tested out voting function, which seemed to work perfectly fine. We went back to the main menu, searched for another post, and decided to post a couple answers to our questions. More bugs were discovered and fixed. We followed the same approach for privileged users until all the functions seemed to be working fine. The class structure was useful in isolating each function as this helped us thoroughly test independently before integrated into the class. Towards the end, we tested scenarios where users may make an error and took preventative measures to prevent the code form breaking.

Running Instructions

Python 3.5 is required for the program to work. Please run the following command in terminal:

```
python3 main.py database.db
```

The database must be in your program directory.
Make sure to change the name of the database in the command above.

Group Work Break-Down Strategy

We created a repo on GitHub to ensure everything was accessible to all of us. Along with that, all of us kept separate backups of our files on our personal computers to ensure nothing was accidentally lost or deleted. We used a group chat to coordinate and make sure everything was going smoothly. We organized a group meeting on call to split up tasks and follow up meetings every few days to ensure that everyone was on track. We thought it would be fair if everyone did an equal amount of work, that's why we tried to split up tasks equally. Roughly, each of us spent around 25-30 hours in total on our separate parts of this assignment. Person responsible for the following tasks:

Weichen Qiu:

Setting up GitHub repository
Main.py
Post action-Vote function in user.py
Post Action-Edit function in Privileged.py
Additional testing
Fixing Bugs

Raamish Naveed:

All the functions in the User.py file besides Post Action-Vote
Verifying for SQL injection
Additional testing
Fixing bugs
Writing the Design Document

Solomon Song:

All the functions in the Privileged.py file besides Post Action-Edit
Additional testing
Fixing bugs