## Overview of system

Our system consists of the following functions:

a) **main():**

Program connects to the database from this function. Then, this function prompts user to enter their UID. After it gets the UID from the user (possibly anonymous), it shows the main menu and asks user what they want to do.

b) **get_uid():**

This function asks user to enter their UID. UID becomes None if user wants to login anonymously. Then, returns the UID.

c) **show_main_menu():**

This function shows the main menu. Then asks user what they want to do next. And return their choice. Users can choose between posting a question and searching for questions.

d) **show_action_menu():**

This function runs after user wants search for questions. It asks what they want to do with the question. Users can answer the question, list all the answers that question have, vote the question, or go back to the main menu.

e) **post_question(db, uid):**

This function runs after user decides to post a question by pressing ‘p’ from the main menu. Then it asks user to enter a title and a body text for the question. User can also add tags to the question. Posts are stored in Posts collection and Tags are stored in Tags collection.

f) **search(db, uid):**

This function runs after user decides to search for questions by pressing ‘s’ from the main menu. It asks user to provide one or more keywords. Then, it searches those keywords in all questions and displays some of the attributes of questions that contains those keywords. Then, it asks user to choose a post to show all attributes of the question.

g) **post_answer(db, uid, qid):**

This function runs after user selects a question in search(). User enters a title and a body text for the answer. Answer posts are stored in Posts collection.

h) **vote(db, uid, pid):**

This function runs after user selects a question in search() or after user selects an answer after list_answers(). User votes for the post. Vote is stored in Votes collection and Score of the post gets incremented by 1.

**i) list_answers(db, uid, qid):**

This function runs after user selects a question in search(). It prints first 80 character of each answer of the selected question. User can select a post to see all attributes of the post. Then it asks user if they want to vote for the post.

## User Guide

1) In the terminal type the following command:
    a. python3 <program_name>.py
2) Enter the port number you want to connect.
    a. Enter 27012 for default port.
3) The program starts by asking user to enter their user id.
    a. Respond by typing your user id.
    b. If you want to login anonymously, press ‘a’.
4) Once you have completed step 3, three options will appear:
    a. Press ‘p’ to post a question.
        i. You will be instructed the title and the body of the post you wish to submit.
        ii. You will be prompted once the question has been successfully posted.
    b. Press ‘s’ to search for questions by entering keywords.
        i. You will be prompted to enter one or more keywords.
        ii. You will be asked to select one of the questions which appeared on your screen to see all attributes of the question.
        iii. Once you select a question, five options will appear :
            1. Answer the question with title and body (‘a’)
            2. List all answers of the questions (‘l’)
                - After you list all answers, you can select an answer to view all attributes of it and vote on it as you wish.
            3. Vote on the question (‘v’)
            4. Go back to the main menu (‘main’)
            5. Terminate the program (‘q’)
    c. Press ‘q’ to terminate the program.

## Testing strategy

a) Initially we followed the marking rubric in chronological order. We took note of any unexpected behaviours during this initial period of testing. The functions were used as intended (no typos, no intentionally wrong inputs).
b) Once we had established that the program was behaving as expected for good inputs, we moved onto stress testing with wrong/unexpected inputs. By feeding the menu unexpected values, we hoped to uncover more subtle/hidden errors

a. Our main goal was to catch issues in the menu navigation system. This was the case as a result of a few instances where an unexpected input resulted in an unfamiliar error / infinte loop.
b. Using try, except statements in conjunction with else clauses (which handle unexpected input), we were able to successfully fix all the errors which had been brought to our attention.

## Group work break down:

- We communicated on a daily basis throughout the past 10 days using google hangouts, and email.
- Check in's occurred almost every day (especially closer to the deadline).
- During the daily check in's on google hangouts, we would share screen's in order to assist each other with bug fixes and planning.
- We kept track of contributions using a private github repository and daily code reviews using the screen sharing feature on google hangouts.

## Distribution of work:

vwsmith:

- Figured out how to connect to the MongoDB servers with a port number.
- Created collections and filled them with the data in json files.
- Prepared the user login report.
- Coded the search() function.
- Prepared this report

melih:

- Prepared the main menu and the action menu
- Coded the post_question() function.
- Coded all of the action functions.
- Merged search() function and action functions.
- Prepared this report

## Time Allocation:

- It took almost 2 days to figure out how we should connect to the database and load the data to the database. (by wwsmith)
- It took an hour to prepare both menus and merge them with other functions. (by melih)
- It took 2 hours to code the post_question() function. (by melih)
- It took almost 7-8 hours to code all of the action functions and debug them. (by melih)
- It took 2 hours to prepare this report. (by both of us)
- It took 2 hours to prepare the user login report. (by vwsmith)
- It took 4-5 hours to code the search() function and debug it. (by vwsmith)