DESIGN OF SOFTWARE

Created three functions: create_user(), login_menu() and login(). The login_menu() function welcomes the user into the application and asks whether the user is registered or not. If a user is not registered, the system carries out the create_user() function. Create_user() function allows a person to create a user, entering their user ID, their username, a password and the city they live in. Once inputted, the data is inserted into the database. After, the system takes the user back to the login_menu() function. If registered, the user is taken to the login() function where the system asks the user to enter their user ID and password. If the user enters in the right user ID and password, the system takes them to the main menu. If not, the system asks to try again. If the user would like to quit from login_menu(), all the user would have to type is ‘quit’.

Options After logging in:

The system is designed to make sure that after the user logged in, the user is taken to the function, main_menu(). In that function, there are 2 options given to the user: Post a question, and Search for posts.
 If the user picks the first option, the system takes the user to the post_question() function. That function asks the user to enter a title and a body for the post. Once inputted, the system assigns the post with a unique ID and the post date is set to the current date that the post is posted. After, the system takes the user back to the main_menu(). If the user picks the second option, the program takes the user to the search_post() function. This function asks the user to provide one or more keywords so that the database can find 5 posts with the most matches within the title or body of the post or the tag. After printing out the 5 posts, if the user is a normal user, the system gives 2 options: Post an answer for the selected post if it is a question , vote on a post. If the user is a privileged user, the system gives 6 options: Post an answer for the selected post if it is a question, vote on a post, mark an answer post as the accepted answer, give a post a badge, add a tag to a post ,edit a post. If the user enters quit, the program leaves the function and goes back to the login_menu().

Options if User is a Normal user:

As stated above, there are two options given to the user. If the user picks the first option, the system takes the user to the post_action_answer() function. The function checks if the selected post is a question or an answer. If the post is a question, the function asks the user to enter the title and body of the answer post for the question post. Once inputted, the system takes the user back to the search_post() function. If the post is already an answer, the function goes straight to the search_post() function. If the user picks the second option, the system takes the user to the post_action_vote(). The function checks if the user has already added a vote to the post. If so, the function takes the user back to search_post(). If not, the function automatically adds a vote to the post and then takes the user back to search_post().

Options if User is a Privileged User:

As stated above, there are 6 options given to the user. The first two options have been explained in the paragraph above. If the user picks the third option, the system takes the user to the post_action_accepted(). The function checks if the post is a question or an answer. If it’s a question, the system takes the user back to the search_post(). If it’s an answer, the system checks if there is already an accepted answer for the question that the user is answering and if not, the system sets the answer as the accepted answer. After, the system returns to the search_post() function. If the user picks the fourth option, the system takes the user to the post_action_give_badge() function. This function prints out all the badge names and badge types. The function asks the user to pick a badge to attach to the selected post. If the badge is not already attached, the system gives the badge to the post. After, the system returns to the search_post() function. If the user picks the fifth option, the system takes the user to the post_action_tag(). This function asks the user to enter a tag name to assign to the selected post. If the assignment is successful, the system returns to the returns to the search_post() function. Finally, if the user picks the final option, the system takes the user to the post_action_edit(). The function asks the user to enter a new title and body for the selected post. Once updated, the system returns to the search_post() function.

Test plan:

Action to be Tested	**Test Method**
Check whether the username and the password work in login menu	Run the program and check for false positives and true negatives when entering into the system.

Bugs: Stuck in an infinite while loop	
Check whether the user is able to sign up and create a new account. After creating that account is able to login into the program Bugs: Initially implementation did not go as planned due as we had a validation check checking if the user was normal or privilaged. There was also a slight bug during the insert query that updated the new user info to the database.	Fill out the criteria in the create_user(). Use that information to login in. If login is successful, the system allows for the creation of new users. As developers we checked this functionality in SQL to see if user creation is happening successfully.
Check whether Main menu allows user to pick option that they desire and directs to right function Bugs: Stuck in an infinite loop. In the initial implementation did not pass userid which resulted in the entire application to crash.	Login in and test the main function twice. In the first test by entering '1', the main function should redirect the user to the post_question(). Likewise, by inputting '2', the main function should redirect user to the search_post()
Check if post_question correctly lets the user post a question Bugs: Hiccup with using the positional parameters and passing user input through it, initially crashed due to improper implementation. We ran into the binding bug for positional parameters due to improper implementation.	Input the title and the body of the question post. Once inputted, a message of either a successful posting or unsuccessful position should pop up and the user would be redirected to the main menu. As developers, we checked this functionality in SQL to see if a question was posted successfully.
Check if search_post carries out its functionality Bugs: The initial query used for both finding matching keywords and listing out posts based on the schema both produced bugs. The query that dealt with matching the keyword produced incorrect results as instead of counting posts that contained the user inputted keyword it somehow counted all the posts. This issue was just a simple parenthesis issue. The second query was implemented totally wrong to begin with. We did not use a left outer join due to which our results were swayed. The implementation of the left outer join seemed to fix the existing issue.	Run the search_post function. The user can see a series of posts that are matched by the inputted keyword. The user can pick a post of their choice from the list of available posts. If the user enters a post that is not displayed an error message pops up. If a correct post is entered then the user can perform actions on the post. If the user enters a post-action that does not exist, an error message pops up. The number of actions a user can perform on a post depends on whether the user is a privileged user or a normal user. Once a user inputs an action they want to do they should be directed to the appropriate function related to the action. We tested this by going through the steps above numerous times and seeing if there were any unwanted bugs that came up.
Check Normal user : post action - Post_action_answer - Post_action_vote Bugs: The user response got stuck in an infinite while loop. There was a similar binding issue with positional parameters that was fixed later on. There were several bugs with the validation of user input all of which came down to silly mistakes, indentation or being stuck in loops.	This can be tested by continuing the steps in the above test case. Once the user is redirected to the appropriate function, the user is given a prompt explaining the details of that function. A normal user can only perform the two post actions. If the user selects a post_action_answer, user must input a title and a body. If the post action is unsuccessful a message will pop up and after doing so the user will redirect the user to the user menu. If the post action is unsuccessful, a message will pop up and after doing so the user will redirect the user to the user menu. As developers, we can check the rightful insertion in sqlite under the posts, answer table to see if everything was inserted successfully. If the user selects post_action_vote the system will check if the post selected is not already voted. If the post selected has already been voted on then an error message will pop up and the user will be redirected to the user menu. If the post selected does not have a vote the system will assign the vote and the user will be notified of the result, the user will then be redirected to the user menu. As developers we checked this in sqlite under the votes table to see if a post was getting a vote assigned successfully.

Check Privileged user: post action - Post_action_answer - Post_action_vote - Post_action_mark - Post_action_give_badge - Post_action add_tag - Post_action_edit Bugs: Similar to the normal user function the privileged function had similar errors. The user response got stuck in an infinite while loop. There was a similar binding issue with positional parameters that was fixed later on. There were several bugs with the validation of user input all of which came down to silly mistakes, indentation or being stuck in loops.	If a user enters a post-action that does not exist an error message pops up. The post_action_answer and post_action_vote are tested and function the same way as they do for normal users. For post_action_mark a user can either set their current answer as the accepted answer if there isn't an answer for the post or a user can change the answer for a question with an answer. If the user is successful or unsuccessful they will be notified through a message and will be redirected to the user menu. When a user selects post_action_give_badge they can give a badge to a post from a list of given badges. If a user inputs a badge not on the list an error message pops up directing the user to try again. If the badge is assigned successfully or unsuccessful the user will be notified and will be redirected to the user menu. When a user selects a post_action_add_tag they can assign tags for their post. If the tag assignment is successful or unsuccessful the user will be notified of the result and then will be redirected to the user menu. Post_action_edit allows the user to edit the title and body of their respective posts. If the update process is successful or unsuccessful the user will be notified of the result and will be redirected to the user menu. As developers we tested each of these post questions by going to the appropriate tables in sqlite and seeing if the functions behaved the way they should.

Assumptions Made:

- Assumed that the values from the database are not NULL
- Assume there's a specific numbered approach to IDs for all tables so as to auto-generate IDs for votes, posts, etc.
- Assume users will pass the database like: "python3 mp1.py *name of database*"
- Assume that the database will also be in the same folder as the program.