CM50267 - Software Technologies for Data Science

Mini Project Report

Extending on the skeleton that was provided for this assignment I have added the functionality required through the given tasks to perform they expected operations. The server.py file handles the user logging in and out, giving them the ability to add and undo traffic records whilst also providing them a thorough summary which they can check to view the recorded traffic on a user's current session. All whilst still maintaining the integrity of the sessions by validating and handling the parameters and cookies passed through the web browser and requests.

Database Structure

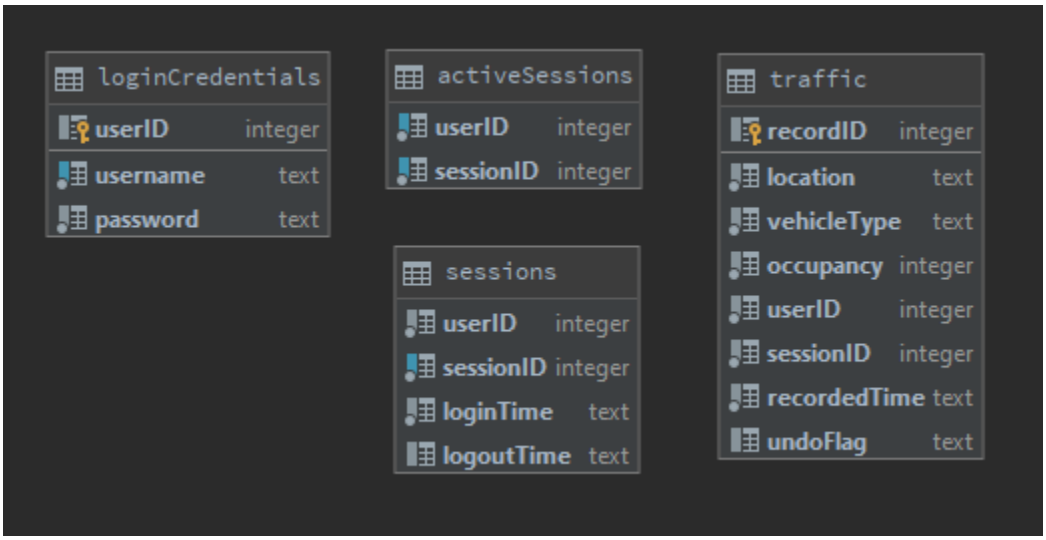

Figure 1 - database structure

The structure of my database as also shown on Figure 1 consists of 4 tables. The 'loginCredentials' table stores the userID as an auto-incremented primary key with the username and the password which is stored using Sha256 Hash encryption. The 'activeSessions' table contains each user why is at that time logged in with their userID and their sessionID which is a randomly generated number using the uuid library generated during login. For the 'sessions' table, it keeps a record of the user's unique session with their login and logout time, using the userID, sessionID and corresponding login and logout times. Lastly, for the 'traffic' table the recorded entries include location, vehicle type, occupancy and time of the recording as well as the userID and the sessionID making it able to possible to identify which user created the record and which other records they created during the same login session. For each entry it also stores an auto-incremented recordID as a primary key to distinguish between the different entries.

Code Description

Logging in

When the user tries to log in by entering the username and password my code queries the database with the username and a sha256 hashed version of the password to check if there is a match in the 'loginCredentials' table. Then it checks if the user is already logged in; maybe from another browser in order to handle giving an appropriate message. If the user is logged in from another browser, their session will be stored in the 'activeSessions' table so a message ' User Already Logged In' will show and reject the user's login attempt. If there are not active sessions for that user, they are logged in, their active session is added to the 'activeSessions' table with the user's userID and randomly generated sessionID, whilst also storing a record in the 'sessions' table with the user's userID, sessionID and login time.

Logging out

When the user finally logs out, their session is deleted from the 'activeSessions' table and the 'sessions' table is updated with their logout time.

Handling Validation

At each command either login, undo, add, back, summary and logout there is always the handle_validate function that checks if the current session has the correct credentials to continue. The function checks if the userID and randomly generated sessionID which was created during login are currently in the 'activeSessions' table. If it finds a matching result it returns True continuing with the commands properly. If not, False is returned and the user is redirected back to the index page.

Add

For adding a traffic entry my code retrieves the location, vehicle type and occupancy through the 'parameters' parameter. Additionally, it finds the time at that instance of the recording and all 6 fields with the inclusion of the userID and the sessionID which are also passed as parameters in the function they inserted into the 'traffic' table as a new entry.

To get the total we count the number of entries for the user at that specific session which their undoFlag does not contain anything.

Undo

In order to undo a previously entered entry the user has to first enter a matching location, vehicle type and occupancy which they had previously added in their current session. If there are matching instances, they order them by the recordID field, and the results are limited to just 1 to get the latest entry in the table. Then the undoFlag field is updated for that specific entry to contain 'undoed' to indicate the undoing of that record.

Again, the total is updated using the same query as in the add function.

Summary

When the user choses to see the summary of their session my code counts the number of entries for each vehicle type adding it in the response refill and at the end it adds the total for all vehicle types for them to be filled in the corresponding placeholders in the summary page.

Handling Malicious / Bad Input

Handling malicious input in the queries is very important in preventing SQL injection attacks. The easiest method I had found was in the SQLite's API python documentation in the 4th example. It describes putting a ? as a placeholder for values, and to provide a tuple of values as the second argument to the cursor's execute() method. This was very easy to implement using the previously provided functions in Lab 5 as I just manipulated the functions to include the tuples of values and was enough to solve the issue of malicious input for the queries.

For the instance of empty inputs in the fields of username, password and location I was checking if their corresponding parameter was present in the 'parameters' dictionary that was passed inside the functions. Additionally, in my code I do lowercase the location input regardless if they had entered capital letters or not because I expect 'Bath' and 'bath' to still indicate the same location.