University of Alberta

George Nassar
Nathan Anderson

SQL Inside Applications: Database Operations in Python

Overview:

We have developed a user-friendly software that allows one (with the proper credentials) to perform a variety of tasks on an SQL database. There are two user types implemented in the software: The Registry Agent (RA) and the Traffic Officer (TO). The Registry Agent is capable of database operations such as registering births, registering marriages, renewing vehicle registrations, processing bills of sale between vehicles and owners, processing ticket payments on vehicles that have outstanding balances, and retrieving a general abstract for drivers registered in the database. The Traffic Officer is capable of more front-end oriented operations, such as issuing a ticket for a traffic violation and retrieving information on car owners based on vehicle descriptions. The RA and the TO are limited to accessing their respective operations.

User Guide:

1. To begin, open your operating system's command line interpreter. For windows users this is the Command Prompt or "CMD", for Unix and Unix-like systems this is the Unix Shell, and for macOS this is Terminal.
2. Navigate to the program's directly within your interpreter. This directory should include the software (miniproject1.py) and your sqlite database (file type .db).
3. Run the software with the following command "python.exe miniproject1.py xx.db" where xx is the name of your database. It is important to note that this software requires python 3+ to run.
4. The software will prompt you for a username, followed by a password. The password is case sensitive.
5. Dependant on your successful login, as well as the credentials of your user type, you will be presented with 6 input options. For the RA, input numerical values 0 through 5 to perform your various functions, or 6 to log out. For the TO, input numerical values 0 or 1 to perform your functions, or 2 to log out.

Software Design:

Our software was created with explicit ease in mind. Inputs are recorded on separate lines, with explicit instructions to reduce user-to-user variability in input formatting. When the formatting of an input seems vague, we include a formatting hint in parentheses to aide the user.

Inside the Program:

Our program was created in a single python file. After importing numerous libraries, we connect to the sql database using the sqlite3 library functions. The first function call inside main() is the login(), barring access to the contents of the other functions without the proper credentials. After the user has logged in, there is a function to display their respective options, dependent on user type. For the RA, the

agent_prompt() function displays and calls to each respective RA operation function. The string input of the user is converted into an integer to match with the numerical options in the prompt(). When a pertinent value is entered, the user moves into a specific RA or TO function, dependant on user type. Within each function there are numerous python sqlite3 module statements coupled with string formatting statements and other python functions. Input errors and type conversion errors are handled within each respective function, and when the function reaches an end point the program returns to the respective user prompt() function, displaying user options once more, with the option to log out. Upon logging out, the user is returned to the login() function, so that they may log in with different credentials if desired.

Testing Strategy:

To test our program, we used a database from one of our previous projects that retained many of the same values and constraints that this project required. Ryan Kang, a classmate, shared a comprehensive database containing various database entries for the purpose of testing, and it served us well again for this project. Where the mentioned database did not satisfy all of the requirements for this program, dummy values were entered using an application called “DB Browser for Sqlite” (https://sqlitebrowser.org/), simply for ease of insertion. Rogue, nonsensical inputs were tested manually and corrected for.

Group Work Strategy:

The project was broken down into what were mostly two parts. Nathan Anderson created the initial python file, and by extension the overarching skeleton of the program. This encompassed the login function as well as the framework for the display() and prompt() functions, as well as set the logical flow of the project. The rest of the workload was split in two, with George Nassar completing the Registry Agent functions, and Nathan completing the Traffic Officer functions. A great deal of collaboration between the two of us was necessary to complete the project, however the bulk of the work was split up individually. George volunteered to do the report, as he enjoys writing. The time spent for each member was about 20 hours, with the greatest amount of progress happening towards the end of the project, as we became more familiar with using SQL inside Python.

The chosen method of coordination was the chat messenger app Discord. Discord allowed us to share code and discuss the project in a comprehensive manner, and its mobile availability was chiefly useful. Github was used to collaborate on the actual program, with separate branches made for our individual workloads that were then combined into the master branch for submission.