Report for CMPUT291 mini-project 1

Autor: Puyuan Liu, Xingran Fang

Date: 2019.11.4

1. Overview

This is a report which introduces the feature and functionalities of the python script *mini.py.*

This report serves as a brief user guide for the script. The following contents include the description of the functionality, diagram of the script, and the description of the detailed design of the primary function,

testing strategy and also the group work strategy.

When the script starts, with a given SQL database which is consisted of tables shown in the following link:<https://eclass.srv.ualberta.ca/pluginfile.php/5149368/mod_page/content/18/prj-tables.sql?time=1571182846772>, it will first ask the user for correct username and password to log in. After successful login, the user should be able to *Register a birth, Register a marriage, Renew a vehicle registration, Process a bill of sale, Process a payment, Get a driver abstract if he/she is a* Registry agent. Moreover, the user would be able to *Issue a ticket, Find a car owner if he/she is a* Traffic officer. [1] The script will quit at any time when the user enters the exit command.

A brief diagram which describes the main process flow of the script is shown below:

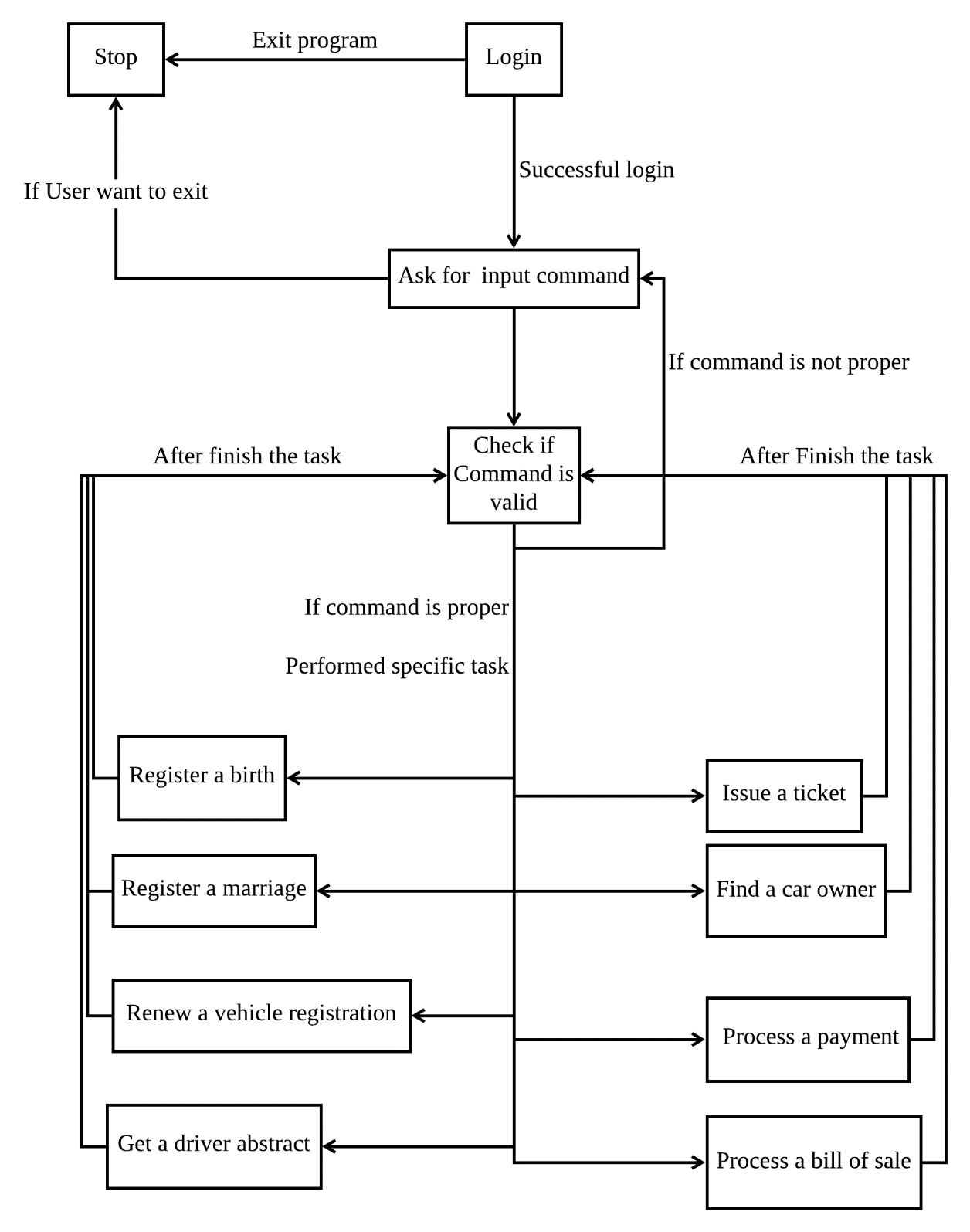


Figure1. Main Flowchart Diagram for the Script.

1. Description of the Detailed Design

There are total 12 primary functions: ***main, read\_file, check\_log\_in, handle\_command, do\_RAB, do\_RAM, do\_RAR, do\_PAB, do\_PAP, do\_GAD, do\_IAT, do\_FAC*.**

\*The last 8 functions perform the 8 different tasks mentioned in the overview respectively, and all of them share a similar structure. Since this is a brief report, we won’t go through the details about how exactly each of these 8 functions archives its task, we will devote most attention to their shared structure.

The ***main*** function controls the main process of the script, it takes the option that the user entered in the terminal as the input and doesn’t have an output. Currently, the only valid option is***-i filename.db***, where ***i*** stands for input, ***filename.db*** is the filename of a valid SQL database.

The function first checks whether the given option a valid. If the option is valid, it calls the ***read\_file*** *function* to check whether the input filename is valid. If the filename is valid, it then calls the ***check\_log\_in*** function to check whether the user can provide the right user name and password to log in. If the user successfully login, it will print a welcome message, all possible commands and the corresponding description for each command as the figure shown below:

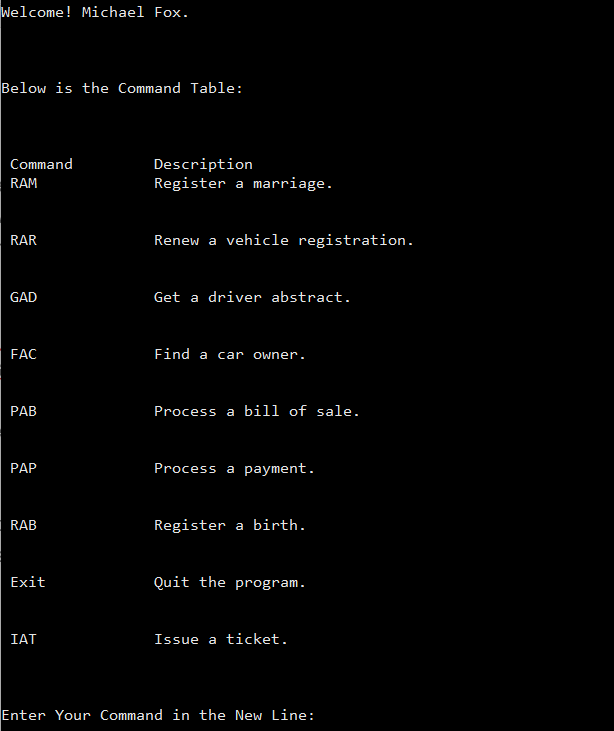


Figure2. Welcome message and command massage for the ***main*** function

Then it will ask for the command that the user wants to execute. After getting the user’s input command, the function passes the command to the ***handle\_command*** function. If the command is handled, either successful or unsuccessful, it will ask for a new command and repeat the process (ask command then execute the command) mentioned above. When the user enters ‘Exit’, no matter in uppercase or lowercase, it will exit the program.

***read\_file*** function checks whether the given filename refers to a valid DB database.

The input is the name of the database filename, the output is a Boolean indicator: True if it’s indeed a valid database and False if it’s not. If the return value is False, a warning will be printed to the terminal which states that the given database is not valid.

***check\_log\_in*** function checks whether the given username and password match with any record in the user table in the given database.

The function takes no input, return True if the user indeed entered a right username and password. It keeps asking the user for the username and password until the user enters the correct one. When the user entered the wrong password, a warning will be printed to the terminal. Furthermore, when the user enters an exit command, it directly terminates the script.

***handle\_command*** function checks whether the given command is valid.

The input is the command string, it doesn’t have an output. If the command is not valid, it prints a warning message to the terminal and returns. If the command is valid, the function calls the right execution function (e.g. ***do\_RAR***) to execute the command. If the execution function returns False, the function prints the warning message which states that the command didn’t go through successfully. If the execution function returns True, the function prints the congratulation message. If the user entered an exit command, it terminates the program.

***do\_RAB*** function performs the operation for Register a birth.

The input is user city, user type. The output is False if the command didn’t execute successfully, True otherwise. A warning will be printed if the user is not a registry agent. The function will ask the user for some required information. These user inputs will also be checked to ensure they meet the required format of the corresponding field in the SQL database. After getting the right input, the function will perform a certain task and update the SQL database. If the command didn’t execute successfully, the reason for the failure will be printed into the terminal. Otherwise, it prints the update of the database.

***do\_RAM, do\_RAR, do\_PAB, do\_PAP, do\_GAD***, has the same structure as ***do\_RAB***.

The only difference will be they are performing different tasks. The task that they are performing is: *Register a marriage, Renew vehicle registration, Process a bill of sale, Process a payment, Get a driver abstract,* *r*espectively.

***do\_IAT*** and ***do\_FAC*** do a similar thing as well. Except they will check whether the user is an officer, if not, a proper warning will be raised. They perform the task *Issue a ticket, Find a car owner, r*espectively.

1. Testing Strategy
2. Work Break-Down Strategy
3. Appendix:

1. *Except for the password which is case-sensitive, all other string matches (include user id, name, etc.) are case-insensitive. This means edmonton will match Edmonton, EDMONTON, edmontoN and edmonton, and you cannot make any assumption on the case of the strings in the database. The database can have strings in uppercase, lowercase or any mixed format. [1]*

2. Warnings will be raised if an improper command or input is entered.

3. The program is not invincible for injection attacks.

Reference:

1. Spec of the CMPUT mini-project: <https://eclass.srv.ualberta.ca/mod/page/view.php?id=3659763>