Mini Project 1

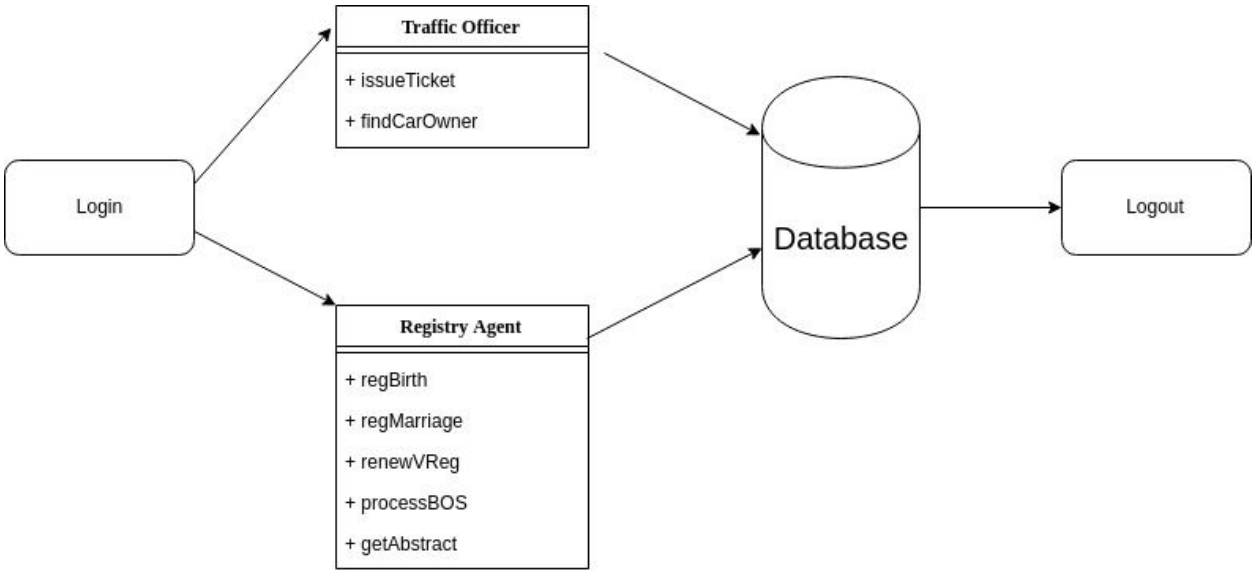
A design document submitted in partial fulfilment of
CMPUT291

November 4th, 2019

Raynard Dizon
Daniel Florendo
Ibrahim Aly

System Overview

Mini Project 1 is a command-line interface system that manages enterprise data and provides services to its users. Users are divided into two categories, registry agents and traffic officers, and each contains various services to manage data. Users are required to provide a username and password to access the services. Once logged in, the user is prompted to enter the command for the service they wish to access. The figure below depicts the flow of data in the system.



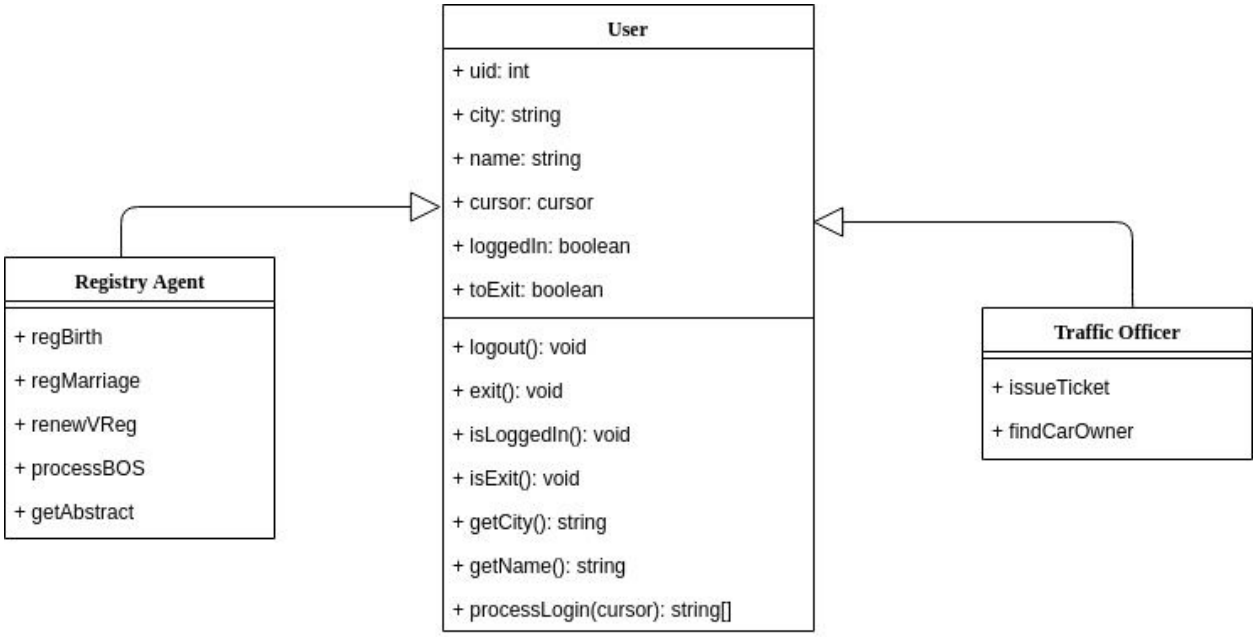
Design

Python is the host language of the program's database. It was chosen because class and method implementations are fast and simple; sqlite3 was chosen as an appropriate database as the program is relatively small.

To allow simultaneous development in different sections of the program, classes and methods were divided into multiple sections. The *agent* directory contains `agent.py` which serves as a class for the agent methods. A *methods* subdirectory contains the class methods, one file for each service, available to the agent user. The `helpers.py` is also located in the methods directory. It contains miscellaneous methods such as `getPhone()` and `checkPerson()` that are used in implementing the agent methods. The *officer* directory contains `officer.py` and is structured in a similar manner to the agent class with the exception that its methods subdirectory does not contain a helper file. This is because the officer methods require less helper methods to implement and thus, can simply import necessary methods from the agent's helper file.

Both agent and officer are subclasses of the superclass *user*. The file `user.py` contains the superclass and is responsible for facilitating the login process. When prompting the user for a

password, the input is masked for security. The following figure depicts the hierarchy of the classes.



Lastly, `main.py` is responsible for the database connection. It serves to create a *cursor* to the database and identifies the user as an agent or officer. Modifications to be made to the database are committed here as well.

To run the program in Linux, open a terminal in the root directory and run the code below:

```
$python3 main.py prj.db
```

Enter the credentials for an agent or officer. The terminal will ask the user for the service command. The user may type ‘help’ to see the available options. Once a valid command is entered, the user can follow the terminal’s prompts. To exit the program, simply type ‘exit’.

Testing Strategy

Existing data from a previous assignment that has a similar relational schema was used in testing the program. For each testing scenario given in the marking rubric, appropriate data was generated to recreate the scenario. After verifying the program passed the rubric, the edge cases were tested. Examples are attempting to retrieve the abstract of a driver not registered into the database, processing a payment of $0, or renewing a vehicle that is not registered. Testing against simple SQL injection attacks were made as well.

Group Work Strategy

The project break-down between partners, including time spent for each task, is as follows:
Raynard Dizon:

- Implemented five agent functions: register a birth, register a marriage, renew a vehicle registration, process a bill of sale, process payment (3 - 4 days)
- Testing (1 day)

Daniel Florendo:

- Implemented one agent function: get a driver abstract (1 day)
- Testing (1 day)
- Wrote the design document and the README file (1 day)

Ibrahim Aly:

- Code restructuring (1 day)
- Wrote all officer functions: issue a ticket, find a car owner (2 days)

Various software were used to enable coordination among group members. GitHub was used to order code updates and facilitate remote development. Trello, a job board, kept track of the team goals while scrum calls were done in Messenger.