Mini Project 1 Design Document

ridderik | yuhang5 | wstix

OVERVIEW

Our system completes the requirements as specified on eclass. Our system is a program that allows the user to do work with the data given by our Professor and our TAs. Our system allows the user to create a new account as either a registry agent, or a traffic officer. Once the user creates an account the system has the following functionalities:

1.1 Register a birth

1.2 Register a marriage

1.3 Renew a vehicle registration

1.4 Process a bill of sale

1.5 Process a payment

1.6 Get a driver abstract

2.1 Issue a ticket

2.2 Find a car owner

Please note, the 1.x functionalities can only be performed by registry agents, and the 2.x functionalities can only be performed by traffic officers.

USER GUIDE

1. Run the program by typing the following:

```
python3 menu.py ./“name of database”
```

Where “name of database” is the name of your database file that is located in the same directory as the 3 python files.

2. Log in to the system, or create a new account by following the prompts on-screen.
3. Choose the functionality from the list by typing in the number of the functionality.
4. Follow the prompts on screen to complete the task.
5. Exit the system from the main menu or the agent menu, or the officer menu whenever you are done by following the prompt on the screen.

DETAILED DESIGN

We wrote our system in Python3. Our program is split between 3 files, a file for the menu, a file for the queries used by the registry agents, and a file for queries used by the traffic officers. Each file has a main class that contains the methods used by that section. The menu class is broken up into main_menu(), a_menu() and o_menu() methods for logging in, doing agent queries, and doing officer queries respectively. The menu class has one parameter, which is a string indicating what the name of the database file to open is. This class handles the opening of the database, the creation of new user accounts, and the setup and navigation of the menus and login screens. The main_menu() method is responsible for performing logins, and determining which menu to proceed to. It also is responsible for creating new users, by calling the register_user() method within the menu class. The a_menu() method is responsible for asking the user which query they would like to perform, and then calling upon methods in the agent_queries class to perform the queries. The o_menu() method is responsible for asking the user which query they would like to perform, and then calling upon methods in the officer_queries class to perform the queries. The registry_agent class is broken up into six main methods, one for each application functionality. register_birth() is the first method, and it is responsible for registering a new birth. register_marriage() is the second method and it is responsible for registering a new marriage. renew_vehicle_registration() is the third method and it is responsible for renewing the registration of a vehicle. process_bill_of_sale() is the fourth method and it is responsible for processing a bill of sale when a vehicle is sold from one person to another. process_payment() is the fifth method and it is responsible for processing a payment towards a ticket. get_driver_abstract is the sixth method and it is responsible for displaying the number of tickets, the number of demerit notices, the total number of demerit points received both within the past two years and within the lifetime for a person. There are also helper methods in addition to the six methods for the six functionalities. The traffic_officer class is broken up two methods, again, one for each functionality. issue_ticket() is the first method and it is responsible for issuing a ticket to a vehicle. find_car_owner() is the second method and it is responsible for finding the owner of a car. It also has helper methods for the functionality methods.

TESTING STRATEGY

We each tested our code separately, using the menu that Yuhang created to navigate through the database, and the data that Warren created, based off of some test data that a fellow student (Jacques Leong-Sit) created and posted to the eclass forum. When we met and combined our code, we had to ensure everything worked together as expected, and we had a couple of roadblocks, such as some lowercase characters not being recognized by our program.

We set up our program to prevent SQL injection attacks, but were unable to test these due to time constraints. We thoroughly tested uppercase and lowercase inputs, and these should not cause any issues. We thoroughly tested non-valid character inputs, and these should not cause any issues.

GROUP WORK STRATEGY

We broke up the project into Story Points, and each of us were assigned an equal amount of Story Points.

The breakdown is as follows (sp means Story Points):

- Register a birth: 2sp, register a marriage: 1sp, renew a vehicle registration: 1sp, process a bill of sale: 2sp, process a payment: 1sp, get a driver abstract: 3sp, issue a ticket: 1sp, find a car owner: 2sp

And each person's work is as follows:

- Cameron: register a birth, register a marriage, renew a vehicle registration, create design document - took approximately 15 hours (approximated by Cameron)
- Warren: process a bill of sale, process a payment, find a car owner, create testing data - took approximately 10 hours (approximated by Warren)
- Yuhang: get a driver abstract, issue a ticket, create menu class - took approximately 20 hours (approximated by Yuhang)

We all fully completed our individual parts, and we communicated through Slack to keep track of each other's progress. We also performed approximately five hours of work where we met as a group.