Mini Project 1 Design Documentation

a. System Overview

The system we designed contains two prompting lines that prompt the user to enter their username and password before entering the main interface menu. The password entered by the user will remain hidden. The username and password are compared to the uid and pwd in the 'Users' table using re.match. If the username and password exist in the database, the user with that username and password will have their username, password, user type, first name, last name, and city returned in the interface.

After the previous details are returned on the interface, the user is sent to the main menu. Depending on if the user type(utype) is an 'a' for agent or an 'o' for officer, the user will be shown a set of messages. There are 6 messages corresponding to the 6 questions for agents and there are 2 questions corresponding to the 2 questions for officers. The user may enter a number from 1 to 6 for each of the 6 questions for agents, respectively, and the user may enter the numbers 1 to 2 for each of the 2 questions for officers, respectively. The user has the option to logout at anytime while they are in the main menu or they are given the option to quit the program as well. The user is prompted to logout by entering the character 'l' and they are prompted to logout by pressing the character 'q'. Once they logout, they will be sent back to the beginning where they will be asked to enter their username and password once again. Furthermore, after each question is completed, the user is sent back to the main menu where the interface is once again displayed to them.

Small User Guide:

1. Run main.py.
2. Enter your correct username and password when prompted to.
3. If the user is an agent, enter any integer from 1 to 6 to be prompted to complete one of the six tasks allocated for agents.
4. If the user is an office, enter either 1 or 2 to be prompted to complete one of the 2 tasks allocated for officers.
5. The user may enter 'l' to logout or 'q' to quit the program entirely.
6. For each of the 8 questions, the user is expected to enter the appropriate input when prompted to do so in order to achieve the expected output.

b. Detailed Software Design

1. Login Section: In order to get user's password to be hidden, the getpass function is imported and password is assigned to getpass(). The main code for logins is nested inside of a while loop that runs while a variable, i, is remains True. If the username and password are both in the database, the loop will end and i will become False. If they are

not in the database, the while loop will keep running until a name and password in the database is entered.

2. Main Menu: A multitude of while loops are used to check when the user logouts or when the user quits. The while loop for logouts will become True when the user chooses to logout which is nested inside of a larger while loop for quits. As for user input, there is another while loop inside of these two while loops that makes sure that the user input is valid input. These loops will be terminated when the correct characters are enters the correct characters as detailed in the System Overview.
3. Question 1: Registering a birth required multiple inputs from the user regarding first and last names, and gender. If the father or mother's name does not exist in the database, user is prompted for information regarding either/both parents. Unique registration numbers were automatically assigned to the newborn by making sure it didn't already exist in the dataset. Date was obtained by the datetime module and all inserted together into the database
4. Question 2: The registration of marriage required the first and last names of two pre-existing individuals in the database. If these people did not exist, then user is prompted to enter information about the individuals, inserting their data into the database. Current date is gained from date module and unique registration number is automatically assigned by the system, and all data is stored in the marriage table.
5. Question 3: Vehicle renewal requires the user to input and registration number that exists in the database. User is continually prompted until a valid registration number is inputted, and the expiration date associated with the number is extended by a year, but if the expiration date has already passed the current date, obtained by the date module, then the new expiration date becomes the current date extended by one year.
6. Question 4: To process a sale, the user is asked to input a vin, using an SQL statement to check if the entered vin is in the vehicles. Then the user is asked to enter the current owner's first and last names. The program uses an sql statement to get the most recent owner of the car with the provided vin, then ensures that the owners first and last names match the first and last name that the user entered. The user is then asked to enter the first and last names of the new owner. The program runs an SQL statement to check if the given user is in the persons table. If not, the program will ask again until it receives a valid name, at which point it asks the user to input a plate number. It then uses an SQL update statement to the current car owner's registration expiry to the current date and then inserts the given values for the new owner into the registration table, creating a unique regno by incrementing the highest regno already in the file
7. Question 5: The program prompts the user to input a ticket number, checking if the ticket number is part of the table tickets, and if so then prompting the user to enter a payment amount. If the payment amount + the total payments for the entered ticket to date are less

than the total fine for that ticket the payment is processed, however if not, the user is asked to either enter a lower amount or can exit back to the main menu

8. Question 6: The program prompts the user to enter a first name and last name for the driver, checking to see if that person is a registered driver. If not, the user is asked to enter again, but if they are, the program then runs various SQL statements getting information on the number of tickets, demerits, total demerit points in the past 2 years and total demerit points for their lifetime. If the driver has tickets, the user can press enter to see the most recent 5 tickets and then press enter again to see the rest of the tickets, or the user can press 'q' to quit.
9. Question 1(Officers): Uses a user-defined function called 'Traffic_1' where the user is prompted to enter a registration number, regno. A variable called type is set to 'o'. The re module is used as re.match to check if the regno entered by the user is equal to a regno in the table. If it is, the regno and type will be used corresponding with '?'s in the c.execute() call where first name, last name, make of vehicle, model of vehicle, and color of vehicle are printed. These are printed using fetchall() and a for loop to loop through each of the 5 items in the tuple selected from users. Afterwards, the user is prompted to enter a violation and a fine amount. The error handling for fine amount is dealt with by using a while loop and a Try and Except clause to check if the value entered by the user is an integer. The user is then prompted for the violation date. For error handling, datetime is imported and date is imported from datetime, where the violation date will be automatically assigned to today's date if it is empty(''). If the user input is not a date in the correct format a while loop and Try and Except clause are used until the user enters the proper format. Next, a new ticket number(tno) is computed and the tno, regno, fine, violation, and violation date are inserted into a new table.
10. Question 2(Officers): The user is prompted to enter a vehicle make, model, year, and plate. After that, we did not manage to complete it, but each possible case for the 5 categories is handling using curs.execute() 5 times separately. The curs.execute would be in 5 SELECT *... statements where each of the resulting tuples are added to a bigger list. From there, a logical approach would be to use a dictionary and remove duplicates when 4 or more matches are found until the correct names or placed together in a set.

c. Testing strategy

In order to test the shortcomings of our code, we used incorrect type inputs and a large variety of strings to catch any errors that were outputted. Multiple test cases were used to error handle many inputs that a user may make. The print function was used to make sure all code gave the correct result and that no error would be passed into the database.

d. Group work strategy

For this mini-project, we randomized each question and assigned a set of at least 2 questions to each person in the group. Joshua Vuong implemented the registration of newborn children, identity of people, marriage between partners, and renewal of vehicle registrations for agents. Nana Andoh designed the login interface, and created functions used to process bills of sales, payment towards tickets, and obtaining a driver's abstract. Jason Branch-Allen focused more towards the tasks of traffic officers, implementing tasks to help officers issue tickets to offenders, and find car owners based on physical characteristics.

Each member of this group spent considerable time creating and designing an intuitive program that can help either agents or traffic officers with an optimized structure. Nana Andoh spent 30 hours designing the interface, leading the group, and helping others when needed. Jason Branch-Allen tackled several difficult questions and helped construct theory behind the code spending 22 hours total. Joshua Vuong, who spent 25 hours, implemented multiple functions regarding social registrations and helped design functions used throughout multiple parts of the interface and was directly involved with the creation of data in the database.

Every member worked on the project both inside and outside school hours, collaborating through a shared space in the Computing Science Centre at the University of Alberta. All members were also communicated virtually through the internet to ensure progress was on track and to assist one another.