(a) A general overview of your system with a small user guide

Log in with your username and password. Please note that for security reasons, the password is not displayed when typed. A menu will be displayed according to your role. As a registry agent, you can register a birth, register a marriage, renew a vehicle registration, process a bill of sale, process a payment and get a driver abstract. As a traffic officer, you can issue tickets and find a car owner. When given the operations on the menu, either enter the number associated with the operation or write the operation itself. You are able to logout at anytime by typing “Logout” in the command line or exit by entering a blank line.

(b) A detailed design of your software with a focus on the components required to deliver the major functions of your application

The software has been broken down into several functions:

- Login function
  Its main purpose is for the user to authenticate. From the credentials, the function will query the information about the user from the database. It will parse this information to the subsequent functions. For the authentication process, the function uses standard input functions except for the password which is hidden using the *getpass* function from the *getpass* module. The username and password are then checked for validity using the *re* module before they are used to query user information from the database.
- Menu function
  Main purpose: to display the operations unique to each role(*utype)* and to call the functions associated with each of these operations but also including the logout and exit options. The menu function takes as arguments *utype, uid* which it receives from the Login function. Using the *utype*, it will display the corresponding menu and its operations. It contains the main loop which is responsible for the running of the program which is halted only when the user exits.
- Question 1: registerBirth function
  All information provided by the user are obtained using standard input functions. The birth date of the newborn is converted from *str* to a *datetime object*. This is to ensure the validation of the input (that is, the date which was inputted fit to the format required). We then do input validation on the names by using the *re* module and verifying that the names contain only alphanumeric characters. To generate a unique id for each birth, we “auto-increment” the largest ID already found in the births table. Otherwise, we assign it the value “1” if there is no birth registered. We check the input once again to make sure no errors occur: no blanks inputted, parents don’t have the same first name, birth has not already been registered.
  We check if the parents are in the database by querying for their names. If either one of them or both are missing, the user is prompted to provide the info using the *input* function. They are then inserted in the persons tables. We search for the mother’s address and phone number using queries and use that to register the birth. After getting all the required information and making sure that everything is valid, the birth is inserted into the persons tables and the births table.

- Question 2
  This function can prompt the user to input the details of the two partners in a marriage. Everytime the user input a name, the program will compare the lowercase of the name with the lowercase of the name in the “persons” table. If the name does not show up in the persons table, the program will ask the user to input some more details of this person and insert these details into the persons table. In the end, insert the new marriage into marriages table, and the regplace

In the table is get from the current user's city.

- Question 3
  This function is to ask the user to input the registration number to renew a new registration date. The program will check is the regno is exist in the database. The current date and new expiry date is computed by incrementyear function. If the registration is expired or expired today. The program update the new expiry date(one year after today's date) by using the query of update. If the registration not expired. The program updates the new expiry date by using the query of update. For both cases, the new expiry date will be printed on the user interface.
- Question 4
  To process a sale, the program will ask the user to input the vin of the car, the new and old name of the car owner. And it will also ask the new plate to prepare for the new registration. The program will check if the old and new owner's name is in persons table or not. Only when both of them show up in the persons table, the sale can continue. And the function will also check if the old owner is the latest owner of the car by combining two tables vehicles and registrations together. If the old owner is not the latest owner, the sale will stop. The update of the registrations table will change the old owner's expiry date will be set to today. In the new registration, the regdate will be set at today and the expiry date will be set after one whole year.
- Question 5
  User input is provided using standard input functions. The current date is obtained using the *time* module. Input validity is checked using the method *isdigit* since both *tno* and *amount* should be integers. The ticket number *tno* is then queried in the database to make sure it exists. We use two queries to check if the payment is valid, one to make sure that the payment in addition to previous payments do not exceed fine amount and the other one is used if it is the first payment made. We also use a query to ensure that a payment can only be done once a day.
- Question 6
  The program asks the user to input the first and last name to find the record in the registrations and persons table. If the name is not in the two table, the search cannot continue. It will count the total 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 by counting in the combination of tickets, demeritNotices and registrations table. The user can choose to see the details of the ticket in the order from the latest to the oldest or not. The ticket number, the violation date, the violation description, the fine, the registration number and the make and model of the car for which the ticket is issued. At most 5 tickets detail can be shown at one time, user can choose to see more or not.
- Question 7
  The program asks the user to input the registration number, model, make, year and the color of the registered vehicle will be printed on the interface. If the user does not input the violation date, the default setting is to set the violation date as today's date. The program will use the query to find the corresponding first name and last name by regno. After that, the program generates unique tno by finding the existing tno + 1, if there is no any tno in the database. Set the new tno as 1. Then the programs use a query which includes type violation, violation date and the number of fine so the new and unique ticket number will be insert to the database.

- Question 8:
  This function is to find the information about the car owner, The user will be asked to input one or more out of the five categories: make, model, colour, year, plate. If there is no any input.The

program will return false. Convert all of the input into lower() to ensure consistency. The programs will distinguish every input to consider if there are a match or not by find all of the vin with query. Only consider the parts if the parts can be connected by vin. If no vin from the input. Return to the manu. If there are more than four persons or more exists in the results. The program will ask the user to choose one of the five categories: make, model, colour, year, plate and print the chose information. If the input is none of the five, return to the menu(out of the function)
On the other hand, if there are less than four people who exist in the results, the program not only prints the information for the five categories mentioned previously, but also the information about the registration date, and expiry date, and person's last name and first name.

(c) The testing strategy:

For testing, we choose to test our functions separately. To be specific, every single function is tested individually while other functions are being disabled. Every group members input some correct data to test whether the query is working and other parts of the function can be activated effectively to generate the correct result.
On the other hand, we input incorrect data like gibberish to test whether the function is able to handle unexpected errors or not. During the process of testing, we found some major error which can cause the program to fail to malfunction. Such as the wrong query for selecting, wrong error handling, and syntax errors. Those errors come from the lack of paying attention or the lack of considering corner cases. Some small issues like lack of string matching are relatively easy to fix.

(d) The group work break-down strategy

Xiutong Jing is responsible for completing the 3rd question and the last two questions for the police officer.

Work Task	Time spent	Progress made
Task 3	3 hours	100%
Task 7	2 hours	100%
Task 8	3.5 hours	100%

Gary Ng Kwong Sang was responsible for creating the login screen and doing task 1 and task 5 of the registry agent.

Work Task	Time spent	Progress made
Login Screen	2.5 hours	100%
Task 1	4 hours	100%
Task 5	2 hours	100%

Xichen Pan was responsible for doing tasks 2, 4, 6 and the connection of all parts for the registry agent.

Work task	Time spent	Progress made
Task 2	1 hour	100%
Task 4	2 hours	100%
Task 6	2 hours	100%
Connection	3 hours	100%

Method of coordination

We would meet once or twice a week to discuss the mini-project: problems encountered, progress made, and other possible improvements for the program. We communicate with each other using WeChat, which is a messaging application. On the group chat, we would express any current problems with our individual work tasks.

Documentation about decisions made

When dealing with renewing the dates, we took into account that depending on the expiry dates, the following year might be a potential leap year. Hence, to prevent any possible problems, we created a helper function that takes the date as an argument and checks if one year from that date will be a leap year and increment the date accordingly.