# CMPUT291 Project1 Document

Group Members: Zhijie Shen, Liyao Jiang, Hongru Qi

**System Overview:**
This system allows users to offer, search, book, request, and manage rides. This system is built using python and runs in the command window. The database is assumed to be provided by the TA and our program is using python to host sqlite3 to connect and operates on the database.

**User Guide:**
To use this system, go to command line and run the following command

**'python3 291project.py (database file)'**

Where the (database file) would be replaced with the name of desired database.
For example: **'python3 291project.py project1.db'**

Starting out, the system will ask whether or not the user has an account, if the user has an account they will be prompt to login, if user has no account they will be prompted to register. After logging in, the user will have 6 options: offer a ride, search for rides, book members or cancel bookings, post ride requests, search and delete ride requests, and logout.

After logging out, the user will be put back to the login/register page, where the user can use different accounts or quit the system.

For more detailed step by step guide, see the Software Design section for each function.

**Software Design:**

Login&Register:

In order to use our system, user need to log in or register a new account. Firstly, user has option to choose whether to log in, sign up or quit. If log in, user is asked to enter username and password which will be verified by executing search. If the user successfully logged in, all the unread messages of the user will be displayed and marked as read. Then user will see the main selection page will allow to select wanted feature. If sign up, user will be asked to enter user name, password, name and phone number. User name must be a valid email which the format will be verified. Password is formed with only letter, number and underscore with no length restriction. The format will also be checked. Name is letter only and phone is integer only. After that, user is successfully signed up and is able to see the same selection page and select features.

1. Offer a ride:

   The idea is to get user's input and then once all the information is gathered, a prompt is displayed asking for user's confirmation before all the changes are committed and inserted into the database.

First the system asks for the user's input on: date, number of seats offered, the price per seat, and a luggage description. Regular expressions are used to check that the inputs are valid, and some of the input is converted into correct data type to insert into database.
Then, the system will interactively ask for user's location inputs: source location and destination location. The system will take the input, run a search in the database using queries. Once results are found, the results will display 5 at a time and allow user to select one or view more options. Enroute locations are entered in the same way, except it will prompt multiple times.
An optional car number can be added, if a car number is entered, the system will check if the car belongs to the member currently logged in.
After all the information has be entered and user confirms posting the ride, a unique rno will be assigned and all the information will be grouped in the correct order into a tuple and inserted into rides and enroutes, the changes are then committed.

2. Search for rides:

   This feature allows user to search for rides by entering up to three location keywords. Then display all the results and let user to select any one of the rides and send message to the driver who offered that ride.
   First of all, ask user to enter the location keywords separated by space or (quit) to go to the main selection page. The reason why I use "(quit)" instead of "q" is because "q" can be a possible keyword to be entered. Then, use cursor to search all the lcodes correspond to each user input keywords. Then search corresponded rides with these lcodes by first union the rides of all the lcodes of each keyword and then find the intersection of each union (by keyword so maximum three). Then display the results and let user to make selection. The maximum number of displayed outputs is 5 and user has option to see more. If user select one of the rides to send message, the system will ask user to input the message and the message will be sent to the driver's inbox which user selected.
   The display and make selection part will iterate by first asking if user wants to send another message or q to quit the iteration. If the user chooses to quit, the system will ask the user to enter keyword again or (quit) to go back to the main selection page.

3. Book members or cancel bookings:

   This feature contains four subfeatures, that is (1) list all the bookings on the rides provided by the current user; (2) cancel a booking on the rides provided by the current user by giving the bno; (3) list all the rides provided by the current user; (4) book a member to a ride provided by the current user, the user will be asked to input the required information about this booking.
   (1), This is done by executing a query to find all the rides where the rides.driver attribute is equal to the current user's email. Each qualified rides entry is printed to a line. And the program will go back to the previous sub menu.

(2), In the following steps of asking input, the program will ask the user to enter y to give a input again otherwise quit the feature when the input is not valid. This function will ask the user to input a bno for the ride he/she wants to cancel. The program will check the booking exists in the database and will compare the current user's email and the email of that booking (the email is found by a SQL query) to validate that the user is the one who provides this ride. Then the error messages will be printed. If a valid bno is provided the program will execute the SQL delete clause to delete that booking. The program will also add a message entry to the inbox table, the receiver will be the member who's booking got canceled, the sender is the current user, datetime is now, the ride number is the associated ride, the content of the message contains the bno of the canceled booking. The 'seen' attribute of this message is 'n'.
(3), This function list all the rides provided by the current user and the available seats of each ride. The program will create a ride_info view containing the rno, the driver, and the available seats of that ride. And it will execute a query to select all the rides provided by the user. The program will print the rides 5 at a time if there is more than 5 of them, the user can enter y to see more. If there is no ride found, it will print the information out.
(4), This function allows the current user to book a member on a ride provided by himself/herself. The program will ask for input to get the rno, it will first check it the rno exists in the database, and if the current user is the driver of that ride. Invalid inputs will be prompt to enter y to input again, otherwise quit the function. If the rno is valid, the user will be asked to input a valid existing member email. The number of seat input will be checked against the available seat calculated in the ride_info view, and if the user is overbooking, the system will ask the user for permission to do so. Then the system will ask for the cost per seat which should be non-negative integer. The user will then input the pickup and dropoff lcode, and the lcode will be checked against the database. The system will then generate a unique bno by incrementing the existing bno, if there is no existing bno, the bno will start at 1. The booking will be inserted by executing the SQL insert clause. The message will also be added to the inbox table, and the receiver is the booked member.

4. Post ride requests:

   This feature allows the user to post a ride requests by providing some information. That is, the user is asked to provide a valid date follows the yyyy-mm-dd format, a existing pickup location lcode, an existing drop location lcode, and the amount (non-negative) willing to pay per seat. All of these inputs is checked by the program to make sure they are valid, if invalid input is given, the user will be prompt to enter y to try input again or otherwise quit this function. The date format check is done by using the datetime module in python. The lcode input checks are done by checking if the given lcode is in the locations table. Each ride request has a unique rid, which is generated by incrementing 1 to the existing rid, if there is no existing rid, it will take the value 1. The email attribute is set to the logged in user's email. After getting all the necessary information about the request, the program will use the SQL cursor to execute the insert clause. When the ride request is added successfully, the program will print a confirmation line and the rid to the command line.

5. Search and delete ride requests:
   This feature allows user to search and delete their rides or search rides by location and then display the searched result and allow user to send message to the one who posted the request.
   Similar to other features, the maximum number of displayed results each time is 5 and user has the option to see more.
   First of all, ask user to select if they want to search or delete requests. If search, user then select to search their requests or search by location. If their requests, display up to 5 results each time. If search by location, user then select to search by keyword or by city and ask the user to enter the keyword or the city name. The display of results follows the same format and user is allowed to select requests to send message to the one who posted that request. User has option to choose to send another message or perform another search. All searches are done using cursor to execute SQL commands.

**Testing strategy:**
Because each functionality is implemented by different members of the group, testing is done separately. Each functionality is tested through making test functions inside main() for each py file to make sure the functions are behaving as intended. The strategy of testing on unexpected invalid inputs is used to make the program robust.

**Group work break-down:**
The project have five main features and a login screen, and each feature is implemented separately and combined together at the end in 291project1.py. Here is the project break-down between partners.

Each member implemented and tested the followings:
Zhijie Shen: offer_a_ride.py, queries.py
Liyao Jiang: feature3.py, feature4.py, 291project1.py
Hongru Qi: login.py, rideRequest.py, searchRides.py

Time allocated and spent for each feature and progress made: we are expecting 4 hours each for the first three features. We assigned one feature to each member at the beginning and started to implement each feature in a different '.py' file, each member is also responsible for testing the functions.  At the end of 4 hours, we each completed the implementation and testing of the assigned feature. Then we assigned the rest 3 parts to each member, since the rest feature is smaller in size, we finished them in 3 hours. When challenges are encountered, we pair up and try to solve the challenge. The rest of the two functions and the documentation is done together by us.