USER GUIDE

This databases handles bookings and rides for members and drivers. All data and functionalities will be printed to the terminal window of whoever is using it.

Upon startup, the user is asked to enter a .db file to query the data with.

Then it will ask the user to login, or create an account with actions (EX “existing”) or (NEW “new account”)

Upon making a new account, the user will be taken back to login.

When a user logs in, they will be taken to their homepage, where their inbox will be shown to them if there are any unread messages in their inbox. If there are no unread messages, then the user will be asked to perform some functions. The functions are as follows.

1. RG = register a car: a user can register their car by providing all asked details about their car. Their car will then be registered in the database.
2. OR = offer a ride: a user can offer a ride to two locations. The user is asked to select a car they want to use if the user has more than one care. Else, system assumes the user will use their only car. If user has no cars, then the user must register a car, or exit OR. The user must enter a date, price of seats, luggage description, and optionally an enroute location.
3. SR= search for a ride: a user can enter up to three keywords to search for a ride. All results will be returned based on how specific the keywords are. If there are more than five results, then the user will only see five at a time, with the option to see more if they wish.
4. CB = cancel a booking: a user can cancel a booking they are booked to drive for. The user must enter the ride number of the booking, and the booking must be associated with that user’s ride. The booked member will be notified by email.
5. BM = book a member: a user can book a member of the database on a ride, and enter the cost of a ride, how many seats they wish to book, a date, and a ride number. If the user is overbooking their ride, they can have the option to abort or continue. The booked member will be notified by email.
6. PR = post a ride request: a user can post a ride request to a location.
7. DR = delete a ride request: a user can delete a ride request by delete a RID associated with their email. The user cannot delete ride requests not associated with their email.
8. SQ = search a ride request: the user can search for posted ride requests in the database. The user will be given the option to send an email to a member asking them to check out a ride if they wish.
9. A user can LO (logout) and be returned to the main screen.
10. A user can Q (quit) and the program will close.

DESIGN:

Language = python

MAIN()

The program is called from a main function titled database() which calls the login() function If login succeeds, then main calls homepage() which is where all functionality is performed.

LOGIN()

Login takes input of whether a user is an existing member or a new user. The input given must be “EX” or “NEW” and the user will be told if input is not correct. User is taken to next functions accordingly.

EXMEM()

Logs in an existing member. If the user wants to go back to the login screen they can enter EX and that will return them there. Otherwise the user must enter their email and a password. Upon enter an email and password if no email exists in the system will return a message saying input email is not in system.

Elsewise if the password does not match the given email, then the user will have to try again. For injection security, input can only contain A-Z 1-9 and @ for the case of email. If login is successful, email is passed to function homescreen as the email that will be updated as user performs functionalities.

NEWMEM()

User is able to enter a name which can be anything, and an email which upon entry can be anything as well. User must then enter a phone number if the phone number is not valid : either it is too long or cannot be converted to type int, then user must try and enter a phone number again. If phone entry works, then move on to password. Password must be entered twice, each assigned to a different variable. If the passwords do not match, then the user will have to enter the passwords again. If passwords match, then the program checks to make sure both password and email contain only legal characters, if they do not they must begin the process again. If successful, then all information is inserted into the database and is passed to homescreen(), and that will be the email with which the user performs functionalities.

HOMESCREEN(email):

The home screen is where almost all functionalities of the programme are called from. Upon entering the home screen, if user’s inbox has unread messages, they will be displayed to the user one at a time, until the keyboard receives any key. Once a message is read, seen is set to ‘y’ and is updated in the database. This happens for every unread message. Once messages have been read, the user is greeted and is given options to perform functionalities. Which will be detailed in sections following this. From this point functions will be named and described from which function they are called.

REGISTERCAR(email)

CALLED BY: homescreen(email) and offerRide(email) ---- to be described later.

A user is given the option to register a car under their name. The user is asked the make of their car and the model of their car. The user is then asked what year their car is, and when all have been input, the system assigns a unique CNO to the user by taking the largest car number from the database, and adding 1 to it. All input values, the automatically created CNO and the user’s email is inserted into the database.

The CNO is then then displayed.

OFFERRIDE(email)

CALLED BY homescreen(email)

A user can offer a ride to a location only if they have cars registered to them. If no cars are registered to the user, then the user is given to the option to add a car. If they choose to then the user enters a car with registercar(email), and then is taken to the home screen. If the user does have a car one of two things can happen. If a user has only one car, then that car is the assumed car to be used in the ride. If the user has more than one car, then the user must choose which car they wish to use. The user can then search for a location code, or any location input if they do not know the location code they wish to enter. If user wishes to search, then they are asked for a keyword, which is passed as function searchLoc(loc) which searches for row values for all columns in locations LIKE that keyword and returns the data found. It then is passed to display results which incrementally displays the results by five, or all results if there are less than five. If no results, then no results will be displayed. The user will be asked if they wish to see more results, and if not then they will be prompted to enter a location code twice, once for src once for dst.

Both must be valid, and user will be asked to either try to enter again if invalid, or start over to search for more location codes.

If location codes pass, then the user is asked to enter the cost of a seat, which must be an int, a luggage description, and a date, which must be in proper format, can’t be longer than format length, and must be before the current date (today’s date). User must then enter the number of seats on a ride, which must be an int. If all pass, the ride is entered into the database, and then the RNO and DATE of the ride will be displayed. The user will then be asked to enter an enroute location if they wish, and if they do wish, then they must enter a valid lcode. If lcode is valid, then current RNO and LCODE will be inserted into the database. If user does not enter an enroute, then they are taken back to homescreen.

SEARCHRIDE()

Search ride asks the user to enter up to three keywords to search through the database. The assumption we operated under is that more keywords provides more specific (less) data. If the user enters less than one key word then an error will be displayed to them. If a user enters more than 3, an error will be displayed to them. For keywords entered, a sql query is posed to the database, and intersected with the adjacent query. Those results are then printed to the screen by fives. If there are less than five results left, then only those results will be shown.

CANCELBOOKING()

User can cancel a booking they are booked to provide a ride for. The databases searches for the user’s rides, and then sees if any bookings are associated with their rides. If bookings don’t exist, then user will be told they have no bookings to cancel. If bookings do exist, then user will be told to enter a booking number to delete. If booking number is not associated with one of the user’s rides, then they will be given an error. If no error goes through, then an email will be sent to the booked members inbox telling them their booking has been canceled.

BOOKMEMBER(EMAIL)

A members rides are taken from the database. If a member has no rides, then they will not be able to offer a ride. If a member does have a ride, the user will be asked which user they would like to book, and a list of users will be given to them. If member is not in the database, the user cannot book the member. The user will then be asked to choose a ride, and their RNOs will be listed to them. If the input RNO is not associated with the user, the user will be given an error. The program then selects the the total number of seats already booked on a ride, and the total seats in a the vehicle that the ride is booked with. The user is then asked how many seats they want to book. If the user is going to overbook their ride, then they will be given the choice to overbook, or abort. If they abort, then they are taken back to their home screen. If not, then the user must enter a valid pickup and dropoff. If they have trouble entering a valid pickup or dropoff, then the user can abort and be sent to the homescreen. If the lcodes pass, then the booking is inserted into the database, and a message is sent to the user who is being booked.

RIDEREQUEST(EMAIL)

A user can enter a ride request. The user has to enter a valid date, which cannot be in the past. The user will then have to enter a valid pickup and dropoff location, and if they wish to abort to search through locations again, they will be given the option to. The user must then enter a valid price. If the price is not float, then the user will be given an error. If price passes, then the values are inserted into the database.

DELETEREQUEST(EMAIL):

A user can delete a ride request. The system searches for all ride requests associated with the user’s email and displays them to the user. If the user has no ride requests, then they will be taken back to the homescreen. If the user does have ride requests, then they will be displayed to the user five at a time. If a user wishes to delete one of their requests then they input YES, and then they must enter the RID of the request. If the RID is not associated with that user, then a program error will be displayed. Otherwise, the data will be deleted from the database, and the user will be notified.

SEARCHREQUEST(email)

A user can search for a ride request, by providing a location keyword. If the keyword returns results, then those results will be printed five at a time. If the user wants to see more results, then they can until no more results are available. If no results are returned then they can try searching again, or returning to main menu. The user will then be asked if they wish to send a message to another user about a ride request.

SENDMESSAGE(email)

When entering this function, seen is set to ‘n’ (not seen). The user will then be asked to enter a message that they will send. All rides are taken from the database, and displayed to the user. The user must choose one of the rides to send to a member, which must be a RNO from the database. The user then is asked to

send to another member. If member does not exist in database, then user cannot send email. The system then generates a time to enter into the database, and the email is inserted into database and sent to chosen member’s inbox.

TESTING STRATEGY:

Our testing strategy consisted of writing the functions one a time, and trying them out by passing out all other written functions in the program. When all functions were written, we then put them all back into the program, and ran them together. We tried entering as many possible errors and strange input into our program as possible, to check for all errors in our code. We also tested on all of our computers, and tested by ssh in a terminal to run the code on a lab machine.

GROUP WORK STRATEGY

We initially tried working with Tkinter, but ran into many problems with it. So we switched to printing into the terminal. Tkinter implementation was done by Joshua, and translating the Tkinter into terminal printing was also done by Joshua. The questions were divided to group members. did 1 & 2 as well as handling injections and the hidden password. did questions 3 & 4 and also did questions 4 & 5. We emailed code back and forth frequently and had a group chat to coordinate meeting and problems we had with our project.