1 Introduction

This project is a simple ride booking platform that provides members the ability to perform multiple tasks. The tasks include booking rides, requesting rides, offering rides, sending emails, and deleting requests among other functionalities.

1.1 Purpose of Mini Project 1

The purpose of Mini Project 1 is to allow students in the CMPUT 291 class to expand their knowledge of programming and sqlite3 by implementing embedded SQL into project design.

1.2 Mini Project 1 Overview

- Section 2 – General Overview.
- Section 3 – Design.
- Section 4 – Testing Strategy. Chapter
- Section 5 – Group Work Breakdown.

1.3 Identification

This project was done by Max Melendez De La Cruz and Raymond Masikonte.

1.4 References

Our main references were Mr. Davood Rafiei, his class slides, and the Lab Slides.

2 General Overview

2.1 Background Information

Mini Project 1 is built on fundamentals taught in the CMPUT 291 class and is primarily an implementation of Assignment 1 and Assignments 2.

2.1 Main

Mini Project 1 is a project that focuses on the fundamentals of database design and management. The project is implemented using Python 3 and sqlite3 to provide database management capabilities without having to have an external server.

The project implements a database named "miniproject1.db" which was created by Tanner Chell. The database contains 8 tables and different data associated with the tables. The tables are named: members, rides, bookings, locations, inbox, requests, cars and enroute. The tables all contain specific attributes, some are unique (primary key) and and some are used to help us, the programmers, reference other tables (foreign key).

The flow of data is applied using a series of queries. The queries allow the user to use the system to do multiple functionalities. The queries are used as a form of communication between the database, "miniproject1.db" and the programming language, i.e. Python 3.

Our design also implemented a graphical user interface (GUI). The GUI of choice was tkinter which, just like sqlite3, is an already implemented library in Python3 and hence an import function is all that is required. The graphical display allows the user to access the different capabilities of the system easily through the implementation of buttons and menus.

3 Design

The project is divided into different sections. The first section is the login, sign up, and logout. Members should have the capability to login, using their email and password and in the case new people want to be members, they should be able to sign up. For this we used classes. Class main menu was the code that implemented the main page that anyone would see initially. In this class, a function to log in was implemented by creating a button and executing the query that would search the database for a match in username and password and if there exists a match, the user would be allowed to log in and see their unread messages. The sign up section was implemented using a class called create account. This class included a function called register which in turn added a new member to the database by using an insert query. The logout was then applied by a function that would break out of the loop and exit back to the main menu as soon as the button was clicked.

The second section involved searching for rides. The program implemented a class called search rides. The search rides section involved providing an option menu that allows a user to search for rides using keywords. Any keyword associated to the ride is able to search through the database using a query that is implemented in the program. The query used the 'Like' connector in order to allow users to search the database by inputing anything that is similar to that particular search. The search displays a result of the top 5 searches for the user to see. The members can then use this section to book rides which was the third section.

The members are able to book or cancel bookings by the use of the class book rides and cancel rides. Members would have another button where they would click the book ride button and in this case be given options as to what details they need to input in order to book a ride. This is then implemented in the database using an insert query which adds a new ride into the database, specifically to the bookings table.

In the case that a member wanted to cancel a booking, the option was provided by a function which would use the delete query in the bookings table. However, the members still have to provide a few details in order for the ride to be searched which is implemented using a select query.

The fourth section was the ride offers. This was in turn implemented using an offer rides class. The class gives a member the ability to offer rides to other members. Hence, they should be able to give specific details about the ride which are added to the rides table by running an insert query in order to add specific details about the new rides offered.

The fifth section allows users to request for rides. The class request ride was used to run this section. To request a ride, a request ride button was added. For members, as soon as this button is clicked, the menu to request would open. This requires the members to input specific data regarding the rides requested and hence would be added to the request table of the database "miniproject1.db". This as well uses an insert query to add the details to the table and returns a success message.

Finally, members are able to search for and delete ride requests. If a member provides a ride request, they have the power over that request. This is to say they can search for that ride request which we would run a select query in order to search through the ride request table and return the requests posted by that member. The member can delete these requests by hitting the delete button and this in turn runs the delete query on the requests table and deletes the specific request.

Some basic error handling is implemented. For date, members must input a valid date, if the date is invalid, the program would return an error message. if the locations input by the member were invalid, the program would return an error message. Basically, if the member used an invalid login, offered a ride to an invalid location code, searched for an invalid location an error message was displayed on the screen.

4 Testing Strategy

The testing strategy used was a rather straight forward approach. Knowing all details of our given database, we used valid and invalid data to run the code and check whether the program run properly. For example, to log in we used an unidentified user, and the log in failed. But when we used, the username 'Bob@123.ca' with a password of 'bpass' the log in was successful and returned a message of, 'Welcome Agent'. The same system was applied to the other functionalities of the program. To book a ride, we provided valid information and that added the ride to bookings. Canceling the ride worked as it deleted the details from bookings. Searching for a ride as well was done. If all the info the program requested was valid the ride was found but if the ride was invalid, the error, 'ride does not exist' was returned.

In general, the testing strategy was basically from the programmers perspective and knowing all details associated with the database.

5 Group Work Breakdown

The work was divided as follows. Most of the work was done by Max Melendez De La Cruz. He implemented the GUI (Graphical User Interface) and most of the python code that was run in the program. Raymond Masikonte, came up with the queries associated with specific areas of the code and wrote up the design document and the README file associated with this assignment. It was more of a 65% to 35% ratio in work division. Knowing that it should have been an even 50-50 split, the truth is it was not mainly because of time constraints and capabilities of members of the group. Max is much stronger than Raymond and hence did the bulk of the assignment with assistance from Raymond.