CarShare Project Proposal

Project Summary:

CarShare is a car renting/leasing web application that allows users to rent/lease a vehicle. The main purpose is to make it hassle-free for lessors who want to rent their car for a short period of time when the car is not in use. Typically, the duration of the period would be in weeks to months. Our proposed application also makes it easier for the lessee to lend cars according to their car availability and closest CarShare station.
CarShare takes a lot of constraints into account such as user location, valid insurance, valid license and valid age of lessee (21+). Furthermore, the application also mentions the car make, model, type and capacity, availability, and location.

Description:
The main problem we are trying to solve is to have a hassle-free and easy process for common people with cars to make some extra money through their asset while it is not in use.

We are planning on adding functionalities using HTML/CSS for the front-end and Python Flask for the back-end. We will have various tables in our database such as buyer details, car details, payment details, user login, car availability, car ownership and other details.

The user will have to register an account/login to an existing account before being able to access the rest of the website. Once logged in, the user can choose to either rent a car, in which they can choose from a list of available cars catered to their specific needs, or lease a car, in which the user will have to input the details of the car they are leasing. The basic functionality of the application includes: creating renter/rentee login account (with associated user/payment/car details), checking the eligibility of the lessee (age and driving license validity) and the lessor (car insurance expiry), store all these information in the database tables (in the backend), providing user interface through frontend web tools mentioned above, performing CRUD (Create-Read-Update-Delete) operations. Complex functions of the applications include: retrieving user location details through front end and finding out nearest available car stations/lot for the renter (distance measured through road routes, and additional distance filters applied to limit available cars to proximal rentee's location), storing transactions (if a renter rents a car, the carId, renterId, payment details, pickup location, pickup and drop off time and date are all stored).

Usefulness:

CarShare does have similar websites such as Hertz car rentals, Turo , etc. However, the main point that makes our application stand out is that the other car rental companies own the cars they are renting while CarShare does not own any car. It is simply a medium between lessors and lessees. Just like how Uber does not own any car of their own, while other taxi companies

such YellowCab have their own taxis. Anyone who has a car (with valid license) can rent out for their convenient time and duration and any person (age 21+) who has a valid driver license can rent cars. Location of the renter and all available cars (based on the renter's demands) within some fixed radius is shown for that particular renter to rent out.

In addition, our application will make the renting and leasing process as streamlined and hassle free as possible for lessees and renters. It has several real-world applications as an alternative to the above mentioned solutions that already exist. Our application is meant to be a faster way of renting a car.

Realness

The data is collected from Kaggle data source - Mobility Uber Peru Dataset (https://www.kaggle.com/datasets/marcusrb/uber-peru-dataset) which is based on the similar idea presented. The location of the rentee (or person who wants to take the rent) is extracted from the Google Maps API service or manually entered by the rentee (depending upon project timelines and other constraints). Similarly, the renter (or the person who wants to give out the car for rent) will specify in the place where they have parked their car (it could be our CarShare station or the owner's parking lot, all these locations are fixed) and the availability (date, and duration he/she wishes to rent). All other information such as Renter and Rentee related details, Payment details and transactions will be stored in the database with every successful user registration into the application.

Functionality

Following gives an overview of the functionalities and how the final application would look like. This goes over what users can do and highlights some of the simple and complex functionalities of the application.

1. The first page will be the user login/register page the user (could be renter or rentee) where they can enter their details and car details (if renter) and validation happens in the backend which basically checks for correctness/uniqueness of license (both car/driver). i.e. Login page - Checks with user login info table to find match, and the Registration page validations for username/password length, etc.
2. After successful login, the user can see their profile information in their profile page
3. If the user logged in is a rentee (person who wants to rent a car), then details such as current location, pickup/drop off date, pickup/dropoff time and other additional details such as car type needed, capacity (seats) etc are asked.
4. If the user logged in is a renter (person who wants to rent out their car), then details such as car to rent, duration of rental (both date and timings), charge_per_hour, and location of the car (fixed) are asked to enter and submitted.
5. Now, the application calculates the distance between the rentee and all available cars in the Champaign city and filters only those that match the rentee's conditions (as given in

step 4) along with some specific radii of distance from the rentee location. Only those cars (along with their details) will be displayed to the rentee.
6. For each available car, the rentee has an option to book them and pay (No cancellation or change of trip duration is possible after booking and payment)

CREATE Operation:
The user will have to create an account in order to get access to the full functionality of the website, add their details, details of the car (if he/she is a lessor) and availability of cars.
SEARCH Operation:
The person renting the car can search for a car closest to their relative location and may choose to reserve the car.
UPDATE Operation:
If the user chooses to reserve a car, the listing/availability of the car they choose would be updated as “unavailable” for the selected times the user chooses. Additionally, the person can update other information such as email, phone number, car license (if renewed) etc.
DELETE Operation:
If the user chooses to no longer rent his/her car, they can delete the particular details. If the user wants to delete their profile from the database, he/she can do so.

Searching for records, updating records and deleting rows will be implemented according to this schema.

Low fidelity UI mockup

CarShare Project

PAGE I: (for user login)

Rentor/Rentee Login/Register

Login
Email: [ ]
Password: [ ]

} checked with our database details

PAGE IA: (for registration)

Username [ ]
Email [ ]
Password [ ]
Confirm Password [ ]
(Signup)

} Fields are checked for validity, correctness/ uniqueness

PAGE II (User Profile)

Account Info
Name:
Phone:
Email:

PAGE III

After Rentee Login

current location [ ]
pickup date [ ]
pick up time [ ]
dropoff date [ ]
dropoff time [ ]
:
Additional filters
[SUBMIT]

After Rentee Login

Car Id to Rent: [ ]
FROM: [ ] TILL: [ ]
cost per hour: [ ]
Car Location: [ ]
[SUBMIT]

AFTER MATCH

List of all cars matching the rentee's criteria
a) carId available time date
b) " " " "
c) " " " "

Project Work Distribution

The project workload will be distributed evenly amongst group members.

Michal Trzupek - Fullstack, Design

Sreekar Bathula - Django Database Interaction

Praveen Kumar Kurugaiah - Create database, user login/registration pages, Sql flask integration, backend logic for distance calculation

Vivek Bhatt - Python Flask API Interaction