Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

The whole project idea, usefulness, realness, and detailed functionalities remained very similar to the proposed plan.

We previously planned on implementing both the customer and the admin sides with full CRUD for users, cars, reservations, and reviews, but we implemented the Customer side CRUD. We are considering that the admin has database ownership, so we don't need to particularly implement his functionalities.

Rather than checking if the person is below 25 and adding an insurance val, now we have a table of insurance val and we take out and use values from that particular table. We got rid of allowing the user to change their account information, they should contact the admin for help regarding these tasks.

We have completely followed the other requirements, including the dynamic location based deposits based on the car theft dataset and also properly implemented the search features. For the search feature rather than ordering them price, we are ordering them by average ratings (DESC) followed by price ASC). We wanted to provide customers with options that our other users have found to be good.

Discuss what you think your application achieved or failed to achieve regarding its usefulness.

Achieve:

1. Ability to book, modify, delete reservations for cars and reviews
2. Ability to see the dynamic/discounted prices along with prices per month prices
3. Implement a dynamic search bar to the customers that provides them with best options for them (This depends on lot of factors and changes regularly)
4. Use the car theft dataset, to provide better deposits to the rental service and also insurance value based on the age of the customer.

Failed to achieve:

1. No map based search integration limiting visual exploration of cars (Creative component!!!!)
2. No real images of cars because kaggle data image urls did not exist.

Discuss if you change the schema or source of the data for your application

1. No changes in the schema
2. Source of the data is still Kaggle 2 datasets

Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

Originally, our original design was to have five tables: Customer_Info, Booking_Reservations, Car_Rental_Info, Rating_and_Reviews, and Car_Theft. However, after applying normalization, we implemented an additional table, Insurance_Detail, which we thought was more suitable because Customer_Id is a superkey, but we also have an additional dependency for Age and Insurance_val, and Insurance_val isn't part of the candidate key. We thought this would be more suitable to minimize data duplication through the idea of splitting a larger table into smaller tables when possible.

Discuss what functionalities you added or removed. Why?

Based on our original proposal, we implemented most of the features that we sought to implement. These were the user registration, booking and reservation management, and the ratings and reviews section. We felt that these would be valuable to the website because they were core functionalities that a user would need when renting a car. We also added functionality for determining whether a user could get a discount or not, which we believed was needed to encourage returning users.

We also left off some of the proposed functionalities. The biggest one was the creative component, which was supposed to be a map-based search of cars, which we thought would be unnecessary and might just end up cluttering the website a little. We also got rid of all administrator-related functionalities because the administrator would have direct access to working with the database, and we felt that it was unnecessary to add this type of functionality for our users.

Explain how you think your advanced database programs complement your application.

So, here, we have used 4 different advanced queries in our application. We use 2 of these queries in a stored procedure which we use to implement the keyword search. It gives us the cars that are not booked in the next 30 days and also the ones that have had a really high rating based on previous customer experience. This allows us to have a really dynamic search bar which provides real time improved and enhanced search results to the customer.

Along with this, our other 2 advanced queries were used to implement the rent now section, one of the query combined the theft data, insurance data and price info to get the total price to the customer whereas the other query checked if based on the customer previous booking history, we would want to provide them a discounted daily price for the car!!! (We look after the regular customer). These queries also allowed us to make real changes in our databases to provide proper pricing information to the customers.

For the constraints part of the DB, we have added and properly implemented the foreign keys so we didn't need to add any new particular constraints.

We implemented our database on GCP.

Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.

Aryan: I faced a few issues around sending an api endpoint request from the frontend to the backend while I was working on the initial blueprint controllers. It took me some time to understand how to work with react and flask together. It is important to make sure that all the routing and environments are being made and modified properly. Over time, as I worked on the next stages, I was able to understand and improve on this front.

Dhruv: Writing code and building is the easy part. I struggled with connecting to the Illinois Project Git, Installing the GCP CLI and integrating the Database connectors with each other. It is important to learn the basics and create appropriate .env files to run and connect them successfully on multiple devices.

Jun: Initially, I found it challenging to try to model the database that we would be using; there were so many components that we had to implement to not only be exhaustive, but also effective and efficient. It took me some time to fully model what the database would look like, how it could be improved, and how it would interact with the rest of our website. Through this, I learned through trying to fully understand the concepts from our online lectures, and I also learned how important planning is beforehand; if the app is not properly structured, the development can get really messy fast.

Joey: I faced an issue of failing to retrieve all reviews of a particular car from the database while putting these reviews on the Rent Now page. The Rent Now page only shows the latest review rather than all previous reviews. I initially thought it's a problem of frontend not properly dealing with the retrieved data, but soon eliminated the possibility of that by checking the cached retrieved data. However, the backend function also looks good as it's calling cursor.fetchall(). I lastly figured out that the loop that serialized the results was overwriting itself instead of accumulating and fixing it. I think it's important to have a clear mind while debugging and eliminate the possibility of glitched modules one by one.

Are there other things that changed comparing the final application with the original proposal?

No, we have already defined and explained most of the changes. We had to make minute changes to the advanced queries from stage 3 to make sure that they work with their functionalities properly.

Describe future work that you think, other than the interface, that the application can improve on

Although the current version of the application is already powerful and comprehensive, there are still a few points we could improve in the future. Firstly, we could work on enriching our dataset by sourcing authentic and more detailed vehicle information like photos, owners, mileage restrictions, and more.. Vast information could provide users a clear expectation about the current status of the car and ensure users' satisfaction by preventing them from feeling scammed. Also, we might work on a personalized recommendation engine. It will suggest the most suitable cars based on user profiles, rental histories, and other behaviors like requiring child seats. Moreover, we could also add support for multiple languages and localization on both UI and data, including currencies and time zones, to serve more regions.

Lastly, we could contribute to a smaller carbon footprint by adding relevant features for estimating potential emissions based on the model and historical data, adding a special logo for electric vehicles, and adding carbon-offset options and discounts for lower-emission vehicles at checkout.

Describe the final division of labor and how well you managed teamwork.

The work was divided based on different functionalities. Jun is responsible for the majority frontend development. Aryan, Dhruv, and Joey are responsible for various backend functionalities. Our team has an efficient and great workflow. We have two weekly meetings for task assignments and progress checks, respectively. This guarantees that everyone in the team has a clear image of unsolved problems and the progress of the entire project.