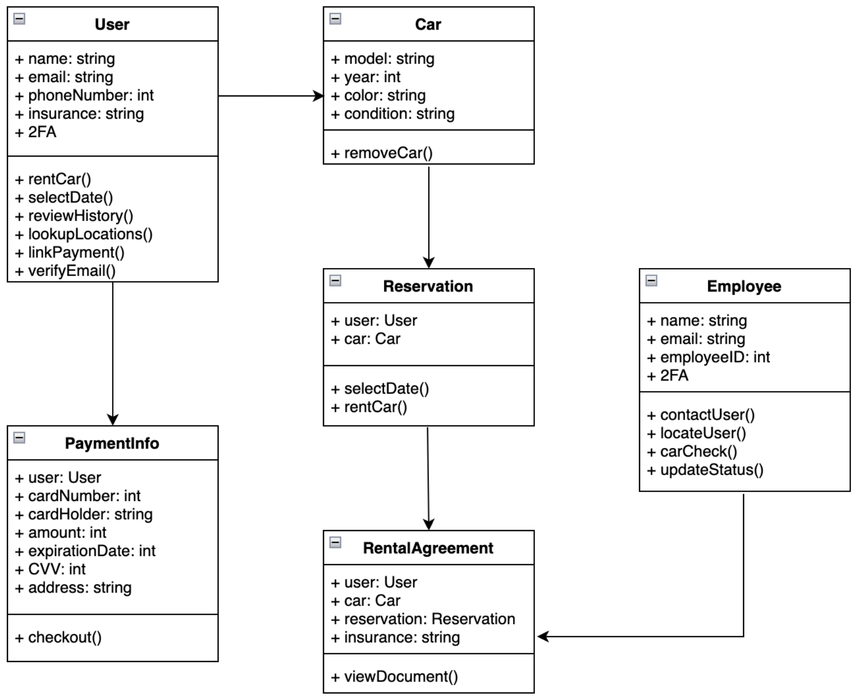
Car Rental System

Alejandro Pacheco, Kelly Aycock, Steven Trujillo

System Description

This system will make renting a car for any person an effortless task through this well-built car rental system. The system will operate through an app that can be downloaded on any mobile device or can be accessed in our locations at the front desk to make renting out cars easily. Users will simply sign in with their name, email, phone number, driver's license, and payment information and then be able to select through a large assortment of vehicles on the lot. In the app, the users after signing in will be directed to select the model, year, color, and how long they would like to reserve the selected car. To complete the process of checking out the vehicle the user will link their insurance information or be able to pay extra to go through the onsite insurance agency. After renting out the vehicle the renter or employees of the car rental business will be able to view the rental agreement at any time through the app. Employees will have access to more information through the employee login in the app which requires an additional employee id number to enter. Once logged into the administration side of the system employees can contact car renters in case of any emergency or to update them on new policies. On top of that, they will be able to locate cars that are currently rented out. Most importantly they can check the availability of cars to better assist customers and update the status of cars if for example they are damaged or out for maintenance. This system will also be made in a manner that it is able to be scaled to other car rental businesses. In total, this system will make renting cars easy for the renter and the employees of the car rental business.

UML Class Diagram



Description

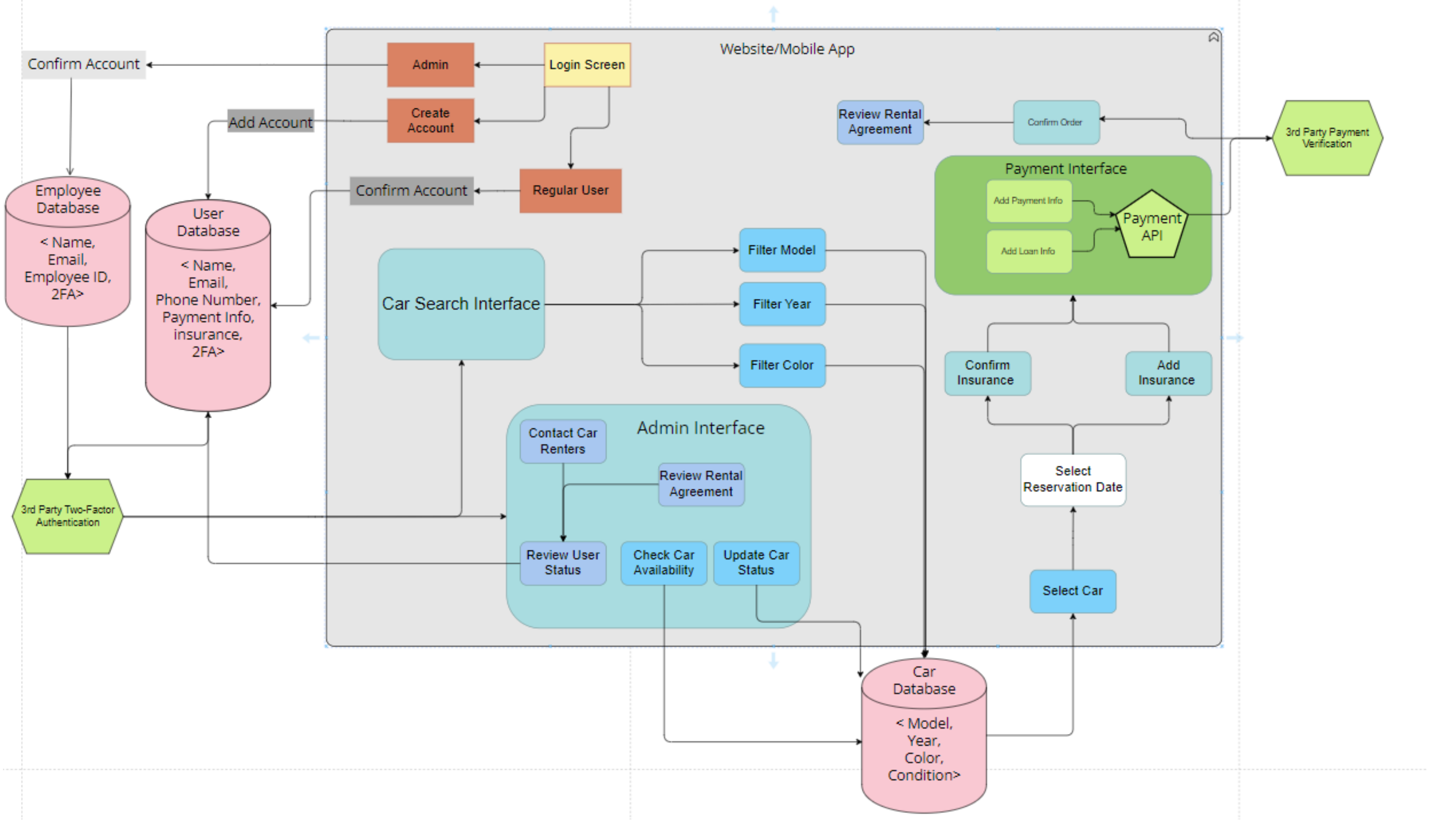
In this UML diagram, we have the User class acting as the base. This class contains the user's information needed by the company in order to add them to the car rental system. There is an additional two factor authentication for security purposes. The PaymentInfo class that is inherited from the User class provides the system the information needed for the system to allow a rental to take place. In addition, our user class contains other methods that will allow the user to rent a car, select a booking date, review their own history, among other functions. These all work together in order to give the user the best experience that will give them all the functionalities needed.

The PaymentInfo class enables the user to make payments for their car rentals. This object will store all the information about the user and credit card information. The class also has a checkout function that will let the user checkout after giving their account info.

In addition, we have a car class that contains models, year, color, condition, which are parameters that will classify our car class. If needed, we have a function that removes the car from the inventory if it has run its service or has no more function. The car class serves as the basis of this system, as it connects to databases and serves at the main pivotal point of the program. We also connect this class with a reservation that will store the user's reservation with their car. The reservation takes in parameters of a user and car and contains functions that will let them select their reservation date, and which car they are interested in. Lastly, this class connects with the rental agreement that takes in a user, car, reservation, and insurance info. This rental agreement will have one function that is viewDocument, which lets the user or employee access the rental agreement that was previously agreed upon.

Lastly, our employee class takes in a name, email, employee ID, and ensures two factor authentication is enabled. Our employee has access to functions like contacting the user, locating the user, checking a car status, and updating the car status. These functions are crucial to allow our employees to have superior actions over a user, and allow them to modify the stock and availability of cars.

Software Architecture Diagram



Description

This architecture diagram illustrates the various components of the car rental system. To begin our workflow, the user will be introduced to the website or app through a login screen. They will be unable to perform any actions here until they login or create an account. There will also be an admin sign in button at the bottom right that will allow employees to log into the system. After this, their credentials will be verified through the databases, and a third party two factor authentication system will verify the login.

Depending on the user that signed into the system, it will either direct them to the car search interface or the admin interface.

If our user is a customer, they will now be able to filter their car choice by model, year, or color. Once these parameters are chosen, the database will be monitored to determine which cars are available for rental. Once this is done, the system will now prompt the user to select a car, as well as a reservation date. From here the user will be prompted to confirm or add their insurance. The last vital step to this system is now having the user pay. They can either add their payment info like a credit card, or they can apply their loan information directly on the website. From here, a payment API will confirm the credentials and send the information to a secure third party payment source. Once this is done, the order will be confirmed and the user is able to review their rental agreement.

However, if our user is an employee they have several permissions to modify certain aspects of the program. Firstly, they can check car availability if needed, as well as update a car's status. Furthermore, they can review rental agreements that have previously been set, and contact renters if anything is needed.

Development Plan

The development for this system will undergo several different phases. The first phase is to build a functioning website with basic buttons and code flow. We'd like to establish a baseline for what the website should look like, and further the development from there. This step should include a login screen, a car feature selection page, and lastly a basic payment window. This step should take no longer than 1-2 weeks. To ensure the best quality, we're planning to finish this step in 2 weeks. Alejandro will be assigned to work for this part of the project.

During this time Kelly will start working on the payment API, as well as verification of our user's insurance. This part is not dependent on any other aspects of the program for now, so ensuring that a properly functioning payment system that interacts with the third parties will be crucial. It's a little challenging to be able to manage user's payments securely and safely, so this task will be given for about 3 weeks.

Now Steven will be assigned with developing and managing our database systems for our users, employees, and car inventory. He will be implementing this in SQL and other database management languages that will allow us to properly store and update our cars, users, and employees. This will likely take us about 2 weeks to complete, and will align with the initial first phase of developing the website.

Once Steven and Alejandro both complete their parts at around the same time, they can come together and start to develop the more intricate portions of the website. At this point code flows can be merged together to work simultaneously. This part will take coordination and proper usage of API's to allow for continuous development. This task is expected to be completed within 2-3 weeks.

Last part is taking Kelly's contributions with the payment system and incorporating them into the remainder of our system that we've developed. Once the payment system is fully integrated into our system, we'll incorporate a database or storage that will keep track of all of our customer transactions and insurance info. This may take 1 week at most.

Overall, the timeline for the implementation of this program will be around 4-5 weeks with most efficient production. We truly do shoot for around 6-7 weeks to get the software fully implemented and ready to go. Throughout all this time, we will be running tests to ensure that our progress is on the right track. Any different changes or modifications that are done during testing will be reported to the group and the timeline can be modified if needed.