Dylan Dvorachek
V00863468

SENG 330: Assignment 1

Problem Statement A: “Who put the driver’s seat back like that?”

A car-sharing company is looking to construct a software system in support of their daily operations. Customers “time-share” a car – that is, they pay a specific yearly fee, and that fee gives them access to a car for a maximum number of hours per year. That time may be used all at once, or it may be used in one-hour increments. There are various models of vehicles from sub-compacts up to small trucks. Vehicle depots are located around the city, and users of the service may pick up and drop off a vehicle at the most convenient depot. Individual vehicles must be maintained periodically which keeps them out of service for a short period. Once they have paid their fee, customers will want to both make their own bookings, along with checking their future and past bookings.

TABLE OF CONTENTS

Part A: DOMAIN MODEL AND GLOSSARY

In this section I define the ubiquitous language of the domain model within a glossary. Included are a list of assumptions that are made within the model and a list of questions for the client. This section ends with a rich description of the application domain of the problem.

Glossary:

User: A customer who has been registered within the software system.

User Account: A class which manages customer information and encapsulates the user when interacting with the application interface.

Employee Account: An employee account is used to manage the vehicles within the system. This account is the same as a user account, with higher level permissions.

Role: A class which is used to authorize the activities of an account. Each Account is within a role.

Driving Hours: The currency sold to the customer. Driving hours are equal to real world hours and are consumed for each hour the vehicle is in use.

Vehicle: Various models of vehicles from sub-compact cars to small trucks which may be rented.

Vehicle Depot: Acts as a pick up or drop off location for vehicles being used by customers.

Booking: Users may reserve a car at a specific depot, so that they are guaranteed a vehicle when needed.

Assumptions:

First time user registration must be done in person, as customer driver's license information must be entered into the system.

A yearlong subscription will start on the signup date, and will persist until the same day of the following year.

If a user runs out of driving hours before the end of the year, they may purchase another year long subscription. These newly purchased hours will expire a year from the purchase date.

Users may not book vehicles for periods that exceed the amount of driving hours they have left.

Beyond the first time sign up, all future transactions/bookings can be made online.

Questions:

How many driving hours are given to a user for a yearlong subscription?

Are there different subscriptions? (e.g. different amount of driving hours at varying costs)

How should a user proceed if they wish to add additional driving hours to their account?

For how long are vehicles decommissioned for maintenance?

How often are vehicles decommissioned for maintenance? (After every rental? Once a month?)

Description:

The car sharing company will use a software system in order to manage their daily operations. Customers pay a specified yearly fee in exchange for driving hours. These hours act as a currency, and may be used either all at once, or in one-hour increments. After a customer has finished registration/payment, they may immediately book a vehicle for use. There are a variety of vehicles from sub-compacts up to small trucks. Vehicle depots are located around the city, and users of the service may pick up and drop off a vehicle at the most convenient depot. For tracking and organization within the system, each user, vehicle and vehicle depot will be given a Globally Unique Identifier (GUID). Each individual vehicle's location will be tracked along with its status (e.g. parked in a depot, in use, under maintenance). Users may look at their booking history or future reservations by logging into the system and looking at their profile. Under the user's profile they may also view the amount of driving hours they have remaining.

Part B: USE CASES
In part B, ten use cases that are implementation independent are briefly described. A use-case diagram is created for three of the most important use cases.

Use Cases:

Register User: Create a new account for the customer and charge customer a specific yearly fee.

Purchase Additional Time: If a user requires additional hours, they should have the option of purchasing more.

View Time Remaining: Users should be able to view the remaining driving hours on their account.

Renew Subscription: At the end of the subscription period the user should be warned and given payment options.

Create Bookings: Users may reserve a vehicle of any type at a specified depot for use.

Vehicle Pick Up: The user may select an unreserved vehicle for use at any location. The depot inventory must be updated.

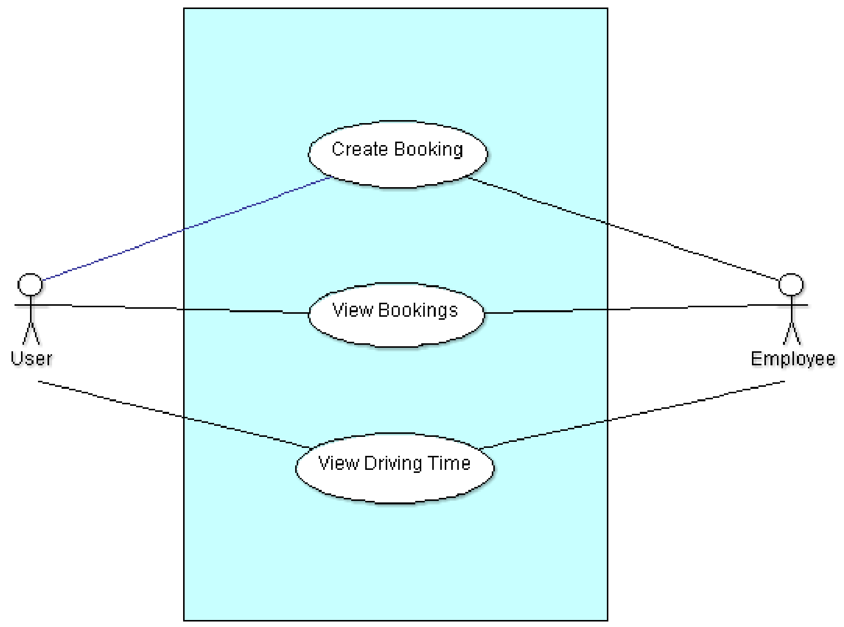
Vehicle Drop Off: Accept the returned vehicle and mark vehicle for inspection. Depot inventory will be updated.

Vehicle Transfer Request: If a location is understocked or if a specific vehicle is requested at a location, transfer of vehicles may be required.

Booking History: Users should be able to view their past bookings, as well as any reservations made.

Cancel Booking: Users should be able to cancel a booking they made.

Use-Case Diagram:



Part C: REQUIREMENTS

In this section I outline five requirements of the domain and business model. The requirements are broken down into: brief description, the preconditions for the requirement, the post-conditions, and then the basic course of action of how the requirement will be implemented.

1. New User Registration

Description: Gather user information and register the user into the system.

Actors: The customer, Account with employee Role, Role, Payment, Subscription.

Preconditions: The user passes a driving history check.

Post-conditions: The user will be able to book vehicles for use.

Basic Course of Action:

1. Create a user account within the system.

2. Update user account with information provided by the user. Due to the sensitivity of the information. Signup should be done in person.
3. A subscription plan is required upon registration. This is to avoid flooding the system with inactive accounts.

2. Create Bookings

Description: Users create a reservation for a vehicle on a certain date. Even if a vehicle if to be used immediately a reservation is made in the system for tracking purposes.

Actors: Account, Role, Booking, Vehicle, Depot, Schedule.

Preconditions: The user has enough driving hours in their account for the reserved period.

Post-conditions: The user and vehicle will have their schedules updated to reflect the reservation.

Basic Course of Action:
1. User logs into the system.
2. User selects the bookings tab.
3. User creates a new booking.
4. The app will prompt the user for preferred vehicle, location and date. Bookings that require a specific vehicle require time (to be determined by the client) for vehicle transfer.
5. Booking confirmation is required and then a notification is sent to the user.
6. Depending on far in advanced the booking was made, additional reminders may be sent.

3. Renew Subscription

Description: At the end of a subscription period, users should be given the option of renewing their subscription.

Actors: Subscription, Account, Payment.

Precondition: User's subscription is almost finished.

Post-condition: User will continue to have access to vehicle booking.

Basic Course of Action:

1. User logs into the system.
2. User selects the subscription tab.
3. A payment option and subscription type is chosen.
4. Confirmation is required and then a receipt is sent to the user.

4. Cancel a Booking

Description: A user may cancel a reservation on a car, prior to the date.

Actors: Account, Booking, Vehicle, Depot, Schedule.

Precondition: User has an upcoming reservation.

Post-condition: The user's reservation is cancelled.

Basic Course of Action:

1. User logs into the system.
2. User selects the current reservations tab.
3. User selects cancel booking from a list of reservations they have made.

5. Vehicle Pick Up

Description: A user picks up a vehicle for use from a depot, and an employee account updates the vehicle status.

Actors: History, Account, Vehicle, Locate Vehicle, Depot, Booking, Schedule.

Precondition: The user either picks up a vehicle they have booked, or one that has no reservations.

Post-condition: The vehicle status is updated to "in use".

Basic Course of Action:

1. Employee logs into the system.
2. Employee updates the vehicle's status under an employee only vehicle tab.
3. If the user hasn't made a booking with the vehicle, then the employee links the vehicle to the user.

PART D: CLASSES and UML class diagram

In this section, I have compiled a list of possible entities, value objects and service objects of which the system will be composed. A UML diagram has been created to show a first draft relationship between the classes.

Classes:

Entities:

- Account
- Role
- Vehicle
- Depot
- Subscription
- ~~Repo~~ // originally I was looking at repo as some sort of database data broker. Perhaps it is not needed at this stage?

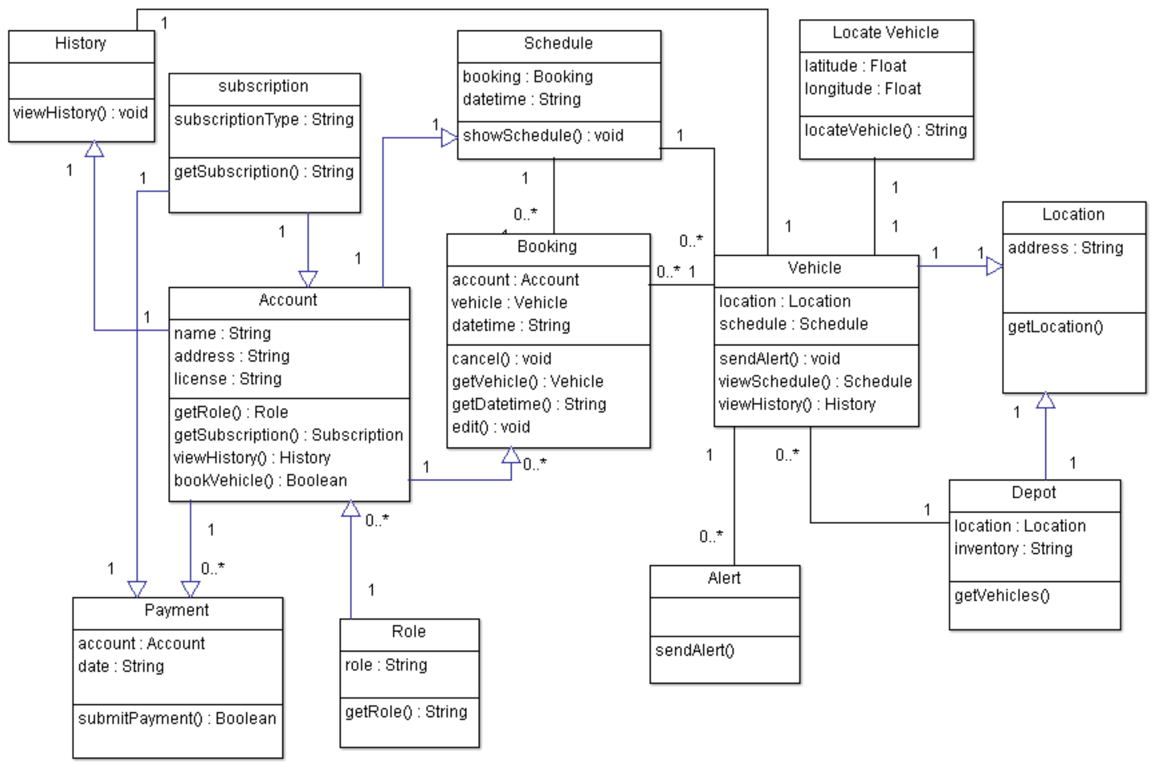
Value Objects:

- Location
- Alert
- Schedule
- Booking

Service Objects:

- Locate Vehicle
- History
- ~~Booking~~
- Payment

UML class diagram:



Changes Made:

Part C:

- Added actor field under each use case

Part D:

- Changed Booking to a value object
- Removed Repo class
- Updated UML Class Diagram to show relationships.