**Aaron, Claire, Rahil, Casey. Group 21. Parking Lot Management System**

**FINAL REPORT**

**RESPONSIBILITIES**

Aaron: Class ParkingLot, Class ParkingSpot, Class Vehicle, Final Report, and Update UML’s

Claire: Class User, Required Features

Rahil: Class Steward, Optional Features

Casey: Class Admin, Required Features

**UPDATED INITIAL DESIGN**

While coding are final project, we realized there needs to be an update on our design since it was flawed. For example, we created a ParkingLot, ParkingSpot, and Vehicle class with instance variables, the constructor, and getters and setters. We did this so that we can create and use these objects for the Customer and Admin Classes.

The Vehicle class is has instance variables int spotsNeeded; ie: 1,2, etc.  String vehicleSize; ie: small, large, etc. The spotsNeeded and vehicleSize need to stay consistent. Finally, vehicle also has a String user to attach one’s name to the car. The constructor and all getters and setters are generated from these instance variables. A toString method is also there in case we need to represent this object as a string.

The ParkingSpot class has instance variables Vehicle vehicle, so that one can park their car in a spot. There is an int spotNumber, to represent which spot has been taken. A int size, for the size of the parking spot. The getSize() should equal the getSpotsNeeded method. And finally, a boolean vacant which says if a spot is empty or not. Again, the constructor and all getters and setters are generated from these instance variables. A toString method is also there in case we need to represent this object as a string.

The ParkingLot class has an ArrayList<ParkingSpot> parkingSpots so that we can store the parking spots that are in a parking lot into an appropriate data structure. A parking lot also has a number of rows which is int rows. Int spots, which is the number of spots in the lot. This should be equal to the getSize() of the arrayList pakingSpots.  And finally, an int levels to determine how many stories a parking lot or garage has. For simplicity, we will set that to a number less than or equal to 5. The constructor and all getters and setters are generated from these instance variables. A getInfo is also made that acts as a toString method.

We also tweaked out admin class to best accommodate this program. We made static instance variables and methods so that we can run this program in main. We have String username, String password, String fullName, int userID, HashMap<String, String> usernamePasswordMap, HashMap<String, Integer> nameIDMap, ArrayList<ParkingSpot> parkingSpots, and finally ArrayList<ParkingLot> parkingLots. This class has a constructor, methods String signIn(),

Int carSize(), void addLot(), void removeLot(String remover), int GenerateId(), int numSpots(), void changeCapacity(), int numRows.

We made a class customer that would display the customer’s end of our program. We made instance variables int balance, String username, and String password. This class had a constructor, methods Boolean requestSpot(Vehicle myCar, ParkingLot myLot), ArrayList<ParkingSpot> rearrangement (ArrayList<ParkingSpot>), void charge(int cost), void pay(int payment), void userPaySystem(), voi printCustomerInfo(String filename, HashMap<String, String map), String getUsername, void setUserName(String username), void giftCoupon(), and void leaveParkingLot().

A lot of the methods in the customer class are actually needed and can be reused for the Steaward. Other notes for this project were things were a little bit jumbled but this was due to trying to work remotely with others and trying to keep everything consistent, so we kept committing and pushing multiple times. Even though our actual implementation is pretty different than what we designed, I feel like that was part of the process.

**UPDATED UML’S**

Below are the updated UML diagrams, sorry if some of the handwriting is messy, but it is everything I talked about in the final report but in UML language instead of English.

**A close up of text on a white background

Description automatically generated**

**A close up of text on a white background

Description automatically generated**

**A close up of text on a white background

Description automatically generated**