Virtual Project Feasibility & Analysis Report

Author: Mehmet Alper Genç

July 2020

1. INTRODUCTION

The project that will be presented and discussed in this report is a smartphone application that lets its users browse parking lots near the location that they desire to travel. The application will collect data through cameras located at the parking lots and will process these photos using image processing and machine learning to derive the necessary information.

This report will first argue the feasibility of this project. Afterwards, the report will analyze the project by giving an overview of the system, listing its functional and non-functional requirements and showing the system models related to the analysis. Finally, it will provide the user-interface mockups of the application.

2. FEASIBILITY

2.1. Target Audience

The target audience for this application is those looking to travel to busy and crowded locations, and could possibly face difficulties in finding a parking spot. I believe that for those seeking to travel to crowded locations, finding a parking spot could end up being very time-consuming, and they may end up having to park in potentially risky spots, such as the side of a busy road, that may end up damaging their car. Furthermore, this application could help those that are new to a particular city, since they would not need to be accustomed with the location that they are travelling to.

2.2. Comparison to Other Applications

2.2.1. Parkkolay

Parkkolay (https://www.parkkolay.com/) is a mobile application that provides a similar service compared to this project. They have a number of parking lots in Istanbul, Turkey that they have signed with, and they provide directions to, and possibly discounts at, these parking lots that they have signed with.

One advantage that this project has in comparison to Parkkolay may be on the subject of automation. In their website, the details of their algorithms are not revealed to the observer, though it is unlikely that they use any advanced technologies like machine learning while managing the information that they have on their parking lots. On the other hand, this project automatically gathers how many parking spots are empty in the parking lots.

One disadvantage that this project has compared to Parkkolay is the ease of use that they provide. Parkkolay provides discounts in some of the parking lots they have signed, and they have the functionality to enter the parking lot with a QR code once the user has paid the parking fee from the application. The project explained in this report currently has no such functionalities, but such a functionality could be added in future iterations.

2.2.2. Parker

Another such application that is similar to my project is Parker (https://play.google.com/store/apps/details?id=com.streetline.parker&hl=en_US). Parker is overall a more advanced application, as it provides sensor-driven data in contracted parking lots, and it has some other external services such as parking ticket avoidance, depending on the US state that the user is in. One advantage Parker has over this project is the number of services they provide. Furthermore, physical sensor-driven data could potentially have a higher confidence rate than image recognition. However, the field of image recognition in computer science is growing very fast. As image processing grows, I believe that its confidence rate will approach, if not surpass, the confidence rate of physical data, and it will be easier to retrieve data using images, as it is done in this project. So, in that way, the usage of image processing could become an advantage over sensors in the near future.

One advantage that my project has over Parker is the ease of setup. While Parker does provide sensor data in some parking lots, one could face challenges in setting up such sensors in small parking lots, or in getting parking lot owners to agree to it. Thus, the cost of implementing a project like Parker in real life could be more than the project explained in this report, which only requires a small number of cameras set up in certain locations in the parking lot. Furthermore, functionalities such as parking ticket avoidance could slow down the spread of this application, as it would have to be implemented differently in each state, country or city, as laws can differ depending on location.

3. ANALYSIS

3.1. Overview

The aim of this application is to help the users find parking lots that are empty. The application will track how full the signed parking lots are using periodically received image feeds from cameras installed in certain locations in the parking lots. These images will be posted to the API of the server using HTTP POST requests, which will then be processed through image processing and machine learning to get the necessary information about the parking lot. The implementation idea is that the image recognition module will retrieve the RGB data from the image and turn it into a format that can be handled easily by the machine learning module. The machine learning module will then receive this data, and depending on the RGB values of certain areas in the image data, it will determine whether or not a car is currently in that parking spot or not. It will then update the entry of the parking lot in the database with a certain confidence rate.

For the server-side implementation, Tensorflow could be used along with Python or JavaScript, as it includes both image processing and machine learning tools.

3.2. Functional Requirements

The system should let its users:

- Create an account
- Search for parking spots close to their destination
- Mark their favorite parking spots
- Get directions to a parking spot (which will be done by sending the location information to the phone's default navigation application)

The entire system should be able to:

- Process image data sent from the parking lot cameras
- Serve the data through the API

3.3. Non-functional Requirements

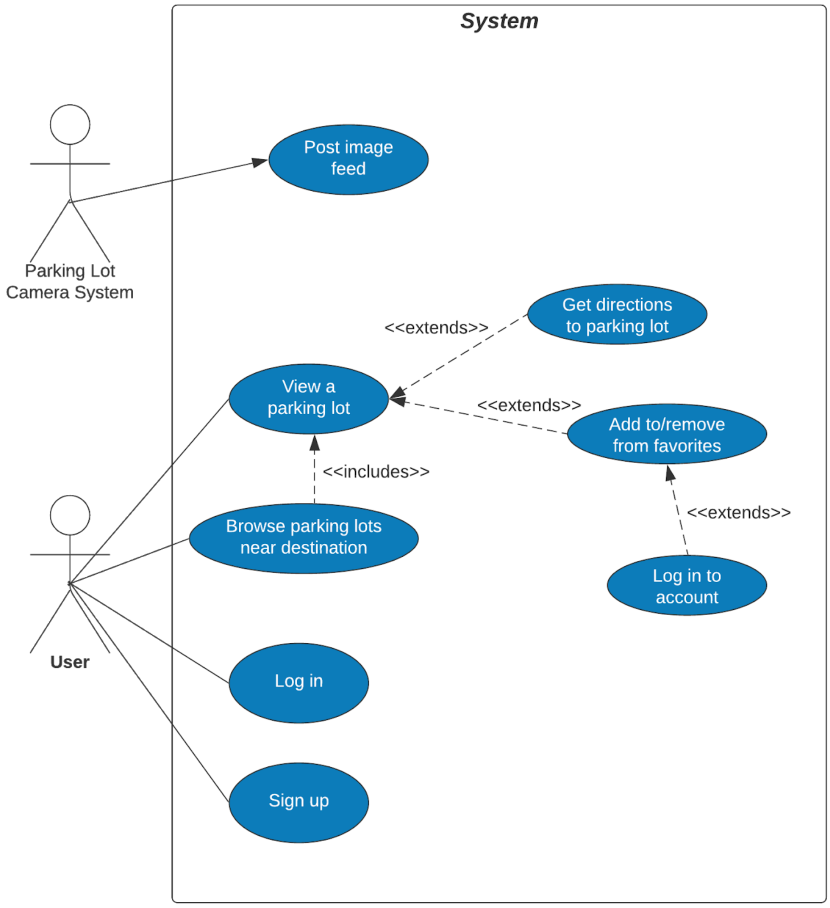
Scalability: If a parking lot is large, or is narrow, more cameras would be required to determine the fullness of a parking lot with high confidence, which is why the system should be able to correctly and completely process all requests even when the number of requests becomes large.

Maintainability: One of the main objectives of this application is to save time for its users, which is why errors in the system must quickly be fixed in order to provide correct information. For this, certain software design patterns and techniques should be used while writing the code to make debugging and fixing less tedious for the developers.

Reliability: Again, since one of the main objectives of this app is to save time, the system should not provide incorrect or erroneous data. To maximize reliability in the real world, the cameras should be placed in optimal locations to capture the data as clear as possible. In the software domain, image processing and machine learning algorithms used should be written in a way to maximize confidence rate.

3.4. System Models

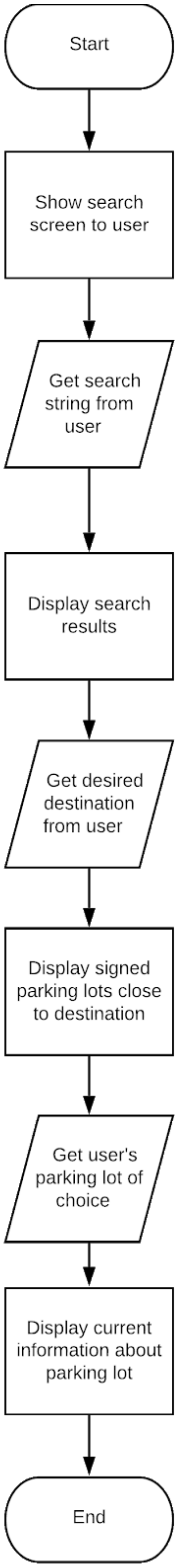
3.4.1. Use Case Diagram



3.4.2. Flowcharts

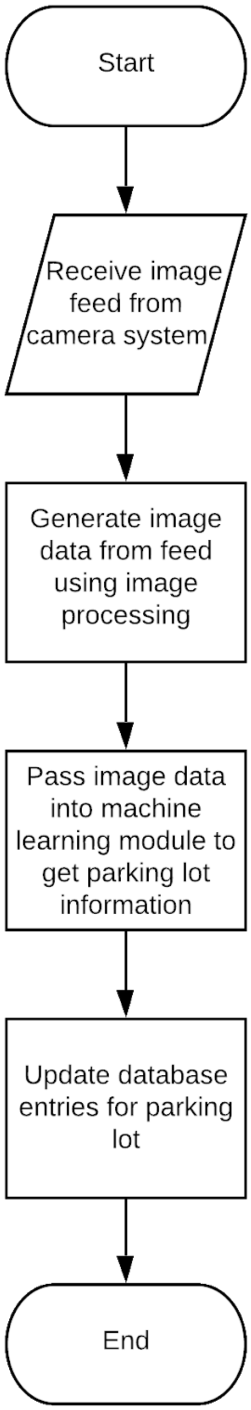
3.4.2.1. Flowchart for User-System interaction

This flowchart represents the most common interaction between the system and the user, which is the sequence of the user searching for their destination, and browsing parking lots close to that destination.

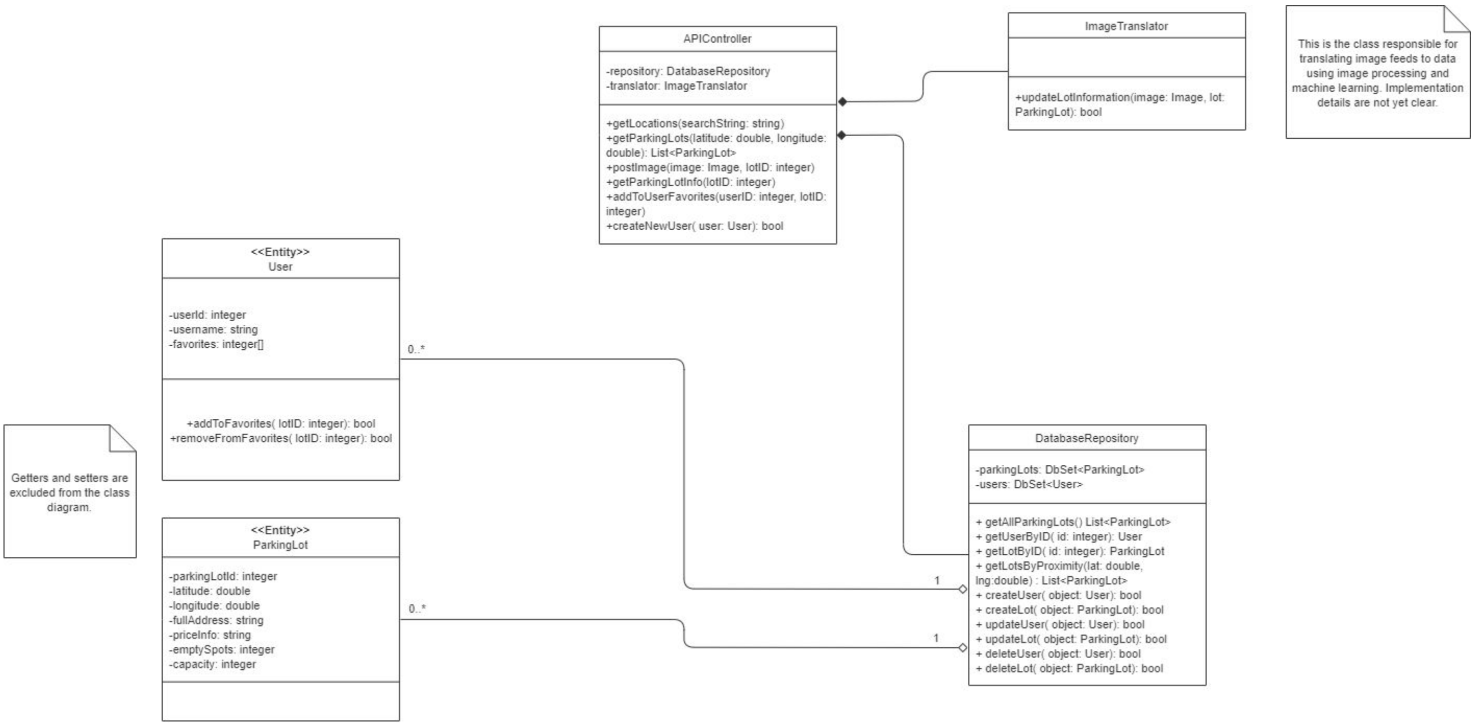


3.4.2.2. Flowchart for receiving image feed

This flowchart represents the sequence of processes where the system receives image feed from cameras and translates this image to parking lot information.

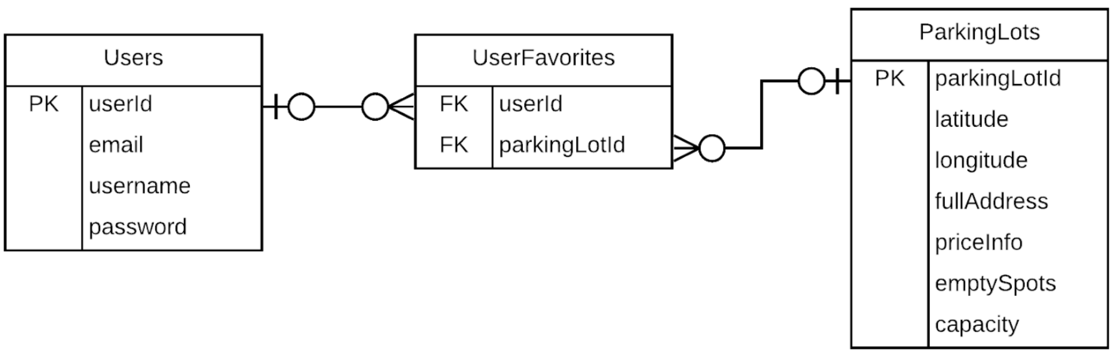


3.4.3. UML Class Diagram



A larger and horizontal version of this diagram can be found in Appendix A.

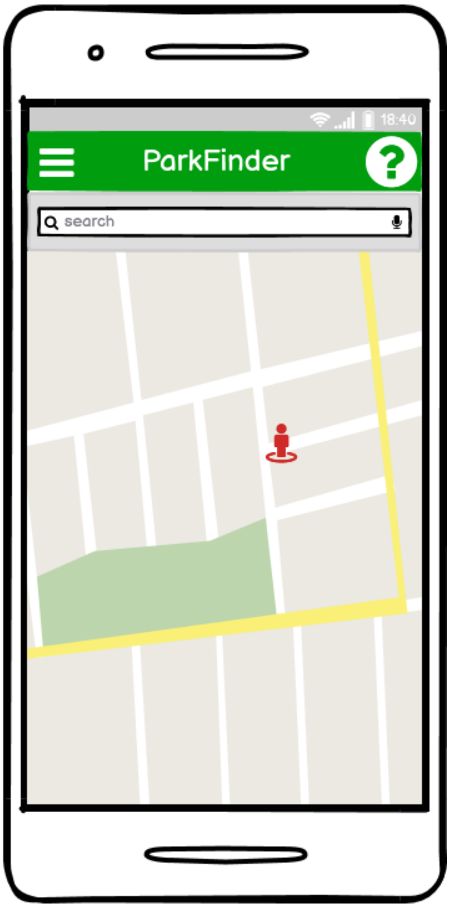
3.4.4. Entity Relationship Diagram



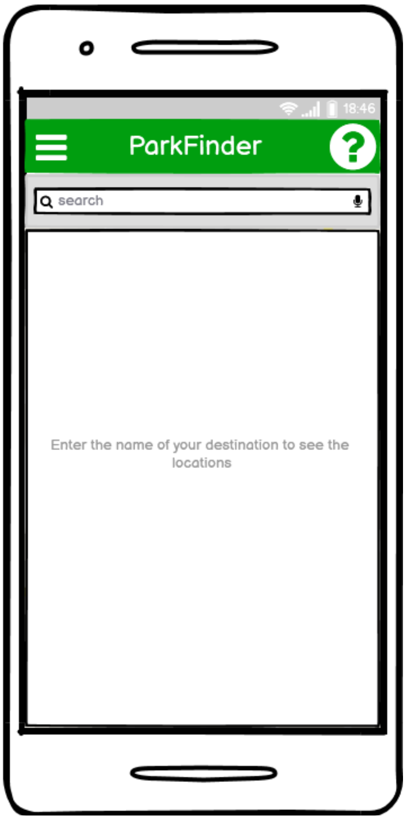
3.5. User Interface Mockups

In this section, the user interface mockups will be shown while following the most common usage sequence of the application.

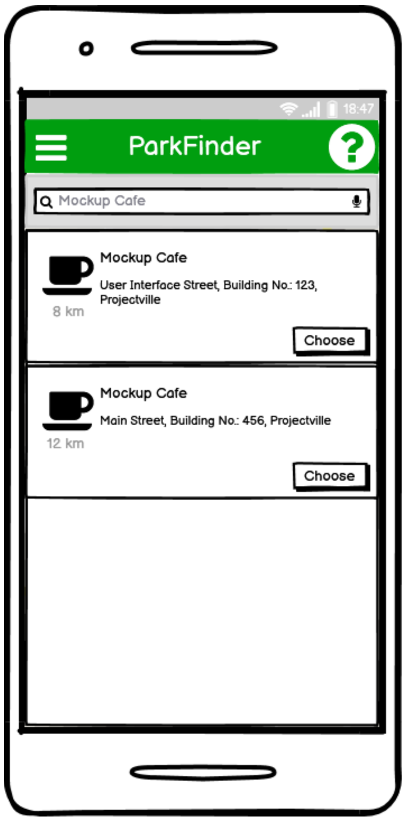
3.5.1. Initial Screen



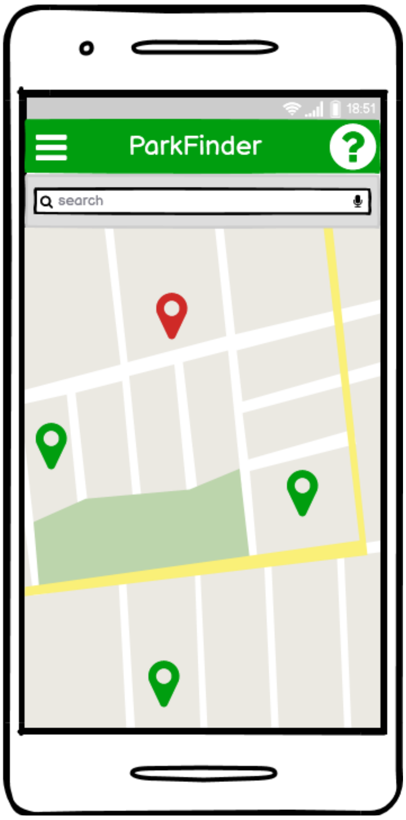
3.5.2. After Clicking Search Box



3.5.3. After Entering Search String

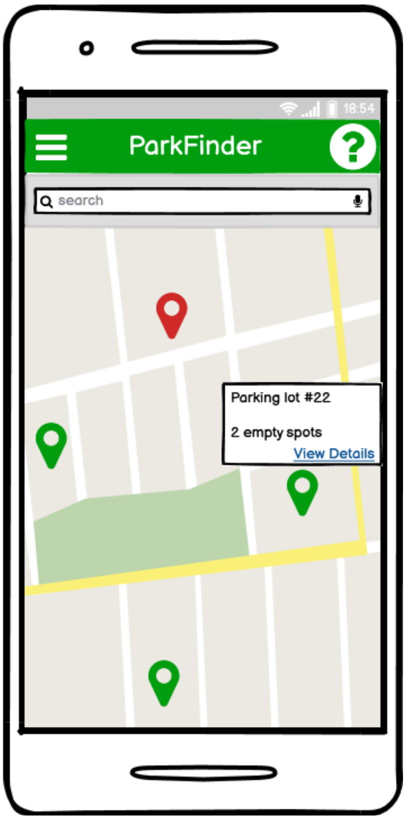


3.5.4. After Selecting Destination

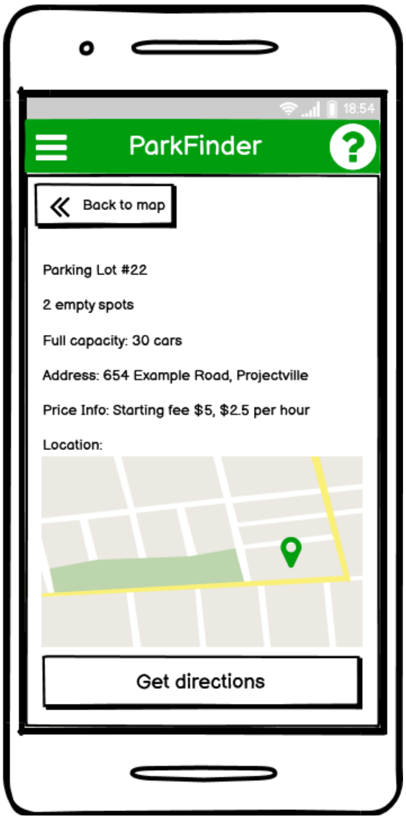


The red marker is the destination that the user seeks to go to, and the green markers are the signed parking lots that are close.

3.5.5. Clicking On Parking Lot Marker



3.5.6. Viewing Parking Lot Details

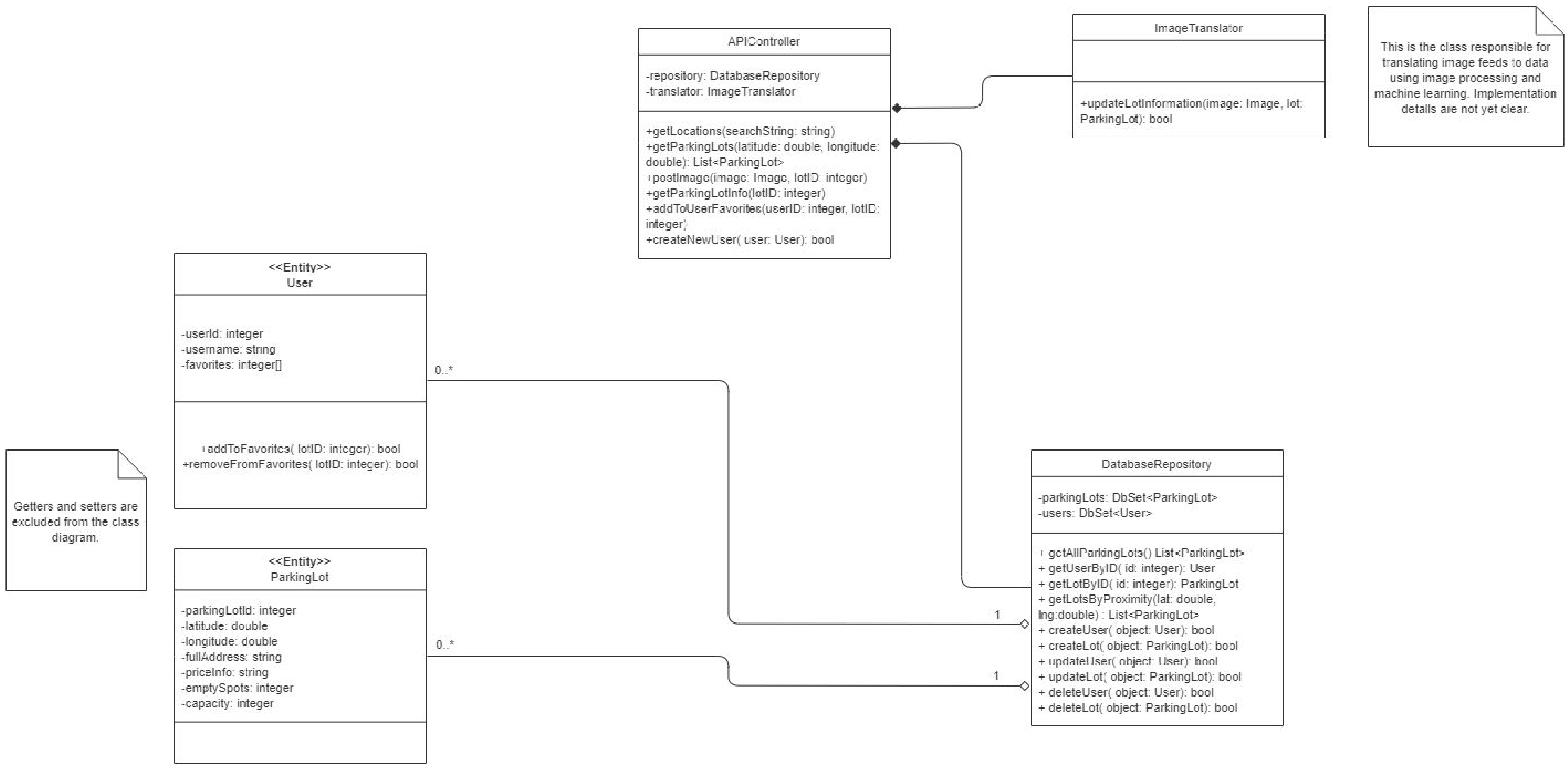


If the user chooses to get directions, they will be redirected to their smartphone's default navigation application.

4. APPENDIX

Appendix A: Class Diagram

The larger version of the class diagram is in the next page.



APIController
-repository: DatabaseRepository
-translator: ImageTranslator
+getLocations(searchString: string)
+getParkingLots(latitude: double, longitude: double): List<ParkingLot>
+postImage(image: Image, lotID: integer)
+getParkingLotInfo(lotID: integer)
+addToUserFavorites(userID: integer, lotID: integer)
+createNewUser(user: User): bool
ImageTranslator
+updateLotInformation(image: Image, lot: ParkingLot): bool
This is the class responsible for translating image feeds to data using image processing and machine learning. Implementation details are not yet clear.
<<Entity>>
User
-userId: integer
-username: string
-favorites: integer[]
+addToFavorites(lotID: integer): bool
+removeFromFavorites(lotID: integer): bool
0..*
Getters and setters are excluded from the class diagram.
<<Entity>>
ParkingLot
-parkingLotId: integer
-latitude: double
-longitude: double
-fullAddress: string
-priceInfo: string
-emptySpots: integer
-capacity: integer
0..*
1
1
DatabaseRepository
-parkingLots: DbSet<ParkingLot>
-users: DbSet<User>
+ getAllParkingLots() List<ParkingLot>
+ getUserByID(id: integer): User
+ getLotByID(id: integer): ParkingLot
+ getLotsByProximity(lat: double, lng:double) : List<ParkingLot>
+ createUser(object: User): bool
+ createLot(object: ParkingLot): bool
+ updateUser(object: User): bool
+ updateLot(object: ParkingLot): bool
+ deleteUser(object: User): bool
+ deleteLot(object: ParkingLot): bool