Information-Centric Design of Context-Aware Systems

Course Project

Parkbaan

Milad Gholami

1. Abstract

When campus members forget to check event schedules going on campus, they might park their cars in parking lots that are supposed to be emptied for those events at specific times. This is the cause for many parking citations received by campus members every year. This project aims to solve this common problem of car owners in campus community. Our system is supposed to inform the users to move their car from a parking lot before they get parking citation. We have designed a simple mobile application that is very easy for the user to work with and can do this job efficiently.

2. Introduction

Parking is a primary problem for everyone and affects college campuses as well. Consider our own campus of University of Maryland. One of the problems that many students, faculty members and staff face on campus every day is getting parking citations. Many people get citations on campus not because they intentionally park in the spots that they do not have permit for but because they are not always aware of the events running around the campus. These events include important games that hold in the stadium. Although the schedule of events are informed to the drivers via email and through website, most of the campus drivers miss reading these emails and checking the schedule on the website since they are too busy to do so. This project aims to design a simple system in order to prevent campus community from getting unwanted citations.

The main goals of our system can be summarized in three main categories. The first goal of our system is to inform the user to move her car at a certain time before her parking permit expires. The second main goal for the system is to be able to respond to the queries of user about having permit to park in a certain parking lot and for how long the permit lasts. The user is also informed about the duration of her permit for the specific parking lot that she parks her car upon parking her car. The final main goal is to inform the user of the parking lot that her car is parked in upon her request.

The user scenario for our system can be summarized as the following. Upon parking the user must push a button to inform the system that she has parked her car. By the use of GPS, the system is supposed to figure out which parking lot she has parked her car in. The system must also inform the user of the time her parking permit expires upon user's parking. Checking permit for any parking lot and the duration of permit is available for users on system. The user is notified at a specific time before her parking permit expires by the system. The user can also ask for the parking lot number that her car is parked in, in case she forgets.

For implementing this system, we initially had many ideas. One was using cameras on parking doors or at each parking spot. Although this could be quite beneficial for our system in the sense that our system could provide the users with other additional and exact information, this idea would be too costly to be practical. Therefore, we decided to use cell phones which do not pose any additional costs for us. Using cell phones, we had different options for choosing communication channels such as texting and mobile applications. In the initial steps we intended to use texting in order to make it usable for those campus members who do not have smart phones and in order for the users to be able to use the system without internet connection. However, there are many cons associated with this infrastructure such as unreliability of texting infrastructure which removed this idea from our consideration. Finally we decided to design our system as a mobile application which does not pose any cost on user. Also people in today's world are comfortable with mobile applications to a high extent.

The architecture, data base and user interface of our system are explained in details in the next sections.

In the final section we propose some ways to improve our application in the future. These improvements include leading users to less crowded parking lots, realizing that the users have parked their cars in a parking lot without requiring them to inform the application manually, and informing the user not only about the parking lot that she has parked her car in but the location of her car in that parking lot upon her request.

3. Motivation:

One of the big problems for the drivers around college campuses is getting parking citations. These citations are sometimes just because lack of knowledge of important events running around the campus which require certain parking lots to be emptied for those participating in these events. These events might be ceremonies or important games in university stadium. As an example if there is a basketball game in the campus stadium, the parking lots around the stadium are supposed to be emptied for people who want to watch the game, and the driver of any car parked in these spots would get a parking citation unless she has bought a game ticket. The parking permit of a campus member might be expired any time during the day based on the starting time of the events. The schedule of events and requirements for moving cars are informed by the Department of Transportation Services (DOTS) to anyone via email. Furthermore, these requirements can be found on the website of this department (http://www.transportation.umd.edu). However, students and faculty members usually are very busy and might miss reading the emails sent by DOTS. Also there are rarely students who check out event schedules for parking permits on this website regularly. These problems are the main motivation behind this project. Our goal is to design a system to help the campus members with these issues such that they do not receive any parking citations just because they are not aware of some events. We aim to make this system as simple as possible. This is very important in the sense that people should have an incentive to use our system rather than searching for the relative emails or checking out DOTS website.

4. Goals

The main goals of our system can be fit into three categories. First, our system is supposed to inform the user to move her car at a certain time before her parking permit expires. As an example consider the following situation. A student has parked her car in parking lot 2 at 10 a.m. There is a game starting at 6 p.m. Therefore, her parking permit expires at 3 p.m. In order for her to be able to move her car in time, our system is supposed to inform her at 1:30 that her parking permit is about to be expired and remind her also at 2:30 if she has not moved her car yet (These specific time limits for reminding the users are just examples to illustrate what our system does. Our system does not necessarily work with these time limits).

The second goal of our system is to be able to respond to the queries of user about having permit to park in a certain parking lot and for how long if she does. This includes the time that she parks her car in a parking lot. She is informed of the duration that her car can be parked in this specific parking lot. As an example the user might request the system whether she has permit to park in parking lot 1 and for how long. A reasonable question might arise here. What is the point of informing the user of her permit being expired when we inform her about the duration she can park in a specific parking lot firsthand. The user can decide not to park in this parking lot if she wants to leave after the time her parking permit expires instead of going through the trouble of parking once and moving her car again. Consider a student who arrives the campus at 10 a.m and her classes are finished by 4 p.m. This student is informed by our system that her parking permit for parking lot 1 is expired at 2 p.m. Therefore, it is reasonable for her not to park her car in parking lot 1 instead of parking in this parking lot and moving her car again to another parking lot at 1:30. Hence, It might seem that considering the option of informing users about expiration of their permit at a specific time is irrelevent given the fact that we can inform them of the availability of parking lot exactly at the time they park. There are two reasons which address the need for this option in our system. First, the schedule of each person may vary throughout the day. For example, suppose a student arrives to campus at 10 a.m and knows her class is finished by noon. Therefore, she assumes she can safely park her car in a specific parking lot in spite of the system's notification that her parking permit in parking lot 1 is expired at 2. However, after her class the professor might ask her to attend a meeting which might end at 4. If she forgets about the notification received in the morning, she would end up getting a parking citation. Another reason is that many students might be in a hurry when they arrive to the campus. For example if a user's class starts at 10 a.m in CSIC department and she arrives to the campus at 9:57, she might want to park her car as fast as possible in the parking lot closest to CSIC department, and is probably not paying attention to the fact that she is leaving in the afternoon and her parking permit might be expired by that time.

The final main goal is to inform the user of the parking lot that her car is parked in upon her request. This goal originates from the common problem of many drivers who forget in which parking lot they have parked their cars. Suppose a student arrives to campus at 9:20 a.m and wants to park her car as fast as possible in order to be able to get to her class on time. She might be careless about the parking lot she chooses to park her car in. Therefore, she must be able to check this on our system whenever she wants.

5. User Scenario

The user scenario for our system is as follows. The user is supposed to push a button when she parks her car to inform the system that she has parked her car. The system then realizes which parking lot she has parked her car in and informs her for how long she can park her car in this parking lot. For example, a student might arrive to campus at 10 a.m and park her car in parking lot 1. When she pushes the button the system realizes that she has parked her car and logs her parking lot using GPS. She is then informed that her parking permit expires at 2 p.m.

The user can also check whether she has permit to park in a specific parking lot whenever she wants. The user is also informed about the length of time that she can park her car in a specific parking lot upon this request if she has permit to park in that specific parking lot at all. For example the user might ask the system at 10 a.m whether she has parking permit to park in parking lot 3 and receive an answer from the system that she has parking permit until 2 p.m.

The user is also notified whenever her permit is about to be expired. This notification is sent at a time such that the user can conveniently move her car by the time her parking permit is expired.

Finally, the user can request the parking lot number where she has parked her car, and get the number of the parking lot upon request.

6. Implementation

We considered many different ideas for implementing an efficient system. The first idea was using cameras on the entrance/exit doors of parking lots. These cameras were supposed to check and log the plate number and arrival/leaving time of each car. Checking the plate number were supposed to be done using image processing techniques. This idea if implemented could be quite helpful and would provide us with a lot of information in addition to the information of entrance or exit of a single car to a specific parking lot. For example at each time we would know how many cars are parked in a specific parking lot. This could help us to inform users which parking lots are less crowded and lead them to those parking lots. Unfortunately, this idea is impractical. First of all implementing this idea would cost a lot. Each of these cameras are supposed to be able to read the plate number of each car. This requires high technology cameras with image processing abilities. It would be very time consuming to design and implement such cameras. Cameras similar to the ones explained above have already been designed. However, even buying and installing them on the doors of the parking lots around the campus would be quite costly. Consider the parking lots around the campus. Some of them have many entrance/exit doors. Furthermore, for some parking lots around the campus there are actually no specific doors and the cars can enter or exit from anywhere around the parking lots. This issue requires installing many of such cameras. Even by installing more cameras we cannot guarantee that entrance or exit of some cars would not be missed by our system. Considering all these problems, this idea was repealed in the initial states of our work.

The previous arguments led us to using cell phones. The most dominant advantage of using cell phones as the base of our system is that a very high percentage of people have them. Furthermore, it does not pose any additional cost for us. All in all, using cell phones is quite more reasonable than designing and installing cameras around the campus.

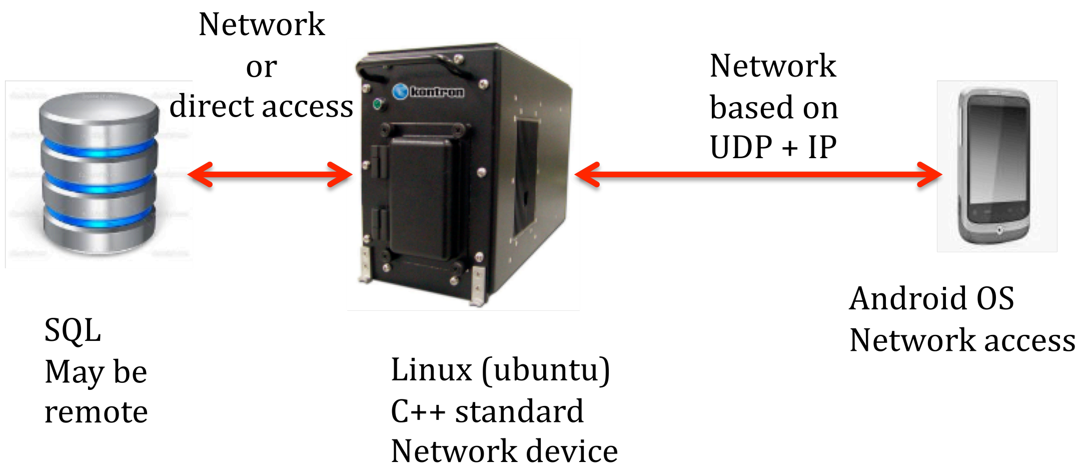
Using cellphones for our system, we had different options for communication channels. Our first suggested communication channel was texting. The motivation behind using texting was to make our system usable for those who do not have smart phones as well as the others. Furthermore, the users would not have to connect to internet and could access our system without using their mobile data and before arriving to the campus. The user scenario using texting as infrastructure would be as follows. The user sends a message to the system upon parking with her parking lot number. The user receives a message from the system at a specific time before her permit is about to be expired. However, the cons of using texts in designing our system outweighs the pros. First of all it poses cost on users which may discourage many users from using our system. Also texting has a week infrastructure and is unreliable. These reasons led us to design a mobile app instead of using text as the infrastructure of our system.

Mobile apps are internet applications which are supposed to run on smartphones and some other mobile devices. Mobile applications help users by connecting them to Internet services. The pros of using mobile apps as the base of our system are quite obvious. In today's world, mobile applications have become an inseparable part of people's lives. People use mobile applications in order to transfer money from their bank account, schedule their daily program, catch the buses on time, remember their important tasks, etc. The three main advantages of mobile applications are speed, volume of information, and advertising. Mobile apps significantly decrease the waiting time that people had to face working with computers. Furthermore, updating the little information is done in the background which allows access to even more information for users. Mobile apps if designed in a good way, can be very easy to work with for users, and can become popular among people quickly.

7. Architecture

Now we explain the main architecture of our system. We use a three layers structure at the first layer we have users which are using a mobile device in our case cell phone (this could be any mobile devices that have smart OS). In this layer users have an application that simply provide them an interface to interact with our system. They basically have their plate number in the application wants to verify if they are eligible permitted to park.

In the second layer we have a server that has a server program which has been implemented in C++. In this layer we basically made the network backbone program to communicate with users and database (we will explain about database in detail later). The networking program is very simple and use UDP+IP to communicate with cell phones. One might ask why UDP and why not TCP? The answer is because we do not have that much over head in our system and we do not need to keep a session always on and our data are very short. Our server is Linux base. We chose Linux (Ubuntu) because it was very convenient to have the C++ and many libraries available in open source.



The third layer of our architecture is Database which we elaborate more in the next section. This figure illustrate the overview of our architecture.

8. Database

In this section we describe the structure of the Database used in our system. As we explained earlier, our system needs to save the cars' information into a server(s) and load the information when a query request has been sent by a user. So far, we described the general architecture of our system. In our architecture the database needs to be connected to the server(s). We explained how the data could be sent from a mobile device to the server(s).

Our choice of database could be several database systems such as Oracle, SQL, MySQL, DB2, … . We found MySQL very convenient for our task. Oracle was too heavy and complicated for our task. We will describe later in this section that we only need three tables that can cover all the needs in our system. As an extension in our system one might want to save more data but in this project we stay only with the essential information. Among all the other possibilities of Database systems we chose MySQL because it was very easy to connect to any server systems and also it was very straightforward to connect to C++. Since the program that makes connection between server and Database system and mobile device is based on C++, we would be more consistent in general structure of our system by MySQL.

As we mentioned above, our Database is a relational database which has three tables that are related to each other via Primary and Foreign keys. Lets first do a brief study on what information do we actually need to save. When we see the scenario from the user point of view, all the user wants is to press a button when he gets off of his car. Therefore, the main information at that time that is needed to sent to server is the user's plate number and current parking lot information. Parking lot information is nothing but GPS coordinate with a radius error bound but here we use the parking lot number we will have later the another table in our DB that preserve the parking lots info there we save the GPS coordinate for each parking. Obviously, the exact timing is also needed, in order to verify the permission. The timing information must be of the precision of minutes (in our campus). We call these information as current parking information. It is more like issuing a parking ticket for the user when the query has been sent.

Once these information has been sent to the server after some sanity check now system must response to the user and let him to know that if he is eligible to park in this parking or not and if yes for how long. And also system must alert the user when it is about to expiration of the current parking ticket. Therefore, the most essential information that is required to be able to verify the park permission are user's registration information and current parking status.

In this paragraph we explain the data needed to be saved for parking registration. The very general information about registrations is the duration of the registration for every user. It could be of the precision of days. We use encode the duration by an issuing date and an expiration date. Obviously this is not enough to verify the current ticket permission for park. We also need to know for this particular user what the weekly schedule is. For example, undergrad student might not be allowed to park their car in weekends. Therefore, we need to save the weekly permission flag for every user for every days in week. Usually in campus all the parkings have a time limit . This time limit shows that in what time of day the permission is needed for a car to park in the parking. Also these timing might change due to some events. For example if there is a basketball game in a particular day then the time limit for the parking nearby the events would change on that day. Therefore, it is very crucial to be updated about the time limit at each day for each parking. In order to verify the exact park permission for a car in a particular parking we need to save these timing limits in database. We designed two fields for these timing limits in the registration information which is indicated by starting time and ending time. These are of the precision of minute for campus. Other issue that we must take care of it is that for some events some students are not eligible in the parking and some others are eligible, so we need to have a bit of information that if the reqested query is from some one that is actually permitted for that particular day or nat. We implanted that by a permission flag in the registration table. Each table needs a primary key in order to identify each row of table for the registration we use the plate number of cars as their primary key because they are unique for every car in the country or at least state. The registration table alone can not be useful because we need to have updated information from parking lots to be able to check that the assigned parking is on the work or not. Therefore, we need to connect this table to the parking info table (we will describe later). In the relational databases the connection between tables can be made via foreign keys. Foreign keys are unique keys that are primary key in other tables. Here we have the parking lot number as primary key for the parking info table so we use the same field in the registration table as a foreign key to connect these two tables.

As we mentioned above we need to have the updated information about each parking lot. Some times it happens that a parking lot is under reconstruction and has to be closed and then an alternate parking lot would be suggested by Department of Transportation Services (DOTS). Having a table separately for parking information is an inevitable need for our system. In this table we have a number for each parking which is unique across campus we also may have a name assigned to each parking but the name may not be unique. We use the parking number as a primary key for the table. We also require a location information for each parking by that we mean we need to know where is the parking with a particular number located in the campus area. The exact location of the each parking are measured by GPS coordinate numbers. For each parking lot we can have the estimated area of the parking lot and estimate of its center. We measure the location by the GPS coordinate of the center of each parking and then we set a decision boundary by considering a radius that covers the entire parking lot with a circle. Basically, we say that if a GPS coordinate was located in a certain distance of the center location of the parking we can verify that the car is in that parking lot. As we discussed earlier we need to have the up to date information about each parking therefore, we need a field in this table that inform us about current situation of the parking. Here we used the most simple information which is a binary status value of a parking and just says that whether the parking is available or not.

We explained the three main table of our database. Here we list the three tables with their fields:

Parking Info:

• Parking Number (primary-key)

• Parking Name

• Parking GPS coordinate

• Parking Status

Registration Info:

• Issue Date

• Expiration Date

• Days Flags

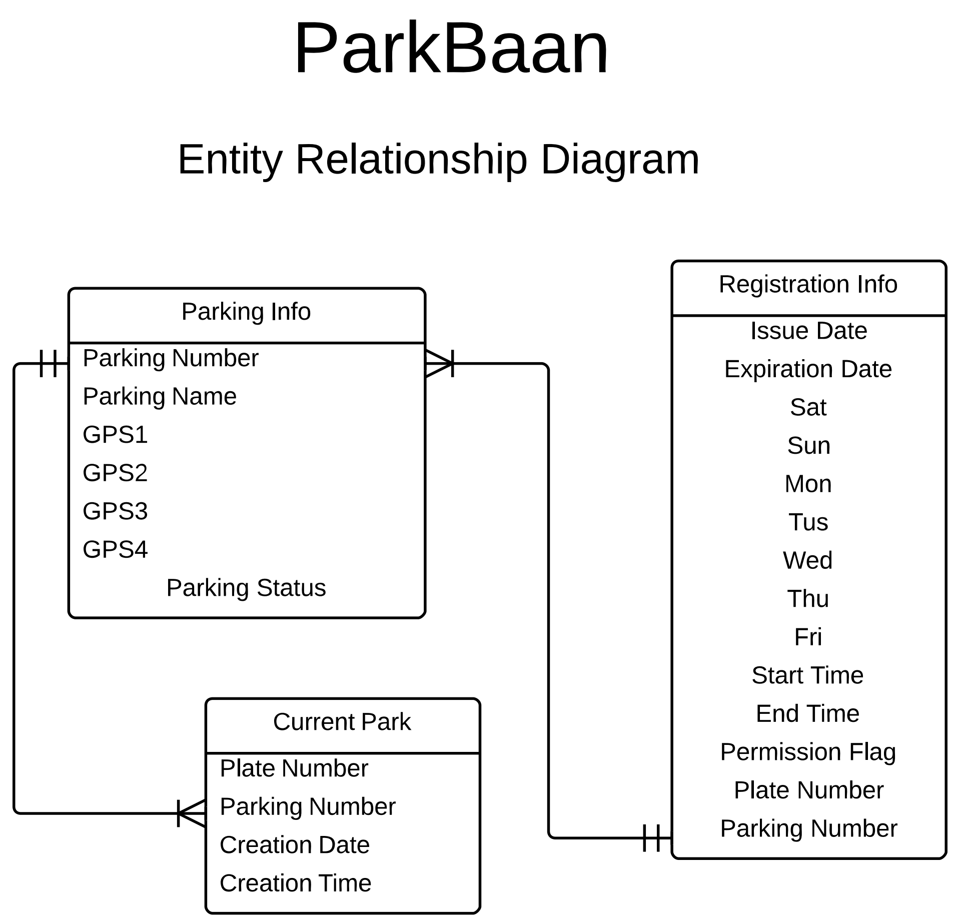
• Sun, Sat, Mon, Tues, Wed, Thu, Fri

- Starting Time
- Ending Time
- Permission Flag
- Plate Number (Primary Key)
- Parking Number (Foreign Key to Parking Info Table)

Current Park Info :

- Plate Number (Primary-Key)
- Parking Number (Foreign-Key to Parking Info Table)
- Creation Date
    - *The day of park is important for checking the eligibility*
- Creation Time

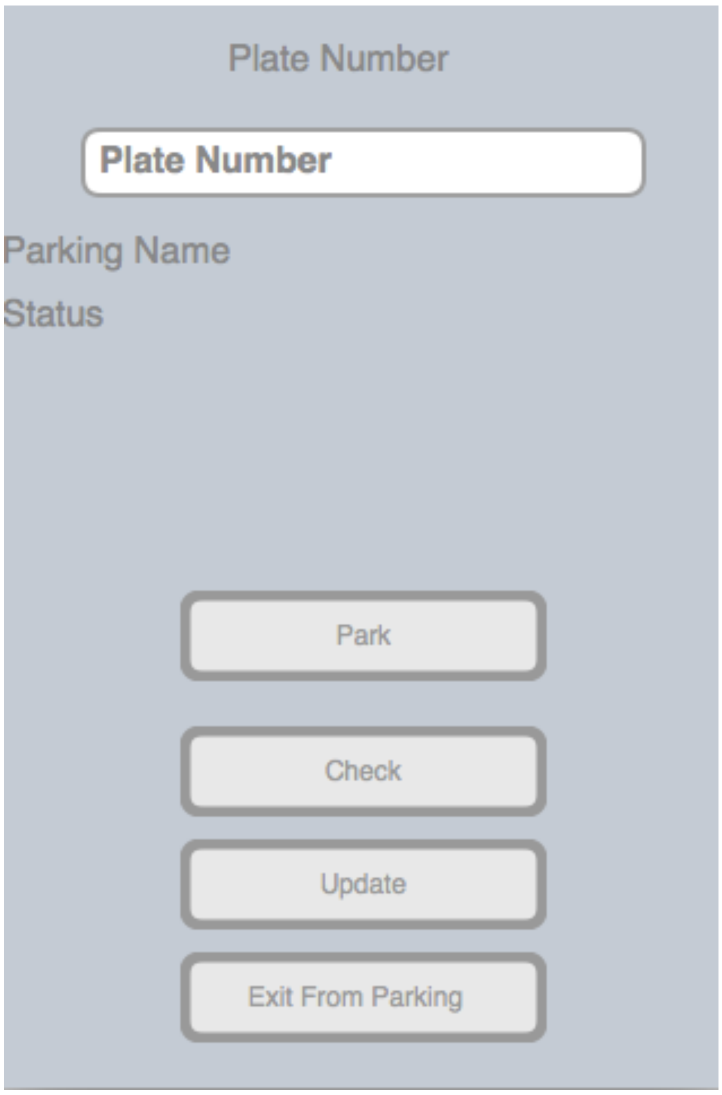
Now we illustrate the entity relationship diagram of our Database. This diagram represent each table in our database as an entity that has a relation with other entity and in general these tables create an schema of our database.



9. User Interface:

Now in this section we explain the user interface design in our system. The key of success in our system is simplicity. Being simple is the most important point of any interactive systems. Usually , users are not convenient when they are facing with a system that forces them to perform many complicated tasks . More specially , in our system that user wants to check the permission of his current park, he may be out side of the car and don't want to spend a lot of times to figure out if he is allowed to park or not. Depending on many situation it may be really critical to be simple. For example, if it is raining then standing on rain is not desirable for anyone or it might happen that someone is in hurry to get to a meeting and then just to enter his park information and doing many complicated stuff is not convenient.

By using all has been said in above, we end up to design our interface to be as simple as possible that user just need to press a button and wait a few seconds for a response. We had several choices of OS to implement or interface on it. iPhone, Android and Windows mobile. We chose Android because we had an easy time to program the android with C++. Also, Android is one of the most popular mobile OS that is installed on many cell phone devices. We found Android Developer Tools (ADT) in the Eclipse as a very convenient environment to implement our program for Android OS. The ADT has a simulator that allows to set up the entire system on PC and program on the cell phone easily. Basically, it provides a graphical interface for programming on Android. For the first page that appears on the monitor of cell phone we have this figure :



At the very first time after user registered in the server he just needs to enter his plate number and push the Check button . When the Check button has been pressed the system acquire the GPS location of the car send the plate number along with the GPS coordinate to our server and then the server verify the the permission and sent back a message that shows the permission of park.

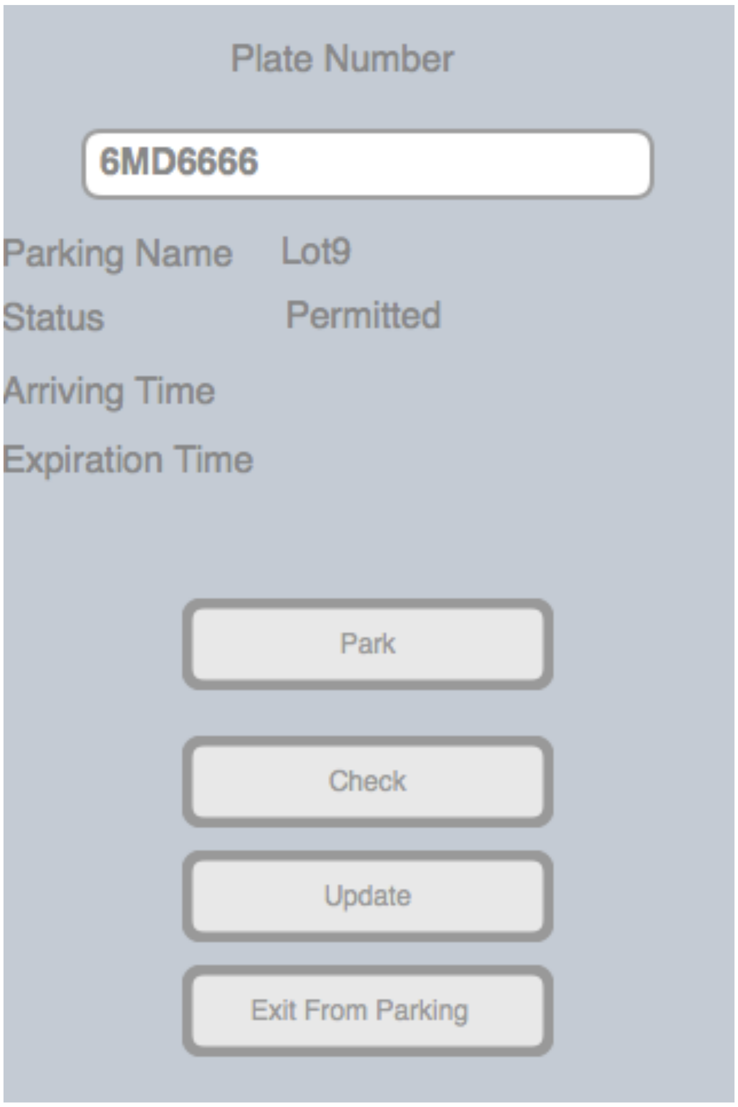


Plate Number

6MD6666

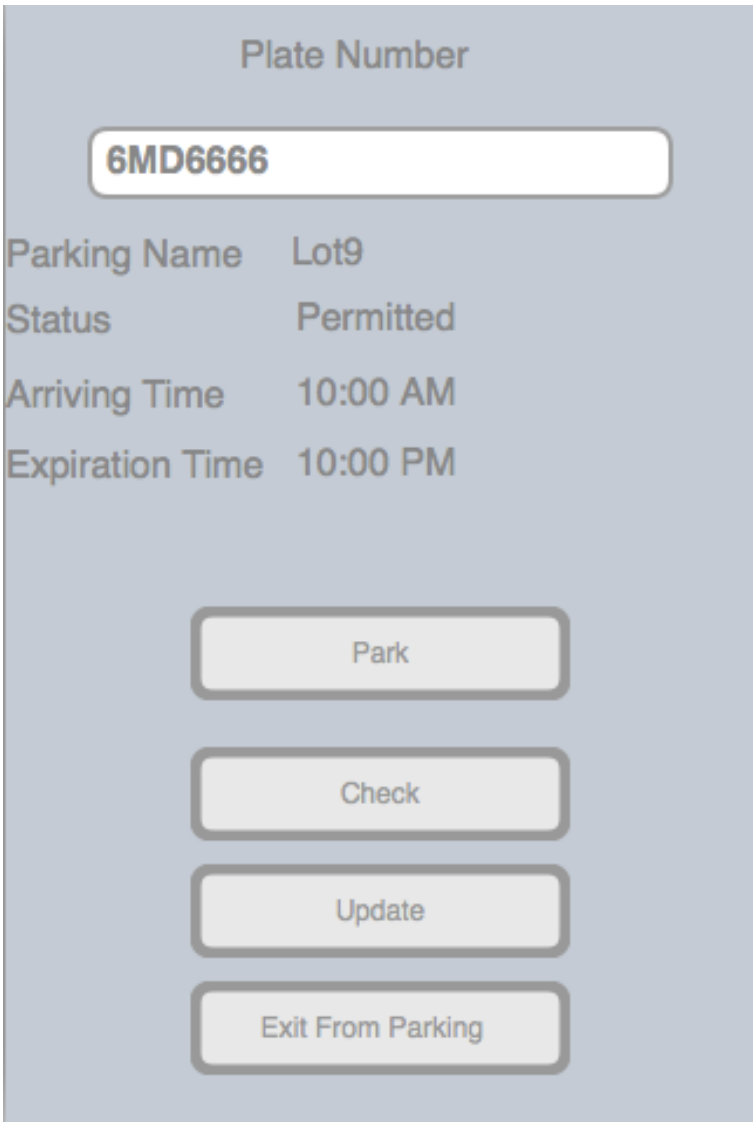
Parking Name Lot9

Status Permitted

Arriving Time

Expiration Time

Once the user received the permission message, if he was permitted then he press the Park button. By pressing the Park button he commits the query and the ticket will be issued for him. And system sent him a message that for what period of time on that they he is permitted. A user may want to check the status of his park after his park. Suppose that someone sitting in his office and want to check that if his car is still safe to park or not. We created an Update button that if the user press it will get the updated information about his current park.



Parking Name	Lot9
Status	Permitted
Arriving Time	10:00 AM
Expiration Time	10:00 PM

System will alert the user by a message when the current parking ticket is about to expire (we set the gap time to 15 mins before expiration). The gap time could be customize by users but for now we stay with a fix gap time for all the users.

10. Future works:

The designed app in this project can be expanded in many possible ways. We can provide the users with more information which would help them park their car in campus parking lots easier. For example suppose we could guess the number of cars parked in a specific parking lot. Then we could lead the users to the parking lots which are less crowded so that they can park their cars faster and they would not spend lots of time looking for empty parking spots. If we had cameras installed on parking lot doors this could be done exactly. However, as we discussed in previous sections installing cameras is impractical. Another way to guess how many cars are parked in a specific parking lot is to estimate this number by assuming that people using our application are a random sample of the whole campus population. This does not pose any costs for implementation and can be easily added to our application. However, this approach is inexact and requires many users to use our application in order to be able to estimate the number of cars parked in a parking lot within a reasonable range.

Another improvement to our application is realizing that the user has parked her car without requiring the user to push a button. Many times user is in a hurry and might forget to push the button in application to inform the system that she has parked her car. Therefore, we can add the ability to realize the user has parked her car to our system. This can be done in many different ways. One is action recognition. If the user's state has changed from sitting to walking or standing and she is in a parking lot the system realizes that the user might have parked her car in this parking lot and can notify the user to move her car from this specific parking lot later if she has parked her car in it. Another way to realize that a user has parked her car in a specific parking lot is by learning. For example consider a user who parks her car every Monday at 10 a.m in parking lot 2 and leaves at 6 p.m. On a Monday when the parking permit is expired at 3 p.m the user is informed about this event by the system even if she has not pushed the button since the system can guess that her car is probably parked in parking lot 2.

As we mentioned earlier our system can inform the user of the parking lot she has parked her car in upon her request. Many times the user might remember the parking lot number but might not remember the exact location that she has parked her car. The system can inform the user of the parking location using GPS system. However, the GPS system cannot realize the exact location of the car. This is not a significant problem since we can assume that the user can find her car if we give her an estimation of the location of her car.