Implementation Document

Team 408: Duoc Nguyen, Huy Nguyen, Robin Goh

Preface

There are always circumstances when we suddenly need a parking spot but none are available or are available at very high prices. On the other hand, many people have open parking spaces which are left unused. Our application platform's purpose is to provide users ways to share those resources with each other conveniently and hassle-free. Users can put up their own parking spaces up for renting, reserve parking spaces from other users, and communicate with each other through messaging. All the negotiations are discussed and processed by the hosts and seekers themselves; the ParkHere platform only acts as the middle-man to make sure the process finishes smoothly and both parties are happy at the end.

In this implementation document we will go over our implementation design and specify any changes that we made to our project design in terms of architectural and detailed design change. Then we will consider a requirement change from our client and discuss how our implementation may or may not change.

Introduction

ParkHere is an application which provides its users a platform to share their private parking spaces with each other throughout the United States. In many cities, parkings might be extremely hard to find or can be costly. The goal of this application is to help users share parking resources with each other at a considerably lower price than available public options. The ParkHere platform will gain profit through commission fees based on a certain percentage of the total transaction fee. We will embed our system in an Android application for user interface, and use Firebase for storage and back-end operations.

On architectural change, we will be enlisting the architectural changes we made during our implementation as well as explanation on why and how we made the changes.

On Detailed design change, we will list and explain any detail design changes during our implementation process ranging from UI design to backend data structure.

On requirement change, we will consider a client requirement change and discuss how we can accommodate the changes with details on how it will change our design.

Implementation Document

Team 408: Duoc Nguyen, Huy Nguyen, Robin Goh

Architectural Change

We still use the MVC model for our app but have added several classes for the View part. For example, we added DetailParkingActivity class to have a View to show more detail regarding a parking space which available to be booked.

In addition, we also added SignInActivity and SignUpActivity classes in order for users to have a UI to sign up with a new account and sign in with his/her previously-created. We used Firebase Email and Password Authentication for account authentication.

Detailed Design Change

We initially planned to design our app as a single page application, which would be more responsive and provide better user experience. An example would be the Home page, where users are able to enter a location and/or date of a parking they are looking for. Upon the press of the search button, the results would appear right below.
However, we weren't able to learn the necessary skills to implement it yet. So for now, we will stick to a more approachable method, where separate activities will be created and directed to as users navigate throughout the application, mainly with the press of the buttons.

Once we have all the functionalities working properly, we will return back to improving the user experience design discussed in above.

Requirement Change

Imagine now your ParkHere system should support "split" booking. For example, the availability of a parking spot is 10/01/2017 -10/08/2017. A seeker can book this parking spot from 10/02/2017 to 10/05/2017, and another seeker can book the same parking spot from 10/05/2017 to 10/07/2017. Similarly, "hour" booking can be split as well. Please answer the following questions.
1.Does this change your design?
2.If not, why not?
3.If so, what should be changed and how?

First, we considered owners wanting to put a parking up for booking for a range of time and we implement it in a way that our database would be able to store it. For example, if owners want to list their parking space from 10/01/2017 - 10/08/2017. Our database would split the time duration into individual date. The reason behind doing this is we want to be able to retrieve

parking space for specific date fast rather than having to traverse through the database making the search time an O(n) time complexity. So when a seeker reserve a parking for just one day, 10/02/2017 for example, seekers will be able to do that and since we split the listing already, someone else will be able to search and book the other dates. However, we have not consider searching for a parking for a range of time like from 10/02/2017 - 10/04/2017. Searching for three days like the requirement change. Right now if a seeker wants to book a parking for three days, they would have to make 3 searches and would be able to book 3 days in a row if there's an available parking space.

After considering this requirement change, our team decided to keep the way we store listing in our database, but our design will now implement searching for a range of of time rather than just one day.

Now in terms of searching for an available hour, as a team we did not take that into consideration since if a seeker want to search for a parking on that specific hour, they can still do it by searching if there are any parking available for that day and it will show all the available parkings the day with available hours. We will verify with our clients and if our clients want to be able to search for specific hours, then our team will accommodate the change. However, the design in how we store these listings will not change dramatically. The only changes will be adding a child node for time. Then, we can implement a function to split the booking hours by first retrieving that parking space from database, then at seeker's booking request, we will split the parking and update our database.