ParkHere

Team: O(k)
Team Members:
Sinjin Baglin (SJSUID:009032153)
Jonnel Alcantara (SJSUID:009101339)
Andrew Hon (SJSUID:009115678)
Daniel Vu (SJSUID:009254791)

Table of Contents

Preface, Introduction, Architectural Design	pg 3 - 4
Detailed Design	pg 4 - 5
Split Booking	pg 5

1. Preface

   Version 1.0: Created Requirement Docs 9/21/17
   Version 1.1: First Revision 9/24/17
   Version 1.2: Final Revision 9/26/17
   We created a requirement Document for the ParkHere Application to describe the needed functionality of the project.
   Version 2.0: Created Design Diagram Document 10/2/17
   Version 2.1: Final Revision of Design Diagram Document 10/8/17
   We created the design document for the ParkHere application which describes the application for developers.
   Version 3.0: Implementation Document 11/2/17
   Version 3.1: Final Revision of Design Diagram Document 11/5/17
   We created the implementation document for the ParkHere application to describe any implementations that occurred during the implementation process of the application.

2. Introduction

   This version of our ParkHere application contains implementations of all the core features. We have included an authentication feature and profiles for users to register accounts. Through these accounts, users will be able to search for and post listings. In this document we will go over our implementation of these features and the changes we made that deviated from the original design document.

3. Architectural Change
   a. For the most part, our architectural design remained the same. The change we did end up making during our implementation was to get rid of the SeekUser and HostUser classes and just have one user class. Other than that, our implementation consisted of all parts outlined in our layered architectural design, with the exception of the utilities layer and the reviews application service. Our application first greets users with a login and registration page. Upon creating an account, the user has access to the application services. Users are able to post parking spot listings. These listings include a location, price, times, and dates. A user can have multiple listings, which is stored under their unique user id in our Firebase database. These listings can be removed by the owner if necessary. Users can query other open parking spots through a search of parking spot listings. After a user has found a listing, the user can book that parking spot.
   b. The rationale behind the change to our Users class was that we felt it was redundant to have two separate classes for our users. We wanted users to be able to both create parking spot listings and book parking spot listings, so we got rid of both HostUser and SeekUser to have just one User class that can do both.
4. Detailed Design Change
   a. Our implementation differs greatly in how we organized our files in our design diagram. We decided not to separate the files into the smaller subcategories configuration, application and utility. We do not have transaction management, parking spot reviews, or user profile yet as we felt that it was not core

functionality. The sequence diagrams outlined in the design document remain accurate to our implementation, with the exception of the unimplemented aspects.

b. We found that the dependencies in the diagram does not accurately reflect our implementation. We decided to exclude transaction management, parking spot reviews, or user profile because we wanted to focus on our core functionality. This includes the functionality of creating a parking spot listing, searching for a parking spot, booking parking spots, and deleting parking spot listings. These four functionalities drive the application . Our diagram consisted of a three tiered architecture, which is a representation of how are program should be structured. We still followed the overall structure, but not perfectly.

5. Requirement Change: Spilt Booking
    a. The addition of the functionality of spilt booking would require us to change our design for ParkHere.
    b. As of right now, our design requires us to push a Parking Spot object with its attributes to the Firebase Database. When a user attempts to book a spot, they would fully take that time slot for the spot. With our design, we also need to change the way we are modeling parking spots. This new model would require a split method. Where the model will be split based on time and date. Another way to implement this would to be to create a new Model called Split booking which represents a modified parking spot model that is able to split both time allotted and dated allotted among multiple users. With the addition of split booking, the user would be prompted to a new activity where they can choose the times and dates out of the parking spot availability. After the times and dates have been chosen, that same parking spot would need to be pushed as a new listing to the database with the remaining time. This new listing would populate a new “table” in the database as we are using Firebase