Rubber Ducky Parking App

Requirements Definition

Group 6

Peyton Kiel, Bryson Meiling, Andrew Rasmussen, Brandon Garrett

February 12, 2021

Summary

The purpose of this project is to create a parking reservation system that allows *Users* to rent out parking spots and *Hosts* to create income by renting out their parking spots. Similar to *AirBnB*, parking spots are a commodity that are needed for certain times, for example, during sporting events or business conferences. Rubber Ducky Parking allows the *user* to find available parking on their phone and rent out a spot for a specified amount of time. A specific parking spot is then reserved, removing the stress of driving from one lot to the next. The App also allows a Host to rent out parking spots of various sizes (standard, motorcycle, RV, tailgating, etc) from a normal parking lot, dirt lots, or even their own driveway. The Host will love the features that allow them to utilize their empty parking space for profit!

Team Organization

The members are:

1. Peyton
2. Bryson
3. Brandon
4. Andrew

We all have experience with Git, Web Development including DJango and Vue.js, programming in Python, Java, and JavaScript, and group projects through our time at Uni. We use a Agile based software development strategy that will allow us to focus on coding and minimizing bulky documentation.

Communication

Our plan is to use GitHub to host the repository and Slack to communicate. We anticipate meeting virtually at least once a week while also completing assigned tasks during the week.

Risk Analysis

This project is never anticipated to be used in real world situation in this current version. One large factor that will hold this project in the realm of development is the current situation of money exchange.

An in app option to take credit cards and other payment would be the best, but suffice it to say, this is outside of the current scope of work. Therefore, risk is minimized greatly.
The remaining risk of the project will be evaluated at three distinct times:

1. During requirement gathering and system design
2. After requirement definition and before development
3. During testing and deployment

The inherent bug that occur in software development will be flushed out that the core operation of the system will function. In addition to common software bugs, delayed development, and underestimation of cost for developers (oh, wait...), we anticipate the following risk with this project:

1. User's transaction failed to log, resulting in a parking spot to being reserved although the User was under that assumption
2. Overbooking of parking lots due to system and/or Host errors
3. Empty parking lots because the search engine of the system overlooks a parking lot
4. General Frustration for using the app

Setup

See the *README.md* file in the project source code for configuration management and setup

1 Introduction and Context

Parking is always frustrating as it is time-consuming. This problem multiplies when attending large sporting events where everyone is trying to park as close at they can to a common location, and traditional parking lots are too expensive but getting a parking ticket costs even more. For those that live close to popular areas around the event, they are constantly fighting to maintain their parking areas, whether that be the driveway entrance or the street in front of their house.

Figure 1. Crowded Event Parking

This document describes the user goals and requirements for a software application and website, called Rubber Ducky Parking (RDP), that aims to help users rent unused parking space. The target users of Rubber Ducky Parking include event attendees, parking lot owners, and household owners close to events. Users will use the app to find available parking space that is close to the desired location while still be competitively priced. Once the user find a suitable parking location based on their vehicle size, duration, and needs, they will buy the parking spot and receive a confirmation with a QR code. The host will use Rubber Ducky Parking to rent out their available parking spaces, whether that be a whole parking lot, an unused driveway, or their front yard. The host will receive confirmation of purchase, a description of the car and a way to verify a QR code that is given to the buyer.
Section 2 describes the user and their goals in more detail. A suggested overall organization of the user interface for these features is described in Section 3. Section 4 describes functional requirements for a proposed set of software features that will satisfy these user goals. The software will be built using an incremental development process, constrained by the non-functional requirements enumerated in Section 5.
Note that RDP is intended to be a lightweight tool that users of all level of computer skills should be able use. Furthermore, its features are intentional focused on the highest priority user goals and limited to those that can be completed in three months. Interesting, potential features that cannot be included in the first version are listed in Section 7.
... A description of the overall software development process (provided by the instructor) Policies, procedures, or tools for communication, including plans for team meetings, online-coordination, reporting, etc. Risk analysis reference to the README.md for the configuration management plan

2 Users and Their Goals

The UML Use Case Diagrams in Figures 2-6 describe the key actors and user goals for Rubber Ducky Parking. The actors are color coded by major area. Things shown in gray are secondary goals that do not have to be satisfied by the first version of the software.

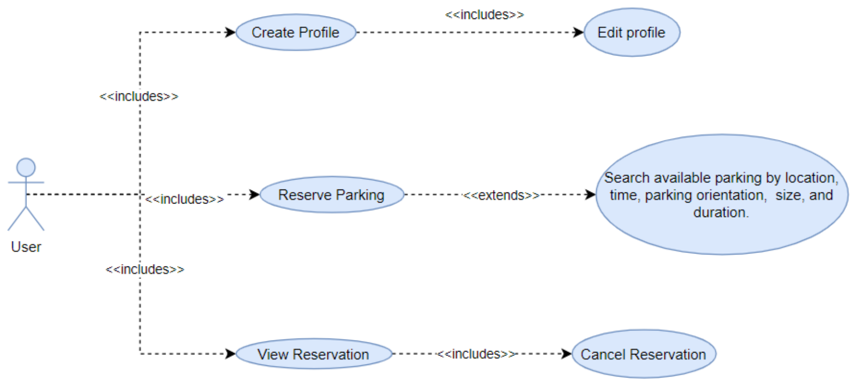


Figure 2. User Goals

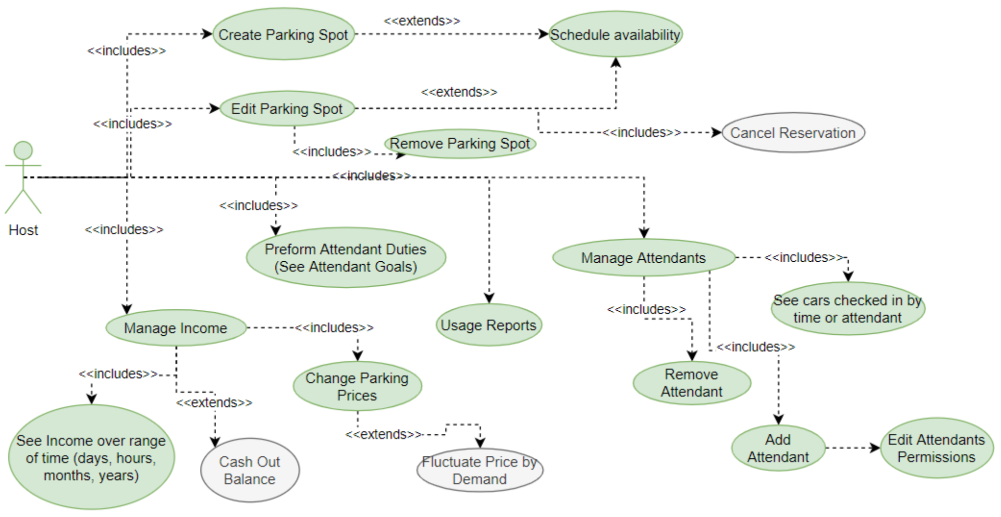


Figure 3. Host Goals

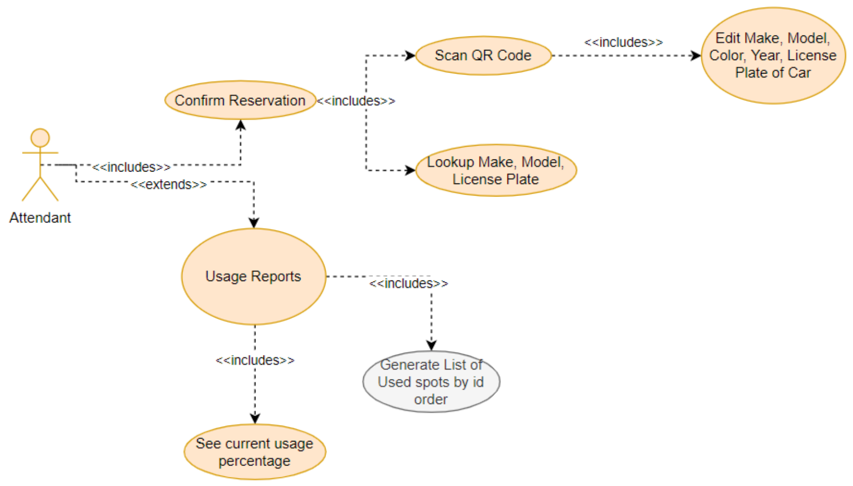
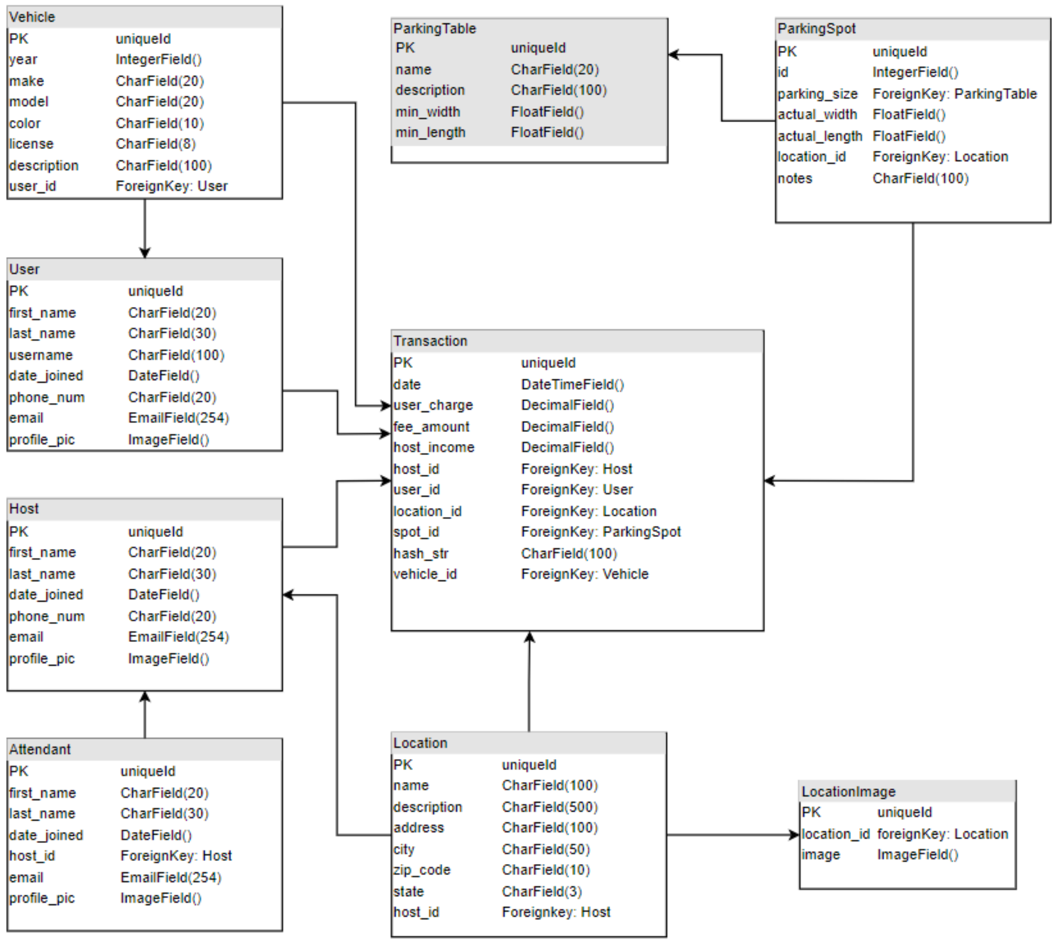

Scan QR Code
<<includes>>
Edit Make, Model, Color, Year, License Plate of Car
Confirm Reservation
<<includes>>
Lookup Make, Model, License Plate
<<includes>>
<<extends>>
Attendant
Usage Reports
<<includes>>
Generate List of Used spots by id order
<<includes>>
See current usage percentage

Figure 4. Attendant Goals

Table 1. The anticipated skill and attitude of the agents using Rubber Ducky Parking

Actor	Expected Skill Level	Anticipated Attitude
User	Basic Internet Skills (navigating to a website, creating a profile, completing a transaction form	Apathetic towards learning a new system just to park, curious if this system will help them with a semi-frustrating problem in their live that occasionally shows up.
Host	Basic Internet Skills	Excited to utilize available space. Hopeful that this system makes his/her empty space profitable.
Attendant	Basic Internet Skills, knows how to scan a QR code	Maybe a bit annoyed that they have to validate every car. Just using it for their job since they didn't buy $GME or $AMC

3 Classes of Objects and the Relationships



Vehicle	
PK	uniqueId
year	IntegerField()
make	CharField(20)
model	CharField(20)
color	CharField(10)
license	CharField(8)
description	CharField(100)
user_id	ForeignKey: User

ParkingTable	
PK	uniqueId
name	CharField(20)
description	CharField(100)
min_width	FloatField()
min_length	FloatField()

ParkingSpot	
PK	uniqueId
id	IntegerField()
parking_size	ForeignKey: ParkingTable
actual_width	FloatField()
actual_length	FloatField()
location_id	ForeignKey: Location
notes	CharField(100)

User	
PK	uniqueId
first_name	CharField(20)
last_name	CharField(30)
username	CharField(100)
date_joined	DateField()
phone_num	CharField(20)
email	EmailField(254)
profile_pic	ImageField()

Transaction	
PK	uniqueId
date	DateTimeField()
user_charge	DecimalField()
fee_amount	DecimalField()
host_income	DecimalField()
host_id	ForeignKey: Host
user_id	ForeignKey: User
location_id	ForeignKey: Location
spot_id	ForeignKey: ParkingSpot
hash_str	CharField(100)
vehicle_id	ForeignKey: Vehicle

Host	
PK	uniqueId
first_name	CharField(20)
last_name	CharField(30)
date_joined	DateField()
phone_num	CharField(20)
email	EmailField(254)
profile_pic	ImageField()

Attendant	
PK	uniqueId
first_name	CharField(20)
last_name	CharField(30)
date_joined	DateField()
host_id	ForeignKey: Host
email	EmailField(254)
profile_pic	ImageField()

Location	
PK	uniqueId
name	CharField(100)
description	CharField(500)
address	CharField(100)
city	CharField(50)
zip_code	CharField(10)
state	CharField(3)
host_id	Foreignkey: Host

LocationImage	
PK	uniqueId
location_id	foreignKey: Location
image	ImageField()

Figure 2. Database Tables

4 Functional Requirements

Any Requirement Which Specifies What The System Should Do. There is a strong cause and effect relationship present.

1 **User**

1.1 MUST

1.1.1 The system must send a QR code to the user through text or email once a transaction is finalized

1.1.2 The system must allow the user to cancel a reservation.

1.2 SHOULD

1.2.1 The system should send a receipt to the user through text or email once a transaction is finalized

1.3 COULD

1.3.1 The user could receive texts or emails when their reservation time is about to begin and end.

1.3.2 The user could be allowed to 'fav' parking spots for quick access later.

1.3.3 The User could report an issue (i.e. parking size rented doesn't reflect real size, no available spots, etc)

1.3.4 The System could let the user know if it think the car wont fit in the parking spot that they are choosing, but could be overridden.

1.4 WON'T

1.4.1 The system won't stop the User from canceling at anytime.

1.4.2 The system won't charge a cancellation fee

2 **Host and Attendants**

2.1 MUST

2.1.1 The system must create an integer ID with each parking spot.

2.1.2 The Host must be able to do everything an attendant can do.

2.1.3 The host must not be allowed to over schedule a parking spot.

2.2 SHOULD

2.2.1 The system should show the make, model, color, and license plate of the car under the reservation.

2.3 COULD

2.3.1 The Host/ Attendant could receive a text or email when a car's reservation time has ended.

2.3.2 The Host could cancel a reservation before a user arrives and refund the user.

2.4 WON'T

2.4.1

3 **System**

3.1 MUST

3.1.1 The System must not allow overbooking of a parking spot

3.2 SHOULD

3.2.1 The System should account for Users over staying on their time and not allow more reservations on a spot until the User is confirmed to be gone.

3.3 COULD

3.3.1 The System could have an option to allow Users to change measurements from feet to meters.

3.4 WON'T

3.4.1 The system won't do this when that occurs

host (owner) (owner can have 0 to many attendants) attendant (works for the lot)

5 Non-functional Requirements

Or when the person reserves a parking spot it could add their name to a list that the attendant has. Then if the attendant scans the QR code it marks off the name or they can manually mark off the name of the customer.

4 **User**

4.1 MUST

4.1.1 The system must use DJango as a web server and as an API service

4.1.2

4.2 SHOULD

4.2.1 The system should send a receipt to the user through text or email once a transaction is finalized

4.2.2 The User should be able to leave a rating for the location and/or the attendants.

4.2.3 The User should enter a password to enter into their account.

4.3 COULD

4.3.1 The user could receive texts or emails when their reservation time is about to begin and end.

4.3.2

4.4 WON'T

4.4.1 The system won't stop the User from canceling at anytime.

4.4.2 The system won't charge a cancellation fee

5 **Host and Attendants**

5.1 MUST

5.1.1 The system must create an integer ID with each parking spot.

5.1.2 The Host must be able to do everything an attendant can do.

5.2 SHOULD

5.2.1 The system should do this when that occurs

5.2.2 The Host/Attendant should enter a password to enter into their account.

5.3 COULD

5.3.1 The Host/ Attendant could receive a text or email when a car's reservation time has ended.

5.3.2 The attendant/ Host could be asked to reply to a text with a random code to confirm that the phone they are using.

5.4 WON'T

5.4.1 There won't be a fee

6 **System**

6.1 MUST

6.1.1 The system must do this when that occurs

6.2 SHOULD

6.2.1 The system should do this when that occurs

6.3 COULD

6.3.1 The system could do this when that occurs

6.4 WON’T

6.4.1 The system won’t do this when that occurs

6 User Interface Organization

user(buyer) Example use case diagrams we can make 1.person orders parking lot 2. attendant receives customer 3. host (owner) creates parking spot 4. host creates multiple spots on a cycle 5. host want to check money in account 6. host wants to see number of unsold spots for a given day 7. user wants a refund 8. user needs to find their QR code again (happens to me all the time) 9. attendant want to see make and models of cars that are supposed to be in their lot at that time (generates a list) 10. After searching and receiving a list of possible parking lots, clicking to know more about a parking spot opens in a new tab
allow for browsing without signing it, but have to have an account to reserve spot
user must add at least on car
save cars for user to they can add a new one or quick add.
allow users to change the car on their reservation (must match size)

Table 2. Each screen and its purpose on the Rubber Ducky Parking website

Screen	Purpose/ Content
Login Screen	Allow a user to login with a username and password. Contains branding and welcome devices. See Req. Def. 1.1.
Main Screen	Allows the user to navigate to the three main areas of functionality. Contains menu and maybe other kinds of context and navigational aids

7 Future Features

8 Glossary

credits