Rubber Ducky Parking App

Requirements Definition

Group 6

Peyton Kiel, Bryson Meiling, Andrew Rasmussen, Brandon Garrett

April 14, 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. Andrew
2. Peyton
3. Bryson
4. Brandon

We all have experience with Git, Web Development including DJango and React.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. We also use the sprint planning papers and the retroactive spring planning papers that are provided in the class to track progress. Additionally, we take advantage of the tools on Github such as the project KanBan Boards, and Issues board to keep track of what things are left to do and what things we need to fix.

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 wages 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.

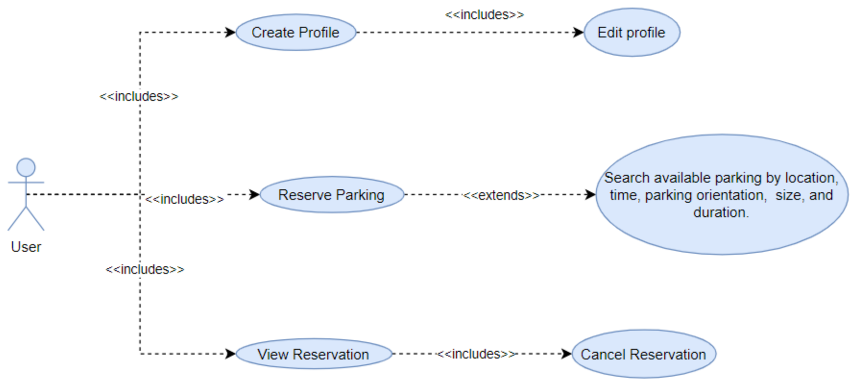


Figure 2. User Goals

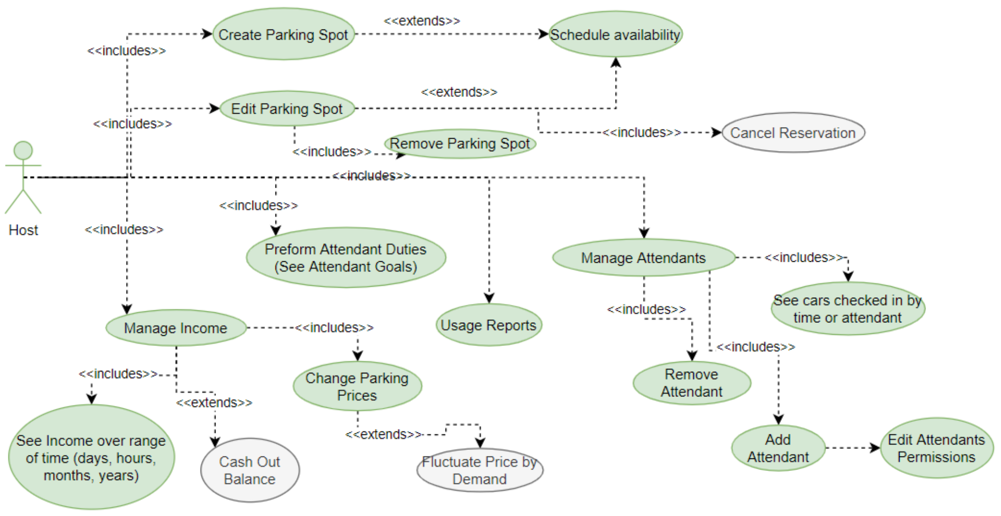


Figure 3. Host Goals

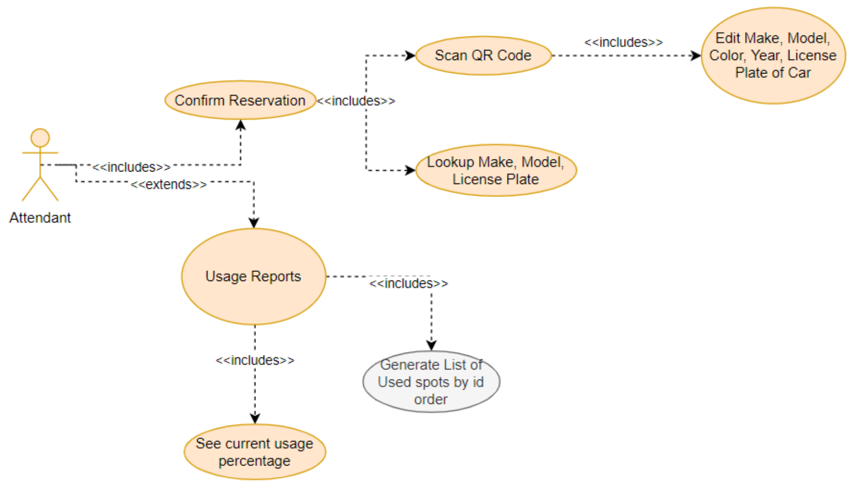
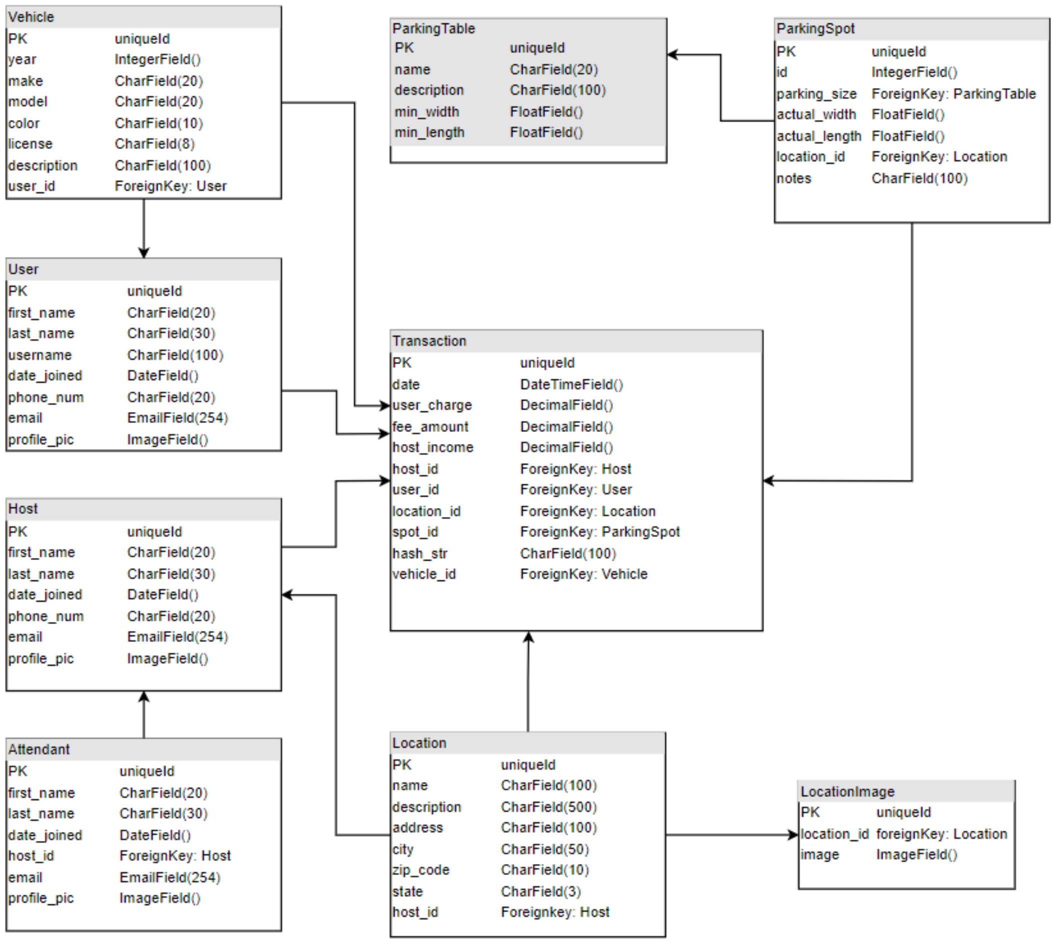

Scan QR Code
<<includes>>
Edit Make, Model, Color, Year, License Plate of Car
Confirm Reservation
<<includes>>
<<includes>>
<<extends>>
Lookup Make, Model, License Plate
Attendant
Usage Reports
<<includes>>
<<includes>>
Generate List of Used spots by id order
See current usage percentage

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 show a QR code that is access able through the /reservation/id/ url.

1.2 SHOULD

1.2.1 The system show the user's reservation in their profile

1.3 COULD

1.3.1 The user could be allowed to 'fav' parking spots which would show a red heart when searching for parking spots

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 implement a scheduler that stops overbooking of a spot

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.

2.2 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 could see the incoming reservations like an attendant

2.4 WON'T

2.4.1 The System won't enforce any rules about scheduling due to the short time span of development

2.4.2 The Host won't control who is a host for them

3 **System**

3.1 MUST

3.1.1 The system must work when the web server and API is running

3.2 SHOULD

3.2.1 The System should create a OToken to authenticate users

3.2.2 The system should have defined security roles for users, hosts, attendants.

3.2.3 The System should allow overbooking

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 implement a company account that takes a cut of the payment

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 an API service and React.JS as a web server.

4.2 SHOULD

4.2.1 The system should not send a receipt

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

4.2.3 The User log in page could use REACT.js to make sure that all the spots of the form are filled out

4.3 COULD

4.3.1 The system could stop back form entry with react and Django input checking

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

4.4.2 The system won't communicate with the users by text or email

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

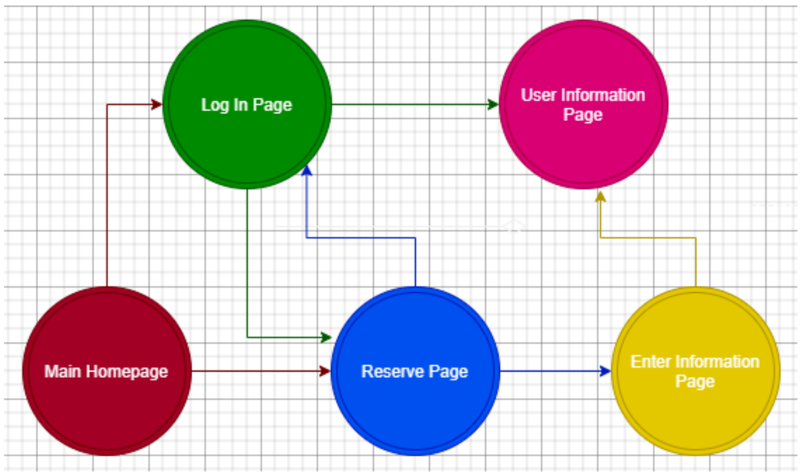
5.3.1 The attendant/ Host could be asked to reply to a text with a random code to confirm that the phone they are using.

5.4 WON'T

5.4.1 There won't be a fee

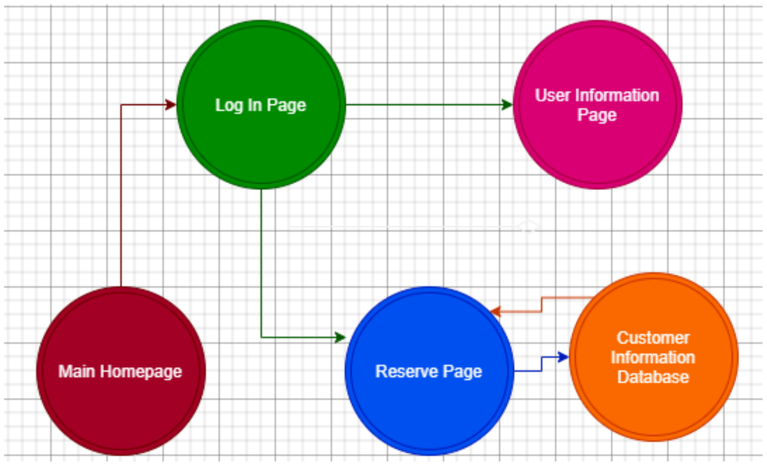
6 User Interface Organization

The User Interface will have to be able to function and handle multiple different users will different permissions.



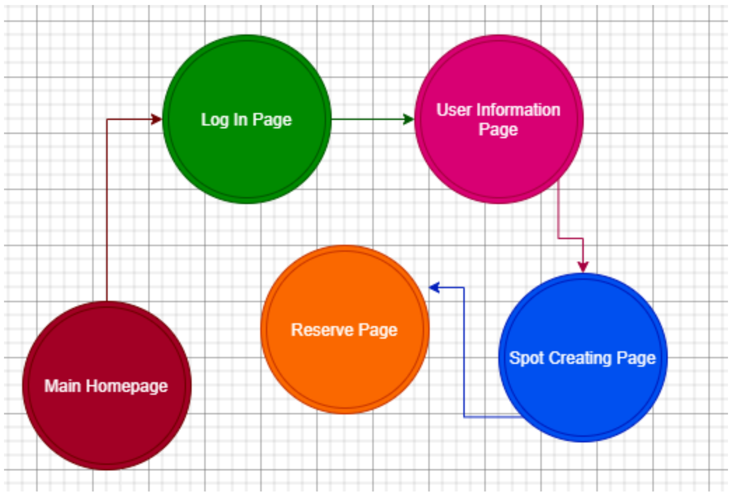
Use Case 1. User wants to reserve a spot

The User will be able to head to the main homepage of the application. They are then able to log in and then head to the reservation page to reserve their spot. They can also head directly to the reservation page to view the spots available, and after clicking on one to reserve, will be taken to the login page. After the user is logged in and the spot is chosen, the User will be prompted to enter their information, such as car make and model, name, phone number, and payment information. After entering information they can view their reservations or change input data in the Customer Information tab.



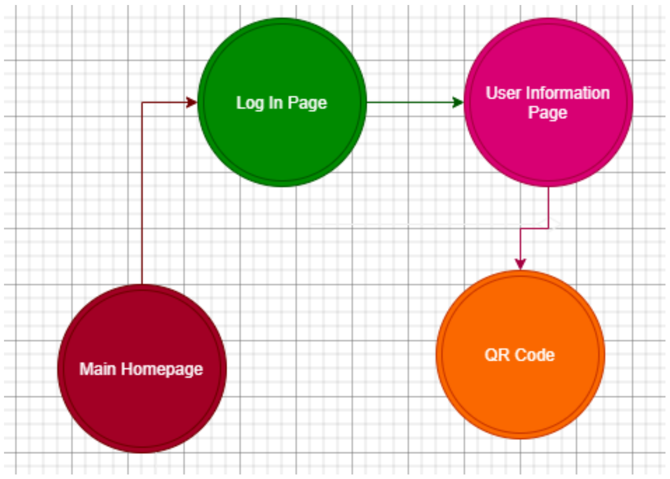
Use Case 2. Attendant receives and directs a customer

The Attendant will be able to head to the application homepage and to the log in page. The log in page will then all the Attendant to access the Attendant Dashboard page with elevated credentials. These credentials will allow the Attendant to see the information of each car at each spot as well as allow the Attendant to scan the customer's QR code an direct them to the correct location of their parking spot.



Use Case 3. Host wants to create another parking spot for use

The Host of the parking spots will be able to go to the main page and then the log in page. From here after logging in, the Host can access the Profile page where they can register as a Host if not already. If registered they can go to the /mangehost Page and add one or multiple locations and parking spots to host. The spots will be linked to the Host's account.



Use Case 4. User forgot QR code and needs to pull it up again

The User will be able to go to the main page and log in with their credentials. They will then be able to go to the Profile page and here see all their reserved spots. Clicking on a spot will pull up the QR code. After generating the QR code they can screenshot or email themselves a copy of the code.

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.
Sign In Screen	Lets the user create a new profile and with a name, username, and password
Home Screen	Allows the user to navigate to the main areas of functionality. Contains a search bar for finding parking. Contains the main search bar and links to all the rest of the pages
Profile Screen	Allows the user to navigate to and view their personal information and reserved spots, register as a host, if already a host can view their hosted spots and attendants
Host Dashboard Screen	Has the host's parking spots and locations. They can also add to these spots and locations
Attendant Dashboard Screen	Has a list of user's that are coming to the parking lot so that the attendant can see their upcoming reservations. Useful for knowing what cars to expect.
About Screen	Basic information that anyone can see whether they are logged in or not
Search Screen	Displays all the spots available based on what the user entered in the search bar at the main screen.
Details Screen	Displays all the information about a single spot and has a button to rent that spot for a desired amount of time
Reservation Screen	Displays all the information about a single spot, when the reservation time is, and QR code for the attendant to scan

7 Future Features

- **Scheduling** - Restricts the renting of parking spot so it cannot be double booked.
- **Better Search Function** - The search function is more capable of giving what the user wants and other options in the area. This function would work in tangent with google maps for close locations an the database parking spot scheduler.
- **Weather Function** - Allowing the user to see the predicted weather of the day they're planning on scheduling for
- **Pop Up Hovering** - When hovering over specific things (parking spots, host spots) information will be displayed about it

8 Glossary

Attendant - The user who will be the one present at the event who will be directing the cars to their spots as well as checking to make sure there is no unauthorized parking.

Customer - The user who will be using the application to rent a parking spot for an event.

Django - The web server Rubber Ducky Parking will be hosted on.

Host - The user who will host parking spots for Customers to rent out. They can be anyone as long as they have a spot to host for events.

Page, Login - The page that the user will be able to login to. Based on their permissions they will be able to do different functions.

Page, Main - The page that all users will go to initially and will serve as the homepage for the application.

Page, Reservation - The page that will display the parking spots that are available on the respective event date.

Page, User Info - The page that any of the users will be able to go to and will display their personal information; for the hosts, the spots they're hosting will also be displayed.

QR Code - Quick Response Code, a distinct pattern that will be able to be scanned and show where a spot is at.

Spot, Large - A parking spot that will allow the parking of RVs, trailers, or other vehicles of similar size.

Spot, Medium - A parking spot that will allow the parking of normal sized cars or trucks.

Spot, Small - A parking spot that will allow the parking of smaller vehicles such as motorcycles or mopeds.

User - The User is anyone who uses the application.