Software Design and Development

Requirements Gathering

October 18th, 2023

Valet Buddy

Airport Valet Car Locator Mobile App

VALET BUDDY

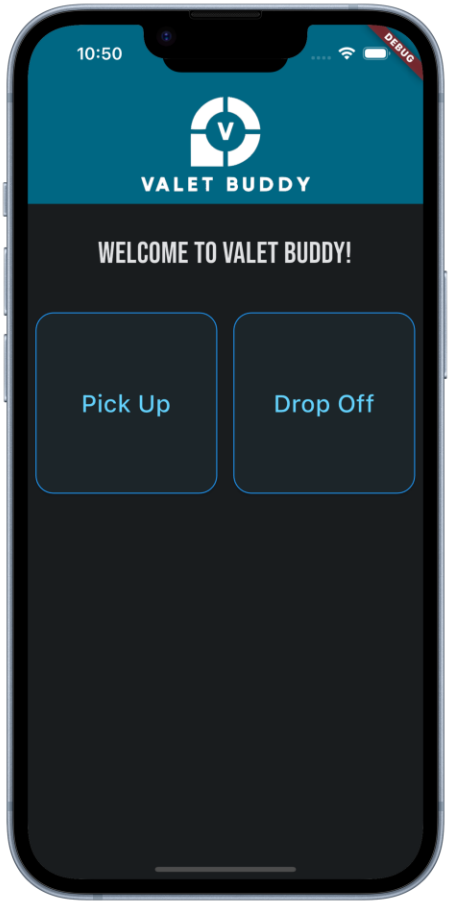
Project Team

Our team is Team 1 comprised of the following students:

- **Jordan Treutel**
- **Richard Hoehn**
  - **Ian Hurd**
- **Patrick Burnett**

Scope Statement

Describe the product of your project in broad terms.



The project is a mobile app for iOS and Android that allows a Traveler (car owner) to drop off their car with a Valet (driver) that parks the car and sets a GPS location for the parked car based on their phone's current GPS location. In addition to the car's GPS location the associated license plate number, name of the car owner, and picture of the car are also captured during the "Drop-Off" phase.

Another user (Pick-Up-User) can then retrieve the location of the car by entering the license plate number. The storing and retrieval of the car is facilitated by a server that accesses a database for getting and setting GPS coordinates, the license plate and image for later retrieval.

With this app (Valet Buddy), GPS can be used to "Set" and "Get" the car's location and displayed on a user's mobile phone.

The business case of this app is simple... Using technology (specifically GPS) the Airport Car parking owner will not have to "update" and paint numbers on the parking spaces anymore, saving time, and money. Instead, they can rely on the "Valet Buddy" app and backend server to keep track of all the cars that are parked on their lot and quickly and efficiently retrieve them when a car owner returns from the travels at the airport.

Stakeholders/Users

Individuals or groups who have an interest, influence, or concern in a project and its outcomes.

Airport Administration

The airport administration oversees all the ancillary third-party services related to the airport's operations. In detail they would be interested in the Valet Buddy, because they care about the reputation of the airport travelers' experiences and since parking is part of a traveler's experience this requires their interest.

Airport Valets (Drivers) – Main Users of App

The drivers are the main users of the app. They take the cars, park and tag the locations of the car via the app's GPS coordinates. In addition, they also take a picture and set the license plate of the cars.

Airport Travelers (car owners)

The traveler is essentially the owner of the car that drops it off at the airport parking lot and takes a bus or simply walks to the airport. Upon return, they ask the valet driver to get their car based on their name and license plate.

Owner of the Airport Parking Lot (users of Valet Buddy)

The owner of the valet lot and users is essentially the company that purchased / commissioned the Valet Buddy.

Business & Financial staff at Valet Buddy-owning company

These users are support staff and technical support for the continued operations of the Valet Buddy app and server.

Functional Requirements

Specific features, capabilities, and behaviors that a system or product must have to satisfy the needs of its users and meet the project objectives.

- The ability for a user to input the car's license plate number
- User can upload a picture of their car to the app
- User can save GPS coordinates of a car's location
- There exists a list of cars, each entry storing a picture of the car, the car's license plate number, the name of the car's owner, and the car's location
- Users can access a list of cars and select one to retrieve its coordinates
- Users can only see cars from their own airport, unless they are an app administrator.
- Users can report damaged or stolen cars
- Each car is given a 'ticket' which tracks all data about its drop-off and pick-up.

Non-Functional Requirements

Aspects of a system of project that do not relate to specific behaviors or features but rather characterize how the system performs certain functions (performance, usability, security, scalability, etc.).

In Scope (class deliverables)

- JSON for data transfer
- Flutter (mobile framework)
- Google Maps API
- Dart (mobile programing language)
- Python
- Flask (REST)
- Database (TinyDB – Native Python)

Out of Scope (for project)

- All communications via "https"
- Backing up Database and Car Images to S3
- For demo, will store files on server.
- Monetization
- Weekly Map of all locations of the cars on map

- User management, won’t have separate database for valet user management

Input/Output Data

The following is a brief overview of the data that flows into the server and out of the server.

Input:

- License plate number
- Name of Traveler (car owner)
- Image of the car
- Phone geolocation data
- Ticket Status: Parking, Parked, Retrieving, Complete
- Notes (Text)

Output:

- List of cars in the parking lot (includes an image of the car and its location, as well as its license plate number and GPS coordinates)
- Current “Ticket Status” of a specific car
- JSON object of ticket details:
    - License Plate Number
    - URL for Image of Car
    - Name of Owner
    - Lat & Lng of car location
    - Created Date Time
    - Status of Ticket

Use Cases

Detailed description of how users interact with a software system to achieve a specific goal. Includes the steps a user takes, the interactions with the system, and the expected outcomes.

Valet Login to App

1. Valet open app
2. Valet enters their email address & password

Traveler (car owner) drives to the valet.

1. Valet “Starts” a Ticket
2. Traveler gives the valet driver their name.
3. The valet driver opens the app and sees the menu screen.
4. The driver chooses “Drop-Off”, which opens a new form screen containing text fields for “Traveler Name” and “License Plate Number”.
5. The driver fills out the form with the traveler’s name and their car license plate number and clicks a “submit” button. This saves a “ticket” in the TinyDB and sets the status to “Currently Parking”.

Valet driver parks the car.

1. The driver opens the app and sees the menu screen.
2. The driver chooses “Vehicle List”, which opens a list of current tickets in the database.
3. The driver then has 2 options:
4. They can scroll through the list of vehicles
5. Or they can use a search bar at the top of the screen to filter the vehicle list down based on the license plate number or the name of the traveler. This will shorten the list and display only those options that match the searched text.
6. The driver clicks on the list entry of the vehicle that they just parked.
7. The application displays the traveler’s name, the license plate number, the status of the vehicle, and a button to “Add Image & GPS Tag”.
8. The driver clicks on “Add Image & GPS Tag”, which allows the driver to take an image of the car and its surroundings in the parking lot. Doing this will also submit a GPS tag of the current location to the database and automatically mark the status of the ticket to “Parked”.

Valet driver needs to retrieve a car from the parking lot.

1. The driver opens the app and sees the menu screen.
2. The driver chooses “Vehicle List”, which opens a new screen that displays a list of vehicles currently parked in the lot.
3. The driver then has 2 options:
4. They can scroll through the list of vehicles
5. Or they can use a search bar at the top of the screen to filter the vehicle list down based on the license plate number or the name of the traveler. This will shorten the list and display only those options that match the searched text.
6. The driver clicks on the list entry of the vehicle they need to retrieve.
7. The application displays the traveler’s name, the license plate number, the status of the vehicle, an image of the vehicle, a map with a location marker, and a “Retrieve Car” button.
8. The driver travels to the location of the car. Upon entering the car, the driver will click “Retrieve Car”. The car’s ‘ticket’ status is set to “Retrieving”.

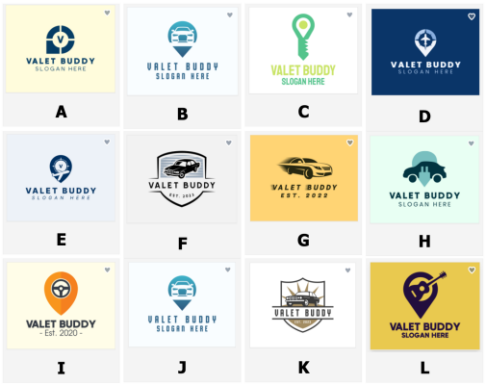
Valet Driver Returns Car to Traveler (Car Owner)

1. The driver opens the app and sees the menu screen.
2. The driver chooses “Vehicle List”, which opens a new screen that displays a list of vehicles currently parked in the lot.
3. The driver then has 2 options:
4. They can scroll through the list of vehicles
5. Or they can use a search bar at the top of the screen to filter the vehicle list down based on the license plate number or the name of the traveler. This will shorten the list and display only those options that match the searched text.
6. The driver clicks on the list entry of the vehicle they just delivered to the traveler.
7. The application displays the traveler’s name, the license plate number, the status of the vehicle, an image of the vehicle, and a “Close Ticket” button.
8. The driver clicks “Close Ticket”, and the ticket is cleared from the database. The car will no longer appear on the “Vehicle List” screen.

Meeting Minutes & Notes

The below are simple meeting minutes of the discussions we as a group had during the development of this "Requirement Gathering" document.

2023-10-09 – In Class Meeting:



We picked a logo during class. We got an "AI" to generate 6-8 different options. We decided to use "A" by voting and discussing what we like. Richard will take care of getting the logo created for use in this project. We then reviewed some details on the Flutter app that we will be using for the development of this project. And finally, we discussed when we would meet outside of class again. We decided that it would be best to meet after class on Wednesday for 30min to make sure all are on the same page.

2023-10-11 – In Class Meeting:

Discuss in detail the uses cases of the Valet Buddy by talking about who uses them. We came up with a bunch of them and started talking. About how the set-by-step interactions would follow for the different stakeholders. We came up with the drop-off, parking, parker, picking-up, and completed state machine that will be implemented.

We also spoke about each car parking is a "ticket" that holds the state of the parking and pickup process. We concluded by saying that we would meet on Friday morning at 9:00am to review progress and get more done.

2023-10-13 – Friday Morning Meeting:

Started zoom call with the team members at 9:00am. Made updates to Requirements Gathering documents.

We decided to add the "users" login to the app for authentication but will not be implementing the full user management system. It may be that we will use a simple "PIN" for user authentication and selection.

Overall, we feel this looks good and decided to each individually work more on this document and then submit it after class on Wednesday 18th of October. The PDF is not due until midnight that evening.