

# Team 34 – Product Backlog

*Charlie Schmidt, Leon Yee, Caleb Chua, Sriram Kiron*

## Problem Statement:

There are many different places to park on the Purdue campus, and there are multiple types of permits you can have. Permissions for different permits change based on the day and time, and even if all of these details are figured out, parking can still be hard to find. Lighthouse makes this process easy, allowing the user to input what kind of permit they have, if any, and shows on a map all the closest available parking to a destination. Other factors, such as parking availability and frequency of parking tickets, can be crowdsourced from other users as well.

## Background Information:

### Problem/Audience

Purdue has many different options for parking, and it can be complex. Our app aims to streamline the process of finding a parking spot and provide insight to the user about parking availability, where the most parking tickets are given out, and notifications about the parking status of their car. Students and faculty on campus can use the app to allow them to park quickly and safely, and avoid unnecessary parking tickets.

### Similar Platforms

The most similar app would be SpotHero, which is a very popular parking app that allows you to find, reserve, and pay for parking spots in cities all from the app. You can see the cheapest parking options closest to where you want to go. SpotHero is focused on large urban areas, and is less prevalent in smaller rural areas such as college campuses. The key focus of SpotHero is that they enable users to secure their parking space in advance, preventing any uncertainty of finding parking upon arrival.

### Limitations

While SpotHero works well in large cities, it is extremely lacking in places like college campuses. Lighthouse focuses on college campuses and integrates the available parking options with Purdue’s parking permits, catering directly towards students and faculty. SpotHero also doesn’t provide crowdsourced information, such as where the most parking tickets are given out. We hope that by using Lighthouse, students and professors, especially at universities such as Purdue, have a thorough and precise understanding of parking options on campus.

## The Functional Requirements:

1. As a user, I want to register for an account using email and password
2. As a user, I want to log into my account
3. As a user, I want to reset my password if I forget it
4. As a user, I want to be able to manage my account (email, password, etc.)
5. As a user, I want to update my profile with specific vehicle information (permit type, compact car, electric car, etc)
6. As a user, I want the app to ask if it can track my location and track my location if I allow it
7. As a user, I want to be able to view all the Purdue parking options as a list view
8. As a user, I want to be able to click into a list item and see the specific parking information
9. As a user, I want to see a map of my surroundings
10. As a user, I want the map to display available parking spots near my location
11. As a user, I want the map to display parking spots based on my permit type
12. As a user, I want the map to display parking spots based on the time of day
13. As a user, I want the map to display changes in parking availability given football and basketball game days
14. As a user, I want to see all the parking options on campus as icons on the map
15. As a user, having the icons on the map bunch together through a heatmap as I zoom out will be a better viewing experience
16. As a user, I want to be able to filter parking options and vehicle information, and only see parking options that fit my selected filter
17. As a user, clicking on a specific parking option on the map should reveal a modal popup
18. As a user, I want the popup to display parking garage information
19. As a user, I want the popup to display changes in parking garage information given football and basketball game days
20. As a user, I want to set my parking status
21. As a user, I should be able to see whether a street parking spot is taken or not, and how long it has been taken for
22. As a user, I want a separate settings page to edit my preferences for alerts
23. As a user, I want to receive alerts about parking time expiry
24. As a user, I want to receive alerts about weather-related advisories
25. As a user, I want to receive alerts about moving my car during game days
26. As a user, I want to be able to input my school schedule into the application
27. As a user, I want the app to generate places I should park given my schedule
28. As a user, I want the app to generate times I should park and/or move my car given my schedule
29. As a user, I want to select a destination on the interactive map and get the directions to my selected destination (on the app or through an external app)
30. As a user, I want to see all the closest parking spots to my selected destination
31. As a user, I want to see where parking tickets are frequently given out
32. As a user, if I get a parking ticket, I want the ability to input where and when it occurred
33. As a user, I want to see where there is more/less parking availability
34. As a user, I want to be able to input into the app if there is more/less parking availability
35. As a user, I want to be able to find parking outside of Purdue
36. As a user, I want the UI for this information to be almost the same as the UI for Purdue parking, except for information that can’t be found
37. As a maintainer, I want to automatically get this information through web scraping

## Non-Functional Requirements:

### Architecture and Performance

We plan for the application to be deployed on mobile phones, including both iOS and Android operating systems, but we will specifically target iOS devices for testing. To complete this task, we would use React Native, a software UI framework that allows cross-platform application deployments. The frontend and backend will be written in NodeJS, a JavaScript runtime environment that is compatible with React Native. The main benefit is that we can use a single codebase for both sides of development, making it faster and easier. It is also very scalable, allowing our application to handle a large number of users with ease.

The database we will be using to store user/login information and data is Firebase, as it is most efficient with storing information and authenticating users. Firebase has ample documentation for connecting it with React Native, which makes it an intriguing platform to use.

### Security

It’s important that all user information stays safe and is encrypted. This is achieved through Google’s Firebase platform, as it automatically encrypts data using HTTPS and isolates customer data. Luckily, Firebase is also invulnerable to SQL-injection attacks, as it is a noSQL database. In addition to establishing safe database rules, we'll be conducting thorough checks to ensure there are no unintentional data exposures due to misconfigured settings.

### Usability

We aim to create a very user-friendly design for our application, as it requires many essential elements to ensure good functionality. To begin, we need to prioritize clear pinpoints on a map with a simple menu layout, allowing users to change views without confusion. The user-interface should also be clean and visually attractive to prevent overwhelming users. We also want to prioritize a responsive layout, allowing the application to work on a multitude of devices, including tablets and phones of all sizes.

### Hosting/Deployment

To deploy our application, we first have to deploy the React Native app on Expo, a platform for generating the proper IPA and APK files. With these in mind, the respective application files can be uploaded to the App Store or Google Play Store. The backend database is linked along with Expo, which makes deploying even more seamlessly. If required, we will be able to deploy our code onto a GitHub repository, and utilize the GitHub actions to automatically deploy on Expo through a CI/CD pipeline.