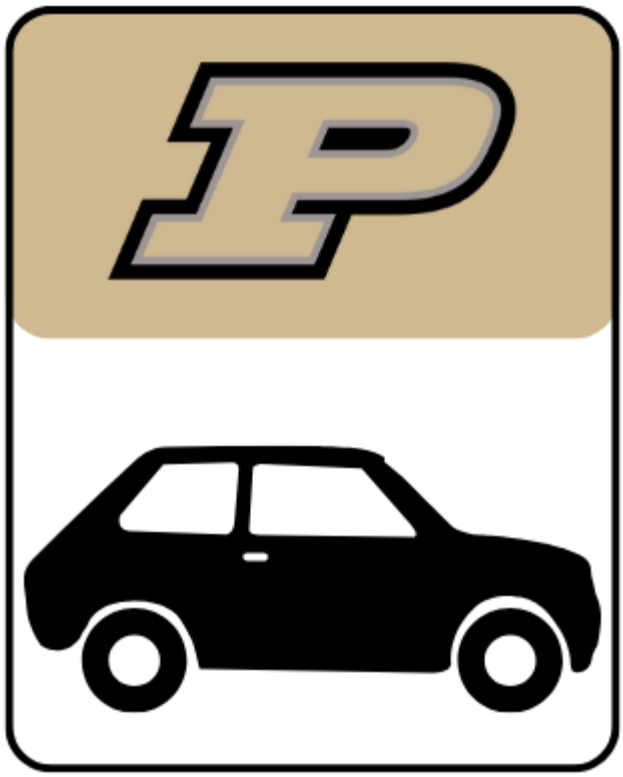

CS 307 Sprint 2 Planning - Group 6

BoilerPark

Pranay Nandkeolyar
Utkarsh Majithia
Anthony McCrovitz
Logan Portscheller
George Samra
Neel Vachhani

Sprint Overview:

This sprint focuses on adding new features that make BoilerPark more useful and informative for users. The team will work on showing detailed lot information like hours, rules, and walking times, adding confidence levels for data accuracy, showing which lots have covered parking, creating a calendar screen for upcoming events, and building a navigation view that shows ETAs and nearby garages. Both the backend and frontend will be updated to support these new screens and data connections. By the end of the sprint, users should be able to see clearer lot details, plan their routes more easily, and trust the information shown in the app.

Scrum Master: Utkarsh Majithia

Meeting Plan: Sunday @ 8pm Wednesday @ 9am (with project coordinator) & Friday @ 4:30pm - Change to Monday/Wednesday/Friday @ 10:30 am once lectures for class are done.

Risks and Challenges:
The biggest risks of this sprint are connecting all the new features without breaking existing ones and making sure the data between the backend and frontend stays consistent. External APIs for maps or ETAs may be slow or unreliable, which could affect performance. Testing will be more complicated since several features depend on each other. There's also a chance of merge conflicts or timing issues as different parts of the team work on related sections of the app.

Current Sprint Detail

User Story #1 - As a developer, I want to sync the camera to the CV model

Task #	Task Description	Estimated Time	Owner
1	Configure the camera to expose a video/data stream and verify local connection over HTTP	2 hours	Pranay
2	Set up a lightweight server (Flask, FastAPI, or MQTT broker) on the edge device	1 hour	Pranay
3	Connect the camera stream to the server and ensure data flow	1 hour	Pranay
4	Refactor the CV model for each garage.	3 hours	Pranay
5	Test and ensure that the Car Counter processes are independent and correctly update the Redis Cache.	3 hours	Pranay

Acceptance Criteria:

- Given that the camera is set up and connected to the CV model, when a car moves into the garage, then the Redis Cache is accurately increased for cars inside and the parking lot count for said garage is updated on the UI.
- Given that the camera is set up and connected to the CV model, when a car moves out of the garage, then the Redis Cache is accurately decreased for cars inside and the parking lot count for said garage is updated on the UI.
- Given that the camera is set up and connected to the CV model, when no car is moving in or out of the garage then the Redis Cache is stable and no updates are propagated to the frontend.

User Story #2 - As a user, I want to receive a notification when parking passes go on sale (Rollover from Sprint 1)

1	Add backend logic to notify active users when parking passes go on sale	3 hours	Anthony
2	Test the notifications are sent out	1 hour	Anthony

Acceptance Criteria:

- Given that I have granted push notification permissions to the app when I enable push notifications, then my device is registered in the backend and I am sent a notification to test.
- Given I disable push notifications in the app, when a sale notification is sent, then I do not receive the push notification.
- Given a parking pass sale is created in the system, when the backend sends the notification, then all opted-in users receive it.

- Given I have multiple devices linked to my account, when a sale notification is sent, then each active device receives exactly one notification.
- Given I have a device which is logged in and has push notifications enabled, the notifications are personalized and I am sent notifications with my name in them.

User Story #3 - As a user, I want to search for lots by name/code so I can find them quickly

Task #	Task Description	Estimated Time	Owner
1	Add the available parking lots to the database with the initial values	1 hour	Neel
2	Develop a Django route to query the parking lots based on various features, such as availability	2 hours	Neel
3	Test the backend route to see if it works	1 hour	Neel
4	Develop the Search bar on the Parking List Tab	2 hours	Neel
5	Develop the search pop ups on the Parking List tab	2 hours	Neel
6	Test that the Search UI correctly fetches the correct data for different parameters	2 hours	Neel

Acceptance Criteria:

- Given the user is on the Parking List Screen, when the user types in the name of a parking lot, then the correct lot should appear.
- Given the user searches based on availability, when they specify a minimum availability, then only lots with at least that availability should be displayed.
- Given the user picks a lot from the search list, when they select it, then the information for that lot should be loaded and shown.
- Given that the user is looking for a lot, when the user searches for open lots, then only open lots should be shown.
- Given that the user is looking for a lot, when the user searches for their favorite lots, then only their favorite lots should be shown.

User Story #4 - As a user I want to see a bar chart that shows how full a given lot is at the given time

Task #	Task Description	Estimated Time	Owner
1	Add an insights tab on the UI	2 hour	Pranay
2	Develop a Django route to query the parking lot availability	2 hours	Pranay

3	Test the backend route to see if it works	1 hour	Pranay
4	Develop the bar charts on the UI	2 hours	Pranay
5	Color the bar carts based on the availability	2 hours	Pranay
6	Test that the frontend queries the backend to show updated counts	1 hour	Pranay

Acceptance Criteria:

- Given the user is viewing the Insights tab on the UI, when the user selects or views a specific parking lot, then a bar chart should be displayed showing how full that lot is at the current time.
- Given the backend route is functioning correctly, when the UI requests parking lot availability data from the Django API, then the chart should dynamically update with the most recent availability data.
- Given the bar chart is displayed, when the availability level changes, then the color of the bars should update based on availability thresholds (e.g., green for available, yellow for moderate, red for full).
- Given the user views the Insights tab over time, when new data is fetched periodically, then the bar chart should automatically refresh to reflect real-time changes in lot occupancy.

User Story #5 - As a user, I want garages suggested to me based on my parking logs and the locations where I most frequently park, so that I can quickly find convenient parking options.

Task #	Task Description	Estimated Time	Owner
1	Create a PostgreSQL Table to connect users to the amount of parking lots they have been to	2 hours	Pranay
2	Develop a backend algorithm to analyze parking frequency and location patterns	2 hours	Pranay
3	Add a Django endpoint to fetch personalized garage recommendations	4 hours	Pranay
4	Implement a frontend UI section for “Suggested Garages” on the Parking List Screen	2 hours	Pranay
5	Integrate backend recommendations with the frontend display	1 hour	Pranay
6	Test that frequent parking patterns lead to accurate recommendations	1 hour	Pranay

Acceptance Criteria:

- Given the user has historical parking logs stored in the system, when the user opens the app or visits the Parking List Screen, then the system should analyze past parking patterns to identify frequently used areas.
- Given the system has identified frequently used areas, when the user views the suggested garages section, then garages near those areas should be displayed as recommendations.
- Given the user parks in a new location multiple times, when new logs are recorded, then the recommendation model should update to include new frequently visited areas.
- Given the user's logs are incomplete or new, when there isn't enough data to make a personalized recommendation, then the system should display nearby popular garages as default suggestions

User Story #6 - As a user I want to know when to leave my house to make it to events on time using a maps API

Task #	Task Description	Estimated Time	Owner
1	Add a destination selector for events/classes on the UI	3 hours	George
2	Research maps APIs (Google vs Apple vs OpenStreetMap	2 hours	George
3	Call chosen API and integrate it with Django	2 hours	George
4	Create Django API route to send data to front end	2 hours	George
5	Integrate with a maps API to link to a maps app	3 hours	George
6	Display total ETA on the UI	3 hours	George
7	Test ETA accuracy and ensure dynamic recalculation when destinations change	2 hours	George

Acceptance Criteria:

- Given that the user is on the map, when they select a destination I want to see the ETA at that destination
- Given that the user has selected a location, when they open up the list view, then I want to see a button that links to a map.
- Given that the user has selected a location, when they click the button that links to the map I want to take them to a maps app.
- Given that the user has selected a location, when they click the button that links to the map I want to take them to the app with the correct destination put in.
- Given that the user is on the maps screen, when they select a destination I want to see an ETA displayed on the screen.

User Story #7 - As a user, I want a detailed view (hours, floors, rules, walking time to destination) so I can evaluate options.

Task #	Task Description	Estimated Time	Owner
1	Design and plan layout for the detailed view (decide sections for hours, floors, rules, and walking time).	2 hours	Utkarsh
2	Update backend or data model to include all required details (hours, floors, rules, coordinates).	2 hours	Utkarsh
3	Build the frontend detailed view screen and connect it to the backend.	3 hours	Utkarsh
4	Add a walking time calculation and display.	1 hour	Utkarsh
5	Test the detailed view for accuracy and fix minor UI issues.	2 hours	Utkarsh

Acceptance Criteria:

- Given that the detailed view is up, when I go to see a lot's open hours, I should see the location's opening and closing hours clearly.
- Given that the detailed view is open, when I tap or scroll to the rules section, I should see all the rules displayed clearly.
- Given that I am trying to go to class, when I select a destination, I should see an estimated or dummy travel time from my current or selected location.
- Given that there is a change in the garage like a change in the available times, when it is changed in the backend, the detailed view should show the updated information after I refresh or reopen it.
- Given that the detailed view is working, when I view the detailed screen on my device, all text and layout should be readable and properly formatted.

User Story #8 - As a user, I want to see which lots have covered spots for shade during the hot summer months or protection from the elements like rain and snow.

Task #	Task Description	Estimated Time	Owner
1	Design and plan how covered and uncovered lots will be displayed (icon, label, or color difference).	2 hours	Utkarsh
2	Update backend or data model to include a field for "covered" status for each lot.	2 hours	Utkarsh
3	Add backend API updates to return	2 hours	Utkarsh

	covered/uncovered data with each lot.		
4	Update the frontend list and detailed view screens to show covered/uncovered indicators.	2 hours	Utkarsh
5	Test the display on different devices and verify that all covered status information is accurate.	2 hours	Utkarsh

Acceptance Criteria:

- Given I am viewing the detailed list of parking lots, when I open the detailed view, then I should be able to see which lots have covered spots.
- Given a parking lot has covered spots, when I view its details, then it should be clearly labeled as "Covered" or shown with an icon.
- Given a parking lot does not have covered spots, when I view its details, then it should be labeled as "Uncovered" or have no icon.
- Given I am viewing the detailed view of a lot, when I scroll through its information, then I should see whether it offers shade or protection from rain and snow.
- Given I am using the app on any device or mode, when I view the covered/uncovered indicators, then they should be easy to see and understand.

User Story #9 - As a user, I want to set a default "home" or "commute origin" so travel estimates always start from there

Task #	Task Description	Estimated Time	Owner
1	Update the backend user profile to store a default origin.	1 hour	Anthony
2	Implement an API endpoint to set and retrieve the default origin.	2 hours	Anthony
3	Build the user interface in "Settings" for managing the default origin.	2 hours	Anthony
4	Integrate the saved default origin into the travel time calculation service	2 hours	Anthony
5	Conduct end-to-end testing for the default origin feature	1 hour	Anthony

Acceptance Criteria:

- Given I am on the "Settings" screen, when I enter a valid address into the "Default Origin" field and tap "Save," then my origin should be saved to my profile.
- Given I have a default origin saved, when I navigate back to the "Settings" screen, then I should see my saved address pre-populated in the "Default Origin" field.

- Given I have a default origin saved, when I view a parking lot's detailed view or request an ETA (per Story #6), then the travel estimate should be calculated from my saved origin.
- Given I do *not* have a default origin saved, when I request an ETA, then the app should default to using my device's current location (or its previous behavior).

User Story #10 - As a user, I want a calendar of known closures/events per lot so I can plan ahead

Task #	Task Description	Estimated Time	Owner
1	Investigate and document the existing backend model for lot events.	2 hours	Anthony
2	Create an API endpoint to fetch upcoming events for a specific lot.	2 hours	Anthony
3	Design and implement the "Upcoming Events" UI on the lot's detailed view.	2 hours	Anthony
4	Connect the frontend UI to the new API to display the event data.	2 hours	Anthony
5	Test the full feature, from creating an event in the admin to seeing it in the app.	2 hours	Anthony

Acceptance Criteria:

- Given an admin has added an event (e.g., "Homecoming Closure") for "Lot A" from Oct 25 to Oct 27, when I view the "Detailed View" for "Lot A," then I should see the event and its dates listed.
- Given a lot has no upcoming events or closures, when I view its "Detailed View," then the "Upcoming Events" section should display a message like "No known closures or events."
- Given an event's end_time is in the past, when I view the "Detailed View" for that lot, then the past event should not be displayed in the list.
- Given multiple events are scheduled for the same lot, when I view the “Detailed View” for that lot, then all upcoming events should be listed in order by date and time.
- Given that an event has been planned for a future date, when I look at closures for a lot for the current date, then I should not see a closure for the current date.

User Story #11 - As a user, I want push notifications when lots are closing

Task #	Task Description	Estimated Time	Owner
1	Develop a backend scheduled task that runs daily to check for an upcoming closure form story #10	2 hours	Anthony
2	Implement the targeting logic to decide *which* users get the alert (e.g., users who have that	3 hours	Anthony

	lot in their "frequent" list from Story #5)		
3	Integrate this logic with the existing push notification service to send alerts (e.g., "Heads up: [Lot Name] will be closed on [Date]").	1 hours	Anthony
4	Add a new toggle in the "Settings" UI to allow users to opt in/out of "Lot Closure Alerts."	2 hours	Anthony
5	Test the feature by adding a closure event and verifying the correct users receive the notification.	2 hours	Anthony

Acceptance Criteria:

- Given an admin has added a closure for "Lot A" for tomorrow, when the daily task runs, then users who frequently park in "Lot A" should receive a push notification.
- Given a user has opted out of "Lot Closure Alerts" in settings, when the task runs, then they should *not* receive the notification.
- Given a lot has no upcoming closures, when the task runs, then no notifications should be sent for that lot.
- Given that a lot had an emergency closure, when the admin updates the lot's status and triggers the update, users with their notifications on should receive an update.
- Given that a lot is closing soon, when the user is on the application, they should get a notification that a lot is closing soon.

User Story #12 - As a user, I want a confidence level on each lot's count (High/Medium/Low) so I can better assess how much I can trust the data for certain lots.

Task #	Task Description	Estimated Time	Owner
1	Design and plan how to display confidence levels (icons, colors, or text indicators for High/Medium/Low).	2 hours	Utkarsh
2	Update backend or data model to include a confidence field for each lot's count.	2 hours	Neel
3	Implement backend logic to calculate or assign confidence levels based on data freshness or camera reliability.	3 hours	Logan
4	Update the frontend list and detailed view screens to show the confidence indicator next to each lot's count.	3 hours	Logan

5	Test the confidence display across devices and ensure the logic and visuals are accurate.	1 hours	Utkarsh

Acceptance Criteria:

- Given that I am on the parking list page, when I view the parking lots, I should see a confidence label (High, Medium, or Low) next to each lot's count.
- Given that the data is reliable, when the user views the reliability of the count, it should show "High" confidence with a clear color or icon.
- Given that the data is moderately reliable, when the user views the reliability of the count, it should show "Medium" confidence with a distinct but neutral color or icon.
- Given that the user opens a detailed view, when they check to see the parking lot count, then they should see the confidence level displayed along with its count.
- Given that the parking list page is open, when viewing the app on different devices or in dark/light mode, then all confidence indicators should remain readable and consistent.

User Story #13 - As a student or employee, I want to filter lots by my parking pass so I can know what garages I'm allowed to park in

Task #	Task Description	Estimated Time	Owner
1	Compile a list of the parking passes eligible for each lot	2 hours	Neel
2	Add parking lot model to the backend	3 hours	Neel
3	Add parking pass to preferences screen	2 hour	Neel
4	Create frontend element	3 hours	Neel
5	Create API route to accept user preference	2 hours	Neel
6	Store user preference in User model	2 hour	Neel
7	Test the storing of preferences	1 hour	Neel
8	Test the frontend element	1 hour	Neel

Acceptance Criteria:

- Given that I am on the main app screen, I want to see a frontend selector presented
- Given that I am on the main app screen, and given that the frontend selector exists, when I use it I want it to accurately filter the lots and only show the ones I choose
- Given that I'm logged in, when I open the settings screen I expect to see a parking pass option
- Given that I'm logged in, when I open the app and have preferences saved I expect them to persist
- Given that my preferences persist, when I open the app I expect the screen to be filtered.

User Story #14 - As a driver, I want to filter for ADA/accessible parking spots if the university provides that data.

Task #	Task Description	Estimated Time	Owner
1	Compile a list of garages that have ADA/accessible spots	2 hours	Logan
2	Add ADA info to parking model	1 hour	Logan
3	Create frontend element to present ADA info and allow filtering	3 hours	Logan
4	Create API route to accept user preference	3 hours	Logan
5	Store user preference in User model	1 hour	Logan
6	Test the storing of preferences	1 hour	Logan
7	Test the frontend element	1 hour	Logan

Acceptance Criteria:

- Given that the user is on the main app screen, when the screen loads, then they should see a frontend element that allows filtering by ADA/accessible parking.
- Given that the user is on the main screen and the filtering option exists, when they apply a filter by number of ADA spots, then the list should update to reflect lots accordingly to the selected ADA spot count.
- Given that the user is on the main screen and a garage has ADA Spots, when they open the detailed view of that garage, they should see an element indicating how many total ADA spots that structure/lot has.
- Given that the user is on the preferences screen and logged in, when they view their preferences, then they should see an ADA filtering preference saved.

User Story #15 - As a user, I want to favorite lots so they show at the top of my list

Task #	Task Description	Estimated Time	Owner
1	Create star element to allow users to “favorite’ a garage	1 hours	Logan
3	Create API route to send favorites to backend	3 hours	Logan
4	Store favorite lots in User model	2 hour	Logan
5	Test the storing of favorites	1 hour	Logan
6	Test the frontend element	1 hour	Logan

Acceptance Criteria:

- Given that the user is on the main app screen with a list of garages visible, when the screen loads, then they should see a star element that allows them to favorite a lot.
- Given that the user is on the main screen, when they favorite a specific garage, that garage should move to the top of that list.
- Given that the user is on the main screen and logged in, when they favorite a garage, then it should be saved to their user preferences.
- Given that the user is on the main screen and logged in with previously favorited garages, when they open the app, then their favorites should automatically be stored and shown at the top.

User Story #16 - As a user, I want to see the price of parking in different lots so that I can pick the best garage considering my parking budget. (Grant St and Harrison St Garages)

Task #	Task Description	Estimated Time	Owner
1	Research what lots require payment, when they do, and how many spots are paid	3 hours	George
2	Add the researched info to the backend models	2 hour	George
3	Add the cost of each garage to the frontend model	2 hours	George
4	Add a frontend element to filter by price	2 hours	George
5	Test the filtering of the elements by price	1 hour	George

Acceptance Criteria:

- Given that the user logs into the app and lands on the main screen, when the screen loads, then the user should see an indication of the price of a garage.
- Given that the user logs into the app and lands on the main screen, when the screen loads, then they should see an element that allows them to filter garages by price.
- Given that the user is on the main screen, when they click on a specific garage, then they should see the exact price and the times or conditions when payment is required.
- Given that the user is on the main screen, when they click on a specific garage, then they should see any event prices listed for that garage.

User Story #17 - As a user, I want a calendar screen that incorporates all events.

Task #	Task Description	Estimated Time	Owner
1	Design and plan the calendar layout showing all upcoming events with dates and times.	3 hours	Utkarsh
2	Create or update the backend endpoint to fetch all events with relevant details.	2 hours	Neel

3	Build the calendar screen on the frontend and connect it to the backend data.	3 hours	Utkarsh
4	Add date formatting, sorting, and grouping logic (e.g., “Today,” “This Week,” “Upcoming”).	2 hours	Logan
5	Test the calendar list for correct display and verify that all events load properly.	1 hours	Utkarsh

Acceptance Criteria:

- Given that the user opens the calendar list screen, when the screen loads, then the user should see all upcoming events displayed in an organized list.
- Given that multiple events occur on the same day, when the events are displayed, then they should be grouped together by date.
- Given that the user views an event from the list, when the event details are shown, then the user should see its name, date, time, and location clearly.
- Given that the user scrolls through the event list, when the list is updated, then older events should not appear, only current or future events should be visible.

User Story #18 - As a user, I want a navigation view where I can input my destination and see estimated travel times and the closest parking garages, so I can plan where to park more easily.

Task #	Task Description	Estimated Time	Owner
1	Design and plan the navigation input view (search bar, map area, and results section).	3 hours	George
2	Set up backend or API integration to fetch travel times (ETAs) and nearest garages based on user input.	3 hours	George
3	Implement the frontend navigation screen with an input field and results display.	3 hours	George
4	Add logic to calculate and display ETAs and distances for nearby garages.	3 hours	George
5	Test navigation input, ETA accuracy, and garage recommendations for correctness.	2 hours	George

Acceptance Criteria:

- Given that the user opens the navigation view, when the screen loads, then the user should see an input field where they can type or select a destination.

- Given that the user enters a valid destination, when the user submits or selects it, then the system should show estimated travel times (ETAs) from their current location.
- Given that the ETAs are displayed, when the system has the destination details, then the user should also see a list or map of the closest parking garages near that destination.
- Given that the user enters an invalid or unreachable destination, when the system attempts to find it, then the user should see a clear error message indicating that the destination cannot be reached or found.

Hour Summary for each person:

Logan: 28 hours

George: 31 hours

Anthony: 32 hours

Utkarsh: 30 hours

Pranay: 32 hours

Neel: 30 hours

Remaining Backlog

1. **Product Backlog**
   1.1. **Functional**

Authentication and Onboarding:

~~1.1.1. As a user, I want sign in with Apple to make creating a user account easier~~
~~1.1.2. As a user I want sign in with Google to make creating a user account easier~~
~~1.1.3. As a user I want sign in with email and password to allow me to make an account and save my preferences~~
1.1.4. As a new user, I want a simple onboarding flow (tutorial/walkthrough) so I understand the app quickly.

Core Functionality:

~~1.1.5. As a driver I want a screen that shows updated counts for each garage~~
~~1.1.6. As a user I want to be able to see accurate availability of lots (Whether the lot is open or not e.g. football games)~~
~~1.1.7. As a user, I want to see a "Last updated <timestamp>" label on each lot so I can trust freshness.~~
~~1.1.8. As a user, I want to search for lots by name/code so I can find them quickly.~~
~~1.1.9. As a user I want to have a map with all of the garages/lots listed on them~~
~~1.1.10. As a user, I want a detailed view (hours, floors, rules, walking time to destination) so I can evaluate options.~~
~~1.1.11. As a user, I want a color gradient on each parking structure (from green to red) that signifies how full a given parking structure is (e.g. green for empty and red for full).~~

Computer Vision:

~~1.1.12. As a developer I want to set up a camera in a parking lot~~
~~1.1.13. As a developer I want to use computer vision to detect cars using online training data~~

~~1.1.14. As a developer I want the computer vision algorithm to not detect other objects or vehicles that are not cars~~

~~1.1.15. As a user I want an accurate count of total parking spots in all garages~~

~~1.1.16. As a developer, I want to sync the camera to the CV model~~

~~1.1.17. As a developer I want to be able to count the automobiles accurately and store the data in a database and thus find the available spots~~

Mapping and Navigation:

~~1.1.18. As a user I want to know when to leave my house to make it to events on time using a maps API~~

1.1.19. As a driver I want a link to a navigation app set to my lot of choice so I can easily see directions to the garage I'm going to

1.1.20. As a user I want to be able to add significant arrival locations to reduce friction and make creating plans easier

~~1.1.21. As a user, I want to set a default "home" or "commute origin" so travel estimates always start from there~~

Trip Planning and Calendar:

~~1.1.22. As a user, I want to link my calendar to the app~~

~~1.1.23. As a user I want to see a bar chart that shows how full a given lot is at the given time~~

1.1.24. As a user I want those logs to be used to see what garages I end up in most frequently by day

1.1.25. As a user I want the garage I end up in and at what time to be logged by the app

~~1.1.26. As a user I want garages suggested to me based on my logs and where I end up most~~

1.1.27. As a user, I want the app to suggest the nearest alternative parking structure when my chosen lot is full.

~~1.1.28. As a user, I want a calendar of known closures/events per lot so I can plan ahead~~

1.1.29. As a user, I want walking time from lot to event/class added to ETA so plans are realistic.

1.1.30. As a user, I want to see a 'Time-to-Full' estimate on the lot detail screen so I can determine whether the lot could be filled by the time I get there or pick an alternative garage.

~~1.1.31. As a user, I want a confidence level on each lot's count (High/Medium/Low) so I can better assess how much I can trust the data for certain lots.~~

~~1.1.32. As a user, I want to see the price of parking in different lots so that I can pick the best garage considering my parking budget. (Grant St and Harrison St Garages)~~

1.1.33. As a a user, I want to see an average user rating for each lot so that I can factor in the quality of different lots when deciding which lot to park in.

~~1.1.34. As a user, I want to see which lots have covered spots for shade during the hot summer months or protection from the elements like rain and snow.~~

Eligibility, Accessibility and Personalization:

~~1.1.35. As a student or employee, I want to filter lots by my parking pass so I can know what garages I'm allowed to park in~~

~~1.1.36. As a driver, I want to filter for ADA/accessible parking spots if the university provides that data.~~

~~1.1.37. As a user, I want to favorite lots so they show at the top of my list.~~

Analytics:

1.1.38. As a user I want to compare the insights from 2 or more garages

1.1.39. As a user, I want a bar chart displaying historical average hourly insights to see when the best time to park in a given garage is.

1.1.40. As a user, I want seasonal/weekly based historical insights to see which days it may be easier to park in a week and how seasonal weather affects parking

1.1.41. As a developer, I want to train a model to predict parking spots based on historical hourly, weekly, and seasonal trends as well as current data and active users

1.1.42. As a user, I want the app to prompt me with alternative transportation methods if most parking lots are expected to be full by a given departure time.

Notifications and Preferences:

1.1.43. As a user I want push notifications that tell me when to leave

~~1.1.44. As a user I want push notifications when lots are closing~~

1.1.45. As a user I want push notifications when my car is in a lot that is getting towed

1.1.46. As a user I want push notifications when my car is frequently associated with a lot that is being towed

~~1.1.47. As a user, I want to manage notification preferences (which alerts I get and when).~~

1.1.48. As a user, I want push notifications when a favorite lot drops below N spaces so I can leave sooner

~~1.1.49. As a user I want a push notification that tells me when parking passes go on sale.~~

1.1.50. As a user I want a push notification after I enter a garage that tells me to park safely between the lines

User Feedback and Sharing:

1.1.51. As a user I want to be able to share a parking schedule with others over text, email e.g.

1.1.52. As a user I want to see a map with the live counts displayed on top of the garages

1.1.53. As a user I want to be able to filter that map by parking pass or favorite lot

~~1.1.54. As a user, I want to toggle between a list view and a map view depending on my preference.~~

1.1.55. As a user, I want to report incorrect lot status (e.g., the app says open but it's actually full) so developers can improve data quality.

1.1.56. As a user, I want to rate accuracy ("was this lot accurate?") so the model improves.

User Interface:

~~1.1.57. As a user I want an attractive app icon that follows the new liquid glass design language for iOS users.~~

~~1.1.58. As a user, I want the color scheme of the app to follow the Purdue color scheme as a Purdue-associated app.~~

1.1.59. As a user I want smooth animations that make using the app more pleasant

Miscellaneous:

1.1.60. As a student or staff member, I want the app to correctly work on the latest versions of iOS and/or Android so that I can use it no matter my phone

~~1.1.61. As a developer I want to set up a Redis cache and a Postgres SQL database that stores the data and sends it to the frontend~~

1.1.62. As a user, I want fallback messaging ("Data temporarily unavailable") instead of blank screens.

1.1.63. As a user, I want the app to integrate with my car's infotainment system (CarPlay/Android Auto) so I don't have to look at my phone while driving. (If time permits)

1.1.64. As a developer, I want health checks for cameras/CV services so I'm alerted when a feed drops.

1.1.65. As a developer, I want structured logs + metrics (latency, error rate, queue lag) so I can trace problems.

1.1.66. As a developer, I want to import permit rules from a campus feed so eligibility stays current.

1.1.67. As a developer, I want to ingest event calendars (athletics, concerts) so predicted demand is accurate.

1.2. Non-Functional

1.2.1. Architecture and Performance:

Our Project's Frontend will be written in React Native for cross platform delivery in iOS and Android. The backend will be implemented in Java/Python and will be deployed as containerized microservices. Redis will be present to provide low latency caching for live counts. PostgresSQL will be used as the main database to store daily counts, analytics and metadata. A lightweight computer-vision service will process camera streams and publish enter/exit events to the backend. We will target an end-to-end median response time of < 500 ms for API reads with 95% < ls and a mobile app cold start time of < 2 s on mid-range devices. The live availability counter will reflect events with a freshness target of ≤ 10 s from vehicle passage to user display with 95% < 20s. The system should support ≥ 1,000 concurrent users. The app should have 24hr uptime, apart from scheduled maintenance and outages.

1.2.2. Security and Privacy:

All network communication will use TLS 1.2+. Authentication will support Sign in with Apple/Google and email/password and tokens will be short-lived with secure refresh. At rest, Redis and PostgreSQL volumes will be encrypted; secrets will be managed via environment variables with no secrets in source control. Camera processing will not store raw video frames unless explicitly authorized for test/debug, the default pipeline emits counting events only (no faces/plates). Any temporary debug

captures will be time-boxed and access-controlled. Access to admin tools will require role-based access control.

1.2.3. Usability and Design:

The interface will be designed for one-handed use, with primary actions reachable within 2 taps from the home screen. We will support dark/light modes, large-text settings, and ensure high contrast for key screens. Map and list views will be interchangeable, with persistent preferences. Critical information such as lot status and “Last updated <timestamp>” will be visible at a glance. We will implement a functional UI with smooth transitions that will look pleasing on both Android and iOS devices.

1.2.4. Compliance and Ethics:

We will respect campus policies on camera placement and data handling, and commit to privacy-preserving CV (no biometric identification, no license plate retention). Location and calendar access will be opt-in with clear purposes stated. User-generated feedback will be moderated to remove sensitive data or misuse. If required by campus policy, we will provide a Data Protection Impact Summary for review.