CS 307 Sprint 1 Planning - Group 6

BoilerPark

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

Sprint Overview:

During this sprint, we hope to initialize the critical backend components such as the Redis Cache and PostgreSQL database so the count of cars can be tracked and logged. Also, we will attempt to train a Computer Vision model that will track vehicles entering and exiting parking structures. Furthermore, we will begin constructing the backend that transmits updates from the edge computers (cameras) to the Redis and Postgres databases. Additionally, we will create a rough and incomplete draft of the UI using React Native for planning and design purposes. Finally, we will begin to construct the backend that handles data aggregation and processing for parking lot availability data.

Scrum Master: George Samra

Meeting Plan: Sunday @ 8pm Wednesday @ 9am (with project coordinator) & Friday @ 4:30pm

Risks and Challenges: The big challenge for this sprint will be getting all of the disparate pieces of our app architecture up and running and connected. We've got a Redis cache, a PostgreSQL database, a React frontend, and a Python backend that all need to talk to one another and share data. On top of the app architecture, getting the CV model up and running will be a big part of our first sprint. Once that is working, we'll be able to accurately test passing data and updating the frontend. We're also going to be creating the basic framework of our UI as well as user accounts and allowing our users to sign in with Apple and Google. We hope to have user accounts and a functioning architecture by the end of this sprint.

Current Sprint Detail

User Story #1 - As a developer I want to set up a Redis cache and a PostgreSQL database that stores the data and sends it to the frontend**.**

Task #	Task Description	Estimated Time	Owner
1	Create a Redis Cache to store the parking lot availability counters	1 hour	Neel
2	Create a PostgreSQL database to store historical data and login information	2 hour	Neel
3	Configure the Redis Cache to store key-value pairs tracking remaining parking spots	2 hours	Neel
4	Configure the Redis Cache to store key-value pairs for tracking last updated time	2 hours	Neel
5	Configure the PostgreSQL database with tables for login information and each parking lot	2 hours	Neel
6	Configure a Redis Pub/Sub channel to send key-value pair updates to clients	2 hours	Neel
7	Test the robustness and functionality of the Redis Cache and PostgreSQL databases	2 hours	Neel

Acceptance Criteria:

- Given that a Redis Cache value is updated, when a client is active, the Redis Cache swiftly and accurately posts the updated value to the Pub/Sub channel and the client shows the update onscreen.
- Given an active and working PostgreSQL database, when the user creates a new account, the account details are stored into the login information table and they are able to log back in.
- Given a correctly-configured Redis Cache, when an update is sent to the Redis Cache, the appropriate key-value pairs are updated.
- Given a correctly-configured Redis Cache, when a 15-minute time period passes, the current availability metrics are sent to the PostgreSQL database and then sent to the frontend where they are displayed.

User Story #2 - As a developer I want to set up a python backend using django and implement basic routes and server functionalities.

Task #	Task Description	Estimated Time	Owner
1	Create Python backend and initialize Django framework	3 hours	George
2	Implement basic routing to the Redis cache and to the PostgreSQL database	4 hours	George
3	Test the connection by incrementing and decrementing Redis and checking backend	2 hours	George
4	Test the connection to the PostgreSQL database	2 hours	George

Acceptance Criteria:

- Given that the Redis cache has been set up, when I increment/decrement Redis then I expect the backend to also increment/decrement
- Given that the Redis cache has been set up, when I make a request from the backend then I expect the correct timestamp to appear in the backend
- Given that the PostgreSQL database has been set up, when I query the database I expect to receive a valid result
- Given that the PostgreSQL database has been set up and the Django ORM has been set up, when I query the database I expect the result to fit into my ORM

User Story #3 - As a user, I want sign in with Apple to make creating a user account easier

Task #	Task Description	Estimated Time	Owner
1	Create and register app with Apple	1 hour	Utkarsh
2	Install Apple SDKs and libraries	1 hour	Utkarsh
3	Create a button on the frontend to sign in with Apple	2 hours	Utkarsh
4	Connect to the backend and implement endpoints for OAuth, tokens, and verification	5 hours	Utkarsh
5	Test integration to the PostgreSQL database, backend, and frontend	2 hours	Utkarsh

Acceptance Criteria:

- Given that the app is registered with Apple, when I sign in, I expect a button that gives the option to sign in with my Apple account on the sign-in page
- Given that the the Apple SDKs and libraries are installed and correctly integrated, when I click on the Apple sign-in button, I expect to be prompted for my Apple credentials or face ID

- Given that the the Apple SDKs and libraries are installed and correctly integrated, when I input correct Apple credentials or face ID, I expect to be verified and directed to the home page
- Given I use the app for the first time, when I complete Apple sign-in, then I'm taken to a short one-time onboarding screen to confirm the display name before entering the app.
- Given that the endpoints for OAuth, tokens, and verification are connected to the backend, I expect my personal account information to be remembered from my previous sessions with my Apple account

User Story #4 - As a user I want sign in with Google to make creating a user account easier

Task #	Task Description	Estimated Time	Owner
1	Register the app in Google Cloud Console and obtain OAuth 2.0 client credentials (Client ID & Secret).	1 hour	Utkarsh
2	Install Google OAuth SDKs and libraries	1 hour	Utkarsh
3	Create a button on the frontend to sign in with Google	2 hours	Utkarsh
4	Connect to the backend and implement endpoints for Google OAuth callback, token exchange, and Google ID token.	5 hours	Utkarsh
5	Test integration with PostgreSQL database, backend, and frontend	2 hours	Utkarsh

Acceptance Criteria:

- Given I am on the login screen, when the page loads, then I see a "Sign in with Google" button styled with Google's brand guidelines and a visible hover/focus state.
- Given I click "Sign in with Google," when I am redirected, then I land on the official Google sign-in/consent page in a new tab or the same tab without any mixed-content or certificate warnings.
- Given I use the app for the first time, when I complete Google sign-in, then I'm taken to a short one-time onboarding screen to confirm the display name and (if applicable) username before entering the app.
- Given I have previously signed up with Google, when I sign in again, then I bypass onboarding and land on the default post-login destination (e.g., dashboard).
- Given that the backend endpoints for callback, token exchange, and verification are handled correctly, I want to have a JWT token generated and the corresponding PostgreSQL tables updated.

User Story #5 - As a user I want sign in with email and password to allow me to make an account and save my preferences

Task #	Task Description	Estimated Time	Owner
1	Create a user data type in the backend and database	1 hour	George
2	Create REST API routes for basic login and signup	2 hours	Utkarsh
3	Create some generic preferences that can be used to test persistence	3 hours	Logan
4	Create unit tests for the login and signup	2 hours	Logan
5	Test with admin users and run end to end tests	2 hours	Utkarsh

Acceptance Criteria:

- Given a frontend with two input fields I should be able to input an email and password and have them checked to be unique and strong respectively.
- Given the ability to create a user I should be able to create a user and receive a success message on user creation
- Given a login page I should be able to login with my previously entered details and receive the same information I had entered
- Given some app preferences, I should be able to change and save my preferences and receive them on a separate login

User Story #6 - As a developer I want to use computer vision to detect cars using online training data

Task #	Task Description	Estimated Time	Owner
1	Research pre-labeled car detection models from Open CV (COCO, Pascal VOC, Open Images, Roboflow)	3 hours	Pranay
2	Download pre-trained model and verify environment set up	3 hours	Pranay
3	Download parking lot and non parking lot video footage	2 hour	Pranay
4	Verify and test car image is detected by model	1 hour	Pranay
5	Verify and test that the test negative image is not detected by the model.	1 hour	Pranay

Acceptance Criteria:

- Given the need for a car detection model, when I research various Open CV options for car detection models, then I will document my findings and organize them into pros and cons list.
- Given that the pre-trained model is downloaded, when I attempt to run the program, the program correctly bounds the car with the box.
- Given that the model identifies the bounding box for the car, the resulting bounding box image is correctly saved to a file.
- Given that the model is downloaded, the model does not identify any cars in the image without cars when a sample video is run and the frontend does not show any updated counts.

User Story #7 - As a driver I want a screen that shows updated counts for each garage

Task #	Task Description	Estimated Time	Owner
1	Link Redis Cache to PostgreSQL DB through Python HTTP request on edge computer	2 hours	Pranay
2	Develop backend Django route to query the Redis Cache	2 hours	Pranay
3	Develop backend route to query the PostgreSQL DB as a fallback	2 hours	Pranay
4	Develop display to show the count for the parking lot	2 hours	Pranay
5	Test backend routes and user display	2 hours	Pranay

Acceptance Criteria:

- Given that the Redis Cache is linked to the PostgreSQL DB, when I access the parking lot screen, then I should see the current count of cars for each garage displayed
- Given that the Redis Cache value updates, when I refresh or load the parking lot screen, then the displayed count should reflect the latest value from Redis
- Given that Redis is unavailable, when I access the parking lot screen, then the displayed count should fall back to the value from the PostgreSQL DB.
- Given that multiple garages are listed, when I select or switch between garages, then the display should update to show the count for the selected garage.

User Story #8 - 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	Define event schema for edge to backend messages	1 hr	Pranay
5	Develop edge computing code to detect cars coming in and cars coming out	2 hours	Pranay
6	Connect the edge computing resource to the Redis Cache and test	3 hours	Neel

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

Task #	Task Description	Estimated Time	Owner
1	Design App Icon	3 hours	George
2	Create Apple version of the App Icon with dark and clear versions	2 hours	George
3	Upload it to the codebase and check the correctness	1 hour	George
4	Design a React components that utilises the icon and displays it on the home screen	3 hours	Logan
5	Add the app icon to a splash screen that appears on load	1 hour	Logan

Acceptance Criteria:

- Given that we have an app framework done, when I load it onto a simulator I expect to see our app icon
- Given that I am loading the app framework onto an iOS device, when I load it onto a simulator running iOS I expect to see it with the liquid glass design
- Given that I am loading the app framework onto an iOS device, when I load it onto a simulator running iOS I expect to be able to change to dark or clear mode and have the icon update to match
- Given that I am loading the app after having closed it or after downloading it, I am able to see a splash screen with the app icon on it
- Given that I am using the app with completed components, I am able to see at least one component with the app icon in it

User Story #10 - As a developer I want to setup a react native frontend with a proper file structure and basic API routes to connect to the frontend and redis

Task #	Task Description	Estimated Time	Owner
1	Design React Native frontend architecture and format	1 hour	Anthony
2	Create an easy-to-understand file structure to store and organize frontend files	1 hour	Anthony
3	Develop basic theme (colors and format) that will be consistent across pages	2 hours	Anthony
4	Connect the frontend and redis cache	3 hours	Neel
5	Develop working log-in page	1 hour	Logan
6	Develop working home page (list of Purdue garages)	2 hours	Logan
7	Develop buttons on the bottom of pages to change between pages	1 hour	Anthony
8	Test integration of pages	1 hour	Anthony

Acceptance Criteria:

- Given that the basic frontend pages have been developed, when I click the app icon, I expect to be taken to the log-in page
- Given that multiple frontend pages have been developed, when I am logged in to the app, I expect to have an array of buttons at the bottom to switch between different pages
- Given that multiple frontend pages have been developed, when I change between different pages in the app, I expect the color combinations to be consistent across pages
- Given that Redis has been set up, when I view the parking lot availabilities, I expect to see availability data regularly updated (from the Redis database)

User Story #11 - As a user, I want to link my calendar to the app

Task #	Task Description	Estimated Time	Owner
1	Set up the frontend element to accept the uploading of an iCS file	3 hours	Anthony
2	Send the iCS file to the backend	2 hours	George
3	Ingest and clean data	1 hour	George
4	Create a basic display for uploaded calendar events	2 hours	Pranay
5	Test accuracy of calendar upload	2 hours	George

Acceptance Criteria:

- Given that the user wants to upload their calendar, when make the decision I want an upload button on screen for them to press
- Given that the button exists, when pressed I want it to successfully open an upload dialog
- Given that the user successfully uploaded their calendar, when they do so I want the python backend to take it in and clean it
- Given that a calendar event has been created, display a basic view of the event so the user can confirm that the event has been uploaded
- Given that the user has connected the app to their calendar have a basic icon display to repeat the event

User Story #12 - As a user, I want to see a “Last updated <timestamp>” label on each lot so I can trust freshness.

Task #	Task Description	Estimated Time	Owner
1	Set up the PostgreSQL table to have a last updated column	1 hour	Neel
2	Setup the frontend client to subscribe to the Redis Pub/Sub channel	1 hour	Neel
3	Develop a Python script to update the PostgreSQL table with the updated count for parking lots every 15 minutes	2 hours	Neel
4	Develop a Django route to fetch the count	1 hour	George
5	Test the connection between the database, the Redis cache, and the backend	1 hour	Logan

Acceptance Criteria:

- Given that the PostgreSQL table is created, when I query the database for all columns, then I should see the Last updated time stamp column
- Given that the PostgreSQL table is created, when the user selects a parking lot, they should see the last updated count.
- Given that the PostgreSQL table is created, when the user selects a parking lot, they should see the last updated timestamp value filled in..
- Given that the user asks for the current open spots, the user sees a bar with a color responding to how full the lot is.
- Given that a Python script to query the Redis Cache is created, when I run the script, then I should see the correct value from the Redis Cache.
- Given that the Redis Cache and PostgreSQL database are created, when the user selects a parking lot to see, the updated data should be shown
- Given that the crontab job is set, when I check the count after 5 minutes, I should see a different value if I have updated the Redis cache when using the Django route to fetch the data.

User Story #13 - As a user, I want to manage notification preferences (which alerts I get and when).

Task #	Task Description	Estimated Time	Owner
1	Create a list of notifications that we want to offer users	2 hours	Neel
2	Create a list view in the frontend that contains a list of all the notifications and allows a user to select which ones they want to receive	4 hours	Utkarsh
3	Set up route to the backend that accurately conveys the information	4 hours	Neel
4	Test backend user notification settings with a fake user	2 hours	Logan

Acceptance Criteria:

- Given that the frontend has been set up, when I load the app I expect to be able to find a list of the notifications that our app offers
- Given that the list has been implemented, when I load the view I expect to be able to change my notification preferences
- Given that I can change my notification preferences, when I try to change my preferences I want that change to be reflected in the backend
- Given that I can log into the app, when I change my preferences and close the app, I expect that when I open it back up that they will have saved
- Given that I have granted push notification permissions to the app when I enable push notifications, I am sent a notification to test.

User Story #14 - As a user I want a push notification that tells me when parking passes go on sale

Task #	Task Description	Estimated Time	Owner
1	Add push notification permissions and token registration in the mobile app	2 hours	Logan
2	Create API endpoints to enable user to enable push notifications	2 hours	Logan
3	Store and manage user device tokens in the PostgreSQL database	2 hours	Logan
4	Add backend logic to notify active users when parking passes go on sale	3 hours	George
5	Test that devices receive push notifications when a sale is active	2 hours	Logan

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 #15 - As a user I want to have a map with all of the garages/lots listed on them

Task #	Task Description	Estimated Time	Owner
1	Find/create Purdue map	2 hours	Anthony
2	Add pindrop of tracked parking lots on the map	2 hours	Logan
3	Create map page on UI	2 hours	Logan
4	Add number on pindrops that display how full the lot is	2 hours	Logan
5	Connect availability data from database to frontend map	3 hours	Neel
6	Test availability updates from database	2 hours	George

Acceptance Criteria:

- Given that a Purdue map has been added, when I open the map page in the app, then I should see the base campus map displayed.
- Given that parking lots and garages are tracked, when I load the map page, then I should see pins for all available lots on the map.
- Given that each pin represents a parking lot, when I view a pin, then I should see a number showing how full the lot is.
- Given that availability data is stored in the database, when the frontend map connects, then the pins should update to reflect the latest availability.
- Given that new data is pushed into the database, when I refresh or wait for updates, then I should see the map reflect the new availability values.
- Given that the map UI is implemented, when I open the app and navigate to the map page, then the map should load without errors and allow me to view all pins.

User Story #16 - As a user, I want to toggle between a list view and a map view depending on my preference.

Task #	Task Description	Estimated Time	Owner
1	Add a toggle button (e.g., “List / Map”) to the UI.	2 hours	Utkarsh
2	Implement a list view that displays lots/garages with availability info.	3 hours	Utkarsh
3	Connect toggle logic to switch between the list view and the map view.	2 hours	Utkarsh
4	Ensure data (lots and availability) is consistent across both views.	2 hours	Utkarsh
5	Test switching between views to confirm state persists and no errors occur.	1 hour	Pranay

Acceptance Criteria:

- Given I am on the parking page, when I press the “List” button, then I should see a list of all garages/lots with their availability.
- Given I am on the parking page, when I press the “Map” button, then I should see the map view with pins for all lots.
- Given I have switched views, when I return to the parking page, then my selected view should still be shown (state persists).
- Given that availability data updates, when I view either the list or the map, then both should show the same, up-to-date data.
- Given I rapidly toggle between views, when the screen updates, then there should be no crashes or missing data.

User Story #17 - As a user, I want to choose dark mode/light mode so the app fits my phone theme

Task #	Task Description	Estimated Time	Owner
1	Create a toggle on the frontend that effectively switches between light and dark mode	3 hours	Anthony
2	Send those changes to the backend through the proper channels	1 hour	Anthony
3	Create UI elements in both light and dark modes	4 hours	Anthony
4	Test the accessibility of the UI elements in the light and dark modes	1 hour	Anthony

Acceptance Criteria:

- Given that the frontend has been set up, when I load the app I expect to find a toggle that will allow me to switch between light and dark mode
- Given that user accounts have been set up, when I have set my preference, I expect to close and open the app and see my preferred scheme.
- Given that there is a system preference of light mode or dark mode, when I first open the app the app is in my system color scheme
- Given that there is consistent sleek UI, the app should contain dark mode and light mode versions of every page and component created

User Story #18 - As a user, I want the color scheme of the app to follow the Purdue color scheme as a Purdue-associated app.

Task #	Task Description	Estimated Time	Owner
1	Find and decide what actual hex colors to use in our app	1 hour	Anthony
2	Create UI elements with our specified colors in both light and dark modes	4 hours	Anthony
3	Test the legibility of the UI elements in both light and dark mode	2 hours	Anthony

Acceptance Criteria:

- Given that the frontend has been set up and I choose dark mode, when I load up the application, I should see Black and Gold as the primary colors of the application on each page.
- Given that the frontend has been set up and I choose light mode, when I load up the application, I should see White and Gold as the primary colors of the application on each page.

- Given that frontend has completed UI components, when I go to a page I should be able to see a consistent color scheme drawn from a limited selection of colors

Link to Gantt Chart: CS307 Sprint 1 Gantt Chart - Team 6

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

~~1.1.60. t~~

Miscellaneous:

1.1.61. 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

~~1.1.62. 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~~

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

1.1.64. 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)

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

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

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

1.1.68. 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% < 1s 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.