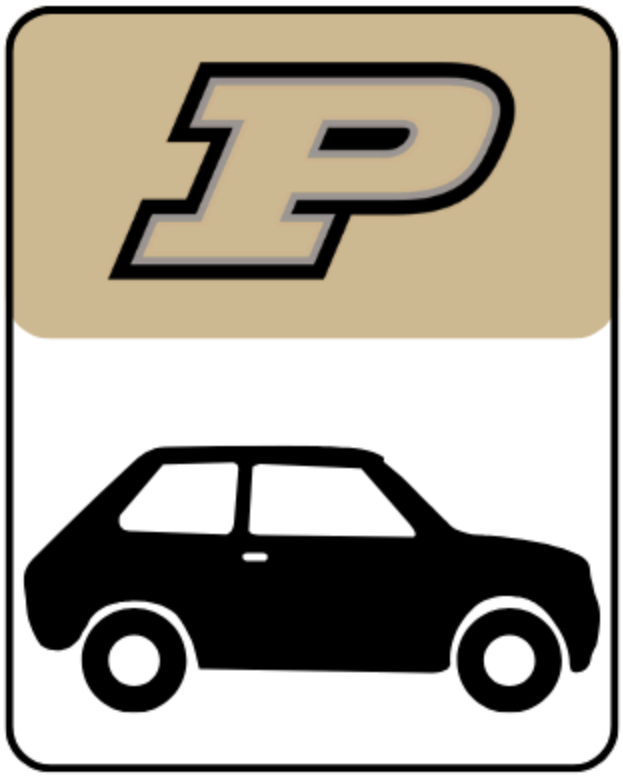

CS 307 Sprint 2 Retrospective - Group 6

BoilerPark

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

What went well during Sprint 2?

We split up the user stories better this sprint. Each user story, for the most part, was assigned completely to only one person. This allowed us to work individually and get our stories done without having to navigate a web of complexity in assigning tasks and allowed each person to get their work done on a timeline that worked for them. Additionally, we were more vigilant about our merges, ensuring that they were done early and often. This prevented us from having long nights of trying to merge disparate branches close to the deadline. We also used Github more effectively this sprint. By having a dedicated dev branch, we were able to continually update our local code with the remote changes, which also made merging easier. Our communication and flexibility also were assets this sprint. After struggling with communication last sprint, using the dedicated time that we used to have lectures allowed us to have better and clearer communication. Finally, all of us showed up to the sprint review, which was obviously an improvement.

User Story #1 - As a developer, I want to sync the camera to a parking lot video

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
1 and 2 (new)	Record an existing parking lot until we can see cars going in and out of the lot.	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

<u>Completed:</u>

While we were not able to get a meeting scheduled with Purdue Parking, we used our own camera outside of a test parking lot near one of our homes. Additionally, we were able to then refactor our code base to connect that video to the backend car counter. We configured the video and were able to refactor the crossing logic for the specific lot and camera angle quickly. The correct Redis Value was then updated and reflected on 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

Completed:
When the user logs into the application on the development build on test flight they are currently prompted to enable push notifications and the token is saved in our backend to send notifications for that user. We are able to send the notification requests to users via curl requests to verify the notification is personalized. What currently needs to be finished with this user story is that we need to fully setup the dev build to have all of the servers working so we can have the push notifications be sent out on a physical device.

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

Completed:
When the user searches for a lot by name or code, the lots matching the search criterion appear on the screen and all other garages disappear. When the search criterion is removed, then all the lots re-appear like normal. Additionally, all parking lots with approximately greater than 50 spots are added to the frontend list for users to see.

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 data 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

Completed:
The data tabs appear on the frontend. We also developed a backend route to query a parking lot's availability across different time ranges. Therefore, when a user wants to pick a parking lot, each and every lot can be picked. Additionally, users can select whether or not they want availability based on the last day, week, or month. The latest availability is filled in as a number on the frontend per lot and the averages the user selects show in the bar chart.

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
1 (new)	Fill in PostgreSQL table with dummy parking lot data to simulate parking trends	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
4 (new)	Implement a frontend UI section which displays garage availability based on when a user plans on parking at a lot based on the day of the week and time of day	2 hours	Pranay
5	Integrate backend recommendations with the frontend display	1 hour	Pranay
6	Test that frequent parking patterns lead to	1 hour	Pranay

	accurate recommendations		

Completed:
We switched from doing personalized recommendations based on user history to finding out averages based on times users are interested in. We made this decision so we can use more data to build more accurate insights. A user can filter based on an availability threshold and check whether or not a lot is x% full at specific times the user is interested in. Additionally, when the user selects a lot, the accurate rolling average appears for the given time, allowing users to see when lots are full rather than when they usually go to park. The Django route has been tested and accurately sends the data to the frontend which displays it correctly.

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

Completed:
We were able to access the Google Maps API to allow the user to enter a parking garage they wanted to go to. The API was pretty easy to use and allowed us to both support autocompleting queries and to restrict the potential destinations both by location and by type. At the end, we were able to display the proper ETA on the frontend and allow the user to be taken to a maps app with the correct destination inputted.

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

Completed:
We had a couple snags with the UI and readability of some of the detailed views that we will be fixing in the upcoming sprint. Apart from that the detailed view works well in terms of functionality apart from the ETA to the garage that is dependent on another story.

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 covered/uncovered data with each lot.	2 hours	Utkarsh
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

Completed:
The shade icons worked as intended and the design was reliant on the detailed view design so as that gets updated so will the shade icon designs. The backend section works completely as intended. So this story is completely implemented.

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

Completed:
This user story is fully implemented. The user is able to go to the settings screen and they are able to put in their starting location where they want their ETA to be calculated from. The ETA currently displayed in the detail view shows the time to a garage and shows time in minutes and distance to the garage in miles.

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

Completed:
This user story was fully completed. I documented the existing backend model for lot events, built a new API endpoint to fetch upcoming events, and connected it to the frontend. I also designed and implemented

the “Upcoming Events” section on each lot’s detailed view and tested the full flow to ensure events created in the admin appear correctly in the app. This feature lets users easily see upcoming closures and plan ahead.

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	3 hours	Anthony
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

Completed:
This user story was completed successfully. Push notifications now work when lots are closing, and the backend logic correctly sends alerts to opted-in users. We verified functionality in local and development environments, but we still need to set up the TestFlight build to ensure notifications work properly on iOS devices. Once that’s configured, the feature will be fully ready for production use.

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

Completed:
We were able to successfully calculate confidence levels for each garage by using dummy values since we have not been able to install cameras and generate real data. However, the confidence levels worked using the dummy values, and we were able to successfully display confidence levels on the detailed view for each garage. We decided against displaying the confidence value directly on the garage list to prevent overcrowding the UI and decluttering the space to make the app design more intuitive and usable.

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

Completed:
When the user opens the app, they now have an option to filter parking lots by paid and parking passes next to the search bar. When the user filters by one/many parking passes, only parking lots and structures relating to those specific filters are shown. Additionally, when the user relaunches the app, if existing filters were applied, the filters would persist and filtered lots would show upon launch.

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	2 hours	Logan

	ADA/accessible spots		
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

Completed:
We were able to find and display ADA parking lot information on the front end. The ADA information can be found by toggling over a garage on the map page, as well as being displayed on the garage list screen. Additionally, we were successfully able to create a filter based on ADA spots on the garage list. This is displayed as a wheelchair button at the top of the page. When pressed, the garages are listed in descending order of ADA spots (this was tested and proven to work correctly).

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

Completed:
We were able to add the favorite button on the garage list screen. The favorite button is displayed as a hollow star to the right of each garage that gets filled in when a user pressed on it. Additionally, when the user presses on the favorite button, the garage is considered a favorite for that user. Along with the button, the garage page also has a “sort by favorite” button that will bring the user’s favorite garages to the top of the list. This favorite sorting is the default sorting method for the garage page.

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

Completed:
This was a pretty straightforward user story. Researching the paid elements to the garages was not especially difficult, and we were able to get it added to the app. I made a paid garage-specific sorting element, but when Neel showed the way that he was filtering garages, it made a lot of sense to incorporate the element into his filtered view, so I worked on doing that.

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

Completed:
This user story is pretty much complete except for the backend endpoint that allows stories to be events to be stored in the database. A user can now go to the calendar view and see the list of events that they have created as well as any future events that they have implemented from their calendar file.

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

Completed:

This user story also went pretty well. The Google Maps API has a search nearby feature that allows us to pick the garages that are closest to any location in the West Lafayette area. We were then able to take that data, format it, and list it along with directions to that parking garage. This was a user story that we felt was able to make our app more usable from a finding what parking spots work perspective.

User Stories/Tasks Not Completed:

User Story #1 - As a developer, I want to sync the camera to a parking lot video

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
1 and 2 (new)	Record an existing parking lot until we can see cars going in and out of the lot.	1 hour	Pranay

Not Completed:

We were not able to schedule a meeting with Purdue Parking to set up a camera or get a camera from Purdue CS. Instead, we filmed a parking lot near one of our apartments and used that footage to rework the backend car counter computer vision code.

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

Not Completed:

When the user logs into the application on the development build on test flight they are currently prompted to enable push notifications and the token is saved in our backend to send notifications for that user. We are able to send the notification requests to users via curl requests to verify the notification is personalized. What currently needs to be finished with this user story is that we need to fully setup the dev build to have all of the servers working so we can have the push notifications be sent out on a physical device.

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
1 (new)	Fill in PostgreSQL table with dummy parking lot data to simulate parking trends	2 hours	Pranay
4	Implement a frontend UI section for “Suggested Garages” on the Parking List Screen	2 hours	Pranay
4 (new)	Implement a frontend UI section which displays garage availability based on when a user plans on parking at a lot based on the day of the week and time of day	2 hours	Pranay

Not Completed:

Instead of developing personal insights, we thought that it would make more sense to take entire parking lot averages to have more data for insights based approaches. This allowed us to analyze rolling averages of parking lot averages based on the time of day and time of week. This allows users to see times they are interested in like Mondays at 9 AM before their first class and see on average how full it is and whether or not it is above a certain threshold they are comfortable with.

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

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

Not Completed:
We decided to keep all of the processing on the client side for this user story(e.g. a fat client). The autocomplete element had to be frontend only, and at that point there was not a good reason to make extra calls to the Django backend that would simply slow down the process.

Not Completed:

During development, we realized that instead of tracking the last updated time in Redis, it would make more sense to use the local clock to show the last updated time. This means that as users refresh the list, the time shown will reflect when they last refreshed the list, not when the value in Redis itself was actually updated. This realization made this task obsolete. Additionally, we were not able to properly store a user's favorite garages in the backend. A user's favorite garage does not persist when the user logs back in.

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	3 hours	Anthony
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

Not Completed:
This user story was completed successfully. Push notifications now work when lots are closing, and the backend logic correctly sends alerts to opted-in users. We verified functionality in local and development environments, but we still need to set up the TestFlight build to ensure notifications work properly on iOS devices. Once that's configured, the feature will be fully ready for production use.

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
5	Store user preference in User model	1 hour	Logan
6	Test the storing of preferences	1 hour	Logan

Not Completed:
We were not able to implement saving a user's preference for ADA filtering with the user model due to complex API and loading errors. To limit UI complexity, we decided to save the user's preference for ADA filtering every time they hit the ADA filtering button (instead of a switch in settings). However, we were not able to fully integrate this functionality into the app this sprint. Luckily, this is not an essential part of the app functionality.

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
4	Store favorite lots in User model	2 hour	Logan
5	Test the storing of favorites	1 hour	Logan

Not Completed:
We were not able to successfully store a user's favorite garages in the user model. Due to this, if a user logs out and logs back in, their favorite garages will not be "remembered." This was not completed because of numerous bugs occurring during backend-frontend communication.

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

Not Completed:
The only thing not implemented is the long term storage and this will be the next thing implemented and should not take too long as we already have a model to store the events just need to implement the routes and fix it.

What did not go well during Sprint 2?

We improved a lot during this sprint, but there are some things that we'd like to do better for Sprint 3. Our sprint demo went better than last time, partially owing to the fact that we were all there, but we'd like to be able to practice a bit more before our next one. A couple of our features worked Thursday night, but somehow got lost in the shuffle of merging and changing things at the last minute. Given more time to practice, we'd be able to ensure those bugs are ironed out. We also did not have all of our features ready on our dev build of our app at the time of the sprint demo. We had to switch screens at the end to show features that we should've had on the demo build. Someone (George) kept committing the Google Maps API key to Github.

How should we improve?

For the last sprint, we'd like to be more on top of getting work done early. We're aiming to be all done with coding new features by November 25. This will allow us to have time to both spend more time practicing our demo and to spend more time quashing bugs for a more seamless, practiced demo. We also would like to start working on our testing doc sooner, we realized relatively late that we had to fill it out. On top of the process improvements, we'd like to pick user stories that are more complementary, last sprint it felt like we picked stories that were a bit less connected. Finally, we want to make a .env file so that we don't commit more API keys (cough cough George).