Capstone Portfolio

Disc Golf Tracker Application

ESOF 423 - Spring 2026

Nathan Mosher & Jaden Hampton

Section 1: Program and Tech Stack

Discussing the start of the project lifecycle, we looked into complaints about the current tracking tools for disc golf. Although trackers already exist, they are not built to be durable and are designed to be replaceable, as the user creates technical waste from the trackers breaking down after usage, with no long-term benefit to the player outside of finding their disc. The goal of this project was to consult with disc golfers who play in various course environments for what they would like in a tracker software application, so that as we improve our tracker device and make it more durable and usable, players will have more features at their disposal to use on the course.

Our Client for the project, Marcus Alexander, is an avid disc golf player who often plays various courses and in league games. His desired product aims are a tracker tool that works reliably, provides useful tooling to determine data about his throws, and allows him to compare and determine more accurately what discs he should throw for pin layouts. The application needs to be quick, customizable, and easy to use, so that they don't spend too much time messing with the application.

Focusing on these needs, our team began developing the web application. The solution to the problem was to build a simple yet intuitive GUI paired with a small but kitted GPS microcontroller device with a protective casing to protect the tracker for reusability after harsh impacts. Using NextJS as the base tech stack and TypeScript as the primary language for application frontend components, interface props, and settings adjustments, Golang for server logic, and C/C++ for the microcontroller configuration. Using binary to increase the time between the tracker and the user’s device when sending data packets, giving faster tracked feedback. All of the users' tracked discs and data are collected in databases, for quick data pulls for stats by the user while they play disc golf.

If a user makes a bad throw, they have access to delete (update in the DB) that throw, so it does not impact their session statistics or general ones.

They also have the option to manually save their throws, so only the throws in the session they care about can be saved, saving the time of deleting the throw later if they don't like the autosave feature being enabled. Giving the user the ability to customize how they collect their game data. Focusing on the 80/20 rule, where the 20% of the features that are used more are the tracker to find their discs, and the other 80% features are data collection to benefit the player with on-course decision making if desired.

Webapp link

GitHub Repo

Section 2: Teamwork

Jaden - Hardware, backend, and server developer (Team member 1)

Jaden's contributions focused on making and building the hardware so it can pass data to the GUI components and the database, using Golang for the server-side configuration and setting up the database for user data collection using Drizzle ORM. As well as building the user auth features for accounts using .env configuration.

Nathan - Frontend developer and Intuitive GUI designer (Team member 2)

Nathan's contributions were focused on making the app features customizable and simple. From a high-fidelity Figma prototype, the application was redesigned to be much simpler, using a singular dashboard, settings, and user statistics panels. All of the features are done on the frontend using TypeScript props and state in the NextJS tech stack. Once features are completed on the front-end, they are handed to Jaden for the backend hookups to the server and Database, keeping the integration flow going back and forth from front to backend. Using TypeScript props

and state allows the web app to be customizable by the user, as needed. E.g., specific control over the timing of flight, stat saving, and user statistics. The typography, color palette, and font scaling are mobile-first and follow the guidelines outlined in WCAG AA.

GitHub Task Distribution

Both members created forks from the main development branch; Nathan's fork is focused on frontend development. Jaden's fork is for backend and server-side development. Jaden also has hosted a Vercel webpage on a fork, which is separate from the main development branch in case of breaks from merge conflicts, file curruptions, and unverified pushes to the main fork. When both party members can confirm that the main development branch is ready to be synced, the hosted fork is synced to main, and other branches are only used to resolve potential merge conflicts by using pull requests when necessary.

Section 3: Design Pattern

Our Design pattern for the Disc golf tracking software is a Model-View-Controller architecture. Keeping the versatility of ReactJS, NextJS, and NodeJS for Frontend functionality features, with the backend using a Golang server-side technology. Here's the breakdown:

Model

Database schemas using Drizzle ORM to handle queries, actions, and routing for user statistic data for long-term and short-term usage.

View

React server components and client components, which are located in the app folder, page.tsx, layout.tsx, reusable TypeScript UI components (props and state), Recharts for data visuals, and Tailwind CSS for general styling.

Controller

-NextJS handles the frontend and some backend functionality, such as running on a server (Vercel), handling requests, and calling the database, as well as other services. NextJS also does route handling and server actions, such as GET/POST-related tasks, as well. This serves as a watch to fetch and update features and settings when tasked.

-Golang handles only backend hardware by communicating with the GPS+Accelerometer microcontroller and processing the user's throw data using API endpoints to connect the user's device to the tracker. Also includes logic to translate data faster using binary instead of JSON.

Section 4: Technical writing

User documentation

GitHub/Developer documentation

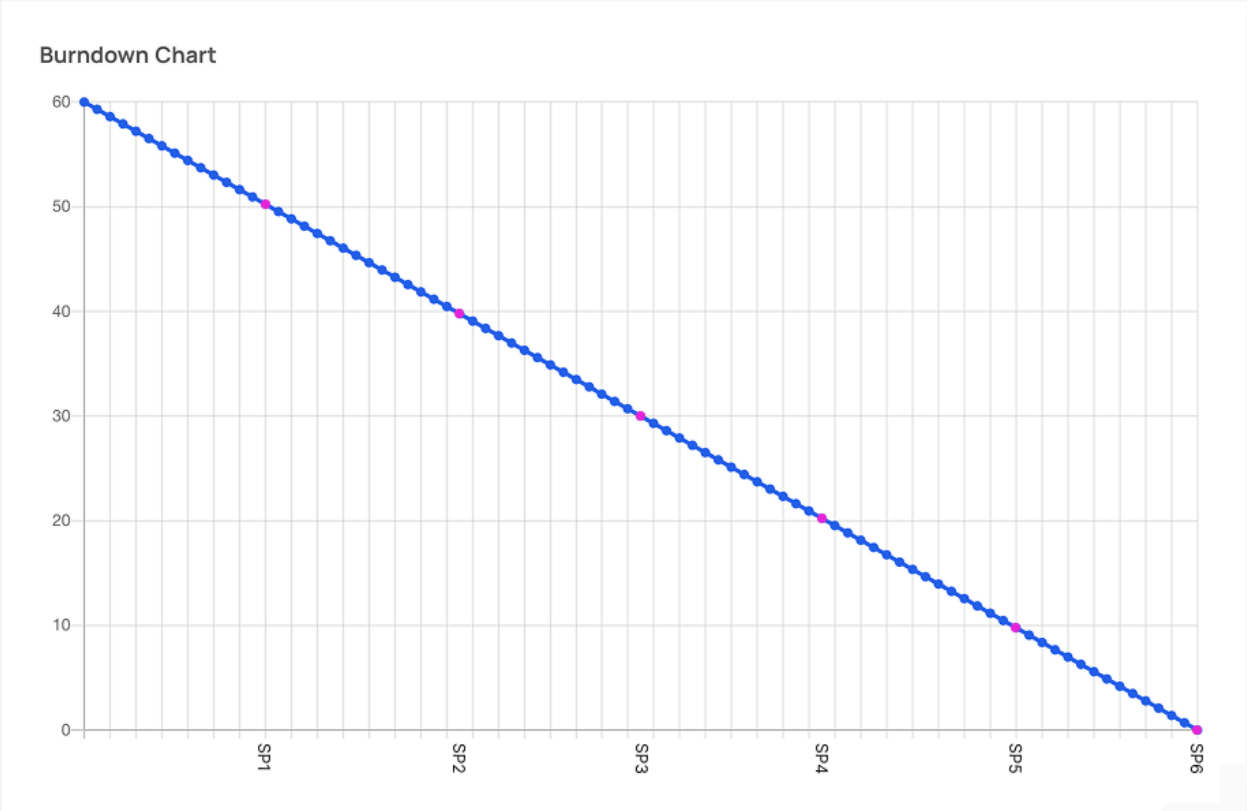
Project Backlog (planned 60pt breakdown):

- Physical prototype component collection (5pts)
- Tracker feature (15pts)
- Prototype assembly and testing (20pts)
- Disc pathing, flight, and diagnostics feature (20pts)

Client information:

- Marcus Alexander - Avid Disc golf league player

  Needs: Looking for a disc golf tracker/tool for games



SPRINT 1 Retrospective:

Goal: For this 2-week sprint, the team will:

- Implement the core tracker UI section from the wireframe prototype (tracker and grid interface for diagnostic data) using React/Next.js/etc, achieving at least 70% visual/style match to the “Tracker” prototype screens.
- Research, finalize, and place an order for all required microcontroller components (GPS module, accelerometer, battery + charger components) with order confirmation. This will unblock backend API development and physical prototype assembly for future sprints.

Sprint Backlog (planned 10pt breakdown):

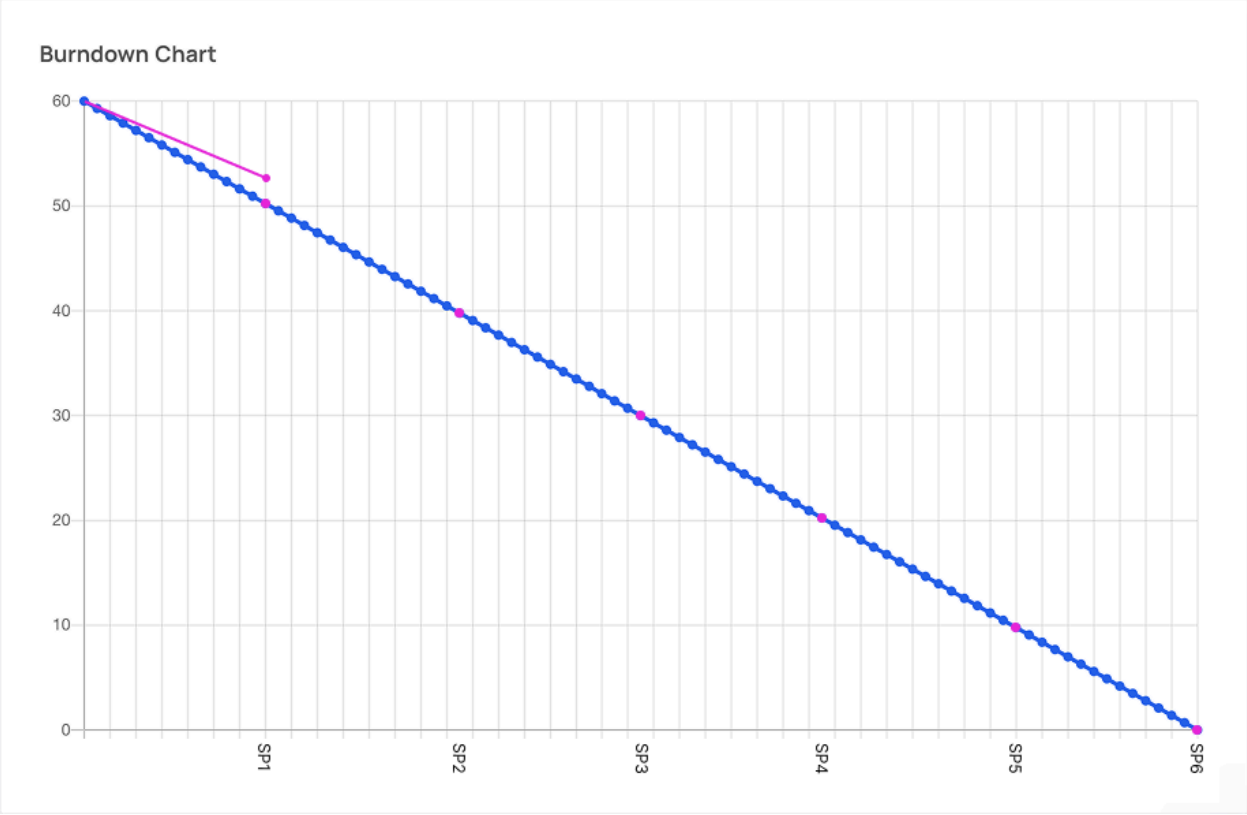
- Review prototype and determine changes for web app production (1pt) -Both group members

- Discuss and order hardware components for the device prototype (2pts)
  -Nathan will provide discs, Jaden is ordering components for the tracker.

- Research Golang and consider options for backend API techstack (2pts)
  -Both group members

- Set up the Next.js project for the frontend (1pt) - Jaden

- Begin building the web app frontend using TypeScript and other needed styling and formatting languages. (4pts)
  -Nathan will focus on styling and UI components, and Jaden will be setting up backend endpoints for the frontend.



Member participation:

Nathan:

Planned Accomplishments: Review the prototype, begin building the frontend, and style it.
Actually Accomplished: Reviewed prototype, built a landing page, updated styling to meet WCAG AA standards.

Blockers: Did not have time to review the Go for the tech stack for this sprint, focused on building a front-end framework this sprint.
Log of Work Hours: 8 hours
What could have been done differently: Could have focused more on organizing the next file structure and working on an icon suite.

Jaden:
Planned Accomplishments: Authentication Backend Setup, Auth Routing Pages, Header, Footer, Project Structuring, Setup Database & Deployment
Actually Accomplished: Auth Setup, Began Auth Routing Pages, Header, Footer, Project Structuring,
Blockers: Time constraints due to travel, need to review documentation before fully implementing the GO backend and embedded system
Log of Work Hours: 6 hours
What could have been done differently: Time management, better documentation for Nathan for file structuring.

Group reflection:
Was the goal met: Yes, we have our landing page and styling figured out. We are now working on improving those details. 8 points for our sprint burndown were less than projected, but due to circumstances and project setup, we are still mostly on track. With it being easy to catch up.
Changes for next sprint: Adding features for the frontend, starting up backend, and device prototyping. Beginning our first feature and interacting with the DB.

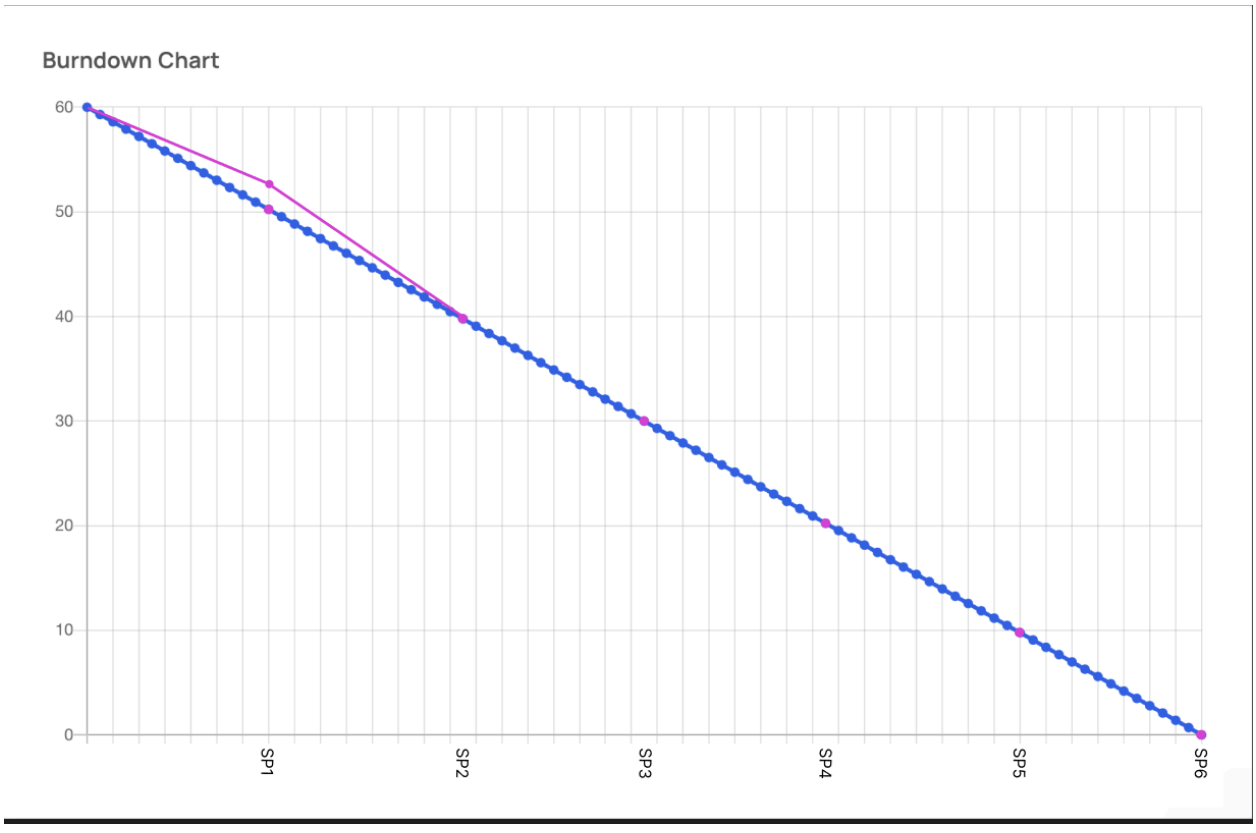
SPRINT 2 Retrospective:

Goal: For this 2-week sprint, the team will:

- Jaden - DB connection, DB schema → Auth and drizzle ORM integration,
- Nathan plans to work on the homepage and incorporate the primary app feature. Reviewing recharts and examining the Stats page GUI options will be determined for future sprints, so that the potential data can be analyzed, displayed, and easily understood.

Sprint Backlog (planned 12pt breakdown):

- 3pts DB setup and config - Jaden
- 3pts Auth and drizzle ORM integration - Jaden
- 4pts homepage/feature 1 GUI (pre hardware setup) - Nathan
- 2pts data GUI prototype and client/user research feedback -Nathan



Member participation:

Nathan:

Planned Accomplishments: Implementing feature 1 on the homepage through hardcoding, and gathering user feedback on the hardcoded GUI.
Actually Accomplished: Homepage completed, Feature 1 hardcode GUI complete, awaiting hardware for implementation changes.
Blockers: Auth errors and dependency conflicts with repo merges.
Log of Work Hours: 7 hours
What could have been done differently: Based on user/client feedback, suggestions included adding an accelerometer option to the GUI for the GPS hardware, which will be implemented in the future. Additionally, a timer delay for manual recording was suggested as a feature for the settings section.

Jaden:
Planned Accomplishments: Complete auth implementation, DB config, ORM config, and implementation. And general backend debugging.
Actually Accomplished: auth implemented, DB setup, ORM configured and implemented.
Log of Work Hours: 7 hours
What could have been done differently: Security could have been improved, less dependencies would help with future security.

Group reflection:

Was the goal met? Yes, the goal was met for all sprint goals have been achieved, putting us back on track on the burndown chart.

Changes for next sprint: None, focus is more directed on hardware, and other GUI for features, as we chip away at it, we can stay on track.

Peer project adimirations (include 2):
-Sports psychology client app: Mood and stress motivation rating seem really useful, especially for initial clients. The GUI appears clean and well-organized.
-ESC Dashboard: I like how it can format presentations.

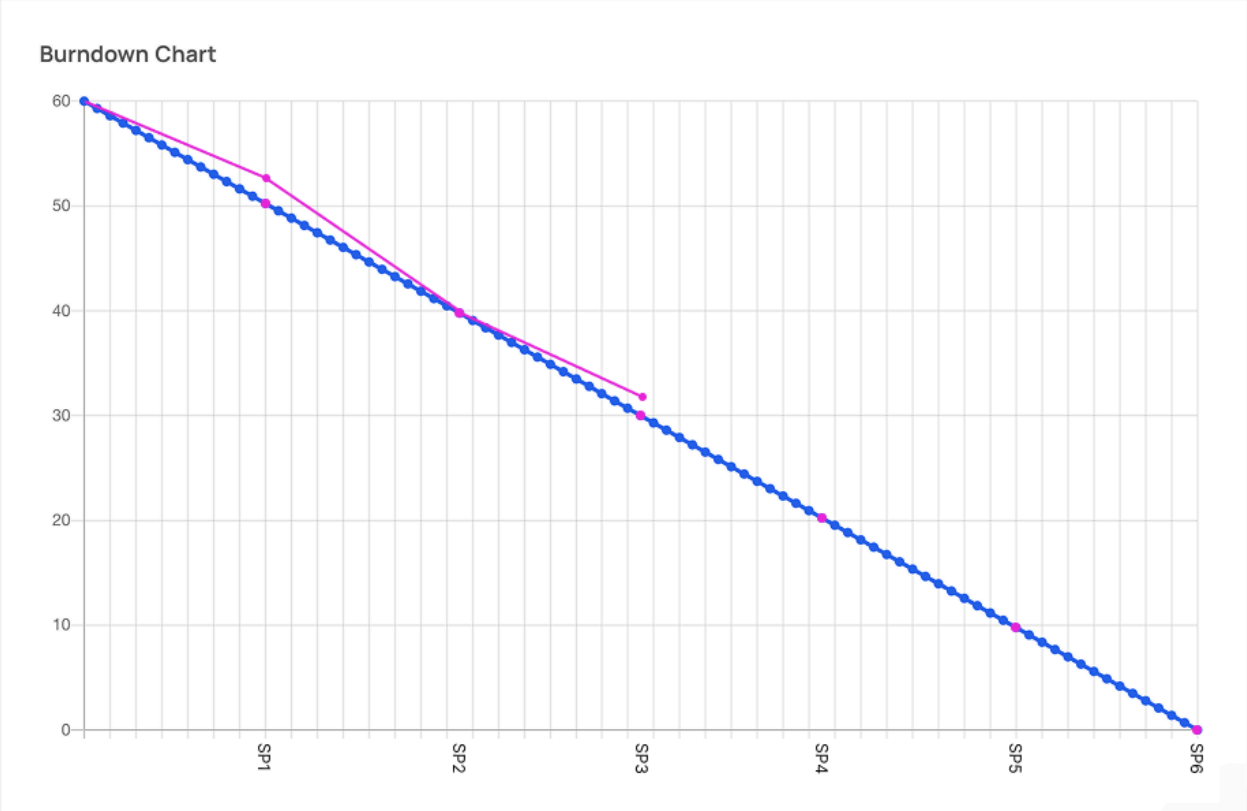
SPRINT 3 Retrospective:

Goal: For this 2-week sprint, the team will:

- Jaden - plans on doing the CAD model for the physical prototype, beginning the hardware assembly, and general debugging
- Nathan - will be working on the settings panels and functionality

Sprint Backlog (planned 10pt breakdown):

- 1pt code debugging and subpages - Nathan and Jaden
- 2pts CAD modeling -Jaden
- 2pts start of Go server and hardware configuration -Jaden
- 3pts working settings features-Nathan
- 2pts settings panel -Nathan



Member participation:

Nathan:

Planned Accomplishments: Planned on debugging and adding settings features and a panel so that they can be set up on the Go server in the next sprint.

Actually Accomplished: Updated documentation to reflect the GUI. Added settings features that affect the dashboard, allowing users to change the information they want to preview. Reworked dropdown features to be individual files instead of one massive file, and added sessions.
Blockers: Waiting on the DB to set up server hookups; data is currently hardcoded.
Log of Work Hours: 10 hours
What could have been done differently: Instead of having to rebuild the whole dropdown features from a single file to multiple, I would've done it separately to begin with.

Jaden:
Planned Accomplishments: Skeleton backed in Go, wrote part of the hardware firmware in C, and made debugging fixes.
Actually Accomplished: Added objects to handle throw sessions, GPS pinging from device to frontend (will be worked on further next sprint). Implemented PostGIS for telemetry data.
Log of Work Hours: 7 hours
What could have been done differently: not write web-encoding in JSON, but instead use protobuf to web sockets.

Group reflection:
Was the goal met? For the most part, yes, we both procrastinated a slight bit, but the general features and framework are present; our only shortcoming this sprint is the CAD model needing to be finished.

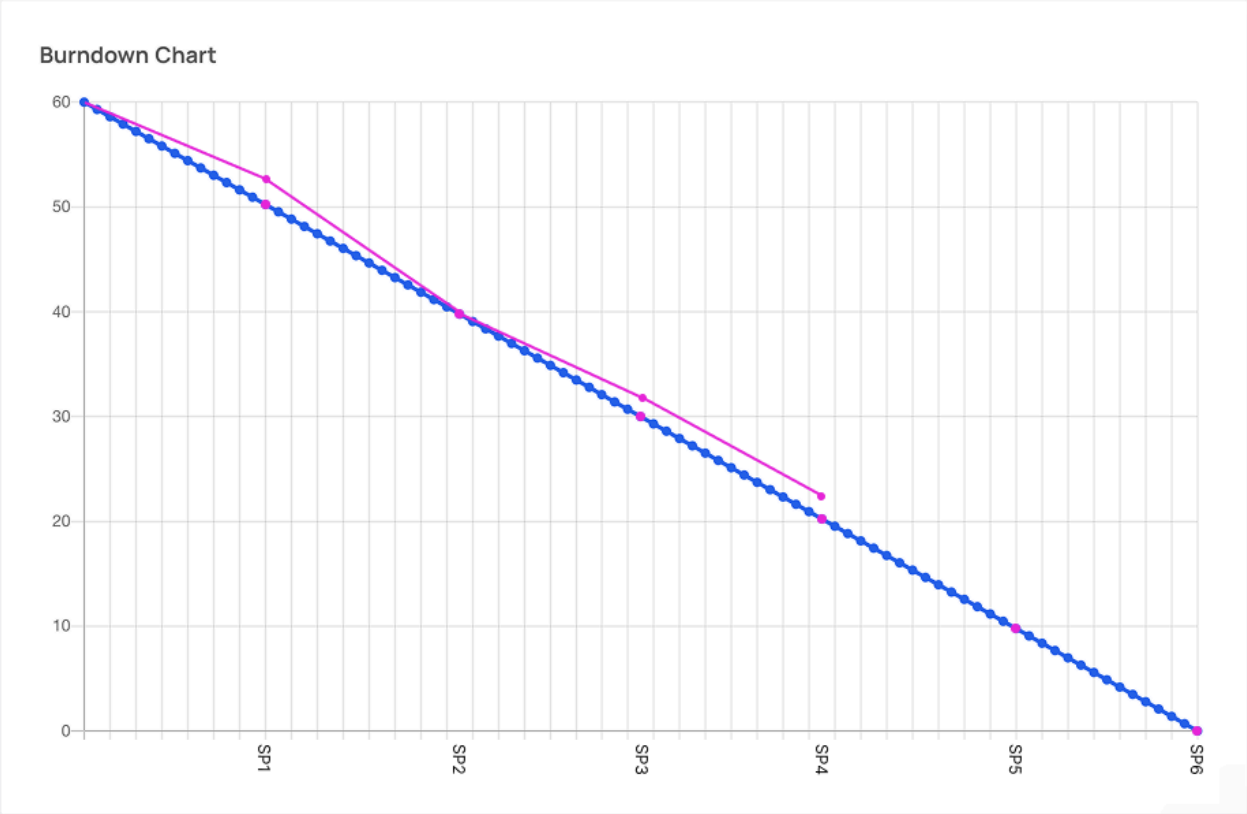
Changes for next sprint: Updating Schemas to handle PostGIS data and Go Server data for the frontend to replace hard-coded data. Going from JSON encoding to protobuf encoding for WebSocket.

SPRINT 4 Retrospective:

Goal: For this 2-week sprint, the team will:

- Jaden - connect database schema to frontend, continue hardware assembly.
- Nathan - annotating frontend for backend handoff, start statistics page.

Sprint Backlog (planned 11pt breakdown):
- 2pt CAD model design and hardware assembly - Jaden
- 2pts Go server setups for frontend hookups - Jaden
- 3pts postGIS database implementation to handle the Go implementation - Jaden
- 2pts annotations in typescript for backend hookups to Go Server - Nathan
- 2pts Begin implementing the Statistics page - Nathan



Member participation:

Nathan:

Planned Accomplishments: Feature Debugging and handoff notes for the feature hookups for backend (GO server), begin statistics page, 3D print refined prototype.

Actually Accomplished: debugged features, annotated files for backend handoff, and refined the add disc feature to take another prompt, printed V4 TPU prototype of the tracker housing from the original PLA model.

Blockers: Waiting for the full backend hookup before building the user stats page

Log of Work Hours: 8 hours

What could have been done differently: using pull requests to improve communication of what needs to be done between front and back end. For organizational purposes

Jaden:

Planned Accomplishments: CAD model, Go server hookups to frontend, PostGIS database setup, microcontroller, and device prototype hardware versions 1-5, initial PLA prints for hardware housing.

Actually Accomplished: Cad model is done and is now being refined in smaller iterations for usability improvements. Hardware housing is done and is being refined as well to be lighter and smaller to accommodate potential hardware issues. The PostGIS database is mostly set up,

with roughly 90% done; the next sprint will be adding Timescale DB as an extension for handling telemetry data. GO server hook-ups are mostly done for core features. The microcontroller fits the housing and is currently being configured.
Blockers: Firmware conflicts, pivoting from firmware to software, and general debugging.
Log of Work Hours: 15 hours
What could have been done differently: pull requests for handoff, initial firmware implementation could have been done differently.

Group reflection:
Was the goal met?
Yes, the backend-to-frontend software is mostly set up, and housing and hardware have been assembled and are being refined for the improvement of usability. The user stats page will be set up after the frontend and backend are both completely hooked up together.

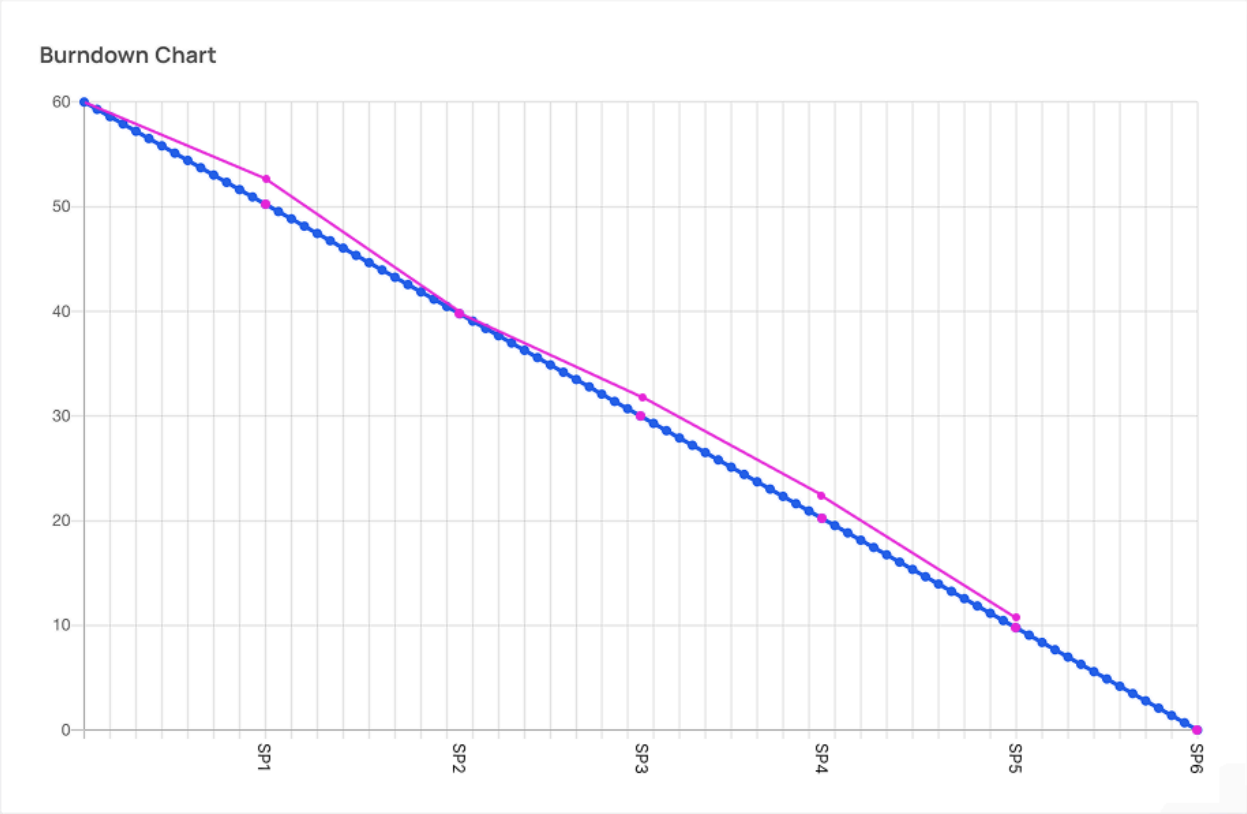
Changes for next sprint: Tying backend loose ends, refactoring, and debugging firmware. Adding pull requests to handoffs.

SPRINT 5 Retrospective:

Goal: For this 2-week sprint, the team will:

- Implementation of backend hookups to the GO server - Jaden
- Prototype assembly - Jaden
- User Stats page build - Nathan
- Print prototype casing using TPU - Nathan

Sprint Backlog (planned 13pt breakdown):
- 6pts Backend hookups to frontend for the GO server. -Jaden
- 4pts User stats page assembly and configuration. -Nathan
- 2pts hardware assembly
- 1pt hardware casing print



Member participation:

Nathan:

Planned Accomplishments: User Stats page, front-end configuration ready for backend handoff, and hardware casing 3D printed.
Actually Accomplished: User stats page and Directional tracking UI added at the client's request, and are handed off for go server connections. The 3D printed casing is done.
Blockers: No blockers this sprint had more than what was projected.
Log of Work Hours: 5 hours
What could have been done differently: more frontend unit testing, only testing 2 users could have done more.

Jaden:

Planned Accomplishments: frontend to backend GO server hookups, and hardware assembly
Actually Accomplished: All features are hooked up to the backend GO server. Web socket debugging with APIs, middleware debugging. Hardware waiting to be soldered.
Blockers: Busy sprint with other course tasks going on.
Log of Work Hours: 9 hours
What could have been done differently: Not vibe-coding the backend hookups, wasted time trying to save time. Had to be done manually anyway.

Group reflection:
Was the goal met? Mostly the Goal met. The only thing that fell short of this sprint was wiring the hardware and testing it. Frontend testing had a few users and received good feedback for future updates, as well as implementing directional tracking this sprint as an extra feature suggested by the client.
Changes for next sprint: more focus on testing once the hardware is fully wired, tidying bugs, and design to prepare for release.

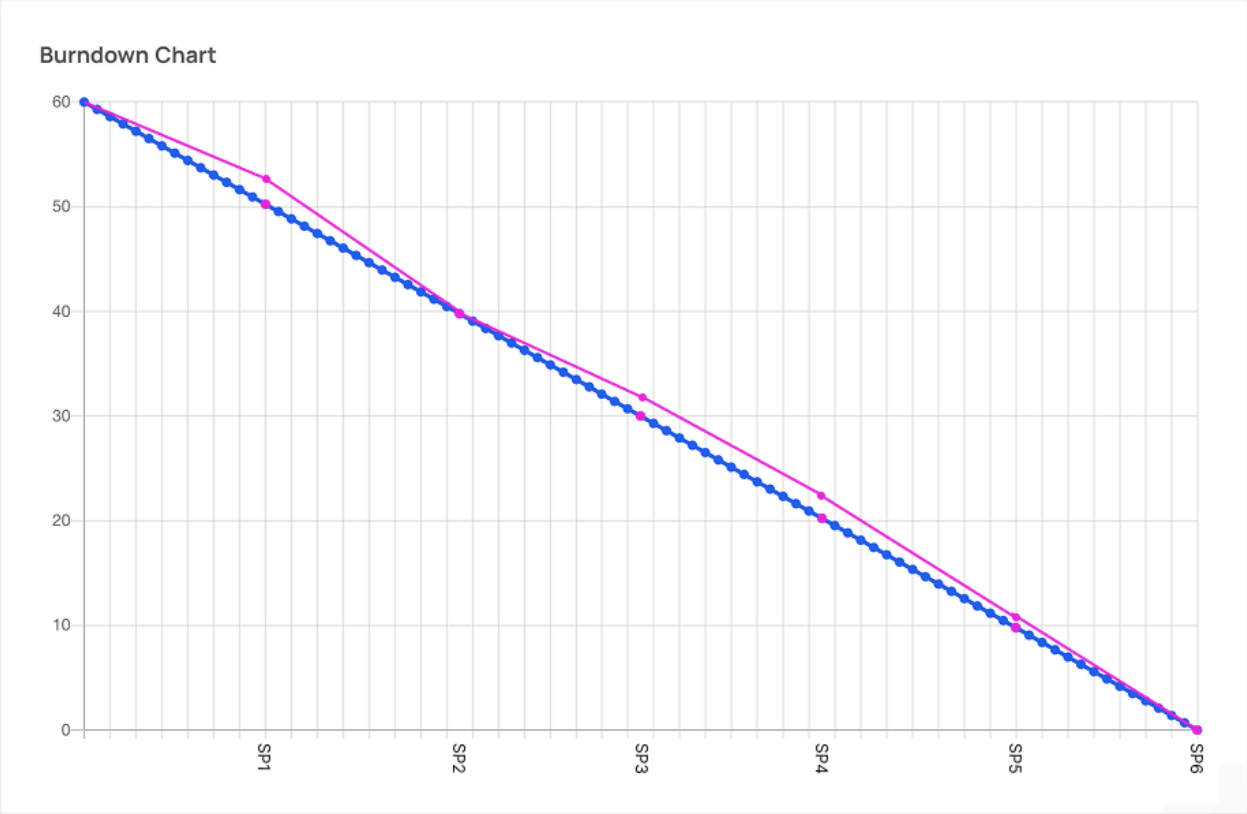
SPRINT 6 Retrospective:

Goal: For this 2-week sprint, the team will:

- Finish hardware hookups to the backend and debugging - Jaden
- Clean up UI following backend hookups - Nathan
- General debugging and testing of hardware - Both members

Sprint Backlog (planned 11pt breakdown):

- 6pts hardware hookups - Jaden
- 3pts UI clean up - Nathan
- 2pt general debugging and hardware testing - Both members



Member participation:

Nathan:

Planned Accomplishments: Planned to debug any final frontend GUI bugs and add user documentation to the dashboard page

Actually Accomplished: Finished debugging front-end GUI and added dashboard page to Dashboard.

Blockers:

Log of Work Hours: 3 hours

What could have been done differently: Updating user docs before adding them to the Dashboard would've been a better move to save a little time

Jaden:

Planned Accomplishments: Getting the Hardware device to work with the backend and frontend fully

Actually Accomplished: backend and frontend are configured, Dockerfile will be pushed to Vercel, and hardware should be fully functional.

Blockers: DB giving server errors, GitHub push reverted work, needed to play catch-up, and datastream issues with hardware, too much crashing of the datastream.

Log of Work Hours: 8 hours

What could have been done differently: Datastream decoding could have been better handled to prevent crashes, stashing backend changes before GitHub pulling.

Group reflection:
Application is complete, hardware prototype works, but will need to be improved for another interaction using a more advanced microcontroller to give more advanced and longer distances feedback. Datastream is now fixed, so the application should pull all of the data from the hardware into the software.

Final Release Images and Descriptions:

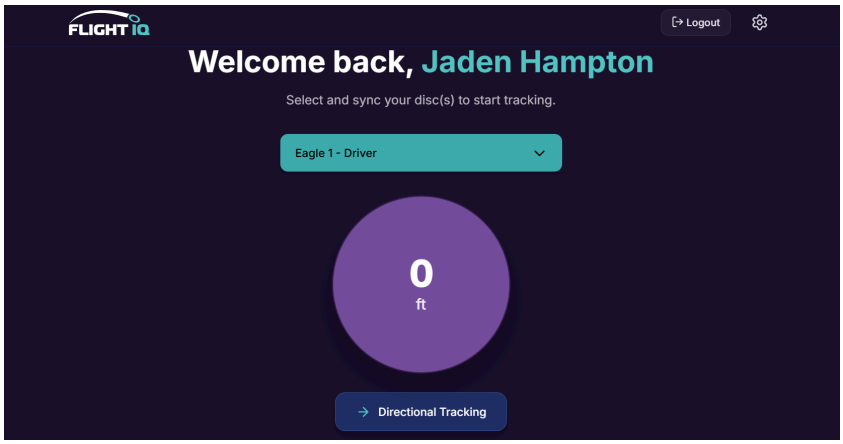
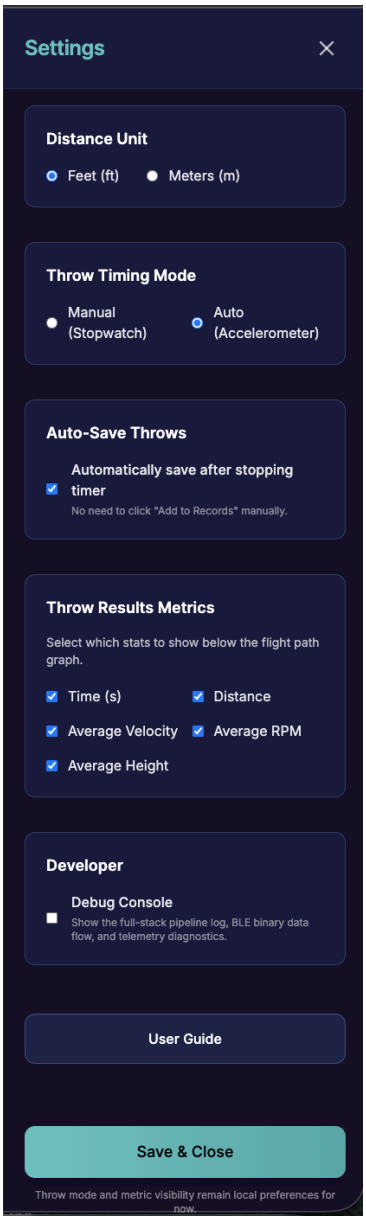


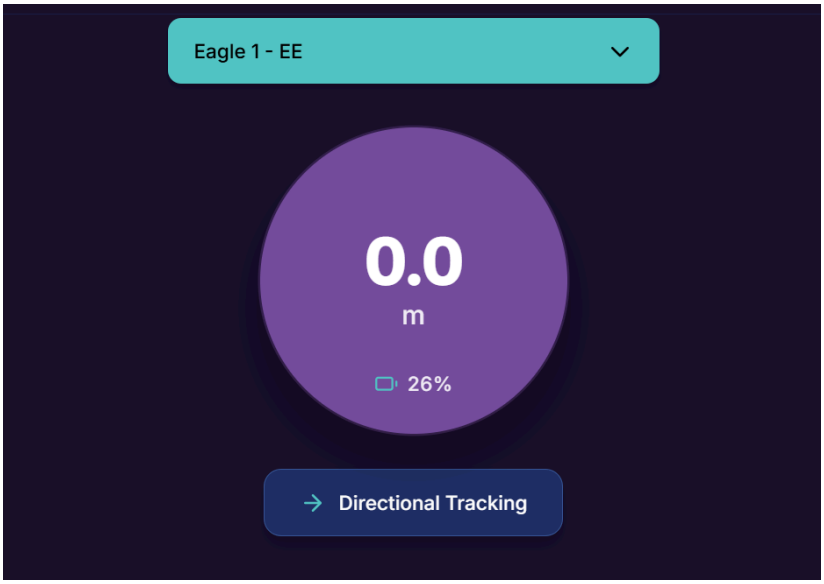
Image of a connected disc and tracker display that provides distance from the tracker device



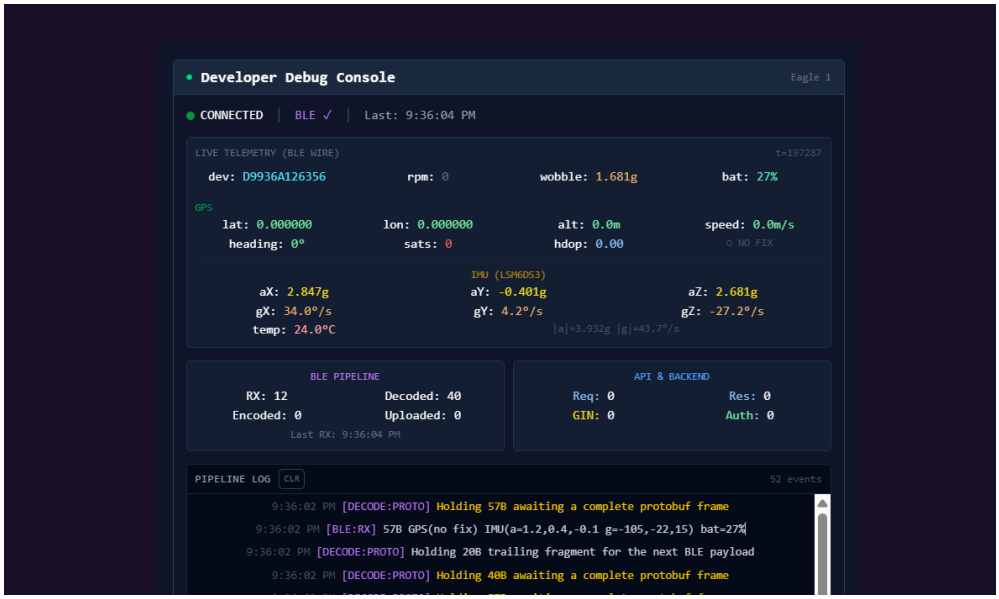
Throw statistics overlay that shows users' throw data, which can be used to see top-down flight paths, determine distance and velocity, and compare and contrast throw data across sessions.



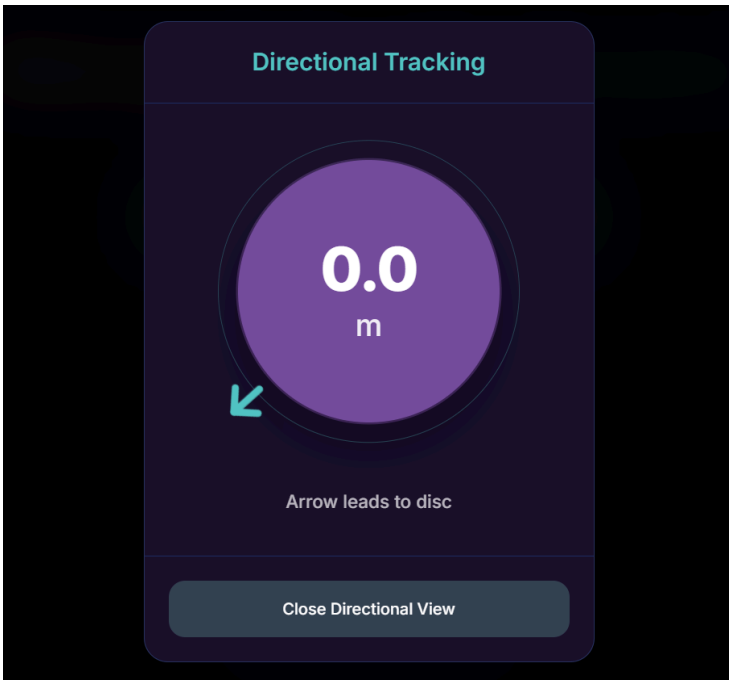
The settings panel, which allows the user to customize how they collect and display their disc flight data. The feature also includes a debug console for hardware troubleshooting and a user guide (user docs) for help.



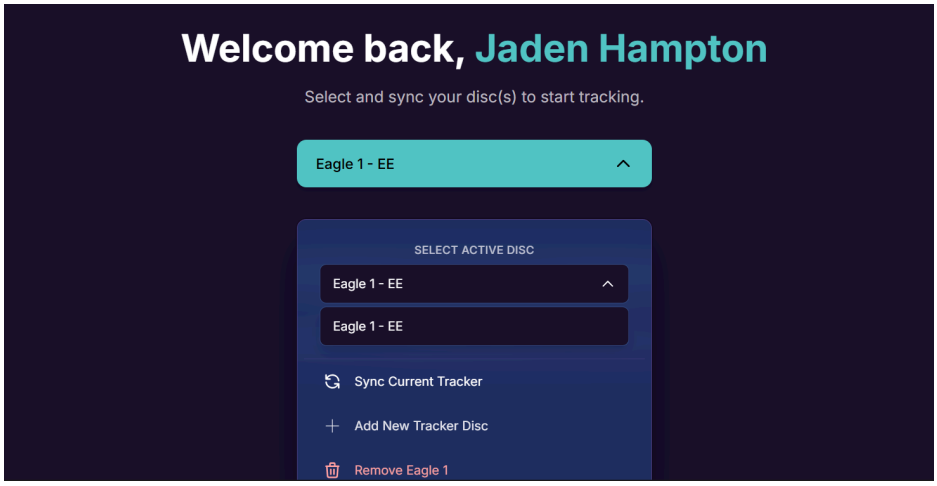
Tracker dot provides distance and battery percentage. For user guidance and device feedback.



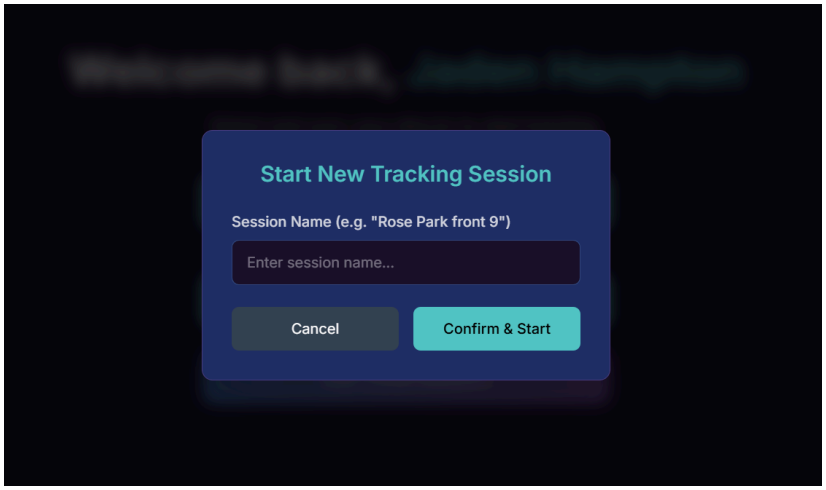
Developer debug console for troubleshooting and ensuring device hardware connectivity.



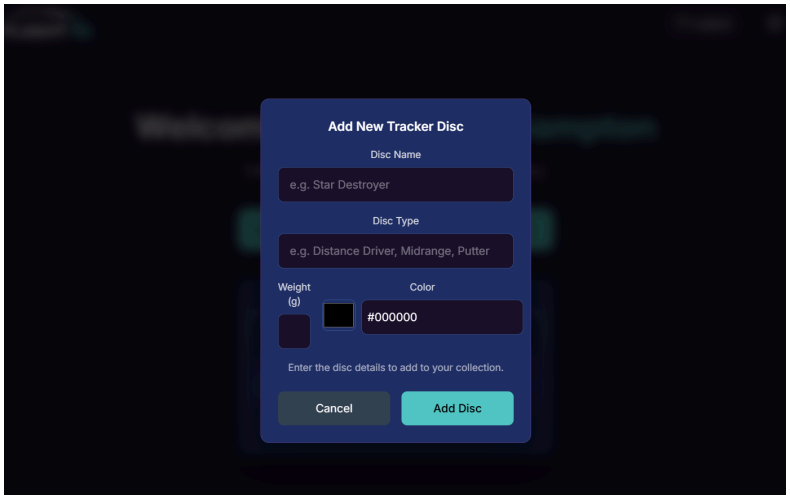
Directional tracker overlay used for enhanced disc searches by the user.



The dropdown actions menu is used to help select, sync, add, and delete discs.



Start a new session pop-up.



Add a new tracker device pop-up for adding a new tracker disc to the database.

Section 5: UML

[UML chart]

Section 6: Design Tradeoffs

The application does come with some tradeoffs in its current configuration:

Simplicity over scalability - Prototype obsolescence

Due to the need for quick on-course access, the features introduced in the high-fidelity prototype were reduced. Narrowing down to the app's core

functionality and the client's product need, which is being a tracker for discs and helping the users determine the best throws for the courses they play. The original prototype had features like tips and tricks, as well as a disc bag. For this adaptation, the extra features seemed unnecessary for on-course play, which is why the features were removed. Scalability can still be achieved using user stats to add more visual information about throws as application development continues. Outside of the core functionality, however, it should not change more than that. These changes needed revision and have made the original prototype much different, which is why it is not overly relevant.

NodeJS vulnerabilities

Adding more dependencies creates more vulnerabilities, according to the NodeJS terminal feedback. So our team tended to avoid adding too many NodeJS dependencies in order to avoid more vulnerabilities and security risks. This tradeoff is essential if we want to keep our end-user satisfaction long-term. Unfortunately, this may impact scalability as well. Plans to switch over to HTMX for future development are being considered for better usability, reduced vulnerabilities, and scalability.

Webapp not IOS or Android

The application is built for the web for two reasons: we don't want to pay for IOS access at the current development stage, or use Android software for our release build, because it can limit the amount of user access for releases early on. The web is the best option since it is the most accessible platform for all users. Our future goal is to focus on making it work in areas that lack cellular access. To improve play in remote locations, if that means switching to IOS and Android mobiles, we will cross that bridge when it arises.

Section 7: Software development Lifecycle model

For this project, we chose an agile method, a Continuous Integration and Continuous Delivery (CI/CD)- aligned software development lifecycle, with

extra steps to ensure the build always works on the hosted page. Because both sides of the project, Jaden and Nathan, have their working forks where they can merge changes to the main branch. Each member can pull from there to ensure changes work for both parties whenever changes are made using automated tools on GitHub Actions, by checking for merge conflicts and verifying pushes to catch bugs before they become problems.

This is so the build does not get synced to the Vercel-hosted fork until both parties can ensure that the merged main branch is functional. Since the hosted branch is always active. All that needs to be done when the changes are made in main is to sync the hosted branch with main. Then the site is built and up to date automatically.