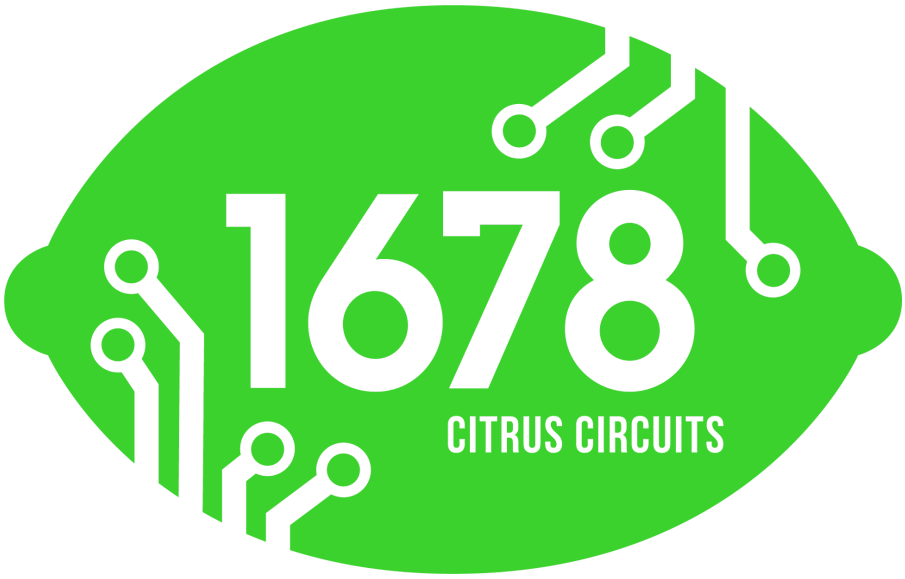


2025

SCOUTING WHITEPAPER

Table of Contents

Introduction

Navigating This Whitepaper

Over the years, our annual Scouting Whitepaper has evolved significantly, expanding from 7 pages in 2013 to 106 pages in 2025. This section provides some guidance in navigating this large document based on your specific needs.

The Whitepaper begins with an introduction to our subteam and system and covers our history and major updates since last year. This section offers a high-level overview of the goals and structure of 1678 Scouting.

Each main section of the Whitepaper provides technical documentation for different parts of our system, such as apps, data processing systems, or APIs. These sections start with an overview of each component's purpose and methodology, followed by more detailed subsections.

Given the Whitepaper's length, we recommend reading it in a specific order based on your needs:

- **If you're interested in building your own app, data processing server, or scouting system**, use the Table of Contents to locate the relevant section, review the source code in the 2025 Public GitHub Repositories, and explore our guide on Starting Your Own Scouting System. You may also find useful insights in our Old Whitepapers or Fall Workshops recordings.
- **For an overview of our management structure**, which coordinates 20+ students building code and 25+ students scouting at competition, start by reading the brief introductions to each section to gain an understanding of our goals, resources, and requirements. Then, visit the Subteam Management section for details on our administrative processes.

- **For a complete understanding of 1678 Scouting**, you can simply read the document sequentially. There are quite a few pages though, so it might take a while!

If you have any additional questions not covered in the Whitepaper, feel free to reach out to us at 1678scouting@gmail.com.

History

Citrus Circuits' electronic scouting system was first developed and used during the 2013 FRC season of Ultimate Ascent. The team had previously used a paper scouting system but was overwhelmed by the management issues that arose when scouting 100 teams in its division at the World Championship. A software development team of four students created the original system, which has since grown into a full subteam of 21 students with two different components and six apps.

The system was structured in 2013 with two relatively unique attributes: (1) collection using Android tablets for both objective scoring—each focused on a single robot—and subjective ordinal rankings of robots' driving strengths and abilities within each match alliance, and (2) delivery of collected and processed data to a cell phone app in near real-time for use by the drive team to prepare for matches. The objective and subjective data were quantitatively combined using weighted scoring to provide two ranked priority draft lists: one for the first pick and one for the second pick during alliance selection before the playoffs. The system maintains that basic structure today, with additional attributes described further in this white paper.

The system proved successful in its initial use at the 2013 Central Valley Regional, when 1678 fully assembled an alliance based on collected scouting data to win the event from the 6th seed. The system again proved its worth at the Championship, where 1678 won the Curie Division using a well-noted drafting strategy that prevented the top teams from allying together. The entire system ran with eight Scouts and two programmers, hard-wired together through a Raspberry Pi.

The scouting system has evolved in several ways, addressing many issues along the way. Initially, the system was wired together but switched to Bluetooth communication in

2014 to avoid the prohibition on Wi-Fi while increasing the number of available ports for tablets. The first significant revision occurred in 2015 with the first multi-object game presented by FIRST. However, the scouting system failed to provide usable data at the first event of the year, although it was up and running later in the season. An opportunity to prepare a picklist with 118 Robonauts at Championships led to significant advances in data collection and presentation. These advances included further development of the Picklist Editor, added visualization in both the Viewer and Picklist Editor, and deeper use of the editor during the meeting. In 2016, the Software Scouting team created a project management schedule that delivered substantial improvements to the system. Another app was added in 2016 to collect specific data from pit visits. In 2017, the crew of Scouts was expanded, assigning three Scouts to each robot for collecting objective data to improve accuracy. In 2018, Bluetooth became less reliable due to the prevalence of smartphones at events, and Match Collection was updated to use QR codes to transfer data.

In 2020, the Match Collection, Pit Collection, and Viewer apps were rewritten in Kotlin for Android phones. Support for the iPhone Viewer was dropped, as fewer students were familiar with the Swift language. The team acquired Google Pixel phones for drive team members to replace iPhones.

The system has relied on several online database programs, first adopting Firebase and later moving to MongoDB in 2020. At the first competition in 2020, the system did not deliver data, but the COVID-19 pandemic terminated the season before further updates could be made. In 2021, the subteam continued to meet remotely and worked on improving the structure of the 2020 system without needing to focus on a specific game. The 2022 system resulted from substantial improvements made since the Kotlin rewrite in 2020, and those revisions set the foundation for future features. In 2023, one major change was switching the web server communication interface from Cardinal (described in earlier white papers) to Grosbeak.

In 2025, major projects were completed on the back-end side, focusing on modernization and data quality. Kestrel, a much simpler and more robust web API, was created to replace Grosbeak. A new data validation system was implemented to address problems with scouting

accuracy at our first regional. DoozerNet, a new AI match prediction ecosystem, has been developed and remains a major active project going into 2026.

Subteam and System Overview

Students in Software Scouting are split into either Back-End or Front-End. Each End has a student lead, chosen by the previous year's team captains and head mentors. Back-End students code in Python and are responsible for the Server, which manages the database and handles calculations. Front-End students mainly use Kotlin to create all the apps that users interact with. Each End requires members to specialize in specific programming knowledge and concepts. However, Software Scouting is still considered a single subteam, and students from both Ends regularly participate in full subteam discussions and system tests. Some students also occasionally write code for the other End. Veteran members guide newer members, work with them on tasks, and teach them important concepts. The Software Scouting subteam works closely with the Strategy subteam, as we formulate strategies using scouting data, and most Scouting members are also part of the Strategy subteam.

The Citrus Circuits scouting system is divided into three main stages: collection, processing, and visualization. The collection stage includes the Match Collection app, the Pit Collection app, and the Stand Strategist app. During the processing stage, the Server organizes the data and runs calculations, which the Viewer app and the Picklist Editor then visualize by pulling the data through Grosbeak. At competitions, scouts use three main collection apps: the Match Collection app, the Pit Collection app, and the Stand Strategist app. The users of the collection apps are as follows:

- **Match Collection (Objective)**: Scouts
- **Match Collection (Subjective)**: Subjective Scouts
- **Match Collection (Playoff Scouting)**: Selected Scouts
- **Stand Strategist**: Stand Strategists and Subjective Scouts

- **Pit Collection (Objective)**: Pit Scout (Match Strategist with assistance from Citrus Seals)
- **OverRate**: Selected Scouts and Strategists

Summary of Major Changes Since 2024

2025 has been a year of innovation, with new projects and a reorganization of existing systems. We haven't been afraid to delete parts of our system and replace them with something better, and we've spent a lot of time expanding into areas we've never touched before. Specifically, we:

- Replaced Grosbeak with Kestrel, a new lightweight, flexible, and robust data transfer API connecting MongoDB with the Viewer and Pit Collection apps
- Created a data validation and override system to monitor and correct scouts' inaccuracies at competition
- Developed a new AI match prediction network achieving 90% accuracy
- Created a new TBA and Statbotics communicator system to pull global match and team data
- Added multiple new quality-of-life features to the Viewer
- Created a "Simplified Match Collection" app to scout in-shop drive practices and competition playoff matches
- Added click-through animations to Viewer's Auto Paths page
- Modified the QR Details screen in Objective Collection, which now provides a summary of intaking and scoring data after every match
- Added new features to Overrate, such as the parallel leaderboard

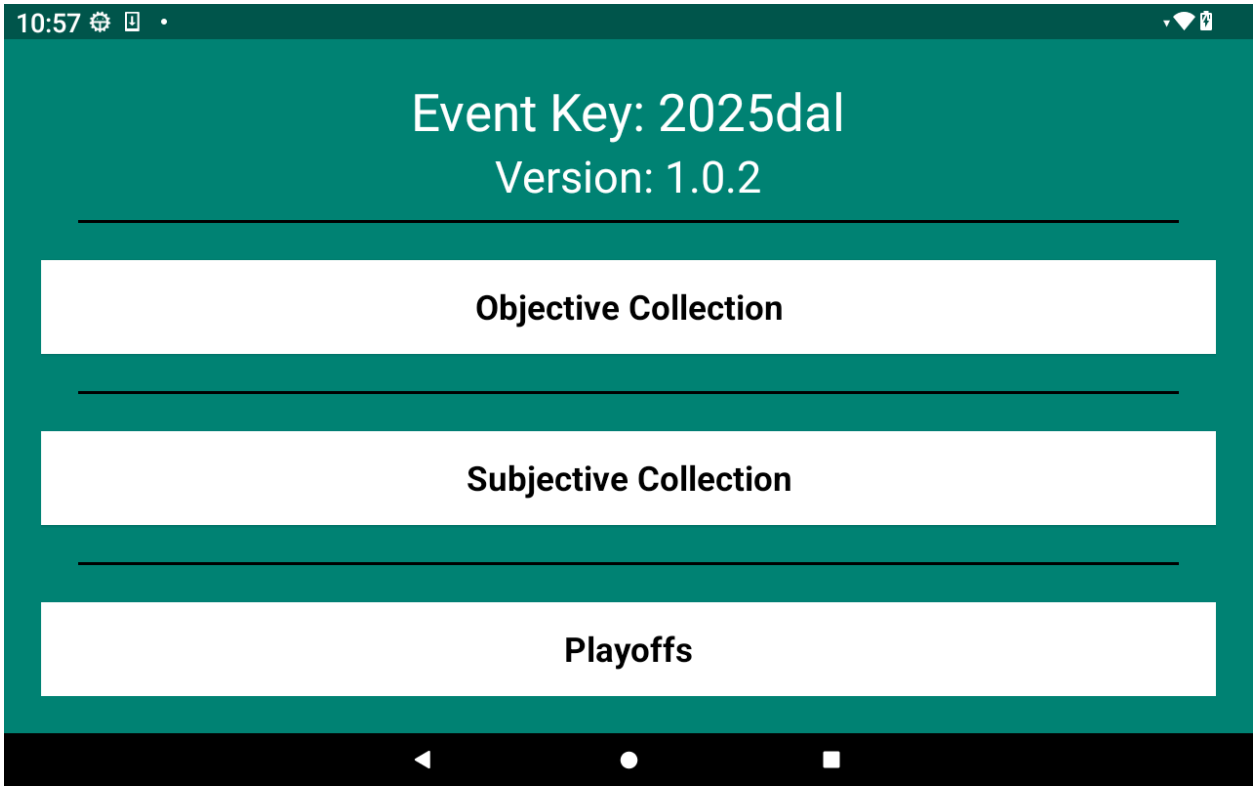
Match Collection

Scouts in the stands use the Match Collection app during matches to collect both quantitative and subjective performance data on robots. The app runs on Lenovo Android tablets and was developed using Android Studio with Kotlin and XML. It has three modes: Objective Collection, Subjective Collection, and Playoffs Scouting. During each qualification match, there are 18 Objective Scouts (three per robot) and 2 Subjective Scouts (one per alliance).

- Objective Collection gathers quantitative data such as a robot's scoring and intake statistics.
- Subjective Scouts use Subjective Collection to rank the performance of each robot in an alliance qualitatively (for example, the field awareness between the robots on the Red alliance).
- Playoffs Scouting allows Scouts to record Net, Processor, and Reef scores during playoff matches.

Starting Screen

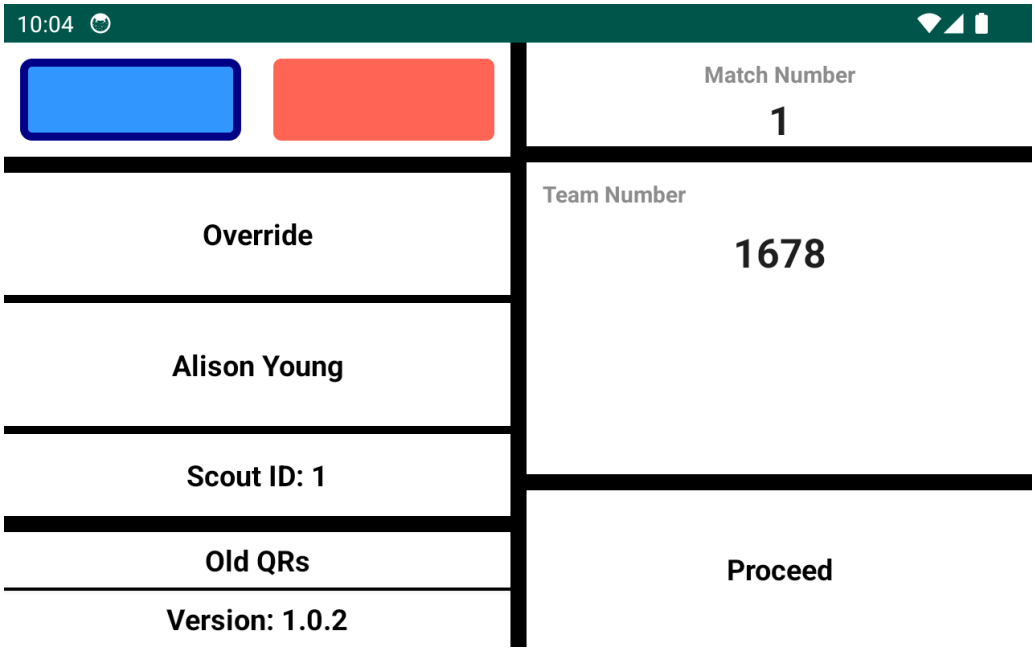
The Objective/Subjective/Playoff starting screen is the first screen in the app. From here, the user can enter one of these three scouting modes by selecting the corresponding option.



Objective/Subjective/Playoff screen

Objective Collection

Match Input



Match Input Screen

The Objective Collection Match Input screen contains several elements. First, the Assignment button allows the user to set the assignment to either Automatic or Override. Automatic Assignment pre-sets the team and alliance color being scouted. In contrast, selecting Override lets the scout change the alliance color and team number they are scouting, as well as override some features later in the app. Scouts can also edit the match number, scout name, or scout ID on the input screen. Additionally, an “Old QRs” button enables scouts to view all past QR codes in case they forget to scan the QR code after a match. Finally, scouts use the Proceed button to transition to the Objective Collection Starting Position screen.

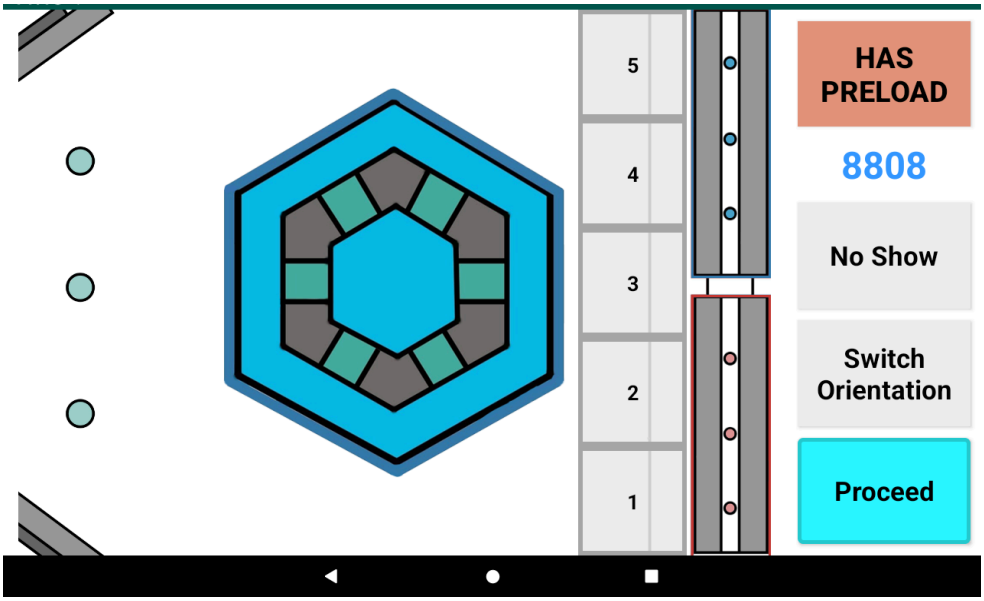
Randomized Team Assignments

Match Collection randomizes team assignments for different Scout IDs to minimize the concentration of errors that scouts may make on a few robots. These random assignments are pulled from a file resource containing 100+ randomized orders of Scout IDs generated ahead of the competition. A group of three Scouts will watch a single robot, and then all three groups will shuffle for the next match. This feature ensures that a scout will scout different robots throughout the competition, helping prevent data from becoming inaccurate due to inexperienced scouts consistently scouting the same teams. It also allows us to compare scout performance (see Scout Precision Ranking (SPR) in the Server) and better validate our data against The Blue Alliance.

Data Collection Screens

This year, we implemented a hybrid map scouting feature for our Objective Match Collection app. This app structure creates a map of the field, but the buttons are not necessarily located at corresponding positions on the map. This implementation allows for better use of screen space while providing a visual reference for Scouts.

Starting Position



Objective Starting Position screen

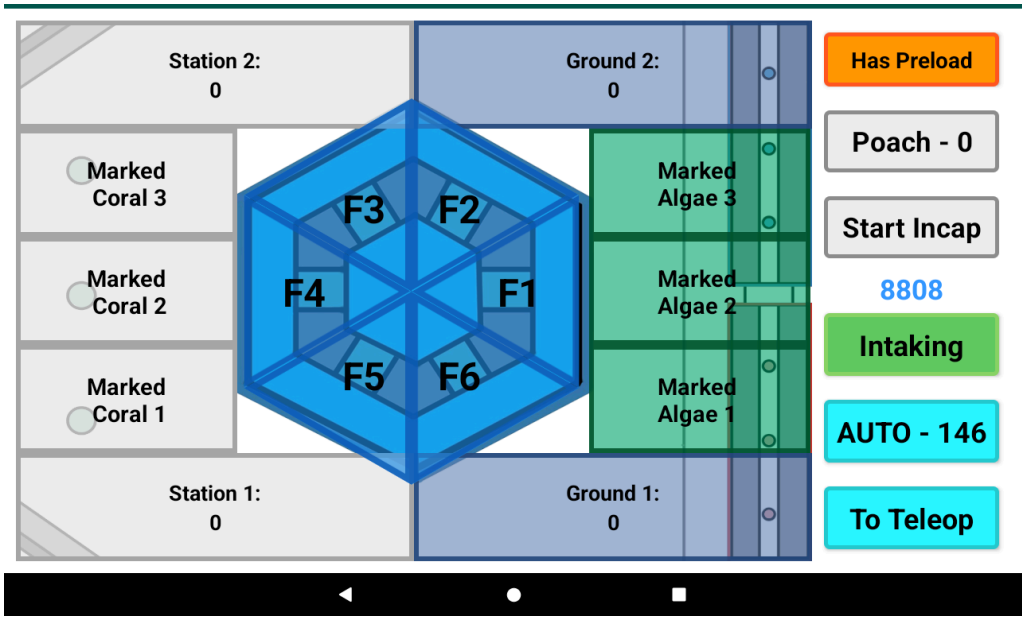
On the Starting Position screen, Scouts must input the robot's starting position by pressing one of the five starting position buttons. If the assigned robot is not on the field, the Scout presses the "No Show" button. There is also a preload button, which Scouts use to record whether or not the robot has a preloaded Coral. Scouts can use the "Switch Orientation" button when they are not in an optimal location in the stands and need to adjust the app's orientation to match their view of the field. The data collected from the Starting Position and Auto screens is used to formulate auto paths.

Auto

The Auto screen displays a diagram of the alliance's side of the field, with a different set of buttons depending on whether the team has a Coral or an Algae. While on the Auto screen, users can long-press the timer to reset it. This feature is disabled once any action is input.

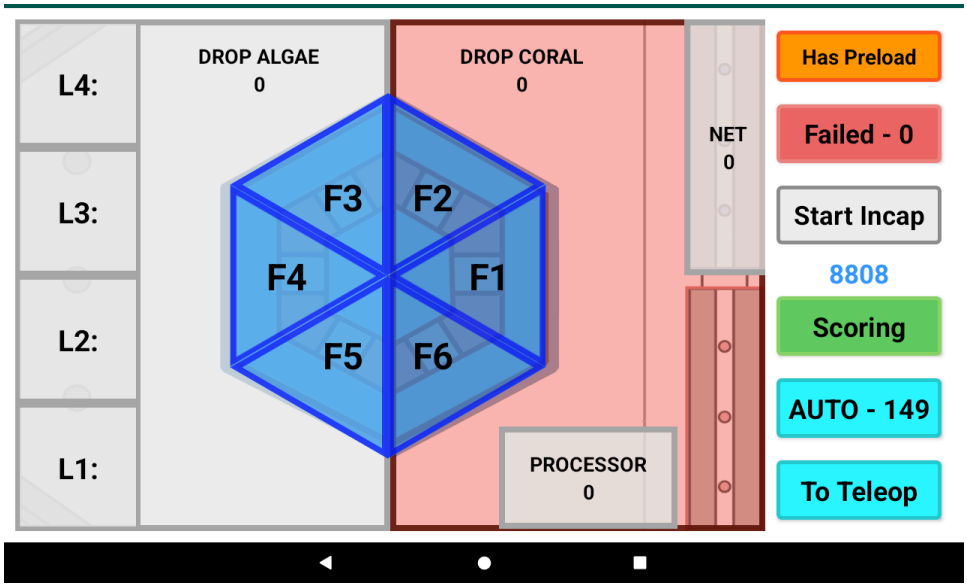
Scouts can change whether the robot has a preloaded Coral as long as no actions have been taken other than switching between Auto, Teleop, and Endgame. This feature is helpful in cases where it is difficult to tell if the robot has a preload, allowing the scout to quickly update it. This button replaces the Undo and Redo buttons until the user performs another action.

Due to the nature of the reef, we created six individual triangle-shaped buttons that, together, form a hexagon. Having six separate buttons presents both advantages and challenges. On the positive side, it provides Strategists with insight into which face of the reef a robot intakes and scores from during Auto. On the downside, it increases the app's complexity, which Scouts may not prefer. Ultimately, it was decided that the benefits outweighed the added complexity, and the hexagon was implemented.



Auto Intake

The Auto Intake screen has 16 buttons for recording intakes: six for each pre-placed Algae on the reef, three for each pre-placed Coral on the field, three for each pre-placed Algae on the field, and a pair of ground and station buttons for each Coral station. Collecting the locations of intaken Algae and Coral helps us track the auto paths teams used.



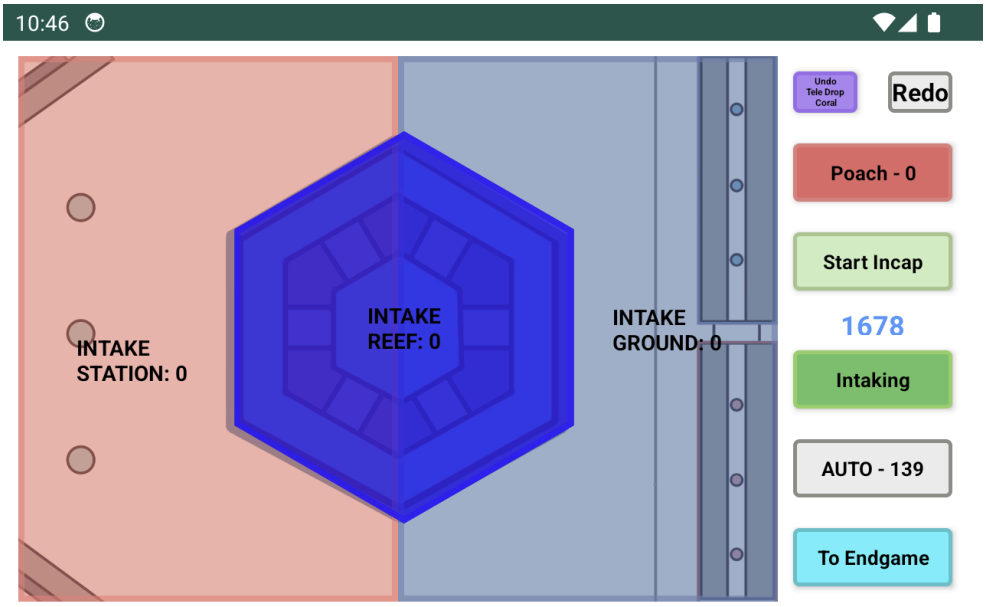
Auto Scoring

The Auto Scoring screen consists of 16 buttons: one for each scoring location of an Algae (Processor and Net), six for each side of the Reef, four for each level on the Reef (L1 being the lowest, L4 being the highest), two for dropping either a Coral or an Algae, a Fail button (e.g., scoring path: F5, L4), and (new this season) a button to manually switch between the scoring and intake screens (due to the ability to hold both Algae and Coral simultaneously).

A feature this added year for the Auto Scoring screen is the ability to track game elements recorded by Scouts. This allows us to gray out and disable/re-enable buttons corresponding to the game elements that have been intaken (e.g., if a Scout intakes a Coral from Station 2, all intake and scoring buttons related to Coral are grayed out). Graying out buttons provides visual feedback to Scouts, helping them keep track of the actions they've completed while scouting a match.

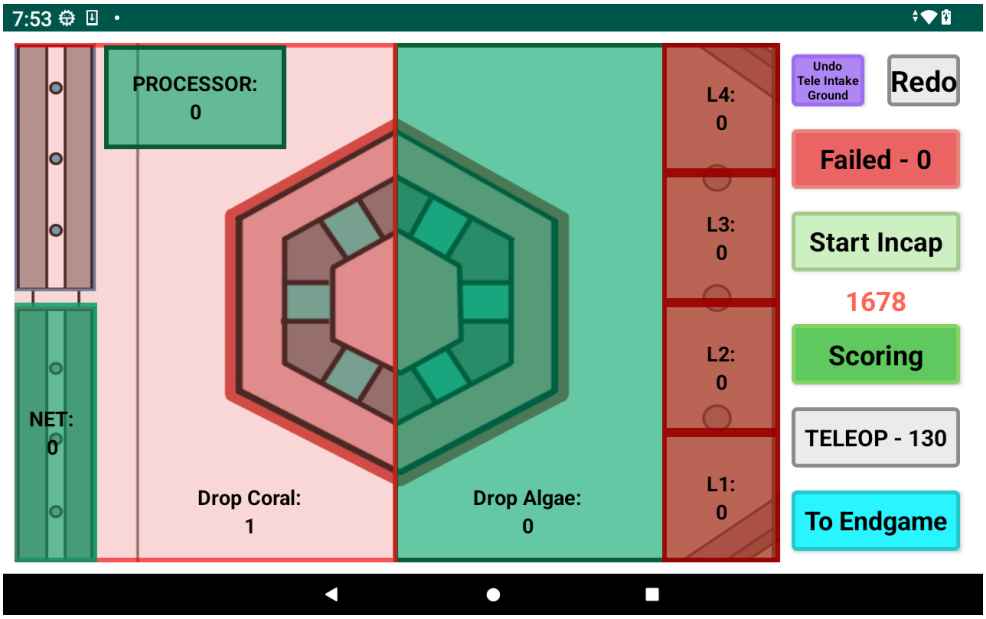
Teleop

After switching to the Teleop screen, like the Auto screens, it displays the corresponding half of the field based on the alliance color and shows a different set of buttons depending on whether the team starts with a preloaded Coral. There are two Teleop screens: the Teleop Intake screen and the Teleop Scoring screen.



Teleop Intake

The Teleop Intake screen has three buttons for recording intake actions from various locations: the Reef (center of the field), the ground, and the two Coral Stations.

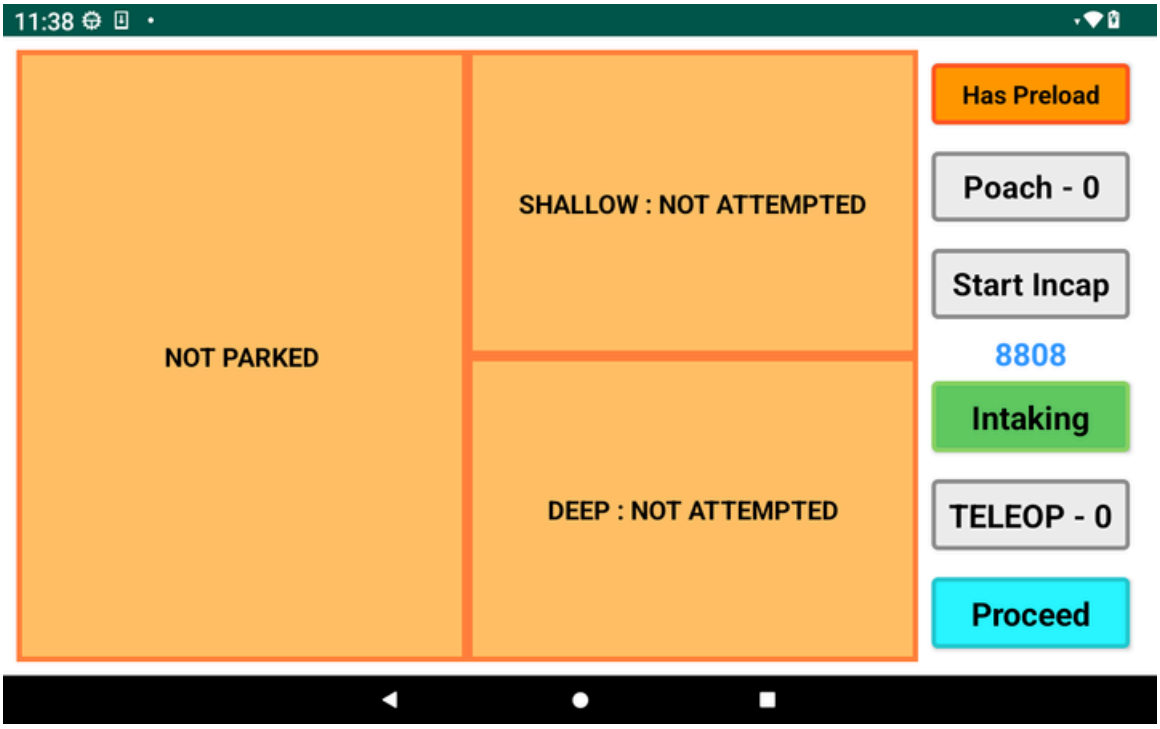


Teleop Scoring

The Teleop Scoring screen includes nine buttons for tracking how teams score Algae and Coral during Teleop. Each level of the Reef (L1, L2, L3, and L4) has a dedicated button to track Coral scored there. The Drop Algae and Drop Coral buttons are used to record when the robot drops a piece, either intentionally or accidentally. The Processor button records when the robot scores Algae in the Processor, and the Net button records when either the robot or the human player scores Algae in the Net.

Endgame

For the 2025 Reefscape season, collecting endgame data is done on a separate screen. The screen consists of three toggle buttons—one for each type of climb: shallow, deep, or park. To track whether a robot attempted to climb (shallow or deep), the toggle can be switched between unattempted, success, and fail.



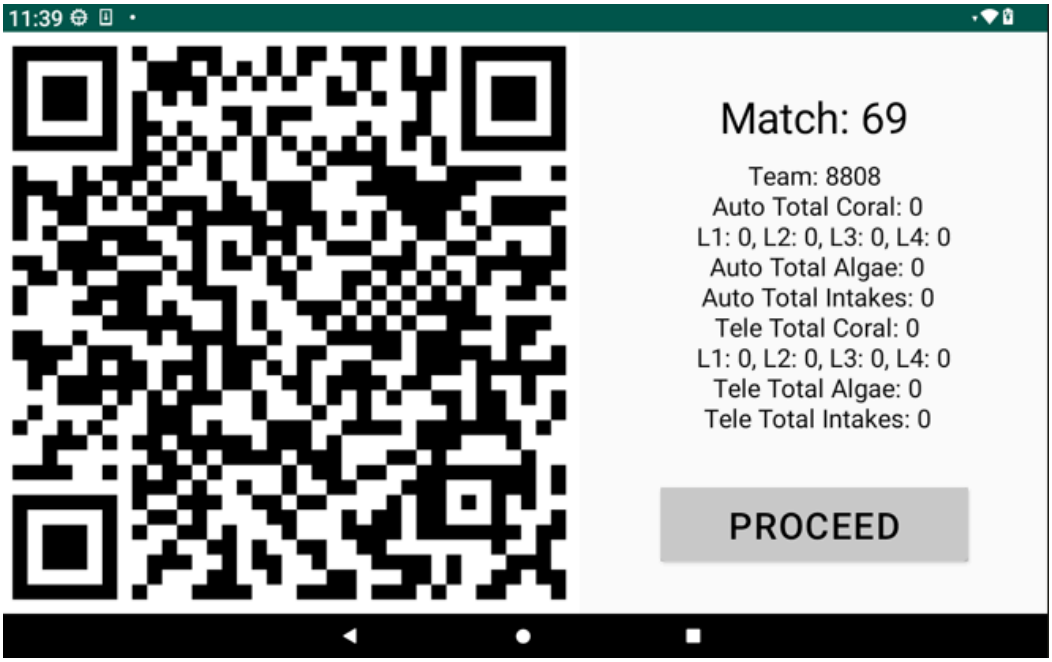
Endgame Screen

There is no “fail” state for the park option, as it was considered an insignificant detail. When a climb button is set to success, the other two options are grayed out. Clicking the same button again switches it to the fail state. If a climb attempt is marked as failed, the park button can then be selected as a successful outcome.

QR Details

The QR Details screen displays the QR code containing the match data collected by the Scout. This QR code is located on the leftmost part of the screen. To the right, the match number and associated data are shown. The heading displays the match number, with the team number listed directly below it.

Next comes the essential Auto data, starting with the total number of Coral scored in Auto, followed by a list of the specific levels where the robot scored. Below this list are the Auto Total Algae and Auto Total Intakes, each represented as a single string.



QR Details Screen (Objective)

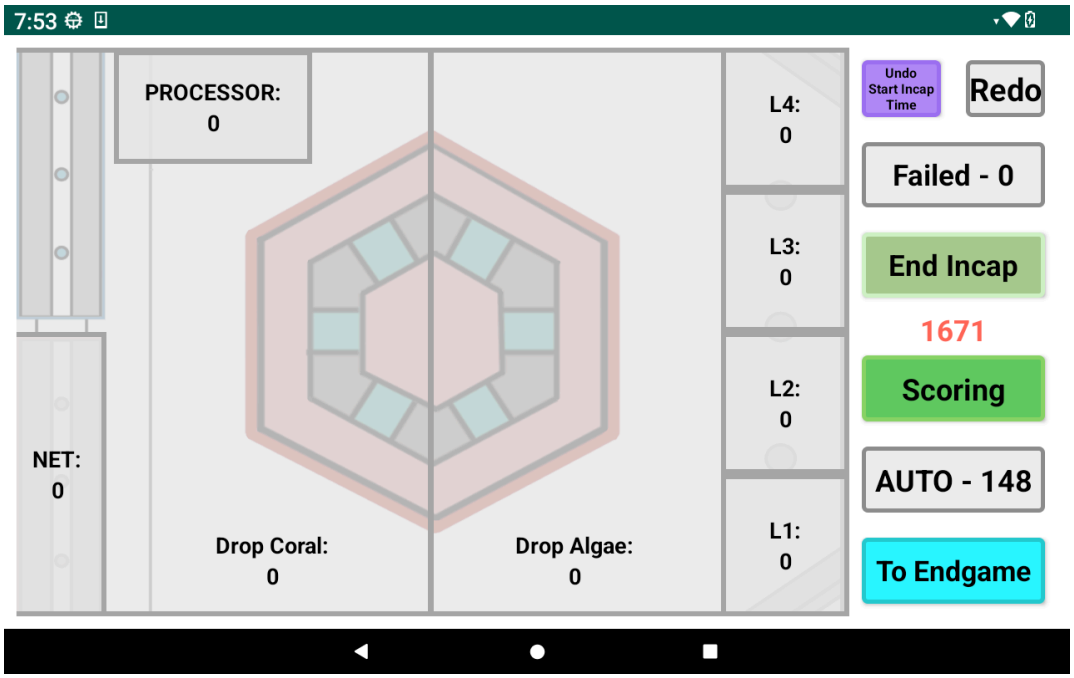
Next, the Tele data is displayed, starting with Tele Total Coral at the top. Below it is a list of the specific levels where Coral was scored, followed by Tele Total Algae and Tele Total Intakes at the bottom. Below the data display, there is a Proceed button that returns the user to the pre-match screen.

Utility Buttons

Navigation Between Auto, Teleop, and Endgame

Like previous years, we had a button to progress the match from Auto to Teleop. Continuing into this season with the Endgame screen, the "To Endgame" button allows the user to switch from Teleop to Endgame. When a user presses the "Failed" button, the "Proceed" button is disabled while the user is failing an action. Additionally, the user has the option to return to the previous screen if no actions have been input after switching screens. This process is initiated by long-pressing the "To Endgame" button (to return from Teleop back to Auto) or the "Proceed" button (to return from Endgame back to Teleop). If the user is in override mode, they can also proceed past the Endgame screen using the "Proceed" button, a feature that allows for faster field testing.

Incap Duration Using Timestamps



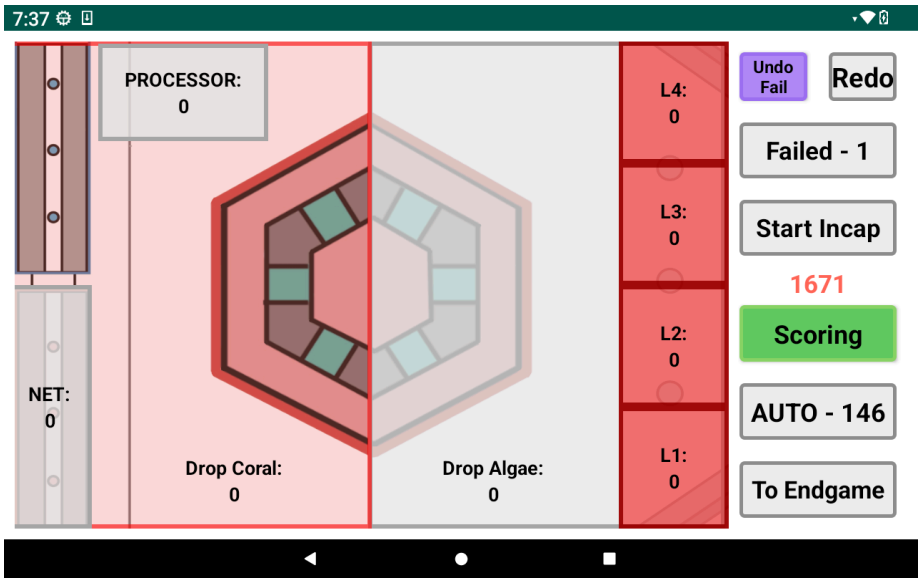
Incap

In Teleop and Endgame only, Scouts mark a robot as incapacitated when it is disabled or its drivetrain movement is significantly impaired. A toggle button records the start and stop timestamps of the incapacitation period. This allows us to calculate the total incap time of a robot during a match. Note that if a team is marked as incapacitated for less than 8 seconds, the time is considered inconsequential and is disregarded in the match data.

Switching Intake and Scoring Buttons

When a scout taps an intake button, the app switches from the intake screen to the scoring screen, and vice versa. Due to robots this season having the ability to hold both algae and coral simultaneously, there is a button that allows Scouts to manually switch between the Scoring and Intake screens. Certain buttons, like Undo, Redo, and Incap, will always remain.

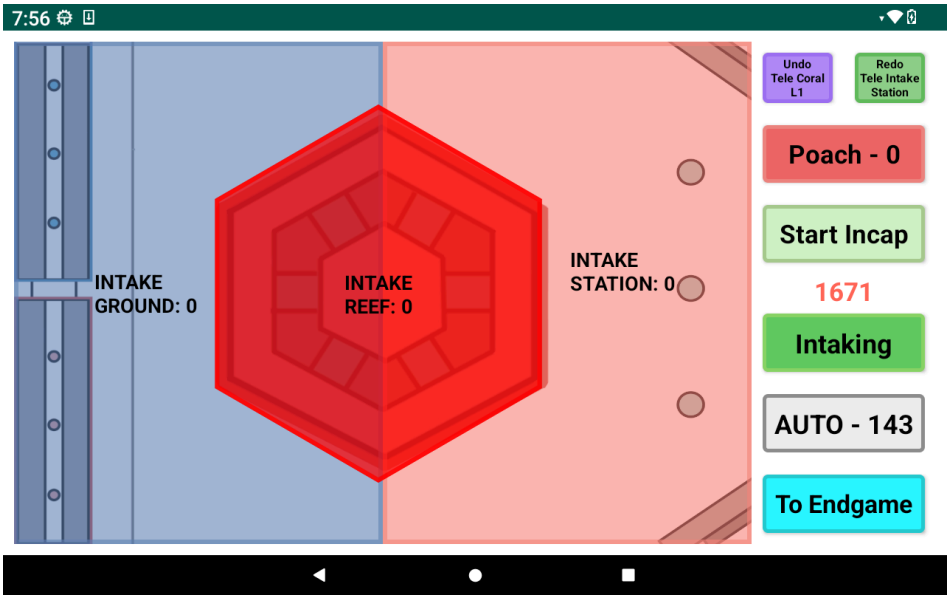
Fail Button



Failing

For the 2025 Reefscape season, the Fail button has two different functions. During the scoring phase, a scout can record a fail by pressing the Fail button, followed by the location where the robot scores (e.g., example path: Fail F2 L3). The second function is to serve as the poach button during the intaking phase (Auto or Teleop). Recording poaches helps differentiate between robots intaking algae on their own side of the field versus their opponent's side.

Undo and Redo



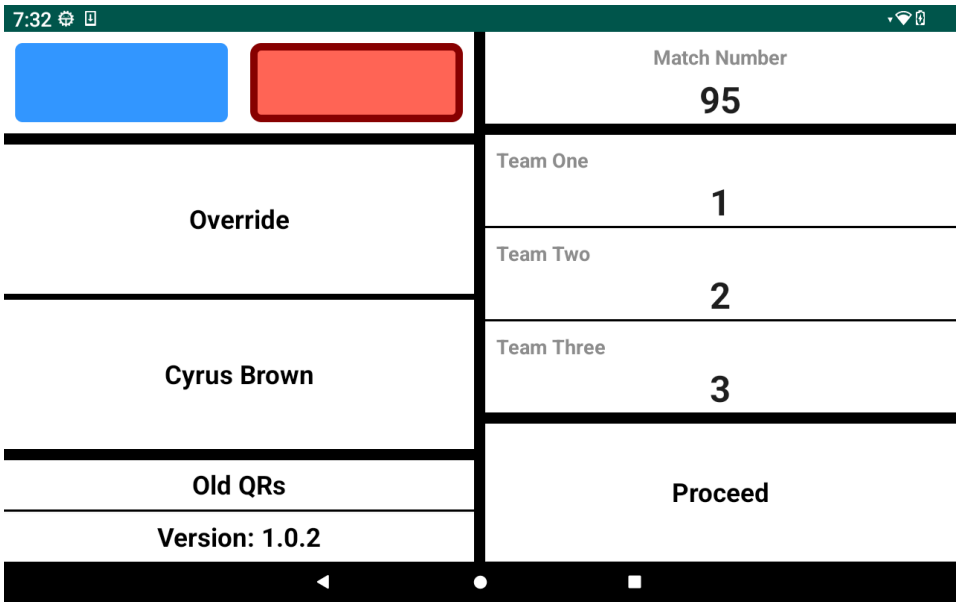
Undo and Redo Buttons

When users press the Undo or Redo button, it undoes or redoes the most recent action recorded or undone, respectively. When a user undoes or redoes a failed action, both the fail and the actual action are undone or redone.

Subjective Collection

Throughout the competition, we have two Subjective Scouts scouting at all times. Each Scout watches all three robots in one alliance and ranks them against each other for data points like driver ability. Subjective Scouts rank the robots relative to one another by analyzing each driver's performance. Ranks are from 1 to 3, with 3 indicating the best robot in that category in the alliance. They also record how many seconds are left in the match when the robot attempts to climb, whether the robot is tippy, and other data points.

Match Input



Subjective Collection Input Screen

The Subjective Collection Input Screen is similar to the input screen for Objective Collection. The main differences are that each Subjective Scout collects data for all three teams in an alliance, and instead of assigning teams using Scout IDs, assignments are made based on the alliance color they are scouting.

Collected Datapoints

Died: Records if the robot goes incap for the duration of the match.

Can Cross Barge: Records if a robot can cross under the barge to the opposite field.

Tippy: Records if the robot was unstable and unbalanced during the match.

HP from Team: Records if the barge human player is from their team.

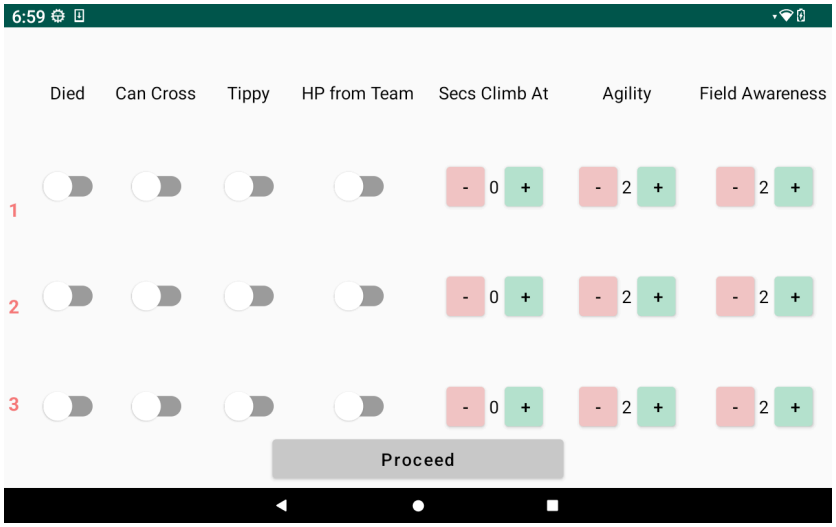
Time Left to Climb (Secs Climb At): Records approximately how much time remains in the match when each robot approaches the barge to climb. Subjective Scouts round the time to the nearest 10 seconds and

Field Awareness: Ranking of driver's awareness of the field and game elements as well as the quality of each robot's interactions with other teams on the field, including on the opposing alliance's side of the field. Ranks are from 1 to 3.

Agility: Ranking of robot's speed and maneuverability on the field. Ranks are from 1 to 3.

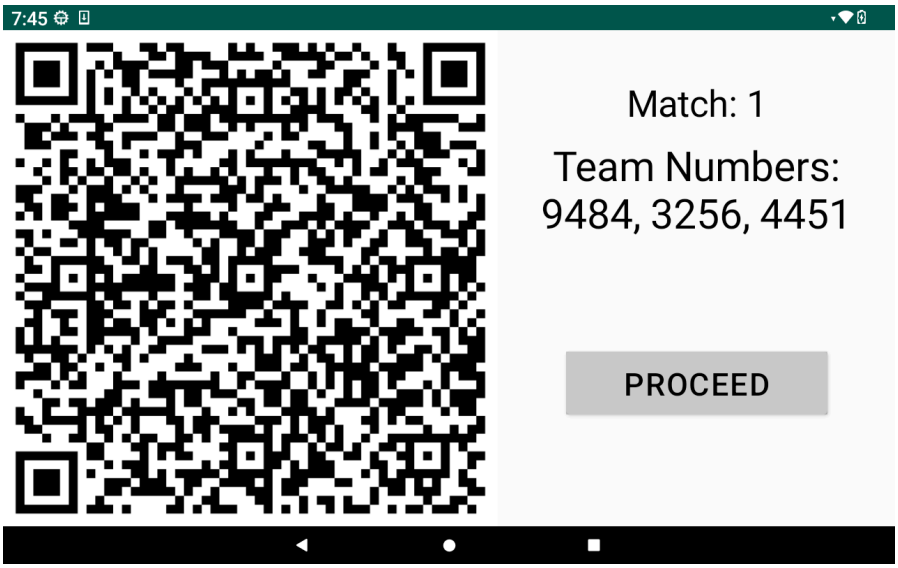
Screens

Data Collection



Subjective Collection Screen

QR Details



QR Essentials Screen (Subjective)

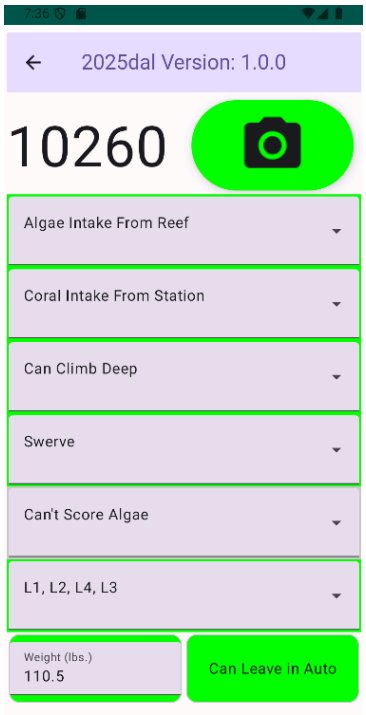
At the end of the match, the Subjective Scout scans the QR code on a screen similar to the Objective QR screen. Subjective collection is essential to strategy during the picklist process, as it gathers valuable qualitative information that Objective Collection may miss.

Pit Collection

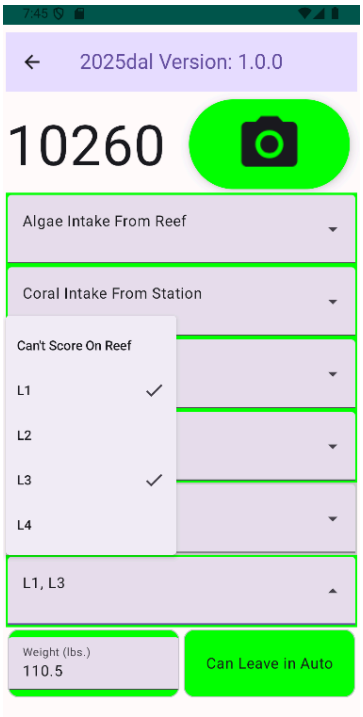
The Pit Collection app is a phone app that our Pit Scout uses to collect robot data and photos at each event on practice day, before matches start. It's an important tool to provide strategists with some data about every robot during the first few hours of competition, when scouting data is still sparse. Pit Scouts record information like drivetrain type, scoring levels, intake locations, and climbing ability using dropdowns in the app. They also take two photos of each robot—one full view and one side view—which are saved with the team number in a special folder.

Pages

Main Data Collection



Data Collection Screen

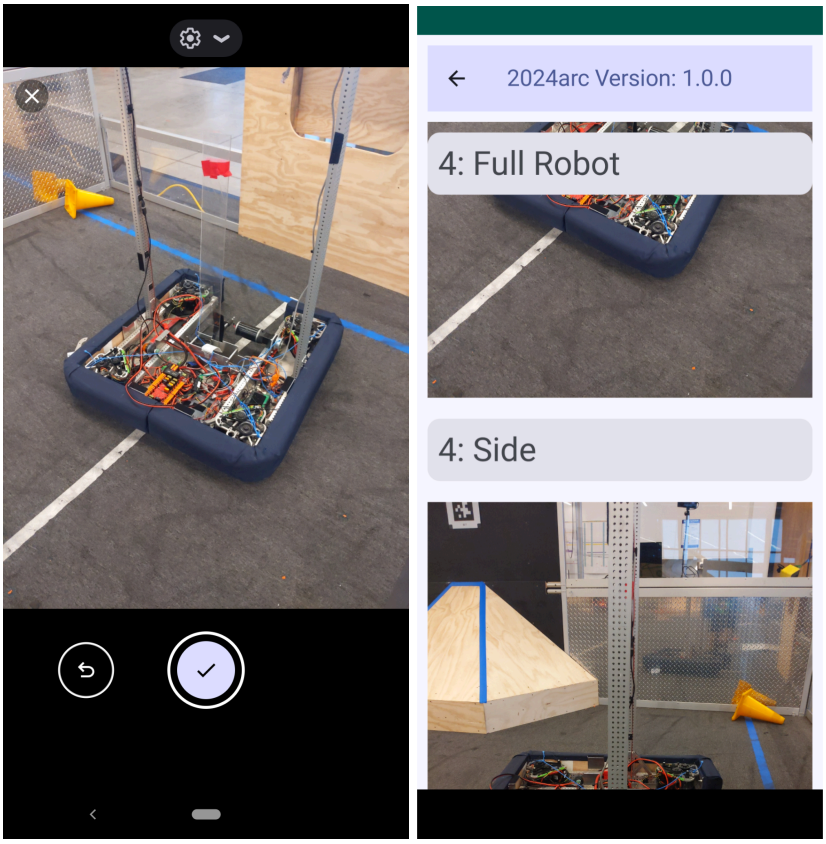


Multiple Select Dropdowns

Data points collected include the reef levels the robot can score on,the areas it can intake coral from (station and/or ground), the areas it can intake algae from (reef and/or

ground), where it can score algae (processor and/or net), which climb the robot can accomplish (deep or shallow), its weight, and the type of drivetrain it has. In the screenshot above, green signifies that an input has been completed.

Robot Photos

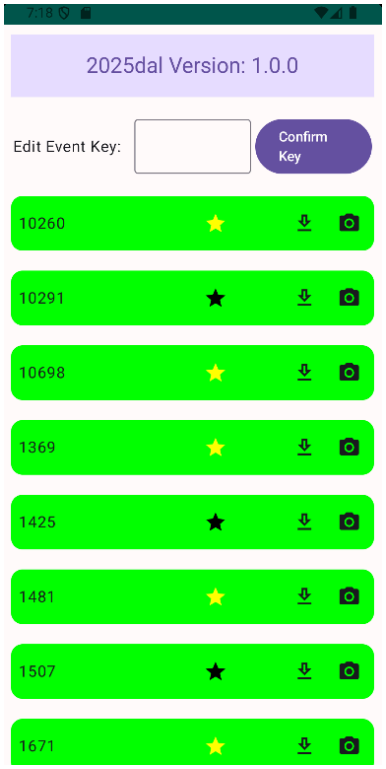


Robot Photo Screen

In past years, the Match Strategist used the Pit Collection to collect pictures of each robot's intake, indexer, shooter, climber, and drivetrain, as well as two photos of the entire robot. These pictures are taken in the app using the phone camera and are stored in a folder called robot_pictures in the local Downloads folder. The photos are automatically transferred to the cloud MongoDB database via Kestrel. The pictures are named with the team number first, followed by the picture angle (e.g., 1678_full_robot.jpg and 1678_side.jpg). Users can preview pictures by holding down the camera icon on the team list screen.

Utility Features

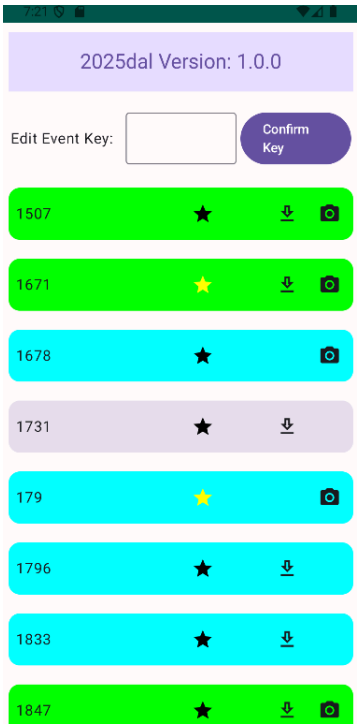
Starring Teams



Team List Screen with Starred Teams

The Match Strategist can use Starred Teams to mark robots for additional attention by tapping the star icon in the list of teams. Starred Teams display a yellow star instead of the default black one.

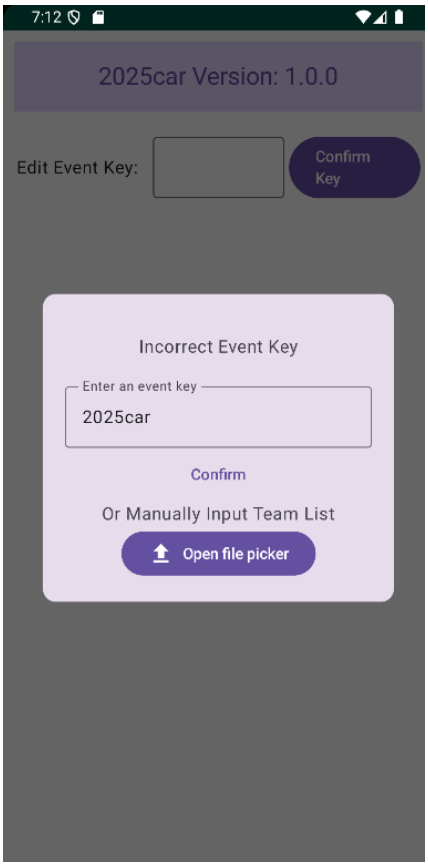
Highlighting Progress



Team List Screen with Highlighted Teams

When viewing the list of teams, the color of each team's cell depends on what kind of data the Pit Scout has collected for that team. If there is no data, the cell appears gray; if there is partial data (only pictures or only other data), the cell appears cyan; and if both pictures and data are present, the cell appears green. Icons also indicate collection status: the download icon means the team has data, and the camera icon means the team has photos. This allows the Pit Scout to see which data still needs to be collected.

Editing Event Key



Event Key Pop-up

Users can change which event the Pit Collection app pulls the team list from by entering a different event key. The Blue Alliance provides event keys, and if a user enters an invalid key, an error screen prompts them to enter a valid one. The app keeps the most recent key until a valid key is entered.

Naming Photos and JSON Files

The app stores the data for every event key in a separate folder within the Downloads folder. Each event key's specific folder is named after the event key (e.g., 2025dal). The Pit Collection app stores the collected data in its folder as a JSON file called pit_data.json for each event key.

Stand Strategist

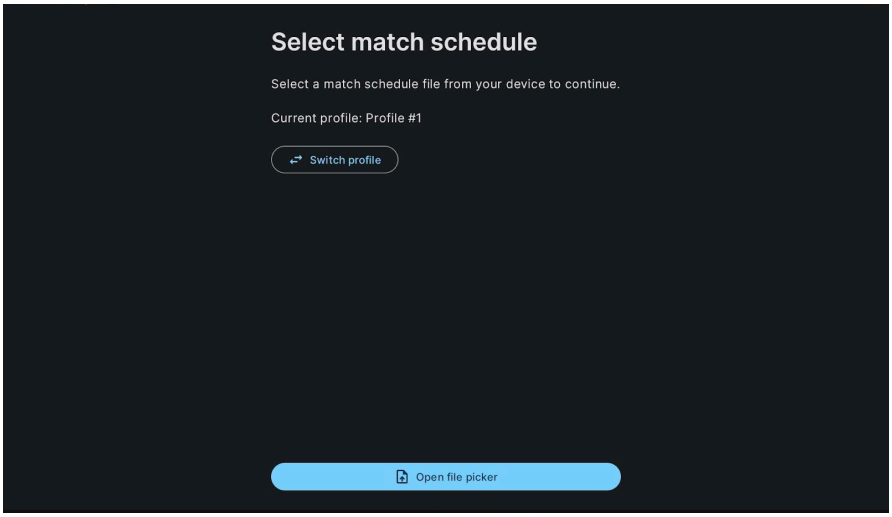
Stand Strategist is an app used by Stand Strategists to take notes about the teams in an alliance during each match. Stand Strategist is written in Kotlin and built using the Compose Multiplatform UI framework by JetBrains.

Throughout the competition, the two Stand Strategists use this app to record detailed notes and observations about each team that our other scouting apps do not provide. Then, during the picklist meeting, they use these notes to add context to the rest of our data. This helps us identify if any numbers might be inflated or skewed and provides a way to track observations not recorded by the rest of the scouting system. All of this gives us greater insight into teams when building our picklist.

This year, we kept the app very similar to last year. However, we added several visual improvements to better distinguish alliances when comparing all the data for one team.

Entering Data

When entering data into Stand Strategist at a competition, the user is first prompted to select or create a user profile (see Profile Management). If the user has created a new profile, they are then asked to select a match schedule file:

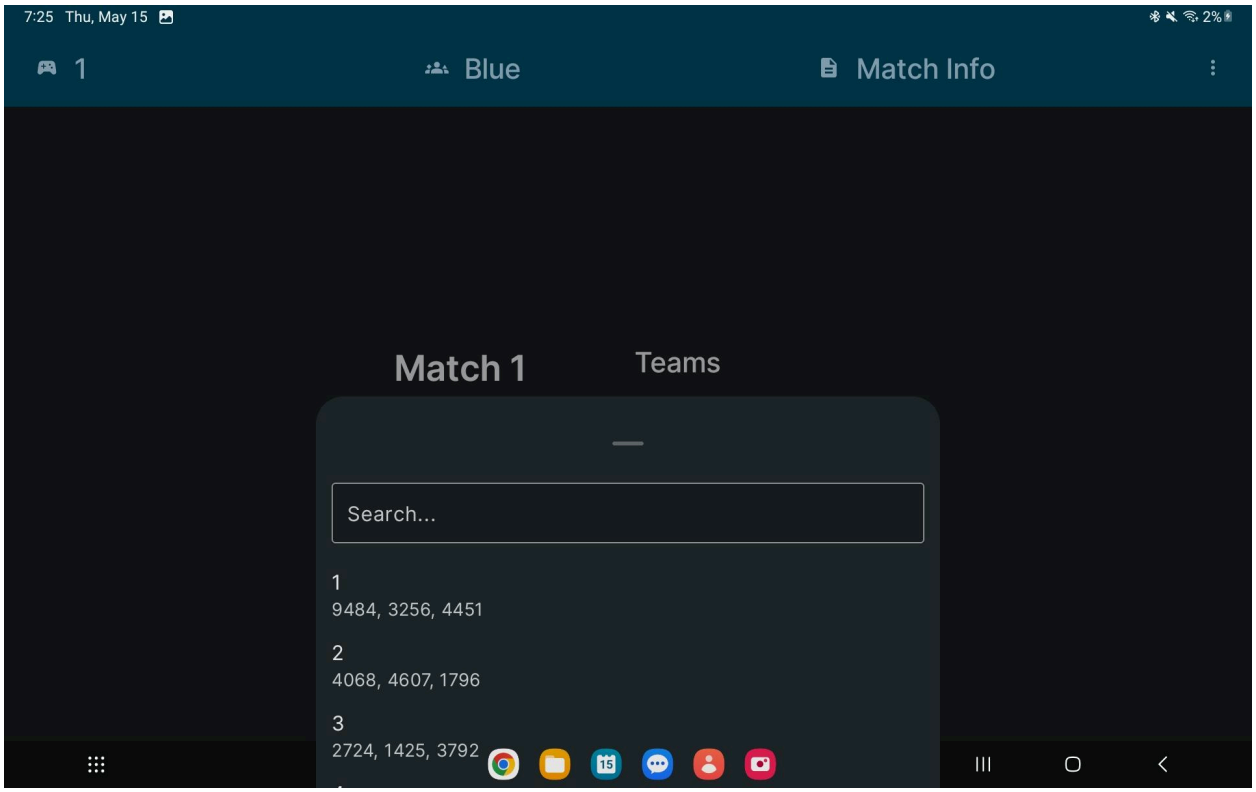


Match Schedule Selection Screen

Selecting the “Open file picker” button opens the system file dialog, allowing the user to select a match schedule JSON file generated by our Server.

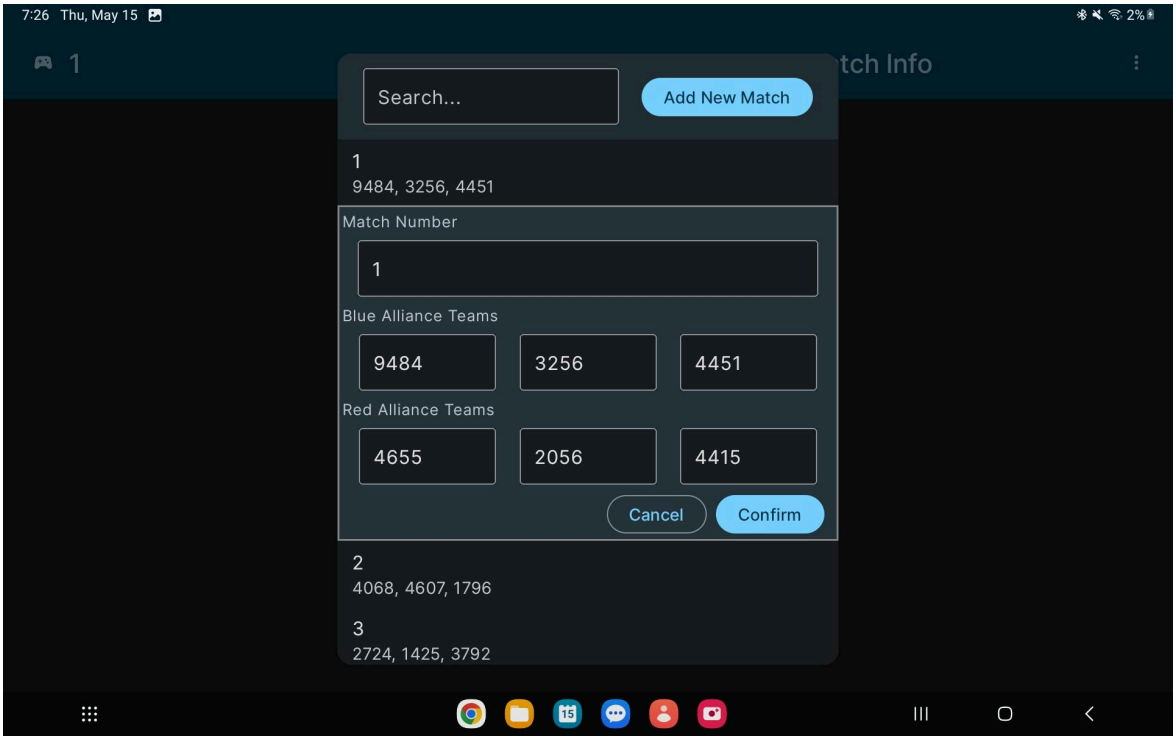
Match Selection

By clicking the controller icon in the navigation bar, the user can open the Match Selection tab, which shows a list of every match in the match schedule:



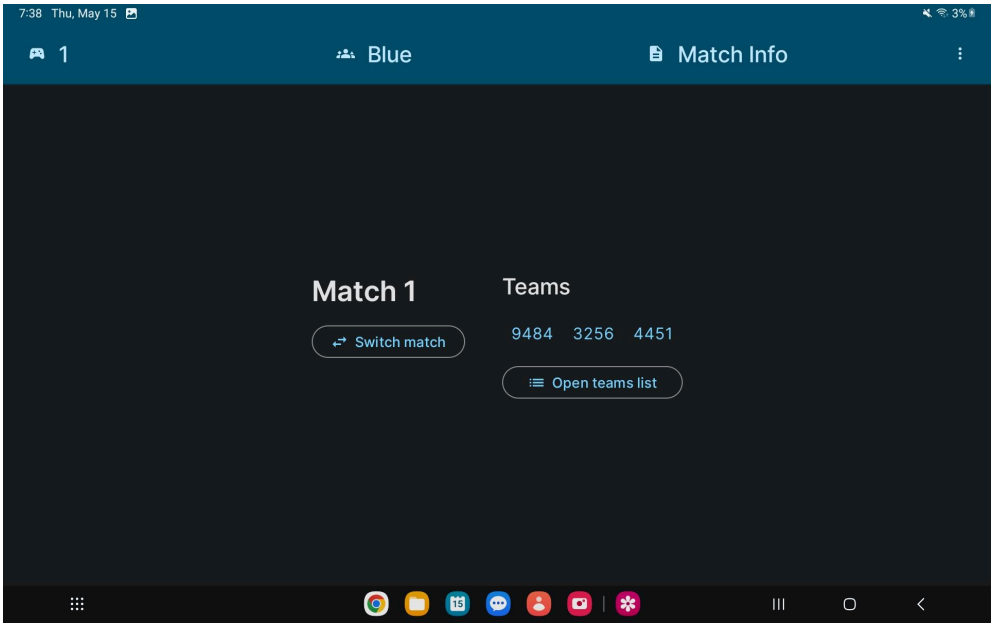
Match Schedule pop-up

Users can create a custom match by clicking the “Add New Match” button. This is useful for strategists who want to scout practice matches on the first day of competition. Users can also click on an existing match to edit it:



Edit Match Schedule pop-up

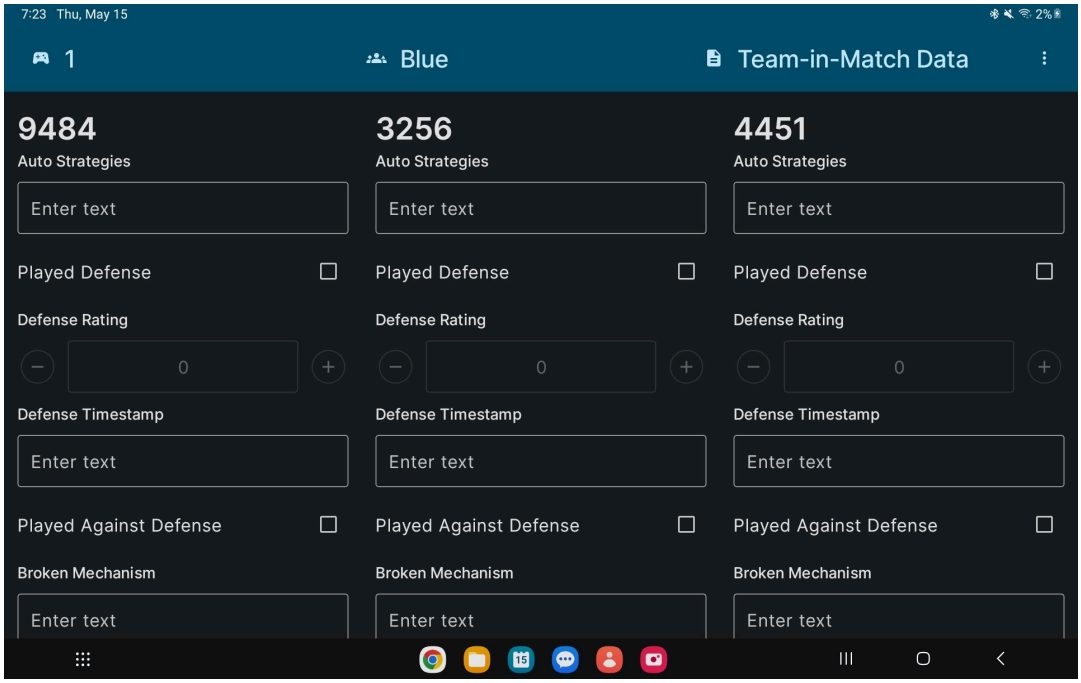
Once a match is selected, the Match Info page is displayed. This page shows the match number, the teams in the match on the user's selected alliance, and two navigation buttons. Pressing a team number on the Match Info page will take the user to that team's Team Screen.



Match Info page

Team-in-Match

During the match, the user can enter data and observations on the Team-in-Match Data page and the Team Data page:



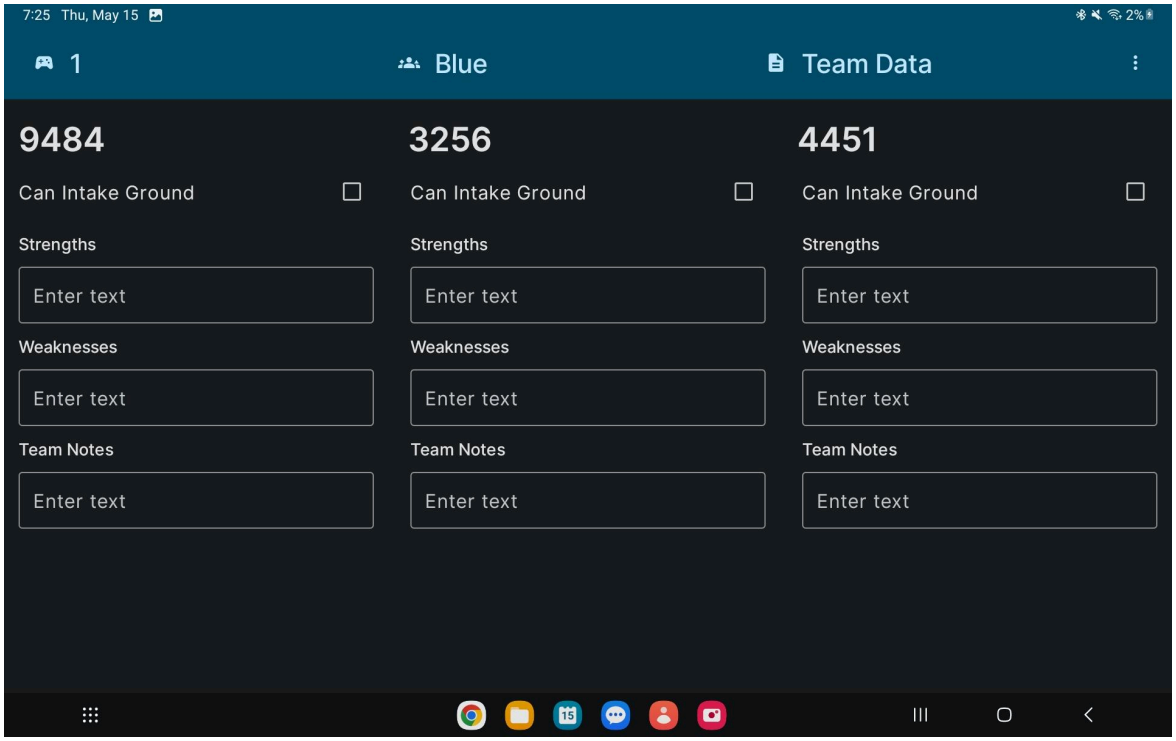
Team-in-Match Data page

To give our strategists a wider range of information about each robot, we collect these datapoints: “Can Intake Ground,” “Strengths,” “Weaknesses,” “Team Notes,” “Played Defense,” “Defense Rating,” “Defense Timestamp,” “Played Against Defense,” “Broken Mechanism,” “Match Notes,” and “Auto Strategies.”

The Team-in-Match Data page records a team’s performance within a specific match. The data entered here is match-specific, meaning it only shows for that team in that match. This year, we added more specific datapoints to better evaluate team performance. For example, “Played Against Defense” helps determine how well a team played against defense.

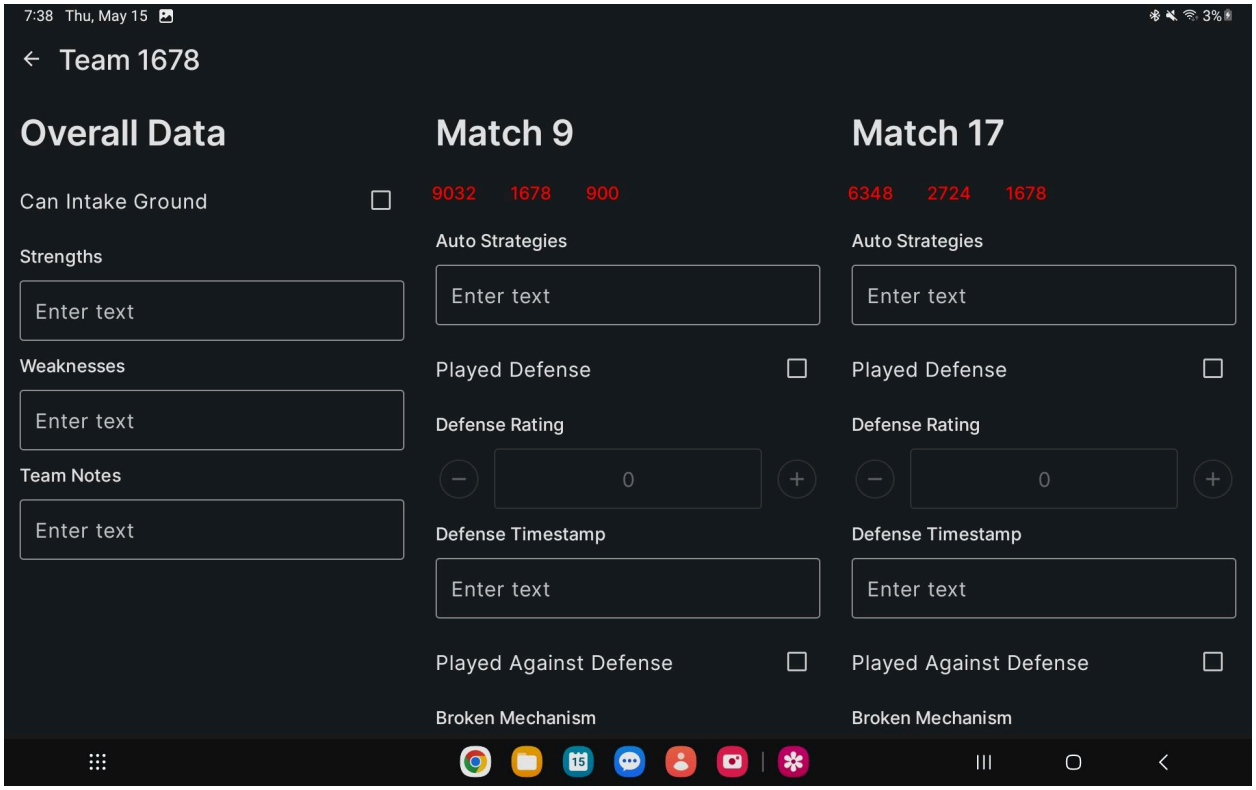
Team Data

The Team Data page records more general comments on a team's overall performance across all matches. Data entered on this page for a team is shown and updated for all its matches throughout the competition.



Team Data page

All notes for a team can be viewed by clicking on a team number in the team list, searching for a team number in the search bar, or clicking a team number on the Match Info page.

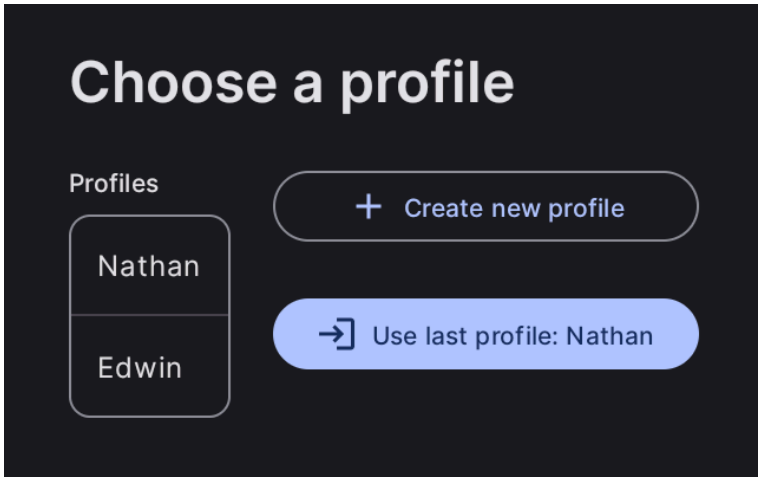


Team Overall Data page

Navigation

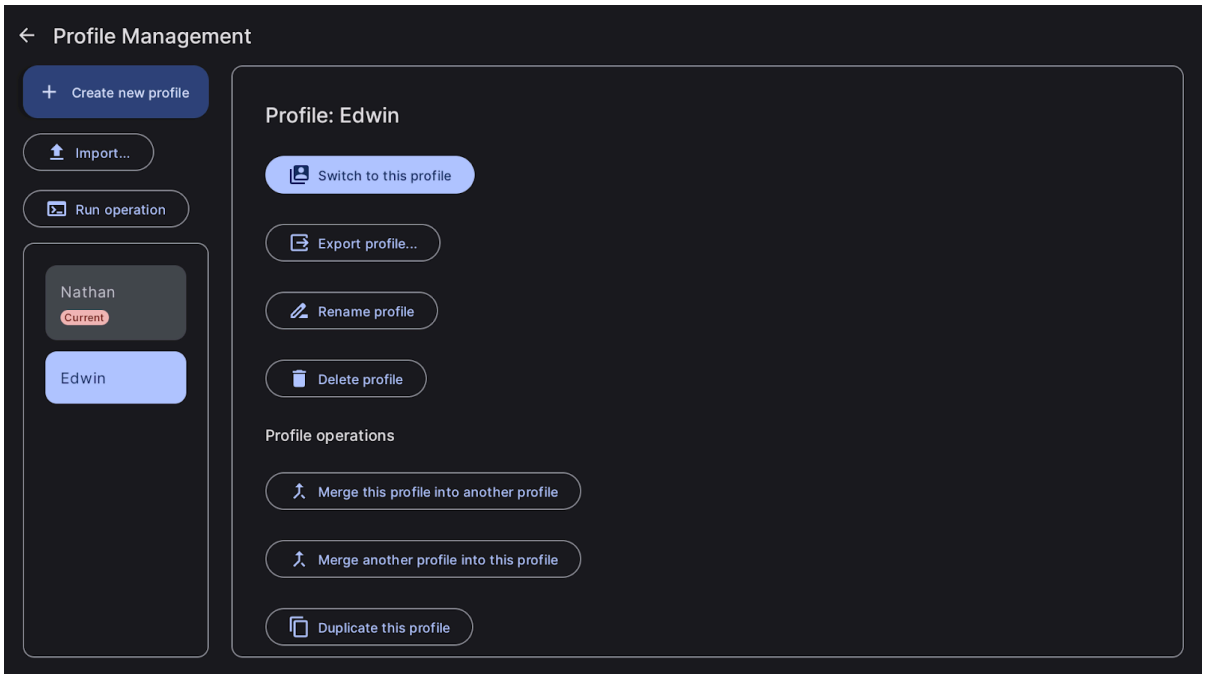
Stand Strategist users can navigate between data collection pages by swiping left or right on the tablet screen or by using specific keyboard shortcuts. When the user moves through all the pages for a match, the match number increments or decrements accordingly. There are also several buttons the user can press to open specific pages or pop-ups.

Profile Management



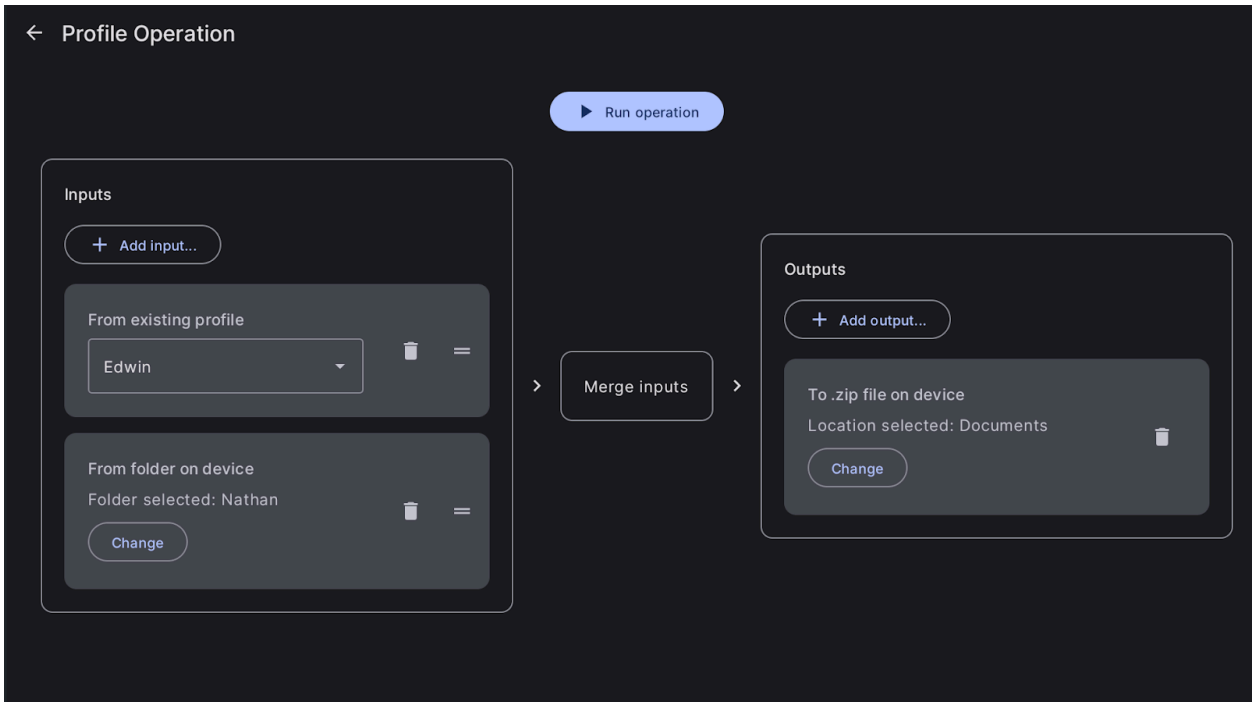
Profile selection screen, shown when the app opens

At competitions, we sometimes need multiple Stand Strategists to collect data using a limited number of tablets. To address this, we implemented profiles to separate different users' data. The profile management screen shows various actions that can be taken on profiles:



Profile management screen

We also decided to implement options for importing and exporting profiles, as well as the ability to merge them. Since profile operations can be complex, we built a screen to create and manage these:



Profile operation screen

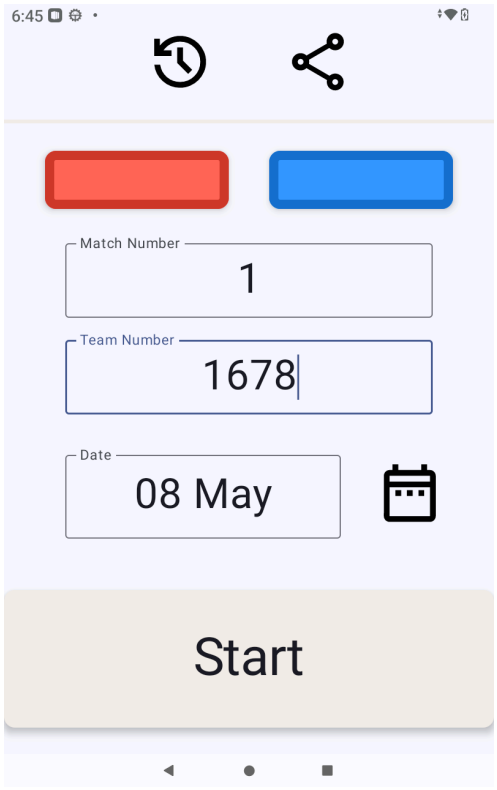
The profile operation screen is a common interface for importing, exporting, and merging profiles in different formats.

On the device running Stand Strategist, each profile has its own folder in the storage directory. Each folder contains the profile's match schedule, profile-specific settings, and collected data.

Playoff Scouting

This year, we introduced a simple Playoff Scouting app designed for data collection during elimination matches and for scouting in-shop drive practices. In a simple scouting system, this app may be enough for all objective data collection needs; it tracks scored pieces and failed scores.

Match Information

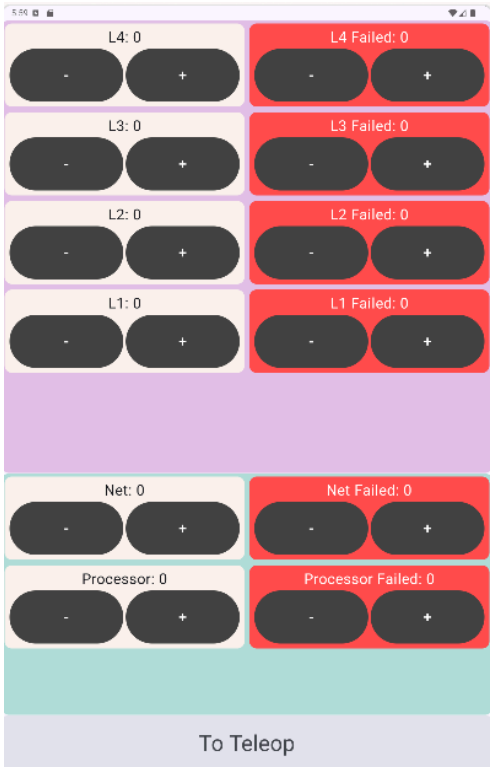


Match Information Screen

The Match Info screen allows the user to select the match number, team number, and alliance color. Scouts can also view previously submitted results to verify or reference earlier data. This page ensures that all match context is correctly set before scouting begins.

Data Collection Screens

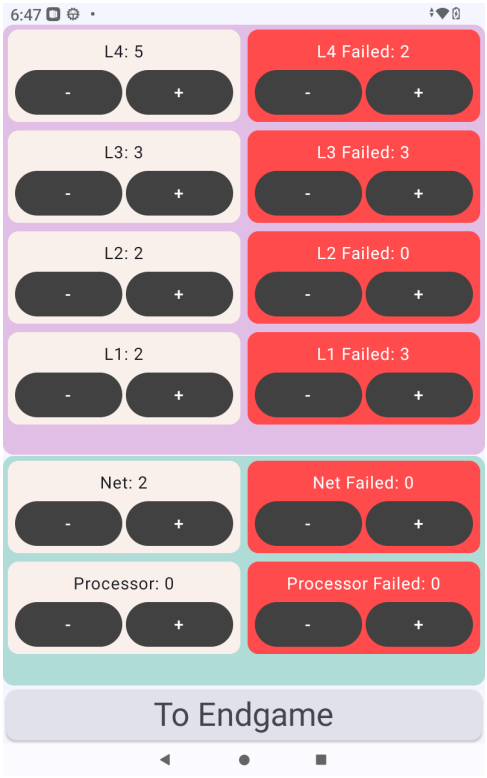
Auto



Auto Screen

The Auto screen enables scouts to input scoring data for Reef Levels L1 through L4, with corresponding fail buttons for each level. Additional buttons are provided to record scores in the Processor and the Net.

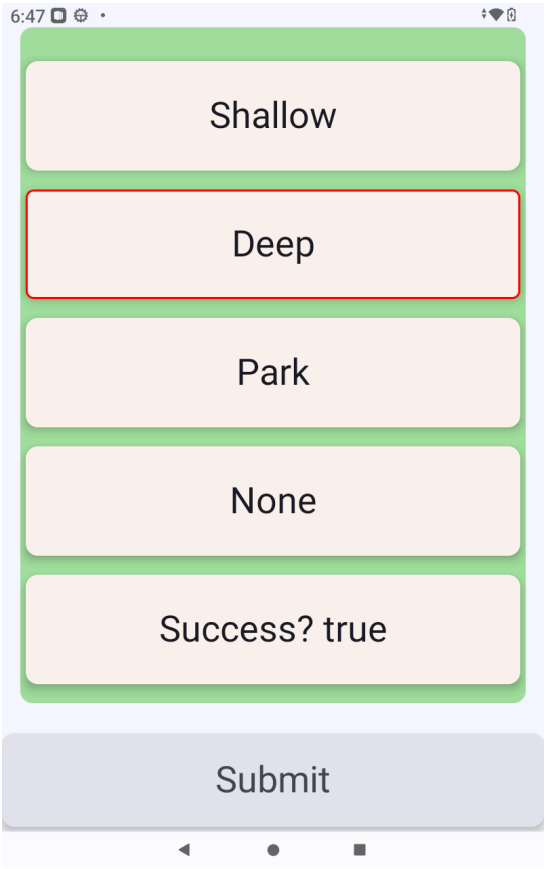
Teleop Page



Teleop Screen

The Teleop screen features the same layout and functionality as the Auto screen. Scouts can input scoring data for Reef Levels L1–L4, failed attempts, and actions involving the Processor and the Net. This separation ensures a clear distinction between Auto and Teleop.

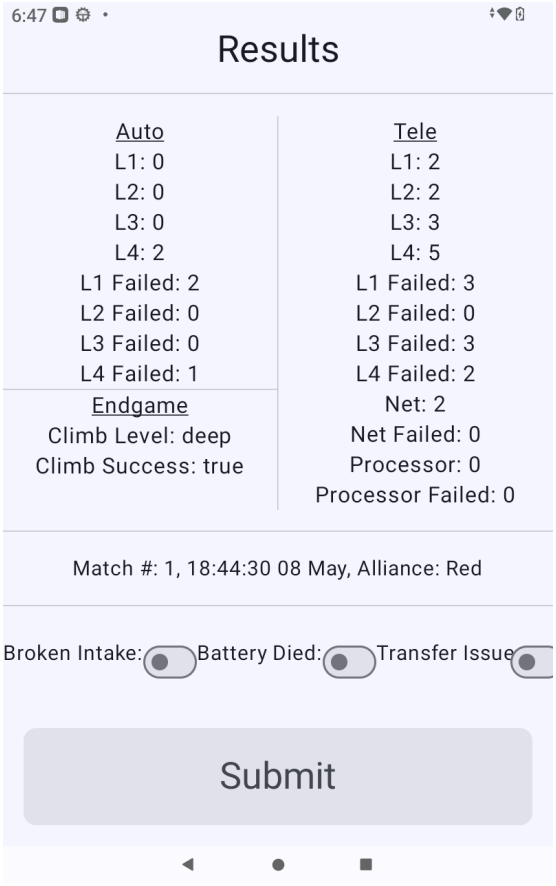
Endgame Page



Endgame Screen

On the Endgame screen, scouts select the robot's climb type (shallow, deep, or park) and indicate whether the attempt was successful or failed.

Results



Results Screen

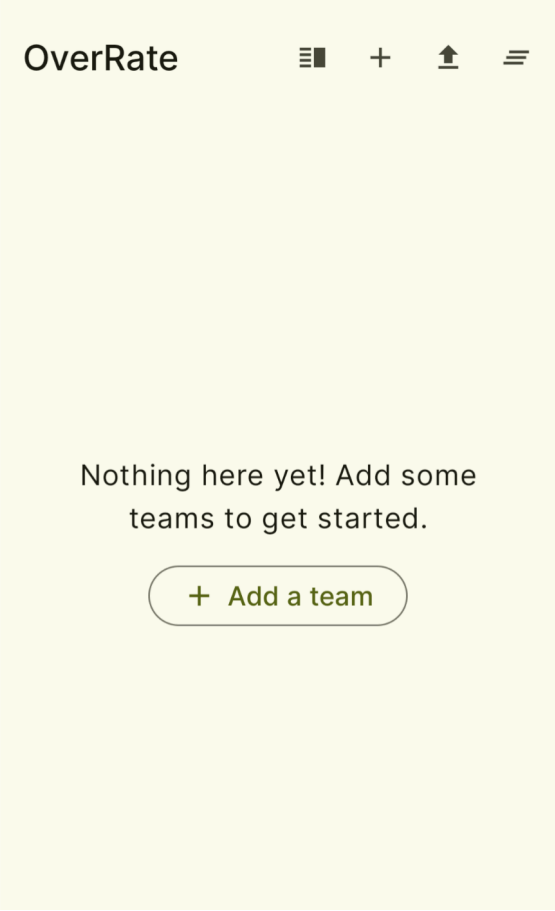
The Results screen displays a summary of the data collected during the match. Scouts can quickly review their inputs before finalizing the submission. This page helps catch potential mistakes and ensures completeness before the data is uploaded.

OverRate

OverRate is an Android app, built entirely using Jetpack Compose, designed for strategists to rate and rank teams relative to each other in real time during an event. Users can add teams to a list, assign numeric ratings, reorder them, and organize them with dividers. The app is flexible and supports a wide range of use cases. It is not meant to promote any specific method of rating or organizing teams. For example, a strategist mentor used it on the first day to identify teams that might not fit well into our alliance, and on the second day to rank groups of teams with similar attributes. The stored data is not used elsewhere in the scouting system, so the app functions solely as a notepad.

Importing Data

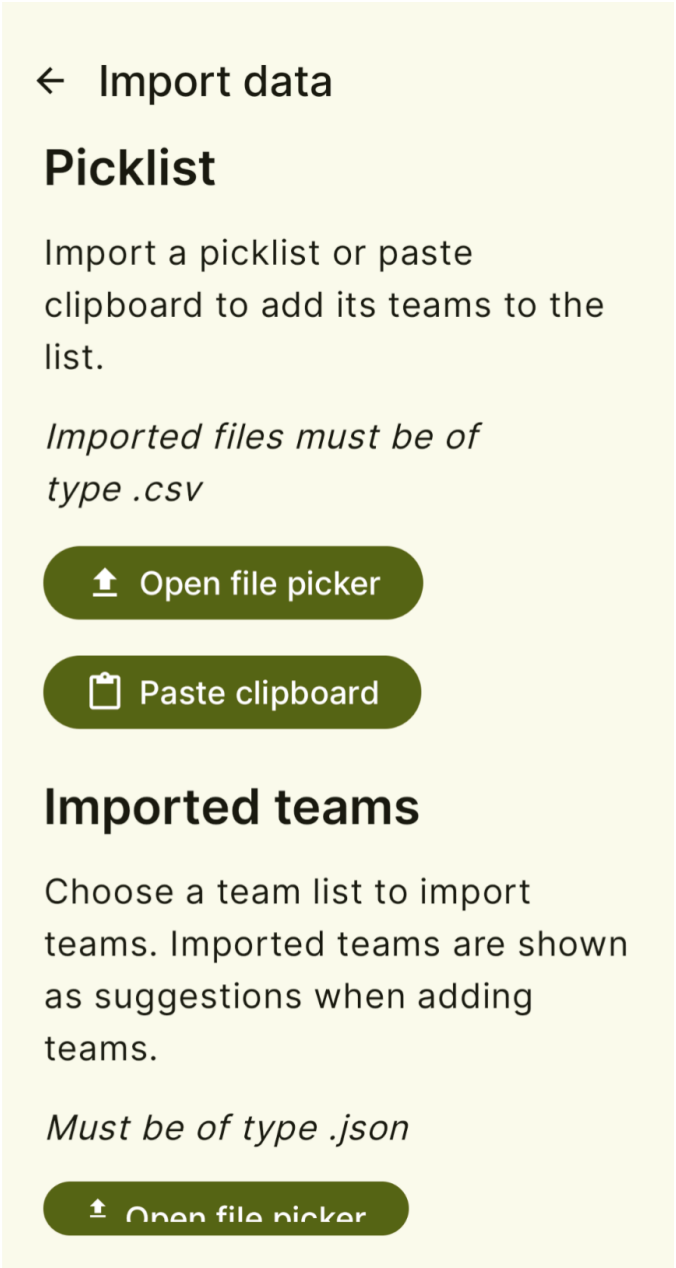
When first opened, the app displays a blank slate:



OverRate's Main List View

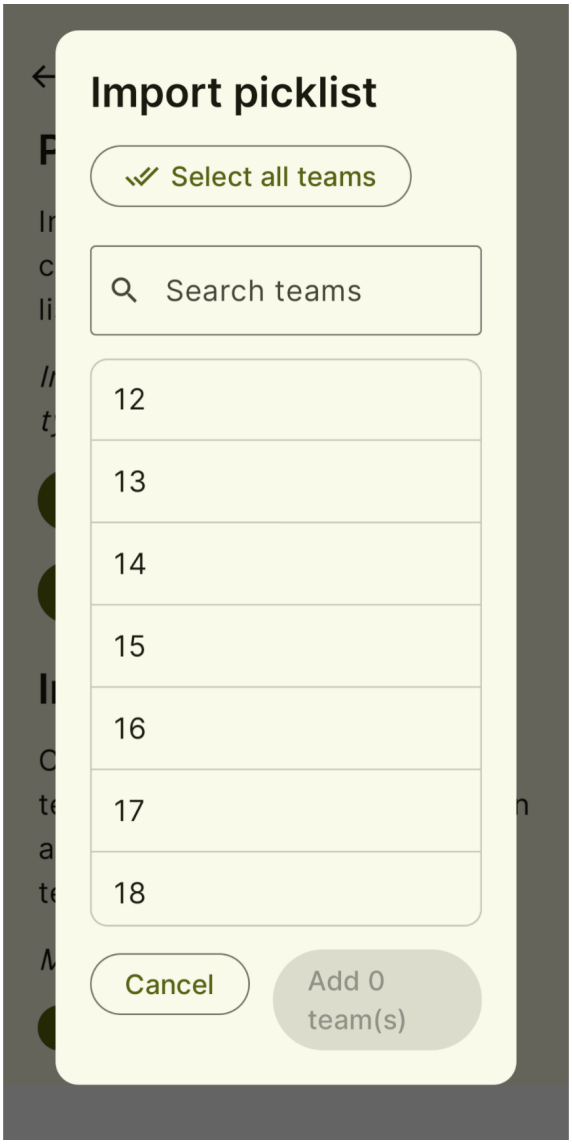
The user can import teams from a team list to display as suggestions when adding new teams. Alternatively, the user can import a picklist to immediately add its teams to the list in the order they appear in the file.

When importing teams from a team list, the user can select a team list file (the same format used by the rest of the scouting system) to add its teams to the app's internal list of imported teams. If multiple team lists are added, they are merged. The user can see how many teams have been imported and can also clear the saved teams.



Importing Data Screen

To import a picklist, the user has two options. They can tap the button labeled “Open file picker.” The teams must be separated by line breaks within the file. After selecting the file, the user can choose which teams to import, and the selected teams are added to the main list. Each team added to the main list has a comment showing its original rank in the imported picklist.



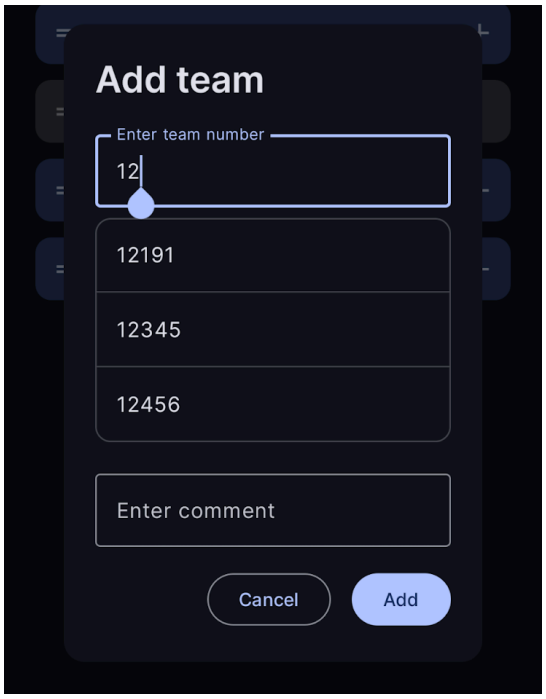
Importing a Picklist

Main List

The main list displays all teams in a list view, each with its own customizable rank, position, and short note.

Reordering Teams

The main list can include both teams and dividers. Each team has a rating that can be adjusted using the plus and minus buttons. Every item in the list has a drag handle, allowing the user to reorder items. Users can press and hold an item to edit its details or delete it.



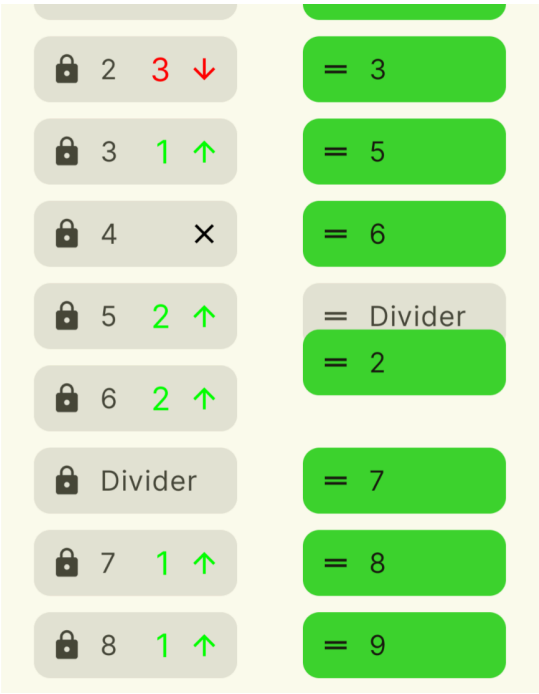
Adding a Team

The user can add a team to the list by opening a dialog and entering a team number. If a team list has been uploaded, the app will suggest team numbers from that list. The user can also

add an optional comment, which appears next to the team number. Dividers are added in a similar way.

Parallel Leaderboard

In 2025, we added a parallel leaderboard option that lets the user make a copy of the main list. This copy can be reordered, with teams and dividers removed and comments edited. Once the user is satisfied, they can either accept or discard the changes. At the end of a competition, the user can use the Clear button at the top of the app to reset the entire list.

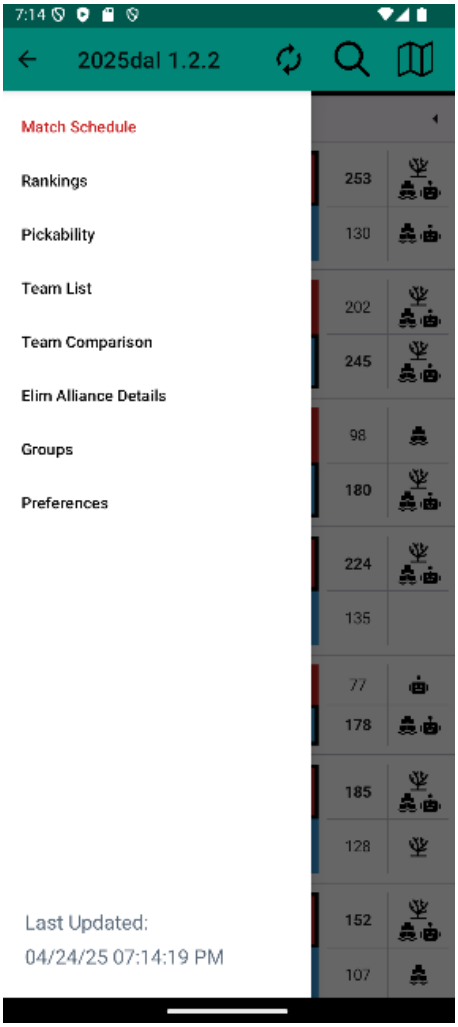


Editing the Parallel Leaderboard

Viewer

The Viewer is an Android app developed in Kotlin. It is the main visualization app that allows strategists to view, organize, and analyze processed scouting data live during competitions to make better-informed match strategies. This year, we replaced most of the XML with Jetpack Compose and added many useful new features to the app.

Navigation

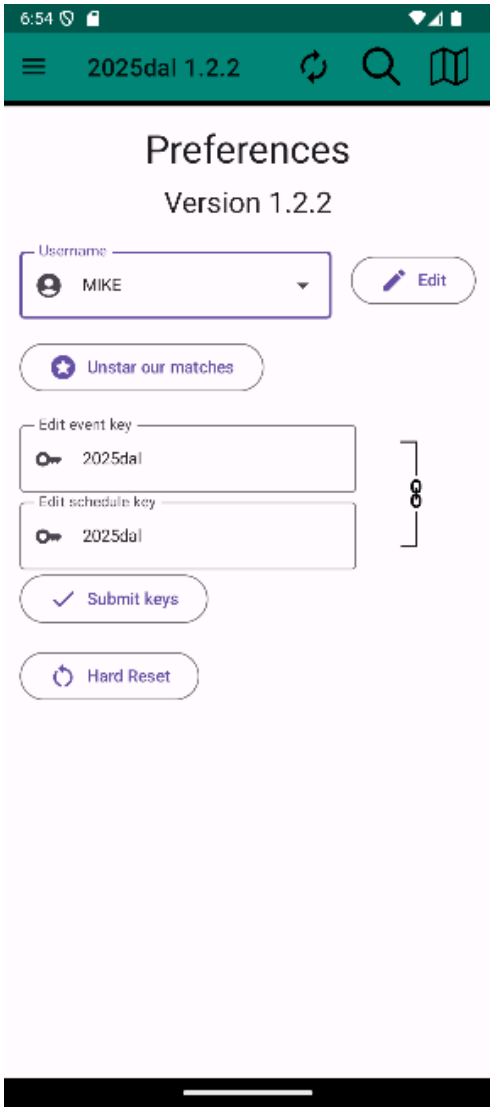


Navigation Sidebar

The Viewer app uses a sidebar to help users navigate its primary pages. There are also various places where users can navigate to related pages (e.g., from the Match Schedule to a Match Details page). The sidebar is accessible from any screen in the app to optimize navigation. Additionally, the top of the sidebar displays the current event and version number.

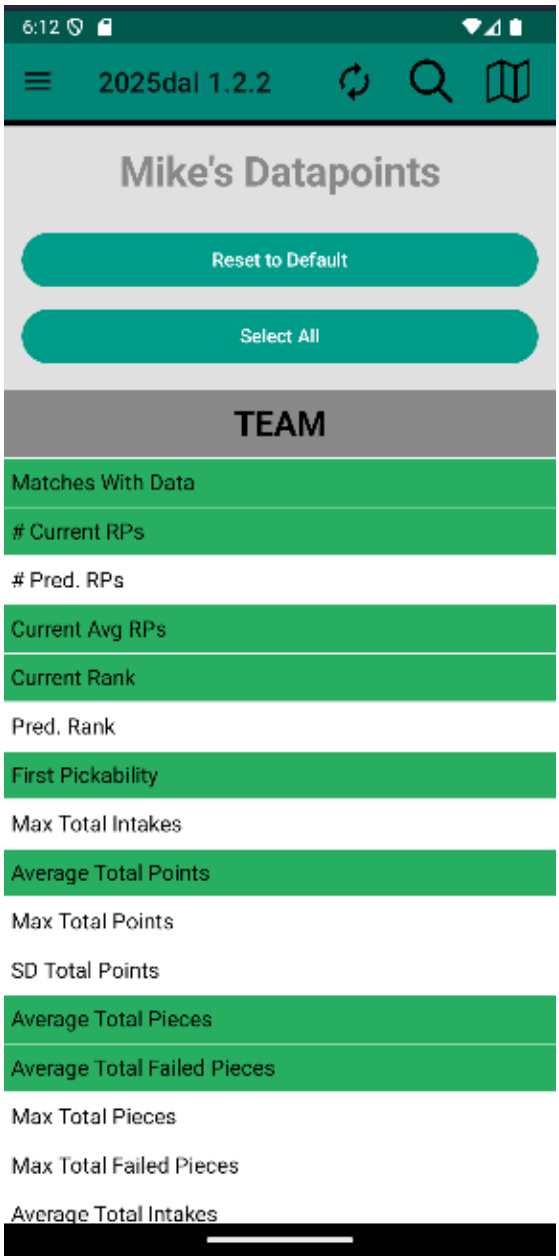
Pages

Preferences



Preferences page

On the Preferences page, the user can change the displayed event to view information from previous events. There is also a button to star or unstar all matches, allowing users to filter and see only starred matches on the Match Schedule page.

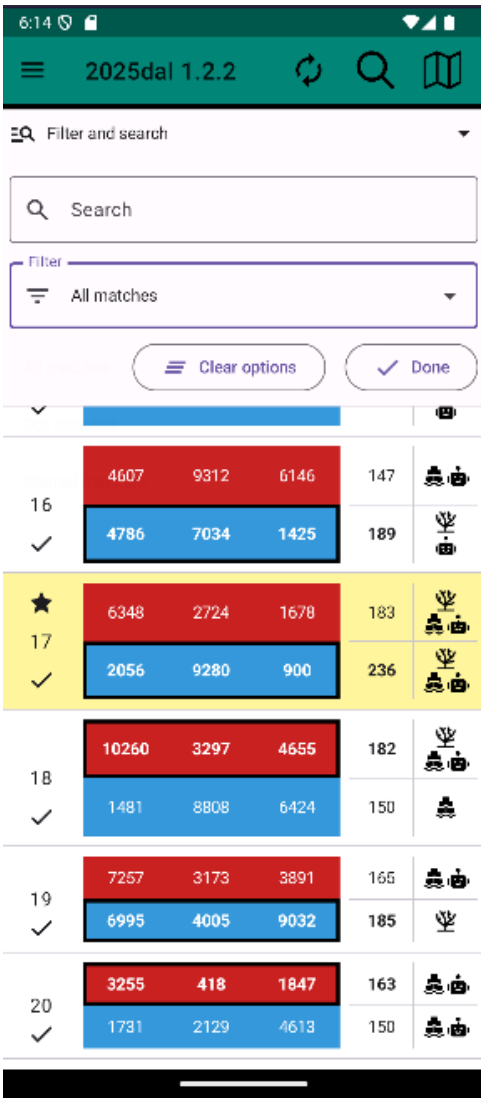


User Preferences

By pressing “Edit” on the Preferences page, users can change which datapoints are displayed in the app. This feature is important because Viewer has multiple users—one user may want to

see more data on a team's average total points, while another may want more data on a team's endgame. Viewer saves the selected datapoints in a file stored in the device's Downloads folder, so the settings are kept after closing or updating the app. Viewer also supports multiple profiles, allowing users to have their own default list of datapoints. During the 2025 season, Viewer's preferences and user settings were rewritten from Kotlin and XML to just Jetpack Compose.

Match Schedule

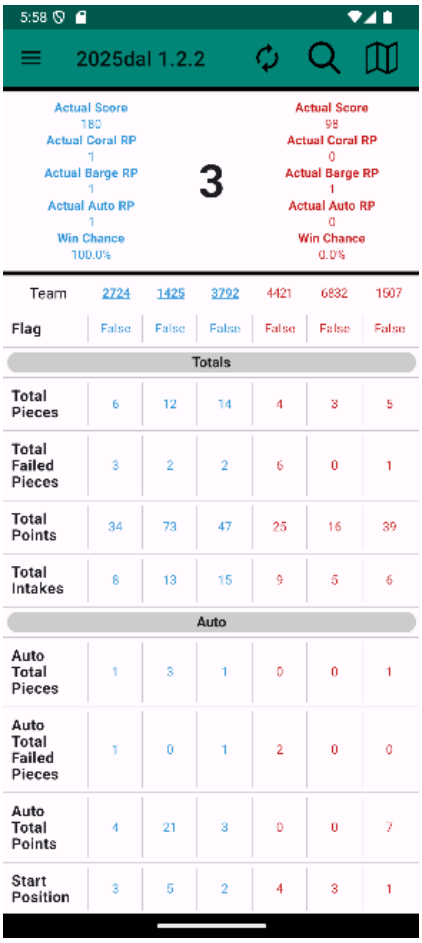


Match Schedule

The Match Schedule page displays the match schedule, including which teams are in each match, the total match score, and Ranking Points earned. Viewer shows predicted data for unplayed matches. A checkmark icon appears below the match number if the match is completed, and a clock icon if it has yet to be played. The winning alliance is marked with a black border. A robot icon is shown if the alliance earns the Auto RP, a coral icon if it earns the Coral RP, and a ship icon if it earns the Barge RP.

Users can use a search bar to filter matches by team number and go directly to that team's Team Details page. A drop-down menu allows filtering by All Matches, Our Matches, or Starred Matches. Users can star a match by long-pressing it; a star icon will then appear above the match number. If a team is starred, all matches they play in will be highlighted. The list of starred matches is saved to a file in the Downloads folder, so it is not reset when the app is reopened.

Match Details



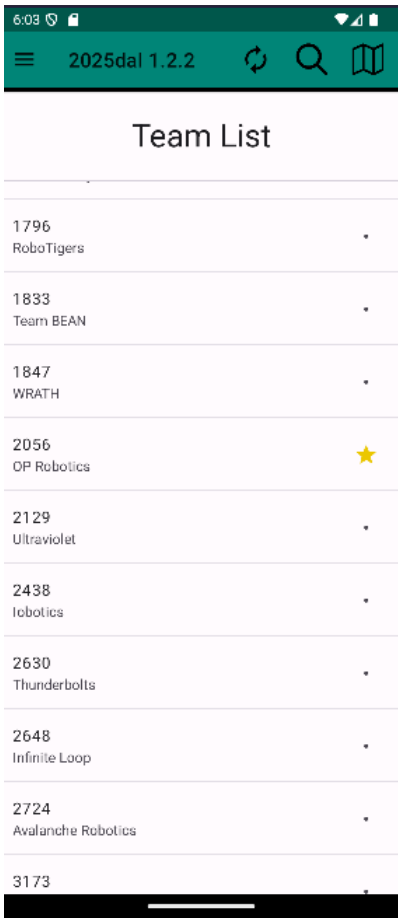
Match Details

After tapping a match on the Match Schedule page, the match's Match Details page opens. If the match is unplayed, the displayed datapoints will be predictions and Objective Team data averages. If the match has been played, the datapoints show the actual data collected by our scouts. The teams in the winning alliance are bolded and underlined.

In the header, alliance-specific data is displayed (e.g., predicted score). In the data section, a table shows team data. Users can access multiple screens through shortcut buttons to improve navigation. For example, clicking a team number goes to that team's Team Details page. Long-pressing Current Avg RPs opens the Rankings page. Long-pressing rankable datapoints

shows the rankings page for that datapoint. Long-pressing a datapoint from the Auto category opens the Auto Paths page for that team.

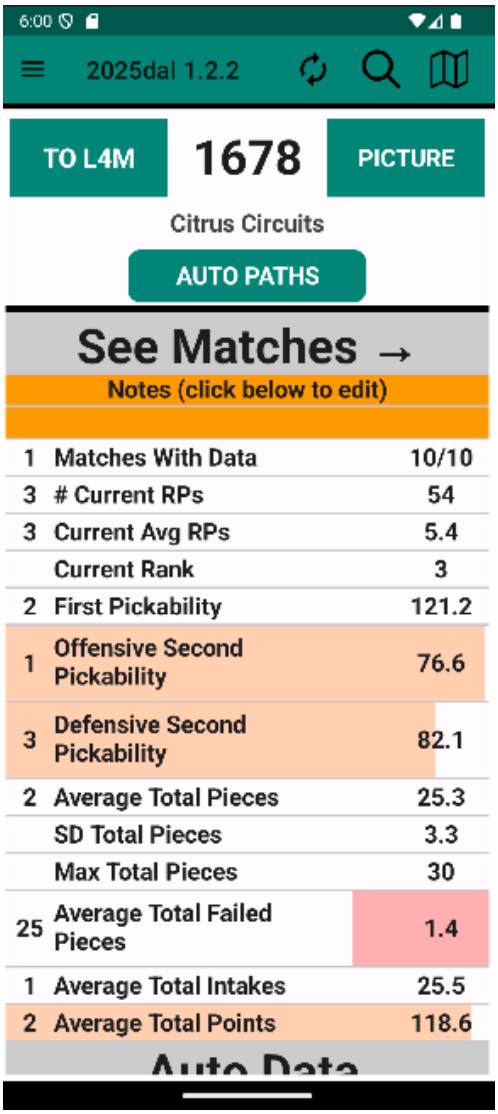
Team List



Team List

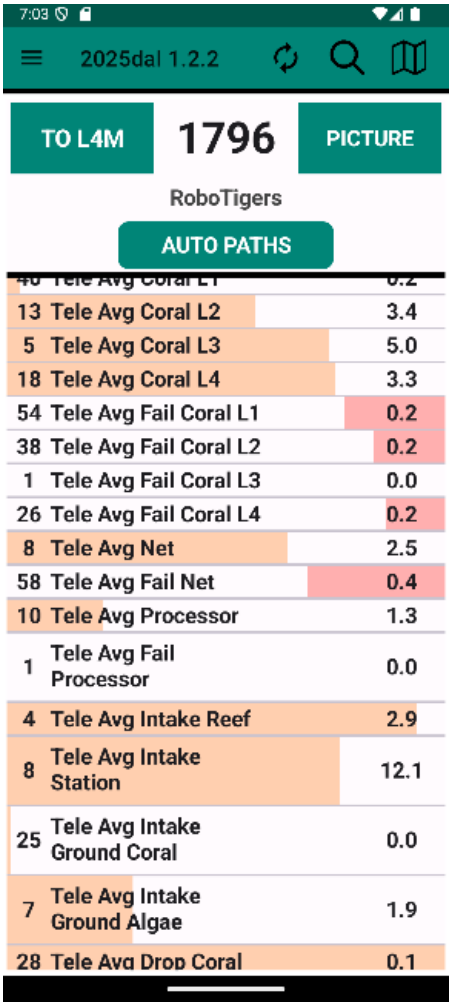
The Team List page lets users view all teams participating in the event. Users can star teams by tapping the area to the right of the team number. They can also go to the Team Details page by clicking on a team. On the Match Schedule page, matches with a starred team are highlighted in light yellow. If a match has more than one starred team, it turns a darker shade of yellow.

Team Details



Team Details

Every team at the competition has a Team Details page that shows calculated datapoints, such as averages, standard deviations, and more. Rankings appear in the left column, displaying the team's rank for each specific datapoint compared to other teams.



Data Bars

On the Team Details screen, users can display colored data bars behind each datapoint to visualize how the team compares with all other teams at the competition. For datapoints like incap time and fouls—where higher values indicate worse pickability—the data bars go from right to left and are colored red. Users can also navigate to the team's matches by clicking the "See Matches" header.

Auto Paths

Auto Paths was a feature added for the 2023 Championships after strategists realized how important it was to have a compatible auto. This feature visualizes the Auto Paths of teams,

starting with their most common autos and allowing users to scroll through their other recorded paths. We've continued to use and improve this feature over the past two years.

It is a screen in the Viewer app that records intake locations and scoring positions collected by scouts. This year, there were 12 possible intake locations and 48 possible scoring locations—one for every location in the reef. Each reef intake and scoring position is calculated using trigonometry and the angles of the hexagon.

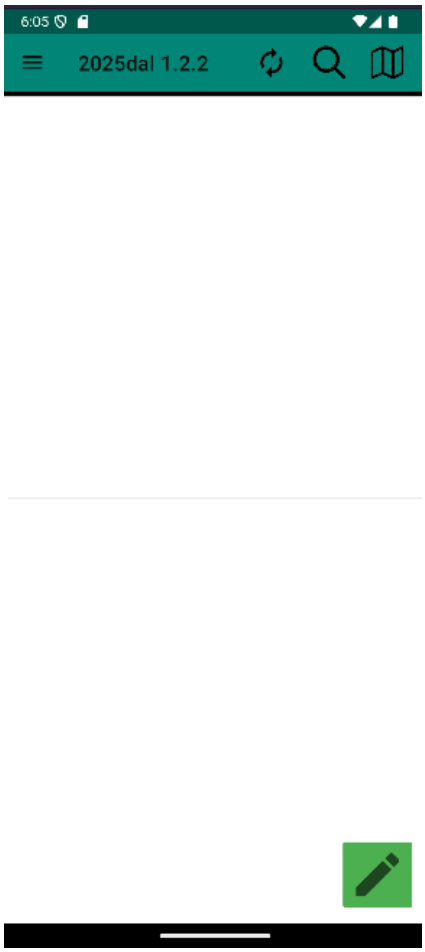
Additionally, this screen includes several other features:

- **Auto Path Notes**: Stand Strategist notes from the Stand Strat app transferred to Viewer.
- **Action Timeline**: A list of every action performed in the match.
- **Coloring**: If the team intakes or scores coral, the path and icon are red. If the team intakes or scores algae, the path and icon are blue. The later the action occurs, the more transparent the path becomes. At the starting position, the color is green if there is a preload and gray if there isn't.
- **Numbering**: The map shows auto intake or scoring successes over attempts and the order in which these actions occurred. For example, if it displays *x/y* over the reef position *F1 L1*, the team attempted to score coral *y* times and succeeded *x* times at that position across all their Auto Paths.
- **Play-through**: Buttons allow users to clear the path and step through it forward or backward.
- **Line Opacity**: As the auto path progresses, the line becomes more transparent and lighter.
- **Navigation**: Users can navigate Auto Paths by swiping or pressing the arrow buttons.

Action Timeline *Auto Path Notes*

Changes this year included the play-through feature, UI and coloring updates, and new calculations for offset locations. Auto Paths are used to review autonomous routines throughout the competition and are an essential part of Viewer for determining compatible alliance partners and developing advanced strategy.

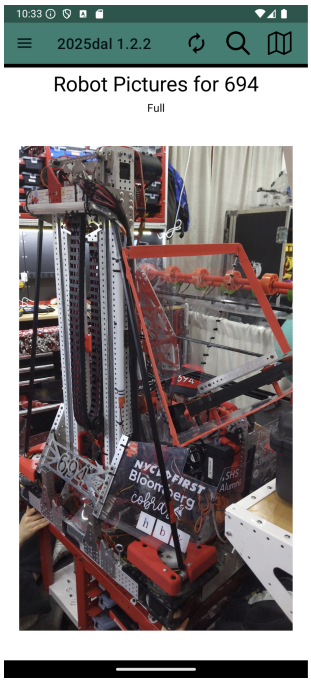
Team Notes



Team Notes

On the team list, users can long-press a team to open the team notes page. They can write notes by clicking the pencil button in the bottom right corner of the page.

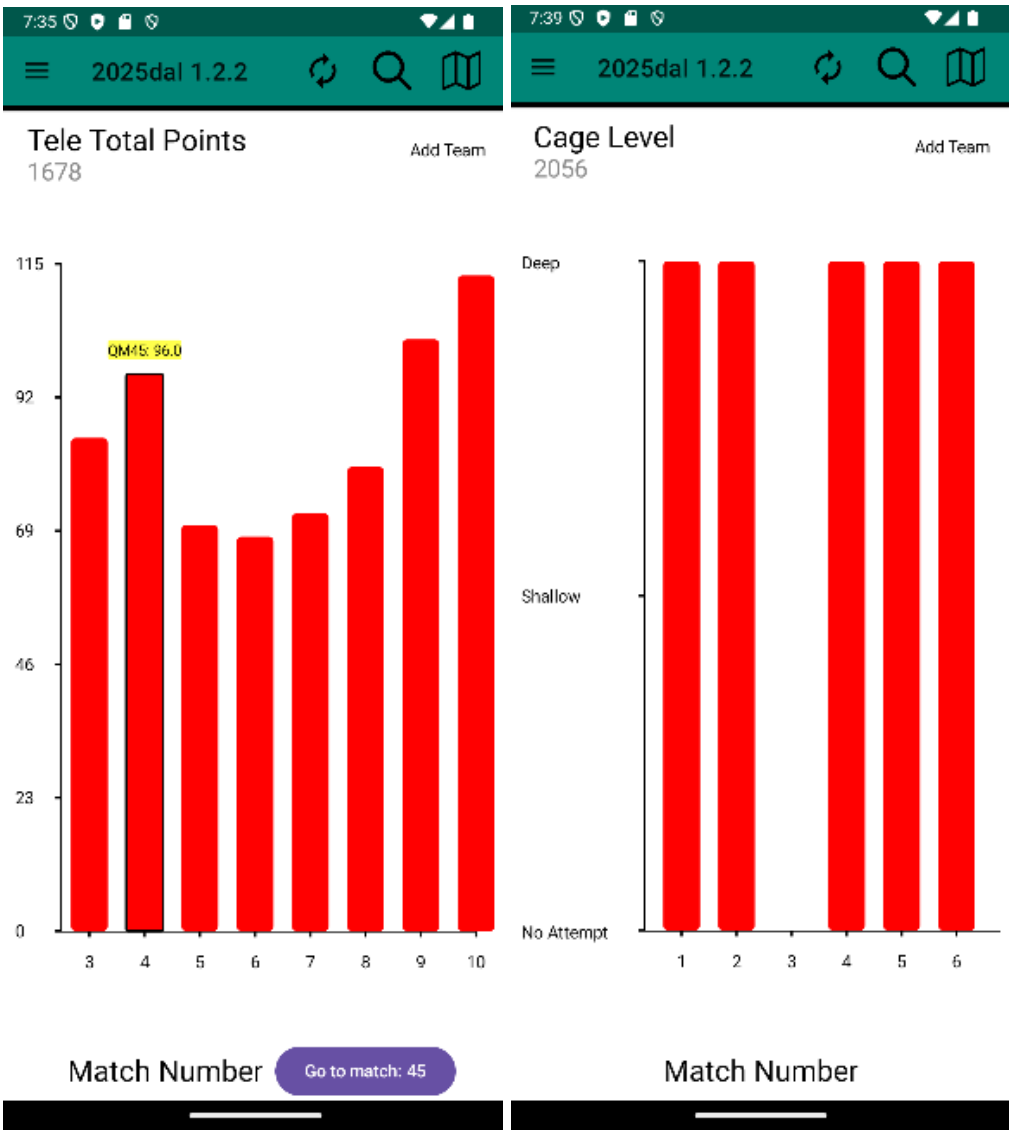
Robot Images



Robot Images

On a Team Details page, users can view images of the team's robot by pressing the Pictures button. Each team has multiple photos showing the robot from different angles. The Pit Scout takes these photos using the Pit Collection app, and we manually transfer them to other phones by connecting the phones to the Server.

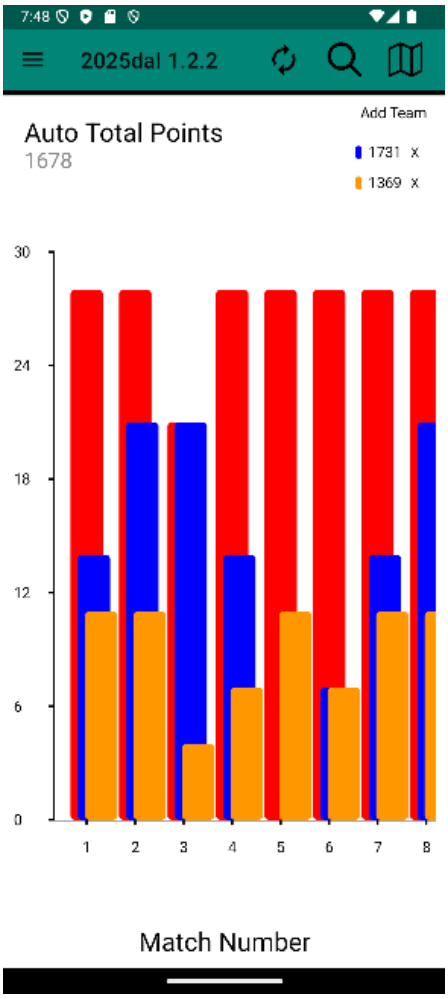
Data Graphs



Data Graph for Tele Total Points and Cage Level

Tapping a datapoint on the Team Details page opens its corresponding TIM (Team-in-Match) Data Graph, showing a bar chart of that datapoint across all the team's matches. Pressing a bar reveals the specific qualification match and the exact recorded value. A prompt then appears, offering a quick link to navigate directly to that match's details page.

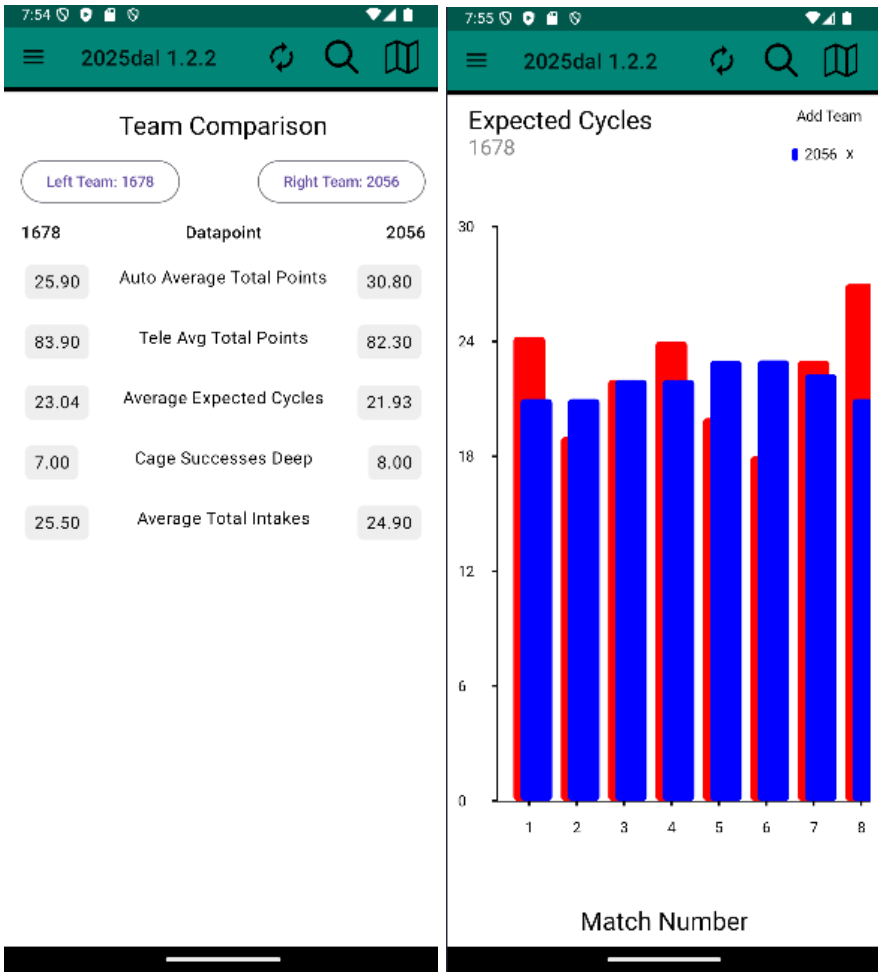
Multi-Team Graphs



Data Graph for Auto Total Points for teams 1678, 1731, and 1369

This year, we added a useful feature to the Graphs page: the ability to add additional teams. Pressing the “Add Team” button opens a dropdown menu of all available teams. Selecting a team adds it to the list and displays its data alongside the existing chart for easy comparison.

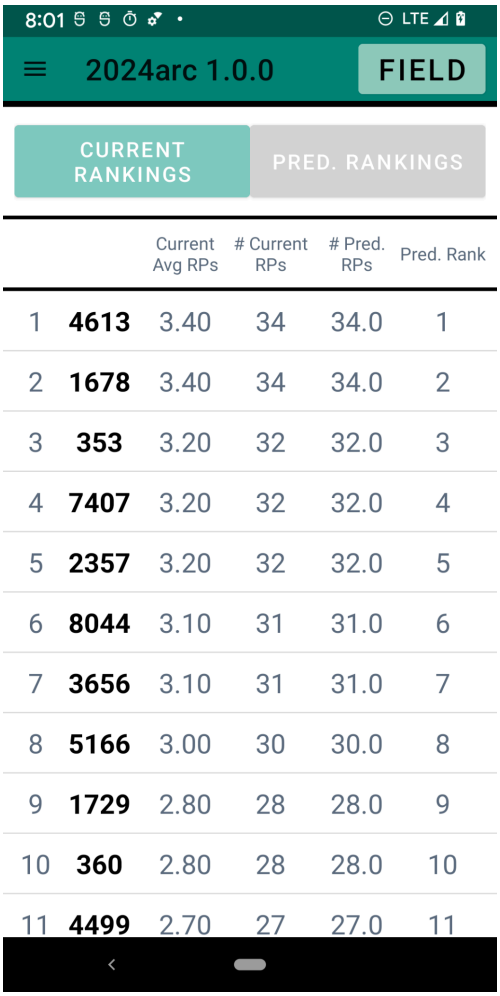
Team Comparison Page



Team Comparison page and Graph navigated from Team Comparison

This year, we added a Team Comparison Page that offers a quick and easy way to compare key data points between two teams. Users can select teams using the buttons on the left and right sides of the screen, which display a dropdown of all the teams at the event. The page shows values for Auto Average Total Points, Teleop Average Total Points, Average Expected Cycles, Deep Cage Successes, and Average Total Intakes. Tapping on any datapoint navigates to the corresponding graph, displaying data for both teams side by side for deeper analysis.

Team Rankings

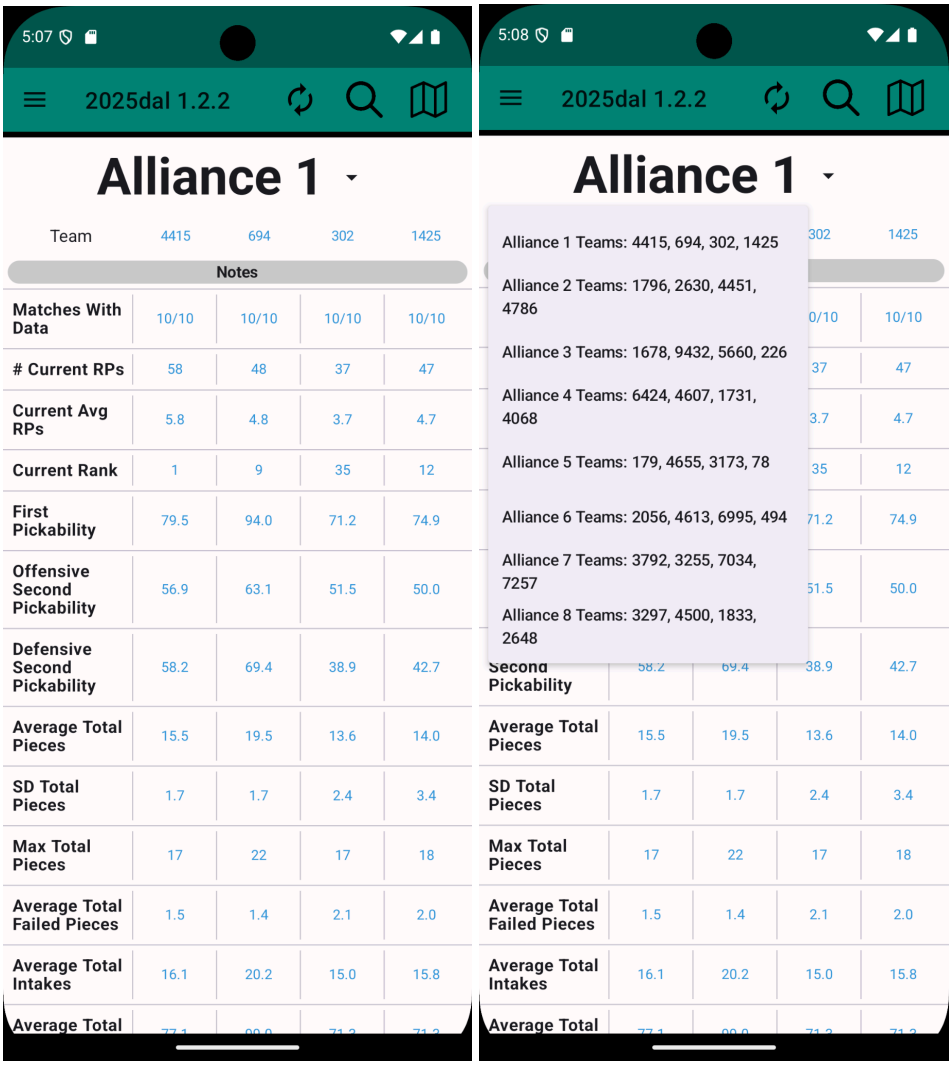


		Current Avg RPs	# Current RPs	# Pred. RPs	Pred. Rank
1	**4613**	3.40	34	34.0	1
2	**1678**	3.40	34	34.0	2
3	**353**	3.20	32	32.0	3
4	**7407**	3.20	32	32.0	4
5	**2357**	3.20	32	32.0	5
6	**8044**	3.10	31	31.0	6
7	**3656**	3.10	31	31.0	7
8	**5166**	3.00	30	30.0	8
9	**1729**	2.80	28	28.0	9
10	**360**	2.80	28	28.0	10
11	**4499**	2.70	27	27.0	11

Team Rankings Page

Users can access the Team Rankings page through the navigation sidebar. By pressing the option they want, they can toggle between current and predicted rankings.

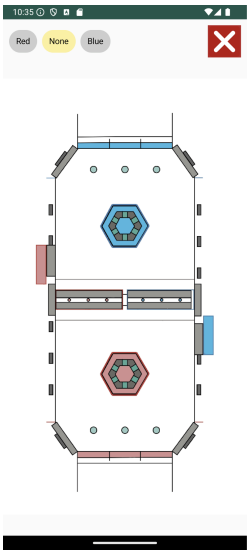
Elim Alliance Details



Team Alliance Details Page (Left) and Playoff Alliance Selection Drop-Down Menu (Right)

The Elim Alliance Details page was reimplemented this year for use by our strategy mentors at the Championship. It displays the team-in-match data for each of the alliances in the playoffs. The datapoints shown are identical to those on the Match Details page. Users can select which alliance they want to view from a dropdown menu.

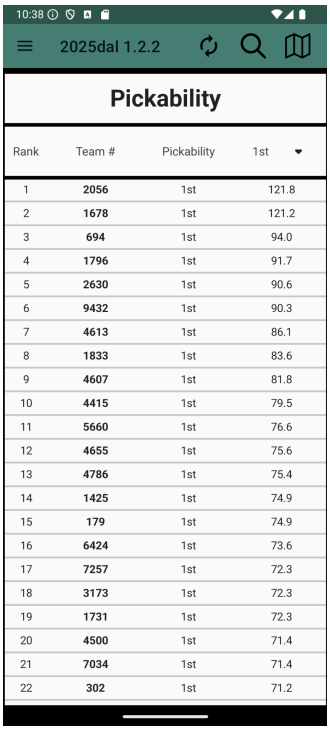
Field Map



Field Map

In Viewer, users can view the Field Map by pressing the FIELD button at the top right. This view shows possible starting positions, driver stations, and the starting positions of the Coral. Users can switch between the Red and Blue alliances to reverse the map orientation.

Pickability

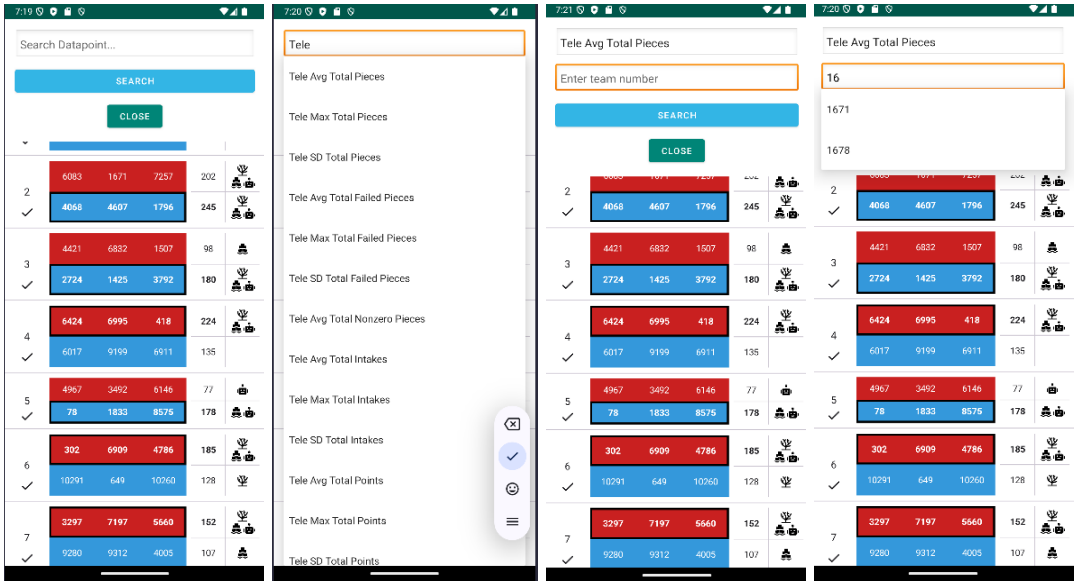


Pickability Page

Users can rank teams based on pickability metrics such as 1st, 2nd, Defensive 2nd, or Scoring 2nd. These values are calculated using preset weightings that reflect what is most important for a chosen robot. Users can switch between the different pickability metrics using the dropdown in the top right. Clicking on a team number opens that team's Team Details page.

Utility Features

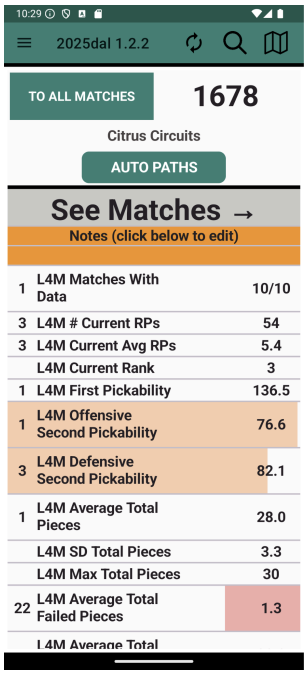
Datapoint Search



Datapoint Search

New this year, the always-visible toolbar at the top of the screen includes a magnifying glass icon next to the field map and refresh buttons. Tapping the icon opens a search bar that lets you find any datapoint in the app. The search predicts your query using both substring and fuzzy matching. After selecting a datapoint from the dropdown, a secondary field appears, prompting you to enter a team or match number, depending on the datapoint. Once submitted, the app navigates to the relevant page and automatically scrolls to the corresponding section.

Last Four Matches



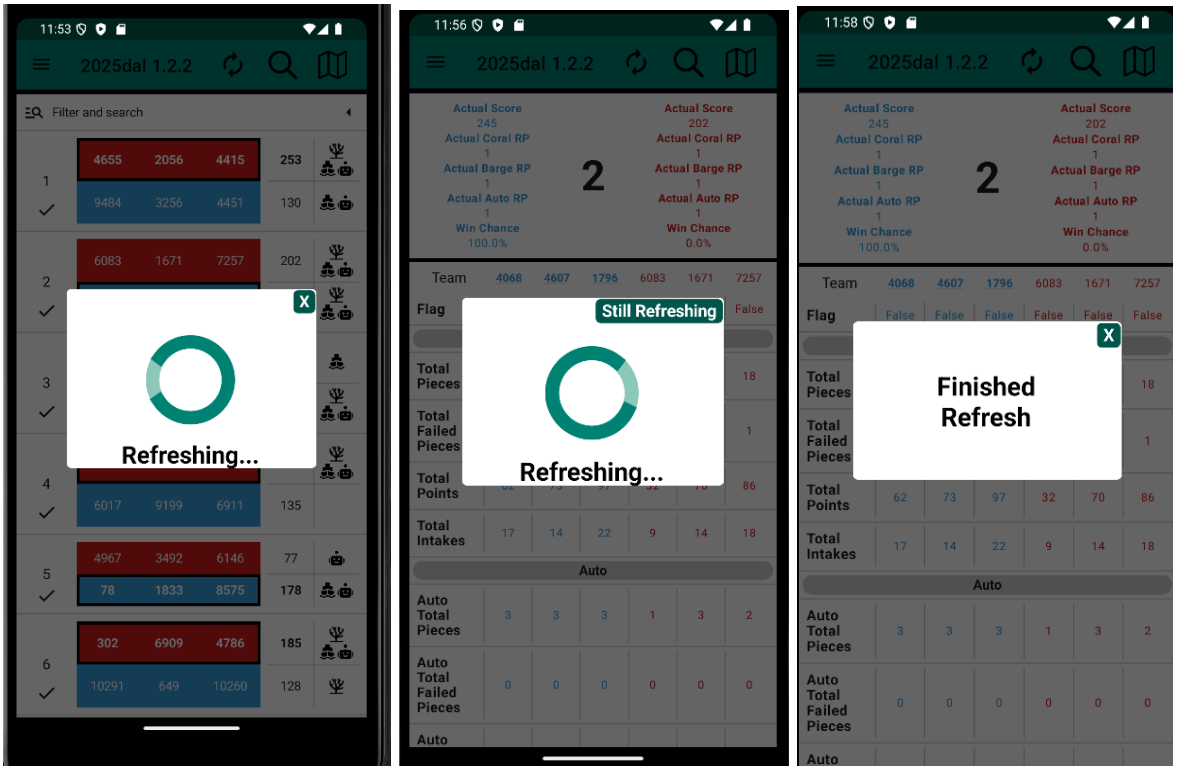
Last Four Matches

On a Team Details page, users can toggle between datapoints calculated from all matches and the same datapoints calculated only from a team's last four matches. Strategists use this feature to analyze teams that have improved during the competition and to evaluate them based on their most recent performance.

Data Refreshing

Viewer's data refreshes automatically by fetching data and updating caches at set time intervals. When the data refreshes, Viewer runs callbacks on all active pages to update the UI.

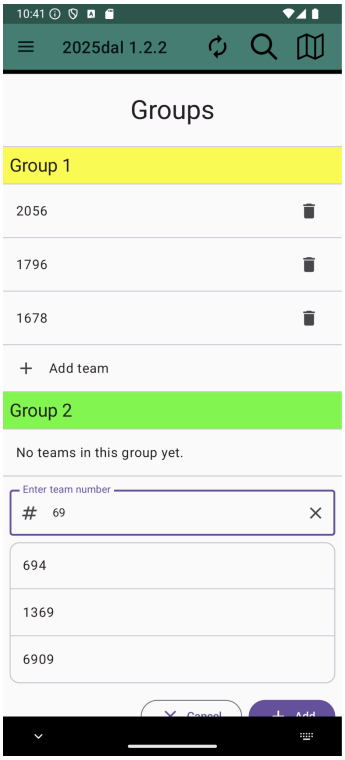
This year, we added a button to manually refresh the data shown in the Viewer app. By pressing the refresh icon, the app begins updating data for all teams and matches by pulling data from Kestrel. When pressed, a pop-up appears to inform the user that data is reloading. This pop-up can be closed, and another pop-up will appear when the refresh is finished. If the button is pressed while a refresh is already in progress, a different pop-up will notify the user that the data is still refreshing. These pop-ups appear automatically on any page.



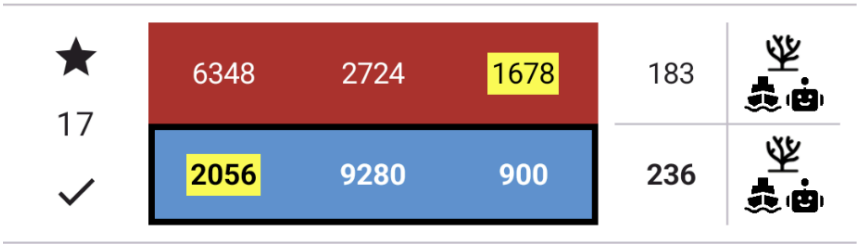
Refresh Button Clicked *Still Refreshing* *Finished Refresh*

Groups

First developed the year prior, Groups allows users to put teams into color-coded groups that are highlighted in the match schedule. Users can organize teams into groups for any purpose, such as marking teams to compare for picklist positions or identifying teams that may affect matches important for rankings.



Groups page



A match with teams from a group

Predictions

During competitions, Server makes predictions on future match scores, ranking points, and win probabilities, as well as final rankings for each team. Predicted Alliance In Match (AIM) data is calculated using Objective Team data and data from The Blue Alliance (TBA). This includes each alliance's chances of earning each RP, their final score, and their probability of winning.

This year, score predictions were kept simple, as each scoring method had a fixed point value. In general, we multiplied a team's average in each scoring category by that category's point value.

All three RPs were also calculated using simple logic:

- **Auto RP**: We checked if the sum of the auto leave rates for all three robots was greater than or equal to 2.5, and if the total average auto coral scored was at least 1. If both conditions were met, the alliance was predicted to earn the RP.
- **Coral RP**: We summed the average auto coral values across all levels. If the total exceeded a certain threshold—25 at Champs—the RP was predicted.
- **Barge RP**: We checked if the sum of the expected climb values for all alliance members was at least 14 (or 16 at Champs).

The biggest change to predictions this year was the introduction of a brand-new AI system for match predictions. For more details, see the section on DoozerNet.

Server

The Server is the Back-End data processing system that takes in raw scouting data and provides the Viewer, Picklist, and other visualization apps with accurate data and strategic insights. As matches progress during a competition, Scouts collect data that is then sent to our server laptop. In addition, the Server collects certain data from The Blue Alliance because it is a more reliable and consistent source of information for specific game actions. The Server runs many calculations on the data and then stores both the raw and calculated data in a local MongoDB database on the designated laptop. When the Server is done processing, the data is uploaded to a MongoDB database in the cloud, which is then accessed by our Kestrel web API. Finally, the Viewer app requests the data from Kestrel.

Schema

The schema serves as the blueprint for our data structure within the scouting system. This standardized framework is used across all our applications and is saved in YAML files, allowing developers to modify data fields from a single source. This approach enhances code reusability across different FRC games and supports adjustments we may want to make during the season. By eliminating hard-coded data fields and replacing them with schema-defined fields, we can make quick and simple changes, giving us more time to focus on other priorities. Additionally, the schema links calculated data fields to fundamental calculations (like averages and counts), which helps developers quickly create new calculations without needing to write extra code. Developers can adjust the weighting for calculations involving other metrics without modifying the server code. This flexibility enables picklist strategists to easily update weight values during competitions. As a result, any data field errors (such as typos) can be corrected quickly during the event.

Calculations

Calculations are performed on raw data collected through our apps. Processing of this data is split into multiple calculation files, determined by the category of the data being analyzed. Each

calculation file has its own schema file and collection in the database. All calculations are subclasses of the parent class BaseCalculations, which includes the calculation methods that most calculation files use. Each calculation file also has a run function that checks the database for new changes, calls other functions within the file to create new data, and then writes these changes to the database. Calculations include an LFM (last four matches) datapoint for objective data, which shows a team's performance in their most recent four matches. The server.py file imports these calculations in the order listed in calculations.yml and simply calls the run() function to execute each calculation.

QR Decompression

Our system is designed to process and interpret QR codes that represent collected match data. It handles two types of QR codes—objective and subjective—which are structured differently depending on the type of data they contain. The system first identifies the type of QR code and extracts key metadata, such as the match number, scout name, and team number.

At a high level, the process begins by reading and applying a predefined schema that defines how data is encoded within the QR codes. The system then decompresses this encoded data by translating shorthand values into readable and meaningful information. Special logic is included to handle structured data such as timelines, which record sequences of actions during matches.

The system also includes safeguards and correction mechanisms. For example, it can detect and correct inconsistencies in action sequences and handle complex compressed formats using specialized templates. This structured and modular design ensures accurate data extraction and supports downstream analysis or storage in a database.

Consolidation

Our consolidation methods combine data collected by multiple scouts on one robot in a match. Because different scouts may report slightly different values for the same robot, we aim to estimate the most accurate final values. After we decompress QRs into unconsolidated Scout

in Match (SIM) data, the system considers all SIMs for each robot in every match and applies a variety of logic to combine every datapoint. Scouts report several types of data: numbers (such as how many times something happened), true/false responses (such as whether a climb was successful), and category choices (such as the climb level). Our method handles each type of data slightly differently:

- Numbers: We look for agreement among scouts. If a mode exists, we use it. If not, we calculate a weighted average, with weights equal to the Z-score of each datapoint to give less weight to outliers.
- True/False values: We use the mode value.
- Categorical choices: We use the mode value. If there's no mode, we take the equivalent of a "None" value (e.g., if for a climb, two scouts report None and another two report Deep, we take None).

After the data is run through this consolidation system, we can calculate the Objective TIM data, which is then used for all subsequent calculations.

Team and TIM Data

Objective

Our TIM (Team In Match) data processing system is designed to transform raw, consolidated data into meaningful insights on match-by-match team performance. For a full list of objective TIM and Team datapoints, see the Codebook.

After consolidation, we calculate how many pieces each robot scored or failed to score in each location, intake locations, their reef face scoring in auto, incap time, and more. Next, we calculate some aggregate metrics, such as expected cycles or total piece breakdowns for auto, tele, and endgame. We also compute point values for each section of the match.

These objective TIM datapoints are then used to calculate objective Team datapoints, including averages, standard deviations, percentages of success, medians, modes, and maximums of all the objective TIM data, as well as data from the robot's last four matches.

Subjective

Subjective Scouts gather important match data that Objective Scouts do not. This includes agility and field awareness scores, which are recorded as subjective TIM datapoints. In addition, Subjective Scouts track details such as whether a team was tippy in a match and how much time remained before they attempted to climb. These datapoints are collected for each match and can be aggregated by team.

Subjective team datapoints—such as field awareness, agility, driver ability, defensive ability, and proxy ability—are calculated by aggregating the corresponding subjective TIM datapoints. For each team datapoint, we find all corresponding TIM datapoints and calculate an average or rate, depending on the datapoint.

For boolean variables, like was_tippy or can_cross_barge, we assess whether a team is capable of performing the action. If they can, the value is recorded as true. If they cannot—or if none of their TIM datapoints contain a true value—the entry is marked as false. Other TIM datapoints are assigned scores, which are averaged to determine final ratings.

Auto Paths

Auto Paths calculation tracks every action a team performs during the autonomous period, including scoring and intake actions. By analyzing the sequence of these actions, we can identify the general path a robot follows during auto. Our strategists primarily use this data to determine whether our autonomous routines are compatible with those of other teams. Auto Paths data also plays a key role in Pickability calculations and helps evaluate a team's overall ability.

To calculate Auto Paths, timeline data from various Scouts is consolidated, removing all non-auto actions and creating a single chronological list of autonomous actions. Instead of listing auto paths by match, similar paths are grouped together to assess the success rates of each unique path per team. Paths are matched based on starting position and intake/scoring attempts, as these actions define each auto path. By categorizing paths based on attempts rather than successes, we can determine success rates for scoring.

This year, we made minimal updates to Auto Paths. We added a feature that tracks whether teams have an auto compatible with ours (central starting position + ability to score in the face closest to the barge), making it easier to identify alliance partners with compatible auto routines.

Pickability

Pickability is a metric used to presort teams before strategists refine the picklist order. Each team has a first and second pickability, which estimate their suitability as a first or second pick, respectively. Pickability is typically calculated by adding weighted values from specific data points. In most cases, regression analysis is used to determine the weights for scoring actions.

After an event, teams are categorized into perceived ranks—such as high, medium, or low. Several multivariate linear regressions are then run to identify the best model for predicting each team's assigned rank. The coefficients from the selected model serve as weights for the relevant data points. Once these are applied, pickability values are calculated for each team and ranked from best to worst across all estimated models. A blind comparison is then conducted to determine which model produces the most logical rankings. The chosen model is used for future competitions and covers both first and second pickability metrics.

Additionally, all pickability metrics include a version based on the last four matches. This provides insight into a robot’s recent performance rather than its overall track record. These metrics are especially useful when a robot performs poorly in its early matches.

First Pickability

This year, our first pickability metric estimated the average points robots scored in their matches. All scoring actions were considered, with their weights assigned based on their respective point values. However, reef scores were standardized at 4.5 points regardless of level, while processor scores were set at 2 points (most human players achieve near-perfect accuracy when scoring in the net).

Second Pickability

Since the second pick in this year's game can play either offense or defense, we have two separate second pickability metrics: offensive and defensive.

The offensive second pickability measures the additional points a second pick can contribute to our alliance while complementing our gameplay style. This value is calculated by summing the point contributions from the autonomous, teleoperated, and endgame phases. Autonomous scoring accounts for how well a team's starting position and actions align with our auto paths. Teleoperated scoring evaluates the team's most efficient scoring method, excluding L4 scoring (since we typically score in L4), and estimates their cycle contributions based on scoring within that game element. Endgame scoring is determined by the team's average climb points. The final offensive second pickability score is the sum of these three components.

For defensive second pickability, we maintained a weighted sum approach but incorporated additional data points to reflect our strategists' emphasis on robustness and driver ability. At the start of each competition, a pit scout evaluates all robots based on mechanical and electrical robustness, as well as overall complexity. These factors, along with negatively weighted incap time, driver agility, and driver field awareness, are used to determine defensive second pickability. The weights are established using the regression analysis detailed above, with driver agility receiving the highest weight, followed by incap time, electrical robustness, mechanical robustness, and driver field awareness.

Scout Precision

Scout Precision Rating (SPR) measures how closely a Scout's reported scores match the actual scores from TBA, helping teams evaluate the reliability of their scouting data. Since TBA only provides total scores for alliances—not individual teams—the system analyzes combinations of Scouts rather than individual performance in isolation. In our Scouting System, each alliance has nine Scouts, with three Scouts assigned per robot. This setup creates 27 possible Scout combinations per match. The reported scores from each combination are totaled and compared with TBA's official scores.

To understand how much a specific Scout contributed to the accuracy or inaccuracy of the data, consider a Scout named X. To calculate X's SPR, the system starts with a match that X scouted. It forms combinations where X is grouped with two other Scouts—each covering the other two robots on the alliance. For each of these combinations, the reported scores from all three Scouts are added together to create a theoretical alliance score, which is then compared to the official TBA alliance score. The difference between the two is the error for that combination. For instance, if Scout X reported 12 points, Scout Y reported 31 points, and Scout Z reported 10 points, and the TBA score was 50, the error would be 3 points. The system repeats this for all valid combinations that include Scout X and Scouts who covered the other two alliance robots. The average of all these errors becomes Scout X's average combination error for that match. This same process is repeated for every Scout in the match to calculate their individual average combination errors. This is the Scout-In-Match (SIM) Precision—a measure of how accurately Scout X performed in that particular match.

A Scout's overall SPR is the average of their SIM Precision scores across all matches they participated in during a competition. Lower SPR values indicate higher accuracy and more trustworthy scouting.

Predictions

During competitions, the Server predicts future match scores, ranking points, win probabilities, and final team rankings. The Predicted Alliance In Match (AIM) data is generated using Objective Team data and TBA data, which include an alliance's likelihood of earning ranking points, final score, and probability of winning.

This year, score predictions were relatively straightforward, as each scoring method had a fixed value. Generally, we calculated a team's predicted score by multiplying its average performance in each scoring method by the corresponding point value. However, human players scoring algae from opponent processor scores introduced complexity, so we chose to exclude that factor.

All three Ranking Points (RPs) were fairly simple to determine. For the auto RP, we checked if the combined auto leave rates of all three robots met or exceeded 2.5 and if the total average

auto coral count was at least 1. If both conditions were met, the alliance was predicted to earn the RP. For the coral RP, we summed the average auto coral values across all levels and compared the total to the RP threshold. The barge RP was determined by summing the expected climb values for all alliance members and checking whether the total met or exceeded 14 (or 16 at championships).

By far, the most significant change to our prediction process this year was the introduction of a new AI system for match predictions. For more details, refer to the DoozerNet section.

Interactions with TBA and Statbotics

We collect data from The Blue Alliance (TBA) and Statbotics using their respective APIs. While this data isn't extensively used in our core server calculations, we've developed infrastructure to pull and process key datapoints, validate our scouting data, and provide our Strategy subteam with metrics for pre-scouting before competitions.

From TBA, we extract datapoints like auto leave and climb level and cross-reference reported scoring data with our own scouting results. We run automated scripts to pull data on autos, scoring, and climbs for all matches across specified competitions. The Strategy subteam uses this by-match data to track team performance over time. Additionally, we aggregate the data to calculate team-level statistics, such as average match score, win rate, leave rate, and climb rate. The last two metrics are particularly valuable to our SEALs team, who collaborate with alliance partners to ensure they can leave and climb during our matches.

Although our core system doesn't directly integrate data from Statbotics, our Strategy subteam heavily relies on it for pre-scouting. We've developed dedicated scripts that extract team-level information from the Statbotics API, including EPA values, bonus RP estimates, and rankings at the state, national, and global levels.

Some parts of these scripts, like extracting EPA breakdowns or match data across all competitions for the season, are handled asynchronously. This design allows us to export data quickly, even when processing large datasets from entire seasons. The system is flexible

enough to pull data for individual events or broader date ranges, enabling us to analyze important matches that we couldn't scout live.

Data Validation

This year, we faced challenges with the accuracy of our scouting data, particularly at our first regional. During the Pinnacles Regional, inexperienced scouts and insufficient training led to an average error of around 10 points per match. In response, we enhanced our scout training and implemented a comprehensive data validation system to improve the reliability of our data.

To measure the accuracy of our scouting, we developed a calculation that compares our reported match data with the official results from TBA. This calculation is a simplified version of the Scout Precision. For each match, we sum the individual reported scores of all three alliance robots and compare this total with the official TBA-reported alliance score. The error is the absolute difference between the reported sum and the official TBA value. We calculate these errors for total match scores and also for subcategories like reef, net, and processor points in both autonomous and teleoperated periods, as well as total endgame points to verify climb accuracy.

Additionally, we monitor discrepancies between scouts' reports for the same robot to identify inconsistencies using the simple method of calculating the range of reported values for each datapoint. For example, for three scouts assigned to robot 1, the discrepancy for L4 scores = maximum reported L4 score - minimum reported L4 score. Our analysis revealed a recurring issue: scouts were often less accurate when our team, 1678, was on the field. This was likely due to the distraction of watching our own robot compete, leading scouts to focus more on cheering than scouting. To address this, we provided targeted training, which significantly improved the quality of data collected at subsequent events.

To automate data quality control, we developed a flagging system that identifies data falling outside of acceptable ranges. This system flags matches with unusually high scoring errors or significant discrepancies between scouts. Flagged data is reviewed by our Scouting Developers, who correct inaccuracies by re-scouting the affected matches. The thresholds for flagging are adjusted depending on the accuracy requirements of each event.

At the World Championships, we assigned a dedicated Data Validation Developer to review flagged matches in real time. This role involved re-scouting robots, correcting data entries, and blocklisting (excluding from calculations) unreliable QR codes (see the QR Override section).

Thanks to this data validation system and enhanced scout training, we greatly improved the accuracy and reliability of our scouting data throughout the season, even when individual scouts faced challenges.

Stand Strategist and Pit Data

To provide strategists with crucial qualitative information about robot performance and to create the most optimal Picklist, we rely on Stand Strategist and Pit Data. This data is first inserted into MongoDB, where it can be accessed by the Viewer or Picklist applications as needed.

Unlike the Scout apps, the Stand Strategist app does not use QR codes for data collection. Instead, we manually retrieve the data from devices using a USB cable and the Android Debug Bridge. Similarly, the Pit Collection app does not export data as QR codes. However, since these apps run on Google Pixel phones with cell service, they directly upload the collected data to MongoDB through the new Kestrel API.

The Server loop continuously checks the connection status of any Pit Phones or Stand Strategist Tablets and retrieves data from designated folders on these devices. Unlike data gathered by Objective and Subjective Scouts, we do not perform calculations on Stand Strategist or Pit Data. Instead, we focus on addressing any empty values and converting data points into the appropriate Python data types before uploading them to MongoDB. This year, robot images were automatically transferred from the Pit Phones to the Viewer Phones via the Kestrel API.

Transferring Data

QR Codes

We primarily use QR codes for transferring data from match collection to the server, which constitutes the bulk of our data transfer. Scouts format their collected data into QR codes, which are then scanned using QR scanners. These scanners upload the data directly to the database. Additionally, QRs can be manually pulled from tablets connected to the server laptop, as the tablets store QR data locally.

QR data is stored in a compressed string format and later decompressed into a readable arrangement using the decompressor Schema. Compressing QR strings reduces the amount of storage space needed, and Schema formatting ensures that the data remains compact and easy to manage.

This year, we implemented an improved QR override and blocklist script to allow modifications or exclusions of QRs from calculations. When executed, the script prompts the user to identify a QR based on the scout's name and match number. The user can then specify which datapoint to override and provide a replacement value, or choose to exclude the QR from calculations. These overrides are stored in a JSON file. Each time a server file is executed, it reads the JSON and applies the necessary overrides to the local database.

USB Connection

Most other data transfers are done manually via USB connection using the Android Debug Bridge. We transfer data from the Stand Strategist app through a direct USB connection to the server computer. Additionally, match schedules, team lists, and picklist JSON files are sent to all devices via USB. Our tablet cases support multi-device connections (up to 15 devices per case)—see the Hardware section for more details. Direct USB connections offer a simpler, more robust solution compared to other methods that may seem convenient but are actually more complex.

Cloud API

This year, the new Kestrel API has enabled us to transfer some data over the cloud. All data from the cloud database is transferred to the Viewer via Kestrel. Additionally, all Pit Collection data—including robot images—is now transferred through the cloud via the Kestrel API.

Exporting Data

Although all collected data points are viewable on the Viewer app, it's often more efficient to display everything on one screen during the picklist meeting, so all team members can easily reference it. To facilitate this, we developed a Python script to export both Team and Team-In-Match (TIM) data as CSV files, which can then be imported into the Picklist Editor for further analysis and used in post-competition calculations.

The script we use, export_csvs.py, pulls data from a variety of collections in the database, including TBA score breakdowns via the TBA API. It features a base class that formats the data into CSV format and writes it to a file, along with specific classes for each CSV type to be exported. Once all the necessary files are created, the script compiles them into a zip file and sends it via Slack.

After the competition, these exported files are invaluable for analyzing calculations, robot performance, predictions, data accuracy, scout accuracy, and other key metrics. While it’s technically possible to automate the transfer of this data directly to spreadsheets or similar software, we found that manually sending the data via Slack is more practical. An automated system would add unnecessary complexity without offering significant benefits.

Picklist Editor

The Picklist Editor is a tool designed to help create an informed picklist for competitions using our scouting data. The app displays the team list for the competition, along with statistics for each robot. Its primary feature is the ability to reorder teams based on their performance and pickability metrics.

Picklist Editor runs on Google Sheets and uses the Google Apps Script platform. With the official command-line interface, Clasp, it's possible to clone the scripts to a local development environment as JavaScript files. For development, we use TypeScript, a superset of JavaScript that provides static type checking and enhances the development experience. Clasp helps facilitate this process while working on the Picklist Editor. However, most features—such as team rank ordering, removing teams, switching pickability modes, and toggling to Last Four Matches datapoints—are implemented using Google Sheets formulas.

Pages

The Picklist spreadsheet is split into a couple pages, each with a specific use. During our picklist meetings, we switch between the pages to evaluate teams' performances in all aspects.

Main Editor

In the main editor, the first column lists all the teams in the competition, initially sorted by their Pickability ratings (based on either their first or second pick rank scores). Ranks are displayed in the second column.

Each row represents a team and displays their data. This data is pulled using VLOOKUP formulas from a separate raw data sheet, which is exported from our MongoDB database by the server at the end of each competition day.

In addition to the raw scouting data, strategists manually input additional data on the first day of matches. This includes subjective assessments of each robot's mechanical and electrical

robustness, as well as their complexity (measured by degrees of freedom). These data points are collected in the pits and during matches, and are stored in a separate spreadsheet. They are then used in the pickability calculations and exports.

To reorder teams on the picklist, the picklist operator can edit the ranking number (for example, changing a team's rank from 2 to 1.5 to move them between first and second). The scripts behind the Picklist Editor will automatically reorder the teams and update their ranks to whole numbers. The spreadsheet uses bubble sorting, starting from the top and working downwards.

A checkbox in the top-left corner allows users to toggle the displayed datapoints to reflect data from a team's last four matches. These datapoints are useful for evaluating a team's most recent performance. Unchecking the box will revert the display to the full set of collected data.

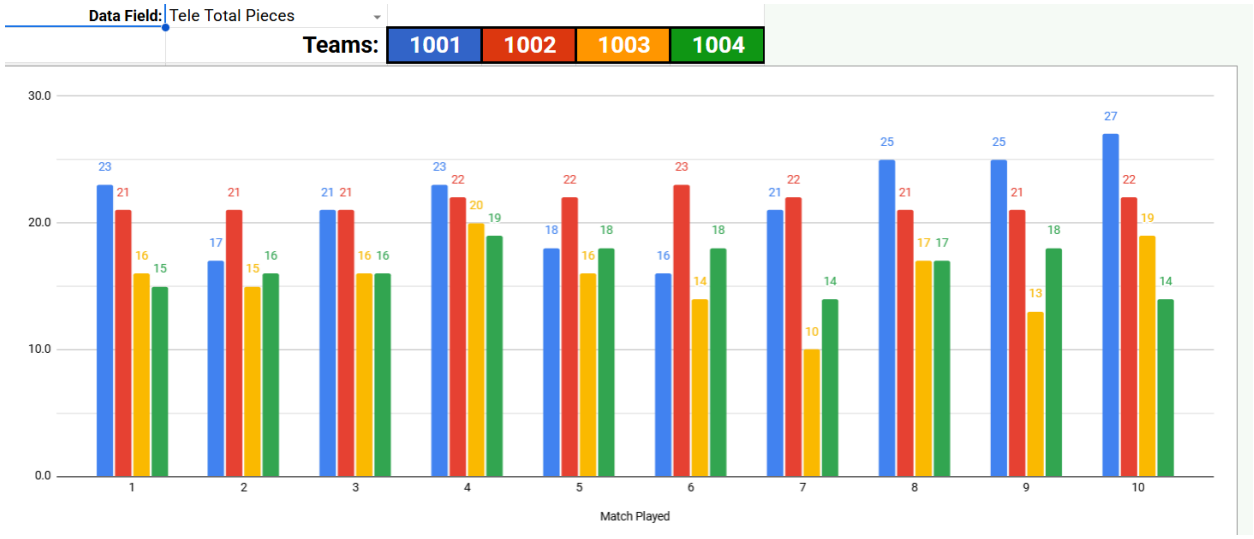
The datapoints displayed in the Picklist Editor and their order can change throughout the season based on feedback from students and mentors during picklist meetings. The editor is designed to easily add or remove datapoints during these meetings, but strategists try to determine all datapoints in advance to save time. To add a new datapoint, simply insert a new column, copy over the necessary formulas, and enter the datapoint name in the second cell of the column.

Data Field Index (Team Ra		112	104	344	108	336	356	5.0	6.0	7.0	48.0	26.0	52.0	114.0	105.0	327.0	80.0	326.0	366.0	397.0
Data Field Name		first_pick	defensive	offensive	electrical_robustnes	mechanical_robus	robot_complexity	auto_avg_	auto_avg_	auto_avg_c	auto_max_	auto_avg_to	auto_max_to	has_compati	driver_ability	matches_tipp	avg_defense	matches_pla	tele_avg_co	tele_max_c
	Rank	1st	Def. 2nd	Off. 2nd	Electrical Robustness	Mechanical Robustne	Robot Complexity	Auto Avg L2	Auto Avg L3	Auto Avg L4	Auto Max Net	Auto Avg Piece	Auto Max Piece	Comp. Auto	Driver Ability	Matches Tippy	Avg Def Rating	Defense Match	Tele Avg L1	Tele Max L1
1015	1	121.8	92.5	76.3	5	5	3.5	0.0	0.0	4.4	0.0	4.4	5	TRUE	2.2	5.0	0.0	0	2.9	11.0
1009	2	121.2	82.1	76.6	4	4	3.5	0.0	0.0	3.7	0.0	3.7	4	TRUE	2.3	0.0	0.0	0	1.5	7.0
1012	3	91.6	85.3	58.1	4	4	4.0	0.0	0.0	3.0	0.0	3.0	4	TRUE	2.7	0.0	0.0	0	0.2	2.0
1019	4	90.6	75.9	64.9	3	4	4.0	0.0	0.0	2.3	0.0	2.3	4	TRUE	2.3	0.0	0.0	0	1.2	6.0
1013	5	83.6	57.3	53.0	3	3	3.5	0.0	0.0	2.7	0.0	2.7	4	TRUE	0.4	0.0	0.0	0	3.2	8.0
1035	6	79.5	58.2	56.9	3	4	4.5	0.0	0.0	1.3	2.0	1.7	3	TRUE	0.4	0.0	0.0	0	0.2	1.0
1005	7	75.0	42.7	50.0	2	2	3.5	0.0	0.0	2.4	0.0	2.4	3	TRUE	0.0	3.0	0.0	0	1.1	4.0
1011	8	74.9	62.7	56.9	3	4	3.5	0.0	0.4	1.3	0.0	1.7	3	TRUE	0.7	0.0	0.0	0	0.5	4.0
1010	9	72.3	57.3	51.2	3	3	4.5	0.0	0.0	2.1	0.0	2.1	3	TRUE	0.2	4.0	0.0	0	0.5	2.0
1023	10	72.3	48.1	50.5	3	3	3.0	0.0	0.0	2.3	0.0	2.3	3	TRUE	0.6	4.0	2.0	1	1.7	6.0
1022	11	71.2	38.9	51.5	2	2	3.5	0.0	0.0	2.7	0.0	2.7	3	TRUE	0.3	4.0	0.0	0	0.0	0.0
1033	12	69.6	56.9	45.6	3	3	3.5	0.0	0.0	2.5	2.0	2.7	3	TRUE	0.5	3.0	0.0	0	0.3	2.0
1024	13	69.3	53.3	53.1	3	3	3.5	0.0	0.0	1.4	2.0	1.6	3	TRUE	-0.2	3.0	0.0	1	0.1	1.0

Main Editor

Team Comparison Graphs

When a match-by-match comparison between teams is needed, the Picklist Editor includes a graphing feature that allows users to compare the data of up to four teams for a single datapoint. These graphs can also display changes in data over multiple matches, making it easier to track team performance trends over time.



Team Comparison Graph Page

DNP’d Teams

To save time and streamline the picklist process, strategists identify teams that may not meet our performance standards or align with our preferred strategy and remove them from the list at the start of the meeting. Prior to the meeting, the Match Strategist, Stand Strategists, and strategy mentors discuss these teams and document them in a separate spreadsheet. To remove a team from the main list, simply type “DNP” or “d” in the rank cell. If a team is reconsidered later, it can be added back by checking its box on the DNP page.

Teams	Send back (click)
1004	☐
1005	☐
1006	☐
1007	☐

DNP page

Robot Photos

Many strategy team members find that viewing pictures and videos of a robot helps recall other details about its performance. This year, we stored robot images and match videos in a Google Drive folder and displayed them on a separate screen for easy reference.

Operation

Before 1678's picklist meetings, the picklist operator receives raw data in CSV format from the Server. During the meeting, the operator displays the Picklist Editor via a projector.

At the start of the meeting, the operator DNPs teams that were agreed upon earlier. Potential captains are identified using Statbotics simulations and moved to the top of the list.

The team review process starts with the first pick order from the ranking equations. Teams are reviewed top-down and compared on a pairwise basis. Attendees decide for each team whether it should be moved up the picklist and by how many places. Once the potential first picks (usually narrowed down to 10 or 12 teams) are ranked, those rows are hidden, and the process continues for the potential second picks.

In previous years, the main editor was copied to a Final Picklist sheet, which was modified throughout Day 2 of regionals. This year, we simplified the process by creating a sheet that strategists can use during Alliance Selection to track teams that have been picked.

	Team	Change
☑	~~1015~~	~~0~~
☐	**1009**	**0**
☑	~~1012~~	~~2~~
☑	~~1019~~	~~-1~~
☑	~~1013~~	~~-1~~
☐	1035	0
☐	1005	2

Final Picklist Page

Example Picklist

Here is an example picklist: 1678 Scouting 2025 Example Picklist

Kestrel

Kestrel is our third API, developed in 2025 to replace Grosbeak, which had become outdated and unreliable. Built using Python and the FastAPI library, Kestrel focuses on simplicity, maintainability, and flexibility. It serves as a bridge between our cloud database and Front-End applications, making it easier for apps like Viewer to fetch and update data.

Kestrel interacts with the cloud database via pymongo and provides endpoints for key data such as Team in Match (TIM), Team data, AIM, auto paths, and raw pit data. To meet the specific needs of Viewer, Kestrel ensures consistent data formatting across endpoints like obj_team and obj_tim, and offers specialized endpoints for unique collections like auto_paths.

The architecture of Kestrel is straightforward. Upon startup, the API connects to our database and loads various routers. Each router contains endpoints, similar to functions that are accessed over HTTP. Most routers include authentication, verifying that the request header contains the correct key. Once authenticated, each endpoint loads data from the database, performs simple processing, and returns the result.

We hosted Kestrel on DigitalOcean's App Platform, running 3 workers with a shared 2 GB RAM and a single CPU. While this might seem excessive for an API transferring JSON data, the inclusion of pit images required additional resources for smooth performance.

A significant upgrade this season is the auto-syncing feature in the Pit Collection app. Previously, data had to be manually transferred by connecting phones to the server. Now, phones automatically upload raw pit data and robot images directly to the cloud via Kestrel.

Kestrel is designed to be simple for developers, easy to maintain, and adaptable to seasonal needs. It ensures reliable and efficient communication between the back-end and front-end, improving workflows and reducing manual effort.

DoozerNet

DoozerNet is the prototype machine learning system created for this season. It currently includes a single model that predicts the winner of a match, which reached a peak accuracy of 95% at the Contra Costa Regional.

DoozerNet is composed of multiple components, most of which are written in Python using PyTorch for machine learning. The main system includes a storage bucket for trained models, an API for using the models to make predictions, a website for manual predictions, and a high-powered, GPU-equipped virtual machine for training the models.

The training script runs continuously during the competition, retraining models repeatedly, as each new piece of data is valuable when working with such a small dataset. After training a new model, it is evaluated, and if it performs better than the current model, it is uploaded to the storage bucket. The API is similar to Kestrel but significantly more complex, as it must run predictions using models retrieved from the storage bucket.

The model architecture evolved rapidly, but by Champs it was structured as follows: for each team in a match, their obj_tim data is combined with stand strategist data from each of their matches and placed into a list. For each team, an attention block combines all match data into a single representation. Then, another attention block combines the representations of teams on the same alliance. These two alliance vectors are passed into a multilayer perceptron, ending in a single output neuron that predicts the match outcome (0 for red winning, 1 for blue).

The web frontend for DoozerNet is written in Svelte using SvelteKit. It offers a predicted match schedule, where the model forecasts every match in a competition; a custom match predictor, where users can input red and blue alliances with any teams to see which is predicted to win; and a simulation runner, where users select their first and second picks along with the opponent captain, first pick, and second pick. With these options selected, all possible match combinations are simulated, and statistics are generated to determine the best possible picks according to the model.

Video System

The Video System is a crucial component of our data collection and strategic development processes. It enables match footage to be reviewed and analyzed for strategic planning, drive team feedback, and performance assessment during picklist discussions. The system was implemented as a solution to delays and inconsistencies in the upload times and camera positioning of official match recordings.

Video System Operators play a dual role as both video recorders and Objective Scouts. At the start of each day, they set up cameras and tripods in an elevated, centralized position that provides a clear view of the entire field. After each match, the footage is renamed according to the match number and transferred from the SD card to a hard drive. Some matches are also copied to USB drives for immediate review between qualification matches.

During elimination matches, videos are copied to USB drives and provided to strategists, allowing them to quickly analyze the footage and refine strategies for the next match. This timely access to video data supports more informed and responsive decision-making, helping the team adjust their game plan based on real-time footage.

Overall, the Video System plays an essential role in ensuring that reliable video data is available when needed, contributing significantly to both our strategic development and performance analysis throughout the competition.

Subteam Management

Subteam Management in Software Scouting is structured to support learning, collaboration, and strong preparation throughout the year. With 20+ regular subteam members working together to build a complex scouting system that requires 25+ scouts at competition, it's important to have a consistent procedure of training, building, testing, and operating throughout the season. New members start with training in either Front-End or Back-End, learning key skills like Kotlin, Python, Git, and how to use our system. During build season, team members work together to plan, build, and test scouting apps and databases. We run regular field tests to catch bugs and improve the system before competition. At competitions, Scouts, Developers, and Strategists all have specific roles, with clear shifts and responsibilities to make sure the system works well and all data is collected accurately.

Season Timeline

Before Kickoff — All veteran Software Scouting members train new members. They also prepare for off-season competitions by making code improvements and working on off-season projects. We inventory all materials before each season and order any needed supplies.

1/6 — All Citrus Circuits students watch the Kickoff broadcast and participate in a full-team discussion about what 1678 will attempt to do in the new season.

1/7 — The Scouting subteam meets with Strategy members to determine which datapoints are necessary to collect and which ones we want to display. For each datapoint on the final list, the following information is recorded:

- The data type
- A description of what the datapoint represents
- Some example values
- Which database collection it will be stored in

- Whether it will be collected as raw data or calculated from other datapoints
- (If it’s a raw datapoint) Who will collect it: Scouts, Subjective Scouts, Pit Scouts, Stand Strategists, or The Blue Alliance API
- Whether it will be displayed in the Viewer, and if so, how it will be visualized

1/7 to 1/13 — Back-End students update the Schema files to include the new datapoints for the season. Front-End students update the Match Collection and Pit Collection apps with the new data and UI designs.

1/13 to 1/30 — Back-End students update the calculation files to match the new Schema. Front-End students finish the first version of the collection apps and begin updating the Viewer and Stand Strategist apps to add new features and datapoints. The visual apps are often worked on at the same time as the collection apps to prevent conflicts during code merges.

1/30 to 2/27 — Software Scouting begins conducting end-to-end field tests to ensure the system runs smoothly. Front-End collects user feedback on the apps' UI. Based on strategy discussions, the list of datapoints to calculate or collect is updated. Students test the scouting system with artificial data before real match videos are available. Scouts and Subjective Scouts are trained on how to use the Match Collection app as competition season approaches.

2/27 to 4/23 — During competition season, the subteam runs a full-system test before each competition to catch and fix any bugs. At the first meeting after returning from each event, the full subteam participates in a debrief and communicates with Viewer and Picklist users and mentors to prioritize which changes to make before the next competition.

4/23 to 5/21 — Members work on the white paper and clean up the codebase to prepare for public release.

New Member Training

During the offseason, the initial months are dedicated to training new members. This period starts with an introduction to the basics of both Front-End and Back-End roles, allowing new members to familiarize themselves with the team structure and get to know one another.

Afterward, members decide which End (Front-End or Back-End) they'd like to join. During this time, all new members also download Git and receive Git training to ensure they are prepared for version control throughout their development process.

Once the new members have chosen their subteams, specific training tailored to each role begins.

Front-End Training

Front-End training starts with fundamental lessons covering Kotlin programming. These lessons include essential concepts such as variables, functions, for-loops, classes, and more. Each lesson incorporates hands-on assignments where members apply the skills they've learned. After mastering the basics of Kotlin syntax, new members proceed to their final training project: the Mini Scout.

Developed in 2021, the Mini Scout project involves building a scaled-down version of our Match Collection app. The project is designed to be progressively challenging, with initial instructions being specific and gradually becoming more open-ended. Throughout the Mini Scout, new members will develop core components like a match start screen, data input screen, and match data edit screen.

This project provides practical experience with the app development process, ensuring members understand how the application interfaces with the user and handles data collection.

Back-End Training

Back-End training begins with a foundational understanding of Python programming, delivered through slideshows and practical assignments. Once new members are comfortable with Python syntax, they move on to learn about the Schema used in our system and dive deeper into the MongoDB database. This ensures they understand how data is structured, stored, and queried in our system.

In addition to learning the technical foundations, new Back-End members are introduced to our Server, where they explore how data is transferred, processed, and organized across the

system. Members are also trained on GitHub, with a focus on creating pull requests, reviewing code standards, and understanding the importance of maintaining a clean and efficient codebase.

The Back-End training also includes team-wide lessons on general best practices and standards, equipping members with the skills necessary to contribute to the development and maintenance of the system.

Field Testing

Field tests are used to prepare for actual competitions. During a field test, we simulate aspects of a real event, such as live match scouting and sequential data updates. Our goal is to test every part of the system to ensure no bugs go unnoticed before competition. A full-system field test at 1678 usually proceeds as follows:

1. Before the test: Collect match videos—ideally high-scoring matches from a previous regional or district competition. If the competition season hasn't started yet, use Week 0 videos or screen recordings from the xRC Simulator.

2. Create a new Server branch to merge any hotfixes needed during the field test. Name the branch with the date of the field test. Pull any unmerged PRs that need to be tested into this branch.

3. Decide on a TBA event key to use (e.g., 2025cada for the 2025 Sacramento Regional). Ideally, this should match the event used for the videos, but it doesn't have to—it can be from a previous year as long as it has a match schedule and team list. Use the event key to create a test database, team list, and match schedule file.

4. Set up the system as it would be at a real competition, and send the newest .apk files to the tablets and phones.

5. Recruit volunteers to use the Match Collection app to collect data from the match videos. If fewer than 20 students are available, have people use multiple tablets at once. The main goal is not to get accurate data, but to thoroughly test every feature of the

collection apps. The match videos are just a guide, so it's okay if the team number assigned to a Scout doesn't match the robot they're watching.

6. Additionally, have users enter test data using the Pit Collection and Stand Strategist apps.
7. After each match, scan the data into the Server. Ensure data is being entered into both the local and cloud databases, and that the web server can send it to the Viewer. Monitor the Viewer to ensure data is updating and displaying correctly.
8. Have users test the OverRate and Playoff Scouting apps.
9. Write down every bug or suggestion for improvement as it comes up. After the test, review the list of bugs and prioritize which ones to address first.

Scout Training and Management

The week before each competition, Scouts are trained to collect accurate data using the scouting apps. Objective Scouts are trained by the Lead Scout. Training begins with an explanation of behavior standards at the competition and an overview of the itinerary. Scouts spend at least two hours practicing scouting while watching matches from the current season. Subjective Scouts are trained by our Strategy mentors and practice by watching matches while discussing their rankings with a mentor. After repeating this process and becoming more confident in their rankings, they collect data independently with no input from mentors and receive feedback after submitting their rankings. Video System, Pit Collection, and Stand Strategist users are trained by members of Software Scouting who have experience with the collection process. This training is informal, and knowledge is shared between users.

21 Objective Scouts and 3 Subjective Scouts are brought to each competition. Breaks are set by the Lead Scout ahead of time, and we try to give each Objective Scout at least three 30-minute breaks per day. During breaks, Scouts can go to the pits, meet other teams, and explore the venue. Each Scout has a Scout ID that is used to ensure three people are scouting each team. These Scout IDs are pre-assigned but change when breaks rotate. A Scout

count-off, where each Scout says their ID, is often done to ensure all 18 active Scouts are present.

Competition Roles

The roles at competition are divided into Developers, Scouts, and Strategists.

Developers — The Developers consist of one Front-End developer and one Back-End developer. They fix bugs as they arise and assist with Scout ID assignments and handing out tablets. The Back-End developer is also responsible for uploading scanned data to the Server after each match. Two Objective Scouts, who are primarily Scouts but can assist when needed, also serve as backup developers.

Lead Scout — The Lead Scout manages all logistics for the Scouts, including meals, shifts, handing out tablets, and anything else they may need. The Lead Scout does not scout, so they can focus on managing shifts and communicating with developers and mentors if any issues arise. There is also an Assistant Lead Scout who helps cover for the Lead Scout when needed and serves as an Objective Scout when not needed.

Scouts — Since 18 Objective Scouts are needed per match, about 21 Scouts travel to each competition to allow three Scouts to take breaks at a time. In addition, there are three Subjective Scouts so that one can be on break while the other two scout. Since Subjective Scouts are required to have extensive experience on the Strategy subteam, the student on break often chooses to help other Strategists.

Picklist Operator — A student, usually from Software Scouting with experience in programming, operates the Picklist spreadsheet during picklist meetings. The Picklist Operator is responsible for updating and maintaining the picklist spreadsheet throughout the season.

Video System Operators — Two trained students record, name, save, and send match videos to Strategists. Video System Operators also serve as Objective Scouts.

Stand Strategists — Two Stand Strategists write specific notes on each team while watching matches and collect needed subjective data. They edit the picklist at regionals based on each

team's performance. On the second day of the competition, they also participate in picklist meetings. Additionally, our Match Strategist uses the Pit Collection app to collect pit data on teams during the practice day and works with the drive team and Strategists throughout the competition. Our Strategists are often skilled veteran members of the Strategy subteam.

Subjective Scouts — Three Subjective Scouts collect qualitative data during matches, such as robot speed, driver awareness, and maneuverability. They use a separate Subjective Collection app and are trained separately from Objective Scouts.

Pit Scout — Our Match Strategist usually serves as a Pit Scout. On practice day, the Pit Scout collects robot data such as robot weight, drivetrain, and vision. They also take pictures of robots in the pits. All data is collected in our Pit Collection app.

Other Resources

2025 Public GitHub Repositories

Match Collection: https://github.com/frc1678/match-collection-2025-public

Viewer: https://github.com/frc1678/viewer-2025-public

Pit Collection: https://github.com/frc1678/pit-collection-2025-public

Stand Strategist: https://github.com/frc1678/stand-strategist-2025-public

OverRate: https://github.com/frc1678/overrate-2025-public

Playoffs Scouting: https://github.com/frc1678/elims-scouting-2025-public

Server + Schema: https://github.com/frc1678/server-2025-public

Picklist Editor: https://github.com/frc1678/picklist-editor-2025-public

Kestrel: https://github.com/frc1678/kestrel-2025-public

Old Whitepapers

Our old whitepapers can be found on our team website (https://www.citruscircuits.org/scouting.html).

Fall Workshops

Every year, we hold workshops to help students from other FRC teams learn the skills necessary to grow competitively as a team. Our previous Fall Workshops are linked here (https://www.citruscircuits.org/fallworkshops.html).

Conclusion

Lessons Learned

Overall, 2025 was a successful year for the Scouting sub-team. However, we identified two key areas for improvement: field testing and communication between the Front-End and Back-End teams.

Our first event, the Pinnacles Regional, was rough. A lack of thorough testing led to significant bugs in both the Viewer and Server. In response, we revised our testing protocols to ensure all system features are thoroughly tested in advance. We introduced a “competition simulation” weekend, where the entire system was run as if it were a live event. This approach reinforced a critical lesson: test *every* component thoroughly and simulate real competition conditions.

Another challenge was the lack of communication between the Front-End and Back-End teams. At Pinnacles, mismatched data schemas and formatting inconsistencies—particularly with auto path positions—caused further issues. Moving forward, it’s essential that both teams maintain consistent communication regarding data formats, along with routine collaborative testing.

While no major issues persisted by the end of the season, these lessons have positioned us well for a stronger and more refined performance in 2026.

Starting Your Own Scouting System

We recommend teams start with a small system structure: you can use a web app or paper and pencils—whichever is easiest for you. Citrus Circuits has successfully used Google Forms to scout off-season events in order to train new members in scouting principles and methods. Then, prioritize training your Scouts. We highly recommend finding at least one hour a week where your Scouts can watch match videos and discuss them with one another. Great discussion questions include “What would you have done differently?”, “Which team did well?”, “What were some minor mistakes?”, and “Who had the best agility?”

If you can spare only about two members of your team for scouting, we recommend either having each one take notes on the performance of one alliance (training is critical when recording qualitative notes) or reaching out to fellow teams to gauge interest in forming a scouting alliance. Each team can get a copy of the data to review for their matches and picklist meeting, and none of them have to give up as many members. However, if you are part of a scouting alliance, try to create a uniform training method (e.g., schedule a two-hour Zoom training where they practice taking notes or using your scouting app).

If you have any questions about starting your own scouting system, want our team's advice based on your resource level, or have any other questions, please contact us at 1678scouting@gmail.com.

Future Steps

We plan to use the lessons learned from this season to improve for the next. This season was by far our most successful in terms of the app, data transfer, and code. However, it is essential to recognize our shortcomings and address them adequately during the off-season months. We plan to improve our new member and scout training, as well as continue projects we started this year on data validation, AI match predictions, and more.

There are many new features to implement, code to clean, and processes to streamline. There is no "perfect system," and we will continue to work hard to improve it.

Appendices

Codebook

Our codebook contains every datapoint we collect, grouped into the respective collections they belong to (e.g. Objective Team or Pickability). The codebook spreadsheet contains each datapoint name, its Python data type, and a short description of the datapoint. In total, we collected 529 datapoints this year.

Link to codebook:
https://docs.google.com/spreadsheets/d/1vZbIxE61mztuSgBBYGbb_m6Ej-Me9YM2I691eljpezo/edit?usp=sharing

Hardware

This Scouting System requires multiple pieces of hardware to ensure it runs smoothly and efficiently. For our apps, we use over 30 tablets for Match Collection and four Android phones for the Pit Collection and Viewer apps. For data transfer, we use two QR scanners that transfer data from the tablets directly to a single laptop running the Server. All hardware is packed into five cases: two tablet cases, a Server case, a Video System case, and a gray case for cables, spares, and power strips. More specific details about the cases and our hardware can be found in Section 4.2 of our 2020 Whitepaper.