2024 Scouting Whitepaper

Table of Contents

Introduction

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 that has since grown into a full subteam of 21 students with two different ends and six apps.

The system was structured in 2013 with two relatively unique attributes: (1) collection using Android tablets for both objective scorings—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 to be used 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 for first and second picks for 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 left the top teams unable to ally 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 wifi 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. Those advances included further development of the Picklist Editor, added visualization in both the Viewer and Picklist Editor, and a 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 to assign three Scouts to each robot for collecting objective data to improve accuracy. In 2018, Bluetooth became less reliable with 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 then 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 the system could be updated further. In 2021, the subteam continued to meet remotely and worked on improving the structure of the 2020 system without the need to focus on a specific game. The 2022 system resulted from substantial improvements since the Kotlin rewrite in 2020, and the revisions set the foundation for future features. In 2023, one major change was switching from our web

server communication interface from Cardinal (described in earlier white papers) to Grosbeak. The switch was made because, due to a lack of documentation, it was difficult to maintain Cardinal, and we had already been using Grosbeak for the picklist in Viewer.

The current 2024 system builds on the successful upgrades in 2023, which added new features to Viewer and created the new Stand Strategist and OverRate apps.

Summary of Major Changes Since 2023

Throughout the 2024 season, we made many changes to ensure our apps were efficient and had all the needed features. This season, we switched our new app for our Stand Strategists to record notes about an alliance and collect subjective data from Windows to Android. They had been using Google Sheets before 2023, which was unreliable due to the lack of stable WiFi connections at events. Additionally, we made significant progress in transitioning to Jetpack Compose, Google's modern toolkit for building UI for Android, instead of XML. We used Compose to create several screens in our apps, including programming Stand Strategist solely in Kotlin and Compose. We also refined our Viewer app by adding complex features, including auto paths and comparison bars, to our Team Details screen. Finally, we created an entirely new app, Overrate, to help Strategists rank and rate teams in a competition live.

Match Collection was updated for structured playoff scouting. During playoffs, we had one of our top scouts (based on scout precision rankings) and one of our strategists watching each robot on the field (6 scouts and 6 strategists total). The scouts would use the Playoff Scouting part of the Match Collection app to record the amount and location of Notes scored by the robot. The strategists would take notes on their assigned robot, and at the end of the match, the scouts would show the strategists their collected data. The data would then be posted in a Slack thread organized by playoff alliance, and the match strategist and drive team would use the data to create match strategies. In the past, we had no structured form of scouting during playoffs, but to enhance our competitiveness (especially at the World Championships), we thought it was necessary to collect this extra data.

System Overview

The Citrus Circuits scouting system is separated into three main stages: collection, processing, and visualization. Collection consists of the Match Collection app, the Pit Collection app, and the Stand Strategist app. In the processing stage, the Server organizes and runs calculations on the data, which the Viewer app and the Picklist Editor can then visualize by pulling the data through Grosbeak.

Scouts use three different collection apps at competitions: the Match Collection app, the Pit Collection app, and the Stand Strategist app.

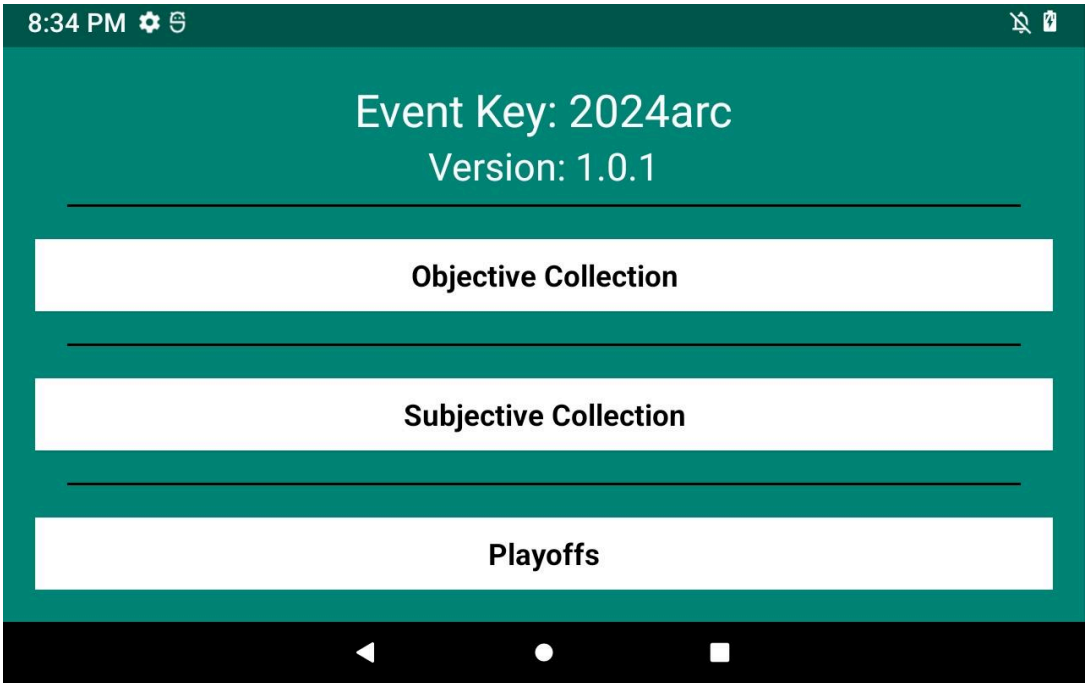
The users of the three collection apps are as follows:

- 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)
- Pit Collection (Subjective): Team members focused on 1678's alliance partners (Citrus Seals and mentors)

Match Collection

Scouts in the stands use the Match Collection app during matches to collect quantitative and subjective performance data on robots. The app runs on Lenovo Android tablets and was developed with Android Studio using 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 Super Scouts (one per alliance).

- Objective Collection collects 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 Quickness between the robots in the Red alliance).
- Playoffs Scouting allows Scouts to record Amp, Lob Ferry, Unamplified Speaker, and Amplified Speaker scores during playoff matches. We reserve Playoffs Scouting for Championships.



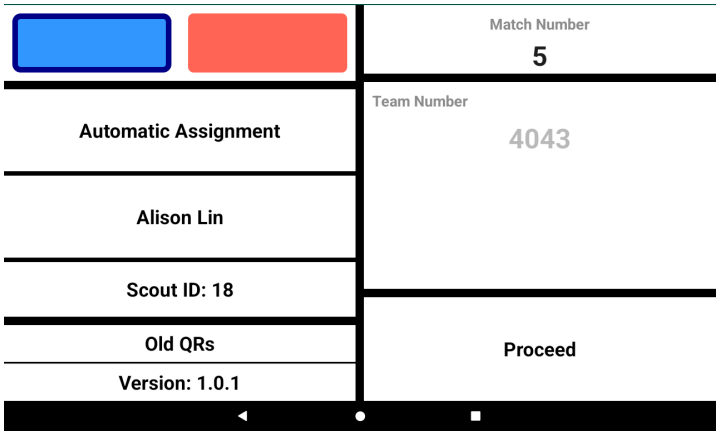
Objective/Subjective/Playoff screen

Starting Screen

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

Objective Collection

Objective Collection Input



Objective Collection Input screen

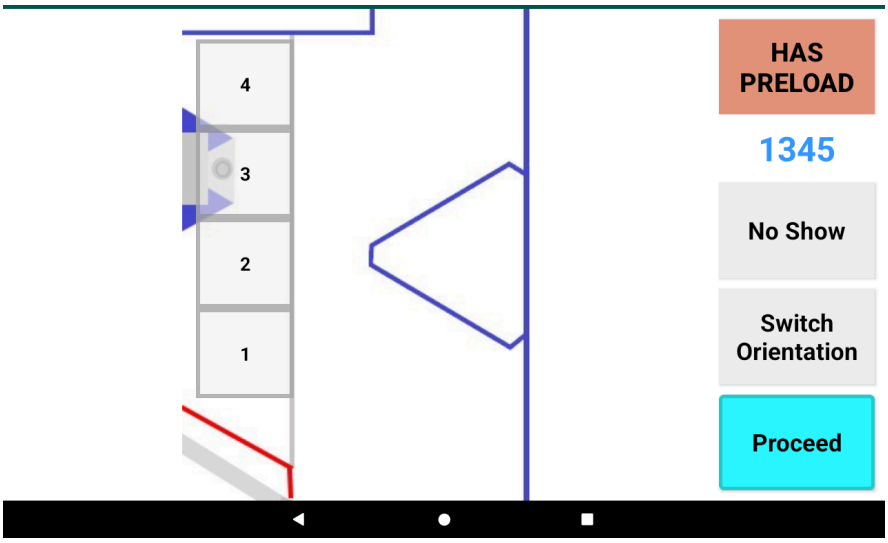
The Objective Collection Input screen has multiple elements. Firstly, the Assignment button can set the assignment to Automatic Assignment or Override Assignment. Automatic Assignment pre-sets the team and the alliance color they are scouting. On the other hand, Override Assignment allows the scout to change the alliance color and team number they are scouting, along with overriding some features later in the app. Scouts can edit the match number, scout name, or scout ID in the input screen. Furthermore, an “Old QRs” button allows scouts to view all past QR codes in case the scout forgets to scan the QR code after a match. Finally, scouts use the Proceed button to transition onto the Objective Collection Starting Position Screen.

Randomizing Scout IDs

Match Collection randomizes Scout ID assignments to minimize concentrating errors Scouts make on a few robots. A group of three Scouts will watch a single robot, and then all three groups will shuffle for the next match. This feature ensures a scout will scout different robots

throughout the competition, which helps prevent certain teams' data from becoming inaccurate due to less competent scouts consistently scouting them. It also helps us compare the scouts' performance (see Scout Precision Ranking (SPR) in Server) and better check our data against The Blue Alliance. These random assignments come from a file resource with 100+ randomized orders of Scout IDs generated ahead of the competition.

Starting Position Screen



Objective Starting Position screen

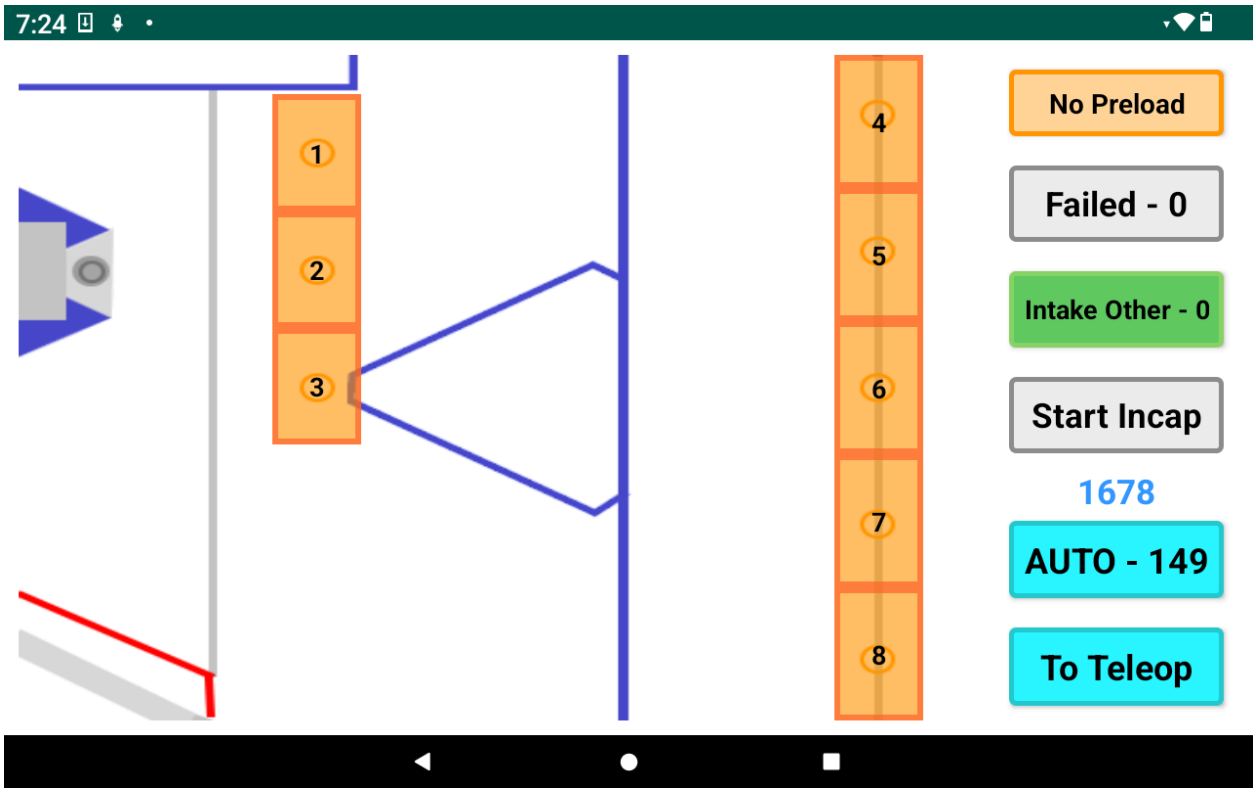
On the Starting Position screen, Scouts must input their robot's starting position by pressing one of the four starting position buttons. The Scout presses the "No Show" button if the robot is not on the field. There is also a preload button on the screen, which Scouts use to record whether or not the robot has a preloaded Note. Scouts use the "Switch Orientation" button when they do not get the optimal location in the stands and have to switch the app's orientation to match their view of the field. We use the data collected from the Starting Position and Auto screens to formulate auto paths.

Objective Collection Screen

This year, we implemented map scouting for our Objective Match Collection app, an app structure that creates a map of the field and positions collection buttons at corresponding locations on the map. This change allowed us to better collect when robots did specific actions in certain areas of the field.

Auto

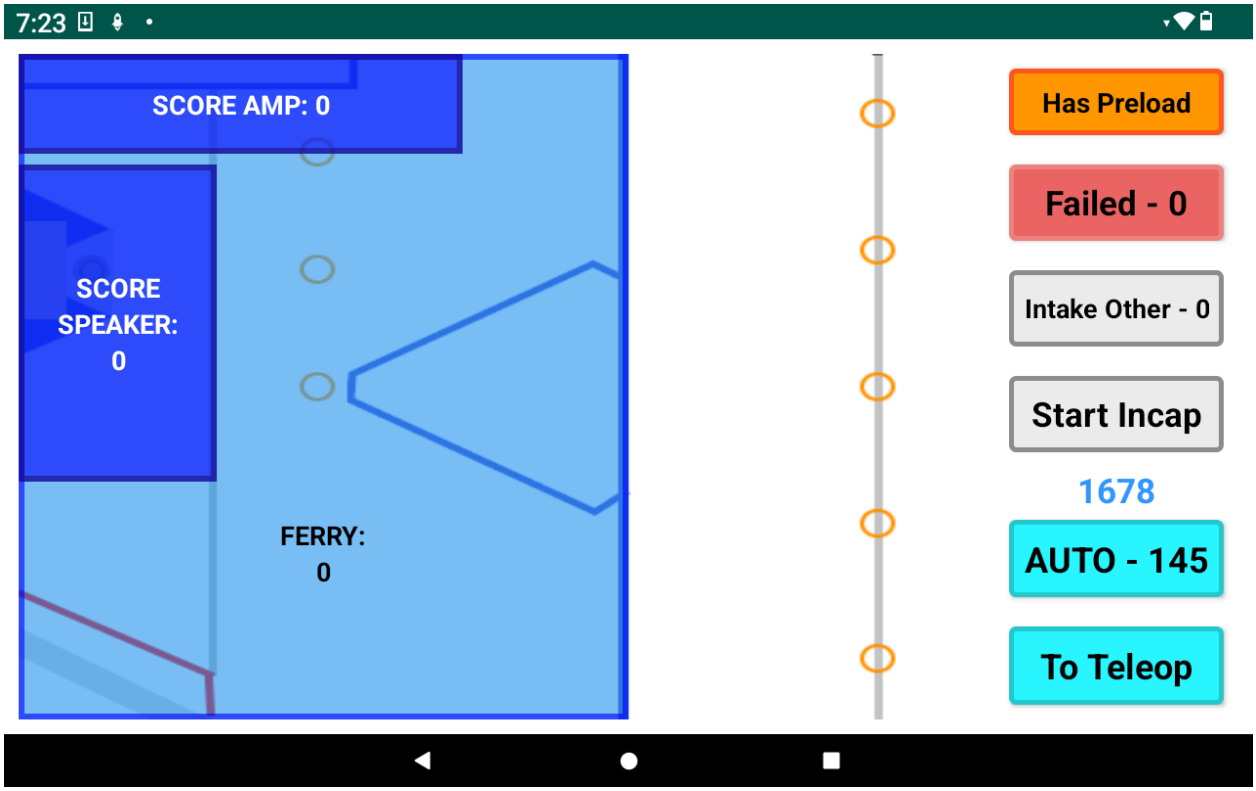
The Auto screen depicts a diagram of the alliance's side of the field, with a different set of buttons depending on whether the team has a Note. While on the Auto screen, users can also long press the timer to reset it, this feature is disabled once any action is inputted.



Auto Intake

The Auto intake screen has nine buttons for recording intakes: one for each pre-placed Note and an additional button called Intake Other for intaking Notes that were not in their original

positions (because of dropping/ferrying or failed scores). Collecting the location of intaken Notes helps us track the auto paths teams used.

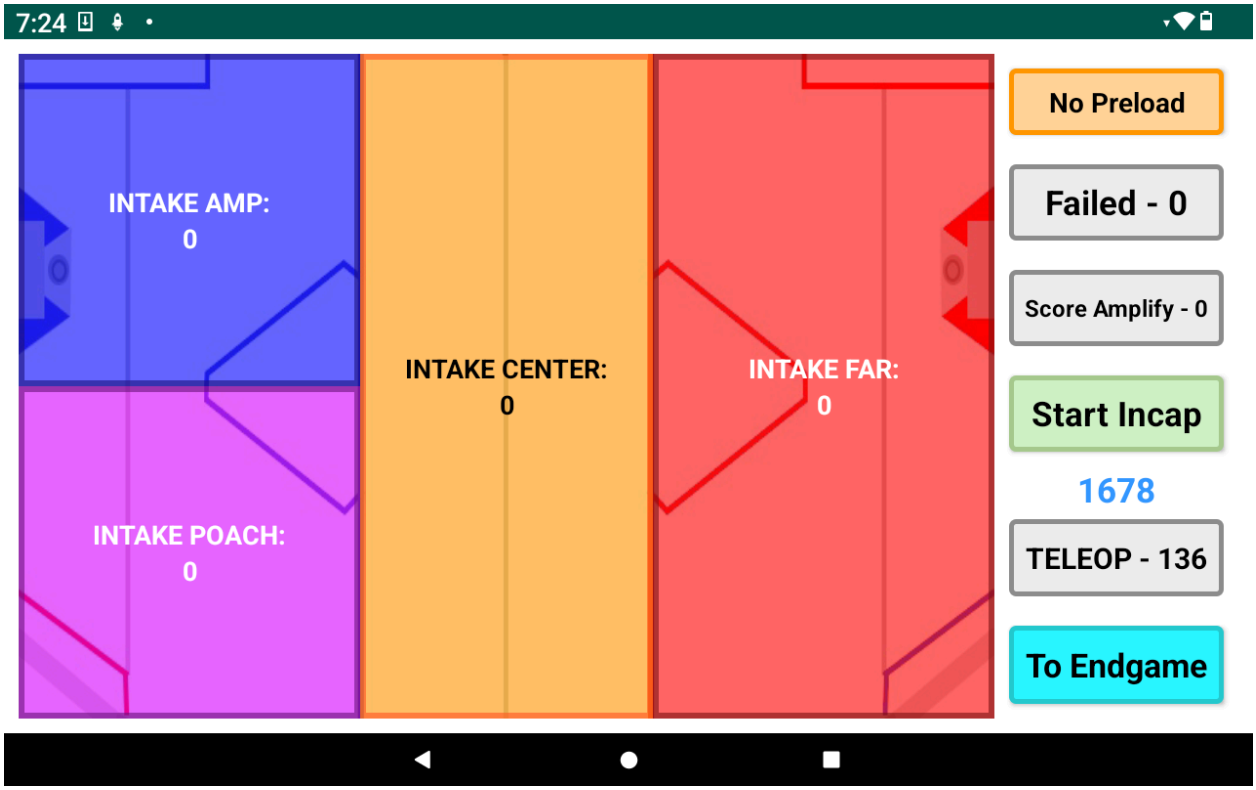


Auto Scoring

The Auto scoring screen consists of four scoring buttons, one for each scoring action a team can take with a Note and a Fail button.

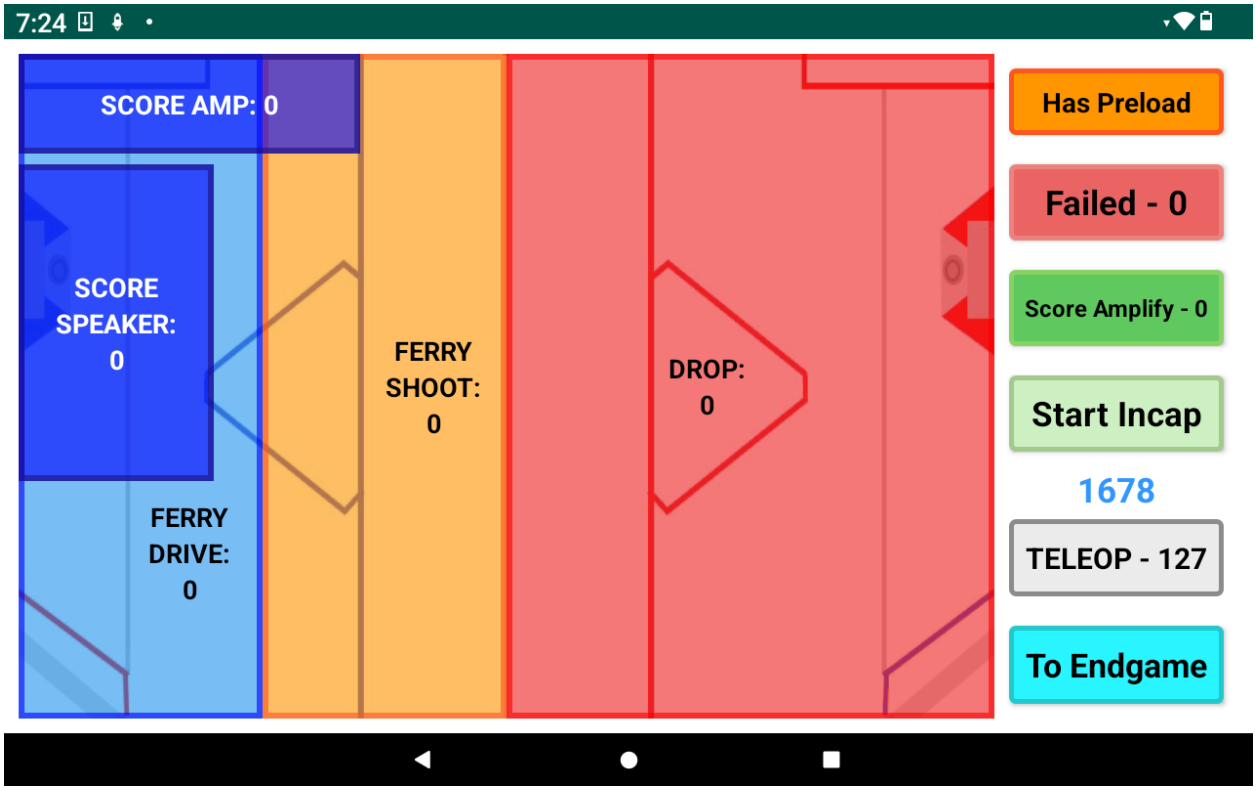
Teleop

After switching to Teleop, the field diagram changes from showing only half to showing the entire field. Like the Auto screens, it has a different set of buttons depending on whether or not the team has a Note.



Teleop Intake

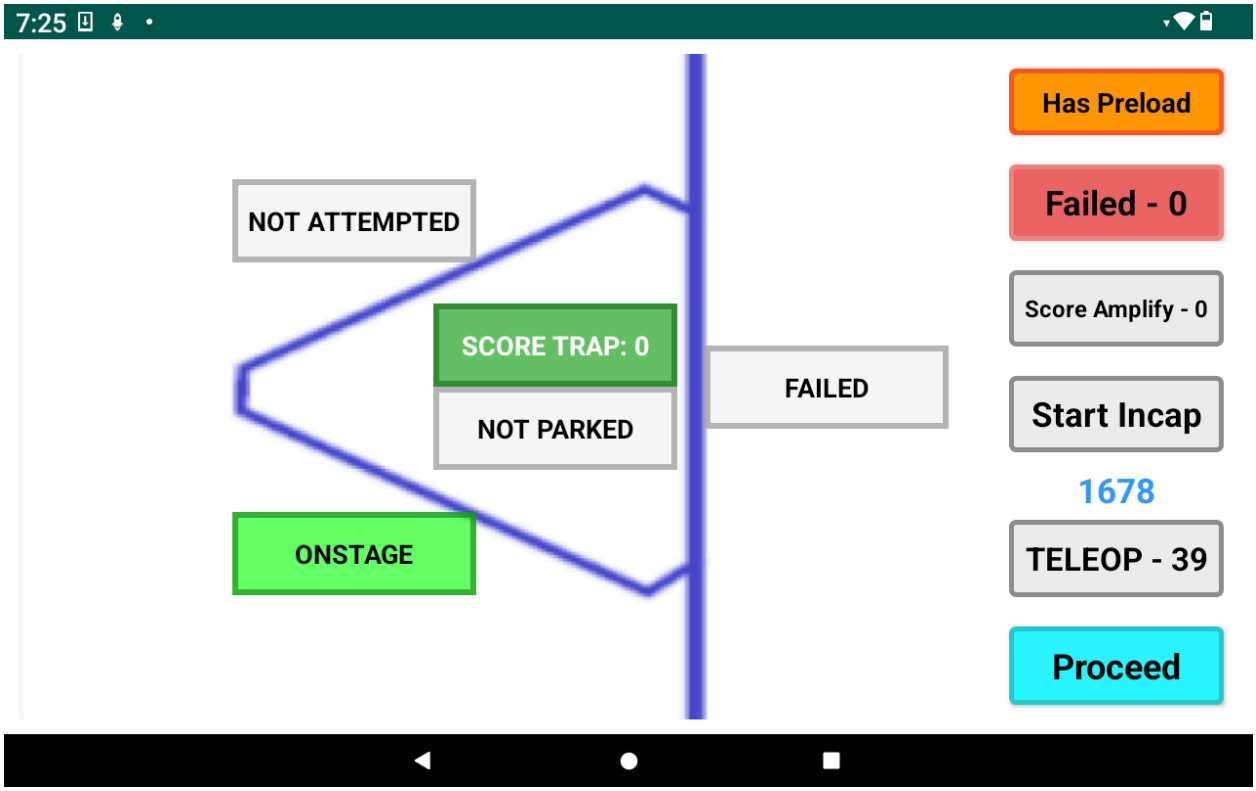
The Teleop intake screen has four intake buttons for collecting intake instances from different locations. These areas include the opposing Wing, the center of the field, near the Amp, and near the opposing Source.



Teleop Scoring

The Teleop scoring screen has seven buttons to collect what teams do with Notes they get during Teleop. Originally, the screen only had six buttons. However, right before the Championships, we split Ferry into Ferry Shoot (shooting a Note into the Wing) and Ferry Drive (driving up and dropping the Note) due to changes in our cycle time calculations.

Endgame



Endgame Screen

This year, we added an entire screen for collecting Endgame data, unlike last year when we had a popup. It consists of a Stage diagram with four toggle buttons, one for each of the chains on the Stage, and two other buttons (for scoring in the Trap and the Fail button). This change allowed us to track which chain a team climbed on, whether or not they failed to climb on any chain, or if they parked. Initially, the Trap button was a toggle, too, but when we found that some teams had scored multiple traps in a single match, we changed it to a counter, just like the other scoring buttons. The app disables the Trap button if the robot does not have a Note.

Switching Intake and Scoring Buttons

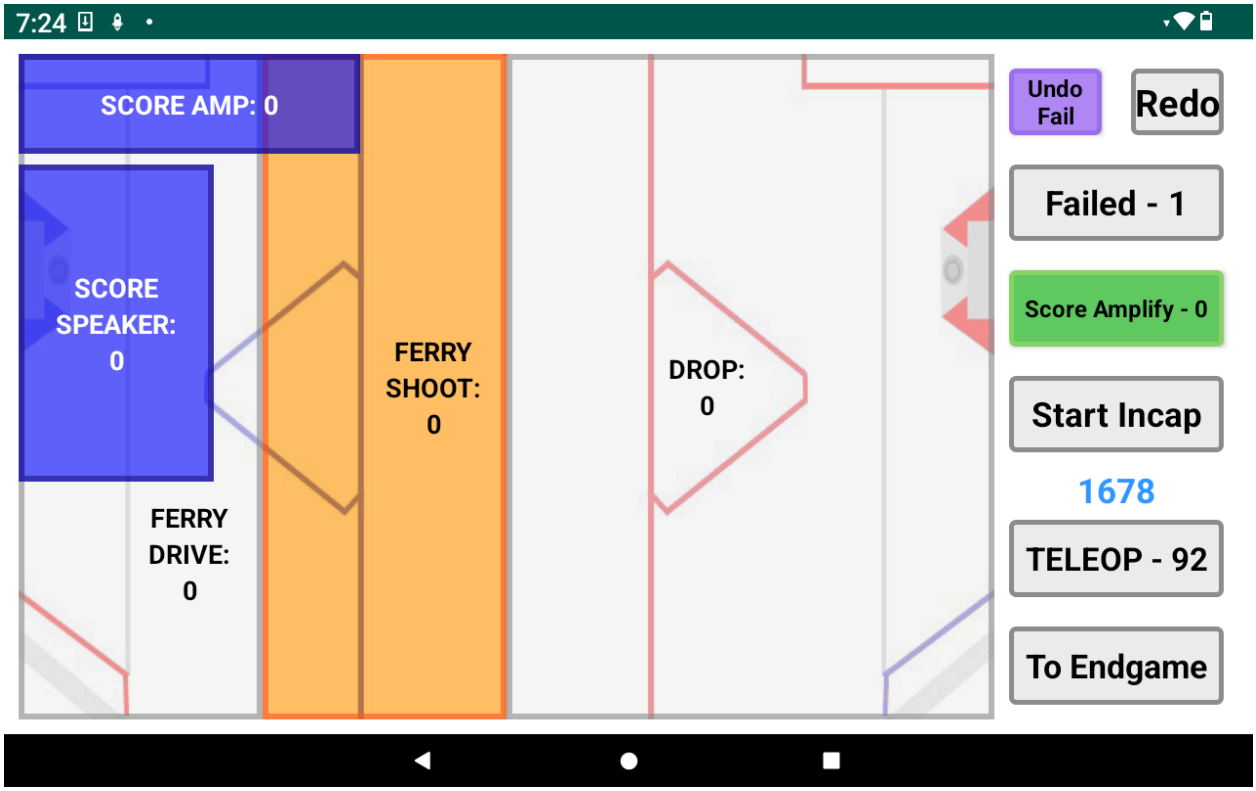
Because robots can only hold one Note at a time, the intake and scoring buttons are not displayed simultaneously. Whenever a scout taps an intake button, the app switches all the

intake buttons to the scoring buttons and vice versa. Certain buttons like Undo, Redo, and Incap will always remain.

Preload Button

Scouts can change whether the robot has a preloaded Note as long as no actions have been taken other than switching between Auto, Teleop, and Endgame. This feature is helpful in case it is hard to tell whether the robot has a preload so that the scout can quickly change it. Note that this button replaces the Undo and Redo buttons until the user inputs other actions.

Fail Button

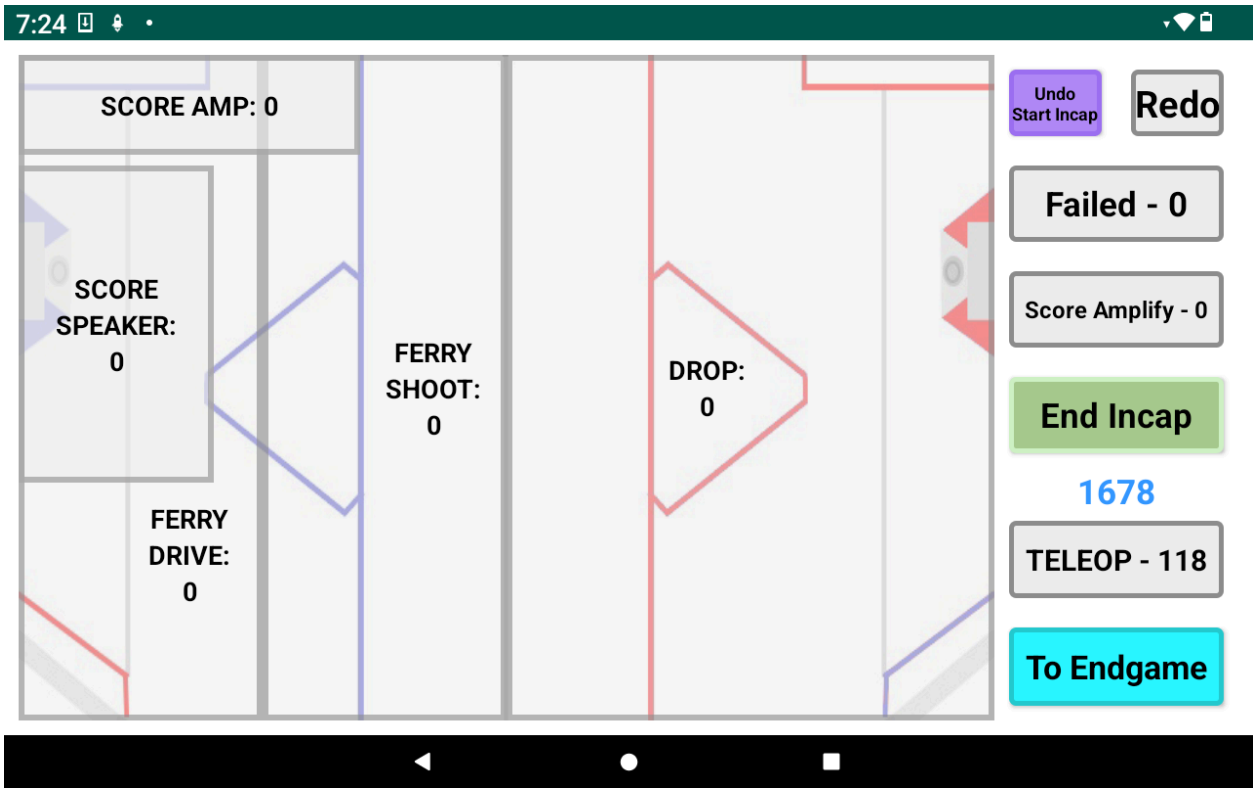


Failing

This year, we changed our Failed button to collect which action they failed (e.g. Score Speaker, Amp, Trap, or Ferry), unlike last year when we could only collect that they failed to score a gamepiece. This change was significant in calculating success rates for each action. Note that

after pressing the Failed button, certain buttons, such as the Drop and Ferry Drive buttons, are disabled until the user undoes the fail action or presses another button.

Incap Duration Using Timestamps



Incap

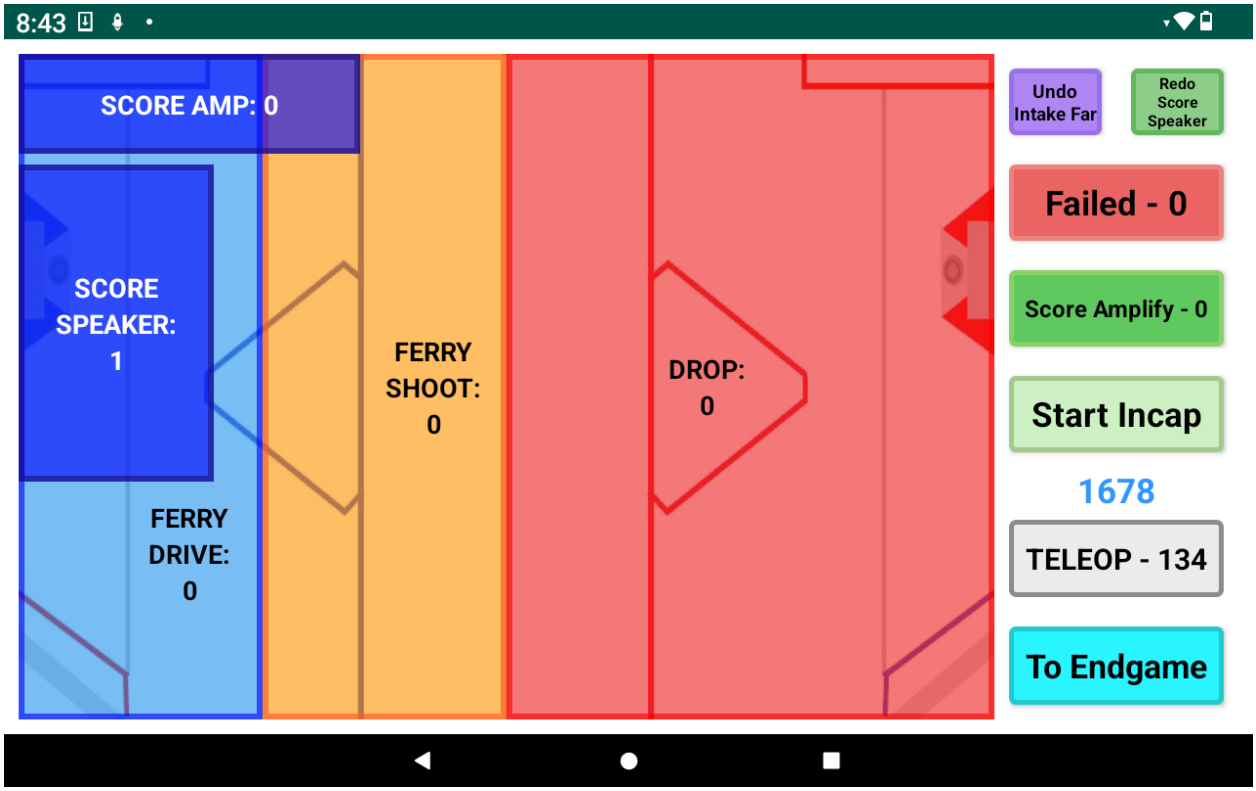
In Teleop and Endgame only, Scouts mark a robot as incapacitated when it is disabled or its drivetrain movement is significantly impaired. A toggle button records the start/stop timestamps of the incapacitation period. This allows us to find the total incap time of a robot during a match—note that if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

Navigation Between Auto, Teleop, and Endgame

Like previous years, we had a button to progress the match from Auto to Teleop, but with the Endgame screen this year, we added a To Endgame button to switch from Teleop to Endgame.

With the changes to the Failed button, we also had to disable the Proceed buttons while the user was failing an action. Additionally, we added the capability to return to the previous screen if the user didn't input actions after switching screens. This process is initiated by long pressing the To Endgame (to return from Teleop back to Auto) or Proceed (to return from Endgame back to Teleop) buttons. Note that because teams could potentially score multiple Traps in a match, we added the capability to do this on the Endgame screen unless they are Incap, failing a score, or Onstage. If the user is on override mode, they can also proceed using the Proceed button past the Endgame screen, a feature that allows for faster field testing.

Undo and Redo



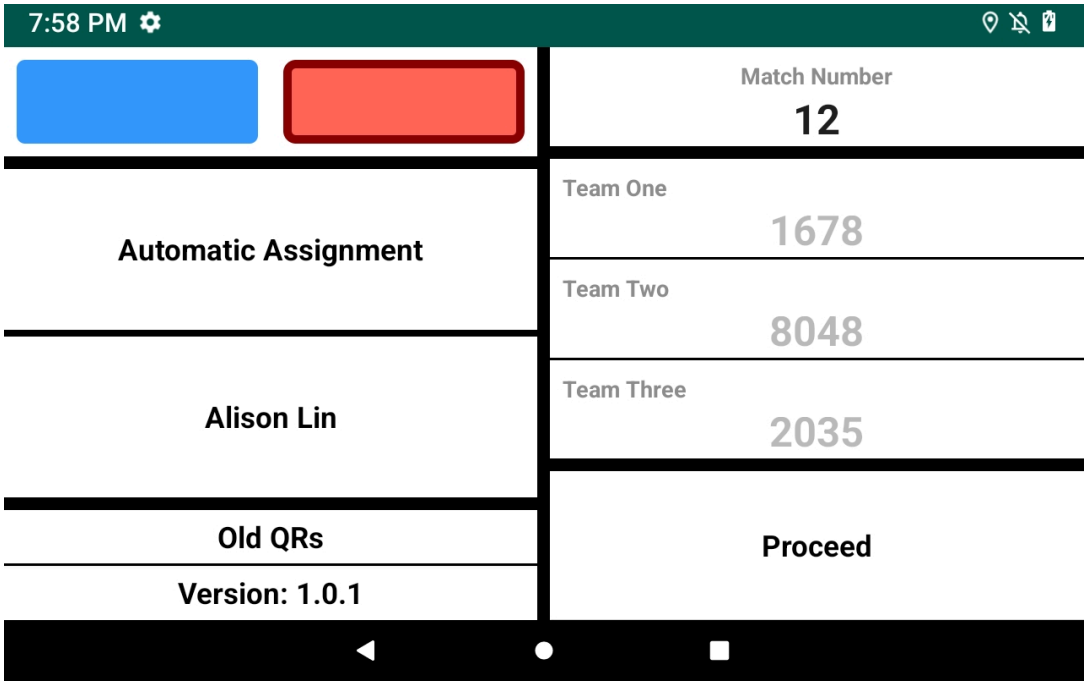
Undo and Redo Buttons

Like in previous years, we had buttons for undoing and redoing recorded actions. When the user presses the Undo or Redo button, it undoes or redoes the most recent action recorded or undone, respectively. Due to the Fail button changes, undoing or redoing a failed action undoes or redoes both the fail and the actual action.

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 the robots against each other in quickness and field awareness. They also record how many seconds are left in the match when the robot goes to climb and if the team harmonizes.

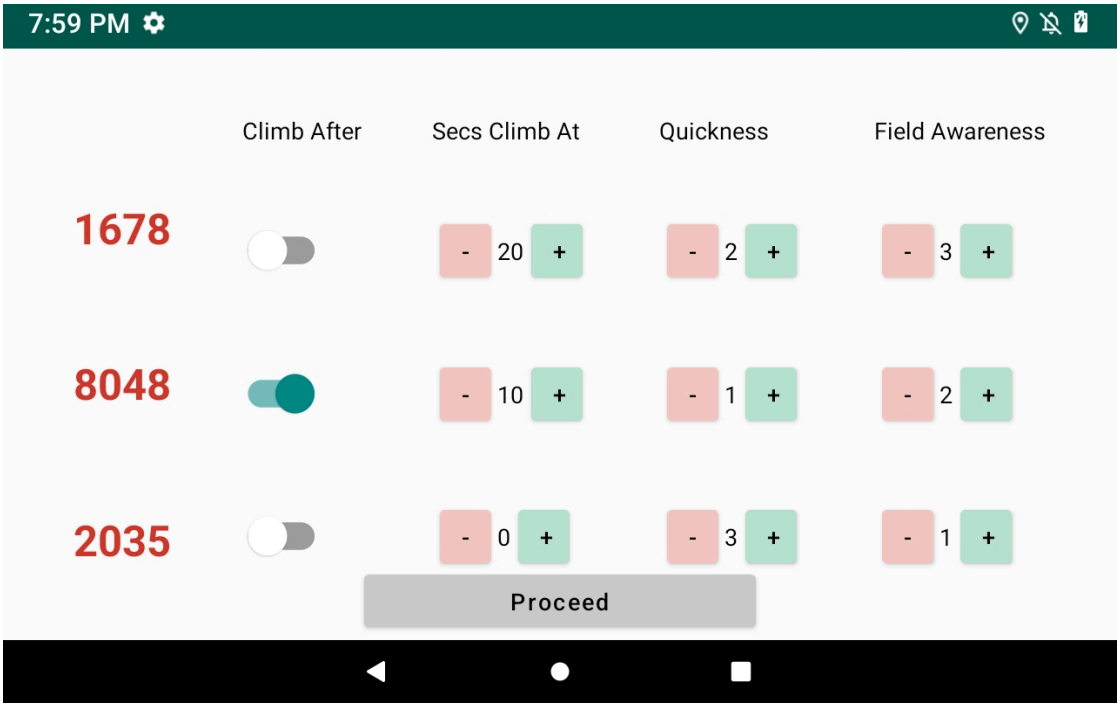
Subjective Collection Input



Subjective Collection Input Screen

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

Subjective Collection Screen



Subjective Collection Screen

Besides changing the datapoints we are collecting this year, the main difference with the Subjective Collection Screen is that we removed the match timer. We made this change as the Subjective Scouts no longer collected whether a team plays defense, subsequently removing the need to collect timestamps, as we did last year for a defense checkbox.

Climb After

Marking Climb After indicates whether a team climbs on a chain that already has another robot on it. Strategists use this datapoint to determine whether and how consistently a team can harmonize.

Seconds Climbed At

Subjective Scouts mark how many seconds are left in the match when a robot goes to climb. Seconds Climb At can be changed in increments of 10 up to 60 seconds. This datapoint

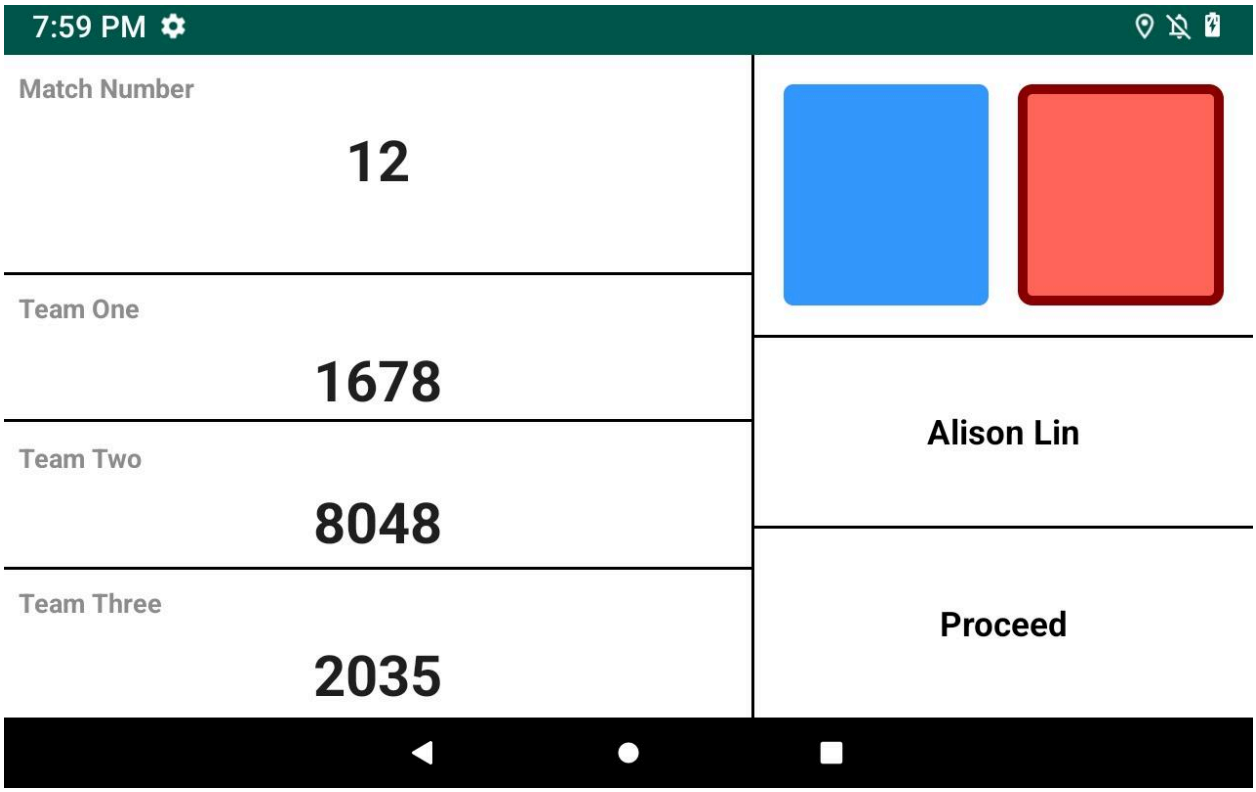
provides insight into how long a robot needs to climb successfully so that Strategists can plan the Endgame timings of the match strategy.

Quickness and Field Awareness

Subjective Scouts rank each robot on an alliance relative to the other robots by Quickness and Field Awareness. Ranks are from 1 to 3, with 3 indicating the best robot in that category in the alliance. Quickness is based on the speed and maneuverability of the robot, while Field Awareness is based on how aware the robot is of where it is on the field, where other robots are on the field, where other Notes are on the field, how well the robot scores, and how smooth its cycles are.

Objective & Subjective

Match Information Edit Screen



Match Information Edit Screen (Subjective)

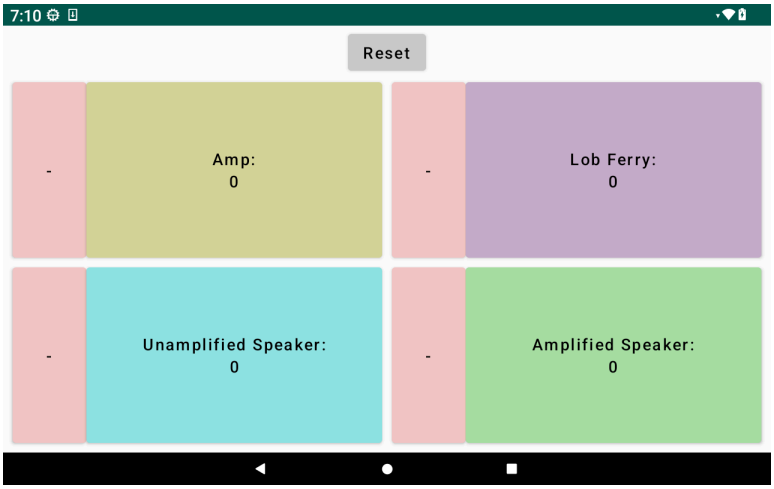
The Scout enters the Match Information Edit Screen after pressing the Proceed Button on either the Input Screen when in Subjective Collection or the Endgame screen in Objective Collection. After a match, the Scout can view and change match information, including the match number, team number(s), alliance color, and scout name. This screen is used when the Scout accidentally scouted the wrong robot or inputs inaccurate information.

The QR Schema Format

QR code generator screen (Subjective)

QR codes follow a specific format defined in Schema to reduce their size. Letters of the alphabet represent Datapoint names, and special symbols such as '$.' delimit each section and datapoint. This way, the app quickly generates QR codes that are small enough to contain all match data. A smaller QR code makes scanning easier, and large QRs may fail to render.

Playoffs Scouting



Playoffs Scouting Screen

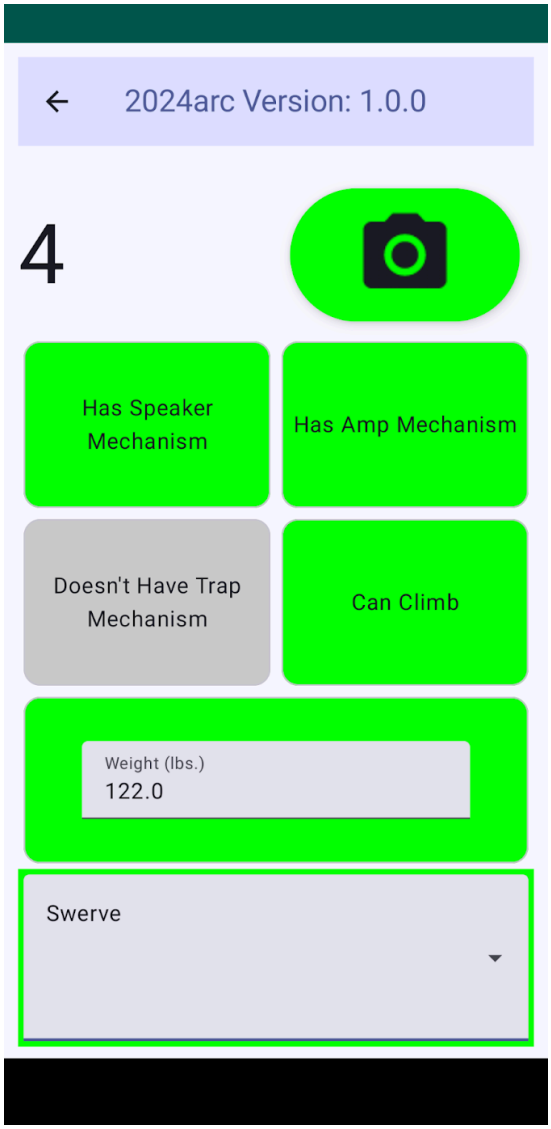
We only used Playoffs Scouting at the Championship this season. It consists of a screen with counter buttons for four specific actions: Amp scoring, Lob Ferrying (if a robot ferries by shooting over the stage), Unamplified Speaker scoring, and Amplified Speaker scoring. When a Scout taps a counter button, it increments that action's counter. If a Scout taps the corresponding subtraction button, it decrements that action's counter. At the end of the match, Scouts record the number of each scoring type and share the data in Slack (Citrus Circuits' communication platform).

Pit Collection

The Match Strategist uses the Pit Collection app with the assistance of other strategists to collect mechanical data and robot pictures. It runs on Android phones, and we built it with Kotlin and XML. This year, we completely revamped the app using the Jetpack Compose framework.

All the main features from previous years are still present in this year's Pit Collection app, but we discontinued a few, such as the pit map and flagging, due to their lack of use in previous years.

Datapoints Collected

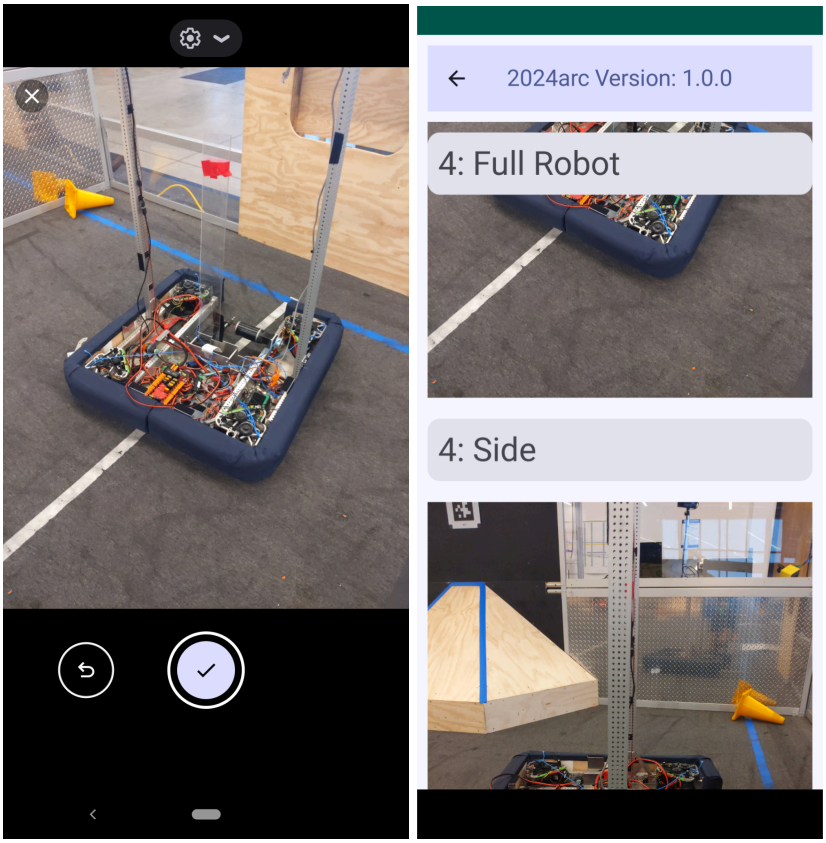


Data Collection Screen

The Match Strategist uses the Pit Collection app to collect several datapoints about a robot's physical characteristics. These include whether or not the robot has a Speaker, Amp, Trap,

and/or climb mechanism, its weight, and what kind of drivetrain it has. In the screenshot above, green signifies the completion of an input.

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, and two photos of the entire robot from different angles. However, in this year's app, there are only two categories of robot pictures: the full robot and side profile, as those are enough to get a good idea of what a robot looks like. These pictures are taken in the app using the phone camera and then stored in a folder called `robot_pictures` in the local Downloads folder. The Developers pull the `robot_pictures` folder from the Server computer via a USB connection. The pictures are named with the team number first, followed by the picture angle, for example, `1678_full_robot.jpg` or

`1678_side.jpg`. Users can preview pictures by holding down the camera icon on the team list screen.

Naming Photos and JSON Files

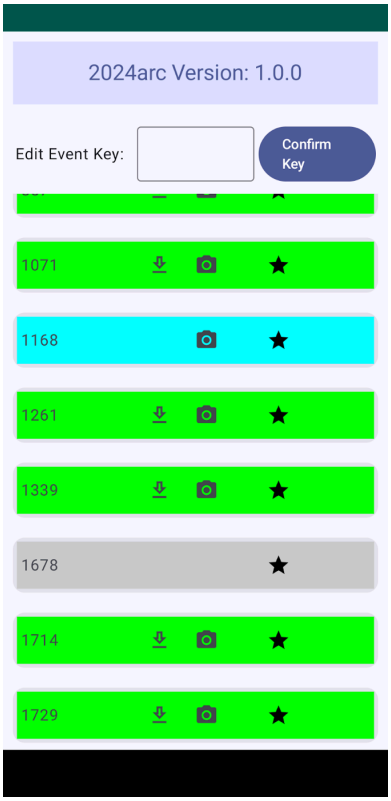
The app stores the data for every event key in a separate folder in the Downloads folder. Each event key's specific folder is named the event key, e.g., `2025dal`. The Pit Collection app then stores the collected data in its folder as a JSON file called `pit_data.json` for each event key.

Starred Teams to Organize Multiple Scouts

Team List Screen with Starred Teams

To organize scouting teams with multiple Pit Scouts, a user can tap on the star icon in the list of teams to turn the star yellow. This way, Pit Scouts can divide the teams among themselves. Pit Scouts star teams individually to allow for nonnumerical pit setup orders.

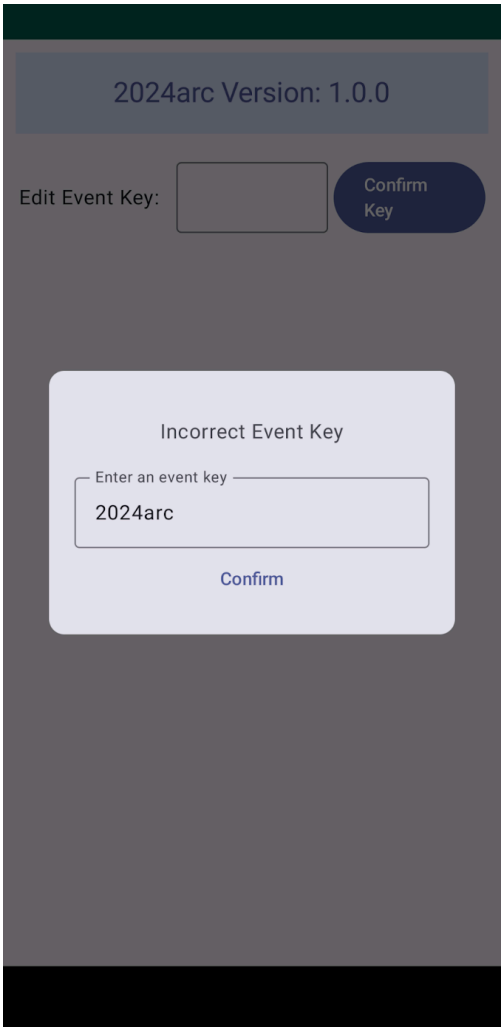
Highlighting to Show Scouting Progress



Team List Screen with Highlighted Teams

When looking at the list of teams, the color of each team's cell depends on what kind of data the Pit Scout has collected on that team. If there is no data on a team, the cell appears gray; if there is partial data (only pictures or only data), the cell appears cyan; and if there are both pictures and data, the cell appears green. There are also icons signifying the collection status: the download icon indicates that the team has data; the camera icon means the team has photos. The highlighting allows the Pit Scout to see which data still needs to be collected.

Editing Event Key



Event Key Screen

Users can change which event the Pit Collection app pulls a team list from by changing the event key. The Blue Alliance provides event keys, and if a user inputs an invalid event key, an error screen prompts the user to reenter a valid event key. Defaulting to the most recent event, if the user inputs another invalid event key, the screen reopens until they input a valid key.

Stand Strategist

Overview

Stand Strategist is an app for Stand Strategists to take notes about teams in an alliance in each match. At the competition, our Stand Strategists record detailed notes and observations about each team for use during picklist meetings. We previously used Google Sheets to organize these notes but found that its offline support (which is required at events where an internet connection is infeasible) did not suit our needs. We originally developed Stand Strategist as an offline-first, local desktop app running on Windows, macOS, and Linux, allowing our Stand Strategists to take notes on their own laptops even with the connectivity constraints of a competition environment. Stand Strategist is now written in Kotlin and built using the Compose Multiplatform UI framework by JetBrains.

Changes from Last Year

The Stand Strategist app was changed from a Windows-based desktop app to an Android app this season since we previously ran into problems with the battery life of the Stand Strategists' laptops. This resulted in many structural changes, but we kept the user interface as close as possible to what it was before. Throughout the season, we also added several new features to improve user experience. We updated the datapoints to better reflect aspects of Crescendo and replaced and removed some datapoints to make it easier for the Stand Strategists to consolidate and collect notes. We also added new ways to organize profiles within the app and export profile data. Finally, we allowed users to search for teams by team number and a given defense rating.

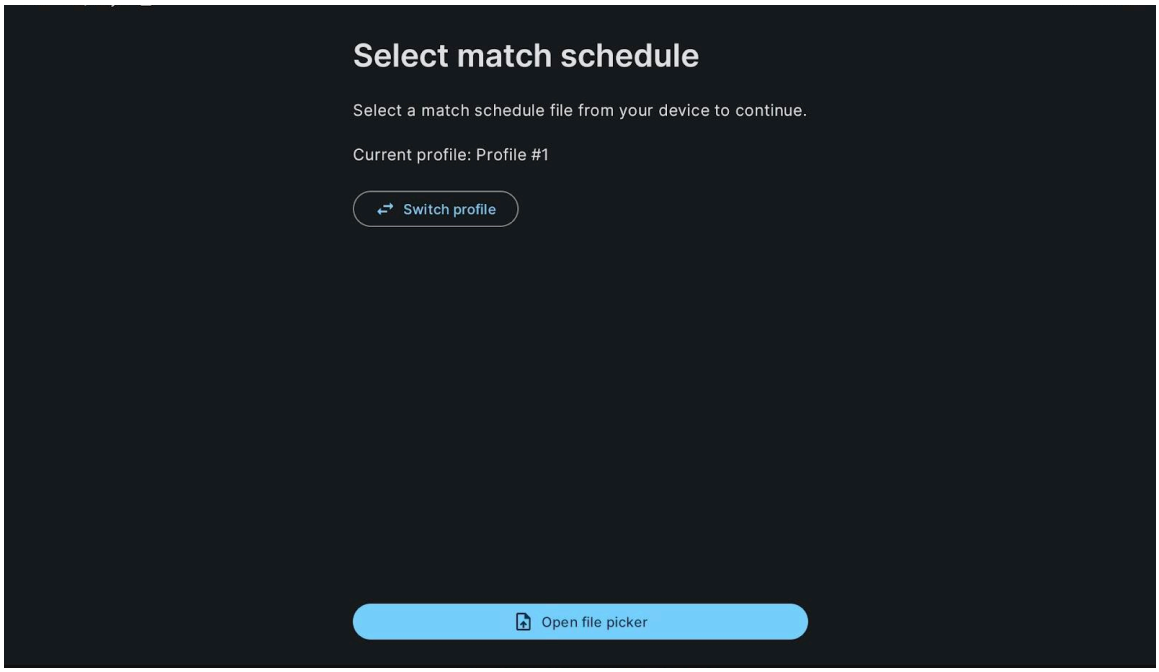
Navigation

Stand Strategist users can navigate between the data collection pages by swiping left and right on the tablet screen or through specific keyboard shortcuts. When the user navigates through

all the pages for a match, the match number is incremented or decremented accordingly. There are also several buttons the user can press to open specific pages or popups.

Entering Data

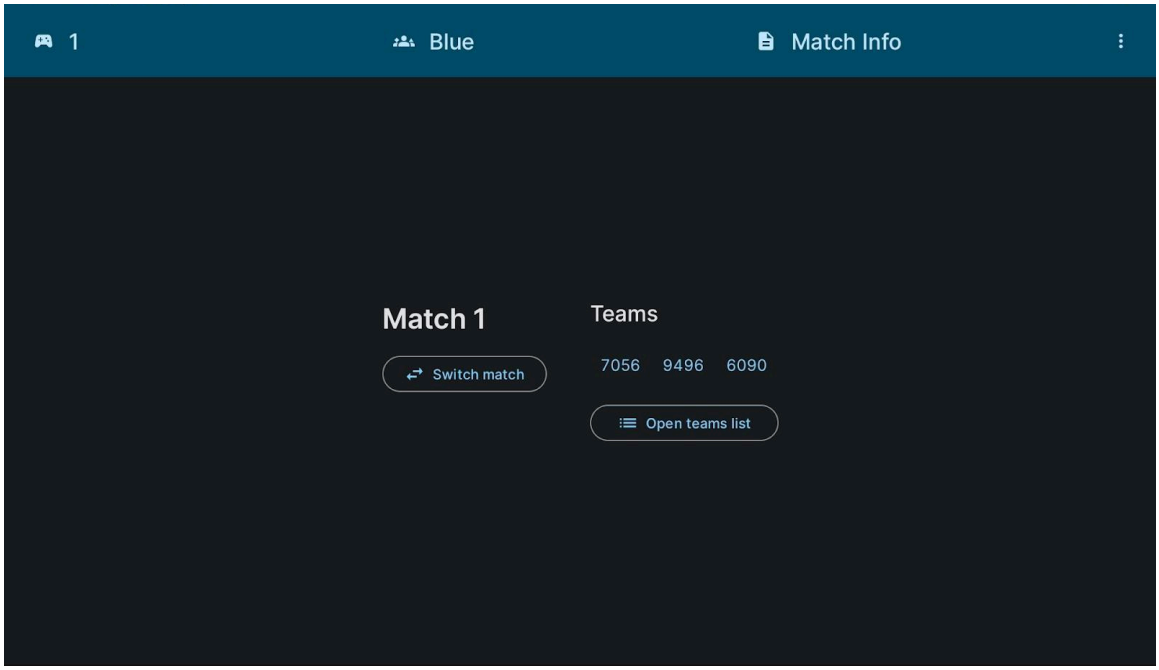
When entering data into Stand Strategist at a competition, the user is first prompted to select or create a user profile (see *Profile Management*). If the user has created a new profile, they will then be asked to select a match schedule file:



Match Schedule Selection Screen

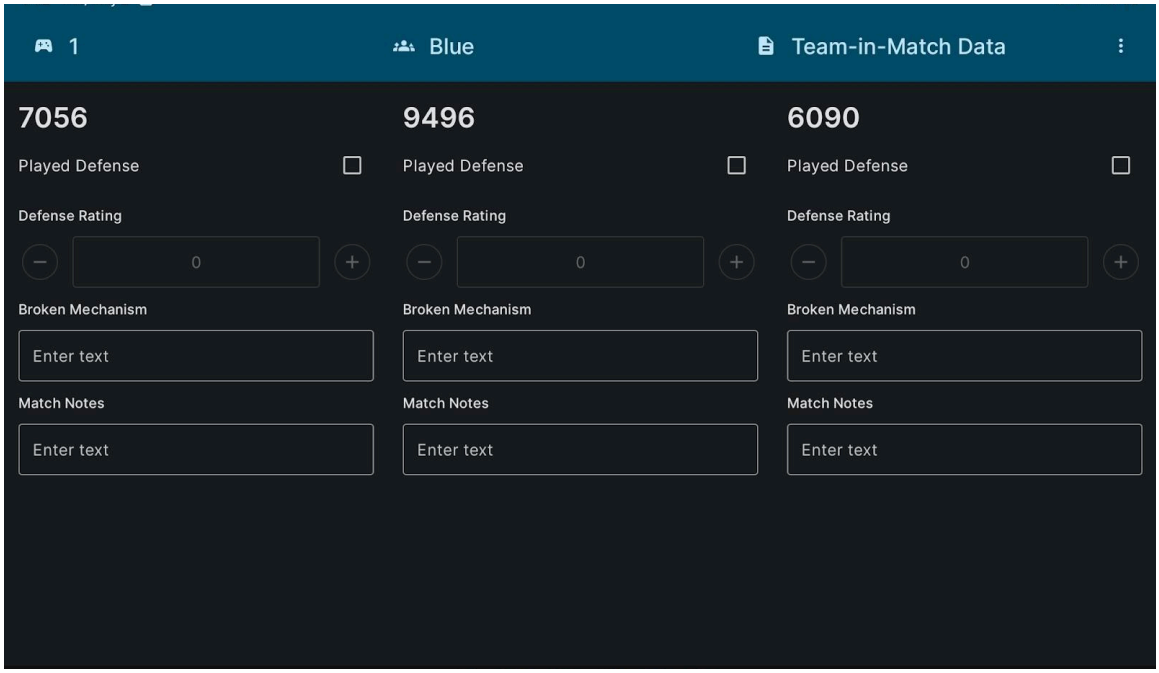
Selecting the ‘Open file picker’ button opens the system file dialog, which allows the user to select a match schedule JSON file generated by our Server.

Once a match schedule is selected, the Match Info page is displayed. The Match Info page contains the match number, the teams in the match on the user’s selected alliance, and two navigation buttons. Pressing on a team number on the Match Info page will bring the user to that team’s Team Screen.

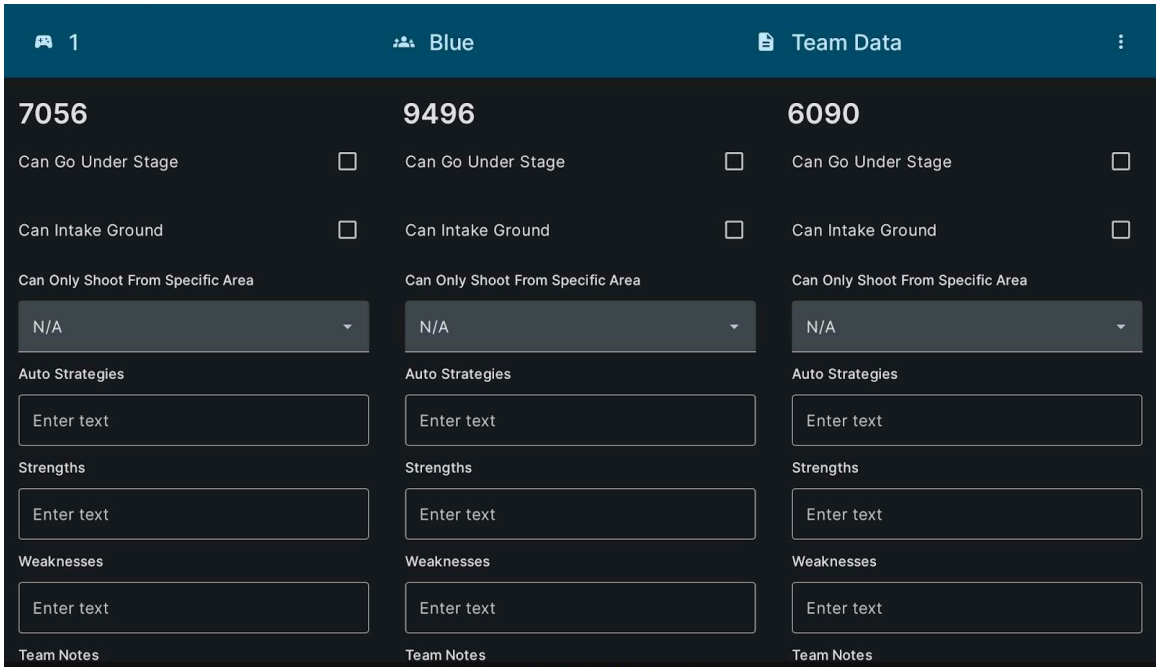


Match Info page

Then, 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

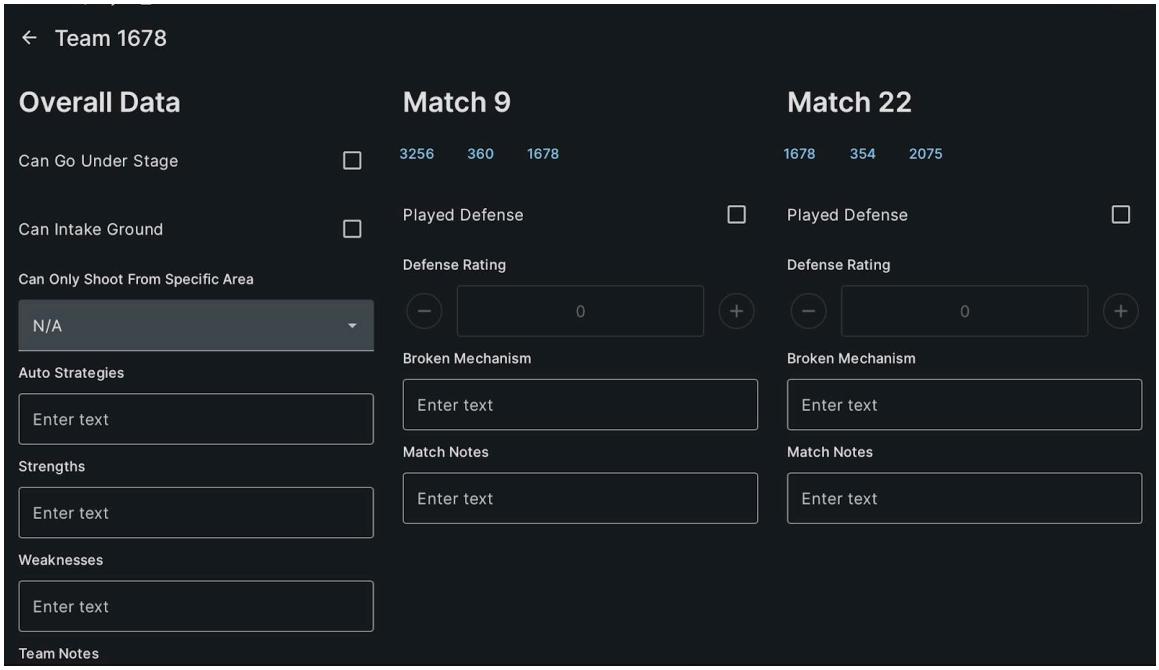


Team Data page

In order to give our strategists a wider range of information about each robot, we collect these datapoints: “Can Go Under Stage”, “Can Intake Ground”, “Can Only Shoot From Specific Areas”, “Auto Strategies”, “Strengths”, “Weaknesses”, and “Team Notes”.

The Team-in-Match Data page records a team’s performance within a specific match. The data entered on this page is match-specific, meaning that the data for a team in a match will only show up for that team in that match.

The Team Data page records more general comments on a team’s overall performance throughout all its matches. The data entered on the Team Data page for a given team is shown and updated for all its matches throughout the entire competition. All of a team’s notes can be viewed by clicking on a team number from the team’s list, searching a team number with the search bar, or clicking on a team number on the Match Info page.



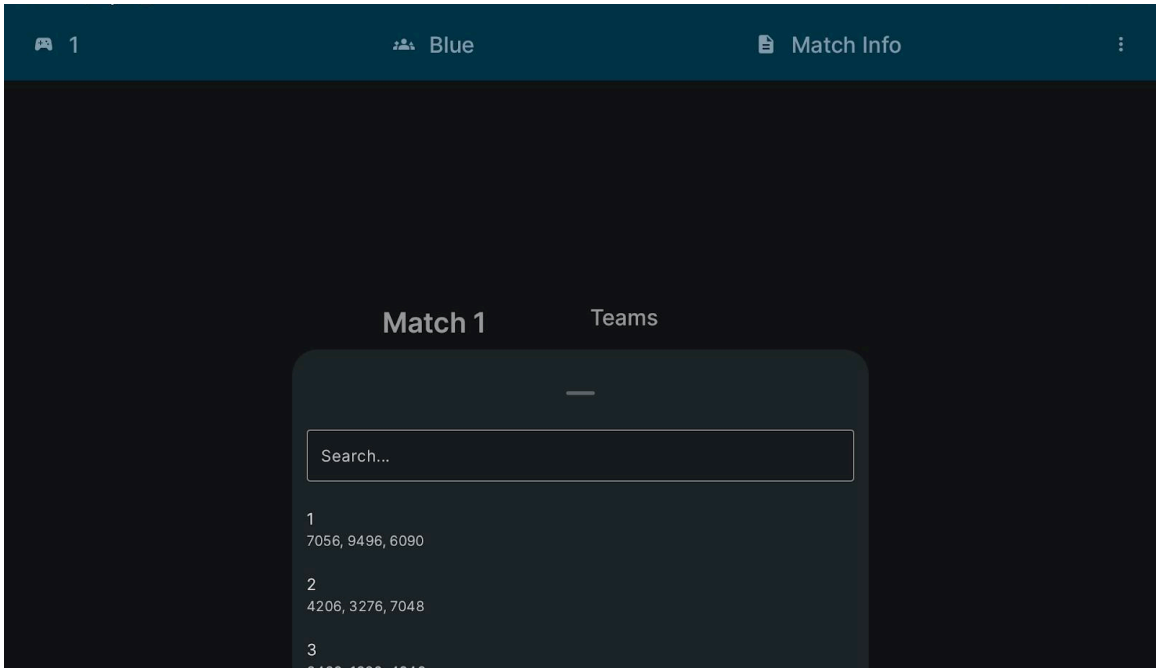
Team Overall Data page

Usage During Competition

Throughout the competition, the two Stand Strategists use this app to take notes on different teams. Then, during the picklist meeting, the Stand Strategists use these notes to give more context to the rest of our data. This lets us know if any numbers might be inflated or otherwise skewed and also gives us a way to keep track of observations not recorded by the rest of the scouting system. All of this gives us greater insight into teams when building our picklist.

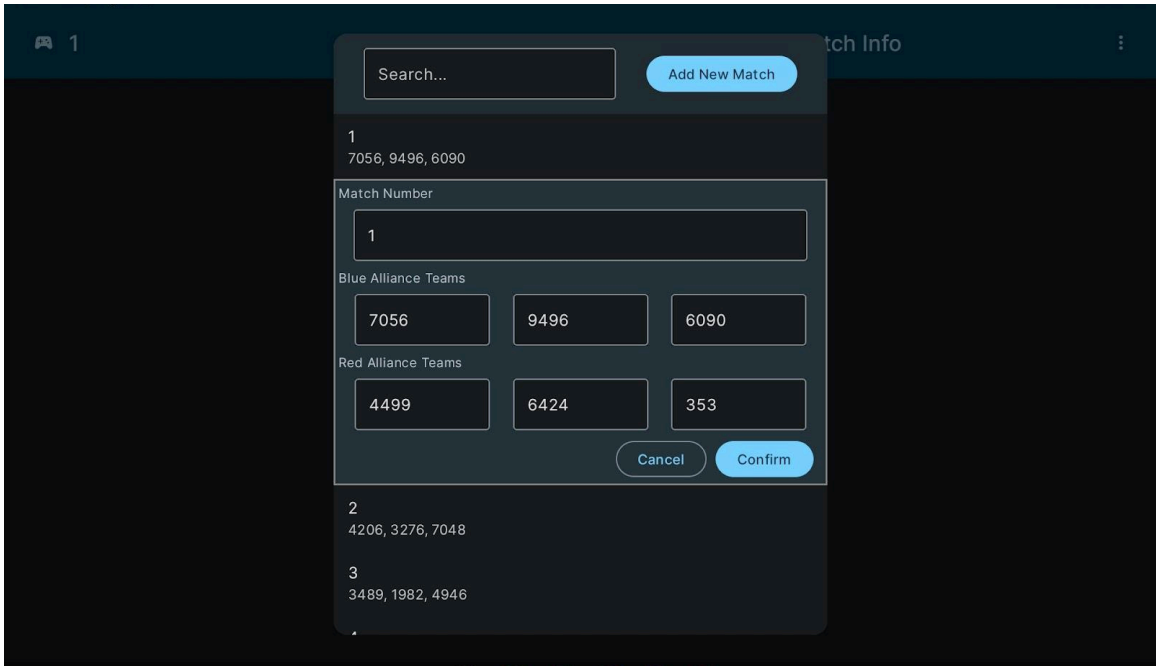
Match Selection

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:



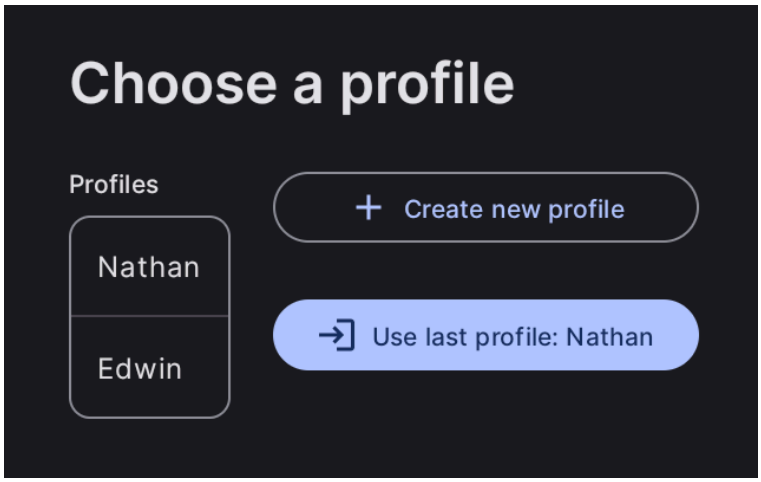
Match Schedule popup

Due to our strategists not having any data on the other robots on the field going into the first few qualification matches, our Stand Strategists wanted to scout practice matches in addition to normal qualification matches, so we also implemented the creation and deletion of custom matches. By clicking the ‘Add New Match’ button, users can create a new match. Users can also click on a match to edit it:



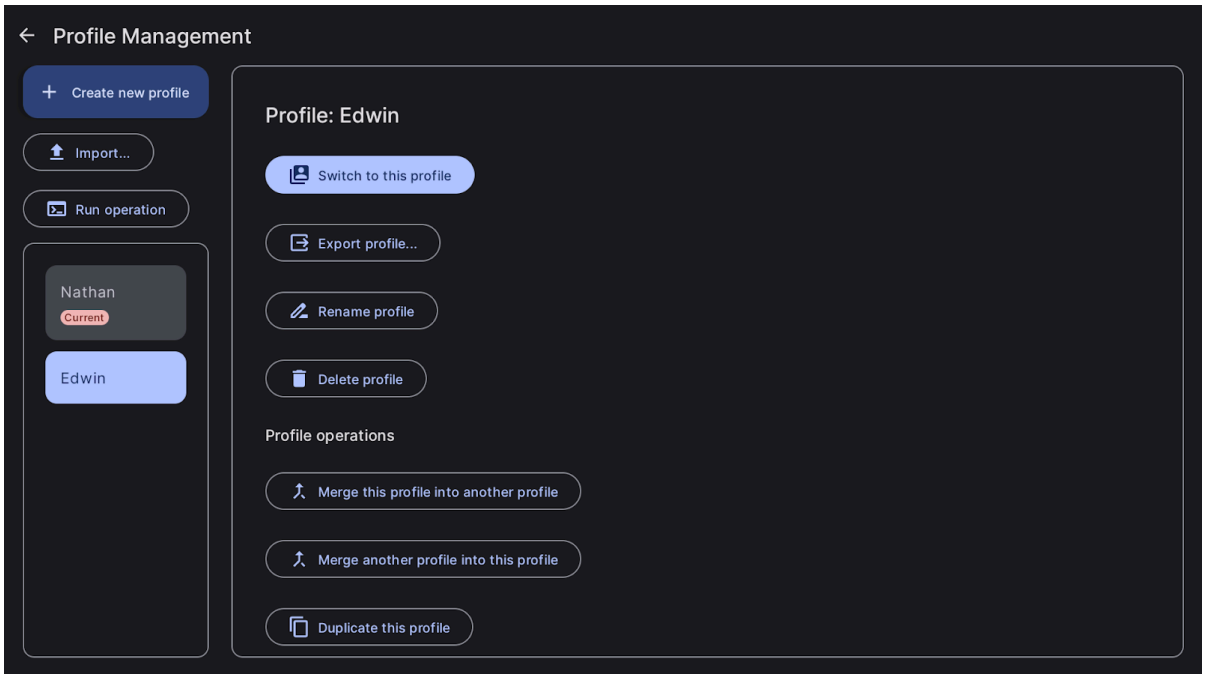
Edit Match Schedule popup

Profile Management



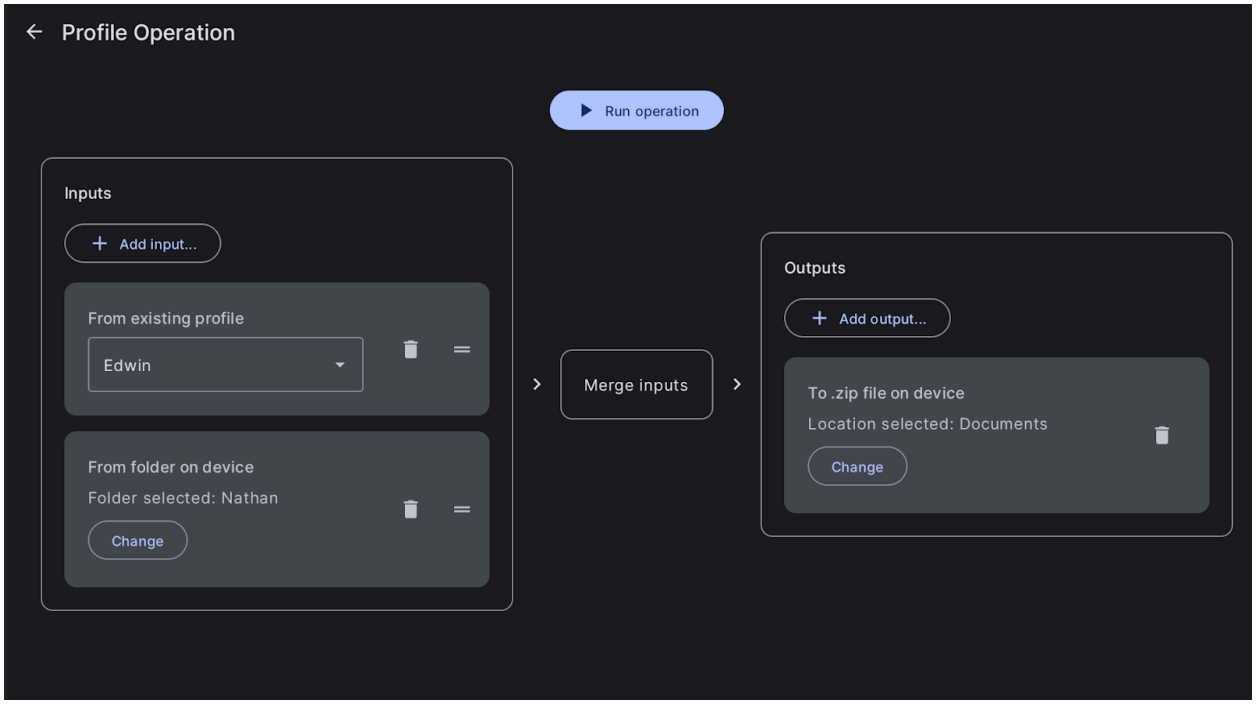
Profile selection screen, shown when the app opens

At competitions, we sometimes need multiple Stand Strategists to collect data with a limited number of tablets. To address this, we implemented profiles, which 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 various options for importing and exporting profiles, as well as the ability to merge profiles. Since profile operations could get complex, we built a screen for creating and running these:



Profile operation screen

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

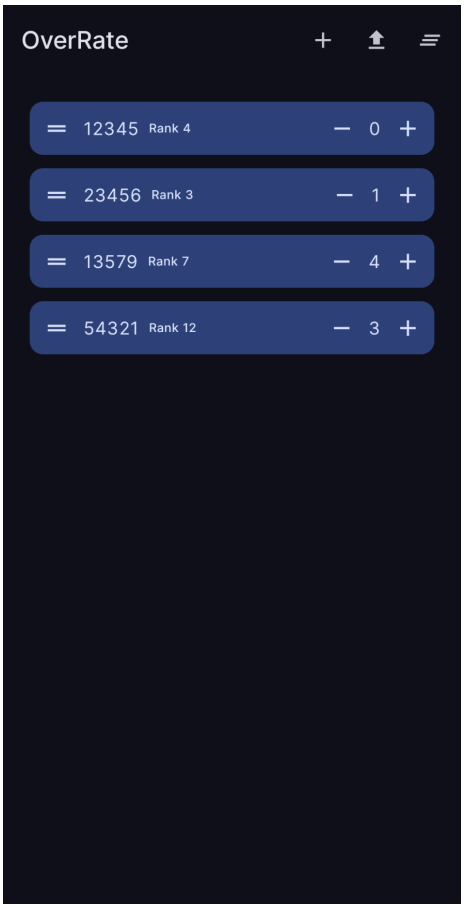
On the device running Stand Strategist, each profile has a separate folder in the storage directory, with each folder containing the profile's match schedule, profile-specific settings, and collected data.

OverRate

Overview

OverRate is an Android app newly created for the 2024 season built entirely using Jetpack Compose. The app is 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, give them numeric ratings, reorder them, and organize them with dividers. The app is intentionally designed to be flexible and support a wide range of use cases and does not promote any particular method of assigning

ratings or organizing teams. For example, a strategist mentor used it on the first day to identify teams that were unlikely to fit well into our alliance and on the second day to rank groups of teams with particular attributes relative to each other. The stored data is not used anywhere else in the scouting system, and so the app serves solely as a scratch pad.

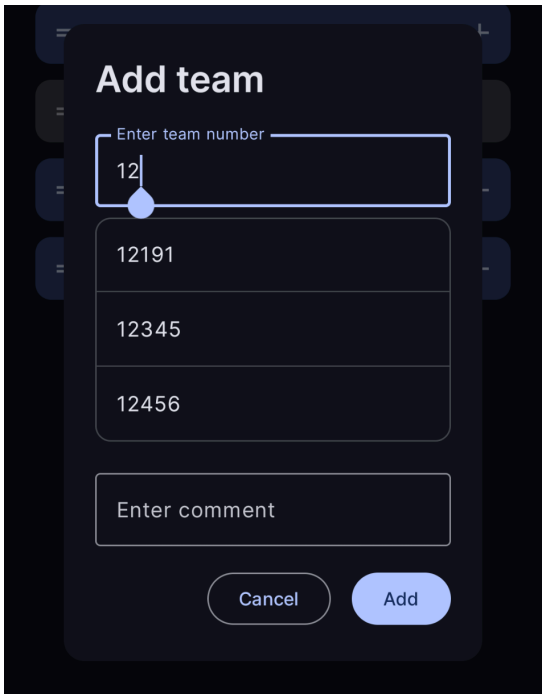


OverRate's main list view

Using the App

Reordering teams

The main list can contain teams and dividers. Each team has an associated rating, which can be edited using the plus and minus buttons. Each item in the list has a drag handle for reordering items. The user can long press items to edit their details or delete them.

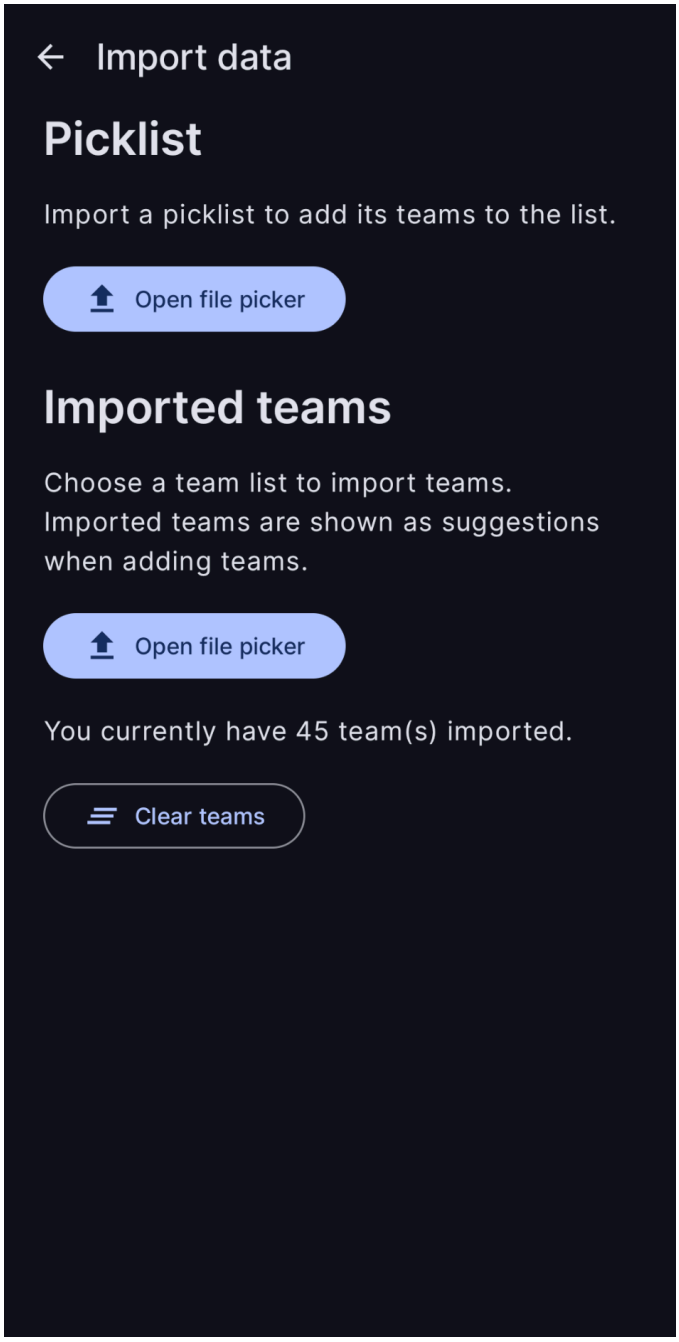


Adding a team

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 shows suggested team numbers from the team list. The user can also add an optional comment, which will show next to the team number. The user can also add dividers separately and edit divider labels.

At the end of a competition, the user can use the Clear button to clear the entire list.

Importing Data

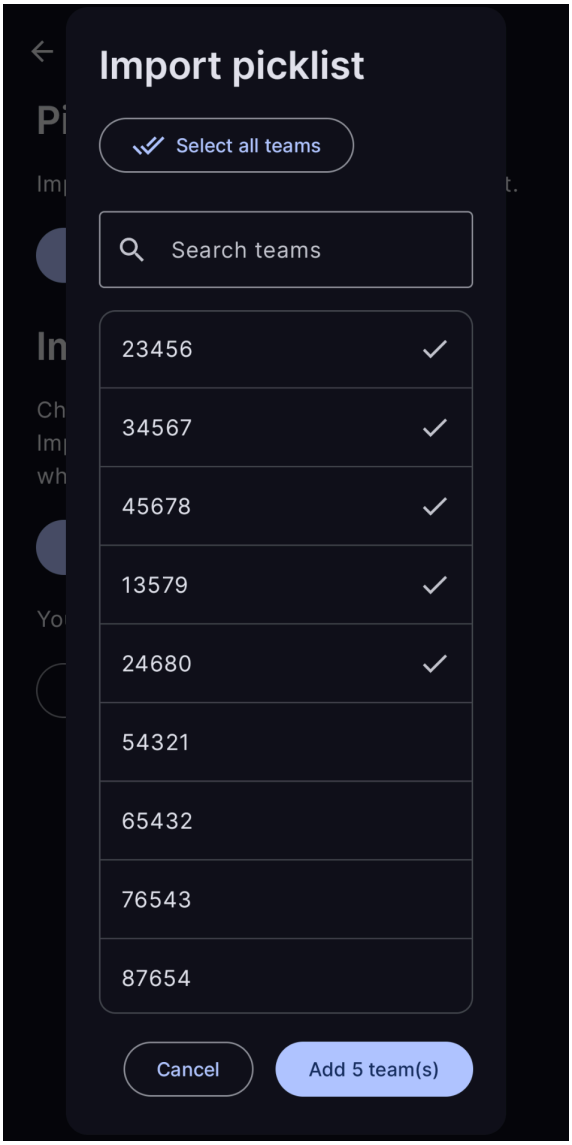


Page for importing data

The user can import teams from a team list to show as suggestions when adding new teams or import a picklist to add its teams to the list.

When importing teams from a team list, the user can select a team list file (the same as the team list file 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 are imported and clear the saved teams.

To import a picklist, the user first exports the Main Editor sheet of the competition's Picklist Editor to a CSV file and then select the file from the app. Then, the user can select which teams to import, and the selected teams are added to the main list. The teams added to the main list also each have a comment showing the original rank of the team in the imported picklist.

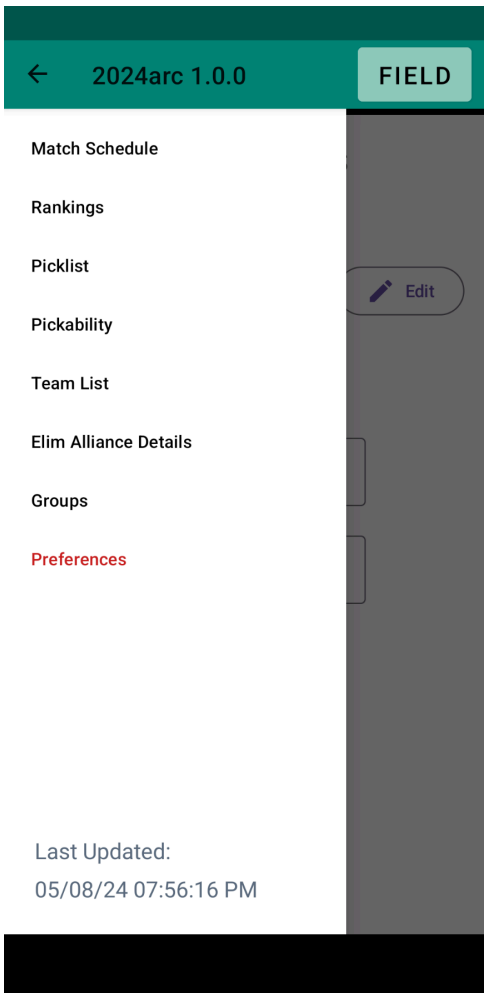


Importing a picklist

Viewer

Viewer is an Android app written in the Kotlin programming language. The app allows strategists to review, organize, and visualize processed scouting data live during competition to create educated match strategies. This year, we have integrated Kotlin's Jetpack Compose framework into many of Viewer's screens.

Navigation

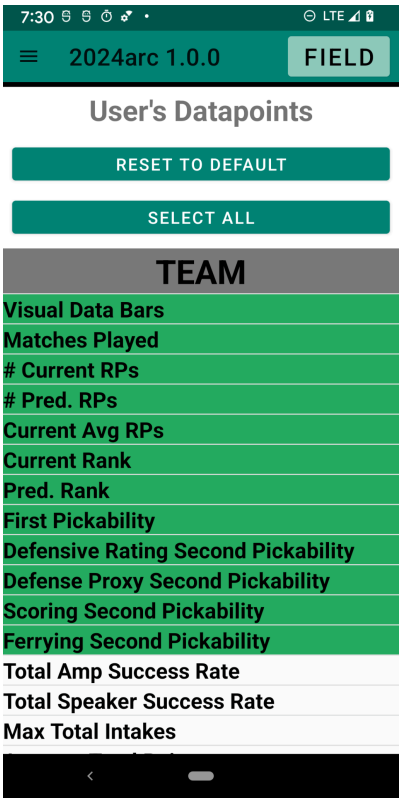


Navigation Sidebar

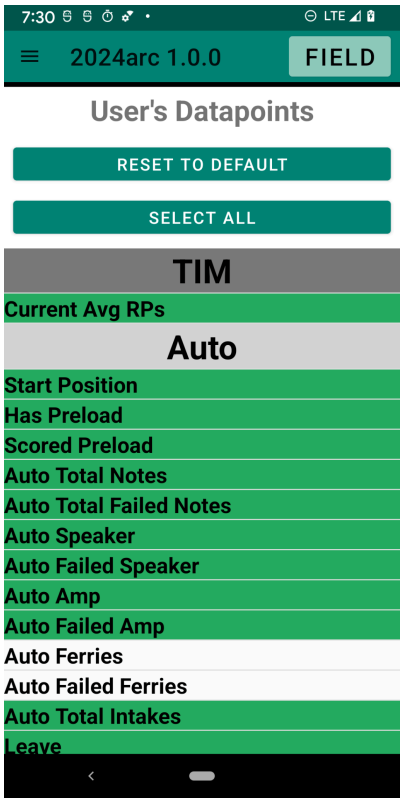
Viewer uses a sidebar to help users navigate through its primary pages. There are also various locations where the user can redirect to other related pages (e.g., from the Match Schedule to a

Match Details page). Users can view the sidebar from any screen or page in the app to optimize navigation—additionally, the top displays the current event and version number.

User Preferences

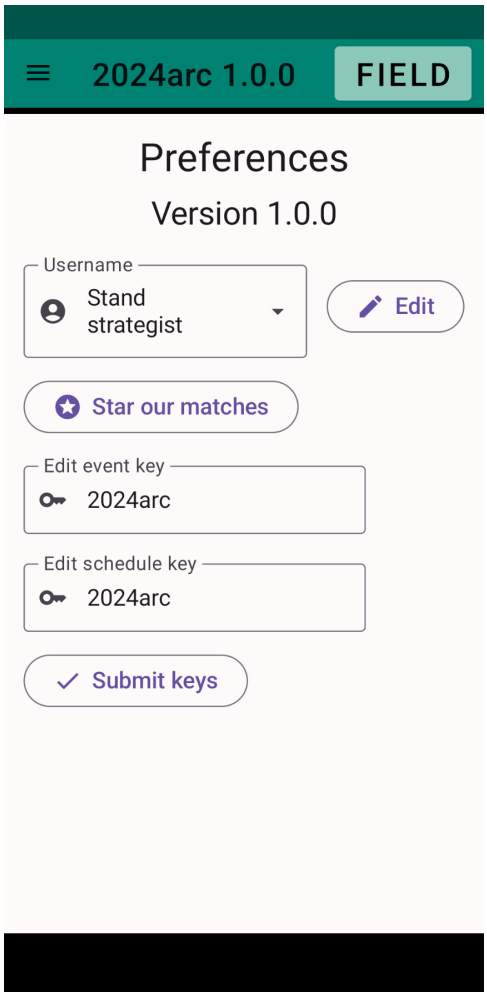


Team Datapoints Preferences List



TIM Datapoints Preferences List

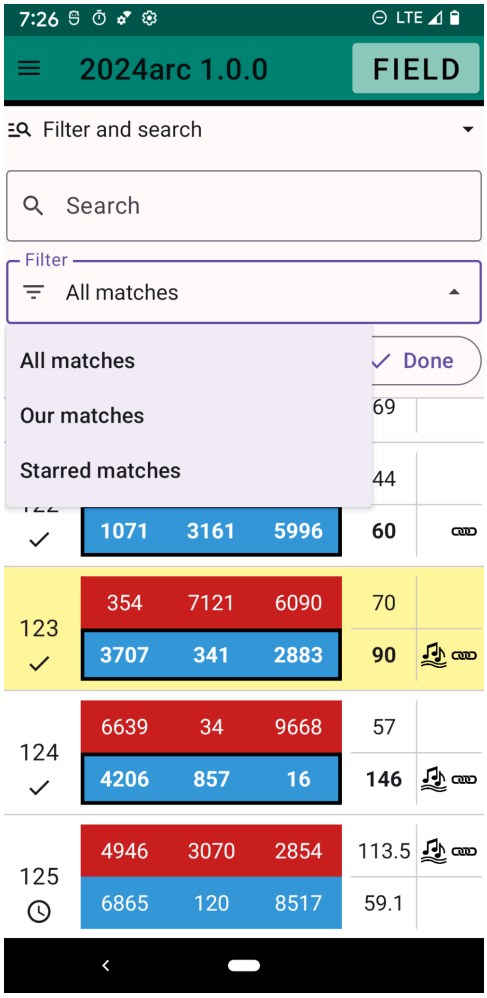
In the Preferences page, users can change which datapoints are displayed in the app. This feature is essential because Viewer has multiple users; one user may want to display more data on a team's intake positions, while another may want more data on a team's endgame. Viewer saves the selected datapoints in a file stored in the Downloads folder on the device, so they don't reset after closing or updating the app. Viewer has multiple profiles, allowing users to have their own different default list of datapoints that the app displays. The only new feature added to the Preferences page in 2024 was the addition of a selector for Team-in-Match datapoints to be displayed in the Match Details screen.



Preferences Screen

In addition, from the Preferences page, the user can change the displayed event, allowing the user to look at information from previous events. There is also a button to star all our matches, allowing users to see them when filtering for all starred matches on the Match Schedule page.

Match Schedule



Match Schedule

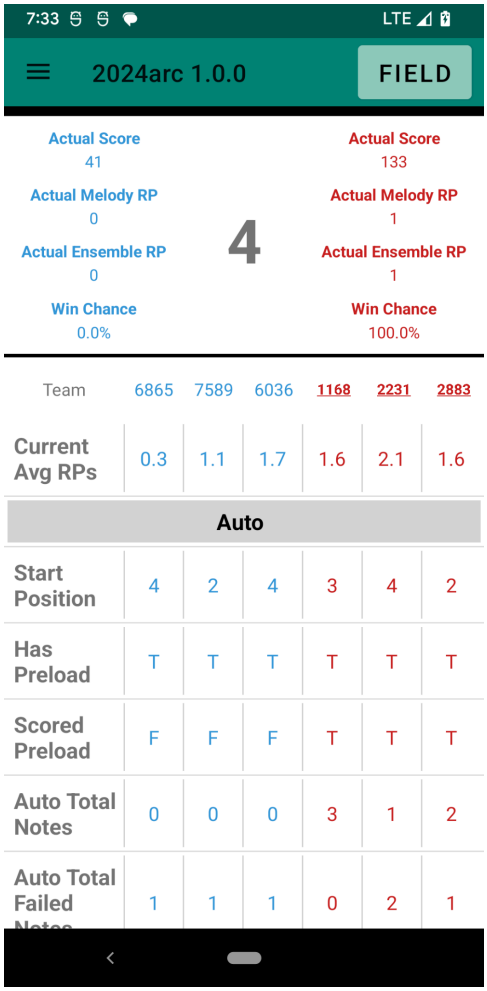
The Match Schedule page displays the match schedule, including which teams are in each match, the total match score and RPs earned. Viewer displays predicted data for unplayed matches. A checkmark icon is displayed below the match number if the match has been completed and a clock if it has yet to be played. The alliance that won the match is marked with a black border. A note icon is also displayed if the alliance earns the Melody RP, and a chain icon is displayed if it earns the Ensemble RP.

Users can use a search bar to filter matches by a team number and submit the search to go directly to the team's Team Details page. A drop-down in the search menu allows filtering by All

Matches, Our Matches, or Starred Matches. A user can star a match by long pressing it, and a star symbol will appear above the match number. If a team is starred, the matches they play in will be highlighted. The list of starred matches is saved to a file in the Downloads folder, so they aren't reset if the app is reopened.

A new feature was added after Week 6 this year to have the Match Schedule automatically scroll to the most recently played match whenever the page is opened unless the user was previously looking at a Match Details page.

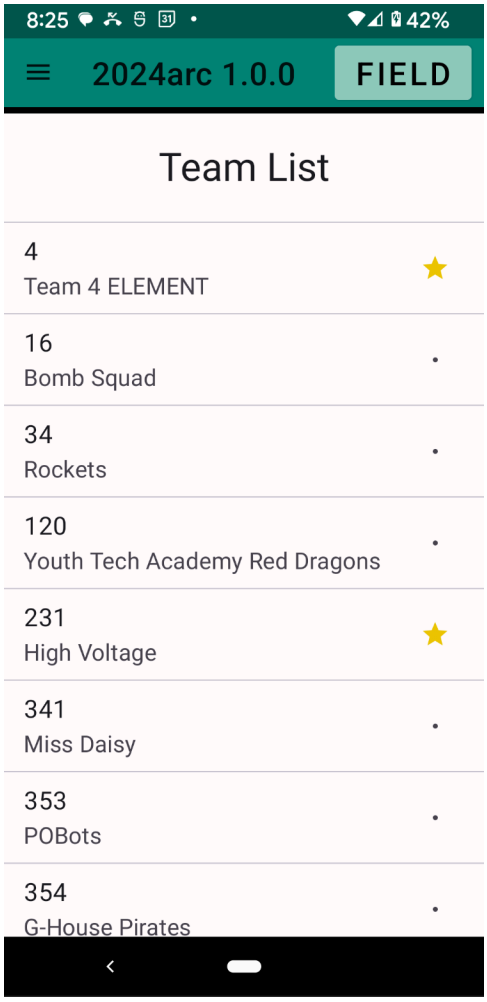
Match Details



Match Details for Played Match

After tapping on a match in the Match Schedule page, the match's Match Details page opens. If the match is unplayed, the datapoints displayed will be predictions and averages. If played, the datapoints display the data collected by our scouts and data pulled from The Blue Alliance. The teams in the alliance that won the match are bolded and underlined. In the header, alliance-specific data is displayed (e.g., predicted score); in the data section, a table displays team data. Users can access multiple screens through various shortcut buttons to improve navigation. These include clicking on a team number to go to the team's Team Details page and a few newly added shortcuts such as long pressing on Current Avg RPs to go to the Rankings page, long pressing rankable datapoints to show the rankings page for that datapoint, and long pressing a datapoint from the Auto category to open the Auto Paths page for that team. Another feature that was added this year was having the displayed datapoints be affected by the user's datapoint preferences.

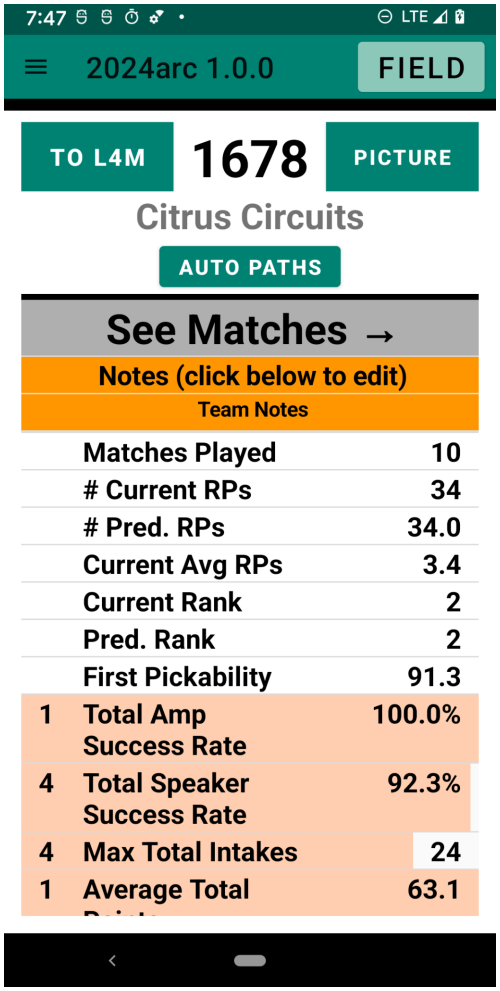
Team List



Team List

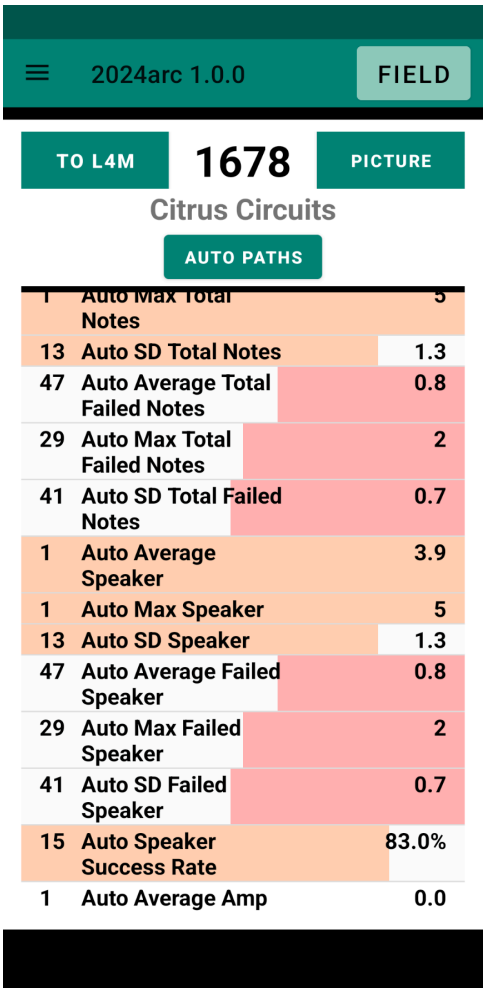
The Team List page allows users to view a list of all teams participating in the event. Users can also star teams by tapping to the right of the team number. Users can go to the Team Details page by clicking on a team. On the Match Schedule page, a match with a starred team turns a light yellow; if it has more than one starred team, the match will turn a darker yellow.

Team Details



Team Details

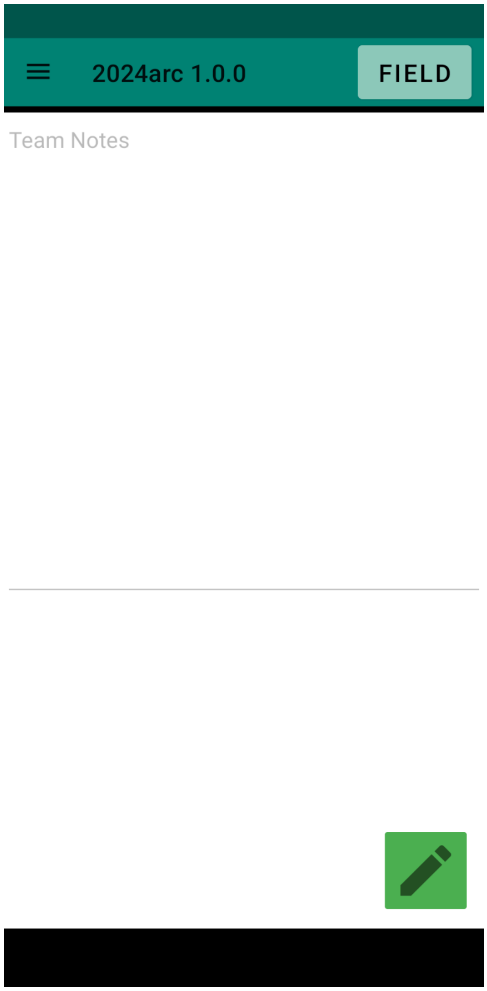
Every team at the competition has a Team Details page featuring calculated datapoints such as averages, standard deviations, and more. Rankings are shown on the left side column to display the team's ranking for that specific datapoint compared to other teams.



Data Bars

On the Team Details Screen, users have the option to display colored data bars behind each datapoint to visualize how that team compares in that datapoint to others. The percentage of the orange-colored data bar is the team's value for that datapoint divided by the highest value across all the teams at the competition. Data bars for datapoints like incap time and fouls (datapoints where the higher the number of the datapoint, the worse the team's pickability is) go from right to left and are colored red. Users can also navigate to the team's matches through the See Matches header.

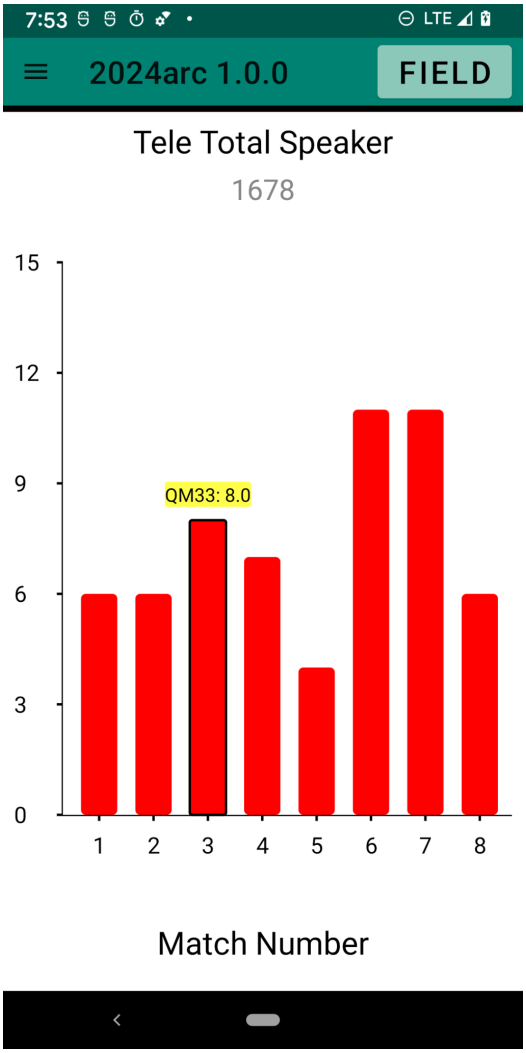
Team Notes



Team Notes

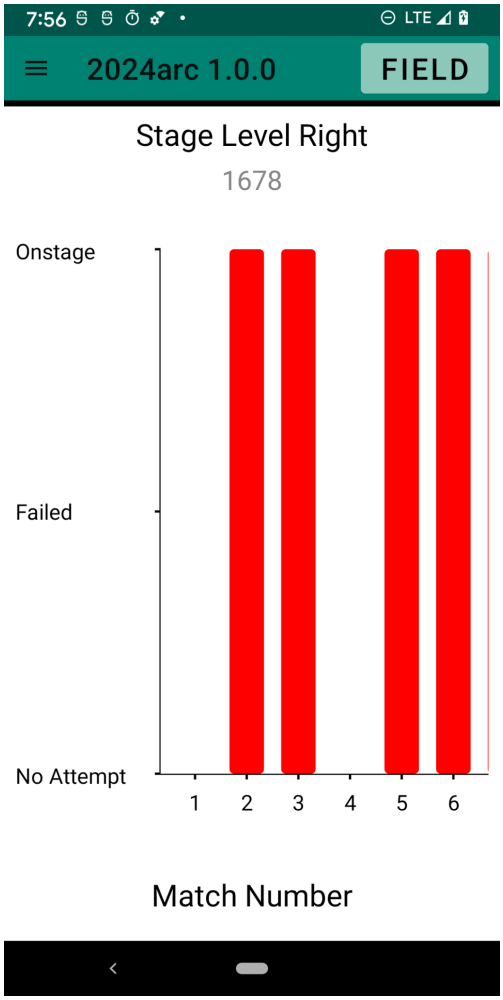
Users can write notes on teams by clicking on the orange Notes button on each team's Team Details page. We store the notes they take on the Grosbeak web server, and all the other devices pull from that so that all other users can share notes.

Data Graphs for Specific Datapoints



Data Graph for Tele Total Speaker Notes

Tapping on a datapoint in a Team Details page opens up its TIM (Team-in-Match) Data Graph—a bar graph of the match number vs. the datapoint's value. For example, by tapping on the datapoint "Tele Average Total Speaker," a graph will appear. Each of this graph's bars shows the number of Notes scored into the speaker during Teleop by the team in the corresponding match. Tapping on a bar displays the specific qualification match and datapoint value in the format of QM{match number}: {datapoint value}, e.g., QM33: 8.0. The axes are labeled differently for unconventionally graphed datapoints such as the stage level or a boolean value. The photo below displays an example of this.



Stage Level Right Graph

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

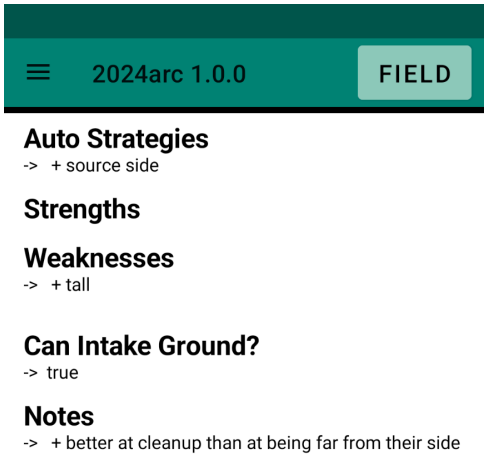
Users can access the Team Rankings Page through the navigation sidebar. By pressing whichever one they want to display, they can switch between displaying current and predicted rankings.

Rankings for Specific Datapoints

Rankings for Tele Average Total Notes

By long pressing on a datapoint on a Team Details page, a user can view a ranked list of all the teams by that datapoint. Viewer displays these ranks on the left side of Team Details cells, but Viewer doesn't refresh them as often to avoid lag. By viewing a ranked list, users can see which teams are above and below a particular team. By clicking on the cells in the ranked list, users can open a team's Team Details page. We highlight the team that the user was previously viewing for easy visibility of their performance.

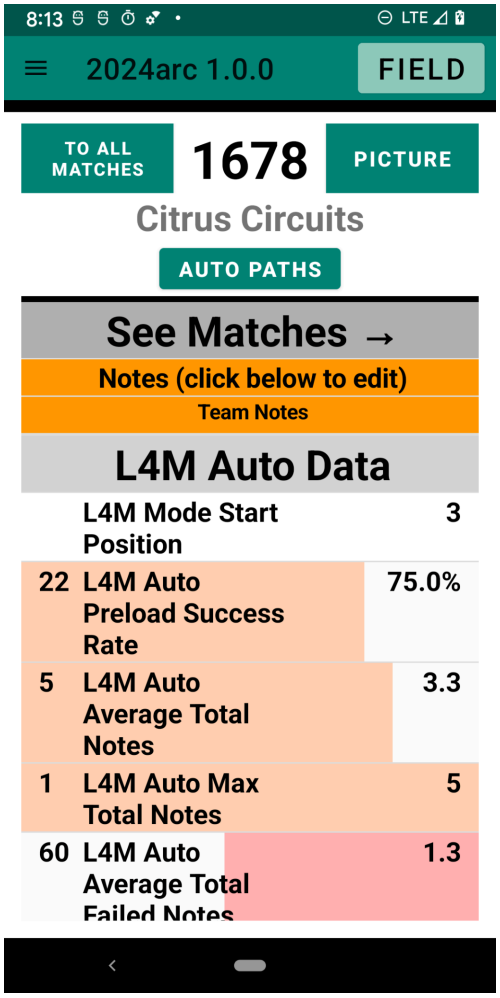
Stand Strategist Notes



Stand Strategist Notes

Our Stand Strategists collect subjective data about teams that Subjective Scouts do not. We upload that data to our server, which Viewer displays in the Stand Strategist Notes page. Users can navigate to this page by clicking the Stand Strategist Notes field on a team's Team Details page.

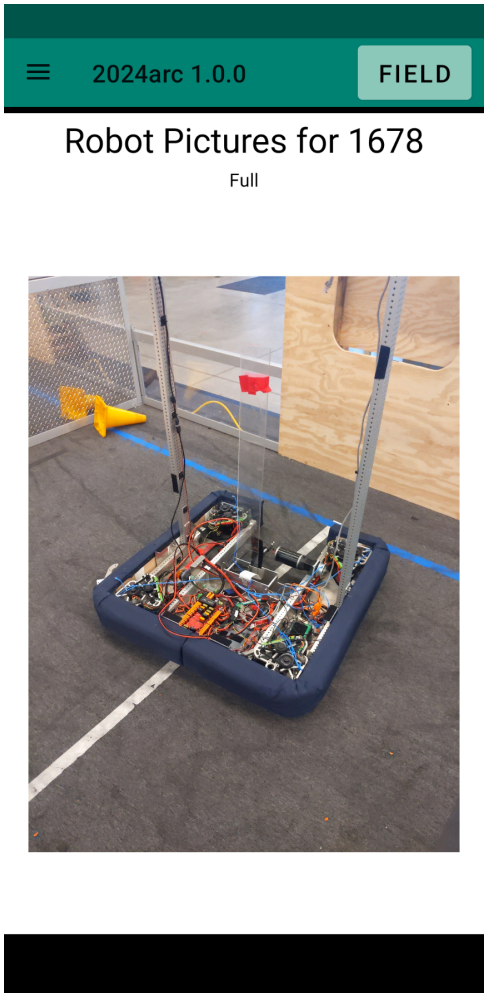
Last Four Matches



Last Four Matches

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

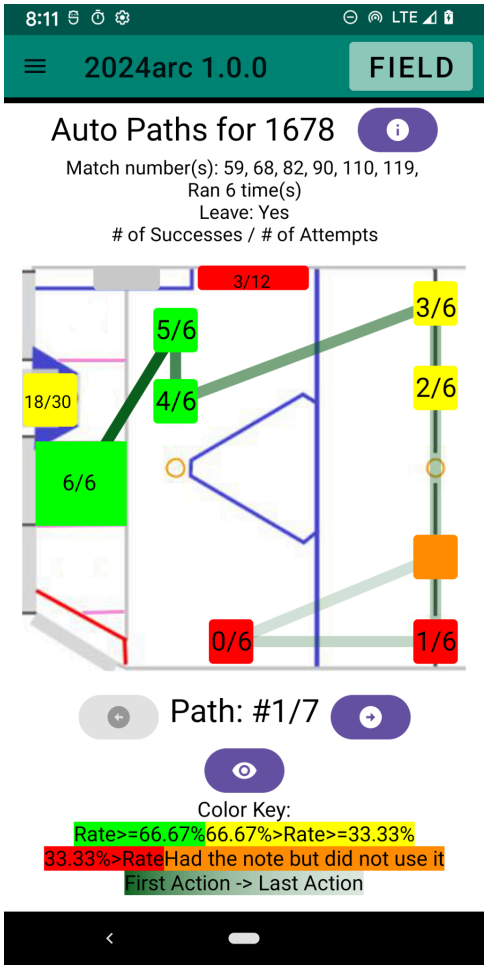
Robot Images



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 pictures showing the robot from different angles. The Pit Scout uses the Pit Collection app to take these photos, and we manually transfer them to the other phones by plugging in the phones to the Server.

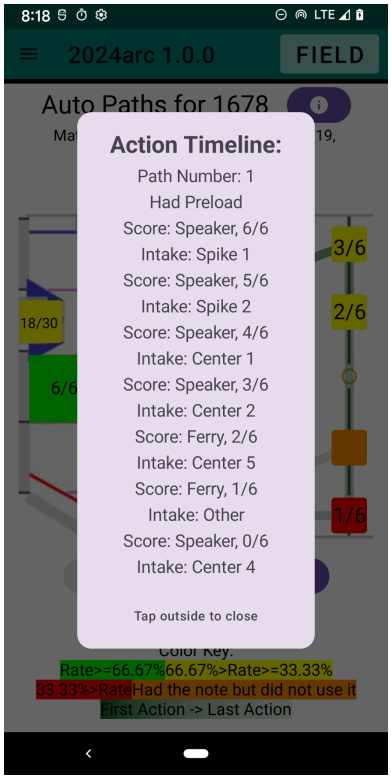
Auto Paths



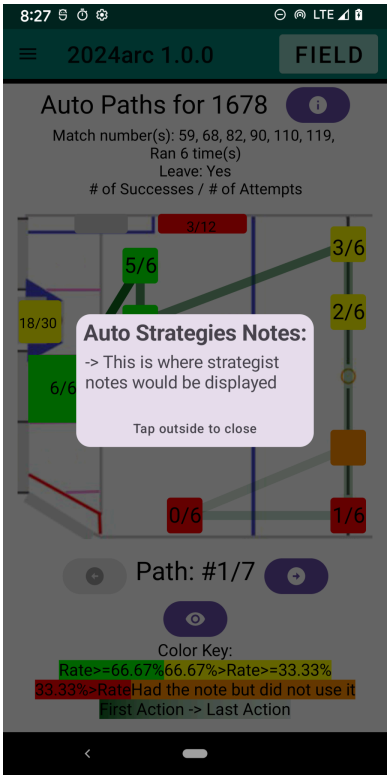
Auto Paths

Auto Paths was a new feature added for Championship 2023 once strategists realized how vital having a compatible auto was. We've continued to use and update this feature throughout this year. This feature visualizes the Auto Paths of teams, starting with their most common autos and scrolling through other Auto Paths they have. They can navigate the Auto Paths by swiping to the sides or pressing the arrow buttons. 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 Speaker, the team attempted to score a Note into the Speaker *y* times and succeeded *x* times. If it shows *s/a* over an intake position, the team tried to intake and score the Note there *a* times, and succeeded in intaking and scoring it *s* times (we consider failed intakes and failed

scores the same for Auto Paths). If a team collected a Note and did not attempt to score it, that position would be colored orange. There are also displays for ferries and intaking Notes that were not in regular positions.



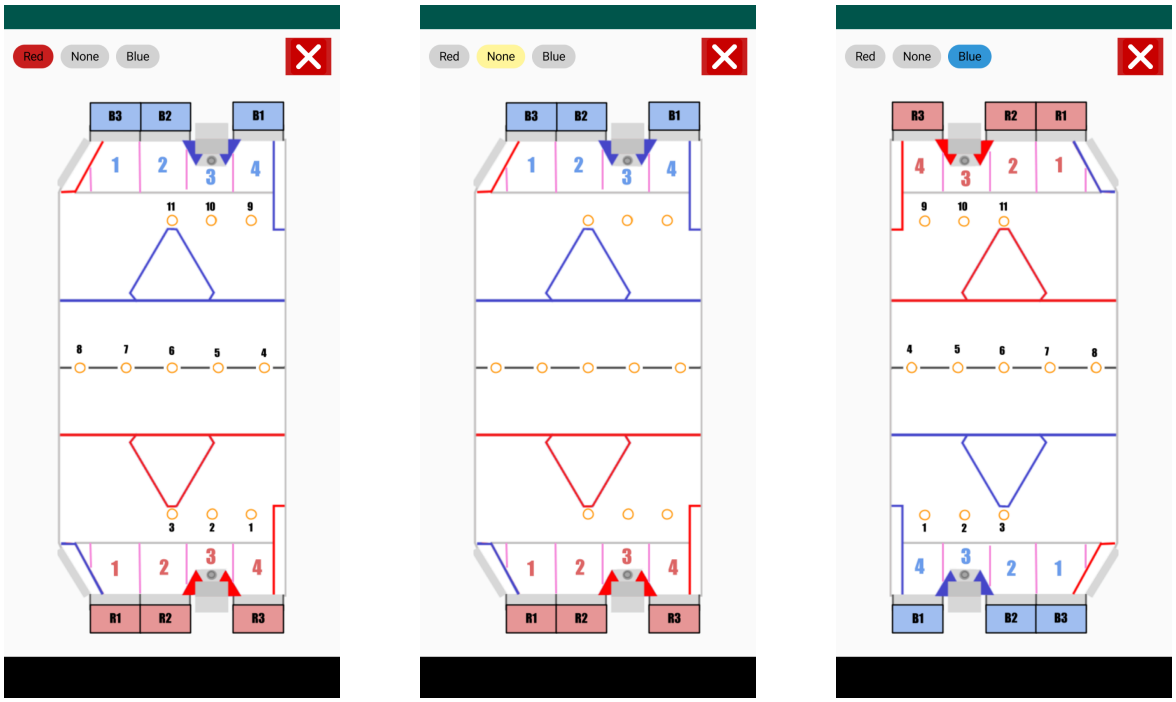
Action Timeline



Strategist Notes

This year, we added several features to convey additional information, including action timelines, notes taken by strategists on autos, and match numbers that the auto was used in, which, when pressed, open the corresponding Match Details page. The app also now shows color-coded paths on a map to help visualize actions.

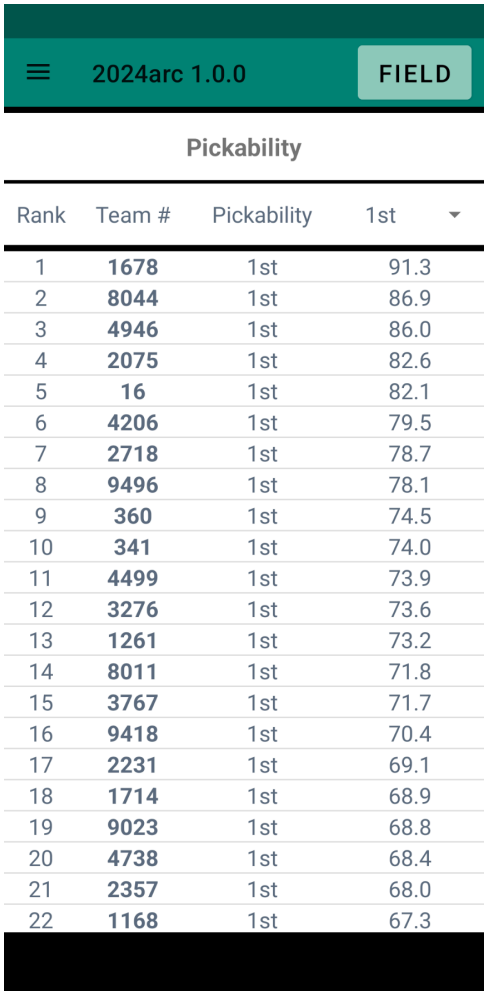
Field Map



Field Map

In Viewer, users can view the Field Map by pressing the FIELD button at the top-right. This view shows the possible starting positions, driver stations, and positions of the starting Notes. Users can switch between the red and blue alliance to reverse the map orientation.

Pickability



Rank	Team #	Pickability	1st
1	**1678**	1st	91.3
2	**8044**	1st	86.9
3	**4946**	1st	86.0
4	**2075**	1st	82.6
5	**16**	1st	82.1
6	**4206**	1st	79.5
7	**2718**	1st	78.7
8	**9496**	1st	78.1
9	**360**	1st	74.5
10	**341**	1st	74.0
11	**4499**	1st	73.9
12	**3276**	1st	73.6
13	**1261**	1st	73.2
14	**8011**	1st	71.8
15	**3767**	1st	71.7
16	**9418**	1st	70.4
17	**2231**	1st	69.1
18	**1714**	1st	68.9
19	**9023**	1st	68.8
20	**4738**	1st	68.4
21	**2357**	1st	68.0
22	**1168**	1st	67.3

Pickability

Users can rank teams based on pickability metrics such as 1st, 2nd, Defensive 2nd, or Scoring 2nd. In previous years, only the 1st and 2nd pickability values existed. This year, we added additional pickability metrics to better judge different aspects of teams. We calculate these values using preset weightings based on what is deemed most important for a chosen robot (see Pickability). Users can switch between the various pickability values using the dropdown in the top right. Clicking on a team number will open the team's Team Details page.

Picklist

Last year, Viewer had live and offline picklist editors that allowed users to edit their personal picklists and see the most updated ones. It has two versions, online and offline. The online version fetches the most recent version from Grosbeak and is not editable, while the offline version is editable and is stored locally on the device. We did not use this feature this year as our new app, OverRate, allowed strategists to rank teams on demand.

Elim Alliances

Last year, Viewer had an Elim Alliances feature to give further insight into playoff matches and alliances. It displayed each alliance and predicted scores for Auto, Teleop, Charge, and Total based on the averages of all the teams in the alliance. We did not use this feature this year as we didn't find it helpful, but we may reuse it in future years.

Data Refreshing

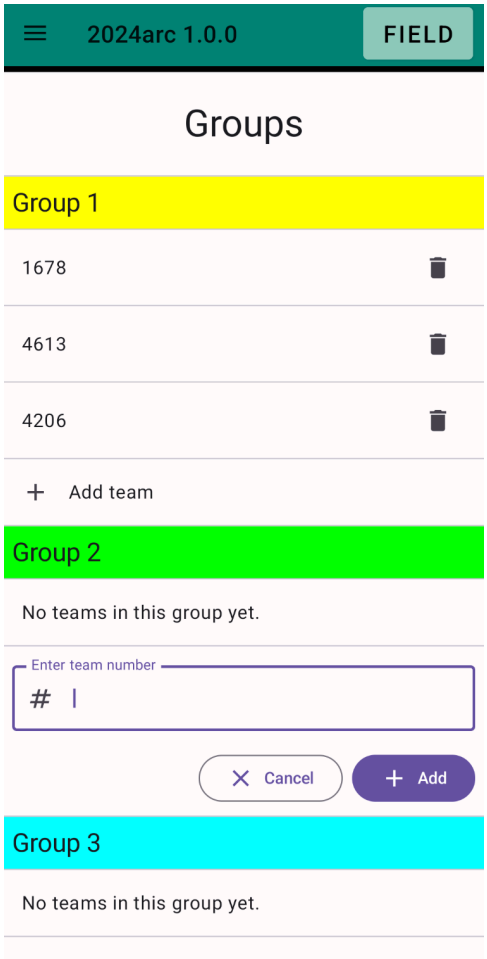
Viewer's data refreshes automatically by fetching data and updating caches at a set time interval. When the data refreshes, Viewer runs callbacks in all active pages to update the UI.

During Championship 2024, we had to temporarily increase the request timeout time and the refresh interval because the cell service was too slow in the large venue. Due to the slow service, the requests to fetch data from Grosbeak would time out. This prevented Viewer from getting new data. In addition, if the app was closed and then reopened later, users would not be able to access any part of the app due to not having any data at all. In future years, we may try to store some data locally in case similar issues occur so that users would at least be able to view data from the last time the app got data.

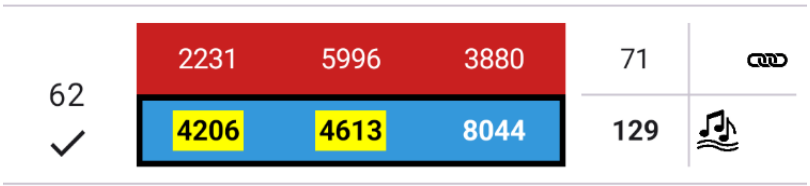
Groups

Groups is a new feature for this season, allowing users to put teams into color-coded groups and have them highlighted in the match schedule. Users can organize teams into groups for

any purpose, including marking teams to be compared for picklist positions or marking teams as potentially affecting matches important for rankings.



Groups page



A match with teams from a group

Server

During a competition, the Server works in tandem with the Scouts to provide accurate data to the strategists and the Picklist. As matches progress, the 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 on certain game actions. The Server runs many calculations on the data and then stores the raw and calculated data on a local MongoDB database on the designated laptop. When the server is done, the data is uploaded to a MongoDB database in the cloud, which is then accessed by our Grosbeak web server. Finally, the Viewer app requests the data from Grosbeak. For more information on databases, see Section 4.1.6.1 Local and Cloud Databases in the 2020 Whitepaper.

Schema

In our scouting system, Schema is the blueprint for our data structure. We use YAML files to store the standardized data structure all our apps use. These YAML files allow our developers to modify data fields in a single location, thus increasing code reusability between different FRC games and modifications we want during the season. Our ultimate goal is to remove all hard-coded data fields and replace them with schema data fields. A structured data system allows quick and easy changes throughout the code, allowing us to invest time in other tasks. Schema associates the most calculated data fields with a basic type of calculation, such as averages or counts, allowing developers to quickly create new calculations without writing new code. Developers can also easily change the weighting for calculations that combine other metrics without altering Server code, making it easier for picklist strategists to change the value of weights during competition. Data field bugs (like typos) can be quickly fixed during competitions without wasting time.

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 into the database. Calculations include an LFM (last four matches) datapoint for objective data, which shows a team's performance in the last four matches. The `server.py` imports these calculations in the order listed in `calculations.yml` and simply calls the `run()` function to execute the calculation. For more information on specific calculations, see Section 4.1.7.2 Consolidation and Objective TIM Calcs in the 2020 Whitepaper.

Team and TIM Data

Data collected by Objective Scouts is used to record objective Team-in-Match (TIM) datapoints of how a team played in a match, including how many Notes they scored or failed to score in each location, if they parked, trapped, or went onstage, their cycle times, incap time and more. Objective TIM datapoints are then used to calculate objective team datapoints, including the averages, standard deviations, percentages of success, medians, modes, and maximums of all the objective TIM data, as well as the objective TIM data of the robot's last four matches. It can also calculate their expected cycle times, how many matches they played, harmonized, were incap, and more.

The quickness score and field awareness score datapoints that Subjective Scouts collect about a team during a match are recorded as subjective TIM datapoints. The subjective team datapoints, including field awareness, quickness, ability, defensive ability, and proxy ability, are calculated using the averages of the subjective TIM datapoints.

Auto Paths

The Auto Paths calculation collects every action a team performs during the autonomous period, including scoring and intake actions. By looking at the order in which the actions were done, we can get a sense of the general path a robot takes during auto. Our strategists mainly

use the data to check if our autos are compatible with the autos of other teams. Data from our Auto Paths calculation can be used in Pickability calculations and is an important part of assessing a team's overall ability.

When calculating Auto Paths, the timeline data collected by each different Scout is consolidated, filtering out all non-auto actions and creating a single list of auto actions sorted in chronological order. However, rather than listing every auto path by match, similar auto paths are matched together to calculate the success rates of every unique path per team. Auto paths are matched together based on start position and intake/score attempts since these actions will be unique for every different auto path. By matching auto paths based on attempts rather than successes, each auto path can have success rates for scoring.

This year, we made minimal changes to Auto Paths; instead, we made many changes in their display. For example, we added additional pop-ups that displayed data such as a timeline of actions. For more details, see Auto Paths in Viewer.

Pickability

Pickability is a metric that allows teams to be pre-sorted before strategists discuss and refine the picklist order. Each team has a first and second pickability, which approximates their suitability as a first pick and a second pick, respectively. Pickability is calculated using a weighted sum of teams' values for certain datapoints.

After an event, all the teams are sorted into their perceived ranks, such as high, medium, low, etc. Next, a few different multivariate linear regressions are run to see what model best predicts each team according to the assigned ranks. We then use the given coefficients as our weights for each component datapoint. After this, we calculate the pickability values for each team and rank them from best to worst for every model estimated. Finally, we have a blind comparison to see what model leads to the rankings that make the most sense and choose to use that pickability model for the following competition. This would be run for both first and second pickability metrics.

This year, we added the ability for Stand Strategist datapoints to be included in the pickability calculations. Since Stand Strategists collect important data related to defense, we included

their datapoints in our Second Pickability calculations. Also, since there were many possible strategies for the second pick to play in this year's game, we made multiple second pickability metrics for each strategy: ferrying, defense, and scoring (offense). Additionally, the pickability metrics are calculated based on data from the last four matches to show the recent performance of the robots rather than their overall performance. These metrics are helpful when robots have a poor first few matches.

Expected Fields

A new "Expected Fields" calculation was added to our Objective TIM data this year. This calculation produces a collection of datapoints to determine the number of cycles a robot can do. We developed this calculation this year since our 2024 Match Collection app allows scouts to record specific intake locations, such as "far," "amp," or "center" (for more details, see the Teleop section in Match Collection). To do this, the calculation uses each robot's intake and score or ferry location in every match. Additionally, to find an expected cycle time, the calculation uses the time from the start of Teleop to the time at which a robot goes to climb. In a game where each robot can play many different roles, these datapoints helped our Strategists compare robots' offensive capabilities and served as a useful datapoint in predicting match scores.

By weighting each intake and score location differently, we can accurately predict how many full field cycles a robot can do and the number of Notes it's expected to score. Speaker and amp scores were weighted differently because amp scores took slightly longer than speaker scores. Ferried Notes were weighted lower than scores since they are faster as well. Additionally, a robot's incap time is subtracted from its total Teleop time.

Predictions

During competitions, Server makes predictions on future match scores, ranking points, and probabilities of winning, in addition to final rankings for each team. Predicted Alliance In Match (AIM) data is calculated using Objective Team and The Blue Alliance (TBA) data. This data

collection includes an alliance's chance of getting each RP, final score, and probability of winning.

This year, score predictions were more challenging due to the non-linear nature of amplification scoring. A simple sum of team averages for speaker scores, amp scores, etc., did not work this year since the number of amplified speakers (the main contributor to team score) heavily depends on a team's alliance partners. We initially struggled with developing a one-size-fits-all score prediction formula and experimented with several competing formulas at our first competition. We found that during week 2-4 events, approximating a speaker score as 3.4 points regardless of amplification worked well. However, this formula was not scalable to lower- and higher-scoring competitions. We eventually settled on a formula derived from a team's expected Notes scored (see the Expected Fields section) and the assumption that all speakers are amplified. Using this model, we solve a system of equations for amp and speaker scores. The equations model an alliance that maximizes their score by balancing amplified speakers with amp scores. Appropriate constraints are included: among others, the alliance's total Notes scored must equal the sum of their expected Notes. We then solve for the number of speakers and amps that satisfy the system. This formula worked well for us throughout Sacramento Regional and East Bay Regional (~7 points off on average per match).

Melody RP predictions were relatively simple; an alliance was predicted to achieve the Melody RP if the sum of their expected Notes was greater than or equal to 90% of the Notes threshold. This worked quite well for us throughout the season, predicting the correct outcome in ~80% of matches. We used the lowered Coopertition threshold as Coopertition was achieved in most matches.

Ensemble RP predictions were more complex. We used the two easiest-to-achieve Ensemble RP stage scenarios as a threshold: one trap + two climbs and harmony + third climb. From our Objective Scouts, we had individual climb, trap, and harmony success rates for each robot. Assuming the climbs and traps were independent events, we multiply the individual probabilities to get an alliance probability of achieving the RP. This is done for every possible permutation of each scenario, and a maximum probability is taken as the probability of achieving the RP. This method was quite accurate, predicting the correct outcome in ~80% of matches. It's important to note that our Melody RP prediction was *not* a probability, while the

Ensemble RP prediction *is* a probability. (This caused some misunderstanding in thresholds and displaying RPs in Viewer.)

This year, we used a new win-chance formula. In the 2023 Whitepaper, we used a logistic regression based on alliance score differences. This method calibrated itself too slowly and depended heavily on past data and estimations, affecting the prediction accuracy. Instead, this year, we took advantage of our standard deviation datapoints—for each score datapoint we collect, we also have a corresponding standard deviation datapoint. Assuming team scores behave according to a normal distribution (this is not ideal and is one reason to change our predictions; see last paragraph), we can then calculate a mean and standard deviation for a whole alliance's score distribution using the fact that the sum of two independent normal random variables is still a normal random variable. We then run a one-tailed t-test for both alliance's distributions in a match to determine the probability that one alliance wins. This method was remarkably more accurate than the logistic regression, predicting the correct match outcome 90% of the time. However, while the logistic regression is time-dependent, this new method requires more matches played to be accurate.

Special calculations are made for the playoffs—the predicted auto and tele scores for each alliance are calculated separately. In the case of four-team alliances, calculations are run for multiple permutations, such as the captain and first pick and one of the second or third picks.

Predicted Team calculations use each team's predicted RPs in each match to predict the final rankings at the end of qualifications.

During this upcoming offseason, we're completely overhauling our prediction formulas. We're still trying out new methods of predictions in hopes of achieving complete autonomy. In the past, we relied on specific formulas for each prediction—in 2023, Charge and Link RPs had individual formulas, while in 2024, formulas for Melody and Harmony RPs had to be constantly adjusted from competition to competition. Furthermore, a full update is required yearly, entailing more hours of brainstorming formulas and writing code. A new system based on linear and logistic regressions using past competition data was created in response to the constant changes. In this new system, weights for each action (e.g. speaker scores and onstage rates) are estimated by running a multiple linear or logistic regression on match scores vs. each

action. Linear regressions are used for score predictions, while logistic regressions are used for RP and win chance predictions. We ran into problems with this method before the 2024 World Championships caused by large amounts of robots that only ferried Notes, which we did not account for in the models. If successful, this new method will allow predictions to essentially run themselves; at the beginning of a season, we set certain actions as independent variables, and the action weights will self-update throughout the rest of the season. If this change is successful, hopefully, you'll see a more concise predictions section on the 2025 whitepaper!

Scout Precision

Scout Precision is a metric that calculates the number of points a Scout is off from the actual total. Scout Precision Ranking (SPR) compares Scout data against The Blue Alliance (TBA) data.

TBA only reports official scores for entire alliances, not individual teams, so Scout Precision calculations use the combined data of all Scouts on an alliance to find how far off a specific Scout is from their expected data. Since there are nine Scouts per alliance, with three on each robot, there are 27 combinations of three Scouts that will contain one Scout from each robot. For each of these combinations, the values of the datapoints for each Scout are totaled to get the overall alliance score. The official value for each datapoint is pulled from TBA and then compared against the total scouted score for the alliance to calculate the amount of error made by Scouts. A match's average error for a particular Scout is calculated by taking the average of all errors in all combinations.

Once the average match errors for each Scout in a match are calculated, the formula looks at each three-Scout combination that a specific Scout was in. The average errors of the other two Scouts in that combination are divided by 3 (since errors result from three-Scout combinations) and totaled to get the expected error of that combination. Then, the actual error of the combination, including the Scout in question, is subtracted from the expected error to find how much the Scout contributed to the error of that combination. The average of this value for all of a Scout's combinations in a specific match is that Scout's Scout-In-Match (SIM) Precision.

The average of a Scout's SIM Precision values for all matches in a competition is that Scout's overall Scout Precision Rating (SPR). The lower this value, the better since it represents how much average error a Scout contributed to their combinations. For a more detailed example of the Scout Precision Ranking calculation, see the SPR Calculation Walkthrough appendix.

We have considered utilizing SIM Precision in TIM consolidation. This could be done by subtracting each Scout's calculated error from the scouted value before consolidating. We have also considered using overall Scout Precision in Auto Paths calculations by favoring the most accurate Scout's timeline. However, we have not implemented these due to the long runtime of Scout Precision calculations and the possibility of not having TBA data due to the *FIRST* API not updating. (This was a rather serious issue during the Sacramento Regional, and we have now taken steps to run our entire system flawlessly without external data.)

This season, the formulas for Scout Precision largely stayed the same. Instead, we lowered the runtime of Scout Precision by 66%, allowing it to be run more frequently, as it no longer takes up most of a full Server loop's duration. Additionally, we can now exclude Scout Precision from our default Server loop. This is mainly due to the issues that arose when the *FIRST* API (and, therefore, TBA) did not update during the competition. We also added the ability to calculate SPR for specific datapoints. For example, we calculate individual SPR for speaker and amp Notes. These individual SPR metrics are used to see individual scouts' strengths and weaknesses. This year, we found that our scouts were particularly poor at recording amplified speaker Notes.

Interactions with TBA and Statbotics

We collect data from Statbotics and TBA through their respective APIs, each providing data in JSON format, making it easy to process programmatically. TBA's data is primarily used for detailed information about specific matches, which is crucial for calculating metrics such as scout precision, climbing vision, leave, and spotlight. Climbing vision, for example, indicates whether a team can climb onto a chain opposite their driver station (and thus, can't easily be seen). This is derivable because the order of teams TBA lists matches the driver stations' order.

On the other hand, the data we receive from Statbotics predominantly focuses on event-wide, match-specific, and seasonal statistics. A key metric from Statbotics is EPA (Expected Points Added), a moving average reflecting team performance. Unlike TBA data, which we actively process, the information from Statbotics, particularly the EPA values, is used directly without further calculations. Our strategy team then directly utilizes this pre-calculated data and our calculated data for match strategy and forming a picklist.

Testing and Code Standards

We write automated testing for most Server code using the pytest and unittest libraries. Each calculation and script file has a corresponding test file that uses fake data to test that a file is performing properly. Test files are generally structured so that every function in the original file has a corresponding test function that will fail if the output is incorrect. Anytime code changes, the corresponding tests must also be updated to ensure everything functions as intended. This form of automated testing provides several advantages. Previously, it was difficult to standardize the environment in which code was run, leading to code that would function on one developer's device but fail on another. Tests also serve as documentation for the main code, acting as an example of how it should run. Running tests with pytest is also convenient since it allows parts of the Server to run individually.

A GitHub Action automatically tests each pull request when it is opened and displays whether it succeeds or passes. This ensures that each pull request is functional before it is merged. Every pull request is also checked against Black, a Python code formatter. This allows the codebase to have standardized formatting even while different developers contribute with unique styles.

Pulling Data from Devices

Data collected by Scouts is formatted into a QR and scanned using QR scanners. The scanners then upload the data to the database. QRs can also be pulled from tablets connected to the server laptop via the `qr_input.py` file. Tablets can store QR data locally if QR scanning doesn't work. Furthermore, the Server laptop pulls images and JSONs collected in Pit Collection from plugged-in phones. This data is uploaded to the database as well. Processed

data is pushed to the Downloads folder of connected Viewer phones using the `send_viewer_images.py` script, allowing the Viewer app to see the images and data models.

QR data is stored in a compressed string format and later decompressed into a readable arrangement using the decompressor Schema. Compressing QR strings decreases the amount of space needed to store data, and Schema formatting for QR strings decreases the amount of stored data.

If a QR in the system needs to be removed or modified, it is blocklisted or overridden rather than deleting or altering the raw data. A QR may be blocklisted if a Scout inputted inaccurate data (ex., missed a portion of the match) or if a match gets replayed. Instead of being completely deleted from the database, a blocklisted QR is simply ignored in all calculations. Suppose a specific datapoint for a team in a given match needs to be modified. In that case, the QR can be overridden by flagging the datapoint's calculation to use a given value instead of the one from the QR code. After blocklisting or overriding a QR, calculations must be rerun to get accurate data.

Stand Strategist and Pit Data

Stand Strategist and Pit Data are crucial to our team's Picklist and give strategists very important qualitative insights on robot performance. Thus, it is necessary that we insert the data into MongoDB, where it can then be sent to the Viewer or Picklist as needed.

Since the Stand Strategist and Pit Collection apps don't export data as QR codes and are separate from Match Collection, we pull data from them manually using a USB cable and the Android Debug Bridge. The Server loop checks if any Pit Phones or Stand Strategist Tablets are connected, and then pulls data from specified folders on the devices. For robot images, we have a `send_viewer_images.py` script that sends JPEGs from the Pit Phones to the Viewer Phones.

Unlike data collected by Objective and Subjective Scouts, we do not perform calculations on Stand Strategist or Pit Data. Instead, we handle empty values and convert datapoints into suitable Python data types before uploading them to MongoDB.

Exporting Data

Even though all collected datapoints are viewable on the Viewer app, it is imperative to be capable of working with data on a spreadsheet during the competition itself. As a result, specific data is exported as CSV files to be easily used in a spreadsheet or another data visualization tool. Using a Python script, Team and Team-In-Match (TIM) data are exported together with data from TBA in CSV format, allowing strategists to work with the data in a spreadsheet whenever necessary.

The Picklist Editor receives data from CSV exports. After a competition, the data is exported to help analyze predicted calculations, data accuracy, scout accuracy, and other data. Although it is plausible to develop a system to automatically send exported data to spreadsheets or similar software, we found sending data manually via Slack easier because an actual system would require unnecessary effort and wouldn't provide any additional benefits.

Grosbeak

Grosbeak is a Python FastAPI web server providing reliable data transfer between our cloud database, MongoDB, and front-end apps. It was created during our 2022 competition season as a backend for our Live Picklist feature but has since replaced our old web server, Cardinal, which was built during the 2021 offseason.

In the 2022 offseason, we implemented serving data and static files (match schedule and team list) and converted our WebSocket-based Live Picklist into a simpler REST API. We also restructured the data being served to consolidate similar documents across collections, simplifying data fetching in Viewer. For example, a team's objective and subjective data would be combined into one document. We attempted to implement server-side caching of consolidated data to improve performance, but we ran into limitations with MongoDB.

At the beginning of each competition, we upload the competition's match schedule and team list to Grosbeak through a simple web interface. Viewer and Pit Collection can use endpoints in Grosbeak to retrieve the match schedule and team list for a competition.

To retrieve data, the Viewer makes requests to an HTTP endpoint that serves all necessary data. Grosbeak reads data from MongoDB, processes it to consolidate documents, and returns it to the Viewer.

This season, we implemented a set of Stand Strategist endpoints in Grosbeak, enabling Stand Strategist to transfer collected data to and from MongoDB without having to process data through the Server. However, we did not use this feature as we did not acquire data plans for the tablets on which Stand Strategist is run.

The Live Picklist feature in Viewer relies on Grosbeak to synchronize the picklist between devices. However, we did not use the Live Picklist feature this season since it was unnecessary and problematic.

Picklist Editor

Overview

Picklist Editor is a tool developed to construct an informed picklist at competition using our scouting data. The app depicts the competition's team list and statistics for each robot, with its main feature being the ability to reorder teams. Picklist Editor runs on Google Sheets using the Google Apps Script platform, and the code is written in TypeScript. All features except team rank ordering, removing teams, the last four matches checkbox, and some Final Picklist capabilities are written with Google Sheets formulas.

Team Rank Ordering

In the main editor, the first column shows a list of the teams in the competition, initially ordered by their 'pickability' ratings (pickability ratings are based on either first or second pick rank

scores; for more information on how pickability is calculated, see Pickability in Server). Ranks are displayed in the second column.

Each row displays the data for its respective team. The data is retrieved using VL00KUP formulas from a separate raw data sheet exported from the MongoDB database by our Server at the end of the competition day.

Some additional datapoints our strategists collect are manually inputted throughout the first day of matches. These datapoints include how robust a robot is mechanically and electrically and are collected subjectively through examinations in the pits and matches.

To reorder the teams, the picklist operator edits the ranking number (e.g., to move a team between first and second, change their rank to 1.5). Then, the scripts behind the Picklist Editor will automatically reorder the teams and update their ranks to whole numbers. The spreadsheet is designed for bubble sorting, starting from the top and working downwards.

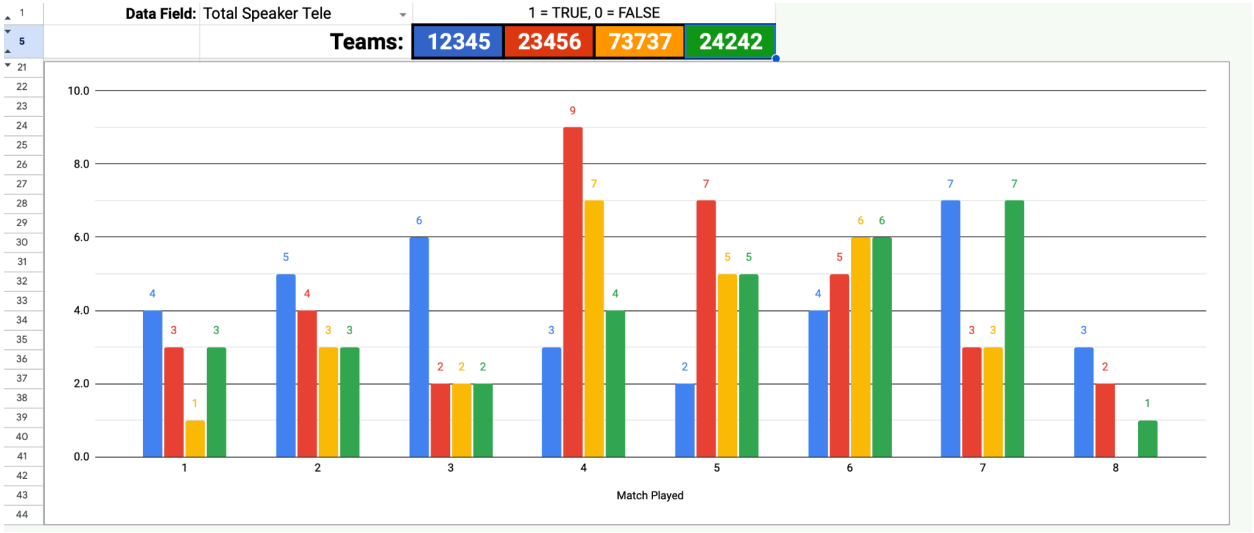
At the top left corner, a checkbox exists to change the displayed datapoints to show data for the last four matches played. These datapoints are used to rank teams based on their most recent performance. Unchecking it will revert the datapoints.

	A	B	C	D	E	F	G	H	L	M	N	O	P	Q	R	S	T
3	☐	Rank	DNP	Electrical Robustness	Mechanical Robustness	1st	2nd	Which Strategy	Auto Avg Speaker	Auto Max Notes	Compatible Spike Auto	Tele Avg Amp	Tele Avg Speaker	Avg Intakes	Avg full field Cycle Time	Tele Max Amp	Avg Exp Field
4	12345	11	1.0	4.0	4.0	105.2	66.7	Ferrying	3.9	5.0	TRUE	4.1	8.0	19.7	11.8	9.0	11.1
5	23425	13	2.0	4.0	5.0	80.7	48.1	Ferrying	1.1	3.0	FALSE	3.6	5.7	16.2	22.1	10.0	9.4
6	23466	5	2.0	3.0	3.0	62.7	43.8	Scoring	2.3	4.0	TRUE	4.2	4.2	11.9	37.6	10.0	7.7
7	45474	9	2.0	4.0	4.0	70.0	38.2	Scoring	2.0	3.0	FALSE	4.5	5.4	13.1	21.6	8.0	8.0
8	90943	3	2.0	4.0	4.0	84.2	50.4	Ferrying	1.6	3.0	FALSE	3.3	3.8	14.4	16.9	6.0	7.6
9	34134	6	2.0	3.0	4.0	92.9	64.4	Ferrying	2.5	4.0	TRUE	2.7	4.9	17.0	12.0	9.0	10.3
10	87878	7		4.0	3.0	43.9	36.9	Scoring	1.3	2.0	TRUE	3.9	3.3	9.1	22.2	6.0	7.0
11	23465	8		3.0	4.0	48.4	23.5	Scoring	1.6	3.0	FALSE	2.7	4.3	10.3	69.1	7.0	5.7
12	19393	4		3.0	3.0	51.6	30.5	Scoring	1.5	3.0	TRUE	3.1	3.9	10.6	43.3	7.0	6.2
13	24242	10	4.0	3.0	4.0	112.7	90.6	Ferrying	2.0	3.0	TRUE	3.6	4.9	16.9	14.0	9.0	10.3
14	91919	1	4.0	3.0	4.0	89.8	76.8	Ferrying	2.1	4.0	TRUE	1.1	2.9	16.5	13.1	5.0	9.9
15	34343	12	3.0	4.0	3.0	86.3	62.5	Ferrying	1.5	3.0	FALSE	1.1	4.4	16.3	13.6	6.0	9.7
16	24222	2	2.0	4.0	4.0	82.4	59.0	Ferrying	2.6	5.0	TRUE	2.5	3.2	12.8	20.1	6.0	7.4
17	11010	14	1.0	3.0	3.0	92.4	70.3	Ferrying	1.5	5.0	TRUE	3.0	3.0	14.6	14.7	7.0	9.3
18	33333	15		2.0	4.0	99.2	84.7	Ferrying	1.5	2.0	TRUE	3.1	3.8	16.9	13.1	8.0	9.9

Main Editor

Team Performance Comparison Graphs

When a match-by-match comparison between teams is necessary, the Picklist Editor has a graphing feature that compares the data of up to four teams in a single datapoint. The graphs can also show changes in data over multiple matches.



Team Comparison Graphs

Removing Teams

To limit the teams to rank and save time, Strategists determine teams that aren't performing at the level they'd like for our alliance or may not be a good fit for our preferred strategy and remove them from the picklist at the beginning of the meeting. Before the picklist meeting, these teams are discussed between the Match Strategist, Stands Strategists, and strategy mentors and given ratings in a column adjacent to the rankings column. A team is removed from the list on the main page by typing "DNP" or "d" in their rank cell. If the team is later reconsidered, the operator can send the team back by checking its box on the DNPs page.

	Teams	Send back (click)
2	12341	☐
3	23423	☐
4	23424	☐
5	95959	☐
6	10909	☐
7	44444	☐

DNP Page

Updating Datapoints

The datapoints shown in the editor and their order can change drastically throughout the season based on feedback from students and mentors in picklist meetings. Due to the way that the Picklist Editor is set up, it was easy to add or remove datapoints in the middle of picklist meetings, although strategists did attempt to determine all datapoints beforehand to save time. Datapoints can be added by inserting a new column, copying over the formulas, and entering the datapoint name in the second cell of the column.

Operation

Before 1678's picklist meetings, the picklist operator is sent raw data in CSV format from the Server. During picklist meetings, the picklist operator displays the Picklist Editor through a projector.

Then, starting with the first pick order from the ranking equations, the teams are reviewed beginning at the top and compared on a pairwise basis, progressing down the list. At the picklist meeting, attendees decide for each team whether it should be moved up the picklist and by how many places. Once the potential first picks (usually down to 10 or 12 teams) are ranked, the rows containing those teams are hidden, and the potential second picks are ranked similarly.

At a regional, where matches continue on the second day, strategists can make further changes to an additional sheet called the Final Picklist. This sheet is created after the initial

picklist has been finalized and also contains the functionality to reorder or remove teams in case certain teams' ranks need to be changed. Some additional functions in this sheet include the ability to remove but not DNP teams, add teams not included in the initial picklist to it, and a column that tracks the difference in rank compared to the initial picklist.

The purpose of the Final Picklist is for strategists to make further changes to the picklist based on further information and a closer review of specific teams by the Stand Strategist and other key strategy team members.

	Rank	Delta	DNP	Electrical Robustness	Mechanical Robustness	1st	2nd	Which Strategy	Auto Avg Speaker	Auto Max Notes	Compatible Spike Auto	Tele Avg Amp	Tele Avg Speaker	Avg Intakes	Avg full field Cycle Time	Tele Max Amp
12345	10	-1.0	1.0	4.0	4.0	105.2	66.7	Ferrying	3.9	5.0	TRUE	4.1	8.0	19.7	11.8	9.0
23425	14	1.0	2.0	4.0	5.0	80.7	48.1	Ferrying	1.1	3.0	FALSE	3.6	5.7	16.2	22.1	10.0
23466	4	-1.0	2.0	3.0	3.0	62.7	43.8	Scoring	2.3	4.0	TRUE	4.2	4.2	11.9	37.6	10.0
45474	9		2.0	4.0	4.0	70.0	38.2	Scoring	2.0	3.0	FALSE	4.5	5.4	13.1	21.6	8.0
90943	3		2.0	4.0	4.0	84.2	50.4	Ferrying	1.6	3.0	FALSE	3.3	3.8	14.4	16.9	6.0
34134	6		2.0	3.0	4.0	92.9	64.4	Ferrying	2.5	4.0	TRUE	2.7	4.9	17.0	12.0	9.0
87878	7			4.0	3.0	43.9	36.9	Scoring	1.3	2.0	TRUE	3.9	3.3	9.1	22.2	6.0
23465	8			3.0	4.0	48.4	23.5	Scoring	1.6	3.0	FALSE	2.7	4.3	10.3	69.1	7.0
19393	5	1.0		3.0	3.0	51.6	30.5	Scoring	1.5	3.0	TRUE	3.1	3.9	10.6	43.3	7.0
24242	16	6.0	4.0	3.0	4.0	112.7	90.6	Ferrying	2.0	3.0	TRUE	3.6	4.9	16.9	14.0	9.0
91919	1		4.0	3.0	4.0	89.8	76.8	Ferrying	2.1	4.0	TRUE	1.1	2.9	16.5	13.1	5.0
34343	12		3.0	4.0	3.0	86.3	62.5	Ferrying	1.5	3.0	FALSE	1.1	4.4	16.3	13.6	6.0
24222	2		2.0	4.0	4.0	82.4	59.0	Ferrying	2.6	5.0	TRUE	2.5	3.2	12.8	20.1	6.0
11010	13	-1.0	1.0	3.0	3.0	92.4	70.3	Ferrying	1.5	5.0	TRUE	3.0	3.0	14.6	14.7	7.0
33333	15			2.0	4.0	99.2	84.7	Ferrying	1.5	2.0	TRUE	3.1	3.8	16.9	13.1	8.0

Final Picklist

Google Apps Script

The Picklist Editor is built on Google Apps Script, similar to JavaScript. Using the official command-line interface Clasp, it is possible to clone the scripts into a local development environment as JavaScript files. We use the TypeScript programming language, a superset of JavaScript providing static type checking and an improved developer experience, with Clasp to develop the Picklist Editor.

Robot Photos

Many strategy team members find that viewing pictures of a robot helps prompt other observations about the robot and its team's performance. However, due to performance issues from previous years, we decided this year that showing robot photos in the sidebar of the Picklist Editor interface was not worth it. Throughout this season, we decided to have a separate spreadsheet to display photos, which has proven much more effective.

Video System

The Video System is a vital part of data collection and strategization, allowing strategists to review and rewatch matches, which helps them create match strategies for upcoming matches, provide feedback to our drive team, and rewatch a team's performance in a specific match during the picklist meeting. We implemented this system due to inconsistency in official match videos' upload times and camera angles.

Video System Operators take videos of all qualification and elimination matches, doubling as Objective Scouts. They start by setting up the tripod and camera in an elevated and centered position overlooking the field. After each match, they rename the video and copy it from the SD card to a hard drive. They also move select match videos to USB drives to review between Qualification matches. During elimination matches, videos are put on a USB flash drive and given to match strategists, enabling them to develop strategies for the next match quickly.

Conclusion

Season Recap

This year, our scouting system was at its best. We added feature after feature throughout the season, expanding upon the system architecture we have been iterating on for years. Our scouting system was crucial in selecting competition-winning picks and generating winning

match strategies. By continuing to build a robust and effective scouting system year after year, we ensure success at every competition we attend.

Lessons Learned

Training

Once again, training was lacking this year. A lack of proper training in both our Front-End and Back-End teams hindered our start to the build season. New members had trouble understanding MongoDB and Python's `unittest.mock` class.

This season, we learned how critical proper training is to a software development team's success. Our new training presentations on Python and Kotlin and the Mini-Scout training activity for our new Front-End members allowed our new team members to learn the skills required to work on our apps efficiently. Next year, we plan to conduct training on basic statistics along with our standard code training.

Documentation

The importance of documentation has only been more apparent this season. The lack of documentation for subteam processes and procedures was worrying and caused unnecessary problems early in the build season. If certain people were missing during team meetings, it would become a huge issue when their knowledge was needed. Other people would be unable to work or test without the presence of someone else. Proper documentation would have eliminated these types of situations and improved our workflow.

Data Plans

Unfortunately, at Champs this year, the phone data plans were highly inconsistent and unreliable after working smoothly at all our previous competitions. Heavy traffic at Champs and miscommunication regarding data usage resulted in significant issues, making our Viewer app unusable. In the future, we plan on getting a faster and larger data plan and creating backup plans for what to do if the data plans stop working again.

Starting a 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 a Google Forms scouting system at off-season events 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 then discuss with one another: What would you have done differently? What team did well? What were some minor mistakes? Who had the best speed?

If you can spare only about two members of your team for scouting, then we recommend either having each one take notes about the ability of one alliance (training is critical when you are recording qualitative notes) or you can reach out to fellow teams and 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 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 given your resource level, or have any other questions, please contact us at softwarescouting@citruscircuits.org.

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. In addition, we plan on improving our new member training by creating more hands-on assignments and training on statistics, MongoDB, and Tableau.

There are many new features to be implemented, code to be cleaned, and processes to be streamlined. There is no "perfect system," and we will continue to work hard to improve it.

Resources

Old Whitepapers

Our old whitepapers can be found on our team website.

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 can be found on our team website.

2024 Public GitHub Repositories

Match Collection: https://github.com/frc1678/match-collection-2024-public

Viewer: https://github.com/frc1678/viewer-2024-public

Server: https://github.com/frc1678/server-2024-public

Schema: https://github.com/frc1678/schema-2024-public

Pit Collection: https://github.com/frc1678/pit-collection-2024-public

Stand Strategist: https://github.com/frc1678/stand-strategist-2024-public

OverRate: https://github.com/frc1678/overrate-2024-public

Picklist Editor: https://github.com/frc1678/picklist-editor-2024-public

Appendices

Subteam Structure

Students in Software Scouting are split into either Back-End or Front-End. Each end has its student lead, who is 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, including Match Collection, Pit Collection, Stand Strategist, Viewer, and Picklist. Each end requires members to specialize in specific programming knowledge and concepts. However, Software Scouting is still considered a single subteam, and students on both ends regularly review each other's code and participate in full subteam discussions and system tests. Some students also occasionally write code for the opposite end. Veteran members guide newer members, work with newer members on tasks, and teach new members concepts. The Software Scouting subteam works closely with the Strategy subteam as we formulate our strategies using scouting data, and most Scouting members are on the Strategy subteam.

How to Run a 1678 System Field Test

Before the test, collect match videos that are ideally high-scoring matches from a previous regional/district competition. If the competition season hasn't started yet, use Week 0 videos or screen recordings of the xRC Simulator.

Make a new Server branch to merge in any hotfixes that may need to be made during the field test. Name it with the date of the field test. Pull any unmerged PRs that need to be tested onto this branch.

Decide on a TBA event key to use (e.g., 2024cada for the 2024 Sacramento Regional). This will ideally match the event used for the videos, but it doesn't have to—it can be an event 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.

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

Recruit volunteers to use the Scout and Super Scout apps to collect data from match videos. If fewer than 20 students are available, have people use multiple tablets simultaneously. The main concern is not getting accurate data but thoroughly using every feature of the collection apps. The match videos are only there as a guide, so it's okay if the team number assigned to a Scout differs from the robot they are scouting. Additionally, have a user enter test data using the Pit Collection app.

After each match, scan the data into the Server. Make sure data is being entered into the local and cloud databases and that the web server is able to send it to the Viewer. Monitor the Viewer to ensure data is updating and being displayed correctly.

Write down every bug or suggestion for improvement as it comes up, no matter how small. Afterward, go through the list and prioritize which ones to address first.

Training

During the offseason, the first months are spent training the new members. New members start by working with all software subteams (Robot, Front-End, and Back-End) to learn the basics about what each subteam does and to get to know each other before selecting which one they'd like to be a part of. After new members are separated into their subteams, End-specific training begins. Front-End training starts with basic slideshows covering coding basics in Kotlin, including variables, functions, for loops, classes, etc. Each lesson has assignments that are hands-on challenges the members must complete by utilizing the skills that they learn throughout the training. Once basic syntax training has finished, Front-End new members begin on their last training project: the Mini Scout. This was a training method developed in 2021 where new members create their miniature version of our Match Collection app following instructions that get less specific as they progress through the project. The Mini Scout has members create a match starting screen, data input screen, and match data edit screen. Back-End training begins with the basics of Python, with lessons being organized through slideshows and assignments. Once syntax training is complete, Back-End new members then

learn about our Schema and the MongoDB database. Additionally, Back-End members are walked through the Server and how data gets transferred and organized. General standards and tips are taught together to the entire subteam. These lessons include training on GitHub, creating pull requests to push their code, reviewing standards, and general system overviews.

Scout Training and Management

The week before each competition, Scouts are trained to collect accurate data using the app. Objective Scouts are trained by a Scout Lead and Assistant Lead, who are in charge of organizing Scouts during competitions. The training begins with an explanation of behavior standards at the competition and an overview of the itinerary. Afterward, Objective Scouts participate in a Kahoot, quizzing Scouts on what to do in certain situations while scouting to ensure that we get consistent data. Scouts also spend at least 2 hours practicing scouting while watching matches from this 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, when scouts become more confident with their rankings, they collect data independently with no input from the mentors and get feedback after inputting 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 the knowledge is passed down between users.

21 Objective and 3 Subjective Scouts are brought to each competition. Breaks are set by the Scout Lead ahead of time, and we try to give each Objective Scout 30-minute breaks at least once a day. Scouts can go to the pits during this time, meet new teams, and explore the venue. Subjective Scouts organize their breaks. Each Scout has a Scout ID that is used to ensure that three people are scouting each team. This Scout ID is preassigned to Scouts but changes when breaks switch off. A Scout count-off where each Scout says their ID is often implemented to ensure we have all 18 active scouts.

SPR Calculation Walkthrough

Consider a Scout named X. To calculate X's SPR, the formula starts with a single match that X scouted. It finds two other Scouts in that match, one scouting each of the other teams in the alliance. Now, it adds the scout-reported scores from the three Scouts together to calculate a theoretical alliance score and compares it against The Blue Alliance's official alliance score to get that combination's error. For example, in a combination where X said Robot 1 scored 12 points, Y said Robot 2 scored 31 points, and Z said Robot 3 scored 10 points, and TBA reports that the entire alliance scored 50 points, that combination's error is 3 points.

Then, the formula takes the error of another combination with X and two different Scouts on the other two teams. This process repeats until it has gone through all the possible combinations containing X and the other two teams that X didn't scout. The average error of all these combinations is X's average combination error in that match. The process is repeated for all Scouts in that match to find their average combination errors.

Then, the formula returns to look at each combination that X was in. For each combination, it finds the average combination errors of the other two Scouts and divides each by 3. It sums the two errors to get the expected error of that combination. For example, in a combination with X, Y, and Z, where Y has an average error of 15 and Z has an average error of 4, it would find the expected error of that combination to be 6.33.

Then, it subtracts the error from the specific combination from the expected error. If X had been completely accurate, the error from that single combination should be similar to the expected error, so the result should be close to 0. If X had been off, they would have contributed further error to the combination error, so the result would be less than or greater than 0 depending on how far above or under they were. If the error of the XYZ combination was 6.33, then the formula would determine that X did not contribute any extra error on top of the 6.33 that was already contributed by Y and Z. If the error of the XYZ combination was 7, then the formula would determine that X contributed approximately 0.66 extra points in error.

This process is repeated for every Scout in that match, for all matches in a tournament. Each Scout's average score across all their matches is averaged to calculate their overall Scout precision.

Picklist Editor Scripts

These are the main Google Sheets formulas that exist in the Picklist Editor. The formulas shown are examples and may reappear many times throughout the spreadsheet.

Formula to get which second pickability strategy a team is playing. This is responsible for selecting either ferrying, scoring, or defense from the columns AM, AL, and AK, depending on whether it equals the value in H23. In this example, columns AM, AL, and AK contain distinct second pickability ratings that we calculated, and the value in H23 is the maximum value of AM, AL, and AK.

```
=SWITCH(H23, AM23, "Ferrying" , AL23, "Scoring" , AK23, "Defense" , "Defense")
```

Formula to look up the column index of a datapoint name, given in the cell M2 in this example.

```
=MATCH(M$2, 'Team Raw Data'!$A$1:$$1 ,0)
```

Formula to look up data from raw data given cell A21, which corresponds to a team number, and N1, which corresponds to the column index found by the Match formula above.

```
=VLOOKUP($A21, 'Team Raw Data'!$A$1:$ZZ$99, N$1, FALSE)
```

Formula to find change in rank in the Final Picklist, B5 corresponds to the rank displayed in the Final Picklist and is what strategists change during the second day of competition, and the Index function finds the initial rank from cell A5 of the Main Editor. If the team has been put on our do not pick list, their rank will show as "d."

```
=IFNA(-$B5+INDEX('Main Editor'!B:B,MATCH(A5,'Main Editor'!A:A,0),0), "d")
```

Codebook

Our codebook contains every datapoint we collect, grouped into the respective “collections” they belong to (e.g. “Objective Team” or “Pickability”). This year, we decided to move our codebook into a Google Sheet in order to avoid cluttering the Whitepaper document. The spreadsheet contains each datapoint name, its Python data type, and a short description of the datapoint. In total, we collected 557 datapoints this year.

Link to Codebook:
https://docs.google.com/spreadsheets/d/1BFb8P5c_2-tTHnJa8F47yu4t2fAVcXeSRaD17wcFTfQ/edit?usp=sharing

Season Timeline

Before Kickoff — All Veteran Software Scouting members train new members and 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.

1/6/2024 — 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.

1/7/2024 — The Scouting subteam meets with Strategy members to determine what datapoints are necessary to collect and which we want to display. For each datapoint on the final list, the following information is noted down:

- The data type
- A description of what the datapoint represents
- Some example values
- Which database collection would it be stored in
- Whether it would be collected raw or if it needed to be calculated from other datapoints
    - If it was a raw datapoint, how would it be collected (by Scouts, Super Scouts, Pit Scouts, Stands Strategists, or The Blue Alliance API)

- If it would be displayed in the Viewer, and if so, how it would be visualized

1/7/2024 to 1/13/2024 — Back-End students update the Schema files to contain the new datapoints for the new season. Front-End students update the Match Collection and Pit Collection apps with new data and UI designs.

1/13/2024 to 1/30/2024 — 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 to update the Viewer and Stand Strategist to add new features and datapoints. The visual apps are also often worked on simultaneously with the collection apps to prevent conflicts between code merges.

1/30/2024 to 2/27/2024 — Software Scouting begins conducting end-to-end field tests to ensure the system runs together. Front-End collects user feedback on the apps' UI. Based on strategy discussions, the list of datapoints to calculate/collect is updated. Students watch xRC Simulator matches to test the scouting system before real match videos are available. Scouts and Super Scouts are trained on how to use the Match Collection app as competition season gets closer.

2/27/2024 to 4/23/2024 — During competition season, the Saturday before each competition is a feature freeze, a deadline that stops development on all new features until after the competition. The subteam runs a full-system test before every competition to catch and fix last-minute bugs. On the first meeting after returning from each competition, the full subteam participates in a debrief and communicates with users and mentors to prioritize which changes to make before the next competition.

4/23/2024 to 5/22/2024 — Members work on the Whitepaper and code cleanup to prepare for public release.

Competition Roles

The roles in competition are divided into Developers, Scouts, Operators, and Strategists. These are all student roles, with the exception of mentors assisting Stand Strategists, but even then, the students are the primary voices.

Developers — The developers consist of one Front-End developer and one Back-End developer. The developers fix bugs as they arise and assist with Scout ID assignments and handing out tablets. The Back-End developer is also in charge of uploading scanned data to the MongoDB cloud database 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 of the logistics for the Scouts, including meals, shifts, handing out tablets, and anything else they might need. The Lead Scout does not scout, so they may 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 is an objective scout.

Scouts — Since 18 Objective Scouts are needed per match, about 21 Scouts travel to each competition to give three Scouts a break at a time. In addition, there are three Subjective Scouts so that one can be on break while the other two scout. Since Super 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 who's usually on Software Scouting and has experience in programming Picklist operates the spreadsheet during picklist meetings. The Picklist Operator is responsible for updating and maintaining the picklist spreadsheet throughout the season.

Video System Operators — Two Scouting or Business and Media subteam members 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. Stand Strategists edit the picklist at regionals depending 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 on the practice day and works with the drive team and Strategists throughout the competition. Our Strategists are often veteran members of the Strategy subteam and have a keen eye for strategy.

Super/Subjective Scouts — Three Subjective Scouts collect qualitative data during matches, such as robot speed, driver awareness, and maneuverability. They use a separate 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.

Hardware

This Scouting System requires multiple different 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 Pit Collection and the Viewer app. For data transfer, we use two QR scanners that transfer data from the tablets directly to a single laptop that runs the Server. All hardware is packed into five cases: Two tablet cases, a Server case, a Video System case, and a Gray case for extra space. More specific details about the cases and our hardware can be found in section 4.2 of our 2020 Whitepaper.