

2023

SCOUTING WHITEPAPER

TABLE OF CONTENTS

Table of Contents.....	2
Introduction.....	7
System Overview.....	7
History.....	8
Match Collection.....	9
Objective Collection.....	10
Information Input Screen.....	10
Randomizing Scout IDs.....	11
Starting Position Screen.....	11
Collection Screen.....	12
Switching Intake and Scoring Buttons.....	12
Charge Station Pop-up.....	14
Subjective Collection.....	15
Information Input Screen.....	15
Starting Screen.....	16
Collection Screen.....	16
Screens Reused in Objective & Subjective Modes.....	18
Edit Information Screen.....	18
QR Schema Format.....	19
Playoff Scouting.....	19
Pit Collection.....	20
Data Collection.....	20
Team List Screen.....	22
Starring Teams.....	22
Flagging Teams.....	22



Automatic Highlighting to Show Scouting Progress.....	23
Editing Event Key.....	24
Pit Map Screen.....	25
Stand Strategist.....	26
Overview.....	26
Navigation.....	26
Selecting Match Schedule.....	27
Navigation Flow.....	27
Match Information Screen.....	27
Match Data & Team Notes Screens.....	28
All Team Notes Screen.....	29
Match Selection.....	29
New Match Screen.....	30
Exporting Data.....	30
In-season Changes.....	30
Server.....	31
Schema.....	31
Collecting Charge Station Data.....	31
Device Data Pulling.....	31
QR Handling.....	32
CSV Exporting.....	32
Auto Paths.....	33
Predicted Calculations.....	33
Pickability.....	34
Scout Precision Ranking.....	35
Grosbeak.....	37
Viewer.....	38



Navigation.....	38
User Preferences.....	39
Match Schedule.....	40
Match Details.....	41
Team List.....	42
Team Details.....	43
Team Notes.....	44
Data Graphs for Specific Data Points.....	45
Rankings for Specific Data Points.....	46
Last Four Matches.....	47
Robot Images.....	48
Auto Paths.....	49
Field Map.....	50
Pickability.....	51
Picklist.....	52
Elim Alliances.....	54
Data Refreshing.....	54
Picklist Editor.....	55
Overview.....	55
Team Rank Ordering.....	55
Updating Data Points.....	56
Removing Teams.....	56
Team Comparison Graphs.....	56
Operation.....	57
Google Apps Script.....	57
Robot Photos.....	57
Video System.....	58



Conclusion.....	59
System Accomplishments.....	59
Summary of Major Changes Since 2022.....	59
Lessons We Learned.....	59
Training.....	59
Field Testing.....	60
Documentation.....	60
Data Plans.....	60
Starting a Scouting System.....	60
Future Steps.....	61
Resources.....	61
Past Whitepapers.....	61
Fall Workshops.....	61
GitHub Repos.....	61
Contact Info.....	62
Appendices.....	63
Subteam Processes.....	63
Subteam Structure.....	63
How to Run a 1678 System Test.....	63
Training.....	64
Scout Training & Management.....	64
SPR Calculation Walkthrough.....	65
Season Analysis.....	66
Timeline.....	66
Competition Roles.....	67
Other Information.....	68
Hardware.....	68



Pickability Weights.....	68
Codebook.....	69



INTRODUCTION

System Overview

The Citrus Circuits scouting system is broken up into three main stages: collection, processing, and visualization. Collection consists of the Match Collection app (with its three modes), 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. A subteam of Citrus Circuits students is dedicated to creating this system (see Section 3 of our [Team Handbook](#) for more on subteams).

This year, the users of the three collection apps were as follows:

- Match Collection (Objective): Objective Scouts
- Match Collection (Subjective): Subjective Scouts
- Match Collection (Playoff Scouting): Objective Scouts
- Stand Strategist: Stand Strategists
- Pit Collection: Pit Scout/Match Strategist (typically the same student)

Specific details of each role can be found in Section 6.2 of our [Team Handbook](#).

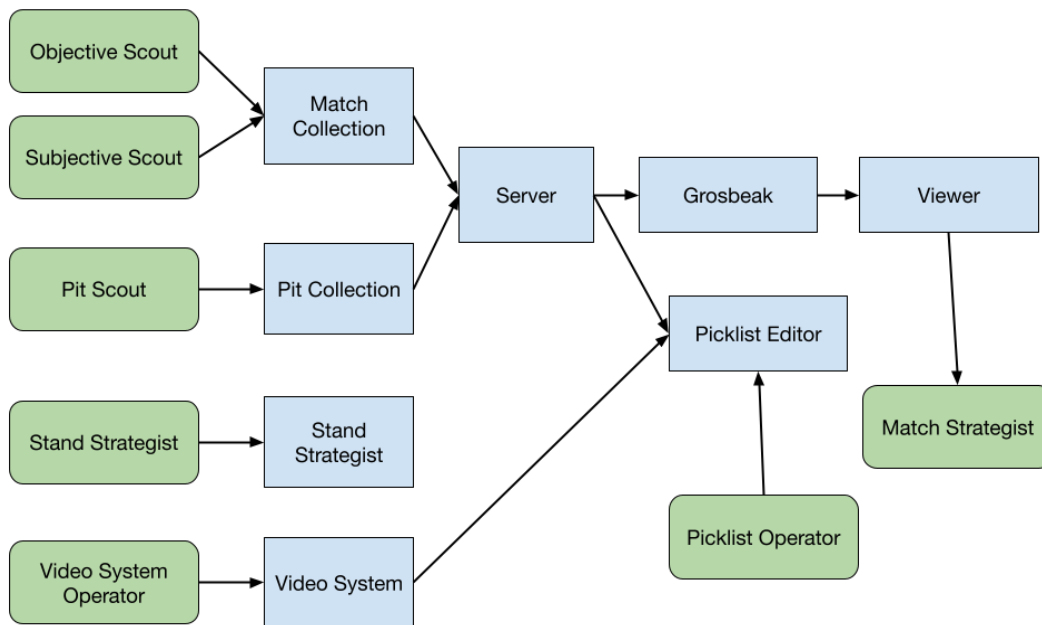


Diagram of the Scouting System (users are green, software is blue)

History

Citrus Circuits' electronic scouting system was first developed and used during the 2013 FRC season of Ultimate Ascent. The team had previously utilized a paper scouting system, but was overwhelmed by the management issues that arose when scouting hundreds of teams at the World Championship. A software development team of four students created the original system that has since grown into a full subteam of 18 students with multiple apps.

The system's original purposes have always remained consistent: (1) the collection of objective scoring for each individual robot and subjective ordinal rankings of robots' abilities within each alliance; (2) the processing and delivery of this data to the drive team for match preparation; (3) the combination of this data to create two ranked draft lists for alliance selections.

The system debuted at the 2013 Central Valley Regional, running with eight scouts and two programmers—it enabled 1678 to assemble an alliance based on scouting data and win as the 6th seed. At the World Championship, 1678 won the Curie Division.

In 2014, the system changed from wired data transfer to Bluetooth transfer. We also considered cellular data, but could not afford to purchase eight tablet data plans.

No major changes to the system were documented in 2015.

In 2016, an app designed for pit data collection was added to the system.

In 2017, three Objective Scouts (as opposed to one) were assigned to each robot to improve objective data accuracy and provide more opportunities for team members at competition.

In 2018, Bluetooth became less reliable with the prevalence of smartphones at events, and QR code communication was added for match data collection.

No major changes to the system were documented in 2019.

In 2020, the Match Collection, Pit Collection, and Viewer apps were rewritten in Kotlin for ease of development, and iOS support was discontinued in favor of Android. The apps have relied on several online database programs, initially adopting Firebase and then moving to MongoDB in 2020. At the first competition in 2020, the system was unable to deliver data due to both poor work distribution on and technical knowledge of the visualization apps. The COVID-19 pandemic ended the season before it could be updated further.

In 2021, the subteam met remotely and made improvements to the structure of the 2020 system without the need to focus on a specific game. The 2022 system was a result of substantial improvements since the Kotlin rewrite in 2020—the revisions set the foundation for future features. We based the current 2023 system off 2022's, and have since implemented new features to the app Viewer and created a new Stand Strategist app.



MATCH COLLECTION

The Match Collection app is used by scouts in the stands during matches to collect both quantitative and qualitative data on robots. The Match Collection app runs on Lenovo Android tablets and was developed with Android Studio using Kotlin and XML. In the Match Collection app, there are three modes: Objective Collection, Subjective Collection, and Playoff Scouting.

- Objective Collection is used to collect quantitative data such as intake location, climb level, and pieces scored.
 - Used during qualification matches; 3 Objective Scouts per robot (18 total)
- Subjective Collection is used to rank the performance of each robot on an alliance (for example, a ranking based on the Quickness of all the robots on the Red alliance).
 - Used during qualification matches; 1 Subjective Scout per alliance (2 total)
- Playoff Scouting allows scouts to record the locations of cone and cube placements and whether nodes are Supercharged.
 - Used during elimination matches; 1 Objective Scout per robot (6 total)

Starting Screen




Mode Selection Screen

The app opens to the Mode Selection Screen, allowing the user to choose the appropriate scouting mode by clicking on the corresponding button.

Objective Collection

Information Input Screen

 	Match Number 129
Automatic Assignment	Team Number 2630
Backup 1	
Scout ID: 8	
Old QRs	Proceed
Version: 1.0.1	

Objective Information Input Screen

The Information Input Screen has multiple elements.

Assignment Mode: There are two assignment modes—Automatic Assignment and Override Assignment. Automatic Assignment pre-sets the team and the alliance color based on a match schedule file downloaded to the tablet. Override Assignment allows the scout to change the alliance color or the team number they are scouting—it is used in case something goes wrong in the automatic assignments and during testing.

Old QRs: All past QR codes are saved onto the tablet and can be displayed in case the scout forgets to scan after a match.

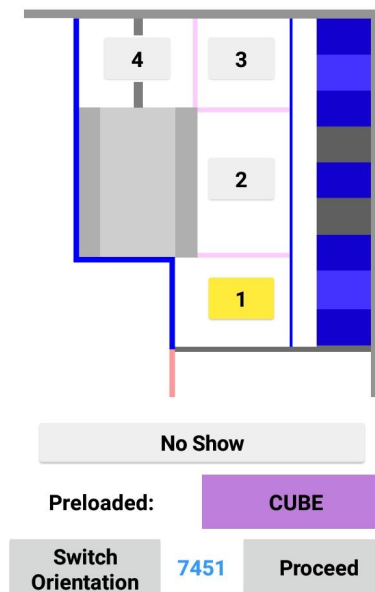
Scout ID: Scout IDs are used for assigning the scout a randomized team number while ensuring that each robot is scouted three times per match. Only scouting developers who run the scouting system during competition are permitted to change Scout IDs.

Other information: Scouts can edit the match number and change their scout name. The alliance color cannot be changed (unless on Override Assignment) but is used in the code to set text color where relevant. The Proceed button continues to the next screen (Objective Collection Starting Position Screen).

Randomizing Scout IDs

In previous years, each Scout ID corresponded with a specific driver station position. We began randomizing Scout ID assignments last year—essentially, if an Objective Scout is assigned to watch the robot in the Red 3 position in one match, they won't necessarily watch Red 3 again the next. This ensures that no group of three Objective Scouts consistently all scout the same robot, which helps us get better comparisons between different scouts (see [Scout Precision Ranking](#) in Server) and allows us to better compare our data with The Blue Alliance. These random assignments come from a file resource with 100+ randomized orders of Scout IDs (generated ahead of the competition—randomization is not done in real-time).

Starting Position Screen

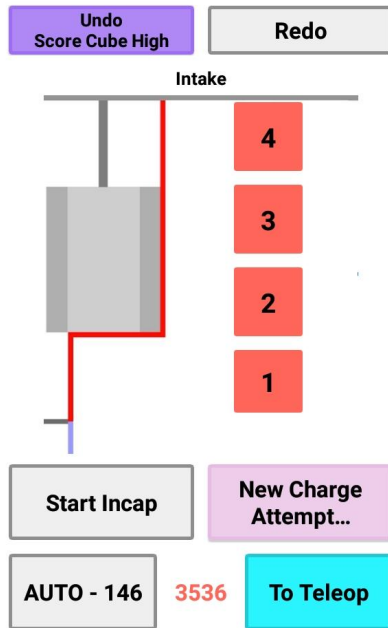


Objective Starting Position Screen (Blue Alliance)

In the Starting Position Screen, Objective Scouts input their robot's starting position as one of four areas we designated in the community zone. The preload button enables scouts to record their robot's preloaded game piece. The data collected in this screen (in addition to scoring locations, intake locations, and timestamps—all collected during the Autonomous period) allow our Server to calculate the auto paths of any given team.

Collection Screen

Intaking



Auto Intakes



Teleop Intakes

Auto Intakes: The intake screen is a diagram of the alliance's side of the field with four buttons, one where each game piece is placed before the match. Collecting the location of the intaken game piece helped us track the auto paths teams had.

Teleop Intakes: The diagram of the field is replaced with three buttons—Ground Intake, Shelf Intake, and Row Intake. Row Intake opens a popup for the scout to select High Row, Mid Row, or Low Row which collects when teams remove a scored game piece (whether unintentionally or intentionally) from a node.

Switching Intake and Scoring Buttons

Because robots can only hold one game piece at a time, the intake buttons and scoring buttons are not displayed simultaneously. Whenever a scout taps an intake counter, the app switches all the intake buttons to the scoring buttons, and vice versa. Certain buttons such as Undo, Redo, Incap, and New Charge Attempt will always remain.

Scoring

Cone	Cube	None
Scoring		
High Cube - 0	High Cone - 0	
Mid Cube - 0	Mid Cone - 0	
Low Cube - 0	Low Cone - 0	
Failed - 0		
Supercharge		
Start Incap	New Charge Attempt...	
AUTO - 149	1591	To Teleop

Auto Scoring

Undo Intake Single	Redo
Scoring	
High Cube - 0	High Cone - 0
Mid Cube - 1	Mid Cone - 0
Low Cube - 0	Low Cone - 0
Failed - 0	
Supercharge	
Start Incap	New Charge Attempt...
TELEOP - 102	3536
	Proceed

Teleop Scoring

Scoring is recorded throughout the entire match. After recording an intake, the scout records the corresponding game piece type and location that the robot scored; if the robot loses control of the intaken piece (whether intentional or unintentional), then the scout selects Failed.

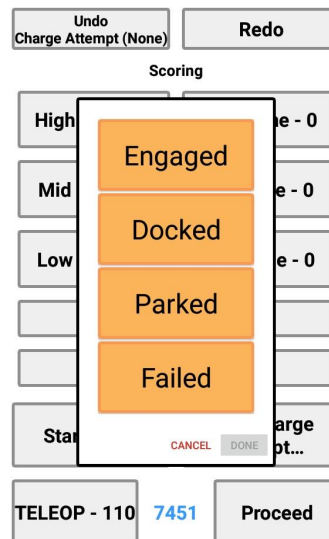
Supercharged

Undo Supercharge	Redo
Scoring	
High Cube - 1	High Cone - 1
Mid Cube - 0	Mid Cone - 0
Low Cube - 0	Low Cone - 0
Failed - 0	
Supercharge	
Start Incap	New Charge Attempt...
TELEOP - 23	1591
	Proceed

Scoring Screen after Supercharged

After *FIRST*'s Supercharging update, we added a Supercharge button to the Collection Screen. The scout presses this button when their robot places a game piece in a node already containing a scored game piece (regardless of whether all nodes were filled).

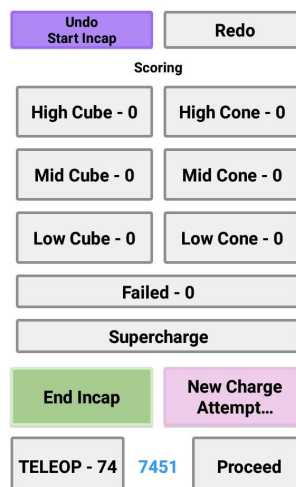
Charge Station Pop-up



Charge Station Pop-up

Pressing the New Charge Attempt button opens the charge station pop-up. During auto, scouts can select Docked, Engaged, or Failed. During teleop, scouts can select Docked, Engaged, Parked, and Failed.

Incap Duration Using Timestamps





Incap

In teleop 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.

Subjective Collection

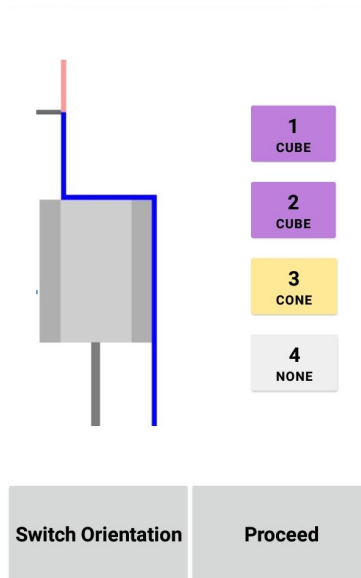
Information Input Screen

 	Match Number 30
Automatic Assignment	Team One 5454
	Team Two 1339
Backup 3	Team Three 2052
Old QRs	Proceed
Version: 5.0.0	

Subjective Information Input Screen

The Subjective Information Input Screen uses the same code as the Objective Information Input Screen. Its only differences are having three team numbers, no Scout ID, and allowing scouts to change the alliance color. Scout IDs are not assigned to Subjective Scouts since only two scout in a given match, and thus management of scouted alliances is simple. For the same reason, alliance color can be changed in Automatic Assignment mode.

Starting Screen

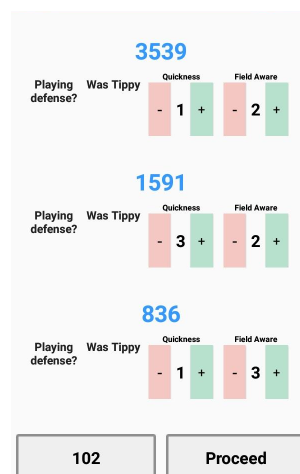


Subjective Starting Screen (Blue Alliance)

This screen is used to record an alliance's initial game piece setup on the field by displaying a diagram of their side of the field. Pressing a numbered button will change the game piece type.

Collection Screen

Quickness and Field Awareness



Quickness and Field Awareness Ranking (Right)

In each match, Subjective Scouts rank each robot on an alliance—relative to the other robots on the alliance—in 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 maximum speed and maneuverability of the robot. Field Awareness is based on the driver's awareness of where their robot is on the field, where other robots are, and where game pieces are.

Defense Checkbox

Defense	Was Tippy	Quickness	Field Aware
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

146 Proceed

Defense Checkbox (Left)

Subjective Scouts use a checkbox to indicate whether a robot is playing defense. This toggle button collects timestamps that are utilized if a strategist wants to review a robot's defensive ability in a match video.

Tippiness Checkbox

Playing defense?	Was Tippy	Quickness	Field Aware
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

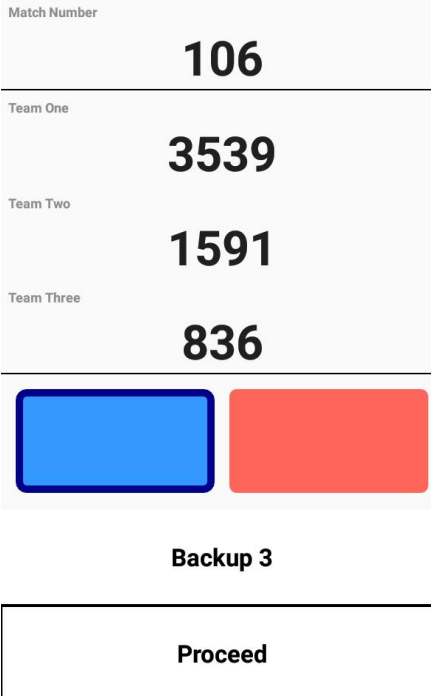
65 Proceed

Tippiness Checkbox (Right of Defense Checkbox)

A robot is marked as “tippy” if a Subjective Scout believes that it had collided with another robot, the former would have fallen over. This data ensures we know if a robot has a high center of gravity.

Screens Used in Objective & Subjective Modes

Edit Information Screen



The screenshot displays the 'Edit Information Screen' with the following layout:

- Match Number:** 106
- Team One:** 3539
- Team Two:** 1591
- Team Three:** 836
- Below the scores are two colored rectangular buttons: a blue one on the left and a red one on the right.
- Below the buttons is the text **Backup 3**.
- At the bottom is a large white button with a black border labeled **Proceed**.

Edit Information Screen

After the scout finishes scouting a match, they can edit relevant fields in case they accidentally scouted the wrong robot or otherwise inputted inaccurate information.

The code for this screen and the Information Input Screen was written with reusability in mind—neither screens contain game-specific fields, only requiring minor changes such as updating scout names for each competition. This enables us to update the Match Collection app quickly after each season’s Kickoff; we only need to change collected data fields.

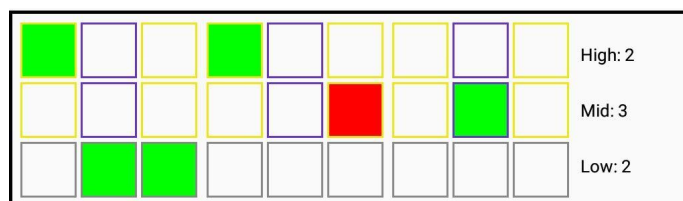
QR Code Screen



QR Code Generator Screen

QRs follow a specific formatting defined in Schema to minimize the amount of data that needs to be displayed. Data Point names are represented by letters of the alphabet, and each section and data point is delimited with special symbols such as '\$'. This way, the QRs can be easily generated and contain match data without being too large in size. We generate our QR codes using Sumimakito's [AwesomeQRCode](#) on GitHub.

Playoff Scouting



Playoff Scoring Screen

Playoff Scouting was implemented for the World Championship, consisting of buttons that represent the alliance grid. When a scout taps a box (representing a node) once, it becomes green, signifying one scored game piece in that location. If the same box is tapped again, it becomes red, signifying that the node is Supercharged. At the end of the match, scouts record the number of game pieces scored in each row and post the data in Slack (Citrus Circuits' communication platform).

PIT COLLECTION

Each year, the Pit Collection app is used primarily by the Match Strategist to collect mechanical data and robot pictures. During this year's World Championship, the app was also used by SEALS to record forkability and whether a robot has ground intake. Citrus SEALS is a team of students which supports upcoming alliance partners and ensures their robots' functionality. Pit Collection runs on Android phones and was built using Kotlin and XML.

In past years, there was a subjective mode used by SEALS to record the feasibility of installing a new mechanism on a team ("cheesecaking"). It was not implemented this year as we found subjective pit data unnecessary to collect.

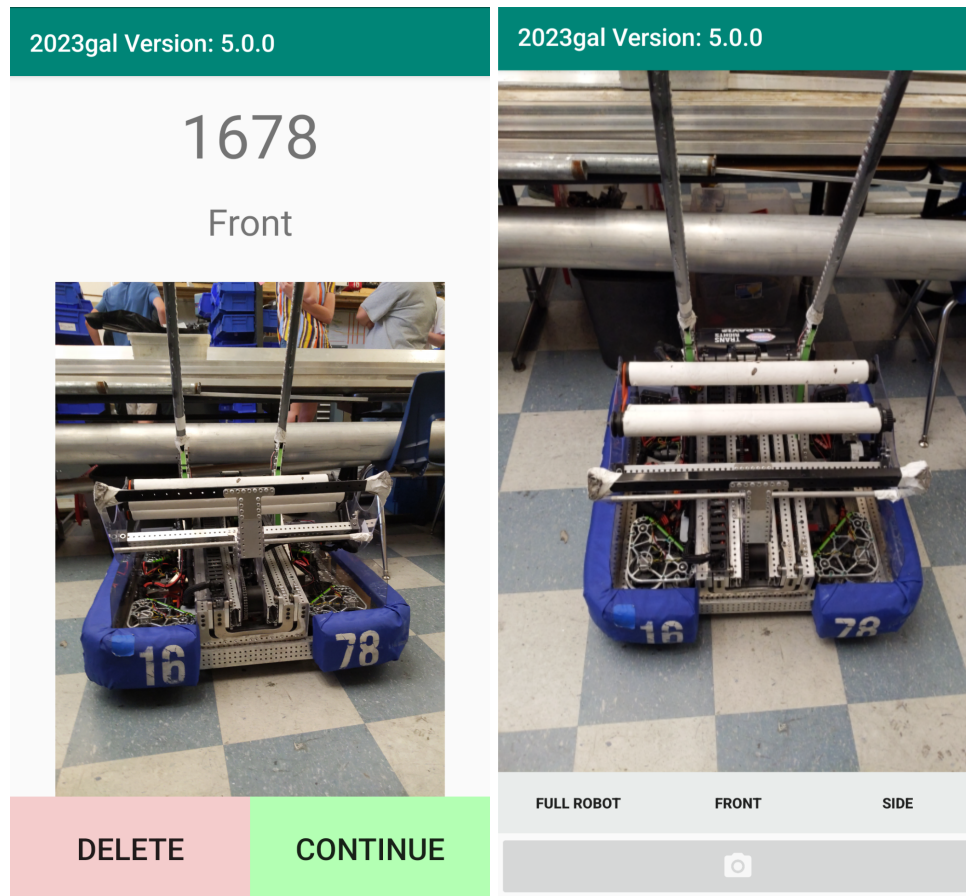
Data Collection

The screenshot shows the 'Data Collection' screen of the Pit Collection app. At the top, a teal header bar displays '2023gal Version: 5.0.0'. Below this, the robot number '59' is prominently displayed in a large font, with a camera icon to its right. The interface is divided into several sections: a green box labeled 'HAS COMMUNICATION DEVICE' and a pink box labeled 'DOES NOT HAVE VISION'; a section with 'Has Ground Intake' and 'Is Forkable'; a 'Weight (lbs.)' field with the value '80' and a 'Length (in.)' field with the value '120'; a 'Width (in.)' field with the value '53'; a 'Mecanum' checkbox; a 'Cim' checkbox; and a '3' in a box at the bottom. The background is a light gray with a subtle grid pattern.

Data Collection Screen

The Pit Collection app is used to collect data on a robot's physical characteristics—this includes whether a robot can charge, whether it has a ground intake, whether it has a device for communication with human players, whether our robot was able to use its fork mechanics on it, whether it has a camera for vision on the far side of the field, its dimensions, its drivetrain, its drivetrain motor type, and the number of drivetrain motors.

Robot Photos

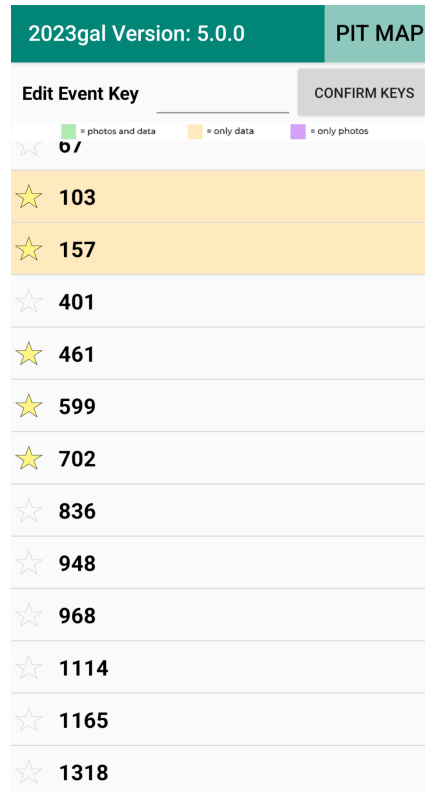


Robot Photo Screen

Robot photos are used primarily during picklist creation for strategists to instantly remember a robot's characteristics and appearance. When pit scouting, the Match Strategist will take photos of the full robot, its front, and its side (the app utilizes the phone's built-in camera). The photos are stored in the local Downloads folder to be pulled by the Server computer later via a USB connection. The user can review a photo by holding down the name of the photo type.

Team List Screen

Starring Teams



Team List Screen with Starred Teams

If a team's cell is held down, a yellow star will appear on the left side of the cell. Its primary uses are if the user wants to return to a team later or if there are multiple Pit Scouts dividing up teams to scout.

Flagging Teams

Flagging allows users to quickly obtain a list of teams that meet certain criteria. For instance, if a certain drivetrain type is a flag, a list of teams with that drivetrain is saved to the device's clipboard whenever the user marks a flag for a team. Criteria are set while developing the app.

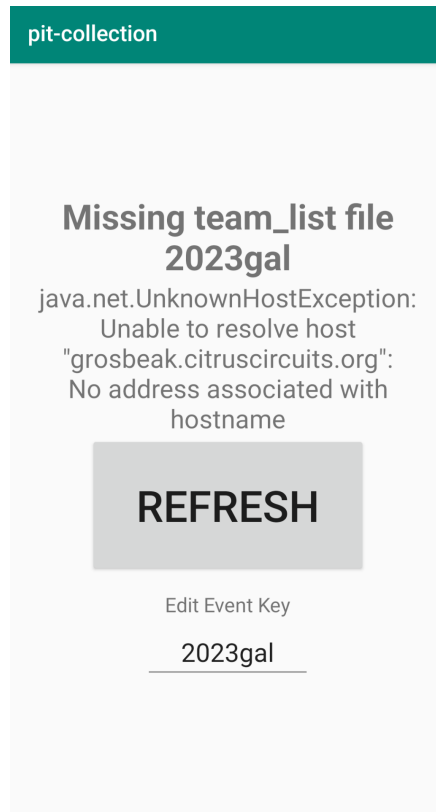
Automatic Highlighting to Show Scouting Progress

2023gal Version: 5.0.0		PIT MAP
Edit Event Key		CONFIRM KEYS
<div><div></div> = photos and data</div>		<div><div></div> = only data</div>
		<div><div></div> = only photos</div>
59		
☆ 67		
☆ 103		
★ 157		
☆ 401		
☆ 461		
★ 599		
☆ 702		
☆ 836		
☆ 948		
☆ 968		
☆ 1114		
☆ 1165		

Team List Screen with Highlighted Teams (Purple and Yellow)

In the team list, the color of a team's cell is determined by the amount of data already stored for that team. If there is no data or pictures of a team, the cell is white. If there are only pictures, the cell is pink. If there are only data points, the cell is yellow, and if there are both data and pictures, the cell is green. This shows the Pit Scout what still needs to be collected.

Editing Event Key



Event Key Screen

Users are able to change which event the Team List Screen corresponds to by changing the event key. Each event key is provided by The Blue Alliance, and if an invalid event key is inputted, an error screen will prompt the user to reenter a valid event key. The event key is stored in a text file in the device's Downloads folder so it is not reset after closing or updating the application.

Pit Map Screen

ROTATE

Galileo Division (Green Drape)

E25 9138	E26 5987	F25 5985	F26 3536	G25 3534	G26 1701	H25 1690	
E23 9128	E24 6702	F23 5801	F24 3539	G23 3476	G24 1708	H23 1678	
E17 9120	E18 6722	F17 5586	F18 3603	G17 3467	G18 1902	H17 1622	H18 59
E15 9105	E16 6800	F15 5557	F16 3996	G15 3229	G16 2194	H15 1591	H16 67
E13 9097	E14 7197	F13 5427	F14 4009	G13 3042	G14 2337	H13 1318	H14 103
E11 9072	E12 7258	F11 5199	F12 4381	G11 3005	G12 2383	H11 1165	H12 157
E09 9031	E10 7451	F09 5166	F10 4415	G09 COLMN	G10 2551	H09 1114	H10 401
E07 8033	E08 7461	F07 5010	F08 4422	G07 2910	G08 2614	H07 968	H08 461
E05 8013	E06 7657	F05 4786	F06 4451	G05 2881	G06 2630	H05 948	H06 599
E03 Radio	E04 INSP	F03 INSP	F04 4481	G03 2834	G04 2714	H03 836	H04 EMT
	E02 INSP	F01 INSP	F02 4571	G01 2830	G02 2771	H01 702	H02 EMT

Galileo Field Map

The Pit Map screen contains a full map of the pits with a visualization of every team's number and location. The map is taken directly from the user's phone's Downloads folder and its orientation can be changed. It is provided by the competition organizer.



STAND STRATEGIST

Overview

The Stand Strategist app is a desktop app for Stand Strategists to take notes about teams on an alliance in a match. These notes are used during picklist creation as a way to track observations not recorded by the rest of the system. Previously, we used Google Sheets, but found that Internet connections were too unreliable for effective note-taking. We developed Stand Strategist as an offline app running on Windows, macOS, and Linux, allowing our strategists to take notes on their own laptops even with the connectivity constraints of a competition environment. Stand Strategist is written in Kotlin and built using the [Compose Multiplatform](#) UI framework by JetBrains.

Navigation

Stand Strategist uses an in-house navigation implementation designed to be flexible to modifications to the navigation flow.

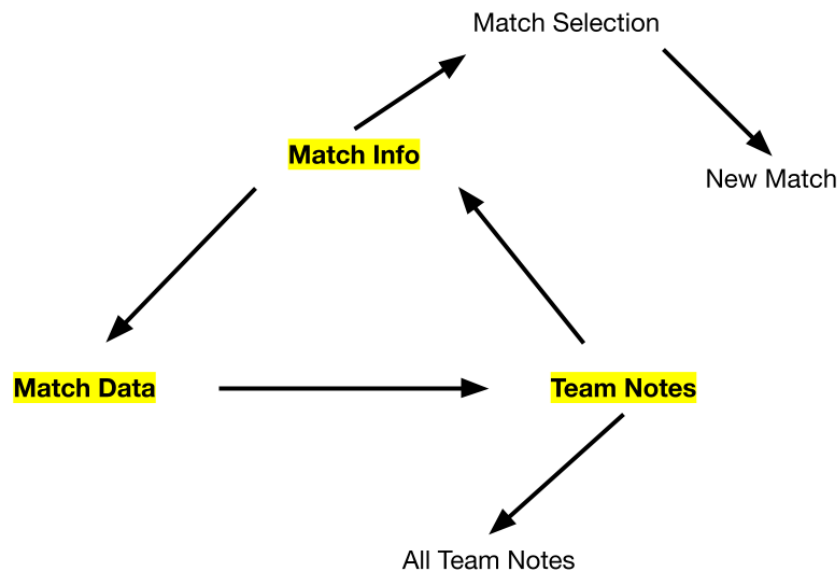
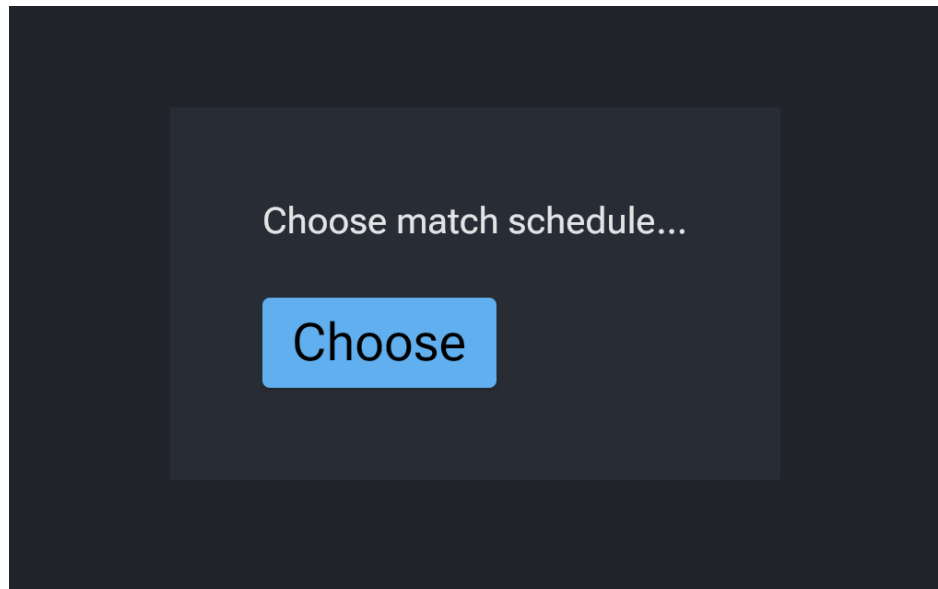


Diagram of Pages in the Navigation Flow

The highlighted pages in the diagram are intended to run in a cycle for each match played, so on those pages, the “Previous” and “Next” buttons will navigate to the relevant page in the cycle. This provides users easy access to the most relevant pages.

Selecting Match Schedule

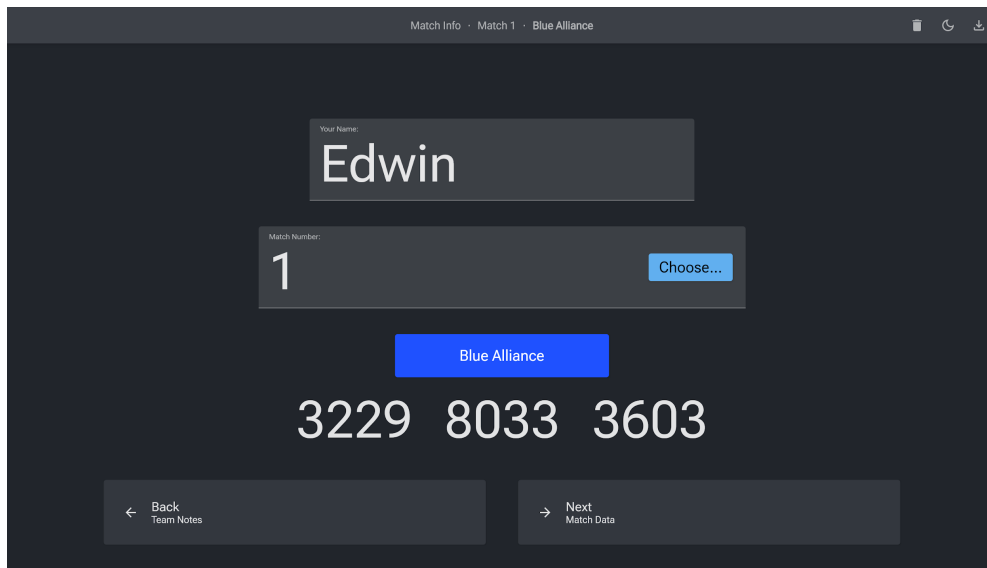


Match Schedule Selection Screen

To enter data into Stand Strategist, the user is first prompted to select a match schedule. Clicking “Choose” allows them to select the relevant file downloaded onto the laptop.

Navigation Flow

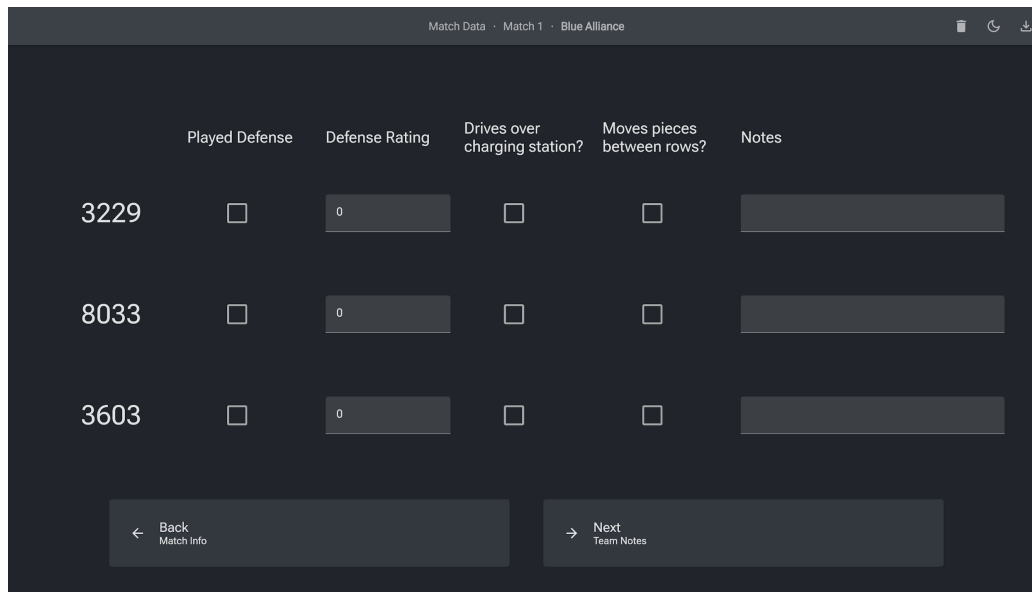
Match Information Screen

A dark-themed screen for match information input. At the top, a breadcrumb trail reads "Match Info · Match 1 · Blue Alliance". Below this, there's a "Your Name:" label followed by a text input field containing "Edwin". Underneath is a "Match Number:" label followed by a text input field containing "1" and a blue "Choose..." button. Below the match number field is a blue button labeled "Blue Alliance". Underneath that, the numbers "3229 8033 3603" are displayed in a large font. At the bottom, there are two buttons: "← Back Team Notes" and "→ Next Match Data".

Match Information Input Screen

Once a match schedule is selected, the Match Information page is displayed, allowing the user to edit their name, match number, and the alliance color they are scouting.

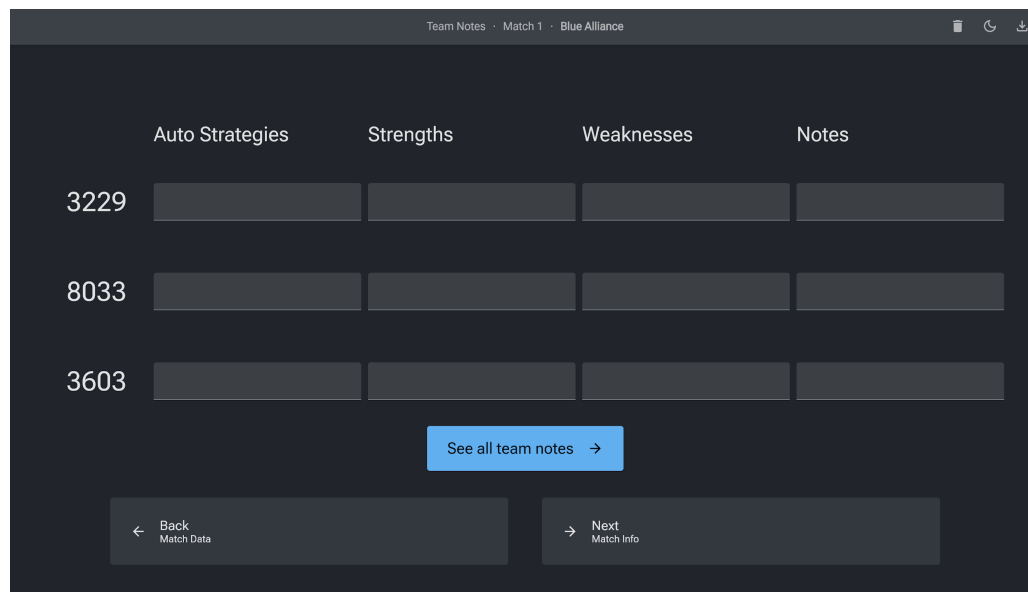
Match Data & Team Notes Screens



The screenshot shows the 'Match Data' screen for 'Match 1' and 'Blue Alliance'. It features a table with columns for 'Played Defense', 'Defense Rating', 'Drives over charging station?', 'Moves pieces between rows?', and 'Notes'. Three rows of data are visible for match numbers 3229, 8033, and 3603. Each row has a checkbox for 'Played Defense', a text input for 'Defense Rating' (all containing '0'), checkboxes for the other two categories, and a text input for 'Notes'. At the bottom, there are two buttons: 'Back Match Info' and 'Next Team Notes'.

	Played Defense	Defense Rating	Drives over charging station?	Moves pieces between rows?	Notes
3229	<input type="checkbox"/>	0	<input type="checkbox"/>	<input type="checkbox"/>	
8033	<input type="checkbox"/>	0	<input type="checkbox"/>	<input type="checkbox"/>	
3603	<input type="checkbox"/>	0	<input type="checkbox"/>	<input type="checkbox"/>	

Match Data Collection Screen



The screenshot shows the 'Team Notes' screen for 'Match 1' and 'Blue Alliance'. It features a table with columns for 'Auto Strategies', 'Strengths', 'Weaknesses', and 'Notes'. Three rows of data are visible for match numbers 3229, 8033, and 3603. Each row has four text input fields corresponding to the columns. At the bottom, there is a blue button labeled 'See all team notes' with a right arrow, and two buttons: 'Back Match Data' and 'Next Match Info'.

	Auto Strategies	Strengths	Weaknesses	Notes
3229				
8033				
3603				

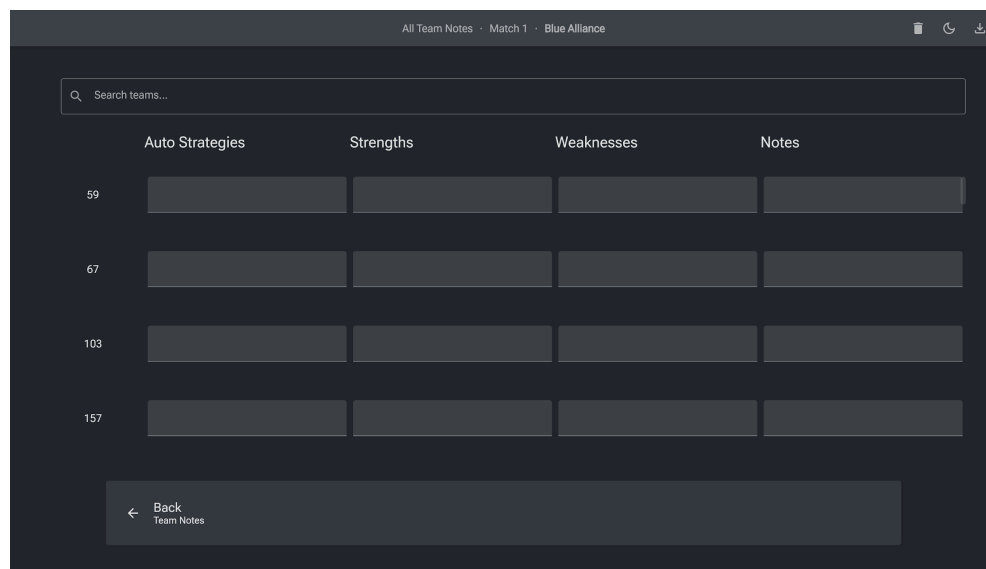
Team Notes Screen

During the match, the user can enter data and observations in the Match Data and Team Notes pages. The data entered in the Match Data page is relevant only to that specific match, while



notes in the Team Notes page for a given team are used across the entire competition. Thus, notes are more general comments on a team’s overall performance through all their matches.

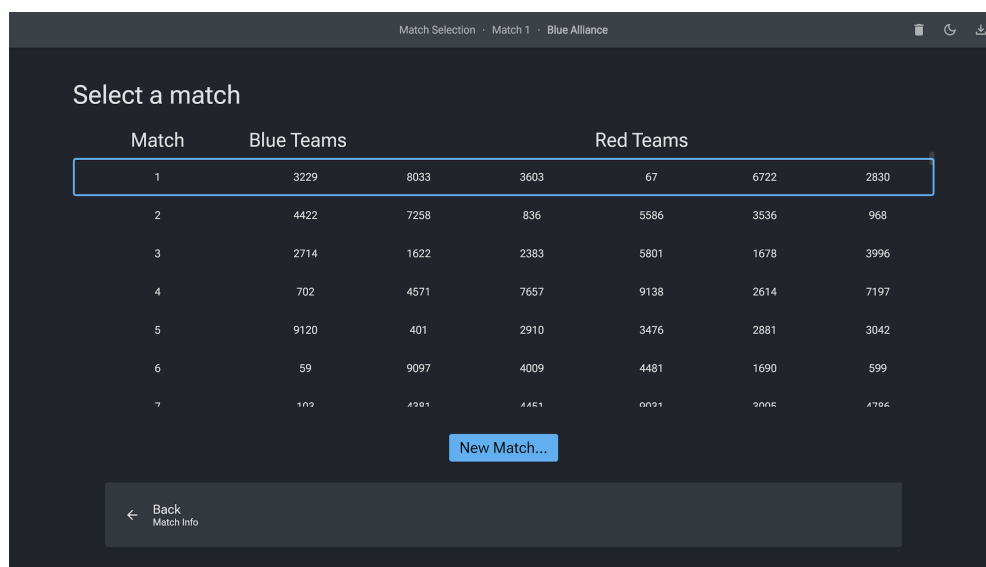
All Team Notes Screen



All Team Notes Screen

If the user wants to view and edit notes for all teams, they can click the “See all team notes” button on the Team Notes Screen. The list of team notes is scrollable and searchable.

Match Selection

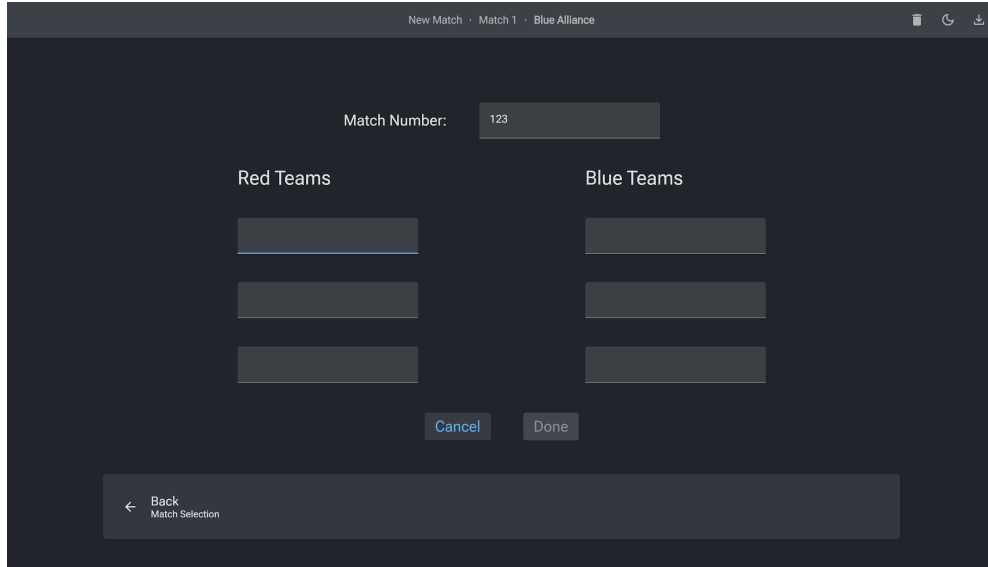


Match Selection Screen



By clicking the “Choose” button on the Match Information page, the user can open the Match Selection page, which shows a list of every match in the match schedule.

New Match Screen

The screenshot shows a mobile application interface for creating a new match. At the top, a breadcrumb trail reads "New Match · Match 1 · Blue Alliance". In the top right corner, there are three icons: a trash can, a refresh/clock icon, and a download icon. The main form has a "Match Number:" label followed by a text input field containing the number "123". Below this, there are two columns of input fields. The left column is titled "Red Teams" and contains three empty text boxes. The right column is titled "Blue Teams" and also contains three empty text boxes. At the bottom center of the form are two buttons: "Cancel" and "Done". At the very bottom of the screen is a navigation bar with a back arrow icon and the text "Back Match Selection".

New Match Screen

The ability to create custom matches is a feature created for Stand Strategists to practice scouting and taking notes. By clicking the ‘New Match’ button on the Match Selection page, users can create a new match with various editable fields. Matches can also be deleted.

Exporting Data

The data recorded across the entire competition can be exported by clicking the download icon at the top right of any page. An export is formatted as an Excel workbook file containing a Match Data sheet and a Team Notes sheet. This allows multiple exports from different Stand Strategists to be merged together into one Excel workbook or Google Sheets spreadsheet.

To clear all the data, the user can click the trash can icon at the top right.

In-season Changes

As the Stand Strategist app was new for this season, many changes were made throughout this year. We initially used [Kotlin Dataframe](#) as our in-memory tabular data storage solution. However, we ran into issues at our first regional due to type-checking corner cases and the app repeatedly crashed. We then switched to [kotlin-csv](#), which fixed our issues. The ability to create and delete matches was also added during the competition season in response to Stand Strategists’ feedback.

SERVER

Server is the main processing component of the scouting system and is coded in Python. During a competition, the Server works in tandem with the scouts to provide accurate data to the strategists and the Picklist Editor. As matches progress, the scouts collect data that is sent to a laptop that runs the Server. The Server performs various calculations on the data and stores the raw and calculated data on a local MongoDB database on the laptop. When the Server finishes running, the data is uploaded to a MongoDB database in the cloud, which is then accessed by our Grosbeak web server. For more information on databases, see Section 4.1.6.1 in our [2020 Whitepaper](#).

Schema

Schema provides an organized database design for the scouting system. It serves as the connecting “map” among the many apps to facilitate data transfer, allowing for efficient calculations and editing of data points. Data points are updated at the beginning of each competition season to relate to the game, and are also often changed as the season progresses. Variables can be changed without accessing the code directly. The ability to change all the datapoint information in Schema ensures that only minimal changes need to be made to Server files and prevents hard coding of values.

Collecting Charge Station Data

After the release of this year’s game, we determined that we wanted to differentiate between charging station Engages and Docks in the scouting data. Although TBA does report which robots engage/dock in Auto and Endgame, the information it provided was not sufficient for our purposes. Without having manually collecting charge station attempts, we are unable to tell if a robot attempted to engage but failed, which would limit us from calculating charge success rates. Additionally, TBA reports a robot as engaged if they were fouled and awarded an engage even if the robot did not actually engage (rule G209). In order to ensure complete accuracy, Objective Scouts were required to scout charge station attempts.

Device Data Pulling

Data collected from tablets must be pulled in order to be processed by Server. Data collected by scouts are stored as QRs, which can be scanned with QR scanners and uploaded to the database. Alternatively, the `qr_input.py` script can pull QR data obtained through Match Collection from all tablets plugged into the server laptop. The QR data is also stored locally on the tablets in case QR scanning does not work. Additionally, data in the form of JSONs and images obtained through Pit Collection are pulled when the phones are connected to the Server laptop. Server then stores this data in a directory and uploads it to the database. Once



all the data is processed, phones with the Viewer app can connect to the Server computer and all images can be pushed to the Downloads folder of the phone using the `send_viewer_images.py` script. Images and data models can then be viewed through the Viewer app.

QR Handling

Match Collection data is transferred from the tablets to the server computer in a compressed string format as a QR code. The QR code is then decompressed into a usable format using our Schema.

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 missed a large portion of the match, leading to inaccurate data, or if a match gets replayed. Blocklisting a QR will not delete it from the database but will flag it to be ignored by calculations. In the case that a specific datapoint for a team in a given match needs to be modified, the QR can be overridden by flagging the specific data point's calculation to use a given value instead of the one from the QR code. The ability to override specific data points was implemented this year due to anticipating using it to override ferrying teams' failed scores (our definition of failed scores included robots who were intentionally dropping game pieces). We are also able to reverse an override in case of an error. Calculations must be rerun after blocklisting or overriding the QR in order for the changes to affect the data.

CSV Exporting

Although all the data points collected are viewable on the Viewer app, it is necessary to be able to work with the data on a spreadsheet mid-competition. Because of this, certain data is exported as CSV files to be easily used in a spreadsheet or another data visualization tool.

Team and Team-In-Match data are exported along with data from TBA in CSV format, which allows strategists to work with the data in a spreadsheet whenever necessary. Most importantly though, CSV exports are how data is given to the Picklist Editor.

Once competitions are over, data is exported to help us analyze our predicted calculations, data accuracy, scout accuracy, and much more.

While it is possible to create a system that automatically sends the exported data to a spreadsheet or similar software, we found it easier to manually send the data through Slack, as an actual system would require unnecessary effort and would not provide additional benefit.



Auto Paths

This year, a new auto paths calculation was added to collect every action a team performs in the autonomous period—collecting all scoring, intake, and charging actions. By using all the actions a team performed in the order in which they were done, we gain a sense of what their "path" in auto was. This is particularly useful for strategists to compare our team's auto compatibility with other teams. Calculated auto paths data was used in our pickability metrics and was a key component of assessing a team's overall performance and ability.

The calculation first consolidates all timelines (lists of dictionaries containing all the actions a team performed in the match in chronological order) collected on a robot in the match by different Objective Scouts. Then, it filters out all actions except the ones that happened in auto and sorts the actions into a readable order. The output consists of variables storing their preload, first intake & score, second intake & score, charge level, etc.

Rather than listing every auto path by match, auto paths are matched together to calculate the success rates of a team's auto path. Auto paths are matched together based on start position, intake/score attempts, and charge station 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 the charge station and scoring.

Predicted Calculations

During competitions, Server makes predictions on future match scores, ranking points (RPs), probabilities of winning, and final rankings for each team.

Predicted Alliance In Match (AIM) data is calculated using previously calculated and The Blue Alliance (TBA) data. This data collection includes the chance of getting each RP, the final score, and the probability of winning.

For the grid, the average count of cubes/cones in each row is summed for all teams on the alliance. Any leftover high pieces over the possible number (3 for cubes and 6 for cones) are treated as middle row pieces, any leftover middle pieces as low, and any leftover low pieces as supercharged (if there is a filled grid). Optimal placement of game pieces is assumed when predicting the number of links.

The predicted charge score is calculated by assuming the alliance sends the robot with the highest expected charge score to engage/dock in auto and using the sum of all teams' engage/dock percent success in teleop.

The number of parked teams is calculated by multiplying the number predicted to fail docking by the average park success percent of the alliance. The mobility points are predicted by summing the mobility chance for each team on the alliance using data from The Blue Alliance.



The predictions for whether the Link RP will be achieved check whether the number of links is greater than or equal to six. If it is five, the value is the percentage of qualification matches played (from TBA) where Coopertition occurred.

The predicted Charge RP is calculated by multiplying the chance of docking (of the team with the highest docking percent) in auto by the engaging percent (of each of the two best engagers) in teleop.

Finally, the Predicted Score is calculated by multiplying each of the predicted totals for the number of links, pieces on each row, mobility, auto pieces, supercharged pieces, and parked robots by each respective point value and adding our predicted charge scores for auto and teleop.

Using this, the Win Chance for each match is calculated. To do this, we generate a mathematical function by running logistic regression on played matches—fitting past matches' predicted point differences against actual winning alliances (using data from TBA). This generates a function that compares the predicted point difference and the chance of winning. Then, the predicted point difference between the alliances in each future match is put into the function, which returns the win chance.

There are special calculations for playoffs—the predicted auto and teleop 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.

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 data points.

We improve our pickability metrics after each competition. 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 data point. 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 season, first pickability was calculated using a weighted sum of the data points which measure a team's expected offensive ability in a match. On the other hand, second pickability changed quite a lot throughout the season. Originally, second pickability was split into offensive and defensive pickabilities, since at the beginning it was thought that both an offensive and defensive second robot would have their situations. However, it quickly became apparent that defense was not a factor and second pickability became one calculation once again. And once the auto paths calculation was created (see [Auto Paths](#)), it became possible to create different pickability calculations based on the compatibility of our autos. The goal of second pickability is to measure a team's auto compatibility with us and our first pick. To see the specific weights for each pickability calculation, see the [Pickability Weights](#).

Scout Precision Ranking

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 Objective Scouts on an alliance to find how far off a specific scout is from their expected data. Since there are nine Objective Scouts per alliance, with three on each robot, there are 27 combinations of three Objective Scouts that will contain one scout from each robot. For each of these combinations, the values of the data points for each scout are totaled to get the overall alliance score. The official value for each data point is pulled (and calculated based on a basic schema programming language) from TBA and then compared against the total scouted score for the alliance to calculate the amount of error. 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 error for each scout in a match is calculated, the formula looks at each three-Scout combination that a specific scout was in. The average errors of the other two Objective Scouts in that combination are divided by 3 (since errors are the result of 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 value. 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](#).



We have considered utilizing SIM Precision in TIM consolidation. This could be done by subtracting each scout's calculated error, for each data point in each match, 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, this has not been implemented due to the long runtime of Scout Precision calculations, leading us to only run it during long breaks at competition instead of every match.



GROSBEAK

Grosbeak is a Python FastAPI web server that allows secure and reliable data transfer between the Viewer app and MongoDB. It was created during the 2022 competition season for Viewer's Live Picklist feature but has since fully replaced our old web server, Cardinal. Cardinal was built in a short timeframe during the 2021 offseason and was replaced due to poor documentation and reliability issues.

In the 2022 offseason, we fleshed out support for fetching data and static files (match schedule and team list) as well as converting our WebSocket-based Live Picklist into a simpler REST API called Online Picklist. We also added a new method of retrieving data that combines collection documents with other similar documents in related collections. For example, if we have the same team in subjective and objective data sets, the documents are combined. This method greatly simplified code in Viewer and increased performance by offloading some looping to the server. We attempted to add caching for this because requests were taking longer than they should have, but we were not able to because of limitations with MongoDB Atlas.

To get all the data points, Viewer sends an HTTP request and everything is returned. Viewer and Pit Collection also use an HTTP request to get the match schedule and team list for a competition.

Previously, for Cardinal, we had to manually copy static files to the computer hosting the web server to make them accessible in Viewer. In Grosbeak, we created a simple webpage using HTML and JavaScript that allows us to easily upload static files.

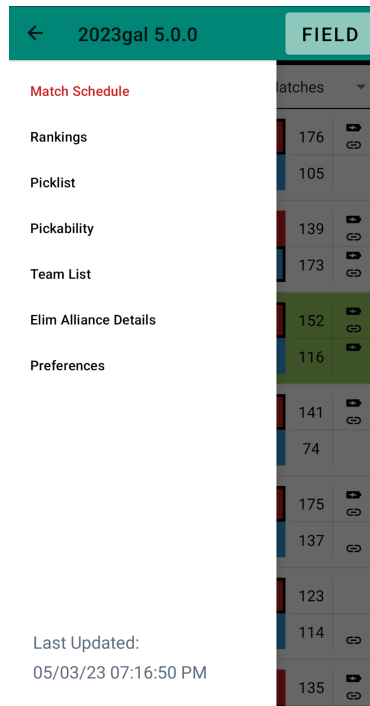
After adding that feature, we added synchronization from Picklist Editor to the Online Picklist that Viewer uses.



VIEWER

Viewer is an Android phone app that allows strategists to review, organize, and visualize processed scouting data live during competition in order to create educated match strategies. It is built in Kotlin and XML.

Navigation



Navigation Sidebar

Viewer uses a sidebar for users to 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). To optimize navigation, the sidebar can be viewed from any screen or page in the app. Additionally, the current event and version number are displayed at the top.

User Preferences

≡ 2023gal 5.0.0 FIELD

User's Datapoints

RESET TO DEFAULT

SELECT ALL

Visual Data Bars

Matches Played

Current RPs

Pred. RPs

Current Avg RPs

Current Rank

Pred. Rank

First Pickability

Second Pickability

Auto

Mode Start Position

Middle Compatibility

No Show

Start Pos 1

Start Pos 2

Start Pos 3

Start Pos 4

Avg All Low

Avg Cubes Low

Preferences Data Points List

In the Preferences page, users can select which data points are displayed in the Team Details page. This is essential because we have multiple users, and one user may want to quickly view more data on a team's average intakes, while another would rather view a team's motor type. The selected data points are saved in a file stored in the Downloads folder on the device, so they aren't reset after closing or updating the app. There are multiple profiles on Viewer, allowing our users to each have their own different default list of data points that will be displayed.

2023gal 5.0.0

FIELD

Preferences

User Name

Other

EDIT

STAR OUR MATCHES

Version 5.0.0

Edit Event Key

2023gal

Edit Schedule Key

2023gal

CONFIRM KEYS

Preferences Screen

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

Match Schedule

2023gal 5.0.0 FIELD						
Team #		All Matches				
1	✓	67	6722	2830	176	
		3229	8033	3603	105	
2	✓	5586	3536	968	139	
		4422	7258	836	173	
3	✓	5801	1678	3996	152	
		2714	1622	2383	116	
4	✓	9138	2614	7197	141	
		702	4571	7657	74	
5	✓	3476	2881	3042	175	
		9120	401	2910	137	
6	✓	4481	1690	599	123	
		59	9097	4009	114	
7		9031	3005	4786	135	

Match Schedule



A list of matches is displayed in the Match Schedule, showing which teams are in each match, the total match score, and any RPs earned. Matches that have been played have a checkmark icon below the match number, and if not yet played, displays predicted data. The alliance that won the match is boxed with a black border. Additionally, a battery icon is displayed if the alliance got the charge RP and a link icon is shown if they got the link RP. The matches that our team plays in are highlighted green. A search bar allows users to filter matches by a team number and they can submit the search to go directly to the team's Team Details page. A drop down at the top right allows filtering by All Matches, Our Matches, or Starred Matches. A user can star a match by long tapping it, this makes a star symbol appear above the match number. The list of starred matches is saved to a file in the Downloads folder so that if the app needs to be reinstalled, it saves the starred matches.

Match Details

≡		2023gal 5.0.0				FIELD			
Actual Score		116				Actual Score		152	
Actual Charge RP		1				Actual Charge RP		1	
Actual Link RP		0				Actual Link RP		1	
Win Chance		0.0%				Win Chance		100.0%	
3									
Team		2714	1622	2383	5801	1678	3996		
Current Avg RPs		2.1	2.1	2.3	2.7	4.0	2.8		
Auto									
Start Position		3	1	2	2	1	3		
Preloaded Piece		▲	▲	■	▲	▲	▲		
# Cubes Low		0	0	0	0	0	0		
# Cubes Mid		0	0	0	0	0	0		
# Cubes High		0	0	1	0	1	1		

Match Details

After tapping on a match, the match's Match Details page is opened. If the match has not been played, the data points displayed will be predictions and averages, and if it has been played, the data points will be the actual data collected by our scouts and data pulled from The Blue Alliance. In the header, alliance specific data is displayed (e.g. predicted score) and in the data section, team data is displayed in a table.



Team List

2023gal 5.0.0 FIELD		
Team List		
★	Team Number	Team Name
★	59	RamTech
•	67	The HOT Team
•	103	Cybersonics
•	157	AZTECHS
•	401	Copperhead Robotics
•	461	Westside Boiler Invasion
★	599	The Robodox
•	702	Bagel Bytes
•	836	The RoboBees

Team List

Users can view a list of all teams at the event using this page. By tapping to the left of the team number, users can also star teams. On the Match Schedule page, if a match has a starred team, it turns a light yellow, if it has more than one, the match will turn a darker yellow.

2023gal 5.0.0 FIELD						
Team #		All Matches				
4	✓	9138	2614	7197	141	📄
		702	4571	7657	74	
5	✓	3476	2881	3042	175	📄
		9120	401	2910	137	📄
6	✓	4481	1690	599	123	📄
		59	9097	4009	114	📄
7	✓	9031	3005	4786	135	📄
		103	4381	4451	180	📄
8	✓	3539	1114	2834	167	📄
		1902	5557	2551	124	📄
9	✓	5199	3467	948	188	📄
		461	157	9128	130	
10		1591	9072	2771	124	📄

Match with Starred Team (Match 6)



Team Details

2023gal 5.0.0

FIELD

TO L4M

1678

PICTURE

Citrus Circuits

AUTO PATHS

See Matches →

Notes (click below to edit)

Team Notes

Matches Played	10
1 # Current RPs	40
1 # Pred. RPs	40.0
1 Current Avg RPs	4.0
1 Current Rank	1
1 Pred. Rank	1
3 First Pickability	55.8
? Second Pickability	?

Auto

Team Details

Every team at the competition has a Team Details page that features calculated data points such as averages, standard deviations, and more. Rankings are shown on the left side column to display the ranking of the team for that specific data point in comparison to other teams.

61 Avg Cubes High	0.1
66 Avg Cubes Total	1.0
66 Max Cubes	2
54 Avg Cones Mid	0.5
55 Avg Cones High	0.9
59 Avg Cones Total	1.4
60 Max Cones	3
69 Avg Game Pieces	2.4
63 Max Game Pieces	5
65 Max Incap	86
75 # Matches Incap	4
16 Matches Played	0

Data Bars

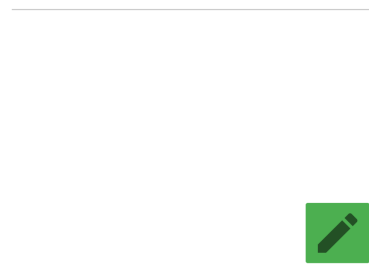
Optional colored data bars can be displayed behind each data point to visualize how that team compares in that data point to others. The percentage of the colored data bar is the team's

value for that data point divided by the highest value for that data point across all the teams at the competition. Data bars for data points related to scoring (ex. Avg All Low) are colored in green, and go from left to right. Data points related to cubes (ex. Avg Cubes Low) are colored in purple, while data points related to cones are colored in yellow. Data bars for data points like incap time and fouls (data points where the higher the value of the data point, the lower 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

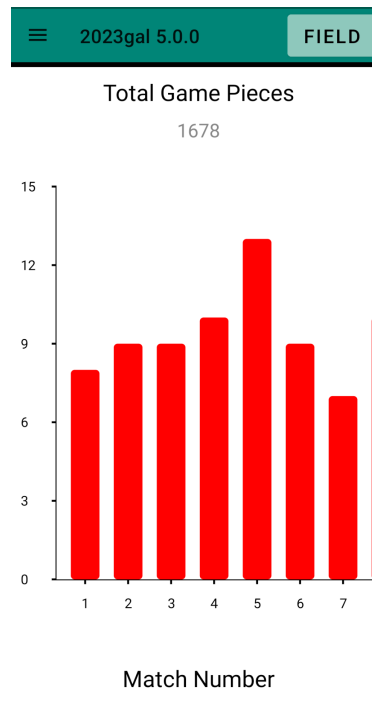


Team Notes

Users can write notes on teams by tapping on the orange Notes button in each team's Team Details page. The notes they take will be stored on the Grosbeak web server and pulled by all the other devices so that all other users can share notes.



Data Graphs for Specific Data Points

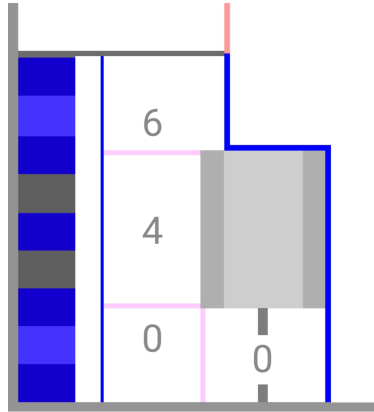


Data Graph for Total Game Pieces

Tapping on a data point in a Team Details page opens up its TIM (Team-in-Match) Data Graph—a bar graph of match number versus the data point’s value. For example, when tapping on the data point “Avg Game Pieces”, a graph will appear displaying the team’s average game pieces per match. Each of the graph’s bars show the number of game pieces scored by the team in the corresponding match. Tapping on a bar opens the corresponding match’s Match Details page. This used to be implemented using the [MPAndroidChart](#) library but we switched to [YCharts](#) this year because it was easier to implement and significantly reduced the amount of required code. Another type of graph can be viewed by tapping on any of the start position data points on the Team Details page. Using this graph, the user can see the number of times the team started in each start position.

Start Position

1678



Start Position Graph

Rankings for Specific Data Points

Avg Game Pieces

1	1678	9.60
2	3005	9.22
3	3476	8.60
4	6722	8.10
5	2910	7.80
6	4481	7.78
7	5987	7.70
7	3467	7.70
9	67	7.60
10	1690	7.50
10	1114	7.50
12	461	7.40
12	2614	7.40
14	8033	7.30

Rankings for Avg Game Pieces

By long pressing on a data point on a Team Details page, a user can view a ranked list of all the teams by that data point. These ranks are also displayed on the left side of Team Detail cells, but aren't refreshed as often to avoid lag. By viewing a ranked list, users can see which teams are above and below a certain team. By clicking on the cells in the ranked list, users can open a team's Team Details page.

Last Four Matches

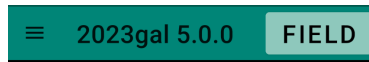
≡	2023gal 5.0.0	FIELD
TO ALL MATCHES	1678	PICTURE
Citrus Circuits		
AUTO PATHS		
See matches →		
Notes (click below to edit)		
Matches Played		10
1	# Current RPs	40
1	Current Rank	1
1	Current Avg RPs	4.0
1	# Pred. RPs	40.0
1	Pred. Rank	1
3	First Pickability	55.8
?	Second Pickability	?
L4M Auto		

Last Four Matches

In a Team Details page, users can toggle between all-matches data points to the same data points except calculated only from a team's last four matches. This is used to estimate a team's performance during eliminations, since their performance at the beginning of the competition is likely different from at the end of qualification matches.

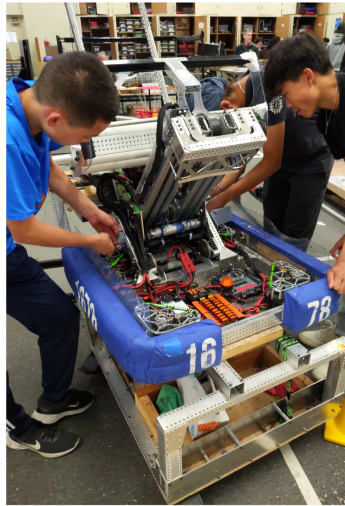


Robot Images



1678

Citrus Circuits

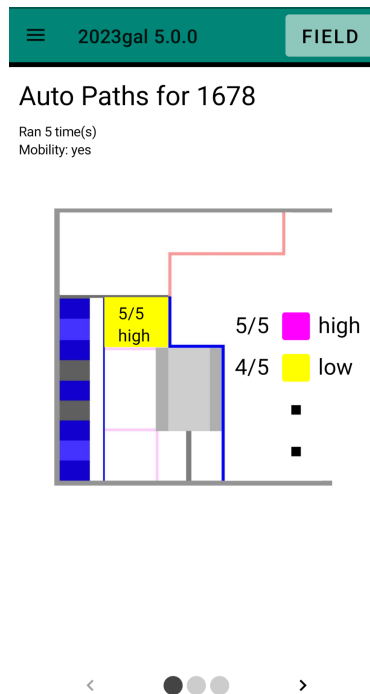


[View Robot Images](#)

In a Team Details page, users can view images of the team's robot by tapping the Pictures button. There are multiple pictures for each team, each showing the robot from a different angle. These pictures are taken using our Pit Collection app and are manually transferred to the other phones by plugging in the phones to Server.



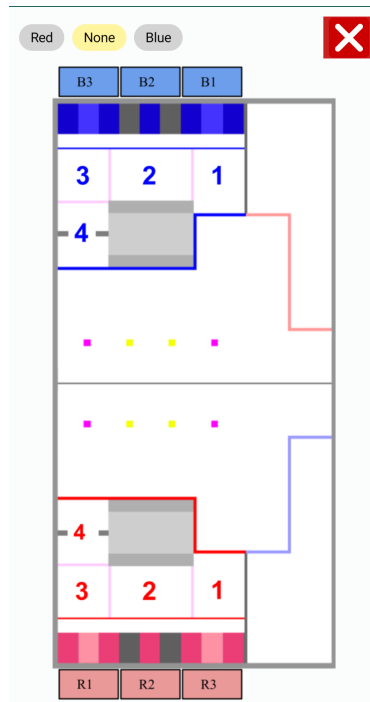
Auto Paths



Auto Paths

Auto Paths was a new feature that was added for Championship 2023 once strategists realized how important having a compatible auto was. This visualizes the auto paths of teams, starting with their most common autos and scrolling through other auto paths they have. The map shows auto intake successes, scoring successes, and engage/dock successes. For example, if it shows 4/5 high in yellow near the grid, it means that the team tried scoring a high cone in that grid area 5 times and made 4 of them. This additional page was imperative to creating our 2nd and 3rd picklist for Championship to know the success rates and compatibility of teams' autos.

Field Map



Field Map

In Viewer, users can view the Field Map. This view shows the possible starting positions, driver stations, and positions of the starting game pieces.

Pickability

2023gal 5.0.0			FIELD
Pickability			
Rank	Team Number	1st	▼
1	3476	58.5	
2	3005	57.0	
3	1678	55.8	
4	2910	54.9	
5	1114	53.4	
6	1690	53.2	
7	67	52.2	
8	4481	50.2	
9	7461	50.2	
10	5987	49.8	
11	8033	49.6	
12	2614	49.4	
13	1591	48.6	
14	2630	48.3	
15	1318	48.1	
16	3467	47.3	
17	461	47.0	
18	401	46.9	
19	836	46.4	
20	4381	45.8	
21	2337	45.0	
22	6800	44.9	

Pickability

Pickability ranks teams based on 1st and 2nd pickability. These calculations are determined by preset weightings based on what is deemed as the most important for a chosen robot to have. Users can switch from 1st and 2nd pickability using the dropdown in the top right. Clicking on the team number will open the team's Team Details page. During this season, we began to switch 2nd pickability to two separate calculations: Offensive and Defensive 2nd pickability. 2nd pickability was later removed out of a lack of necessity from users.



Picklist

2023gal 5.0.0 FIELD	
REFRESH	ONLINE OFFLINE
Team Number	Picklist Rank
1678	1
3476	2
1114	3
103	4
8013	5
6800	6
4451	7
3539	8
7197	9
3005	10
2910	11
3467	12

Online Picklist

2023gal 5.0.0 FIELD		
↓	↑	ONLINE OFFLINE
Team Number	Local Rank	Imported Rank
103	1	1
1114	2	2
1165	3	3
1318	4	4
157	5	5
1591	6	6
1622	7	7
1678	8	8
1690	9	9
1701	10	10

Offline Picklist

Picklist allows users to edit their own personal picklist and see the most updated picklist. It has two versions, online and offline. The online version fetches the most recent version from Grosbeak and is not editable.

The offline version is editable and is stored locally on the device. Additionally, there are two methods to initially populate the picklist with teams: it can either be taken directly from the original team list of the event or from a copy of the current online picklist. In the past, a strategist could edit their offline picklist and then export their edited version with a password so that all other users would get their version of the picklist. However, we disabled this feature to make the Picklist Editor spreadsheet the single source for our picklist.

Elim Alliances

2023gal 5.0.0					FIELD	
<input type="checkbox"/>	1 3rd	1678	3476	461	Auto 71.00 Teleop 152.15	Charge 35.78 Total 208.75
<input type="checkbox"/>	2 3rd	1114	3005	836	Auto 68.06 Teleop 141.84	Charge 36.67 Total 202.76
<input type="checkbox"/>	3 3rd	103	2910	7461	Auto 50.80 Teleop 131.21	Charge 38.39 Total 182.01
<input type="checkbox"/>	4 3rd	8013	3467	2614	Auto 55.80 Teleop 128.20	Charge 32.89 Total 184.00
<input type="checkbox"/>	5 3rd	6800	67	6722	Auto 54.40 Teleop 145.52	Charge 40.89 Total 199.02
<input type="checkbox"/>	6 3rd	4451	4381	401	Auto 55.44 Teleop 118.31	Charge 32.28 Total 173.75
<input type="checkbox"/>	7 3rd	3539	1591	4481	Auto 54.44 Teleop 132.52	Charge 40.00 Total 186.97
<input type="checkbox"/>	8 3rd	7197	8033	1690	Auto 49.80 Teleop	Charge 40.75 Total

Elim Alliances

Elim Alliances was a new feature implemented to give further insight on playoff matches and alliances. It displays each alliance number and team numbers. During Champs, it shows both the 3rd pick and 4th pick alliances. The data is predicted scores for Auto, Teleop, Charge, and Total based on the averages of all the teams in the alliance. Users can also check the checkbox on the left hand side to cross out the teams as a visual reminder of which alliances have been eliminated. This list of eliminated alliances is stored to a file on the phone to save even when the app is reinstalled.

Data Refreshing

Data refreshes automatically by fetching the data and updating caches on a set time interval. When the data refreshes, callbacks are run in all active fragments in the backstacks which each provide their own implementations for updating the UI.



PICKLIST EDITOR

Overview

Picklist Editor is an automated spreadsheet that allows strategists to construct an informed picklist at competition based on our scouting data. It displays a list of teams along with raw and processed data for each robot (retrieved from a data export generated by Server at the end of a competition day), and automatically reorders teams based on an inputted rank on the picklist. Picklist Editor runs on Google Sheets using the Google Apps Script platform, and the code is written in TypeScript.

Team Rank Ordering

In the main editor, a list of the teams in the competition is shown in the first column, initially ordered by their “pickability”, based on either first or second pick rank scores. Pickability is displayed in the first two columns, but is typically hidden throughout our picklist meeting as it is only a starting place for the picklist order.

To reorder the teams, the picklist operator edits the ranking number in the “Order” column (e.g. to move a team in between first and second, change their rank to 1.5). Then, the sheet’s scripts will automatically reorder the teams and update their ranks to whole numbers. The sheet is designed for bubble sorting, or starting from the top and working downwards.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
3	1678	Order	First Pickability	Second Pickability	Driver Ability	Quickness	Field Aware	Avg Time Incap (s)	Max Tele Balls High	Tele Avg Balls High	Tele Avg Balls Low	Auto Avg Balls High	Auto Avg Balls Low	Climb Attempts	Low	Mid	High	Traversal
4	27574	59	6.5	63.9	4.6	2.0	1.9	0.0	27.0	22.0	0.1	4.6	0.0	6.0	0.0	1.0	0.0	5
5	28345	38	6.4	62.1	4.5	1.9	2.0	0.0	31.0	22.0	0.0	3.9	0.0	7.0	0.0	0.0	0.0	7
6	92997	45	4.7	49.9	3.8	1.6	1.7	0.0	19.0	15.7	0.0	3.8	0.0	4.0	0.0	2.0	2.0	0
7	19088	42	4.4	55.2	4.1	1.9	1.7	0.0	16.0	12.7	0.0	4.0	0.0	4.0	0.0	1.0	2.0	1
8	57578	55	3.9	46.5	3.5	1.4	1.7	0.0	21.0	12.0	0.0	2.3	0.2	6.0	0.0	0.0	0.0	5
9	66583	5	3.2	38.4	2.5	1.0	1.4	2.8	12.0	8.8	0.0	2.3	0.0	6.0	0.0	0.0	4.0	2
10	10586	13	3.1	43.5	2.6	1.3	1.2	0.0	10.0	6.6	0.0	2.6	0.0	6.0	0.0	0.0	0.0	6
11	34757	17	3.0	35.9	2.6	1.1	1.4	0.0	13.0	8.7	0.2	2.3	0.0	6.0	0.0	1.0	3.0	1
12	28477	25	3.0	43.0	3.8	1.8	1.6	0.0	12.0	9.3	0.3	1.9	0.0	6.0	0.0	2.0	5.0	0
13	11111	29	2.9	36.1	3.7	1.6	1.7	0.0	16.0	11.1	0.0	2.1	0.0	3.0	1.0	0.0	0.0	0
14	12313	52	2.9	44.0	3.3	1.3	1.7	0.0	10.0	7.0	0.0	2.3	0.0	6.0	1.0	0.0	2.0	3
15	56738	35	2.7	43.2	3.5	1.4	1.7	0.0	9.0	7.0	0.0	2.6	0.0	7.0	0.0	1.0	4.0	1
16	28388	1	2.6	39.5	2.7	1.3	1.3	0.0	10.0	6.0	0.0	1.8	0.0	5.0	0.0	0.0	2.0	3
17	28475	34	2.2	26.9	2.5	1.2	1.2	0.0	12.0	8.1	0.0	0.9	0.0	7.0	0.0	7.0	0.0	0
18	28475	23	2.2	26.9	2.5	1.2	1.2	0.0	12.0	8.1	0.0	0.9	0.0	7.0	0.0	7.0	0.0	0
19	48848	21	1.7	26.5	1.8	0.9	1.0	21.5	6.0	4.0	0.0	0.5	0.0	3.0	0.0	1.0	0.0	2
20	76937	22	1.7	23.3	1.7	0.9	0.9	5.6	6.0	3.8	0.0	0.4	0.4	5.0	0.0	1.0	1.0	2
21	23745	4	1.5	14.3	1.1	0.7	0.7	3.5	0.0	0.0	7.0	0.0	1.3	3.0	0.0	2.0	0.0	0
+ Main Editor			DNPs		Team Comparison Graphs		Settings	TIM Raw Data	Team Raw Data	Image Raw Data								

Main Editor Sheet



Updating Data Points

The data points displayed in the editor and their order often change based on feedback from those involved in picklist creation. Due to Picklist Editor's code structure and its deployment on Google Sheets, it was easy to add or remove data points in the middle of picklist creation, although strategists attempted to determine these fields beforehand to save time. Data Points can be added by inserting a new column and copying the appropriate formulas.

Removing Teams

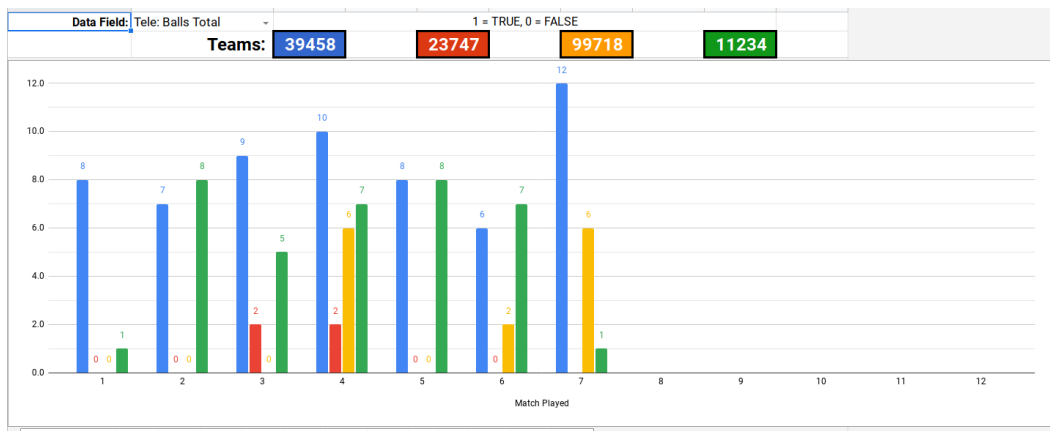
Before picklist creation, strategists and strategy mentors discuss teams that are not performing at the level they'd like for a potential alliance. This limits the number of teams to rank and saves time. A team is removed from the list on the main page by inputting "DNP" (Do Not Pick) in their cell in the "Order" column. In case the team is later reconsidered, the operator can send the team back by checking its box on the DNPs page.

Team Number	Click to send back
123456	<input type="checkbox"/>
111111	<input type="checkbox"/>
99999	<input type="checkbox"/>
100000	<input type="checkbox"/>
135790	<input type="checkbox"/>

DNP List

Team Comparison Graphs

When a match-by-match comparison between teams is necessary, the Picklist Editor's graphing feature can compare a single data point for up to four teams. The graphs can also show changes in data over multiple matches.



Team Comparison Graph

Operation

Picklist creation occurs during a meeting the night before alliance selections. In these meetings, the picklist operator displays the Picklist Editor through a projector.

First, we rank ourselves at the top of the list (regardless of our performance) since we cannot pick our own team. Then, starting with the initially top ranked team based on First Pickability, teams are compared on a pairwise basis progressing down the list—specifically, whether or not each team should be moved up the picklist and by how many places. Once potential first picks have been ranked (typically 10 or 12 teams), the rows containing those teams are hidden in the editor and the potential second picks are then ranked in the same manner. Once the re-ranking process is complete, a CSV export of the final picklist is sent back to the Server. At a regional, where qualification matches continue on the day of alliance selections, strategists make changes to the picklist based on further information and closer review of teams of interest.

Google Apps Script

The Picklist Editor is built on Google Apps Script, which is very 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—to develop Picklist Editor.

Robot Photos

Viewing pictures of a robot when discussing its rank often prompts other observations about the robot and its performance. This year, displaying robot photos in the sidebar of the Picklist Editor interface had major performance issues, hindering the effectiveness of the feature. These were difficult to fix due to the limitations of Google Apps Script, and our temporary solution for this season was to open the robot photos separately and manually find photos for each robot. We hope to find solutions for these issues by next season.



VIDEO SYSTEM

The Video System is a vital part of data collection and strategization, allowing strategists to review and rewatch matches. The match videos allow strategists to 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 while 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 USB drive. Last year, a script written in Python was used to rename and copy the videos, but the script added a level of complexity that was no longer needed.

During elimination matches, videos are put on a USB flash drive and given to match strategists, enabling them to develop a better strategy for the next match in little time.



CONCLUSION

System Accomplishments

This season, the Citrus Circuits scouting system was the most feature-rich, reliable, and effective it has ever been. Our system was crucial in constructing a great picklist at each competition we participated in and creating useful match strategies for every match. Our performance at every competition this season would not have been possible without the advantages provided by our scouting system.

Summary of Major Changes Since 2022

Throughout the 2023 season, we made many changes to ensure our apps were efficient and had all the necessary features. One major change was switching from Grosbeak to Cardinal as our web server to transfer data to the apps—due to a lack of documentation, it was difficult to maintain Cardinal, and we had already been using Grosbeak for Viewer’s picklist feature. Additionally, a new app was created for our Stand Strategists to record notes about an alliance and collect subjective data. They had previously used Google Sheets, which was unreliable due to the lack of stable WiFi. Additionally, we began integrating Jetpack Compose (Google’s modern toolkit for building Android UI) into our apps, as it is more maintainable, future-proof, and concise. The Stand Strategist app solely uses Kotlin and Compose. We also refined our Viewer app by adding complex features to our Team Details screen, including auto paths and comparison bars.

A need for structured playoff scouting—to optimize our playoff strategies and thus competitive ability—led us to create a new mode in Match Collection. During playoffs, one of our top scouts (based on Scout Precision rankings) and one strategist watched each robot on the field (6 scouts and 6 strategists total). The scouts used the Playoff Scouting mode of the app to record the number and location of both normal and Supercharged game pieces scored by the robot. The strategists took notes on their assigned robot, and at the end of the match, the scouts showed the strategists their collected data. The data was then posted in a Slack thread, and the match strategist and drive team used the data to create match strategies.

Lessons We Learned

Training

This season we learned how critical proper training is to a software development team’s success. Our new training presentations on Python and Kotlin, as well as a Mini-Scout training activity for our new Front-End members, allowed our new team members to efficiently learn the skills required to work on our apps.



Field Testing

During our first regional this season, our QR code scanners failed to work properly which was a major issue which could have been avoided with better field testing. To prevent this from happening again, a field testing checklist was created and multiple tests were run before every competition. Now, all system functions are timely and thoroughly evaluated in a way that mimics an actual competition setting.

Documentation

The importance of documentation was incredibly apparent this season. The lack of documentation for subteam processes and procedures caused unnecessary problems early on in the build season, from GitHub repository errors to breaking QR scanners. If certain team members with strong technical knowledge in a specific area were not present at a meeting, it often slowed down our subteam's workflow. Additionally, many conversations relevant to the entire subteam (e.g. discussing the purpose of certain data points) occurred in DMs, which made communication much less efficient and prevented the spread of knowledge across our subteam. In the future, we will work harder on properly documenting our processes, as well as communicating relevant information in public spaces.

Data Plans

At the World Championship this year, after working smoothly at all our previous competitions, the data plans for our phones were extremely inconsistent and unreliable. Heavy traffic at Champs, a lack of documentation on how we managed our data plans, and miscommunication regarding our data usage resulted in our Viewer app being unable to load data. In the future, we plan on getting a faster and larger data plan, creating contingency plans in case the data stops working, and better documenting the process of purchasing data.

Starting a Scouting System

We recommend teams to start with a small system structure; you can use a web app or paper and pencils—whichever is easiest for your team. Citrus Circuits has successfully used paper scouting and a Google Forms scouting system at offseason events to train new members in scouting principles. Next, prioritize training your scouts. A great method is to show old match videos, and as a group, discuss the strategies in the match. Some example guiding questions: Why is this alliance successful? What makes this robot good at defense? Why did this team captain choose this first and second pick and what makes this a strong alliance? Would you prioritize the same attributes?

If you are able to spare only a few members of your team for scouting, we recommend either having each one take qualitative notes on one alliance or reaching out to other teams and



gauge interest in forming a scouting alliance. With the latter, each team can obtain a copy of the data to review for their matches and picklist creation, and no single team needs a large group of scouts. If you are a 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 at all about starting your own scouting system, want our team's advice given your resource level, or have any other inquiries, please reach out to us at softwarescouting@citruscircuits.org, we are happy to help!

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 important to recognize our own shortcomings and address them during the offseason. We also plan on improving our new member training by creating more hands-on assignments and doing training on Tableau (Tableau is helpful for visualizing and analyzing scouting data). We are also considering implementing Zebra tagging into our system to further supplement our data for robot speed and positions throughout the match. Furthermore, we'd like to start phasing out XML for Jetpack Compose, because it allows for more complicated UI structures and is written in the same language as most of our apps.

There are many new features to be implemented, code to be cleaned, and processes to be streamlined. There's no "perfect system" and we'll continue to work hard to improve the one that we have.

Resources

Past Whitepapers

Our whitepapers from previous seasons can be found [on our team website](#).

Fall Workshops

Every year, we hold workshops to help students from other FRC teams learn skills necessary to grow competitively as a team. Our previous Fall Workshops can be found [on our team website](#).

GitHub Repos

Match Collection: <https://github.com/frc1678/match-collection-2023-public>

Pit Collection: <https://github.com/frc1678/pit-collection-2023-public>

Stand Strategist: <https://github.com/frc1678/stand-strategist-2023-public>



Server: <https://github.com/frc1678/server-2023-public>

Schema: <https://github.com/frc1678/schema-2023-public>

Grosbeak: <https://github.com/frc1678/grosbeak-2023-public>

Viewer: <https://github.com/frc1678/viewer-2023-public>

Contact Info

Email us at softwarescouting@citruscircuits.org with any questions or concerns you have about our Whitepaper and its contents, or if you would like us to provide more resources. We'd love to hear from you!



APPENDICES

Subteam Processes

Subteam Structure

Students on Software Scouting are split into either Back-End or Front-End, with each end having its own student lead. Back-End students code in Python and are responsible for creating and maintaining the Server. Front-End students mainly use Kotlin and create all the apps that users interact with, including Match Collection, Pit Collection, Stand Strategist, Viewer, and Picklist Editor. Each end enables members to specialize in specific programming knowledge and concepts. However, Software Scouting is still one subteam working together to create a unified system, and students on both ends regularly review each other's code and participate in full subteam discussions and system tests. Students are almost always paired to work together, as collaboration is critical to great software development. Additionally, the subteam works closely with Citrus Circuits' secondary subteam, the Strategy subteam, and most Software Scouting members are on the Strategy subteam.

How to Run a 1678 System Test

Before the test, collect match videos. Ideally, these are 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.

Create 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 (eg. 2023cada for the 2023 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 files.

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 scouting apps to collect data from match videos. If there are fewer than 20 students available, have people use multiple tablets at once. The main purpose of a field test is not to get accurate data, but rather to thoroughly test every feature of the system. The match videos are only there as a guide, so a team number assigned to a scout can be different from the robot they watch. 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 make sure data is updating and being displayed correctly.

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

Training

During the offseason, the first months are spent training new members. New members start by working with both of Citrus Circuits' software subteams (Software Robot and Software Scouting) to learn general software development concepts. These lessons include training on GitHub, creating pull requests to push their code, and code review standards. New members also learn the basics about what each subteam does 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 hands-on challenges the members must complete by utilizing the skills gained throughout the training. Once basic syntax training has finished, new Front-End members begin on their last training project: the Mini Scout. This was a training method developed in 2021 where new members create their own miniature version of our Match Collection app, following instructions that get less specific as they progress through the project—specifically, students create their own match starting screen, data input screen, and match data edit screen.

Back-End training begins with Python basics, with lessons taught through slideshows and assignments. Once syntax training is complete, new Back-End members learn about Schema and the MongoDB database. Additionally, Back-End members are walked through the Server and how data gets transferred and organized.

General subteam standards and tips are taught to all Software Scouting students. These lessons include training on GitHub and Git, review standards, and principles of the system.

Scout Training & Management

The week before each competition, scouts are trained on how to collect accurate data. Objective Scouts are trained by a Lead Scout and Assistant Lead Scout who are in charge of organizing scouts during competitions. Note that the Lead Scout role is different from a Software Scouting Lead; the Lead Scout does not necessarily have to be on the Software Scouting subteam and is responsible for stands management at competition (see Section 6.2 of our [Team Handbook](#) for more information). The training begins with an explanation of behavior standards at the competition and an overview of the itinerary. Afterward, scouts



participate in a Kahoot, quizzing members on possible scouting scenarios to ensure consistent data. Scouts then spend at least 2 hours watching recent matches, discussing the purpose of the data they're collecting and how it contributes to strategies observed in the matches.

Subjective Scouts are trained by our strategy mentors and Match Strategist, and practice by watching matches while discussing their rankings with the mentor. Once the scout becomes more confident with their rankings, they collect data on their own with no input from the mentors and get feedback after deciding on rankings.

Video System, Pit Collection, and Stand Strategist users are trained by members of Software Scouting that have experience with the collection process. This training is informal and the knowledge is passed down between users. We do not have specific training for Viewer users as they are experienced strategists, but we create a short user guide updated with each year's features.

Typically, 21 Objective and 3 Subjective Scouts are brought to each competition. Breaks are set by the Lead Scout ahead of time to ensure equity, typically 30-minutes at a time. During this time, scouts can go to the pits, meet new teams, and explore the venue. Subjective Scouts organize their own breaks. An template for a scout schedule can be found here:

[!\[\]\(e78f798d4ea5c530c9db49e7d26e6b95_img.jpg\) PUBLIC Scout Organization Template](#) .

Each scout has a Scout ID that is used to ensure that each team is being scouted by 3 people. This Scout ID is preassigned to Objective Scouts but will change for a scout that returns from a break. We utilize Scout ID count-offs to ensure that all 18 Objective Scouts are active and have different IDs.

In order to ensure a positive culture around scouting, we present to the entire team before competitions begin, explaining the purpose of scouting and its role in our competitive success. It is critical that scouts understand the responsibility they have and how vital their efforts are during competition. In the stands, scouts can have fun demonstrating their team spirit in fruit-themed costumes, and the Lead Scout brings scout-only candy!

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 entire process is repeated for all scouts in that match to find their average combination errors.

Then, the formula returns to look at each individual 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.

Season Analysis

Timeline

Prior to 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 take inventory of all materials before each season and order any needed supplies.

1/7/2023 — 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.

1/8/2023 — The Scouting subteam meets with Strategy members to determine what data points are necessary to collect and which we want to display. For each data point on the final list, the following information is noted down:

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



- Whether it would be collected raw, or if it needed to be calculated from other data points
 - If it was a raw data point, how it would be collected (by Objective Scouts, Subjective Scouts, Pit Scouts, Stand Strategists, or The Blue Alliance API)
- If it would be displayed in the Viewer, and if so, how it would be visualized

1/8/2023 to 1/15/2023 — Back-End students update the Schema files to contain the new data points for the new season. Front-End students update the Match Collection and Pit Collection apps with new data and UI designs.

1/15/2023 to 1/30/2023 — Back-End students updated 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 data points. The visual apps are also often worked on at the same time as the collection apps to prevent conflicts between code merges.

1/30/2023 to 2/27/2023 — Software Scouting begins to conduct end-to-end field tests to ensure the system runs together. Front-End collects user feedback on the UI of apps. The list of data points to calculate/collect is updated based on strategy discussions. Students watch xRC Simulator matches to test out the scouting system before real match videos are available. Coming closer to competition season, Objective Scouts and Subjective Scouts are trained on how to use the Match Collection app.

2/27/2023 to 4/23/2023 — During competition season, the Saturday before each competition is a feature freeze, a deadline that completely stops development on all new features until after the competition. The subteam runs a full-system test before every competition in order 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 in order to prioritize which changes to make before the next competition.

4/23/2023 to 5/23/2023 — Members work on the Whitepaper and code cleanup to get ready for public release.

Competition Roles

Roles at competition are broken up 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 — 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 also is in charge of uploading scanned data to the MongoDB cloud after



each match. Two Objective Scouts also serve as backup developers who primarily scout but can assist when needed.

Scouts — Since 18 Objective Scouts are needed per match, about 21 scouts travel to each competition in order 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 Subjective Scouts are required to have extensive experience on the Strategy subteam, the student on break often chooses to help other strategists. The Lead Scout manages all of the logistics for the scouts: training, meals, shifts, handing out tablets, and anything else they might need.

Operators — One student who is usually on Software Scouting and has experience in programming Picklist Editor operates the spreadsheet during picklist meetings. Additionally, two Video System operators record, name, save, and send match videos to strategists. Video System operators are Objective Scouts while recording videos.

Strategists — Two Stand Strategists write specific notes on each team while watching matches and collect needed subjective data. At regionals, Stand Strategists edit the picklist depending on each team's performance. On the second day of 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. All of our Strategists are often veteran members of the Strategy subteam and have a keen eye for strategy.

Other Information

Hardware

This scouting system requires the use of multiple different pieces of hardware in order to ensure it runs smoothly and efficiently. For our apps, we use just over 40 tablets for Match Collection and four Android phones for Pit Collection and the Viewer app. For data transfer, we use three QR scanners that transfer data from the tablets directly to a single laptop which runs the Server. All hardware is packed into five cases: two tablet cases, a Server case, a Video System case, and a Gray Tote for extra space. More specific details about the cases and our hardware can be found in section 4.2 of our [2020 Whitepaper](#).

Pickability Weights

This year, first pickability weights remained constant throughout the season. The most accurate way to measure a robot's offensive capabilities proved to be the expected amount of points scored by that robot every match. Thus, first pickability was the average total points scored by a robot.



However, second pickability was a different story. As there are multiple different auto configurations we could run, multiple second pickabilities were used to evaluate robots depending on the auto configurations we and our first pick would run.

Here are the metrics and weights used for the second pickabilities at Champs:

3-Bump Auto		3-Clean		2-Bump Auto	
Metric	Weight	Metric	Weight	Metric	Weight
Tele Game Pieces Scored	3	Tele Game Pieces Scored	3	Tele Game Pieces Scored	3
Auto Mobility	3	Auto Mobility	3	Auto Mobility	3
Tele Dock%	6	Tele Dock%	6	Tele Dock%	6
Tele Engage%	10	Tele Engage%	10	Tele Engage%	10
Auto Game Pieces Scored	4 (Capped)	Auto Game Pieces Scored (Middle)	4 (Capped)	Auto Game Pieces Scored	4
Middle-Engage (Engage Rate)	12	Middle-Engage (Engage Rate)	12	Middle-Engage (Engage Rate)	12
Middle-Engage (Dock Rate)	8	Middle-Engage (Dock Rate)	8	Middle-Engage (Dock Rate)	8
		2-bump	12		
		1-bump-mobility	9.5		
		Auto Game Pieces Bump	4		

Each second pickability name, for example 3-Bump Auto, stands for each path that our first pick/captain robot would run in a match, and the 3 stands for the number of game pieces scored. The weights were chosen in order to find a second pick that could allow us to max out our auto points (7 game pieces scored, mobility, and an engage). The 3-Clean auto is quite different from the other two, in that there could be two different autos our robot could run (2 over the center or 2 over the bump) if our first pick/captain was running a 3-Clean. Therefore, the middle and bump paths must be tested for our second pick since they could run a 1-middle, 2-middle, 1-bump, or 2-bump. Whichever path returns the highest auto points is used.

Codebook

Team in Match Data Set

DATA POINT	DATA TYPE	DESCRIPTION
confidence_rating	Integer	The number of scouts who scouted a robot.
auto_cube_low	Integer	The number of cubes scored in the low row during auto.



auto_cube_mid	Integer	The number of cubes scored in the mid row during auto.
auto_cube_high	Integer	The number of cubes scored in the high row during auto.
auto_cone_low	Integer	The number of cones scored in the low row during auto.
auto_cone_mid	Integer	The number of cones scored in the mid row during auto.
auto_cone_high	Integer	The number of cones scored in the high row during auto.
auto_charge_attempt	Integer	The number of times a robot attempted to charge in auto.
tele_cube_low	Integer	The number of cubes scored in the low row during teleop.
tele_cube_mid	Integer	The number of cubes scored in the mid row during teleop.
tele_cube_high	Integer	The number of cubes scored in the high row during teleop.
tele_cone_low	Integer	The number of cones scored in the low row during teleop.
tele_cone_mid	Integer	The number of cones scored in the mid row during teleop.
tele_cone_high	Integer	The number of cones scored in the high row during teleop.
tele_charge_attempt	Boolean	The number of times a robot attempted to charge in teleop.
intakes_ground	Integer	The number of game pieces taken in from the ground.
intakes_double	Integer	The number of game pieces intaken from the double substation.
intakes_single	Integer	The number of game pieces intaken directly from the single substation.
intakes_low_row	Integer	The number of gamepieces intook from the low row.
intakes_mid_row	Integer	The number of gamepieces intook from the mid row.
intakes_high_row	Integer	The number of gamepieces intook from

		the high row.
failed_scores	Integer	The number of game pieces taken in but not scored.
auto_total_cubes	Integer	The number of cubes scored during auto.
auto_total_cones	Integer	The number of cones scored during auto.
auto_total_gamepieces	Integer	The number of game pieces scored during auto.
auto_total_gamepieces_low	Integer	The number of game pieces scored in the low row during auto.
tele_total_cubes	Integer	The number of cubes scored during teleop.
tele_total_cones	Integer	The number of cones scored during teleop.
tele_total_gamepieces	Integer	The number of game pieces scored during teleop.
tele_total_gamepieces_low	Integer	The number of game pieces scored in the low row during teleop.
total_intakes	Integer	The number of game pieces intaken during the match.
total_charge_attempts	Integer	Number of times a team attempted to charge in a match.
incap	Integer	The amount of time the robot was incapacitated for.
median_cycle_time	Integer	The median of the team's cycle times.
auto_charge_level	String	Level a team charged in auto: N for none, D for docked, E for engaged.
tele_charge_level	String	Level a team charged in tele: N for none, P for parked, D for docked, E for engaged.
start_position	String	The starting position of the robot on the field.
preloaded_gamepiece	String	The type of game piece the robot was preloaded with.

Objective Team Data Set

DATA POINT	DATA TYPE	DESCRIPTION
auto_avg_cone_high	Float	Average number of cones scored by the team in the high row in auto.
auto_avg_cone_mid	Float	Average number of cones scored by the team in the middle row in auto.
auto_avg_cone_low	Float	Average number of cones scored by the team in the low row in auto.
auto_avg_cone_total	Float	Average number of cones scored by the team in auto.
auto_avg_cube_high	Float	Average number of cubes scored by the team in the high row in auto.
auto_avg_cube_mid	Float	Average number of cubes scored by the team in the middle row in auto.
auto_avg_cube_low	Float	Average number of cubes scored by the team in the low row in auto.
auto_avg_cube_total	Float	Average number of cubes scored by the team in auto.
auto_avg_gamepieces	Float	Average number of game pieces scored by the team in auto.
tele_avg_cone_high	Float	Average number of cones scored by the team in the high row in teleop.
tele_avg_cone_mid	Float	Average number of cones scored by the team in the middle row in teleop.
tele_avg_cone_low	Float	Average number of cones scored by the team in the low row in teleop.
tele_avg_cone_total	Float	Average number of cones scored by the team in teleop.
tele_avg_cube_high	Float	Average number of cubes scored by the team in the high row in teleop.
tele_avg_cube_mid	Float	Average number of cubes scored by the team in the middle row in teleop.
tele_avg_cube_low	Float	Average number of cubes scored by the team in the low row in teleop.
tele_avg_cube_total	Float	Average number of cubes scored by the team in teleop.



tele_avg_gamepieces	Float	Average number of game pieces scored by the team in teleop.
auto_avg_gamepieces_low	Float	Average number of game pieces scored by the team in the low row in auto.
tele_avg_gamepieces_low	Float	Average number of game pieces scored by the team in the low row in teleop.
avg_incap_time	Float	Average amount of time the team's robot was incap.
avg_intakes_ground	Float	Average number of times the team intook a game piece from the ground.
avg_intakes_double	Float	Average number of times the team intook a game piece from the double substation.
avg_intakes_single	Float	Average number of times the team intook a game piece directly from the single substation.
avg_intakes_low_row	Float	Average number of times the team intook a game piece from the low row.
avg_intakes_mid_row	Float	Average number of times the team intook a game piece from the middle row.
avg_intakes_high_row	Float	Average number of times the team intook a game piece from the high row.
avg_total_intakes	Float	Average number of times the team intook a game piece.
avg_failed_scores	Float	Average number of times the team failed to score a game piece after intaking it.
lfm_auto_avg_cone_high	Float	Average number of cones scored by the team in the high row in auto in the last four matches.
lfm_auto_avg_cone_mid	Float	Average number of cones scored by the team in the middle row in auto in the last four matches.
lfm_auto_avg_cone_low	Float	Average number of cones scored by the team in the low row in auto in the last four matches.
lfm_auto_avg_cone_total	Float	Average number of cones scored by the team in auto in the last four matches.
lfm_auto_avg_cube_high	Float	Average number of cubes scored by the

		team in the high row in auto in the last four matches.
lfm_auto_avg_cube_mid	Float	Average number of cubes scored by the team in the middle row in auto in the last four matches.
lfm_auto_avg_cube_low	Float	Average number of cubes scored by the team in the low row in auto in the last four matches.
lfm_auto_avg_cube_total	Float	Average number of cubes scored by the team in auto in the last four matches.
lfm_auto_avg_gamepieces	Float	Average number of game pieces scored by the team in auto in the last four matches.
lfm_tele_avg_cone_high	Float	Average number of cones scored by the team in the high row in teleop in the last four matches.
lfm_tele_avg_cone_mid	Float	Average number of cones scored by the team in the middle row in teleop in the last four matches.
lfm_tele_avg_cone_low	Float	Average number of cones scored by the team in the low row in teleop in the last four matches.
lfm_tele_avg_cone_total	Float	Average number of cones scored by the team in teleop in the last four matches.
lfm_tele_avg_cube_high	Float	Average number of cubes scored by the team in the high row in teleop in the last four matches.
lfm_tele_avg_cube_mid	Float	Average number of cubes scored by the team in the middle row in teleop in the last four matches.
lfm_tele_avg_cube_low	Float	Average number of cubes scored by the team in the low row in teleop in the last four matches.
lfm_tele_avg_cube_total	Float	Average number of cubes scored by the team in teleop in the last four matches.
lfm_tele_avg_gamepieces	Float	Average number of game pieces scored by the team in teleop in the last four matches.
lfm_auto_avg_gamepieces_low	Float	Average number of game pieces scored by the team in the low row in auto in the last four matches.

lfm_tele_avg_gamepieces_low	Float	Average number of game pieces scored by the team in the low row in teleop in the last four matches.
lfm_avg_intakes_ground	Float	Average number of times the team intook a game piece from the ground in the last four matches.
lfm_avg_intakes_double	Float	Average number of times the team intook a game piece from the double substation in the last four matches.
lfm_avg_intakes_single	Float	Average number of times the team intook a game piece directly from the single substation in the last four matches.
lfm_avg_intakes_low_row	Float	Average number of times the team intook a game piece from the low row in the last four matches.
lfm_avg_intakes_mid_row	Float	Average number of times the team intook a game piece from the middle row in the last four matches.
lfm_avg_intakes_high_row	Float	Average number of times the team intook a game piece from the high row in the last four matches.
lfm_avg_total_intakes	Float	Average number of times the team intook a game piece in the last four matches.
lfm_avg_incap_time	Float	Average amount of time the team's robot was incap in the last four matches.
lfm_avg_failed_scores	Float	Average number of times the team failed to score a game piece after intaking it in the last four matches.
auto_sd_cone_high	Float	Standard deviation of cones scored in the high row during auto.
auto_sd_cone_mid	Float	Standard deviation of cones scored in the mid row during auto.
auto_sd_cone_low	Float	Standard deviation of cones scored in the low row during auto.
auto_sd_cone_total	Float	Standard deviation of cones scored during auto.
auto_sd_cube_high	Float	Standard deviation of cubes scored in the high row during auto.

auto_sd_cube_mid	Float	Standard deviation of cubes scored in the mid row during auto.
auto_sd_cube_low	Float	Standard deviation of cubes scored in the low row during auto.
auto_sd_cube_total	Float	Standard deviation of cubes scored during auto.
auto_sd_gamepieces	Float	Standard deviation of game pieces scored during auto.
lfm_auto_sd_gamepieces	Float	Standard deviation of game pieces scored during auto in the last four matches.
tele_sd_cone_high	Float	Standard deviation of cones scored in the high row during teleop.
tele_sd_cone_mid	Float	Standard deviation of cones scored in the mod row during teleop.
tele_sd_cone_low	Float	Standard deviation of cones scored in the low row during teleop.
tele_sd_cone_total	Float	Standard deviation of cones scored during teleop.
tele_sd_cube_high	Float	Standard deviation of cubes scored in the high row during teleop.
tele_sd_cube_mid	Float	Standard deviation of cubes scored in the mid row during teleop.
tele_sd_cube_low	Float	Standard deviation of cubes scored in the low row during teleop.
tele_sd_cube_total	Float	Standard deviation of cubes scored during teleop.
tele_sd_gamepieces	Float	Standard deviation of game pieces scored during teleop.
lfm_tele_sd_gamepieces	Float	Standard deviation of game pieces scored during teleop in the last four matches.
auto_charge_attempts	Integer	Number of times the team attempted to charge during auto.
auto_dock_successes	Integer	Number of times the team succeeded in at least docking during auto.
auto_dock_only_successes	Integer	Number of times the team only succeeded in docking during auto.

auto_engage_successes	Integer	Number of times the team succeeded in engaging during auto.
tele_charge_attempts	Integer	Number of times the team attempted to charge during teleop.
tele_dock_successes	Integer	Number of times the team succeeded in at least docking during teleop.
tele_dock_only_successes	Integer	Number of times the team only succeeded in docking during teleop.
tele_engage_successes	Integer	Number of times the team succeeded in engaging during teleop.
tele_park_successes	Integer	Number of times the team succeeded in parking during teleop.
position_zero_starts	Integer	Number of times the team started a match in position zero.
position_one_starts	Integer	Number of times the team started a match in position one.
position_two_starts	Integer	Number of times the team started a match in position two.
position_three_starts	Integer	Number of times the team started a match in position three.
position_four_starts	Integer	Number of times the team started a match in position four.
matches_incap	Integer	Number of matches the team's robot was incap in at least once.
matches_played	Integer	Number of matches the team has played.
lfm_position_zero_starts	Integer	Number of times the team started a match in position zero in the last four matches.
lfm_position_one_starts	Integer	Number of times the team started a match in position one in the last four matches.
lfm_position_two_starts	Integer	Number of times the team started a match in position two in the last four matches.
lfm_position_three_starts	Integer	Number of times the team started a match in position three in the last four matches.



lfm_position_four_starts	Integer	Number of times the team started a match in position four in the last four matches.
lfm_auto_charge_attempts	Integer	Number of times the team attempted to charge during auto in the last 4 matches.
lfm_auto_dock_successes	Integer	Number of times the team succeeded in at least docking during auto in the last four matches.
lfm_auto_dock_only_successes	Integer	Number of times the team only succeeded in docking during auto in the last four matches.
lfm_auto_engage_successes	Integer	Number of times the team succeeded in engaging during auto in the last four matches.
lfm_tele_charge_attempts	Integer	Number of times the team attempted to charge during teleop in the last four matches.
lfm_tele_dock_successes	Integer	Number of times the team succeeded in at least docking during teleop in the last four matches.
lfm_tele_dock_only_successes	Integer	Number of times the team only succeeded in docking during teleop in the last four matches.
lfm_tele_engage_successes	Integer	Number of times the team succeeded in engaging during teleop in the last four matches.
lfm_tele_park_successes	Integer	Number of times the team succeeded in parking during teleop in the last four matches.
lfm_matches_incap	Integer	Number of matches the team's robot was incap in in the last four matches.
matches_tippy	Integer	Number of matches the team's robot was marked as "tippy".
lfm_matches_tippy	Integer	Number of matches the team's robot was marked as "tippy" in the last four matches.
matches_played_defense	Integer	Number of matches the team played defense.
lfm_matches_played_defense	Integer	Number of matches the team played defense in the last four matches.

auto_max_cone_high	Integer	The maximum number of cones the team scored in the high row during auto.
auto_max_cone_mid	Integer	The maximum number of cones the team scored in the middle row during auto.
auto_max_cone_low	Integer	The maximum number of cones the team scored in the low row during auto.
auto_max_cones	Integer	The maximum number of cones the team scored during auto.
auto_max_cube_high	Integer	The maximum number of cubes the team scored in the high row during auto.
auto_max_cube_mid	Integer	The maximum number of cubes the team scored in the middle row during auto.
auto_max_cube_low	Integer	The maximum number of cubes the team scored in the low row during auto.
auto_max_cubes	Integer	The maximum number of cubes the team scored during auto.
auto_max_gamepieces	Integer	The maximum number of game pieces the team scored during auto.
tele_max_cone_high	Integer	The maximum number of cones the team scored in the high row during teleop.
tele_max_cone_mid	Integer	The maximum number of cones the team scored in the middle row during teleop.
tele_max_cone_low	Integer	The maximum number of cones the team scored in the low row during teleop.
tele_max_cones	Integer	The maximum number of cones the team scored during teleop.
tele_max_cube_high	Integer	The maximum number of cubes the team scored in the high row during teleop.
tele_max_cube_mid	Integer	The maximum number of cubes the team scored in the middle row during teleop.
tele_max_cube_low	Integer	The maximum number of cubes the team scored in the low row during teleop.

tele_max_cubes	Integer	The maximum number of cubes the team scored during teleop.
tele_max_gamepieces	Integer	The maximum number of gamepieces the team scored during teleop.
max_incap	Integer	The maximum time the team's robot was incap for.
max_auto_charge_level	String	The maximum charge level the team has achieved during auto.
max_tele_charge_level	String	The maximum charge level the team has achieved during teleop.
lfm_auto_max_cone_high	Integer	The maximum number of cones the team scored in the high row during auto in the last four matches.
lfm_auto_max_cone_mid	Integer	The maximum number of cones the team scored in the middle row during auto in the last four matches.
lfm_auto_max_cone_low	Integer	The maximum number of cones the team scored in the low row during auto in the last four matches.
lfm_auto_max_cones	Integer	The maximum number of cones the team scored during auto in the last four matches.
lfm_auto_max_cube_high	Integer	The maximum number of cubes the team scored in the high row during auto in the last four matches.
lfm_auto_max_cube_mid	Integer	The maximum number of cubes the team scored in the middle row during auto in the last four matches.
lfm_auto_max_cube_low	Integer	The maximum number of cubes the team scored in the low row during auto in the last four matches.
lfm_auto_max_cubes	Integer	The maximum number of cubes the team scored during auto in the last four matches.
lfm_auto_max_gamepieces	Integer	The maximum number of game pieces the team scored during auto in the last four matches.
lfm_tele_max_cone_high	Integer	The maximum number of cones the team scored in the high row during teleop in the last four matches.

lfm_tele_max_cone_mid	Integer	The maximum number of cones the team scored in the middle row during teleop in the last four matches.
lfm_tele_max_cone_low	Integer	The maximum number of cones the team scored in the low row during teleop in the last four matches.
lfm_tele_max_cones	Integer	The maximum number of cones the team scored during teleop in the last four matches.
lfm_tele_max_cube_high	Integer	The maximum number of cubes the team scored in the high row during teleop in the last four matches.
lfm_tele_max_cube_mid	Integer	The maximum number of cubes the team scored in the middle row during teleop in the last four matches.
lfm_tele_max_cube_low	Integer	The maximum number of cubes the team scored in the low row during teleop in the last four matches.
lfm_tele_max_cubes	Integer	The maximum number of cubes the team scored during teleop in the last four matches.
lfm_tele_max_gamepieces	Integer	The maximum number of gamepieces the team scored during teleop in the last four matches.
lfm_max_incap	Integer	The maximum time the team's robot was incap for in the last four matches.
lfm_max_auto_charge_level	String	The maximum charge level the team has achieved during auto in the last four matches.
lfm_max_tele_charge_level	String	The maximum charge level the team has achieved during teleop in the last four matches.
mode_preloaded_gamepiece	List	Most common preloaded game piece.
mode_start_position	List	Most common start position.
lfm_mode_start_position	List	Most common start position in the last four matches.
mode_auto_charge_level	List	Most common charge state during auto.
lfm_mode_auto_charge_level	List	Most common charge state during auto in the last four matches.

mode_tele_charge_level	List	Most common charge state during teleop.
lfm_mode_tele_charge_level	List	Most common charge state during teleop in the last four matches.
charge_percent_success	Float	The percentage of auto and teleop successful charge attempts vs fails.
lfm_charge_percent_success	Float	The percentage of auto and teleop successful charge attempts vs fails in the last four matches.
auto_dock_percent_success	Float	The percentage of auto successful dock attempts vs fails.
lfm_auto_dock_percent_success	Float	The percentage of auto successful dock attempts vs fails in the last four matches.
auto_engage_percent_success	Float	The percentage of auto successful engage attempts vs fails.
lfm_auto_engage_percent_success	Float	The percentage of auto successful engage attempts vs fails in the last four matches.
tele_dock_percent_success	Float	The percentage of teleop successful dock attempts vs fails.
lfm_tele_dock_percent_success	Float	The percentage of teleop successful dock attempts vs fails in the last four matches.
tele_engage_percent_success	Float	The percentage of teleop successful engage attempts vs fails.
lfm_tele_engage_percent_success	Float	The percentage of teleop successful engage attempts vs fails in the last four matches.
tele_park_percent_success	Float	The percentage of matches parked where they didn't dock or engage.
lfm_tele_park_percent_success	Float	The percentage of matches parked where they didn't dock or engage in the last four matches.
tele_dock_only_percent_success	Float	The percentage of matches where they docked without engaging in teleop.
lfm_tele_dock_only_percent_success	Float	The percentage of matches where they docked without engaging in teleop in the last four matches.

auto_dock_only_percent_success	Float	The percentage of matches where they docked without engaging in auto.
lfm_auto_dock_only_percent_success	Float	The percentage of matches where they docked without engaging in auto in the last four matches.
median_nonzero_incap	Float	The median of the team's nonzero incap times.
lfm_median_nonzero_incap	Float	The median of the team's nonzero incap times in the last four matches.
tele_avg_charge_points	Float	The average number of points the team has received by charging during teleop.
lfm_tele_avg_charge_points	Float	The average number of points the team has received by charging during teleop in the last four matches.
auto_avg_charge_points	Float	The average number of points the team has received by charging during auto.
lfm_auto_avg_charge_points	Float	The average number of points the team has received by charging during auto in the last four matches.
auto_avg_total_points	Float	The average number of points the team has scored in auto.
tele_avg_total_points	Float	The average number of points the team has scored in teleop.
avg_total_points	Float	The average number the team has scored per game.
total_incap	Integer	The total amount of time the team was incap for.
lfm_total_incap	Integer	The total amount of time the team was incap for in the last four matches.

Predicted Team Data Set

DATA POINT	DATA TYPE	DESCRIPTION
current_rank	Integer	The team's current rank.
current_rps	Integer	How many Ranking Points the team currently has.
current_avg_rps	Float	The team's current average Ranking



		Points.
predicted_rps	Float	The number of RP points predicted.
predicted_rank	Integer	The predicted rank at the end of qualification matches.

Pickability Data Set

DATA POINT	DATA TYPE	DESCRIPTION
first_pickability	Float	Weighted average for first pickability.
second_pickability	Float	Weighted average for second pickability.

Predicted Alliance in Match Data Set

DATA POINT	DATA TYPE	DESCRIPTION
alliance_color_is_red	Boolean	Whether or not the alliance being predicted for is the red alliance in the match.
has_actual_data	Boolean	If the robot has actual data. Set to true when TBA has data.
actual_score	Integer	The actual score of the alliance.
actual_rp1	Float	If the Charge Station bonus was achieved. 1 if yes, 0 if no.
actual_rp2	Float	If the Link bonus was achieved. 1 if yes, 0 if no.
won_match	Boolean	If the alliance actually won.
predicted_score	Float	An alliance's predicted match score.
predicted_rp1	Float	The chance of the Charge Station bonus being achieved.
predicted_rp2	Float	The chance of the Link bonus being achieved.
win_chance	Float	The estimated probability of the alliance winning the match.

Scout in Match Precision Data Set

DATA POINT	DATA TYPE	DESCRIPTION
scout_name	String	The name of the scout.
alliance_color_is_red	Boolean	If the alliance color is red.
sim_precision	Float	The calculated points a scout is off by.
auto_cone_low_precision	Float	The calculated auto cones low row a scout is off by.
tele_cone_low_precision	Float	The calculated teleop cones low row a scout is off by.
auto_cone_mid_precision	Float	The calculated auto cones mid row a scout is off by.
tele_cone_mid_precision	Float	The calculated teleop cones mid row a scout is off by.
auto_cone_high_precision	Float	The calculated auto cones high row scout is off by.
tele_cone_high_precision	Float	The calculated teleop cones high row a scout is off by.
auto_cube_low_precision	Float	The calculated auto cubes low row a scout is off by.
tele_cube_low_precision	Float	The calculated teleop cubes low row a scout is off by.
auto_cube_mid_precision	Float	The calculated auto cubes mid row a scout is off by.
tele_cube_mid_precision	Float	The calculated teleop cubes mid row a scout is off by.
auto_cube_high_precision	Float	The calculated auto cubes high row a scout is off by.
tele_cube_high_precision	Float	The calculated teleop cubes high row a scout is off by.

Scout Precision Data Set

DATA POINT	DATA TYPE	DESCRIPTION
scout_name	String	The name of the scout.



scout_precision	Float	The average number of points a scout is off by.
-----------------	-------	---

TBA Team in Match Data Set

DATA POINT	DATA TYPE	DESCRIPTION
mobility	Boolean	If the robot achieved mobility.

TBA Team Data Set

DATA POINT	DATA TYPE	DESCRIPTION
team_name	String	The name of the team.
foul_cc	Integer	Calculated Contribution (OPR) for fouls.
link_cc	Integer	Calculated Contribution (OPR) for links.
mobility_successes	Integer	Total matches in which the robot achieved mobility.
lfm_mobility_successes	Integer	Number of matches in the last 4 matches in which the robot achieved mobility.

Objective Pit Data Set

DATA POINT	DATA TYPE	DESCRIPTION
drivetrain	Enum[str]	The type of drivetrain used.
drivetrain_motors	Integer	The number of motors used in the drivetrain.
drivetrain_motor_type	Enum[str]	The type of motors that are used in the drivetrain.
has_vision	Boolean	If the robot has vision.
has_communication_device	Boolean	If the robot has a method of communicating with a human player.
weight	Float	The weight of the robot.
length	Float	The length of the robot.



width	Float	The width of the robot.
-------	-------	-------------------------

Subjective Team Data Set

DATA POINT	DATA TYPE	DESCRIPTION
driver_field_awareness	Float	Z-score of field awareness.
driver_quickness	Float	Z-score of driver awareness.
driver_ability	Float	Z-score of driver ability.
auto_pieces_start_position	List	A list of four items, one for each auto game piece start position. 0 for cone, 1 for cube, 2 for none.

