

# Tourney Track

## 6.170 Final Project Design Document

Team Members: Jamar Brooks, Yachi Chang, Chris Rogers, Katharine Xiao

### I. Design Overview

System Overview

Purposes

Easy Tournament Management

Easy Access to Accurate Tournament Information

Context Diagram

### II. Design Model

Data Model and Concepts

### III. Design Behavior

Security Concerns

Policy

Requirements

Non-Requirements

Threat Model/What the attacker can do + Mitigations

User Interface

Action Flow and State Transitions:

Wireframes (for MVP)

Login Page ("login.html", "sign\_up.html")

User Profile Page ("profile.html")

Team Profile Page

New Tournament Page

All Tournaments Page

Tournament Page

New Team Page

Edit Tournament Page

Tournament Modal Page (for result approval)

Match Page

Match Outcome Modal Page

Design Challenges

How do we keep track of match outcome data?

How do we protect the integrity of match data entered into our system?

Should Teams be within the context of a tournament?

Should Teams be immutable?

Should players be allowed to be part of multiple Teams within a Tournament?

Implementation Challenges

How to keep track of references in the DB to meet the needs of the UI

Maintaining the logged in user from page to page

Presenting the pending outcome submissions to the tournament admin

# I. Design Overview

## System Overview

TourneyTrack is a web application that organizes small to medium scale tournaments, such as intramurals and dorm tournaments. An administrator can create tournaments with multiple brackets, either of elimination or round robin type, and players can create teams to play in various tournaments. Our app would then create a clear visualization of tournament status, schedule, and stats, allowing players to easily determine who they should play next and admins to easily manage the big picture.

## Purposes

### Easy Tournament Management

Creating and maintaining tournaments is often difficult and unorganized, especially for informal tournaments such as dorm or intramural level tournaments. The main solutions to this problem are:

- Automated tournament match making
  - Creating matches between players can be cumbersome, since it requires keeping track of individual match outcomes. Our app allows an organizer to be able to specify the type of tournament and its participants, and the system then generates all matches of the tournament.
  - Tournaments are automatically updated and advanced when the results of a match are approved
- Low-latency match outcome updates
  - Often, especially with informal tournaments, match outcomes are reported via email. This creates quite a bit of latency, because it requires the admin to check his/her email and update the tournament spreadsheet accordingly.
  - Our app allows the outcomes of a match to be viewable shortly after the completion of that match.

### Easy Access to Accurate Tournament Information

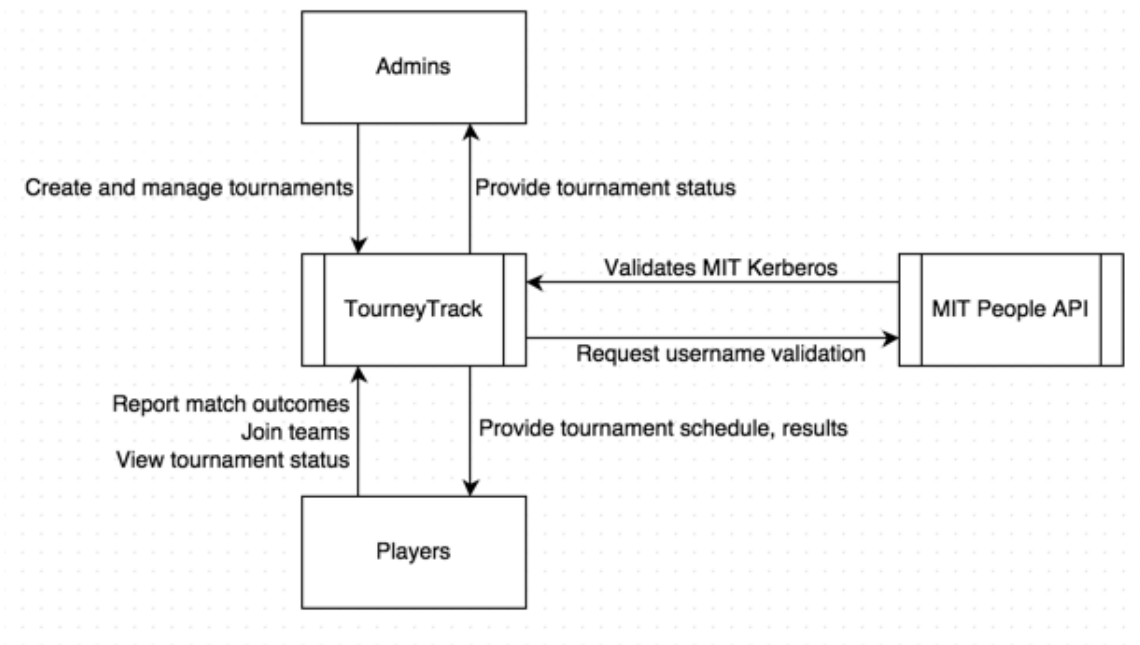
A big problem many participants of informal tournaments face is that big-picture information about the tournament is not immediately clear. Often, participants can only see outcomes to individual matches, making it hard to visualize the standings of all participants in the overall tournament. The following exhibit solutions to this problem:

- Automatically updated team-level statistics
  - Teams should be able to view how they are performing over the course of a given tournament. This information should be available immediately, once match outcome data has been entered into the system.

- Many solutions provide too many statistics (too fine-grained or too coarse-grained), which causes the stats that many users care about to be lost in a sea of information.
- Big-picture match schedule visualization
  - Users should be able to quickly view a tournament and see what matches they participate in, and how these matches relate to other matches in the tournament hierarchy.
  - Many existing solutions do not provide an intuitive representation of the matches for a tournament.

## Context Diagram

Our context diagram is as follows:



<https://www.draw.io/#G0B2Mr49HAZeoOMzFHdmtwLU5hdVk>

Tournament administrators will be able to create and manage tournaments, while the app provides admins with the current tournament status. Players can use the app to see what matches they still have to play, and also browse tournaments to find teams to join and play. After a match has been completed, each player in that match can report the match outcome on TourneyTrack, and the app will then notify the tournament admin to approve those results.

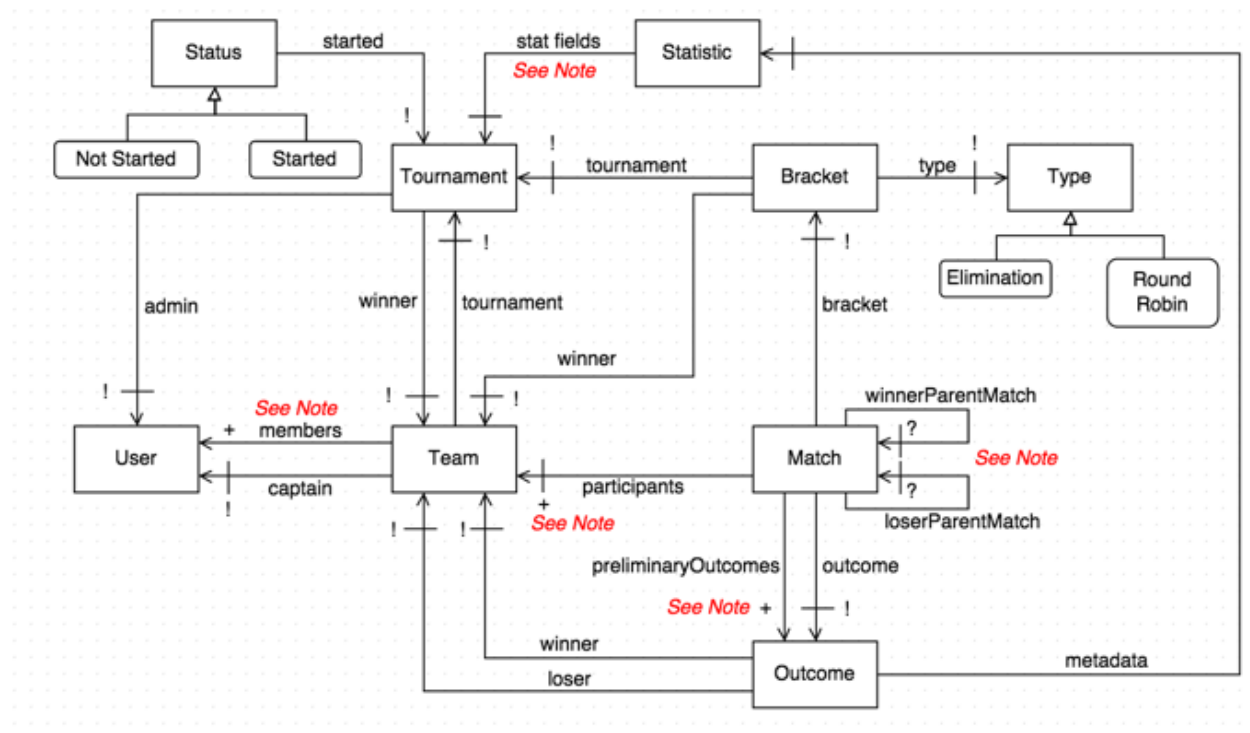
The scope of our app is currently restricted to the MIT community. All players and admins must register with a valid Kerberos, and upon registration, our app validates the entered username against MIT People API to make sure it's valid.

## II. Design Model

### Data Model and Concepts (Updated)

To model our application, we introduce the following concepts:

- **Admin:** In a Tournament set up, there needs to be an Admin responsible for the accuracy of the match structure and the integrity of the outcome data. When a User obtains the admin status for a Tournament, he/she is responsible for the creation, management, and verification of all set-up and results of matches in the Tournament. Additionally, an Admin is simply a status that a User can have in relation to a Tournament. As a result, a User can use the same account to manage and participate in Tournaments, which is representative of how a member would interact in a more informal tournament. This concept allows someone to be responsible for the integrity of the tournament while still reflecting the flexibility of roles in informal dorm tournaments.
- **Outcome Reporting:** Each Match will have an Outcome, which stores the winner and loser of a match, what the scores were, etc. By originating from a player and requiring admin approval, an Outcome ensures that the match result is represented accurately and makes result-reporting more efficient.
- **Bracket Auto-generation:** A Bracket is a collection of dependent Matches between multiple Teams. Matches and their dependencies will be automatically generated based on the Bracket's type (elimination or round robin), and Teams will be automatically matched up. Additionally, when a Match has been played, the winning Team will be automatically advanced to the next round of Matches and the losing Team will be dropped into a loser's Bracket (if it exists). Once a bracket is generated, a graphical representation of it is displayed on the tournament page, providing an intuitive visualization of tournament information.
- **Statistics Auto-aggregation:** When all the Matches in a Bracket have been played, we also will automatically aggregate the Outcomes and declare either the Team who won the most matches to be the winner of the Bracket in the case of Round Robin, or the Team who won the final elimination match in the case of Elimination.
- **Team Formation Scheme:** Users have the ability to join Teams within a Tournament, and as a result, players have the power to organize themselves into Teams. However, an admin still has ultimate control, and can 'close' a Tournament. Once the admin closes a Tournament, Teams can no longer join that Tournament and the admin can now start adding Brackets. This feature reflects how informal, smaller-scale tournaments are organized, and allows the process of signing up for a Tournament to flow naturally.

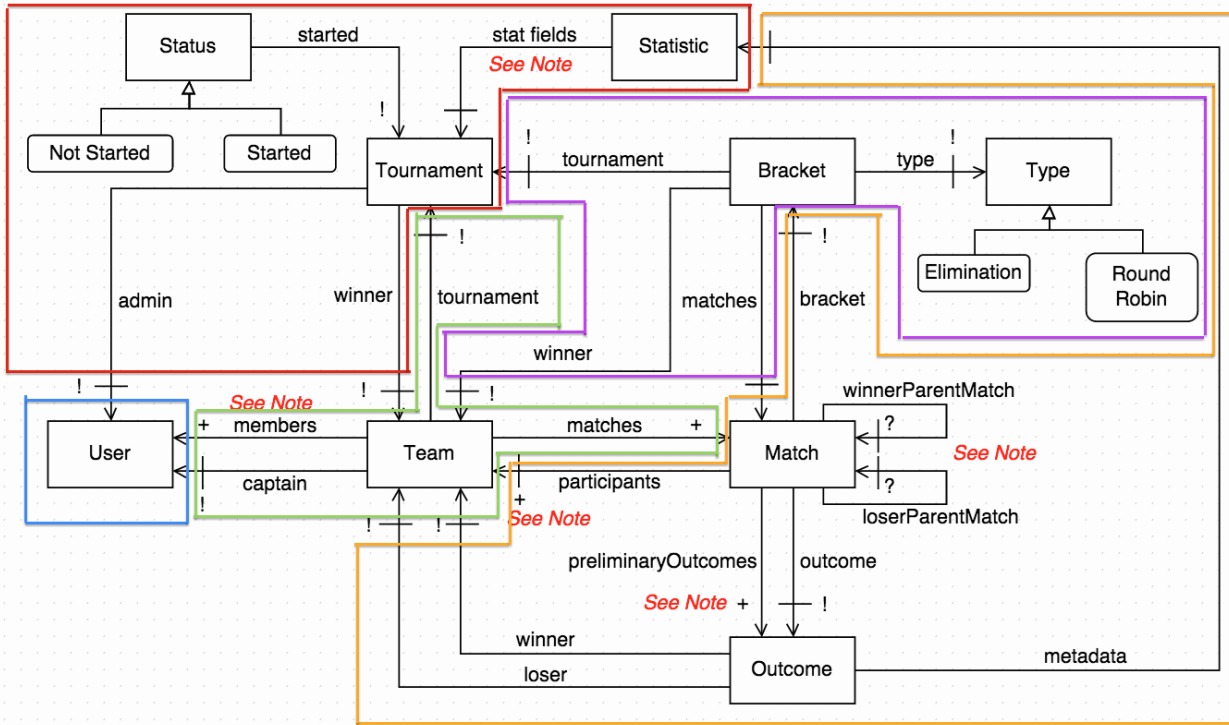


(<https://www.draw.io/#G0B2Mr49HAZeoOUmM4VFhTYkxxTGM>)

#### Note about Data Model:

1. A Team's members can change until the Tournament the Team belongs to has started. After the Tournament is started, the Team's members list becomes immutable.
2. A Match can have 1 or 2 participants. If there is only 1 participant, that Team is automatically declared the winner.
3. The winner's next Match in the Bracket will be winnerParentMatch, and the loser's next Match in the Bracket will be loserParentMatch.
4. When a Match is finished, each Team can report a preliminary Outcome for the Tournament admin to see. The Tournament admin can either choose one of the reported outcomes or compromise between them and submit it as the final outcome.
5. We want to enable tournament-specific stat tracking. This means that when a tournament is created, the admin should specify the type of Statistic fields the Match outcomes should include in its metadata. And when the Teams submit their preliminary Outcomes, the Team captains must include data for these specific Statistic fields.

And our contour diagram is as follows:



Note: If the majority of the arrow is contained within a contour, then that relation is owned by the schema that that contour corresponds to.

To illustrate our entities, in terms of intramural sports, a possible Tournament may be “Soccer Fall 2014 League A”, which would be a round robin Bracket of Matches between multiple Teams, such as “Next House”, “Conner 4”, and etc., and each Team would have a list of User, namely MIT affiliates.

In terms of dorm tournaments, a possible Tournament may be “Next House Ping Pong”, which would consist of elimination Brackets “Main Bracket” and “Loser’s Bracket”. Each Team would start in the Main Bracket, and the first round losers would be filled into the Loser’s Bracket. And each Team would simply be a Team of one User.

### III. Design Behavior

#### Security Concerns (Updated)

##### Policy

##### Requirements

- Availability
  - The information stored and calculated in our site should be easily and readily accessible to all users, not heavily delayed by long-running requests
- Integrity

- The information recorded for each match outcome should only enter the database if approved by the tournament's administrator so that the data is an accurate representation of the actual outcome of that match
- User Authentication
  - Users should only be able to sign up as themselves and not use another user's kerberos to sign up and act as that person

#### Non-Requirements

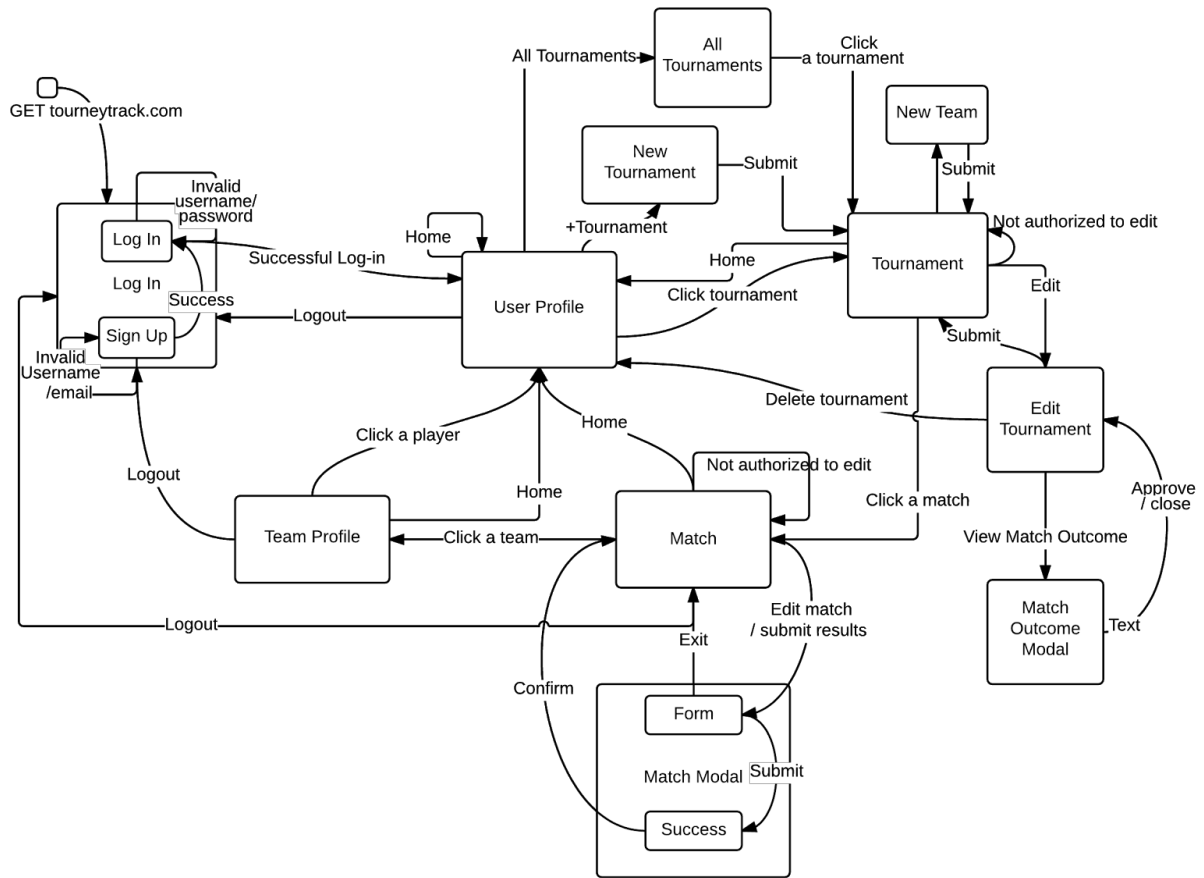
- Privacy
  - All player statistics should be viewable from all other users of the system

#### **Threat Model/What the attacker can do + Mitigations**

- Send false match outcome data to the admin for approval
  - *Mitigation:* Use a two-layered match data approval scheme, where the creators of a tournament have the final say regarding match outcomes and creators of tournaments must approve new teams that join and creators of teams must approve new team members
- Attempt to alter a tournament set-up or send bogus information to our system directly
  - *Mitigation:* Use authentication to prevent unauthorized tournament modification
  - *Assumption:* Trust Heroku to keep database secure
- Spam the system with a large number of requests at a time from outside of the site
  - *Mitigation:* Respond to request with pointers instead of auto-populating fields so that requests are responded to more quickly by the server, allowing other requests to get in to the server amidst a spamming attack
- Attempt to access usernames and passwords in database, or to gain access to another User's account
  - *Mitigation:* Encrypt passwords to protect user accounts
  - *Mitigation:* Send verification emails to a user's mit email address to confirm a User's identity
  - *Assumption:* Trust MIT People API to validate current MIT affiliate Kerberos ID
- XSS attacks in text inputs
  - *Mitigation:* Use validator.js and AngularJS \$sanitize service to sanitize inputs to prevent javascript injection attacks

#### **User Interface**

## Action Flow and State Transitions:



\*On any view except the modal windows and Log In, clicking the TourneyTrack logo or “My Profile” will point to User Profile, and clicking Logout will point to Log In. Clicking either of those on the Log In page will redirect to itself.

## Wireframes (for MVP)

Below are the wireframes for our MVP:



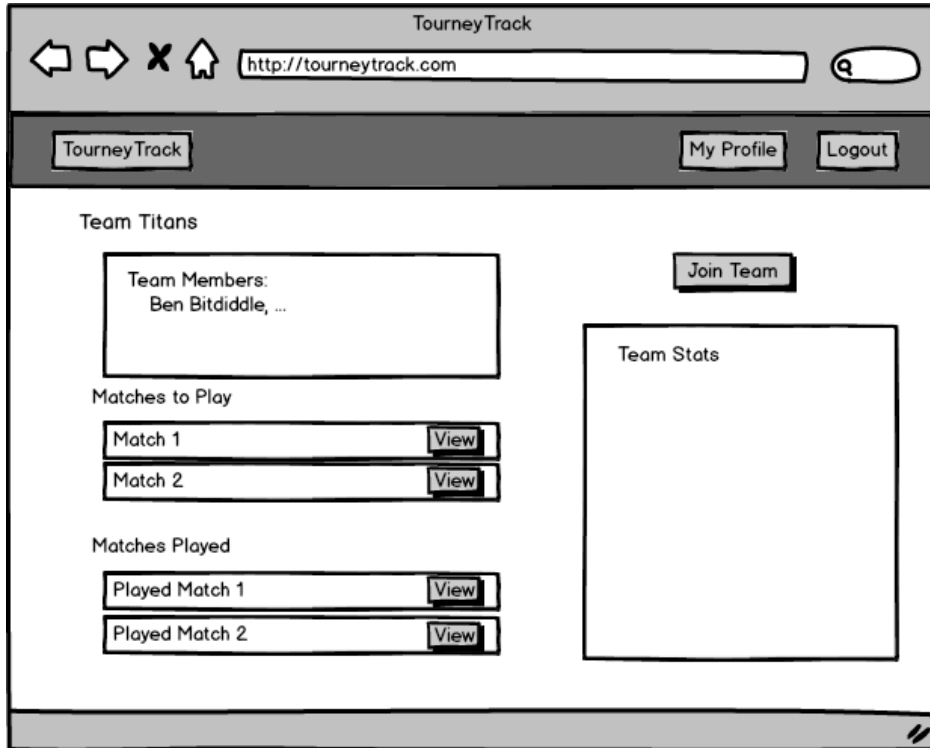
Login Page (“login.html”, “sign up.html”)

The screenshot shows a web browser window titled "TourneyTrack" with the address bar displaying "http://tourneytrack.com". The page has a dark header bar with the "TourneyTrack" logo on the left and "My Profile" and "Logout" buttons on the right. The main content area is divided into two columns. The left column, titled "Login", contains a "Username" input field, a "Password" input field, and a "Login" button. The right column, titled "Signup", contains a "Username" input field, an "E-mail" input field, a "Password" input field, a "Confirm Password" input field, and a "Signup" button.

User Profile Page (“profile.html”)

The screenshot shows a web browser window titled "TourneyTrack" with the address bar displaying "http://tourneytrack.com". The page has a dark header bar with the "TourneyTrack" logo on the left and "My Profile" and "Logout" buttons on the right. The main content area displays the user's profile for "Ben Bitdiddle". It includes sections for "My Matches" (with "Match 1" and "Match 2" each having a "View" button), "My Teams" (with "Team Titans" having a "View" button), and "My Tournaments" (with "Tournament 1" and "Tournament 2" each having a "View" button, and a "+Tournament" button). A "View All Tournaments" button is located at the bottom of the tournament section. On the right side, there is a large empty box labeled "User Stats".

## Team Profile Page



A browser window titled "TourneyTrack" with the URL "http://tourneytrack.com". The page has a navigation bar with "TourneyTrack", "My Profile", and "Logout" buttons. The main content area is titled "Team Titans" and contains several sections: "Team Members" with a list showing "Ben Bitdiddle, ...", a "Join Team" button, "Matches to Play" with two entries "Match 1" and "Match 2" each with a "View" button, "Matches Played" with two entries "Played Match 1" and "Played Match 2" each with a "View" button, and a "Team Stats" box. The browser window includes standard navigation icons and a search bar.

TourneyTrack

http://tourneytrack.com

TourneyTrack My Profile Logout

Team Titans

Team Members:  
Ben Bitdiddle, ...

Join Team

Team Stats

Matches to Play

Match 1 View

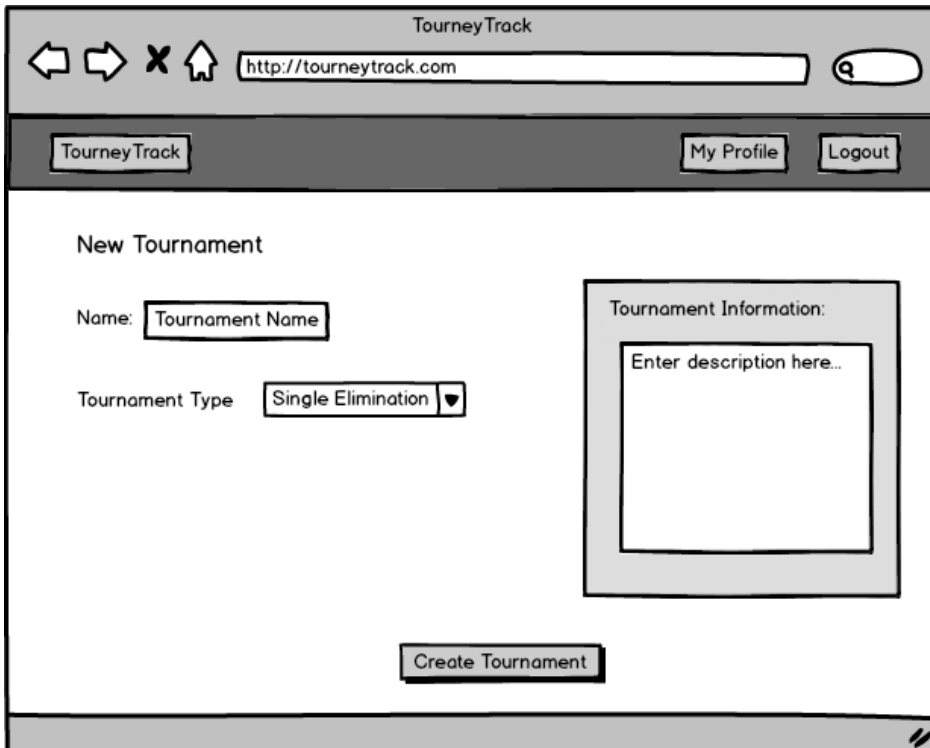
Match 2 View

Matches Played

Played Match 1 View

Played Match 2 View

## New Tournament Page



A browser window titled "TourneyTrack" with the URL "http://tourneytrack.com". The page has a navigation bar with "TourneyTrack", "My Profile", and "Logout" buttons. The main content area is titled "New Tournament" and contains a form with "Name:" and "Tournament Type" fields, a "Tournament Information" box with a description input, and a "Create Tournament" button. The browser window includes standard navigation icons and a search bar.

TourneyTrack

http://tourneytrack.com

TourneyTrack My Profile Logout

New Tournament

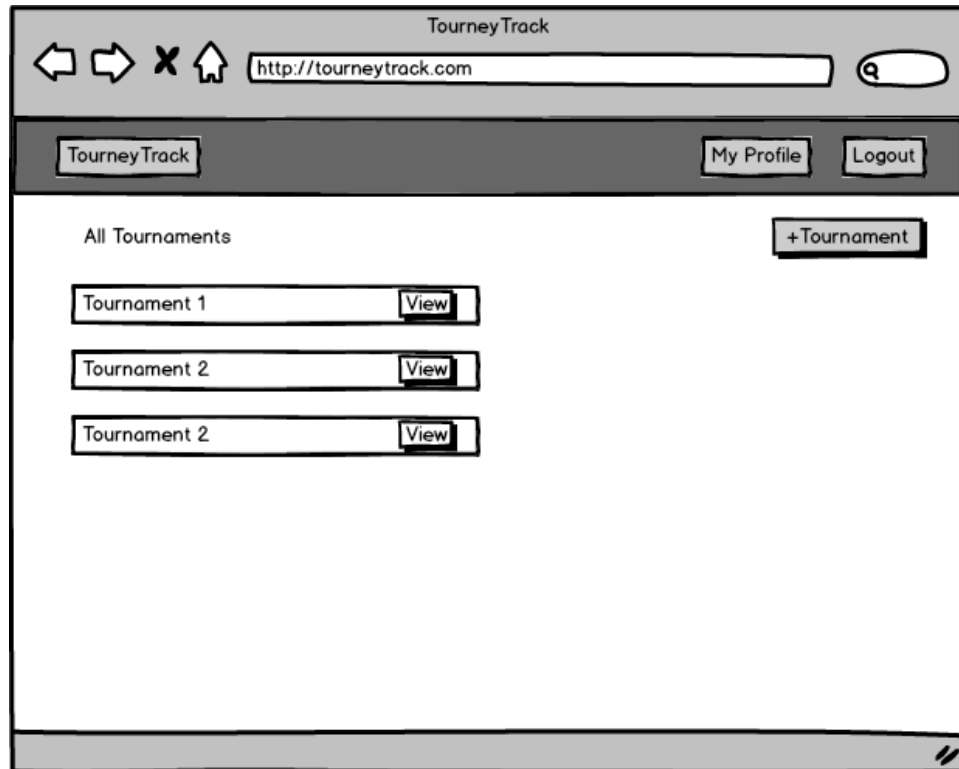
Name: Tournament Name

Tournament Type Single Elimination ▼

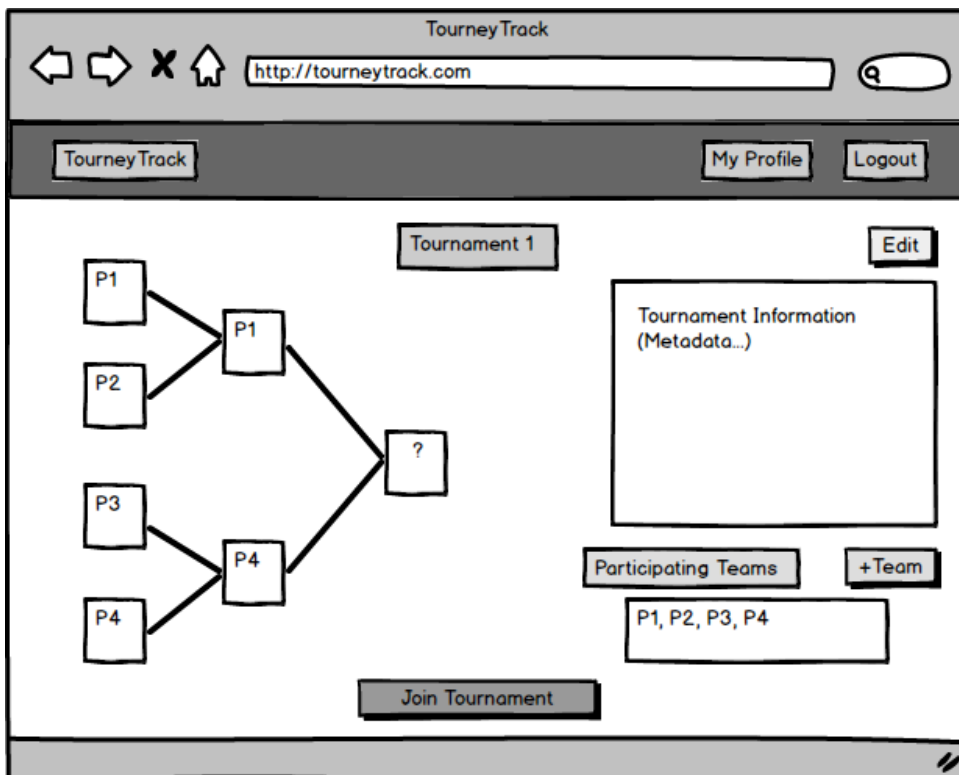
Tournament Information:  
Enter description here...

Create Tournament

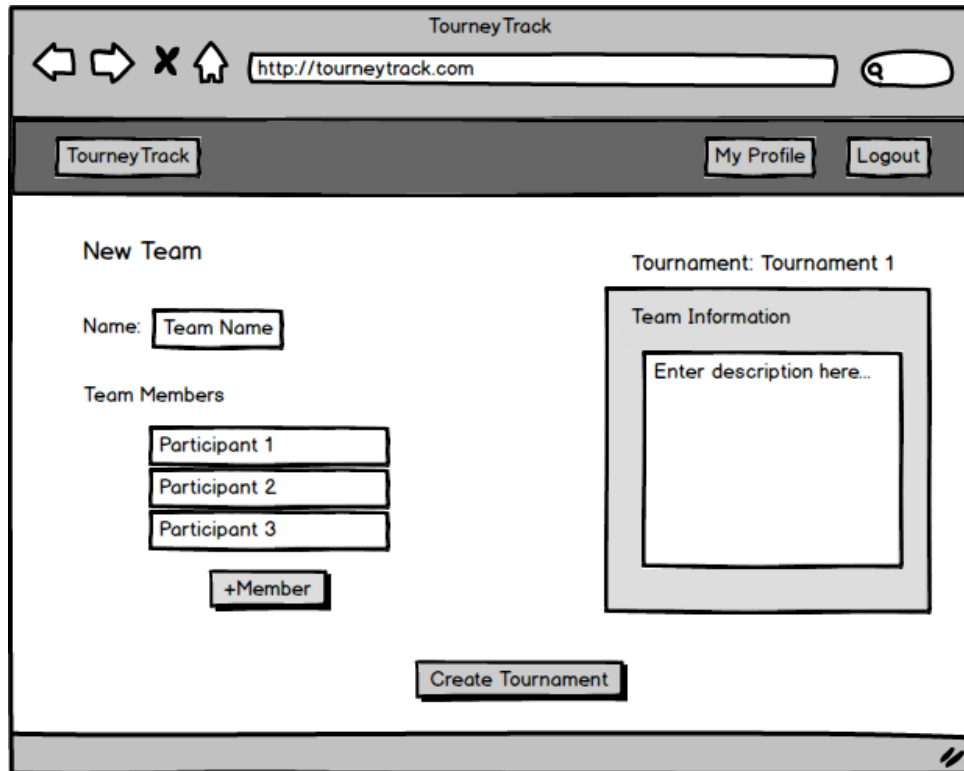
## All Tournaments Page



## Tournament Page

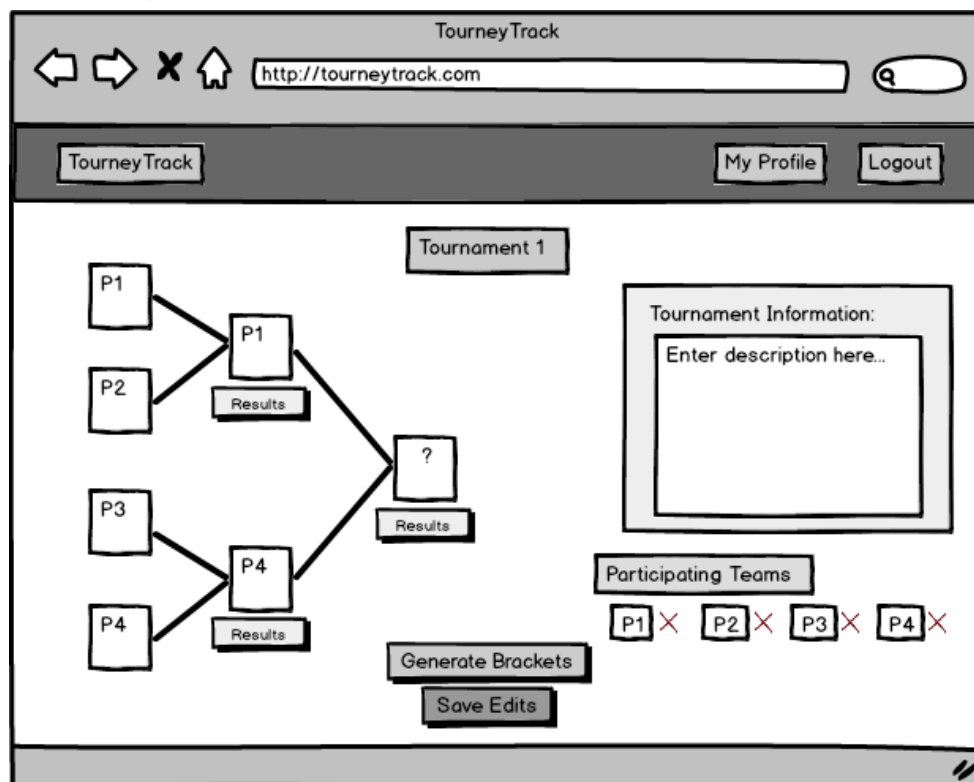


## New Team Page



A screenshot of a web browser window titled "TourneyTrack" with the URL "http://tourneytrack.com". The browser's address bar shows the URL and a search icon. The page has a dark header bar with "TourneyTrack" on the left and "My Profile" and "Logout" buttons on the right. The main content area is titled "New Team". It features a "Name:" label followed by a text input field containing "Team Name". Below this is a "Team Members" section with three stacked text input fields labeled "Participant 1", "Participant 2", and "Participant 3", followed by a "+Member" button. To the right, under the heading "Tournament: Tournament 1", is a "Team Information" box containing a large text area with the placeholder "Enter description here...". At the bottom center is a "Create Tournament" button.

## Edit Tournament Page



A screenshot of a web browser window titled "TourneyTrack" with the URL "http://tourneytrack.com". The browser's address bar shows the URL and a search icon. The page has a dark header bar with "TourneyTrack" on the left and "My Profile" and "Logout" buttons on the right. The main content area is titled "Tournament 1". It displays a tournament bracket diagram on the left. The diagram shows four participants (P1, P2, P3, P4) on the left, with P1 and P2 competing in a match labeled "Results", and P3 and P4 competing in a match labeled "Results". The winners of these matches (P1 and P4) are shown competing in a final match labeled "Results". To the right of the bracket is a "Tournament Information:" box containing a large text area with the placeholder "Enter description here...". Below this is a "Participating Teams" section showing four boxes labeled "P1", "P2", "P3", and "P4", each followed by a red "X". At the bottom are two buttons: "Generate Brackets" and "Save Edits".

### Tournament Modal Page (for result approval)

The screenshot shows a web browser window titled "TourneyTrack" with the URL "http://tourneytrack.com". The page has a navigation bar with "TourneyTrack", "My Profile", and "Logout" buttons. A sidebar on the left lists participants P1, P2, P3, and P4. The main content area is titled "Tournament 1" and displays a modal window titled "Match 3 Results". The modal contains a "Winner:" field with "P1" entered, a "Score:" field with "3" and "1" entered, and an "Approve" button. Below the modal, there are buttons for "Generate Brackets" and "Save Edits". A red "X" is visible next to the P4 participant label.

TourneyTrack

http://tourneytrack.com

TourneyTrack My Profile Logout

Tournament 1

P1 P2 P3 P4

Match 3 Results

Winner: P1

Score: 3 vs 1

Approve

Generate Brackets

Save Edits

### Match Page

The screenshot shows a web browser window titled "TourneyTrack" with the URL "http://tourneytrack.com". The page has a navigation bar with "TourneyTrack", "My Profile", and "Logout" buttons. The main content area is titled "Match A" and displays a bracket diagram with participants P1 and P3 on the left, leading to a central box with a question mark. To the right of the bracket is a "Match Information (Metadata...)" box. Below the bracket is a "Winner:" field with a question mark. At the bottom, there is an "Enter Results" button.

TourneyTrack

http://tourneytrack.com

TourneyTrack My Profile Logout

Match A

P1 P3

Match Information (Metadata...)

Winner: ?

Enter Results

## Match Outcome Modal Page

The diagram illustrates a web browser window for 'TourneyTrack'. The browser's address bar shows 'http://tourneytrack.com'. The page has a header with 'TourneyTrack', 'My Profile', and 'Logout' buttons. A modal form titled 'Enter Results for Match A:' is displayed in the center. The modal contains a 'Winner:' label with a 'Winner Name' input field, and a 'Score:' label with 'P1 Score' and 'P3 Score' input fields separated by 'vs'. A 'Submit' button is at the bottom of the modal. To the left of the modal is a 'P1' label, and to the right is a 'P3' label. Below the modal is an 'Enter Results' button.

## Design Challenges

How do we keep track of match outcome data?

*Problem:*

How should we keep track of all the match outcomes in order to support many different types of sporting events? Sports have varying rules for determining match winners, as well as different relevant statistics, such as number of goals scored, number of strikes, etc.

*Potential solutions:*

- 1) Support a fixed number of sports and tailor the system to handle those sports in their specific ways
  - Pros:
    - Better for the user, since it allows sport-specific analysis and an easier process of reporting data
  - Cons:
    - Doesn't allow our system to grow.

- Forces us to manually enter new sports into the system whenever a new sport is requested.
- 2) Generalize the concept of match outcomes to “winning” and “losing”
  - Pros:
    - The concept of winning and losing fits most sports, since every match will have a winner that moves on to some parent match, and a loser who may move on to some other parent match
    - Allows our system to easily keep track of tournament progression
  - Cons:
    - We would lose the ability to track specific data regarding match outcomes, like points scored
- 3) Allow users to put in completely customized outcome data
  - Pros:
    - Allows all desired data to be collected for each sport
  - Cons:
    - Difficult for our system to interpret winner and loser from data, since each sport has its own rules regarding who wins
    - Difficult to maintain consistency over all similar sports.

*Our Choice:*

We chose a mixture of *Solution #2* and *#3*. Each match will specify the winner and loser, as well as specify additional statistics that can be stored with every match. These stats will not affect the outcome of the match, since winning and losing should ultimately decide this. But this allows statistics to be aggregated along the entire tournament.

## How do we protect the integrity of match data entered into our system?

*Problem:*

Teams may not report accurate data into the system. Specifically, if a team lost, but they report to the system that they have won, this would have major implications to the rest of the tournament.

*Potential solutions:*

- 1) Trust users to be truthful and update the match outcome themselves
  - Pros:
    - Low-latency for seeing updated results on the site
    - Less centralized system; doesn't require one person to enter scores for every match in the tournament.
  - Cons:
    - Users can put in false data into the system, which can ruin the structure of a tournament
- 2) The admin is responsible for inputting all match data
  - Pros:

- Places trust into one person, rather than requiring trust over all users in a tournament.
  - Easy to implement
- Cons:
  - Requires admin to know the outcome of every match.
  - Either the admin will have to attend every match, or be emailed the results, which reduces into the system that already exists
- 3) Have users input results, but require admin approval
  - Pros:
    - Admin doesn't need to know the results of the match beforehand
    - Keeps data truthful in the system, as admin has final say on the tournament outcome
    - Low latency for match results
  - Cons:
    - Harder to implement
    - Admin must settle discrepancies offline

### *Our Choice*

We decided on *Solution #3* - to allow users to input match outcome, but to have the admin have the final say in the match outcome. We have to trust someone to input correct data, and the most logical person to trust is the person who created the tournament. We will allow teams to post their scores, which will be submitted to the admin for approval. If both competing teams submit conflicting outcomes for a given match, the admin can contact the teams, make the proper changes, and submit the agreed-upon outcome. Since we want our system to focus on low-latency updates of team outcome data, we don't want this extra-step validation process to slow things down too much. To handle this, we will send notifications to both the opposing team and the admin when a team submits outcome data. This reminds the opposing team to report their outcome data, so the admin will have both outcome data statements to check against each other. The notification also serves as a reminder to the admin, so he/she can quickly approve the submission, allowing the update to surface in the system.

### **Should Teams be within the context of a tournament?**

#### *Potential solutions:*

- 1) A team be able to compete over multiple tournaments.
  - Pros:
    - Concept of Team is more akin to literal definition of team, since a team would be defined solely by its members
    - Avoid re-entering new teams for competing in different tournaments
  - Cons:
    - Doesn't allow subsets of members to participate in certain tournaments; everyone would participate in every tournament the team is signed up for.



- Difficult to aggregate statistics across different tournaments, since different tournaments may be for different sports
- 2) A team is specific to a tournament.
  - Pros:
    - Easy to implement
    - Team statistics are easier to maintain
    - Simplifies process of adding participants to tournament, since it will be understood that the entire team will participate
  - Cons:
    - Teams consisting of the same members will have to be duplicated for each tournament they join
    - Statistics cannot be aggregated across tournaments

### *Our Choice*

We chose *Solution #2*, to have teams be specified under the context of a tournament. This makes more sense in terms of keeping track of team statistics, since the data entered for matches will be uniform over all matches in a given tournament. Also, this solution encourages groups of people to make separate teams for different events. So the 5th floor can have a separate team for ping pong and basketball. Another benefit to this solution is that it simplifies the process of adding participants to a tournament. By creating a team under the context of a tournament, a user is also registering this team as a participant of the tournament.

### **Should Teams be immutable?**

Should players be able to be added to a team after a team is created?

Potential solutions:

- 1) Allow players to be added after team creation
  - Pros:
    - Allows users to join a team themselves
    - Allows team to be formed before all of the members are decided upon
  - Cons:
    - Allowing people to join a team halfway through a tournament is unfair in the context of a competition
- 2) Make the teams immutable after creation
  - Pros:
    - Maintains that members remain constant throughout tournament
  - Cons:
    - Entire team must be entered into system upon creation
    - Doesn't allow users to join a team themselves

### *Our Choice*

We will *combine Solutions 1 & 2*: Players can join a team themselves up until the tournament starts. Once the tournament starts, the team will become immutable and players can no longer join the team.

## Should players be allowed to be part of multiple Teams within a Tournament?

### *Problem:*

Should a player be able to be a participant in multiple teams in the same tournament?

### *Potential Solutions:*

- 1) Have no restrictions on letting players participate in multiple teams in the same tournament.
  - Pros:
    - Easy to implement
    - Gives more flexibility in the system
  - Cons:
    - Doesn't fit real world model: If two teams have the same participant, which does that participant play with when the two teams compete?
    - Will be difficult if we ever want to implement user statistics
- 2) Restrict it so that no one user can be a captain of more than one team in a given tournament
  - Pros:
    - Gives more flexibility in the system
  - Cons:
    - Same problem as first Con of Solution 1: does not restrict a user from being a participant of two opposing teams
    - Will be difficult if we ever want to implement user statistics
- 3) Restrict users to participate in at most one team in any given tournament
  - Pros:
    - Fits the real-world model more closely
    - More intuitive to the user
  - Cons:
    - More difficult to implement/enforce

### *Our Choice:*

Solution 3: Players can be a member of at most one team in any given tournament. This prevents the situation where a player is scheduled to play a match against himself.

## Implementation Challenges

### How to keep track of references in the DB to meet the needs of the UI

#### *Problem:*

When viewing a tournament, we display all matches of each bracket in the tournament. We need a way to group matches by bracket, while still being able to retrieve all matches for a given tournament.

*Potential Solutions:*

- 1) Have matches keep track of tournament and bracket that they are in
  - Pros:
    - Easy to query for matches that occur in specific brackets
  - Cons:
    - Requires multiple queries to first find all brackets, and then find all matches in each bracket
    - Duplication of data, since brackets also keep track of what tournament they are in
- 2) Have matches keep track of only bracket, and have bracket keep track of tournament
  - Pros:
    - Minimizes the amount of duplicate data in the DB
  - Cons:
    - We would have to query for all brackets in a tournament, and then for all matches in each bracket
    - Another step for grouping the matches into brackets on the front-end
- 3) Have a bracket keep track of the tournament it belongs to and the matches that are in this bracket
  - Pros:
    - When we query for all brackets in the tournament, we get the list of all matches in the bracket without further calls to the API
  - Cons:
    - Duplication of data in the DB, since matches also point to the bracket they are in

*Our Choice:*

Solution 3: We cannot delete the reference in matches to their bracket, since we need a way to reference their tournament. However, by having a two-way relationship between matches and brackets, we gain the ability to query for all brackets in a tournament, and receive all matches grouped by bracket without any further queries. Additionally, since matches can never be removed or added to a bracket, the list of matches that a bracket contains is immutable after bracket creation and the bracket that a match belongs to doesn't change. This means that with this two-way reference, we gain functionality without having to worry about keeping the information consistent.

## Maintaining the logged in user from page to page

*Problem:*

In order to prevent unauthorized users from editing teams or tournaments that they don't own, we must maintain the identity of the user at all times in the system. We use a session

variable on the server side, but we need a way of verifying and persisting the user information on the front-end.

*Potential Solutions:*

- 1) Query the API each time we change urls on Angular
  - Pros:
    - Very easy to implement, given that we are using Passport on the server for authentication
  - Cons:
    - Requires a call to the API on every page of the app
- 2) Store the identity in cookies and keep this data as we navigate across urls on Angular
  - Pros:
    - Only requires one authentication call to the server
  - Cons:
    - Added difficulty of figuring out how to use Angular to make cookies persist from controller to controller

*Our Choice:*

Solution 2, as it cuts down on the unnecessary calls to the API. The additional difficulty is worth the benefit of not having to make an extra API call on every page.

**Implementing the outcome report flow, allowing team captains to report outcomes and requiring admin approval**

*Problem:*

We needed to allow both competing teams to be able to report the outcome of a match for admin approval. Should teams be able to submit multiple outcomes? How do we store all the reported outcomes and differentiate between the reported outcomes and the approved outcome? We needed a way to report these outcome requests to the admin in a way that is both intuitive and flexible.

*Potential Solutions:*

- 1) Only allow one reported outcome per team, and allow the admin to choose one of the two for approval. Whichever outcome is not approved is deleted and the approved outcome is set as the final outcome.
  - Pros:
    - Does not require us to keep track of which report belongs to which team to allow for resubmission
    - Eliminates the hassle of the admin having to enter final outcome data into a form
  - Cons:
    - Prevents admin from approving any data that differs from a reported outcome (which may be necessary to resolve conflicts)
    - Teams cannot update their submitted results in the case of an erroneous submission

- Admin does not know which team reported which outcome
- 2) Instead of an outcome field, store a list of outcomes that is appended to whenever a team captain reports an outcome. The admin can see a list of all reported outcomes, and approves one from that list.
  - Pros:
    - If there are discrepancies or mistakes, a team captain can submit an alternative result for the admin to approve
    - Generalized to allow more than two teams
  - Cons:
    - Admin does not know which team reported what outcome
    - Admin might see outcomes that were accidentally submitted and have no relevance to the actual result - rely on admin to weed out erroneous data
- 3) Store the reported outcomes in a separate field and keep track of which team each outcome was reported by, so that there is one most recent reported outcome for each team. Display this information to the admin, and allow the admin to submit a final outcome that does not necessarily have to be a copy of a reported outcome by filling in entries in a separate form.
  - Pros:
    - Provides a way for the admin to directly address any discrepancies between the outcome submissions
    - Allows the admin to know which team reported each outcome
    - Allows teams to overwrite previous outcomes with a more recent version
    - Doesn't show the admin irrelevant information, like erroneous reports that were later overwritten
    - Generalizes scheme to allow more than two teams, while restricting each team to one (updateable) reported outcome
  - Cons:
    - Requires admin to manually type in the final outcome and scores, which can be tedious and cause errors

#### *Our Choice:*

Solution 3: This solution is the most flexible and efficient. It puts the responsibility of getting rid of erroneous outcome submissions on the Team captain, and the responsibility of overriding discrepancies between Teams on the Admin, which is most reflective of the actual responsibilities of those two roles. We also plan on streamlining the admin approval process by implementing automated comparisons between the two reported results to auto-fill fields that have no discrepancies.

#### **How to enforce uniform outcome metadata across each tournament**

##### *Problem:*

In order to support multiple sports, we decided to allow users to enter sport-specific metadata in the match outcomes (see design challenge above). We need to ensure that the metadata content is uniform throughout the tournament, in order to provide useful team-specific statistics.

*Potential Solutions:*

- 1) Have the admin specify the required metadata fields upon tournament creation
  - Pros:
    - Easy way to enforce constraints on metadata, as constraints will be defined before any outcomes can be reported
  - Cons:
    - Forces users to keep track of the required data field for each match they submit outcomes for
    - We might get bogus data if users are forced to fill in every field regardless of whether or not they have data for those fields
- 2) Do not constrain the metadata entered, and aggregate the union of all statistics in the tournament
  - Pros:
    - Easier to implement
    - Allows users to only input the data they remember into the system
  - Cons:
    - Statistics may not make sense or may not represent true statistics of the tournament if some matches are not included in the aggregation

*Our Choice:*

Solution 1: This constraint on the system allows for better, more useful statistic aggregation for each tournament. The statistics fields specified for a Tournament are meant to span the entire Tournament, and will be interpreted as so. If some fields have insufficient data and do not include all matches, the statistics will no longer be representative of that Tournament. Additionally, requiring all fields may encourage users to report more data than they would if they could simply submit blank fields.