|  |  |
| --- | --- |
| GhostRunner Design Document  Design and functional specification |  |

**(covers parts of the system only)**

**Version:** 1.0

**Last Updated:**  June 21, 2017

**Copyright**: GhostRunner.

**Author****:** Alamgir Farouk

**Table of Contents**

1 Overview 4

2 Enitities 4

2.1 Owners 4

2.2 Players 4

2.3 PlayerSeason 4

2.4 Teams 4

2.5 TeamPlayers 5

2.6 TeamLineUp 5

2.7 TeamLineUpPlayers 5

2.8 GameUserLineupPlayers 5

2.9 Game 5

2.10 GameEvents 6

2.11 Viewers 7

3 Database Schema 7

4 Game States and transition 8

4.1 Inviter states 8

5 Game Login 10

6 Game Broker signaling 11

7 Player Importing 13

7.1 Differences between types 13

7.2 Import mechanism 13

8 Invitation cycle 14

8.1 Selecting team (Inviter) 14

8.2 Selecting a pitcher 15

8.3 Proposed modifications to the current UI: 16

8.4 Selecting the lineup 17

8.5 Proposed UI changes 17

8.6 Game start 18

8.7 In game properties 19

9 Game Field UI 21

9.1 Proposed changes to UI 21

9.2 Player Cards 21

9.3 Data (TBD) 24

10 API Samples 25

**Legal Page**

**Confidentiality Agreement**

The undersigned reader acknowledges that the information provided by GhostRunner in this document is confidential; therefore, the reader agrees not to disclose it without the express written permission of GhostRunner Inc.

It is acknowledged by the reader that information to be furnished in this document is in all respects confidential in nature, other than information which is in the public domain through other means and that any disclosure or use of same by the reader may cause serious harm or damage to GhostRunner Inc.

Upon request, this document is to be immediately returned to GhostRunner.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Signature

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Name (typed or printed)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Date

# Overview

This document provides the schema and API details for the Baseball Game engine.

The schema is based on Baseball rules, GhostRunner game and remote interaction requirements. We describe each table separately, and show their combined relationship at the end.

UID (User identifier) represents a user (logged in). Complete description of User entity is omitted in this document.

NOTE: Actual table names may be different based on schema requirements.

# Enitities

These are the high level entites we will use.

## Owners

An Owner is a user in the game, identified by UID and other meta data.

## Players

This represents a player in the ‘pool’ of players. This table captures personal (non-game) related information about a player.

## PlayerSeason

A particular player’s performance during a season is caputured in this table. It is also known as the ‘player card’.

## Teams

One owner can have multiple teams. So, teamId is incrememented. Teams may become dormant or retired. We can decide how many active teams are possible.

## TeamPlayers

This represents a player in the ‘pool’ of players. This table captures all the information about historical players, possibly using auxilliary tables for historic data.

## TeamLineUp

A lineup is created by an owner using his team players. A lineup can have players only from a team. So the Team comes first .

## TeamLineUpPlayers

(not used)

## GameUserLineupPlayers

This table hold the lineup for any particular game and each player related information.

## Game

A game represents a new game being played between two TeamLineUps. At any time, the game entity records the state of the game, who is playing, who is standing where, etc. It should record all information such that a game can be reconstructed and play can continue. Current scores are also cached although they could, in theory, be computed from the playevents entries.

Games

* Home(id)
* Away(id)
* id(id)
* current inning
* player positions
* [game state]
* [ui state]
* [cached score]
* …
* Inninings
* Who is at bat
* Who is pitching
* Out status (0,1 or 2)
* Base status(0 through 7) (who where)

Figure Games table

## GameEvents

Each play event is a chronological record of the events that combine together to form the game. The events maintain a sequence, which is important for multiplayer distributed games.

GameEvents

* gameId(id)
* sequeneId(id)
* inningId
* eventType
* resultType
* …

GameSubEvents

* gameId(id)
* sequeneId(id)
* subSequenceId(id)
* eventType
* resultType
* …

Figure BaseballGamePlays

## Viewers

A viewer is a user who is allowed to observe an ongoing game. A viewer-game relation table will be created (omitted as too straightforward)

# Database Schema

The following diagram shows the overall schema of the game engine. Most attributes are omitted, only the relational model is shown.

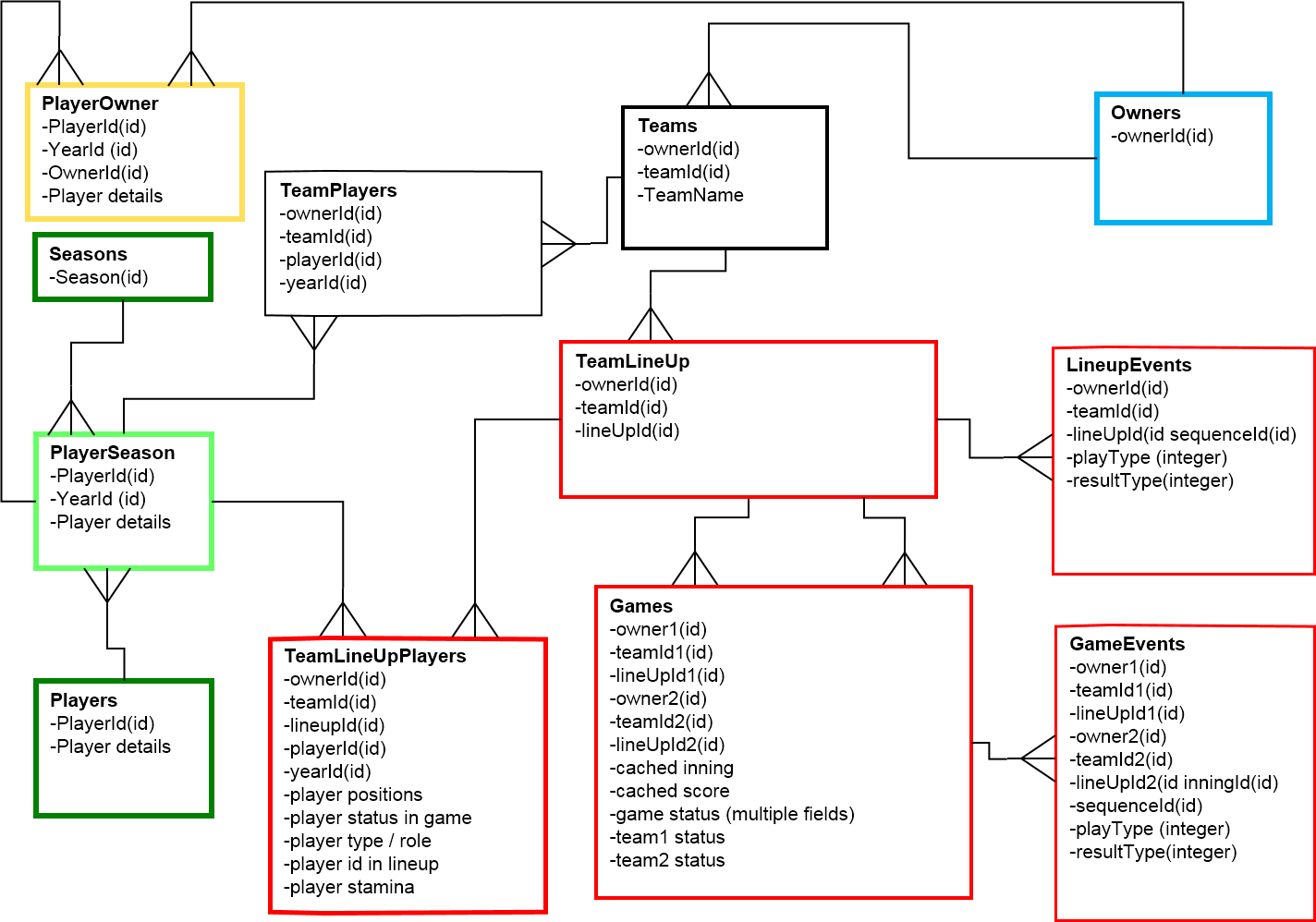


Figure ER diagram for game and teams

# Game States and transition

The two teams as well as the game itself go through unambiguous states which are persisted at any time, all transitions between them being atomic.

We omit the game states for simplicity. In general, the game state will capture the ‘turn’ information (Owner1, Owner2 or System), but within the turn, the actual state of team will be maintained in the TeamLineup table, and individual player states will be maintained for the appropriate player in TeamLineUpPlayer table.

## Inviter states

We only consider the state transitions for the Inviter (the one who initiates the game) and the Invitee.

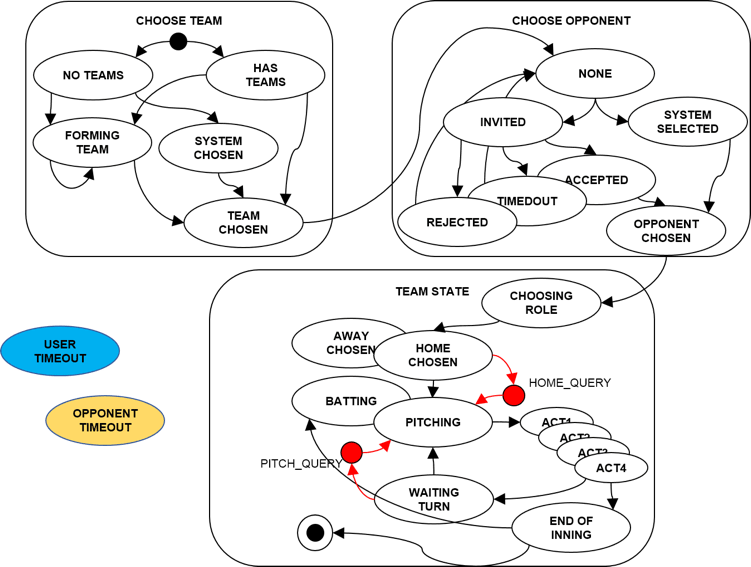


Figure Inviter (the initiator of the game) state transitions.

Invitee states

The person being invited goes through a slightly different set of states at the beginning, but we repeat the full life-cycle for completeness.

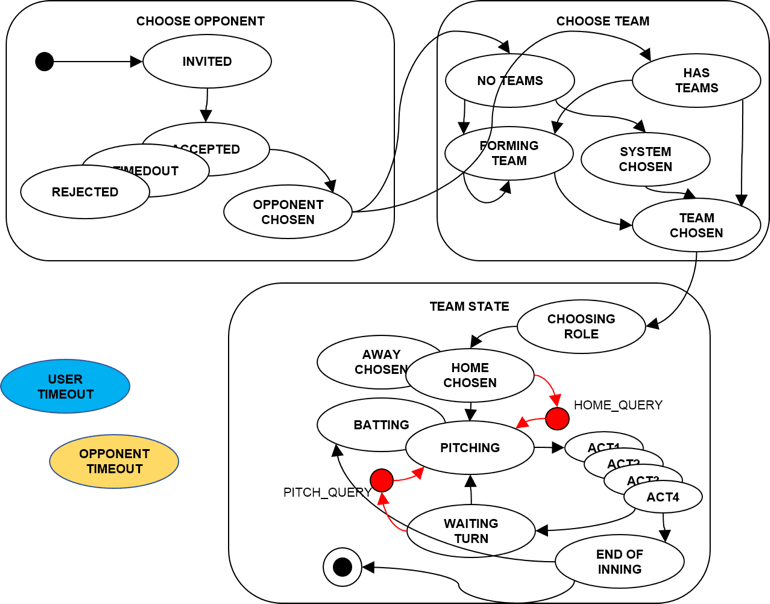


Figure Invitee (person who was invited) state transitions.

Game flow

The game flow itself is modeled as a flow diagram, to better illustrate the actors (owner1, owner2 or system)

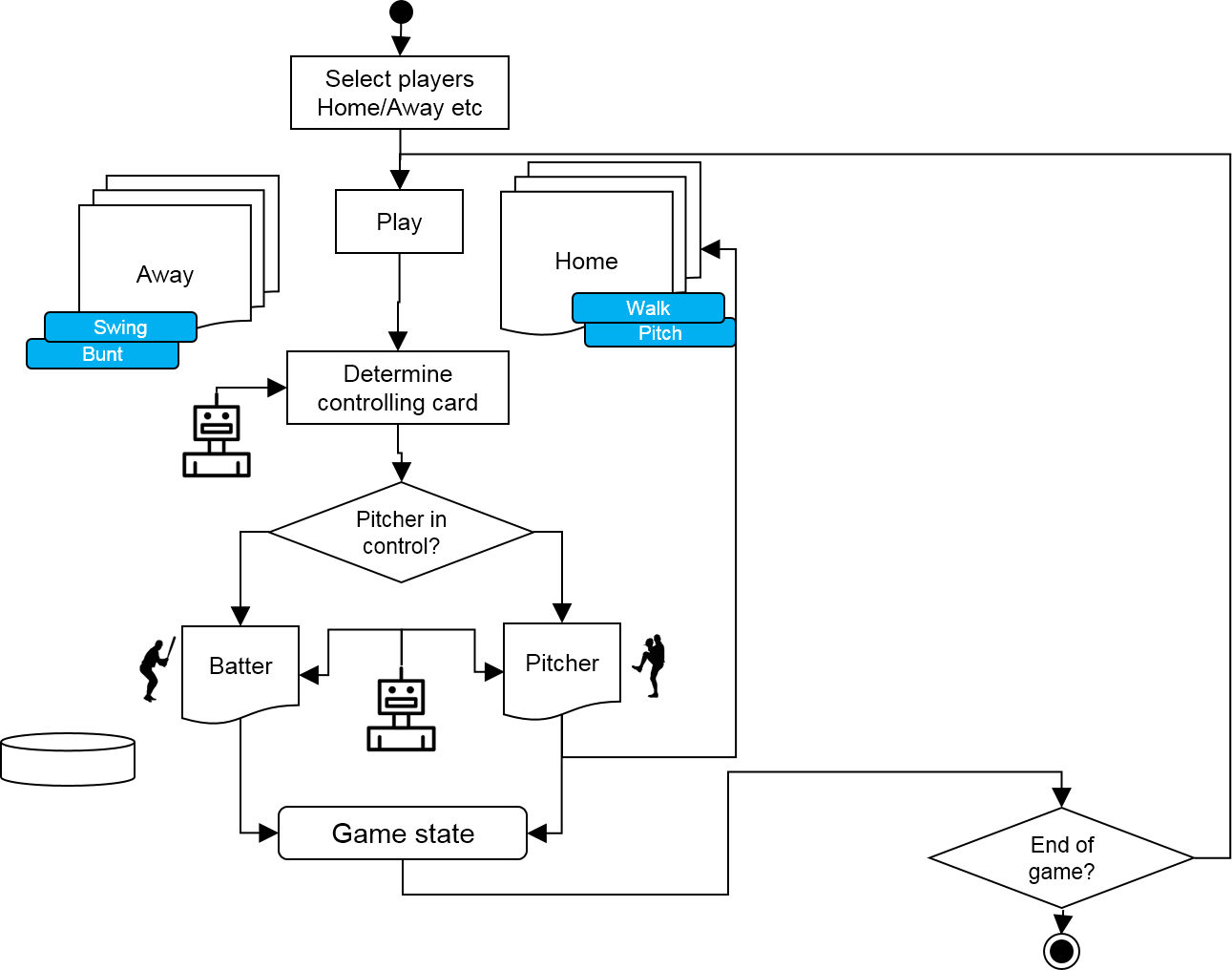


Figure Game flow diagram.

# Game Login

The login will use the standard authentication from our authentication package, with the following requirements

The login component (javascript object?) will be a separate component but it can be built as part of the package.

The login will set a client side cookie (as usual) with the UID as well as fire an event on login and logout and allow attaching listeners to it. Other pages may also want to listen to this event, so it may be a good idea not to have any dependencies other than jquery etc.

The login will maintain all user details (username, UID) by retrieving them from the server. If retrieval fails, state will be logged out, cookie removed (same as now).

The login will parse the URL and use the embedded UID and retrieve the username etc., when the URL contains a UID. Invalid UIDs (when server throws an error) will be ignored silently.

Valid UIDs will result in logged in status as usual.

# Game Broker signaling

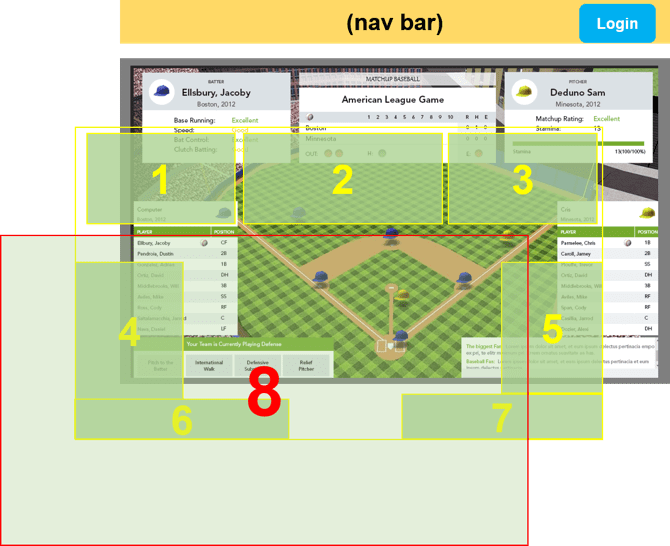


Figure 7 Main UI components

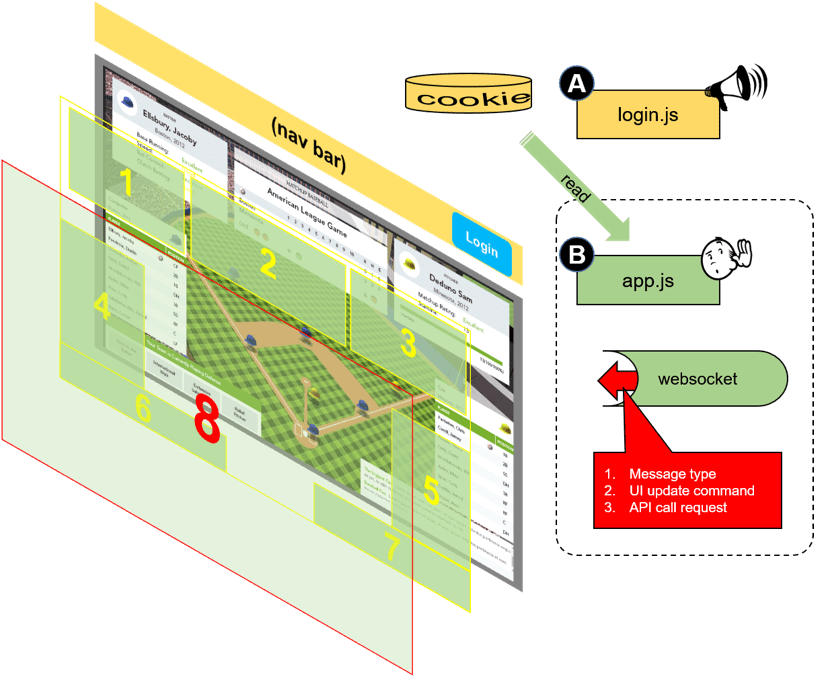


Figure 8 Signaling via websocket will control the masks and UI state. Default state will be full mask (8)

# Player Importing

Two types of players are identified, Pitchers and Fielders. They do not have the same set of data, which means some database will be null/undefined for respective types.

## Differences between types

The main differences between fielders and pitchers will in the way they are rated, and used for lineup creation.

## Import mechanism

The excel worksheets will be read by an importing software which will convert the excel data into json (JSON) files, which, in turn will be used as arguments for the player creation APIs.

# Invitation cycle

The invitation cycle consists of selecting an existing member or entering an email whence an adhoc-member is created and then selected for invitation. Additionally, a ‘starter’ Pitcher must be selected by the inviter. The figure below shows the combined member-selection, and email entering screen.

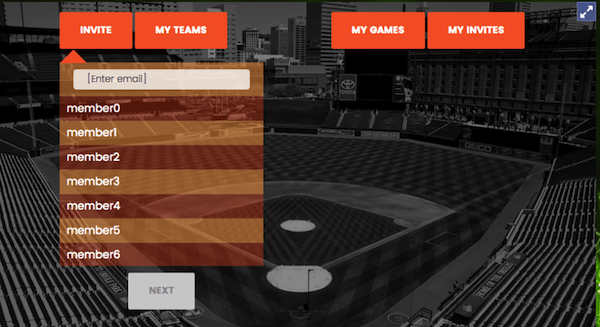


Figure Selecting member or entering email

## Selecting team (Inviter)

The inviter, upon selecting user as described above, will be presented with a UI for selecting a team.

1. He can choose from exiting (free) teams
2. or use a team he has created from a pool of available players.

Option one will be implemented first.

The figure below shows the view for selecting the team. NOTE: Only free teams are shown in the example. Teams that the user owns will have ‘edit’ and ‘delete’ buttons next to them.

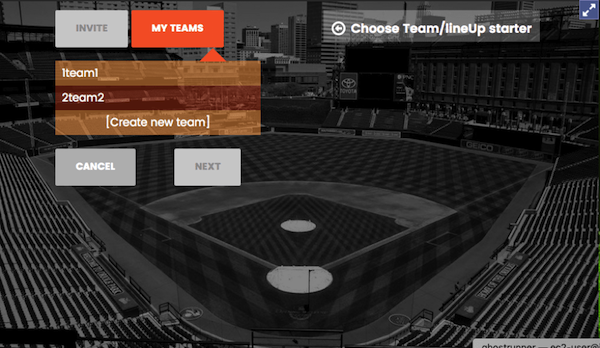


Figure Team selection screen.

Note that the current UI also provides an option for creation of a team. This may be hidden or removed during deployment as it may be a paid function.

## Selecting a pitcher

Once a team has been chosen, we will show only the pitchers from that team and allow only one selection (radio button).

**NOTE**: A pitcher is identified as a player that was imported from the ‘Pitcher’ worksheets. No other variable is considered in determining who should be included in the pitcher list.

**NOTE**: the user has to enter a lineup name first. This lineup name is for display purposes only, it has no significance otherwise.



Figure Selecting starter pitcher from team.

## Proposed modifications to the current UI:

1. Remove player ‘cost’ (currently showing random numbers).
2. Radio buttons will moved to the right-most column
3. The pitcher rating will be shown (Currently ‘UU’)

The pitcher rating will use the field ‘MU\_Rating’ as the rating for the pitcher.

Note: we do not display the ‘Role’ from the database for a pitcher in this view.

Upon successful submission of the invitation, the ‘inviter’ will have to go through the same steps.

Once the inviter has also selected his starter, the invitee will have to select his lineup (from the same team).

## Selecting the lineup

The user is presented with a list of ALL players in his team (chosen earlier).

1. Players may be sorted by fielders and pitchers.
2. Fielders will be shown TPV, and choice of field positions based on P1, P2 etc values.
3. Fielders will be shown choice of Batting order (BO)
4. Pitchers will be shown rating and pitching role, and choice of pitching role
5. All players will be shown selection checkbox.

The figure below shows the UI for lineup selection.

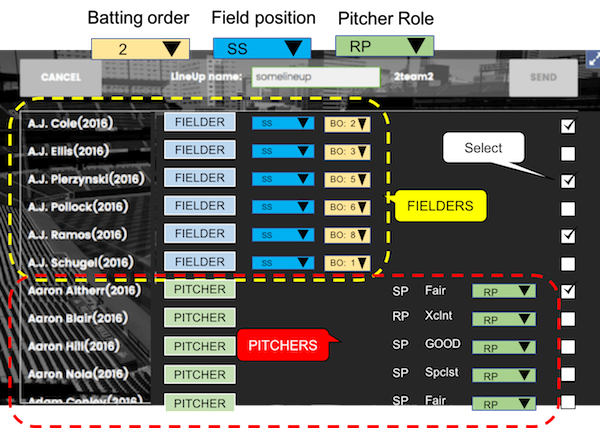


Figure Lineup selection UI. (see modifications proposed below)

The lineup selection UI will show ALL available players from a team. The team name and lineup name will be displayed on top for reference.

## Proposed UI changes

1. The ‘cost’ (TPV) will **not** be shown for Pitchers. These columns will be empty for the pitcher.
2. A new column, Pitcher ‘Rating’ will be added (so this column will have empty cells for ‘FIELDER’ players.
3. The UI will enforce the TPV value such that total TPV is 75 or less. Selecting FIELDERs that exceed this total will not be possible.
4. The UI will enforce PITCHER selection via the following rule:
   1. Only one xlnt, one or more fair, one or more good etc (need to fill out the full rule)
   2. Staying below the rule will be allowed (so having no exclent pitcher will be allowed, etc).
5. The chckeboxes will be move to the far right.
6. A drop down will be provided for each player for the properties which must be configured during line up selection. .
   1. For FIELDERs it will be populated by P1, P2 etc. values from the worksheet. Every entry the FIELDER has in the worksheet will allow a new entry in the dropdown. This results in the ‘POSITION’ value for this FIELDER.
   2. For PITCHERs, the dropdown will be populated by possible roles (SP, etc). NEED to clarify max/min bounds, for PITCHER dropdown options. This will result in the ‘ROLE’ value for the PITCHER in the line up.
7. The current Total of ‘values’ of FIELDERs will be shown on the top bar. There will be no corresponding roll up for PITCHERs selected.
8. The total number of players will be 9 (need to verify this)

## Game start

Once both players have selected lineups, the inviter is notified that the invitee has accepted his invitation and he can start the game. Once a Game is started it is switched to RUNNING state.

NOTE: A user can have only one RUNNING game at a time. But a user can have unlimited number of games in any other state, such as INVITING, PAUSED, etc.

The invitation cycle automatically pauses any RUNNING game, so that when this game is RUNNING other games will be paused. Other users will be notified without option.

## In game properties

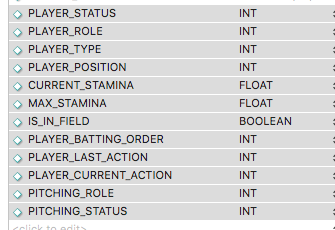


Figure In-game player variables

PLAYER\_STATUS: General purpose variable we can use for maintaining the status of a player (we define possible values as needed)

PLAYER\_ROLE: For pitchers this will be the typical pitch roles as recognized in Baseball terminology. For fielders they will be defined by us, and may simply be one constant (‘fielding’)

PLAYER\_TYPE: PITCHER or FIELDER in order to enable overriding their original roles. Initially they will match original player roles.

PLAYER\_POSITION: For fielders these will be standard Baseball terminology (e.g. SS etc). For Offense players, they will be positions defined by us to indicate base, home plate etc. Players not in game or not playing will have special positions defined by us.

CURRENT\_STAMINA: used for pitchers while pitching, and ignored (negative or null) for all other players.

IS\_IN\_FIELD : a Boolean to indicate if the player is physically in the field or not.

PLAYER\_BATTING\_ORDER: Used for capturing the current batting order.

PLAYER\_LAST\_ACTION: We will maintain the last action taken by a player mostly for calculations or game display.

PLAYER\_CURRENT\_ACTION: This is the action the player is taking at the moment. May be null or not needed depending on whether we make all moves ‘atomic’ (so moves must end) or not.

PITCHING\_ROLE: We use standard baseball terminology to capture current pitching role for pitchers. For Fielders we will use derived placeholders.

PITCHING\_STATUS: We may not need this, based on whether we consider pitches as atomic or not.

**Please add any other in-game (RUNNING) state variable we may need for EVERY player (fielders, pitchers, batters).**

# Team creation UI

A user can create a team from available players. While this may be a paid function later, for now we are implementing it as a free/available function to get it working.

The UI is shown below.

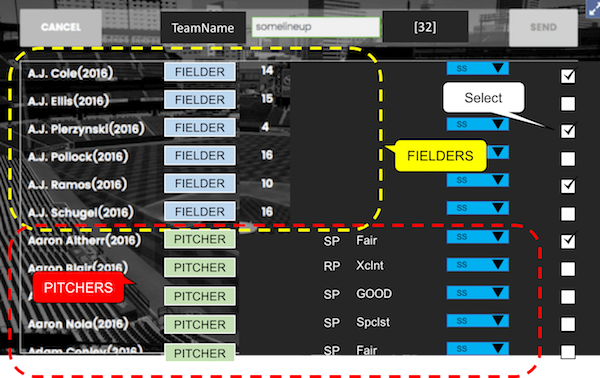


Figure We show costs, which will be added for total of 75. Similar restrictions for pitchers.

# Game Field UI

Once the game is started (RUNNING) the UI switches to the ‘field’ mode as shown below.



Figure Field UI during a RUNNING game. (NOTE FICTIOUS DATA USED)

## Proposed changes to UI

The following changes are proposed or in progress

1. The unidirectional hats will be replaced by hats at various angles of rotation. We will generate hats at 45 degree intervals for now.
2. The top right ‘out’, h, e, values need to be clarified. Are they all Boolean or integers.
3. The chat icon will be removed (it will not be implemented in this version)
4. The background picture needs to be replaced with a picture we have rights to.

## Player Cards

Right clicking on any player on the field will display a player card. The design will be derived from the following sketches.

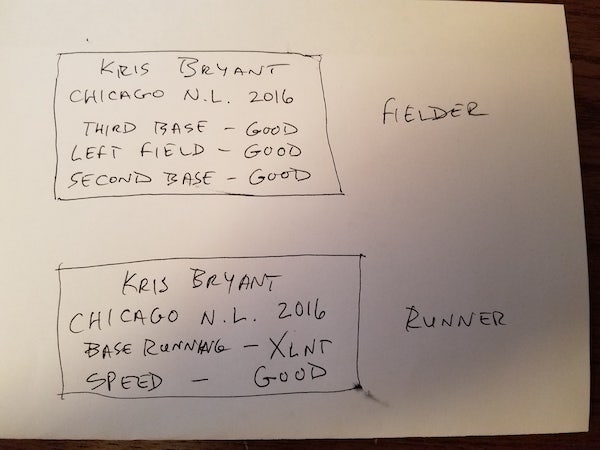


Figure card sketch 1

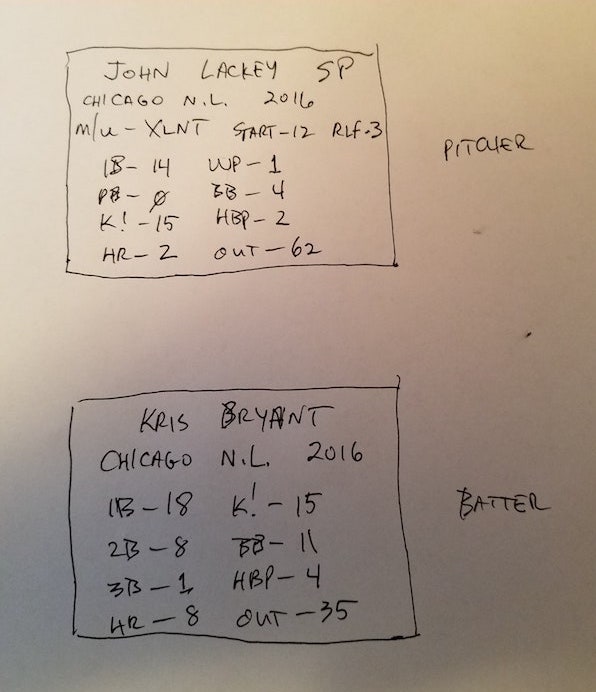


Figure card sketch 2

We identify the following:

The player card depends on the field position of a player (not the type or role etc).

Following four positions are possible

1. Fielder (DEFENSE)
2. Runner (OFFENSE)
3. Batter (OFFENSE)
4. Pitcher (DEFENSE)

## Data (TBD)

# API Samples

This is a sample API description

***isUserNameAvailable***

|  |  |
| --- | --- |
| **REST Mapping:** | *[GET] /authentication/isUserNameAvailable/?username=username0* |
| **Description:** | *Check if the username is available or not.* |
| **Parameters:** | *username* |
| **Return Value:** | *Result object, success must be true.* |
| ***Exceptions:*** | *UnableToComplyException is thrown if the supplied arguments are invalid. PanicException is thrown if there is an internal server error.* |