Reclusive Planning Poker

# Overview:

The reclusive planning poker game is intended to reduce the amount of meeting time needed for story estimations. Additionally, the data store for the game can be analyzed to provide insight on estimation reliability.

Not Started:

Clarify:

Functional:

The goals are as follows:

* Integration with JIRA
  + Use JQL to retrieve issues to estimate, story point field is customfield\_10004
  + Save estimated stories back to JIRA
  + Pull up games to get report of estimate accuracy, vs. JIRA
* Teams of users
  + Allow users to create teams of players to use on games
* Games
  + History of each estimation made by team members
  + Date & Time (games expire)?
  + Discussion - possible integration with JIRA/BitBucket?
  + Report on consensus - Users can update vote & most recent vote wins
* Users Game queue
  + Show players games they are a part of when the log into the system
  + Notify players when a new game is ready to play, possibly with a Chrome Extension?
  + Reminders with Expires to gauge players performance
  + When all members have added an estimate or $estimateTimeout expires then push another notification with the final result
  + Allow members time to discuss and update their estimate based on the discussion
  + Once consensus is reached or estimate manager has closed the estimation, report results back to the team

# Game Play:

* Anyone can create a game
  + Games take a team of players
  + If you create a game
    - you own that game
    - you must have created a team of players first to attach to the game
* Anyone can add you as a player on a game
  + If someone adds you as a player on a game
    - The game will show up in your game queue.
* When playing the game
  + If you are not the game owner
    - Playing part of a game will set the queue status to partially played
    - Completing all stories in a game will set the queue status to played
  + If you are the game owner
    - Accepting a game
      * Removes it from other players queues, but it will remain in your game list
      * Deleting a game will remove it entirely from your list, and other players queus

# Architecture – Controllers:

## Login Controller:

* Functions:
  + Link:(GameOwnerCntrl) Login Game Owner’s
  + Link: (AccntCntrl) Create New Game Owner Account

## Account Controller

* Functions:
  + Create new game owner account

## ResetAccount Controller

* Functions:
  + Allow user to reset email

## Game Owner Controller:

* Functions:
  + Show logged in user
  + Link:(TeamCntrl) Create New Teams
  + Link:(CreateGameCntrl) Create New Poker Game:
  + Show list of games created by game owner (Played, Unplayed)
  + Show list of games in queue for play

## Team Controller:

* Functions:
  + Create Teams
    - Players
      * Add
      * Edit
      * Remove
  + Edit Teams
  + Delete Teams

## Create Game Controller:

* Functions:
  + Create A New Game
  + Attach a Team
  + Import JIRA Stories
  + Upon Game Creation – Reroute to Game-Owner Controller

## Add Stories Controller:

* Functions:
  + Allows game owners to add new stories to an existing game
  + Only accessible from a game
  + Allows structured stories

## Play Game Controller:

* Functions:
  + Allow individual to play each story in solitude, leaving and resuming a game.
  + Show highlighted cards for team mates who have not yet played
  + Upon a players completion, change players status on the game
  + Upon game completion, change a the games status
  + If game is opened by game owner
    - Show current players cards
    - Allow Owner to accept game

# Firebase Data Structure:

Firebase is a non-relational data store. [This article is helpful.](https://www.firebase.com/blog/2013-04-12-denormalizing-is-normal.html)  Querying is limited in scope. Most of the filtering done in game works off the notion of the firebase $priority field, similar to a secondary key. If the $priority is set when data is saved into firebase, we can then use the .startAt() and .endAt() methods to retrieve those objects by priority.

An example of the non-relational nature of the data would be as follows. When creating a new game, a team is attached, or rather copied into the game object. If the team is modified it will not affect the team attached to the game. The game will have a team, as it existed when the game was created.

## Top Level Nodes:

* Users : $priority = user email
  + Name: Users Full Name
  + Email: company email address.
* Games : $priority = user email
  + Status: Queued, Played, Accepted
  + Owner: game owners email
  + Team: team is used to populate players onto each story
  + Stories
    - Players
      * Estiment
      * Status: Queue
      * Played: True, False
      * Email: For data scraping
* GameQueue : $priority = games players email
  + GameId: firebase id of the game
  + Status: has the player played the game
  + Name: game name
  + Player: players email
* Teams : $priority = email
  + Name: team name
  + Players
    - Email: players email – this is the key
    - Name: Players Name