

This is where I dump ideas, so there may be some redundancies.

Use-Case Driven UIs

Mid-Season Draft Tool

- The Player List UI 1 (Fig. 1), but modified so that each row allows some extra info to be displayed, including one or more of
  - some summary representation of several player ranking measures
  - a small message or number conveying extremely salient contextual information (injured: projected return date, teammate projected return date)
  - would-be-if-drafted team measures, or
  - the difference between  $m(T)$  and  $m(T - \{p\})$  (where  $T$  is user’s team and  $p$  is the player) for currently-selected measure  $m$ , or
  - a summary representation of the above-defined difference for multiple measures.
- Clicking on a player will display a pop-up with details
  - performance stats for season games to date
  - additional contextual fields in each row of game data - roster, quarterback, target share
  - Potentially also some creative “timeline” display against it, showing team states and their duration.

Compare

Justindiaz74

to

PHYS

☐ Highlight Waivers

☐ Filter

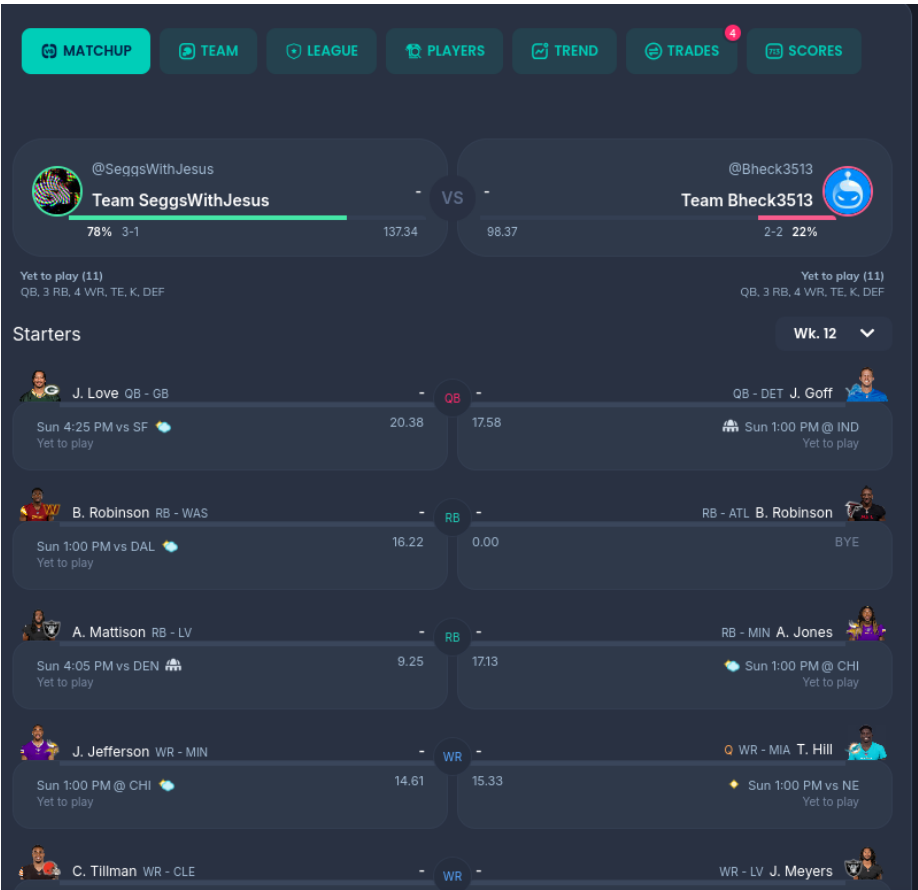
\*Indicates Starter.

<b>QB</b>	1. Lamar Jackson	7240
	2. Jalen Hurts	6055
	3. Josh Allen	5641
	4. Joe Burrow	4633
	5. Jayden Daniels	3608
	6. Kyler Murray	3218
	7. Baker Mayfield	2761
	8. Justin Herbert	2479
	9. Jordan Love	2468
	10. Patrick Mahomes	2384
	11. Brock Purdy*	2379
	12. C.J. Stroud	1901
	13. Jared Goff	1694
	14. Bo Nix	1651
	15. Kirk Cousins	998
	16. Russell Wilson	822
	17. Tua Tagovailoa	754
	18. Geno Smith	711
	19. Anthony Richardson	687
	20. Drake Maye*	550
	21. Caleb Williams	535
	22. Matthew Stafford	525
	23. Sam Darnold	447
	24. Aaron Rodgers	378

 **RB** | 1. Saquon Barkley | 10527 ||  | 2. Christian McCaffrey\* | 9623 |
	3. Bijan Robinson	9334
	4. Derrick Henry\*	8774
	5. Jahmyr Gibbs	8043
	6. De'Von Achane	7571
	7. Joe Mixon	7532
	8. Alvin Kamara\*	7439
	9. Breece Hall	7180
	10. Kyren Williams	6441
	11. Josh Jacobs	5839
	12. Jonathan Taylor\*	5836
	13. Kenneth Walker	5796
	14. David Montgomery	5078
	15. James Cook	4964
	16. James Conner	4646
	17. Chase Brown	4195
	18. J.K. Dobbins	3394
	19. Chuba Hubbard	3357
	20. Isaiah Pacheco	3120
	21. D'Andre Swift	3085
	22. Aaron Jones	3023
	23. Brian Robinson	2861
	24. Tyrone Tracy	2703
**WR**	1. Ja'Marr Chase\*	10402
	3. Amari St. Brown	8614
	4. Nico Collins\*	7826
	5. A.J. Brown	7646
	6. Puka Nacua\*	6681
	7. Cooper Kupp	6490
	8. Drake London\*	6091
	9. Tyreek Hill	6076
	10. CeeDee Lamb\*	6005
	11. Malik Nabers	5316
	12. Garrett Wilson	5030
	13. Mike Evans\*	4710
	14. Marvin Harrison Jr.\*	4423
	15. DK Metcalf	4359
	16. George Pickens\*	4147
	17. Terry McLaurin	4143
	18. Tee Higgins	3805
	19. Zay Flowers	3576
	20. Deebo Samuel	3260
	21. Davante Adams	3016
	22. Jayden Reed	2942
	23. Jaxon Smith-Njigba	2690
	24. Brian Thomas	2421
**TE**	1. Travis Kelce	4866
	3. George Kittle\*	4052
	4. Trey McBride	3947
	5. T.J. Hockenson\*	1939
	6. David Njoku	1819
	7. Sam LaPorta	1710
	8. Cade Otton	1501
	9. Mark Andrews	1274
	10. Kyle Pitts	1100
	11. Evan Engram	1081
	12. Dallas Goedert	1055
	13. Dalton Kincaid	865
	14. Taysom Hill	718
	15. Jake Ferguson	675
	16. Tucker Kraft	652
	17. Jonnu Smith	603
	18. Hunter Henry	394
	19. Zach Ertz	378
	20. Dalton Schultz	308
	21. Cole Kmet	261
	22. Pat Freiermuth	246
	23. Will Dissly	233
	24. Mike Gesicki	233

width: “50%” }

1



{ width: “50%” }

## Trade Tools

**Trade Manual Analysis** Could be integrated into previously-mentioned roster UI styled after the typical “Matchup” UIs (Fig. 2). We want to show comparison of two values:  $m(t_0)$  and  $m(t_1)$ , where  $m$  is the currently-selected measure,  $t_0$  and  $t_1$  are user’s team before and after the trade, respectively.

**Trade Finder** Search league for trade that would optimize some user-specified objective function  $f(X)$  and satisfy a set of user-specified constraints  $g_1(X) < c_1, \dots, g_n(X) < c_n, X \in D$ . We could define the feasible set  $D$  in multiple ways, but for the moment let  $X$  be some matrix representation of both teams’ before and after states. (Obviously, the UI will enable the user to specify these in a more intuitive way. No mathematical optimization experience required.) For example

- “mutually advantageous”
  - objective function:  $f(X) = \text{gain}(t_{\text{user}}, \text{trade})$
  - constraints:  $\text{gain}(t_{\text{opponent}}, \text{trade}) > 0$ ,
  - where  $\text{gain}(t, \text{trade})$  is how many more fantasy points team  $t$  is projected to score, rest-of-season, after making **trade**. (where we consider **trade** to be a function over the collection of sets of players,  $\text{gain}(t, f) = p(f(t)) - p(t)$  where  $f$  is the trade and  $p$  is projected rest-of-season fantasy points.)
  - i.e. maximize our benefit whilst still being beneficial to the other team.
- “maximizing difference between apparent and actual opponent-team gain”
  - e.g.
    - \* apparent: free rankings provided online that don’t take into account league settings
    - \* actual: projected points using league settings
    - \* this wouldn’t be a great thing to optimize, but would be interesting to see. A better set up would be “maximize actual user-team gain while keeping apparent opponent-team gain over a minimum threshold”

We’ll have to show the above defined optimization problem can be solved, or approximated to some degree of tolerance, reasonably efficiently. On second thought, it might be best to think of the criteria/constraints ourselves and give the user a set of pre-defined options. Or just present the trades we find using our hard-coded optimization problems to the user. I do think an easy-to-understand (potentially simplified) explanation of what was optimized would be very beneficial.

**Trade Retrospective** This would be a *great* way to draw people to site. Go through past trades and enable evaluation of their fairness/quality based on

- information available at the time
- how they actually affected teams afterwards

## General Entity-Representation UIs

### Player

We want to have a page that conveys a lot of information about a player at one time, including

- how that player's projections have been trending over the last week
- their performance history
  - raw performance metrics
  - fantasy points, using league settings
- context for past performances - Look into timeline UIs
  - their injuries (obviously)
  - teammates' injuries
  - team roster changes
  - strength of matchups
- performance projections
  - stats projections
  - ScoringConfig (stats projections) -> fantasy projections
- context for future
  - returning teammates
  - strength of matchups

## NFL Teams

- fantasy points and stats allowed - to position - drill down into wide receiver types
- fantasy points and stats allowed - to play type - short/long pass, qb rush
- fantasy points and stats allowed - to (passing, rushing, receiving)
- defensive roster changes

## League Teams

- 2-Team Comparison of current players' past performance
- 2-Team Comparison of current players' projected performance
- Able to change ranking measure for each of listed UIs
  - ranking measures
    - \* (past/projected performance) by (points/stats)
    - \* projected performance add'l inputs
      - can use raw-rankings projections for standard-scoring (PPR/HALF/STD), as there are *a lot* more opinions available than with stats projections
      - next week/ROS
  - UIs
    - \* PlayerListUI1 (Fig 1.)
    - \* "Matchup"-style UI (Fig 2.) with players' values next to them, and totals at op
      - some sort of summary comparison