Fantasy Player Portal Final Project Report

Pedro Avalos Jimenez, Marion Geary, Jacob Snyder, and Claire Wagner

I. Development Tools

Frontend

- JavaScript, HTML, and CSS (for frontend development)
- React.js (JavaScript library for UI development)
- Node.js (JavaScript runtime environment)

Backend

- Python (for backend development)
- R (for data analysis)
- Django (Python-based web framework)
- nflverse (packages for NFL data)

Development

- Visual Studio Code and RStudio (IDEs)
- GitHub repository (https://github.com/jakesnyder7/pyball) (for version control)
- Google Drive (for collaboration on project reports and presentations)

II. Requirements

User and System Requirements

A. ManipulatePlayerSpreadsheet

1. The user will be able to view data on players by position.
    1.1. The user will be able to view statistics about players in each fantasy-football relevant position in a spreadsheet format.
    1.2. The system will automatically fetch the data and populate the spreadsheet.
2. The user will be able to sort, filter, and apply conditional formatting to the data.
    2.1. The user will be able to sort player data by any displayed statistic.
    2.2. The user will be able to filter player data based on keywords.
    2.3. The user will be able to apply conditional formatting to a selected column by inputting a range and selecting a color.

B. CompareTwoPlayers

1. The user will be able to compare players.
    1.1. The user will be able to search for two different players to compare.
    1.2. The user will be able to view graphs comparing the two players.
    1.3. The user will be able to see a data table comparing relevant statistics for the two players.
2. The user will be able to decide which player is a better choice for their team.
    2.1. The user will be able to see suggestions about which player is the better choice through readable, relevant statistics.
    2.2. The user will be able to see suggestions about which player is the better choice through the presentation of the data.
    2.3. The user will be able to see the results of player comparison analyses.

C. EditTeam

1. The user will be able to create a new team of NFL players.
    1.1. The user will be able to configure the roster of the team.
    1.2. The user will be prevented from having more than one team at a time.
2. The user will be able to add players to the team.
    2.1. When adding a player to a given position (e.g. a quarterback), the user will be able to search for the player they wish to add.

2.2. The system will update the roster based on the search results.

2.3. The system will prevent the user from adding a player to a roster entry if the position of that player does not match the definition of that roster entry.

2.4. The user will be able to specify whether the player should be a starter or benched.

3. The user will be able to replace players on the team.

   3.1. The user will be able to remove a player.

   3.2. The user will then be able to add a replacement player in the same way that they added the original player.

4. The user will be able to see information associated with the players on their team.

   4.1. The system will display relevant information and statistics for each player (such as position and fantasy points).

5. The user will be able to access the team that they have created across browser sessions.

   5.1. The system will preserve information about the user’s team and roster configuration across sessions by storing it in local storage.

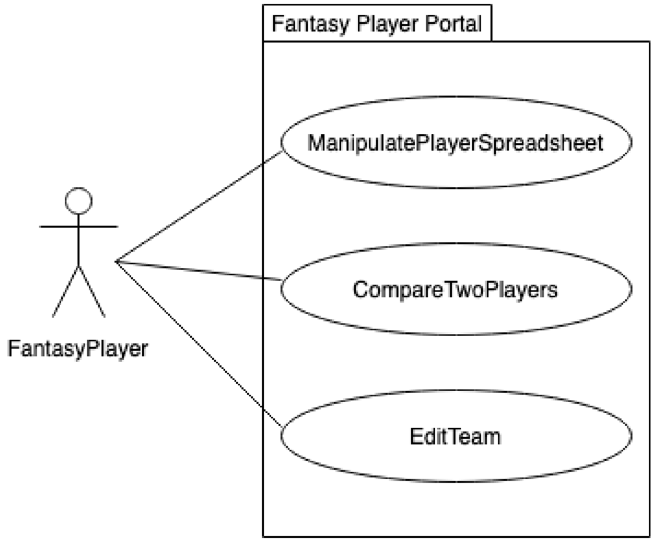
   5.2. The system will make this information available to other use cases.

Domain / Non-Functional Requirements

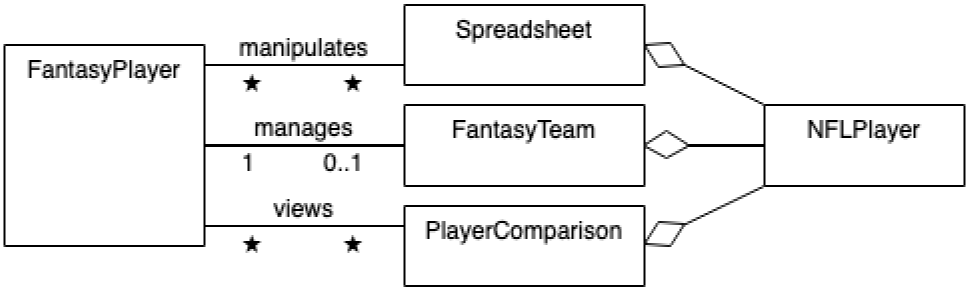
1. The system should support concurrent users.
2. The system should not present the wrong user's data and settings.
3. The system should store user information in a manner that is secure and respects the user’s privacy.
4. The system should use local storage to load, save, and maintain information associated with the user across sessions.
5. The system should load all pages within 2 seconds.
6. The system should load all requested data from the backend server within 5 seconds.
7. If data cannot be fetched from the backend server, the frontend should not crash.

III. Diagrams

Use Case Diagram

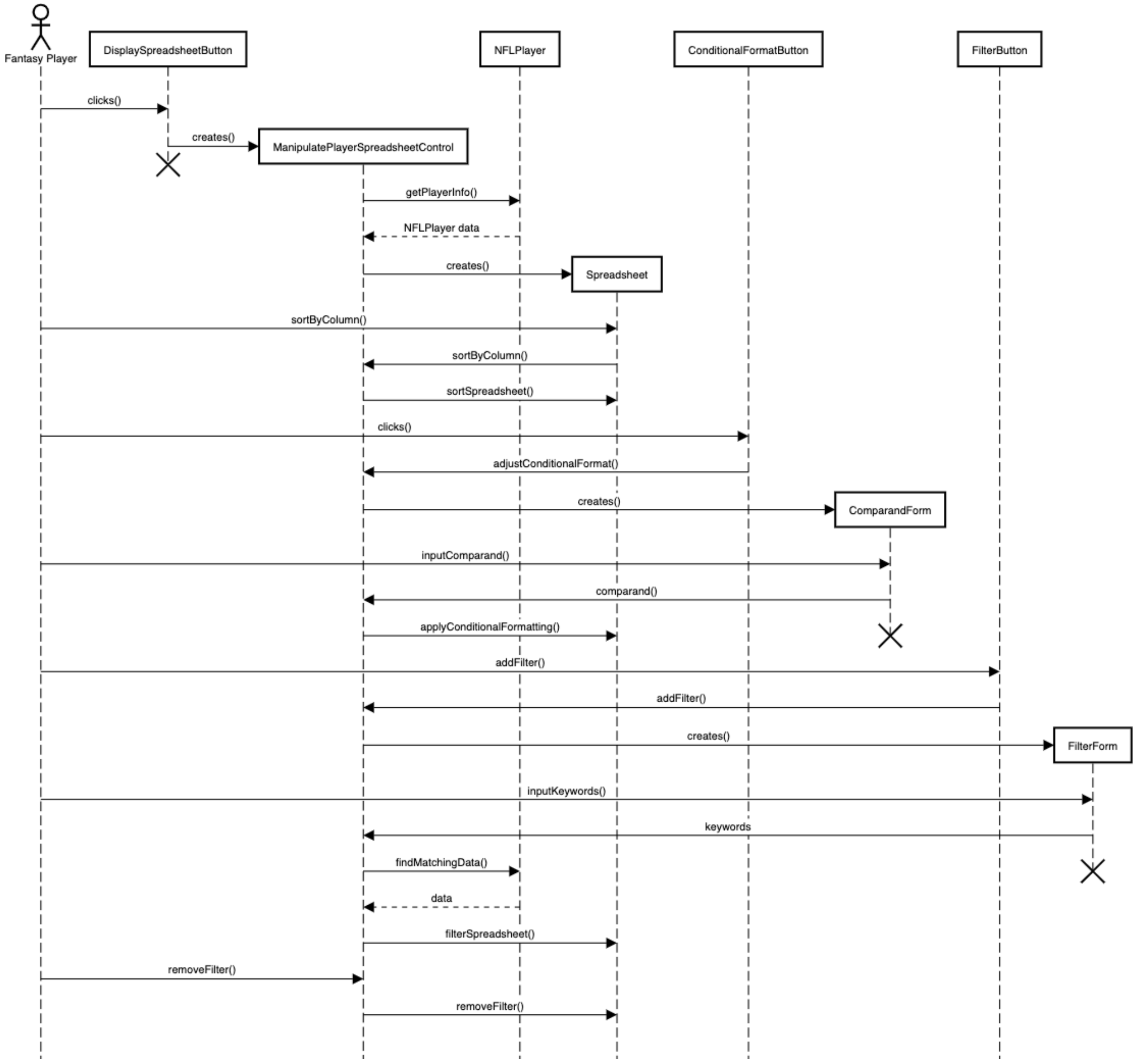


Class Diagram

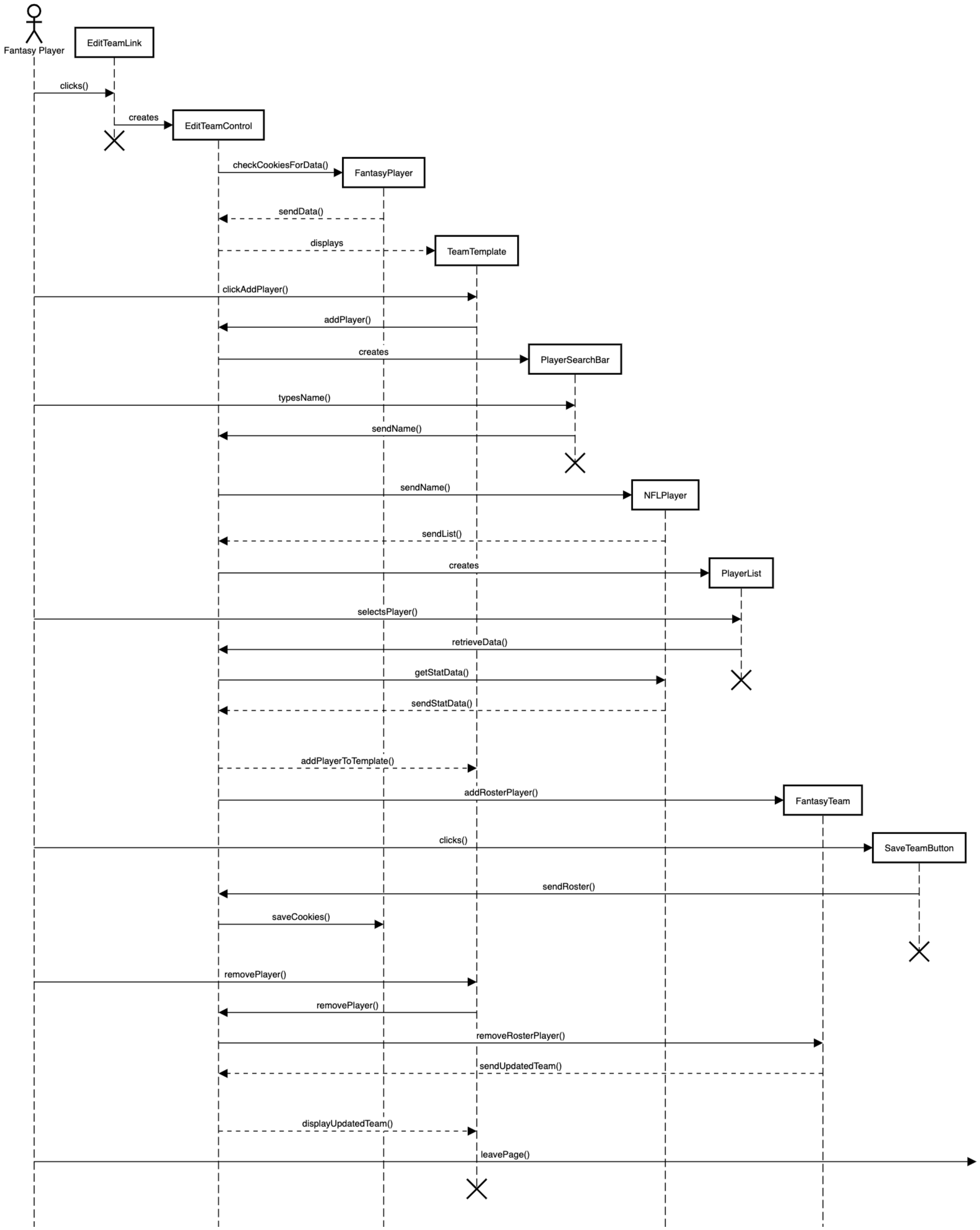


Sequence Diagrams

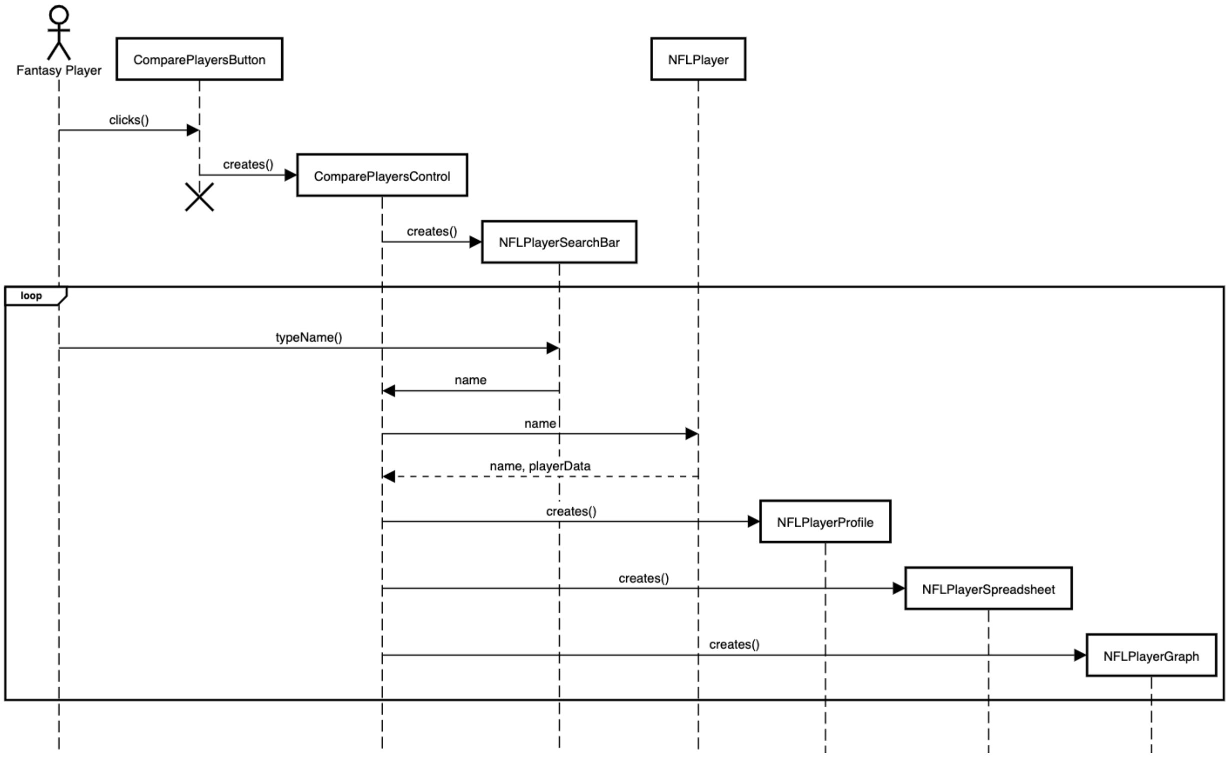
ManipulatePlayerSpreadsheet



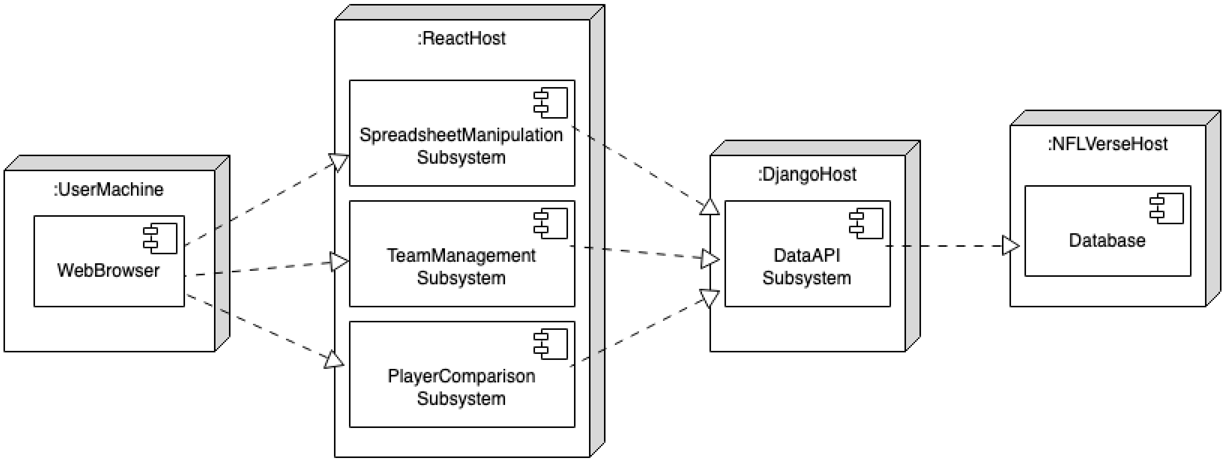
EditTeam


Fantasy Player
EditTeamLink
clicks()
creates
EditTeamControl
checkCookiesForData()
FantasyPlayer
sendData()
displays
TeamTemplate
clickAddPlayer()
addPlayer()
creates
PlayerSearchBar
typesName()
sendName()
sendName()
NFLPlayer
sendList()
creates
PlayerList
selectsPlayer()
retrieveData()
getStatData()
sendStatData()
addPlayerToTemplate()
addRosterPlayer()
FantasyTeam
clicks()
SaveTeamButton
sendRoster()
saveCookies()
removePlayer()
removePlayer()
removeRosterPlayer()
sendUpdatedTeam()
displayUpdatedTeam()
leavePage()

CompareTwoPlayers



Deployment Diagram



IV. Written Specifications

Name: ManipulatePlayerSpreadsheet

Participating actors: FantasyPlayer

Flow of events

1. The FantasyPlayer navigates to the page containing the spreadsheet of all NFL player data.

    2. The system communicates with the database to retrieve the data for the spreadsheet and displays the default view of the spreadsheet.

3. The FantasyPlayer hovers over a column header.

    4. The system provides hover text explaining the meaning of the statistics in that column.

5. The FantasyPlayer selects a column and requests to sort the spreadsheet by that column.

    6. The system sorts the spreadsheet by the selected column.

7. The FantasyPlayer selects a column and requests to apply conditional formatting to that column.

    8. The system asks the FantasyPlayer to provide a highlight color and a comparand to determine which cells in the column should be highlighted.

9. The FantasyPlayer selects a highlight color and inputs a comparand.

    10. If the comparand is not a valid integer, no conditional formatting will be applied.

    11. Otherwise, the system uses the provided comparand and highlight color to apply conditional formatting to the appropriate cells in the selected column.

12. The FantasyPlayer requests to filter the spreadsheet.

    13. The system asks the FantasyPlayer to provide one or more keywords by which to filter the spreadsheet.

14. The FantasyPlayer inputs one or more keywords by which to filter the spreadsheet.

    15. The system updates the spreadsheet to display only the matching data.

16. The FantasyPlayer requests to remove the spreadsheet data filter.

    17. The system displays the entire spreadsheet again.

Entry condition: The FantasyPlayer is able to navigate to the page containing the spreadsheet and the system is able to access up-to-date data on all active NFL players.

Exit condition:

- The spreadsheet will preserve the results of the FantasyPlayer's past manipulations of the data until the FantasyPlayer performs a further manipulation that overrides one or more of the past ones (such as sorting by a different column), reloads the page, or navigates to another page.

Name: CompareTwoPlayers

Participating actors: FantasyPlayer

Flow of events

1. The FantasyPlayer navigates to the player comparison page.
   2. The system displays the default view of the empty player comparison page. This includes two search bars that FantasyPlayer uses to search for NFL players.

3. The FantasyPlayer enters an NFL player's name into one of the search bars.
   4. The system offers autofill suggestions as the FantasyPlayer is typing.

5. The FantasyPlayer selects one of the autofill suggestions.
   6. The system retrieves the NFL player's data from the database.
   7. The system displays an image of the NFL player along with basic information.
   8. The rest of the NFL player's data is not displayed until a second search is completed.

9. The FantasyPlayer enters an NFL player's name into the other search bar.
   10. The system offers autofill suggestions as the FantasyPlayer is typing.

11. The FantasyPlayer selects one of the autofill suggestions.
    12. The system retrieves the NFL player's data from the database.
    13. The system displays an image of the NFL player along with basic information.
    14. The system displays a table comparing the weekly statistics of the two NFL players based on the position of the first player.
    15. The system presents a graph comparing the fantasy points of the two NFL players.

16. The FantasyPlayer selects a week for which to view the statistics of the NFL players in the table.
    17. The system updates the table to display the statistics for that week.

18. The FantasyPlayer hovers over one of the points plotted in the graph.
    19. The system presents information about the data represented by that point.

Entry condition: The FantasyPlayer is able to navigate to the compare NFL players page and the system is able to access up-to-date data on all active NFL players.

Exit condition:

- The comparison will preserve the FantasyPlayer's search results until the FantasyPlayer searches for a new player, reloads the page, or navigates to another page.

Name: EditTeam

Participating actors: FantasyPlayer

Flow of events

1. The FantasyPlayer navigates to the team builder page.
    2. The system prompts the FantasyPlayer to create a template for the team roster.

3. The FantasyPlayer creates a template for the team roster.
    4. The system builds and displays the roster based on the template.

5. The FantasyPlayer selects a position in the template to add an NFL player to their team.
    6. The system displays a search bar to find a specific NFL player.

7. The FantasyPlayer types in the name of that player in the search bar.
    8. The system offers autofill suggestions as the FantasyPlayer is typing.

9. The FantasyPlayer selects a player from the drop down list.
    10. The system adds the NFL player to the team and displays the relevant statistics.
    12. The system saves the current state of the FantasyPlayer's team in local storage.

13. The FantasyPlayer selects an NFL player in the team to remove.
    14. The system prompts the FantasyPlayer to confirm the removal of the NFL player.

15. The FantasyPlayer confirms the removal of the NFL player.
    16. The system removes the NFL player from the FantasyPlayer's team in local storage.
    17. The system removes the NFL player from the FantasyPlayer's team from the display.

18. The FantasyPlayer leaves the page but then returns at another time.
    19. The system loads and displays the previously built team from local storage.

20. The FantasyPlayer requests to change the roster template.
    21. The system warns the FantasyPlayer that all previous changes to the team will be discarded and asks the FantasyPlayer to confirm the change request.

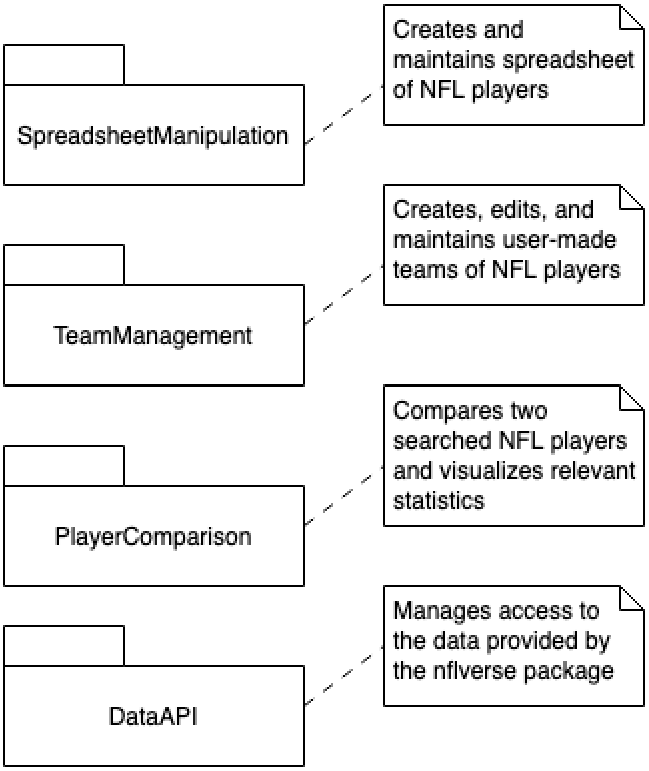
22. The FantasyPlayer confirms the change request.
    23. The system removes the previous team from local storage.
    24. The system prompts the user to create a template for the team roster.

Entry condition: The FantasyPlayer is able to navigate to the team builder page and the system is able to access and modify local storage and access up-to-date data on all active NFL players.

Exit condition:

- The FantasyPlayer's team will be saved in local storage so that upon returning to the team builder page the FantasyPlayer can resume editing the same team.

V. Subsystems


SpreadsheetManipulation
Creates and maintains spreadsheet of NFL players
TeamManagement
Creates, edits, and maintains user-made teams of NFL players
PlayerComparison
Compares two searched NFL players and visualizes relevant statistics
DataAPI
Manages access to the data provided by the nflverse package

VI. Interface Specifications

```
/**
 * PositionTable, which is part of the ManipulatePlayerSpreadsheet
 * subsystem, defines a table that displays data for all players in
 * a particular position. This table supports various manipulations
 * of the data such as filtering, sorting, and conditional formatting.
 */
public class PositionTable {

     /**
      * The position for this table.
      */
     private String position;

     /**
      * The data to display in this table.
      */
     private Collection<Player> data;

     /**
      * The columns to display in this table.
      */
     private Collection<Column> columns;

     /**
      * Constructor.
      * @pre data != null
      * @pre position != null
      * @pre for each Player p in data, p.position == position
      * @post columns != null
      */
     public PositionTable(data, position) {...}

     /**
      * Filter the table by the specified column.
      * @pre colIndex >= 0 && colIndex < columns.length
      * @post The table has been filtered by the specified column.
      */
     public void filter(int colIndex, String query) {...}
```

```
    /**
     * Sort the table by one or more columns in ascending or
     * descending order.
     * @pre for i in [0, colIndices.length), colIndices[i] >= 0
     * && colIndices[i] < columns.length
     * @post The table has been sorted in the specified order by
     * the specified column or columns.
     */
    public void sort(int[] colIndices, boolean[] descending) {...}

    /**
     * Set the range of values that should be highlighted in the
     * specified column.
     * @pre colIndex >= 0 && colIndex < columns.length
     * @post The range of values that should be highlighted in the
     * specified column has been updated.
     */
    public void setHighlightRange(int colIndex, float max,
          float min) {...}

    /**
     * Set the highlight color for the specified column.
     * @pre colIndex >= 0 && colIndex < columns.length
     * @post The highlight color for the specified column has been
     * updated.
     */
    public void setHighlightColor(int colIndex, String color) {...}
}
```

```
/**
 * RosterRow, which is part of the EditTeam subsystem, defines a row
 * in the user's roster and provides operations to add, remove, and
 * visualize the stats of the player in that row.
 */
public class RosterRow {

        /**
         * The data for the player in this roster row.
         * @invariant data.name == roster[rosterIndex]
         * @invariant if data != null, then validPlayer(data.name)
         */
        private Player data;

        /**
         * A collection of positions that are valid for this roster row.
         */
        private Collection<String> positions;

        /**
         * A reference to a collection containing the names of the
         * players who are in the roster.
         * @invariant roster[rosterIndex] == (data == null) ? null :
         * data.name
         */
        private Collection<String> roster;

        /**
         * The index of the entry in roster to which this roster row
         * corresponds.
         */
        private int rosterIndex;

        /**
         * Constructor.
         * @pre positions != null
         * @pre roster != null
         * @pre rosterIndex >= 0 && rosterIndex < roster.length
         * @post (roster[rosterIndex] == null)
         * ? data == null : data.name == roster[rosterIndex]
         */
        public RosterRow(positions, roster, rosterIndex) {...}
```

```
    /**
     * Add a player to this roster row.
     * Assumes that no player is currently in this roster row.
     * The player will only be added if validPlayer(playerName)
     * returns true.
     * @pre data == roster[rosterIndex] == null
     * @post if validPlayer(playerName), then
     * data.name == roster[rosterIndex] == playerName
     * @post if !validPlayer(playerName), then
     * data == roster[rosterIndex] == null
     */
    public void addPlayer(String playerName) {...}

    /**
     * Remove the player who is currently in this roster row.
     * Assumes that there is currently a player in this roster row.
     * @pre data != null and roster[rosterIndex] == data.name
     * @post data == roster[rosterIndex] == null
     */
    public void removePlayer() {...}

    /**
     * Get a TableRow that displays data for this player.
     * @pre data != null
     */
    public TableRow getTableRow() {...}

    /**
     * Determine whether or not the player with this name is
     * a valid match for this roster entry.
     * @pre positions != null
     */
    private boolean validPlayer(String playerName) {...}
}
```

```java
/**
 * CompareTwoPlayers, which is part of the CompareTwoPlayers
 * subsystem, allows the user to compare two NFLPlayers.
 * The user inputs two NFLPlayer names and is able to see and compare
 * their important statistics.
 */
public class CompareTwoPlayers {

      /**
       * The Player returned from the first search bar.
       * @invariant: if !validPlayer1, then player1 == emptyPlayer
       */
      private Player player1;

      /**
       * Indicates whether the input into the first search bar
       * is valid.
       */
      private boolean validPlayer1;

      /**
       * The Player returned from the second search bar.
       */
      private Player player2;

      /**
       * Indicates whether the input into the second search bar
       * is valid.
       * @invariant: if !validPlayer2, then player2 == emptyPlayer
       */
      private boolean validPlayer2;

      /**
       * Placeholder object for a player with no associated data.
       */
      private Player emptyPlayer;

      /**
       * Constructor.
       * @post player1 == player2 == emptyPlayer
       */
      public CompareTwoPlayers(){...}
```

```
    /**
     * Set player1 to the appropriate Player based on the search
     * query.
     * @post If a match for the search query was found, then
     * player1 is set to the data from that match. Otherwise,
     * player1 is set to emptyPlayer.
     * @post validPlayer1 = (player1 != emptyPlayer)
     */
    public void setPlayer1(Player player) {...}

    /**
     * Set player2 to the appropriate Player based on the search
     * query.
     * @post If a match for the search query was found, then
     * player2 is set to the data from that match. Otherwise,
     * player2 is set to emptyPlayer.
     * @post validPlayer1 = (player2 != emptyPlayer)
     */
    public void setPlayer2(Player player) {...}

    /**
     * Get a ComparisonTable that displays the data from the
     * two players in spreadsheet format based on their positions.
     * @pre validPlayer1 == true && validPlayer2 == true
     */
    public ComparisonTable
        getComparisonTable(Player p1, Player p2) {...}

    /**
     * Get a ComparisonChart that displays the data from the
     * two players in chart format. Displays a line chart graphing
     * the fantasy football points of each player over the course
     * of the current season.
     * @pre validPlayer1 == true && validPlayer2 == true
     */
    public ComparisonChart
        getComparisonChart(Player p1, Player p2) {...}
}
```

VII. Data Storage

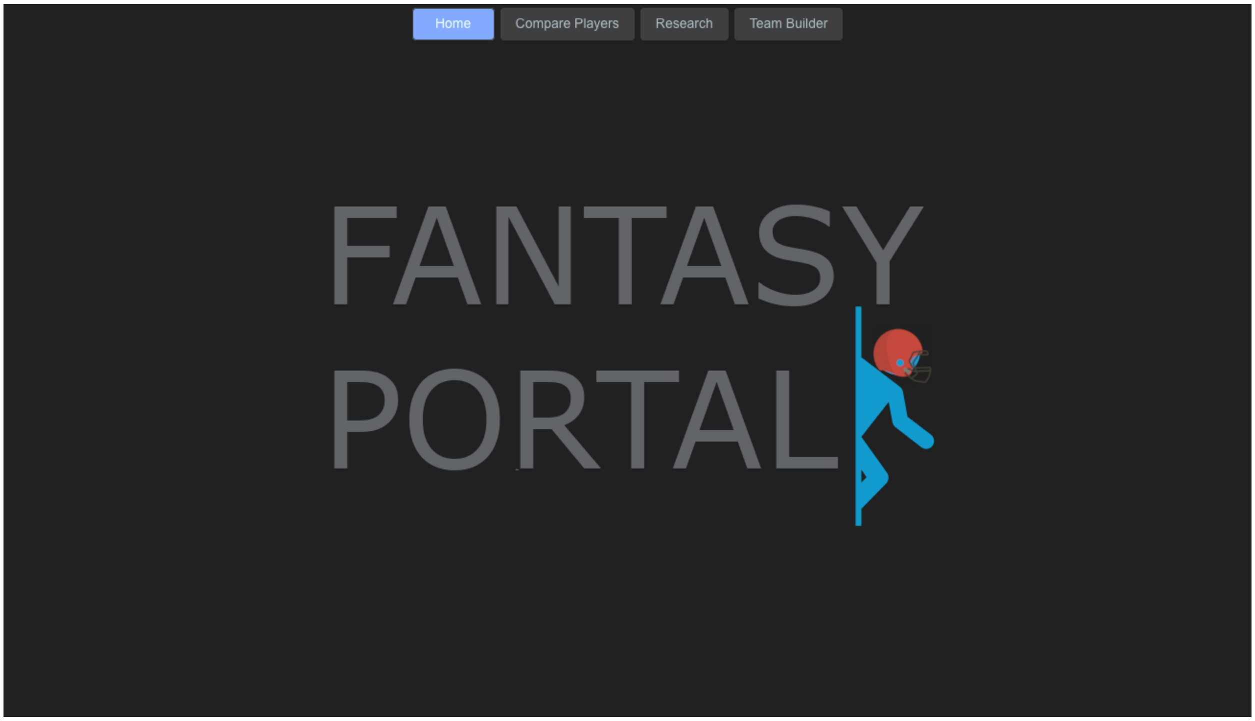
1. Information about the user's team is stored in local storage in the user's web browser. This information is maintained across browser sessions and is accessible to our entire web app. For more information about local storage, see this link: https://developer.mozilla.org/en-US/docs/Web/API/Storage.

2. All of our data about NFL players comes from the nflverse package in R. The DataAPI subsystem that accesses the data pulls from the GitHub repository at this link: https://github.com/nflverse/nflverse-data/releases/. The data is regularly updated by the maintainers of the package.

3. We have data stored locally in the file APIData.Rdata for improving the speed of the API. This file stores R tables with data for NFL players of the desired position so that the system can access this information more quickly. This data will be kept up-to-date by a regularly run script called ReloadData.R.

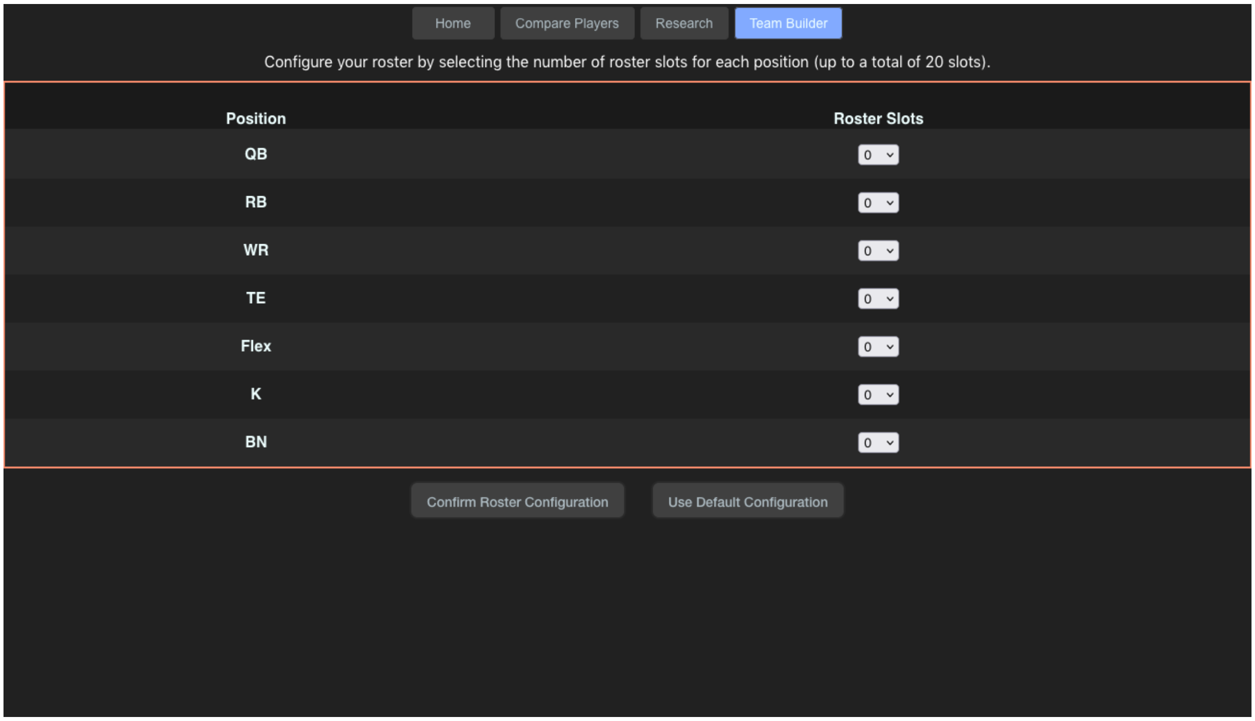
4. We have data stored locally in the file PlayerList.json to enable autocomplete suggestions for search bars. This file stores the names and positions of NFL players in the nflverse database and is used to autocomplete player names based on the user's queries.

VIII. User Manual



This is the homepage of the app. The navigation bar at the top allows the user to navigate between the three different pages in the app: Compare Players, Research, and Team Builder.

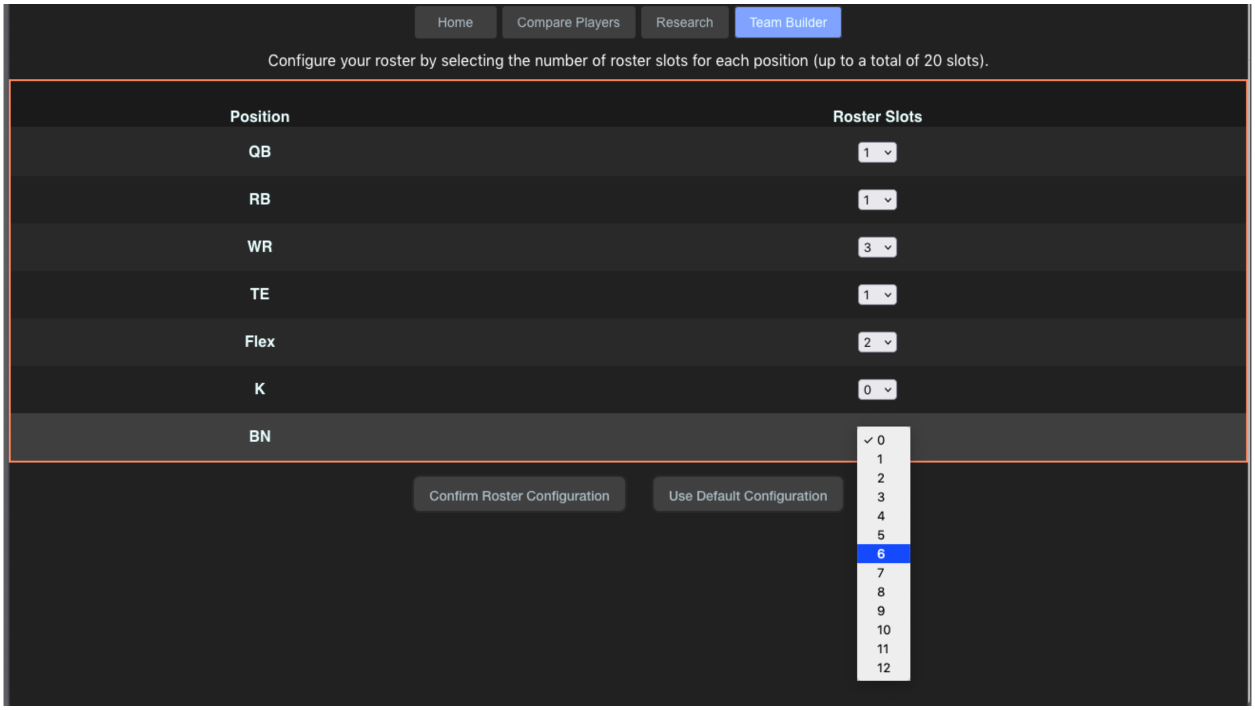
Team Builder



Configure your roster by selecting the number of roster slots for each position (up to a total of 20 slots).

Position	Roster Slots
QB	0
RB	0
WR	0
TE	0
Flex	0
K	0
BN	0

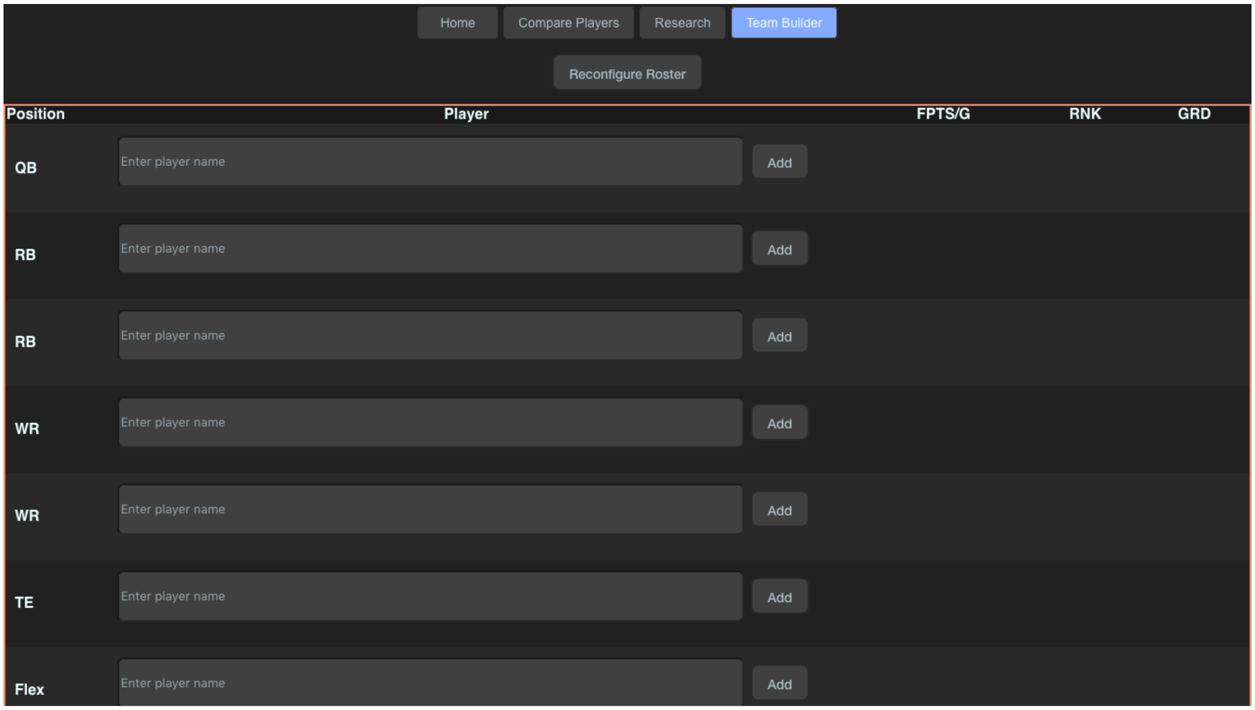
When the user first navigates to the Team Builder page, they will be presented with this roster configuration interface, which allows them to customize the configuration of their roster.



Configure your roster by selecting the number of roster slots for each position (up to a total of 20 slots).

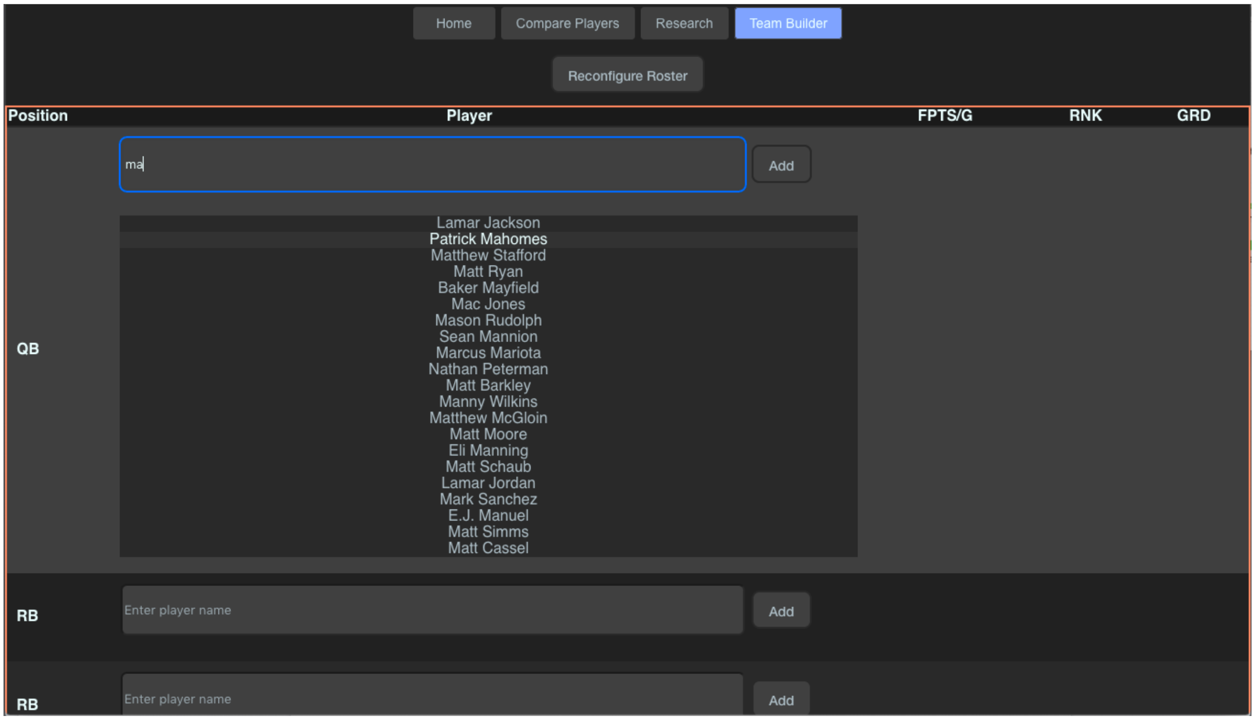
Position	Roster Slots
QB	1
RB	1
WR	3
TE	1
Flex	2
K	0
BN	

The user can customize their roster by using the dropdown menus to select the number of roster slots to assign to each NFL position, up to a total of 20 slots. When they are finished, they can click the button labeled “Confirm Roster Configuration” to save their custom configuration. Alternatively, they can opt to use a predefined, standard configuration by clicking the button “Use Default Configuration.” Once the user has clicked either button, the system will remember their configuration settings across sessions.



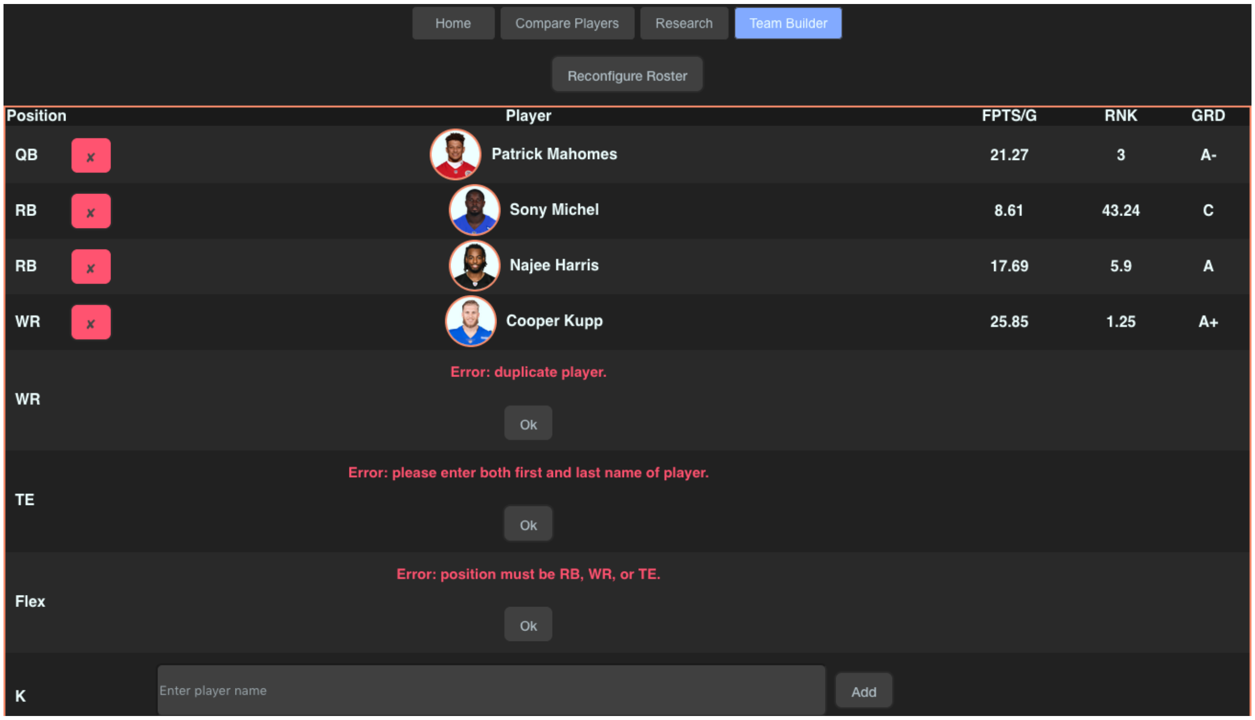
Position	Player	FPTS/G	RNK	GRD
QB	Enter player name Add			
RB	Enter player name Add			
RB	Enter player name Add			
WR	Enter player name Add			
WR	Enter player name Add			
TE	Enter player name Add			
Flex	Enter player name Add			

Once the user has saved their configuration settings, whether during the current session or a previous one, the page will automatically change to display the team builder interface, which allows the user to add and remove NFL players from their roster. The roster displayed in this interface is directly based on the user’s configuration settings. (In the screenshot, the “Use Default Configuration” option has been selected.) Any changes the user makes to their roster will be saved and restored during the next session.



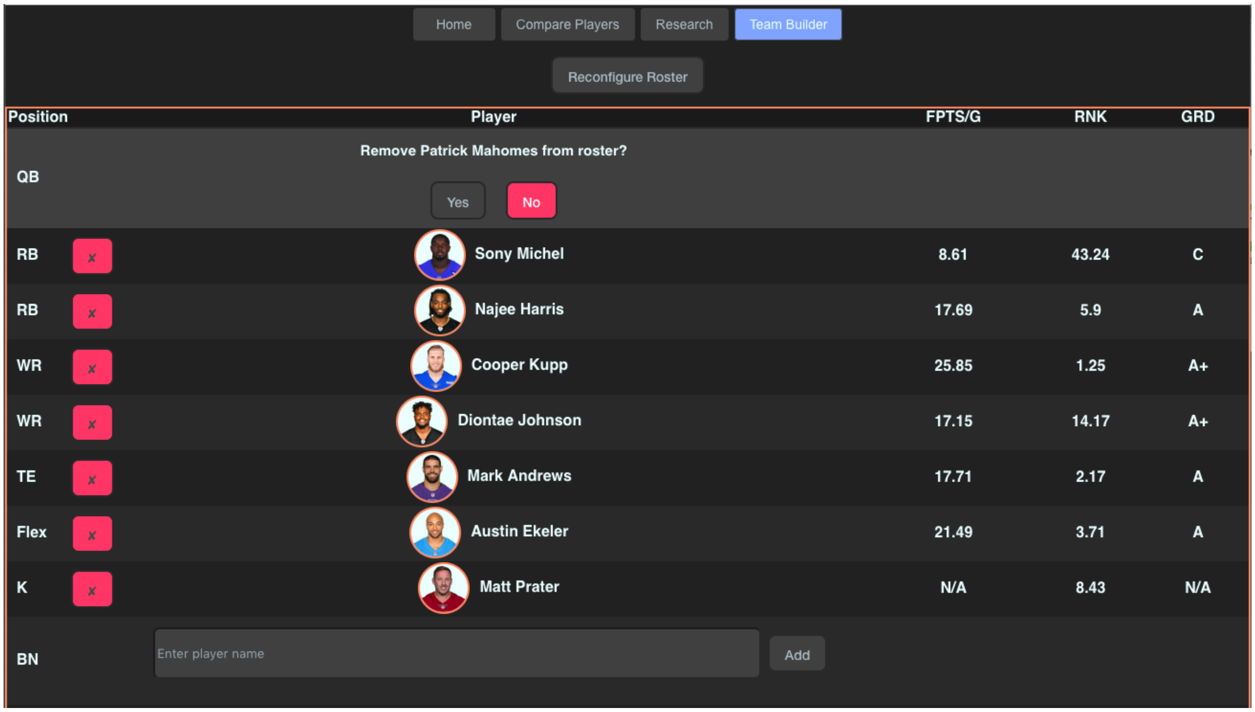
Position	Player	FPTS/G	RNK	GRD
QB	ma Add			
RB	Enter player name Add			
RB	Enter player name Add			

The user can search for a player to add to each roster slot by entering the player’s name into the search bar. A dropdown menu of autocomplete suggestions for players, filtered by the position for that slot, will be displayed based on the query. Once the user selects one of the autocomplete suggestions, hits the “Enter” key, or clicks on the “Add” button, their query will be submitted.



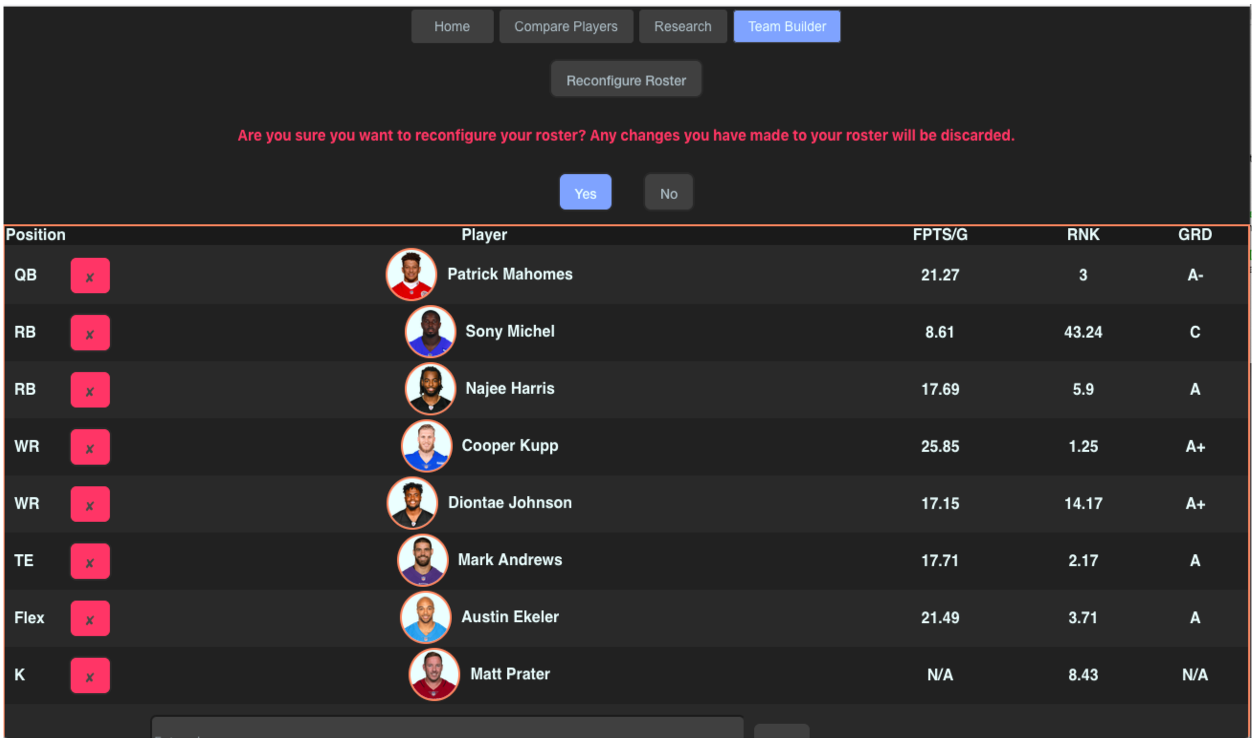
Position		Player	FPTS/G	RNK	GRD
QB	x	Patrick Mahomes	21.27	3	A-
RB	x	Sony Michel	8.61	43.24	C
RB	x	Najee Harris	17.69	5.9	A
WR	x	Cooper Kupp	25.85	1.25	A+
WR		Error: duplicate player. Ok			
TE		Error: please enter both first and last name of player. Ok			
Flex		Error: position must be RB, WR, or TE. Ok			
K		Enter player name Add			

Once a query is submitted, if an error occurs (for example, because the query was malformed or no player with that name could be found), or if the player is not a valid option for that roster slot (for example, the player has already been added to another roster slot, or the player's position is incompatible with this roster slot), then the system will alert the user and no player will be added. (In the screenshot, this has occurred with the "TE," "Flex," and second "WR" roster slots.) Otherwise, the player will be added to the roster, and the system will display relevant stats associated with that player. (In the screenshot, this has occurred with the "QB," "RB," and first "WR" roster slots.)



Position	Player	FPTS/G	RNK	GRD
QB	Remove Patrick Mahomes from roster?			
RB	Sony Michel	8.61	43.24	C
RB	Najee Harris	17.69	5.9	A
WR	Cooper Kupp	25.85	1.25	A+
WR	Diontae Johnson	17.15	14.17	A+
TE	Mark Andrews	17.71	2.17	A
Flex	Austin Ekeler	21.49	3.71	A
K	Matt Prater	N/A	8.43	N/A
BN				

The user can remove a player from the roster by clicking the red button to the left of that player's name. The system will then prompt the user to confirm or cancel the removal. (In the screenshot, this has occurred with the "QB" roster slot.)

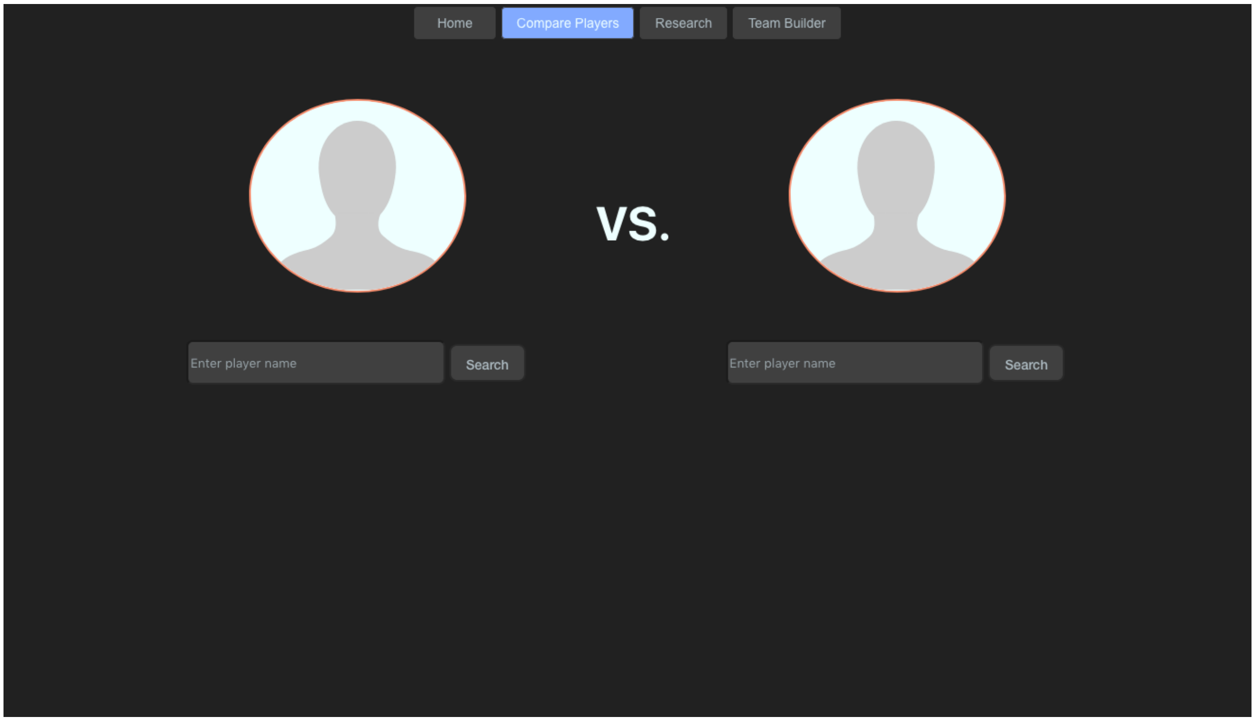


Are you sure you want to reconfigure your roster? Any changes you have made to your roster will be discarded.

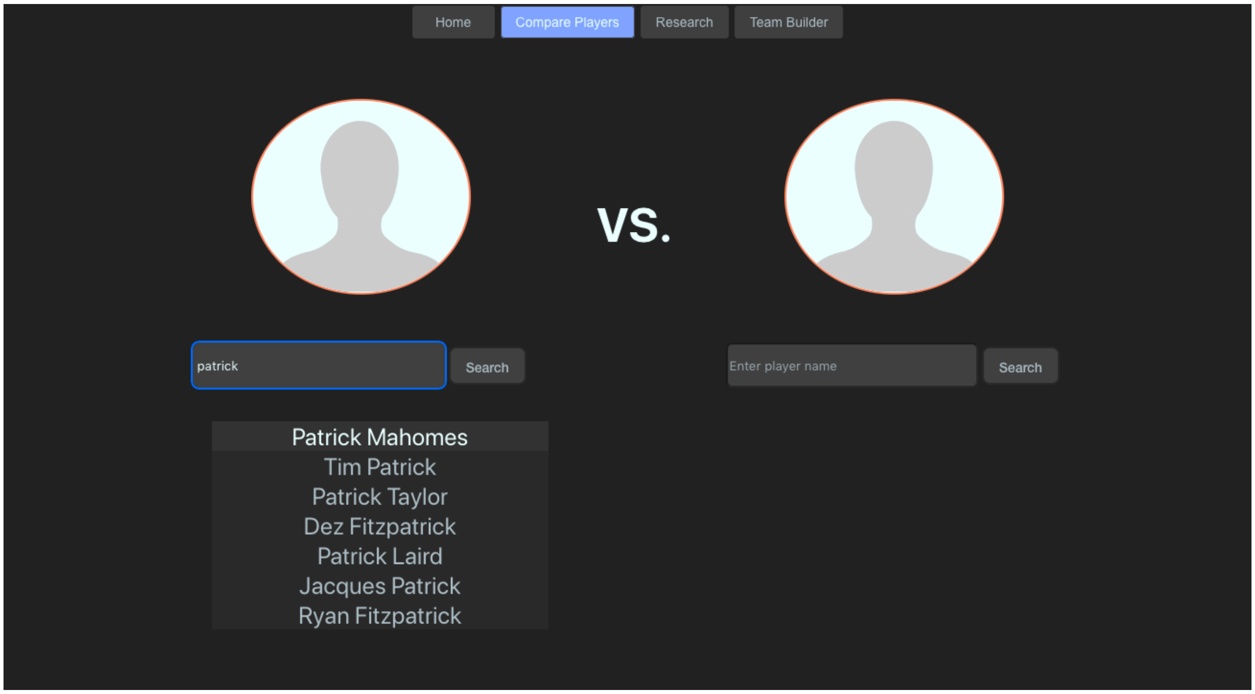
Position		Player	FPTS/G	RNK	GRD
QB	x	Patrick Mahomes	21.27	3	A-
RB	x	Sony Michel	8.61	43.24	C
RB	x	Najee Harris	17.69	5.9	A
WR	x	Cooper Kupp	25.85	1.25	A+
WR	x	Diontae Johnson	17.15	14.17	A+
TE	x	Mark Andrews	17.71	2.17	A
Flex	x	Austin Ekeler	21.49	3.71	A
K	x	Matt Prater	N/A	8.43	N/A

The user can change their roster settings by clicking the “Reconfigure Roster” button. The system will respond by warning the user that reconfiguring the roster will cause any changes that have been made to the roster to be discarded. The system will then prompt the user to confirm or cancel the action by selecting “Yes” or “No.” If the user clicks “Yes,” then their previous changes to the roster will be discarded and they will be taken back to the roster configuration interface.

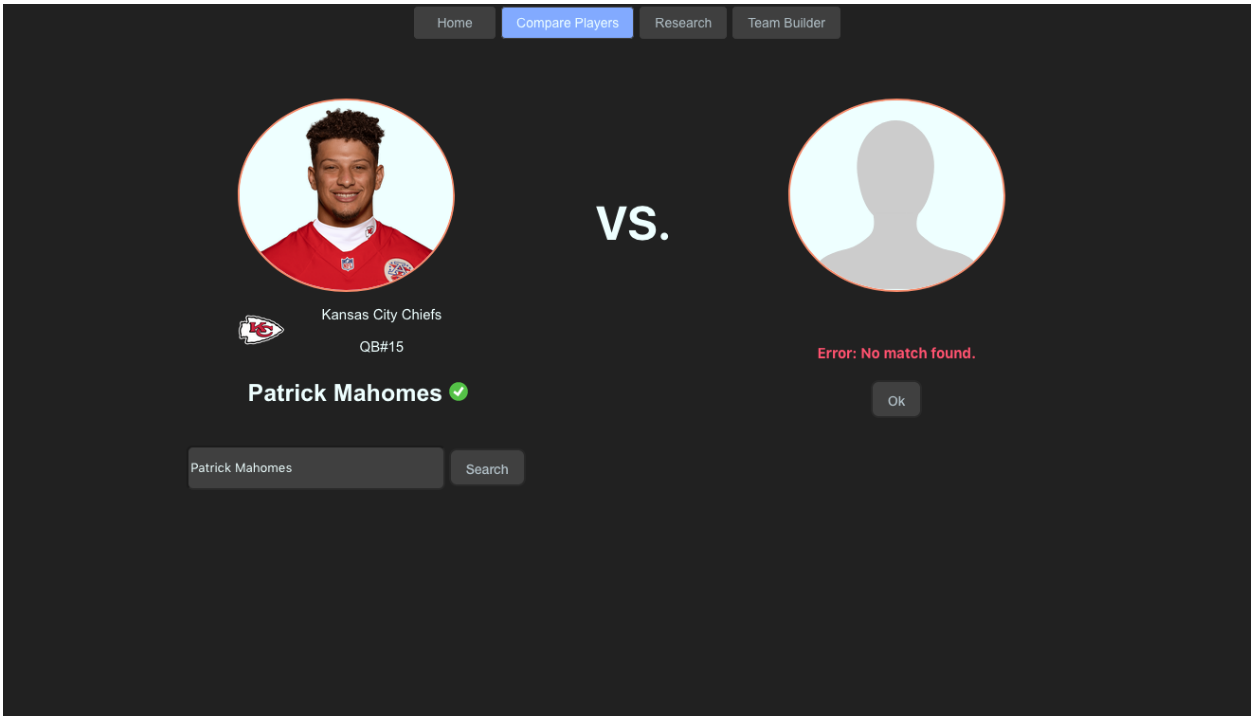
Compare Players



When the user first navigates to the Compare Players page, they will be presented with this interface, which allows them to search for and compare two NFL players at a time.

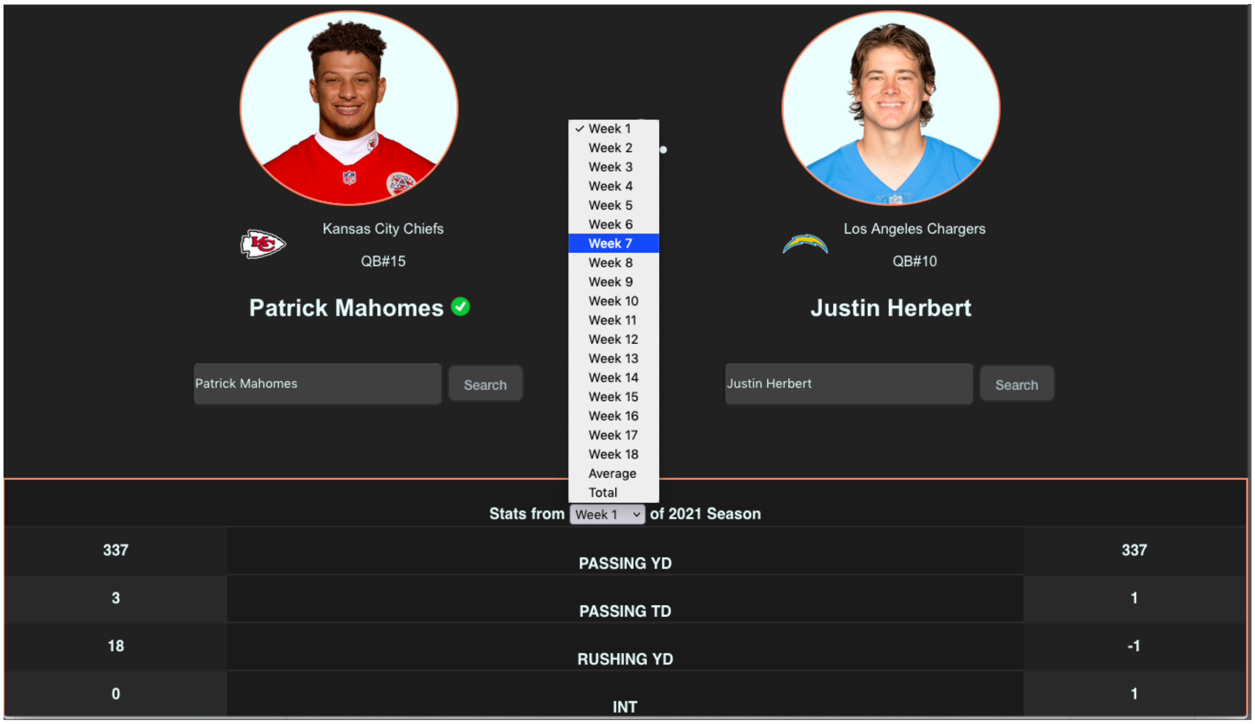


A dropdown menu of autocomplete suggestions for players in relevant positions will be displayed based on the query. Once the user selects one of the autocomplete suggestions, hits the “Enter” key, or clicks on the “Add” button, their query will be submitted.



Error: No match found.

If the user’s query is malformed, or if no player with that name could be found, then the system will alert the user. (In the screenshot, this has occurred with the search bar on the right.) Otherwise, relevant information about that player, including a photo, team information, a table of statistics, and a chart, will be displayed. (In the screenshot, this has occurred with the search bar on the left.)



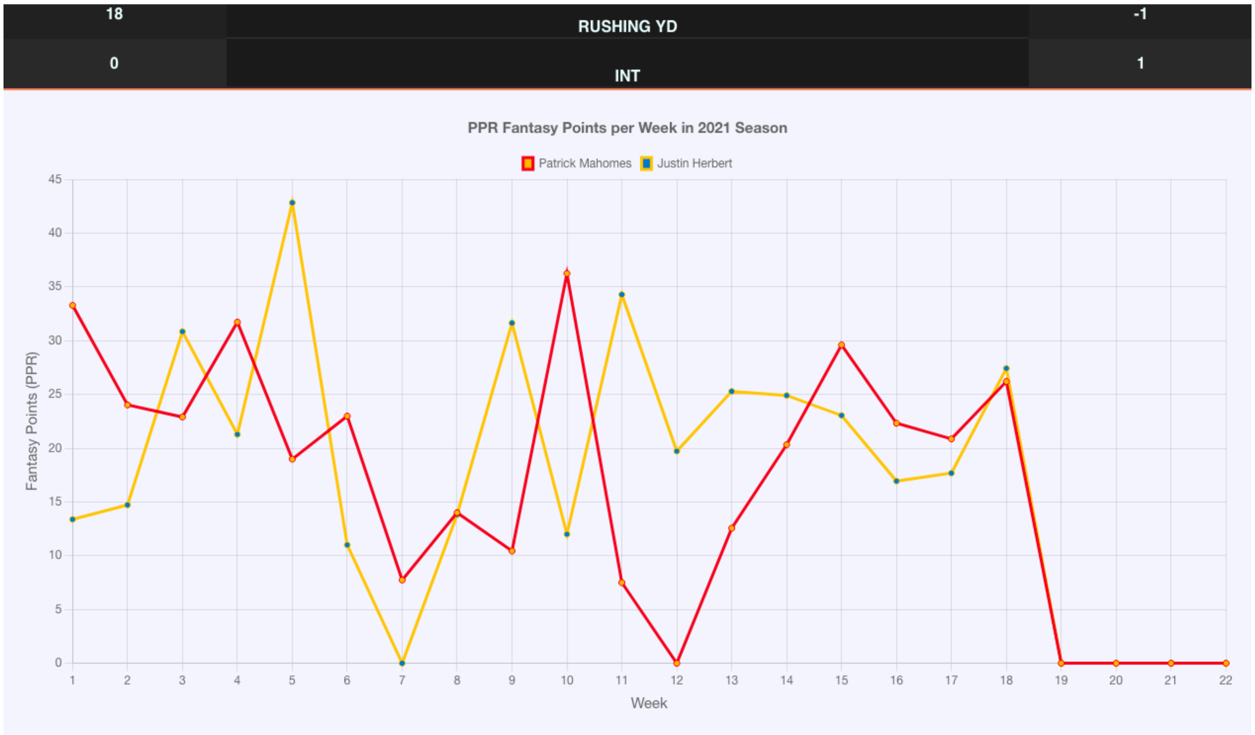
Patrick Mahomes

Justin Herbert

Stats from Week 1 of 2021 Season

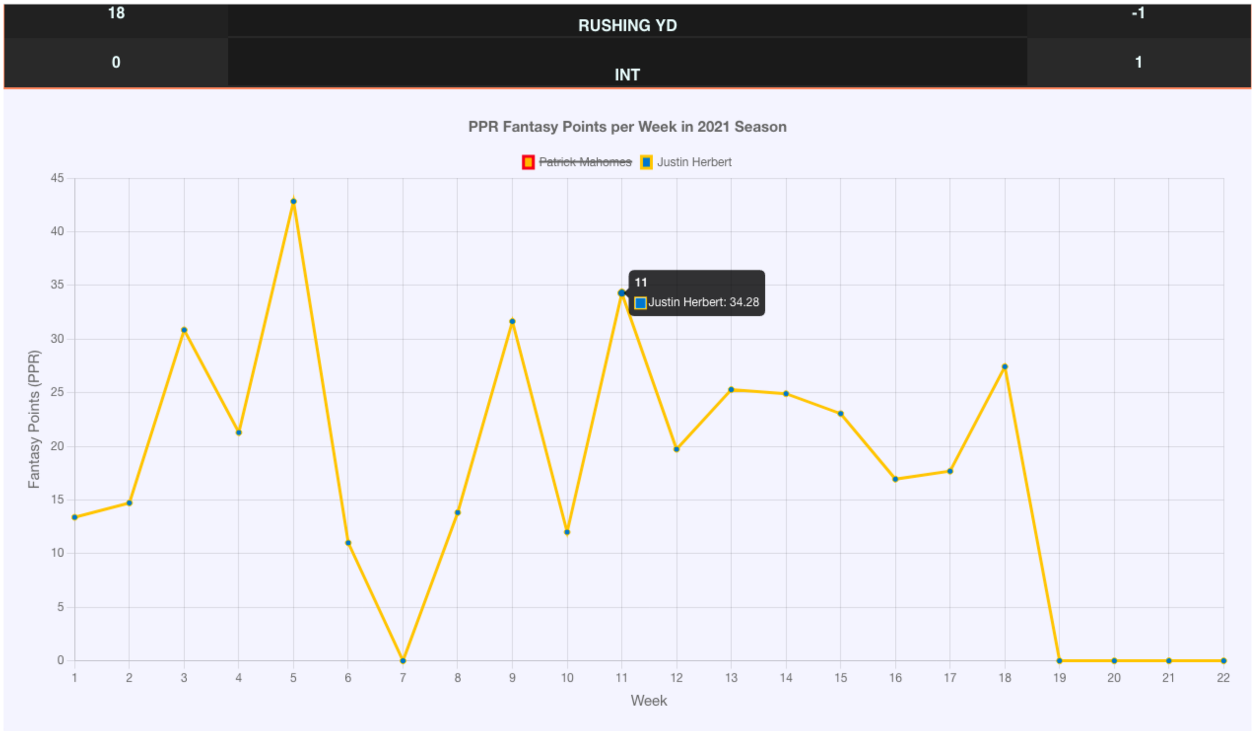
337	PASSING YD	337
3	PASSING TD	1
18	RUSHING YD	-1
0	INT	1

Once two players have been selected, a table of stats relevant to those players’ positions will be displayed. The user can use the dropdown menu to choose between weekly, average, and total views of the data.



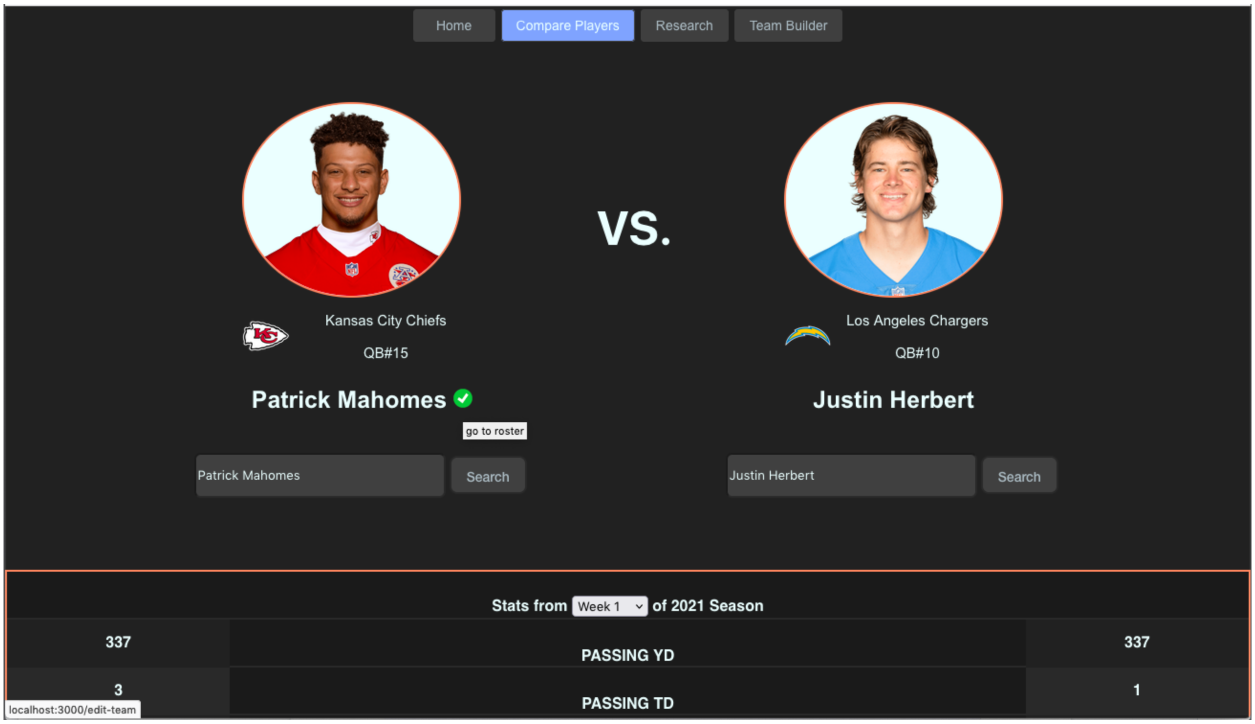
18	RUSHING YD	-1
0	INT	1

Once two players have been selected, a chart graphing their fantasy points per week will be displayed underneath the table.



18	RUSHING YD	-1
0	INT	1

The chart is interactive and will display information about each data point when the user hovers over it. Additionally, the user can click on each player's name to hide or show the line in the graph that corresponds to that player. (In the screenshot, the line corresponding to the player on the left has been hidden.)



Patrick Mahomes

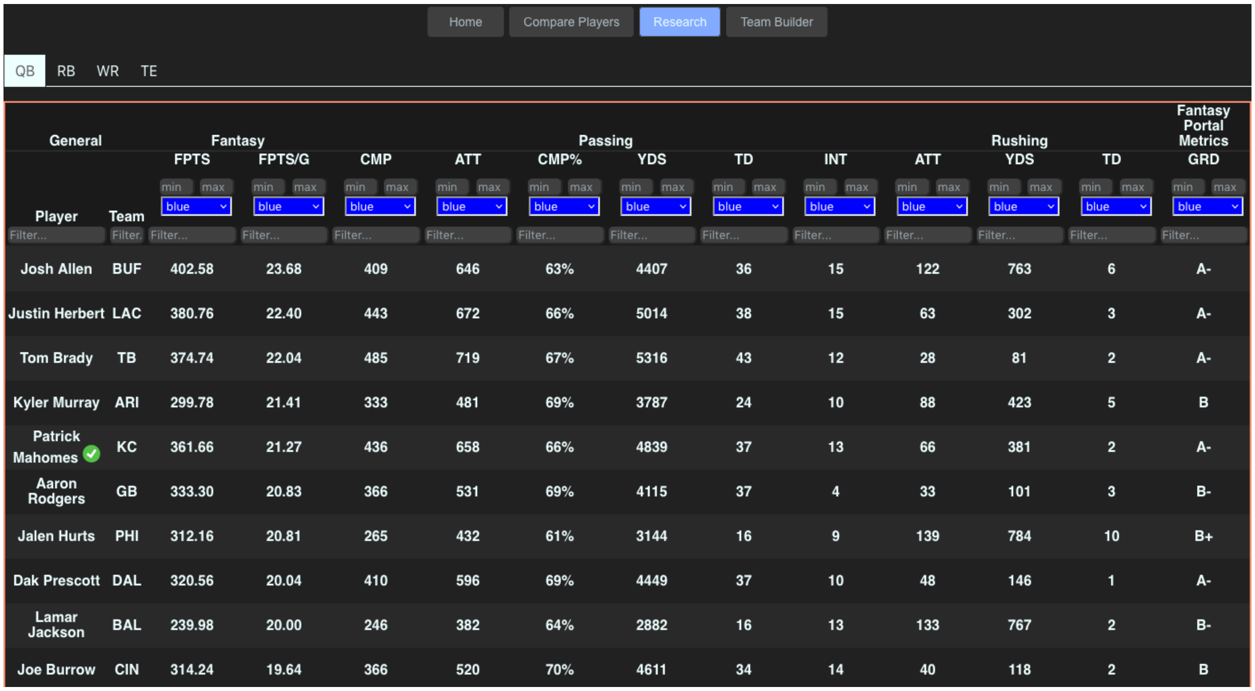
Justin Herbert

Stats from Week 1 of 2021 Season

337	PASSING YD	337
3	PASSING TD	1

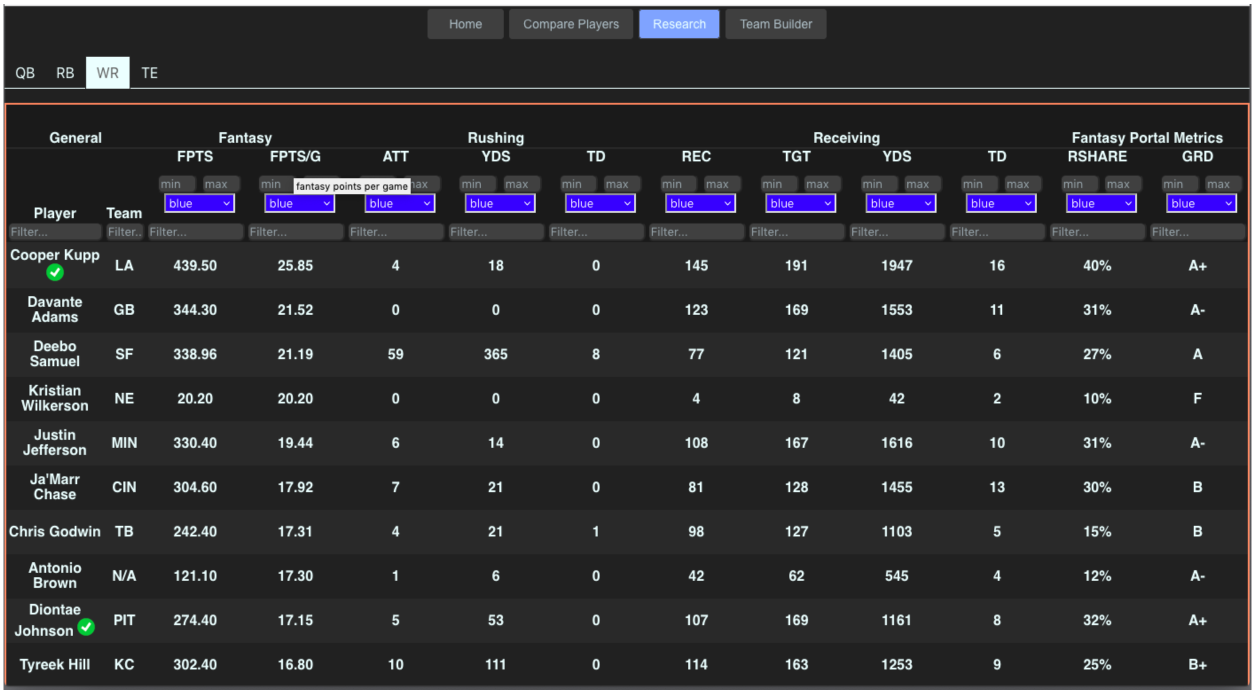
If a player has been added to the user’s roster, then a green check mark will appear next to that player’s name. This check mark will appear or disappear in real time based on the user’s edits to their roster without the user needing to reload the page. If the user clicks on the check mark, then they will be taken to the Team Builder page in a new tab.

Research



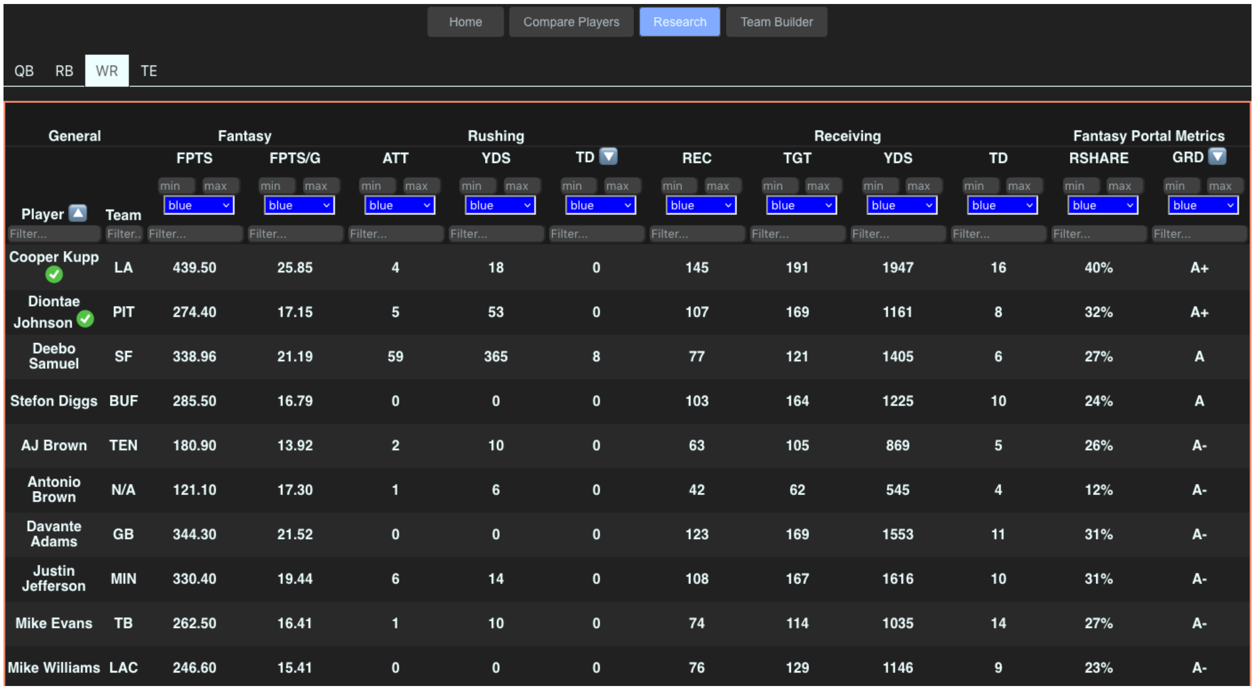
General		Fantasy		Passing						Rushing			Fantasy Portal Metrics
		FPTS	FPTS/G	CMP	ATT	CMP%	YDS	TD	INT	ATT	YDS	TD	GRD
Player	Team	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue
Filter...	Filter.	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...
Josh Allen	BUF	402.58	23.68	409	646	63%	4407	36	15	122	763	6	A-
Justin Herbert	LAC	380.76	22.40	443	672	66%	5014	38	15	63	302	3	A-
Tom Brady	TB	374.74	22.04	485	719	67%	5316	43	12	28	81	2	A-
Kyler Murray	ARI	299.78	21.41	333	481	69%	3787	24	10	88	423	5	B
Patrick Mahomes ✓	KC	361.66	21.27	436	658	66%	4839	37	13	66	381	2	A-
Aaron Rodgers	GB	333.30	20.83	366	531	69%	4115	37	4	33	101	3	B-
Jalen Hurts	PHI	312.16	20.81	265	432	61%	3144	16	9	139	784	10	B+
Dak Prescott	DAL	320.56	20.04	410	596	69%	4449	37	10	48	146	1	A-
Lamar Jackson	BAL	239.98	20.00	246	382	64%	2882	16	13	133	767	2	B-
Joe Burrow	CIN	314.24	19.64	366	520	70%	4611	34	14	40	118	2	B

When the user first navigates to the Research page, they will be presented with this interface, which allows them to view and manipulate data about NFL players in spreadsheet form.



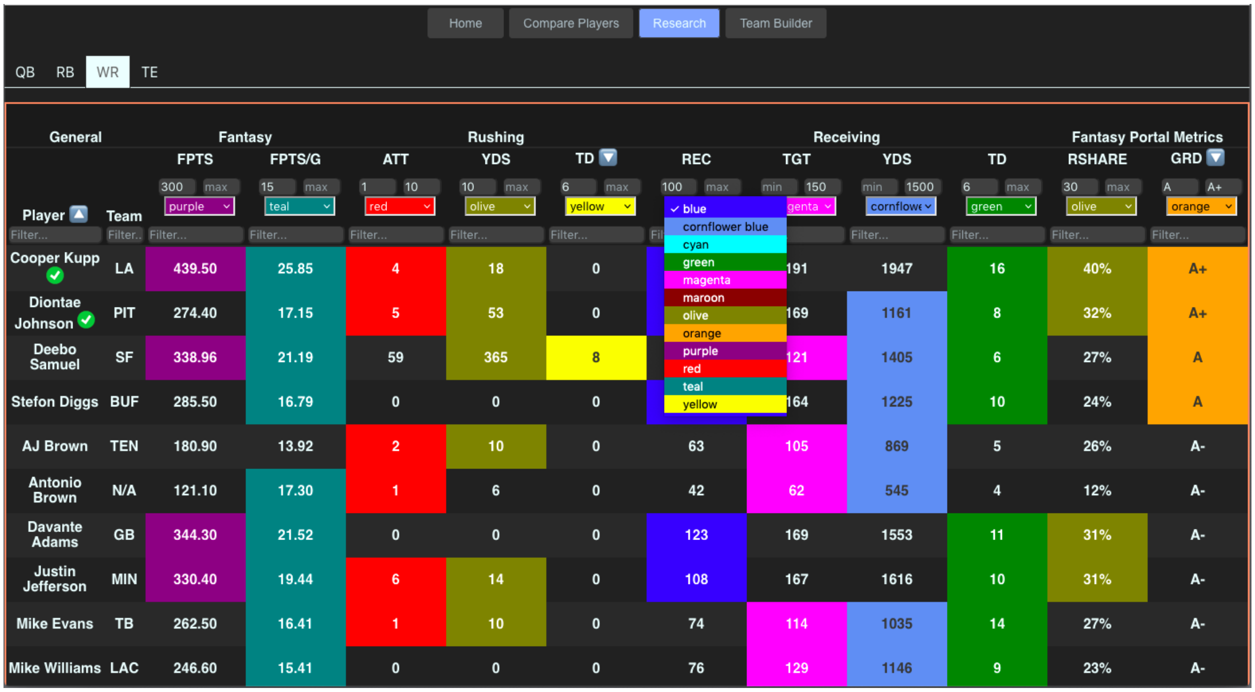
General		Fantasy		Rushing			Receiving				Fantasy Portal Metrics	
Player	Team	FPTS	FPTS/G	ATT	YDS	TD	REC	TGT	YDS	TD	RSHARE	GRD
Cooper Kupp ✓	LA	439.50	25.85	4	18	0	145	191	1947	16	40%	A+
Davante Adams	GB	344.30	21.52	0	0	0	123	169	1553	11	31%	A-
Deebo Samuel	SF	338.96	21.19	59	365	8	77	121	1405	6	27%	A
Kristian Wilkerson	NE	20.20	20.20	0	0	0	4	8	42	2	10%	F
Justin Jefferson	MIN	330.40	19.44	6	14	0	108	167	1616	10	31%	A-
Ja'Marr Chase	CIN	304.60	17.92	7	21	0	81	128	1455	13	30%	B
Chris Godwin	TB	242.40	17.31	4	21	1	98	127	1103	5	15%	B
Antonio Brown	N/A	121.10	17.30	1	6	0	42	62	545	4	12%	A-
Diontae Johnson ✓	PIT	274.40	17.15	5	53	0	107	169	1161	8	32%	A+
Tyreek Hill	KC	302.40	16.80	10	111	0	114	163	1253	9	25%	B+

The user can click on the tabs at the top left to switch between spreadsheets for various positions. When the user hovers over any column header, hover text will appear to give more information about the stats that are displayed in that column. (In the screenshot, this has occurred with the PPTS/G column.) As with the Compare Players page, a green check mark will appear next to the name of any player who has been added to the user’s roster.



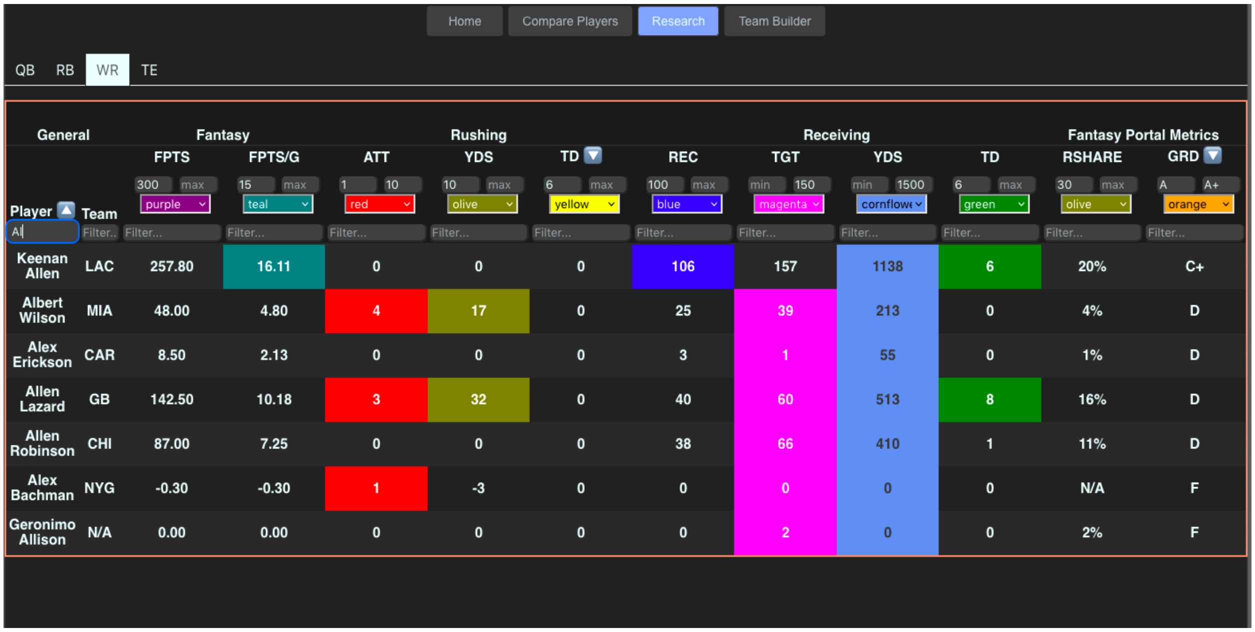
General		Fantasy		Rushing			Receiving				Fantasy Portal Metrics	
Player ▲	Team	FPTS	FPTS/G	ATT	YDS	TD ▼	REC	TGT	YDS	TD	RSHARE	GRD ▼
		min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue	min max blue
Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...
Cooper Kupp ✓	LA	439.50	25.85	4	18	0	145	191	1947	16	40%	A+
Diontae Johnson ✓	PIT	274.40	17.15	5	53	0	107	169	1161	8	32%	A+
Deebo Samuel	SF	338.96	21.19	59	365	8	77	121	1405	6	27%	A
Stefon Diggs	BUF	285.50	16.79	0	0	0	103	164	1225	10	24%	A
AJ Brown	TEN	180.90	13.92	2	10	0	63	105	869	5	26%	A-
Antonio Brown	N/A	121.10	17.30	1	6	0	42	62	545	4	12%	A-
Davante Adams	GB	344.30	21.52	0	0	0	123	169	1553	11	31%	A-
Justin Jefferson	MIN	330.40	19.44	6	14	0	108	167	1616	10	31%	A-
Mike Evans	TB	262.50	16.41	1	10	0	74	114	1035	14	27%	A-
Mike Williams	LAC	246.60	15.41	0	0	0	76	129	1146	9	23%	A-

The user can click on a column header to sort in ascending or descending order. The user can sort by multiple columns at the same time by holding down the Shift key. Each column has a predetermined initial sort order that is appropriate for the type of information that it contains (for example, player names are initially sorted in ascending order, whereas fantasy points are initially sorted in descending order). The user can invert the sort order by clicking on the column header for a second time, and can return the column back to its presorted state by clicking for a third time. (In the screenshot, the spreadsheet has been primarily sorted by the GRD column in descending order, secondarily sorted by the Rushing TD column in descending order, and tertiarily sorted by the Player column in ascending order.)



General		Fantasy		Rushing			Receiving				Fantasy Portal Metrics	
		FPTS	FPTS/G	ATT	YDS	TD ▼	REC	TGT	YDS	TD	RSHARE	GRD ▼
		300 max	15 max	1 10	10 max	6 max	100 max	min 150	min 1500	6 max	30 max	A A+
Player ▲	Team	purple	teal	red	olive	yellow	blue	genta	cornflowe	green	olive	orange
Filter...	Filter..	Filter...	Filter...	Filter...	Filter...	Filter...	Fil		Filter...	Filter...	Filter...	Filter...
Cooper Kupp ✓	LA	439.50	25.85	4	18	0		191	1947	16	40%	A+
Diontae Johnson ✓	PIT	274.40	17.15	5	53	0		169	1161	8	32%	A+
Deebo Samuel	SF	338.96	21.19	59	365	8		121	1405	6	27%	A
Stefon Diggs	BUF	285.50	16.79	0	0	0		164	1225	10	24%	A
AJ Brown	TEN	180.90	13.92	2	10	0	63	105	869	5	26%	A-
Antonio Brown	N/A	121.10	17.30	1	6	0	42	62	545	4	12%	A-
Davante Adams	GB	344.30	21.52	0	0	0	123	169	1553	11	31%	A-
Justin Jefferson	MIN	330.40	19.44	6	14	0	108	167	1616	10	31%	A-
Mike Evans	TB	262.50	16.41	1	10	0	74	114	1035	14	27%	A-
Mike Williams	LAC	246.60	15.41	0	0	0	76	129	1146	9	23%	A-

The user can apply conditional formatting to each column by specifying a range and a highlight color. All values in that range will be highlighted with the specified color. The user can define the range for each column by entering inclusive minimum and maximum bounds into the text boxes beneath each column header; if the user inputs two invalid bounds, then no highlight will be applied, but if the user inputs one invalid bound, it will be treated as negative or positive infinity. The user can use the dropdown menu beneath each column header to select the highlight color for that column.



General		Fantasy		Rushing			Receiving				Fantasy Portal Metrics	
		FPTS	FPTS/G	ATT	YDS	TD	REC	TGT	YDS	TD	RSHARE	GRD
		300 max	15 max	1 10	10 max	6 max	100 max	min 150	min 1500	6 max	30 max	A A+
Player	Team	purple	teal	red	olive	yellow	blue	magenta	cornflowe	green	olive	orange
Al	Filter..	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...
Keenan Allen	LAC	257.80	16.11	0	0	0	106	157	1138	6	20%	C+
Albert Wilson	MIA	48.00	4.80	4	17	0	25	39	213	0	4%	D
Alex Erickson	CAR	8.50	2.13	0	0	0	3	1	55	0	1%	D
Allen Lazard	GB	142.50	10.18	3	32	0	40	60	513	8	16%	D
Allen Robinson	CHI	87.00	7.25	0	0	0	38	66	410	1	11%	D
Alex Bachman	NYG	-0.30	-0.30	1	-3	0	0	0	0	0	N/A	F
Geronimo Allison	N/A	0.00	0.00	0	0	0	0	2	0	0	2%	F

The user can filter the data in the spreadsheet by typing keywords into the search bars beneath each column header. The user can filter by multiple keywords in multiple columns at a time. (In the screenshot, the data has been filtered by the keyword “Al” in the Player column.)