# Product Backlog (refined for sprint 3)

| Task | Estimation | TODO | WIP | Completed | ID |
|---|---|---|---|---|---|
| As a user, I want to be able to select a match and see different representations of the match history. Estimation: 40 Priority: 2 | 1 hour |  | JS - use inputs from key events to change between graphs and adapt data accordingly (pair programming) | HTML and CSS | 1 |
| As a user I want to be able to see key events in the match. Estimation: 20 (may remove) Priority: 3 | 20 minutes |  | Clean up the CSS. | HTML<br>CSS<br>JS - use API and react to changes in the match as it progresses (pair programming) | 2 |
| As a user, I want to be able to view the location and time of the match in progress. Estimation: 3 | 10 minutes |  | Js - update the time along with location. | HTML<br>CSS<br>JS - use data from API and backend (pair programming) | 3 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Priority: 2 |  |  |  |  |  |
| As a user, I want to be able to change the statistic being measured (ie goals, red cards), so that I can find the player stats I want.<br>Estimation: 3<br>Priority: 2 | 20 minutes |  |  | HTML<br>CSS<br>JS - use inputs to change/adapt on-screen representation. | 4 |
| As a user I want to be able to change the season of the stats.<br>Estimation: 8<br>Priority: 3 | 50 minutes |  |  | HTML<br>CSS<br>JS - use inputs to change/adapt on-screen representation. | 5 |
| As a user I want to be able to select a stat mode (total or average or running-monthly-average) - average is dictated by a user input on a slider, specifying the games that are being | 1 hour and 20 minutes |  | JS- use the API and JS to change the graphs as the user selects a different team | HTML<br>CSS | 6 |

| | | | | | |
|---|---|---|---|---|---|
| averaged. Estimation: 13<br>Priority: 2 | | | | | |
| As a user I want to be able to select a player. Estimation: 1<br>Priority: 2 | 20 minutes | | JS- using JS to update the graph with player stats accordingly. | HTML<br>CSS | 7 |
| As a user I want a table that has the top 10 players in the league for a selected stat<br>Estimation: 1<br>Priority: 3 | 20 minutes | | | HTML<br>CSS<br>JS - using JS to format and present data | 8 |
| As a user, I want to input a period of time in the future to extrapolate relevant data. Estimation: 40<br>Priority: 4 | 4 hours | JS - use JS to create regression analysis and predict data in the period specified. | | HTML<br>CSS | 9 |
| As a user, I want to predict who will score the most, who will get the most assists. Estimation: | 4 hours | JS - use JS to create regression analysis and predict data in the period specified. | | HTML<br>CSS | 10 |

| | | | | | |
|---|---|---|---|---|---|
| 40<br>Priority: 3 | | | | | |
| As a user, I want to predict the results of next year. Estimation: 20<br>Priority: 4 | 2 hours | JS - use JS to create regression analysis and predict data in the period specified. | | HTML<br>CSS | 11 |
| As a user I want to be able to change the chart type so that I get the right representation.<br>Estimation: 13<br>Priority: 2 | 1 hour and 20 minutes | | JS - Change the chart variable inside the chart object. If pie charts are also included as an option some data manipulation will be required | HTML<br>CSS | 12 |
| As a user I want to be able to select a team so that I can view their match history.<br>Estimation: 13<br>Priority: 3 | 2 hours | | JS - using JS to update this seasons match history and being able to use data from an API to display the team's entire match history across seasons. | HTML<br>CSS | 13 |
| As a user, I want to be able to view the entire league table so that I | 30 minutes | | | HTML<br>CSS<br>JS- Using JS to get | 14 |

| | | | | | |
|---|---|---|---|---|---|
| know the standings<br>Estimation: 5<br>Priority: 4 | | | | the data from the API's and be able to present them in a table. | |
| As a user, I want to be able to see my favourite team's upcoming matches so that I am updated.<br>Estimation: 8<br>Priority: 5 | 1 hour and 20 minutes | JS - use localStorage to remember specified favourite team, and present data dynamically depending on API calls. | | | 15 |
| Learning how to use the chosen APIs<br>Estimation: 5<br>Priority: 1 | 30 minutes | | | Study how to query data and what data is given back<br>JS - spike code to test calls and provide information. | 16 |
| HTML navigation<br>Estimation: 5<br>Priority: 1 | 30 minutes | | | When button is pressed navigate to the correct page<br>JS - transferring data persistently after navigation (ties in with id 20) | 17 |

| | | | | | |
|---|---|---|---|---|---|
| Get localStorage working for static data (topScorers)<br>Estimation: 5<br>Priority: 1 | 50 minutes | | | Use JS to initially retrieve static data for the user and put into localStorage. | 18 |
| Deployment of Website<br>Estimation: 5<br>Priority: 3 | 30 minutes | | Learn how to host a website | | 19 |
| Learn how to pass variables between pages<br>Estimation: 3<br>Priority: 1 | 30 minutes | | | Create onclick event with navbar items and use localStorage to keep track of the currently selected league. | 20 |
| As a user I want to be able to select a team and see all their player stats<br>priority : 2 | 1 hour | | | HTML<br>JS<br>CSS | 21 |
| Fix usability of charts in ‘statistics’ widget<br>Priority: 1<br>NEW | 30 mins | | HTML/CSS | | 22 |

| | | | | | |
|---|---|---|---|---|---|
| Design Improvements (application wide)<br>Priority: 1<br>NEW | 1 hour | | HTML/CSS | | 23 |
| Landing for specific league design improvement (options.html)<br>priority: 1<br>NEW | 1 hour | | HTML/CSS | | 24 |
| (REFACTORING) Create class for storing player data from different sports. This would greatly improve ease of access to data within player objects and would reduce code reuse.<br>Priority: 5<br>NEW | 1-2 hours | JS | | | 25 |
| Scale Statistics Widget to work with Basketball | | | JS - adapt existing code | | |

| | | | | | |
|---|---|---|---|---|---|
| Priority: 3<br>NEW | | | | | |
| Scale League Table Widget to work with Basketball<br>Priority: 3<br>NEW | | | JS - adapt existing code | | |
| Scale Live Match Widget to work with Basketball<br>Priority: 3<br>NEW | | | JS - adapt existing code | | |

# Sprint 2 (8/9/21 - 22/9/21)

## Sprint goal:

**Scrum Master:** Max

Get the core components for soccer sport working for each widget creating the backend and integrating with the frontend, as outlined below:

Navigation:

- User should be able to select from the leagues available for soccer, this should then navigate them to a landing page that lets the user select a widget page (from league table, stats widget or live fixtures)
- The scripts controlling the widget pages should update the data based on what league is selected

Stats widget:

- User should be able to select a team and then the graph should update showing all the players stats for the selected team
- User should be able to select a season, which, will update the teams that can be selected (teams can get relegated), and update the stats that are displayed
- User should be able to select a stat type e.g. goals, assists, etc. The graph should then show the players stats (of the selected stat type)
- User should be able to see top scorers from the league for the current season (future sprint will implement all seasons)
- Experimental work done with chart representations (not on master branch)

Live fixtures widget

- Users should be able to see the upcoming matches of the selected league.
- Users should be able to see the table that displays the livescore of all teams in the selected league.
- User should be able to see the location and time of an upcoming match.

League table widget

- Users should be able to view the match history and scores of the teams directly when opening the page for the current season
- Users should then be able to see the entire match history for a selected season of the teams after clicking on them.
- Users can filter the season they would like to see and for which sport they would like to see it for.

## Task Allocations / Sprint Backlog:

1. **Antony**: task 4, 5, 7 - can be found in the product backlog

   Acceptance criteria/DoD - task 4: the user should be able to select a stat type e.g. goals or assists, after selecting a stat the graph should update displaying the correct data for the currently selected options

   Acceptance criteria/DoD - task 5: the user should be able to select a season from a drop down, when the season is selected the teams should be fetched from the API again, as different seasons will have different teams (relegation/promotion). After the teams have been fetched the graph should update as different seasons will have different statistics

   Acceptance criteria/DoD - task 7: the user should be able to select a player and the graph should update with the selected players data.

   Also worked on task 12, however, is not completed.

2. **Eric**: task 13, 14 - Setup JS for the league table (can be found in product backlog)

   Acceptance criteria/DoD - task 13: The user should be able to see some of the teams match history in a hidden table when they click on a team within the standings table.

   Acceptance criteria/DoD - task 14: The user should be able to see the league standings working for each league and be able to change seasons accordingly and have them shown in a table with the statistics filled for each respective league and season.

3. **Max**: task 8, 12 - can be found in product backlog

Acceptance Criteria/DoD - Task 8: Top Table should load correct data for the selected (soccer) league. User should be able to switch between different statistic types, with correct data loading for each one.

Acceptance Criteria/DoD - Task 12: Change chart representation to a pie graph, loading in correct information and easy readability.

4. **Mursal**: task 20 - set league ids, help with LiveMatch widget (1, 2, 3) - reference numbers can be found in product backlog

   Acceptance Criteria/Dod: Task 20 (setting league ids) - as the user selects a different league in the soccer sport, there should be a different league set in local storage, to be used in api calls for different widgets.

   LiveMatch widget: Acceptance Criteria/DoD specified by Suryadeep below.

5. **Suryadeep**: task 1,2,3 - can be found in product backlog

   Acceptance criteria/DoD - Task 1: The Live Stats should load the current event data for the selected match in key events. Users should be able to switch between different teams.

   Acceptance criteria/DoD - Task 2: The key events should load and display the current events happening in real time and when clicked on, should display statistics, lineups and players.

   Acceptance criteria/DoD - task 3- The Location and time section should take in the location and time of each match from key events and display it in the location and time section.

**Note:**

Pair Programming: If the teammate(Suryadeep) who is allocated to the task of developing live fixtures needs support with the tasks, other teammates (Max and Mursal) plan to complete it by using the means of Pair programming.

# Sprint 2 Retrospective - 22/09/21

**What went well?**

1. Target velocity was relatively well met, with few minor struggles.

2. Risk monitoring strategies were followed, minimising the chances of accounted risks occurring. Frequent communication between group members and weekly meetings alleviated most of the potential issues. Group chat communications established in the project management plan played a large role in this as well, meaning team members could receive quick feedback and help if problems arose.

**What could have gone better?**

1. The team had some pair programming events earlier but the frequency reduced drastically, more pair programming events should have been conducted.

2. Task 7 was changed as the API only returned total stats rather than a stat count for each game, this meant that graphing individual players had no visual purpose, so instead the team decided it would be better to graph all the players for a selected team.

**What will we try next?**

1. The team will try having more pair programming sessions so that problem in hand can be understood better.
2. Work on different branches in git, and then send a merge request (when merging) so that relevant members can check that merged code meets functional and nonfunctional requirements.
3. If the team has time to implement tests, set up CI on gitlab so that every push we can check that the application still passes all tests. This will lay the foundations for future potential development / expansion on the project.

**What questions do we have?**

1. What is the scope currently? Will we have time to implement more than one sport? Is it better to perfect one sport in terms of appearance and functionality or to add more sports (quality vs quantity)?