# 5-a-side Techincal Task - Thoughts / Planning / Reasonings

- Using react to build the 5-a-side app:

  I felt that I wanted to use react over other frameworks for a few reasons. Firstly i felt it provided as nice template for building apps. I also felt it was the most challenging framework that we had used during the course and thus wanted to challenge myself and use this as an opportunity to increase my understanding of something which can be relatively complex.

- Using redux to manage the state:

  I figured that while storing 10 names in state is quite simple, if i wanted to expand the app to include things such as player attributes it would be useful to employ a state management system such as redux so as to make my code simpler, and easier to follow.

- Using React over React Native:

  I chose to use react over react native as I thought that while a simple team sorting app would probably be quite easy to display on a mobile screen and useful for a user to have on the go, if increased team sizes or player attributes were added, the number of inputs and varying pieces of information that would have to be displayed might make creating a mobile version of the app quite complicated and require a number of screens (possibly overlapping) to add to the complications.

- User Journey:

  - *Possibly:* Be able to select the team sizes before generating the list
  - *Possibly:* Select players skill level
  - **Add Players to a list**
  - **Have the current list displayed on the page with option to delete players**
  - **Upon completion of the list generate the teams, displaying each team as lists**
  - *Possibly:* Have the option to edit teams / move certain players around
  - *Possibly:* Be able to reshuffle the list or completely clear it

- Possible improvements that could be made (should time have allowed it) and reasonings as to why they may not have been included:

- An obvious improvement would be to implement the advanced functionality of balancing teams based on skill level. I felt that at this time the complexity involved (or my lack of understanding) would have meant that it would have been too time consuming of a task to undertake, and would have meant that more important base functionality of the app may have suffered as a result.

- The use of a drop down menu to set maximum number of players works well if you are just considering football, but restricts the choice a user has - in this case between 8 and 22 players - possibly limiting the potential for the app to be used to split teams for any kind of activity.

- The implementation of editing directly within the app without the need for re-entering values in the inputs could have been a useful feature, but again could have taken me a considerable amount of time to implement.

- There is currently an issue in that refreshing the app completely loses all data as the state is set back to the default values. A possible solution for this could have been to store the data in a local API, and fetch it on refresh so as to maintain the generated teams.

- **<u>Known Issues</u>**
    - There are conditions in place which enables the generate team button only when the maximum players equals the number of players submitted. The props keep track of these values and even update correctly, however if the user has manually changed the maximum players the condition is never met, despite both numbers being equal to each other. As a result I have had to comment out this condition and have the generate team button permanently enabled which can cause errors if pressed before both teams have been assigned at least one player (which can happen as players are assigned randomly until one team is full).
    - There is a similar issue with the player submit button, in that when the button is pressed after players are already at max level the app will crash. The condition I had in place to disable the button also does not work in a similar vein to the above issue.