CLARITY OF CERTAIN BUSINESS REQUIREMENTS

While the majority of the business requirements were fairly clear (albeit left fairly open to interpretation), there was one which gave me pause as to how to approach its development - list display options by song track or album.

1. As this was listed beneath the search feature, was this a requirement that must be applied to search results?
2. Was this a requirement that must be applied to songs in playlists only?
3. Was this a requirement that must be applied to every instance of a song list, whether it be in a search result, playlist or generic all songs list?

I was confused by where and how this feature needed to be implemented, and in a workplace scenario I would a) expect more thorough requirements including UI templates provided by a UI/UX designer and b) have the ability to go back to the provider to clarify these requirements. As I felt unclear and not confident in what was expected, and due to time restrictions, I left this requirement in the backlog.

A DEVELOPER IS SIMPLY THAT

In a realistic Agile/SAFE business scenario, 40 hours would be an extremely unrealistic expectation of time for a developer to create something from scratch without support. Within Together, a developer would not be expected to create the UI and UX of an application (Digital team, I used Figma), make decisions on features dependent on the business requirements (Business Analyst, I had to make my own judgements), create and manage API's and databases (Architecture, I created diagrams to support my decisions) nor manage PBI's and priority (Product Owner/Scrum Master, I used Trello to manage tasks and priority). In some businesses, a developer would also not run tests on their work, this would be passed to a team of test engineers.

I found it extremely difficult to manage everything in this project while meeting the requirements of the brief.

APIS AND OPEN SOURCE COMPONENTS

I struggled to get api calls to work with the UI for this project, possibly due to time pressure and constraints behind my knowledge of MERN stack and routing. Potentially this could have been avoided if a different method of implementation was used, such as the technology that we use at Together for creating React-driven projects (known as Outsystems). Had I had this knowledge, I could have created playlists with songs in them and not just empty arrays. However, I feel I have successfully implemented CRUD methods into this project, just perhaps not to my own high standards.

A problem that I had that hindered progress for a while was a bug caused by the npm-react-player component that I was using to control music playback. It took a while to realise that the player was incompatible with the most recent version of React and the 'strict mode' tags wrapping the app component. Situations such as these could easily be mitigated if the developer of the npm module had kept their documentation up to date and maintained it efficiently.

IMPROVEMENTS & FEATURES TO IMPLEMENT FOR V2

Edit and delete playlists - while I managed to implement editing the name of a playlist and deleting the playlist within the music player, I would have liked the ability to add songs to playlists and remove them.

Allow the user to skip or rewind an audio file - I would have liked to have added a skip and rewind button to the main music player. The npm-react-player component does not include this natively, so it would have required additional time to implement this.

Carousel home menu to add continuous improvements - Instead of having static buttons on the home menu, I would have loved to include a carousel feature so that accessing menus was dynamic (swiping). Having a carousel would also allow for further icons to be added in, especially if during further development additional features were added (video player, dedicated settings menu, user guide, etc).

Song focus tray to control current music when in different menus - This is a common feature in existing music players, where the user can back out of the main player and a mini player will bump up from the bottom of the screen. This allows for both music control and navigation of other menus simultaneously. I believe the npm-react-player has some functionality for this, but again this would require additional time to implement.

Built-in help menu with basic user guide - Eluded to previously when mentioning carousel, I would have liked to include a PDF viewer with the user guide pre-installed so that it can be viewed on the device. This saves users from struggling if they have misplaced a paper copy that comes in the box.

Swipe to edit and delete - To tidy up the UI a bit, I would love to add a swipe menu to each component so that the buttons aligned to the right (for example, the edit button on the playlists) were hidden behind swipe actions. This is a common feature associated with touch devices and I feel it would be a fairly natural addition.

Show the user remaining device battery life and current time - Again, a customary feature in most, if not all, portable battery charged devices. This would indicate to the user when it is time to recharge the device and also provide a convenient feature for time, which could be adjusted in a settings menu.

‘**Favourite’ songs into a global, default playlist -** Another common feature amongst music players, notably within Spotify, is the ability to ‘heart’, or ‘favourite’ a song to add it to a ‘global’ playlist. This playlist could also be used to recognise user behaviour and recommend artists or songs if a larger API were to be used.