React Try-out

Objective

The goal is to replicate the screen provided while meeting all the evaluation criteria. Correct use of React, use of best practices, reusability and clean, legible code are points to keep in mind.

Evaluation Criteria

The following points are all to be considered. Keep in mind that skipping these will result in penalties:

- Logically diving the screen in as many reusable components as possible (where it makes sense)
- Correct use of stateful and stateless components
- Correct interaction between parent/child components
- Delegating style related issues to CSS/HTML as much as possible, where applicable
- Correct use of controlled components
- Correct use of event handlers
- Correct implementation of lists rendering
- Correct use of JSX elements
- Use of SASS
- Clean, understandable code (both CSS and JS)
- Use of best practices
- Adding default values to props and telling React which data types they should be.

Brownie Points

Although not mandatory, using the following technologies is desirable and will greatly increase the chances of being selected in the project:

- Styling using styled-components
- Use of Redux instead of React's state, where applicable
- Allowing the screen to navigate with the keyboard, like if it were a form

Screen Design and Overview

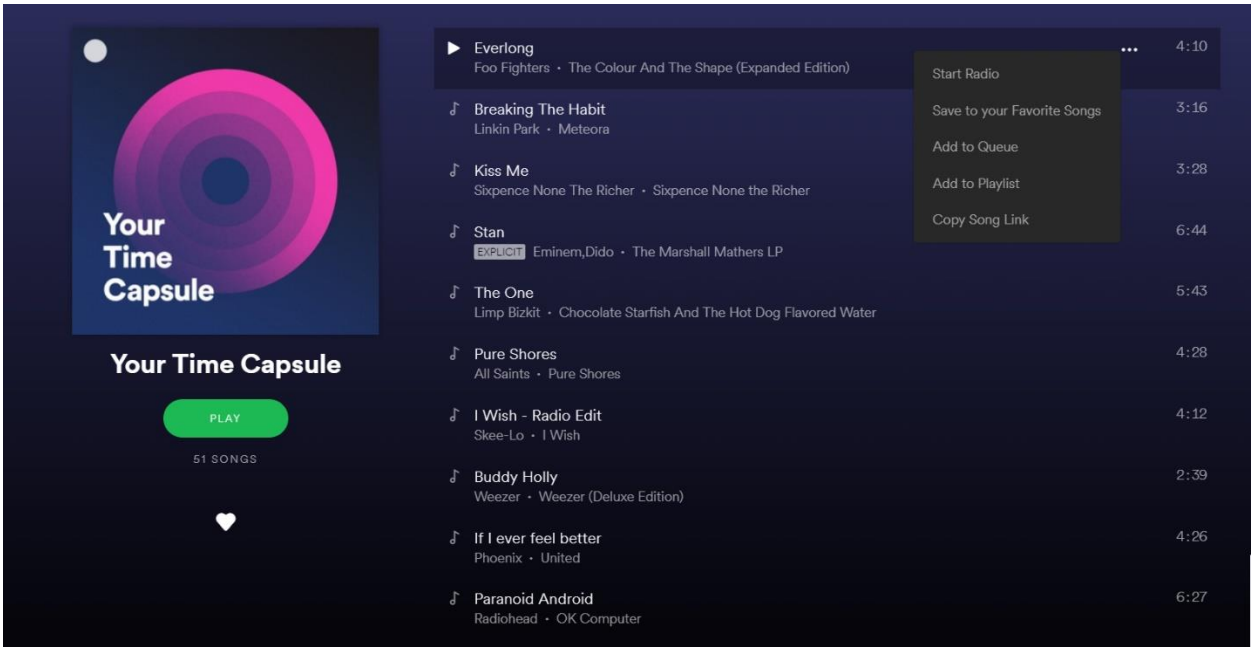

Figure 1: Default playlist state

As seen in figure 1, the screen to replicate is a playlist, as seen in a widely known music streaming service. At the left side of the screen you can find the cover image for the playlist, the playlist's name, a "PLAY" button which will start playing the currently selected song (or the first song of the playlist if none has been selected), the number of songs currently in the playlist and a heart icon, which indicates if the current playlist is amongst the favorite ones of the user.

At the right side of the screen we can find the playlist itself. Each song consists of a title, the artist or musician(s) behind the song, the album it belongs to and an "EXPLICIT" tag if the song makes use of explicit language. You'll also find the length of the song and an ellipsis icon that will open a sub menu when clicked. Hovering each option from this sub menu will change its background color to a darker one. The only way for that sub menu to disappear is by clicking in one of the options, or by clicking somewhere else outside that menu. Needless to say, only one of such menus can be displayed at any given time.

Hovering over a song highlights it, as seen in figure 1, and changes the musical note icon to a play icon. Clicking on a song leaves it highlighted, until another one is chosen. Keep in mind that this does not prevent other songs from getting highlighted if the mouse hovers them. Clicking on the play icon of a particular song will play that song.

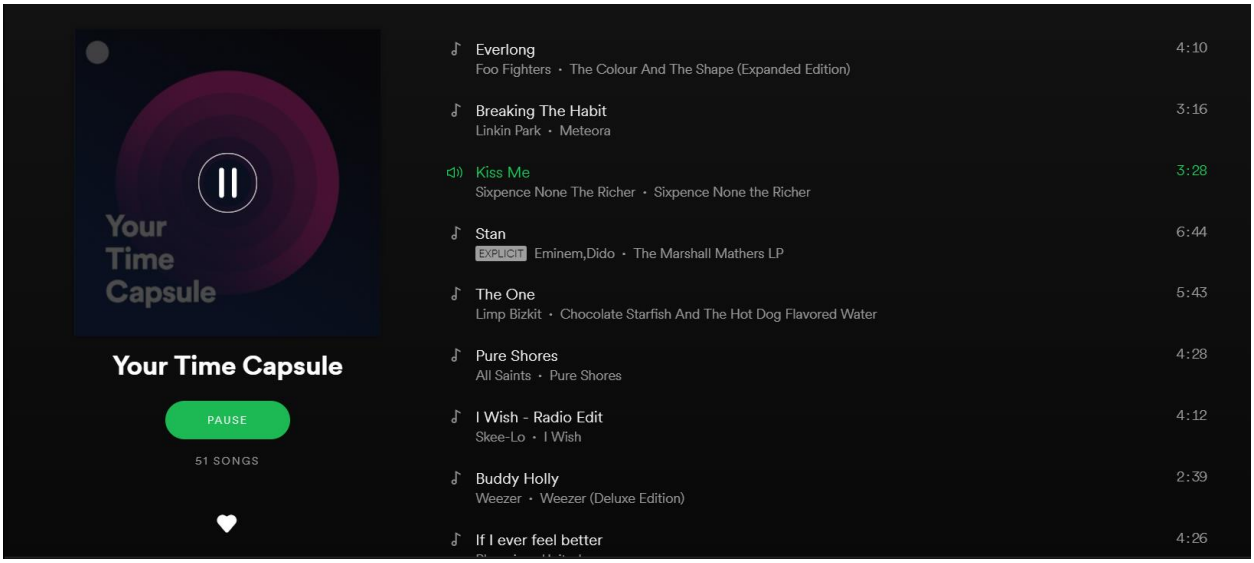

Figure 2: Playlist state when a song is playing

If a song gets played, the play icon on the song changes to a speaker icon. Hovering over the speaker icon will change it to a pause icon (see figure 3 below) and clicking on the pause icon will pause the currently played song. When a song is played, the song's title and duration will change its text color to green.

Figure 3: Song playing

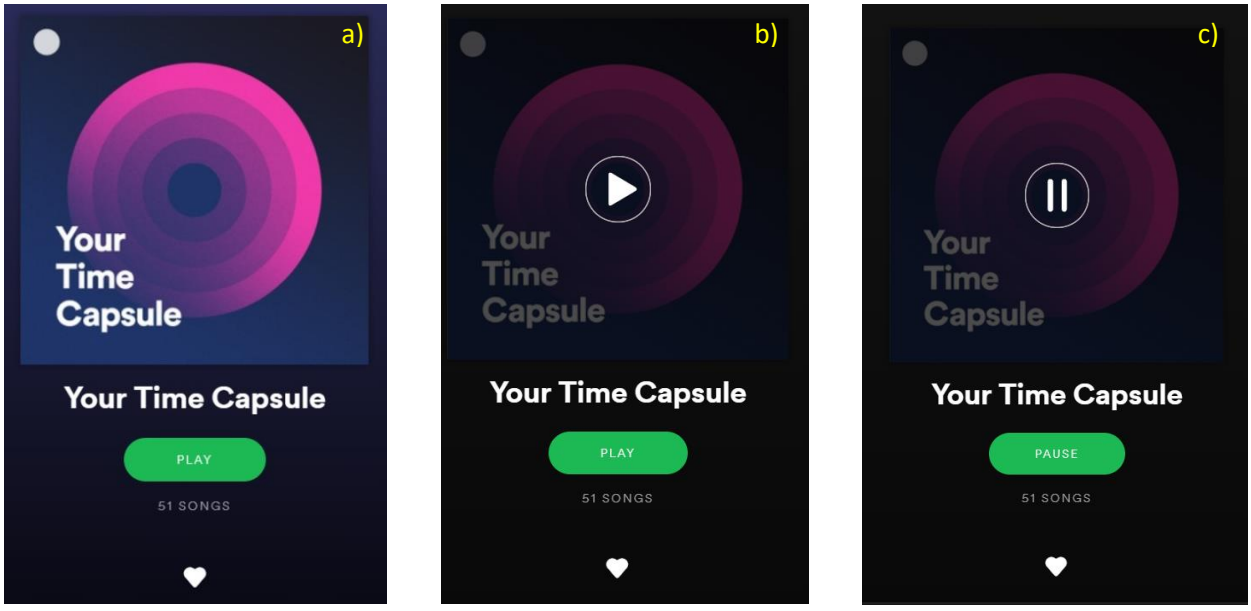

Figure 4: Playlist cover and its different states

If no song is playing and the mouse is not hovering the playlist's cover, the cover should stay in state a), figure 4. Hovering over the playlist's cover when a song is not playing will change it to state b), figure 4 . Notice how the cover darkens and a play button inside a circle appears over it. In state b), the user can click on the play button in the playlist's cover to start playing the first song of the playlist if none has been played before, or it will resume playing the latest song paused.

If a song is playing, the playlist's cover will change to state c), as seen in figure 4. As long as a song is playing, the cover will stay in state c) unless the currently played song is paused. A song can be paused if the user clicks on the pause icon of the song currently playing or by clicking on the pause icon inside the circle in the cover's playlist, as seen in state c).

In a similar fashion, the green rounded button at the left of the screen will display the "PLAY" text if no song is playing, and when clicked, it'll resume the previously paused song (or will begin playing the first song in the playlist if no song has been played before). Similarly, if a song is playing, the green rounded button at the left of the screen will display the "PAUSE" text and clicking on it will pause the currently played song.

Notice that if a song gets paused, the song's title and duration will stay green and the pause icon will change to a musical note, as seen in figure 5. Hovering the currently paused song will change the musical note icon into a play button, however, remember that only by clicking the play icon the user can reproduce a song.

Figure 5: Song paused

Regarding the heart icon, by clicking on it the playlist will be saved as a favorite. In that case it will change from an empty state to a filled state. If the playlist has already been favorited, clicking on the heart icon will remove the playlist from the favorites, and the heart will revert to its empty state.

Finally, when hovering over the green "PLAY"/"PAUSE" icon, the heart icon or the ellipsis icons, they'll slightly increase their size to send the user a visual cue that they're currently being pointed at. See figure 6 for an example for the green "PLAY"/"PAUSE" button, as it also changes slightly the brightness of the button's color.

Figure 6: Green button slightly increasing in size and getting brighter when being hovered.

Considerations

- Don't worry too much on making an exact replica of the screen. What's mainly being evaluated here is your logic and use of Javascript, so try focusing on that if you're having a bad time with CSS.
- Feel free to change the cover image of the playlist. An image has already been supplied to you, but it's up to you if you want to use it.
- Don't change the font-family supplied to you. In it you'll find some of the icons that are used in the screen. Below you'll find a table with the icons that you should be using. Be sure to use the "content" property in your stylesheet to insert their corresponding UTF codes:

Icon	UTF Code	Font Size
Heart (filled)	\f3dc	24px
Heart (empty)	\f3db	24px
Ellipsis	\f12c	16px
Musical Note	\f156	16px

- The following icons are supplied as svgs. Here are the details for them:

Icon	File	Width	Height
Play	play.svg	14px	14px
Speaker	speaker.svg	16px	16px
Pause	pause.svg	14px	14px

- Structure the data to display as you please. Be it either React's state or Redux's store, model the data currently being used in the screen in a way that matches the components you'll construct. Be sure to include as many songs as you need necessary.
- Whenever there's a change in state, be sure to console log that change so it can be easily tracked by the evaluator.
- The green color used in the "PLAY"/"PAUSE" button and as highlighter is `#1db954`.
- Not everything is defined in this document. Feel free to interpret it and fill in the gaps for such cases.

Environment Setup

A base project should have been already supplied to you via email. The intention is for you to take such project as a base and start constructing the screen on top of it. NPM packages such as styled-components, prop-types and classnames have already been added to package.json, so you should have everything ready to start developing your screen after installing them.

On the root folder of the project simply type, in a new command window:

```
npm install
```

Also, feel free to use your favorite IDE and structure your folders as you think is the best for your project. However, keep in mind that:

- All files and folders you add should be inside the src folder. Otherwise, the app will start to complain.
- As an exception, image files should go inside the public folder, otherwise you'll get broken references to them.

To run the app and see the changes as you make them, simply type:

```
npm start
```

Make sure to have the latest version of Node installed in your computer. When finished, please send in an email your project with everything **EXCEPT the node_modules folder** to your evaluator.