## #Assignment\_08: Assignment: Songsaver

Today you are going to work on creating a Single Page Application (SPA) in which you can save all your favorite songs with a rating.

### Requirements

In order to meet the requirements of the (imaginary) employer, we expect your SPA to meet the following **core functionalities**:

* As a user, I want to enter the following information about my song: title, artist, genre, rating (dropdown menu is not required).
* As a user, I want to be able to click on one button so that my entered song is added to my playlist.
* As a user, I want to be able to see my songs in an overview (my playlist), in which all entered data are visible.

### Other conditions

* You are completely free to build your components yourself: Stateful and/or Smart versus Dumb, or with Class components, or Functional Components (with the UseState Hook)....
  + *.....But, beware that you don't get spaghetti code, in which each component does everything (managing State, managing functions, and UI display, etc.)*

### Passed = Core Functionalities + 3 points (!)

In addition to these Core Functionalities above, you are required to choose 3 additional points from the list below.

**Additional points:**

* **1 POINT** – **Sort** the songs by name (a-z or z-a) or artist (a-z or z-a), sort by stars (5-1 or 1-5)
* **1 POINT** – **Delete button**: delete songs from the state
* **1 POINT** – **Categorize:** each genre gets its own list and the song you add ends up with the respective genre
* **2 POINTS** – **Filter** songs by:
  + *genre (select which genres you want) in the UI you see either a dropdown with which you can select 1 genre, or you create a checkbox per genre, if that checkbox is checked: show the genre (so this can also be several or none be one)*
  + *stars (sort which results with x stars you want to see) in the UI you see either a dropdown with which you can select 1 rating, or you create a checkbox per rating, if that checkbox is checked: show all songs of that rating (this so there can be several or none)*
* **2 POINTS**

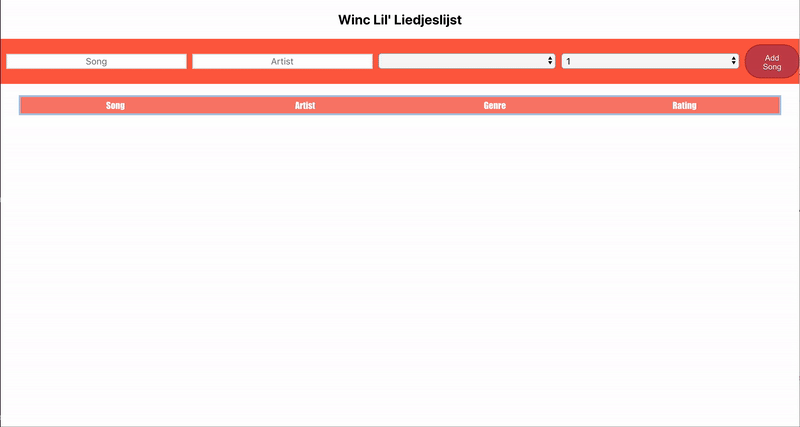
– [Routing](https://reactrouter.com/docs/en/v6/getting-started/tutorial) (/Navigation)**:** add a navbar or menu with a link and page "About us ", containing a short story about yourself and this project (max 10 sentences)

* **2 POINTS** – **REDUX**

### design

This week the focus is 100% on React. Within React, all the "logic" and functionality comes from using JavaScript with real-world examples. CSS should be the **last** thing you work with.

But what should it look like?... (so this is just a guideline - you can certainly deviate from this)



### GitHub

* We would very much like to see you get the Core Functionalities working on the final Master Branch.
* From this base, you can build any additional point you choose from the list above from (and on) a separate feature industry (eg sorting).
  + *This has the advantage that, if you get stuck on a certain feature, you can stop this work for a while to continue with another feature.*
  + *Later you can always come back to a feature by switching to that branch*

But..... we don't send you out without navigation. As a "compass" we give you a piece of basic code that you have to point in the right direction to complete this assignment.

(If you decide to use Redux, you will need to modify the base code below slightly)

**class** SongOverview **extends** Component {

constructor() {

**super**()

**this**.state =

{

songs: []

}

}

addSong = (song) **=>** {

*// do something to change the state*

}

render() {

**return** (

**<div>**

<SongForm addSong={**this**.addSong}/>

**<table** style={{width: "100%"}}**>**

**<tr** className="song-header"**>**

**<th** className="song-row\_\_item"**>**Song**</th>**

**<th** className="song-row\_\_item"**>**Artist**</th>**

**<th** className="song-row\_\_item"**>**Genre**</th>**

**<th** className="song-row\_\_item"**>**Rating**</th>**

**</tr>**

**</table>**

<SongList songs={**this**.state.songs}/>

**</div>**

);

}

}

export default SongOverview;