* ~~[4] add hero credit~~
* [5] You must include some type of CSS transition and animation effect on this page. The more interesting it is, the more marks to be gained. I would also like you to use react-transition-group for this effect. See https://reactjs.org/docs/animation.html for more information. (do Later)
* ~~[6] header should have a logo, logo should return to the Home View~~
* [7] The API will take some time to retrieve this data. Display a loading animation (there are many free animated GIF available) until the data is retrieved. Simply display the image just before the fetch and then hide it after fetch is successful.
* ~~[7] Play Details view however, only a few plays have text available. Indicate this status visually, perhaps by styling the title differently, or add an icon (grey out read button and cursor not-allowed)~~
* ~~[7] store the content retrieved from the play list API in local storage after you fetch it. Your page should thus check if play data from this API is already saved in local storage: if it is then use it, otherwise fetch it and store it in local storage. This approach improves initial performance by eliminating an early fetch in future uses of the application.~~
* ~~[8] toggle the visibility of the favorites panel~~ with transition
* ~~[8] clicking on favorite item opens details~~
* [8] favorite item delete animation/transition on the play title to provide visual feedback that the play is being deleted. (do Later)
* ~~[9] in about Include group members, github link, technology used, any thirdparty source code, etc.~~
* ~~[10] play browser filters genre no duplicates~~
* ~~[10] If no matching plays are found, notify the user within the Play List/Matches area.~~
* ~~[11] Play Details Missing everything~~
* ~~[12] Text Tab Missing everything~~

Assignment 2:

APIs:

* Create an API using Node and MongoDB
  + Need to host two (maybe three?) collections
    - All get requests require an authenticated user
    - Routes needed:
      * /api/list
      * /api/play/id
      * /api/user/id
      * /api/login
* Possibly need cross-origin resourcing (only if API and front end are on different domains

Website/React:

* Improve visual design using UI library
  + Ant Design
  + React Bootstrap
  + Material-UI
  + Argon
* About:
  + Include first three API urls
* Login System (use Node + EJS):
  + First thing to appear when accessing website if not already logged in
  + Form:
    - Email address
    - Password
    - Create sample credentials
  + If email and pass work, immediately redirect to website homepage
  + System for “remembering” whether user is logged in (Passport package in Node).
* Logout button
* Profile that can be accessed via profile button
  + First name
  + Last name
  + City
  + Country
  + Picture
  + Date joined