CPSC 1520 Assignment 3: Album Search and Save Tool

Introduction

You've increased the search functionality on your album creator application by having a backend implementation of the data; you've also increased your data

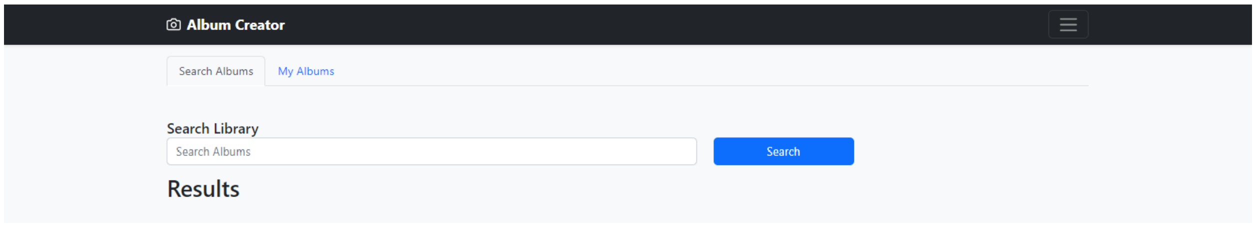
This assignment will test your knowledge of the last few topics of this course by implementing this album search and save tool. Please refer back to all of the examples and all of the slides to complete this assignment, but it mainly focuses on the topics of "Arrays and loops," "Fetch Fundamentals," "DOM APIs and Timers," and "NPM, Tools and ES Modules."

Note you'll also be running a separate backend to your project that will be included, so you'll need to run the backend and the frontend (your assignment) in different terminals and leave them running.

Overview of functionality

Part 1: Tab Navigation

Here is a sample of part 1. When you click a single tab, you should be able to see the appropriate page. If you click "Searched Albums," you should visit this page.



Album Creator

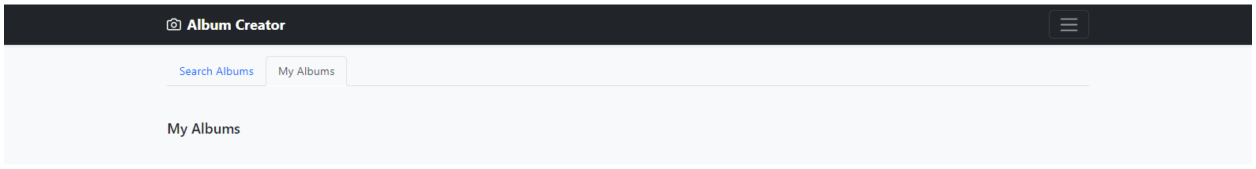
Search Albums | My Albums

Search Library

Search Albums | Search

Results

Here is another sample of part 1: If you click "My Albums," you should see the following tab.



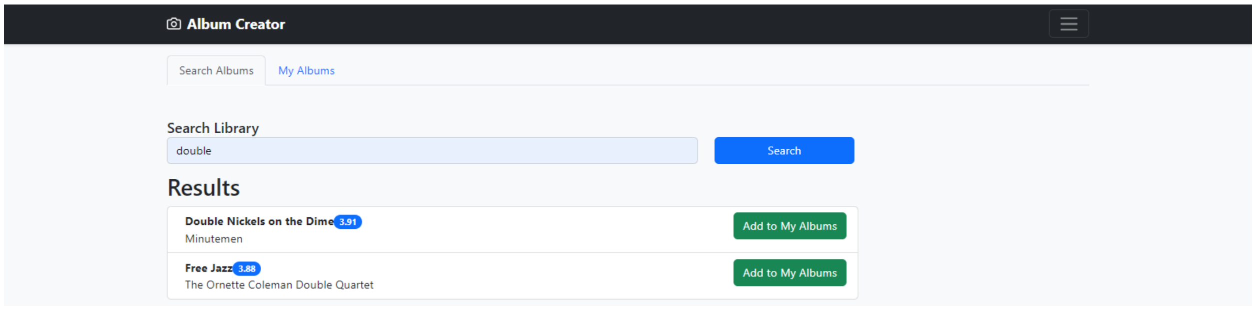
Album Creator

Search Albums | My Albums

My Albums

Part 2: Search by an Album with REST API backend

Example for part 2: When you have implemented the search, your application should look like this if you search for "double."



Album Creator

Search Albums | My Albums

Search Library

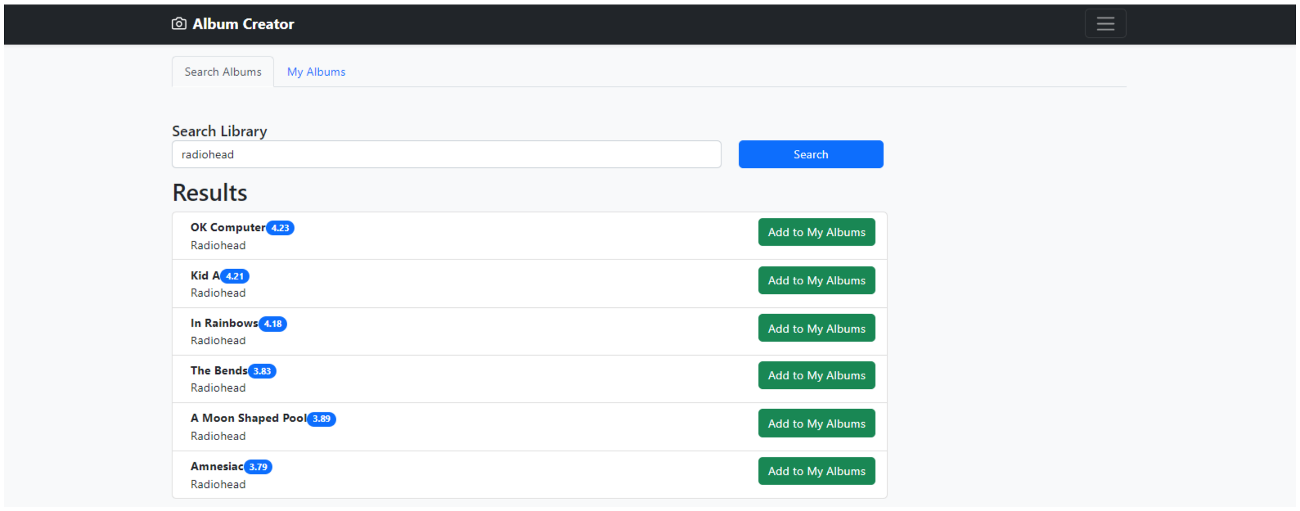
double | Search

Results

Double Nickels on the Dime 3.91
Minutemen
Add to My Albums

Free Jazz 3.88
The Ornette Coleman Double Quartet
Add to My Albums

Another example for part 2: When you have implemented the search, your application should look like this if you search for "Radiohead." Note: the content should reset when you execute another search.

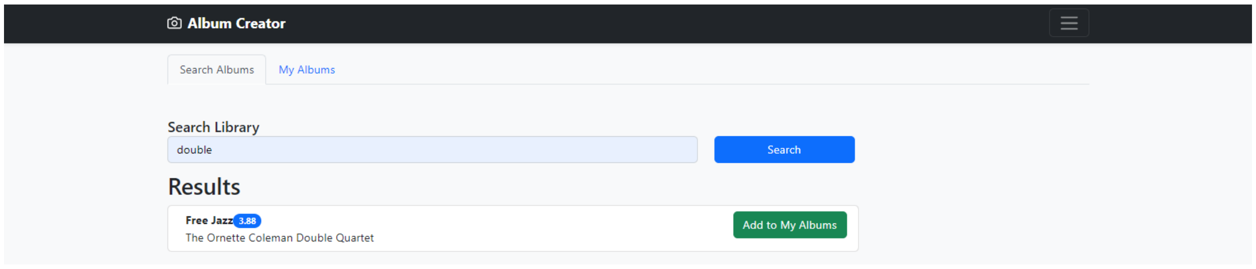


Results

OK Computer 4.23 Radiohead	Add to My Albums
Kid A 4.21 Radiohead	Add to My Albums
In Rainbows 4.18 Radiohead	Add to My Albums
The Bends 3.83 Radiohead	Add to My Albums
A Moon Shaped Pool 3.89 Radiohead	Add to My Albums
Amnesiac 3.79 Radiohead	Add to My Albums

Part 3: Add to My Albums

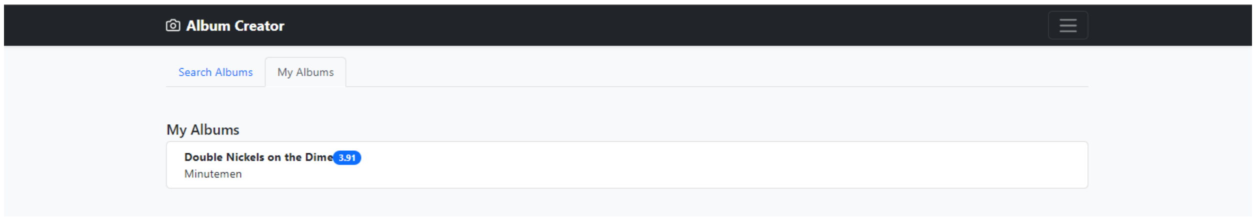
Example for part 3: When you have searched for "double" and clicked "Add to My Albums," the album is removed from the searched list and added to the page "My Albums," and the button is removed. Here is the search page after the button has been clicked.



Results

Free Jazz 3.88 The Ornette Coleman Double Quartet	Add to My Albums

If you then click the "My Albums" tab, the list item is added to the My albums section, and the button "Add to My Albums" is removed.



My Albums

Double Nickels on the Dime 3.91
Minutemen

NOTE: Duplicates are allowed in "My Albums".

Bonus Part: Saving Albums using the REST API (Do not ask your instructor this is a challenge)

Every time you save to my albums (please refer to the README.md I the “assignment-3-backend” project), you’ll make a POST request to the backend saving the album data selected. You’ll need to create an endpoint for this in your “api.js” file.

If you reload the page, your application should already have loaded your saved albums, and you should see the saved albums in the “albums_app_db.json” in your “assignment-3-backend”. It should look like “Part 3: Add to My Albums,” but your albums are still present when you reload the page.

Required Tasks

Part 0: Setup the Project

- Install and run “assignment-3-backend” in a separate terminal and folder than your “assignment-3-starter”.
    - Extract the project and Install all of the dependencies.
    - Look at the “scripts” in the “package.json” file and figure out how to run the backend server, leave this command running in the background.
    - Look at the README.md in the “assignment-3-backend” and test the “GET Album Search Endpoint” and make a request in a REST API client of your choice.
- In a separate terminal, install the packages bootstrap and parcel, and setup the required scripts using npm.
    - Your project should be using parcel and not live server when running the project (it won’t work anyways if you try it that way).
        - Your scripts need to be correct and are the same as other projects you have done in class with your instructor. (This will project will be using parcel refer to the documentation if you are having trouble running the project
    - Once “bootstrap” is installed, import the minified “bootstrap” libraries, and import your custom CSS at the top of your “main.js” file like below.

```
/*
Import bootstrap then your css below
*/
import 'bootstrap/dist/css/bootstrap.min.css'
import '../css/cover.css'

// Javascript imports below
```

    - Run your project.
    - NOTE: In your solution there should be no node_modules folder or “parcel-cache” folder. Marks will be taken off if they are included.

Part 1: Tab Navigation

- Get the necessary elements.
    - Select the elements with the Ids “album-tab-navigation,” “search-album-tab” and “my-albums-tab” and give them appropriate variable names.
    - From the “album-tab-navigation” element, get the nested children that are links and make separate variables for them.

- Display the correct Tab.
    - Add an event listener on the “album-tab-navigation” element that will listen to click events.
    - In the event handler, get the “textContent” from the “event.target” and save that to a “tabName” variable (also display what it is for yourself so you can see what it is).
    - If the “tabName” is strictly equal to “Search Albums” then do the following:
        - Add the class “active” to the search tab and remove it from the myAlbums tab
        - Remove the class “d-none” (which is the CSS property “display: none” in “bootstrap”) from the “search-album-tab” element and add it to the “my-albums-tab” element..
    - If the “tabName” is strictly equal to “My Albums” then do the following:
        - Add the class “active” to the myAlbums tab and remove it from the search tab
        - Remove the class “d-none” from the “my-albums-tab” element and add it to the “search-album-tab” element.

Part 2: Search by an Album with REST API backend

- Create a function (that takes a single value as a parameter) in the “api/album.js” file that will use fetch to call “assignment-3-backend” and return the data from that endpoint.
    - The function should be exported (using the named export syntax) from that file and imported into your “main.js” file.
    - You can use the promise syntax or the async await syntax to return the data from this function.
- NOTE: you’ll need this for the following parts, but it’s strongly recommended that you test this function.
- Create a function that will create a list element using your knowledge of the DOM API in the file “dom/albumElements.js.”
    - This function will take in one or more arguments which will be the information you need to display in the list items (depending on the implementation). Please look at the part 2 overview of functionality to see what the list item should look like.
    - This function should be exported (using the named export syntax) and import it in your main.js file.
- In the index.js file, create an event listener to listen to the form submission event (use the form with the id search-album-form) and make the searched list items based on the searched value.The form should be prevented from submitting.
    - It should remove all previous items from the searched list every time you submit.
    - You should loop through the results and use your DOM API functions to add to the list element.
        - NOTE: You should use your function from “api/album.js” to get the data and use your function from “dom/albumElements.js” to create the searched list item elements
- Once you see results populate the page, then you should move on to the final step.

Part 3: Add to My Albums

- In your index.js file, create an event listener that will listen to the click events on the list element with the id “searched-albums-list.”
    - Check if the “event.target” includes the class “my-album-button” to ensure it’s the button in the list item that is clicked.
    - If the button is clicked, append this element to the list element with the id of “my-albums” and remove the button element. If you are wondering what this looks like, please look at the part 3 section in “Overview of functionality.”
- NOTE: The instructor will lay out the formatting preferred.

Bonus Part: Saving Albums using the REST API (Do not ask your instructor this is a challenge)

- When you perform a search save, the results in an array of objects in the variable “searchedAlbums” given.
- Modify your part 3 to get the search album index clicked and save the entire object to the “myAlbums” section of the backend rest API.
    - NOTE: Look at the “POST myAlbums Endpoint” in the README to the backend.
- The second part of the bonus is that when you refresh the page in the “My Albums” section, it should load the results and populate the list with what you’ve already clicked.
- NOTE: marks on this will only be given if the rest of the application is functional (subjective and based on the instructor).

Marking key

Tasks	Grade	Marks	Total
NPM and Packages. • Packages are installed correctly and are in the correct dependency section (dev dependency vs dependency) • Scripts are setup correctly and the source attribute in the package.json is setup correctly. • node_modules folder and parcel cache are included (if you included them it’s negative marks on your assignment)		**3** **3** **-5**	
Fetch • The fetch request is takes in the search query and returns the data from the fetch request as a promise OR uses the correct async await syntax. • The function is in the correct file and exported properly. • The function is imported correctly in the index.js file		**5** **1** **1**	
Dom API and Manipulation. • Tab Navigation works properly, adds and removes on the elements are properly written so that the proper sections are displayed. • Create list item element item function is in the correct file and exported correctly. • The function is imported correctly in the index.js file • The create list item element function uses ONLY DOM API to create the html structure (if innerHTML is used this section is zero.) • The create list item element returns the element created. • The form is intercepted and prevented from submitting in the index.js.		**5** **1** **1** **5** **1** **1** **5**	

• The event handler of the form calls the fetch api function, create list item element and attached it to the correct area on the page. • On the recently search list items an event handler is added and using the DOM API it appends the searched albums to the my albums section and removes the button correctly.		**3**	
Bonus • Selecting correct object and POST request executed properly and album is saved to the “myAlbums” section of the json server. • Load the “myAlbums” from the backend server when the page is loaded.		5	
Code • Code Formatting and Style • Errors running the project		**-5** **-5**	

Marking Rubric

Marks	5 Marks Criteria
5	**Task was completed with the highest of proficiency adhering to best practices and followed subject matter guidelines all tasks were completed to a professional standard.**
4	**Task was completed well some minor mistakes. Well above average work shows good understanding of the task and high degree of competence**
3	**Satisfactory work some features missing or incorrectly implemented. Show a moderate level of understanding in the task with room for improvement.**
2	**Below average work. Task was poorly complete. Show understanding of the task and the requirements to implement but implementation was poorly executed.**
1	**Some of the task was completed. Showed a lack of understanding in the subject matter and very poorly executed**
0	**Not completed.**

Marks	**3 Marks Criteria**
3	**Proficient shows a high degree of competence in completing task.**
2	**Capable above average degree of competence in completing task**
1	**Satisfactory shows a satisfactory degree of competence in completing task.**
0	**Shows a limited degree of competence in completing task.**

Marks	**1 Marks Criteria**
1	**Task Completed satisfactorily**
0	**Task was not executed.**