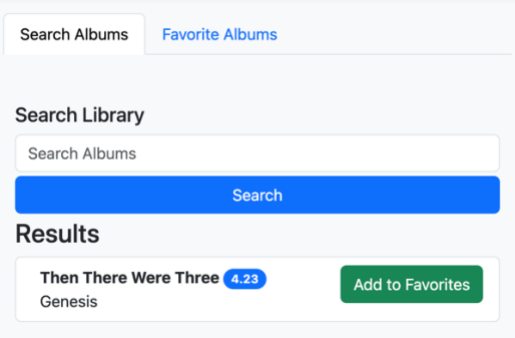
# CPSC 1520 Assignment 3: Album Search and Save Tool

## Introduction

For this assignment you are going to use the Mock API service you set up in Lab 7 and create a simple application using the Web Fetch API to make GET, DELETE and POST requests on both the albums and favorites resources.

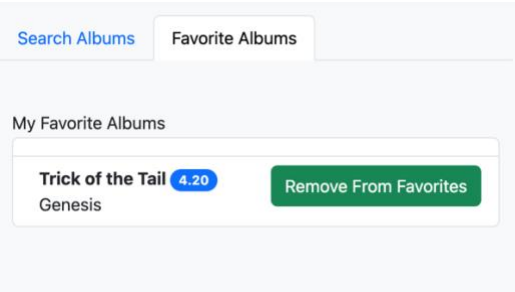
### Search Album View

For the UI view you need to add the search albums functionality. Search the album data objects artistName and albumName properties. Return any results that match the end user query and render the album data using the template provided in the markup. Use interactive templating so that the end user can an album to their favorites list.




### Favorite Albums View

The favorite album view is where you will display your end user favorites list. You can use the same template as the search results album card and change the text to remove from favorites. The data for this view will come from the Mock API favorites resource. Use interactive templating and remove the favorite album from the view and the Mock API service when the remove button is clicked. You must keep the local and remote data in sync.



## Task 1 Tab Between Views

First you need to be able to switch between the two UI tab views. This is done by swapping out class on the Tab Navigation, Search Tab, and the Favorites Tab. You can add remove or edit any class names, id attributes, data attributes as it suits your logic.

```
<!-- TAB NAVIGATION -->
<ul id="tab-navigation" class="nav nav-tabs">
    <li class="nav-item">
        <button
            id="search-button"
            type="button"
            data-tab="search"
            class="nav-link active"
            aria-current="page">
            Search Albums
        </button>
    </li>
    <li class="nav-item">
        <button id="favorites-button" type="button" data-tab="favorites" class="nav-link">
            Favorite Albums
        </button>
    </li>
</ul>
```

**Favorites Tab**

When the user clicks on the favorites tab (Favorite Albums button)  you have to add or remove the following classes.

Search button remove the active class

Favorite albums button add the active class

Favorites Tab remove the d-none class (short for display none, this will show the tab contents)

Search Tab add the d-none class (short for display none, this will show the tab contents)

**Search Tab**

When the user clicks on the Search tab (Search Albums button)  you have to add or remove the following classes.

Search button add the active class

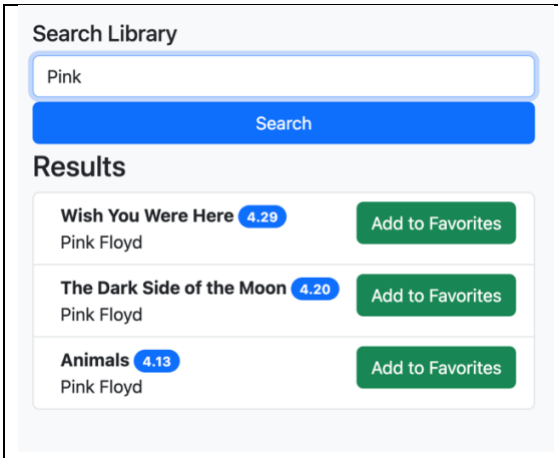
Favorite albums button remove the active class

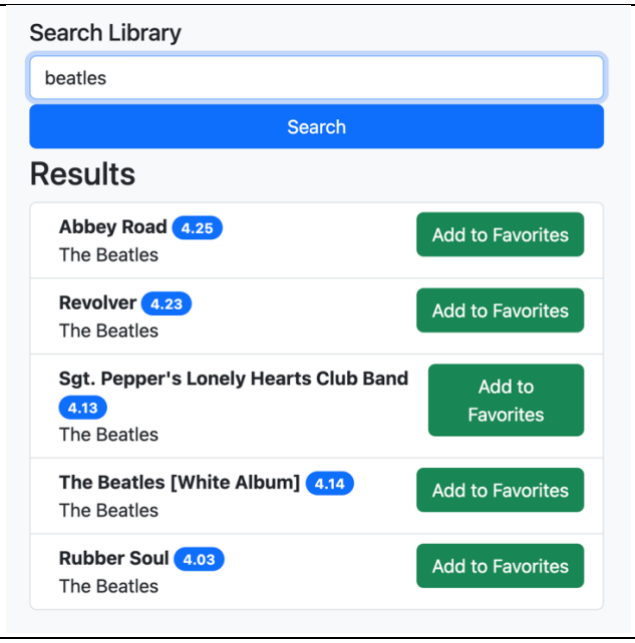
Favorites Tab add the d-none class (short for display none, this will show the tab contents)

Search Tab remove the d-none class (short for display none, this will show the tab contents)

## Task 2 Search Functionality

For the search functionality when the user requests a search, query the album data for matches on both the artistName and the albumName.  Use the template provided in the HTML document to render the results of the query. Below is some example searches that you can check your logic.



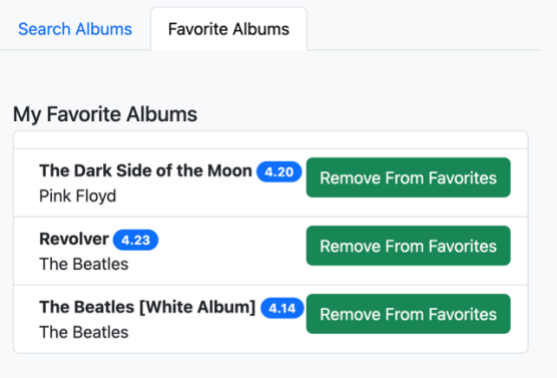



## Task 3 Add to Favorites Button

When the add to favorites button is clicked the album data object should be added to a data store for the favorite's albums. You need to keep track of the albums being added to and later removed from the favorites data store.

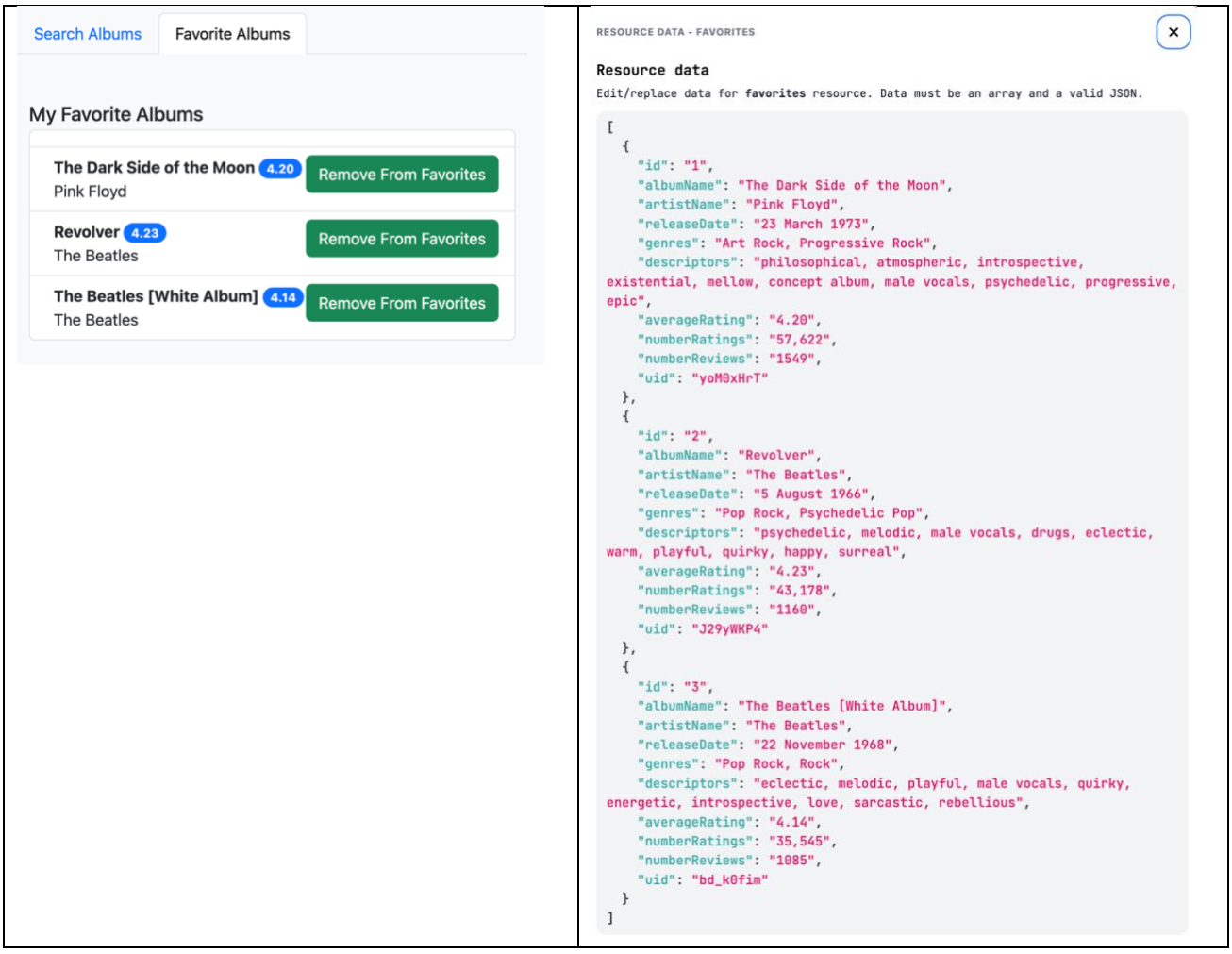
## Task 4 Displaying Favorite Albums

When the user switches to the favorites view, they should see a list of all their favorited albums. You can use the same template that you used in the album search functionality. You will need to change the button text to remove from favorites instead of, add to favorites. When clicked this button should remove the album from the DOM and remove it from the local favorite data store keeping the UI and the data synched. Below is a screen shot of the Favorite Albums Tab when there are favorites in the data store.



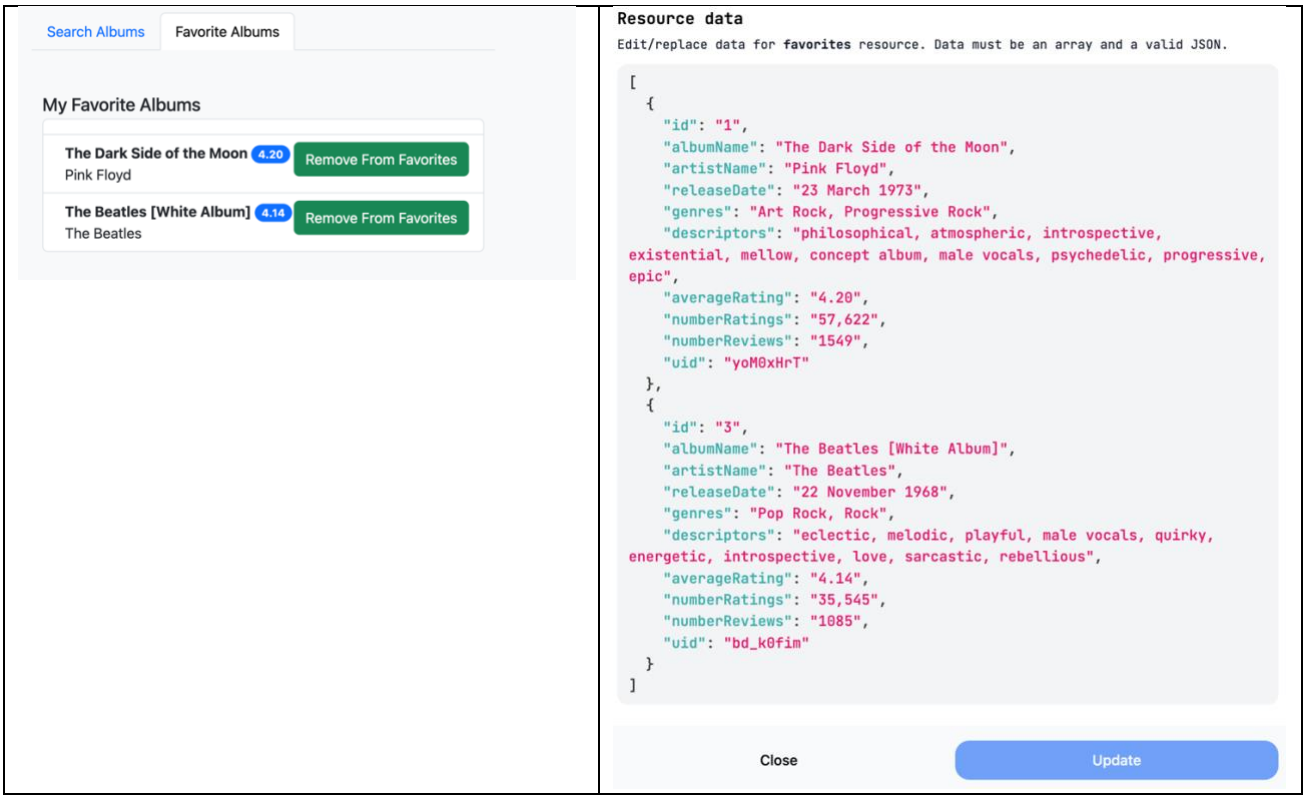
## Task 5  Using Mock API to save  Album Favorites.

The last task is to add any album to the favorites resource for the mock api you created. Only one copy of a favorite album should be added. If an album already exists, you shouldn't see duplicate copies in the UI or in the favorites data inside the mock api dashboard. For the three albums shown below in the users favorite ablums tab the Mock API data resource should contain the data objects for each of these albums.



## Bonus Task Using Mock API remove  Album  from Favorites.

If you want to push yourself when the remove button is clicked remove the item from the DOM and from the favorite's resources. From the last image if you were to remove the Revolver album the favorite album's view is shown below and the favorites resource data.



## Submission Guidelines.

### Netlify Deployment URL

Before your site you have to create a production bundle using vite. Open the VSCode terminal and run the **npm run build** command in the terminal window. This will create a folder called **dist** in the root of the project.  Upload the **dist** folder to Netlify. Check your lab works before submitting the Netlify URL to Moodle

### Assignment Zip Files

Before submitting your assignment zip files **remove the node_modules directory from** the assignment folder. **If your upload node_modules directory is uploaded to Moodle, then you will receive a 5 mark deduction**

### GitHub Repository

Create a GitHub repository for your assignment. Your **node_modules** directory should be excluded from the GitHub repository.

# Marking key

| Tasks | Grade | Marks | Total |
|---|---|---|---|
| **Fetch API**<br>GET Request for the albums to the MOCK API service.<br>GET Request for the favorites resource on the MOCK API service.<br>POST Request to the favorites resource on the MOCK API service. | | **1**<br>**2**<br>**2** | **5** |
| **Search Album Functionality**<br>Clicking on the Search Albums button makes the Tab active.<br>Search uses the mock api service albums resource to filter any results.<br>Search results rendered to the DOM using interactive templating. | | **2**<br>**2**<br>**2** | **6** |
| **Favorite Albums Functionality**<br>Clicking on the favorites albums button makes tab active.<br>Favorite albums resource data on Mock API used to render favorite albums list.<br>Favorite albums are rendered using interactive templating.<br>Clicking on the remove from favorite button removes album from favorites list.<br>Note that you do not have to remove the deleted album from the favorites resource on the Mock API service. | | **2**<br>**2**<br>**2**<br>**2**<br>**2** | **10** |
| **Submission Requirements**<br>Working Netlify Deployment URL<br>GitHub Repository URL<br>Assignment zip files submitted to Moodle | | **2**<br>**1**<br>**1** | **4** |
| **Penalty**<br>**Node_modules directory uploaded to Moodle** | | | **-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. |