Champlain College - Lennoxville
Final project

PROGRAM:	420.B0 Computer Science Technology	
COURSE:	Client Web Applications	
COURSE CODE:	420-330-LE	
WEIGHT:	20% of the final score (5% for analysis, 15% for implementation)	
SEMESTER:	Fall 2023	
INSTRUCTOR:	Francis Gauthier fgauthier@crcmail.net	Office C-239

A client web application to listen to music

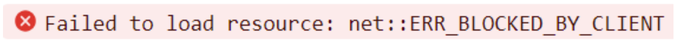
Goal:

1. To analyze the steps to accomplish a complex client web application
2. To practice building more complex React application
3. To practice using HTTP requests, state variables and localStorage
4. To create a complex web application that integrates with the Spotify Web API

A Music Listener Application

You can access the sample application here to know what the result should look like:

Click to open: Example result



```
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
```

*if the Spotify section does not work, with this error in the console, use private mode (incognito) or try another browser

Working in teams

The scope of this project is meant for a team of 2 students. Each student are encouraged to work in pairs, but if students want to work alone, they can.

Collaborating with code

Students working in Teams are encouraged to use **git** (source control) to collaborate. As **git** is not known by all students in this course, the application is separated in two main views. The idea is to facilitate collaboration by providing a way so that each student can work on their separate view without impacting the other student.

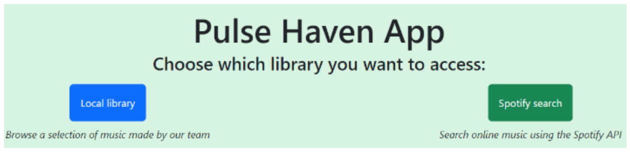
The 3 views

The app should be composed of three main views:

(1) The home page

The home page is simple. It allows us to navigate to the two other views using buttons:

1. The local library
2. The Spotify library



Pulse Haven App

Choose which library you want to access:

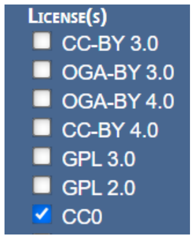
Browse a selection of music made by our team

Search online music using the Spotify API

(2) The Local library view

The local library uses **static files** that are situated in the application assets. The files will not change. The user cannot add or modify the content of that library. The developers will have to select a few songs to be available.

Acquiring local music files



License(s)

- [ ] CC-BY 3.0
- [ ] OGA-BY 3.0
- [ ] OGA-BY 4.0
- [ ] CC-BY 4.0
- [ ] GPL 3.0
- [ ] GPL 2.0
- [✓] CC0

For this view, you will need to acquire 5 to 10 music files that are license free and royalty free.

You can use this website to find free music. Make sure to use the **CC0** filter. This ensures that the music is completely free and does not require to credit the author. You can use any other website as well, if you follow the same guidelines.

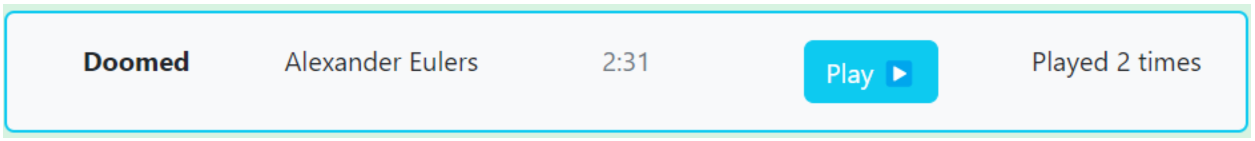
You will need the artist name to be displayed, so locate the artist name for each track/pack downloaded.

Displaying the tracks

Each track should display:

1. The track name
2. The track artist
3. The track duration
4. A play button to listen to the track
5. A counter that tracks the number of full listens

Example:



Doomed Alexander Eulers 2:31 Play ▶ Played 2 times

Playing the track

Clicking on play should display a music player that plays the track selected. The music player should allow to play, pause, adjust volume and go to a specific time.

Keeping track of the listen count

The library should display the number of times that the song was listened to. The count is only updated <u>once a specific song ends,</u> not when the song starts. To do so, use localStorage to keep that information. Therefore:

- The listen count is bound to one browser and one machine
- The listen count can be reset by clearing the localStorage

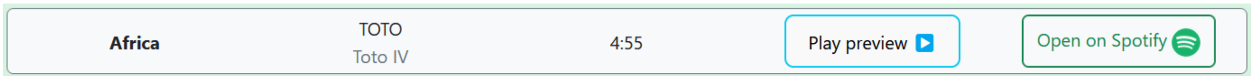
(3) The Spotify library view

The Spotify library uses search term to access the Spotify Web API to display dynamic results. The results are based on the user input and the response from the API.

With the API data, we can display:

1. The track name
2. The track artist
3. The track album
4. The track duration
5. Play a free 30 second preview of the track (only available on some track)
6. Provide a link to that track in the Spotify app

Example:



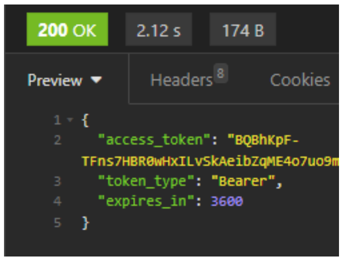
Getting access to the Spotify API

The Spotify API is free to use for developers but requires proper authentication. Your teacher will provide you with a simple way to get a temporary access token that allows to access to many resources of the API.

The documentation of Spotify API is available here:
https://developer.spotify.com/documentation/web-api

In the Getting Started section, you will not have to perform step 1 and 2. Instead, you can request an access token by using this route:

URL	https://mywebapp-775f4.ue.r.appspot.com/spotify/token
Method	GET
Parameters	none



```
{
  "access_token": "BQBhKpF-
TFns7HBR0wHxILvSkAeibZqME4o7uo9m
  "token_type": "Bearer",
  "expires_in": 3600
}
```

In the response, you will receive an access token that expires after 1 hour. At any time, you can call the server to request a new access token.

Accessing the API resources

Once you got an access token, you can use it in your subsequent request to access some routes of the API, like artists, albums, search, etc.

The access token documentation is explained here:
https://developer.spotify.com/documentation/web-api/concepts/access-token

Check out the search page, as this route will be extremely useful for the Spotify view.

Preparation and analysis - first submission

You will be asked to submit a brief analysis of the work to be done. You will be also asked to showcase that you can work with two main concepts required for this project: a music player and the Spotify API.

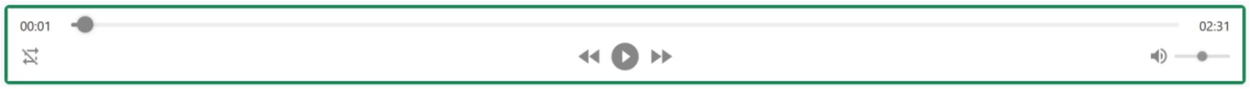
Before the end of the first week, you are required to submit:

1. A prototype of your application. The requirements are:
    - The app is made with React
    - Bootstrap 5.3 is installed
    - The app contains a music player example component
2. A JSON example result of an API call to the Spotify API
3. An analysis document of your components

More details below.

The music player

Install a library that allows you to use a simple, efficient music player within a React application. An example of such a library is https://www.npmjs.com/package/react-h5-audio-player
You can also use any type of music player that you can found in the npm registry.
Example:



The music player must be able to play one song but does not need to have the ability to switch between songs yet.

Some Spotify JSON data

Next, along with your prototype, you must submit an entire JSON file filled with the data associated with one **track**.

1. Locate any song that exists within the Spotify database
2. Fetch the information of that track using the **/tracks/{id}** API route.
3. Copy and save the response JSON into a file called **track_sample.json**

A component analysis

Finally, in your team, look at the example application and try to break it down into several smaller components.
To spot different components, you can think of:

- A component can be reused multiple times and customized
- A component can be used as a specific view/page to organize code
- A component can be used to encompass or isolate some specific logic or behavior

In a Word document, write down a list of all the components that you will implement in your application.
For each component, specify:

1. The component name (a descriptive name)
2. The component purpose
3. It is related to any other components (parent, children, etc.)
4. Are there props expected by this component? If so, what are the name of those props.
5. Any state variables that will be needed in that component (data that the component holds)

Grading

Criteria	**Weight (5% of course grade)**
Application prototype - React + Bootstrap installed properly - Music player is visible and can play a song track	/1.00 /1.00
JSON of Spotify API - Correct retrieval of a track data	/1.00
Analysis document - Proper identification of the required components and breakdown of the application - Proper detail documentation of each component	/1.00 /1.00

Submission - analysis

Before you submit:

- Delete the **node_modules** folder of the project
    - *You can always regenerate it with* ***npm install*** *after your submission*

Then, zip

- your prototype of the React application
- your *track_sample.json* file
- your *analysis.docx* Word document

One submission per team. Specify who you are working with in the comments when submitting.

Submission must be made through LEA. Submission deadline is Friday December 1st 11:30AM (before class). Late submissions accepted. 10% penalty per day late.