D424 – Software Engineering

Task 3

Capstone Proposal Project Name: Music Catalog Web Application (MCWA)

Student Name: Randy Elias Garcia

Table of Contents

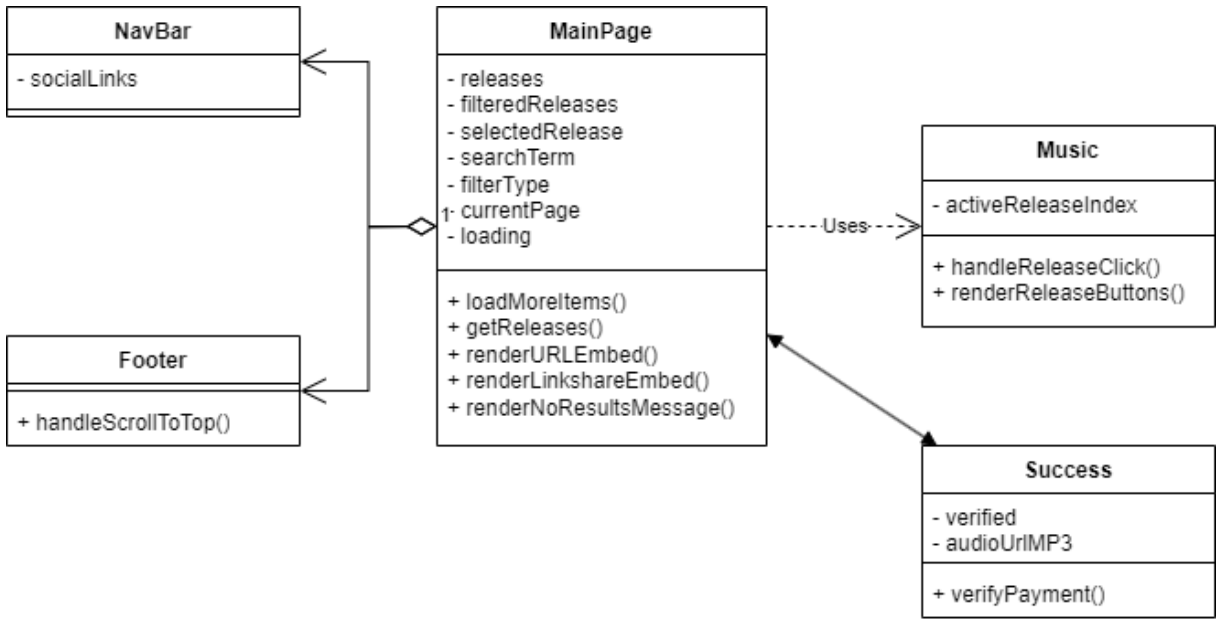
Application Design and Testing

Class Design

In this class diagram, the structure, and relationships of the primary components in a minimum viable product (MVP) for a Music Catalog Web Application (MCWA) are represented. The application is designed to manage music releases, with the ability to generate embedded media players associated with each release card button and allow users to purchase access to exclusive releases. The class design for the Music Catalog Web Application includes various React components that handle different functionalities of the application.

The Music Catalog Web Application is built using React and consists of several primary components such as MainPage, Music, NavBar, Footer, and Success. These components interact with each other to provide a seamless user experience. The back-end is handled by serverless functions that fetch data from a DynamoDB database, verify payments using Stripe, and provide users access to exclusive audio files stored in an AWS S3 bucket.

Class Diagram



UI Design

The User Interface (UI) design of the MCWA provides a seamless and intuitive experience for users, enabling them to effectively navigate an artist's music catalog. The UI design process included the completion of both low-fidelity and high-fidelity wireframes that outline the visual layout and structure of the application.

The UI design of the Music Catalog Web Application focuses on providing a clean and intuitive interface for users to browse, search, and play music from a variety of platforms. The design is responsive, ensuring that the application looks and functions well on both desktop and mobile devices.

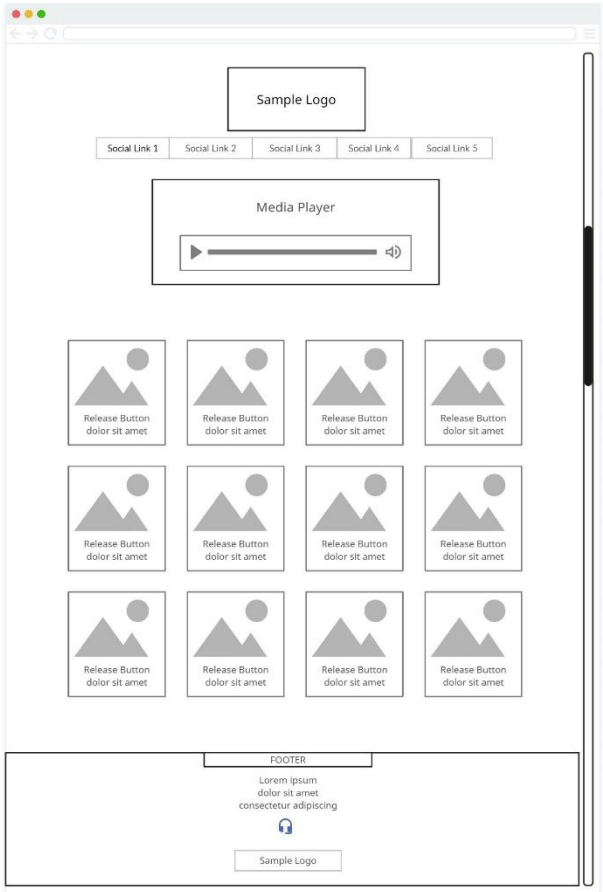


Figure 1: Low Fidelity

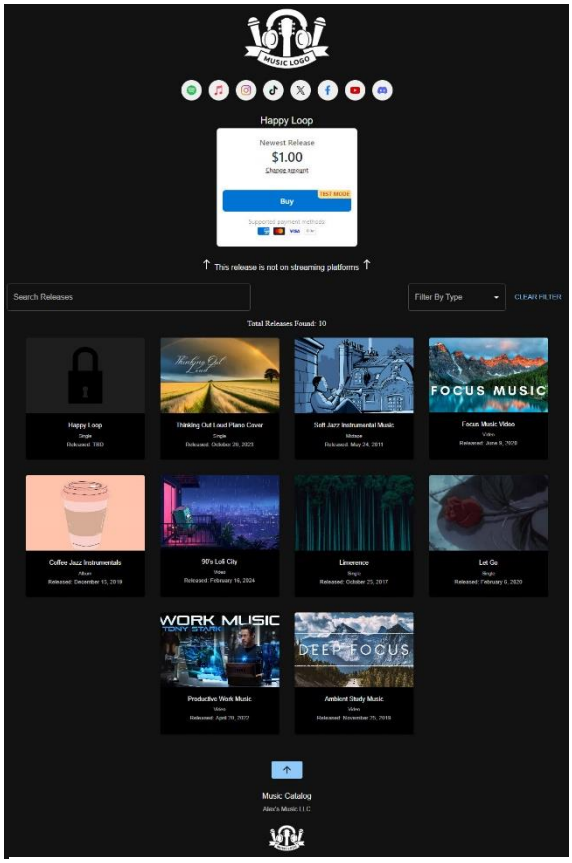


Figure 2: High Fidelity

Unit Test Plan

Introduction

Purpose

The unit testing aims to ensure that individual components of the Music Catalog Web Application work as intended. By testing each component in isolation, we can identify and fix bugs in the development process. The results of these tests will confirm the functionality of the components and highlight areas needing remediation.

Overview

The unit tests focus on key functionalities such as music release card button rendering and proper calling of critical methods. Similar methods are used across the application to maintain consistency in testing. The functions tested include music release card button rendering and calling of “onReleaseSelect” method.

Test Plan

Items

- Development Environment: A development environment set up with Node.js and React installed.
- Testing Framework: Jest with Babel, a popular framework used for testing JavaScript and React applications.
- Sample Data for Testing

Features

- Music release card button rendering
- Proper calling of onReleaseSelect function

Deliverables

- Test Scripts: A test script for the Music component is part of the source codebase.
- Test Results: Screenshots of the test results are provided within this document.

Tasks

- Set up the testing environment with the required tools, including Node.js, Jest, and Babel.
- Ensure test scripts for the selected features are written and testing properly.
- Execute tests scripts using the Jest framework and document the results.
- Analyze the test outcomes and perform necessary fixes.

Needs

- Software Requirements: The application dependencies, such as Node.js, React, and any other required libraries must be installed. These dependencies and their appropriate versions can be found within the package.json file within the codebase.
- Testing Tools and Libraries: For this project, Jest as the testing framework along with Babel need to be installed and configured properly to perform testing.
- Access to Source Code and Test Scripts: Access to the applications source code and test scripts is needed during the testing process.
- Sample Music Release Data: Sample music release data is required to perform adequate testing of components such as Music.js component.

Pass/Fail Criteria

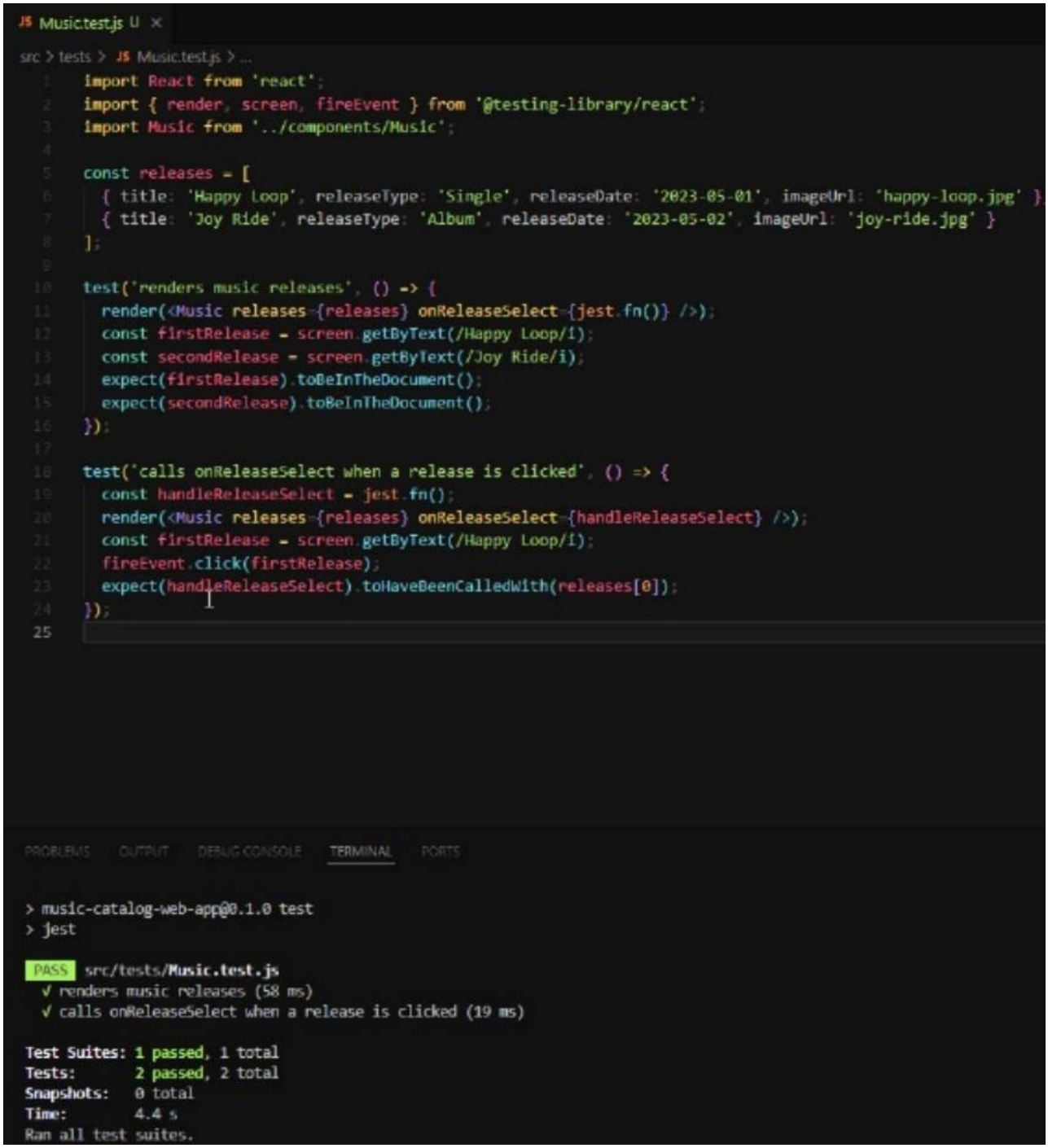
- Music Release Card Button Rendering
    - Pass: Component correctly processes and displays the provided data.
    - Fail: Component does not correctly process and display the provided data.
- Proper Calling of onReleaseSelect Function
    - Pass: Click handling logic in the Music component works as intended
    - Fail: Click handling logic in the Music component does not work as intended

If a test failed during the testing process, the following remediation strategies and documentation requirements were applied to address the issue:

- Identify the root cause: Investigate the failed test to determine the underlying cause of the issue. Examine the error messages, logs, or any other relevant information to understand the nature of the problem.
- Document the issue: Create a bug report or issue a ticket to document the problem. Include details such as the specific test that failed, the error message, reproduction steps, and any other relevant information.

Specifications

Example of a test code screenshot taken from the MCWA source codebase:



Procedures

1. Test Case Preparation: Critical functionalities and features of the application were identified. Test cases were then developed to cover a range of scenarios.
2. Set Up Environment: Made sure Node.js, React, and Jest were all installed properly but running “npm install” to install all dependencies. Also configured Babel and setupTests.js files.
3. Write Test Scripts: Created test files for the Music component in the `tests` directory. Wrote the test suite cases covering key functionalities of component.
4. Execute Tests: Executed “npm test” to run the test scripts and observe the results in the terminal.
5. Document Results: Captured screenshots of test outcomes to record pass/fail status and any issues encountered.
6. Remediation: If needed, reviewed failed tests. Began debugging and fixing issues in the code. After modifying source code, tests were re-ran to ensure fixes were effective.

Results

The test results for the MCWA were obtained after executing the test scripts using the Jest testing framework. The following are examples of the testing results:

- Music Release Card Button Rendering: The test passed which verifies that the Music component successfully renders the titles "Happy Loop" and "Joy Ride" from the releases array.

- Proper Calling of onReleaseSelect Function: The test passed which verifies that when a user clicks on a release (in this case, "Happy Loop"), the Music component correctly triggers the onReleaseSelect callback with the expected release data.

Results Screenshot – all tests passed.

Hosted Web Application

Hosted Web Application Link:

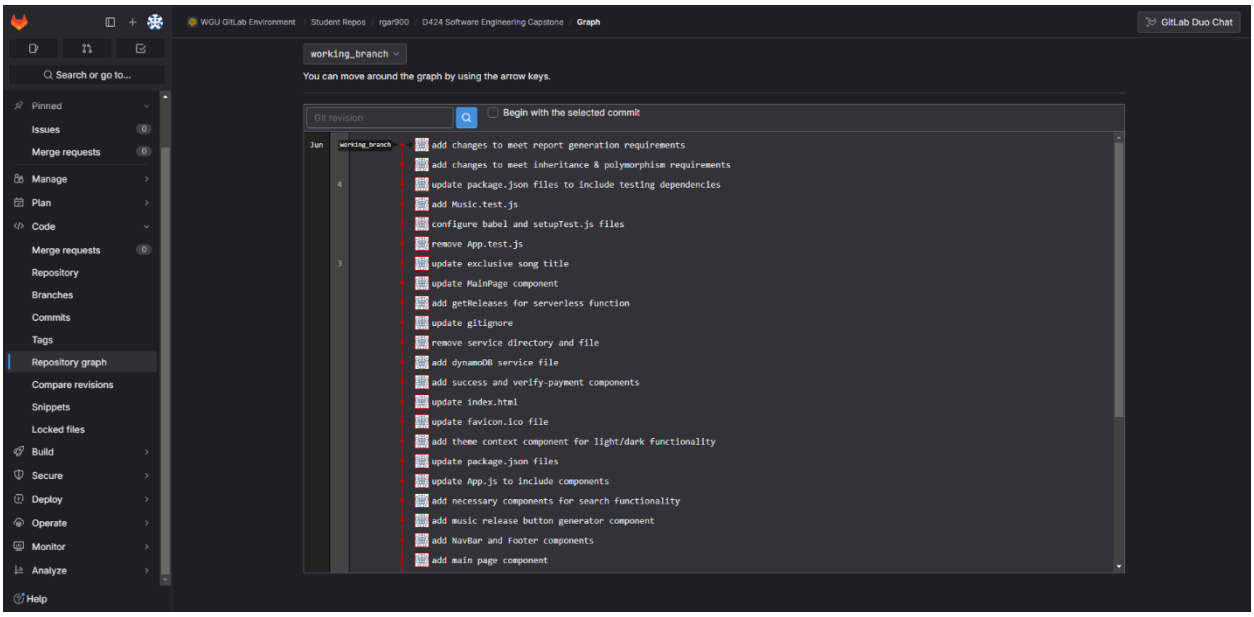
The Music Catalog Web Application is hosted on Vercel. You can access it here.

GitLab Repository & Branch History

GitLab Repository Link:

The source code for the project is available on GitLab. You can access the repository here.

GitLab Branch History:



User Guide for Initial Setup & Running the Application

Introduction

This guide provides step-by-step instructions for setting up and running the music catalog web application. Follow these detailed steps to clone the repository, install dependencies, and run the application frontend locally. Additionally, there are also steps to learn how to configure Stripe for payment processing, set up an AWS account to manage exclusive song files with S3 storage, use DynamoDB to render release buttons, and create an IAM user to handle permissions. By the end of this guide, you'll have a fully functional application deployed on Vercel, complete with backend serverless functions.

Installation and Using the Application

1. Clone the repository to access the source code:
    - Clone the repo from here.
2. Install the required dependencies:
    - Enter “npm install” into terminal **(Using VS Code is recommended)**
3. Run the application:
    - Enter “npm start” into terminal to run application frontend locally.
    - **Note: The Stripe and AWS API functions will not work when running the application locally. This means that the music release buttons will not be generated. This is because Vercel deployment is required to call the backend serverless functions.**

For full functionality please see steps 4-7…

4. Setup Stripe account to obtain secret key:
    - Navigate to Stripe to create an account.
    - **Note: After creating your account, Stripe will prompt you to input all your business information (business name, address, bank information, etc.) You can bypass this step by exiting and skipping the "Complete profile" process which will allow you to use the Stripe service in Test Mode. Test Mode simply means there is no real money involved.**
    - Create a product payment link and ensure that After Payment tab redirects customers to your website URL + `success?session_id={CHECKOUT_SESSION_ID}`
    - Create a buy button for the payment link. The buy button embed code will need to be copied into the MainPage component.
    - Navigate to the dashboard home and copy the secret key. It will be used to create an environmental variable on Vercel.
5. Setup AWS account for S3 bucket storage and DynamoDB:
    - Create an AWS Account:
        - Navigate to AWS and create an account.
        - Follow the on-screen instructions to complete your account setup, including providing your credit card information for billing purposes.
    - Create an S3 bucket for exclusive song files:
        - Sign in to the AWS Management Console.
        - Navigate to the S3 service.
        - Click on "Create bucket".

- Provide a unique name for your bucket.
- Select a region and click "Create bucket" to finalize.
- Upload your exclusive song files to this bucket.

- Create a DynamoDB table for release button generation:
  - Navigate to the DynamoDB service in the AWS Management Console.
  - Click on "Create table".
  - Enter a name for your table.
  - Set the primary key as “releases” (String).
  - Set the sort key as “releaseId” (Integer).
  - Click "Create table" to finalize.
  - Click the “Tables” tab, the name of your table, and then click the “Explore table items” button in the top left.
  - Scroll down and click the “Create Item” button to add items to your table. (JSON formatted data is recommended)

6. Create IAM User to Handle Permissions:

- Create an IAM User:
  - Navigate to the IAM service in the AWS Management Console.
  - Click on "Users" in the sidebar and then "Add user".
  - Provide a username and click Next.
- Attach policies directly to IAM User:
  - Click on "Attach policies directly".
  - Attach the following policies:
    - AmazonS3FullAccess

        - AmazonDynamoDBFullAccess
    - Click "Next" and finally "Create user".
    - Go back to “Users” and click on the newly created username.
    - Click “Create access key” in the top left and then copy the access key and the secret access key. They will be used to create environmental variables on Vercel.

7. Setup Vercel deployment to use serverless functions:

- Deploying the Application to Vercel:
    - Navigate to Vercel and log in or create an account.
    - Import your GitHub repository to Vercel or deploy the application directly from VS Code.
- During the setup process, add the following environmental variables via your project settings:
    - STRIPE_SECRET_KEY - Your Stripe secret key.
    - AWS_ACCESS_KEY_ID - Your AWS Access Key ID.
    - AWS_SECRET_ACCESS_KEY - Your AWS Secret Access Key.
    - AWS_REGION - Your selected AWS region.
- Finalizing Deployment:
    - Once the environmental variables are set, you will need to redeploy the application so that Vercel can inject them during the rebuild.
    - After deployment, your application should be live, and the backend serverless functions should be operational.

8. Using the Application:

- Access your deployed application URL from Vercel.
- Ensure the Stripe and AWS functionalities are working correctly:
    - The DynamoDB functionality will be apparent as buttons will be generated on the main page of the application if the data is being fetched properly.
    - The Stripe functionality can be tested by clicking the “Buy” button, entering mock payment information, and clicking the Pay button.
    - **Note: Be sure to use `4242 4242 4242 4242` for the card number to simulate a successful payment.**
    - After the successful payment is made, the user should be navigated to the Success component page and the exclusive song that was uploaded to the AWS S3 storage should be playable within the audio player.

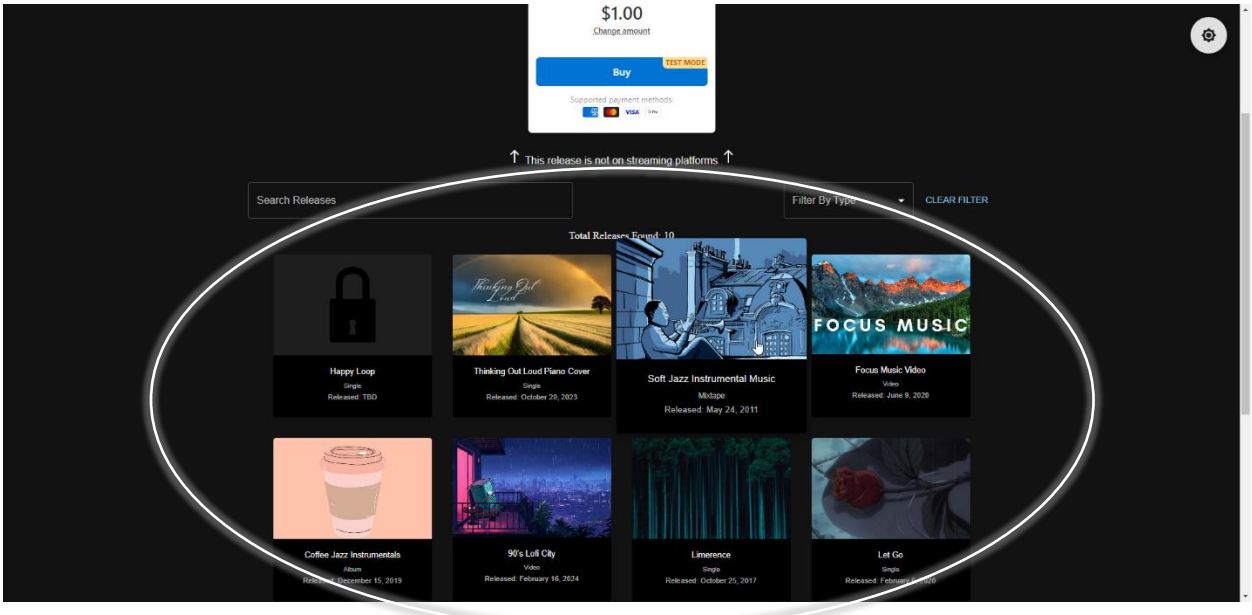
User Guide for Running the Application from User Perspective

Introduction

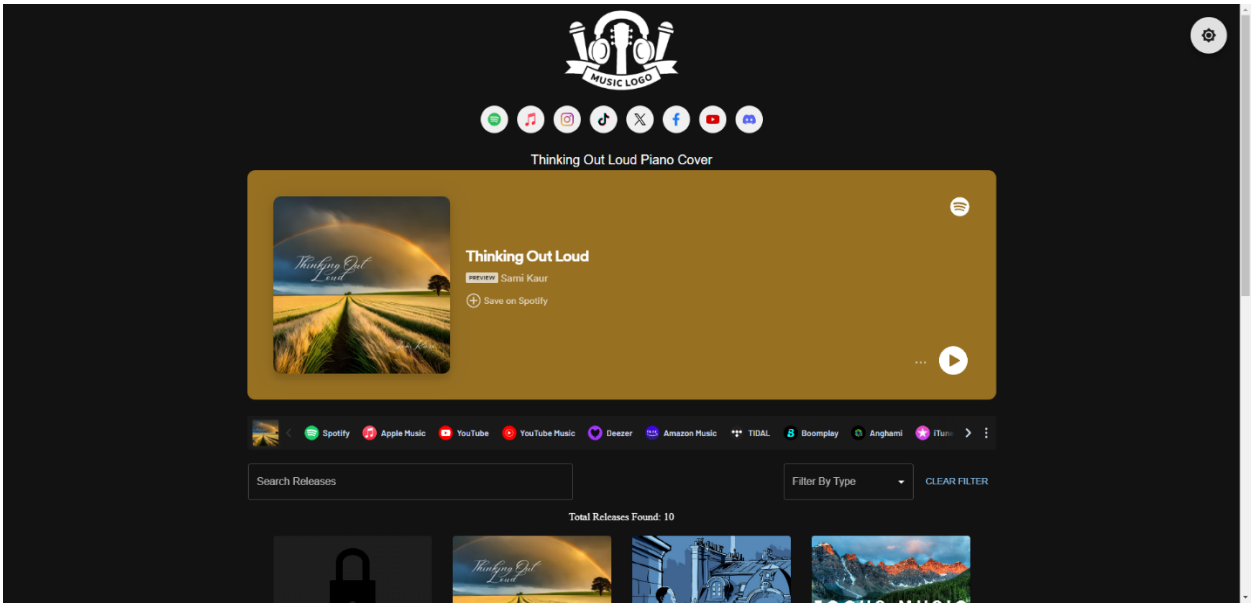
This user guide is designed to offer you a comprehensive overview of all the available functions of the Music Catalog Web Application and guide you through listening to the music, searching for specific releases, and purchasing access to exclusive releases.

Main Page

1. Once you navigate to the “Home” page of the application, you will see the Main Page
2. On the Main Page of the application, you will see the music release card buttons at the bottom of the screen. If you are on a desktop device, you should also notice that the card buttons get a little bit bigger when your cursor hovers over them to indicate which one is being selected.

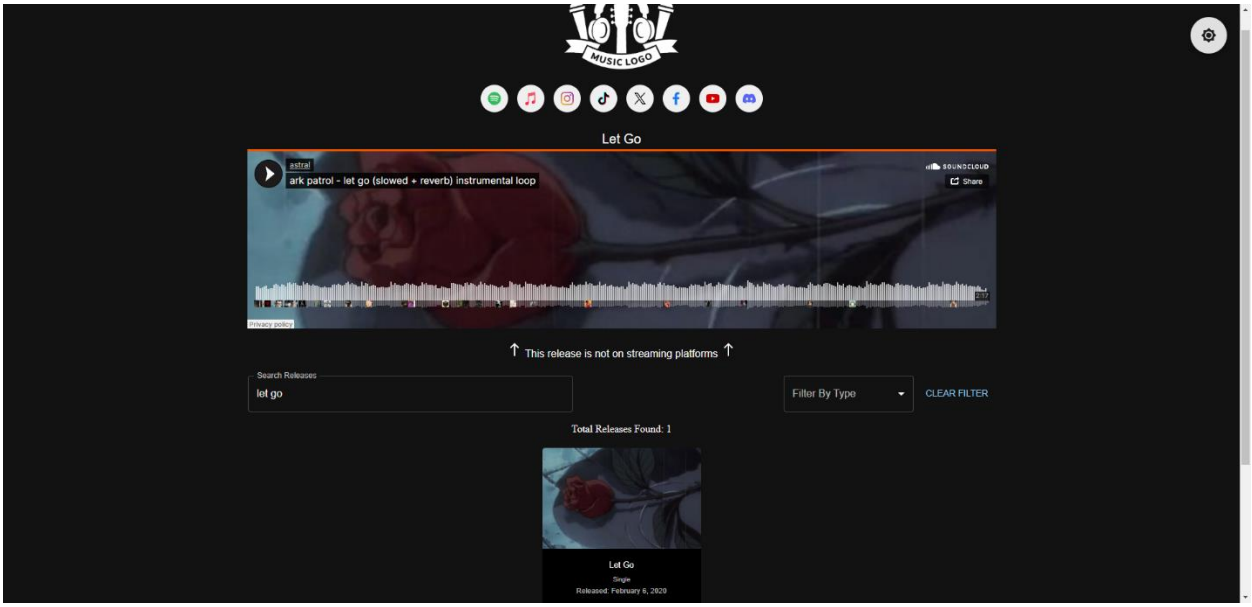


3. When you click on a music release card button, the “embed” section will change and you will be able to interact with it. For instance, if it is a song or a video you will be able to play it right in your browser. Also, depending on which release you choose there may be links that can redirect you to your favorite place to listen to music.

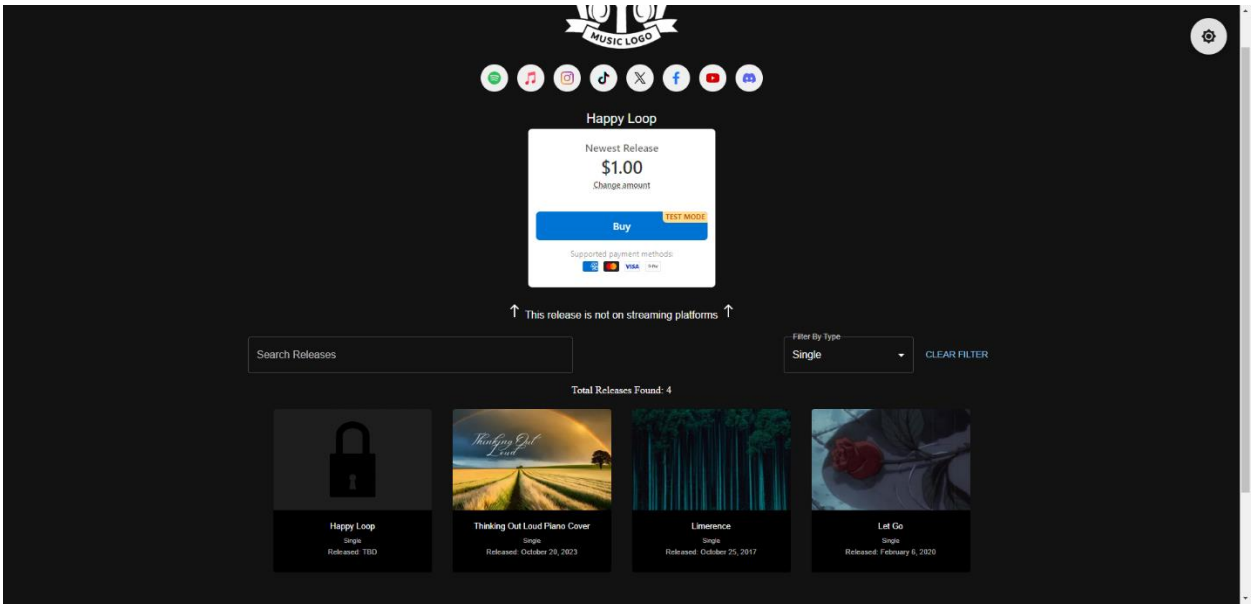


Searching & Filtering

1. You can search for your favorite release within the music catalog by simply typing the song or album title into the “Search Releases” text box.

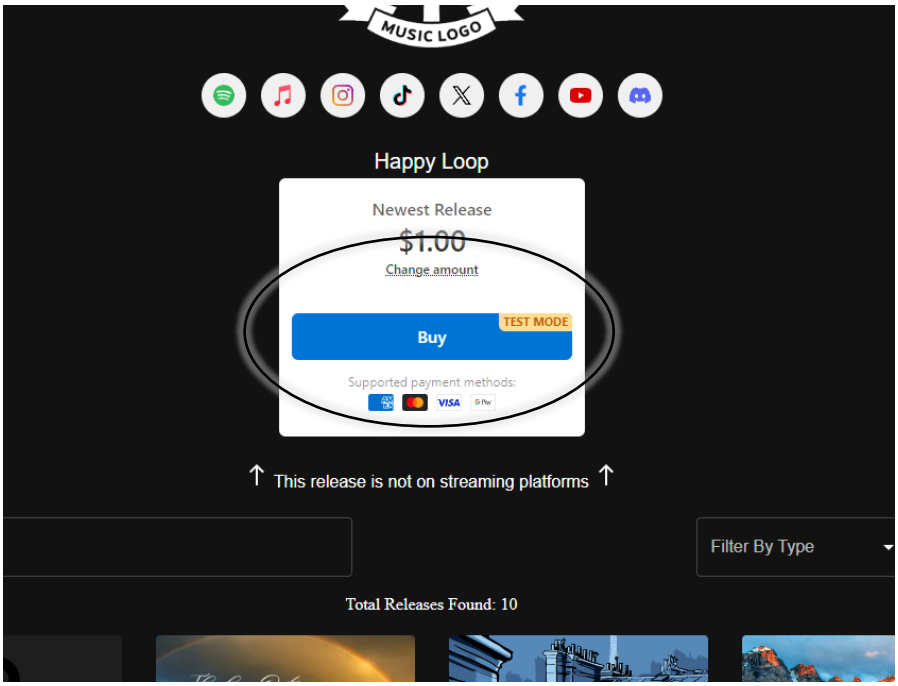


2. You can also filter the music catalog by release type. For instance, if you are only looking for singles you can simply use the “Filter By Type” dropdown menu to select singles.

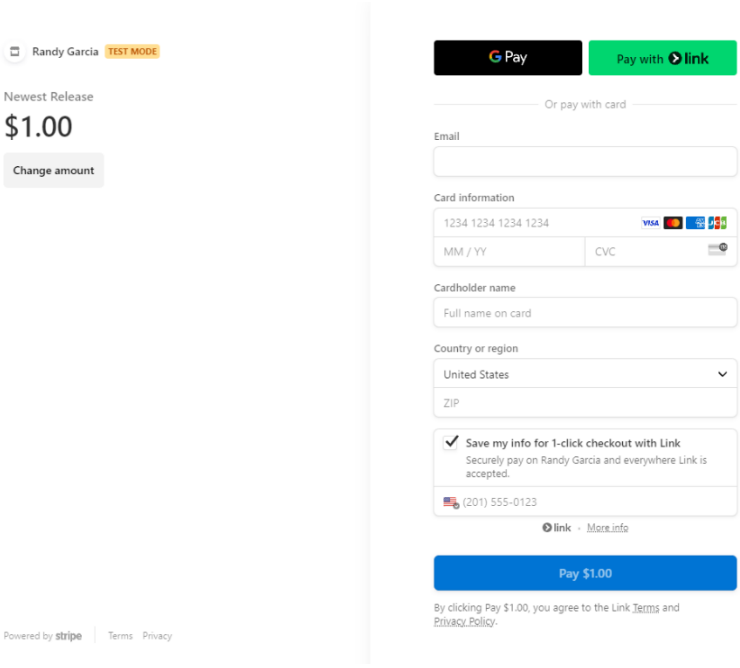


Purchasing Exclusive Access to New Releases

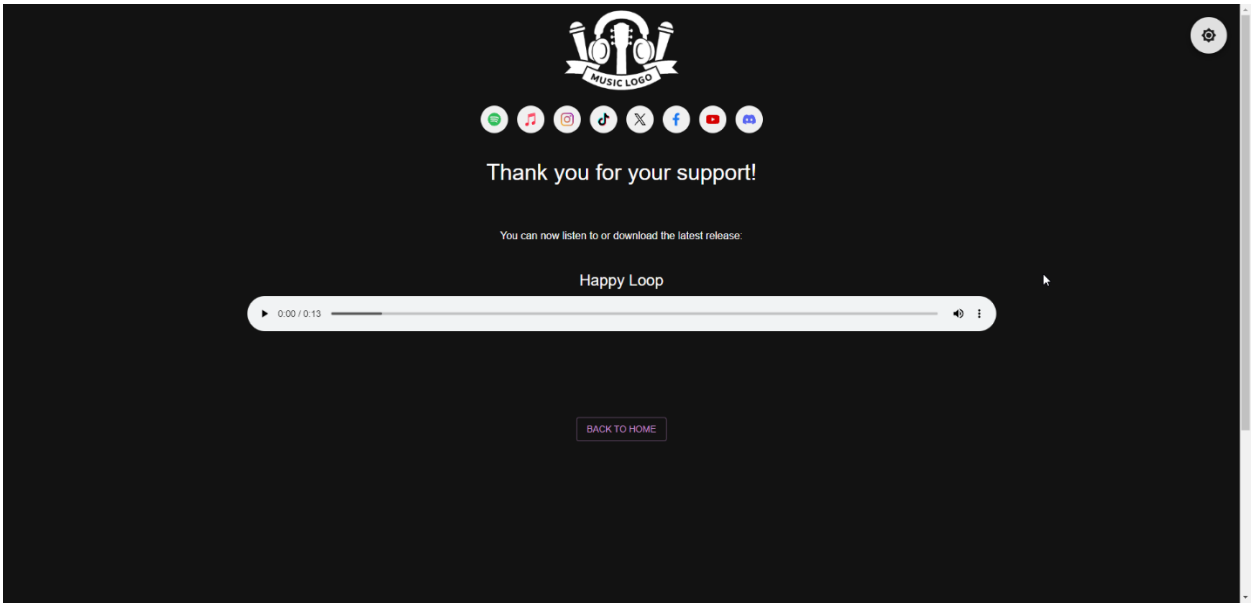
1. Exclusive releases will have embedded “Buy” buttons instead of a media player. These releases will require you to purchase access to listen to them.



2. When you click on the “Buy” button, you will be directed to a checkout page where you can input your payment information.



3. After successfully completing your payment, it will be verified, and you will be redirected to the “success” page of the application where you will be able to listen to the new release via an audio player.



4. Note: If for any reason the payment is not successful, the success page of the application will display “Verifying payment…” to indicate that something went wrong and you have not been granted access to the exclusive content.