Hullabaloo

by

Matt Ritchie, Tyler Getsay, and Aidan Hembree

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Matthew Ritchie
Matt Ritchie – Project Manager

Tyler Getsay
Tyler Getsay – Software Developer

Aidan Hembree
Aidan Hembree – UI Developer & Business Analyst

4/10/2019
Date

Abdou Fall
Abdou Fall – Technical Advisor

4/27/2019
Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2019

Hullabaloo

Prepared by:

Matt Ritchie, Project Manager
Tyler Getsay, Software Developer
and Aidan Hembree, UI Developer & Business Analyst

Students of
University of Cincinnati
College of Education, Criminal Justice, and Human Services
School of Information Technology
cech.uc.edu/it

April 2019

TABLE OF CONTENTS

Section **Page**

Abstract 1

1. Introduction 1-2
   - 1.1 Problem Statement 1-2
   - 1.2 Project Description and Solution 1-2
   - 1.3 Design Objectives 1-3
   - 1.4 Methodology 1-3
2. User Profile 2
   - 2.1 User Profile 2
     - 2.1.1 Application Title 2
     - 2.1.2 Potential Users 2
     - 2.1.3 Software, Interface, and Related Experience 2
     - 2.1.4 Experience with Similar Applications 2
     - 2.1.5 Task Experience 2
     - 2.1.6 Frequency of Use 2
     - 2.1.7 Key Interface Design Requirements That the Profile Suggests 2
   - 2.2 Use Case Diagram 2-2
3. Project Management 3
   - 3.1 Budget 3
   - 3.2 Objectives/Deliverables 3-2
   - 3.3 Project Schedule *(Gantt Chart and WBS)* 3-3
4. Technical Elements 4
   - 4.1 Application 4
   - 4.2 Database 4
   - 4.3 User Interface 4-2
5. Test Plan 5
   - 5.1 Overview 5
   - 5.2 Scope 5
   - 5.3 Objectives 5
   - 5.4 Logging Tests and Procedures 5-2
   - 5.5 Test Cases 5-3
   - 5.6 Pass/Fail Conditions 5-4
   - 5.7 Test Results 5-5
6. Screenshots and Graphics 6
   - 6.1 User Interface (PC) 6
   - 6.2 User Interface (Mobile Devices) 6-5
7. Conclusion 7
   - 7.1 Recommendations 7
   - 7.2 Fall Semester 2018 7-2
   - 7.3 Spring Semester 2019 7-3
8. Appendix A. Additional Info (Tech Info, Poster, Presentation) ..... a
9. Appendix B. References ..... b

LIST OF ILLUSTRATIONS

TABLES

No. Page

Table 1. User Profile……..…...2
Table 2. Budget……...…….....3
Table 3. Project Objectives/Deliverables Due Dates..……....3-2

FIGURES

No. Page

Figure 1. Use Case Diagram.....2-2
Figure 2. Project Schedule and Gantt Chart.……..……...…3-3
Figure 3. Test Results.....5-5
Figures 4-11. Web Based User Interfaces..…….……..…..….6 – 6-4
Figures 12-16. Mobile Based User Interfaces…..……...……..…..6-5 – 6-7

ACRONYMS AND ABBREVIATIONS

API – application program interface
UI – user interface

ABSTRACT

Keeping up with new music and finding similar songs to our favorite tunes is hard. That's why we developed Hullabaloo! Hullabaloo is an online music scraping web app featuring regular audio player controls like shuffle, skipping, volume, and more. Hullabaloo collects data on popular music from websites like YouTube, Spotify, Soundcloud, Reddit, etc. With this data, Hullabaloo will display trending songs. The songs list of the site allows users to browse top songs immediately or a user can further customize the music displayed with sources and playlists. Users are able to add their own sources for Hullabaloo to scrape songs from and then create playlists broken down by genre, artist, mood, length, or popularity. An important feature of Hullabaloo is allowing music tracks from different sources to be added to the same playlists, so a user can listen to a song from Soundcloud, following one from YouTube. With a focus on new and popular music, Hullabaloo updates the songs it displays every couple hours to keep up with the changing landscape of tunes. Hullabaloo is planned to be a cross-platform service, reaching those on Android and iOS mobile devices.

1. INTRODUCTION

1.1 Problem Statement

Today, music is found everywhere and new songs are always being released. It can be hard for the average person to keep up with the many trending tunes or discover new music they would like to listen to. Even with all the different music and social media apps or websites, it can still be difficult to find new songs to fill your playlists. Sure, you can check the top charts for each genre and listen to those songs, but sometimes songs are suggested or trend on Reddit, YouTube, Spotify, or other sites. It would be a lot easier if there was a simple site that displayed recent, popular music and let a user create their own playlists of favorite songs. In conclusion, it's hard to stay updated with popular music or find new tunes and it's annoying to use multiple sites and apps to listen to your favorite tracks.

1.2 Project Description and Solution

What our team has done to solve these problems, is create a web app that scrapes popular or new songs off music and social media sites to compile them into one central place for users to browse through. This will give the user the music he or she wants without a lot of hoops to jump through. Songs are updated frequently so the site is always displaying new and popular music. Hullabaloo allows users to create playlists of similar songs broken down into any categories like: artist, genre, mood, popularity, and more. Our scraping tool finds new tunes from popular music websites and embeds songs onto our Hullabaloo site as well as store them on a server. Hullabaloo also serves as a music player with common functions like playlists, shuffle, repeat, skip, volume, and sorting by genre, artist, album, etc. Since Hullabaloo is a web app, it is a cross-platform service, reaching those on Android and iOS mobile devices. Anyone who appreciates good music will benefit from using our site Hullabaloo to discover new tracks to listen to and enjoy!

1.3 Design Objectives

Our goal is a web app that can be utilized on desktops and mobile phones regardless of operating systems. We wanted to keep simplicity at the forefront with a focus on the music of course so we took a minimalist approach to the frontend. There's an element to display songs or playlists and a second element which is the audio player controls at the bottom of the screen. It was vital that the songs could be listened to on the website so users had a reason to stay, instead of just checking out the new music. Initially, we had talks of including a machine learning algorithm to track a logged in user's listening habits, to later suggest new or similar songs, artists, or genres. This idea had to be pushed out of scope because creating an AI takes a lot of time and calculations to bring to fruition. Nevertheless, Hullabaloo accomplishes what we set out to do, which was create a web app to make it easier for users to find new and popular music.

1.4 Methodology

Our project didn't specifically adhere to one methodology during development. If it had to be given a label, then waterfall with iterations was the methodology for Hullabaloo. We developed the actual scraping and storing of songs first, followed up by designing a landing page to display these songs. From there on, development jumped around to complete different tasks needed to finish the site like: the audio player, a playlists page, and updating the overall design. The design of the frontend has changed a lot and with it came some bug fixes to the backend which is why the project felt like it went through iterations.

2. USER PROFILE

2.1 User Profile

Table 1: User Profile presents the user profile used through the development of the product. It provides the team with details of the audience we are creating this product for and the team will ensure we do not lose sight of the real reason for this project, the users.

Application:
Hullabaloo
Potential Users:
Music lovers and users who want quick access to popular music.
Software, Interface, and Related Experience:
The application will have a basic user interface that anyone with even a little experience with the internet will be able to navigate. Users don't need to be "tech savvy" or "audio buffs" to use all the features of Hullabaloo. In addition, the player controls for the music will be similar to any other controls from other websites, iPhones, car radios, home audio systems, etc.
Experience with Similar Applications:
- iTunes
- Pandora
- Spotify
- Soundcloud
- YouTube

Task Experience:
Most users will have prior experience of how to operate music player controls and scroll through playlists of songs. For example, almost every car has a radio that drivers have probably used at some point. If a user wants to add their own sources of music, the process is straightforward and can be finished quickly.
Frequency of Use:
Hullabaloo can replace a user's regular music app or be a tool to discover new songs. A user can access the app or site for a short time each week to check the new music list or create their own playlists and use Hullabaloo daily.
Key Interface Design Requirements that the Profile Suggests:
- Clean, simple, and responsive UI to ensure use on desktop or mobile
- Landing page that can begin playing new and popular music
- Playlist page that sorts playlists by categories like: genre, popularity, mood, etc.
- Music player controls in nav bar or wrapper to control music in background
- Sources page where users can add new sites for Hullabaloo to scrape song data from

Table 1: User Profile

2.2 Use Case Diagram

Figure 1: Use Case Diagram presents the use case diagram. This represents the requirements we have for our application. After these requirements have been met, more features can be added to better the customers' experience.

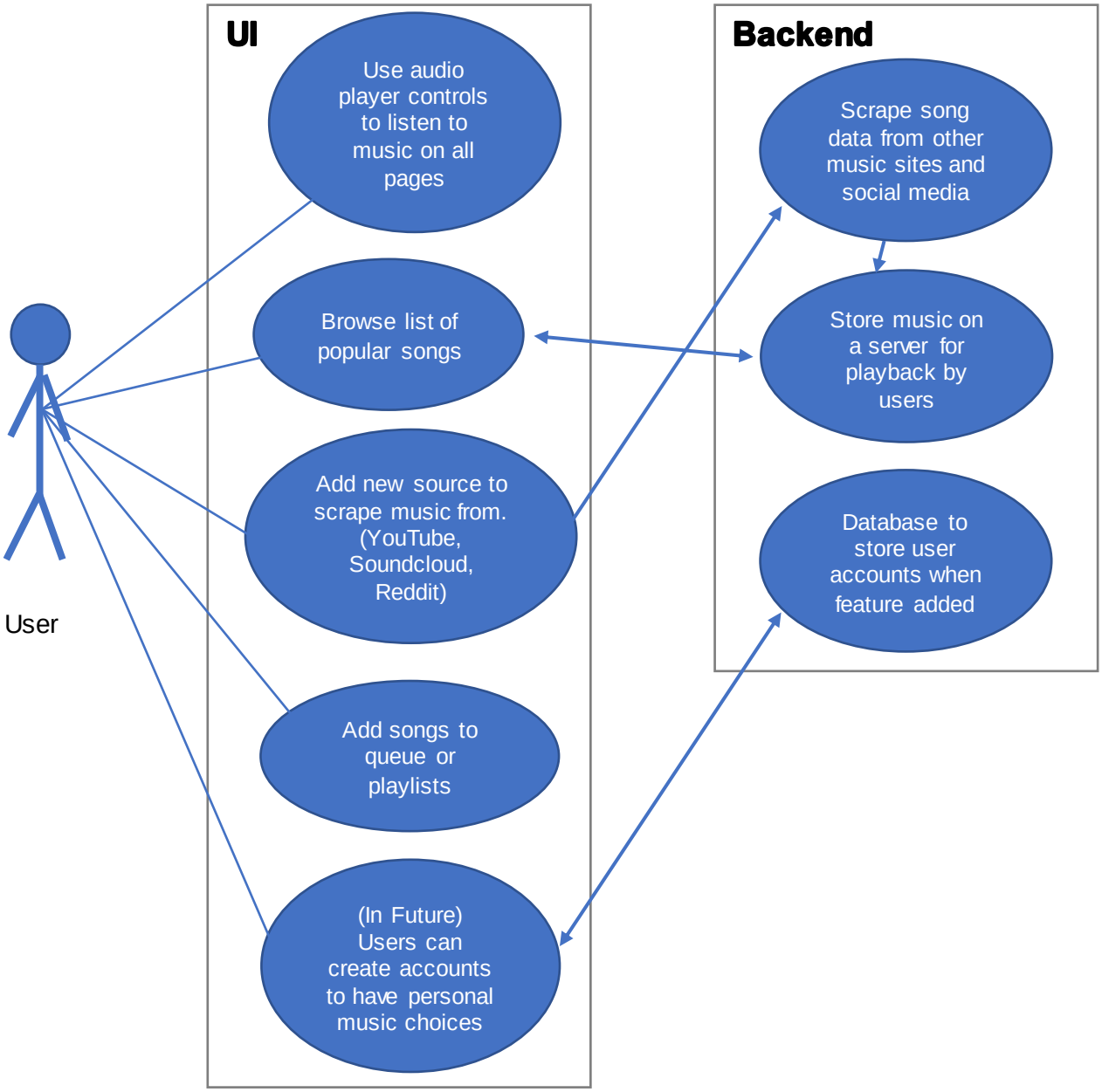


Figure 1: Use Case Diagram

3. PROJECT MANAGEMENT

3.1 Budget

Everything used for the development of Hullabaloo is open-source or owned by our group. While not including labor cost the total price to create this project was zero. If we had not used our own server or one provided by the University of Cincinnati, then a server would have had to be purchased. All software used is open-source and there is no outside materials needed to complete this project. We, as students completing a senior design project, aren't being paid, but we'll calculate labor costs as if we were making $15 an hour for this software development project. We estimated that Hullabaloo took around 40 hours of development and 20 hours of research and planning. To calculate, it's $15/hour for 3 group members working 60 hours, with labor costs totaling at $2,700. Below, ***Table 2: Budget*** presents Hullabaloo's final budget.

Item	Estimated Cost	Actual Cost
Hardware	$0.00	N/A
Software	$0.00	N/A
Server	$0.00	$0.00
Research/Planning	$0.00	N/A
Music Licenses	N/A	$0.00
Presentation	$40.00	$0.00
Labor	$2,700.00	$0.00
Total Costs	**$2,740.00**	**$0.00**

Table 2: Budget

3.2 Objectives/Deliverables

Table 3: Project Objectives/Deliverables Due Dates presents the project deliverables and deadlines. The left side of the table displays the timeline due dates of the major phases of the project. The right side includes some of the major milestones the capstone course requires.

Major Project Milestones (Deliverables)

FALL OF 2018 MILESTONES				
Initiation Phase	9/10/18		Team Contract Written	9/24/18
Research Phase	9/21/18		Project Abstract Drafted	10/15/18
Design Phase	10/5/18		User Profile & Use Cases Drafted	10/22/18
Security Testing	11/19/18		Elevator Speech	10/29/18
Implementation Phase	12/20/18		Fall Report Drafted & Fall Presentation	11/26/18
SPRING OF 2019 MILESTONES				
Alpha Test Phase	1/14/19		Spring Presentation	3/11/19
Deployment Phase	2/4/19		Tech Expo	4/9/19
Security Testing	2/18/19			
Beta Test Phase	3/11/19			
Redesign/Software Update Phase	4/1/19			

Table 3: Project Objectives/Deliverables Due Dates

3.3 Project Schedule

Figure 2: Project Schedule and Gantt Chart is our planned schedule with the major milestones listed.

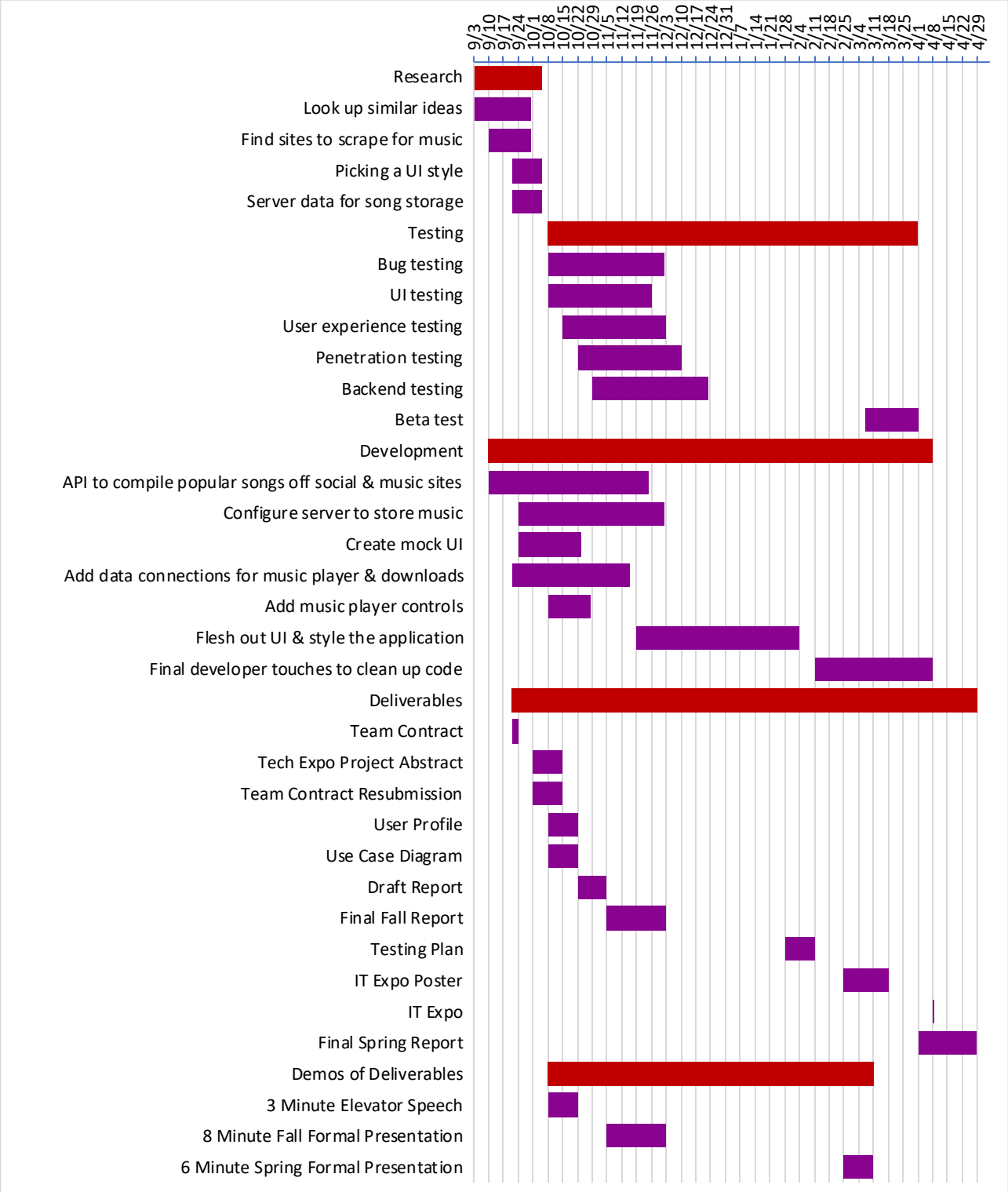


Figure 2: Project Schedule and Gantt Chart

4. TECHNICAL ELEMENTS

4.1 Application

We wanted our application to run smoothly on the web or mobile so we decided to use Vue JS as the frontend programming tool with Laravel supporting the entirety of the app. They are popular development tools to program the frontend, backend, and the UI or webpage. Vue JS is an open-source JavaScript framework builder specializing in single page applications. Laravel is a PHP web framework utilizing the model-view-controller development pattern. Docker is a container software that runs the music scraping feature. Hullabaloo can let a user access our music database through an API running in the background of the application. More information on our music database is in the next section. A feature we considered, but decided not to implement was a form of AI or machine learning. We dream of letting the Hullabaloo bot learn what music a user enjoys listening to and using that information to suggest new songs, genres, or artists. If given the opportunity, development could continue on Hullabaloo to broaden the project's scope.

4.2 Database

Our first hurdle to jump over was how to store popular music for our users to listen to. We needed a database and through research we turned to PostgreSQL – a powerful, open source object-relational database with a SQL style. Additionally these music files would need data querying and manipulation which GraphQL does a really good job with. Hullabaloo continuously accesses this database using the API to save data on popular songs, display playlists, and most importantly – play songs. This is the largest part of the project and will take the most development, but the scraping API and the database are the heart of Hullabaloo.

4.3 User Interface

The UI of Hullabaloo was planned from the beginning to be a simple and clean audio player, so a user can immediately find new music for their playlists. The player controls are at the bottom center of the screen - on a navigation bar, for ease of use and with a focus on the music. Song and playlist menus are sorted as lists. Users are able to browse these lists to quickly find new and popular music. Sources for Hullabaloo to scrape from can be added on another tab and it's a simple way to create a separate playlist of songs. Users can add music to the queue and/or playlists with just the touch of a button. Vue JS, Laravel, and Tailwind CSS support us in developing the UI. Tailwind CSS was used to stylize the Hullabaloo site as it allows quick, simple, and custom user interfaces.

5. TEST PLAN

5.1 Overview

This section describes the testing methodology for Hullabaloo and should be reviewed by the following individuals:

- Developers (Frontend or Backend)
- Project Managers
- Q/A & QC Personnel
- Testers
- Team Members

5.2 Scope

The scope of testing is to test the operation and functionality of Hullabaloo on desktop and mobile devices. Laptops, tablets, smartphones, and other mobile devices capable of navigating to our web app should be able to use Hullabaloo to its full potential. Tests will be created based on the requirements of our project.

5.3 Objectives

The objective of testing is to verify the web app and features are working as expected and functioning up to par with developers or users. The tests are designed to test key features of the Hullabaloo web app, the functionality of it, and the responsiveness of the web page. Tests allow developers to confirm that the project is working as intended or could identify areas of improvement.

5.4 Logging Tests and Procedures

During a test, if a bug is found then the team member conducting the test should record what can be observed about the bug. This includes information on the actual issue and any details about the hardware or software of the device. Group will meet and discuss bugs found during the last testing session and brainstorm methods to solve the issues. Bugs will be dealt with according to severity.

Steps for testing are as follows:

1. Create test scenarios and cases.
2. Prepare test document of cases and expected results
3. Run test and record the actual results and any bugs

Below are the tests to be performed:

1. Responsiveness Test – This test confirms if the web app will change size and shape to fit the device it's being accessed on. This test is technically run first as the pass/fail is whether the web app displays correctly.
2. UI Test – This test consists of using the web app like normal. Clicking buttons, links, and selecting components on the page to ensure everything is working correctly. This test is run first on each new page loaded as a bug on the UI would impede any following tests.
3. Music Test – This test will test whether sound plays when clicking a song and it goes further by making sure all the audio player controls work. Songs should be able to be added to the queue and playlists as well.
4. Functionality Test – This test will determine that Hullabaloo is actually working. The song scraping feature is the basis of our project. Users should be able to add a source to scrape songs from and the API should begin display songs on the site within the next hour. New songs added to the site will then tested by the "music test" above.

5.5 Test Cases

1. Open Hullabaloo web app
    a. Steps:
        i. Enter URL or click Hullabaloo icon (on mobile)
    b. Expected Outcome:
        i. Hullabaloo loads and displays homepage
        ii. Page layout should fit the device it's on – "Responsivenss Test"
2. Test the homepage
    a. Steps:
        i. Scroll down the page
        ii. Click any links or buttons
        iii. Click links in the nav bar (next page will load, ignore this page for now and return to the homepage to continue the test)
    b. Expected Outcome:
        i. Hullabaloo homepage is displayed correctly on device
        ii. Links and buttons work as intended
3. "UI Test" the song list page
    a. Steps:
        i. Navigate to the song list by any link or button
        ii. Click all buttons or links to confirm they work
    b. Expected Outcome:
        i. Song list is displayed correctly on device
        ii. Scrolling down should load more songs
4. "Music Test" the song list page (continue testing the audio player controls on the next 3 webpages. Final step below only, but same expected outcomes.)
    a. Steps:
        i. Click a song to play it
        ii. Add a few songs to the queue
        iii. Click all buttons on the audio player (skips, pause, volume slider/mute, shuffle, scrub through the song's timeline, and delete a song from the queue)
    b. Expected Outcome:
        i. Music plays and volume can be adjusted by the slider
        ii. Queue actively updates with songs added or removed
        iii. Skip buttons will replay current/previous song or play next song
        iv. Pause/Play works
        v. Scrubbing through the song's timeline will fast forward or rewind the current song playing
        vi. Shuffle will play songs from the queue in random order
5. Test the playlists page
    a. Steps:
        i. Navigate to the playlist page by any link or button
        ii. Click all buttons or links to confirm they work
        iii. Attempt to create a playlist then add and remove songs from it
    b. Expected Outcome:
        i. Playlists page is displayed correctly on device
        ii. Playlist should be created and have songs added to it

iii. Audio player should be able to skip through playlist songs

6. Test the sources page
    a. Steps:
        i. Navigate to the sources page by any link or button
        ii. Click all buttons or links to confirm they work
        iii. Attempt to add a new source (YouTube channel, Soundcloud chart, or Reddit Subreddit)
    b. Expected Outcome:
        i. Sources page is displayed correctly on device
        ii. Sources can be added and songs will begin to be scraped and displayed on the song list page within an hour

5.6 Pass/Fail Conditions

It's expected that for Hullabaloo to meet expectations all these tests must pass. The web app should display correctly according to the device's size. Everything on the UI should work and do what's expected of it (links, buttons, adding songs/playlists/sources). Music should be playable and audio player controls easy to figure out. Added sources should update song list page with new songs within an hour that can be listened to or added to the queue or playlists. Any unwanted result or failure with a request should be documented and investigated further to figure out the problem causing it and a solution to solve it.

5.7 Test Results

Figure 3: Test Results displays an example of what a completed test document looks like for Hullabaloo.

Test Case #	Input	Expected Output	Actual Output	Pass/Fail	Reason for success/failure	Date
1	Open Hullabaloo. Enter URL or click Hullabaloo icon (for mobile).	Hullabaloo web app opens to the homepage on PC or song list on mobile devices.	Hullabaloo opens quickly.	Pass	Web app loads quickly on PC & mobile devices.	4/8/2019
2	Click every button & link on the homepage.	All buttons & links work.	All buttons & links work.	Pass	Buttons & links aren't broken.	4/8/2019
3	Click every button & link on the song list page.	All buttons & links work.	All buttons & links work.	Pass	Buttons & links aren't broken.	4/8/2019
4.i & 4.ii	Click a song to play it. Add a few songs to the queue.	Any song will play. Songs can be added to queue in order.	Song plays. Songs are added to queue.	Pass	Clicking a song will play it immediately. Adding a song to the queue just takes a button click.	4/8/2019
4.iii	Click every button on the audio player.	All buttons work.	All buttons work except shuffle and skip buttons when the queue is empty.	Pass	Buttons & links aren't broken, except when song queue is cleared there aren't any songs to skip or shuffle.	4/8/2019
5.i & 5.ii	Click every button & link on the playlists page.	All buttons & links work.	All buttons & links work.	Pass	Buttons & links aren't broken.	4/8/2019
5.iii	Create a playlist, then add & remove songs from it.	Playlist will be created by naming it. Songs can be added and removed from playlist.	Playlist cannot be created on this page.	Fail	"Add playlist" button doesn't do anything. The "add playlist" widget doesn't pop up.	4/8/2019
6.i & 6.ii	Click every button & link on the sources page.	All buttons & links work.	All buttons & links work.	Pass	Buttons & links aren't broken.	4/8/2019
6.iii	Add a source.	Sources can be added and will populate the song list with new music.	A YouTube, Soundcloud, or Reddit source can be added and songs begin showing up within an	Pass	"Add souce" widget pops up and allows any YouTube, Soundcloud, or Reddit source to be scraped by our API for new music.	4/8/2019

Figure 3: Test Results

6. SCREENSHOTS AND GRAPHICS

6.1 User Interface (PC)

Figures 4 - 11: Web Based User Interfaces displays screenshots of the user interface on a desktop or laptop PC. Starting with the homepage, then moving on to songs, playlists, and sources. The final two screenshots show what adding a source looks like.

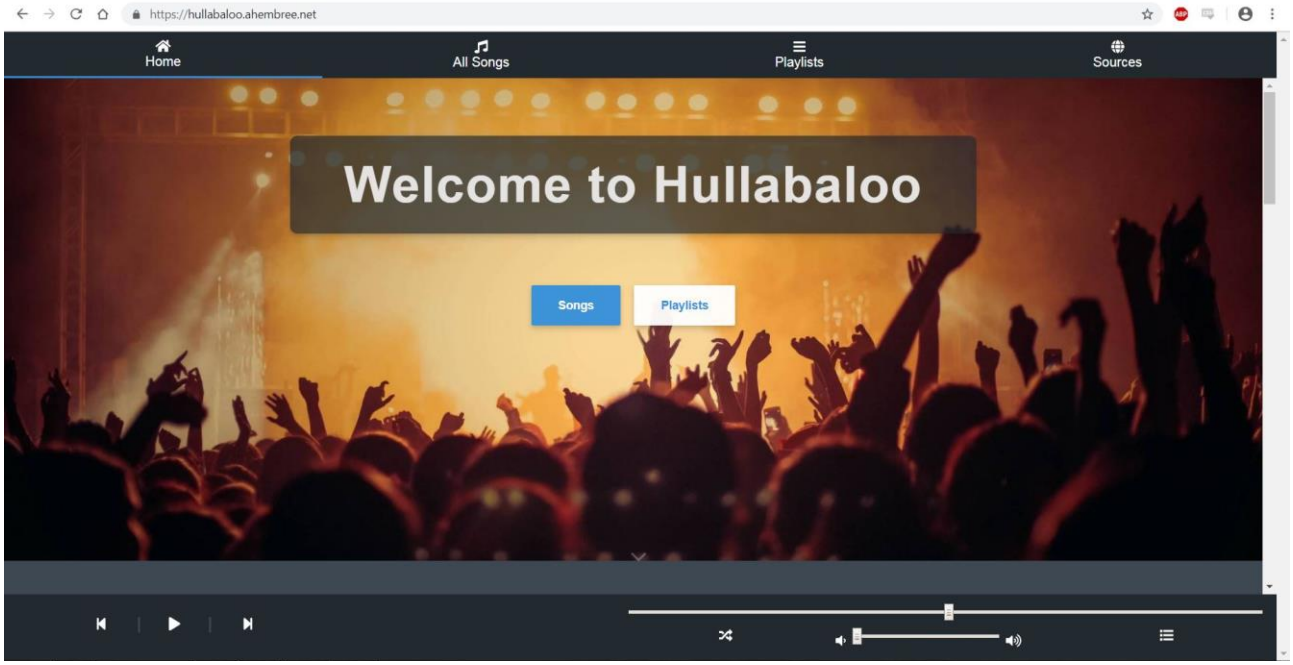


Figure 4: Homepage

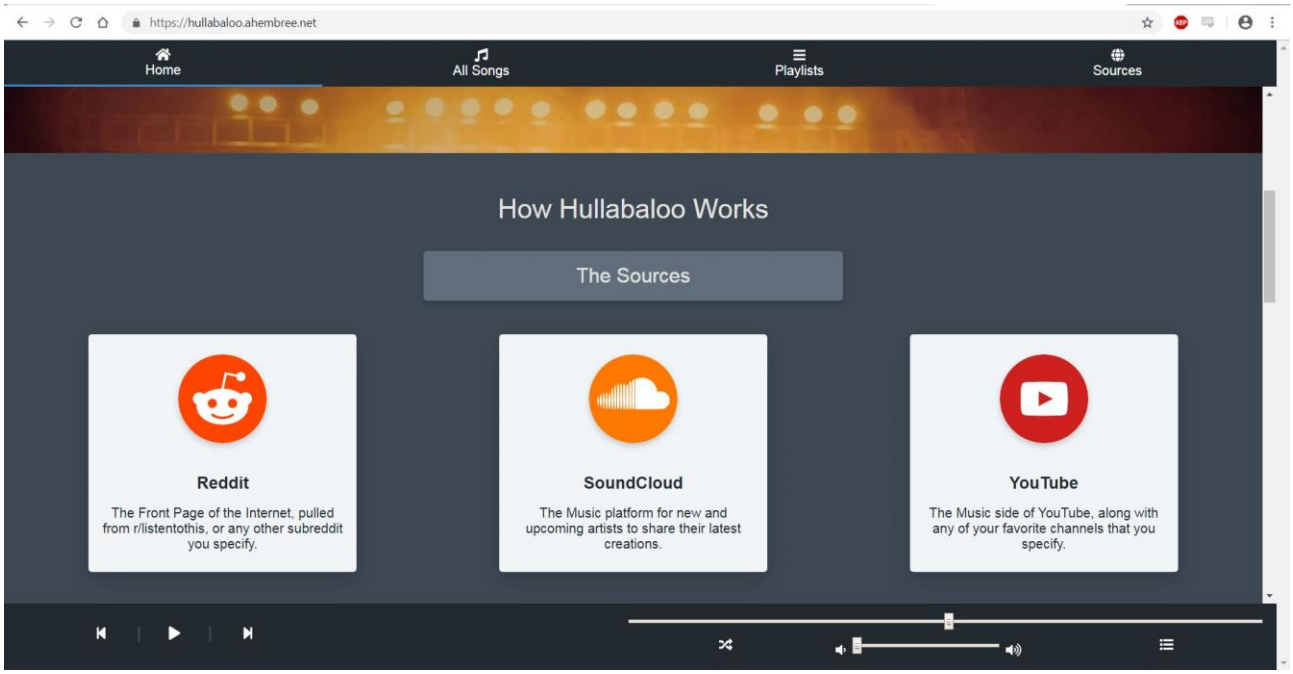


Figure 5: Homepage (cont.)

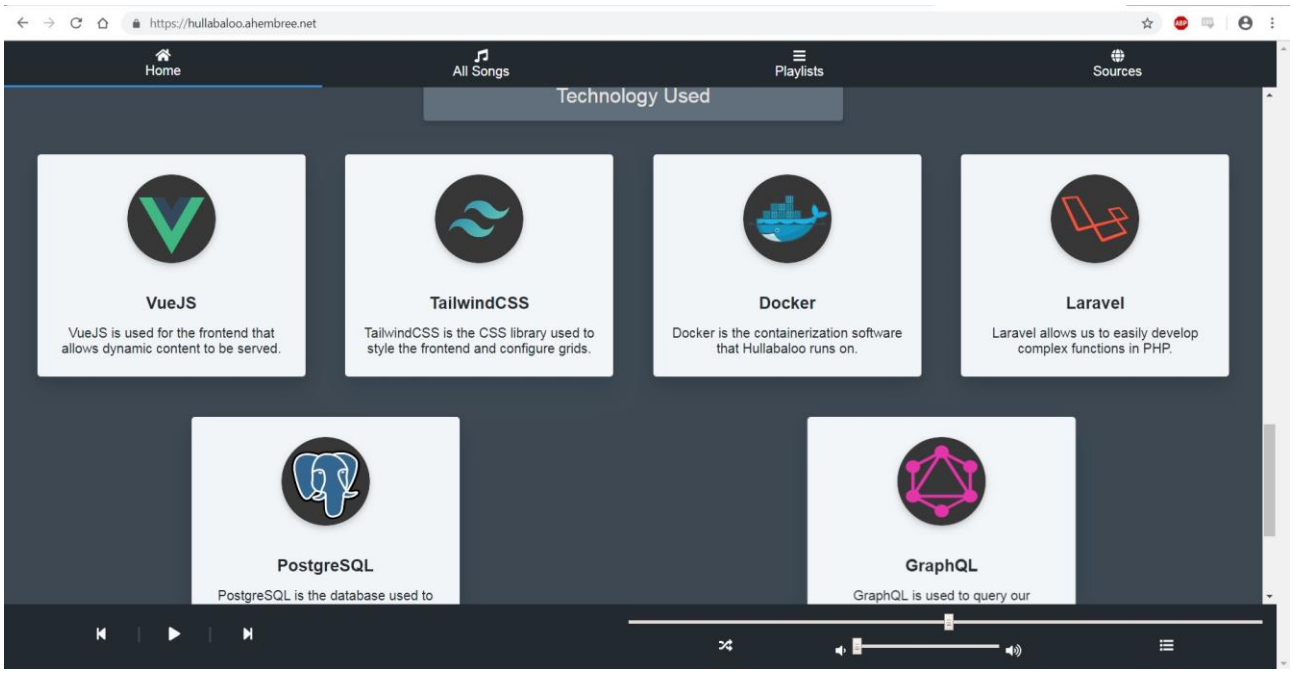


Figure 6: Homepage (cont..)

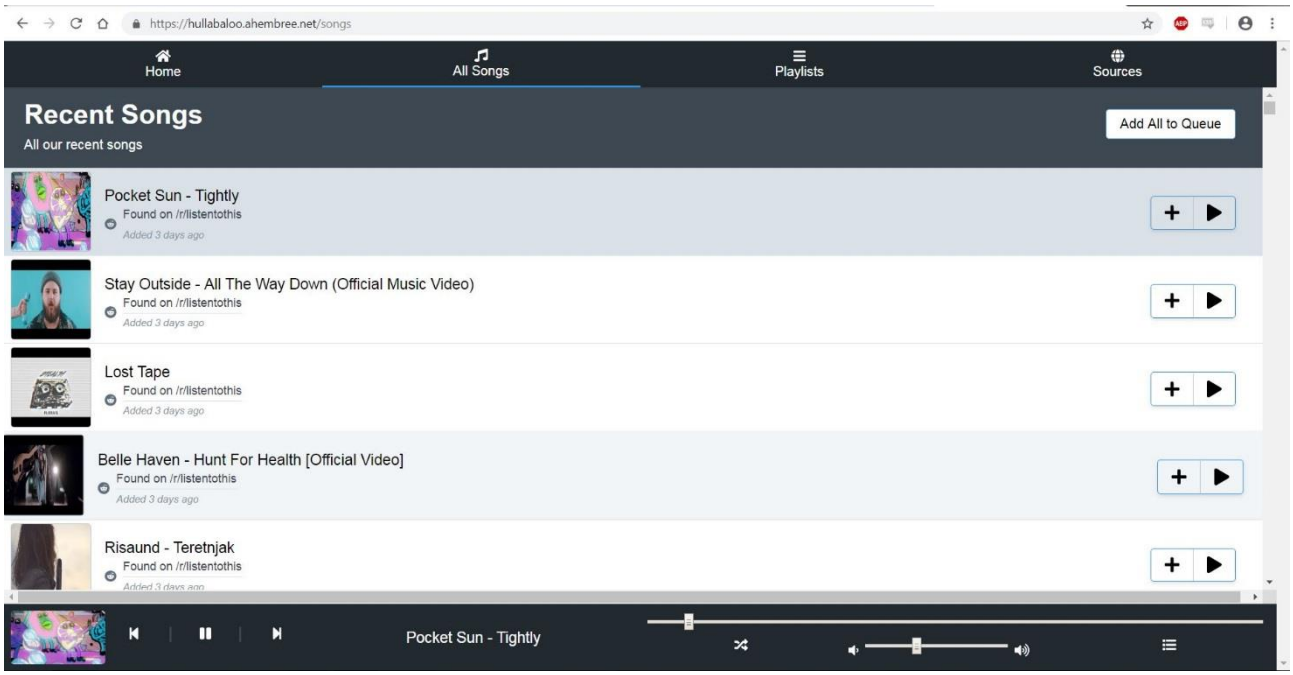


Figure 7: Song list page

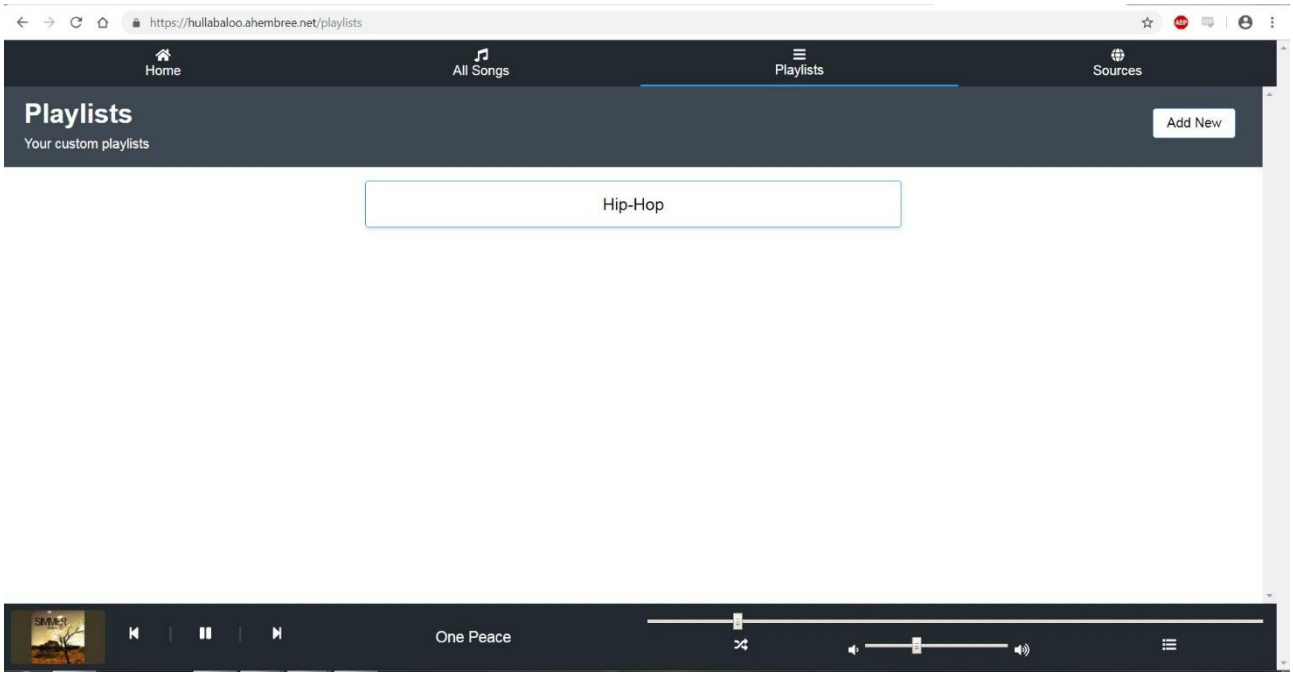


Figure 8: Playlists page

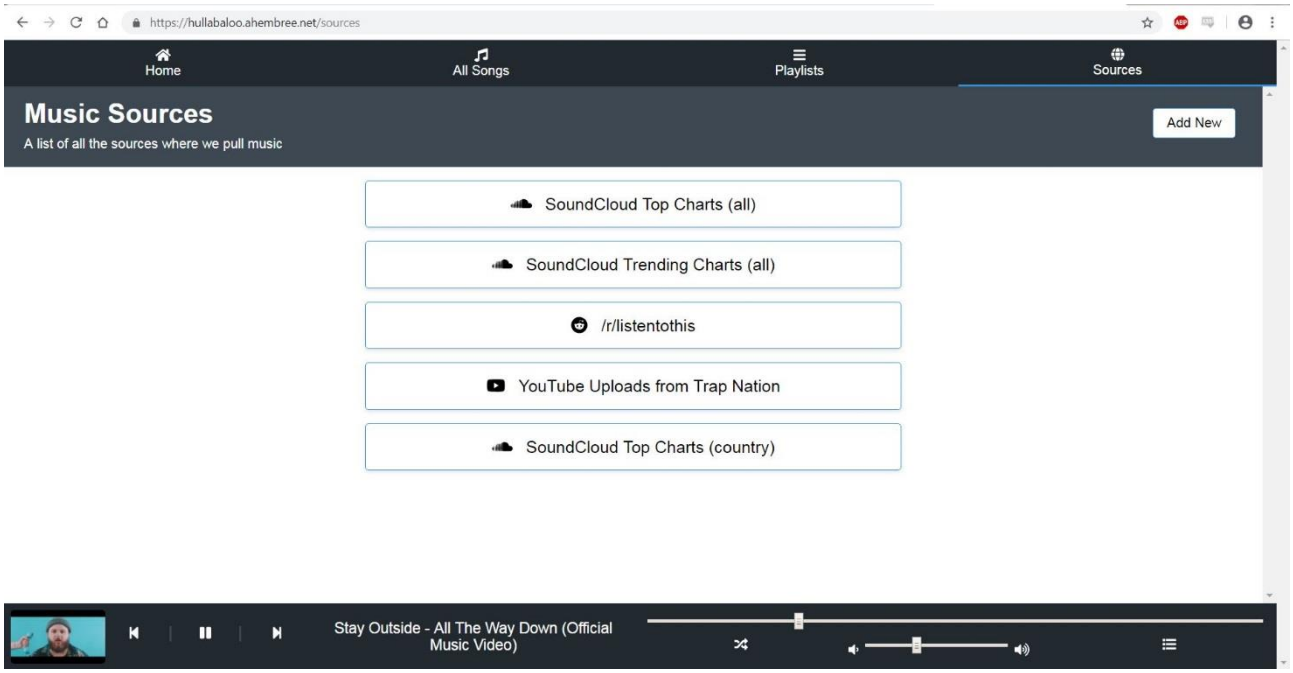


Figure 9: Sources page

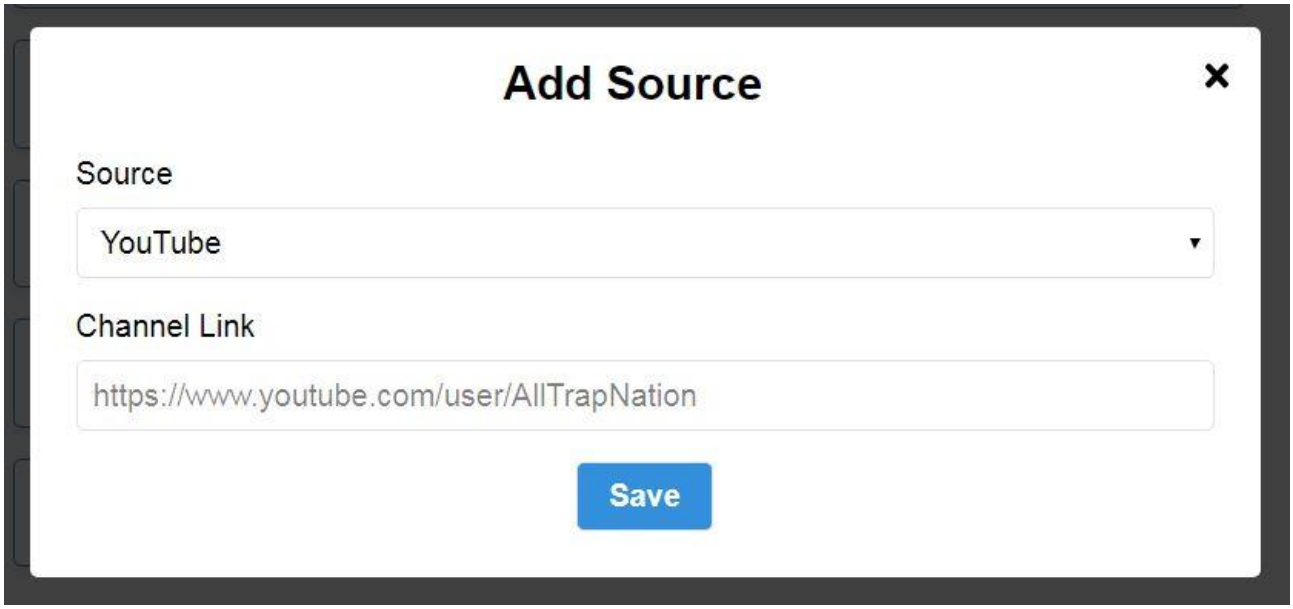


Figure 10: Adding a YouTube source

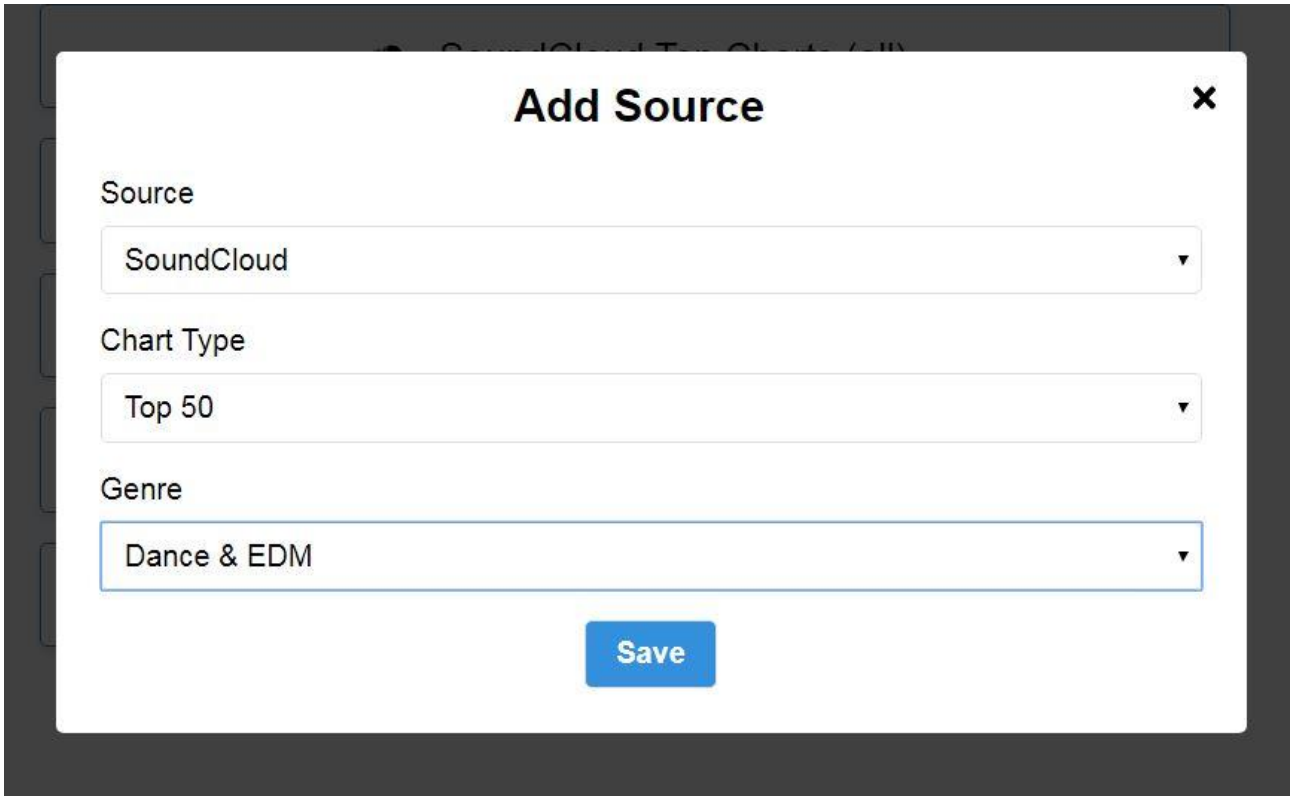


Figure 11: Adding a SoundCloud source

6.2 User Interface (Mobile Devices)

Figures 12 - 16: Mobile Based User Interfaces displays screenshots of the user interface on a smartphone, tablet, or other mobile device. Starting with the homepage, then moving on to songs, playlists, and sources. The final two screenshots show what adding a source looks like.

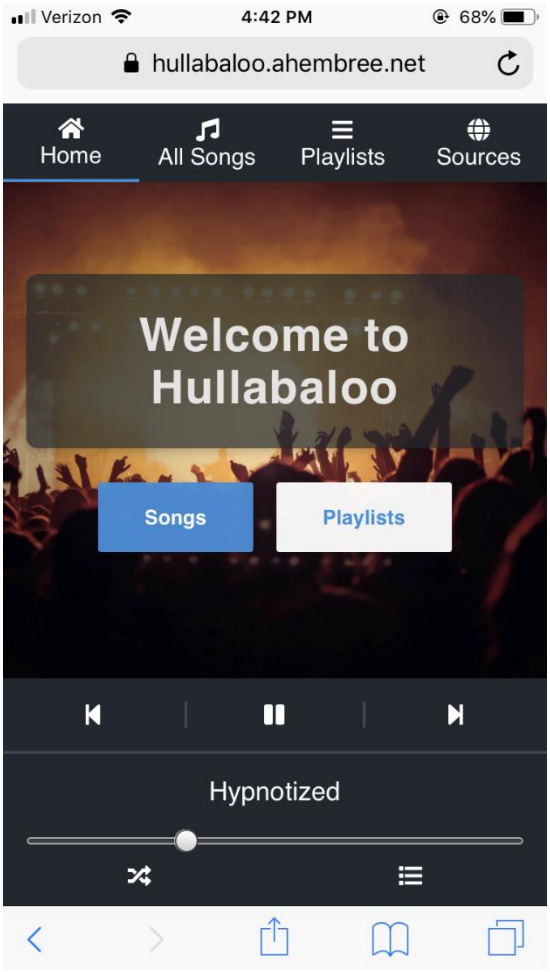


Figure 12: Homepage

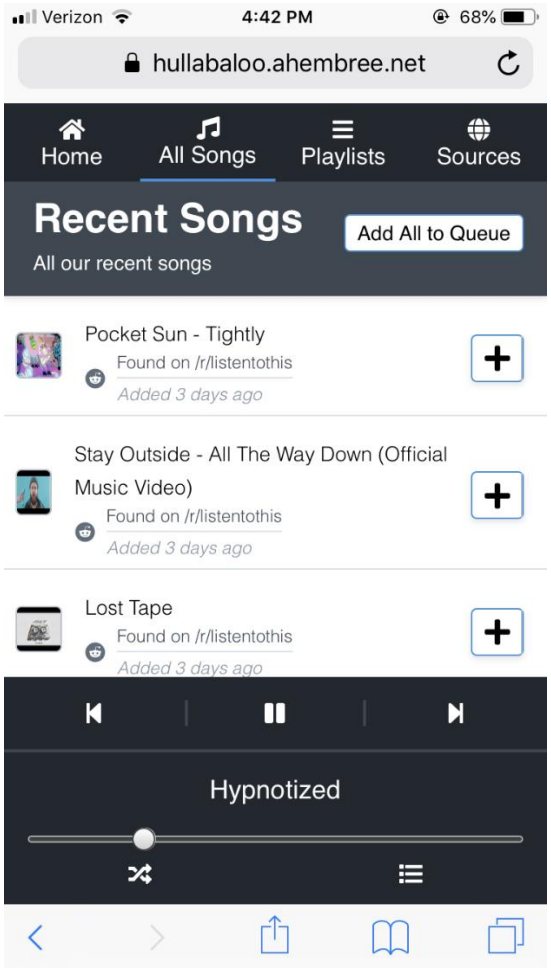


Figure 13: Song list page

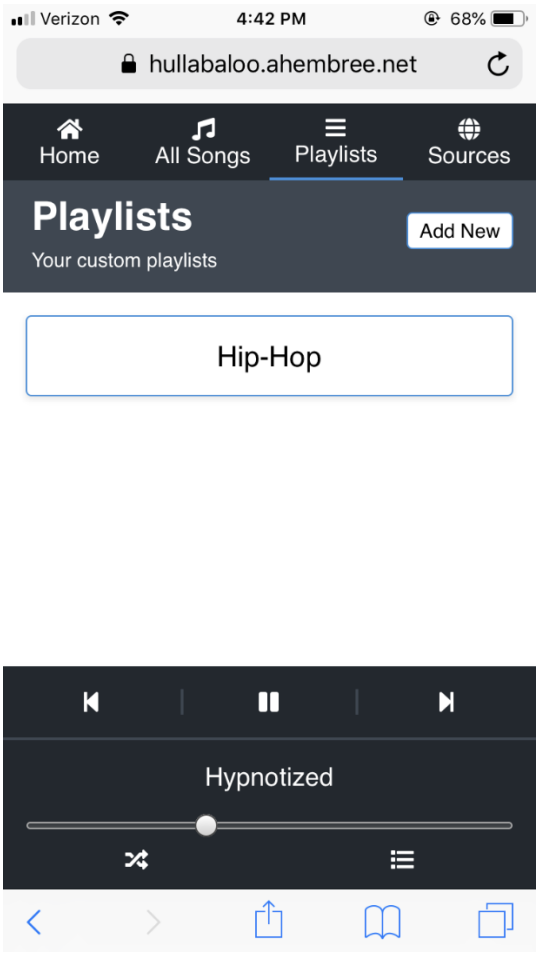


Figure 14: Playlists page

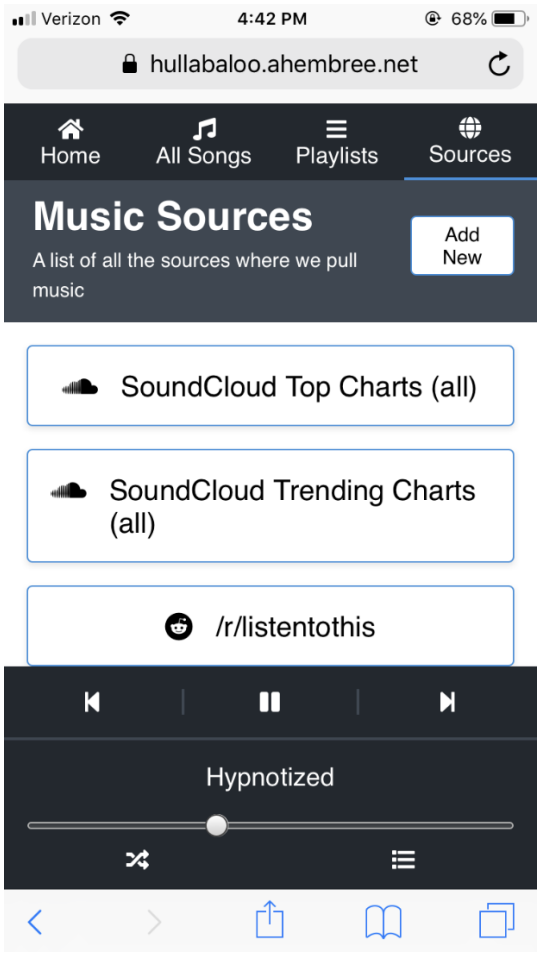


Figure 15: Sources page

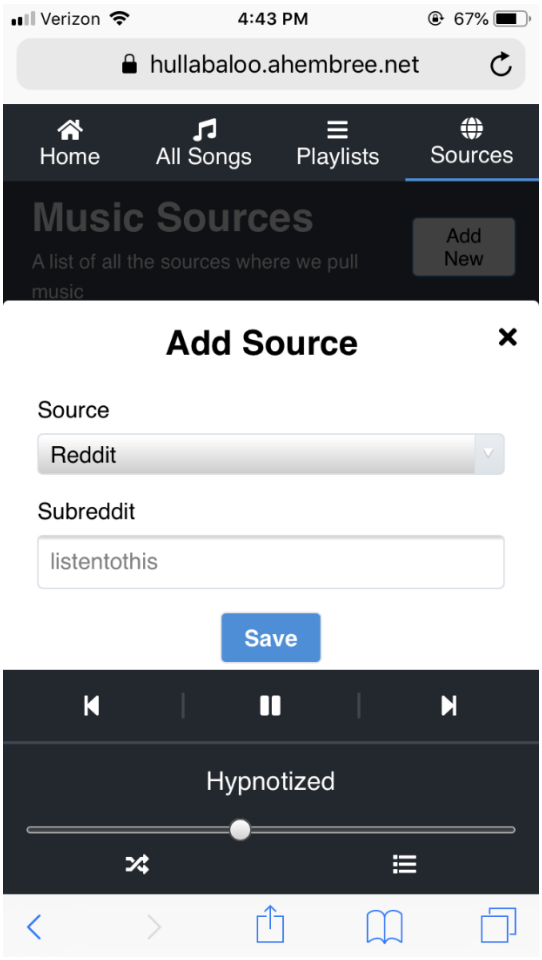


Figure 16: Adding a Reddit source on mobile device

7. CONCLUSION

7.1 Recommendations

When our group looks back at the past two semesters we see a roller coaster. There were some ups and downs, but overall we enjoyed creating Hullabaloo. If we had to do this all over again we would start earlier - with everything. Research software and the idea earlier so we can get to development earlier. Begin assignments and presentation preparation earlier. Basically, our time management was up to par, but only just. A recommendation for anyone starting a project would be to lay out a project plan with dates and deadlines. Then stick to it so development moves along smoothly.

We would like to continue development on Hullabaloo if we had more time. During the early phases of development, we decided to push the machine learning algorithm of Hullabaloo out of scope to focus on creating a perfect web app. We wanted Hullabaloo to track a users' listening habits to suggest similar music that they would enjoy. With more time, Hullabaloo could begin creating the machine learning feature which takes a lot of calculations and time to implement. Artificial intelligence (AI) or machine learning is still a young field that takes even the best developers months to create so it's no surprise we pushed it out of scope.

Finally, we have no future plans for Hullabaloo as of now. Everyone in the group is moving on with beginning their Information Technology careers after graduation. Hullabaloo is a complete web app, but there is always room for improvement when it comes to technology. If we wanted to move forward with the project we would need an income to keep development going. It is recommended that anyone working on a project should have a plan for the future after finishing development.

7.2 Fall Semester 2018

During the fall semester, we began our research phase and looked into how a music scraper like Hullabaloo would work. We decided to use Laravel as the backbone software as it's a powerful tool to link a backend, frontend, and UI or html webpage altogether. Another step was to determine how to store music data on a server for our application to interact with. The solution was PostgreSQL which is a simple, but effective database tool. We installed the database on a server and could begin coding. For the UI we wanted to keep it clean and easy to use on mobile devices so we looked at a bunch of layouts before picking one right for use. Vue JS and Tailwind CSS were the development programs that gave us a lot of control over how our UI was displayed. The API was developed and with the UI integrated, we had a webpage. Development continues, but we have a simple, working version of the webpage using the API to access the music database in order to play a song. Finally, our group had a presentation in front of the class before winter break. We were able to discuss our concept, vision, and a prototype demo. In the spring semester, development continued with changes from the feedback we received from students.

7.3 Spring Semester 2019

In the spring, development continued as we only had a prototype of Hullabaloo in the fall. The web app began taking shape as we upgraded the layout of the homepage. The song list was more or less complete, but we prettied up the layout as well. We created the sources page so users could add their own source of new music. This presented some problems as it broke our song scraping API for a bit, but through bug testing we figured it out. We learned a couple lessons that it's necessary to test a feature before moving on to another one and if everything starts breaking, don't be afraid to roll back the code. The final piece of coding to finish was the audio player so a user could play music and control how they listen to it. It was difficult to adjust the size of this audio player when the screen size changes from desktop to mobile. We discovered a frontend option that only took one line of code to fix this issue after discussing the problem as a team.

Our group advanced our coding skills while finishing up Hullabaloo and learned some new things about PHP and even basic frontend development. Time management skills are important when it comes to a project and we almost learned this the hard way by falling behind. But we didn't and project development moved forward. It's not all about coding though as there were presentations, posters, assignments, and journals to complete. This balance made the process more fun and gave us the full project planning life cycle. Every role in a project team is vital and keeps work on track. Our group learned lessons on project management and working together as a team which will be used frequently throughout our future careers.

At the end of spring semester we attended the University of Cincinnati's IT Expo with other senior design groups to present our projects to the public. This experience was actually fun and our group enjoyed talking to people about what we completed over the past school

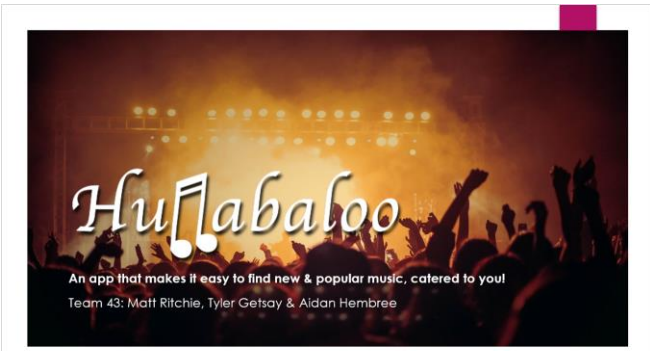
year. We arrived early in the morning to set up our booth and were able to see the size of the expo while waiting for it to open. For about four hours our group actively conversed with attendees about Hullabaloo and received input from a few of them. People agreed with our vision that a simple solution to find new and popular music was a good idea and liked seeing the web app. We learned that presenting a project to people from different walks of life or professional experience can be tough. A key lesson is to adapt your presentation or discussion to the audience and make it personal to them. The judging process caught our team by surprise so we were flustered at first, but it's easy to talk about a project that has taken two semesters to build. Finally, the last lesson learned from IT Expo is talk with everyone. Everyone has a background which makes them interesting and they offer something to learn from. Networking is a great skill to be worked on by anyone in any career.

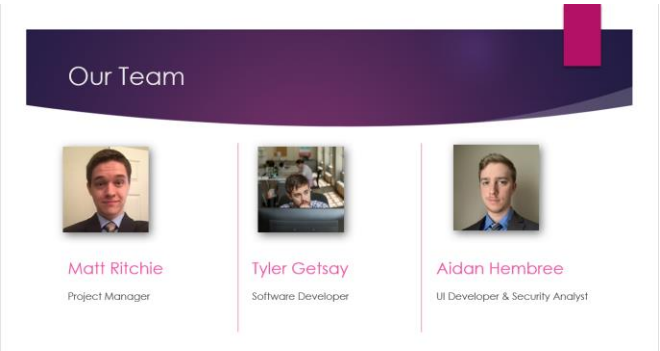
Overall this senior design project was an important course that teaches a student a lot about project development and prepares one for the real world. The skills and lessons learned will be taken into the future from everyone in the class. Additionally, working on Hullabaloo has been a great time for our group and we enjoyed it. Hullabaloo came to fruition because of this class and our group, so it's awesome to be a part of it. A wonderful project was created and a problem was solved. Remember that if you need to find new or popular music, then Hullabaloo is the site for you!

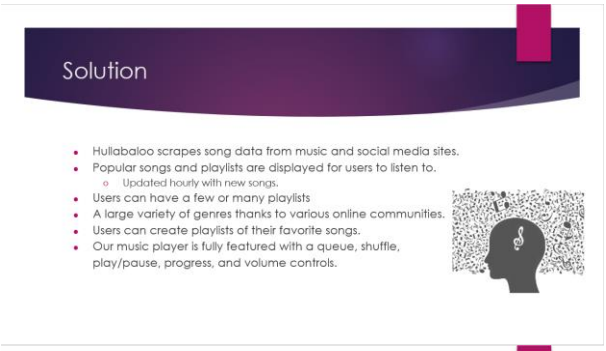
APPENDIX A. ADDITIONAL INFORMATION

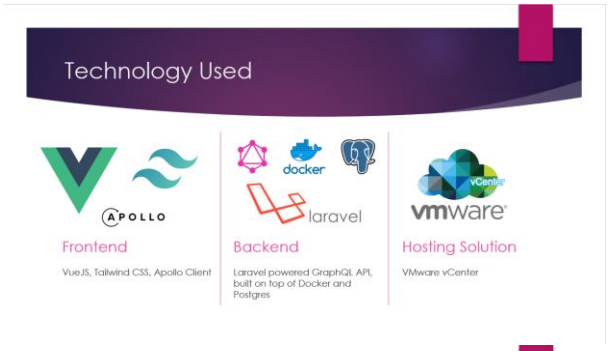
IT Expo “Hullabaloo” Poster:

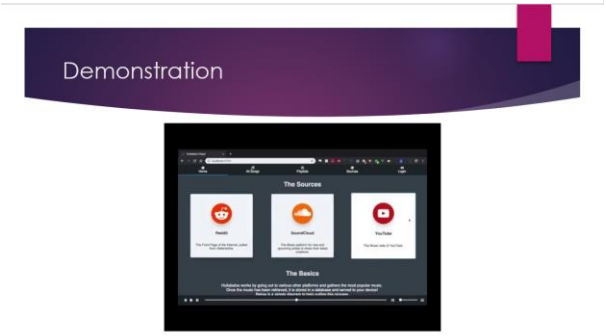
Presentation Slides:

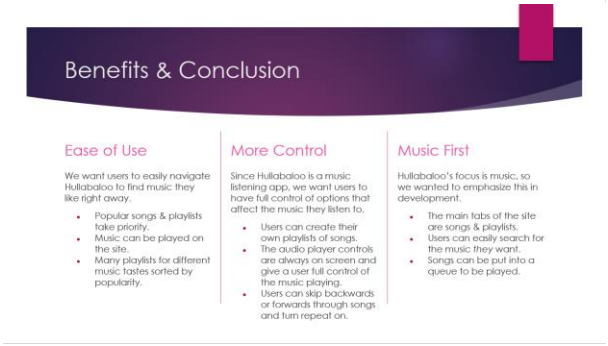












APPENDIX B. REFERENCES

[1] "VueJS," 2018. [Online]. Available: https://vuejs.org/.

[2] "Tailwind CSS," 2018. [Online]. Available: https://tailwindcss.com/.

[3] "Laravel," 2018. [Online]. Available: https://laravel.com/.

[4] "Docker," 2018. [Online]. Available: https://www.docker.com/.

[5] "PostgreSQL," 2018. [Online]. Available: https://www.postgresql.org/.

[7] "GraphQL," 2018. [Online]. Available: https://graphql.org/.