MuseNews Report

OVERVIEW

MuseNews is a source for news and information about today's top artists developed by Team 7. The site contains detailed information about a variety of songs and artists, while also allowing access to the latest music news. This report will detail the status of the project at the end of phase one.

Team Members

- Venkata Ravila, vkr339, venkataravila@gmail.com, GitHub Profile Link: vravila
- Sam Dauenbaugh, sad2929, s.dauenbaugh@gmail.com, GitHub Profile Link: sDauenbaugh
- Alexander Tekle, at36953, tekle.alexander@gmail.com, GitHub Profile Link: AlexanderTekle
- Daniel Walsh, dws956, danwalsh98@gmail.com, GitHub Profile Link: donuthole55
- Kedar Raman, kvr336, kedar.v.raman@gmail.com, GitHub Profile Link: kedaraman

Github Repo Link: https://github.com/vravila/MuseNews

Website Link: http://musenews.appspot.com (must be http)

Canvas Group: afternoon-7

Project Name: Muse-News

MOTIVATION AND USERS

We were motivated to create this application because we wanted to create a database for music modeled after the IMDB database. We felt like this would be beneficial to the world because, although there are many different sources of information on music artists and songs, they are rarely consolidated. We wanted to combine the artists and their songs with news about each of them to allow users to get up-to-date news about their favorite artists and songs. Artists communicate to their audience through social media applications like Twitter and Instagram, and there is constantly new news about them and their music. We wanted to combine this information into one application, so that users can quickly explore artists and their lives.
The target audience for this application is anyone interested in music. Whether someone is a passionate fan of a certain artist or a novice seeking out new music, our application allows people to explore artists and be up-to-date with their work.

REQUIREMENTS

The requirements for this project were obtained from our team's collective brainstorming as well as our customer team. Our customer team is Team 7.

User Stories

Customer User Stories “Team 7”:

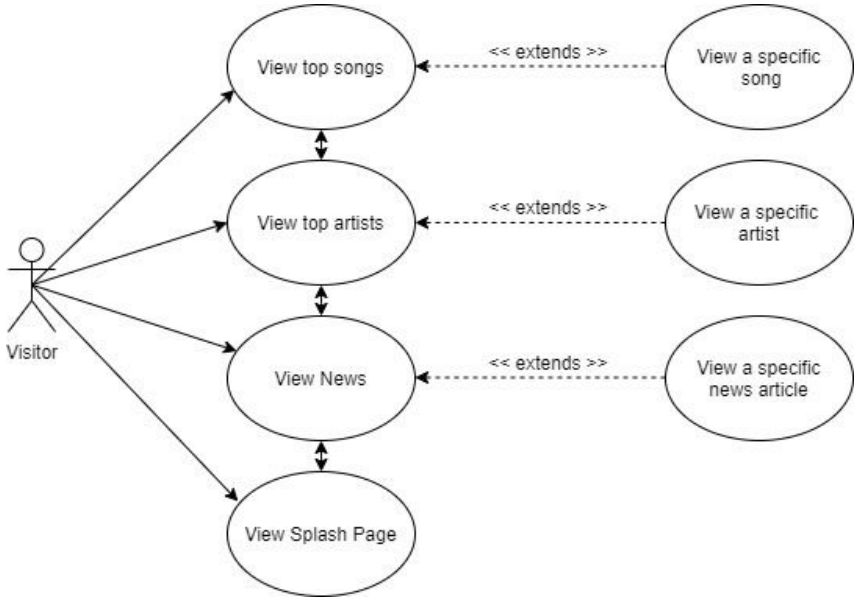
- As a local DJ, I want to know the top 3 songs on the charts, so that the music I play is up to speed with the current hits.
    - Estimated Time: 7 hours
    - Actual Time: 5 hours
- As a representative from a record-label company, I want to know news on an artist that is my client, so that I can ensure that favorable press is being released for my client.
    - Estimated Time: 5 hours
    - Actual Time: 5 hours
- As an avid music listener, I want to know the top songs by a certain artist, so that I can discover new artists that I like and can explore their best work.
    - Estimated Time: 10 hours
    - Actual Time: 8 hours
- As a new music listener, I want to know more about an artist's background, so that I can explore his or her past work and learn more about them.
    - Estimated Time: 5 hours
    - Actual Time: 9 hours
- As a listener, I want to access news articles and reviews about songs, so that I can see what critics are saying about songs.
    - Estimated Time: 8 hours
    - Actual Time: 9 hours

Our User Stories:

- As a listener, I want to be able to reference an artist's biography after seeing a news article about them, so that I can get more information about artists whose articles I found interesting.
    - Estimated Time: 1 hour
    - Actual Time: 1 hour

- As a huge fan of The Weeknd and Billie Eilish, I want to see if they are going on tour, so that I can go to their concerts when they are touring.
    - Estimated Time: 1 hour
    - Actual Time: 0.5 hours
- As a writer on Medium about artists, I want to display a concise preview of the article that highlights key topics on the Muse-News page, so that users are inclined to redirect to my page for the full article and contribute to my views.
    - Estimated Time: 2 hours
    - Actual Time: 1-2 hours
- As a listener who wants to explore new music, I want to quickly go between an artist and their songs, so that I can explore new artists and songs.
    - Estimated Time: 1 hour
    - Actual Time: 1 hour
- As a software developer, I want to see the tags for each artist and song and their respective listening and play-count, so that I can analyze the statistics and create data models for the most popular types of music and categories.
    - Estimated Time: 0.5 hours
    - Actual Time: 1 hour

Use Case Diagram



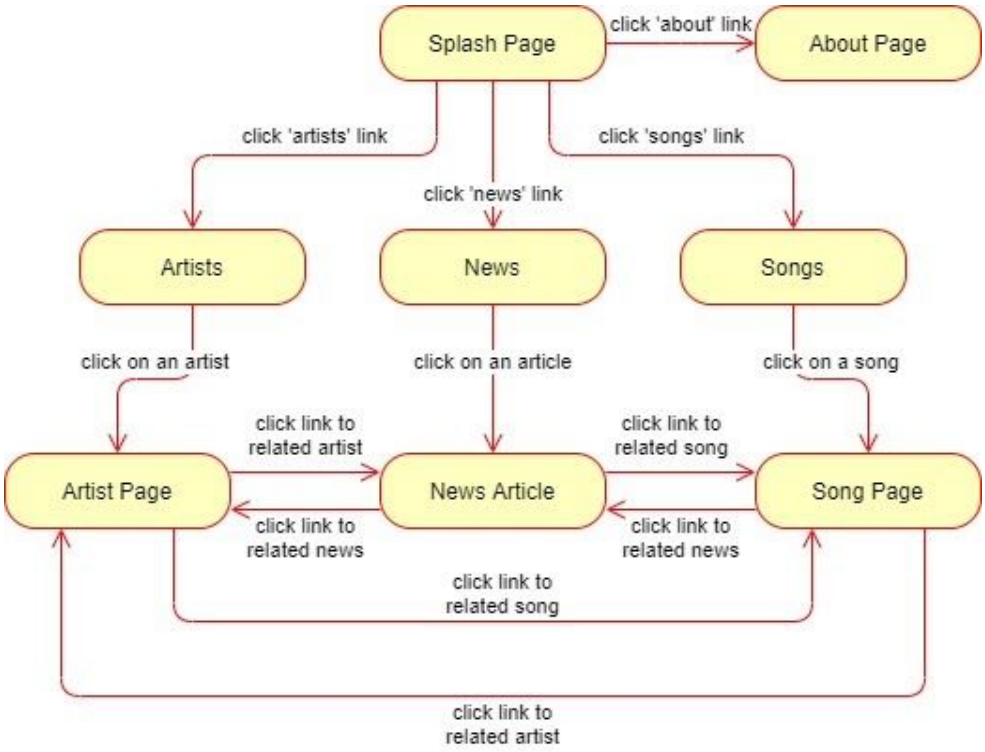
DESIGN

The website consists of three models (songs, artists, news), a splash page, an about page, and three instances of each model. This creates a total of fourteen pages that a visitor to the site can access. Each page features a navigation bar that links directly to home, about, news, songs, and artists. These links go to their respective pages or models. Each instance of a model follows the same blueprint as other instances of the same model, but has different data brought in from the API sources. There are links between models so that users can navigate between models that are related. For example, there are links between the song “Blinding Lights”, The Weeknd, and a news article about “Blinding Lights”, since they are all related.

To design the application, we started off by splitting it by model. We designed a home page that linked to a separate page for each model. Each model homepage was developed individually for functionality. Pages were added from each model homepage for each instance. Once each model had been developed, we added API calls to dynamically populate data fields and created the links between each model. We then standardized our formatting, created the About page, and ensured that everything worked well together.

State Diagram

The internal links embedded in the site's pages can be seen in our state diagram. The state diagram does not contain any transitions through the navbar or through external links.



TESTING

Phase 1 of this project was tested manually for bugs, as it was largely front-end development with some back-end API calls. To start the development process, pages were tested individually as they were developed. The elements of each page were tested to ensure that they displayed the right contents. As we started putting the pages together and linking them, we tested the links. Once we connected APIs to pages, we tested the data thoroughly to ensure that it was accurate and what we wanted. Formatting was done at the end with CSS and Bootstrap, which was also tested manually to ensure that the site looked appealing.

Frontend

To test the front-end elements of the site, the team manually clicked through all of the pages to verify that each page displayed as intended. Additionally, links between individual instances were tested to verify that the connections between models work.

Backend

The back-end work for phase one is limited to the use of some of the APIs. Code written for using the APIs was tested manually, but in the future will be tested through unit tests.

MODELS

The site has three models pertaining to songs, artists, and news. Each model is linked to other related instances from other models. For example, an artist page will link to the artist's songs and news about the artist.

Songs

The first model shows a song's title, album art, and artist gathered from the Spotify API. By clicking on either the song title or the artist name, the user is taken to the respective page. The individual song's page shows the same information as the song model, but only for the specific instance being viewed. In addition, the instance will have a link to a related news article. Data was also added describing the song, how many listens and plays it has, and its tags. This data was pulled from the LastFM API.

Artists

The second model details an artist by getting their biographical data, number of listens and plays, tags, similar artists, and whether they are on tour through the LastFM API. Each artist's name is shown on the landing page for the model, and each instance shows the biographical information taken from the LastFM API. Further information about artists can also be pulled from the Spotify API. In addition, each artist is linked to news and songs related to the artist in order to connect the models together.

News

The third model describes recent news related to the artists and songs contained in the other two models. This model uses the Google News API to draw the top news stories pertaining to each artist and song on the site. By clicking on the button provided on the screen for an article, the user is taken to a page showing more information about the article. This page shows the title, author, and preview text of an article and also provides a link to the source. In order to be more connected with the other models, the article instances also link to a related artist or song mentioned in the article.

SOFTWARE

The bulk of the website's code is written in JavaScript using Node.js and React.js. In addition to those frameworks, we used four different RESTful APIs to gather the information necessary for the models and their instances.

Frameworks

In order to make the website easier to design and build we decided to use the MERN model (MongoDB, Express.js, React.js, and Node.js). For this phase specifically, we used the Node.js and React.js frameworks. These frameworks helped aid in the development of the pages and allowed us to integrate the APIs and JavaScript code with the HTML elements displayed on screen.

React

React is a JavaScript library that assists with web development. It includes functions that make pages easier to manage and makes code more readable. It also allows object oriented approaches to problems through the use of Components. It also provides libraries for numerous elements, which was helpful with front-end development.

Node.js

Node.js is a JavaScript framework that helps with the back-end development of a website. The framework provides tools for building, compiling, testing, and running code. We used Node.js to create API calls to the Spotify API and the Google News API that will enhance our scalability in future phases.

APIs

Four different API's were used in making this site. Each API related directly to the information being presented in one of our models.

Spotify

The Spotify API allows us to access information about artists and songs. This API is used in the song model to get album art and song names associated with an artist.

Last.fm

The Last.fm API allows access to information about songs and artists. It is used in the songs model to get a description of the song, how many listens and plays it has, and related tags. It is used in the artists model to get a biography of the author, how many listens and plays they have, tags, similar artists, and whether they are on tour.

Google News

The Google News API allows access to hundreds of news articles from various sources across the internet. This API is searchable by a number or parameters which is used in the news model to get relevant news articles and to link directly to the source of the article.

Github

The Github API allows access to statistics about the development of the project. This is used in the about page of the site to show the contributions of each developer on the team in terms of commits and issues.

Tools

In order to aid the development of the site, our team used a variety of frontend and backend tools.

Bootstrap

Bootstrap allows for the creation of user-friendly interfaces by providing style sheets and JavaScript components. Bootstrap is used throughout the site to create a consistent look and feel. Components provided through bootstrap help create simple GUIs for the models and the individual instances.

REFLECTION

Looking back, our team did a great job with dividing the work up, completing our tasks, and integrating the parts back together. The majority of us were not very familiar with frameworks like React.js and Node.js, meaning there was a steep learning curve. We did a great job of learning these technologies quickly and applying them to our application. In the future, we should start the phase's requirements earlier, so that we have more time to ensure that everything works well together. We should also spend more time planning, as we had to come together after we designed our model pages to decide on a uniform style. We also could have done a better job using Github. We had a couple of mistakes in the development process, and we could have made better use of branches, although it was not necessary as much of our individual work was isolated.