Group #10

Band Together

Technical Report

By: Abhi, Adam, Faezah, Gavin, and Jason

Table of Contents

Overview

1

1.1 Project Purpose

This website is created with the purpose of enabling users to find concerts in any location from artists and bands they enjoy. It is hosted on the domain https://bandtogether.events.

1.2 Models

The information on Band Together is divided into three main models. Artists, Concerts, and Locations. The models are defined and related to each other in the following ways. Location pages define places where concerts and artists will be, additionally, the page gives a short description of the location and allows the user to visit the page for that location, as well as visit the artist and concert pages relevant to the location. Artist pages allow users to find artists they want and track where and when they will be performing. The Artist pages also display information and allow them to follow the links for the locations and concerts they will be performing at. Finally, Concert pages show the user what concerts are coming up and displays the artist, city, and venue. From the Concert pages, the user can view information on the concert and the artist.

1.3 Technical Specs

The front-end of the site is done mainly through ReactJS. Routes control the paths for the site and allow you to simply input the pathname to link to any of the pages. There are some dependencies imported into the React App, you can download them once you have the file and are in the band-together directory by running npm install with node. This should download everything you need to run the app locally using `npm run dev`. To import any new components or tools that you wish to use, create the import statement and use `npm install <input>`, this will update the requirements and allow anyone to get it once they run `npm install` again.

1.4 File Structure

From the website repository, the initial folder contains a `README` with some information if you want to get started faster, but here we will be going more in-depth. You can see there are three folders and three package files. The folders in no particular order are `dist`, `src`, and `node_modules`. For the most part, you can ignore the `node_modules` folder as it is only to hold build tools that are installed from `npm`. The `dist` folder's files can also be mostly ignored.

What is important about `dist` is that it holds the resources used on the website, such as the splash art or the front page image. The `src` folder is the main code folder, it holds two folders, `layout`, and `pages`. The `layout` folder holds the JS code responsible for the main site-wide navbar in `navbar.js`, the code for an image slideshow in `Slide.js`, and the code for special element boxes in the `Card.js` set of files. More in-depth guides on these files will be covered later on in this report.

1.5 Resources

All image resources for Band Together are stored in the `images` folder, which itself is contained in the `dist` folder. All images or other media that are not loaded dynamically from some other website can be found here.

Front End

2

2.1 Amazon Web Services (AWS)

The website is currently being hosted by Amazon Web Services. The site is stored in an S3 bucket and distributed using CloudFront.

2.2 GitLab

Gitlab is the source for version control within the app. The Front-End repo is at https://gitlab.com/Adam-Bomb/band-together

2.3 App.js

The App.js file contains the Switch for our React Router. Inside the Switch are all of the different pages and their respective urls.

2.4 index.js

The index.js file renders the app. It uses a React Router so that we only have to host the index.html and bundle.js files in our S3 buckets.

2.5 App.test.js

The App.test.js file runs the frontend tests using the Mocha framework and Chai package. The tests ensure the frontend functionality such as which pages get rendered and what goes on them works as expected. The Chai package helps with testing connections.

2.6 guiTesting.py

The guiTesting.py is a script you can run to test the gui of the site, it checks that the links work correctly across the site and that the pages load correctly. You can make changes by downloading selenium and updating the path for the webdriver on your machine.

2.7 Searching, sorting, filtering

Each list page has a GUI to customize sorting, ordering, and filtering options as well as a search bar. The sorting menu lets you choose one active sort by option at a time, order the

results by ascending or descending, and the filters allow you to use a range slider to adjust min and max values for certain attributes. Searching for a term will highlight the attribute that it matched with.

2.8 Comparing

Each instance on the list page has a comparing button to toggle selecting it. When at least two instances are marked to compare the list page's compare button becomes active. Clicking it toggles a compare view on and off which replaces all the visible instances with only the ones marked to be compared.

Back End

3

3.1 Amazon Web Services (AWS)

The backend server is hosted on an Amazon EC2 Instance at 52.202.44.47. We bought a domain for it as well, bandtogetherapi.xyz. The database is being hosted on an AWS RDS that Django connects to.

3.2 GitLab

Gitlab is the source for version control for the backend of the app. The link to the backend GitLab is here https://gitlab.com/Adam-Bomb/bandtogetherapi

3.3 Django & Django REST

The RESTful API is being handled by Django's REST Framework, and is the core process behind the code serving our API data. It also defines the models (Artist, Concert, and Location as our main models, with sub-models of Venue, and Album), and the views for each of the models in their own python files.

3.4 Docker & Apache

The API pages are being hosted and served to the internet through Apache which is being run inside a Docker container running on the EC2 instance. This Docker container is running Ubuntu (as opposed to the CentOS run outside the container), and its own separate installation of

Python 3.7 with a few different libraries and dependencies installed. But besides those, the files and code inside the container are exactly the same as the files outside the container.

3.5 API Format

The API has list and detail endpoints for each of the three main models as well as a detail endpoint for a 4th venue model (necessary for one to many relationships). Each list endpoint returns a paginated response containing all instances of that model. Each detail endpoint uses the id passed in the url to grab a certain instance of that model.

3.6 Postgres

The database for the server is running Postgres version 11 and is hosted at band-together-db.cp7swib6datx.us-east-1.rds.amazonaws.com. The database was originally done in SQLite as SQLite is what Django defaults to, but eventually we migrated the database to Postgres.

3.7 Testing

In order to test the backend code, we used Python's unittest testing framework. Within these tests, we ensured that the data was being populated correctly, accessed correctly and connected to each other correctly. These tests can be found in the backend repository in restapi/tests.py

3.8 Searching, sorting, filtering

We implemented a search bar on the Splash page that returned results gathered across all models. On each model page, in addition to the search bar, there are options to filter the results based on Elevation and Population for the Location model, Popularity and Followers for the Artist model and Ticket Price for the Concert model. There are also options to sort by Region, Elevation, Population and Timezone for the Location model, Genre, Followers and Popularity for the Artist model and Venue, Location, Artist and Ticket price for the Concert model.

To implement searching in the backend we altered the list endpoints to accept parameters that allow a user to sort by a variety of attributes, filter by numerical attributes, and search by keyword matching. The parameters are required but query can be left blank. With this design, a user is able to perform a search and then filter on the results of this search.

List Pages

4

4.1 Overview

All ‘List’ pages are accessible from the `src/components/pages` folder. Each file is saved as `<model>ListPage.js` where `<model>` is the model the page is for. The ‘List’ pages are pages directly accessible from the site’s navbar. These pages are the places where users can browse all existing instances of each model. Clicking on an instance will take you to its related Instance page. All of these files are currently stored in the `src` directory.

4.2 Concerts

The list for concerts uses instances of `ConcertCard.js` to display its information. The main info listed is the concert name (often just the name of the artist/band), the city the concert is taking place in, and the venue of the concert. There are additional links in the concert’s card that link the user to either the concert’s instance page, the artist’s instance page, or the location’s instance page. The time and date of the concert are also displayed on the card. For more information on ConcertCard.js, see *Section 5.3* of this document.

4.3 Artists

The list for artists uses instances of `ArtistCard.js` to display its information. The main information listed is the name and a picture of the artist, the genre of music, year started, their record label, hometown and their most popular song. Additionally, there are links to upcoming concerts and their respective locations. For more information on `ArtistCard.js`, see *Section 5.3* of this document.

4.4 Locations

The list for locations uses instances of `LocationCard.js` to display its information. The main information shown is an image of the location, the name of the city which also serves as a link to that city’s page. Additionally, there is information given on the population size of the location, major concert venues, the main airport of the location, and the per capita crime rate of the location. For more information on `LocationCard.js`, see *Section 5.3* of this document.

Instance Pages

5

5.1 Overview

The instance pages are where the bulk of the information is displayed. They are what the list pages point to when a user wants to see the individual item in the list. The instance pages are generated by inputting information to their `<model>DetailPage.js` where `<model>` is the model the instance page is for. The information is input in the corresponding list for that detail page and is generated based on what item the user has selected.

5.2 Concerts

Concert instance pages will display the following information in a table.

- Style of music
- Artist
- Venue
- Average ticket price
- Number of seats available (Capacity)
- Address
- Starting Time
- Number of tickets left
- Noise level
- Whether the venue is indoors or outdoors
- Whether alcohol is sold on the premises
- The weather forecast for the day of the concert

5.3 Artists

Artist instance pages will display the following information.

- Venues they will be playing at
- Cities they will be touring
- Year they started
- Music Genre
- Albums
- Top played songs
- Members if they are a band
- A short bio about the artist
- Related Artists
- Record label

5.4 Locations

Location instance pages will display the following information.

- City
- Country
- Different venues in the city
- Population
- Crime/safety rating
- Food Availability
- Nearby hotels
- Airports
- Public Transport Availability
- Weather forecast
- Map of the location

Miscellaneous Files

6

6.1 Navbar.js

This file is located in the `src/components/layout` folder. The navbar allows the user to traverse to any of the main pages at any point. Clicking the icon in the left corner returns the user to the main page. The navbar is able to be seen on every single page of the website with no exceptions. The code itself is relatively straight-forward, as it is simply a rendered set of links passed into `<a>` tags, with an image from our `images` folder being displayed as the logo.

6.2 AboutPage.js

This file is located in the `src/components/pages` folder and contains information about the developers including an image, bio, number of commits to the GitLab and the number of issues they have solved. It dynamically pulls this information from the GitLab API. More information on using the GitLab API can be found here: https://docs.gitlab.com/ee/api/ and will not be covered in this document.

6.3 Cards

As mentioned earlier in this document, we have a set of `<page>Card.js` files in the `src/components/layout` folder where `<page>` is the page (like the 'About' page) utilizing the Card displays. These are simple, compact containers for displaying information and the only major difference between each file are the React props they are expecting.

Visualizations

7

7.1 About the Visualizations

We used the D3 JS Library to create visualizations for both our website and our Developer's website. We pulled the data from our API and our developer's API using a Python script 'GenVisData.py' in the '/components/extras' folder. Also in the 'extras' folder is all the code and .json datafiles the visualizations are using to display the graphs. We have added both sets of visualizations to a page titled Visualizations.js, found with the other components.

7.2 Visualizations for Band Together

The first visualization we did using our data was to count up the average ticket prices of all the concerts (rounded to every $10) and graph those results in a scatter plot. In the graph you can clearly see that most concerts are priced on average below $200. This is likely because most artists in our database are not going to be the uber-popular artists that sell out stadiums with ticket prices that hit the thousands, most artists perform in smaller venues, like bars or small theaters, and therefore cannot charge the extreme prices some other artists can.

The next visualization we did was a pie chart of the most common genres artists labelled themselves as. Since the data for genres depended on what the artist's themselves put on Spotify, the results had a lot of very specific or niche genres listed with only one or two artists, so to keep the data clear, any genre with less than 6 artists was lumped into a 'Misc' category, which ended up totalling to 200 different artists, which was about a third of all the artists, far exceeding the numbers of any other genre.

The final visualization we did for our data, was to count up the number of concerts being held by state and sorting the results in descending order. This resulted somewhat unsurprisingly with Texas and California at the top of the list with over 250 concerts each, all the way down to Missouri that our data found was only holding one concert.

7.3 Visualizations for our Developer (Park Protection)

Our Developer was Park Protection, which related 3 models Parks to Plant and Animal Species. This first visualization is a pie chart showing the proportion of Endangered, Threatened and Recovering Animal species found in the United States. From the chart you can clearly see that a majority of animal species from the data are endangered.

The second visualization we have displays the number of endangered plant species by state. This shows that California has the most endangered species by far, which might indicate that California has the most data available out of all of the states.

The last visualization is a stacked bar chart that shows what proportion of each plant type is Endangered, Threatened, or Recovering. This chart emphasizes just how few species are actually recovering and how many are endangered. You can also see that although there are many different species of Forb/Herb plants, many are endangered.