Technical Report

Jonathan Peacher | Tika Lestaris | Shida Shen | Sarah Baxter | Mitch Chaiet

Github: https://github.com/djpeacher/IDB

Motivation:

Our team was motivated to create an IMDb-like application for everything you need to know about artists and music. We were inspired by Spotify and Wikipedia when we came to the realization that there needs to be a central place where people can find simple and quick information about an artist's biography, discography, and tour dates. Our website is named "BandDB", which stands for Band Database. We believe this website will be useful for music lovers and others alike who want to discover and learn a little bit more about their favorite bands and artists.

Before we got started on the project, we used PlanITPoker to define and estimate 10 user stories for features that we will have in this first phase. This special site provides an interface that allows us to create user stories and anonymously vote on the complexity or estimated time of completion for each user story or feature. After we voted on each story, we were then able to discuss our votes and see the average estimate for each user story. The estimations are on a scale from 1 to 10, which represents the amount of time we estimated we would spend on each user story or feature. 1 represents 10 minutes, 2 represents 20 minutes, 3 represents 30 minutes, and so on.

For the first phase, we aimed to create simple features on our website such as viewing specific information about an artist. For example, one user story we created on PlanITPoker was titled "A user wants to see an artist's age." For this case, we simply wanted information about the artist's age readily available in the page that contains other rich content about the artist, such as their birthday, role, and origin. The user could easily navigate to the "Artists" page and find an artist that they like, and the age will be on that page. Since we currently only have a static website and have hard-coded data on 3 instances of each model, we only have information about 3 artists to test our use cases on. In the future, we plan to create a dynamic website with many instances of each model and navigation features such as search, filtering, pagination, and sorting.

We were also motivated to include a feature that allows users to collaborate and add or edit information on the website. This will be done.

Model Description

This will be an interactive website for information about music bands. Users will be able to browse information in the database as well as add and edit existing information. Our goal is to create a platform where people can collaborate information and data about bands. We were inspired by the notion that there are plenty of undiscovered artists out there, and our website will eventually be the go-to place for people to discover new artists and inform their friends and family about new uprising artists. Our models are designed such that music information is categorized into five models: Bands, People, Albums, Tours, and Songs. These models contain simple text information and any model can be visited directly in the easily accessible navigation bar of our website. Each model contains at least 5 attributes, which are pieces of information that are related to the model that users might be interested in knowing. These attributes are:

- Band - name, genre, years active, people, albums, tours
- People - names, birthday, role, origin, bands, albums, tours
- Album - title, year, US Chart Position, label name, songs, bands
- Tour - venues, locations, dates, lineup, songs, people, bands
- Song - name, creation date, run time, US Chart Position, genre, bands, albums

These models are related to one another and may contain content that may lead you to content in a different model. One goal of banddb.me is to allow users to easily access these different categories of information on a single website. For this reason, each model links to at least two other models. The relationships between these models are:

- Band - links to people, albums, tours
- People - links to bands, albums
- Album - links to songs, bands
- Tour - links to songs, albums, bands, people
- Song - links to bands, albums

Each model will have its own page where users can view the different instances of the model and sort and filter the data. In addition, our website will include a homepage, which will feature a carousel of links to related pages, and an about page, which will include information about the creators of the website and the project itself. Each page will also include clickable elements and multimedia, which will be dynamically generated by Flask.

Use Cases

Banddb.me is a great place to learn more information about your favorite bands. Here are use cases we want to tell about BandDB.me.

A potential user of banddb.me might be someone who really likes a specific band and wants to learn more about them. They can go onto banddb.me and see how long the band has been active and see all of their

albums and songs. They can see all of the members of that band and visit those members' pages to learn more about them, such as where they are from, how old they are, and what their role is in the band.

One potential user of banddb.me might be someone who just heard a great song on the radio by a band they don't know. She wants to learn more about this band and their music. She can go to banddb.me and search for the song in our database. From there, she can see information about the band, each of its members, other popular songs from the band that she might like, and even see if the band will be touring in her city in the near future.

Another user might have just been to a Taylor Swift concert and really liked the band that opened for her. Unfortunately, the can't remember the name of the band. They can go to banddb.me and see the lineup for Taylor Swift's tour. From there, they can see the name of the band that opened for her and discover more information about the band, including their most popular songs. They can get all of this information without having to visit several websites or doing multiple google searches.

A user heard a song from the 1980s. She then grows a great interest in music from the 1980s. So she goes to BandDB.me to find out what bands were active in the 1980s in America. She clicks the tab Artists and uses the filter to select bands active in the 1980s. She might also be interested in using a sort button to sort those bands by US chart positions.

A user is a great fan of a band called X. X happens to be not so mainstream. It has not been listed in BandDB.me yet. Then she goes ahead to BandDB.me to add band X. Other users may then know about band X.

A user finds out that information about band Y is old. She goes to BandDB.me to edit the information and makes it up-to-date. She may also want to share the page to her Facebook. She clicks a button and shares the link to Facebook.

RESTful API

Apiary is used to document APIs to be implemented for our website BandDB.me. There are four major endpoints for this website: /artists/, /albums/, /tours/, /songs/. Each for displaying artists, albums, tours, and songs.

Without any specific URL or query string appended to them, all contents will be listed in response HTML; if id or query strings are provided, then specific content will be displayed. For example, domain/artists/ display all artists in the database, while /artists/idoftaylor_swift/ displays information about singer Taylor Swift. Query strings will be further used to display detailed information about Taylor Swift, like age, origin, genre, etc.

General format will be /artist/{id}{?query1,query2}. All endpoints can be visited by HTTP GET method as well HTTP POST method. POST will allow the user to edit information of the content, input new content (artists or

albums they know, etc) or maybe simply make a vote (like user’s favorite songs etc.).

Tools

Hosting

- AWS EC2 - Elastic Compute Cloud is a web service that provides computing capacity from Amazon's cloud.
- Namecheap - the domain name registrar from which we obtained our banddb.me domain.

API

- Apiary - an API documentation tool that enables quick API design. It generates an API blueprint that can be used as a contract between businessmen and developers, an agreement between front-end and back-end developers or a roadmap for database design.

GUI

- Bootstrap - a free and open-source front-end web framework for designing websites and web applications. It contains HTML- and CSS-based design templates as well as optional JavaScript extensions. We used Bootstrap throughout our website to style and format our data.

Frontend

- ReactJS - an open-source JavaScript framework for building user interfaces. React allows web pages that use data to change over time without reloading the page and for real-time rendering of web pages without the need of several static web pages. We will use ReactJS in future phases of this project to create web pages that are dynamically rendered on the client side. This will allow users to filter and search data and will provide a more interactive, user-friendly experience.
- JavaScript - is a high-level, dynamic, untyped, interpreted run-time language. We used JavaScript with Bootstrap to design the front-end of our web application.
- CSS 3 - The 3rd version of CSS, a style sheet language used for describing the presentation of a document written in a markup language. We used this with Bootstrap to style our web application.
- HTML 5 - The 5th version of HTML, the standard markup language for creating web pages and web applications. We used this with Bootstrap for the front-end of our web application.

Backend

- Python Flask - a lightweight Python web framework. Flask uses Jinja2 as its templating system and

Werkzeug as its WSGI utilities. Flask has BSD license.

Database

- SQLAlchemy - an open source SQL toolkit and object/relational mapper (ORM) for the Python language. We will use SQLAlchemy in the future phases of this project.
- PostgreSQL - an object-relational database. We will use PostgreSQL in the future phases of this project.
- yUML - In order to properly document the database that was implemented in this project, we used yUML to visually map the project as a whole.

Other

- Github - a web-based version control repository that our team used to maintain our source code.
- Slack - A communication tool our team used to collaborate over the course of the project. Slack is also connected to the Github repository, which allowed us to monitor the commits of individual team members and the overall progress of the team.
- Trello - A project management application that our team used to list and categorize the necessary tasks for developing our web application. It allowed us to track the progress of our team over the course of the project.
- PlanitPoker - an online application that provides an interface for Agile project teams to estimate the complexity or completion time of projects. Our team used this application to break each phase of our project into tasks and estimate the time each task would require.
- Box - In order to host this technical report we are required to host it on the University of Texas's file hosting service. The service essentially is the universities private DropBox.
- GitPitch - For presentation purposes, we will be using GitPitch to summarize this project as a whole once completed.
- Grammarly - This site is used to check that proper grammar was used in the creation of this technical report. The hope here is that this technical report will be readily and in a format that will be easy to understand.

Hosting

There are two main components to the hosting of this website; Namecheap and AWS.

Namecheap

Namecheap was used to obtain the domain name banddb.me. In order to link the AWS server with the domain name, we followed the instruction laid out in this article: https://u.osu.edu/walujo.1/2016/07/07/associate-namecheap-domain-to-amazon-ec2-instance/. Essentially, the article had us do two things after altering some

settings in AWS which is talked about below.

First, under Advanced DNS settings for the domain in Namecheap, we changed the A Record, `@`, to hold the value of the public IP of our AWS server, in this case, `52.15.240.176`.

Second, also under Advanced DNS settings for the domain in Namecheap, we changed the CNAME Record, `www`, to hold the value of the public DNS for our AWS server, in this case, `ec2-52-15-240-176.us-east-2.compute.amazonaws.com`.

Finally in order to allow AWS to let the domain, banddb.me link to the actual server we had to change a few settings there as well. Luckily that part was simple and all we had to do here was associate an Elastic IP with the actual server which we linked to on Namecheap.

AWS

AWS was used to host the actual server that will run python flask in the backend to generate the web pages one would see on the frontend. In order to accomplish this, we followed the setup process laid out in this presentation: https://www.cs.utexas.edu/users/downing/cs373/slides/SWE-RHMAP-Presentation.pdf. The presentation describes created a VPC, an EC2, and then created a Flask App on it.

For a VPC we simply created an account with AWS and followed the standard procedure to create a default VPC with a default endpoint.

For the EC2, we created a general purpose Ubuntu instance that allowed for auto-assigning of the public IP. Along with that, we made sure an HTTP security group was created to avoid HTTP conflicts later.

Once the instance was launched we were able to SSH in and alter the server in preparation for Flask. The main steps that were needed here were to install Apache and Flask done through a simple ‘apt-get’ command and then uploading a standard Hello World py file to test that Flask was up and running.