Big Binary Brand - TrackSpot

Project Overview

Our project vision remains very similar to our original goal. We would like to provide a destination in which users can review albums and songs much like Rotten Tomatoes and IMDB provide for movies. We want to be a location for which user and critic reviews can be aggregated into a holistic score for a given entity. We also want to provide functions which allow users to quickly find and easily find related, and trending entities which may be interesting.

Team Members

John Sweeney, Daniel Macdonald, Dongwon Park, Tommy Do, Aric Huang, Cameron Scigliano

GitHub Repository

https://github.com/jswny/trackspot

Design Overview

Our data model includes the following entities; artists, albums, songs, genres, reviews, users, and critics. For our critic model, we had it inherit from the user model, which contains all of the relevant common fields, since a critic is a user in our data model. Then, critic implements its own organization field which is specific to it.

For our URL routes, we have a home page, and then a detail view for each of the following entities; albums, artists, critics, songs, and users. These dispatch the appropriate ID field from the URL to the appropriate view (one for each entity). We do not consider it important to our prototype at the time to have list views besides for albums or genre views at all, as the home page and related sections from our other pages will serve those functions and navigation from those places will be more meaningful. Reviews are populated on their appropriate pages and don't require their own views either.

Problems and Successes

During the project we found that it was difficult to figure out how to use Django's ORM to query, filter, and sort using associated models. We also found it a bit difficult to work with dates in models. Next time, we will have certain members become more familiar with the ORM so that they can handle more difficult implementation topics. We also found that it was difficult to design a model to handle a single review entity which would be interpreted as either a song review or an album review. However, it was relatively easy to work with models inside of templates; getting proper absolute URLs, calling model functions, and even getting associated data from said models. Finally, we found Git to be very difficult to work with a large team. Conceptually not all members are on the same page, despite having done good work practicing and researching Git. Branching and updating branches from master, and vice versa are still difficult topics and often merges and history need to be manually fixed.

For our team, we found it was slightly difficult to coordinate project development from Slack sometimes, and that often physical meetings were needed to ensure everyone was on the same page and knew how to do what needed to be done. For our next project, we will try to meet in person more often which will make the process go more smoothly.

Write-Up: John Sweeney

Approximate contribution: 27%

I worked on setting up the common framework for the application so that others could easily implement their parts of the project. First I created the Django project itself and then I took the data model that we came up with and implemented it in the Django ORM with all the necessary associations. Next, I created a base generic view with all of the necessary blocks and parts that all of the pages should have in common, for example the Navbar. Then, copied over each person's page into a template compatible with the Django view framework I had set up and set all of the blocks as needed based on the existing views that our members had created from the last part of the project. Finally, I wrote basic view functions for the pages to be filled in by each member and then I created URL mappings in the project URLs file for all of the views.

For my own page (the home page), I filled out my view function with all of the necessary data and functions to get the data into my template. Then, I used the mock data added by Dongwon to test it out and make sure that the data was being displayed correctly. I also used my previous knowledge of Git and web programming to help others with problems when they encountered them, especially with Git which is quite hard to use because it is new to most members.

Finally, I did some of the team write-up and of course completed my own write-up.

Write-Up: Cameron Scigliano

Approximate contribution: 15%

After Joe set up the base templates for how each of our pages was, I worked and completed everything on the Song Page. This included completing the song part of the views and also on the song part of the html document working with the templates completely on my own. There were a couple trivial templates like adding the song name, artist name, genre, image, and song description that I had no problem with along with some harder ones that gave me trouble along the way. Specifically, the one that gave me the most trouble was finding out how to extend the Users to separate between the user reviews and the critic reviews, as we have those as two separate tables on the song page. Along the same lines once i had all the reviews separated into user and critic, it took me a little bit to find the average scores for each of them respectively. Another bump in the road was figuring out how to deal with the database when there were no reviews associated with a specific song. I just had to do an if else statement to display “no Reviews” and not try to calculate the average if this were the case.

The other thing that i did was add mock data to the songs part of the database. I added the descriptions and everything that went with that. Also, part way through we realized that we need an image_url for artist pictures and user profile pictures so i went into models and added those. And then i obviously completed my own individual write up.

Write-Up: Daniel Macdonald

Approximate contribution: 15%

Since we have six people on this team, a completely even distribution of work would equate to around 16% of the project. I think most of us were pretty close to hitting this benchmark in terms of our actual distribution. In terms of my individual contribution, I would estimate that I contributed roughly between 14-15% of the project. While these are estimates, I believe I contributed somewhere around the average amount of work while compared to my teammates. To detail the work I did on this project: Since each member worked on their individual pages as a way to distribute workload, I worked on the artist's page of the application. Joe created a generic base template using our previous HTML documents and the models from the relations we defined as a team. To complete the template, we all had to define our own view. This proved more difficult for me than I thought simply because of how complex the querying turned out to be for my relations. The difficult part of this mainly stemmed from the fact that I needed access to a specific album_id or song_id to query reviews for calculating scores. Once I figured out how to get the correct album and song it was much easier because others had already done the calculations. Despite the struggles, I was successfully able to complete the template and the view.

Write-Up: Dongwon Park

Approximate contribution: 15%

I mainly worked on “Album” page and the Mock Data. For the mock data, I worked on putting mock data in sqlite DB using Django admin, including 6 albums, 6 artists, 6 user/critic profiles, 14 songs, 11 reviews, etc. The DB is still growing up and I am creating a pull request whenever I add some data on DB so that every team members can share the most updated DB. I also added the ‘description’ field in Album class in models.py in the Django ORM.

For my own album section, besides the html that I created which display each album with album information and user/critic reviews, I created one additional html page which extends the base generic html so that it can display whole albums rather than an album as an entry. I added a url mapper, based on all the frames that Joe made for our team, in urls.py for the url ‘album/’ to the addition of existing ‘album/<int:pk>’. Then, I created all necessary variables and fulfilled the render function to display/render in my template.

In a team meeting, I suggested and drawed a UML diagram on a whiteboard with Aric to help team members understand our data model visually.

Write-Up: Tommy Do

Approximate contribution: 14%

I worked on developing the user profile page to be dynamically displayed. To do this, I needed to create a URL mapping which linked to a view class in the views.py file. The view class helped to populate a HTML template for the user profile page. The URL mapping for each page was set-up by John and then I mainly dealt with the views class and creating the HTML template. The views class was very straightforward as I used a generic DetailView class in which I just needed to specify the model the class was based on. The majority of work was figuring out how to populate the HTML template. It was easy to populate the HTML template with User data because the view class allowed me to easily reference data stored in the User class model. The difficult part was populating the template with other data not explicitly defined in the User class model. However, I ended up creating some functions within this User class to handle this.

Write-Up: Aric Huang

Approximate contribution: 12%

I worked on the critics page, from the UI to drawing data from the database to dynamically fill out the page. Populating the page was difficult since coordination our pages and adding to the data model and database was challenging. But with us populating the database with the required links and pictures that my page has, I was able to fill out my page well. Another challenge I overcame was learning to contextually add attributes into my page in the views.py file for django to correctly understand what data I want. It took a little of going back to the tutorials but I eventually figured it out.

The majority of the work I did was updating the views.py file to contain my pages attributes as well as updating my HTML to contain django to sql database calls so my page can pull the data from SQLite3. Also I created the data model diagram for our group and did my own write up for this part of the project.