The Jam

Summary

The Jam is a music-focused social network application that connects users through events using Eventbrite's API. The Jam application allows users to search for music events by keyword, comment on events, and manage bookmarked events through their profile. Once logged into the Jam platform, users can access their profile page to view a list and map of bookmarked events, a feature I built using the Google Maps API. Users can also connect with other Jam users via an emailing feature that users can access from the profile page of another user. This emailing feature uses SendGrid's emailing API to send emails directly through the Jam platform.

The Task

During my time at Hackbright's Software Engineering program for women, I was given the task of building an independent passion project which would solve some problem that I encountered in my day-to-day life. I took the task to heart and gave serious thought to issues that I regularly faced in my daily pursuits, and what sorts of problems I was interested in solving. Given that I am extremely passionate about attending live music events, I decided to concentrate my efforts on solving a void in the music event industry — the total lack of a comprehensive site to connect other people in the area who were also attending these events. The essential dilemma was that in order to attend a live music event in a new area, I had to jump around on numerous websites such as Resident Advisor, Eventbrite and Facebook to find live-music events in my area, buy tickets for these events, and connect with others in my area that were going to these events. Finding a show that interested me involved not only finding the right event and venue, but also finding the right social group. I regularly scoured Facebook to see which friends were going to the same music events in order to connect with a community of people to attend the event. This search for events also allowed me to find last-minute tickets to shows that were sold out online, that I otherwise would not have been able to attend.

This problem sparked the idea for an application that would allow users to search for music events in their area based on the genre of live-music that interested them. The app would contain a comment feature that would allow users to get more information about a specific event, connect with others in their area, and give users a profile that would store their upcoming bookmarked events, all in one seamless application. The need for a more unified integration of searching for events and connecting with others sparked the idea for the Jam.

The Plan

The first step in building the Jam was deciding how I was going to store the relevant data for the users in a database. Basically, I needed to design a relational database that would store information about the users of the Jam, details about the events returned from Eventbrite's API, bookmarks that users had pinned as "going" or "interested," comments for each event, and information about the users' respective past search history. I began at the whiteboard—mapping out a relational database with tables for each of these categories of data.

My next step in the planning phase was to decide what sorts of relationships I would need to define among tables in my database. For example, in order to preserve referential integrity, I decided to make a

"bookmark_types" table with two fields: bookmark_type_id and bookmark_type; that way, I could have a primary-key/foreign-key relationship between the bookmark_type_id in the "bookmarks" table and the bookmark_type_id in the "bookmark_types" table. Similarly, in my "comments" table, I define a foreign key relationship to my "users" table and to my "events" table so that I can make useful queries (such as getting all of the comments for a specific event) and have a way to connect the users to their comments.

After I planned my relational database, it was time to dive into the user flow of the app. Initially, I had to determine how each user would interact with the Jam, and I began jotting down my ideas using simple wireframes. I decided that the user should be able to search for events by keyword, date, and location straight from the homepage of the application. The user would then be redirected to an events search page, where the user could view thumbnails of all of the events matching the user's search criteria. From the events search page, the user could click on a specific event that he or she was interested in, and would be taken to an event details page which would have an extended description of the event, a list of all Jam users that had bookmarked the event as "going," and a commenting feature where the user could comment on events and scroll through a list of comments from other Jam users.

I also decided that the user should be able to click to another Jam user's profile from two places: i) from the list of other Jam users who had bookmarked the event as "going;" and ii) from the list of comments. Once on another user's profile, I wanted the user of the app to be able to access the emailing feature to email the user of that profile. Similarly, I decided that the current user should be able to see a list of bookmarked events on the specific user's profile that the user was reviewing. This feature would enhance the social aspect of the app and allow the user to discover similar events from other users' profiles with similar music tastes.

Execution

In order to build these features, I started by creating a flask app that I connected to my PostgreSQL database. I wrote my server and scripts that made requests to Eventbrite and Sendgrid's API in Python, and I made queries to my database using SQLAlchemy. For my front end, I used HTML and CSS to build my pages, and I used JavaScript, jQuery and Jinja to make my content more dynamic and handle user interactions within the Jam on the client side.

My initial MVP (minimum viable product) plan for the Jam included these features discussed above. I decided to break my progress up into two, two-week sprints. After week two, I would re-evaluate my progress and move on to my 2.0 features. I had some ideas in mind for my 2.0 features, but I gave myself the freedom to modify these ideas once the Jam took shape.

I finished my minimum viable product in about two and a half weeks and decided to work on two major features for my 2.0 version during the second sprint of the project that I felt would improve the overall purpose of the Jam. Once I built the search and social aspect of the app, I wanted to make the Jam more of a space that users could visit to keep their events organized, and I felt that adding a map to the profile page would serve that purpose and would provide a visual representation of where the venues of their upcoming events were located. I also took the time to think about the essence of the Jam and recalled that part of the inspiration behind the Jam dealt with event discovery—I felt that building an event recommendations feature based on users' past searches would add increased functionality to the Jam by recommending events to users that they may not have stumbled upon naturally.

In order to build the Google maps feature, I integrated Google Maps API into my project, and used longitude and latitude data saved in my database to create markers on the users' maps for each event that they had bookmarked as "going." I implemented the maps feature on the users' profile pages and dynamically generated these event markers based on the users' bookmarked events.

For my event recommendation feature, I had to restructure my database and add a table called "searches" where I would save a user's past searches. This allowed me to write a query to grab a user's past five searches, which I used to make a request to Eventbrite in order to return some recommendations to the user based on his or her musical interests. This feature was my biggest challenge in building the app because my script initially was making one API call to Eventbrite for each of the user's past 5 searches. Waiting for the five API calls to return to the server in order to load the HTML on the page took a long time, and in-turn, caused my homepage to load very slowly. I knew that I needed to optimize this feature, and I did so by making one batched API request to Eventbrite. I wrote a script that would batch the five requests into one request and return the batched results as JSON. I then wrote a multi-part function that first parses through the data and returns a list of suggested, personalized events and then compiles a list of these personalized events and returns a nicely formatted dictionary with the suggested event details.

My next challenge with this feature was taking the details for these suggested events and loading them on my homepage in a way that would not cause the HTML on the homepage to load slowly. Because the batched API call was still slower than expected, I decided to load the HTML on the homepage, and then asynchronously load the suggested events using Javascript once the document was ready. I did this by embedding a loading GIF into the event-recommendations div and then swapping out the GIF with the event recommendations once the request was received from Eventbrite.

Overall, the Jam app accomplished my goal of integrating the search and discovery of music events, while also allowing users to connect with other people attending the events. I enjoyed working on this passion project and look forward to creating other exciting projects in the future.