Capstone Project

4 May 2022

Grace Knudsen, Elizabeth Martens, Rachel Litscher,

Austin Lam, Brendan Lorn

# Final Product

Our hosted website is available at the following link: <https://ticx.discovery.cs.vt.edu/>

Video demo of the website: <https://www.youtube.com/watch?v=shxP8rrRUXM>

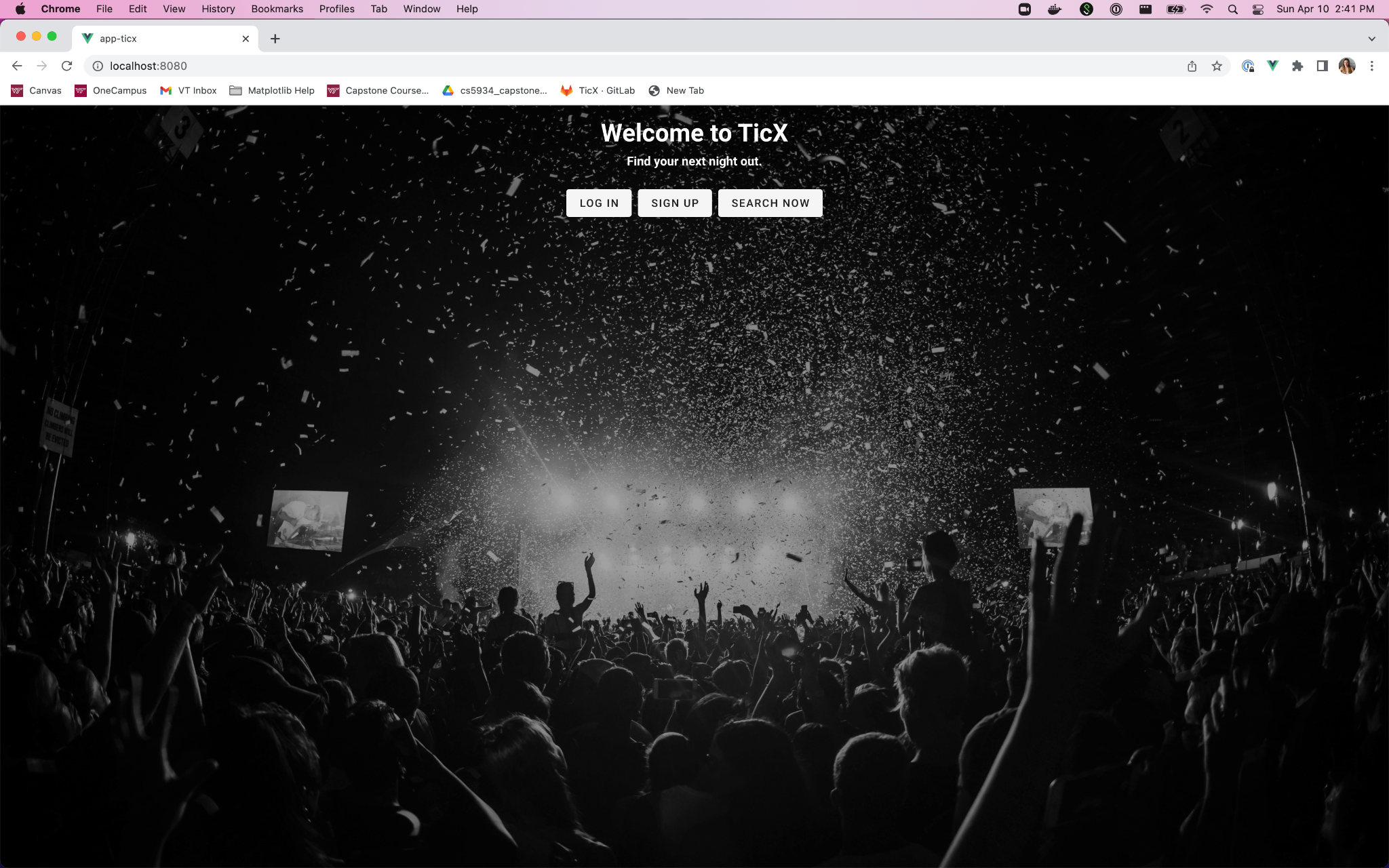
# Product Description

Ticx is a web application built for music lovers to discover new live music events near them, explore events by their favorite artists, and keep a history of all the events they have attended. With the use of multiple ticketing services, such as Stubhub and Ticketmaster, Ticx acts as a one stop shop for concert information. Not only does it present a detailed description of each event, the application catches the eye of a user by providing the ability to view multiple concert venues or events on an interactive map. This is great for those who would love to see different events happening around them or wherever they may be traveling to during a particular time frame.

# Product Functionalities

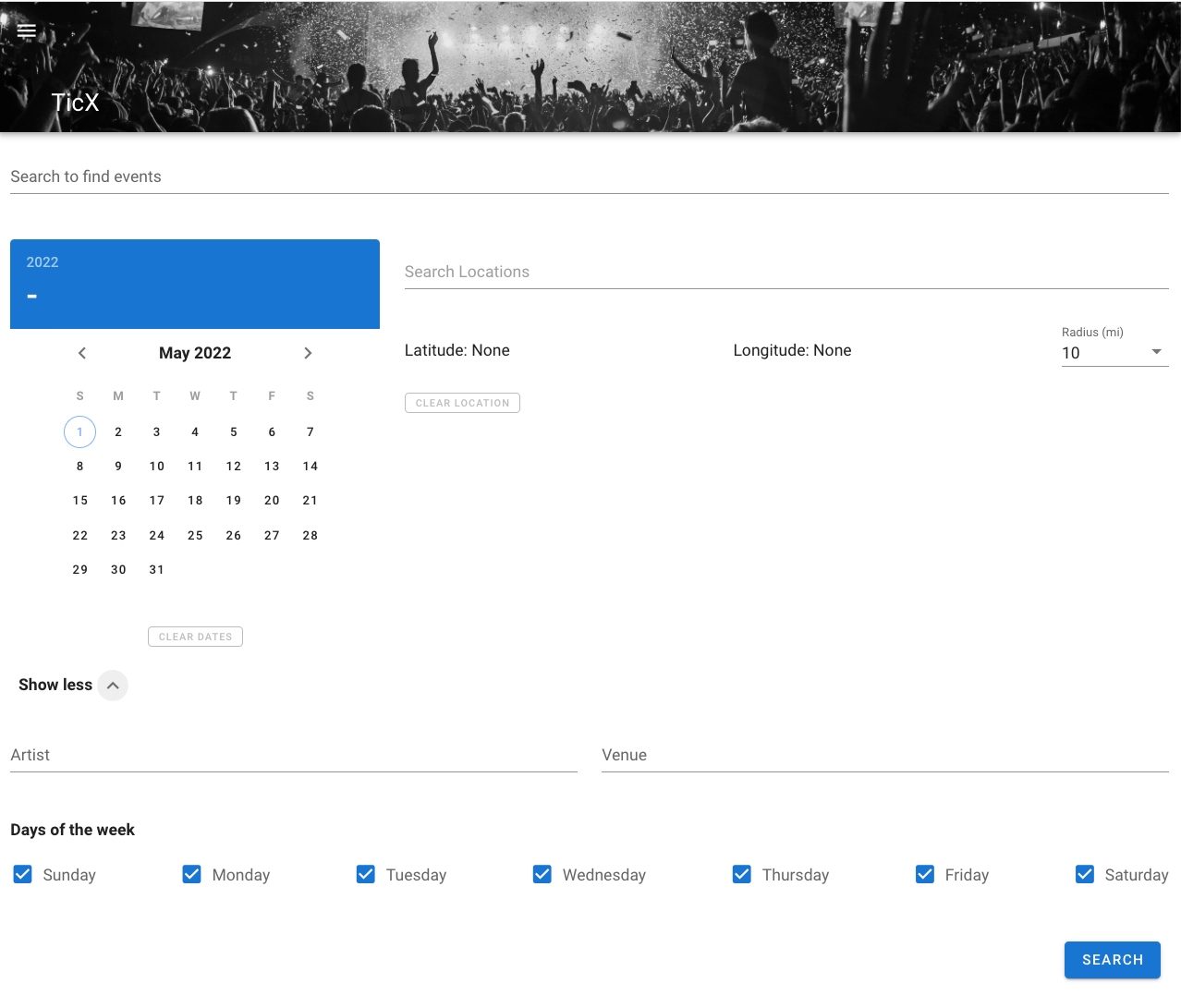
**Home and Search**

When a user lands on the TicX website, they are greeted with a landing page that gives them the option to search, login, or sign up.



*Landing page for the TicX website, where a user can log in, sign up, or start searching.*

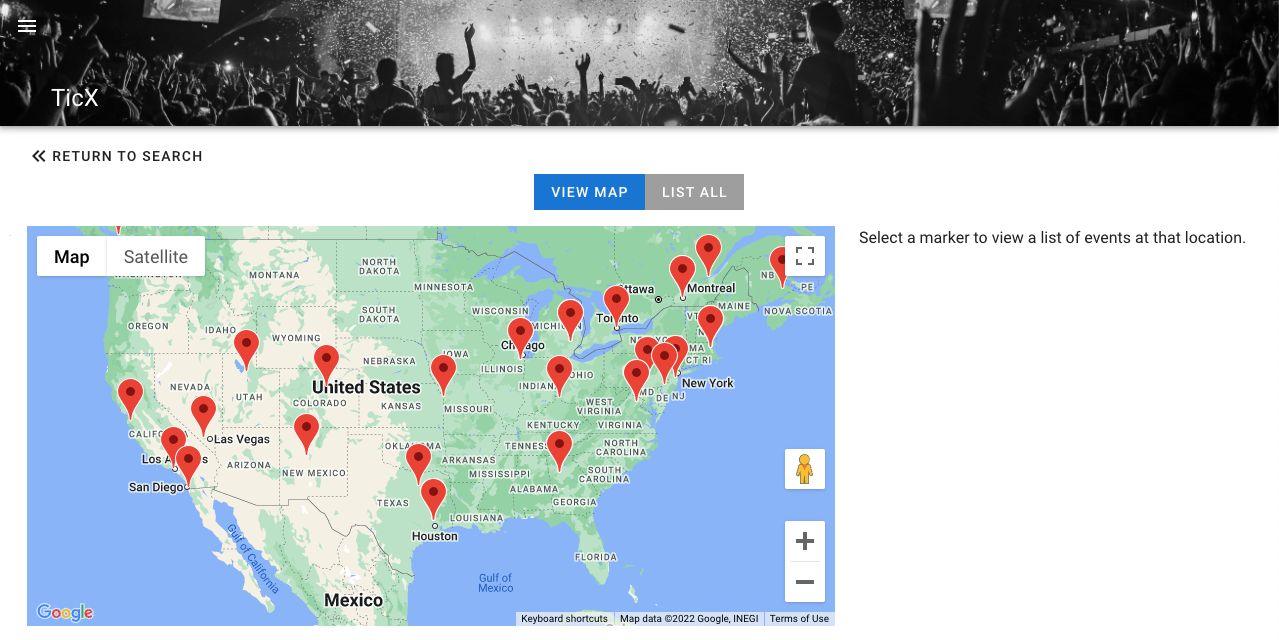
If a user clicks “Search Now”, that takes the user to the page below. Here, a user can search by a variety of criteria, including a fuzzy search at the top, locations, venues, and dates. The dates can be further customized by selecting specific days of the week; if a user wants to search for a concert on the weekend in the month of June in Alexandria, VA, those search criteria can all be combined into one query.



*Search page, which includes a plethora of filtering criteria.*

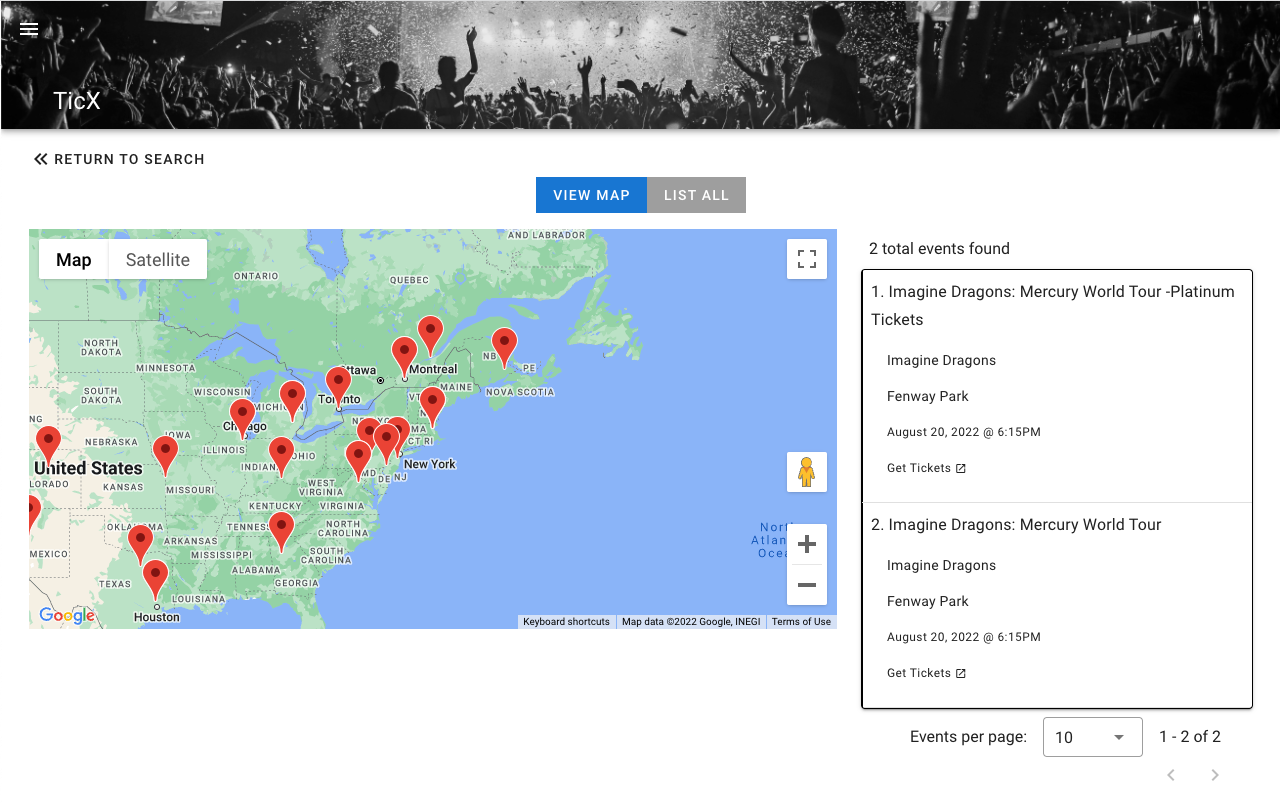
**Search Results**

After clicking search or hitting enter, a user can see the results of the search displayed on a map. In the below screenshot, the search was a fuzzy search for “Imagine Dragons,” a band that is on tour. The results page essentially shows the band’s entire tour.



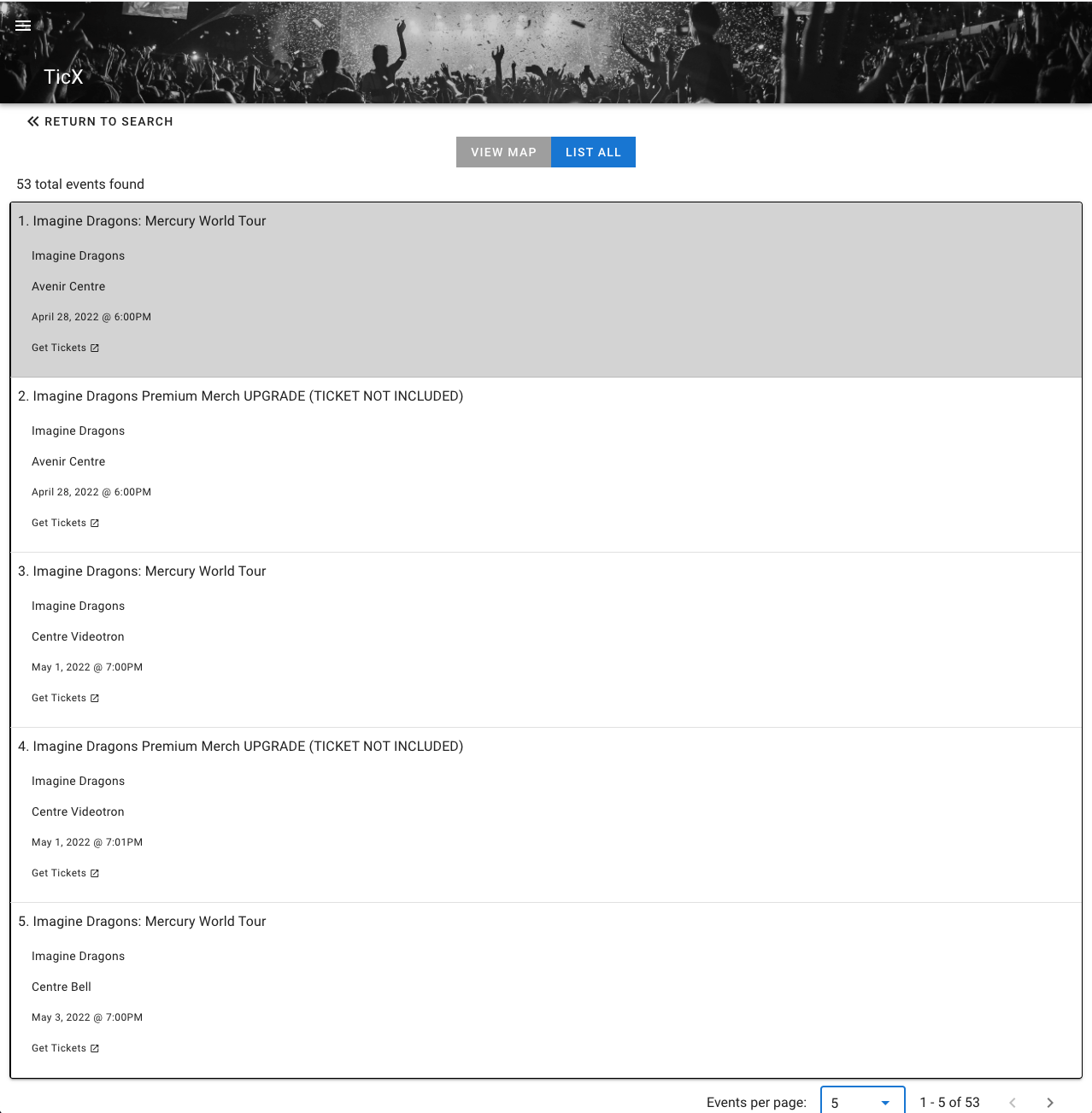
*Initial search results for fuzzy search on “Imagine Dragons.”*

Clicking on a specific marker gives the following information on the right hand side:



*Selecting a marker pops up a list on the right side.*

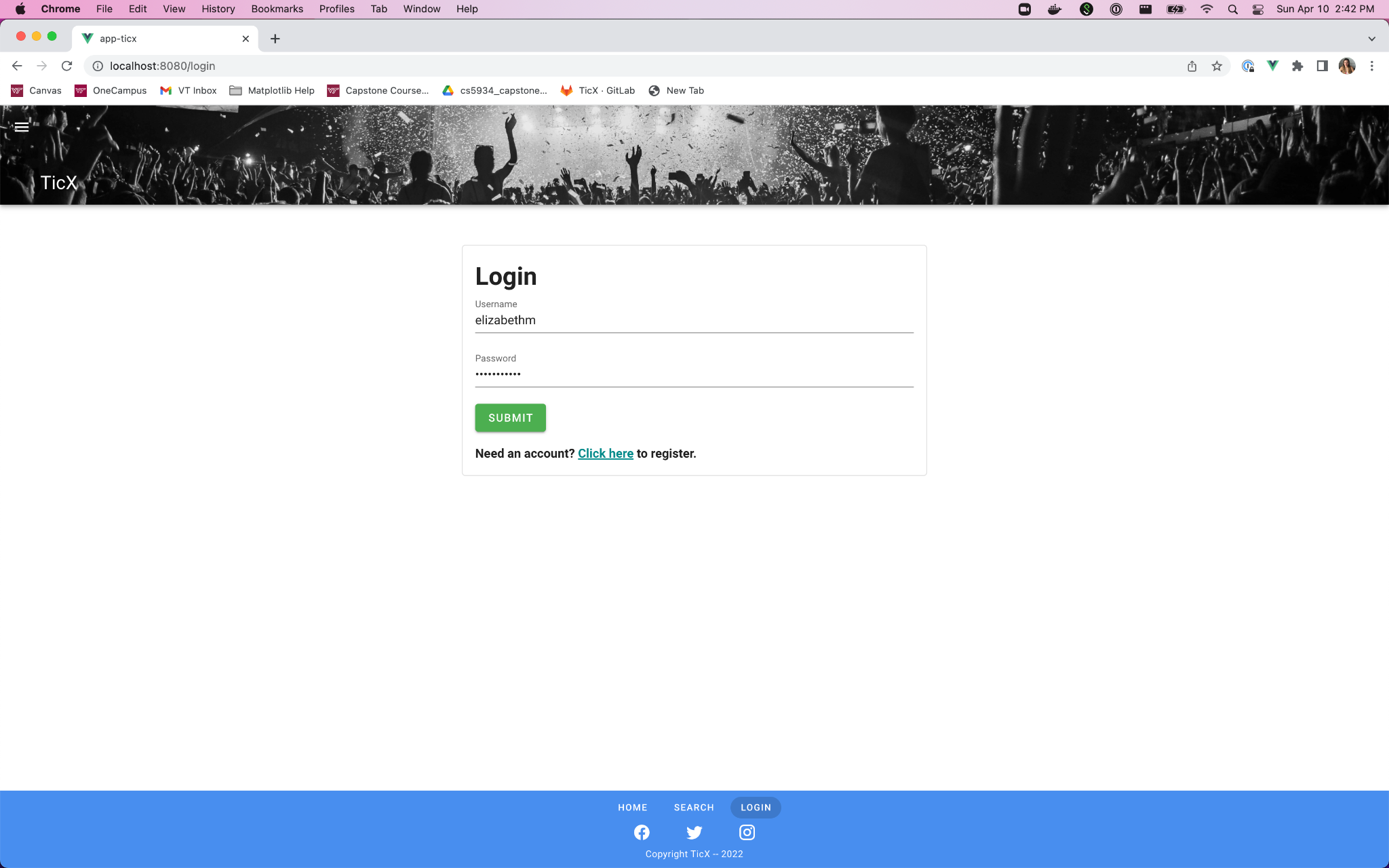
If a user selects the “List All” button at the top, they can view all results in chronological order, with the newest events at the top:



*Search results in list form.*

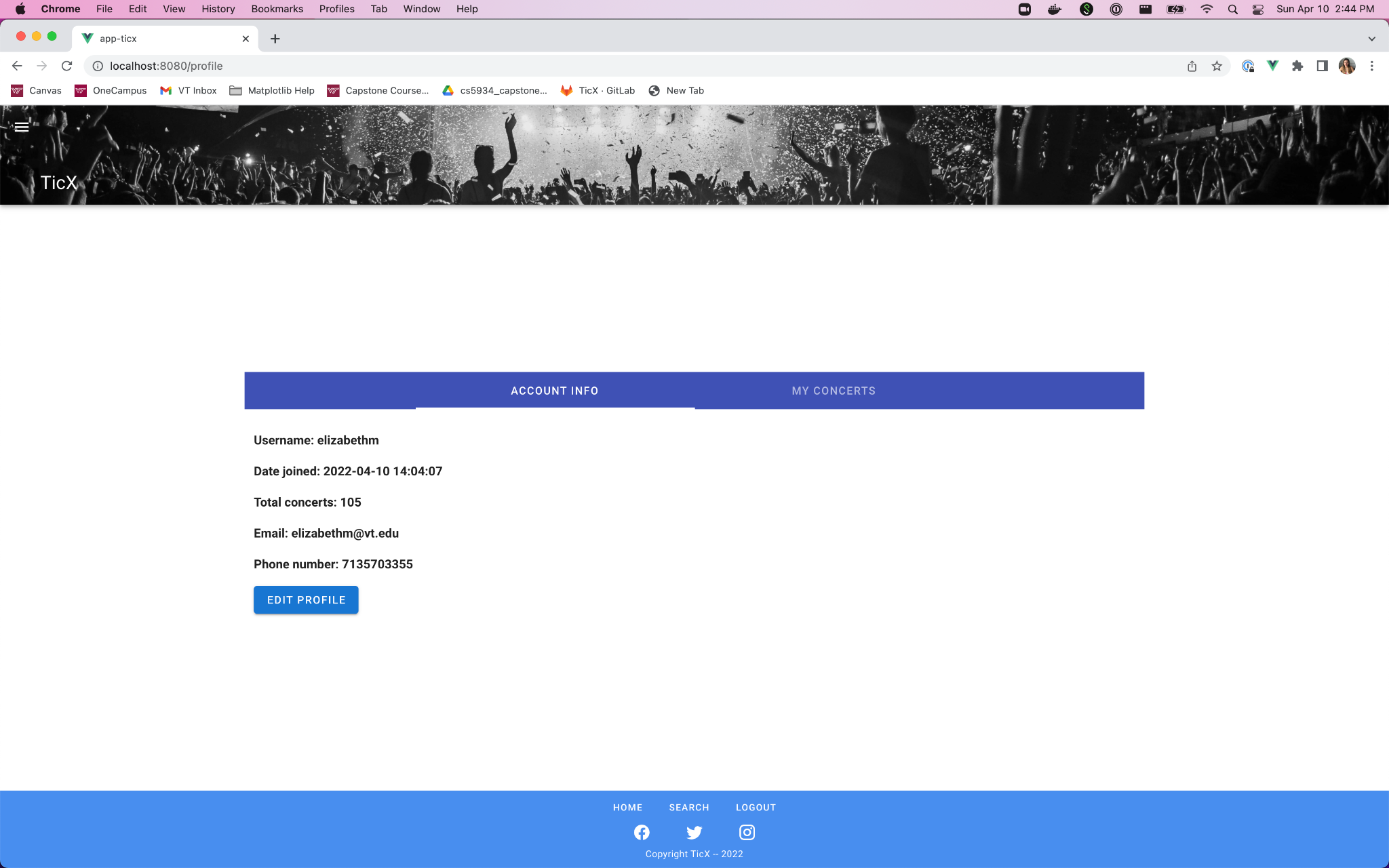
**Profile**

If a user selects log in when they first enter the site, they are brought to the following page that prompts them to enter a username and password.



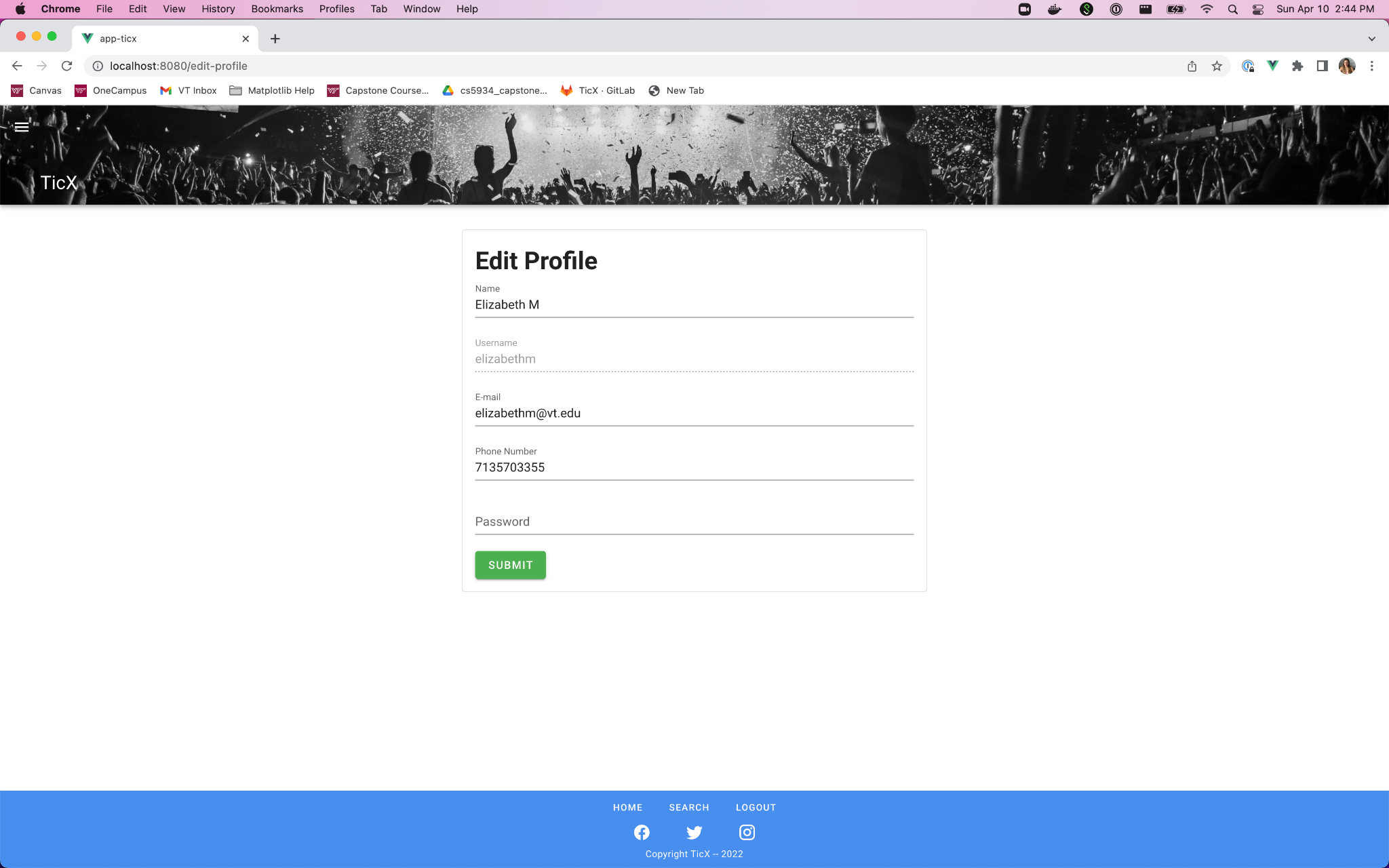
*Login page.*

Upon hitting submit for a valid username and password, a user is redirected to their profile page, as shown below. From here, a user can edit their profile as necessary.



*A user’s profile page.*

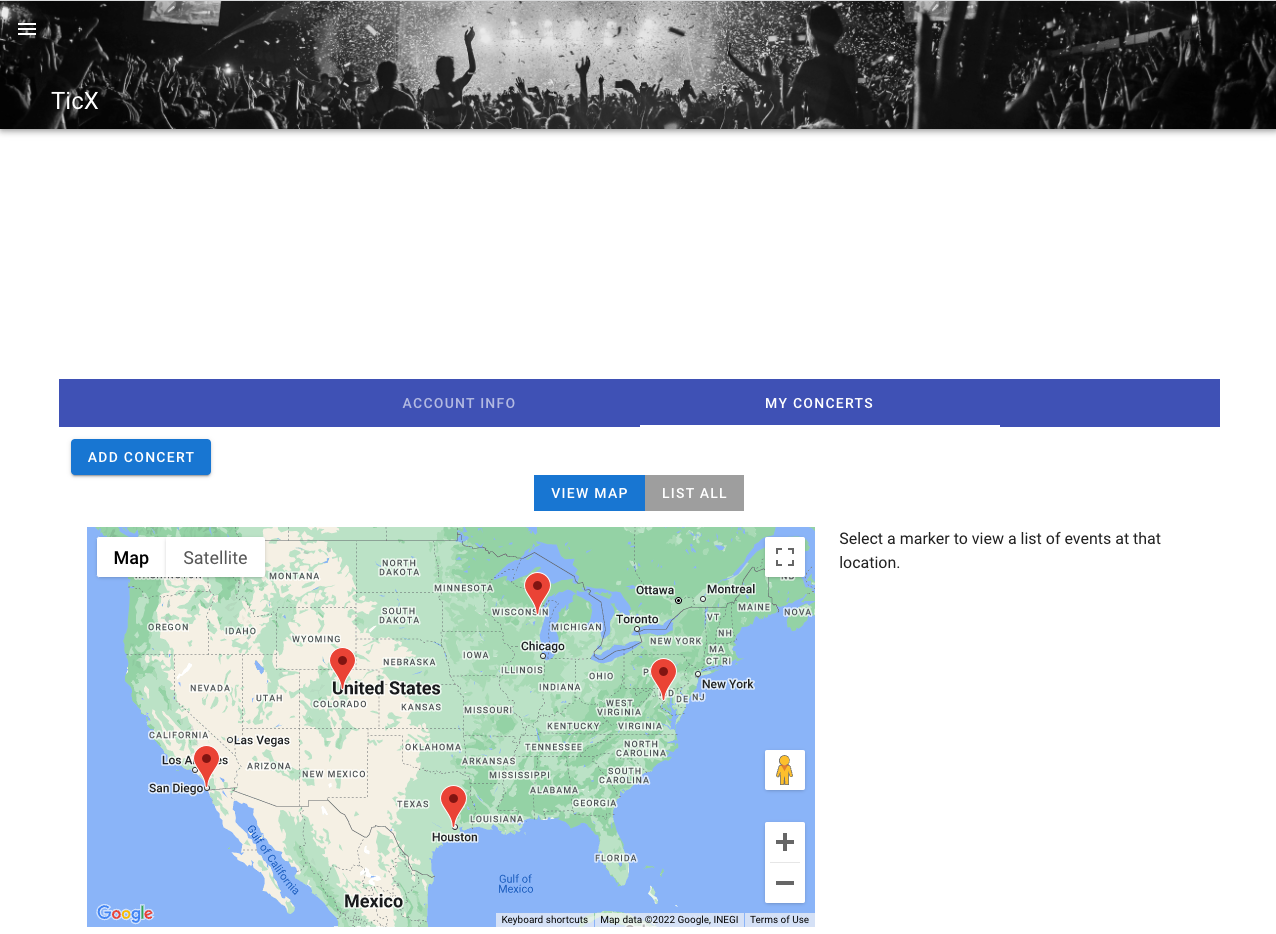
Profile editing is done with the following form input, which pre-populates a users data into the form.



*Edit profile functionality.*

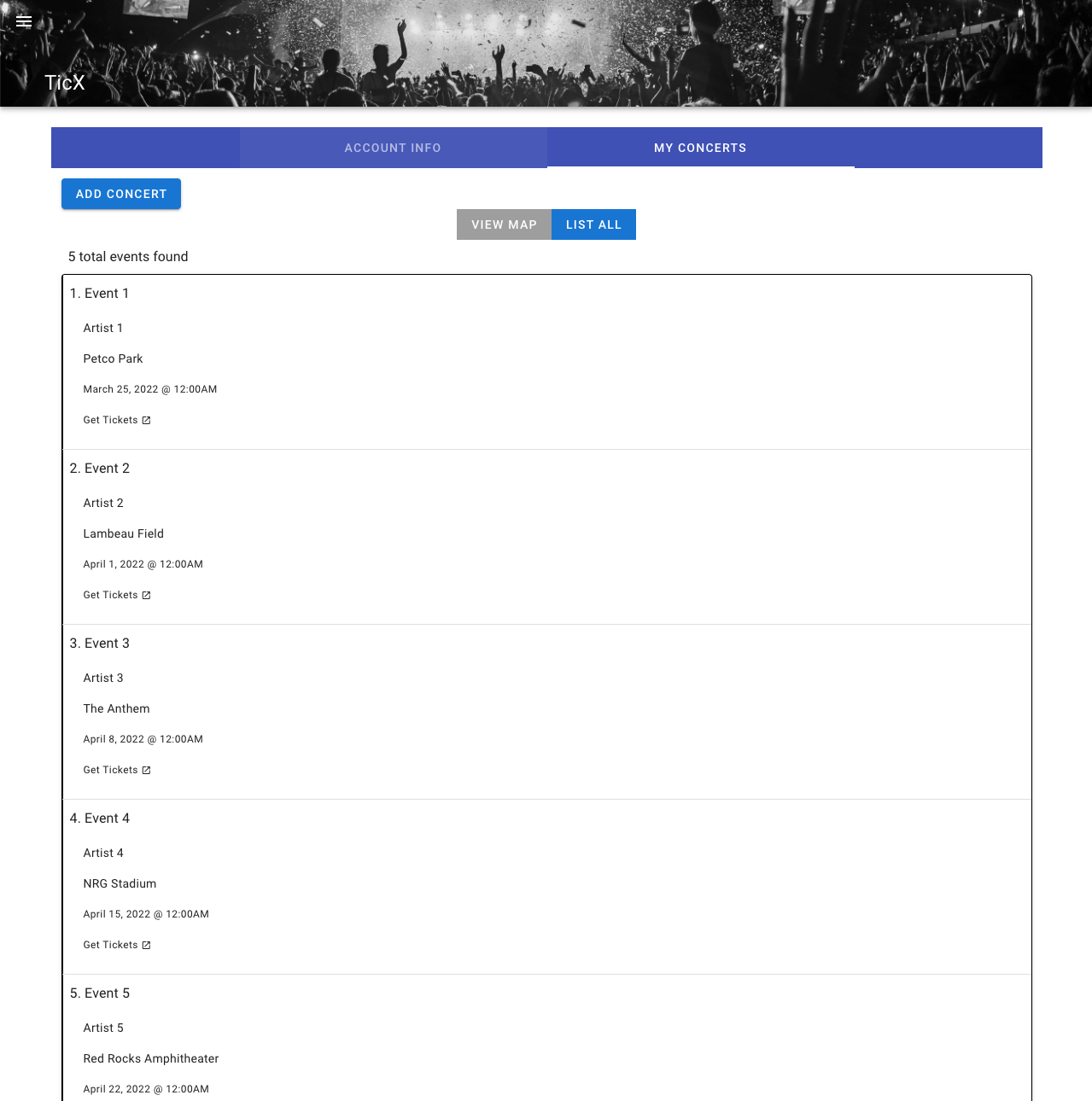
**User Concert History**

The final piece of key functionality is displaying a user’s concert history in a map form. Below is a view of a user’s concert history, displayed as a map containing five different markers. This reuses the search results components and yields a similar look, but tailored to the data a user has recorded in their concert history, which is stored in a database.



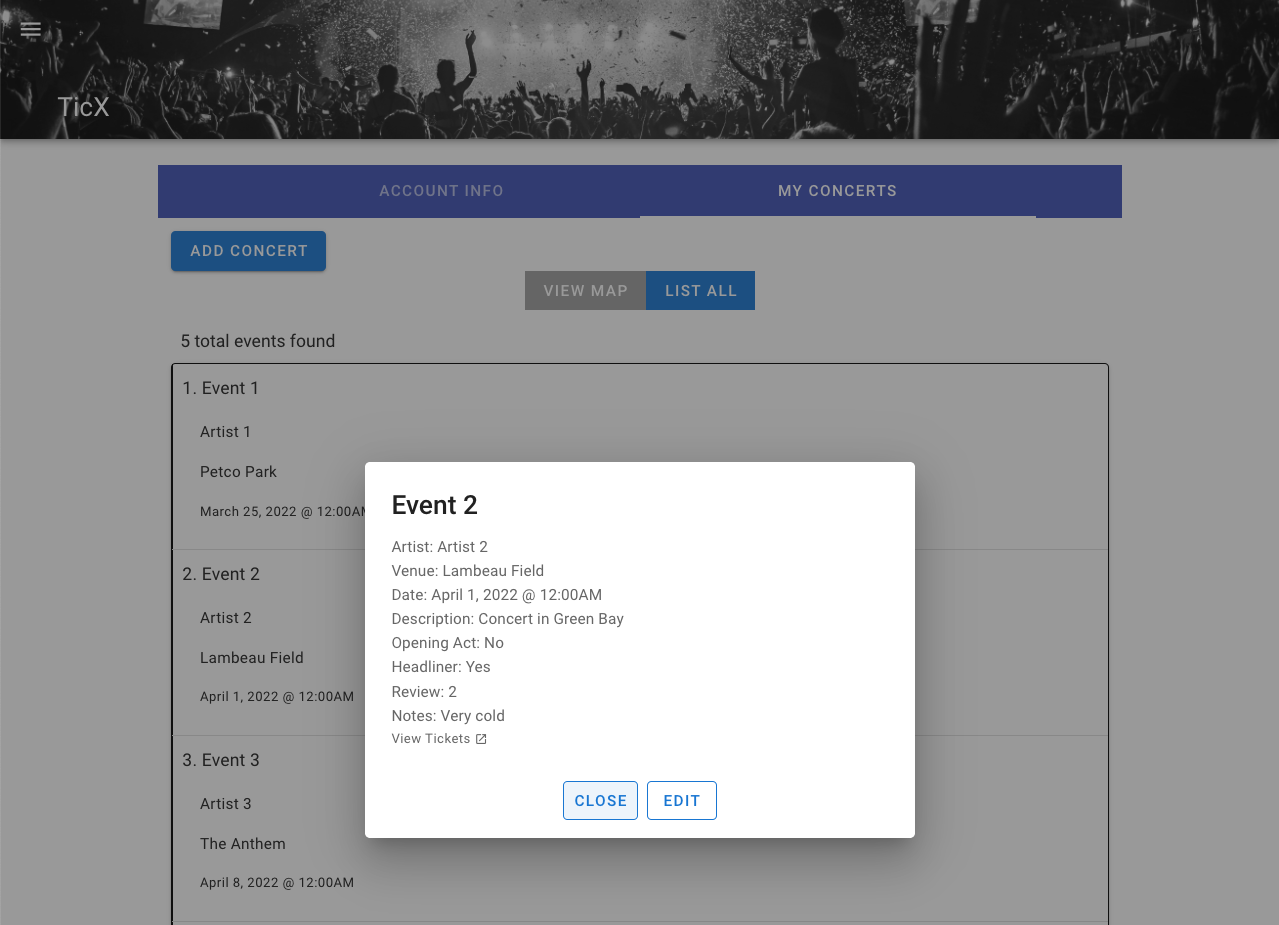
*A user’s concert history displayed as a map.*

If a user wants to view their concert history as a list, they can do so by clicking on the “List All” button.



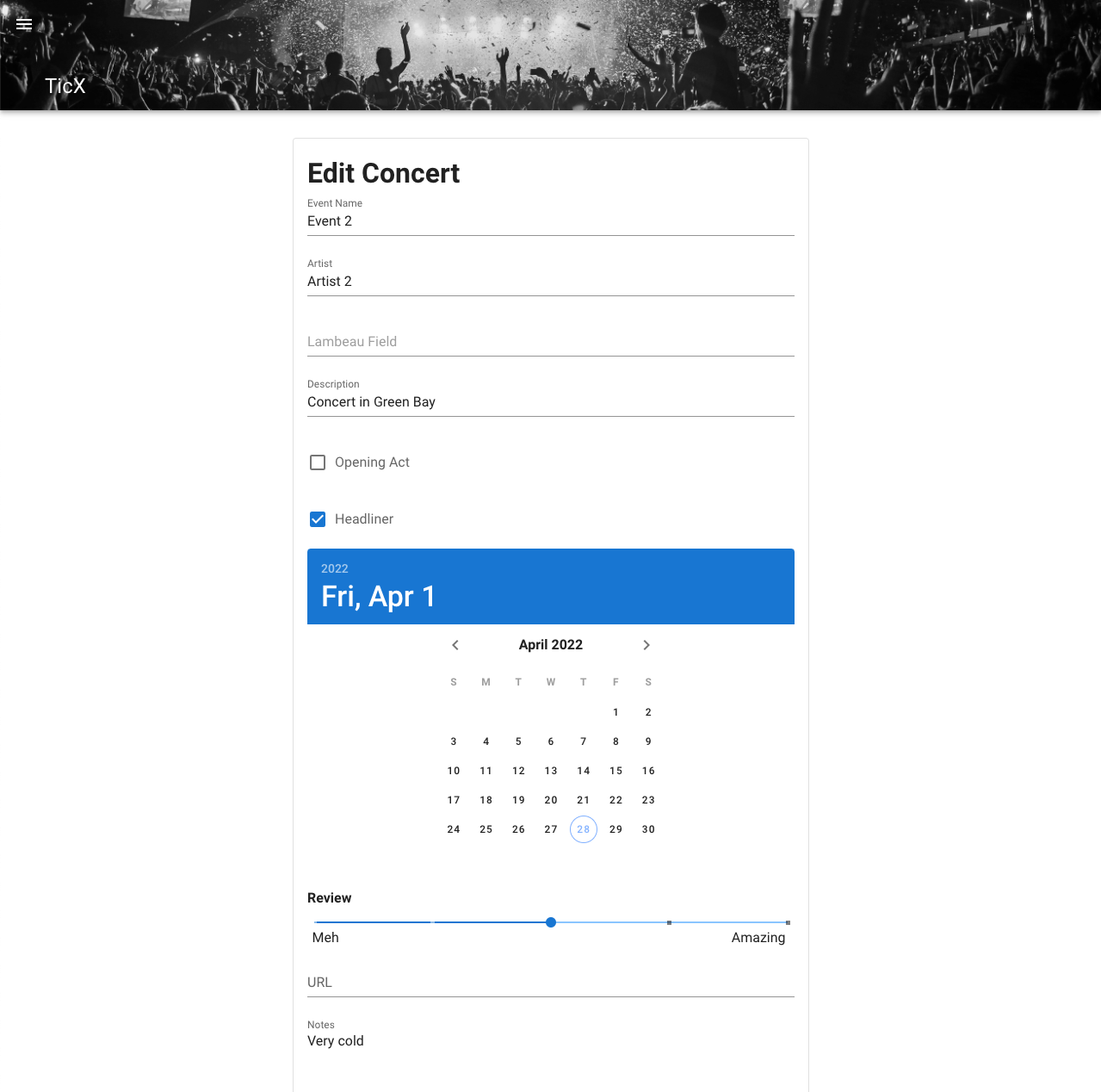
*Concert history viewed as a list.*

Clicking on an event produces the following pop up dialogue, where a user can view and edit all of the data associated with the event, including their own personal notes about the show and their review.



*User event pop up dialogue.*

Clicking “Edit” on the event pop up dialogue gives the user another form input, similar to how they might edit their profile information.



*Editing a user event item.*

# 

# Design

The technology stack for Ticx consists of Vue2 on the front-end, Flask on the back-end, MySQL as the database, and Docker to host the application. For the front-end we imported several useful libraries into Vue, a framework consisting of html, javascript, and css, to simplify building front end components. One of these libraries was Vuetify, a material design framework, providing a lot of small, easy to customize components. These small components played a large role in how we displayed input fields as well as some of the data to the user. It also includes customizable styles, icons, and themes that we took full advantage of to cut down on the time required to develop professional-looking components.

We also imported the vue2-google-maps from npm to implement the interactive map for marking different venues and events in search results and a user’s event history. Again, there were different properties we had to alter to fit the goals and features of our application, such as which locations to display, the types of the locations, the zoom properties, and where to center the map using latitude and longitude. These libraries had a major impact on the design and visual aspects for what the user sees, allowing us to create an appealing user interface in three short sprint timelines.

On the back-end, we chose to use Flask because it offers a simple framework for creating a RESTful API. We also chose Flask because it is developed in Python which is the coding language a majority of our team was most familiar with. The API is used to handle the bulk of our data collection and data processing. It is where the front-end sends user search queries to be aligned with the APIs of the ticketing services and obtain concert event information. Once data is retrieved from each of the different ticketing services, we then normalize and aggregate the results to fit our own data schema that can be interpreted by our front-end application.

Again, we chose to include MySQL as the database for our design because many of us have had experience working with and writing SQL statements. Choosing the best fitting database was not as critical because our application does not store large amounts of data. The goal of our project is to provide real time search results for events obtained by external ticketing services and so we chose not to cache or store event information in local storage or one of our databases; this could yield outdated event information. The database was used, however, to store three different tables: users, venues, and events, which all store data related to users. The combination of the three provides the functionality for a user to login and track a history of all events they have attended in the past.

We used Docker to host the web application and made it accessible to users from their web browser. Initially we had discussed using Amazon Web Services to host the application because AWS provides a bunch of great cloud services, even in their free tier, that make deployments and hosting easy. However, we ended up using Docker because Virginia Tech offers a cloud environment to students free of charge. Within the VT Cloud, we built three different docker images: one for the front-end, one for the api, and one for our database. These images are run on pods within the same namespace so that they could communicate with each other. We chose to configure our Docker images this way so that changes to one part of the application would not necessitate us rebuilding the entire application in Docker.

# Retrospection

The overall process of building our product went very well. Considering more than half of the team work full time jobs as well as taking other classes, it was surprising that we were able to make time to meet up weekly to ensure we were on track and hitting our goals for the ongoing sprint. Our team included a variety of different skill sets, some having more experience with web development than others. A combination of those dedicated to learning and others willing to share their knowledge led to a lot of pair-programming that was both productive and fun.

The Ticx application itself came out quite well. We were able to complete most of the features we had initially set out to complete as well as adding some nice to haves along the way. There were a few roadblocks that arose during sprint two when integrating with the ticketing services. Stubhub and Ticketmaster had query limits on their APIs in which we did not initially expect, and we did not want to pay to expand the query limit. As a result, we had to refactor our code to get the most out of the APIs, which primarily meant limiting a user’s search radius to a maximum of 10 miles for searching events within a given location.

One of our biggest challenges, which was discussed in some of our sprint deliverables, was that we all had different operating systems and different preferred developer environments. The mix of Windows and macOS operating systems, plus different package managers and command line environments meant that we spent a lot of time figuring out configuration issues. This meant we lost some time that we otherwise could have spent on expanding functionality or creating a more customized look to the website. We primarily addressed this challenge by creating detailed readme files and adding in notes where we saw differences between Windows and macOS systems.

We also encountered many issues working with the VT Cloud and Docker. There is not a lot of documentation on how to integrate with the VT Cloud but with the assistance of another team, we were able to figure it out. To address the lack of instructions, we had also maintained some of our own documentation of how we used docker so that other teammates and others who may want to stand up their own environment and run Ticx can do so. In the end, we are very happy with the design of the Ticx interface and the key functionalities of our product.

# Recommendations for Future Work

Our team came up with a list of future functionality that we think would be useful for this platform, particularly if we chose to continue developing the TicX application for market use. Some sample user stories could be the following:

* As a user, I want to be able to see event recommendations so that I can explore more events
* As a user, I want to be able to share my profile data to social media
* As a user, I want to be able to use the TicX platform on a mobile application
* As a user, I want to be able to log in with existing account information from other platforms, like Google and Facebook accounts.
* As a user, I want to be able to receive notifications about announcements for new shows at my favorite venues and/or my favorite artists.
* As a user, I want to bookmark future events that I discover in the TicX platform, to save them to my profile.
* As a developer, I want to customize the appearance of google maps markers in map views for the search results and user event history components.
* As a developer, I want to integrate sponsored content into the TicX platform so that I can monetize the application.
* As a developer, I want to query more APIs and add more ticketsellers to the platform so I can offer more complete event information to users.

These all represent mostly simple expansions to functionality, but one of the key things we would change in a market-ready platform is the way we’ve structured user accounts and the user table in our database. Allowing users to log in with external account information would give us more information about who is using the platform and implement more complex user-specific functionality, like recommending specific events or sponsored content.

Another big step in functionality would be adding iOS and Android apps to the platform that could allow users to access TicX in a mobile-friendly format. We recognize that a lot of our user personas circle around people who travel, so it would be ideal to provide them the full user experience directly on their phones rather than in a mobile browser. Additionally, this would make the “sharing events to social media” experience more seamless, since people normally access social media through their mobile devices and not web browsers.