Event Recommender & Event Sharer Amongst Friends

Members

Roshan Babu Koottiyaniyel- rbk2145

Sairam Haribabu - sh4188

Aishwarya Sivakumar- as6418

Kaylee Trevino - kt2846

Description

Sometimes, it is difficult to decide what to do on the weekends, especially because you are not always aware of all of the possible events there are for you and your friends to do in the area. For our web service, we will maintain an online system that allows users to select what kind of events they are interested in, such as outdoor events, food and bar-related events, and show-related events, as well as what location that user is looking from. We can even use less-common categories such as looking for events that are “funny”, which may be related to comedy-shows for example.

Based on the user’s interests, we will recommend events to the user that fits their preferences. For each of these events, we will provide information such as the venue, location, website, and more so that the user can further inspect if they’re interested. These kinds of events can be found using an API such as eventbrite’s API.

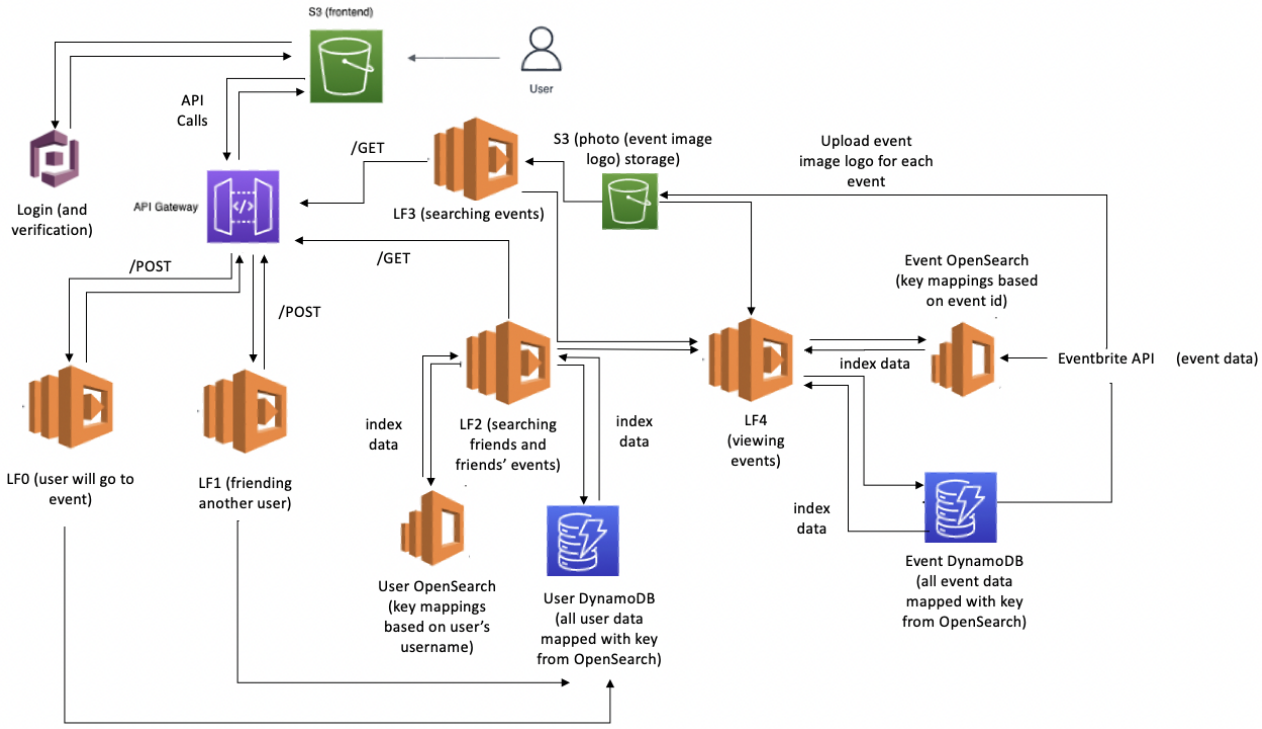
We will also maintain a profile for each user that maintains their profile information and the user’s “saved”/”favorited” events. With this service, we will also maintain a social media-like aspect of this service so that users can have “friends”, and allow their users to view their saved events which they plan to attend, as well as look at their friend’s profiles to see what their friends are planning to do. Additionally, we will also allow users to send events they like to their friends so that their friends can become aware of events, introducing a more social aspect to this service.

Finally, unlike other applications, we will provide a screen where a user can view a feed of events their friends are visiting. This gives a more social media-like experience.

Clickable Prototype

Figma Clickable Prototype (press play button to see popup window that will show prototype)

Architecture Diagram



The flow is as follows:

1. User accesses the frontend hosted on S3
2. User must login with valid username and password, which will be verified using Cognito
3. After login, user can interact with frontend hosted on S3
    a. User can search for events based on filters
        i. These filters include venue, location, event category, and more
        ii. Results that match with users requests will be displayed on screen
    b. User can look at events which their friends (whom they have friended on the app) have decided to attend
        i. Results will be displayed on screen
4. S3 makes API calls via API Gateway
5. API Gateway has two /GET calls

    Get Information about upcoming events

    a. The GET call uses LF3 which fetches basic information about all the upcoming events from an S3 bucket, such as the event logo (or thumbnail), the event name and the event ID.

b. When the user clicks on any event, LF4 is triggered which will use the event ID to get the key mapping from EventOpenSearch and use that index to query in EventDynamoDB to fetch the complete details of that event

Get Information about the events your friends are attending

a. For the user, we first hit UserOpenSearch to find the list of friends for that user.
b. For each friend, we hit UserDynamoDB, which will have the complete information of each friend. From this entry, we fetch the list of eventIDs of the events the friend is going to. We return this information to LF2, which then hits the S3 bucket and fetches the event names and logos for each eventID. When the user clicks on any event, LF4 is triggered which will use the event ID to get the key mapping from EventOpenSearch and use that index to query in EventDynamoDB to fetch the complete details of that event

6. API Gateway has two /POST calls
    a. One indicates that a user has decided to attend an event
        i. This event will then be updated in the Users DynamoDB so that this event is now listed as one of the events this user will attend
    b. The other is called when a user decides to friend another user
        i. Each friend will then be updated in the Users DynamoDB so that the friends will be listed in each other's list of friends in the database
7. Data in the Events OpenSearch, Events DynamoDB, and S3 photos bucket will be populated in advance from Eventbrite API
    a. The S3 photos bucket will hold the event image logos associated with each event (since S3 is more suitable for photos than DynamoDB)
8. Entries in the Users OpenSearch and Users DynamoDB will be populated for each user as a user signs up and creates a username and password
9. Query results and updates to a user's profile (such as adding a friend or an event) will be displayed in the frontend for the users to see

API

/GET (for events)

Description:

With the JSON response returned from the lambda function describing the events and the event icon images found in the S3 bucket (using a string photo key), we will display the images and event information on the frontend. This can also be based on requested information from the user, where the user will specify a desired location, category, and date of the event if they would like to do so. The \GET call will allow us to retrieve these photos from S3 and related event information from the database.

Request Body {

location: string,

category: string,

date: Date

}

Response Body {

event: [Event1, Event2, …]

}

Each Event object is in the following format:

Event = {

name: string,

description: string,

venue_name: string,

venue_address: string,

image: image,

time: Time,

status: string,

photo_key: string

}

/GET (for friends and their events)

Description:

With the JSON response returned from the lambda function describing the events and the event icon images found in the S3 bucket (using a string photo key) only for the user's friends, we will display only

those images and event information on the frontend. The \GET call will allow us to retrieve these photos from S3 and related event information from the database.

Request Body {

location: string,

category: string,

date: Date

}

Response Body {

event: [Event1, Event2, …]

}

/POST

Description:

After selecting an event, a user will click a button indicating they will be attending that event.

Request Body {

user_id: string,

event_id: string

}

Response Body {

success: Boolean

/POST

Description:

A user will click a button indicating they would like to friend another user

Request Body {

user_id: string,

friend_id: string

}

Response Body {

success: Boolean

Descriptions of External APIs

Eventbrite API

- Can list events by Venue ID, Organization ID, or Series ID

    - An event series is a repeating event w/ a sequence of multiple & various dates
- For each event, can use another call to get event capacity, description, schedule, and series information
- For each event object, we will obtain the following provided fields: id, name, summary, description, url, start, end, created, status (draft, live, started, ended, completed, canceled), currency, and if the event is an online event
- For each venue, we will obtain the following provided fields: address, id, age restriction, capacity, name

TASKS

https://docs.google.com/spreadsheets/d/1svLnFyY7eAvKJKJ8Iarg3DQbFAwBg3FONop3B3ZG6-s/edit#gid=0