COMSE6998_013_2021_3 - CLOUD COMPUTING AND BIG DATA SYSTEMS FINAL PROJECT

Submitted By:
Chandan Suri (CS4090)
Katie Jooyoung Kim (JK4534)
Umang Raj (UR2136)
Viswajit Vinod Nair (VV2339)

Table Of Contents

Project Proposal

Description

DrinkEasy is a web app designed to make socializing easier for NYC. Our app will collect data on bars/clubs near the user or a location of the user's choice and make smart suggestions based on relevant information such as the level of crowdedness, operating times, user reviews, ratings, lgbtq friendliness etc. The aim here is to empower the user to make an efficient and faster decision about the place that they want to visit based on the most important parameters.

Motivation

The main motivation is to develop an application exclusively for 'Party People' aka people who like to party!

The search for bars, pubs, nightclubs, and other 'party' places online subject to parameters like price levels, ratings etc. can be a cumbersome task. This also applies to the search for places based on the availability of the place (crowdedness/ popularity) at any given time. For instance, a basic Google search would entail the filtering criteria to be explicitly mentioned in the search text, which is not always very convenient and does not provide all relevant information in one single place.

Also, searching for places based on categories is not available through google search directly. Similarly, for searching according to the crowdedness levels is not a functionality that google provides. Basically, you can look up the popular times data through google search but there is no way to directly search on that basis, you would need to wade through the list of the bars in order to find one according to your suitability. Here is where we shine the brightest!

An additional reason is the limited number of solutions available online that provide intelligent functionality that cater to this requirement. For example, our proposed web app will provide an option to users to make certain places 'favorites' and also see what other users are liking without having to go through the reviews.

Project Prototype

Description

The technical stack that has been used for the project is:
1. ReactJS for Frontend
2. AWS Lambda Functions and "Pre-Processing" scripts (Written in Python 3) for Backend
3. AWS DynamoDB, Open Search and S3 for Database

We have three working tabs on the UI.

1. Explore Tab:

In this tab, the user can make queries based on a particular zip code. There is a search bar located on the top of the screen, where users can enter the zip code where they want to view the bars or pubs in.

There are filters like LGBTQ+ friendliness (checkbox with two values: true or false), Crowdedness Level (Dropdown with five values: 1, 2, 3, 4, 5 where 1 means least crowded and 5 means most crowded), Date (date picker), Time (time picker), Rating (Dropdown with four values: 1, 2, 3, 4+) and Pricing (Dropdown with values $, $$, $$$, $$$$ where $ signifies least costly and $$$$ signifies most costly).

There is also a 'Search' button that has to be clicked once the zip code is entered and filters are applied to display the results.

We have a 'Trending Places' section which displays all the bars and pubs that are fairly crowded in a particular area for all those days when you just feel more social. This is populated by setting the crowdedness level as 3 and above.

We have a 'Top Rated Brands' section which displays all the bars and pubs that are rated 4 and above in a particular area.

We have individual cards showing:
(a) Image of Establishment (Fetched from S3 Bucket using PlaceID)
(b) Name of Establishment
(c) Rating
(d) Price Level
(e) Crowdedness Level (In the form of Wine Glasses)
(f) LGBTQ+ friendliness (Pride flag denotes that the establishment is LGBTQ+ friendly)
(g) Contact Information
(h) Google Maps link that takes you to the Google Maps page of the establishment that would display other relevant information that is not available directly on the website.
(i) Favorite/Unfavorite button.

2. Choose Your Vibe Tab:

In this tab, we divide the establishments into trendy and relevant categories like ‘Sports Bars’, ‘Jazz Bars’, ‘Night Clubs’ and ‘LGBTQ Friendly’ that have been generated by preprocessing data gathered from Google reviews of various establishments.

There is a search bar located on the top of the screen, where users can enter the zip code where they want to view the bars or pubs in.

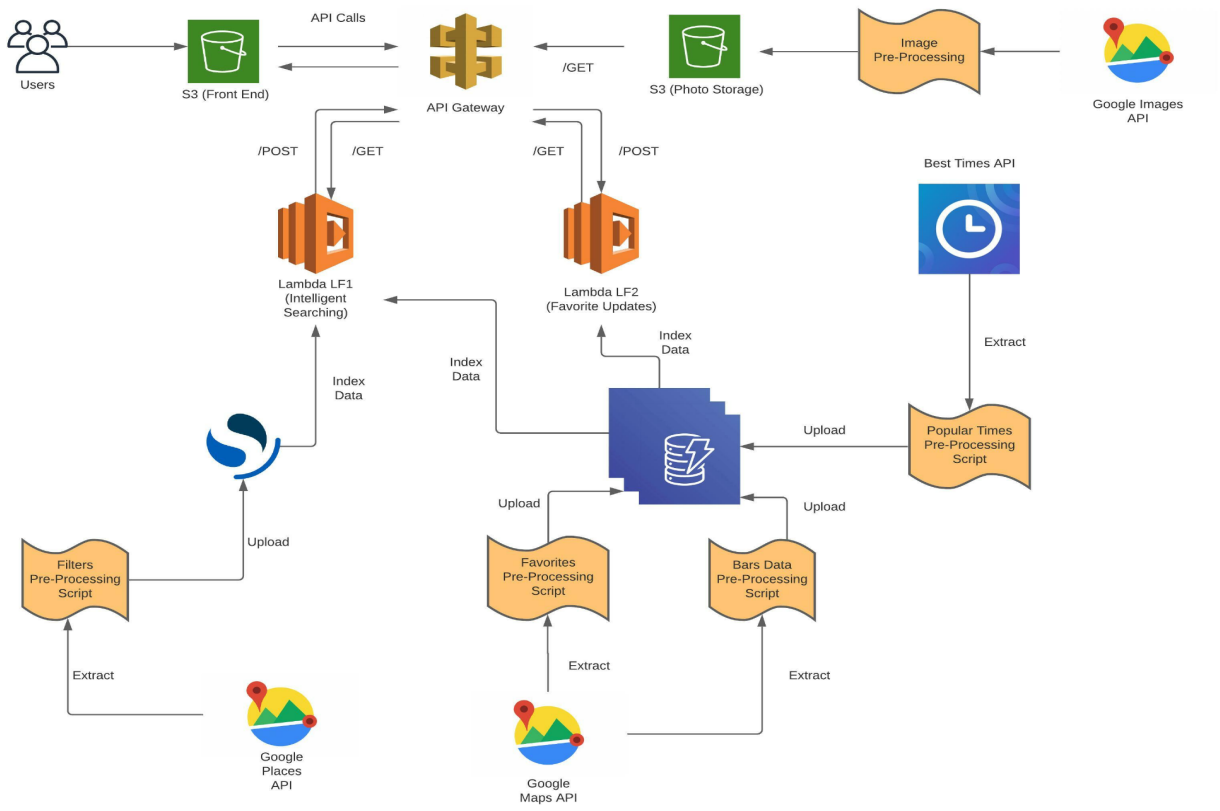
There are filters like Crowdedness Level (dropdown with five values: 1, 2, 3, 4, 5 where 1 means least crowded and 5 means most crowded), Date (date picker), Time (time picker), Rating (dropdown with four values: 1, 2, 3, 4+) and Pricing (dropdown with values $, $$, $$$, $$$$ where $ signifies least costly and $$$$ signifies most costly).

There is no ‘Search’ button. A user can enter the zip code and select the relevant filters and then click on a particular ‘Category’ to fetch all the establishments belonging to that particular category, filtered by the user-defined parameters.

3. Favorites Tab :

In this tab, the user can view all the establishments that they ‘favorited’ using the ‘Favorite’ button located on the card corresponding to the establishment for easy access.

Architecture Diagram



Description of Components

1. S3 (Front End): It is used to host the frontend of the application.

2. API Gateway: It is used to define API calls for the application and acts as an interface between the frontend and the lambda functions.

3. S3 (Photo Storage): It is used to store the images of the establishments referenced by the unique place_id.

4. Lambda LF1 (Intelligent Searching): It is used to support zip code based queries, filtering and categorization queries.

5. Lambda LF2 (Favorite Updates): It is used to support ‘favorite’ functionality and for fetching favorites for a particular user.

6. DynamoDB: It is used to store data relevant to the application. Full description of the tables is shown under the database model section.

7. OpenSearch: It is used to store all the filtering information applicable to our use case. This is done as such as it's faster to retrieve the bar IDs based on the filtering criterions first and then getting the full data. The data for this is fetched using multiple API calls to the Google Places API as explained below.

8. Google Places API: Mainly used for getting all the data for the bars. This data is stored in the Dynamo DB according to the key for the bars. We can filter out all the bars from the Google Places stored and store all the data in the DynamoDB beforehand. This is done to optimize the time needed for searching for the bars according to the filter stated by the user. The data would primarily include the following (for each bar):

a. Key for the Bars: Used as a primary key to retrieve some particular bars.
b. Zip Code: zip code in which the bar is stated.
c. Geolocation coordinate: latitude and longitude data for the bar location.
d. Name of the bar
e. Complete address of the bar
f. Open Hours
g. User's Rating
h. Phone Number/s
i. How Costly? This tells us how costly the place is. Numeric (1-5, 5 being exorbitant)
j. LGBTQ Friendliness (Gradation = 1-5)
k. Link to Google Search
l. Associated Tags
m. Associated Photos

The 'key', 'zip code' and 'geolocation' attributes are used as a "Composite Primary key" for optimized data retrieval.

9. Google Maps API: Mainly used for mapping the user entered zip code to the actual geolocation coordinates. This is in turn used to key the data in the elastic search database for retrieving the nearest coordinates.
This gives us the following data:

a. Location or Zip Code: This is key here as according to this we get the geolocation coordinates.
b. Geolocation coordinates: This gives us the actual coordinates of the location. This is used to map the coordinates entered by the user (user location in terms of zip code) to the geolocation coordinates.

10. BestTimes API: This is the data that tells us the crowdedness of the bar at some point in time. This gives us the full popular times map for a day (using data analysis). This data is scraped according to location entered by the user. Initially the mapping is taken from the API through calling the API link. The data is linked by the place IDs to busy hours, and the crowdedness level for the busy hours is stored as peak hours. This mapping is retrieved through the API calls and is remapped to the place IDs for the data retrieved. The data would include:

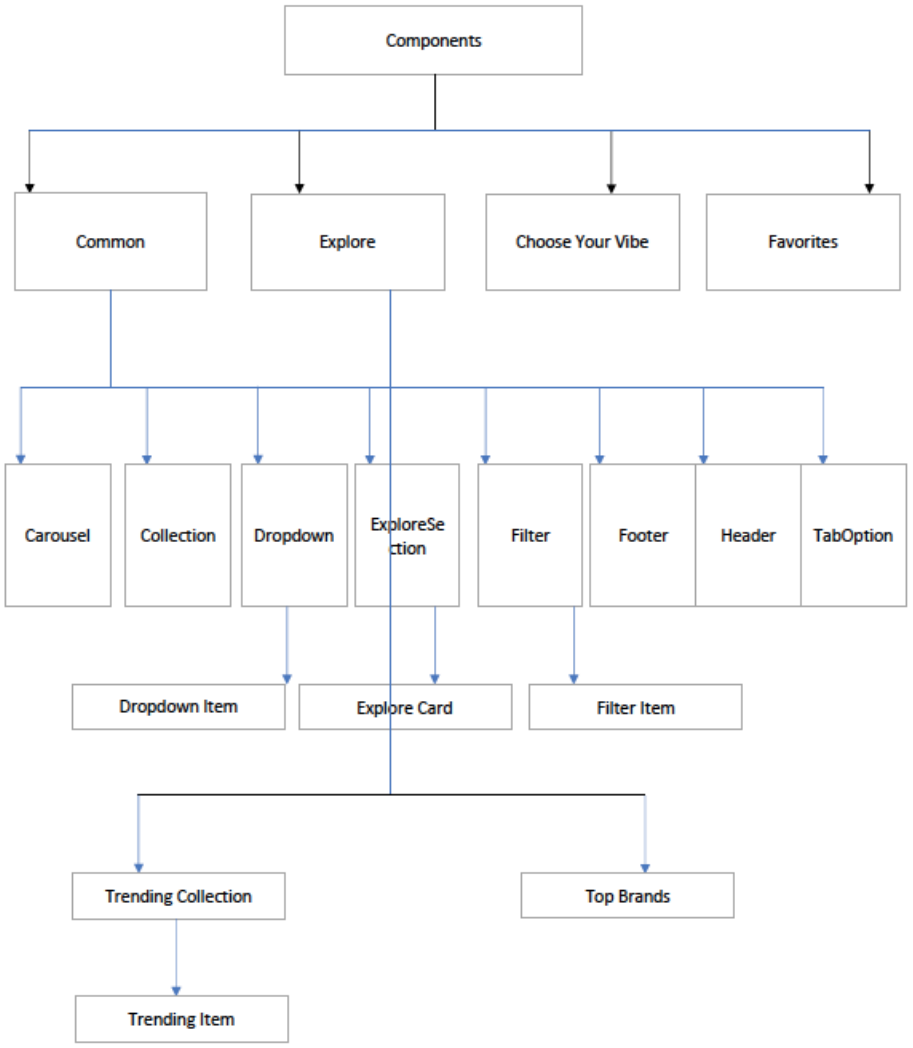
a. Bar Key
b. Bar Name
c. Popular Times Map (for each hour of the open hours) - Crowdedness level for an hour in the day is suggested by a numeric value in the range of 0-5. It is stored as a string of 24 characters, each character suggesting the crowdedness level for an hour of the day.

The Popular Times Map or the crowdedness level is also shown in the UI of our website. This data is retrieved from the API and filtered on the basis of the bar keys (a part of the primary key for the data stored in the DynamoDB).

11. Google Images API : Used for obtaining the cover images corresponding to each of the venues on the frontend. Once the data for each of the venues is collected through the Google Maps API, the unique URL for the cover image of the venue is stored in DynamoDB. This unique URL can then be used to retrieve the actual photos through calling the Google Images API. In turn, all of the images fetched using the API calls are then stored in a S3 bucket and the name of the images is the same as the bar ID. This naming system facilitates efficient retrieval of the image from the S3 bucket.

Code Structure

Flow:



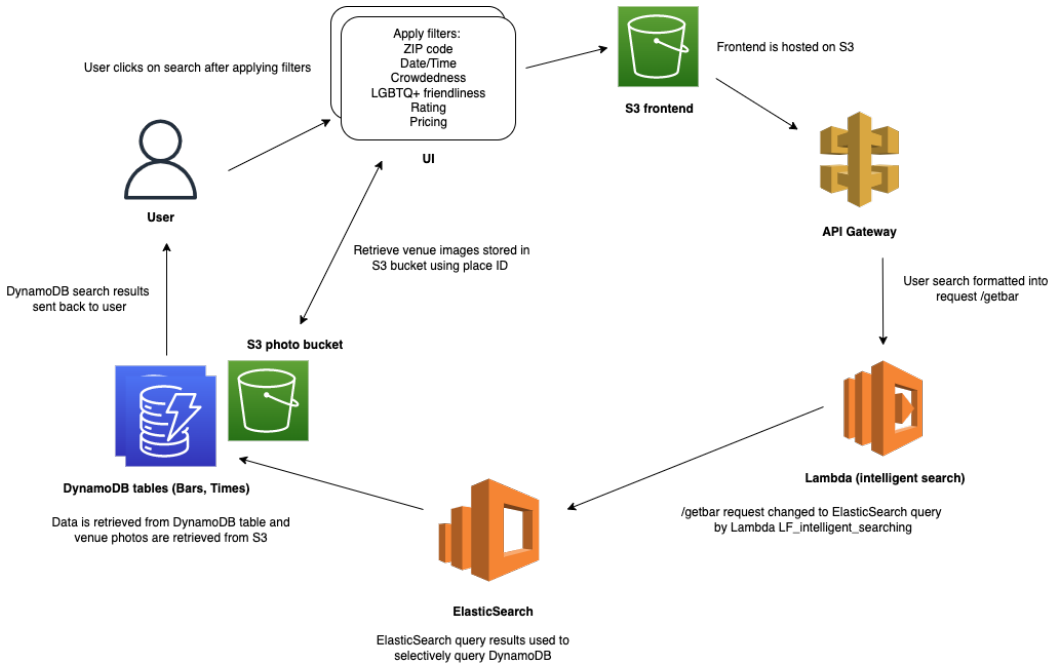
Front End Code Structure

Description:

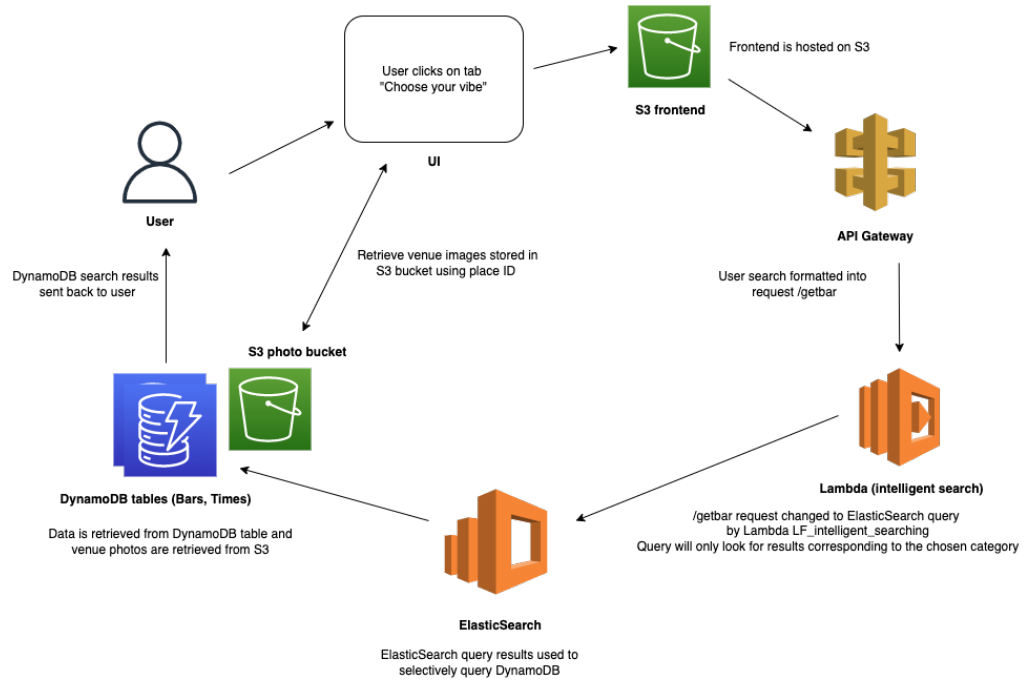
Common: All the common components that are reusable across different tabs/components.
Carousel: Carousel Settings for Top Rated Brands and Trending Section
Collection: Used to populate 'Category Types' for Choose Your Vibe Tab.
Dropdown and Dropdown Item: Used for each of the dropdown components.
ExploreSection and ExploreCard: Used to populate the individual cards displaying establishment details.
Filter and FilterItem: Used for setting up filtering components on the pages.
Footer: Footer of the website.
Header: Header of the website containing the logo and the search bar.
TabOption: Used to create 3 different tabs : Explore, Choose your Vibe and Favorites.

Explore: Used to create the Explore Tab.
TrendingCollection and TrendingItem: Used to populate Trending Places Carousel on the Explore Tab.
TopBrands: Used to populate Top Brands Carousel on the Explore Tab.
Choose Your Vibe: Used to set up the Choose Your Vibe Tab.
Favorites: Used to set up the Favorites Tab.
Note: The Backend Code Structure is suggestible through the architecture diagram itself.

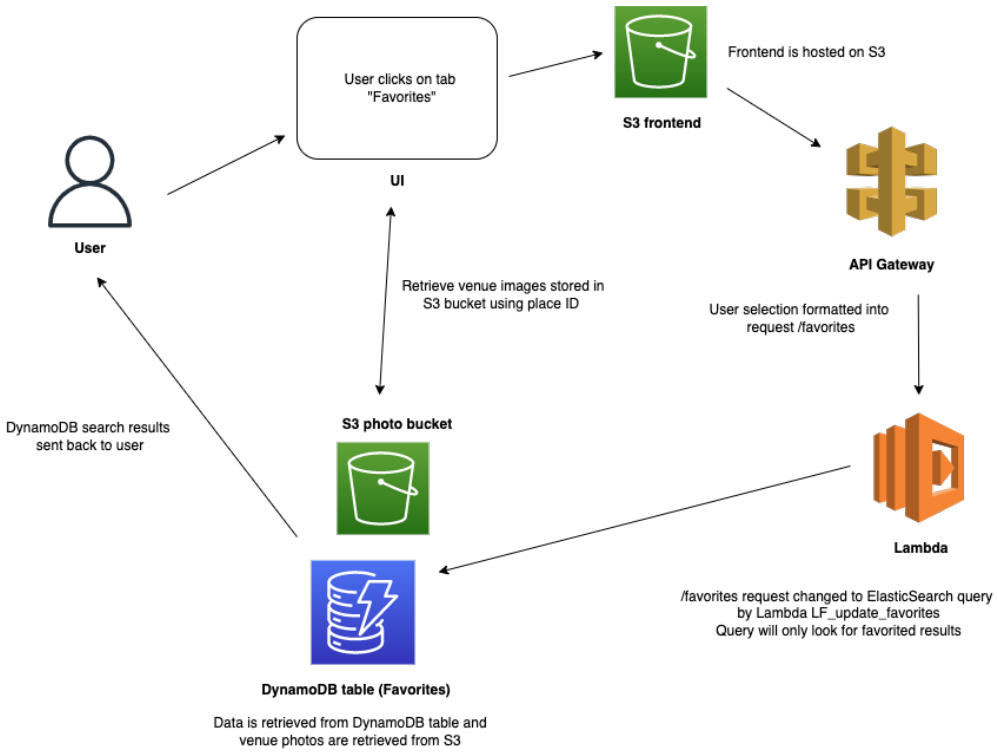
Architecture Flows



Flow 1: User searches for establishments based on zip code and applies filters



Flow 2: User looks for establishment under the ‘Choose Your Vibe/Categories’ tab



Flow 3: User looks for establishments that have been favorited

Database Model

Description:

Dynamo DB includes the following tables:

Bars:

Field	Description
place_id	Primary Key
formatted_address	Address of the establishment
formatted_phone_number	Contact Information of establishment (if available)
name	Name of establishment
opening_hours	An array of strings containing 7 elements for opening hours for 7 days of the week and 7 elements for closing hours for 7 days of the week. *Example :* ["close02200","close12200","close22200","close32200","close42200","close52300","close62300","open01600","open11600","open21600","open31600","open41600","open51600","open61600"]
opening_hours_display	An array of strings with 7 entries, displaying the opening and closing hours obtained in 'opening_hours' column. *Example :* ["Friday: 4:00 – 11:00 PM"," Monday: 4:00 – 10:00 PM", "Saturday: 4:00 – 11:00 PM", "Sunday: 4:00 – 10:00 PM", "Thursday: 4:00 – 10:00 PM", "Tuesday: 4:00 – 10:00 PM", "Wednesday: 4:00 – 10:00 PM"]
photos	ID of the photo of the establishment, which can be used to reference the photo in the Photos S3 bucket.
reviews	List of strings containing reviews of an establishment.
types	List of strings containing categorizations for an establishment as defined by Google. *Example :* ["bar"," establishment", "food"," point_of_interest", "restaurant"]
url	Google Maps URL of the establishment.
zip	Zip Code of establishment
price_level	Integer value from 1-5 denoting price range (1 being least

	expensive, and 5 being most expensive)
rating	Float value from 0-5 denoting rating of an establishment
jazz	Either 1 (Jazz) or 0 (Non-Jazz/Insufficient Data)
lgbtq	Either 1 (LGBTQ friendly) or 0 (Not LGBTQ friendly/Insufficient Data)
sports	Either 1 (Sports Bar) or 0 (Not Sports Bar/Insufficient Data)
night_club	Either 1 (Nightclub) or 0 (Not Nightclub/Insufficient Data)

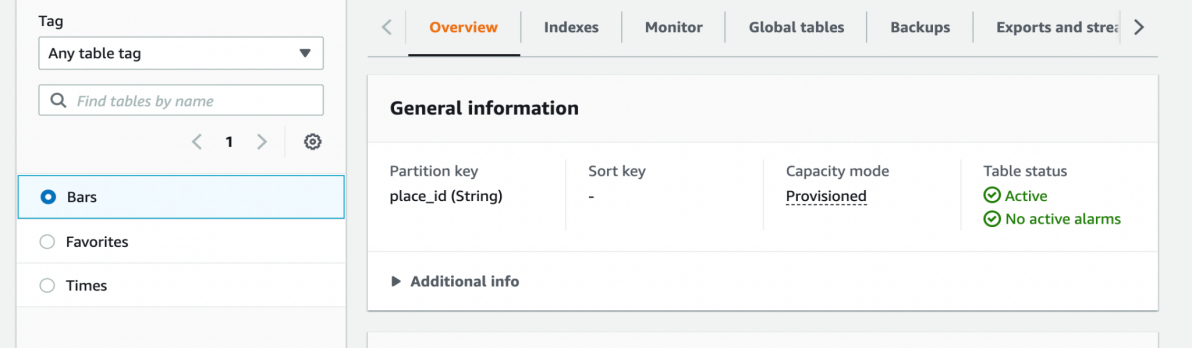
Times:

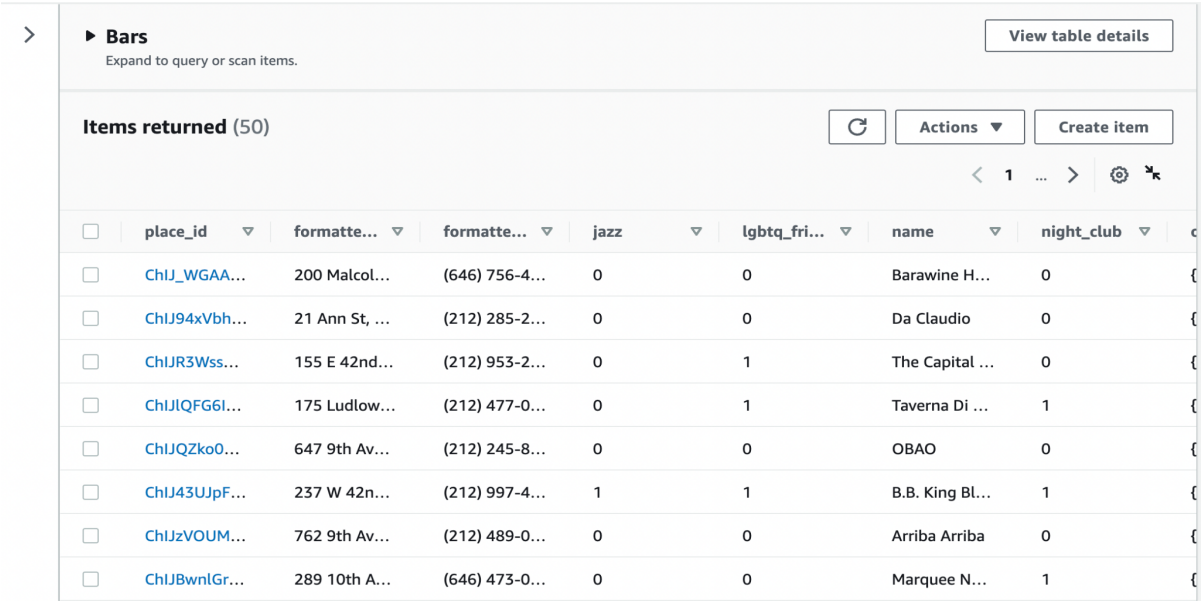
Field	Description
place_id	Primary Key
Monday	24-bit string containing values from 0-5 to indicate crowdedness, whereas 0 means closed, 1 means least crowded and 5 means most crowded for Monday. *Example :* 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 2 3 3 4 0 0
Tuesday	24-bit string containing values from 0-5 to indicate crowdedness, whereas 0 means closed, 1 means least crowded and 5 means most crowded for Tuesday.
Wednesday	24-bit string containing values from 0-5 to indicate crowdedness, whereas 0 means closed, 1 means least crowded and 5 means most crowded for Wednesday.
Thursday	24-bit string containing values from 0-5 to indicate crowdedness, whereas 0 means closed, 1 means least crowded and 5 means most crowded for Thursday.
Friday	24-bit string containing values from 0-5 to indicate crowdedness, whereas 0 means closed, 1 means least crowded and 5 means most crowded for Friday.
Saturday	24-bit string containing values from 0-5 to indicate crowdedness, whereas 0 means closed, 1 means least crowded and 5 means most crowded for Saturday.
Sunday	24-bit string containing values from 0-5 to indicate crowdedness, whereas 0 means closed, 1 means least crowded and 5 means most crowded for Sunday.

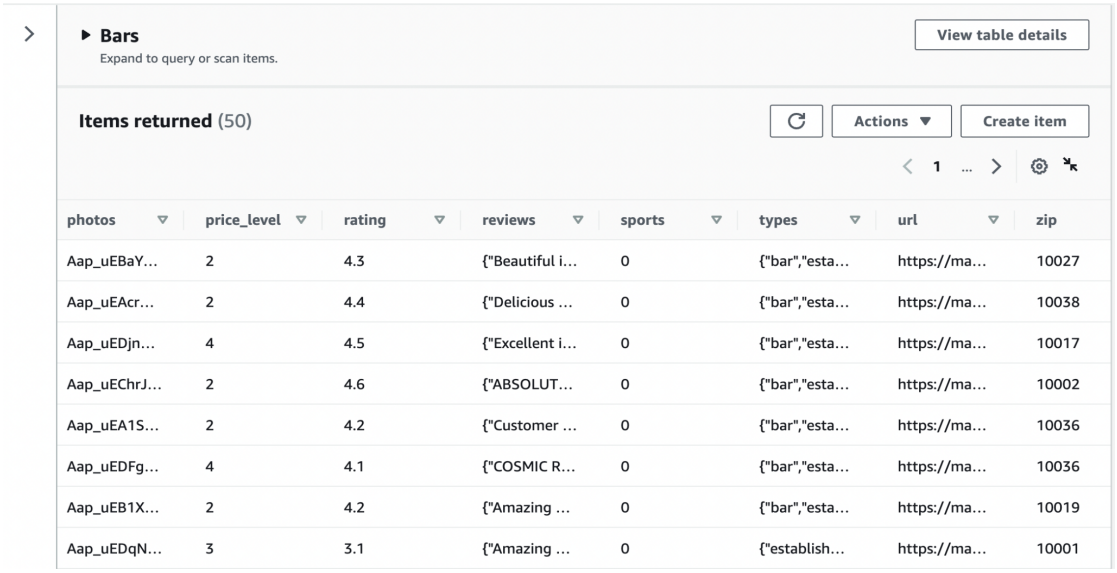
Favorites:

Field	Description
place_id	Primary Key
favorite	Either “1” or “0” indicating favorite or not favorite.

Some snapshots of the data in the DynamoDB are shown below:







Bars

Expand to query or scan items.

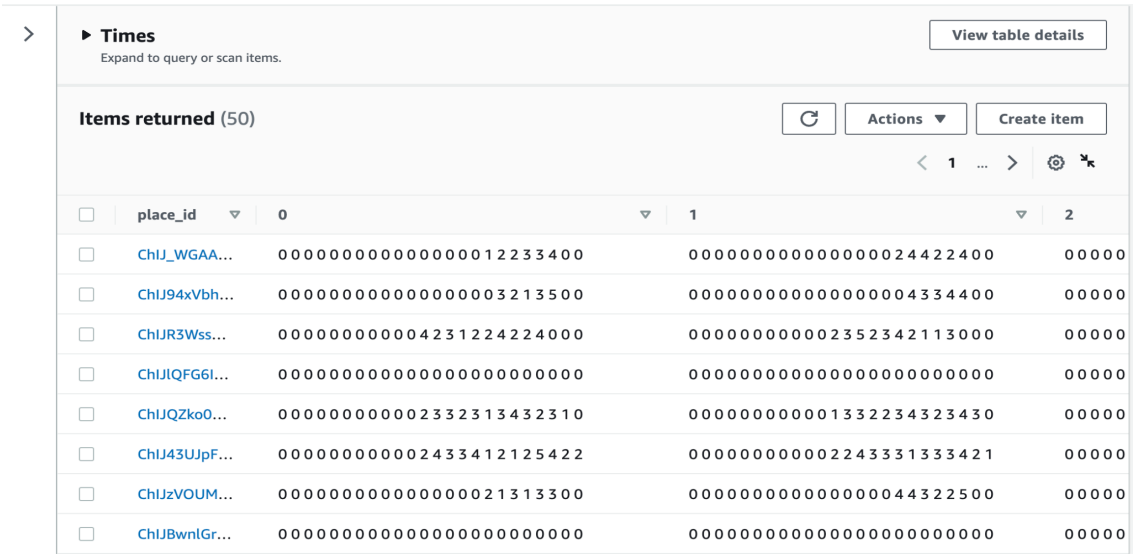
View table details

Items returned (50)

Actions | Create item

1 ...

photos	price_level	rating	reviews	sports	types	url	zip
Aap_uEBaY...	2	4.3	{"Beautiful i...	0	{"bar","esta...	https://ma...	10027
Aap_uEAcr...	2	4.4	{"Delicious ...	0	{"bar","esta...	https://ma...	10038
Aap_uEDjn...	4	4.5	{"Excellent i...	0	{"bar","esta...	https://ma...	10017
Aap_uEChrJ...	2	4.6	{"ABSOLUT...	0	{"bar","esta...	https://ma...	10002
Aap_uEA1S...	2	4.2	{"Customer ...	0	{"bar","esta...	https://ma...	10036
Aap_uEDFg...	4	4.1	{"COSMIC R...	0	{"bar","esta...	https://ma...	10036
Aap_uEB1X...	2	4.2	{"Amazing ...	0	{"bar","esta...	https://ma...	10019
Aap_uEDqN...	3	3.1	{"Amazing ...	0	{"establish...	https://ma...	10001



Times

Expand to query or scan items.

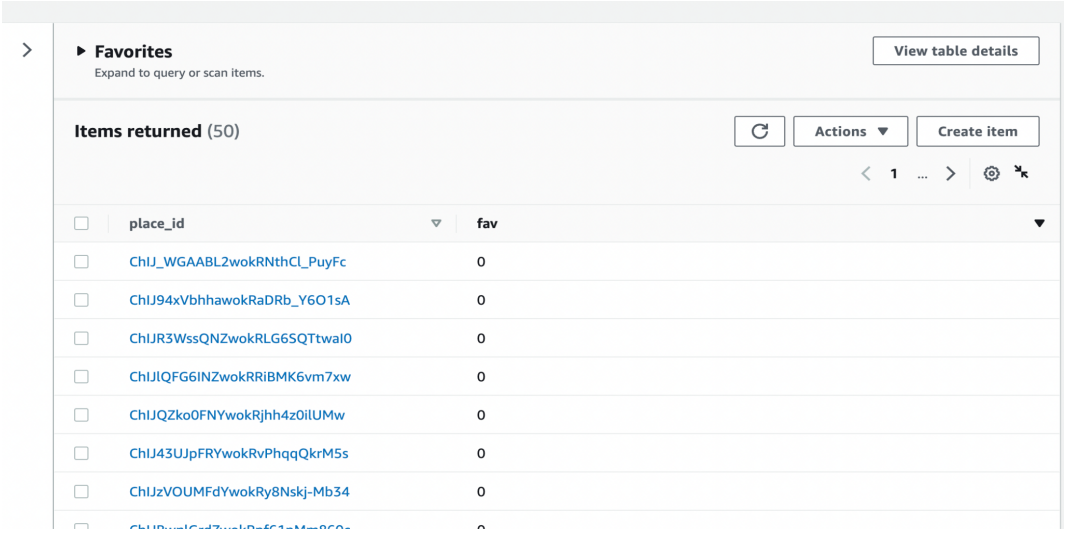
View table details

Items returned (50)

Actions | Create item

1 ...

place_id	0	1	2
ChIJ_WGAA...	0000000000000000012233400	0000000000000000024422400	00000
ChIJ94xVbh...	0000000000000000003213500	0000000000000000004334400	00000
ChIJR3Wss...	0000000000042312242240000	0000000000023523421130000	00000
ChIJlQFG6I...	0000000000000000000000000	0000000000000000000000000	00000
ChIJQZko0...	0000000000023323134323100	0000000000013322343234300	00000
ChIJ43UJpF...	0000000000024334121254220	0000000000022433313334210	00000
ChIJzVOUM...	0000000000000000021313300	0000000000000000044322500	00000
ChIJBwnlGr...	0000000000000000000000000	0000000000000000000000000	00000



Favorites

Expand to query or scan items.

View table details

Items returned (50)

Actions | Create item

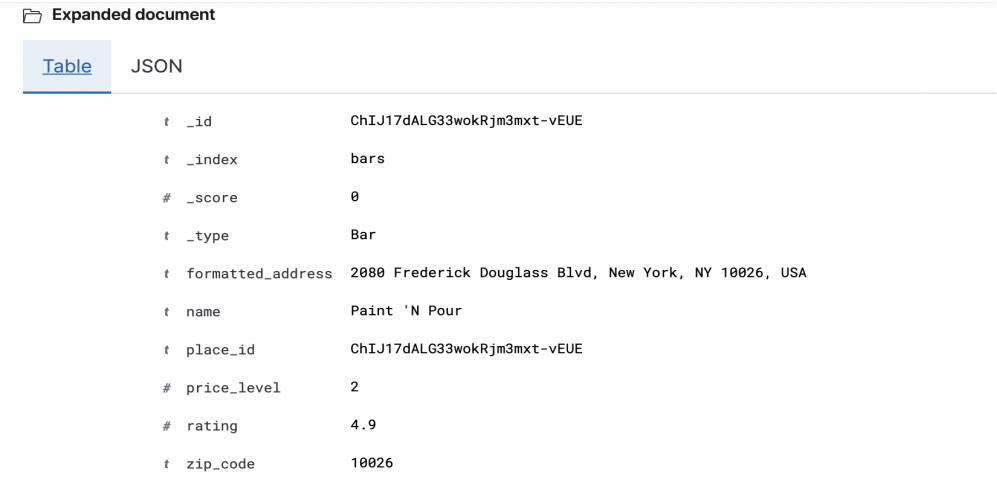
1 ...

place_id	fav
ChIJ_WGAABL2wokRNthCl_PuyFc	0
ChIJ94xVbhhawokRaDRb_Y6O1sA	0
ChIJR3WssQNZwokRLG6SQTtwaI0	0
ChIJlQFG6INZwokRRiBMK6vm7xw	0
ChIJQZko0FNYwokRjhh4z0ilUMw	0
ChIJ43UJpFRYwokRvPhqqQkrM5s	0
ChIJzVOUMFdYwokRy8Nskj-Mb34	0

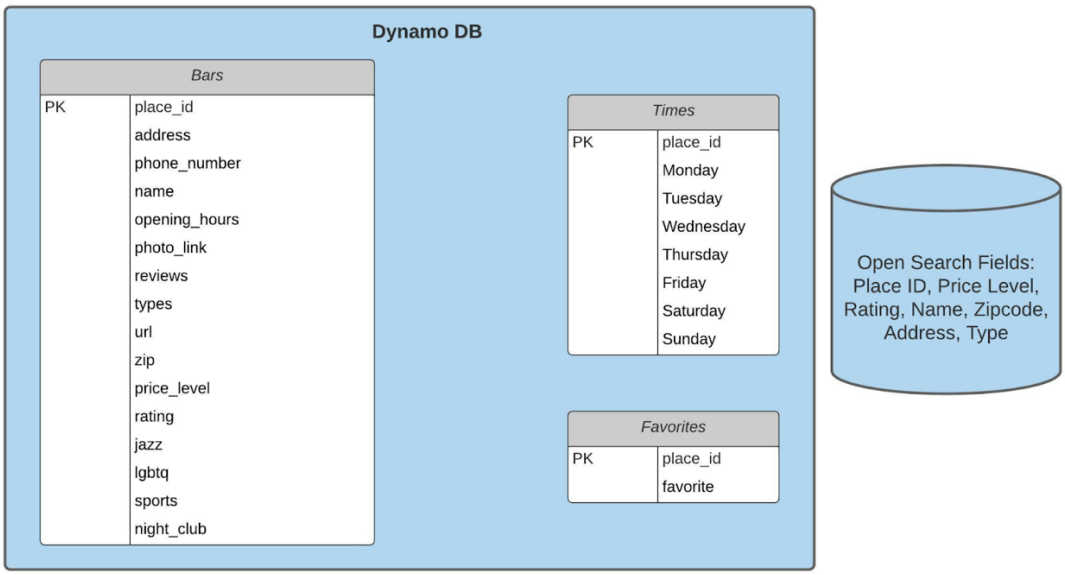
The Open Search includes the following data:

1. Place ID: This includes the bar ID and is actually the unique ID with which all the entries can be directly mapped.
2. Name: The name of the bars.
3. Price Level: This is the data that tells us how costly the place could be and is used as one of the filters in the UI.
4. Rating: This includes the place rating of the bar.
5. Zip Code: This includes the zip code of bars that is used to filter the data based on the location.

A snapshot of one of the entries in the database is shown below:



Schema:



All the relationships between the tables are one-to-one based on the place_id primary key

API Model

1. /POST /getbar:

Description:
API to fetch establishment/bar details according to applied filters.

Query String Parameters/Request:
crowd: none if no filter applied/else selected dropdown value(1-5)
date: current date in YYYY-MM-DD format/else date specified by user
lgbtq: none if no filter applied/else value of checkbox selected (false or true)
price_level: none of no filter applied/else selected dropdown value (1-4)
time: current time in HH format/else time specified by user in HH format
Zip_code: zip code of the current location of user/else zip code entered by user in the search bar.
rating: none if no filter applied/else selected dropdown value (1-4)
category: none if no category selected/value selected by user: 1 for “jazz”, 2 for “sports”, 4 for “night clubs”, “3” for lgbtq

Response Body:

```
{
        bars = [
                Bar1,
                …,
                Bar5
        ]
}
```

Example :
[{"Id": "ChIJCwnQmCT2wokRT1vH5l91_ck", "Name": "The Ellington", "Address": "2745 Broadway, New York, NY 10025, USA", "ZipCode": "10025", "Rating": "4.2", "PriceLevel": "2", "PhoneNumber": "(212) 281-3011", "LGBTQ": "0", "Url": "https://maps.google.com/?cid=14554918625485478735", "OpenHours": ["Saturday: 10:00 AM \u2013 12:00 AM", "Monday: 4:00 PM \u2013 12:00 AM", "Sunday: 10:00 AM \u2013 12:00 AM", "Wednesday: 4:00 PM \u2013 12:00 AM", "Tuesday: 4:00 PM \u2013 12:00 AM", "Friday: 4:00 PM \u2013 12:00 AM", "Thursday: 4:00 PM \u2013 12:00 AM"], "Sports": 0, "Jazz": 0, "NightClub": 0, "Crowdedness": 4}]

As you can see above, each bar object is:

```
Bar = {
        Name: string,
        Address: string,
        PhoneNumber: string,
       ID: string,
       Crowdedness: number (0-5, based on how crowded a place is),
       Jazz: bool (0 or 1),
       LGBTQ: bool (“0” or “1”),
       NightClub: bool (0 or 1),
       OpenHours: Array of Strings with 7 entries containing opening and closing
hours    for each day of the week.

        PriceLevel: number (“1”, “2”, “3” or “4” from least to most expensive)
        Rating: float (0-5)
        Sports: bool (0 or 1),
        URL: string (google maps URL of the establishment)
        ZipCode: string
}
```

2. /GET /favorite:

Description:

Fetches a list of all the user’s favorite establishments.

Response Body:

Same as /POST /getbar

Example:

[{“Id”: “ChIJN-93ZsBZwokRonoT5y_yB1c”, “Name”: “The Standard Grill”, “Address”: “The Standard, High Line, 848 Washington St, New York, NY 10014, USA”, “ZipCode”: “10014”, “Rating”: “4.1”, “PriceLevel”: “3”, “PhoneNumber”: “(212) 645-4100”, “LGBTQ”: “0”, “Url”: “https://maps.google.com/?cid=6271247293690903202”, “OpenHours”: [“Tuesday: 7:00 AM \u2013 12:00 AM”, “Friday: 7:00 AM \u2013 12:00 AM”, “Monday: 7:00 AM \u2013 12:00 AM”, “Thursday: 7:00 AM \u2013 12:00 AM”, “Sunday: 7:00 AM \u2013 12:00 AM”, “Wednesday: 7:00 AM \u2013 12:00 AM”, “Saturday: 7:00 AM \u2013 12:00 AM”], “Sports”: 0, “Jazz”: 0, “NightClub”: 0, “Crowdedness”: “5”}, {“Id”: “ChIJu55raJRZwokReRL9b1tLmQo”, “Name”: “Smalls Jazz Club”, “Address”: “183 W 10th St, New York, NY 10014, USA”, “ZipCode”: “10014”, “Rating”: “4.6”, “PriceLevel”: “2”, “PhoneNumber”: “N/A”, “LGBTQ”: “0”, “Url”: “https://maps.google.com/?cid=76372446790897317 7”, “OpenHours”: [“Sunday: Closed”, “Monday: 5:30 PM \u2013 12:00 AM”, “Saturday: 5:30 PM \u2013 12:00 AM”, “Wednesday: 5:30 PM \u2013 12:00 AM”, “Thursday: 5:30 PM \u2013 12:00 AM”, “Tuesday: 5:30 PM \u2013 12:00 AM”, “Friday: 5:30 PM \u2013 12:00 AM”], “Sports”: 0, “Jazz”: 1, “NightClub”: 1, “Crowdedness”: “4”}, {“Id”: “ChIJGy8Umr9ZwokRVas9YjRxMfo”, “Name”: “Dos Caminos”, “Address”: “675 Hudson St, New York, NY

10014, USA", "ZipCode": "10014", "Rating": "4", "PriceLevel": "2", "PhoneNumber": "(212) 699-2400", "LGBTQ": "0", "Url": "https://maps.google.com/?cid=18028315253141252949", "OpenHours": ["Sunday: 11:30 AM \u2013 9:00 PM", "Saturday: 11:30 AM \u2013 10:00 PM", "Tuesday: 11:30 AM \u2013 9:00 PM", "Thursday: 11:30 AM \u2013 9:00 PM", "Friday: 11:30 AM \u2013 10:00 PM", "Wednesday: 11:30 AM \u2013 9:00 PM", "Monday: 11:30 AM \u2013 9:00 PM"], "Sports": 0, "Jazz": 0, "NightClub": 0, "Crowdedness": "0"}, {"Id": "ChIJjY7y6ZJZwokRXF5CA_X4l-I", "Name": "LB", "Address": "20 7th Ave S, New York, NY 10014, USA", "ZipCode": "10014", "Rating": "4.5", "PriceLevel": "3", "PhoneNumber": "(212) 929-4360", "LGBTQ": "0", "Url": "https://maps.google.com/?cid=16327792705260379740", "OpenHours": ["Friday: 6:00 PM \u2013 2:00 AM", "Thursday: 6:00 PM \u2013 1:00 AM", "Tuesday: 6:00 PM \u2013 12:00 AM", "Wednesday: 6:00 PM \u2013 1:00 AM", "Sunday: 6:00 PM \u2013 12:00 AM", "Saturday: 6:00 PM \u2013 2:00 AM", "Monday: 6:00 PM \u2013 12:00 AM"], "Sports": 0, "Jazz": 1, "NightClub": 0, "Crowdedness": "5"}, {"Id": "ChIJ2_DOLm32wokRLmQMP8n07nM", "Name": "Red Lobster", "Address": "261 W 125th St, New York, NY 10027, USA", "ZipCode": "10027", "Rating": "3.8", "PriceLevel": "2", "PhoneNumber": "(212) 280-1930", "LGBTQ": "0", "Url": "https://maps.google.com/?cid=8353883504002229294", "OpenHours": ["Friday: 11:00 AM \u2013 11:00 PM", "Sunday: 11:00 AM \u2013 10:00 PM", "Tuesday: 11:00 AM \u2013 10:00 PM", "Saturday: 11:00 AM \u2013 11:00 PM", "Wednesday: 11:00 AM \u2013 10:00 PM", "Thursday: 11:00 AM \u2013 10:00 PM", "Monday: 11:00 AM \u2013 10:00 PM"], "Sports": 0, "Jazz": 0, "NightClub": 0, "Crowdedness": "2"}, {"Id": "ChIJzYwpP5RzwokRk_zUBWseSBY", "Name": "Mezzrow", "Address": "163 W 10th St, New York, NY 10014, USA", "ZipCode": "10014", "Rating": "4.6", "PriceLevel": "2", "PhoneNumber": "N/A", "LGBTQ": "0", "Url": "https://maps.google.com/?cid=16055667121657601 47", "OpenHours": ["Tuesday: 7:00 PM \u2013 12:30 AM", "Monday: Closed", "Thursday: 7:00 PM \u2013 12:30 AM", "Friday: 7:00 PM \u2013 12:30 AM", "Saturday: 7:00 PM \u2013 12:30 AM", "Wednesday: 7:00 PM \u2013 12:30 AM", "Sunday: 7:00 PM \u2013 12:30 AM"], "Sports": 0, "Jazz": 1, "NightClub": 1, "Crowdedness": "5"}, {"Id": "ChIJ5S6h_19ZwokRXWMbYXSG0Q0", "Name": "The Patio NYC", "Address": "12 Little W 12th St, New York, NY 10014, USA", "ZipCode": "10014", "Rating": "4.4", "PriceLevel": "4", "PhoneNumber": "(347) 921-0002", "LGBTQ": "0", "Url": "https://maps.google.com/?cid=9957248270291034 53", "OpenHours": ["Sunday: 12:00 \u2013 11:00 PM", "Tuesday: Closed", "Monday: Closed", "Friday: 5:00 PM \u2013 4:00 AM", "Saturday: 5:00 PM \u2013 4:00 AM", "Wednesday: 5:00 PM \u2013 4:00 AM", "Thursday: 5:00 PM \u2013 12:00 AM"], "Sports": 0, "Jazz": 0, "NightClub": 1, "Crowdedness": "5"}]

3. /POST /favorite

Description:

To add or remove an establishment from user's favorites list.

Query String Parameters/Request:

Id: Place Id of the establishment
isfavorite: 1 if the place is added to the favorites list or 0 if it's removed from the favorites list.

Response Body:

No response.

Features Implemented

1. Zip Code based search query for finding establishments in that particular area.

2. Filtering of establishments based on LGBTQ friendliness, ratings, price range.

3. Filtering of establishments based on crowdedness level, date of visit and time of visit (selling point)

4. Categorization of establishments into ‘Trending Places’ (based on their crowdedness level: establishments that are fairly crowded are placed into this category)

5. Categories of establishments into ‘Top Rated Brands’ (based on their ratings: establishments that are rated 4 and above are placed into this category)

6. Displaying search results in the form of intuitive and easy to read cards with all the relevant information like Name, Price Range, Rating, Contact Number and Crowdedness Level.

7. Embedded Google Maps link in each card to take the user to the google maps link corresponding to the establishment.

8. Categorization of establishments into ‘trendy’ categories like LGBTQ bars, Jazz Bars, Sports Bars and Nightclubs for easy access.

9. Feature to add or remove establishments from ‘Favorites’ : This will enable users to have a consolidated list of all the places that they have liked and will help them make faster and efficient decisions, the next time they plan an outing.

Technical Challenges

- Retrieving categorization data for fields like “categories” and “lgbtq”: Wading through large datasets for insights.

- Mapping the data from times and bars data to cater to suit our use case and scenarios which required a lot of reformatting to enable easier filtering.

- Designing the UI in the most intuitive way possible.

- Integrating and hosting the UI on AWS Cloud.

- Designing a database to maximize performance and efficiency:
    - S3 (for photos)
    - DynamoDB (3 tables)
    - OpenSearch (for filtering criteria)

Conclusion and Future scope

DrinkEasy achieved most of the goals we initially had in mind and went even further in its implementation.

For the backend, we were able to get the desired data and reformat and restructure for our use case. We obtained a good database with filtered data which is easy to understand and use. For the frontend, we created an intuitive UI for the “Party People”. While it is possible to search for bars and different categories of bars, it is also possible to view venues that other people have favorited and get inspiration.

Potential future improvements include user logins and CloudWatch triggers. The former would enable users to save their data and plan far in advance for future events. The latter would allow the database to be updated automatically every 30 days, thus reflecting seasonal differences in crowdedness levels. Finally, it would be possible to add a “COVID safe” filter based on crowdedness level and venue square footage data to reassure users.

Relevant Links

FINAL PROJECT (S3 BUCKET):
http://frontend-drinkeasy.s3-website-us-east-1.amazonaws.com/

FINAL PROJECT PRESENTATION:
https://docs.google.com/presentation/d/1cyBxY8F7ydYHqPSRbpgQEb76znPw9xrnyQZO1q31uHY/edit?usp=sharing

FINAL PROJECT DEMO VIDEO (YOUTUBE):
https://youtu.be/d2EJh02CfHk

MVP DEMO VIDEO (YOUTUBE):
https://youtu.be/7BH80bY5kTs

CLICKABLE PROTOTYPE:
https://invis.io/U811Y33I73NC#/460403355_DefaultScreen
Password To Access : drinkeasy

PROJECT PROPOSAL DOCUMENT:
https://docs.google.com/document/d/1np7KKFEqMSLxFfG2kILgu5l-_I6lFygEr_JcnhzwzWw/edit

Also, all of the files have been submitted as a zip with the submission.