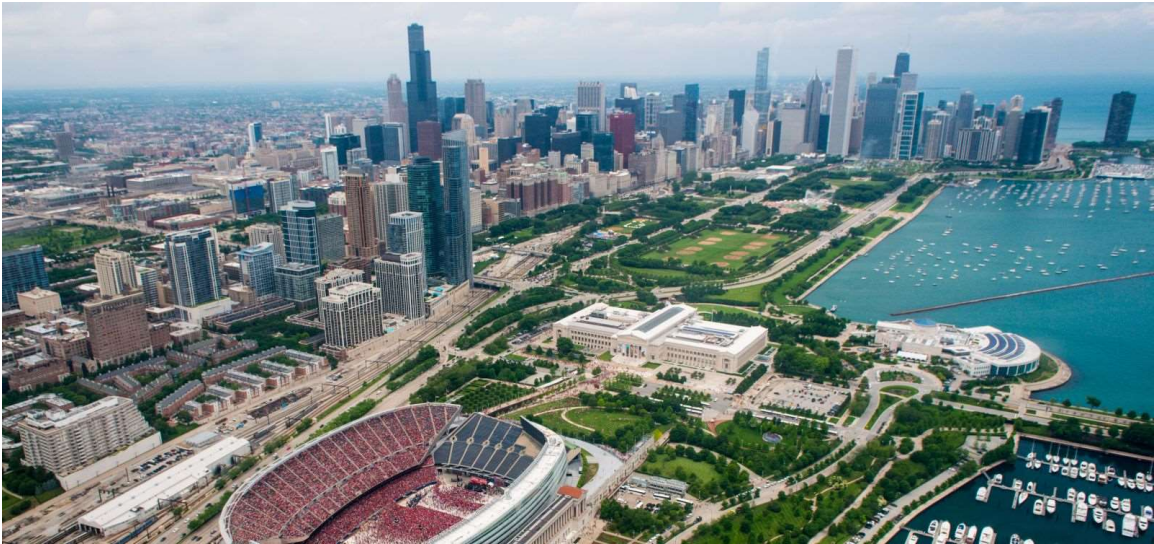


ChicagoSocialHub

Real-Time Web-App



Project Overview Statement:

In this project we will design and implement a web-based real-time application for Divvy bikers that will allow them to search and chart Yelp reviewed Chicago social places (restaurants, bars, coffee shops, etc) and plot real-time available docks in near-by Divvy docking stations on Chicago downtown area map; we will call our application **ChicagoSocialHub**. The following sections document the technologies, data sources, and the detailed requirements

Technologies and Platforms:

The following is the list of technologies, platforms, and packages used in the design and development of the **ChicagoSocialHub** real-time app

1. Angular 7 or higher
2. Node.js/Express
3. D3.js
4. Google Maps (AGM)
5. PostgreSQL – to store Divvy stations real-time status
6. ElasticSearch – to store Yelp reviews for Chicago Businesses
7. ElasticSearch – to create and store Divvy real-time logs and store Divvy trips anonymized data
8. Chrome browser that is compliant with ECMAScript 2015 scripting 2015, (ES6): <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes>. List of browsers/platforms that support ES6 can be found under **modern browsers** link on this URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

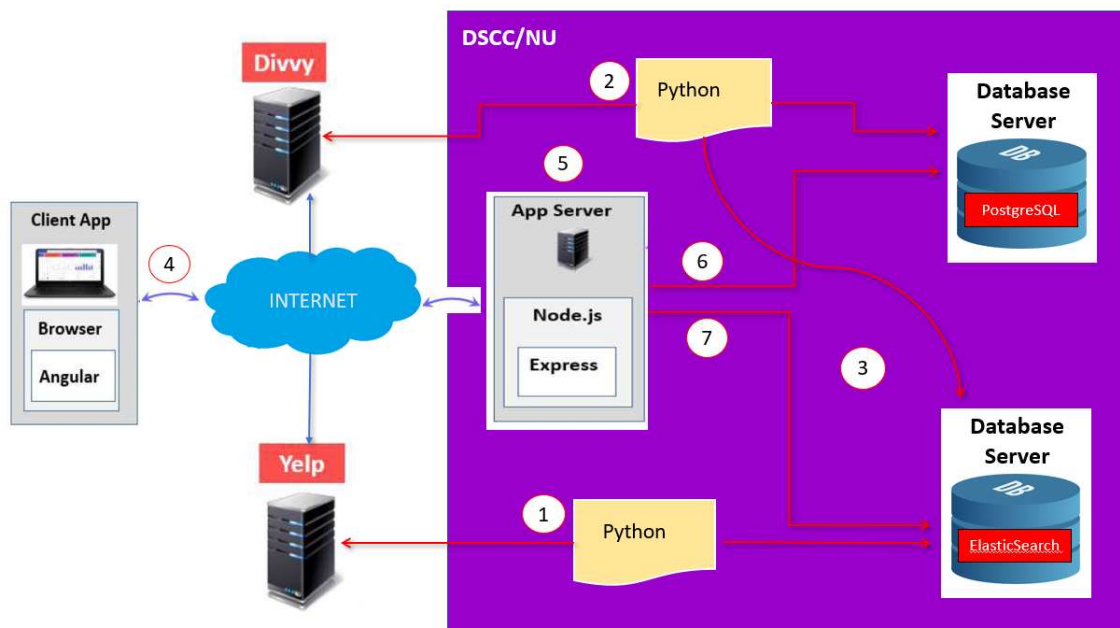
Data Sources:

For this project there are two data sources that we will use for our application:

1. We will use **Yelp** business reviews (See Appendix A) to make recommendations for restaurants in Chicago downtown area that are highly reviewed and got at least 3-stars on **Yelp**.
 - a. Here is the URL for Yelp API:
https://www.yelp.com/developers/documentation/v3/get_started
 - b. Here is the URL for businesses search:
https://www.yelp.com/developers/documentation/v3/business_search
 - c. Here is the URL for the supported search categories:
https://www.yelp.com/developers/documentation/v3/all_category_list
2. We will use **Divvy** real-time data (See Appendix B) about the status of their docking stations
 - a. Here is the URL for the real-time data they publish:
<https://gbfs.divvybikes.com/gbfs/gbfs.json>
 - b. Here is the URL for the real-time status for the docking stations:
https://gbfs.divvybikes.com/gbfs/en/station_status.json
 - c. Here is the URL for the information for the docking stations:
https://gbfs.divvybikes.com/gbfs/en/station_information.json
 - d. Here is the URL for the anonymized data for Divvy trips:
<https://www.divvybikes.com/system-data>

Architecture and High-Level Design:

The **ChicagoSocialHub** real-time web-app has the architecture detailed in the following diagram. Python scripts pulls data from Yelp and Divvy and store the data on PostgreSQL server and ElasticSearch server hosted on DSCC. Node.js/Express server receives requests from Angular clients and access the database servers to collect data and send the data back to Angular clients.

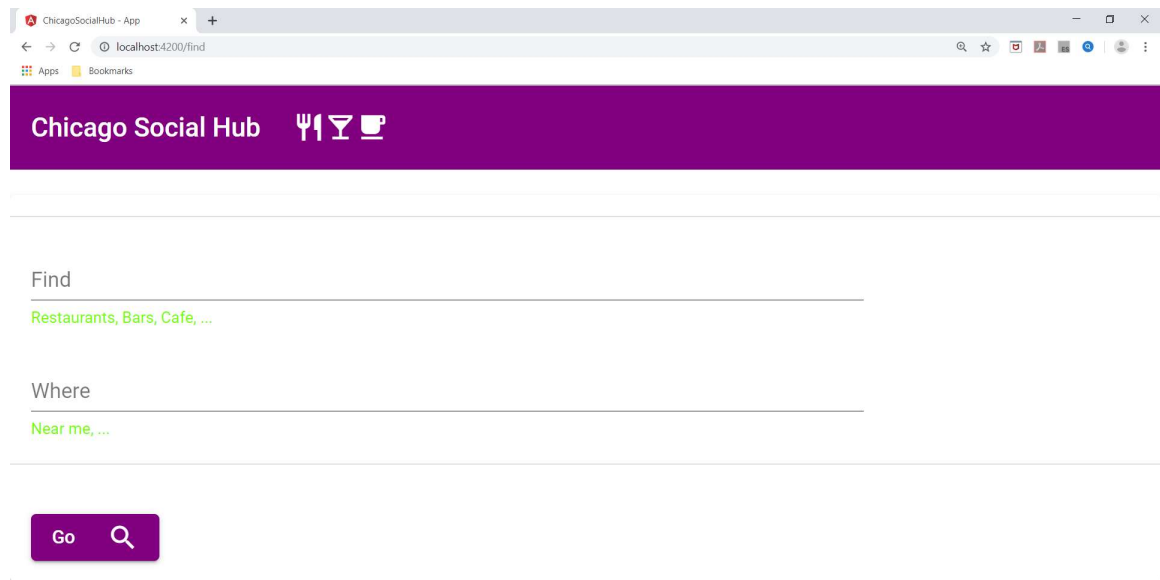


Requirements specification:

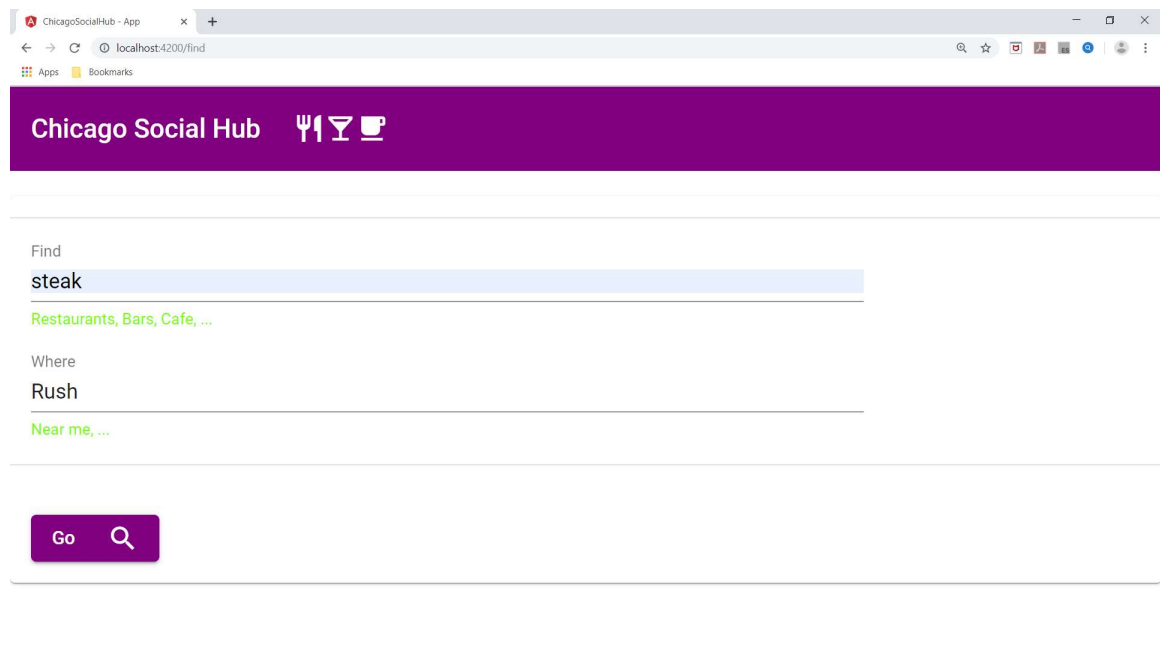
You will design and develop a web-based real-time application that meets the following requirements:

1. Develop the infrastructure for the **ChicagoSocialHub** real-time web-app utilizing **Yelp** and **Divvy** APIs.
2. Use Node.js/Express to create the back-end application server
3. Use PostgreSQL and ElasticSearch for SQL and NoSQL databases to retrieve data
4. Use Google Maps to plot data for the Geospatial queries
5. Divvy bikers would like to search for places located on certain streets in Chicago downtown area and plot divvy nearest docking stations for a selected place on Google Map for Chicago downtown area.
6. Use Angular for to create the front-end (client-side) application

7. The app-user shall be able to specify the search conditions using two fields to represent the pair (business-category, street-name), for example, Italian on Wabash, steak on Rush, pizza on Michigan, etc. Your web-page shall look as follows:



The screenshot shows a web browser window with the title "ChicagoSocialHub - App". The address bar shows "localhost:4200/find". The page has a purple header with the text "Chicago Social Hub" and icons for a fork, a glass, and a cup. Below the header, there are two input fields. The first field is labeled "Find" and has a dropdown menu with the option "Restaurants, Bars, Cafe, ...". The second field is labeled "Where" and has a dropdown menu with the option "Near me, ...". Below the input fields, there is a purple button with the text "Go" and a magnifying glass icon.



The screenshot shows the same web browser window as the previous one, but with the search criteria filled in. The "Find" field now has the text "steak" entered, and the "Where" field now has the text "Rush" entered. The dropdown menus are still open, showing the same options as before. The "Go" button remains at the bottom.

8. The search results for top rated Yelp-reviewed places based on the specified filter by the app-user shall be displayed on a web-page.

ChicagoSocialHub - App

localhost:4200/list_of_places

Apps

Bookmarks

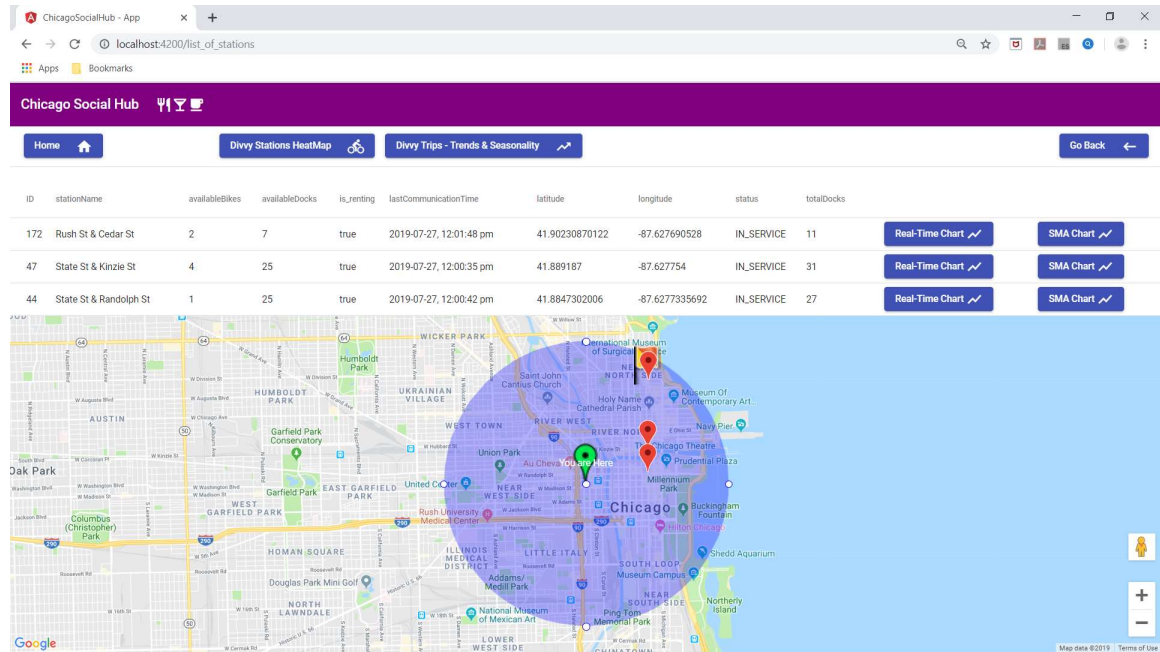
Chicago Social Hub

Home

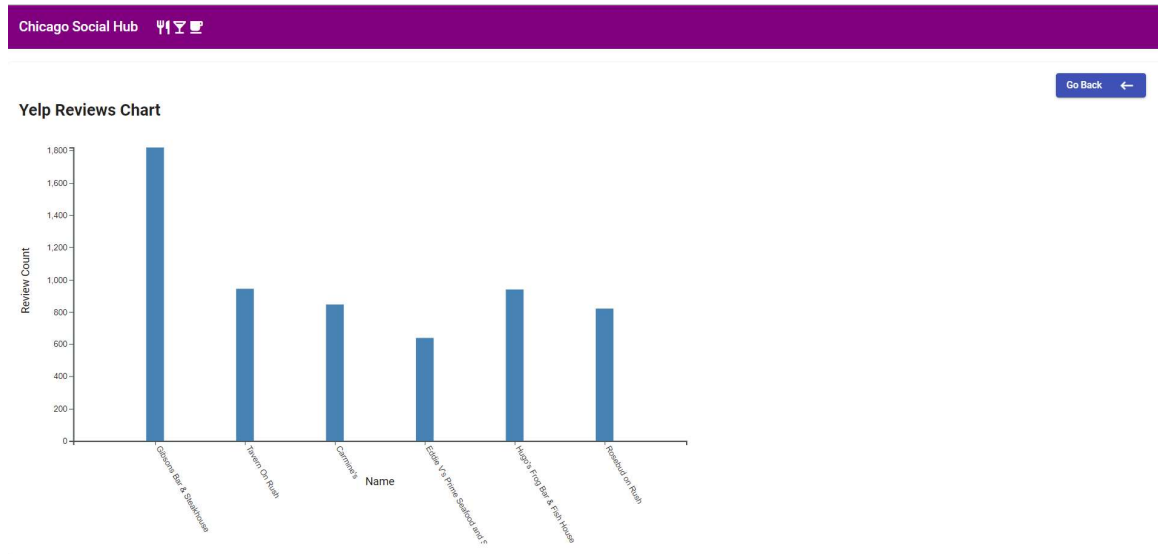
Yelp Reviews Chart

Name	Phone	Address	Closed	Rating	Review Count	
Gibsons Bar & Steakhouse	(312) 266-8999	1028 N Rush St	false	4	1818	Divvy Near by
Tavern On Rush	(312) 664-9600	1031 N Rush St	false	3.5	944	Divvy Near by
Carmine's	(312) 988-7676	1043 N Rush St	false	3.5	844	Divvy Near by
Eddie V's Prime Seafood and Steaks	(312) 595-1114	521 N Rush St	false	4.5	640	Divvy Near by
Hugo's Frog Bar & Fish House	(312) 640-0999	1024 North Rush	false	4	940	Divvy Near by
Rosebud on Rush	(312) 266-6444	720 N Rush St	false	4	820	Divvy Near by

9. The app-user shall be provided with the capability to view the real-time status for the near-by Divvy docking stations of the selected place along with Google map for the current location, the selected place location, and the nearest 3 Divvy docking stations of the selected place.



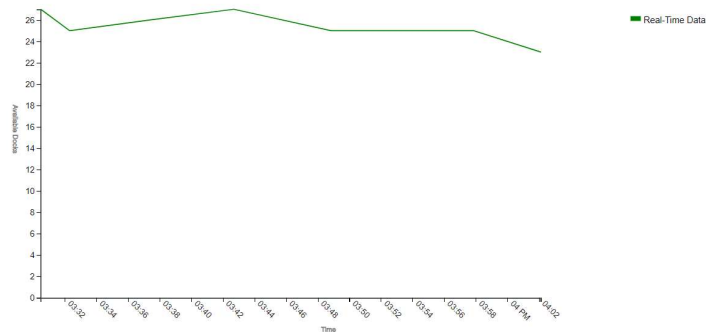
10.The app-user shall be provided with the capability to view the bar-chart for yelp reviews.



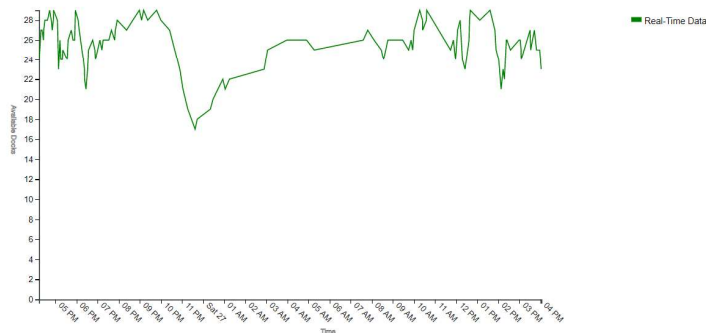
11. The app-user shall be provided with the capability to view the real-time line-chart for the status of the selected Divvy docking station. The Line-chart shall provide the app-user with the capability to select the time-range: past hour, past 24 hours, past 7 days data.



Divvy Dock Station: State St & Kinzie St



Divvy Dock Station: State St & Kinzie St



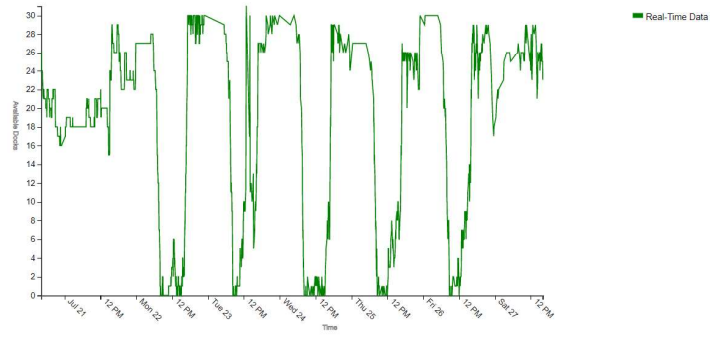


Home

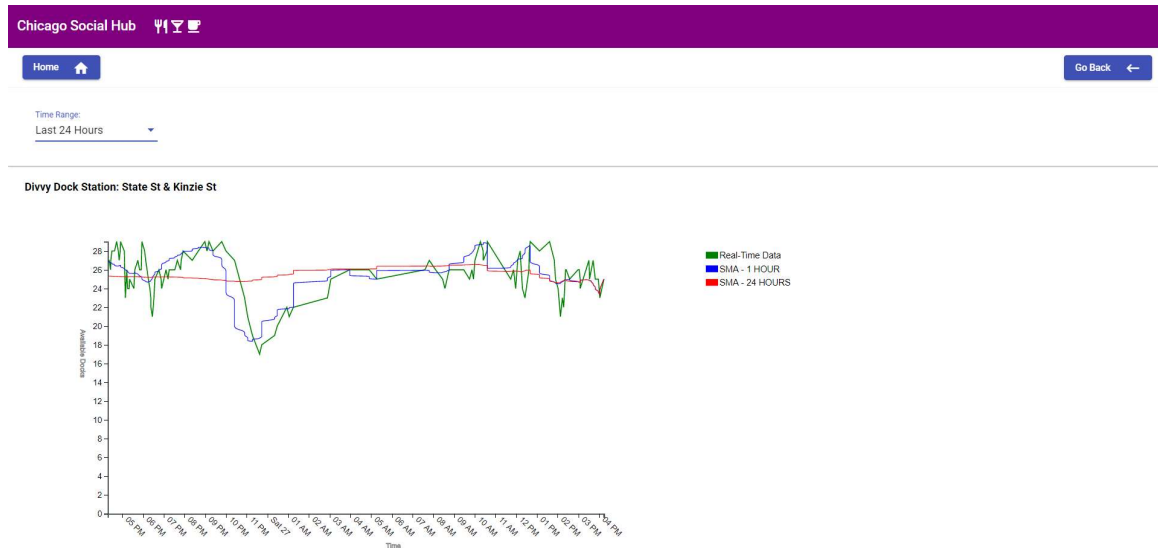
Go Back

Time Range:
Last 7 Days

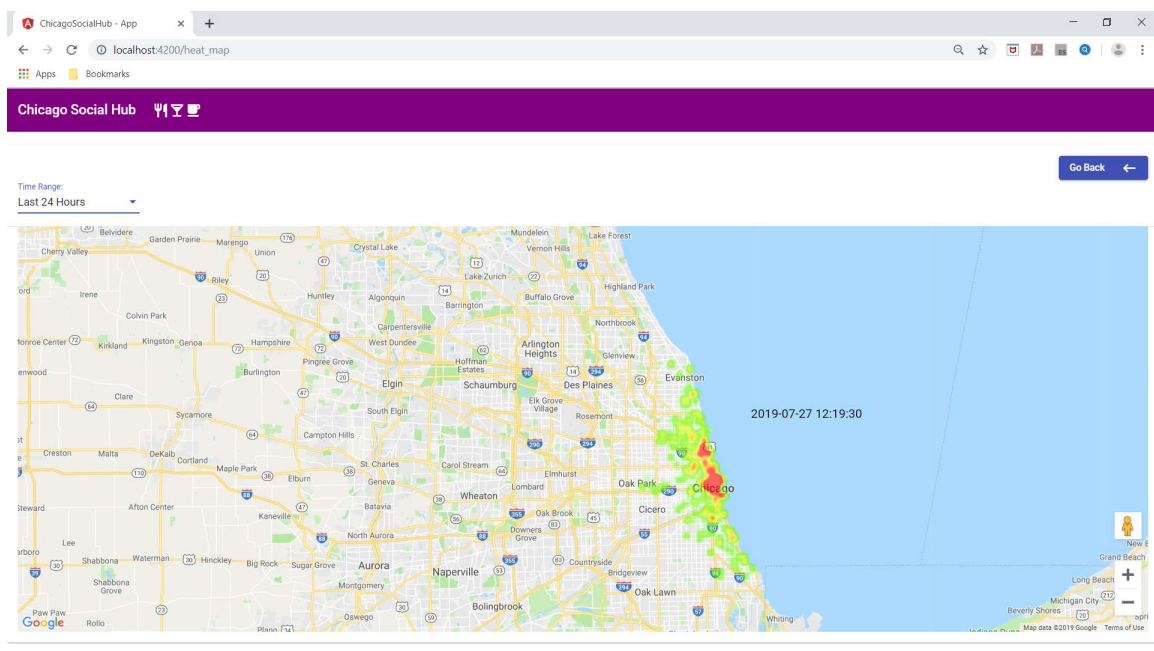
Divvy Dock Station: State St & Kinzie St

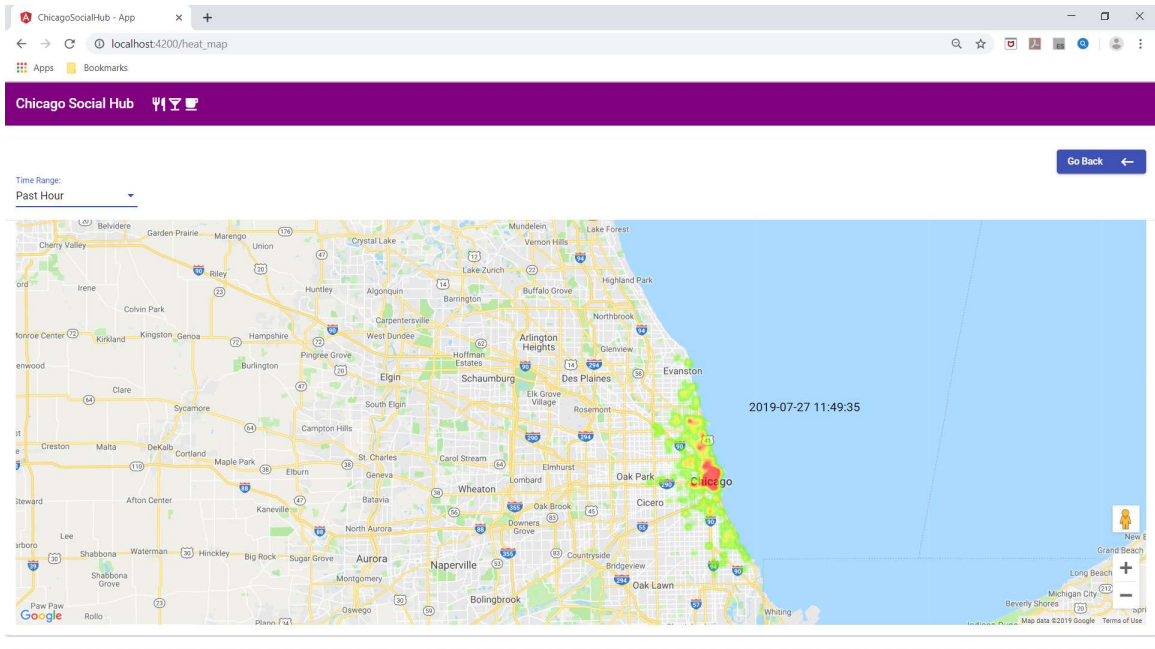


12. The app-user shall be provided with the capability to view the Simple Moving Average (past hour and past 24 hours) in a real-time line-chart for the status of the selected docking station. The Line-chart shall provide the app-user with the capability to select the time-range: past hour, past 24 hours, past 7 days data.

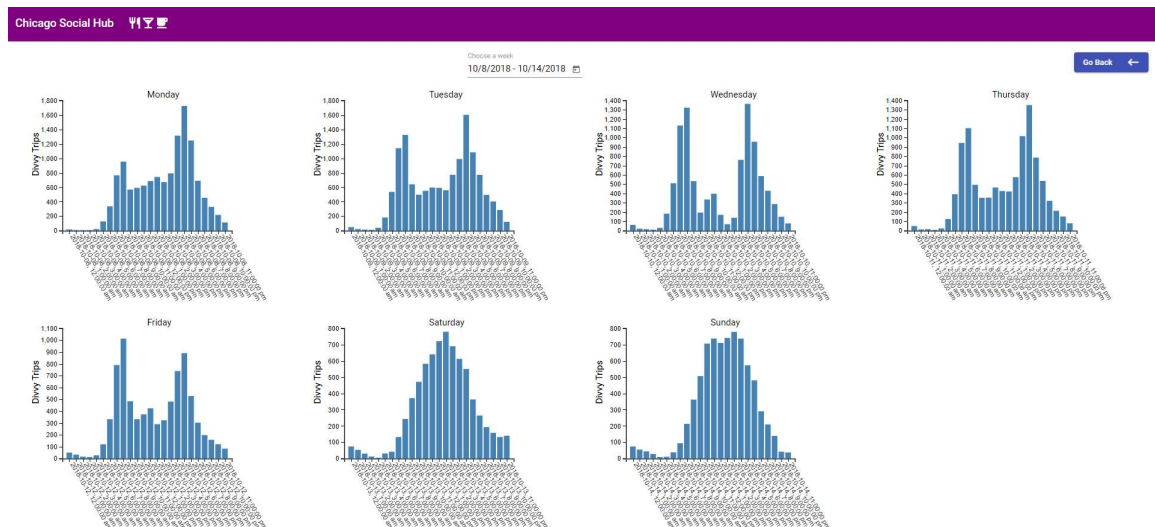


13. The real-time line-chart for the status of the selected Divvy docking station shall be updated/refreshed automatically every 2 minutes without the app-user manual refresh. The line-chart shall provide the app-user with the capability to select the time-range: past hour, past 24 hours, past 7 days data.
14. Create real-time Heatmaps for Divvy docking stations. Divvy can utilize the HeatMap to decide at which stations to place a valet (<https://www.chicagotribune.com/redeye/redeye-divvy-stations-locations-20140901-story.html>) to make sure riders have spots to dock at stations popular for drop-offs
15. The app-user shall be provided with the capability to view the Heatmap in a real-time for the status of the docking stations for the entire city of Chicago. The Heatmap shall provide the app-user with the capability to select the time-range: past hour, past 24 hours, past 7 days data.






16. The app-user shall be provided with the capability to use Divvy anonymized log data to view the count of Divvy hourly trips for every day of the week in order to analyze the chart patterns: double tops bar-chart for morning rush-hours and evening rush-hours. And the bell-curve bar-chart for the weekends or holiday days.



Appendix A. Yelp Business Reviews



Fusion

Fusion API

GraphQL

Manage App

Log In

Sign Up

General

Create App

Email / Notifications

Display Requirements

Terms of Use

FAQ

Yelp Fusion

Introduction

Get Started

Daily Access Limiting

QPS Rate Limiting

Code Samples

Authentication

Additional Resources

Get started with the Yelp Fusion API

The Yelp Fusion API allows you to get the best local content and user reviews from millions of businesses across 32 countries. This tutorial provides an overview of the capabilities our suite of APIs offer, provides instructions for how to authenticate API calls, and walks through a simple scenario using the API.

Authentication

The Yelp Fusion API uses private key authentication to authenticate all endpoints. Your private API Key will be automatically generated after you create your app. For detailed instructions, refer to our [authentication guide](#).

Endpoints

All Yelp Fusion API endpoints are under <https://api.yelp.com/v3>. Below are Yelp Fusion's current endpoints. Click the links for detailed documentation. You can also try it out by yourself using [Postman](#)

▶ Run in Postman

You can also find small code samples for Node.js, Python, Ruby, and other languages at [Yelp Fusion Code Samples on Github](#).

Name	Path	Description
Business Search	/businesses/search	Search for businesses by keyword, category, location, price level, etc.
Phone Search	/businesses/search/phone	Search for businesses by phone number.
Transaction Search	/transactions/{transaction_type}/search	Search for businesses which support food delivery transactions.
Business	/businesses/{id}	Get rich business data, such as name, address, phone

▶ Run in Postman

You can also find small code samples for Node.js, Python, Ruby, and other languages at [Yelp Fusion Code Samples on Github](#).

Name	Path	Description
Business Search	/businesses/search	Search for businesses by keyword, category, location, price level, etc.
Phone Search	/businesses/search/phone	Search for businesses by phone number.
Transaction Search	/transactions/{transaction_type}/search	Search for businesses which support food delivery transactions.
Business Details	/businesses/{id}	Get rich business data, such as name, address, phone number, photos, Yelp rating, price levels and hours of operation.
Business Match	/businesses/matches	Find the Yelp business that matches an exact input location. Use this to match business data from other sources with Yelp businesses.
Reviews	/businesses/{id}/reviews	Get up to three review excerpts for a business.
Autocomplete	/autocomplete	Provide autocomplete suggestions for businesses, search

General

Create App

Email / Notifications

Display Requirements

Terms of Use

FAQ

Yelp Fusion

Introduction

Business Endpoints

Business Search

Phone Search

Transaction Search

Business Details

/businesses/search

This endpoint returns up to 1000 businesses based on the provided search criteria. It has some basic information about the business. To get detailed information and reviews, please use the Business ID returned here and refer to [/businesses/{id}](#) and [/businesses/{id}/reviews](#) endpoints.

Note: at this time, the API does not return businesses without any reviews.

Request

GET <https://api.yelp.com/v3/businesses/search>

Parameters

These parameters should be in the query string.

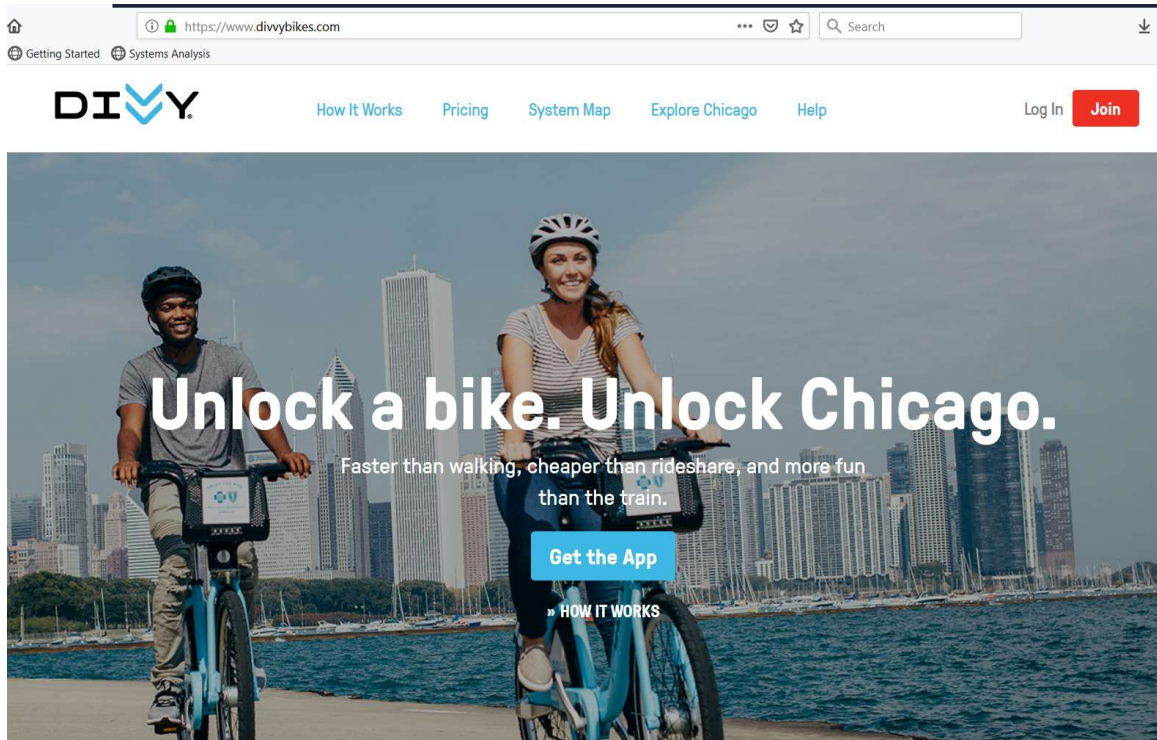
Name	Type	Description
term	string	Optional. Search term, for example "food" or "restaurants". The term may also be business names, such as "Starbucks". If term is not included the endpoint will default to searching across businesses from a small number of popular categories.
location	string	Required if either latitude or longitude is not provided. This string indicates the geographic area to be used when searching for businesses. Examples: "New York City", "NYC", "350 5th Ave, New York, NY 10118". Businesses returned in the response may not be strictly within the specified location.
latitude	decimal	Required if location is not provided. Latitude of the location you want to search nearby.

Yelp Fusion

Introduction
Business Endpoints
Business Search
Phone Search
Transaction Search
Business Details
Business Match
Reviews
Autocomplete
Event Endpoints
Category Endpoints
Changelog
Fusion VIP

Name	Type	Description
term	string	Optional. Search term, for example "food" or "restaurants". The term may also be business names, such as "Starbucks". If term is not included the endpoint will default to searching across businesses from a small number of popular categories.
location	string	Required if either latitude or longitude is not provided. This string indicates the geographic area to be used when searching for businesses. Examples: "New York City", "NYC", "350 5th Ave, New York, NY 10118". Businesses returned in the response may not be strictly within the specified location.
latitude	decimal	Required if location is not provided. Latitude of the location you want to search nearby.
longitude	decimal	Required if location is not provided. Longitude of the location you want to search nearby.
radius	int	Optional. A suggested search radius in meters. This field is used as a suggestion to the search. The actual search radius may be lower than the suggested radius in dense urban areas, and higher in regions of less business density. If the specified value is too large, a AREA_TOO_LARGE error may be returned. The max value is 40000 meters (about 25 miles).
categories	string	Optional. Categories to filter the search results with. See the list of supported categories . The category filter can be a list of comma delimited categories. For example, "bars,french" will filter by Bars OR French. The category identifier should be used (for example "discgolf", not "Disc Golf").
locale	string	Optional. Specify the locale into which to localize the business information. See the list of supported locales . Defaults to en_US.

Appendix B. Divvy Bikes and Docking Stations





Divvy Data

Historical trip data available to the public

Here you'll find Divvy's trip data for public use. So whether you're a policy maker, transportation professional, web developer, designer, or just plain curious, feel free to download it, map it, animate it, or bring it to life!

Note that we'll be releasing trip data twice a year: once following the end of calendar Q2 and once following the end of calendar Q4. This data is provided according to the [Divvy Data License Agreement](#).

The Data

Each trip is anonymized and includes:

- Trip start day and time
- Trip end day and time
- Trip start station
- Trip end station
- Rider type (Member, Single Ride, and Explore Pass)

Browser window showing the JSON data from <https://gbfs.divvybikes.com/gbfs/gbfs.json>.

JSON structure:

```
{  "last_updated": 1564240687,  "ttl": 2,  "data": {    "en": {      "feeds": {        "0": {          "name": "station_status",          "url": "https://gbfs.divvybikes.com/gbfs/en/station_status.json"        },        "1": {          "name": "system_information",          "url": "https://gbfs.divvybikes.com/gbfs/en/system_information.json"        },        "2": {          "name": "system_alerts",          "url": "https://gbfs.divvybikes.com/gbfs/en/system_alerts.json"        },        "3": {          "name": "station_information",          "url": "https://gbfs.divvybikes.com/gbfs/en/station_information.json"        },        "4": {          "name": "system_regions",          "url": "https://gbfs.divvybikes.com/gbfs/en/system_regions.json"        }      },      "es": {        "feeds": {          "0": {            "name": "station_status",            "url": "https://gbfs.divvybikes.com/gbfs/es/station_status.json"          },          "1": {            "name": "system_information",            "url": "https://gbfs.divvybikes.com/gbfs/es/system_information.json"          },          "2": {            "name": "system_alerts",            "url": "https://gbfs.divvybikes.com/gbfs/es/system_alerts.json"          },          "3": {            "name": "station_information",            "url": "https://gbfs.divvybikes.com/gbfs/es/station_information.json"          }        }      }    }  }
```