

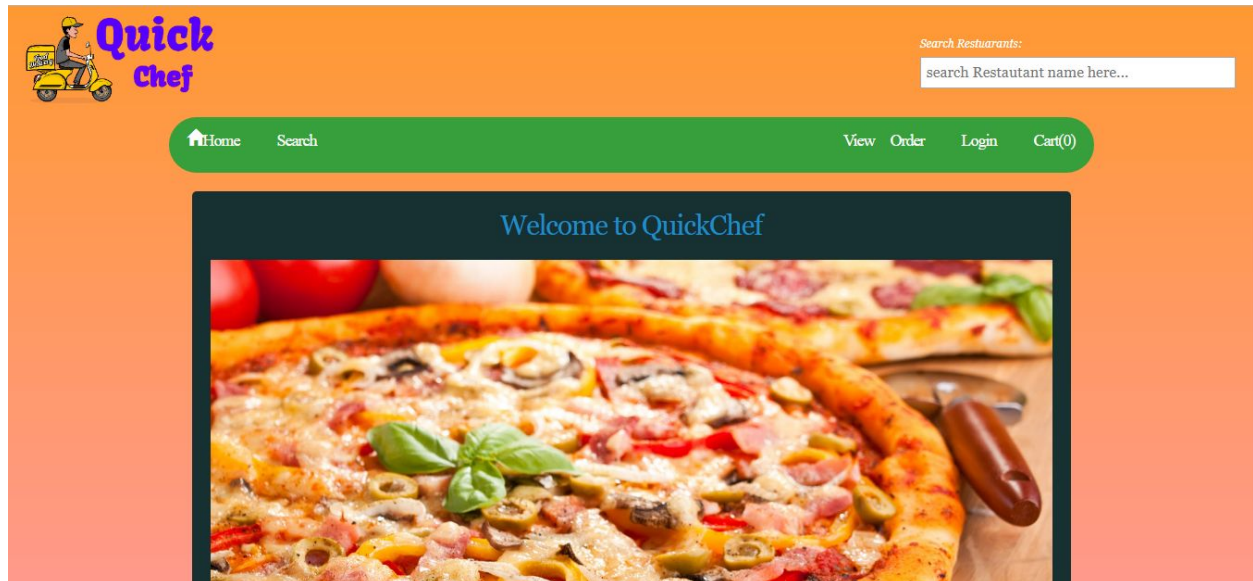
Srinivasa, Jaikerthi - [jsrinivasa@hawk.iit.edu](mailto:jsrinivasa@hawk.iit.edu)

Vijay, Suraj - [svijay1@hawk.iit.edu](mailto:svijay1@hawk.iit.edu)

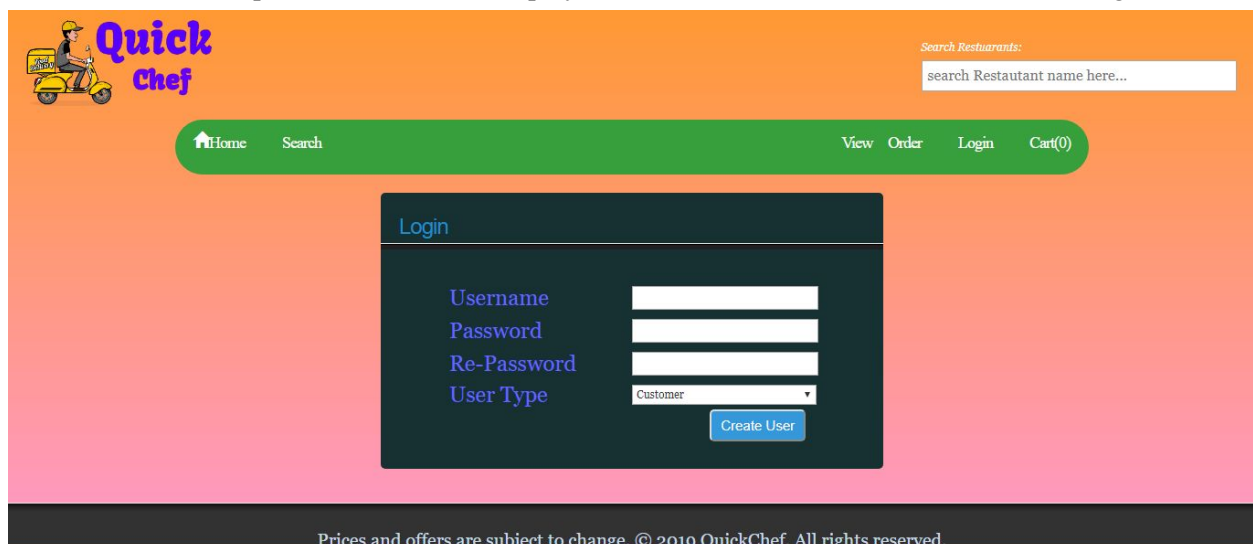
# Enterprise Web Applications

## Team 20 - QuickChef


### Final Project Snapshots



**Home Page** with a search tab(with autocomplete feature) with which we can search restaurants by its name. Clicking the search button on the menu tab, it redirects to the list of restaurants. View button is to view the orders for a particular user. Cart displays the items that the user has added for ordering.



**Login and Registration Screen.** Existing users can login to the application and new user can register using this portal.



Search Restaurants:

A

Au Cheval

Aloha Eats

ArePA George

Ali Baba Doner

Aloha Eats

[Home](#)
[Search](#)
[ViewOrder](#)
[Hello,Jai](#)
[Account](#)
[Logout](#)
[Cart\(0\)](#)


Address Search

Restuarants

Seoul Taco

738 N Clark St Chicago, IL 60634\$

4.0




Buy From this Restuarant

Farm & Sea Turkish Cuisine

2833 N Broadway Chicago, IL 60637\$

5.0




Buy From this Restuarant

Papa's Cache Sabroso

2317 W Division St Chicago, IL 60622\$

4.0




Buy From this Restuarant

SoJu BBQ

36 S Ashland Chicago, IL 60607\$

4.5




Buy From this Restuarant

Au Cheval

800 W Randolph St Chicago, IL 60607\$

4.5




Buy From this Restuarant

Cracked on Milwaukee

1359 N Milwaukee Ave Chicago, IL 60622\$

4.5

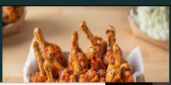


Buy From this Restuarant

Landbirds

2532 N California Ave Chicago, IL 60647\$

4.5

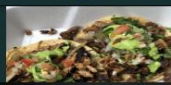


Buy From this Restuarant

Taco Loco Of Pilsen

2022 S Leavitt St Chicago, IL 60608\$

5.0




Buy From this Restuarant

Small Cheval

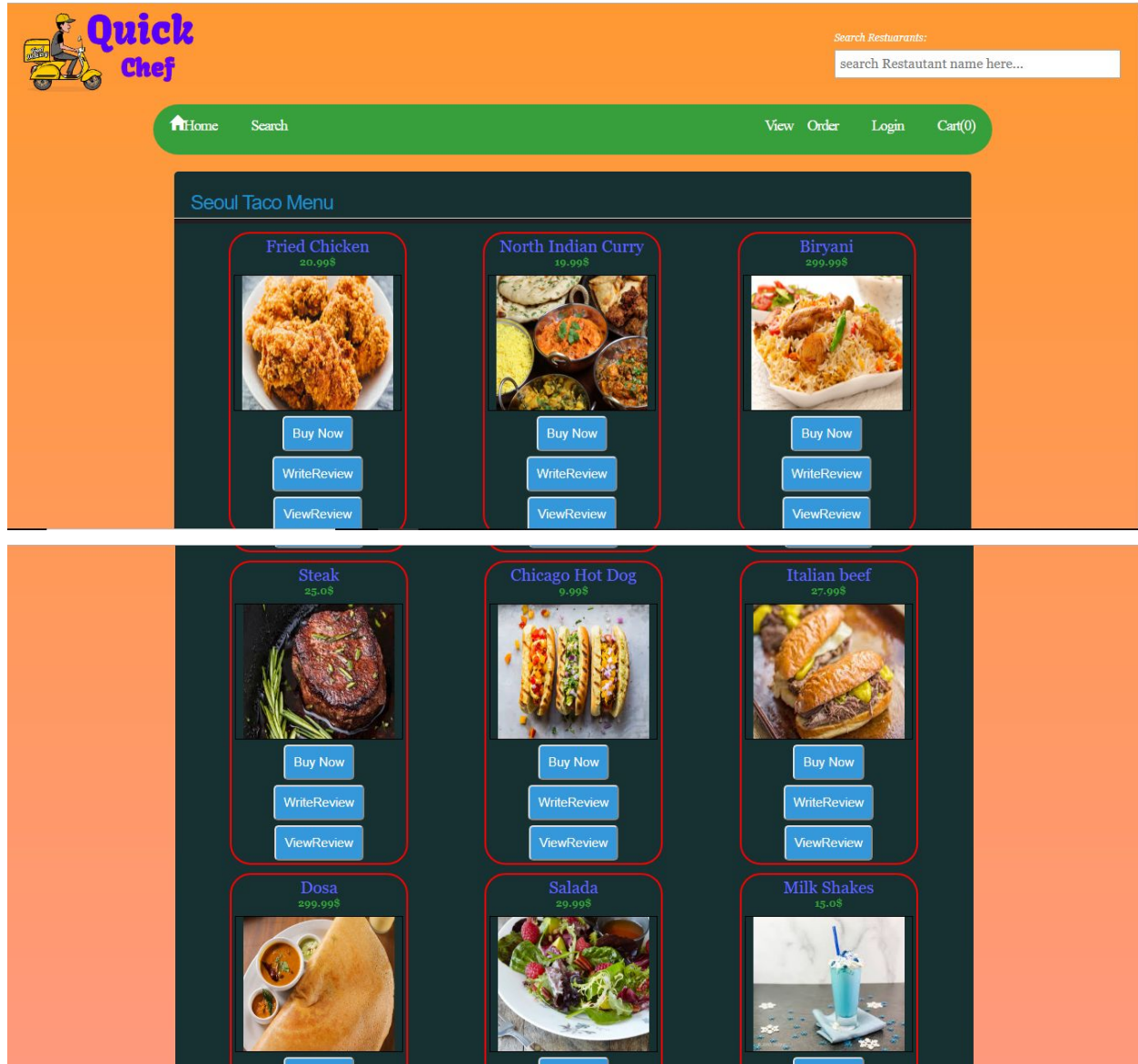
1732 N Milwaukee Ave Chicago, IL 60647\$

4.5

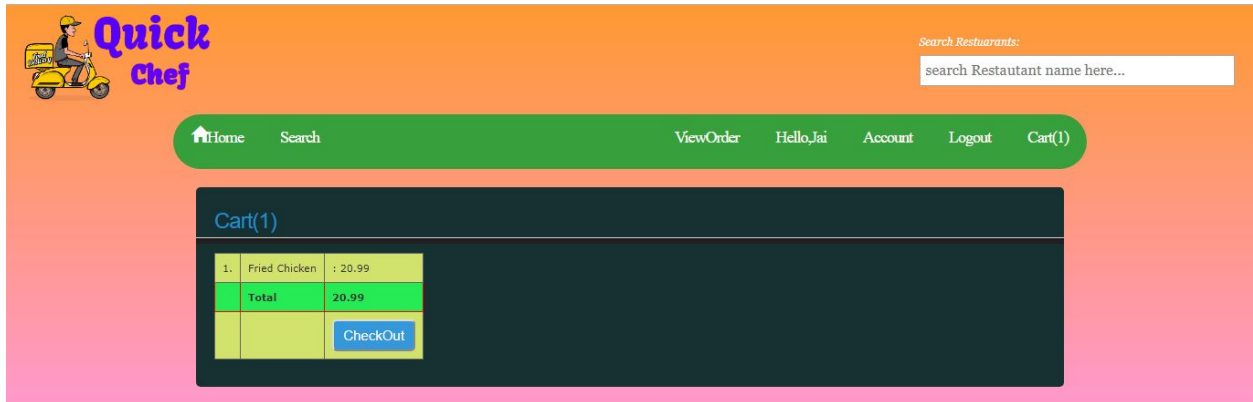


Buy From this Restuarant

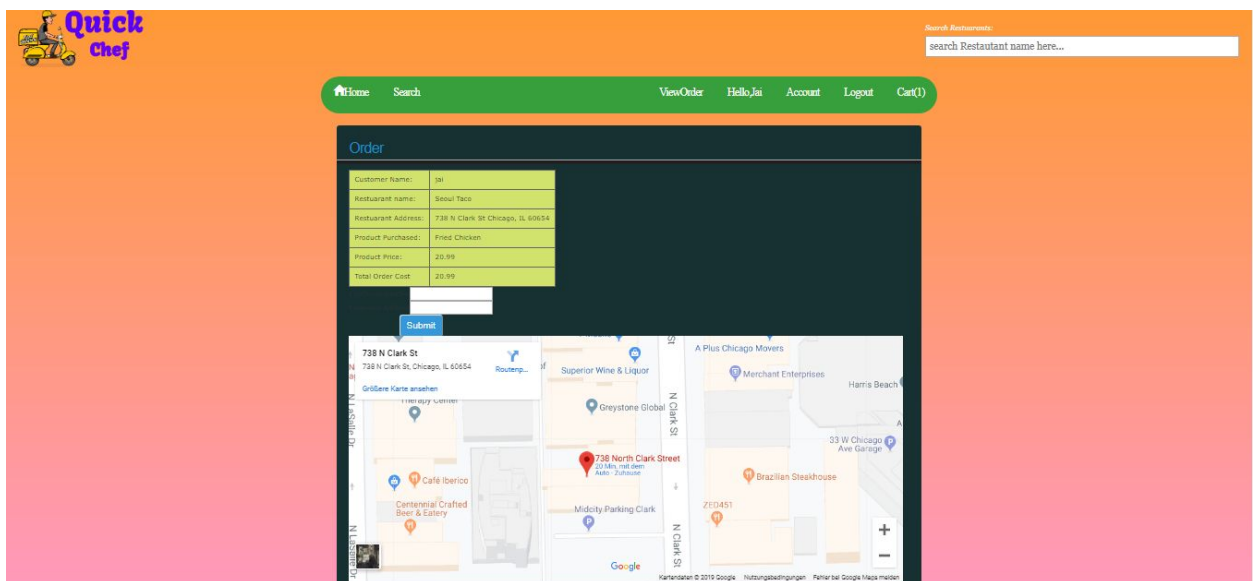
**Search Page.** On click of the search button, this page is displayed which shows the list of the restaurants. There is also another search bar where user can search the restaurants based on the address of the restaurant. All the restaurants, their images, names, addresses and their ratings are being displayed. Users have to select the restaurant that they need to order by clicking on 'Buy From this Restaurant' button.



**Menu Page.** On click of 'Buy From this Restaurant' button, it is redirected to the menu page where in it displays all the menu available in that restaurant. Users can add the food from the list and can proceed to check out. Users can write reviews and view the reviews for a particular dish.

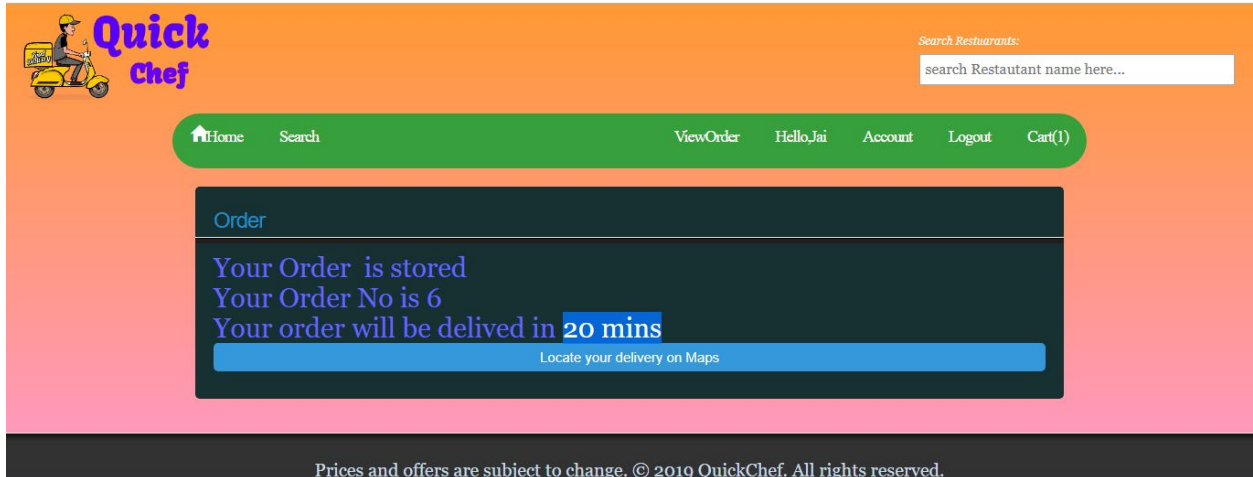


**Cart** displays all the items that are added by the user along with their total price.

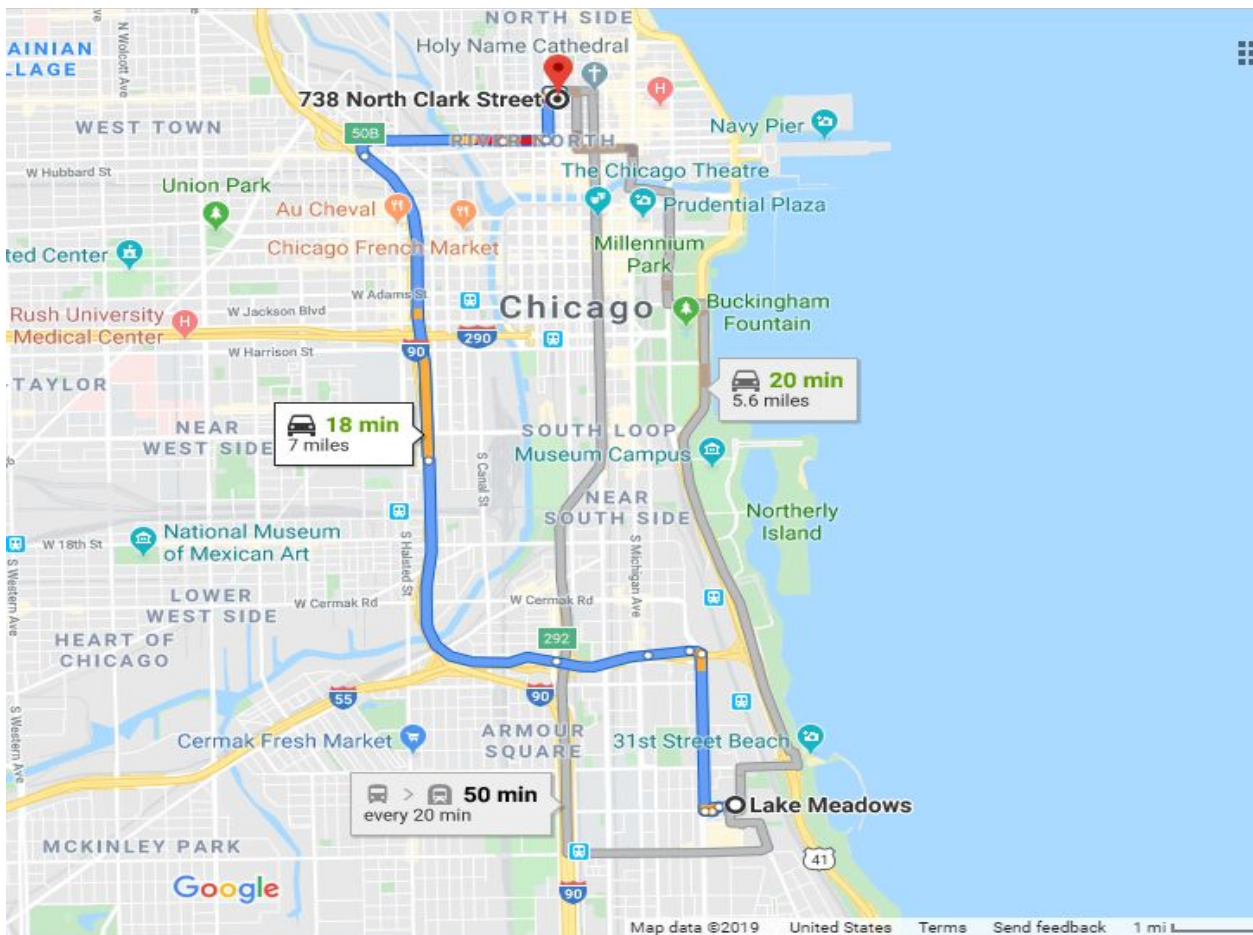


**CheckOut.** Order Summary Page displays the order details and along with it, it displays the restaurant's location on the map it use the *google maps API*. Users need to enter their valid card details and address to place an order.






Users will be displayed with their order number and along with that, users can know the approximate delivery time of their order in this window. It uses *google distance matrix API* for prediction. Users can locate the distance and approximate time of arrival in a map view by clicking on 'Locate your delivery on Maps' button.



A map view of the direction, distance and approximate time of arrival from the restaurant to the user's location.



Search Restaurants:


Home
Search
ViewOrder
Hello,Jai
Account
Logout
Cart(0)

### Account

User Name:	jai
User Type:	customer

	OrderId:	UserName:	productOrdered:	productPrice:	
<input type="radio"/>	1.	jai	Fish Fry	Price: 99.99	<button>CancelOrder</button>
<input type="radio"/>	2.	jai	North Indian Curry	Price: 19.99	<button>CancelOrder</button>
<input type="radio"/>	3.	jai	Biryani	Price: 299.99	<button>CancelOrder</button>
<input type="radio"/>	4.	jai	Dosa	Price: 299.99	<button>CancelOrder</button>
<input type="radio"/>	5.	jai	North Indian Curry	Price: 19.99	<button>CancelOrder</button>


Users can **cancel the order** if they wish to, and can place a new order.




Search Restaurants:

Home
Search
Addproduct
Updateproduct
Deleteproduct
Hello,Kee
Logout
Cart(0)

### Welcome to QuickChef



This window is for the **store manager**. He can add, delete, and update order upon user's request




Search Restaurants:

Home
Search
Create
Customer
ViewOrder
Hello,Jk
Logout
Cart(0)

### Order

	OrderId:	UserName:	productOrdered:	productPrice:	
<input checked="" type="radio"/>	2.	jai	North Indian Curry	Price: 19.99	<button>CancelOrder</button>

**Admin** can cancel the order placed by the user.




Search Restaurants:

[Home](#)
[Search](#)
[ViewOrder](#)
[Hello,Jai](#)
[Account](#)
[Logout](#)
[Cart\(0\)](#)

### Review

Product Name:	chicken
Restuarants	Seoul Taco
Product Price:	20.99
Review Rating:	4
Retailer Zip Code:	60616
Retailer City:	Chicago
Review Date:	24-11-2019
Review Text:	Good!!

Users can rate the food in that restaurant and **write reviews** about it .



Search Restaurants:

[Home](#)
[Search](#)
[ViewOrder](#)
[Hello,Jai](#)
[Account](#)
[Logout](#)
[Cart\(0\)](#)

### Review

Product Name:	chicken
userName:	jai
price:	20
Retailer City:	Chicago
Review Rating:	4
Review Date:	2019-11-24
Review Text:	Good!!

Prices and offers are subject to change. © 2019 QuickChef. All rights reserved.

**Review** written by the user is stored in MongoDB(Nosql database)

```
{ "id" : ObjectId("5dd88289b8c19f28080a1669"), "title" : "myReviews", "userName" : "jk", "retailername" : "Jai", "retailerstate" : "IL", "productsale" : "Yes", "manufacturerebate" : "Yes", "userid" : "3454634", "usage" : "26", "usergender" : "Male", "useroccupation" : "Master", "productName" : "iPhone7", "productType" : "games", "productMaker" : "apple", "reviewRating" : 5, "reviewDate" : "2019-11-24", "reviewText" : "Very good!", "retailerpin" : "60616", "retailercity" : "Chicago", "price" : 400 }
{ "id" : ObjectId("5dd8832cb8c19f28080a166c"), "title" : "myReviews", "userName" : "kj", "retailername" : "bhata", "retailerstate" : "IL", "productsale" : "Yes", "manufacturerebate" : "Yes", "userid" : "1234", "usage" : "24", "usergender" : "Male", "useroccupation" : "Pro", "productName" : "LG2", "productType" : "televisions", "productMaker" : "Ig", "reviewRating" : 5, "reviewDate" : "2019-11-21", "reviewText" : "GOOD", "retailerpin" : "60616", "retailercity" : "Chicago", "price" : 130 }
{ "id" : ObjectId("5dd89da7b8c19f28080a1670"), "title" : "myReviews", "userName" : "jai", "retailername" : "Jaik eerthi", "retailerstate" : "IL", "productsale" : "Yes", "manufacturerebate" : "Yes", "userid" : "13898", "usage" : "20", "usergender" : "Male", "useroccupation" : "Govt Employee", "productName" : "att", "productType" : "wirelessplans", "productMaker" : "Basic", "reviewRating" : 3, "reviewDate" : "2019-11-22", "reviewText" : "Good", "retailerpin" : "60616", "retailercity" : "Chicago", "price" : 50 }
{ "id" : ObjectId("5dd8a094b8c19f28080a1674"), "title" : "myReviews", "userName" : "jai", "retailername" : "Jaik eerthi", "retailerstate" : "IL", "productsale" : "Yes", "manufacturerebate" : "Yes", "userid" : "1234", "usage" : "23", "usergender" : "Male", "useroccupation" : "Bussiness Man", "productName" : "Microsoft", "productType" : "soundsystems", "productMaker" : "Microsoft", "reviewRating" : 5, "reviewDate" : "2019-11-22", "reviewText" : " Good", "retailerpin" : "60616", "retailercity" : "Chicago", "price" : 238 }
{ "id" : ObjectId("5dd8a395b8c19f28080a1679"), "title" : "myReviews", "userName" : "jai", "retailername" : "Jaik eerthi", "retailerstate" : "IL", "productsale" : "Yes", "manufacturerebate" : "Yes", "userid" : "3454634", "usage" : "23", "usergender" : "Male", "useroccupation" : "Bussiness Man", "productName" : "att", "productType" : "wirelessplans", "productMaker" : "Basic", "reviewRating" : 2, "reviewDate" : "2019-11-13", "reviewText" : " Good", "retailerpin" : "60616", "retailercity" : "Chicago", "price" : 50 }
{ "id" : ObjectId("5dd8a485b8c19f28080a167d"), "title" : "myReviews", "userName" : "john", "retailername" : "Jaik eerthi", "retailerstate" : "IL", "productsale" : "Yes", "manufacturerebate" : "Yes", "userid" : "23432543", "usage" : "22", "usergender" : "Male", "useroccupation" : "Bussines", "productName" : "Sharp3", "productType" : "televisions", "productMaker" : "sharp", "reviewRating" : 4, "reviewDate" : "2019-11-06", "reviewText" : " Hi", "retailerpin" : "60616", "retailercity" : "Chicago", "price" : 169 }
{ "id" : ObjectId("5dd8a4a0b8c19f28080a1680"), "title" : "myReviews", "userName" : "john", "retailername" : "Jaik eerthi", "retailerstate" : "IL", "productsale" : "Yes", "manufacturerebate" : "Yes", "userid" : "1234", "usage" : "23", "usergender" : "Male", "useroccupation" : "Bussiness Man", "productName" : "HuaweiP20", "productType" : "games", "productMaker" : "huawei", "reviewRating" : 4, "reviewDate" : "2019-11-22", "reviewText" : " Hi", "retailerpin" : "60616", "retailercity" : "Chicago", "price" : 50 }
{ "id" : ObjectId("5dd9eddb6b8c19f0fcb4c4e4b8"), "title" : "myReviews", "userName" : "jai", "productName" : "northindiancurry1", "productType" : "tablets", "productMaker" : null, "reviewRating" : 2, "reviewDate" : "2019-11-16", "reviewText" : " Good!", "retailerpin" : "60616", "retailercity" : "Chicago", "price" : 19 }
{ "id" : ObjectId("5ddb1dc4b8c19f04189003f1"), "title" : "myReviews", "userName" : "jai", "productName" : "chicken", "productType" : "tablets", "productMaker" : null, "reviewRating" : 4, "reviewDate" : "2019-11-24", "reviewText" : " Good!!!", "retailerpin" : "60616", "retailercity" : "Chicago", "price" : 20 }
```

MongoDB console which is used for storing reviews.

## API's Used:

The screenshot displays the Yelp Fusion API documentation for the `/businesses/search` endpoint. The page is divided into several sections:

- General:** Includes links for 'Manage App', 'Email / Notifications', 'Display Requirements', 'Terms of Use', and 'FAQ'.
- Yelp Fusion:** A sidebar menu with options like 'Introduction', 'Business Endpoints', 'Business Search' (highlighted), 'Phone Search', 'Transaction Search', 'Business Details', and 'Business Match'.
- /businesses/search:** The main content area describing the endpoint. It states that this endpoint returns up to 1000 businesses based on the provided search criteria. It includes a 'Request' section with the GET URL `https://api.yelp.com/v3/businesses/search`.
- Parameters:** A table listing the query parameters:
 

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.
- User Profile:** A dropdown menu for 'Jaik eerthi S.' showing options like 'About Me', 'Find Friends', 'Cash Back NEW', 'Account Settings', and 'Log Out'.

Yelp API developers account and Yelp business search API.

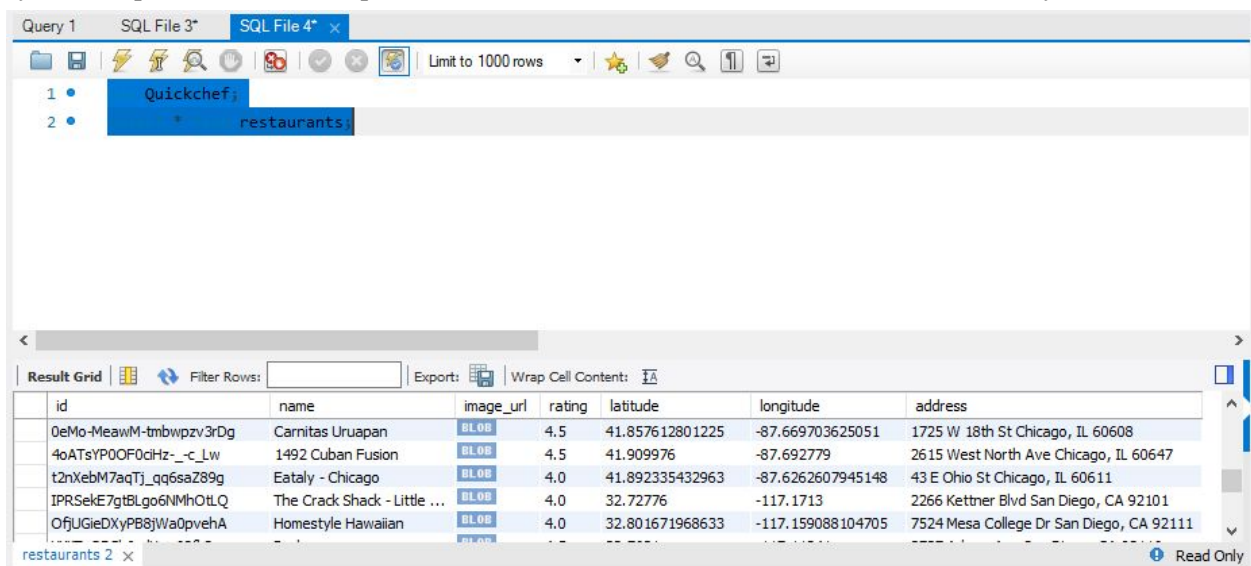


```

1 import requests
2 import json
3 import mysql.connector
4 mydb = mysql.connector.connect(
5     host="localhost",
6     user="root",
7     passwd="root",
8     database="Quickchef"
9 )
10 mycursor = mydb.cursor()
11
12 business_id = "Hy_H5ebrXaxCBdYuAd0asQ"
13
14 API_KEY = "q4RM_cQcKxK-B9W9k7a6LRr9Ze300IS_LNQxScN3FcmINjpFGFqgUMSoVeaB8D0pWe7RL616efHNaX1HcE3PF5R7K5W9nesISyQ35qIF0jqX90U1HN4GYahNjEXXYx"
15 ENDPOINT = "https://api.yelp.com/v3/businesses/search"
16 HEADERS = {'Authorization': 'bearer %s' % API_KEY}
17
18 PARAMETERS = {'term': 'food',
19               'limit': 20,
20               'location': 'USA'}
21
22 response = requests.get(url = ENDPOINT,
23                         params = PARAMETERS,
24                         headers = HEADERS)
25
26 business_data = response.json()
27 # print(business_data)
28 for i in business_data['businesses']:
29     address = '.join(i['location'][:display_address])
30     sql = "INSERT INTO restaurants (id,name, image_url,rating,latitude,longitude,address) VALUES (%s, %s,%s, %s,%s,%s,%s)"
31     val = (i['id'],i['name'], i['image_url'],i['rating'],i['coordinates']['latitude'],i['coordinates']['longitude'],address)
32     mycursor.execute(sql, val)
33     mydb.commit()
34     print(mycursor.rowcount, "record inserted.")
35
36 # print ("Name:", i['name'])
37 # print ("Image URL:", i['image_url'])
38 # print ("Rating:", i['rating'])
39 # print("coordinates-latitude",i['coordinates']['latitude'])
40

```

Python script that fires the Yelp API to fetch the restaurant details and store it into the MySQL database



Query 1 SQL File 3\* SQL File 4\* x

Limit to 1000 rows

1 • Quickchef;

2 • restaurants;

Result Grid Filter Rows: Export: Wrap Cell Content: [f](#)

id	name	image_url	rating	latitude	longitude	address
0eMo-MeawM-tmbwpzv3rDg	Carnitas Uruapan		4.5	41.857612801225	-87.669703625051	1725 W 18th St Chicago, IL 60608
4oATsYP0OF0ciHz-_c_Lw	1492 Cuban Fusion		4.5	41.909976	-87.692779	2615 West North Ave Chicago, IL 60647
t2nXebM7aqTj_qq6saZ89g	Eataly - Chicago		4.0	41.892335432963	-87.6262607945148	43 E Ohio St Chicago, IL 60611
IPRSekE7gtBLgo6NMhOtLQ	The Crack Shack - Little ...		4.0	32.72776	-117.1713	2266 Kettner Blvd San Diego, CA 92101
OfjUGieDxyPB8jWa0pvehA	Homestyle Hawaiian		4.0	32.801671968633	-117.159088104705	7524 Mesa College Dr San Diego, CA 92111

restaurants 2 x Read Only

View of MySQL database that stores the restaurants data that is fetched from the Yelp API.

Google Cloud Platform quickchefewa

APIs & Services

Credentials

Create credentials Delete

Create credentials to access your enabled APIs. For more information, see the [authentication documentation](#).

API keys

Name	Creation date	Restrictions	Key
API key 1	Nov 18, 2019	None	AlzaSyC8PidIvRq_tvF9GMvnPhj1u7vD-D-NUw

Google Maps Platform Overview Products Pricing Documentation Blog

Search Language

Get Started

Developer Guide

Get API Key

Web Services

Best Practices

Client Libraries

Policies and Terms

Usage and Billing

Policies

Terms of Service

Other Web Service APIs

Directions API

Elevation API

Geocoding API

Geolocation API

Places API

Roads API

Time Zone API

### Sample request and response

You access the Distance Matrix API through an HTTP interface, with requests constructed as a URL string, using **origins** and **destinations**, along with your API key.

The following example requests the distance matrix data between Washington, DC and New York City, NY, in JSON format:

```
https://maps.googleapis.com/maps/api/distancematrix/json?units=imperial&origins=Washington,DC&destinations=NewYorkCity,NY&key=YOUR_API_KEY
```

**Try it!** You can test this by entering the URL into your web browser (be sure to replace **YOUR\_API\_KEY** with your **actual API key**). The response includes the distance and duration between the specified origins and destinations.

View the [developer's guide](#) for more information about [building request URLs](#) and [available parameters](#) and [understanding the response](#).

Below is a sample response, in JSON:

```
try{
    address = address.replaceAll("\\s", "+");
    String url = "https://maps.googleapis.com/maps/api/distancematrix/json?origins="+myaddress+"&destinations="+add
    URL obj = new URL(url);
    HttpURLConnection con = (HttpURLConnection) obj.openConnection();
    con.setRequestMethod("GET");

    con.setRequestProperty("User-Agent", "Mozilla/5.0");
    int responseCode = con.getResponseCode();

    BufferedReader in = new BufferedReader(
        new InputStreamReader(con.getInputStream()));
    String inputLine;
    StringBuffer response1 = new StringBuffer();
    while ((inputLine = in.readLine()) != null) {
        response1.append(inputLine);
    }
    in.close();

    JSONObject myResponse = new JSONObject(response1.toString());
}
```

Google Distance Matrix API is used to predict the time of arrival from the restaurant to the user's location

The screenshot displays the Google Maps Platform documentation page for the Directions API. The page layout includes a top navigation bar with links for Overview, Products, Pricing, Documentation, and Blog, along with a search bar and a language selector. A left sidebar contains a 'Guides' section with links to 'Get Started', 'Developer Guide', and 'Get an API Key', as well as 'Web Services' including Distance Matrix API, Elevation API, Geocoding API, Geolocation API, Places API, Roads API, and Time Zone API. The main content area is titled 'Sample request and response' and explains that the Directions API is accessed via an HTTP interface. It provides a sample URL: `https://maps.googleapis.com/maps/api/directions/json?origin=Disneyland&destination=Universal+Studio`. A 'Try it!' section encourages users to test the request by replacing `YOUR_API_KEY` with their actual API key. A right sidebar lists the 'Contents' of the page, including 'Sample request and response', 'Start coding with our client libraries', 'Authentication, quotas, pricing, and policies', 'Activate the API and get an API key', 'Quotas and pricing', 'Policies', and 'Learn more'.

Google Maps Platform

Overview Products Pricing Documentation Blog

Search

Language

Send feedback

Guides Support

Get Started

Developer Guide

Get an API Key

Web Services

Best Practices

Client Libraries

Policies and Terms

Usage and Billing

Policies

Terms of Service

Other Web Service APIs

Distance Matrix API

Elevation API

Geocoding API

Geolocation API

Places API

Roads API

Time Zone API

★ This service is also available as part of the client-side [Maps JavaScript API](#), or for server-side use with the [Java Client](#), [Python Client](#), [Go Client](#) and [Node.js Client](#) for Google Maps Services.

## Sample request and response

You access the Directions API through an HTTP interface, with requests constructed as a URL string, using text strings or latitude/longitude coordinates to identify the locations, along with your API key.

The following example requests the driving directions from Disneyland to Universal Studios Hollywood, in JSON format:

```
https://maps.googleapis.com/maps/api/directions/json?origin=Disneyland&destination=Universal+Studio
```

**Try it!** You can test this request by entering the URL into your web browser (be sure to replace `YOUR_API_KEY` with [your actual API key](#)). The response returns the driving directions.

View the [developer's guide](#) for more information about [building request URLs and available parameters](#) and [understanding the response](#).

Below is a sample response, in JSON:

Contents

- Sample request and response
- Start coding with our client libraries
- Authentication, quotas, pricing, and policies
  - Activate the API and get an API key
- Quotas and pricing
- Policies
- Learn more

Google Maps API is used to provide map view of users and restaurant's location, distance and approximate time of arrival