



PROJECT: CORNUCOPIA

OUR TEAM

HUY LE

PHAT PHAN

DU NGUYEN

**PHAT
NGUYEN**

CONTENTS

01

FRONT END DESIGN

02

BACK END DESIGN: GOOGLE MAP API

03

BACK END DESIGN: USERS SERVER


04

BACK END DESIGN: LOGIC STRUCTURE



INTRODUCTION

Cornucopia is a restaurant finder app designed to provide customers with a personalized and seamless experience.



VISION

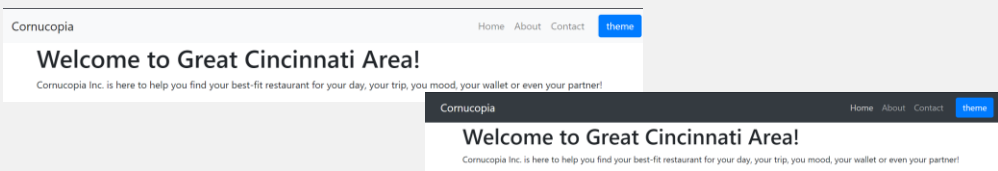
- Database of restaurant info and reviews from Google and Yelp
- **Target audience:** customers looking for restaurant and restaurant owners in Cincinnati
- **Competitors:** Yelp, Google, and Tripadvisor
- Website focuses on restaurant suggestions and reviews of dishes
- User interface like an "online-dating" app for picking a restaurant
- Description generated from databases or restaurant owners' profiles.



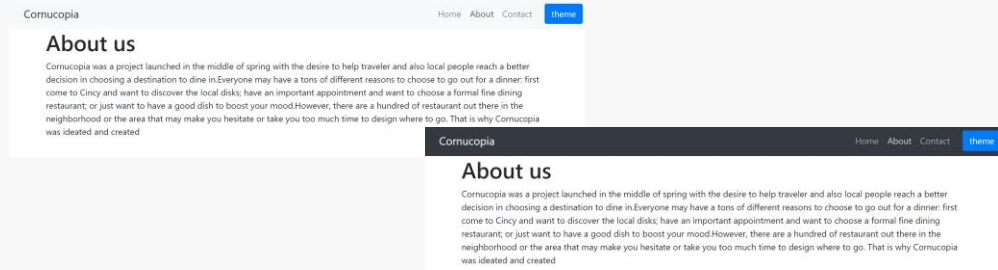
FRONT END DESIGN: MAIN PAGES

The Web Application will consist at least 3 main pages

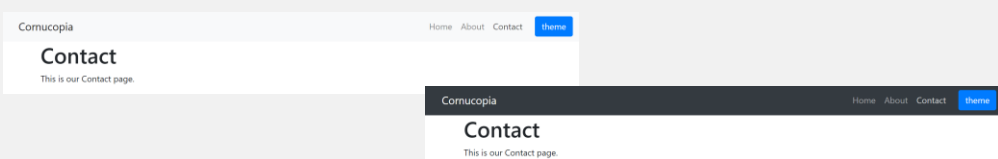
Home Page: This is where majority of the web interaction will happen



About us: This is where we tell our customer about our company's mission and vision



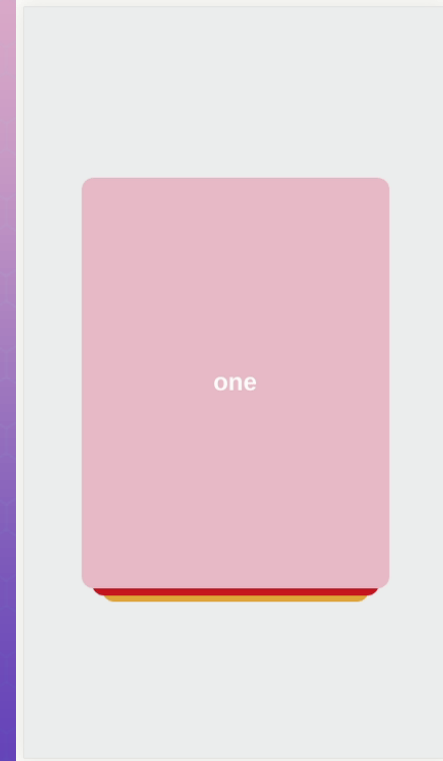
Contact: contact information for collaboration



FRONT END DESIGN: CARD STACK FOR SWIPE-RIGHT

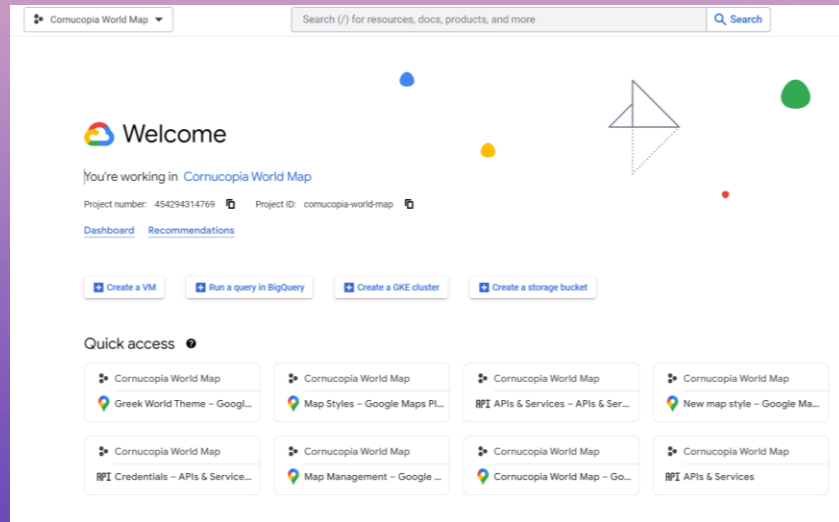
The idea is to generate a stack of cards of every restaurant containing their image and some of the information to swipe left (or hit button).

This is an application based on Vue-2-interact to utilize the API and create an interactive webpage. Here is the demo of the swipeable card stack.



BACK-END USING GOOGLE MAP API

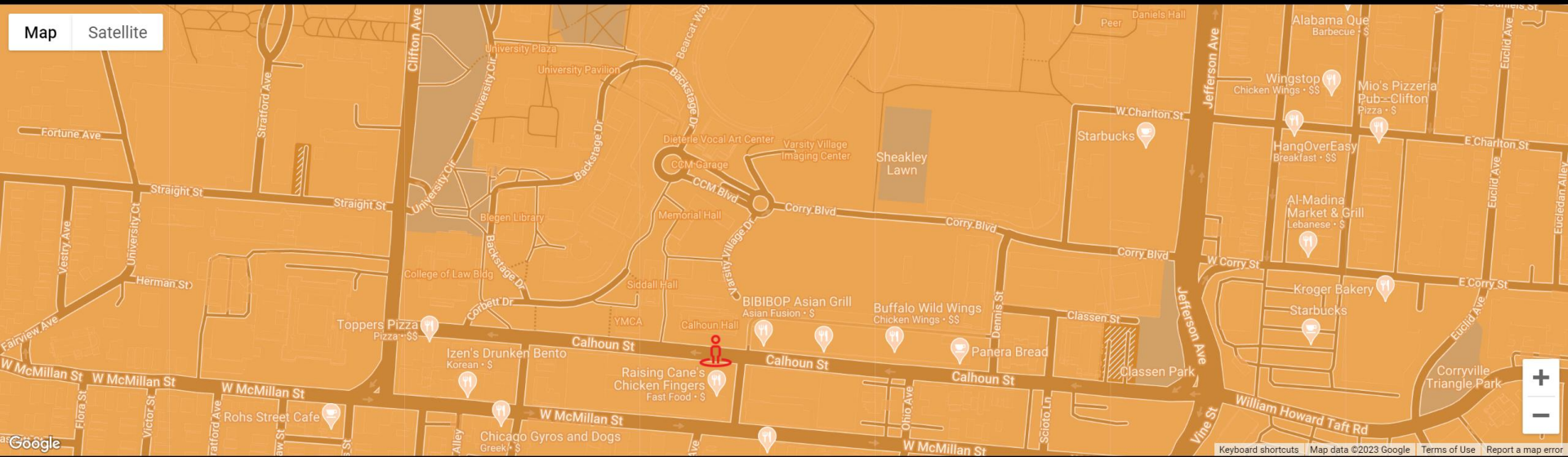
The restaurant data is collected using Google Cloud Platform. Data is scraped using Python and stored in Excel files. Explored the basic using Google API, enhance feature with Google supported functionality and Google Map JavaScript Map API



A	B	C	D	E	F	G
	business_status	geometry	icon	background	mask_bas	nam
0	OPERATIONAL	{'location': {'lat': 39.1433487, 'lng': -84.5197 https://n#FF9E67			https://m:Arlin's Bar & Restaurant	
1	OPERATIONAL	{'location': {'lat': 39.106826, 'lng': -84.51253 https://n#FF9E67			https://m:On the Rhine Eatery	
2	OPERATIONAL	{'location': {'lat': 39.1413241, 'lng': -84.5196 https://n#FF9E67			https://m:Baladi Restaurant & Bakery	
3	OPERATIONAL	{'location': {'lat': 39.1350931, 'lng': -84.5031 https://n#FF9E67			https://m:Piada Italian Street Food	
4	OPERATIONAL	{'location': {'lat': 39.1105411, 'lng': -84.5162 https://n#FF9E67			https://m:Salazar Restaurant • Chef Jose Sa	
5	OPERATIONAL	{'location': {'lat': 39.1620763, 'lng': -84.5394 https://n#FF9E67			https://m:Blue Jay Restaurant	
6	OPERATIONAL	{'location': {'lat': 39.1278306, 'lng': -84.5207 https://n#FF9E67			https://m:Good Plates Eatery	
7	OPERATIONAL	{'location': {'lat': 39.1028885, 'lng': -84.5114 https://n#FF9E67			https://m:Nada	
8	OPERATIONAL	{'location': {'lat': 39.1629041, 'lng': -84.5396 https://n#FF9E67			https://m:Katch The Kitchen Restaurant	
9	OPERATIONAL	{'location': {'lat': 39.0847476, 'lng': -84.5176 https://n#FF9E67			https://m:Bouquet Restaurant	
10	OPERATIONAL	{'location': {'lat': 39.1556341, 'lng': -84.6069 https://n#FF9E67			https://m:Santorini Family Restaurant	
11	OPERATIONAL	{'location': {'lat': 39.1909043, 'lng': -84.4528 https://n#FF9E67			https://m:Demetrios V	
12	OPERATIONAL	{'location': {'lat': 39.1557123, 'lng': -84.6086 https://n#FF9E67			https://m:Flope Bar & Restaurant	
13	OPERATIONAL	{'location': {'lat': 39.1028645, 'lng': -84.5110 https://n#FF9E67			https://m:Sotto	
14	OPERATIONAL	{'location': {'lat': 39.0858026, 'lng': -84.5216 https://n#FF9E67			https://m:The Fifth Restaurant	
15	OPERATIONAL	{'location': {'lat': 39.140027, 'lng': -84.51312 https://n#FF9E67			https://m:Corinthian Restaurant	
16	OPERATIONAL	{'location': {'lat': 39.0836563, 'lng': -84.5096 https://n#FF9E67			https://m:Coppin's Restaurant & Bar	
17	OPERATIONAL	{'location': {'lat': 39.1062773, 'lng': -84.5151 https://n#FF9E67			https://m:Lalo Chino Latino Kitchen	
18	OPERATIONAL	{'location': {'lat': 39.1191058, 'lng': -84.5994 https://n#FF9E67			https://m:Price Hill Chili Family Restaurant	
19	OPERATIONAL	{'location': {'lat': 39.1651481, 'lng': -84.5019 https://n#FF9E67			https://m:Five Stars mediterranean restaura	
20	OPERATIONAL	{'location': {'lat': 39.1278333, 'lng': -84.5175 https://n#FF9E67			https://m:Waffle House	
21	OPERATIONAL	{'location': {'lat': 39.1436194, 'lng': -84.5190 https://n#FF9E67			https://m:Skyline Chili	
22	OPERATIONAL	{'location': {'lat': 39.127979, 'lng': -84.51692 https://n#FF9E67			https://m:Raising Cane's Chicken Fingers	
23	OPERATIONAL	{'location': {'lat': 39.0609043, 'lng': -84.5418 https://n#FF9E67			https://m:Fort Wright Family Restaurant	
24	OPERATIONAL	{'location': {'lat': 39.1486716, 'lng': -84.4447 https://n#FF9E67			https://m:J. Alexander's Restaurant	
25	OPERATIONAL	{'location': {'lat': 39.1284323, 'lng': -84.5163 https://n#FF9E67			https://m:BIBIBOP Asian Grill	

CUSTOM THEME MAP WITH DATA FROM GOOGLE

Explore the restaurants around you in this Greek Theme



BACK-END: Server management and data caching

The codes are designed to effectively manage and communicate data to servers and the app.

```
1 {
2   "name": "backend",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7   "scripts": {
8     "test": "echo \"Error: no test specified\" && exit 1"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "body-parser": "^1.19.0",
15    "bson": "^4.2.2",
16    "cors": "^2.8.5",
17    "dotenv": "^8.2.0",
18    "express": "^4.17.1",
19    "mongodb": "^3.6.4"
20  }
21 }
```

```
75 int iAvailable = N;
76
77 // While there are free men
78 while (iAvailable > 0)
79 {
80   // Pick the first free man
81   int m;
82   for (m = 0; m < N; m++)
83     if (wDataFree[m] == false)
84       break;
85
86   // One by one go to all women according to m's preferences.
87   // Here m is the picked free man
88   for (int i = 0; i < N && wDataFree[i] == false; i++)
89   {
90     int w = prefs[m][i];
91
92     // The woman of preference is free, w and m become
93     // partners (can be changed). So we can say they are taken
94     if (wData[w-N] == -1)
95     {
96       wData[w-N] = m;
97       wDataFree[m] = true;
98       iAvailable--;
99     }
100
101     else // If w is not free
102     {
103       // Find current engagement of w
104       int m1 = wData[w-N];
105
106       // If w prefers m over her current engagement m1,
107       // then break the data between w and m1 and match w with m.
108       if (preflist[m1, w, m, m1] == false)
109       {
110         //
111       }
112     }
113   }
114 }
```

BACK-END TRANSFERRING DATA FROM SERVER TO APP

```
1 import app from './server.js'
2 import mongodb from 'mongodb'
3 import dotenv from 'dotenv'
4 import RestaurantsDAO from './dao/restaurantsDAO.js'
5 import ReviewsDAO from './dao/reviewsDAO.js'
6 dotenv.config()
7 const MongoClient = mongodb.MongoClient
8
9 const port = process.env.PORT || 8000
10
11 MongoClient.connect(
12   process.env.RESTREVIEWS_DB_URI,
13   {
14     poolSize: 50,
15     wtimeout: 2500,
16     useNewUrlParser: true
17   }
18 )
19 .catch(err => {
20   console.error(err.stack)
21   process.exit(1)
22 })
23 .then(async client => {
24   await RestaurantsDAO.injectDB(client)
25   await ReviewsDAO.injectDB(client)
26   app.listen(port, () => {
27     console.log(`listening on port ${port}`)
28   })
29 })
```

```
21
22 class Graph
23 {
24   private:
25     int Vert;
26     GraphNode* Adjlist;
27     int vertices_visited[];
28
29   public:
30     Graph(int input_list[])
31     {
32       this->Vert = input_list[0];
33       this->Adjlist = new GraphNode[Vert];
34       this->vertices_visited[Vert];
35       int i = 0;
36       for(i; i < Vert; i++)
37       {
38         Adjlist[i] = NULL;
39         vertices_visited[i] = 0;
40       }
41
42       i = 1;
43       while(input_list[i] != -1)
44       {
45         CreateEdge(input_list[i], input_list[i+1]);
46         i += 2;
47       }
48
49     }
50
51   ~Graph()
52   {
53     int i = 0;
54     GraphNode* edge = this->Adjlist[0];
55     GraphNode* temp;
```

Customers Usage Data

- Including a login page for customers
- Storing user's data in a separate servers
- Analyzing users' references, we could implement a suggestion system based on their past restaurant.

