

# Find 'N' Ride

*THE WORLD'S FIRST APP THAT DELIVERS YOU TO THE FOOD!!!!*

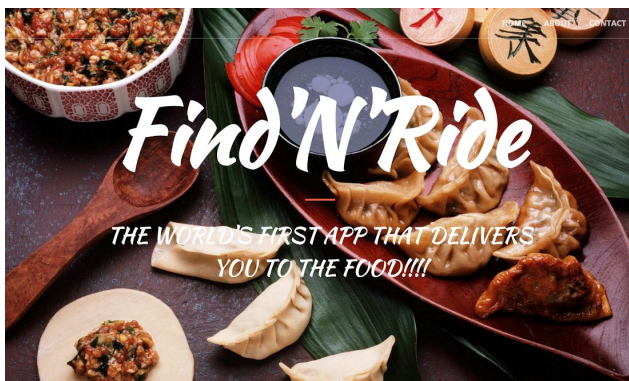
by Brian Goodness, Carlo Liquido, Keshav Potluri, Richa Prajapati

## Introduction

The main idea of this project is to create a platform that integrates mobility in a city with fast querying abilities. After deliberation on how to scope our project, we decided to build a tool that queried for food options and the fastest possible way to reach there. Our team developed the "Find and Ride" app which enables users to locate nearby restaurants and/or Yelp-listed establishments and provides taxi transportation prices (Uber) to those destinations via a map-based interface. The app is being hosted live, at: <http://find-and-ride.herokuapp.com/>

## Application Features

Following is a sample walkthrough of the application. Along with the walkthrough, we have also described all the features and capabilities supported by our application at different stages:



This is the homepage of Find 'N' Ride. It describes what Find 'N' Ride offers to the users in a visual and simple way.

It is essentially a scrolling page with 3 sections:

1. The first section (with the picture) is the 'Home' section which gives a one-liner about the application's functionality.
2. The second section (in orange) is the 'About' section. It describes in detail when the application can be used. It also offers a 'Get Started' button.
3. The third section (in white) is the 'Contact' section which provides the user a way to contact us and provide us with feedback or suggestions.

Hungry, but feeling Lazy? We've got what you need!!!

Hungry but can't wait for the food to deliver? Your favorite restaurant does not deliver food? Your favourite joint too far to deliver? Want to go out for dinner but don't know where? Don't worry we won't let you be hungry anymore. FatNLazy will search the food you want to eat and take you to the food. Try out our new app, free and easy to use. No strings attached!

Enter your location and get started!!!

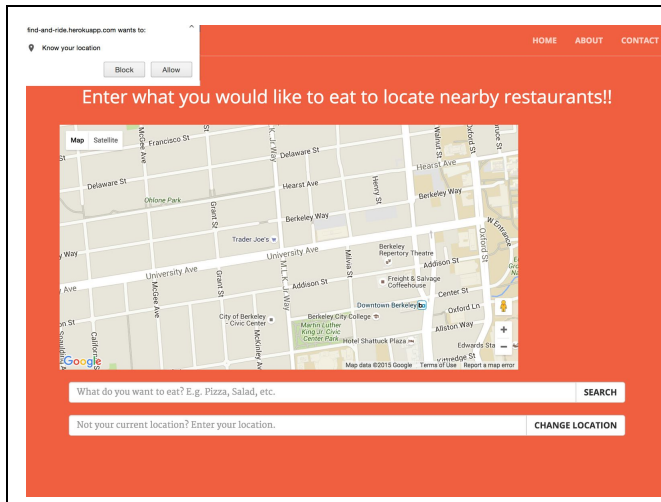
GET STARTED!

Let's Get In Touch!

Like our app? We would really like to have your feed back!!! Please call us or mail us.

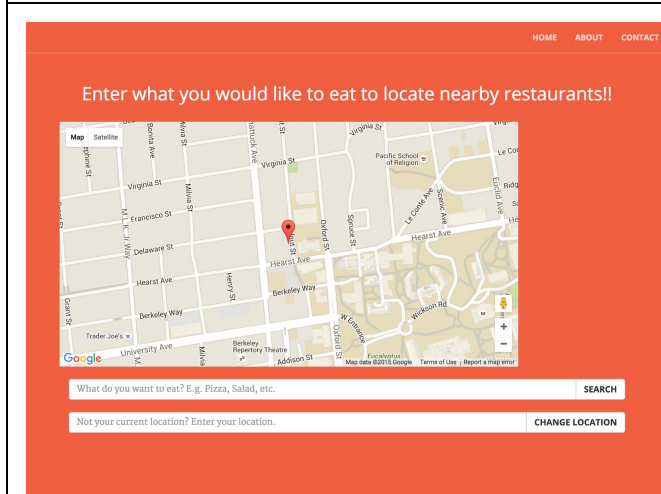
123-456-6789

feedback@fatandlazy.com



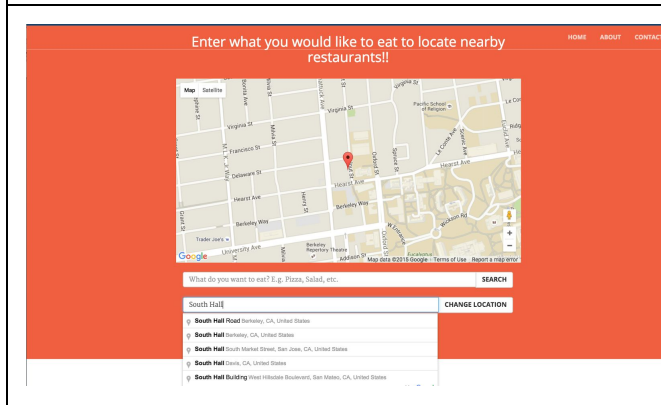
This is the page where most of the action happens. A user is taken to this page when he/she clicks on the ‘Get Started’ button in the home page. In order to get the user’s location on load, the user is asked whether he/she wants the application to access their location.

If a user says yes, then the map is updated with the user’s IP address. If the user says no, then the map is updated with a random location which can be updated entering a location in the “Change Location” textbox.



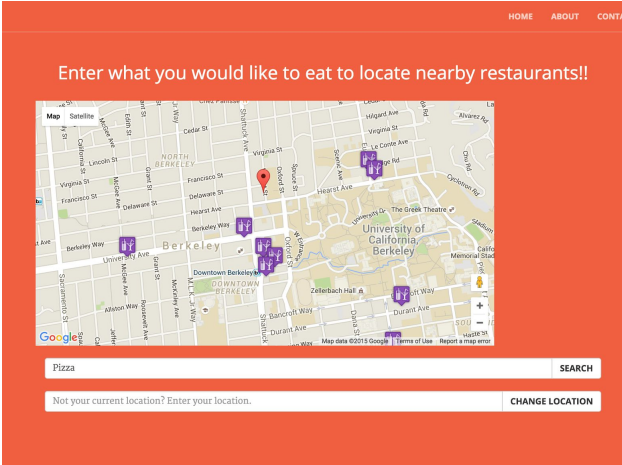
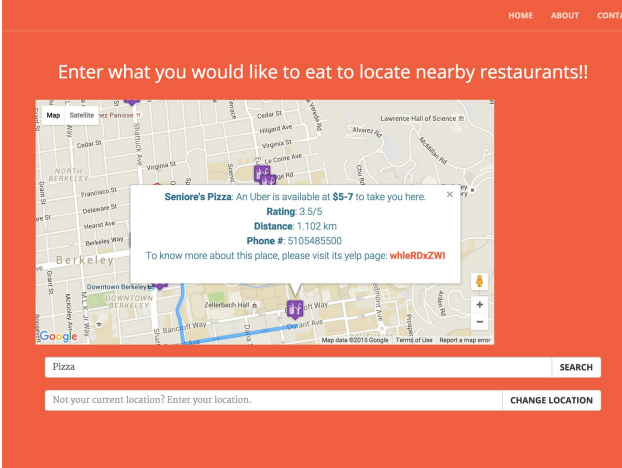
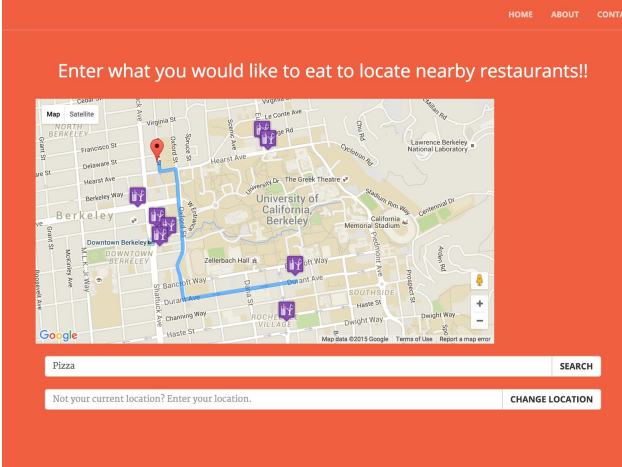
If a user says yes to the “Access Location” option, then a red marker shows the user’s IP address on the map. If the user says no, then the map is updated with a random location which can be updated upon entering a location in the “Change Location” textbox.

This provides the user with the facility to access nearby restaurants or Yelp-listed establishments without actually being there. Thus, the user can also use this app for planning in advance.



When a user enters something in the “Change Location” textbox to change his/her current location, our application tries to “auto-complete” the address by providing a list of all possible addresses which exist in Google’s database matching the text present in the textbox.

The user can select one of these addresses and the red marker is updated to this new current location.

	<p>Once the user's location has been captured, the user can enter the type of food he/she wants to eat in the nearby locality in the "Search" textbox.</p> <p>This search hits the Yelp and Uber API and returns the top 10 restaurants or Yelp-listed establishments results based on search query.</p> <p>The location of these restaurants or Yelp-listed establishments is highlighted with purple markers on the map.</p>
	<p>Clicking on one of these purple markers will open an Information window which gives various information like:</p> <ul style="list-style-type: none"> <li>• Name of the restaurant</li> <li>• The estimated Uber fare to reach that restaurant from the user's location on map (or location the user queried)</li> <li>• The restaurant's Yelp rating</li> <li>• The restaurant's distance from the user's location on map</li> <li>• The restaurant's phone number</li> <li>• A URL-shortened link to the restaurant's Yelp page</li> </ul> <p>The driving direction to reach the restaurant is also drawn on the map</p>
	<p>By closing the Information window, one can clearly see the driving direction to reach the restaurant is drawn on the map.</p>

	<p>The user can continue clicking on other nearby restaurant markers to gather more information about them.</p> <p>When he/she does so, the information window and directions of the previously opened restaurant are automatically updated on the map.</p>
--	---

The user can keep changing his/her location and food preferences in the way described above and the map will keep updating to reflect this changes.

## Technologies Used

Our team used the following web-related technologies:

### Frontend: Bootstrap JavaScript and CSS libraries, Google Fonts

The front end was primarily designed using Django templates that utilizes HTML to create web forms. The styling was done using custom CSS as well as the Twitter Bootstrap Libraries. The fonts on the website were obtained using the Google Fonts API that allows the user to use any of the fonts publicly made available by Google. Javascript was used to integrate google maps with the web pages. Rendering the map, getting user location, plotting the user location on the map, getting auto complete for locations, making an AJAX call to call other APIs, use the response from other APIs to plot restaurants on the map, plot directions on the map, plot other details about businesses on the map and handle any kind of user actions. Since Google Maps API uses Javascript, it was the best option for integrating them with our project.

### Web Framework: Django

Brian had some prior work experience with Django, but had not before configured a fully-functional Django-based web application from scratch. Therefore, implementing Django was a learning experience, but, it was also used in order to leverage the framework's built-in features, such as its URL routing and forms API, which includes field validations. In addition, Django makes it especially easy to specify app models and to configure it with a SQL-based database.

Moreover, using a Django-based framework allowed the team to work independently and develop the site with a modular infrastructure. Each member was able to instantiate their own application, from which other apps could import and borrow models, page templates, and views (logical classes written in Python) from one another person's application within the same Django project, all in a compact and complete manner. For instance, in our project, one application served the main web pages, but called the Yelp API from a different application within

the same project. Moreover, the main app that served the site's web pages accessed the database for reading and writing short and long URL paths through a different application.

From a machine configuration and security perspective, we also leveraged environmental variables for project settings and secret keys (e.g., database connection, Yelp API). Storing secret keys as environmental variables allowed us to avoid hardcoding them into version control for public view. Moreover, we were able to configure the web application to behave differently, based on which machine it was being served from-- e.g., the Django debugger was turned on when serving the project's web pages on our local computers, but, set to be turned off when deployed live on Heroku. On a local computer, this achieved by editing the `~/.bash_profile` (on mac) or `~/.bashrc` (on linux) file, and on Heroku, using the 'heroku config:add' command.

### APIs: Google Maps, Yelp, Uber

During the initial brainstorming stages of the project, we were all motivated by the idea of creating a platform that integrated mobility in a city with fast querying abilities. After deliberation on how to scope our project, we decided to build a tool that queried for food options from Yelp. In terms of other APIs, Google Maps and Uber APIs were selected for their comprehensiveness and easy integration.

#### Google Maps

There are various modules of **Google Maps API** that we have incorporated in our application. To begin with, the application tries to access the user's location in two ways - either the user can allow Google Maps API to directly derive his/her location based on the user's IP address, or the user can input his/her location into the "Search Location" textbox. While the user enters his/her location in the "Search Location" text box, we use the **AutoComplete** module of Google Maps to predict the possible addresses that matches user query in the Google Maps database. The Google Maps API then retrieves the user's location using its **GeoCoder** module. The results from the GeoCoder module is used to map the user's location on the map using the **Marker** module. Simultaneously, the GeoCoder results along with user input from the "Search" query are also sent to the Yelp API using asynchronous javascript.

On getting the results from Yelp and API, we used the Marker module again to plot the returned establishment's location on the map. We also used the **InfoWindow** module to display an establishment's Yelp and Uber results whenever the user clicks on the establishment's marker. Further, we also use the **Directions API** to draw the driving route from the user's location to the establishments on the map.

#### Yelp

The Yelp API returns, in JSON, the ten closest establishments based on keyword matches from user input. It also returns various attributes of the establishments present in the Yelp database like location of the establishments, ratings, phone number and Yelp URL. These results are sent both to the Google Maps (so that it can plot these establishments on the map) and the Uber (so that it can get the taxi fare estimate from user's current location to each of these establishments) APIs.

#### Uber

The way the Uber API works is that it takes in origin latitude and end latitude and posts the time, type of Uber vehicle, and price. The process to hit ten different locations is slow, as each call to Uber needs to be made



separately. Once it has the Uber fare estimate for all the 10 locations, it sends it back to Google Maps API for display.

### **App Server: Heroku**

We deployed our web application to Heroku so that it could be hosted publicly on the web, which proved to be an informative learning exercise. Moreover, it compelled our team to control the project's environment across different machines, which could vary widely in base configurations and setups, in a disciplined and sanitary manner. In order to successfully deploy the app to Heroku, our team had to install new project dependencies and complete development within a virtual environment.

In addition, we connected the Heroku application to our Github repository so that upon any new commits by team contributors to the repository's master branch, it re-deployed the web application, including installing any new project dependencies on the production server. This allowed us to benefit from both version control management via Github, and continuous integration via Heroku.

Finally, we hosted our PostgreSQL database on Heroku, using the Heroku Postgres service, which made it simple for the web application to interact with the database live on the web.

### **Database: PostgreSQL (Hosted on Heroku)**

We configured a PostgreSQL database as a persistent data source for storing and retrieving the short and long URLs. PostgreSQL was used as a learning experience, for gaining practice with (1) setting up a local instance of the database, (2) syncing the database with the Django project (via preparing and making database migrations), and (3) backing-up and pushing the local database to the live instance on Heroku. Django completed the database's migrations based on the models that are specified in a Django app's models.py file. Our project did not leverage the full power of a PostgreSQL database (e.g., PostGIS and JSON schemas), but we look forward to implementing these capabilities in the future.

### **File Server: Amazon Simple Storage Service (S3)**

By default, Django does not support serving static files (e.g., images, CSS, Javascript) in production.<sup>1</sup> As a result, we hosted our static files using Amazon's S3 server and made them publicly-accessible so that our Heroku-hosted application could access the files.

### **Challenges Faced**

Different team members worked on different APIs with the hope of integrating the technologies during the final stage. However, working in JavaScript vs Python made sending data incompatible in certain situations. One of the takeaways of this project was to streamline the technologies used and communicate regularly. This proved especially true for our work with these three different APIs.

Google Maps API is designed to be used using Javascript. While Uber and Yelp are being called from the server backend. The biggest challenge we faced was to call the Google API to get geographic coordinates and then send it to Yelp API. But since Google API works on the client side JS while Yelp works on server side python code, we were having difficulty passing the coordinates from Google API to Yelp API and then get the results back

---

<sup>1</sup> <https://devcenter.heroku.com/articles/django-assets>

from Yelp to Google. We decided on making an AJAX call from the JS to the server, but again we did not want to reload the page and hence did not want to send an HTML response back from the server. In the end we ended up taking the coordinates from the Google API, making an AJAX call to the server, passing the coordinates to the Yelp API, obtaining the response from Yelp, converting it to JSON, returning it to the client side and using JS to plot the results on the map. This was a longer work around as we were working on different technologies, but we managed to make them work together.

## **Future Work**

Future work could involve adding bicycle, foot, and bus routes as well. Also, the App could give the functionality to include search in other areas and not just the nearby locations. Different categories of Uber fares can be included (X, XL, Black etc.). The App could also include the functionality that would allow booking Uber. The Web App can also be transformed to a mobile App. The App could also include user profiles to allow the users to save their searches and favorite places. Based on the saved places, the App could also suggest similar restaurants to the user. The App potentially can be expanded to add a lot more features.