

Project Name: Credible Edibles

Team Name: LipgLOSSisPoppin

Team Members: Stephanie Yoon, Olivia Gallager, Shariar Kabir, Lorenz Vargas

The Idea:

Credible Edibles will take in an address or the user's location, as well as their preferences for current location/address, distance, and ratings, and create a map of restaurants near them that match their input as closely as possible with the addresses and ratings (?) of the restaurants listed below. The project would use the Yelp, Mapbox, and Google Maps APIs.

Program Components:

- devlog
- README
- app.py
- utils/
 - rauth/
 - __about__.py
 - __init__.pyc
 - oauth.py
 - service.pyc
 - utils.py
 - __about__.pyc
 - compat.py
 - oauth.pyc
 - session.py
 - utils.pyc
 - __init__.py
 - compat.pyc
 - service.py
 - session.pyc
 - geomapbox.py
 - googlemaps.py
 - yelp.py
- templates/
 - basic.html
 - homepage.html
 - results.html
- static/
 - bootstrap/
 - css/
 - style.css
 - bootstrap-theme.min.css
 - bootstrap-theme.min.css.map

- bootstrap.min.css
- bootstrap.min.css.map
- fonts/
 - glyphicons-halflings-regular.eot
 - glyphicons-halflings-regular.woff
 - glyphicons-halflings-regular.svg
 - glyphicons-halflings-regular.woff2
 - glyphicons-halflings-regular.ttf
- js/
 - bootstrap.min.js
 - npm.js

Components Breakdown:

- app.py
 - homepage()
 - Renders homepage.html
 - results()
 - Takes in form responses from homepage (things like location, rating, etc.).
 - Sets **loc** to be a list of two elements, lat and lng, either from locate() or geocode(address).
 - Sets **businessesList** to makeBusinessesList(getSearchResults(lat, long, rating, distance))
 - Sets **jsList** to makeJsList(businessesList)
 - Renders results.html, loc, **jsList**.
- geomapbox.py
 - geocode(address)
 - Constructs and opens a query string. (Note: Set limit to 1.) Loads the JSON object string to get a dictionary.
 - Within this dictionary is a list called “features”, within which there is a dictionary, within which there is a dictionary called “geometry”, within which there is a list called “coordinates”. Set loc to be “coordinates”.
 - Return **loc**.
- googlemaps.py
 - locate()
 - Constructs and opens a query string. Loads the JSON object string to get a dictionary. Within this dictionary is a dictionary with the key “location.” Create a list called loc, set loc[0] to be the value of the “lat” key and loc[1] to be the value of the “lng” key of the “location” dictionary.
 - Return **loc**.
- yelp.py
 - makeBusinessesList(searchResults)

- Constructs a list, called `businessesList`, based on the return value `searchResults` helper function.
- Structured as a dictionary where the key is the name, and the value pair is a list of attributes to that restaurant
- Returns **`businessesList`**.
- `makeJsList(businessesList)`
 - This returns a list of dictionaries, called `jsList`, that's consistent with the JS format required to display map on mapbox
 - Takes data from `businessesList`
 - Each venue/restaurant gets a dictionary
 - Keys: "type" , "geometry", "properties" (which includes "title")
 - Returns **`jsList`**.
- `getSearchResults(lat, long, rating, distance)`
 - Takes input from mapbox/google location data and form data, passed as parameters when called by app*
 - Constructs and opens a query string based on the parameters. Loads the resultant JSON object string to get a dictionary, called `searchResults`.
 - Returns a dictionary of all necessary information, without extraneous info.
- `basic.html`
 - HTML document, {% block head %}, {% block body %}
- `homepage.html`
 - Form
 - Radio: loc
 - Current Location
 - Text Input: address
 - Radio: distance
 - Bird's Eye
 - Walking
 - 4 Blocks
 - Driving
 - Radio: rating
 - *
 - **
 - ***
 - ****
 - *****
 - Form action = /results/
- `results.html`
 - Map

- Wrapping JS for loading the map, centered at `{{loc}}`, with features in `{{jsList}}`, adding layer with labels.
- List
 - Loops through the elements of the jsList, displaying the names, addresses, and ratings.

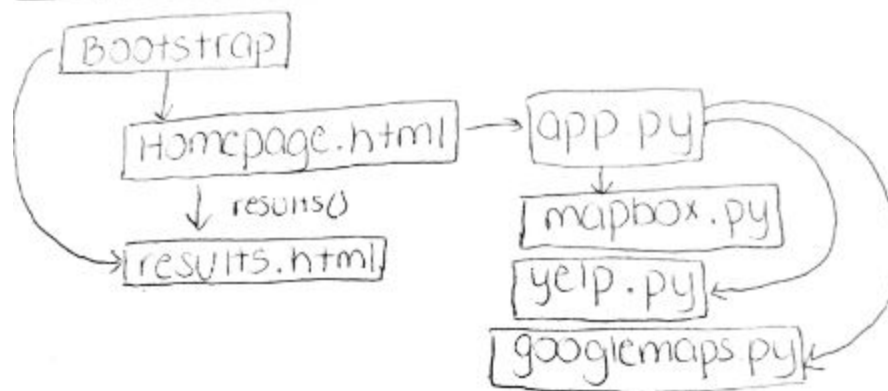
Site Map: <https://app.moqups.com/syoon1/O1bKRE53p9/view>

(See next page for images.)

A browser window titled "CredibleEdibles" showing the homepage at `http://127.0.0.1:5000/homepage.html`. The page has a title "Credible Edibles" and a subtitle "by LipgLOSSisPoppin". Below this are several filter sections: "Location" with radio buttons for "Current Location" and "Address" (the latter is selected and has a text input field); "Price" with radio buttons for "\$", "\$\$", "\$\$\$", and "\$\$\$\$"; "Distance" with radio buttons for "Bird's Eye", "4 Blocks", "Walking", and "Driving"; and "Ratings" with radio buttons for "*", "**", "***", "****", and "*****". At the bottom is a "Find Credible Edibles" button.

A browser window titled "CredibleEdibles" showing the results page at `http://127.0.0.1:5000/results.html`. The page has a title "Credible Edibles" and a subtitle "by LipgLOSSisPoppin". At the top is a "Back to Home" button. Below this is a map showing a street grid with a location pin. Under the map, there are two restaurant entries: "Restaurant 1 *****" with an "Address" label, and "Restaurant 2 *****" with an "Address" label. A vertical scrollbar is visible on the right side of the page.

Component map



Tasks/Assignments:

Stephanie - Frontend (HTML, CSS, Foundation), JS

Olivia - Yelp

Lorenz - Mapbox and Google Maps

Shariar - Flask app

Timeline:

Monday - Design Doc

Tuesday - HTML, CSS, JS, app.py

Wednesday - yelp.py, mapbox.py, googlemaps.py

Thursday - Testing

Notes/Resources:

Project Info -

<http://www.stuycs.org/courses/software-development/dw/projects/project1getyourdataoffofmeyo-udamndirtyapis>

locate() - <https://developers.google.com/maps/documentation/geolocation/intro>

geocode() - <https://www.mapbox.com/api-documentation/?language=Python#geocoding>

JS and makeJSList() - <https://www.mapbox.com/mapbox-gl-js/example/geojson-markers/>

Color Scheme: <https://coolors.co/>

Bootstrap: <http://getbootstrap.com/components/>

Yelp Authentication: <https://www.yelp.com/developers/documentation/v2/authentication>

Yelp Search: https://www.yelp.com/developers/documentation/v2/search_api

- **Consumer Key:** Ov-ytNFKKBZfdHYTkdQAoQ
- **Consumer Secret:** b1z2H2DCRH4hf4aQo1zqaTyYYJA
- **Token:** ZCsTHSJC7DAISVmSQSoe7pxQDDCH_Thk
- **Token Secret:** CNMwy2PUHaXTyXTQe4Qv2lk4BuE
- Yelp uses OAuth 1.0a for authenticating API requests as per the OAuth [specification](#) (Accessing Protected Resources).

- oauth_consumer_key
 - Your OAuth consumer key
- oauth_token
 - The access token obtained
- oauth_signature_method
 - hmac-sha1
- oauth_signature
 - The generated request signature, signed with the oauth_token_secret obtained
- oauth_timestamp
 - Timestamp for the request in seconds since the Unix epoch.
- oauth_nonce
 - A unique string randomly generated per request.
- These parameters may be passed in the HTTP (Authorization) header as URL query keys or in the POST data.