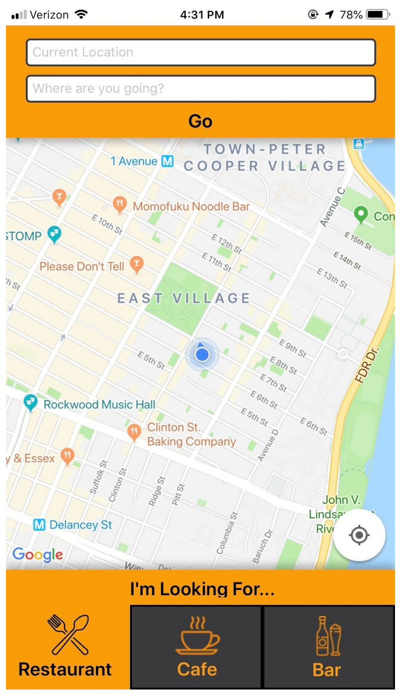
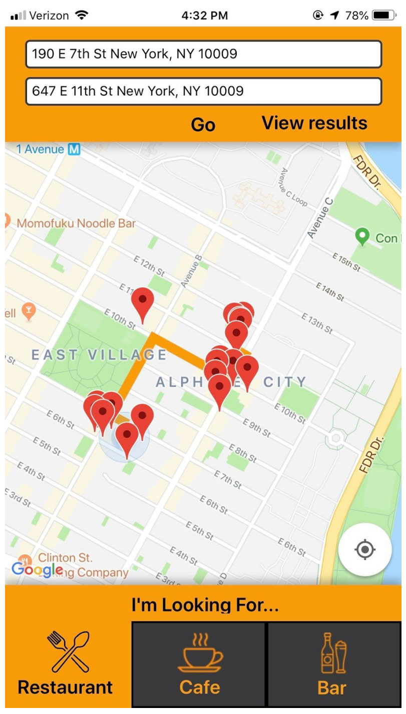
Path

By Abhi Dankar, Leshya Bracaglia, and Roy Lee

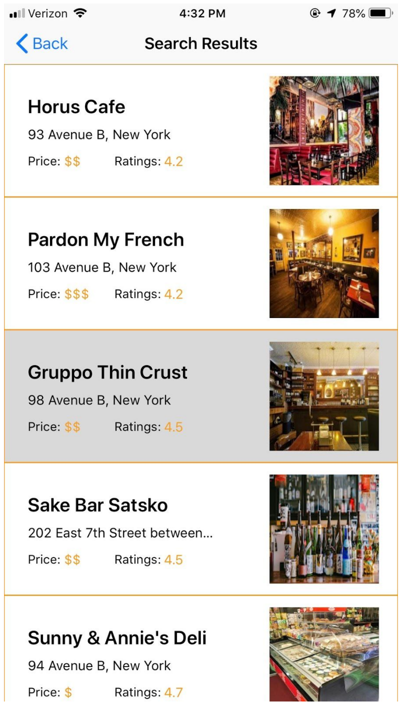
User Manual with Screenshots



Once the app loads, the user is faced with this starting screen. They may then enter their current address and the address of their destination. They can also choose between a search for restaurants, cafes and coffee shops, or bars along the bottom.



Once the user hits go, Path contacts the Google Routes API and displays the fastest route to their destination. In addition, pins for the locations matching their choice will display as pins on the map. These pins will only constitute businesses that lie within or very close to the resulting path. Once the pins have dropped, a “View Results” button will appear to the right of the “Go” button.



Verizon 4:32 PM 78%

Back Search Results

Horus Cafe
93 Avenue B, New York
Price: $$ Ratings: 4.2

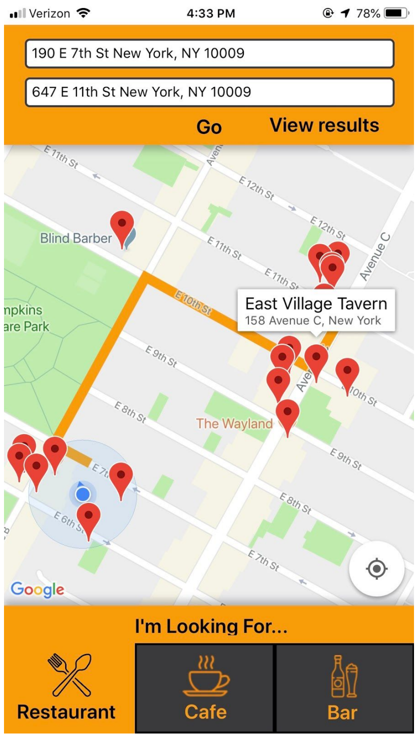
Pardon My French
103 Avenue B, New York
Price: $$$ Ratings: 4.2

Gruppo Thin Crust
98 Avenue B, New York
Price: $$ Ratings: 4.5

Sake Bar Satsko
202 East 7th Street between...
Price: $$ Ratings: 4.5

Sunny & Annie's Deli
94 Avenue B, New York
Price: $ Ratings: 4.7

When the “View Results” button is pressed, a table or list view will appear, showing the search results along with their addresses, price range, and Google Places ratings.



Verizon 4:33 PM 78%

190 E 7th St New York, NY 10009

647 E 11th St New York, NY 10009

The users can then decide on their restaurant of choice based on their location on the path, the price points of the restaurants, and the ratings. They can then follow the path in real-time and approach the pins along their route to find their quick stops. (Currently, on xCode, the user location marked by the blue marker will not show the correct user location, but when using this app on an actual iPhone device, the blue marker will be in the correct location so that users can see their location as they follow along the path)

Technical Description and Logic

For our app, we utilized Google Maps, Google Places, Google Directions, Swifty JSON, and CoreLocation, and CocoaPods to set these up. The first lets us display our map and route, the second allows us to get businesses, the third lets us pull a route, the fourth allows our app to use location features, and the fifth is used to parse the JSON data returned by Google. Our code works by taking a start and endpoint from the user, and then displays a route. We use dispatch groups and completion handlers to make sure we get the data from the APIs as they are supposed to, otherwise we encountered many pieces of code executing out of order and attempts to unwrap null data. The method fetchNearPlacesCoordinate in the GoogleDataProvider uses our API key to pull the places we need from Google. Our findNearbyPlaces method uses a value called DistanceFilter to make sure we don't take more points than we need from our route, skipping points if they're too close to the previous one. With the array that we create from the first method I mentioned, the second one mentioned can search through those places to find nearby businesses around them. The businesses are displayed via markers on the map, and we also included a second view controller to display the restaurants in an easy to read fashion.

Individual Responsibilities

When we were tackling the development of the major features for the application, we usually met up and worked together so that none of us were truly bounded to merely a single functionality. However, we did have main areas that each one of us were initially assigned to. Leshya worked primarily on the user interface and the front end of the application. Roy was assigned to the implementation of the Google Places API and json parsing. Abhi was assigned to the implementation of the Google Maps and Directions API to fetch the routes and location points. Once again, we ended up collaborating on all aspects of the application as we figured out during the development process that implementing teamwork was the best and most efficient way to create this application.

Future Enhancements

In the future, we would like to implement some sort of auto-complete in both of the start and end destination text boxes. We would also like to implement a login feature for users so that they could have options to save their frequently used paths and star restaurants that they want to revisit in the future. Also, we would also like to implement a feature that limits the search of restaurants to a low, user-specified number and randomly generates results bounded to that number. This would enhance our application's mission of time-efficiency to users who are on a time constraint to quickly find places to eat.