Overview

The goal of this exercise is to write a simple iOS Application that will display information about currently open food trucks, given a source of food truck data from the City and County of San Francisco's API.

The Task

Data

The San Francisco City website hosts a public data source of food trucks (https://data.sfgov.org/Economy-and-Community/Mobile-Food-Schedule/jjew-r69b). The data can be accessed in a number of forms, including JSON, CSV, and XML. How you access the data is up to you, but you can find some useful information about making an API request to this data source here (https://dev.socrata.com/foundry/data.sfgov.org/bbb8-hzi6).

You shouldn't need to get an API key for the purposes of this submission, it's unlikely you will hit the request limits.

The Problem

Write an iOS application that connects to the data source provided above and displays the data. Work through the steps below to implement the application in stages. It's worth reading through all the steps even if you don't make it to the end. You should spend 3 - 4hrs on the task including writeup. We don't necessarily expect you to make it to the end and prefer you concentrate on the quality of your implementation than completing all the tasks in the time.

1. Connect up to the data source and retrieve the data. *We suggest you think carefully about how you architect this in order to achieve the following steps.*

2. On the first screen, show a list of food trucks that are currently open. For example, if I opened the app at noon on a Friday, I would see a list with the details of all the food trucks that are open then. In this list, please display the food truck name, address, description, and the times that they are open. Your app should launch to this list screen.

3. Add a second screen, display the currently open food trucks on a map. The map should be zoomed to fit all of the pins on the map. When a user selects a pin on the map, show them the details about the corresponding food truck. These details will again include the

food truck name, address, description, and the times that they are open. The user should always be able to switch between this screen and the previous list view.

Additional notes

Use the screenshots provided in the example below as a model for what your UI should look like. There should be no need for you to create any icons or to design the UI yourself.

Use of any external libraries or packages is permitted.

Please don't use SwiftUI

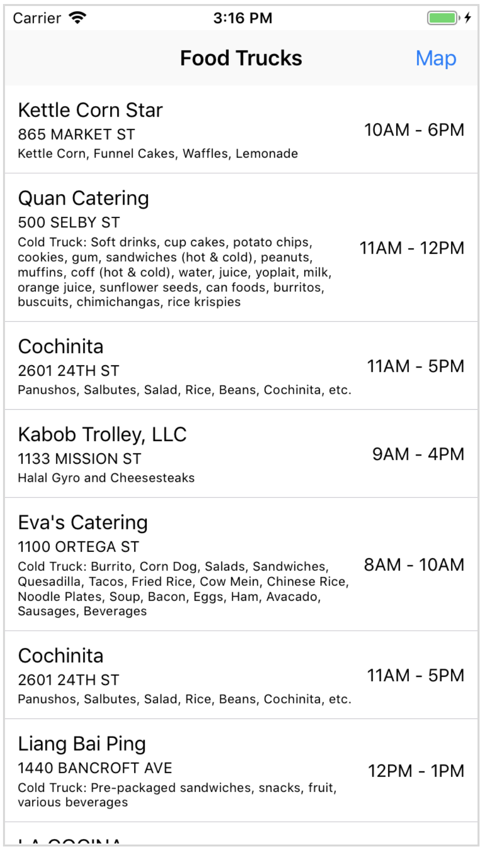
Nice to have

Include a few tests (anywhere from 2 - 5 test cases should suffice) demonstrating the testability of your code.
Tips: Write code in such a way that it is easy to test in the first place. You can use dependency injection, protocols/interfaces to make your code more testable.

We'd like to be respectful of your time. And so as stated above, a few tests should suffice. This is a nice to have and so your submission will not be penalized in case you run out of time and couldn't add tests

Helpful Documentation
- MapKit
    - MKMapView

Example


Carrier 3:16 PM

Food Trucks Map

Kettle Corn Star
865 MARKET ST
Kettle Corn, Funnel Cakes, Waffles, Lemonade
10AM - 6PM

Quan Catering
500 SELBY ST
Cold Truck: Soft drinks, cup cakes, potato chips, cookies, gum, sandwiches (hot & cold), peanuts, muffins, coff (hot & cold), water, juice, yoplait, milk, orange juice, sunflower seeds, can foods, burritos, buscuits, chimichangas, rice krispies
11AM - 12PM

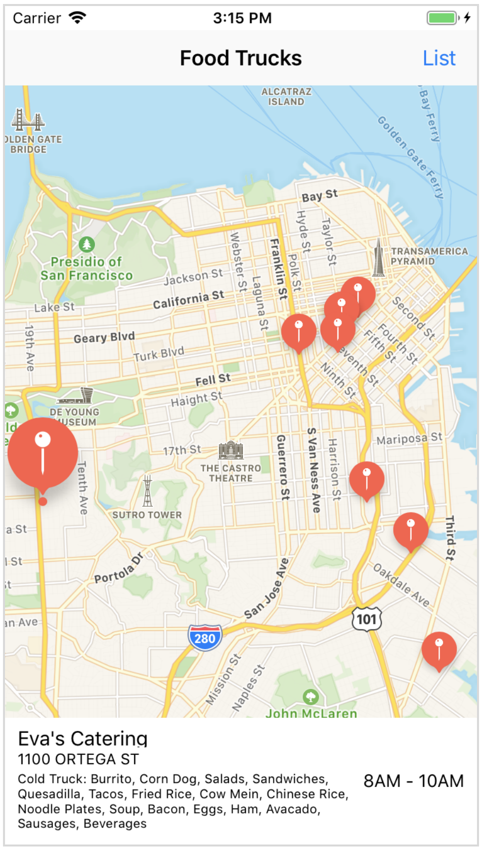
Cochinita
2601 24TH ST
Panushos, Salbutes, Salad, Rice, Beans, Cochinita, etc.
11AM - 5PM

Kabob Trolley, LLC
1133 MISSION ST
Halal Gyro and Cheesesteaks
9AM - 4PM

Eva's Catering
1100 ORTEGA ST
Cold Truck: Burrito, Corn Dog, Salads, Sandwiches, Quesadilla, Tacos, Fried Rice, Cow Mein, Chinese Rice, Noodle Plates, Soup, Bacon, Eggs, Ham, Avacado, Sausages, Beverages
8AM - 10AM

Cochinita
2601 24TH ST
Panushos, Salbutes, Salad, Rice, Beans, Cochinita, etc.
11AM - 5PM

Liang Bai Ping
1440 BANCROFT AVE
Cold Truck: Pre-packaged sandwiches, snacks, fruit, various beverages
12PM - 1PM



Eva's Catering
1100 ORTEGA ST
Cold Truck: Burrito, Corn Dog, Salads, Sandwiches, Quesadilla, Tacos, Fried Rice, Cow Mein, Chinese Rice, Noodle Plates, Soup, Bacon, Eggs, Ham, Avacado, Sausages, Beverages
8AM - 10AM

Criteria

We will evaluate programs primarily on code quality and output correctness.

For quality, we expect code to be easy to read and maintain, performant, and reliable. You should submit code that you are proud to have written. We will also be judging based on architectural decisions and reasoning behind them in the writeup.

For correctness, we expect that whenever the app is run, it displays all trucks open at that time and no food trucks that are closed at that time, with the UI as described above.

Submitting your work

Please email us a zipped folder containing your work. Your submission should include:

1. Your code.

    a. We should be able to open, build and run this in the latest **non-beta** version of Xcode
    b. Make sure you include any dependencies
    c. Make sure the code doesn’t include anything that would stop it compiling on a different computer.
2. A one or two paragraph write-up describing, at a high level, the design decisions you made in building this application.
    a. Include anything you would do differently if you were asked to build upon this for a full-scale production application in the App Store. Please focus on the technical differences between your solution and a production application, rather than on the product differences.
3. Please don’t upload your code to any public Git repositories

Submission Notes

We recommend that you spend 3-4 hours on this project.

We don’t expect your write-up to take more than thirty minutes to complete. Therefore, keep your response high-level and not more than two paragraphs. Do spend time polishing and packaging up your submission so it is easy to build, run, and review.

We also don’t expect that spending significantly more than the recommended time will be beneficial in improving your response. The recommended time frame should be sufficient for building a complete solution and we aren’t looking for any functionality beyond what’s described above. In your on-site interview, we will discuss what you would change and improve with more time and in an application that included more functionality.

While we expect more experienced candidates to manage a complete solution in 3-4hrs more junior candidates may struggle. As we don’t expect you to spend large amounts of time on this please get as far as you can in the 3-4hr time window. If you opt to spend longer than this to complete the challenge please note how long it took you in your submission (this helps us better understand how long candidates are actually taking so we can realign if necessary).