NYC Planning – Open Source Engineering Team

Code Challenge: Delivery Manager

You have been tasked with building a simple full stack application for delivery drivers to use while completing deliveries. You have been given a file of GeoJSON containing a FeatureCollection with 5 Point Features. Each Feature represents a delivery in New York City that the delivery driver using your application must complete. In addition to the geometry containing the destination coordinates for the delivery, each feature has a "properties" object containing data about that delivery including an ID, the recipient's name, notes, and a boolean "delivered" flag that indicates if a given delivery has been completed.

Your solution should contain two applications: a REST API for reading and updating deliveries, and a front-end web application that uses that API to show delivery drivers that data and allows them to update it.

You will be asked to submit all of your code in one zip file but you can build the API and front-end in the same or separate git repositories, whichever you prefer. Your front-end should be built with HTML, CSS, and JS (or a toolset that compiles down to them) but your API may be built in whichever language you would like. In both cases, feel free to use whichever open source frameworks or libraries will help you complete the tasks in the allotted time. If you have an existing boilerplate project or know of a CLI tool that will help you quickly bootstrap your project, feel free to use those as well. However, be sure to document to us how to install dependencies and run your projects. If we have trouble running them, we may reach out via email with questions. You should spend no more than 4-6 hours on this task.

The API

Your API should return JSON responses and have two endpoints. The first should be a GET request to "/deliveries" that simply returns the geojson you have been given in its JSON response. The second should be a PUT or POST to "/deliveries/:id", where ":id" is a parameter corresponding to the "id" property of the delivery that is to be updated. This endpoint should receive a request body that contains a value for "notes" and/or "delivered" and update those values in the JSON file for the delivery with the given id (if the request body has "notes" but not "delivered", the value for "delivered" should be left as-is, and vice versa). This endpoint should return a response with a 404 response code if no delivery for the given id is found.

Your API code should read and write directly to a JSON file containing the data we have supplied to you. You may find it easiest to start by simply moving the file we give you next to your API code and read/write to it from there.

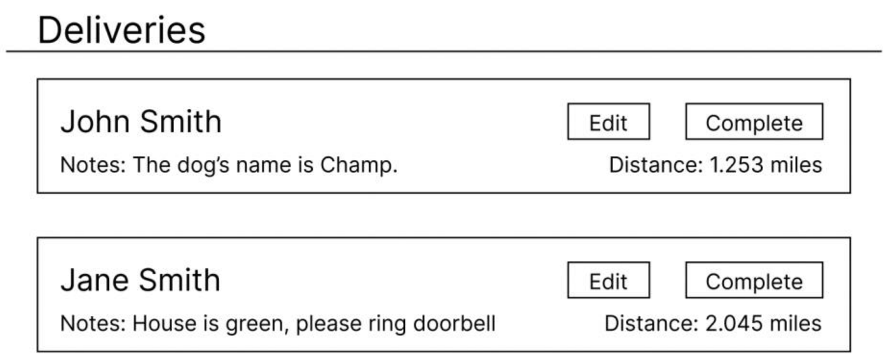
The Front-End

The front-end of your application should be a single web page that uses the API you built to do the following:

- When the page is visited, calls the "/deliveries" endpoint in your API to get the list of deliveries and display them in a simple list. For each delivery, the driver needs to be able to see the recipient name and delivery notes.
- For each delivery, calculate the distance from the user's position to each delivery using the browser's Geolocation API (https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API). The distance you calculate should be just the distance of a straight line between the user's location and delivery destination (I.e. "as the crow flies" distance) and you may display the distance in any unit of measurement you would like, so long as it is clear which you chose. Display the distance in your list alongside the recipient name and notes. You may find it helpful to search for open source libraries that allow you to easily calculate the distance between two GeoJSON Points or lat/long pairs.
- Drivers should also be able to make updates to the "notes" for a delivery and mark it as "delivered" via the PUT or POST endpoint in your API. You may implement this via a form on the page that is separate to the list or within the list itself.

- Your list should be sorted by distance in ascending order (closest deliveries first) and only show deliveries with a "delivered" value of "false". To be clear, you do not need to recalculate distance if the user moves while they have the app open, your app should get the user's location once when it first loads and use that location for distance for the entire session.

Here is a low-fidelity wireframe of what your deliveries list might look like:



Ultimately, we will be assessing your work based on the quality of your code and how well it meets requirements. With that in mind, please feel free to keep the visual design and layout of your front-end as simple as you would like. We want to see that you can build something that looks professional and is user friendly, and that you can write clean and concise styles. We encourage you to build toward "minimum viable product" for styling and then return to improving them and using your creativity if you have time.

Stretch Goal

If you complete **all** of the requirements outlined above and have time left, you may attempt to add an interactive map to your front-end with dots at the coordinates for each delivery overlaid on a map of New York City. Clicking on or hovering over those dots should display a tooltip with the recipient name and notes properties. You may find it helpful to search for open source JS libraries that allow you to easily display GeoJSON on top of an underlying "basemap".

Documentation

You may document your code through commit messages, code comments, a README.md file, or any combination thereof. Treat these tools as a way to communicate your thinking to us, particularly for bits of code where it may not be obvious at first glance.

Submitting

Details about where to send your submission will be communicated via email. Good luck!