CARTO

Front-end Engineer assignment

Intro

We really appreciate your dedication. We're going to give you proper support during the test, so apart from this document you should receive by email an invitation to join a #slack channel in CARTO, where you can ask questions to our engineers.

Asking questions is fine, don't be shy, we'll evaluate that positively. We prefer you ask questions rather than spend hours on your research to get stuck finally. See the **Support** section at the end of this doc to know more.

In this assignment we will evaluate the following aspects:

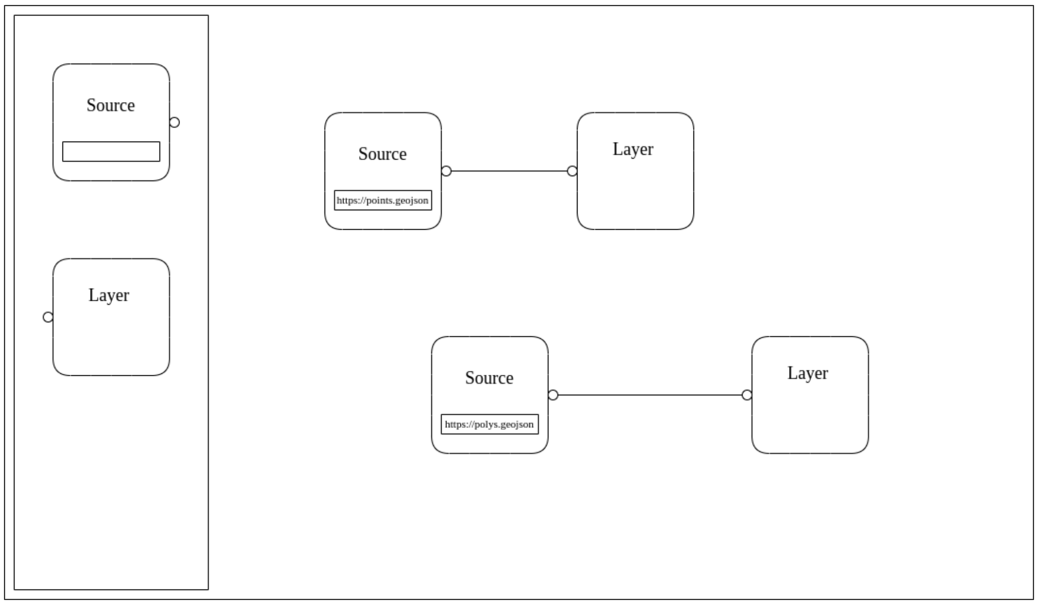
- **Simple and clean code**. You should develop a clear component-oriented solution (good naming, hierarchy, folder structure, and so on...). Avoid over-engineering, the simpler the better.
- **Communication**. During the whole test: in the slack channel, in the code, when delivering...

Description

We work on the development of geospatial analytics and visualization tools. This test consists of a simple approach to both aspects using the libraries that we use and contribute to, for our products.

Part 1 - Create a basic React app with ReactFlow

1. Develop a simple web application with the following features:
    a. Add nodes (drag & drop)
    b. Connect nodes with edges
    c. Delete nodes and edges
    d. Save and load the diagram state from the local storage
2. Implement the following custom nodes
    a. **Source** node: This input node contains one source port and an input text field called “url”, where the user can enter free-form text.
    b. **Layer** node: This output node contains one target port.

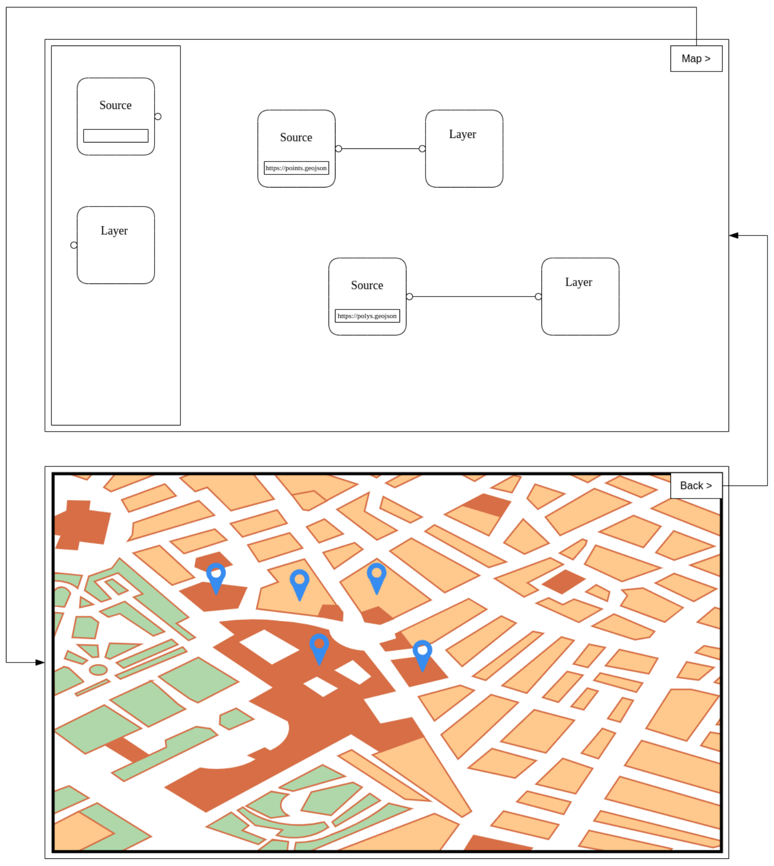


For more information, check the following example from ReactFlow: https://reactflow.dev/examples/nodes/custom-node.

CARTO

Part 2 - Add a map visualization with Deck.gl

1. Develop a map view from the diagram
    a. Render the map from the nodes defined in the diagram
    b. Add a hover tooltip to visualize the properties of each geometry
2. Implement a view transition system using the Map/Back buttons

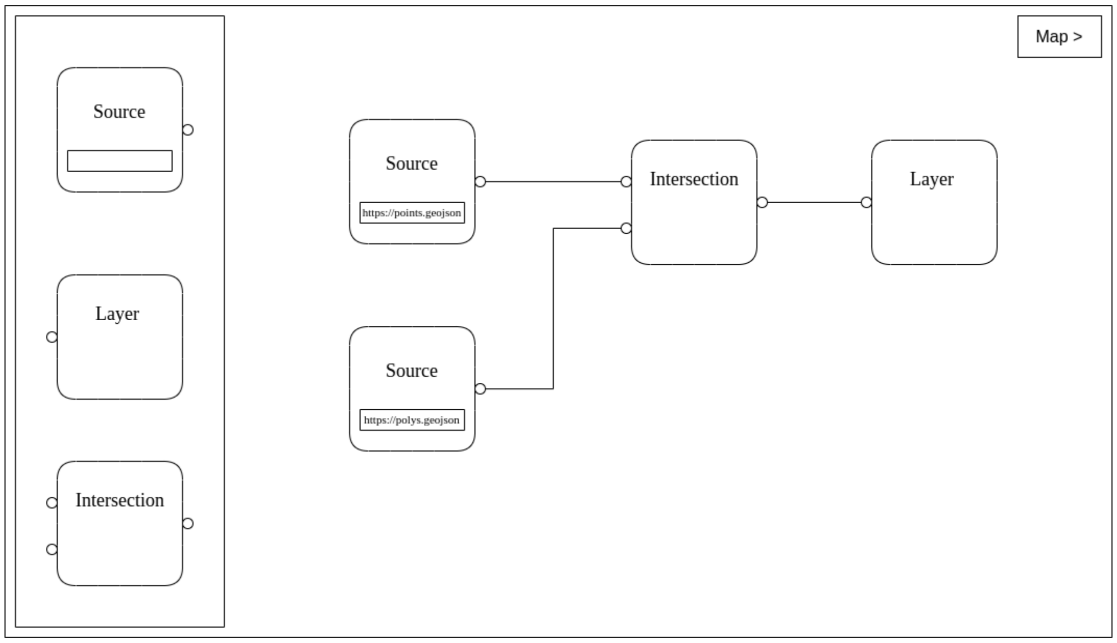


The map will detect and render the layers in the diagram, using their corresponding GeoJSON sources. The order of the layers will be determined by the vertical position in the diagram.

For more information, check the following example from Deck.gl: https://deck.gl/docs/api-reference/layers/geojson-layer.

Bonus (optional) - Support an intersection node

1. Implement a new type of node that intersects two source nodes
   a. **Intersection** node: input/output node with two source ports and one target port.



For more information, check the following example from Turf: https://github.com/Turfjs/turf/tree/master/packages/turf-intersect.

Tech stack

- React with Typescript
- If you use a UI lib, then you can rely on MaterialUI v5 and/or styled-components (choose whatever you feel more comfortable with). In any case, we would like you to use a basic “CSS in JS” approach.

CARTO

Delivery

Once you've finished your work, we would like to have:

- access to the git repo with the test
- some kind of README with instructions & any detail you may find relevant for us (eg. reasons to choose a technical approach in the code, things that could be improved, etc.)
- and ideally your feedback about the test!

Support

If you have any issues, you can contact Ana Sosa at amsosa@cartodb.com.