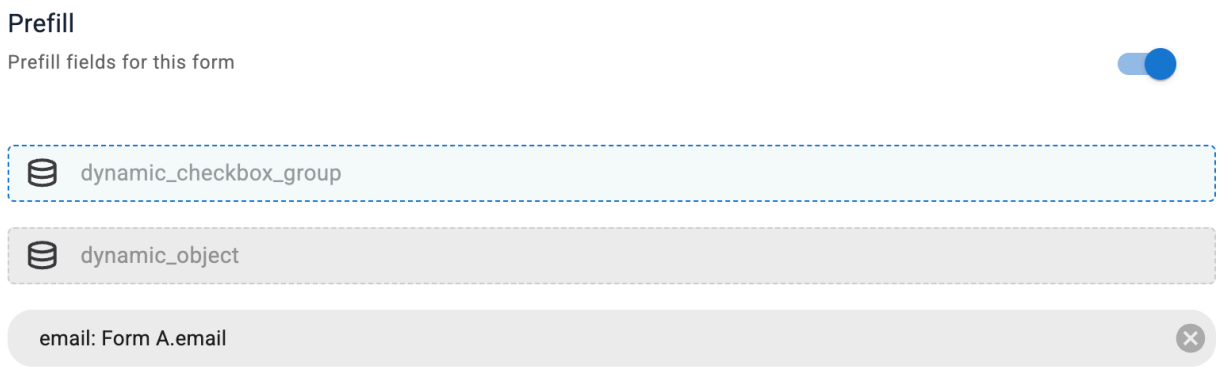
Journey Builder React Coding Challenge

In this challenge, you will reimplement a small portion of an app we're currently building at Avantos. The portion you'll work on is a node-based UI that shows a DAG of forms:

When a form has been submitted, the values from the form fields can be used to prefill the fields of a downstream form. E.g., values from Form A's fields can be used to prefill Form B's or Form C's fields.

First, you will use our docs to hit our `action-blueprint-graph-get` endpoint and the react flow docs to render the returned nodes and edges.

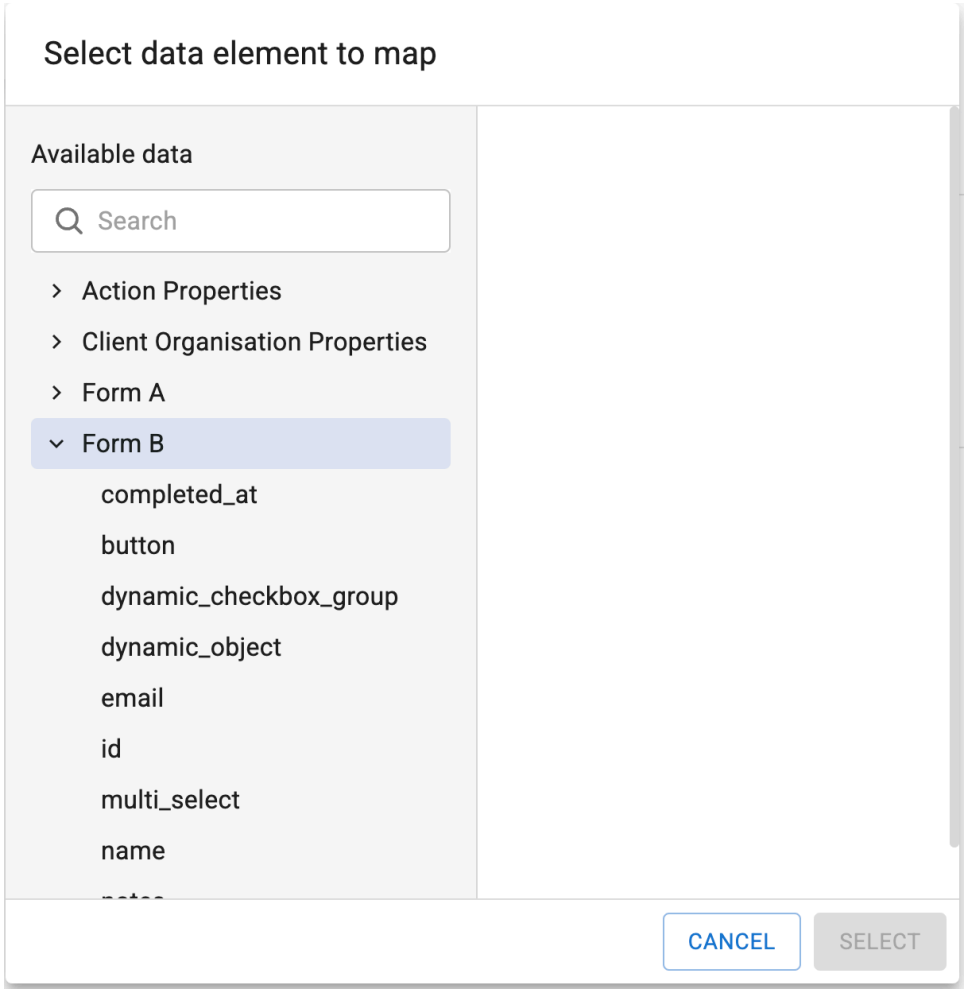
Next, you will implement the prefill UI for Forms. It doesn't need to be pretty, but this UI will need to view and edit the prefill mapping. We show this mapping when a user clicks a node and the mapping looks like this in our UI:



This shows the prefill configuration for three fields on Form D above. The first two fields are called "dynamic_checkbox_group" and "dynamic_object" and they currently have no prefilled data. The last field is called "email" and will be prefilled with the value from Form A's email field.

Clicking the X button on the far right of the email field clears the prefill configuration for that field. Clicking a field without a configuration opens this

modal:



In this modal, we see 3 types of data that can be used to prefill a form:

1. Form fields of forms that Form D directly depends on (Form B)
2. Form fields of forms that Form D transitively depends on (Form A)
3. Global data (Action Properties and Client Organization Properties)

1 and 2 will require traversing the form DAG. For 3, you can ignore Action Properties and Client Organization Properties and use whatever global data you want.

You should design your code so that any combination of these data sources can be easily used without code changes. Moreover, you should design for easy support of future, new data sources.

Prerequisites

- Use
  - React
  - TypeScript (optional but *strongly* recommended)
  - Create React App, Vite, or Next.js
- This repo has a mock server with the aforementioned `action-blueprint-graph-get` endpoint.

Rules

SUBMISSION: Your submission should contain:

1. A link to your solution as a Github repo.
2. A link to a screen recording of you working on the solution. In this video, we are looking to see that you can code independently without relying entirely on an LLM. We don't need a video of all the code you write. Just 30 minutes of you working should do the trick. An unlisted youtube video link works great for this. **If you do not submit a video link we will not review your submission.**

TIME LIMIT: Send us a link to your Github repository within 4 working/business days from the date you receive this. You may submit your work earlier.

Evaluation Criteria

We will be particularly interested in:

1. Code Organization
   - Clear separation of concerns
   - Well-defined interfaces between components
   - Thoughtful component hierarchy and composition
2. Extensibility
   - **Does the project have good tests?**
   - How easily new features can be added?

- Reusable and composable React components

3. Documentation
    - How do I run this locally?
    - How do I extend with new data sources?
    - What patterns should I be paying attention to?
4. Code Quality
    - Clean, readable code
    - Reasonable, readable var names
    - Appropriate use of modern React practice