Acre Technical Exercise (DDF)

Acre Platforms Limited
Registered in England and Wales
11508306

Samuel House
6 St Alban's Street,
London, United Kingdom,
SW1Y 4SQ

hello@acresoftware.com
+44 208 132 4240

Technical test

Specification

When applying for a mortgage, "fact-find" is a major part of a customer's journey. The data brokers are required to collect ranges from personal information, to relationships and dependants, to financial circumstances. These requirements can also change from brokerage to brokerage, meaning that the questions you may be asked by one broker can be different to those you might be asked by another.

Developing a means of collecting data that is both flexible and robust is a major challenge within our domain. Hard-coding forms for one brokerage means compromising on the needs of another. So how can we address this?

Enter: data-driven forms.

Part 1 - Displaying data-driven input fields

Design a JSON schema capable of describing the form structure mentioned above. It will need fields for:

- Job title
- Employment period (radio button, "Fixed term" or "Permanent")
- Current employment (checkbox, toggle yes/no)
- Start date
- End date
- Employer name
- Contact number
- Email address

Now, provide an implementation capable of processing this JSON object and subsequently rendering a form containing each of the corresponding input fields. Each field's values should be captured by the form's state, which should

be logged to the browser console or otherwise outputted to the user when submitted. (No need to implement a backend at present.)

Part 2 - Input validation

Currently, the application has no input validation, meaning a user is free to enter any email address or phone number, as well as leaving certain required fields blank.

For this part of the exercise, extend your implementation such that users can define how input fields should be validated by modifying their schema. If an input field requires validation, the end user should see a warning/error message if the input's current value is invalid.

Assume the following validation requirements:

- Job title (required)
- Employment period (required)
- Current employment
- Start date
- End date
- Employer name (required)
- Contact number (valid phone number)
- Email address (valid email address)

Part 3 - Conditional rendering

In its current state, the form displays both the "start date" and the "end date", even if the user has checked "current employment" (meaning there is no end date).

Extend your implementation such that users can define dependencies between fields within the schema, thus conditionally render them based on form state.

For example: If the user has checked 'current employment', hide the 'end date' field.

Part 4 – Dynamic Journeys

The details covered in Parts 1-3 are a small part of fact-find. There are a large number of data points that we may want to collect in different order depending on the case type and scenarios. (e.g., for a FTB the Deposit information is one of the key questions and likely to be one of the first questions we ask; for a Remortgage we instead want to ask if they wish to pay down any of the mortgage, and it is much later in the journey.

Give thought to how you would approach this problem, this can be either as code or as a write up.