Front-End Engineer - Technical Assessment

Overview

In this exercise, you will build a simple React application that consumes a JSON schema from an API and dynamically renders a UI based on the data provided. The goal is to demonstrate your ability to create flexible, component-based architectures and handle dynamic layouts. The challenge is to render the interface entirely driven by the schema you receive from the server.

Context

Server-driven UIs (SDUI) allow for dynamic user interfaces where the server dictates the structure and layout of the front end via a schema or configuration. This allows for rapid changes on the front end without redeployment.

Requirements

1. **Dynamic Rendering:** Render a dynamic UI based on a JSON schema which defines the UI components to be displayed, their order, type, properties, and possibly nested components (not part of this exercise).
2. **Schema-Driven Components:** Components and their behaviors should be generated based on the API schema. For instance, if the API defines a text input with a placeholder, label, and required field, the frontend should render a corresponding text field with these properties. For this exercise, only the components `page-title` and `checkbox-list-panel` need to be available.
3. **Error Handling:** Gracefully handle potential errors (e.g., invalid schema or missing required properties).

Instructions

1. **API Endpoint (Mocked)**
   Use a local JSON file to simulate the backend response:


```
{
  "title": "Wallee Test Page",
  "components": [
    {
      "type": "page-title",
      "label": "Payment Methods"
    },
    {
      "type": "checkbox-list-panel",
      "onSubmit": "logToConsole",
      "options": [{
        "value": "10001",
        "title": "American Express",
        "subtitle": "Worldline Acquiring",
        "imageUrl": "https://app-wallee.com/resource/web/image/payment/card-brand/american-express.svg"
      }, {
        "value": "10002",
        "title": "Visa",
        "subtitle": "PostFinance Acquiring",
        "imageUrl": "https://app-wallee.com/resource/web/image/payment/card-brand/visa.svg"
      }, {
        "value": "10003",
        "title": "Maestro",
        "subtitle": "Worldline Acquiring",
        "imageUrl": "https://app-wallee.com/resource/web/image/payment/card-brand/maestro.svg"
      }, {
```

```
        "value": "10004",
        "title": "Mastercard",
        "subtitle": "Worldline Acquiring",
        "imageUrl": "https://app-wallee.com/resource/web/image/payment/card-brand/mastercard.svg",
        "checked": true
      }, {
        "value": "10005",
        "title": "Direct Debit (SEPA)",
        "subtitle": "Adyen",
        "checked": true
      }, {
        "value": "10006",
        "title": "Digital Payments powered by Mastercard",
        "imageUrl": "https://app-wallee.com/resource/web/image/payment/card-brand/mastercard.svg"
      }, {
        "value": "10007",
        "title": "QR-Invoice Payment Method with PostFinance (disabled)",
        "subtitle": "QR-Invoice Processor with PostFinance (disabled)",
        "imageUrl": "https://app-wallee.com/resource/web/image/payment/method/invoice.svg",
        "disabled": true
      }]
    },
  ]
}
```

2. **Rendering Components**
   - Parse the JSON schema and dynamically render the appropriate React components based on the provided types.
   - Support the following component types:
     - `page-title`
     - `checkbox-list-panel` including simple search, reset and submit functionality
3. **Error States**
   - Handle potential error scenarios such as an invalid schema or a missing required property in a graceful way.
4. **UI Design**
   - The UI should follow basic design principles (use a simple layout and basic styling).
   - The components should be built according to the designs: https://www.figma.com/design/UGaBwwSkM1SjUHRU7DLsom/wallee-AG---Front-End-Engineer---Technical-Assessment?node-id=0-1&m=dev&t=452zxOnVlVi6ymWe-1

Deliverables

- A working React application that fulfills the requirements.
- A `README.md` file explaining how to run the project and how your solution works.
- The application and all relevant files and resources are stored in a public repository (e.g., GitHub or GitLab).
- Prepare to discuss your architectural choices, trade-offs, and any challenges you faced in the follow-up interview.

Tools/Stack

- React
- Typescript
- CSS, SCSS, or any CSS framework
- Optional: Supplementary libraries that assist in completing the task

Time Expectation

The assessment should take around **4-6 hours**, but it's up to you to decide how much time to invest in it.

Discussion Topics for the Follow-Up Interview

- How did you approach dynamic component rendering?
- Did you encounter any challenges, and how did you solve them?
- If given more time, how would you improve the project?
- How would you scale this solution for more complex UIs?