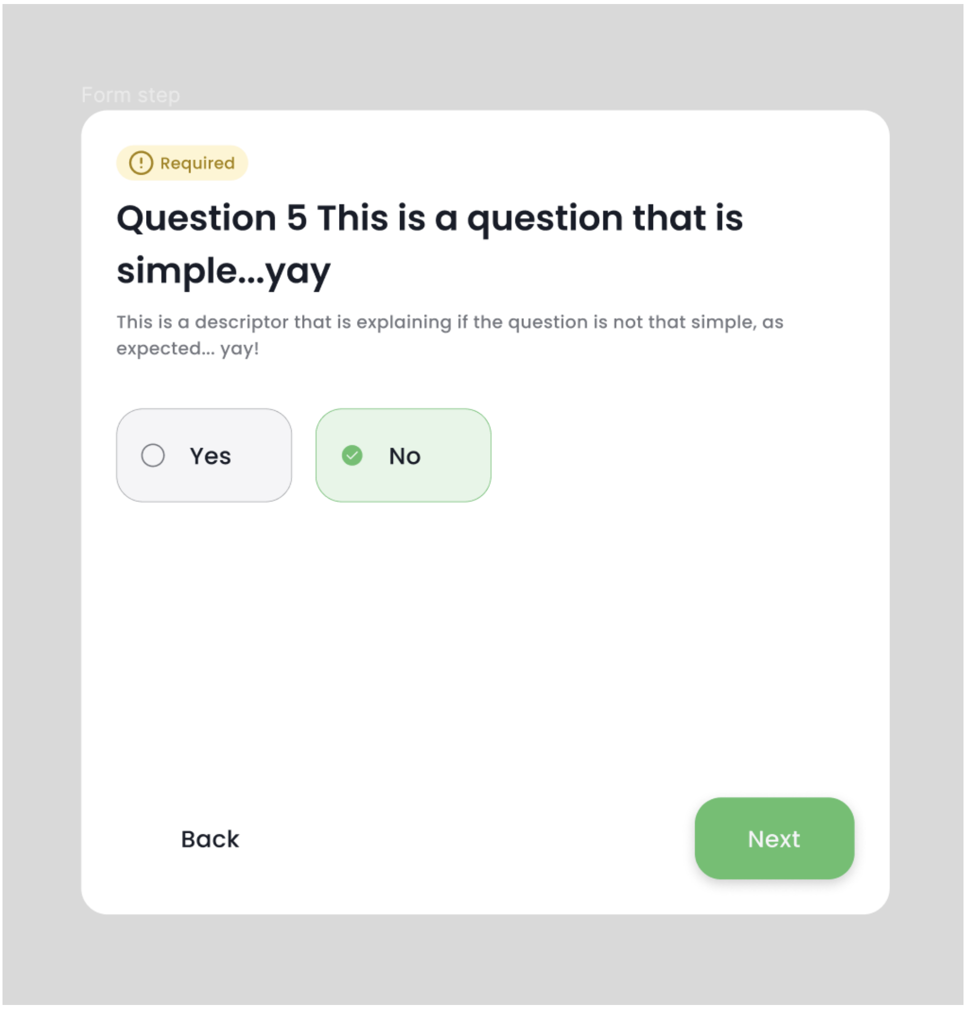
enter

Enter technical task for Senior Frontend Developer

`version 1.0.5`

Description

Your task is to write a program that takes JSON-formatted a form structure with render logic as an input data and render form split by steps with fields the following design:


Form step
Required
Question 5 This is a question that is simple...yay
This is a descriptor that is explaining if the question is not that simple, as expected... yay!
Yes
No
Back
Next

At enter you would be provided a Figma file, but here we would like you to do your best to derive the UI from an imperfect design file. Pixel perfectness is not required, but we appreciate getting the details right. Each step should contain fields and buttons to navigate between steps. The final step should prepare output structure in JSON format when user click a last primary button. Result should appear on a new screen as formatted text.

Input data

Input JSON consist of two main part corresponding forms steps presentation and logic for steps.

```
{
  steps: <steps collection>[]
  logic: <logic collection>[]
}
```

steps

A collection of form fields with handlers names in each of them:

```
{
  stepId: <string>
  fields: <fields collection>[]
  secondaryButtonLabel: <string>
  primaryButtonLabel: <string>
}
```

fields

Each step would contain one or more fields with following structure:

```
{
  fieldId: <string>
  type: <field type>
  properties: <field properties>
  validation: <validation matchers>
}
```

field type

Describes of field type for steps. In this task you will have only two field types:

```
<field type>: <input> | <radio>
```

properties

Fields properties are collection of key value pairs for form elements. In real live it corresponds of a browser form API but for the task it presents the main properties for render fields in steps:

```
{
  id: <string>
  placeholder: <string>
  label: <string>
  description: <string>
  value: <any>
}
```

validation

Validation contains matchers for a field. For this task you will work only with two matchers: email and required field.

```
{
  email: <boolean>
  required: <boolean>
}
```

logic

For this task we would ask to write some logic on client side to navigate a user to the form logic tree. The main idea is check fields in target step and navigate user to the destination step id.

```
{
  target: <step id>
  condition: <actions logic>[]
}
```

condition

Field id inside step, value for matching and distillation step id if match was successful.

```
{
  fieldId: <field id>
  value: <string>
  destination: <step id>
}
```

`See a full example in the attched json file.`

Output data

The output JSON should contain only fields answered by user according to logic tree.

```
{
  <field id>: <value>
}
```

Example of the output:

```
{
  "firstName": "Angela",
  "secondName": "Merkel",
  "born": "1954"
}
```

Tech and tools

We prefer to use TypeScript but JavaScript submissions consider too. 3rd party libraries and frameworks are allowed.

Approach and expectations

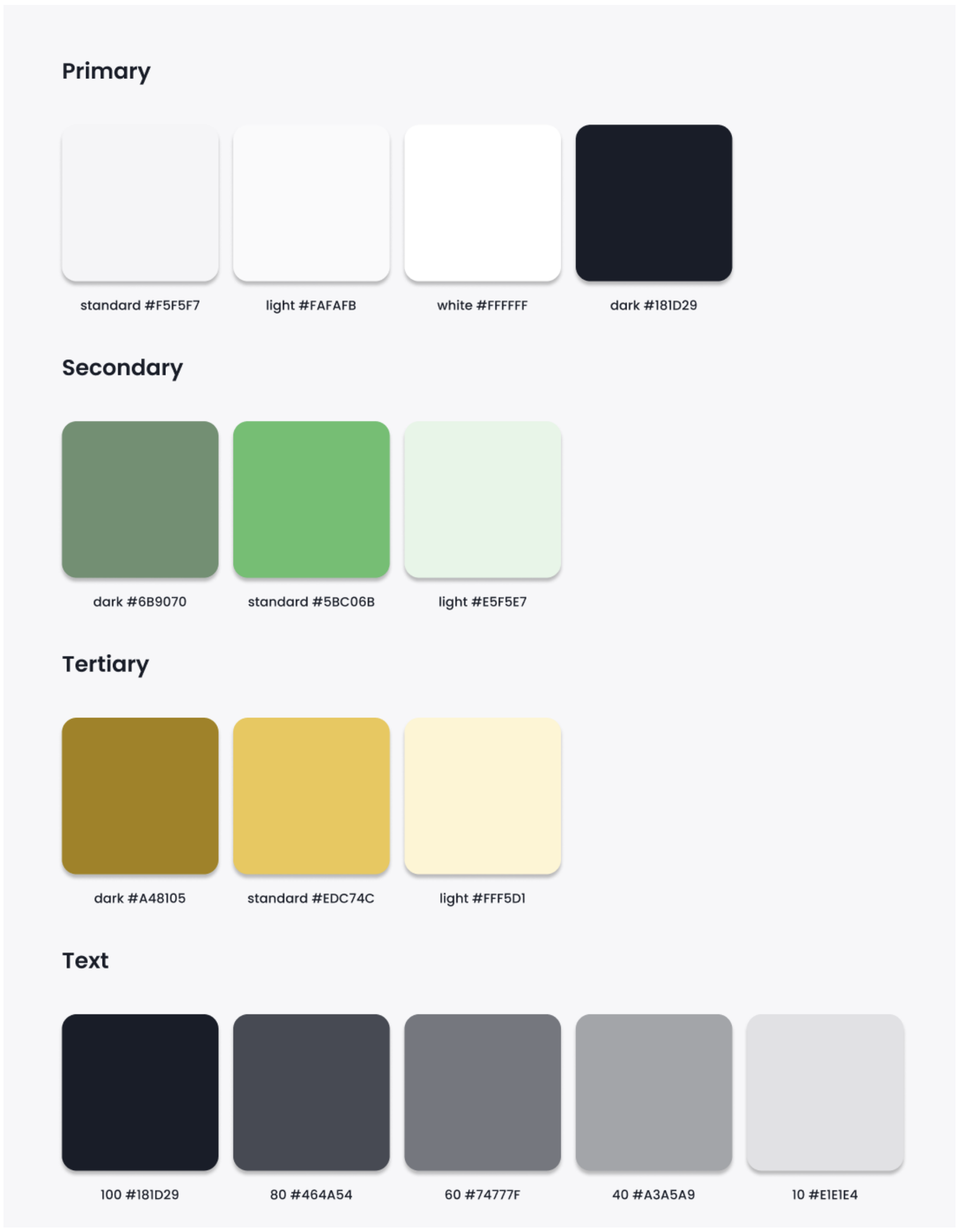
The original idea of the approach is use a server to provide all the data for the form, while the client is responsible for displaying them. Our suggestion is focusing on render the form on client logic of form and interface implementation. Despite the fact that the task does not require writing tests, their presence will be considered a big bonus. Feel free to change an input JSON structure. Try to implement this solution with the same quality and care as you would expect from production-level pull-request, even if some typical requirements (like automatic error reporting) don't make sense to be added.

Deliverable

Bundle everything into a Zip archive and email it to us. Remember that it is easier for us to review your task if we can test / run it. A good check before sending the Zip is to un-archive it into a new folder and check that building and running the project works using the steps you define in README.md. Forgotten dependencies and instructions can sometimes happen even to the best of us.

Design assets

Design assets to help you implement the UI


Primary
standard #F5F5F7
light #FAFAFB
white #FFFFFF
dark #181D29
Secondary
dark #6B9070
standard #5BC06B
light #E5F5E7
Tertiary
dark #A48105
standard #EDC74C
light #FFF5D1
Text
100 #181D29
80 #464A54
60 #74777F
40 #A3A5A9
10 #E1E1E4