Тестовое задание

Общая информация

Для выполнения тестового технического задания обязательно использование стека технологий `React` (версии не ниже 18) и `Typescript` (не ниже 5.6.0). Базовое приложение, на основе которого выполняется задание, может быть создано любым удобным вам способом.

По результатам выполнения тестового задания предоставьте его исходный код (можно в виде ссылки на github или другой репозиторий). Набор файлов должен включать в себя файл `README.md` с подробным описанием запуска проекта и шагов для проверки задания.

1 Построитель формы

Самостоятельно реализуйте компонент, осуществляющий построение произвольной формы. Поля формы и их валидация определяются JSON-схемой (draft 7).

Необходимо ограничиться следующими примитивными типами данных

- `string`
- `integer`
- `boolean`
- `enum`

а также обрабатывать объекты и массивы неограниченного уровня вложенности. Конструкции вида `anyOf`, `oneOf`, и условные блоки (`if`, `then`, ...) **поддерживать не нужно**.

Валидация формы должна быть реализована на базе этой же JSON-схемы. Осуществляется проверка обязательности полей объектов (массив ключей `required`), количества элементов в массивах (`minItems` и `maxItems`), длины и формата строк, минимальные и максимальные значения числовых полей и т. д. Для выполнения функционала валидации допускается использование сторонних библиотек (например, `https://www.react-hook-form.com`).

Приветствуется использование библиотеки `https://mui.com` для визуальных элементов компонента. Пример схемы для выполнения задания вы можете найти в направленном вам архиве.

2 DSL для хранения информации о дашбордах

Реализуйте DSL для хранения информации о стилях и необходимых данных графиков на дашборде. Дашборд это страница где отображается список графиков в виде карточек, он может быть разбит на N секций, с некоторыми отступами друг от друга, например 10 px. Ширина и высота графика в таком случае может быть указана в количестве строк и колонок (см. Рис. 1).

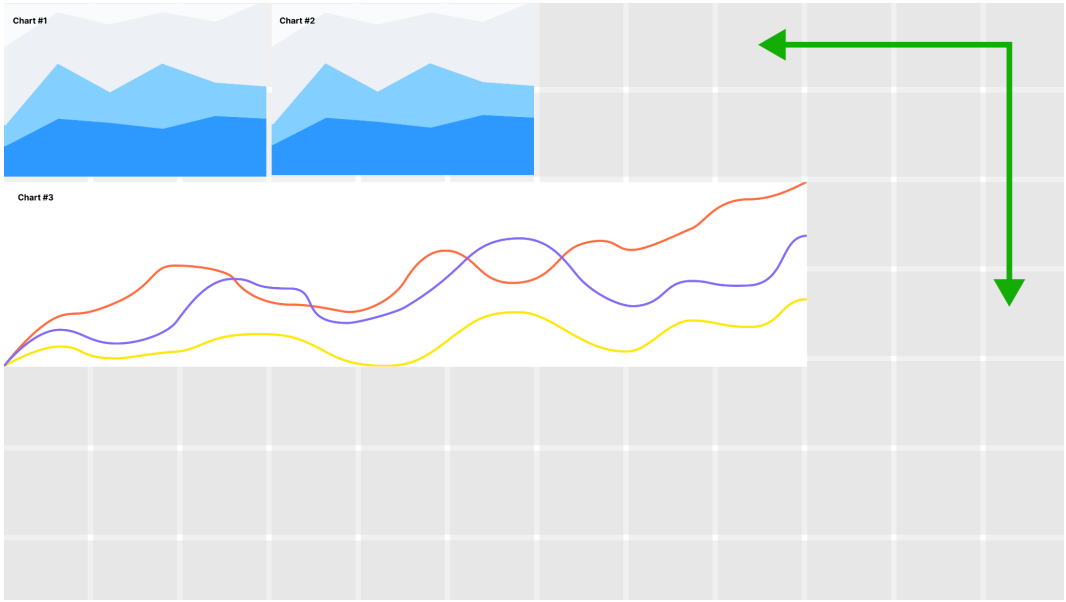

Рис. 1. В разрешении 1920 × 1080 делим страницу на 12 колонок с отступами по 10 px, в результате чего получаем “сетку”

У графика есть следующие поля:

- название
- тип (linear, area, bar)
- цветовая схема — массив цветов, используемых для цветовой идентификации линий
- метрики (значения), которые необходимо отобразить на графике

Данные для построения графика приходят раз в N секунд и содержат поля

```
{
  "time": DateTime,
  "metrics": [
    { "metricName": metricValue },
    ...
  ]
}
```

- `time` — время последнего обновления данных
- `metrics` — массив с метриками и их значениями
- `metricName` — название метрики, строка
- `metricValue` — значение метрики, число

На основе составленного DSL должно быть возможным построить страницу дэшборда с заданными графиками нужных типов и цветовой схемой, нужными метриками. Для записи DSL предпочтительно использовать формат JSON-схемы или `typescript`.

3 Подсчет совпадений в массивах

Представьте, что у вас есть массив A произвольной длины N и массив B произвольной длины M, содержащие в себе лишь случайные натуральные числа ($n = 1, 2, 2, 3, 5, 5, \dots$), совпадения возможны).

При помощи `typescript` составьте наиболее эффективную с точки зрения быстродействия и потребления памяти функцию, которая выполнит подсчет повторений уникальных элементов массива A и массиве B. Обоснуйте выбор использованного решения. Приведите качественную (в терминах big-O) и количественную оценку производительности (для бенчмарка можно воспользоваться, к примеру, https://www.npmjs.com/package/tinybench).