Тестовое задание для фронтенд-разработчика

Цель задания:

Оценить навыки кандидата в разработке на Vue 3 с использованием современных технологий, таких как Composition API, Pinia, Tailwind CSS, а также умение работать с API, реализовывать динамическую подгрузку данных и создавать интуитивно понятный интерфейс.
Функционал простой, загрузить данные с тестового сервиса jsonplaceholder и отобразить в виде таблички.

Технологии:

- Vue 3 (Только Composition API, <script setup>) TypeScript использовать нет необходимости.
- Pinia для управления состоянием.
- Tailwind CSS для стилизации.
- Radix-vue (shadcn-vue) по желанию (для UI компонентов).
- https://jsonplaceholder.typicode.com/guide/ Бесплатный API для получения данных (фотографий), может понадобиться впн.

Шапка приложения:

- Инпут фильтр:
  - Поле для ввода ID альбома (например, "1 2 3").
  - Поддержка множественного ввода через пробел.
  - Если поле пустое, запрашиваются все фотографии.
  - Если указаны ID альбомов, запрашиваются фотографии только для этих альбомов.
  - Пример запроса для нескольких альбомов: https://jsonplaceholder.typicode.com/photos?albumId=1&albumId=2&albumId=3.
- Кнопка "Поиск". При нажатии отправляется запрос на API с учетом введенных ID альбомов.

Блок с таблицей:

- Максимальные размеры:
  - Высота: 600px, ширина: 600px.
  - расположена по центру, на любых экранах,
  - Если контент превышает размеры, появляется внутренний скролл.
- Динамическая подгрузка:
  - Изначально отображается 30 строк.
  - При прокрутке до конца блока подгружаются следующие 20 строк в таблице.
- Отображение данных:
  - Все поля, полученные из API, отображаются в таблице **как есть, в текстовом виде**.
  - Столбцы по порядку:
    - "id" - 'Ид',
    - "albumId" - 'Альбом',
    - "title" - 'Название',
    - "url" - 'Ссылка',
    - "thumbnailUrl" - 'Миниатюра',
  - Если текст не помещается в ячейку, он обрезается (...) и отображается в подсказке (title).
- При клике в ячейку шапки таблицы:
  - сортируется таблица по рядам, количество рядов 30 как при изначальном отображении.

Сценарий работы:

- При запуске приложения автоматически происходит поиск и отображение фотографий в таблице.
- Отображается окно загрузчик (или скелетон, например рядов, ячеек или всей таблицы) в момент отправки запроса. Или заблокированная кнопка “Поиск” с иконкой анимацией загрузки.
- Автоматически отправляется запрос на получение всех фотографий.
- Отображается таблица с возможностью сортировки по столбцам.

Дополнительный функционал (необязательно, но желательно):

- Шапка таблицы прилипает к верхней части блока и видна при прокрутке.
- Переключение тем кнопка для переключения между светлой и темной темами.
- Сохранение Ид альбомов в хранилище localStorage,
- Подсказки для текста, если текст не помещается в ячейку, он обрезается и отображается в подсказке при наведении (tooltip).

Как сдавать задание:

1. Загрузите код на GitHub (или другой Git-хостинг), предоставить открытый доступ.
2. Опубликуйте проект на любом удобном вам хостинге (Vercel, Netlify, Beget и т.д.).
3. Пришлите ссылку на репозиторий и ссылку на демо.

Примечание:

Всё, что не указано в тестовом задании или описано не полностью, остаётся на усмотрение разработчика. Небольшие отклонения от ТЗ не считаются грубыми ошибками. Результат оценивается в целом: учитывается качество выполнения задачи, чистота и понятность кода, а также общий подход к решению. Визуальная составляющая имеет значимость.

Примерный срок выполнения:
1-3 рабочих дня.

Задачи понятны? Если есть вопросы, уточняйте! Удачи в выполнении!