Тестове завдання React

Рівень складності: мінімальний досвід

**Огляд завдання**

Створити React застосунок для управління транзакціями. Застосунок повинен дозволяти користувачам імпортувати список транзакцій з CSV файлу, змінювати статус транзакцій, видаляти транзакції та експортувати відфільтрований список транзакцій у CSV файл.

**Функціональні вимоги**

1. Імпорт транзакцій:

- При натисканні на кнопку "Імпорт" користувач повинен бачити діалогове вікно для завантаження CSV файлу.

- Розібрати CSV файл та зберегти транзакції у стані застосунку.

- Перевіряти формат CSV файлу та показувати відповідні повідомлення про помилки, якщо формат неправильний.

2. Експорт транзакцій:

- При натисканні на кнопку "Експорт" згенерувати та завантажити CSV файл з інформацією про транзакції на основі вибраних фільтрів (тип транзакції, статус).

- Дати можливість користувачу вибрати колонки, які будуть включені у експортований файл.

3. Список транзакцій:

- Відображати дані транзакцій у таблиці з пагінацією.

- Реалізувати можливості сортування та пошуку для списку транзакцій.

- Надати фільтри для типу транзакції та статусу для динамічного оновлення відображених транзакцій.

4. Редагування транзакції:

- При натисканні на кнопку "Редагувати" показувати модальне вікно, що дозволяє користувачу змінити статус транзакції.

- Включити валідацію форми для забезпечення правильного оновлення статусу.

5. Видалення транзакції:

- При натисканні на кнопку "Видалити" показувати діалогове вікно з підтвердженням, яке запитує користувача, чи впевнений він, що хоче видалити цю транзакцію.

- Якщо підтверджено, видалити транзакцію зі стану та оновити інтерфейс відповідно.

**Технічні вимоги**

1. Управління станом та отримання даних:

- Використовувати React Query для управління станом сервера та обробки асинхронних операцій.

- Використовувати вбудовану SQL базу даних (наприклад, SQLite або JavaScript-бібліотеку для SQL) для зберігання початкових даних транзакцій та імітації відповідей API.

2. Обробка форм:

- Використовувати react-hook-form для обробки форм у застосунку.

- Реалізувати валідацію форм та обробку помилок.

3. UI Framework:

- Використовувати ChakraUI для стилізації та макета.

4. Інструменти розробки:

- Використовувати Vite для налаштування проекту та бандлінгу (не використовувати CRA).

- TypeScript є обов'язковим для забезпечення типобезпеки та якості коду.

5. Додаткові покращення (бонусні бали):

- Реалізувати styled-components для покращеної стилізації.

- Додати модульні тести та інтеграційні тести з використанням фреймворку для тестування, такого як Jest та React Testing Library.

**Процесні вимоги**

1. Якість коду:

- Весь код, коментарі та інформація про проект мають бути англійською мовою.

- Дотримуватися кращих практик для React та TypeScript.

2. Контроль версій:

- Завантажити готовий код у GitHub репозиторій.

- Використовувати зрозумілі повідомлення комітів та підтримувати чисту історію комітів.

**Додаткові вимоги**

1. Аутентифікація користувача:

- Додати базовий механізм аутентифікації (наприклад, JWT) для забезпечення безпеки застосунку.

- Створити сторінку входу та захистити маршрути, що вимагають аутентифікації.

2. Розширені фільтри:

- Реалізувати розширені можливості фільтрації, такі як фільтри за діапазоном дат, фільтри за діапазоном сум та пошук за ключовими словами по декількох полях.

3. Візуалізація даних:

- Інтегрувати візуалізацію даних (наприклад, графіки чи діаграми) для надання інсайтів щодо даних транзакцій (наприклад, обсяги транзакцій з часом, розподіл типів транзакцій).

4. Оптимізація продуктивності:

- Оптимізувати застосунок для підвищення продуктивності, включаючи ліниве завантаження компонентів, розподіл коду та мемоізацію де це необхідно.

5. Доступність:

- Забезпечити доступність застосунку та дотримання стандартів WCAG.

- Реалізувати навігацію за допомогою клавіатури та підтримку екранних рідерів.