Онлайн-кошелёк (аналог CoinKeeper) на ReactJS

1. Цель проекта:

Разработать одностраничное веб-приложение онлайн-кошелька, которое позволяет пользователю отслеживать доходы и расходы, видеть баланс, добавлять категории и транзакции, а также просматривать статистику за выбранный период.

2. Функциональные требования:

2.1 Регистрация и аутентификация

- Регистрация нового пользователя (email + пароль)
- Вход и выход пользователя

2.2 Основной экран (Dashboard)

- Отображение текущего баланса
- Список последних транзакций (доходы и расходы)
- Кнопка «Добавить транзакцию»

2.3 Транзакции Добавление новой транзакции:

- Тип: Доход / Расход
- Сумма
- Категория (выбор из существующих или создание новой)
- Дата
- Комментарий (необязательно)
- Редактирование и удаление транзакций

2.4 Категории

- Просмотр, создание и удаление пользовательских категорий
- Возможность добавления иконок и цветов для категорий (опционально)

2.5 Статистика

- Построение графиков (PieChart, BarChart) доходов и расходов по категориям за выбранный период
- Фильтрация по дате

3. Нефункциональные требования:

- Приложение должно быть разработано на ReactJS (с использованием Webpack или Vite)
- Использование хуков: useState, useEffect, useContext
- Управление состоянием через Context API или Redux (предпочтительно Redux)
- Реализация взаимодействия с API для хранения и получения данных
- Верстка с использованием CSS-модулей или TailwindCSS

4. Технологии:

- ReactJS
- React Router DOM

- Axios или fetch для работы с API
- Redux или Context (предпочтительно Redux)
- Chart.js или Recharts (для построения графиков)
- TailwindCSS (по желанию)

5. Страницы:

- /login — Вход
- /register — Регистрация
- /dashboard — Главная панель
- /stats — Статистика
- /settings — Настройки категорий

6. Возможности для расширения:

- Добавление поддержки нескольких счетов
- Адаптация интерфейса под мобильные устройства

7. Критерии готовности MVP (Minimum Viable Product):

- Реализована регистрация и вход в систему
- Добавление и удаление транзакций
- Реализовано взаимодействие с API для хранения и получения данных
- Отображение текущего баланса и списка транзакций
- Построение статистики по категориям

Примечание:

Основное внимание уделяется логике приложения, качественному пользовательскому опыту и чистоте кода.

Референс:

https://coinkeeper.me/introduce-yourself

Online Wallet (CoinKeeper Analog) in ReactJS

1. Project Goal:

Develop a single-page web application for an online wallet that allows users to track income and expenses, view their balance, add categories and transactions, and view statistics for a selected period.

2. Functional Requirements:

2.1 Registration and Authentication

- User registration (email + password)
- User login and logout

2.2 Main Screen (Dashboard)

- Display of current balance
- List of recent transactions (income and expenses)
- "Add Transaction" button

2.3 Transactions

Adding a new transaction:

- Type: Income / Expense
- Amount
- Category (select from existing or create new)
- Date
- Comment (optional)
- Edit and delete transactions

2.4 Categories

- View, create, and delete user-defined categories
- Optionally add icons and colors to categories

2.5 Statistics

- Generate graphs (PieChart, BarChart) of income and expenses by category for a selected period
- Filter by date

3. Non-functional Requirements:

- The application must be developed using ReactJS (with Webpack or Vite)
- Use of hooks: useState, useEffect, useContext
- State management via Context API or Redux (Redux preferred)
- API integration for data storage and retrieval
- Styling with CSS Modules or TailwindCSS

4. Technologies:

- ReactJS
- React Router DOM
- Axios or fetch for API requests
- Redux or Context (Redux preferred)
- Chart.js or Recharts (for chart generation)
- TailwindCSS (optional)

5. Pages:

- /login — Login
- /register — Registration
- /dashboard — Dashboard
- /stats — Statistics
- /settings — Category settings

6. Expansion Opportunities:

- Support for multiple accounts
- Responsive interface for mobile devices

7. MVP (Minimum Viable Product) Criteria:

- User registration and login implemented
- Transaction addition and deletion
- API interaction for data storage and retrieval
- Display of current balance and transaction list
- Statistics generation by category

Note:
Focus is placed on application logic, quality user experience, and clean code.

Reference:
https://coinkeeper.me/introduce-yourself