Expense Tracker - Project Documentation

Project Overview

The expense tracker allows a user to record daily expenses, view them in a list, edit existing entries, and keep the data saved even after refreshing the page.

Tech Stack:

- React
- Tailwind CSS for styling
- Vite (build tool)

Project Structure

```
src/
├── components/
│   ├── Header.jsx                  # App title and logo
│   ├── ExpenseForm.jsx             # Form to add new expenses
│   ├── ExpenseList.jsx             # Container for all expenses
│   ├── ExpenseItem.jsx             # Individual expense card
│   └── ErrorMessage.jsx            # Toast notification for errors
    └── ExpenseSummary.jsx          #Category and amounts
├── assets/
│   └── Images/
│       └── headerImage.png         # Logo/header image
├── Utilities/
        └── FormatCurrency.jsx      # Currency formatting
├── App.jsx                         # Main app component (state management)
└── index.css                       # Global styles
```

---

1. Add Expense

The user can add a new expense by entering:

- Amount (required)
- Category (required)
- Date (required)
- Note (optional)

If required fields are missing, the app shows a temporary error message and prevents submission.

---

2. Expense Storage (Local Storage)

All expenses are saved in the browser's local storage.

- Data remains available after page refresh or browser restart.
- When the app loads, previously saved expenses are automatically restored.
- Any change (add, edit, delete) updates local storage immediately.

---

3. Expense List

- Displays all recorded expenses in a clean list.
- Each expense shows category, date, amount, and optional note.
- If no expenses exist, an empty-state message is shown.

---

4. Edit Expense

- Each expense has an edit option.
- Clicking edit loads the selected expense back into the form.
- The form switches to “Edit mode” to indicate an update is happening.
- After updating, the existing expense is modified instead of creating a new one.
- Edit mode can be canceled to return to normal add mode.

---

5. Delete Expense

- Each expense can be removed individually.
- Deleted expenses are immediately removed from the list and local storage.

---

6. Clear All Expenses

- A single action removes all stored expenses.
- Used when the user wants to reset tracking completely.
-

Data Flow Architecture

Adding an Expense

User fills the expense form

↓

User clicks Add Expense

↓

Expense form validates required fields

↓

Expense form sends expense data to App component

↓

App component adds a unique ID to the expense

↓

Expense is added to the expenses list in state

↓

Updated expenses list is saved to local storage

↓

Expense list re-renders with the new expense displayed

Editing an Expense

User clicks Edit on an existing expense

↓

Selected expense data is sent to the App component

↓

App sets the selected expense as the editing expense

↓

Expense form is populated with existing expense data

↓

User modifies the expense fields

↓

User clicks Update Expense

↓

Updated data is sent back to the App component

↓

Matching expense is updated in the expenses list

↓

Updated expenses list is saved to local storage

↓

Expense list re-renders with the updated expense

Deleting an Expense

User clicks Delete on an expense

↓

Expense ID is sent to the App component

↓

Matching expense is removed from the expenses list

↓

Updated expenses list is saved to local storage

↓

Expense list re-renders without the deleted expense

Local Storage Synchronization

Application loads

↓

Expenses are read from local storage

↓

Expenses are loaded into application state

↓

UI renders using stored expense data
