1. Core Features and Functionality

Expense Management

- **Add Expense:**
  Users can fill out a form to input the amount, select a category (e.g., food, transport), enter a description, and pick a date.
- **Edit/Delete Expense:**
  Each expense entry in the list/table includes options to edit or delete, allowing users to correct mistakes or remove outdated entries.

Expense Display

- **Recent Expenses Table:**
  Shows a list of the latest expenses with columns for each detail (amount, category, description, date). The table updates automatically when users add, edit, or delete entries.

---

2. Filtering and Search

Category Filter

- Users can select a specific category (e.g., groceries) to only view related expenses.

Date Range Filter

- Users can choose a start and end date to see expenses within that period. This helps track spending over weeks or months.

---

3. Data Handling and Storage

LocalStorage or JSON-based

- **LocalStorage:**
  All expense data is stored in the browser's LocalStorage. This means data persists after refreshes but is only available on the same device/browser.
- **Data Structure:**
  Each expense is stored as an object in an array. The array is serialized to JSON and stored/retrieved as needed.
- **CRUD Operations:**
  Adding, editing, and deleting expenses all update the LocalStorage data, ensuring consistency between what's displayed and what's stored.

---

4. Visual Analytics (Bonus)

Monthly Expense Chart

- A visual chart (bar, pie, or line) displays how much was spent in each category for the current month.
- Helps users quickly spot where most of their money goes (e.g., "Food: $120, Transport: $50").

---

5. UI/UX Design

Responsive Layout

- The app adjusts layout for different screen sizes (desktop, tablet, mobile) using responsive design principles.
- Forms, tables, and charts all remain readable and usable on any device.

Intuitive Design

- Clear button placement (e.g., “Add Expense”, “Edit”, “Delete”).
- Form validation prevents incomplete or incorrect entries.
- Filters and charts are easy to access and use.

---

6. Code Quality and Structure

Modularity

- The code is organized into functions or components (if using a framework) for:
    - Form handling
    - Table rendering
    - Filtering logic
    - Chart generation
    - Data storage operations

Clean Practices

- Consistent naming conventions
- Separation of concerns (UI, logic, storage)
- Comments and documentation for maintainability

---

7. Evaluation Alignment

- **Functionality:**
  Complete CRUD (Create, Read, Update, Delete) for expenses.
- **UI/UX:**
  User-friendly, responsive, and visually appealing.
- **Data Handling:**
  Reliable, persistent storage and retrieval from LocalStorage.
- **Code Quality:**
  Well-structured, maintainable, and modular code.
- **Bonus:**
  Effective chart/analytics integration for spending insights.