Step 1: Setting Up the Basic Structure

- **Goal:** Create the HTML structure for the expense tracker.
- **Tasks:**
  - Add a form with input fields for category and amount, along with a submit button.
  - Create a container (`div`) to display categories and their respective expenses.
- **Practice:**
  - Create basic HTML with IDs: `addExpenseForm`, `category`, `amount`, and `categoriesContainer`.

Step 2: Adding an Expense

- **Goal:** Implement functionality to add an expense under a category.
- **Tasks:**
  - Write the `addExpense` method to create an expense object with an ID, amount, and date.
  - Group expenses by category in the `expenses` object.
  - Validate that the amount is greater than zero.
  - Render categories and expenses after adding a new expense.
- **Practice:**
  - Test adding expenses to new and existing categories.

Step 3: Rendering Categories and Expenses

- **Goal:** Display categories and their respective expenses dynamically in the DOM.
- **Tasks:**
  - Write the `renderCategories` method to loop through categories and their expenses.
  - Display the total amount for each category.
  - List each expense with details (amount and date) and a "Remove" button.
- **Practice:**
  - Style the categories and expenses list for better appearance using CSS.

Step 4: Calculating Total Expenses for a Category

- **Goal:** Compute the total expenses for each category.
- **Tasks:**

    - Write the `getTotalExpenses` method to sum up the amounts for all expenses in a category.
    - Use this method to display the total amount in the category header.
- **Practice:**
    - Test with multiple categories and expenses to verify totals.

Step 5: Removing an Expense

- **Goal:** Allow users to remove an individual expense.
- **Tasks:**
    - Write the `removeExpense` method to filter out the expense by its ID.
    - Remove the category if all its expenses are deleted.
    - Re-render categories after removing an expense.
- **Practice:**
    - Test removing expenses from different categories.

Step 6: Resetting the Form

- **Goal:** Clear the form after adding an expense.
- **Tasks:**
    - Use `e.target.reset()` to clear the input fields after form submission.
- **Practice:**
    - Test adding expenses to ensure the form resets each time.

Step 7: Polishing the User Interface

- **Goal:** Enhance the visual design and usability of the application.
- **Tasks:**
    - Style the categories, expense list, and buttons for a professional look.
    - Use CSS grid or flexbox for layout and spacing.
    - Add hover effects for buttons and improve readability.
- **Practice:**
    - Test the application on different screen sizes to ensure responsiveness.

Bonus Steps

1. **Add Local Storage:**

    - Save the `expenses` object to `localStorage` to persist data between sessions.
    - Load data from `localStorage` when the application starts.
2. **Add a Search Feature:**
    - Allow users to search for expenses by category name or date.
    - Filter the displayed categories and expenses based on the search query.
3. **Add Expense Sorting:**
    - Add options to sort expenses within a category by date or amount.
4. **Generate Reports:**
    - Display a summary of total expenses across all categories.
    - Generate a downloadable CSV or JSON report of all expenses.
5. **Add Dark Mode:**
    - Add a toggle to switch between light and dark themes.

Practicing Each Step Implement and test each step independently. Once all steps are complete, integrate them to create a fully functional **Expense Tracker Application**.