Step-by-Step Guide for Building an Expense Tracker Application

1. Define Your Goal

Understand the core functionalities of the application:

- Add expenses under specific categories.
- Remove expenses by ID.
- Calculate and display the total expenses for each category.
- Dynamically render categories and their associated expenses.

---

2. Plan the Structure

Break the application into smaller components:

1. **Data Storage**: Use an object to store categories and their expenses.
2. **Methods**: Define functions for adding expenses, removing expenses, calculating totals, and rendering categories.
3. **Event Handling**: Handle interactions like form submissions and button clicks.

---

3. Start Writing Code

Begin with the basics and build incrementally:

A. Set Up an Expense Tracker Object

Create a JavaScript object (`expenseTracker`) to manage expenses and their associated actions:

- Properties:
  - `expenses`: Object to store categories as keys and an array of expenses as values.
- Methods:
  - `addExpense(category, amount)`
  - `getTotalExpenses(category)`
  - `removeExpense(category, expenseId)`
  - `renderCategories()`

B. Implement Core Methods

Write the methods in a modular way:

- `addExpense`: Add a new expense under a specified category. Create the category if it doesn't exist.
- `getTotalExpenses`: Calculate the total expenses for a specific category.
- `removeExpense`: Remove an expense by its unique ID from a specific category. Delete the category if it becomes empty.
- `renderCategories`: Dynamically update the DOM to display categories and their associated expenses.

C. Attach Event Listeners

Handle the interactions:

- **Form Submission**: Use `addEventListener` on the form to trigger `addExpense` for adding new expenses.
- **Remove Buttons**: Dynamically generate buttons and attach handlers for removing expenses.

D. Test and Debug

- Test each method individually in the browser console.
- Validate inputs for edge cases (e.g., empty fields, negative or zero amounts).

---

4. Order of Implementation

Follow this sequence:

Initialize the Expense Tracker Object:

```
const expenseTracker = { expenses: {} };
```

1.
2. **Add Core Methods to the Object**:
    - `addExpense(category, amount)`
    - `getTotalExpenses(category)`
    - `removeExpense(category, expenseId)`
    - `renderCategories()`
3. **Create Event Handlers**:
    - Write the logic for form submission and attach it to the "Submit" button.
    - Write the logic for remove actions and attach them to dynamically generated buttons.

4. **DOM Manipulation**:
    - Use `document.createElement` and `appendChild` to render categories and expenses.
    - Dynamically update the DOM for category totals and individual expenses.
5. **Test the Flow**:
    - Add expenses under different categories.
    - Remove expenses and check for proper updates.
    - Calculate and display category totals.
    - Ensure the DOM updates correctly.

---

5. Add Features Incrementally

Once the basics work, enhance the application:

- **Date Filtering**: Allow filtering expenses by date range.
- **Category Sorting**: Enable sorting of categories alphabetically.
- **Validation**: Ensure fields are filled correctly and amounts are valid before submission.
- **Styling**: Apply CSS classes for better UI and user experience.

---

6. Checklist for Completion

- Expenses are added correctly under the specified categories.
- Expenses can be removed by their unique ID.
- Total expenses for each category are calculated and displayed accurately.
- Categories and expenses render dynamically and update after actions.

By following this roadmap, you will systematically build the Expense Tracker Application with clarity and focus.