Undo Functionality Design Document

Objective

The goal of this undo functionality is to allow users to reverse the most recent transaction removal. This improves the app's usability and aligns with standard UI principles that help users recover from mistakes. Specifically, if a user removes a transaction by mistake, they should be able to undo that action and have the transaction (and total cost) restored.

Design Overview

We will follow the Model-View-Controller (MVC) architecture to structure the undo functionality in a maintainable and modular way. Here's how each component will support undo:

Model (ExpenseTrackerModel)

- **Data Structure**: Add a `Stack<Transaction>` named `removedTransactions` to keep track of recently deleted transactions.
- **Method:** Add `undoLastRemoval()` which:
  - Checks if the `removedTransactions` stack is not empty.
  - Pops the top transaction.
  - Re-adds the transaction to the main `transactions` list.
  - Updates the total cost.
- **Changes Required:**
  - Modify `removeTransaction(Transaction t)` to push the removed transaction onto the stack.
  - Ensure immutability principles are still followed.

Controller (ExpenseTrackerController)

- **Method:** Add `handleUndo()` which:
  - Calls `model.undoLastRemoval()`.
  - Notifies the view to refresh and reflect the updated transaction list and total.
- **Integration:** This method is triggered by an event from the view (e.g., button click).

View (ExpenseTrackerView)

- **UI Element:** Add an "Undo" button to the interface.
- **Event Handling:**
  - When the button is clicked, it triggers the `handleUndo()` method in the controller.
  - After undo is performed, the transaction table and total cost label are refreshed.

User Flow Example

1. User removes a transaction.
2. The app stores that transaction in a stack.
3. User clicks "Undo".
4. The app restores that transaction to the list and updates the total cost.
5. The view refreshes to show the restored transaction.

Summary of Class Changes

Component	New or Modified Members
ExpenseTrackerModel	Stack<Transaction> removedTransactionsundoLastRemoval() methodModified removeTransaction()
ExpenseTrackerController	handleUndo() method
ExpenseTrackerView	"Undo" button + event listener

Edge Cases Considered

1. Multiple undo calls: Only the last removed transaction is restored per call.
2. Empty undo stack: Calling undo does nothing or shows a friendly message (optional).

Conclusion

This design supports extensibility, keeps logic in the appropriate layers of MVC, and allows users to safely and intuitively reverse removal actions. It can be implemented incrementally without disrupting existing functionality.