

# Budget Tracker

## Section 1 - Overview of Program

- This program is about Budget Tracker to help users or basically myself to monitor my spending across different categories. It will help solve problem like losing track of daily expenses by organizing it in a clear and structured way as most of the people are in tight budget and something it becomes hard to manage our money without the proper tracker. This program will be useful as it will show how much has been spent and how much is left in each category and it also provides warnings when I or someone using the program goes over the budget. It is also educational as it teaches the user how budgeting works and encourages better financial habits. The main features will include adding expenses, viewing spending by category, calculating total spending, and checking for overspending. In nut shell, it will be like an interactive way to understand personal finances.

## Section 2 – Technical Requirements

### Dictionary Usage

- I will uses two dictionaries.(can be more in future)
- First will be budget, which stores category names like transportation, food and others and it will put some limits on spending as values.
- Second will be expenses, which uses the same category as in budget but stores the amount of money spend in each one.
- The program will uses these dictionaries to update spending, compare totals to limits as will as display the user's financial status.

### Function Usage

- It will use a function called calculate\_total(expenses\_dict).
- It will take expenses dictionary as a parameter and adds up all the spending amounts.
- It will return a total spending of overall.

### Loop Usage

- I will use a while loop to show the main menu until the user chooses the exit.
- There will also be different loops while going through each category when showing spending and checking for overspending.

### Input

- Menu choices (1-5)

- Category names when adding expenses
- Amount of money spent

### Outputs

- Menu Options
- Expense confirmations
- Spending totals
- Category-by-category spending
- Overspending warnings
- Exit messages

## Section 3: Pseudocode

### Start

- Display “Welcome to the Budget Tracker!”

### Creating Budget Dictionary

- Create dictionary budget with category names as keys and spending limits as values:
  - “food” → 200
  - “transportation” → 100
  - “entertainment” → 150

### Creating Expenses Dictionary

- Create dictionary expenses with the same category names but all values set to 0:
  - “food” → 0
  - “transportation” → 0
  - “entertainment” → 0

### Defining calculate\_total(expenses\_dict)

- Set **total = 0**
- For each category in expenses\_dict:
  - Add its value to total
- Return total

### Main Menu Loop

- While True:
  - Display:
    - “1. Add an expense”
    - “2. View spending by category”

- “3. View total spending”
- “4. Check overspending”
- “5. Exit”
- Ask user for **choice**

#### If choice is 1: Add Expense

- Ask user: “Enter category name”
- If category not in budget:
  - Display “Category not found”
  - Continue loop
- Ask user: “Enter amount spent”
- Convert amount to number
- Add amount to expenses[category]
- Display “Expense added successfully”

#### If choice is 2: View Spending by Category

- For each category in expenses:
  - Display category and amount spent

#### If choice is 3: View Total Spending

- Call **calculate\_total(expenses)**
- Store returned value in **total\_spent**
- Display “Total spent: \$total\_spent”

#### If choice is 4: Check Overspending

- For each category in budget:
  - If expenses[category] > budget[category]:
    - Display “Warning: You overspent in [category]!”
  - Else:
    - Display “[category]: Within budget.”

#### If choice is 5: Exit

- Display “Goodbye!”
- Break loop

#### Invalid Input

- If choice is not 1–5:
  - Display “Invalid choice, try again.”

## Section 4: Test Cases

### Test Case 1: Adding Expenses Normally

Shows a normal user flow where expenses are added correctly.

#### Inputs:

- Choice: 1
- Category: food
- Amount: 50

#### Expected Output:

- “Expense added successfully.”
- expenses[“food”] becomes 50  
(it confirms that expense-adding works correctly.)

### Test Case 2: Overspending Warning

**Description:** Demonstrates overspending detection.

#### Inputs:

- Add expense: category “entertainment”, amount 200
- Then choose option 4 (check overspending)

#### Expected Output:

- “Warning: You overspent in entertainment!”  
(This shows that conditional logic and dictionary comparison are working.)

### Test Case 3: Error Handling for Invalid Category

**Description:** Tests whether the program handles unknown category names.

#### Inputs:

- Choice: 1
- Category: “clothes”

### **Expected Output:**

- “Category not found.”  
(This ensures program handles incorrect user input safely).

## **Section 5: Reflection**

I choose the Budget Tracker because as a student I find it hard to keep track of my real spending and I quickly become short on the money. It will be great help to people like me who finds it hard to keep track of their spending. The structure of the program made sense to me as separating information into dictionaries keeps the budget limits and expenses organized and it is also easy to update. The main challenge while I was planning was deciding which features are necessary and how to structure the menu so it stayed useful as well as simple. I tried using dictionaries as they allow fast lookups and make it a lot easier to match category data with its spending limit. If I had used list, it would have been a lot confusing because I would need to separate lists for names, limits, and totals, which increases the chances of getting more bug. To sum up, this design will help make program clear, readable and easy to expand later if new ideas come across.