**Technical Design Document Template**

**Name:** Aaden Gabriel Abarintos

**Date Created:** 9/28/2025

**Program Description:**

This program asks the user to list their expenses and the amount of each expense. This program then stores the data in a dictionary and calculates the total expenses, the highest expense, and the lowest expense.

**Functions used in the Program (list in order as they are called):**

1. **Function Name:** main()

**Description:** This function creates an empty dictionary to store the user’s expenses and then asks the user for their expenses using a while loop. The function then uses lambda functions to calculate the total expense, the highest expense, and the lowest expense.

**Parameters:** N/A

**Variables:**

1. expenses(dictionary): used to hold the input(s) from the user.

2. expense\_type: the type of the expense the user inputs. Stored in the dictionary.

3. expense\_amount: the amount of the expense the user inputs. Stored in the dictionary.

4. total\_expenses(lambda function): A function used to calculate the total expenses.

5. highest\_expense(lambda function): A function used to calculate the highest expense.

6. lowest\_expense(lambda function): A function used to calculate the lowest expense.

2. **Function Name:** total\_expenses (lambda)

**Description:** This function is a lambda function that calculates the sum of all the expenses entered by the user.

**Parameters:**

1. acc: the accumulator of the function (the running total)
2. x: the part of the dictionary that represents a single expense.

**Variables:**

1. acc: Keeps track of the sum of the expenses.
2. x[“amount”]: The amount of a specific expense.

**Logical Steps:**

1. Start with the accumulator (acc) which is 0.
2. For each expense add the amount of that expense to 0 until all expenses are processed.
3. Return the total.

**Returns:** Returns the total expenses.

3. **Function Name:** highest\_expense (lambda)

**Description:** This function is a lambda function that calculates the highest expense of the user.

**Parameters:**

1. acc: The current highest expense.
2. x: The next expense in the dictionary.

**Variables:**

1. acc[“amount”]: The amount of the highest expense.
2. x[“amount”]: The amount of the next expense.

**Logical Steps:**

1. Start with the accumulator or first expense.
2. Compare the accumulator with the next expense.
3. If the next expense is higher, return it.
4. Else return the first expense.
5. Continue until all expenses have been compared.

**Returns:** Returns the highest expense.

4. **Function Name:** lowest\_expense (lambda)

**Description:** This function is a lambda function that calculates the lowest expense of the user.

**Parameters:**

1. acc: The current lowest expense.
2. x: The next expense in the dictionary.

**Variables:**

1. acc[“amount”]: The amount of the lowest expense.
2. x[“amount”]: The amount of the next expense.

**Logical Steps:**

1. Start with the accumulator or first expense.
2. Compare the accumulator with the next expense.
3. If the next expense is lower, return it.
4. Else return the first expense.
5. Continue until all expenses have been compared.

**Returns:** Returns the lower expense.

**Logical Steps:**

1. Call the main function.
2. Calls and defines the total\_expenses lambda function within the main.
3. Calls and defines the highest\_expense lambda function within main.
4. Calls and defines the lowest\_expense lambda function within main.

**Link to your repository:** <https://github.com/7aaden/COP2373-Semester-1>

**Output Screenshot:**

