**Technical Design Document For Chapter 3 Exercise**

**Name:** Jackson Hodge

**Date Created:** 2/4/25

**Program Description:**

This program asks the user to input different types of expenses they make and how much they cost. The program then calculates and displays the total amount of money spent and displays the highest and lowest expenses.

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

1. **Function Name:** user\_expenses

**Description:** This function gets input from the user and creates a list of their expenses with the type of expense and the amount.

**Parameters:** This function does not take in any parameters.

**Variables:**

1. expenses is equal to an empty list.
2. go is equal to input from the user. (This is used to run the while loop that allows the user to enter multiple expenses)
3. type is equal to input from the user for the type of expense.
4. amount is equal to input from the user for the amount of the expense.

**Logical Steps:**

1. Initialize the list of expenses.
2. Ask the user if they’d like to use the program to run the while loop.
3. Ask the user for the type of expense.
4. Ask the user for the amount of the expense.
5. Return expenses.

**Returns:** The function returns expenses for it to be used in analyze\_expenses.

2. **Function Name:** analyze\_expenses(expenses) + reducer(accumulator, expenses)

**Description:** analyze\_expenses analyzes the expenses from the user. The function initializes an accumulator for the reduce function. reducer sets a total amount money, the highest expense, and the lowest expense.

**Parameters:** analyze\_expenses takes in expenses and reducer takes in accumulator and expense.

**Variables:**

1. initial\_value is equal to a dictionary that is used as an accumulator to add the expenses to.
2. analysis is equal to a reduce function. The reduce function takes in reducer, expenses, and initial\_value.
3. Inside reducer, three variables are created.
   1. total is equal to the total key in the accumulator and adds the current expense the function is working on.
   2. highest is equal to the highest key in the accumulator.
   3. lowest is equal to the lowest key in the accumulator.

**Logical Steps:**

1. Inside analyze\_expenses, define the reducer function.
2. Set the total amount of money spent equal to total.
3. Set the highest expense equal to highest.
4. Set the lowest expense equal to lowest.
5. An if statement is used to find the highest expense.
6. An if statement is used to find the lowest expense.
7. Return the dictionary containing the total, highest, and lowest expense.
8. Inside analyze\_expenses after the reducer function, create the accumulator for the reducer function.
9. Set analysis to reduce reducer, expenses, and the accumulator.
10. Return analysis.

**Returns:** The dictionary containing the total, highest, and lowest expense and analysis.

1. **Function name:** print\_analysis

**Description:** This function is responsible for displaying the output.

**Parameters:** This function takes in analysis.

**Variables:** There are no variables.

**Logical Steps:**

1. Print the header.
2. Print the total expenses.
3. Print the highest expense.
4. Print the lowest expense.

**4. Function name:** main

**Description:** This function calls the other functions to run the program.

**Parameters:** This function does not take in any parameters.

**Variables:**

1. expenses calls user\_expenses.
2. analysis calls analyze\_expenses.

**Logical Steps:**

1. user\_expenses is called.
2. analyze\_expenses is called.
3. print\_analysis is called.

**Logical Steps of the Program:**

1. main is called.
2. user\_expenses is called inside main.
3. analyse\_expenses is called inside main.
4. print\_analysis is called inside main.

**Link to your repository:** <https://github.com/Jackson112607/COP2373-ProgrammingConceptsII>