Pseudo Code Design

Personal Finance Tracker v2.0 | 4COSC006C[PRO] | Coursework Part B

Name: Ambagahage Shan Emalka Fernando
IIT Student No: 20233126
UoW Student No: w2082285

CONTENT

File handling functions

1. load_transactions()

```
BEGIN FUNCTION load_transactions()
     TRY
         OPENFILE file_name_1 IN read AS file
             LOADFILE file TO transactions
             CLOSEFILE file
     EXCEPT FileNotFoundError
         DISPLAY ("File not found! New file has been created.")
         CALL save_transactions()
     EXCEPT JSONDecodeError
         CALL save_transactions()
     FINALLY
         CALL read_bulk_transactions_from_file()
 END FUNCTION
```

2. save_transactions()

```
BEGIN FUNCTION save_transactions()
    OPENFILE file_name_1 IN write AS file
        SAVEFILE file FROM transactions
        CLOSEFILE file
END FUNCTION
```

Feature Implementations

3. read_bulk_transactions_from_file()

```
BEGIN FUNCTION read_bulk_transactions_from_file(read_message = FALSE)

        BEGIN FUNCTION message()

            DISPLAY ("1. Open 'finance.txt' and add transactions to it.")

            DISPLAY ("2. Save 'finance.txt'.")

        END FUNCTION


        BEGIN FUNCTION main()

            TRY

                OPENFILE file_name_2 IN read AS file

                    lines = file.READLINES()

                    DECLARE i AS 0

                    FOR line IN lines

                        i = i + 1

                        IF line is empty THEN

                            CONTINUE

                        ELSE

                            TRY

                                SPLIT FROM "," and ADD to elements

                                IF length of elements is not 4 THEN

                                    DISPLAY ("Invalid format in line "i". Transaction not
added!. Each line should contain [Category, Amount, Type(income/expense), date]")

                                    CONTINUE

                                END IF

                                IF elements[1] is not a number THEN

                                    DISPLAY ("Amount(elements[1]) is not valid.
Transaction not added!")

                                    CONTINUE

                                END IF

                                IF elements[2] is not "Income" or "Expense" THEN

                                    DISPLAY ("Type(elements[2]) is not valid. Transaction
not added!")

                                    CONTINUE

                                END IF
                                IF elements[3] is not a date THEN
                                    DISPLAY ("Date(elements[3]) is not valid. Transaction
not added!")
                                    CONTINUE
                                END IF
                            EXCEPT
                                CONTINUE
                            END TRY
                        END IF

                        temporary = {}
                        category, amount, type, date = SPLIT line FROM ","
                        IF type is "Income" THEN
                            temporary["amount"] = amount
                        ELSE
                            temporary["amount"] = -amount
                        END IF
                        append_transaction(temporary, category)
                    END FOR
                    CLOSEFILE file
            EXCEPT FileNotFoundError
                DISPLAY ("File not found! New file has been created.")
                OPENFILE file_name_2 IN append AS file
                    CLOSEFILE file
            FINALLY
                CALL save_transactions()
                CALL clear_text_file()
            END TRY
        END FUNCTION
        IF read_message is TRUE THEN
            message()
        END IF
        main()
    END FUNCTION
```

4. add_transaction()

```
BEGIN FUNCTION add_transaction()
```

```
    DECLARE transactions dictionary as global
    temporary = {}
    WHILE True DO
        TRY
            ASSIGN prompt() TO temporary_list
            ASSIGN temporary_list[1] TO temporary["amount"]
            ASSIGN temporary_list[3] TO temporary["date"]
            CALL append_transaction(temporary, temporary_list[0])
            BREAK
        EXCEPT ValueError
            DISPLAY ("Invalid input!")
            CONTINUE
        FINALLY
            CALL save_transactions()
        END TRY
    END WHILE
END FUNCTION
```

5. view_transactions()

```
BEGIN FUNCTION view_transactions()
    IF (transactions available in dictionary) THEN
        CALL view_transactions
        FOR key, value IN transactions dictionary DO
            DECLARE i AS 0
            FOR entry IN value DO
                i = i + 1
                DISPLAY key
                DISPLAY i
                DISPLAY entry["amount"]
                DISPLAY entry["date"]
            END FOR
        END FOR
    ELSE
        CALL add_transaction()
    END IF
END FUNCTION
```

6. update_transaction()

```
    BEGIN FUNCTION update_transaction()
        DECLARE transactions dictionary as global
        IF (transactions available in dictionary) THEN
            CALL view_transactions
            WHILE TRUE DO
                TRY
                    GET category
                    IF category is not in dictionary THEN
                        DISPLAY ("Category not found!")
                        CONTINUE
                    END IF
                    GET index
                    IF index-1 is not in dictionary[category]
THEN
                        DISPLAY ("Index not found!")
                        CONTINUE
                    END IF
                    ASSIGN prompt(FALSE,
transactions[category][index-1]["amount"]) TO temporary_list
                    ASSIGN temporary_list[0] TO
dictionary[category][index-1]["amount"]
                    ASSIGN temporary_list[2] TO
dictionary[category][index-1]["date"]
```

```
                BREAK
            EXCEPT
                DISPLAY ("Invalid input!")
                CONTINUE
            FINALLY
                CALL save_transactions()
            END TRY
        END WHILE
    ELSE
        CALL add_transaction()
    END IF
END FUNCTION
```

7. delete_transation()

```
BEGIN FUNCTION delete_transaction()
    DECLARE transactions dictionary as global
    IF (transactions available in dictionary) THEN
        CALL view_transactions
        GET category
        IF category is not in dictionary THEN
            DISPLAY ("Category not found!")
            CALL add_transaction()
        ELSE
            GET index
            IF index-1 is not in dictionary[category] THEN
                DISPLAY ("Index not found!")
                CALL add_transaction()
            ELSE
                DELETE dictionary[category][index-1]
                IF dictionary[category] is empty THEN
```

```
                    DELETE dictionary[category]
                END IF
                CALL save_transactions()
            END IF
        END IF
    ELSE
        CALL add_transaction()
    END IF
END FUNCTION
```

8. display_summery()

```
BEGIN FUNCTION display_summary()
    DECLARE total_income as 0
    DECLARE total_expense as 0
    FOR value IN transactions dictionary DO
        FOR item IN value DO
            IF item["amount"] > 0 THEN
                total_income = total_income + item["amount"]
            ELSE
                total_expense = total_expense + item["amount"]
            END IF
        END FOR
    END FOR
    DECLARE balance as total_income + total_expense
    DISPLAY ("Total Income: " + total_income)
    DISPLAY ("Total Expense: " + total_expense)
    DISPLAY ("Financial Balance: " + balance)
END FUNCTION
```

Helper Functions

9. prompt()

```
BEGIN FUNCTION prompt(flag = TRUE, amount_income_or_expense = 0)
    DECLARE temporary as {}
    IF flag is TRUE THEN
        WHILE TRUE DO
            GET category
            IF category is empty THEN
                DISPLAY ("Invalid input!")
            ELSE
                temporary["category"] = category
                BREAK
            END IF
        END WHILE
    END IF

    WHILE TRUE DO
        TRY
            GET amount
            IF amount <= 0 THEN
                DISPLAY ("Enter amount higher than zero.")
                CONTINUE
            ELSE
                temporary["amount"] = amount
                BREAK
            END IF
        EXCEPT
            DISPLAY ("Invalid input!")
            CONTINUE
        END TRY
    END WHILE

    IF flag is TRUE THEN
        WHILE TRUE DO
            GET transaction_type
            IF transaction_type is "I" or "Income" THEN
                temporary["type"] = "Income"
                BREAK
            ELSE IF transaction_type is "E" or "Expense" THEN
                temporary["type"] = "Expense"
                BREAK
            ELSE
                DISPLAY ("Invalid input!")
            END IF
        END WHILE
    ELSE
        IF amount_income_or_expense < 0 THEN
            temporary["type"] = "Expense"
        ELSE IF amount_income_or_expense > 0 THEN
            temporary["type"] = "Income"
        END IF
    END IF

    WHILE TRUE DO
        TRY
            GET year, month, day
            IF month in {1, 3, 5, 7, 8, 10, 12} THEN
                IF day < 1 or day > 31 THEN
                    DISPLAY ("Invalid day! Please re-enter.")
```

```
                        CONTINUE
                    END IF
                ELSE IF month in {4, 6, 9, 11} THEN
                    IF day < 1 or day > 30 THEN
                        DISPLAY ("Invalid day! Please re-enter.")
                        CONTINUE
                    END IF
                ELSE IF month is 2 THEN
                    IF (year is not divisible by 4 and year is
divisible by 100) or (year is not divisible by 400) THEN
                        IF day < 1 or day > 28 THEN
                            DISPLAY ("Invalid day! Please re-
enter.")
                            CONTINUE
                        END IF
                    ELSE
                        IF day < 1 or day > 29 THEN
                            DISPLAY ("Invalid day! Please re-
enter.")
                            CONTINUE
                        END IF
                    END IF
                END IF
                EXIT LOOP
            EXCEPT
                DISPLAY ("Invalid input!")
                CONTINUE
            END TRY
        END WHILE

        temporary["date"] = year + "-" + month + "-" + day
        RETURN temporary
```

```
END FUNCTION
```

10. clear_text_file()

```
BEGIN FUNCTION clear_text_file()
        OPENFILE file_name_2 IN write AS file
            CLEAR file
        CLOSEFILE file
    END FUNCTION
```

11. transaction_availability()

```
BEGIN FUNCTION transaction_availability(message)
    WHILE TRUE DO
        GET user_input
        IF user_input is "Y" THEN
            CALL add_transaction()
        ELSE IF user_input is "N" THEN
            RETURN
        ELSE
            DISPLAY ("Please enter Y or N.")
        END IF
    END WHILE
END FUNCTION
```

12. category_id_verification()

```
BEGIN FUNCTION category_id_verification(message)
        DECLARE temp as {}
        WHILE TRUE DO
            GET user_input_category
            IF user_input_category is in transactions dictionary
THEN
```

```
                temp["category"] = user_input_category
                BREAK
            ELSE
                DISPLAY ("Category does not exist. Enter valid
category!")
                CONTINUE
            END IF
        END WHILE


        WHILE TRUE DO
            TRY
                GET user_input_id
                IF user_input_id < 1 or user_input_id > length of
transactions[temp["category"]] THEN
                    DISPLAY ("Transaction ID not found! Please
Input a Valid ID.")
                    CONTINUE
                ELSE
                    temp["id"] = user_input_id
                    BREAK
                END IF
            EXCEPT
                DISPLAY ("Invalid input!")
                CONTINUE
            END TRY
        END WHILE
        RETURN temp
    END FUNCTION
```

13. append_transaction()

```
    BEGIN FUNCTION append_transaction(dictionary, category)
        IF category is in transactions dictionary THEN
```

```
            APPEND dictionary to transactions[category]
        ELSE
            transactions[category] = [dictionary]
        END IF
    END FUNCTION
```

14. check_type()

```
BEGIN FUNCTION check_type(t_type)
      IF t_type is "I" or "Income" THEN
          RETURN "Income"
      ELSE IF t_type is "E" or "Expense" THEN
          RETURN "Expense"
      ELSE
          RETURN "Invalid Input!"
      END IF
  END FUNCTION
```

15. main_menu()

```
FUNCTION main():
    CALL load_functions
    display menu
    GET input
    while True do
        if input is equal to 1 then
            CALL add_transaction()

        else if input is equal to 2 then
            CALL view_transactions()

        else if input is equal to 3 then
            CALL update_transaction ()

        else if input is equal to 4 then
            CALL delete_transaction ()

        else if input is equal to 5 then
            CALL display_summery()

        else if input is equal to 6 then
            display "Exiting program"
            BREAK loop

        else do
            display "Invalid choice. Please try again"
        END if
    END while
END FUNCTION
```