LBYCPA1-EQ5

Programming Logic and Design Laboratory

Final Project Proposal

College Students' Finance Tracker

John Karlo C. Alamillo,
Vivienne Nicole M. Yap

INTRODUCTION

Financial independence in college can be challenging for students as many are managing their finances for the first time. These students often rely on keeping track of expenses mentally, resulting in inefficiencies and poor financial management. Not only can this result in debt accumulation and inefficient resource allocation, but this has also been proven to worsen students' academic performance.

According to the study on Filipino university students from rsisinternational.org, "(university) students are experiencing financial independence for the first time, which makes this a very difficult time for them… students find it challenging to take care of their fundamental necessities while attending classes (Moore et al., 2021)" Consequently, the students' financial worries not only heightens their stress levels, but also disrupts their ability to perform well in school ("The Relationship of Financial Worries and Psychological Distress Among Psychology Students," 2024).

Karissa Ellard, a financial aid specialist at Southern New Hampshire University (SNHU), offers a solution to this issue. She states that "budgeting helps you avoid overspending and accumulating too much debt. Without a budget, you might end up spending more on things you don't need, which could leave you short on money for important expenses" (*Why Is a Budget Important as a College Student?*, 2024).

Our finance tracker thus offers a solution for students who wish to improve their financial discipline, lessen their financial stress, and shift their focus on completing their studies. They will be able to see where their money goes explicitly, providing a much more efficient means of tracking expenses, and thus, prompting changes in how they spend and allocate their money.

Compared to other expense trackers, our program is tailored to college students' expenses. These include transportation fees, condo/apartment rent, and food expenses. The program will not contain expenses that most university students have to deal with such as health insurance, life insurance, and income tax payments. However, there will be an option for the user to input their own expense categories as needed, making the tracker customizable. Lastly, the program will include a goal budget, encouraging students to optimize their spending habits.

PROJECT DESCRIPTION

This project centers on the development of a College Finance Tracker, a

command-line application built using the Python programming language. Its primary objective is to help college students manage their personal finances more effectively by tracking their income, categorizing their expenses, and evaluating their monthly savings performance. Many students experience financial independence for the first time during college, often without the proper tools or knowledge to manage their money wisely. This application aims to bridge that gap by providing a structured and interactive platform that fosters budgeting habits, financial awareness, and self-discipline.

The system was developed using Python. The application is designed to run on standard terminal interfaces across various operating systems, including Windows and macOS/Linux.

The program is built with a modular structure that enhances readability and maintainability. Each function serves a specific purpose, such as validating user input, collecting expenses, displaying results, and offering financial suggestions. The system guides users through entering their allowance (either weekly or monthly), selecting or customizing expense categories, and inputting expense amounts per category. Weekly figures are automatically converted to monthly values for consistent analysis.

The system components and functionalities include the following:

1. User Interface (Command-Line Prompts): Provides interactive text-based prompts to guide users through each step of the budgeting process—from selecting income type to viewing the final summary.
2. Input Validators: Ensures that all user inputs are valid. The system includes validators for positive numeric values and yes/no responses to reduce the risk of incorrect data processing.
3. Expense Category Manager: Displays a list of default categories such as transportation, food, rent/utilities, school-related expenses, and allows users to add their own categories, provided they are unique and non-empty.
4. Expense Collector: Gathers the user's expense data for each category and, if necessary, converts weekly values into monthly figures for standardized analysis.
5. Budget Analyzer: Calculates total monthly expenses, net savings or deficit, daily and weekly spending averages, and projected annual savings. It also compares the actual net value against the user’s predefined goal budget to evaluate financial performance.

6. Summary and Recommendation Generator: Presents a comprehensive results display including a breakdown of each expense category with cost and percentage, highlights the top three spending areas, and provides practical financial tips tailored to the user's data.
7. Loop and Repeat Functionality: After completing an analysis, the system gives users the option to re-run the tracker with new or hypothetical data, allowing for multiple scenarios to be explored in one session.

METHODOLOGY

The method of implementing this project involved 5 key stages: system planning, data collection, program implementation, data analysis, and testing.

The initial phase involved identifying the program's core functionalities which include allowance/income tracking, customizable expense categories, savings goal comparison, and financial performance analysis. Data collection was primarily user-driven, as upon running the program, users were prompted to choose whether to input their income and expenses on a weekly or monthly basis. If weekly data was selected, all values were internally converted to monthly equivalents by multiplying them by four. Users are then asked to provide their total allowance and their target end-of-month savings goal. The program included default expense categories such as transportation, food, rent/utilities, and school-related costs, while also allowing users to add custom categories, provided they were unique and non-empty. Following this, users input their expenses for each category. Once all data is collected, the program proceeds with the analysis, in which it computes total monthly/weekly expenses, and a 12-month financial projection. It also calculates the percentage of total expenses spent on each category, identifying the top three categories and provides tips that the user can take to action based on the highest spending categories. Once the program has been completed, the final phase is launched, which involves testing. Various input scenarios were tested to ensure that all functionalities were performing as expected. Special attention was given to input validation, correct financial computations, and appropriate display of messages and suggestions. Through this implementation, this project effectively simulates a tool that promotes financial awareness and responsible budgeting among college students.

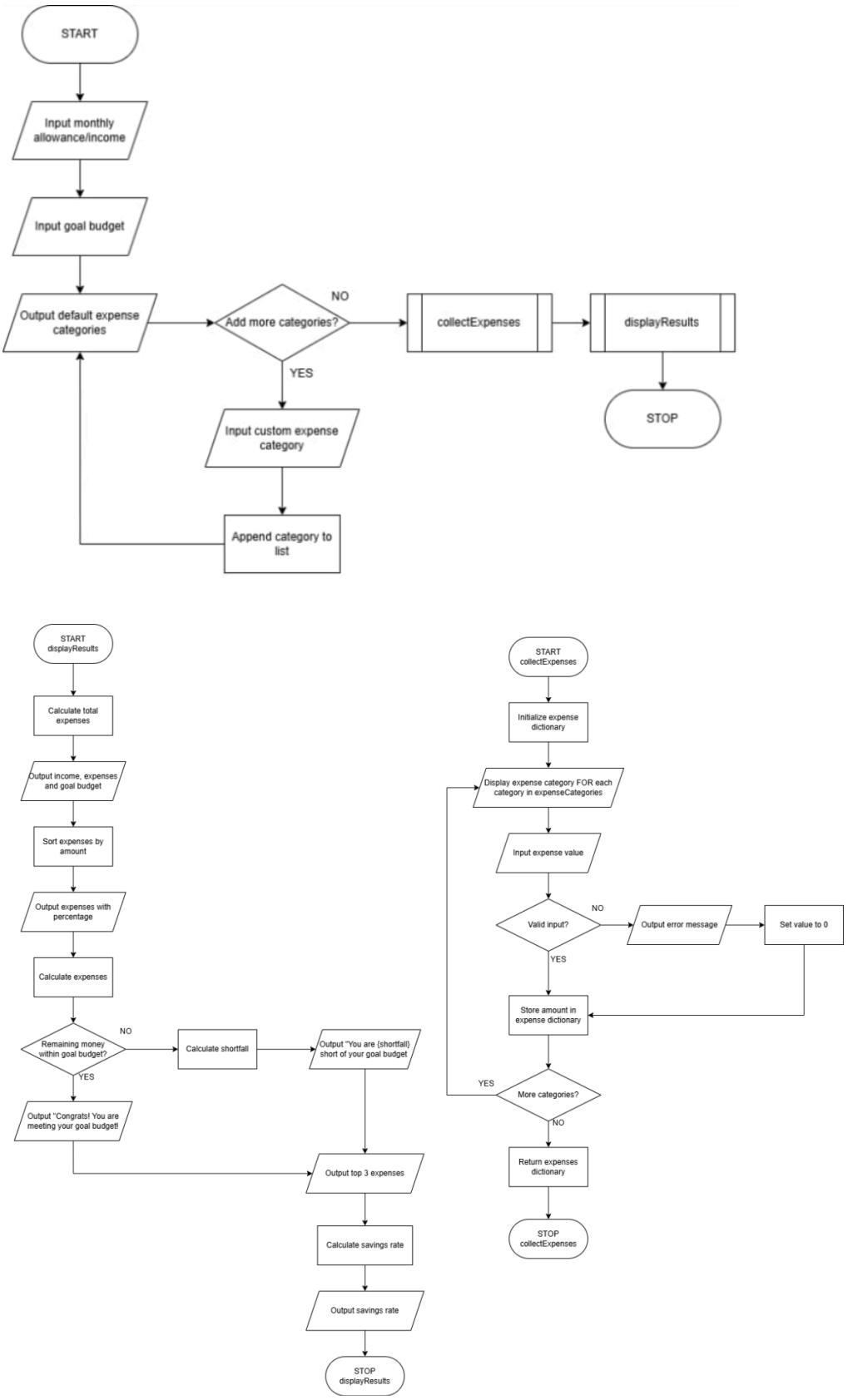

Fig. 1. Flowchart of the program

The programming features that were used were input/output, conditional statements, mathematical equations, and functions. Inputs will be used for the user to enter the value of their expenses, goal budget, and income/allowance. Outputs will be used to print the table of expenses as well as other analytics. Conditional statements will be used primarily for the user to be able to customize their list of expenses, inputting as many categories as they wish. Mathematical equations will also be used to calculate several analytics, such as the net loss/gain of the user and the percentage breakdown of expenses. Lastly, functions will be integrated throughout the program.

SCHEDULE OF ACTIVITIES

Task	Person in charge	Start Date	End date
Create Project Proposal	Both	Feb 12 2025	Feb 19 2025
Modify project proposal	Both	Feb 24, 2025	March 3, 2025
Create program	Both	March 3, 2025	March 10, 2025
Testing and debugging	Both	March 10, 2025	March 17, 2025
Modify program features	Both	March 17, 2025	March 24, 2025
Documentation/ Submission of program	Both	March 31, 2025	April 12, 2025

RESULTS AND DISCUSSION

Upon launching the program, users are greeted with a welcome message explaining the tool's purpose—to help analyze their monthly finances based on their allowance, expenses, and savings goal (Refer to ***Fig. 2***). The first interactive prompt asks whether the user wishes to track their finances on a weekly or monthly basis. If the user selects weekly, the system automatically multiplies all values by four to standardize data for monthly analysis. For instance, if the weekly allowance is PHP 1,000, the program will compute a monthly allowance of PHP 4,000. This normalization ensures consistent analysis regardless of the user's preference.

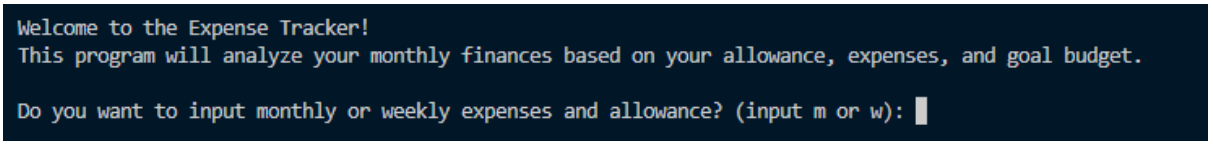

Fig 2. Screen on launch

After specifying the allowance, the user is asked to input their goal budget—how much they aim to save by the end of the month (Refer to ***Fig. 3.***). Next, the program displays a set of default expense categories, including the ff:

- Transportation fees
- School-related expenses
- Food
- Rent/Utilities
- Miscellaneous
- Personal expenses

Users are given the option to add custom categories, which are validated to be non-empty and unique. This customization supports varied student lifestyles, allowing for more personalized and relevant financial tracking.

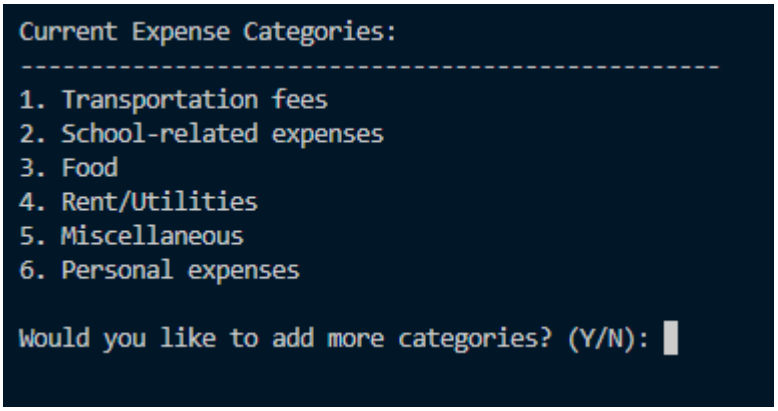

Fig 3. Expense category menu

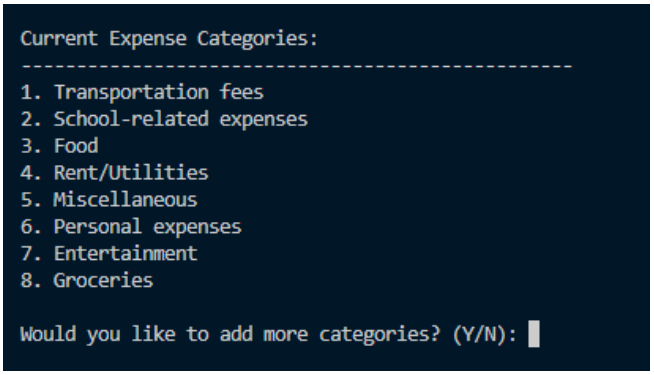

Fig. 4. Expense category with menu with custom categories

Once the final list of categories is established, the program prompts the user to input their expenses per category. If the user had initially chosen weekly tracking,

each input is again multiplied by four to convert to monthly values (Refer to ***Fig. 5***).

```
Great! You've entered all your expense categories.
Now, input your expenses for each category. If a category doesn't apply to you, enter 0.

Enter the weekly amount for Transportation fees in PHP: 100
Enter the weekly amount for School-related expenses in PHP: 24
Enter the weekly amount for Food in PHP: 500
Enter the weekly amount for Rent/Utilities in PHP: 0
Enter the weekly amount for Miscellaneous in PHP: 15
Enter the weekly amount for Personal expenses in PHP: 90
Enter the weekly amount for Entertainment in PHP: 150
Enter the weekly amount for Groceries in PHP: 270

Press enter to view your budget summary...
```

Fig. 5. Expense input

After all expense data is collected, the system performs several calculations:

- Total Monthly Expenses
- Net Value (Income – Expenses)
- Daily and Weekly Averages
- Percentage of Allowance Spent or Saved
- Projection for 12 Months (either savings or deficit)

The results are presented in a clear, structured format (Refer to ***Fig. 6***). For example, if a student had a monthly allowance of PHP 5,000, spent PHP 4,200, and set a goal budget of PHP 1,000, the program would compute a net value of PHP 800, falling PHP 200 short of the goal. This would be flagged with a caution message, encouraging the user to review their spending.

Moreover, the program includes a detailed expense breakdown, showing:

- Each category's cost
- Its percentage of the total expenses
- A list of the top 3 spending categories

This visual breakdown gives users a better grasp of where their money goes each month. If default expenses appear in the top three expenses, relevant budgeting tips would be displayed (e.g., "Cook instead of ordering food").

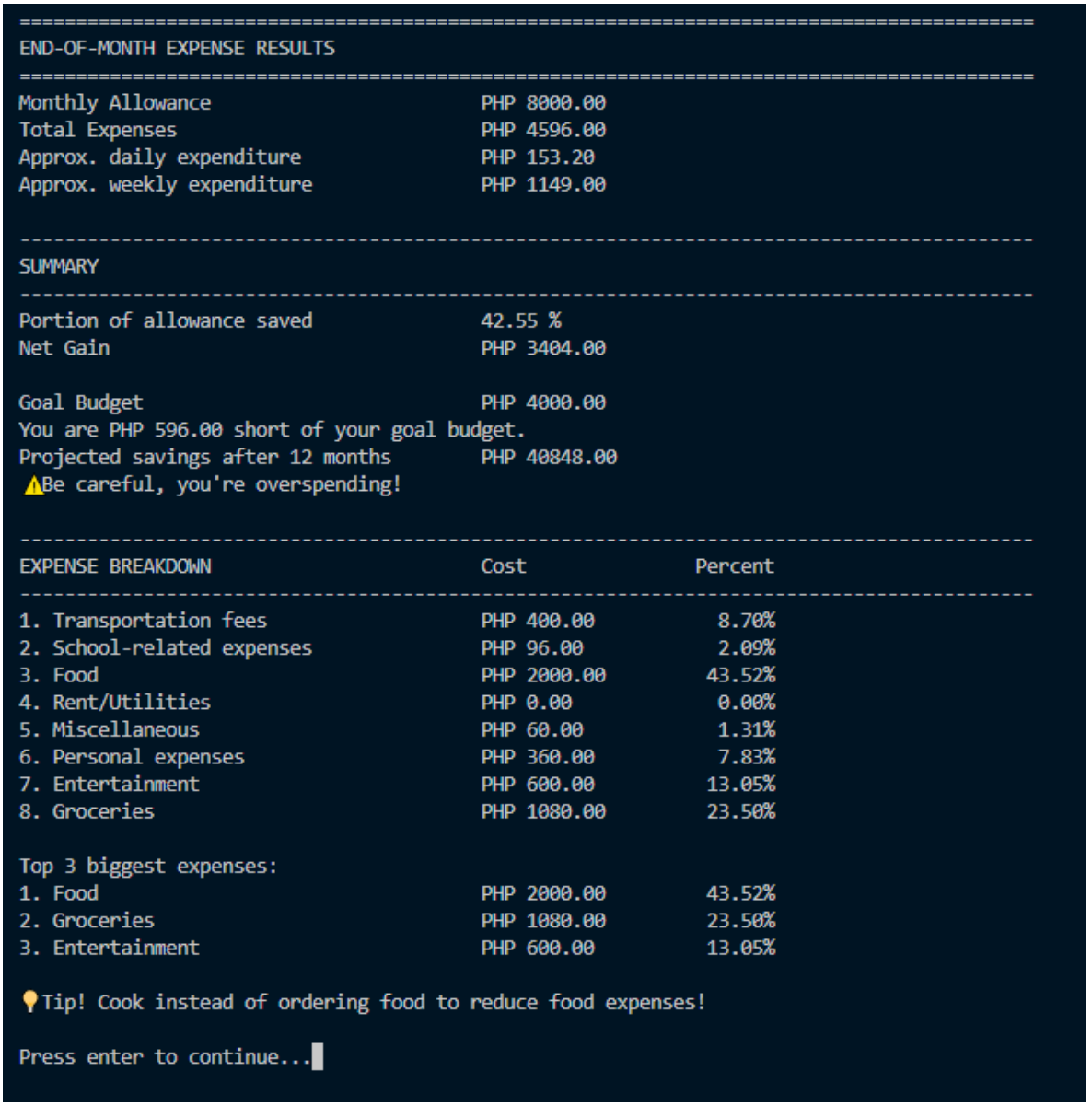

Fig. 6. Analysis and Budget Summary Display

The program also ensures that all invalid inputs are caught, preventing the program from crashing and allowing the user to continue using the tracker. This includes inputting negative values for the allowance or budget, and inputting invalid characters when asked different prompts, as seen in ***Fig 7***.

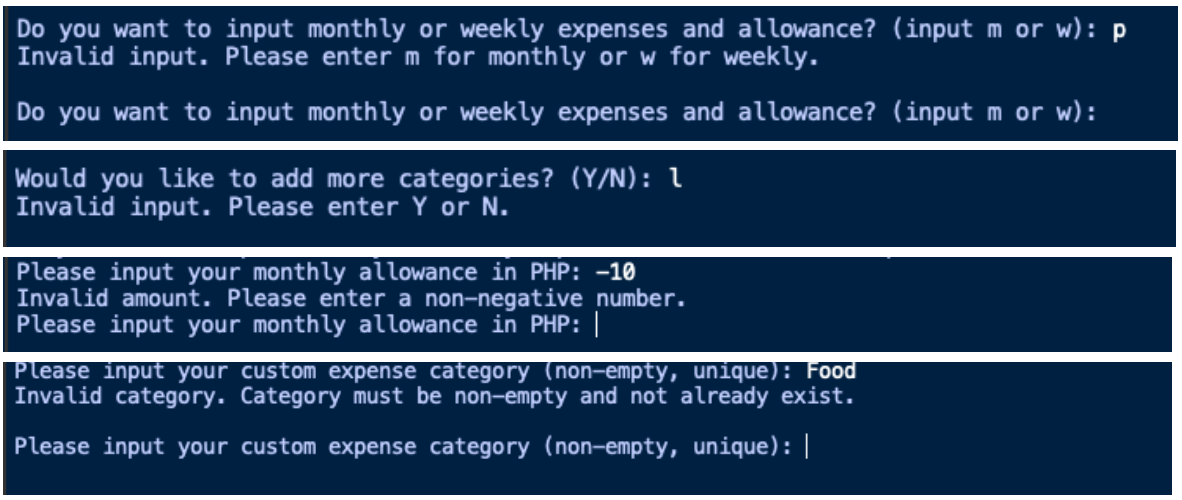

Fig 7. Invalid input handling

The program showed strong engagement throughout the testing. In terms of data accuracy, the program consistently applied validation and conversion logic, such as validating that inputted values are non-negative, and transforming weekly figures into monthly values. This ensured the integrity of the data used in calculations and prevented issues caused by invalid or inconsistent input. Beyond its functional use, the program also delivered educational value. The inclusion of goal-setting, net value computation, and performance-based feedback (e.g., alerts for deficits or congratulations for reaching a goal) helped reinforce budgeting concepts and encouraged responsible financial behavior. Tips tailored to the user's top spending categories further contributed to this learning experience. Another key insight was the program's flexibility and adaptability. While designed mainly for college students, its structure and logic can easily be modified to accommodate other users, such as working professionals or households, by adjusting category labels or financial terminology.

CONCLUSION AND RECOMMENDATIONS

This tracker effectively meets its objective of helping college students manage their personal finances through an interactive budgeting system. It provides a user-friendly interface that allows students to input their financial information, customize their spending categories, and evaluate their financial performance as a whole. The program is able to perform accurate calculations, generate relevant financial insights, and offer tips that users can use based on their top expense categories. For future improvements, it is recommended that a graphical user interface (GUI) be developed in order to enhance user experience, especially for those who are unfamiliar with command line. Integration with spreadsheet export

features can also help users retain and monitor their spending habits and budgeting data over time. Lastly, developing a mobile application can expand its reach and practicality to ensure that students are able to track their finances on the go.

REFERENCES

1. The relationship of financial worries and psychological distress among psychology students. (2024). *IJRISS*. https://rsisinternational.org/journals/ijriss/Digital-Library/volume-8-issue-3/1614-1644.pdf
2. *Why is a Budget Important as a College Student?* (2024, October 24). https://www.snhu.edu/about-us/newsroom/education/budgeting-for-college-students

Appendix A: Program Code

```
# Name: Vivienne Nicole M. Yap, John Karlo Alamillo
# Course & Section: BS-CPE, EQ5
# Date & Time Completed: Apr 11 2025
# Final project code: Finance Tracker

# This project is designed to help college students track their allowance and expenses.
Users can choose between weekly or monthly allowance input.
# They can add custom expense categories and enter the amount for each.
# The program calculates total expenses, savings or deficits, and gives budgeting tips.

import os

def clear_screen(): #to clear the screen
    os.system('cls' if os.name == 'nt' else 'clear')

def validateInput(prompt, validator_func, error_message): #general invalid input function
    while True:
        try:
            user_input = input(prompt)
            if validator_func(user_input):
                return user_input
            print(error_message)
        except ValueError:
            print(error_message)

def validatePositiveFloat(value): #to ensure non-negative numbers are entered
    try:
        float_value = float(value)
        return float_value >= 0
    except ValueError:
        return False

def validateYesNo(value):
    """
    Validate Yes/No input.
    """
    return value.upper() in ['Y', 'N']

def addExpenseCategory(): #function to add expense categories to default list of expenses
    expenseCategories = [
        "Transportation fees",
        "School-related expenses",
```

```
        "Food",
        "Rent/Utilities",
        "Miscellaneous",
        "Personal expenses"
    ]

    while True:
        clear_screen()

        #display current expense categories
        print('\nCurrent Expense Categories:')
        print('-' * 50)
        for i, category in enumerate(expenseCategories, 1):
            print(f"{i}. {category}")

        #input and validate add more categories choice (either yes or no)
        addMoreChoice = validateInput(
            '\nWould you like to add more categories? (Y/N): ',
            validateYesNo,
            "Invalid input. Please enter Y or N.\n"
        ).upper()

        if addMoreChoice != "Y":
            break

        #input and validate new category input: must not already exist and must not be empty
        newCategory = validateInput(
            'Please input your custom expense category (non-empty, unique): ',
            lambda x: x.strip() and x not in expenseCategories,
            "Invalid category. Category must be non-empty and not already exist.\n"
        )
        expenseCategories.append(newCategory)

    return expenseCategories

def collectExpenses(expenseCategories, isWeekly): #function to input expense amount per
category
    expenses = {}
    timeframe = 'weekly' if isWeekly else 'monthly'

    clear_screen()
    print("Great! You've entered all your expense categories.")
    print("Now, input your expenses for each category. If a category doesn't apply to you,
enter 0.\n")

    #loop through each categroy, inputing the respective amount
    for category in expenseCategories:
        # Validate expense amount for each category
```

```
        amount = float(validateInput(
            f"Enter the {timeframe} amount for {category} in PHP: ",
            validatePositiveFloat,
            "Invalid amount. Please enter a non-negative number.\n"
        ))

        #convert weekly expense to its monthly equivalent if needed(multiply by 4)
        if isWeekly:
            amount *= 4

        expenses[category] = amount

    return expenses

def displayResults(monthlyIncome, expenses, goalBudget): #function to calculate and
display results

    totalExpenses = sum(amount for amount in expenses.values())

    clear_screen()
    print('='*90)
    print("END-OF-MONTH EXPENSE RESULTS") #print table header
    print('='*90)

    #print monthly allowance, total expenses, daily and weekly average expenditure
    print(f"{'Monthly Allowance': <38} {'PHP':>5} {monthlyIncome:<10.2f}")

    print(f"{'Total Expenses':<38} {'PHP':>5} {totalExpenses:<10.2f}")
    net = monthlyIncome - totalExpenses
    projectedValue = 12 * net

    daily_avg = totalExpenses / 30
    print(f"{'Approx. daily expenditure':<38} {'PHP':>5} {daily_avg:<10.2f} {"}") #Assuming 30
days in the month

    weekly_avg = totalExpenses / 4
    print(f"{'Approx. weekly expenditure':<38} {'PHP':>5} {weekly_avg:<10.2f}") #Assuming 4
weeks in the month

    print('\n'+'-'*90)
    print("SUMMARY")
    print('-'*90)

    #display net value and portion of allowance saved/spent:
    if net == 0: #for no net gain/deficit
        print(f"{'Net value': <38} {'PHP':>5} {net:<10.2f}")
        print("You did not experience a net gain or deficit this month.")
    elif net < 0: #for net deficit
```

```
        portionSpent = (totalExpenses / monthlyIncome) * 100
        print(f"{'Portion of allowance spent': <40} {portionSpent:<3.2f} {'%':<3}") #display portion
of allowance spent
        print(f"{'Net Deficit': <38} {'PHP':>5} {net:<10.2f}")
        print("❌ Deficit alert!")
    else: #for net gain
        portionSaved = (net / monthlyIncome) * 100
        print(f"{'Portion of allowance saved': <40} {portionSaved:<3.2f} {'%':<3}") #display
portion of allowance saved
        print(f"{'Net Gain': <38} {'PHP':>5} {net:<10.2f}")

    print(f"{'\nGoal Budget':<39} {'PHP':>5} {goalBudget:<10.2f}")     # Display how user is
faring towards goal budget
    if net == goalBudget: #if net gain is exactly equal to the goal budget
        print("Congratulations! You are exactly meeting your set goal budget!")
        print(f"{'Projected savings after 12 months':<38} {'PHP':>5} {projectedValue:<10.2f}")
        print("👍 Great budgeting!")
    else:  #for net values not equal to the goal budget
        difference = abs(net - goalBudget)
        if net > goalBudget: #exceeding goal budget
            print(f"Congratulations! You have met your set goal budget and still have PHP
{difference:.2f} remaining!")
            print(f"{'Projected savings after 12 months':<38} {'PHP':>5} {projectedValue:<10.2f}")
            print("👍 Great budgeting!")
        elif net < goalBudget and net >= 0: #net is less than goal budget but non-negative
            print(f"You are PHP {difference:.2f} short of your goal budget.")
            print(f"{'Projected savings after 12 months':<38} {'PHP':>5} {projectedValue:<10.2f}")
            print("⚠️ Be careful, you're overspending!")
        else: #net is negative(and thus less than goal budget)
            print(f"You are PHP {difference:.2f} short of your goal budget.")
            print(f"{'Projected deficit after 12 months':<38} {'PHP':>5} {projectedValue:<10.2f}")
            print("⚠️ Be careful, you're overspending!")

    print("\n" + '-'*90)
    print(f"{'EXPENSE BREAKDOWN': <40} {'Cost' :<15} {'Percent':>10}")
    print('-'*90)

    def percentage(amount): #function to calculate a category's portion of the total expense
        return (amount / totalExpenses) * 100 if totalExpenses > 0 else 0

    #display all expenses with percentages
    for i, (category, amount) in enumerate(expenses.items(),1):
        print(f"{i}. {category:<35} {'PHP':>5} {amount:<10.2f} {percentage(amount):>10.2f}%")

    #display Top 3 biggest expenses
    print("\nTop 3 biggest expenses:")
    sortedExpenses = sorted(expenses.items(), key=lambda x: x[1], reverse=True) #sort by
descending order
```

```
    topExpenses = sortedExpenses[:3]  #get top 3 expenses
    for i, (item, cost) in enumerate(topExpenses, 1):
        print(f"{i}. {item:<35} {'PHP':>5} {cost:<10.2f} {percentage(cost):>10.2f}%")

    #display saving tips for top 3 expenses
    topExpenseCategories = [item for item, cost in topExpenses]
    print('')
    if "Transportation fees" in topExpenseCategories:
        print("🚗 Tip! Walk, carpool, or commute to reduce transportation costs!")
    if "School-related expenses" in topExpenseCategories:
        print("📚Tip! Buy used textbooks, use library resources, and borrow materials from
classmates to reduce school-related expenses!")
    if "Food" in topExpenseCategories:
        print("👨‍🍳Tip! Cook instead of ordering food to reduce food expenses!")
    if "Rent/Utilities" in topExpenseCategories:
        print("⚡Tip! Find roommates to split rent and utilities, and conserve electricity and
water!")
    if "Personal expenses" in topExpenseCategories or "Miscellaneous" in
topExpenseCategories:
        print("💡 Tip! Avoid impulse purchases and avail of second-hand items to reduce
unnecessary expenses!")
    print('')

def runTracker(): #function to run tracker
    clear_screen()
    print('Welcome to the Expense Tracker!')
    print('This program will analyze your monthly finances based on your allowance,
expenses, and goal budget.')

    #input and validate income input type (weekly or monthly)
    while True:
        choice = validateInput(
            '\nDo you want to input monthly or weekly expenses and allowance? (input m or w): ',
            lambda x: x.lower() in ['m', 'w'],
            "Invalid input. Please enter m for monthly or w for weekly."
        ).lower()

        #input and validate income amount
        if choice == 'w':
            weeklyAllowance = float(validateInput(
                'Please input your weekly allowance in PHP: ',
                validatePositiveFloat,
                "Invalid amount. Please enter a non-negative number."
            ))
            monthlyIncome = 4 * weeklyAllowance
            isWeekly = True
            break
        elif choice == 'm':
```

```
            monthlyIncome = float(validateInput(
                'Please input your monthly allowance in PHP: ',
                validatePositiveFloat,
                "Invalid amount. Please enter a non-negative number."
            ))
            isWeekly = False
            break

    #input and validate goal budget
    goalBudget = float(validateInput(
        'Please input your end-of-month goal budget in PHP: ',
        validatePositiveFloat,
        "Invalid amount. Please enter a non-negative number."
    ))


    expenseCategories = addExpenseCategory() #allow user to add expense categories
    expenseDictionary = collectExpenses(expenseCategories, isWeekly) #user inputs all
expense amounts
    print("")
    input("Press enter to view your budget summary...")
    displayResults(monthlyIncome, expenseDictionary, goalBudget) #calculate and display
analysis of expenses

def main():
    while True:
        runTracker()

        input("Press enter to continue...")
        clear_screen()
        #allow user to run the tracker again
        again = validateInput(
            "\nWould you like to track your finances again or input hypothetical expense values?
(Y/N): ",
            validateYesNo,
            "Invalid input. Please enter Y or N."
        ).upper()
        if again != 'Y':
            print("\nThank you for using our college finance tracker! Goodbye and happy
budgeting! 💰\n")
            break

main()
```