DSA Project Progress Report

Course Code: 201L DSA	**Program:** BSCpE
Course Title: Data Structure Analysis	**Date Performed:** October 11, 2025
Section: 2-B	**Date Submitted:** October 11, 2025
Name: Kurt Marwin C. Mamanao Justine Poliño Jan Lawrence M. Ramos Junichiro H. Uy Yuan Hessed O. Vasig	**Instructor:** Engr. Maria Rizette H. Sayo

• Objectives

This Project “Money Rider” aims to:

- To add Undo and Redo button
- To add Json Dile

2. Discussion

To improve the "Money Rider" project, we implemented two significant changes: JSON file integration for persistent data storage and a feature called "undo&redo button" to give users greater control and error correction capabilities. We used structured data handling with earnings information while action history was stored in logical loops using stack-based logic, which immediately makes the app more usable and reliable as if one were on day-to-day money management. This helped our project become even more effective than before.

3. Materials and Equipment

Computer: used to make the source code for the Skill-Test

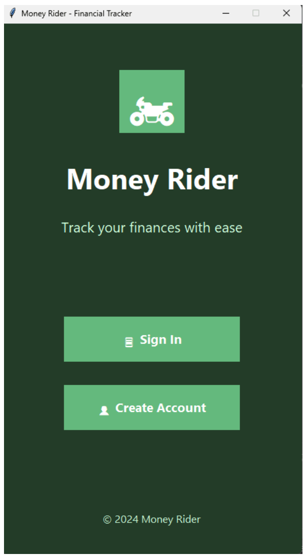
Visual Studio Code: used for running the program

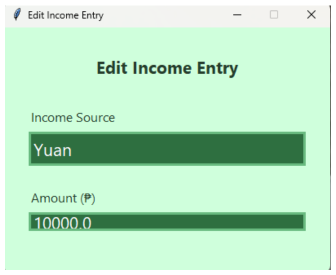
Github: used to collaborate and manipulate the program

Mouse: used to navigate the Computer

Keyboard: Used to type the source code

4. Output

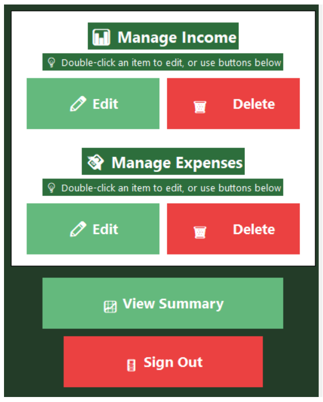

Money Rider - Financial Tracker
Money Rider
Track your finances with ease
Sign In
Create Account
© 2024 Money Rider

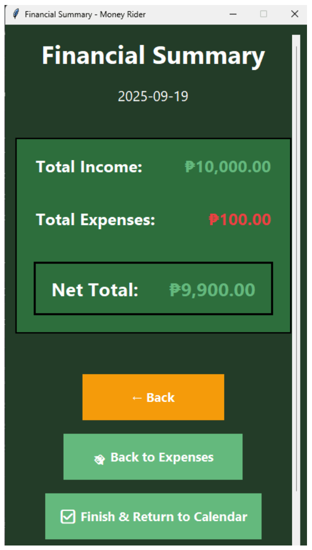


Edit Income Entry

Income Source: Yuan

Amount (₱): 10000.0


Manage Income
Double-click an item to edit, or use buttons below
Edit
Delete
Manage Expenses
Double-click an item to edit, or use buttons below
Edit
Delete
View Summary
Sign Out



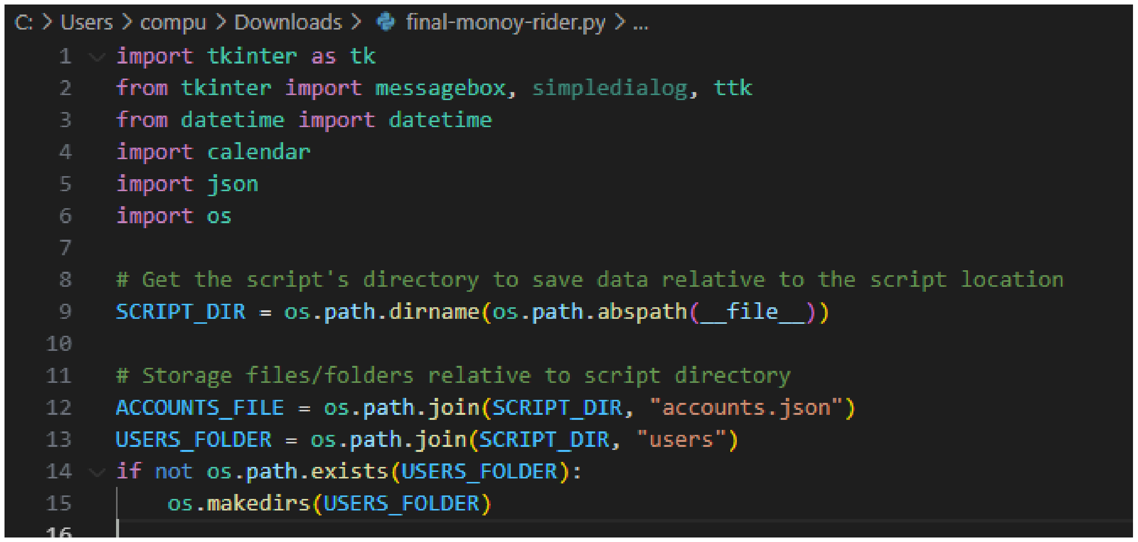
Financial Summary

2025-09-19

Total Income: ₱10,000.00

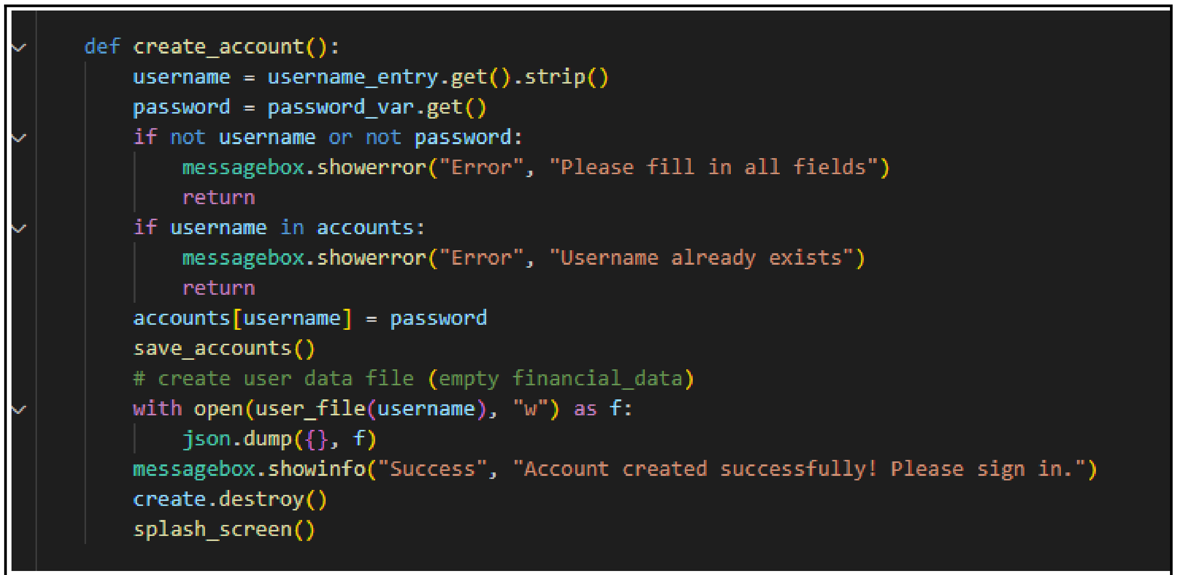
Total Expenses: ₱100.00

Net Total: ₱9,900.00

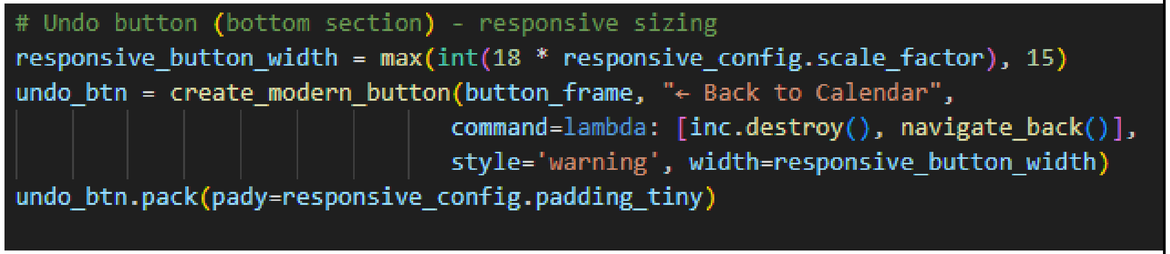


C: > Users > compu > Downloads > final-monoy-rider.py > ...

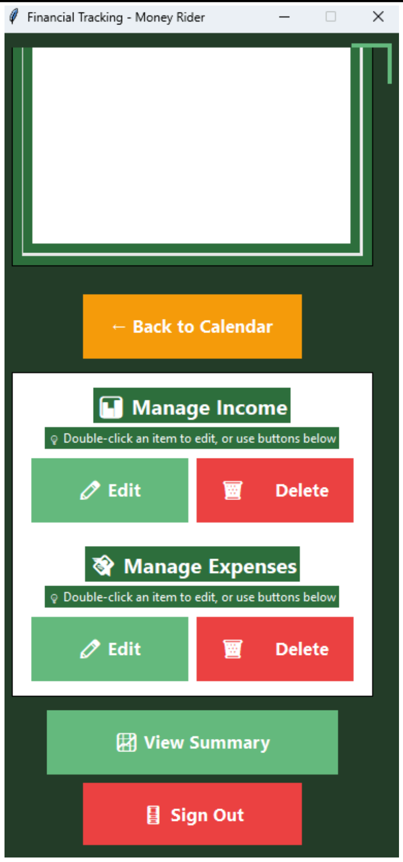
```python
import tkinter as tk
from tkinter import messagebox, simpledialog, ttk
from datetime import datetime
import calendar
import json
import os

# Get the script's directory to save data relative to the script location
SCRIPT_DIR = os.path.dirname(os.path.abspath(__file__))

# Storage files/folders relative to script directory
ACCOUNTS_FILE = os.path.join(SCRIPT_DIR, "accounts.json")
USERS_FOLDER = os.path.join(SCRIPT_DIR, "users")
if not os.path.exists(USERS_FOLDER):
    os.makedirs(USERS_FOLDER)
```



```python
def create_account():
    username = username_entry.get().strip()
    password = password_var.get()
    if not username or not password:
        messagebox.showerror("Error", "Please fill in all fields")
        return
    if username in accounts:
        messagebox.showerror("Error", "Username already exists")
        return
    accounts[username] = password
    save_accounts()
    # create user data file (empty financial_data)
    with open(user_file(username), "w") as f:
        json.dump({}, f)
    messagebox.showinfo("Success", "Account created successfully! Please sign in.")
    create.destroy()
    splash_screen()
```



```python
# Undo button (bottom section) - responsive sizing
responsive_button_width = max(int(18 * responsive_config.scale_factor), 15)
undo_btn = create_modern_button(button_frame, "← Back to Calendar",
                                command=lambda: [inc.destroy(), navigate_back()],
                                style='warning', width=responsive_button_width)
undo_btn.pack(pady=responsive_config.padding_tiny)
```


Financial Tracking - Money Rider
← Back to Calendar
Manage Income
Double-click an item to edit, or use buttons below
Edit
Delete
Manage Expenses
Double-click an item to edit, or use buttons below
Edit
Delete
View Summary
Sign Out

5. Procedure

We designed the "Money Rider" program with procedures: initial design used history stacks (analytes in arrays stored with user actions) to build undo and redone feature, making data repatriation and reversal easy; reading from this file as an accounting string and saving it in / on - iframes while inside the application also uses conversion of account information to a JSON string for reading and analysis of those same details, which is added to the JX file into its running code. Throughout this process, the array concept was central to manipulating and managing the data for the calendar and financial entries.

6. Conclusion

This progress report demonstrates how we employed data structures to further manipulate the program. First, we programmed the red do and undo functions first. We used stack structures to keep track of user input history so we could sequence restore/revert actions that were not in use. Additionally, we utilized the management of JSON files by implementing functions that transformed the financial data array into a string for storage and analysis, guaranteeing data consistency across all sessions. In order to create a permanent record of an individual's financial information, the stack and data serialization were utilized to implement the undo/rede commands.