Personal Budgeting Application – Process Summary

Justice Williamson

Southern New Hampshire University

CS-500: Introduction to Programming

Instructor: Omar Hamdy

October 26, 2025

Object-Oriented Programming (OOP) Principles

The Personal Budgeting Application contains three main classes that form the structure of the program.

The UserBudget class acts as the main controller that manages categories and transactions while also handling income and savings goals.
The Transaction class represents a financial record that includes details such as amount, category, date, and description.
The BudgetCategory class allows users to create spending categories with optional monthly spending limits.

Grouping related data and behaviors within these classes helps keep the program organized and easy to maintain. The program uses these classes to manage information instead of relying on raw data. This structure shows how encapsulation and abstraction keep the system organized, while the relationship between UserBudget and Transaction demonstrates composition through the way each budget contains multiple transactions.

Functions and Modules for Reusability

The program uses a modular design that makes it easier to modify, maintain, and expand.

Several input validation functions handle user data checks in a consistent way. The File I/O functions take care of saving and loading information to and from storage. The two main functions, print_menu() and main(), manage how the user interacts with the program through the command line.

The program includes clear docstrings, consistent formatting, and meaningful variable names that make the code simple to read and follow. Each section of the code has a specific purpose, and this separation makes it possible to update one area without affecting others.

Coding Style and Readability

The program uses descriptive variable names and detailed comments that help explain how each section works. Each part of the program performs its own task, such as handling input, performing calculations, managing files, or displaying information to the user. This structure keeps the logic organized and easy to follow.

The command-line interface presents commands in a clear and consistent format that users can easily understand. The overall layout of the code makes it simple for anyone reading it to trace the flow of information and see how the system operates.

Use of Python's Libraries

The application uses several Python libraries to handle essential functions.

It uses the json library to save and reload user data so that information remains available between sessions. The datetime library validates and formats dates that users enter for their income and expense records. The pathlib library makes sure files are created and saved correctly on any computer system.

The program also includes a small visualization feature that uses matplotlib to create a bar chart showing spending by category. This feature helps users see where their money is going each month in an easy-to-understand visual format.

Secure and Reliable Coding Practices

The program includes safety checks and practices that keep it dependable and protect user data.

It validates all user input to prevent blank entries, negative numbers, and invalid dates. The program also uses try and except statements to handle user input errors without crashing. When saving data, it first writes to a temporary file before replacing the original one. This prevents data loss if the program closes unexpectedly.

These steps help the program stay reliable and make sure that information remains safe and accurate during use.

Error Handling and Stability

The program displays an error message whenever a user enters incorrect information, then returns to the main menu so the user can try again. The code includes exception handling around all major actions such as adding income, recording expenses, and updating categories. This design prevents the program from crashing if something unexpected happens.

This system keeps the application running smoothly even when mistakes occur. It creates a better experience for users and ensures that the information in the program remains accurate.

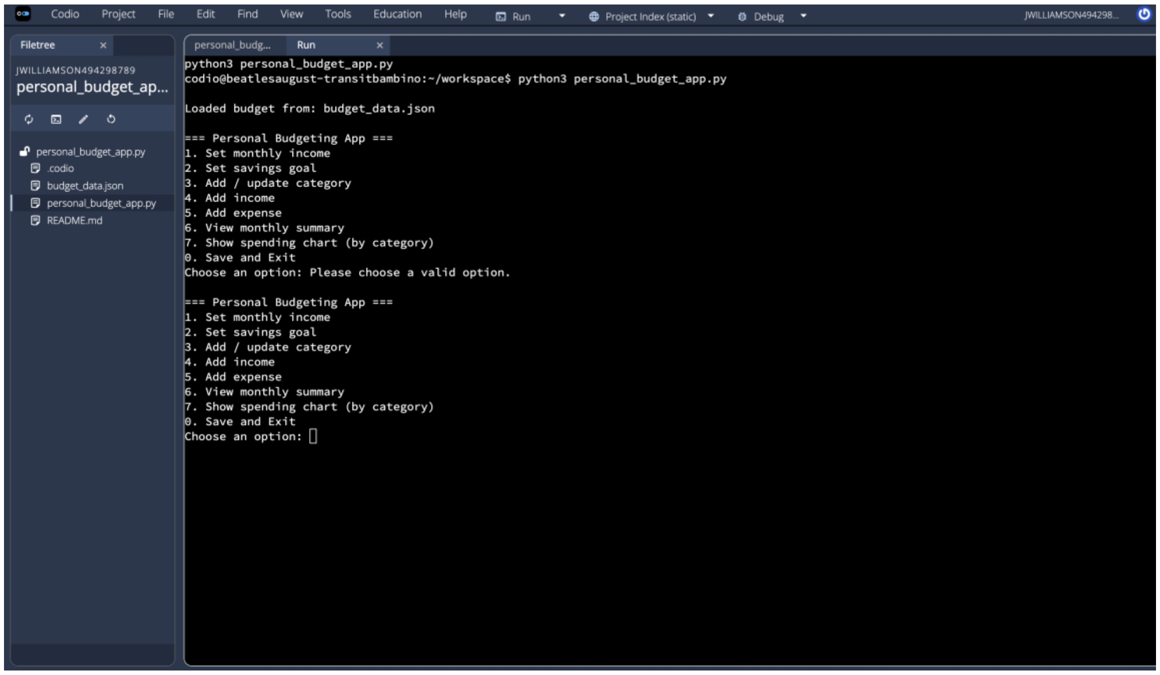
Testing

Manual testing was performed to confirm that each feature worked correctly.

Feature Tested	Description	Result
Monthly Income Input	Entered valid and invalid income values.	Program accepted valid input and rejected invalid entries.
Expense Entry	Added expenses with categories, descriptions, and dates.	Expenses were categorized and saved accurately.
Savings Goal	Set and updated savings goals.	Progress was calculated correctly.
Summary Output	Viewed the monthly summary.	Totals and category breakdown were accurate.
Data Persistence	Restarted the program to confirm data reloaded from file.	Data was successfully retained.
Visualization	Tested the chart feature.	Displayed spending by category in a bar chart.

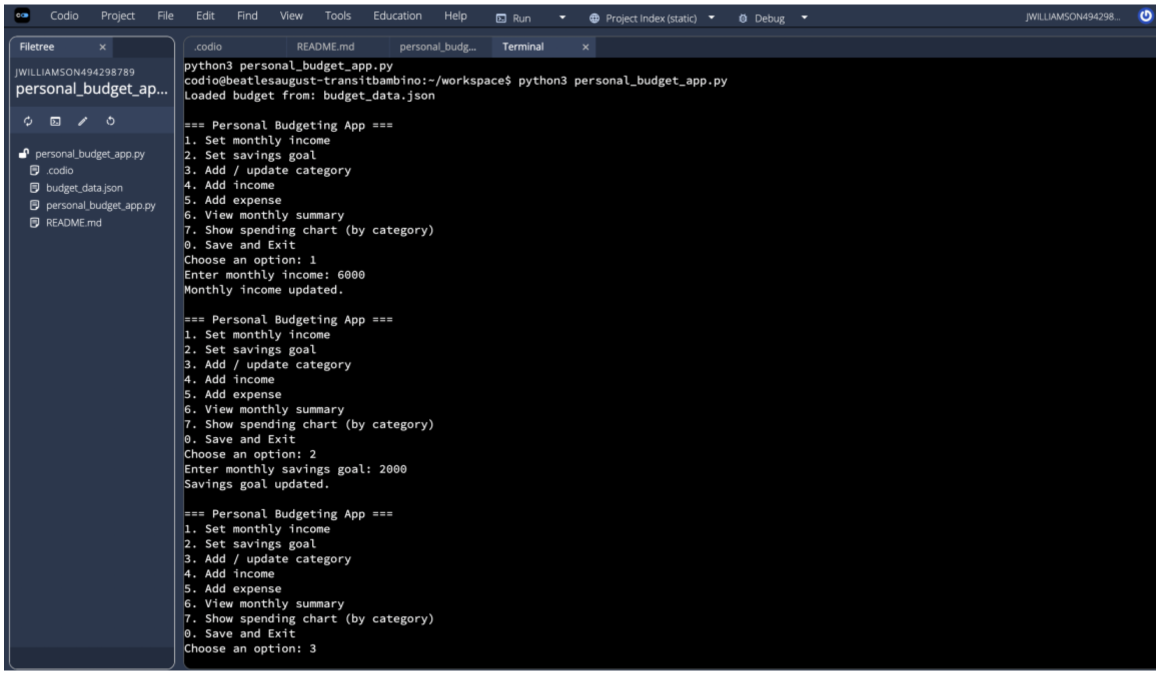
Screenshots

Figure 1. Program main menu running in Codio IDE.



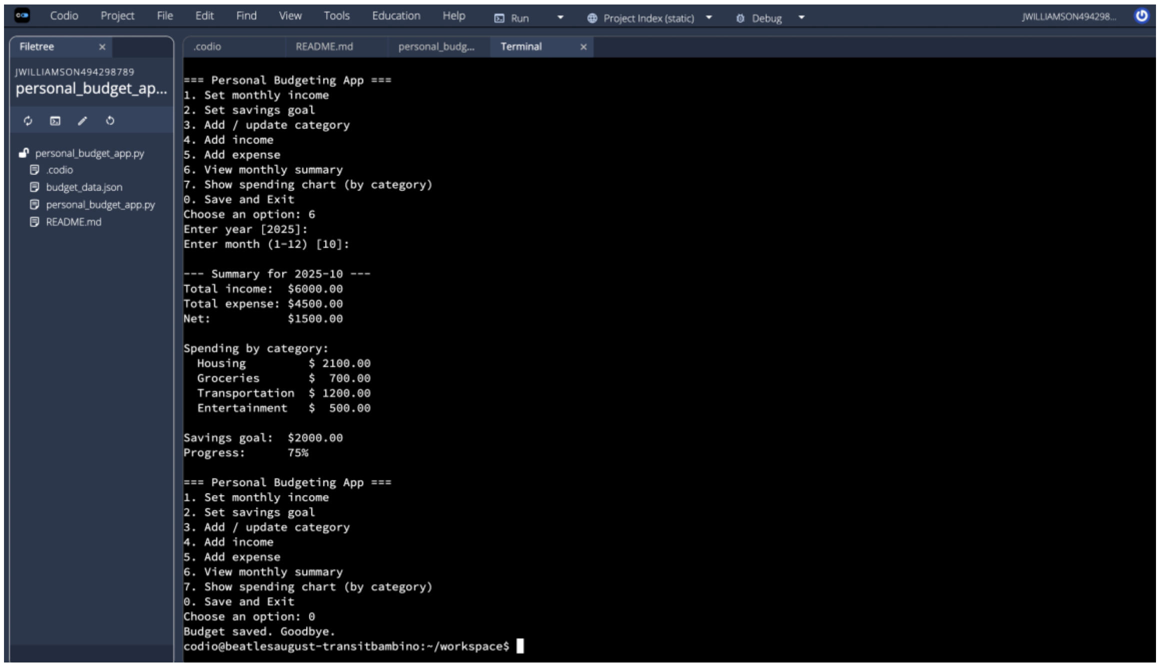
```
python3 personal_budget_app.py
codio@beatlesaugust-transitbambino:~/workspace$ python3 personal_budget_app.py

Loaded budget from: budget_data.json

=== Personal Budgeting App ===
1. Set monthly income
2. Set savings goal
3. Add / update category
4. Add income
5. Add expense
6. View monthly summary
7. Show spending chart (by category)
0. Save and Exit
Choose an option: Please choose a valid option.

=== Personal Budgeting App ===
1. Set monthly income
2. Set savings goal
3. Add / update category
4. Add income
5. Add expense
6. View monthly summary
7. Show spending chart (by category)
0. Save and Exit
Choose an option: 
```

Figure 2. Adding income and expense entries in the CLI.



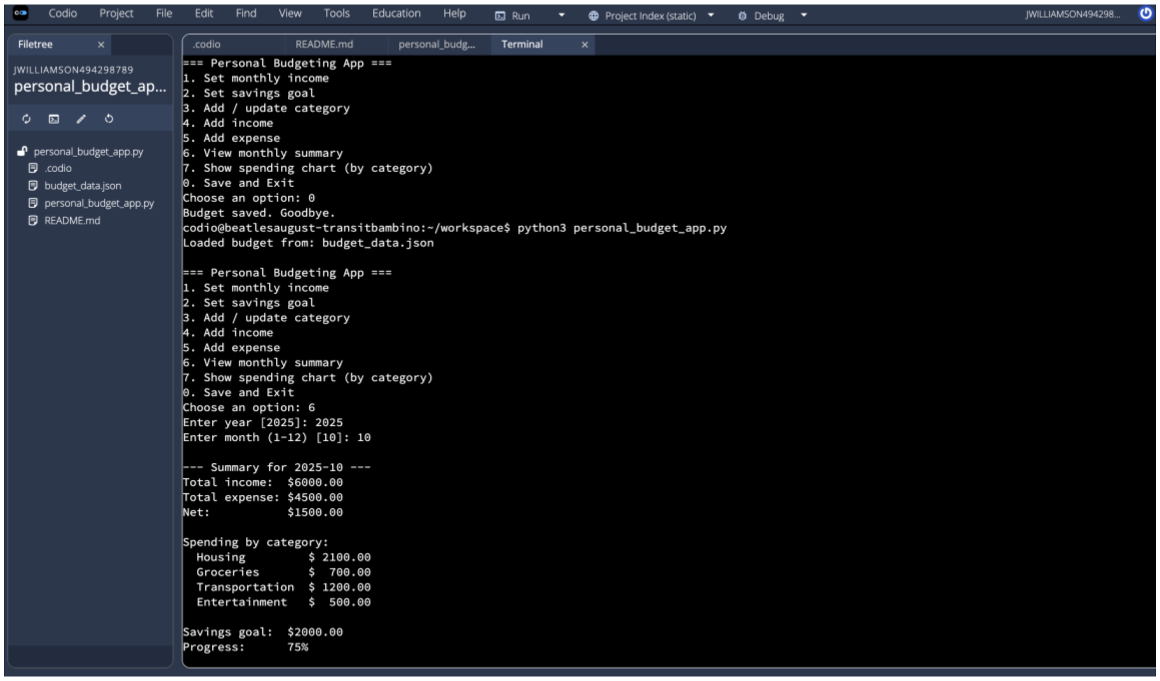
```
python3 personal_budget_app.py
codio@beatlesaugust-transitbambino:~/workspace$ python3 personal_budget_app.py
Loaded budget from: budget_data.json

=== Personal Budgeting App ===
1. Set monthly income
2. Set savings goal
3. Add / update category
4. Add income
5. Add expense
6. View monthly summary
7. Show spending chart (by category)
0. Save and Exit
Choose an option: 1
Enter monthly income: 6000
Monthly income updated.

=== Personal Budgeting App ===
1. Set monthly income
2. Set savings goal
3. Add / update category
4. Add income
5. Add expense
6. View monthly summary
7. Show spending chart (by category)
0. Save and Exit
Choose an option: 2
Enter monthly savings goal: 2000
Savings goal updated.

=== Personal Budgeting App ===
1. Set monthly income
2. Set savings goal
3. Add / update category
4. Add income
5. Add expense
6. View monthly summary
7. Show spending chart (by category)
0. Save and Exit
Choose an option: 3
```

Figure 3. Monthly summary output showing category breakdown and savings progress.



```
=== Personal Budgeting App ===
1. Set monthly income
2. Set savings goal
3. Add / update category
4. Add income
5. Add expense
6. View monthly summary
7. Show spending chart (by category)
0. Save and Exit
Choose an option: 6
Enter year [2025]:
Enter month (1-12) [10]:

--- Summary for 2025-10 ---
Total income:  $6000.00
Total expense: $4500.00
Net:           $1500.00

Spending by category:
  Housing         $ 2100.00
  Groceries       $  700.00
  Transportation  $ 1200.00
  Entertainment   $  500.00

Savings goal:  $2000.00
Progress:      75%

=== Personal Budgeting App ===
1. Set monthly income
2. Set savings goal
3. Add / update category
4. Add income
5. Add expense
6. View monthly summary
7. Show spending chart (by category)
0. Save and Exit
Choose an option: 0
Budget saved. Goodbye.
codio@beatlesaugust-transitbambino:~/workspace$
```

Figure 4. Data persistence verification after restart (same totals reloaded from budget_data.json).



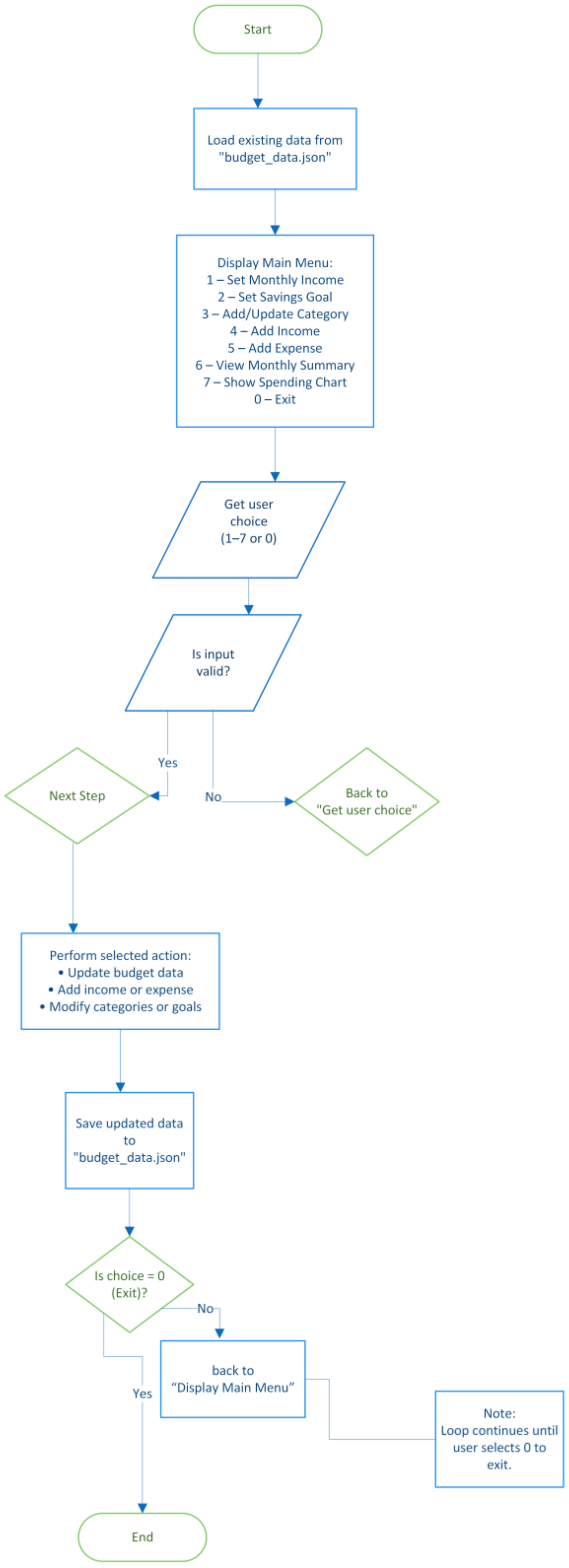
```
=== Personal Budgeting App ===
1. Set monthly income
2. Set savings goal
3. Add / update category
4. Add income
5. Add expense
6. View monthly summary
7. Show spending chart (by category)
0. Save and Exit
Choose an option: 0
Budget saved. Goodbye.
codio@beatlesaugust-transitbambino:~/workspace$ python3 personal_budget_app.py
Loaded budget from: budget_data.json

=== Personal Budgeting App ===
1. Set monthly income
2. Set savings goal
3. Add / update category
4. Add income
5. Add expense
6. View monthly summary
7. Show spending chart (by category)
0. Save and Exit
Choose an option: 6
Enter year [2025]: 2025
Enter month (1-12) [10]: 10

--- Summary for 2025-10 ---
Total income:  $6000.00
Total expense: $4500.00
Net:           $1500.00

Spending by category:
  Housing         $ 2100.00
  Groceries       $  700.00
  Transportation  $ 1200.00
  Entertainment   $  500.00

Savings goal:  $2000.00
Progress:      75%
```

Flowchart

The following flowchart illustrates the overall program logic.

Figure 5. Personal Budgeting Application Flowchart created in Microsoft Visio.

Personal Budgeting Application Flowchart



The flowchart begins with data loading, moves through the main menu and user input, and loops back to the menu after each operation. The loop continues until the user chooses 0 to exit.

Reflection

Creating this project helped me improve my skills in Python programming and object-oriented design. I learned how to write modular code, validate user input safely, and manage files so that data stays secure and accessible.

Working in Codio gave me experience with a professional development environment where I could test and save my work. Building this project taught me how to create a simple, user-friendly program that also maintains a strong internal structure and protects user information effectively. Overall, this project strengthened my ability to approach programming challenges logically and document my work clearly from design through testing.