Project Report: Personal Finance Tracker Application

Course: CSE
Theme: Finance & Banking
Student Name: Aadeesh Awasthi
Registration Number: 25MIB10045

1. Problem Definition

Financial literacy is a growing concern among students and young professionals. Many individuals struggle to maintain a healthy financial status simply because they lack a consolidated view of their income versus their expenses. Manual tracking using paper or generic spreadsheets is often tedious and error-prone.

Objective:

The objective of this project is to develop a desktop application titled "Finance Tracker." This application aims to simplify personal financial management by allowing users to record income and expenses, view transaction history, and visualize their financial health through dynamic charts.

2. Requirement Analysis

2.1 Functional Requirements

- **Data Entry:** The system must allow users to input financial transactions (Income and Expenses) with a specific category and amount.
- **Input Validation:** The system must reject non-numeric inputs or negative numbers to ensure data integrity.
- **Data Visualization:** The system must generate a visual representation (Pie Chart) of the current financial balance.
- **User Interface:** The application should feature a modern, dark-mode interface with easy navigation between different views.

2.2 Software Requirements

- **Language:** Python 3.x
- **GUI Framework:** customtkinter (for modern UI elements), tkinter (for Treeview tables).
- **Data Visualization:** matplotlib (for generating charts).
- **Operating System:** Windows/MacOS/Linux.

3. Top-Down Design & Modularization

The project follows an Object-Oriented approach, encapsulating the application logic within a main Application class. The code is modularized into three distinct sections:

1. **The View (UI Layer):**
    - __init__: Initializes the main window and grid layout.
    - create_transaction_frame: A reusable method that generates the input forms and tables for both Income and Expense screens.
    - hide_frames: Manages screen transitions.
2. **The Controller (Logic Layer):**
    - add_income / add_expense: Handles the logic of capturing user input and updating the internal data lists.
    - get_amount_from_entry: A helper method specifically for validation and error handling.
3. **The Visualization (Output Layer):**

- update_plot: Integration with Matplotlib to dynamically render the pie chart based on current totals.

4. Algorithm Development

Algorithm: Transaction Entry & Validation

Plaintext

1. START
2. WAIT for User Input (Category, Amount)
3. IF "Add" button is clicked:
   a. CALL get_amount_from_entry(Amount)
   b. IF Input is NOT a number OR Input <= 0:
      i. RAISE ValueError
      ii. SHOW Error Message "Please enter a positive number"
      iii. RETURN
   c. ELSE:
      i. Parse Amount as Float
      ii. Append (Category, Amount) to transaction_list
      iii. Update Total_Sum variable
      iv. CALL update_table() to refresh the UI list
      v. CLEAR input fields
4. END

Algorithm: Dashboard Visualization

Plaintext

1. START
2. User navigates to "Balance" tab
3. CLEAR previous figure
4. CREATE new subplot
5. FETCH total_income and total_expense
6. PLOT Pie Chart with labels ['Income', 'Expense']
7. DRAW canvas on Tkinter widget
8. END

5. Implementation Details

The application was built using **Python**. I chose customtkinter over the standard tkinter library to ensure the application looks modern and supports high-DPI displays.

Key Implementation Highlights:

- **Grid Layout Manager:** Used extensively to ensure the sidebar and main content areas resize correctly.
- **Matplotlib Backend:** Used FigureCanvasTkAgg to embed a scientific plot directly into the Tkinter window, allowing for seamless integration of data analytics.
- **Exception Handling:** A try-except block is implemented in the input field to prevent the program from crashing if a user types text instead of numbers.

6. Conclusion

The "Finance Tracker" project successfully demonstrates the application of Python in solving real-world financial tracking problems. By integrating logic handling with a graphical user interface, the project fulfills the requirements of a functional desktop application.

Future Enhancements:

- Implement a database (SQLite) to save data permanently so it remains after closing the app.
- Add a "Budget Goal" feature to alert the user if expenses exceed a certain limit.