EXPENSE TRACKER

A PROJECT REPORT

Submitted by

DAKSHIN S(113223031036)

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

VELAMMAL ENGINEERING COLLEGE CHENNAI

ANNA UNIVERSITY, CHENNAI - 600 025

APRIL 2024

VELAMMAL ENGINEERING COLLEGE
DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report titled "**EXPENSE TRACKER**" is the bonafide work of **DAKSHIN S** (113223031036) who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. B. MURUGESHWARI, **PROFESSOR & HEAD OF THE DEPARTMENT**	**Mrs. R.AMIRTHAVALLI** **ASSISTANT PROFESSOR** **SUPERVISOR.**
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING VELAMMAL ENGINEERING COLLEGE SURAPET CHENNAI - 600 006	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING VELAMMAL ENGINEERING COLLEGE, SURAPET CHENNAI- 600 066

CERTIFICATE OF EVALUATION

COLLEGE NAME : VELAMMAL ENGINEERING

COLLEGE BRANCH : COMPUTER SCIENCE

ENGINEERING SEMESTER: II

Sl. No	Name of the student who has done the project	Title of the Project	Name of Examiner with designation

This project report, submitted by the above mentioned student, has been reviewed, evaluated, and confirmed as the original work carried out by the student and subsequently assessed.

Submitted for Internal Evaluation held on.......................

EVALUATION CRITERIA	MARK
Code Explanation(Logic & Flow	
Output Demonstration	
Written Documentation	
Presentation Skills	
Time Management	
TOTAL	

Examiner

ABSTRACT

The Expense Tracker is a console-based C program designed to efficiently manage personal and household financial records. The system allows users to add, update, and delete expense and income entries, categorize transactions, and view summaries of total spending and earnings over specific periods. It supports multiple expense categories such as food, transport, utilities, and entertainment, providing a clear overview of financial habits.

The program ensures data persistence by storing all records in a file, allowing users to retain their financial information even after the program is closed. It prevents duplicate entries for the same transaction and provides simple calculations for balances, budgets, and trends. This system enhances financial organization, reduces manual errors, and offers a structured approach to tracking and managing daily expenses for individuals and households.

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who achieved it, without whom it would ever have come into existence. To them we express our words of gratitude.

We give all the glory and thanks to our almighty **GOD** for showering upon, the necessary wisdom and grace for accomplishing this project. We express our gratitude and thanks to **Our Parents** first for giving health and sound mind for completing this project.

First of all, we would like to express our deep gratitude to our beloved and respectable **Chairman Thiru M.V. Muthuramalingam** and our **Chief Executive Officer Thiru M.V.M. Velmurugan** for their kind encouragement.

We express our deep gratitude to **Dr. S. Satish Kumar, Principal,** Velammal Engineering College for his helpful attitude towards this project. We wish to express our sincere thanks and gratitude to **Dr.B.MURUGESHWARI, Professor and Head of the Department**, Department of Computer Science for motivating and encouraging in every part of our project.

We express our sincere gratitude to the Project Coordinator and Project Guide Guide Name, **Assistant Professor Mrs.R.AMIRTHAVALLI,** Department of Computer Science for their invaluable guidance in shaping of this project without them this project would not have been possible.

Our thanks to all other **Faculty and Non-Teaching staff members** of our department for their support and peers for having stood by me and helped me to complete this project.

TABLE OF CONTENTS

CHAPTER NO.	**TITLE**	**PAGE NO.**
1	ABSTRACT ACKNOWLEDGEMENT TABLE OF CONTENTS **INTRODUCTION** PROJECT OUTLINE TOOLS/PLATFORM MOTIVATION PROBLEMS	iv v vi 1 1 1 2 2
2	CHAPTER 2 **LITERATURE SURVEY**	3
3	CHAPTER 3 **SOFTWARES USED**	4
4	CHAPTER 4 **MODULE IMPLEMENTATION**	5
5	CHAPTER 5 **CODING AND OUTPUT**	7
6	CHAPTER 6 **FUTURE ENHANCEMENT**	13

CHAPTER 1
INTRODUCTION

INTRODUCTION:

The Expense Tracker is a computerized solution for managing and monitoring personal and household finances. It replaces traditional manual record-keeping with an efficient, reliable, and user- friendly program developed in the C programming language. The system enables users to record multiple types of transactions, such as income and expenses across different categories, calculates totals and balances, and maintains a persistent record of all financial activities. By automating the process, it reduces manual errors, saves time, and provides a structured and organized approach to tracking daily expenditures and budgeting effectively.

PROJECT OUTLINE:

This project provides the following functionalities:

- Add a new expense or income with amount, category, and date.
- Update or remove a transaction and recalculate balances.
- Display the list of all recorded transactions.
- Show total income, total expenses, and current balance.
- Prevent duplicate entries for the same transaction.
- Store all data in a file for future reference.
- The system uses predefined categories for expenses and incomes, calculates totals, and provides summaries for financial management.

TOOLS/PLATFORM:

Programming Language: C
Compiler: GCC (GNU Compiler Collection) / Code::Blocks / Turbo C (C89 compatible)
Operating System: Windows / Linux
Text Editor/IDE: Code::Blocks, Dev-C++, or any standard C IDE
Storage: Text file (expense_data.txt) for persistent data

MOTIVATION:

Managing personal or household finances is a common challenge, especially with multiple income sources and expense categories. Manual record-keeping often results in miscalculations, missed transactions, and poor visibility of spending habits. The motivation for this project is to create a lightweight, fast, and easy-to-use system that can handle day-to-day financial tracking effectively. By providing an automated, file-based system, the project aims to bring accuracy, organization, and convenience to managing finances without requiring complex software or online tools.

PROBLEMS:

Without anautomated expense tracking system, the following problems are common:

- Difficulty in tracking daily expenses and income manually.
- Errors in calculating totals, balances, and budgets.
- Loss of financial records when using paper-based logs.
- Duplicate or missed entries leading to inaccurate reports.
- Lack of proper summaries and insights for financial planning.

The proposed system addresses these issues by providing real-time tracking, accurate calculations, and data persistence.

CHAPTER 2
LITERATURE SURVEY

With the introduction of personal computers in the 1980s and 1990s, early personal finance software began to emerge. These systems used simple programs or basic databases to record and manage income and expenses, which reduced manual errors and improved tracking. However, they lacked portability, real-time summaries, and automation, limiting their usefulness for dynamic financial management.

For academic projects and small households, file-based expense tracking systems developed in C programming are a practical choice. These systems store financial data in text files, allowing records to persist between program runs. They are easy to implement, cost-effective, and do not require specialized software, making them ideal for personal or small-scale use.

In recent years, advanced finance management tools and mobile apps have become popular, integrating cloud storage, analytics, and budgeting recommendations. Despite their high efficiency, the complexity and subscription costs make them unsuitable for students or budget-constrained users.

The current project adopts this file-based approach to create a lightweight, efficient, and affordable Expense Tracker. It offers persistent storage, automatic calculation of balances and totals, categorization of transactions, and prevents duplicate entries. This combination of features ensures a balance between simplicity, cost, and functionality, making it well-suited for individuals and small households.

Traditional manual bookkeeping methods have been in use for decades, relying on notebooks or spreadsheets to record financial transactions. While these methods have a low setup cost and are simple to operate, they are prone to human errors, difficult to maintain consistently, and often lead to mistakes in budget calculations.

CHAPTER 3
SOFTWARES USED

The Expense Tracker was developed using the C programming language and tested on both Windows and Linux operating systems. For compilation, the GCC (GNU Compiler Collection) was used in Linux environments, while MinGW and Turbo C were utilized for Windows environments. The program was written and executed using Integrated Development Environments (IDEs) such as Code::Blocks, Dev-C++, and Visual Studio Code with the C/C++ extension. Data persistence is achieved by storing all financial records in a local text file named expense_data.txt, ensuring that all details are retained even after the program is closed.

LIBRARIES / MODULES USED

stdio.h This standard input/output library is used for performing essential operations such as printing messages to the console, taking user input, and managing file operations. It handles reading from and writing to the data file that stores vehicle parking information.

string.h This library is used for string manipulation, including comparing and copying vehicle numbers and types. It allows the program to verify duplicate entries and store text-based details efficiently.

time.h The time library manages all time-related operations in the system. It records the entry time of vehicles, calculates the duration of parking based on exit time, and formats the time for user-friendly display.

CHAPTER 4
MODULE IMPLEMENTATION

1. **Data Storage Module** This module is responsible for storing and retrieving financial records. It uses a text file named expense_data.txt to maintain persistent records of transactions, including the amount, category, date, and type (income or expense). The saveData() function writes the current transaction data to the file, while the loadData() function retrieves the information when the program starts. This ensures that data is not lost even if the application is closed.

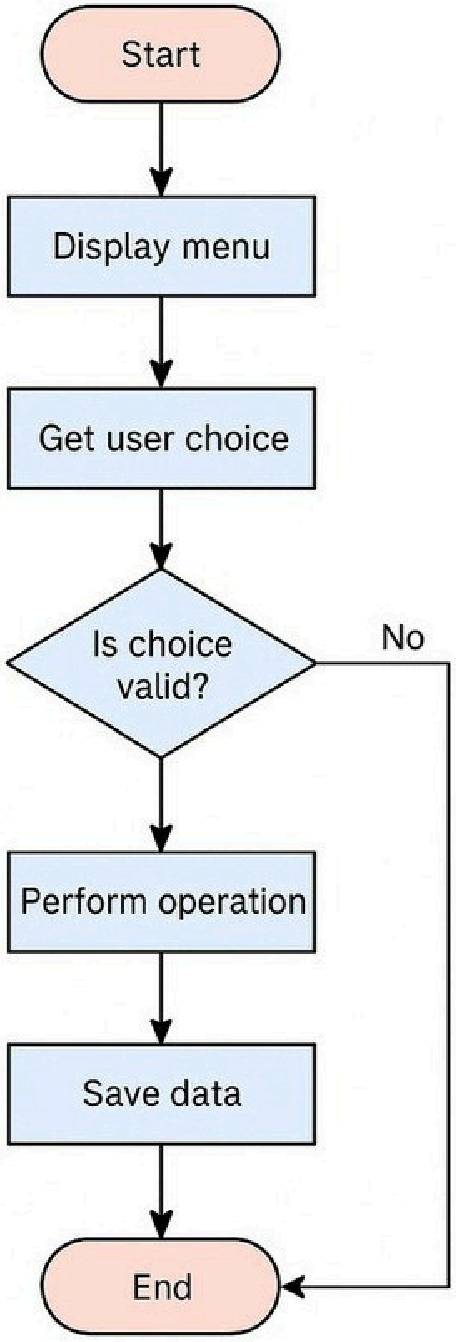
2. **Add Transaction Module** The addTransaction() function handles the process of recording a new expense or income. It takes input for the amount, category, type, and date, checks for duplicate entries to maintain data integrity, and stores the details in memory. If the input is invalid, it displays an appropriate error message.

3. **Remove Transaction Module** The removeTransaction() function manages the deletion of a transaction. It searches for the entered transaction based on amount, category, or date, removes it from memory, and updates totals accordingly. This ensures that the records and balance remain accurate.

4. **Display Module** The viewTransactions() function lists all recorded transactions along with their details such as category, type, amount, and date. This helps the user quickly review their financial records. The viewSummary() function displays total income, total expenses, and current balance, allowing easy monitoring of finances.

5. **Calculation Module** The calculateTotals() function computes totals for income, expenses, and balances. The system can also generate summaries based on categories or date ranges. This modular approach allows easy updates to calculations without affecting other parts of the program.

WORK FLOW:


Start
Display menu
Get user choice
Is choice valid?
No
Perform operation
Save data
End

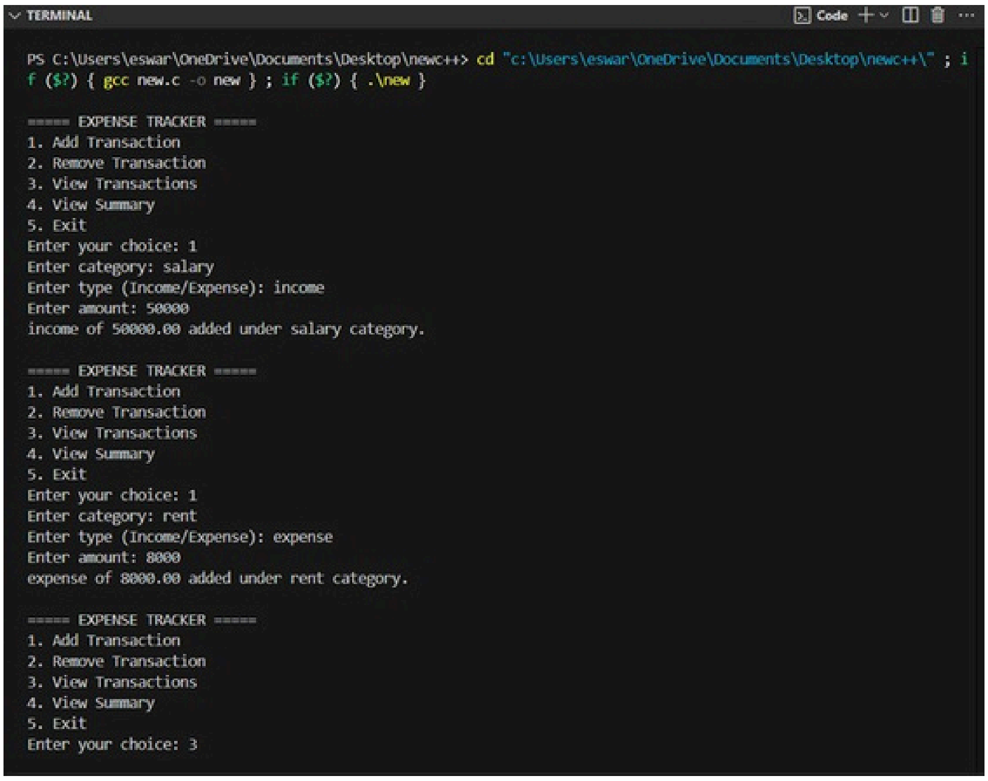
CHAPTER 5
CODING AND OUTPUT

```
#include <stdio.h>
#include <string.h>
#include <time.h>

#define MAX_TRANSACTIONS 100
#define DATA_FILE "expense_data.txt"

typedef struct {
   char category[20];
 char  type[10];  //  "Income"  or  "Expense"
double    amount;        time_t    date;    }
Transaction;


Transaction transactions[MAX_TRANSACTIONS];
int transaction_count = 0;
double total_income = 0, total_expense = 0;

void saveData() {
FILE *fp = fopen(DATA_FILE, "w");
   if (!fp) return;
 fprintf(fp, "%d %.2lf %.2lf\n", transaction_count, total_income, total_expense);
   for (int i = 0; i < transaction_count; i++) {
      fprintf(fp, "%s %s %.2lf %ld\n", transactions[i].category, transactions[i].type,
            transactions[i].amount, transactions[i].date);
   }
   fclose(fp);
}
```

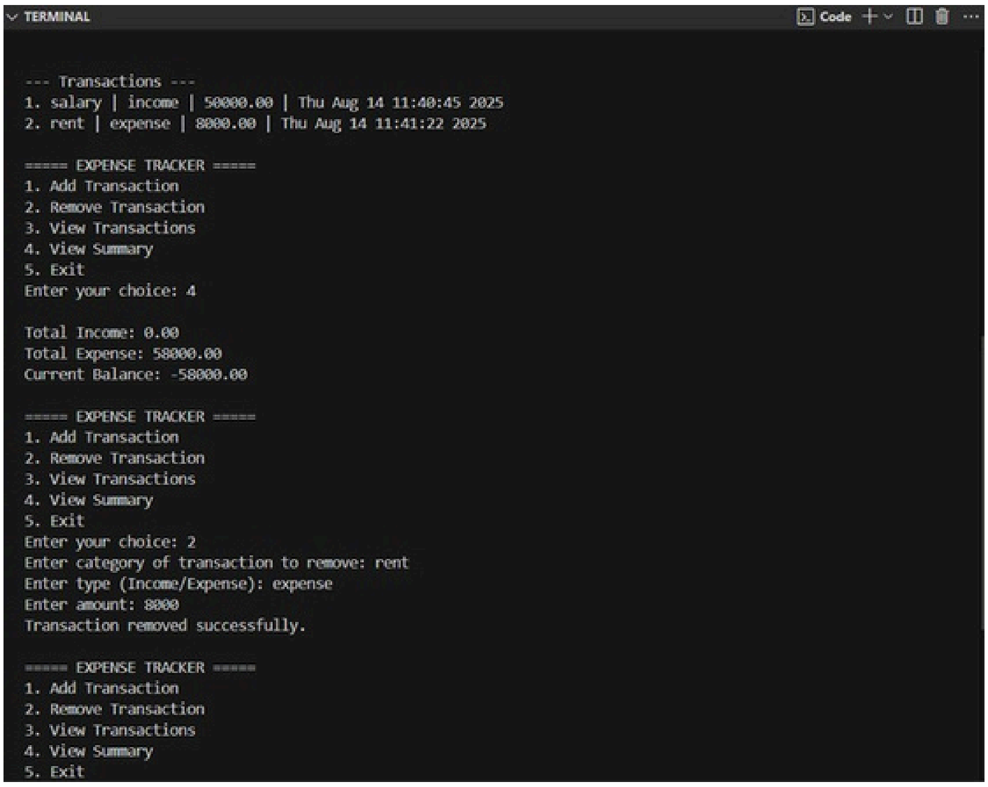
```
void addTransaction() {
if (transaction_count >= MAX_TRANSACTIONS) {
        printf("Transaction limit reached!\n");
        return;
    } char category[20],
    type[10];
 double amount;
printf("Enter category: "); scanf("%s", category);
printf("Enter type (Income/Expense): "); scanf("%s", type);
printf("Enter amount: "); scanf("%lf", &amount);

    strcpy(transactions[transaction_count].category, category);
    strcpy(transactions[transaction_count].type, type);
 transactions[transaction_count].amount = amount;
transactions[transaction_count].date = time(NULL);

 if (strcmp(type, "Income") == 0) total_income += amount;
else total_expense += amount;

    transaction_count++;
 printf("%s of %.2lf added under %s category.\n", type, amount, category);
saveData();
}

void removeTransaction() {
    char category[20], type[10];
 double amount;
printf("Enter category of transaction to remove: "); scanf("%s", category);
printf("Enter type (Income/Expense): "); scanf("%s", type);
printf("Enter amount: "); scanf("%lf", &amount);

    for (int i = 0; i < transaction_count; i++) {
        if (strcmp(transactions[i].category, category) == 0 &&
            strcmp(transactions[i].type, type) == 0 &&
            transactions[i].amount == amount) {

            if (strcmp(type, "Income") == 0) total_income -= amount;
            else total_expense -= amount;

            for (int j = i; j < transaction_count - 1; j++) {
                transactions[j] = transactions[j + 1];
            }
transaction_count--;
          printf("Transaction removed successfully.\n");
          saveData();
          return;
       }
   } printf("Transaction not
   found!\n");
}

void viewTransactions() {
   printf("\n--- Transactions ---\n");
   if (transaction_count == 0) {
      printf("No transactions recorded.\n");
      return;
   } for (int i = 0; i < transaction_count; i++)
   {
      printf("%d. %s | %s | %.2lf | %s", i + 1, transactions[i].category,
           transactions[i].type, transactions[i].amount,
           ctime(&transactions[i].date));
   }
}

void viewSummary() {
printf("\nTotal Income: %.2lf\n", total_income);
printf("Total Expense: %.2lf\n", total_expense);
printf("Current Balance: %.2lf\n", total_income - total_expense);
}

intmain() {
   loadData();
   int   choice;
   while (1) {
      printf("\n===== EXPENSE TRACKER =====\n");
      printf("1. Add Transaction\n");
      printf("2. Remove Transaction\n");
      printf("3. View Transactions\n");
```

```
 printf("4. View Summary\n");
printf("5. Exit\n");
printf("Enter your choice: ");
 scanf("%d", &choice);
 switch (choice) {
case 1: addTransaction(); break;
case 2: removeTransaction(); break;
case 3: viewTransactions(); break;
case 4: viewSummary(); break;
case 5:
 printf("Exiting... Goodbye!\n");
 saveData();
 return 0;
 default:
 printf("Invalid choice! Try again.\n");
 }
}
}
```

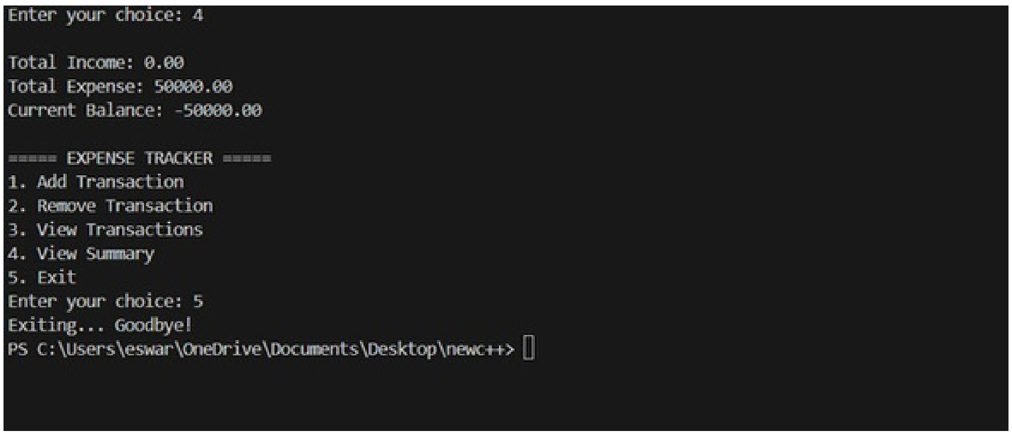
OUTPUT :



```
PS C:\Users\eswar\OneDrive\Documents\Desktop\newc++> cd "c:\Users\eswar\OneDrive\Documents\Desktop\newc++\" ; i
f ($?) { gcc new.c -o new } ; if ($?) { .\new }

===== EXPENSE TRACKER =====
1. Add Transaction
2. Remove Transaction
3. View Transactions
4. View Summary
5. Exit
Enter your choice: 1
Enter category: salary
Enter type (Income/Expense): income
Enter amount: 50000
income of 50000.00 added under salary category.

===== EXPENSE TRACKER =====
1. Add Transaction
2. Remove Transaction
3. View Transactions
4. View Summary
5. Exit
Enter your choice: 1
Enter category: rent
Enter type (Income/Expense): expense
Enter amount: 8000
expense of 8000.00 added under rent category.

===== EXPENSE TRACKER =====
1. Add Transaction
2. Remove Transaction
3. View Transactions
4. View Summary
5. Exit
Enter your choice: 3
```



```
--- Transactions ---
1. salary | income | 50000.00 | Thu Aug 14 11:40:45 2025
2. rent | expense | 8000.00 | Thu Aug 14 11:41:22 2025

===== EXPENSE TRACKER =====
1. Add Transaction
2. Remove Transaction
3. View Transactions
4. View Summary
5. Exit
Enter your choice: 4

Total Income: 0.00
Total Expense: 58000.00
Current Balance: -58000.00

===== EXPENSE TRACKER =====
1. Add Transaction
2. Remove Transaction
3. View Transactions
4. View Summary
5. Exit
Enter your choice: 2
Enter category of transaction to remove: rent
Enter type (Income/Expense): expense
Enter amount: 8000
Transaction removed successfully.

===== EXPENSE TRACKER =====
1. Add Transaction
2. Remove Transaction
3. View Transactions
4. View Summary
5. Exit
```



```
Enter your choice: 4

Total Income: 0.00
Total Expense: 50000.00
Current Balance: -50000.00

===== EXPENSE TRACKER =====
1. Add Transaction
2. Remove Transaction
3. View Transactions
4. View Summary
5. Exit
Enter your choice: 5
Exiting... Goodbye!
PS C:\Users\eswar\OneDrive\Documents\Desktop\newc++>
```

CHAPTER 6
FUTURE ENHANCEMENT

The current Expense Tracker is a functional and lightweight solution designed for individuals and small households. However, several enhancements can be implemented to improve efficiency, usability, and scalability in the future.

One possible enhancement is the integration of a Graphical User Interface (GUI) using libraries such as GTK or Qt for C, which would make the system more user-friendly and visually appealing compared to the existing text-based interface. Another improvement could involve database integration with MySQL or SQLite to replace the text file storage, allowing for faster data retrieval, better scalability, and advanced query capabilities. The system could also be enhanced with data visualization features, such as graphs and charts, to provide users with better insights into spending habits and trends.

A mobile application or web interface could be developed to allow users to add, view, and manage transactions on the go. For enhanced financial management, features like budget alerts, recurring transactions, and category-wise spending limits could be implemented. Additionally, integration with banking APIs or payment gateways could automate the recording of income and expenses directly from bank accounts. Finally, support for multi-user accounts and cloud-based synchronization could allow multiple users to track finances collaboratively. These enhancements would transform the system into a fully automated, smart personal finance management solution capable of meeting modern financial tracking needs.

REFERENCE :

1.Brian W. Kernighan, Dennis M. Ritchie, The C Programming Language, 2nd Edition,Prentice Hall, 1988.
2.W3Schools, C Language Basics and String Functions, [Accessed: January 29, 2024], https://www.w3schools.com/c/c_strings.php
3.Microsoft Documentation, time.h Functions Reference, [Accessed: January 29, 2024], https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/ctime-ctime-t?view=msvc-170
4.TutorialsPoint, C Programming Language Tutorial, [Accessed: January 29, 2024], https://www.tutorialspoint.com/cprogramming/index.htm
5.GeeksforGeeks, File Handling in C, [Accessed: January 29, 2024], https://www.geeksforgeeks.org/file-handling-c-classes/