

CIA 3 : PART-B

TOPIC: Personal Expense tracker

Submission Date: 8 September 2024

Under the guidance of

Dr. SHONEY SEBASTIAN

Submitted by

Vaibhav Tomar

2447157

Team Members:

1. Debin Robert - 2447117
2. Abhinav S- 2447102
3. Vaibhav Tomar - 2447157

**Personal Expense Tracker**

The project titled **Personal Expense Tracker** is designed to help users manage their daily expenses effectively. The system provides features for **user registration**, **secure login**, and **transaction management**. Users can log their daily expenses, view detailed transaction history, and analyse their spending across different categories and time periods.

The system includes features for calculating **weekly and monthly expense percentages** by category, providing insights into how much users spend on different aspects of their lives.

**Features Used in the Program**

1. **Preprocessor Directives**  
   Preprocessor directives allow the inclusion of necessary libraries.
   * Usage: #include <stdio.h> for input/output functions and #include <string.h> for string manipulations.
   * Example:

#include <stdio.h>

1. **Structures**  
   Structures are used to store related data such as user details and transaction records.
   * Usage: struct Transaction stores transaction information like date, amount, category, and account type.
   * Example:

struct Transaction {

int day;

int month;

int year;

float amount;

char categoryCode;

char accountCode;

char note[100];

};

1. **File Handling**  
   File handling functions such as fopen(), fwrite(), and fread() are used to save and retrieve transactions and user credentials from binary files.
   * Usage: Transaction data is stored in a binary file for each user.
   * Example:

FILE \*filePtr = fopen("users.bin", "rb");

fwrite(&newUser, sizeof(struct User), 1, filePtr);

1. **Loops**  
   **While** and **For** loops are used for iterating through the menu options and handling transaction data.
   * Usage: Loop through transactions to display or analyse data.
   * Example:

while (fread(&t, sizeof(struct Transaction), 1, filePtr)) { // Process transactions }

1. **Conditionals**  
   Conditional statements (if, else, switch) control the program's flow based on user input and data.
   * Usage: Check user credentials, validate inputs, or display menus.
   * Example:

if (strcmp(currentUser.username, username) == 0) {

// Username matches

}

1. **Functions**  
   Functions organize the program into manageable sections. Key functions include user login, transaction entry, and analysis of expense data.
   * Usage: Modular functions handle different tasks, such as adding transactions, showing percentages, and validating input.
   * Example:

void addTransaction(char \*filename) {

// Function to add transaction

}

1. **String Functions**  
   String functions like gets(), strcmp(), and strcpy() are used to manage and manipulate strings.
   * Usage: Handling user input for usernames, passwords, and other text fields.
   * Example:

gets(username); // Reads username input

1. **Error Handling**  
   The program checks for valid input (e.g., date, amount, and category) and prompts the user to correct any errors.
   * Usage: Ensures the user enters valid data, especially for dates and categories.
   * Example:

if (t.month < 1 || t.month > 12) {

printf("Invalid month, please re-enter.\n");

}

1. **I/O Functions**  
   Input/output functions (printf, scanf, gets) allow interaction with the user.
   * Usage: Prompts the user to enter details for transactions and display output like transaction history.

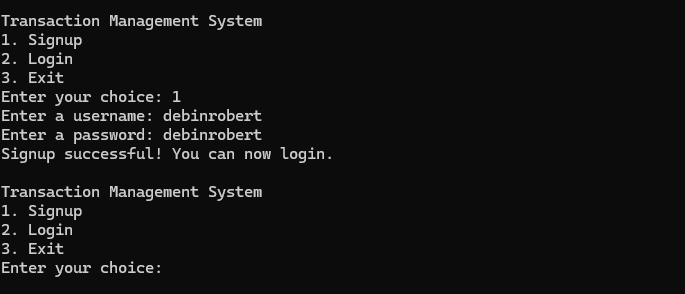
Example: printf("Enter Amount: "); scanf("%f", &t.amount);

**Functionalities of the Program**

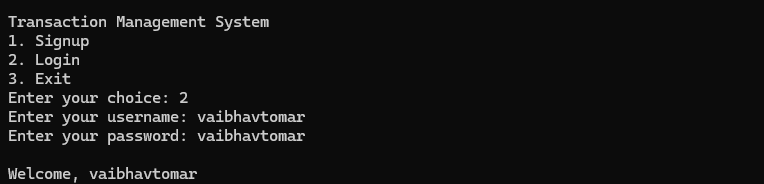
1. **User Functionalities:**
   * **Signup:** Users register with a username and password. The data is saved in a binary file.
   * **Login:** Users log in using their credentials. If correct, they can manage their transactions.
   * **Add Transaction:** Users can add transactions by entering the date, amount, category, account type, and a note.
   * **View Transactions:** Users can view all transactions, with details like date, amount, category, and account type.
   * **View Transactions by Month/Year:** Filter transactions by a specific month and year.
   * **Show Weekly Expense Percentages:** Break down expenses for a specific month by week and show the percentage spent each week.
   * **Show Monthly Expense Percentages:** Break down expenses by category for a specific month and show the percentage spent in each category.
2. **General Functionalities:**
   * **Data Validation:** Ensures correct input for dates, amounts, and categories. Invalid inputs prompt the user to re-enter.
   * **File Operations:** All transaction data and user details are stored and retrieved from binary files.
   * **Menu Navigation:** Users can navigate through the system using a menu that offers options for adding transactions, viewing history, and analysing expenses.
3. **Exit:**
   * Users can log out or exit the system using the appropriate menu option.

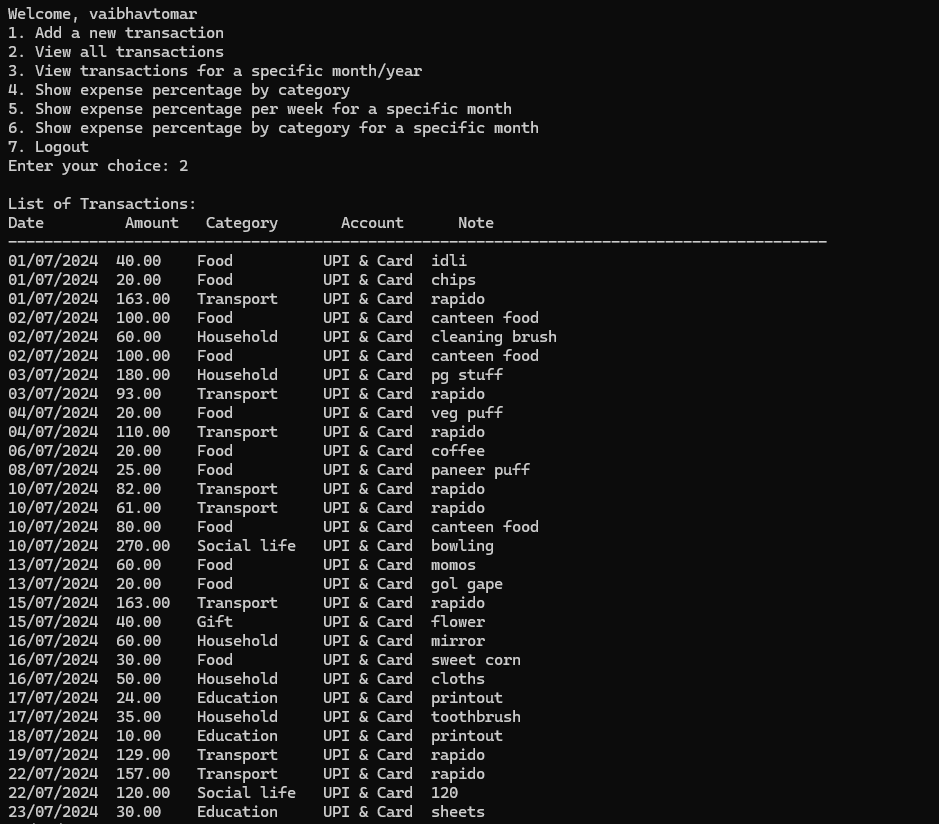
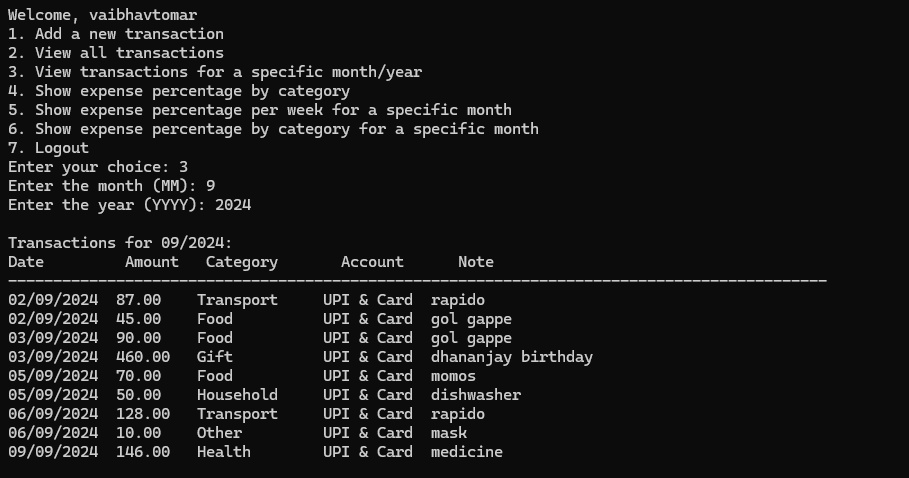
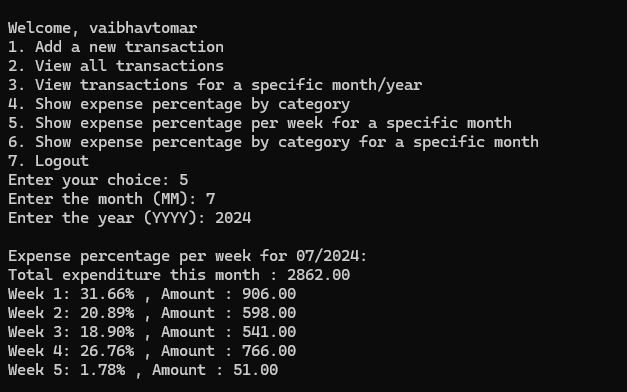
**Sample Outputs**

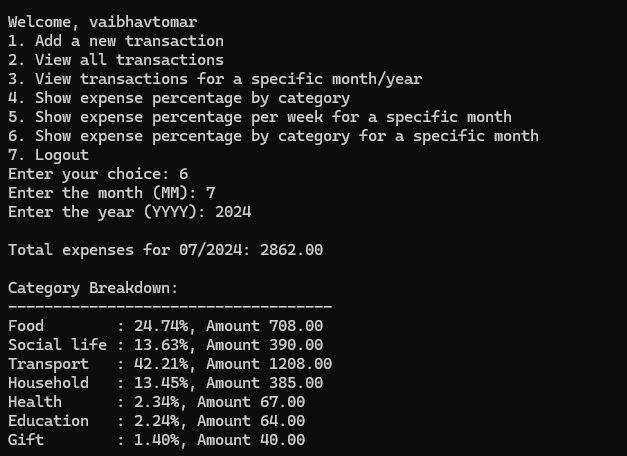
* Signup

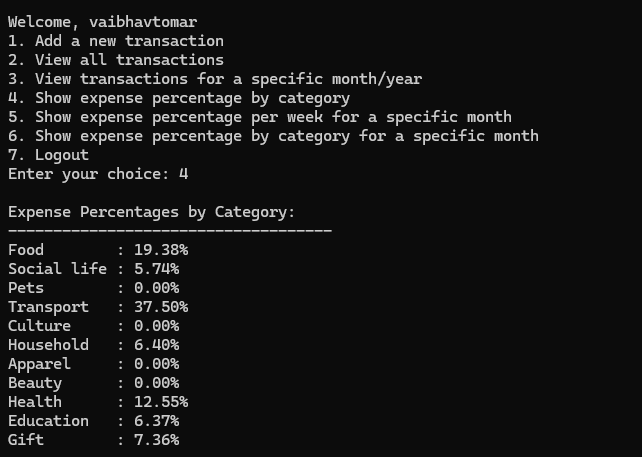


* Login



* Add transaction 
* View all transactions 
* View transactions by month 
* View weekly breakdown of expenditure percentages 
* View category-wise breakdown of transactions by month



* View category-wise breakdown of lifetime transactions 

**Future Scope**

The future scope of this project includes:

• Implementing more categories for better classification of transactions.

• Adding functionality for users to delete or update transactions.

• Introducing data visualization for expense analysis using graphs.

• Multi-user support with secure password encryption.