C PROGRAMMING PROJECT REPORT

- **Project Title:** Daily Expense Tracker
- **Course:** Programming in C (B.Tech 1st Semester)
- **Course Code:** CSEG1041_5
- **Submitted By:** Anviksha Maurya
- **SAP ID:** 590026976
- **Submitted To:** Mohsin F. Dar
- **Date of Submission:** 02/12/2025

Problem Statement:

People often find it difficult to keep track of their day-to-day spending. Manual recording of expenses can be time-consuming and prone to mistakes.

This project aims to solve this problem by creating a simple C program that allows users to record daily expenses, view them anytime, calculate the total amount spent, and save all data to a text file for future reference.

Objective of the Project:

- To develop a simple and user-friendly expense tracking system in C.
- To store multiple expenses with name, amount, and date.
- To calculate the total money spent.
- To save all expenses in a text file for later use.

Software / Tools Used:

Compiler: GCC compiler, VS Code

Operating System: Windows

Algorithm:

main.c –

1. Start the program.
2. Declare an array expenses[MAX] to store expense records.
3. Set count = 0.
4. Set choice = 0.
5. Repeat until choice == 5:
   a. Display the menu.
   b. Take user input for choice.
   c. Use switch-case:
      - If choice = 1 → call addExpense().
      - If choice = 2 → call viewExpenses().
      - If choice = 3 → call totalSpent().
      - If choice = 4 → call saveToFile().
      - If choice = 5 → print exit message.
      - Otherwise → print invalid choice message.
6. End the program.

functions.c –

1. Start the program.
2. Create a function addExpense()

```
    a. Ask user to enter expense name.
    b. Read the name with spaces.
    c. Ask the user to enter amount.
    d. Read amount.
    e. Ask user to enter date.
    f. Read date.
    g. Store these values in expenses[count].
    h. Increment count by 1.
    i. Print success message.
3. Create a function viewExpenses()
    a. Check if count == 0:
       If yes, print "No Expenses" and exit
       function.
    b. Print detail headings.
    c. Loop from i = 0 to count – 1:
       Print each expenses's details.
4. Create a function totalSpent()
    a. Set sum = 0.
    b. Loop from i = 0 to count – 1:
       Add each expense amount to sum.
    c. Print the total sum.
5. Create a function saveToFile()
    a. Open file "expenses.txt" in append mode.
```

   b. If file fails to open, print error and exit function.
   c. Loop from i = 0 to count – 1:
      Write each expense into the file.
   d. Close the file
   e. Print success message.

6. End the program.

functions.h –

- Define the maximum number of expenses (MAX).
- Declare the structure Expense with name, amount, and date.
- Declare all function prototypes so that other files can use them.
- Prevent multiple inclusions using header guards.

Pseudocode:

main.c –

```
START

Create array expenses[MAX]

Set count = 0
Set choice = 0
WHILE choice != 5
    Display menu options
    Read choice
    SWITCH choice
        CASE 1:
            addExpense(expenses, count)
        CASE 2:
            viewExpenses(expenses, count)
        CASE 3:
            totalSpent(expenses, count)
        CASE 4:
            saveToFile(expenses, count)
        CASE 5:
            Print exit message
        DEFAULT:
            Print "Invalid Choice"
END WHILE
END
```

functions.c –

```
START
FUNCTION addExpense(expenses, count)
    Print "Enter Name"
    Read name into expenses[count].name
    Print "Enter Amount"
    Read amount into expenses[count].amount
    Print "Enter Date"
    Read date into expenses[count].date
    count = count + 1
    Print "Expense Added Successfully"
END FUNCTION
FUNCTION viewExpenses(expenses, count)
    IF count == 0 THEN
        Print "No Expenses Recorded"
        RETURN
    ENDIF
    Print "S.No  Name   Amount   Date"
    FOR i = 0 TO count - 1
```

```
        Print i+1, expenses[i].name,
expenses[i].amount, expenses[i].date
    END FOR
END FUNCTION
FUNCTION totalSpent(expenses, count)
    sum = 0
    FOR i = 0 TO count - 1
        sum = sum + expenses[i].amount
    END FOR
    Print "Total Money Spent =", sum
END FUNCTION
FUNCTION saveToFile(expenses, count)
    Open file "expenses.txt" in append mode
    IF file not opened THEN
        Print "Error Opening File"
        RETURN
    ENDIF
    FOR i = 0 TO count – 1
```

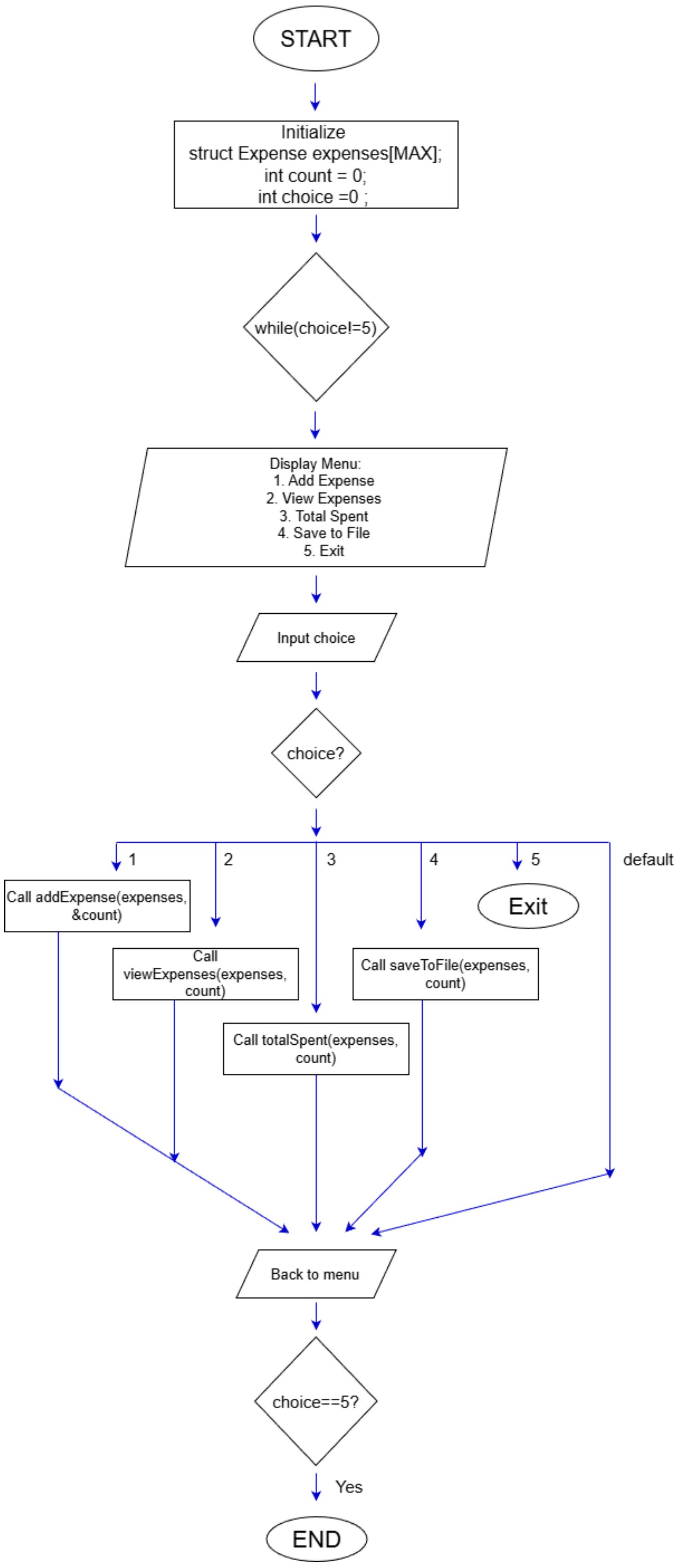
```
        Write expenses[i].name, expenses[i].amount,
expenses[i].date to file
    END FOR
    Close file
    Print "Expenses Saved Successfully"
END FUNCTION
END
```

functions.h –

```
DEFINE MAX = 100
DEFINE STRUCT Expense
    name
    amount
    date
END STRUCT
DECLARE addExpense()
DECLARE viewExpenses()
DECLARE totalSpent()
DECLARE saveToFile()
```

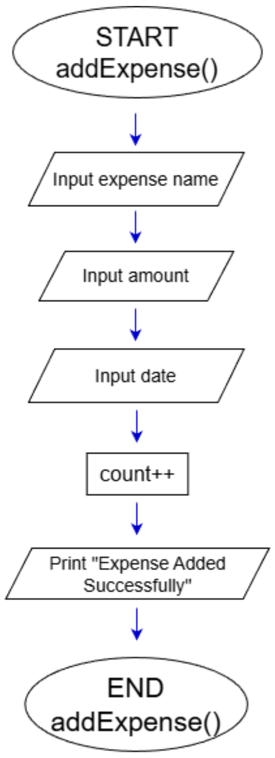
Flowchart:

main.c –

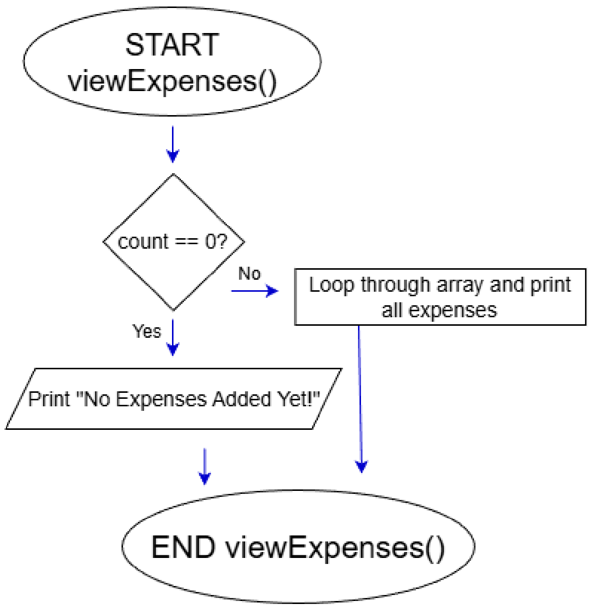


functions.c -

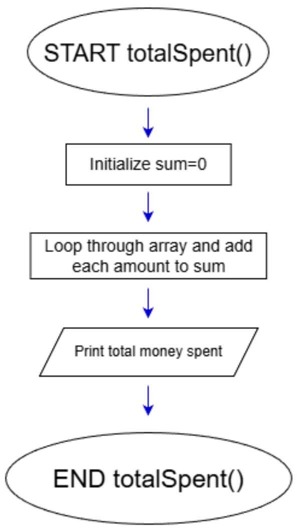
- **addExpense():**



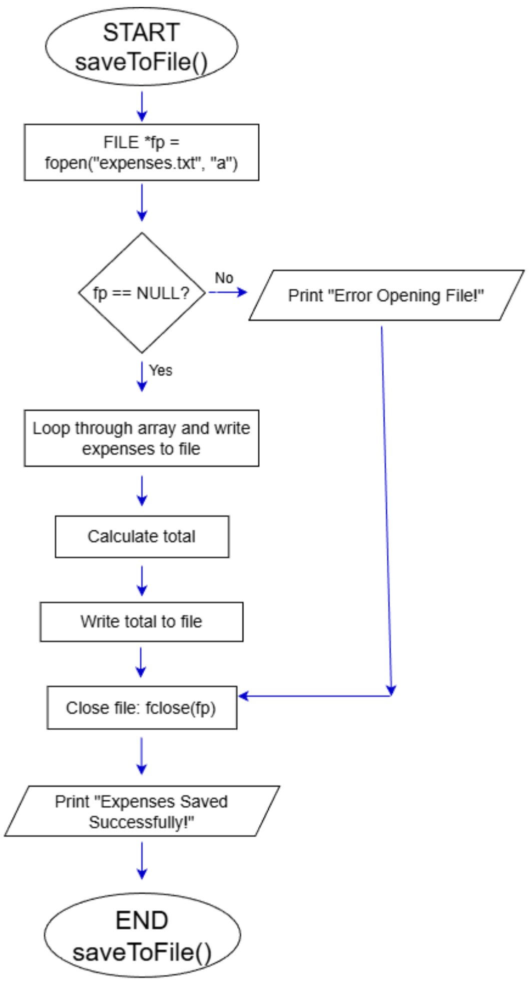
- **viewExpenses():**


START viewExpenses()
count == 0?
No
Loop through array and print all expenses
Yes
Print "No Expenses Added Yet!"
END viewExpenses()

- **totalSpent():**

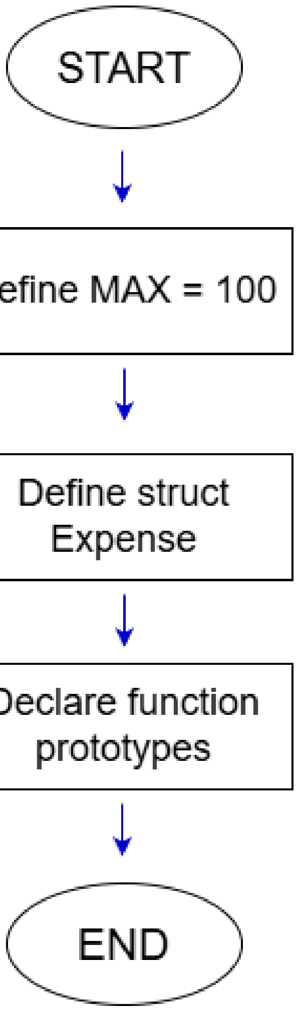


- **saveToFile():**

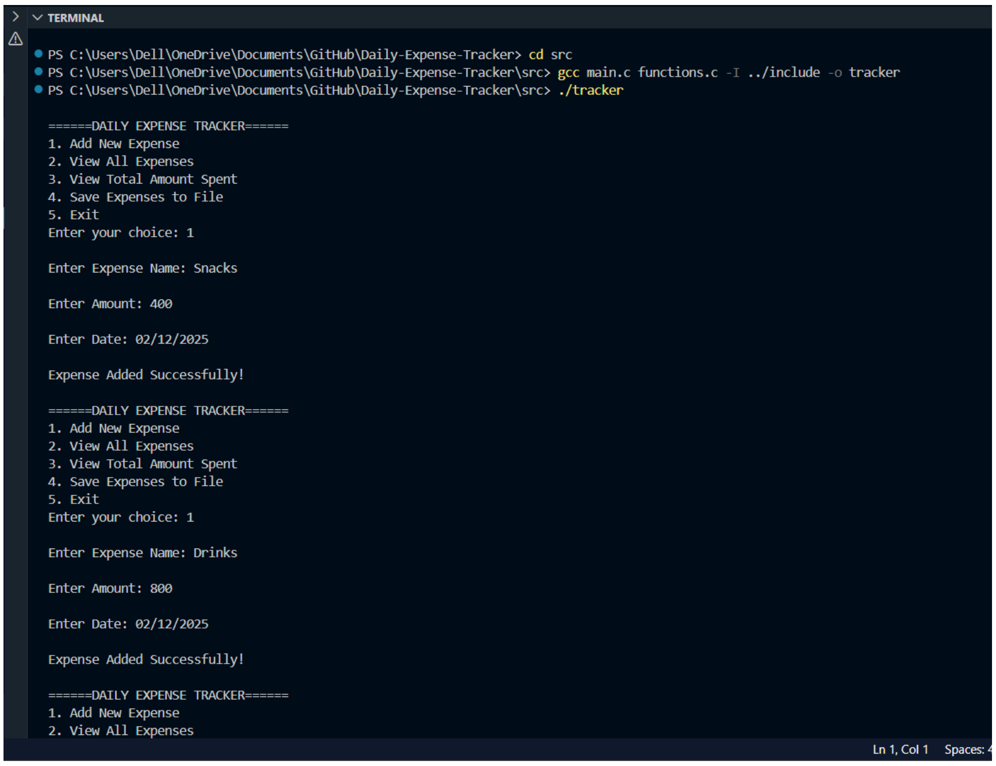


START saveToFile()
FILE *fp = fopen("expenses.txt", "a")
fp == NULL?
No
Print "Error Opening File!"
Yes
Loop through array and write expenses to file
Calculate total
Write total to file
Close file: fclose(fp)
Print "Expenses Saved Successfully!"
END saveToFile()

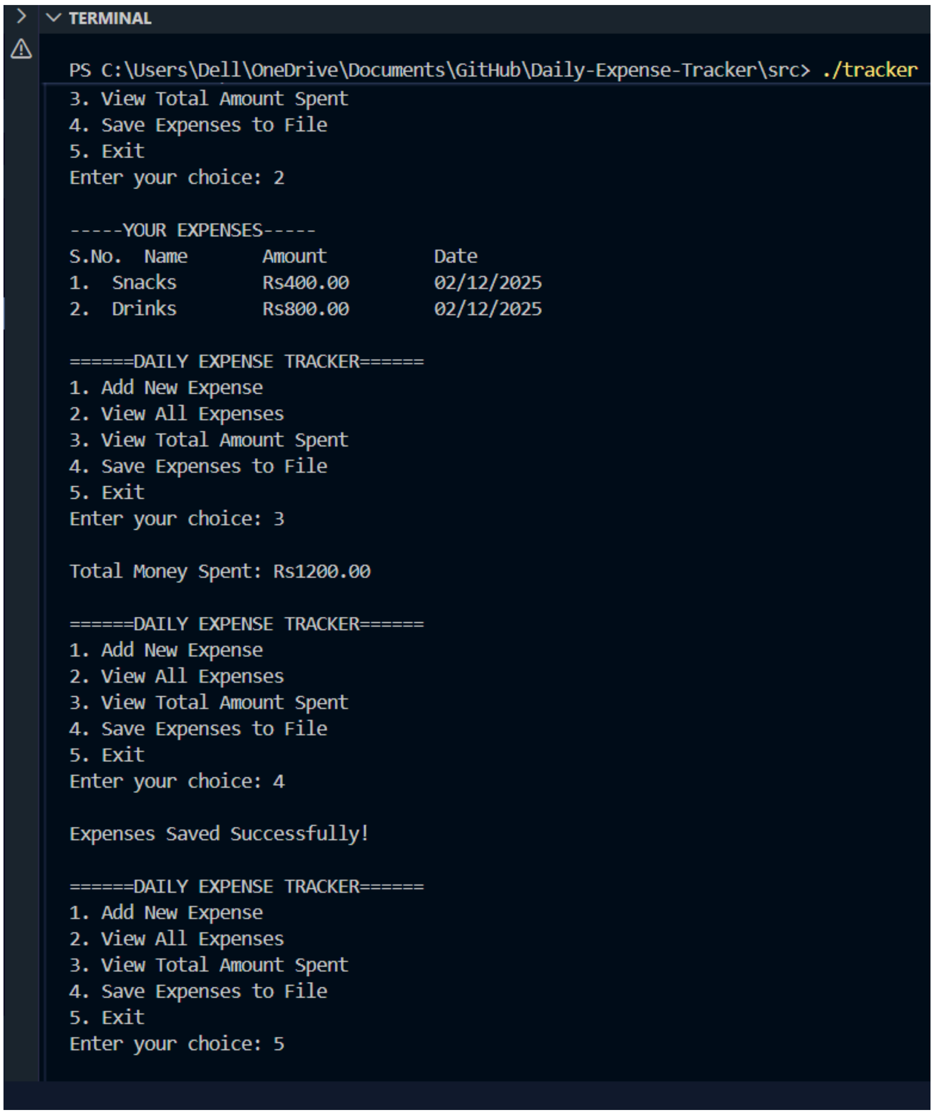
functions.h -


START
Define MAX = 100
Define struct Expense
Declare function prototypes
END

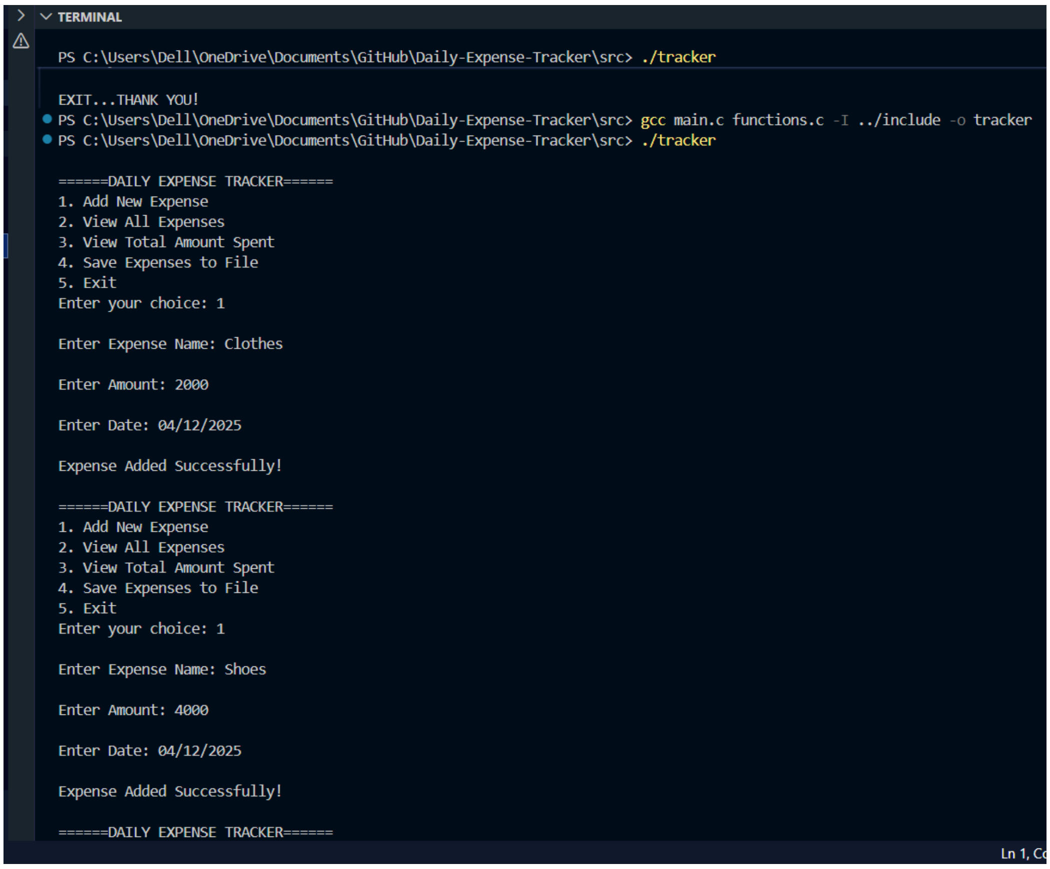
Output Screenshots:



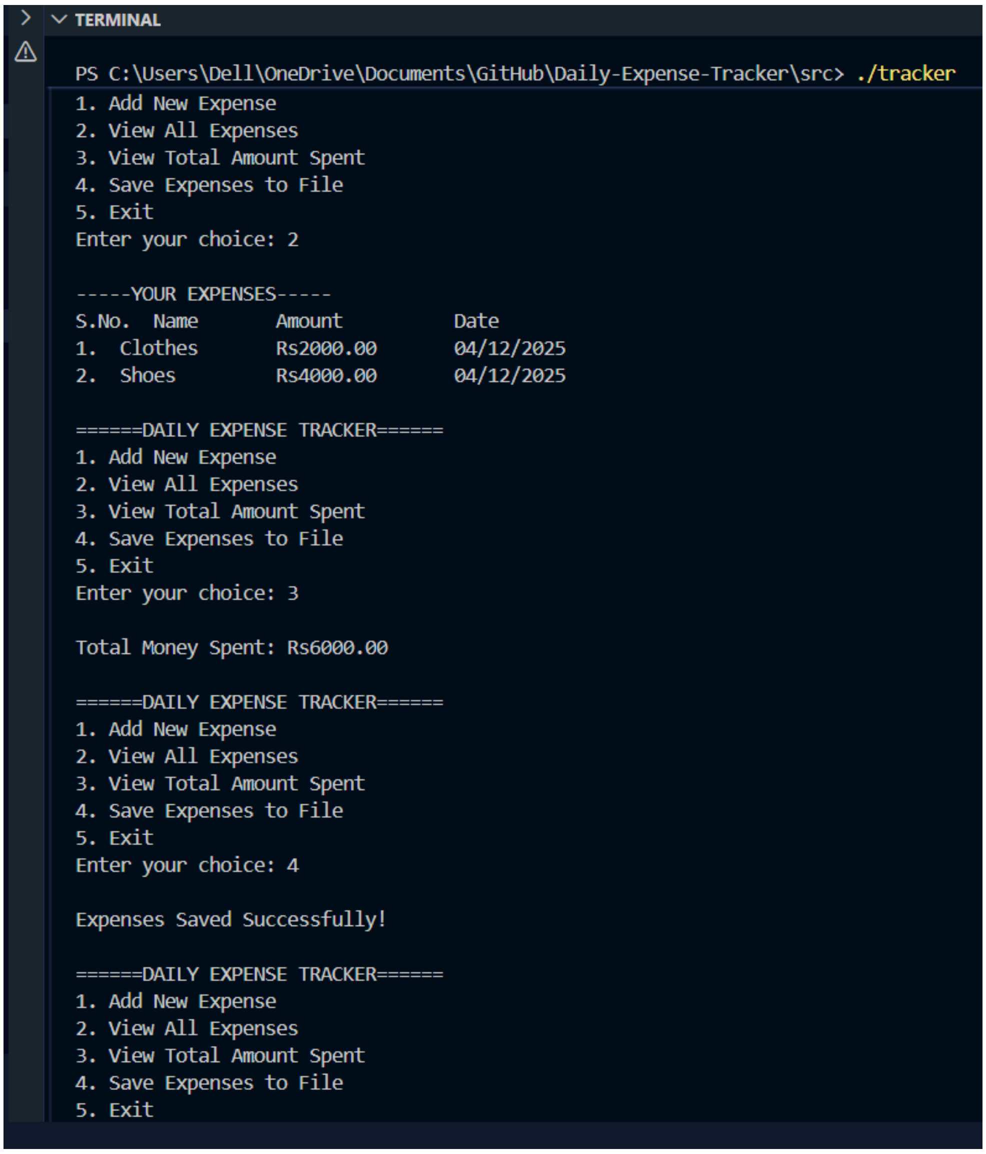
```
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker> cd src
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker\src> gcc main.c functions.c -I ../include -o tracker
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker\src> ./tracker

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 1

Enter Expense Name: Snacks

Enter Amount: 400

Enter Date: 02/12/2025

Expense Added Successfully!

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 1

Enter Expense Name: Drinks

Enter Amount: 800

Enter Date: 02/12/2025

Expense Added Successfully!

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
```



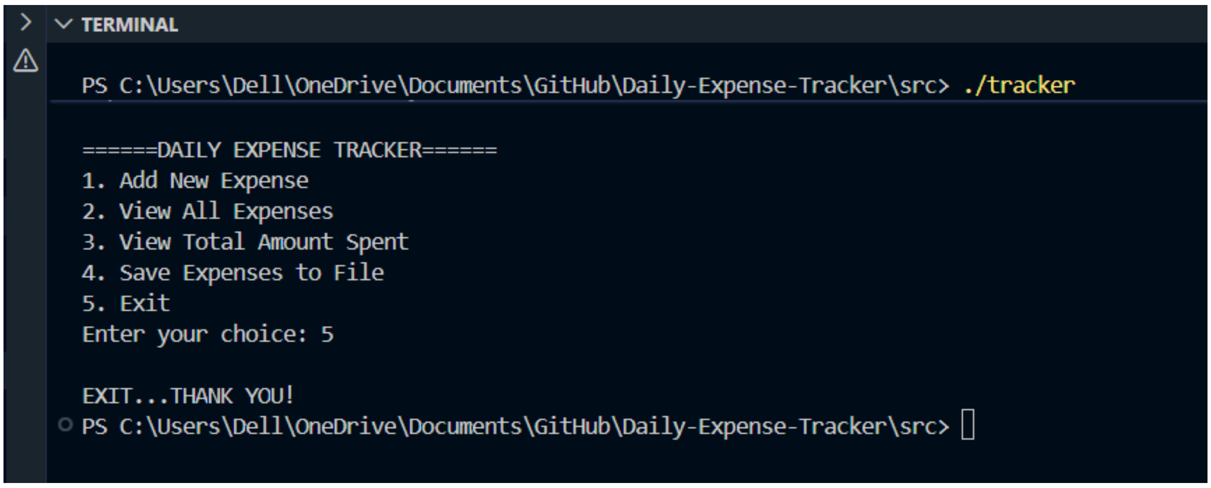
```
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker\src> ./tracker
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 2

-----YOUR EXPENSES-----
S.No.  Name       Amount        Date
1.   Snacks      Rs400.00      02/12/2025
2.   Drinks      Rs800.00      02/12/2025

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 3

Total Money Spent: Rs1200.00

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 4

Expenses Saved Successfully!

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 5
```



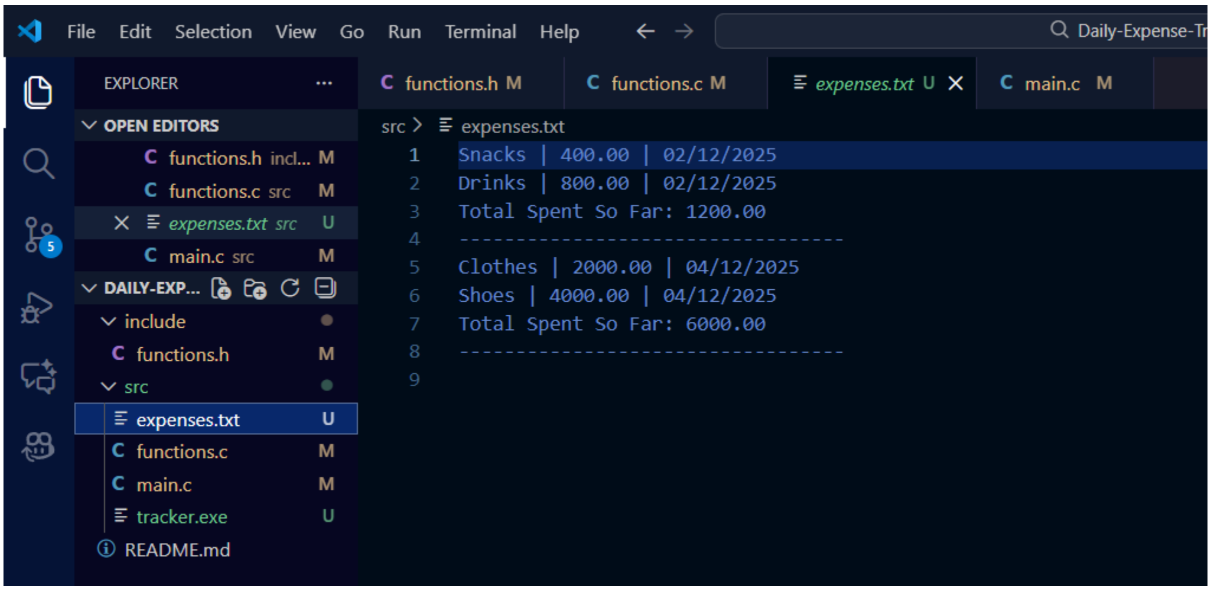
```
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker\src> ./tracker

EXIT...THANK YOU!
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker\src> gcc main.c functions.c -I ../include -o tracker
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker\src> ./tracker

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 1

Enter Expense Name: Clothes

Enter Amount: 2000

Enter Date: 04/12/2025

Expense Added Successfully!

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 1

Enter Expense Name: Shoes

Enter Amount: 4000

Enter Date: 04/12/2025

Expense Added Successfully!

======DAILY EXPENSE TRACKER======
```



```
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker\src> ./tracker
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 2

-----YOUR EXPENSES-----
S.No.  Name       Amount          Date
1.   Clothes      Rs2000.00       04/12/2025
2.   Shoes        Rs4000.00       04/12/2025

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 3

Total Money Spent: Rs6000.00

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 4

Expenses Saved Successfully!

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
```



```
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker\src> ./tracker

======DAILY EXPENSE TRACKER======
1. Add New Expense
2. View All Expenses
3. View Total Amount Spent
4. Save Expenses to File
5. Exit
Enter your choice: 5

EXIT...THANK YOU!
PS C:\Users\Dell\OneDrive\Documents\GitHub\Daily-Expense-Tracker\src>
```



```
Snacks | 400.00 | 02/12/2025
Drinks | 800.00 | 02/12/2025
Total Spent So Far: 1200.00
----------------------------------
Clothes | 2000.00 | 04/12/2025
Shoes | 4000.00 | 04/12/2025
Total Spent So Far: 6000.00
----------------------------------
```

Source Code:

main.c –

```
#include <stdio.h>
#include "functions.h"
int main()
{
   struct Expense expenses[MAX]; //Array to
store all expenses
   int count = 0; //Current number of expenses
   int choice = 0; //User menu choice
   //Controled while loop which runs until user
chooses 5
   while (choice != 5)
   {
      //Display menu
      printf("\n======DAILY EXPENSE
TRACKER======\n");
      printf("1. Add New Expense\n");
      printf("2. View All Expenses\n");
      printf("3. View Total Amount Spent\n");
      printf("4. Save Expenses to File\n");
      printf("5. Exit\n");
      printf("Enter your choice: ");
      scanf(" %d", &choice);
      //Handle user choice using switch
```

```
    switch (choice)
    {
       case 1:
       addExpense(expenses, &count); //Call
function to add expense
       break;
       case 2:
       viewExpenses(expenses, count); //Call
function to view expenses
       break;
       case 3:
       totalSpent(expenses, count); //Call
function to calculate total
       break;
       case 4:
       saveToFile(expenses, count); //Call
function to save expenses
       break;
       case 5:
            printf("\nEXIT...THANK YOU!\n"); //Exit
message
            break;
            default:
            printf("\nInvalid Choice! Please Try
Again.\n"); //Invalid input
        }
    }
    return 0;
}
```

functions.c –

```
#include <stdio.h>
#include <string.h>
#include "functions.h"
//Function to add a new expense
void addExpense(struct Expense expenses[], int
*count)
{
    printf("\nEnter Expense Name: ");
    scanf(" %49[^\n]", expenses[*count].name);
//Read string with spaces
    getchar();
    printf("\nEnter Amount: ");
    scanf("%f", &expenses[*count].amount); //Read
amount spent
    printf("\nEnter Date: ");
    scanf(" %s", expenses[*count].date); //Read date
    (*count)++; //Increase the the total expense
count
    printf("\nExpense Added Successfully!\n");
}
//Function to view all expenses
void viewExpenses(struct Expense expenses[], int
count)
{
    if (count == 0)
    {
        printf("\nNo Expenses Recorded Yet!\n");
        return;
    }
    printf("\n-----YOUR EXPENSES-----\n");
    printf("S.No.  Name  \t  Amount  \t  Date\n");
    //Loop through all expenses and print them
    for (int i = 0; i < count; i++)
    {
        printf("%d.  %s  \t  Rs%.2f  \t  %s\n", i+1,
expenses[i].name, expenses[i].amount,
expenses[i].date);
    }
}
//Function to calculate total money spent
void totalSpent(struct Expense expenses[], int
count)
{
    float sum = 0;
    //Sum all expense amounts
    for (int i = 0; i < count; i++)
    {
        sum += expenses[i].amount;
    }
    printf("\nTotal Money Spent: Rs%.2f\n", sum);
}
//Function to save expenses to a text file
void saveToFile(struct Expense expenses[], int
count)
{
    //Open file in append mode so that the old data
is not erased
    FILE *fp = fopen("expenses.txt", "a");
    if (fp == NULL)
    {
        printf("\nError Opening File!\n");
        return;
    }
    //Write each expense to the file
    for (int i = 0; i < count; i++)
    {
        fprintf(fp, "%s | %.2f | %s\n",
expenses[i].name, expenses[i].amount,
expenses[i].date);
    }
    //Calculate total before writing to file
    float total = 0;
    for (int i = 0; i < count; i++)
    {
        total += expenses[i].amount;
    }
    //Write total amount below expenses
    fprintf(fp, "Total Spent So Far: %.2f\n", total);
    //Separator for readability
    fprintf(fp, "--------------------------------\n") ;
    fclose(fp); //Close the file
    printf("\nExpenses Saved Successfully!\n");
}
```

functions.h –

```
#ifndef FUNCTIONS_H
```

```
#define FUNCTIONS_H

#define MAX 100 //Maximum number of
expenses

//Structure to store expense details

struct Expense

{

   char name[50]; //Name of the expense

   float amount; //Amount spent

   char date[15]; //Date of the expense

};

void addExpense(struct Expense expenses[], int
*count); //Function to add a new expense

void viewExpenses(struct Expense expenses[], int
count); //Function to view all expenses

void totalSpent(struct Expense expenses[], int
count); //Calculate total money spent
```

void saveToFile(struct Expense expenses[], int count); //Save all expenses to a text file

#endif

Conclusion:

The Daily Expense Tracker successfully records and manages everyday expenses.

It allows the user to enter multiple expenses, view all entries, calculate total spending, and save the data in a text file.

The project demonstrates the use of structures, arrays, file handling, and modular programming in C.

Future Enhancements:

- Add categories (Food, Travel, Shopping, etc.).
- Search expenses by date or name.
- Delete or edit an existing expense.
- Monthly or weekly report generation.