A Project Report

ON

EXPENSE TRACKER

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DIPLOMA IN

COMPUTER SCIENCE ENGINEERING

SUBMITTED TO

HARYANA STATE BOARD OF TECHNICAL EDUCATION, HARYANA PANCHKULA

SUBMITTED BY

S.No. Name Enrollment No.

1 Aaryan 230010824001

2 Ankush 230010824004

3 Harsh Saini 230010824006

4 Prince 230010824010

5 Simran 230010800060

Guided By

Payal Mam



GOVT. POLYTECHNIC, AMBALA CITY

**ACADEMIC YEAR**

**2023-2026**

# CERTIFICATE

**This is to certify that the project report entitled “**EXPENSE TRACKER**” was successfully completed by students of fourth semester Diploma in Computer engineering.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | SUBMITTED BY   |  |  |  | | --- | --- | --- | | S.No. | Name | Enrollment No. | | 1 | Aaryan | 230010824001 | | 2  3  4  5 | Ankush  Harsh Saini  Prince  Simran | 230010824004  230010824006  230010824010  230010800060 | |  |  | |  |
|  |  |  |
|  |  |  |

**In partial fulfillment of the requirements for the award of the Diploma in**

**COMPUTER ENGINEERING and submitted to the Department of COMPUTER ENGINEERING of Govt. Polytechnic, Ambala City work carried out during a period for the academic year 2023-2026 as per curriculum.**

Name of Guide: Name HOD: Name of Principal:

Payal Mam Virender Sir Dr. Rajiv Sapra

ACKNOWLEDGMENT

I express my heartfelt gratitude to all those who have contributed to the successful completion of my semester project, **"Expense Tracker."**

First and foremost, I would like to extend my sincere gratitude to Dr. Rajeev Sapra, Principal of , Govt. Polytechnic, Ambala City, for providing us with an excellent academic environment and continuous encouragement throughout our learning journey.

I am deeply grateful to Mr. Virender Gupta, Head of the **Computer Engineering**, for their constant guidance, support, and for creating opportunities that enabled me to successfully undertake this project.

I would also like to express my appreciation to my project guide, **Payal Mam**, whose valuable insights, encouragement, and constructive feedback have been instrumental in shaping this project. Their support has greatly contributed to my learning and understanding of the subject.

Furthermore, I extend my thanks to my institution, **Govt. Polytechnic , Ambala City**, for providing me with the necessary resources and facilities that made this project possible.

A special thanks to my friends and peers for their support, valuable discussions, and motivation during the project. Their encouragement has helped me overcome challenges and enhance the quality of my work.

Lastly, I would like to express my deep gratitude to my family for their unwavering support, patience, and encouragement throughout my academic journey.

This semester project has been an enriching experience, and I am grateful to everyone who played a role in its successful completion.

ABSTRACT

Managing personal finances efficiently is crucial in today’s fast-paced world. The **"Expense Tracker"** project aims to provide an easy-to-use solution for individuals to monitor and manage their daily expenses. This project is designed as a simple yet effective **C-based program** that allows users to record, categorize, and analyze their financial transactions.

The system enables users to **add expenses, view categorized spending, and track their financial habits** over time. By organizing transactions systematically, the application helps users maintain financial discipline and make informed budgetary decisions.

The project follows a structured programming approach, utilizing **file handling in C** to store and retrieve expense data. The system ensures efficient data management and allows users to interact with the program through a user-friendly console interface.

This project is an essential step towards **developing financial awareness** and serves as a foundation for future enhancements, such as integrating graphical user interfaces (GUI) or database support. Overall, the **Expense Tracker** is a practical and educational tool that strengthens programming concepts while addressing a real-world problem.

# **INDEX**

### ****1. Preliminary Pages****

* **1.1 Candidate’s Declaration** ........................................ Page:- 01
* **1.2 Certificate** .................................................................. Page:- 02
* **1.3 Acknowledgment** .................................................. Page:- 03
* **1.4 Abstract** .................................................................. Page:-04
* **1.5 Index** ...................................................................... Page:- 05

### ****2. Project Overview (****Page:- 6 & 7)

* **2.1 Introduction** .......................................................
  + **2.1.1 Overview of the Project** ................................
  + **2.1.2 Objective of the Project** ...................................
  + **2.1.2 Scope of the Project** ...................................
  + **2.1.2 Problem Statement**...................................

### ****3. System Requirements (****Page:- 8)

* **3.1 Software Requirements** ......................................................
* **3.2 Hardware Requirements** ....................................................

### ****4. Implementation & Features (Page:- 9 to 11 )****

* **4.1 Expense Management** ...........................................................
* **4.2 Category Management** .......................................................

### ****5. Conclusion & Future Scope (****Page:- 12 to 17)

* **5.1 Conclusion** ..............................................................................
* **5.2 Future Enhancements** ............................................................
* **5.3 References** ..............................................................................

### ****6.**** Appendix : Source Code ****(****Page:- 18+ )

# **INTRODUCTION**

## ****1.1 Overview****

Managing expenses efficiently is essential for both individuals and organizations. A **Project Expense Tracker** is a system designed to help users record, categorize, and analyze their expenditures systematically. This project aims to provide an easy-to-use expense tracking solution that ensures better financial management and planning.

## ****1.2 Objective****

The primary goal of this project is to develop an **efficient and user-friendly expense tracking system** that allows users to:

* Record daily expenses effortlessly.
* Categorize transactions for better analysis.
* Monitor spending patterns over time.
* Generate reports for better financial decision-making.

## ****1.3 Scope****

This **expense tracker** can be used by:

* **Students** to manage their daily expenses.
* **Working professionals** to track monthly budgets.
* **Small businesses** to keep track of project expenditures.

The system will support **manual data entry** and provide **basic analytical insights** to help users stay within budget. Future enhancements may include **automated data import, AI-based spending analysis, and cloud integration** for better accessibility.

## ****1.4 Problem Statement****

Many individuals and small businesses struggle with financial management due to **poor expense tracking** and **lack of budgeting tools**. Traditional methods, such as maintaining physical records or using spreadsheets, are often inefficient and time-consuming. This project aims to bridge the gap by providing a **simple, intuitive, and effective** solution for expense management.

### ****System Requirements****

1. **Software Requirements:**
   * Operating System: Windows/Linux/MacOS
   * Compiler: GCC (MinGW for Windows)
   * Programming Language: C
   * Text Editor: VS Code, Code::Blocks, Dev-C++
2. **Hardware Requirements:**
   * Minimum RAM: 2GB
   * Processor: Intel Core i3 or higher
   * Storage: At least 100MB free space

# **IMPLEMENTATION & FEATURES**

## ****4.1 Implementation****

The **Expense Tracker** is implemented in **C programming language**, utilizing **file handling** to store expenses persistently. It allows users to **add, view, delete, and categorize expenses**, making financial tracking more manageable.

### ****Key Functionalities:****

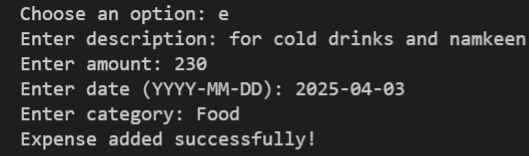
* **Adding an Expense** – Users input an amount, category, and date.
* **Viewing Expenses** – Displays expenses stored in the file.
* **Deleting an Expense** – Removes an expense entry from the system.
* **Generating Reports** – Summarizes expenses by category or date.
* **Category Management** – Users can classify expenses into predefined or custom categories.

## ****4.2 Features & Code Implementation****

### ****1. Adding an Expense****

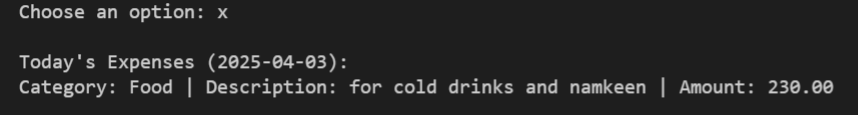
This feature allows users to enter expense details and store them in a file. The user can enter description along with every expense he wil add . User also have a option to fill the category if he want and they also have to add date along with them and amount is the main thing. Here is a example snippet

:

**

### ****2. Viewing Expenses****

Displays all the expenses he stored for the current date . [ if user want to generate all the expenses he can use the GENERATE REPORT feature….



### ****4.4 Category Management****

Users can manage expense categories by adding, deleting, and viewing them to better organize financial data. The user can add category and remove category and view category in this . Here are a complete explaination with example snippet:

#### ****4.4.1 Adding a Category****

Users can create a **new category** if the predefined ones do not fit their needs. This allows for a **personalized tracking system**.

#### add category.png

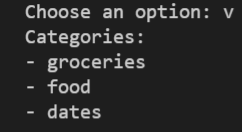
#### ****4.4.2 Deleting a Category****

If a category is no longer needed, users can remove it. This keeps the list **clean and relevant**.

#### remove category.png

#### ****4.4.3 Viewing Categories****

Users can view all available **expense categories**, including both predefined and custom ones, for quick reference.

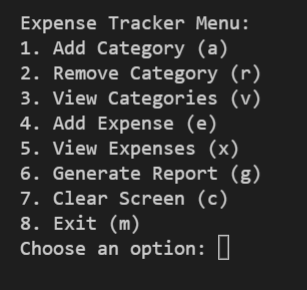


**CONCLUSION**

The Expense Tracker project provides a fundamental approach to managing daily expenses efficiently. By implementing basic features like **adding, viewing, deleting, and categorizing expenses**, users can track their spending habits more effectively. The program offers a **simple, structured interface** for expense management, making financial organization more accessible.

While the current system focuses on **core functionalities**, it lays the groundwork for future improvements. Users can interact with the system effortlessly and manage their expenses without the complexity of advanced financial software. However, the program is limited in **data storage**, as it does not yet retain information after exiting the session.

Overall, this project serves as a **basic but functional prototype** for expense tracking. It demonstrates the potential of a structured financial management tool and provides a strong foundation for **further enhancements** in usability, efficiency, and feature set.



# 

# **FUTURE ENHANCEMENTS**

To improve the Expense Tracker, several key enhancements can be implemented to make the system more practical and user-friendly. The current version operates within a **single runtime session**, meaning data is lost upon program exit. Enhancing the system with **persistent data storage** and additional features will significantly improve its usability.

P**lanned Upgrades:**

* **Persistent Storage (File Handling or Database)** – Implementing file handling or database integration will allow expenses to be saved permanently. This will ensure users can access their data even after restarting the application.
* **Graphical User Interface (GUI)** – Replacing the command-line interface with a **GUI-based application** will enhance user experience, making interactions more intuitive and visually appealing.
* **Expense Filtering & Search** – Implementing search and filtering features based on **date, amount, or category** will help users quickly retrieve specific expense records.
* **Expense Summarization & Statistics** – Adding **monthly summaries, category-based breakdowns, and spending trends** will allow users to analyze their financial habits and make informed decisions.
* **Multi-User Support** – Enabling a system where multiple users can manage their own expense records will make the tool more versatile for group or family expense tracking.
* **Security Features** – Implementing **password protection or authentication** will ensure users’ financial data remains secure.

By incorporating these enhancements, the Expense Tracker can evolve into a **powerful financial management tool**, catering to a broader audience with **better functionality and data security.**

# **REFERENCES**

📌 During the development of this project, various resources were used to understand programming concepts, design patterns, and software development methodologies. Below are some of the key references that guided the implementation:

1. **Programming Concepts:**
   * Fundamentals of **C programming** – Loops, Functions, and Structs.
   * Understanding **user input handling and menu-driven programs**.
   * Basics of **dynamic memory allocation and data structures**.
2. **Development Tools & Environment:**
   * **Code Editor**: [Mention the IDE used, e.g., Code::Blocks, VS Code].
   * **Compiler**: [Mention the compiler used, e.g., GCC, Turbo C].
   * Debugging and testing tools to identify and fix logical errors.
3. **Additional Learning Resources:**
   * Online **C programming tutorials** and forums for troubleshooting.
   * Open-source **expense tracker projects** for feature inspiration.
   * Research papers and blogs on **budgeting and expense tracking software**.

These references helped in designing a **structured, functional, and expandable** expense tracking system. Further improvements will continue to build upon these foundational concepts.

## ****Appendix : Source Code****

### ****Overview:****

This section contains the complete source code for the **Expense Tracker** project developed in the C programming language. The program allows users to add and view daily expenses using simple file handling techniques.

### ****Expense Tracker – Full C Program Code****

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#define MAX\_DESC 100

#define MAX\_CAT 50

typedef struct Expense {

char description[MAX\_DESC];

float amount;

char date[11]; // Format: YYYY-MM-DD

char category[MAX\_CAT];

struct Expense\* next;

} Expense;

typedef struct Category {

char name[MAX\_CAT];

struct Category\* next;

} Category;

Expense\* expenseHead = NULL;

Category\* categoryHead = NULL;

void clearScreen() {

system("clear || cls");

}

void addCategory() {

Category\* newCategory = (Category\*)malloc(sizeof(Category));

printf("Enter category name: ");

scanf("%s", newCategory->name);

newCategory->next = categoryHead;

categoryHead = newCategory;

printf("Category added successfully!\n");

}

void removeCategory() {

char name[MAX\_CAT];

printf("Enter category name to remove: ");

scanf("%s", name);

Category\* current = categoryHead;

Category\* previous = NULL;

while (current != NULL && strcmp(current->name, name) != 0) {

previous = current;

current = current->next;

}

if (current == NULL) {

printf("Category not found!\n");

return;

}

if (previous == NULL) {

categoryHead = current->next;

} else {

previous->next = current->next;

}

free(current);

printf("Category removed successfully!\n");

}

void viewCategories() {

Category\* current = categoryHead;

if (current == NULL) {

printf("No categories available.\n");

return;

}

printf("Categories:\n");

while (current != NULL) {

printf("- %s\n", current->name);

current = current->next;

}

}

void addExpense() {

Expense\* newExpense = (Expense\*)malloc(sizeof(Expense));

getchar(); // Clear any leftover newline before taking input

printf("Enter description: ");

fgets(newExpense->description, MAX\_DESC, stdin);

newExpense->description[strcspn(newExpense->description, "\n")] = '\0'; // Remove newline

printf("Enter amount: ");

if (scanf("%f", &newExpense->amount) != 1) {

printf("Invalid input for amount. Try again.\n");

free(newExpense);

return;

}

while (getchar() != '\n'); // ✅ Ensure buffer is cleared after amount input

printf("Enter date (YYYY-MM-DD): ");

fgets(newExpense->date, sizeof(newExpense->date), stdin);

newExpense->date[strcspn(newExpense->date, "\n")] = '\0'; // Remove newline

while (getchar() != '\n');

printf("Enter category: ");

fgets(newExpense->category, MAX\_CAT, stdin);

newExpense->category[strcspn(newExpense->category, "\n")] = '\0'; // ✅ Removes newline properly

newExpense->next = expenseHead;

expenseHead = newExpense;

printf("Expense added successfully!\n");

}

void viewExpenses() {

if (expenseHead == NULL) {

printf("No expenses recorded today. Use 'Generate Report' to view past expenses.\n");

return;

}

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

char today[11];

snprintf(today, sizeof(today), "%d-%02d-%02d", tm.tm\_year + 1900, tm.tm\_mon + 1, tm.tm\_mday);

Expense \*current = expenseHead;

int found = 0;

printf("\nToday's Expenses (%s):\n", today);

while (current != NULL) {

if (strcmp(current->date, today) == 0) {

printf("Category: %s | Description: %s | Amount: %.2f\n",

current->category, current->description, current->amount);

found = 1;

}

current = current->next;

}

if (!found) {

printf("No expenses recorded today. Use 'Generate Report' to view past expenses.\n");

}

}

void generateReport() {

if (expenseHead == NULL) {

printf("No expenses recorded.\n");

return;

}

int choice;

printf("\nGenerate report for:\n");

printf("1. Weekly\n");

printf("2. Monthly\n");

printf("3. Yearly\n");

printf("4. Overall\n");

printf("Choose an option: ");

scanf("%d", &choice);

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

int currentYear = tm.tm\_year + 1900;

int currentMonth = tm.tm\_mon + 1;

int currentDay = tm.tm\_mday;

double total = 0.0;

int count = 0; // Count number of days included

Expense\* current = expenseHead;

while (current != NULL) {

struct tm expenseDate = {0};

sscanf(current->date, "%d-%d-%d", &expenseDate.tm\_year, &expenseDate.tm\_mon, &expenseDate.tm\_mday);

expenseDate.tm\_year -= 1900;

expenseDate.tm\_mon -= 1;

time\_t expenseTime = mktime(&expenseDate);

double daysDifference = difftime(t, expenseTime) / (60 \* 60 \* 24);

int include = 0;

if (choice == 1 && daysDifference < 7) { // Weekly

include = 1;

} else if (choice == 2 && (expenseDate.tm\_year == tm.tm\_year) && (expenseDate.tm\_mon == tm.tm\_mon)) { // Monthly

include = 1;

} else if (choice == 3 && (expenseDate.tm\_year == tm.tm\_year)) { // Yearly

include = 1;

} else if (choice == 4) { // Overall

include = 1;

}

if (include) {

printf("Date: %s | Category: %s | Description: %s | Amount: %.2f\n",

current->date, current->category, current->description, current->amount);

total += current->amount;

count++;

}

current = current->next;

}

double avgDailyExpense = (count > 0) ? (total / count) : 0; // Avoid division by zero

printf("\nTotal Expenses: %.2f\n", total);

printf("Average Daily Expense: %.2f\n", avgDailyExpense);

}

void displayMenu() {

printf("\nExpense Tracker Menu:\n");

printf("1. Add Category (a)\n");

printf("2. Remove Category (r)\n");

printf("3. View Categories (v)\n");

printf("4. Add Expense (e)\n");

printf("5. View Expenses (x)\n");

printf("6. Generate Report (g)\n");

printf("7. Clear Screen (c)\n");

printf("8. Exit (m)\n");

}

int main() {

char choice;

while (1) {

displayMenu();

printf("Choose an option: ");

scanf(" %c", &choice);

switch (choice) {

case 'a':

addCategory();

break;

case 'r':

removeCategory();

break;

case 'v':

viewCategories();

break;

case 'e':

addExpense();

break;

case 'x':

viewExpenses();

break;

case 'g':

generateReport();

break;

case 'c':

clearScreen();

break;

case 'm':

printf("Exiting the program.\n");

exit(0);

default:

printf("Invalid option! Please try again.\n");

}

}

return 0;

}