R. P. Shaha University

- an institution of Kumudini Welfare Trust of Bengal (BD) Ltd.

Department: COMPUTER SCIENCE AND ENGINEERING
Semester: Spring 2024

Program: Bachelor of Computer Science and Engineering
Course Title: Algorithms Lab
Course Code: CSE 235

DOCUMENTATION OF PROJECT

Student Name: Munna Molla

Student ID: 22300056

Student Batch: 25th

Submission Date: 8/July/2024

Course Teacher: Dr. Kingkar Prosad Ghosh

NARAYANGANJ 2024

INDEX

Aim of the Project :

To provide individuals with an efficient, user-friendly tool to manage their finances by tracking income, expenses, budgets, and financial goals.

Abstract :

The "Personal Finance Tracker" project is designed to help users manage their personal finances by recording transactions, setting budgets, and defining financial goals. The system aims to provide an intuitive interface for tracking income and expenses, generating detailed financial reports, and ensuring better financial planning and decision-making.

Introduction :

The "Personal Finance Tracker" project aims to improve personal financial management by providing a straightforward and effective tool for tracking income and expenses, setting budgets, and defining financial goals. This software leverages data storage to maintain records securely and generate detailed financial reports.

In today's fast-paced world, managing personal finances can be challenging. This project addresses the need for a systematic approach to personal finance management, offering features to track financial activities and provide insights into spending and saving patterns. The intuitive interface simplifies the process of recording transactions and generating reports, enabling users to make informed financial decisions.

The system allows users to:

- Record income and expense transactions.
- Set and manage budgets for different categories.
- Define and track progress toward financial goals.
- Generate detailed financial reports.

Background Study :

Effective personal finance management requires meticulous tracking of income, expenses, and savings. Many individuals struggle with manual record-keeping or rely on disparate tools that fail to provide a comprehensive view of their financial situation. The "Personal Finance Tracker" addresses these issues by offering a unified platform for managing personal finances.

This project leverages structured data storage to ensure data integrity and quick access to financial records. The user-friendly interface is designed to minimize the learning curve, making it accessible to users with varying levels of financial literacy. By providing real-time feedback and comprehensive reports, the system helps users understand their financial health and make better financial decisions.

Clarification of the Problem :

In developing the "Personal Finance Tracker" project, several key challenges need to be addressed, particularly concerning CRUD (Create, Read, Update, Delete) operations, data management, and user interface design.

CRUD Operations:

- **Create:** Efficiently creating new transaction records while ensuring all necessary data fields are accurately filled out and saved.
- **Read:** Allowing quick and reliable retrieval of transaction data, including searching for specific records without delays.
- **Update:** Updating existing transactions when modifications are necessary, such as changes in amounts or categories, while maintaining data integrity.
- **Delete:** Safely and securely deleting transactions or budget entries, ensuring compliance with data retention policies and preventing unauthorized data loss.

Data Management:

- **Data Storage:** Designing a robust data structure that can handle large volumes of data, ensuring scalability and reliability.
- **Data Integrity:** Ensuring the accuracy and consistency of data through validation and normalization.
- **Security:** Protecting sensitive financial information through appropriate security measures such as encryption and access controls.

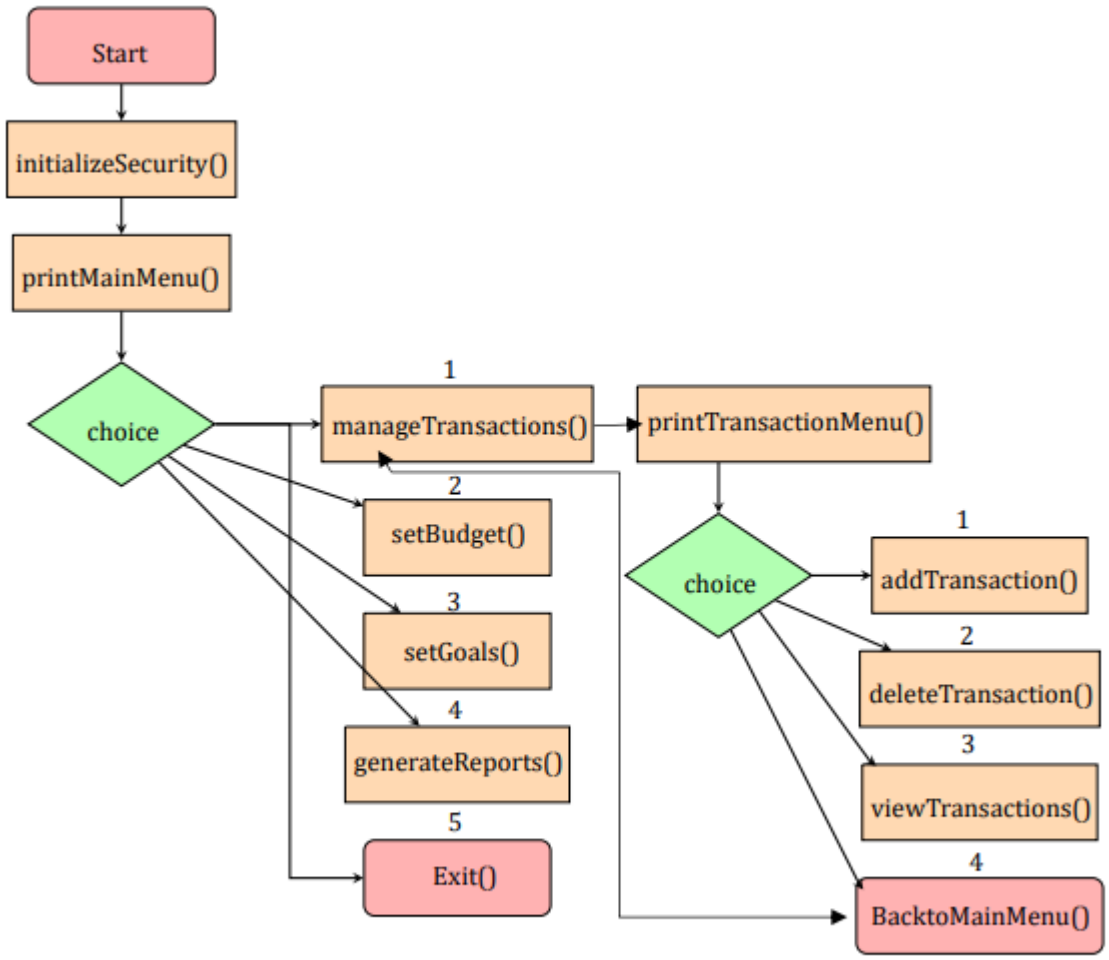
User Interface:

- **Usability:** Designing an intuitive and user-friendly interface that simplifies the process of recording transactions and generating reports.
- **Accessibility:** Ensuring the interface is accessible to all users, including those with disabilities, and can be effectively used on various devices.
- **Integration:** Seamlessly integrating the interface with other personal finance tools and systems to provide a cohesive user experience.
- **Feedback:** Providing real-time feedback to users, such as alerts for potential budget overruns or missing information, to enhance accuracy and financial awareness.

Algorithm of the solution:

```
Step 1: Initialize data structures for transactions, budgets, and
goals.

Step 2: Display main menu options.

Step 3: Wait for user input.

Step 4: If user chooses:

1: Add Transaction:
Prompt for transaction details, validate, and save.
2: Set Budget:
Prompt for budget details, validate, and save.
3: Set Goal:
Prompt for goal details, initialize saved amount, validate, and save.
4: Generate Report:
Calculate income, expenses, net savings.
Display transaction details, budgets, goals, and remaining amount.
5: Exit the application.

Step 5: Repeat from Step 2.
```

Flowchart :


Start
initializeSecurity()
printMainMenu()
choice
1
manageTransactions()
printTransactionMenu()
2
setBudget()
3
setGoals()
4
generateReports()
5
Exit()
choice
1
addTransaction()
2
deleteTransaction()
3
viewTransactions()
4
BacktoMainMenu()

Code of the project :

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_TRANSACTIONS 100
#define MAX_CATEGORIES 10
#define MAX_GOALS 5

typedef struct {
    char type[10];
    char category[20];
    float amount;
} Transaction;

typedef struct {
    char name[20];
    float amount;
} Budget;

typedef struct {
    char goalName[20];
    float goalAmount;
    float savedAmount;
} Goal;

Transaction transactions[MAX_TRANSACTIONS];
Budget budgets[MAX_CATEGORIES];
Goal goals[MAX_GOALS];

int transactionCount = 0;
int budgetCount = 0;
int goalCount = 0;

void initializeSecurity();
void printMainMenu();
void manageTransactions();
void addTransaction();
void deleteTransaction();
void viewTransactions();
void setBudget();
void viewBudget();
void setGoals();
void viewGoals();
void generateReports();
void generateReport();
void viewReport();

void clearInputBuffer();

int main() {
    initializeSecurity();
    int choice;
    while (1) {
        printMainMenu();
        scanf("%d", &choice);
        clearInputBuffer();

        switch (choice) {
            case 1:
                manageTransactions();
                break;
            case 2:
```

```
63.                 setBudget();
64.                 break;
65.             case 3:
66.                 setGoals();
67.                 break;
68.             case 4:
69.                 generateReports();
70.                 break;
71.             case 5:
72.                 printf("Exiting...\n");
73.                 return 0;
74.             default:
75.                 printf("Invalid choice! Please try again.\n");
76.         }
77.     }
78.     return 0;
79. }
80.
81. void clearInputBuffer() {
82.     while (getchar() != '\n');
83. }
84.
85. void initializeSecurity() {
86.     printf("\n*****************************************\n");
87.     printf("*         Initializing Security          *\n");
88.     printf("*****************************************\n");
89. }
90.
91. void printMainMenu() {
92.     printf("\n*****************************************\n");
93.     printf("*         Personal Finance Tracker       *\n");
94.     printf("*****************************************\n");
95.     printf("* 1. Manage Transactions                 *\n");
96.     printf("* 2. Set Budget                          *\n");
97.     printf("* 3. Set Financial Goals                 *\n");
98.     printf("* 4. Generate Reports                    *\n");
99.     printf("* 5. Exit                                *\n");
100.    printf("*****************************************\n");
101.    printf("Enter your choice: ");
102. }
103.
104. void printTransactionMenu() {
105.    printf("\n*****************************************\n");
106.    printf("*          Manage Transactions           *\n");
107.    printf("*****************************************\n");
108.    printf("* 1. Add Transaction                     *\n");
109.    printf("* 2. Delete Transaction                  *\n");
110.    printf("* 3. View Transactions                   *\n");
111.    printf("* 4. Back to Main Menu                   *\n");
112.    printf("*****************************************\n");
113.    printf("Enter your choice: ");
114. }
115.
116. void manageTransactions() {
117.    int choice;
118.    int backToMainMenu = 0;
119.    while (!backToMainMenu) {
120.        printTransactionMenu();
121.        scanf("%d", &choice);
122.        clearInputBuffer();
123.
124.        switch (choice) {
125.            case 1:
126.                addTransaction();
127.                break;
```

```c
            case 2:
                deleteTransaction();
                break;
            case 3:
                viewTransactions();
                break;
            case 4:
                backToMainMenu = 1;
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
}

void addTransaction() {
    if (transactionCount >= MAX_TRANSACTIONS) {
        printf("Transaction limit reached!\n");
        return;
    }

    printf("Enter transaction type (income/expense): ");
    scanf("%s", transactions[transactionCount].type);
    printf("Enter category: ");
    scanf("%s", transactions[transactionCount].category);
    printf("Enter amount: ");
    scanf("%f", &transactions[transactionCount].amount);
    transactionCount++;
}

void deleteTransaction() {
    if (transactionCount == 0) {
        printf("No transactions to delete!\n");
        return;
    }

    int index;
    printf("Enter the transaction index to delete (0 to %d): ", transactionCount - 1);
    scanf("%d", &index);
    if (index < 0 || index >= transactionCount) {
        printf("Invalid index!\n");
        return;
    }

    for (int i = index; i < transactionCount - 1; i++) {
        transactions[i] = transactions[i + 1];
    }
    transactionCount--;
    printf("Transaction deleted.\n");
}

void viewTransactions() {
    printf("\n--- Transactions ---\n");
    for (int i = 0; i < transactionCount; i++) {
        printf("Index: %d, Type: %s, Category: %s, Amount: %.2f\n", i, transactions[i].type,
transactions[i].category, transactions[i].amount);
    }
}

void printBudgetMenu() {
    printf("\n****************************************\n");
    printf("*              Set Budget              *\n");
    printf("****************************************\n");
    printf("* 1. Set Budget                        *\n");
    printf("* 2. View Budget                       *\n");
```

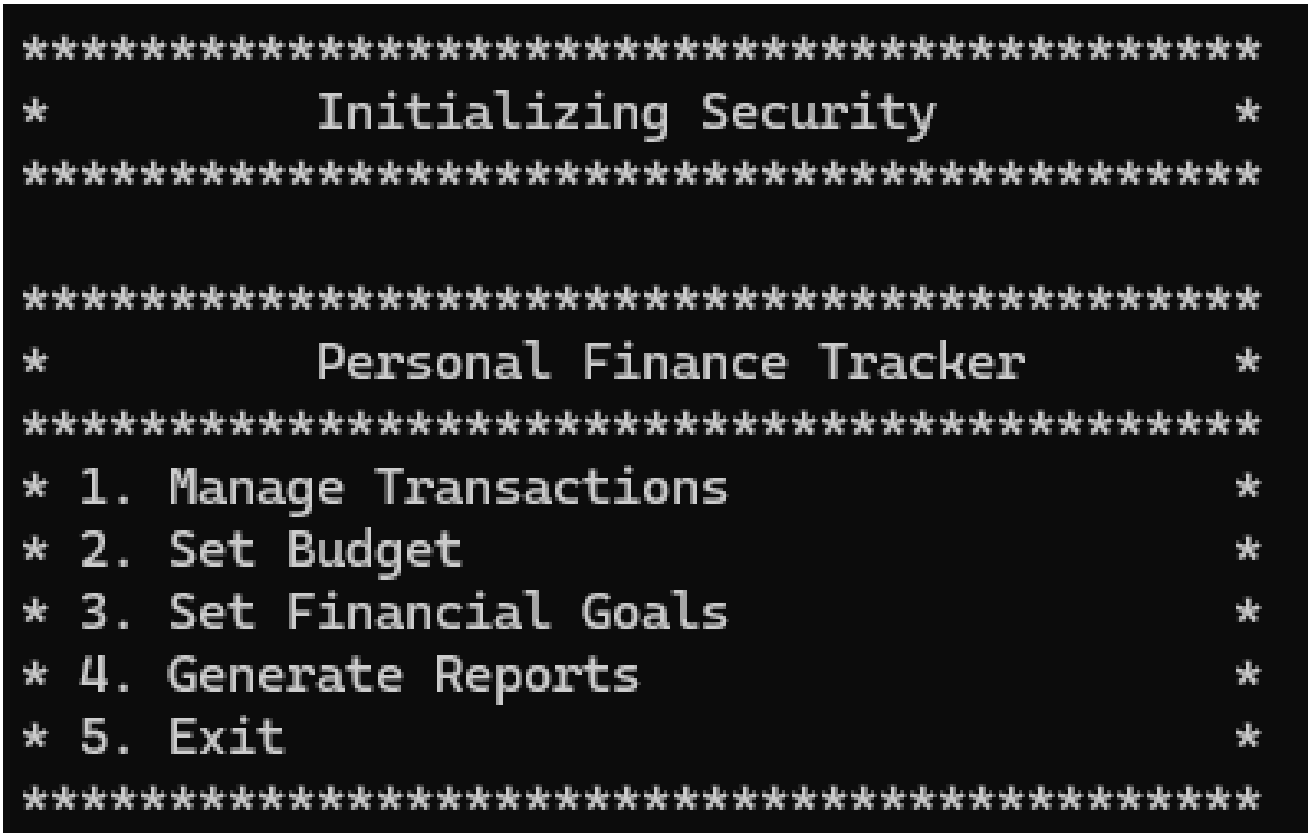
```
192.     printf("* 3. Back to Main Menu                   *\n");
193.     printf("*****************************************\n");
194.     printf("Enter your choice: ");
195. }
196.
197. void setBudget() {
198.     if (budgetCount >= MAX_CATEGORIES) {
199.         printf("Budget limit reached!\n");
200.         return;
201.     }
202.
203.     printf("Enter budget category: ");
204.     scanf("%s", budgets[budgetCount].name);
205.     printf("Enter budget amount: ");
206.     scanf("%f", &budgets[budgetCount].amount);
207.     budgetCount++;
208. }
209.
210. void viewBudget() {
211.     printf("\n--- Budgets ---\n");
212.     for (int i = 0; i < budgetCount; i++) {
213.         printf("Category: %s, Amount: %.2f\n", budgets[i].name, budgets[i].amount);
214.     }
215. }
216.
217. void printGoalMenu() {
218.     printf("\n*****************************************\n");
219.     printf("*         Set Financial Goals             *\n");
220.     printf("*****************************************\n");
221.     printf("* 1. Set Goals                            *\n");
222.     printf("* 2. View Goals                           *\n");
223.     printf("* 3. Back to Main Menu                    *\n");
224.     printf("*****************************************\n");
225.     printf("Enter your choice: ");
226. }
227.
228. void setGoals() {
229.     if (goalCount >= MAX_GOALS) {
230.         printf("Goal limit reached!\n");
231.         return;
232.     }
233.
234.     printf("Enter goal name: ");
235.     scanf("%s", goals[goalCount].goalName);
236.     printf("Enter goal amount: ");
237.     scanf("%f", &goals[goalCount].goalAmount);
238.     goals[goalCount].savedAmount = 0; // Initialize saved amount to 0
239.     goalCount++;
240. }
241.
242. void viewGoals() {
243.     printf("\n--- Goals ---\n");
244.     for (int i = 0; i < goalCount; i++) {
245.         printf("Goal: %s, Goal Amount: %.2f, Saved Amount: %.2f\n", goals[i].goalName,
goals[i].goalAmount, goals[i].savedAmount);
246.     }
247. }
248.
249. void printReportMenu() {
250.     printf("\n*****************************************\n");
251.     printf("*          Generate Reports               *\n");
252.     printf("*****************************************\n");
253.     printf("* 1. Generate Reports                     *\n");
254.     printf("* 2. View Report                          *\n");
255.     printf("* 3. Back to Main Menu                    *\n");
```

```
256.     printf("****************************************\n");
257.     printf("Enter your choice: ");
258. }
259.
260. void generateReports() {
261.     int choice;
262.     int backToMainMenu = 0;
263.     while (!backToMainMenu) {
264.         printReportMenu();
265.         scanf("%d", &choice);
266.         clearInputBuffer();
267.
268.         switch (choice) {
269.             case 1:
270.                 generateReport();
271.                 break;
272.             case 2:
273.                 viewReport();
274.                 break;
275.             case 3:
276.                 backToMainMenu = 1;
277.                 break;
278.             default:
279.                 printf("Invalid choice! Please try again.\n");
280.         }
281.     }
282. }
283.
284. void generateReport() {
285.     float totalIncome = 0, totalExpense = 0;
286.
287.     for (int i = 0; i < transactionCount; i++) {
288.         if (strcmp(transactions[i].type, "income") == 0) {
289.             totalIncome += transactions[i].amount;
290.         } else if (strcmp(transactions[i].type, "expense") == 0) {
291.             totalExpense += transactions[i].amount;
292.         }
293.     }
294.
295.     float netSavings = totalIncome - totalExpense;
296.
297.     printf("\n--- Financial Report ---\n");
298.     printf("Total Income: $%.2f\n", totalIncome);
299.     printf("Total Expense: $%.2f\n", totalExpense);
300.     printf("Net Savings: $%.2f\n", netSavings);
301.
302.     printf("\n--- Income Transactions ---\n");
303.     for (int i = 0; i < transactionCount; i++) {
304.         if (strcmp(transactions[i].type, "income") == 0) {
305.             printf("Category: %s, Amount: $%.2f\n", transactions[i].category,
transactions[i].amount);
306.         }
307.     }
308.
309.     printf("\n--- Expense Transactions ---\n");
310.     for (int i = 0; i < transactionCount; i++) {
311.         if (strcmp(transactions[i].type, "expense") == 0) {
312.             printf("Category: %s, Amount: $%.2f\n", transactions[i].category,
transactions[i].amount);
313.         }
314.     }
315.
316.     printf("\n--- Budgets ---\n");
317.     for (int i = 0; i < budgetCount; i++) {
318.         printf("Category: %s, Amount: $%.2f\n", budgets[i].name, budgets[i].amount);
```

```
319.     }
320.
321.     printf("\n--- Goals ---\n");
322.     for (int i = 0; i < goalCount; i++) {
323.         printf("Goal: %s, Goal Amount: $%.2f, Saved Amount: $%.2f\n", goals[i].goalName,
goals[i].goalAmount, goals[i].savedAmount);
324.     }
325.
326.     printf("\nTotal Remaining Amount: $%.2f\n", netSavings);
327. }
328.
329. void viewReport() {
330.     generateReport();
331. }
332.
```

Output :

1



2

```
Enter your choice: 1

****************************************
*           Manage Transactions        *
****************************************
* 1. Add Transaction                   *
* 2. Delete Transaction                *
* 3. View Transactions                 *
* 4. Back to Main Menu                 *
****************************************
```

3

```
****************************************
Enter your choice: 1
Enter transaction type (income/expense): income
Enter category: freelance
Enter amount: 50000

****************************************
*          Manage Transactions         *
****************************************
* 1. Add Transaction                   *
* 2. Delete Transaction                *
* 3. View Transactions                 *
* 4. Back to Main Menu                 *
****************************************
Enter your choice: 1
Enter transaction type (income/expense): expense
Enter category: phone
Enter amount: 20000

****************************************
```

4

```
****************************************
*          Generate Reports            *
****************************************
* 1. Generate Reports                  *
* 2. View Report                       *
* 3. Back to Main Menu                 *
****************************************
Enter your choice: 1

--- Financial Report ---
Total Income: $50000.00
Total Expense: $20000.00
Net Savings: $30000.00

--- Income Transactions ---
Category: freelance, Amount: $50000.00

--- Expense Transactions ---
Category: phone, Amount: $20000.00

--- Budgets ---
Category: camera, Amount: $30000.00

--- Goals ---
Goal: house, Goal Amount: $1000000.00, Saved Amount: $0.00

Total Remaining Amount: $30000.00

****************************************
```

Conclusion:

The "Personal Finance Tracker" project is designed to help users manage their personal finances efficiently. By providing a comprehensive tool for tracking income and expenses, setting budgets, and defining financial goals, this system aims to enhance financial planning and decision-making for individuals. The user-friendly interface and detailed reporting capabilities make it a valuable resource for anyone looking to improve their financial management practices.

Github Repository Link :

https://github.com/MunnaMollah

References :

- Youtube Playlist link: https://youtube.com/playlist?list=PLBlnK6fEyqRggZZgYpPMUxdY1CYkZtARR&si=DJ9dB0GYA20C3QaP
- Google Article : https://stackoverflow.com/questions/797701/financial-tracking-for-software-projects
- ChatGPT
- Reddit