# Budget Safe ( Budget Tracker with Data Privacy and Security )

*by* Muhammad Umer

**Submission date:** 25-Nov-2025 01:07PM (UTC+0500)
**Submission ID:** 2827197213
**File name:** Project_Result_Report_25K0983_25K0916_1.pdf (767.58K)
**Word count:** 674
**Character count:** 3801


NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES
nu

**University Name:** FAST NUCES Karachi Campus

**Department:** Department of Computer Science

**Course:** Programming Fundamentals

**Project Title:** Budget Safe ( Budget Tracker with Data Privacy and Security )

**Submitted By:** Muhammad Umer (25K-0983) & Khizar Khurseed (25K-0916)

**Submitted To:** Sir Sheeraz Iqbal

**Semester:** Fall 2025

**Date:** 22 Nov 2025

## Abstract

**Budget Safe** is a budget tracker system which ensures user-specific budget record management as well as having some basic data privacy and security features

## 1. Introduction

Most people want to manage their budget effectively and for this purpose, they use either apps or websites. But ignoring the fact that their data can be compromised. That's our problem statement which we have addressed by introducing **"Budget Safe"**

**Budget Safe** is a functional Budget Tracking project. Users can:

- Register and login securely
- Add, view, edit, delete expenses or income
- Generate monthly reports or financial alerts

## 2. Objectives

- To develop a secure user account system
- To preserve data by storing it in a file
- To allow users to add, view, and track their transaction records with ensuring basic data privacy

## 3. System Design

### Flow of the program:

Start
→ Home Menu → Login/ Register
→ If Login Success → Load User Dashboard
→ Dashboard Menu → Add/ View/ Edit/ Delete transactions
→ Dashboard Menu → Generate Monthly Report/ Alerts
→ Save data to file to avoid data loss when program stops
Exit

### Algorithm

1. Start the program
2. Display home menu (login / signup)
3. If user registers:
    - Prompt user for username
    - Prompt user for password
    - Hash password using hashing algorithm
    - Save these credentials to file
4. If user logs in:
    - Validate credentials
    - If matches → load dashboard

5. Dashboard operations:
    - Add record → encrypt amount → save to file
    - View record → decrypt amount → display
    - Edit/delete → rewrite records in file
    - Monthly report → total sum/ expense by month
    - Alert → compare total expense with given limit

6. Continue until user exits

### Input & Output

**Input:**

- Username
- Password

- Date
- Amount
- Transaction type (income/expense)

**Output:**

- Login success or failure
- Transaction recorded successfully
- Perform CRUD operations on transaction

## 4. Implementation

Language used: C

Compiler/IDE: GCC/ Visual Studio (VS) Code

### Key Features

- ➢ CLI interface with clean menu navigation
- ➢ Password hashing
- ➢ Basic data encryption
- ➢ CRUD operations on transactions (Add/View/Edit/Delete)
- ➢ Monthly report
- ➢ Expense alert

### Code Snippet

**1. Password Hash Function**

```c
unsigned long hashString(const char *str) {
    unsigned long hash = 5381;
    int c;
    while (*str != '\0') {
        c= *str;
        hash = ((hash << 5) + hash) + c;  // (hash * 33) + 112
        str++;
    }
    return hash;
}
```

### 2. Data structure

```
struct user {
    char username[50];
    unsigned long passwordHash;
};
```

```
struct budgetRecord{
    char date[30]; // in specific format DD-MM-YYYY
    float amount;
    char type[20]; // is it income or expense
};
```

### 3. Encryption algorithm

```
float encryptAmount(float amount) {
    return (amount *4) + 147.59;
}
```

### Sample Output

**For hashing function**

```
Enter a password: pass123
Hash Value: 399470012

Enter a password: pass12
Hash Value: 2615115593
```

## 5. Testing & Results

| Case | Input | Expected Output | Actual Output | Status |
|---|---|---|---|---|
| 1 | Login (correct credentials ) | Login successful | Login successful! | Pass |
| 2 | Login (wrong password) | Invalid password | Invalid username or password.<br>Attempts left n out of t (where n = attempts made by user & t=total attempts made) | Pass |
| 3 | Add transaction (correct data input) | Record added! | (Save the record in a user specific text file)<br>Record added! | Pass |
| 4 | Add transaction (with incorrect data format) | Cause runtime errors | Data got saved but it was useless to use in future | Fail |
| 5 | View transaction | Display all records | Display all records but decrypting the amount value | Pass |

The system performed consistently for the most of time as it was intended. However, case 4 failed because it will be considered in next versions.

## 6. Conclusion, Limitations & References

### Conclusion

This project showcases how by implementing the fundamental concepts of programming, a CLI based application can be built similar to Budget Safe. Additionally, security and data privacy features can be added to the application in order to provide a safe environment for users to interact with.

### Limitations

- Password hashing is basic & obsolete
- Amount encryption is easily reversible
- Basic analytics

### Future Enhancements

- Use of stronger encryption for all data as financial data is crucial
- Use of stronger hashing algorithms such as 256
- Add backup system for data
- Add charts/graphs for good analysis

### References


- Money Tracker-Expense & Budget App (By Horoscope365)

# Budget Safe ( Budget Tracker with Data Privacy and Security )

ORIGINALITY REPORT

| 7% | 5% | 0% | 5% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | w3resource.com<br>Internet Source | 3% |
| 2 | Submitted to Higher Education Commission Pakistan<br>Student Paper | 2% |
| 3 | www.slideshare.net<br>Internet Source | 1% |

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | < 3 words |
| Exclude bibliography | On | | |