Asset Management System

Almirol, Joel A.
Cacas, Jeff N.
Esguerra Paula G..
Doplayna, Clifford Lee C.

Technological Institute of the Philippines
Quezon City

November 2025

Table of Contents

Introduction

Digital currency offers a path toward a more efficient and secure financial system. Adopting it can address key limitations of using paper money, such as the risk of theft or loss, as well as the inconvenience of carrying large amounts of cash (ClearlyPayments, 2024; PayComplete, 2023). The formation of digital money institutions, which is also often called digital currencies, virtual money and cryptocurrencies, is a new evolving trend impacting the payment and monetary-system transformation (Adrian & Mancini-Griffoli, 2021; Investopedia, 2024). In this context, digital money is viewed “as a combination of two elements: an asset and an exchange mechanism which allows payment and settlement through the use of distributed-ledger technology” (Larin & Akimov, 2020, as cited in Li, 2024). By leveraging the best features of existing platforms, a new application can create a seamless and secure experience that encourages widespread adoption. To effectively protect against the inherent risks of digital currencies, such as hacking, phishing scams, and cybersecurity threats, the platform would implement a comprehensive security framework that goes beyond simple bug checks and surveys. This framework would include advanced authentication, continuous monitoring, and data protection (Bank of Canada, 2020)..

The main focus point is to develop an assets management system that allows customers to securely manage their financial assets. Poor money management skills, are a strong negative predictor of the tendency to overspend or worry about financial affairs and increase financial stress (Garðarsdottir & Dittmar, 2012). The application will enable users to create and access digital financial assets for managing savings, budgets, and future expenses. It aims to provide a secure and reliable asset management system, designed to empower both individuals and small businesses in managing their finances effectively. First, app development requires a careful blend of market research, to identify a clear target market (user group) that requires solution(s) to a specific problem. Second, the app's design should be uncomplicated (app features and behavior change techniques) (French et al., 2020). The platform will serve as a centralized space where users can create, access and monitor their financial assets. By offering tools for saving budgeting, and tracking future expenses the system aims to help users gain control over their money and make smarter financial decisions. Unlike basic mobile wallet applications, this system will focus on long-term financial management by offering detailed financial reports, budget planning, savings tracking, and asset categorization. Through an intuitive interface and secure data handling, the application will serve as a trusted digital financial space bridging the gap between everyday digital payments and comprehensive financial management.

The proposed solution is a secure, centralized digital asset management platform designed to assist individuals to efficiently manage both of the cash and digital funds. It will include tools that allow users to track their finances, whether for savings, paying bills, or budgeting. For the security features, the application will feature multi-factor authentication and encryption to protect users from hackers and scammers. Additionally, a feature will allow users to record and track funds lent to friends, family, or businesses. When a user lends money, they can log the transaction in a “Borrowers” tab by entering details such as the amount and recipient. This will help users easily monitor who still owes them money.

The Project

This project will develop a secure, centralized digital asset management system designed to help individuals, and small businesses to manage their assets well. The system will be built with security features to protect the user and the user's assets. The financial knowledge, understanding and basic financial skills which are fundamental to making good savings, investment and money management decisions. These are cognitive skills such as financial literacy as well as an understanding of financial products and services (French et al., 2020). Having confidence in your ability to manage your financial situation is key to improving financial well-being (Fernandes, Lynch, & Netemeyer, 2014; Letkiewicz, Robin-son, & Domian, 2016).

Objectives

The objective of the project is to develop an asset management system for individuals, businesses and financial advisors to track their financial growth efficiently. Specifically the project aims to:

1. Develop a system that:
    a. Allows users to manage both cash and digital funds in one interface.
    b. Provide users with functionalities for budget planning, savings tracking, and expense categorization.
    c. Integrate multi-factor authentication (MFA) and end-to-end encryption to safeguard user accounts and transactions.
    d. Help users understand spending habits to improve financial discipline.
2. Test and evaluate the system's functionality.

Flowchart of the System

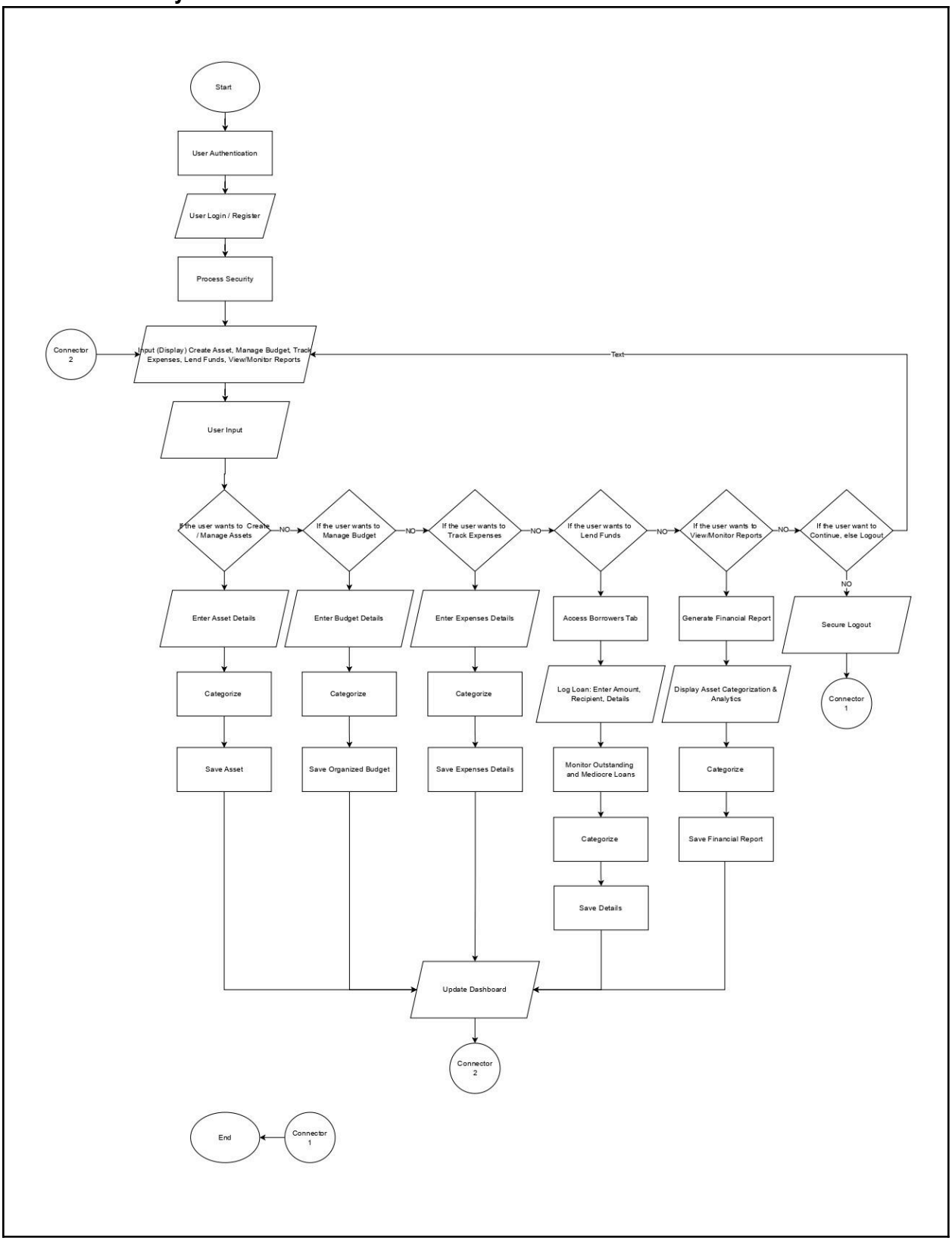


Figure 1: Asset management system

Figure 1 demonstrates the complete cycle of our financial management program. The cycle begins with the user logging in to the system using an authentication process to secure their information, ensuring that the user is the only one who has access to the account for the program. After the user logs in, he/she is able to select an action. The actions could be managing assets, managing budget, tracking expenses, lending funds, or reviewing reports. Moreover, once the user selects the option, the system will guide him/her through the step of entering necessary information such as asset information, budget numbers, or loan obtaining information, in each step. The last step required to do is for the user to categorize. From here, the user will then be able to save and retrieve information they entered. Additionally, for every action they conduct, the system will update a central dashboard which collects every financial data in one place that allows the user to have an overview of it. When the user is done, he/she can log out securely. In conclusion, this flowchart shows how our program provides a well-organized step-by-step method to manage their finances and keep track of their necessary financial matters.

Pseudocode

1.1 System Overview

The pseudocodes show the outline of a Secure Financial Management System. This program is designed to handle financial tasks while maintaining security measures throughout the interaction.

1.1.1 Initialization and Security

The system performs an authentication before accessing the main dashboard. This ensure the accounts will be secured before they access the dashboard.

1.1.2 Main Dashboard

Once the user is done in the authentication part they can already access the main dashboard which they can access or perform different actions such as creating or managing assets, tracking funds, view financial reports, adjust the system settings.

1.1.3 Logout

Once the session is done users can access the logout action to ensure that the account is properly closed and secured.

```
FUNCTION Authentication_System() // Steps: Create Account, Login, Verify Credentials
User_Choice = GET_INPUT("Enter '1' to Register or '2' to Login")

FUNCTION DISPLAY_MAIN_MENU() DISPLAY "Options: Create/Manage Assets | Manage
Budget |  Track Expense | Lend Funds | View / Monitor Reports | View Analytics Summary |
Logout" END FUNCTION

FUNCTION Manage_Assets() // Steps: Enter Asset Details, Categorize, Save Asset Asset_Data
= GET_INPUT("Asset details") CALL PROCESS_AND_SAVE(Asset_Data, "Asset") END
FUNCTION

FUNCTION Manage_Budget() // Steps: Enter Budget Details, Categorize, Save Organized
Budget Budget_Data = GET_INPUT("Budget details") CALL
PROCESS_AND_SAVE(Budget_Data, "Budget") END FUNCTION

FUNCTION Track_Expenses() // Steps: Enter Expense Details, Categorize, Save Expenses
Expense_Data = GET_INPUT("Expense details") CALL
PROCESS_AND_SAVE(Expense_Data, "Expense") END FUNCTION

FUNCTION Lend_Funds() // Steps: Enter Borrower Name, Enter Loan Amount. Enter Loan Tag
= GET_INPUT("Loan Details") CALL_PROCESS_AND_SAVE END FUNCTION

FUNCTION View_Report() // Steps: Retrieve Financial Data, Display Report Report_Data =
CALL GET_FINANCIAL_DATA()

FUNCTION View_Analytics_Summary() // Steps: Retrieve Data Totals, Display Analytics
Summary Analytics_Data = CALL GET_ANALYTICS_DATA()

FUNCTION Logout() // Steps: End Session, Return to Main Menu CALL
END_USER_SESSION() DISPLAY("Secure logout complete. Returning to main menu...") CALL
MAIN_MENU()

END FUNCTION
```

Figure 2: Pseudo code of Asset Management System

Figure 2: Pseudo code of Asset Management System presents the structure that shows the program flow and how it will work. The program begins by performing an authentication to ensure security among users. Once finished it will enter the main dashboard where users can navigate in their assets, budget and report. The user is prompted to select in the dashboard. Each choice has their own function that the user can use. Once the user chooses to logout the session is terminated and it will go back to the first interface which is the authentication panel.

Data Dictionary

The Data Dictionary of Asset Management System provides the detailed description of all the data used within the system. It defines each data's name, size, data type and description about the purpose. This ensures clarity in data usage across different modules.

The table of the Data Dictionary for the Financial Management System provides all the essential data elements used within the program. It defines each field's purpose, type, and role in managing financial

records. This table ensures consistency and accuracy of data across modules such as assets, budgets, expenses, reports, and loans.

Table 1: Data Dictionary

Data Name	Size	Data Type	Description
1. MAX_USERS	10	const int	Maximum number of users the system can store.
2. MAX_ENTRIES	100	const int	Maximum number of financial entries for assets,budget, etc the system can store
3. usernames	MAX_USERS (10)	string[ ]	Array to store all registered usernames.
4. passwords	MAX_USERS (10)	string[ ]	Array to store passwords corresponding to usernames.
5. userCount	1	int	Counter to track the current number of registered users.
6. assets	MAX_ENTRIES (100)	string[ ]	Array to store asset names/descriptions
7. assetTags	MAX_ENTRIES (100)	string[ ]	Array to store category tags of assets.
8. assetValues	MAX_ENTRIES (100)	float[ ]	Array to store monetary values of assets.
9. assetCount	1	int	Counter to track the current number of assets recorded.
10. budgets	MAX_ENTRIES (100)	string[ ]	Array to store budget categories/descriptions
11. budgetTags	MAX_ENTRIES (100)	string[ ]	Array to store tags for budget entries
12. budgetAmounts	MAX_ENTRIES	float[ ]	Array to store monetary

	(100)		amounts for budgets
13. budgetCount	1	int	Counter to track the current number of budget entries recorded
14. expenses	MAX_ENTRIES (100)	string[ ]	Array to store expense descriptions
15. expenseTags	MAX_ENTRIES (100)	string[ ]	Array to store tags for expense entries
16. expenseAmounts	MAX_ENTRIES (100)	float[ ]	Array to store monetary amounts for expenses
17. expenseCount	1	int	Counter to track the current number of expenses recorded
18. borrowers	MAX_ENTRIES (100)	string[ ]	Array to store names of people who borrowed funds
19. loanTags	MAX_ENTRIES (100)	string[ ]	Array to store tags for loan entries
20. loanAmounts	MAX_ENTRIES (100)	float[ ]	Array to store monetary amounts for loans lent
21. loanCount	1	int	Counter to track the current number of loans recorded
22. choice	1	int	Stores the user's menu selection in the dashboard
23. currentUser	1	string	Stores the username of the current logged-in user
24. uname	1	string	Temporary variable for entering a username during login
25. pword	1	string	Temporary variable for entering a password during login

26. running	1	bool	Flag to control the loop in the dashboard
27. totalAssets	1	float	Accumulator of the total value in all assets in showAnalytics
28. totalBudget	1	float	Accumulator for the total value of all budget in showAnalytics
29. totalExpenses	1	float	Accumulator for the total value of all expenses in showAnalytics
30. totalLoans	1	float	Accumulator for the total value of all loans in showAnalytics
31. i	1	int	Standard loop counter used in several functions

Code

This program implements a basic console-based financial management system. It allows users to register, log in, and perform financial tracking tasks such as managing assets, tracking budgets, logging expenses, and recording loans. All data is stored in memory using arrays, it is temporary and lost when the program ends. The system operates entirely within the console, providing a menu-driven interface for user interaction.

```
#include <iostream>
#include <string>
using namespace std;

// Constants
const int MAX_USERS = 10;
const int MAX_ENTRIES = 100;

// User database
string usernames[MAX_USERS];
string passwords[MAX_USERS];
int userCount = 0;

// Financial data arrays
string assets[MAX_ENTRIES], assetTags[MAX_ENTRIES];
float assetValues[MAX_ENTRIES];
int assetCount = 0;

string budgets[MAX_ENTRIES], budgetTags[MAX_ENTRIES];
float budgetAmounts[MAX_ENTRIES];
int budgetCount = 0;

string expenses[MAX_ENTRIES], expenseTags[MAX_ENTRIES];
float expenseAmounts[MAX_ENTRIES];
int expenseCount = 0;

string borrowers[MAX_ENTRIES], loanTags[MAX_ENTRIES];
float loanAmounts[MAX_ENTRIES];
int loanCount = 0;

// Function declarations
void registerUser();
```

```
bool loginUser(string &currentUser);
void dashboard(string currentUser);
void createAsset();
void manageBudget();
void trackExpenses();
void lendFunds();
void viewReports();
void showAnalytics();
void secureLogout();

int main() {
    int choice;
    string currentUser;

    while (true) {
        cout << "\n--- Welcome to Financial Management System ---\n";
        cout << "1. Register\n2. Login\n3. Exit\nChoose: ";
        cin >> choice;

        switch (choice) {
            case 1: registerUser(); break;
            case 2:
                if (loginUser(currentUser)) {
                    dashboard(currentUser);
                }
                break;
            case 3: cout << "Goodbye!\n"; return 0;
            default: cout << "Invalid choice.\n";
        }
    }
}

// Register new user
void registerUser() {
    if (userCount < MAX_USERS) {
        cout << "Enter new username: ";
        cin >> usernames[userCount];
        cout << "Enter new password: ";
        cin >> passwords[userCount];
```

```
        userCount++;
        cout << "Registration successful!\n";
    } else {
        cout << "User limit reached.\n";
    }
}

// Login existing user
bool loginUser(string &currentUser) {
    string uname, pword;
    cout << "Username: ";
    cin >> uname;
    cout << "Password: ";
    cin >> pword;

    for (int i = 0; i < userCount; i++) {
        if (usernames[i] == uname && passwords[i] == pword) {
            cout << "Login successful!\n";
            currentUser = uname;
            return true;
        }
    }
    cout << "Login failed.\n";
    return false;
}

// Dashboard
void dashboard(string currentUser) {
    int choice;
    bool running = true;

    while (running) {
        cout << "\n--- Dashboard (" << currentUser << ") ---\n";
        cout << "1. Create / Manage Assets\n";
        cout << "2. Manage Budget\n";
        cout << "3. Track Expenses\n";
        cout << "4. Lend Funds\n";
        cout << "5. View / Monitor Reports\n";
        cout << "6. View Analytics Summary\n";
```

```
        cout << "7. Logout\n-------------------------\n";
        cin >> choice;

        switch (choice) {
            case 1: createAsset(); break;
            case 2: manageBudget(); break;
            case 3: trackExpenses(); break;
            case 4: lendFunds(); break;
            case 5: viewReports(); break;
            case 6: showAnalytics(); break;
            case 7: secureLogout(); running = false; break;
            default: cout << "Invalid choice.\n";
        }
    }
}

// Asset Management
void createAsset() {
    if (assetCount < MAX_ENTRIES) {
        cout << "Enter asset name: ";
        cin >> assets[assetCount];

        cout << "Enter asset value (PHP): ";
        cin >> assetValues[assetCount];

        cout << "Enter asset category tag: ";
        cin >> assetTags[assetCount];

        assetCount++;
        cout << "Asset saved and categorized.\n";
    } else {
        cout << "Asset storage full.\n";
    }
}

// Budget Management
void manageBudget() {
    if (budgetCount < MAX_ENTRIES) {
        cout << "Enter budget category: ";
```

```
        cin >> budgets[budgetCount];

        cout << "Enter budget amount (PHP): ";
        cin >> budgetAmounts[budgetCount];

        cout << "Enter budget tag: ";
        cin >> budgetTags[budgetCount];

        budgetCount++;
        cout << "Budget saved and categorized.\n";
    } else {
        cout << "Budget storage full.\n";
    }
}

// Expense Tracking
void trackExpenses() {
    if (expenseCount < MAX_ENTRIES) {
        cout << "Enter expense description: ";
        cin >> expenses[expenseCount];

        cout << "Enter expense amount (PHP): ";
        cin >> expenseAmounts[expenseCount];

        cout << "Enter expense tag: ";
        cin >> expenseTags[expenseCount];

        expenseCount++;
        cout << "Expense saved and categorized.\n";
    } else {
        cout << "Expense storage full.\n";
    }
}

// Lending Funds
void lendFunds() {
    if (loanCount < MAX_ENTRIES) {
        cout << "Enter borrower name: ";
        cin >> borrowers[loanCount];
```

```
        cout << "Enter loan amount (PHP): ";
        cin >> loanAmounts[loanCount];

        cout << "Enter loan tag: ";
        cin >> loanTags[loanCount];

        loanCount++;
        cout << "Loan recorded and categorized.\n";
    } else {
        cout << "Loan storage full.\n";
    }
}

// Reports
void viewReports() {
    cout << "\n--- Financial Report ---\n";

    cout << "\nAssets:\n";
    for (int i = 0; i < assetCount; i++)
        cout << assets[i] << " [" << assetTags[i] << "]: PHP " <<
assetValues[i] << "\n";

    cout << "\nBudgets:\n";
    for (int i = 0; i < budgetCount; i++)
        cout << budgets[i] << " [" << budgetTags[i] << "]: PHP " <<
budgetAmounts[i] << "\n";

    cout << "\nExpenses:\n";
    for (int i = 0; i < expenseCount; i++)
        cout << expenses[i] << " [" << expenseTags[i] << "]: PHP " <<
expenseAmounts[i] << "\n";

    cout << "\nLoans:\n";
    for (int i = 0; i < loanCount; i++)
        cout << borrowers[i] << " [" << loanTags[i] << "]: PHP " <<
loanAmounts[i] << "\n";
}
```

```
// Analytics Summary
void showAnalytics() {
    float totalAssets = 0, totalBudget = 0, totalExpenses = 0,
totalLoans = 0;

    for (int i = 0; i < assetCount; i++) totalAssets +=
assetValues[i];
    for (int i = 0; i < budgetCount; i++) totalBudget +=
budgetAmounts[i];
    for (int i = 0; i < expenseCount; i++) totalExpenses +=
expenseAmounts[i];
    for (int i = 0; i < loanCount; i++) totalLoans += loanAmounts[i];

    cout << "\n--- Analytics Summary ---\n";
    cout << "Total Assets: PHP " << totalAssets << "\n";
    cout << "Total Budget: PHP " << totalBudget << "\n";
    cout << "Total Expenses: PHP " << totalExpenses << "\n";
    cout << "Total Loans: PHP " << totalLoans << "\n";
}

// Logout
void secureLogout() {
    cout << "Secure logout complete. Returning to main menu...\n";
}
```

Figure 3. Asset Management System

The provided code is structured to offers various functions, organized around core data management tasks. The program utilizes global variables to store all application data. Constants like MAX_USERS and MAX_ENTRIES define the fixed size of the arrays used for storage.The main function acts as the entry point and primary control loop for the application's initial phase. It presents a menu to register, log in, or exit. If a user successfully logs in, the control is passed to the dashboard function, it manages the application's main functionality loop until the user chooses to log out. The functions createAsset(), manageBudget(), trackExpenses(), and lendFunds() all follow a pattern, they check if the relevant data array is full, prompt the user for input using cin, and store the new entry at the current count index before incrementing the count. The viewReports() and showAnalytics() functions iterate through the global arrays to display a summary of all recorded entries and calculate basic totals.

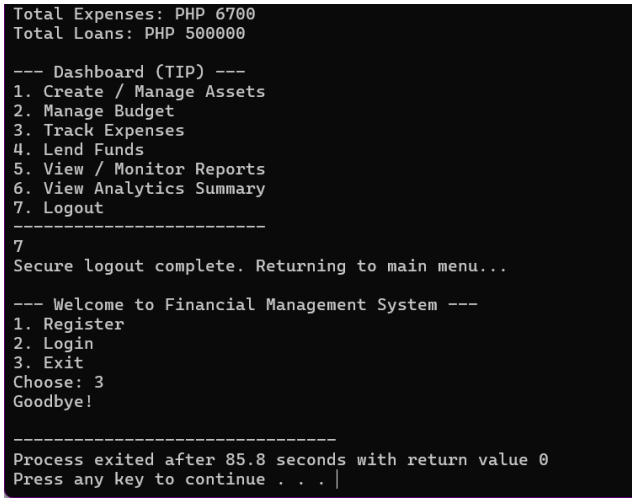
Results and Discussion

```
--- Welcome to Financial Management System ---
1. Register
2. Login
3. Exit
Choose: 1
Enter new username: TIP
Enter new password: tip
Registration successful!

--- Welcome to Financial Management System ---
1. Register
2. Login
3. Exit
Choose: 2
Username: TIP
Password: tip
Login successful!

--- Dashboard (TIP) ---
1. Create / Manage Assets
2. Manage Budget
3. Track Expenses
4. Lend Funds
5. View / Monitor Reports
6. View Analytics Summary
7. Logout
-------------------------
1
Enter asset name: A
```

```
Enter asset value (PHP): 200
Enter asset category tag: a
Asset saved and categorized.

--- Dashboard (TIP) ---
1. Create / Manage Assets
2. Manage Budget
3. Track Expenses
4. Lend Funds
5. View / Monitor Reports
6. View Analytics Summary
7. Logout
-------------------------
2
Enter budget category: B
Enter budget amount (PHP): 330.67
Enter budget tag: b
Budget saved and categorized.

--- Dashboard (TIP) ---
1. Create / Manage Assets
2. Manage Budget
3. Track Expenses
4. Lend Funds
5. View / Monitor Reports
6. View Analytics Summary
7. Logout
-------------------------
3
Enter expense description: C
```

```
Enter expense amount (PHP): 6700
Enter expense tag: c
Expense saved and categorized.

--- Dashboard (TIP) ---
1. Create / Manage Assets
2. Manage Budget
3. Track Expenses
4. Lend Funds
5. View / Monitor Reports
6. View Analytics Summary
7. Logout
-------------------------
4
Enter borrower name: D
Enter loan amount (PHP): 500000
Enter loan tag: d
Loan recorded and categorized.

--- Dashboard (TIP) ---
1. Create / Manage Assets
2. Manage Budget
3. Track Expenses
4. Lend Funds
5. View / Monitor Reports
6. View Analytics Summary
7. Logout
-------------------------
5
```

```
--- Financial Report ---

Assets:
A [a]: PHP 200

Budgets:
B [b]: PHP 330.67

Expenses:
C [c]: PHP 6700

Loans:
D [d]: PHP 500000

--- Dashboard (TIP) ---
1. Create / Manage Assets
2. Manage Budget
3. Track Expenses
4. Lend Funds
5. View / Monitor Reports
6. View Analytics Summary
7. Logout
-------------------------
6

--- Analytics Summary ---
Total Assets: PHP 200
Total Budget: PHP 330.67
Total Expenses: PHP 6700
```



The figure 1 demonstrates the steps of how the financial management program operates. It begins with authentication where the user logs in or registers to ensure account security. After logging in successfully, the user will select an option on how the user wants to use the program. It will then proceed to entering information to what they selected, which will be collected by the system to save and store each data. All saved information is displayed on a dashboard where the overview of the user's status is seen. After all of this, the user can finally log out safely. Figure 2 illustrates the pseudocode that explains how the Asset Management System operates. Similar to figure 1, it will start with the steps of authentication to ensure the user's data security. This will be followed by entering a dashboard where the user can review their assets, budget, and report. From there, the user will select an option, each with its own function. When the user completes their tasks, they can log out safely, ending the session and returning to the login screen. Figure 3 describes a simple console-based Financial Management System where users can register and log in using plain-text credentials stored in global arrays, limited to 10 users. Upon login, a dashboard menu allows managing financial data, including creating assets, managing budgets, tracking expenses, and recording loans, each stored in separate arrays with names, values in PHP, and tags, capped at 100 entries per category. The system features detailed reports of all entries and an analytics summary calculating total sums, uses nested loops for navigation with cin for input and cout for output, but lacks data persistence, input validation, security, or dynamic sizing, making it ideal for basic demonstrations rather than production use.

Conclusion

The systems developed by the team create a safe and simplified central system that can be used by clients and small businesses for efficient and objective financial asset management. Having key features that include planning a budget, tracking savings and streamlining the savings expense logging and loan monitoring, the customer can make confident decisions regarding their resources. The system is user friendly with a simplified controller and organized modules, while the varying logging system protects the system with encryption and multiple access thresholds. Design and implemented performance has built a solution regarding financial management with outstanding ease.

Prior to this the systems could include more modules that account for mobile access, systems real time tracking capabilities, and tailored advice that tracks the customers access pattern of the system. These features on top of what is already developed adds to the systems purpose and expands the systems overall. With the time and user suggestions implemented the system can be further developed to be a more efficient system to account for the financial resource tracking and management in the current times.

References

Array declaration - cppreference.com. (n.d.-b). https://en.cppreference.com/w/cpp/language/array.html

Adrian, T., & Mancini-Griffoli, T. (2021). A new era of digital money. IMF F&D. https://www.imf.org/external/pubs/ft/fandd/2021/06/online/digital-money-new-era-adrian-mancini-griffoli.htm

Bank of Canada. (2020). Security and convenience of a central bank digital currency (SAN 2020-21). https://www.bankofcanada.ca/wp-content/uploads/2020/10/san2020-21.pdf

Clearly Payments. (2024, March 18). Statistics for cash and credit card use for payments in 2024. https://www.clearlypayments.com/blog/statistics-for-cash-and-credit-card-use-for-payments-in-2024/

Garðarsdottir & Dittmar, (2012). Personal finance apps and low-income households https://www.researchgate.net/publication/353109750_Personal_finance_apps_and_low-income_households

French et al. (2020). Personal finance apps and low-income households https://www.researchgate.net/publication/353109750_Personal_finance_apps_and_low-income_households

Fernandes, Lynch, & Netemeyer, (2014); Letkiewicz, Robin-son, & Domian, (2016). Personal finance apps and low-income households https://www.researchgate.net/publication/353109750_Personal_finance_apps_and_low-income_households

Investopedia. (2024). Types and characteristics of digital currencies: Pros, cons. https://www.investopedia.com/terms/d/digital-currency.asp

Li, B. (2024). Research on digital currency and financial technology: Central bank digital currency (CBDC) and its role. https://www.shs-conferences.org/articles/shsconf/pdf/2024/20/shsconf_sess2024_01009.pdf

O. I. Larina, O. M. Akimov (2020). Digital Money at the Present Stage: Key Risks and Development Direction https://www.researchgate.net/publication/343731025_Digital_Money_at_the_Present_Stage_Key_Risks_and_Development_Direction

PayComplete. (2023). The advantages and disadvantages of cash payment. https://paycomplete.com/advantages-and-disadvantages-of-cash/

Standard library header - cppreference.com. (n.d.). https://en.cppreference.com/w/cpp/header/string.html