

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Smart Expense Tracker**

**(An AI-driven Budget Tracker)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CS-112 Project Report**

**Instructor:** M. Ahmad Nawaz

Group Members:

**Name Reg #**

Reyan Kashif 2024538

Ali Kamran 2024090

Irab Zara 2024244

**Table of Contents**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Sr. No Section Title**

1. Introduction and Problem Statement

2. System Design and Architecture

3. Explanation of OOP Concepts in C++

4. Implementation Details and Challenges

5. Screenshots and Outputs

6. Conclusion and Future Improvements

## 

## **Introduction**

Our devised system is an advanced tool designed to manage finances as it automates expense tracking, categorization, and reporting by using Artificial Intelligence. It presents a user-friendly interface, real-time tracking by linking bank accounts, and hence generates detailed spending reports. The software allows for better budgeting by generating alerts for overspending and additionally ensures data security through cryptographic principles. Primarily built on the basis of Object-Oriented Programming concepts, it enhances maintainability and scalability, making it an effective solution for managing personal finances in today's busy world.

## **Problem Statement**

Manually managing personal expenses is inefficient and more importantly, prone to human error. Users may forget to log transactions, categorize expenses incorrectly, or they may entirely lose track of their budgets. While several expense tracking applications exist already, many of them require substantial manual input, lack personalized categorization, or may not provide sufficient security for sensitive user information.

* This project aims to solve these problems by developing an AI-powered budgeting system that:
* Enables secure user authentication based on **SHA-256 hashing** and **AES-256 encryption** methods.
* Automatically classifies expenditures based on **AI API integration**.
* Saves user information and categorized expenditures locally on CSV files and does not need cloud storage.
* Offers insights and reports for spending based on user input.
* Has a plain but efficient user interface for the addition, control, and presentation of expenses to users.

With expense categorization being automated and enhanced security, the system reduces manual effort, provides financial awareness equipped with figures, and provides users with an efficient way to track and then manage their budgets.

# 

# **System Design and Architecture**

# **System Design Overview**

The Expense Tracker is a C++ based application to manage user credentials, record expenses, and provides categorized reports of spending securely. It follows a modular, Object-Oriented Programming structure meaning it produces code that is reusable, scalable, and easy to maintain.

The system has four main components:

**1. Authentication Module**

Handles user signup and login using SHA-256 hashing for passwords and AES-256 encryption for usernames and emails.

**2. Expense Manager Module**

Allows users to record expenses through two ways- AI-based categorization and backup option- manual entry. Expense data is stored in AI categorized CSV files.

**3. AI Categorization Module**

Utilizes the OpenAI API to auto-extract the amount, description, and suitable category from the user's input text.

**4. Reporting and Visualization Module**

Displays detailed spending reports based on category, and summary reports displaying their overall total expenses to the user.

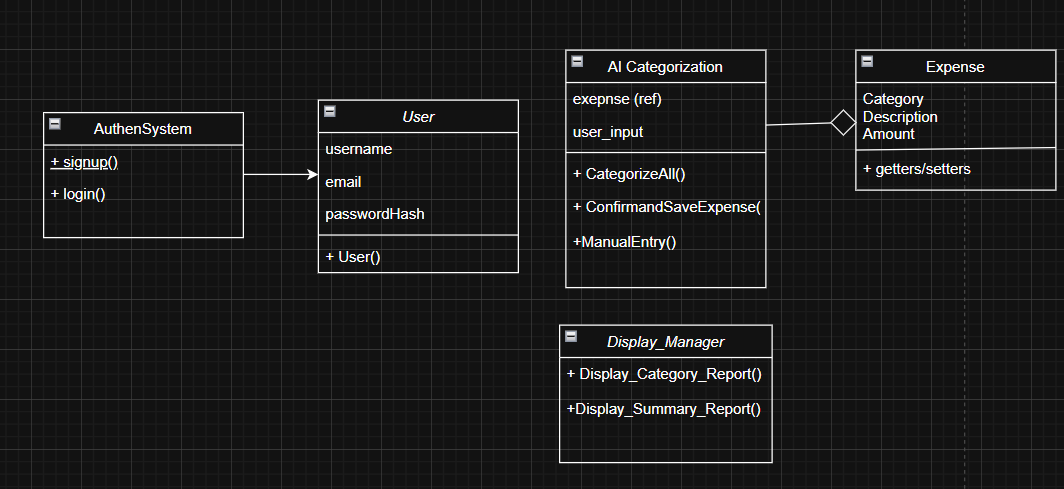
This unit-based design ensures clear separation, and enables easy future enhancements, such as adding new AI models or integrating banking APIs.

## 

## **System Architecture Diagram (UML)**

Provided below is a simple UML class diagram to represent our project’s structure:

* **AuthenSystem** handles secure user login/signup
* AI\_Categorization operates directly on an **Expense** object to populate its fields
* **Display\_Manager** reads expense files and generates reports.



# **How OOP Concepts Were Applied**

The **Smart Expense Tracker – AI Powered Budget Planner** applies Object-Oriented Programming (OOP) concepts to ensure the system produced is modular, secure and easy to expand and maintain. The concepts used are:

## **1. Encapsulation**

Encapsulation is achieved by keeping class attributes private and providing public methods for accessing and modifying them.

**Example:**

* In the Expense class, the attributes Amount, Category, and Description are private.
* Public setter and getter methods control how these values are accessed and updated.

class Expense {

private:

string Amount;

string Category;

string Description;

public:

void set\_amount(string amount);

string get\_amount();

};

## **2. Abstraction**

Abstraction is applied by hiding the complexity-like the logic of encryption, AI categorization, and file handling behind simple methods that are typically called.

**Examples:**

* Users interact with high-level methods like signup(), login(), and CategorizeAll() without needing to know the internal details of password hashing, encryption, or AI API communication.
* The AI\_Categorization class abstracts the OpenAI API call and text processing internally

## **3. Inheritance**

Although direct inheritance was not implemented in this version of the project, future versions could introduce inheritance by creating specialized expense classes like FoodExpense, TransportExpense, etc., inheriting from the base Expense class.

## **4. Polymorphism**

In future versions, different methods for categorization (different AI models, manual input) can override a common interface for expense entry.

## **5. Modular Design**

Each major functionality is carried out within a separate class:

* **Authentication** is handled by AuthenSystem.
* **Expense details** are managed by the Expense class.
* **AI categorization** is handled by AI\_Categorization.
* **Reports** are generated by Display\_Manager.

By creating these independent code units, the project follows the OOP principle of "Separation of Concerns" and makes the code easy to update and debug.

**Summary:**

|  |  |
| --- | --- |
| **OOP Concept** | **How It Was Used** |
| Encapsulation | Private attributes + getters/setters |
| Abstraction | Hiding complex operations behind simple interfaces |
| Inheritance | Could be implemented in future versions (specialized expenses) |
| Polymorphism | Could be implemented for multiple AI categorization methods |
| Modular Design | Different classes for authentication, expenses, reporting |

**Implementation Details and Challenges**

**Implementation Details**

The Smart Expense Tracker – AI Powered Budget Planner was implemented in C++, following contemporary Object-Oriented Programming (OOP) techniques. The project is designed around a number of essential parts, each dealing with a different functionality of the program.

**1. User Authentication**

The user authentication technique provides protection for user information. It employs two fundamental encryption methods:

- SHA-256 hashing to hash the password in the sign-up and login process.

- AES-256 encryption to securely store the username and email so that sensitive user information remains safe.

Process Details:

- When signing-up, the user enters the username, email, and password. Password hashing is done via SHA-256, and the username/email is encrypted using AES-256 before being stored in a local file (users.txt).

- To log back in, the user enters the same credentials, and the system hashes the password and encrypts the username/email to compare with the stored values. If they match, access is allowed.

**2. Expense Categorization**

The app uses AI-based categorization to auto- categorize expenses. This is achieved via integration with the OpenAI API, which processes the user input and identifies the category (e.g., food, transport).

Process Details:

- Users enter the expense information in text format (e.g., "500 at KFC").

- The application calls the AI\_Categorization class to forward the user input to the OpenAI API, which returns the category, amount, and description. The AI categorizes the expense (e.g., "Food") and saves it in a CSV file of match.

**3. Expense Management**

Expenses are saved locally in CSV files, grouped based on their type. In case the type does not exist, the system will create a new file for it. The system also allows manual entry, where users can manually enter expense information without AI intervention or in absence of the internet.

Process:

- The Expense class stores data for each expense, such as the amount, type, and description.

- Expenses are stored in individual CSV files for each category (e.g., food.csv, transport.csv), with each CSV file storing the timestamp, amount, and description.

**4. Reporting**

The Display\_Manager class gives spending reports by reading information from the CSV files. It allows users to:

- View category-wise spending reports (e.g., what they spent on food, transport, etc.).

- View a summary of total spending across all categories.

**5. Data Encryption**

To secure user data:

- AES-256 is utilized to encrypt sensitive data (such as usernames and emails).

- SHA-256 hashing is used to ensure that passwords are saved securely and cannot be reversed to uncover the original password.

**Challenges Faced**

During the implementation of the Smart Expense Tracker, quite a few technical challenges were encountered:

1. **Adding External Libraries**

Integrating cURL for API communication and OpenSSL for encryption proved challenging, particularly when it came to VS Code's dependency linking and configuration settings. Compiler flag management and environment path issues required a number of adjustments.

1. **Unstable Development Environment**

Unexpected compiler errors, VS Code behaving unexpectedly, and frequent terminal bugs halted environment development. To ensure a smooth workflow, careful environment setup and ongoing troubleshooting during compilation and execution were necessary.

1. **AI Response**

AI's occasional misinterpretation of prompts led to inaccurate or insufficient expense detail extraction. The prompt structure and API call parameters needed to be improved in order to ensure consistent AI classification.

1. **Management of Expense Data**

Errors like duplicate expense entries or incorrect amount saving were occasionally the result of manual entry. The integrity of the saved costs was preserved by improving input validation and simplifying the file writing logic.

**Conclusion and Future ImprovementsConclusion**

The project meets the increasing demand for effective and secure personal finance management. With the addition of AI-driven expense categorization and user authentication and local storage, the app offers a comprehensive and simple-to-use platform for tracking and managing personal expenditure. The key achievements of the project are:

* Secure user authentication through SHA-256 hashing and AES-256 encryption.
* Robust, efficient AI expense categorization through the OpenAI API.
* Easy-to-use reporting tools that produce sophisticated spending breakdowns.
* Unit-based system architecture that is easy to maintain and expand.

Through this project, we were able to show how OOP concepts, namely encapsulation, abstraction, and aggregation can be effectively used in a real-world system to address security as much as functionality.

**Future Improvements**

While the project is functional, there are several areas where improvements and extensions could be made:

1. **Integration with Banking APIs:**

Integrating live banking APIs (such as Plaid) into the system to auto-retrieve transaction data directly from users' bank accounts without manual entry would improve the model.

1. **Multi-Currency Support:**

Including support for multiple currencies would be a great addition, particularly for international users. A currency exchange API, one up to date with current currency rates can be integrated to exchange expenses into the user's desired currency.

1. **Mobile Application Development:**

The project would be scalable into a mobile app, enabling users to monitor their expenses with ease while on the move. This would necessitate utilizing mobile app frameworks like Flutter for example.

1. **Enhanced Data Visualization:**

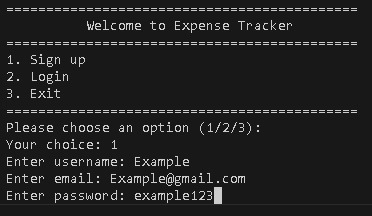
To improve user experience, data visualization tools like pie charts & bar charts can be added to provide pictorial insights.

1. **Cloud Storage and Backup:**

Saving data from local CSV files onto cloud storage would allow users to access their data on various devices and provide a layer of security for data backup.

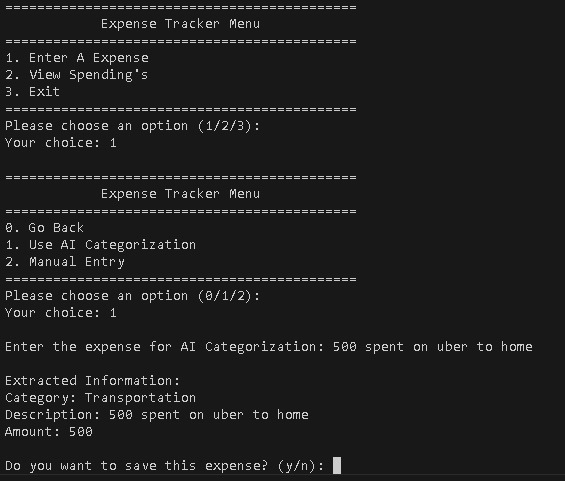
This project sets up the groundwork to develop a more advanced, AI-driven personal finance management tool, and with these enhancements, it can be an even more useful and tailored tool for users who want to manage their finances.

**Screenshots and Outputs1. Signup Screenshot**



*(***Figure 1:** User entering credentials to create a new account. The system securely encrypts the data using SHA-256 hashing and AES-256 encryption before storing.)

**2. Expense Entry Screenshot**

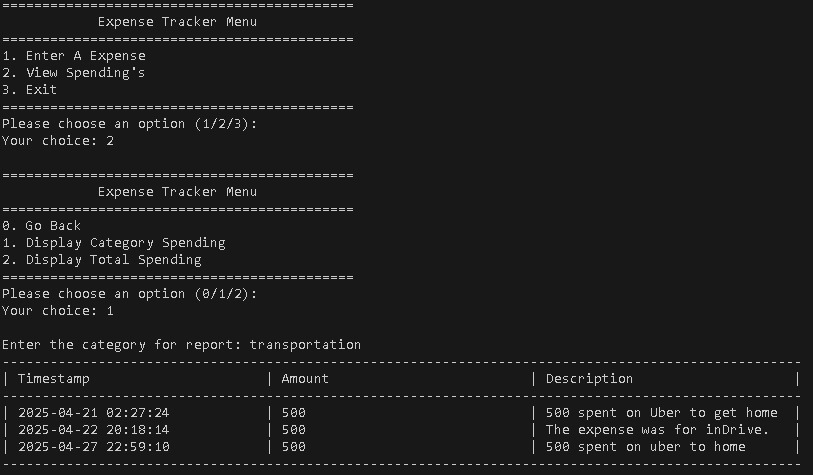


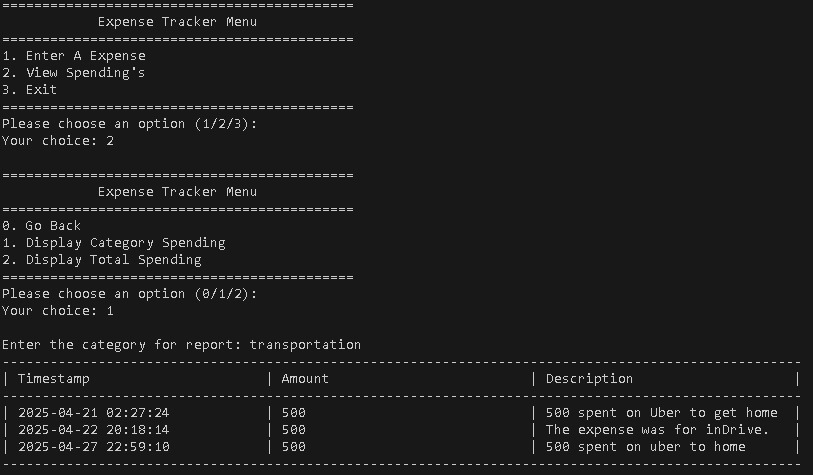
A screenshot of a computer

AI-generated content may be incorrect.

*(***Figure 2:** User entering an expense through AI-based categorization. The system automatically extracts the amount, category, and description from user input using the OpenAI API)

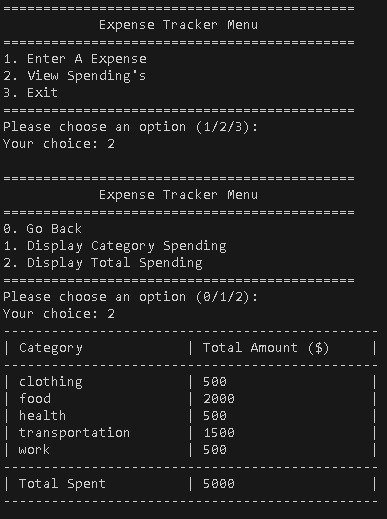
**3. Category Report Output**





(**Figure 3:** Display of a detailed category-wise spending report, showing expenses stored under a specific category from the local CSV file.)

**4. Summary Report Output**



**(Figure 4:** Summary report generated by the system showing total expenses across all categories. This gives the user a consolidated view of their spending patterns.)