Personal Finance Manager

by Group 7 - 3Q

Start Presentation

Introduction

Nowadays, individuals face numerous challenges in managing their finances effectively. From tracking spending to analyzing investment portfolios, financial management can be complex. The Personal Finance Manager system aims to assist individuals in this endeavor by monitoring their transactions and generating reports, offering a clearer view of their monthly financial activities.

Read More

Objectives and Purpose

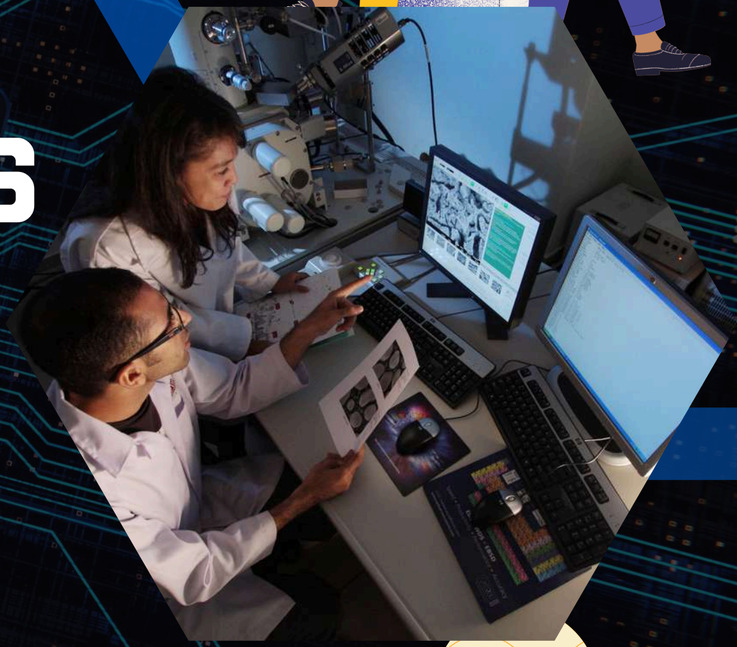
To record the users' expenses of different categories such as food, shopping, transportation and so on.

To help users plan their budget to maintain a positive cash flow.

To record users' income which can be divided into passive income and active income

The project designs

Read More



Understanding Our System

There are some features:

Income Management:

- Users can input their income, both active (e.g., salary) and passive (e.g., investments) income.
- The system calculates the total income, including both active and passive sources.

Expense Tracking:

- Users can enter their expenses categorized into various categories (e.g., food, transport, bills).
- Each expense entry includes the amount spent, date of the expense, and the expense category.
- Users can remove existing expenses from their records if needed.

Understanding Our System

There are some features:

Budget Calculation:

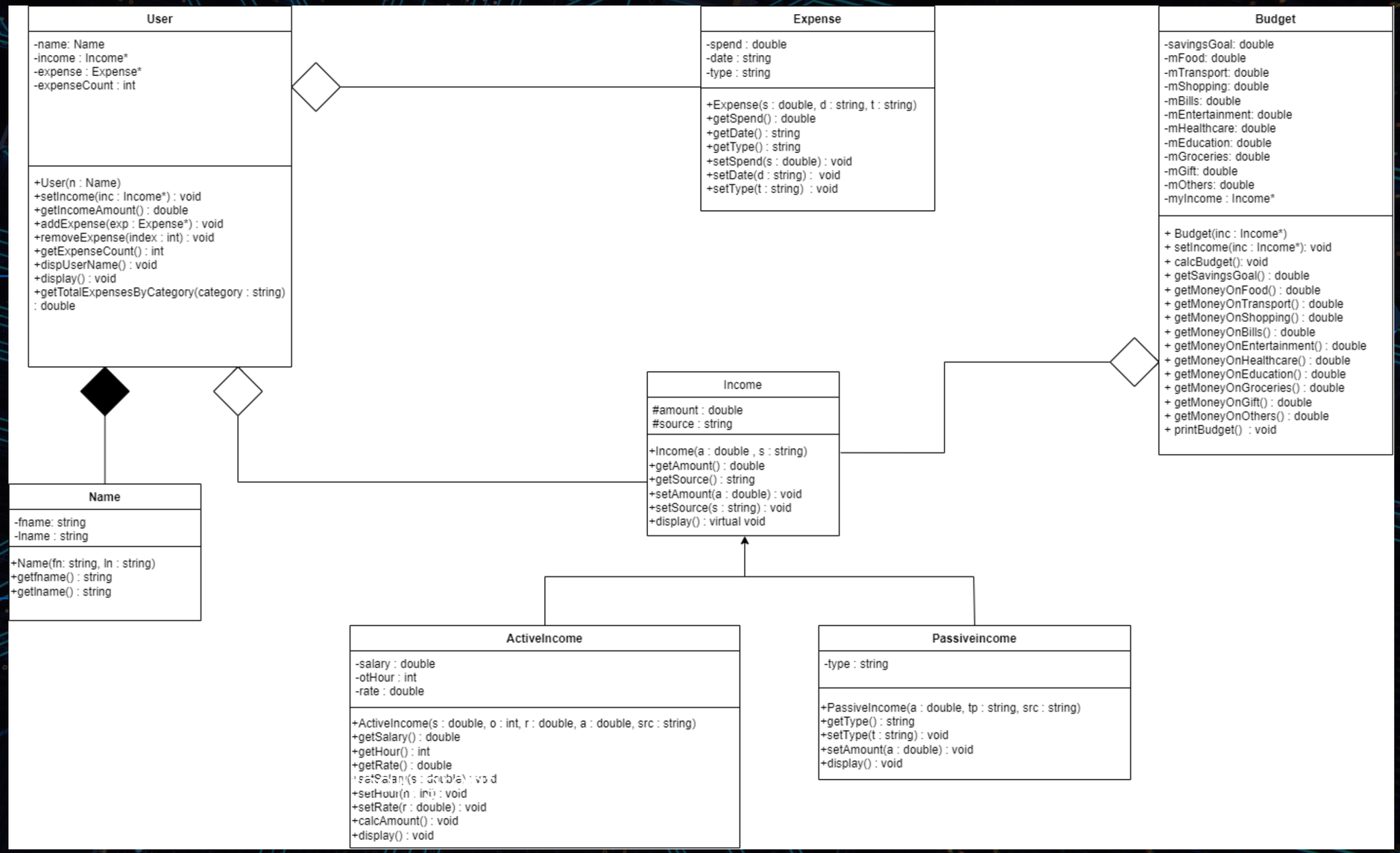
- The system calculates a budget based on the user's income and predefined allocation percentages for different expense categories. For example, we predefined a rate of 10% for food budget
- Users can view their budget summary, including savings goals and allocated amounts for each expense category.

Overall Reporting:

- Users can generate an overall report summarizing their financial activities, including all type of income and expenses.
- The report includes total income, total passive income and total expenses by category.

UML Class Diagram

This shows how the classes are related to each other



Concepts OO Employed

Encapsulation

1. class Expense
- Private attributes:
  - double spend
  - string date
  - string type
- Public methods:
  - Expense(double s, string d, string t)
  - double getSpend()
  - string getDate()
  - string getType()
  - setSpend(double s)
  - setDate(double d)
  - setType(double t)

2. class Name
- Private attributes:
  - Name name
  - Income *income
  - Expense *expense
  - int expenseCount
- Public methods:
  - User(Name n)
  - void setIncome(Income *inc)
  - double getIncomeAmount()
  - void addExpense(Expense *exp)
  - void removeExpense(int index)
  - int getExpenseCount()
  - void dispUserName()
  - void display()
  - doublegetTotalExpenseByCategory(string category)

Concepts OO Employed

Encapsulation

3. class Name
- Private attributes:
  - string fname
  - string lname
- Public methods:
  - Name(string fn, string ln)
  - string getfname()
  - string getlname()

4. class Income
- Protected attributes:
  - double amount
  - string source

- Public methods:
  - Income(double a, string s)
  - double getAmount()
  - string getSource()
  - void setAmount(double a)
  - void setSource(string s)
  - virtual void display()

Concepts OO Employed

Encapsulation

5. class PassiveIncome

- Private attributes:
  - string type

- Public methods:
  - PassiveIncome(double a, string tp, string src)
  - string getType()
  - void setType(string t)
  - void setAmount(double a)
  - void display()

6. class ActiveIncome

- Private attributes:
  - double salary
  - int otHour
  - double rate
- Public methods:
  - ActiveIncome(double s, int o, double r, double a, string src)
  - double getSalary()
  - int getHour()
  - double getRate()
  - void setSalary(double s)
  - void setHour(int h)
  - void setRate(double r)
  - void calcAmount()
  - void display()

Concepts OO Employed

Encapsulation

7. class Budget

- Private attributes:
  - double savingsGoal
  - double mFood
  - double mTransport
  - double mShopping
  - double mBills
  - double mEntertainment
  - double mHealthcare
  - double mEducation
  - double mGroceries
  - double mGift
  - double mOthers
  - Income *myIncome
- Public methods:
  - Budget(Income *inc)
  - void setIncome(Income *inc)
  - void calcBudget()
  - double getSavingsGoal()
  - double getMoneyOnFood()
  - double getMoneyOnTransport()
  - double getMoneyOnShopping()
  - double getMoneyOnBills()
  - double getMoneyOnEntertainment()
  - double getMoneyOnHealthcare()
  - double getMoneyOnEducation()
  - double getMoneyOnGroceries()
  - double getMoneyOnGift()
  - double getMoneyOnOthers()
  - void printBudget()

Concepts OO Employed

Composition

```
class User {
    Name name; //Composition
    Income *income;
    Expense *expense[100];
    int expenseCount;
```

Concepts OO Employed

Aggregation 1) class User and Expense

```
class User {
    Name name;
    Income *income;
    Expense *expense[100]; //Aggregation
    int expenseCount;
```

Concepts OO Employed

Aggregation

2) class User and Income

```
class User {
    Name name;
    Income *income; //Aggregation
    Expense *expense[100];
    int expenseCount;
```

Concepts OO Employed

Aggregation 3) class Budget and Income

```
class Budget {
    double savingsGoal;
    double mFood;
    double mTransport;
    double mShopping;
    double mBills;
    double mEntertainment;
    double mHealthcare;
    double mEducation;
    double mGroceries;
    double mGift;
    double mOthers;
    Income *myIncome; //Aggregation
```

Concepts OO Employed

Inheritance (Parent Class - Income)

```
class Income {
protected:
    double amount;
    string source;
public:
    Income(double a = 0, string s = "");
    double getAmount() const;
    string getSource() const;
    void setAmount(double a);
    void setSource(string s);
    virtual void display() const;
};
```

Concepts OO Employed

Inheritance (Child Class - ActiveIncome)

```
class ActiveIncome : public Income {
    double salary;
    int otHour;
    double rate;
public:
    ActiveIncome(double s = 0, int o = 0, double r = 0, double a = 0.0, string src = "Active Income");
    double getSalary() const;
    int getHour() const;
    double getRate() const;
    void setSalary(double s);
    void setHour(int h);
    void setRate(double r);
    void calcAmount();
    void display() const;
};
```

Concepts OO Employed

Inheritance (Child Class - PassiveIncome)

```
class PassiveIncome : public Income {
    string type;
public:
    PassiveIncome(double a = 0.0, string tp = "", string src = "Passive Income");
    string getType() const;
    void setType(string t);
    void setAmount(double a);
    void display() const;
};
```

Concepts OO Employed

Polymorphism

```
class Income {
protected:
    double amount;
    string source;
public:
    Income(double a = 0, string s = "");
    double getAmount() const;
    string getSource() const;
    void setAmount(double a);
    void setSource(string s);
    virtual void display() const; //Polymorphism
};
```

Concepts OO Employed

Array of Objects

In the main function :

```
int main() {
    Income income;
    ActiveIncome activeIncome;
    PassiveIncome passiveIncome;
    PassiveIncome pasIncome[100]; //Array of Object
```

1) Array to store list of passive income

```
do{
    cout << "What is the passive income type : ";
    cin >> type;
    cout << "Enter passive income amount     : ";
    cin >> value;

    while(value<=0){
        cout << "***Please re-enter your passive income and make sure it is more than 0***\n\n";
        cout << "Enter passive income amount     : ";
        cin >> value;
    }

    pasI[count].setType(type);
    pasI[count].setAmount(value);
    count++;
    total+=value;
    clearScreen();
    cout << "Do you wish to continue entering your passive income? (press Y/y for yes) : ";
    cin >> input;
    cout << endl;
}while(input == 'y' || input == 'Y');
```

Concepts OO Employed

Array of Objects

2) Array to store list of expenses

In the User class :

```
class User {
    Name name;
    Income *income;
    Expense *expense[100];
    int expenseCount;
```

```
void User::addExpense(Expense* exp) {
    if (expenseCount < 100) {
        expense[expenseCount++] = exp;
    } else {
        cout << "Expense limit reached!" << endl;
    }
}
```

In the main function :

```
user.addExpense(new Expense(value, date, category));
```

Concepts OO Employed

Exception Handling

1) Exception to verify the date format

```
try {
    if (isValidDateFormat(date) == false) {
        throw "Invalid date format!\n";
    }
}
catch (const char *msg) {
    cout << msg;
}
```

Concepts OO Employed

Exception Handling

2) Exception to verify the index entered

```
try {
    if (index > 0 && index <= user.getExpenseCount()) {
        user.removeExpense(index - 1);
        throw "Expense removed successfully.\n";
    }
    else
        throw "Invalid index.\n";
}

catch (const char *msg) {
    cout << msg;
}
```

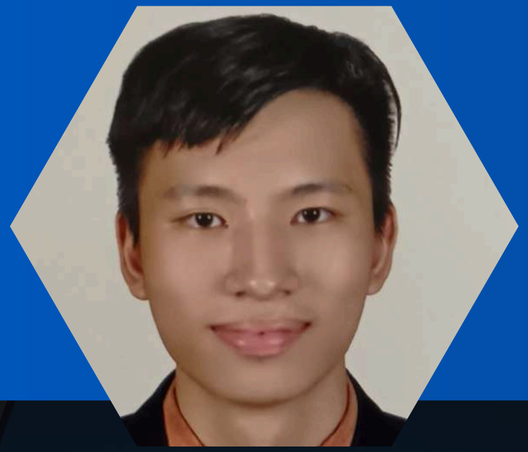
Our Services

Our Personal Finance Manager system provides users with a comprehensive financial management solution, empowering them to track their income, manage expenses, set budgets, and analyze their financial status effectively. With user-friendly features and robust functionality, it aims to simplify financial management and promote financial well-being.

Read More

Home About Contact

Tan Keqin



Chong Lun Quan

Mavis Lim Hui Qing

Meet Our Team

Thank You

For Your Attention