Budget App

Create a console application to help users track income, expenses, and savings. Users can add transactions, categorize spending, track balances, and save/load all financial data between sessions.

Class: Transaction (Base Class)

Stores details about a parent transaction.

ATTRIBUTES:

- double amount – refers to transaction amount
- string date – date of transaction
- string category – transaction category (food, rent, entertainment, etc…)
- string description – transaction description

METHODS:

- virtual void display() - prints transaction details
- virtual string getType() - returns transaction type (“Income” or “Expense”)

*constructors, destructors, and getters/setters are **NOT** explicity listed in **METHODS** but should be implemented and used where needed.

Class: IncomeTransaction (Inherits “Transaction”)

Represents an income transaction.

METHODS:

- void display() - override display() to show details about this income transaction.
- string getType() - overrides getType(), returns “Income”

Class: ExpenseTransaction (Inherits “Transaction”)

Represents an expense transaction.

METHODS:

- void display - override display() to show details about this expense transaction.
- string getType() - overrides getType(), returns “Expense”

*constructors, destructors, and getters/setters are **NOT** explicity listed in **METHODS** but should be implemented and used where needed.

Class: App

Manages all transactions for a single user.

ATTRIBUTES:

- vector<Transaction*> transactions – dynamically allocated transactions

METHODS:

- void addIncome() - prompts user to add an income transaction
- void addExpense() - prompts user to add an expense transaction
- void viewAllTransactions() - displays all transactions
- double calculateBalance() - calculates net balance (income – expense)
- void viewSpendingByCategory() - prompts user for a category and displays total expenses for that category.
- void saveToFile(ofstream& output) (saves all transactions to a file)
- void loadFromFile(ifstream& input) (loads all transactions from a file)

Class: User

Represents a user profile and contains the user's App instance.

ATTRIBUTES:

- string username – User's name
- double savingsGoal – target savings amount
- App myBudget – user's personal budget app instance

*constructors, destructors, and getters/setters are **NOT** explicity listed in **METHODS** but should be implemented and used where needed.

METHODS:

- void display() - displays user info and current balance
- void showSavingsProgress() - shows progress towards savings goal
- void saveUserData(string filename) - saves user info and app data to file
- void loadUserData(string filename) - loads user info and app data from file
- void startSession() - launches main menu for user

What the user can do

```
===========================
USER
===========================
1. Load User Profile
2. Create New User
3. Exit
```

```
===========================
MAIN MENU
===========================
1. Add Income
2. Add Expense
3. View All Transactions
4. View Balance
5. View Spending by Category
6. Show Savings Progress
7. Save
8. Load
9. View User Profile
10. Return to Main Menu
```

*constructors, destructors, and getters/setters are **NOT** explicity listed in **METHODS** but should be implemented and used where needed.

ACTIONS BREAKDOWN

1. Add Income	App::addIncome()
2. Add Expense	App::addExpense()
3. View All Transactions	App::viewAllTransactions()
4. View Balance	App::calculateBalance()
5. View Spending by Category	App::viewSpendingByCategory()
6. Show Savings Progress	User::showSavingsProgress()
7. Save	User::saveUserData()
8. Load	User::loadUserData()
9. View User Profile	User::display()
10. Return to Main Menu	Same as exit

*constructors, destructors, and getters/setters are **NOT** explicity listed in **METHODS** but should be implemented and used where needed.