Clean Code Development

1. **Reducing amount of code in void main() for better view and creating functions instead**

THEN:

```
34  161                 case 1:
35  162                 {
36      -                       double amount;
37      -                       string category, description;
38      -           int categoryChoice;
39      -
40      -                       cout << "Enter expense amount: ";
41      -
42      -                       if (!(cin >> amount))
43      -                       {
44      -                               cout << "Invalid input for amount. Please enter a valid number.\n";
45      -                               break;
46      -                       }
47      -
48      -                       cin.ignore();
49      -
50      -           financeManager.printExpenseCategories();
51      -
52      -           cout << "Choose an expense category (enter the corresponding number): ";
53      -
54      -           if (!(cin >> categoryChoice) || categoryChoice < 1 || categoryChoice > financeManager.getExpenseCategories().size())
55      -           {
56      -               cout << "Invalid category choice. Please enter a valid number.\n";
57      -               break;
58      -           }
59      -
60      -           category = financeManager.getExpenseCategories()[categoryChoice - 1];
61      -
62      -           cin.ignore();
63      -
64      -                       cout << "Enter expense description: ";
65      -                       getline(cin, description);
66      -
67      -                       financeManager.addExpense(amount, category, description);
68      -                       cout << "Expense added successfully.\n";
```

NOW:

```
189         switch (choice)
190         {
191         //adding expence
192         case 1:
193         {
194             adding_Expense(amount, financeManager, categoryChoice, category, d
195             if (retFlag == 2) break;
196             break;
197         }
198
199         //adding income
200         case 2:
201         {
202             adding_income(income, incomeDescription, financeManager, retFlag);
203             if (retFlag == 2) break;
204             break;
205         }
206
```

```
void menu() { ... }

void save_and_exit(FinanceManager& financeManager) { ... }

void monthly_report(int& month, int& year, FinanceManager& financeManager, int

void Balance(FinanceManager& financeManager) { ... }

void adding_income(double& income, std::string& incomeDescription, FinanceMana

void adding_Expense(double& amount, FinanceManager& financeManager, int& categ

void loadUser(User& user) { ... }

void user_creation(User& user, FinanceManager& financeManager) { ... }

void generate_budget_plan(int& month, int& year, double& salary) { ... }

int main()
{
```

2. Explanatory variables

```
163  int main()
164  {
165      User user = User::loadUserFromFile();
166
167      int choice, month, year, categoryChoice, retFlag;
168      double income, amount, salary;
169      string incomeDescription, category, description;
170
```

3. Good comments

```
#include "FinanceManager.h"

//Methods add
void FinanceManager::addExpense(double amount, std::string category, std::stri

void FinanceManager::addIncome(double amount, std::string description) { ... }

//Methods i/o and gen
void FinanceManager::generateMonthlyReport(int month, int year) { ... }

void FinanceManager::saveTransactions() { ... }

void FinanceManager::loadTransactions() { ... }

void FinanceManager::loadExpenseCategories() { ... }

//Methods expense Catergories
void FinanceManager::printExpenseCategories() { ... }

bool FinanceManager::isCategoryValid(std::string category) { ... }

//mathmetods
double FinanceManager::calculateBalance() { ... }
```

```
#pragma once

#include <vector>

#include "User.h"

class FinanceManager
{
private:
    User user;
    std::vector<Transaction> transactions;
    std::vector<std::string> expenseCategories;

public:

    FinanceManager(User user) { ... }

    //Methods add
    void addExpense(double amount, std::string category, std::strin
    void addIncome(double amount, std::string description);

    //Methods i/o and gen
    void generateMonthlyReport(int month, int year);
    void saveTransactions();
    void loadTransactions();
    void loadExpenseCategories();

    const std::vector<std::string>& getExpenseCategories() const {

    //Methods expense Catergories
    void printExpenseCategories();
    bool isCategoryValid(std::string category);

    //mathmetods
    double calculateBalance();
```

4. Error handling

```cpp
void Budgeting::writeToCSV(const std::string& filename, const std::string& data)
{
    std::ofstream file(filename);
    if (file.is_open()) {
        file << data;
        file.close();
    }
    else {
        std::cerr << "Error: Unable to open file " << filename << " for writing." << st
    }
}
```

5. Intuitive Function names and clear intension of the function/method

```cpp
void generate_budget_plan(int& month, int& year, double& salary)
{
    system("cls");
    cout << "Enter month (1-12): ";

    if (!(cin >> month))
    {
        cout << "Invalid input for month. Please enter a valid number.\n";
    }

    cin.ignore();
    cout << "Enter year: ";
    if (!(cin >> year))
    {
        cout << "Invalid input for year. Please enter a valid number.\n";
    }

    cin.ignore();
    cout << "Enter monthly salary: ";
    if (!(cin >> salary))
    {
        cout << "Invalid input for salary. Please enter a valid number.\n";
    }

    system("cls");
    Budgeting budget(month, year, salary);
    budget.generateBudgetPlan();
    budget.printSummary();
}
```