**A**

**PROJECT REPORT**

**ON**

**“EXPENSES MANAGER SYSTEM”**

SUBMITTED BY:

**Miss. Zalte Jayashree Subhash (2124UCEF1065)**

SUBJECT:

**CORE C++**

**PROGRAMMING**

Under the guidance of

**Miss. ISHWARI TRISE**

****

**Department of Computer Science and Engineering**

**Sanjivani Rural Education Society’s**

**SANJIVANI UNIVERSITY**

**KOPARGAON – 423603, DIST : AHMEDNAGAR**

**2024-2025**

**INDEX**

|  |  |  |
| --- | --- | --- |
| SR.  NO | CONTENT | PAGE NO. |
| 1. | INTRODUCTION | 3 |
| 2. | CODE | 4 |
| 3. | OUTPUT | 7 |
| 4. | CONCLUSION | 8 |

**INTRODUCTION**

Expense Manager is launched for Android Expense Manager, the latest and one of best running version in Finance category. It enables users to create expense, list down them based on different categories and check or search for specific entry. GensobrazilCpp has two main classes using object-oriented programming paradigms: one is Expense (which represents an individual expense, containing the attributes name of cost, amount and category) and a second class named as ExpenseManager which performs managing actions for handling a collection of all objects called Expenses. A straightforward system provides clear, helpful insight for people to understand where their money goes.

**CODE**

#include <iostream>

#include <string>

#include <vector>

class Expense {

private:

std::string name;

double amount;

std::string category;

public:

Expense(const std::string& name, double amount, const std::string& category)

: name(name), amount(amount), category(category) {}

std::string getName() const { return name; }

double getAmount() const { return amount; }

std::string getCategory() const { return category; }

friend std::ostream& operator<<(std::ostream& os, const Expense& expense) {

os << "Name: " << expense.name << ", Amount: " << expense.amount

<< ", Category: " << expense.category;

return os;

}

};

class ExpenseManager {

private:

std::vector<Expense> expenses;

public:

void addExpense(const Expense& expense) {

expenses.push\_back(expense);

std::cout << "Expense added successfully.\n";

}

void viewExpenses() const {

if (expenses.empty()) {

std::cout << "No expenses recorded.\n";

return;

}

for (const auto& expense : expenses) {

std::cout << expense << "\n";

}

}

void searchExpense(const std::string& name) const {

bool found = false;

for (const auto& expense : expenses) {

if (expense.getName() == name) {

std::cout << expense << "\n";

found = true;

break;

}

}

if (!found) {

std::cout << "Expense not found.\n";

}

}

};

int main() {

ExpenseManager manager;

manager.addExpense(Expense("Groceries", 150.50, "Food"));

manager.addExpense(Expense("Electricity Bill", 75.25, "Utilities"));

manager.addExpense(Expense("Gym Membership", 50.00, "Fitness"));

std::cout << "\nViewing all expenses:\n";

manager.viewExpenses();

std::cout << "\nSearching for 'Groceries':\n";

manager.searchExpense("Groceries");

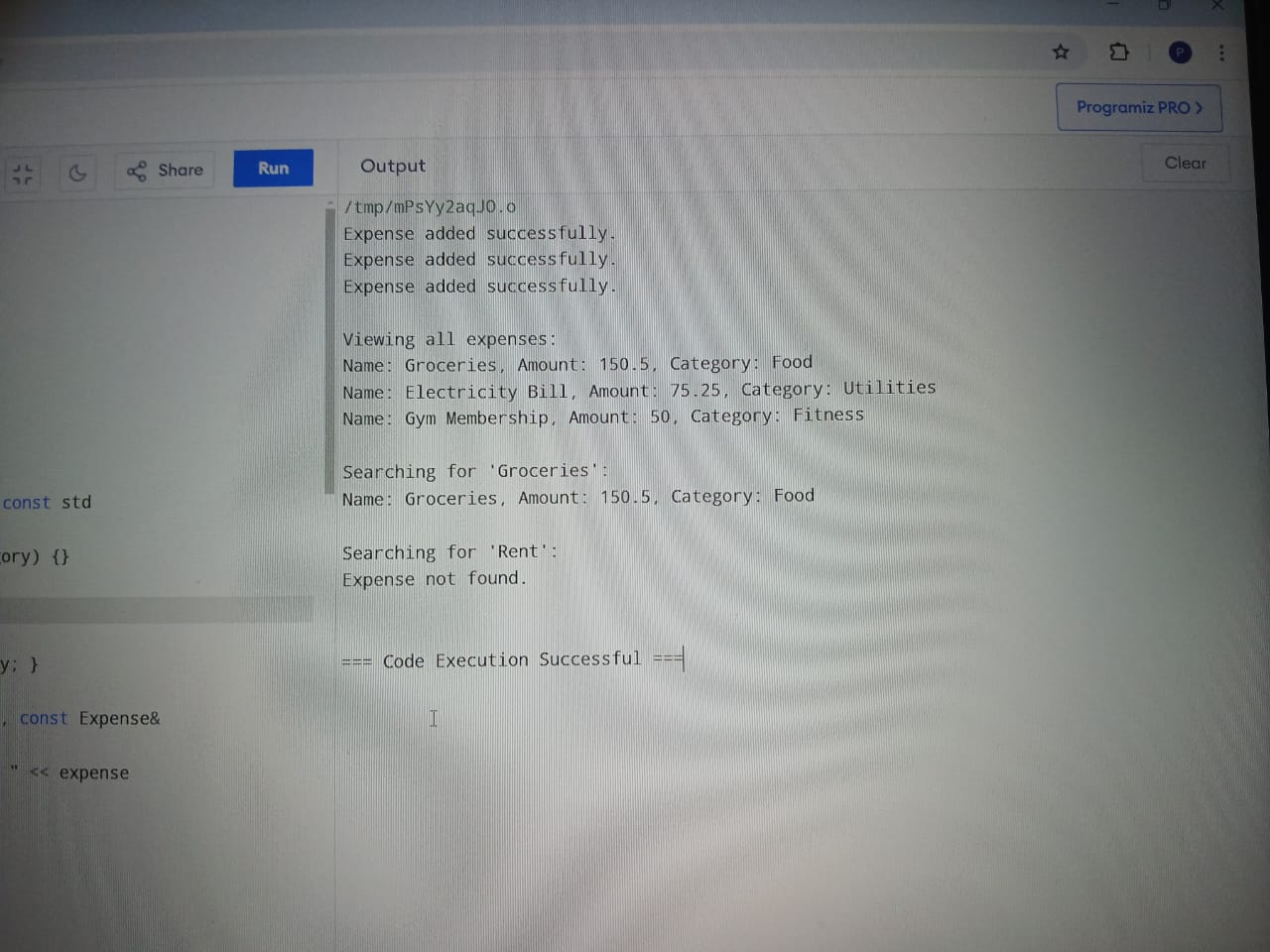
std::cout << "\nSearching for 'Rent':\n";

manager.searchExpense("Rent");

return 0;

}

**OUTPUT**



**CONCLUSION**

To sum up, the Expense Manager is a minimalist solution for keeping track of personal funds. It gives some level of freedom to the users by enabling them add, read and search for expenses which is very conducive in keeping tabs on their spending so far over a period of time or just enough required information to help you make better financial decisions.

This straight forward expense management system shows object-oriented programming principles through the use of classes to structure data and functionalities. The Expense class simply acts as blueprint (an already-defined user-created data type) for expense objects; the Expense Manager, meanwhile, else provides methods and operations that allow us to those define / manipulate expenses. This code allows you to create, see and search for our expenses but if we want more functionalities as categorization or edit an expense, reporting etc this just a start of a basic example tool that anyone could use in order to help him control their financial life.