**BAHRIA UNIVERSITY, Karachi Campus)**

# Department of Software Engineering

# ASSIGNMENT # 03 – Fall 2024

|  |  |  |
| --- | --- | --- |
| Course Title: **Software Construction** |  | Course Code: **SEC-311** |
| Class: **BSE – 5(C)** |  | Shift: **Morning** |
| Course Instructor: **Engr. Misbah Perveen** |  | Date: **2nd Dec 2024** |
| Due Date: **25th Dec 2024** |  | Max. Marks: **10.0 Marks** |
| Name: **Bia Fatima Khan Niazi**  Enrollment: **02-131222-116** |

**Question No. 1: [CLO#04, 10.0 marks]**

You are required to design and implement a small application to demonstrate your understanding of **software construction principles**. Choose **one application** from the list below and develop it with a focus on object-oriented design, code quality, and basic functionality. Based on the last digit of your enrollment number, choose a scenario from the list below.

**List of Applications**

1. Library Management System
2. Student Information System
3. Inventory Management System
4. Hospital Appointment Booking System
5. To-Do List Application
6. Expense Tracker
7. Employee Attendance System
8. Quiz Management System
9. Online Voting System
10. Hotel Reservation System

**Expense Tracker**

1. **Source Code**:

class Expense:

def \_\_init\_\_(self, description, amount, category):

self.description = description

self.amount = amount

self.category = category

def \_\_str\_\_(self):

return f"{self.description}: ${self.amount} ({self.category})"

class ExpenseTracker:

def \_\_init\_\_(self):

self.expenses = []

def add\_expense(self, description, amount, category):

expense = Expense(description, amount, category)

self.expenses.append(expense)

return f"Expense added: {expense}"

def view\_expenses(self):

if not self.expenses:

return "No expenses recorded."

return "\n".join(str(expense) for expense in self.expenses)

def delete\_expense(self, index):

if 0 <= index < len(self.expenses):

removed\_expense = self.expenses.pop(index)

return f"Removed expense: {removed\_expense}"

return "Invalid index."

def categorize\_expenses(self):

categorized = {}

for expense in self.expenses:

if expense.category not in categorized:

categorized[expense.category] = 0

categorized[expense.category] += expense.amount

return categorized

# Example Usage

def main():

tracker = ExpenseTracker()

while True:

print("\nExpense Tracker Menu:")

print("1. Add Expense")

print("2. View Expenses")

print("3. Delete Expense")

print("4. Categorize Expenses")

print("5. Exit")

choice = input("Enter your choice: ")

if choice == "1":

description = input("Enter description: ")

amount = float(input("Enter amount: "))

category = input("Enter category: ")

print(tracker.add\_expense(description, amount, category))

elif choice == "2":

print("\nExpenses:")

print(tracker.view\_expenses())

elif choice == "3":

index = int(input("Enter index of expense to delete: "))

print(tracker.delete\_expense(index))

elif choice == "4":

print("\nCategorized Expenses:")

categorized = tracker.categorize\_expenses()

for category, total in categorized.items():

print(f"{category}: ${total}")

elif choice == "5":

print("Exiting Expense Tracker. Goodbye!")

break

else:

print("Invalid choice. Please try again.")

1. **Documentation:**

## **Application Overview**

The Expense Tracker application provides a simple yet effective solution for managing expenses. Users can dynamically add expenses by entering details such as description, amount, and category. The application allows viewing all recorded expenses, deleting specific entries, and categorizing expenses by type for better financial insights. Built with Object-Oriented Principles, it ensures clean, maintainable, and scalable code. With its intuitive menu-driven interface, the application makes tracking and managing expenses easy for everyone.

## **Features Implemented**

1. **Add Expense:** Allows users to record a new expense with a description, amount, and category.

2. **View Expenses:** Displays a list of all recorded expenses.

3. **Delete Expense:** Enables users to remove an expense by its index.

4. **Categorize Expenses:** Summarizes expenses by category, showing the total amount spent in each.

## **Instructions to Run the Application**

1. Ensure Python is installed on your system.

2. Copy the provided source code into a Python file (e.g., expense\_tracker.py).

3. Open a terminal or command prompt in the directory containing the file.

4. Run the application using the command: python expense\_tracker.py

5. Follow the prompts displayed in the terminal to interact with the application.

1. **Unit Test Evidence**

A unit test for the 'add\_expense' method has been implemented to verify its functionality. Below is an example of the test result:

# Unit Test

def test\_add\_expense():

tracker = ExpenseTracker()

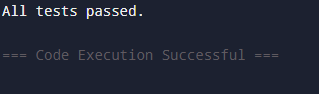
result = tracker.add\_expense("Test", 100, "Miscellaneous")

assert result == "Expense added: Test: $100 (Miscellaneous)", "Test Failed"

print("All tests passed.")

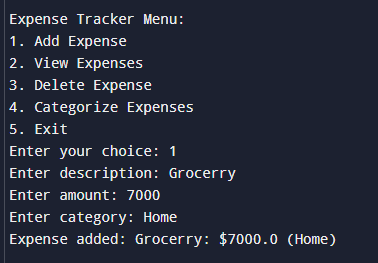
if \_\_name\_\_ == "\_\_main\_\_":

test\_add\_expense()

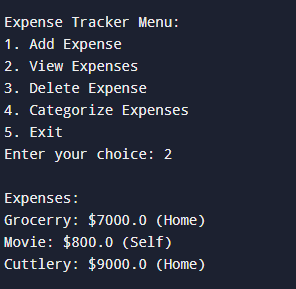


1. **Sample Output:**

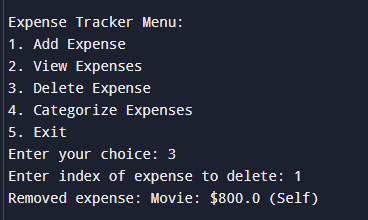
* **Add Expense:**



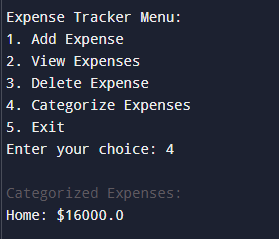
* **View Expense:**



* **Delete Expense:**



* **Categorize Expenses:**



* **Exit:**

