MOTION CUT WEEK 3

1)User Input and Data Management: Develop a system that allows users to input their daily expenses ?

Python Code:

```
import csv
from datetime import datetime

def display_menu():
   print("Expense Tracker")
   print("1. Add Expense")
   print("2. View Expenses")
   print("3. Exit")

def add_expense():
   date = input("Enter the date (YYYY-MM-DD): ")
   description = input("Enter expense description: ")
   amount = float(input("Enter the amount: "))

   with open('expenses.csv', 'a', newline='') as file:
      writer = csv.writer(file)
      writer.writerow([date, description, amount])

   print("Expense added successfully.")

def view_expenses():
   try:
      with open('expenses.csv', 'r') as file:
         reader = csv.reader(file)
         print("\nDate      | Description          | Amount")
         print("------------------------------------------")
         for row in reader:
            date, description, amount = row
            print(f"{date} | {description:<20} | {amount:>7}")
   except FileNotFoundError:
        print("No expense records found.")
    print()

def main():
    while True:
        display_menu()
        choice = input("Enter your choice: ")

        if choice == '1':
            add_expense()
        elif choice == '2':
            view_expenses()
        elif choice == '3':
            print("Exiting the Expense Tracker.")
            break
        else:
            print("Invalid choice. Please try again.")

if _name_ == "_main_":
    main()
```

OUT PUT :

```
Expense Tracker

1. Add Expense
2. View Expenses
3. Exit

Enter your choice: 1
Enter the date (YYYY-MM-DD): 2024-09-02
Enter expense description: Coffee
Enter the amount: 4.50
Expense added successfully.


Enter your choice: 2
```

```
Date      | Description         | Amount
------------------------------------------
2024-09-02 | Coffee             |    4.50
```

```
Enter your choice: 2
No expense records found.
```

```
Enter your choice: 3
Exiting the Expense Tracker.
```

```
2024-09-02, Coffee, 4.50
```

2)Data Storage: Implement a mechanism to store and manage the entered expense data ?

```
import json
from datetime import datetime

class ExpenseTracker:
   def _init_(self):
      self.expenses = []

   def add_expense(self, description, amount):
      date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
      expense = {
         'description': description,
         'amount': amount,
         'date': date
      }
      self.expenses.append(expense)
      print("Expense added successfully.")

   def view_expenses(self):
        if not self.expenses:
            print("No expenses recorded.")
            return
        print(f"{'Description':<20} {'Amount':<10} {'Date':<20}")
        print('-' * 50)
        for expense in self.expenses:
            print(f"{expense['description']:<20}
{expense['amount']:<10} {expense['date']:<20}")

    def save_expenses(self, filename='expenses.json'):
        with open(filename, 'w') as file:
            json.dump(self.expenses, file, indent=4)
        print(f"Expenses saved to {filename}.")

    def load_expenses(self, filename='expenses.json'):
        try:
            with open(filename, 'r') as file:
                self.expenses = json.load(file)
            print(f"Expenses loaded from {filename}.")
        except FileNotFoundError:
            print("No saved expenses found.")

if _name_ == "_main_":
    tracker = ExpenseTracker()
    tracker.load_expenses()

    while True:
        print("\nExpense Tracker")
        print("1. Add Expense")
        print("2. View Expenses")
        print("3. Save Expenses")
        print("4. Load Expenses")
        print("5. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            description = input("Enter description: ")
            amount = float(input("Enter amount: "))
            tracker.add_expense(description, amount)
        elif choice == '2':
            tracker.view_expenses()
        elif choice == '3':
            tracker.save_expenses()
        elif choice == '4':
            tracker.load_expenses()
        elif choice == '5':
            tracker.save_expenses()
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 
5.")
```

OUT PUT:

```
Expense Tracker
1. Add Expense
2. View Expenses
3. Save Expenses
4. Load Expenses
5. Exit
Enter your choice: 1
Enter description: Coffee
Enter amount: 3.50
Expense added successfully.

Expense Tracker
1. Add Expense
2. View Expenses
3. Save Expenses
4. Load Expenses
5. Exit
Enter your choice: 1
Enter description: Lunch
Enter amount: 12.00
```

```
Expense added successfully.

Expense Tracker
1. Add Expense
2. View Expenses
3. Save Expenses
4. Load Expenses
5. Exit
Enter your choice: 2
Description          Amount    Date
--------------------------------------------------
Coffee               3.5     2024-09-02 12:34:56
Lunch               12.0     2024-09-02 12:35:10

Expense Tracker
1. Add Expense
2. View Expenses
3. Save Expenses
4. Load Expenses
5. Exit
Enter your choice: 3
Expenses saved to expenses.json.

Expense Tracker
1. Add Expense
2. View Expenses
3. Save Expenses
4. Load Expenses
5. Exit
Enter your choice: 4
Expenses loaded from expenses.json.

Expense Tracker
1. Add Expense
2. View Expenses
3. Save Expenses
4. Load Expenses
5. Exit
Enter your choice: 2
```

```
Description         Amount    Date
--------------------------------------------------
Coffee              3.5     2024-09-02 12:34:56
Lunch               12.0     2024-09-02 12:35:10

Expense Tracker
1. Add Expense
2. View Expenses
3. Save Expenses
4. Load Expenses
5. Exit
Enter your choice: 5
Expenses saved to expenses.json.
```

3)Expense Categories: Categorize expenses into different categories for better organization ?

Python code:

```python
# Define categories
categories = ["Food", "Transport", "Entertainment", "Utilities",
"Rent", "Other"]

# Initialize expense data
expenses = []

# Function to add an expense
def add_expense(category, amount):
    if category not in categories:
        print(f"Category '{category}' is not valid.")
        return
    expenses.append({'category': category, 'amount': amount})
    print(f"Added {amount} to {category}")

# Function to display expenses by category
def display_expenses():
    expense_summary = {cat: 0 for cat in categories}
    for expense in expenses:
        expense_summary[expense['category']] +=
expense['amount']

    for category, total in expense_summary.items():
        print(f"{category}: ${total:.2f}")

# Main program loop
def main():
    while True:
        print("\nExpense Tracker")
        print("1. Add Expense")
        print("2. View Expenses")
        print("3. Exit")

        choice = input("Choose an option: ")

        if choice == '1':
            category = input("Enter category (Food, Transport,
Entertainment, Utilities, Rent, Other): ")
            try:
                amount = float(input("Enter amount: "))
                add_expense(category, amount)
            except ValueError:
                print("Invalid amount. Please enter a number.")
        elif choice == '2':
            display_expenses()
        elif choice == '3':
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please select a valid option.")

if _name_ == "_main_":
    main()
```

OUT PUT:

```
Expense Tracker
1. Add Expense
2. View Expenses
3. Exit


Choose an option: 1
Enter category (Food, Transport, Entertainment, Utilities, Rent,
Other): Food
Enter amount: 25.50
Added 25.50 to Food


Choosean option: 1
Enter category (Food, Transport, Entertainment, Utilities, Rent,
Other): Transport
Enter amount: 15.75
Added 15.75 to Transport


Choose an option: 1
Enter category (Food, Transport, Entertainment, Utilities, Rent,
Other): Rent
Enter amount: 500.00
Added 500.00 to Rent


Choose an option: 2
Food: $25.50
Transport: $15.75
Entertainment: $0.00
Utilities: $0.00
Rent: $500.00
Other: $0.00

Choose an option: 3
Exiting...
```

4)Data Analysis: Provide users with insights into their spending patterns, such as monthly

summaries and category-wise expenditure ?

PYTHON CODE :

```
import pandas as pd

# Sample data: replace this with your actual data or load from a
file
data = {
   'Date': ['2024-01-15', '2024-01-22', '2024-02-05',
'2024-02-20', '2024-03-10'],
   'Category': ['Food', 'Transport', 'Food', 'Entertainment',
'Food'],
   'Amount': [50, 20, 30, 100, 25]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Convert the 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Add a 'Month' column for easier grouping
df['Month'] = df['Date'].dt.to_period('M')

# Monthly summary
monthly_summary = df.groupby('Month')
['Amount'].sum().reset_index()
monthly_summary.columns = ['Month', 'Total Spent']

# Category-wise expenditure
category_summary = df.groupby('Category')
['Amount'].sum().reset_index()
category_summary.columns = ['Category', 'Total Spent']

# Display the results
print("Monthly Summary:")
print(monthly_summary)

print("\nCategory-wise Expenditure:")
print(category_summary)
```

```
pip install pandas
```

```
python expense_tracker.py
```

OUT PUT :

```
Monthly Summary:
  Month  Total Spent
0  2024-01        70
1  2024-02       120
2  2024-03        25

Category-wise Expenditure:
     Category  Total Spent
0       Food        105
1 Entertainment        100
2   Transport        20
```

5)User-Friendly Interface: Create a user-friendly interface for a seamless user experience ?

PYTHON CODE:

```
import tkinter as tk
from tkinter import messagebox

class ExpenseTracker:
   def _init_(self, root):
       self.root = root
```

```
        self.root.title("Expense Tracker")

        # Create UI components
        self.create_widgets()

    def create_widgets(self):
        # Labels and Entries
        tk.Label(self.root, text="Description").grid(row=0,
column=0, padx=10, pady=10)
        self.description_entry = tk.Entry(self.root,
width=30)
        self.description_entry.grid(row=0, column=1,
padx=10, pady=10)

        tk.Label(self.root, text="Amount").grid(row=1,
column=0, padx=10, pady=10)
        self.amount_entry = tk.Entry(self.root, width=30)
        self.amount_entry.grid(row=1, column=1, padx=10,
pady=10)

        tk.Label(self.root, text="Date").grid(row=2,
column=0, padx=10, pady=10)
        self.date_entry = tk.Entry(self.root, width=30)
        self.date_entry.grid(row=2, column=1, padx=10,
pady=10)

        # Buttons
        tk.Button(self.root, text="Add Expense",
command=self.add_expense).grid(row=3, column=0,
columnspan=2, pady=10)
        tk.Button(self.root, text="Show Expenses",
command=self.show_expenses).grid(row=4,
column=0, columnspan=2, pady=10)

        # Text Area for displaying expenses
        self.expense_list = tk.Text(self.root, height=10,
width=50)
        self.expense_list.grid(row=5, column=0, columnspan=2,
padx=10, pady=10)

        # Data storage
        self.expenses = []

    def add_expense(self):
        description = self.description_entry.get()
        amount = self.amount_entry.get()
        date = self.date_entry.get()

        if not description or not amount or not date:
            messagebox.showwarning("Input Error", "All fields are 
required!")
            return

        try:
            amount = float(amount)
        except ValueError:
            messagebox.showerror("Input Error", "Amount must be a 
number!")
            return

        expense = f"{date} | {description} | ${amount:.2f}"
        self.expenses.append(expense)
        self.description_entry.delete(0, tk.END)
        self.amount_entry.delete(0, tk.END)
        self.date_entry.delete(0, tk.END)
        messagebox.showinfo("Success", "Expense added 
successfully!")

    def show_expenses(self):
        self.expense_list.delete(1.0, tk.END)
        if not self.expenses:
            self.expense_list.insert(tk.END, "No expenses to show.")
        else:
            for expense in self.expenses:
                self.expense_list.insert(tk.END, expense + "\n")

if _name_ == "_main_":
    root = tk.Tk()
    app = ExpenseTracker(root)
    root.mainloop()
```

OUTPUT:

```
2024-09-01 | Lunch | $15.50
```

6)Error Handling: Implement error handling to ensure the application can handle unexpected inputs gracefully ?

PYTHON CODE:

```
import json

# Sample data file path
DATA_FILE = 'expenses.json'

def load_expenses():
    try:
        with open(DATA_FILE, 'r') as file:
            return json.load(file)
    except FileNotFoundError:
        return []
    except json.JSONDecodeError:
        print("Error: Could not decode the data file.")
        return []
    except Exception as e:
        print(f"Unexpected error while loading data: {e}")
        return []

def save_expenses(expenses):
    try:
        with open(DATA_FILE, 'w') as file:
            json.dump(expenses, file, indent=4)
    except IOError:
        print("Error: Could not write to the data file.")
    except Exception as e:
        print(f"Unexpected error while saving data: {e}")

def add_expense(expenses):
    try:
        description = input("Enter the expense description: ")
        amount = float(input("Enter the expense amount: "))
        expenses.append({'description': description, 'amount': 
amount})
        print("Expense added successfully.")
    except ValueError:
        print("Error: Invalid amount entered. Please enter a numeric 
value.")
    except Exception as e:
        print(f"Unexpected error while adding expense: {e}")

def list_expenses(expenses):
    if not expenses:
        print("No expenses recorded.")
        return
    for expense in expenses:
        print(f"Description: {expense['description']}, Amount: 
{expense['amount']}")

def main():
    expenses = load_expenses()

    while True:
        print("\nExpense Tracker")
        print("1. Add expense")
        print("2. List expenses")
        print("3. Exit")

        choice = input("Choose an option: ")

        if choice == '1':
            add_expense(expenses)
            save_expenses(expenses)
        elif choice == '2':
            list_expenses(expenses)
        elif choice == '3':
            break
        else:
            print("Error: Invalid choice. Please select a valid option.")

if _name_ == "_main_":
    main()
```

OUT PUT:

```
Expense Tracker
1. Add expense
2. List expenses
3. Exit
Choose an option: 1
Enter the expense description: Lunch
Enter the expense amount: 15.50
Expense added successfully.

Expense Tracker
1. Add expense
2. List expenses
3. Exit
Choose an option: 2
Description: Lunch, Amount: 15.5

Expense Tracker
1. Add expense
2. List expenses
3. Exit
Choose an option: 3
```

7)Documentation: Document your code effectively to

demonstrate clarity and understanding ?

PYHTON CODE:

```
import json

# Path to the data file where expenses will be saved
DATA_FILE = 'expenses.json'

def load_expenses():
    """
    Load expenses from the JSON file.

    Returns:
        list: A list of expense records. If the file is not found or an
error occurs, returns an empty list.
    """
    try:
        with open(DATA_FILE, 'r') as file:
            return json.load(file)
    except FileNotFoundError:
        return []
    except json.JSONDecodeError:
        print("Error: Could not decode the data file.")
        return []
    except Exception as e:
        print(f"Unexpected error while loading data: {e}")
        return []

def save_expenses(expenses):
    """
    Save the list of expenses to the JSON file.

    Args:
        expenses (list): A list of expense records to be saved.

    Returns:
        None
    """
    try:
        with open(DATA_FILE, 'w') as file:
            json.dump(expenses, file, indent=4)
    except IOError:
        print("Error: Could not write to the data file.")
    except Exception as e:
        print(f"Unexpected error while saving data: {e}")

def add_expense(expenses):
    """
    Add a new expense to the list of expenses.

    Prompts the user for an expense description and amount, then
appends
    the expense to the list.

    Args:
        expenses (list): A list of current expense records.

    Returns:
        None
    """
    try:
        description = input("Enter the expense description: ")
        amount = float(input("Enter the expense amount: "))
        expenses.append({'description': description, 'amount':
amount})
        print("Expense added successfully.")
    except ValueError:
        print("Error: Invalid amount entered. Please enter a numeric
value.")
    except Exception as e:
        print(f"Unexpected error while adding expense: {e}")

def list_expenses(expenses):
    """
    List all the expenses currently in the list.

    Args:
        expenses (list): A list of expense records.

    Returns:
        None
    """
    if not expenses:
        print("No expenses recorded.")
        return
    for expense in expenses:
        print(f"Description: {expense['description']}, Amount:
{expense['amount']}")

def main():
    """
    Main function to run the expense tracker application.

    Provides a menu for the user to add expenses, list expenses, or
exit the program.
    """
    expenses = load_expenses()

    while True:
        print("\nExpense Tracker")
        print("1. Add expense")
        print("2. List expenses")
        print("3. Exit")

        choice = input("Choose an option: ")

        if choice == '1':
            add_expense(expenses)
            save_expenses(expenses)
        elif choice == '2':
            list_expenses(expenses)
        elif choice == '3':
            break
        else:
            print("Error: Invalid choice. Please select a valid option.")
```

```
if _name_ == "_main_":
    main()
```

OUT PUT:

```
Expense Tracker
1. Add expense
2. List expenses
3. Exit
Choose an option: 1
Enter the expense description: Dinner
Enter the expense amount: 25.00
Expense added successfully.

Expense Tracker
1. Add expense
2. List expenses
3. Exit
Choose an option: 2
Description: Dinner, Amount: 25.0

Expense Tracker
1. Add expense
2. List expenses
3. Exit
Choose an option: 3
```