Assignment – 2 Expense Tracker Using Python

Overview:

In this project, you will build a simple expense tracker application that allows users to record daily expenses, categorize them, and generate spending reports. You willmaster Python programming concepts such as data structures, control structures, and functions. Key tasks include user authentication, expense management, and report generation. You will implement features like user log in, expense categorization, and CSV export, gaining real-world Python development experiencewith a focus on error handling, data validation, and user interaction.

Instructions:

- Review the learning materials in Fundamentals of Python Programming
- Carefully read the situation, tasks, actions, and results in sections to graspthe assignment fully
- Complete and submit your assignment through the Learning ManagementSystem (LMS)
- Follow the provided guidelines closely, ensuring your report includes all
- required analyses and interpretations

Situation:

As part of your role in a financial services company, you are tasked with developing a Python project to create a robust expense tracker application for the organization.This application will enable users to record and categorize daily expenses and generate comprehensive reports to analyze spending habits. The system should include features such as user authentication and efficient expense management, providing valuable insights for personal and corporate financial planning.

Task:

As a Python developer, you are tasked to create an expense tracker application that helps users manage and analyze their daily expenses by providing functionalities for expense management, user authentication, and reporting.

Action:

1. Open the Python environment:
    a. Open the Python practice lab and start your work in a blank code file.

2. Create Classes and Methods:
    a. Define a class named **Expense** to represent an expense record
    b. Use the **__init__** method to initialize an expense object with the following attributes:
        i. **expense_id**: A unique identifier for the expense
        ii. **date**: The date of the expense
        iii. **category**: The category of the expense (e.g., food, transportation)
        iv. **description**: A brief description of the expense
        v. **amount**: The amount spent

    c. Define the **__str__** method to return the string representation of the expense object

3. Data Storage:
    a. Create an empty list named **expenses** to store expense records
    b. Define the following functions to manipulate the list:
        i. **add_expense(expense)**: Adds a new expense object to the list
        ii. **update_expense(expense_id, new_expense)**: Updates an existing expense object based on expense_id
        iii. **delete_expense(expense_id)**: Deletes an expense object from the list based on expense_id
        iv. **display_expenses()**: Displays all expense objects in the list

4. User Authentication:
    a. Create a dictionary named **users** with predefined usernames and passwords

b. Define the function **authenticate_user(username, password)**.
This function will:
   i. Check if the provided username exists in the user's dictionary
   ii. Verify if the provided password matches the password in the user dictionary
   iii. Print a success message if authentication is successful; otherwise, print a failure message
   iv. Return True if authentication is successful, otherwise returns False

5. Categorization and Summarization:
   a. Define the function **categorize_expenses()**. This function will:
      i. Create an empty dictionary named categories
      ii. Iterate over each expense in the expenses list
      iii. Add the expense amount to the corresponding category in the categories dictionary
      iv. Return the categories dictionary

   b. Define the function **summarize_expenses()**. This function will:
      i. Initialize a variable total to 0
      ii. Iterate over each expense in the expenses list and add the expense amount to total
      iii. Return the total sum of expenses

6. Functions for Repetitive Tasks:
   a. Define the function **calculate_total_expenses()**:
      i. Use a generator expression to sum the amount of all expenses in the expenses list
      ii. Return the total sum of expenses

   b. Define the function **generate_summary_report()**:
      i. Call categorize_expenses() to get the categorized expenses
      ii. Print the total amount for each category
      iii. Print the total sum of all expenses by calling **calculate_total_expenses()**

7. Simple CLI for Interaction:
   a. Define the function cli () to provide a menu for user interaction:
      i. Print the menu options
      ii. Take user input to select an option
      iii. Write if-Elif conditions to execute the corresponding function based on user input:
         1. Adds a new expense
         2. Updates an existing expense
         3. Deletes an expense
         4. Displays all expenses
         5. Generates a summary report
         6. Exits the application

   b. Call the **authenticate_user()** function before showing the menu to ensure the user is authenticated

8. Run the program and verify the results:
   a. Run the program by executing the file in your Python environment
   b. Follow the prompts and inputs to simulate user interactions and admin functions

Result:

By completing this project, you have developed a functional expense tracker application that supports user authentication, expense management, and reporting. To build a real-world application, you have gained practical experience in applying fundamental Python programming concepts, including data structures, control structures, and functions. This project has prepared you for more advanced projects and professional work in Python development. Save and submit your solution file (.ipynb file format) in LMS.