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 userinteraction

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 report

Action

1. **Open the Python Environment:**
   - Open the Python practice lab and start your work in a blank code file.
2. **Create Classes and Methods:**
   - **Define a class** named `Expense` to represent an expense record.
   - **Use the `__init__` method** to initialize an expense object with the following attributes:
     - `expense_id` : A unique identifier for the expense
     - `date` : The date of the expense
     - `category` : The category of the expense (e.g., food, transportation)
     - `description` : A brief description of the expense
     - `amount` : The amount spent
   - **Define the `__str__` method** to return the string representation of the expense object.
3. **Data Storage:**
   - **Create an empty list** named `expenses` to store expense records.
   - **Define the following functions** to manipulate the list:
     - `add_expense(expense)` : Adds a new expense object to the list
     - `update_expense(expense_id, new_expense)` : Updates an existing expense object based on `expense_id`
     - `delete_expense(expense_id)` : Deletes an expense object from the list based on `expense_id`
     - `display_expenses()` : Displays all expense objects in the list
4. **User Authentication:**
   - **Create a dictionary** named `users` with predefined usernames and passwords.

- **Define the function** `authenticate_user(username, password)`:
  - Check if the provided username exists in the `users` dictionary
  - Verify if the provided password matches the password in the dictionary
  - Print a success message if authentication is successful; otherwise, print a failure message
  - Return `True` if authentication is successful, otherwise return `False`

5. **Categorization and Summarization:**

- **Define the function** `categorize_expenses()`:
  - Create an empty dictionary named `categories`
  - Iterate over each expense in the `expenses` list
  - Add the expense amount to the corresponding category in the `categories` dictionary
  - Return the `categories` dictionary
- **Define the function** `summarize_expenses()`:
  - Initialize a variable `total` to 0
  - Iterate over each expense in the `expenses` list and add the expense amount to `total`
  - Return the total sum of expenses

6. **Functions for Repetitive Tasks:**

- **Define the function** `calculate_total_expenses()`:
  - Use a generator expression to sum the amount of all expenses in the `expenses` list
  - Return the total sum of expenses
- **Define the function** `generate_summary_report()`:
  - Call `categorize_expenses()` to get the categorized expenses
  - Print the total amount for each category
  - Print the total sum of all expenses by calling `calculate_total_expenses()`

7. **Simple CLI for Interaction:**

- **Define the function** `cli()` to provide a menu for user interaction:
  - Print the menu options
  - Take user input to select an option
  - Use `if-elif` conditions to execute the corresponding function based on user input:
    A. Adds a new expense
    B. Updates an existing expense
    C. Deletes an expense
    D. Displays all expenses
    E. Generates a summary report
    F. Exits the application
- **Call the** `authenticate_user()` **function** before showing the menu to ensure the user is authenticated.

8. **Run the Program and Verify the Results:**

- Run the program by executing the file in your Python environment.
- Follow the prompts and inputs to simulate user interactions and admin functions.

ing

In [1]:
```
from collections import defaultdict
import datetime

#Expense enitity
class expenseTracker:
    def __init__(self, expense_id, date, category,description, amount):
        self.expense_id = expense_id
        self.date = date
        self.category = category
        self.description = description
        self.amount = amount
    def __str__(self):
        return(f"{self.expense_id:<5}|\t{self.date:<12}|\t{self.category:<12}|\t{s

#Expense store
expenses=[]

#insert
def add_expense(expense):
    expenses.append(expense)
    print(f"\n{'*'*5}Expense Added successfully{'*'*5}\n")
    return True
#delete
def delete_expense(expense_id):
    for expense in expenses:
        if expense_id == expense.expense_id:
            expenses.remove(expense)
            return True
        else:
            print("Item not found")
#update
def update_expense(expense_id, new_expense):
    for expense in expenses:
        if expense_id == expense.expense_id:
            expense.date = new_expense.date
            expense.category = new_expense.category
            expense.description = new_expense.description
            expense.amount = new_expense.amount
            return True
    return False
#view
def display_expense():
    print(f"\n{'ID':<5}|\t{'Date':<12}|\t{'Category':<12}|\t{'Description':<15}|{'
    print(f"{'-'*70}")
    if len(expenses) <= 0:
        print("No items")
    else:
        for i in expenses:
            print(i)

expense = expenseTracker("101", '20/10/2001', 'travel','bus ride', 10)
expense2 = expenseTracker("102", '20/10/2001', 'travel','bus ride', 10)
```

```
expense3 = expenseTracker("103", '20/10/2001', 'travel','bus ride', 10)

# add_expense(expense)
# add_expense(expense2)
# add_expense(expense3)


#user cred store
user_db = {'username':'admin', 'password':'root'}
#user login
def authenticate_user(username, password):
    if username == user_db['username'] and password == user_db['password']:
        print("Login Successful")
        return True
    else:
        print(f"\n{'!'*5} Login failed {'!'*5}\n")
        return False

#catergory and its expenses
def categorize_expenses():
    #default dict to handle the KeyError while updating the amount
    categories = defaultdict(int)
    for expense in expenses:
        categories[expense.category] += expense.amount
    return categories

#Total amount
def  summarize_expenses():
    total = 0
    for expense in expenses:
        total += expense.amount
    return total

#Total amount
def calculate_total_expenses():
    return sum(expense.amount for expense in expenses)

#generate summary
def generate_summary_report():
    data = dict(categorize_expenses())
    print(f"\n{'Category':<10}| {'Total Expense':>15}")
    print("-"*30)
    for category, expense in data.items():
        print(f"{category:<10}|{expense:>10}")
    print("\nTotal Expense: ", calculate_total_expenses())


def cli():
    while True:
        print(f"\n\n{'-'*5}Menu{'-'*5}")
        print('''\n1. Add an expense
              \n2. Update an expense
              \n3. Delete expense
              \n4. Displays all expenses
              \n5. Generate summary report
              \n6. Logout and Exit''')
        ch = int(input("\nEnter choice: "))

        if ch == 1:
            expense_id = input("Enter expense_id: ")
```

```
            found = False
            for expense in expenses:
                if expense_id == expense.expense_id:
                    found = True
            if found:
                print(f"\n{'!'*5} Id already exist {'!'*5}")
            else:
                date = input("Enter date in DD/MM/YYYY format: ")
                category = input("Enter category: ")
                description = input("Enter description: ")
                while True:
                    try:
                        amount = int(input("Enter amount: "))
                        expense = expenseTracker(expense_id, date, category, descr
                        if add_expense(expense):
                            break
                    except ValueError as e:
                        print(f"\n{'!'*5} Invalid price format {'!'*5}\n")

        elif ch == 2:
            expense_id = input("Enter expense_id: ")
            found = False
            for expense in expenses:
                if expense_id == expense.expense_id:
                    found = True
            if found:
                date = input("Enter date in DD/MM/YYYY format: ")
                category = input("Enter category: ")
                description = input("Enter description: ")
                while True:
                    try:
                        amount = int(input("Enter amount: "))
                        new_expense = expenseTracker(expense_id, date, category, c
                        if not update_expense(expense_id, new_expense):
                            print(f"\n{'!'*5} Expense not found {'!'*5}")
                        else:
                            print(f"\n{'-'*5} Expense updated successfully {'-'*5}
                    except ValueError as e:
                        print(f"\n{'!'*5} Invalid price format {'!'*5}\n")
                    break
            else:
                print(f"\n{'!'*5} Expense not found {'!'*5}")
        elif ch == 3:
            found = False
            expense_id = input("Enter expense id: ")
            for expense in expenses:
                if expense.expense_id == expense_id:
                    found = True
            if found:
                if delete_expense(expense_id):
                    print(f"\n{'*'*5}Expense deleted successfully{'*'*5}")
            else:
                print(f"\n{'*'*5}Expense not found{'*'*5}")
        elif ch == 4:
            display_expense()
        elif ch == 5:
            generate_summary_report()
        elif ch == 6:
            print("\nLogging out and exiting applicaiton...\n")
            break
```

```
        else:
            print("\nInvalid choice")
#Login
def login():
    print(f"\n{'-'*5}LOGIN{'-'*5}\n")
    username = input("Enter username: ")
    password = input("Enter password: ")
    if authenticate_user(username, password):
        cli()

if __name__ == '__main__':
    login()
```

```
-----LOGIN-----

Login Successful


-----Menu-----

1. Add an expense
2. Update an expense
3. Delete expense
4. Displays all expenses
5. Generate summary report
6. Logout and Exit
ID  |  Date  |  Category  |  Description  |  Amount
--------------------------------------------------------------------
No items


-----Menu-----

1. Add an expense
2. Update an expense
3. Delete expense
4. Displays all expenses
5. Generate summary report
6. Logout and Exit
```

```
*****Expense Added successfully*****


-----Menu-----

1. Add an expense

2. Update an expense

3. Delete expense

4. Displays all expenses

5. Generate summary report

6. Logout and Exit
ID   |  Date        |   Category    |   Description    |  Amount
---------------------------------------------------------------------
101  |  17/08/2024  |   Food        |   Lunch          |  100


-----Menu-----

1. Add an expense

2. Update an expense

3. Delete expense

4. Displays all expenses

5. Generate summary report

6. Logout and Exit
----- Expense updated successfully -----


-----Menu-----

1. Add an expense

2. Update an expense

3. Delete expense

4. Displays all expenses

5. Generate summary report

6. Logout and Exit
```

```
Category   |   Total Expense
-----------------------------
Food       |       500

Total Expense:  500


-----Menu-----

1. Add an expense

2. Update an expense

3. Delete expense

4. Displays all expenses

5. Generate summary report

6. Logout and Exit
*****Expense Added successfully*****



-----Menu-----

1. Add an expense

2. Update an expense

3. Delete expense

4. Displays all expenses

5. Generate summary report

6. Logout and Exit
ID    |  Date          |   Category     |   Description     |  Amount
-------------------------------------------------------------------------
101   |  17/08/2024    |   Food         |   Dinner          |  500
102   |  16/08/2024    |   Petrol       |   Trip to ust     |  150


-----Menu-----

1. Add an expense

2. Update an expense

3. Delete expense

4. Displays all expenses

5. Generate summary report

6. Logout and Exit
```

```
Category   |   Total Expense
-----------------------------
Food       |       500
Petrol     |       150

Total Expense:  650


-----Menu-----

1. Add an expense

2. Update an expense

3. Delete expense

4. Displays all expenses

5. Generate summary report

6. Logout and Exit
*****Expense deleted successfully*****


-----Menu-----

1. Add an expense

2. Update an expense

3. Delete expense

4. Displays all expenses

5. Generate summary report

6. Logout and Exit
Logging out and exiting applicaiton...
```