**Lab Assignment 11: User Management Automation**

You are developing a command-line task management system for a small team of users.

**User Management:**

* Implement a user registration system where users can sign up and log in.
* Store user data in a file, including usernames and hashed passwords.

Code –

*import* hashlib

*import* os

def hash\_password(*password*):

*return* hashlib.sha256(password.encode()).hexdigest()

def register():

    username = input("Enter username: ")

    password = input("Enter password: ")

*with* open("users.txt", "a") *as* f:

        f.write(f"{username},{hash\_password(password)}\n")

    print("Registration successful.")

def login():

    username = input("Enter username: ")

    password = input("Enter password: ")

*with* open("users.txt", "r") *as* f:

*for* line *in* f:

            stored\_user, stored\_hash = line.strip().split(",")

*if* stored\_user == username and stored\_hash == hash\_password(password):

                print("Login successful.")

*return*

    print("Invalid username or password.")

def main():

*while* True:

        choice = input("1. Register\n2. Login\n3. Exit\nChoose an option: ")

*if* choice == "1":

            register()

*elif* choice == "2":

            login()

*elif* choice == "3":

*break*

*if* not os.path.exists("users.txt"):

    open("users.txt", "w").close()

main()

Output –

A screenshot of a computer screen

Description automatically generated

**Lab Assignment 12: Household Expenses Tracker**

You have been tasked with creating a Python program to help manage household expenses. The program should allow family members to input their daily expenses, store them in a CSV file, and provide functionalities for analysis and reporting.

**1. Expense Logging:**  
Create a Python program that allows users to input their daily expenses. The program should prompt the user for their name, date of the expense, description, and amount spent. The data should be stored in a CSV file named expenses.csv with columns **'Name'**, **'Date'**, **'Description'**, and **'Amount'**.

**2. Expense Analysis:**  
Develop a function that reads the expenses.csv file and calculates the total expenses for each family member. Display the total expenses for each member along with the average daily expense for the household.

**3. Expense Trends:**  
Implement a feature that generates a line chart using a plotting library (e.g., Matplotlib) to visualize the expense trends over the last month. The x-axis should represent the dates, and the y-axis should show the cumulative expenses for each day.

**4. Expense Categorization:**  
Enhance the program to allow users to categorize their expenses. Prompt the user to assign a category (e.g., groceries, utilities, entertainment) to each expense entry. Update the CSV file to include a **'Category'** column.

**5. Expense Reporting:**  
Create a monthly expense report by reading the data from expenses.csv and generating a report that includes the following:

* Total expenses for each family member for the month.
* A breakdown of expenses by category.
* A comparison of monthly expenses over different months using bar charts.

**6. Expense Budgeting:**  
Add an option for users to set a monthly budget for each category. After entering expenses, the program should calculate the remaining budget for each category and provide a warning if the budget is exceeded.

**7. Data Backup and Restore:**  
Implement a backup and restore feature that allows users to save a copy of the expenses.csv file to a backup location and restore it if needed. Handle cases where the file might be missing or corrupted.

Code –

*import* csv

*import* os

*from* datetime *import* datetime

*import* matplotlib.pyplot *as* plt

*import* pandas *as* pd

def log\_expense():

    name = input("Enter your name: ")

    date = input("Enter the date (YYYY-MM-DD): ")

    description = input("Enter description: ")

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

    category = input("Enter category (e.g., groceries, utilities, entertainment): ")

*with* open("expenses.csv", "a", *newline*="") *as* f:

        writer = csv.writer(f)

        writer.writerow([name, date, description, amount, category])

def analyze\_expenses():

    expenses = pd.read\_csv(

        "expenses.csv", *names*=["Name", "Date", "Description", "Amount", "Category"]

    )

    total\_expenses = expenses.groupby("Name")["Amount"].sum()

    avg\_daily\_expense = expenses["Amount"].sum() / len(expenses["Date"].unique())

    print("Total expenses per member:")

    print(total\_expenses)

    print(f"Average daily expense for the household: {avg\_daily\_expense:.2f}")

def show\_expense\_trends():

    expenses = pd.read\_csv(

        "expenses.csv", *names*=["Name", "Date", "Description", "Amount", "Category"]

    )

    expenses["Date"] = pd.to\_datetime(expenses["Date"])

    monthly\_expenses = (

        expenses.groupby(expenses["Date"].dt.date)["Amount"].sum().cumsum()

    )

    plt.plot(monthly\_expenses.index, monthly\_expenses.values)

    plt.xlabel("Date")

    plt.ylabel("Cumulative Expenses")

    plt.title("Expense Trends Over the Last Month")

    plt.xticks(*rotation*=45)

    plt.show()

def generate\_expense\_report():

    expenses = pd.read\_csv(

        "expenses.csv", *names*=["Name", "Date", "Description", "Amount", "Category"]

    )

    monthly\_expenses = expenses.groupby("Name")["Amount"].sum()

    category\_expenses = expenses.groupby("Category")["Amount"].sum()

    print("Monthly Expenses per Member:")

    print(monthly\_expenses)

    print("Expenses by Category:")

    print(category\_expenses)

    months = expenses["Date"].apply(lambda *x*: *x*[:7]).unique()

    monthly\_totals = [

        expenses[expenses["Date"].str.startswith(month)]["Amount"].sum()

*for* month *in* months

    ]

    plt.bar(months, monthly\_totals)

    plt.xlabel("Month")

    plt.ylabel("Total Expenses")

    plt.title("Monthly Expenses Comparison")

    plt.show()

def set\_budget():

    budgets = {}

*with* open("budgets.csv", "a+", *newline*="") *as* f:

        f.seek(0)

        reader = csv.reader(f)

        budgets = {rows[0]: float(rows[1]) *for* rows *in* reader}

*while* True:

            category = input("Enter category to set budget (or 'done' to finish): ")

*if* category == "done":

*break*

            budget = float(input(f"Enter budget for {category}: "))

            budgets[category] = budget

            writer = csv.writer(f)

            writer.writerow([category, budget])

def check\_budget():

    expenses = pd.read\_csv(

        "expenses.csv", *names*=["Name", "Date", "Description", "Amount", "Category"]

    )

    category\_totals = expenses.groupby("Category")["Amount"].sum()

    budgets = pd.read\_csv("budgets.csv", *names*=["Category", "Budget"])

*for* \_, row *in* budgets.iterrows():

        category = row["Category"]

*if* category in category\_totals and category\_totals[category] > row["Budget"]:

            print(f"Warning: Budget exceeded for {category}")

def backup\_data():

*if* os.path.exists("expenses.csv"):

        timestamp = datetime.now().strftime("%Y%m%d%H%M%S")

        os.rename("expenses.csv", f"backup\_expenses\_{timestamp}.csv")

        print("Backup completed.")

*else*:

        print("No data to backup.")

def restore\_data():

    backups = [file *for* file *in* os.listdir() *if* file.startswith("backup\_expenses\_")]

*if* backups:

        latest\_backup = sorted(backups)[-1]

        os.rename(latest\_backup, "expenses.csv")

        print("Data restored from latest backup.")

*else*:

        print("No backup file found.")

def main():

*if* not os.path.exists("expenses.csv"):

        open("expenses.csv", "w").close()

*if* not os.path.exists("budgets.csv"):

        open("budgets.csv", "w").close()

*while* True:

        choice = input(

            "1. Log Expense\n2. Analyze Expenses\n3. Show Expense Trends\n4. Generate Expense Report\n5. Set Budget\n6. Check Budget\n7. Backup Data\n8. Restore Data\n9. Exit\nChoose an option: "

        )

*if* choice == "1":

            log\_expense()

*elif* choice == "2":

            analyze\_expenses()

*elif* choice == "3":

            show\_expense\_trends()

*elif* choice == "4":

            generate\_expense\_report()

*elif* choice == "5":

            set\_budget()

*elif* choice == "6":

            check\_budget()

*elif* choice == "7":

            backup\_data()

*elif* choice == "8":

            restore\_data()

*elif* choice == "9":

*break*

main()

Output –

A screenshot of a computer

Description automatically generated A black background with white text

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A blue square with numbers and a white background

Description automatically generatedA screenshot of a computer

Description automatically generated 