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 –

```
1. Register
2. Login
3. Exit
Choose an option: 1
Enter username: aryanranderiya
Enter password: password
Registration successful.
1. Register
2. Login
3. Exit
Choose an option: 2
Enter username: aryanranderiya
Enter password: password
Login successful.
1. Register
2. Login
3. Exit
Choose an option: 3
```

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().cums
um()
    )
    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"]
```

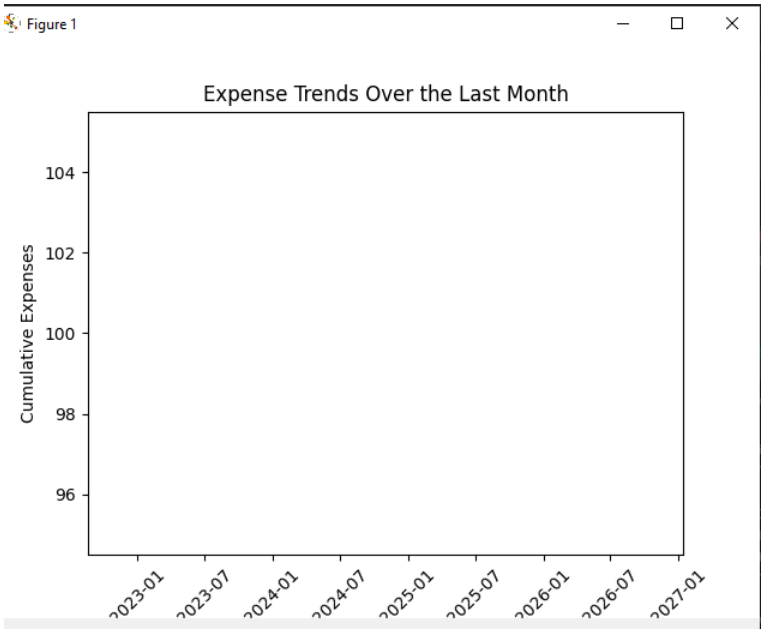
```
    )
    category_totals = expenses.groupby("Category")["Amount"].sum()
    budgets = pd.read_csv("budgets.csv", names=["Category", "Budget"])
    for _, row in budgets.iterrows():
        category = row["Category"]
        if category in category_totals and category_totals[category] >
row["Budget"]:
            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()
```

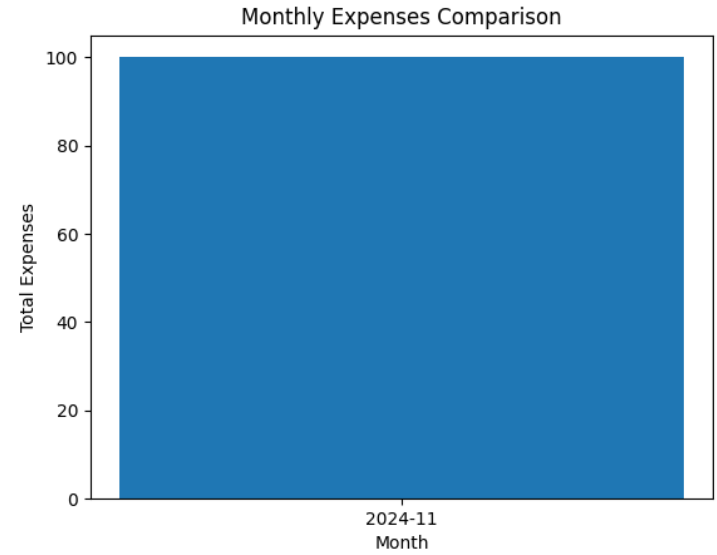
```
        elif choice == "5":
            set_budget()
        elif choice == "6":
            check_budget()
        elif choice == "7":
            backup_data()
        elif choice == "8":
            restore_data()
        elif choice == "9":
            break


main()
```

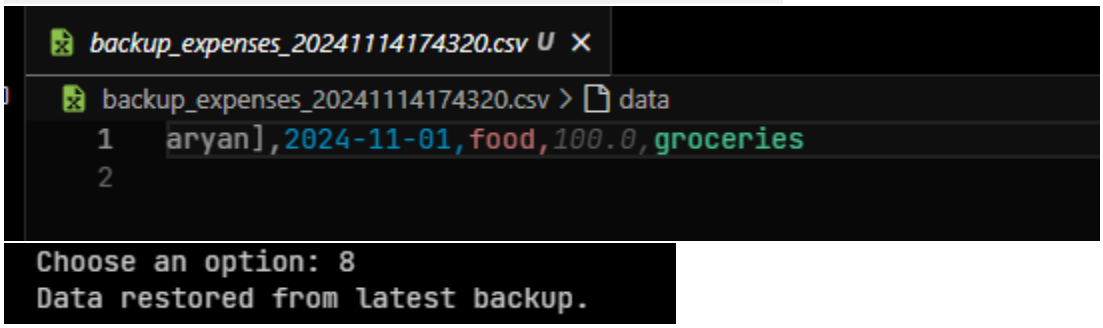
Output –

```
1. Log Expense
2. Analyze Expenses
3. Show Expense Trends
4. Generate Expense Report
5. Set Budget
6. Check Budget
7. Backup Data
8. Restore Data
9. Exit
Choose an option: 1
Enter your name: aryan]
Enter the date (YYYY-MM-DD): 2024-11-01
Enter description: food
Enter amount: 100
Enter category (e.g., groceries, utilities, entertainment): groceries
```

```
Choose an option: 2
Total expenses per member:
Name
aryan]    100.0
Name: Amount, dtype: float64
Average daily expense for the household: 100.00
```








```
Choose an option: 8
Data restored from latest backup.
```