Bill Sharing Application

Problem Definition:

To create a bill sharing application (like splitwise)

The application should be able to provide the following features:

- The application must have a concept of a group - which is basically a collection of registered users. A registered user is known by his/her unique name. A registered user can belong to multiple groups. All users are registered users.
- A person should be able to add bills to the application with the following information:
    - Group to which that bill get added

  The application should keep track of all such bills. Once is bill is added that is splitted evenly across all the members of that group.

To summarise, we expect the following features

- Ability to create a group
- Add persons to a particular group
- Ability to add bills to group given a group_name and list of person_name with amount_paid
- Given a group_name and person_name output the balance of a person
- Given a group_name output the balances of all the users
- **Give the balance of person across all groups**

Requirements:

Expectations:

1. Code should be demo-able (Most Important). Either by using a main driver program or test cases.
2. Should support this using in-memory DS constructs, use of DB not allowed.
3. Create the sample data yourself. You can put it into a file, test case or main driver progran itself.
4. Code should be modular, extensible and should have follow OO design.
5. Code should be extensible.
6. Code should be able to handle edge cases properly and fail gracefully.
7. Code should be legible, readable and DRY(Don't Repeat Yourself).

Guidelines:

1. Please discuss the solution with the interviewer.
2. Please do not access internet for anything EXCEPT syntax.
3. You are free to use the language of your choice.
4. All work should be of your

1) CREATE_GROUP : Input : (<group_name>)
   Sample : CREATE_GROUP("Nirma")

2) ADD_PERSONS : Input : (<group_name>,<list of persons>)
   Sample : ADD_PERSONS ("Nirma",["Hema","Rekha","Jaya","Sushma"])

3) ADD_TRANSACTION(<group name>,<List<(person1,amount paid)>>)
   Sample : ADD_TRANSACTION ("Nirma",[("Hema",500),("Rekha",200)])

4) BALANCE_OF_PERSON_IN_GROUP(<group_name>,<person_name>)
   Sample : BALANCE_OF_PERSON_IN_GROUP : ("Nirma","Jaya")

5) BALANCE_OF_PERSONS_IN_GROUP(<group_name>)
   Sample : BALANCE_OF_PERSONS_IN_GROUP : ("Nirma")

6) BALANCE_OF_PERSON_ACROSS_ALL_GROUPS(<person_name>)
   Sample : BALANCE_OF_PERSON_ACROSS_ALL_GROUPS("Rekha")

Test-cases :

1. CREATE_GROUP("FLIPKART_PARTY")
2. ADD_PERSONS("FLIPKART_PARTY", ["Ashish", "Devbrat", "Anmol", "Abhishek"])
3. CREATE_GROUP("SCHOOL_FRIENDS")
4. ADD_PERSONS("SCHOOL_FRIENDS",["Ashish","Rohit"])
5. ADD_TRANSACTION(("FLIPKART_PARTY",[("Ashish",40),("Devbrat",160)]))
6. ADD_TRANSACTION(("FLIPKART_PARTY",[("Ashish",30),("Abhishek",60),("Anmol",110)
   ]))
7. ADD_TRANSACTION(("SCHOOL_FRIENDS",[("Ashish",100),("Rohit",60))
8. BALANCE_OF_PERSON_IN_GROUP("FLIPKART_PARTY","Ashish")

   Output : -30

   Explanation : Ashish paid 70(40+30), Devbrat paid 160,Anmol paid 110,Abhishek paid 60 . Total amount paid is 400 . Share per person is 100 . Hence, Ashish owes 30(100 - 70 =)
9. BALANCE_OF_PERSONS_IN_GROUP("FLIPKART_PARTY")

   Output :

   Ashish : -30

   Devbrat : 60

   Anmol : 10

   Abhishek: -40
10. BALANCE_OF_PERSON_ACROSS_GROUPS("Ashish")

    Output :

    Ashish : -10

    Explanation : Ashish's balance in FLIPKART_PARTY is -30 and in SCHOOL_FRIENDS is 20 . Hence, Ashish owes 10 (-30+20)