Splitwise (Sharing App)

Low Level Design

Rough Flow:

user
└─ Opens App →

Splitwise

- List of Friends
- list of groups
- Total amount
  spent:
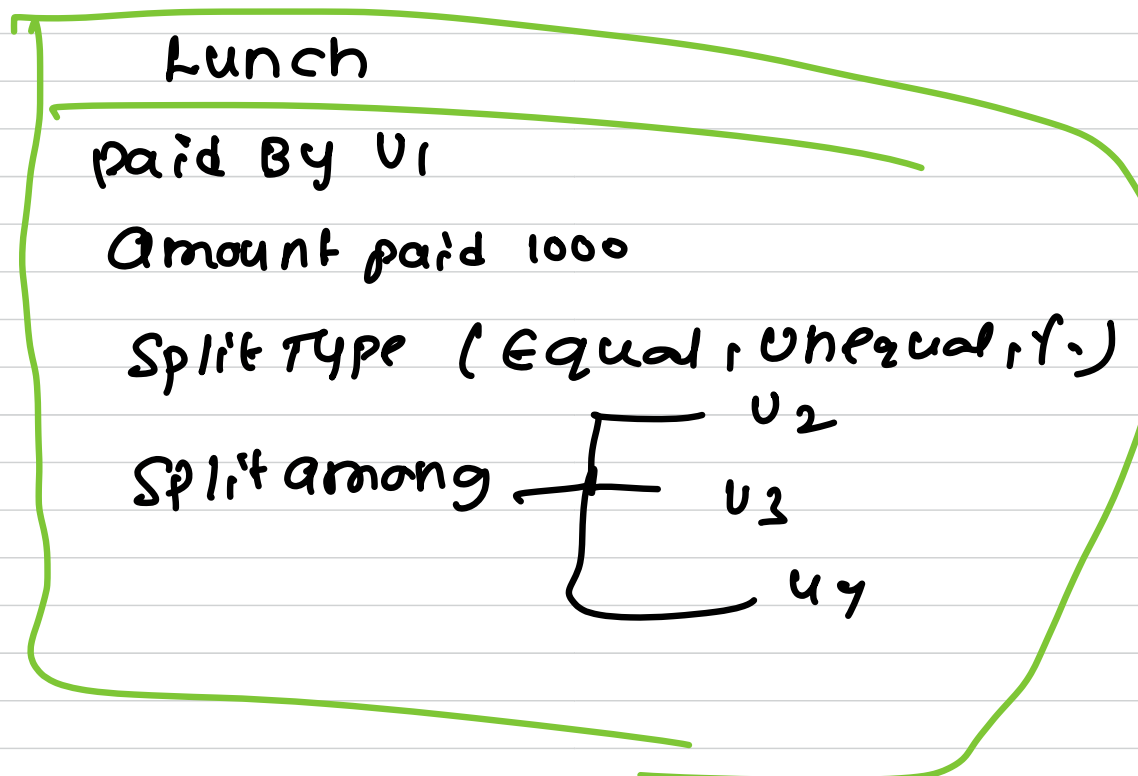 Owe:
  getback:
  expenses:
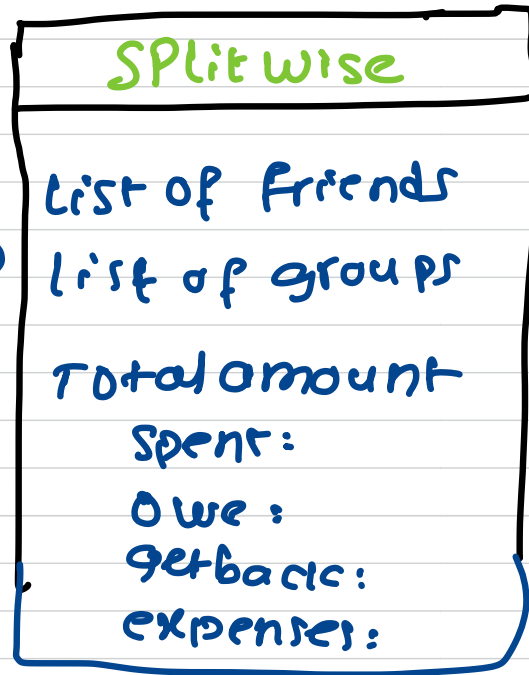
All this info appears on the screen.

Working:

Adding Expense:

U1 paid 1000/- for Lunch

Lunch
Paid By U1
Amount paid 1000
Split Type (Equal, Unequal, ...)
Split among
├─ U2
├─ U3
└─ U4

SO1
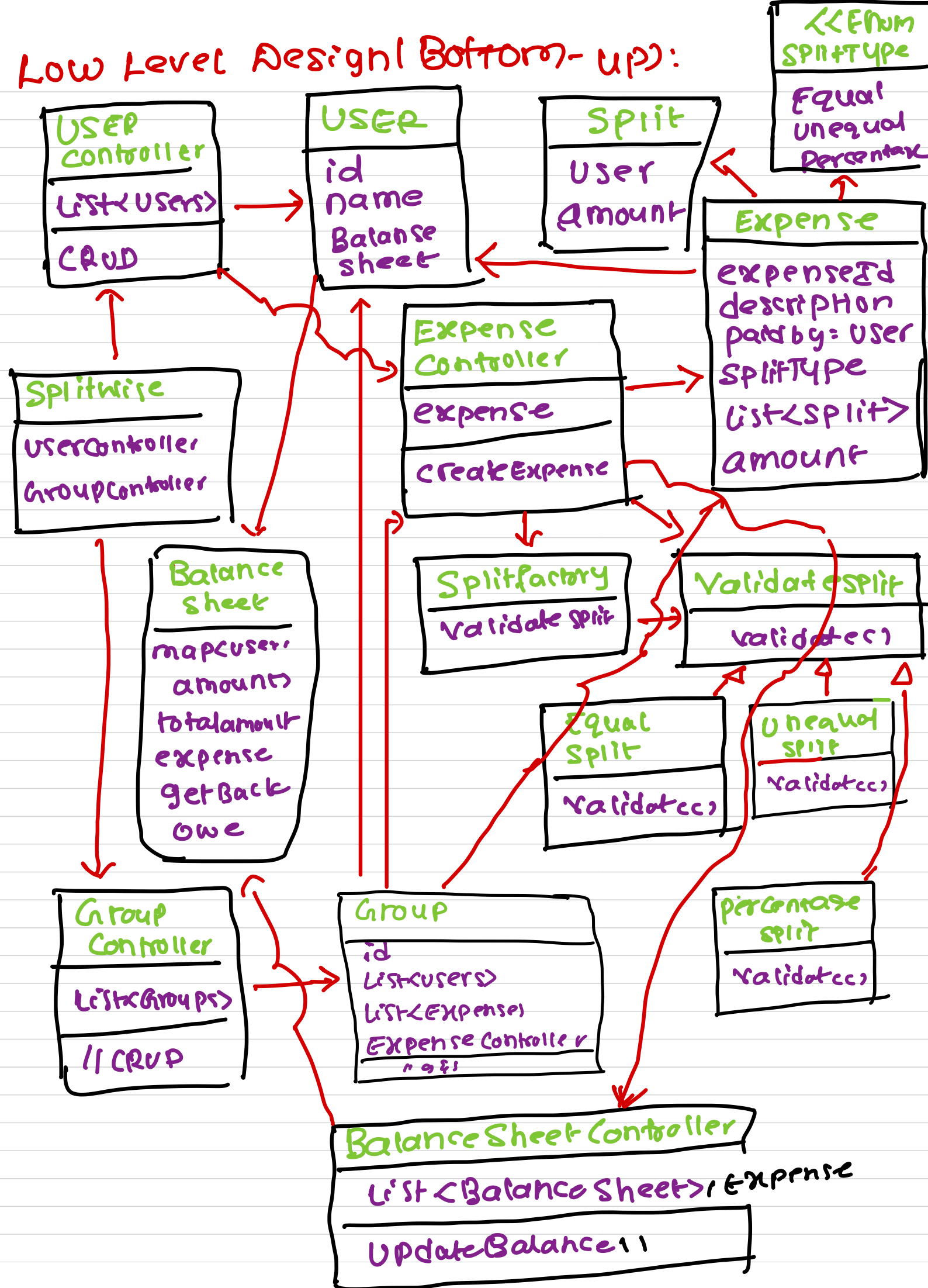
U1 opens splitwise add expenses which is split among friends.

Requirements Gathering:

1) Add a friend (CRUD)
2) Add a group (CRUD)
3) Add an expense to a friend
4) Add a friend to a group
5) Add an expense to a group
6) Expense Spliting Types
    a) Split Equally
    b) Split Unequally
   c) % split
7) Update individual balance sheet

OBJECTS:

1) User
2) Group
3) Expense
4) Balance Sheet
5) Split
6) Splitwise

Low Level Design(Bottorm-up):

USER Controller
- List<Users>
- CRUD

USER
- id
- name
- Balance sheet

Split
- User
- amount

<<Enum>> SPlitType
- Equal
- Unequal
- Percentage

Expense
- expenseId
- description
- paidby: User
- SPLitType
- List<Split>
- amount

Splitwise
- UserController
- GroupController

Expense Controller
- expense
- CreateExpense

Balance Sheet
- map<user, amount>
- totalamount
- expense
- get Back
- owe

SplitFactory
- Validate SPlit

Validate SPlit
- validate()

Equal Split
- validate()

Unequal SPlit
- validate()

Group Controller
- List<Groups>
- // CRUD

Group
- id
- List<Users>
- List<Expense>
- Expense Controller ref

Percentage Split
- validate()

BalanceSheet Controller
- List<Balance Sheet>, Expense
- UpdateBalance()

Balance Sheet Update:

When Expense Controller creates an expense -- Balance Sheet Controller which has list of Balance Sheets updates each Balance Sheet of the user

Rough Flow:

U_1 — U_2: getback = 250
over = 100 U_1 pays 500 for U_2
— U_3
— U_4

U_2 — U_1 owe: 250
— U_3
— U_4
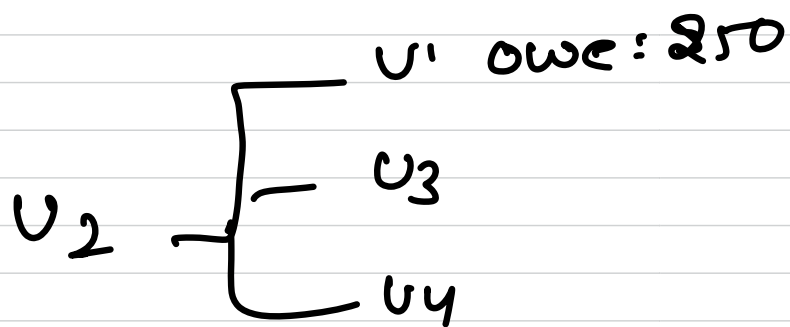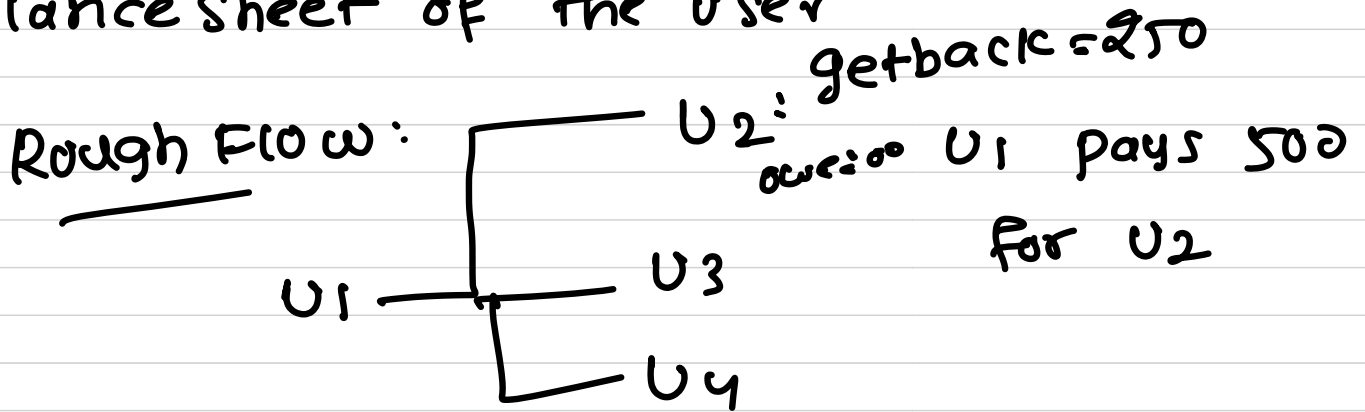
U_1 BalanceSheet

Total amount : 500
Expenses : 250
getBack = 250
owe = 00

U_2 BalanceSheet:

Total amount = 00
Expenses = 250
getBack = 0
owe = 250