Building a Simple Pay-Later Service

As a pay later service we allow our users to buy goods from a merchant now, and then allow them to pay for those goods at a later date.

The service works inside the boundary of following simple constraints -

- Let's say that for every transaction paid through us, merchants offer us a discount.
    - For example, if the transaction amount is Rs.100, and the merchant discount offered to us is 10%, we pay Rs. 90 back to the merchant.
    - The discount varies from merchant to merchant.
    - A merchant can decide to change the discount it offers to us, at any point in time.
- All users get onboarded with a credit limit, beyond which they can't transact.
    - If a transaction value crosses this credit limit, we reject the transaction.

Use Cases

There are various use cases our service is intended to fulfill -

- allow merchants to be onboarded with the amount of discounts they offer
- allow merchants to change the discount they offer
- allow users to be onboarded (name, email-id and credit-limit)
- allow a user to carry out a transaction of some amount with a merchant.

Bonus(attempt this when you atleast have a skeleton code which can achieve the points mentioned above)

- allow a user to pay back their dues (full or partial)
- Reporting:
    - how much discount we received from a merchant till date
    - dues for a user so far
    - which users have reached their credit limit
    - total dues from all users together

Concentrate on:

- Application design, use of design patterns
- DB schema modeling
- The core transaction logic is important, not the getters, setters etc.
- Code extensibility, modularity etc.
- Lastly, I do not expect the entire application with all functionalities to be implemented, what I expect instead is you giving me an idea on how you code.

Sample I/O

```
new user u1 u1@email.in 1000    # name, email, credit-limit
new merchant m1 2%              # name, discount-percentage
new txn u1 m2 400               # user, merchant, txn-amount
update merchant m1 1%           # merchant, new-discount-rate
payback u1 300                  # user, payback-amount
report discount m1
report dues u1
report users-at-credit-limit
report total-dues
```

Example Flow

```
> new user user1 u1@users.com 300
user1(300)

> new user user2 u2@users.com 400
user2(400)

> new user user3 u3@users.com 500
user3(500)

> new merchant m1 m1@merchants.com 0.5%
m1(0.5%)

> new merchant m2 m2@merchants.com 1.5%
m2(1.5%)

> new merchant m3 m3@merchants.com 1.25%
m3(1.25%)


> new txn user2 m1 500
rejected! (reason: credit limit)

> new txn user1 m2 300
success!
```

```
> new txn user1 m3 10
rejected! (reason: credit limit)

> report users-at-credit-limit
user1

> new txn user3 m3 200
success!

> new txn user3 m3 300
success!

> report users-at-credit-limit
user1
user3

> report discount m3
6.25

> payback user3 400
user3(dues: 100)

> report total-dues
user1: 300
user3: 100
total: 400
```