1. Good Evenin
2. Let's begin at 9:08 pm
3. Topic - Design Splitwise

Agenda

1. Recap
2. Design Splitwise

Recap

1. Pen → Class Diagram
2. TTT → Interactive App
3. Parking Lot → MVC

4. [Book My Show → MVC
+
[Model Attributes ( )]
DJ Attributes (SB)
Repository Attributes]

5. Splitwise → MVC
+
Model Attr
DJ Attr
Repository Attr
Restful Api Attr ✓

└→ Atleast 1 end to end data flow

Splitwise

Design
- CD
- SD

Algorithm
→ [DSA problem
+
TTT Undo
TTT O(1)]

Coding
1. Restful Api
2. Command Design Pattern
3. End to End flow of data.

Designing Splitwise

30 min → RG

1.30 min → Coding

30 min → Discussion

Overview

1. Align with **interviewer**
2. **Expecation for type** of system
   - CD
   - Interactive
   - **MVC**

→ Aware of the system ✗

→ Not aware of system ✓

→ Splitwise is an expense sharing app

→ User are clubbed into groups

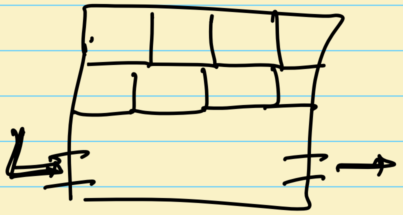
→ Add expenses to the groups
- Who paid?
- Everyone's share

★ Help settlements

↳ Print transactions b/w users so that noone owes anyone anything

Requirement Gathering

1. Draw a sketch



PL
Gate
PS

2. User journey

Ticket
Bill
Payment

★ Requirement Document is shared

Clarifying Requirement

1. Read the doc
2. Take 5 minutes

→ Clarifications

9:30

Clarifications?

1. [Can an expense be paid by multiple users?] → Yes ✓
2. [Will not the expense be shared equally among participants?] → [Not equally shared.]

A, B, C, D ⇒ RoomMates

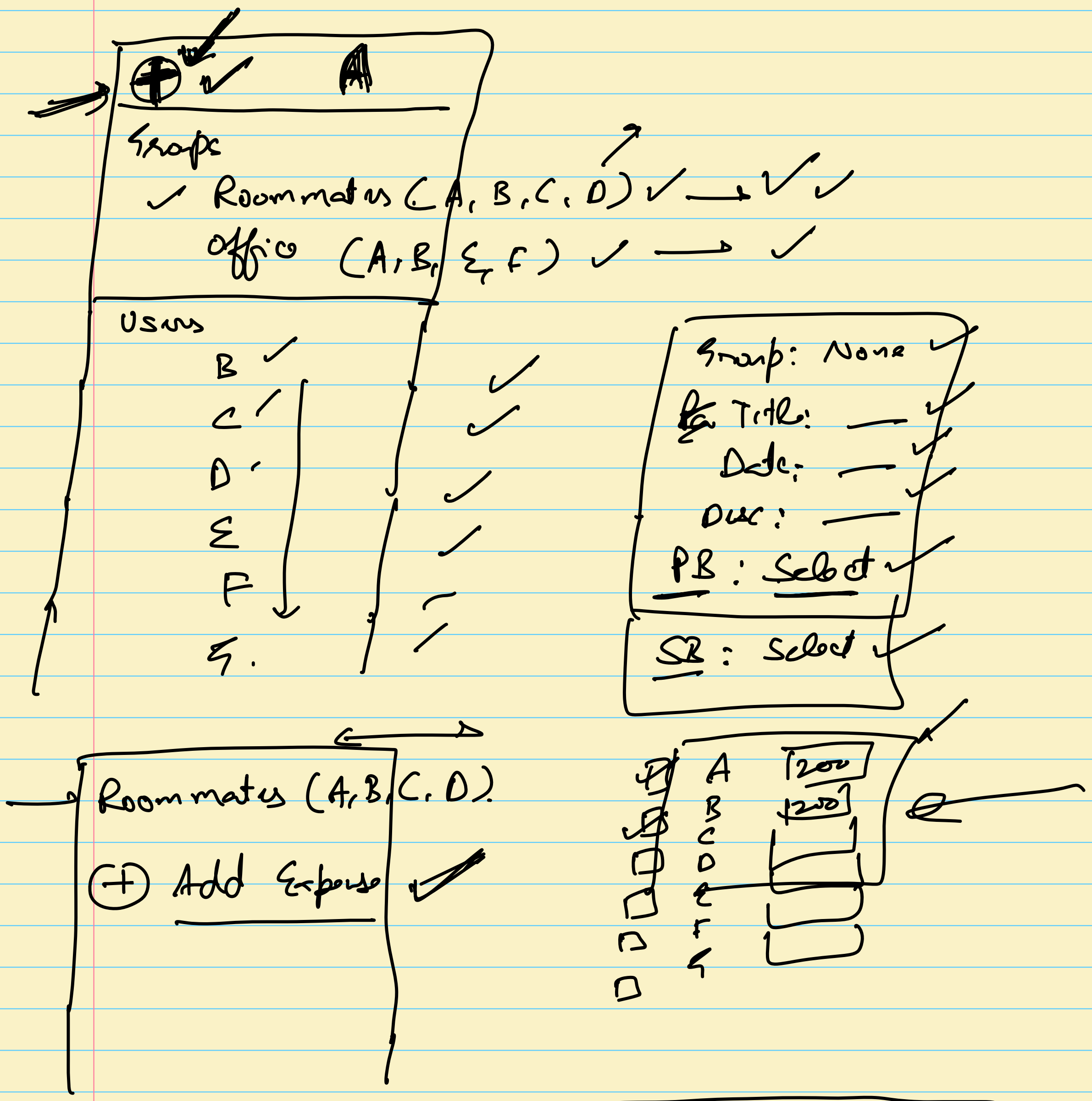
Christmas Party 26/12/2022

Amount : 1000

Paid By : A (500) + B (500)

Shared By : A (300), B(300), C (200), D(200)

3. [Can expense be added without a group? ✓ e.g. A ▷ G]



4. If we add an expense from inside the group, it is not necessary that all members are part of an expense.

5. A user can leave a group if he owes noone, & noone owes him anything → for the expenses of the group.

RM (A, B, C)

→ E1 : 1000
PB : A(500), B(500)
SB : A(400), B(300), C(300)

A → +100 ✓
B → +200
C → -300

x

A has paid 100 extra

Office (A, C, D)

→ E2 : 1000
PB : ~~B~~ D(1000)
SB : A(300), C(300)
D(400)

A → -300
C → -300
D → +600

y

A has paid 300 less

z

A has paid 200 less

Leave Roommates → x
" Office → y
" Splitwise → z

6. Settlement

↳ List of transactions

1. Can be done within a group

2. " " " outside a group ✓

[for the particular user]

Only expense of group will be considered

All expenses involving user [inside any group / w/o any group]

RM (A, B, C, D)

[E1
E2
E3]

∴ Group controller

L<E> getAllExpenses (gid) usrid

X

RM (A, B, C, D)

$\Sigma 1 = 1000$

PB : A(500), B(500)

SB : A(250), B(250), C(250), D(250)

$\Sigma 2 = 1000$

PB = D(1000)

SB = A(350), B(350), C(150), D(150)

A = + 500 - 250 + 0 - 350 = -100] 0

B = + 500 - 250 + 0 - 350 = -100] 0

C = + 0 - 250 + 0 - 150 = - 400] 0

D = 0 - 250 + 1000 - 150 = + 600] 500] 400 0

No of

1. A pay to D 100
2. B pay to ~~B~~D 100
3. C pay to D 400

1. A pay to B 100

No of transaction is least

A ~~-100~~ ✓ 0
B ~~-100~~ 0
C ~~-400~~ 0
D ~~+600~~ / ~~500~~ ~~400~~ 0

1. $A \xrightarrow{100} D$
2. $B \xrightarrow{100} D$
3. $C \xrightarrow{400} D$

←

A ~~-100~~ 0
B ~~-100~~ ~~-200~~ 0
C ~~-400~~ ~~-600~~ 0
D ~~+600~~ 0

1. $A \xrightarrow{100} B$
2. $B \xrightarrow{200} C$
3. $C \xrightarrow{600} D$

A ✓ ~~-200~~ 0
B ~~-100~~ 0
C ~~+200~~ 0
D ~~+100~~ 0
↑

$A \xrightarrow{200} C$
$B \xrightarrow{100} D$

✓

A ~~-200~~ 0
B ~~-100~~ ~~-300~~ 0
C ~~+200~~ ~~-100~~ 0
D ~~+100~~ 0

1. $A \xrightarrow{200} B$
2. $B \xrightarrow{300} C$
3. $C \xrightarrow{100} D$

✗

Least number of transactions
- Algo 1
- Greedy Algo.

Break = 10:25 to 10:35

1. Models

2. Schema Design

GroupParticipant ✓

- id ✓
- group ✓
- User ✓
- ~~Role~~ ✓ ☆

User ✓

- id ✓
- name ✓
- pno ✓
- pwd ✓

Group ✓

- id ✓
- name ✓
- User cb ✓
- List<GroupParticipant> part ✓

☆

Expense ✓

- ✓ id
- ✓ title
- ✓ desc
- ✓ createdAt
- ✓ User createdBy →
- ✓ amount ✓
- ✓ List<ExpenseLedger> pb
- ✓ List<ExpenseShare> sb
- GroupExpense ge. → ☆

E	:	U
1	:	1
m	:	1

GroupExpense ✓

- id
- group ✓
- Expense ✓

G	:	E ✓
1	:	m.
1	:	1
1	:	m

ExpensePayer ✓

✓ id
Expense
User
✓ amount

ExpenseSharer ✓

id
Expense
User
amount

Role

Normal
Admin

Schema Design

✓ 1. Create Table for all classes & enums

✓ 2. Make columns for all primitive, strings, date & enums.

3. Define Relations [1:1
1:m
m:m]

1. user (id, name, pno, pwd) ✓

2. group (id, name, cbusrid) ✓

3. Expense (id, title, desc, createAt, amount, created By)

4. GroupExpense (id, gid, eid) ←

5. ExpensePayer (id, amt, expid, usrid)

6. ExpenseSharer (id, amt, expid, usrid)

7. groupparticipants (id, gid, usrid, role)

8. Role (id, name)

Algorithms

1. Settlement algo 1 = Round trip
2. " " 2 := Greedy
3. TTT Undo 3 algo
4. TTT WS

Final class

1. Command Design
2. Restful Api
3. 1 end to end data flow

Admin

2

```
Select * e.*
from group g
join ge ge
     on g.grid = ge.grpid
join expense e
     on ge.expid = e.expid
where g.grid = 1
```

grp

id	name
1	A
2	B

expense

id	name	amount
1	CP	100
2	NP	200
3	RD	300

grpexp

id	gid	eid
1	1	1
2	1	2

A → CP, NP

B → X

X → RD