Util

Models //Layer interacting with DB

Services // Business Logic

Routes/controller //Service to be triggered/validations

App.js //Entry Point

Interface userInterface{

Void createUser(username,password)

getUserDetails(username)

}

Class user implement user{

Void createUser(username,password){

throw new (“”)

return true

}

getUserDetails(username){

throw new (“User Not Found”)

return user

}

}

Class handleError{

Function handlerror(error){

if(“user not found”) throw new Error(“User\_Not\_Found”)

}

}

—---

F1 - f2 - 1000

F2 - f3 - 3000

F1 - f3 - 2000

* receive
* give

F1 +1000

F2 -1000

F2 +3000

F3 -3000

F1 +2000

F3 -2000

F1 +4000

F2 +3000

F3 -5000

F4-2000

F1 -> f3 = 3000

F2 -> f3 = 2000

2 Transaction:

/\*—---

F1 - f2 - 1000

F2 - f3 - 3000

F1 - f3 - 2000

receive

give

F1 +1000

F2 -1000

F2 +3000

F3 -3000

F1 +2000

F3 -2000

F1 +4000

F2 +3000

F3 -5000

F1 -> f3 = 3000

F2 -> f3 = 2000

2 Transaction:

\*/

let input = [["F1","F2",1000],["F2","F3",3000],["F1","F3",2000]]

console.log(minimizeTransactions(input))

function minimizeTransactions(input){

let map = getNetDebt(input)

console.log(map)

let receiver = new Array()

let giver = new Array()

for(key of map){

if(key[1] > 0){

receiver.push(key)

}

if(key[1] < 0){

giver.push(key)

}

}

receiver.sort(function (a,b){

return b[1]-a[1]

})

giver.sort(function (a,b){

return b[1]-a[1]

})

let giverInd = 0

let receiverInd = 0

while(giverInd < giver.length && receiverInd < receiver.length){

let transactionAmount = Math.min(Math.abs(receiver[receiverInd][1]),Math.abs(giver[giverInd][1]))

console.log(giver[giverInd][0],"->",receiver[receiverInd][0],transactionAmount)

giver[giverInd][1] += receiver[receiverInd][1]

receiver[receiverInd][1] -= giver[giverInd][1]

if(giver[giverInd][1] >= 0) giverInd +=1

if(receiver[receiverInd][1] <= 0) receiverInd +=1

}

}

function getNetDebt(trans){

let map = new Map();

let n = trans.length

for(let i=0;i<n;i++){

let giver = trans[i][1]

let receiver = trans[i][0]

let amount = trans[i][2]

if(map.has(giver)){

map.set(giver,map.get(giver)+amount)

}else{

map.set(giver,amount)

}

if(map.has(receiver)){

map.set(receiver,map.get(receiver)-amount)

}else{

map.set(receiver,-amount)

}

}

return map

}