Recommend algorithm

Inputs : recommenderAddress and recommenderScore

Output : updateSocialRecomendationScore from recommended amount

Process :

S1 <- recommendedAmount = moneyValue - extraMoney

S2<- weight, scores = getRecomendationWeightAndScore

S3 <- updateSocialRecomendationScore

Lend Algorithm

Inputs : investorAddress

Output : investedMoney

Process :

S1 <- totalInvestedAmount = totalInvestedAmount + moneyValue - extraMoney

S2 <- Invested as much as totalInvestedAmount

Repay Algorithm

Inputs : repaidAmount

Output : updateBorrowerSocialScore and repaidAmount

Process :

S1 <- repaidAmount = repaidAmount + (moneyValue - extraMoney)

Once the loan is finished, investor can ask for interest

S2 <- updateLoanState

Increase borrower social score

S3 <- updateBorrowerSocialScore

claimWithInterest Algorithm

Inputs : boolean canAskForInterest, amount, interestRate

Output : pay interestAmount

Process :

If canAskForInterest == true

S1 <- continue

else

S1 <- return

calculate interest amount for lender

if lender == true

multiple the amount by interest rate

S2 <- returnInterestAmount += (((10 \*\* interest \* (100 - recommendersInterestRate)\*1e18) / 10000)\*investedAmounts) + investedAmounts

calculate interest amount for recommender

if recommender == true

if state != fraud

S3 <- recommendedAmount = senderAmount

else

S3 <- recomendedAmount = ((10\*\* ( totalRecommendedAmount - repaidAmount )) / totalRecommendedAmount) \* recommendedAmounts

S4 <- returnInterestAmount += ((10 \*\* interest \* recommendersInterestRate \* 1e18 / 10000) \* recommendedAmount ) + recommendedAmount

Transfer the return amount with interest to the lender

S5 <- pay(toSender) with amount = returnInterestAmount

applyForLoan Algorithm

Inputs : requestedAmount, repaymentCount

Output : createL oan

Process :

check user not fraudulent

if userFraudStatus == false

S1 <- continue

else

S1<- return

The user must not have any loan in progress or its loan is finished OR check if contract is at interest returns stage

if userActiveLoan == true && usersActiveLoan() == isExpiredAndNotFunded

S2 <- destructActiveLoan

else

if userActiveLoan == false || userActiveLoan() == LoanStateInteresReturns

S2 <- continue

else

S2 <- return

S3 <- loanAddress = createLoan

S4 <- usersSetActiveLoan with loanAddress

updateSocialRecomendationScore Algorithm

Inputs : recomenderAddress, recommendedAmount, recomendedScore

Output : socialRecomendetionScore

Process :

for( I as long as weight.length)

S1<- weightedScore = (weights[i]\* scores[i])+ weightedScore

if loan.isNew == true

add previous Score

S2<- socialScore = ((10 \*\* loan.previousTScoreWeight()\*1e18}) \* userSocialRecommendationScore()) + socialScore;

add new recommenders voted score

S3 <- socialScore = (((10 \*\* (100 - loan.previousTScoreWeight())\*1e18) \* weightedScore) + socialScore)

S4 <- socialScore = socialScore /100

else

S4 < - socialScore = weightedScore

S5 <- setSocialRecomendationScore with socialScore

getRecomenderWeightAndValue Algorithm

Inputs : recomenderCount, recomenderAddress, recomendedScore

output :weight and scores

Process :

for (I as long as recomenderCount)

if recommender == true

S1 <- recommendedAmount = 10\*\*recommendedAmounts

S2 <- weight = recommendedAmount / 10 \*\* totalRecommendedAmount

S3 <- scores = recomendedScore

S4 <- return (weight ,scores)