Code Email for Engineers

Note to the candidates

Thanks for your interest in joining Amartha.

We expect all engineers to be able to write and have nuanced conversation around code. We ask you to solve a small simple problem around the problem statements we have in Amartha, so as to understand how you solve problems and implement solutions in code.

There are 3 problems in the following email. Please pick any one that interests you and think best highlights your skills.

Please post the code that solves the specific problem. We are happy to receive the code as a ZIP file of the folder(git please) or an online link to the code repository in the email. And we will review it in a week and respond back.

While implementing please think about simplicity of the code, and clarity for the reviewer(an Amartha engineer). Please make any reasonable assumption that is not explicit in the problem statement.

We are looking for quality of code, and the completeness of the solution(for the problem that is specified). We'd love to look at the tests esp for the basic requirements on the problems.

Proposed Code Problems

Example 1: Billing Engine (System Design and abstraction)

We are building a billing system for our Loan Engine. Basically the job of a billing engine is to provide the

- Loan schedule for a given loan(when am i supposed to pay how much)
- Outstanding Amount for a given loan
- Status of weather the customer is Delinquent or not

We offer loans to our customers a 50 week loan for Rp 5,000,000/- , and a flat interest rate of 10% per annum.

This means that when we create a new loan for the customer(say loan id 100) then it needs to provide the billing schedule for the loan as

W1 : 110000

W2: 110000

W3: 110000

…

W50: 110000

The Borrower repays the Amount every week. (assume that borrower can only pay the exact amount of payable that week or not pay at all)

We need the ability to track the Outstanding balance of the loan (defined as pending amount the borrower needs to pay at any point) eg. at the beginning of the week it is 5,500,000/- and it decreases as the borrower continues to make repayment, at the end of the loan it should be 0/-

Some customers may miss repayments, If they miss 2 continuous repayments they are delinquent borrowers.

To cover up for missed payments customers will need to make repayments for the remaining amounts. ie if there are 2 pending payments they need to make 2 repayments(of the exact amount).

We need to track the borrower if the borrower is Delinquent(any borrower that who's not paid for last 2 repayments)

We are looking for at least the following methods to be implemented

- GetOutstanding : This returns the current outstanding on a loan, 0 if no outstanding(or closed),
- IsDelinquent : If there are more than 2 weeks of Non payment of the loan amount
- MakePayment: Make a payment of certain amount on the loan

Example 2: reconciliation service (Algorithmic and scaling)

Design and implement a transaction reconciliation service that identifies unmatched and discrepant transactions between internal data (system transactions) and external data (bank statements) for Amartha.

Problem Statement:

Amartha manages multiple bank accounts and requires a service to reconcile transactions occurring within their system against corresponding transactions reflected in bank statements. This process helps identify errors, discrepancies, and missing transactions.

Data Model:

- Transaction**:**
  - `trxID`: Unique identifier for the transaction (string)
  - `amount`: Transaction amount (decimal)
  - `type`: Transaction type (enum: DEBIT, CREDIT)
  - `transactionTime`: Date and time of the transaction (datetime)
- Bank Statement**:**
  - `unique_identifier`: Unique identifier for the transaction in the bank statement (string) (varies by bank, not necessarily equivalent to `trxID`)
  - `amount`: Transaction amount (decimal) (can be negative for debits)
  - `date`: Date of the transaction (date)

Assumptions**:**

- Both system transactions and bank statements are provided as separate CSV files.
- Discrepancies only occur in amount.

Functionality:

- The service accepts the following input parameters:
  - System transaction CSV file path
  - Bank statement CSV file path (can handle multiple files from different banks)
  - Start date for reconciliation timeframe (date)
  - End date for reconciliation timeframe (date)
- The service performs the reconciliation process by comparing transactions within the specified timeframe across system and bank statement data.
- The service outputs a reconciliation summary containing:
  - Total number of transactions processed
  - Total number of matched transactions
  - Total number of unmatched transactions
    - Details of unmatched transactions:

        - System transaction details if missing in bank statement(s)
        - Bank statement details if missing in system transactions (grouped by bank)
    - Total discrepancies (sum of absolute differences in amount between matched transactions)

Example 3: Loan Service (system design and abstraction)

we are building a loan engine. A loan can a multiple state: proposed , approved, invested, disbursed. the rule of state:

1. proposed is the initial state (when loan created it will has proposed state):
2. approved is once it approved by our staff.
    a. a approval must contains several information:
        i. the picture proof of the a field validator has visited the borrower
        ii. the employee id of field validator
        iii. date of approval
    b. once approved it can not go back to proposed state
    c. once approved loan is ready to be offered to investors/lender
3. invested is once total amount of invested is equal the loan principal
    a. loan can have multiple investors, each with each their own amount
    b. total of invested amount can not be bigger than the loan principal amount
    c. once invested all investors will receive an email containing link to agreement letter (pdf)
4. disbursed is when is loan is given to borrower.
    a. a disbursement must contains several information:
        i. the loan agreement letter signed by borrower (pdf/jpeg)
        ii. the employee id of the field officer that hands the money and/or collect the agreement letter
        iii. date of disbursement

movement between state can only move forward, and a loan only need following information:

- borrower id number
- principal amount
- rate, will define total interest that borrower will pay
- ROI return of investment, will define total profit received by investors
- link to the generated agreement letter

design a RESTFful api that satisfy above requirement.