Backend Problem Statement

Introduction:

Welcome to RobusTest technical evaluation round 2, In this round we provide you with a problem statement which you will have to understand and provide with the solution using the tech stack mentioned by us.

Evaluation:

- The purpose of this round is to evaluate
    - your approach to problem solving.
    - how well you understand the domain (backend)
    - approach you take to understand any new technologies.
- We will not judge you on the basis of how quick you submit your solution, but we will rather be judging you on how well you understood the problem.
- You will have a extra advantage if you attempt bonus section but only once you are have build everything that is been asked in Functional/System Requirements.

Tech Stack you will be using for the development of this application

- Backend language: Golang
- API Architecture: REST
- No need to use DB, you can use json files to store data.
- You can't use any external/internal library for API communication, for example "net/http" , "fiber", "gorilla" , "gin" etc.
- For your API to communicate you need to create a TCP server
    - that runs on port "8080"
    - handles more than one request at a given point
    - Supports REST protocol
- Create separate APIs for Account Administrator (Bank) and Account user (Individual)

Problem Statement

Description

- Create a banking system that allows
  1. user to send and receive money
  2. user to view transactions, and balance
  3. administrator to view balance and transactions based on the user accounts

- Following are the detailed specifications according to which the system has to be built
  - **Functional/System Requirements**
    - **Administrator account (bank)**
      - They should be able to view transactions based on user ID.
        - We should be able to see at least the last 50 transactions.
      - Can send money from user A account to user B's account.
        - The money should only be transferred when
          - User A account ID is correct
          - User B's account ID is correct
          - User A should have sufficient balance in the account to make transactions.
        - You should be able to check the balance for the user.
          - Based on the user account ID.
    - **Account user (Individual)**
      - There should be a way for the user to send money to some other account
        - The money should be only transferred to the account
          - If the receiver account ID is valid
          - There is a sufficient amount present in the account to make the transactions.
      - There should be a way for users to see
        - Account balance
        - At least the last 10 transactions.

How to submit your solution

- Submit your code solution as a public Git repo
- A document that helps us in understanding how we can run this application.
- The platform should have some prefilled entries (Whatever you decide is best) but it should satisfy all basic/general requirements of a banking account data
- Submit the solution to us via email to our technical recruiter (please check your email for this information)
    - Make sure the email has the following information
        - Subject line: Submission for the technical evaluation round 2.
        - It should have the link to your git repository.

Bonus:

- If you can create a bare minimum UI for testing this application will be great.
- You can allow the admin to see
    - Select count for last X transactions.
    - Based on month + year.
- You can provide a way to access the API using Accesskey.
    - This will allow you to access these APIs without logging in.

Additional Information

Throughout this round if you have any questions/queries with respect to the problem statement or how to submit your solution you can contact our technical recruiter through email

- We would request you to create a email thread with the following subject line.
    - Queries/Question with respect to technical evaluation round 2.
- Use single thread throughout this conversation