Vendia Core Platform Engineer Assignment

Banking API

Overview

This document outlines several sets of requirements for an application we'd like you to build. The application will be deployed and reviewed by a Vendia engineer. The requirements of this application should be kept confidential during and after your interview process. During later phases of the interview process we may ask you to extend this application in a live coding exercise and to explain specific design decisions.

Product Requirements

Create a business to business banking service. The service must be exposed via an authenticated API. The service is used by multiple business partners to transfer money from accounts they own to accounts owned by their partners. Business partners have multiple accounts with their own names such as "long term savings", "operating capital", or "payroll". Each account has a current balance representing the number of dollars in the account.

The banking service has 2 required APIs:

`GET_BALANCE(account_name)`
Returns the current balance for the specified account type for the authenticated partner.

`TRANSFER(amount, src_account_name, dest_partner, dest_account_name)`
Transfer the amount from the source account type for the authenticated partner into the dest account name owned by the dest partner. The Transfer API has the following requirements:

1. The source partner must be determined using authentication information provided to the API.
2. Source partners can only transfer money out of accounts that they own.
3. Transfers that would cause the source account balance to go negative must be rejected.
4. Transfers to an unknown partner must be rejected.

Technical Requirements

Below are the technical requirements for the assessment and some reference resources.

1. Your solution must run using a cloud serverless compute service. If there is no traffic to your service there must be no compute cost incurred (i.e. the solution must not pay for a server waiting to service web traffic during hours of inactivity). Some example serverless compute services:

    a. AWS Lambda
    b. GCP Cloud Functions
    c. Azure Functions
2. Your solution must be fully deployable using an Infrastructure-as-Code solution such as:
    a. CDK
    b. SAM
    c. CloudFormation
    d. Terraform
3. Your solution does not need to use a serverless database service. If there is no traffic to your database it is ok if there is a cost incurred (i.e. you can use Amazon DynamoDB, Azure CosmosDB, or host a SQL server on GCP).
4. You may use any language you are comfortable with.
    a. Verify that the language you've chosen is supported by the serverless compute service you're using.
5. The service must use an industry standard authentication mechanism. Some examples include:
    a. API Keys, OAuth, cloud specific auth (i.e. AWS SigV4 for API Gateway or GCP OIDC token)
6. Adding new business partners and new accounts to the application can be a manual operation. This operation can require downtime and/or direct database access.
    a. You do not need to implement a user registration or credential management workflow (i.e. no sign in api, no password reset)
7. The account balance can fit inside of an integer.
    a. The account balance represents whole dollar amounts. We're using an integer to avoid floating point math.
8. Each business partner is limited to 10 accounts.
9. Besides limits imposed by the cloud provider you've chosen, there should be no limit to the number of business partners your system can support.
10. Your code should be documented to your typical level of documentation.
11. Money must not be lost during transactions.
12. Money must not be generated during transactions.
13. The banking service must log enough information to determine all transactions that occurred successfully.
14. You can assume there will be less than 20 concurrent transactions for a single account.

Assessment Specific Requirements

Given that this is a take home tech assessment we have several additional requirements:

1. All code and instructions for how to deploy the solution must be in a private GitHub repository that is shared with Vendia.
    a. The deployment of the service must be fully automated with no clicking inside of a cloud console required.

2. Include an example of how to interact with your banking service to transfer 1000 dollars from the “operating capital” account of a business partner named “Vendia” to the “inbound payments” account of a business partner named “Mega Corp Enterprises”.
3. You may use any information available to assist you (i.e. Azure documentation, StackOverflow articles, Google searches). You may not work with other people.
    a. You may ask Vendia personnel clarifying questions.