innoscripta

CASE STUDY

Golang Developer

Golang Backend Take Home Challenge

Welcome to the take-home challenge for the Golang Backend Web Developer position. We are excited to see your skills and experience in action. The challenge is to build the backend functionality for a banking ledger website that manage bank accounts and transactions and serves them to the frontend application.

Requirements:

You are developing a banking ledger service designed to reliably manage account operations, even under high load. The service must:

1. **Support the creation of accounts** with specified initial balances.
2. **Facilitate deposits and withdrawals** of funds.
3. **Maintain a detailed transaction log** (ledger) for each account.
4. **Ensure ACID-like consistency** for core operations to prevent double spending or inconsistent balances.
5. **Scale horizontally** to handle high spikes in transaction volume.
6. **Integrate an asynchronous queue or broker** to manage transaction requests efficiently.
7. **Include a comprehensive testing strategy**, covering feature tests and mocking for robust validation.

Challenge Guidelines

1. The output expected from this challenge is a backend project using Golang that provide a docker-compose.yml (or equivalent) that spins up all required services (API gateway, queue/brocker, transaction processor, and database).
2. Create API endpoints to allow external interaction, including creating accounts, processing transactions, and retrieving transaction history.
3. Utilize data fetching and storage mechanisms for account and ledger data, leveraging a relational database (e.g., PostgreSQL) for account balances and a NoSQL database (e.g., MongoDB or DynamoDB) for storing transaction logs to enable efficient querying and scalability. Additionally, integrate an asynchronous queue or broker (e.g., RabbitMQ/Kafka) to manage transaction requests and ensure reliable message delivery.
4. Provide a comprehensive testing strategy, including unit, integration, and feature tests with mocking for robust validation.
5. Incorporate best practices of software development, including DRY (Don't Repeat Yourself), KISS (Keep It Simple, Stupid), and SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation, Dependency inversion) principles.
6. Focus on clean, maintainable code with proper documentation, ensuring adherence to best practices and demonstrating scalability.

Please note that this challenge is centered solely on the backend development of the banking ledger service using Golang, with a focus on reliability, scalability, and adherence to best practices.