Technical Project

We've designed this project to test your code comprehension skills, your knowledge about databases and minimally test your Go skills. Remember to review the Go Tour (https://go.dev/tour/welcome/1) if you're new to Go. To be fair to all candidates, with respect to time, we expect to have something back from you by **the morning of Wednesday, March 16, 2022**.

💡 This project should take a couple of hours rather than a full day or the entire time that is allotted. Let us know if you have any questions or clarifications. We're happy to answer any questions about the prompt or parameters, even over a weekend. We want to help you succeed.

The Application

Our technical project tasks you with improving and reviewing a simple Go microservice for handling orders, called `order-up`. You can envision this microservice living in the cloud with other microservices as part of a larger application. Customers will be placing orders on a hypothetical website and employees will be charging and fulfilling orders using a hypothetical internal website. The `order-up` service will need to oversee all aspects of an order: creating, charging, fulfilling, and cancelling, however the actual charging and fulfillment is carried out by other microservices. For the purposes of this project these other microservices in the application are mocked.

Getting Started

You should be able to create a new GitHub repository using the `order-up` repository (https://github.com/levenlabs/order-up) as a template by clicking the **"Use this template"** button (avoid using the Fork button) or you can download this repository and work on it offline.

Feel free to make commits as you usually would and work in whatever fashion you're used to. You will not be making a pull request and you can just push to the default branch when you're satisfied. You don't need to host the final result and sending it as a tar/zip is sufficient if you don't want to create a public repository on GitHub. The README provides details on the project's structure and how to get started.

Requirements

There are five components to this evaluation:

1. **Add a Cancel API endpoint**

   Customers need the ability to cancel and refund orders. You need to implement a cancel API endpoint that refunds the customer if they were already charged and updates an order’s status to cancelled. The charge service’s /charge endpoint allows you to pass a negative amountCents value to refund a card for a particular amount. If the refund failed you should not update the status of the order and return the error. If an order has already been fulfilled then you must return a 409 Conflict error since we’ve already fulfilled the order and cannot cancel it.

   You are expected to write any tests necessary to ensure your endpoint is working property. The other tests can be used as a reference.

   We’re hoping to understand your code comprehension and ability to work within a new repository. Additionally this will test your REST knowledge and ability to write tests for your code.

2. **Add a Fulfill API endpoint**

   Employees will be using an internal website to mark orders as fulfilled when they’re ready. You need to implement a fulfill API endpoint that ensures the order’s already been charged, makes a PUT /fulfill request to the fulfillment service for each line item, and finally updates the order’s status to fulfilled. The fulfillment service’s /fulfill endpoint returns a 200 OK if the fulfillment succeeded. If any of the fulfillment service requests fail you should not update the status of the order and return the error. The fulfillment service is idempotent as long as an orderID is passed so if an employee tries again you can call /fulfill again on the fulfillment service and it will ignore items that are already be fulfilled.

   You are expected to write any tests necessary to ensure your endpoint is working property. The fulfillment service mock has been provided for you. The other tests can be used as a reference.

Just like the first component, we’re hoping to understand your code comprehension, ability to work within a new repository, REST knowledge and test writing. Additionally, this will test error handling.

3. **Write database code**

   Only the function skeletons and tests exist for in the storage package/folder and so you need to decide on a database (choose one you’re most familiar with, we don’t care which one) and add the appropriate initialization code and CRUD code to each function in the storage package. There are some packages and tutorials in the project’s README to help you get started.

   The tests in the storage package should pass to ensure you’ve correctly implemented each function as the api package expects them.

   This component tests your database knowledge. We’re looking for you to write the necessary CRUD queries as well as the necessary code for connecting to the database.

4. **Limit concurrent charges**

   Our charging service can only support 1 concurrent charge at a time. The order service needs to limit itself to only having 1 outgoing /charge request at a time and most importantly, any incoming requests should be queued up until any previous ones have completed. You can assume there’s only 1 instance of the order-up service running. A test has already been written to verify that only 1 concurrent request is being sent to the charge service.

   This component tests your ability to use Go-specific functionality to deal with concurrency and restraints imposed by downstream services.

5. **Document API**

   Finally, you need to write documentation for a frontend developer who might be interacting with this service. You should include all necessary details for them to build the public website and internal site assuming they have no existing knowledge about the service or it’s implementation. You can write the documentation in whatever format is easiest for you, for example Markdown, Google Docs, or a PDF, etc..

   This final component demonstrates your ability to communicate documentation to fellow coworkers necessary for interacting with something you’ve built. We are not evaluating style or formatting.

Next Steps

Once you've completed the project to the best of your ability, let us know and we'll review your project. We'll have a follow-up discussion to go over any feedback or questions we have and ask you to describe the issues you found and your reasoning behind different decisions you made when implementing the new functionality.