Benzinga - Backend Golang Example Project

You will need to create a relatively basic project using Go meeting the requirements described below. The purpose of this project is to demonstrate an understanding of several things you'll be expected to employ on a regular basis such as knowledge of Go, version control systems, REST API servers and clients, JSON, Docker, logging as well as basic load management and resiliency patterns.

The application should be a basic webhook receiver that has two endpoints.

1. GET /healthz - should return HTTP 200-OK and "OK" as a string in the body
2. POST /log - should accept JSON payloads

Example Payload:

```
{
   "user_id": 1,
   "total": 1.65,
   "title": "delectus aut autem",
   "meta": {
       "logins": [
           {
               "time": "2020-08-08T01:52:50Z",
               "ip": "0.0.0.0"
           }
       ],
       "phone_numbers": {
           "home": "555-1212",
           "mobile": "123-5555"
       }
   },
   "completed": false
}
```

The application should have three configurable values: *batch size*, *batch interval* and *post endpoint*, that should be read from environment variables.

The application should deserialize the JSON payloads received at /log endpoint into a struct and retain them all in-memory using a method of your choice. When a set batch size is reached OR the batch interval has passed the application should forward the collected records as an array to the post endpoint and clear the in-memory cache of objects.

Using a logging library that offers structured and leveled logging the application should log an initialization message on startup and on each request, this may require custom middleware. Each time it sends a batch it should log the batch size, result status code, and duration of the POST request to the external endpoint.

If the POST fails it should retry 3 times, waiting 2 seconds before each retry. After 3 failures the application should log this failure and exit.

For this project only the Go standard library should be used with the exceptions for a structured logging library and server framework such as Chi, Echo, Gin or similar. Dependency management should use Go modules and the project will be linted using https://github.com/golangci/golangci-lint.

It should be delivered via a public Git repository and will be run using Docker using a Dockerfile that should be provided by you in the repository. Testing for the Post endpoint output will be done against a service such as http://requestbin.net, unit-tests are optional.

Please don't hesitate to reach out for clarification on any requirements or expectations.