Golang Assignment

THE CHALLENGE

We'd like you to create a RESTful API with two endpoints.

1. One of them that fetches the data in the provided MongoDB collection and returns the results in the requested format.

2. Second endpoint is to create(POST) and fetch(GET) data from an in-memory database.

REQUIREMENTS

- The code should be written in Golang without using framework. (includes mux, router etc)
- MongoDB data fetch endpoint should just handle HTTP POST requests.
- The application should be deployed on AWS or Heroku. You don't need to use any API Gateway, Load Balancers or any other layer than the developed application.
- The up to date repo should be publicly available in Github, Bitbucket or equivalent.

DELIVERABLES

- The public repo URL which has the source code of the project, and a set of
- instructions if there is any project specific configurations needed to run the project.
- Public endpoint URLs of the deployed API which is available for testing.

Worth Highlighting

We expect these requirements can be delivered in 4 to 8 hours. However, it is not a speed test. Take your time! Your feedback on how much actual time you were needed to deliver the task will be very helpful but will not be used for the evaluation. You are free to use any libraries (except web frameworks) to deliver the needed functionality, but be prepared to explain other solutions that you would have implemented if you have more time.

Crucial Points

- Delivering a Working RESTful API.
- Clean and Production Ready Code
- Error Handling
- Comments and Documentation
- Unit and/or Integration Tests
- Avoid Over Engineering

Good luck with this assignment! Try to make good use of this task to demonstrate and show off your coding skills. If you have any questions, don't hesitate to ask your contact person within Getir.

Required Information

MongoDB URI:

mongodb+srv://challengeUser:WUMglwNBaydH8Yvu@challenge-xzwqd.mongodb.net/getir-case-study?retryWrites=true

Table Name: "records"

1. **For the First Endpoint (fetch data from mongodb)**

   - Request Payload
   The request payload of the first endpoint will include a JSON with 4 fields.
   - "startDate" and "endDate" fields will contain the date in a "YYYY-MM-DD" format. You should filter the data using "createdAt"
   - "minCount" and "maxCount" are for filtering the data. Sum of the "count" array in the documents should be between "minCount" and "maxCount".

Sample:

```
{
    "startDate": "2016-01-26",
    "endDate": "2018-02-02",
    "minCount": 2700,
    "maxCount": 3000
}
```

- Response Payload

Response payload should have 3 main fields.

- “code” is for status of the request. 0 means success. Other values may be used for errors that you define.
- “msg” is for description of the code. You can set it to “success” for successful requests. For unsuccessful requests, you should use explanatory messages.
- “records” will include all the filtered items according to the request. This array should include items of “key”, “createdAt” and “totalCount” which is the sum the “counts” array in the document.

Sample:

```
{
    "code":0,
    "msg":"Success",
    "records": [
        {
            "key":"TAKwGc6Jr4i8Z487",
            "createdAt":"2017-01-28T01:22:14.398Z",
            "totalCount":2800
        },
        {
            "key":"NAeQ8eX7e5TEg7oH",
            "createdAt":"2017-01-27T08:19:14.135Z",
            "totalCount":2900
        }
    ]
}
```

2. **In-memory Endpoints**
    a. **POST Endpoint**

    - Request Payload

    The request payload of POST endpoint will include a JSON with 2 fields.

- “key” fields holds the key (any key in string type)
- “value” fields holds the value (any value in string type)

Sample:

```
{
    "key": "active-tabs",
    "value": "getir"
}
```

- Response Payload

Response payload should return echo of the request or error (if any).

b. **GET Endpoint**

- Request Payload

The request payload of GET endpoint will include 1 query parameter. That is “key” param holds the key (any key in string type)

Sample:
http://localhost/in-memory?key=active-tabs

- Response Payload

Response payload of GET endpoint should return a JSON with 2 fields or error (if any).

- “key” fields holds the key
- “value” fields holds the value

Sample:

```
{
    "key": "active-tabs",
    "value": "getir"
}
```