Assignment

The Challenge

We'd like you to create a RESTful API with a single endpoint that fetches the data in the provided MongoDB collection and return the results in the requested format.

Requirements

- The code should be written in Node.js using express framework
- The 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.
- The public endpoint URL of the deployed API which is available for testing.

Worth Highlighting

We expect these requirements can be delivered in 3 to 6 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 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 (Jest is preferable but Mocha also works)
- 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

Request Payload

The request payload 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 of 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
        }
    ]
}
```