Jane Technologies Inc. - Advertising Coding Challenge

Introduction

Thank you for taking the time to complete this coding challenge for Jane Technologies! Please be sure to read and understand these instructions completely.

Evaluation Points

Your assignment will be evaluated based on the following areas:

- Readable code.
- Implement all 3 endpoints.
- Correctness. **Note**: we will run your code and use the examples below to test.
- Reasonable test coverage.
- Ability to handle requests from multiple clients.
- README with instructions for running your code and any documentation you wish to include.

What You'll Do

We would like you to build one ad server using Go. Your implmenetation should be as straightforward and self contained as possible. We encourage you to perform all logic and state storage in memory. Persistance between executions is not required.

If you're new to Go, we encourage you to try it out on this take-home task. If you prefer not to use Go, please implement it in Java.

The following is a description of a simplified digital advertising(ad) process.

1. First, advertisers set up campaigns.
2. Second, when a user visits a website, the ad server finds a matching campaign and returns an ad to the user.

When the ad is shown to a user, it's called an impression. That is, the user is impressed by an ad. This action is recorded via a impression url callback.

Problem

You will implement an ad server that has the following 3 endpoints.

POST /campaign

This endpoint creates a new campaign.
The request body is a JSON with the following fields. All fields are required.

- `start_timestamp` : The time the campaign becomes active. The timestamp is the seconds since the Unix epoch (00:00:00 UTC on 1 January 1970).
- `end_timestamp` : The time the campaign becomes inactive. The timestamp is the seconds since the Unix epoch. It should be larger than start_timestamp. The end_timestamp must be in the future.
- `target_keywords` : A list of strings. For example, ["iphone", "5G", "headphones", "Apple"].
- `max_impression` : The maximum number of times the campaign can be impressed.
- `cpm` : Cost Per Mille, i.e., the price for 1000 impressions. The price is a positive float number.

If the request body is valid, return status code 200. The response body is in JSON with the following fields.

- `campaign_id` : an integer that identifies the campaign.

If the request body is invalid, e.g. missing required field, return status code 400 and empty response body.

POST /addecision

This endpoint returns an ad. The body is a JSON with the following fields. All fields are required.

- `keywords` : A list of strings. For example, ["android", "5G"].

If the request body is invalid, return status code 400 and empty response body. If the request body is valid and there is a matching campaign, return status code 200. The response body is JSON with the following fields.

- `campaign_id` : An integer that identifies a campaign. If there are multiple matching campaigns, return the campaign with the highest CPM. If there are still multiple campaigns, returns a random campaign.

- `impression_url` : A callback url that is called when the ad is impressed. You can choose any pattern you like for impression_url.

When there are more than one matching campaigns, returns the campaign with the highest `cpm` . If there are still ties, return the campaign with the earliest `end_timestamp` . If there are still ties, return the campaign wi th the smallest `campaign_id` .

A campaign matches an ad request if and only if:

1. The campaign is active.
2. The campaign shares at least one common keyword with the ad request.
3. The user from the ad request has been impressed by the campaign less than the campaign's "max_impression."

If there is no matching campaign, returns status code 200. The response body is empty.

GET [impression-url]

This endpoint records an impressed ad.
If the impression-url is valid, return status code 200 and empty response body.
If the impression-url is invalid, return status code 400 and empty response body.

Example

Assuming the ad server is at localhost:8000. The following is an example of requests in chronological order.

1. `curl -i --header 'Content-Type: application/json' http://localhost:8000/campaign --data-raw '{"start_timestamp":1642493821,"end_timestamp":1742580221,"target_keywords":["iphone","5G"],"max_impression":1,"cpm":20}'`
   200 OK.
   response body: {"campaign_id":1001}

2. `curl -i --header 'Content-Type: application/json' http://localhost:8000/campaign --data-raw '{"start_timestamp":1642493821,"end_timestamp":1742580221,"target_keywords":["iphone","5G"],"max_impression":1,"cpm":10}'`

200 OK.
response body: {"campaign_id":1002}

3. `curl -i --header 'Content-Type: application/json' http://localhost:8000/addecision --data-raw '{"keywords":["iphone"]}'`
   200 OK.
   response body: {"campaign_id":1001,"impression_url":"http://localhost:8000/lahqlu1"}. The "impression_url" in your implementation may be in different pattern.

4. `curl -i --header 'Content-Type: application/json' http://localhost:8000/addecision --data-raw '{"keywords":["android"]}'`
   200 OK. response body is empty.
   There is no matching campaign because the keyword does not match with any active campaigns.

5. `curl -i "http://localhost:8000/lahqlu1"`
   200 OK response body: {}

6. `curl -i --header 'Content-Type: application/json' http://localhost:8000/addecision --data-raw '{"keywords":["iphone","5G"]}'`
   200 OK.
   response body: {"campaign_id":1002,"impression_url":"http://localhost:8000/lyadm1"}. The "impression_url" in your implementation may be in different pattern.

7. `curl -i "http://localhost:8000/lyadm1"`
   200 OK response body: {}

8. `curl -i --header 'Content-Type: application/json' http://localhost:8000/addecision --data-raw '{"keywords":["iphone"]}'`
   200 OK.
   response body is empty.
   All campaigns have reached max impressions.