WalkMe - Mobile Backend Team Coding Test

This is part of the evaluation process we use to determine your suitability to the Mobile Backend team at WalkMe. This test should reflect how you approach a problem; design a solution; write clean, testable and maintainable code; and deliver a piece of working software.

Notes

1. Coding should be done in Java or Node
2. Use as much time as you need, but please limit yourself to a few hours
3. Delivery can be a zip file (remove the **`node_modules`** or **`target`** directory!) or a public git repository
4. If you get stuck or have any questions, don't hesitate to email for assistance!

Definitions

Campaign: A piece of data that holds information regarding a single campaign that should run on a user's device. A JSON representation of a campaign object looks like this:

```
{
	"id": 100,		// id (type: integer)
	"name": "March Sales",	// campaign name
	"thresholds": {		// campaign thresholds:
		"max_total": 300,	// maximum total calls made to this campaign
		"max_per_user": 3,	// maximum calls to this campaign per user
	},
	"data": {}		// campaign data (empty for this exercise)
}
```

Campaign Thresholds: For each campaign, once the appropriate counter (total calls, or calls per user) exceeds any of the matching thresholds (if any exist), the campaign will no longer be sent back as a response to the API call. Two types of thresholds exist:
- **max_total**: The maximum total number of times this campaign was sent back (for all users)
- **max_per_user**: The maximum number of times this campaign was sent back for each user

API URL: The request URL that should be implemented for this exercise looks like this:

```
GET /campaigns?user_id=7357
```

(user_id is an integer)

Instructions

1. We're building a web service that, given a user ID, will return the list of campaigns that have not yet reached their threshold.
2. Each campaign can be configured with both thresholds (with different values for each, of course); with one of the two; or with no thresholds at all.
3. Each call to the API (with a given user ID) goes through the following process:
    a. Find all the matching campaigns, according to their counters and thresholds
    b. Update the counters of the matching campaigns
    c. Return a list of the matching campaigns in the HTTP response

Example

Say we have three campaigns:

ID	**Name**	**Thresholds**
101	Holiday Specials	max_total: 1000 max_per_user: 10
102	Rate this app	max_per_user: 3
103	First come first served	max_total: 4

In the example above:

- Calling `/campaigns?user_id=2786` three times will return all campaigns.
- Calling `/campaigns?user_id=2786` a fourth time will return campaigns **101** and **103**; but not campaign **102**, since its per-user threshold for user **2786** has exceeded.
- Calling `/campaigns?user_id=1337` will return campaign **101** (no threshold exceeded yet) and campaign **102** (the counter for user **1337** is still zero at this point); But it will not return campaign **103** because its total count has exceeded the **max_total** threshold because of the previous 4 calls

What we're looking for

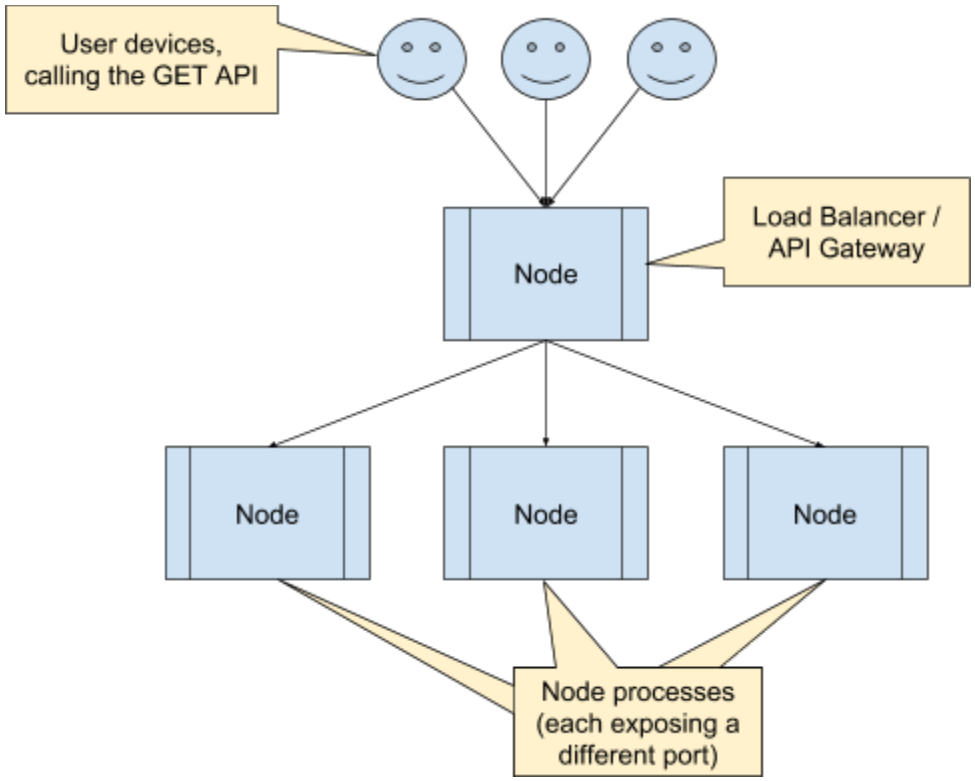
- Something that works - even if you didn't implement every feature you planned to do
- Instructions on how to build, install and run it; plus some usage examples
- Clean and tested code - you don't have to provide 100% test coverage, an occasional test here and these would do (look at either **tap** or **ava** for Node testing frameworks)
- A short description of where your system might fail (Heavy load; Bad database connections; ease of adding new features; stupid users; etc.)

- **Note**: All data should be transient and in-memory; no real database is required

Extra Points

In no particular order, try to add the following capabilities

1. Some basic back-office UI
    - Present the status of each campaign (how many users got it; how many times the per-user threshold has been reached; whether the total threshold has been reached; etc.)
    - Reset a campaign's counters
    - Add / remove / edit campaigns
2. Microservice deployment - running multiple processes to minimize response time
    - Suggested deployment:



    - Leading questions: How would you keep data integrity? What might be weak spots, where data may be overwritten by two (or more) concurrent calls? What would be a good solution?