# Coding Challenge

## About this challenge

This challenge focuses on RESTful API and data model design. It is composed of 3 parts for which you should present a single solution.

We recommend that you write up a few short paragraphs explaining the decisions you made during the implementation of your solution, as well as any missing improvement steps to become a production-ready project.

Any questions you may have please contact.

## Context

This test's main goal is to create a simplified version of the Offers API platform, which allows customers to connect brands (ex: Starbucks) to offers.

*Example: Add 5% cashback offer to Starbucks Oxford street location*

Feel free to browse our docs to familiarize yourself with our current commercial offering. When implementing your solution, you should take **high request volume and concurrency** into consideration.

It is expected your solution to use the below technologies:

- AWS platform as provider,
- Lambda,
- DynamoDB,
- Deployment tool (we recommend the serverless framework or the open sourced version),

## Documentation

Documentation is key. Please use SOLUTION.md to document the process that you took and describe your solution and anything else we should know.

It may include things like assumptions, decisions, diagrams, deployment instructions, etc.

# Part 1

---

Create a DynamoDB data model here are examples to consider.:

Offers


```
[
{
"name": "Super Duper Offer",
"id": "d9b1d9ff-543e-47c7-895f-87f71dcad91b",
"brandId": "692126c8-6e72-4ad7-8a73-25fc2f1f56e4",
"locationsTotal": 0,
"description": "5% cashback"
}
]
```


---

Locations


```
[
{
"id": "03665f6d-27e2-4e69-aa9b-5b39d03e5f59",
"address": "Address 1",
"brandId": "692126c8-6e72-4ad7-8a73-25fc2f1f56e4",
"hasOffer": false
```

```
},
{
"id": "706ef281-e00f-4288-9a84-973aeb29636e",
"address": "Address 2",
"brandId": "692126c8-6e72-4ad7-8a73-25fc2f1f56e4",
"hasOffer": false
},
{
"id": "1c7a27de-4bbd-4d63-a5ec-2eae5a0f1870",
"address": "Address 3",
"brandId": "692126c8-6e72-4ad7-8a73-25fc2f1f56e4",
"hasOffer": false
}
]
```


**NOTE: The above examples are not what is expected to be stored in the database but actually what the client would get from the API, how the data is stored should be decided by you.**

# Part 2

---

Implement a Lambda function with an API endpoint that allows clients to link a location to an offer. When a location is linked:

- The offer's `locationsTotal` counter should be incremented
- The location should be marked as `hasOffer: true`

## Requirements

---

- A location (e.g., “5th Avenue, London”) may be linked to multiple offers (e.g., 10% and 20% discounts)
- Your data model should support an efficient access pattern for retrieving all locations linked to an offer
    - Note: You do not need to implement this endpoint, only ensure the data model supports it.

# Part 3

---

This part will be conducted during the interview process.

You’ll be given a set of requirements and asked to discuss your proposed system design using tools like draw.io or any other diagramming tool of your choice. Be prepared to explain your architectural decisions and trade-offs.