Tokenomy Backend Test

Requirements

- The implementation can be written in any programming language, but we prefer Go if its possible
- The implementation MUST include unit tests
- All of the code MUST be committed into a repository using git
- Inside the repository create a README that explain how to run and test the application

Test scenario

Implement an HTTP GET API that filters dummy data based on value of request parameter.

The HTTP GET API use the following request format,

```
GET /?id=<list of numbers>
```

The query parameter name "id" can contain a single number or two or more numbers separated by comma, for example: "1" or "1,2".
If the "id" parameter is empty it will return all dummy data.
If the "id" value is not valid it must return the error in the response, describing why it's not valid.
If the "id" parameter contains only a single value and the ID is not found, it must return HTTP status code 404 with the response describing the error.

The HTTP GET API return the response as JSON in the following format,

```
{
	"code": <HTTP status code>,
	"message": <Error message>,
	"data": <List of data>
}
```

Dummy data that will be used in implementation, in JSON format,

```
[{
	"id": 1,
	"name": "A"
}, {
	"id": 2,
	"name": "B"
}, {
	"id": 3,
        "name": "C"
    }]
```

Note: You did not need to store this in the database, a single variable on the service itself would be sufficient.

Example of test cases

The following test cases only provided as an example on how the response should be when tested, you may add other possible cases.

Case 1: Request without parameter id

Request

```
GET / HTTP/1.1
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "code": 200,
    "data": [{
        "id": 1,
        "name": "A"
    }, {
        "id": 2,
        "name": "B"
    }, {
        "id": 3,
        "name": "C"
    }]
}
```

Case 2: Request with single id

Request

```
GET /?id=2 HTTP/1.1
```

Response

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json

{
	"code": 200,
	"data": [{
		"id": 2,
		"name": "B"
	}]
}
```

Case 3: Request with multiple ids

Request

```
GET /?id=1,3,4 HTTP/1.1
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
	"code": 200,
	"data": [{
		"id": 1,
		"name": "A"
	}, {
		"id": 3,
		"name": "C"
	}]
}
```

Case 4: Request with invalid ID

Request

```
GET /?id=xxx HTTP/1.1
```

Response

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
	"code": 400,
	"message": "invalid or empty ID: \"xxx\""
}
```

Case 5: Request with ID not found

Request

```
GET /?id=4 HTTP/1.1
```

Response

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
	"code": 404,
	"message": "resource with ID 4 not exist"
}
```

Deliverable

The implementation can be committed in a public or private git repository and shared with us by the link or by archiving the whole repository in tar.gzip format and emailing it to us.