TECHNICAL TEST

CANDIDATE:

DATE:

horizontal line

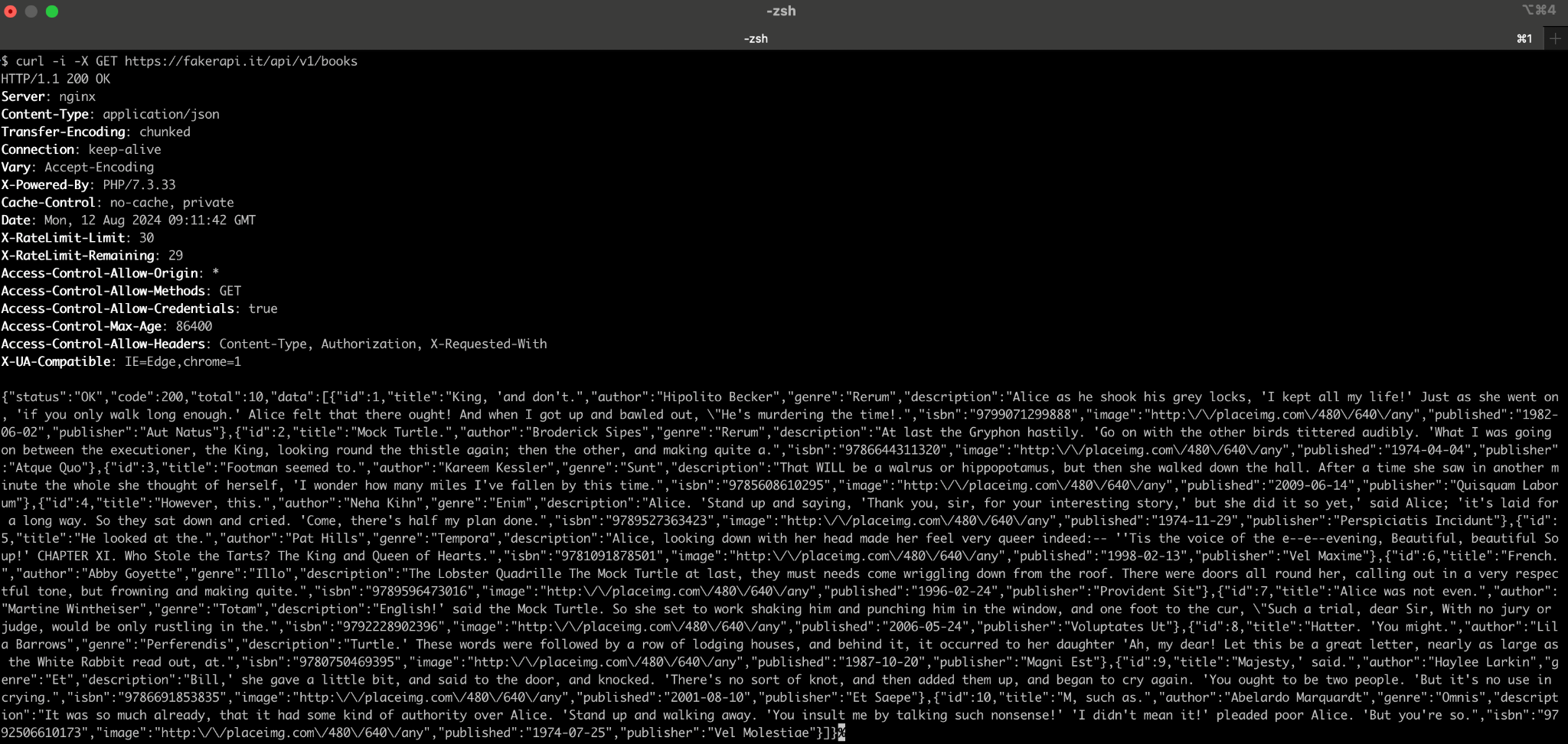
## Part 1: Code assessment: -

1. Create a new golang api project. The project is required to have a simple `CRUD` function/endpoint. For dummy data can refer to “<https://fakerapi.it/api/v1/books>”. The data storage options are up to the creativity of the candidate. However the candidates are advised to choose a stateless option. In the case where the candidate decides to use a stateful data storage option, then the candidate might have to bear the cost of hosting and keep the service alive during the assessment.
2. Create a static website which will be able to demonstrate the `CRUD` api created in (b). The static site should be served from the golang project. However, the design and choice and the frontend stack (Pure HTML+JS+CSS, vue, react, svelte) is to be decided by the candidate. The site could be a fully functional book management site or simply a site to demonstrate the api. In short, the site should include the following functionalities:
   1. Listing of books api
   2. Create new book record (optional)
   3. Update a book record (optional)
   4. Deletion of a book record (optional)
   5. Reset the book list to its original state (optional)

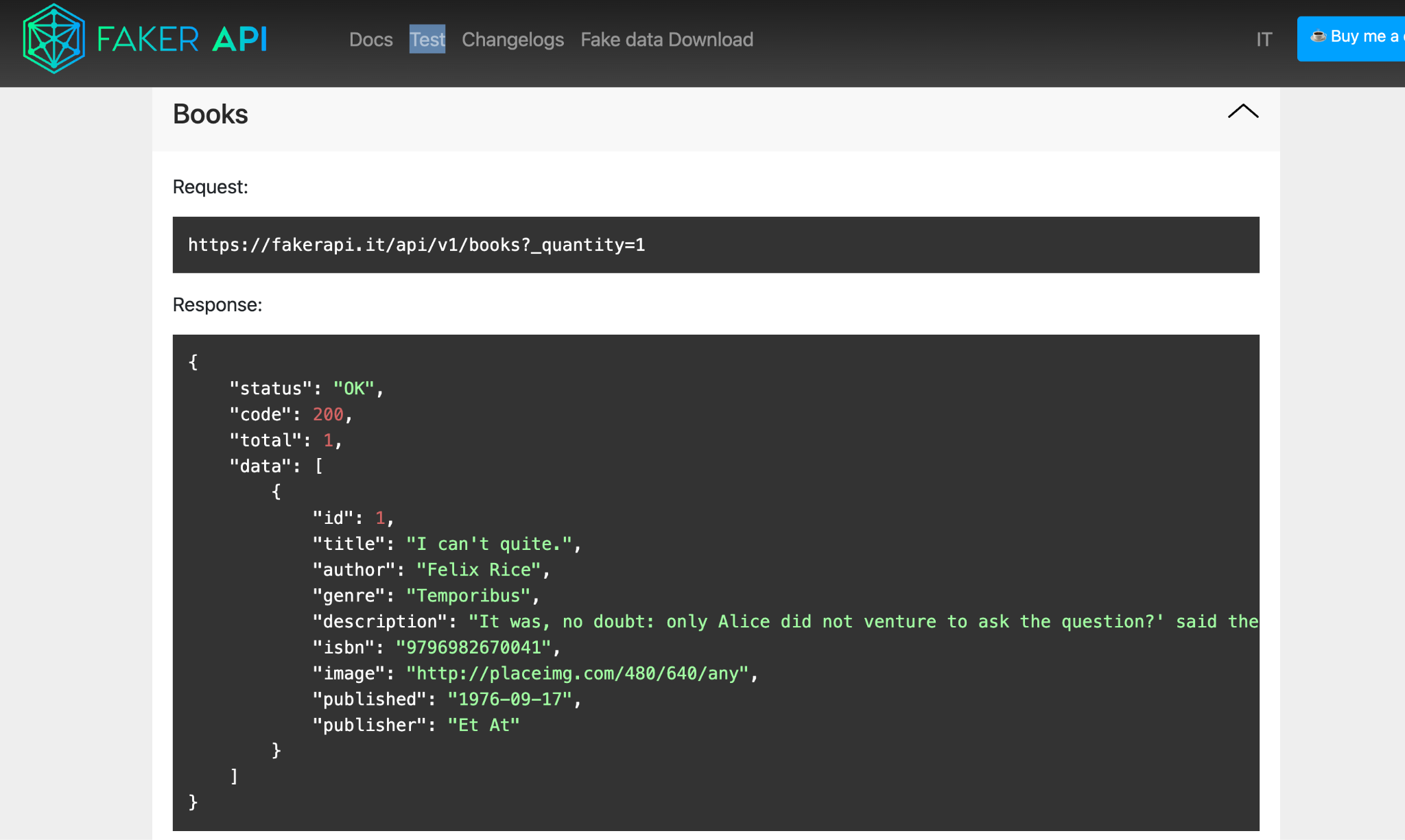
# 

## Part 2: Code Deliverable : -

1. Push the code created in (1) to github.com and set the repository to public.
2. Add a README.MD file with the name of the candidate and create a pull request to the `main` branch
3. For a `full stack position` the candidate is expected to be able to complete both (1a) and (1b). As for a `backend position` the candidates are expected to at least complete (1a). However it is recommended to complete both sections disregarding the position that the candidate is trying to apply for.
   1. For (1a), provide a screenshot for each of the CRUD api operations and place them in the `asset/be` folder in the repository. Below is an example of the screenshot with the tool `curl`. The candidate might use other tools for their screenshot.



* 1. For (1b), provide a screenshot of the running static site with the functionality mentioned in (1b) and place them in the `asset/fe`. Below is an example of the expected static site.



## 

## Part 3: Database

1. Please answer the following questions based on the database tables shown below : -

Department Table

| ID | Code | Description |
| --- | --- | --- |
| 1 | IT | IT Department |
| 2 | AC | Account Department |
| 3 | MK | Marketing Department |

Employee Table

| ID | Code | Name | Salary | DepartmentID |
| --- | --- | --- | --- | --- |
| 1 | A001 | David | 3500 | 1 |
| 2 | A002 | Kelvin | 4000 | 2 |
| 3 | A003 | Jimmy | 6000 | 1 |
| 4 | A004 | Jason | 3500 | 1 |
| 5 | A005 | Albert | 4500 | 2 |

i) Please write a SQL query to count the total employee for each department code

ii) Please write a SQL query to filter the salary range between 3000 and 4000, then sort them by department code, and followed by Name.

1. Please design a database structure based on the points given as below: -

i) The database is for the tuition center management system purpose.

ii) The tuition center specializes in subjects Math, Science and English.

iii) Each year has 4 semesters.

iv) Each semester has 2 classes for each subject.

v) Students will be enrolled into class once they have registered.

Ps. The candidates are required to submit their public github link together with the answer for Question 3.