For this assignment, you must develop a REST API for the fictitious beer importer that we used in Assignment One as well *as* writing a brief report (see task 7).

The starter code for this assignment is largely the same as the starter code you used for Assignment One, however, there are some minor changes (the inclusion of images and a minor tweak to the DB script).

1. You must expose the beer and brewery tables in the *GlobalBeers* database with a suitable API. The following is **required** functionality for the **beers** table where you must support the GET ᵗ, POST, PUT and DELETE HTTP methods. All responses from these HTTP methods should be in JSON.

    ᵗ GET an individual beer based on id.

    ᵗ GET all beers (appropriate use of pagination is required).

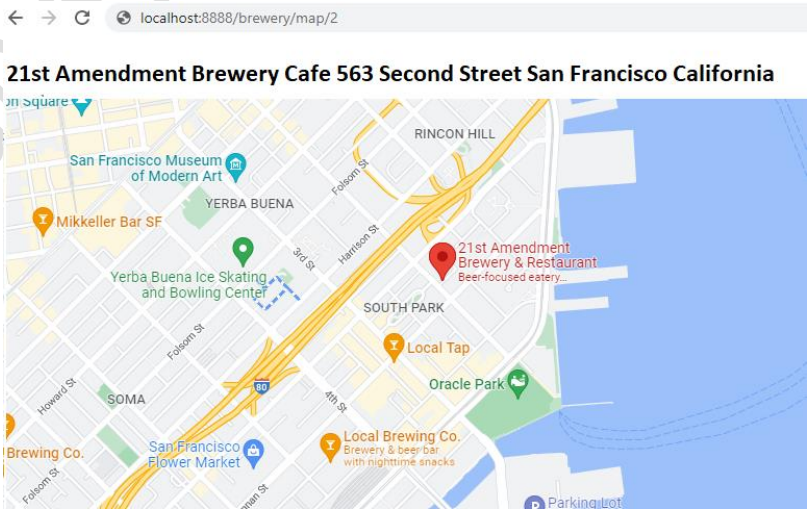
    Both GET requests should adhere to HATEOAS principles:

    1. The GET an individual beer should include a link to all beers.
    2. The GET all beers should include two links. One "self" link and another link that when followed will display the

    You should aim to be **creative** and **innovative** around the API for the **brewery** table and should look to avoid rehashing the functionality you develop for the beer table. Feel free to use 3rd party API's/libraries to help you with this task as well as **varying the request/response types the methods consume/produce.**

    At least one of the GET and one of the POST requests for this task should be consumed by a client application written in a language other than Java.

2. Your API should return, for a specified brewery, a (Google) map with the address of the brewery plotted on it. The name **and** address of the brewery must also be displayed on the *html page* that is returned from API. For example:



You must **not** create a view for this feature. The API call simply returns the relevant HTML with the address of the brewery plotted on a map.

3. Your API should return, for a specified brewery, a QR code, that when scanned (by your phone for example) will attempt to add a brewery to your phone's list of contacts. For example, the following image is returned when a QR Code is requested for Brewery #2 (21st Amendment Brewery Café).

If you scan the above code, it will attempt to add a contact to your phone for *"21st Amendment Brewery Café".* When a contact is being added to your phone, the following data should be recorded at a minimum for each brewery:

- Brewery name
- Brewery address
- Phone number.
- Email address.
- Web address.

I used the ZXing (pronounced Zebra Crossing) library to produce the above QR code. There are other libraries available, and some even sit on top of ZXing.

4. Your API should return, for a specified beer, an *image*. As part of the request, the client should be able to specify if they require a *large* or *thumbnail* image. You should only write one method to accomplish this task (and not one to handle a large image and another to handle a thumbnail).

5. Your API should return a compressed (zipped) file containing all beer images.

6. Your API should return, as a PDF, a brochure (in other words, promotional material) for a specified beer. You must include the following in the PDF at a minimum.

    - Beer Name (from *beers* table)
    - ABV (from *beers* table)
    - Description (from *beers* table)
    - Image (from *beers* table)
    - Sell Price (from *beers* table)

- Brewery name (from the *brewery* table)
- Website (from the *brewery* table)
- Beer category (from the *category* table)
- Beer style (from the Style table)

You must adhere to best practice when developing your REST API and it must be intuitive to use.

7. Along with your code, you must also upload a report to Moodle. This report (including a title page) is to be divided in three parts.
   - **7.1 Documentation**. You must document your API as you see fit (possibly using some 3rd party software).
   - **7.2 Self-Evaluation**. Critically appraise the strengths and weakness of the API you have developed (circa 500 words).
   - **7.3 Benchmarking and Enhancements**. Measure the benefits that the creation of an API brings to a project and what enhancements you feel the API would benefit from (circa 300 words).

Marks Breakdown:

Task	Marks (sum to 300)
1 Core API Development.	100
2 Brewery plotted on a Map.	20
3 QR Code containing a contact for a brewery.	25
4 Image returned for a specified beer.	20
5 A compressed file containing all beer images.	25
6 Return a PDF	30
7API Documentation + Report.	80

Notes and Stipulations:

1. This assignment is due on April 1st at 4pm. I am using Github classroom to manage this assignment and you are required to commit your work to the repository before 4pm each Friday. The reality is that you will more than likely commit much more regularly than that. For each commit that you miss your final mark will be reduced by 5%.
2. You must upload your final solution to Moodle. The version you upload to Moodle should correspond to the final commit on GitHub. Your solution must include the client that you developed for task 1).
3. GET requests can be tested in a web browser. All others can be tested using Postman or equivalent.
4. This assignment is to be developed using **Spring Boot**.
5. You must use **MAVEN** to manage your projects dependencies.
6. You must use **the H2 Database.**
7. You must use suitable **bean validation** where necessary.
8. You must use (embedded) **Tomcat** as your server/container.
9. Use appropriate measures to ensure that all **errors are handled** gracefully.
10. Your report is to be submitted in **PDF** format.