Please answer the following set of questions. Provide your response via email, referencing additional resources (gists, public Git repos) where appropriate. Feel free to describe any assumptions you had to make. If you need any clarification or have questions, do not hesitate to contact us.

1. You are creating a new project from scratch. What is on your to-do list for getting the project ready for deployment, aside from writing code? How do you ensure code quality? How do you ensure your main branch is always in a releasable state? How do you deploy the code?

2. You have received a bug report with the following exception: `UnicodeEncodeError: 'ascii' codec can't encode character u'\xf6' in position 1: ordinal not in range(128).` It references the function found here, specifically the last line. How do you fix it? What does this tell you about the Python environment that the application is running in? Are there any other things you might fix about this function? Provide your fixed function as a gist, and any commentary in your response email.

3. What is your thought process in ensuring your code has proper test coverage, both when seeking to merge a specific pull request and more generally for a project? What test frameworks do you have experience with?

4. Discuss what issues you see with how errors are handled in the function found here.

5. If you create a class in Python and use multiple base classes, in what order are classes checked to resolve method calls? Using the example here please write a Python program that can check, at runtime, if your answer is correct. Provide your answer as a gist. If you can, speak to how this may differ in Python from another language you are familiar with.

6. What is problematic about the code here, and how could it be improved?

7. If you were starting a project from scratch, what do you prefer to use for authentication, cookies or web tokens? What are some advantages and disadvantages to each? What are the security trade offs?

Engineering Task

In order to better assess how you work, we believe it is useful to have you create a small application which can be used as a point of discussion between you and an interviewer, as well as a consistent point of comparison between you and other candidates. The goal is to produce a fully working (if limited) service. In addition to submitting code, be prepared to:

- Present your solution and demonstrate its use, end-to-end. - Discuss major design decisions, alternatives, and why you chose what you chose. - Discuss runtime characteristics of the system.
- Discuss how the service would be deployed.

Understand that your solution needn't be the most advanced - it simply needs to meet the minimum specification. Being able to communicate the weaknesses and how you would resolve them is sufficient, and there is some ambiguity here specifically to allow for you to make decisions and discuss the pros and cons of the approach you took. If anything feels sufficiently ambiguous that you feel it's difficult to understand what we want please reach out.

Problem Statement

Using Flask and any backing datastore you choose, implement a service which exposes two endpoints, described below. Ensure that your service handles expected error conditions gracefully, and returns HTTP status codes in line with industry norms (we aren't here to be dogmatic about whether 400 or 422 is appropriate for valid JSON that fails schema validation - just don't be returning 500 for that. You may be *asked* about your choice of return code, but primarily from a perspective of understanding your thought process rather than because an answer is necessarily wrong).

Not strictly required, but desirable if you are comfortable with it is to Dockerize your solution, presenting us with a `docker-compose.yml` which handles running all required services.

This service is for a library, and we are adding a new `/request` endpoint which allows a user to request a book by title so that they can later be emailed when it is available. Endpoints should accept and return valid JSON. You are not required to consider authentication or

authorization for this service.

Add Request Endpoint

This allows a user to request a particular book, and returns the request object along with an id by which it can be referred to. Ensure `email` is of a valid format, and that `title` exists in our list of titles in the database. You can seed this set of values as part of application startup or however you see fit.

`/request`
`[POST]`

- **`email`** `{string}` the email address of the requesting user
- **`title`** `{string}` the title of the requested book

Returns a JSON response with the following:

- **`email`** `{string}` the email address of the requesting user
- **`title`** `{string}` the title of the requested book
- **`id`** `{string}` an id which the newly created user can be referred to by
- **`timestamp`** `{string}` a timestamp set to the date and time that the user made the initial request.

Retrieve Request(s) Endpoint

This allows a user to retrieve an existing request by using an `id`, or a list of all existing requests if `id` is omitted.

`/request/{id}`
`[GET]`

Returns a JSON response with values depending on whether `id` was provided. If it was, there should only be one value, and it should match return value of the `POST` endpoint above. If there is no `id` given, it should be a list of all requests, each being in the same format as a single one would be.

Delete Request Endpoint

This allows a user to remove an existing request.

```
/request/{id}
[DELETE]
```

Returns no body.

Deliverables

- Source in a public Git repository. - Installation and running instructions. - Demonstration for an interviewer, as a prompt for further discussion.

Reminder

- Don’t spend more than a couple of hours on this. The main things we are looking for is code quality, organisation, and proper quality assurance methods. While your solution should be complete, it needn’t be production ready or capable of withstanding heavy traffic. - Remember, it’s not meant to be perfect, but rather serves as a prompt for further discussion. Explaining choices made, even if the reasoning was “I didn’t have time” are fine, as long as you can talk in depth about the thing you would have implemented had you had more time.