Full Stack Engineering Exercise

Overview:

Brightwheel uses internal and third party API's extensively throughout our platform to process and store data, as well as manage communication to end users. The goal of this exercise is to build and consume an internal API and display the data according to the specifications outlined below. This exercise is meant to assess full stack coding ability, code quality, use of of object oriented principles and your aesthetic.

In order to prevent downtime during an email service provider outage, you're tasked with creating a service that provides an abstraction between two different email service providers. This way, if one of the services goes down, you can quickly failover to a different provider without affecting your customers.

Specifications:

Please create an HTTP service that accepts POST requests with JSON data to a '/email' endpoint with the following parameters:

- `to` The email address to send to
- `to_name` The name to accompany the email
- `from` The email address in the from and reply fields
- `from_name` the name to accompany the from/reply emails

- `subject` The subject line of the email
- `body` the HTML body of the email

Example Request Payload:

```
{
  “to”: “fake@example.com”,
  “to_name”: “Mr. Fake”,
  “from”: “noreply@mybrightwheel.com,
  “from_name”: "Brightwheel",
  “subject”: “A Message from Brightwheel,
  “body”: “<h1>Your Bill</h1><p>$10</p>”
}
```

Your service should then do a bit of data processing on the request:

- Do the appropriate validations on the input fields (NOTE: all fields are required).
- Convert the ‘body’ HTML to a plain text version to send along to the email provider. You can simply remove the HTML tags. Or if you’d like, you can do something smarter.

Once the data has been processed and meets the validation requirements, it should send the email by making an HTTP request (don’t use SMTP) to one of the following two services:

- Mailgun
  - Main Website: www.mailgun.com

○ Simple Send Documentation:

- http://documentation.mailgun.com/quickstart.html#sending-messages

- Mandrill

  ○ Main Website: www.mandrillapp.com

  ○ Simple Send Documentation:

  https://mandrillapp.com/api/docs/messages.JSON.html#method send

Both services are free to try and are pretty painless to sign up for, so please register your own test accounts on each.

Your service should send emails using one of the two options by default, but a simple configuration change and redeploy of the service should switch it over to the other provider.

Implementation Requirements:

Please do not use the client libraries provided by Mandrill or Mailgun. In both cases, you’re making simple post requests do this with a lower level package or your language’s builtin commands.

This is a simple exercise, but organize, design, document and test your code as if it were going into production.

Please include a README file in your repository with the following information:

○ How to install your application

○ Which language, framework and libraries you chose and why
○ Tradeoffs you might have made, anything you left out, or what you might do differently if you were to spend additional time on the project
○ Anything else you wish to include.

Provide us the link to your final product as a cloneable repo on github or bitbucket.

Bonus Points:

You’ll receive bonus points for well written documentation and tests, and anything you do that goes above and beyond the implementation requirements.

How we Review:

- Functionality: Does the app do what we asked?
- Code Quality: Did you stick to OO principles? Is the code easy to understand and maintainable? Is it well tested?
- Technical Choices: Do your choices of libraries, architecture etc seem appropriate for the chosen app?