Payment Service

Abstract:
This document describes the specifications for the initial backend implementation of Griddy's payment service.

Background:
Griddy's pay-as-you-go automated billing system debits users daily based on their usage, leading to a high volume of frequent transactions. Given the billing granularity, statements made available to users must of course be detailed, but also as real time as possible. In an effort to balance ease of integration and reliability of information, we've decided that a Stripe integration would best meet our business needs.

Goals:

- Stand up a payment service to reliably capture and report on payments across our user base

Non-Goals:

- For ease of development, we can defer the implementation of an authentication mechanism.
- Though encouraged, persistence across server restarts is not required for this initial implementation

Implementation:
The backend will implement the logic for the following endpoints:

POST: /payments

This endpoint allows clients to hit our servers with the following payment information JSON request body:

```
{
    "account_id": <uuid string identifying the user account>,
    "amount": <string represent the payment amount to be captured>,
}
```

Response JSON:

```
{
    "id": <string identifier representing the payment>,
    "amount": <string representing the payment amount>,
    "status": <string representing the Stripe status of the payment>
}
```

Response Status Codes: 200 if success, 400 if not

GET: / {account_id} /payments

This endpoint allows clients to poll for payments associated with the specified **{account_id}**.

Response JSON:

```
{
    “payments”: [
        {
            “id”: <string identifier representing the payment>,
            “amount”: <string representing the payment amount>,
            “status”: <string representing the Stripe status of the payment>
        },
        <...payment objects...>
    ]
}
```

Response Status Codes: 200 if success, 400 if not

Future Enhancements:

We’d like to run this system at scale to support hundreds of thousands of simultaneous Griddy debiting events and user read operations across multiple nodes. What are some limitations of your existing implementation and how could they be improved upon in upcoming iterations?

Fill in with your response in the submission response file

Open Questions:

List of unresolved implementation questions. Please only fill in (in the submission response file) if relevant.