What concerns would you have from a testing perspective?
As part of Github QA team tasked to validate the end points, below would be my concerns/questions:
- Necessary test environment, database, server, application setup needs to be prepared before Testing.
- QA needs to understand the functionality of the each and every API program and clearly define the scope of the program.
- Input Parameters for the API need to be planned and defined.
- Validating our API limits will need mimicing of virtual requests which will need enough time and infrastructure
- What API's Call other API's/Events
- Version Test Q's
    * What happens when user use old versions
    * what happens when user use versions which does not exists
    * Are older version supported
    * What happens if there is a version completely depreciated
- Area's of importance
    * Wrong params/body to validate error messages
    * API limit tests
    * Authentication
- Automated testing does not eliminate the need for manual testing; some things can only be done manually.
- Will need the Dev team to create a POC connecting to API's so that QA team can validate the execution of the POC.

How would you go about tackling the QA for this work?
- Test Plan
    * What to test, how to test, entry and exist criterias

- Test Setup
    * Create the prerequisites like mock applications
    * Test Repos
    * Test Suite and Ability to run, collect and view test results

- The approach would be to build up a wrapper covering core part of communicating with API such as
    * versioning
    * HTTP requests and responses
    * Params validation
    * Error Messages validation

- Type of Testing
    * Black Box majorly

What sort of tests would be worth describing or worth automating?
In general, tests where lot of repetition and retest is needed.
It is not usually cost effective to automate a test if it will be executed only a few times, and a manual test is the easiest way to quickly verify new functionality.
In the current discussion of Github end points testing, all extensible & customizable API's and critical areas are worth automating.
Example - commits, pull requests and merges

What tools would you use?
- Postman or Wiztools rest client
- Ruby and its gems/libraries
- AWS
- Blaze meter or Jmeter for performance

2. Assume you are part of an engineering team that is building a loyalty app for a large retailer. You are in a meeting in which the following stories are being discussed by the product owner and engineering team:
a) As a customer, I want to enroll in the loyalty program.
b) As a program participant, I want to check my balance of reward points.
c) As a program participant, I want to redeem some of my points for a reward.

Describe how you might participate in this meeting to ensure that the development work for these stories can be demonstrated to the product owner. What are your areas of concern? How would you address them?

Concerns:
- How is a customer identified?
- Are there roles in the system other than customers?
- What is the enrolling process of existing customer vs new customers? What would be a difference?
- Assuming the reward are assigned based on their buying activity, do we calculate and preassign rewards to existing users?
- Explain the rewards points program? How any points are given for what activity?
- What does a user get when redeeming the points?
- Is there a cut-off for redmeeing the points? Does user has to reach total no of points to redeem, what is that number?

QA has the best-end-to-end view for an application and thus they are better equipped to think holistically and act as liaisons between the engineering team and the product owner. QA team can work closely with the product owner to write detailed acceptance criteria for the user stories. Based on what the team learns after the retrospective meetings, QA can help the product owner modify or enhance the existing user stories to better reflect the true requirements.