CognitiveScale Testing Exercise

First Section of Exercise:

Using the documentation of the GitHub API at < https://devlooper.github.com/v3/ >
Suppose that you worked with a team that was tasked with implementing this specification.

- What concerns would you have from a testing perspective?
- How would you go about tackling the QA for this work?
- What sort of tests would be worth describing or worth automating?
- What tools would you use?

Please note:: per instructions I have placed code and scripts for review up in a public repository of mine on github under hacktheodds/githubtests. I'll describe in detail what is there at the end of this document.

Answer to the first question -- (What concerns would you have from a testing perspective?)

This the GitHub specification that is being proposed to be tested. It is a highly popular and highly used web-based tool in the software development community and is very frequently a key part of the software development workflow of any software development team. Many teams consider it essential and a foundational tool of that simply must work. It is a software version control system that provides for the tight coordination of merging and consolidation of code and supports a high degree of automation. This makes it an integral part of software development teams continuous integration efforts. Third party tools that are part of the overall work flow process that many teams use rely on GitHub to keep everything in order. Even leaving out the UI and focusing only on the API there is a large surface area that needs to be tested and on what is a highly visible and highly trafficked feature set. Given collaborative team dynamics my general approach would be to first itemize and break down into documentation each of the major features listed in the spec. All items listed in the spec are an important concern and weigh in a similar way to any highly functional product that delivers a high degree of value to the customer base.

Despite this I have a higher degree of concern for what I see are key items that must be addressed during testing:

(1) File and code check ins and versioning paths
(2) Third Party CI/CD integration and
(3) User Authentication, Authentication and Security
(4) Account Management

These things must work correctly or none of the other features matter. Because it is a web-facing product and has an extremely large volume of traffic with many teams relying on it to securely store and track their very valuable assets, in my opinion, security and account management rate very highly as items to be tested and give me cause for the most concern.

The versioning, code check in capabilities as well as CI/CD integration are core functionality that are central to the value proposition of the software utility that GitHub offers and these are critical parts of the testing effort.

Answer to Second Question:
(How would you go about tackling the QA for this work?)

In almost any testing task my approach starts with getting a good understanding of the problem. This includes all everything that goes into scoping and sizing up the problem space. Then I start lining up my resources and tools, which may or may not require some research and then after that follow up with solving the problem and planning the work. The first thing that comes to mind is attempting to finding out who else has worked on this problem and what kinds of testing tools have the used or created. I'll use problem description as the guide to do a quick round of research online. During my research of tools during my evaluation this time I found a very thorough and well organized open source effort performed by members of the Python community who created an API interface intended to provide a very well integrated API solution for Python. I am including this in my published results as I feel it is a very relevant guide to my testing efforts in this case. The tools I found most useful and relevant for testing in my research were the following:

(1) PyGithub – Python based GitHub API integration
(2) SoapUI – good for automating security testing
(3) Postman – great for quick generation of python test scripts
(4) Circle Ci – continuous integration

Why did I choose these tools over others? For me these are all highly recognized and widely used set of tools. The REST standard is not new and these tools would offer a very good base to start from. To begin with I am more familiar with Python over other languages and it has one of the larger and more dynamic communities and broader base of support where answers to a broad range problems can very quickly be found on the web. I am certain there are other languages that have similar solutions but Python is definitely able to get the job done here. At the very least it is a really good place to begin the effort and has proven to have been able to do the job. Since this is a testing task we are not overly concerned with throughput and performance that we would be lead to choose a lower level compiled language such as Java, C++ or C#. I see SOAPUI as a solid tool that could be used to perform various types of vulnerability testing which is built in part of its tooling and would make for a ready made way to get high level security testing done on an web-based API. Postman is low cost tool that allows for rapid prototyping of API calls that provides a converter that allows for rapid generation of Python scripts. Circle CI is CI/CD build scheduling and hosting tool that would allow for rapid coordination and deployment of the testing effort from a team perspective. Circle CI is a well recognized and highly used tool and I would consider a tool like this a vital part of the effort in being able to leverage the skills and abilities of the team.

After selecting the tools I would then focus back on the spec. I would begin breaking down testing by itemizing all the different endpoints. Once these tasks were done I would correlate the endpoints to my top concerns listed above in my first answer:

(1) File and code check ins and versioning paths
(2) Third Party CI/CD integration and
(3) User Authentication, Authentication and Security
(4) Account Management

This would yield my overall structure and approach to the testing effort.

Answer to Third Question:
(What sort of tests would be worth describing or worth automating?)

Certainly most endpoints would be good candidates for automation though we can be sure that not all flows are easily automatable. The PyGithub project applied an extensive number of tests and many of these centered on functionality exposed in endpoints. In addition to this they performed a large amount of exception testing and these are very worthwhile. Test automation of CI/CD workflows is easily achievable and very worthwhile in my opinion

Answer to Fourth Question: (What tools would you use?)
Of course I did already answer this to some degree above but here's a more complete list:

(1) Clearly written expectations(stories) of what the new Github API is supposed to do
(2) A well written set of test plans including well defined entrance and exit criteria
(3) The command line with access to the internet or the GitHub instance being tested
(4) Jira for issue and feature tracking and team scheduling
(5) A well configured build server running Circle CI or Jenkins
(6) Git Client and Server Side Hook Script commands
(7) Python and the other tools already mentioned (SOAPUI, Postman, CircleCI)
(8) Sample files to commit
(9) OWASP Rest Security cheatsheet
(10) Use of a Proxy to collect requests for analysis
(11) An adequate test environment to test on (client and server resources)
(12) Clearly specified delivery dates.
(13) Well defined placement and integration with the teams larger schedule
(14) An Agile methodology

Second Section of Exercise:

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

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.

My answer:

Generally my approach toward participation in a design or story point meeting is to ask questions about new potential features and service offerings. As a test engineer I am usually in the role of supporting a new initiative which typically means that others are making design and architectural decisions. I have found that the Socratic method goes far in creating a dialog and understanding. Asking good questions is the best way to raise concerns I might have. Given this particular scenario and the descriptive stories offered I would ask the following questions

(1) How tightly coupled is the loyalty program expected to be with the existing systems already deployed?
(2) What kind of performance impact does the new loyalty program have on a system with a known existing load and site traffic? What happens with an increased or increasing load and site traffic? Are there any new performance thresholds introduced?
(3) Will there be any phased rollout or A/B type of testing expected to employed? What will this look like and on what kind of schedule?
(4) How much database work will be required and are there any potential conflicts with the current table structure?
(5) How many locations and access points (links) in the user interface is the loyalty program information presented?
(6) Where in the interface can the reward points be viewed and what are the paths to view that information?
(7) How are the points redeemed and how specifically can they be applied to a purchase?
(8) Is there expected to be an expiration system applied to the reward points?
(9) Are these reward points tied into any external system? What kind of impact is expected? Is there any extra integration work or customization required?
(10) What kind of installation impact is expected to occur with existing systems already in the field.
(11) Is there any customization or professional services and customer services support required for deployment?
(12) If any what is the projected timeline for delivery of the feature set.

Addressing these concerns more thoroughly is contingent on their answers. Depending on the answers provided in return I would know what kind of resources to commit to supporting the new loyalty program feature set and it would help me to know what to expect and correspondingly how to scope and what to plan for. I would have an idea of any extended impacts and demands to my schedule and the resources and the people needed to assist me in performing the testing tasks.

At the very least I would want to know the projected schedule of the stories and a better understanding of where to begin my efforts. I could then plan generally to look for the features and their corresponding sets of data.

Description of GitHub Files

I've included two sets of files in two directories in the githubtest repository. The one under postman are some Python scripts using postman to generate the query and the python stub. The one under github is the much larger open source PyGitHub effort which I took a close look at and would use as reference to model the test effort described in this exercise. Using that structure I was looking closely at the circle ci integration and how that would tie into providing what I think would be the CI/CD portion an subsequently very important part of the overall testing of the API. The postman tests are only a few representative tests showing the Search query API endpoint.