https://www.instagram.com/developer/endpoints/comments/

API test strategy

Purpose

- Prove implementation is working correctly as expected.
- Ensure implementation is working as specified, according to requirements specification.
- Provide useful information when tests fail so that it is easy for developers to reproduce and fix.
- Prevent regressions between releases

Approach

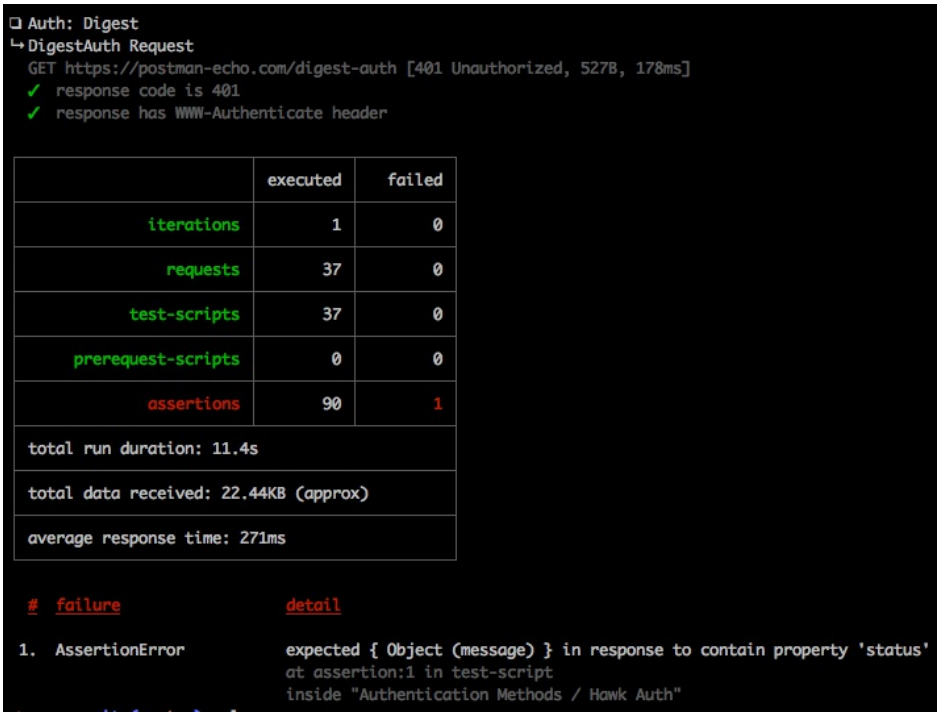
- Fully automated test system implemented in nodejs.
- Autorun after API deployment.
- After running, create a clear report of all test cases: request, expected response, actual response

Simple unit test for this API

Use **newman**, in conjunction with CI build to run API testing right after deployment process completed.

Sample project (to describe how to use newman in a nodejs project):
https://github.com/dinhnhat0401/api_test/blob/master/test/run-collections-in-directory.js
https://github.com/dinhnhat0401/api_test/blob/master/test/collections/sample-collection.json

Test coverage report is supported by newman.
Sample test coverage result: https://github.com/dinhnhat0401/api_test/tree/master/newman



Scenario test possible for the API

A. Incorrect handling of valid argument values
B. Fails to handle error conditions gracefully.
C. Inconsistent error handling
D. Performance
E. Multiple concurrent requests issue
F. Stress test

A. Incorrect handling of valid argument values

GET/media/media-id/comments

- Test 1: valid media-id, valid access_token, expect status code 200
- Test 2: valid media-id, valid access_token, expect response has valid json body
- Test 3: valid media-id, valid access_token, expect response object contains attribute `data`, response.data is empty array
- Test 4: valid media-id, valid access_token, expect response object contains attribute `data`, response.data is non-empty array, response.data[0].created_time is string, parseInt(response.data[0].created_time, 10) is valid timestamp, response.data[0].text is string, response.data[0].id is string, response.data[0].from is object
- Test 5: valid media-id, valid access_token, expect response object contains attribute `data`, response.data is non-empty array, response.data[0] to have property created_time with value 1280780324, response.data[0] to have property text with value `Really amazing photo!`, response.data[0] to have property id with value 420,

POST/media/media-id/comments

- Test 1: valid media-id, valid access_token, valid text, expect status code 200
- Test 2: valid media-id, valid access_token, valid text, expect response has valid json body
- Test 3: valid media-id, valid access_token, valid text, expect response has valid json body, json body contain `meta`, json().meta to have property code with value 200, json().data is null

DEL/media/media-id/comments/comment-id

- Test 1: valid media-id, valid access_token, valid comment-id, expect status code 200
- Test 2: valid media-id, valid access_token, valid comment-id, expect response has valid json body
- Test 3: valid media-id, valid access_token, valid comment-id, expect response has valid json body, json body contain `meta`, json().meta to have property code with value 200, json().data is null

B. Fails to handle error conditions gracefully.

GET/media/media-id/comments

- Test 1: invalid media-id, valid access_token, expect status code NOT equal 200
- Test 2: valid media-id, invalid access_token, expect status code NOT equal 200
- Test 3: valid media-id but scope is NOT contain public_content, valid access_token NOT the owner of media-id, expect status code NOT equal 200

POST/media/media-id/comments

- Test 1: invalid media-id, valid access_token, expect status code NOT equal 200
- Test 2: valid media-id, invalid access_token, expect status code NOT equal 200
- Test 3: valid media-id (scope is NOT contain public_content), valid access_token NOT the owner of media-id, expect status code NOT equal 200
- Test 4: valid media-id (scope is NOT contain comments), valid access_token NOT the owner of media-id, expect status code NOT equal 200
- Test 5: valid media-id, valid access_token, text with 301 characters, expect status code is NOT equal 200
- Test 6: valid media-id, valid access_token, text with 1000 characters, expect status code is NOT equal 200
- Test 7: valid media-id, valid access_token, text with 5 hashtags, expect status code is NOT equal 200
- Test 8: valid media-id, valid access_token, text with 2 URL, expect status code is NOT equal 200
- Test 9: valid media-id, valid access_token, text with all characters capitalized, expect status code is NOT equal 200

DEL/media/media-id/comments/comment-id

- Test 1: invalid media-id, valid access_token, expect status code NOT equal 200
- Test 2: valid media-id, invalid access_token, expect status code NOT equal 200
- Test 3: valid media-id (scope is NOT public_content), valid access_token NOT the owner of media-id, expect status code NOT equal 200
- Test 4: valid media-id (scope is NOT contain comments), valid access_token NOT the owner of media-id, expect status code NOT equal 200

C. Inconsistent error handling

Try each test scenario described in B. Fails to handle error condidtions gracefully 5 times(or more)
Expected results: response status code and reponse json is consistent.

D. Performance

Call the API with appropriate params.
Expected results: response is received in less than α ms (e.g: 200 ms)

- Sample:
  https://github.com/dinhnhat0401/api_test/blob/master/test/collections/sample-collection.json#L54

E. Multiple concurrent requests issue

Create 2 or more threads.
On each thread, call same API simultaneously.
Expected results:
Response time should not be greater than α seconds for β percent of total request completed.

- Sample:
  https://github.com/dinhnhat0401/api_test/blob/master/test/parallel-collection-runs.js#L29

F. Stress test

Use tool like JMeter to stress test request to api server with first 10 concurrent users increasing concurrent users +10 each time. Monitoring server CPU to see when it around 100%. Check if capacity of server is enough to handle desired request number.

Expected results:
Application hosting process should not recycle because of deadlock or memory consumption.
Response time should not be greater than α seconds for β percent of total request completed.
“Server busy” errors should not be more than 10 percent of the total response because of contention-related issues.