Google Geocoding Test Plan

Introduction

Geocoding is the process of converting addresses into geographic coordinates. Google geocoding API provides an endpoint for HTTP request allowing user to retrieve the address information either by querying the address or latitude/longitude. This test plan will identify the test strategy, scope and deliverables for geocoding API tests.

API endpoint will accept GET request and will return JSON/XML response with address details. The test plan will majorly focus on testing the response field with different request parameters. The assumption is that time frame for testing effort is limited to less than one day and the test suite will be triggered hourly in CI.

Limitations / Out of Scope

The test plan will have following limitations due to limited time and resource. More detailed discussion will be in Next Steps session later.

1. JSON/XML Schema validation
2. XML response verification
3. HTTPS connection with proper API Keys
4. Limited address type
5. Optional parameter combinations
6. Localization
7. Region/Viewport Bias
8. Performance benchmarking/load/stress testing
9. Real-world data loading and testing

Scope

Functionality Tests

1. Required ‘’address’ parameter – querying street address, verify response with each field, numbers of address components, number of results, status code, latitude/longitude, bounds, formatted address
2. Test with different Address Type: street address, route, political, city, administrative level 1..5, country, postal code, neighborhood, airport, locality, establishment
3. Required “lat/lng” parameter – query with latitude/longitude combination, with int/float data type, result should be the same with query with accurate street address
4. Inaccurate/Typo in address – verify approximate result returned or address recommendation based on input.
5. Optional “bounds” parameter – accepts integer and float, returns approximate address type. When address is not found in bounds specified, the parameter will be ignored.
6. Component filter – ‘components’ filter can be used in combine with ‘address’ or alone. Verify when multiple matches for address entered, only address type specified in filter

will return. If component type is invalid or doesn't match any result, the parameter will be ignored. When using component filter alone, the parameter requires valid input.

7. Optional 'region' parameters – region accepts 2 chars of country code. Only address matched the region specified will return.
8. Special characters in URL – when properly encoded, verify special character in URL is properly handled.
9. Max chars in URL – API accepts URL with maximal 8192 characters.

Negative Tests

1. Missing required parameters – No 'address', 'lat/lng', 'component filter' or 'place_id' parameters, or empty value provided. Verify return 400 invalid request code.
2. Undefined parameters – parameter with random name from user, verify server will ignore it (further expanded in security test)
3. Invalid data type for parameter – String for 'lat/lng', 'bounds' parameter, verify 400 status code with invalid request
4. Out of range for latitude/longitude – provide value with latitude greater than 90 or longitude greater than 180, verify return either invalid request or zero result depend on other parameters
5. Invalid address input – long random generated String/int/float value for address parameter, verify zero result returned.
6. Exclusive/Invalid Component Filter – adding mutual exclusive values for component filters, e.g. country:US|country:FR, verify zero result returned. Using component filter alone with invalid component type, verify invalid request returned.
7. Multiple required parameters – using address and lat/lng parameters together with different address, verify zero result.
8. Invalid Region value – provide value with more than 2 chars for region parameter, verify the parameter is ignored.
9. URL exceeds max characters – verify invalid request with 400 status code
10. Invalid URL – verify 404 status code

Security Test

a. No Authorization – Use 'place_id', 'location_type' parameters in request without API key, user_id or HTTPS, verify the request is denied.
b. Method not Allowed – POST/PUT/DELETE request against the endpoint should return 405.

Performance Test

Current performance test suite is limited. Test 1000 GET requests with 100 threads, record the execution time.

Next Steps

1. Add test suites for XML response verification

2. Add JSON/XML schema validation
3. Security test with SSL and proper API keys, monitor request quota limits.
4. More tests on invalid user input, focused on unsafe characters with proper encoding.
5. Adding more tests to cover address type of intersection, sub-locality, transit-port, park, room, floor, premise, parking, postal-box, point of interests, natural features, ward, colloquial area.
6. Expand on optional parameter combination, verify zero/single/multiple results returned
7. Localization with supported languages.
8. Expand on performance testing. Use load test tool to generate test plans, monitor network throughput, response time, CPU/memory usage, execution queue etc. Have performance benchmarking and identify the bottleneck.

Deliverables

1. Java test functionality/performance automation suites using TestNG and RestAssured framework.
2. Sample HTML test report
3. Gradle build and test tasks to execute test suites.
4. GitHub Repo with source code and instruction
5. Travis CI to run test integrated with GitHub repo.