Agoda - API Testing Plan

Summary

The test plan document tracks the test strategy and different test approaches to be taken in order to test the following two API endpoints.

1. GET {hostName}/fetch/inventory
2. POST {hostName}/update/inventory

Testing Strategy

For both the API endpoints, it is necessary to focus on various types of testing explained briefly below, post which we will define a test scenario table that will be a one-liner of all the scenarios to be executed in the complete testing of the API endpoints.

1. ***Functional Testing:*** In this particular testing type, QA would ensure that the function of the APIs remain in the acceptable range. Different types of tests would be performed to ensure the proper functioning of the API endpoint. This will include but not limited to following variations:
    a. Positive tests, this will include checking API responses, headers using a valid payload and query parameters, also checking expected correct response codes.
    b. Negative tests, this will include checking proper API responses, or Error responses that the API should handle without getting into a bad state. Some examples include sending an empty payload, using incorrect query parameters or using methods that are not supported by the endpoints.
    c. Error Handling: For every bad input, QA would make sure to validate proper error handling both through error response and API response messages.
    d. Boundary Value Analysis: Since these APIs consume query parameters that are a date range, it becomes an important aspect to check API responses for date ranges that lie on the boundary, for example: sending scheduled params at 12:00 AM for two different dates and checking the correctness of the output.

2. ***Non-Functional Testing:*** Non functional testing majorly includes checking the API for its performance (based on expected times.), handling extraordinary usage by spawning multiple users hitting the API at the same time.
    a. Load Testing: Loading the API for let's say 1000 users at the time bombarding the endpoint with different, same query parameters.

    b. Stress Testing: QA needs to make sure that the API endpoint is validated to where it stops working, by stretching the usage of the API by a constant increase in test users, and payload variations / query params.
3. ***Security Testing:*** Security testing is another important part of the API testing domain where we need to make sure correct API keys when passed along in the API help authenticate valid users and prevent/block any unexpected or unauthenticated user base in the system.

Additional Testing Strategies - based on components:

Apart from the umbrella of functional and non functional testing, it is very important to check integration of two system APIs in tandem because one updates data and the other one gets that updated data.

1. ***Integration Testing:*** Running multiple workflows QA would make sure that the requests stay idempotent, when run together, the upserted or inserted data (POST API) should be fetched correctly from the GET endpoint.

Test Scenarios

- Tier 1: Smoke Suite - test critical features.
- Tier 2: High priority tests.
- Tier 3: Medium priority tests.

Test Type	**Scenario**	**Test Action Category**	**Automation Tag**
Functional	Check API response correctness for valid parameters; valid date range, valid hotel_id and valid room_id (individual and combination)	Validate Response JSON	Tier 1
Functional	Check the HTTP methods that are supported by the endpoint work - test for negative scenarios for methods unsupported should be properly handled.	Validate Status Codes	Tier 2
Functional	Check API response for invalid date range, date range in the past. Should be properly handled. <An error prompt to enter correct date range>	Validate exception handling	Tier 1

Functional	Check API response for incorrect hotel_id, proper exception handling	Validate exception handling	Tier 2
Functional	Check API response for incorrect room_id, proper exception handling	Validate exception handling	Tier 2
Functional	Check data should be returned only when required params are sent along with the request.	Validate exception handling	Tier 2
Functional	Check correct data should be for required + optional parameters. In addition, check the following parameters: – filter: ensure the response is filtered on the specified value. – sort: specify field on which to sort, test ascending and descending options. Ensure the response is sorted according to the selected field and sort direction. – skip: ensure the specified number of results from the start of the dataset is skipped – limit: ensure dataset size is bounded by specified limit. – limit + skip: Test pagination	Validate Response	Tier 2
Functional	Check for various other negative scenarios: - Wrong content-type in payload - Content with wrong structure - Illegal characters in parameters or payload		Tier 3
Functional	Check correct headers are returned for valid query params and payload data	Validate Headers	Tier 1
Functional	Check correct status codes are returned for bad requests, unauthenticated requests.	Validate Status codes	Tier 1
Functional	Check correctness of data returned for dates at boundaries, last and first days of the month. Dates, that are going to change in some time, midnight hours	Boundary Value analysis - Validate response	Tier 1
Functional	Check the correctness of the response json, all valid keys should be present. Example Json: { "hotel_id": 1,	Validate response	Tier 1

	"room_id": 101,
	"start_date": "2021-02-19",
	"end_date": "2021-02-21",
	"data": [
	{
	"date": "2020-02-19",
	"remainingRoom": 5,
	"rate": 199
	}
	]
	}		

Functional	Check for proper error Handling: - Empty Payload (POST) - Bad Payload (POST) - Empty query params (GET) - Bad query params (GET)	Validate exception handling	Tier 2
Performance	Check correctness of response returned for a huge payload, POST request.	Load test	Tier 2
Performance	Check response time for response returned for a huge payload, POST request.	Load test	Tier 2
Performance	Check correctness of response returned for a big date range param, GET request.	Load test	Tier 2
Performance	Check response time for response returned for a big date range query param, GET request.	Load test	Tier 2
Performance	Check API response time, latency, TTFB/TTLB in various scenarios (in isolation and under load)		Tier 2
Integration Testing	Run POST request to upsert data, Now run the GET request to check the correctness/accuracy of the data served. - Multiple workflows to be decided for this scenario.	Imitating common user actions.	Tier 2
Security and Authorization	Check that the API is designed according to correct security principles: deny-by-default, fail securely, least privilege principle, reject all illegal inputs, etc. Positive: ensure API responds to correct authorization via all agreed auth methods – Bearer token, cookies, digest, etc. – as		Tier 1

	defined in spec Negative: ensure API refuses all unauthorized calls		
Security and Authorization	Role Permissions: ensure that specific endpoints are exposed to users based on role. API should refuse calls to endpoints which are not permitted for user's role		Tier 2
Security and Authorization	Protocol: check HTTP/HTTPS according to spec		Tier 2
Security and Authorization	Data leaks: ensure that internal data representations that are desired to stay internal do not leak outside to the public API in the response payloads		Tier 3
Security and Authorization	Rate limiting, throttling, and access control policies		Tier 2

CI/CD Integration

Test Suite<Automation Tag>	**CI / CD Schedule**			
	VCS Triggers	Branch	On schedule	Branch
Tier 1: Build Verification, Smoke Suite	Upon PR creation Note: Merge allowed only after successful pass.	feature-branch	Every day on build, and deployment to QA-UAT environment	qa-uat
Tier 2: Sanity Suite	Upon PR creation Note: Merge allowed only after successful pass.	feature-branch	Every alternate day, and deployment to qa-uat environment	qa-uat
Tier 3: Medium priority tests	N/A	N/A	Bi-Weekly: twice a week	qa-uat

Out of Scope

List of items that are outside the QA scope to be listed here.

Approvals

Version	**Reviewed By**	**Approved By**	**Date**
v1.0.0	Reviewer Name	Approver Name	07/25/2021