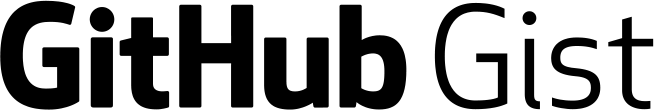
{Github’s Gists Api}



**Test Strategy Document** **for the API behaviour of GitHub Gists.**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version |  | Description |
| 3May 2022 | 0.1.0 | Pankaj Kumar | Test strategy document of github’s gists api |

# Scope

The test strategy is the high-level description of the test requirements from which a detailed test plan can later be derived, specifying individual test scenarios and test cases.

In scope of please consider following endpoints(authenticated & unauthenticated):

* [Creating a Gist](https://developer.github.com/v3/gists/#create-a-gist)
* [Reading a Gist](https://developer.github.com/v3/gists/#get-a-gist)
* [Updating a Gist](https://developer.github.com/v3/gists/#update-a-gist)
* [Deleting a Gist](https://developer.github.com/v3/gists/#delete-a-gist)
* Listing Gists for a user ([authenticated](https://developer.github.com/v3/gists/#list-gists-for-the-authenticated-user) & [unauthenticated](https://developer.github.com/v3/gists/#list-gists-for-a-user))

Verify the API types.

* SOAP API / REST API

# Test Approach

2.1 Types of Testing

### Unit Testing

The tests are written to run with each build of the application automatically. It is written close to the code and should pass while running the build of the application.

### Integration Testing

Integration testing ensures that the application takes the correct parameters with correct constraints while logically sanitizing the incoming traffic.

### Performance testing

Performance testing takes everything into account including the traffic spikes, multiple heavy processes running simultaneously, and interfering with one another.

### Runtime error detection

This testing allows the APIs to report any defects that occur while it is in process.

### Security testing

Security testing focuses on several aspects of the API testing strategy. The entry points of APIs need to be taken into account along with the data flow and shadow APIs.

### Interoperability testing

How API testing occurs when the APIs interact with third-party applications.

### Fuzz tests

The random data is sent to the API endpoints and the results are inspected carefully. The server should not crash from this unexpected traffic and should not display any unexpected behaviour.

### Validation Testing

The validation testing ensures that the application meets the business requirements. The test execution results need to match the expectations according to the test plan.

## Test scenarios for API Testing.

## Functional Testing

Our first concern is functional testing — ensuring that the API functions correctly.

## Error conditions

Execute API calls with invalid input, e.g.

Missing or invalid authorization token or Missing required parameters

## Boundary Value Analysis

Validation of keys with the minimum and maximum range of APIs. Test in all the boundaries

## Security Testing

It ensures API responds to correct authorization via all agreed auth methods – Bearer token, cookies, digest, etc. – as defined in spec

Negative: ensure API refuses all unauthorized calls

## Performance Testing

Verify the performance of api under different load and stress in order to measure its endurance.

## Usabilty Testing

To ensure the usability of the API for users without prior knowledge of our system.

* 1. **Variation in Apis endpoint with respect to authentication.**

A subset of our API is the /*users* endpoint, which includes the following API calls:

* Unauthenticated APIs and their endpoint

|  |  |
| --- | --- |
| API Call | Action |
| Read :GET /users/{username}/gists | Get list of public gists for specific user |
| Read :GET /gists | List all public gists |
| Read : GET /gists/{gists\_id} | Get gist by gist id |

* Authenticated APIs and their endpoint.

|  |  |
| --- | --- |
| API Call | Action |
| Read :GET /users/{username}/gists | Get list of public gists for specific user |
| Create :POST /gists | Create a new gist |
| Delete : DELETE /gists/{gists\_id} | Delete existing gist for gist id |
| Update : PATCH / gists/{gists\_id} | Update existing gist for gist id |

* 1. **Overview of main test scenarios for Manual test**

|  |  |  |  |
| --- | --- | --- | --- |
| # | **Test Scenario Category** | **Test Action Category** | **Test Action Description** |
| **1** | **Basic positive tests (happy paths)** |  |  |
|  | Execute Gists API call with **valid required parameters** | Validate status code: | 1. All requests should return 2XX HTTP status code 2. Returned status code is according to spec:  – 200 OK for GET requests – 201 for POST requests creating a new gists   – 200 OK for PATCH gists requests  – 200, 202, or 204 for a DELETE operation. |
|  |  | Validate payload: | 1. Response is a well-formed JSON object  2. Response structure is according to data model (schema validation) |
|  |  | Validate state: | 1. For GET requests, verify there is NO STATE CHANGE in the system. 2. For POST, DELETE, PATCH, operations – Ensure action has been performed correctly in the system by |
|  |  | Validate headers: | Verify that HTTP headers are as expected, including content-type, connection, cache-control, expires, access-control-allow-origin, keep-alive, and other standard header fields – according to spec. |
|  |  | Performance sanity: | Response is received in a timely manner. |
| **2** | **Positive + optional parameters** |  |  |
|  | Execute API call with **valid required parameters AND valid optional** parameters |  |  |
|  |  | Validate status code: | 1. All requests should return 2XX HTTP status code 2. Returned status code is according to spec:  – 200 OK for GET requests – 201 for POST requests creating a new gists   – 200 OK for PATCH gists requests  – 200, 202, or 204 for a DELETE operation. |
|  |  | Validate payload: | Response is a well-formed JSON object  2. Response structure is according to data model (schema validation). |
| **3** | **Negative testing – valid input** |  |  |
|  | Execute gist API calls with **valid input**that attempts illegal operations.  – Attempting to create a gist without authentication permission.  – Attempting to delete a gist that doesn’t exist.  – Attempting to update a gists with illegal valid data.  – Attempting to delete a gist without authentication permission. |  |  |
|  |  | Validate status code: | 1. Verify that an erroneous HTTP status code is sent (NOT 2XX)  2. Verify that the HTTP status code is in accordance with error case as defined in spec |
|  |  | Validate payload: | 1. Verify that error response is received  2. Verify that error format is according to spec. e.g., error is a valid JSON object or a plain string.  3. Verify that there is a clear, descriptive error message/description field.  4. Verify error description is correct for this error case and in accordance with spec. |
|  |
|  |  |  |  |
| **4** | **Negative testing – invalid input** |  |  |
|  | Execute API calls with **invalid** input.  – Missing or invalid authorization token.  – Missing required parameters.  – Invalid value for endpoint parameters, e.g.: – Invalid {user\_id} or {gist\_id} in path or query parameters. – Payload with invalid model (violates schema)  – Invalid values in HTTP headers |  |  |
|  |  | Validate status code: | 1. Verify that an erroneous HTTP status code is sent (NOT 2XX)  2. Verify that the HTTP status code is in accordance with error case as defined in spec |
|  |  | Validate payload: | 1. Verify that error response is received  2. Verify that error format is according to spec. e.g., error is a valid JSON object or a plain string.  3. Verify that there is a clear, descriptive error message/description field.  4. Verify error description is correct for this error case and in accordance with spec. |
| **5** | **Destructive testing** |  |  |
|  | Intentionally attempt to fail the API to check its robustness:  – Wrong content-type in payload.  – Small concurrency tests ,concurrent API calls that write to the same gists (DELETE + PATCH, etc.)   – Overflow payload  Empty payloads.  Illegal characters in parameters or payload  Boundary value testing  Other exploratory testing |  |  |
|  |  | Validate status code: | API should fail gracefully. |
|  |  | Validate payload: | API should fail gracefully. |
|  |  | Validate headers | API should fail gracefully. |

* 1. **Overview of main test scenarios for Automation Test**

|  |  |  |  |
| --- | --- | --- | --- |
| # | **User Journey** | **Test Action Category** | **Test Action Description** |
| **1** | Create new gist with user token. Save the gist\_id and later fetch created gist with get\_gist api. |  |  |
|  | Execute Gists API call with **valid required parameters** | Validate status code: | 1. All requests should return 2XX HTTP status code 2. Returned status code is according to spec:  – 200 OK for GET requests – 201 for POST requests creating a new gists |
|  |  | Validate payload: | 1. Response is a well-formed JSON object.  2. Schema validation |
|  |  | Validate headers: | 1 Verify that HTTP headers are as expected, including  *contenttype,*  *connection,*  *cache-control,*  *expires, access-control-allow-origin,*  *keep-alive*,  and other standard header fields – according to spec. |
| **2** | Delete existing gist with **delete\_api** and later try to fetch deleted gist with **get\_gist** api. | Validate status code: | 1. All requests should return respective HTTP status code 2. Returned status code is according to spec:  – 200 OK for DELETE requests – 404 for GET requests to fetch deleted gists |
| 3 | Create / Delete a gists with **invalid / no token**. | Validate status code: | 1. All requests should return respective HTTP status code 2. Returned status code is according to spec:  – 403 for CREATE requests – 403 for DELETE requests |
| 4 | Fetch the gists with invalid **{user\_id}** or **{gist\_id}** | Validate status code: | 1. All requests should return respective HTTP status code 2. Returned status code is according to spec:  – 404 for GET requests |
|  |
|  |  |  |  |

# Test Environment

The following table documents the testing environment criteria for each testing task:

| Testing Task | Environment | Database Name |
| --- | --- | --- |
| Unit Test | Dev Environment |  |
| Integration Test | Dev Environment |  |
| Functional Test | QA Environment |  |
| Automation Test | QA Environment |  |
| Performance Test | Pre-Prod Environment |  |
| Acceptance Test | Pre-Prod Environment |  |

# Testing Tools

The following testing tools will be used:

| Testing Tool | Purpose |
| --- | --- |
| Request module (Python) | To test and automate gists apis |
| Postman | To test apis manually |
| Jira | Test Management |

# Review and Approvals

This document has been approved as the for the Gists API.

Following approval of this document, changes will be governed by the project’s change management process, including impact analysis, appropriate reviews and approvals, under the general control of the Master Project Plan and according to Project Support Office policy.

|  |  |  |
| --- | --- | --- |
| Prepared by | Signature | Date |
| Pankaj Kumar  Sr QA  Thoughtworks |  |  |

|  |  |  |
| --- | --- | --- |
| Accepted by | Signature | Date |
| [Client Acceptor’s Name]  [Title]  [Organization] |  |  |

|  |  |  |
| --- | --- | --- |
| Approved by | Signature | Date |
| [Client Approver’s Name]  [Title]  [Organization] |  |  |
| [Client Approver’s Name]  [Title]  [Organization] |  |  |
| [Project Manager’s Name]  [Title]  [Organization] |  |  |
| [IMG Approver’s Name]  [Title]  [Organization] |  |  |