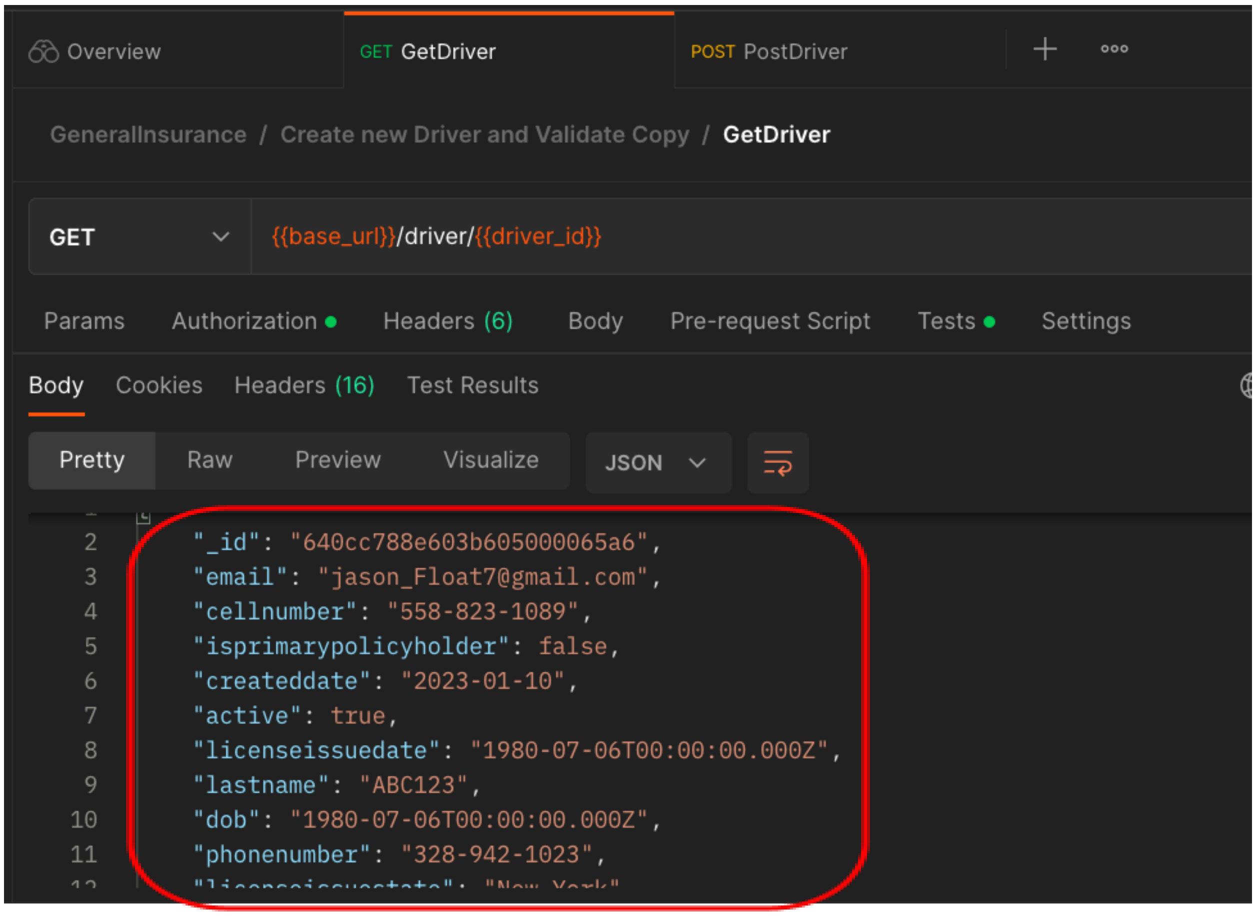
What is an API response?

In order to ensure that these APIs are functioning correctly, it's important to test them thoroughly. One important aspect of API testing is verifying the API response. In this article, we will explore what an API response is, and how it is tested.



```
{{base_url}}/driver/{{driver_id}}
```

```
    "_id": "640cc788e603b605000065a6",
    "email": "jason_Float7@gmail.com",
    "cellnumber": "558-823-1089",
    "isprimarypolicyholder": false,
    "createddate": "2023-01-10",
    "active": true,
    "licenseissuedate": "1980-07-06T00:00:00.000Z",
    "lastname": "ABC123",
    "dob": "1980-07-06T00:00:00.000Z",
    "phonenumber": "328-942-1023",
    "licenseissuestate": "New York"
```

An API response is the data that is returned by an API when a request is made to it. An API request typically consists of a request method, a URL, and a set of parameters or data that are passed in the request body. When the API receives the request, it processes the data and returns a response.

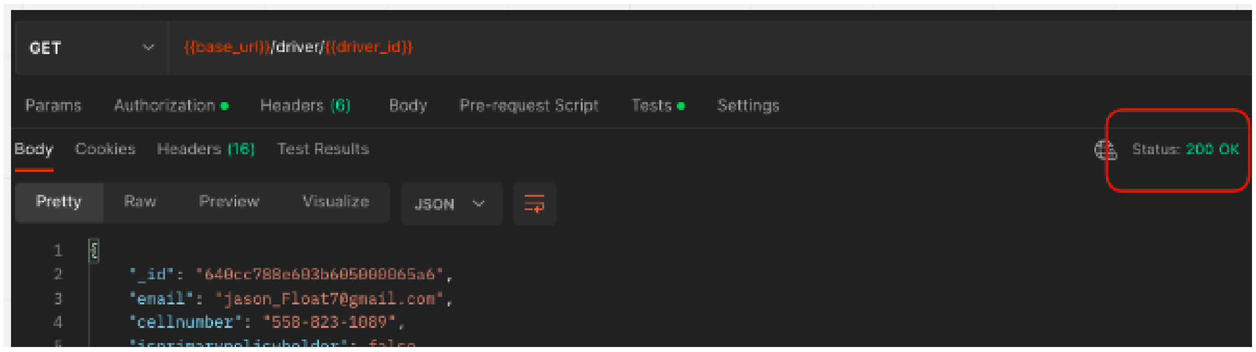
The API response typically contains the requested data in a structured format such as JSON or XML. In addition to the requested data, the API response may also include metadata such as response headers, status codes, and error messages.

The API response is a critical aspect of API testing. It is important to verify that the API response contains the expected data and that it is in the correct format. Additionally, it is important to verify that the response contains the appropriate metadata, such as response headers and status codes.

API Response Types

There are several types of API responses that an API can return. These include:

1. **Successful Response:** A successful response is returned when the API request is processed successfully. The response typically contains the requested data along with a status code of 200 OK.

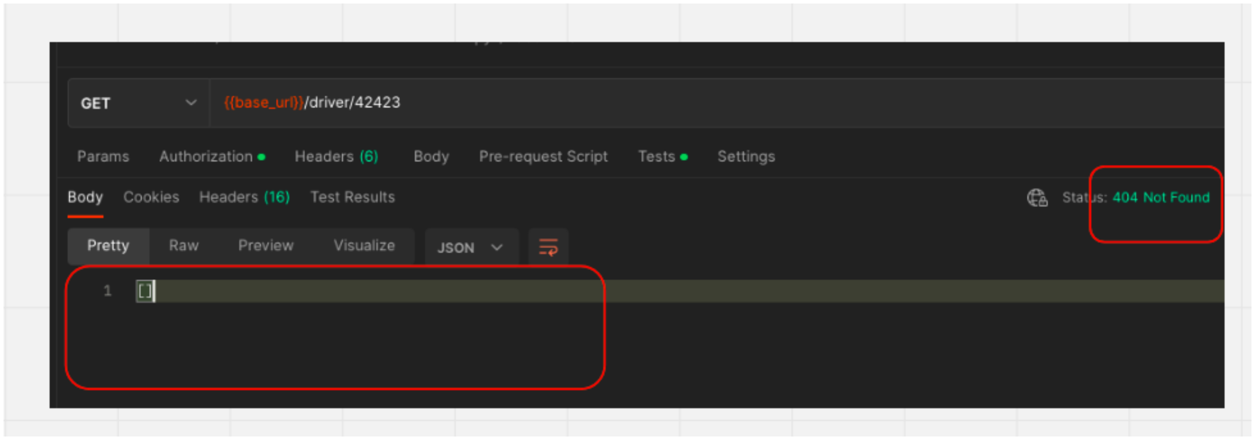


GET {{base_url}}/driver/{{driver_id}}

Status: 200 OK

```
{
    "_id": "640cc788e603b605000065a6",
    "email": "jason_Float7@gmail.com",
    "cellnumber": "558-823-1089",
```

2. **Error Response:** An error response is returned when the API request is unsuccessful. The response typically contains an error message along with a status code indicating the type of error that occurred. Some common status codes for error responses include 400 Bad Request, 401 Unauthorized, and 404 Not Found.



GET {{base_url}}/driver/42423

Status: 404 Not Found

```
[]
```

3. **Empty Response:** An empty response is returned when the API request is successful, but there is no data to return. The response typically contains a status code of 204 No Content.

4. **Redirect Response:** A redirect response is returned when the API request is successful, but the requested resource has been moved to a different URL. The response typically contains a status code of 301 Moved Permanently or 302 Found, along with the new URL to which the request should be redirected.

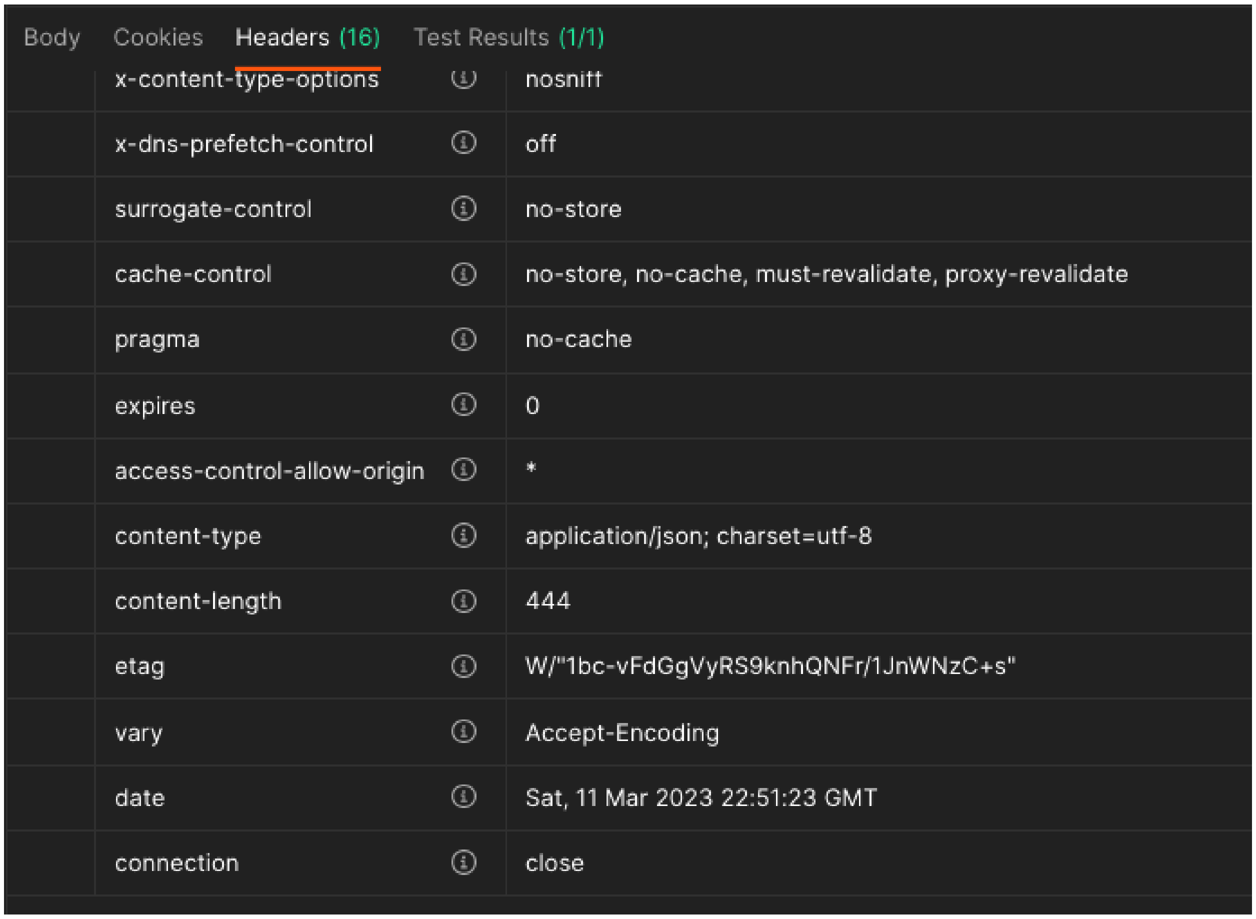
Testing API Responses

Testing API responses involves verifying that the API is returning the expected data in the correct format, along with the appropriate metadata such as response headers and status codes. There are several aspects to consider when testing API responses, including:

1. **Response Format:** The API response format should be consistent with the documentation or specification for the API. For example, if the API is designed to return JSON data, then the response should be in JSON format.
2. **Response Data:** The API response data should be accurate and complete. This means that the data returned by the API should match the data requested in the API request. Additionally, the data should be in the correct format, such as the correct date format or the correct currency format.
   See below a sample Driver Get response.

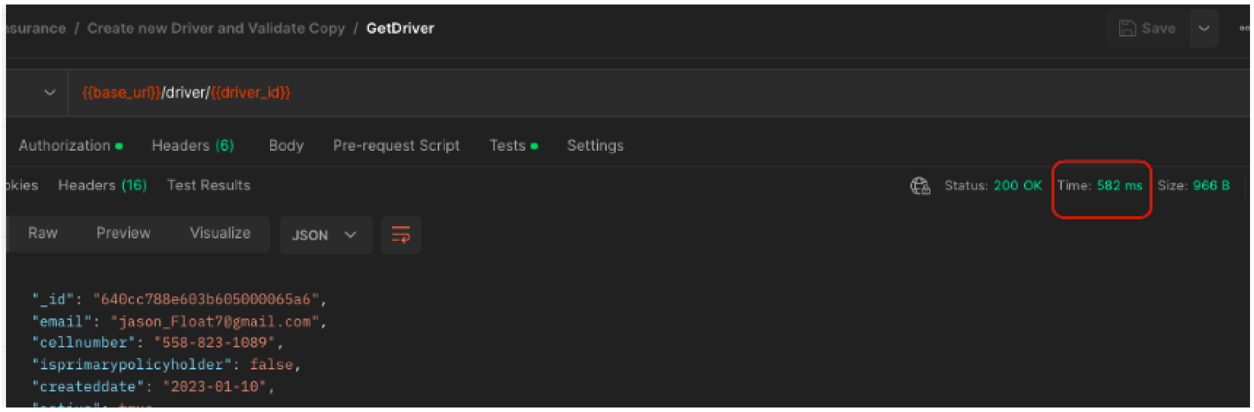
```
{
  "_id": "640cc788e603b605000065a6",
  "email": "jason_Float7@gmail.com",
  "cellnumber": "558-823-1089",
  "isprimarypolicyholder": false,
  "createddate": "2023-01-10",
  "active": true,
  "licenseissuedate": "1980-07-06T00:00:00.000Z",
  "lastname": "ABC123",
  "dob": "1980-07-06T00:00:00.000Z",
  "phonenumber": "328-942-1023",
  "licenseissuestate": "New York",
  "firstname": "Float123",
  "middleinitial": "Mr",
  "policy_id": 45645216,
  "ssn": "239-11-3311",
  "maritalstatus": "single",
  "ID": 525
}
```

3. **Response Metadata:** The API response metadata should be accurate and complete. This includes the response headers, status codes, and error messages. The status code should indicate whether the request was successful or not, and any error messages should be clear and descriptive.
   See below how Response header looks like



x-content-type-options	nosniff
x-dns-prefetch-control	off
surrogate-control	no-store
cache-control	no-store, no-cache, must-revalidate, proxy-revalidate
pragma	no-cache
expires	0
access-control-allow-origin	*
content-type	application/json; charset=utf-8
content-length	444
etag	W/"1bc-vFdGgVyRS9knhQNFr/1JnWNzC+s"
vary	Accept-Encoding
date	Sat, 11 Mar 2023 22:51:23 GMT
connection	close

4. **Response Time:** The API response time should be within acceptable limits. This means that the API should return the response within a reasonable amount of time, such as within a few seconds.



```
{{base_url}}/driver/{{driver_id}}
```

Status: 200 OK Time: 582 ms Size: 966 B

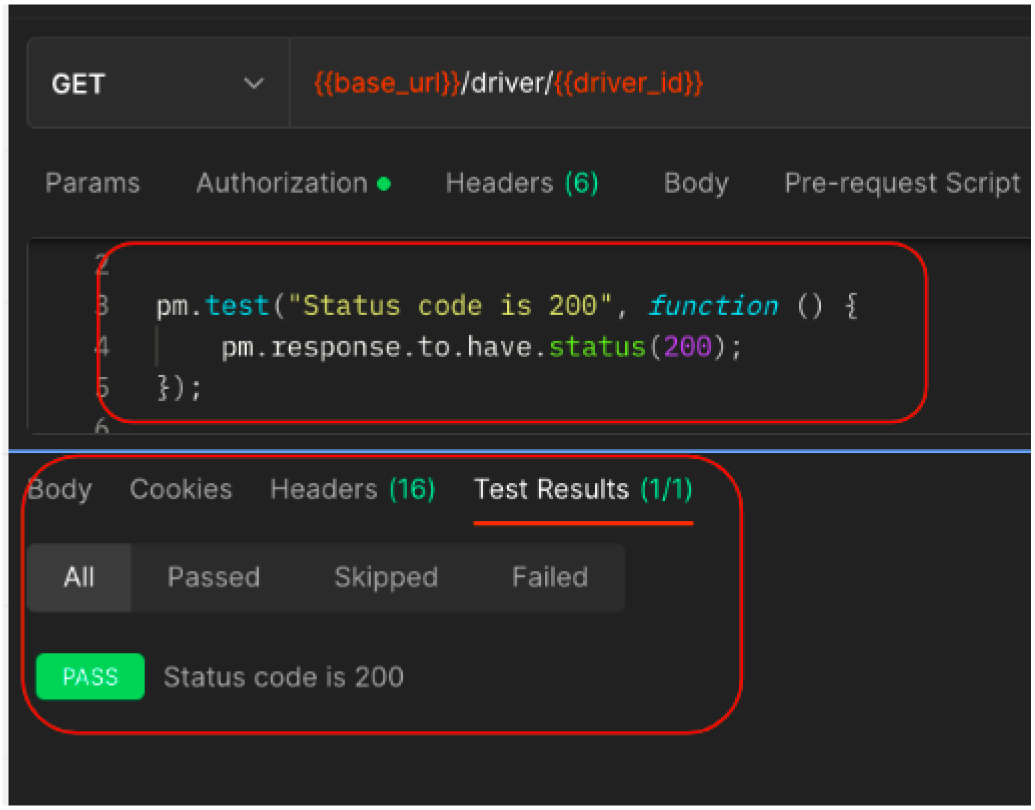
```
"_id": "640cc788e603b605000065a6",
"email": "jason_Float7@gmail.com",
"cellnumber": "558-823-1089",
"isprimarypolicyholder": false,
"createddate": "2023-01-10",
```

5. **Response Handling:** The API response should be handled correctly by the application that is consuming the API. This means that the application should be able to handle all possible response types, including successful responses, error responses, empty responses, and redirect responses.

API Response Testing Techniques

Several techniques for testing API responses. Here are some commonly used ones:

1. **Status Code Testing:** The HTTP status codes indicate the success or failure of an API request. The most common status codes are 200 for success, 400 for bad requests, and 500 for server error. Status code testing involves verifying that the API response returns the correct status code. Below test case validate if status code is 200.



```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

```
PASS Status code is 200
```

2. **Response Payload Testing:** This involves verifying that the API response contains the expected data, such as specific fields, values, or structures. You can use tools such as JSON schema to define the structure of the expected response and then use automated tests to check that the actual response matches the schema.

Let's say we have an API that allows users to create a new account. The API endpoint is POST /api/users and the payload should include the user's name, email, and password.
Here's an example of API payload testing for this scenario:
Positive test case: Send a valid payload with all the required fields (name, email, and password). Verify that the API returns a success response (e.g. 200 OK) and that a new user account is created with the provided information.
Negative test case: Send a payload without the required fields (e.g. only provide the name and email). Verify that the API returns an error response (e.g. 400 Bad Request) indicating that the password field is missing.
Negative test case: Send a payload with invalid data (e.g. an email address that is not in a valid format). Verify that the API returns an error response indicating the specific validation error (e.g. 422 Unprocessable Entity).
Negative test case: Send a payload with data that exceeds the allowed limits (e.g. a name that is too long or a password that is too short). Verify that the API returns an error response indicating the specific validation error (e.g. 422 Unprocessable Entity).
Negative test case: Send a payload with a duplicate email address. Verify that the API returns an error response indicating that a user with the same email address already exists (e.g. 409 Conflict).
Negative test case: Send a payload with an invalid authentication token. Verify that the API returns an error response indicating that the request is unauthorized (e.g. 401 Unauthorized).
These are just a few examples of the types of tests that can be performed on an API payload. The goal is to ensure that the API behaves correctly and returns appropriate responses for various input scenarios.

3. **Error Handling Testing:** APIs should return clear and informative error messages when there is an issue with the request. Error handling testing involves verifying that the API returns the correct error message when an invalid request is made.
4. **Performance Testing:** This involves testing the API's response time, throughput, and scalability under different load conditions. Tools such as Apache JMeter or LoadRunner can be used to simulate large numbers of API requests and measure the response time and other performance metrics.

   See below a test case in Postman that will capture each request response time. Although performance testing through Postman is not ideal, if you notice a normal request is taking several seconds such as 15 plus seconds. This should be reported.

```
pm.test("Response time is less than 200ms", () => {
 pm.expect(pm.response.responseTime).to.be.below(200);
});
```

5. **Security Testing:** APIs may be vulnerable to various security attacks, such as SQL injection, cross-site scripting, and brute force attacks. Security testing involves verifying that the API has implemented the necessary security measures to prevent these types of attacks.
6. **Integration Testing:** This involves testing the API's integration with other systems or services. For example, if an API is used to retrieve data from a database, integration testing would involve verifying that the API can connect to the database and retrieve the correct data.
7. **Parameter Testing:** APIs may have various parameters that can be used to modify the response. Parameter testing involves verifying that the API returns the expected response when different parameters are used. For example, if an API returns a list of products, parameter testing could involve verifying that the API returns the correct products when different sorting or filtering parameters are used.
   In the below examples we are searching people with the name joe by using query parameters.

   ```
   https://<url>/people?q={"name": "Joe"}
   ```

   Match multiple fields (find Joe who's 17):

   ```
   https://<url>/people?q={"name": "Joe", "age": 17}
   ```

8. **Versioning Testing:** APIs may have different versions that are used by different clients. Versioning testing involves verifying that each version of the API returns the expected response. This can be done by comparing the response of different versions of the API and verifying that they are consistent.
9. **Authentication Testing:** APIs may require authentication before access is granted. Authentication testing involves verifying that the API returns the expected response when different authentication credentials are used. This can be done by testing different authentication scenarios, such as valid credentials, invalid credentials, and expired credentials.
10. **Concurrency Testing:** APIs may need to handle multiple requests at the same time. Concurrency testing involves verifying that the API returns the expected response when multiple requests are made simultaneously. This can be done by using load testing tools to simulate multiple simultaneous requests and verifying that the API can handle them correctly.
11. **Caching Testing:** APIs may use caching to improve performance. Caching testing involves verifying that the API returns the expected response when cached data is used. This can be done by testing different caching scenarios, such as cache hits and cache misses, and verifying that the API returns the correct response in each case.

Overall, there are many different techniques that can be used to test API responses, and the specific techniques used will depend on the requirements of the API and the testing goals. It is important to have a comprehensive testing strategy that covers all the different aspects of API testing.

Response testing more examples:

API response testing involves verifying the response of an API (Application Programming Interface) when it is called with a particular set of input parameters. Here are some real-life examples of API response testing:

1. **Payment Gateway API:** Payment gateways are used by e-commerce websites to process transactions. API response testing for payment gateway APIs involves checking the response of the API for different payment types, payment amounts, and payment methods.
2. **Social Media APIs:** APIs of social media platforms such as Facebook, Twitter, and Instagram can be used to create and manage user accounts, posts, comments, and likes. API response testing for social media APIs involves testing for response time, error handling, and data validation.
3. **Weather API:** Weather APIs provide weather information for a specific location. API response testing for weather APIs involves testing the response time of the API, the accuracy of the data provided, and the API's ability to handle different weather conditions.
4. **Google Maps API:** Google Maps API is used to embed maps into web pages or mobile applications. API response testing for Google Maps API involves testing for different map features, response time, and error handling.
5. **Flight Booking API:** Flight booking APIs are used to search for and book flights. API response testing for flight booking APIs involves testing for response time, data accuracy, and error handling for different types of flights and airlines.
6. **Healthcare APIs:** Healthcare APIs provide medical information, insurance, and payment processing services. API response testing for healthcare APIs involves testing for data accuracy, response time, and error handling.

These are just a few examples of APIs that require response testing. In general, any API that returns data or performs an action can benefit from response testing.