

Use endpoint from <https://jsonplaceholder.typicode.com/> for API test cases:

**Q1. Create all possible positive negative test cases for following endpoints**

**GET /posts**

```
pm.test("Response of Json with body", function(){  
    pm.response.to.be.ok;  
    pm.response.to.be.withBody;  
    pm.response.to.be.json;  
});  
  
pm.test("response should be error free", function () {  
    pm.response.to.not.be.error;  
    pm.response.to.not.have.jsonBody("error");  
});  
  
pm.test("Status code is 200", function () {  
    pm.response.to.have.status(200);  
});  
  
pm.test("Response time is less than 200ms", function () {  
    pm.expect(pm.response.responseTime).to.be.below(200);  
});  
  
pm.test("Content-Type is present", function () {  
    pm.response.to.have.header("Content-Type");  
});  
  
pm.test("All keys included", function () {  
    pm.expect(pm.response.text()).to.include("userID" & "id" & "title" & "body");
```

```
});

pm.test("100 IDs", function () {

    pm.expect(pm.response.json()[0].id).eq(1);

    pm.expect(pm.response.json()[99].id).eq(100);

});

pm.test("Id to be ordered", function () {

    pm.expect(pm.response.json().id).to.be.ordered;

});
```

The screenshot displays the Postman interface for a GET request to `https://jsonplaceholder.typicode.com/posts`. The 'Tests' tab is active, showing a test script with five lines of code. Below the script, the 'Test Results (8/8)' tab is selected, showing a list of eight passed tests. Each test is preceded by a green 'PASS' button.

Test Name	Status
Response of Json with body	PASS
response should be error free	PASS
Status code is 200	PASS
Response time is less than 200ms	PASS
Content-Type is present	PASS
All keys included	PASS
100 IDs	PASS
Id to be ordered	PASS

## **GET /posts/1**

```
pm.test("Response of Json with body", function(){
    pm.response.to.be.ok;
    pm.response.to.be.withBody;
    pm.response.to.be.json;
});

pm.test("response should be error free", function () {
    pm.response.to.not.be.error;
    pm.response.to.not.have.jsonBody("error");
});

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

pm.test("Response time is less than 100ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(100);
});

pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Content-Type");
});

pm.test("Only 1 ID", function () {
    pm.expect(pm.response.json().id).not.greaterThan(1);
});

pm.test("Verify ID", function () {
```

```
    pm.expect(pm.response.json().id).eq(1);
  });

  pm.test("Verify userID", function () {

    pm.expect(pm.response.json().userId).eq(1);

  });

  pm.test("Verify title", function () {

    pm.expect(pm.response.json().title).eq("sunt aut facere repellat provident occaecati excepturi optio reprehenderit");

  });

  pm.test("Verify body", function () {

    pm.expect(pm.response.json().body).eq("quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto");

  });
```

API testing 2 / **/posts/1** test cases

GET ▼ <https://jsonplaceholder.typicode.com/posts/1>

Params Authorization Headers (6) Body Pre-request Script **Tests ●** Settings

```
18 });
19 pm.test("Only 1 ID", function () {
20     pm.expect(pm.response.json().id).not.greaterThan(1);
21 });
22 pm.test("Verify ID", function () {
23     pm.expect(pm.response.json().id).eq(1);
24 });
25 pm.test("Verify userID", function () {
```

Body Cookies Headers (25) **Test Results (10/10)**

All Passed Skipped Failed

- PASS** Response of Json with body
- PASS** response should be error free
- PASS** Status code is 200
- PASS** Response time is less than 100ms
- PASS** Content-Type is present
- PASS** Only 1 ID
- PASS** Verify ID
- PASS** Verify userID
- PASS** Verify title
- PASS** Verify body

### GET **/posts/1/comments**

```
pm.test("Response of Json with body", function(){
    pm.response.to.be.ok;
    pm.response.to.be.withBody;
    pm.response.to.be.json;
});

pm.test("response should be error free", function () {
    pm.response.to.not.be.error;
```

```
    pm.response.to.not.have.jsonBody("error");
  });

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

  pm.test("Response time is less than 100ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(100);
  });

  pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Content-Type");
  });

  pm.test("5 IDs", function () {
    pm.expect(pm.response.json()[0].id).eqls(1);
    pm.expect(pm.response.json()[4].id).eqls(5);
  });

  pm.test("Verify postID", function () {
    pm.expect(pm.response.json()[0,4].postId).eqls(1);
  });

  pm.test("Id to be ordered", function () {
    pm.expect(pm.response.json().id).to.be.ordered;
  });
```

GET <https://jsonplaceholder.typicode.com/posts/1/comments>Params Authorization Headers (6) Body Pre-request Script **Tests ●** Settings

```
1 pm.test("Response of Json with body", function(){
2     pm.response.to.be.ok;
3     pm.response.to.be.withBody;
4     pm.response.to.be.json;
5 });
6 pm.test("response should be error free", function () {
7     pm.response.to.not.be.error;
8     pm.response.to.not.have.jsonBody("error");
9 });
10 pm.test("Status code is 200", function () {
```

Body Cookies Headers (25) **Test Results (8/8)**

All Passed Skipped Failed

**PASS** Response of Json with body**PASS** response should be error free**PASS** Status code is 200**PASS** Response time is less than 100ms**PASS** Content-Type is present**PASS** 5 IDs**PASS** Verify postID**PASS** Id to be ordered**GET /comments?postId=1 //----->(Same as above)**

```
pm.test("Response of Json with body", function(){
```

```
    pm.response.to.be.ok;

    pm.response.to.be.withBody;

    pm.response.to.be.json;
  });

  pm.test("response should be error free", function () {

    pm.response.to.not.be.error;

    pm.response.to.not.have.jsonBody("error");
  });

  pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);
  });

  pm.test("Response time is less than 100ms", function () {

    pm.expect(pm.response.responseTime).to.be.below(100);
  });

  pm.test("Content-Type is present", function () {

    pm.response.to.have.header("Content-Type");
  });

  pm.test("5 IDs", function () {

    pm.expect(pm.response.json()[0].id).eq(1);

    pm.expect(pm.response.json()[4].id).eq(5);
  });

  pm.test("Verify postID", function () {

    pm.expect(pm.response.json()[0,4].postId).eq(1);
  });
});
```



```
pm.test("Id to be ordered", function () {  
  
    pm.expect(pm.response.json().id).to.be.ordered;});
```

The screenshot displays the Postman interface for a GET request to `https://jsonplaceholder.typicode.com/comments?postId=1`. The 'Tests' tab is active, showing the following test script:

```
18 }  
19 pm.test("5 IDs", function () {  
20     pm.expect(pm.response.json()[0].id).eq(1);  
21     pm.expect(pm.response.json()[4].id).eq(5);  
22 }  
23 pm.test("Verify postID", function () {  
24     pm.expect(pm.response.json()[0,4].postId).eq(1);  
25 }  
26 pm.test("Id to be ordered", function () {  
27     pm.expect(pm.response.json().id).to.be.ordered;  
28 }  
29
```

Below the script, the 'Test Results (8/8)' tab is selected, showing a list of all passed tests:

All	Passed	Skipped	Failed
PASS	Response of Json with body		
PASS	response should be error free		
PASS	Status code is 200		
PASS	Response time is less than 100ms		
PASS	Content-Type is present		
PASS	5 IDs		
PASS	Verify postID		
PASS	Id to be ordered		

**POST /posts**



```
pm.test("response should be error free", function () {  
    pm.response.to.not.be.error;  
});  
  
pm.test("POST request", function () {  
    pm.expect(pm.response.code).to.be.oneOf([201, 202]);  
});  
  
pm.test("Response time is less than 200ms", function () {  
    pm.expect(pm.response.responseTime).to.be.below(100);  
});  
  
pm.test("Content-Type is present", function () {  
    pm.response.to.have.header("Content-Type");  
});  
  
pm.test("Post status", () => {  
    pm.response.to.have.status("Created");  
});  
  
pm.test("Verify data types", () => {  
    pm.expect(pm.response.json().userId).to.be.a("number");  
    pm.expect(pm.response.json().id).to.be.a("number");  
});
```

```
pm.expect(pm.response.json().title).to.be.a("string");

pm.expect(pm.response.json().body).to.be.a("string");

});

pm.test("Non-empty fields", function () {

    pm.expect(pm.response.json().userID).not.null;

    pm.expect(pm.response.json().id).not.null;

    pm.expect(pm.response.json().title).not.null;

    pm.expect(pm.response.json().body).not.null;

});
```

The screenshot displays the Postman interface for a POST request. The top bar shows the method 'POST' and the URL 'https://jsonplaceholder.typicode.com/posts'. Below this, the 'Tests' tab is selected, showing a list of 10 test cases. The bottom section, 'Test Results (7/7)', shows that all 7 tests passed, each with a green 'PASS' label.

Method	URL
POST	https://jsonplaceholder.typicode.com/posts

Params   Authorization   Headers (8)   Body ●   Pre-request Script   **Tests ●**   Settings

```
1 pm.test("response should be error free", function () {
2   pm.response.to.not.be.error;
3 });
4 pm.test("POST request", function () {
5   pm.expect(pm.response.code).to.be.oneOf([201, 202]);
6 });
7 pm.test("Response time is less than 200ms", function () {
8   pm.expect(pm.response.responseTime).to.be.below(100);
9 });
10 pm.test("Content-Type is present", function () {
```

Body   Cookies   Headers (25)   **Test Results (7/7)**

All	Passed	Skipped	Failed
PASS	response should be error free		
PASS	POST request		
PASS	Response time is less than 200ms		
PASS	Content-Type is present		
PASS	Post status		
PASS	Verify data types		
PASS	Non-empty fields		

## PUT /posts/1

PUT

https://jsonplaceholder.typicode.com/posts/1

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1  [
2    ... "userId": 1,
3    ... "id": 1,
4    ... "title": "Hi",
5    ... "body": "Hello world"
6  ]
```

Body

Cookies

Headers (24)

Test Results (9/9)

Pretty

Raw

Preview

Visualize

JSON

```
1  [
2    "userId": 1,
3    "id": 1,
4    "title": "Hi",
5    "body": "Hello world"
6  ]
```

```
pm.test("Response of Json with body", function(){
```

```
    pm.response.to.be.ok;

    pm.response.to.be.withBody;

    pm.response.to.be.json;
  });

  pm.test("response should be error free", function () {

    pm.response.to.not.be.error;

    pm.response.to.not.have.jsonBody("error");
  });

  pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);
  });

  pm.test("Response time is less than 100ms", function () {

    pm.expect(pm.response.responseTime).to.be.below(100);
  });

  pm.test("Content-Type is present", function () {

    pm.response.to.have.header("Content-Type");
  });

  pm.test("Verify userId", function () {

    pm.expect(pm.response.json().userId).eqls(1);
  });

  pm.test("Verify ID", function () {

    pm.expect(pm.response.json().id).eqls(1);
  });

  pm.test("Verify New title", function () {
```

```
pm.expect(pm.response.json().title).eqls("Hi");

});

pm.test("Verify New Body", function () {

    pm.expect(pm.response.json().body).eqls("Hello world");

});
```

PUT

▼

https://jsonplaceholder.typicode.com/posts/1

ParamsAuthorizationHeaders (8)Body ●Pre-request ScriptTests ●Settings

1

pm.test("Response of Json with body", function(){

2

pm.response.to.be.ok;

3

pm.response.to.be.withBody;

4

pm.response.to.be.json;

5

});

6

pm.test("response should be error free", function () {

BodyCookiesHeaders (24)Test Results (9/9)

AllPassedSkippedFailed

PASS

Response of Json with body

PASS

response should be error free

PASS

Status code is 200

PASS

Response time is less than 100ms

PASS

Content-Type is present

PASS

Verify userId

PASS

Verify ID

PASS

Verify New title

PASS

Verify New Body

## PATCH /posts/1

The screenshot shows a REST client interface with the following details:

- Method:** PATCH
- URL:** https://jsonplaceholder.typicode.com/posts/5
- Body Tab:** Selected, showing a JSON body:

```
1 {  
2   "title": "Hey"  
3 }
```
- Format:** JSON
- Response Tab:** Selected, showing a JSON response:

```
1 {  
2   "userId": 1,  
3   "id": 5,  
4   "title": "Hey",  
5   "body": "Hello world"  
6 }
```

```
pm.test("Response of Json with body", function(){
```

```
  pm.response.to.be.ok;
```

```
  pm.response.to.be.withBody;
```

```
  pm.response.to.be.json;
```

```
});  
pm.test("response should be error free", function () {  
    pm.response.to.not.be.error;  
    pm.response.to.not.have.jsonBody("error");  
});  
pm.test("Status code is 200", function () {  
    pm.response.to.have.status(200);  
});  
pm.test("Response time is less than 100ms", function () {  
    pm.expect(pm.response.responseTime).to.be.below(100);  
});  
pm.test("Content-Type is present", function () {  
    pm.response.to.have.header("Content-Type");  
});  
pm.test("Verify userId", function () {  
    pm.expect(pm.response.json().userId).eq(1);  
});  
pm.test("Verify ID", function () {  
    pm.expect(pm.response.json().id).eq(5);  
});  
pm.test("Verify New title", function () {  
    pm.expect(pm.response.json().title).eq("Hey");  
});  
pm.test("Verify Body", function () {  
    pm.expect(pm.response.json().body).eq("Hello world");  
});
```



PATCH

▼

https://jsonplaceholder.typicode.com/posts/5

ParamsAuthorizationHeaders (8)Body●Pre-request ScriptTests●Settings

```
1 pm.test("Response of Json with body", function(){
2     pm.response.to.be.ok;
3     pm.response.to.be.withBody;
4     pm.response.to.be.json;
5 });
6 pm.test("response should be error free", function () {
7     pm.response.to.not.be.error;
8     pm.response.to.not.have.jsonBody("error");
9 });
```

BodyCookiesHeaders (24)Test Results (9/9)

AllPassedSkippedFailed

PASS

Response of Json with body

PASS

response should be error free

PASS

Status code is 200

PASS

Response time is less than 100ms

PASS

Content-Type is present

PASS

Verify userId

PASS

Verify ID

PASS

Verify New title

PASS

Verify Body

**DELETE** /posts/1

```
pm.test("Correct Response", function(){

    pm.response.to.be.ok;

    pm.response.to.be.json;

});

pm.test("response should be error free", function () {
```

```
pm.response.to.not.be.error;

pm.response.to.not.have.jsonBody("error");

});

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

pm.test("Response time is less than 200ms", function () {

    pm.expect(pm.response.responseTime).to.be.below(200);

});

pm.test("Content-Type is present", function () {

    pm.response.to.have.header("Content-Type");

});

pm.test("Verify deletion", function () {

    pm.expect(pm.response.json().id).not.exist; });
```

DELETE

https://jsonplaceholder.typicode.com/posts/1

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTests●Settings

```
1 pm.test("Correct Response", function(){
2   pm.response.to.be.ok;
3   pm.response.to.be.json;
4 });
5 pm.test("response should be error free", function () {
6   pm.response.to.not.be.error;
7   pm.response.to.not.have.deepBody(["error"]);
```

BodyCookiesHeaders (23)Test Results (6/6)

AllPassedSkippedFailed

PASS

Correct Response

PASS

response should be error free

PASS

Status code is 200

PASS

Response time is less than 200ms

PASS

Content-Type is present

PASS

Verify deletion

## Q2. Assert response code and value from the response body.

GET

https://jsonplaceholder.typicode.com/posts

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTests●Settings

```
1 pm.test("Response code", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Response code", function () {
6   pm.response.to.have.status("OK");
7 });
```

BodyCookiesHeaders (25)Test Results (2/2)

AllPassedSkippedFailed

PASS


Response code


PASS

Response code


🌐

Status: 200 OK 1

GET  https://jsonplaceholder.typicode.com/comments

Params Authorization Headers (6) Body Pre-request Script **Tests ** Settings

```
1 pm.test("Response code", function () {
2   | pm.response.to.have.status(200);
3 });
4
5 pm.test("Response code", function () {
6   | pm.response.to.have.status("OK");
7 });
```

Body Cookies Headers (25) **Test Results (2/2)**  Status: 200 OK

All Passed Skipped Failed

**PASS** Response code

**PASS** Response code

GET  https://jsonplaceholder.typicode.com/albums

Params Authorization Headers (6) Body Pre-request Script **Tests ** Settings

```
1 pm.test("Response code", function () {
2   | pm.response.to.have.status(200);
3 });
4
5 pm.test("Response code", function () {
6   | pm.response.to.have.status("OK");
7 });
```

Body Cookies Headers (25) **Test Results (2/2)**


All Passed Skipped Failed

**PASS** Response code

**PASS** Response code


**Q3. Verify for id 100 exists in response data**

<https://jsonplaceholder.typicode.com/posts>

GET  https://jsonplaceholder.typicode.com/posts

Params Authorization Headers (6) Body Pre-request Script **Tests ●** Settings

```
1 pm.test("Id 100 exists", function () {
2   pm.expect(pm.response.json()[99].id).toEqual(100);
3 });
```

Body Cookies Headers (25) **Test Results (1/1)**  Status: 200 OK

All Passed Skipped Failed


**PASS** Id 100 exists

OR

GET  https://jsonplaceholder.typicode.com/posts

Params Authorization Headers (6) Body Pre-request Script **Tests ●** Settings

```
1 pm.test("Id 100 exists", function () {
2   pm.expect(pm.response.text()).toContain(100);
3 });
```


Body Cookies Headers (25) **Test Results (1/1)**  Status: 200 OK T


All Passed Skipped Failed

**PASS** Id 100 exists


**Q4. Verify email id for from given response whose id is 5**

<https://jsonplaceholder.typicode.com/comments>

GET  https://jsonplaceholder.typicode.com/comments

Params Authorization Headers (6) Body Pre-request Script **Tests ** Settings

```
1 pm.test("Email verified for ID 5", function () {
2   |   pm.expect(pm.response.json()[4].email).to.eql("Hayden@althea.biz");
3   | });
4
```

Body Cookies Headers (25) **Test Results (1/1)**  Status: 200 OK

All Passed Skipped Failed

**PASS** Email verified for ID 5

## Q5. Verify title from a given endpoint for tenth response

<https://jsonplaceholder.typicode.com/albums>

GET  https://jsonplaceholder.typicode.com/albums

Params Authorization Headers (6) Body Pre-request Script **Tests ** Settings

```
1 pm.test("Title for 10th response ", function () {
2   |   pm.expect(pm.response.json()[9].title).to.eql("distinctio laborum qui");
3   | });
4
```

Body Cookies Headers (25) **Test Results (1/1)**  Status: 200 OK

All Passed Skipped Failed

**PASS** Title for 10th response