

API SECURITY RISK ANALYSIS REPORT :

Assessment of Public Demo API – JSON Placeholder

Intern Name : Thejas R Shetty

CIN ID : FIT/FEB26/CS6248

Organization : Future Interns

Task 3 -> API Security Risk Analysis

Repository Name : FUTURE_CS_03

Submission Date : 23/02/2026

1. Overview :

This report presents a security assessment of the public demo API JSONPlaceholder.

The objective of this analysis was to identify :

- ❖ Authentication weaknesses
- ❖ Authorization gaps
- ❖ Excessive data exposure
- ❖ Missing rate limiting
- ❖ Error handling issues

All testing was conducted using read-only HTTP GET requests.

No exploitation or attack attempts were performed.

2. Tools Used :

- Postman Tool
- Browser DevTools

3. Scope of Assessment : The assessment focused on ,

- ✓ Authentication requirements
- ✓ Authorization controls
- ✓ Rate limiting
- ✓ HTTP status responses
- ✓ Error handling

Testing was limited to publicly available endpoints

3. API Tested :

API Name : JSON Placeholder

Base URL Tested :

<https://jsonplaceholder.typicode.com>

4. Endpoints Tested :

- ❖ GET /users
- ❖ GET /users/1
- ❖ GET /posts
- ❖ GET /comments
- ❖ GET /admin (Invalid endpoint test)

5. Testing Methodology :

The API was analyzed using :

- ❖ Postman for sending HTTP requests
- ❖ Browser Developer Tools (Network tab) for inspecting headers and responses

The following process was followed :

1. Sent HTTP GET requests using Postman
2. Inspected Authorization tab
3. Reviewed response headers
4. Tested direct object access
5. Verified error handling using invalid endpoint
6. Observed network activity via Browser Developer Tools

3. Endpoint Testing Evidence :

◆ 4.1 Users Endpoint Analysis :

Endpoint Tested :

GET /users

Observation :

1. Accessible without authentication
2. Returns full user details

Screenshots :

The screenshot shows the Postman application interface. On the left, the sidebar displays 'THEJAS R SHETTY's Workspace' with 'My Collection' selected. Under 'My Collection', there is a 'Get data' item. The main workspace shows a 'GET https://jsonplaceholder.typicode.com/users' request. The 'Params' tab is selected, showing an empty table. Below the request, the 'Body' tab shows a JSON response with one user object. The response body is:

```
1 [  
2   {  
3     "id": 1,  
4     "name": "Leanne Graham",  
5     "username": "Bret",  
6     "email": "Sincere@april.biz",  
7     "address": {  
8       "street": "Kulas Light",  
9       "suite": "Apt. 556",  
10      "city": "Gwenborough",  
11      "zipcode": "92998-3874",  
12      "geo": {  
13        "lat": "-37.3159",  
14        "lng": "81.1496"  
15      }  
16    },  
17    "phone": "1-770-736-8031 x56442",  
18  }]  
19 ]
```

The status bar at the bottom indicates a '200 OK' response with 101 ms and 2.94 KB. The bottom right corner shows the date and time as 4:47 PM on 2/22/2026.

Fig 1 :Postman Full Request –GET /users showing 200 OK

The screenshot shows the Postman interface with a dark theme. On the left, there's a sidebar with 'Collections' (My Collection), 'Environments' (History, Flows), and 'Files' (BETA). The main area has a 'GET Get data' request to 'https://jsonplaceholder.typicode.com/users'. Under the 'Authorization' tab, 'No Auth' is selected. The response section shows a 200 OK status with a timestamp of 'Sun, 22 Feb 2026 11:17:27 GMT'. The bottom status bar shows 'High UV Now' and the date '2/23/2026'.

Fig 2 :Authorization Tab – Showing “No Auth”

This screenshot shows the same Postman interface as Fig 2, but with the 'Headers' tab selected. It lists numerous response headers with their values. Some of the headers include 'Date' (Sun, 22 Feb 2026 11:17:27 GMT), 'Content-Type' (application/json; charset=utf-8), 'Content-Length' (1847), 'Connection' (keep-alive), 'access-control-allow-credentials' (true), 'Cache-Control' (max-age=43200), 'Content-Encoding' (gzip), 'etag' (W/"80d-1eMsxsJfnnLRBmYJStCLZ1qQ"), 'expires' (-1), 'nei' ({"report_to": "heroku-nei", "response_headers": {"Via": "max_age": 3600, "success_fraction": 0.01, "failure_fraction": 0.1}), 'pragma' (no-cache), 'report-to' ({"group": "heroku-nei", "endpoints": [{"url": "https://nei.herokuapp.com/reports?": "%2BodXF0t4kr98BvDBhtMcLDasj0vgWCvub..."}]), 'reporting-endpoints' (heroku-nei="https://nei.herokuapp.com/reports?%2BodXF0t4kr98BvDBhtMcLDasj0vgWCvub...&id=e11707...")), 'Server' (cloudflare), and 'vary' (Origin, Accept-Encoding).

Fig 3: Headers – Showing Response Headers

◆ 4.2 Single User Access Test :

Endpoint Tested :

/users/1

Observation :

1. Direct object access allowed
2. No identity verification required

Screenshots :

The screenshot shows the Postman application interface. On the left, the sidebar displays 'THEJAS R SHETTY's Workspace' with collections like 'My Collection' and environments. The main workspace shows a 'Get data' request for 'https://jsonplaceholder.typicode.com/users/1'. The 'Headers' tab is selected, showing an empty table. The 'Body' tab shows a JSON response with 23 lines of data, including fields like id, name, address, and company. The status bar at the bottom indicates a 200 OK response with 128 ms and 1.44 KB.

```
1 {
2   "id": 1,
3   "name": "Leanne Graham",
4   "username": "Bret",
5   "email": "Sincere@april.biz",
6   "address": {
7     "street": "Kulas Light",
8     "suite": "Apt. 556",
9     "city": "Gwenborough",
10    "zipcode": "92998-3874",
11    "geo": [
12      {
13        "lat": "-37.3159",
14        "lng": "81.1496"
15      }
16    ],
17    "phone": "1-770-736-8031 x56442",
18    "website": "hildegard.org",
19    "company": {
20      "name": "Romaguera-Crona",
21      "catchPhrase": "Multi-layered client-server neural-net",
22      "bs": "harness real-time e-markets"
23    }
}
```

Fig 1 :Single User Endpoint – /users/1)

◆ 4.3 Posts Endpoint Analysis :

Endpoint Tested :

GET /posts

Observation :

1. All posts publicly accessible
2. No authentication required

Screenshots :

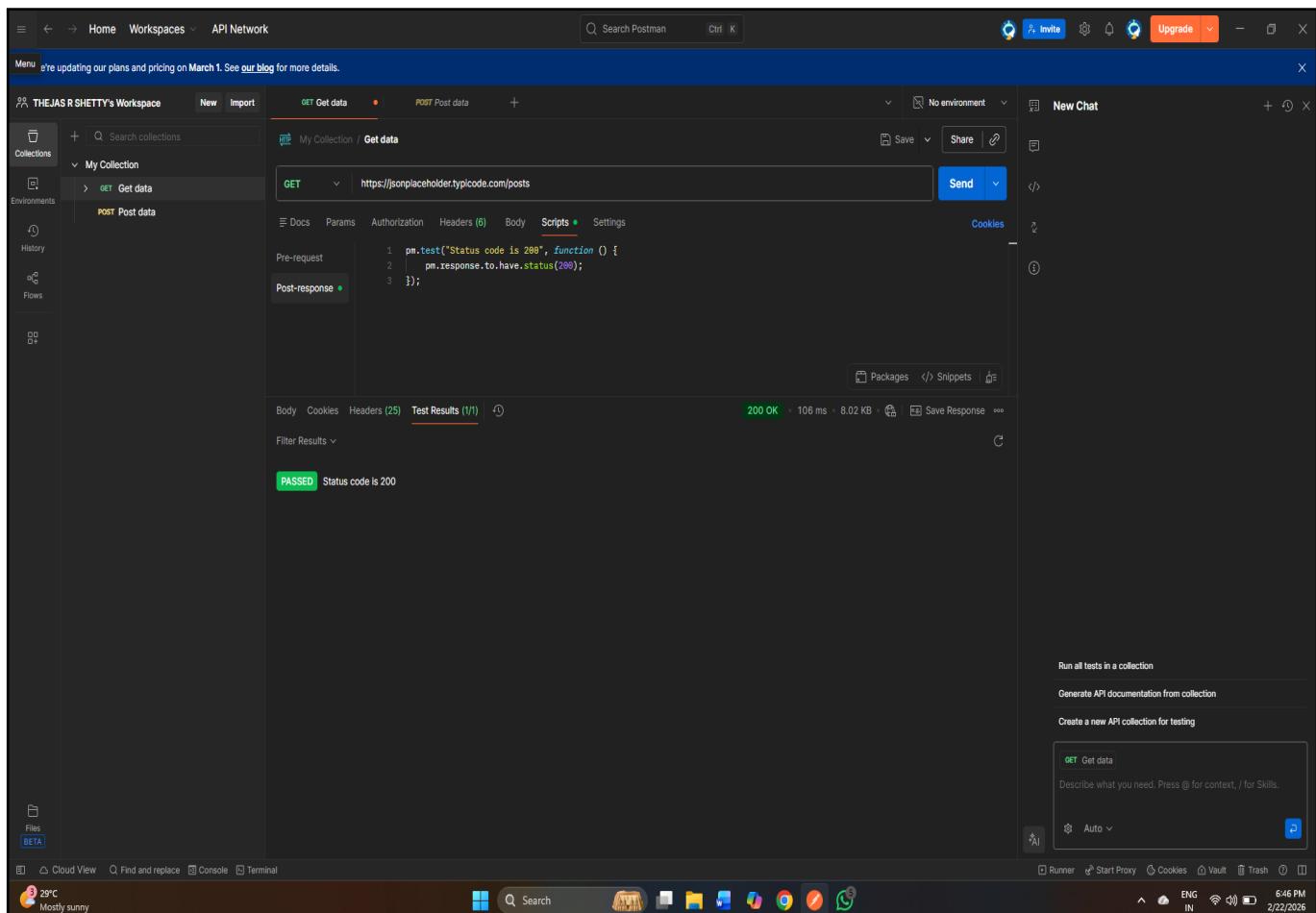


Fig 1: Posts Endpoint Screenshot – GET /posts

◆ 4.4 Comments Endpoint Analysis :

Endpoint Tested :

GET /comments

Observation :

- 1.Returns email addresses and comment data
 - 2.No access restriction applied

Screenshots :

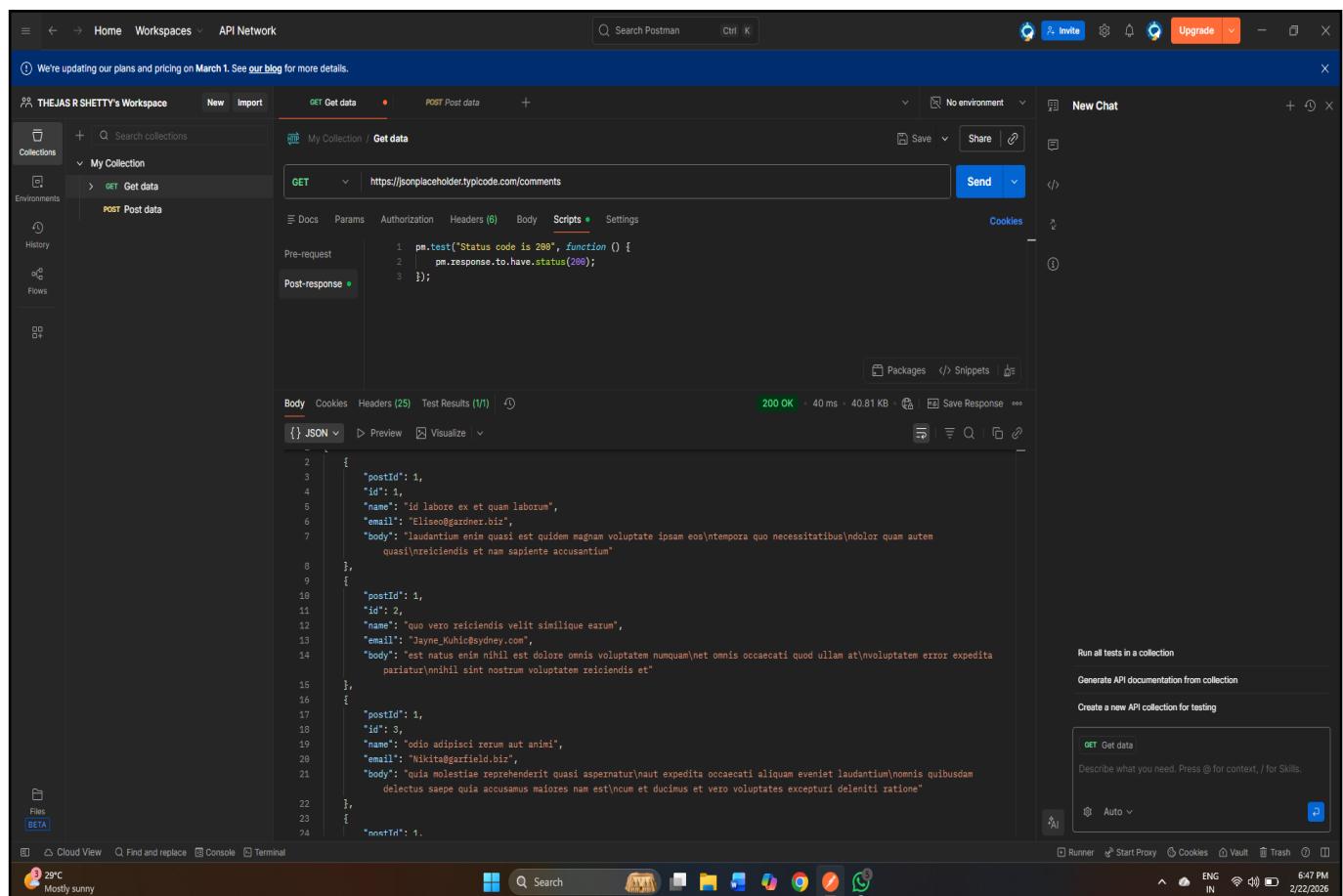


Fig 1: Comments Endpoint – GET /comments

◆ 4.5 Invalid Endpoint Test (Error Handling Check):

Endpoint Tested :

GET /admin

Observation :

- 1.Returns 404 Not Found
- 2.Proper HTTP status handling observed

Screenshot :

The screenshot shows the Postman application interface. In the center, there is a collection named "My Collection" with a single GET request to "https://jsonplaceholder.typicode.com/comments". The "Scripts" tab contains a pre-request script:

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

The "Test Results" section shows a 200 OK response with a duration of 40 ms and a size of 40.81 KB. The "Body" section displays a JSON array of comments, which is expected for a valid endpoint like /comments. The status bar at the bottom indicates "29°C Mostly sunny" and the system clock "6:47 PM 2/22/2026".

Fig 1 : 404 Not Found Screenshot – GET /admin

◆ 4.6 Browser Network Inspection :

Network analysis performed using Chrome Developer Tools.

Observation :

- 1.Response headers visible
- 2.Proper HTTP status codes

No authentication tokens detected

Screenshot :

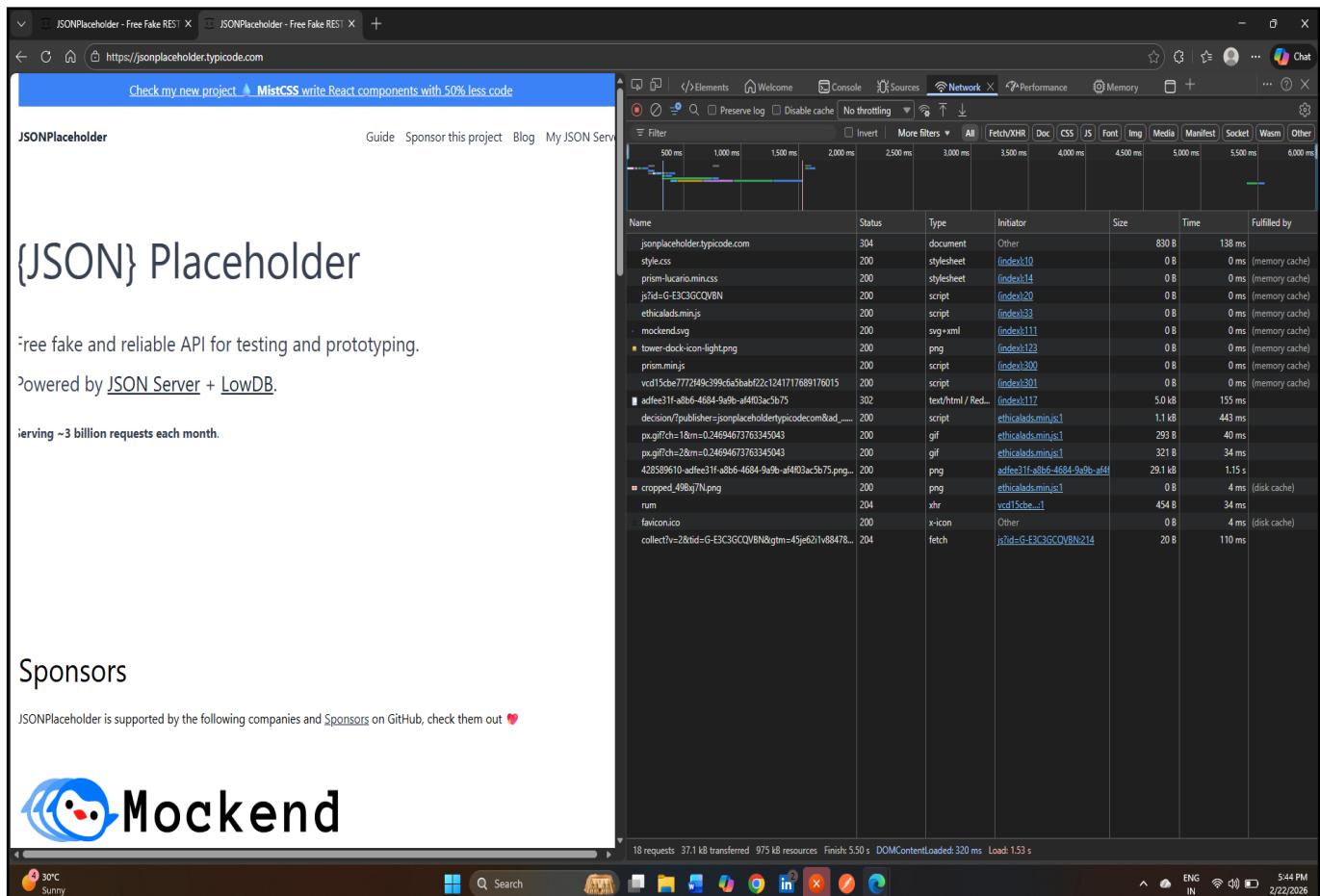


Fig 1 :Browser DevTools Network Analysis

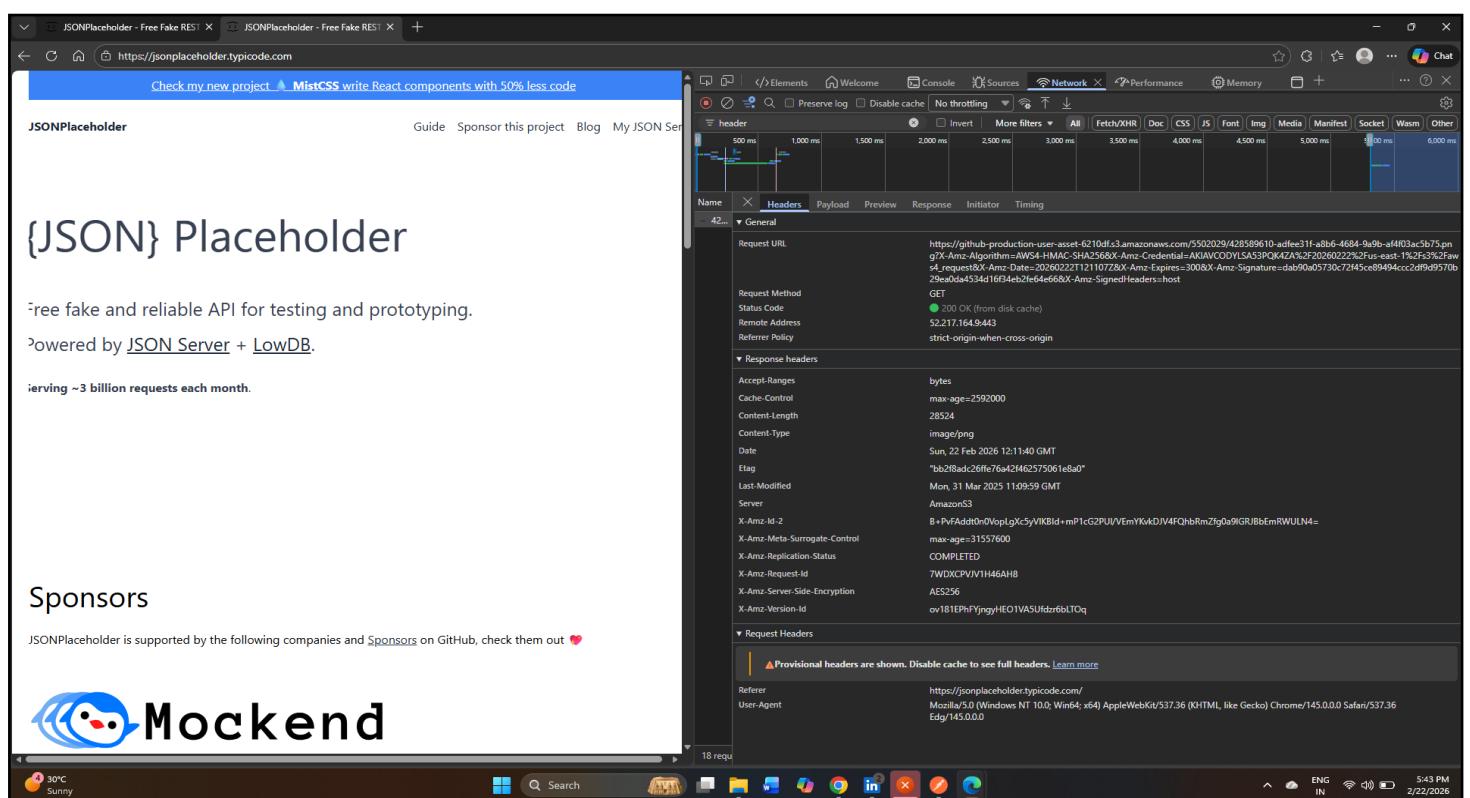
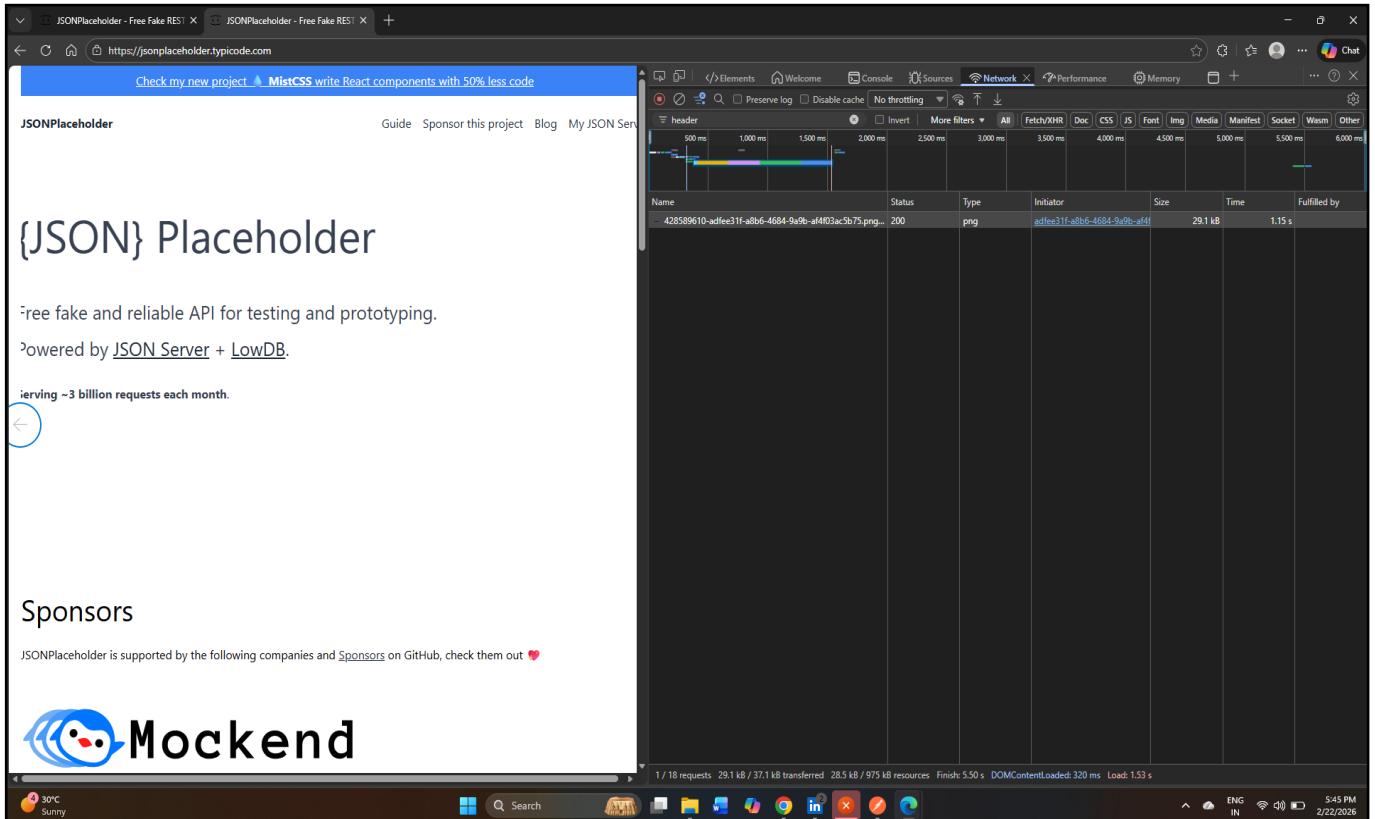


Fig 2 :Browser DevTools Header Analysis

5. Identified Security Risks :

Risk 1: No Authentication Mechanism	
Severity	Medium
Issue	All endpoints are accessible without login, API key, or token.
Business Impact	Unauthorized individuals could access and collect user data.
Remediation Suggestion	<ol style="list-style-type: none">1. Implement API key authentication.2. Use OAuth 2.0 or JWT.

Risk 2: Excessive Data Exposure	
Severity	Medium
Issue	Full user information (email, address, company) is returned.
Business Impact	Privacy risk and regulatory non-compliance.
Remediation Suggestion	<ol style="list-style-type: none">1. Return only required fields2. Apply data minimization

Risk 3: Lack of Authorization Controls (Potential IDOR)

Severity	Medium
Issue	Direct access to specific user IDs without validation.
Business Impact	Users may access other users' data.
Remediation Suggestion	Implement Role-Based Access Control (RBAC).

Risk 4 : No Visible Rate Limiting

Severity	Low–Medium
Issue	Multiple repeated requests are allowed without restriction.
Business Impact	API abuse and potential denial-of-service.
Remediation Suggestion	<ol style="list-style-type: none">1.Implement request limits2.Use API gateway

Risk 5: Error Handling Information Exposure

Severity	Low
Issue	Improper error handling may reveal endpoint structure.
Business Impact	Attackers can map API structure.
Remediation Suggestion	1. Use standardized error responses. 2. Avoid revealing system details

6. Risk Summary Table :

Risk	Severity	Business Impact	Remediation
No Authentication	Medium	Unauthorized Access	Implement Authentication
Excessive Data Exposure	Medium	Privacy Risk	Limit Returned Fields
Lack of Authorization	Medium	Data Leakage	Apply RBAC
No Rate Limiting	Low–Medium	API Abuse	Add Request Limits
Error Handling Exposure	Low	Info Disclosure	Standardized Errors

7. Conclusion :

The JSONPlaceholder API is intended for testing and educational purposes. However, similar security gaps in a production SaaS environment could result in data breaches, privacy violations, and system abuse.

Strengthening authentication, authorization, rate limiting, and response filtering mechanisms would significantly improve API security posture.

***** End of Report *****