# 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 :

- o Postman Tool
- o 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

**<u>The following process was followed :</u>**

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 :

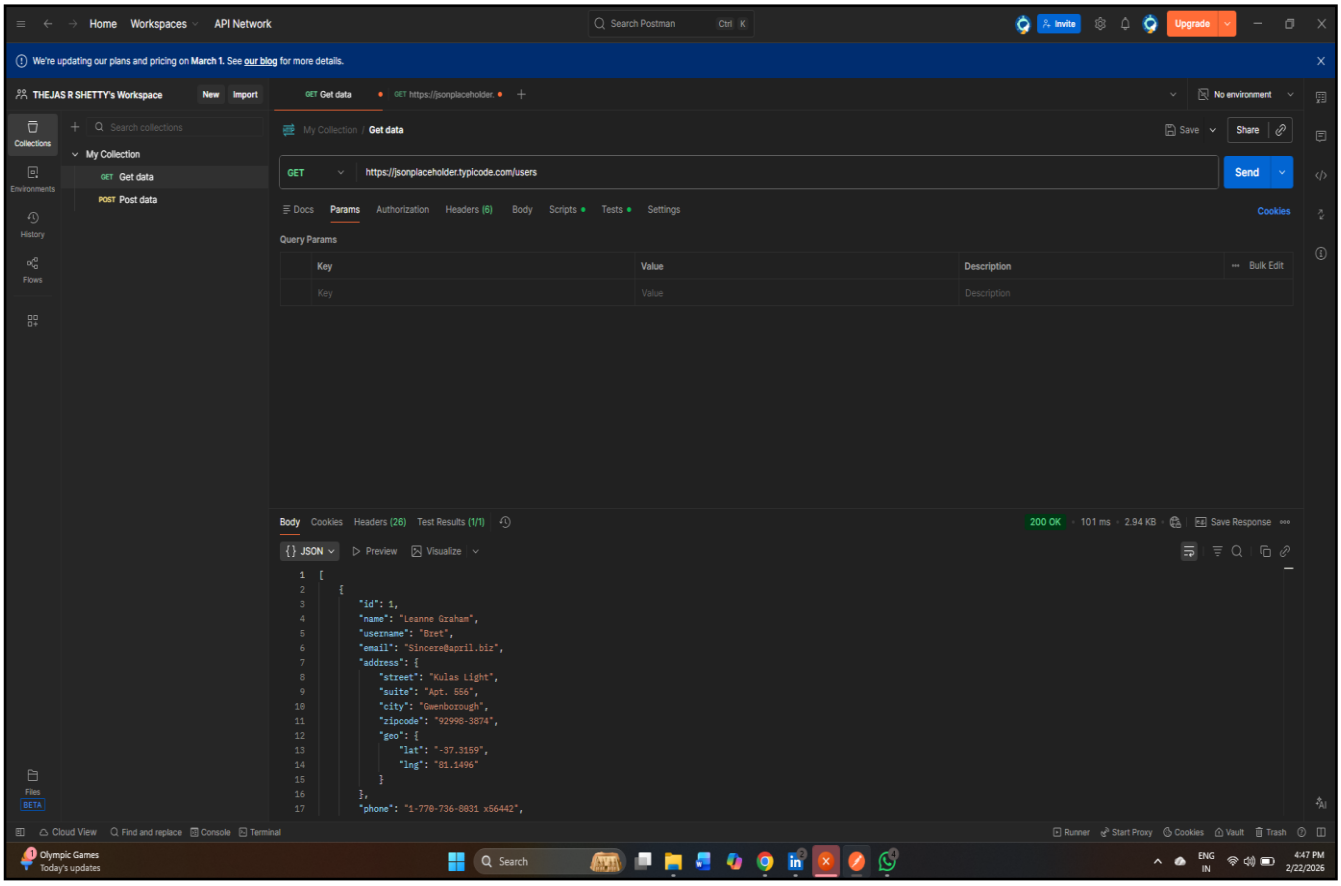



**Fig 1 :**Postman Full Request –GET /users showing 200 OK

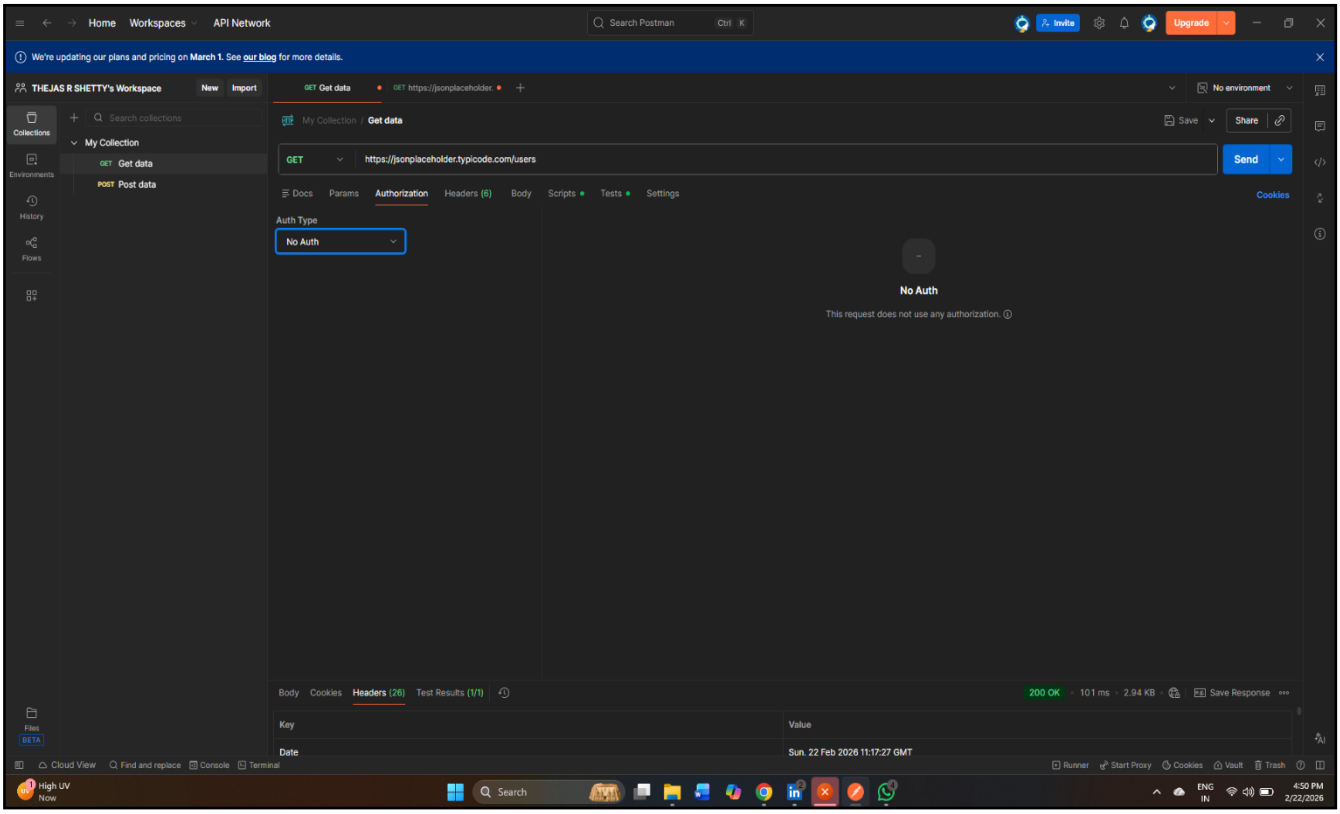



**Fig 2 :**Authorization Tab – Showing "No Auth"

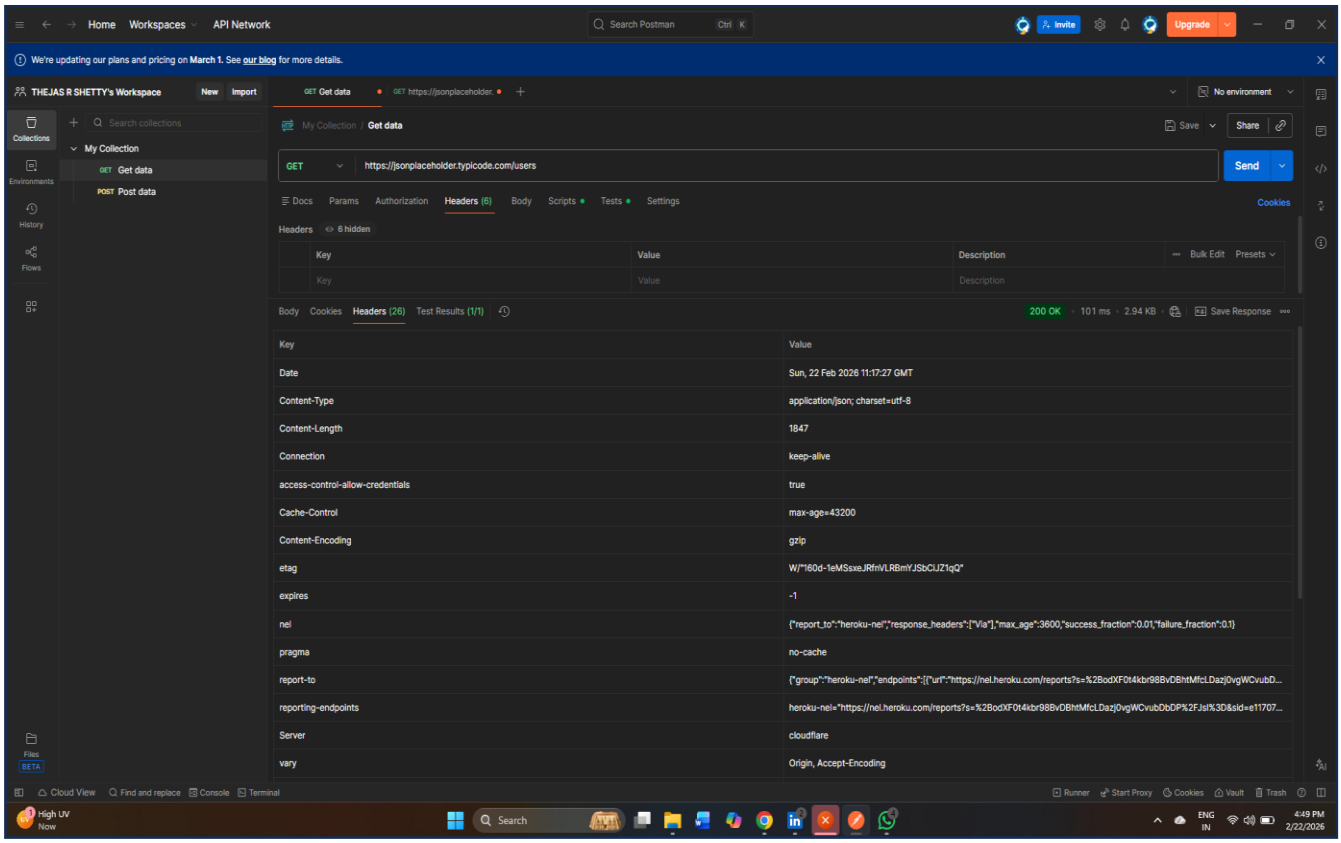



**Fig 3:** Headers – Showing Response Headers

- **4.2 Single User Access Test :**

## Endpoint Tested :

/users/1

## Observation :

1.Direct object access allowed

2.No identity verification required

## Screenshots :

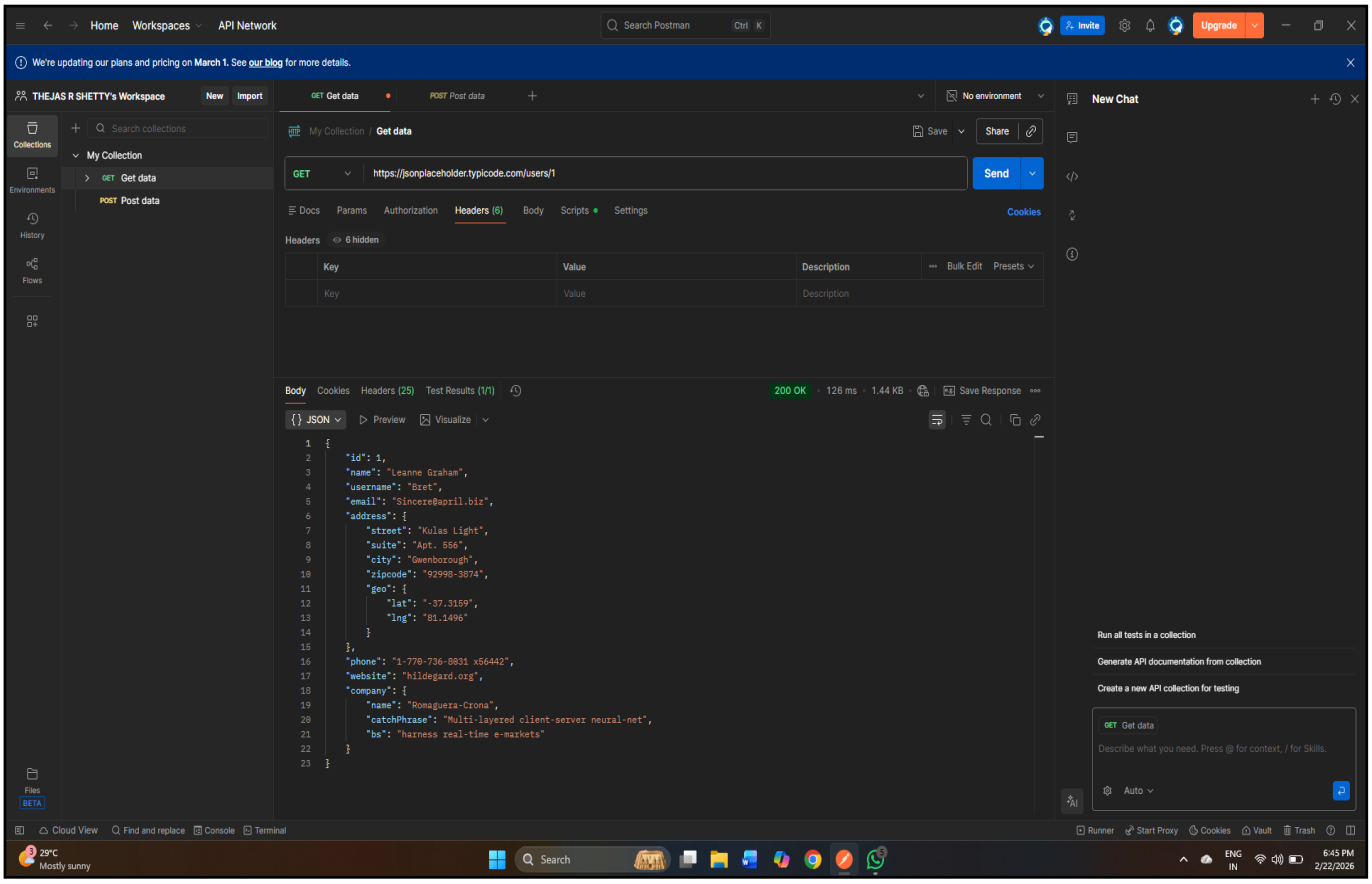



**Fig 1 :**Single User Endpoint – /users/1)

## 4.3 Posts Endpoint Analysis :

**Endpoint Tested :**

GET /posts

**Observation :**

1. All posts publicly accessible

2.No authentication required

**Screenshots :**

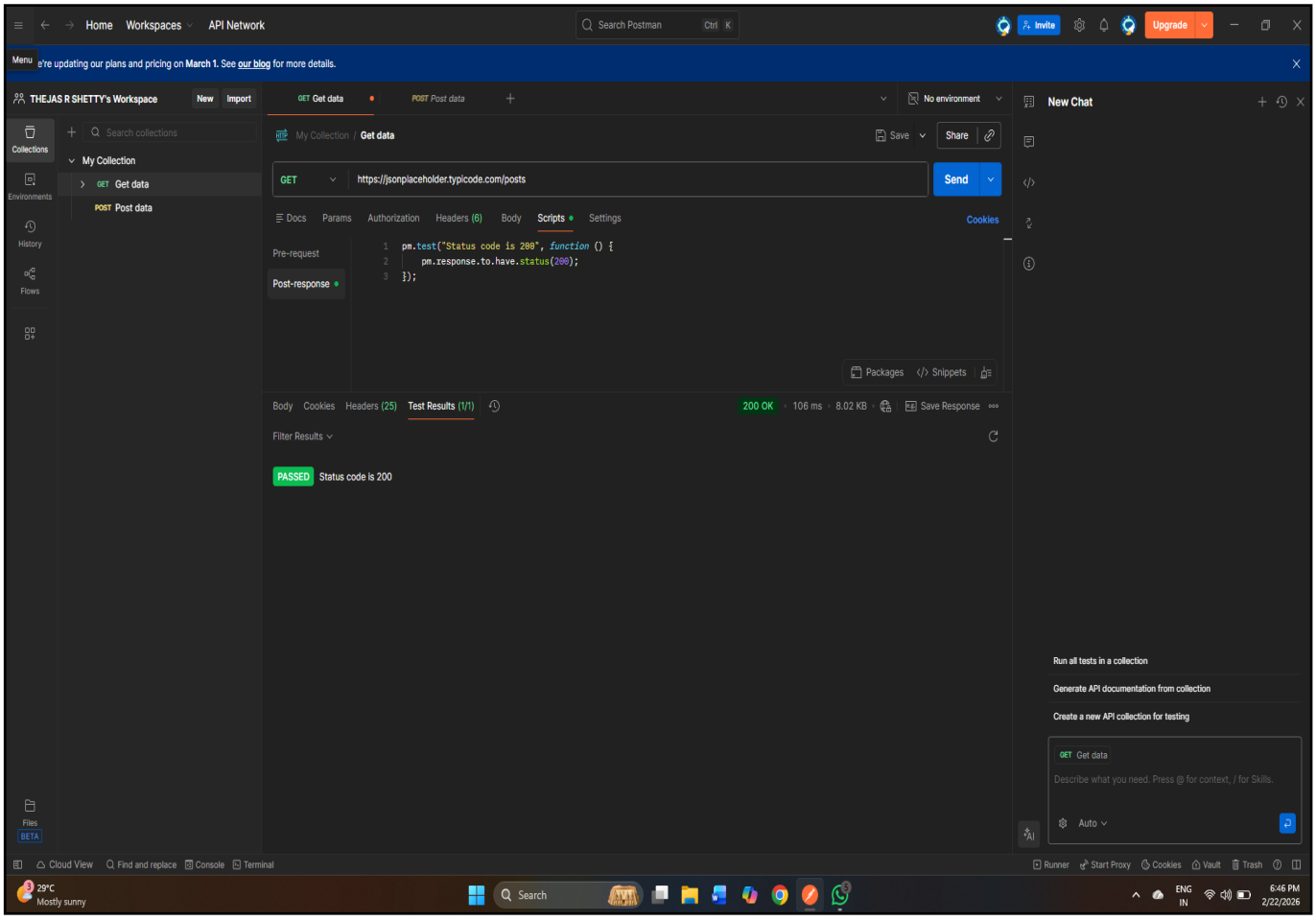



**Fig 1:** Posts Endpoint Screenshot – GET /posts

## 4.4 Comments Endpoint Analysis :

**Endpoint Tested :**

GET /comments

**Observation :**

1.Returns email addresses and comment data

2.No access restriction applied

**Screenshots :**

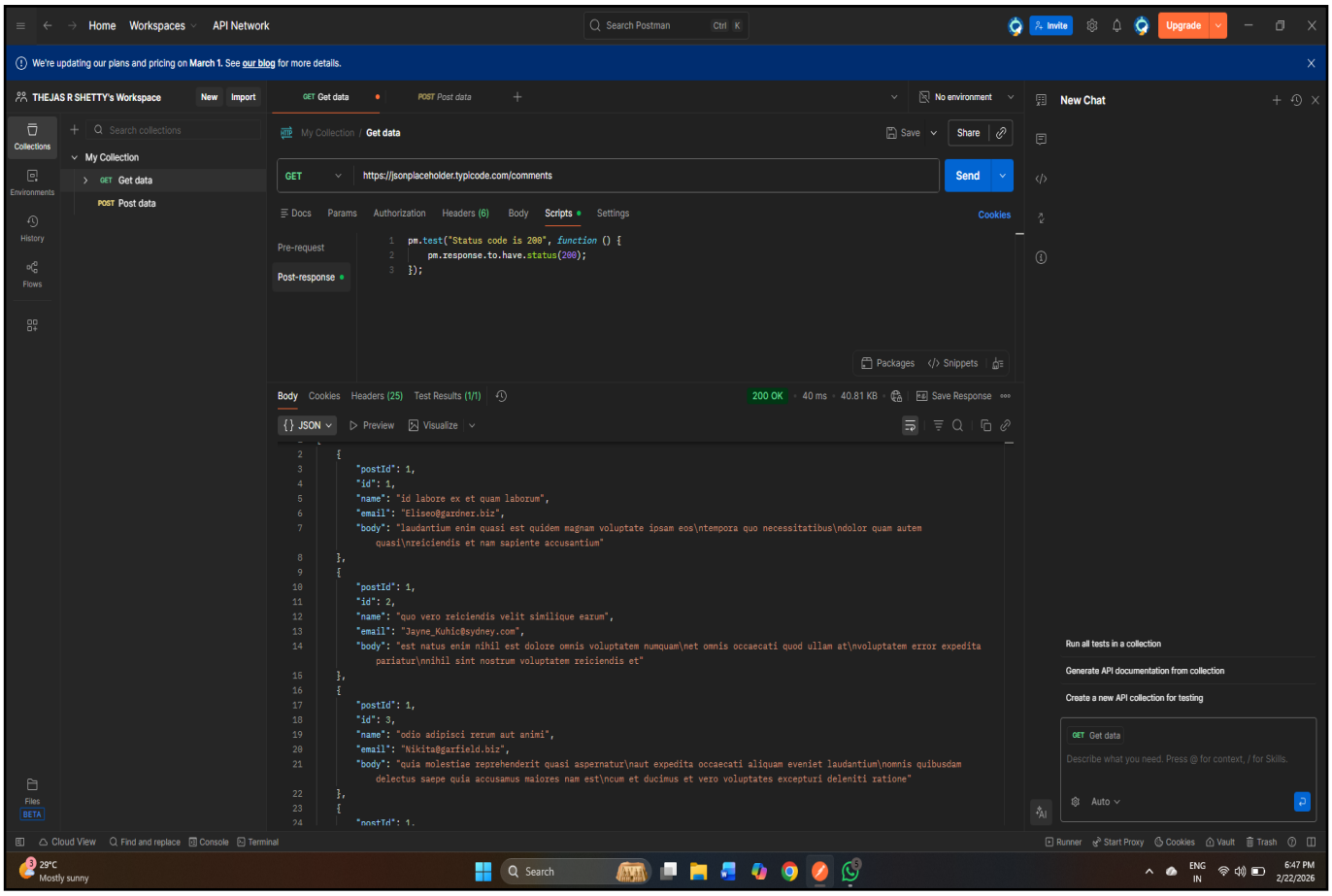



**Fig 1:** Comments Endpoint – GET /comments

- **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 :**

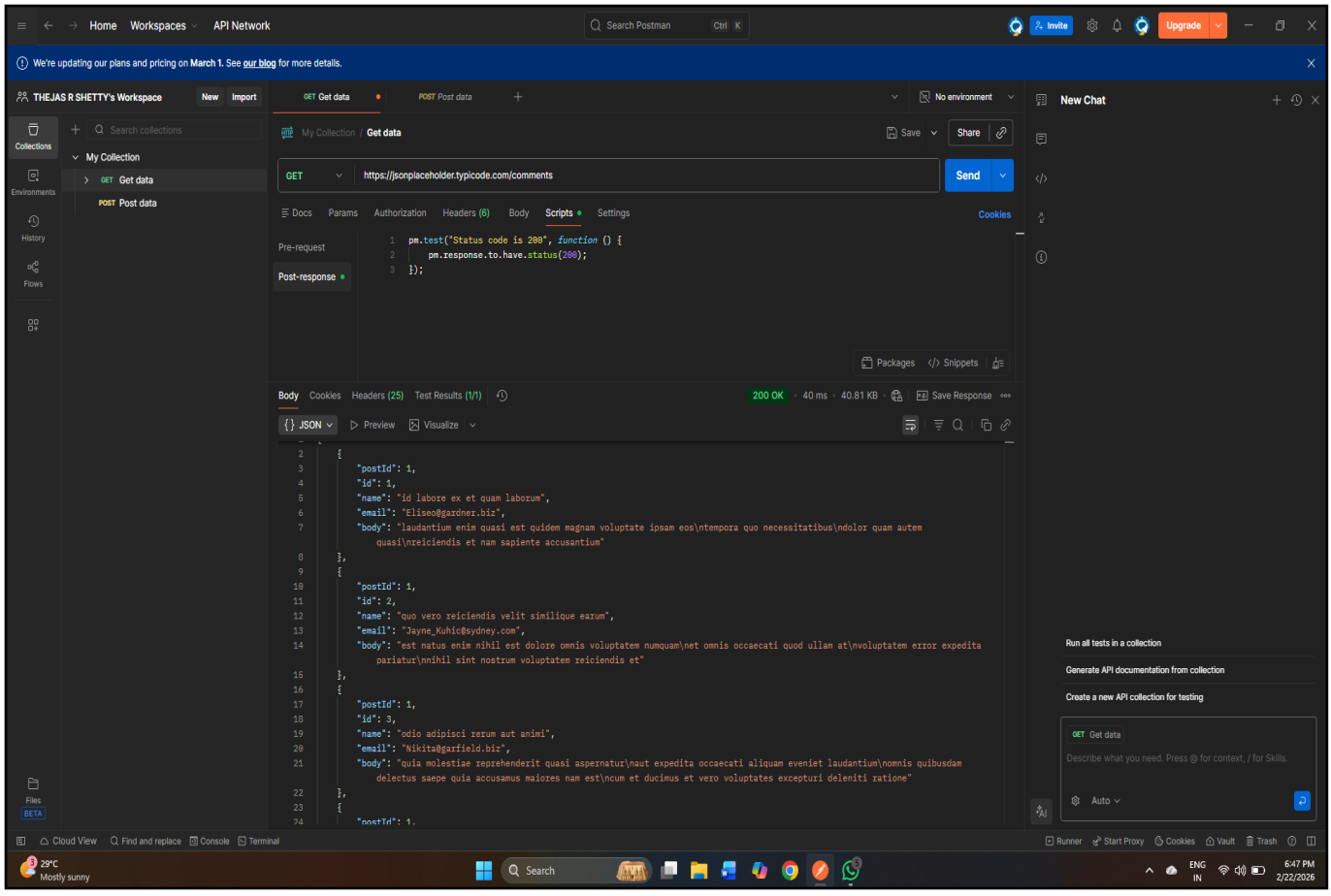



**Fig 1 : 404** Not Found Screenshot – GET /admin

## 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 :

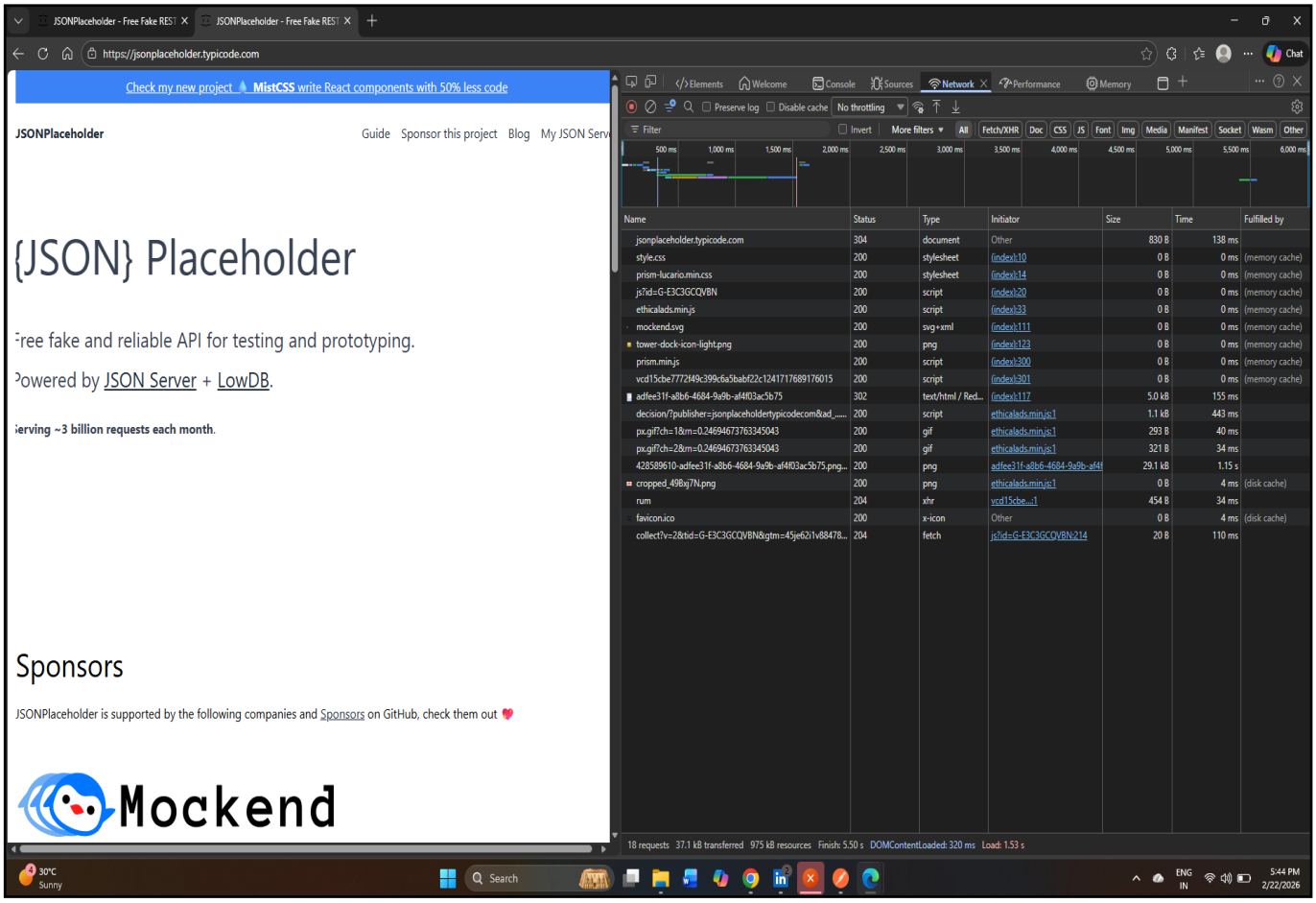
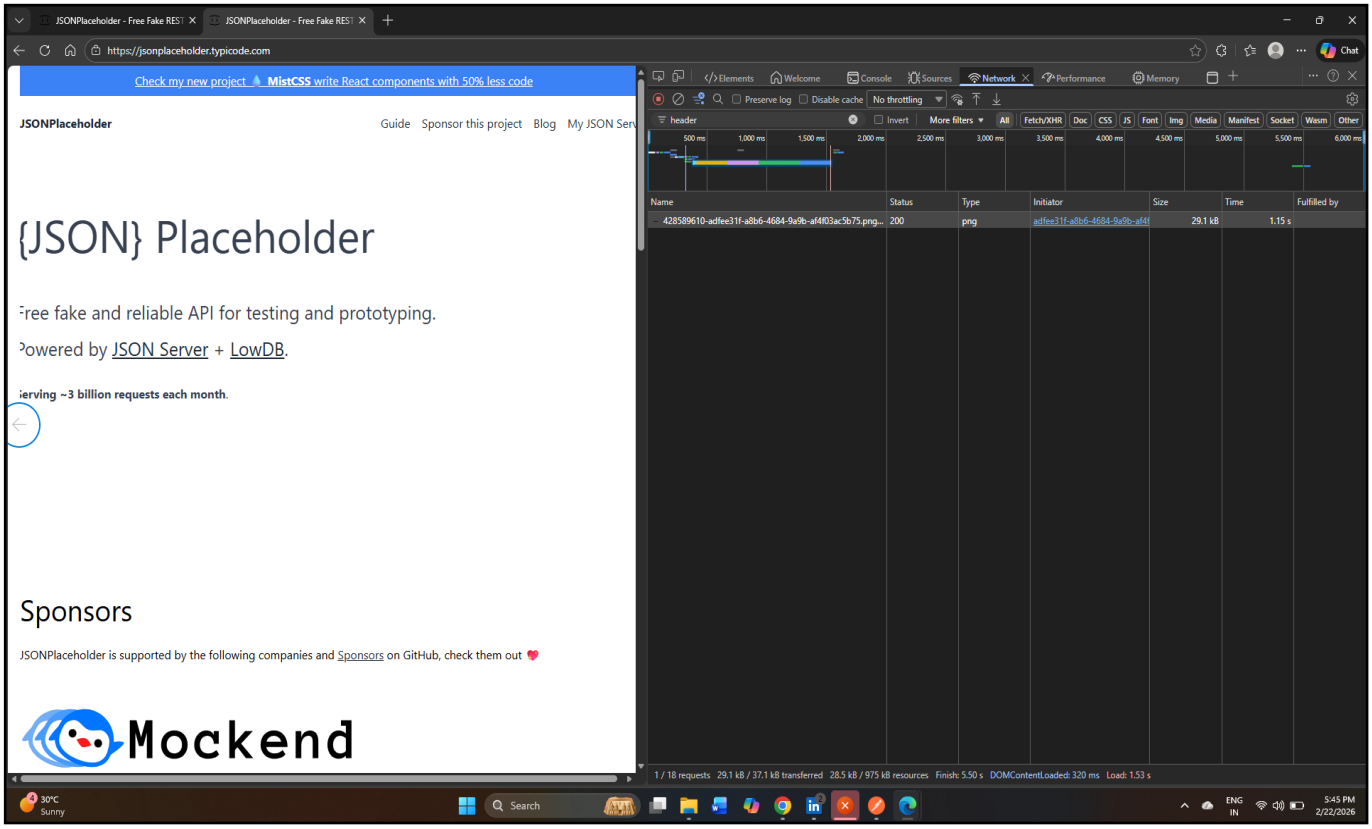



**Fig 1** :Browser DevTools Network Analysis



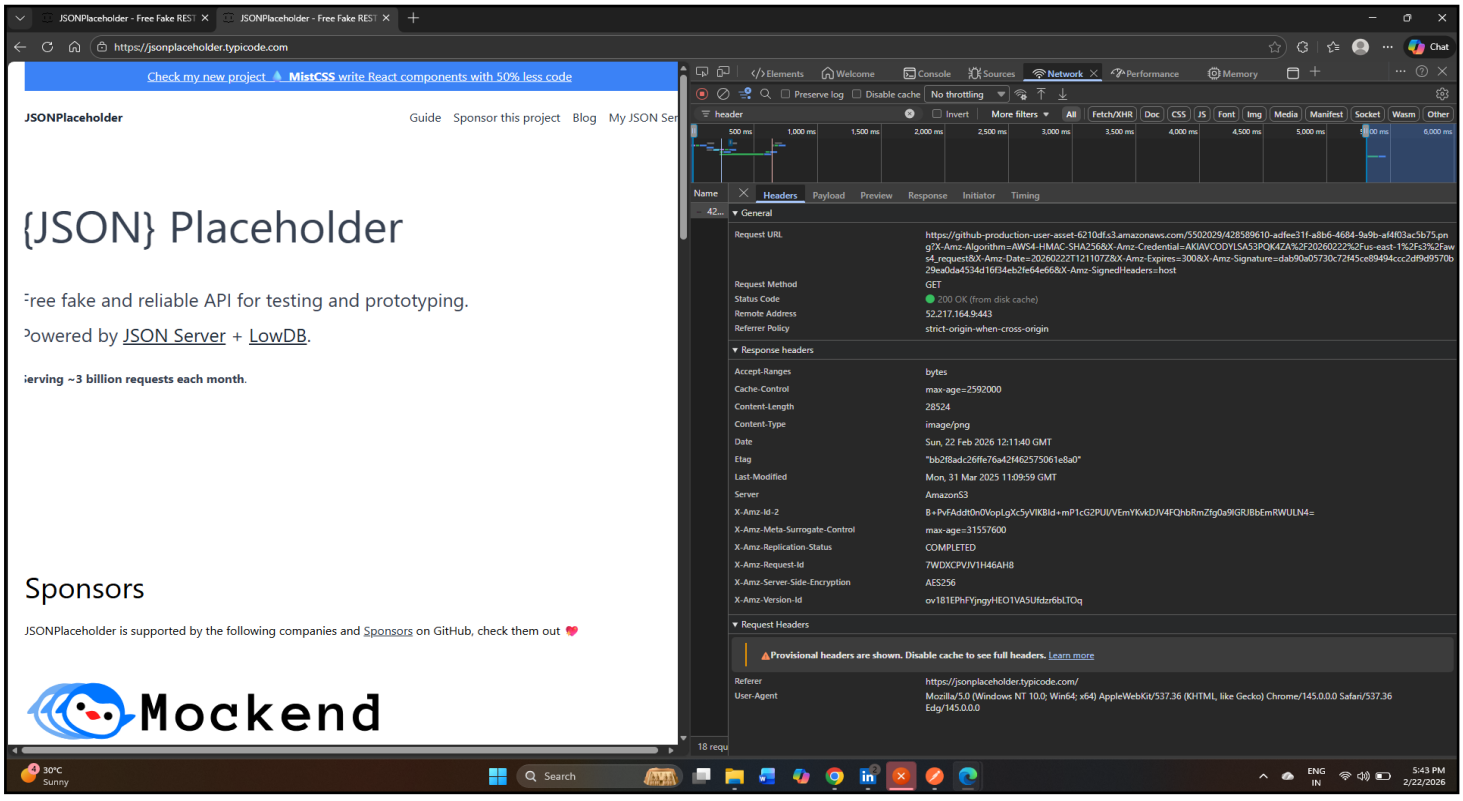




**Fig 2** :Browser DevTools Header Analysis

# 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** | 1. Implement API key authentication.<br>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** | 1. Return only required fields<br>2.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** | 1.Implement request limits<br>2.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.<br>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 **********