API integration debug prompt template

Using API-integrated debugging prompts allows programmers to approach problems in a more systematic and efficient way. Instead of manual and undirected debugging, these prompts support detailed analysis of request/response, headers, validation, logging, and runtime behavior.

In modern software development, API (Application Programming Interface) integration is an indispensable part of connecting systems, exchanging data, and extending functionality. However, this is also the area most prone to errors due to its dependence on many external factors such as networking, authentication, data formats, or changes from third-party services.

Debugging APIs often becomes complicated when errors are not limited to client-side code but also involve server issues, environment configuration, or unexpected data returns. Problems such as timeouts, incorrect endpoints, authentication errors (401/403), or incorrect schema responses can severely disrupt the application.

Therefore, using API-integrated debugging prompts helps programmers approach problems more systematically and efficiently. Instead of manual and undirected debugging, these prompts support detailed analysis of request/response, headers, validation, logging, and runtime behavior. As a result, the debugging process becomes faster, more accurate, and more reusable in various situations.

This article will provide a practical prompt template to help you quickly identify the cause of errors and suggest solutions when working with APIs in different environments such as REST, GraphQL, or microservices.

API integration debug prompt template

Prompt will help troubleshoot API issues.

```
G? l?i v?n ?? tích h?p API này: API: [TÊN API/ENDPOINT] L?i: [THÔNG BÁO L?
I HO?C HÀNH VI] Yêu c?u: [DÁN CODE YÊU C?U] Ph?n h?i: [DÁN PH?N H?I] Tài li
?u: [URL TÀI LI?U API] Hãy giúp tôi: 1. Xác ??nh v?n ?? 2. S?a ??nh d?
ng yêu c?u 3. X? lý các tr??ng h?p ngo?i l? 4. Thêm x? lý l?i phù h?p
```

Best suited for: GPT-5, Gemini 2.5 Pro

How to use sample prompts

Replace the following information in order:

1. API: JSONPlaceholder
2. Errors: Server errors (HTTP 4xx, 5xx), No response received (network/timeout), Request configuration errors, Custom code errors.
3. Request:

```
// apiDebugDemo.js const axios = require("axios"); // T?o instance axios v?
i config c? b?
n const apiClient = axios.create({ baseURL: "https://jsonplaceholder.typicode.co
??c khi g?
i apiClient.interceptors.request.use( (config) => { console.log("=== API REQUEST
?n ???
c apiClient.interceptors.response.use( (response) => { console.log("=== API RESP
? v? response l?
i (4xx, 5xx) console.error("Status:", error.response.status); console.error("Dat
?i ?i nh?ng không nh?n ???
c response console.error("No response received"); } else { // L?
i khi setup request console.error("Error:", error.message); } console.error("===
?
i API async function fetchPosts() { try { const response = await apiClient.get("
?y th? fetchPosts();
```

1. Response: Response in the code =
    1. response.status ? 200 (success)
    2. response.data ? list of articles from API
2. Documentation: https://jsonplaceholder.typicode.com

The example prompt results were generated using GPT-5.

```
G? l?i v?n ?? tích h?p API này: API: JSONPlaceholder L?i: L?i t?
 server (HTTP 4xx, 5xx), Không nh?n ???c response (network/timeout), L?i c?
u hình request, L?i custom trong code Yêu c?
u: // apiDebugDemo.js const axios = require("axios"); // T?
o instance axios v?i config c? b?
n const apiClient = axios.create({ baseURL: "https://jsonplaceholder.typicode.co
??c khi g?
i apiClient.interceptors.request.use( (config) => { console.log("=== API REQUEST
?n ???
c apiClient.interceptors.response.use( (response) => { console.log("=== API RESP
? v? response l?
i (4xx, 5xx) console.error("Status:", error.response.status); console.error("Dat
?i ?i nh?ng không nh?n ???
c response console.error("No response received"); } else { // L?
i khi setup request console.error("Error:", error.message); } console.error("===
?
i API async function fetchPosts() { try { const response = await apiClient.get("
?y th? fetchPosts(); Ph?n h?i: Ph?n h?i trong ?o?n code = response.status ?
 200 (thành công) response.data ? danh sách bài vi?t t? API Tài li?
u: https://jsonplaceholder.typicode.com Hãy giúp tôi: 1. Xác ??nh v?n ??
 2. S?a ??nh d?ng yêu c?u 3. X? lý các tr??ng h?p ngo?i l? 4. Thêm x? lý l?
i phù h?p
```

ChatGPT

Chia sẻ

Dựa trên đoạn code và mô tả của bạn, mình sẽ phân tích theo 4 yêu cầu một cách rõ ràng, thực tế như khi debug production.

---

1. Xác định vấn đề

Code của bạn **chạy được trong điều kiện bình thường**, nhưng có các điểm dễ gây lỗi:

⚠️ Các vấn đề tiềm ẩn

(1) Không phân biệt rõ loại lỗi

- Bạn chỉ log lỗi nhưng không **trả về thông tin lỗi cụ thể**
- `fetchPosts()` luôn trả `undefined` khi lỗi → khó debug phía caller

(2) Timeout có thể gây lỗi giả

- `timeout: 5000` (5s) → nếu mạng chậm sẽ trigger `ECONNABORTED`

(3) Thiếu validate response

- Không kiểm tra `response.status` có đúng 2xx k ↓ g (dù axios mặc định xử lý, nhưng vẫn nên explicit)

ChatGPT

Chia sẻ

- Không kiểm tra `response.status` có đúng 2xx không (dù axios mặc định xử lý, nhưng vẫn nên explicit)

(4) Không retry khi lỗi mạng

- Network fail → bỏ luôn (không retry)

(5) Không chuẩn hóa error object

- Mỗi loại lỗi trả về khác nhau → khó xử lý ở layer trên

---

2. Sửa định dạng yêu cầu (request format)

Hiện tại request OK, nhưng có thể cải thiện:

✅ Cải tiến:

- Thêm `headers` rõ ràng
- Chuẩn hóa params
- Tách config ra reusable

ChatGPT

Chia sẻ

JavaScript

```
const apiClient = axios.create({
    baseURL: "https://jsonplaceholder.typicode.com",
    timeout: 5000,
    headers: {
        "Content-Type": "application/json",
        "Accept": "application/json"
    }
});
```

ChatGPT

Chia sẻ

---

3. Xử lý các trường hợp ngoại lệ

Bạn đã handle 3 case cơ bản, nhưng nên mở rộng:

🎯 Các case cần cover đầy đủ:

Trường hợp	Ví dụ	Cách xử lý
4xx	404, 401	Sai request → báo rõ
5xx	500	Server lỗi → retry
Timeout	ECONNABORTED	retry
Network	no response	retry
Config	sai URL	fail fast

ChatGPT

Chia sẻ

---

4. Thêm xử lý lỗi phù hợp (bản nâng cấp)

Dưới đây là phiên bản **production-ready hơn**:

JavaScript

```
const axios = require("axios");

// Tạo instance axios
const apiClient = axios.create({
    baseURL: "https://jsonplaceholder.typicode.com",
    timeout: 5000,
    headers: {
        "Content-Type": "application/json",
        "Accept": "application/json"
    }
});

// REQUEST INTERCEPTOR
apiClient.interceptors.request.use(
    (config) => {
        console.log("=== API REQUEST ===");
```

ChatGPT

Chia sẻ

JavaScript

```
        console.log("URL:", config.baseURL + config.url);
        console.log("Method:", config.method.toUpperCase());
        console.log("Params:", config.params);
        console.log("====================\n");
        return config;
    },
    (error) => Promise.reject(formatError(error))
);

// RESPONSE INTERCEPTOR
apiClient.interceptors.response.use(
    (response) => {
        console.log("=== API RESPONSE ===");
        console.log("Status:", response.status);
        console.log("====================\n");

        // Validate status
        if (response.status < 200 || response.status >= 300) {
            throw formatError({
                response,
```

ChatGPT

Chia sẻ

JavaScript

```
                response,
                message: "Unexpected status code"
            });
        }

        return response;
    },
    async (error) => {
        const formattedError = formatError(error);

        console.error("=== API ERROR ===");
        console.error(formattedError);
        console.error("====================\n");

        // Retry với lỗi network / timeout
        if (formattedError.isRetryable && error.config && !error.config._retry) {
            error.config._retry = true;
            console.log("Retrying request...");
            return apiClient(error.config);
        }
```

ChatGPT

Chia sẻ

JavaScript

```
        return Promise.reject(formattedError);
    }
);

// Chuẩn hóa lỗi
function formatError(error) {
    if (error.response) {
        return {
            type: "SERVER_ERROR",
            status: error.response.status,
            data: error.response.data,
            message: "Server responded with error",
            isRetryable: error.response.status >= 500
        };
    }

    if (error.code === "ECONNABORTED") {
        return {
            type: "TIMEOUT",
            message: "Request timeout",
```

ChatGPT

Chia sẻ

JavaScript

```
    }

    if (error.code === "ECONNABORTED") {
        return {
            type: "TIMEOUT",
            message: "Request timeout",
            isRetryable: true
        };
    }

    if (error.request) {
        return {
            type: "NETWORK_ERROR",
            message: "No response received",
            isRetryable: true
        };
    }

    return {
        type: "CONFIG_ERROR",
        message: error.message,
```