⚠️ 1. API Errors – Lỗi khi gọi API

🧠 Ý tưởng:

Khi frontend gọi API (REST hoặc GraphQL), có thể gặp lỗi như:

- 400 Bad Request (dữ liệu sai)
- 401 Unauthorized (token hết hạn)
- 403 Forbidden
- 500 Internal Server Error

👉 Để **xử lý lỗi tập trung**, ta nên sử dụng **Axios Interceptors** (hoặc fetch wrapper) để:

- Hiện thông báo lỗi cho người dùng (toast)
- Tự động logout nếu token hết hạn (401)
- Gửi request làm mới token (refresh)
- Ghi log lỗi để theo dõi

🧪 Ví dụ Axios Interceptor:

ts

```
// api.ts
import axios from 'axios';
import { showToast } from './toast'; // function hiện toast

const api = axios.create({ baseURL: '/api' });

api.interceptors.response.use(
  res => res,
  error => {
    const status = error.response?.status;

    if (status === 401) {
      // Token hết hạn, logout hoặc refresh
      logout();
    } else if (status === 500) {
```

```
      showToast('Có lỗi từ server, thử lại sau.');
    } else {
      showToast(error.response?.data?.message || 'Đã xảy ra lỗi.');
    }

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

💥 2. In-App Errors – Lỗi xảy ra trong component

🧠 Ý tưởng:

Một component có thể gây crash toàn bộ ứng dụng nếu không được xử lý cục bộ.

React hỗ trợ **Error Boundaries** – dùng để "bao quanh" 1 vùng UI và bắt lỗi nếu xảy ra. Nó giống như "try-catch" nhưng cho component.

🧪 Ví dụ Error Boundary:

tsx

```
// ErrorBoundary.tsx
import React from 'react';

class ErrorBoundary extends React.Component {
  state = { hasError: false };

  static getDerivedStateFromError() {
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    console.error('Error caught:', error, errorInfo);
    // có thể gửi lên Sentry
  }

  render() {
    if (this.state.hasError) {
      return <h2>Đã xảy ra lỗi. Vui lòng tải lại trang.</h2>;
    }
    return this.props.children;
  }
}
```

tsx
```
// App.tsx
<ErrorBoundary>
  <ChatComponent />
</ErrorBoundary>
```

➡️ Có thể dùng **nhiều ErrorBoundary** ở các khu vực quan trọng như Dashboard, Form, Modal,... thay vì chỉ đặt 1 cái ở cấp cao nhất.

🔎 3. Error Tracking – Theo dõi lỗi trong môi trường thật

🧠 Ý tưởng:

Khi deploy app, bạn không thể thấy lỗi người dùng gặp nếu không ghi lại.

Nên dùng các dịch vụ như:

- **Sentry**
- LogRocket
- Datadog

Chúng giúp:

- Gửi báo cáo lỗi + stacktrace
- Cho biết lỗi xảy ra ở trình duyệt nào, hệ điều hành gì
- Tích hợp dễ với React/NextJS/Vite/Node...
- Cho phép xem được dòng code thật nếu **upload source maps**

⚙️ Tích hợp Sentry với React:

ts
```
import * as Sentry from '@sentry/react';

Sentry.init({
  dsn: 'https://your-sentry-dsn-url',
```

```
  integrations: [new Sentry.BrowserTracing()],
  tracesSampleRate: 1.0,
});
```

✅ **Tổng kết chiến lược xử lý lỗi:**

Loại lỗi	Cách xử lý
Lỗi gọi API	Dùng interceptor (Axios) để xử lý tập trung + toast + logout/refresh
Lỗi render component	Dùng ErrorBoundary để bao component, không crash toàn app
Lỗi production	Tích hợp Sentry để theo dõi + debug từ xa
Lỗi người dùng không thấy	Ghi log cục bộ + gợi ý gửi phản hồi