Case Study 1: E-Commerce Product Listing (Fake Store API)

API: https://fakestoreapi.com/products

Scenario:
An online shop wants to display a catalog of products (title, price, image). Users should also be able to select a **category** (electronics, clothing, jewelry, etc.), and the products should update automatically.

How `useEffect` works here:

- When the component first loads, `useEffect` fetches all products.
- When the user selects a category, the **category state changes**, triggering another API call inside `useEffect` to fetch only that category's products (`/products/category/electronics`).
- While waiting for the data, the UI shows a **"Loading…" message**.
- If the API fails, `useEffect` handles the error and displays a retry option.

Outcome:
Users can switch between categories (e.g., "Electronics", "Clothing") and instantly see updated product listings without refreshing.

Case Study 2: Shopping Cart with Product Details (Fake Store API)

API: https://fakestoreapi.com/products/:id

Scenario:
When a user clicks on a product in the catalog, they should be able to see **detailed information** (title, price, description, rating).

How `useEffect` works here:

- The product ID is passed from the previous page (say `/product/3`).
- `useEffect` runs whenever the product ID changes, fetching the corresponding product details.
- While fetching, a **spinner** is shown.
- If the user navigates to another product, `useEffect` runs again with the new ID, updating details instantly.

Outcome:
Smooth navigation between product pages with real-time details fetched from API.

Case Study 3: User Management Dashboard (JSONPlaceholder API)

API: https://jsonplaceholder.typicode.com/users

Scenario:
An admin dashboard displays a list of all registered users (name, email, phone). Clicking on a user should show their profile details.

How `useEffect` works here:

- On initial load, `useEffect` fetches the list of users and stores it in state.
- When a user is selected, the **selectedUserId** state changes, triggering another `useEffect` call to fetch `/users/:id`.
- Errors are handled gracefully with a message like "Could not load user details."

Outcome:
Admin can switch between users without reloading the whole dashboard. Data always stays fresh.

Case Study 4: Posts with Comments (JSONPlaceholder API)

API:

- Posts → https://jsonplaceholder.typicode.com/posts
- Comments → https://jsonplaceholder.typicode.com/posts/1/comments

Scenario:
A blog app shows a list of posts. When a user clicks a post, its comments should load below it.

How `useEffect` works here:

- On first render, `useEffect` fetches the list of posts.
- When a post is selected, the **postId** state changes, triggering a new `useEffect` to fetch the comments for that post.
- A loading indicator is shown while comments are fetched.

Outcome:
Users can explore blog posts and read comments dynamically, similar to real blogs.

Case Study 5: Todo Tracker (JSONPlaceholder API)

API: https://jsonplaceholder.typicode.com/todos

Scenario:
A productivity app shows a list of todos, with completed and pending tasks.

How `useEffect` works here:

- On initial render, `useEffect` fetches the first 20 todos.
- If the user clicks “Show Completed Only”, the **filter state** updates, triggering another run of `useEffect` that filters todos locally.
- The UI shows a count of completed vs pending tasks.

Outcome:
Users get a clear overview of tasks and can filter between completed and pending seamlessly.