The Axios API Client Architecture

From Zero to Professional Scale

You mentioned you only know `axios.get`. That is the "Command Line" approach. We are going to build an **Enterprise Factory**.

1. The Visual Architecture

Imagine your app is a restaurant.

- **Components (UI):** The Customers ordering food.
- **Service Layer:** The Waiters. They know the menu (endpoints) and take orders.
- **API Client (Axios Instance):** The Kitchen Manager. They ensure every dish is cooked specifically (headers), reject bad orders (error handling), and manage the flow.
- **Interceptors:** The Quality Control line.
  - *Request Interceptor:* Adds garnish (Auth Token) to every plate before it leaves.
  - *Response Interceptor:* Tastes the food. If it's burnt (Error 500), they throw it out and apologize. If it needs salt (Refresh Token), they fix it.

The Flow Graph

```
sequenceDiagram
    participant UI as React Component
    participant Service as User Service
    participant Client as API Client (Axios)
    participant Interceptor as Interceptors
    participant Server as Backend API

    UI->>Service: "Get me the user profile"
    Note right of UI: UI doesn't know about URLs or Axios

    Service->>Client: client.get('/users/me')

    rect rgb(240, 248, 255)
        Note right of Client: REQUEST PHASE
        Client->>Interceptor: Run Request Interceptors
        Interceptor->>Interceptor: Inject Auth Token?
        Interceptor->>Interceptor: Log Request?
        Interceptor->>Server: Send HTTP Request
    end

    rect rgb(255, 240, 245)
        Note right of Server: RESPONSE PHASE
        Server-->>Interceptor: Return JSON Data or Error

        alt Success (200 OK)
            Interceptor-->>Client: Extract data.data
            Client-->>Service: Return clean User Object
            Service-->>UI: Display User Name
```

```
        else Failure (401 Unauthorized)
            Interceptor->>Interceptor: Catch 401 Error
            Interceptor->>UI: Trigger "Force Logout"
        end
    end
```

2. The Golden Rules of Scalability

1. **The Singleton Rule:** Never import `axios` directly in a component. Create **one** instance and reuse it.
2. **The Interceptor Rule:** Never manually add `Authorization` headers in a component. The interceptor does it globally.
3. **The Unwrapping Rule:** Axios returns an object like `{ data: { user: ... }, status: 200, ... }`. Your UI only cares about the `user`. Unwrap the response in the client layer.
4. **The Central Error Rule:** Don't write `try/catch` for 401 (Unauthorized) errors in every component. Handle it in one place (the interceptor).

3. Implementation Strategy

A. The Core (`api-client.js`)

This file creates the Axios instance. It sets the `baseURL` (so you don't type `http://localhost...` 50 times) and attaches the **Interceptors**.

B. The Service Modules (`auth.service.js`)

We group API calls by feature. Instead of scattering URLs, we create methods like `login()`, `register()`, `getProfile()`.

C. The Hook/Component

The component just calls `AuthService.login()`. It's clean, readable, and decoupled.