💦

useFetch Doc

1. Supabase Client Setup (`supabase.ts`)

This file initializes and exports a Supabase client instance for interacting with your database.

```
import { createClient } from "@supabase/supabase-js"

const SUPABASE_URL = process.env.EXPO_PUBLIC_SUPABASE_URL
const SUPABASE_ANON_KEY = process.env.EXPO_PUBLIC_SUPABASE_ANON_KEY

if (!SUPABASE_URL || !SUPABASE_ANON_KEY) {
  throw new Error("Missing Supabase environment variables")
}

export const supabase = createClient(SUPABASE_URL, SUPABASE_ANON_KEY)
```

- **Environment Variables Required**:
    - `EXPO_PUBLIC_SUPABASE_URL` – Your Supabase project URL
    - `EXPO_PUBLIC_SUPABASE_ANON_KEY` – The public API key (Anon key)
- Throws an error if these variables are missing.

2. Custom React Hook: `useFetch` (`useFetch.ts`)

A reusable data-fetching hook that simplifies API calls and handles **loading**, **error**, and **data state**.

- **Generic Type** `<T>` : Ensures type safety for the returned data.

- **Handles API states**:
  - `loading` → Boolean for pending requests.
  - `error` → Captures and returns errors.
  - `data` → Stores the fetched result.
- **Methods provided**:
  - `refetch()` → Manually trigger the fetch again.
  - `reset()` → Clear data, error, and loading states.
- **autoFetch**: If set to `true`, the fetch is triggered automatically on mount.

Fetch Functions Examples:

3. Fetching Listings (`fetchListings.ts`)

- Fetches all records from the `listings` table.
- Supports **case-insensitive search** by title using `.ilike("title", pattern)`.
- Throws an error if Supabase request fails.
- Returns an array of `Listing` objects.

4. Fetching Restaurant Names (`fetchRestaurantName.ts`)

- Fetches restaurant names from the `profiles` table.
- Supports **case-insensitive search** on `restaurant_name`.
- Returns an array of `{ restaurant_name: string }`.

Usage:

```
const { data: listings } = useFetch(() ⇒ fetchListings({ query: "" }), true)
const { data: listings2 } = useFetch(() ⇒ fetchListings({ query: "Special Galbi"
}), true)
const { data: restaurants, loading, error } = useFetch(() ⇒ fetchRestaurantNa
me({}), true)
const { data: restaurant } = useFetch(() ⇒ fetchRestaurantName({ query: "Mango" }), true)
```