Dự án Personal Task Management App

1. Tổng quan cấu trúc & đánh giá nhanh

- **Component**: TaskForm.tsx, TasksPage.jsx
- **State**: store.js, AuthSlice.js
- **API**: supabaseClient.js
- **Hook**: useAuth.js

2. Đánh giá chi tiết & gợi ý cải tiến

a. Coding Convention

- Đặt tên biến, hàm, component đều theo camelCase, PascalCase, khá chuẩn.
- Một số file dùng tiếng Việt trong UI, tiếng Anh trong code, nhất quán và dễ hiểu.
- Nên bổ sung comment cho các hàm quan trọng, props, hoặc logic phức tạp (hiện tại hầu như không có comment).
- Nên thống nhất dùng TypeScript cho toàn bộ codebase (hiện tại mix giữa .js và .tsx/.ts).

b. SOLID Principles

- **Single Responsibility**:
- TaskForm chỉ lo form, TasksPage chỉ lo trang task, tốt.
- useAuth chỉ lo auth, tốt.
- **Open/Closed**:
- Các component chưa có nhiều extension point, nhưng cũng chưa vi phạm.
- **Liskov, Interface Segregation, Dependency Inversion**:
- Chưa áp dụng rõ ràng do app còn đơn giản, nhưng có thể tách interface cho props, dùng DI cho API client nếu mở rộng.

c. Design Pattern

- Đã áp dụng custom hook (useAuth), Redux slice (pattern chuẩn).
- Chưa thấy áp dụng các pattern phức tạp (Factory, Strategy, v.v.), nhưng với quy mô hiện tại là hợp lý.

d. DRY (Don’t Repeat Yourself)

- Không thấy lặp code rõ rệt, nhưng có thể tách các alert/thông báo lỗi thành 1 helper hoặc custom hook để tái sử dụng.
- Logic validate form (TaskForm) có thể tách ra nếu form phức tạp hơn.

e. KISS (Keep It Simple, Stupid)

- Code ngắn gọn, dễ hiểu, ít logic lồng nhau, rất tốt.

- Tuy nhiên, có thể tách nhỏ hơn nữa nếu component/phần logic phát triển thêm.

f. Khả năng mở rộng, bảo trì, test

- Đã tách các phần chính (component, state, api, hook).
- Nên bổ sung test cho các custom hook, component
  (hiện tại chỉ thấy test cho auth page).
- Nên tách config Supabase ra file .env thay vì hardcode.

3. Gợi ý điều chỉnh cụ thể

1. Tách config Supabase ra file .env

```
// src/api/supabaseClient.js
const supabase = createClient(
  process.env.REACT_APP_SUPABASE_URL,
  process.env.REACT_APP_SUPABASE_KEY
);
```

Và thêm vào .env:

```
REACT_APP_SUPABASE_URL=...
REACT_APP_SUPABASE_KEY=...
```

2. Thêm type cho props (với TypeScript)

```
// src/components/TaskForm.tsx
interface TaskFormProps {
  onSave: (data: any) => void;
  onCancel: () => void;
  initialData?: { title?: string; description?: string; dueDate?: string };
  content?: string;
}
export default function TaskForm({ onSave, onCancel, initialData = {}, content }: TaskFormProps) { ... }
```

3. Tách logic alert/thông báo lỗi thành custom hook hoặc helper

```
// src/utils/notify.js
export function notifyError(message) {
  alert(message);
}
```

Và dùng: notifyError("Tạo task thất bại")

4. Thêm comment cho các hàm quan trọng, props, hoặc logic phức tạp

5. Thống nhất dùng TypeScript cho toàn bộ codebase

6. Tách logic validate form nếu form phức tạp hơn

7. Bổ sung test cho custom hook, component

4. Tổng kết

- Codebase hiện tại khá sạch, dễ đọc, tuân thủ KISS, DRY, SRP.
- Nên chuẩn hóa TypeScript, tách config, thêm comment, tách helper, và bổ sung test để tăng khả năng bảo trì, mở rộng.
- Nếu app phát triển lớn hơn, cân nhắc áp dụng thêm các design pattern, interface, DI, v.v.