สรุปแนวทางการพัฒนา Frontend ด้วย React (Vite)

1 ภาพรวมสถาปัตยกรรม

1) Form Handling

- React Hook Form
- Yup (schema validation) / Yup Resolver

2) State Management (Client State - เฉพาะ UI State)

- Redux Toolkit
- ใช้ slices แยกตาม feature

3) Server State / Data Fetching (API Data)

- TanStack Query
- useQuery() สำหรับ GET
- useMutation() สำหรับ POST / PATCH / DELETE

4) Routing

- React Router DOM v6
- AuthRoutes (public area)
- PrivateRoutes (protected area)
- ProtectedRoute (ตรวจสอบ token/login)

2 โครงสร้างโฟลเดอร์ที่ควรใช้ (Feature-based structure)

```
web/
└──── src/
    ├──── components/        # ฟอร์ม + UI components
    ├──── pages/             # หน้าแต่ละฟีเจอร์
    ├──── layouts/           # AuthLayout + MainLayout
    ├──── routes/            # AuthRoutes, PrivateRoutes, ProtectedRoute
    ├──── services/          # API services + React Query hooks
    ├──── schemas/           # Yup form validation
    ├──── features/
    │     └──── auth/        # authSlice + reducers
    ├──── store/             # configureStore()
    ├──── App.jsx
    ├──── main.jsx
```

3 การตั้งค่า Environment (.env)

ใช้ Vite prefix VITE_

4 ส่วนของ Layout

1) AuthLayout.jsx

- ใช้กับหน้า Login, Register, Forgot Password
- มีเพียง Container + Outlet

2) MainLayout.jsx

- ใช้หลัง login แล้ว
- มี Sidebar, Navbar, Footer, Logout, Role-based Menu, <Outlet /> สำหรับแสดงหน้า Dashboard

5 ระบบ Authentication

- ใช้ Redux Toolkit เก็บข้อมูล
- เก็บ token ใน localStorage
- ใช้ ProtectedRoute ตรวจสอบ ถ้าไม่ล็อกอิน redirect ไป /login

6 React Query การจัดการ Login

- ไฟล์ services/authApi.js
- ขั้นตอน ได้แก่

1) เรียก POST /jwt/create/ ได้รับ token

2) ใช้ token เรียก GET /users/me/ ดึง User Object จาก Backend

3) dispatch(loginSuccess()) เก็บ user + role ลง Redux

7 RBAC (Role Based Access Control)

- แยกเป็น 2 กลุ่ม ได้แก่

1) AuthRoutes.jsx สำหรับ public area ใช้ AuthLayout

2) PrivateRoutes.jsx สำหรับ protected area /dashboard/* ครอบด้วย ProtectedRoute ใช้ MainLayout

8 ขั้นตอนสำหรับการพัฒนาเมนูใหม่ในระบบ

1) API Integration

- สร้างไฟล์ เช่น src/services/...Api.js
- โฟกัสการสร้าง Hook สำหรับเข้าถึง API แบบเป็นระบบ
- กำหนด API endpoint ที่จำเป็น เช่น GET (ดึงข้อมูล) POST / PATCH (สร้าง/อัปเดต) DELETE (ลบ)
- สร้าง React Query Hooks เช่น useXxxQuery()
- รองรับ Error Handling / Toast
- invalidate queries หลังจาก update สำเร็จ เช่น queryClient.invalidateQueries(['userProfile'])

ผลลัพธ์ที่ได้

- API พร้อมใช้งาน เชื่อมกับ UI ได้ทันที
- ง่ายต่อการ maintain

2) Form Validation Layer (Yup + React Hook Form)

- สร้างไฟล์ เช่น src/schemas/...Schema.js
- สร้าง Yup Schema แยกจาก UI เช่น profileSchema
- ระบุ validation rules ชัดเจน (required, min, max, regex)

ข้อดีของการแยกเป็น Schema

- UI เรียบง่าย
- นำกลับมาใช้ซ้ำในหลายหน้า
- Logic ความถูกต้องไม่ปนกับ UI

3) UI Components & Business Logic

- สร้างเป็น Component แบบ Reusable
- เชื่อม useForm() YupResolver(schema) Mutation Hooks
- จัดการ Loading / Error State
- แยก “หน้าหลัก” กับ “Form ภายใน” เช่น Profile.jsx = หน้า / ProfileEditForm.jsx = ฟอร์ม

แนวทางมาตรฐาน

- UI ดึงข้อมูลด้วย Query Hook
- อัปเดตด้วย Mutation
- ปิด Modal / Toast เมื่อสำเร็จ
- refresh ผ่าน invalidate query

4) State Management (Redux Slice)

- สร้างไฟล์ เช่น src/features/auth/authSlice.js
- กำหนด state ให้ชัด เช่น user, roles, permissions, filters
- เพิ่ม Reducer เช่น updateUser()
- จัดเก็บข้อมูลร่วมกับ LocalStorage หากจำเป็น

หลักการสำคัญ

- ข้อมูลที่ต้องใช้หลายหน้า ⟶ เก็บ Redux
- ข้อมูลเฉพาะหน้าใดหน้าหนึ่ง ⟶ ใช้ React Query เท่านั้น
- พยายาม sync UI ทันที ไม่ต้อง reload หน้า จะทำให้ UX ดีขึ้นมาก

5) Routing & Navigation Layer

- เพิ่มเส้นทาง (Route)

- เพิ่มเมนูใน Sidebar
- กำหนด permission/role ที่เข้าถึงเมนูได้
- แยกเป็น “Page -> Component -> Form” ตามโครงสร้าง

9 สรุปโครงสร้างสถาปัตยกรรม

- React + Vite
- Redux Toolkit สำหรับ Client State
- React Hook Form + Yup สำหรับ Form
- TanStack Query สำหรับ Server State
- React Router DOM รองรับ layouts (public & private)
- มีระบบ RBAC สำหรับ admin/manager/viewer
- รองรับ JWT Authentication + auto redirect
- แยกโครงสร้างโปรเจคแบบ scalable และ maintainable

10 สรุป Workflow การทำงานโดยรวม

1) เมื่อผู้ใช้เข้าหน้าเว็บ

- App โหลด initial Redux state
- ตรวจสอบ token ที่ localStorage
- ถ้ามี token => isAuthenticated = true

2) เมื่อ Login

- ผู้ใช้ submit form
- ส่ง username/password ไปสร้าง token
- เก็บ token ใน localStorage
- ใช้ token ไปเรียก /users/me
- dispatch(loginSuccess())
- redirect ไป /dashboard

3) เมื่อ Logout

- dispatch(logout())
- ลบ token & user จาก localStorage
- redirect ไป /login
- ProtectedRoute จะ block หน้า dashboard

สรุปขั้นตอนการพัฒนาเมนูใหม่ (Checklist)

1) ออกแบบ API

- ต้องมี endpoint อะไรบ้าง
- ต้องมี Query / Mutation อะไร
- ต้อง invalidate query อะไรหลังอัปเดต

2) สร้าง Schema

- เขียน Yup Schema
- แยก logic validation ออกจาก UI

3) เขียน UI Component

- สร้างฟอร์ม (Form Component)
- จัดการ loading, error
- เชื่อม React Hook Form + Yup + Query/Mutation

4) อัปเดต Redux (ถ้าฟีเจอร์เกี่ยวกับ Global State)

- มี reducer ใหม่หรือไม่
- ต้อง sync กับ LocalStorage หรือไม่

5) เพิ่ม Routing / Sidebar

- เพิ่มเมนูใน Sidebar
- กำหนดสิทธิ์การเข้าถึง
- เพิ่มหน้าใหม่

แหล่งอ้างอิง

- Recommended React Folder Structures -> https://github.com/alan2207/bulletproof-react
- Redux Official Style Guide -> https://redux.js.org/style-guide
- TanStack Query Official Docs -> https://tanstack.com/query/latest/docs/react/guides/queries
- React Hook Form Official -> https://react-hook-form.com/get-started
- Yup Schema Validation -> https://github.com/jquense/yup
- React Router v6 Docs -> https://reactrouter.com/6.28.0/start/tutorial
- TailwindCSS Official -> https://tailwindcss.com/docs/installation/using-vite
- DaisyUI Official -> https://daisyui.com/components/