🚀 A Scalable React Folder Architecture for Real-World Projects

As React developers, we've all hit that point where a growing codebase starts to feel... messy. As features grow and teams scale, a clear and maintainable folder structure becomes crucial. That's why I've been refining the architecture of my React projects, drawing inspiration from real-world experience, my own project needs, and community standards like **Bulletproof React**.

💡 **The Result? A clean, scalable, and team-friendly structure.**

🔍 What This Structure Solves:

✅ Better separation of concerns
✅ Reusability of components and logic
✅ Easier onboarding for new developers
✅ Smooth scalability as features grow
✅ Consistent organization across features and modules

🗂️ Key Highlights:

- **`components/`**: Shared, reusable UI building blocks like Buttons, Modals, Tables.
- **`features/`**: Domain-specific modules with their own API, UI, hooks, services, and types.
- **`layouts/`**: Page wrappers like AuthLayout and MainLayout to standardize layouts.
- **`store/`**: Centralized Redux Toolkit logic with modular slices.
- **`hooks/`**: Globally reusable logic like `useAuth`, `useDebounce`.
- **`utils/`**: Constants, validators, formatters — all those helpful utilities we need every day.
- **`pages/`**: Top-level route views (think: `/home`, `/login`, etc.).
- **`services/`**: Axios instance, error handlers, and token management.
- **`tests/`**: Organized structure for unit and integration testing.

📦 Tech Stack in Mind

- **React + TypeScript**
- **Redux Toolkit (RTK Query)**
- **SCSS Modules**
- **Axios for HTTP**
- **Vite as bundler**

🤝 Inspired by the Community

This isn't reinventing the wheel — it's a pragmatic blend of what works. I've taken cues from community-loved structures like **Bulletproof React**, tweaked for real-world use cases, especially in eCommerce and admin dashboard projects.

🧠 Final Thought

A good folder structure is invisible when done right. It should empower your team, not confuse it. If you're just starting or scaling up, taking the time to organize your codebase pays off **tenfold** in the long run.

Feel free to take inspiration from this, and make it your own.
💬 Would love to hear how you structure your React projects!

🔗 #ReactJS #TypeScript #FrontendArchitecture #WebDevelopment #CleanCode #BulletproofReact #ScalableFrontend #ReactTips #DevCommunity

```
src/
├── assets/                                 # Static
assets like images, icons, fonts
│   ├── images/
│   ├── icons/
│   └── fonts/
├── components/
│    |── common/                            # Shared
reusable UI components
│    │   ├── Button/                        # Button
component (used throughout the app)
│    │   │   ├── Button.tsx
│    │   │   ├── Button.module.scss
│    │   │   └── index.ts
│    │   ├── Modal/                         # Generic
Modal (used globally)
│    │   │   ├── Modal.tsx
│    │   │   ├── Modal.module.scss
│    │   │   └── index.ts
│    │   ├── Table/                         # Generic
Table component
│    │   │   ├── Table.tsx
│    │   │   ├── Table.module.scss
│    │   │   └── index.ts
│    │   └── index.ts
│    ├── layouts/                           # Page
wrappers (auth layout, main layout, etc.)
│        ├── MainLayout/
│        │   ├── MainLayout.tsx
│        │   ├── MainLayout.module.scss
│        │   └── index.ts
│        └── AuthLayout/
│           ├── AuthLayout.tsx
│           ├── AuthLayout.module.scss
│           └── index.ts
├── features/                              # Domain-level
modules
│   ├── product/                           # Product
feature (CRUD, UI, services)
│   │   ├── api/
│   │   │   └── productApi.ts
│   │   ├── components/
│   │   │   └── ProductCard.tsx
│   │   ├── hooks/
│   │   │   └── useProducts.ts
│   │   ├── services/
│   │   │   └── productService.ts         # Optional
non-RTK query APIs
│   │   ├── types/
│   │   │   └── product.ts
│   │   └── index.ts
│   └── user/                             # User feature
(admin/user roles)
│       ├── api/
│       │   └── userApi.ts
│       ├── components/
│       │   └── UserTable.tsx
│       ├── hooks/
│       │   └── useUser.ts
│       ├── services/
│       │   └── userService.ts
│       ├── types/
│       │   └── user.ts
│       └── index.ts
```

```
├── api/                                       # Central API
folder (RTK Query config + slices)
│   ├── baseApi.ts                             # RTK Query base
configuration
│   └── index.ts                               # Export all
APIs
├── hooks/                                     # Global hooks
(reusable logic)
│   ├── useAuth.ts
│   ├── useDebounce.ts
│   └── index.ts
├── routes/                                    # Routing and
route guards
│   ├── AppRoutes.tsx
│   ├── ProtectedRoute.tsx
│   └── index.ts
├── services/                                  # Utilities for
HTTP or interceptors
│   ├── apiClient.ts                           # Axios base
instance (if needed)
│   ├── errorHandler.ts                        # Centralized
error logic
│   └── tokenManager.ts                        # Token
utilities (get, set, clear)
├── store/                                     # Redux store +
slices
│   ├── slices/
│   │   ├── authSlice.ts
│   │   └── productSlice.ts
│   ├── hooks.ts                               # Typed hooks:
useAppSelector, useAppDispatch
```

```
│   └── store.ts                                 # Store
configuration
├── styles/                                      # Global and
scoped SCSS files
│   ├── variables.scss
│   ├── global.scss
│   └── mixins.scss
├── utils/                                       # General helper
functions and constants
│   ├── constants.ts                             # Global
constants (routes, strings)
│   ├── formatters.ts                            # Format
currency, dates, etc.
│   ├── validators.ts                            # Custom
validation logic
│   ├── helpers.ts                               # ID generators,
etc.
│   └── index.ts
├── pages/                                       # Top-level
route views/pages
│   ├── HomePage.tsx
│   ├── LoginPage.tsx
│   ├── ProductPage.tsx
│   └── UserListPage.tsx
├── tests/                                       # Unit and
integration test structure
│   ├── components/
│   ├── pages/
│   └── utils/
├── App.tsx                                      # Main App
component
```

```
├── main.tsx                                  # ReactDOM
render + Vite setup
└── index.html                                # HTML entry
point
```