Hover-like Application Folder Structure (React + TypeScript + Tailwind)

```
hover-app/
├── public/
│   ├── favicon.ico
│   ├── logo.svg
│   ├── images/
│   │   ├── hero/
│   │   ├── icons/
│   │   └── illustrations/
│   └── manifest.json
│
├── src/
│   ├── components/
│   │   ├── ui/
│   │   │   ├── Button/
│   │   │   │   ├── Button.tsx
│   │   │   │   ├── Button.types.ts
│   │   │   │   └── index.ts
│   │   │   ├── Input/
│   │   │   │   ├── Input.tsx
│   │   │   │   ├── Input.types.ts
│   │   │   │   └── index.ts
│   │   │   ├── Modal/
│   │   │   ├── Card/
│   │   │   ├── Badge/
│   │   │   ├── Spinner/
│   │   │   ├── Toast/
│   │   │   ├── Dropdown/
│   │   │   └── index.ts
│   │   │
│   │   ├── layout/
│   │   │   ├── Header/
│   │   │   │   ├── Header.tsx
│   │   │   │   ├── Header.types.ts
│   │   │   │   ├── Navigation/
│   │   │   │   │   ├── Navigation.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   ├── UserMenu/
│   │   │   │   │   ├── UserMenu.tsx
│   │   │   │   │   └── index.ts
│   │   │   │   └── index.ts
│   │   │   ├── Footer/
│   │   │   │   ├── Footer.tsx
│   │   │   │   └── index.ts
│   │   │   ├── Sidebar/
│   │   │   │   ├── Sidebar.tsx
│   │   │   │   └── index.ts
```

```
│   │   ├── Layout.tsx
│   │   └── index.ts
│   │
│   ├── domain/
│   │   ├── DomainSearch/
│   │   │   ├── DomainSearch.tsx
│   │   │   ├── DomainSearch.types.ts
│   │   │   ├── SearchInput.tsx
│   │   │   ├── SearchResults.tsx
│   │   │   └── index.ts
│   │   ├── DomainCard/
│   │   │   ├── DomainCard.tsx
│   │   │   ├── DomainCard.types.ts
│   │   │   └── index.ts
│   │   ├── DomainPricing/
│   │   ├── DomainTransfer/
│   │   ├── DomainManagement/
│   │   └── index.ts
│   │
│   ├── auth/
│   │   ├── LoginForm/
│   │   │   ├── LoginForm.tsx
│   │   │   ├── LoginForm.types.ts
│   │   │   └── index.ts
│   │   ├── RegisterForm/
│   │   ├── ForgotPassword/
│   │   ├── ProtectedRoute/
│   │   │   ├── ProtectedRoute.tsx
│   │   │   └── index.ts
│   │   └── index.ts
│   │
│   ├── dashboard/
│   │   ├── DashboardStats/
│   │   ├── DomainList/
│   │   ├── RecentActivity/
│   │   ├── QuickActions/
│   │   └── index.ts
│   │
│   ├── cart/
│   │   ├── CartDrawer/
│   │   ├── CartItem/
│   │   ├── Checkout/
│   │   └── index.ts
│   │
│   └── common/
│       ├── SearchBar/
│       ├── Pagination/
```

```
│   │   ├── LoadingStates/
│   │   ├── ErrorBoundary/
│   │   │   ├── ErrorBoundary.tsx
│   │   │   └── index.ts
│   │   └── index.ts
│   │
│   ├── pages/
│   │   ├── Home/
│   │   │   ├── Home.tsx
│   │   │   ├── Home.types.ts
│   │   │   ├── HeroSection.tsx
│   │   │   ├── FeaturesSection.tsx
│   │   │   ├── PricingSection.tsx
│   │   │   └── index.ts
│   │   ├── Domains/
│   │   │   ├── DomainSearch.tsx
│   │   │   ├── DomainDetails.tsx
│   │   │   ├── DomainTransfer.tsx
│   │   │   └── index.ts
│   │   ├── Dashboard/
│   │   │   ├── Dashboard.tsx
│   │   │   ├── MyDomains.tsx
│   │   │   ├── Account.tsx
│   │   │   └── index.ts
│   │   ├── Auth/
│   │   │   ├── Login.tsx
│   │   │   ├── Register.tsx
│   │   │   └── index.ts
│   │   ├── Pricing/
│   │   │   ├── Pricing.tsx
│   │   │   └── index.ts
│   │   ├── Support/
│   │   │   ├── Support.tsx
│   │   │   └── index.ts
│   │   └── About/
│   │       ├── About.tsx
│   │       └── index.ts
│   │
│   ├── hooks/
│   │   ├── useAuth.ts
│   │   ├── useDomainSearch.ts
│   │   ├── useCart.ts
│   │   ├── useLocalStorage.ts
│   │   ├── useApi.ts
│   │   ├── useDebounce.ts
│   │   └── index.ts
│   │
```

```
|   ├── services/
|   |   ├── api/
|   |   |   ├── client.ts
|   |   |   ├── domains.ts
|   |   |   ├── auth.ts
|   |   |   ├── payments.ts
|   |   |   ├── users.ts
|   |   |   └── index.ts
|   |   ├── auth/
|   |   |   ├── authService.ts
|   |   |   ├── tokenManager.ts
|   |   |   └── index.ts
|   |   └── utils/
|   |       ├── validation.ts
|   |       ├── formatting.ts
|   |       ├── constants.ts
|   |       └── index.ts
|   |
|   ├── store/
|   |   ├── index.ts
|   |   ├── slices/
|   |   |   ├── authSlice.ts
|   |   |   ├── domainSlice.ts
|   |   |   ├── cartSlice.ts
|   |   |   ├── uiSlice.ts
|   |   |   └── index.ts
|   |   └── middleware/
|   |       ├── apiMiddleware.ts
|   |       └── index.ts
|   |
|   ├── styles/
|   |   ├── globals.css
|   |   ├── tailwind.css
|   |   └── components.css
|   |
|   ├── utils/
|   |   ├── helpers.ts
|   |   ├── formatters.ts
|   |   ├── validators.ts
|   |   ├── constants.ts
|   |   ├── config.ts
|   |   ├── classNames.ts
|   |   └── index.ts
|   |
|   ├── types/
|   |   ├── domain.ts
|   |   ├── user.ts
```

```
│   ├── api.ts
│   ├── auth.ts
│   ├── cart.ts
│   ├── common.ts
│   └── index.ts
│
├── context/
│   ├── AuthContext.tsx
│   ├── ThemeContext.tsx
│   ├── CartContext.tsx
│   ├── AuthContext.types.ts
│   ├── ThemeContext.types.ts
│   ├── CartContext.types.ts
│   └── index.ts
│
├── assets/
│   ├── icons/
│   │   ├── index.ts
│   │   └── svg/
│   ├── images/
│   └── fonts/
│
├── lib/
│   ├── axios.ts
│   ├── queryClient.ts
│   └── validations.ts
│
├── App.tsx
├── main.tsx
└── vite-env.d.ts
│
├── tests/
│   ├── __mocks__/
│   ├── components/
│   │   ├── ui/
│   │   └── domain/
│   ├── pages/
│   ├── services/
│   ├── utils/
│   ├── setup.ts
│   └── test-utils.tsx
│
├── docs/
│   ├── README.md
│   ├── API.md
│   ├── COMPONENTS.md
│   └── DEPLOYMENT.md
```

```
|
├── .env.example
├── .env.local
├── .env.development
├── .env.production
├── .gitignore
├── package.json
├── package-lock.json
├── tsconfig.json
├── tsconfig.node.json
├── tailwind.config.ts
├── postcss.config.js
├── vite.config.ts
├── vitest.config.ts
├── eslint.config.js
├── prettier.config.js
└── README.md
```

Key Structure Principles

1. TypeScript Configuration

- All files use `.tsx` for React components and `.ts` for utilities
- Separate type definitions in `.types.ts` files
- Comprehensive type exports through `index.ts` files
- Strong typing for API responses, component props, and state

2. Tailwind CSS Integration

- No CSS modules needed - all styling through Tailwind classes
- `tailwind.css` imports base Tailwind styles
- `components.css` for custom component styles using `@apply`
- Utility for conditional className joining (`classNames.ts`)

3. Components Organization

- **ui/**: Reusable UI components with TypeScript interfaces
- **layout/**: Layout-specific components (header, footer, sidebar)
- **domain/**: Domain-specific business logic components
- **auth/**: Authentication-related components
- **dashboard/**: User dashboard specific components

4. File Naming Conventions

- Components: `PascalCase.tsx`
- Types: `PascalCase.types.ts`
- Hooks: `camelCase.ts` (prefixed with 'use')
- Utils: `camelCase.ts`
- Constants: `UPPER_CASE.ts`

5. **TypeScript Features**

- Strict type checking enabled
- Interface definitions for all props
- Generic types for reusable components
- Utility types for API responses
- Enum types for constants

6. **Tailwind Best Practices**

- Component-first approach
- Custom design tokens in config
- Responsive design utilities
- Dark mode support
- Component composition over custom CSS

7. **Testing Structure**

- Component tests alongside source files
- Integration tests in dedicated folders
- Mock utilities and test helpers
- TypeScript support in tests

8. **Development Tools**

- ESLint with TypeScript rules
- Prettier for code formatting
- Vitest for unit testing
- Husky for git hooks
- Path mapping in TypeScript config

Example Component Structure

typescript

```tsx
// Button.types.ts
export interface ButtonProps {
  variant: 'primary' | 'secondary' | 'danger';
  size: 'sm' | 'md' | 'lg';
  children: React.ReactNode;
  onClick?: () => void;
  disabled?: boolean;
  className?: string;
}

// Button.tsx
import React from 'react';
import { ButtonProps } from './Button.types';
import { cn } from '@/utils/classNames';

export const Button: React.FC<ButtonProps> = ({
  variant = 'primary',
  size = 'md',
  children,
  onClick,
  disabled = false,
  className,
  ...props
}) => {
  const baseClasses = 'font-medium rounded-lg transition-colors focus:outline-none focus:ring-2';

  const variantClasses = {
    primary: 'bg-blue-600 text-white hover:bg-blue-700 focus:ring-blue-500',
    secondary: 'bg-gray-200 text-gray-900 hover:bg-gray-300 focus:ring-gray-500',
    danger: 'bg-red-600 text-white hover:bg-red-700 focus:ring-red-500'
  };

  const sizeClasses = {
    sm: 'px-3 py-1.5 text-sm',
    md: 'px-4 py-2 text-base',
    lg: 'px-6 py-3 text-lg'
  };

  return (
    <button
      className={cn(
        baseClasses,
        variantClasses[variant],
        sizeClasses[size],
        disabled && 'opacity-50 cursor-not-allowed',
        className
```

```
      )}
      onClick={onClick}
      disabled={disabled}
      {...props}
    >
      {children}
    </button>
  );
};

// index.ts
export { Button } from './Button';
export type { ButtonProps } from './Button.types';
```

This structure provides a scalable, maintainable foundation for a modern React application with TypeScript and Tailwind CSS, similar to domain registration platforms like Hover.