## **PROJECT STRUCTURE & BEST PRACTICES**

### **4.1 Why Structure Matters**

Without a clear structure:

* Files get scattered and unmanageable
* Reusability drops, code duplication increases
* Onboarding new developers becomes painful
* Debugging becomes slower as app grows

A good structure separates concerns:

* UI components
* Navigation logic
* Business logic (hooks, services)
* Config & constants
* Assets

### **4.2 Recommended Folder Structure**

MyApp/

src/

assets/

fonts/

images/

icons/

components/ # Reusable UI parts (Button, Card, etc.)

navigation/ # Stack, tab, drawer navigators

screens/ # Full-page UI components

HomeScreen/

index.tsx

styles.ts

DetailsScreen/

index.tsx

styles.ts

services/ # API calls, backend logic

hooks/ # Custom hooks for data fetching, auth, etc.

utils/ # Helper functions (formatting, calculations)

constants/ # Colors, sizes, strings, API base URLs

types/ # Global TypeScript types & interfaces

App.tsx

react-native.config.js

### **4.3 File Naming Conventions**

* Components & screens → PascalCase (HomeScreen.tsx)
* Functions, variables, hooks → camelCase
* Constants → UPPER\_SNAKE\_CASE
* Styles in separate styles.ts per screen for clarity

### **4.4 Example Component Pattern**

src/components/PrimaryButton/index.tsx

import React from 'react';

import {TouchableOpacity, Text, StyleSheet} from 'react-native';

interface Props {

label: string;

onPress: () => void;

}

export default function PrimaryButton({label, onPress}: Props) {

return (

<TouchableOpacity style={styles.button} onPress={onPress}>

<Text style={styles.text}>{label}</Text>

</TouchableOpacity>

);

}

const styles = StyleSheet.create({

button: {

backgroundColor: '#6200EE',

padding: 12,

borderRadius: 8,

alignItems: 'center',

},

text: {

color: '#fff',

fontSize: 16,

},

});

### **4.5 Example Screen Pattern**

src/screens/HomeScreen/index.tsx

import React from 'react';

import {View, Text} from 'react-native';

import PrimaryButton from '../../components/PrimaryButton';

import styles from './styles';

export default function HomeScreen({navigation}) {

return (

<View style={styles.container}>

<Text>Home Screen</Text>

<PrimaryButton

label="Go to Details"

onPress={() => navigation.navigate('Details')}

/>

</View>

);

}

src/screens/HomeScreen/styles.ts

import {StyleSheet} from 'react-native';

export default StyleSheet.create({

container: {flex: 1, justifyContent: 'center', alignItems: 'center'},

});

### **4.6 API Services Example**

src/services/api.ts

import axios from 'axios';

export const api = axios.create({

baseURL: 'https://example.com/api',

timeout: 5000,

});

src/services/userService.ts

import {api} from './api';

export const getUsers = () => api.get('/users');

### **4.7 Constants Example**

src/constants/colors.ts

export const COLORS = {

primary: '#6200EE',

secondary: '#03DAC6',

background: '#FFFFFF',

};

✅ At the end you have:

* Clean folder structure
* Naming conventions for consistency
* Reusable component pattern
* Centralized constants, services, and hooks