Creating a **production-grade, highly complex React Native project** involves a thoughtful folder structure for scalability, maintainability, and organization. Below is a recommended folder structure tailored for a high-complexity project:

Root Folder Structure

plaintext
Copy code

```
project-root/
├── android/                # Android native code
├── ios/                    # iOS native code
├── src/                    # Main source code for the app
│   ├── assets/             # Static assets (images, fonts, icons,
etc.)
│   ├── components/         # Reusable UI components
│   ├── config/             # App configuration files (e.g., themes,
environment variables)
│   ├── constants/          # App-wide constants
│   ├── hooks/              # Custom hooks
│   ├── modules/            # Feature-specific modules
│   │   └── [FeatureName]/  # Individual feature modules
│   │       ├── components/ # Feature-specific components
│   │       ├── screens/    # Screens for the feature
│   │       ├── services/   # API services for the feature
│   │       └── styles/     # Styles for the feature
│   ├── navigation/         # All navigation-related code
│   ├── redux/              # Redux store and slices
│   ├── screens/            # Main app screens (high-level
navigation)
│   ├── services/           # Shared API calls or integrations
(e.g., auth, analytics)
│   ├── utils/              # Utility functions/helpers
│   └── types/              # TypeScript definitions and interfaces
├── scripts/                # Custom scripts for automation
├── .env                    # Environment variables
├── .gitignore              # Files to be ignored by git
├── app.json                # Expo/React Native app configuration
├── babel.config.js         # Babel configuration
├── index.js                # App entry point
├── metro.config.js         # Metro bundler configuration
├── package.json            # Node dependencies and scripts
└── tsconfig.json           # TypeScript configuration
```

Detailed Explanation

1. `src/`

Contains all your app logic, broken down into reusable and modular components.

`assets/`: Store static files like images, fonts, or videos. For example:
plaintext
Copy code
```
assets/
├── images/
├── fonts/
└── icons/
```

-

`components/`: Houses reusable UI components like buttons, cards, and modal dialogs.
plaintext
Copy code
```
components/
├── Button/
│   ├── Button.tsx
│   ├── Button.styles.ts
│   └── index.ts
```

-

`config/`: Stores configuration-related files, like themes, environment variables, or app constants.
plaintext
Copy code
```
config/
├── theme.ts
└── env.ts
```

-
- **`constants/`**: Contains app-wide constants like API URLs or static text strings.
- **`hooks/`**: Custom React hooks to encapsulate logic (e.g., `useAuth`, `useFetch`).

`modules/`: Feature-specific directories to encapsulate everything a feature requires (components, screens, services, etc.). For instance:
plaintext
Copy code
```
modules/
```

```
├── Authentication/
│   ├── components/
│   ├── screens/
│   ├── services/
│   └── styles/
├── Dashboard/
└── Profile/
```

-

`navigation/`: All navigation code, including stacks, tabs, and navigators.
plaintext
Copy code

```
navigation/
├── AppNavigator.tsx
├── AuthNavigator.tsx
├── RootNavigator.tsx
└── index.ts
```

-

`redux/`: Redux setup, including slices and the store.
plaintext
Copy code

```
redux/
├── slices/
│   ├── authSlice.ts
│   ├── userSlice.ts
│   └── index.ts
├── store.ts
└── middlewares/
```

-
- **`screens/`**: Top-level screens that represent routes in your app.
- **`services/`**: For managing API calls, analytics, or third-party integrations like Firebase.
- **`utils/`**: Helper functions like debouncing, validation, and formatting.
- **`types/`**: All TypeScript type definitions.

---

2. Root-Level Configuration

- **`android/` and `ios/`**: Native codebases for Android and iOS.

- **`scripts/`**: Custom Node.js scripts for automating tasks (e.g., cleaning builds, generating assets).
- **`.env`**: Store sensitive environment variables like API keys (use `react-native-dotenv`).
- **`metro.config.js`**: Customize the Metro bundler, e.g., for resolving custom paths.

Best Practices

1. **Code Splitting**: Keep your code modular to avoid bloated files.
2. **Feature-Based Architecture**: Group related files by feature for maintainability.
3. **Environment Configurations**: Use `.env` files for different environments (dev, staging, production).

Testing: Add unit and integration tests using Jest and React Native Testing Library.
plaintext
Copy code
```
src/
├── __tests__/          # Test files
├── jest.config.js      # Jest configuration
```

4.
5. **Linting**: Use ESLint and Prettier to enforce code style and consistency.
6. **Version Control**: Follow Git workflows with proper commit messages and branching strategies.
7. **Documentation**: Add comments and maintain a `README.md` file.

Create React Native Project

Prerequisite for the react native project:
Env Setup:-https://reactnative.dev/docs/set-up-your-environment

Use the official documentation:
1. CLI:-https://reactnative.dev/docs/getting-started-without-a-framework
2. Expo:-https://reactnative.dev/docs/environment-setup

Sample Project in React Native

Below is a setup for a **Learning App** using the described folder structure. The project will include foundational files with basic implementations to kickstart development.

Project Folder Structure

```
LearningApp/
├── android/
├── ios/
├── src/
│   ├── assets/
│   │   ├── fonts/
│   │   └── images/
│   ├── components/
│   │   └── Button/
│   │       ├── Button.tsx
│   │       ├── Button.styles.ts
│   │       └── index.ts
│   ├── config/
│   │   ├── env.ts
│   │   └── theme.ts
│   ├── constants/
│   │   └── api.ts
│   ├── hooks/
│   │   └── useFetch.ts
│   ├── modules/
│   │   └── Courses/
│   │       ├── components/
│   │       │   └── CourseCard.tsx
│   │       ├── screens/
│   │       │   └── CoursesScreen.tsx
│   │       ├── services/
│   │       │   └── coursesService.ts
│   │       └── styles/
│   │           └── Courses.styles.ts
│   ├── navigation/
│   │   ├── AppNavigator.tsx
│   │   └── index.ts
│   ├── redux/
│   │   ├── slices/
│   │   │   ├── coursesSlice.ts
│   │   │   └── index.ts
│   │   └── store.ts
│   ├── screens/
│   │   └── HomeScreen.tsx
│   ├── services/
│   │   └── apiService.ts
│   ├── utils/
│   │   └── validate.ts
│   └── types/
│       └── courseTypes.ts
├── .env
├── .gitignore
├── app.json
├── babel.config.js
├── index.js
├── metro.config.js
├── package.json
└── tsconfig.json
```

Basic File Implementations

1. **`src/assets/images/`**

Add placeholders for images.

2. **`src/assets/fonts/`**

Add custom fonts if needed (leave empty for now).

3. **src/components/Button/Button.tsx**

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

interface ButtonProps {
  title: string;
  onPress: () => void;
}

const Button: React.FC<ButtonProps> = ({ title, onPress }) =>
{
  return (
    <TouchableOpacity style={styles.button} onPress={onPress}>
      <Text style={styles.text}>{title}</Text>
    </TouchableOpacity>
  );
};

const styles = StyleSheet.create({
  button: {
    backgroundColor: '#007bff',
    padding: 10,
    borderRadius: 5,
    alignItems: 'center',
  },
  text: {
    color: '#fff',
    fontSize: 16,
  },
});

export default Button;
```

4. **`src/config/env.ts`**

```
export const API_BASE_URL = 'https://api.learningapp.com';
```

5. **`src/config/theme.ts`**

```
export const theme = {
  primaryColor: '#007bff',
  secondaryColor: '#6c757d',
  backgroundColor: '#f8f9fa',
  textColor: '#343a40',
};
```

---

6. **`src/constants/api.ts`**

```
export const API_ROUTES = {
  GET_COURSES: '/courses',
  GET_COURSE_DETAILS: (id: string) => `/courses/${id}`,
};
```

---

7. **`src/hooks/useFetch.ts`**

```
import { useState, useEffect } from 'react';
import axios from 'axios';

const useFetch = (url: string) => {
  const [data, setData] = useState<any>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await axios.get(url);
        setData(response.data);
      } catch (err) {
        setError('An error occurred while fetching data.');
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, [url]);

  return { data, loading, error };
};

export default useFetch;
```

---

8. **`src/modules/Courses/screens/CoursesScreen.tsx`**

```
import React from 'react';
import { View, Text, FlatList } from 'react-native';
import useFetch from '../../../hooks/useFetch';
import CourseCard from '../components/CourseCard';
import { API_BASE_URL } from '../../../config/env';
import { API_ROUTES } from '../../../constants/api';

const CoursesScreen: React.FC = () => {
  const { data: courses, loading, error } =
useFetch(`${API_BASE_URL}${API_ROUTES.GET_COURSES}`);

  if (loading) return <Text>Loading...</Text>;
  if (error) return <Text>{error}</Text>;

  return (
    <FlatList
      data={courses}
      keyExtractor={(item) => item.id.toString()}
      renderItem={({ item }) => <CourseCard course={item} />}
    />
  );
};

export default CoursesScreen;
```

---

9. **`src/modules/Courses/components/CourseCard.tsx`**

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

interface CourseCardProps {
  course: {
    id: string;
    title: string;
    description: string;
  };
}

const CourseCard: React.FC<CourseCardProps> = ({ course }) =>
{
  return (
    <View style={styles.card}>
      <Text style={styles.title}>{course.title}</Text>
      <Text
style={styles.description}>{course.description}</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  card: {
    backgroundColor: '#fff',
    marginBottom: 10,
    padding: 15,
    borderRadius: 5,
    shadowColor: '#000',
    shadowOpacity: 0.1,
    shadowRadius: 5,
    elevation: 3,
  },
  title: {
    fontSize: 18,
    fontWeight: 'bold',
    marginBottom: 5,
  },
  description: {
    fontSize: 14,
    color: '#6c757d',
  },
});

export default CourseCard;
```
