React Native Full Project Guide: Build a Complete App from Scratch

Subtitle: Combine all your React Native skills to create a fully functional mobile app.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

After learning React Native fundamentals, navigation, state management, APIs, forms, and deployment, it's time to **build a complete app**. This guide walks you through creating a **To-Do List App** that integrates all core concepts.

Step 1: Project Setup

1. Create a new React Native project:

```
npx react-native init TodoApp
```

or for Expo:

```
expo init TodoApp
```

2. Install necessary libraries:

```
npm install @react-navigation/native @react-navigation/native-stack react-native
npm install axios
```

3. Set up **NavigationContainer** and Stack Navigator.

Step 2: App Structure

Organize your project like this:

```
/TodoApp
  /components
    TodoItem.js
  /screens
```

```
  /screens
    HomeScreen.js
    AddTodoScreen.js
  /services
    api.js
  App.js
```

- **components/TodoItem.js** → displays each task.
- **screens/HomeScreen.js** → list of tasks.
- **screens/AddTodoScreen.js** → form to add new task.
- **services/api.js** → handles API calls if needed.

Step 3: Home Screen

- Display a list of tasks using **FlatList**.
- Use **state** to store tasks.
- Add **buttons** to mark as completed or delete.

Example:

```
const [tasks, setTasks] = useState([]);

<FlatList
  data={tasks}
  keyExtractor={item => item.id.toString()}
  renderItem={({ item }) => <TodoItem task={item} onDelete={deleteTask} />}
/>
```

Step 4: Add Todo Screen

- Use **TextInput** for task title.
- Validate input before submission.
- Pass new task back to HomeScreen via **navigation params or Context API**.

Example:

```
<Button
  title="Add Task"
```

```
  onPress={() => {
    if (taskTitle) {
      setTasks([...tasks, { id: Date.now(), title: taskTitle }]);
      navigation.goBack();
    } else alert('Enter a task');
  }}
/>
```

Step 5: Navigation Between Screens

- Use **Stack Navigator**: Home → Add Todo.
- Pass functions or use **Context API** to update state globally.

Example:

```
<Stack.Navigator>
  <Stack.Screen name="Home" component={HomeScreen} />
  <Stack.Screen name="AddTodo" component={AddTodoScreen} />
</Stack.Navigator>
```

Step 6: Optional Enhancements

- Add **API integration** to save tasks remotely.
- Implement **Redux** for state management in larger apps.
- Add **animations** for task addition/deletion using `Animated`.
- Implement **dark mode** toggle using Context API.

Step 7: Testing & Deployment

- Test the app on **real devices** for both iOS and Android.
- Optimize performance using **FlatList, React.memo, and proper state management**.
- Build and deploy following the **Deployment PDF guide**.

Building a complete React Native project consolidates your knowledge and demonstrates your skills. Completing this app prepares you for **portfolio projects, interviews, and real-world development**.

Visit **haas.dev** for more React Native guides, tutorials, and complete project examples.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app
