Week 14 - Building Practical React Native Applications

Introduction

Welcome to Week 14 of our React Native course! This week, we build upon the advanced concepts learned previously and apply them to create complete, functional applications. We'll focus on two practical projects: an Image Picker and Share application, and a Weather Forecast application. These projects demonstrate how to integrate with device features, fetch remote data, and create polished user interfaces.

Topics Covered in Week 14

- Creating complete, functional React Native applications
- Accessing device features using Expo libraries
- Fetching and displaying real-time data from external APIs
- Implementing permission handling for device resources
- Building reusable, modular components
- Managing application state across components
- Creating intuitive and responsive user interfaces
- Implementing navigation between screens
- Proper error handling and user feedback

Understanding the Week 14 Code

1. ImagePickerAndShare.js - Working with Device Media and Sharing

Concepts Covered:

- Requesting and handling device permissions
- Accessing the device's photo library
- Implementing native sharing functionality
- Conditional rendering based on application state
- Using Expo's ImagePicker and Sharing libraries

Code Breakdown:

- The app allows users to select images from their device's photo library
- Uses permission handling to ensure proper access to media

- Implements conditional rendering to show different UI based on whether an image is selected
- Provides sharing functionality to distribute the selected image
- Demonstrates proper error handling with user-friendly alerts

2. WeatherForecast.js - Fetching and Displaying External API Data

Concepts Covered:

- Using class-based components with state management
- Fetching data from external APIs (OpenWeatherMap)
- Processing and transforming API responses
- Handling user input with form submission
- Implementing complex layouts with multiple UI elements
- Creating responsive backgrounds with images

Code Breakdown:

- Users can enter a ZIP code to fetch weather data for that location
- The application makes API calls to OpenWeatherMap to get current weather data
- Weather information is displayed in a structured, readable format
- Demonstrates proper component organization and data flow
- Shows how to handle asynchronous operations in class components

3. Forecast.js - Building Reusable Display Components

Concepts Covered:

- Creating reusable, specialized components
- Passing data through props
- Structuring content for clear information display
- Using remote images with dynamic URLs
- Implementing consistent styling for information hierarchy

Code Breakdown:

- A specialized component that displays weather information
- Receives detailed weather data as props from the parent component
- Creates a structured display of weather conditions, temperature, and other details
- Demonstrates proper component organization and separation of concerns
- Shows how to build reusable UI components that can be styled consistently

4. open_weather_map.js - API Integration Module

Concepts Covered:

- Creating dedicated modules for API interactions
- Building URL parameters for API requests
- Handling API responses and transforming data
- Implementing proper error handling for network requests
- Date and time formatting for user-friendly display

Code Breakdown:

- Provides a reusable function to fetch weather data based on ZIP code
- Formats API URLs with proper query parameters and API keys
- Transforms raw API response data into a more application-friendly format
- Demonstrates best practices for separating API logic from UI components
- Shows how to handle and process date/time data from APIs

5. App.js - Navigation and Main Application Structure

Concepts Covered:

- Implementing navigation between different screens
- Creating an intuitive application flow
- Building a menu-based interface for application features
- Using TouchableOpacity for interactive UI elements
- Styling buttons and text for consistent UI

Code Breakdown:

- Main application component that provides navigation to different screens
- Uses buttons to allow users to navigate to different features
- Demonstrates how to organize a multi-screen application
- Shows proper implementation of navigation between components
- Creates a clean, user-friendly main menu interface

Key Learning Outcomes

By the end of this week, students will be able to:

- Create complete, functional React Native applications that solve real-world problems
- Implement device features such as photo library access and sharing
- Fetch and display data from external APIs with proper error handling
- Build reusable components that follow best practices for separation of concerns
- Create intuitive navigation systems between different application screens
- Implement permission handling for accessing device resources
- Design responsive and user-friendly interfaces for mobile applications
- Apply proper state management techniques across component hierarchies
- Transform and display complex data in user-friendly formats

- Handle asynchronous operations with proper loading states and error handling