Week 15 - Building Interactive Mobile Applications

Introduction

Welcome to Week 15 of our React Native course! This week, we build upon previous concepts and focus on creating interactive applications with state management and dynamic UI changes. We'll develop three engaging applications: a Coin Flip App with animations, a Dice Roller for generating random outcomes, and a Theme Toggle App demonstrating dynamic styling. We'll also explore counter functionality with a Counter App. These projects showcase fundamental React Native concepts with practical, interactive implementations.

Topics Covered in Week 15

- Managing application state with useState hooks
- Creating interactive UI elements with buttons and touch events
- Implementing countdown animations and timers
- Generating random outcomes for game-like applications
- Building dynamic themes with conditional styling
- Creating responsive layouts for different device sizes
- Implementing functional updates to state
- Structuring component hierarchies for clarity and reusability
- Applying consistent styling with StyleSheet

Understanding the Week 15 Code

1. CoinFlipApp.js - Animated Random Outcome Generator

Concepts Covered:

- Managing multiple state variables simultaneously
- Implementing countdown animations with setInterval
- Generating random boolean outcomes
- Using conditional rendering based on state
- Disabling UI elements during animations
- Loading and displaying remote images

Code Breakdown:

- The app simulates flipping a coin with a 3-second countdown animation
- Uses multiple useState hooks to track coin result, countdown status, and button state
- Implements a timed countdown using setInterval and clearInterval
- Conditionally renders UI elements based on the current countdown state
- Demonstrates proper state management for interactive applications
- Shows how to disable UI elements during animations to prevent multiple triggers

2. DiceRoller.js - Random Number Generation with Visual Feedback

Concepts Covered:

- Managing numeric state with useState
- Mapping numeric values to visual elements
- Implementing random number generation
- Creating responsive image displays
- Building intuitive user interfaces for games
- Structuring clear application layout

Code Breakdown:

- The app simulates rolling a standard 6-sided die with visual feedback
- Uses useState to track the current dice value
- Maps numeric values (1-6) to corresponding die face images
- Implements a random number generator with Math.random() and Math.floor()
- Demonstrates how to update state based on random outcomes
- Shows proper application structure with title, image, result text, and action button

3. ThemeToggle.js - Dynamic Theme Switching

Concepts Covered:

- Implementing light and dark themes
- Using boolean state for UI mode switching
- Applying conditional styling based on state
- Creating separate StyleSheet objects for different themes
- Combining base and theme-specific styles
- Dynamically updating UI text content based on state

Code Breakdown:

- The app toggles between light and dark display modes
- Uses a boolean state variable to track the current theme
- Demonstrates how to conditionally apply different StyleSheet objects
- Shows how to combine base styles with theme-specific styles
- Updates both styling and text content based on the current theme

- Implements a simple, clean UI with clear visual feedback for theme changes

4. CounterApp.js - Basic State Management

Concepts Covered:

- Managing numeric state with useState
- Implementing functional state updates
- Creating multiple action buttons for different operations
- Building a consistent button layout with proper spacing
- Using explicit function declarations for event handlers
- Applying clean, readable styling

Code Breakdown:

- The app maintains a counter with increment, decrement, and reset functionality
- Uses useState to track the current count value
- Implements functional updates with previous state reference
- Demonstrates three different state operations (increase, decrease, reset)
- Shows proper button organization with consistent spacing
- Creates a clean, focused UI with clear visual hierarchy

Key Learning Outcomes

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

- Create interactive applications with state management using React hooks
- Implement animations and timers with JavaScript's setInterval
- Generate and use random outcomes for game-like applications
- Build applications with dynamic UI changes based on state
- Implement theme switching with conditional styling
- Create clean button layouts with proper spacing and organization
- Use functional updates for reliable state management
- Apply conditional rendering based on application state
- Structure applications with clear component organization
- Implement proper error handling and user feedback