250-Day React Native & Backend Development Roadmap

From Beginner to Advanced Level (1 Hour Daily Per Task)

Phase 1: React Native Fundamentals Deep Dive (Days 1-40)

Day 1: React Native Architecture & Core Concepts

- **Understanding the Bridge Architecture**: Study how JavaScript communicates with native modules
- **Thread Model**: Learn about JS thread, UI thread, and native modules thread
- **Virtual DOM vs Native Components**: Deep dive into reconciliation process
- **Practical Exercise**: Create a diagram mapping the React Native architecture

Day 2: JSX & Component Rendering Mechanism

- **JSX Transformation**: Understand how JSX converts to React.createElement calls
- **Rendering Pipeline**: Study the component lifecycle in detail
- **Fiber Architecture**: Learn about React's reconciliation algorithm
- **Practical Exercise**: Write components both in JSX and pure JavaScript to compare

Day 3: Props Deep Dive

- **Props Flow & Immutability**: Understand unidirectional data flow
- **Prop Types & Validation**: Learn PropTypes and TypeScript interfaces
- **Default Props & Destructuring**: Master advanced prop patterns
- **Practical Exercise**: Build a reusable Card component with various prop configurations

Day 4: State Management Fundamentals

- **useState Hook Deep Dive**: Understand state batching and updates
- **State vs Props**: Learn when to use each
- **State Update Patterns**: Functional updates and prev state
- **Practical Exercise**: Create a counter with multiple state variables demonstrating different update patterns

Day 5: Component Lifecycle - Class Components

- **Mounting Phase**: componentDidMount and initialization
- **Updating Phase**: shouldComponentUpdate, componentDidUpdate
- **Unmounting Phase**: componentWillUnmount and cleanup
- **Practical Exercise**: Build a data-fetching component using class component lifecycle

Day 6: Functional Components & Hooks Philosophy

- **Hooks Rules**: Deep understanding of why rules exist
- **Hook Execution Order**: How React tracks hooks
- **Mental Model**: Closures and hooks relationship
- **Practical Exercise**: Convert a class component to functional component analyzing behavior differences

Day 7: useEffect Hook Mastery - Part 1

- **Effect Execution Timing**: After render, layout effects
- **Dependency Array Mechanics**: How React compares dependencies
- **Cleanup Functions**: When and why they run
- **Practical Exercise**: Create components demonstrating different useEffect patterns

Day 8: useEffect Hook Mastery - Part 2

- **Common Pitfalls**: Infinite loops, stale closures
- **Data Fetching Patterns**: Proper async handling in useEffect
- **Effect Optimization**: Minimizing re-renders
- **Practical Exercise**: Build a search component with debounced API calls

Day 9: useContext Hook & Context API

- **Context Creation & Provider Pattern**: Understanding context tree
- **Context Re-render Behavior**: When consumers re-render
- **Multiple Contexts**: Composition patterns
- **Practical Exercise**: Create a theme system using Context API

Day 10: useReducer Hook Deep Dive

- **Reducer Pattern**: Pure functions and predictable state
- **Complex State Logic**: When to use useReducer over useState
- **Dispatch Mechanism**: Understanding action objects
- **Practical Exercise**: Build a todo app with useReducer managing all state

Day 11: useMemo Hook Optimization

- **Memoization Concept**: Caching expensive computations
- **When to Use**: Performance vs complexity tradeoff
- **Dependency Array**: Proper usage and pitfalls
- **Practical Exercise**: Optimize a list filtering component with useMemo

Day 12: useCallback Hook Optimization

- **Function Identity**: Why functions need memoization
- **Preventing Re-renders**: Child component optimization
- **Dependencies Management**: Proper callback dependencies
- **Practical Exercise**: Create a parent-child component demonstrating useCallback benefits

Day 13: useRef Hook & Refs Deep Dive

- **Ref Object Persistence**: Across renders without re-rendering
- **DOM Access**: Focusing inputs, measuring elements
- **Previous Value Storage**: Implementing usePrevious hook
- **Practical Exercise**: Build a custom input component with focus management

Day 14: Custom Hooks Creation - Part 1

- **Hook Composition**: Building hooks from primitive hooks
- **Reusability Patterns**: Extracting common logic
- **Naming Conventions**: Best practices
- **Practical Exercise**: Create useToggle, useLocalStorage custom hooks

Day 15: Custom Hooks Creation - Part 2

- **Advanced Patterns**: Hooks with complex state logic
- **Hook Testing**: Strategies for testing custom hooks
- **Performance Considerations**: Optimizing custom hooks
- **Practical Exercise**: Build useAsync, useFetch, and useDebounce hooks

Day 16: Core Components - View & Text

- **View Component**: Flexbox container properties deep dive
- **Text Component**: Styling, nesting, and typography
- **Accessibility Props**: Supporting screen readers
- **Practical Exercise**: Create a responsive card layout using only View and Text

Day 17: Core Components - Image & ImageBackground

- **Image Loading**: Understanding the loading lifecycle
- **Image Caching**: Strategies and limitations
- **Remote vs Local Images**: Performance implications
- **Practical Exercise**: Build an image gallery with loading states and error handling

Day 18: Core Components - ScrollView

- **ScrollView Internals**: How it differs from web scrolling
- **Performance Considerations**: When not to use ScrollView
- **Scroll Events**: Handling onScroll and scroll positions
- **Practical Exercise**: Create a parallax scrolling effect

Day 19: Core Components - FlatList Fundamentals

- **VirtualizedList Foundation**: Understanding virtualization
- **RenderItem Function**: Optimization patterns
- **Key Extraction**: Proper key strategies
- **Practical Exercise**: Build a contacts list with FlatList

Day 20: FlatList Advanced Patterns

- **Performance Optimization**: windowSize, maxToRenderPerBatch
- **Item Separators**: Custom separators and headers
- **Pull to Refresh**: Implementing refresh functionality
- **Practical Exercise**: Create an infinite scroll feed with pull-to-refresh

Day 21: SectionList Mastery

- **Section Structure**: Headers, data organization
- **Sticky Headers**: Implementation and styling
- **Performance**: Compared to FlatList
- **Practical Exercise**: Build a contacts app with alphabetical sections

Day 22: TextInput Deep Dive

- **Controlled vs Uncontrolled**: State management strategies
- **Input Validation**: Real-time validation patterns
- **Keyboard Management**: Types, return key handling
- **Practical Exercise**: Create a form with multiple input types and validation

Day 23: Touchable Components

- **TouchableOpacity vs TouchableHighlight**: Differences and use cases
- **Pressable Component**: Modern touch handling
- **Touch Events**: onPressIn, onPressOut, onLongPress
- **Practical Exercise**: Build custom button components with different touch feedbacks

Day 24: Modal Component

- **Modal Types**: Fullscreen, slide, fade animations
- **Modal Presentation**: Understanding modal stack
- **Accessibility**: Proper modal implementation
- **Practical Exercise**: Create a reusable modal system with different animation types

Day 25: ActivityIndicator & Loading States

- **Loading Patterns**: Spinners, skeletons, progress bars
- **User Experience**: When to show loading indicators
- **Custom Loading Components**: Creating branded loaders
- **Practical Exercise**: Build a comprehensive loading state management system

Day 26: Switch, Button & Basic Form Controls

- **Switch Component**: Platform differences
- **Button Component**: Limitations and customization
- **Form State Management**: Handling multiple inputs
- **Practical Exercise**: Create a settings screen with various form controls

Day 27: Platform-Specific Code

- **Platform Module**: Detecting and adapting to platforms
- **Platform-Specific Files**: .ios.js and .android.js
- **Conditional Rendering**: Based on platform
- **Practical Exercise**: Build a component that renders differently on iOS and Android

Day 28: Dimensions & Screen Metrics

- **Dimensions API**: Getting screen dimensions
- **Responsive Design**: Handling different screen sizes
- **Orientation Changes**: Listening and adapting
- **Practical Exercise**: Create a responsive grid layout that adapts to screen size

Day 29: KeyboardAvoidingView & Keyboard Management

- **KeyboardAvoidingView Props**: Behavior modes
- **Keyboard Events**: Listening to keyboard show/hide
- **Input Scrolling**: Ensuring inputs are visible
- **Practical Exercise**: Build a chat interface with proper keyboard handling

Day 30: Alert & Toast Notifications

- **Alert API**: Platform-specific behaviors
- **Custom Toast Implementation**: Creating toast notifications
- **Notification Patterns**: User feedback strategies
- **Practical Exercise**: Create a reusable notification system

Day 31: Linking & Deep Linking Basics

- **Opening URLs**: Linking to external apps
- **Deep Link Handling**: Receiving and parsing deep links
- **Universal Links**: iOS and Android App Links
- **Practical Exercise**: Implement email, phone, and web URL launching

Day 32: AsyncStorage Fundamentals

- **Storage API**: setItem, getItem, removeItem
- **Data Serialization**: JSON storage patterns
- **Storage Limitations**: Size and performance
- **Practical Exercise**: Build a simple note-taking app with local persistence

Day 33: AsyncStorage Advanced Patterns

- **Error Handling**: Proper async/await and try-catch
- **Batch Operations**: multiGet, multiSet
- **Data Migration**: Versioning stored data
- **Practical Exercise**: Create a shopping cart with AsyncStorage persistence

Day 34: NetInfo - Network Connectivity

- **Network State Detection**: Online, offline, cellular, wifi
- **Connection Quality**: Network speed considerations
- **Listeners**: Reacting to connectivity changes
- **Practical Exercise**: Build offline-first functionality with sync

Day 35: Clipboard API

- **Copy and Paste**: Text clipboard operations
- **Image Clipboard**: Handling images (platform differences)
- **Use Cases**: When to use clipboard
- **Practical Exercise**: Create a code snippet manager with copy functionality

Day 36: Vibration & Haptic Feedback

- **Vibration Patterns**: Creating custom vibrations
- **Haptic Feedback**: iOS haptic types
- **User Experience**: When to use haptics
- **Practical Exercise**: Add haptic feedback to a game or interactive UI

Day 37: StatusBar Customization

- **StatusBar Styling**: Color, style, visibility
- **Platform Differences**: iOS vs Android StatusBar
- **Dynamic Updates**: Changing StatusBar based on screen
- **Practical Exercise**: Create a screen with dynamic StatusBar changes

Day 38: Safe Area Context

- **Notches and Insets**: Handling device-specific areas
- **SafeAreaView**: Built-in vs react-native-safe-area-context
- **Safe Area Insets**: Using useSafeAreaInsets
- **Practical Exercise**: Build layouts that properly handle safe areas

Day 39: Debugging Fundamentals

- **React DevTools**: Component inspection
- **Chrome DevTools**: JavaScript debugging
- **Console Logging**: Effective debugging strategies
- **Practical Exercise**: Debug a complex component with multiple issues

Day 40: Performance Monitoring Basics

- **Performance Metrics**: FPS, memory usage
- **Identifying Bottlenecks**: Using performance monitor
- **Profiling**: React DevTools profiler
- **Practical Exercise**: Profile and optimize a slow component

---

Phase 2: Styling & Layouts Mastery (Days 41-60)

Day 41: Flexbox Deep Dive - Part 1

- **Flex Container**: flexDirection, justifyContent
- **Main Axis vs Cross Axis**: Understanding flex layout
- **Flex Items**: flex property variations
- **Practical Exercise**: Recreate common UI layouts using flexbox

Day 42: Flexbox Deep Dive - Part 2

- **alignItems vs alignSelf**: Individual item alignment
- **flexWrap**: Wrapping flex items
- **flexGrow, flexShrink, flexBasis**: Detailed understanding
- **Practical Exercise**: Build a responsive dashboard layout

Day 43: Position Property

- **Relative vs Absolute**: Understanding positioning context
- **Z-Index**: Stacking context in React Native
- **Position Strategies**: When to use each type
- **Practical Exercise**: Create overlapping UI elements and tooltips

Day 44: Dimensions & Percentages

- **Width and Height**: Fixed vs percentage
- **Aspect Ratio**: Maintaining proportions
- **Min/Max Dimensions**: Constraints
- **Practical Exercise**: Build a photo grid with consistent aspect ratios

Day 45: Margins & Padding Mastery

- **Box Model**: Understanding spacing
- **Negative Margins**: Advanced positioning
- **Shorthand Properties**: Efficient styling
- **Practical Exercise**: Create pixel-perfect spacing systems

Day 46: Borders & Border Radius

- **Border Properties**: width, color, style
- **Border Radius**: Creating rounded corners

- **Platform Differences**: Border rendering
- **Practical Exercise**: Design various card and button styles

Day 47: Background Colors & Opacity

- **Color Formats**: hex, rgb, rgba
- **Opacity vs Alpha**: Differences and use cases
- **Transparent Overlays**: Creating modal backgrounds
- **Practical Exercise**: Build a color picker component

Day 48: Shadows & Elevation

- **iOS Shadows**: shadowColor, shadowOffset, shadowRadius
- **Android Elevation**: elevation property
- **Cross-Platform Shadows**: Consistent styling
- **Practical Exercise**: Create a card component with platform-appropriate shadows

Day 49: Transform Property

- **Translate**: Moving elements
- **Scale**: Sizing transformations
- **Rotate**: Rotation transformations
- **Practical Exercise**: Build animated loading indicators using transforms

Day 50: StyleSheet API Deep Dive

- **StyleSheet.create**: Performance benefits
- **Style Composition**: Combining styles
- **Conditional Styles**: Dynamic styling patterns
- **Practical Exercise**: Create a theme-based styling system

Day 51: Absolute & Relative Units

- **Pixel Density**: Understanding dp/pt
- **PixelRatio API**: Adapting to screen density
- **Responsive Units**: Creating adaptive layouts
- **Practical Exercise**: Build truly responsive components

Day 52: Typography & Text Styling

- **Font Properties**: fontFamily, fontSize, fontWeight
- **Text Alignment**: align, justify
- **Line Height & Letter Spacing**: Fine-tuning typography
- **Practical Exercise**: Create a typography system with hierarchy

Day 53: Custom Fonts Integration

- **Font Files**: Adding custom fonts to project
- **Platform-Specific Setup**: iOS and Android configuration
- **Font Loading**: Async font loading patterns
- **Practical Exercise**: Integrate Google Fonts into your app

Day 54: Styled Components Pattern

- **Component-Based Styling**: Encapsulation
- **Props-Based Styles**: Dynamic styling
- **Theme Integration**: Theming patterns

- **Practical Exercise**: Refactor components to use styled components pattern

Day 55: Design Systems & Tokens

- **Design Tokens**: Colors, spacing, typography
- **Component Library**: Building reusable components
- **Documentation**: Commenting and organizing
- **Practical Exercise**: Create a mini design system

Day 56: Responsive Design Patterns - Part 1

- **Breakpoints**: Defining responsive breakpoints
- **useWindowDimensions Hook**: Reactive dimensions
- **Responsive Layouts**: Grid and list adaptations
- **Practical Exercise**: Build a responsive navigation system

Day 57: Responsive Design Patterns - Part 2

- **Orientation Handling**: Portrait vs landscape
- **Tablet vs Phone Layouts**: Adaptive UIs
- **Aspect Ratio Strategies**: Maintaining visual balance
- **Practical Exercise**: Create a universal app layout (phone & tablet)

Day 58: Dark Mode Implementation

- **Color Schemes**: Defining light and dark themes
- **useColorScheme Hook**: Detecting system theme
- **Theme Context**: Managing app-wide theme
- **Practical Exercise**: Implement full dark mode support

Day 59: Accessibility Styling

- **Color Contrast**: Meeting WCAG standards
- **Touch Target Sizes**: Minimum tap areas
- **Focus Indicators**: Visual feedback
- **Practical Exercise**: Audit and improve component accessibility

Day 60: Advanced Layout Techniques

- **Sticky Headers**: Implementation strategies
- **Parallax Effects**: Scroll-based transformations
- **Masonry Layouts**: Pinterest-style grids
- **Practical Exercise**: Build a complex photo browsing interface

---

Phase 3: Navigation & Routing (Days 61-75)

Day 61: React Navigation Fundamentals

- **Navigation Library**: Why React Navigation
- **Navigation Container**: Setting up navigation
- **Navigator Types**: Stack, Tab, Drawer overview
- **Practical Exercise**: Set up basic navigation structure

Day 62: Stack Navigator Deep Dive - Part 1

- **Screen Configuration**: Options and params
- **Navigation Prop**: navigate, goBack, push
- **Route Prop**: Accessing params
- **Practical Exercise**: Build a multi-screen flow

Day 63: Stack Navigator Deep Dive - Part 2

- **Header Customization**: Styling headers
- **Header Buttons**: Adding action buttons
- **Modal Presentation**: Modal vs card style
- **Practical Exercise**: Create custom headers with context-aware actions

Day 64: Stack Navigator Advanced

- **Nested Navigators**: Composition patterns
- **Screen Options**: Dynamic options
- **Navigation Lifecycle**: Focus and blur events
- **Practical Exercise**: Implement nested navigation with shared headers

Day 65: Bottom Tab Navigator

- **Tab Configuration**: Icons, labels, badges
- **Tab Bar Customization**: Custom tab bars
- **Tab Navigation**: Jumping between tabs
- **Practical Exercise**: Build a social media-style tab navigation

Day 66: Drawer Navigator

- **Drawer Configuration**: Drawer content customization
- **Drawer Actions**: Opening and closing programmatically
- **Nested Drawer Navigators**: Complex navigation patterns
- **Practical Exercise**: Create an app with side menu navigation

Day 67: Navigation Parameters

- **Passing Params**: Between screens
- **Type Safety**: TypeScript param types
- **Updating Params**: setParams usage
- **Practical Exercise**: Build a detail-edit screen flow with params

Day 68: Navigation State Management

- **Navigation State**: Understanding state structure
- **State Persistence**: Saving and restoring navigation
- **Deep Linking**: URL-based navigation
- **Practical Exercise**: Implement navigation state persistence

Day 69: Deep Linking Advanced

- **Link Configuration**: Mapping URLs to screens
- **Dynamic Links**: Handling parameters in URLs
- **Universal Links**: iOS and Android setup
- **Practical Exercise**: Implement complete deep linking system

Day 70: Navigation Guards & Auth Flow

- **Conditional Navigation**: Auth-based routing
- **Navigation Guards**: Protecting routes
- **Auth Context**: Managing authentication state
- **Practical Exercise**: Build login/signup/protected routes flow

Day 71: Navigation Transitions

- **Transition Animations**: Custom animations
- **Gesture Configuration**: Swipe gestures
- **Platform-Specific Transitions**: iOS vs Android
- **Practical Exercise**: Create custom screen transitions

Day 72: Header Configurations Advanced

- **Transparent Headers**: Overlay headers
- **Large Titles**: iOS-style large headers
- **Collapsible Headers**: Scroll-based header changes
- **Practical Exercise**: Build a screen with collapsing header

Day 73: Tab Bar Advanced Patterns

- **Custom Tab Bar**: Building from scratch
- **Animated Tab Bar**: Smooth transitions
- **Badge System**: Notification badges
- **Practical Exercise**: Create an animated custom tab bar

Day 74: Navigation Performance

- **Screen Optimization**: Lazy loading screens
- **Navigation Events**: Handling focus efficiently
- **Memory Management**: Unmounting strategies
- **Practical Exercise**: Optimize navigation in large app

Day 75: Navigation Testing

- **Testing Navigation Flows**: Jest and testing library
- **Mocking Navigation**: Test utilities
- **Integration Tests**: Multi-screen tests
- **Practical Exercise**: Write comprehensive navigation tests

---

Phase 4: Advanced State Management (Days 76-95)

Day 76: Global State Challenges

- **Prop Drilling Problem**: Understanding limitations
- **State Colocation**: When to lift state up
- **State Management Patterns**: Overview of solutions
- **Practical Exercise**: Analyze a complex app's state needs

Day 77: Context API Advanced Patterns

- **Multiple Contexts**: Separating concerns
- **Context Optimization**: Preventing unnecessary renders

- **Context Composition**: Combining contexts
- **Practical Exercise**: Build a multi-context application

Day 78: Redux Fundamentals - Part 1

- **Redux Principles**: Single source of truth, immutability
- **Store, Actions, Reducers**: Core concepts
- **Redux Data Flow**: Understanding the cycle
- **Practical Exercise**: Set up Redux in a React Native app

Day 79: Redux Fundamentals - Part 2

- **Combining Reducers**: Splitting state logic
- **Action Creators**: Organizing actions
- **Connecting Components**: useSelector, useDispatch
- **Practical Exercise**: Build a todo app with Redux

Day 80: Redux Toolkit Introduction

- **Why Redux Toolkit**: Simplifying Redux
- **configureStore**: Enhanced store setup
- **createSlice**: Simplified reducers
- **Practical Exercise**: Migrate Redux app to Redux Toolkit

Day 81: Redux Toolkit Advanced

- **createAsyncThunk**: Handling async operations
- **Entity Adapters**: Normalized state management
- **RTK Query Basics**: Data fetching abstraction
- **Practical Exercise**: Implement async data fetching with Redux Toolkit

Day 82: RTK Query Deep Dive

- **API Slice**: Defining endpoints
- **Caching Strategies**: Cache management
- **Optimistic Updates**: Immediate UI updates
- **Practical Exercise**: Build a full CRUD app with RTK Query

Day 83: Redux DevTools & Debugging

- **Time Travel Debugging**: Action replay
- **State Inspection**: Exploring state changes
- **Action Monitoring**: Tracking dispatched actions
- **Practical Exercise**: Debug complex Redux state issues

Day 84: Redux Middleware

- **Middleware Concept**: Intercepting actions
- **Redux Thunk**: Async action creators
- **Custom Middleware**: Creating your own
- **Practical Exercise**: Implement logging and analytics middleware

Day 85: Redux Best Practices

- **State Normalization**: Flat state structures
- **Selector Patterns**: Reselect library
- **Performance Optimization**: Memoization strategies

- **Practical Exercise**: Refactor Redux state for performance

Day 86: MobX Fundamentals

- **Observable State**: Reactive programming
- **Actions & Computed Values**: State management
- **Reactions**: Side effects in MobX
- **Practical Exercise**: Build an app using MobX

Day 87: MobX Advanced Patterns

- **Stores Organization**: Multiple stores
- **MobX with React**: observer HOC and hooks
- **Async Actions**: Flow syntax
- **Practical Exercise**: Create a complex MobX store structure

Day 88: Zustand - Lightweight State

- **Zustand Basics**: Simple state management
- **Store Creation**: Minimal boilerplate
- **Middleware**: Persist, immer integration
- **Practical Exercise**: Migrate an app to Zustand

Day 89: Jotai - Atomic State Management

- **Atoms Concept**: Bottom-up state
- **Derived Atoms**: Computed values
- **Async Atoms**: Handling promises
- **Practical Exercise**: Build modular state with Jotai

Day 90: Recoil State Management

- **Atoms and Selectors**: Recoil fundamentals
- **Async Selectors**: Data fetching
- **Atom Effects**: Side effects
- **Practical Exercise**: Create a feature-complete app with Recoil

Day 91: State Machine Patterns

- **Finite State Machines**: Modeling state transitions
- **XState Library**: State machine implementation
- **State Charts**: Visual state management
- **Practical Exercise**: Model complex UI states with XState

Day 92: State Persistence Strategies

- **Redux Persist**: Persisting Redux state
- **Storage Adapters**: AsyncStorage integration
- **Rehydration**: Loading saved state
- **Practical Exercise**: Implement full state persistence

Day 93: Offline State Synchronization

- **Optimistic Updates**: Immediate feedback
- **Conflict Resolution**: Handling sync conflicts
- **Queue Management**: Action queuing when offline
- **Practical Exercise**: Build offline-first state management

Day 94: State Management Performance

- **Render Optimization**: Minimizing re-renders
- **Selector Performance**: Memoization strategies
- **State Splitting**: Avoiding monolithic state
- **Practical Exercise**: Profile and optimize state-heavy app

Day 95: Choosing State Management

- **Decision Framework**: When to use what
- **Comparing Solutions**: Redux vs MobX vs Context
- **Migration Strategies**: Switching state libraries
- **Practical Exercise**: Evaluate state management for different app types

---

Phase 5: API Integration & Data Fetching (Days 96-110)

Day 96: HTTP Basics & REST APIs

- **HTTP Methods**: GET, POST, PUT, DELETE
- **Status Codes**: Understanding responses
- **Headers & Body**: Request structure
- **Practical Exercise**: Manually construct API requests

Day 97: Fetch API Deep Dive

- **Fetch Syntax**: Making requests
- **Promise Handling**: then/catch vs async/await
- **Error Handling**: Network and HTTP errors
- **Practical Exercise**: Build a fetch wrapper with error handling

Day 98: Axios Library

- **Axios vs Fetch**: Differences and benefits
- **Interceptors**: Request/response manipulation
- **Axios Instance**: Configuring defaults
- **Practical Exercise**: Set up Axios with interceptors

Day 99: API Authentication - Part 1

- **Token-Based Auth**: JWT fundamentals
- **Storing Tokens**: Secure storage strategies
- **Token Refresh**: Handling expiration
- **Practical Exercise**: Implement JWT authentication flow

Day 100: API Authentication - Part 2

- **OAuth 2.0**: Social login flows
- **Secure Token Storage**: Keychain/Keystore
- **Auth Context**: Managing auth state
- **Practical Exercise**: Implement OAuth with Google/Facebook

Day 101: API Error Handling Strategies

- **Error Types**: Network, HTTP, parsing errors
- **Retry Logic**: Exponential backoff

- **User Feedback**: Error messages and UI
- **Practical Exercise**: Build robust error handling system

Day 102: Loading States & UX

- **Loading Patterns**: Spinners, skeletons, progress
- **Optimistic Updates**: Immediate feedback
- **Error States**: Retry mechanisms
- **Practical Exercise**: Create comprehensive loading state management

Day 103: Pagination Implementation

- **Offset-Based Pagination**: Page numbers
- **Cursor-Based Pagination**: Scalable pagination
- **Infinite Scroll**: FlatList integration
- **Practical Exercise**: Implement infinite scroll with FlatList

Day 104: Search & Filtering

- **Search Implementation**: Real-time search
- **Debouncing**: Optimizing API calls
- **Server-Side Filtering**: Query parameters
- **Practical Exercise**: Build a search feature with autocomplete

Day 105: Data Caching Strategies

- **Cache Invalidation**: When to refetch
- **Memory Cache**: In-app caching
- **Persistent Cache**: Storage-based caching
- **Practical Exercise**: Implement multi-level caching

Day 106: React Query/TanStack Query - Part 1

- **Query Basics**: useQuery hook
- **Query Keys**: Cache key strategies
- **Automatic Refetching**: staleTime, cacheTime
- **Practical Exercise**: Set up React Query in app

Day 107: React Query - Part 2

- **Mutations**: useMutation hook
- **Optimistic Updates**: Instant UI feedback
- **Query Invalidation**: Cache management
- **Practical Exercise**: Build CRUD operations with React Query

Day 108: React Query Advanced

- **Infinite Queries**: useInfiniteQuery
- **Prefetching**: Anticipating data needs
- **Dependent Queries**: Sequential fetching
- **Practical Exercise**: Implement complex data fetching patterns

Day 109: GraphQL Basics

- **GraphQL Concepts**: Queries, mutations, schemas
- **GraphQL vs REST**: Comparison
- **GraphQL Clients**: Apollo, urql overview

- **Practical Exercise**: Write GraphQL queries and mutations

Day 110: Apollo Client Integration

- **Apollo Setup**: Client configuration
- **useQuery & useMutation**: Apollo hooks
- **Cache Management**: Apollo cache
- **Practical Exercise**: Build a GraphQL-powered app with Apollo

---

Phase 6: Forms & Validation (Days 111-120)

Day 111: Controlled Components Deep Dive

- **Controlled Inputs**: State-backed inputs
- **Input Value Management**: onChange patterns
- **Performance**: Optimizing controlled inputs
- **Practical Exercise**: Build fully controlled form

Day 112: Uncontrolled Components & Refs

- **Ref-Based Forms**: When to use uncontrolled
- **Default Values**: Initial form state
- **Imperative APIs**: Programmatic form control
- **Practical Exercise**: Create uncontrolled form with refs

Day 113: Form State Management

- **Form State Patterns**: Single vs multiple states
- **State Structure**: Nested form data
- **Form Reset**: Clearing and resetting forms
- **Practical Exercise**: Build multi-step form with state

Day 114: React Hook Form - Part 1

- **Library Introduction**: Why React Hook Form
- **useForm Hook**: Form setup and registration
- **Validation Rules**: Built-in validation
- **Practical Exercise**: Convert manual form to React Hook Form

Day 115: React Hook Form - Part 2

- **Controller Component**: Integrating custom inputs
- **Watch & GetValues**: Monitoring form state
- **Form Submission**: handleSubmit patterns
- **Practical Exercise**: Build complex form with custom components

Day 116: React Hook Form Advanced

- **Field Arrays**: Dynamic form fields
- **Form Context**: useFormContext
- **Error Handling**: Custom error display
- **Practical Exercise**: Create dynamic form with add/remove fields

Day 117: Validation Strategies

- **Client-Side Validation**: Immediate feedback
- **Validation Patterns**: Email, phone, etc.
- **Custom Validators**: Business logic validation
- **Practical Exercise**: Implement comprehensive validation rules

Day 118: Yup Schema Validation

- **Schema Definition**: Yup validation schemas
- **Integration**: Yup with React Hook Form
- **Custom Validators**: Extending Yup
- **Practical Exercise**: Define complex validation schemas

Day 119: Zod Schema Validation

- **Zod Basics**: TypeScript-first validation
- **Schema Composition**: Reusable schemas
- **Zod vs Yup**: Comparison and use cases
- **Practical Exercise**: Migrate validation to Zod

Day 120: Form Accessibility

- **Label Association**: Proper label usage
- **Error Announcement**: Screen reader support
- **Keyboard Navigation**: Focus management
- **Practical Exercise**: Make forms fully accessible

Phase 7: Animations & Gestures (Days 121-140)

Day 121: Animated API Fundamentals

- **Animated Values**: Animated.Value basics
- **Animation Types**: timing, spring, decay
- **Animation Composition**: sequence, parallel
- **Practical Exercise**: Create basic animations

Day 122: Animated API - Interpolation

- **Interpolation Concept**: Mapping value ranges
- **Input/Output Ranges**: Creating animations
- **Extrapolation**: Beyond defined ranges
- **Practical Exercise**: Build color and size interpolations

Day 123: Animated API - Combining Animations

- **Sequence**: One after another
- **Parallel**: Multiple animations together
- **Stagger**: Delayed parallel animations
- **Practical Exercise**: Create complex animation sequences

Day 124: Animated API - useNativeDriver

- **Native Driver**: Performance benefits
- **Supported Properties**: Limitations

- **When to Use**: Native vs JS-driven animations
- **Practical Exercise**: Convert animations to use native driver

Day 125: LayoutAnimation API

- **LayoutAnimation Basics**: Automatic layout animations
- **Presets**: easeInEaseOut, spring, linear
- **Custom Configurations**: Creating custom transitions
- **Practical Exercise**: Animate list item additions/removals

Day 126: React Native Reanimated - Setup

- **Library Introduction**: Why Reanimated
- **Worklets**: Running on UI thread
- **Shared Values**: useSharedValue hook
- **Practical Exercise**: Set up Reanimated and create basic animation

Day 127: Reanimated - Animation Functions

- **withTiming**: Smooth transitions
- **withSpring**: Physics-based animations
- **withDecay**: Momentum-based animations
- **Practical Exercise**: Create various animation types

Day 128: Reanimated - Animated Styles

- **useAnimatedStyle**: Reactive styles
- **Worklet Context**: Understanding execution
- **Performance**: Why it's faster
- **Practical Exercise**: Build animated components with Reanimated

Day 129: Reanimated - Gestures Integration

- **Gesture Handler Library**: Installation and setup
- **Pan Gesture**: Drag interactions
- **Gesture State**: Tracking gesture lifecycle
- **Practical Exercise**: Create draggable components

Day 130: React Native Gesture Handler - Pan

- **PanGestureHandler**: Configuration
- **Translation Values**: X and Y movement
- **Velocity**: Gesture speed detection
- **Practical Exercise**: Build a swipeable card component

Day 131: Gesture Handler - Tap & Long Press

- **TapGestureHandler**: Single and multiple taps
- **LongPressGestureHandler**: Hold interactions
- **Gesture Combinations**: Multiple gestures
- **Practical Exercise**: Create interactive touch elements

Day 132: Gesture Handler - Pinch & Rotation

- **PinchGestureHandler**: Zoom interactions
- **RotationGestureHandler**: Rotation detection
- **Scale and Rotation Values**: Math behind gestures

- **Practical Exercise**: Build an image viewer with pinch-to-zoom

Day 133: Gesture Handler - Simultaneous Gestures

- **Simultaneous Recognition**: Multiple gestures at once
- **Gesture Priority**: Handling conflicts
- **Cross-Handler**: Combining gesture handlers
- **Practical Exercise**: Create a multi-gesture interactive canvas

Day 134: Reanimated - Complex Animations

- **withSequence**: Chaining animations
- **withDelay**: Delayed animations
- **withRepeat**: Looping animations
- **Practical Exercise**: Build loading animations and effects

Day 135: Reanimated - Interpolations Advanced

- **interpolate Function**: Value mapping
- **Extrapolation**: Clamp, extend, identity
- **Color Interpolation**: interpolateColor
- **Practical Exercise**: Create smooth color transitions

Day 136: Reanimated - Scroll Animations

- **useAnimatedScrollHandler**: Scroll tracking
- **Scroll-Driven Animations**: Parallax effects
- **Animated FlatList**: Scroll-based item animations
- **Practical Exercise**: Build animated scroll header

Day 137: Shared Element Transitions

- **react-native-shared-element**: Library setup
- **Shared Element Configuration**: Matching elements
- **Navigation Integration**: Transition between screens
- **Practical Exercise**: Create hero image transitions

Day 138: Lottie Animations

- **Lottie Files**: JSON-based animations
- **lottie-react-native**: Library integration
- **Animation Control**: Play, pause, speed
- **Practical Exercise**: Add professional animations to app

Day 139: Animation Performance

- **Performance Monitoring**: FPS tracking
- **Optimization Techniques**: Reducing jank
- **Native Driver Usage**: When and how
- **Practical Exercise**: Profile and optimize animations

Day 140: Micro-Interactions & UX

- **Button Feedback**: Press states
- **Transition Smoothness**: Perceived performance
- **Animation Timing**: UX best practices
- **Practical Exercise**: Polish UI with micro-animations

Phase 8: Native Modules & Advanced Features (Days 141-160)

Day 141: Camera Integration - Part 1

- **react-native-camera**: Library setup
- **Camera Permissions**: iOS and Android
- **Camera Types**: Front and back camera
- **Practical Exercise**: Build basic camera component

Day 142: Camera Integration - Part 2

- **Taking Photos**: Capturing images
- **Recording Videos**: Video capture
- **Camera Settings**: Flash, focus, exposure
- **Practical Exercise**: Create full-featured camera app

Day 143: Image Picker & Media Library

- **react-native-image-picker**: Selecting images
- **Multiple Selection**: Batch image selection
- **Image Cropping**: Integrated cropping
- **Practical Exercise**: Build image selection with preview

Day 144: Image Manipulation

- **react-native-image-resizer**: Resizing images
- **Image Compression**: Reducing file size
- **Image Format Conversion**: JPEG, PNG, WebP
- **Practical Exercise**: Optimize images before upload

Day 145: File System Access

- **react-native-fs**: File operations
- **Reading/Writing Files**: File manipulation
- **Directory Management**: Creating and deleting
- **Practical Exercise**: Build file manager functionality

Day 146: Document Picker

- **react-native-document-picker**: File selection
- **File Types**: PDF, Word, Excel support
- **File Metadata**: Getting file information
- **Practical Exercise**: Create document upload feature

Day 147: Geolocation - Part 1

- **@react-native-community/geolocation**: Setup
- **Location Permissions**: Requesting access
- **Getting Current Position**: One-time location
- **Practical Exercise**: Display user's current location

Day 148: Geolocation - Part 2

- **Watching Position**: Continuous tracking
- **Location Accuracy**: High vs low accuracy

- **Background Location**: Background tracking setup
- **Practical Exercise**: Build location tracking feature

Day 149: Maps Integration - Part 1

- **react-native-maps**: Map library setup
- **Map Types**: Standard, satellite, hybrid
- **Markers**: Adding and customizing markers
- **Practical Exercise**: Display map with custom markers

Day 150: Maps Integration - Part 2

- **Polylines and Polygons**: Drawing on maps
- **Map Regions**: Controlling viewport
- **Animated Map Movement**: Smooth transitions
- **Practical Exercise**: Build route display with directions

Day 151: Push Notifications - Setup

- **Firebase Cloud Messaging**: FCM setup
- **APNs Configuration**: iOS notification setup
- **react-native-push-notification**: Library integration
- **Practical Exercise**: Set up basic push notifications

Day 152: Push Notifications - Advanced

- **Notification Channels**: Android channels
- **Notification Actions**: Interactive notifications
- **Badge Management**: App icon badges
- **Practical Exercise**: Implement rich notifications

Day 153: Local Notifications

- **Scheduling Notifications**: Time-based triggers
- **Repeating Notifications**: Daily, weekly scheduling
- **Notification Sounds**: Custom sounds
- **Practical Exercise**: Build reminder feature

Day 154: In-App Purchases - Setup

- **react-native-iap**: Library configuration
- **Store Configuration**: iOS and Android setup
- **Product Listing**: Fetching available products
- **Practical Exercise**: Display purchasable products

Day 155: In-App Purchases - Implementation

- **Purchase Flow**: Initiating purchases
- **Receipt Validation**: Verifying purchases
- **Subscription Management**: Handling subscriptions
- **Practical Exercise**: Complete IAP implementation

Day 156: Biometric Authentication

- **react-native-biometrics**: Fingerprint and Face ID
- **Biometric Availability**: Checking device support
- **Authentication Flow**: Secure authentication

- **Practical Exercise**: Add biometric login

Day 157: Secure Storage

- **react-native-keychain**: Secure credential storage
- **Encryption**: Data protection
- **Platform-Specific Security**: Keychain/Keystore
- **Practical Exercise**: Store sensitive data securely

Day 158: Share & Social Sharing

- **Share API**: Native share sheet
- **react-native-share**: Advanced sharing
- **Deep Links**: Sharing app content
- **Practical Exercise**: Implement share functionality

Day 159: Bluetooth Integration

- **react-native-ble-manager**: BLE setup
- **Device Scanning**: Finding Bluetooth devices
- **Connecting and Communication**: Data transfer
- **Practical Exercise**: Build Bluetooth device connector

Day 160: Native Module Creation

- **Creating Native Modules**: iOS and Android
- **Bridging Native Code**: Exposing to JavaScript
- **Native Events**: Sending events to JS
- **Practical Exercise**: Create custom native module

Phase 9: Testing & Quality Assurance (Days 161-175)

Day 161: Testing Fundamentals

- **Testing Types**: Unit, integration, E2E
- **Testing Philosophy**: What and how to test
- **Test Coverage**: Meaningful metrics
- **Practical Exercise**: Write testing strategy document

Day 162: Jest Setup & Configuration

- **Jest Basics**: Test runner configuration
- **Test Structure**: Describe, it, test blocks
- **Assertions**: Expect matchers
- **Practical Exercise**: Set up Jest for React Native

Day 163: Component Testing - Part 1

- **React Native Testing Library**: Setup
- **Rendering Components**: render function
- **Querying Elements**: getBy, queryBy, findBy
- **Practical Exercise**: Write basic component tests

Day 164: Component Testing - Part 2

- **User Interactions**: fireEvent and userEvent
- **Async Testing**: waitFor and findBy
- **Component Props**: Testing different props
- **Practical Exercise**: Test interactive components

Day 165: Testing Hooks

- **@testing-library/react-hooks**: Hook testing
- **Testing Custom Hooks**: Isolated testing
- **Async Hooks**: Testing useEffect patterns
- **Practical Exercise**: Test custom hooks thoroughly

Day 166: Mocking Fundamentals

- **Jest Mocks**: Mock functions and modules
- **Mocking Dependencies**: External libraries
- **Spy Functions**: Tracking function calls
- **Practical Exercise**: Mock API calls in tests

Day 167: Testing Navigation

- **Mocking Navigation**: Testing library setup
- **Navigation Testing**: Screen transitions
- **Params Testing**: Route params verification
- **Practical Exercise**: Test navigation flows

Day 168: Testing Redux/State Management

- **Testing Reducers**: Pure function tests
- **Testing Actions**: Action creator tests
- **Connected Components**: Testing with store
- **Practical Exercise**: Comprehensive Redux tests

Day 169: Snapshot Testing

- **Snapshot Concept**: Component snapshots
- **Snapshot Updates**: When to update
- **Snapshot Best Practices**: What to snapshot
- **Practical Exercise**: Create meaningful snapshots

Day 170: Integration Testing

- **Integration Test Strategy**: Testing flows
- **Multi-Component Tests**: User journeys
- **API Integration**: Testing with backends
- **Practical Exercise**: Write end-to-end user flows

Day 171: E2E Testing with Detox - Setup

- **Detox Installation**: iOS and Android setup
- **Test Configuration**: Detox configuration
- **Build Configuration**: Test builds
- **Practical Exercise**: Set up Detox environment

Day 172: E2E Testing with Detox - Writing Tests

- **Detox Matchers**: Finding elements
- **Detox Actions**: Tapping, typing, scrolling
- **Detox Expectations**: Assertions
- **Practical Exercise**: Write complete E2E tests

Day 173: Test Coverage & Reporting

- **Coverage Reports**: Istanbul integration
- **Coverage Thresholds**: Enforcing minimums
- **Coverage Analysis**: Identifying gaps
- **Practical Exercise**: Generate and analyze coverage

Day 174: Continuous Integration

- **CI/CD Pipelines**: GitHub Actions, CircleCI
- **Automated Testing**: Running tests on push
- **Test Reports**: Visualizing results
- **Practical Exercise**: Set up CI pipeline

Day 175: Performance Testing

- **Performance Metrics**: Measuring app performance
- **Memory Leaks**: Detecting and fixing
- **Render Performance**: Testing component speed
- **Practical Exercise**: Create performance benchmarks

Phase 10: Backend Development - Node.js Fundamentals (Days 176-195)

Day 176: Node.js Architecture

- **Event Loop**: Understanding async execution
- **Non-Blocking I/O**: How Node handles operations
- **Single-Threaded Model**: Benefits and limitations
- **Practical Exercise**: Visualize event loop behavior

Day 177: NPM & Package Management

- **NPM Basics**: Installing packages
- **package.json**: Dependency management
- **Versioning**: Semantic versioning
- **Practical Exercise**: Create and manage Node project

Day 178: Node.js Modules System

- **CommonJS**: require and module.exports
- **ES Modules**: import and export
- **Module Resolution**: How Node finds modules
- **Practical Exercise**: Create custom modules

Day 179: File System Operations

- **fs Module**: Reading and writing files
- **Async vs Sync**: File operations

- **Streams**: Efficient file handling
- **Practical Exercise**: Build file manipulation utilities

Day 180: HTTP Module & Web Servers

- **Creating HTTP Server**: http.createServer
- **Request and Response**: Handling HTTP
- **Routing**: Basic URL routing
- **Practical Exercise**: Build basic HTTP server

Day 181: Express.js Fundamentals

- **Express Setup**: Creating Express app
- **Middleware Concept**: Request pipeline
- **Routing**: Defining routes
- **Practical Exercise**: Build REST API structure

Day 182: Express Middleware Deep Dive

- **Application Middleware**: App-level middleware
- **Router Middleware**: Route-specific middleware
- **Error Handling Middleware**: Catching errors
- **Practical Exercise**: Create custom middleware chain

Day 183: Express Routing Advanced

- **Route Parameters**: Dynamic routing
- **Query Parameters**: Handling query strings
- **Route Handlers**: Multiple handlers
- **Practical Exercise**: Build complex routing system

Day 184: Request & Response Handling

- **Request Object**: Accessing request data
- **Response Methods**: Sending responses
- **Status Codes**: Proper HTTP status usage
- **Practical Exercise**: Handle various request types

Day 185: Express Router & Modular Routes

- **Express Router**: Modular routing
- **Route Organization**: File structure
- **Nested Routers**: Router composition
- **Practical Exercise**: Organize API into modules

Day 186: REST API Design Principles

- **RESTful Conventions**: HTTP method usage
- **Resource Naming**: URL structure
- **API Versioning**: Version strategies
- **Practical Exercise**: Design RESTful API structure

Day 187: Request Validation

- **Input Validation**: Validating request data
- **Joi/Yup Validation**: Schema validation
- **Sanitization**: Cleaning user input

- **Practical Exercise**: Implement comprehensive validation

Day 188: Error Handling in Express

- **Error Middleware**: Centralized error handling
- **Custom Error Classes**: Typed errors
- **Error Responses**: Consistent error format
- **Practical Exercise**: Build error handling system

Day 189: MongoDB Basics

- **NoSQL Concepts**: Document databases
- **MongoDB Installation**: Local setup
- **MongoDB Compass**: Visual database tool
- **Practical Exercise**: Set up MongoDB locally

Day 190: MongoDB CRUD Operations

- **Inserting Documents**: Create operations
- **Querying Documents**: Read operations
- **Updating Documents**: Update operations
- **Deleting Documents**: Delete operations
- **Practical Exercise**: Perform all CRUD operations

Day 191: Mongoose ODM - Part 1

- **Mongoose Setup**: Connecting to MongoDB
- **Schema Definition**: Defining data structure
- **Models**: Creating data models
- **Practical Exercise**: Create schemas and models

Day 192: Mongoose ODM - Part 2

- **Validators**: Built-in and custom validation
- **Middleware/Hooks**: Pre and post hooks
- **Virtual Properties**: Computed properties
- **Practical Exercise**: Build complex schema with validations

Day 193: Mongoose Queries

- **Query Methods**: find, findOne, findById
- **Query Builders**: Chaining query methods
- **Population**: Referencing documents
- **Practical Exercise**: Write complex database queries

Day 194: Database Relationships

- **One-to-Many**: Implementing relationships
- **Many-to-Many**: Complex relationships
- **Embedded vs Referenced**: When to use each
- **Practical Exercise**: Model related data

Day 195: Indexing & Performance

- **Database Indexes**: Improving query speed
- **Compound Indexes**: Multiple field indexes
- **Index Strategies**: When to index

- **Practical Exercise**: Optimize database queries

Phase 11: Backend Development - Advanced Backend (Days 196-215)

Day 196: Authentication Fundamentals

- **Authentication vs Authorization**: Understanding difference
- **Password Hashing**: bcrypt implementation
- **Session Management**: Stateful authentication
- **Practical Exercise**: Implement basic auth

Day 197: JWT Authentication

- **JWT Structure**: Header, payload, signature
- **Token Generation**: Signing tokens
- **Token Verification**: Validating tokens
- **Practical Exercise**: Build JWT auth system

Day 198: Refresh Tokens

- **Refresh Token Strategy**: Long-lived tokens
- **Token Rotation**: Security best practices
- **Storage**: Where to store tokens
- **Practical Exercise**: Implement refresh token flow

Day 199: Authorization & Role-Based Access

- **RBAC Concept**: Roles and permissions
- **Middleware Authorization**: Protecting routes
- **Permission Checks**: Fine-grained access
- **Practical Exercise**: Build role-based system

Day 200: OAuth 2.0 Implementation

- **OAuth Flow**: Understanding the process
- **Provider Integration**: Google, GitHub OAuth
- **Token Exchange**: Getting user data
- **Practical Exercise**: Implement social login

Day 201: File Upload Handling

- **Multer Middleware**: Handling multipart data
- **File Storage**: Disk and memory storage
- **File Validation**: Type and size checks
- **Practical Exercise**: Build file upload API

Day 202: Cloud Storage Integration

- **AWS S3 Basics**: S3 setup and configuration
- **Upload to S3**: Using AWS SDK
- **Signed URLs**: Secure file access
- **Practical Exercise**: Upload files to S3

Day 203: Email Services

- **Nodemailer**: Sending emails
- **Email Templates**: HTML email templates
- **Email Services**: SendGrid, AWS SES
- **Practical Exercise**: Implement email notifications

Day 204: Real-time Communication - WebSockets

- **WebSocket Protocol**: Understanding WebSockets
- **Socket.io Setup**: Real-time library
- **Events**: Emitting and listening
- **Practical Exercise**: Build chat server

Day 205: Socket.io Advanced

- **Rooms and Namespaces**: Organizing connections
- **Broadcasting**: Sending to multiple clients
- **Authentication**: Securing socket connections
- **Practical Exercise**: Build multiplayer game backend

Day 206: API Rate Limiting

- **Rate Limiting Concepts**: Preventing abuse
- **express-rate-limit**: Implementation
- **Rate Limit Strategies**: Sliding window, token bucket
- **Practical Exercise**: Add rate limiting to API

Day 207: Caching Strategies

- **Caching Concepts**: When to cache
- **Redis Basics**: In-memory caching
- **Cache Invalidation**: Updating cached data
- **Practical Exercise**: Implement Redis caching

Day 208: API Documentation

- **Swagger/OpenAPI**: API specification
- **swagger-jsdoc**: Generating docs from code
- **Swagger UI**: Interactive documentation
- **Practical Exercise**: Document API endpoints

Day 209: Logging & Monitoring

- **Winston Logger**: Structured logging
- **Log Levels**: Debug, info, error levels
- **Log Storage**: File and cloud logging
- **Practical Exercise**: Implement comprehensive logging

Day 210: Environment Configuration

- **dotenv**: Environment variables
- **Config Management**: Different environments
- **Secrets Management**: Securing sensitive data
- **Practical Exercise**: Set up multi-environment config

Day 211: API Testing with Jest

- **Supertest Library**: HTTP testing
- **Testing Endpoints**: Request/response tests
- **Database Mocking**: Test isolation
- **Practical Exercise**: Write API tests

Day 212: Database Migrations

- **Migration Concept**: Schema versioning
- **Migration Tools**: Mongoose migrations
- **Rollback Strategies**: Undoing changes
- **Practical Exercise**: Create migration system

Day 213: Background Jobs

- **Job Queues**: Bull/BullMQ library
- **Task Processing**: Async task handling
- **Scheduled Jobs**: Cron jobs
- **Practical Exercise**: Implement background processing

Day 214: Microservices Basics

- **Microservices Architecture**: Service decomposition
- **Service Communication**: REST, gRPC
- **API Gateway**: Routing requests
- **Practical Exercise**: Design microservices architecture

Day 215: GraphQL Server

- **GraphQL Basics**: Schema and resolvers
- **Apollo Server**: GraphQL server setup
- **Queries and Mutations**: Defining operations
- **Practical Exercise**: Build GraphQL API

Phase 12: Advanced Mobile & Integration (Days 216-240)

Day 216: Backend-Frontend Integration

- **API Client Setup**: Axios configuration
- **Environment URLs**: Dev, staging, production
- **Request/Response Interceptors**: Token injection
- **Practical Exercise**: Connect mobile app to backend

Day 217: Authentication Flow Integration

- **Login Flow**: Complete auth integration
- **Token Management**: Storing and refreshing
- **Protected Routes**: Auth-based navigation
- **Practical Exercise**: Implement full auth in app

Day 218: Real-time Features

- **Socket.io Client**: React Native integration
- **Real-time Updates**: Live data sync

- **Connection Management**: Reconnection logic
- **Practical Exercise**: Build real-time chat app

Day 219: Offline Mode Implementation

- **Offline Detection**: NetInfo usage
- **Offline Queue**: Queuing requests
- **Sync Strategy**: Uploading queued data
- **Practical Exercise**: Build offline-first feature

Day 220: Image Upload & Optimization

- **Image Compression**: Client-side optimization
- **Upload Progress**: Tracking uploads
- **Error Handling**: Retry logic
- **Practical Exercise**: Complete image upload flow

Day 221: Push Notification Backend

- **FCM Admin SDK**: Server-side notifications
- **Notification Targeting**: User/topic notifications
- **Data Notifications**: Custom payloads
- **Practical Exercise**: Send notifications from server

Day 222: Analytics Integration

- **Firebase Analytics**: Event tracking
- **Custom Events**: Defining analytics events
- **User Properties**: Tracking user attributes
- **Practical Exercise**: Implement analytics tracking

Day 223: Crash Reporting

- **Firebase Crashlytics**: Crash tracking
- **Error Boundaries**: Catching React errors
- **Custom Logs**: Adding context to crashes
- **Practical Exercise**: Set up crash reporting

Day 224: App Performance Monitoring

- **Performance Metrics**: Measuring performance
- **Network Monitoring**: API call tracking
- **Render Performance**: Component profiling
- **Practical Exercise**: Monitor and optimize performance

Day 225: Code Push & OTA Updates

- **CodePush Setup**: Over-the-air updates
- **Update Strategies**: Mandatory vs optional
- **Rollback**: Reverting updates
- **Practical Exercise**: Implement CodePush

Day 226: App Localization

- **i18n Setup**: react-i18next integration
- **Translation Files**: Managing translations
- **Language Switching**: Dynamic language change

- **Practical Exercise**: Add multi-language support

Day 227: Accessibility Advanced

- **Screen Reader Testing**: TalkBack and VoiceOver
- **Accessibility Props**: Complete implementation
- **Keyboard Navigation**: Focus management
- **Practical Exercise**: Make app fully accessible

Day 228: App Security Best Practices

- **Secure Storage**: Sensitive data protection
- **SSL Pinning**: Certificate pinning
- **Code Obfuscation**: ProGuard and minification
- **Practical Exercise**: Implement security measures

Day 229: App Store Optimization

- **App Store Guidelines**: iOS and Android
- **Screenshots & Descriptions**: Marketing assets
- **App Preview Videos**: Creating demos
- **Practical Exercise**: Prepare app store listing

Day 230: Build & Release - iOS

- **Xcode Configuration**: Build settings
- **Code Signing**: Certificates and provisioning
- **TestFlight**: Beta distribution
- **Practical Exercise**: Build iOS release

Day 231: Build & Release - Android

- **Android Studio**: Build configuration
- **Signing APK/AAB**: Release signing
- **Google Play Console**: App submission
- **Practical Exercise**: Build Android release

Day 232: CI/CD for Mobile

- **Fastlane**: Automation tool
- **Automated Builds**: CI pipeline for mobile
- **Automated Testing**: Running tests in CI
- **Practical Exercise**: Set up mobile CI/CD

Day 233: App Monetization

- **Ad Integration**: AdMob setup
- **Banner/Interstitial Ads**: Ad types
- **Rewarded Ads**: Incentivized ads
- **Practical Exercise**: Implement ad monetization

Day 234: TypeScript Advanced

- **Generic Types**: Reusable type definitions
- **Utility Types**: Pick, Omit, Partial
- **Type Guards**: Runtime type checking
- **Practical Exercise**: Refactor app with advanced types

Day 235: Custom Native Modules Advanced

- **Native UI Components**: Creating custom views
- **Native Module Communication**: Callbacks and promises
- **Native Events**: Emitting to JavaScript
- **Practical Exercise**: Build complex native module

Day 236: App Architecture Patterns

- **Clean Architecture**: Layered architecture
- **MVVM Pattern**: Model-View-ViewModel
- **Repository Pattern**: Data abstraction
- **Practical Exercise**: Refactor app with clean architecture

Day 237: Large Scale App Structure

- **Feature-Based Structure**: Organizing by feature
- **Shared Components**: Component library
- **Code Splitting**: Lazy loading screens
- **Practical Exercise**: Restructure large application

Day 238: Performance Optimization Advanced

- **Bundle Size**: Analyzing and reducing
- **Memory Optimization**: Preventing leaks
- **Startup Time**: App launch optimization
- **Practical Exercise**: Comprehensive performance audit

Day 239: Debug & Development Tools

- **Flipper**: Advanced debugging
- **React DevTools**: Component inspection
- **Network Inspector**: API debugging
- **Practical Exercise**: Master all debugging tools

Day 240: Production Monitoring

- **Error Tracking**: Sentry integration
- **Performance Monitoring**: Real-world metrics
- **User Analytics**: Behavior tracking
- **Practical Exercise**: Set up production monitoring

Phase 13: Mastery & Real-World Projects (Days 241-250)

Day 241: E-Commerce App - Planning

- **Requirements Analysis**: Defining features
- **Database Design**: Schema planning
- **API Design**: Endpoint planning
- **Practical Exercise**: Create complete app architecture

Day 242: E-Commerce App - Backend Setup

- **User Service**: Authentication and profiles
- **Product Service**: Product CRUD

- **Order Service**: Order management
- **Practical Exercise**: Build backend services

Day 243: E-Commerce App - Frontend Setup

- **Navigation Structure**: Multi-level navigation
- **State Management**: Redux Toolkit setup
- **Component Library**: Reusable components
- **Practical Exercise**: Build app foundation

Day 244: E-Commerce App - Product Features

- **Product Listing**: Browse and search
- **Product Details**: Detailed views
- **Cart Management**: Add to cart functionality
- **Practical Exercise**: Complete product flow

Day 245: E-Commerce App - Checkout Flow

- **Checkout Process**: Multi-step checkout
- **Payment Integration**: Stripe/PayPal
- **Order Confirmation**: Receipt generation
- **Practical Exercise**: Complete purchase flow

Day 246: Social Media App - Core Features

- **Feed Implementation**: Infinite scroll feed
- **Post Creation**: Image and text posts
- **Commenting System**: Nested comments
- **Practical Exercise**: Build social features

Day 247: Social Media App - Real-time Features

- **Real-time Feed**: Socket.io integration
- **Chat Functionality**: Direct messaging
- **Notifications**: Real-time notifications
- **Practical Exercise**: Add real-time capabilities

Day 248: Fitness Tracking App

- **Activity Tracking**: Recording workouts
- **Charts & Analytics**: Data visualization
- **Goal Setting**: Progress tracking
- **Practical Exercise**: Build complete fitness app

Day 249: Portfolio App with Complex Features

- **Animations**: Advanced animations showcase
- **Offline Mode**: Complete offline functionality
- **Custom Native Modules**: Platform-specific features
- **Practical Exercise**: Build showcase portfolio app

Day 250: Career Preparation & Review

- **Code Review Skills**: Reviewing your own code
- **Interview Preparation**: Technical interviews
- **Portfolio Presentation**: Showcasing projects

- **Practical Exercise**: Prepare for job search

Completion Checklist

By the end of 250 days, you will have:

✅ Mastered React Native from fundamentals to advanced concepts ✅ Built expertise in state management (Redux, MobX, Zustand) ✅ Learned animations with Reanimated and Gesture Handler ✅ Integrated native modules and third-party libraries ✅ Mastered Node.js and Express.js backend development ✅ Built RESTful and GraphQL APIs ✅ Implemented authentication, authorization, and security ✅ Deployed multiple full-stack applications ✅ Created a professional portfolio of projects ✅ Prepared for advanced mobile developer roles

Daily Commitment

- **Time**: 1 hour per day minimum
- **Focus**: Deep understanding over surface learning
- **Practice**: Hands-on exercises for every concept
- **Review**: Revisit previous concepts regularly
- **Build**: Apply learning to real projects

Success Tips

1. **Don't skip days** - Consistency is key
2. **Code every day** - Practical exercises are mandatory
3. **Build projects** - Apply multiple concepts together
4. **Ask questions** - Use documentation and communities
5. **Review regularly** - Revisit earlier topics
6. **Stay updated** - Follow React Native updates
7. **Network** - Join developer communities
8. **Document** - Keep a learning journal

Resources to Supplement Learning

- React Native Documentation
- Node.js Official Docs
- Express.js Documentation
- MDN Web Docs
- Stack Overflow
- GitHub Repositories
- YouTube Tutorials
- Dev.to Articles
- React Native Directory

Remember: This is a marathon, not a sprint. Take your time to deeply understand each concept. Building expertise takes dedication and consistent practice. Good luck on your journey to becoming an advanced React Native developer!