Ratopati App - Features & Workflow

This document highlights the major features I worked on in the Ratopati mobile app, alongside the Flutter development workflow I followed for each. The layout pairs features with their implementation steps for clarity.

Feature: News Feed & Category-wise Content
Implementation / Workflow:
- Studied Figma design for news feed UI
- Set up API services to fetch live news
- Implemented category filters (Politics, Business, Sports)
- Optimized list rendering with ListView.builder and pagination
- Added pull-to-refresh and smooth scrolling
- Handled loading and error states using Riverpod

Feature: Live Cricket Scores
Implementation / Workflow:
- Integrated cricket API for live match data
- Designed responsive UI for match cards
- Used Riverpod AsyncNotifierProvider for real-time updates
- Highlighted ongoing matches and scores
- Cached data to reduce unnecessary API calls
- Tested updates in foreground and background

Feature: Quiz Game
Implementation / Workflow:
- Designed interactive quiz screen from Figma
- Fetched multiple-choice questions dynamically from API
- Implemented timer, scoring, and leaderboard logic
- Added feedback animations for correct/incorrect answers

- Managed state using Riverpod providers
- Tested quiz navigation and score calculation

Feature: Push Notifications
Implementation / Workflow:
- Set up Firebase Cloud Messaging
- Configured notifications for breaking news, cricket updates, and quizzes
- Handled foreground, background, and terminated state notifications
- Implemented deep linking to articles, matches, or quiz screens
- Categorized notifications for better UX

Feature: Performance Optimization & State Management
Implementation / Workflow:
- Centralized color, typography, and spacing in design system
- Built reusable widgets (buttons, cards, inputs)
- Used Riverpod for state management of news, cricket, and quiz data
- Minimized widget rebuilds for smoother performance
- Implemented caching, API error handling, and AsyncValue states
- Conducted unit, widget, and integration tests

Flutter App Development Workflow (Figma -> Production):
1. Analyze Figma Design: Understand UI elements, interactions, reusable components.
2. Set Up Project: Initialize Flutter project, organize modular folders, install dependencies (Riverpod, Dio, GoRouter).
3. Implement Design System: Centralized colors, typography, spacing, and reusable widgets.
4. Build UI Screens: Develop static UI first, integrate dynamic API data, ensure responsiveness.
5. Integrate State Management: Use Riverpod / AsyncNotifierProvider to handle API data, loading, and error states.
6. API Integration: Fetch news, cricket scores, quiz questions; map JSON to Dart models using Freezed/JSON Serializable.
7. Testing: Unit tests, widget tests, integration tests; mock APIs with Riverpod test utilities.
8. Optimization: Implement caching, compress assets, add localization, accessibility.
9. Prepare for Production: Set app metadata, icons, splash screen; enable code obfuscation.

10. Build & Release: Generate APK/App Bundle for Android/iOS, submit to Play Store/App Store, monitor analytics and crashes.

Technologies Used:
Flutter, Riverpod, AsyncNotifierProvider, REST APIs, Modular Architecture, Figma for UI/UX

Rabin Shrestha - Contributions:
Converted Figma designs into a clean Flutter design system, integrated Riverpod state management, implemented live news, cricket scores, and quiz games, optimized performance, and ensured high user engagement and retention.