EntryTestGuru Flutter UX/UI Style Guide

1. Color Palette (Flutter Implementation)

AppColors Class

```dart
// lib/core/theme/app_colors.dart
import 'package:flutter/material.dart';

class AppColors {
  // Primary Brand Colors
  static const Color primary900 = Color(0xFF1B365D);
  static const Color primary700 = Color(0xFF2D5A87);
  static const Color primary500 = Color(0xFF4A7BA7);
  static const Color primary300 = Color(0xFF7BA3C7);
  static const Color primary100 = Color(0xFFE8F2FF);

  // User Tier Colors
  static const Color anonymousPrimary = Color(0xFF6B7280);
  static const Color freePrimary = Color(0xFF2D5A87);
  static const Color paidPrimary = Color(0xFF1B365D);

  // ARDE Probability Colors
  static const Color ardeHigh = Color(0xFFDC2626);    // >70%
  static const Color ardeMedium = Color(0xFFF59E0B);  // 30-70%
  static const Color ardeLow = Color(0xFF6B7280);     // 0-30%

  // Semantic Colors
  static const Color success = Color(0xFF10B981);
  static const Color warning = Color(0xFF59E0B);
  static const Color error = Color(0xFFEF4444);
  static const Color info = Color(0xFF3B82F6);

  // Dark Theme Colors
  static const Color darkBgPrimary = Color(0xFF0F172A);
  static const Color darkBgSecondary = Color(0xFF1E293B);
  static const Color darkBgTertiary = Color(0xFF334155);
  static const Color darkBgAccent = Color(0xFF475569);

  static const Color darkTextPrimary = Color(0xFFF8FAFC);
  static const Color darkTextSecondary = Color(0xFFCBD5E1);
```

```
  static const Color darkTextTertiary = Color(0xFF94A3B8);
  static const Color darkTextMuted = Color(0xFF64748B);

  // Light Theme Colors
  static const Color lightBgPrimary = Color(0xFFFFFFFF);
  static const Color lightBgSecondary = Color(0xFFF8FAFC);
  static const Color lightBgTertiary = Color(0xFFF1F5F9);
  static const Color lightBgAccent = Color(0xFFE2E8F0);

  static const Color lightTextPrimary = Color(0xFF0F172A);
  static const Color lightTextSecondary = Color(0xFF334155);
  static const Color lightTextTertiary = Color(0xFF475569);
  static const Color lightTextMuted = Color(0xFF64748B);
}
```

Theme Data Configuration

```
// lib/core/theme/app_theme.dart
import 'package:flutter/material.dart';
import 'app_colors.dart';

class AppTheme {
  static ThemeData get lightTheme {
    return ThemeData(
      useMaterial3: true,
      brightness: Brightness.light,
      colorScheme: ColorScheme.light(
        primary: AppColors.primary700,
        secondary: AppColors.primary500,
        surface: AppColors.lightBgPrimary,
        background: AppColors.lightBgSecondary,
        error: AppColors.error,
        onPrimary: Colors.white,
        onSecondary: Colors.white,
        onSurface: AppColors.lightTextPrimary,
        onBackground: AppColors.lightTextPrimary,
        onError: Colors.white,
      ),
      scaffoldBackgroundColor: AppColors.lightBgPrimary,
      appBarTheme: AppBarTheme(
        backgroundColor: AppColors.lightBgPrimary,
        foregroundColor: AppColors.lightTextPrimary,
        elevation: 0,
        centerTitle: true,
      ),
```

```
    );
  }

  static ThemeData get darkTheme {
    return ThemeData(
      useMaterial3: true,
      brightness: Brightness.dark,
      colorScheme: ColorScheme.dark(
        primary: AppColors.primary500,
        secondary: AppColors.primary300,
        surface: AppColors.darkBgSecondary,
        background: AppColors.darkBgPrimary,
        error: AppColors.error,
        onPrimary: Colors.white,
        onSecondary: Colors.black,
        onSurface: AppColors.darkTextPrimary,
        onBackground: AppColors.darkTextPrimary,
        onError: Colors.white,
      ),
      scaffoldBackgroundColor: AppColors.darkBgPrimary,
      appBarTheme: AppBarTheme(
        backgroundColor: AppColors.darkBgPrimary,
        foregroundColor: AppColors.darkTextPrimary,
        elevation: 0,
        centerTitle: true,
      ),
    );
  }
}
```

2. Typography (Flutter TextTheme)

Typography Scale

```
// lib/core/theme/app_text_styles.dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

class AppTextStyles {
  // Base font family
  static String get fontFamily => GoogleFonts.inter().fontFamily!;

  // Display Styles (Large headings)
  static TextStyle get displayLarge => GoogleFonts.inter(
```

```
  fontSize: 36.0, // 2.25rem
  fontWeight: FontWeight.w700,
  height: 1.25,
  letterSpacing: -0.5,
);

static TextStyle get displayMedium => GoogleFonts.inter(
  fontSize: 30.0, // 1.875rem
  fontWeight: FontWeight.w600,
  height: 1.25,
  letterSpacing: -0.25,
);

static TextStyle get displaySmall => GoogleFonts.inter(
  fontSize: 24.0, // 1.5rem
  fontWeight: FontWeight.w600,
  height: 1.25,
);

// Headline Styles
static TextStyle get headlineLarge => GoogleFonts.inter(
  fontSize: 20.0, // 1.25rem
  fontWeight: FontWeight.w600,
  height: 1.3,
);

static TextStyle get headlineMedium => GoogleFonts.inter(
  fontSize: 18.0, // 1.125rem
  fontWeight: FontWeight.w500,
  height: 1.3,
);

static TextStyle get headlineSmall => GoogleFonts.inter(
  fontSize: 16.0, // 1rem
  fontWeight: FontWeight.w500,
  height: 1.3,
);

// Body Styles
static TextStyle get bodyLarge => GoogleFonts.inter(
  fontSize: 16.0, // 1rem
  fontWeight: FontWeight.w400,
  height: 1.5,
);

static TextStyle get bodyMedium => GoogleFonts.inter(
  fontSize: 14.0, // 0.875rem
```

```
  fontWeight: FontWeight.w400,
  height: 1.5,
);

static TextStyle get bodySmall => GoogleFonts.inter(
  fontSize: 12.0, // 0.75rem
  fontWeight: FontWeight.w400,
  height: 1.4,
);

// Label Styles
static TextStyle get labelLarge => GoogleFonts.inter(
  fontSize: 14.0,
  fontWeight: FontWeight.w500,
  height: 1.4,
  letterSpacing: 0.1,
);

static TextStyle get labelMedium => GoogleFonts.inter(
  fontSize: 12.0,
  fontWeight: FontWeight.w500,
  height: 1.4,
  letterSpacing: 0.5,
);

static TextStyle get labelSmall => GoogleFonts.inter(
  fontSize: 10.0,
  fontWeight: FontWeight.w500,
  height: 1.4,
  letterSpacing: 0.5,
);

// Specialized Styles
static TextStyle get questionText => GoogleFonts.inter(
  fontSize: 18.0,
  fontWeight: FontWeight.w400,
  height: 1.5,
);

static TextStyle get mcqOption => GoogleFonts.inter(
  fontSize: 16.0,
  fontWeight: FontWeight.w400,
  height: 1.4,
);

static TextStyle get mathContent => GoogleFonts.crimsonText(
  fontSize: 18.0,
```

```
    fontWeight: FontWeight.w400,
    height: 1.6,
  );

  static TextStyle get monoCode => GoogleFonts.jetBrainsMono(
    fontSize: 14.0,
    fontWeight: FontWeight.w400,
    height: 1.4,
  );
}
```

TextTheme Integration

```
// In app_theme.dart, add to ThemeData:
textTheme: TextTheme(
  displayLarge: AppTextStyles.displayLarge,
  displayMedium: AppTextStyles.displayMedium,
  displaySmall: AppTextStyles.displaySmall,
  headlineLarge: AppTextStyles.headlineLarge,
  headlineMedium: AppTextStyles.headlineMedium,
  headlineSmall: AppTextStyles.headlineSmall,
  bodyLarge: AppTextStyles.bodyLarge,
  bodyMedium: AppTextStyles.bodyMedium,
  bodySmall: AppTextStyles.bodySmall,
  labelLarge: AppTextStyles.labelLarge,
  labelMedium: AppTextStyles.labelMedium,
  labelSmall: AppTextStyles.labelSmall,
),
```

3. Spacing & Layout (Flutter Dimensions)

Responsive Framework with Material 3 Window Size Classes

```
// pubspec.yaml - Add responsive_framework package
dependencies:
  flutter:
    sdk: flutter
  responsive_framework: ^1.5.1

  # State Management
  flutter_riverpod: ^2.4.9
```

```yaml
  # Fonts & Icons
  google_fonts: ^6.1.0
  flutter_svg: ^2.0.9

  # UI & Animations
  animations: ^2.0.8

  # Storage
  shared_preferences: ^2.2.2

  # Responsive
  flutter_screenutil: ^5.9.0

  # Accessibility
  flutter_tts: ^3.8.3
```

Material 3 Window Size Classes Implementation

```dart
// lib/core/theme/app_dimensions.dart - Material 3 + Responsive Framew
class AppDimensions {
  // Material 3 Window Size Classes using Responsive Framework
  // Source: https://m3.material.io/foundations/layout/applying-layout

  // Material 3 Breakpoint Names (for Responsive Framework)
  static const String compact = 'COMPACT';        // 0-600dp (Phones,
  static const String medium = 'MEDIUM';          // 600-840dp (Large
  static const String expanded = 'EXPANDED';      // 840dp+ (Tablets,

  // Material 3 Breakpoint Values
  static const double compactEnd = 600.0;         // End of compact ra
  static const double mediumStart = 601.0;        // Start of medium r
  static const double mediumEnd = 840.0;          // End of medium ran
  static const double expandedStart = 841.0;      // Start of expanded

  // Spacing Scale (Material 3 8dp baseline grid)
  static const double space1 = 4.0;   // 0.5 * 8dp
  static const double space2 = 8.0;   // 1 * 8dp
  static const double space3 = 12.0;  // 1.5 * 8dp
  static const double space4 = 16.0;  // 2 * 8dp
  static const double space5 = 20.0;  // 2.5 * 8dp
  static const double space6 = 24.0;  // 3 * 8dp
  static const double space8 = 32.0;  // 4 * 8dp
  static const double space10 = 40.0; // 5 * 8dp
  static const double space12 = 48.0; // 6 * 8dp
```

```dart
  static const double space16 = 64.0; // 8 * 8dp
  static const double space20 = 80.0; // 10 * 8dp

  // Material 3 Border Radius Scale
  static const double radiusNone = 0.0;
  static const double radiusExtraSmall = 4.0;
  static const double radiusSmall = 8.0;
  static const double radiusMedium = 12.0;
  static const double radiusLarge = 16.0;
  static const double radiusExtraLarge = 28.0;
  static const double radiusCircular = 1000.0; // Fully rounded

  // Material 3 Touch Targets
  static const double minTouchTarget = 48.0;    // Material 3 minimum
  static const double comfortableTouchTarget = 56.0; // Recommended si

  // Dynamic Card Padding (Material 3 spacing)
  static const double cardPaddingBase = 16.0;   // Compact windows
  static const double cardPaddingMedium = 24.0; // Medium windows
  static const double cardPaddingLarge = 32.0;  // Expanded windows
}
```

App Setup with Responsive Framework + Material 3

```dart
// lib/main.dart - Complete App Setup
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:responsive_framework/responsive_framework.dart';
import 'core/theme/app_theme.dart';
import 'core/theme/app_dimensions.dart';
import 'providers/theme_provider.dart';
import 'widgets/theme_switcher.dart';
import 'screens/main_screen.dart';

void main() {
  runApp(
    const ProviderScope(
      child: EntryTestGuruApp(),
    ),
  );
}

class EntryTestGuruApp extends ConsumerWidget {
  const EntryTestGuruApp({super.key});
```

```
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final themeMode = ref.watch(themeProvider);

    return MaterialApp(
      title: 'EntryTestGuru',
      theme: AppTheme.lightTheme,
      darkTheme: AppTheme.darkTheme,
      themeMode: themeMode,

      // Responsive Framework Setup with Material 3 breakpoints
      builder: (context, child) => ResponsiveBreakpoints.builder(
        child: Stack(
          children: [
            child!,
            const ThemeSwitcher(), // Always accessible theme switcher
          ],
        ),
        breakpoints: [
          // Material 3 Window Size Classes
          const Breakpoint(
            start: 0,
            end: AppDimensions.compactEnd,
            name: AppDimensions.compact,
          ), // 0-600dp: All phones, portrait tablets

          const Breakpoint(
            start: AppDimensions.mediumStart,
            end: AppDimensions.mediumEnd,
            name: AppDimensions.medium,
          ), // 601-840dp: Large phones, small tablets

          const Breakpoint(
            start: AppDimensions.expandedStart,
            end: double.infinity,
            name: AppDimensions.expanded,
          ), // 841dp+: Tablets, desktops, foldables unfolded
        ],
      ),

      home: const MainScreen(),
    );
  }
}
```

Responsive Utils with Material 3 + Responsive Framework

```dart
// lib/core/utils/responsive_utils.dart - Material 3 + Responsive Fram
import 'package:flutter/material.dart';
import 'package:responsive_framework/responsive_framework.dart';
import '../theme/app_dimensions.dart';

class ResponsiveUtils {
  // Material 3 Window Size Class detection using Responsive Framework
  static WindowSizeClass getWindowSizeClass(BuildContext context) {
    final breakpoint = ResponsiveBreakpoints.of(context).screenType;

    final width = _getWidthSizeClass(breakpoint);
    final height = _getHeightSizeClass(context);

    return WindowSizeClass(width: width, height: height);
  }

  static WindowWidthSizeClass _getWidthSizeClass(String breakpoint) {
    switch (breakpoint) {
      case AppDimensions.compact:
        return WindowWidthSizeClass.compact;    // 0-600dp
      case AppDimensions.medium:
        return WindowWidthSizeClass.medium;     // 600-840dp
      case AppDimensions.expanded:
        return WindowWidthSizeClass.expanded;   // 840dp+
      default:
        return WindowWidthSizeClass.compact;    // Fallback
    }
  }

  static WindowHeightSizeClass _getHeightSizeClass(BuildContext contex
    final height = MediaQuery.of(context).size.height;
    if (height < 480) {
      return WindowHeightSizeClass.compact;
    } else if (height < 900) {
      return WindowHeightSizeClass.medium;
    } else {
      return WindowHeightSizeClass.expanded;
    }
  }

  // Material 3 Navigation Patterns
  static NavigationType getNavigationType(BuildContext context) {
    final widthClass = getWindowSizeClass(context).width;

    switch (widthClass) {
      case WindowWidthSizeClass.compact:
```

```
      return NavigationType.bottomNavigation;  // Phones
    case WindowWidthSizeClass.medium:
      return NavigationType.navigationRail;    // Large phones, smal
    case WindowWidthSizeClass.expanded:
      return NavigationType.navigationDrawer;  // Tablets, desktops
  }
}

// Responsive Framework Helper Methods
static bool isCompact(BuildContext context) {
  return ResponsiveBreakpoints.of(context).equals(AppDimensions.comp
}

static bool isMedium(BuildContext context) {
  return ResponsiveBreakpoints.of(context).equals(AppDimensions.medi
}

static bool isExpanded(BuildContext context) {
  return ResponsiveBreakpoints.of(context).equals(AppDimensions.expa
}

static bool isMediumOrLarger(BuildContext context) {
  return ResponsiveBreakpoints.of(context).largerThan(AppDimensions.
}

static bool isExpandedOrLarger(BuildContext context) {
  return ResponsiveBreakpoints.of(context).largerThan(AppDimensions.
}

// EntryTestGuru Layout Decisions
static bool shouldUseCompactLayout(BuildContext context) {
  return isCompact(context);
}

static bool shouldShowSideBySide(BuildContext context) {
  return isMediumOrLarger(context);
}

static bool shouldShowExplanationPanel(BuildContext context) {
  return isExpandedOrLarger(context);
}

static int getGridColumns(BuildContext context) {
  if (isCompact(context)) {
    return 1; // Single column for questions
  } else if (isMedium(context)) {
    return 2; // Two columns for analytics
```

```
    } else {
      return 3; // Three+ columns for dashboard
    }
  }

  static double getCardPadding(BuildContext context) {
    if (isCompact(context)) {
      return AppDimensions.cardPaddingBase;
    } else if (isMedium(context)) {
      return AppDimensions.cardPaddingMedium;
    } else {
      return AppDimensions.cardPaddingLarge;
    }
  }

  static EdgeInsets getScreenPadding(BuildContext context) {
    if (isCompact(context)) {
      return const EdgeInsets.all(AppDimensions.space4);
    } else if (isMedium(context)) {
      return const EdgeInsets.all(AppDimensions.space6);
    } else {
      return const EdgeInsets.symmetric(
        horizontal: AppDimensions.space8,
        vertical: AppDimensions.space6,
      );
    }
  }

  // Analytics Dashboard Layout
  static int getAnalyticsColumns(BuildContext context) {
    if (isCompact(context)) {
      return 1; // Stack vertically
    } else if (isMedium(context)) {
      return 2; // Two columns
    } else {
      return 3; // Three columns for full dashboard
    }
  }
}

// Material 3 Window Size Classes (same enums as before)
class WindowSizeClass {
  final WindowWidthSizeClass width;
  final WindowHeightSizeClass height;

  const WindowSizeClass({
    required this.width,
```

```
    required this.height,
  });
}

enum WindowWidthSizeClass {
  compact,    // 0-600dp: Phones, portrait tablets
  medium,     // 600-840dp: Large phones, small tablets
  expanded,   // 840dp+: Tablets, desktops, foldables unfolded
}

enum WindowHeightSizeClass {
  compact,    // 0-480dp: Landscape phones
  medium,     // 480-900dp: Most devices
  expanded,   // 900dp+: Very tall screens
}

enum NavigationType {
  bottomNavigation,   // Compact: Bottom nav bar
  navigationRail,     // Medium: Side rail
  navigationDrawer,   // Expanded: Permanent drawer
}
```

Responsive Builder with Material 3 + Responsive Framework

```
// lib/widgets/responsive_builder.dart - Material 3 + Responsive Frame
import 'package:flutter/material.dart';
import 'package:responsive_framework/responsive_framework.dart';
import '../core/theme/app_dimensions.dart';

class ResponsiveBuilder extends StatelessWidget {
  final Widget Function(BuildContext, String) compact;
  final Widget Function(BuildContext, String)? medium;
  final Widget Function(BuildContext, String)? expanded;

  const ResponsiveBuilder({
    super.key,
    required this.compact,
    this.medium,
    this.expanded,
  });

  @override
  Widget build(BuildContext context) {
    final breakpoint = ResponsiveBreakpoints.of(context).screenType;
```

```
    switch (breakpoint) {
      case AppDimensions.compact:
        return compact(context, breakpoint);

      case AppDimensions.medium:
        return (medium ?? compact)(context, breakpoint);

      case AppDimensions.expanded:
        return (expanded ?? medium ?? compact)(context, breakpoint);

      default:
        return compact(context, breakpoint);
    }
  }
}

// Responsive Value Helper
class ResponsiveValue<T> {
  final T compact;
  final T? medium;
  final T? expanded;

  const ResponsiveValue({
    required this.compact,
    this.medium,
    this.expanded,
  });

  T getValue(BuildContext context) {
    final breakpoint = ResponsiveBreakpoints.of(context).screenType;

    switch (breakpoint) {
      case AppDimensions.compact:
        return compact;
      case AppDimensions.medium:
        return medium ?? compact;
      case AppDimensions.expanded:
        return expanded ?? medium ?? compact;
      default:
        return compact;
    }
  }
}

// Usage Helper Widget
class ResponsiveWidget extends StatelessWidget {
```

```
  final ResponsiveValue<Widget> responsiveValue;

  const ResponsiveWidget({
    super.key,
    required this.responsiveValue,
  });

  @override
  Widget build(BuildContext context) {
    return responsiveValue.getValue(context);
  }
}
```

Practice Screen Example with Responsive Framework

```
// lib/screens/practice_screen.dart - Material 3 + Responsive Framewor
import 'package:flutter/material.dart';
import 'package:responsive_framework/responsive_framework.dart';
import '../widgets/responsive_builder.dart';
import '../widgets/app_card.dart';
import '../widgets/app_button.dart';
import '../widgets/arde_badge.dart';
import '../core/theme/app_dimensions.dart';
import '../core/theme/app_text_styles.dart';
import '../core/utils/responsive_utils.dart';

class PracticeScreen extends StatelessWidget {
  const PracticeScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Padding(
          padding: ResponsiveUtils.getScreenPadding(context),
          child: ResponsiveBuilder(
            compact: (context, breakpoint) => _buildCompactLayout(cont
            medium: (context, breakpoint) => _buildMediumLayout(contex
            expanded: (context, breakpoint) => _buildExpandedLayout(co
          ),
        ),
      ),
    );
  }
```

```
Widget _buildCompactLayout(BuildContext context) {
  // 0-600dp: All phones including iPhone 16 Pro Max, Galaxy S24 Ult
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      // Header
      _buildHeader(context),
      const SizedBox(height: AppDimensions.space6),

      // Question Card
      Expanded(
        child: _buildQuestionCard(context, isCompact: true),
      ),
    ],
  );
}

Widget _buildMediumLayout(BuildContext context) {
  // 600-840dp: Large phones landscape, small tablets
  return Row(
    children: [
      // Main content
      Expanded(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            _buildHeader(context),
            const SizedBox(height: AppDimensions.space6),
            Expanded(
              child: _buildQuestionCard(context, isCompact: false),
            ),
          ],
        ),
      ),
    ],
  );
}

Widget _buildExpandedLayout(BuildContext context) {
  // 840dp+: Tablets, foldables unfolded, desktops
  return Row(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      // Question on the left
      Expanded(
        flex: 3,
        child: Column(
```

```
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            _buildHeader(context),
            const SizedBox(height: AppDimensions.space6),
            Expanded(
              child: _buildQuestionCard(context, isCompact: false),
            ),
          ],
        ),
      ),

      const SizedBox(width: AppDimensions.space6),

      // Explanation panel on the right
      Expanded(
        flex: 2,
        child: _buildExplanationPanel(context),
      ),
    ],
  );
}

Widget _buildHeader(BuildContext context) {
  return ResponsiveWidget(
    responsiveValue: ResponsiveValue<Widget>(
      compact: Text(
        'Practice Mode',
        style: AppTextStyles.displayMedium.copyWith(
          color: Theme.of(context).colorScheme.onSurface,
        ),
      ),
      expanded: Row(
        children: [
          Text(
            'Practice Mode',
            style: AppTextStyles.displayLarge.copyWith(
              color: Theme.of(context).colorScheme.onSurface,
            ),
          ),
          const Spacer(),
          _buildProgressIndicator(context),
        ],
      ),
    ),
  );
}
```

```
Widget _buildQuestionCard(BuildContext context, {required bool isCom
  final cardPadding = ResponsiveUtils.getCardPadding(context);

  return AppCard(
    padding: EdgeInsets.all(cardPadding),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        // Question header with ARDE badge
        Row(
          children: [
            Expanded(
              child: Text(
                'Question 5 of 20',
                style: AppTextStyles.labelLarge.copyWith(
                  color: Theme.of(context).colorScheme.onSurfaceVari
                ),
              ),
            ),
            const ArdeBadge(probability: ArdeProbability.high),
          ],
        ),
        SizedBox(height: cardPadding),

        // Question text
        Text(
          'Which of the following is the correct formula for calcula
          style: ResponsiveValue<TextStyle>(
            compact: AppTextStyles.questionText,
            expanded: AppTextStyles.questionText.copyWith(fontSize:
          ).getValue(context).copyWith(
            color: Theme.of(context).colorScheme.onSurface,
          ),
        ),
        SizedBox(height: cardPadding),

        // MCQ Options
        Expanded(
          child: Column(
            children: [
              _buildMCQOption(context, 'A', 'KE = ½mv²', false),
              _buildMCQOption(context, 'B', 'KE = mv²', false),
              _buildMCQOption(context, 'C', 'KE = ½m²v', false),
              _buildMCQOption(context, 'D', 'KE = 2mv²', false),
            ],
          ),
        ),
```

```
// Action buttons
if (isCompact) ...[
  // Compact layout: stacked buttons
  Column(
    children: [
      SizedBox(
        width: double.infinity,
        child: AppButton(
          text: 'Submit Answer',
          type: ButtonType.primary,
          onPressed: () {},
        ),
      ),
      const SizedBox(height: AppDimensions.space3),
      SizedBox(
        width: double.infinity,
        child: AppButton(
          text: 'Skip Question',
          type: ButtonType.outline,
          onPressed: () {},
        ),
      ),
    ],
  ),
] else ...[
  // Medium/Expanded layout: side-by-side buttons
  Row(
    children: [
      Expanded(
        child: AppButton(
          text: 'Skip Question',
          type: ButtonType.outline,
          onPressed: () {},
        ),
      ),
      const SizedBox(width: AppDimensions.space4),
      Expanded(
        flex: 2,
        child: AppButton(
          text: 'Submit Answer',
          type: ButtonType.primary,
          onPressed: () {},
        ),
      ),
    ],
  ),
```

```
        ],
      ],
    ),
  );
}

Widget _buildExplanationPanel(BuildContext context) {
  return AppCard(
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          'Explanation',
          style: AppTextStyles.headlineMedium.copyWith(
            color: Theme.of(context).colorScheme.onSurface,
          ),
        ),
        const SizedBox(height: AppDimensions.space4),
        Text(
          'The correct formula for kinetic energy is KE = ½mv², wher
          style: AppTextStyles.bodyLarge.copyWith(
            color: Theme.of(context).colorScheme.onSurface,
          ),
        ),
        const SizedBox(height: AppDimensions.space6),
        AppButton(
          text: 'Ask AI Tutor',
          type: ButtonType.outline,
          onPressed: () {},
        ),
      ],
    ),
  );
}

Widget _buildProgressIndicator(BuildContext context) {
  return Container(
    padding: const EdgeInsets.symmetric(
      horizontal: AppDimensions.space4,
      vertical: AppDimensions.space2,
    ),
    decoration: BoxDecoration(
      color: Theme.of(context).colorScheme.primaryContainer,
      borderRadius: BorderRadius.circular(AppDimensions.radiusLarge)
    ),
    child: Text(
      '5/20',
```

```
      style: AppTextStyles.labelMedium.copyWith(
        color: Theme.of(context).colorScheme.onPrimaryContainer,
      ),
    ),
  );
}

Widget _buildMCQOption(
  BuildContext context,
  String option,
  String text,
  bool isSelected,
) {
  return Padding(
    padding: const EdgeInsets.only(bottom: AppDimensions.space3),
    child: Material(
      color: Colors.transparent,
      child: InkWell(
        onTap: () {},
        borderRadius: BorderRadius.circular(AppDimensions.radiusMedi
        child: Container(
          width: double.infinity,
          constraints: const BoxConstraints(
            minHeight: AppDimensions.minTouchTarget,
          ),
          padding: EdgeInsets.all(ResponsiveUtils.getCardPadding(con
          decoration: BoxDecoration(
            border: Border.all(
              color: isSelected
                  ? Theme.of(context).colorScheme.primary
                  : Theme.of(context).colorScheme.outline.withOpacit
              width: isSelected ? 2 : 1,
            ),
            borderRadius: BorderRadius.circular(AppDimensions.radius
            color: isSelected
                ? Theme.of(context).colorScheme.primary.withOpacity(
                : Colors.transparent,
          ),
          child: Row(
            children: [
              Container(
                width: ResponsiveValue<double>(
                  compact: 32.0,
                  expanded: 36.0,
                ).getValue(context),
                height: ResponsiveValue<double>(
                  compact: 32.0,
```

```
                    expanded: 36.0,
                  ).getValue(context),
                  decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    color: isSelected
                        ? Theme.of(context).colorScheme.primary
                        : Colors.transparent,
                    border: Border.all(
                      color: isSelected
                          ? Theme.of(context).colorScheme.primary
                          : Theme.of(context).colorScheme.outline,
                      width: 2,
                    ),
                  ),
                  child: Center(
                    child: Text(
                      option,
                      style: AppTextStyles.labelMedium.copyWith(
                        color: isSelected
                            ? Colors.white
                            : Theme.of(context).colorScheme.onSurface,
                        fontWeight: FontWeight.w600,
                      ),
                    ),
                  ),
                ),
                const SizedBox(width: AppDimensions.space4),
                Expanded(
                  child: Text(
                    text,
                    style: AppTextStyles.mcqOption.copyWith(
                      color: Theme.of(context).colorScheme.onSurface,
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Navigation Implementation with Responsive Framework

```dart
// lib/widgets/app_navigation.dart - Material 3 Navigation Patterns
import 'package:flutter/material.dart';
import 'package:responsive_framework/responsive_framework.dart';
import '../core/theme/app_dimensions.dart';
import '../core/utils/responsive_utils.dart';
import 'academic_icon.dart';

class AppNavigation extends StatelessWidget {
  final int currentIndex;
  final Function(int) onIndexChanged;
  final List<NavigationItem> items;

  const AppNavigation({
    super.key,
    required this.currentIndex,
    required this.onIndexChanged,
    required this.items,
  });

  @override
  Widget build(BuildContext context) {
    final navigationType = ResponsiveUtils.getNavigationType(context);

    switch (navigationType) {
      case NavigationType.bottomNavigation:
        return _buildBottomNavigation(context);
      case NavigationType.navigationRail:
        return _buildNavigationRail(context);
      case NavigationType.navigationDrawer:
        return _buildNavigationDrawer(context);
    }
  }

  // Implementation same as before, but using ResponsiveBreakpoints.of
  // instead of MediaQuery for breakpoint detection

  Widget _buildBottomNavigation(BuildContext context) {
    // Only show on COMPACT breakpoint (0-600dp)
    return ResponsiveVisibility(
      visible: ResponsiveBreakpoints.of(context).equals(AppDimensions.
      child: Container(
        decoration: BoxDecoration(
          color: Theme.of(context).colorScheme.surface,
          border: Border(
            top: BorderSide(
              color: Theme.of(context).colorScheme.outline.withOpacity
```

```
            width: 1,
          ),
        ),
      ),
      child: SafeArea(
        child: Container(
          height: 70,
          padding: const EdgeInsets.symmetric(
            horizontal: AppDimensions.space4,
            vertical: AppDimensions.space2,
          ),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: items.asMap().entries.map((entry) {
              final index = entry.key;
              final item = entry.value;
              final isActive = index == currentIndex;

              return _buildNavItem(
                context: context,
                item: item,
                isActive: isActive,
                onTap: () => onIndexChanged(index),
                isMobile: true,
              );
            }).toList(),
          ),
        ),
      ),
    ),
  );
}

// ... rest of the navigation implementations remain the same
}

// Responsive Visibility Helper
class ResponsiveVisibility extends StatelessWidget {
  final bool visible;
  final Widget child;

  const ResponsiveVisibility({
    super.key,
    required this.visible,
    required this.child,
  });
```

```
  @override
  Widget build(BuildContext context) {
    return visible ? child : const SizedBox.shrink();
  }
}
```

This updated implementation gives you:

✅ **Responsive Framework power** with **Material 3 guidelines**
✅ **Custom breakpoint flexibility** while following standards
✅ **Easy responsive values** with `ResponsiveValue<T>` helper
✅ **Clean breakpoint detection** using `ResponsiveBreakpoints.of(context)`
✅ **Material 3 compliant navigation** patterns
✅ **Perfect device coverage** from phones to desktops

You get the best of both worlds - Responsive Framework's flexibility with Material 3's proven breakpoint strategy! 🎯 content that adapts automatically body: _buildResponsiveBody(), // Secondary body for two-pane layouts (tablets+) secondaryBody: _buildResponsiveSecondaryBody(),);

}

Widget _buildCompactLayout() { // 0-600dp: All phones including iPhone 16 Pro Max, Galaxy S24 Ultra return _getSelectedScreen(); }

Widget _buildMediumLayout() { // 600-840dp: Large phones landscape, small tablets return _getSelectedScreen(); }

Widget _buildExpandedLayout() { // 840dp+: Tablets, foldables unfolded, desktops return Row(children: [Expanded(flex: 2, child: _getSelectedScreen(),), const VerticalDivider(width: 1), // Secondary pane handled by secondaryBody],); }

Widget _buildSecondaryPane() { // Only shown on large screens (840dp+) switch (_selectedTab) { case 0: // Practice return _buildExplanationPanel(); case 1: // Exams return _buildExamSummaryPanel(); case 2: // Analytics return _buildDetailedStatsPanel(); default: return const SizedBox.shrink(); } }

Widget _getSelectedScreen() { switch (_selectedTab) { case 0: return const PracticeScreen(); case 1: return const ExamScreen(); case 2: return const

AnalyticsScreen(); case 3: return const AITutorScreen(); case 4: return const SocialScreen(); default: return const PracticeScreen(); } }

Widget _buildResponsiveBody() { // Use ResponsiveBreakpoints for responsive behavior if (ResponsiveBreakpoints.of(context).equals(ResponsiveBreakpoint.xs) || ResponsiveBreakpoints.of(context).equals(ResponsiveBreakpoint.sm)) { return _buildCompactLayout(); } else if (ResponsiveBreakpoints.of(context).equals(ResponsiveBreakpoint.md)) { return _buildMediumLayout(); } else { return _buildExpandedLayout(); } }

Widget _buildResponsiveSecondaryBody() { // Only show secondary pane on larger screens if (ResponsiveBreakpoints.of(context).largerThan(ResponsiveBreakpoint.md)) { return _buildSecondaryPane(); } return const SizedBox.shrink(); }

Widget _buildExplanationPanel() { return const Card(margin: EdgeInsets.all(16), child: Padding(padding: EdgeInsets.all(16), child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [Text('Explanation', style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold,),), SizedBox(height: 16), Text('Detailed explanation of the current question will appear here...',),],),),); }

// ... other panel implementations }

```
### Built-in Breakpoints (Flutter Official)
```dart
// These are automatically handled by ResponsiveWrapper
class Breakpoints {
  static const Breakpoint small = Breakpoint(endWidth: 600);
  static const Breakpoint medium = Breakpoint(beginWidth: 600, endWidt
  static const Breakpoint mediumLarge = Breakpoint(beginWidth: 840, en
  static const Breakpoint large = Breakpoint(beginWidth: 1200, endWidt
  static const Breakpoint extraLarge = Breakpoint(beginWidth: 1600);
}
```

Custom Responsive Widgets (For Fine Control)
```

```dart
// lib/widgets/responsive_question_card.dart
import 'package:flutter/material.dart';
import 'package:responsive_framework/responsive_framework.dart';

class ResponsiveQuestionCard extends StatelessWidget {
  final String questionText;
  final List<String> options;
  final String? explanation;

  const ResponsiveQuestionCard({
    super.key,
    required this.questionText,
    required this.options,
    this.explanation,
  });

  @override
  Widget build(BuildContext context) {
    // Use ResponsiveBreakpoints for responsive behavior
    if (ResponsiveBreakpoints.of(context).equals(ResponsiveBreakpoint.
        ResponsiveBreakpoints.of(context).equals(ResponsiveBreakpoint.
      return _buildCompactQuestion();
    } else if (ResponsiveBreakpoints.of(context).equals(ResponsiveBrea
      return _buildMediumQuestion();
    } else {
      return _buildExpandedQuestion();
    }
  }

  Widget _buildCompactQuestion() {
    return Card(
      margin: const EdgeInsets.all(16),
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              questionText,
              style: const TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.w500,
              ),
            ),
            const SizedBox(height: 20),
            ...options.map((option) => _buildMCQOption(option, false))
```

```
            const SizedBox(height: 16),
            Row(
              children: [
                Expanded(
                  child: OutlinedButton(
                    onPressed: () => _showExplanationModal(context),
                    child: const Text('Explanation'),
                  ),
                ),
                const SizedBox(width: 12),
                Expanded(
                  flex: 2,
                  child: ElevatedButton(
                    onPressed: () {},
                    child: const Text('Submit'),
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }

  Widget _buildExpandedQuestion() {
    return Row(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        // Question on the left
        Expanded(
          flex: 3,
          child: Card(
            margin: const EdgeInsets.all(16),
            child: Padding(
              padding: const EdgeInsets.all(24),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    questionText,
                    style: const TextStyle(
                      fontSize: 20,
                      fontWeight: FontWeight.w500,
                    ),
                  ),
                  const SizedBox(height: 24),
```

```
                ...options.map((option) => _buildMCQOption(option, f
                const SizedBox(height: 24),
                SizedBox(
                  width: double.infinity,
                  child: ElevatedButton(
                    onPressed: () {},
                    child: const Text('Submit Answer'),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),

      // Explanation on the right
      if (explanation != null)
        Expanded(
          flex: 2,
          child: Card(
            margin: const EdgeInsets.all(16),
            child: Padding(
              padding: const EdgeInsets.all(24),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  const Text(
                    'Explanation',
                    style: TextStyle(
                      fontSize: 18,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  const SizedBox(height: 16),
                  Text(explanation!),
                  const SizedBox(height: 20),
                  OutlinedButton(
                    onPressed: () {},
                    child: const Text('Ask AI Tutor'),
                  ),
                ],
              ),
            ),
          ),
        ),
    ],
  );
```

```
}

Widget _buildMCQOption(String option, bool isSelected) {
  return Container(
    width: double.infinity,
    margin: const EdgeInsets.only(bottom: 12),
    child: OutlinedButton(
      style: OutlinedButton.styleFrom(
        padding: const EdgeInsets.all(16),
        alignment: Alignment.centerLeft,
        side: BorderSide(
          color: isSelected ? Colors.blue : Colors.grey,
          width: isSelected ? 2 : 1,
        ),
        backgroundColor: isSelected ? Colors.blue.withOpacity(0.1) :
      ),
      onPressed: () {},
      child: Text(
        option,
        style: TextStyle(
          color: isSelected ? Colors.blue : null,
          fontSize: 16,
        ),
      ),
    ),
  );
}

void _showExplanationModal(BuildContext context) {
  if (explanation == null) return;

  showModalBottomSheet(
    context: context,
    isScrollControlled: true,
    builder: (context) => DraggableScrollableSheet(
      initialChildSize: 0.7,
      maxChildSize: 0.9,
      minChildSize: 0.5,
      builder: (context, scrollController) => Container(
        padding: const EdgeInsets.all(24),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const Text(
              'Explanation',
              style: TextStyle(
                fontSize: 20,
```

```
                fontWeight: FontWeight.bold,
              ),
            ),
            const SizedBox(height: 16),
            Expanded(
              child: SingleChildScrollView(
                controller: scrollController,
                child: Text(
                  explanation!,
                  style: const TextStyle(fontSize: 16),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}

// ... other implementations
}
```

Automatic Navigation Patterns

```
// No code needed! ResponsiveWrapper automatically provides:

// Compact (0-600dp): Bottom Navigation Bar
// - All phones including iPhone 16 Pro Max (428dp)
// - Galaxy S24 Ultra (480dp)
// - Foldables in folded state

// Medium (600-840dp): Navigation Rail
// - Large phones in landscape
// - Small tablets
// - Foldables partially open

// Expanded (840dp+): Permanent Navigation Drawer
// - Tablets (iPads, Android tablets)
// - Foldables fully unfolded
// - Desktop windows
// - Chromebooks
```

🎯 Key Benefits of Flutter's Built-in Approach:

✅ Zero Configuration Required

- ResponsiveWrapper implements the basic visual layout structure for Material Design 3 that adapts to a variety of screens. It provides a preset of layout, including positions and animations, by handling macro changes in navigational elements and bodies based on the current features of the screen, namely screen width and platform

✅ Official Material 3 Compliance

- ResponsiveWrapper is built upon ResponsiveBreakpoints internally but abstracts some of the complexity with presets based on the Material 3 Design specification
- Automatically follows Google's official guidelines

✅ Automatic Foldable Support

- There is some automatic functionality with foldables to handle the split between panels properly
- No custom detection code needed

✅ Built-in Animations

- Smooth transitions between navigation patterns
- Material 3 compliant entrance/exit animations
- Configurable transition durations

✅ Simplified API

- ResponsiveWrapper is much simpler to use but is not the best if you would like high customizability. Apps that would like more refined layout and/or animation should use ResponsiveBreakpoints directly
- Perfect for EntryTestGuru's needs

This approach gives you **everything you wanted** with minimal code:

- ✅ Industry standard Material 3 breakpoints

- ✅ Automatic foldable handling
- ✅ Zero custom device detection
- ✅ Official Flutter support
- ✅ Built-in navigation patterns
- ✅ Perfect for all modern devices

Bottom line: Use `responsive_framework` package - it's the actively maintained Flutter solution that provides excellent responsive behavior! 🎯

Responsive Helper

```dart
// lib/core/utils/responsive_utils.dart
import 'package:flutter/material.dart';
import '../theme/app_dimensions.dart';

class ResponsiveUtils {
  static bool isMobile(BuildContext context) {
    return MediaQuery.of(context).size.width < AppDimensions.tabletBre
  }

  static bool isTablet(BuildContext context) {
    final width = MediaQuery.of(context).size.width;
    return width >= AppDimensions.tabletBreakpoint &&
           width < AppDimensions.desktopBreakpoint;
  }

  static bool isDesktop(BuildContext context) {
    return MediaQuery.of(context).size.width >= AppDimensions.desktopB
  }

  static double getCardPadding(BuildContext context) {
    if (isMobile(context)) return AppDimensions.cardPaddingMobile;
    if (isTablet(context)) return AppDimensions.cardPaddingTablet;
    return AppDimensions.cardPaddingDesktop;
  }

  static EdgeInsets getScreenPadding(BuildContext context) {
    if (isMobile(context)) {
      return const EdgeInsets.all(AppDimensions.space4);
    } else if (isTablet(context)) {
      return const EdgeInsets.all(AppDimensions.space6);
    } else {
      return const EdgeInsets.symmetric(
```

```
        horizontal: AppDimensions.space8,
        vertical: AppDimensions.space6,
      );
    }
  }
}
```

4. Components (Flutter Widgets)

Theme Switcher Widget

```
// lib/widgets/theme_switcher.dart
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class ThemeSwitcher extends ConsumerWidget {
  const ThemeSwitcher({super.key});

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final isDark = Theme.of(context).brightness == Brightness.dark;

    return Positioned(
      top: AppDimensions.space4 + MediaQuery.of(context).padding.top,
      right: AppDimensions.space4,
      child: Material(
        elevation: 4,
        borderRadius: BorderRadius.circular(AppDimensions.radiusRound)
        child: Container(
          width: AppDimensions.comfortableTouchTarget,
          height: AppDimensions.comfortableTouchTarget,
          decoration: BoxDecoration(
            color: Theme.of(context).colorScheme.surface,
            borderRadius: BorderRadius.circular(AppDimensions.radiusRo
            border: Border.all(
              color: Theme.of(context).colorScheme.outline.withOpacity
              width: 2,
            ),
          ),
          child: IconButton(
            onPressed: () {
              // Toggle theme using Riverpod provider
              ref.read(themeProvider.notifier).toggleTheme();
            },
```

```
            icon: AnimatedSwitcher(
              duration: const Duration(milliseconds: 300),
              child: Icon(
                isDark ? Icons.light_mode : Icons.dark_mode,
                key: ValueKey(isDark),
                color: Theme.of(context).colorScheme.onSurface,
              ),
            ),
            tooltip: isDark ? 'Switch to Light Mode' : 'Switch to Dark
          ),
        ),
      ),
    );
  }
}
```

Custom Button Styles

```
// lib/widgets/custom_buttons.dart
import 'package:flutter/material.dart';
import '../core/theme/app_colors.dart';
import '../core/theme/app_dimensions.dart';

class AppButton extends StatelessWidget {
  final String text;
  final VoidCallback? onPressed;
  final ButtonType type;
  final UserTier? userTier;
  final bool isLoading;

  const AppButton({
    super.key,
    required this.text,
    this.onPressed,
    this.type = ButtonType.primary,
    this.userTier,
    this.isLoading = false,
  });

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: AppDimensions.minTouchTarget,
      child: ElevatedButton(
        onPressed: isLoading ? null : onPressed,
```

```
      style: _getButtonStyle(context),
      child: isLoading
          ? const SizedBox(
              width: 20,
              height: 20,
              child: CircularProgressIndicator(strokeWidth: 2),
            )
          : Text(
              text,
              style: Theme.of(context).textTheme.labelLarge?.copyWit
                color: _getTextColor(context),
                fontWeight: FontWeight.w500,
              ),
            ),
    ),
  );
}

ButtonStyle _getButtonStyle(BuildContext context) {
  Color backgroundColor;

  switch (type) {
    case ButtonType.primary:
      backgroundColor = _getPrimaryColor();
      break;
    case ButtonType.secondary:
      backgroundColor = Colors.transparent;
      break;
    case ButtonType.outline:
      backgroundColor = Colors.transparent;
      break;
  }

  return ElevatedButton.styleFrom(
    backgroundColor: backgroundColor,
    foregroundColor: _getTextColor(context),
    elevation: type == ButtonType.primary ? 2 : 0,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(AppDimensions.radiusMedium
      side: type == ButtonType.outline
          ? BorderSide(color: _getPrimaryColor(), width: 2)
          : BorderSide.none,
    ),
    padding: const EdgeInsets.symmetric(
      horizontal: AppDimensions.space6,
      vertical: AppDimensions.space3,
    ),
```

```
    );
  }

  Color _getPrimaryColor() {
    switch (userTier) {
      case UserTier.anonymous:
        return AppColors.anonymousPrimary;
      case UserTier.free:
        return AppColors.freePrimary;
      case UserTier.paid:
        return AppColors.paidPrimary;
      default:
        return AppColors.primary700;
    }
  }

  Color _getTextColor(BuildContext context) {
    if (type == ButtonType.primary) {
      return Colors.white;
    } else {
      return _getPrimaryColor();
    }
  }
}

enum ButtonType { primary, secondary, outline }
enum UserTier { anonymous, free, paid }
```

ARDE Probability Badge

```
// lib/widgets/arde_badge.dart
import 'package:flutter/material.dart';
import '../core/theme/app_colors.dart';
import '../core/theme/app_dimensions.dart';

class ArdeBadge extends StatelessWidget {
  final ArdeProbability probability;
  final bool showLabel;

  const ArdeBadge({
    super.key,
    required this.probability,
    this.showLabel = true,
  });
```

```
@override
Widget build(BuildContext context) {
  return Container(
    padding: const EdgeInsets.symmetric(
      horizontal: AppDimensions.space3,
      vertical: AppDimensions.space2,
    ),
    decoration: BoxDecoration(
      color: _getBackgroundColor().withOpacity(0.1),
      border: Border.all(color: _getColor(), width: 1),
      borderRadius: BorderRadius.circular(AppDimensions.radiusXLarge
    ),
    child: Row(
      mainAxisSize: MainAxisSize.min,
      children: [
        Icon(
          _getIcon(),
          size: 12,
          color: _getColor(),
        ),
        if (showLabel) ...[
          const SizedBox(width: AppDimensions.space1),
          Text(
            _getLabel(),
            style: Theme.of(context).textTheme.labelSmall?.copyWith(
              color: _getColor(),
              fontWeight: FontWeight.w500,
              letterSpacing: 0.5,
            ),
          ),
        ],
      ],
    ),
  );
}

Color _getColor() {
  switch (probability) {
    case ArdeProbability.high:
      return AppColors.ardeHigh;
    case ArdeProbability.medium:
      return AppColors.ardeMedium;
    case ArdeProbability.low:
      return AppColors.ardeLow;
  }
}
```

```dart
  Color _getBackgroundColor() {
    return _getColor();
  }

  IconData _getIcon() {
    switch (probability) {
      case ArdeProbability.high:
        return Icons.trending_up;
      case ArdeProbability.medium:
        return Icons.trending_flat;
      case ArdeProbability.low:
        return Icons.trending_down;
    }
  }

  String _getLabel() {
    switch (probability) {
      case ArdeProbability.high:
        return 'HIGH ARDE';
      case ArdeProbability.medium:
        return 'MED ARDE';
      case ArdeProbability.low:
        return 'LOW ARDE';
    }
  }
}

enum ArdeProbability { high, medium, low }
```

Custom Card Widget

```dart
// lib/widgets/app_card.dart
import 'package:flutter/material.dart';
import '../core/theme/app_dimensions.dart';
import '../core/utils/responsive_utils.dart';

class AppCard extends StatelessWidget {
  final Widget child;
  final bool isInteractive;
  final VoidCallback? onTap;
  final EdgeInsets? padding;
  final double? elevation;

  const AppCard({
    super.key,
```

```
    required this.child,
    this.isInteractive = false,
    this.onTap,
    this.padding,
    this.elevation,
  });

  @override
  Widget build(BuildContext context) {
    final cardPadding = padding ?? EdgeInsets.all(
      ResponsiveUtils.getCardPadding(context),
    );

    return Material(
      elevation: elevation ?? (isInteractive ? 2 : 1),
      borderRadius: BorderRadius.circular(AppDimensions.radiusLarge),
      color: Theme.of(context).colorScheme.surface,
      child: InkWell(
        onTap: onTap,
        borderRadius: BorderRadius.circular(AppDimensions.radiusLarge)
        child: AnimatedContainer(
          duration: const Duration(milliseconds: 200),
          padding: cardPadding,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(AppDimensions.radiusLa
            border: Border.all(
              color: Theme.of(context).colorScheme.outline.withOpacity
              width: 1,
            ),
          ),
          child: child,
        ),
      ),
    );
  }
}
```

5. Iconography (Flutter Icons)

Custom Icon Widget

```
// lib/widgets/academic_icon.dart
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
```

```
class AcademicIcon extends StatelessWidget {
  final AcademicIconType type;
  final double size;
  final Color? color;
  final bool isActive;

  const AcademicIcon({
    super.key,
    required this.type,
    this.size = 24.0,
    this.color,
    this.isActive = false,
  });

  @override
  Widget build(BuildContext context) {
    final iconColor = color ?? Theme.of(context).colorScheme.onSurface

    return AnimatedSwitcher(
      duration: const Duration(milliseconds: 200),
      child: Icon(
        _getIconData(),
        key: ValueKey('${type}_$isActive'),
        size: size,
        color: iconColor,
        weight: isActive ? 600 : 400,
        fill: isActive ? 1.0 : 0.0,
      ),
    );
  }

  IconData _getIconData() {
    switch (type) {
      case AcademicIconType.practice:
        return isActive ? Icons.edit : Icons.edit_outlined;
      case AcademicIconType.exam:
        return isActive ? Icons.timer : Icons.timer_outlined;
      case AcademicIconType.analytics:
        return isActive ? Icons.bar_chart : Icons.bar_chart_outlined;
      case AcademicIconType.aiTutor:
        return isActive ? Icons.psychology : Icons.psychology_outlined
      case AcademicIconType.leaderboard:
        return isActive ? Icons.emoji_events : Icons.emoji_events_outl
      case AcademicIconType.settings:
        return isActive ? Icons.settings : Icons.settings_outlined;
      case AcademicIconType.profile:
```

```
        return isActive ? Icons.person : Icons.person_outlined;
      case AcademicIconType.bookmark:
        return isActive ? Icons.bookmark : Icons.bookmark_outlined;
    }
  }
}

enum AcademicIconType {
  practice,
  exam,
  analytics,
  aiTutor,
  leaderboard,
  settings,
  profile,
  bookmark,
}
```

6. Motion & Interaction (Flutter Animations)

Animation Constants

```
// lib/core/theme/app_animations.dart
class AppAnimations {
  // Duration Constants
  static const Duration fast = Duration(milliseconds: 150);
  static const Duration normal = Duration(milliseconds: 250);
  static const Duration slow = Duration(milliseconds: 400);

  // Curves
  static const Curve easeInOut = Curves.easeInOut;
  static const Curve easeOut = Curves.easeOut;
  static const Curve bounce = Curves.elasticOut;

  // Page Transitions
  static Route<T> slideTransition<T extends Object?>(
    Widget page,
    RouteSettings settings,
  ) {
    return PageRouteBuilder<T>(
      settings: settings,
      pageBuilder: (context, animation, _) => page,
      transitionDuration: normal,
      transitionsBuilder: (context, animation, secondaryAnimation, chi
```

```
        const begin = Offset(1.0, 0.0);
        const end = Offset.zero;
        final tween = Tween(begin: begin, end: end);
        final offsetAnimation = animation.drive(tween);

        return SlideTransition(
          position: offsetAnimation,
          child: child,
        );
      },
    );
  }
}
```

Feedback Animations

```
// lib/widgets/animated_feedback.dart
import 'package:flutter/material.dart';

class AnimatedFeedback extends StatefulWidget {
  final Widget child;
  final FeedbackType type;
  final bool trigger;

  const AnimatedFeedback({
    super.key,
    required this.child,
    required this.type,
    required this.trigger,
  });

  @override
  State<AnimatedFeedback> createState() => _AnimatedFeedbackState();
}

class _AnimatedFeedbackState extends State<AnimatedFeedback>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _animation;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(milliseconds: 600),
```

```
      vsync: this,
    );

    _animation = widget.type == FeedbackType.success
        ? Tween<double>(begin: 1.0, end: 1.1).animate(
            CurvedAnimation(parent: _controller, curve: Curves.elastic
          )
        : Tween<double>(begin: 0.0, end: 1.0).animate(
            CurvedAnimation(parent: _controller, curve: Curves.elastic
          );
  }

  @override
  void didUpdateWidget(AnimatedFeedback oldWidget) {
    super.didUpdateWidget(oldWidget);
    if (widget.trigger && !oldWidget.trigger) {
      _controller.forward().then((_) => _controller.reverse());
    }
  }

  @override
  Widget build(BuildContext context) {
    return AnimatedBuilder(
      animation: _animation,
      builder: (context, child) {
        return Transform.scale(
          scale: widget.type == FeedbackType.success ? _animation.valu
          child: Transform.translate(
            offset: widget.type == FeedbackType.error
                ? Offset(_animation.value * 10 * (1 - _animation.value
                : Offset.zero,
            child: widget.child,
          ),
        );
      },
    );
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }
}

enum FeedbackType { success, error }
```

7. Accessibility (Flutter Implementation)

Focus Management

```
// lib/core/accessibility/focus_helper.dart
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

class FocusHelper {
  static void announceFocus(BuildContext context, String message) {
    SemanticsService.announce(message, TextDirection.ltr);
  }

  static Widget buildFocusableItem({
    required Widget child,
    required VoidCallback onTap,
    String? semanticLabel,
    String? semanticHint,
  }) {
    return Semantics(
      label: semanticLabel,
      hint: semanticHint,
      button: true,
      child: Focus(
        child: Builder(
          builder: (context) {
            final focusNode = Focus.of(context);
            return GestureDetector(
              onTap: onTap,
              child: AnimatedContainer(
                duration: const Duration(milliseconds: 200),
                decoration: BoxDecoration(
                  border: focusNode.hasFocus
                      ? Border.all(
                          color: Theme.of(context).colorScheme.primary
                          width: 3,
                        )
                      : null,
                  borderRadius: BorderRadius.circular(8),
                ),
                child: child,
              ),
            );
          },
        ),
```

```
      ),
    );
  }
}
```

Screen Reader Support

```
// lib/widgets/accessible_text.dart
import 'package:flutter/material.dart';

class AccessibleText extends StatelessWidget {
  final String text;
  final TextStyle? style;
  final String? semanticLabel;
  final bool excludeSemantics;

  const AccessibleText(
    this.text, {
    super.key,
    this.style,
    this.semanticLabel,
    this.excludeSemantics = false,
  });

  @override
  Widget build(BuildContext context) {
    return Semantics(
      label: semanticLabel ?? text,
      excludeSemantics: excludeSemantics,
      child: Text(
        text,
        style: style,
      ),
    );
  }
}
```

8. Responsive Implementation

Responsive Builder

```dart
// lib/widgets/responsive_builder.dart
import 'package:flutter/material.dart';
import '../core/theme/app_dimensions.dart';

class ResponsiveBuilder extends StatelessWidget {
  final Widget Function(BuildContext, BoxConstraints) mobile;
  final Widget Function(BuildContext, BoxConstraints)? tablet;
  final Widget Function(BuildContext, BoxConstraints)? desktop;

  const ResponsiveBuilder({
    super.key,
    required this.mobile,
    this.tablet,
    this.desktop,
  });

  @override
  Widget build(BuildContext context) {
    return LayoutBuilder(
      builder: (context, constraints) {
        if (constraints.maxWidth >= AppDimensions.desktopBreakpoint) {
          return desktop?.call(context, constraints) ??
                 tablet?.call(context, constraints) ??
                 mobile(context, constraints);
        } else if (constraints.maxWidth >= AppDimensions.tabletBreakpo
          return tablet?.call(context, constraints) ??
                 mobile(context, constraints);
        } else {
          return mobile(context, constraints);
        }
      },
    );
  }
}
```

Navigation Implementation

```dart
// lib/widgets/app_navigation.dart
import 'package:flutter/material.dart';
import '../core/theme/app_dimensions.dart';
import '../core/utils/responsive_utils.dart';
import 'academic_icon.dart';
```

```
class AppNavigation extends StatelessWidget {
  final int currentIndex;
  final Function(int) onIndexChanged;
  final List<NavigationItem> items;

  const AppNavigation({
    super.key,
    required this.currentIndex,
    required this.onIndexChanged,
    required this.items,
  });

  @override
  Widget build(BuildContext context) {
    return ResponsiveUtils.isMobile(context)
        ? _buildBottomNavigation(context)
        : _buildSideNavigation(context);
  }

  Widget _buildBottomNavigation(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        color: Theme.of(context).colorScheme.surface,
        border: Border(
          top: BorderSide(
            color: Theme.of(context).colorScheme.outline.withOpacity(0
            width: 1,
          ),
        ),
      ),
      child: SafeArea(
        child: Container(
          height: 70,
          padding: const EdgeInsets.symmetric(
            horizontal: AppDimensions.space4,
            vertical: AppDimensions.space2,
          ),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: items.asMap().entries.map((entry) {
              final index = entry.key;
              final item = entry.value;
              final isActive = index == currentIndex;

              return _buildNavItem(
                context: context,
                item: item,
```

```
              isActive: isActive,
              onTap: () => onIndexChanged(index),
              isMobile: true,
            );
          }).toList(),
        ),
      ),
    ),
  );
}

Widget _buildSideNavigation(BuildContext context) {
  return Container(
    width: 280,
    decoration: BoxDecoration(
      color: Theme.of(context).colorScheme.surface,
      border: Border(
        right: BorderSide(
          color: Theme.of(context).colorScheme.outline.withOpacity(0
          width: 1,
        ),
      ),
    ),
    child: SafeArea(
      child: Column(
        children: [
          const SizedBox(height: AppDimensions.space8),
          // Logo section
          Padding(
            padding: const EdgeInsets.symmetric(
              horizontal: AppDimensions.space6,
            ),
            child: Text(
              'EntryTestGuru',
              style: Theme.of(context).textTheme.headlineMedium?.cop
                fontWeight: FontWeight.bold,
                color: Theme.of(context).colorScheme.primary,
              ),
            ),
          ),
          const SizedBox(height: AppDimensions.space8),
          // Navigation items
          Expanded(
            child: ListView.builder(
              padding: const EdgeInsets.symmetric(
                horizontal: AppDimensions.space4,
              ),
```

```
                itemCount: items.length,
                itemBuilder: (context, index) {
                  final item = items[index];
                  final isActive = index == currentIndex;

                  return Padding(
                    padding: const EdgeInsets.only(
                      bottom: AppDimensions.space2,
                    ),
                    child: _buildNavItem(
                      context: context,
                      item: item,
                      isActive: isActive,
                      onTap: () => onIndexChanged(index),
                      isMobile: false,
                    ),
                  );
                },
              ),
            ),
          ],
        ),
      ),
    );
  }

  Widget _buildNavItem({
    required BuildContext context,
    required NavigationItem item,
    required bool isActive,
    required VoidCallback onTap,
    required bool isMobile,
  }) {
    return Material(
      color: Colors.transparent,
      child: InkWell(
        onTap: onTap,
        borderRadius: BorderRadius.circular(AppDimensions.radiusMedium
        child: Container(
          constraints: BoxConstraints(
            minHeight: AppDimensions.minTouchTarget,
            minWidth: isMobile ? AppDimensions.minTouchTarget : double
          ),
          padding: EdgeInsets.symmetric(
            horizontal: isMobile ? AppDimensions.space2 : AppDimension
            vertical: AppDimensions.space3,
          ),
```

```
decoration: BoxDecoration(
  color: isActive
      ? Theme.of(context).colorScheme.primary.withOpacity(0.
      : Colors.transparent,
  borderRadius: BorderRadius.circular(AppDimensions.radiusMe
),
child: isMobile
    ? Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          AcademicIcon(
            type: item.iconType,
            isActive: isActive,
            color: isActive
                ? Theme.of(context).colorScheme.primary
                : Theme.of(context).colorScheme.onSurface,
          ),
          const SizedBox(height: AppDimensions.space1),
          Text(
            item.label,
            style: Theme.of(context).textTheme.labelSmall?.c
              color: isActive
                  ? Theme.of(context).colorScheme.primary
                  : Theme.of(context).colorScheme.onSurface,
              fontWeight: isActive ? FontWeight.w600 : FontW
            ),
            textAlign: TextAlign.center,
          ),
        ],
      )
    : Row(
        children: [
          AcademicIcon(
            type: item.iconType,
            isActive: isActive,
            color: isActive
                ? Theme.of(context).colorScheme.primary
                : Theme.of(context).colorScheme.onSurface,
          ),
          const SizedBox(width: AppDimensions.space3),
          Expanded(
            child: Text(
              item.label,
              style: Theme.of(context).textTheme.bodyMedium?
                color: isActive
                    ? Theme.of(context).colorScheme.primary
                    : Theme.of(context).colorScheme.onSurfac
```

```
                        fontWeight: isActive ? FontWeight.w600 : Fon
                      ),
                    ),
                  ),
                ],
              ),
          ),
        ),
      );
    }
  }

  class NavigationItem {
    final String label;
    final AcademicIconType iconType;

    const NavigationItem({
      required this.label,
      required this.iconType,
    });
  }
```

9. Theme Provider (Riverpod Implementation)

Theme State Management

```
// lib/providers/theme_provider.dart
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:shared_preferences/shared_preferences.dart';

class ThemeNotifier extends StateNotifier<ThemeMode> {
  ThemeNotifier() : super(ThemeMode.system) {
    _loadTheme();
  }

  static const String _themeKey = 'app_theme_mode';

  Future<void> _loadTheme() async {
    final prefs = await SharedPreferences.getInstance();
    final themeIndex = prefs.getInt(_themeKey) ?? 0;
    state = ThemeMode.values[themeIndex];
  }
```

```dart
  Future<void> setTheme(ThemeMode themeMode) async {
    state = themeMode;
    final prefs = await SharedPreferences.getInstance();
    await prefs.setInt(_themeKey, themeMode.index);
  }

  Future<void> toggleTheme() async {
    final newTheme = state == ThemeMode.dark
        ? ThemeMode.light
        : ThemeMode.dark;
    await setTheme(newTheme);
  }

  bool get isDarkMode => state == ThemeMode.dark;
  bool get isLightMode => state == ThemeMode.light;
  bool get isSystemMode => state == ThemeMode.system;
}

final themeProvider = StateNotifierProvider<ThemeNotifier, ThemeMode>(
  (ref) => ThemeNotifier(),
);
```

10. Usage Examples

Complete App Setup

```dart
// lib/main.dart
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'core/theme/app_theme.dart';
import 'providers/theme_provider.dart';
import 'widgets/theme_switcher.dart';

void main() {
  runApp(
    const ProviderScope(
      child: EntryTestGuruApp(),
    ),
  );
}

class EntryTestGuruApp extends ConsumerWidget {
  const EntryTestGuruApp({super.key});
```

```
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final themeMode = ref.watch(themeProvider);

    return MaterialApp(
      title: 'EntryTestGuru',
      theme: AppTheme.lightTheme,
      darkTheme: AppTheme.darkTheme,
      themeMode: themeMode,
      home: const MainScreen(),
      builder: (context, child) {
        return Stack(
          children: [
            child!,
            const ThemeSwitcher(),
          ],
        );
      },
    );
  }
}
```

Practice Screen Example

```
// lib/screens/practice_screen.dart
import 'package:flutter/material.dart';
import '../widgets/app_card.dart';
import '../widgets/app_button.dart';
import '../widgets/arde_badge.dart';
import '../core/theme/app_dimensions.dart';
import '../core/theme/app_text_styles.dart';
import '../core/utils/responsive_utils.dart';

class PracticeScreen extends StatelessWidget {
  const PracticeScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Padding(
          padding: ResponsiveUtils.getScreenPadding(context),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
```

```
// Header
Text(
  'Practice Mode',
  style: AppTextStyles.displayMedium.copyWith(
    color: Theme.of(context).colorScheme.onSurface,
  ),
),
const SizedBox(height: AppDimensions.space6),

// Question Card
Expanded(
  child: AppCard(
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        // Question header with ARDE badge
        Row(
          children: [
            Expanded(
              child: Text(
                'Question 5 of 20',
                style: AppTextStyles.labelLarge.copyWith
                  color: Theme.of(context).colorScheme.o
                ),
              ),
            ),
            const ArdeBadge(probability: ArdeProbability
          ],
        ),
        const SizedBox(height: AppDimensions.space6),

        // Question text
        Text(
          'Which of the following is the correct formula
          style: AppTextStyles.questionText.copyWith(
            color: Theme.of(context).colorScheme.onSurfa
          ),
        ),
        const SizedBox(height: AppDimensions.space8),

        // MCQ Options
        Expanded(
          child: Column(
            children: [
              _buildMCQOption(context, 'A', 'KE = ½mv²',
              _buildMCQOption(context, 'B', 'KE = mv²',
              _buildMCQOption(context, 'C', 'KE = ½m²v',
```

```
                _buildMCQOption(context, 'D', 'KE = 2mv²',
              ],
            ),
          ),

          // Action buttons
          Row(
            children: [
              Expanded(
                child: AppButton(
                  text: 'Skip Question',
                  type: ButtonType.outline,
                  onPressed: () {},
                ),
              ),
              const SizedBox(width: AppDimensions.space4),
              Expanded(
                flex: 2,
                child: AppButton(
                  text: 'Submit Answer',
                  type: ButtonType.primary,
                  onPressed: () {},
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  ),
  ),
  ),
  ),
  ),
  ),
  );
}

Widget _buildMCQOption(
  BuildContext context,
  String option,
  String text,
  bool isSelected,
) {
  return Padding(
    padding: const EdgeInsets.only(bottom: AppDimensions.space3),
    child: Material(
      color: Colors.transparent,
```

```
child: InkWell(
  onTap: () {},
  borderRadius: BorderRadius.circular(AppDimensions.radiusMedi
  child: Container(
    width: double.infinity,
    constraints: const BoxConstraints(
      minHeight: AppDimensions.minTouchTarget,
    ),
    padding: const EdgeInsets.all(AppDimensions.space4),
    decoration: BoxDecoration(
      border: Border.all(
        color: isSelected
            ? Theme.of(context).colorScheme.primary
            : Theme.of(context).colorScheme.outline.withOpacit
        width: isSelected ? 2 : 1,
      ),
      borderRadius: BorderRadius.circular(AppDimensions.radius
      color: isSelected
          ? Theme.of(context).colorScheme.primary.withOpacity(
          : Colors.transparent,
    ),
    child: Row(
      children: [
        Container(
          width: 32,
          height: 32,
          decoration: BoxDecoration(
            shape: BoxShape.circle,
            color: isSelected
                ? Theme.of(context).colorScheme.primary
                : Colors.transparent,
            border: Border.all(
              color: isSelected
                  ? Theme.of(context).colorScheme.primary
                  : Theme.of(context).colorScheme.outline,
              width: 2,
            ),
          ),
          child: Center(
            child: Text(
              option,
              style: AppTextStyles.labelMedium.copyWith(
                color: isSelected
                    ? Colors.white
                    : Theme.of(context).colorScheme.onSurface,
                fontWeight: FontWeight.w600,
              ),
```

```
                    ),
                  ),
                ),
                const SizedBox(width: AppDimensions.space4),
                Expanded(
                  child: Text(
                    text,
                    style: AppTextStyles.mcqOption.copyWith(
                      color: Theme.of(context).colorScheme.onSurface,
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

11. Dependencies Required

pubspec.yaml additions

```yaml
dependencies:
  flutter:
    sdk: flutter

  # State Management
  flutter_riverpod: ^2.4.9

  # Fonts & Icons
  google_fonts: ^6.1.0
  flutter_svg: ^2.0.9

  # UI & Animations
  animations: ^2.0.8

  # Storage
  shared_preferences: ^2.2.2

  # Responsive
  flutter_screenutil: ^5.9.0
```

```
  # Accessibility
  flutter_tts: ^3.8.3
```

12. File Structure

```
lib/
├── core/
│   ├── theme/
│   │   ├── app_colors.dart
│   │   ├── app_theme.dart
│   │   ├── app_text_styles.dart
│   │   ├── app_dimensions.dart
│   │   └── app_animations.dart
│   ├── utils/
│   │   └── responsive_utils.dart
│   └── accessibility/
│       └── focus_helper.dart
├── widgets/
│   ├── theme_switcher.dart
│   ├── custom_buttons.dart
│   ├── arde_badge.dart
│   ├── app_card.dart
│   ├── academic_icon.dart
│   ├── app_navigation.dart
│   ├── responsive_builder.dart
│   ├── animated_feedback.dart
│   └── accessible_text.dart
├── providers/
│   └── theme_provider.dart
├── screens/
│   └── practice_screen.dart
└── main.dart
```

This Flutter-specific style guide provides:

✅ **Native Flutter implementation** using Material 3 design system
✅ **Dark/Light theme support** with persistent storage
✅ **Always-accessible theme switcher** via fixed position widget
✅ **User tier differentiation** through color variations
✅ **ARDE probability badges** with Flutter widgets
✅ **Responsive design** using LayoutBuilder and MediaQuery

- ✅ **WCAG 2.1 AA compliance** with proper focus management
- ✅ **Academic iconography** using Material Icons with active/inactive states
- ✅ **Typography scale** using Google Fonts and TextTheme
- ✅ **Animation system** with duration and curve constants
- ✅ **Riverpod state management** for theme switching

The code is ready to use in your Flutter project and follows Flutter best practices for theming, accessibility, and responsive design.