Erply UI Package

A comprehensive Flutter UI package with advanced theme management.

Package Structure

```
erply_ui/
├── core/
│   └── typography/
│       ├── font_config.dart
│       └── typography_config.dart
├── theme/
│   ├── models/
│   │   ├── erply_theme_data.dart
│   │   └── theme_registry.dart
│   └── providers/
│       └── theme_provider.dart
└── components/
    ├── buttons/
    └── text_fields/
```

Features

- Flexible Theme Management
- Custom Theme Creation
- Typography Customization
- Custom Font Support
- Theme Registry
- Riverpod State Management
- Responsive Layout
- Custom SVG icon management

Installation

Add the following to your `pubspec.yaml`:

```yaml
dependencies:
  erply_ui: ^0.2.0
  flutter_riverpod: ^2.4.9
```

Usage

Theme Management

Registering Themes

```
// Create a custom theme
class CustomTheme extends ErplyThemeData {
  CustomTheme() : super(
    name: 'CustomTheme',
    typographyConfig: ErplyTypographyConfig(
      displayLarge: ErplyFontConfig(
        fontFamily: 'CustomFont',
        fontSize: 32,
        fontWeight: FontWeight.bold,
        color: Colors.purple,
      ),
    ),
    primaryColor: Colors.purple,
  );
}

// Register the theme (IMPORTANT: Must be done before using)
ErplyThemeRegistry.registerTheme(CustomTheme());
```

Changing Themes

```
class ThemeDemo extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final themeNotifier = ref.read(erplyThemeProvider.notifier);

    return Scaffold(
      body: Column(
        children: [
          // Change to a specific theme (light variant)
          ElevatedButton(
            onPressed: () async {
              await themeNotifier.setThemeByName(
                'CustomTheme',
                variant: ErplyThemeVariant.light
              );
            },
            child: Text('Switch to Light Custom Theme'),
          ),

          // Change to dark variant
          ElevatedButton(
            onPressed: () async {
              await themeNotifier.setThemeByName(
                'CustomTheme',
                variant: ErplyThemeVariant.dark
              );
```

```
            },
            child: Text('Switch to Dark Custom Theme'),
          ),
        ],
      ),
    );
  }
}
```

MaterialApp Configuration

```
runApp(
  ProviderScope(
    child: Consumer(
      builder: (context, ref, child) {
        final themeNotifier = ref.watch(erplyThemeProvider.notifier);

        return MaterialApp(
          theme: themeNotifier.getLightTheme(),
          darkTheme: themeNotifier.getDarkTheme(),
          // Automatically persists and restores theme
          themeMode: themeNotifier.currentVariant ==
ErplyThemeVariant.dark
            ? ThemeMode.dark
            : ThemeMode.light,
          home: MyHomePage(),
        );
      },
    ),
  ),
);
```

Theme Registry Methods

```
// Check if a theme is registered
bool isRegistered = ErplyThemeRegistry.hasTheme('CustomTheme');

// Get all registered theme names
List<String> themeNames = ErplyThemeRegistry.getThemeNames();

// Retrieve a specific theme
ErplyThemeData? theme = ErplyThemeRegistry.getTheme('CustomTheme');
```

Key Features

- Automatic theme persistence

- Fallback to default theme if registered theme is removed
- Support for light and dark variants
- Easy theme registration and management

Font Configuration

```
// Create a font configuration
final fontConfig = ErplyFontConfig(
  fontFamily: 'CustomFont',
  fontSize: 16,
  fontWeight: FontWeight.w400,
  color: Colors.black87,
  letterSpacing: 0.5,
  height: 1.5,
);
```

Adding Custom Fonts

1. Add your font to `assets/fonts/` in your project
2. Update `pubspec.yaml`:

```
flutter:
  fonts:
    - family: CustomFont
      fonts:
        - asset: assets/fonts/CustomFont-Regular.ttf
        - asset: assets/fonts/CustomFont-Bold.ttf
          weight: 700
```

Components

Text Fields

```
ErplyTextField(
  labelText: 'Email',
  hintText: 'Enter your email',
)
```

Buttons

```
ErplyButton(
  text: 'Submit',
```

```
    onPressed: () {},
  )
```

Responsive Layout

Features

- Adaptive navigation between NavigationRail and Drawer
- Responsive design for different screen sizes
- Customizable navigation items
- Support for extended navigation rail on wider screens

Basic Usage

```dart
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ErplyResponsiveLayout(
      title: 'My App',
      selectedIndex: 0,
      navigationItems: [
        ErplyNavigationItem(
          icon: Icons.home,
          label: 'Home',
          onTap: () {
            // Navigate or perform action
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => HomePage())
            );
          },
        ),
        ErplyNavigationItem(
          icon: Icons.settings,
          label: 'Settings',
          onTap: () {
            // Navigate or perform action
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => SettingsPage())
            );
          },
        ),
      ],
      content: Center(
        child: Text('Main Content Area'),
      ),
    );
  }
}
```

Responsive Behavior

- On screens wider than 600px: NavigationRail on the left
- On screens narrower than 600px: Drawer navigation
- NavigationRail extends to show labels on wider screens (>800px)

Customization Options

```
ErplyResponsiveLayout(
  // Custom app bar title
  title: 'Custom Title',

  // Optional app bar actions
  actions: [
    IconButton(
      icon: Icon(Icons.search),
      onPressed: () {/* Search action */},
    ),
  ],

  // Custom background color
  backgroundColor: Colors.grey[100],

  // Initial selected navigation index
  selectedIndex: 1,

  // Navigation items with custom icons and actions
  navigationItems: [
    ErplyNavigationItem(
      icon: Icons.dashboard,
      selectedIcon: Icon(Icons.dashboard, color: Colors.blue),
      label: 'Dashboard',
      onTap: () {/* Dashboard action */},
    ),
    // More navigation items...
  ],

  // Main content
  content: MyCustomContentWidget(),
)
```

Navigation Item Configuration

```
ErplyNavigationItem(
  // Primary icon
  icon: Icons.home,
```

```
  // Optional selected state icon
  selectedIcon: Icon(Icons.home, color: Colors.blue),

  // Label displayed next to or under the icon
  label: 'Home',

  // Callback when the navigation item is tapped
  onTap: () {
    // Navigate or perform an action
  },
)
```

Responsive Design Considerations

- Automatically adapts to screen width
- Provides a consistent navigation experience across devices
- Minimizes layout changes while maintaining usability

Erply Icons

Features

- Custom SVG icon management
- Easy icon loading
- Color and size customization
- Asset-based icon system

Basic Usage

```
// Using ErplyIcons class
SvgPicture dashboardIcon = ErplyIcons.svg(
  'dashboard',
  width: 24,
  height: 24,
  color: Colors.blue
);

// Using string extension
SvgPicture profileIcon = 'profile'.toErplyIcon(
  width: 32,
  height: 32,
  color: Colors.green
);
```

Icon Management

```dart
// Check if an icon exists
bool hasDashboardIcon = ErplyIcons.hasIcon('dashboard');

// Get list of available icons
List<String> availableIcons = ErplyIcons.availableIcons;
```

Error Handling

```dart
try {
  // Throws an error if icon doesn't exist
  SvgPicture nonExistentIcon = 'nonexistent'.toErplyIcon();
} catch (e) {
  print('Icon not found: $e');
}
```

Integration with Widgets

```dart
// In a button or other widget
ElevatedButton(
  child: Row(
    children: [
      'dashboard'.toErplyIcon(width: 16),
      Text('Dashboard'),
    ],
  ),
  onPressed: () {},
)
```

Adding Custom Icons

1. Place SVG icons in `assets/icons/` directory
2. Update `availableIcons` in `erply_icons.dart`
3. Run `flutter pub get`

Design Considerations

- Lightweight SVG icon management
- Easy to extend and customize
- Consistent with Material Design principles

Contributing

Contributions are welcome! Please submit a Pull Request.

License

[Add your license here]