Implement a reusable, production-ready SwiftUI control named ButterflyButton that acts like a boolean toggle with a medallion that spins around an axle. The control must be cross-platform (iOS, macOS, tvOS, visionOS), localizable, accessible, themable, and packaged as a Swift Package. Defaults follow Apple HIG and system colors.

- As a short-hand in this document, BB = ButterflyButton.
- Follow the Apple Human Interface Guidelines (HIG) for all design decisions: https://developer.apple.com/design/human-interface-guidelines/
- Framework: SuiteUI-native.
- .Platform minimums: iOS 17+, macOS 14+, tvOS 17+, visionOS 1.0+.
- Resources: include a Localizable.strings base file and platform variants.
- Documentation: Marked-up doc comments and a README.md with usage and previews.
- Testing: Use the Swift Testing framework for unit tests.
- Testing should include a visual preview of the control.
- The control should auto-adapt its size, colors, and typography to match Dynamic Type and system themes.
- BB is a universal component that can be used on iOS, MacOS, tvOS, and VisionOS.
  - WatchOS is not supported.
- BB is a graphical control that is contained in it's own Swift Package.
- BB adheres to the UIKit UIColorWell and default to system colors.
- BB is localizable.
- BB supports accessibility features.
- BB is a subclass of UIControl.

- BB state is boolean: true or false.
- BB is always a square with the default frame size of 60x60 pt and can be no smaller than 44x44 pt (per HMI best practices for Button).
- The frame size can be set as a property.
- When properties are set, the visual presentation of the BB updates immediately if necessary.

Visual Elements

- BB has 4 visual elements.
  - The **Mount** is a square that represents the outside shape of the control.
    - The Mount is a container for the graphical elements of the BB.
    - The Mount color can be set as a property.
    - The Mount thickness can be set as a property and defaults to 2 pt. The thickness setting must be at least 1 pt and can be no more than one-tenth (1⁄10) of the side length.
    - The Mount gives the visual representation of having depth within its boundaries.
    - The Mount background can be set to a color and defaults to the system color palette.
    - The Mount background can also be transparent.
    - The Mount background can be set to an image.
    - The Mount labels and images can use SF Symbols and images from asset catalogs.
  - BB has a **Label** which is a SwiftUI Label object that is the title of the button.

- The Label is on the outside of the Mount can be on the top, left, right, or bottom.
    - The default position is based on the top.
    - The label has no border.
    - The label has padding defaulted to 2 pt.
    - The label's fonts, sizes, and colors can be customized.
- BB has an **Axle**.
    - The axle is a narrow gauge cylinder that traverses the frame from one point to another.
        - The axle can be centered horizontally.
        - The axle can be centered vertically.
        - The axle can be diagonal either top-left to bottom-right or top-right to bottom-left.
        - The axel color can be set.
        - The axel has no other properties.
- BB has a **Medallion** which is a circular disk (like a coin) through which the axel passes.
    - For convention, the terms 'top' and 'bottom' will be used to describe the two sides of the medallion.
    - The medallion diameter is always 90% of the interior frame of the Mount.
    - The medallion thickness is always 2 pt.
    - The medallion has distinct colors for each side: top and bottom.
    - The medallion has an edge color for the circumference of the medallion.

        - The medallion can display an image for the top side.
        - The medallion can display a second image for the bottom side.
        - Each side of the medallion has a label (UILabel) that is contained within its circumference.
        - The top label defaults to “true.”
        - The bottom label defaults to “false.”
    - Default colors should automatically adapt to light/dark mode.
    - If no color is provided, the system should automatically pick tintColor.

Accessibility & Localization

- The title Label, top Label, and bottom Label support localization using the built-in Localizable.strings.
- For VoiceOver, the control announces it’s title and current state (“true” or “false”).

Behavior & Gestures

- The medallion responds to a click, touch, tap, or flick motion and raises an event for that action.
    - The animation uses SwiftUI’s withAnimation API.
    - The animation uses simulated realistic physics (e.g., spring damping, inertia, etc.).
    - The medallion has an animated spinning motion that starts off quick (based on the momentum of the flick if that method was used) and slows down.
        - The animation spins top to bottom along the axel if the triggering event is ends on the top half.

    - The animation spins bottom to top along the axel if the triggering event is ends on the bottom half.
  - The rate at which the medallions speed slows can be set. The default is 2 seconds.
    - The rate cannot be 0.
  - If the medallion is showing the top side, when the spinning animation completes, it then shows the bottom side.
- On tvOS, a Siri Remote swipe would be treated as a flick.
- On visionOS, the flick motion uses a swipe gesture.
- If the medallion is showing the bottom side, when the spinning animation completes, it then shows the top side.
- The BB provides an event when the medallion spin is concludes with the property of the current value of the BB.
- Support the boolean isOn: bool @Binding.
- Provide a delegate callback function didChangeState(to:).

Error Handling & Logging

- The BB will support logging of events using os.Logger (Unified Logging System).
- The BB will support all inherited functions and properties unless they conflict with the above specifications.
  - If BB cannot support a function or property an appropriate error will be provided.

Primary API Surface

```
public struct ButterflyButton: View {
  @Binding public var isOn: Bool
```

```
  // Sizing
  public var sideLength: CGFloat = 60   // min 44
  public var labelPlacement: LabelPlacement = .auto

  // Mount
  public var mountStrokeColor: Color? = nil         // defaults to
system separator/tint-adaptive
  public var mountStrokeWidth: CGFloat = 2          // clamped: [1,
sideLength * 0.10]
  public var mountBackground: MountBackground = .systemAutomatic //
.systemAutomatic | .transparent | .color(Color) | .image(Image)

  // Axle
  public var axleOrientation: AxleOrientation
= .horizontal // .horizontal | .vertical | .diagonalLTR
| .diagonalRTL
  public var axleColor: Color? = nil // defaults to tinted adaptive

  // Medallion
  public var medallionTopColor: Color? = nil       // defaults
adaptive
  public var medallionBottomColor: Color? = nil    // defaults
adaptive
  public var medallionEdgeColor: Color? = nil      // defaults
adaptive
  public var medallionTopImage: Image? = nil
  public var medallionBottomImage: Image? = nil
  public var medallionTopLabel: LocalizedStringKey = "true"
  public var medallionBottomLabel: LocalizedStringKey = "false"
  public var medallionLabelFont: Font? = nil       // defaults
to .caption (scales with Dynamic Type)
  public var medallionLabelColor: Color? = nil     // defaults
automatic for contrast
```

```
  public var medallionStrokeWidth: CGFloat = 2     // constant by
spec


  // Animation
  public var spinDecelerationDuration: TimeInterval = 2.0 // > 0.0
  public var enableFlickPhysics: Bool = true              // drag
velocity -> initial angular velocity
  public var hapticsEnabled: Bool = true                  // where
supported


  // Events
  public var onSpinBegan: (() -> Void)? = nil
  public var onSpinCompleted: ((_ isOn: Bool) -> Void)? = nil


  public init(isOn: Binding<Bool>, /* all others with defaults */)
}


public enum LabelPlacement { case top, bottom, leading, trailing,
auto }
public enum AxleOrientation { case horizontal, vertical,
diagonalLTR, diagonalRTL }
public enum MountBackground { case systemAutomatic, transparent,
color(Color), image(Image) }


public init(
  isOn: Binding<Bool>,
  @ViewBuilder label: () -> some View,
  /* other params */
)
```

Example Usage

```
struct ContentView: View {
  @State private var isOn = false

  var body: some View {
    ButterflyButton(
      isOn: $isOn,
      sideLength: 60,
      labelPlacement: .auto,
      mountStrokeColor: nil,
      mountStrokeWidth: 2,
      mountBackground: .systemAutomatic,
      axleOrientation: .horizontal,
      axleColor: nil,
      medallionTopColor: nil,
      medallionBottomColor: nil,
      medallionEdgeColor: nil,
      medallionTopImage: Image(systemName: "sun.max.fill"),
      medallionBottomImage: Image(systemName: "moon.fill"),
      medallionTopLabel: "true",
      medallionBottomLabel: "false",
      spinDecelerationDuration: 2.0,
      enableFlickPhysics: true,
      hapticsEnabled: true,
      onSpinBegan: { /* optional */ },
      onSpinCompleted: { newValue in
        // handle state change
      }
    ) {
```

```
      Text("Butterfly")
        .font(.callout)
    }
  }
}
```

Design Notes

Layout & Sizing Rules

- The control is always square. sideLength defaults to 60 pt; clamp to ≥ 44 pt (HIG minimum).
- Outer label sits outside the square “Mount” with 2 pt padding; placement: .auto = uses language direction (leading for LTR languages, trailing for RTL).
- Mount is the outer square container with a stroked border (mountStrokeWidth) and optional background (system adaptive, transparent, color, or image). Stroke width is clamped to [1, sideLength * 0.10].
  - Axle is a narrow line (cylinder metaphor) that spans the mount interior along axleOrientation.
  - Medallion is a circle centered on the axle; diameter = 90% of the mount’s inner square (inside the stroke). Stroke (edge) width is 2 pt.

Visual Defaults & Theming

- If any color is nil, derive an adaptive color palette from environment: tint, secondary, and background that passes contrast checks in light/dark.
- System-aware: respects Light/Dark mode, Increased Contrast, Reduce Motion (see Animations), and Dynamic Type (labels scale).

- Images: Support Image sources including SF Symbols; scale to fit within medallion with content insets to preserve label legibility.

Interaction & State

- Acts as a toggle. Tapping/clicking the medallion toggles isOn.
- Gestures:
  - Tap/Click: triggers spin animation and toggles.
  - Flick/Drag: if enableFlickPhysics == true, compute initial angular velocity from drag velocity projected onto axle axis; direction is determined by where the gesture ends relative to axle:
- End in top half → spin top→bottom.
- End in bottom half → spin bottom→top.
- On completion, state flips: showing top → end showing bottom; showing bottom → end showing top. Update isOn accordingly and invoke onSpinCompleted(isOn).

Animation

- Use SwiftUI animations with a physics-like ease-out:
- Default: an interpolating spring tuned to emulate quick start → decelerate over spinDecelerationDuration.
- Respect Reduce Motion: if enabled, skip spinning and apply a brief fade/scale to indicate change (no rotation).
- spinDecelerationDuration must be > 0. If set to 0 or negative, log an error and fall back to default (2.0s).

Accessibility

- Expose as a Toggle semantic:

- accessibilityRole(.toggle)
- accessibilityValue(isOn ? medallionTopLabel : medallionBottomLabel)
- Combined label uses outer label + current state (localized).
- Large content viewer supported on iOS/iPadOS; text scales via Dynamic Type.
- Hit target ≥ 44×44 pt regardless of visuals (expand contentShape if needed).

Localization

- Provide default localized strings for "true" / "false" (keys: ButterflyButton.true, ButterflyButton.false). Developers can override via initializer.
- Outer label content is a developer-supplied Text/LocalizedStringKey passed via a standard label: init overload:

```
public init(
  isOn: Binding<Bool>,
  @ViewBuilder label: () -> some View,
  /* other params */
)
```

tvOS & visionOS Input

- tvOS: Treat Siri Remote swipe along axle as flick; click as tap.
- visionOS: Support tap and pinch as activation; gaze focus highlights the medallion; maintain the same spin rules.

Error Handling & Logging

- Use Logger (os.Logger) with subsystem com.yourorg.ButterflyButton.

- Log non-fatal issues (e.g., invalid spinDecelerationDuration, mountStrokeWidth clamping).
- No thrown errors in the view layer; fail-safe to defaults.

Previews & Tests

- SwiftUI previews with multiple configurations:
- Light/Dark, Increased Contrast, Reduced Motion, RTL, Dynamic Type sizes.
- Axle orientations, image vs color medallions, boundary sizes (44, 60, 120).

Unit/UI tests:

- Layout constraints (size clamping, medallion = 90% rule).
- Gesture → state transitions and callback firing.
- Accessibility roles, values, and hit testing.