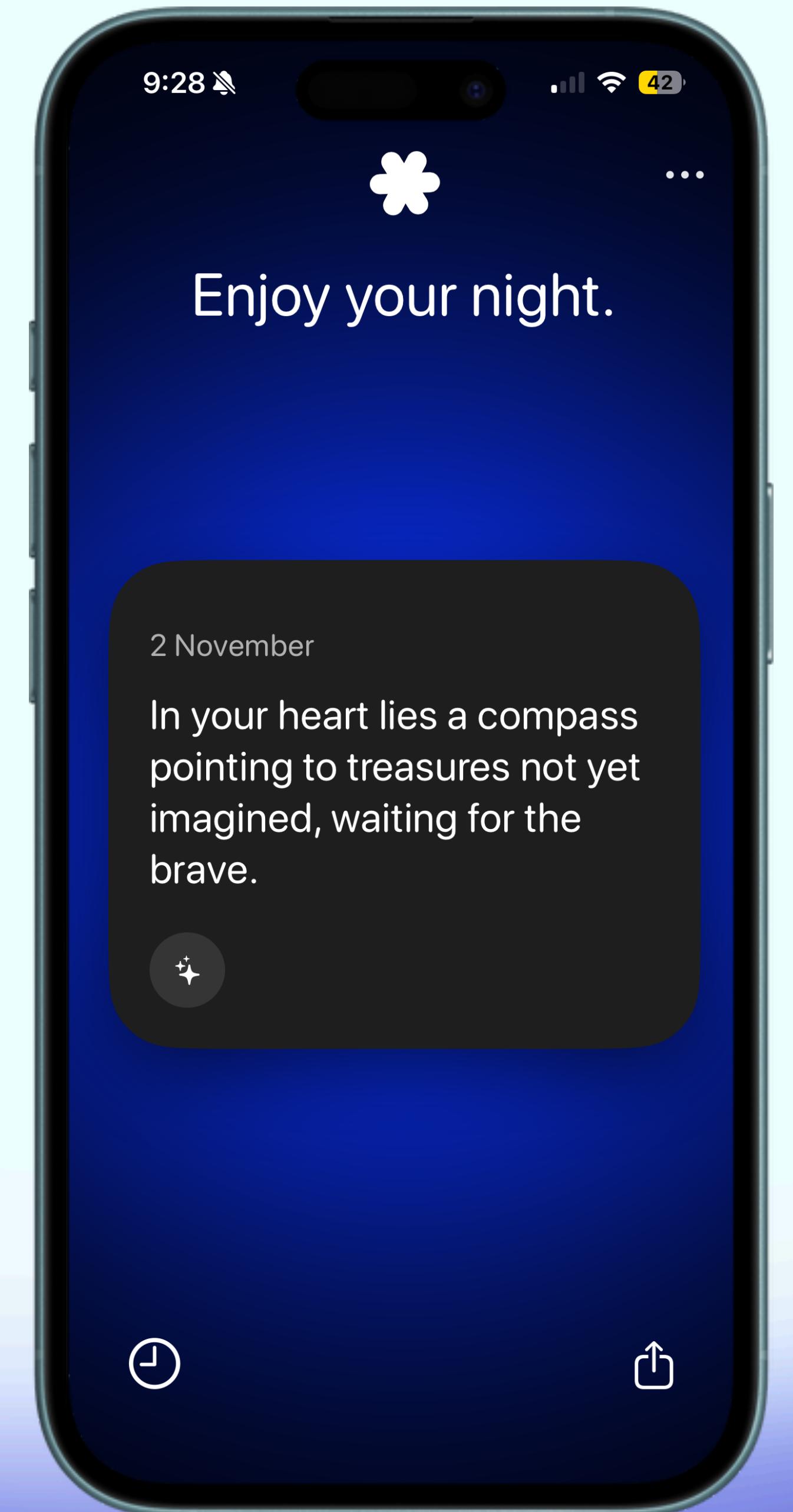
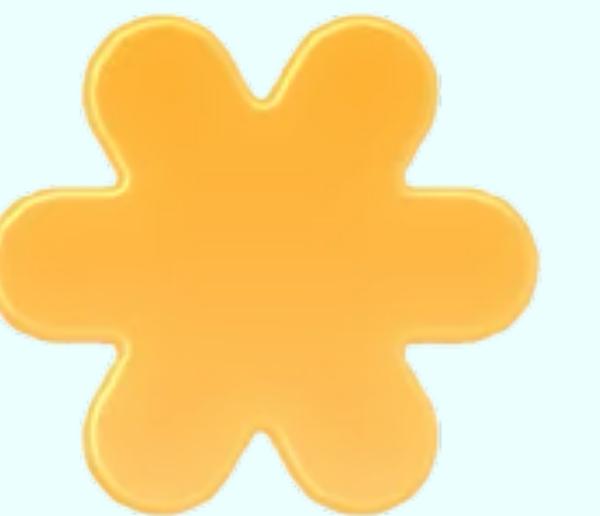
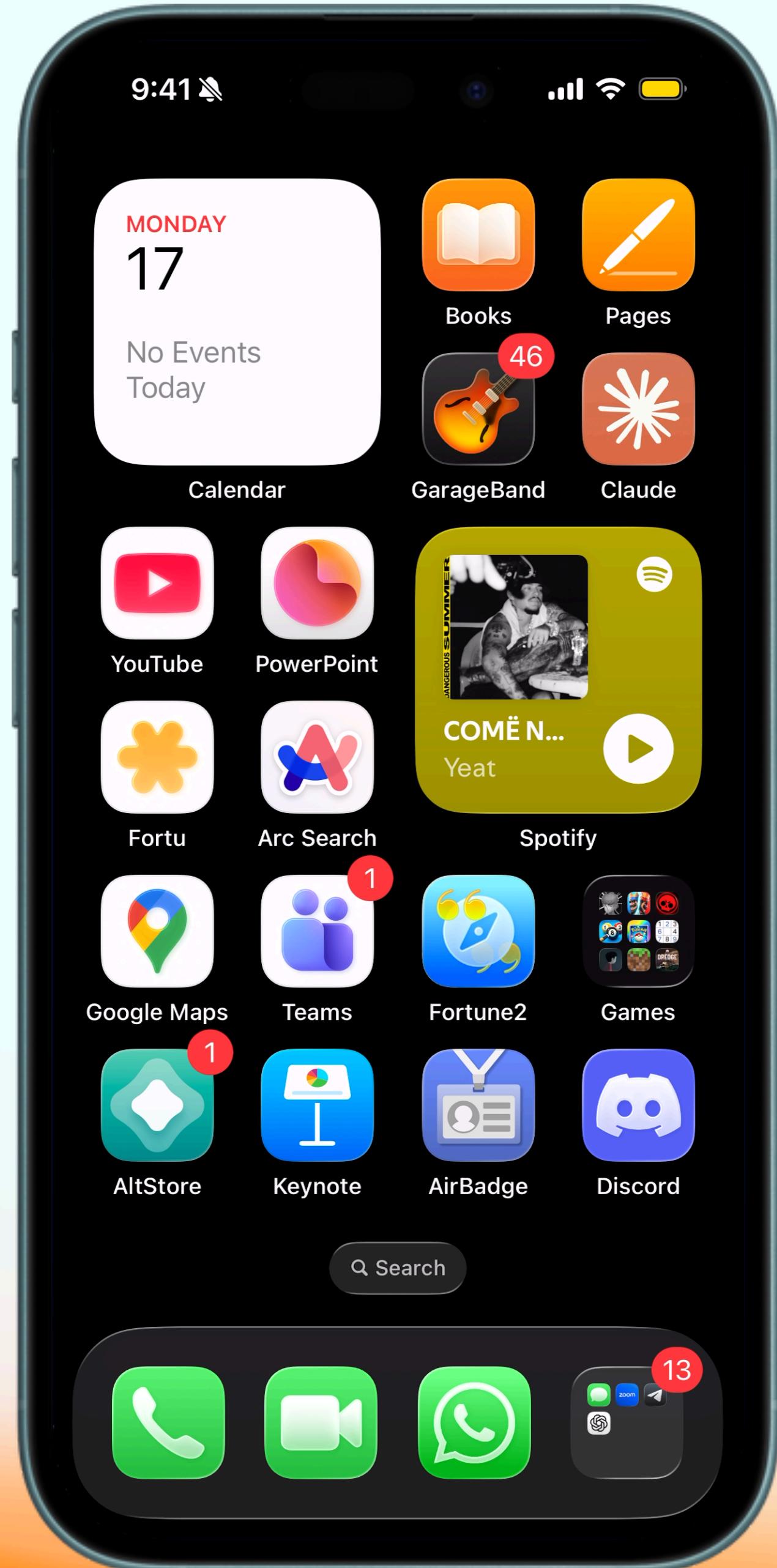


**In your heart lies a compass
pointing to treasures not yet
imagined, waiting for the brave.**



**In your heart lies a compass
pointing to treasures not yet
imagined, waiting for the brave.**





Fortu

Single-screen
Appealing UI
Simplicity



UI design (gradients, time-based color updates, animations)

AI models implementation for quote generation and interaction with user.

Opted for Apple's FoundationModels

Modal views for chat & history views (lists, text, shapes, buttons)

🌟 Quote Assistant

Sound & Haptics

Learning goals

UI design

Interactive
dynamics

Haptics

LLM's

The learning process



The learning process

Daily quotes

🌟 Quote Companion

2:20



52



Good afternoon

5 November

Even the heaviest clouds must part to reveal the sun, trust your journey.



The learning process

Daily quotes

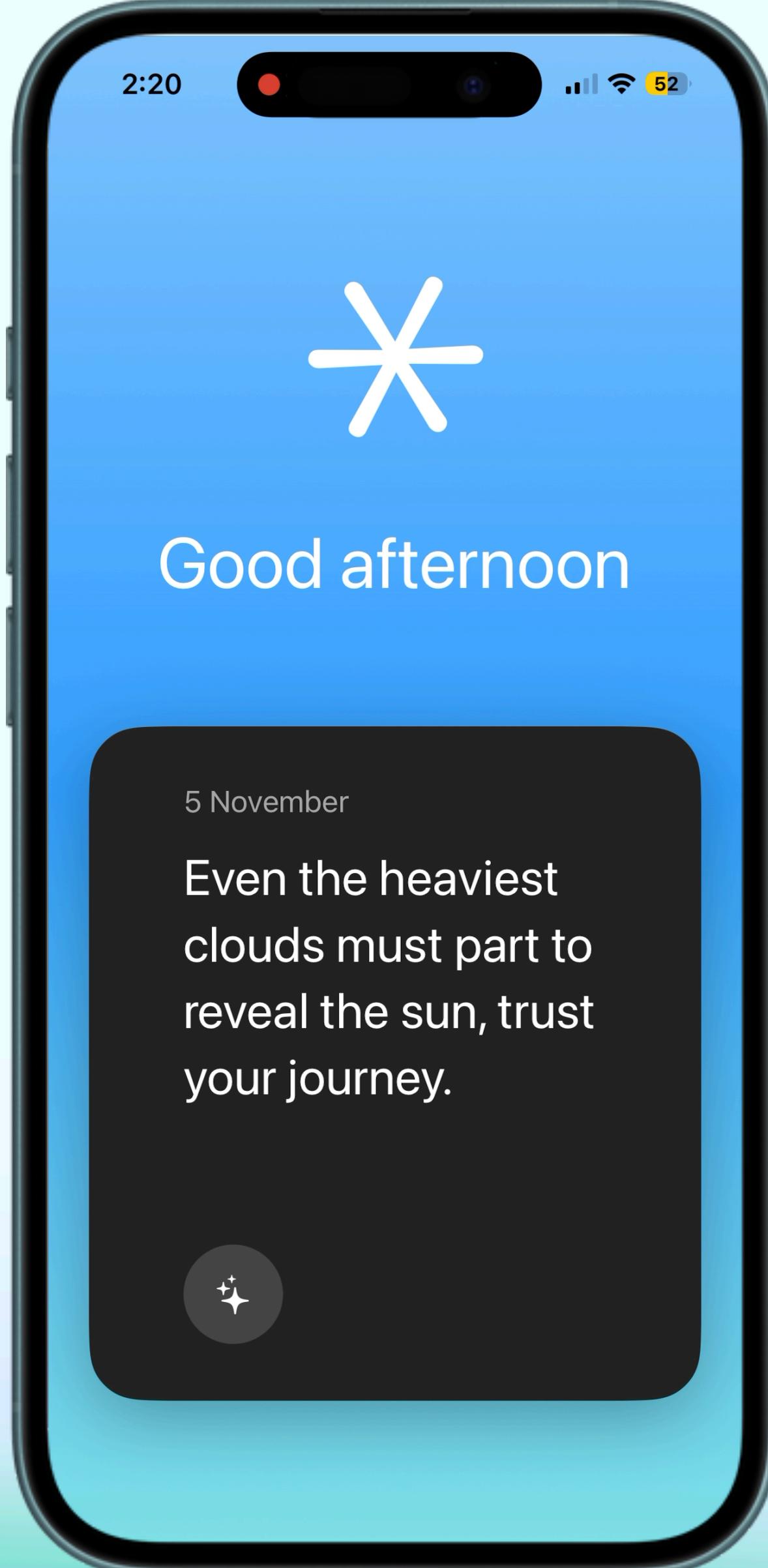
I built a simple LLM engine based on Apple's FoundationModels. It provides a different quote every day, through a dedicated QuoteProvider.

Quote Companion

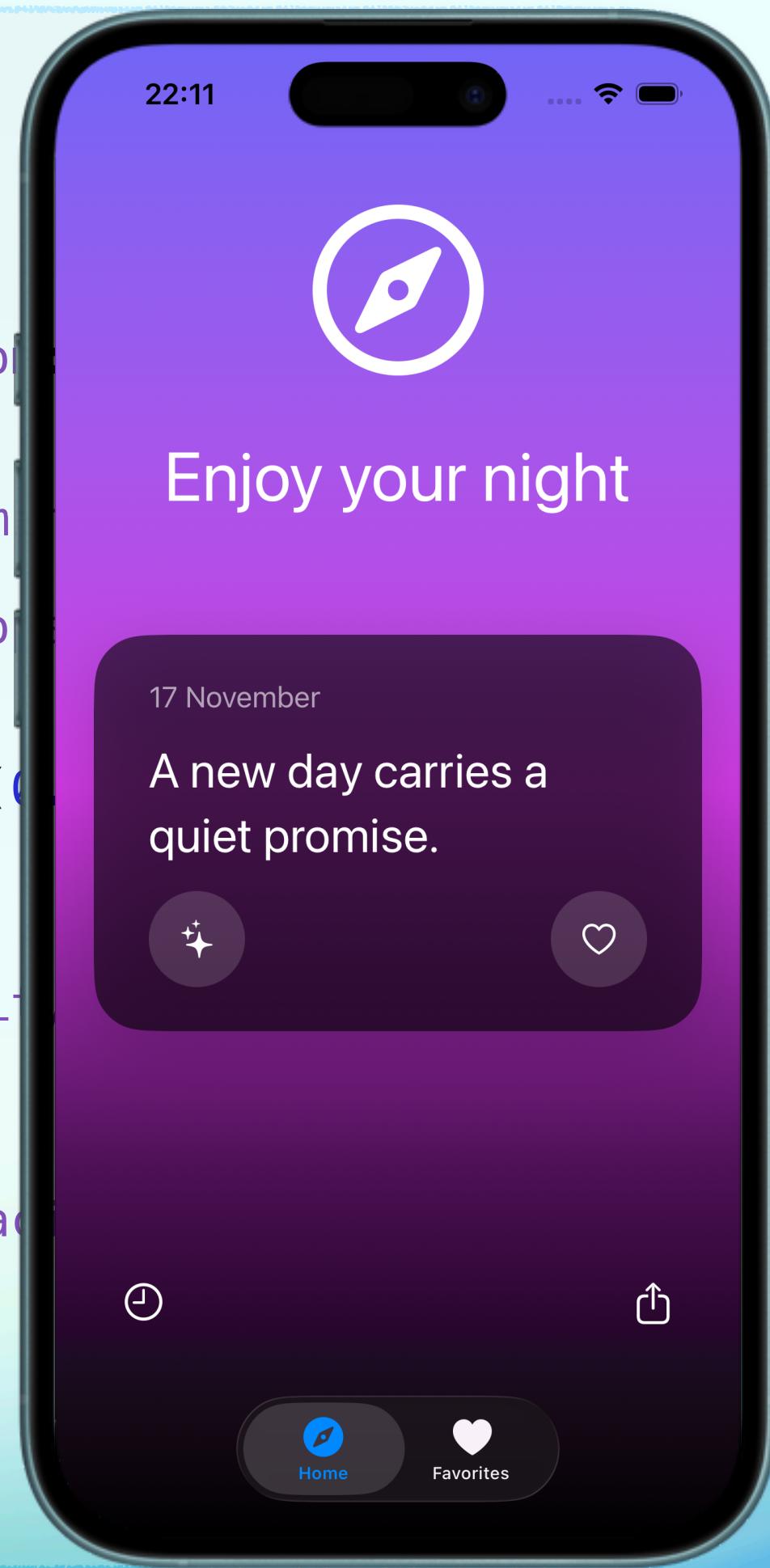
The sparkles button triggers a chatView that will show up as a modal.

The Assistant provides more information about the quote of the day, and can respond to User's inputs through a ChatSessionManager.

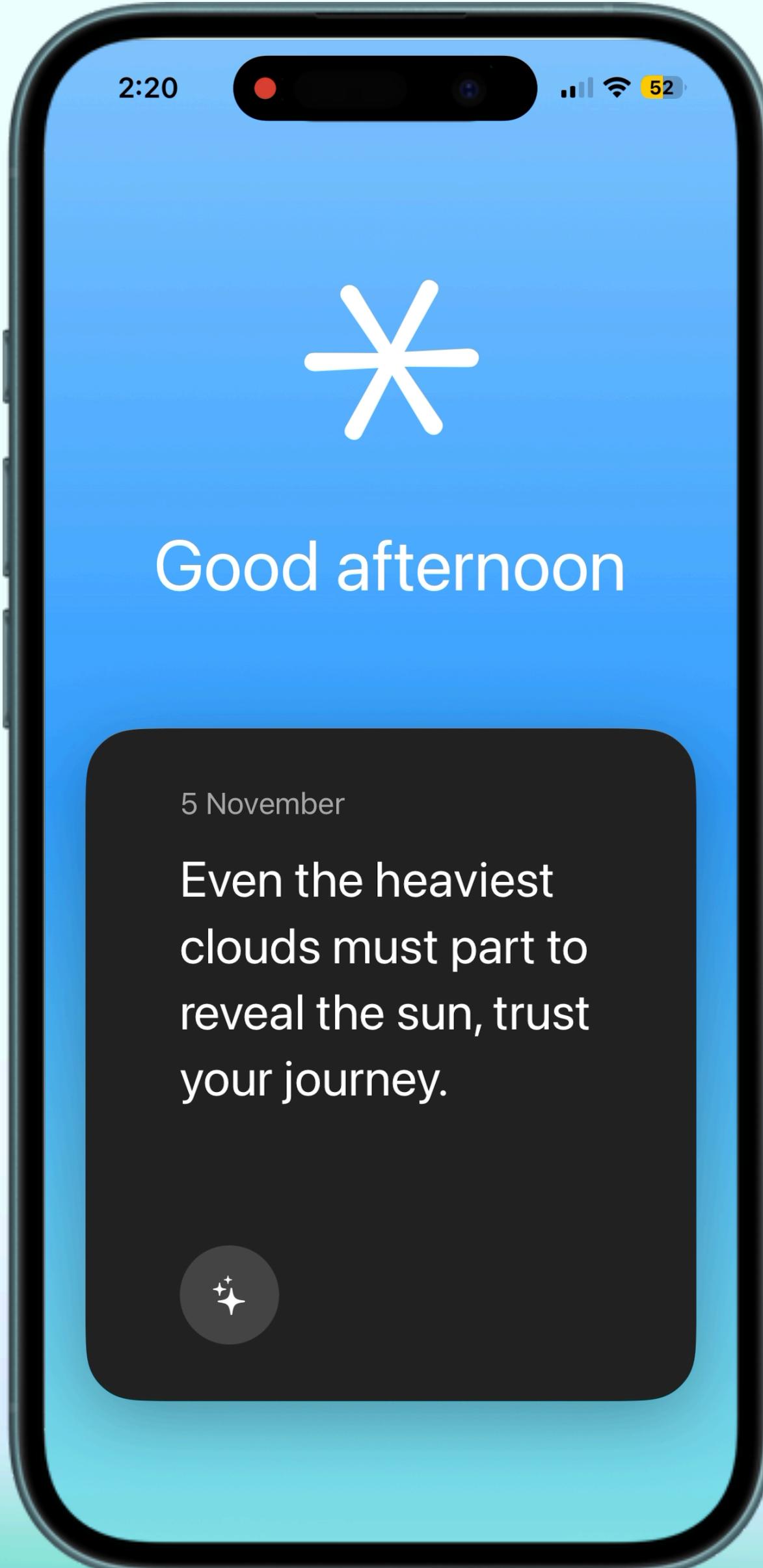
The learning process



```
class BackgroundGradientEngine: ObservableObject {  
    @Published var colors: [Color] = []  
  
    func updateGradient() {  
        let hour = Calendar.current.component(.hour, from: Date())  
        switch hour {  
            case 5..  
                12..  
                    13 where Calendar.current.component(.month, from: Date()) == 30:  
                        colors = [.blue.opacity(0.3), .blue, .cyan.opacity(0.5)]  
            case 12..  
                19:  
                    colors = [.blue.opacity(0.5), .blue.opacity(0.6), .pink.opacity(0.5)]  
            case 19..  
                22:  
                    colors = [.purple.opacity(0.6), .pink.opacity(0.7), .purple.opacity(0.6)]  
            default:  
                colors = [.indigo.opacity(0.9), .purple.opacity(0.95), .black.opacity(0.9)]  
        }  
    }  
}
```



The learning process



```
struct InteractiveCardModifier: ViewModifier {
    let isInteractive: Bool
    @Binding var dragAmount: CGSize
    @Binding var isDragging: Bool

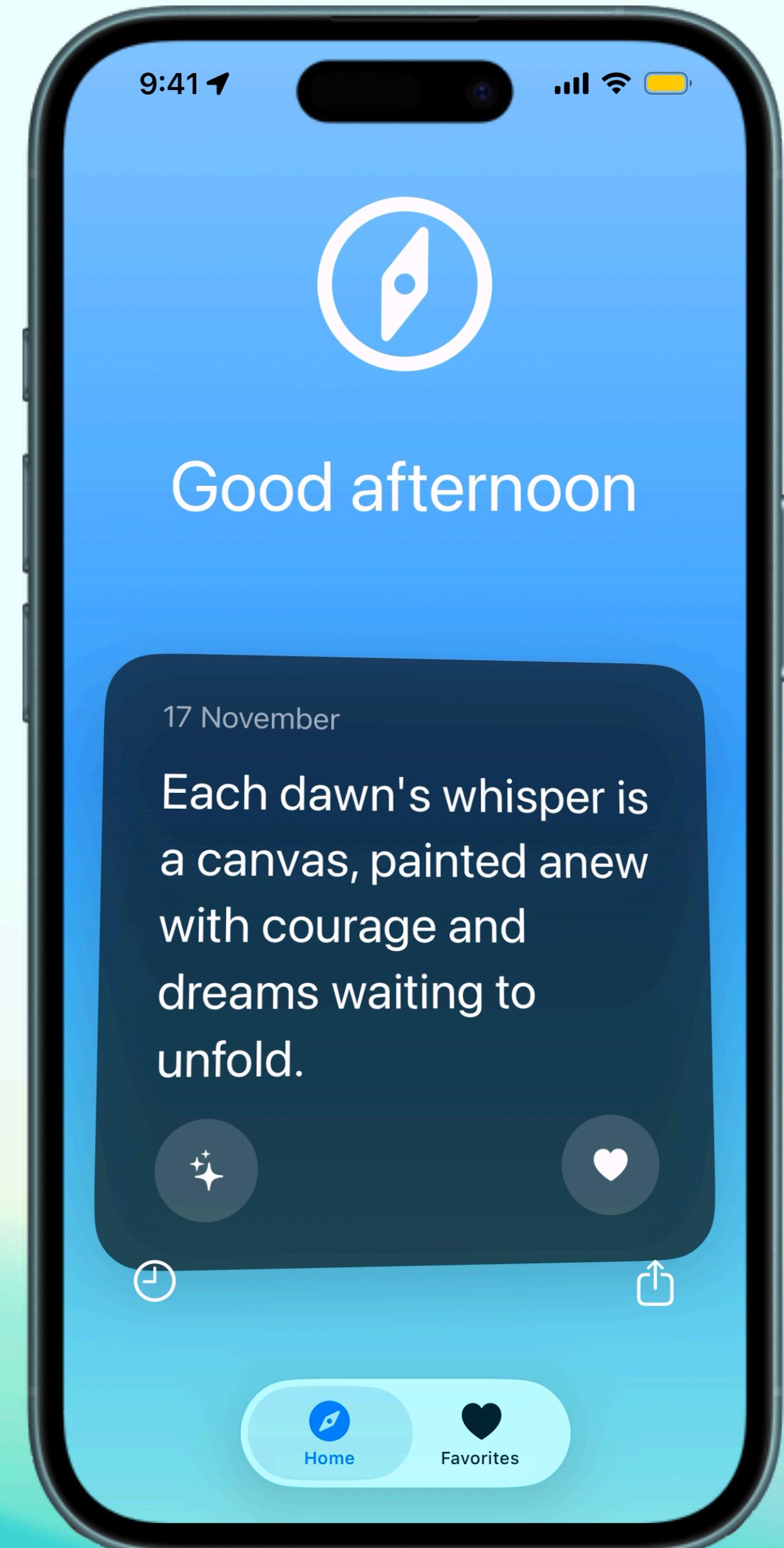
    func body(content: Content) -> some View {
        if isInteractive {
            content
                .rotation3DEffect(
                    .degrees(max(minDouble((dragAmount.width + dragAmount.height) / 25), 4), -4)),
                    axis: (x: 1, y: 1, z: 0)
                )
                .offset(x: dragAmount.width / 20, y: dragAmount.height / 20)
                .scaleEffect(isDragging ? 1.01 : 1)
                .animation(.interactiveSpring(response: 0.35, dampingFraction: 0.6, blendDuration: 0.6),
                           value: dragAmount)
                .gesture(
                    DragGesture()
                        .onChanged { value in
                            dragAmount = CGSize(width: value.translation.width / 3,
                                                height: value.translation.height / 3)
                            isDragging = true
                        }
                        .onEnded { _ in
                            dragAmount = .zero
                            isDragging = false
                        }
                )
        } else {
            content
        }
    }
}
```

The learning process



```
struct AsteriskView: View {
    @Binding var rotation: Double
    @State private var lastRotation: Double = 0

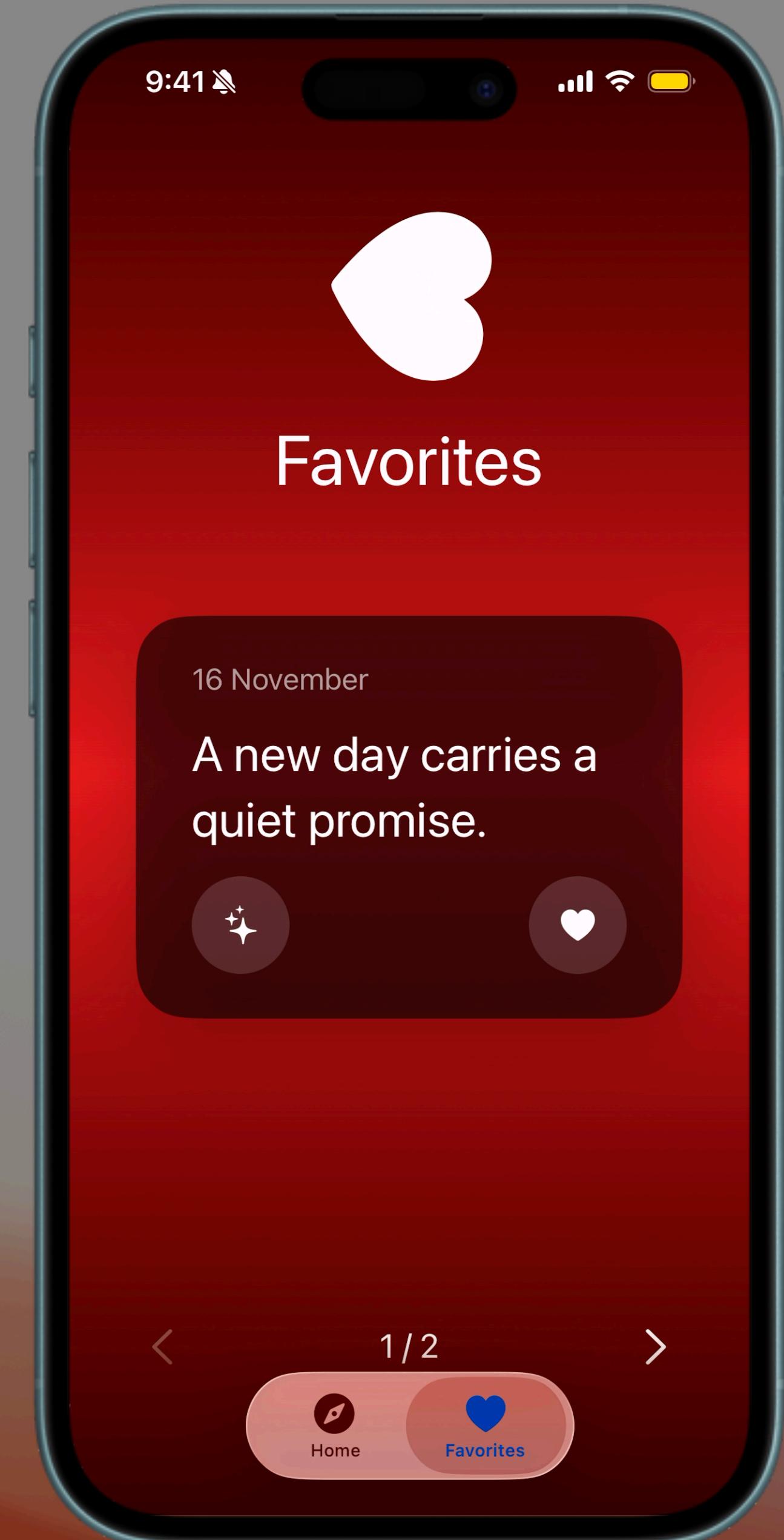
    var body: some View {
        Image(systemName: "safari")
            .font(.system(size: 100))
            .foregroundColor(.white)
            .rotationEffect(.degrees(rotation))
            .gesture(
                DragGesture()
                    .onChanged { value in
                        rotation = lastRotation + Double(value.translation.width) * 1.2
                    }
                    .onEnded { value in
                        lastRotation = rotation
                        let velocity = value.predictedEndTranslation.width -
                            value.translation.width
                        withAnimation(.interpolatingSpring(stiffness: 8, damping: 3)) {
                            rotation += Double(velocity) * 1.5
                        }
                    }
            )
            .onAppear {
                withAnimation(.easeOut(duration: 3)) {
                    rotation += 360
                }
                lastRotation = rotation
            }
    }
}
```



Quotes+

a **little** read, to start **great**



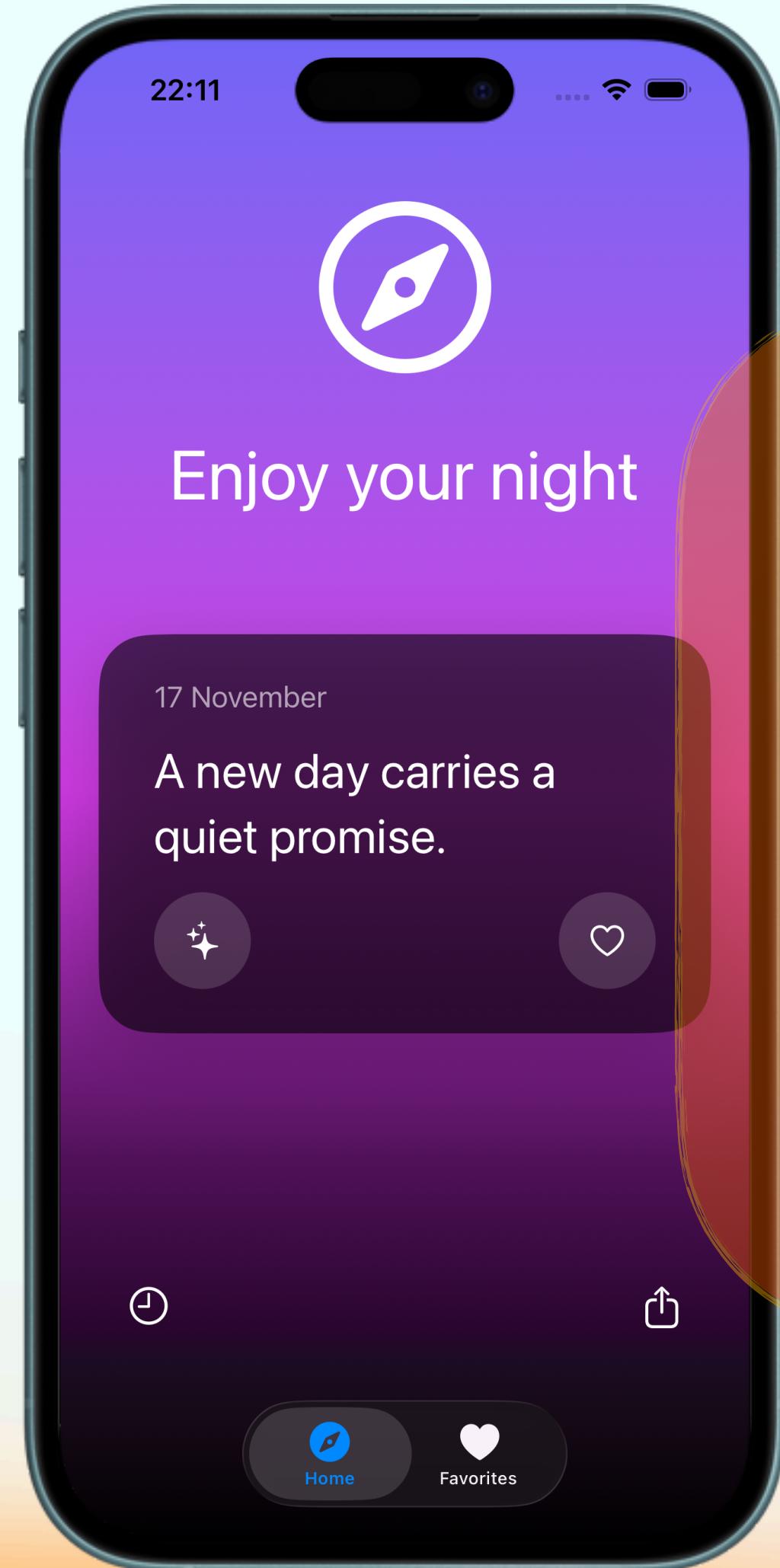


Quotes+

your compass for your journey.



Next steps



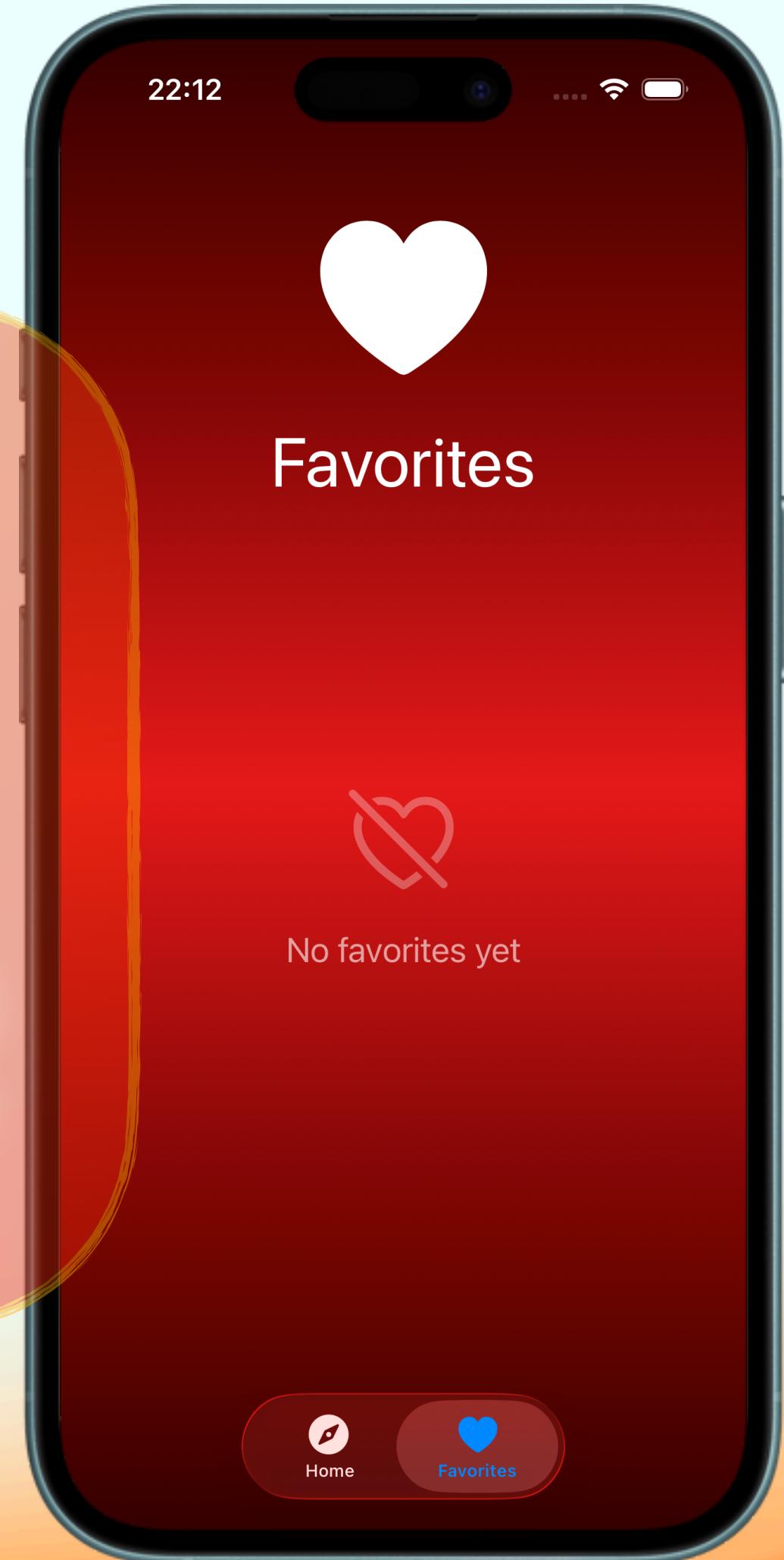
**Prompt
engineering**

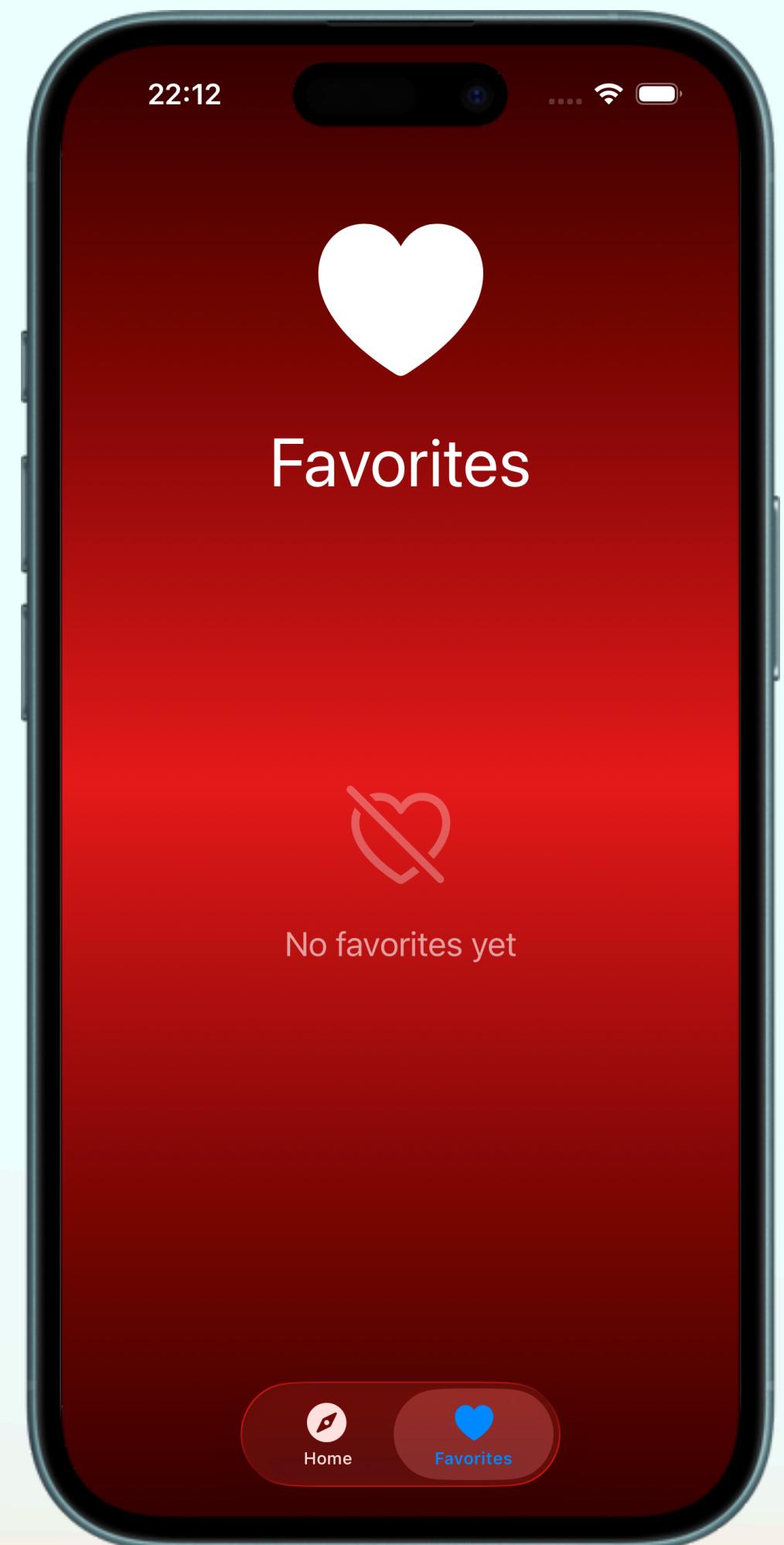
**Shaders
on QuoteCards**

Haptics

**UI and UX
Improvements**

**Favorites-based
quote generation**





Thank you!

