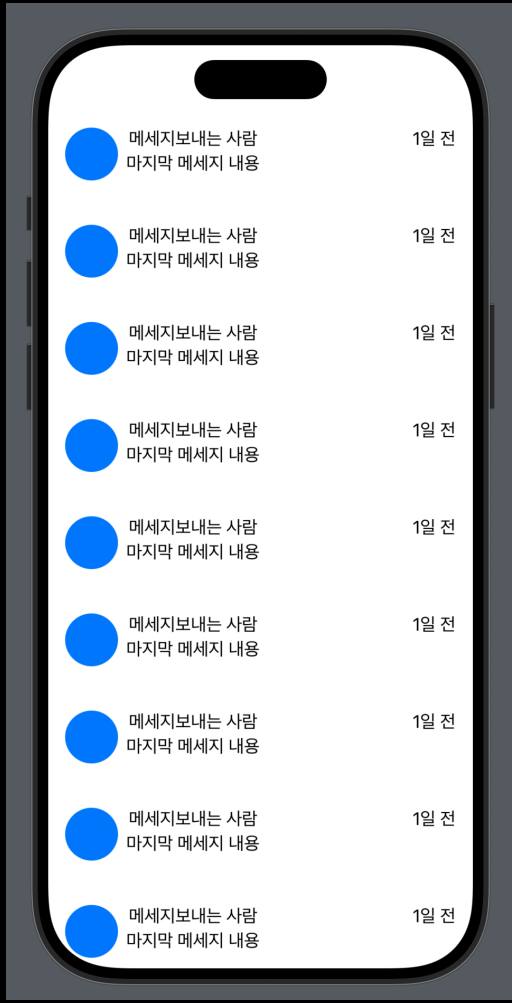
고급 뷰 컴포지션 및 커스텀 뷰 제작

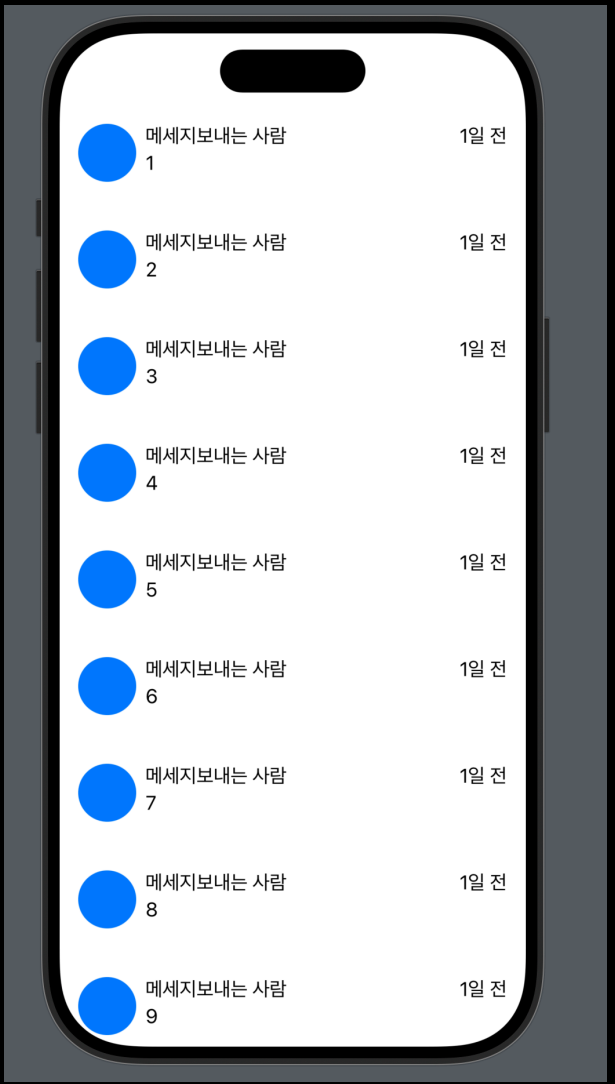
SwiftUI가 UIKit에 비해 코드의 양이 적고, 단순한것은 사실이지만, 그래도 어느정도 화면이 복잡해지면 body안에 선언된 UI구조를 파악하기 힘들어질수 밖에 없음.

```
struct ContentView: View {
    var body: some View {
        ScrollView {
            VStack(
                alignment: .leading,
                spacing: 10
            ) {
                ForEach(
                    1...1000,
                    id: \.self
                ) { item in
                    HStack(alignment: .top) {
                        Rectangle()
                            .fill(Color.blue)
                            .frame(width: 50, height: 50)
                            .clipShape(.capsule)

                        VStack(spacing: 3) {
                            Text("메세지보내는 사람")
                            Text("마지막 메세지 내용")
                        }

                        Spacer()
                        Text("1일 전")
                    }
                    .padding()
                }
            }
        }
    }
}
```



따라서 하나의 거대한 뷰 대신 기능이나 요구되는 디자인 시스템에 맞추어 컴포넌트를 분리할 필요가 있음.

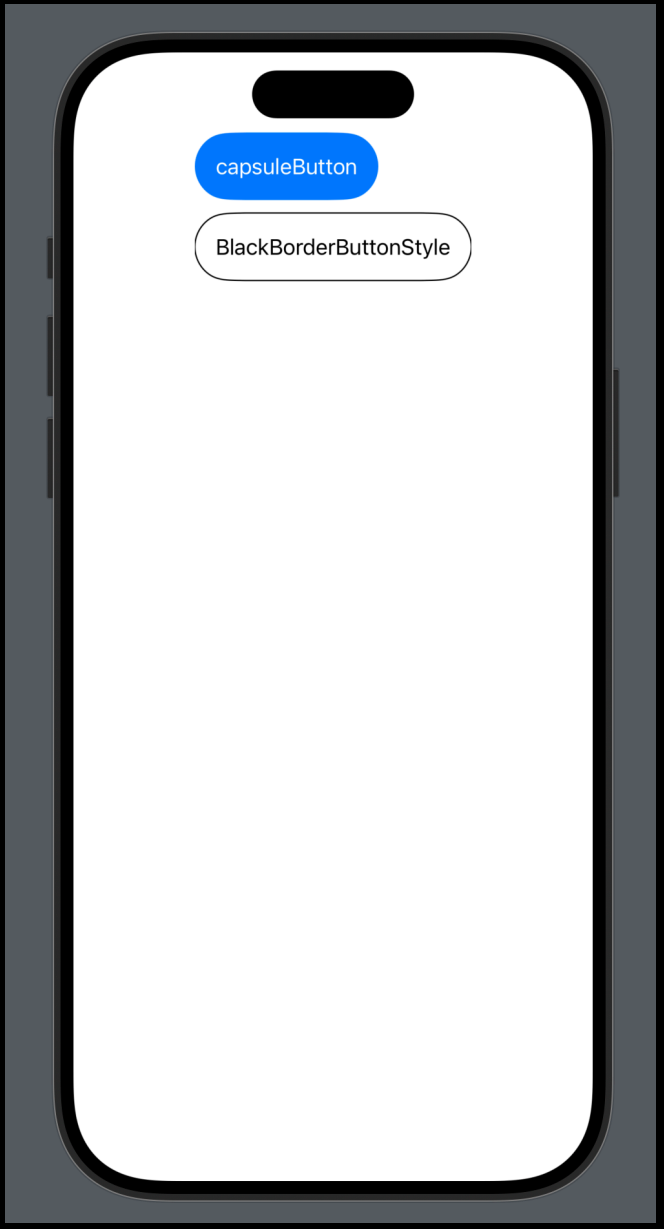
```
struct ContentView: View {
    var body: some View {
        ScrollView {
            VStack(
                alignment: .leading,
                spacing: 10
            ) {
                ForEach(
                    1...1000,
                    id: \.self
                ) { item in
                    ChatRoomView(index: item)
                }
            }
        }
    }
}
```



ButtonStyle을 활용하여 각 버튼의 커스텀을 간단하게 정의 가능

```swift
struct CapsuleButtonStyle: ButtonStyle {
    func makeBody(configuration: Configuration) -> some View {
        configuration.label
            .foregroundColor(.white)
              .padding()
              .background(Capsule().fill(.tint))
    }
}

struct BlackBorderButtonStyle: ButtonStyle {
    func makeBody(configuration: Configuration) -> some View {
        configuration.label
            .foregroundStyle(.black)
            .padding()
            .background(.white)
            .overlay(Capsule().stroke(Color.black, lineWidth: 1))

    }
}
```

```swift
struct ContentView: View {
    var body: some View {
        ScrollView {
            VStack(
                alignment: .leading,
                spacing: 10
            ) {
                Button("capsuleButton") {
                    print("capsuleButton")
                }
                .buttonStyle(CapsuleButtonStyle())

                Button("BlackBorderButtonStyle") {
                    print("BlackBorderButtonStyle")
                }
                .buttonStyle(BlackBorderButtonStyle())
            }
        }
    }
}
```



뷰 모디파이어 확장

ViewModifier란?
-기존 뷰나 다른 뷰에 수정자를 적용하여 원래 값의 다른 버전을 생성하는 수정자.
.font, .padding등 뷰에 새로운 수정자를 적용하여 해당 설정을 적용시키고,
다시 View를 리턴하도록 하여 다른 설정을 하고 싶을경우 자연스럽게 이어나가게 되는 코딩스타일로 작성이 가능.

모든 뷰에 재사용 가능한 수정자를 만들기 위해서는 ViewModifier 프로토콜 채택 후

필수 구현 메서드인 func body(content: Self.Content) -> Self.Body
를 구현하여 커스텀 모디파이어를 작성하여 구현

```swift
struct ContentView: View {
    var body: some View {
        ScrollView {
            VStack(
                alignment: .leading,
                spacing: 10
            ) {
                Button("blackMode") {
                    print("blackMode")
                }
                .modifier(BlackModifier())

                Button("whiteMode") {
                    print("whiteMode")
                }
                .whiteMode()
            }
        }
    }

}

#Preview {
    ContentView()
}


struct BlackModifier: ViewModifier {
    func body(content: Content) -> some View {
        content
            .padding()
            .foregroundStyle(.white)
            .background(Color.black)
            .cornerRadius(16)
            .shadow(color: .gray.opacity(0.2), radius: 8, x: 0, y: 4)

    }
}

struct WhiteModifier: ViewModifier {
    func body(content: Content) -> some View {
        content
            .padding()
            .foregroundStyle(.black)
            .background(Color.white)
            .cornerRadius(16)
            .shadow(color: .gray.opacity(0.2), radius: 8, x: 0, y: 4)
    }
}
```