I use wift language and when. someone click to on new wallet in. Apple a watch and online form he showed me input but I can't copy paste to this input any argument please how I can change this quote to to manage input from my phone which can. accept copy paste values

```
import SwiftUI

struct ContentView: View {
    @StateObject private var viewModel = BalanceViewModel()

    @State private var inputString = ""
    @State private var showingInputDialog = false

    var body: some View {
        VStack {
            Image(systemName: "globe")
                .imageScale(.large)
                .foregroundStyle(.tint)
            Text("Hello, world!")
            Text("Balance:")
                .font(.headline)
                .padding()
            Text("$\(Int(viewModel.balance))")
                .font(.title)
                .padding()

            TextField("Enter new wallet", text: $inputString, onCommit: {
                viewModel.updateWallet(wallet: inputString)

            })
            .textFieldStyle(.plain)

            .padding()
        }
        .padding()
    }
}


#Preview {
    ContentView()
}
```

To make it easier to enter a new wallet from your phone, which can accept copy-paste values, I'll suggest the following:

1. You might want to move the wallet input functionality to a separate screen on the iPhone if the issue is with the Apple Watch's input system.
2. You can use a button to present an action sheet that asks if the user wants to add a wallet from their phone or watch. If they choose the phone, then present a view or screen on the phone to input the value.

Here's a rough idea of how you could structure this:

swift Copy code

```swift
import SwiftUI
```

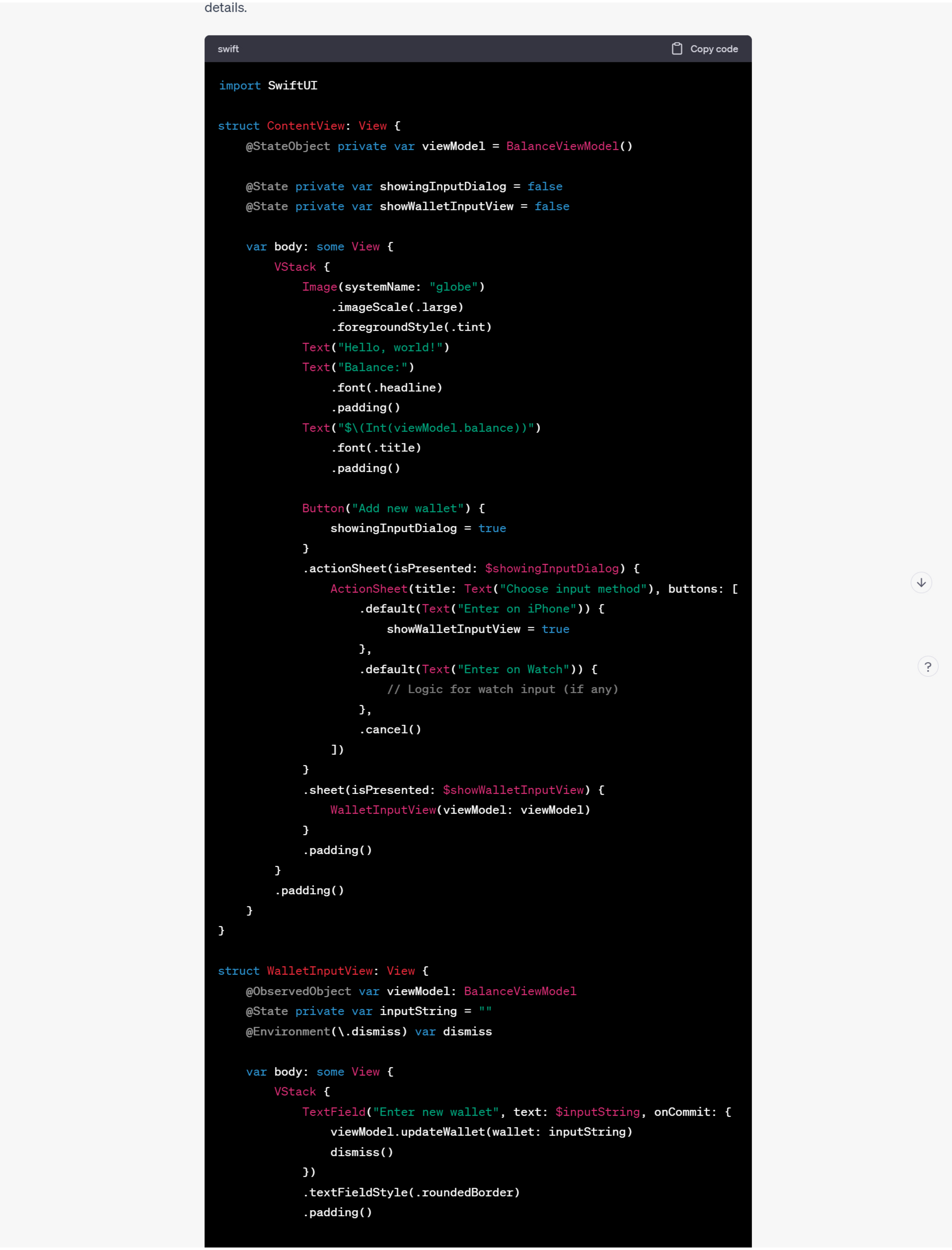
```
struct ContentView: View {
    @StateObject private var viewModel = BalanceViewModel()

    @State private var inputString = ""
    @State private var showingInputDialog = false

    var body: some View {
        VStack {
            Image(systemName: "globe")
                .imageScale(.large)
                .foregroundStyle(.tint)
            Text("Hello, world!")
            Text("Balance:")
                .font(.headline)
                .padding()
            Text("$\(Int(viewModel.balance))")
                .font(.title)
                .padding()

            Button("Add new wallet") {
                showingInputDialog = true
            }
            .actionSheet(isPresented: $showingInputDialog) {
                ActionSheet(title: Text("Choose input method"), buttons: [
                    .default(Text("Enter on iPhone")) {
                        // Present a view or a screen on iPhone for input
                    },
                    .default(Text("Enter on Watch")) {
                        // Here you can retain the existing logic or modify
                    },
                    .cancel()
                ])
            }
            .padding()
        }
        .padding()
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

This approach allows the user to choose how they want to input the new wallet details and provides a clearer path to enter it from their iPhone if that's their preference. If you're integrating with WatchKit or using a shared data model, you'll also need to make sure that any data entered on the iPhone is correctly synchronized with the Apple Watch.

send me code for
 // Present a view or a screen on iPhone for input

Certainly! Below is a modified version of your code to present a new view for iPhone input when the user selects "Enter on iPhone" from the action sheet.

This will transition to a new view (`WalletInputView`) where the user can input the wallet



details.

```swift
import SwiftUI

struct ContentView: View {
    @StateObject private var viewModel = BalanceViewModel()

    @State private var showingInputDialog = false
    @State private var showWalletInputView = false

    var body: some View {
        VStack {
            Image(systemName: "globe")
                .imageScale(.large)
                .foregroundStyle(.tint)
            Text("Hello, world!")
            Text("Balance:")
                .font(.headline)
                .padding()
            Text("$\(Int(viewModel.balance))")
                .font(.title)
                .padding()

            Button("Add new wallet") {
                showingInputDialog = true
            }
            .actionSheet(isPresented: $showingInputDialog) {
                ActionSheet(title: Text("Choose input method"), buttons: [
                    .default(Text("Enter on iPhone")) {
                        showWalletInputView = true
                    },
                    .default(Text("Enter on Watch")) {
                        // Logic for watch input (if any)
                    },
                    .cancel()
                ])
            }
            .sheet(isPresented: $showWalletInputView) {
                WalletInputView(viewModel: viewModel)
            }
            .padding()
        }
        .padding()
    }
}

struct WalletInputView: View {
    @ObservedObject var viewModel: BalanceViewModel
    @State private var inputString = ""
    @Environment(\.dismiss) var dismiss

    var body: some View {
        VStack {
            TextField("Enter new wallet", text: $inputString, onCommit: {
                viewModel.updateWallet(wallet: inputString)
                dismiss()
            })
            .textFieldStyle(.roundedBorder)
            .padding()
            Button("Submit") {
                viewModel.updateWallet(wallet: inputString)
                dismiss()
            }
            .padding()
        }
        .padding()
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Here are some notes about the changes:

1. The `WalletInputView` is a new struct that contains the TextField for entering the wallet details on the iPhone. Once the user has input the wallet details, they can press the "Submit" button to save the input and return to the `ContentView`.
2. I've added the `@Environment(\.dismiss) var dismiss` in the `WalletInputView`. This environment value is used to dismiss the modal view (`WalletInputView`) after the user has input the wallet details.
3. The `ContentView` now has a `.sheet(isPresented: $showWalletInputView)` modifier to present the `WalletInputView` modally when `showWalletInputView` is set to true.

Regenerate

Send a message

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

?