```
import SwiftUI

struct GameView: View {
    @State var deck: [Card] = []
    @State var playerHit: Bool = false
    @State var dealerHasHit: Bool = false
    @State var playerBust: Bool = false
    @State var playerWins: Bool = false
    @State var dealerWins: Bool = false
    @State var playerStand: Bool = false
    @State var dealerHasDrewCards: Bool = false
    @State var playerTotal: Int = 0
    @State var dealerTotal: Int = 0
    @State var setFisherYatesShuffle: Bool = false
    @State var setKnuthShuffle: Bool = false
    @State var setWongShuffle: Bool = false
    @State var playerCards: [Card] = []
    @State var dealerCards: [Card] = []
    @State var deckCreatedAndShuffled: Bool = false
    @State var gameStarted: Bool = false
    @State var showShuffleSheet: Bool = false
    @State var selectedShuffle: Int = 0
    @State private var showText: Bool = false
    @State var walletBalance: Int = 0
    @State var betAmount: String = ""
    @State var playerWinCount: Int = 0
    @State var dealerWinCount: Int = 0



    let shuffleOptions = ["Fisher-Yates", "Knuth"]

    let suits: [String: String] = ["Spades": "Spades", "Hearts": "Hearts", "Diamonds": "Diamonds",
"Clubs": "Clubs"]
    let ranks: [Int: String] = [2: "2", 3: "3", 4: "4", 5: "5", 6: "6", 7: "7", 8: "8", 9: "9", 10: "10"]

    struct Card {
        let rank: Int
        let suit: String
    }

    func createDeck() -> [Card] {
        for rank in ranks.keys {
            for suit in suits.keys {
```

```
                let card = Card(rank: rank, suit: suit)
                deck.append(card)
            }
        }

        if setFisherYatesShuffle == true {
            fisherYatesShuffle(deck: &deck)
        } else if setKnuthShuffle == true {
            knuthShuffle(deck: &deck)
        }

        return deck
    }

//choices for shuffle options
    enum ShuffleOption {
        case fisherYates
        case knuth
        case wong
        case none
    }

    func fisherYatesShuffle(deck: inout [Card]) {
        setFisherYatesShuffle = true
        for i in (0..<deck.count-1).reversed() {
            let randomIndex = Int.random(in: 0...i)
            if i != randomIndex {
                deck.swapAt(randomIndex, i)
            } else {
                continue
            }
        }
    }



    func knuthShuffle(deck: inout [Card]) {
        setKnuthShuffle = true
        for i in stride(from: deck.count - 1, through: 1, by: -1) {
            let randomIndex = Int.random(in: 0...i)
            if randomIndex != i {
                deck.swapAt(i, randomIndex)
            }
        }
```

```
}



//main game func
func playBlackjack() {
    gameStarted = true
    betAmount = betAmount
    deck = createDeck()

    playerCards = [deck[0], deck[1]]
    print("Player's cards: \(playerCards[0].rank) \(playerCards[1].rank)")

    dealerCards = [deck[2], deck[3]]
    print("Dealer's cards: \(dealerCards[0].rank), ???")

    playerTotal = playerCards[0].rank + playerCards[1].rank

    dealerTotal = dealerCards[0].rank + dealerCards[1].rank
}

func hitCard() {
    if playerBust == false {
        playerCards.append(deck[deck.count - 1])
        playerTotal += playerCards[playerCards.count - 1].rank
        deck.removeLast()
        playerHit = true
        if playerTotal > 21 {
            playerBust = true
            playerWins = false
        }
    }
}

func standTurn() {
    if playerBust == false {
        playerStand = true

    }
}

func dealerHits() {
    while dealerTotal < 17 && playerStand == true {
        dealerCards.append(deck[deck.count - 1])
```

```
        dealerTotal += dealerCards[dealerCards.count - 1].rank
        deck.removeLast()
        print("\(dealerCards)")
    }

    //determining the winner
    if dealerTotal > playerTotal && dealerTotal <= 21   {
        playerBust = true
        walletBalance -= Int(betAmount) ?? 0
        dealerWinCount += 1

    } else if playerTotal > dealerTotal && playerTotal <= 21 {
        playerWins = true
        walletBalance += Int(betAmount) ?? 0
        playerWinCount += 1

    } else if dealerTotal >= 21 {
        playerWins = true
        walletBalance += Int(betAmount) ?? 0
        playerWinCount += 1

    } else if playerTotal >= 21 {
        playerBust = true
        walletBalance -= Int(betAmount) ?? 0
        dealerWinCount += 1
    }

    dealerHasDrewCards = true
}

func restartGame() {
    playerHit = false
    playerBust = false
    playerWins = false
    dealerWins = false
    playerTotal = 0
    dealerTotal = 0
    setFisherYatesShuffle = false
    setKnuthShuffle = false
    setWongShuffle = false
    playerCards = []
    dealerCards = []
    deckCreatedAndShuffled = false
    gameStarted = false
```

```
        showShuffleSheet = false
        selectedShuffle = 0
        showText = false
        betAmount = ""
        dealerHasDrewCards = false
    }



    var body: some View {
        VStack {
            if deckCreatedAndShuffled == true {
                VStack {
                    if gameStarted == true {
//                      if playerBust == true {
//                          ZStack {
//                              Color.red
//                                  .ignoresSafeArea()
//                              VStack {
//                                  Spacer()
//                                  Text("YOU LOSE")
//                                      .foregroundColor(Color.white)
//                                      .font(.largeTitle)
//                                      .bold()
//                                      .padding(.top, 20)
//
//                                  Button(action: {
//                                      restartGame()
//                                  }, label: {
//                                      Text("Restart game")
//                                          .foregroundColor(Color.white)
//                                          .font(.title)
//                                          .underline()
//                                  })
//                                  .padding()
//                              }
//                          }
//
//                      } else if playerWins == true {
//                          ZStack {
//                              Color.green
//                                  .ignoresSafeArea()
//                              VStack {
```

```
//                        Spacer()
//                        Text("YOU WIN")
//                            .foregroundColor(Color.white)
//                            .font(.largeTitle)
//                            .padding(.top, 20)
//                        Button(action: {
//                            restartGame()
//                        }, label: {
//                            Text("Restart game")
//                                .foregroundColor(Color.black)
//                                .font(.title)
//                                .underline()
//                        })
//                    }
//                }
//            }
                VStack {
                    HStack {
                        Spacer()
                        Text("Dealer (CPU)")
                            .bold()
                        Spacer()

                        if dealerHasDrewCards == true || playerStand == true {
                            Text("Value : \(dealerTotal)")
                        } else {
                            Text("")
                        }


                        Spacer()
                    }
                    .font(.largeTitle)

                    if playerStand == false && dealerHasDrewCards == false {
                        Image("\(dealerCards[0].suit)\(dealerCards[0].rank)")
                            .resizable()
                            .frame(width: 120, height: 170)
                            .padding()
                    } else if playerStand == true || dealerHasDrewCards == true {
                        ScrollView(.horizontal) {
                            HStack {
                                ForEach(0..<dealerCards.count, id: \.self) { i in
                                    Image("\(dealerCards[i].suit)\(dealerCards[i].rank)")
```

```
                .resizable()
                .frame(width: 120, height: 170)
                .aspectRatio(contentMode: .fit)
                .padding()
            }
        }
        .padding()
    }
}


if playerStand == false {
    if playerTotal < 21 {
        HStack {
            Spacer()

            Button(action: {
                hitCard()
            }, label: {
                Text("Hit")
                    .foregroundColor(.white)
                    .font(.system(size: 20, weight: .bold, design: .rounded))
                    .cornerRadius(8)
                    .padding(10)
                    .background(.red)
            })

            Spacer()

            Button(action: {
                standTurn()
            }, label: {
                Text("Stand")
                    .foregroundColor(.white)
                    .font(.system(size: 20, weight: .bold, design: .rounded))
                    .cornerRadius(8)
                    .padding(10)
                    .background(.black)
            })

            Spacer()
        }
    } else {
        VStack {
```

```
            Text("You lose")
                .foregroundColor(Color.red)
                .font(.largeTitle)
                .bold()

            Text("Wallet balance : \(walletBalance) (-\(betAmount)")
                .font(.title3)

            Button(action: {
                restartGame()
            }, label: {
                Text("Restart Game")
                    .foregroundColor(.white)
                    .font(.system(size: 20, weight: .bold, design: .rounded))
                    .padding()
                    .background(.black)
                    .cornerRadius(8)
            })

        }
    }

} else if playerStand == true  {
    if dealerTotal < 17 {
        VStack {
            Text("Dealer is \(17 - dealerTotal) ranks away from 17")
                .font(.title2)
            Button(action: {
                dealerHits()
            }, label: {
                Text("Let dealer hit")
                    .foregroundColor(.white)
                    .padding()
                    .background(.black)
                    .cornerRadius(8)
            })

        }
    } else {
        if playerBust == true {
            VStack {
                Text("You lose")
                    .foregroundColor(Color.red)
                    .font(.largeTitle)
```

```
                .bold()

            Text("Wallet balance : \(walletBalance) (-\(betAmount)")
                .font(.title3)

            Button(action: {
                restartGame()
            }, label: {
                Text("Restart Game")
                    .foregroundColor(.white)
                    .font(.system(size: 20, weight: .bold, design: .rounded))
                    .padding()
                    .background(.black)
                    .cornerRadius(8)
            })

        }

    } else if playerWins == true {
        VStack {
            Text("You win")
                .foregroundColor(Color.green)
                .font(.largeTitle)
                .bold()

            Text("Wallet balance : \(walletBalance) (+\(betAmount)")
                .font(.title3)

            Button(action: {
                restartGame()
            }, label: {
                Text("Restart Game")
                    .foregroundColor(.white)
                    .font(.system(size: 20, weight: .bold, design: .rounded))
                    .padding()
                    .background(.black)
                    .cornerRadius(8)
            })
        }


    }
}
```

```
}



ScrollView(.horizontal) {
    HStack {
        ForEach(0..<playerCards.count, id: \.self) { i in
            Image("\(playerCards[i].suit)\(playerCards[i].rank)")
                .resizable()
                .frame(width: 120, height: 170)
                .aspectRatio(contentMode: .fit)
                .padding()
        }
    }
    .padding()
}

HStack {
    Spacer()
    Text("You")
        .bold()
    Spacer()
    Text("Value : \(playerTotal)")
    Spacer()
}
.font(.largeTitle)
.padding(.bottom)


HStack {
    Spacer()
    VStack {
        Text("WALLET BALANCE")
            .bold()
        Text("\(walletBalance)")
    }

    Spacer()
    if betAmount == "" {
        VStack {
            Text("BETTING AMOUNT")
                .bold()
            Text("0")
        }
```

```
                    } else {
                        VStack {
                            Text("BETTING AMOUNT")
                                .bold()
                            Text("\(betAmount)")
                        }
                    }

                    Spacer()
                }



            }
        } else {
            VStack {
                Spacer()
                Text("Deck has been created and shuffled.")
                    .multilineTextAlignment(.center)
                    .font(.largeTitle)
                    .padding()

                Text("Wallet Balance : \(walletBalance) Chips")
                    .font(.title)
                    .bold()
                    .padding(.bottom)

                Text("How much are you willing to bet?")
                    .font(.title2)

                TextField("Enter an integer", text: $betAmount)
                        .textFieldStyle(RoundedBorderTextFieldStyle())
                        .padding()
                        .frame(maxWidth: 200)

                if betAmount == "" {
                    Text("")
                } else {
                    Text("You are betting : \(betAmount) Chips")
                        .font(.title)
                        .bold()
                        .padding(.bottom)
                    Text("Wallet balance after win : \(walletBalance + (Int(betAmount) ?? 0))
Chips")
```

```
                        .foregroundColor(Color.green)
                        .font(.title2)
                    Text("Wallet balance after loss : \(walletBalance - (Int(betAmount) ?? 0))
Chips")
                        .foregroundColor(Color.red)
                        .font(.title2)
                }


                Spacer()

                Button(action: {
                    playBlackjack()
                    gameStarted = true
                }, label: {
                    Text("Start Game!")
                        .foregroundColor(.white)
                        .font(.system(size: 20, weight: .bold, design: .rounded))
                        .padding()
                        .background(.black)
                        .cornerRadius(8)
                })
            }
        }
    }
} else {
    ZStack {
        Color.black
            .ignoresSafeArea()
        VStack {
            Spacer()
            Text("Blackjack")
                .foregroundColor(.white)
                .font(.system(size: 60, weight: .bold, design: .monospaced))
                .padding()
            Text("Welcome to the casino")
                .foregroundColor(.white)
                .font(.largeTitle)
                .padding(.bottom)
            Text("Number of times you won : \(playerWinCount)")
                .foregroundColor(.white)
                .font(.title3)
            Text("Number of times dealer won : \(dealerWinCount)")
                .foregroundColor(.white)
```

```
                    .font(.title3)
                Spacer()
                Button(action: {
                    self.deck = self.createDeck()
                    self.showShuffleSheet = true
                }) {

                    Text("Create Deck")
                        .foregroundColor(.white)
                        .font(.system(size: 20, weight: .bold, design: .rounded))
                        .padding()
                        .background(.red)
                        .cornerRadius(8)
                }

            }
            .sheet(isPresented: $showShuffleSheet) {
                    Form {
                        Picker("Shuffle Algorithm", selection: $selectedShuffle) {
                            ForEach(0 ..< shuffleOptions.count) {
                                Text(self.shuffleOptions[$0])
                            }
                        }
                        Button(action: {
                            if self.selectedShuffle == 0 {
                                self.fisherYatesShuffle(deck: &self.deck)
                            } else if self.selectedShuffle == 1 {
                                self.knuthShuffle(deck: &self.deck)
                            }
                            showShuffleSheet = false
                            deckCreatedAndShuffled = true
                        }) {
                            Text("Apply Shuffle")
                        }
                    }
                }
            }

        }
    }
}

struct GameView_Previews: PreviewProvider {
```

```
    static var previews: some View {
        GameView()
    }
}
```