# AUTH

Monday, September 18, 2023 7:36 AM

```swift
//
//  Stream_TutorialsApp.swift
//  Shared
//
//  Created by Balaji on 23/03/21.
//

import SwiftUI
import StreamChat
import Firebase
import JWTKit

@main
struct Stream_TutorialsApp: App {
    
    // calling Delegate...
    @UIApplicationDelegateAdaptor(AppDelegate.self) var delegate
    
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}

// Delegate...
class AppDelegate: NSObject,UIApplicationDelegate{
    
    // diffent way of intializing the Stream...
    
    @AppStorage("userName") var storedUser = ""
    @AppStorage("log_Status") var logStatus = false
    
    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey : Any]? = nil) -> Bool {
        
        // Intializing Firebase...
        FirebaseApp.configure()
        
        // if user already logged in...
        if logStatus{
            
            // Reloading user if logged in...
            // Verifying if user is already in stream SDk or Not
```

```swift
//
//  ContentView.swift
//  Shared
//
//  Created by Balaji on 23/03/21.
//

import SwiftUI

struct ContentView: View {
    
    @StateObject var streamData = StreamViewModel()
    @StateObject var model = LoginViewModel()
    @AppStorage("log_Status") var logStatus = false
    
    var body: some View {
        
        NavigationView{
            
            if !logStatus{
                if model.newUser{
                    
                    Login()
                        .environmentObject(streamData)
                        .navigationTitle("Register")
                }
                else{
                    OtpLogin()
                        .environmentObject(model)
                        .navigationTitle("Login")
                }
            }
            else{
                
                ChannelView()
                
            }
        }
        // Since we have differnt alerts...
        .background(
            
            ZStack{
                Text("")
                    .alert(isPresented:
```

```swift
//
//  Credential.swift
//  Stream Tutorials
//
//  Created by Balaji on 23/03/21.
//

import SwiftUI

let APIKey = ""
let secretKey = ""
```

```
Not...

        // for that we need to intialize the stream sdk
with JWT Tokens...
        // AKA known as Authenticatiog with stream
SDK....

        // generating JWT Token...

        let signers = JWTSigners()
        signers.use(.hs256(key:
secretKey.data(using: .utf8)!))

        // Creating Payload and inserting Userd ID to
generate Token..
        // Here User ID will be Firebase UID....
        // Since its Unique...

        guard let uid = Auth.auth().currentUser?.uid
else{
            return true
        }

        let payload = PayLoad(user_id: uid)

        // generating Token...
        do{

            let jwt = try signers.sign(payload)

            print(jwt)

            let config = ChatClientConfig(apiKeyString:
APIKey)

            let tokenProvider = TokenProvider.closure
{ client, completion in

                guard let token = try? Token(rawValue: jwt)
else{
                    return
                }

                completion(.success(token))
            }

            ChatClient.shared = ChatClient(config: config,
tokenProvider: tokenProvider)

ChatClient.shared.currentUserController().reloadUserI
fNeeded()

        }
        catch{
            print(error.localizedDescription)
        }
    }
```

```
                .alert(isPresented:
$model.showAlert, content: {

                    Alert(title: Text("Message"),
message: Text(model.errorMsg),
dismissButton: .destructive(Text("Ok"),
action: {

                        withAnimation{
                            model.isLoading = false
                        }
                    }))
                })

            Text("")
                .alert(isPresented:
$streamData.error, content: {

                    Alert(title: Text("Message"),
message: Text(streamData.errorMsg),
dismissButton: .destructive(Text("Ok"),
action: {

                        withAnimation{
                            streamData.isLoading =
false
                        }
                    }))
                })
            }
        )
        .overlay(
            ZStack{


                // New Channel View....
                if
streamData.createNewChannel{CreateNewC
hannel()}

                // Lodaing Screen...
                if model.isLoading ||
streamData.isLoading{LoadingScreen()}
            }
        )
        .environmentObject(streamData)
        .onChange(of: logStatus, perform:
{ value in
            if logStatus{
                model.newUser = false
            }
        })
    }
}

struct ContentView_Previews:
PreviewProvider {
        return true
    }

    func application(_ application: UIApplication,
didReceiveRemoteNotification userInfo:
[AnyHashable : Any], fetchCompletionHandler
completionHandler: @escaping
(UIBackgroundFetchResult) -> Void) {

    }
}

// stream API...
extension ChatClient{
    static var shared: ChatClient!
}
```

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

# VIEW MODEL

```
//
//  LoginViewModel.swift
//  Stream Tutorials
//
//  Created by Balaji on 11/05/21.
//

import SwiftUI
import Firebase
import StreamChat
import JWTKit
```

```
//
//  StreamViewModel.swift
//  Stream Tutorials
//
//  Created by Balaji on 23/03/21.
//

import SwiftUI
import StreamChat

class StreamViewModel: ObservableObject {
```

```
class LoginViewModel: ObservableObject {

   // Logi Properties...
   @Published var countryCode = ""
   @Published var phNumber = ""

   // Alert...
   @Published var showAlert = false
   @Published var errorMsg = ""

   // Verification ID
   @Published var ID = ""

   // Loading...
   @Published var isLoading = false

   @AppStorage("log_Status") var logStatus = false
   @AppStorage("userName") var storedUser = ""
   @Published var newUser = false

   func verifyUser(){

      withAnimation{isLoading = true}

      // Undo this if testing with real devices or real ph Numbers...

Auth.auth().settings?.isAppVerificationDisabledForTesting = true

      // Sending Otp And Verifying user...

PhoneAuthProvider.provider().verifyPhoneNumber("+\(countryCode + phNumber)", uiDelegate: nil) { ID, err in

         if let error = err{
            self.errorMsg = error.localizedDescription
            self.showAlert.toggle()
            return
         }

         self.ID = ID!
         self.alertWithTF()
      }
   }

   // Alert With TextField For OTP Code...
   func alertWithTF(){

      let alert = UIAlertController(title: "Verification", message: "Enter OTP Code", preferredStyle: .alert)

      alert.addTextField { txt in
         txt.placeholder = "123456"
      }
```

```
   @Published var userName = ""

   @AppStorage("userName") var storedUser = ""
   @AppStorage("log_Status") var logStatus = false

   // Alert....
   @Published var error = false
   @Published var errorMsg = ""

   // Loading Screen...
   @Published var isLoading = false

   // Channel Data...
   @Published var channels : [ChatChannelController.ObservableObject]!

   // Create New Channel...
   @Published var createNewChannel = false
   @Published var channelName = ""

   func logInUser(){

      // Logging In User....

      withAnimation{isLoading = true}

      // Upadting User Profile...
      // you can give user image url if want....

ChatClient.shared.currentUserController().updateUserData(name: userName, imageURL: nil, userExtraData: .defaultValue) { err in

         withAnimation{self.isLoading = false}

         if let error = err{
            self.errorMsg = error.localizedDescription
            self.error.toggle()
            return
         }

         // Else SUccessful...
         // storing user Name...
         self.storedUser = self.userName
         self.logStatus = true


ChatClient.shared.currentUserController().reloadUserIfNeeded()

      }
   }

   // Fetching All Channels...
   func fetchAllChannels(){

      if channels == nil{
        alert.addAction(UIAlertAction(title: "Cancel",
style: .destructive, handler: nil))
        alert.addAction(UIAlertAction(title: "Ok",
style: .default, handler: { _ in

            if let code = alert.textFields?[0].text{
                self.LoginUser(code: code)
            }
            else{
                self.reportError()
            }

        }))

        // presenting Alert View...

UIApplication.shared.windows.first?.rootViewContr
oller?.present(alert, animated: true, completion: nil)
    }

    // Loggin in User...
    func LoginUser(code: String){

        let credential =
PhoneAuthProvider.provider().credential(withVerific
ationID: self.ID, verificationCode: code)

        Auth.auth().signIn(with: credential) { result, err
in
            if let error = err{
                self.errorMsg = error.localizedDescription
                self.showAlert.toggle()
                return
            }

            // user Successfully Logged In....
            print("success")

            // Verifying if user is already in stream SDk or
Not...

            // for that we need to intialize the stream sdk
with JWT Tokens...
            // AKA known as Authenticatiog with stream
SDK....

            // generating JWT Token...

            let signers = JWTSigners()
            signers.use(.hs256(key:
secretKey.data(using: .utf8)!))

            // Creating Payload and inserting Userd ID to
generate Token..
            // Here User ID will be Firebase UID....
            // Since its Unique...
```

```
            if channels == nil{
            // filter...


ChatClient.shared.currentUserController().reloadUserIfN
eeded()

            let filter =
Filter<ChannelListFilterScope>.equal("type", to:
"messaging")

            let request =
ChatClient.shared.channelListController(query: .init(filter
: filter))

            request.synchronize { (err) in
                if let error = err{
                    self.errorMsg = error.localizedDescription
                    self.error.toggle()
                    return
                }

                DispatchQueue.main.async {

                    // else Successful...
                    self.channels =
request.channels.compactMap({ (channel) ->
ChatChannelController.ObservableObject? in

                        return
ChatClient.shared.channelController(for:
channel.cid).observableObject
                    })
                }
            }
        }
    }

    // Creating New CHannel...
    func createChannel(){

        withAnimation{self.isLoading = true}

        let normalizedChannelName =
channelName.replacingOccurrences(of: " ", with: "-")

        let newChannel = ChannelId(type: .messaging, id:
normalizedChannelName)

        // you can givve image url to channel...
        // same you can also give image url to user....
        let request = try!
ChatClient.shared.channelController(createChannelWith
Id: newChannel, name: normalizedChannelName,
imageURL: nil, extraData: .defaultValue)

        request.synchronize { (err) in
        guard let uid = Auth.auth().currentUser?.uid
else{
            self.reportError()
            return
        }

        let payload = PayLoad(user_id: uid)

        // generating Token...
        do{

            let jwt = try signers.sign(payload)

            print(jwt)

            let config = ChatClientConfig(apiKeyString:
APIKey)

            let tokenProvider = TokenProvider.closure
{ client, completion in

                guard let token = try? Token(rawValue:
jwt) else{
                    self.reportError()
                    return
                }

                completion(.success(token))
            }

            ChatClient.shared = ChatClient(config:
config, tokenProvider: tokenProvider)

            // Reloading ChatClient...

ChatClient.shared.currentUserController().reloadUs
erIfNeeded { err in

                if let _ = err{
                    self.reportError()
                    return
                }

                // Simple Trick to find the user is already
signed up..
                // Just Checking the user having name...
                // if yes then it means the user already
signed up..
                // else new user...

                if let name =
ChatClient.shared.currentUserController().currentUs
er?.name{

                    withAnimation{
                        self.storedUser = name
```

```
        withAnimation{self.isLoading = false}

        if let error = err{
            self.errorMsg = "Try Again Later !!!\n\nAvoid
Using Special Character like $,'%..etc\n\n
\(error.localizedDescription)"
            self.error.toggle()
            return
        }

        // Succes....
        // closing Loading And New Channle View....
        self.channelName = ""
        withAnimation{self.createNewChannel = false}
        self.channels = nil
        self.fetchAllChannels()
    }
  }
}
```

```
                            self.logStatus = true
                            self.isLoading = false
                        }
                    }
                    else{

                        withAnimation{
                            self.newUser = true
                            self.isLoading = false
                        }
                    }
                }
            }
            catch{
                print(error.localizedDescription)
            }

        }
    }

    // Reporting Error...
    func reportError(){
        self.errorMsg = "Please try again later !!!"
        self.showAlert.toggle()
    }
}

struct PayLoad: JWTPayload,Equatable {

    enum CodingKeys: String,CodingKey {
        case user_id
    }

    var user_id: String

    func verify(using signer: JWTSigner) throws {

    }
}
```

# VIEWS

```
//
//  ChannelView.swift
//  Stream Tutorials
//
//  Created by Balaji on 23/03/21.
//

import SwiftUI
import StreamChat
import Firebase

struct ChannelView: View {

   @EnvironmentObject var streamData:
StreamViewModel
   @AppStorage("userName") var storedUser = ""
   @AppStorage("log_Status") var logStatus = false


   var body: some View {

      // Channel View...
      ScrollView(.vertical, showsIndicators: false,
content: {

         VStack(spacing: 20){

            if let channels = streamData.channels{

               ForEach(channels,id: \.channel){listner in

                  NavigationLink(
                     destination: ChatView(listner: listner),
                     label: {

                        ChannelRowView(listner: listner)
                     })
               }
            }
            else{
               // Progress View....
               ProgressView()
                  .padding(.top,20)
            }
         }
         .padding()
      })
      .navigationTitle("Channel")
      // Navigation Bar Buttons....
      .toolbar(content: {

         ToolbarItem(placement: .navigationBarTrailing) {

            Button(action: {
               streamData.channels = nil
               streamData.fetchAllChannels()
```

```
//
//  ChatView.swift
//  Stream Tutorials
//
//  Created by Balaji on 23/03/21.
//

import SwiftUI
import StreamChat

struct ChatView: View {

   // since its observing object so its automaticlly
observing and refreshing....
   @StateObject var listner:
ChatChannelController.ObservableObject

   //Message
   @State var message = ""

   // Color Scheme
   @Environment(\.colorScheme) var scheme

   var body: some View {

      let channel = listner.controller.channel!

      VStack{

         // scrollView Reader for Scrolling down...
         ScrollViewReader{reader in

            ScrollView(.vertical, showsIndicators: false,
content: {

               // Lazy Stack For Lazy Loading...
               LazyVStack(alignment: .center, spacing: 15,
content: {

                  ForEach(listner.messages.reversed(),id:
\.self){msg in

                     // Message Row...
                     MessageRowView(messsage: msg)
                  }
               })
               .padding()
               .padding(.bottom,10)
               .id("MSG_VIEW")
            })
            .onChange(of: listner.messages, perform:
{ value in

               withAnimation{

reader.scrollTo("MSG_VIEW",anchor: .bottom)
```

```
            streamData.fetchAllChannels()
          }, label: {
            Image(systemName:
"arrow.clockwise.circle.fill")
          })
        }

        ToolbarItem(placement: .navigationBarTrailing) {

          Button(action: {

withAnimation{streamData.createNewChannel.toggle()
}
          }, label: {
            Image(systemName: "square.and.pencil")
          })
        }

        ToolbarItem(placement: .navigationBarLeading)
{

          Button(action: {
            // Logging Out...
            logStatus = false
            storedUser = ""
            try! Auth.auth().signOut()
          }, label: {
            Image(systemName: "power")
          })
        }
      })
      .onAppear(perform: {
        streamData.fetchAllChannels()
      })
  }
}

struct ChannelView_Previews: PreviewProvider {
  static var previews: some View {
    ChannelView()
  }
}

// Channel Row View....
struct ChannelRowView: View {

  @StateObject var listner:
ChatChannelController.ObservableObject

  @EnvironmentObject var streamData:
StreamViewModel

  var body: some View{

    VStack(alignment: .trailing, spacing: 5, content: {

      HStack(spacing: 12){
```

```
reader.scrollTo("MSG_VIEW",anchor: .bottom)
          }
        })
        .onAppear(perform: {
          // scrolling to bottom...

reader.scrollTo("MSG_VIEW",anchor: .bottom)
        })
      }

      // TextField And Send Button....
      HStack(spacing: 10){

        TextField("Message", text: $message)
          .modifier(ShadowModifier())

        Button(action: sendMessage, label: {
          Image(systemName: "paperplane.fill")
            .padding(10)
            .background(Color.primary)
            .foregroundColor(scheme
== .dark ? .black : .white)
            .clipShape(Circle())
        })
        // Disabling Button when no txt typed...
        .disabled(message == "")
        .opacity(message == "" ? 0.5 : 1)
      }
      .padding(.horizontal)
      .padding(.bottom,8)
    }
    .navigationTitle(channel.cid.id)
  }

  // sending Message...
  func sendMessage(){

    // since we created a channel for messaging...

    let channelID = ChannelId(type: .messaging, id:
listner.channel?.cid.id ?? "")

    ChatClient.shared.channelController(for:
channelID).createNewMessage(text: message){result in

      switch result{

      case .success(let id):
        print("success = \(id)")

      case .failure(let error):
        // show error...
        print(error.localizedDescription)
      }
    }

    //clearing Msg Field...
```

```
        let channel = listner.controller.channel!

        Circle()
          .fill(Color.gray.opacity(0.4))
          .frame(width: 55, height: 55)
          .overlay(

            // First Letter as Image...
            Text("\(String(channel.cid.id.first!))")
              .font(.title)
              .fontWeight(.semibold)
              .foregroundColor(.primary)
          )

        VStack(alignment: .leading, spacing: 8,
content: {
          Text(channel.cid.id)
            .fontWeight(.semibold)
            .foregroundColor(.primary)

          // Last Msg...
          if let lastMsg = channel.latestMessages.first{

            // showing the last user name...
            (

              Text(lastMsg.isSentByCurrentUser ?
"Me: " : "\(lastMsg.author.id): ")

              +

                Text(lastMsg.text)
            )
            .font(.caption)
            .foregroundColor(.gray)
            .lineLimit(1)

          }
        })

        Spacer(minLength: 10)

        // Time...
        if let time =
channel.latestMessages.first?.createdAt{
          Text(time,style: checkIsDateToday(date:
time) ? .time : .date)
            .font(.caption2)
            .foregroundColor(.gray)
        }
      }
      .frame(maxWidth: .infinity, alignment: .leading)

      Divider()
        .padding(.leading,60)
    })
```

```
    message = ""
  }
}

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

```
        .onAppear(perform: {
            // watching the updates on channel..
            listner.controller.synchronize()
        })
        .onChange(of:
listner.controller.channel?.latestMessages.first?.text,
perform: { value in
            // firing sort...
            print("sort channels...")
            sortChannels()
        })
    }

    // checking if msg is from today then display time
else display date...
    func checkIsDateToday(date: Date)->Bool{

        let calender = Calendar.current

        if calender.isDateInToday(date){
            return true
        }
        else{
            return false
        }
    }

    func sortChannels(){

        let result = streamData.channels.sorted { (ch1,
ch2) -> Bool in

            if let date1 =
ch1.channel?.latestMessages.first?.createdAt{

                if let date2 =
ch2.channel?.latestMessages.first?.createdAt{

                    return date1 > date2
                }
                else{
                    return false
                }
            }
            else{
                return false
            }
        }

        streamData.channels = result
    }
}
```

```swift
//
//  ChatBubble.swift
//  Stream Tutorials
//
//  Created by Balaji on
24/03/21.
//

import SwiftUI

struct ChatBubble: Shape {

    var corners: UIRectCorner

    func path(in rect:
CGRect) -> Path {

        let path =
UIBezierPath(roundedRect:
rect, byRoundingCorners:
corners, cornerRadii:
CGSize(width: 13, height:
13))

        return
Path(path.cgPath)
    }
}
```

```swift
//
//  CreateNewChannel.swift
//  Stream Tutorials
//
//  Created by Balaji on 23/03/21.
//

import SwiftUI

struct CreateNewChannel: View {
    @EnvironmentObject var streamData:
StreamViewModel
    @Environment(\.colorScheme) var scheme
    var body: some View {

        VStack(alignment: .leading, spacing: 15,
content: {

            Text("Create New Channel")
                .font(.title2)
                .fontWeight(.bold)

            TextField("iJustine", text:
$streamData.channelName)
                .autocapitalization(.none)
                .disableAutocorrection(true)
                .modifier(ShadowModifier())

            // Button...
            Button(action:
streamData.createChannel, label: {
                Text("Create Channel")
                    .padding(.vertical,10)
                    .frame(maxWidth: .infinity,
alignment: .center)
                    .background(Color.primary)
                    .foregroundColor(scheme
== .dark ? .black : .white)
                    .cornerRadius(8)
            })
            .padding(.top,10)
            .disabled(streamData.channelName
== "")
            .opacity(streamData.channelName ==
"" ? 0.5 : 1)
        })
        .padding()
        .background(scheme == .dark ?
Color.black : Color.white)
        .cornerRadius(12)
        .padding(.horizontal,35)
        .frame(maxWidth: .infinity,
maxHeight: .infinity)
        .background(Color.primary.opacity(0.2).i
gnoresSafeArea().onTapGesture {
            streamData.channelName = ""
```

```swift
//
//  LoadingScreen.swift
//  Stream Tutorials
//
//  Created by Balaji on 23/03/21.
//

import SwiftUI

struct LoadingScreen: View {
    @Environment(\.colorScheme)
var colorScheme
    var body: some View {

        ZStack{

            Color.primary
                .opacity(0.2)
                .ignoresSafeArea()

            ProgressView()
                .frame(width: 50, height:
50)
                .background(colorScheme
== .dark ? Color.black : Color.white)
                .cornerRadius(8)
        }
    }
}

struct LoadingScreen_Previews:
PreviewProvider {
    static var previews: some View {
        LoadingScreen()
    }
}
```

```
        streamData.channelName =

withAnimation{streamData.createNewChann
el.toggle()}
        })
    }
}

struct CreateNewChannel_Previews:
PreviewProvider {
    static var previews: some View {
        CreateNewChannel()
    }
}
```

```
//
//  Login.swift
//  Stream Tutorials
//
//  Created by Balaji on
23/03/21.
//

import SwiftUI

struct Login: View {

    @EnvironmentObject var
streamData : StreamViewModel

    // changing based on
ColorScheme

@Environment(\.colorScheme)
var colorScheme

    var body: some View {

        VStack{

            TextField("iJustine", text:
$streamData.userName)
                .modifier(ShadowModif
ier())
                .padding(.top,20)
```

```
//
//  MessageRowView.swift
//  Stream Tutorials
//
//  Created by Balaji on 24/03/21.
//

import SwiftUI
import StreamChat

struct MessageRowView: View {

    var messsage: ChatMessage

    var body: some View{

        HStack{

            if messsage.isSentByCurrentUser{
                Spacer()
            }

            HStack(alignment: .bottom,spacing:
10){

                if !messsage.isSentByCurrentUser{
                    UserView(message: messsage)
                        .offset(y: 10.0)
                }

                // Msg With Chat Bubble...
```

```
//
//  OtpLogin.swift
//  Stream Tutorials
//
//  Created by Balaji on 11/05/21.
//

import SwiftUI

struct OtpLogin: View {
    @EnvironmentObject var
model : LoginViewModel
    var body: some View {

        VStack{

            Image("logo")
                .padding(20)

            HStack(spacing: 15){

                TextField("1", text:
$model.countryCode)
                    .keyboardType(.numb
erPad)
                    .padding(.vertical,12)
                    .padding(.horizontal)
                    .frame(width: 50)
                    .background(
        Button(action:
streamData.logInUser, label: {

            HStack{

                Spacer()

                Text("Login")

                Spacer()

                Image(systemName:
"arrow.right")
            }
            .padding(.vertical,10)
            .padding(.horizontal)
            .background(Color.prim
ary)
            .foregroundColor(colorS
cheme == .dark ? .black : .white)
            .cornerRadius(5)
        })
        .padding(.top,20)
        .disabled(streamData.user
Name == "")
        .opacity(streamData.user
Name == "" ? 0.5 : 1)

        Spacer()
      }
      .padding()
   }
}

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

// Creating a Modifier For
Shadow so that it can be used
for some other views...

struct ShadowModifier:
ViewModifier {

   // changing based on
ColorScheme

@Environment(\.colorScheme)
var colorScheme

   func body(content: Content) ->
```

```
            VStack(alignment:
messsage.isSentByCurrentUser ? .trailing : .
leading, spacing: 6, content: {

                Text(messsage.text)


Text(messsage.createdAt,style: .time)
                    .font(.caption)
            })
            .padding([.horizontal,.top])
            .padding(.bottom,8)
            // Current User color is blue and
opposite user color is gray...
            .background(messsage.isSentByCu
rrentUser ? Color.blue :
Color.gray.opacity(0.4))
            .clipShape(ChatBubble(corners:
messsage.isSentByCurrentUser ?
[.topLeft,.topRight,.bottomLeft] :
[.topLeft,.topRight,.bottomRight]))
            .foregroundColor(messsage.isSent
ByCurrentUser ? .white : .primary)
            .frame(width:
UIScreen.main.bounds.width -
150,alignment:
messsage.isSentByCurrentUser ? .trailing : .
leading)


            if messsage.isSentByCurrentUser{
                UserView(message: messsage)
                    .offset(y: 10.0)
            }
        }

        if !messsage.isSentByCurrentUser{
            Spacer()
        }
      }
   }
}

// User View...

struct UserView: View {

   var message: ChatMessage

   var body: some View{

      Circle()
         .fill(message.isSentByCurrentUser ?
Color.blue : Color.gray.opacity(0.4))
         .frame(width: 40, height: 40)
         .overlay(
```

```
RoundedRectangle(cornerRadius:
8)
                    .stroke(model.cou
ntryCode == "" ? Color.gray :
Color("pink"),lineWidth: 1.5)
                )

            TextField("123456789",
text: $model.phNumber)
                .keyboardType(.numb
erPad)
                .padding(.vertical,12)
                .padding(.horizontal)
                .background(


RoundedRectangle(cornerRadius:
8)
                    .stroke(model.ph
Number == "" ? Color.gray :
Color("pink"),lineWidth: 1.5)
                )
        }
        .padding(.top,20)

        Button(action:
model.verifyUser, label: {
            Text("Login")
                .fontWeight(.bold)
                .foregroundColor(.whit
e)
                .padding(.vertical,12)
                .frame(maxWidth: .infi
nity)
                .background(Color("pi
nk"))
                .cornerRadius(8)
        })
        .disabled(model.countryCo
de == "" || model.phNumber ==
"")
        .opacity(model.countryCod
e == "" || model.phNumber ==
"" ? 0.6 : 1)
        .padding(.top,20)

        Spacer()
      }
      .padding()
   }
}

struct OtpLogin_Previews:
PreviewProvider {
   static var previews: some View
{
```

```
some View {

       return content
           .padding(.vertical,10)
           .padding(.horizontal)
           .background(colorScheme
!= .dark ? Color.white :
Color.black)
           .cornerRadius(8)
           .clipped()
           .shadow(color:
Color.primary.opacity(0.04),
radius: 5, x: 5, y: 5)
           .shadow(color:
Color.primary.opacity(0.04),
radius: 5, x: -5, y: -5)
    }
}
```

```
               // Author First Letter...

Text("\(String(message.author.id.first!))")
                   .fontWeight(.semibold)
                   .foregroundColor(message.isSe
ntByCurrentUser ? .white : .primary)
           )
       // COntext Menu For Showing User
Name And Last Active Status...
           .contextMenu(menuItems: {

               Text("\(message.author.id)")

               if message.author.isOnline{
                   Text("Status: Online")
               }
               else{

Text(message.author.lastActiveAt ??
Date(),style: .time)
               }
           })
    }
}
```

```
        OtpLogin()
    }
}
```