```swift
//
//  UploadListView.swift
//  Demo
//
//  Created by Sandeep Kesarwani on 05/12/22.
//

import SwiftUI

struct UploadListView: View {
    var body: some View {

        NavigationView {
            ZStack {
                Image("u").resizable()

                VStack {
                    HStack(spacing: 25){

                        Button(action: {
//                            dissmiss()
                        }, label: {
                            Image(systemName:
                             "arrow.backward").font(.system(size:22,
                             weight:.heavy)).foregroundColor(.white)
                        })

                        Text("Upload List").font(.system(size: 24, weight:
                         .heavy)).foregroundColor(.white)
                        Spacer()
                        Image("cloud_screen").resizable().frame(width: 50,
                         height: 40)

                    }.padding().frame(width: UIScreen.main.bounds.width,
                     height: 65).background(Color("orange"))
//                    HStack{
//                        ZStack{
//                            RoundedRectangle(cornerRadius:
 12).fill(Color.white)
//                            Image("add_stack").resizable().frame(width:
 245, height: 255)
//                        }.background(Color.white).frame(width:
 UIScreen.main.bounds.width/2.1, height: UIScreen.main.bounds.height/3.1)
//                            .cornerRadius(12).shadow(color:
 .gray.opacity(0.5),radius: 1)
//                        Spacer()
//                    }.padding()
                    Spacer()
                }
            }
        }
    }
}

struct UploadListView_Previews: PreviewProvider {
```

```
    static var previews: some View {
//        UploadListView()
        LoginViewExample()
    }
}

struct LoginRequestBodyAuth: Codable {
    let email: String
    let password: String
    let logout_consent:String
}

struct LoginAuthResponse: Codable {
    let access_token: String?
    let email: String?
    let message: String?

}




struct LoginViewExample: View {

    @State var email = ""
    @State var password = ""
    @State var isAuthenticated: Bool = false

    @StateObject private var loginVM = DashBoardLoginViewModel()

       @StateObject private var accountListVM = AuthenticationListService()
    var body: some View {
        NavigationView{


            VStack() {
                Text("Welcome!")
                    .font(.title)
                    .foregroundColor(Color.white)
                    .padding([.top, .bottom], 50)
                    .shadow(radius: 6.0, x: 10, y: 10)

                Image("image")
                    .resizable()
                    .frame(width: 180, height: 180)
                    .clipShape(Circle())
                    .overlay(Circle()
                        .stroke(Color.white, lineWidth: 3))
                    .shadow(radius: 9.0, x: 20, y: 10)
                    .padding(.bottom, 40)

                VStack(alignment: .leading, spacing: 15) {
                    TextField("Username", text: $loginVM.username)
```

```
                    .autocapitalization(.none)
                    .disableAutocorrection(true)
                    .padding()
                    .background(Color(.white))
                    .cornerRadius(25.0)
                    .shadow(radius: 10.0, x: 5, y: 10)

                SecureField("Password", text: $loginVM.password)
                    .textContentType(.password)
                    .padding()
                    .background(Color(.white))
                    .cornerRadius(25.0)
                    .shadow(radius: 10.0, x: 5, y: 10)
            }
            .padding([.leading, .trailing], 50)
            Button(action: {
//                  loginVM.login()
            } ) {
                Text("Forgot password?")
                    .padding([.leading], 150)
                    .foregroundColor(.white)
            }

            Button(action: {
                //                submit()
                loginVM.login()

            }) {


                    Text("Sign In")
                        .font(.headline)
                        .foregroundColor(.white)
                        .padding()
                        .frame(width: 200, height: 60)
                        .background(Color("orange"))
                        .cornerRadius(20.0)
                        .shadow(radius: 10.0, x: 20, y: 10)

            }.padding(.top, 50)
            if loginVM.isAuthenticated{
                Text("Hello User YOu Are Logged In
                 \(loginVM.username)").foregroundColor(.red).onAppear{
                    accountListVM.getDashBoardData()
                }

                Text("\(accountListVM.totalRC)").foregroundColor(.red)
                 .font(.system(size: 35, weight: .heavy))
            }
            else{
                Text("Hello User YOu Are Logged
                 out....").foregroundColor(.yellow)
            }
            Spacer()
            HStack {
```

```
                    NavigationLink(destination: {
                        if loginVM.isAuthenticated{
                            DashboardView()
                        }
                        else{

                        }
                    }, label: {
                        Text("Don't have an account? ")
                            .foregroundColor(.white)
                    })


                    Button(action: {accountListVM.getDashBoardData()}) {
                        Text("Sign Up")
                            .foregroundColor(.yellow)
                    }

                }
            }
            .background(
                LinearGradient(gradient: Gradient(colors: [Color.yellow,
                 Color.green]), startPoint: .top, endPoint: .bottom)
                    .edgesIgnoringSafeArea(.all))
        }
    }
    }


class DashBoardLoginViewModel: ObservableObject {

    @Published var isAuthenticated: Bool = false

    var username: String = "8299544315"
    var password: String = "12345678"
   var logout_consent = 1
    func login() {

        let defaults = UserDefaults.standard

        AuthenticationService().login(username: username, password:
         password) {
            result in
            switch result {
                case .success(let token):
                    defaults.setValue(token, forKey: "access_token")
                    DispatchQueue.main.async {
                        self.isAuthenticated = true
//                        print(token)
                    }
                case .failure(let error):
                self.isAuthenticated = false
                print(error.localizedDescription)
```

```
            }
        }
    }

    func logout() {

           let defaults = UserDefaults.standard
           defaults.removeObject(forKey: "access_token")
           DispatchQueue.main.async {
               self.isAuthenticated = false
           }

       }
}



//


class AuthenticationService {


    func getDashBoardData(token: String, completion: @escaping
     (Result<DashBoardModel, NetworkError>) -> Void) {

        guard let url = URL(string: ApiUtils.loginDashboardApi!) else {
            completion(.failure(.invalidURL))
            return
        }

        var request = URLRequest(url: url)
        request.addValue("Bearer \(token)", forHTTPHeaderField:
         "Authorization")

        URLSession.shared.dataTask(with: request) { (data, response, error)
         in

            guard let data = data, error == nil else {
                completion(.failure(.noData))
                return
            }

            guard let accounts = try?
             JSONDecoder().decode(DashBoardModel.self, from: data) else {
                completion(.failure(.decodingError))
                return
            }

            completion(.success(accounts))



        }.resume()
```

```
        }



    func login(username: String, password: String,completion: @escaping
     (Result<String, AuthenticationError>) -> Void) {

        guard let url = URL(string: ApiUtils.loginAuthurl) else {
            completion(.failure(.custom(errorMessage: "URL is not
             correct")))
            return
        }

        let body = LoginRequestBodyAuth(email: username, password:
         password, logout_consent: "1")

        var request = URLRequest(url: url)
        request.httpMethod = "POST"
        request.addValue("application/json", forHTTPHeaderField:
         "Content-Type")
        request.httpBody = try? JSONEncoder().encode(body)

        URLSession.shared.dataTask(with: request) { (data, response, error)
         in

            guard let data = data, error == nil else {
                completion(.failure(.custom(errorMessage: "No data")))
                return
            }



            guard let loginResponse = try?
             JSONDecoder().decode(LoginAuthResponse.self, from: data) else {
                completion(.failure(.invalidCredentials))
                return
            }

            guard let token = loginResponse.access_token else {
                completion(.failure(.invalidCredentials))
                return
            }

            completion(.success(token))

        }.resume()

    }


}
```

```swift
class AuthenticationListService: ObservableObject {

    @Published var partnerBookRCs = Int()
    @Published var purchasedBooks = Int()

    @Published var guestBooks = Int()

    @Published var ownStacks = Int()

    @Published var totalRC = Int()

    @Published var bookRequestCount = Int()
    @Published var rcFundCounts = Int()
    @Published var successPayCount = Int()



    func getDashBoardData() {

        let defaults = UserDefaults.standard
        guard let token = defaults.string(forKey: "access_token") else {
            return
        }

        AuthenticationService().getDashBoardData(token: token){ (result) in
            switch result {
                case .success(let accounts):
                    DispatchQueue.main.async {
                        self.partnerBookRCs =  accounts.partnerBookRCs
                        self.purchasedBooks = accounts.purchasedBooks
                        self.bookRequestCount = accounts.bookRequestCount
                        self.ownStacks = accounts.ownStacks
                        self.totalRC = accounts.totalRC
                        self.successPayCount = accounts.successPayCount
                        print(accounts)
                    }
                case .failure(let error):
                    print(error.localizedDescription)
            }
        }

    }

}
```