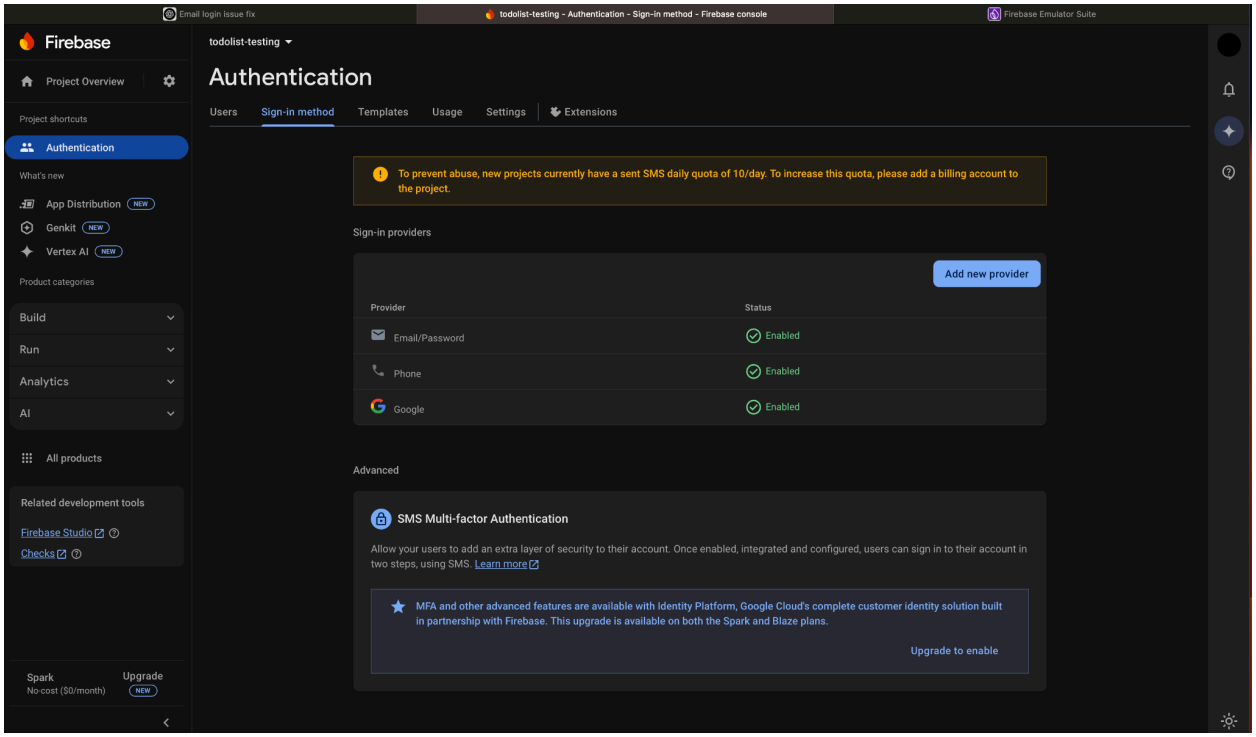
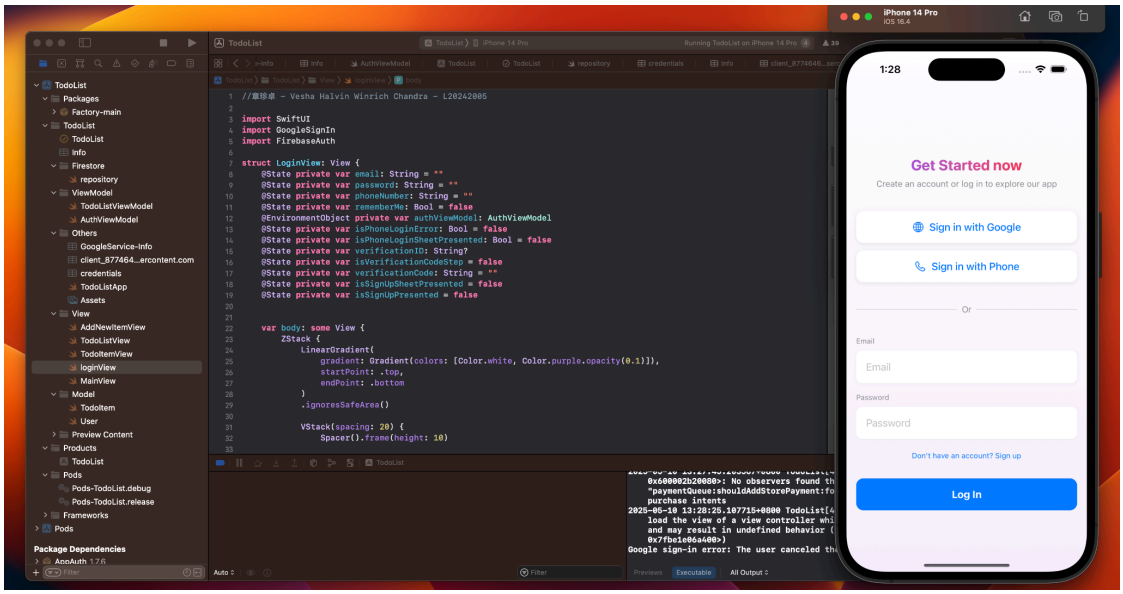
Project Documentation
L20242005 - 章珍卓 Vesha Halvin Winrich Chandra

---

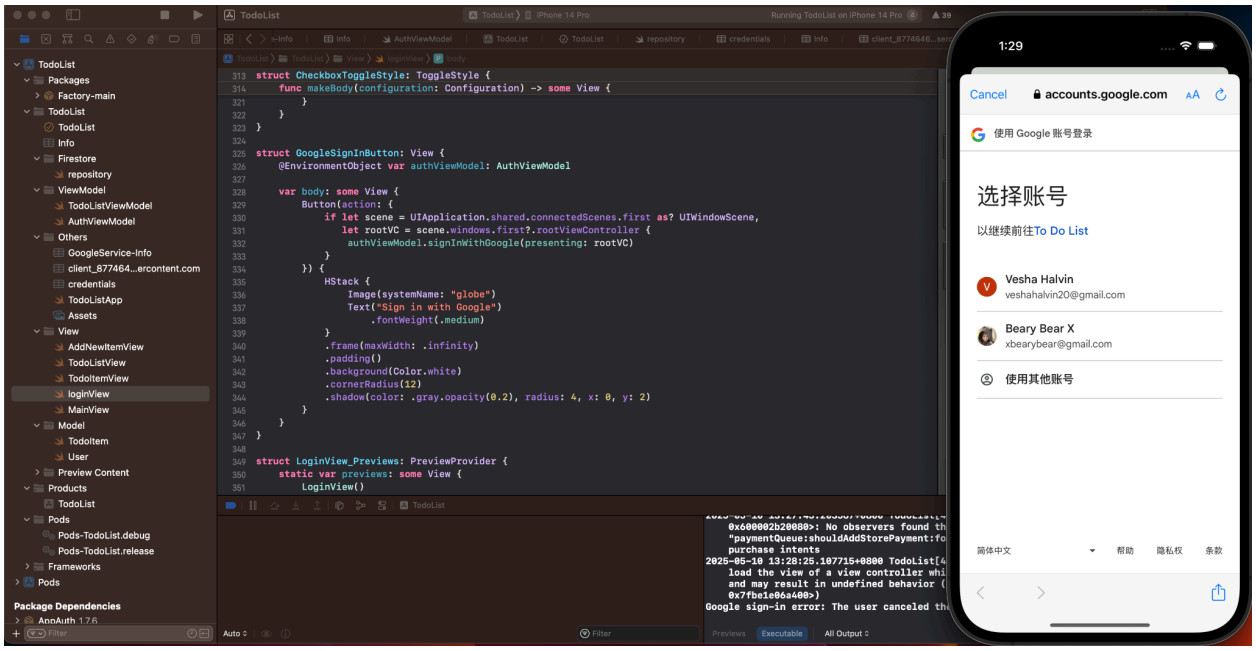
1. Firebase Console Setup
   The project utilizes Firebase for backend services, particularly authentication. The Firebase Console is configured to enable the following sign-in methods: Google Sign-In, Phone Number Authentication, Email/Password Authentication



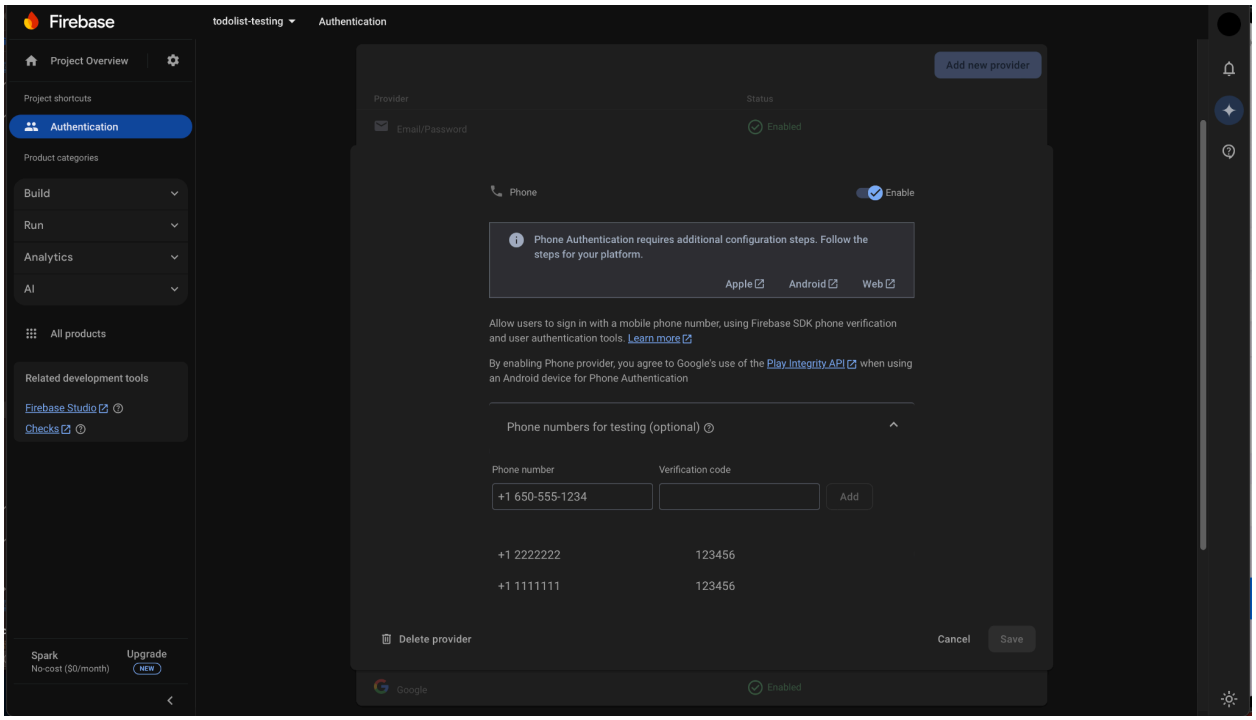
2. There are 3 different kind of login applied to this project. (Google Login, Phone Number, and Email/password)

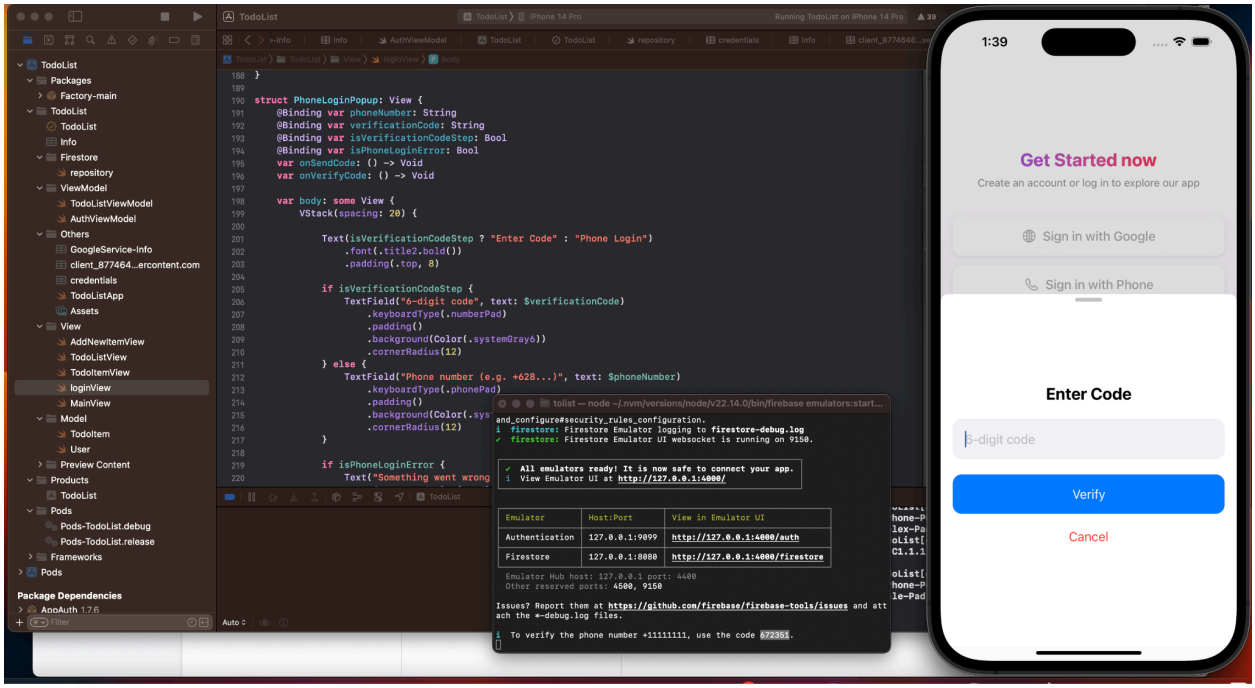


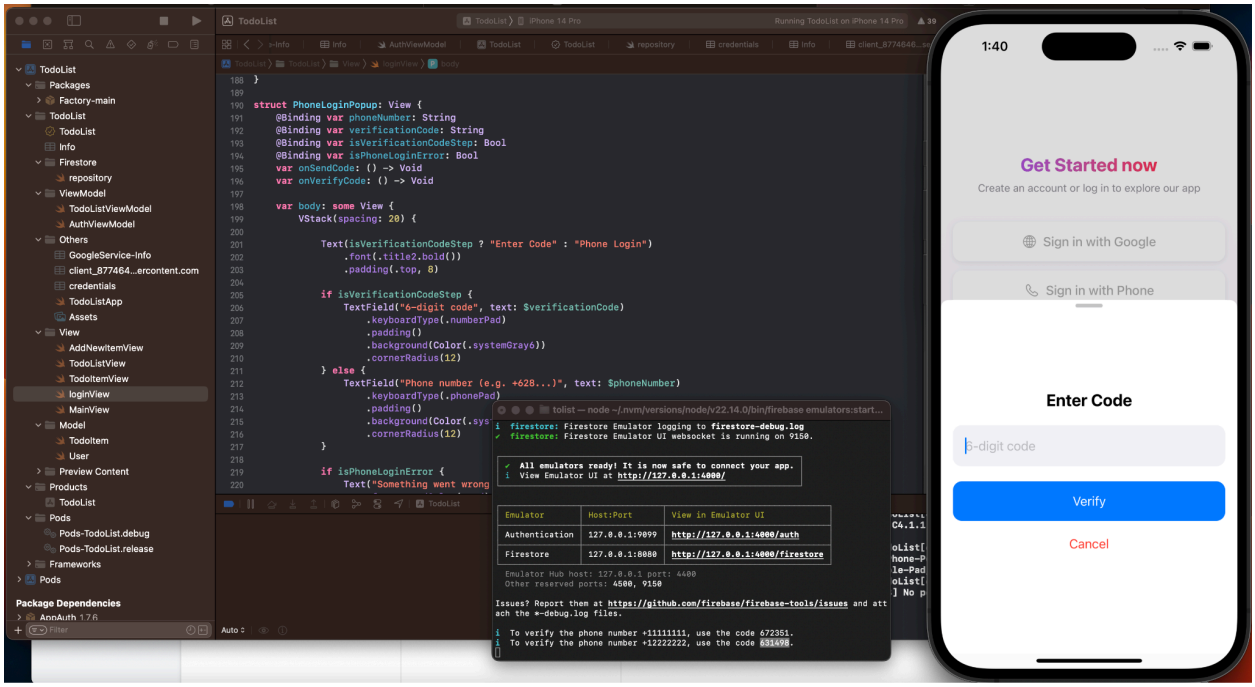
3. Multi-account support is enabled, users can switch or add multiple Google accounts as needed



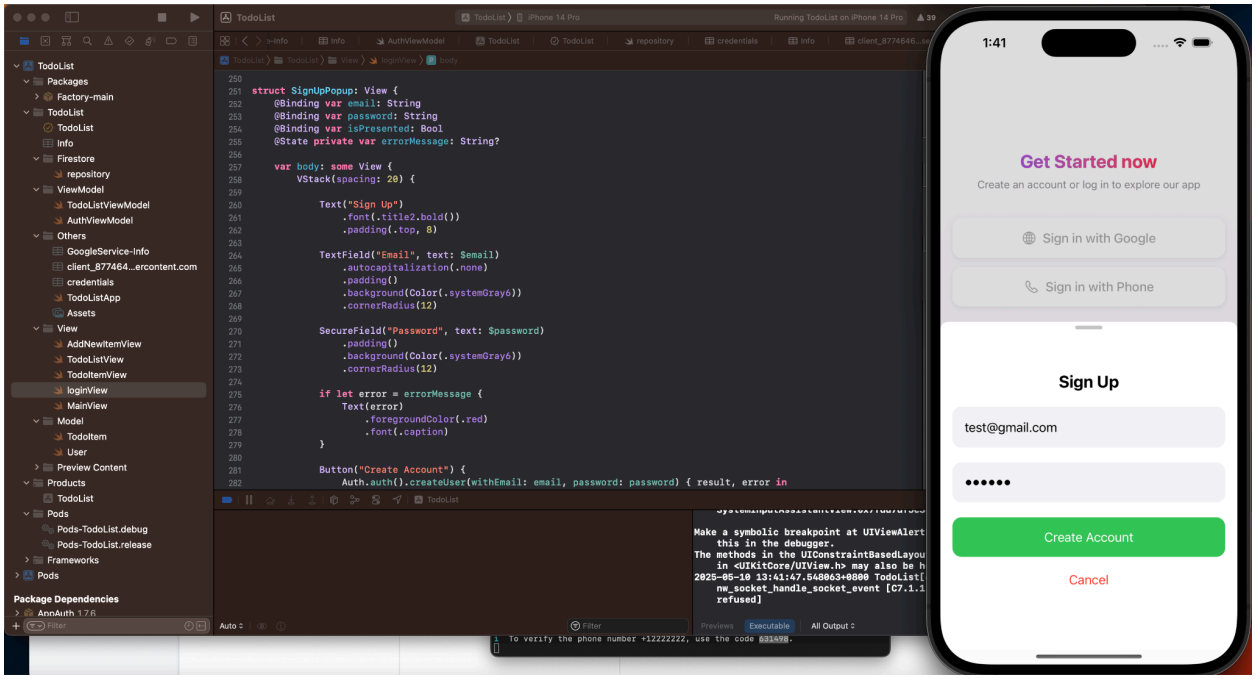
4. Phone number sign-in can be configured through the Firebase Console or dynamically handled within the app.

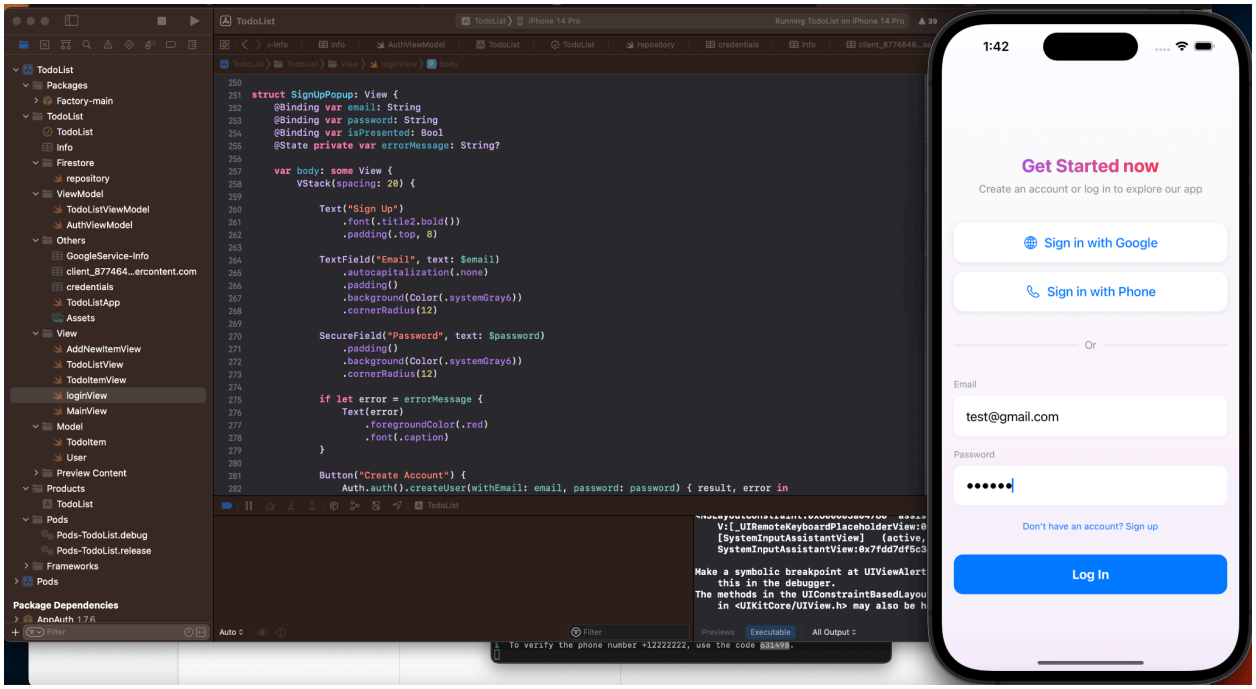



TodoList
Packages
Factory-main
TodoList
TodoList
Info
Firestore
repository
ViewModel
TodoListViewModel
AuthViewModel
Others
GoogleService-Info
client_877464...ercontent.com
credentials
TodoListApp
Assets
View
AddNewItemView
TodoListView
TodoItemView
loginView
MainView
Model
TodoItem
User
Preview Content
Products
TodoList
Pods
Pods-TodoList.debug
Pods-TodoList.release
Frameworks
Pods
Package Dependencies
AppAuth 1.7.6
struct PhoneLoginPopup: View {
@Binding var phoneNumber: String
@Binding var verificationCode: String
@Binding var isVerificationCodeStep: Bool
@Binding var isPhoneLoginError: Bool
var onSendCode: () -> Void
var onVerifyCode: () -> Void
var body: some View {
VStack(spacing: 20) {
Text(isVerificationCodeStep ? "Enter Code" : "Phone Login")
.font(.title2.bold())
.padding(.top, 8)
if isVerificationCodeStep {
TextField("6-digit code", text: $verificationCode)
.keyboardType(.numberPad)
.padding()
.background(Color(.systemGray6))
.cornerRadius(12)
} else {
TextField("Phone number (e.g. +628...)", text: $phoneNumber)
.keyboardType(.phonePad)
.padding()
.background(Color(.systemGray6))
.cornerRadius(12)
}
if isPhoneLoginError {
Text("Something went wrong
tolist — node ~/.nvm/versions/node/v22.14.0/bin/firebase emulators:start...
and_configure#security_rules_configuration.
i firestore: Firestore Emulator logging to firestore-debug.log
✓ firestore: Firestore Emulator UI websocket is running on 9150.
✓ All emulators ready! It is now safe to connect your app.
i View Emulator UI at http://127.0.0.1:4000/
Emulator
Host:Port
View in Emulator UI
Authentication
127.0.0.1:9099
http://127.0.0.1:4000/auth
Firestore
127.0.0.1:8080
http://127.0.0.1:4000/firestore
Emulator Hub host: 127.0.0.1 port: 4400
Other reserved ports: 4500, 9150
Issues? Report them at https://github.com/firebase/firebase-tools/issues and att
ach the *-debug.log files.
i To verify the phone number +11111111, use the code 672351.
1:39
Get Started now
Create an account or log in to explore our app
Sign in with Google
Sign in with Phone
Enter Code
6-digit code
Verify
Cancel

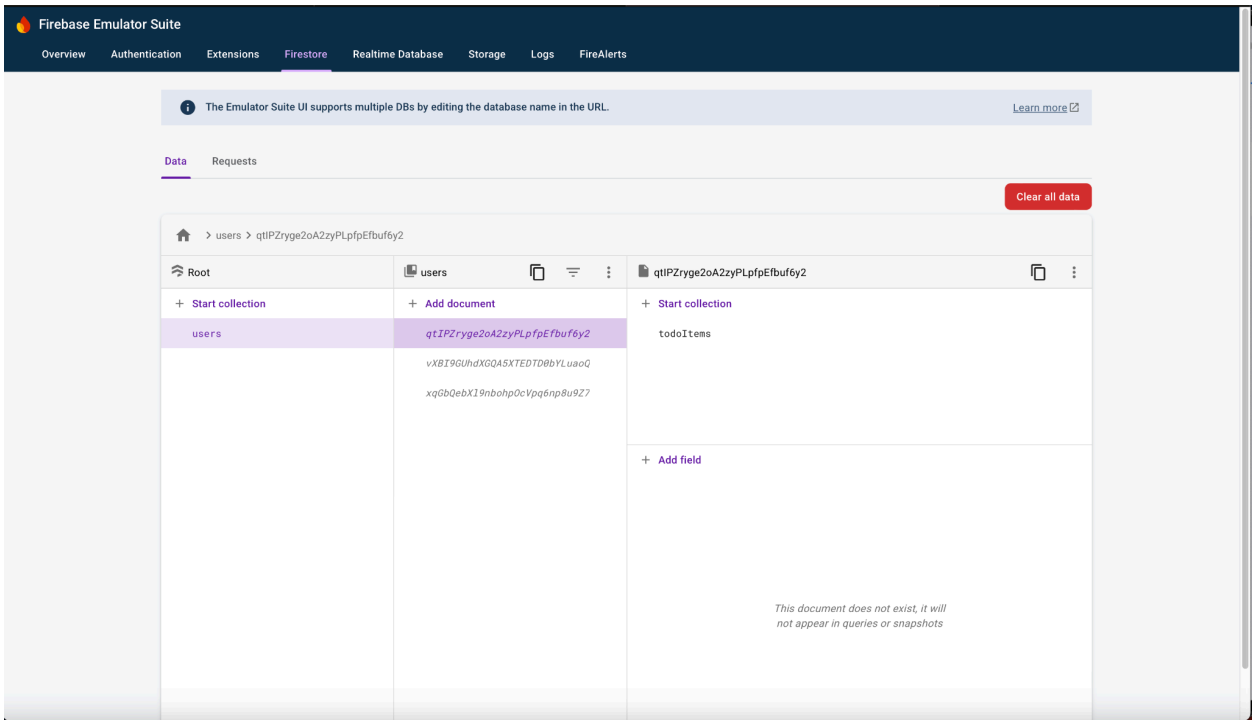

TodoList
Packages
Factory-main
TodoList
TodoList
Info
Firestore
repository
ViewModel
TodoListViewModel
AuthViewModel
Others
GoogleService-Info
client_877464...ercontent.com
credentials
TodoListApp
Assets
View
AddNewItemView
TodoListView
TodoItemView
loginView
MainView
Model
TodoItem
User
Preview Content
Products
TodoList
Pods
Pods-TodoList.debug
Pods-TodoList.release
Frameworks
Pods
Package Dependencies
AppAuth 1.7.6
struct PhoneLoginPopup: View {
@Binding var phoneNumber: String
@Binding var verificationCode: String
@Binding var isVerificationCodeStep: Bool
@Binding var isPhoneLoginError: Bool
var onSendCode: () -> Void
var onVerifyCode: () -> Void
var body: some View {
VStack(spacing: 20) {
Text(isVerificationCodeStep ? "Enter Code" : "Phone Login")
.font(.title2.bold())
.padding(.top, 8)
if isVerificationCodeStep {
TextField("6-digit code", text: $verificationCode)
.keyboardType(.numberPad)
.padding()
.background(Color(.systemGray6))
.cornerRadius(12)
} else {
TextField("Phone number (e.g. +628...)", text: $phoneNumber)
.keyboardType(.phonePad)
.padding()
.background(Color(.systemGray6))
.cornerRadius(12)
}
if isPhoneLoginError {
Text("Something went wrong
tolist — node ~/.nvm/versions/node/v22.14.0/bin/firebase emulators:start...
i firestore: Firestore Emulator logging to firestore-debug.log
✓ firestore: Firestore Emulator UI websocket is running on 9150.
✓ All emulators ready! It is now safe to connect your app.
i View Emulator UI at http://127.0.0.1:4000/
Emulator
Host:Port
View in Emulator UI
Authentication
127.0.0.1:9099
http://127.0.0.1:4000/auth
Firestore
127.0.0.1:8080
http://127.0.0.1:4000/firestore
Emulator Hub host: 127.0.0.1 port: 4400
Other reserved ports: 4500, 9150
Issues? Report them at https://github.com/firebase/firebase-tools/issues and att
ach the *-debug.log files.
i To verify the phone number +11111111, use the code 672351.
i To verify the phone number +12222222, use the code 631498.
1:40
Get Started now
Create an account or log in to explore our app
Sign in with Google
Sign in with Phone
Enter Code
6-digit code
Verify
Cancel

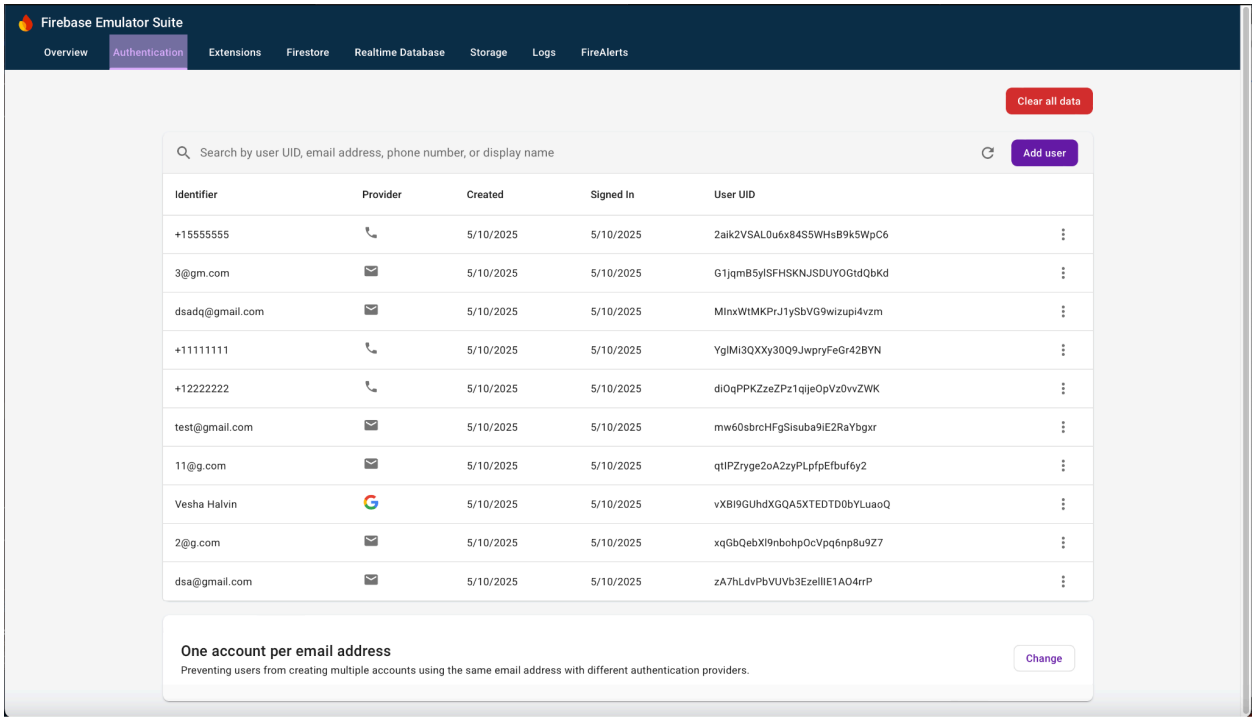
5. Sing in via email/password. If a user attempts to log in with an unregistered email, the app will automatically prompt a Sign-Up Popup to guide the user through account creation.



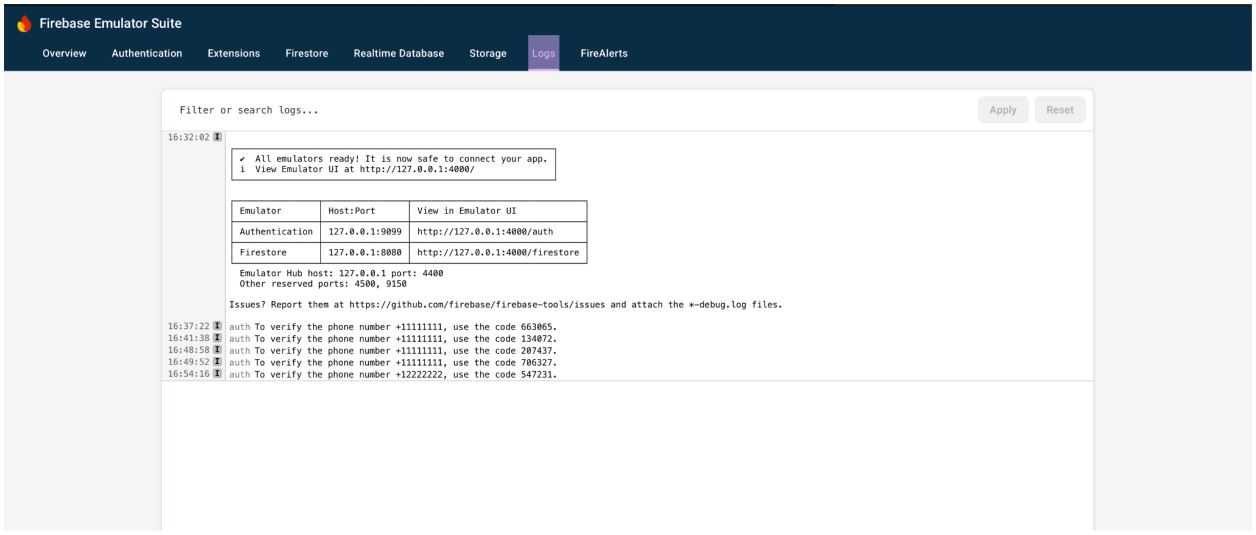


6. Each user's data is stored in a unique dataset tied to their account, This ensures user-specific memory: data displayed or saved within the app is isolated per user.





7. Picture of log

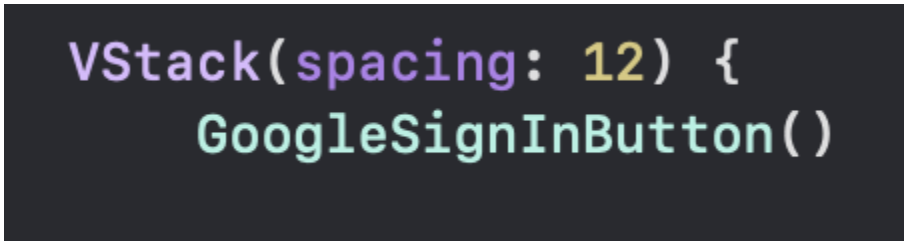


Program Workflow

```swift
//章珍卓 - Vesha Halvin Winrich Chandra - L20242005

import SwiftUI

struct MainView: View {
    @EnvironmentObject var authViewModel: AuthViewModel

    var body: some View {
        if authViewModel.isSignedIn {
            TodoListView()
        } else {
            LoginView()
        }
    }
}


struct MainView_Previews: PreviewProvider {
    static var previews: some View {
        MainView()
    }
}
```

- When the app starts, the `MainView` checks if the user is already signed in by looking at `authViewModel.isSignedIn`.
- If the user is signed in, it navigates to `TodoListView` (the main screen).
- If not, it navigates to the `LoginView`, where the user can choose a login method.

Google Authentication




```
struct GoogleSignInButton: View {
    @EnvironmentObject var authViewModel: AuthViewModel

    var body: some View {
        Button(action: {
            if let scene = UIApplication.shared.connectedScenes.first as? UIWindowScene,
               let rootVC = scene.windows.first?.rootViewController {
                authViewModel.signInWithGoogle(presenting: rootVC)
            }
        }) {
            HStack {
                Image(systemName: "globe")
                Text("Sign in with Google")
                    .fontWeight(.medium)
            }
            .frame(maxWidth: .infinity)
            .padding()
            .background(Color.white)
            .cornerRadius(12)
            .shadow(color: .gray.opacity(0.2), radius: 4, x: 0, y: 2)
        }
    }
}
```

- In the `LoginView`, there is a "Sign in with Google" button.
- When you press this button, the `GoogleSignInButton` component calls `authViewModel.signInWithGoogle(presenting:)`.
- `authViewModel.signInWithGoogle(presenting:)` uses the Google Sign-In SDK to display a Google login screen.
- Once the user logs in successfully, the app returns a Google user credential.

```swift
func signInWithGoogle(presenting: UIViewController) {
    guard let clientID = FirebaseApp.app()?.options.clientID else { return }

    let config = GIDConfiguration(clientID: clientID)
    GIDSignIn.sharedInstance.configuration = config

    GIDSignIn.sharedInstance.signIn(withPresenting: presenting) { result, error in
        if let error = error {
            print("Google sign-in error: \(error.localizedDescription)")
            return
        }

        guard
            let user = result?.user,
            let idToken = user.idToken?.tokenString
        else {
            print("Invalid sign-in result")
            return
        }

        let accessToken = user.accessToken.tokenString

        let credential = GoogleAuthProvider.credential(
            withIDToken: idToken,
            accessToken: accessToken
        )

        Auth.auth().signIn(with: credential) { authResult, error in
            if let error = error {
                print("Firebase sign-in error: \(error.localizedDescription)")
            } else {
                DispatchQueue.main.async {
                    self.isSignedIn = true
                    self.currentUserUID = authResult?.user.uid
                    self.fetchUserData()
                }
            }
        }
    }
}
```

- The user is then signed in, and `authViewModel.isSignedIn` is set to `true`.
- The app will then navigate to `TodoListView`, which displays the main content.

Phone Number Authentication

```
Button(action: {
    isPhoneLoginSheetPresented = true
}) {
    HStack {
        Image(systemName: "phone")
        Text("Sign in with Phone")
            .fontWeight(.medium)
    }
    .frame(maxWidth: .infinity)
    .padding()
    .background(Color.white)
    .cornerRadius(12)
    .shadow(color: .gray.opacity(0.2), radius: 4, x: 0, y: 2)
}
```

```
struct PhoneLoginPopup: View {
    @Binding var phoneNumber: String
    @Binding var verificationCode: String
    @Binding var isVerificationCodeStep: Bool
    @Binding var isPhoneLoginError: Bool
    var onSendCode: () -> Void
    var onVerifyCode: () -> Void

    var body: some View {
        VStack(spacing: 20) {

            Text(isVerificationCodeStep ? "Enter Code" : "Phone Login")
                .font(.title2.bold())
                .padding(.top, 8)

            if isVerificationCodeStep {
                TextField("6-digit code", text: $verificationCode)
                    .keyboardType(.numberPad)
                    .padding()
                    .background(Color(.systemGray6))
                    .cornerRadius(12)
            } else {
                TextField("Phone number (e.g. +628...)", text: $phoneNumber)
                    .keyboardType(.phonePad)
                    .padding()
                    .background(Color(.systemGray6))
                    .cornerRadius(12)
            }

            if isPhoneLoginError {
                Text("Something went wrong.")
                    .foregroundColor(.red)
                    .font(.caption)
            }

            Button(action: isVerificationCodeStep ? onVerifyCode : onSendCode) {
                Text(isVerificationCodeStep ? "Verify" : "Send Code")
                    .frame(maxWidth: .infinity)
                    .padding()
                    .background(Color.blue)
                    .foregroundColor(.white)
```

```
.sheet(isPresented: $isPhoneLoginSheetPresented) {
    PhoneLoginPopup(
        phoneNumber: $phoneNumber,
        verificationCode: $verificationCode,
        isVerificationCodeStep: $isVerificationCodeStep,
        isPhoneLoginError: $isPhoneLoginError,
        onSendCode: {
            PhoneAuthProvider.provider().verifyPhoneNumber(phoneNumber, uiDelegate: nil) { id, error in
                if let error = error {
                    print("Error sending code: \(error.localizedDescription)")
                    isPhoneLoginError = true
                } else {
                    verificationID = id
                    isVerificationCodeStep = true
                }
            }
        },
        onVerifyCode: {
            guard let id = verificationID else { return }
            let credential = PhoneAuthProvider.provider().credential(withVerificationID: id, verificationCode: verificationCode)
            Auth.auth().signIn(with: credential) { result, error in
                if let error = error {
                    print("Phone sign-in failed: \(error.localizedDescription)")
                    isPhoneLoginError = true
                } else {
                    print("User signed in: \(result?.user.uid ?? "unknown")")
                    authViewModel.isSignedIn = true
                    authViewModel.currentUserUID = result?.user.uid
                    isPhoneLoginSheetPresented = false
                }
            }
        }
    ).presentationDetents([.medium])
    .presentationDragIndicator(.visible)
}
```

- In the `LoginView`, there's a "Sign in with Phone" button.
- When you press this button, it presents the `PhoneLoginPopup` where you enter your phone number.
- Once you enter your phone number and click "Send Code", Firebase triggers the `PhoneAuthProvider` to send a verification code to the provided phone number.
- You then enter the received code in the `PhoneLoginPopup`.
- After entering the verification code, the `onVerifyCode` method is triggered.
- The app verifies the code and signs the user in using Firebase's phone authentication.

```
func signInWithTestPhone(phoneNumber: String, verificationCode: String) {
    PhoneAuthProvider.provider().verifyPhoneNumber(phoneNumber, uiDelegate: nil) { verificationID, error in
        if let error = error {
            print("Failed to get verification ID: \(error.localizedDescription)")
            return
        }
        guard let verificationID = verificationID else { return }
            let credential = PhoneAuthProvider.provider().credential(
            withVerificationID: verificationID,
            verificationCode: verificationCode
        )
        Auth.auth().signIn(with: credential) { result, error in
            if let error = error {
                print("Sign-in error: \(error.localizedDescription)")
                return
            }
            DispatchQueue.main.async {
                self.isSignedIn = true
                self.currentUserUID = result?.user.uid
                self.fetchUserData()
                print("Signed in with UID: \(self.currentUserUID ?? "unknown")")
            }
        }
    }
}
```

- If successful, `authViewModel.isSignedIn` is set to `true`, and the app navigates to `TodoListView`.

Email/Password

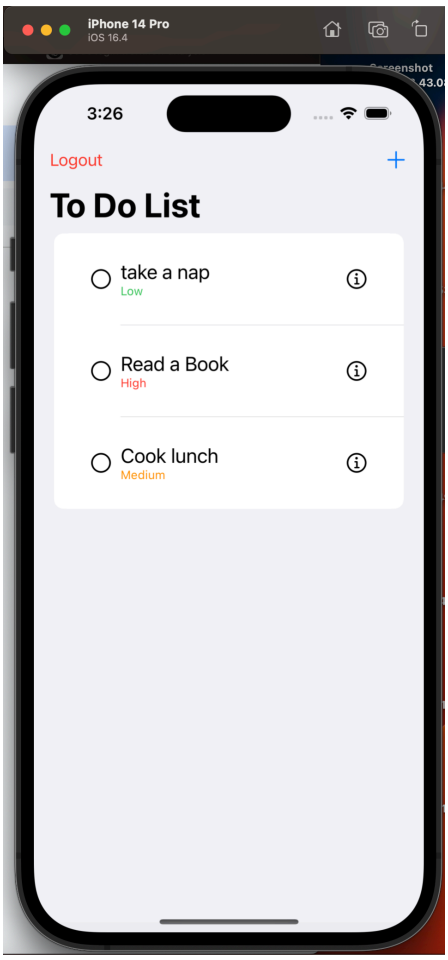
```
        Button(action: {
            authViewModel.signInWithEmail(email: email, password: password) { error in
                if let error = error as NSError? {
                        if error.code == AuthErrorCode.userNotFound.rawValue {
                            DispatchQueue.main.async {
                                isSignUpPresented = true
                            }
                        } else {
                            print("Email login failed: \(error.localizedDescription)")
                        }
                    } else {
                        print("User signed in with email: \(email)")
                    }
            }
        }) {
            Text("Log In")
                .fontWeight(.bold)
                .frame(maxWidth: .infinity)
                .padding()
                .background(Color.blue)
                .foregroundColor(.white)
                .cornerRadius(12)
        }
        .padding(.horizontal)
        .padding(.top, 10)
    }
}
.sheet(isPresented: $isSignUpPresented) {
    SignUpPopup(email: $email, password: $password, isPresented: $isSignUpPresented)
        .presentationDetents([.medium, .large])
        .presentationDragIndicator(.visible)
}
```

- The user enters their email and password in the `LoginView`.
- When the user clicks the "Log In" button, `authViewModel.signInWithEmail(email:password:)` is called.
- If the credentials are correct, the user is signed in via Firebase Authentication, and `authViewModel.isSignedIn` is set to `true`.
- If the email doesn't exist (user is not found), the app shows a message and presents a "Sign Up" popup to create a new account.

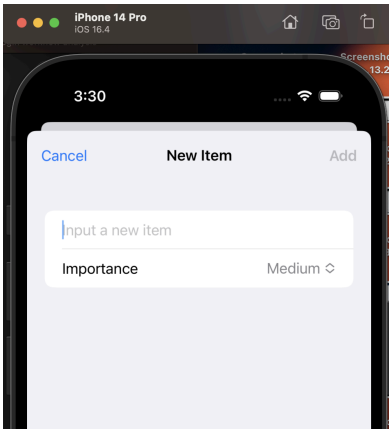
```
func signInWithEmail(email: String, password: String, completion: @escaping (Error?) -> Void) {
        Auth.auth().signIn(withEmail: email, password: password) { result, error in
            if let error = error {
                completion(error)
            } else {
                DispatchQueue.main.async {
                    self.isSignedIn = true
                    self.currentUserUID = result?.user.uid
                    self.fetchUserData()
                }
                completion(nil)
            }
        }
    }
```

- Upon successful login, the user is navigated to `TodoListView`.

TodoListView

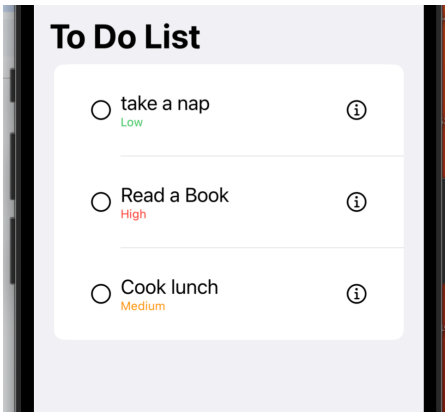


Once a user logs in, they are taken to the TodoListView, which acts as the main screen of the app. It provides a clean and intuitive interface for managing tasks with priority levels.

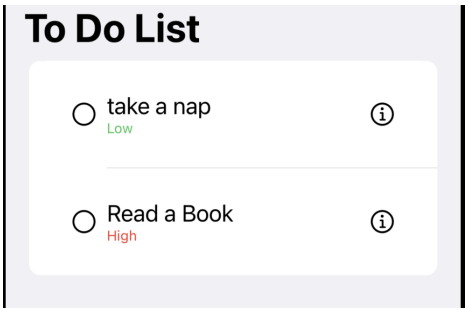


A “plus (+)” button is located in the top right corner. Tapping it opens a form where the user can:

- Enter the task title
- Select the importance level:
    - High 🔴
    - Medium 🟠
    - Low 🟢



- Each task has a circle icon on the left side.
- Tapping the circle triggers a 3-second delay, after which the task is automatically removed from the list and the database.



Logout

```
.toolbar {
    ToolbarItem(placement: .navigationBarLeading) {
        Button(action: logout) {
            Text("Logout")
            .foregroundColor(.red)
        }
    }
```

```swift
func signOut() {
    do {
        try Auth.auth().signOut()
        GIDSignIn.sharedInstance.signOut()

        self.user = nil
        self.isSignedIn = false
        self.currentUserUID = nil

        if Auth.auth().currentUser == nil {
            print("✅ User signed out successfully.")
        } else {
            print("❌ Sign-out failed. Current user still exists: \(Auth.auth().currentUser?.uid ?? "unknown")")
        }

    } catch {
        print("Error signing out: \(error.localizedDescription)")
    }
}
```

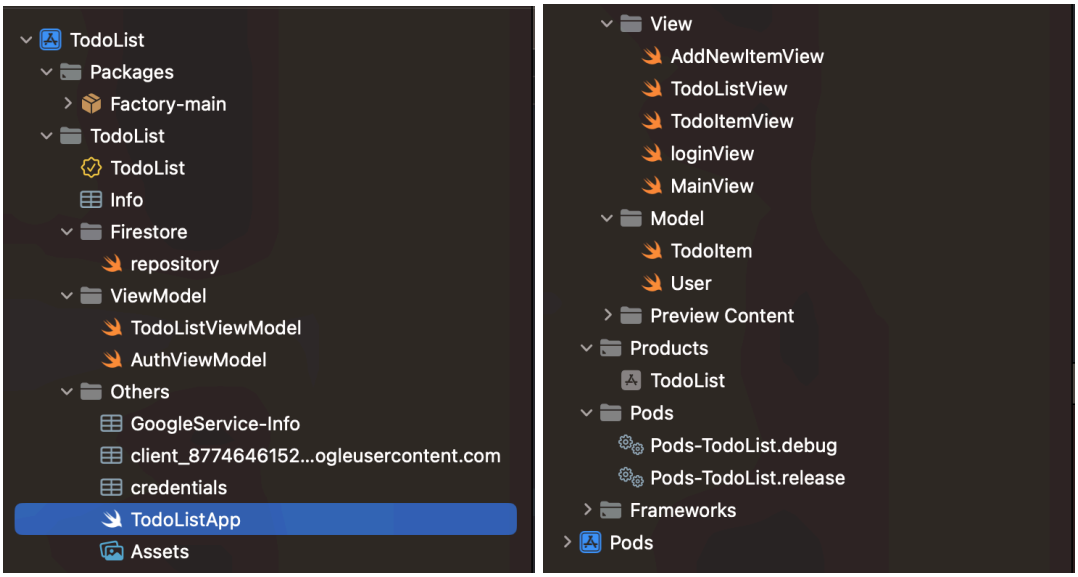
`Button("Logout")`

- Creates a button labeled "Logout" in the top-left corner of the navigation bar.

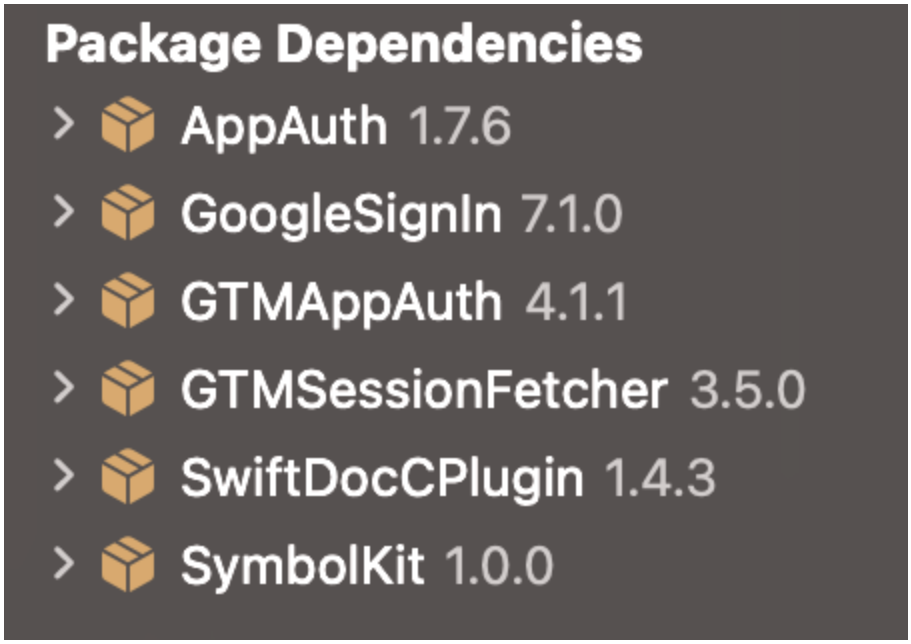
`try Auth.auth().signOut()`

- Call Firebase Authentication's `signOut()` method.
- This clears the current user session, meaning the app no longer considers the user logged in.

Project Structure



Package Dependencies


Package Dependencies
AppAuth 1.7.6
GoogleSignIn 7.1.0
GTMAppAuth 4.1.1
GTMSessionFetcher 3.5.0
SwiftDocCPlugin 1.4.3
SymbolKit 1.0.0