

Firebase User Authentication (iOS)

1

Firebase

A collection of many products

- Cloud Firestore (beta since 2017, GA since 2019)
- **Authentication**
- Cloud Storage
- Realtime DB (beta since 2012, GA since 2014?)
- ML Kit
- Cloud Functions

2

Using Firebase Products in iOS

3

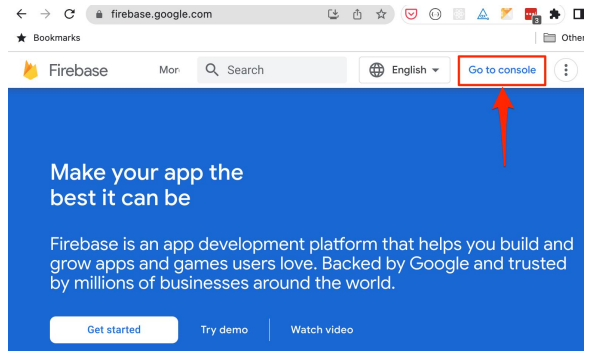
Quick Video Introduction



4

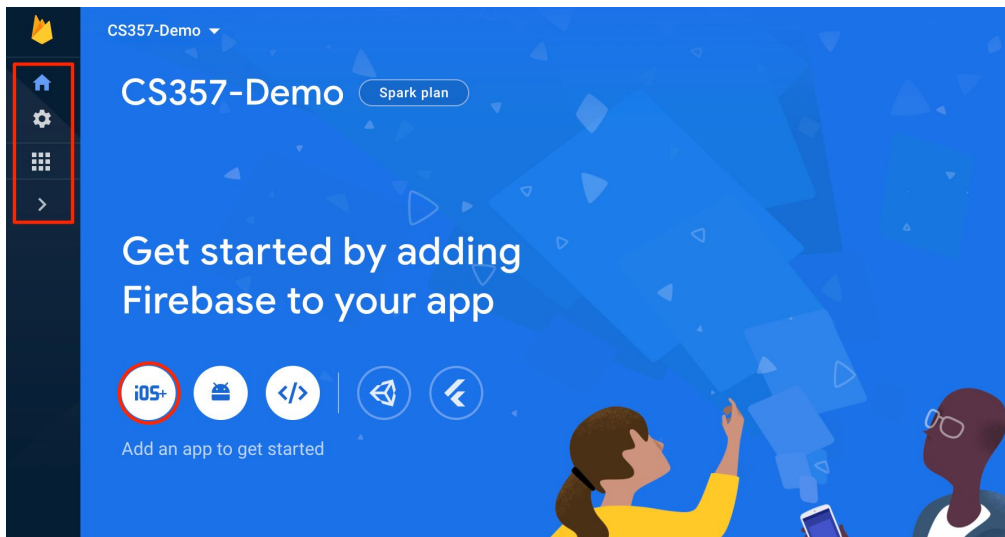
Step 1: Create A Firebase Project

- Login to <https://firebase.google.com>
 - Use your **personal** GMail account
 - Your GVSU GMail (xxxxxx@mail.gvsu.edu) does not have access to Google Developers features
- Go to Firebase Console
- Create a new project
 - Enter project name
 - Disable (or enable) Google Analytics



5

Step 2: Create iOS App



6

Step 3: Copy Bundle Id from Xcode to Firebase

The screenshot shows two side-by-side windows. On the left is the Xcode interface for a project named 'firebase-demo'. The 'Identity' section is visible, showing the 'Bundle Identifier' as 'cis-gvsu-cs357.firebase-demo', which is highlighted with a red box. On the right is the Firebase console 'Add Firebase to your Apple app' page. The 'Register app' step is active, and the 'Apple bundle ID' field contains the same 'cis-gvsu-cs357.firebase-demo' value, also highlighted with a red box. A red arrow in the Xcode window points to the project name in the left sidebar.

7

Step 4: Download Config File To Xcode Project

The screenshot is split into two parts. The left part shows the Firebase console '2 Download config file' step. It features a 'Download GoogleService-Info.plist' button and instructions: 'Move the GoogleService-Info.plist file you just downloaded into the root of your Xcode project for all targets.' Below this, a file icon for 'GoogleService-Info.plist' is shown with a blue arrow pointing to a file explorer window. The file explorer shows a project structure with 'MyApplication' as a subfolder, and 'GoogleService-Info.plist' is highlighted in the 'Products' folder. The right part of the screenshot shows the Xcode project browser with a context menu open over the 'firebase-demo' folder. The menu option 'Add Files to "firebase-demo"...' is selected and highlighted in blue. A red text overlay reads 'Add GoogleService-Info.plist manually to the project'.

8

Step 5: Configure Auth SignIn Provider

The screenshot shows the Firebase Authentication console for a project named 'CS357-Demo'. The left sidebar contains a navigation menu with 'Build' and 'Authentication' highlighted. The main content area is titled 'Authentication' and includes a 'Get started' button. The 'Sign-in method' tab is active, displaying a grid of providers: Native providers (Email/Password, Phone, Anonymous), Additional providers (Google, Facebook, Play Games), and Custom providers (OpenID Connect, SAML). A red arrow points to the 'Email/Password' provider, and a red box highlights the 'Get started' button in the sidebar.

9

Installation

3 Add Firebase SDK [CocoaPods](#) | [Download ZIP](#) | [Unity](#) | [C++](#)

Use [Swift Package Manager](#) to install and manage Firebase dependencies.

1. In Xcode, with your app project open, navigate to **File > Add Packages**
2. When prompted, enter the Firebase iOS SDK repository URL:

```
https://github.com/firebase/firebase-ios-sdk
```
3. **Select the SDK version that you want to use.**
We recommend using the default (latest) SDK version, but you can use an older version, if needed.
4. **Choose the Firebase libraries that you want to use.**

After you click **Finish**, Xcode will automatically begin resolving and downloading your dependencies in the background.

11

Option 1: Xcode Package Manager

1. Open Xcode
2. Select File ⇒ Add Package Dependencies
3. Enter the URL: `https://github.com/firebase/firebase-ios-sdk`
4. Select `firebase-ios-sdk` (be patient...)
5. Then add the following packages to your project
 - a. `FirebaseAuth`
 - b. `FirebaseFirestore`
 - c. `[FirebaseFirestoreSwift]`

12

Option 2: CocoaPods (Old way)

- Install Ruby
- Install RubyGems
- Install Cocoa Pods
- Then, install Firebase Libraries

⟨COCOAPODS⟩

13

Installing CocoaPods (Old way)

```
// On your own Mac using sudo
// From your terminal
sudo gem install cocoapods
```

```
// On your own Mac without sudo
// Add these two lines in your
.zprofile
export GEM_HOME=$HOME/.gem
export PATH=$GEM_HOME/bin:$PATH

// Then from your terminal
gem install cocoapods
```

```
// If installing the latest version (1.11.3) failed,
// fallback to an older version
gem install cocoapods -v 1.8
pod --version
```

Cocoa Pods: Add Firebase SDK (Old way)

```
# Do the following in the same directory
# where you save _____.xcodproj

pod init
vi Podfile

# Install the SDK
pod install

# Open the generated workspace in Xcode
open _____.xcworkspace
```

```
platform :ios, '16.0'

target 'firebase-demo' do
  use_frameworks!

  # Pods for firebase-demo
  pod 'FirebaseAuth'
end
```

Podfile

Search for your libraries at <https://cocoapods.org>

Using Firebase Libraries

Initialize Firebase (SwiftUI)

```
import SwiftUI
import FirebaseCore

class AppDelegate: NSObject, UIApplicationDelegate {
    func application(_ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey : Any]? = nil) -> Bool {
        FirebaseApp.configure()
        return true
    }
}

@main
struct YourApp: App {
    @UIApplicationDelegateAdaptor(AppDelegate.self) var some_variable_name_here
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

17

Initialize Firebase (UIKit)

```
import UIKit
import FirebaseCore

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
    var window: UIWindow?

    func application(_ application: UIApplication,
        didFinishLaunchingWithOptions
        launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        FirebaseApp.configure()
        return true
    }
}
```

AppDelegate.swift

18

Authentication Options

- Email/Password
- Facebook accounts
- GitHub accounts
- Google accounts
- Twitter accounts
- Microsoft accounts
- Yahoo accounts
- Phone numbers
- Online documentation: [firebase.auth](#)

Prerequisites:
Asynchronous Swift

Swift async/await vs. Kotlin suspend

```
// Kotlin
suspend fun doWork() {
    // Perform some work here
}

suspend fun task1() {
    doWork()
}

fun task2() {
    doWork() // Can't call suspendable
             // from non-suspendable
}
```

```
// Swift
func doWork() async {
    // Perform some work here
}

func task1() async {
    await doWork()
}

fun task2() {
    doWork() // Can't call async
             // from non-async
}
```

21

Swift async/await vs. Kotlin suspend

```
// Kotlin
suspend fun doWork() {
    // Perform some work here
}

suspend fun task1() {
    doWork()
}

fun task2() {
    use_a_coroutine_scope_here {
        doWork()
    }
}
```

```
// Swift 5.5 (or later)
func doWork() async {
    // Perform some work here
}

func task1() async {
    await doWork()
}

func task2() {
    Task {
        await doWork()
    }
}
```

22

Async & @Published in ViewModel

```
import SwiftUI
class MyViewModel: ObservableObject {
    @Published var count: Int = 0

    func do_plus_one() {
        Task {
            other_stuff()
            count += 1
        }
    }

    func do_plus_ten() async {
        other_stuff()
        count += 10
    }
}
```

SwiftUI
Publishing changes from background threads is not allowed; make sure to publish values from the main thread (via operators like receive(on:)) on model updates.


```
import SwiftUI
class MyViewModel: ObservableObject {
    @Published var count: Int = 0

    func do_plus_one() {
        other_stuff()
        Task {
            await MainActor.run { // switch to Main thread
                count += 1
            }
        }
    }


    func do_plus_ten() async {
        other_stuff()
        await MainActor.run { // switch to Main thread
            count += 10
        }
    }
}
```

23

Async Functions with Return Value

```
func doWork() async -> Bool { 
    // more code here
    return true // or false
}

func task1() {
    Task {
        if await doWork() {
            print("Yes, it worked")
        }
    }
}
```

```
// This code DOES NOT work!!!
func doWork() -> Bool { 
    Task {
        // check if nth is a prime number
    }
    return true
}
```

24

Firestore Auth: SignIn Option #1 (Async)

```
import SwiftUI
import FirebaseAuth

class MyViewModel: ObservableObject {
    func authenticate(email e:String, password p:String) async -> Optional<String> {
        do {
            let authResult = try await Auth.auth().signIn(withEmail: e, password: p)
            print("User created UID:\(authResult.auth.uid)")
            return nil
        } catch {
            return error.localizedDescription
        }
    }
}
```

25

Firestore Auth: SignIn Option #2 (Async)

```
import SwiftUI
import FirebaseAuth

class MyViewModel: ObservableObject {
    func authenticate(email e:String, password p:String) async -> (Bool,String) {
        do {
            let authResult = try await Auth.auth().signIn(withEmail: e, password: p)
            return (true, authResult.auth.uid)
        } catch {
            return (false, error.localizedDescription)
        }
    }
}
```

26

Invoke the ViewModel Functions from View

```
import SwiftUI

struct MyView: View {
    @ObservedObject var vm: MyViewModel
    var body: some View {
        // Your UI here
        Button("Go") {
            Task {
                let (ok, uid_or_err) = await vm.authenticate("who@test.com", "tops3cr3t")
                if ok {
                    print("Login successful your UID is \(uid_or_err)")
                } else {
                    print("Unable to login: \(uid_or_err)")
                }
            }
        }
    }
}
```

27

Firestore Auth: Create A New Account (Async)

```
import SwiftUI
import FirebaseAuth

class MyViewModel: ObservableObject {
    func newUser(email e:String, password p:String) async -> Optional<String> {
        do {
            let authResult = try await Auth.auth().createUser(withEmail: e, password: p)
            print("User created UID:\(authResult.auth.uid)")
            return nil
        } catch {
            return error.localizedDescription
        }
    }
}
```

28

Firestore Auth: Sign Out

```
import UIKit
import FirebaseAuth

class MyViewModel {
    func doSignout() {
        do {
            try Auth.auth().signOut()
        } catch {
            print("Failed to logout \(error)")
        }
    }
}
```

29

Firestore Auth: SignIn (Old Way)

```
import UIKit
import FirebaseAuth

class MyViewModel {
    private auth = Auth.auth()

    func newUser(email e:String, password p:String) {
        auth.signIn(withEmail: e, password: p) { authOk, error in
            if let authOk {
                // Handle successful login
            } else if let error {
                // Handle successful login
            }
        }
    }
}
```

30

Live Demo

