Firebase User Authentication (iOS)

Firebase

A collection of many products

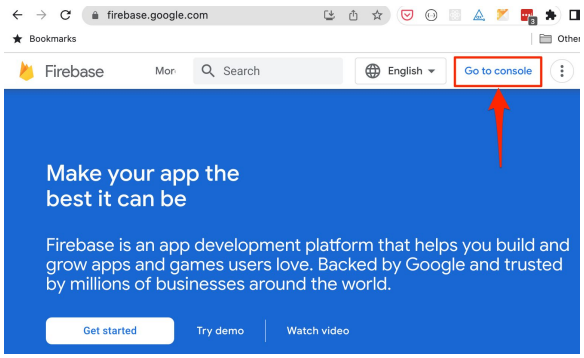
- Cloud Firestore (beta since 2017, GA since 2019)
- **Authentication**
- Cloud Storage
- Realtime DB (beta since 2012, GA since 2014?)
- ML Kit
- Cloud Functions

Using Firebase Products in iOS

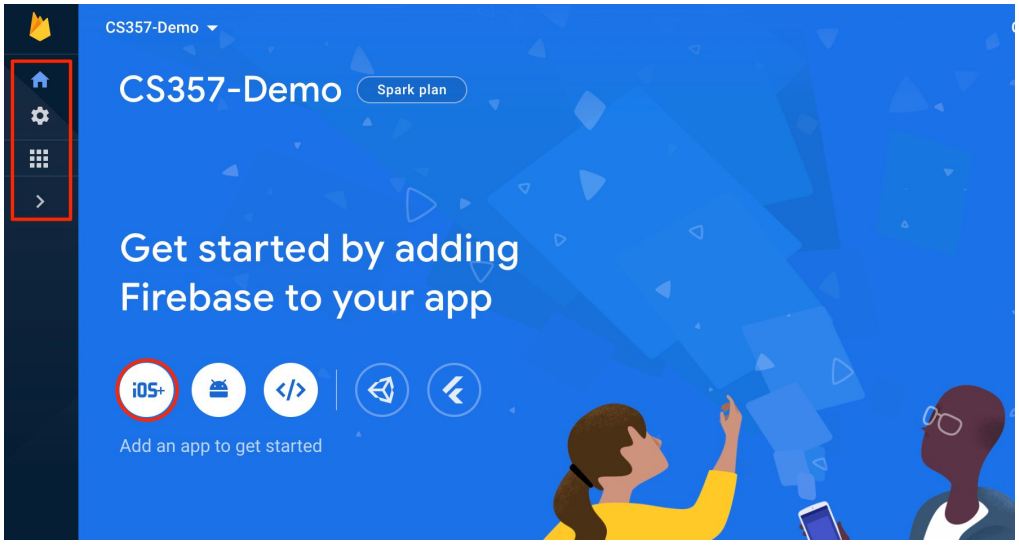
Quick Video Introduction

Step 1: Create A Firebase Project

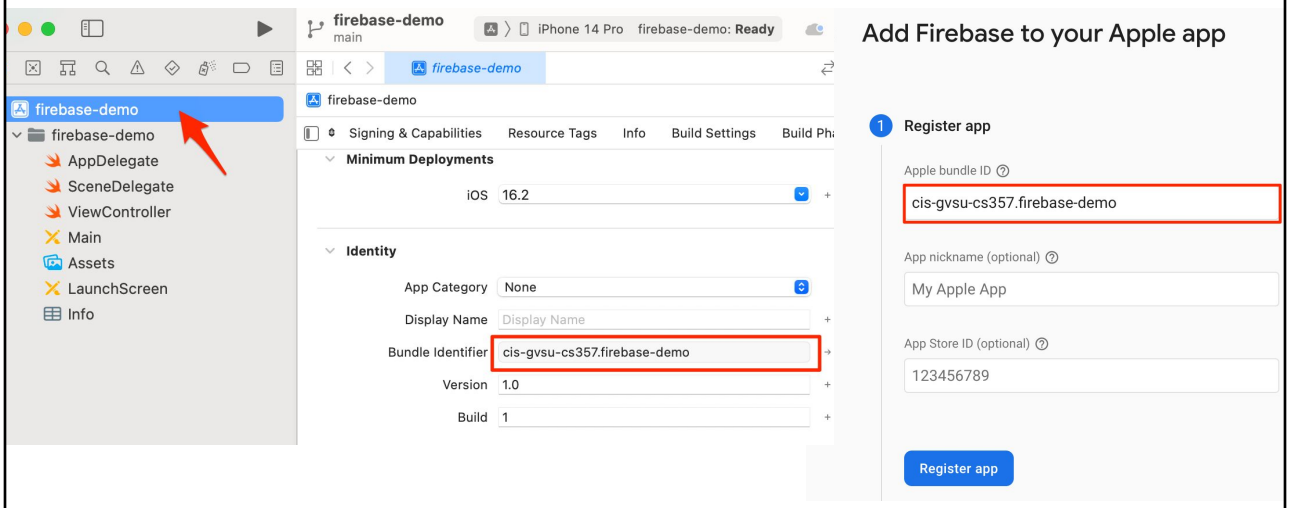
- Login to https://firebase.google.com
  - Use your **personal** GMail account
  - Your GVSU GMail (xxxxxx@mail.gvsu.edu) does not have access to Google Developers features
- Go to Firebase Console
- Create a new project
  - Enter project name
  - Disable (or enable) Google Analytics



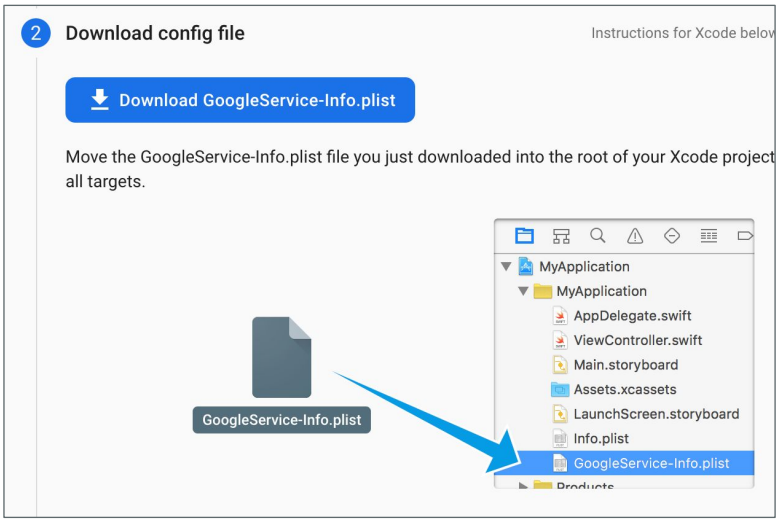
Step 2: Create iOS App

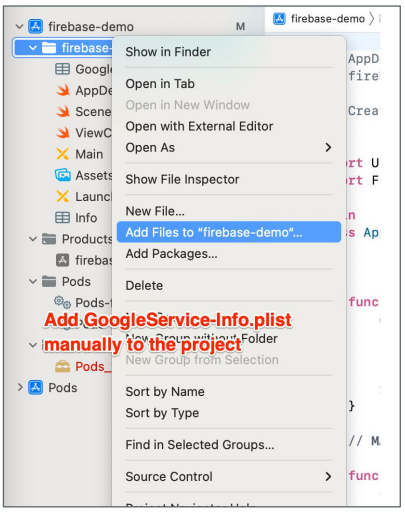


Step 3: Copy Bundle Id from Xcode to Firebase

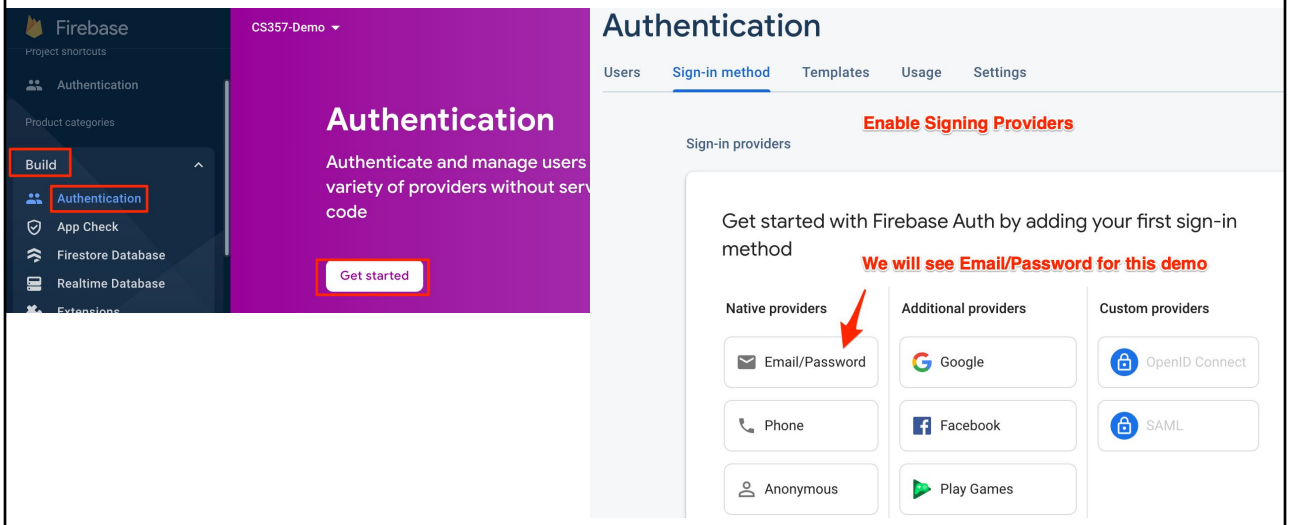


Step 4: Download Config File To Xcode Project

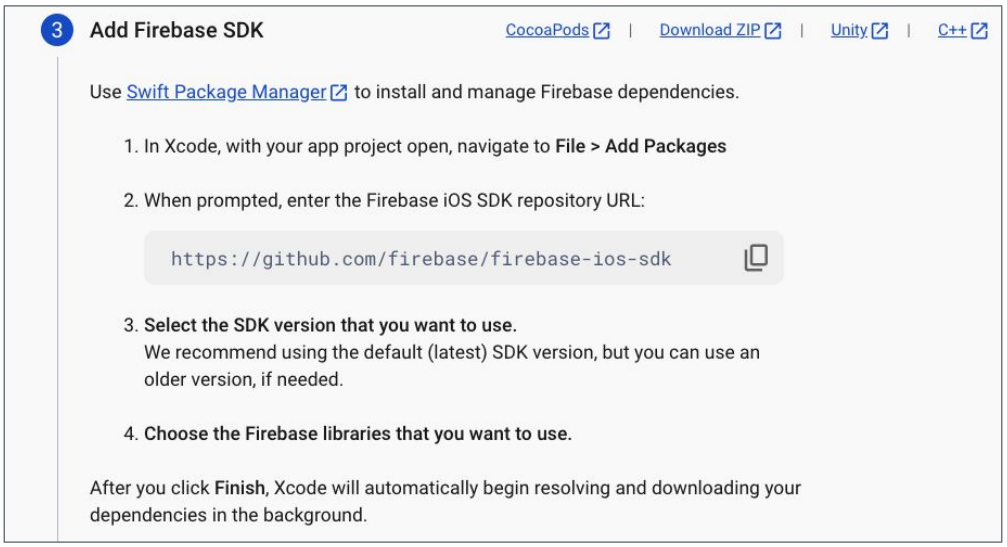




Step 5: Configure Auth SignIn Provider



Installation



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]

Option 2: CocoaPods (Old way)

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

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 _____.xcodeproj

pod init
vi Podfile

# Install the SDK
pod install

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

Podfile

```
platform :ios, '16.0'

target 'firebase-demo' do
  use_frameworks!

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

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()
    }
  }
}
```


Initialize Firebase (UIKit)

AppDelegate.swift

```
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
  }
}
```

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

suspend fun task1() {
   doWork()
}

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

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

func task1() async {
   await doWork()
}

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


Swift async/await vs. Kotlin suspend

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

suspend fun task1() {
   doWork()
}

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

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

func task1() async {
   await doWork()
}

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

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
        }
    }
}
```


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
}
```

Firebase 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
        }
    }
}
```


Firebase 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)
        }
    }
}
```

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)")
                }
            }
        }
    }
}
```


Firebase 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
        }
    }
}
```

Firebase Auth: Sign Out

```
import UIKit
import FirebaseAuth

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


Firebase 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
        }
    }
}
```

Live Demo