

# CocoaPods



---

## Dependency Managers

A *dependency manager* is a tool that makes it easy to add, remove, update, and manage third-party dependencies used by your app.

- For example, instead of reinventing your own networking library, you could simply pull in Alamofire using a dependency manager.
  - You can specify the exact version you want to use, or even a range of acceptable versions.
  - Even if Alamofire gets updated, your app can continue to use the older version until / unless you're ready to update it.

---

## CocoaPods

CocoaPods is a dependency manager designed specifically for Swift and Objective-C projects.

- Contains over 30,000 libraries
- Used in over 2,000,000 apps
- Built on top of Ruby, which ships with all recent versions of Mac OS X
- “pods” are libraries or frameworks added to your project via CocoaPods.

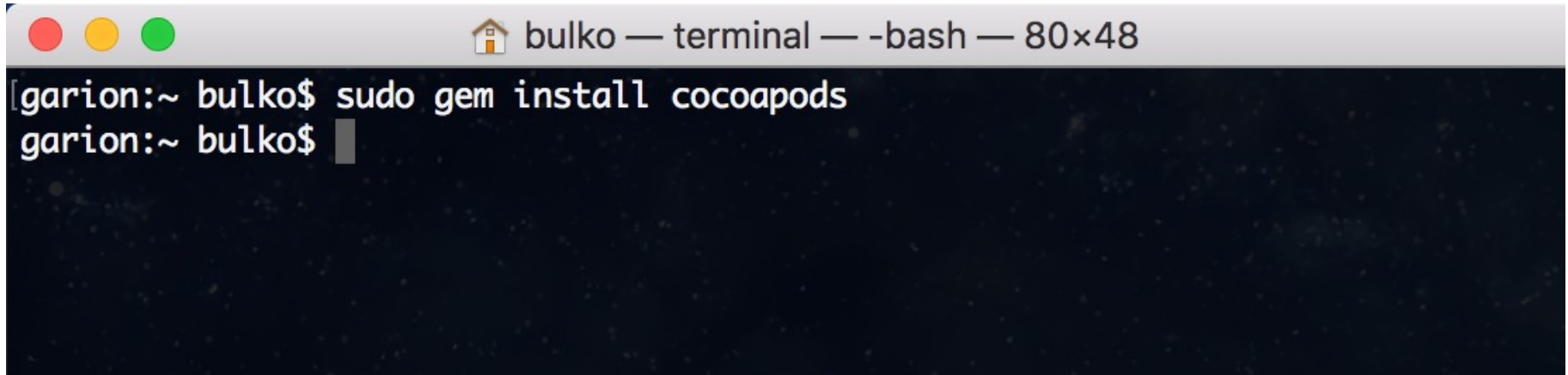
---

## Pods

Prior to iOS 8, CocoaPods were created as “*fat*” *static libraries*. iOS 8 introduced *dynamic frameworks*, which allow code, images, and other assets to be bundled together.

- “fat” means they contained several different code instruction sets (`i386` for the simulator, `armv7` for devices, etc.) but static libraries were not allowed to contain resources like images and assets.
- Dynamic frameworks have namespace classes, and static libraries don’t. If you had two classes with the same name in different static libraries, Xcode would be perfectly happy building a project that has two classes with the same name.

## Installing CocoaPods

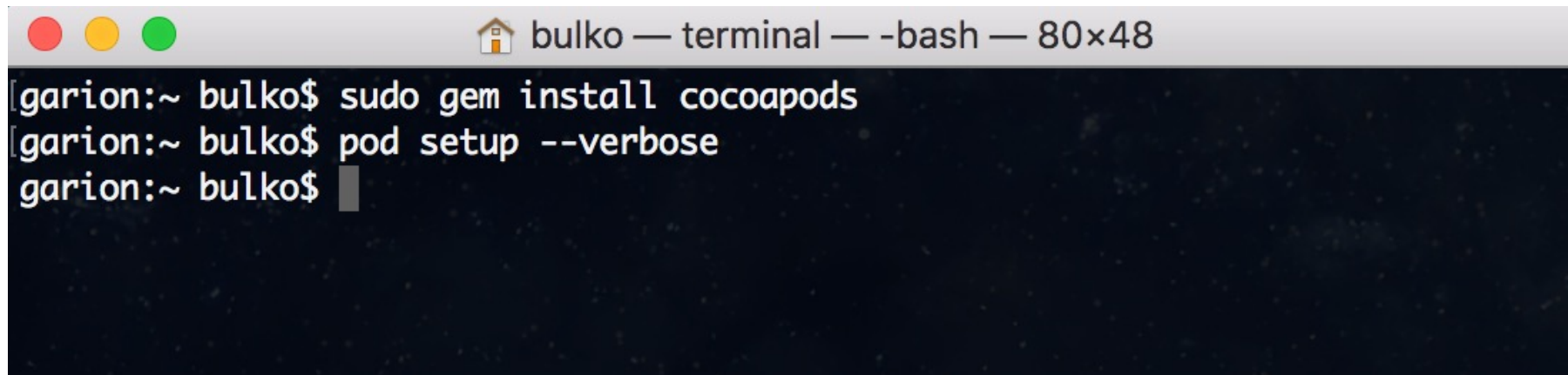
A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, a home icon followed by the text "bulko — terminal — -bash — 80x48" in the center, and a dark background on the right. The terminal content shows a prompt "garion:~ bulko\$" followed by the command "sudo gem install cocoapods" on the same line. A second line shows the prompt "garion:~ bulko\$" with a cursor, indicating the command has been executed.

```
garion:~ bulko$ sudo gem install cocoapods
garion:~ bulko$
```

- `RubyGems` is a packaging system written in the language Ruby. A `gem` is a library packaged in the correct format for `RubyGems` to install it.
- The command `gem install cocoapods` says to install the package called “cocoapods”.
- `sudo` is the Linux command for running a program under superuser privileges. (Enter your root password when prompted.)

So this command simply uses superuser privileges to install CocoaPods.

## Installing CocoaPods

A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, a home icon followed by the text 'bulko — terminal — -bash — 80x48' in the center, and a dark background on the right. The terminal content shows three lines of text: 'garion:~ bulko\$ sudo gem install cocoapods', 'garion:~ bulko\$ pod setup --verbose', and 'garion:~ bulko\$' followed by a cursor. The background of the terminal is dark with a subtle pattern.

```
garion:~ bulko$ sudo gem install cocoapods
garion:~ bulko$ pod setup --verbose
garion:~ bulko$
```

This clones the CocoaPods repository onto your computer.

- This will take a few minutes as it does the unpacking and copying. The `verbose` option is nice because it displays the task's progress, which is reassuring since otherwise you might think it hung.
- The repository will be installed in `~/.cocoapods` .
- <https://guides.cocoapods.org/using/getting-started.html#sudo-less-installation>

# Firestore



---

## What is Firebase?

Firebase is a “Backend-as-a-Service” (BaaS). It is a mobile and web application development platform that manages servers for you so you can focus on your app.

- Real-time database: you can create a database on a server and access it through a Web socket, which is much faster than HTTP.
- File storage: you can save binary files (especially images) securely on Google Cloud Storage
- Authentication: Firebase `auth` has a built-in email/password authentication system you can use for your app.



---

## What is Firebase? (cont.)

Lots of other stuff, too. . .

- Hosting
- Analytics
- Cloud Messaging
- A collection of other Google products, including
  - Remote config
  - Test Lab
  - Crash
  - Notifications
  - Dynamic Links
  - AdMob

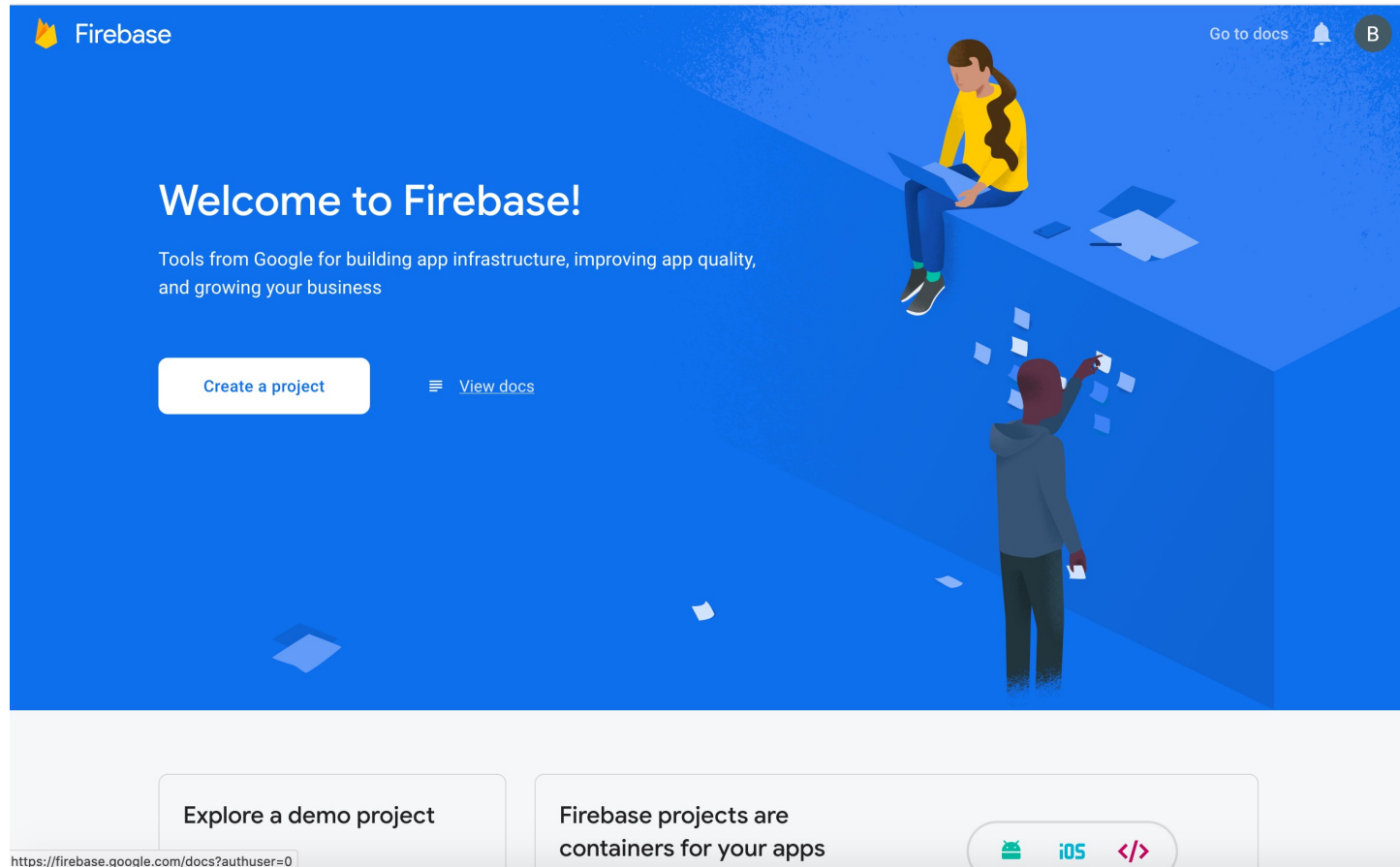
---

## Setting up Firebase

1. Create a Google account, if you don't already have one. (You can use the same one you have for Gmail, Google Drive, etc.)

## Setting up Firebase (cont.)

2. Go to <https://console.firebase.google.com>



3. Add Firebase to your project by clicking on “Create project”

---

## Setting up Firebase (cont.)


### 3. Enter your project name.

× Create a project (Step 1 of 3)

# Let's start with a name for your project

Project name

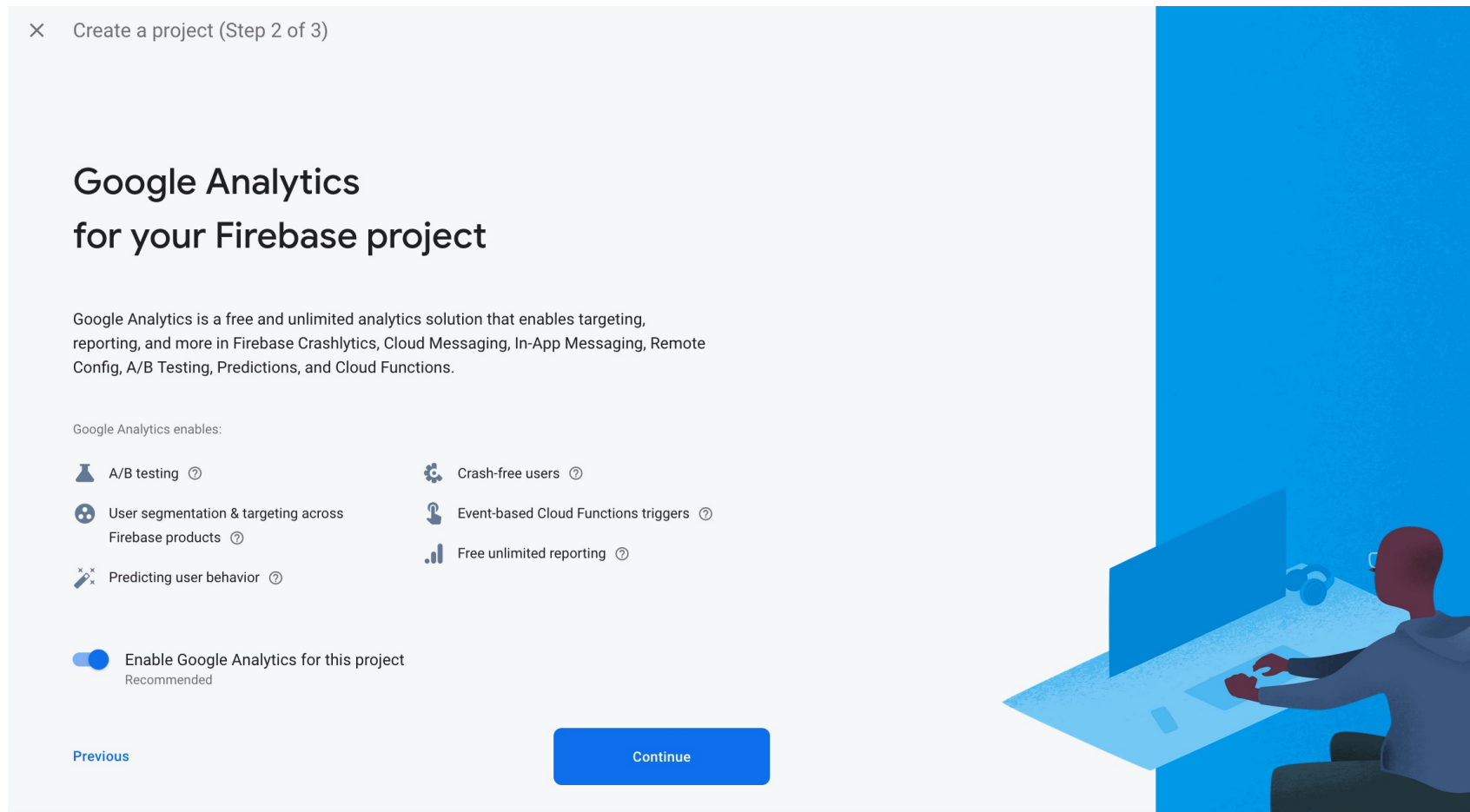
CS371L Homework

 cs371l-homework-82871

Continue

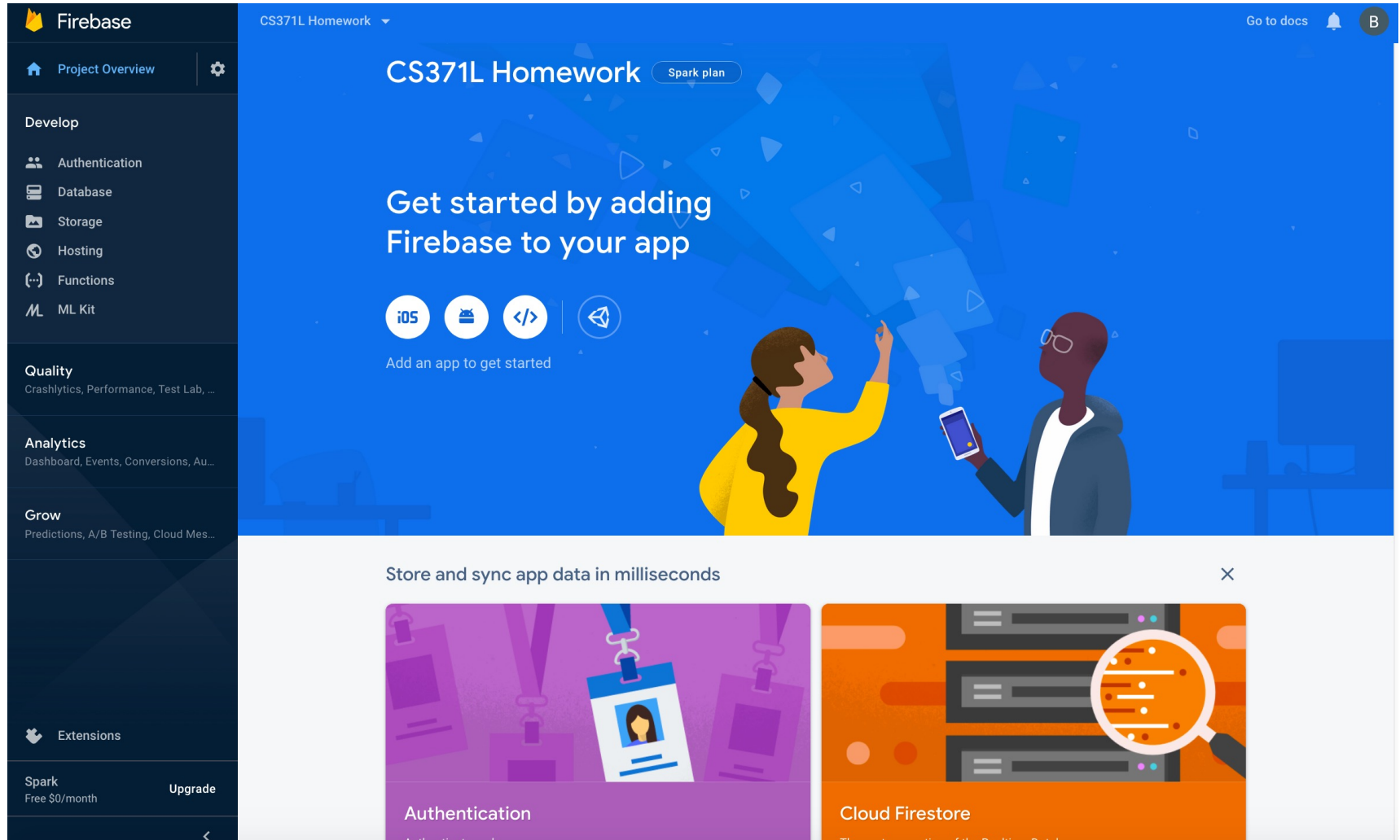
## Setting up Firebase (cont.)

- It will offer to set up Google Analytics. Either accept (and complete an additional screen) or decline and hit “Create Project”. Wait.



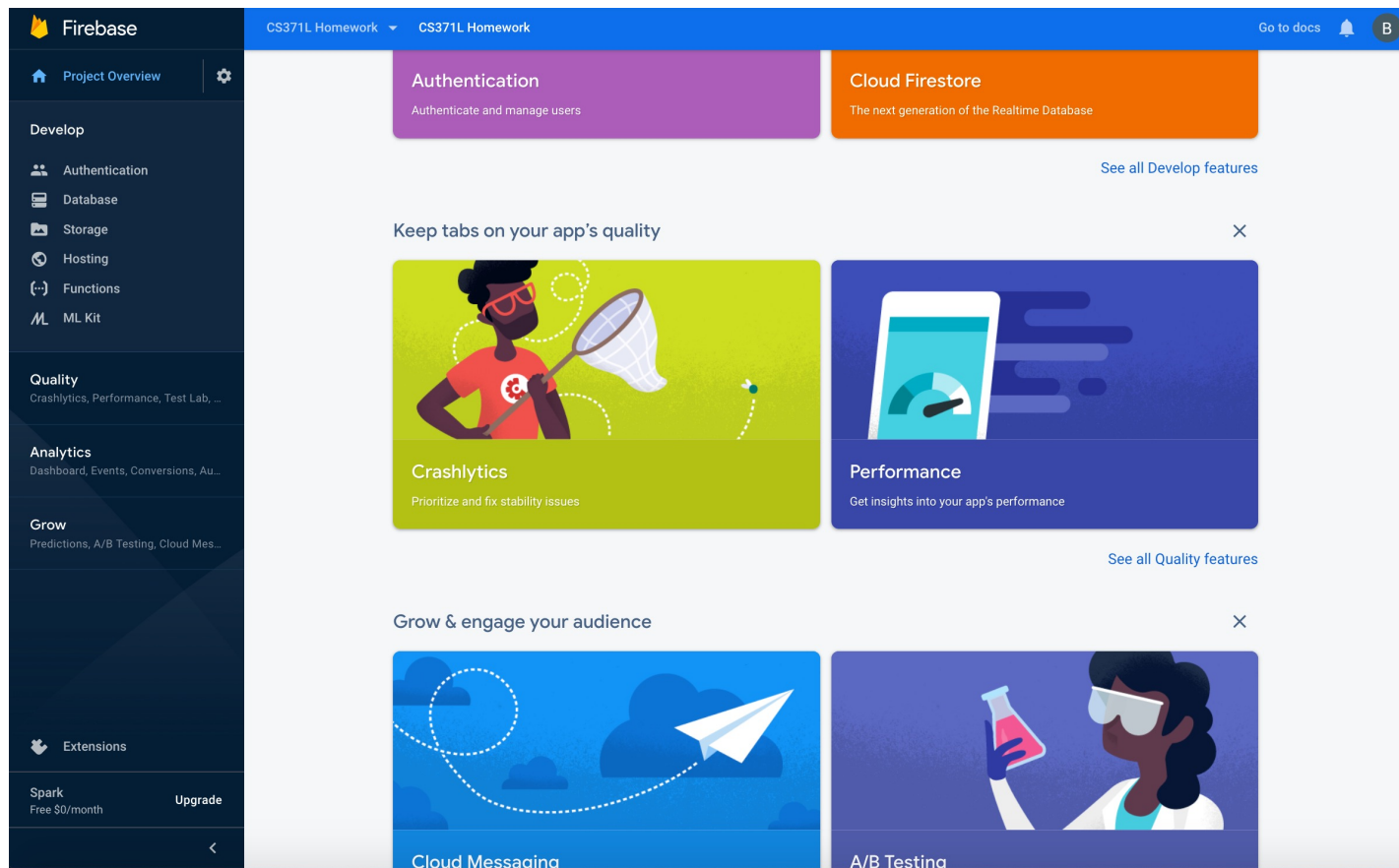
## Setting up Firebase (cont.)

This takes you to the “Get started” screen.



## Setting up Firebase (cont.)

If you scroll to the bottom, you have links to help you manage Authentication, Storage, and any other services you want to include in your project.



## Setting up Firebase (cont.)

Back at the top, click on the "iOS" circle.

The screenshot shows the Firebase console interface for a project named "CS371L Homework". The left sidebar contains navigation links for "Project Overview", "Develop" (with sub-links for Authentication, Database, Storage, Hosting, Functions, and ML Kit), "Quality" (Crashlytics, Performance, Test Lab, ...), "Analytics" (Dashboard, Events, Conversions, Au...), "Grow" (Predictions, A/B Testing, Cloud Mes...), "Extensions", and "Spark" (Free \$0/month, Upgrade). The main content area has a blue header with the project name and a "Spark plan" button. Below this, it says "Get started by adding Firebase to your app" and shows four circular icons for "iOS", "Android", "Web", and "Flutter". A large illustration of two people interacting with a digital interface is also present. At the bottom, there are two featured service cards: "Authentication" (with a purple background and ID cards) and "Cloud Firestore" (with an orange background and server racks).

CS371L Homework

Go to docs

CS371L Homework

Spark plan

Get started by adding Firebase to your app

iOS Android Web Flutter

Add an app to get started

Store and sync app data in milliseconds

Authentication

Cloud Firestore



## Setting up Firebase (cont.)

Enter the Bundle ID for your app. (Remember how to find this?) Click on “Register app”.

× Add Firebase to your iOS app

1

Register app

iOS bundle ID ⓘ

com.company.appname

App nickname (optional) ⓘ

My iOS App

App Store ID (optional) ⓘ

123456789

Register app

2

Download config file

3

Add Firebase SDK

4

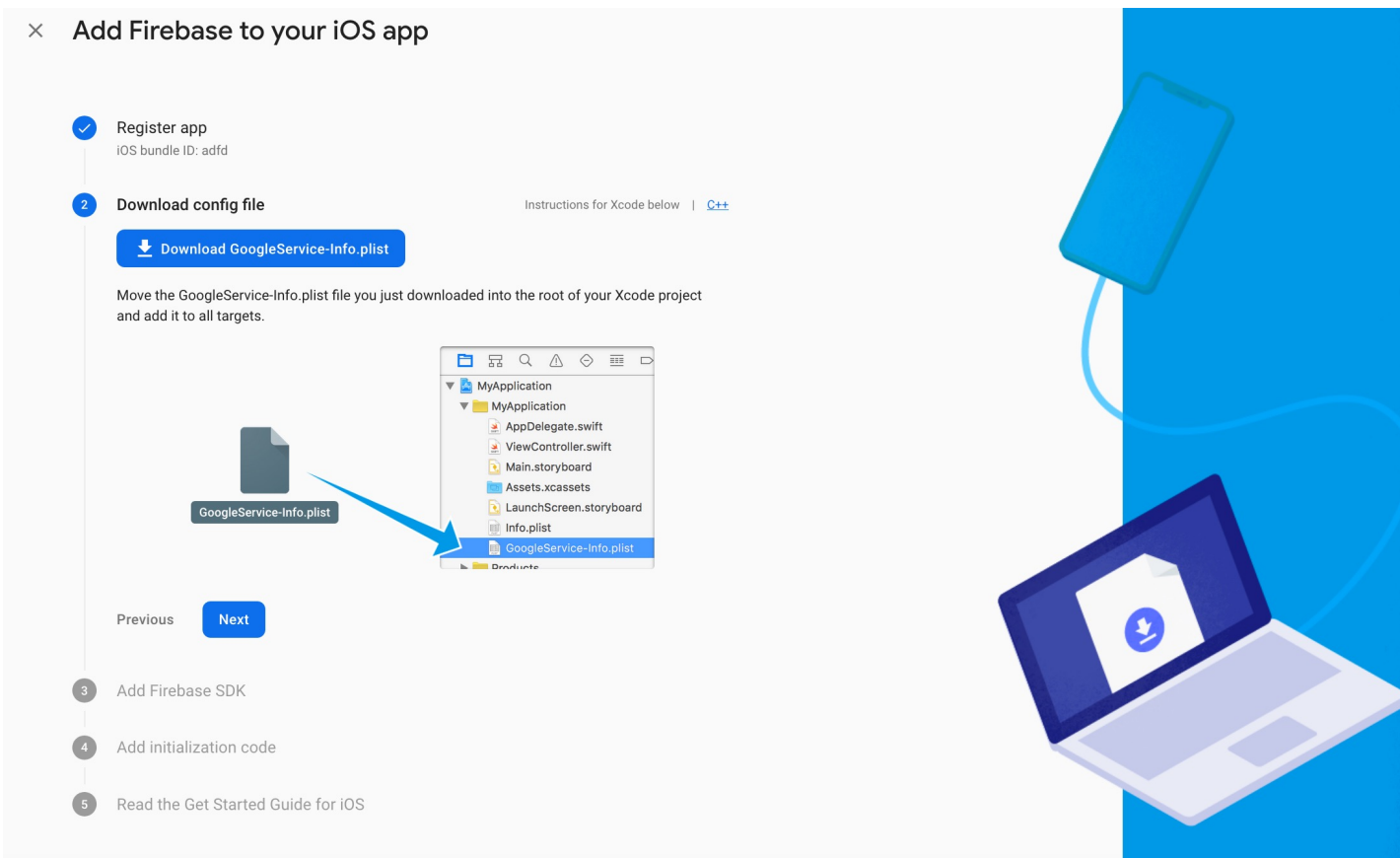
Add initialization code

5

Read the Get Started Guide for iOS

## Setting up Firebase (cont.)

This will create a `GoogleService-Info.plist` file. Follow the instructions, and move it to your project in Xcode. (Be sure to check “Copy Items if needed” in Xcode.)



The screenshot shows the 'Add Firebase to your iOS app' wizard in Xcode. The title bar says 'Add Firebase to your iOS app'. The wizard has five steps: 1. Register app (completed), 2. Download config file (current step), 3. Add Firebase SDK, 4. Add initialization code, and 5. Read the Get Started Guide for iOS.

Step 2: Download config file. It shows a button 'Download GoogleService-Info.plist'. Below it, text says: 'Move the GoogleService-Info.plist file you just downloaded into the root of your Xcode project and add it to all targets.' A diagram shows a file icon labeled 'GoogleService-Info.plist' being moved into a project folder named 'MyApplication'. The project folder contains files like AppDelegate.swift, ViewController.swift, Main.storyboard, Assets.xcassets, LaunchScreen.storyboard, Info.plist, and the newly added GoogleService-Info.plist. A 'Next' button is visible.

Instructions for Xcode below | [C++](#)

Previous Next

## Setting up Firebase (cont.)

Clicking “Next” will give you the instructions on how to add the Firebase SDK to your project. This is the pod stuff we talked about earlier.

×

Add Firebase to your iOS app

✓

Register app  
iOS bundle ID: adfd

✓

Download config file

3

Add Firebase SDK

Instructions for CocoaPods | [Download ZIP](#) [C++](#)

Google services use [CocoaPods](#) to install and manage dependencies. Open a terminal window and navigate to the location of the Xcode project for your app.

Create a Podfile if you don't have one:

```
$ pod init
```

Open your Podfile and add:

```
# add pods for desired Firebase products
# https://firebase.google.com/docs/ios/setup#available-pods
```

Save the file and run:

```
$ pod install
```

This creates an `.xcworkspace` file for your app. Use this file for all future development on your application.

Previous


Next

4

Add initialization code

5

Read the Get Started Guide for iOS



## Setting up Firebase (cont.)

Clicking “Next” from here will show you how to connect Firebase when your app starts. Click “Continue to console”. Build and run your app.

×

Add Firebase to your iOS app

✓

Register app  
iOS bundle ID: adfd

✓

Download config file

✓

Add Firebase SDK

4

Add initialization code

To connect Firebase when your app starts up, add the initialization code below to your main AppDelegate class.

☒ Swift ☐ Objective-C

```
import UIKit
import Firebase

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

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

Previous

Next

---

## Using Firebase in your code

All data stored in Firebase is formatted in JSON. Suppose you created a JSON structure in the database like this:

```
{
  "books": {
    "LOTR": {
      "title": "Lord of the Rings",
      "author": "Tolkien"
    },
    "Potter": {
      "title": "Harry Potter and the
                Sorcerer's Stone",
      "author": "Rowling"  },
    }
  }
```

---

## Using Firebase in your code

All Firebase keys map to *paths*.

That means you can refer to all of the books using the name

```
books
```

and you can refer to the second book this way:

```
books/Potter
```

Note that this means you have to choose these tokens wisely. (Avoid spaces, special characters, etc.)

---

## Using Firebase in your code

To access a Firebase database in your code, add the following to a ViewController:

```
let ref = Database.database().reference(withPath:
    "books")
```

In Firebase terminology, properties are referred to as references. The idea is that they refer to a location in your Firebase database.

---

## Using Firebase in your code (cont.)

To add something to your database:

```
// Create a new book as a dictionary object
```

```
let newItem = [  
    "title": "Horton Hears a Who",  
    "author": "Seuss")  
]
```

```
// Create a new child reference with key "Horton"
```

```
let newItemRef = self.ref.child("Horton")
```

```
// Use setValue to save it to the database
```

```
newItemRef.setValue(newItem)
```



---

## Accessing your database

If you try to run your app now, it won't save anything because you haven't given yourself write access to the database.

From the web console for your app, in the list of services at the bottom of the screen, find “Database” and click on “Get started”.

## Accessing your database (cont.)

Click on "Cloud Firestore", and then "Create Database".

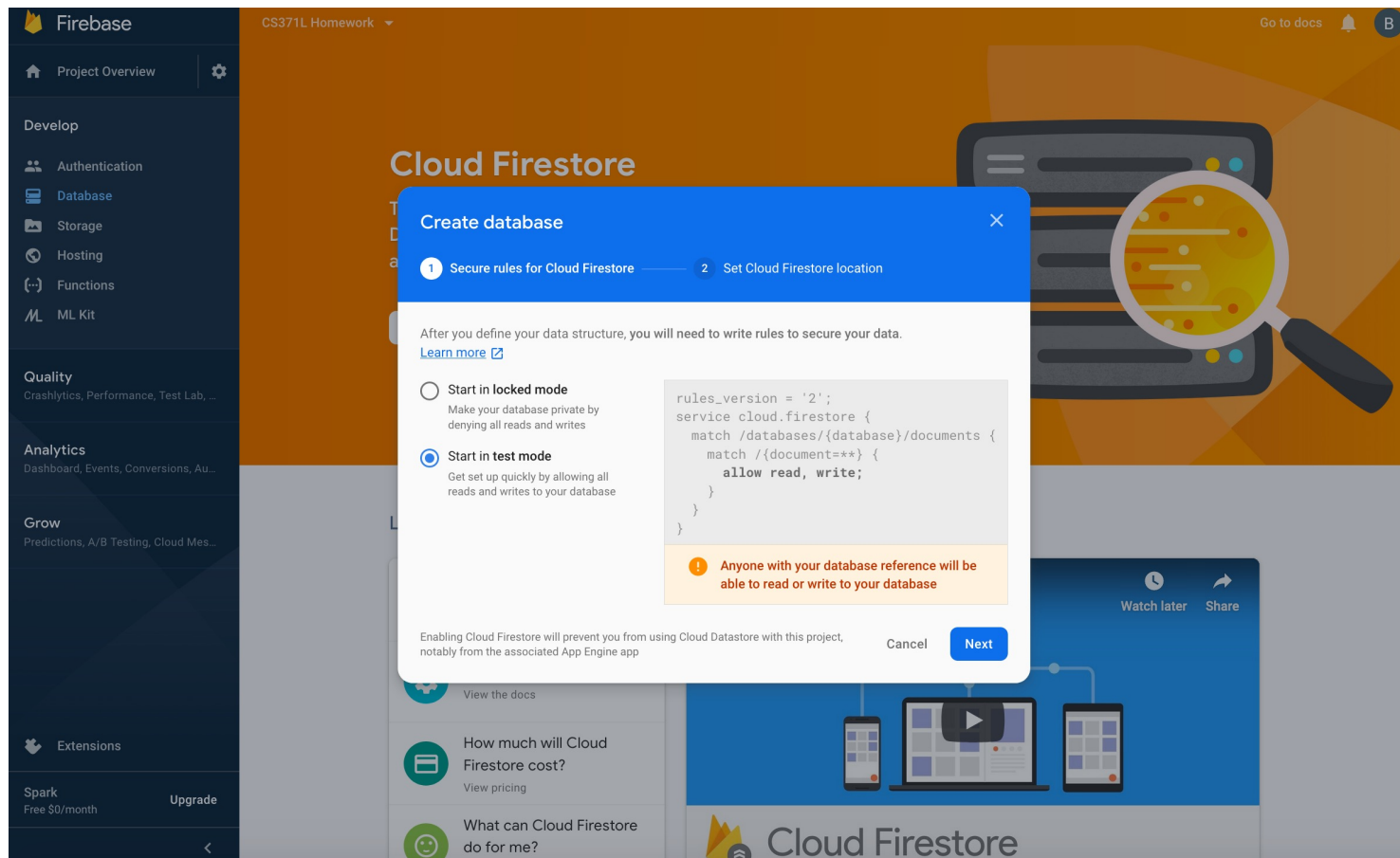
The screenshot displays the Firebase console interface. On the left is a dark blue sidebar with the 'Firebase' logo at the top. Below it, the 'Project Overview' section includes a home icon and a settings gear. The 'Develop' section lists 'Authentication', 'Database' (highlighted in blue), 'Storage', 'Hosting', 'Functions', and 'ML Kit'. The 'Quality' section includes 'Crashlytics, Performance, Test Lab, ...'. The 'Analytics' section includes 'Dashboard, Events, Conversions, Au...'. The 'Grow' section includes 'Predictions, A/B Testing, Cloud Mes...'. At the bottom of the sidebar are 'Extensions', 'Spark' (Free \$0/month), and an 'Upgrade' button.

The main content area has an orange header with 'CS371L Homework' and a dropdown arrow on the left, and 'Go to docs', a bell icon, and a user profile icon 'B' on the right. The central banner features the 'Cloud Firestore' title, the text 'The next generation of the Realtime Database with more powerful queries and automatic scaling', and a 'Create database' button. To the right of the text is an illustration of three server racks with a magnifying glass over the middle one, showing a yellow and orange data visualization.

Below the banner is a 'Learn more' section. It contains a list of links on the left: 'Find out if Cloud Firestore is right for you' (Compare databases), 'How do I get started?' (View the docs), 'How much will Cloud Firestore cost?' (View pricing), and 'What can Cloud Firestore do for me?'. On the right is a video player titled 'Introducing Cloud Firestore' with 'Watch later' and 'Share' buttons. The video thumbnail shows a cloud icon connected to a laptop and two smartphones, with a play button in the center. The video title 'Cloud Firestore' is partially visible at the bottom.

## Accessing your database (cont.)

Select “Start in **test mode**” to give all users who reference your database read and write access.



---

## Retrieving data

Add a call to `ref.observe` to `viewDidLoad()` (or wherever you want the app to update itself):

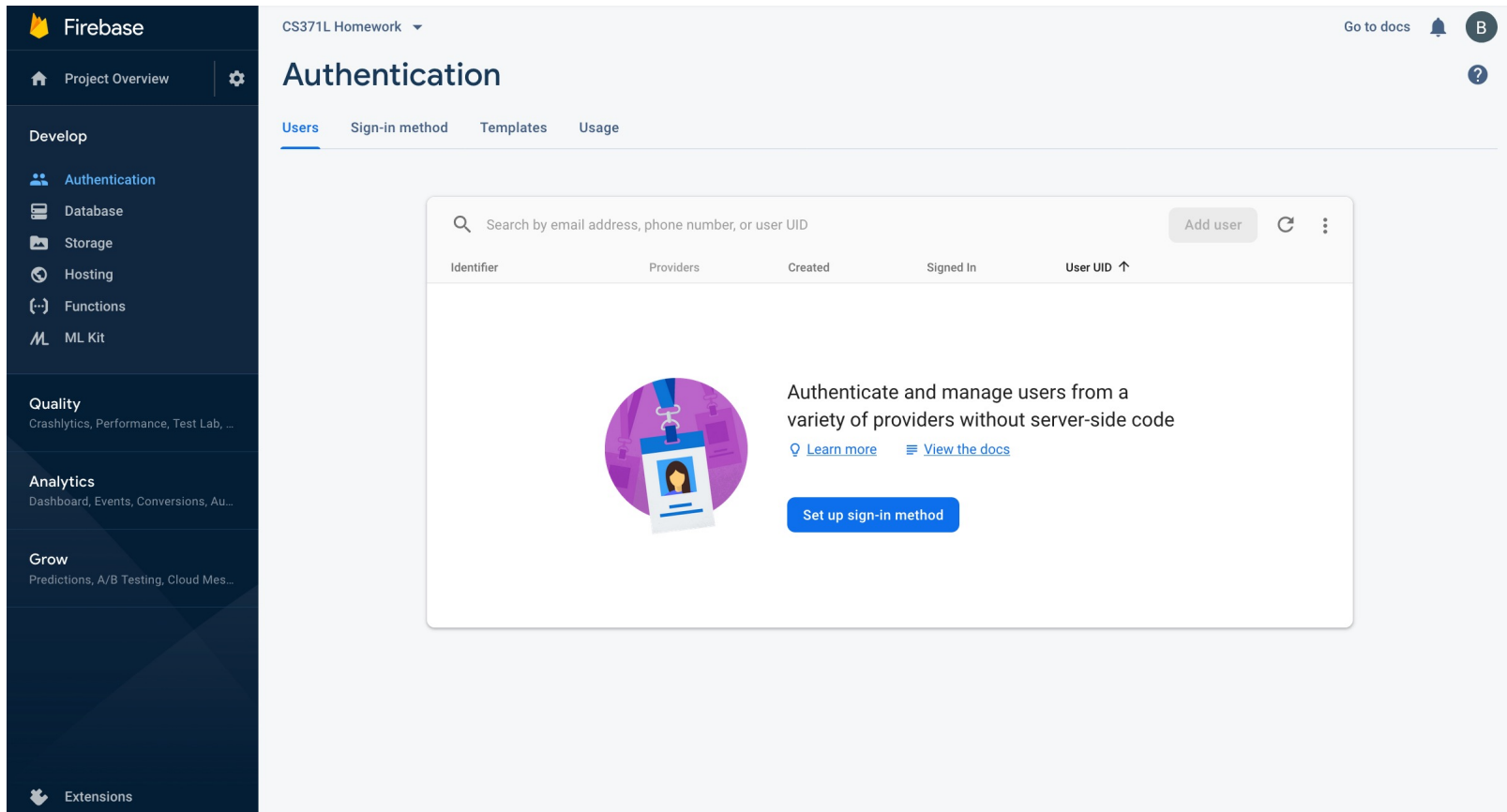
```
ref.observe(.value, with: { snapshot in
    print(snapshot.value as Any)
} )
```

`ref.observe` takes two parameters:

- An instance of `DataEventType`. `.value` means any event, including adds, deletes, and changes.
- A closure that dictates what you want to do when the event occurs. `snapshot` is just a snapshot of the data at that specific moment in time. The above just prints all of the data; you have to write code to parse the JSON.

# Authentication by email address

Find “Authentication” and either click on “Set up sign-in method” or click on the “Sign-in method” tab.



# Authentication by email address (cont.)

Select “Email/Password”.

The screenshot shows the Firebase Authentication console for a project named 'CS371L Homework'. The left sidebar contains navigation links for Project Overview, Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality, Analytics, Grow, and Extensions. The main content area is titled 'Authentication' and has tabs for Users, Sign-in method (selected), Templates, and Usage. Under the 'Sign-in method' tab, there is a table of 'Sign-in providers'.

Provider	Status
Email/Password	Disabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center (Beta)	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Yahoo	Disabled
Microsoft	Disabled
Anonymous	Disabled

Below the table, there is a section for 'Authorized domains' with an 'Add domain' button.

# Authentication by email address (cont.)

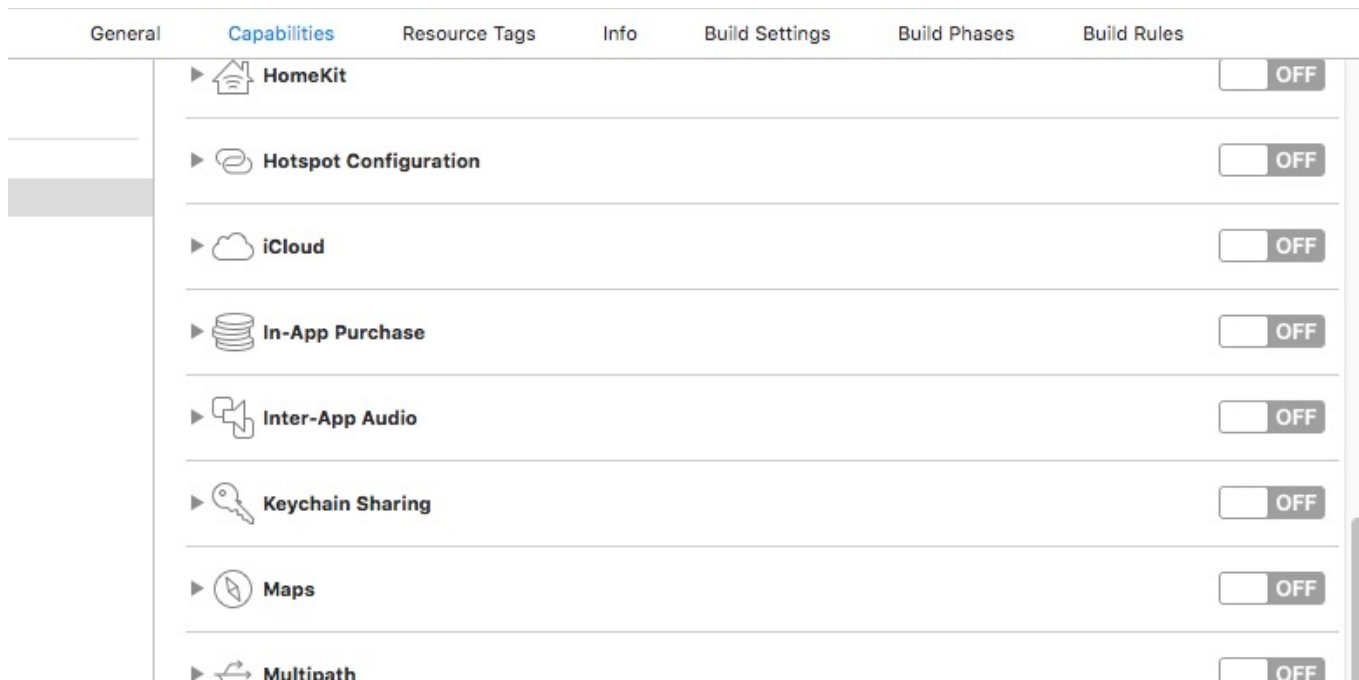
Select “Enable” and then “Save”.

The screenshot shows the Firebase Authentication console for a project named 'CS371L Homework'. The left sidebar contains navigation links for Project Overview, Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality (Crashlytics, Performance, Test Lab, ...), Analytics (Dashboard, Events, Conversions, Au...), Grow (Predictions, A/B Testing, Cloud Mes...), Extensions, and Spark (Free \$0/month, Upgrade). The main content area is titled 'Authentication' and has tabs for Users, Sign-in method (selected), Templates, and Usage. A modal dialog is open for the 'Email/Password' provider, showing an 'Enable' toggle switch turned on. Below the toggle, there is a description: 'Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. [Learn more](#)'. There is also an 'Email link (passwordless sign-in)' section with an 'Enable' toggle switch turned off. At the bottom of the modal are 'Cancel' and 'Save' buttons. In the background, a table lists other sign-in providers: Phone, Google, Play Games, Game Center (Beta), Facebook, Twitter, and GitHub, all of which are currently disabled.

Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center (Beta)	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled

## Authentication by email address (cont.)

Since Firebase stores credentials in the keychain, in Xcode, go to your project properties, and under the Capabilities tab, toggle “Keychain Sharing” on.



Now you can authenticate users for your app using their email and password.



---

## Authenticating by email address (cont.)

Use `Auth.auth().createUser` to register a new user into the authentication database. It takes three arguments:

- `withEmail`: a text string, ideally the `text` parameter from a `textField` object.
- `password`: a similar text string.
- a closure specifying what to do once the user is created.

---

## Authenticating by email address (cont.)

If there are no errors with the email address or password, use `Auth.auth().signIn` to authenticate an email address / password pair. It takes two arguments:

- `withEmail`: a text string, ideally the `text` parameter from a `TextField` object.
- `password`: a similar text string.

Note: Firebase expects both the `userid` and `password` and be at least 7 characters long!

---

## Authenticating by email address (cont.)

If we want to immediately log the user in once the account is created, we put the call to `signIn()` in the closure of `createUser()`:

```
// Assume uidField and pwdField are outlets of text fields

Auth.auth().createUser(
    withEmail: uidField.text!,
    password: pwdField.text!) { user, error in

    if error == nil {
        Auth.auth().signIn(
            withEmail: self.textFieldLoginEmail.text!,
            password: self.textFieldLoginPassword.text!)
    }
}
```