

**ASSIGNMENT-1(PART-2)**

**IOS Development**

**MOBILE APPLICATION DEVELOPMENT(MAD315)**

**SUBMITTED TO- MR. PROF- Manhaar Kapoor**

**SUBMITTED BY- LIKHITHA ANUGU(A00107449)**

**XCode introduction:**

Apple's development experience revolves around the XCode IDE. Xcode is a highly productive environment for developing programmes for the Mac, iPhone, iPad, Apple Watch, and Apple TV. It is tightly linked with the Cocoa and Cocoa Touch frameworks.

Apple's integrated development environment allows you to create, test, and submit your app.

XCode is a set of tools used by developers to create programmes for Apple's various platforms. From building your app through testing, optimising, and submitting it to the App Store, use XCode to manage your whole development cycle.

You can use the XCode > Open Developer Tool menu to start XCode's development tools, or you can use the XCode > Open Developer Tool menu to launch them independently:

When a real device isn't available, use Simulator for speedy development and testing your app in a virtual environment. Simulator supports alternative settings, files, and operating system versions for iPhone, iPad, Apple Watch, and Apple TV devices.

Instruments may be used to profile and analyse your programme, as well as to optimise speed and discover memory issues. Instruments takes data and displays the results using a variety of tools referred to as instruments.

Create and train bespoke machine learning models for your app with Create ML.

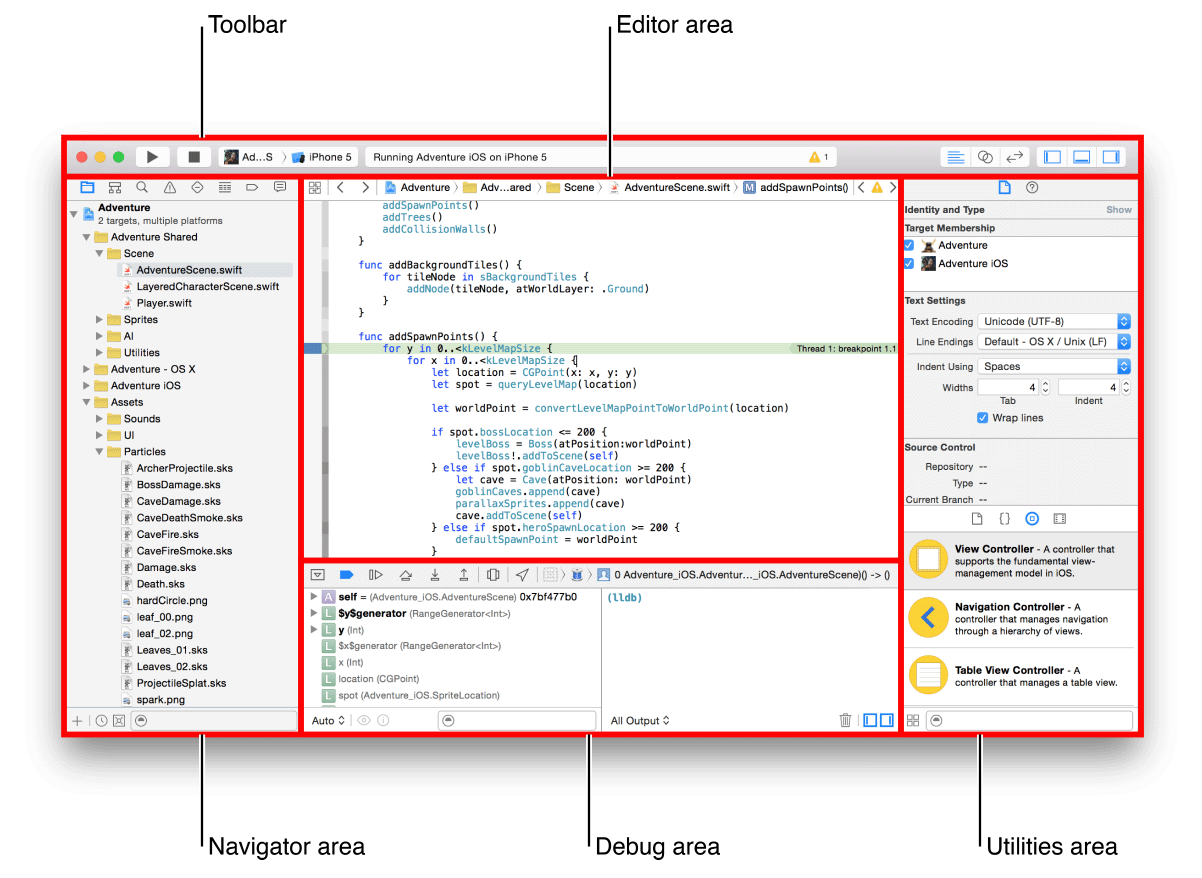
3D compositions and augmented reality (AR) experiences can be created with Reality Composer.

## **Workspace Window Overview**

The navigation area can be shown or hidden. Navigate all aspects of your project in this area, including files, symbols, breakpoints, build issues, tests, breakpoints, and build reports. You may also use the search function to find any string in your project.

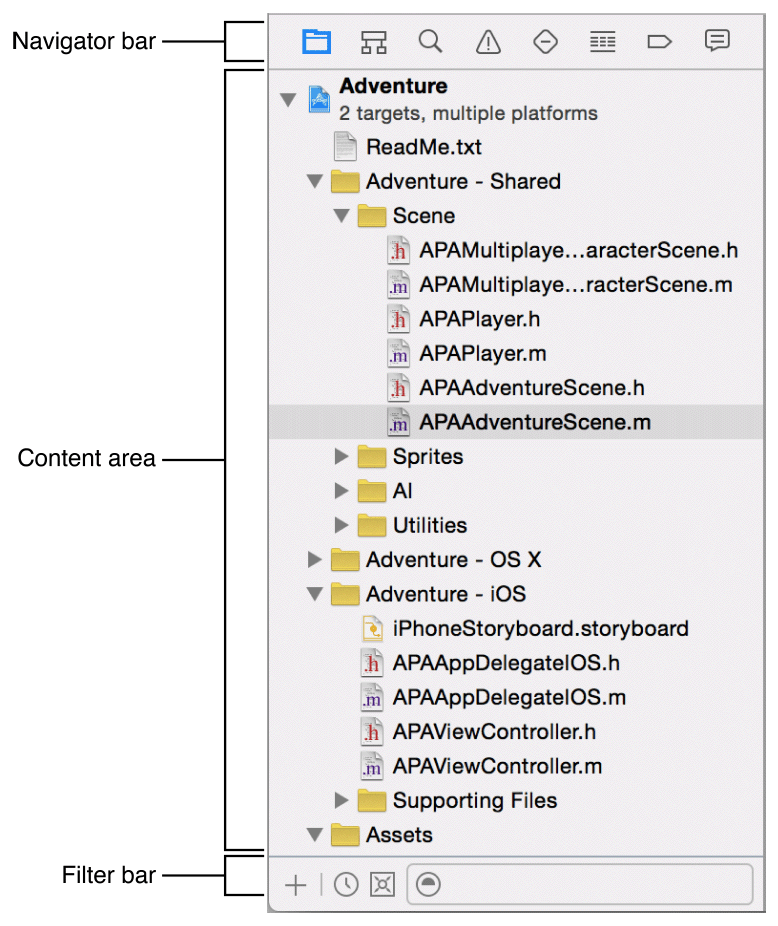
button debug: The debug area can be shown or hidden. This area is used for inspecting variables, communicating with the debugger console, and controlling the programme.

button utilities : The utility area can be shown or hidden. Inspect or change the characteristics of files, graphical user interface elements, sprites, and other project objects in this section. It can also be used to access a library of ready-to-use materials.



## **Navigating Your Workspace**

The navigator section allows you to access files, symbols, unit tests, diagnostics, and other aspects of your project. You select the navigator that is best suited to your task from the navigation bar. Each navigator's content section gives you access to essential parts of your project, and the filter bar on each navigator allows you to limit the content that is displayed.



Navigator for the project In the editor area, you can add, delete, group, and manage files in your project, or you can select a file to see or change its contents.

Navigator with symbols As a list or hierarchy, look at the symbols in your project. Only classes and protocols, only symbols in your project, or only containers can be selected using the buttons on the left side of the filter bar.

Navigator to be found To rapidly locate any string in your project, use search options and filters.

Navigator of the problem When you examine, analyse, and build your project, you'll notice issues like as diagnostics, warnings, and errors.

Navigator put to the test Unit tests should be written, managed, executed, and reviewed.

**Create an XCode project from a template to start developing your app.**

o Create an XCode project for your programme, choose a platform template, and choose the type of app you want to make, such as a single view, game, or document-based app for iOS. XCode templates contain all of the necessary project information and files to get you started developing your app right away.

**Prepare Configuration Information**

Gather the information that XCode requires to identify your app and you as a developer before starting a project:

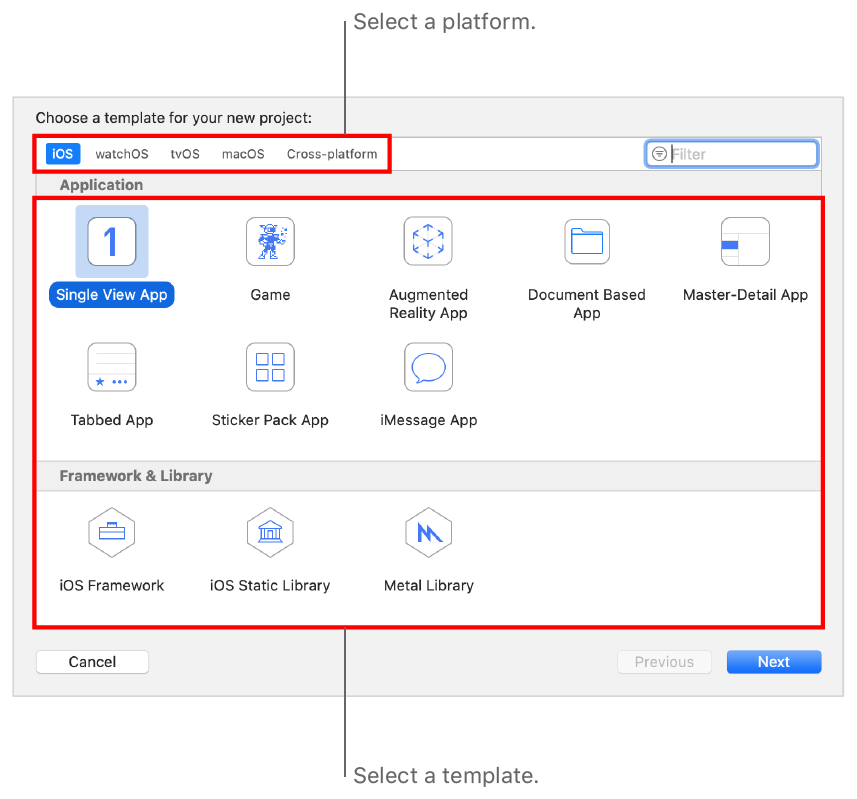
Name of the product. The name of your app as it will appear in the App Store and when installed on a device. The product name must be at least 2 characters long and no longer than 255 bytes in length, and it should be similar to the app name you'll use in App Store Connect later.

Identifier for the organisation. A reverse DNS string that identifies your company. Use com.example instead of a company identity if you don't have one. before distributing your software, replace it with your organization's name.

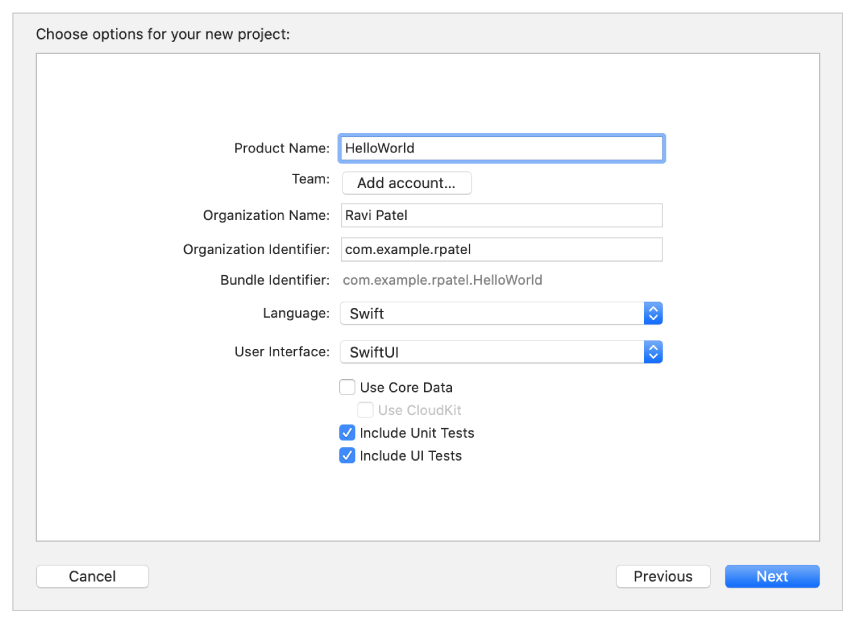
Name of the company. The name in boilerplate language throughout your project folder.

**Create a project:**

Launch Xcode, then select File > New > Project or click “Create a new Xcode project” in the Welcome to Xcode window. Choose the target operating system or platform, as well as a template, from the sheet that appears. Fill out the forms and select options for your project on the following sheets.



Because these are used to construct the bundle identity that identifies your app throughout the system, you must submit a product name and organisation identifier. Enter the name of your company as well. Enter your name if you're not a member of an organisation.



Choose SwiftUI as the user interface before clicking Next on this sheet to build for all platforms and to see an interactive preview of your layout.

# **Install or upgrade the xcode**

Here's how to set up Xcode and use your own Apple ID to develop apps.

We will, as a group,:

1. Go to the Mac App Store and download XCode.

2. Set your Apple ID in Xcode.

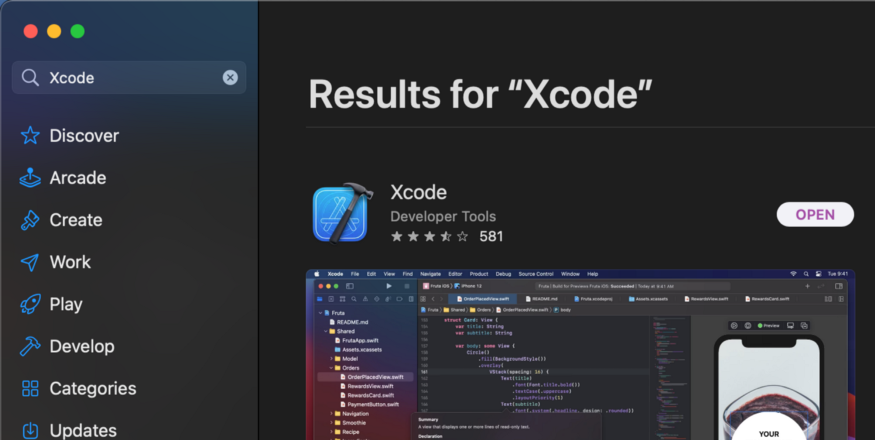
3. Make your email address the default for the "Git" version control system.

Following the “Build an App Like Lego, using SwiftUI” lesson series, you will be ready to start building your apps.

# **Install XCode**

You'll need Apple's XCode software first. It's completely free. It's a big download (12GB), but that's because it includes everything you'll need to make an app for iPhone, iPad, Mac, Apple Watch, and Apple TV.

Launch the App Store on your Mac and look for XCode. It is available for download and installation.



This could take an hour or two, depending on the speed of your Internet connection and your Mac. So, take a break or do something else for a time.

Open Xcode.

When you initially run Xcode, it will most likely ask for permission to instal additional components. Allow it to.

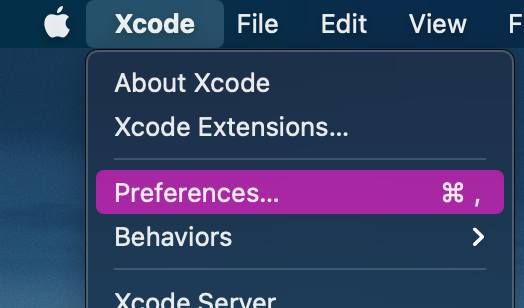
You'll see the welcome to Xcode window when Xcode is complete. For the time being, close the window. First, we'll configure the options in Xcode.



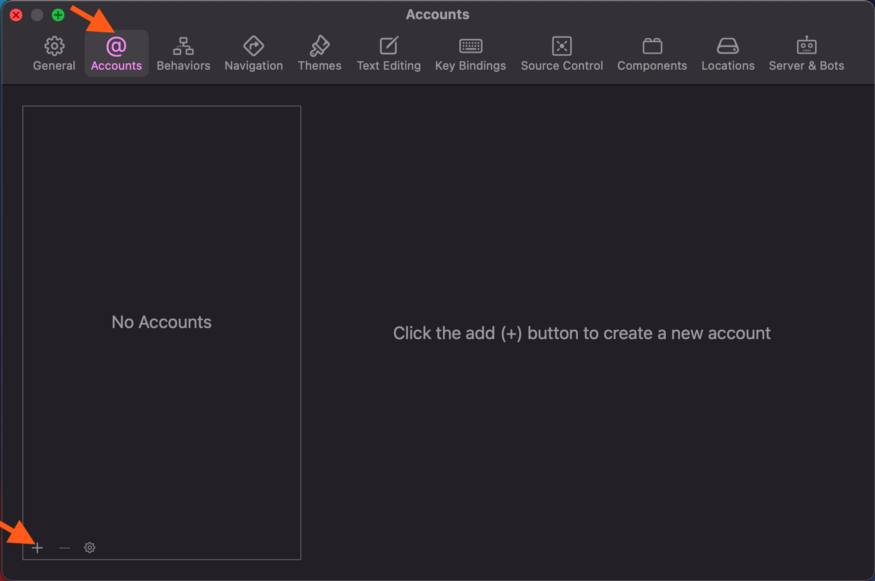
# **Connect to Your Apple ID**

We must inform Xcode as to who is making the changes. It will eventually complain if this is not done. There was no credit given to the author.

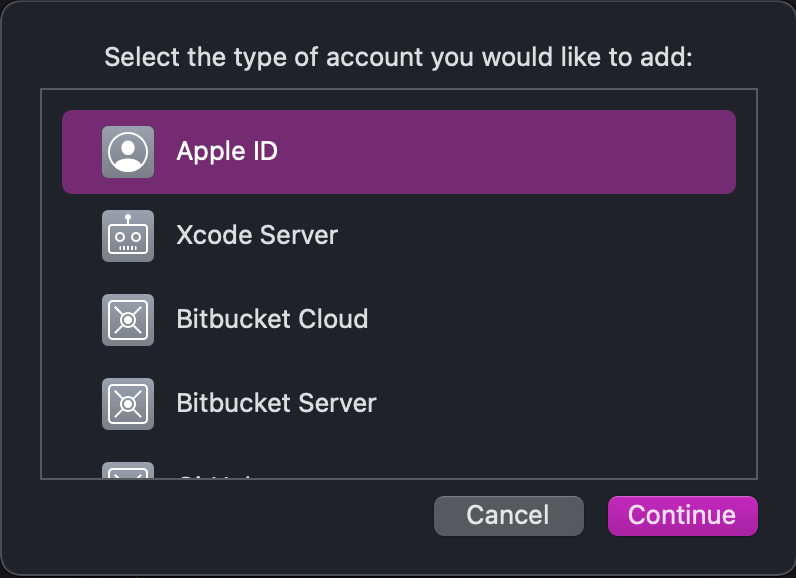
Preferences is found in the Xcode menu.



Go to the Accounts tab. If you don't see any accounts displayed, click the + button to create one.



As the account type, select Apple ID. Continue by pressing the button.



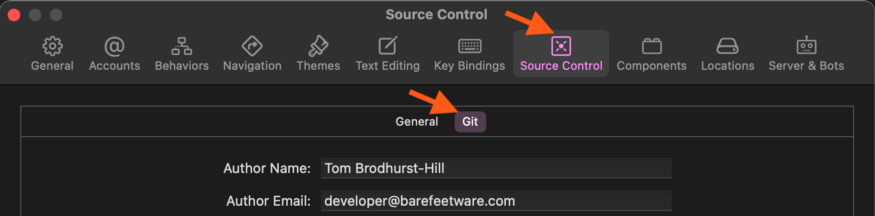
Put your Apple ID and password in the appropriate fields. To finish adding the account, follow the on-screen directions.



**Enter Your Author Details for Git**

In Xcode, you "commit" incremental changes to a "Git" repository. This is simple and clear with Xcode. All you have to do now is inform Xcode who will be making the adjustments.

Select the Source Control tab and the Git tab in the Preferences window of Xcode. Your full name and email address are required fields.



**Xcode and Capabilities:**

The Apple programming experience revolves around the Xcode IDE. Xcode is an extraordinarily productive environment for developing apps for the Mac, iPhone, iPad, Apple Watch, and Apple TV, thanks to its tight integration with the Cocoa and Cocoa Touch frameworks.

Xcode gets you from concept to code to customers in a smooth and efficient manner.

Workflows feel natural because everything is so neatly connected. The Assistant editor displays related source code in a split window pane while you create a new interface. To link UI controls to implementation code, simply drag the mouse.

Apple's LLVM compiler technologies parse your code, making sure that every symbol you see in the LLDB debugger matches what you see in the editor and compiler. That same engine is continually at work as you type, looking for errors and suggesting fixes for your code.

Xcode even interfaces with the Apple developer website, allowing you to quickly add features like Game Center and Passbook to your programme.

Capabilities:

A feature allows your programme to use Apple's app services like CloudKit, Game Center, and In-App Purchase. To use some app services, you must supply your app by using Xcode's project editor to add a capability that correctly configures the app service. Xcode modifies the Entitlements and Information Property List files, as well as adding associated frameworks and configuring your signing assets.

Some app features, such as Game Center and In-App Purchase, require further setup in App Store Connect and your developer account. For example, you'll need to use App Store Connect to publish a geographic coverage file for an app that uses Maps to deliver directions to other apps.

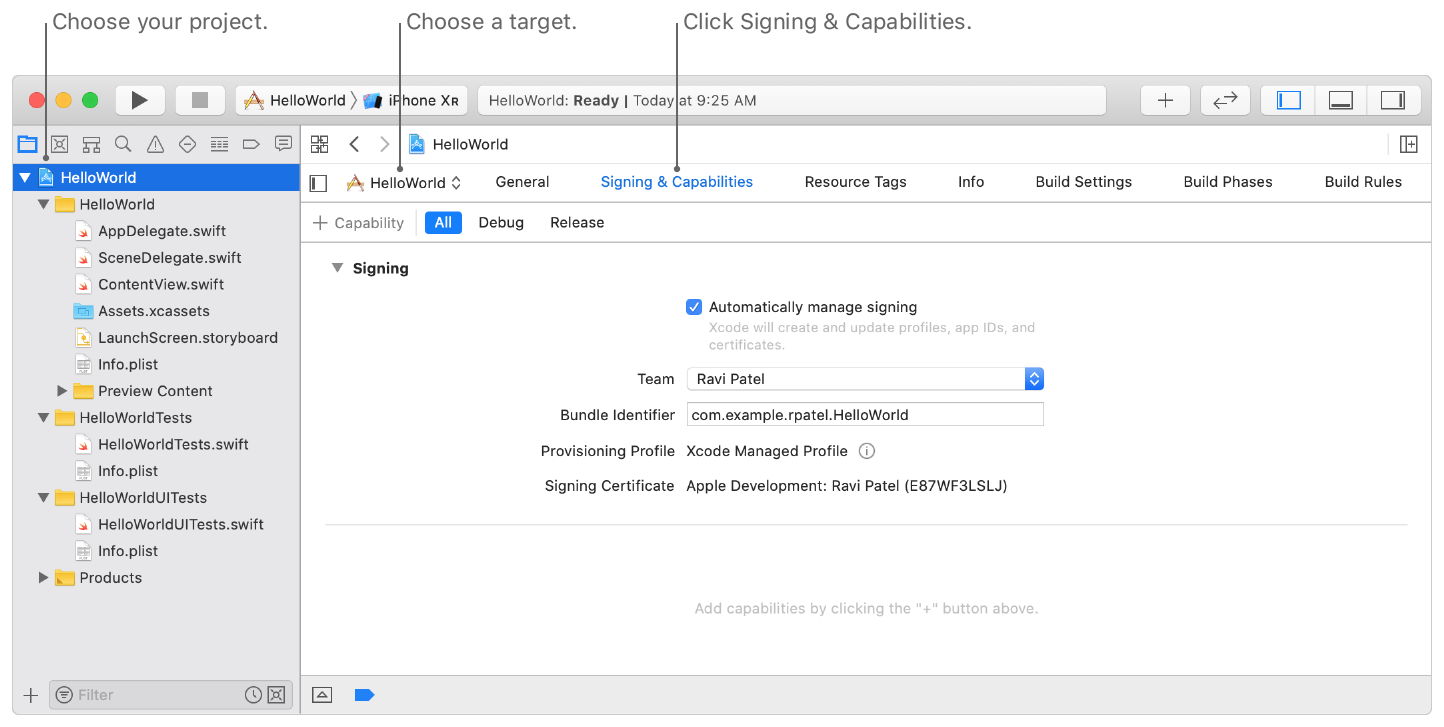
Your app's capabilities may be limited by the platform and whether you're a member of the Apple Developer Program. Go to the Reference area of Developer Account Help to find the supported capabilities—for example, go to Supported capabilities (iOS) to find the capabilities available to iOS apps.

Add your Apple ID account and designate the project to a team before you start so that Xcode can provision your app. Run your app on a device to register a device and establish a development provisioning profile for iOS, tvOS, and watchOS apps.

**Add a capabilities:**

The Signing & Capabilities pane of the project editor is where you add capabilities to your app.

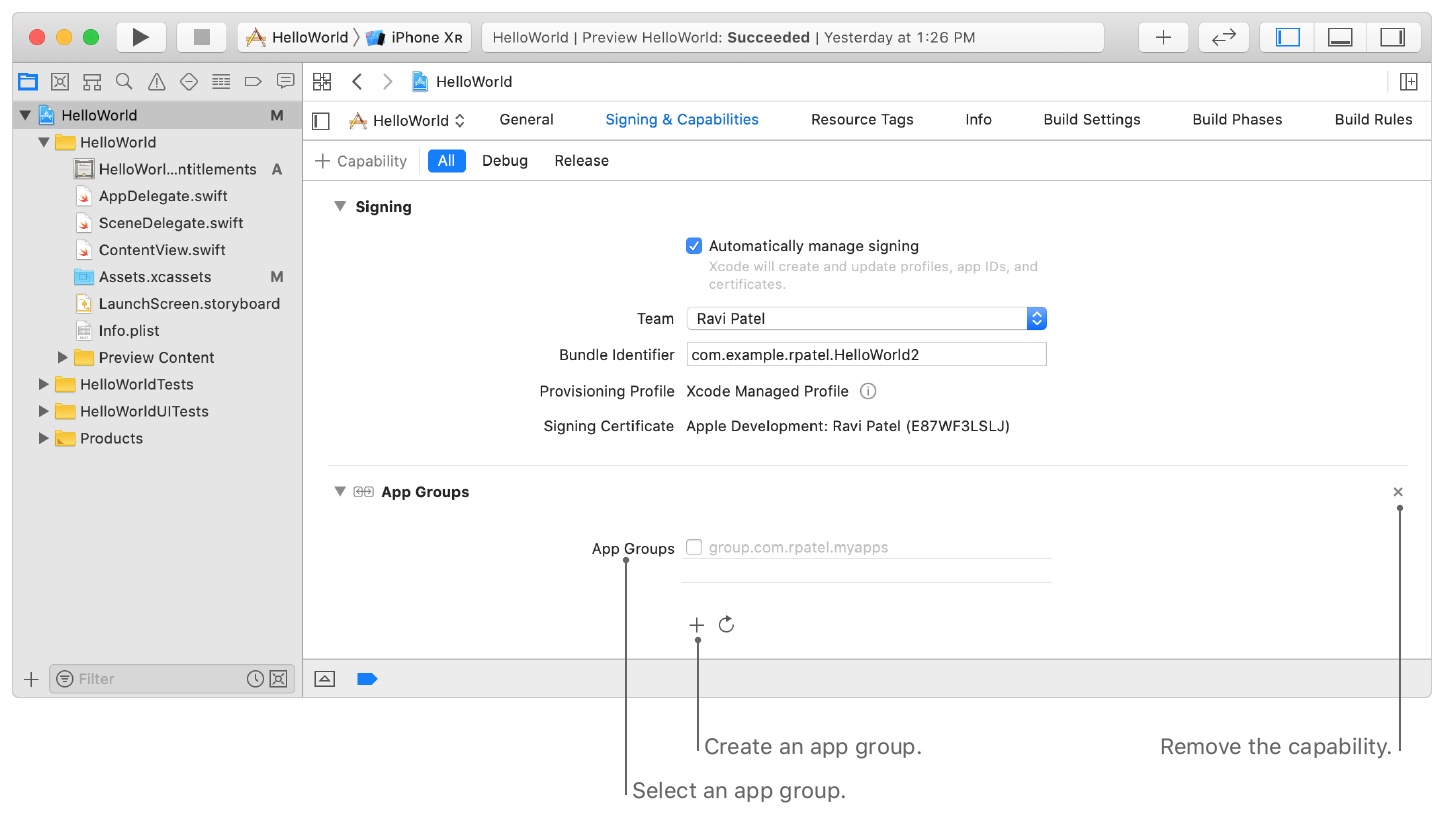
Select the project (the root group with the same name as your app) in the main window's Project navigator, and then the target in the project editor on the right. Either either the Project/Targets pop-up menu or the Targets section of the outline view, select the app's target.



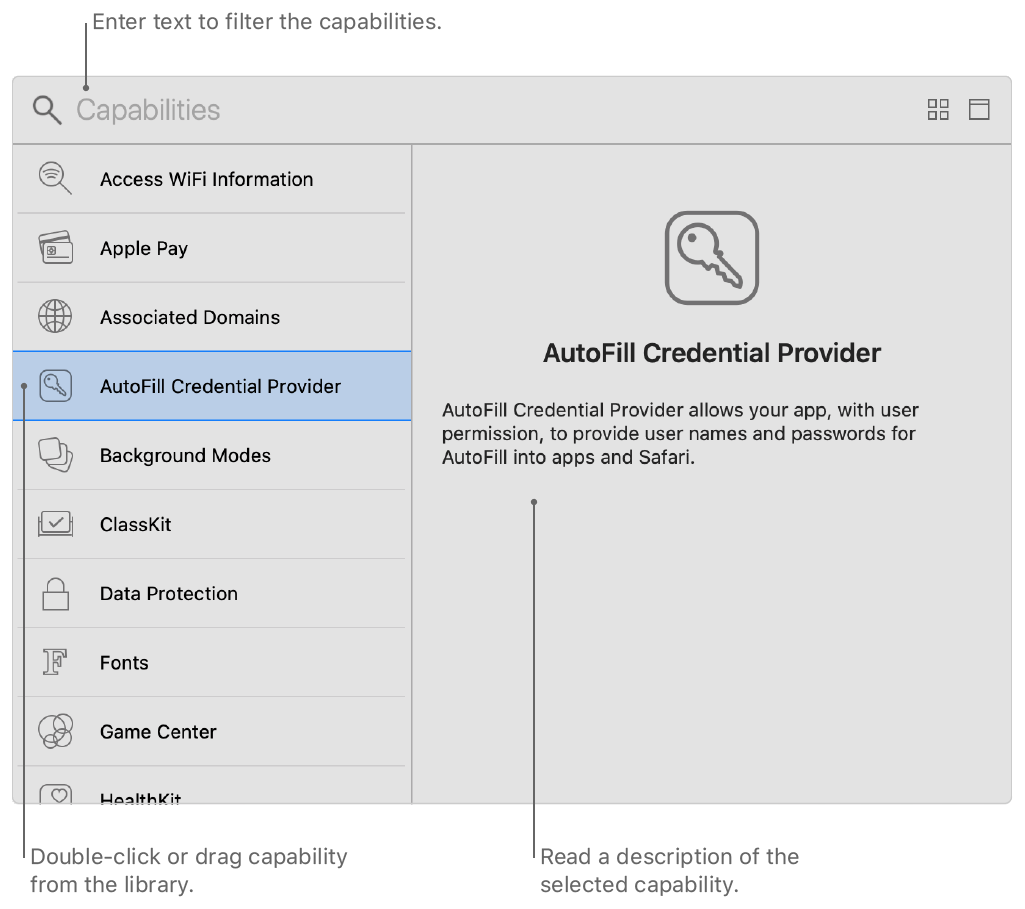
Choose a build configuration if you want to be more specific (All, Debug, or Release). Select Debug instead of All if you only want to add the capability to the Debug configuration.

To access the Capabilities library, click the Library (+) button in the toolbar. Only the capabilities available to the target platform and your membership in the programme are displayed in the Capabilities library.

Click the Show Details button in the library's upper-right corner to obtain a description of each capability. To read a description on the right, choose a capability on the left.



Double-click or drag a capability from the library into the Signing & Capabilities pane to add it to the app target. Below the Signing part, you'll see the capability. If extra setup steps are required, the capability expands to reveal additional controls. To remove a capability, click the X button in the upper-right corner of the capability in the Signing section.

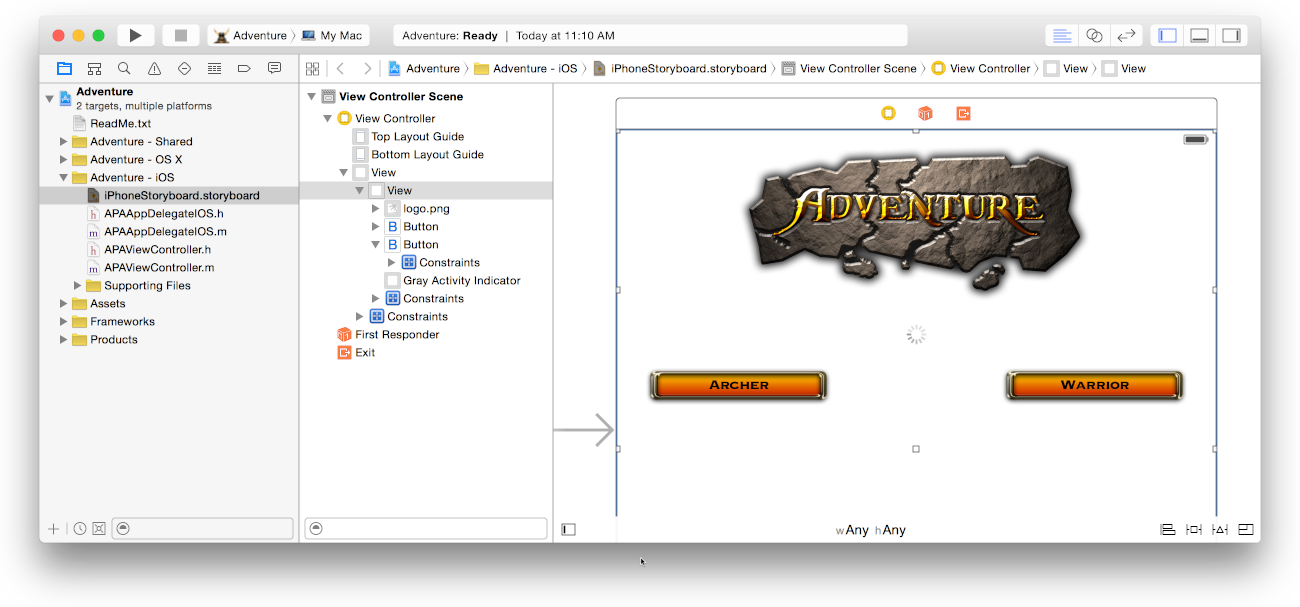


If you get an error notice in the Signing section, read it, fix the problem, and then try again. In the Bundle Identifier box under Signing, for example, the bundle ID (CFBundleIdentifier) must be unique. The organisation identification is concatenated with the app name you specified when creating the project as the default value for the bundle ID.

**Xcode interface:**

The Xcode Interface Builder editor makes it straightforward to develop a complete user interface without having to write any code. To make a working user interface, simply drag and drop windows, buttons, text fields, and other items into the design canvas.

Because Cocoa and Cocoa Touch are based on the Model-View-Controller architecture, it's simple to design your user interfaces irrespective of their implementations. User interfaces are essentially archived Cocoa or Cocoa Touch objects (.nib files), and when the app is run, macOS and iOS will construct the connection between UI and code dynamically.



### Parts of Interface Builder

### The dock (on the left) and the canvas (on the right) are the two main sections of Interface Builder (on the right). The user interface file's objects are listed in the dock. In your app's user interface, the canvas is where you lay out these items.

### image: ../art/DockAndCanvas.pdf

### All objects nested inside higher-level objects are visible in the dock's outline view.

### image: ../art/OutlineView.pdf

### Adding interface objects:

### Open the workspace window's utilities area by clicking button utilities to add an object to your app's user interface. By selecting the Object button picture, you can choose the Object library from the library pane

### Drag the object from the library to the outline view in the dock, the canvas, or the view controller's dock by clicking the icon representing the object. When the app is active, views that have been dragged into the dock can only be opened by segues or API requests. A view controller is dragged onto the canvas in this screenshot.

### 

### You can resize and rearrange objects as you add them to Interface Builder by dragging their handles. Dashed blue lines appear when you move items about the display, assisting you in aligning and positioning them.