iOS 17 App Development for Beginners

Get started with iOS app development using Swift 5.9, SwiftUI, and Xcode 15

Arpit Kulsreshtha

www.bpbonline.com

First published: 2024

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55515-858

www.bpbonline.com

Dedicated to

My Daughter

Anaya Kulsreshtha

About the Author

Arpit Kulsreshtha is an accomplished Developer, Team Lead, and Architect with an 11 year track record in Mobile Application Development. His expertise lies in Data Structures, Software Architecture, and Machine Learning, and he has successfully worked on a wide range of platforms, including iOS, Android, Symbian, macOS, tvOS, and watchOS. Arpit is well-versed in various programming languages and technologies, such as Swift, SwiftUI, Objective-C, J2ME, C#, Xamarin, React Native, and Flutter.

Throughout his career, he has demonstrated his versatility by developing mobile apps for diverse domains, including business, e-commerce, education, health and fitness, lifestyle, entertainment, and medical. His deep knowledge of software architecture, analysis, application integration, and development make him a highly skilled professional. Beyond his technical expertise, he is also a passionate blogger, an avid traveller, and enthusiastic about all things related to the latest technology.

Acknowledgement

I am immensely grateful to the many individuals who have significantly brought this book to life. Their continued and unwavering support throughout the writing process has been invaluable.

First and foremost, I would like to express my deepest gratitude to my mother, Mrs. Anita Kulsreshtha, for her continued support and encouragement, and to my wife, Akanksha Khare, for her constant inspiration, which has been instrumental in motivating my writing endeavors. I am also thankful to my dear family members - Ankit Kulsreshtha, Pallavi Raj, Mukesh Khare, Sandhya Khare, Aadya Kulsreshtha, Aayush Khare, and Sarthak Khare - for their incredible support and for being an integral part of my life.

I would like to express my sincere gratitude to everyone I have been honored to lead, follow, or work with. Their mentorship, leadership, and collaboration have contributed significantly to my professional journey and played a vital role in shaping the foundation of this book.

I am indebted to Dheeraj Takyar, Ravi Tailor, and Nayyar for their invaluable support, guidance, and assistance throughout the writing process. Their contributions have been truly remarkable.

Lastly, I would like to express my sincere gratitude to Mr. Nrip Jain and the entire team at BPB Publications for granting me the opportunity to write this book for them. Their belief in my abilities and support has been remarkable, and I am grateful for this wonderful opportunity.

Preface

Apple iOS app development is a platform to develop mobile applications with Swift for iPhone, iPad, and iPod devices. Apple will recently release its new series of iPhones with version 15 and its new iOS 17, which have updates to new APIs in Core Location, SwiftUI, Share Play, Machine Learning, ActivityKit, VisionKit, visionOS, and Symbol Framework. For years, Apple has used Xcode as their development tool and IDE, which includes the iOS SDK, tools, compilers, and frameworks that require you to design, develop, and write code for your iOS apps.

In this book, we explain how to set up and install an iOS app development environment and tools that will help you learn and practice the Swift language and how to use Swift and SwiftUI for iOS apps. We have explained iOS frameworks such as Core Location and MapKit for GPS with map-enabled applications, the AVFoundation Framework for Camera and media-enabled applications, Core Data for local database implementation, and the Core ML framework for Machine Learning and Artificial intelligence based applications on the iOS platform.

We have also covered mobile application architecture and patterns, which will help you write code with reusable and clean components that will increase the readability of the overall project. Once you can complete iOS app development, learning to publish and manage apps on the Apple Store will help you make your apps live on the Apple platform.

The primary goal of this book is to provide all necessary information and concepts with hands-on code to make you a complete iOS app developer, from which you can learn advanced concepts of the Apple platform to increase your knowledge base. We focus on the essentials and cover the material from a basic to an advanced level for you. Over the 22 chapters of this book, you will learn the following:

Chapter 1: Getting Started with Xcode- Introduces the Apple development IDE with step-by-step installation and setup of tools while learning about the playground that will enable you to practice Swift code with instant and quick results.

Chapter 2: Swift Fundamentals- We discussed the Swift Fundamentals, which include types, constants, and Variables in Swift, Operators, strings, and characters; Collection types; Control flow; Conditional statements; Control transfer; functions. And Closures. This chapter gives you the initial and firm foundation of the Swift language.

Chapter 3: Class, Structure, and Enumerations- We discussed further Object-oriented concepts in Swift, such as implementing Classes and structures. An in-depth explanation of the role of Properties in Swift classes with the types of methods. We also evaluated the Inheritance, initialization, and deinitialization of classes.

Chapter 4: Protocols, Extensions, and Error Handling- introduces protocol-oriented programming, which includes using protocols and Extensions to add new programming features to classes without changing their actual code. We explained the role and benefits of Error handling in Swift too.

Chapter 5: Automatic Reference Counting and Memory Safety- ARC is a memory management technique employed by iOS to automatically keep track of object references, ensuring that objects are deallocated from memory when they are no longer in use. This helps developers focus on writing code without worrying about manual memory management. On the other hand, memory safety is a design principle that aims to eliminate memory-related vulnerabilities, such as crashes and memory leaks, by enforcing strict rules and safeguards to prevent illegal memory access and the deallocation of objects.

Chapter 6: Implementing iOS 17 Architecture- iOS 17 Architecture refers to the underlying framework and structure that power Apple's mobile operating system, iOS. It is designed to provide a secure, efficient, and user-friendly environment for running applications on Apple devices such as iPhones, iPads, and iPods. The architecture of iOS is based on a layered approach, with different components working together to provide essential functionalities, including the kernel, frameworks, libraries, and user interface elements.

Chapter 7: User Interface Design with UIKit- Explained the UI interface design and development with StoryBoard and the UIKit framework that will enable you to learn different UI components such as tab bars, table views, collection views, etc., as well as make that design compatible and responsive for every iOS device that supports the latest development SDK.

Chapter 8: User Interface Design with SwiftUI- Discussed the SwiftUI framework, which is a declarative UI framework where you do not need to play with UI interface design and develop the UI with declarative code that takes less time compared to UIKit interface design development.

Chapter 9: Concurrency in Swift and SwiftUI- Swift introduces new features and enhancements to handle concurrent programming, allowing developers to write efficient and responsive applications. SwiftUI, Apple's declarative UI framework, leverages concurrency to create smooth and fluid user experiences. With the advent of Swift's async/await model and SwiftUI's integration with the new concurrency APIs, developers can easily manage tasks, handle parallelism, and streamline their code, resulting in highly performant and scalable applications.

Chapter 10: Storing Data with SQLite and Core Data- Discussed the local database operations with SQLite and the Core Data framework, which uses Object-relational mapping to map data to app business logic in an object-oriented manner and simplify the overall experience compared to direct SQLite handling. We also discussed the newly introduced SwiftData, which combines with SwiftUI.

Chapter 11: File Handling in iOS- Every iOS app has its container to handle files and cache in iOS. We discussed the App File Manager that makes you develop such a solution via which you will be able to manage file handling properly.

Chapter 12: Core Location with MapKit- Explains the current GPS location tracking with MapKit to track the user's geographic location and show other location references on the map to visualize in real-time.

Chapter 13: Camera and Photo Library- AVFoundation Framework, explains how to use the multimedia features of the camera and photo gallery to use photos and videos in iOS apps.

Chapter 14: Multithreading in iOS- Multithreading in iOS refers to the ability of an iOS application to perform multiple tasks concurrently, improving performance and responsiveness. It allows for the execution of multiple threads or processes simultaneously, enabling tasks to be executed in the background while the main thread remains responsive to user interactions. Multithreading in iOS can be achieved using various techniques, such as Grand Central Dispatch (GCD) and Operation Queues, which provide powerful abstractions for managing concurrent tasks and simplifying thread management.

Chapter 15: Networking in iOS Apps- Discusses the communication of iOS apps with server-side components to sync the data with Apple networking classes and the Alamofire framework.

Chapter 16: Mobile App Architectures, Patterns, and Anti-Patterns- Discusses app patterns and anti-patterns with code Architecture to make your development work more efficient and clean, enabling you to write high-end architectural code with reusability and better readability.

Chapter 17: Publish iOS App on the Apple App Store- Explains the step-by-step process to make your complete and bug-free iOS app live on the Apple app store, which requires learning App store management and understanding App Store Review guidelines to better handle rejections.

Chapter 18: Continuous Integration and Delivery with Xcode Cloud- Apple created the revolutionary cloud-based continuous integration and delivery (CI/CD) platform known as Xcode Cloud. It is designed specifically for iOS, iPadOS, macOS, and watchOS app development, offering a seamless and efficient workflow for teams and developers.

Chapter 19: Advance iOS with New Frameworks- This Chapter explains and offers advanced functionalities and features, allowing developers to leverage cutting-edge technologies in their iOS apps. From augmented reality (ARKit), vision kit, and activity kit to advanced user interfaces (UIKit) and multimedia capabilities (AVFoundation), iOS Advanced Frameworks empower developers to build immersive and innovative experiences that push the boundaries of mobile app development.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/1iqejxk

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/iOS-17-App-Development-for-Beginners**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Getting Started with Xcode

Introduction

After a big success in Mac computers, Apple shifted their focus to consumer electronic products such as iPod, iPhone, iPad, and so on. Apple introduced their new products with the success and advancements of iOS (Apple's OS for their consumer electronic products). With the iPhone achieving an even bigger success, in 2008, Apple introduced the App Store for third party apps, which began a new era of app developments. As the consumers started moving from the Mac machines to mobiles, the application development opened a new platform for the business to acquire more consumers that can use apps on the go while they are walking, traveling etc. Apple already had an IDE Xcode which was used by the developers for Mac application development. They have now made upgrades to it and accommodated the iOS app development in the same Xcode. In this chapter, we will start our application development with Xcode and build our first simple sample application which will look like a blank app in a simulator while testing.

Structure

In this chapter, we will cover the following topics:

- About Xcode IDE
- Downloading and installing Xcode
- Xcode user interface
- Configuring Xcode project

- Running and building Xcode iOS project
- Code with Xcode playground
- Xcode organizer

Objectives

After studying this chapter, you will be able to use Xcode IDE for app development, use a simulator to test the developed applications, and create an iOS Project in Xcode IDE.

Xcode IDE

Xcode is an **Integrated Development Environment** (**IDE**) that contains the software development tools developed by the Apple Company for developing applications and software for MacOS, iPhone, iPad, iPod Touch, iWatch, and Apple TV.

Xcode includes the Xcode IDE, Swift, SwiftUI, and C/ C++/ Objective-C compilers, an instrument analysis tool, simulators, the latest SDKs, and hundreds of other powerful features. Swift is a programming language that is fast, safe, and modern. SwiftUI is a declarative UI language used for the interactive app user interface.

Technical requirements

To fulfill the learning requirements and coding goals of this book, we will require the following:

- An Apple system such as *Mac mini, Macbook, iMac, Mac Studio*, and so on, with MacOS 13 Ventura OS version and later.
- An Apple ID (if you don't have one, this chapter will help you create an Apple ID).
- Apple iOS device (optional) (if you have one, you can run the apps on the device; otherwise, Xcode Simulator will be enough to check how your apps work).

We will start downloading the Xcode IDE for developing iOS applications from the Mac App Store.

Downloading and installing Xcode

Before starting the development of the iOS apps, we need to download and install Xcode; the following are the steps to perform the Xcode installation:

1. Open the Mac App Store.
2. Search for the keyword **Xcode** and press the *return* key.
3. In **Search Results**, Xcode version 15.x will appear, click on **Get and Install**.
4. Now, a box will appear that will ask for an Apple ID; If you already have an Apple ID, fill in the mentioned field, and it will ask for a password for your Apple account. Refer to *Figure 1.1* as follows:

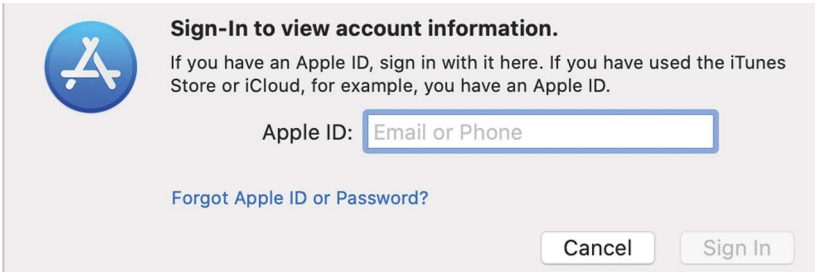

***Figure 1.1**: Account information*

If you don't have an Apple ID, you must create a new Apple ID with the help of the following Apple support article:

https://support.apple.com/en-in/HT204316

1. The latest version of Xcode is 15, which includes Swift 5.9, SDKs for iOS 17, iPadOS 17, tvOS 17, watchOS 10, visionOS 1.0 and macOS Sonoma.
2. Before completing the installation of Xcode, it will ask you to select the platform environment that you would like to install. The watchOS 9.0 and tvOS 16.0 SDKs are optional to install while installing Xcode, although if required, you can download these later on whenever needed, as shown in *Figure 1.2:*

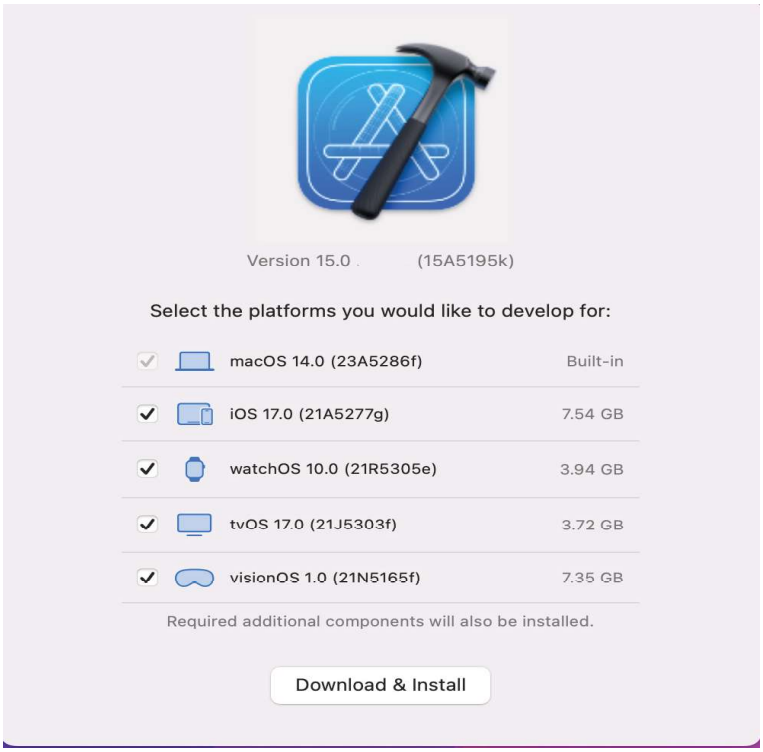

***Figure 1.2**: Xcode Download SDKs*

3. Open Xcode from the App Store. This will install some supported SDK software components for some time, shown in *Figure 1.3* as follows:

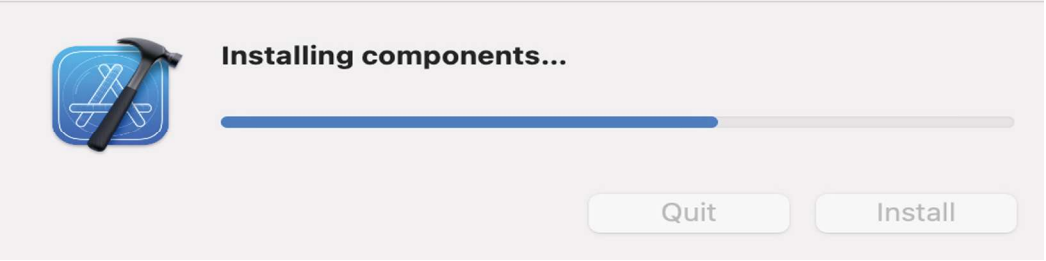

***Figure 1.3**: Xcode Installing Components*

4. After installing all required components, Xcode will open with its **Welcome to Xcode** page which will have three options to start working on an iOS 16 Project, as shown in *Figure 1.4* as follows:

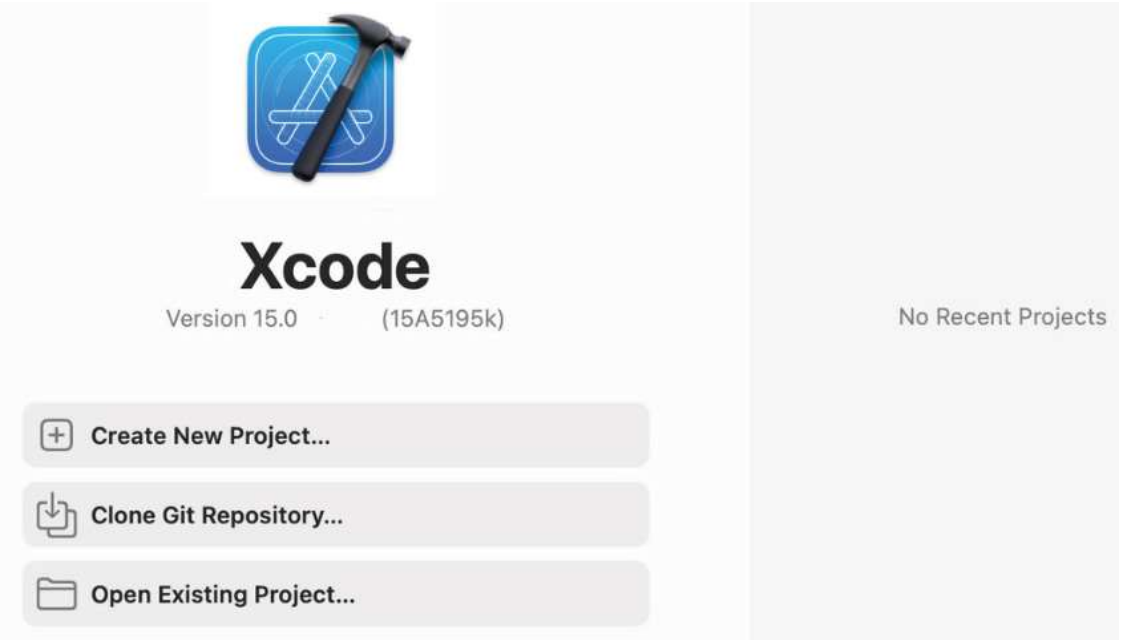

***Figure 1.4**: Xcode Welcome Window*

No recent projects will be seen if you launch it for the first time. Click on **Create a new Xcode project** to set up a new project for iOS app development.

5. If you downloaded an iOS project from any other source, then you can select the option of **Open a project or file** to check the code and run that project.
6. When you create a new project, you will see a set of new options to choose from, shown in *Figure 1.5* as follows:

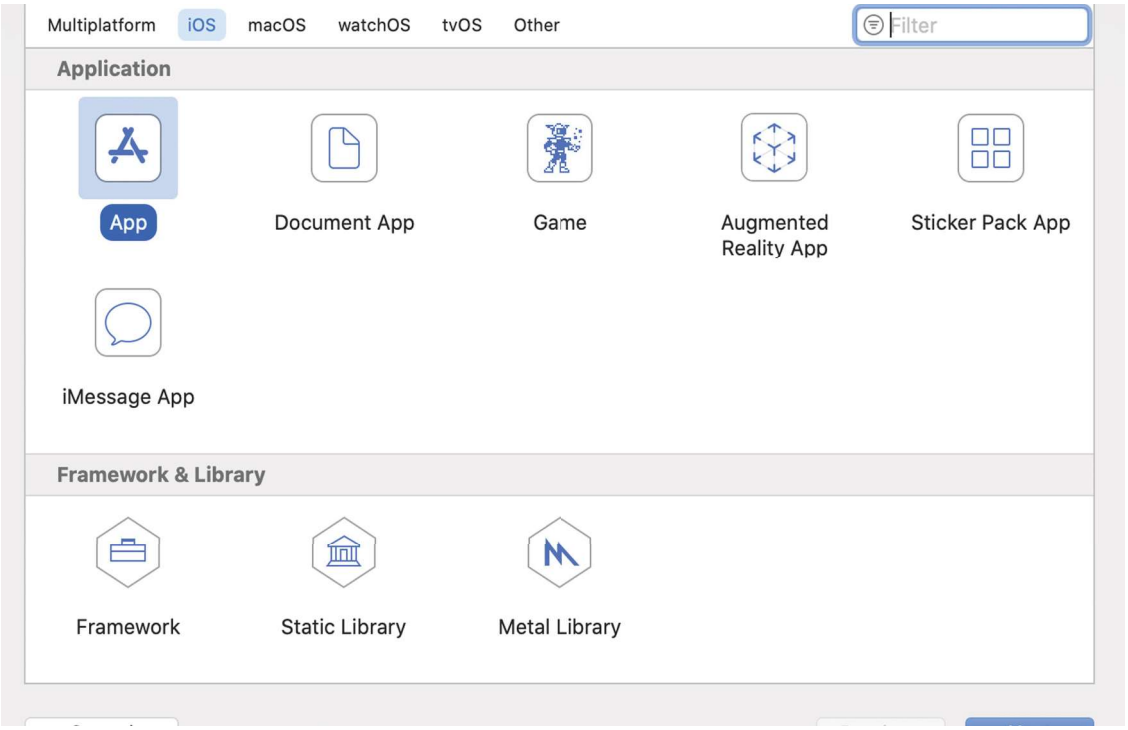

***Figure 1.5**: Select Application Platform*

To make an app for iPhone and iPad, select the option as iOS. Similarly, select watchOS, tvOS MacOS, and multi-platform for iWatch, Apple TV, Mac apps, and hybrid apps, respectively. Then select App and click on **Next**.

7. The next screen will have some fields to fill out regarding the setting up of the project, such as product name, team name, organization name, organization identifier, language, and user interface. Let us understand what possible values these fields could have and how it would impact the project, that is going to be created, as follows:
   - **Product Name**: This will be the name of your app, which will be entered in the text field of the project name.
   - **Team**: Apple Developer account for the project. We will discuss this in detail in *Chapter 20: Publish Apps on the App Store*.
   - **Organization Name**: The name of your company that owns the development of this project. You can just put your name down for now while you learn.
   - **Organization Identifier**: This identifier is created with the conjunction of company name. For now, enter **`com.organizationname`**.
   - **Bundle Identifier**: This field is not allowed to be edited. It's automatically created by combining an organization name with an organization identifier. This identifier is used to uniquely identify the apps on the Apple App Store. Refer to *Figure 1.6* as follows:

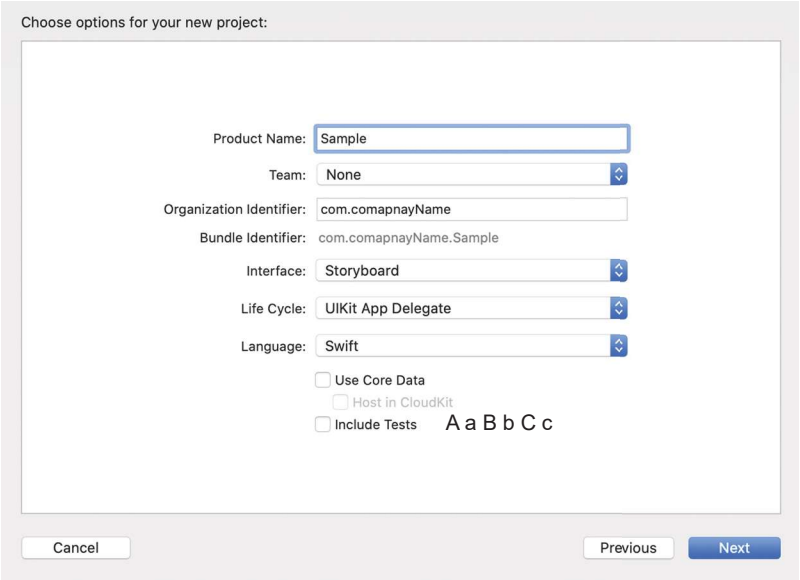

Figure 1.6: New iOS Project

- **Language**: This specifies the programming language to be used in app development. Select **Swift** from the drop-down field.
- **User Interface**: For now, set this to Storyboard. We will discuss another type of user interface selection, SwiftUI.
- **Life Cycle**: When you create a new SwiftUI Application in Xcode, you'll be given the option to choose between **SwiftUI App Lifecycle** and **UIKit App Delegate**. If the user interface is selected as a storyboard, it will be UIKit App Delegate by default.
- **Checkboxes**: These checkboxes are used to include the core data for the purposes of local database, unit tests, and UI unit test cases in the project. Keep them unchecked for now.

8. Choose the location where you want to save this project code on your Mac.

Xcode user interface

Before moving further towards app development, we first need to understand the navigation of Xcode, the tools, and the options used during the development of the iOS applications. Let's discuss the different sections of the Xcode Interface to understand clearly how we will use this IDE during app development, as follows:

- **ToolBar**: This toolbar on the top left side of Xcode is used to build, run, and view the execution progress of your app.
- **Windows pan bar**: This will help you configure the Xcode development environment.
  - **Object Library**: The addition or plus button will be used to add UI or object components to the project.

- **Version Editor**: This will be used to check for differences of codes between two versions.
- **Open and close buttons:** They will be used for Navigator, Debug, and Editor Area in Complete window pane.

- **Navigator area**: This provides quick access to all the parts of the project. The Navigation area is shown by default in the Xcode window.
- **Editor area**: The Editor area allows you to edit your code, user interface components, and resource files, if required in the project.
- **Inspector area**: This area will allow you to edit the information related to the items in the Navigator area and Editor area.
- **Debug area**: This area contains two portions – one for the view of variables and their values and another for the console.

Do not let all of the information about the parts and their relationships overwhelm you because we are trying to become familiar with Xcode's various components. We will study all these in detail and use each portion many times in the upcoming chapters.

Xcode search navigator

In the Project Navigator Area, when you click on the **Search** icon, it will open a search navigation bar where you can search for any text and see the results. The Xcode Editor Navigation area will show all the results of that project containing the text that was searched in the Xcode search navigation bar.

To use it via a shortcut, you can use the command *cmd + shift + F*, which will open the search navigation bar. These command operations would be helpful when you need to search multiple times and be quick. Refer to *Figure 1.7* as follows:

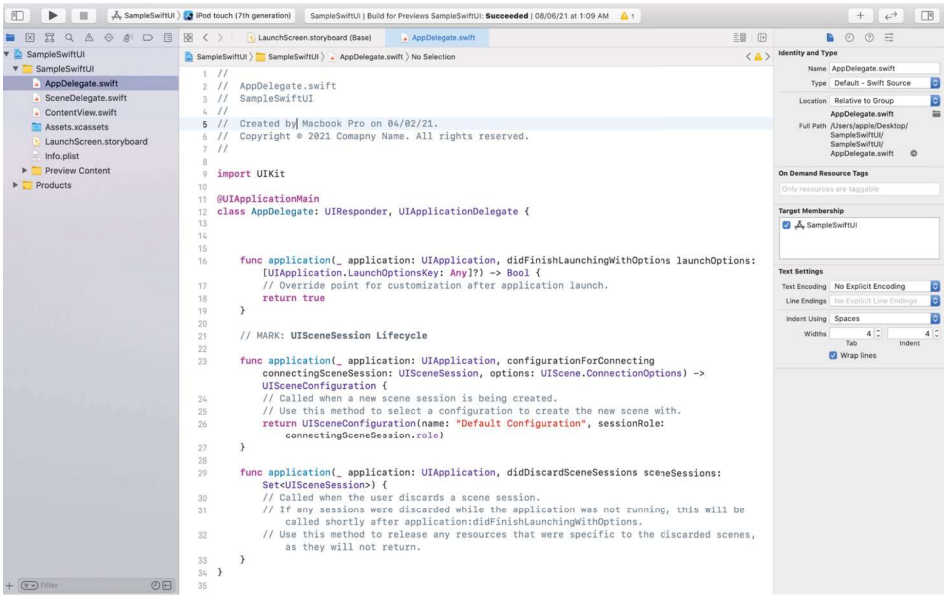

***Figure 1.7**: Application Development window*

Xcode issue navigator

When there are some issues in the project while compiling and executing the app, the project navigation area shows warnings in the yellow symbolic icon and errors in red. A project can be built and run with warnings but not with errors. All errors need to be resolved to run a successful app.

Warnings could be potential issues in the further development of the apps, so we should not ignore the warnings that are listed down in the issue navigation. Similarly, not only in coding but also in the development of the user interface, warnings and errors could occur in the same issue of navigation.

Configuring Xcode project

The actual project configuration file is the main, or root, node of the project navigator, and it has a blue icon. In this section, all the fields are already filled with optional content, and you can also edit those fields if required to change any such configurations of the project. Let's look at what we can change here, as follows:

- You can set or change the name of the project.
- Bundle identifier-related information can be edited.
- You can allow and disallow multiple orientations of the screen for your app.
- Set the minimum iOS version below which your app can't be installed.
- If required, add additional Apple libraries and frameworks.
- App icon and launch image linkage options can be edited.

Default project setup

When you create an Xcode project, it's created with some default components from which you can start the development of your app. We will discuss the default files and components created by Xcode automatically in the project, as follows:

- **`AppDelegate`**: **`AppDelegate`** requires very minimal or no change, usually during development. This component initializes the app container and controls the lifecycle of the app, which we will discuss and use in detail in later chapters.
- **`SceneDelegate`**: **`SceneDelegate`** is responsible for what needs to be displayed on the app screen in terms of user interface and data. The app transfers some of the app handling responsibilities to the **`SceneDelegate`**. This will handle some of the app states that were handled in **`AppDelegate`** earlier in iOS 14, shown in *Figure 1.8* as follows:

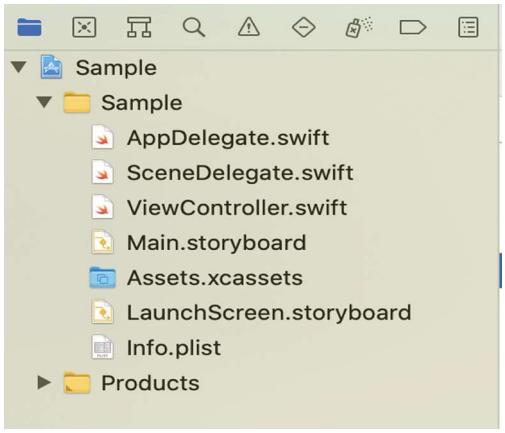

***Figure 1.8**: Application File Menu Window*

- **`ViewController`:** Xcode created this as a default view controller. We can create a new controller if required, or we can use the default one too. The **`ViewController`** class is mainly responsible for the lifecycle and functions of that view.
- **`Main.storyboard`:** The main storyboard is used for the development of the user interface of the app, which we will discuss in detail in the User Interface section of this book.
- **`Assets`:** This component handles the images of different dimensions, such as 1x, 2x, and 3x. As we know, the same app will be run on different resolution Apple devices like the iPhone 15, iPhone 15 Plus, iPad mini, iPad Pro, and so on; therefore, we must give the respective resolution images in 2x and 3x forms, so that in any condition, the images and icons of the app do not get blurred.
- **`1x`:** It is the size (in terms of height and width) of the image component on the interface; so to get the 2x and 3x size information, we need to multiply it by 2 for 2x resolution and 3 for 3x resolution. Every item of a single image in the **`Assets`** component has three placeholders for the respective resolutions.
- **`Info.plist`**: A common property list that contains information about your project is **`Info.plist`**. Plist extension is used by Apple for file formats like XML type. We will discuss plist in *Chapter 12*: *File Handling in iOS*. Sometimes you need to add more information to this property list. For example, if your app wants to access a photo library, the app plist needs to contain a key which explains the reason for accessing the photo library.

Creating and adding new file

In the Project Navigator Area, when you right-click, a few options will be shown to you. To create the new code file in the project, you need to select the **Create New File** option, then a new option Pan will ask to choose the file type – as you know, we selected the Project language Swift, so you need to select Swift File type. Next, this will ask you to name the file and after naming it, this swift extension file will be added to your project.

If you want to add any existing file to your project, right-click on the project name and select the option of adding a **New File**. This will open a window for the selection of files from your Mac system. After the selection of a file, don't forget to select the copy option, because without the copy option, it will only create the reference of the file in your project, and when you change the location of the project in your Mac or any other Mac machines, that file will show as missing from the project as it never became a part of your project while adding.

Build Storyboard UI

After we created the new project, earlier in this chapter, and then selected the option of storyboard for the user interface, Xcode included the **`main.storyboard`** file. The main storyboard contains the UI screen for **`ViewController`** on which the user interface components, such as labels, buttons, and so on, will be placed to inspect or to change their properties as per requirements.

Assistant editor

To open Assistant Editor, go to the **Editor** menu and choose **Assistant**. The Assistant Editor view shows two pane views in the Xcode window in such a way that its storyboard screen and its relative view controller file can be seen together to edit or work on UI and its code efficiently. This editor will help connect the user interface elements from the storyboard view to the view controller code file, so that those elements can be manipulated via code at the time of execution as per the logic of code.

In Xcode 17, the assistant editor shows the build log timeline to help developers of iOS applications enhance the overall parallelism and identify runtime build performance issues.

Utility area

- **File Inspector** is located in the first tab of the Utility area, which shows details about the current highlighted file in the project navigation area. This contains information such as type of file and the physical location of file on the Mac machine, so that you can locate the path of the file.
- **Quick Help Inspector** is located in the second tab of the Utility area, which shows you the documentation about the method, class, and keyword that your typing cursor is currently on in the code file.
- **Attribute Inspector and Size Inspector** will only show as tabs in the Utility area when you select the user interface file's elements in the Storyboard. Using these two inspectors, you can change the properties and size of the selected element. We will further discuss these topics in detail in *Chapter 8: User Interface Design with UIKit*.

Xcode 15 multiplatform support

Applications that run on multiple platforms increase the functionality of your app to each additional platform you want to support. You can use a single, multiplatform target to share the project settings and code for your app across platforms in Xcode 15 or later.

In the case your app is already supporting the multiplatform approach and shares a good amount of code then you need to consider combining all multiple targets into a single target with the help of new settings of Xcode 15. Let us understand with an example - Use one multiplatform target if your current app uses SwiftUI and you also want to use SwiftUI for the new platform. Use different targets, however, if your current app uses UIKit and you want to use AppKit on the Mac app development. You will be able to support destination platform in app configurations as shown in below *Figure 1.9:*

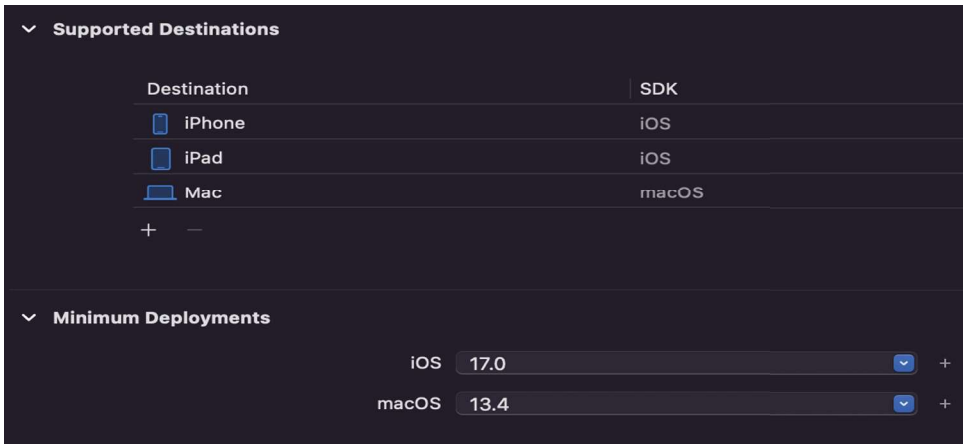

***Figure 1.9**: Multiplatform support*

Another platform can help you find issues with your application's build-time. Add conditional statements around any code that makes use of platform-specific APIs or frameworks to address those problems. By including platform-specific views and features, you can make sure that the user interface and experience of your app meet the requirements of each platform. As shown in figure 1.8, a multiplatform app single target can share the code for iOS, iPadOS, tvOS, macOS but watchOS app will remain in separate target.

Adding Mac destination in multiplatform

In case if you want to add Mac OS application destination target, then you need to choose the destination type as per requirements and experience you want to give to end users. We have three options and each one of them have their own unique design characteristics and specifications.

Mac

In case you are going to start a new mac app then you will select this option while adding Mac in Supported Destinations. You may utilize all of the AppKit and SwiftUI-powered features and APIs from the macOS SDK. Multiplatform applications by default use this option.

Mac catalyst

In case you already have an iPad app, and you are bringing that to mac then select this option while adding Mac in Supported Destinations. The system modifies the way common UIKit interface