MCA253 - Mobile Applications

Unit: 5.0

Introduction to iOS

Dr. Siddesha S MCA, M.Sc Tech (by Research) , Ph.D,
Asst. Professor,
Dept. of Computer Applications,
JSS Science and Technology University
Mysuru – 570 006

1976

1977

1998

2001

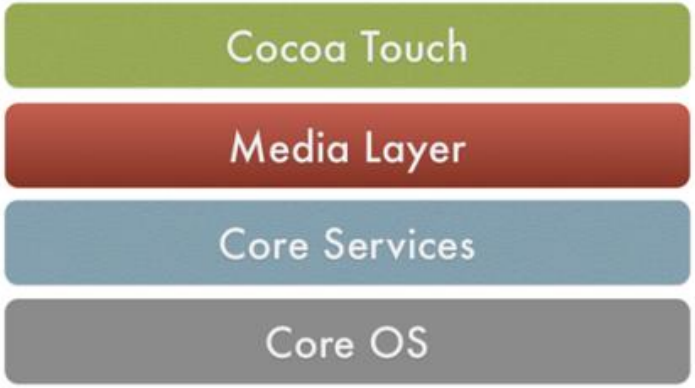
2007

Present

Introduction

- iOS is a mobile operating system developed and distributed by Apple Inc.
- It was originally released in 2007 for the iPhone, iPod Touch, and Apple TV.
- iOS is derived from OS X, with which it shares the Darwin foundation.
- iOS is Apple's mobile version of the OS X operating system used in Apple computers.
- The iOS devices get evolved quite frequently and from experience, we find that at least one version of iPhone and iPad is launched every year.
- Now, we have iPhone 12 Pro max, iPhone 11 pro launched which has its predecessors starting from iPhone, iPhone 3gs, iPhone 4, iPhone 4s……iPhone 11
- Similarly, iPad has evolved from iPad (1st Generation) to iPad (4th Generation) and an additional iPad Mini version.
- Features are Maps, Siri, Facebook and Twitter, Multi-Touch, Accelerometer, GPS, High end processor, Camera, Safari, Powerful APIs, Game centre, In-App Purchase Reminders, Wide Range of gesture

iOS Architecture



- **Core OS Layer:**
- The Core OS layer holds the low level features that most other technologies are built upon.
  - Core Bluetooth Framework.
  - Accelerate Framework.
  - External Accessory Framework.
  - Security Services framework.
  - Local Authentication framework.
- 64-Bit support from IOS7 supports the 64 bit app development and enables the application to run faster

iOS Architecture- Core Service Layer

- **Core Services Layer:** Some of the Important Frameworks available in the core services layers are detailed:
  - **Address book framework** – Gives programmatic access to a contacts database of user.
  - **Cloud Kit framework** – Gives a medium for moving data between your app and iCloud.
  - **Core data Framework –** Technology for managing the data model of a Model View Controller app.
  - **Core Foundation framework –** Interfaces that gives fundamental data management and service features for iOS apps.
  - **Core Location framework –** Gives location and heading information to apps.
  - **Core Motion Framework –** Access all motion based data available on a device. Using this core motion framework Accelerometer based information can be accessed.

iOS Architecture - Core Service Layer

- **Foundation Framework –** Objective C covering too many of the features found in the Core Foundation framework
- **Healthkit framework –** New framework for handling health-related information of user
- **Homekit framework –** New framework for talking with and controlling connected devices in a user's home.
- **Social framework –** Simple interface for accessing the user's social media accounts.
- **StoreKit framework –** Gives support for the buying of content and services from inside your iOS apps, a feature known as In-App Purchase.

iOS Architecture – Media Layer

- **Media Layer:** Graphics, Audio and Video technology is enabled using the Media Layer.

Graphics Framework:

- **UIKit Graphics –** It describes high level support for designing images and also used for animating the content of your views.
- **Core Graphics framework –** It is the native drawing engine for iOS apps and gives support for custom 2D vector and image based rendering.
- **Core Animation –** It is an initial technology that optimizes the animation experience of your apps.
- **Core Images –** gives advanced support for controling video and motionless images in a nondestructive way
- **OpenGl ES and GLKit –** manages advanced 2D and 3D rendering by hardware accelerated interfaces
- **Metal –** It permits very high performance for your sophisticated graphics rendering and computation works. It offers very low overhead access to the A7 GPU.

iOS Architecture – Media Layer

- **Audio Framework:**
  - **Media Player Framework –** It is a high level framework which gives simple use to a user's iTunes library and support for playing playlists.
  - **AV Foundation –** It is an Objective C interface for handling the recording and playback of audio and video.
  - **OpenAL –** is an industry standard technology for providing audio.
- **Video Framework**
  - **AV Kit –** framework gives a collection of easy to use interfaces for presenting video.
  - **AV Foundation –** gives advanced video playback and recording capability.
  - **Core Media –** framework describes the low level interfaces and data types for operating media.

iOS Architecture - Cocoa Touch Layer

- **EventKit framework –** gives view controllers for showing the standard system interfaces for seeing and altering calendar related events
- **GameKit Framework –** implements support for Game Center which allows users share their game related information online
- **iAd Framework –** allows you deliver banner-based advertisements from your app.
- **MapKit Framework –** gives a scrollable map that you can include into your user interface of app.
- **PushKitFramework –** provides registration support for VoIP apps.
- **Twitter Framework –** supports a UI for generating tweets and support for creating URLs to access the Twitter service.
- **UIKit Framework –** gives vital infrastructure for applying graphical, event-driven apps in iOS.

Registering as an Apple Developer

- An Apple ID is most necessary if you are having any Apple device and being a developer, you definitely need it. It's free and hence, no issues in having one. The benefits of having an Apple account are as follows −

  - Access to development tools.
  - Worldwide Developers Conference (WWDC) videos.
  - Can join iOS developer program teams when invited.

- To register an Apple account, follow the steps given below −

Step 1 − Click the link https://developer.apple.com/programs/register/ and select "Create Apple ID“

Step 2 − Provide the necessary information, which is self explanatory as given in the page.

Step 3 − Verify your account with your email verification and the account becomes active.

Step 4 − Now you will be able to download the developer tools like Xcode, which is packaged with iOS simulator and iOS SDK, and other developer resources.

Registering as an Apple Developer

Apple iOS Developer Program

- The first question that would arise to a new developer is – Why should I register for an iOS developer program?
- The answer is quite simple; Apple always focuses on providing quality applications to its user.
- If there was no registration fee, there could be a possibility of junk apps being uploaded that could cause problems for the app review team of Apple.
- The benefits of joining the iOS developer program are as follows −

- Run the apps you develop on the real iOS device.
- Distribute the apps to the app store.
- Get access to the developer previews.

Apple iOS Developer Program

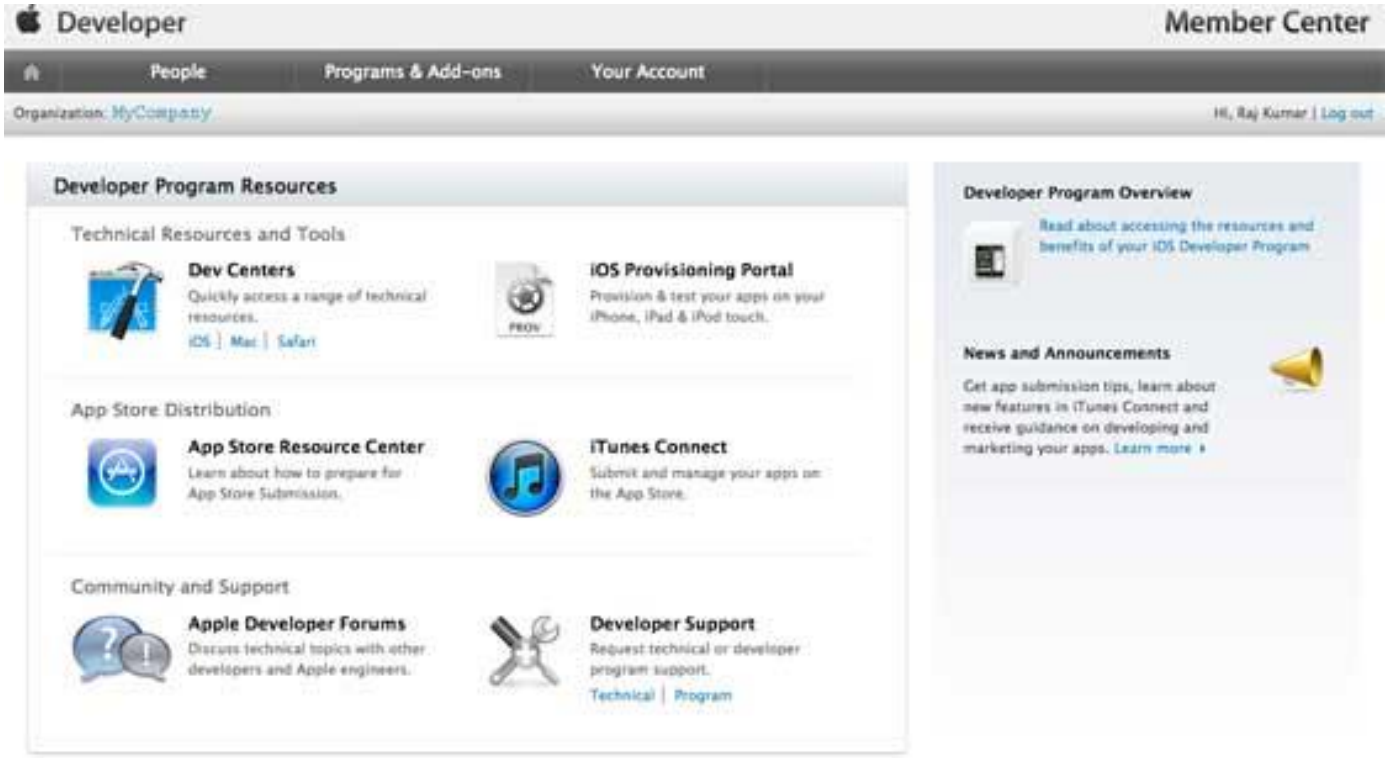
- The steps to join the iOS developer program are as follows −

Step 1 − To register, click on the link − (https://developer.apple.com/programs/ios/).

Apple iOS Developer Program

- **Step 2** − Click on Enroll Now in the page that is displayed.
- **Step 3** − You can either sign in to your existing apple account (if you have one) or create a new Apple ID.
- **Step 4** − Thereafter, you have to select between Individual and Company accounts. Use company account if there will be more than one developer in your team. In individual account, you can't add members.
- **Step 5** − After entering the personal information (for those who newly registers), you can purchase and activate the program by paying with the help of your credit card (only accepted mode of payment).
- **Step 6** − Now you will get access to developer resources by selecting the member centre option in the page.

Apple iOS Developer Program



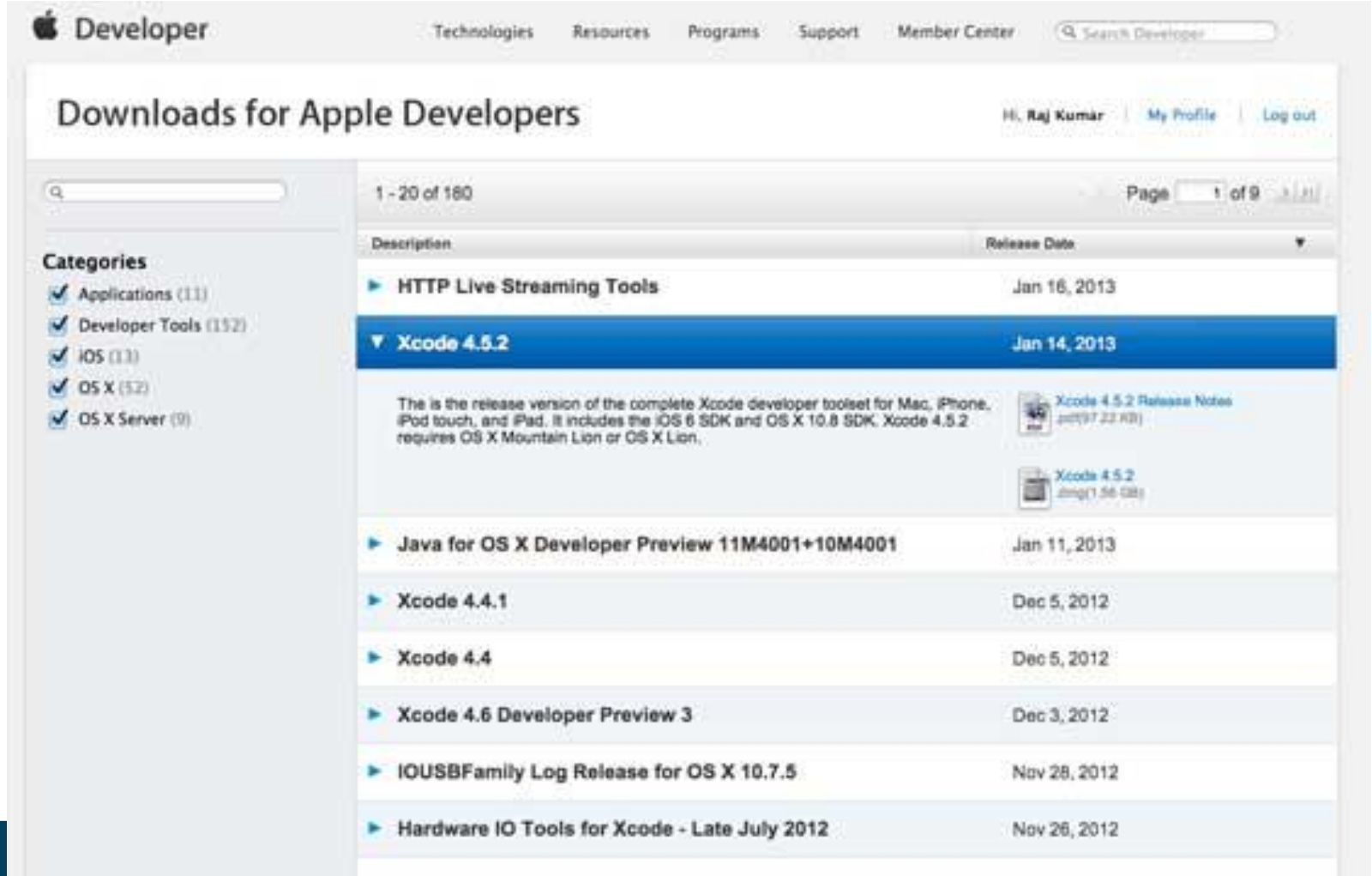
- **Step 7** − Here you will be able to do the following −
  - Create provisioning profiles.
  - Manage your team and devices.
  - Managing application to app store through iTunes Connect.
  - Get forum and technical support.

Xcode

- Xcode is an application that developers use to build apps for Apple's various platforms such as iPhone, iPad, Macs, AppleTV and Apple Watch.
- Xcode is available for Macs only but there are alternative options for PC users. (Using Mac OS on VM)
- It requires macOS 10.14.4 or later and 7.6 GB of hard drive space.
- The latest version of Xcode is 11.

❑ iOS - Xcode Installation

- **Step 1** − Download the latest version of Xcode from https://developer.apple.com/downloads/



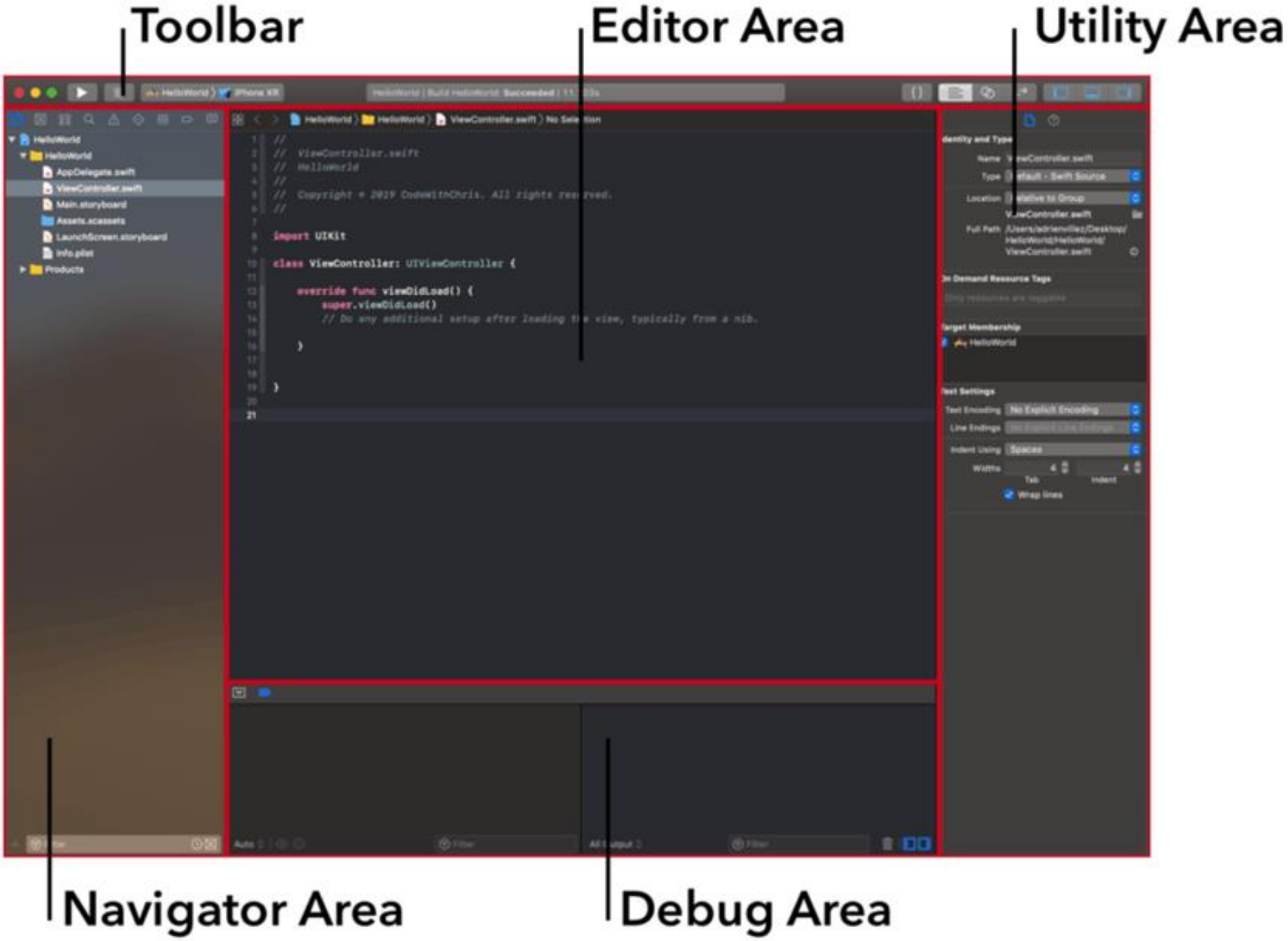
iOS - Environment Setup

- **Step 2** − Double click the Xcode dmg file.
- **Step 3** − You will find a device mounted and opened.
- **Step 4** − There will be two items in the window that's displayed namely, Xcode application and the Application folder's shortcut.
- **Step 5** − Drag the Xcode to application and it will be copied to your applications.
- **Step 6** − Now Xcode will be available as a part of other applications from which you can select and run.

❑ You also have another option of downloading Xcode from the Mac App store and then install following the step-by-step procedure given on the screen.

Interface Builder

- Interface builder is the tool that enables easy creation of UI interface.
- You have a rich set of UI elements that is developed for use.
- You just have to drag and drop into your UI view.
- You have objects library at the right bottom that consists the entire necessary UI element.
- The user interface is often referred as **xibs**, which is its file extension.
- Each of the xibs is linked to a corresponding view controller.


Toolbar
Editor Area
Utility Area
Navigator Area
Debug Area

iOS Simulator

- An iOS simulator actually consists of two types of devices, namely iPhone and iPad with their different versions.
- iPhone versions include iPhone (normal), iPhone Retina, iPhone 5. iPad has iPad and iPad Retina.

iOS Evolution

iOS version	Start Year	End Year	Features
ioS-1	2007	2010	Core iOS UI, Multi-touch gesture, Mobile Safari, Visual voice mail, Maps, Web clips on home screen, Multi-touch keyboard, iTunes Wi-Fi Music store, iTune Sync.
ioS-2	2008	2011	Native 3rd Party Apps, App Store, MS Exchange support, Battery Life and Speed fixes, iTunes Genius Playlist, Dropped Call fixes, Google street view, Podcast downloads, MobileMe.
ioS-3	2009	2012	Cut-Copy-Paste, Voice Control, MMS, Spotlight Search, Push notification, USB & Blue tooth Tethering, Genius features, Ringtone downloads, Remote Lock, Landscape keyboard, Find My iPhone, Voice control over blue tooth.
ioS-4	2010	2013	Multitasking, Home Screen folders, FaceTime video chat, Unified Email Inbox, iTunes home sharing, Threaded Email messages, Game Center, TV Rentals, iTune Ping, HDR Photos, Personal HotSpot(GSM), Air Play for 3rd party Apps, Retina Display support
ioS-5	2011	2014	Siri, Notification Center, PC-free, iTunes Wi-Fi Sync, iMessages, iCloud.

iOS Evolution

iOS version	Start Year	End Year	Features
ioS-6	2012	2015	iCloud Tabs, Siri Enhancements, Facebook integration, Home grown maps and Turn-by-Turn Navigation, Mail enhancements, FaceTime Over Cellular, Passbook
ioS-7	2013	2016	Visual overhaul, Control Center, AirDrop, iTunes Radio, FaceTime Audio, Refreshed Core Apps.
ioS-8	2014	NA	Continuity, Widgets, Extensibility, Homekit, QuickType, iCloud Drive, HealthKit , Family sharing.
ioS-9	2015	NA	Night Shift, Lower power mode, Public Beta Program.
ioS-10	2016	NA	iMessage Apps, Delete built-in Apps
ioS-11	2017	NA	Augmented Reality, Major enhancements on iPad, AirPlay -2.
ioS-12	2018	NA	Grouped notifications, ARKit 2, Siri Improvements, Screen Time, Memoji
ioS-13	2019	NA	System wide dark mode, Overhauled Stock Apps like Reminders and Notes, New Improved Siri voice, New Portrait lighting options, Sign In with Apple User account system.

iOS Evolution

iOS version	Start Year	End Year	Features
iOS-14	2020	NA	System wide dark mode, Overhauled Stock Apps like Reminders and Notes, New Improved Siri voice, New Portrait lighting options, Sign In with Apple User account system enhancement in audio and video.
iOS-15	21-Sep-2021	NA	Redesigned notifications, Siri, "Focus" for reducing distractions, Spatial Audio and SharePlay in FaceTime calls, Text recognition in images, ID cards in Wallet app, Added privacy features, Safari, Maps, Weather, and Notes app redesigns.

Introduction to Swift

- Swift is a new programming language for iOS, macOS, watchOS, and tvOS app development.
- Nonetheless, many parts of Swift will be familiar from your experience of developing in C and Objective-C.
- Swift provides its own versions of all fundamental C and Objective-C types, including Int for integers, Double and Float for floating-point values, Bool for Boolean values, and String for textual data. Swift also provides powerful versions of the three primary collection types, Array, Set, and Dictionary.
- Like C, Swift uses variables to store and refer to values by an identifying name. Swift also makes extensive use of variables whose values can't be changed. These are known as constants, and are much more powerful than constants in C.

Introduction to Swift

- Constants are used throughout Swift to make code safer and clearer in intent when you work with values that don't need to change
- In addition to familiar types, Swift introduces advanced types not found in Objective-C, such as tuples.
- Tuples enable you to create and pass around groupings of values.
- You can use a tuple to return multiple values from a function as a single compound value.
- Swift is a *type-safe* language, which means the language helps you to be clear about the types of values your code can work with.
- If part of your code requires a String, type safety prevents you from passing it an Int by mistake. Likewise, type safety prevents you from accidentally passing an optional String to a piece of code that requires a non-optional String. Type safety helps you catch and fix errors as early as possible in the development process.

Constants and Variables

- Constants and variables associate a name (such as maximumNumberOfLoginAttempts or welcomeMessage) with a value of a particular type (such as the number 10 or the string "Hello").
- The value of a *constant* can't be changed once it's set, whereas a *variable* can be set to a different value in the future.

Declaring Constants and Variables

Constants and variables must be declared before they're used. You declare constants with the let keyword and variables with the var keyword. Here's an example of how constants and variables can be used to track the number of login attempts a user has made:

```
let maximumNumberOfLoginAttempts = 10
var currentLoginAttempt = 0
```

Declaring Constants and Variables

- This code can be read as:
- “Declare a new constant called maximumNumberOfLoginAttempts, and give it a value of 10. Then, declare a new variable called currentLoginAttempt, and give it an initial value of 0.”
- You can declare multiple constants or multiple variables on a single line, separated by commas: var x = 0.0, y = 0.0, z = 0.0

Type Annotations

- You can provide a *type annotation* when you declare a constant or variable, to be clear about the kind of values the constant or variable can store.
- Write a type annotation by placing a colon after the constant or variable name, followed by a space, followed by the name of the type to use.

Type Annotations

- var welcomeMessage: String
- The colon in the declaration means “…of type…,” so the code above can be read as: “Declare a variable called welcomeMessage that is of type String.”
- The welcomeMessage variable can now be set to any string value without error:

```
welcomeMessage = "Hello"
```

- You can define multiple related variables of the same type on a single line, separated by commas, with a single type annotation after the final variable name:

```
var red, green, blue: Double
```

Naming Constants and Variables

- Constant and variable names can contain almost any character, including Unicode characters:

1. **let π = 3.14159**
2. **let 你好 = "你好世界"**
3. **let □ □ = "dogcow"**

- Constant and variable names can't contain whitespace characters, mathematical symbols, arrows, private-use Unicode scalar values, or line- and box-drawing characters. Nor can they begin with a number, although numbers may be included elsewhere within the name.
- You can change the value of an existing variable to another value of a compatible type. In this example, the value of friendlyWelcome is changed from "Hello!" to "Bonjour!":

```
var friendlyWelcome = "Hello!"
friendlyWelcome = "Bonjour!"
```

Printing Constants and Variables

- You can print the current value of a constant or variable with the print(_:separator:terminator:) function:

```
print(friendlyWelcome)
// Prints "Bonjour!"
print("The current value of friendlyWelcome is
        \(friendlyWelcome)")
// Prints "The current value of friendlyWelcome is Bonjour!"
```

Comments

- Use comments to include non executable text in your code, as a note or reminder to yourself.
- Comments are ignored by the Swift compiler when your code is compiled.
- Comments in Swift are very similar to comments in C. Single-line comments begin with two forward-slashes (//):

```
// This is a comment.
```

- Multiline comments start with a forward-slash followed by an asterisk (/*) and end with an asterisk followed by a forward-slash (*/):

```
/* This is also a comment
    but is written over multiple lines. */
```

Comments

- Multiline comments in Swift can be nested inside other multiline comments. You write nested comments by starting a multiline comment block and then starting a second multiline comment within the first block,

```
/* This is the start of the first multiline comment.
/* This is the second, nested multiline comment. */
This is the end of the first multiline comment. */
```

Semicolons

- Unlike many other languages, Swift doesn't require you to write a semicolon (;) after each statement in your code, although you can do so if you wish. However, semicolons *are* required if you want to write multiple separate statements on a single line:

```
let cat = "□ "; print(cat)
// Prints "□ "
```

Integers

- *Integers* are whole numbers with no fractional component, such as 42 and -23. Integers are either *signed* (positive, zero, or negative) or *unsigned* (positive or zero).
- Swift provides signed and unsigned integers in 8, 16, 32, and 64 bit forms. These integers follow a naming convention similar to C, in that an 8-bit unsigned integer is of type UInt8, and a 32-bit signed integer is of type Int32. Like all types in Swift, these integer types have capitalized names.

Floating-Point Numbers

- *Floating-point numbers* are numbers with a fractional component, such as 3.14159, 0.1, and -273.15.
- Floating-point types can represent a much wider range of values than integer types, and can store numbers that are much larger or smaller than can be stored in an Int. Swift provides two signed floating-point number types:
- Double represents a 64-bit floating-point number.
- Float represents a 32-bit floating-point number.

Type Safety and Type Inference

- Swift is a *type-safe* language. A type safe language encourages you to be clear about the types of values your code can work with. If part of your code requires a String, you can’t pass it an Int by mistake.
- Because Swift is type safe, it performs *type checks* when compiling your code and flags any mismatched types as errors.
- This enables you to catch and fix errors as early as possible in the development process.
- Type-checking helps you avoid errors when you’re working with different types of values.
- If you don’t specify the type of value you need, Swift uses *type inference* to work out the appropriate type.
- Type inference enables a compiler to deduce the type of a particular expression automatically when it compiles your code, simply by examining the values you provide.

```
let meaningOfLife = 42
        // meaningOfLife is inferred to be of type Int
let pi = 3.14159
        // pi is inferred to be of type Double
```

- Swift always chooses Double (rather than Float) when inferring the type of floating-point numbers.
- If you combine integer and floating-point literals in an expression, a type of Double will be inferred from the context:

```
let anotherPi = 3 + 0.14159
// anotherPi is also inferred to be of type Double
```

Numeric Literals

- Integer literals can be written as:
  - A *decimal* number, with no prefix
  - A *binary* number, with a 0b prefix
  - An *octal* number, with a 0o prefix
  - A *hexadecimal* number, with a 0x prefix
- All of these integer literals have a decimal value of 17:
  - let decimalInteger = 17
  - let binaryInteger = 0b10001 // 17 in binary notation
  - let octalInteger = 0o21 // 17 in octal notation
  - let hexadecimalInteger = 0x11 // 17 in hexadecimal notation

Numeric Type Conversion

- Use the Int type for all general-purpose integer constants and variables in your code, even if they're known to be nonnegative.
- Using the default integer type in everyday situations means that integer constants and variables are immediately interoperable in your code and will match the inferred type for integer literal values.
- Use other integer types only when they're specifically needed for the task at hand, because of explicitly sized data from an external source, or for performance, memory usage, or other necessary optimization.
- Using explicitly sized types in these situations helps to catch any accidental value overflows and implicitly documents the nature of the data being used.

Integer Conversion

- The range of numbers that can be stored in an integer constant or variable is different for each numeric type.
- An Int8 constant or variable can store numbers between -128 and 127, whereas a UInt8 constant or variable can store numbers between 0 and 255.
- A number that won't fit into a constant or variable of a sized integer type is reported as an error when your code is compiled:

```
let cannotBeNegative: UInt8 = -1
// UInt8 cannot store negative numbers, and so this will
   report an error
let tooBig: Int8 = Int8.max + 1
// Int8 cannot store a number larger than its maximum
  value, and so this will also report an error
```

Integer and Floating-Point Conversion

- Conversions between integer and floating-point numeric types must be made explicit:

```
let three = 3
let pointOneFourOneFiveNine = 0.14159
let pi = Double(three) + pointOneFourOneFiveNine
// pi equals 3.14159, and is inferred to be of type
 Double
```

Type Aliases

- *Type aliases* define an alternative name for an existing type. You define type aliases with the typealias keyword.
- Type aliases are useful when you want to refer to an existing type by a name that is contextually more appropriate, such as when working with data of a specific size from an external source:

```
typealias AudioSample = UInt16
```

- Once you define a type alias, you can use the alias anywhere you might use the original name:

```
var maxAmplitudeFound = AudioSample.min
// maxAmplitudeFound is now 0
```

Booleans

- Swift has a basic *Boolean* type, called Bool. Boolean values are referred to as *logical*, because they can only ever be true or false. Swift provides two Boolean constant values, true and false:

```
let orangesAreOrange = true

let turnipsAreDelicious = false
```

Tuples

- *Tuples* group multiple values into a single compound value. The values within a tuple can be of any type and don't have to be of the same type as each other.
- In this example, (404, "Not Found") is a tuple that describes an *HTTP status code*. An HTTP status code is a special value returned by a web server whenever you request a web page.
- A status code of 404 Not Found is returned if you request a webpage that doesn't exist.

```
let http404Error = (404, "Not Found")
// http404Error is of type (Int, String), and equals
(404, "Not Found")
```

Tuples

- The (404, "Not Found") tuple groups together an Int and a String to give the HTTP status code two separate values: a number and a human-readable description. It can be described as “a tuple of type (Int, String)”.

Optionals

- You use *optionals* in situations where a value may be absent. An optional represents two possibilities: Either there *is* a value, and you can unwrap the optional to access that value, or there *isn’t* a value at all.

Error Handling

- You use *error handling* to respond to error conditions your program may encounter during execution.
- When a function encounters an error condition, it *throws* an error. That function's caller can then *catch* the error and respond appropriately.

```
func canThrowAnError() throws {
    // this function may or may not throw an error
}
```

- Swift automatically propagates errors out of their current scope until they're handled by a catch clause,

```
do {  try canThrowAnError()
      // no error was thrown
      } catch {
      // an error was thrown }
```

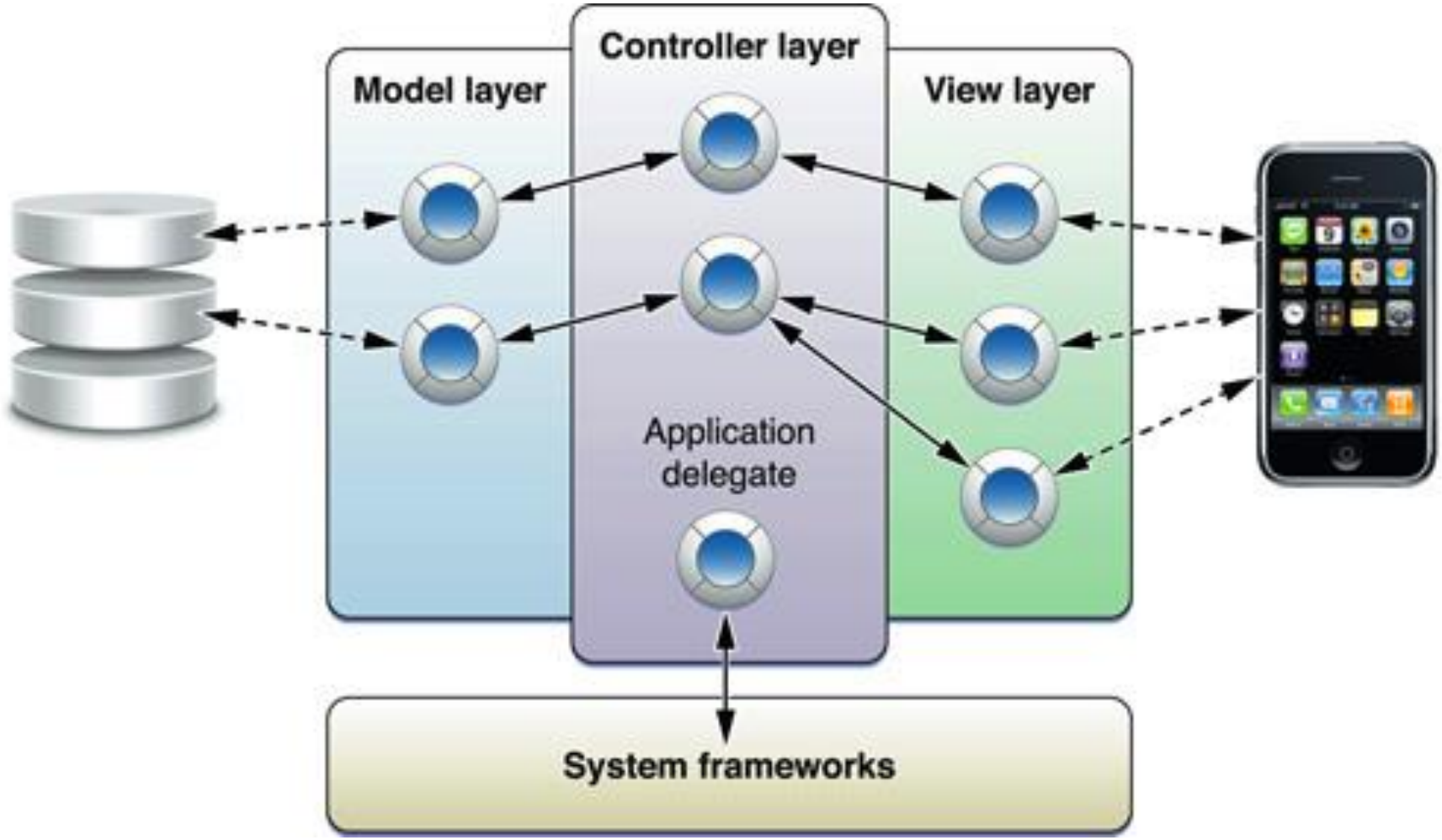
Assertions and Preconditions

- *Assertions* and *preconditions* are checks that happen at runtime.
- You use them to make sure an essential condition is satisfied before executing any further code.
- If the Boolean condition in the assertion or precondition evaluates to true, code execution continues as usual. I
- f the condition evaluates to false, the current state of the program is invalid; code execution ends, and your app is terminated.

Touch frameworks

- **Cocoa Touch** is a UI framework for building software programs to run on iOS for the iPhone and iPod Touch, iPadOS for the iPad, watchOS for the Apple Watch, and tvOS for the fourth-generation Apple TV, from Apple Inc.
- Cocoa Touch provides an abstraction layer of iOS, the operating system for the iPhone, iPod Touch, and iPad.
- Cocoa Touch is based on the macOS Cocoa API toolset and, like it, is primarily written in the Objective-C language.
- Cocoa Touch allows the use of hardware and features that are not found in mac OS computers and are thus unique to the iOS range of devices.
- Just like Cocoa, Cocoa Touch follows a Model-View-Controller (MVC) software architecture.
- Cocoa Touch contains a different set of graphical control elements from Cocoa.
- Tools for developing applications based on Cocoa Touch are included in the iOS SDK.

Touch frameworks



Main Features of Cocoa Touch

- Core Animation: Helps to create rich user experiences by allowing for the smooth movement of visual elements. It also fills in the interim frames of animation with automatic timing and adjustment.
- Core Audio
- Core Data: Provides an object-oriented data management solution and aids in defining an application's data model in a logical and graphical way.
- Cocoa Touch is made up of several frameworks, but the key ones are

  - Audio and Video
    - Core Audio
    - OpenAL
    - Media Library
    - AV Foundation

Main Features of Cocoa Touch

- Data Management
  - Core Data
  - SQLite
- Graphics and Animation
  - Core Animation
  - OpenGL ES
  - Quartz 2D
- Networking and Internet
  - Bonjour
  - WebKit
  - BSD Sockets
- User Applications
  - Address Book
  - Core Location
  - Map Kit
  - Store Kit

Data persistence using Core Data and SQLite

Core Data:

- Graph framework for maintaining objects and their relationships with each other.
- Stores it's data into a SQLite DB but is not a database in itself.
- Manages and updates objects automatically in the memory.
- Can handle a complex graph of objects without any pressure/load on the memory management system of the device.
- Has object versioning system in place which lets you update the object entities and attributes without deleting the previous data.
- Is an Apple created framework, so is automatically updated with any updates in Swift.

Data persistence using Core Data and SQLite

SQLite:

- SQLite is a persistent database model, which make a database and stores it in the file management system of your phone's application.
- It could be used for more lightweight models since it's memory management will be dependant on the Cocoa pods or Frameworks that have been used.
- SQLite Pods or third party providers are easier to use and easier to grasp.
- Can be created and updated just by writing a few custom queries.
- There are many Pods for SQLite in the market but one of the most popular and most stable is **SQLite3**.

Data persistence using Core Data and SQLite

Core Data	SQLite
Core Data is a framework for managing an object graph	SQLite is a relational database
Core Data is a framework for managing an object graph. An object graph is nothing more than a collection of interconnected objects. The framework excels at managing complex object graphs.	SQLite is a library that implements a lightweight database engine that is incredibly performant and, therefore, a great fit for embedded systems, such as mobile devices.
The framework adds a number of other compelling features, such as input validation, data model versioning, and change tracking	Even though SQLite is advertised as a relational database, it is important to realize that the developer is in charge of maintaining the relationships between records stored in the database.
Core Data can use a SQLite database as its persistent store	This itself is a persistent store

Core Data or SQLite ???

- What should you use? Core Data or SQLite?
- Once again, that is the wrong question to ask.
- If you need a lightweight solution and don't need Core Data's feature set, then SQLite may fit your needs.
- If you are managing a complex object graph with many entities, attributes, and relationships, then Core Data is definitely worth considering.
- Getting up to speed with Core Data is easier than you might think.

SQLite

- SQLite can be used in iOS for handling data.
- SQLite is not as powerful as other DMBSs, such as MySQL or SQL Server, as it does not include all of their features. However, its greatness lies mostly to these factors:
  - It’s lightweight.
  - It contains an embedded SQL engine, SQL knowledge can be applied.
  - It works as part of the app itself, and it doesn’t require extra active services.
  - It’s very reliable.
  - It’s fast.
  - It’s fully supported by Apple, as it’s used in both iOS and Mac OS.
  - It has continuous support by developers in the whole world and new features are always added to it.

SQLite 3

- Unfortunately, even though SQLite is supported by Apple, a mechanism or a pre-made database management library does not exist.
- Going into more details, the database class that we will implement will be capable of executing all the standard SQL queries (*select*, *insert*, *update*, *delete*).

SQLite 3 Functions Preview

- ***sqlite3_open***: This function is used to create and open a database file. It accepts two parameters, where the first one is the database file name, and the second a handler to the database. If the file does not exist, then it creates it first and then it opens it, otherwise it just opens it.
- ***sqlite3_prepare_v2***: The purpose of this function is to get a SQL statement (a query) in string format, and convert it to an executable format recognizable by SQLite 3.
- ***sqlite3_step***: This function actually executes a SQL statement (query) prepared with the previous function. It can be called just once for executable queries (insert, update, delete), or multiple times when retrieving data. It's important to have in mind that it can't be called prior to the *sqlite3_preprare_v2* function.

SQLite 3 Functions Preview

- ***sqlite3_column_count***: This method's name it makes it easy to understand what is about. It returns the total number of columns (fields) a contained in a table.
- ***sqlite3_column_text***: This method returns the contents of a column in text format, actually a C string (*char* *) value. It accepts two parameters: The first one is the query converted (compiled) to a SQLite statement, and the second one is the index of the column.
- ***sqlite3_column_name***: It returns the name of a column, and its parameters are the same to the previous function's.
- ***sqlite3_changes***: It actually returns the number of the affected rows, after the execution of a query.

SQLite 3 Functions Preview

- ***sqlite3_last_insert_rowid***: It returns the last inserted row's ID.
- ***sqlite3_errmsg***: It returns the description of a SQLite error.
- ***sqlite3_finalize***: It deletes a prepared statement from memory.
- ***sqlite3_close***: It closes an open database connection. It should be called after having finished any data exchange with the database, as it releases any reserved system resources.

Location handling using iOS Core Location

- Core Location framework is used to determine the current latitude and longitude of a device and to configure and schedule the delivery of location-related events.
- Core Location uses a type of streaming notification so that your application receives updates as the GPS ascertains a more accurate fix.
- There are three technologies that core location uses :
  - GPS Reads microwave signals from multiple satellites to determine the current location
  - Cell Tower Triangulation Determine the current location by calculation based on location of cell towers in iPhone's range.
  - Wi-Fi positioning Service (WPS) Uses IP address from iPhone's Wi-Fi connection by referencing database of service providers and areas they service

Location handling using iOS MapKit

- MapKit is a neat API, comes with the iOS SDK, that allows you to display maps, navigate through maps, add annotations for specific locations, add overlays on existing maps, etc.

Integrating calendar and address book with social media application

Social Media App Feature	Implementation on iOS
Social authorization	Facebook SDK, Twitter SDK, Google+ SDK
Feed	LoadableViews, RxSwift, DTTableViewManager, TRON/Alamofire
Post creation	CoreLocation framework, TRON/Alamofire
Post likes	LoadableViews, RxSwift, TRON/Alamofire
Social notifications	LoadableViews, RxSwift, DTTableViewManager, TRON/Alamofire
User profile	LoadableViews, RxSwift, TRON/Alamofire
Search	LoadableViews, RxSwift, DTTableViewManager, TRON/Alamofire
Chat	We use SwiftActionCable websocket library for communication with Rails ActionCable websockets. CoreData framework is used for offline chats storage.
Push notification module	UserNotifications Framework
Analytics	Analytics SDK

Using Wifi

- With iOS 11, Apple provided public API you can use to programmatically join a WiFi network without leaving your app.
- The class you'll need to use is called NEHotspotConfiguration.
- To use it, you need to enable the Hotspot capability in your App.

```
NEHotspotConfiguration *configuration = [[NEHotspotConfiguration
alloc] initWithSSID:@"SSID-Name"];
configuration.joinOnce = YES; [[NEHotspotConfigurationManager
sharedManager] applyConfiguration:configuration
completionHandler:nil];
```

- This will prompt the user to join the "SSID-Name" WiFi network. It will stay connected to the WiFi until the user leaves the app.

iPhone marketplace.

- **To Promote iOS Apps:**
  - Getting your app/game reviewed ***is a must*** for developers willing to promote their app. Major app review sites like MobileStartupz.com, 148appsm are some good options to consider.
  - Submitting to review sites will provide your app with some initial publicity, and if your app is really good, these sites will also help you ***get noticed*** in the app industry.
  - If you have developed a paid app, **make sure you add a redemption code** so that bloggers can download your paid app for free and write a review.

Uploading to App Store

- Build with Xcode 12.3
- Optimize for iOS 14.3 and iPadOS
- Test your app on devices.
- Submit your apps for review
- Create a App Record in App Store Connect
- Upload the Build