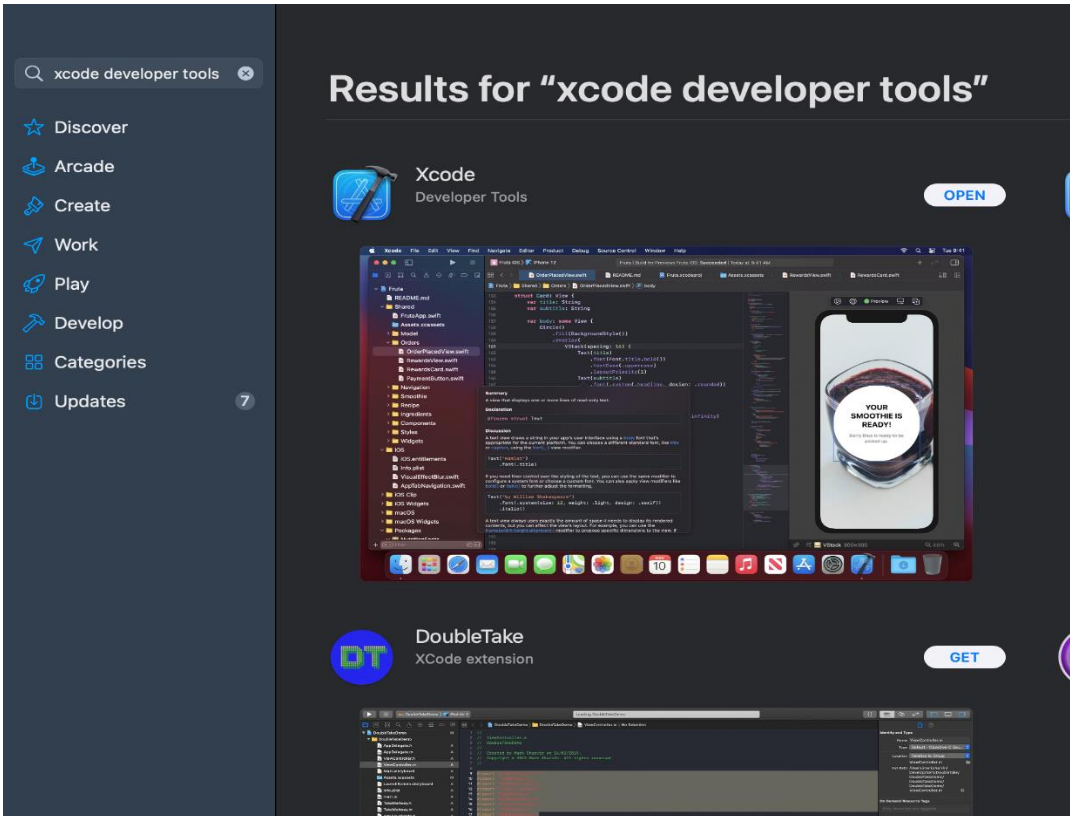
IOS APPLICATION DEVELOPMENT

In this Document , we are going to learn on how to develop a basic IOS Application using Swift programming language from scratch on XCode.

XCode : It is an Apple integrated IDE for macOS used in developing IOS Applications . It is available on App Store .

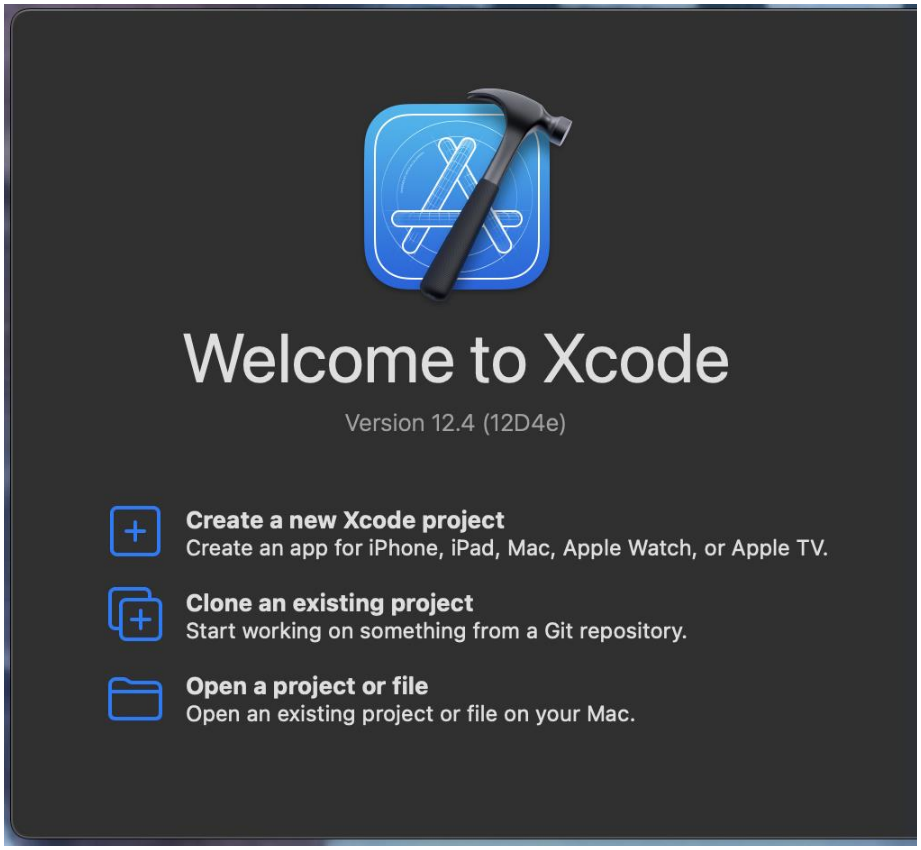
Steps to Install XCode on mac

1. Open App store and search for XCode.
2. If the app is not installed , then download and install the app.



Now , Let us open the app by clicking open , then a window with following options will be displayed.

1. If we want to Create a new Application , then select the first option
2. If we want to clone an existing project , then select the second option
3. If we want to open the existing project which is already present on the mac , then select the third option.



Welcome to Xcode

Version 12.4 (12D4e)

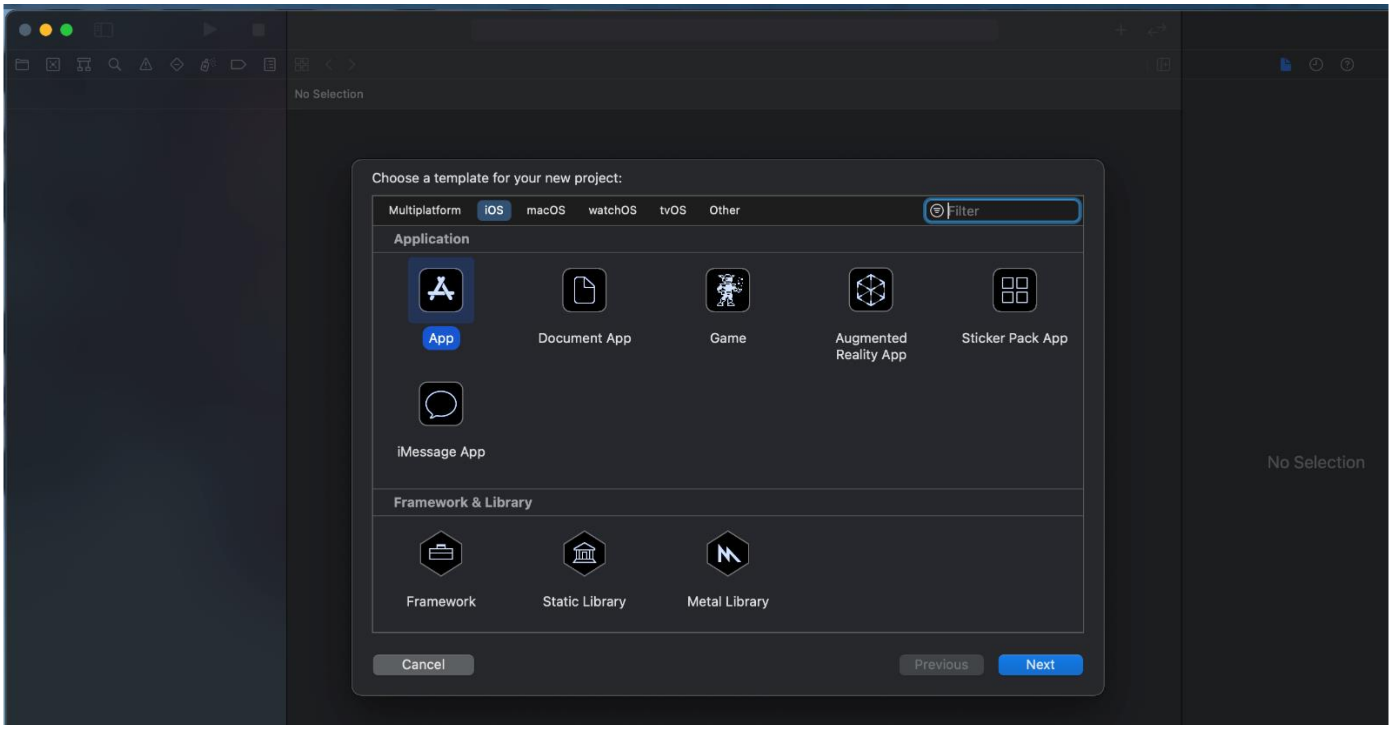
Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.

Clone an existing project
Start working on something from a Git repository.

Open a project or file
Open an existing project or file on your Mac.

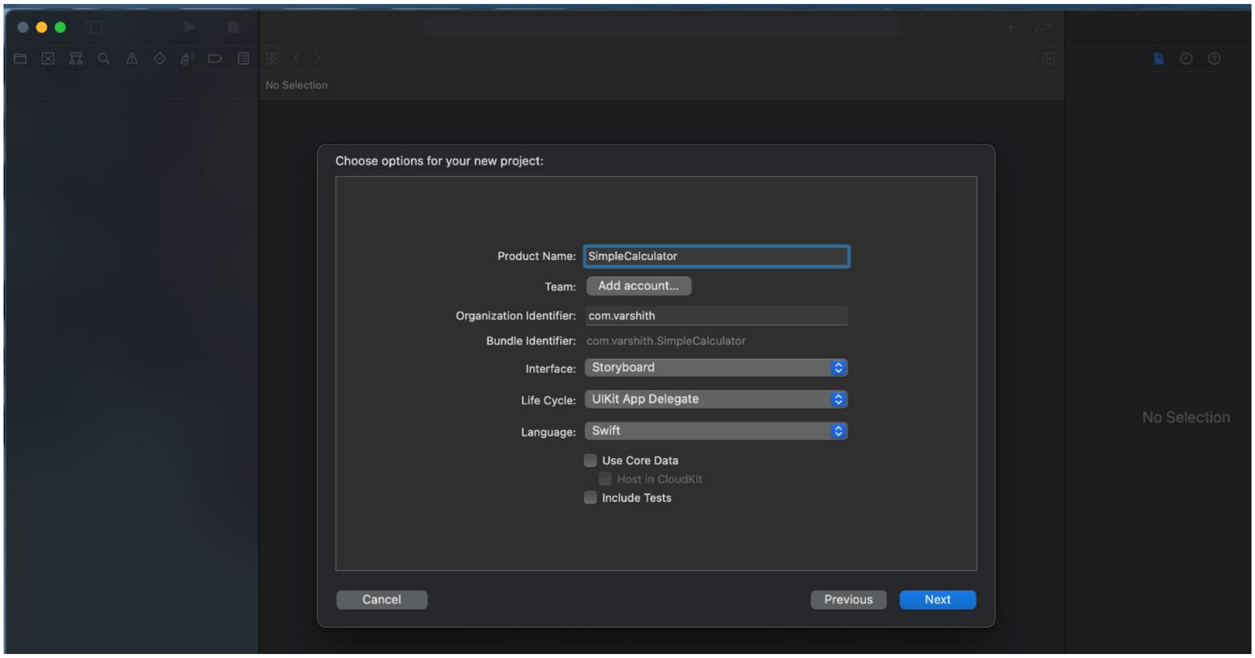
Here we are going to create a new project , so select the first option .

1. A window will be displayed for selecting the template for application.
2. Select the desired template , Here are we will choose **App** template in **iOS** platform



Choose a template for your new project:

Now a window for options of the project is displayed.



Choose options for your new project:

Product Name: SimpleCalculator

Team: Add account...

Organization Identifier: com.varshith

Bundle Identifier: com.varshith.SimpleCalculator

Interface: Storyboard

Life Cycle: UIKit App Delegate

Language: Swift

□ Use Core Data

□ Host in CloudKit

□ Include Tests

1. The first option is Project Name : Give the appropriate project name
2. Team : If the project is developed a group of team , we can create a team and add the account into it.
3. Organization Identifier : If we host a website related to the project , we give the name of the website. We start it with com.
4. Interface : It consists of 2 options
    a. Storyboard
    b. SwiftUI

    In this project we are going to use StoryBoard

5. Language : We are going to Swift Language

After selecting all the options click on next.

The following window gets displayed.

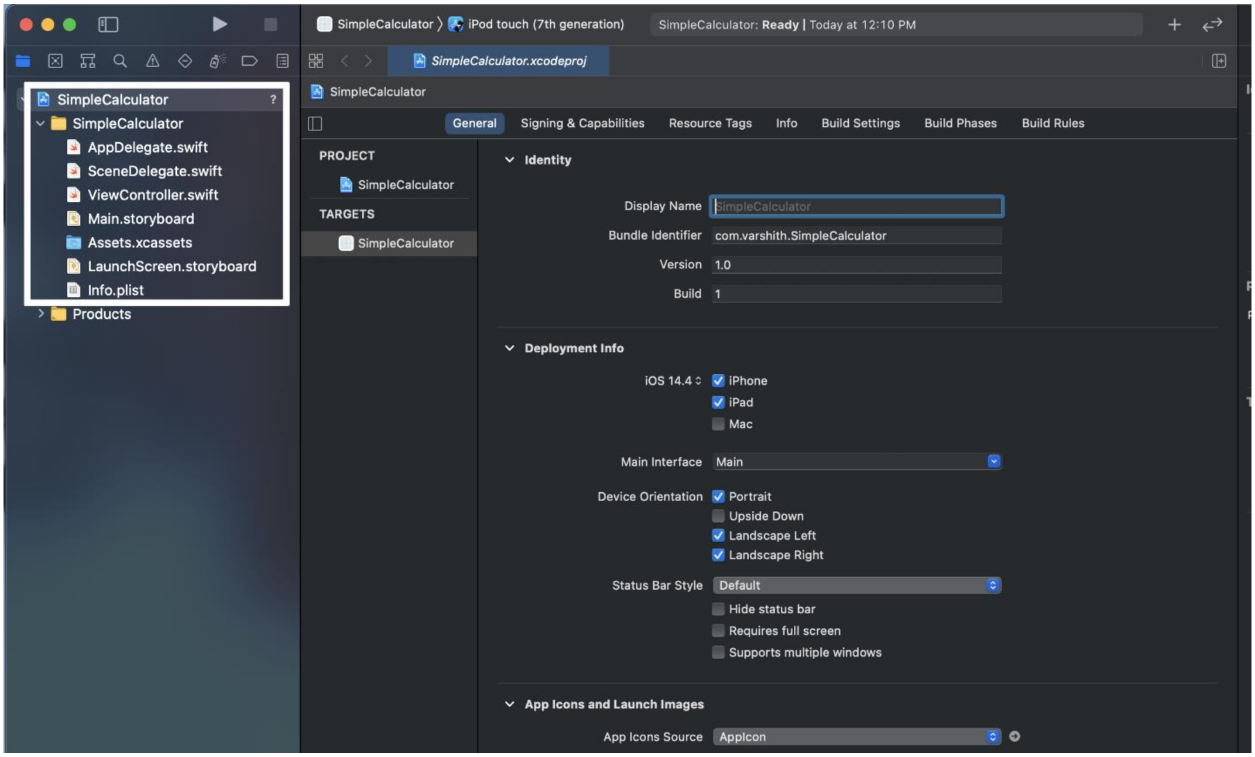
Display Name is the name of the app you are developing and the other attributes are selected based on your preferences.

Deployment information : It shows the software version and the app compatible with.

When ever we create a project , the below highlighted files automatically gets generated.

We most commonly use the below files

1. ViewController.swift
2. Main.storyboard
3. Assets.xcassets



Identity

Display Name SimpleCalculator

Bundle Identifier com.varshith.SimpleCalculator

Version 1.0

Build 1

Deployment Info

iOS 14.4 ☑ iPhone ☑ iPad □ Mac

Main Interface Main

Device Orientation ☑ Portrait □ Upside Down ☑ Landscape Left ☑ Landscape Right

Status Bar Style Default

□ Hide status bar

□ Requires full screen

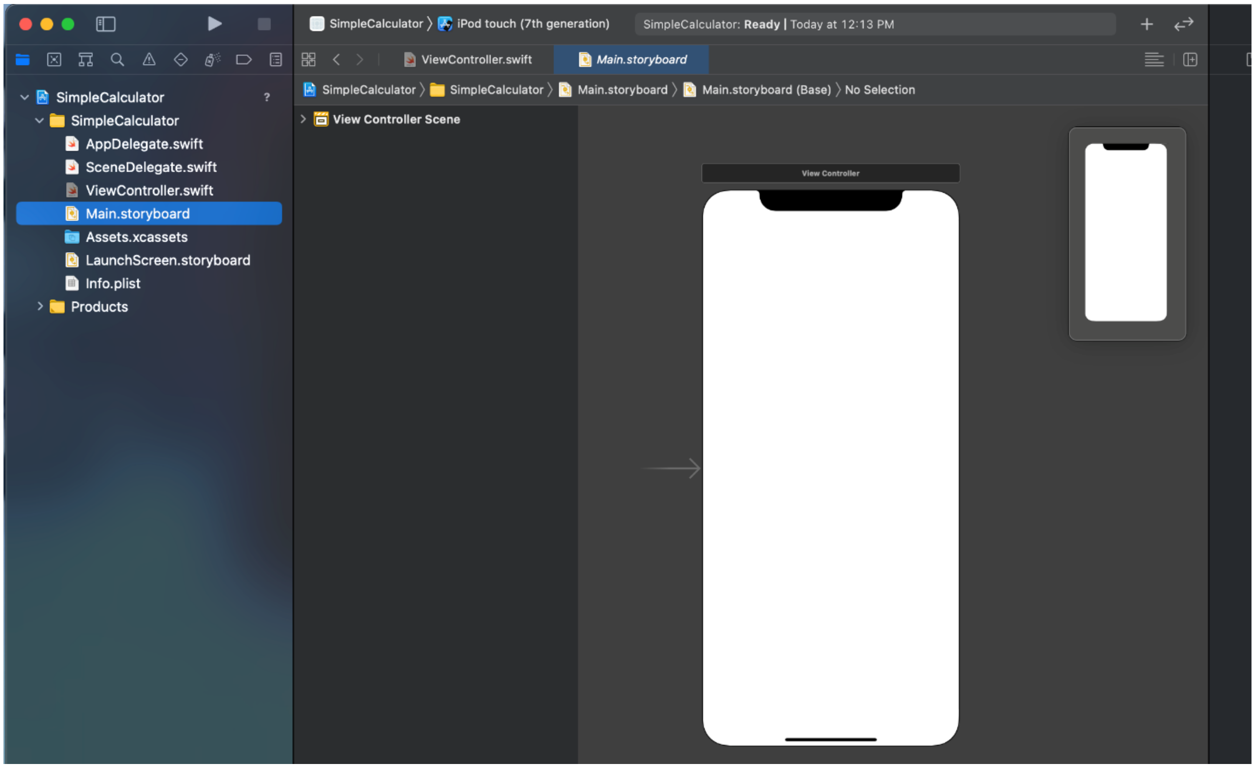
□ Supports multiple windows

App Icons and Launch Images

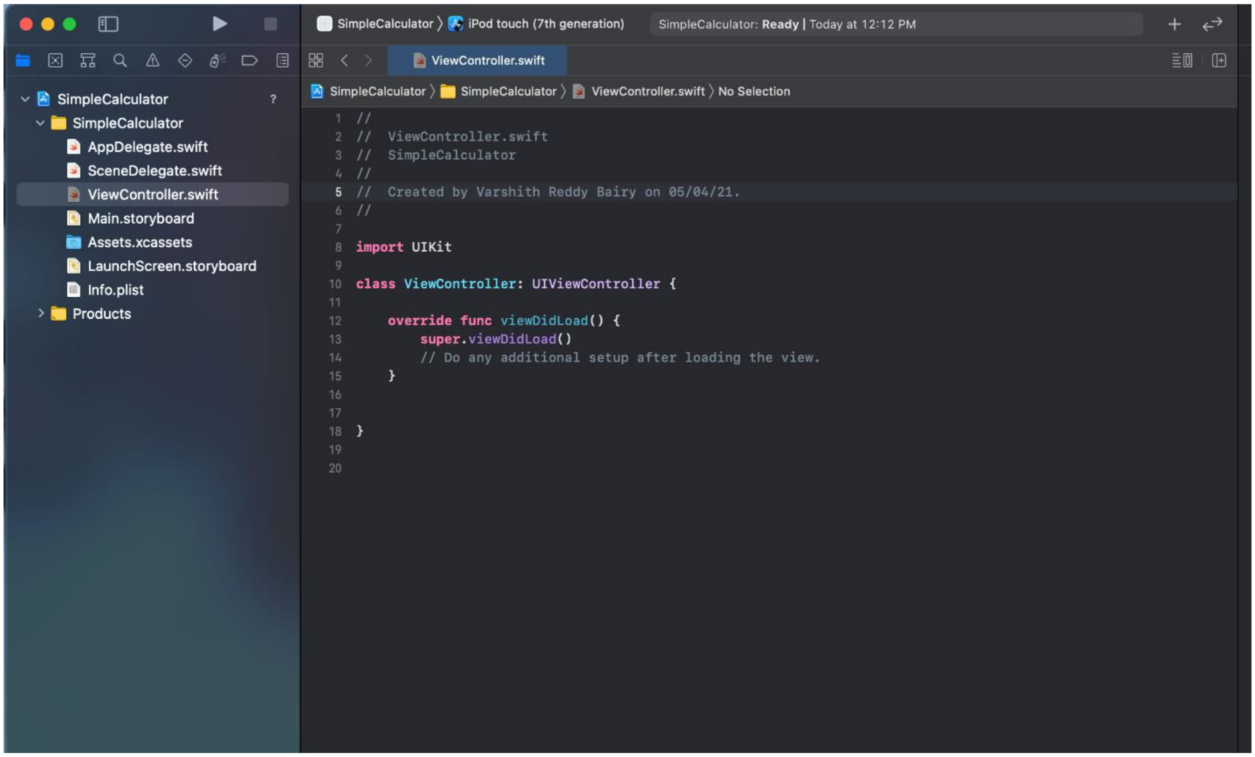
App Icons Source AppIcon

Main.storyboard : It is the representation of user view for an application. It is the screen that user could see when designing the application

Below is the white screen which is displayed before we start developing the app.



ViewContoller.swift : It is the controller to which the view is linked . The actions which we perform in the storyboard are controlled in this ViewController file.



```swift
//
//  ViewController.swift
//  SimpleCalculator
//
//  Created by Varshith Reddy Bairy on 05/04/21.
//

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }


}
```

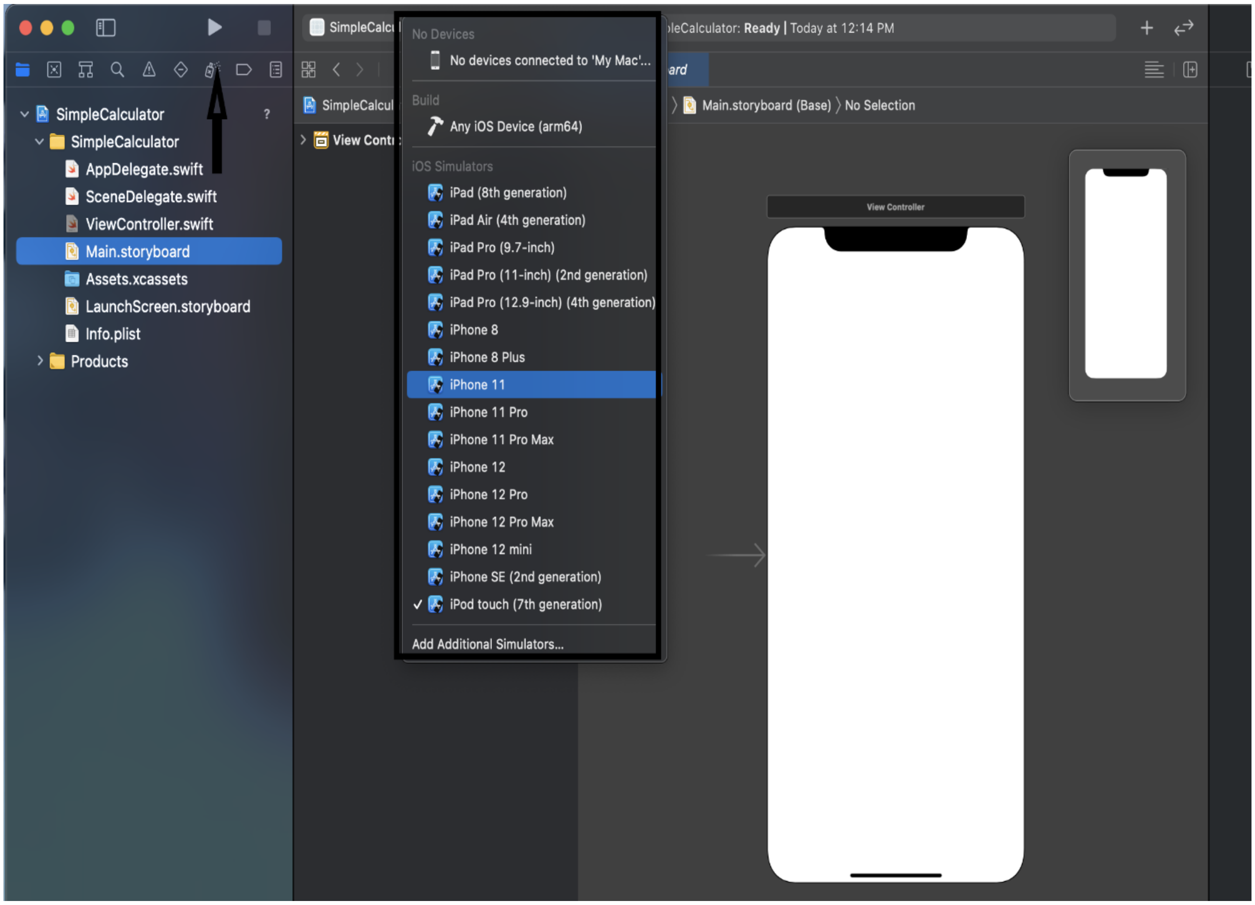
Assets.xcassets : If we want to use any images or logos in the application , we drag and drop the images into this particular folder.

LaunchScreen.storyboard : When we run the application , first the launch screen will be displayed ,

If we want to display anything before the home page of the app gets displayed , we can use this file.

At the top , we can select on which device the application can be run , we can select one from the list.

When ever we click the run button , the app begins to start and a simulator gets popped up where we can use the running app.

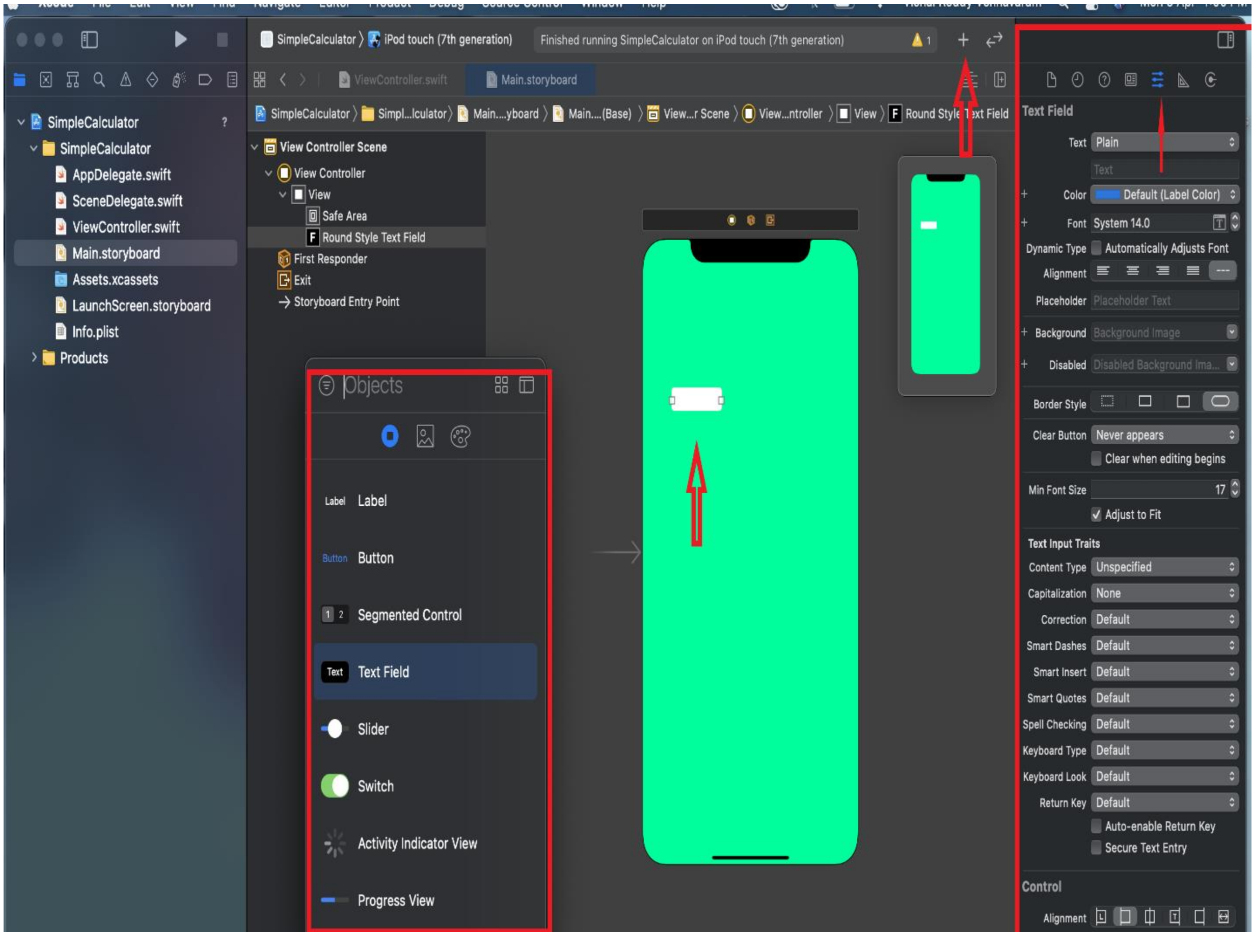


Now in the Storyboard we can add in what ever the fields we want like TextBox , Labels , Buttons

Click the “+” symbol to the top right corner to find the all the available fields . We can search for any field by searching in the search bar.

Now drag the desired field (let us say TextBox as shown in the figure) on to the storyboard.

When we click on the textbox on the storyboard we have a property field to the right of the screen, where we can edit the properties related to the textbox like color of the text , background , height and width etc.



Text Field

Text: Plain

Color: Default (Label Color)

Font: System 14.0

Dynamic Type: ☐ Automatically Adjusts Font

Placeholder: Placeholder Text

Background: Background Image

Disabled: Disabled Background Ima...

Clear Button: Never appears

☐ Clear when editing begins

Min Font Size: 17

☑ Adjust to Fit

Text Input Traits

Content Type: Unspecified

Capitalization: None

Correction: Default

Smart Dashes: Default

Smart Insert: Default

Smart Quotes: Default

Spell Checking: Default

Keyboard Type: Default

Keyboard Look: Default

Return Key: Default

☐ Auto-enable Return Key

☐ Secure Text Entry

Now we have to link the view to the controller , so that we can perform operations on that particular field.

Open the ViewController in the other window by selecting the option highlighted in the screenshot.

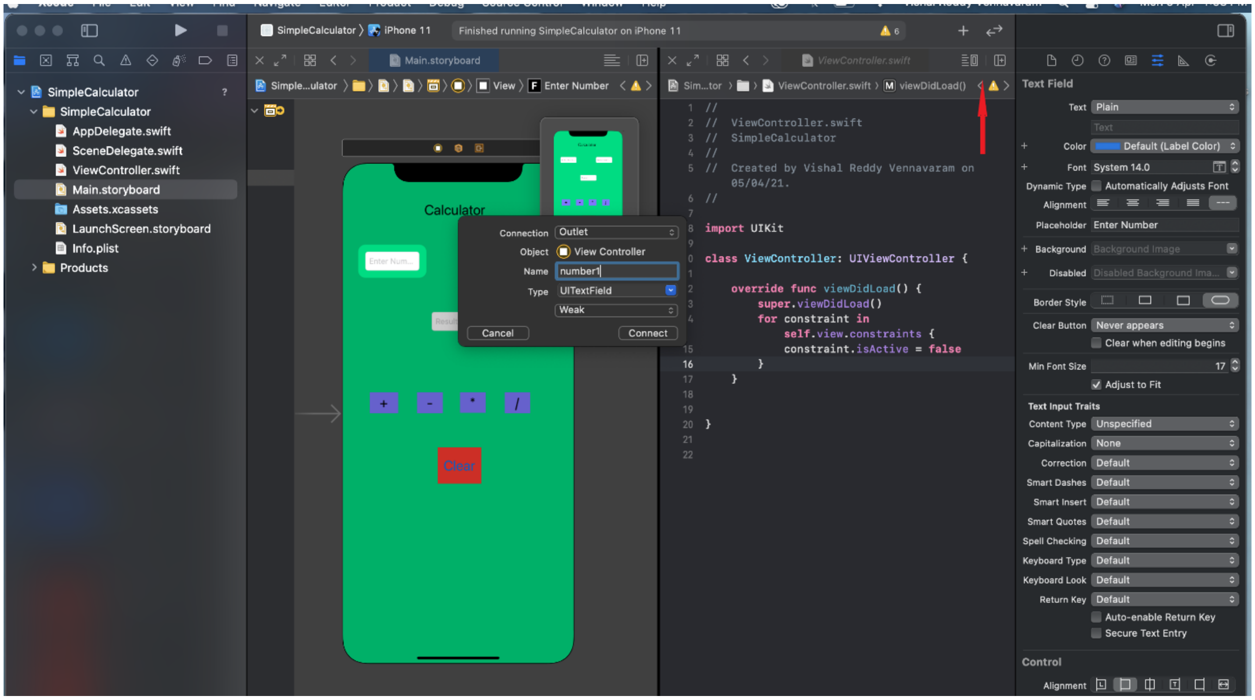
Press **“Control”** and drag the desired field on to the ViewController to link that particular field.

We get several options

1.Connection : we select outlet for textboxes and labels , Action for buttons because some action needs to be performed when this button is clicked

2. Name : It is the name we want to give to that particular field.

3 . Type : There are two options (Any, UIField)



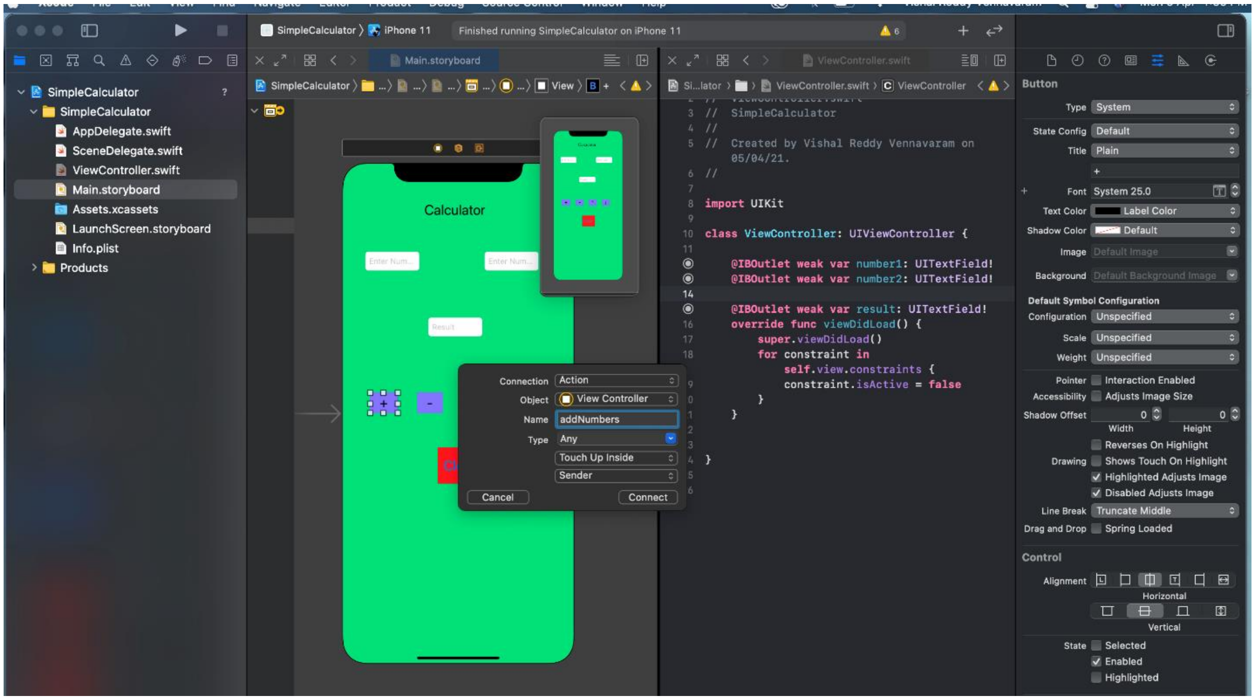
Finished running SimpleCalculator on iPhone 11

Connection: Outlet
Object: View Controller
Name: number1
Type: UITextField
Weak

```
//
//  ViewController.swift
//  SimpleCalculator
//
//  Created by Vishal Reddy Vennavaram on
    05/04/21.
//

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        for constraint in
            self.view.constraints {
            constraint.isActive = false
        }
    }


}
```

Text Field
Text: Plain
Color: Default (Label Color)
Font: System 14.0
Dynamic Type: □ Automatically Adjusts Font
Placeholder: Enter Number
Background: Background Image
Disabled: Disabled Background Ima...
Clear Button: Never appears
□ Clear when editing begins
Min Font Size: 17
☑ Adjust to Fit

Text Input Traits
Content Type: Unspecified
Capitalization: None
Correction: Default
Smart Dashes: Default
Smart Insert: Default
Smart Quotes: Default
Spell Checking: Default
Keyboard Type: Default
Keyboard Look: Default
Return Key: Default
□ Auto-enable Return Key
□ Secure Text Entry



```
//  SimpleCalculator
//
//  Created by Vishal Reddy Vennavaram on
    05/04/21.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var number1: UITextField!
    @IBOutlet weak var number2: UITextField!

    @IBOutlet weak var result: UITextField!
    override func viewDidLoad() {
        super.viewDidLoad()
        for constraint in
            self.view.constraints {
            constraint.isActive = false
        }
    }


}
```

Connection: Action
Object: View Controller
Name: addNumbers
Type: Any
Touch Up Inside
Sender

Type: System
State Config: Default
Title: Plain
Font: System 25.0
Text Color: Label Color
Shadow Color: Default
Image: Default Image
Background: Default Background Image
Default Symbol Configuration
Configuration: Unspecified
Scale: Unspecified
Weight: Unspecified
Pointer: □ Interaction Enabled
Accessibility: □ Adjusts Image Size
Shadow Offset: 0 Width, 0 Height
□ Reverses On Highlight
Drawing: □ Shows Touch On Highlight
☑ Highlighted Adjusts Image
☑ Disabled Adjusts Image
Line Break: Truncate Middle
Drag and Drop: □ Spring Loaded
Alignment: Horizontal, Vertical
State: □ Selected
☑ Enabled
□ Highlighted

For the button to work as expected , we write the code for the button in the controller.

Here for this project I have taken 2 textboxes , 1 label and 5 buttons

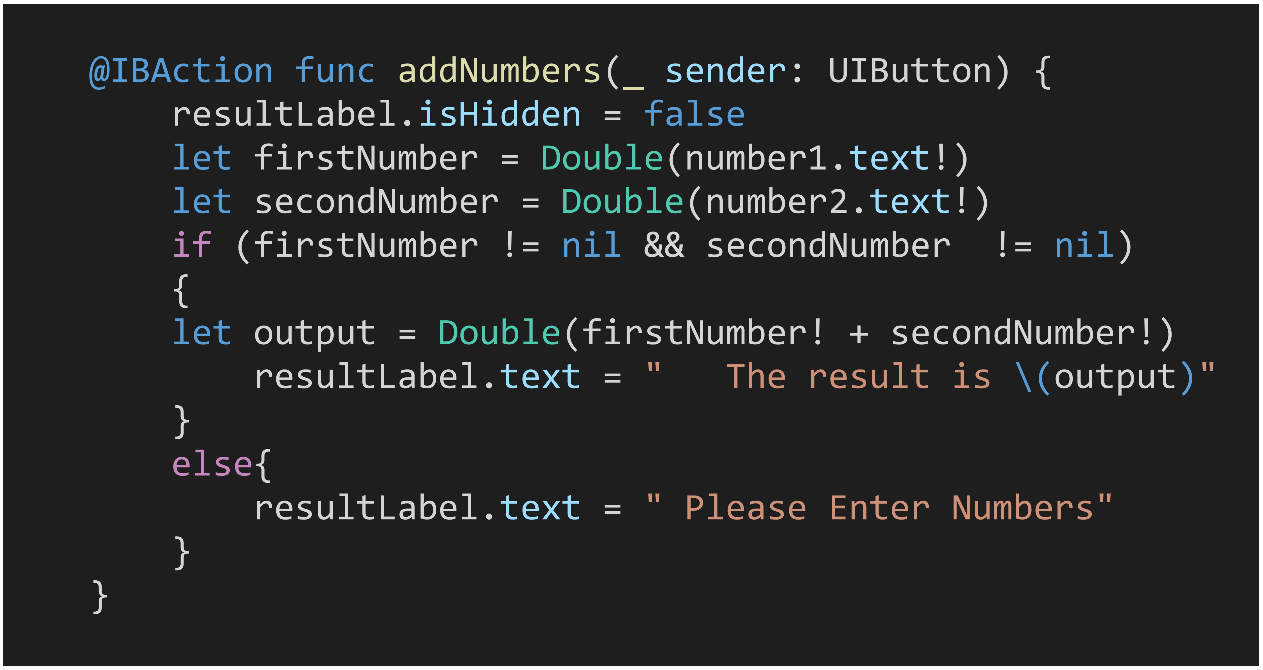
In the textboxes , we enter the numbers on which we perform the mathematical operations like add, subtract, multiply, divide .

The label is to display the output of the operation performed.

The 4 buttons are for add , subtract , multiply and divide respectively .

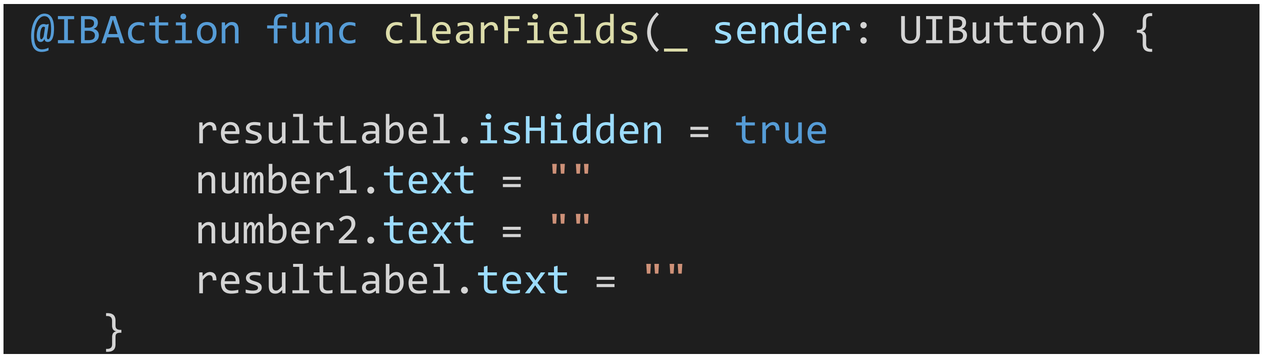
The last button is Clear button , when clicked it clears the text from textboxes and label.

Here is the code for add button



```
@IBAction func addNumbers(_ sender: UIButton) {
    resultLabel.isHidden = false
    let firstNumber = Double(number1.text!)
    let secondNumber = Double(number2.text!)
    if (firstNumber != nil && secondNumber  != nil)
    {
    let output = Double(firstNumber! + secondNumber!)
        resultLabel.text = "   The result is \(output)"
    }
    else{
        resultLabel.text = " Please Enter Numbers"
    }
}
```

For Clear button , the code is as follows.

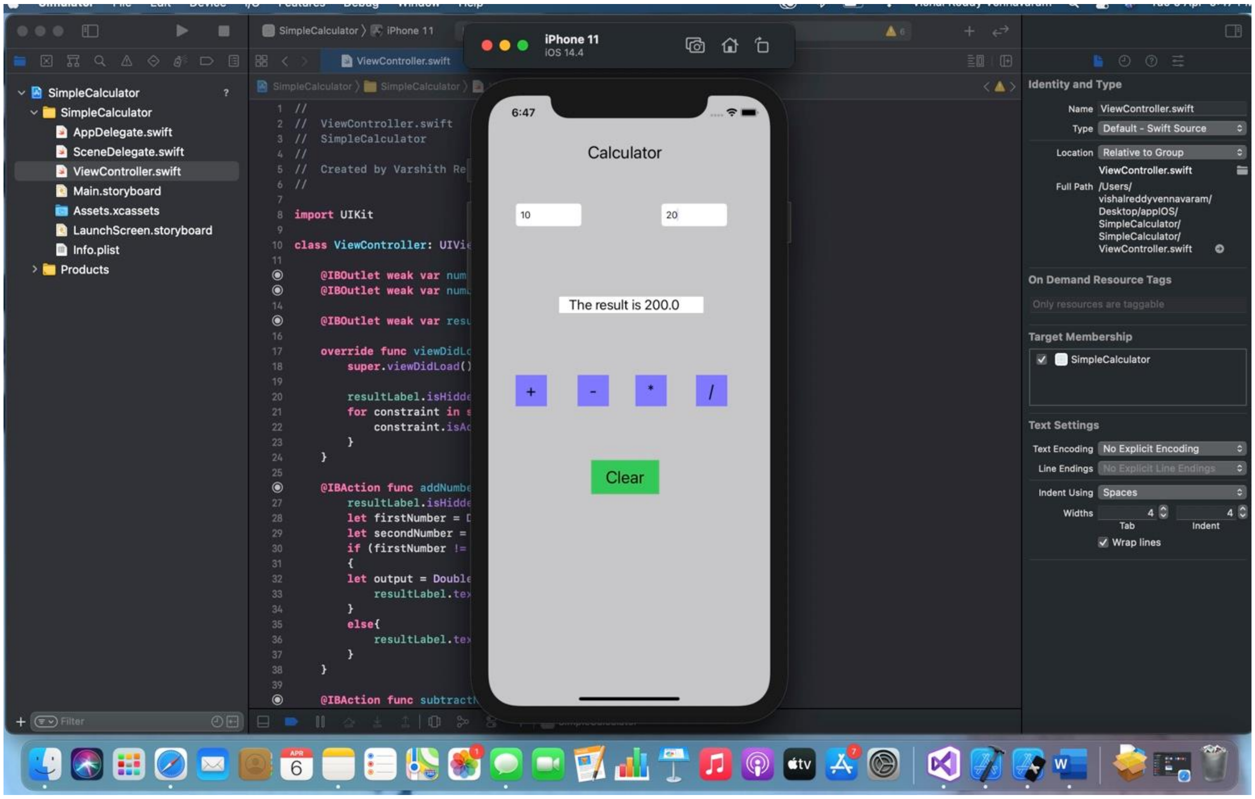


```
@IBAction func clearFields(_ sender: UIButton) {

       resultLabel.isHidden = true
       number1.text = ""
       number2.text = ""
       resultLabel.text = ""
   }
```

Here is the task.

Similarly write the code for remaining buttons and make the buttons work.

Here is the output of the simulator when we multiply 2 numbers.



```
//
//  ViewController.swift
//  SimpleCalculator
//
//  Created by Varshith Re
//

import UIKit

class ViewController: UIVie

    @IBOutlet weak var num
    @IBOutlet weak var num

    @IBOutlet weak var resu

    override func viewDidLo
        super.viewDidLoad()

        resultLabel.isHidde
        for constraint in s
            constraint.isAc
        }
    }

    @IBAction func addNumbe
        resultLabel.isHidde
        let firstNumber = D
        let secondNumber =
        if (firstNumber !=
        {
        let output = Double
            resultLabel.tex
        }
        else{
            resultLabel.tex
        }
    }

    @IBAction func subtract
```

Calculator

10

20

The result is 200.0

Identity and Type

Name ViewController.swift

Type Default - Swift Source

Location Relative to Group

ViewController.swift

Full Path /Users/vishalreddyvennavaram/Desktop/appIOS/SimpleCalculator/SimpleCalculator/ViewController.swift

On Demand Resource Tags

Only resources are taggable

Target Membership

☑ SimpleCalculator

Text Settings

Text Encoding No Explicit Encoding

Line Endings No Explicit Line Endings

Indent Using Spaces

Widths 4 Tab 4 Indent

☑ Wrap lines