**[Lab 2: Exploring Button Creation](https://bbhosted.cuny.edu/webapps/assignment/uploadAssignment?content_id=_79920921_1&course_id=_2286381_1&group_id=&mode=view)**

**1. Create a New SwiftUI Project:**

* Launch Xcode.
* Create a new SwiftUI project.
* Name it "ButtonDemo" (or any name you prefer).
* Save it to a location of your choice.

**2. Creating a Button with Text:**

* Use the Button view with a Text view inside.
* Customize the button's appearance using modifiers like .foregroundColor, .background, and .padding.
* Implement a basic action for the button, e.g., print a message when it's tapped.

***Source code:***

import SwiftUI

struct ContentView: View {

var body: some View {

Button(action: {

print("Button tapped")

}) {

Text("Tap Me")

.foregroundColor(.white)

.padding()

.background(Color.blue)

.cornerRadius(10)

}

}

}

struct ContentView\_Previews: PreviewProvider {

static var previews: some View {

ContentView()

}

}

A screen shot of a computer

Description automatically generated

**3. Using Image as a Button:**

* Explore how to use an Image view as a button by attaching a tap gesture.
* Customize the button's appearance with modifiers.
* Implement a different action for this button.

***Source code:***

import SwiftUI

struct ContentView: View {

var body: some View {

Button(action: {

print("Image Button tapped")

}) {

Image(systemName: "heart.fill")

.resizable()

.frame(width: 40, height: 40)

.foregroundColor(.red)

.padding(10)

.background(Color.gray)

.cornerRadius(20)

}

}

}

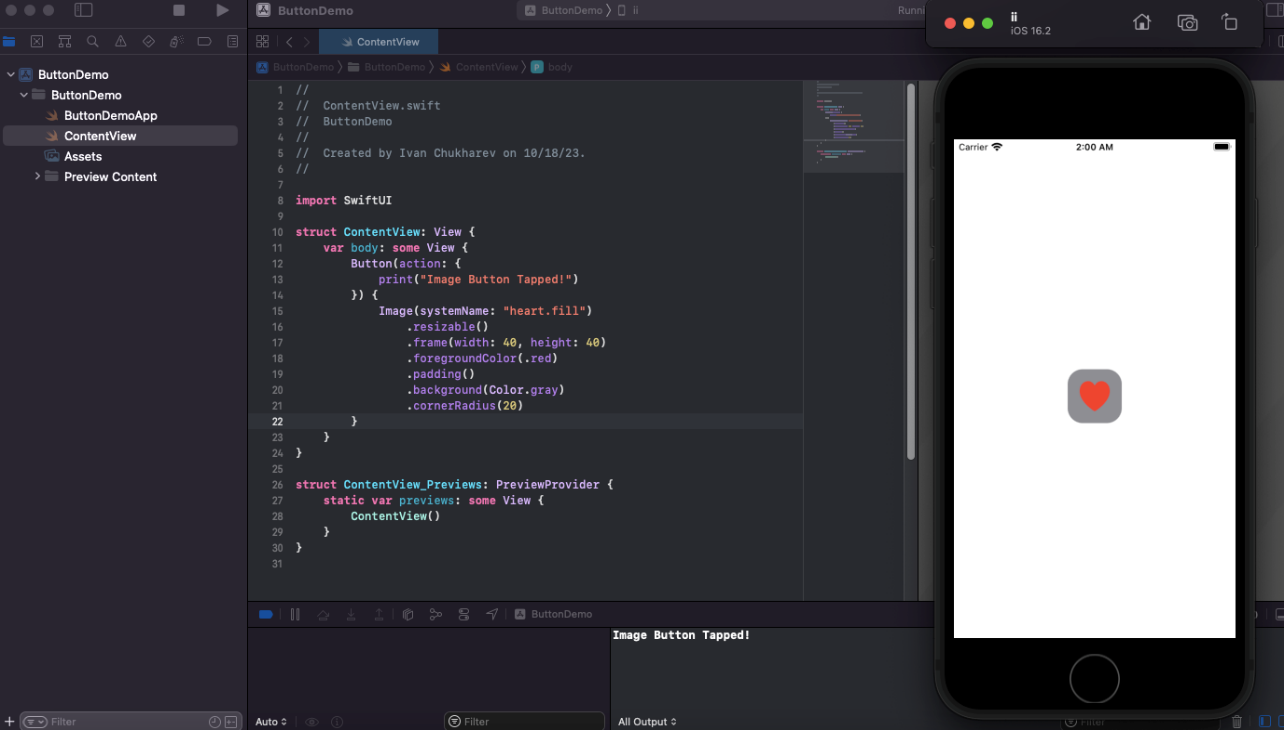
struct ContentView\_Previews: PreviewProvider {

static var previews: some View {

ContentView()

}

}



**4. Creating a Toggle Button:**

* Learn how to create a toggle button using the Toggle view.
* Customize the toggle button's appearance with modifiers.
* Implement actions that respond to the toggle state.

***Source code:***

import SwiftUI

struct ContentView: View {

@State private var showDetails = false

var body: some View {

Button("Show details"){

showDetails.toggle()

}.font(.title)

.padding()

.background(showDetails ? Color.green : Color.gray)

.foregroundColor(.white)

.cornerRadius(10)

if showDetails {

Image("nycpic")

.resizable()

.scaledToFit()

}

}

}

struct ContentView\_Previews: PreviewProvider {

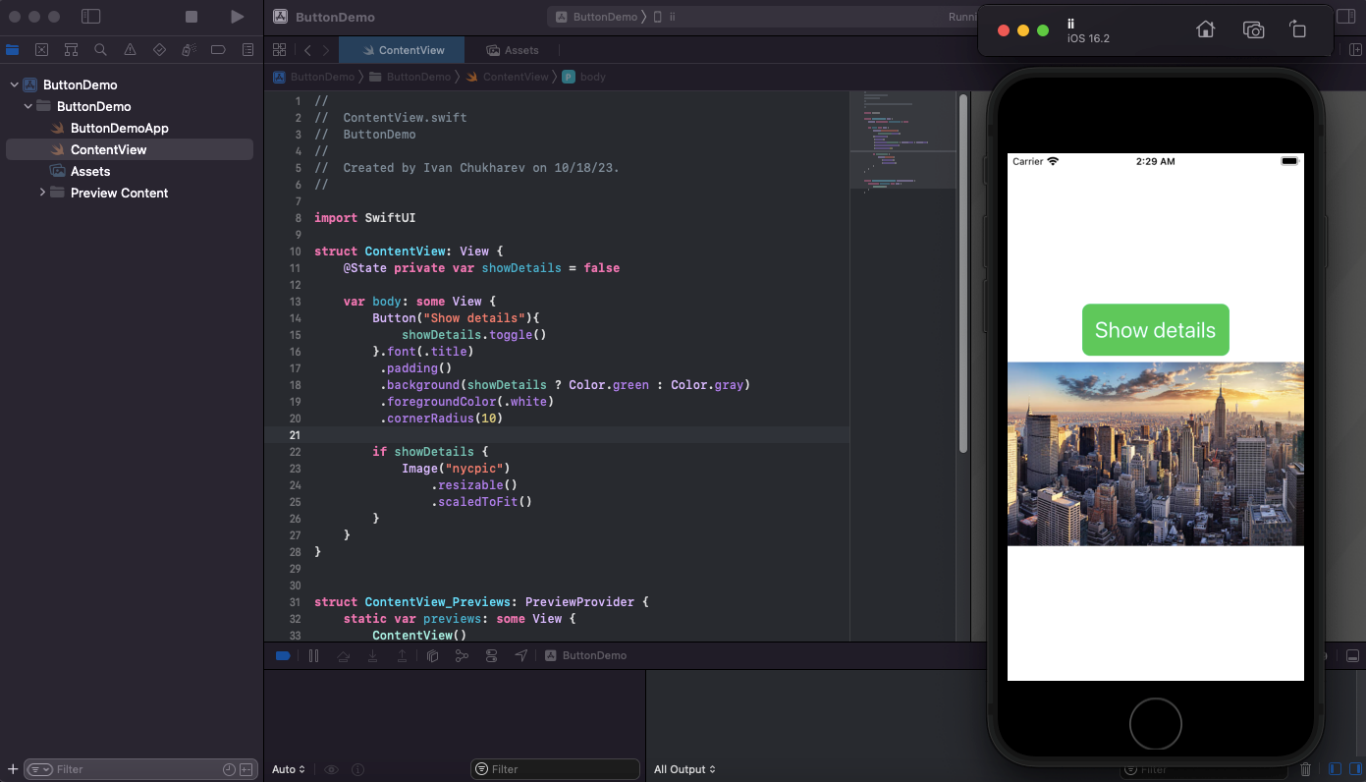
static var previews: some View {

ContentView()

}

}

A screen shot of a phone

Description automatically generated

**5. Using NavigationLink as a Button:**

* Create a button-like behavior using the NavigationLink view.
* Explore how to navigate to a different view when the link is tapped.
* Customize the link's appearance with modifiers.

***Source code:***

import SwiftUI

struct ContentView: View {

var body: some View {

NavigationView {

NavigationLink(destination: Text("New View")) {

Text("Go to New View")

.foregroundColor(.white)

.padding()

.background(Color.blue)

.cornerRadius(10)

}

}

}

}

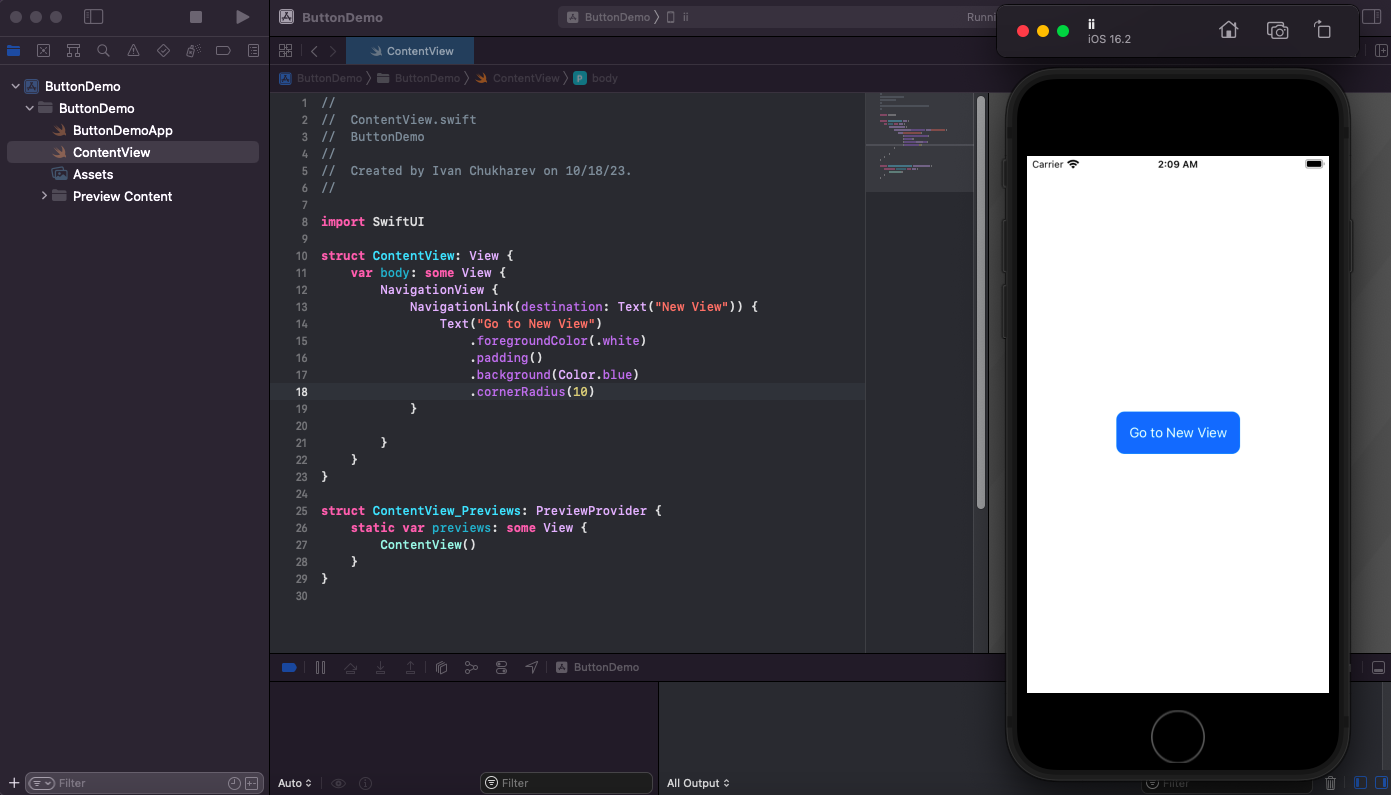
struct ContentView\_Previews: PreviewProvider {

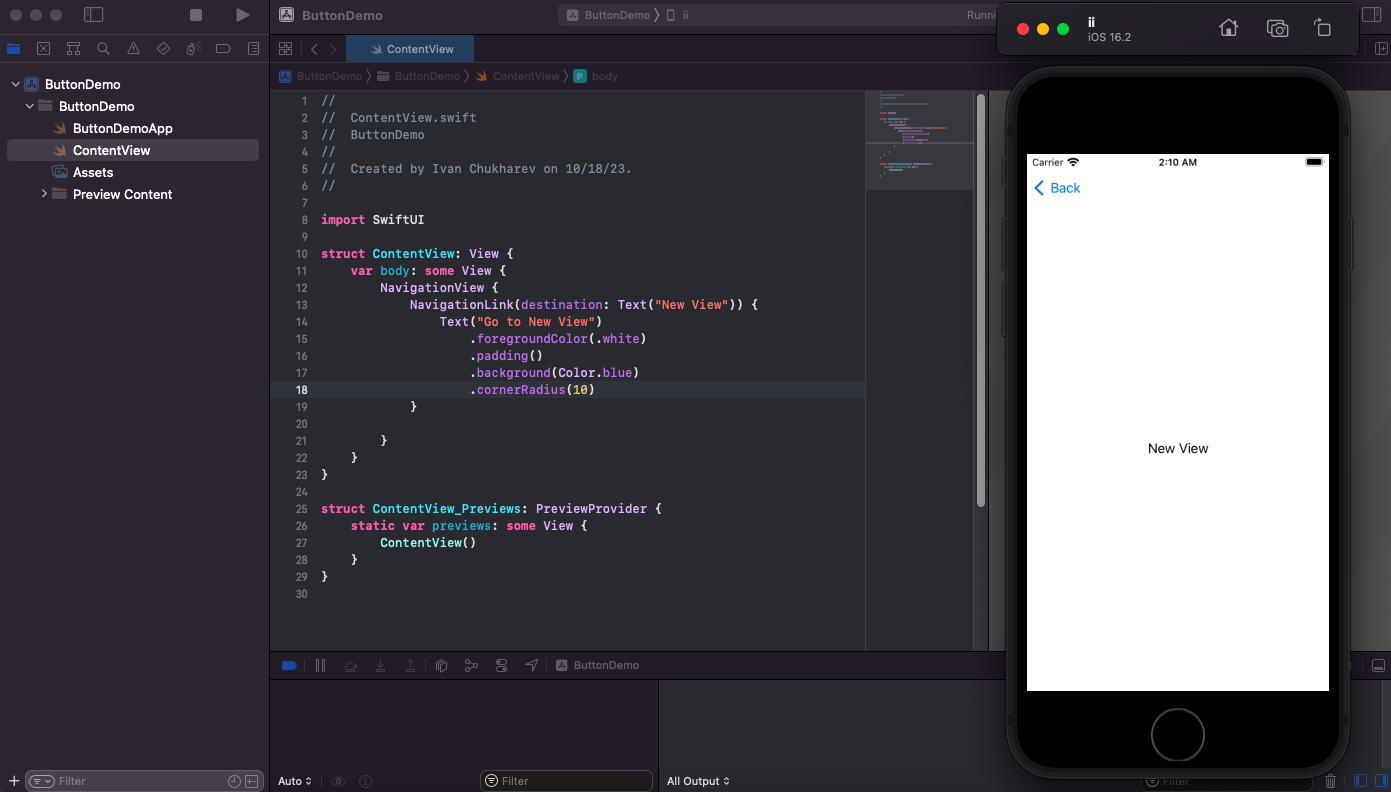
static var previews: some View {

ContentView()

}

}





**6. Button with Icons:**

* Create a button with an icon by combining an Image view and Text view within a Button.
* Customize the button's appearance, and implement an action to respond to its t

***Source code:***

import SwiftUI

struct ContentView: View {

var body: some View {

Button(action: {

print("Button with Icon Tapped!")

}) {

HStack {

Image(systemName: "star.fill")

.resizable()

.frame(width: 20, height: 20)

Text("Favorite")

}

.foregroundColor(.white)

.padding()

.background(Color.purple)

.cornerRadius(10)

}

}

}

struct ContentView\_Previews: PreviewProvider {

static var previews: some View {

ContentView()

}

}

