#WWDC19

SwiftUI Essentials

Matt Ricketson, SwiftUI Engineer
Taylor Kelly, SwiftUI Engineer

© 2019 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

The shortest path to a great app

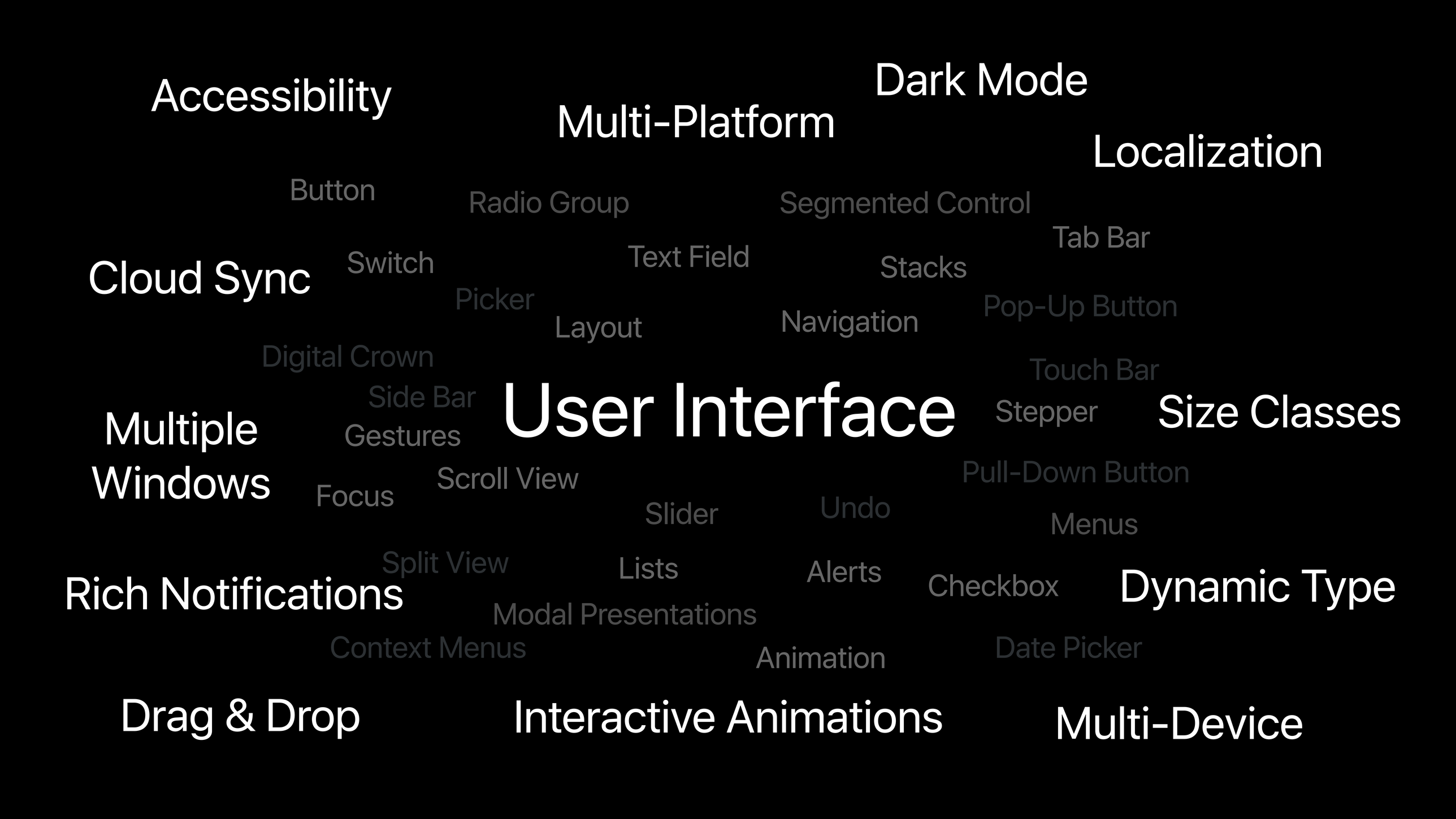
The shortest path to a great UI

User Interface

Button
Radio Group
Segmented Control
Tab Bar
Switch
Text Field
Stacks
Picker
Pop-Up Button
Layout
Navigation
Digital Crown
Touch Bar
Side Bar
Stepper
Gestures
Pull-Down Button
Scroll View
Focus
Slider
Undo
Menus
Split View
Lists
Alerts
Checkbox
Modal Presentations
Context Menus
Animation
Date Picker

User Interface

Button
Radio Group
Segmented Control
Tab Bar
Switch
Text Field
Stacks
Picker
Pop-Up Button
Layout
Navigation
Digital Crown
Touch Bar
Side Bar
Stepper
Gestures
Pull-Down Button
Scroll View
Focus
Slider
Undo
Menus
Split View
Lists
Alerts
Checkbox
Modal Presentations
Context Menus
Animation
Date Picker


Accessibility
Dark Mode
Multi-Platform
Localization
Button
Radio Group
Segmented Control
Tab Bar
Cloud Sync
Switch
Text Field
Stacks
Picker
Pop-Up Button
Layout
Navigation
Digital Crown
Touch Bar
Side Bar
User Interface
Stepper
Size Classes
Multiple Windows
Gestures
Pull-Down Button
Scroll View
Focus
Slider
Undo
Menus
Split View
Lists
Alerts
Checkbox
Rich Notifications
Dynamic Type
Modal Presentations
Context Menus
Animation
Date Picker
Drag & Drop
Interactive Animations
Multi-Device

Your Awesome Feature

Your Surprise & Delight

Accessibility

Multi-Platform

Dark Mode

Button

Radio Group

Segmented Control

Localization

Tab Bar

Cloud Sync

Switch

Text Field

Stacks

Picker

Pop-Up Button

Layout

Navigation

Digital Crown

Touch Bar

Side Bar

User Interface

Stepper

Size Classes

Multiple Windows

Gestures

Pull-Down Button

Scroll View

Focus

Slider

Undo

Menus

Dynamic Type

Split View

Lists

Checkbox

Rich Notifications

Alerts

Modal Presentations

Date Picker

Multi-Device

Your Unique Design

Context Menus

Animation

Your Clever Animation

Drag & Drop

Interactive Animations

Your Custom Control

Your App

Basic
features

Your App

Basic features

Exciting, custom features!

Your App

Getting started: Views and modifiers

Building custom views

Composing controls

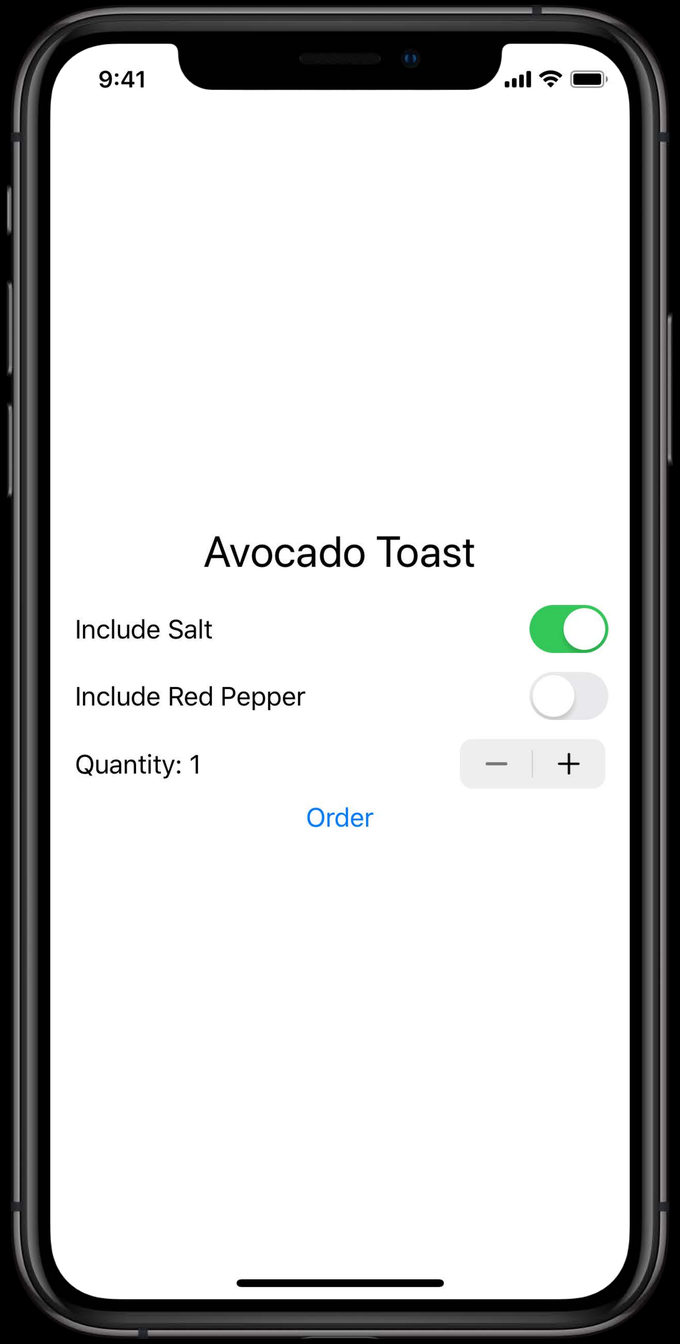
Navigating your app

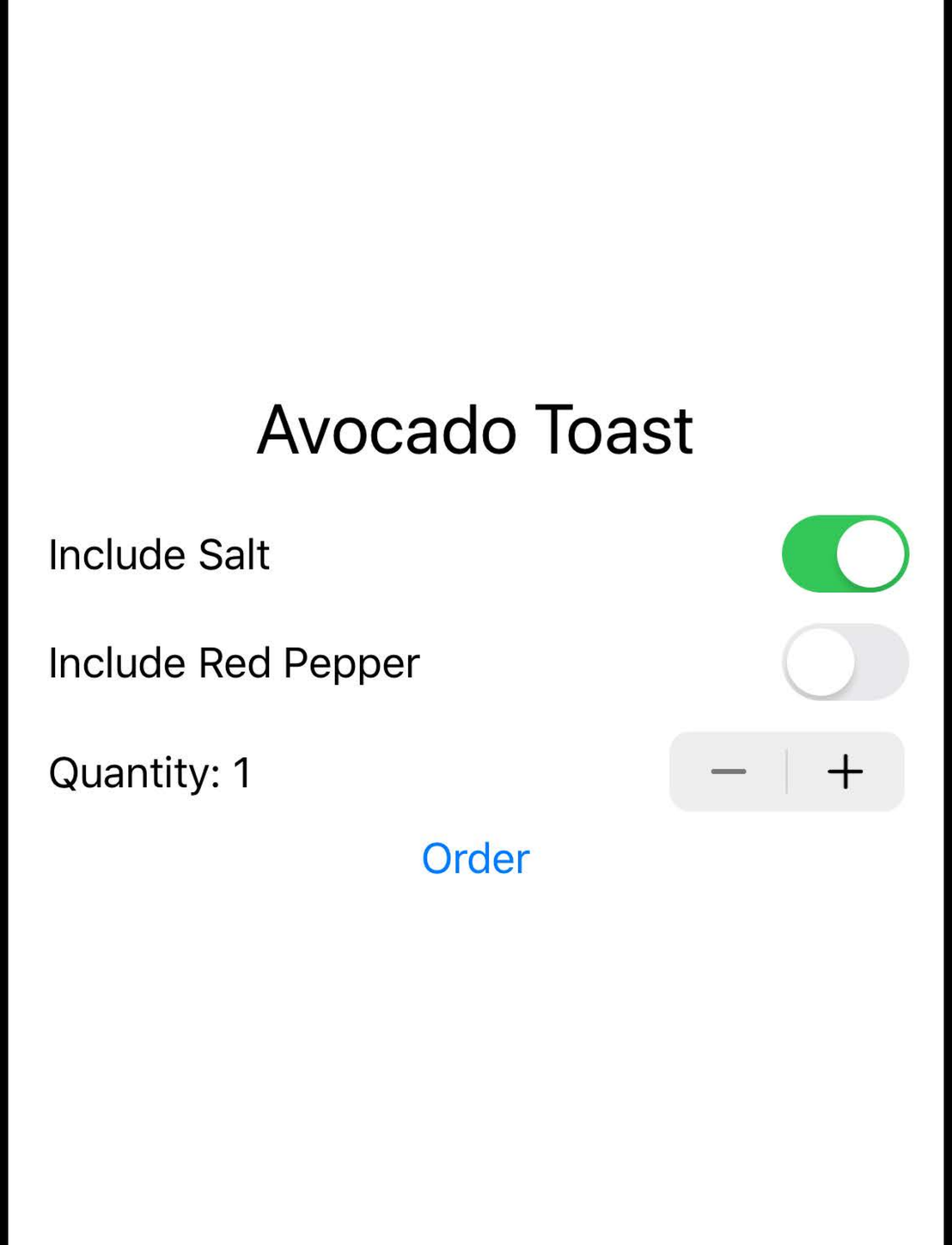
Getting started: Views and modifiers

Building custom views

Composing controls

Navigating your app


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order

Views

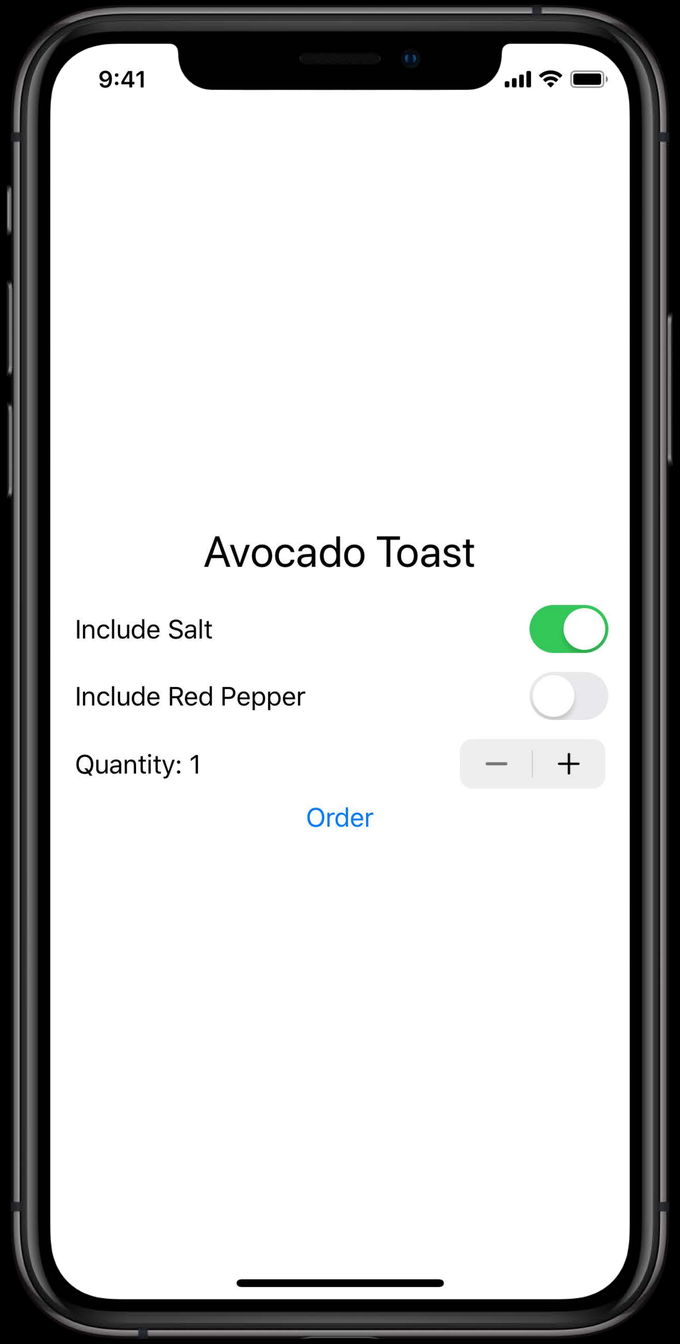
`UIView`

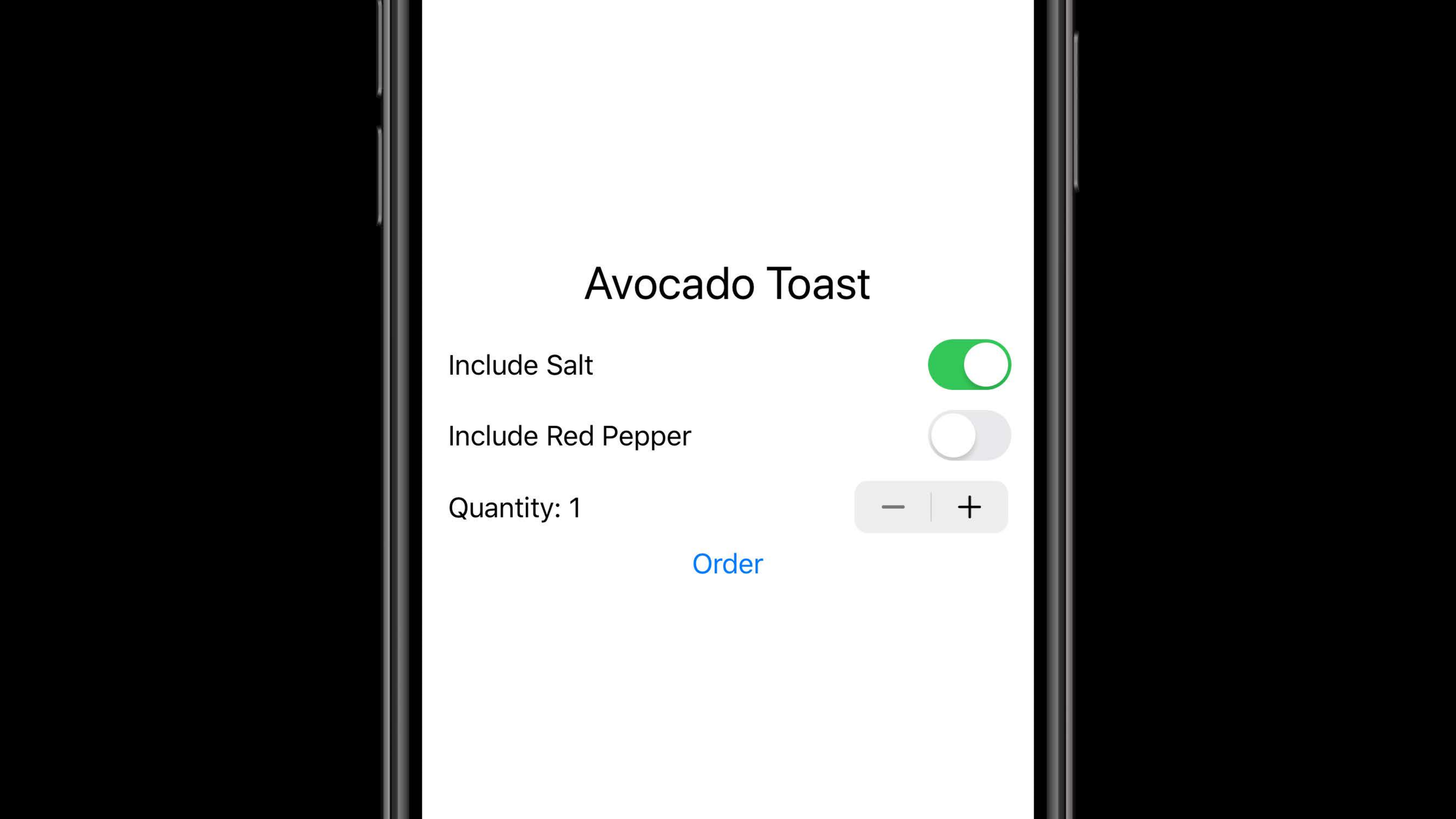
(UIKit)

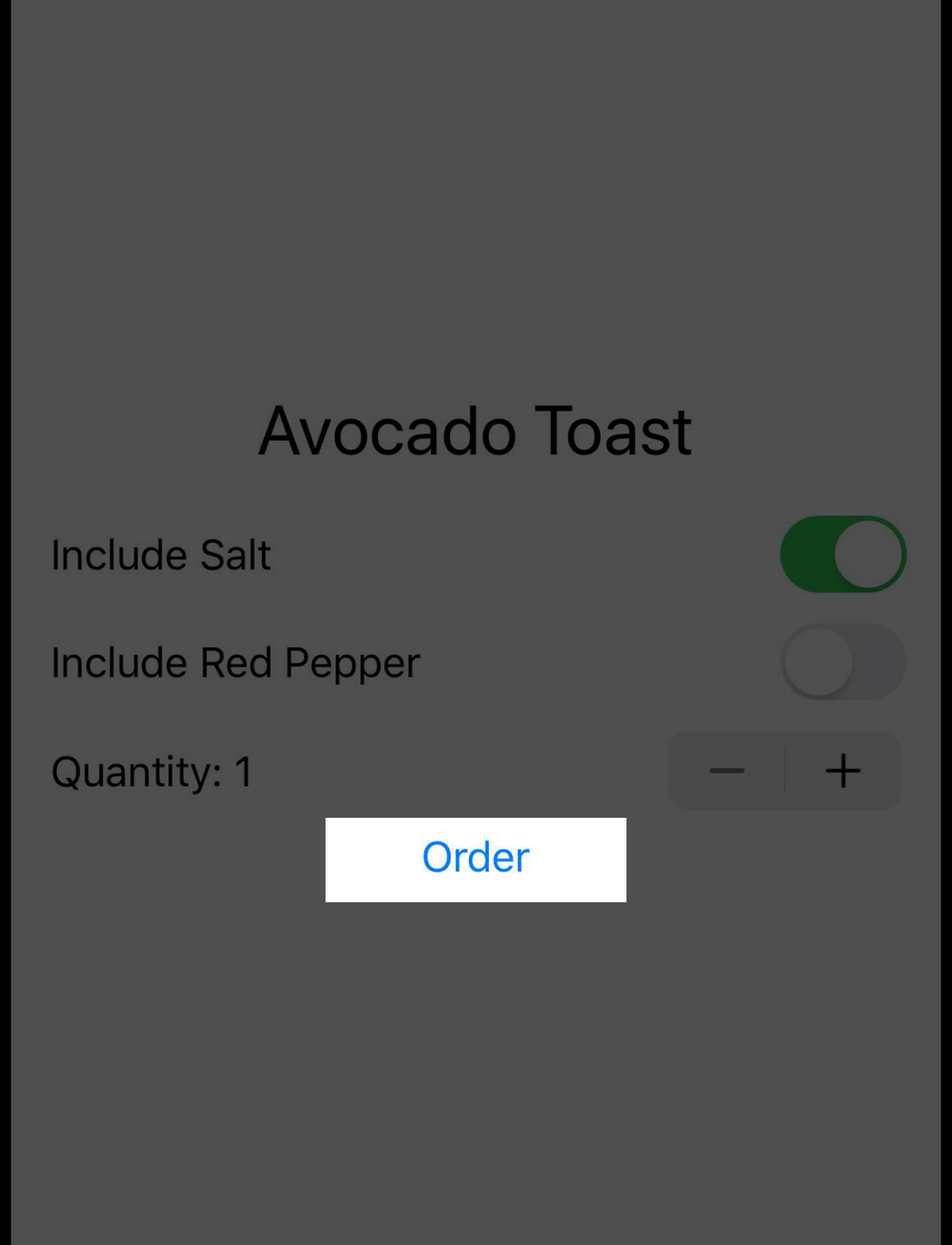
`NSView`

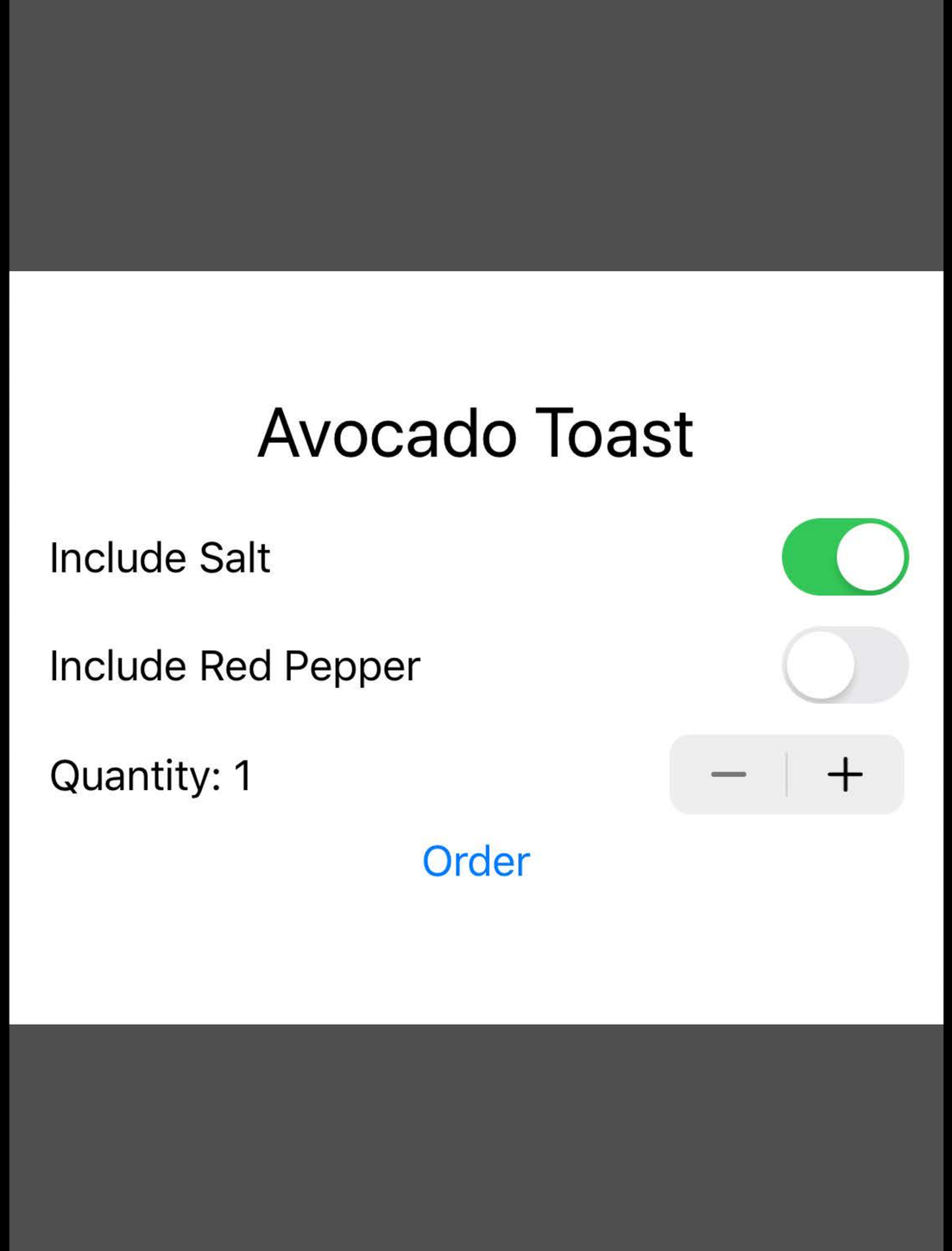
(AppKit)

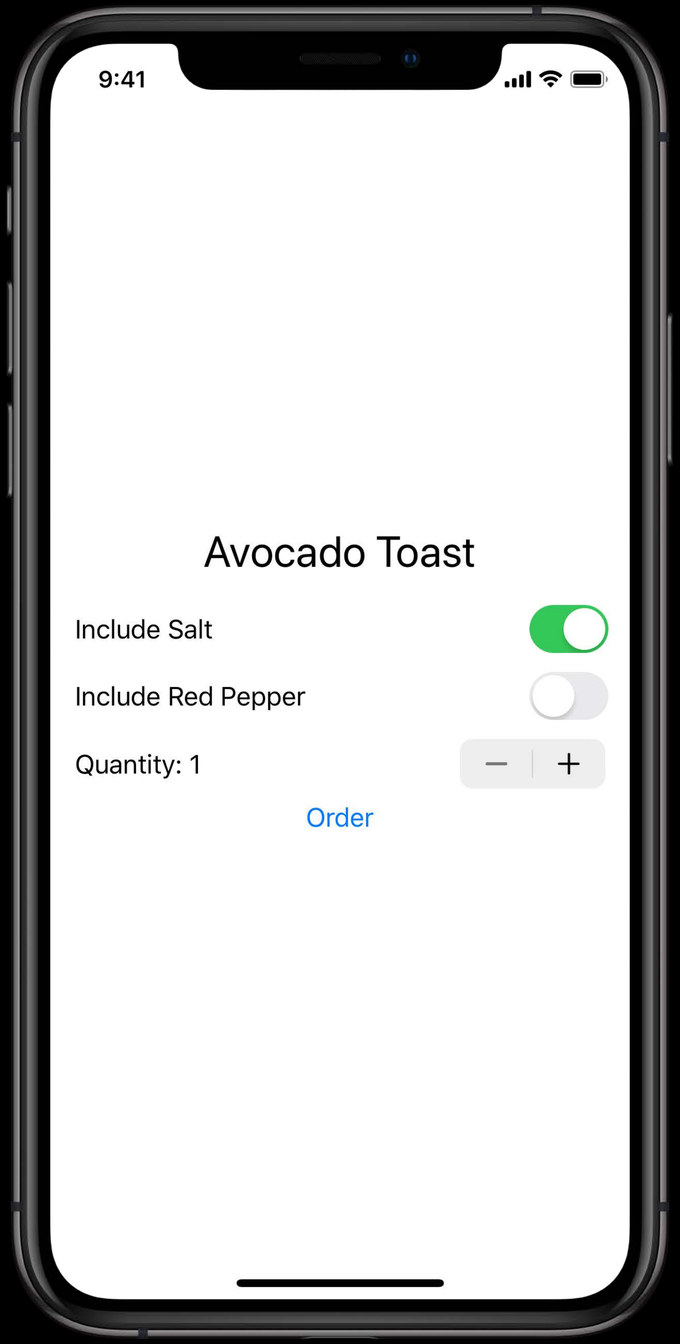
A view defines a piece of UI

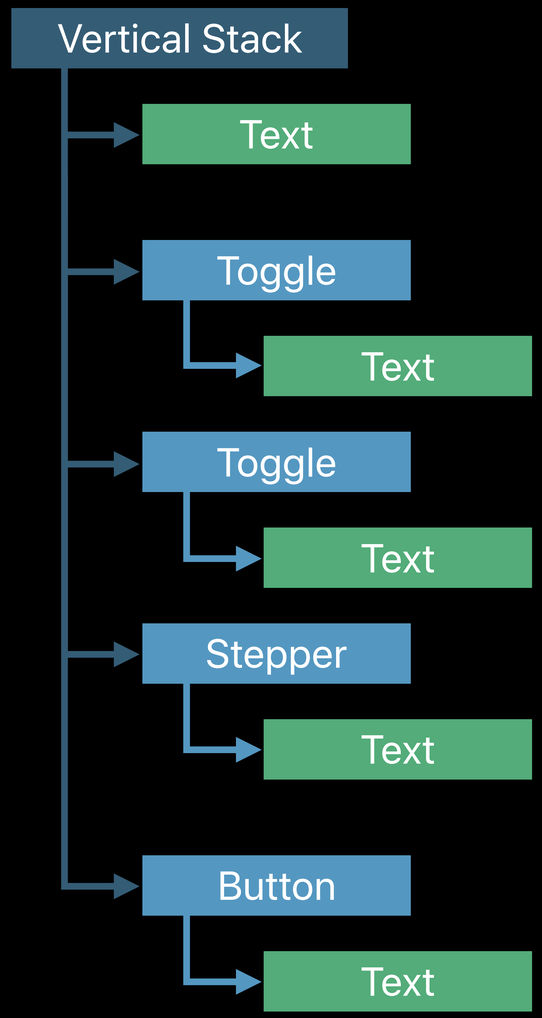

9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order



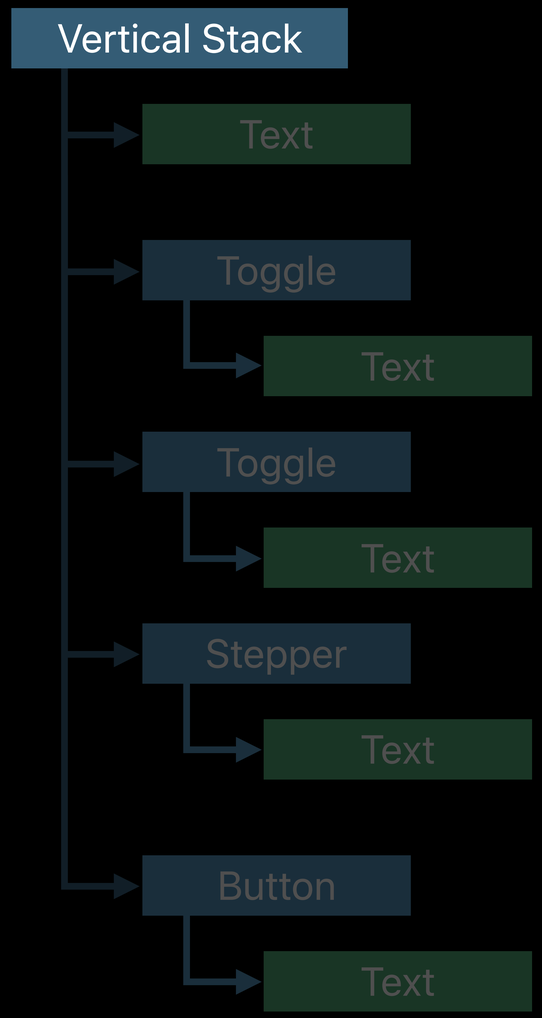
Avocado Toast

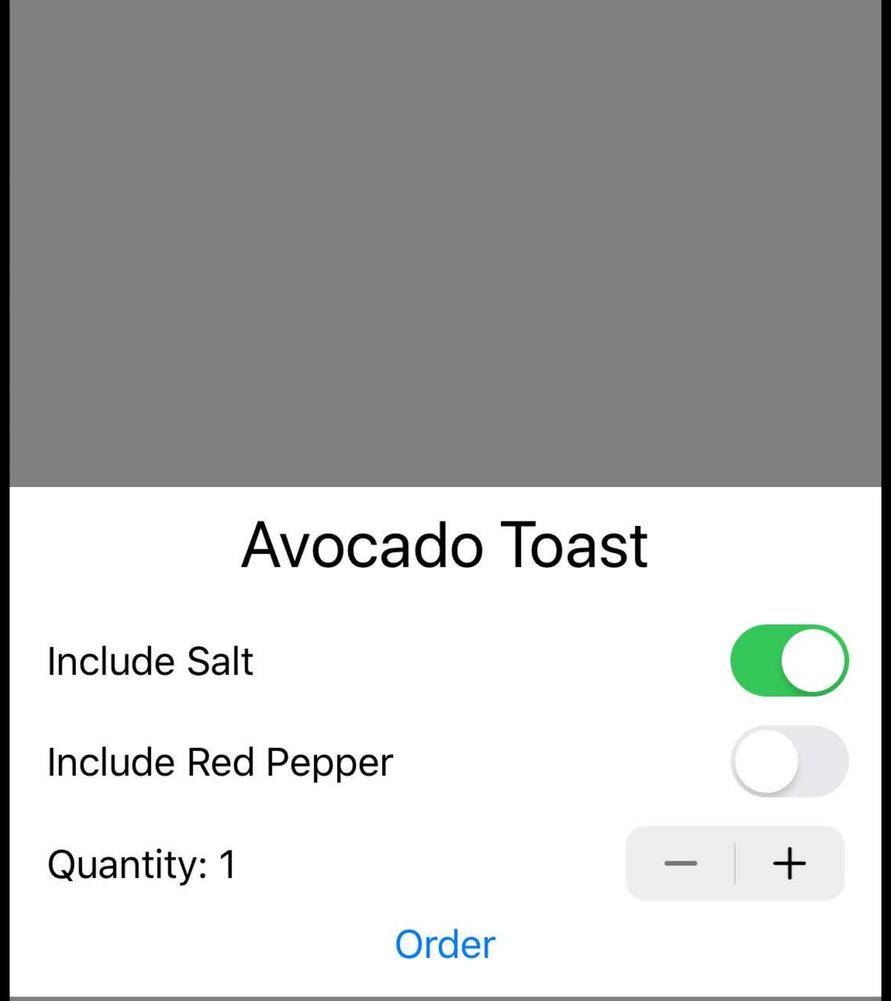
Include Salt

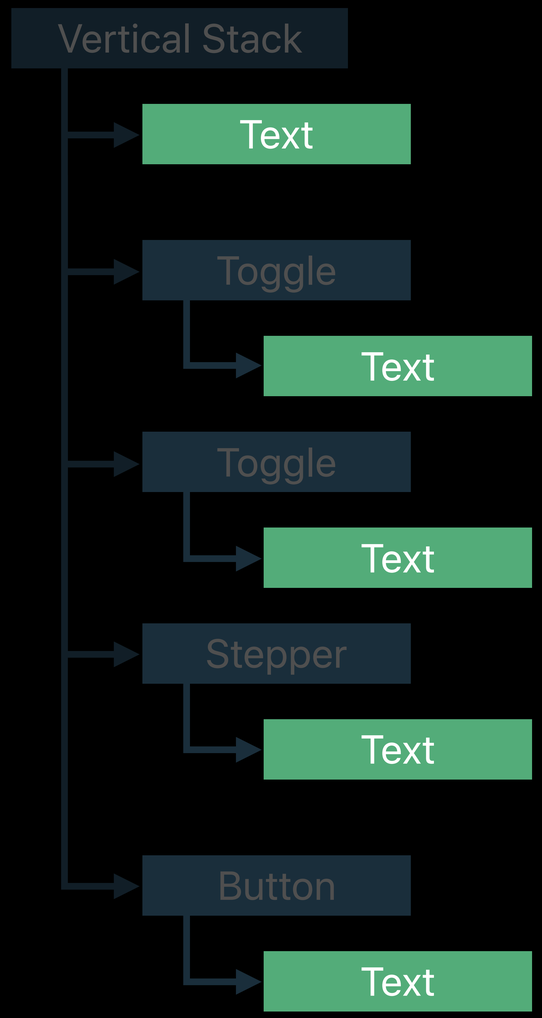
Include Red Pepper

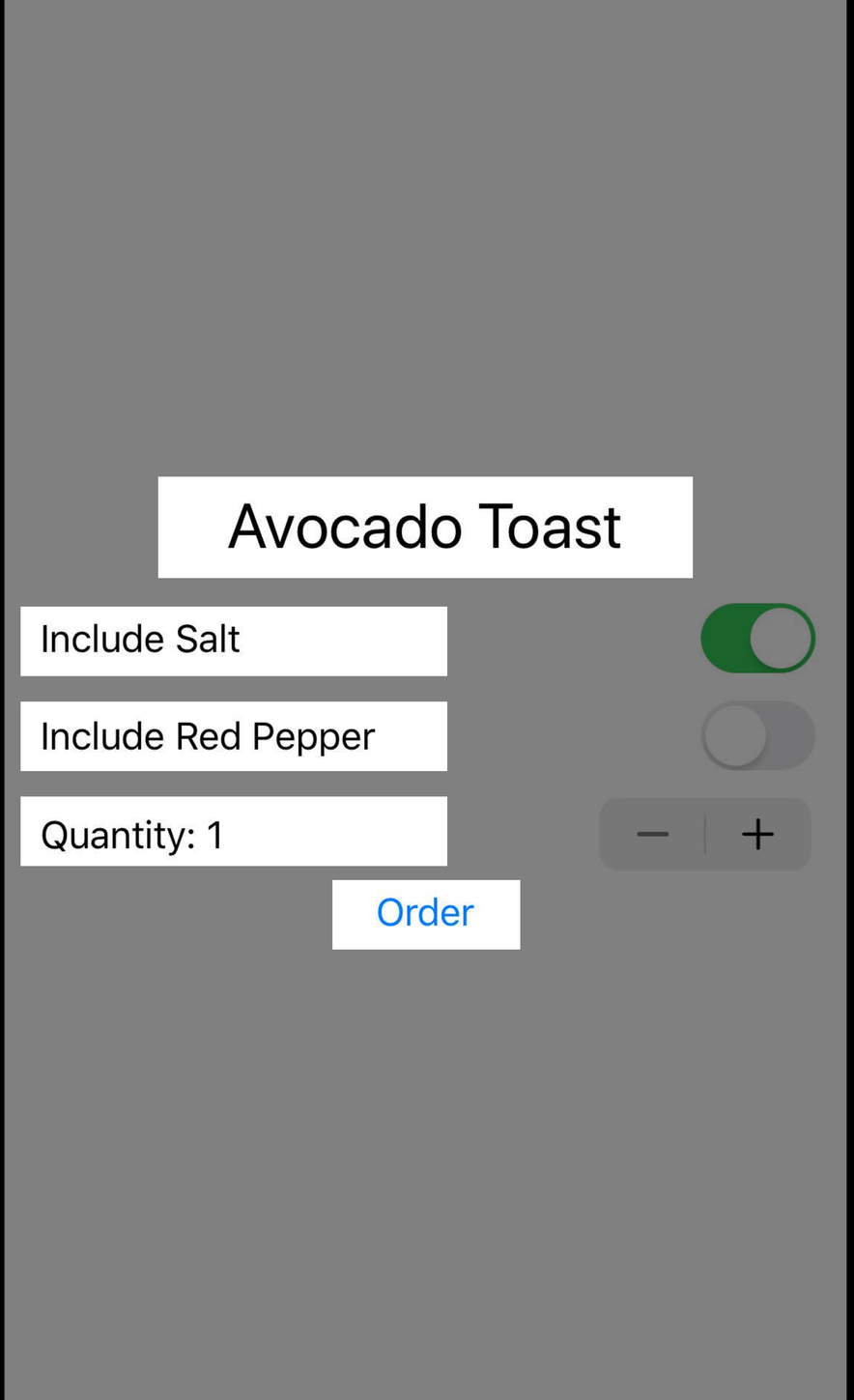
Quantity: 1

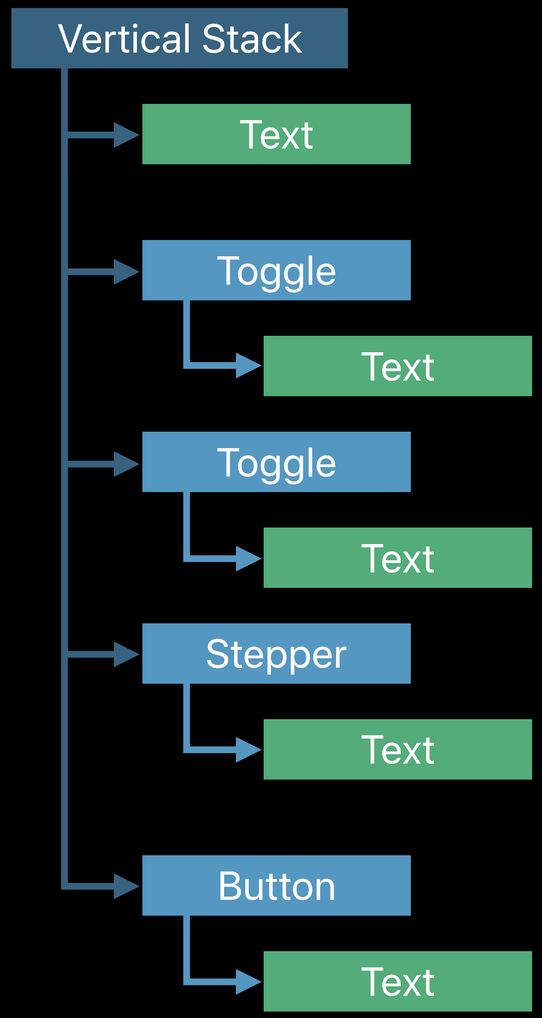
Order


Vertical Stack
Text
Toggle
Text
Toggle
Text
Stepper
Text
Button
Text


Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


Vertical Stack
Text
Toggle
Text
Toggle
Text
Stepper
Text
Button
Text


Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order



Avocado Toast

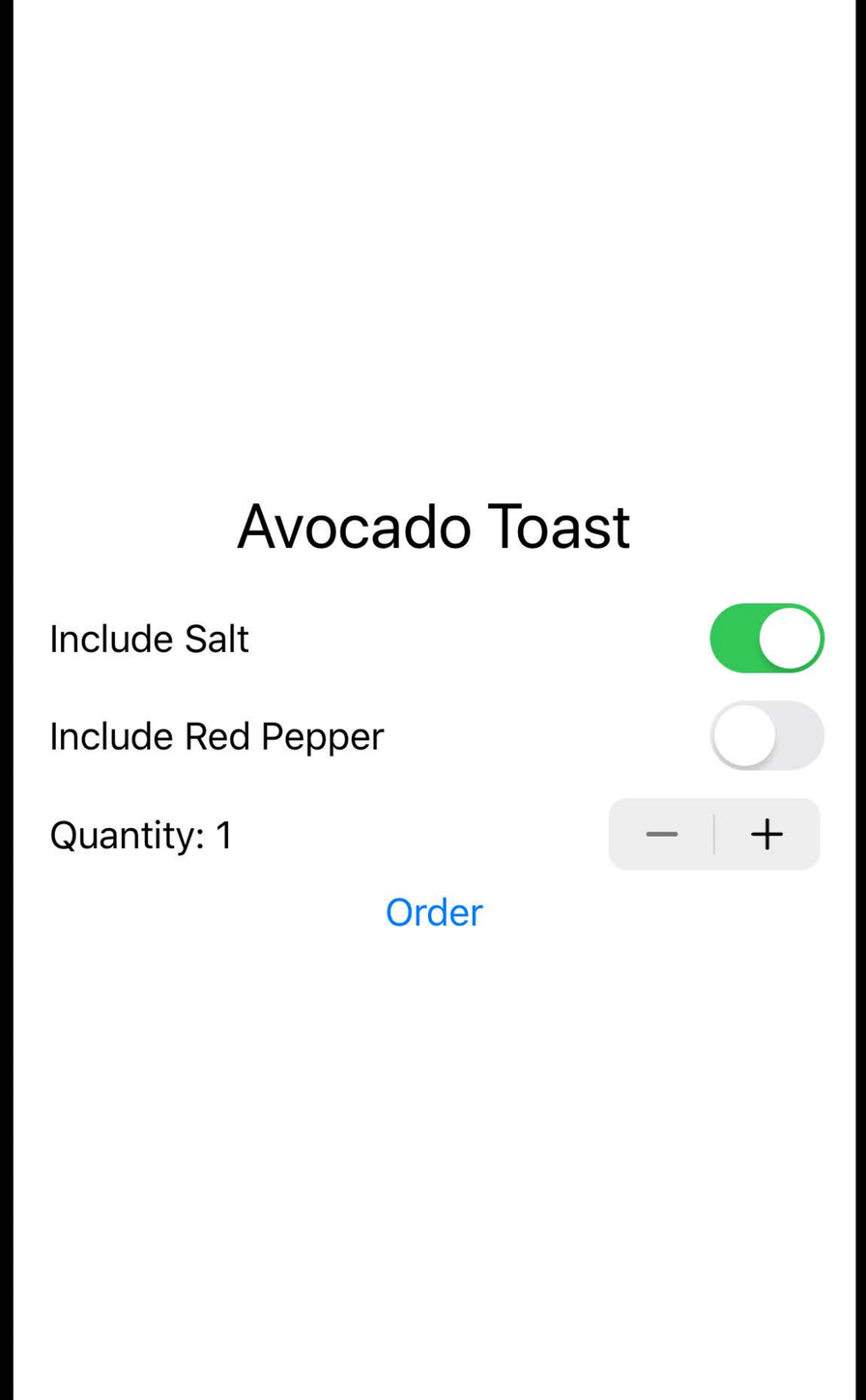
Include Salt

Include Red Pepper

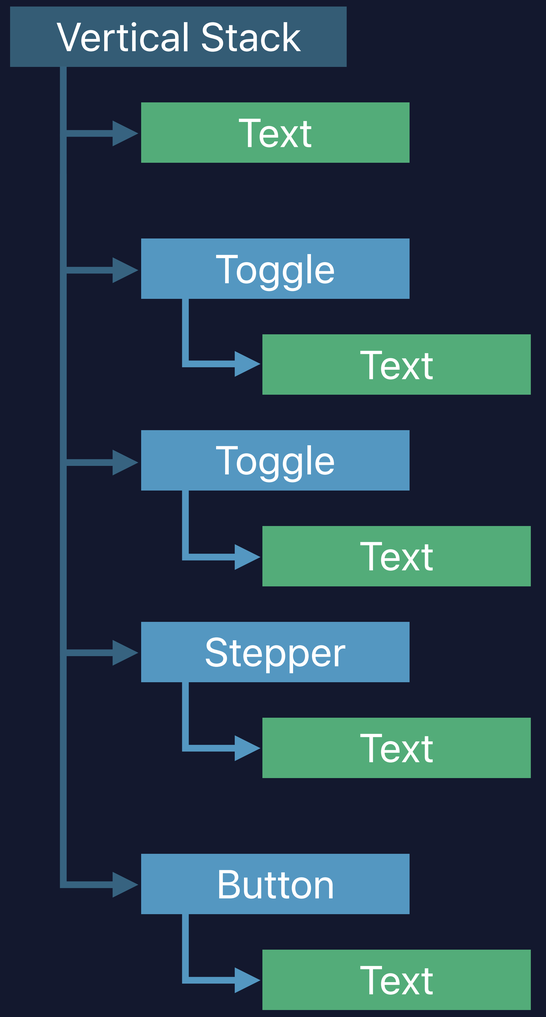
Quantity: 1

Order

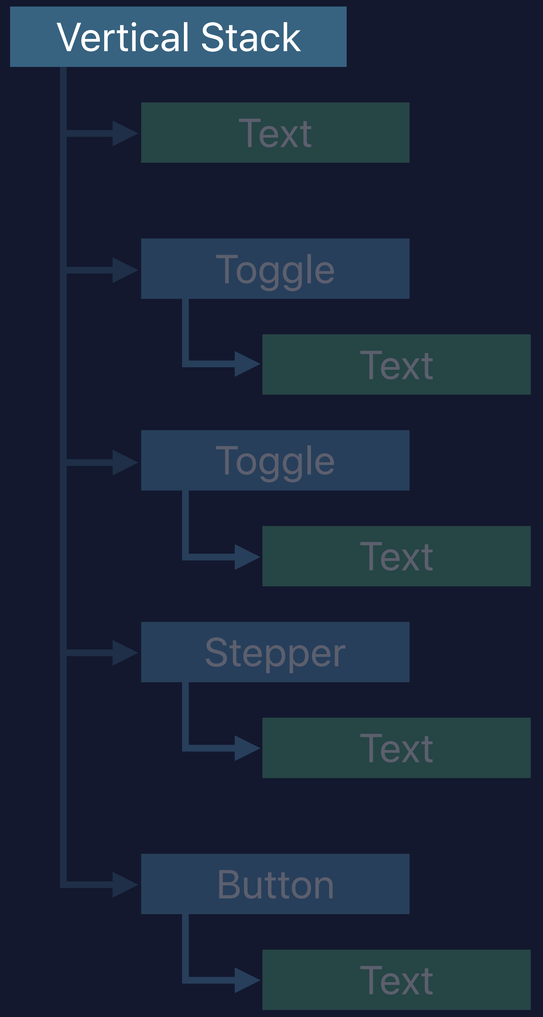
```
VStack {
    Text("Avocado Toast”).font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```



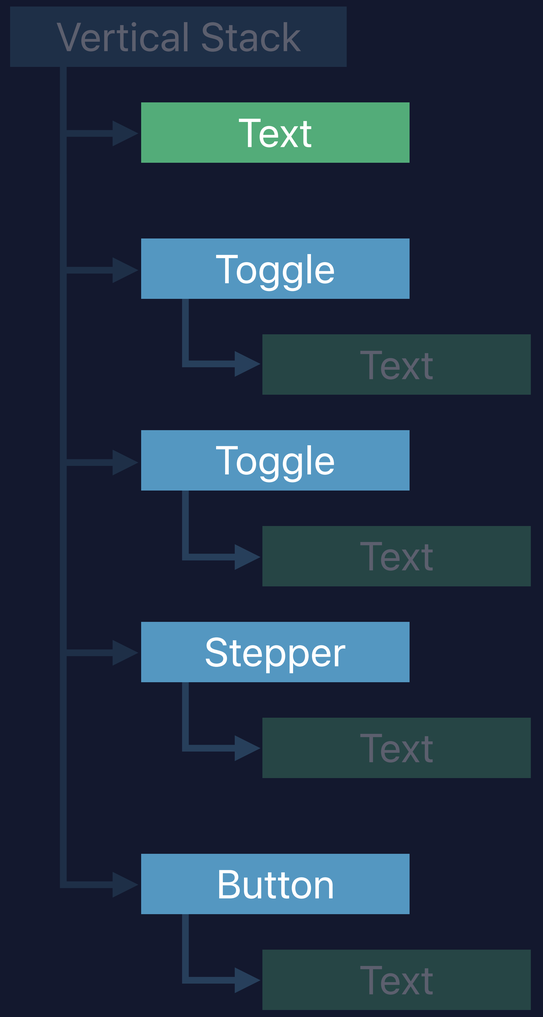
```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```



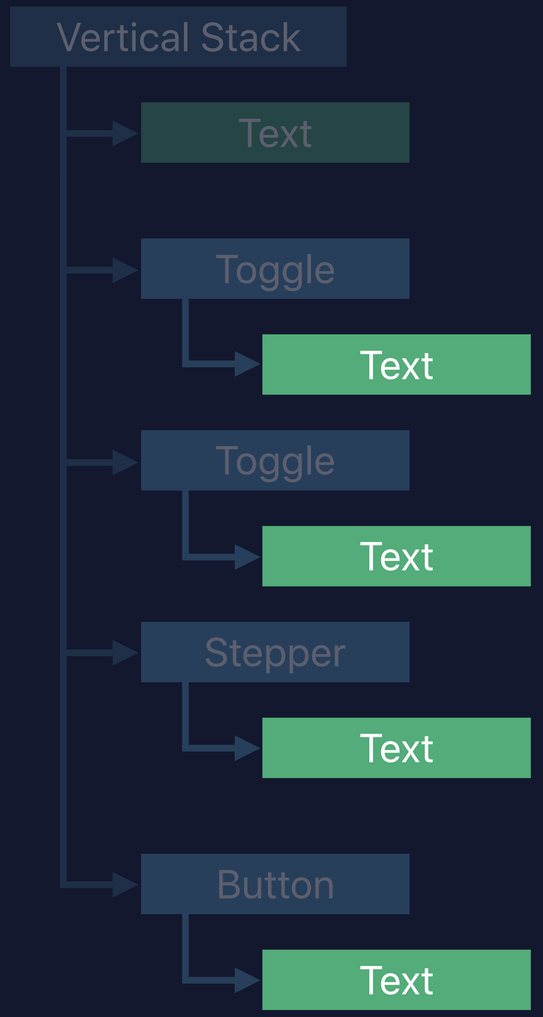
```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```



```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```



```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```



```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```

Imperative Avocado Toast

Imperative Avocado Toast

Imperative Avocado Toast

1. Prepare ingredients: avocado, bread, almond butter, sea salt, red pepper flakes.
2. Gather equipment: toaster, plate, butter knife.

Imperative Avocado Toast

1. Prepare ingredients: avocado, bread, almond butter, sea salt, red pepper flakes.
2. Gather equipment: toaster, plate, butter knife.
3. Lightly toast a slice of bread.
4. Place toast on plate.
5. Spread thin layer of almond butter over toast.
6. Cut avocado in half on its longer axis.
7. Remove avocado core.

4. Place toast on plate.
5. Spread thin layer of almond butter over toast.
6. Cut avocado in half on its longer axis.
7. Remove avocado core.
8. Slice single avocado half into half-centimeter slices.
9. Place avocado slices with 50% overlap over toast.
10. Apply three dashes of sea salt.
11. Apply five dashes of red pepper flakes.
12. Clean toaster and butter knife.
13. Eat avocado toast.
14. Find joy.

Declarative Avocado Toast

Declarative Avocado Toast

Declarative Avocado Toast

“I’d like some avocado toast on charred sourdough with almond butter, sea salt, and red pepper flakes.”

Declarative Avocado Toast

“I’d like some avocado toast on charred sourdough with almond butter, sea salt, and red pepper flakes.”

“Oh, and cut it diagonally.”

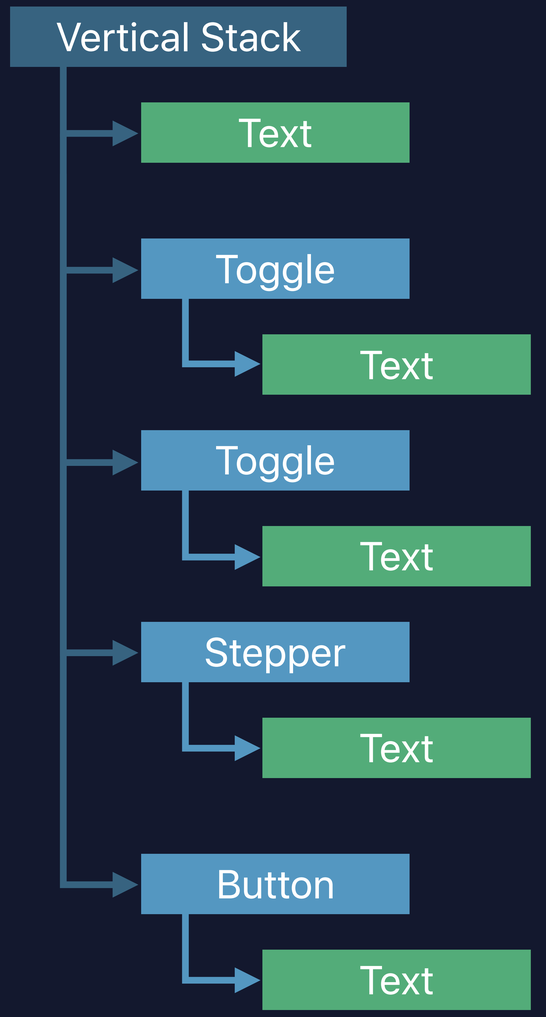
Declarative Avocado Toast

“I’d like some avocado toast on charred sourdough with almond butter, sea salt, and red pepper flakes.”

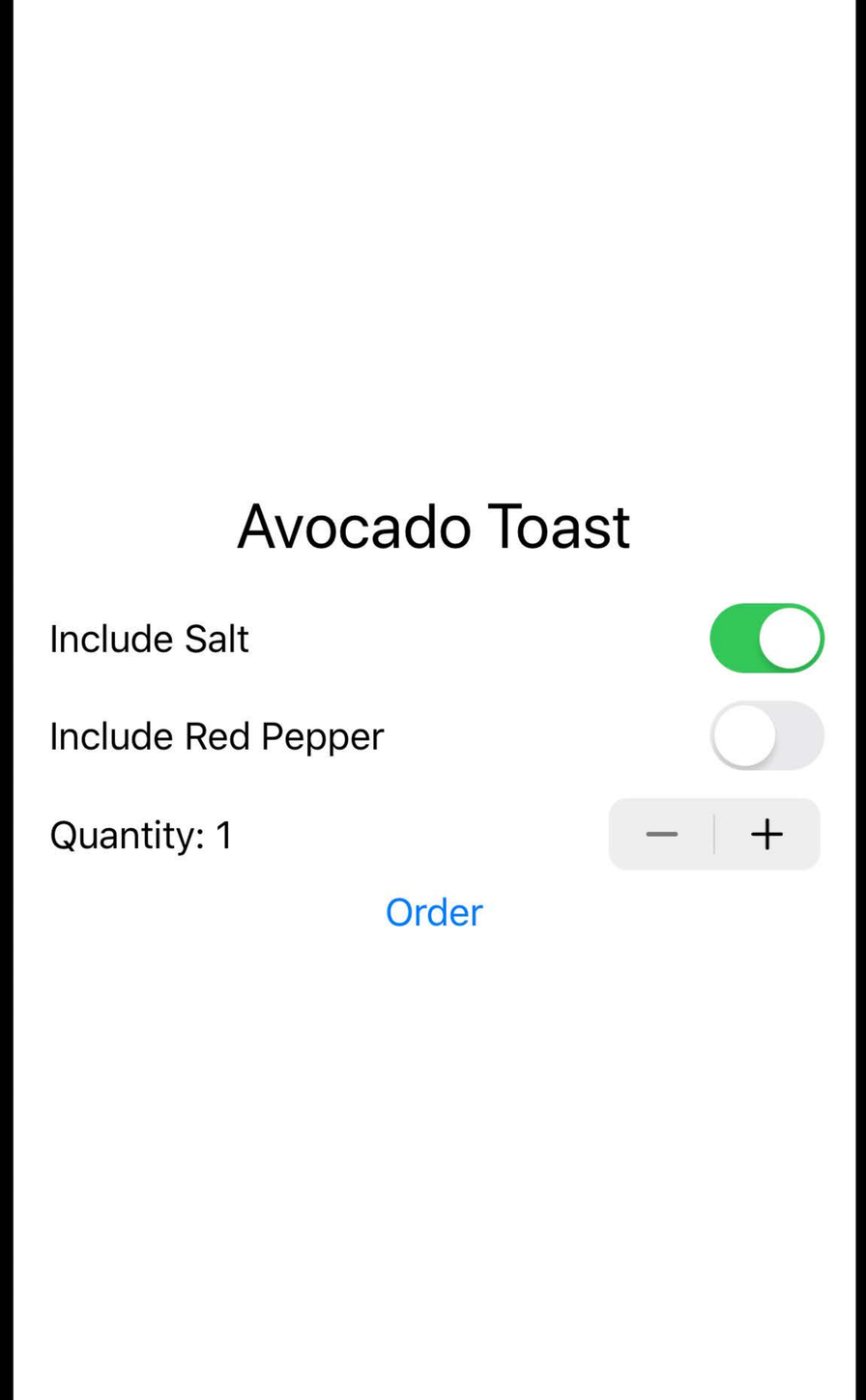
“Oh, and cut it diagonally.”

“Please and thank you!”

```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```



```
VStack {
    Text("Avocado Toast”).font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```



View Container Syntax

```
container {
    content
    content
    •••
}
```

View Container Syntax

```
VStack {
    Image(...)
    Text(...)
    Text(...)
}
```

View Container Syntax

```
VStack {
    Image(...)
    Text(...)
    Text(...)
}
```

```
public struct VStack<Content : View> : View {
    public init(
        alignment: HorizontalAlignment = .center,
        spacing: Length? = nil,
        @ViewBuilder content: () -> Content
    )
}
```

View Container Syntax

```
VStack(alignment: .leading) {
    Image(...)
    Text(...)
    Text(...)
}
```

```
public struct VStack<Content : View> : View {
    public init(
        alignment: HorizontalAlignment = .center,
        spacing: Length? = nil,
        @ViewBuilder content: () -> Content
    )
}
```

API

View Container Syntax

```
VStack(alignment: .leading) {
    Image(...)
    Text(...)
    Text(...)
}
```

```
public struct VStack<Content : View> : View {
    public init(
        alignment: HorizontalAlignment = .center,
        spacing: Length? = nil,
        @ViewBuilder content: () -> Content
    )
}
```

API

```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```

```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```

```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```

```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```

Binding Syntax

```
struct OrderForm : View {
    @State private var order: Order

    var body: some View {
        Stepper(value: $order.quantity, in: 1...10) {
            Text("Quantity: \(order.quantity)")
        }
    }
}
```

Binding Syntax

```
struct OrderForm : View {
    @State private var order: Order

    var body: some View {
        Stepper(value: $order.quantity, in: 1...10) {
            Text("Quantity: \(order.quantity)")
        }
    }
}
```

```
struct Order {
    var includeSalt: Bool
    var includeRedPepperFlakes: Bool
    var quantity: Int
}
```

Binding Syntax

```
struct OrderForm : View {
   @State private var order: Order

   var body: some View {
      Stepper(value: $order.quantity, in: 1...10) {
         Text("Quantity: \(order.quantity)")
      }
   }
}
```

```
struct Order {
   var includeSalt: Bool
   var includeRedPepperFlakes: Bool
   var quantity: Int
}
```

Binding Syntax

```
struct OrderForm : View {
   @State private var order: Order

   var body: some View {
      Stepper(value: $order.quantity, in: 1...10) {
         Text("Quantity: \(order.quantity)")
      }
   }
}
```

```
struct Order {
   var includeSalt: Bool
   var includeRedPepperFlakes: Bool
   var quantity: Int
}
```

Binding Syntax

```
struct OrderForm : View {
   @State private var order: Order

   var body: some View {
      Stepper(value: $order.quantity, in: 1...10) {
         Text("Quantity: \(order.quantity)")
      }
   }
}
```

Data Flow Through SwiftUI Thursday, 9:00

Binding Syntax

```
struct OrderForm : View {
    @State private var order: Order

    var body: some View {
        Stepper(value: $order.quantity, in: 1...10) {
            Text("Quantity: \(order.quantity)")
        }
    }
}
```

Data Flow Through SwiftUI Thursday, 9:00

Binding Syntax

```
struct OrderForm : View {
   @State private var order: Order

   var body: some View {
      Stepper(value: $order.quantity, in: 1...10) {
         Text("Quantity: \(order.quantity)")
      }
   }
}
```

```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```

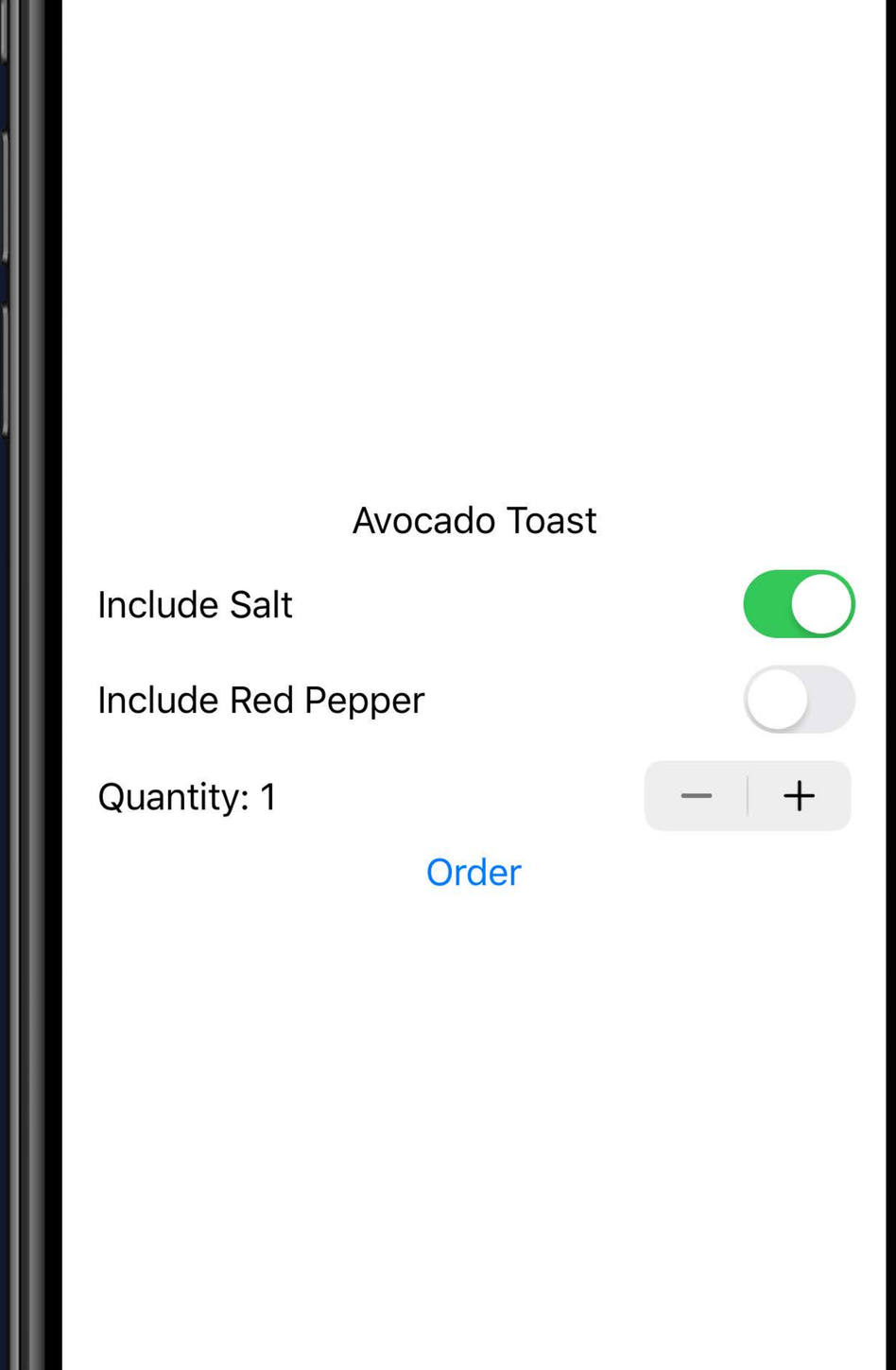
```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```

```
Text("Avocado Toast").font(.title)
```

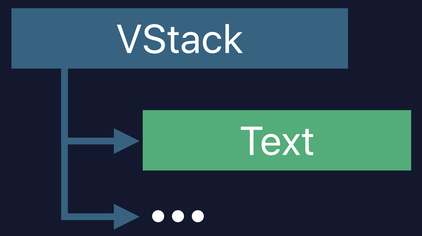
```
Text("Avocado Toast").font(.title)
```

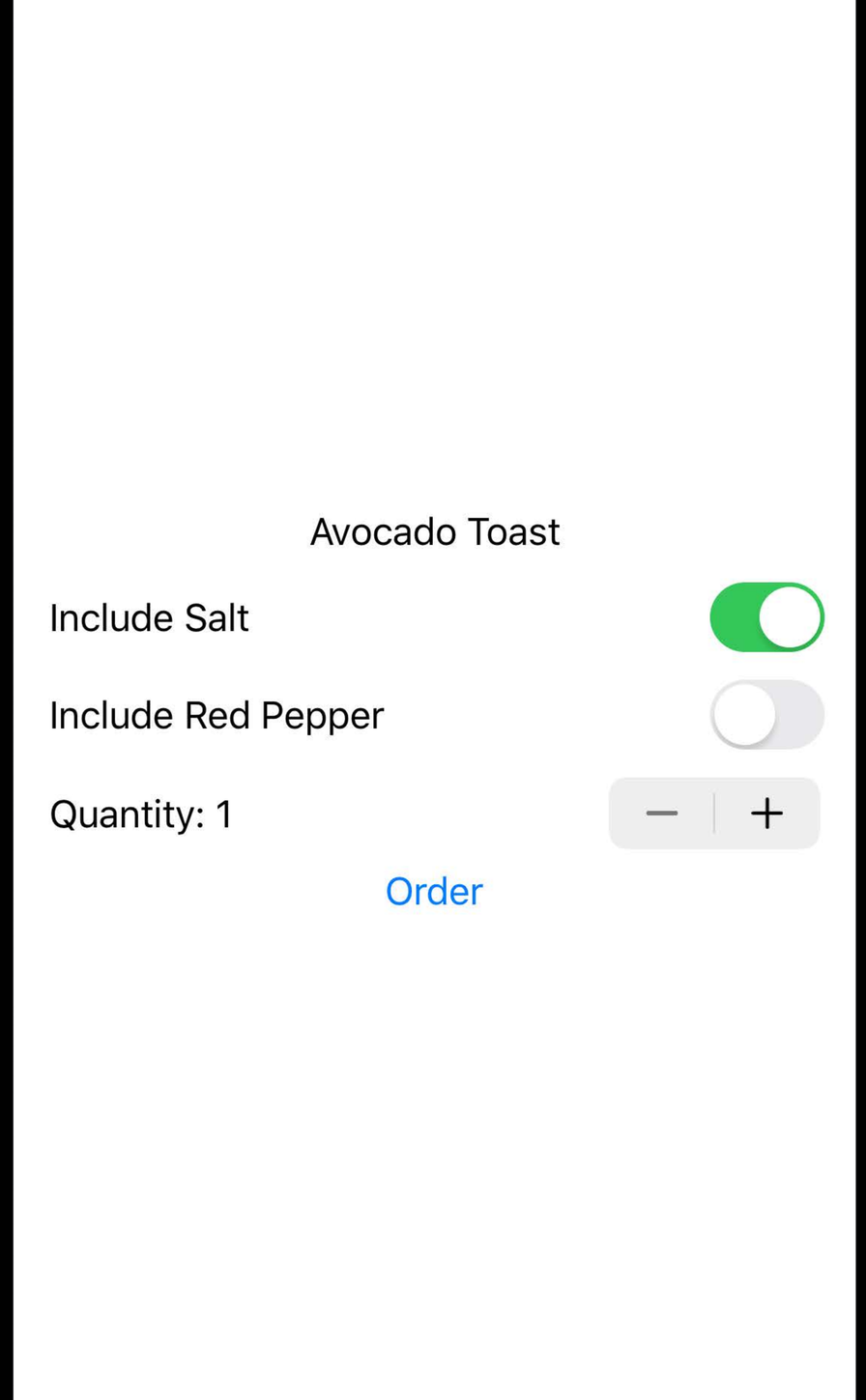
```
Text("Avocado Toast").font(.title)
```

```
VStack {
    Text("Avocado Toast")


    ...
}
```

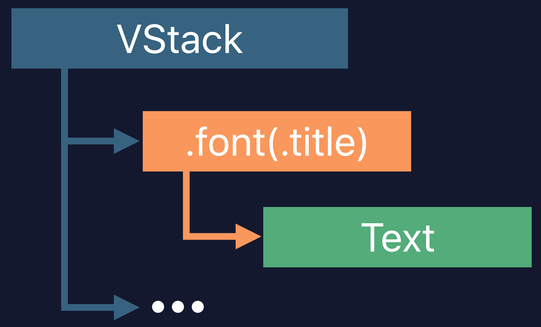


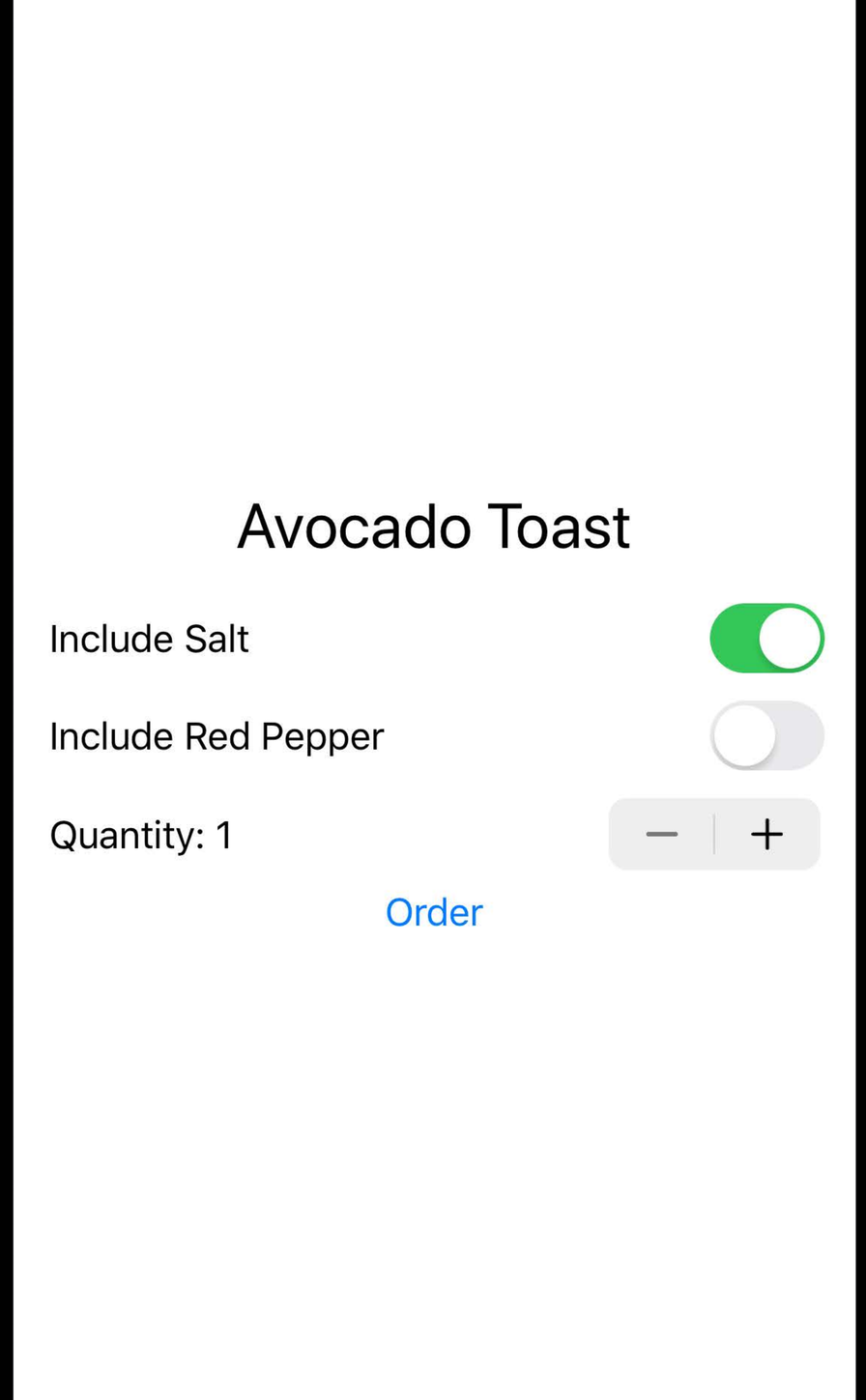
```
VStack {
    Text("Avocado Toast")


    ...
}
```



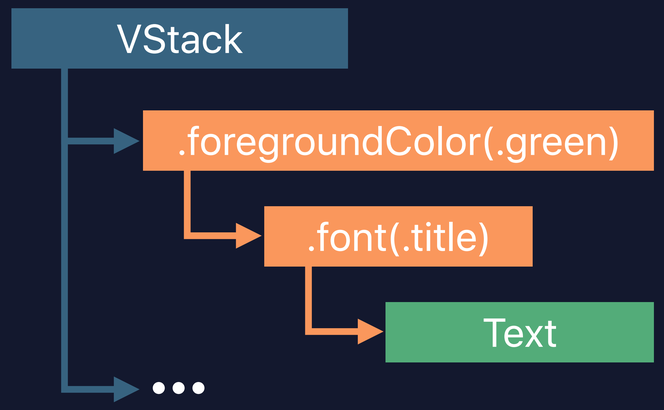


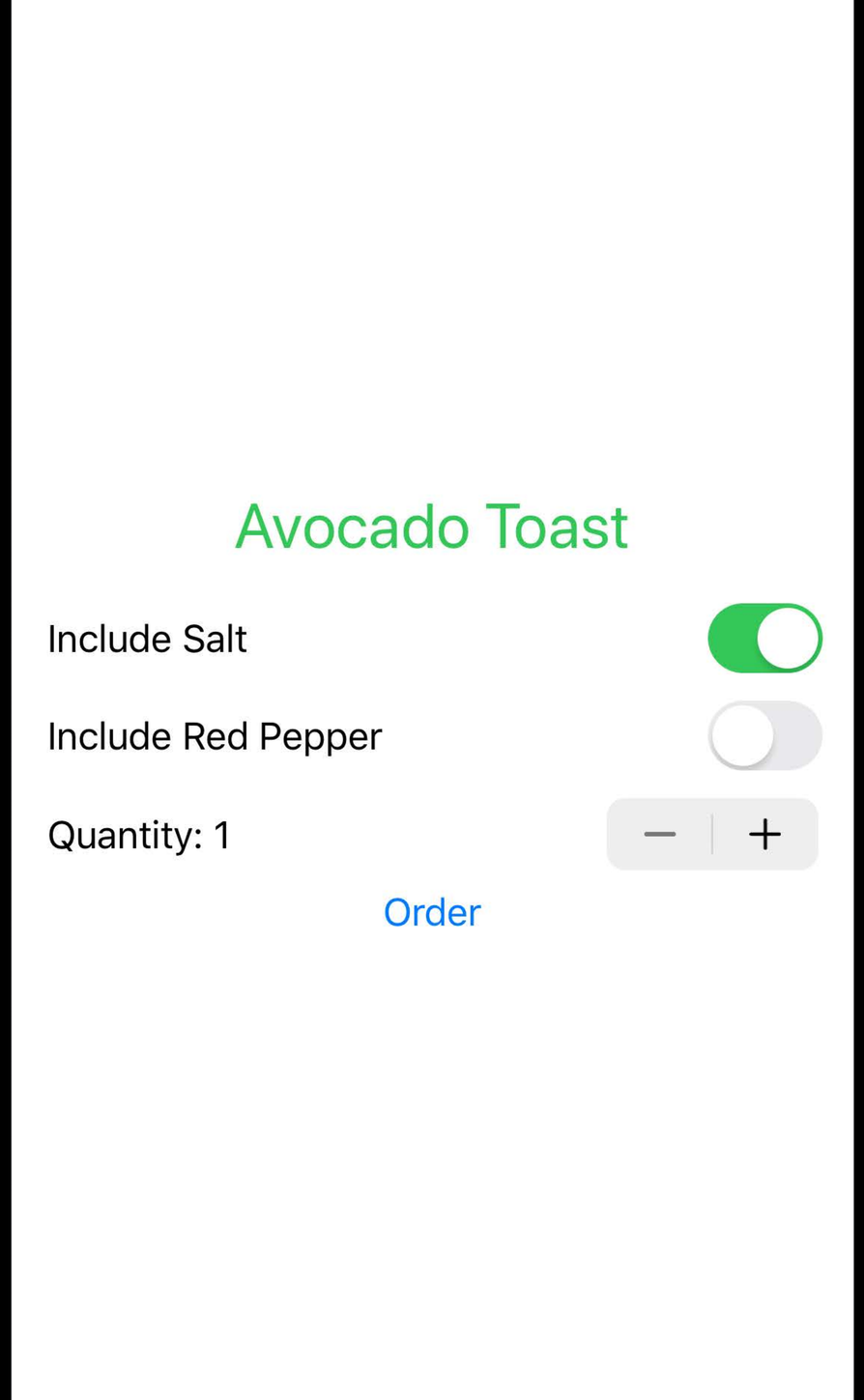
```
VStack {
    Text("Avocado Toast")
        .font(.title)

    ...
}
```



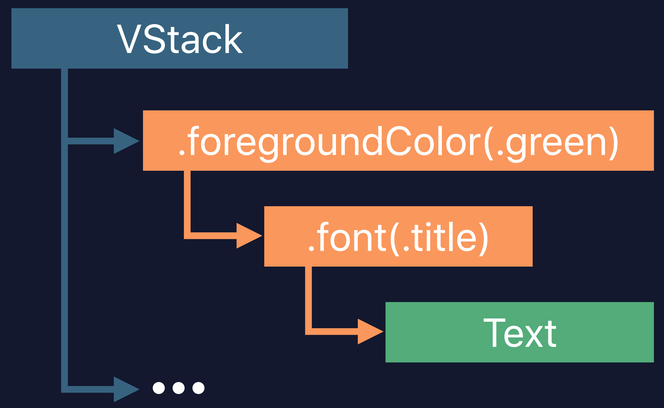


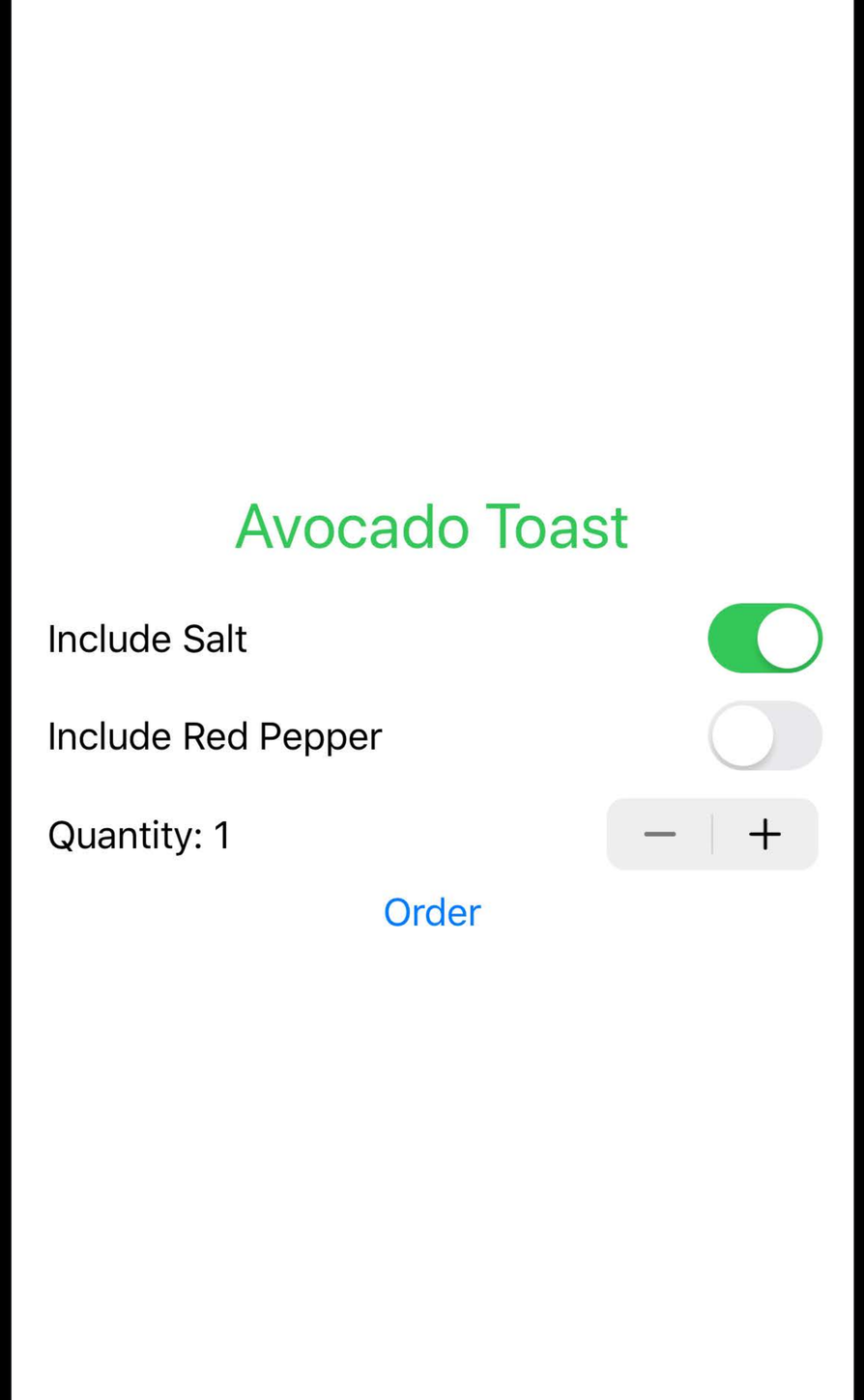
```
VStack {
    Text("Avocado Toast")
        .font(.title)
        .foregroundColor(.green)
    ...
}
```





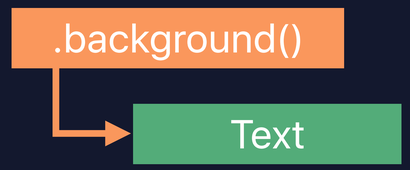
```
VStack {
    Text("Avocado Toast")
        .font(.title)
        .foregroundColor(.green)
    ...
}
```



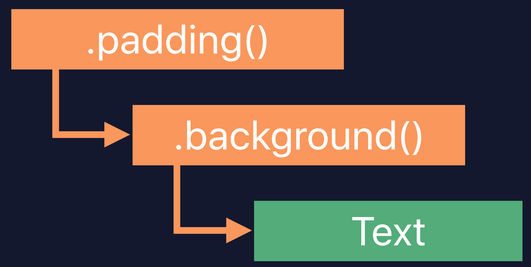



.font(.title)
Text
VStack

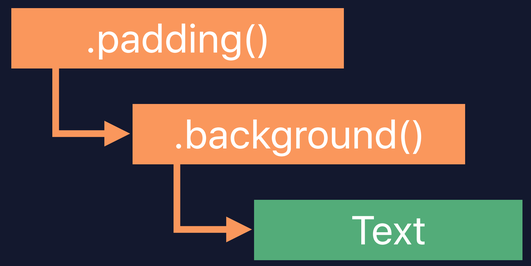
```
Text("🥑🍞")
    .background(Color.green, cornerRadius: 12)
```

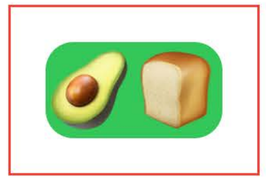


```
Text("🥑🍞")
    .background(Color.green, cornerRadius: 12)
    .padding(.all)
```

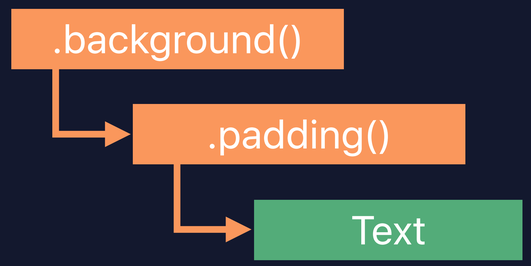


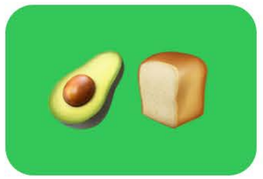
```
Text("🥑🍞")
    .background(Color.green, cornerRadius: 12)
    .padding(.all)
```



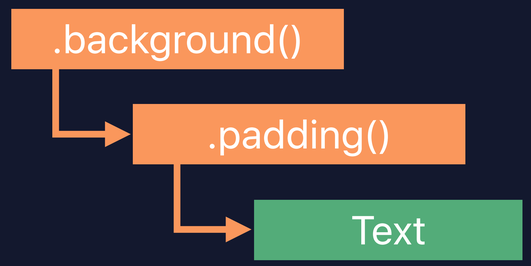


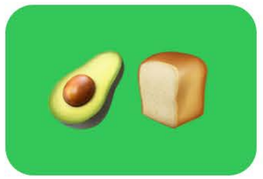
```
Text("🥑🍞")
    .padding(.all)
    .background(Color.green, cornerRadius: 12)
```





```
Text("🥑🍞")
    .padding(.all)
    .background(Color.green, cornerRadius: 12)
```





```
// Sharing Modifiers

VStack(alignment: .leading) {
    Toggle(isOn: $order.includeSalt) { ... }
        .opacity(0.5)

    Stepper(value: $order.quantity, in: 1...10) { ... }
        .opacity(0.5)

    Button(action: submitOrder) { ... }
        .opacity(0.5)
}
```

```
// Sharing Modifiers

VStack(alignment: .leading) {
    Toggle(isOn: $order.includeSalt) { ... }

    Stepper(value: $order.quantity, in: 1...10) { ... }

    Button(action: submitOrder) { ... }
}
.opacity(0.5)
```

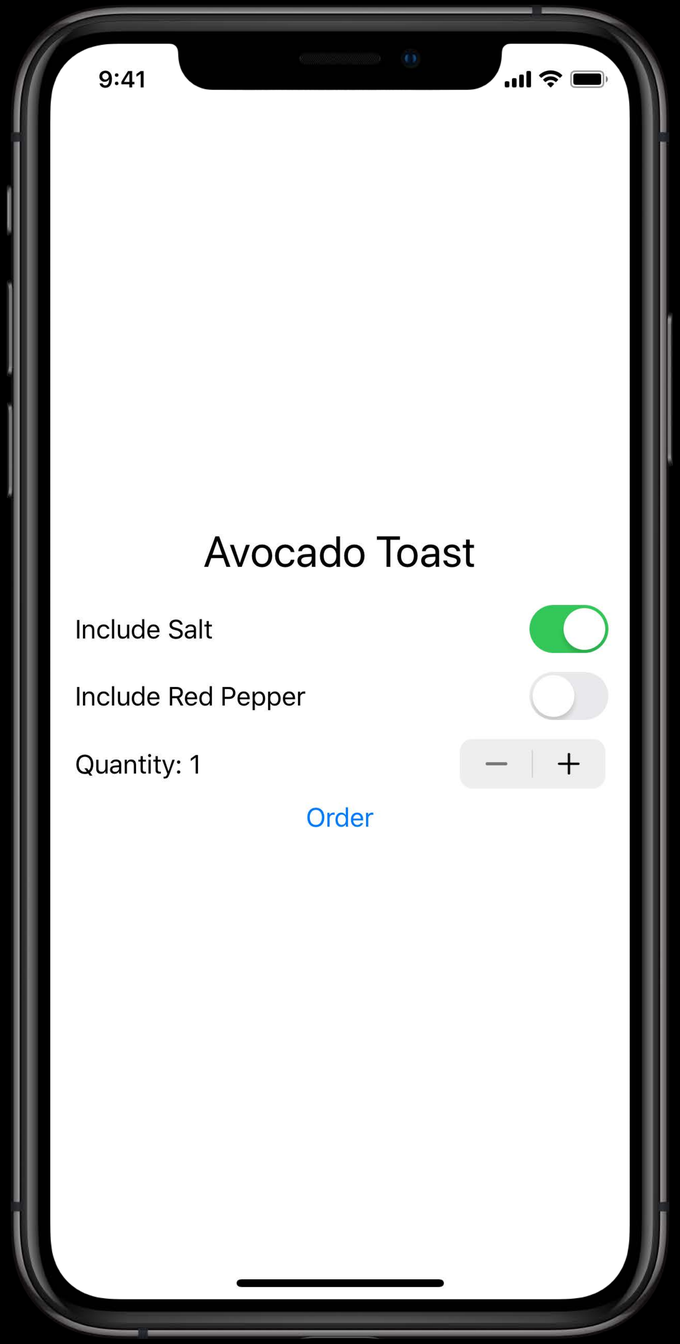
Prefer smaller, single-purpose views

Build larger views using composition

```
Text("Avocado Toast")
```

```
Text("Avocado Toast").font(.title)
```

```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) {
        Text("Include Salt")
    }
    Toggle(isOn: $order.includeRedPepperFlakes) {
        Text("Include Red Pepper Flakes")
    }
    Stepper(value: $order.quantity, in: 1...10) {
        Text("Quantity: \(order.quantity)")
    }

    Button(action: submitOrder) {
        Text("Order")
    }
}
```

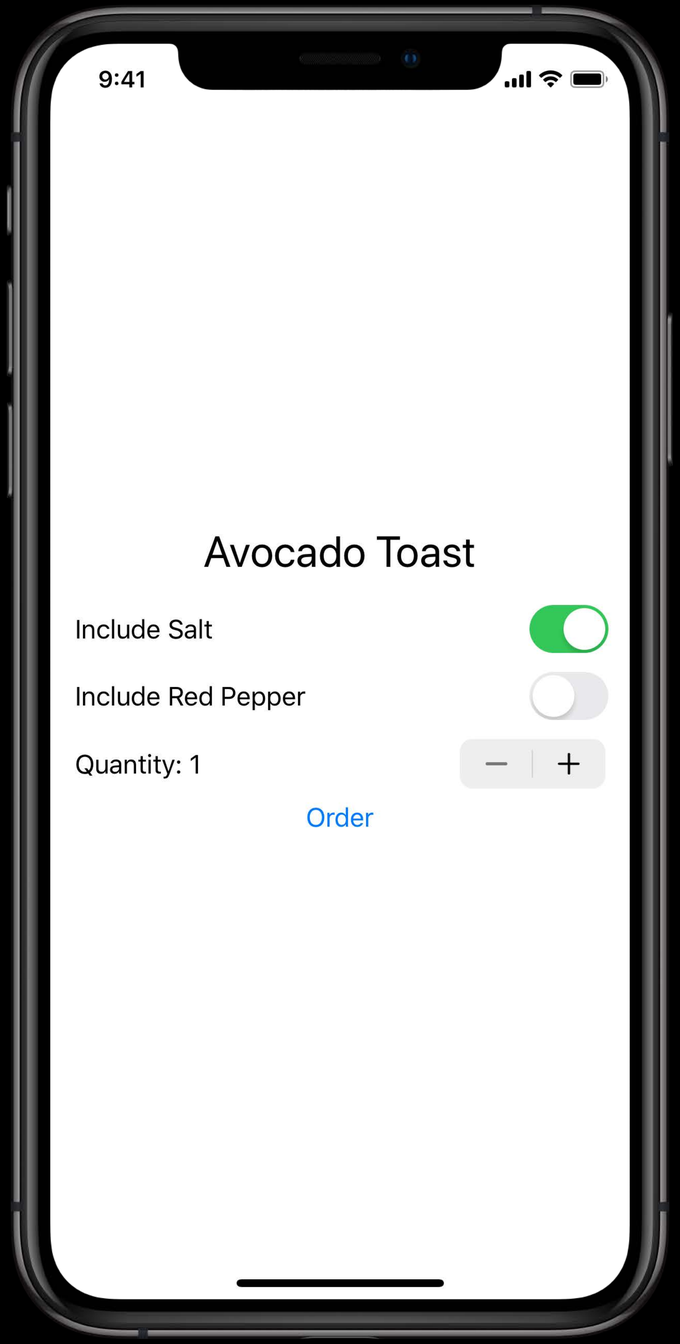

9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order

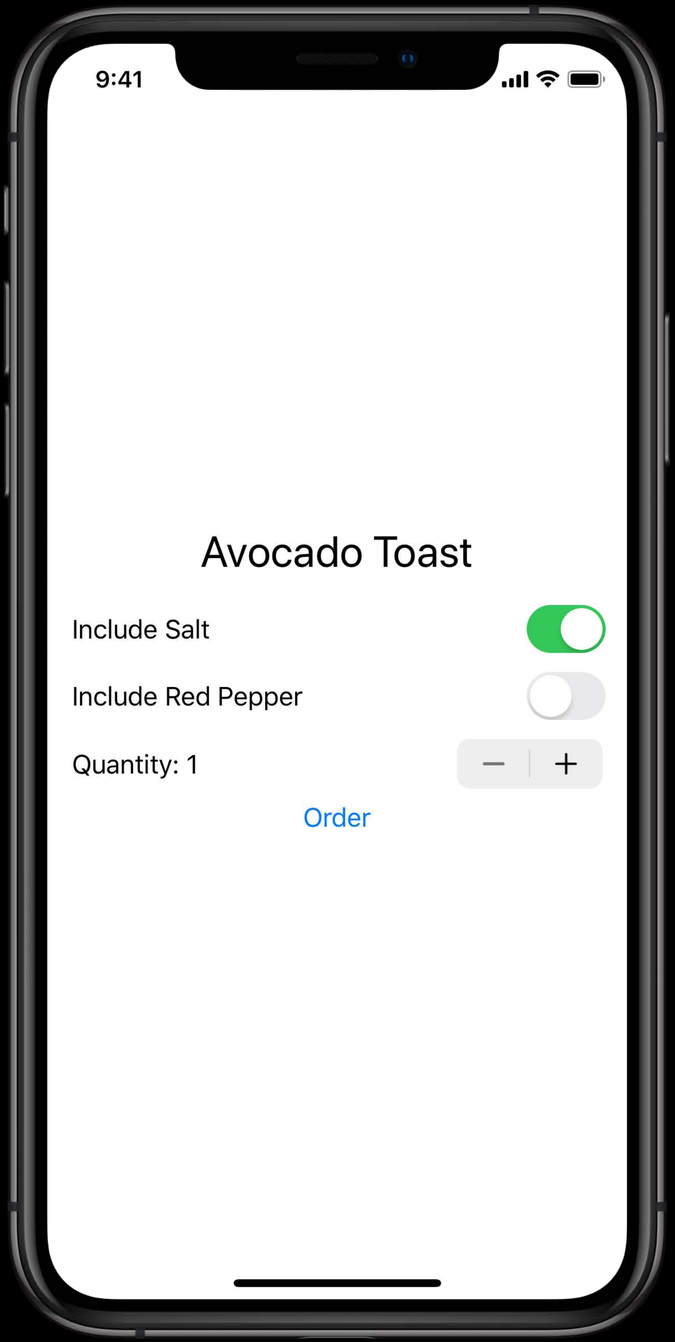
Getting started: Views and modifiers

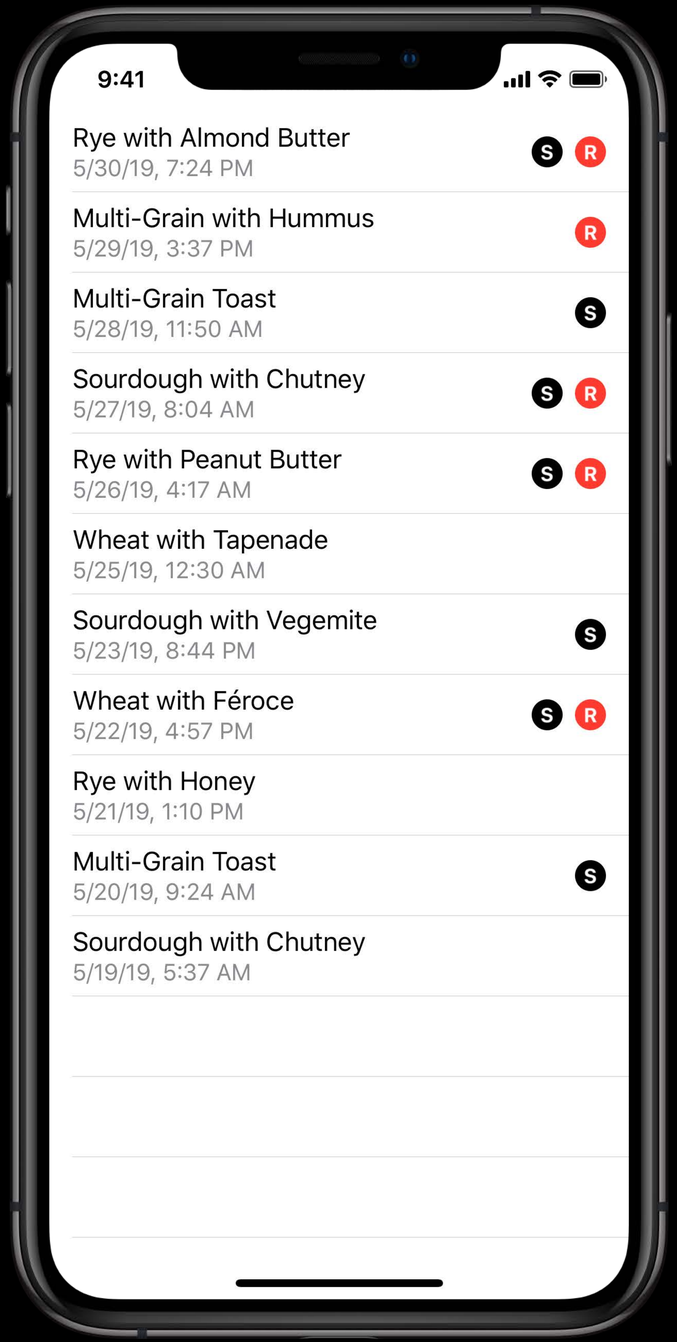
Building custom views

Composing controls

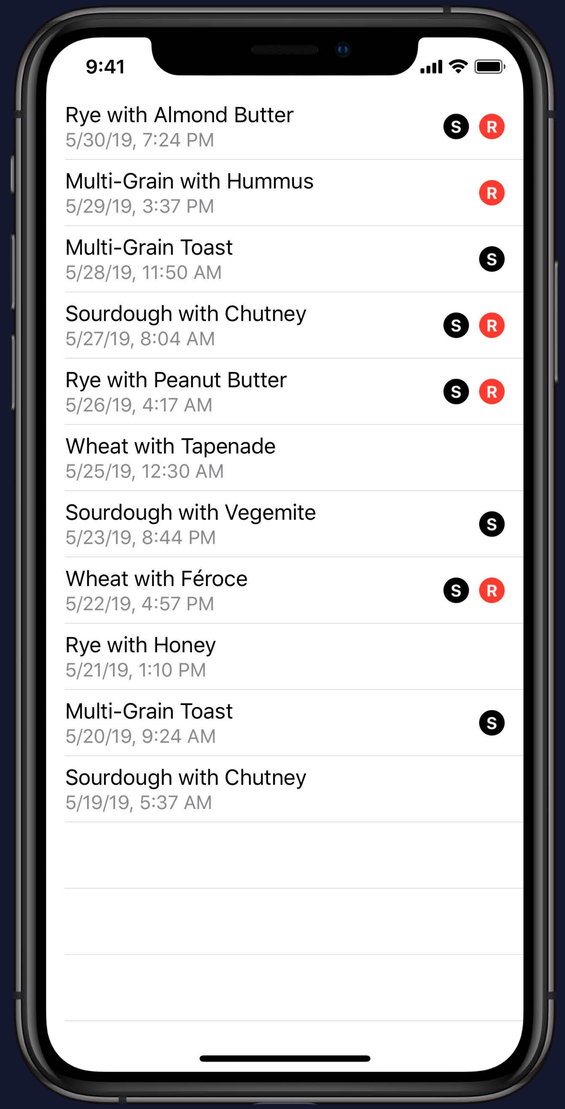
Navigating your app


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order

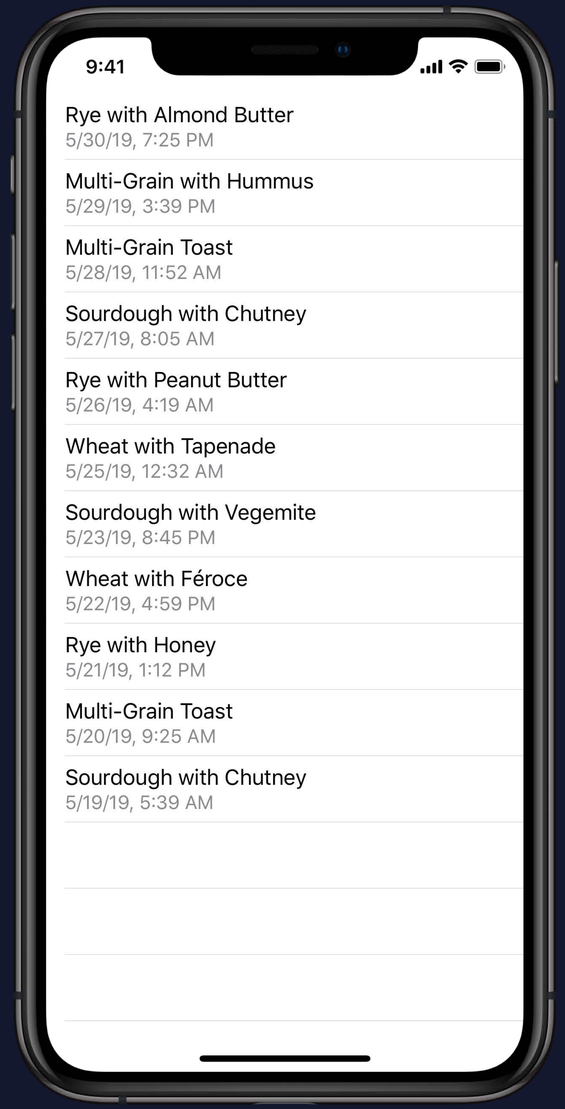

9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Rye with Almond Butter
5/30/19, 7:24 PM
Multi-Grain with Hummus
5/29/19, 3:37 PM
Multi-Grain Toast
5/28/19, 11:50 AM
Sourdough with Chutney
5/27/19, 8:04 AM
Rye with Peanut Butter
5/26/19, 4:17 AM
Wheat with Tapenade
5/25/19, 12:30 AM
Sourdough with Vegemite
5/23/19, 8:44 PM
Wheat with Féroce
5/22/19, 4:57 PM
Rye with Honey
5/21/19, 1:10 PM
Multi-Grain Toast
5/20/19, 9:24 AM
Sourdough with Chutney
5/19/19, 5:37 AM

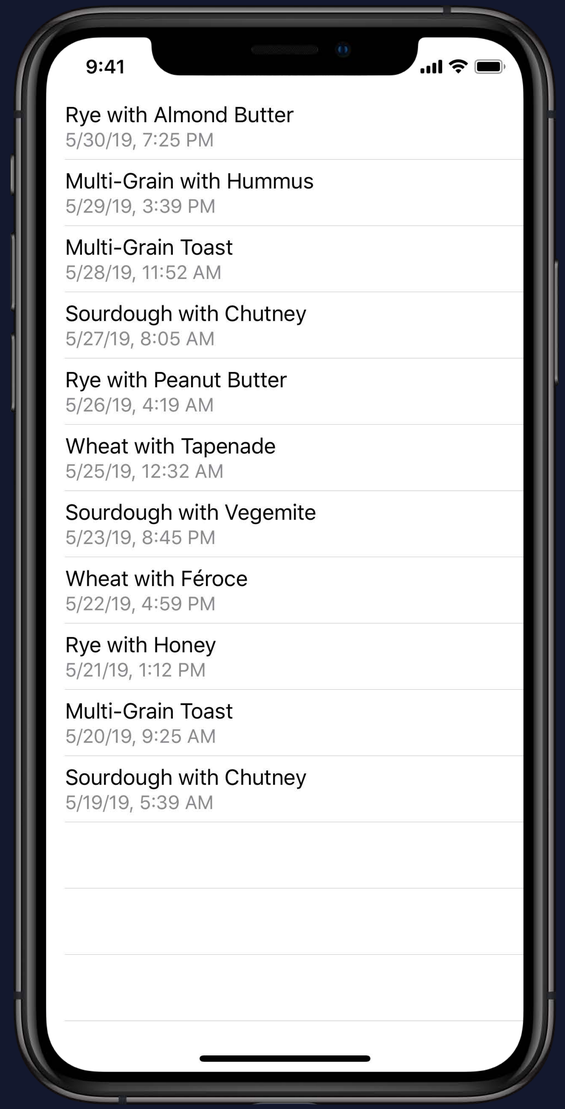
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



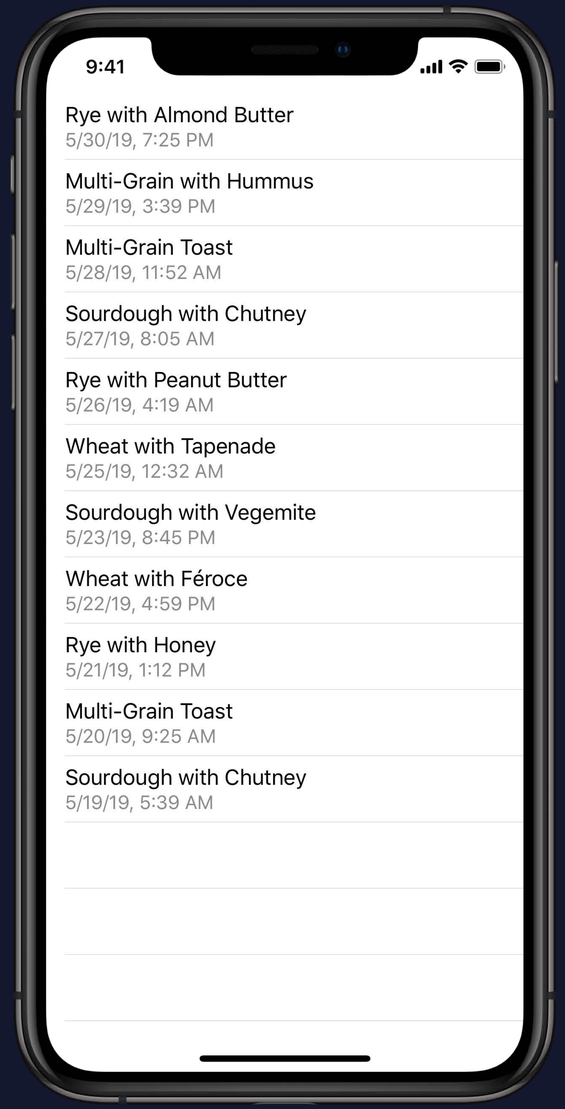
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



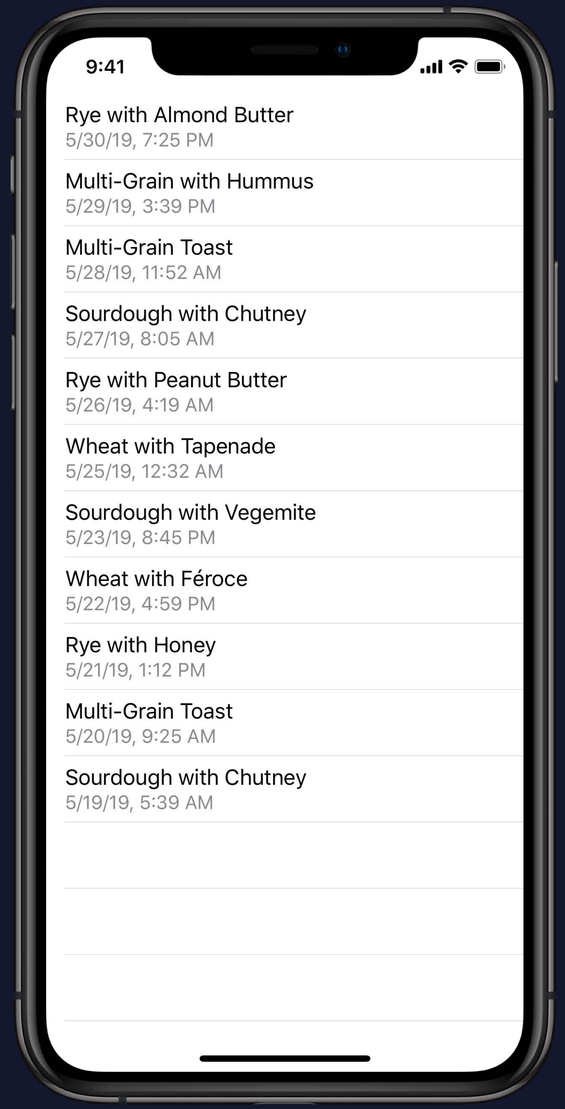
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



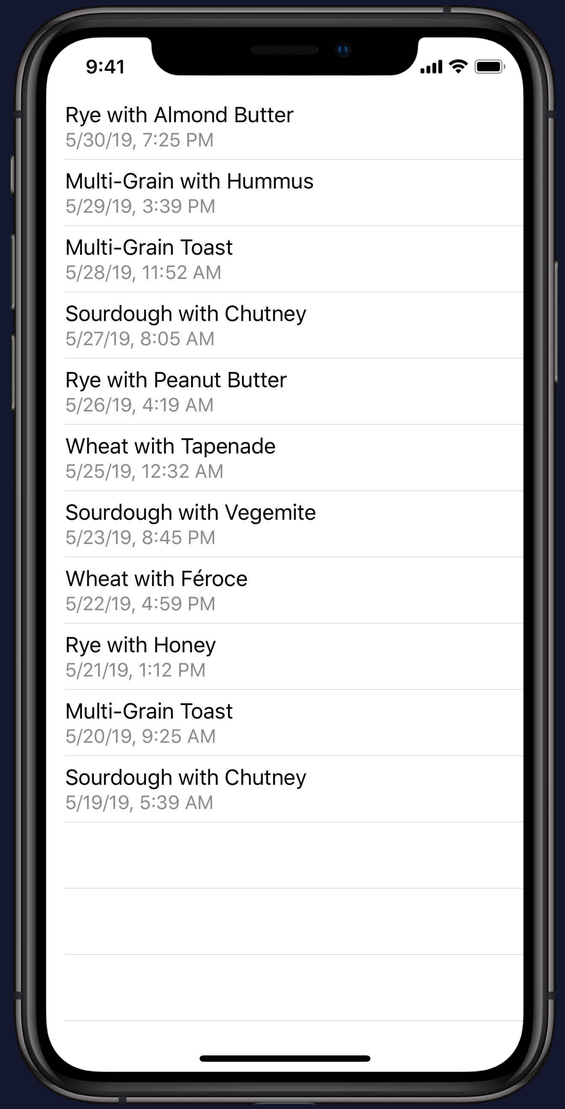
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



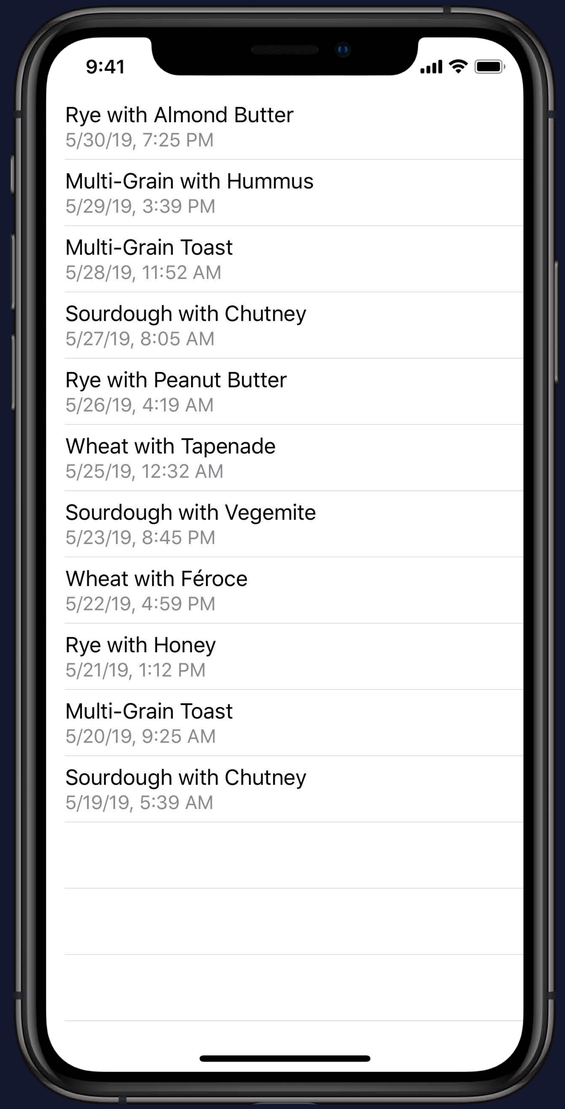
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



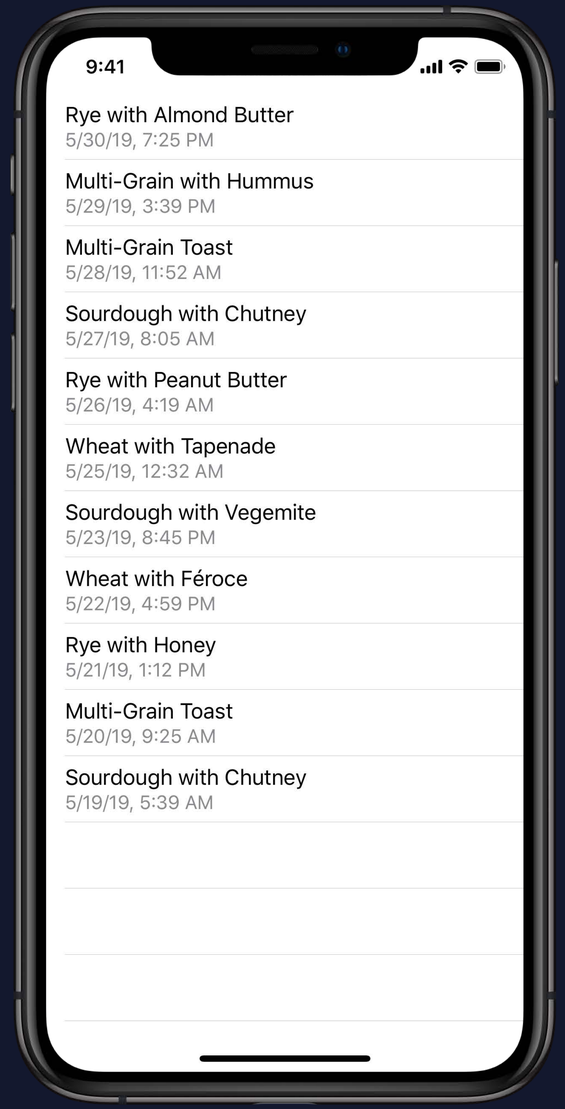
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



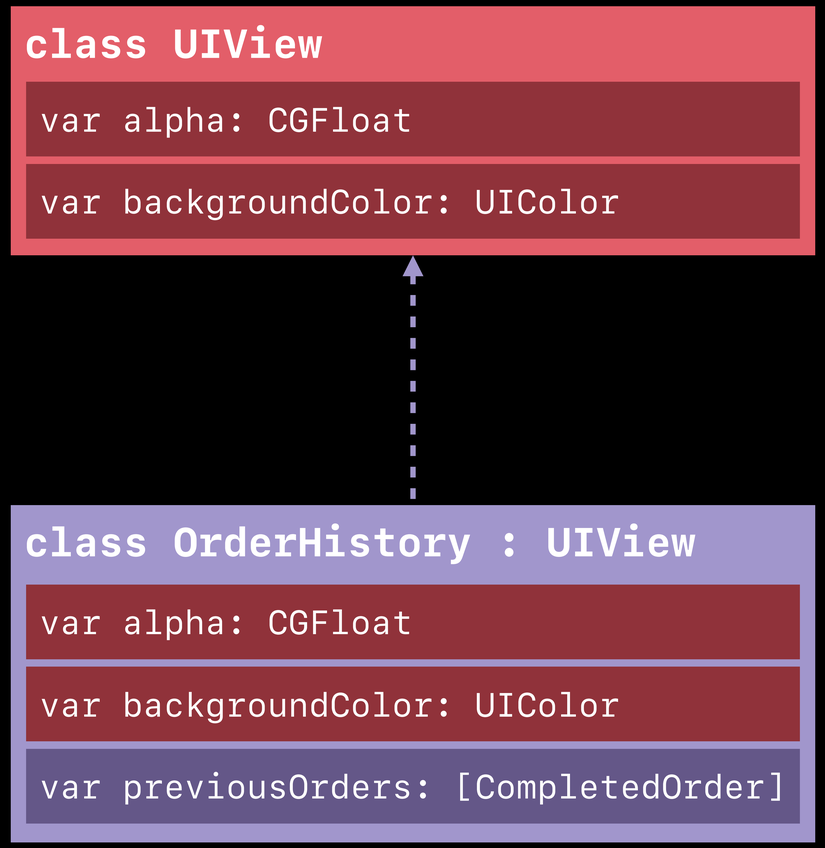
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



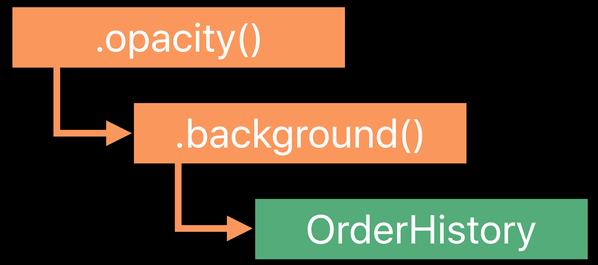
Views in UIKit

```
class UIView
var alpha: CGFloat
var backgroundColor: UIColor
```

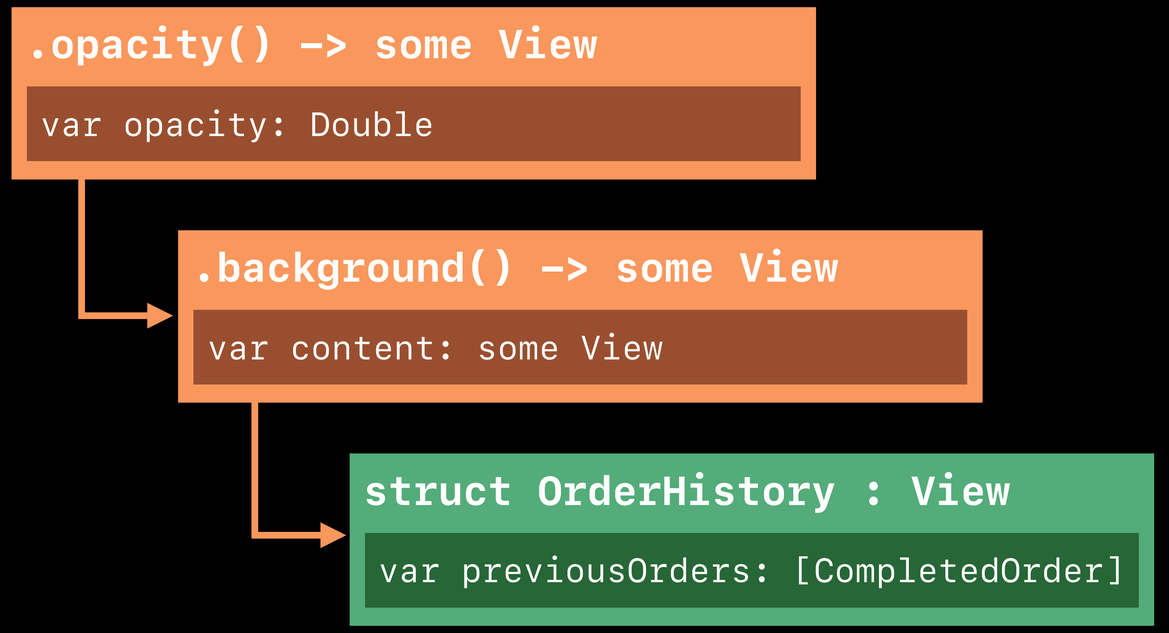
Views in UIKit



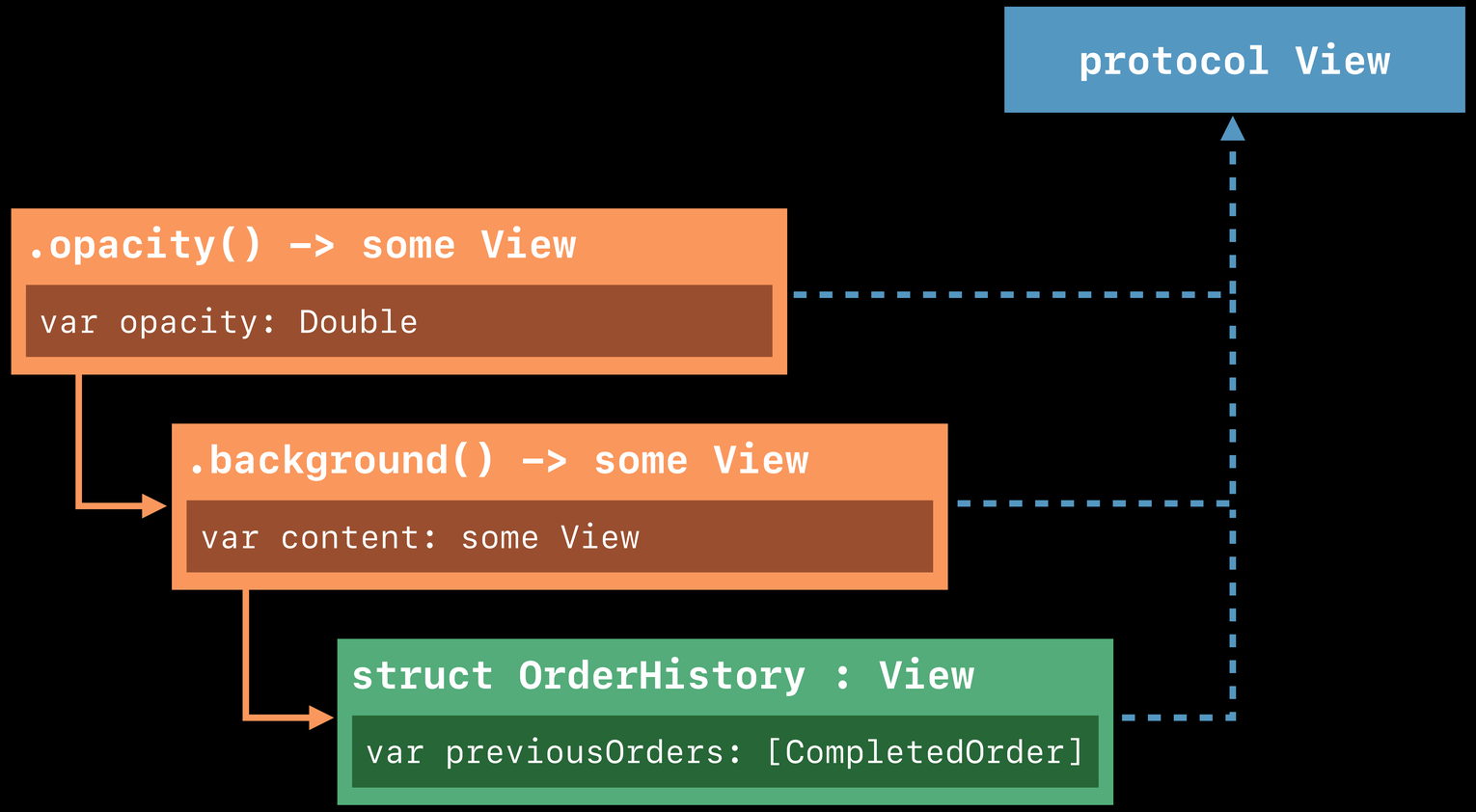
Views Using Modifiers



Views Using Modifiers



Views Using Modifiers



A view defines a piece of UI

```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```

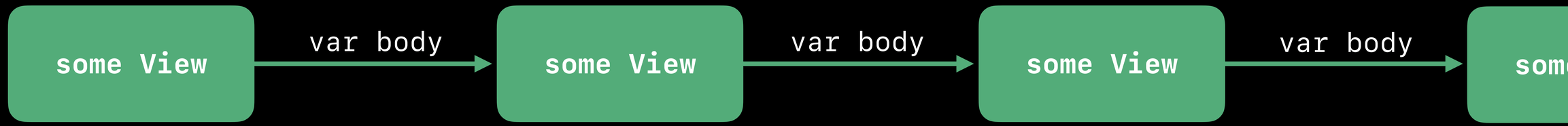
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```


```swift
// A piece of user interface
public protocol View {
    // The type of view representing the body of this view
    associatedtype Body : View

    // Declares the content and behavior of this view
    var body: Body { get }
}
```

Views

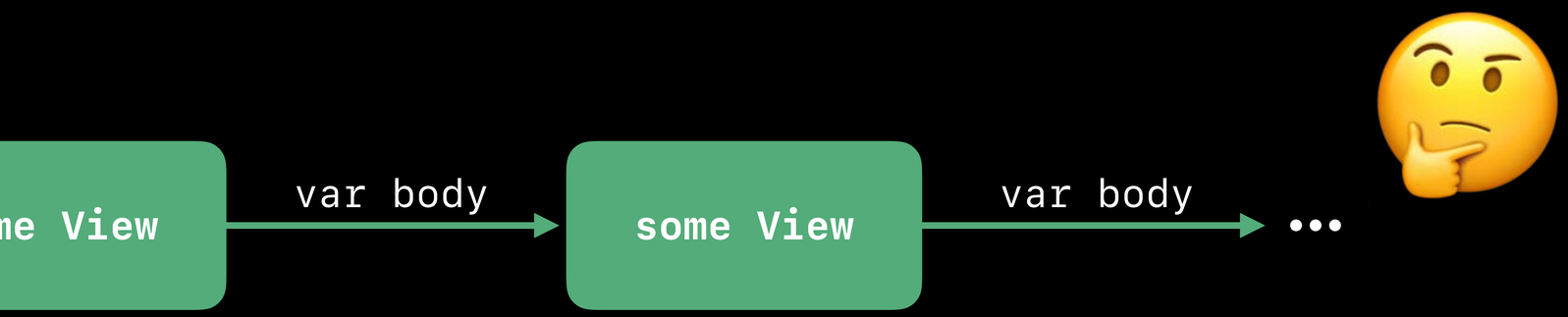
Views

Views



Views

Views



🤔

Primitive Views

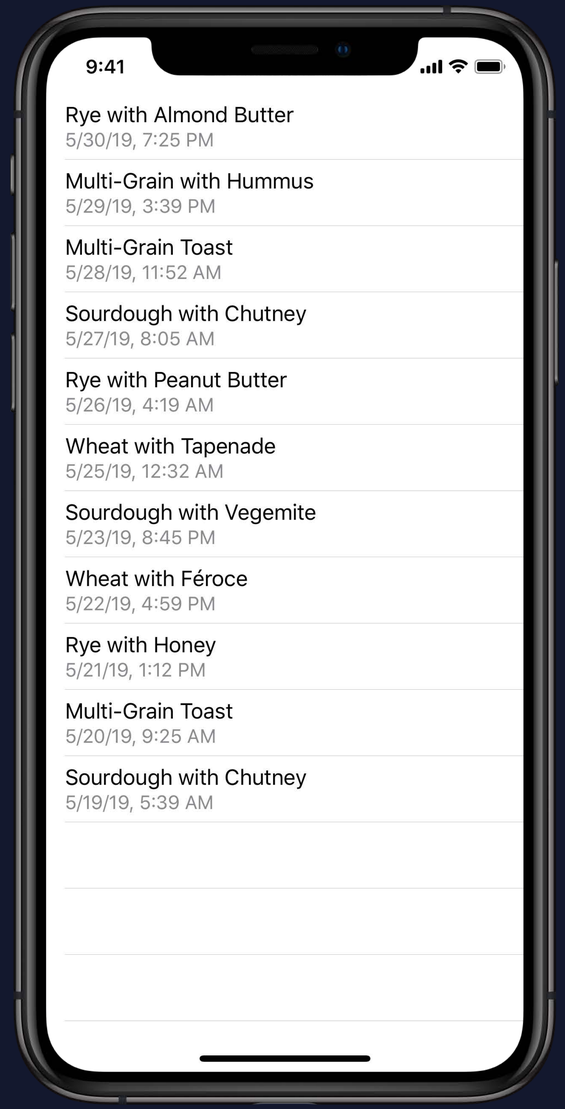
`Text`	`Color`	`Spacer`
`Image`	`Shape`	`Divider`

Primitive Views

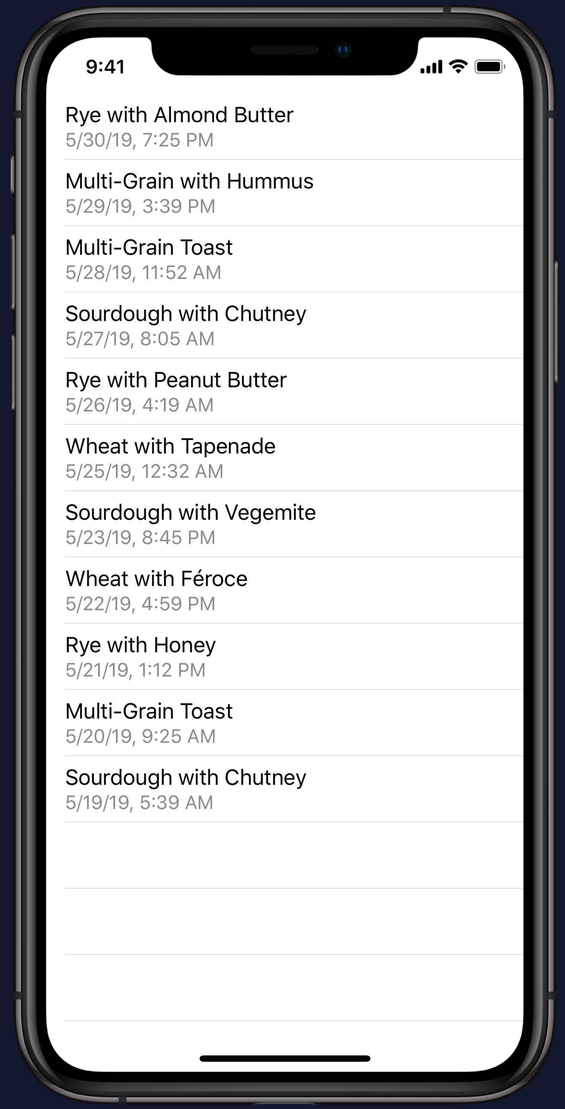
`Text` `Color` `Spacer`

`Image` `Shape` `Divider`

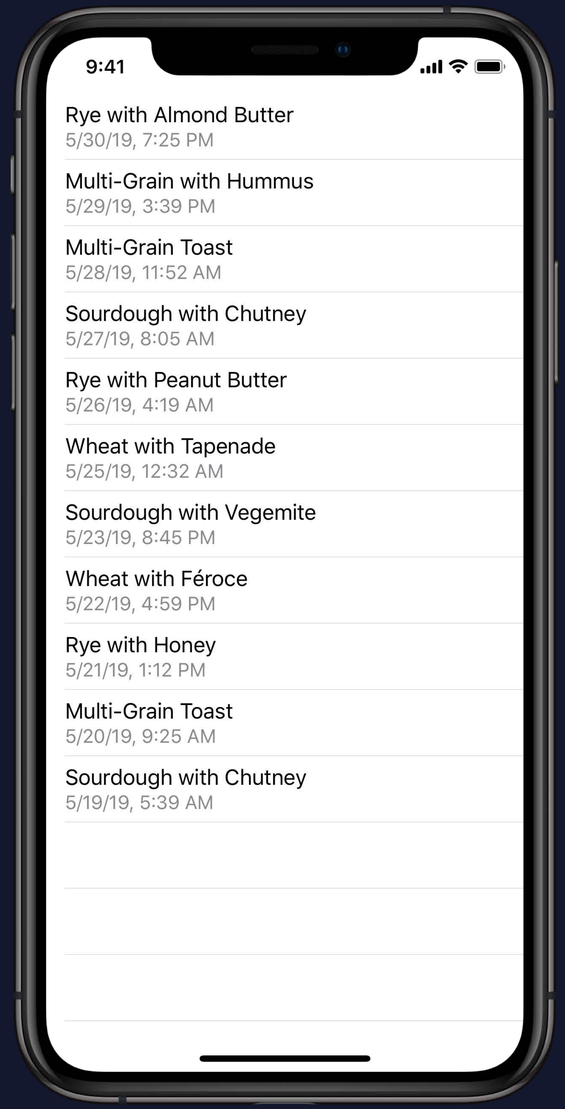
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



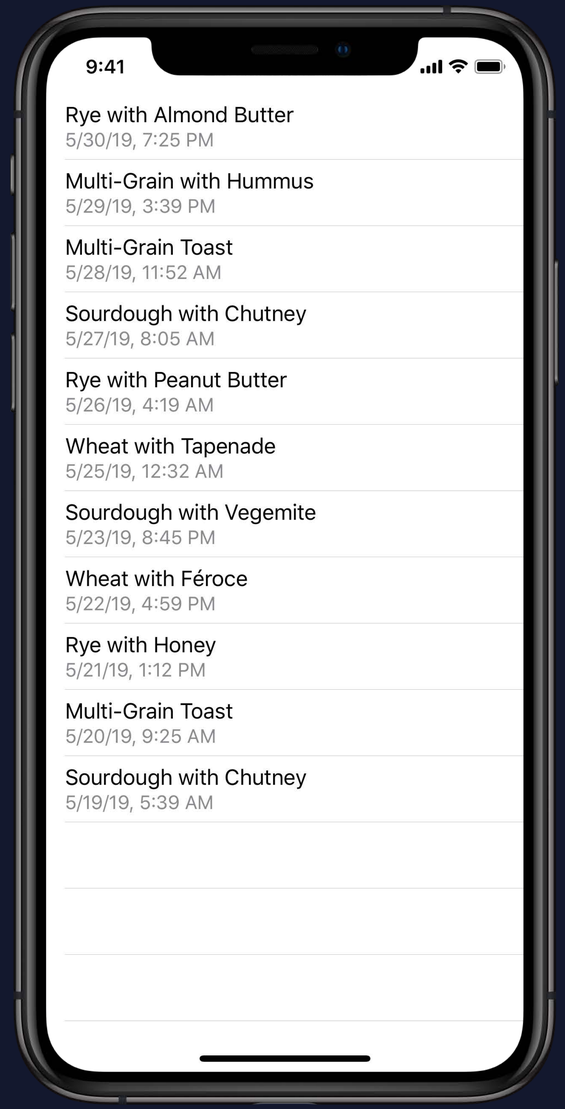
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



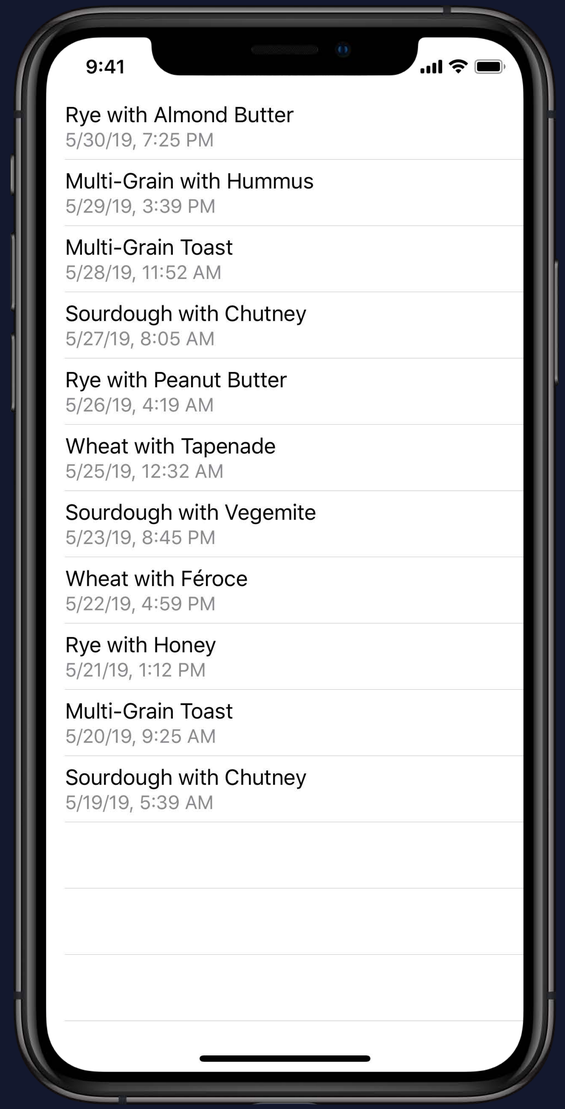
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



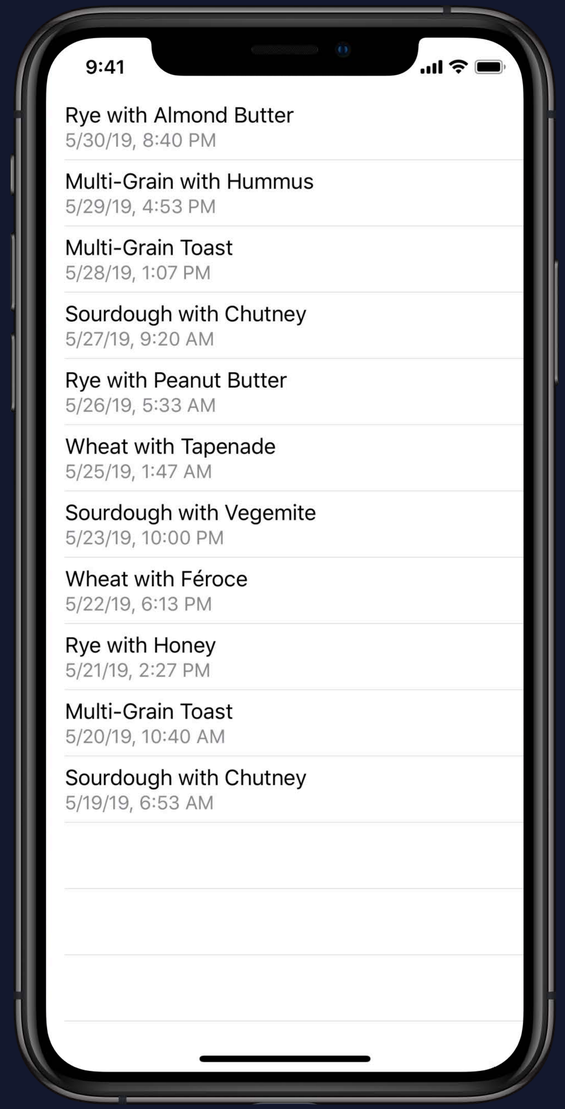
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



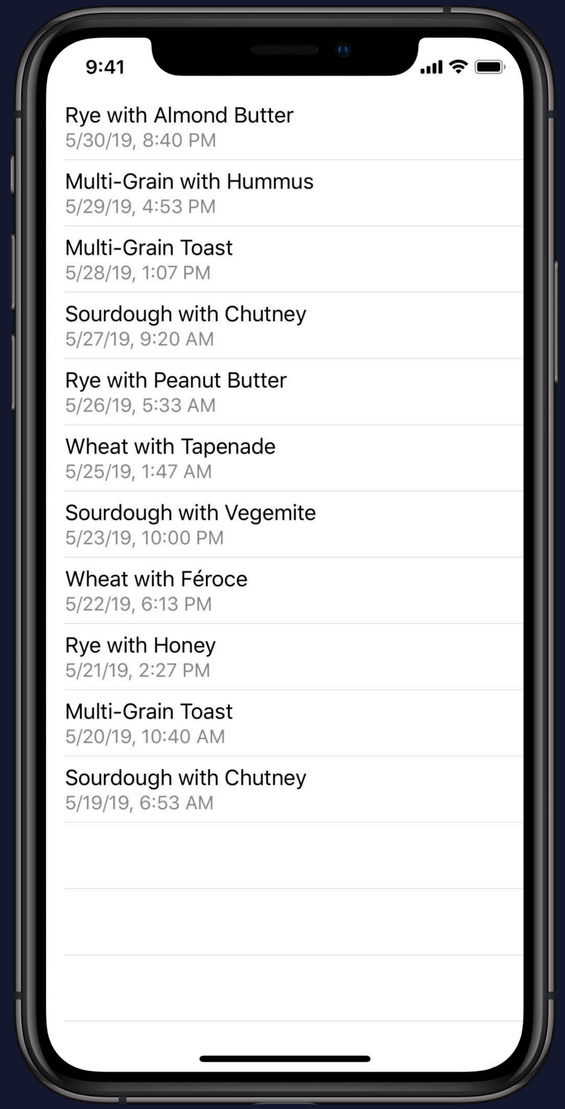
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```

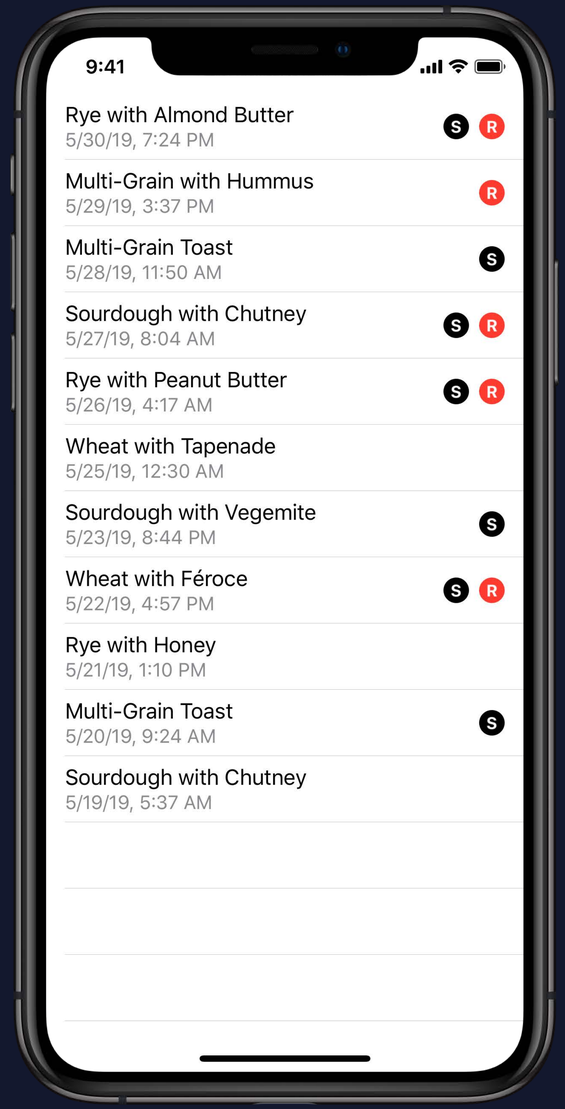


```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```

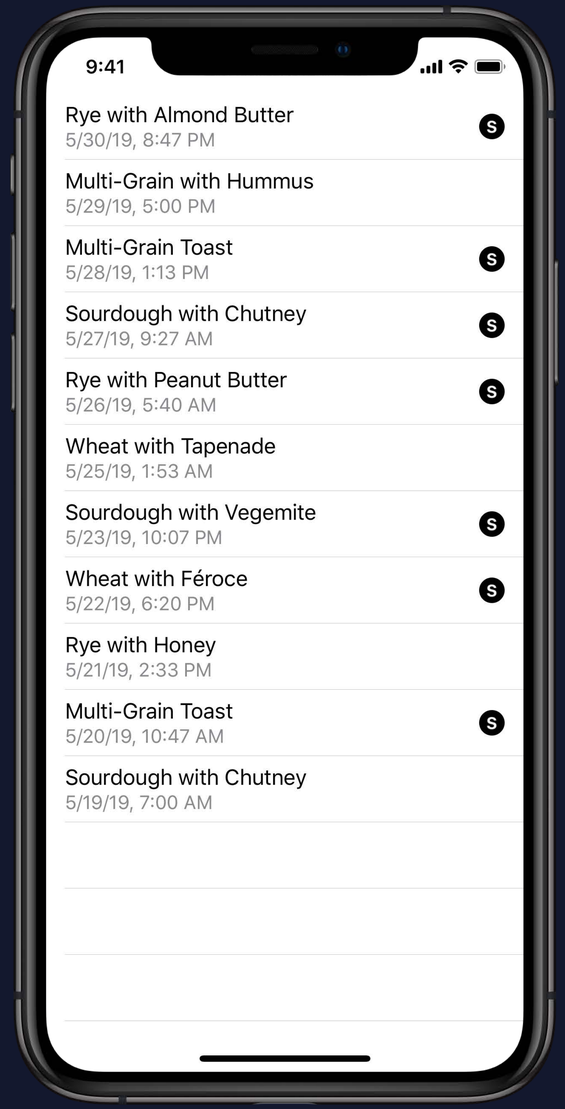



.font(.title)
Text
VStack

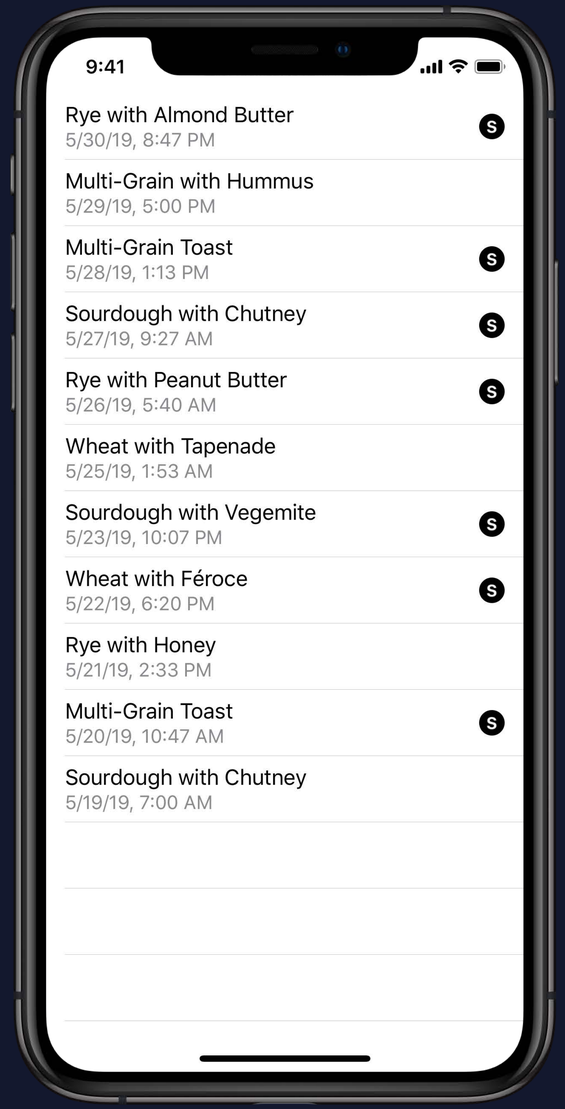
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
        }
    }
}
```



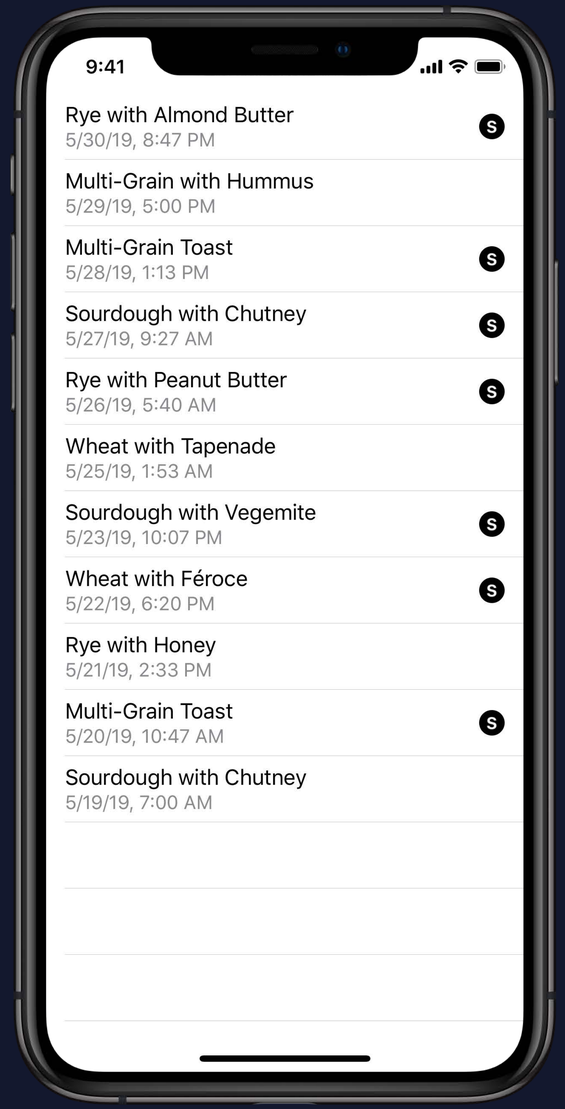
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            HStack {
                VStack(alignment: .leading) {
                    Text(order.summary)
                    Text(order.purchaseDate)
                        .font(.subheadline)
                        .foregroundColor(.secondary)
                }
                Spacer()
                if order.includeSalt {
                    SaltIcon()
                }
            }
        }
    }
}
```



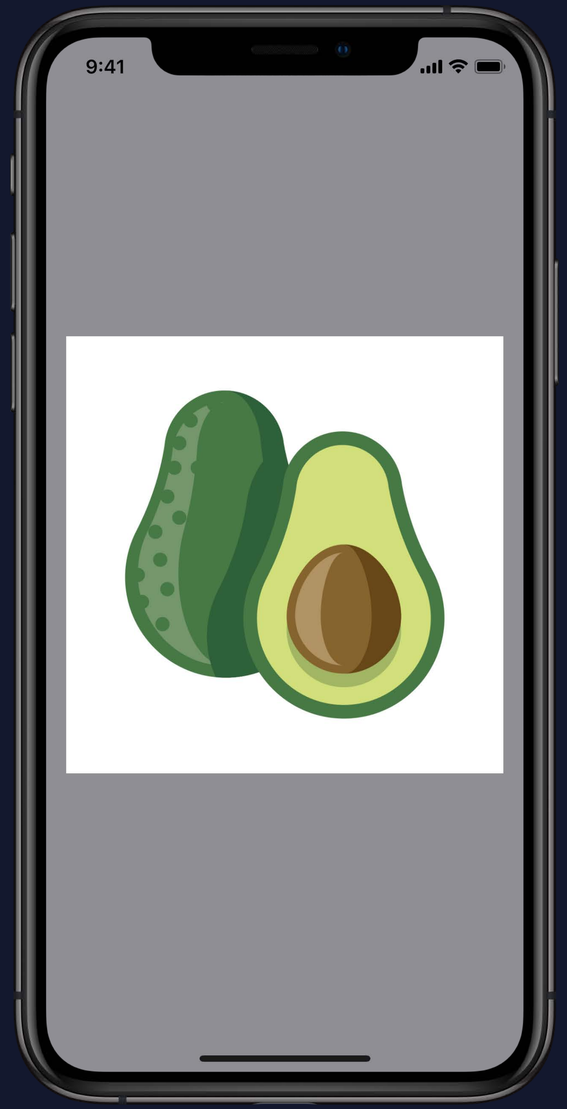
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            HStack {
                VStack(alignment: .leading) {
                    Text(order.summary)
                    Text(order.purchaseDate)
                        .font(.subheadline)
                        .foregroundColor(.secondary)
                }
                Spacer()
                if order.includeSalt {
                    SaltIcon()
                }
            }
        }
    }
}
```



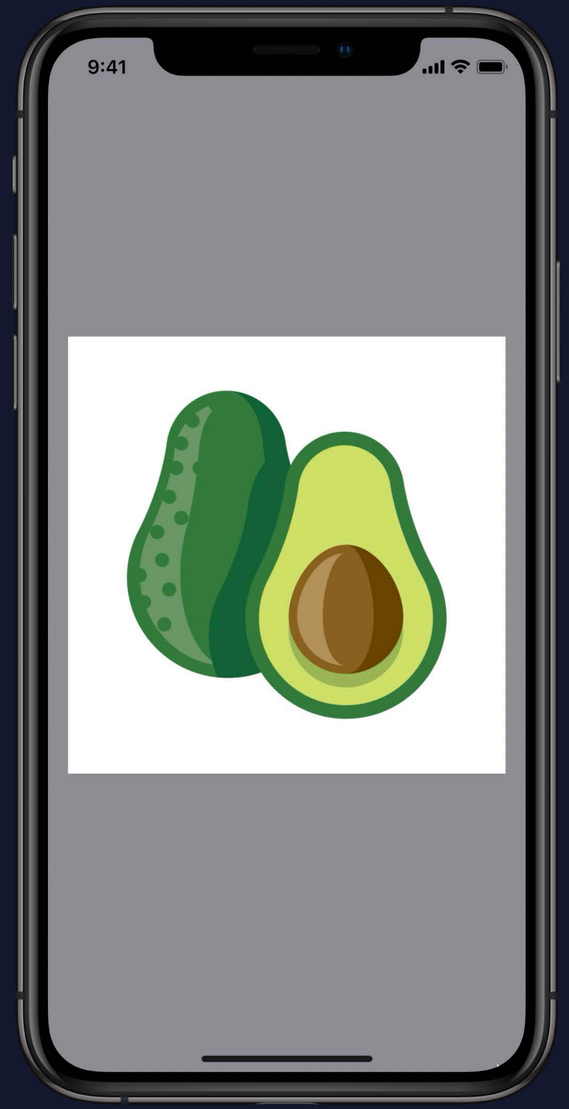
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            HStack {
                VStack(alignment: .leading) {
                    Text(order.summary)
                    Text(order.purchaseDate)
                        .font(.subheadline)
                        .foregroundColor(.secondary)
                }
                Spacer()
                if order.includeSalt {
                    SaltIcon()
                }
            }
        }
    }
}
```



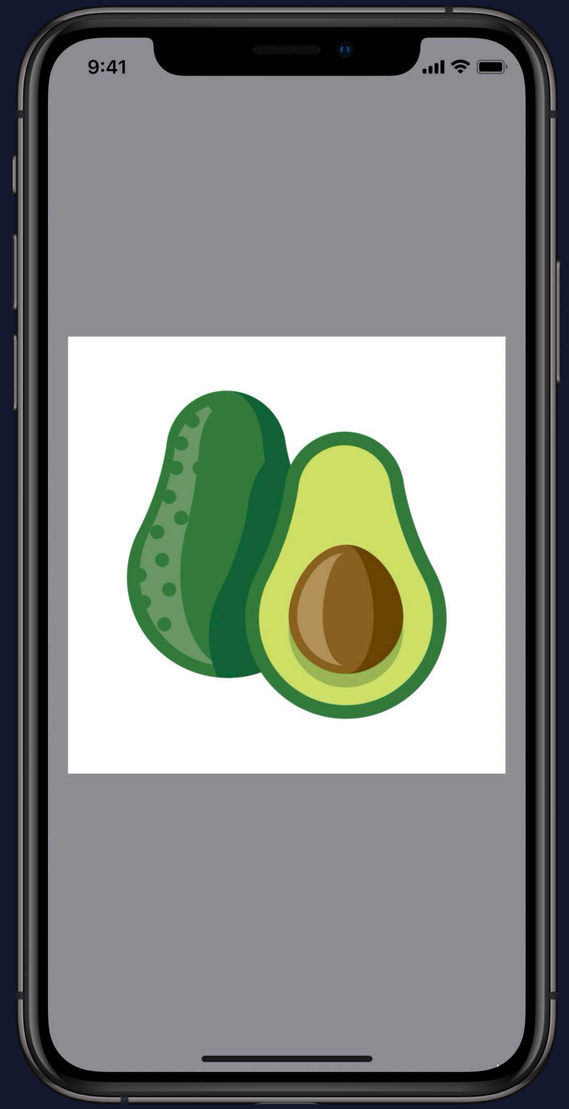
```
struct IconOrientationEditor : View {
    let flipped: Bool

    var body: some View {
        ZStack {
            Color.gray

            if flipped {
                AppIcon()
                    .rotationEffect(.degrees(180))
            } else {
                AppIcon()
            }
        }
    }
}
```



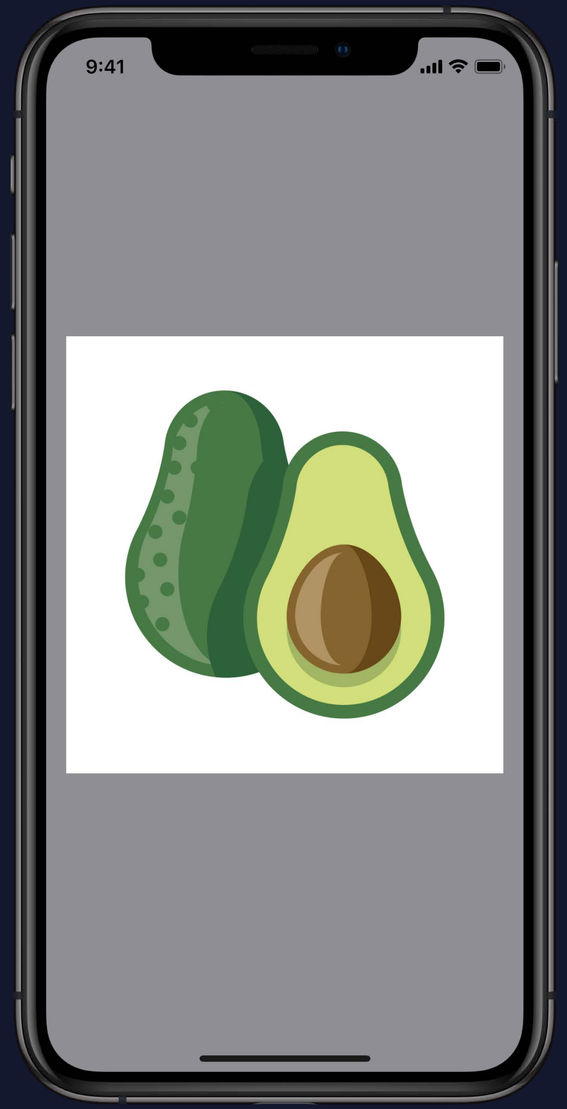
```
struct IconOrientationEditor : View {
    let flipped: Bool

    var body: some View {
        ZStack {
            Color.gray

            if flipped {
                AppIcon()
                    .rotationEffect(.degrees(180))
            } else {
                AppIcon()
            }
        }
    }
}
```



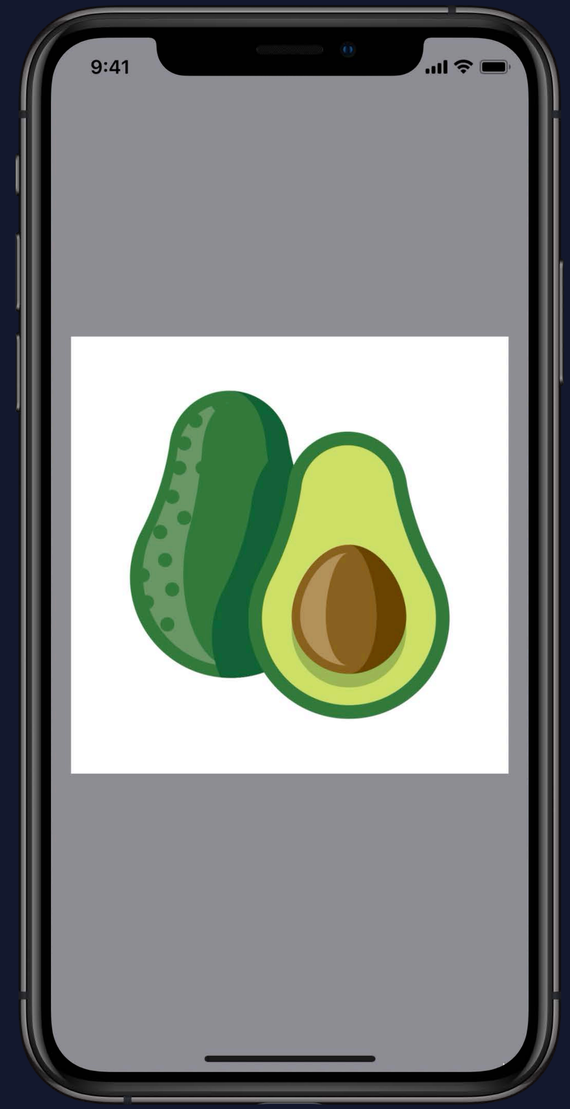
```
struct IconOrientationEditor : View {
    let flipped: Bool

    var body: some View {
        ZStack {
            Color.gray

            if flipped {
                AppIcon()
                    .rotationEffect(.degrees(180))
            } else {
                AppIcon()
            }
        }
    }
}
```



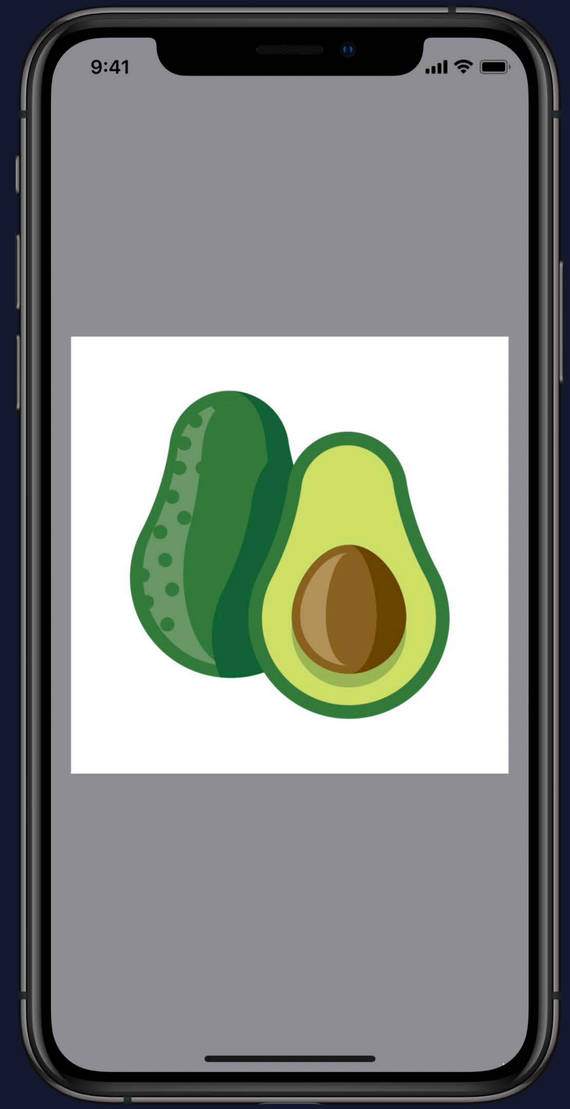
```
struct IconOrientationEditor : View {
    let flipped: Bool

    var body: some View {
        ZStack {
            Color.gray

            AppIcon()
                .rotationEffect(.degrees(flipped ? 180 : 0))
        }
    }
}
```



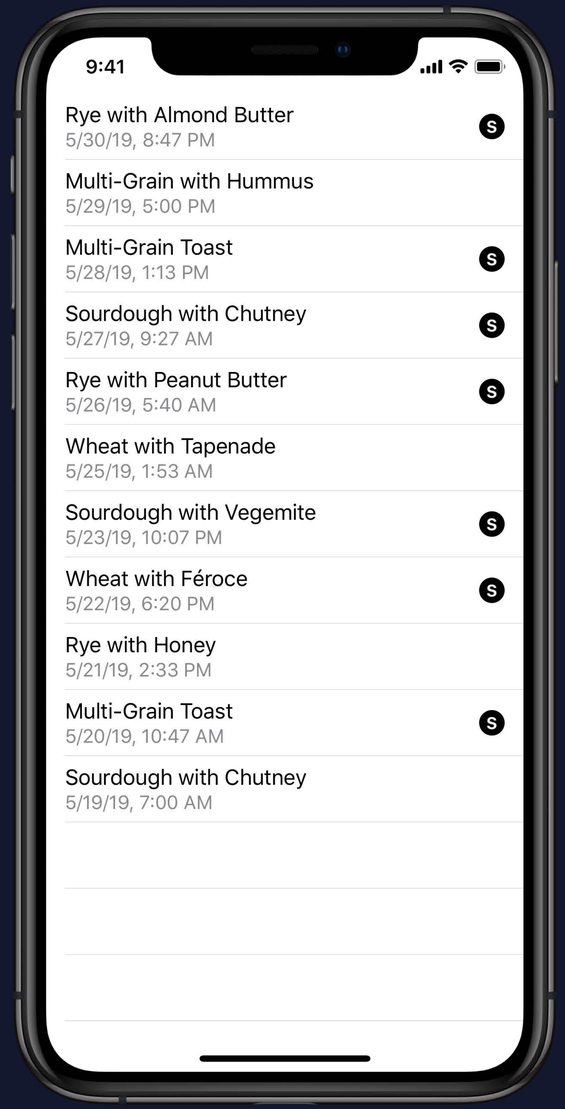
```
struct IconOrientationEditor : View {
    let flipped: Bool

    var body: some View {
        ZStack {
            Color.gray

            AppIcon()
                .rotationEffect(.degrees(flipped ? 180 : 0))
        }
    }
}
```



```
struct IconOrientationEditor : View {
    let flipped: Bool

    var body: some View {
        ZStack {
            Color.gray

            AppIcon()
                .rotationEffect(.degrees(flipped ? 180 : 0))
        }
    }
}
```



```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            HStack {
                VStack(alignment: .leading) {
                    Text(order.summary)
                    Text(order.purchaseDate)
                        .font(.subheadline)
                        .foregroundColor(.secondary)
                }
                Spacer()
                if order.includeSalt {
                    SaltIcon()
                }
            }
        }
    }
}
```



```
struct OrderHistory : View {
   let previousOrders: [CompletedOrder]

   var body: some View {
      List(previousOrders) { order in
         HStack {
            VStack(alignment: .leading) {
               Text(order.summary)
               Text(order.purchaseDate)
                  .font(.subheadline)
                  .foregroundColor(.secondary)
            }
            Spacer()
            if order.includeSalt {
               SaltIcon()
            }
         }
      }
   }
}
```

```
struct OrderCell : View {

   var body: some View {

   }
}
```

```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in









        }
    }
}
```

```
    struct OrderCell : View {



        var body: some View {
            HStack {
                VStack(alignment: .leading) {
                    Text(order.summary)
                    Text(order.purchaseDate)
                        .font(.subheadline)
                        .foregroundColor(.secondary)
                }
                Spacer()
                if order.includeSalt {
                    SaltIcon()
                }
            }
        }
    }
```

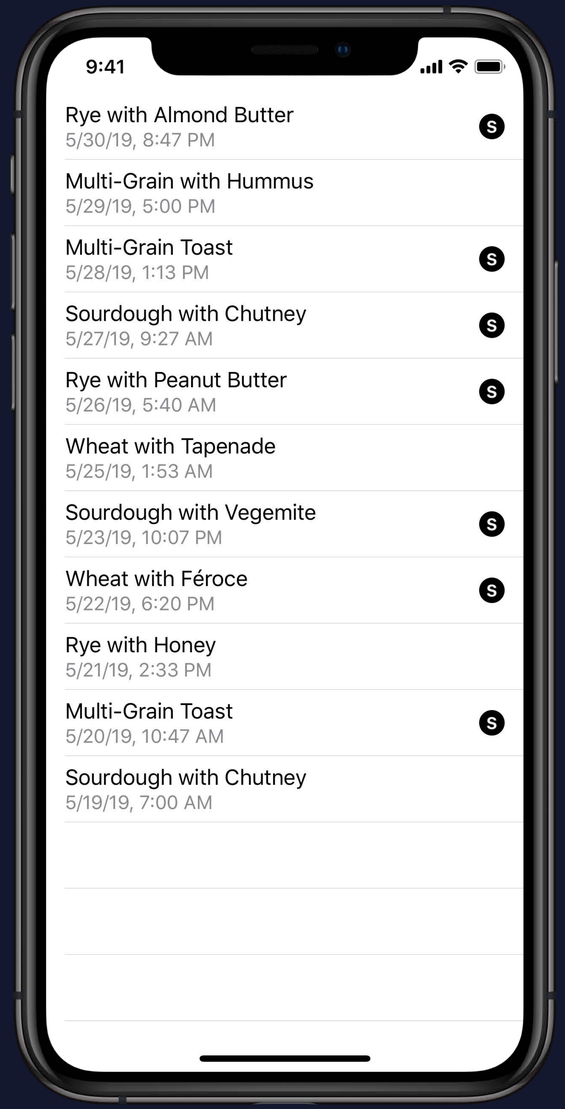
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in









        }
    }
}
```

```
struct OrderCell : View {
    var order: CompletedOrder

    var body: some View {
        HStack {
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
            Spacer()
            if order.includeSalt {
                SaltIcon()
            }
        }
    }
}
```

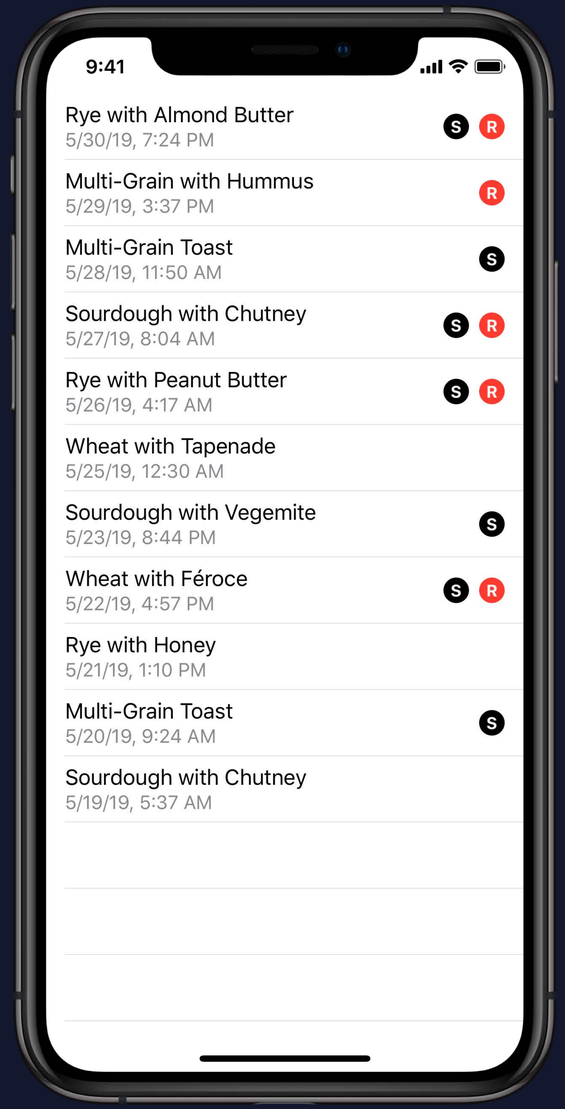
```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            OrderCell(order: order)
        }
    }
}
```

```
struct OrderCell : View {
    var order: CompletedOrder

    var body: some View {
        HStack {
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
            Spacer()
            if order.includeSalt {
                SaltIcon()
            }
        }
    }
}
```

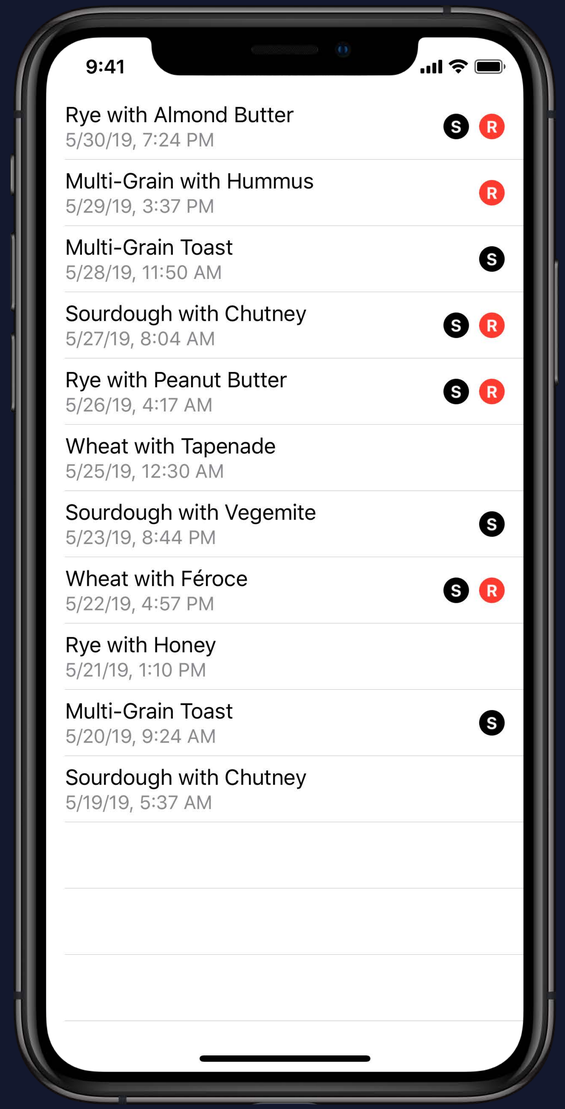
```
struct OrderCell : View {
    var order: CompletedOrder

    var body: some View {
        HStack {
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
            Spacer()
            if order.includeSalt {
                SaltIcon()
            }
        }
    }
}
```



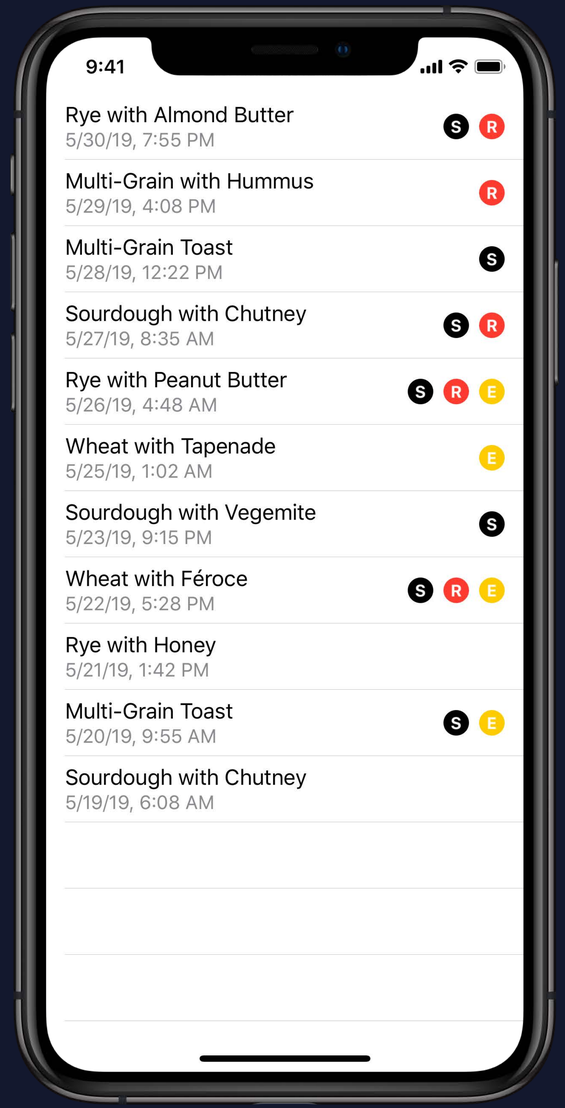
```
struct OrderCell : View {
    var order: CompletedOrder

    var body: some View {
        HStack {
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
            Spacer()
            if order.includeSalt {
                SaltIcon()
            }
            if order.includeRedPepperFlakes {
                RedPepperFlakesIcon()
            }
        }
    }
}
```



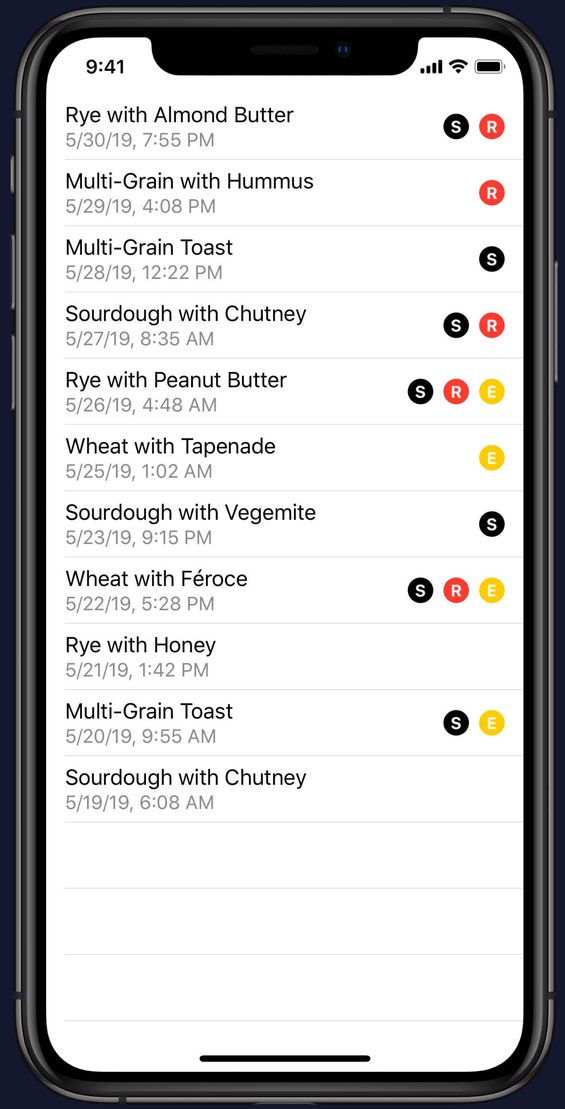
```
struct OrderCell : View {
    var order: CompletedOrder

    var body: some View {
        HStack {
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
            Spacer()
            ForEach(order.toppings) { topping in
                ToppingIcon(topping)
            }
        }
    }
}
```

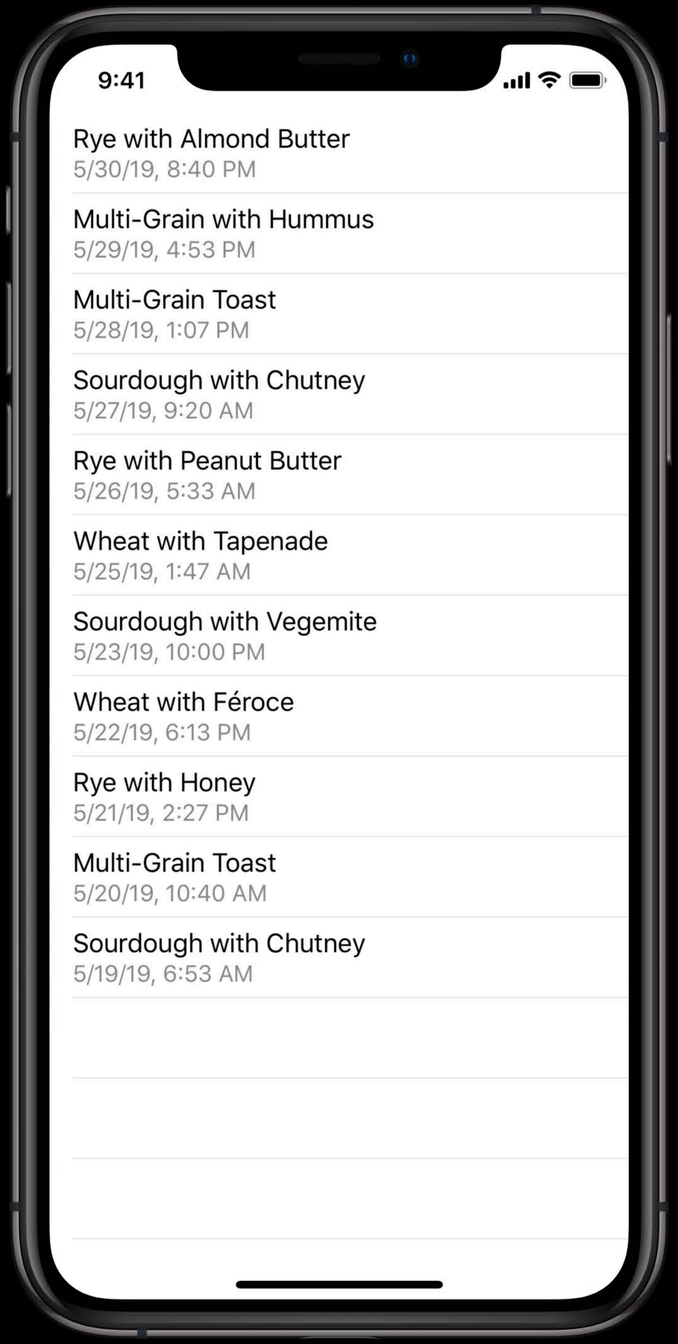


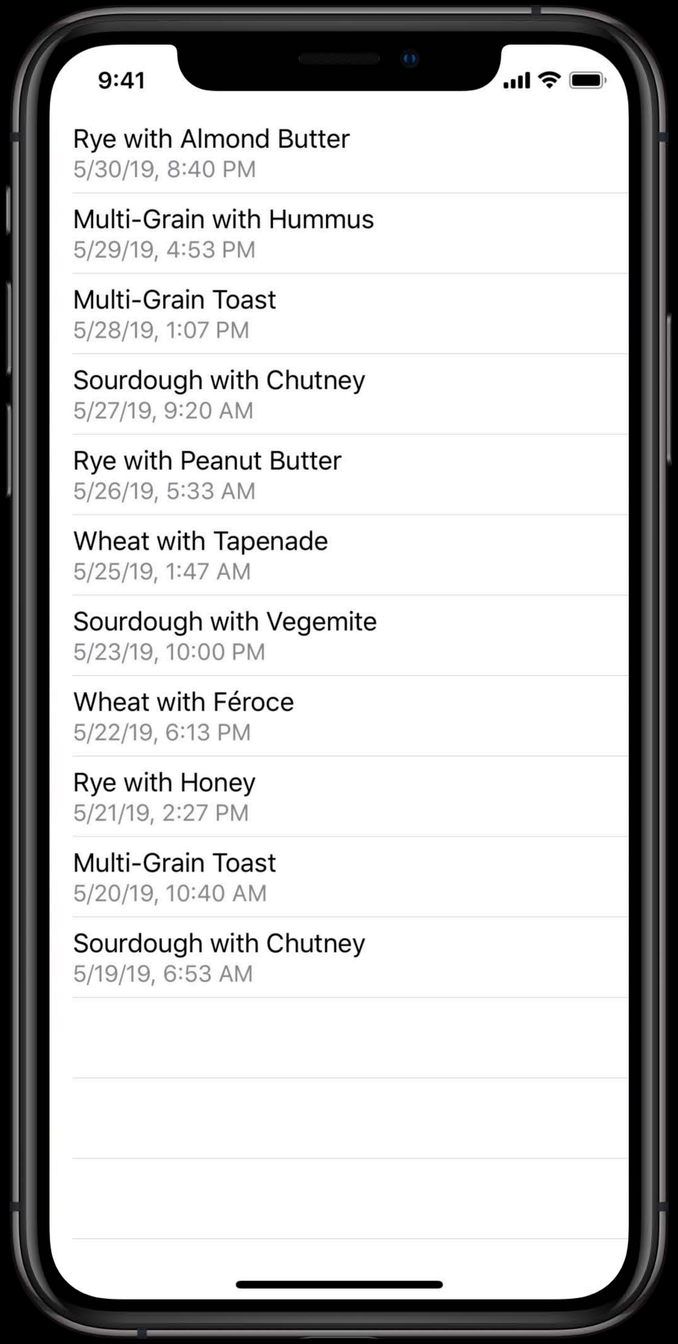
```
struct OrderCell : View {
    var order: CompletedOrder

    var body: some View {
        HStack {
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
            Spacer()
            ForEach(order.toppings) { topping in
                ToppingIcon(topping)
            }
        }
    }
}
```

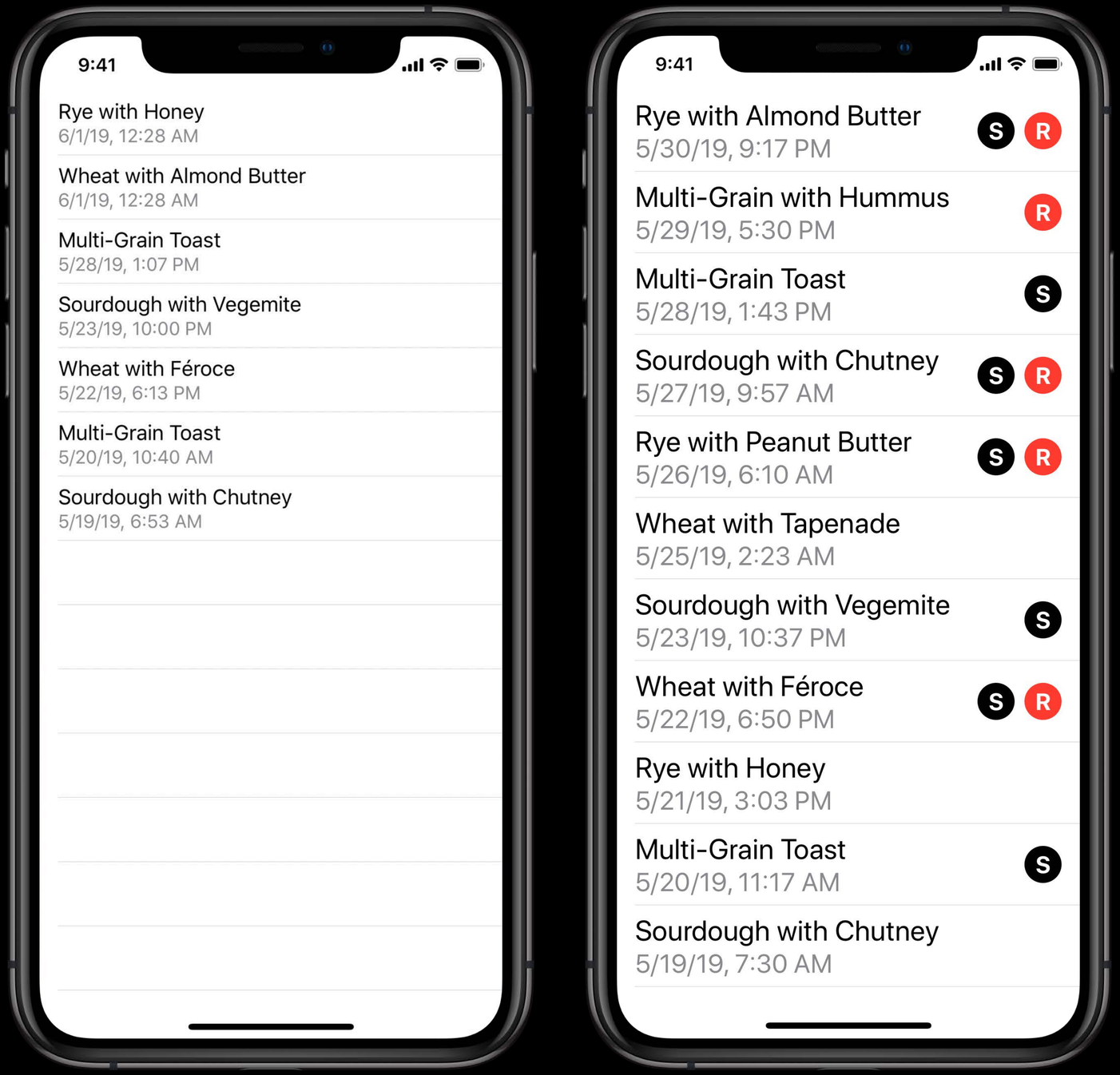


```
struct OrderCell : View {
    var order: CompletedOrder

    var body: some View {
        HStack {
            VStack(alignment: .leading) {
                Text(order.summary)
                Text(order.purchaseDate)
                    .font(.subheadline)
                    .foregroundColor(.secondary)
            }
            Spacer()
            ForEach(order.toppings) { topping in
                ToppingIcon(topping)
            }
        }
    }
}
```

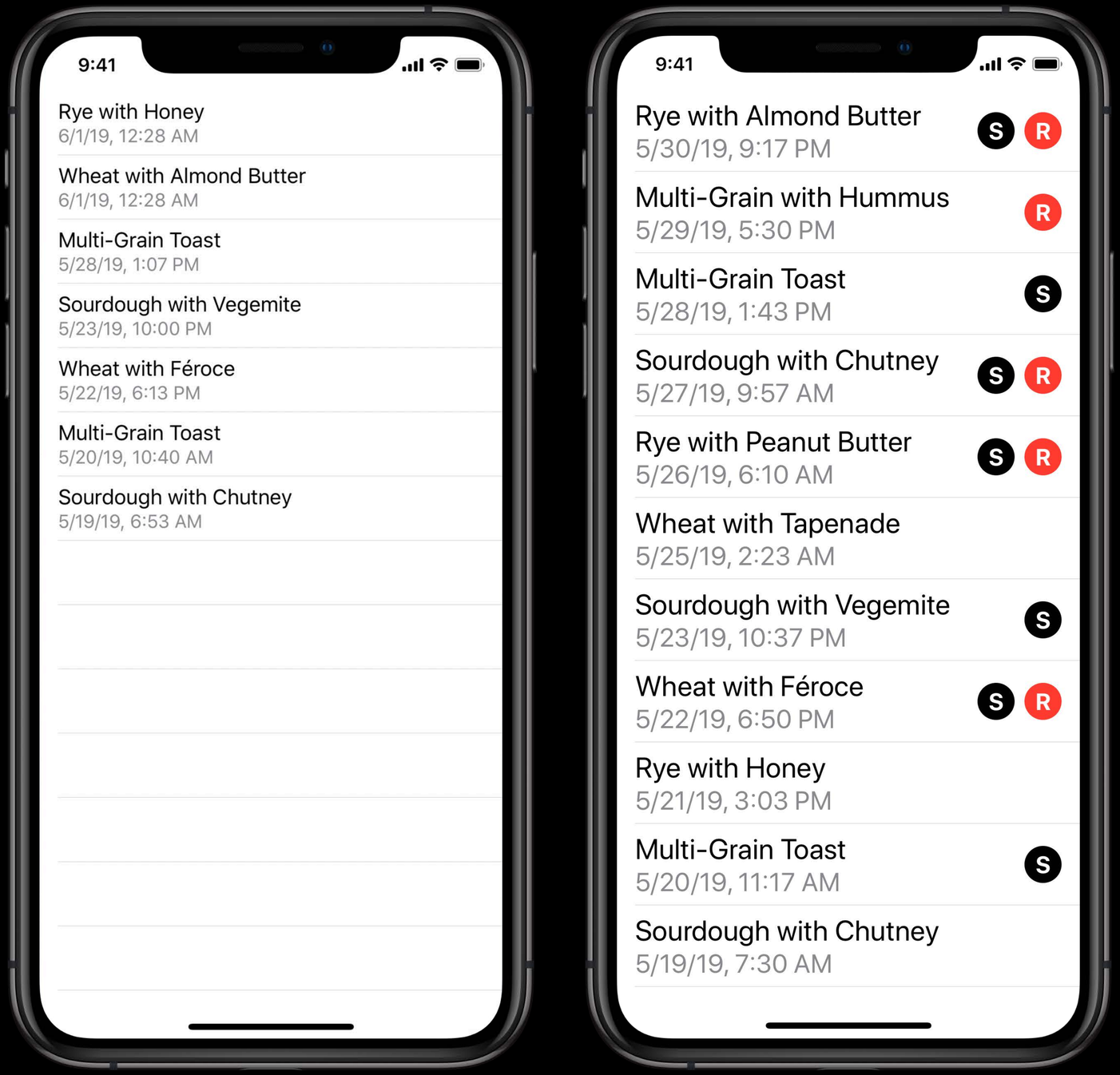



9:41
Rye with Almond Butter
5/30/19, 8:40 PM
Multi-Grain with Hummus
5/29/19, 4:53 PM
Multi-Grain Toast
5/28/19, 1:07 PM
Sourdough with Chutney
5/27/19, 9:20 AM
Rye with Peanut Butter
5/26/19, 5:33 AM
Wheat with Tapenade
5/25/19, 1:47 AM
Sourdough with Vegemite
5/23/19, 10:00 PM
Wheat with Féroce
5/22/19, 6:13 PM
Rye with Honey
5/21/19, 2:27 PM
Multi-Grain Toast
5/20/19, 10:40 AM
Sourdough with Chutney
5/19/19, 6:53 AM

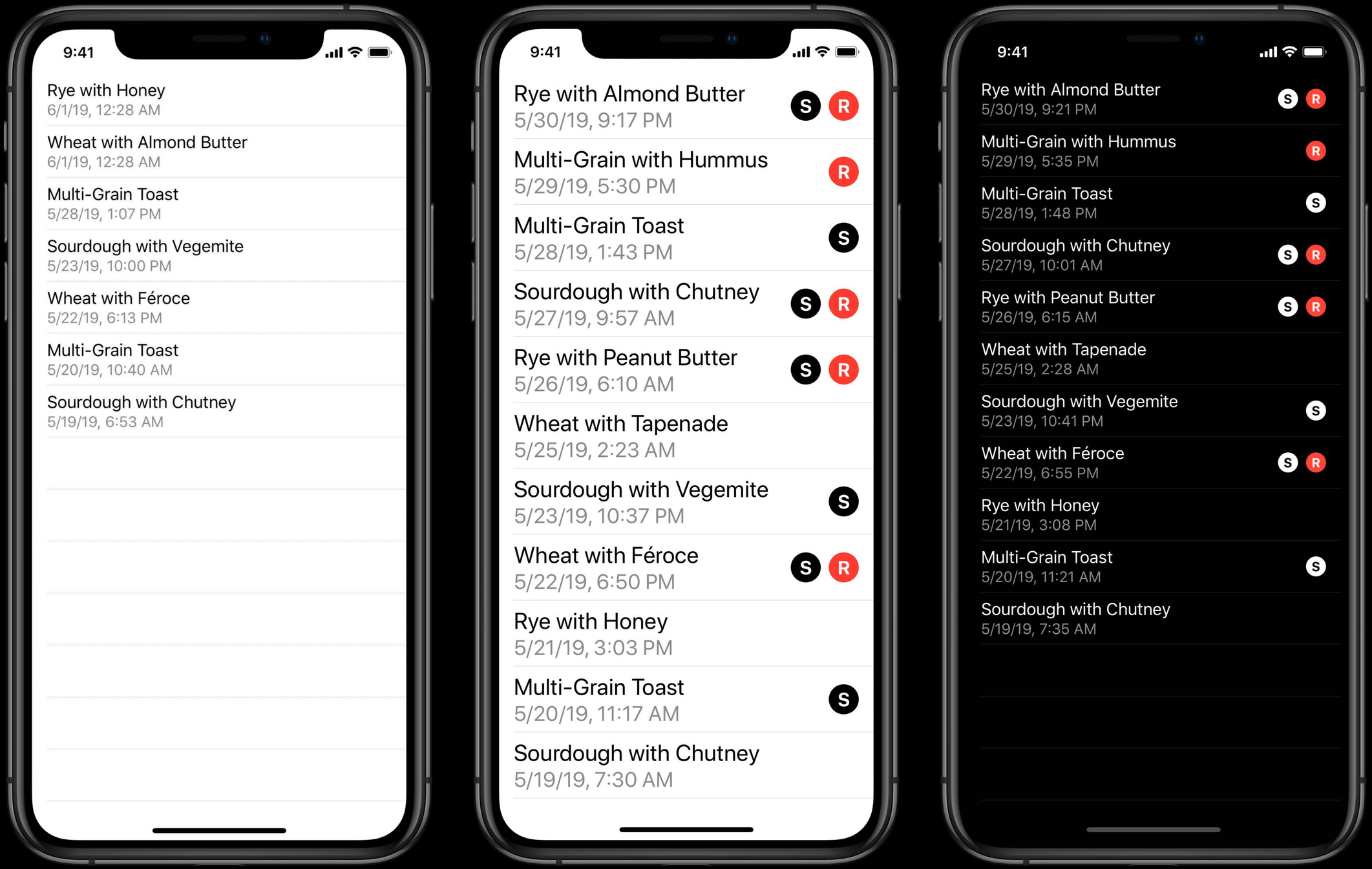

9:41
Rye with Almond Butter
5/30/19, 8:40 PM
Multi-Grain with Hummus
5/29/19, 4:53 PM
Multi-Grain Toast
5/28/19, 1:07 PM
Sourdough with Chutney
5/27/19, 9:20 AM
Rye with Peanut Butter
5/26/19, 5:33 AM
Wheat with Tapenade
5/25/19, 1:47 AM
Sourdough with Vegemite
5/23/19, 10:00 PM
Wheat with Féroce
5/22/19, 6:13 PM
Rye with Honey
5/21/19, 2:27 PM
Multi-Grain Toast
5/20/19, 10:40 AM
Sourdough with Chutney
5/19/19, 6:53 AM

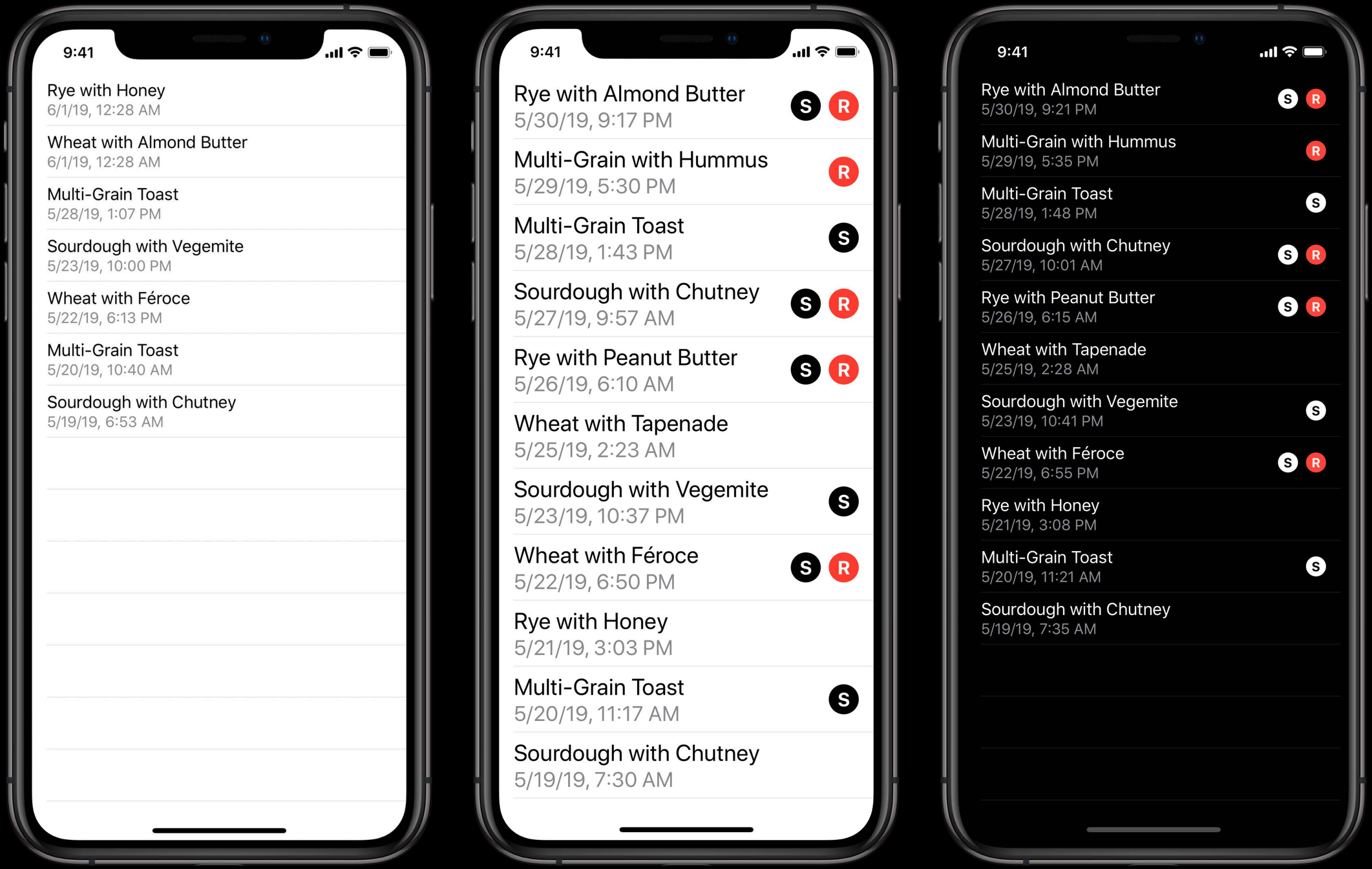

9:41
Rye with Honey
6/1/19, 12:28 AM
Wheat with Almond Butter
6/1/19, 12:28 AM
Multi-Grain Toast
5/28/19, 1:07 PM
Sourdough with Vegemite
5/23/19, 10:00 PM
Wheat with Féroce
5/22/19, 6:13 PM
Multi-Grain Toast
5/20/19, 10:40 AM
Sourdough with Chutney
5/19/19, 6:53 AM

9:41
Rye with Almond Butter
5/30/19, 9:17 PM
S
R
Multi-Grain with Hummus
5/29/19, 5:30 PM
R
Multi-Grain Toast
5/28/19, 1:43 PM
S
Sourdough with Chutney
5/27/19, 9:57 AM
S
R
Rye with Peanut Butter
5/26/19, 6:10 AM
S
R
Wheat with Tapenade
5/25/19, 2:23 AM
Sourdough with Vegemite
5/23/19, 10:37 PM
S
Wheat with Féroce
5/22/19, 6:50 PM
S
R
Rye with Honey
5/21/19, 3:03 PM
Multi-Grain Toast
5/20/19, 11:17 AM
S
Sourdough with Chutney
5/19/19, 7:30 AM


9:41
Rye with Honey
6/1/19, 12:28 AM
Wheat with Almond Butter
6/1/19, 12:28 AM
Multi-Grain Toast
5/28/19, 1:07 PM
Sourdough with Vegemite
5/23/19, 10:00 PM
Wheat with Féroce
5/22/19, 6:13 PM
Multi-Grain Toast
5/20/19, 10:40 AM
Sourdough with Chutney
5/19/19, 6:53 AM

9:41
Rye with Almond Butter
5/30/19, 9:17 PM
S
R
Multi-Grain with Hummus
5/29/19, 5:30 PM
R
Multi-Grain Toast
5/28/19, 1:43 PM
S
Sourdough with Chutney
5/27/19, 9:57 AM
S
R
Rye with Peanut Butter
5/26/19, 6:10 AM
S
R
Wheat with Tapenade
5/25/19, 2:23 AM
Sourdough with Vegemite
5/23/19, 10:37 PM
S
Wheat with Féroce
5/22/19, 6:50 PM
S
R
Rye with Honey
5/21/19, 3:03 PM
Multi-Grain Toast
5/20/19, 11:17 AM
S
Sourdough with Chutney
5/19/19, 7:30 AM


9:41
Rye with Honey
6/1/19, 12:28 AM
Wheat with Almond Butter
6/1/19, 12:28 AM
Multi-Grain Toast
5/28/19, 1:07 PM
Sourdough with Vegemite
5/23/19, 10:00 PM
Wheat with Féroce
5/22/19, 6:13 PM
Multi-Grain Toast
5/20/19, 10:40 AM
Sourdough with Chutney
5/19/19, 6:53 AM
9:41
Rye with Almond Butter
5/30/19, 9:17 PM
S R
Multi-Grain with Hummus
5/29/19, 5:30 PM
R
Multi-Grain Toast
5/28/19, 1:43 PM
S
Sourdough with Chutney
5/27/19, 9:57 AM
S R
Rye with Peanut Butter
5/26/19, 6:10 AM
S R
Wheat with Tapenade
5/25/19, 2:23 AM
Sourdough with Vegemite
5/23/19, 10:37 PM
S
Wheat with Féroce
5/22/19, 6:50 PM
S R
Rye with Honey
5/21/19, 3:03 PM
Multi-Grain Toast
5/20/19, 11:17 AM
S
Sourdough with Chutney
5/19/19, 7:30 AM
9:41
Rye with Almond Butter
5/30/19, 9:21 PM
S R
Multi-Grain with Hummus
5/29/19, 5:35 PM
R
Multi-Grain Toast
5/28/19, 1:48 PM
S
Sourdough with Chutney
5/27/19, 10:01 AM
S R
Rye with Peanut Butter
5/26/19, 6:15 AM
S R
Wheat with Tapenade
5/25/19, 2:28 AM
Sourdough with Vegemite
5/23/19, 10:41 PM
S
Wheat with Féroce
5/22/19, 6:55 PM
S R
Rye with Honey
5/21/19, 3:08 PM
Multi-Grain Toast
5/20/19, 11:21 AM
S
Sourdough with Chutney
5/19/19, 7:35 AM


9:41
Rye with Honey
6/1/19, 12:28 AM
Wheat with Almond Butter
6/1/19, 12:28 AM
Multi-Grain Toast
5/28/19, 1:07 PM
Sourdough with Vegemite
5/23/19, 10:00 PM
Wheat with Féroce
5/22/19, 6:13 PM
Multi-Grain Toast
5/20/19, 10:40 AM
Sourdough with Chutney
5/19/19, 6:53 AM
9:41
Rye with Almond Butter
5/30/19, 9:17 PM
S R
Multi-Grain with Hummus
5/29/19, 5:30 PM
R
Multi-Grain Toast
5/28/19, 1:43 PM
S
Sourdough with Chutney
5/27/19, 9:57 AM
S R
Rye with Peanut Butter
5/26/19, 6:10 AM
S R
Wheat with Tapenade
5/25/19, 2:23 AM
Sourdough with Vegemite
5/23/19, 10:37 PM
S
Wheat with Féroce
5/22/19, 6:50 PM
S R
Rye with Honey
5/21/19, 3:03 PM
Multi-Grain Toast
5/20/19, 11:17 AM
S
Sourdough with Chutney
5/19/19, 7:30 AM
9:41
Rye with Almond Butter
5/30/19, 9:21 PM
S R
Multi-Grain with Hummus
5/29/19, 5:35 PM
R
Multi-Grain Toast
5/28/19, 1:48 PM
S
Sourdough with Chutney
5/27/19, 10:01 AM
S R
Rye with Peanut Butter
5/26/19, 6:15 AM
S R
Wheat with Tapenade
5/25/19, 2:28 AM
Sourdough with Vegemite
5/23/19, 10:41 PM
S
Wheat with Féroce
5/22/19, 6:55 PM
S R
Rye with Honey
5/21/19, 3:08 PM
Multi-Grain Toast
5/20/19, 11:21 AM
S
Sourdough with Chutney
5/19/19, 7:35 AM

Getting started: Views and modifiers

Building custom views

Composing controls

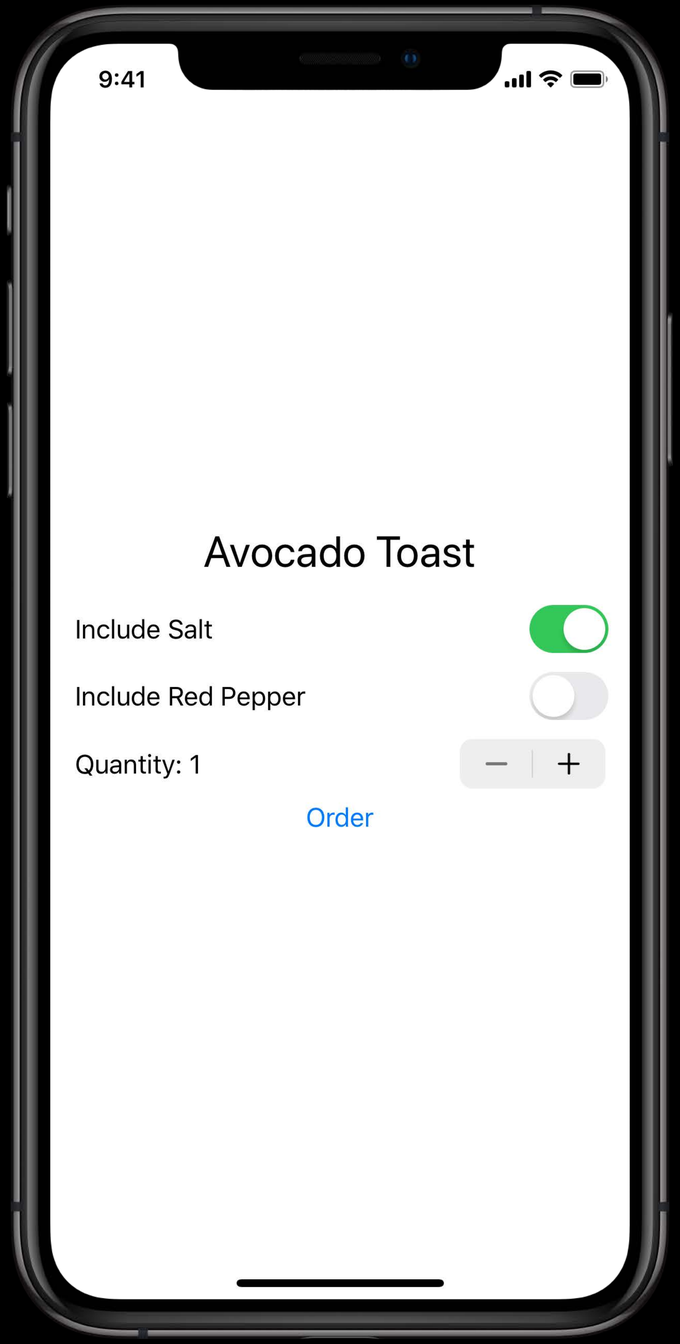
Navigating your app

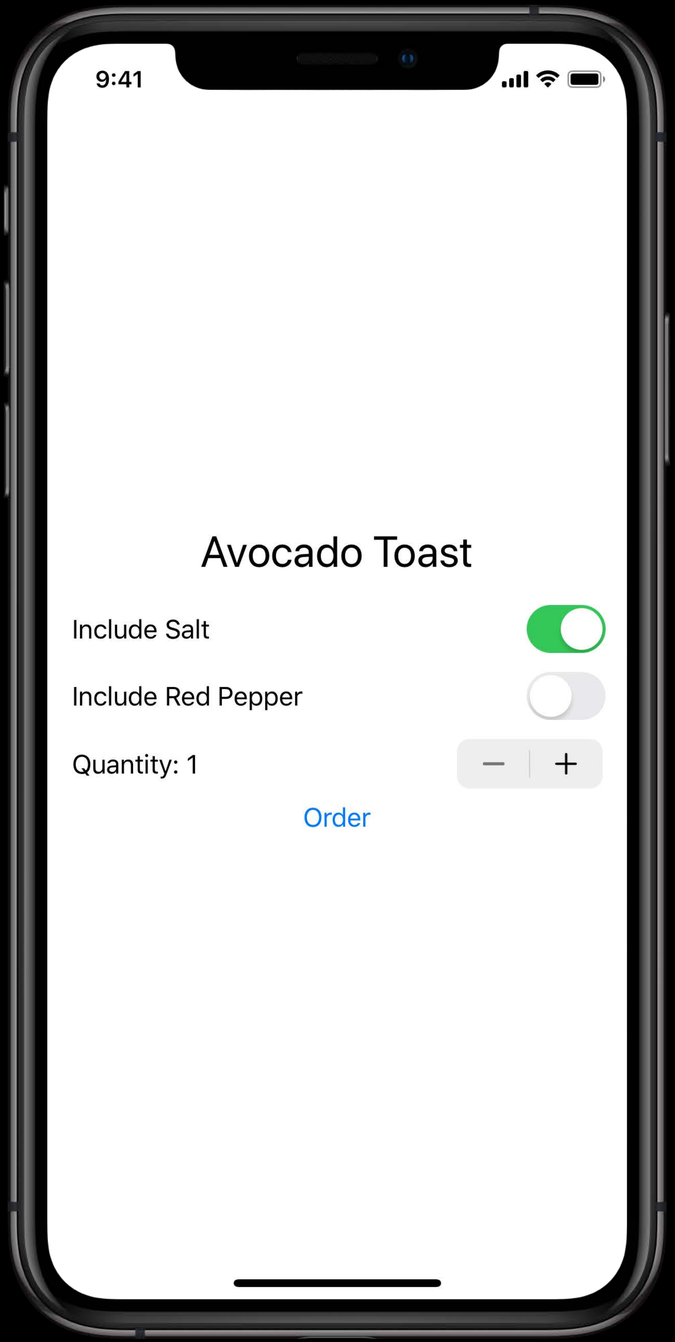
Getting started: Views and modifiers

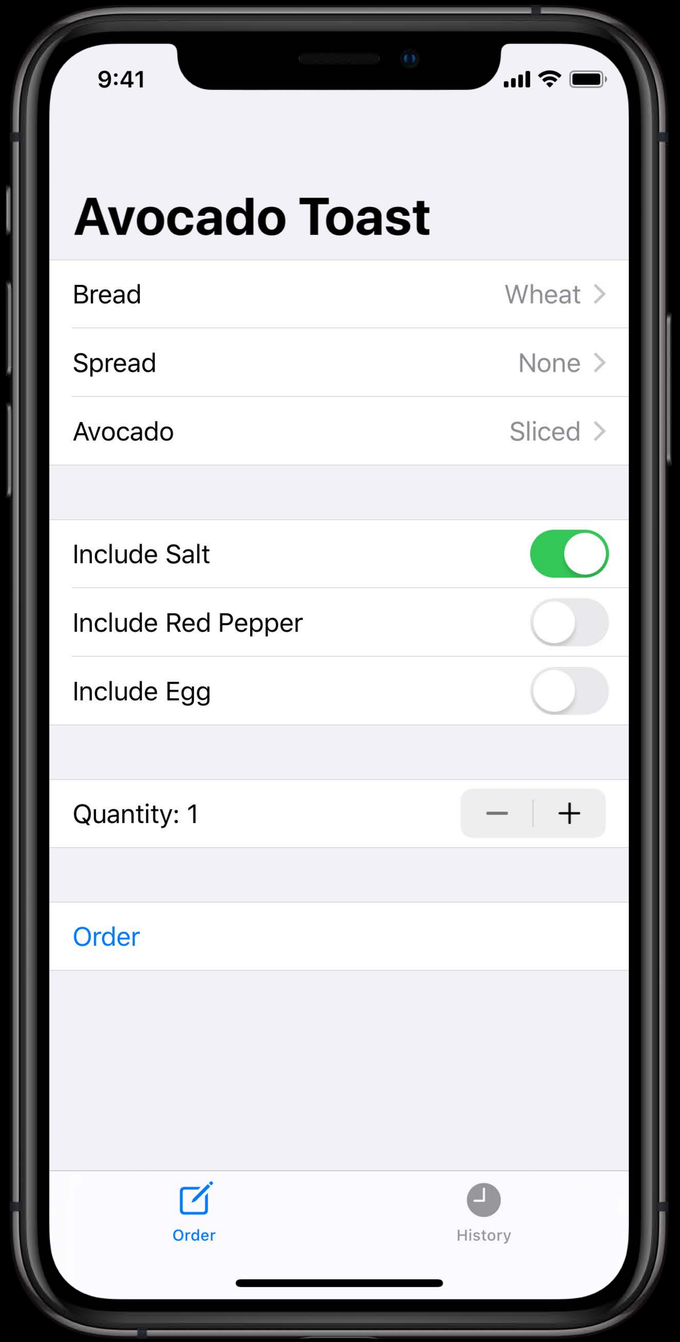
Building custom views

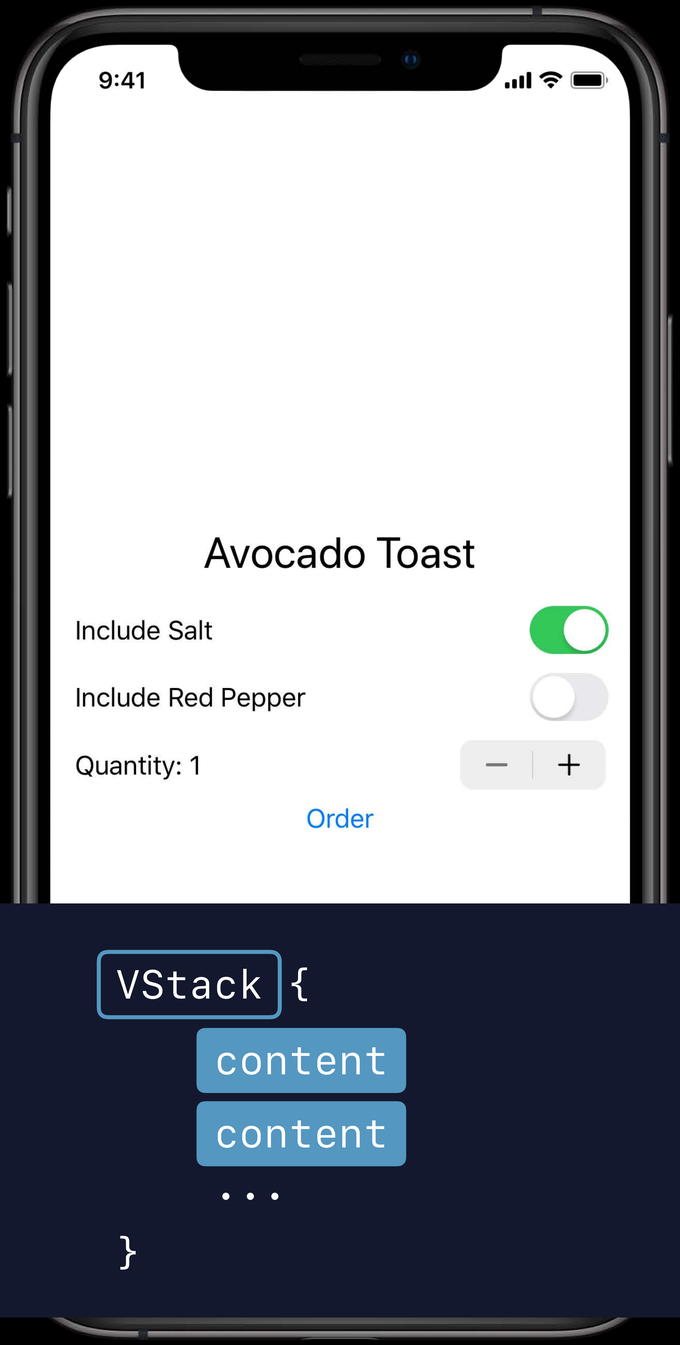
Composing controls

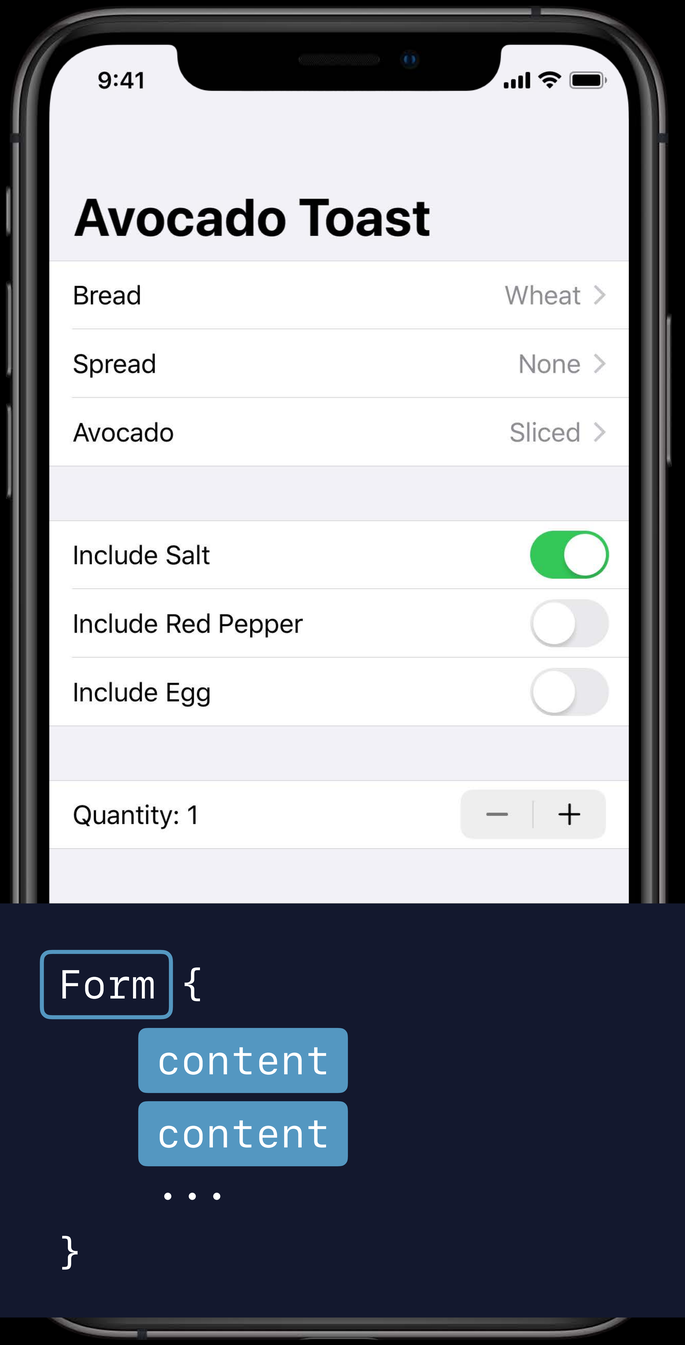
Navigating your app


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Avocado Toast
Bread
Wheat
Spread
None
Avocado
Sliced
Include Salt
Include Red Pepper
Include Egg
Quantity: 1
Order
Order
History

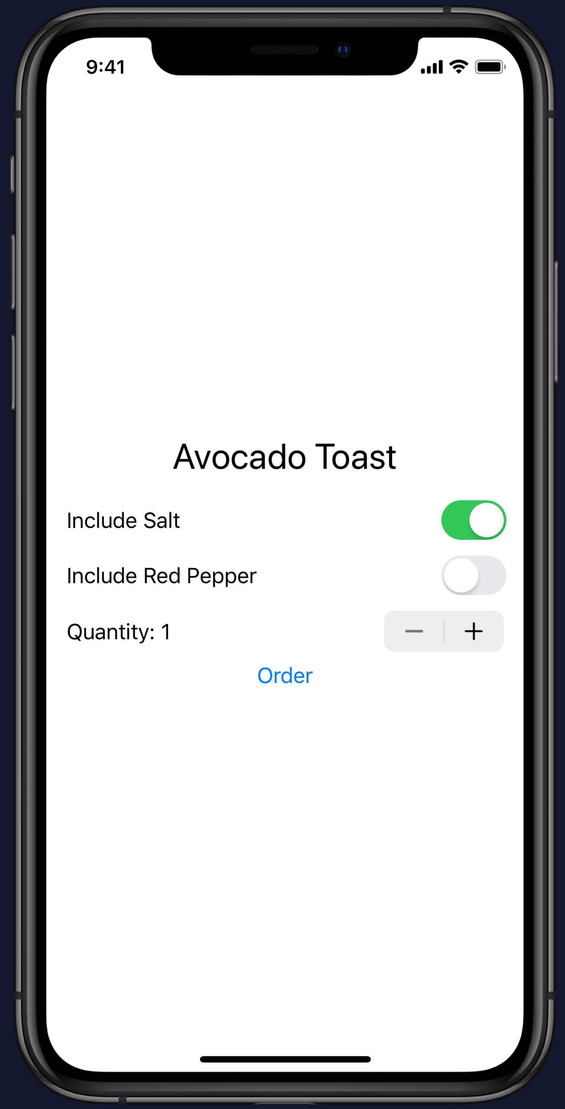




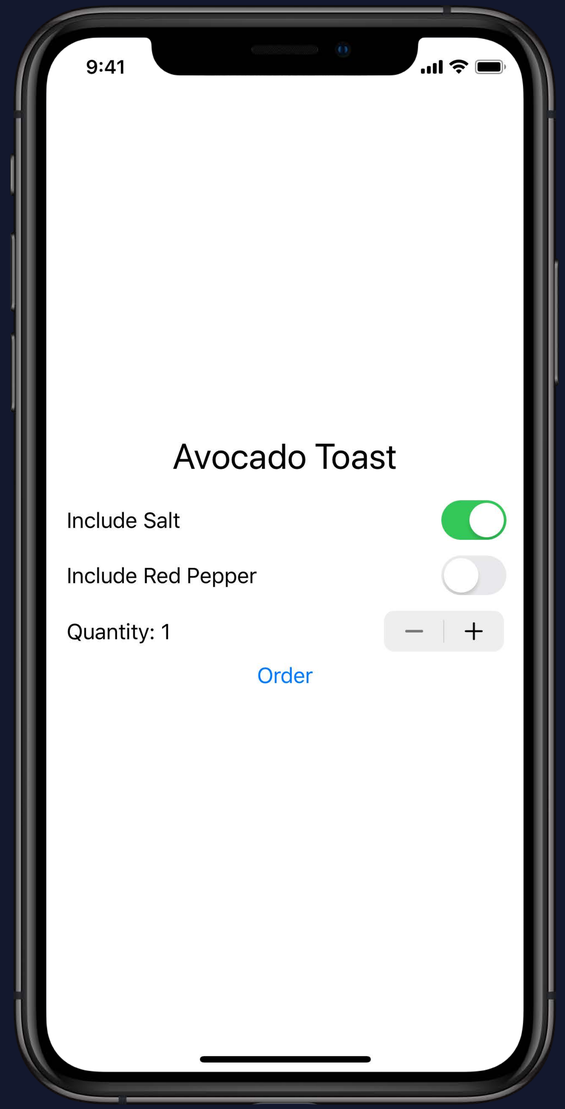

```
VStack {
    content
    content
    ...
}
```

```
Form {
    content
    content
    ...
}
```

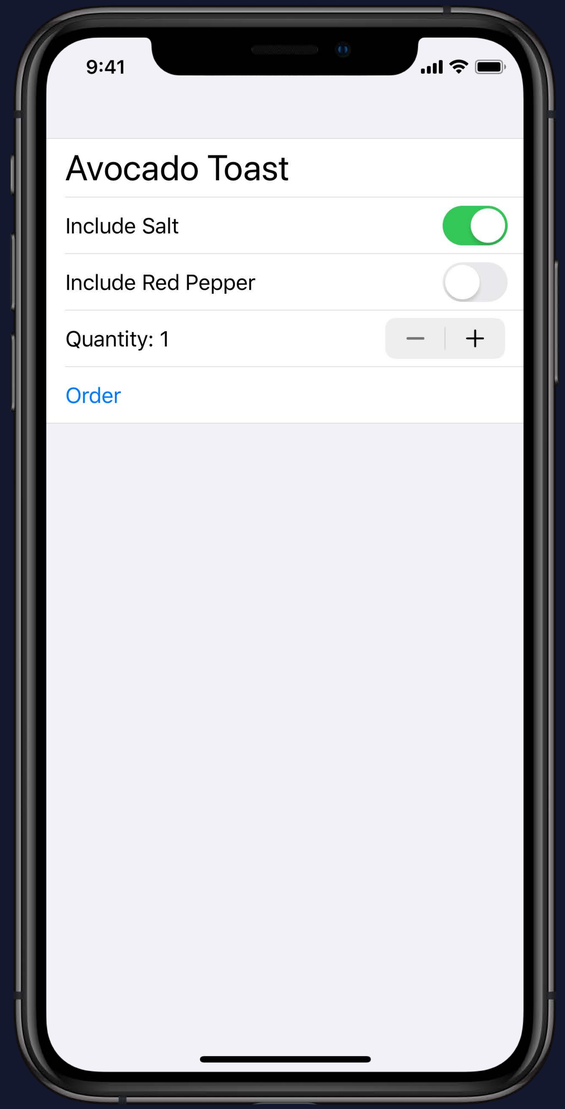
```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) { ••• }
    Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
    Stepper(value: $order.quantity, in: 1...10) { ••• }

    Button(action: submitOrder) { Text("Order") }
}
```



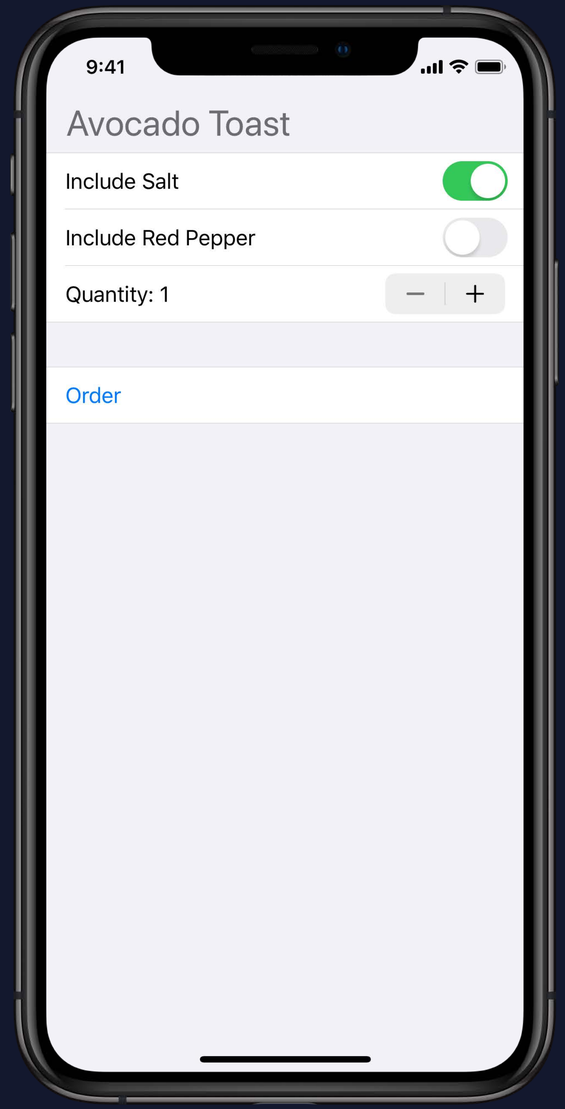
```
VStack {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) { ... }
    Toggle(isOn: $order.includeRedPepperFlakes) { ... }
    Stepper(value: $order.quantity, in: 1...10) { ... }

    Button(action: submitOrder) { Text("Order") }
}
```

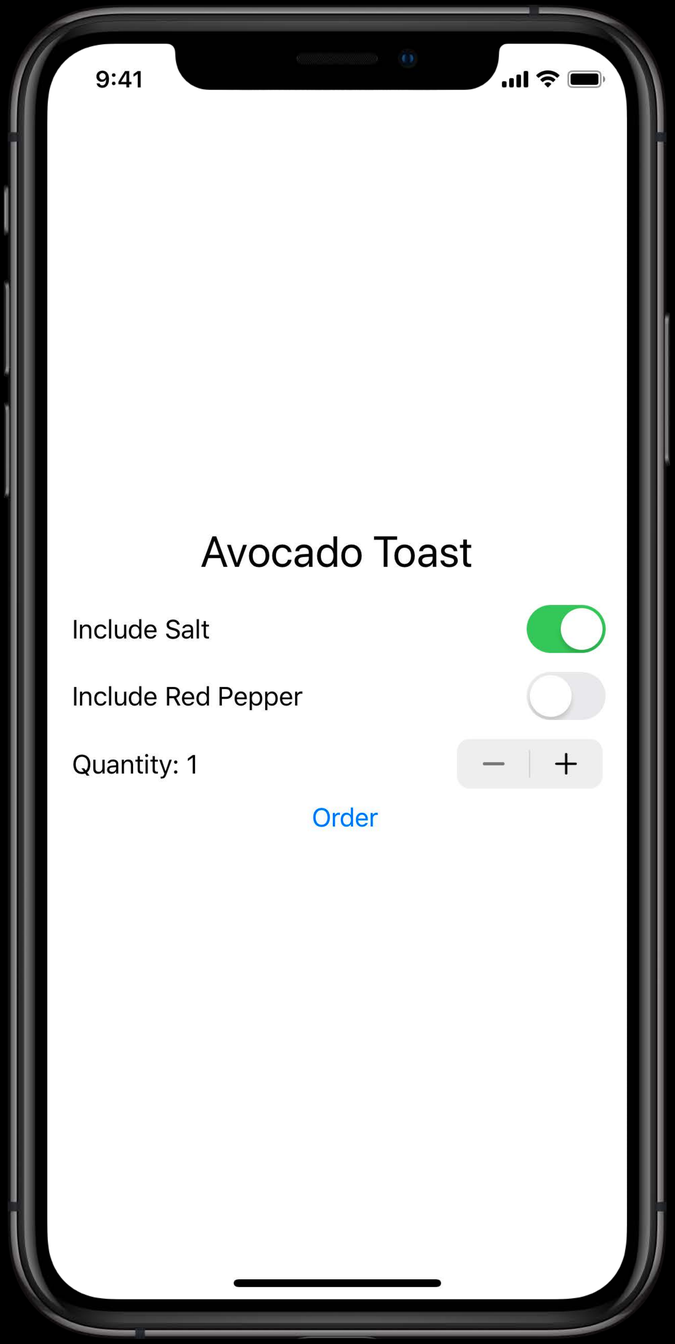


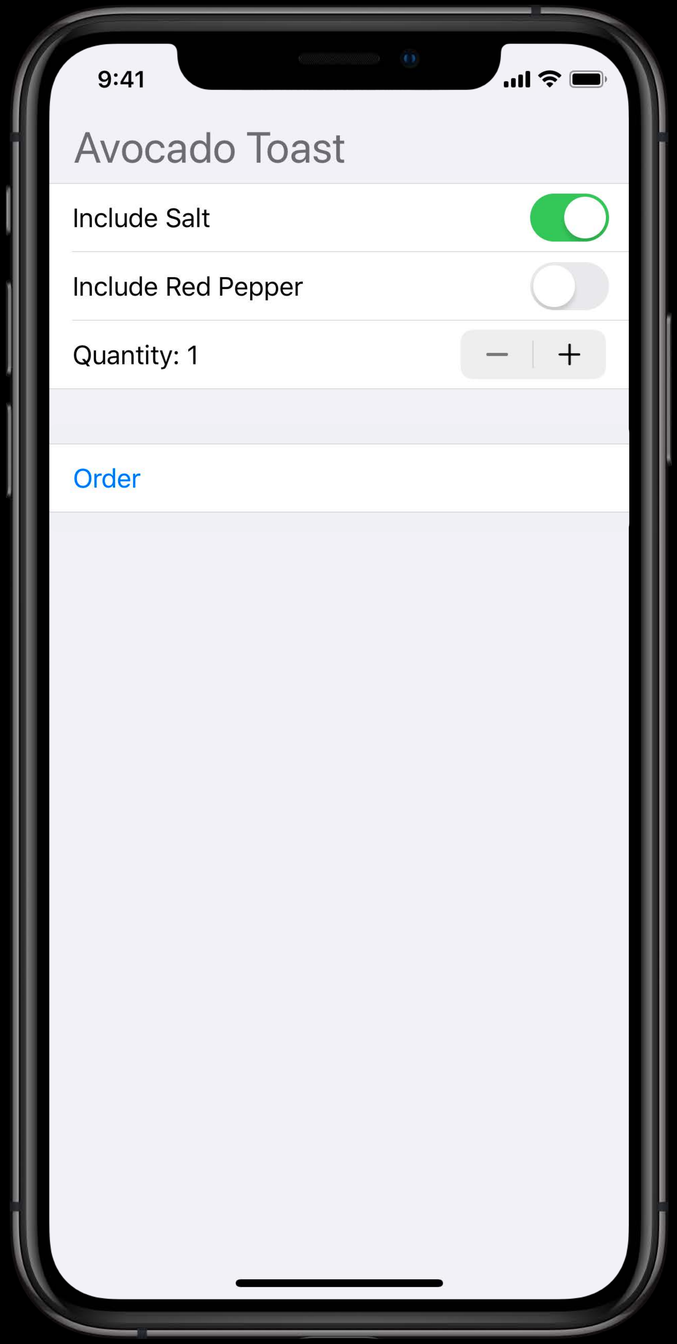
```
Form {
    Text("Avocado Toast").font(.title)

    Toggle(isOn: $order.includeSalt) { ••• }
    Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
    Stepper(value: $order.quantity, in: 1...10) { ••• }

    Button(action: submitOrder) { Text("Order") }
}
```

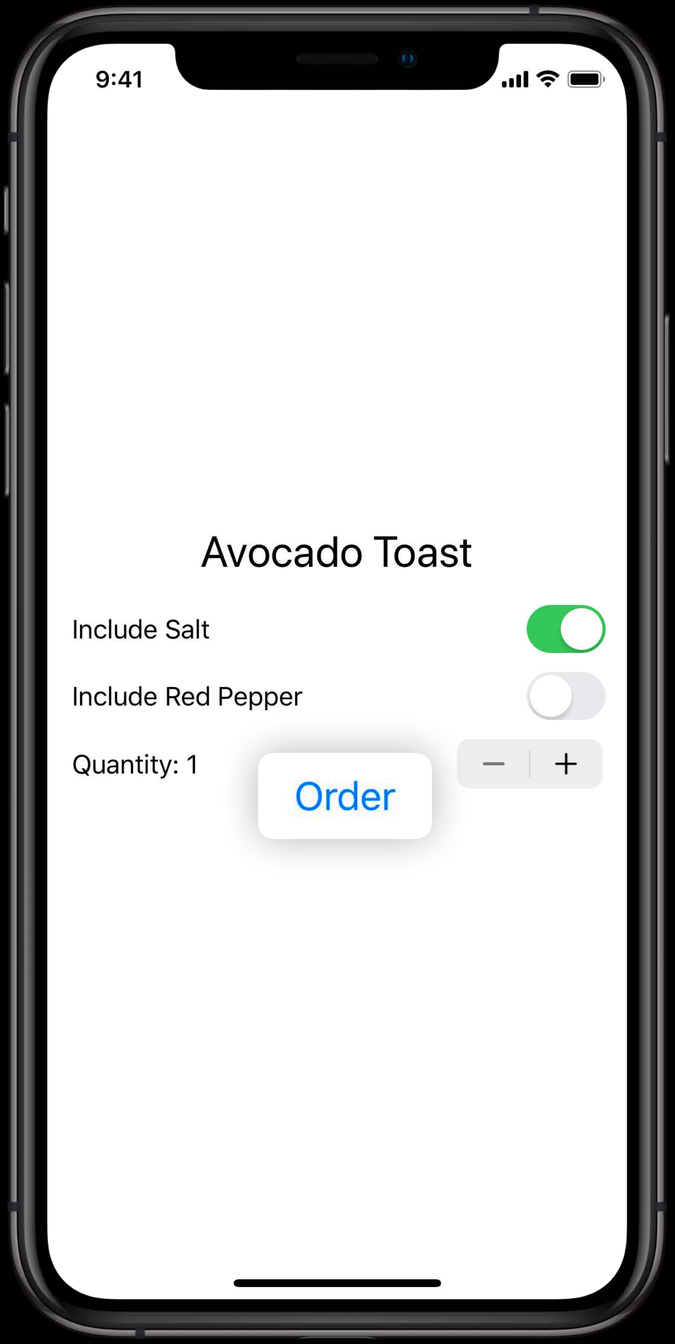


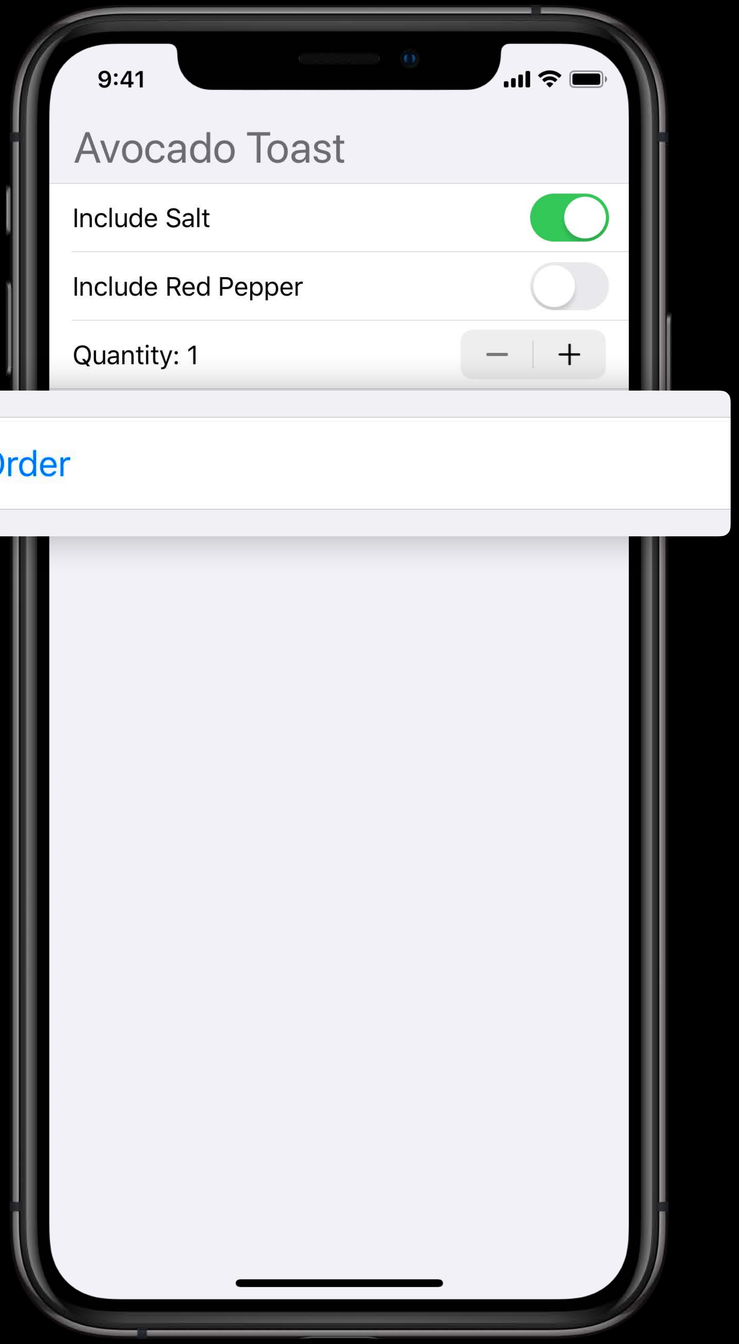
```
Form {
    Section(header: Text("Avocado Toast").font(.title)) {
        Toggle(isOn: $order.includeSalt) { ••• }
        Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
        Stepper(value: $order.quantity, in: 1...10) { ••• }
    }

    Section {
        Button(action: submitOrder) { Text("Order") }
    }
}
```

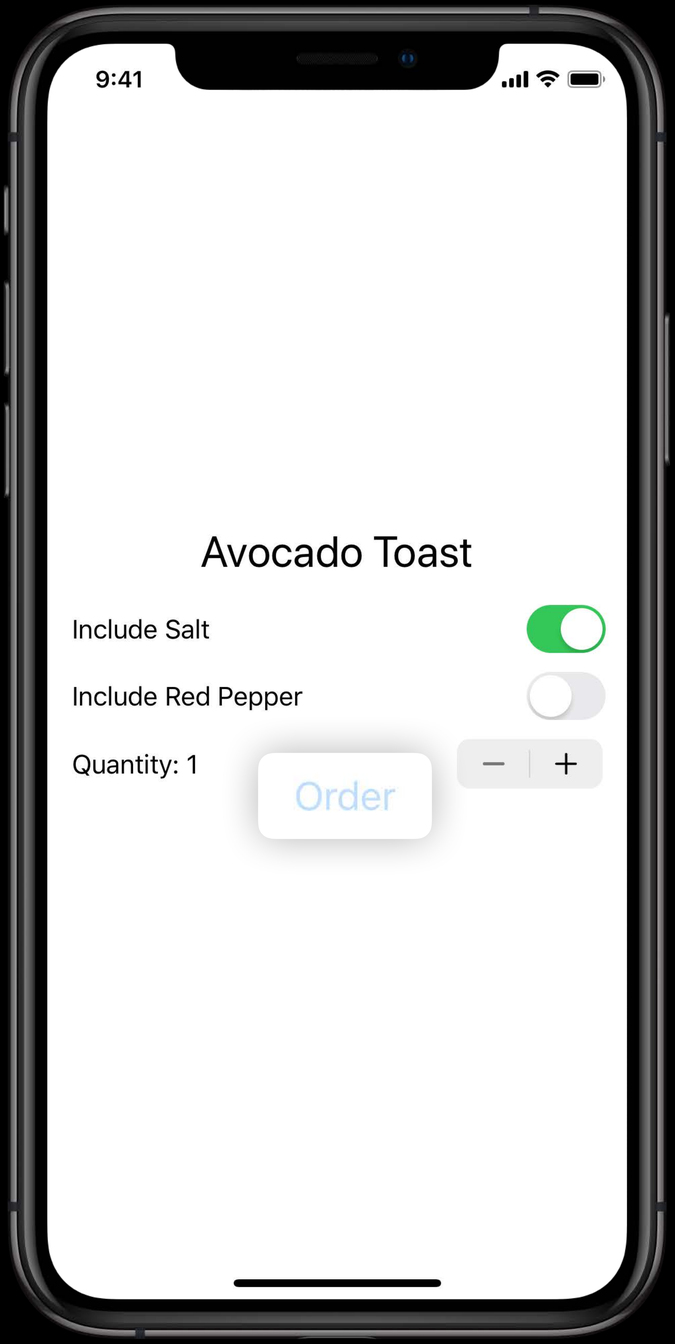


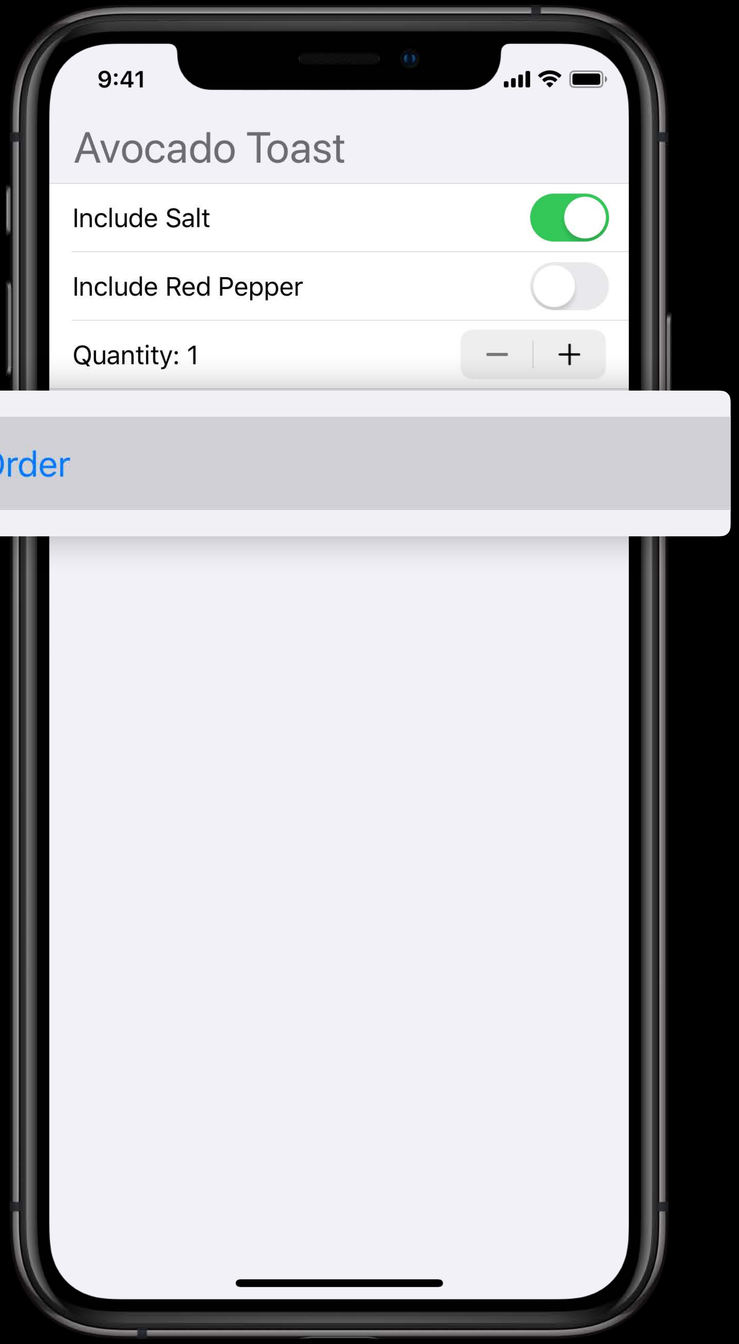

9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order

Button

```
Button(action: submitOrder) {
    Text("Order")
}
```

Button

```
Button(action: submitOrder) {
    Text("Order")
}
```

Button

```
Button(action: submitOrder) {
    Image("Order Toast")
}
```

Button

```
Button(action: submitOrder) {
    VStack {
        Image("Toast")
        Text("Order")
    }
}
```

Button

```
Button(action: action ) {
    label
}
```

Button

```
Button(action: action ) {
    label
}
```

```
public struct Button<Label : View> : View {
    public init(
        action: @escaping () -> Void,
        @ViewBuilder label: () -> Label
    )
}
```

API

Button


Close
Calendars
Create New Account
Connect
Reset
Start
Edit
Add a new episode...
Sign In
Show Traffic
Done
iTunes
Stop
Share
Send
Log Caffeine
Show All
Connect
Snooze
Dismiss
Edit
End
OK
Edit
OK
Don't Send
Cancel
OK
Restore
Start Outdoor Run Workout
Buy
Cancel
Check in
Updates
Groups
Edit
Empty Trash
Calendars
Computer
List View
Edit
Back
Delete
New Folder
Move to Music
Cancel
Go
OPEN
Sign In
Get Started

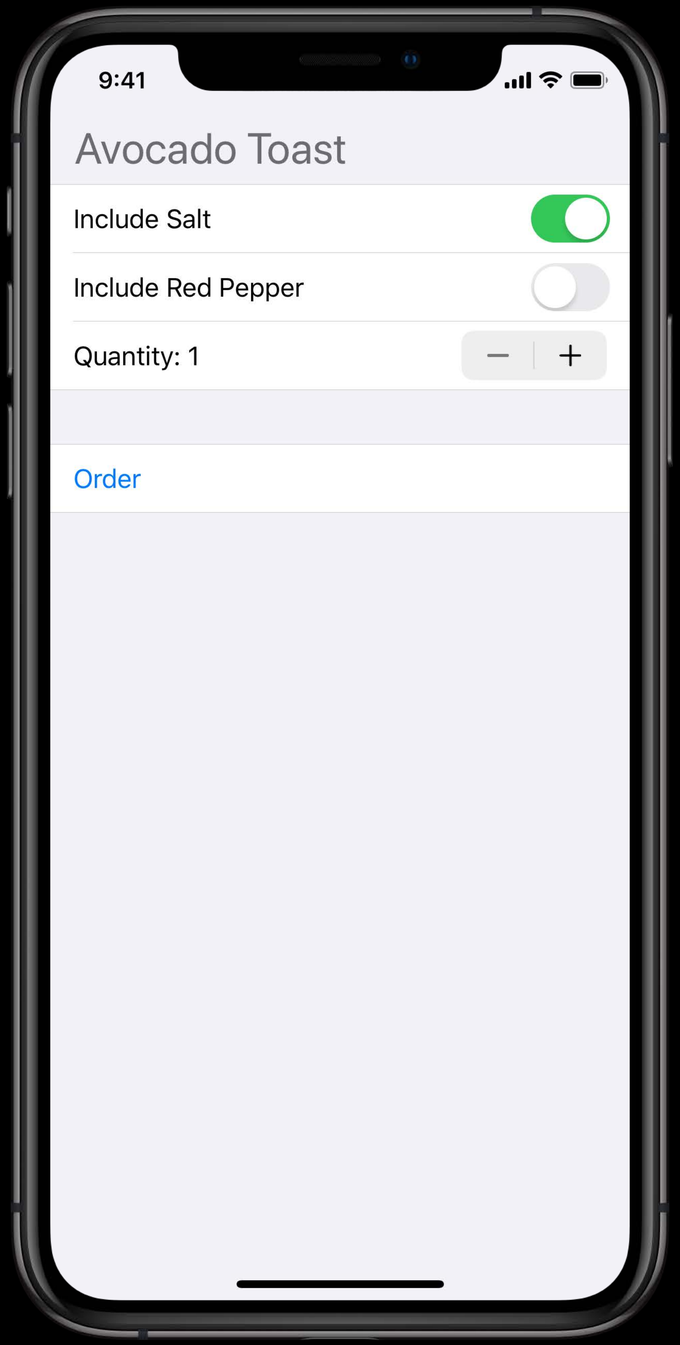
Adaptive Controls

Describe their purpose, not visuals

Smarter default behavior

Highly reusable components

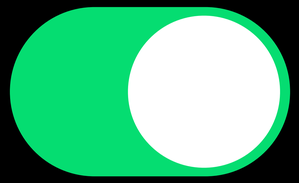
More powerful customization


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order

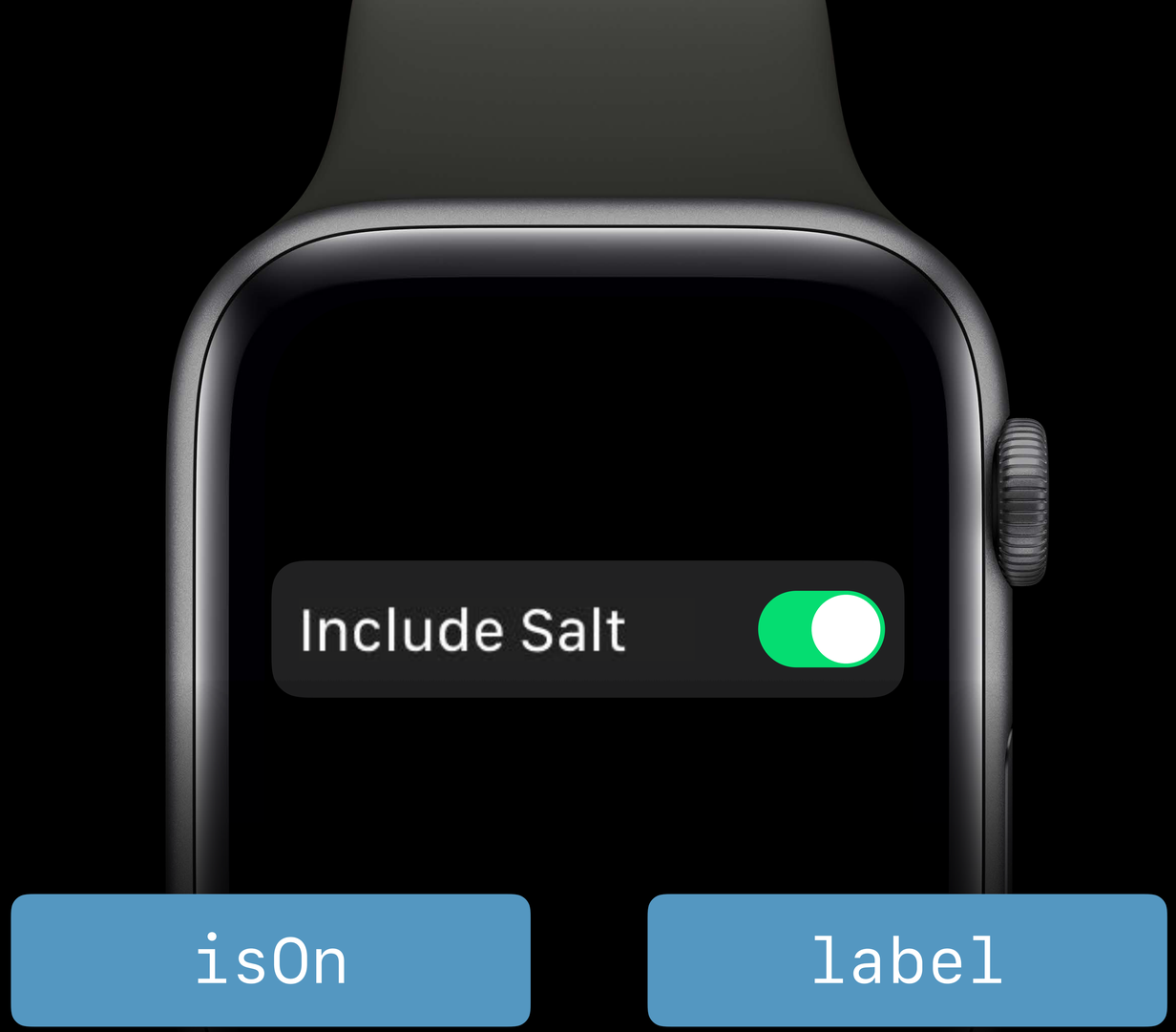
Avocado Toast
9:41
Salt
Red Pepper
Order

Toggle



Toggle

Toggle



Toggle


```
Toggle(isOn: isOn ) {
    label
}
```

Toggle


```
Toggle(isOn: $order.includeSalt) {
    Text("Include Salt")
}
```

Toggle

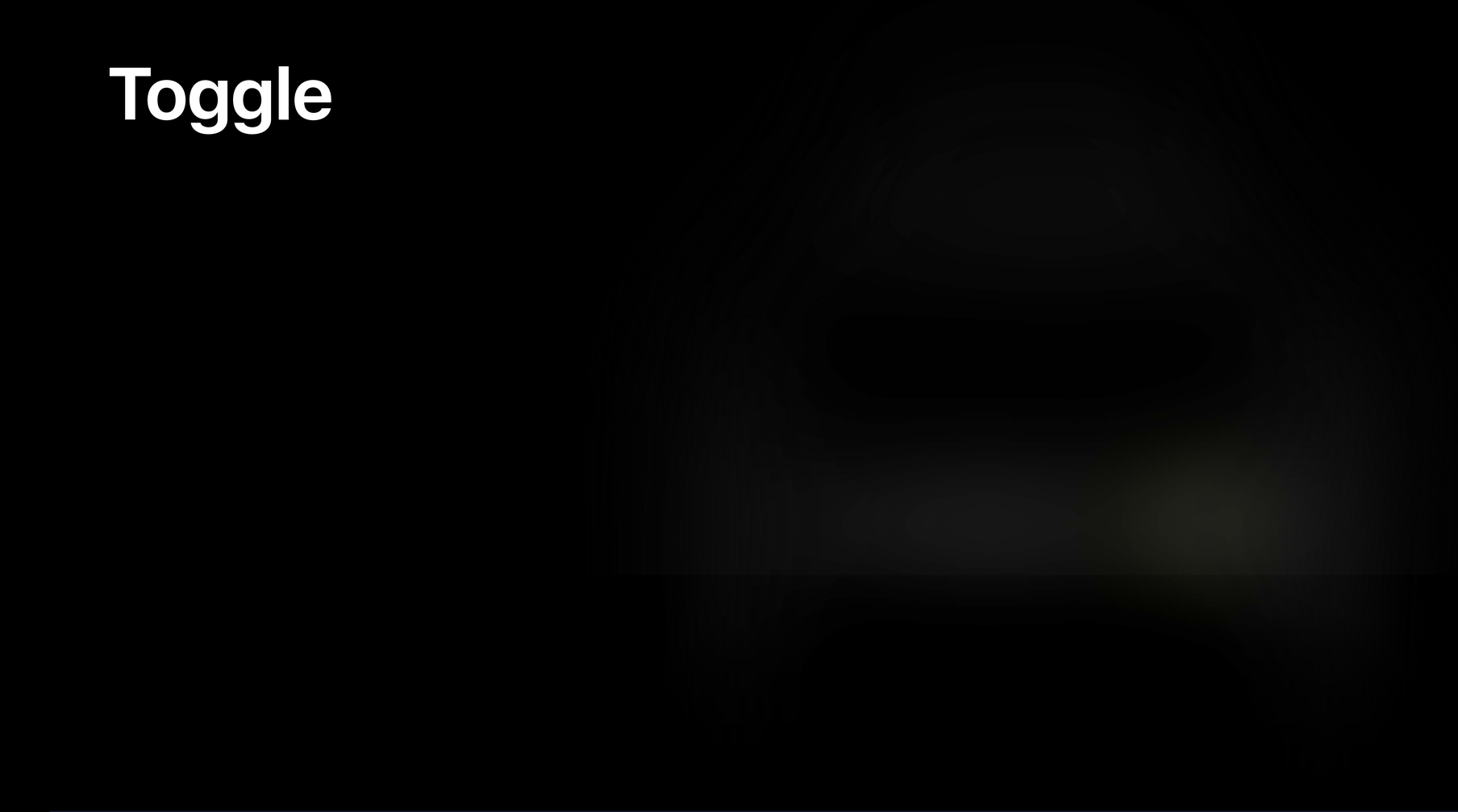
Binding<Bool>

```
Toggle(isOn: $order.includeSalt) {
    Text("Include Salt")
}
```

Toggle

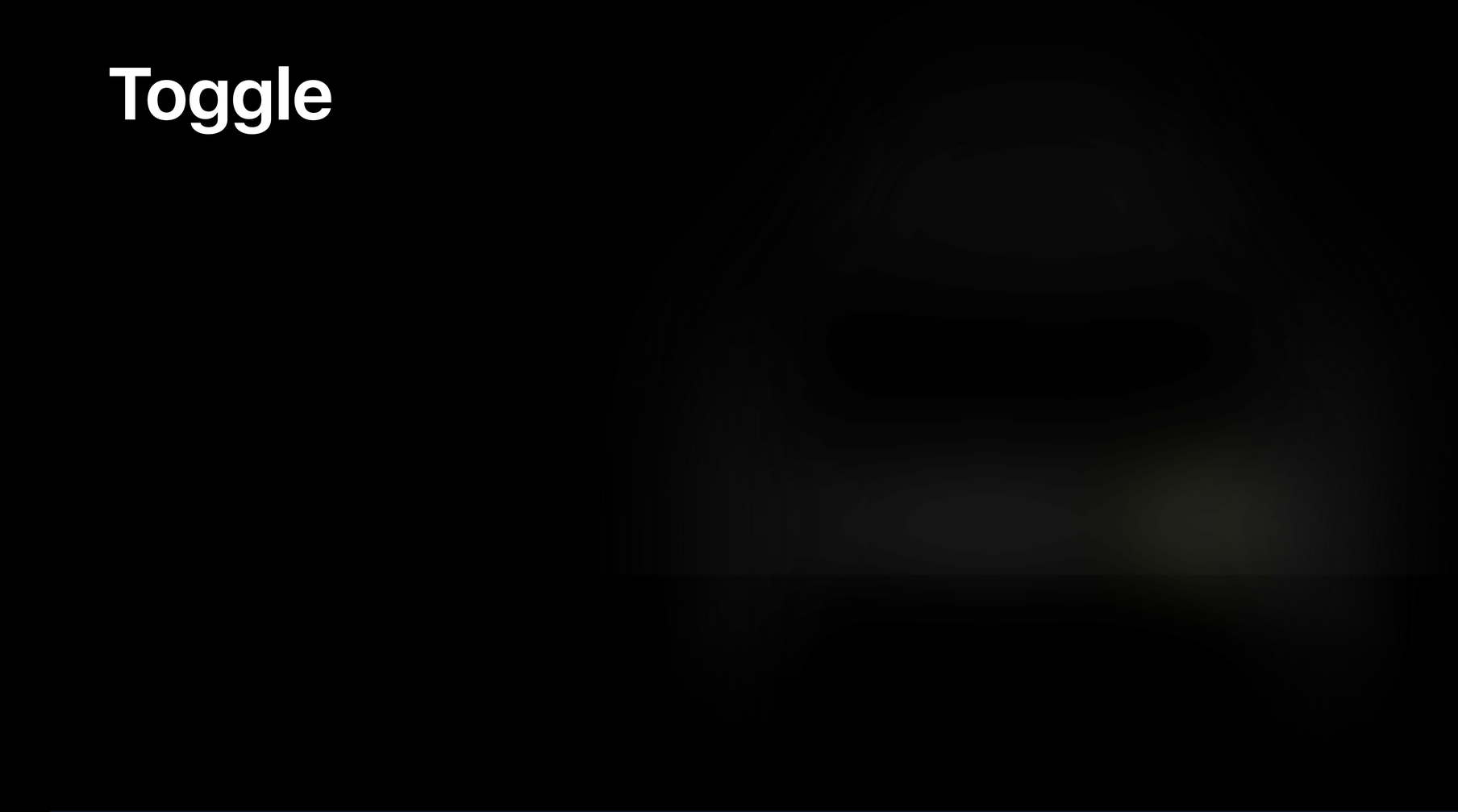

```
Toggle(isOn: $order.includeSalt) {
    Text("Include Salt")
}
```

Toggle



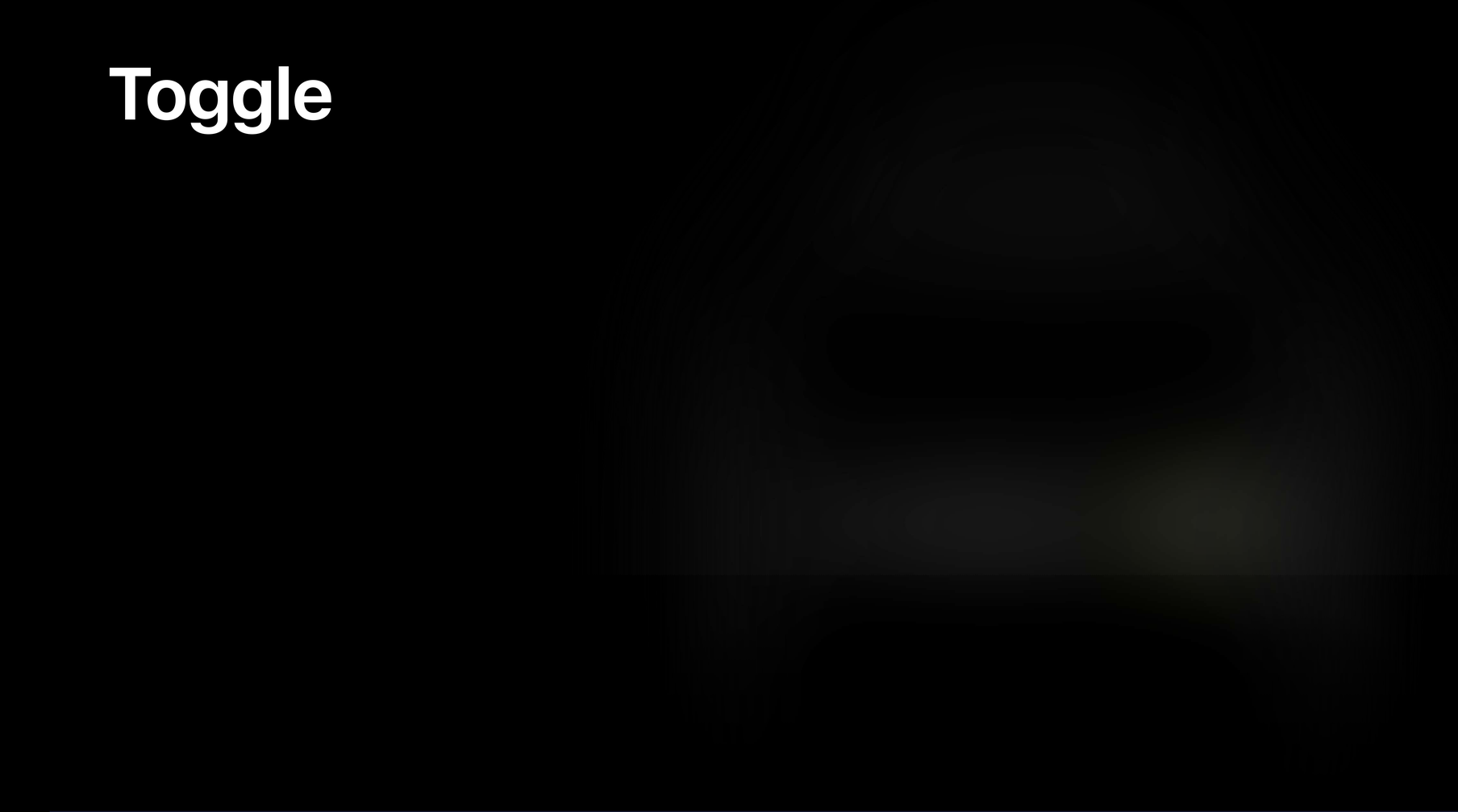
```
Toggle(isOn: $order.includeSalt) {
    Text("Include Salt")
}
```

Toggle



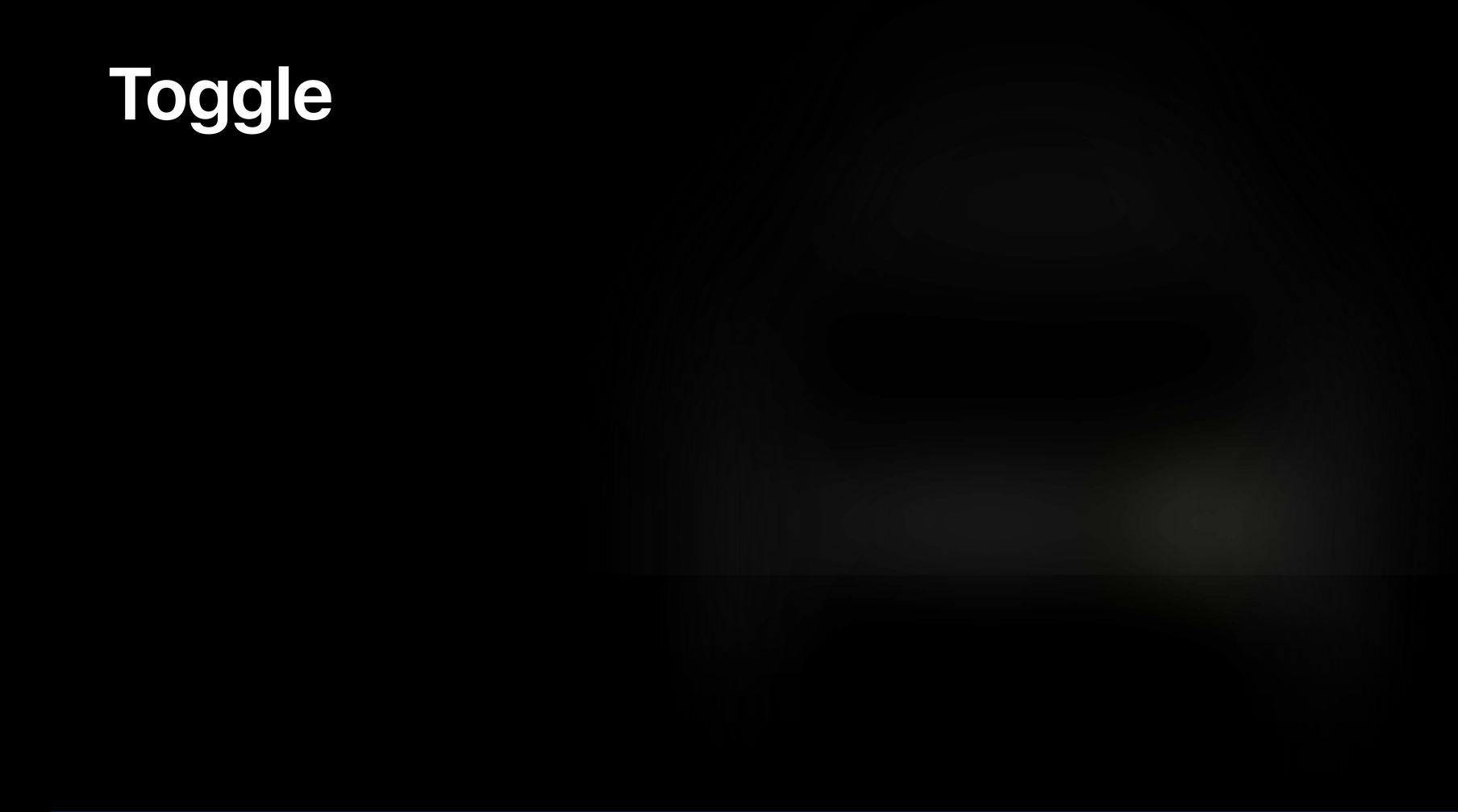
```
Toggle(isOn: $order.includeSalt) {
    Text("Include Salt")
}
```

Toggle



```
Toggle(isOn: $order.includeSalt) {
    Text("Include Salt")
}
```

Toggle



```
Toggle(isOn: $order.includeSalt) {
    Text("Include Salt")
}
```

Toggle

Include Salt, Switch Button, On
Double Tap to Toggle Setting

```
Toggle(isOn: $order.includeSalt) {
    Text("Include Salt")
}
```

Toggle

Include Salt, Switch Button, On
Double Tap to Toggle Setting

```
Toggle(isOn: $order.includeSalt) {
    Image("Salt", label: Text("Include Salt"))
}
```

Toggle

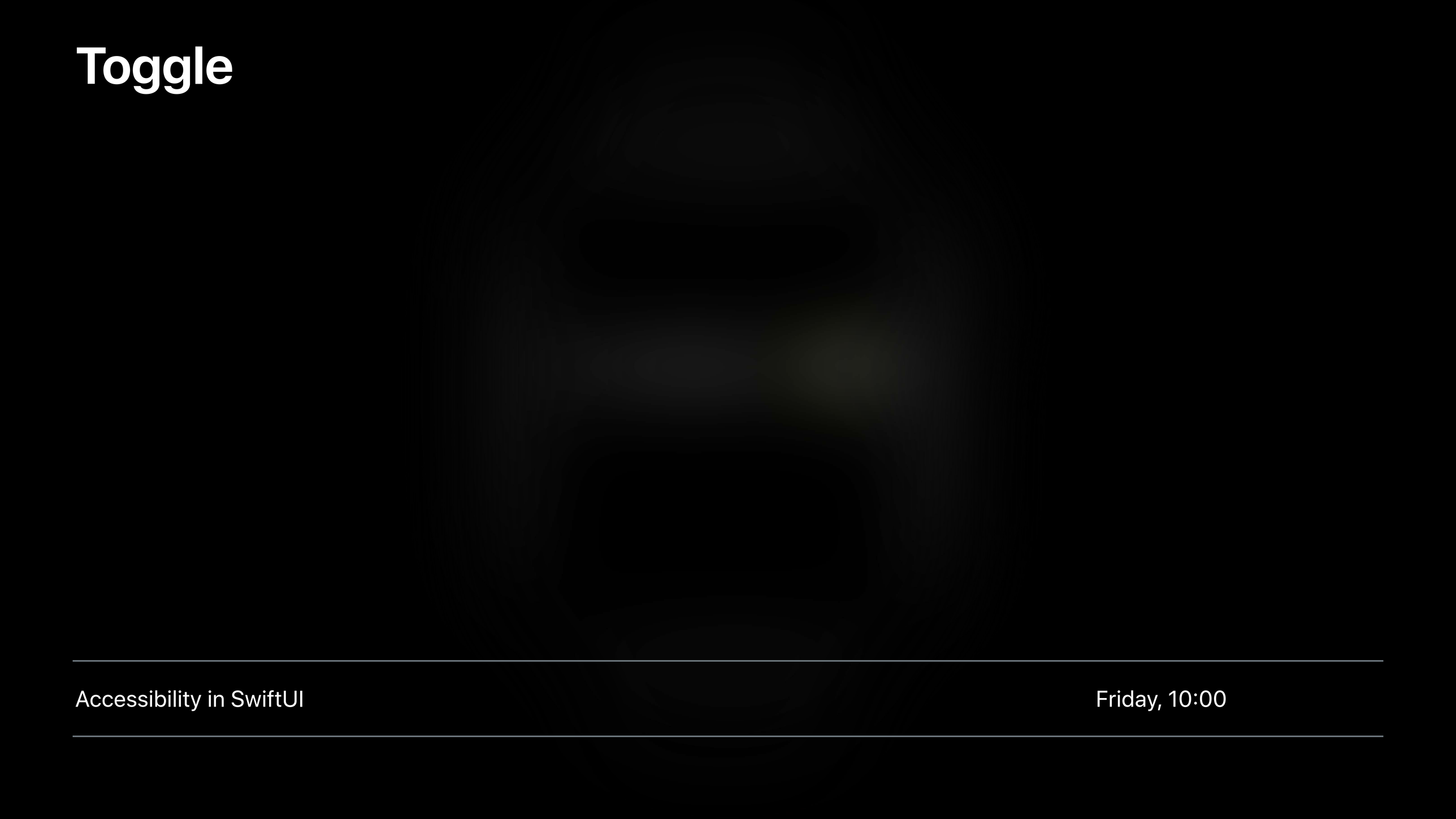
Include Salt, Switch Button, On
Double Tap to Toggle Setting

```
Toggle(isOn: $order.includeSalt) {
    CustomShape().accessibility(label: Text("Include Salt"))
}
```

Toggle

```
Toggle(isOn: $order.includeSalt) {
    CustomShape().accessibility(label: Text("Include Salt"))
}
```

Toggle



Accessibility in SwiftUI

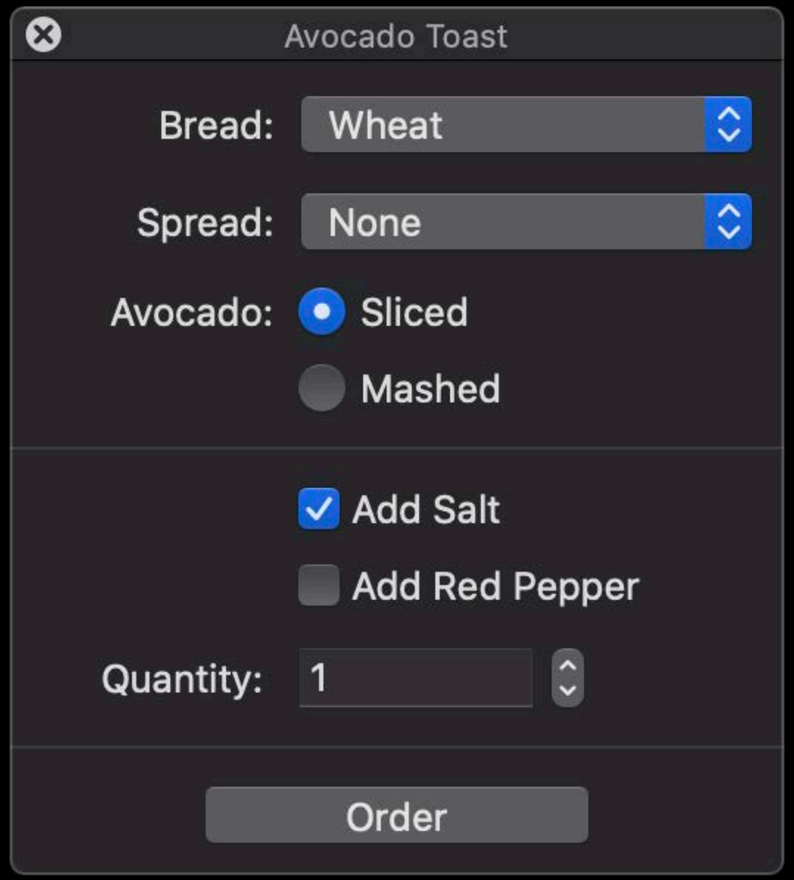
Friday, 10:00


9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order
Avocado Toast
9:41
Salt
Red Pepper
Order

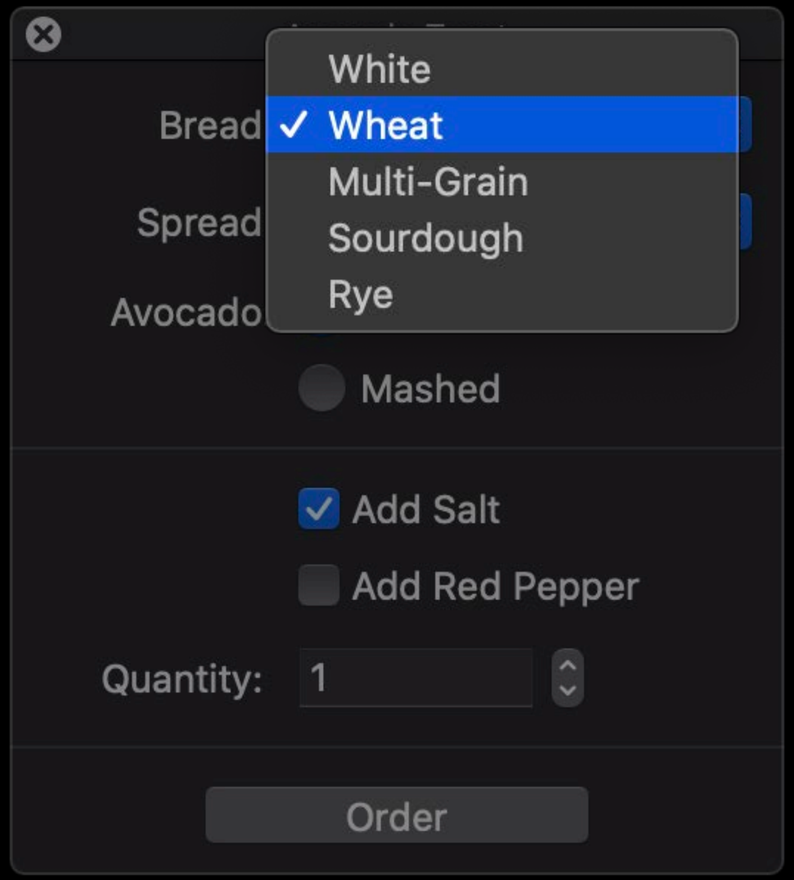

9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order
Avocado Toast
9:41
Salt
Red Pepper
Order


Avocado Toast
Bread: Wheat
Spread: None
Avocado: Sliced
Mashed
Add Salt
Add Red Pepper
Quantity: 1
Order
MacBook Pro
9:41
Avocado Toast
Include Salt
Include Red Pepper
Quantity: 1
Order
Avocado Toast 9:41
Salt
Red Pepper
Order

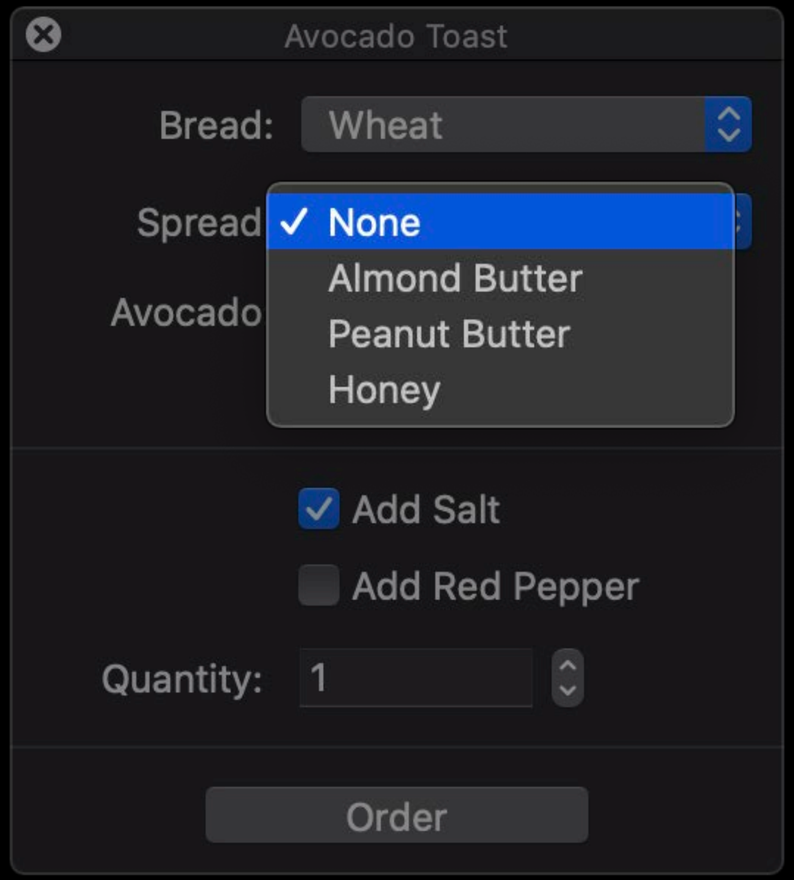
Picker


Avocado Toast
Bread:
Wheat
Spread:
None
Avocado:
Sliced
Mashed
Add Salt
Add Red Pepper
Quantity:
1
Order

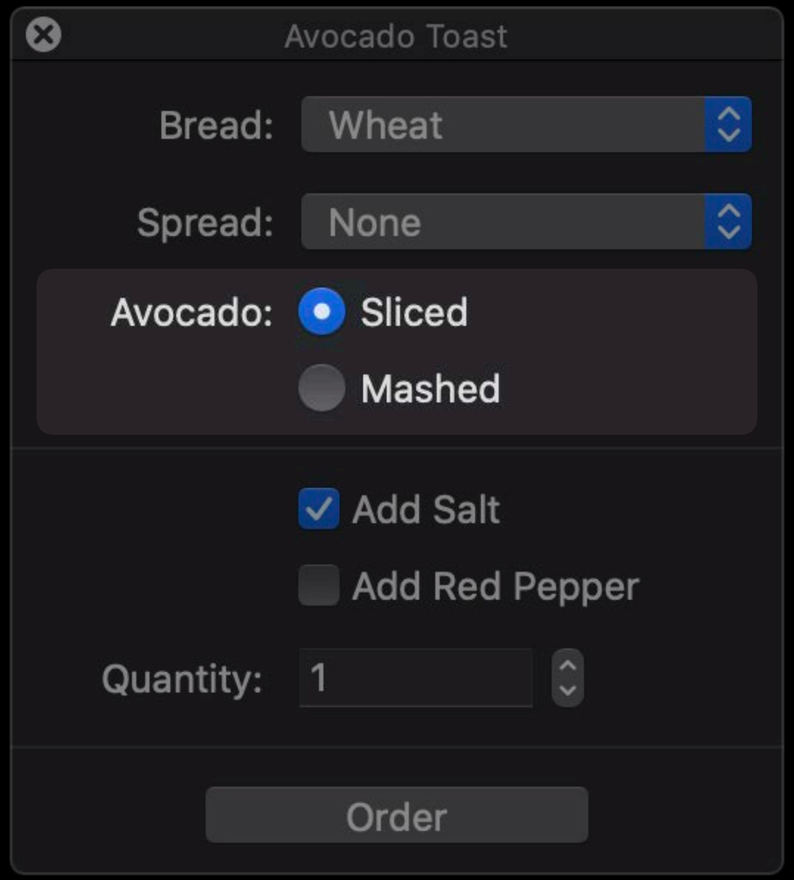
Picker


White
Wheat
Multi-Grain
Sourdough
Rye
Bread
Spread
Avocado
Mashed
Add Salt
Add Red Pepper
Quantity:
1
Order

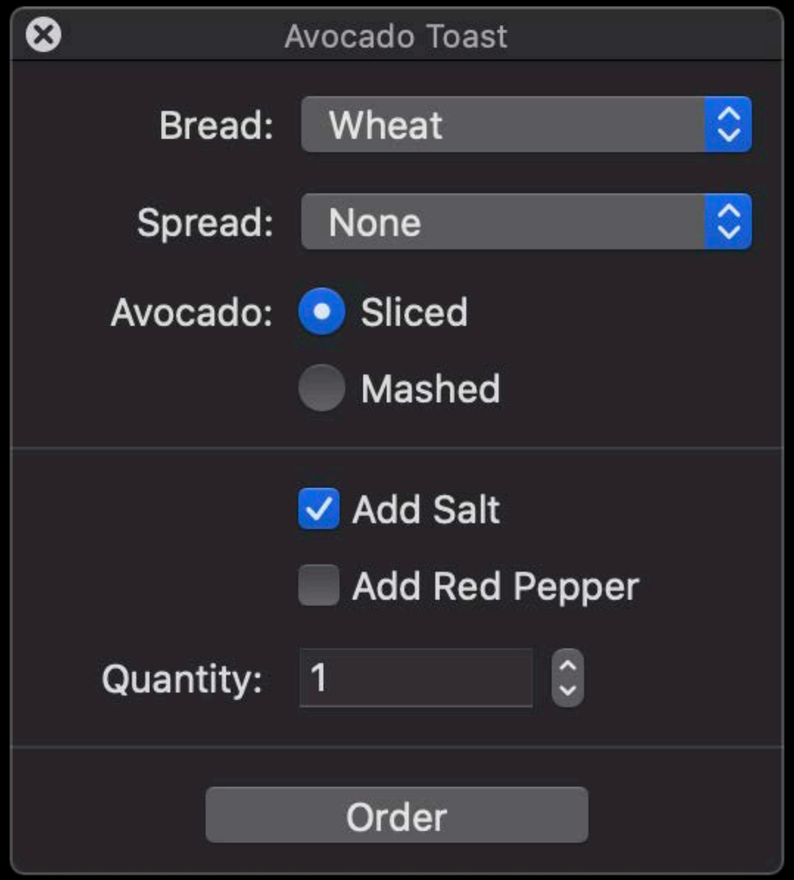
Picker


Avocado Toast
Bread:
Wheat
Spread
None
Almond Butter
Peanut Butter
Honey
Avocado
Add Salt
Add Red Pepper
Quantity:
1
Order

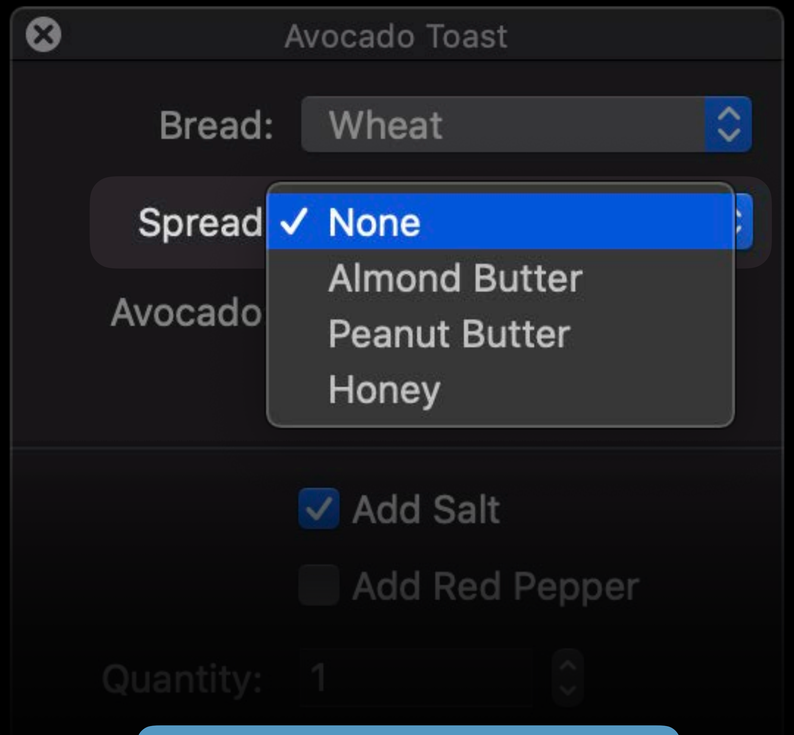
Picker


Avocado Toast
Bread:
Wheat
Spread:
None
Avocado:
Sliced
Mashed
Add Salt
Add Red Pepper
Quantity:
1
Order

Picker

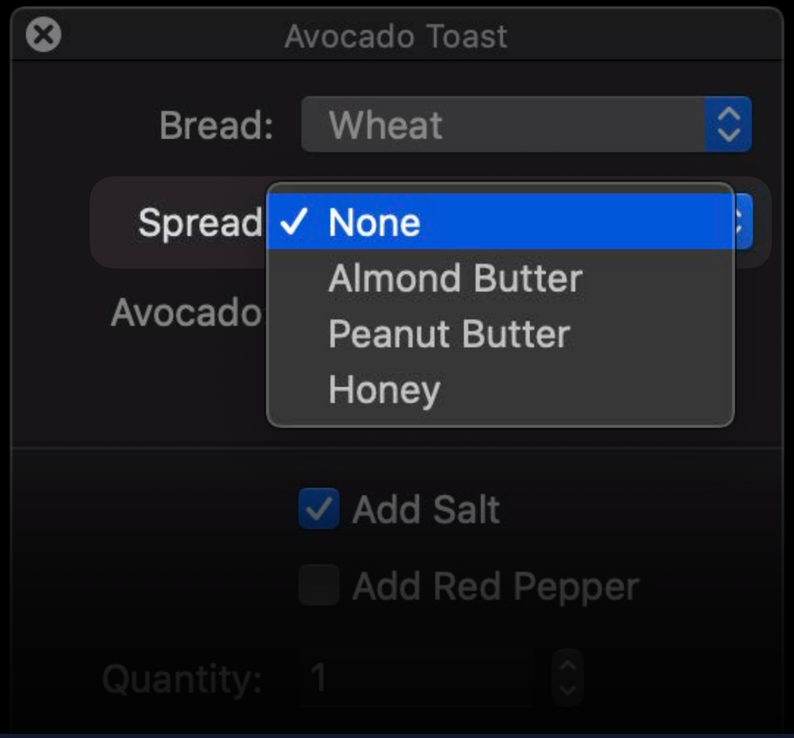

Avocado Toast
Bread:
Wheat
Spread:
None
Avocado:
Sliced
Mashed
Add Salt
Add Red Pepper
Quantity:
1
Order

Picker



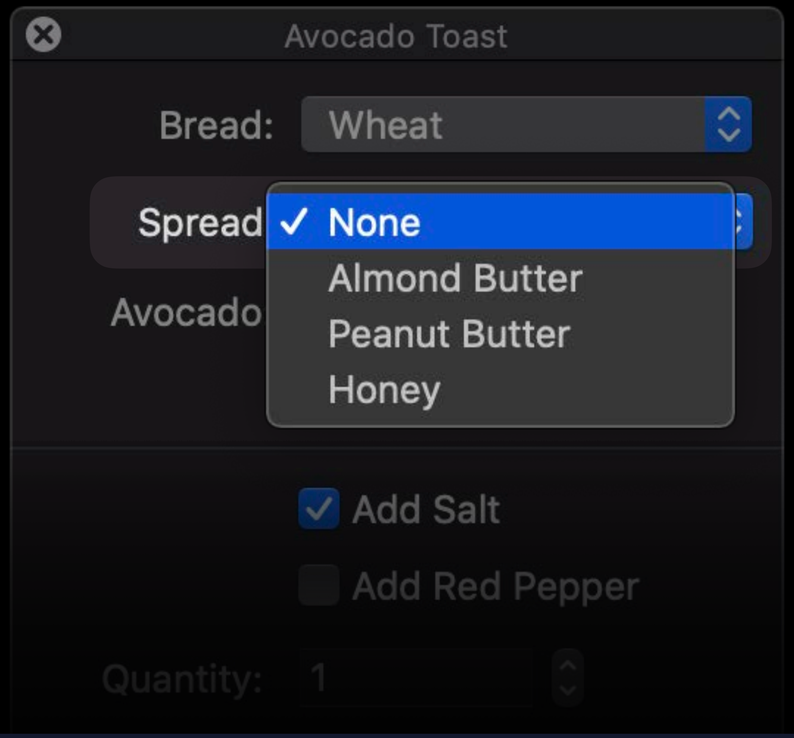
`options` `selection` `label`

Picker



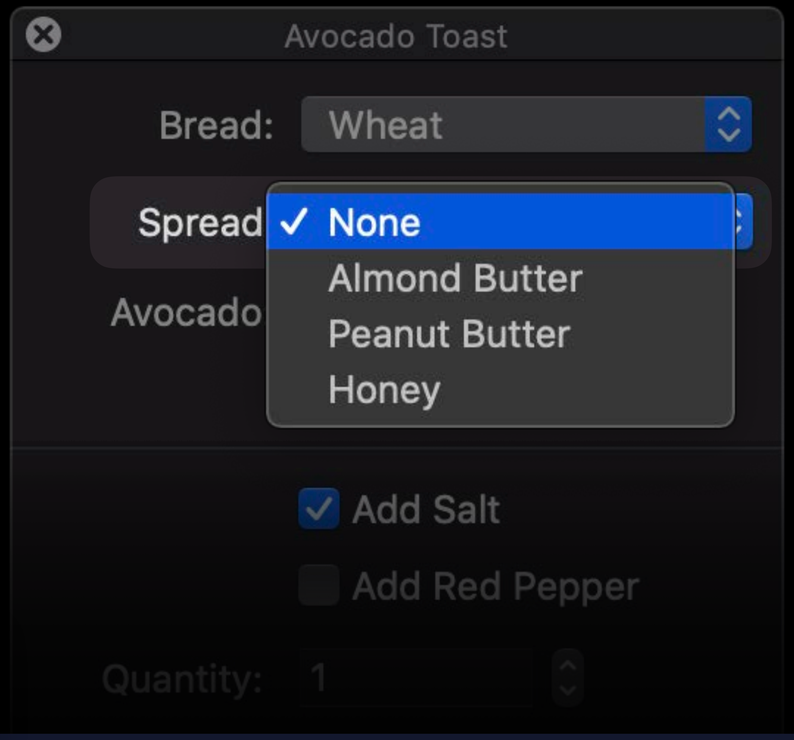

```
Picker(selection: selection , label: label ) {
    options
    options
}
```

Picker




```
Picker(selection: $order.spread, label: Text("Spread:")) {
    Text("None").tag(Spread.none)
    Text("Almond Butter").tag(Spread.almondButter)
    …
```

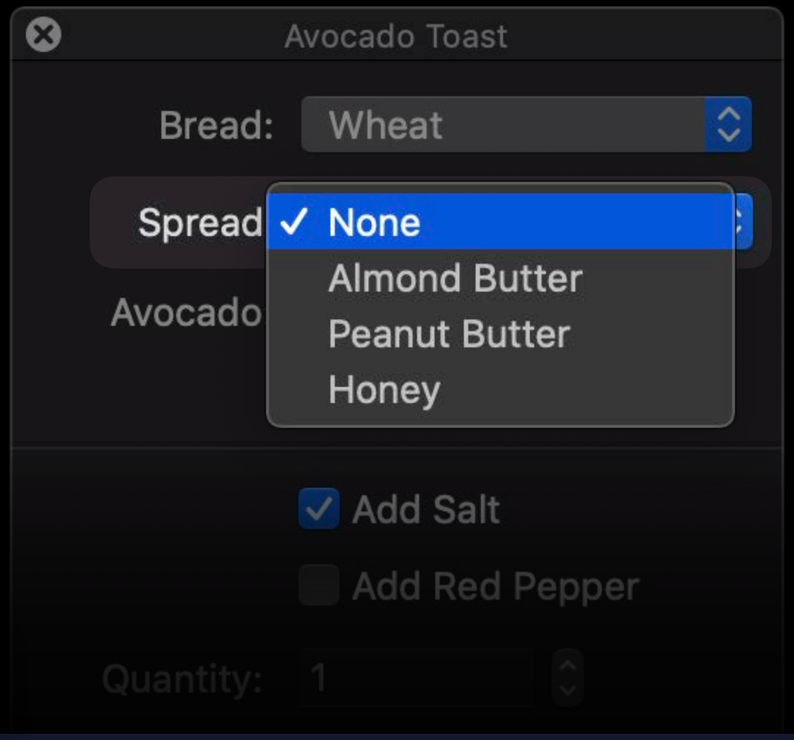
Picker



Binding<Spread>

```
Picker(selection: $order.spread, label: Text("Spread:")) {
    Text("None").tag(Spread.none)
    Text("Almond Butter").tag(Spread.almondButter)
    …
```

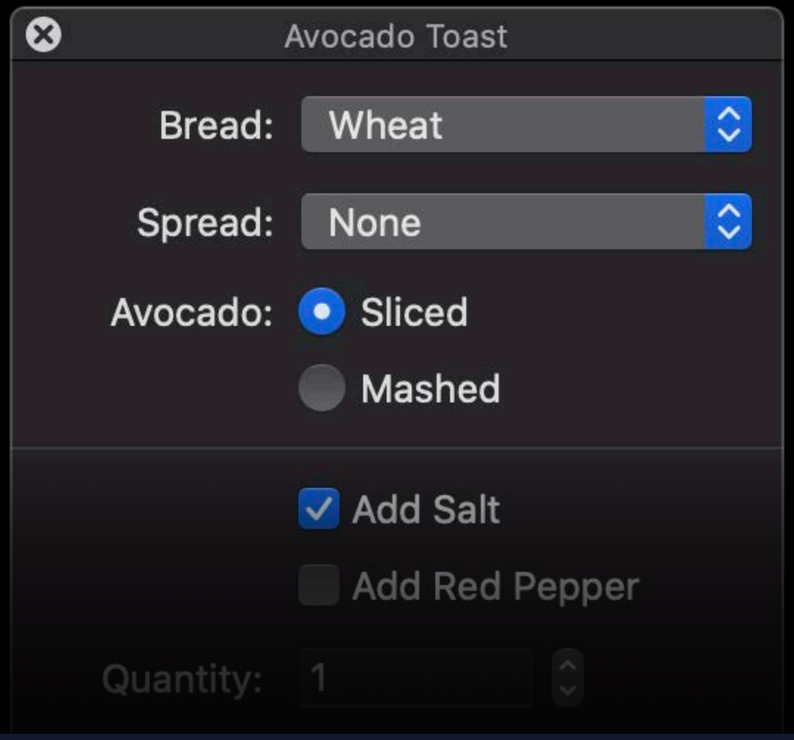
Picker



Binding<Spread>

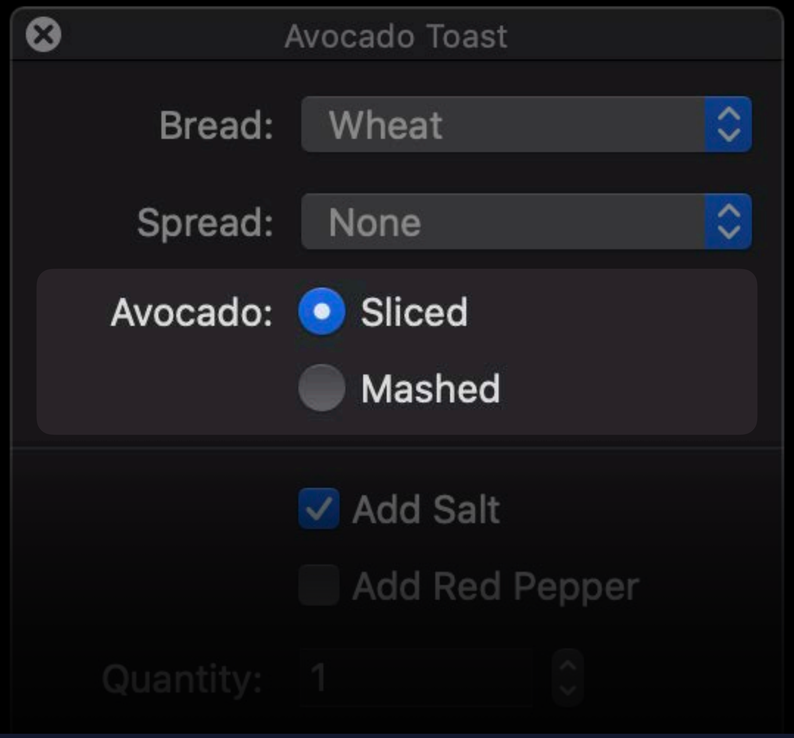
```
Picker(selection: $order.spread, label: Text("Spread:")) {
    Text("None").tag(Spread.none)
    Text("Almond Butter").tag(Spread.almondButter)
    …
```

Picker



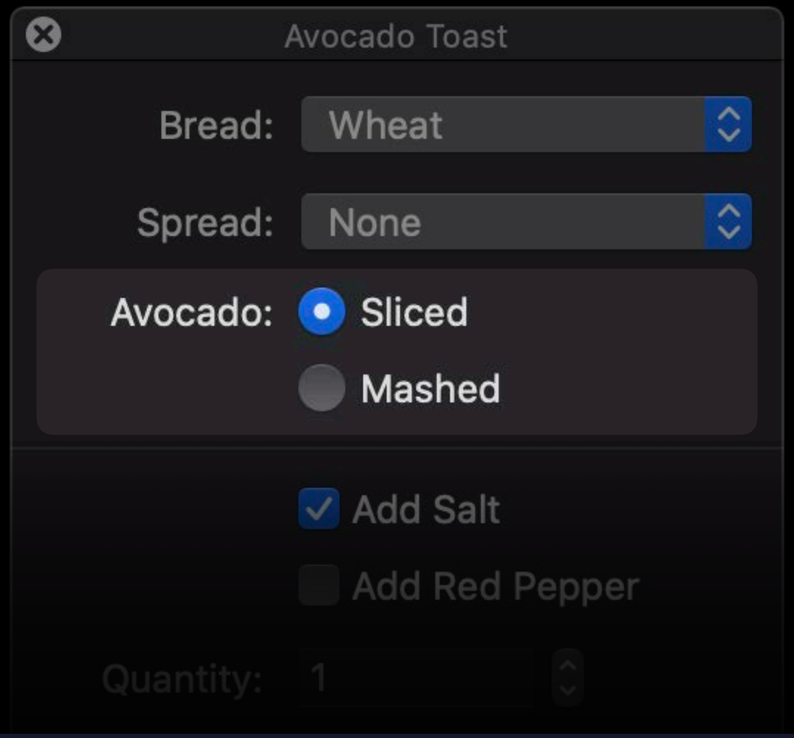

```
Picker(selection: $order.spread, label: Text("Spread:")) {
    Text("None").tag(Spread.none)
    Text("Almond Butter").tag(Spread.almondButter)
    …
```

Picker




```
Picker(selection: $order.avocadoStyle, label: Text("Avocado:")) {
    Text("Sliced").tag(AvocadoStyle.sliced)
    Text("Mashed").tag(AvocadoStyle.mashed)
}.pickerStyle(.radioGroup)
```

Picker




```
Picker(selection: $order.avocadoStyle, label: Text("Avocado:")) {
    Text("Sliced").tag(AvocadoStyle.sliced)
    Text("Mashed").tag(AvocadoStyle.mashed)
}.pickerStyle(.radioGroup)
```

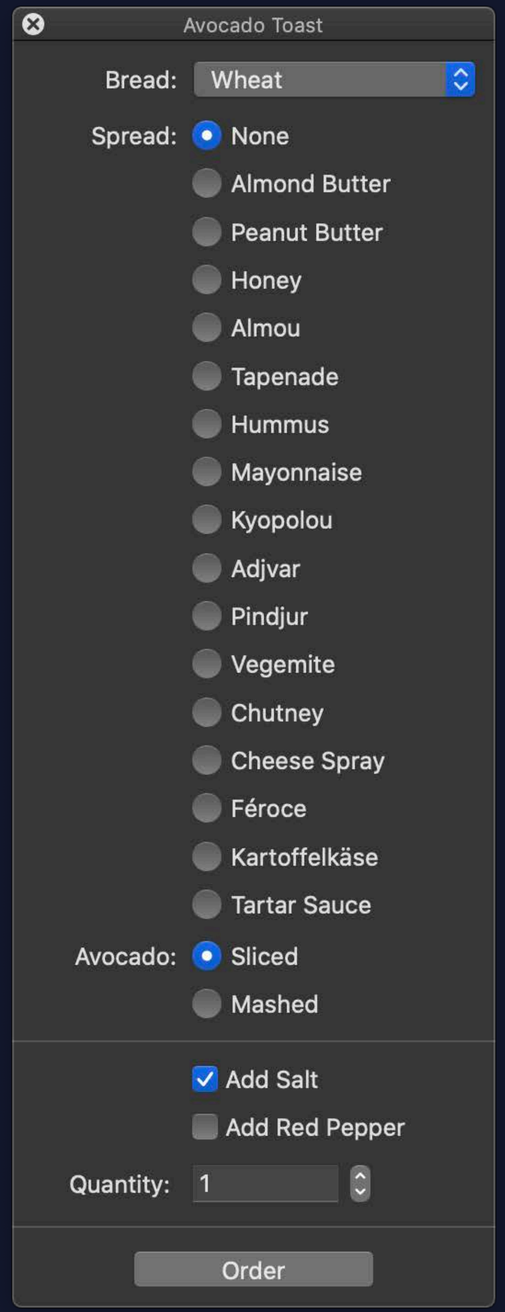
```
enum Spread : CaseIterable, Hashable, Identifiable {
    case none
    case almondButter
    case peanutButter
    case honey
}
```

```
enum Spread : CaseIterable, Hashable, Identifiable {
    case none
    case almondButter
    case peanutButter
    case honey
    case almou
    case tapenade
    case hummus
    case mayonnaise
    case kyopolou
    case adjvar
    case pindjur
    case vegemite
    case chutney
    case cannedCheese
    case feroce
    case kartoffelkase
    case tartarSauce
}
```

```
Picker(selection: $order.spread, label: Text("Spread")) {
```

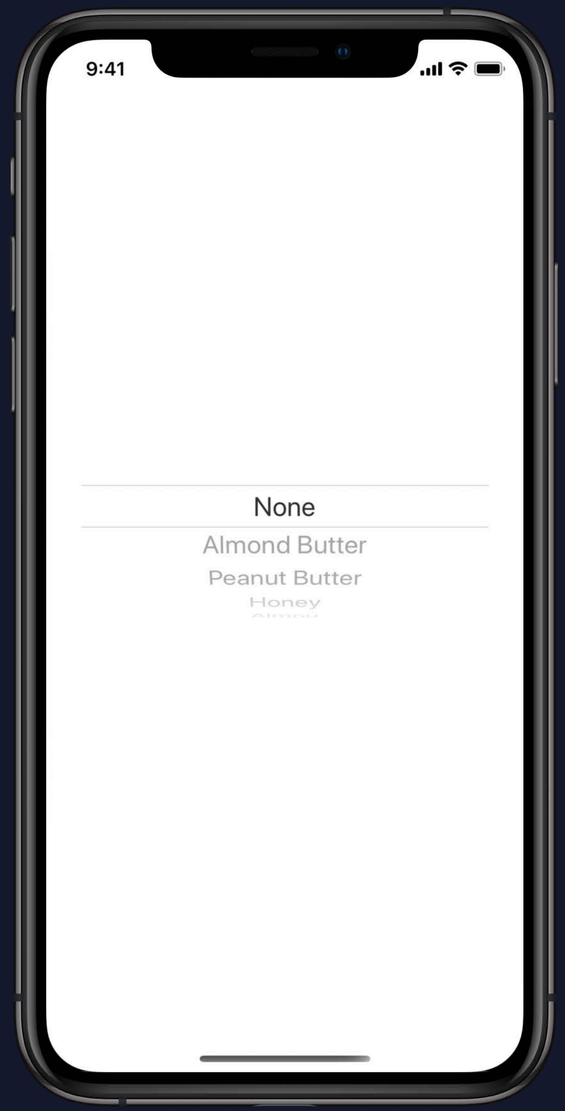
```
Picker(selection: $order.spread, label: Text("Spread")) {
    Text("None").tag(Spread.none)
    Text("Almond Butter").tag(Spread.almondButter)
    Text("Peanut Butter").tag(Spread.peanutButter)
    Text("Honey").tag(Spread.honey)
    Text("Almou").tag(Spread.almou)
    Text("Tapenade").tag(Spread.tapenade)
    Text("Hummus").tag(Spread.hummus)
    Text("Mayonnaise").tag(Spread.mayonnaise)
    Text("Kyopolou").tag(Spread.kyopolou)
    Text("Adjvar").tag(Spread.adjvar)
    Text("Pindjur").tag(Spread.pindjur)
    Text("Vegemite").tag(Spread.vegemite)
    Text("Chutney").tag(Spread.chutney)
    Text("Canned Cheese").tag(Spread.cannedCheese)
    Text("Féroce").tag(Spread.feroce)
    Text("Kartoffelkäse").tag(Spread.kartoffelkase)
    Text("Tartar Sauce").tag(Spread.tartarSauce)
}
```

```
Picker(selection: $order.spread, label: Text("Spread")) {
    Text("None").tag(Spread.none)
    Text("Almond Butter").tag(Spread.almondButter)
    Text("Peanut Butter").tag(Spread.peanutButter)
    Text("Honey").tag(Spread.honey)
    Text("Almou").tag(Spread.almou)
    Text("Tapenade").tag(Spread.tapenade)
    Text("Hummus").tag(Spread.hummus)
    Text("Mayonnaise").tag(Spread.mayonnaise)
    Text("Kyopolou").tag(Spread.kyopolou)
    Text("Adjvar").tag(Spread.adjvar)
    Text("Pindjur").tag(Spread.pindjur)
    Text("Vegemite").tag(Spread.vegemite)
    Text("Chutney").tag(Spread.chutney)
    Text("Canned Cheese").tag(Spread.cannedCheese)
    Text("Féroce").tag(Spread.feroce)
    Text("Kartoffelkäse").tag(Spread.kartoffelkase)
    Text("Tartar Sauce").tag(Spread.tartarSauce)
}
```

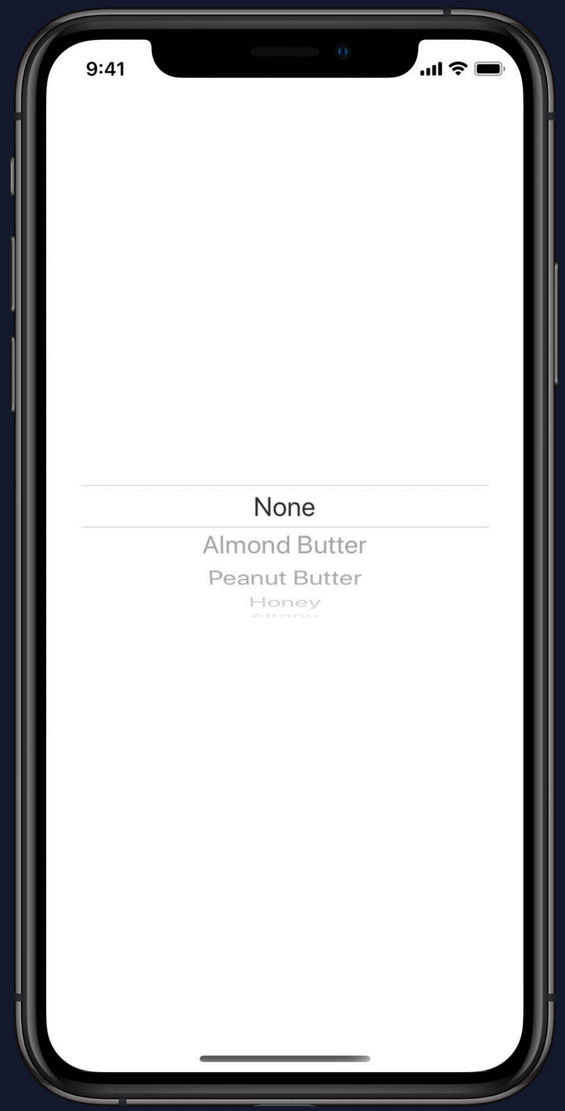


```
Picker(selection: $order.spread, label: Text("Spread")) {
    ForEach(Spread.allCases) { spread in
        Text(spread.name).tag(spread)
    }
}
```

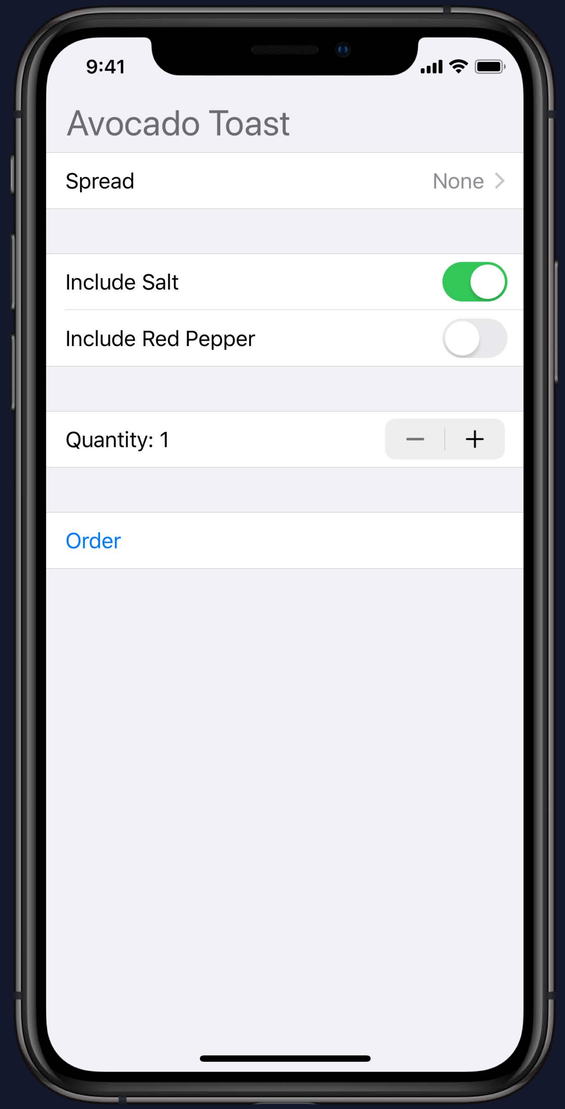
```
Picker(selection: $order.spread, label: Text("Spread")) {
    ForEach(Spread.allCases) { spread in
        Text(spread.name).tag(spread)
    }
}
```



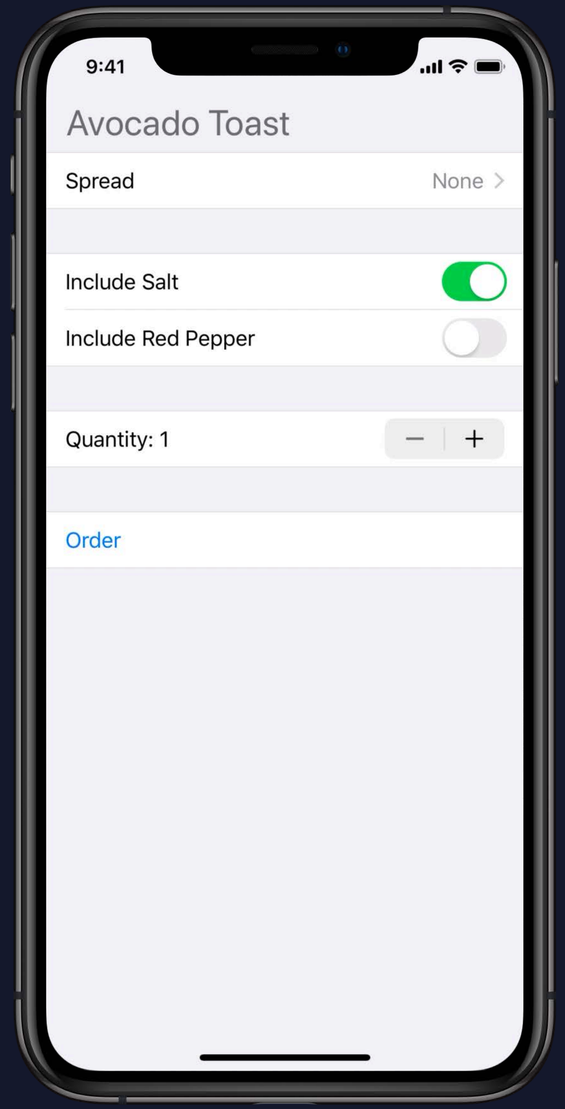
```
Picker(selection: $order.spread, label: Text("Spread")) {
    ForEach(Spread.allCases) { spread in
        Text(spread.name).tag(spread)
    }
}
```



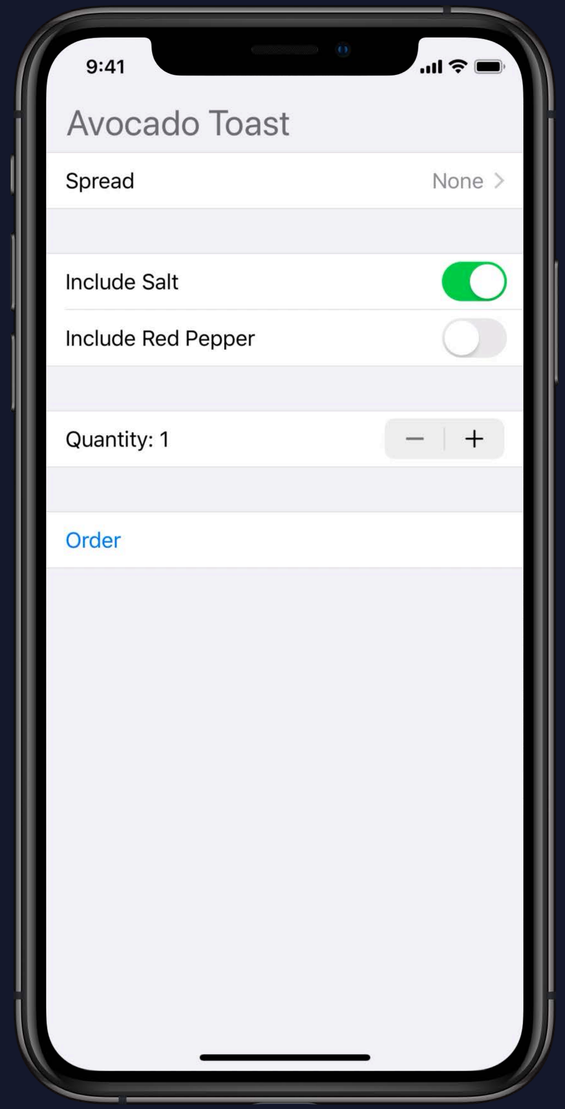
```
Form {
  Section(header: Text("Avocado Toast")) {
      Picker(selection: $order.spread, label: Text("Spread")) {
         ForEach(Spread.allCases) { spread in
            Text(spread.name).tag(spread)
         }
      }
   }
   Section { ... }
   Section { ... }
   Section { ... }
}
```



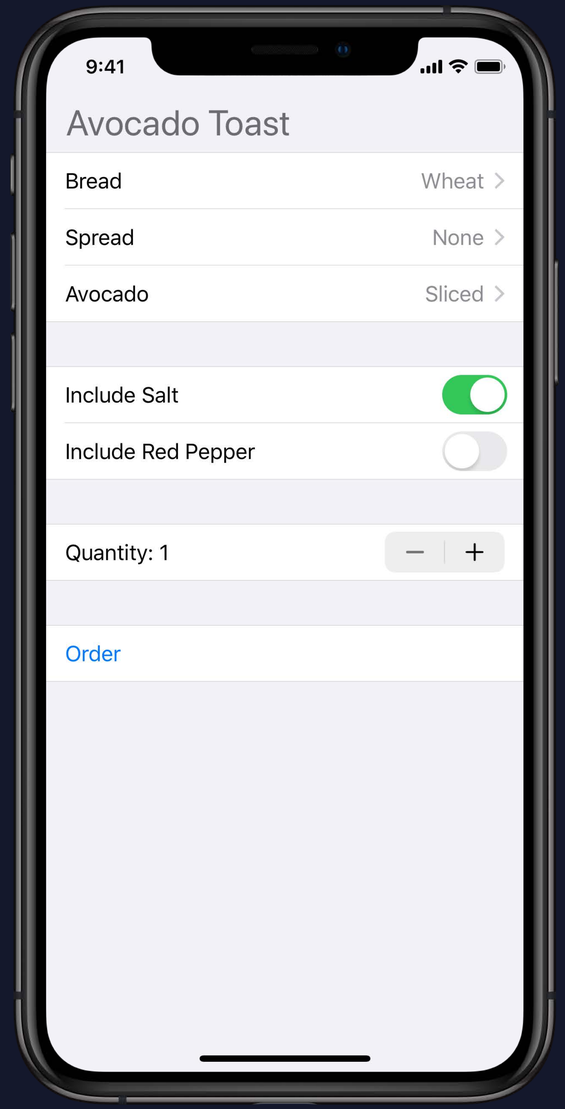
```
Form {
  Section(header: Text("Avocado Toast")) {
      Picker(selection: $order.spread, label: Text("Spread")) {
         ForEach(Spread.allCases) { spread in
            Text(spread.name).tag(spread)
         }
      }
   }
   Section { ... }
   Section { ... }
   Section { ... }
}
```



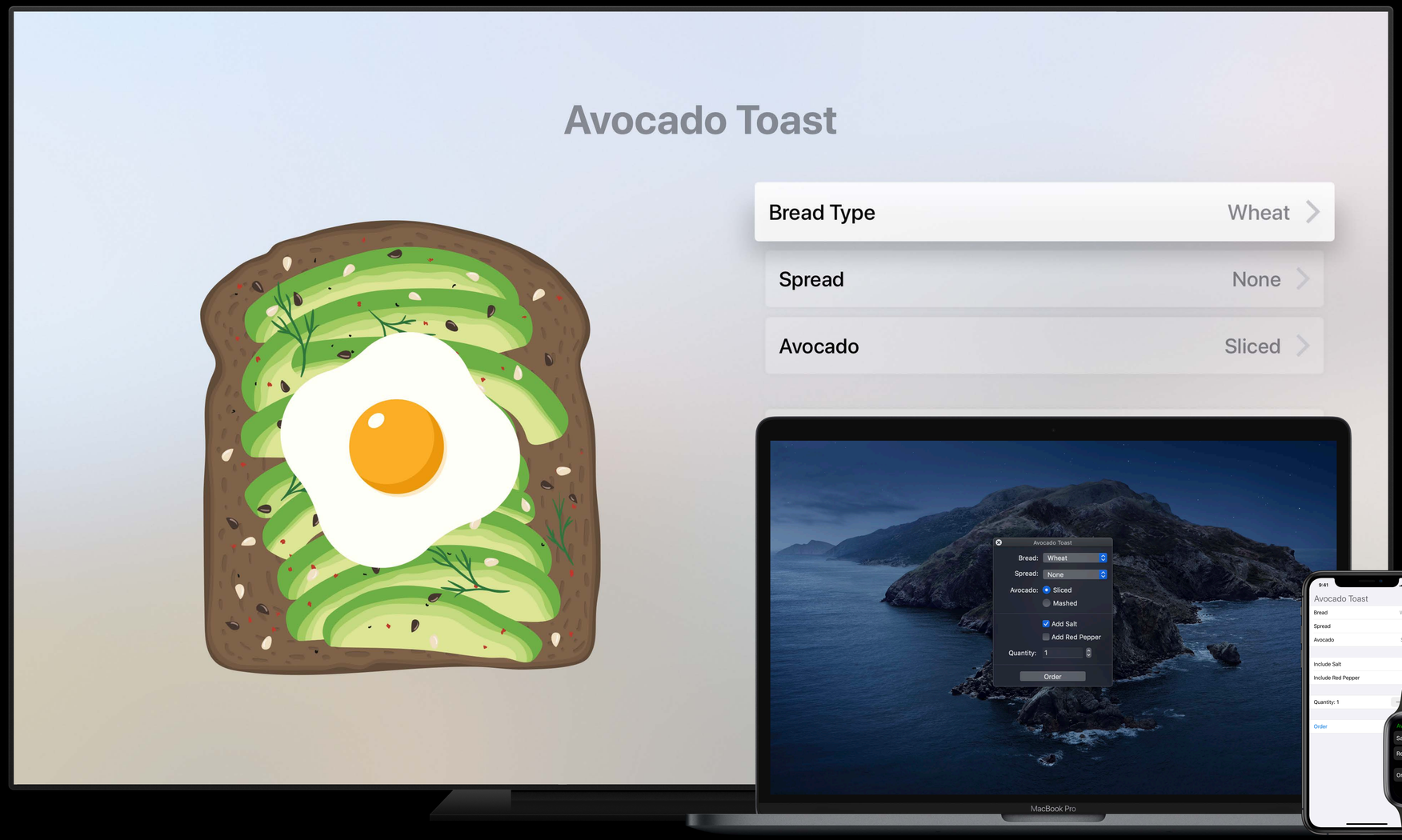
```
Form {
  Section(header: Text("Avocado Toast")) {
      Picker(selection: $order.spread, label: Text("Spread")) {
         ForEach(Spread.allCases) { spread in
            Text(spread.name).tag(spread)
         }
      }
   }
   Section { ... }
   Section { ... }
   Section { ... }
}
```



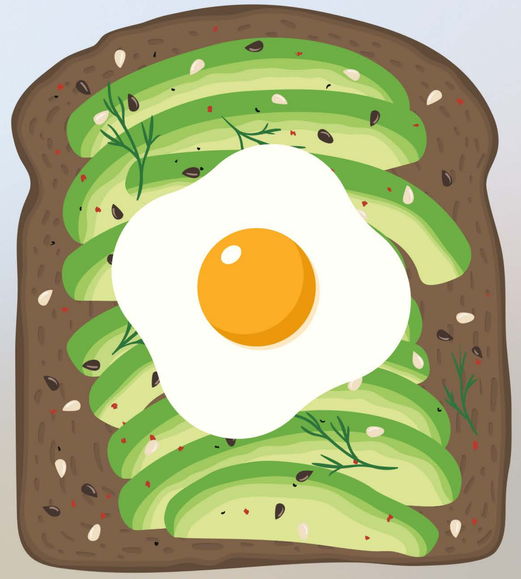
```
Form {
  Section(header: Text("Avocado Toast")) {
      Picker(selection: $order.breadType, label: Text("Bread")) {
          ...
      }
      Picker(selection: $order.spread, label: Text("Spread")) {
          ...
      }
      Picker(selection: $order.avocado, label: Text("Avocado")) {
          ...
      }
   }
   Section { ... }
   Section { ... }
   Section { ... }
}
```




Avocado Toast
Bread: Wheat
Spread: None
Avocado: Sliced
Mashed
Add Salt
Add Red Pepper
Quantity: 1
Order
MacBook Pro
9:41
Avocado Toast
Bread Wheat
Spread None
Avocado Sliced
Include Salt
Include Red Pepper
Quantity: 1
Order
Avocado Toast 9:41
Salt
Red Pepper
Order

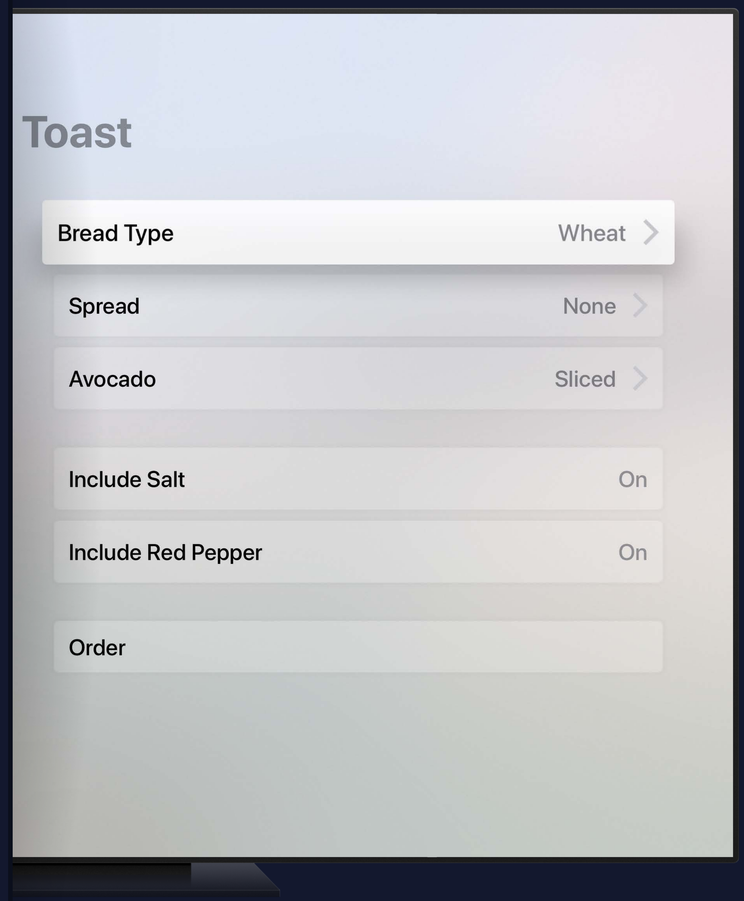

Avocado Toast
Bread Type
Wheat
Spread
None
Avocado
Sliced
Avocado Toast
Bread: Wheat
Spread: None
Avocado: Sliced
Mashed
Add Salt
Add Red Pepper
Quantity: 1
Order
MacBook Pro
9:41
Avocado Toast
Bread
Wheat
Spread
None
Avocado
Sliced
Include Salt
Include Red Pepper
Quantity: 1
Order
Avocado Toast
9:41
Salt
Red Pepper
Order

Avocado Toast



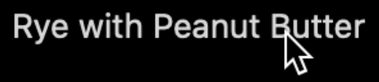
Bread Type	Wheat
Spread	None
Avocado	Sliced
Include Salt	On
Include Red Pepper	On
Order	

```
Form {
  Section {
      Picker(selection: $order.breadType, label: Text("Bread")) {
          ...
      }
      Picker(selection: $order.spread, label: Text("Spread")) {
          ...
      }
      Picker(selection: $order.avocado, label: Text("Avocado")) {
          ...
      }
   }
   Section {
      Toggle(isOn: $order.includeSalt) { ... }
      Toggle(isOn: $order.includeRedPepper) { ... }
   }
   Section {
      Button(action: submitOrder) { ... }
   }
}
```

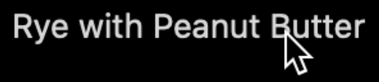


Learn once, use anywhere

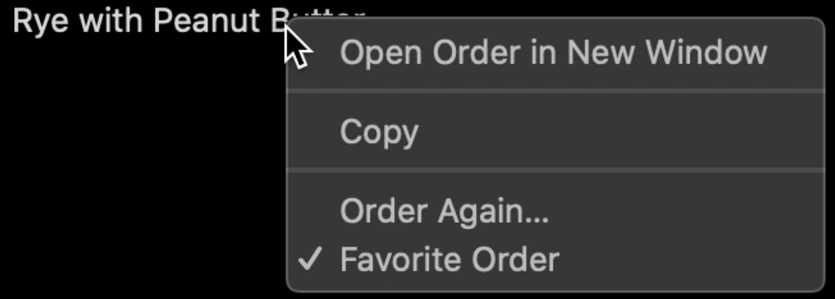
```
OrderDetailRow()
    .contextMenu {
```



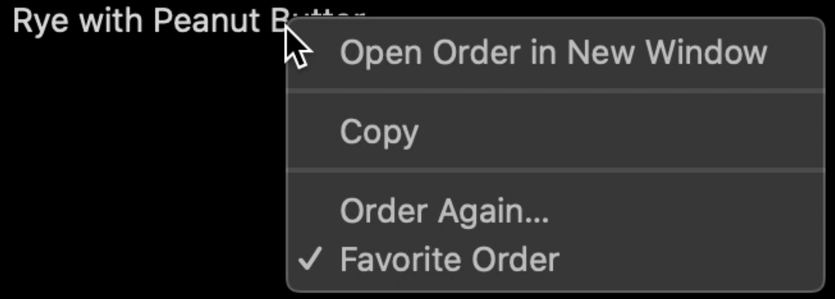
```
OrderDetailRow()
    .contextMenu {
```



```
OrderDetailRow()
    .contextMenu {
        Button(action: openInNewWindow) {
            Text("Open Order in New Window")
        }
        Divider()
        Button(action: copyOrder) {
            Text("Copy")
        }
        Divider()
        Button(action: orderAgain) {
            Text("Order Again…")
        }
        Toggle(isOn: $order.isFavorited) {
            Text("Favorite Order")
        }
    }
```



```
OrderDetailRow()
    .contextMenu {
        Button(action: openInNewWindow) {
            Text("Open Order in New Window")
        }
        Divider()
        Button(action: copyOrder) {
            Text("Copy")
        }
        Divider()
        Button(action: orderAgain) {
            Text("Order Again…")
        }
        Toggle(isOn: $order.isFavorited) {
            Text("Favorite Order")
        }
    }
```



Controls in SwiftUI

Describe purpose, not visuals

Adaptive and reusable

Customizable

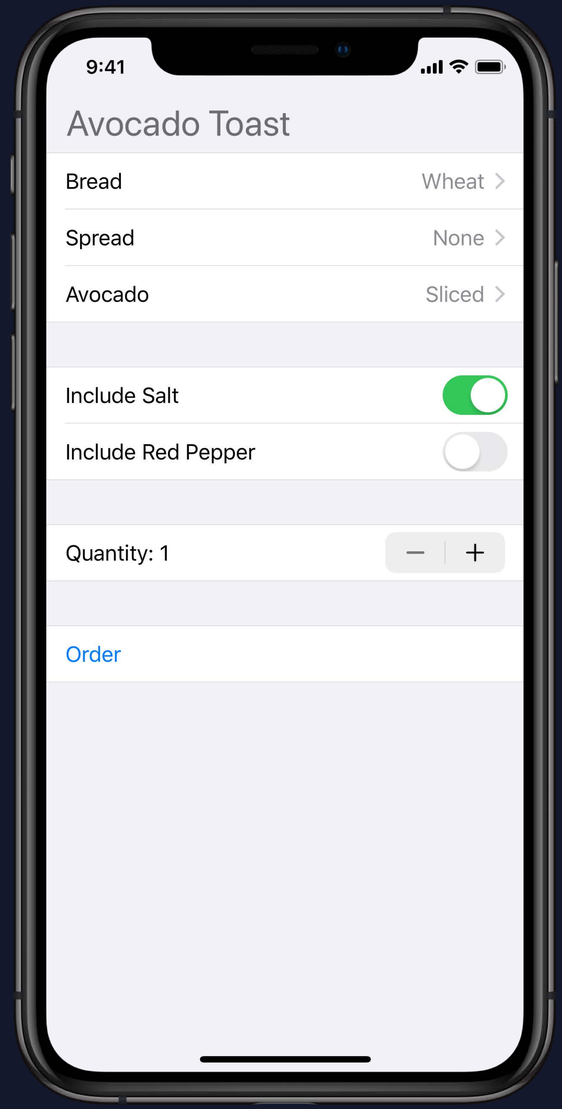
Accessible out of the box

Control Modifiers

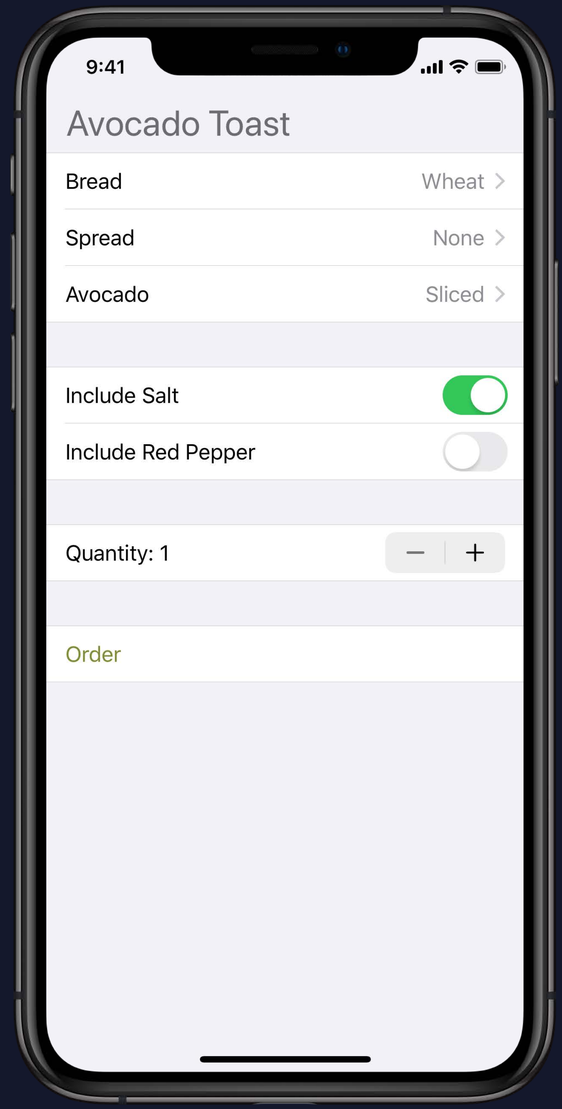
Shared behaviors

Applies to subtree of views

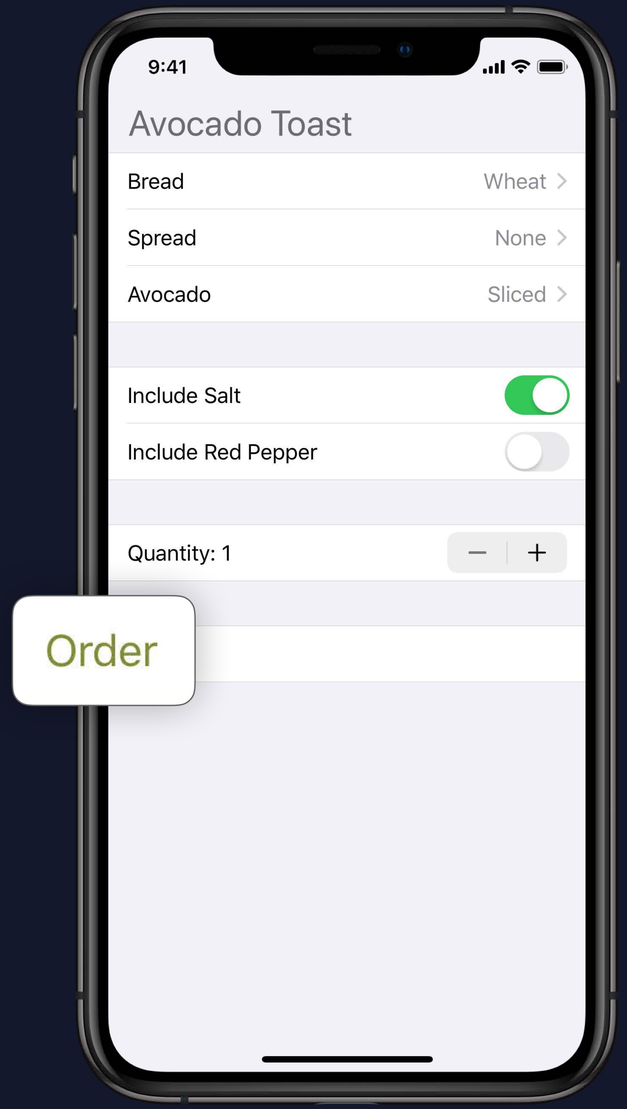
```
Form {
   Section {
      Picker(selection: $order.bread, label: Text("Bread")) { ••• }
      Picker(selection: $order.spread, label: Text("Spread")) { ••• }
      Picker(selection: $order.avocado, label: Text("Avocado")) { ••• }
   }
   Section {
      Toggle(isOn: $order.includeSalt) { ••• }
      Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
      Stepper(value: $order.quantity, in: 1...10) { ••• }
   }
   Section {
      Button(action: submitOrder) { Text("Order") }
   }
}
```



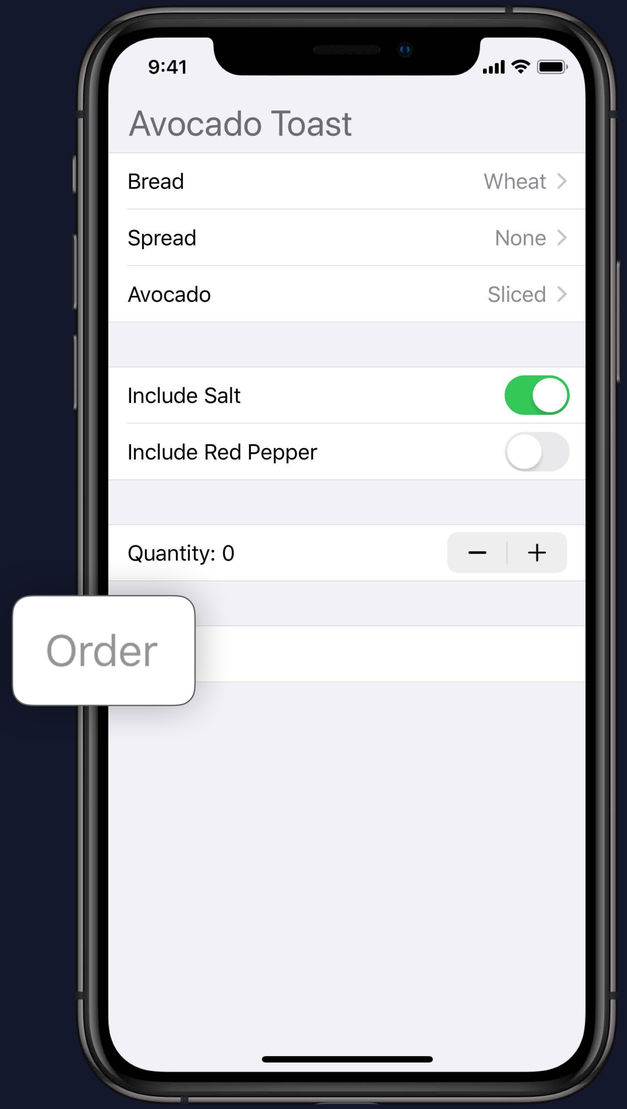
```
Form {
    Section {
        Picker(selection: $order.bread, label: Text("Bread")) { ••• }
        Picker(selection: $order.spread, label: Text("Spread")) { ••• }
        Picker(selection: $order.avocado, label: Text("Avocado")) { ••• }
    }
    Section {
        Toggle(isOn: $order.includeSalt) { ••• }
        Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
        Stepper(value: $order.quantity, in: 1...10) { ••• }
    }
    Section {
        Button(action: submitOrder) { Text("Order") }
    }
}
.accentColor(Color("avocadoGreen"))
```



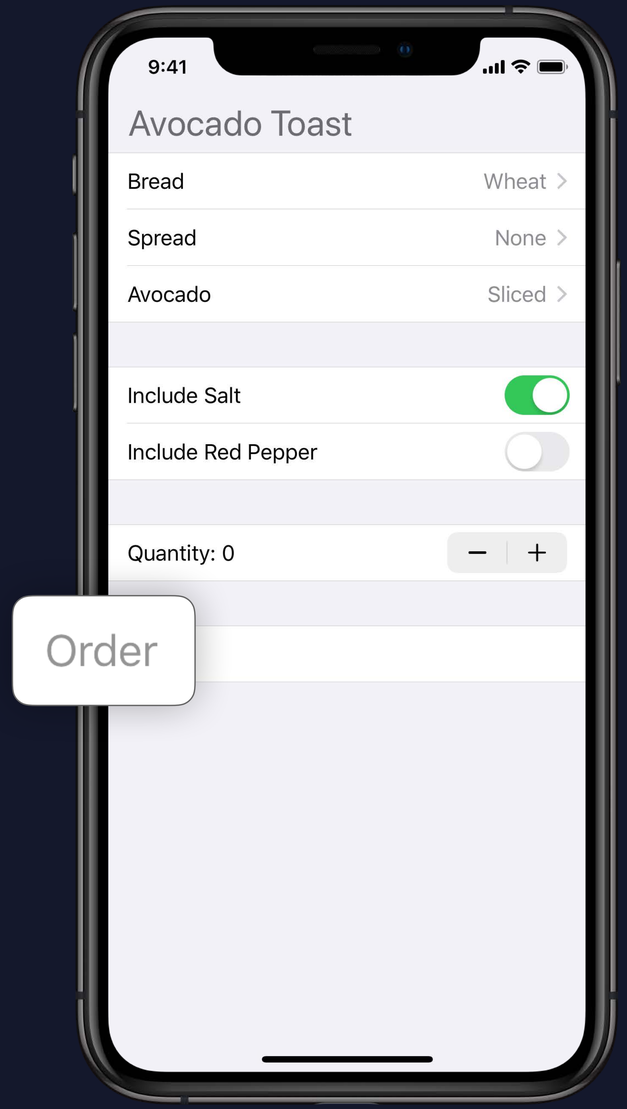
```
Form {
    Section {
        Picker(selection: $order.bread, label: Text("Bread")) { ••• }
        Picker(selection: $order.spread, label: Text("Spread")) { ••• }
        Picker(selection: $order.avocado, label: Text("Avocado")) { ••• }
    }
    Section {
        Toggle(isOn: $order.includeSalt) { ••• }
        Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
        Stepper(value: $order.quantity, in: 1...10) { ••• }
    }
    Section {
        Button(action: submitOrder) { Text("Order") }
    }
}
.accentColor(Color("avocadoGreen"))
```



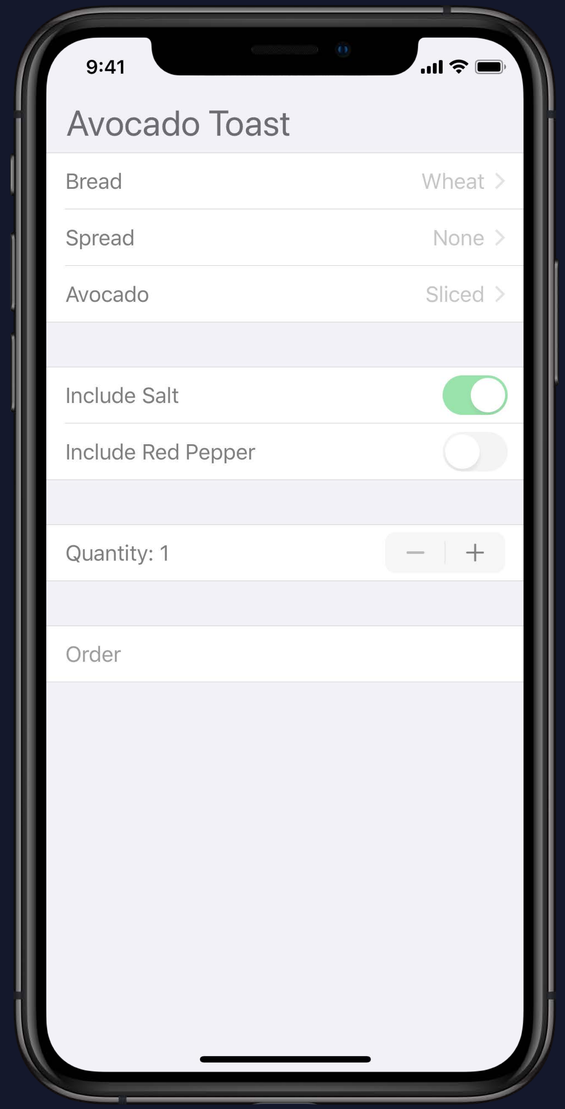
```
Form {
   Section {
      Picker(selection: $order.bread, label: Text("Bread")) { ••• }
      Picker(selection: $order.spread, label: Text("Spread")) { ••• }
      Picker(selection: $order.avocado, label: Text("Avocado")) { ••• }
   }
   Section {
      Toggle(isOn: $order.includeSalt) { ••• }
      Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
      Stepper(value: $order.quantity, in: 1...10) { ••• }
   }
   Section {
      Button(action: submitOrder) { Text("Order") }
         .disabled(order.quantity == 0)
   }
}
.accentColor(Color("avocadoGreen"))
```



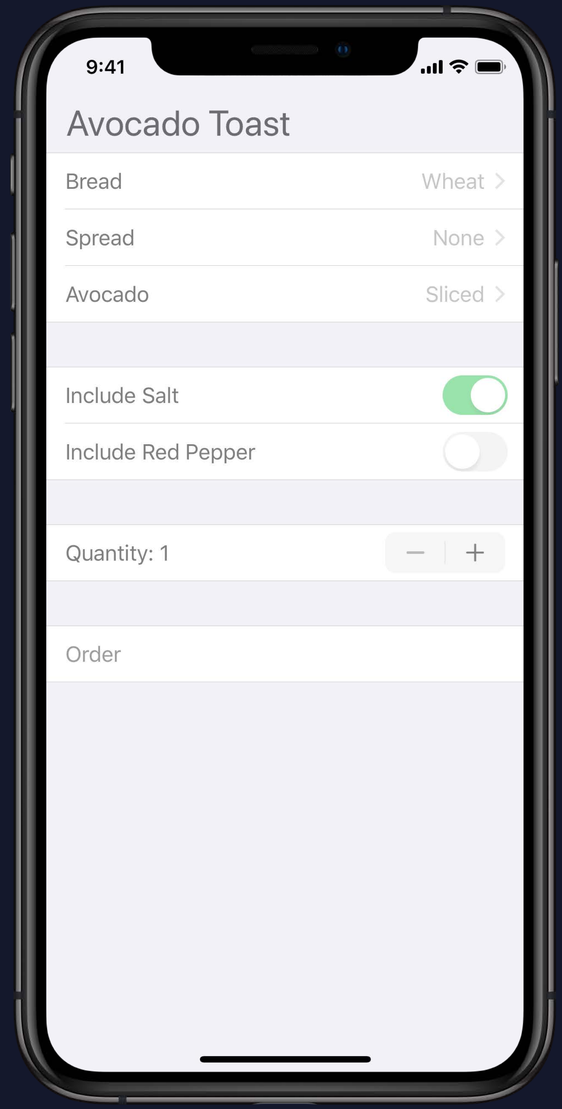
```
Form {
   Section {
      Picker(selection: $order.bread, label: Text("Bread")) { ••• }
      Picker(selection: $order.spread, label: Text("Spread")) { ••• }
      Picker(selection: $order.avocado, label: Text("Avocado")) { ••• }
   }
   Section {
      Toggle(isOn: $order.includeSalt) { ••• }
      Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
      Stepper(value: $order.quantity, in: 1...10) { ••• }
   }
   Section {
      Button(action: submitOrder) { Text("Order") }
         .disabled(order.quantity == 0)
   }
}
.accentColor(Color("avocadoGreen"))
```



```
Form {
    Section {
        Picker(selection: $order.bread, label: Text("Bread")) { ••• }
            .disabled(!connectedToToastNetwork)
        Picker(selection: $order.spread, label: Text("Spread")) { ••• }
            .disabled(!connectedToToastNetwork)
        Picker(selection: $order.avocado, label: Text("Avocado")) { ••• }
            .disabled(!connectedToToastNetwork)
    }
    Section {
        Toggle(isOn: $order.includeSalt) { ••• }
            .disabled(!connectedToToastNetwork)
        Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
            .disabled(!connectedToToastNetwork)
        Stepper(value: $order.quantity, in: 1...10) { ••• }
            .disabled(!connectedToToastNetwork)
    }
    …
```



```
Form {
    Section {
        Picker(selection: $order.bread, label: Text("Bread")) { ••• }
        Picker(selection: $order.spread, label: Text("Spread")) { ••• }
        Picker(selection: $order.avocado, label: Text("Avocado")) { ••• }
    }
    Section {
        Toggle(isOn: $order.includeSalt) { ••• }
        Toggle(isOn: $order.includeRedPepperFlakes) { ••• }
        Stepper(value: $order.quantity, in: 1...10) { ••• }
    }
    Section {
        Button(action: submitOrder) { Text("Order") }
            .disabled(order.quantity == 0)
    }
}
.accentColor(Color("avocadoGreen"))
.disabled(!connectedToToastNetwork)
```



Environment

buttonStyle = .default

colorSchemeContrast = .high

contentSizeCategory = .medium

accentColor = 

displayScale = 2×

colorScheme = .dark

timeZone = PDT

locale = ar_DZ

controlSize = .regular

controlsAppearActive = true

layoutDirection = .rightToLeft

defaultFont = SF

calendar = gregorian

Environment

buttonStyle = .default

contentSizeCategory = .medium

displayScale = 2×

timeZone = PDT

controlsAppearActive = true

defaultFont = SF

colorSchemeContrast = .high

accentColor = ●

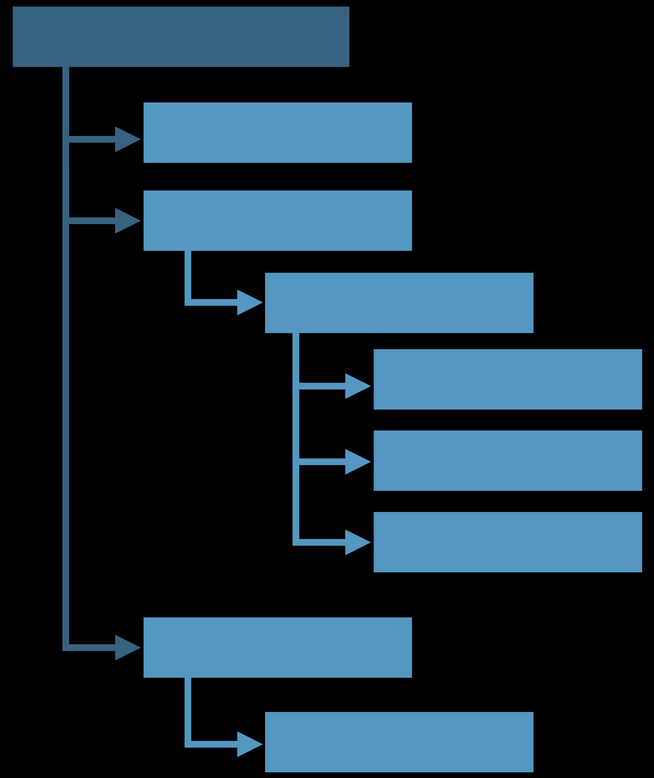
colorScheme = .dark

controlSize = .regular

locale = ar_DZ

layoutDirection = .rightToLeft

calendar = gregorian



Environment

buttonStyle = .default

contentSizeCategory = .medium

displayScale = 2×

timeZone = PDT

controlsAppearActive = true

defaultFont = SF

colorSchemeContrast = .high

accentColor =

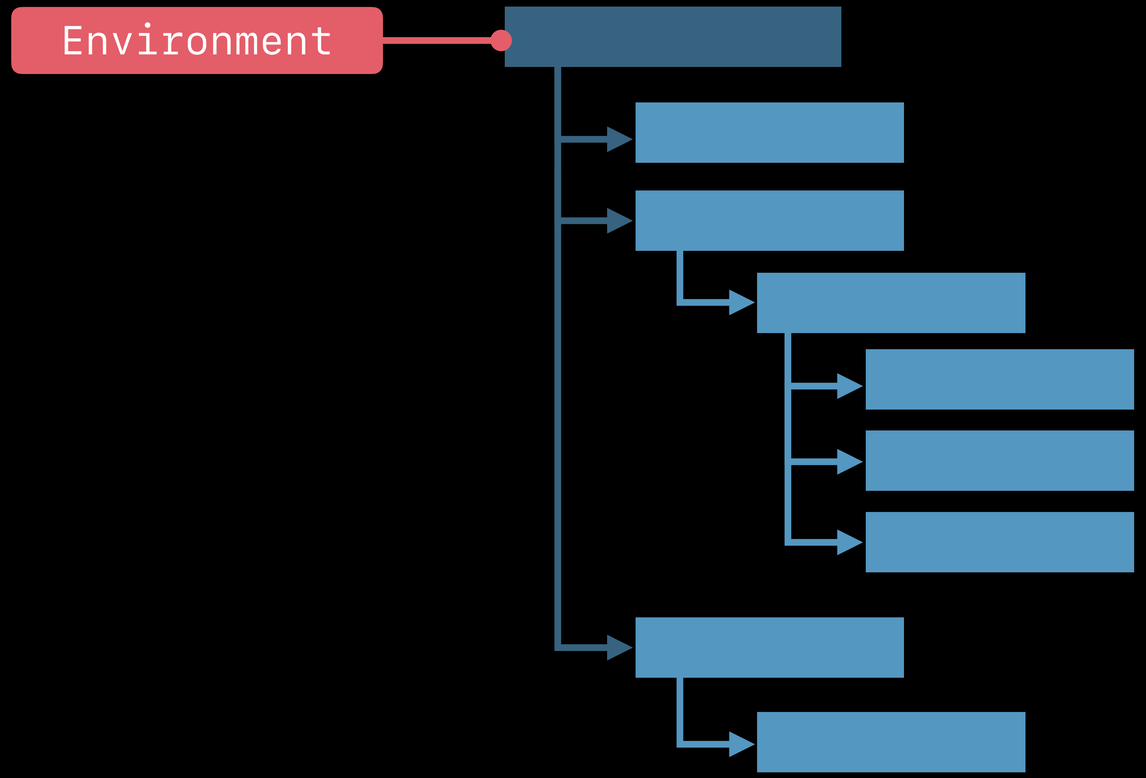
colorScheme = .dark

controlSize = .regular

locale = ar_DZ

layoutDirection = .rightToLeft

calendar = gregorian



Environment

buttonStyle = .default

contentSizeCategory = .medium

displayScale = 2×

timeZone = PDT

controlsAppearActive = true

defaultFont = SF

colorSchemeContrast = .high

accentColor =

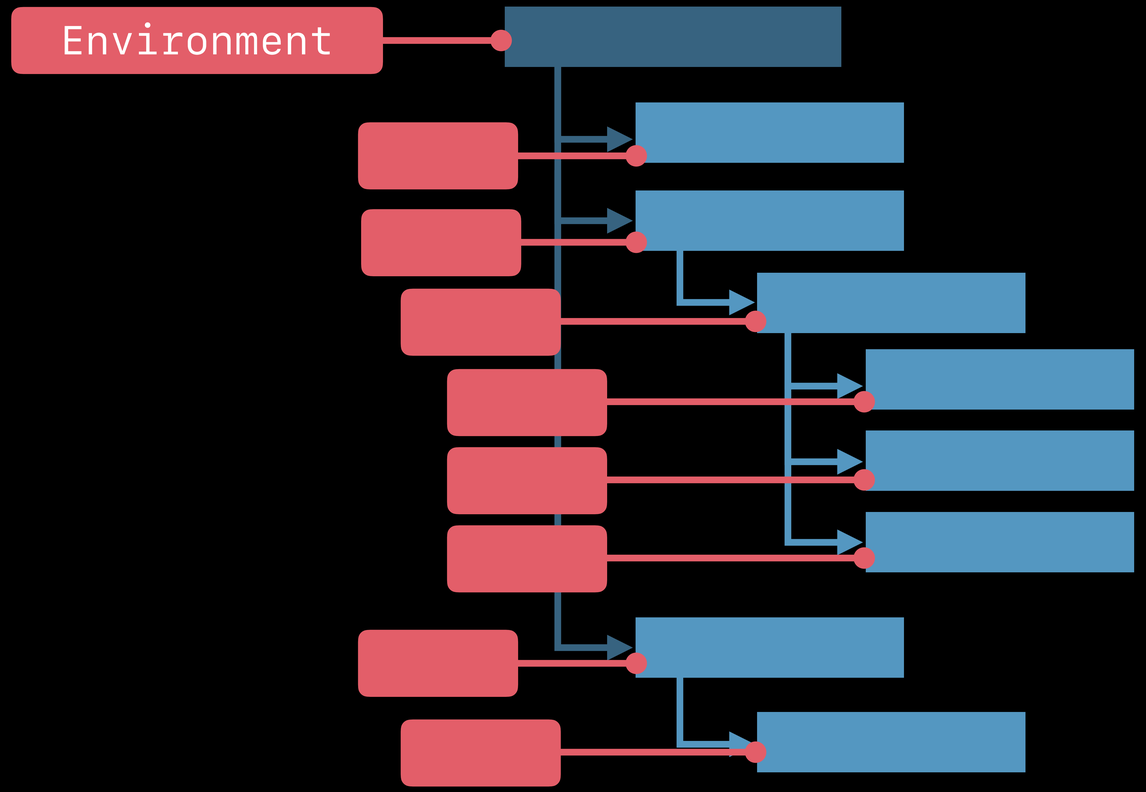
colorScheme = .dark

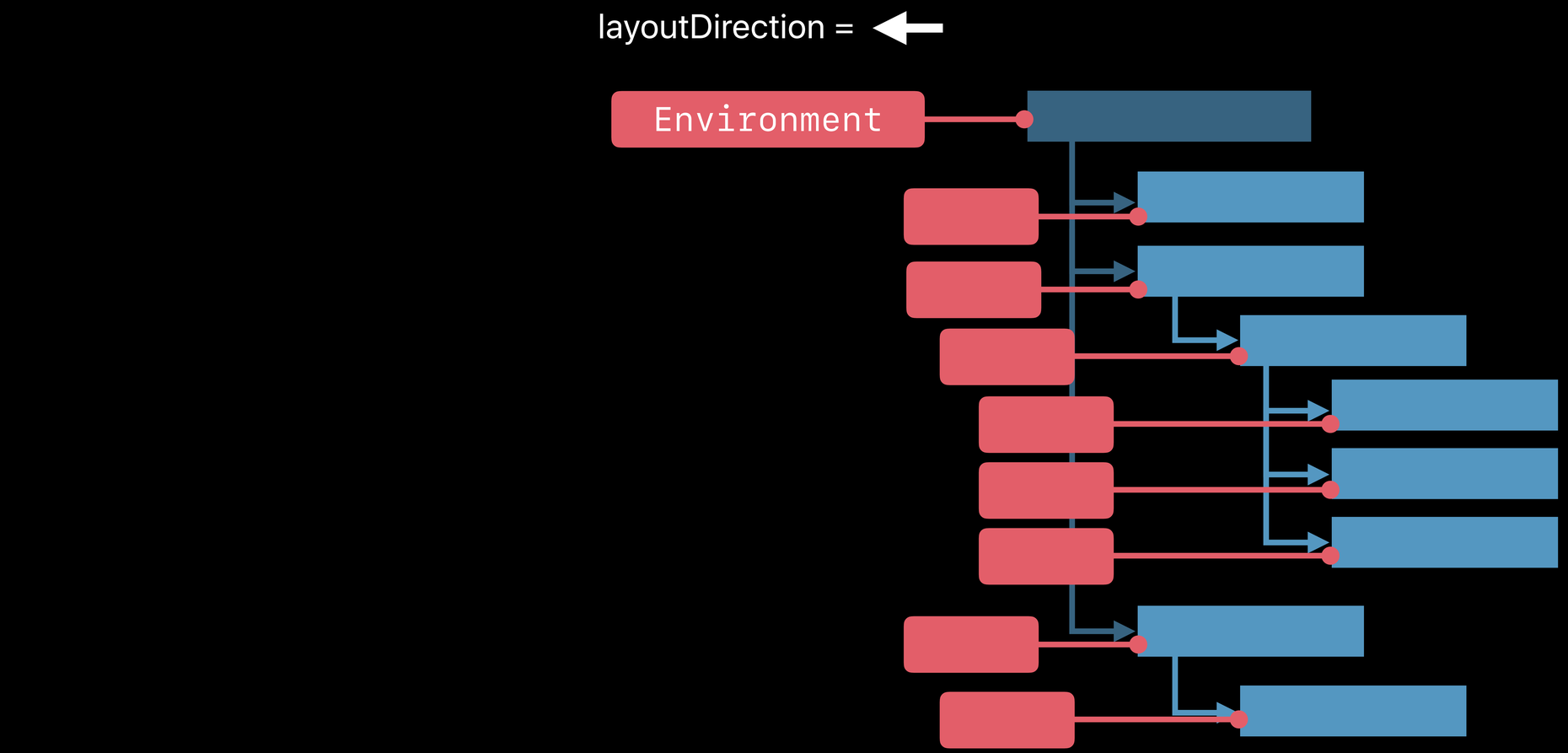
controlSize = .regular

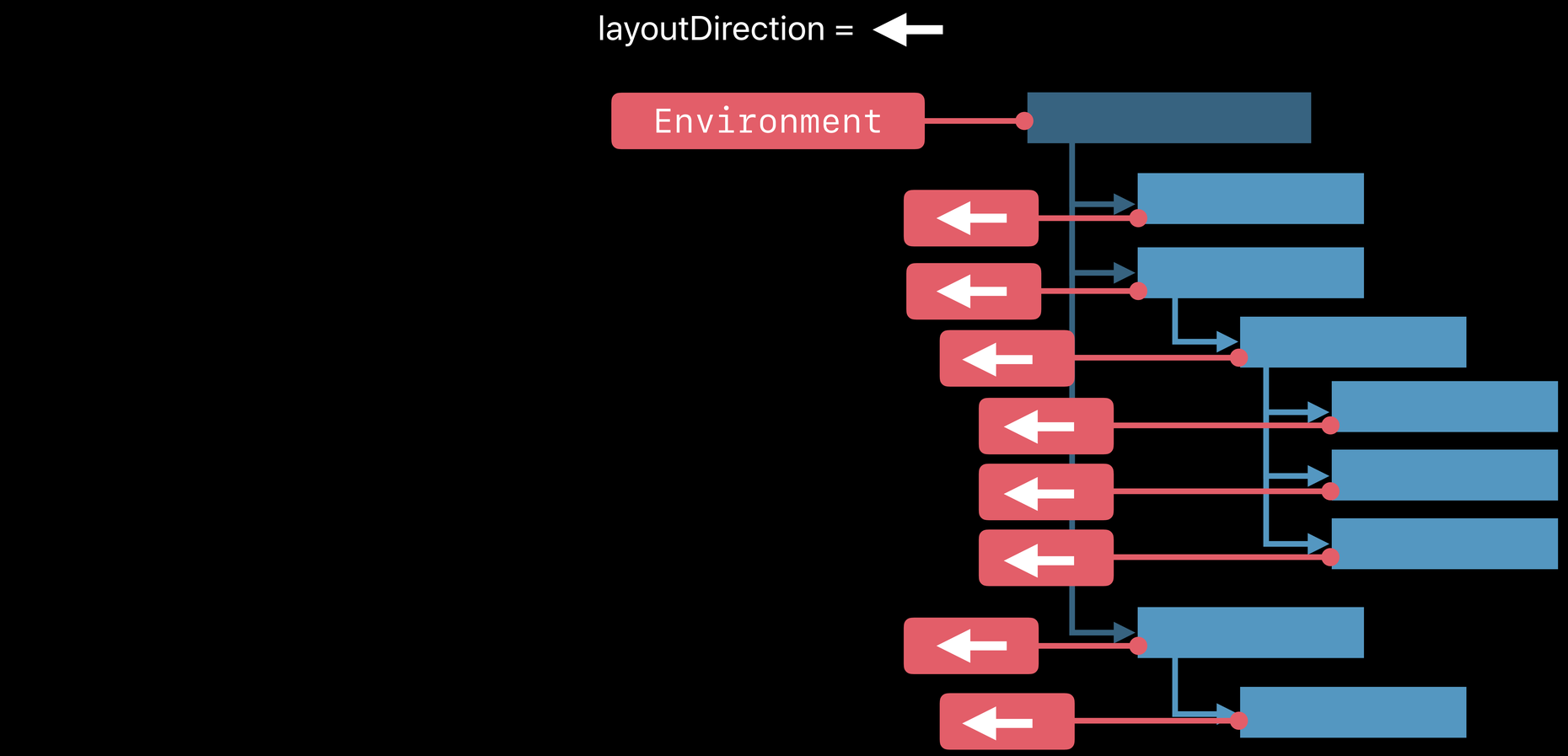
locale = ar_DZ

layoutDirection = .rightToLeft

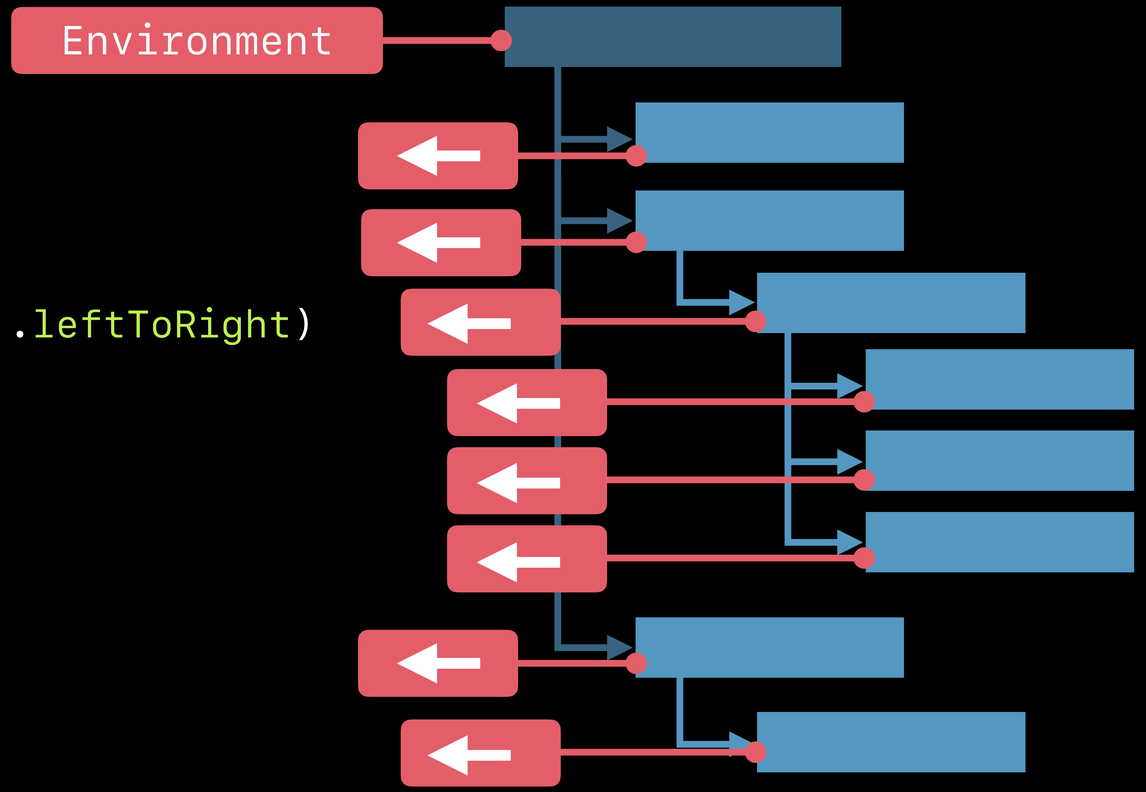
calendar = gregorian



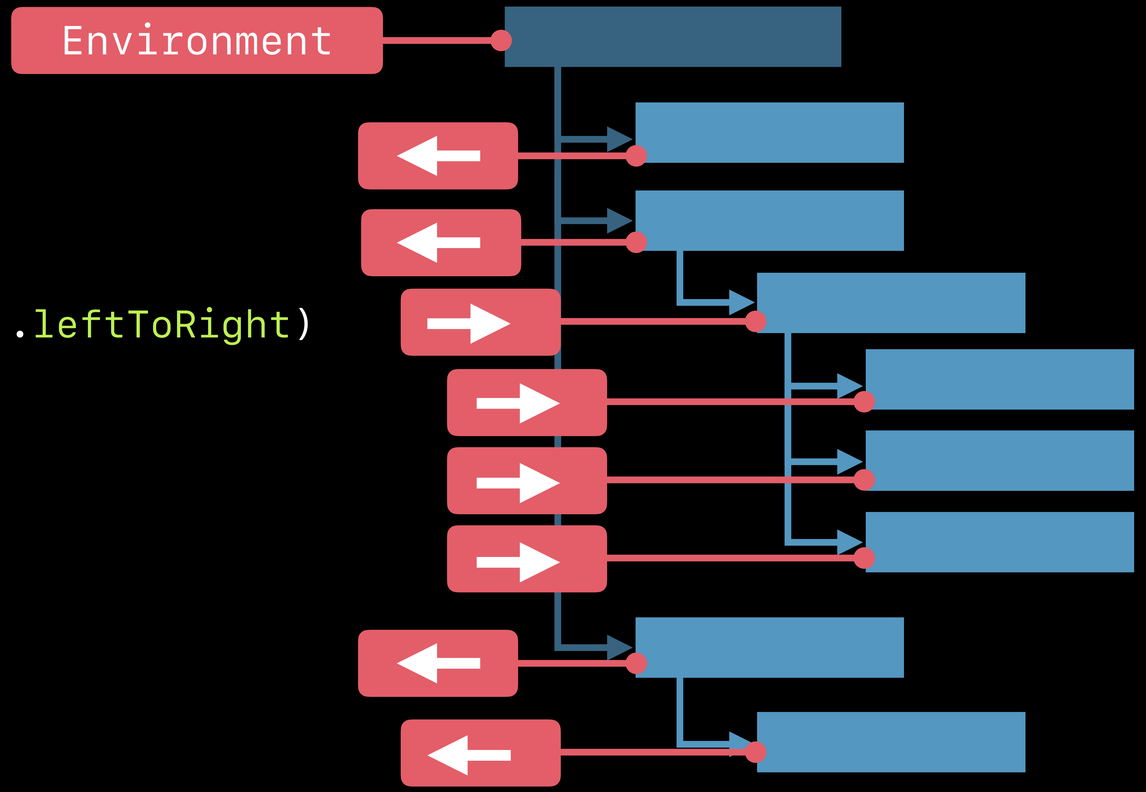

layoutDirection =
Environment

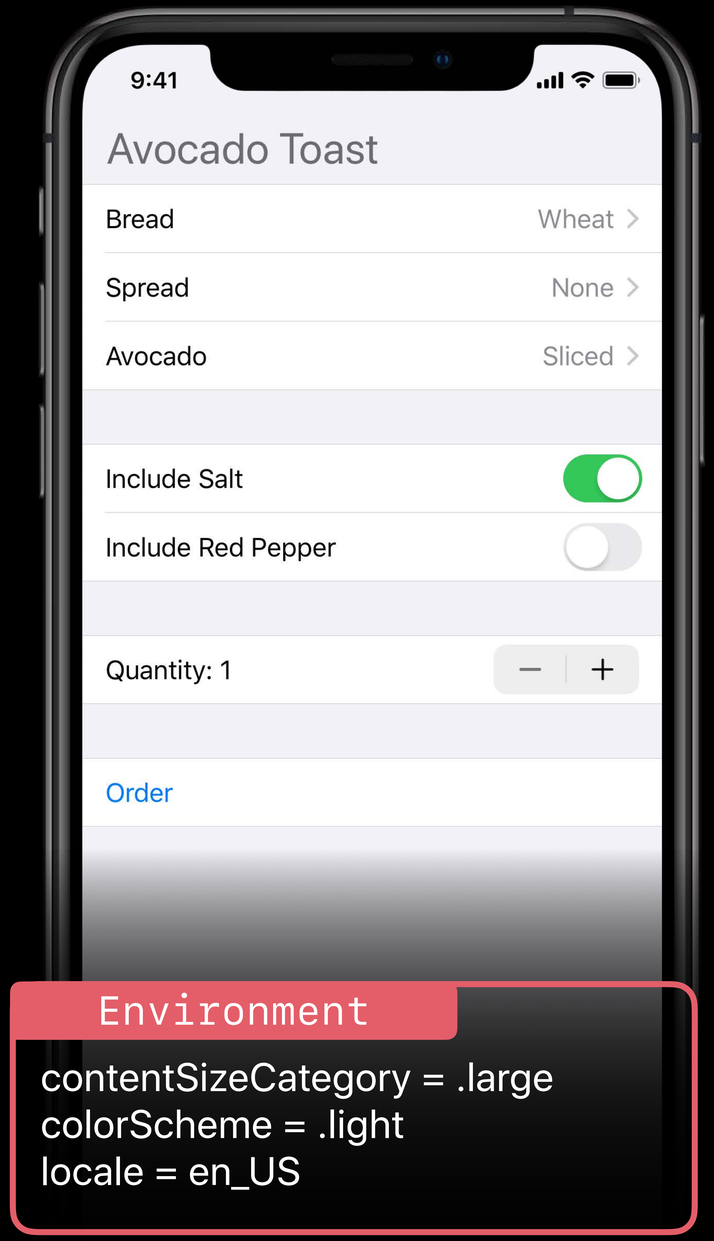

layoutDirection =
Environment

```
.environment(\.layoutDirection, .leftToRight)
```



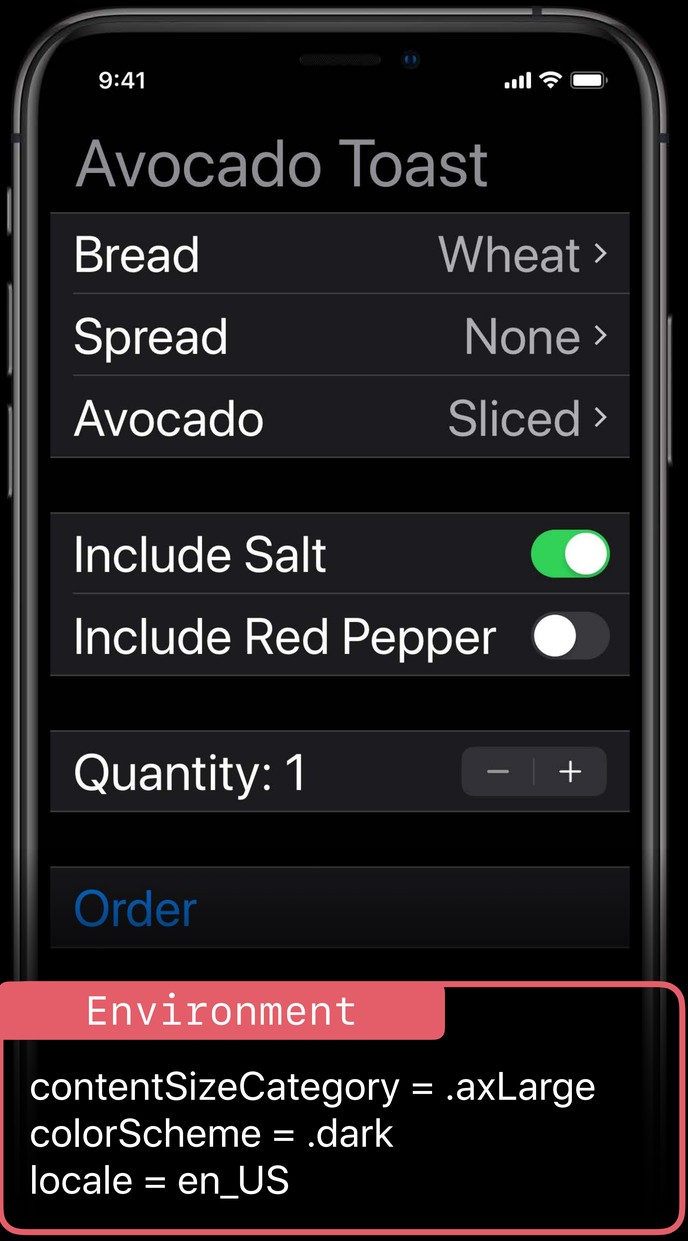
```
.environment(\.layoutDirection, .leftToRight)
```





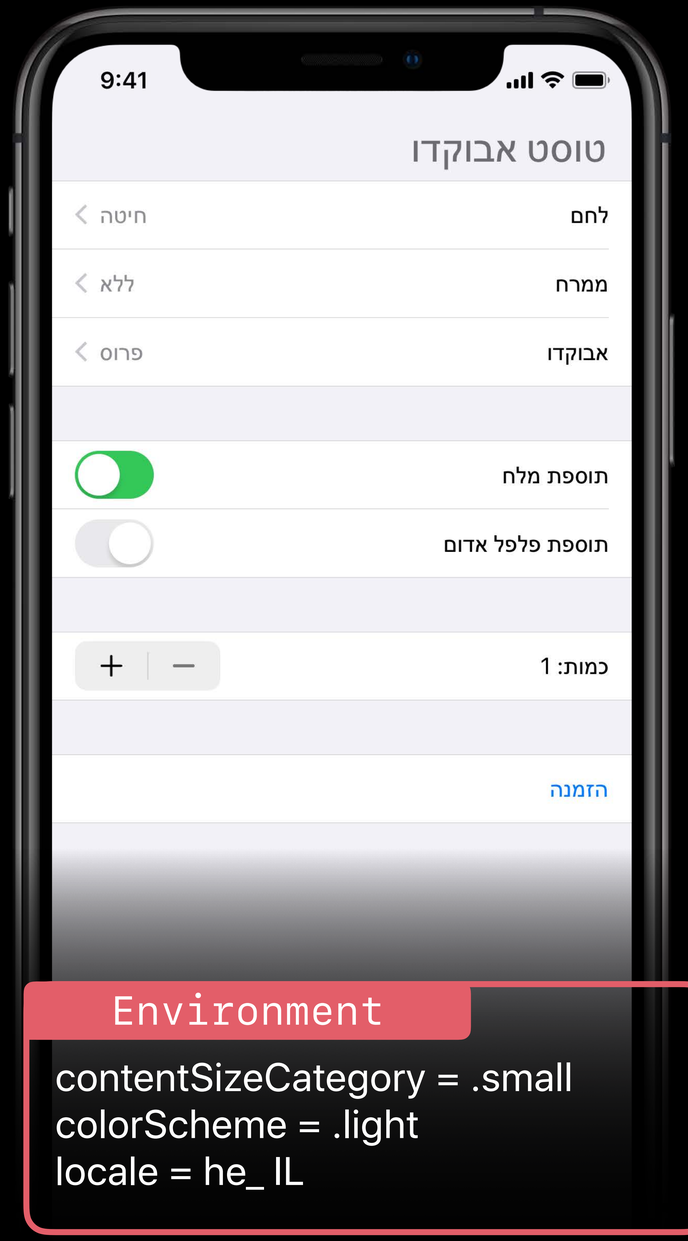
Environment

contentSizeCategory = .large
colorScheme = .light
locale = en_US



Environment

contentSizeCategory = .axLarge
colorScheme = .dark
locale = en_US



Environment

contentSizeCategory = .small
colorScheme = .light
locale = he_ IL

```swift
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint

    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
    }

    private var dragGesture: some Gesture { ... }
    private var eggPosition: CGPoint { ... }
}
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint

    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
    }

    private var dragGesture: some Gesture { ... }
    private var eggPosition: CGPoint { ... }
}
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint

    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
    }

    private var dragGesture: some Gesture { ... }
    private var eggPosition: CGPoint { ... }
}
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint

    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
    }

    private var dragGesture: some Gesture { ... }
    private var eggPosition: CGPoint { ... }
}
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint

    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
    }

    private var dragGesture: some Gesture { ... }
    private var eggPosition: CGPoint { ... }
}
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint

    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
    }

    private var dragGesture: some Gesture { ... }
    private var eggPosition: CGPoint { ... }
}

EggPlacementView($order.eggPlacement).disabled(true)
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint

    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
    }

    private var dragGesture: some Gesture { ... }
    private var eggPosition: CGPoint { ... }
}

EggPlacementView($order.eggPlacement).disabled(true)
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint

    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
    }

    private var dragGesture: some Gesture { ... }
    private var eggPosition: CGPoint { ... }
}

EggPlacementView($order.eggPlacement).disabled(true)
```

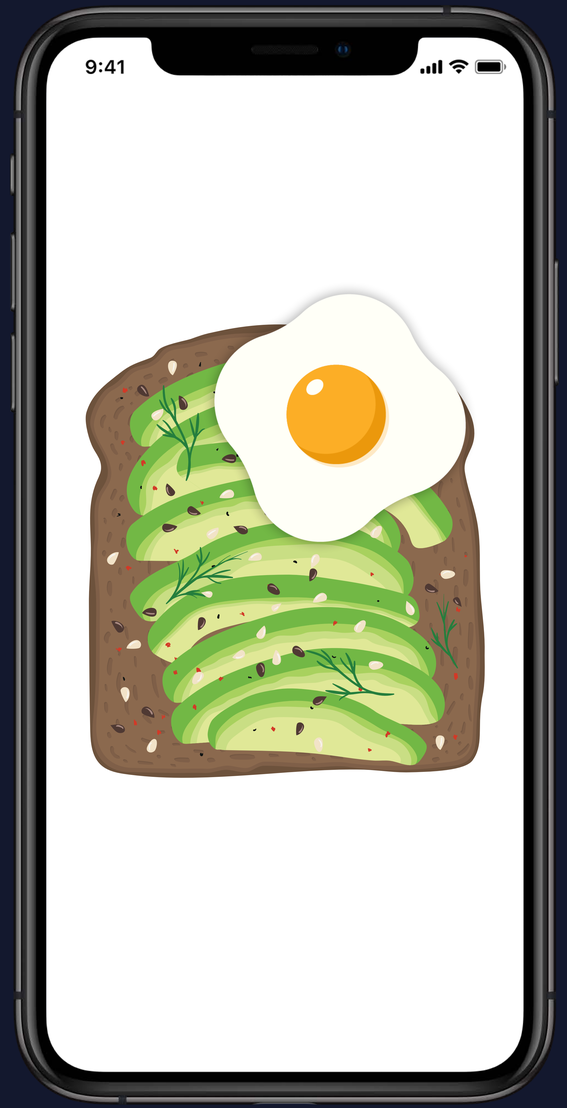
```swift
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint

    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
    }

    private var dragGesture: some Gesture { ... }
    private var eggPosition: CGPoint { ... }
}

EggPlacementView($order.eggPlacement).disabled(true)
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint
    @Environment(\.isEnabled) private var isEnabled: Bool
    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
        .saturation(isEnabled ? 1.0 : 0.2)
    }

    private var dragGesture: some Gesture { ••• }
    private var eggPosition: CGPoint { ••• }
}

EggPlacementView($order.eggPlacement).disabled(true)
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint
    @Environment(\.isEnabled) private var isEnabled: Bool
    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
        .saturation(isEnabled ? 1.0 : 0.2)
    }

    private var dragGesture: some Gesture { ••• }
    private var eggPosition: CGPoint { ••• }
}

EggPlacementView($order.eggPlacement).disabled(true)
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint
    @Environment(\.isEnabled) private var isEnabled: Bool
    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
        .saturation(isEnabled ? 1.0 : 0.2)
    }

    private var dragGesture: some Gesture { ••• }
    private var eggPosition: CGPoint { ••• }
}

EggPlacementView($order.eggPlacement).disabled(true)
```

```
struct EggPlacementView : View {
    @Binding var eggPlacement: UnitPoint
    @Environment(\.isEnabled) private var isEnabled: Bool
    var body: some View {
        ZStack {
            Image("Toast")
            Image("Egg")
                .position(eggPosition)
                .gesture(dragGesture)
        }
        .saturation(isEnabled ? 1.0 : 0.2)
    }

    private var dragGesture: some Gesture { ••• }
    private var eggPosition: CGPoint { ••• }
}

EggPlacementView($order.eggPlacement).disabled(false)
```




saturation
disabled(true)
Environment

The Basics: Views and modifiers

Data-driven views

Composing controls

Navigating your app

9:41

Avocado Toast

Bread Wheat >

Spread None >

Avocado Sliced >

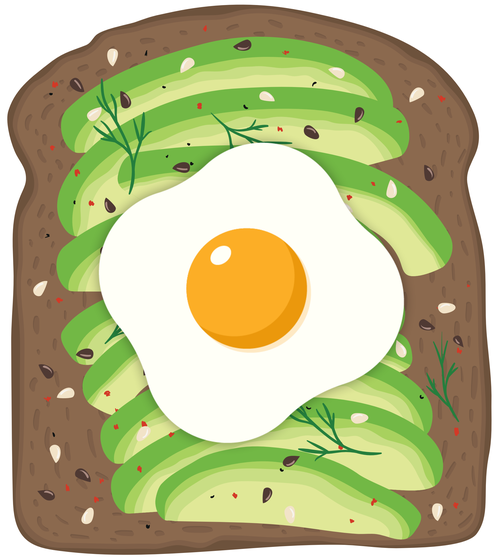
Include Salt

Include Red Pepper

Quantity: 1 − +

Order

9:41



9:41

Rye with Almond Butter
5/30/19, 9:20 PM — S R

Multi-Grain with Hummus
5/29/19, 5:34 PM — R

Multi-Grain Toast
5/28/19, 1:47 PM — S

Sourdough with Chutney
5/27/19, 10:00 AM — S R

Rye with Peanut Butter
5/26/19, 6:14 AM — S R E

Wheat with Tapenade
5/25/19, 2:27 AM — E

Sourdough with Vegemite
5/23/19, 10:40 PM — S

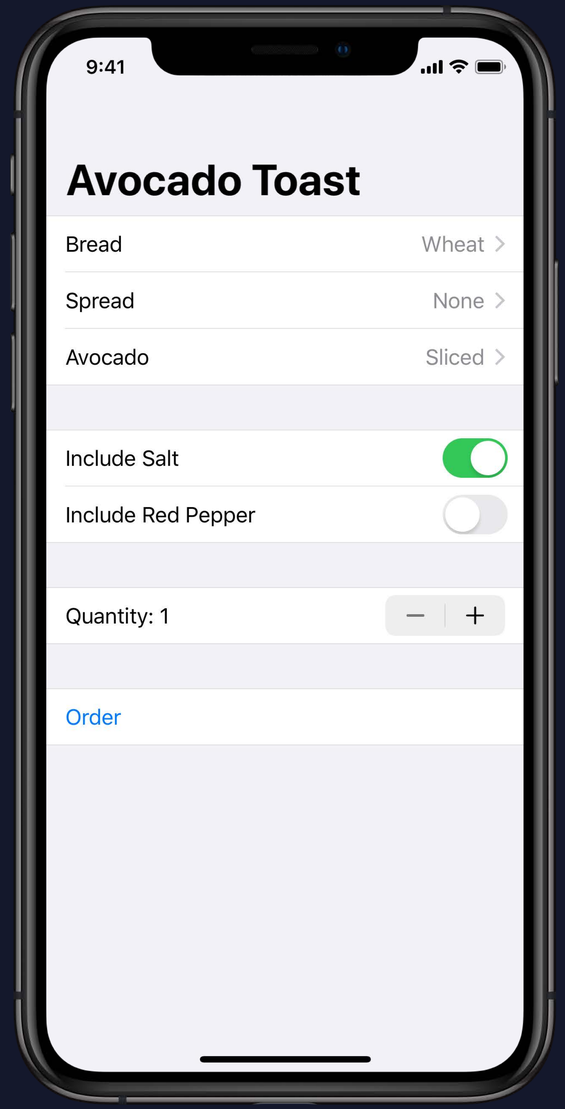
Wheat with Féroce
5/22/19, 6:54 PM — S R E

Rye with Honey
5/21/19, 3:07 PM

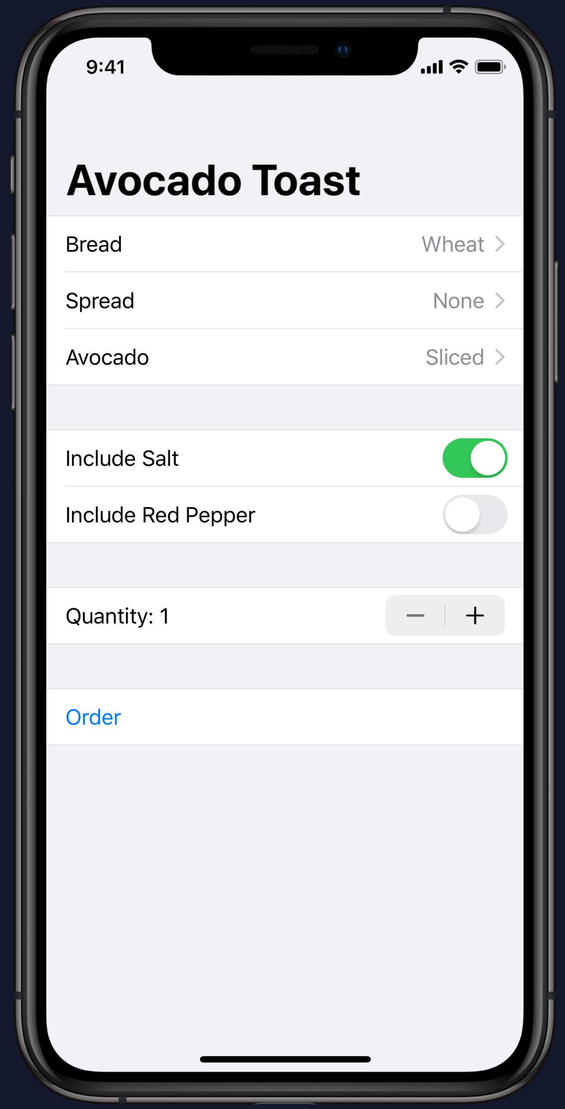
Multi-Grain Toast
5/20/19, 11:20 AM — S E

Sourdough with Chutney
5/19/19, 7:34 AM

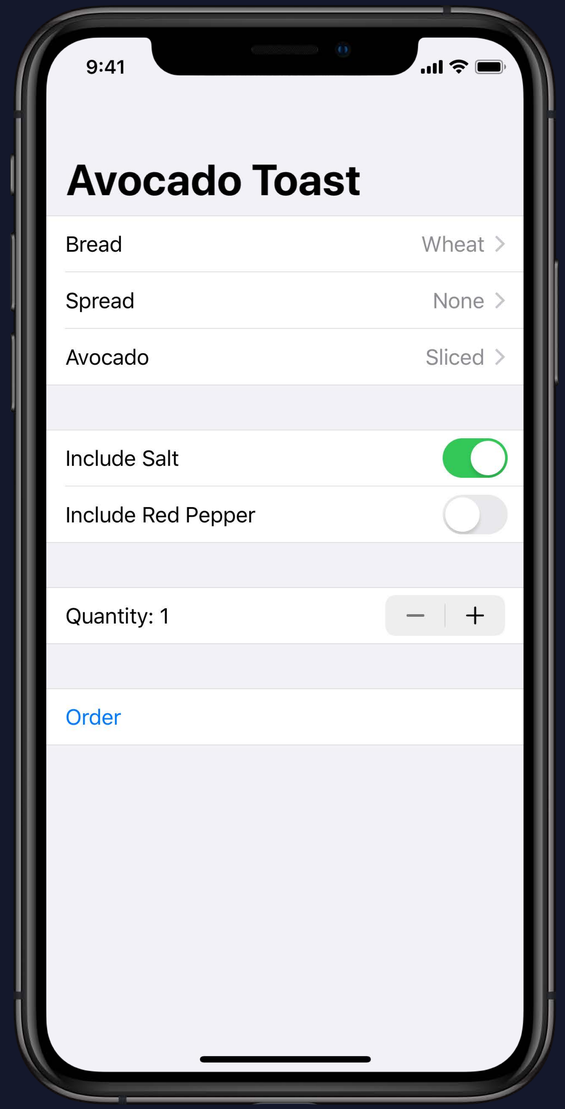
```
// Navigation View

struct ContentView : View {
    var body: some View {
        NavigationView {
            OrderForm()
        }
    }
}

struct OrderForm : View {
    var body: some View {
        Form {
            ...
        }.navigationBarTitle(Text("Avocado Toast"))
    }
}
```



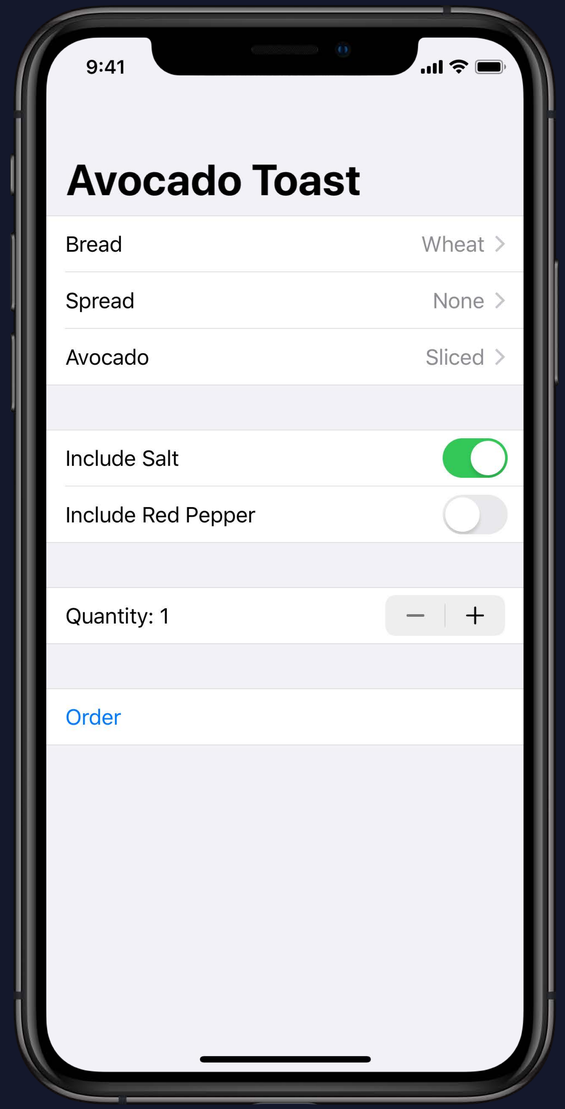
```
// Navigation View

struct ContentView : View {
    var body: some View {
        NavigationView {
            OrderForm()
        }
    }
}

struct OrderForm : View {
    var body: some View {
        Form {
            ...
        }.navigationBarTitle(Text("Avocado Toast"))
    }
}
```



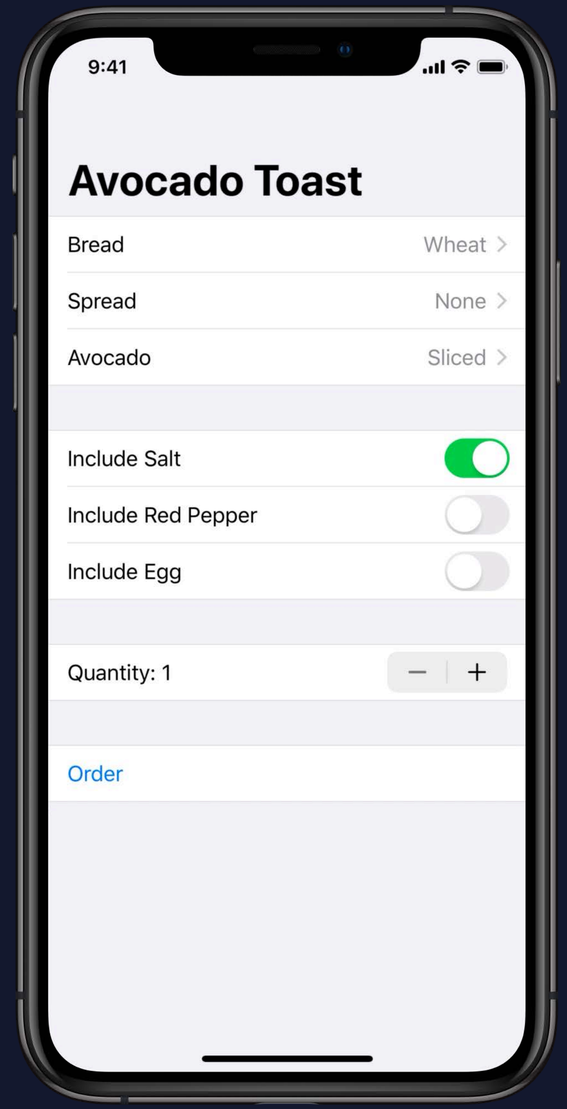
```
// Navigation View

struct ContentView : View {
    var body: some View {
        NavigationView {
            OrderForm()
        }
    }
}

struct OrderForm : View {
    var body: some View {
        Form {
            ...
        }.navigationBarTitle(Text("Avocado Toast"))
    }
}
```



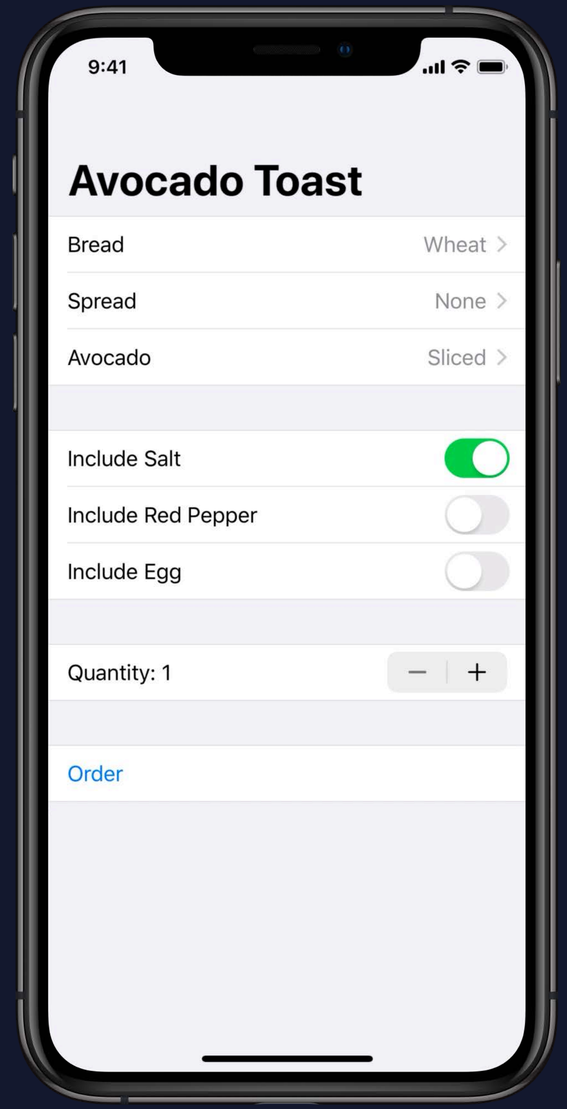
```
struct OrderForm : View {
    var body: some View {
        Form {
            ...
        }.navigationBarTitle(Text("Avocado Toast"))
    }
}
```



```
struct OrderForm : View {
    var body: some View {
        Form {
            ...
        }.navigationBarTitle(Text("Avocado Toast"))
    }
}
```



```
struct OrderForm : View {
    var body: some View {
        Form {
            ...
        }.navigationBarTitle(Text("Avocado Toast"))
    }
}
```



```
struct OrderForm : View {
    var body: some View {
        Form {
            Section { ••• }
            Section {
                Toggle(isOn: $order.includeSalt) { ••• }
                Toggle(isOn: $order.includeRedPepper) { ••• }
                Toggle(isOn: $order.includeEgg.animation()) { ••• }
                if order.includeEgg {
                    NavigationButton(destination: EggLocationPicker($order.eggLocation)) {
                        Text("Egg Location")
                    }
                }
            }
            Section { ••• }
            Section { ••• }
        }.navigationBarTitle(Text("Avocado Toast"))
    }
}
```

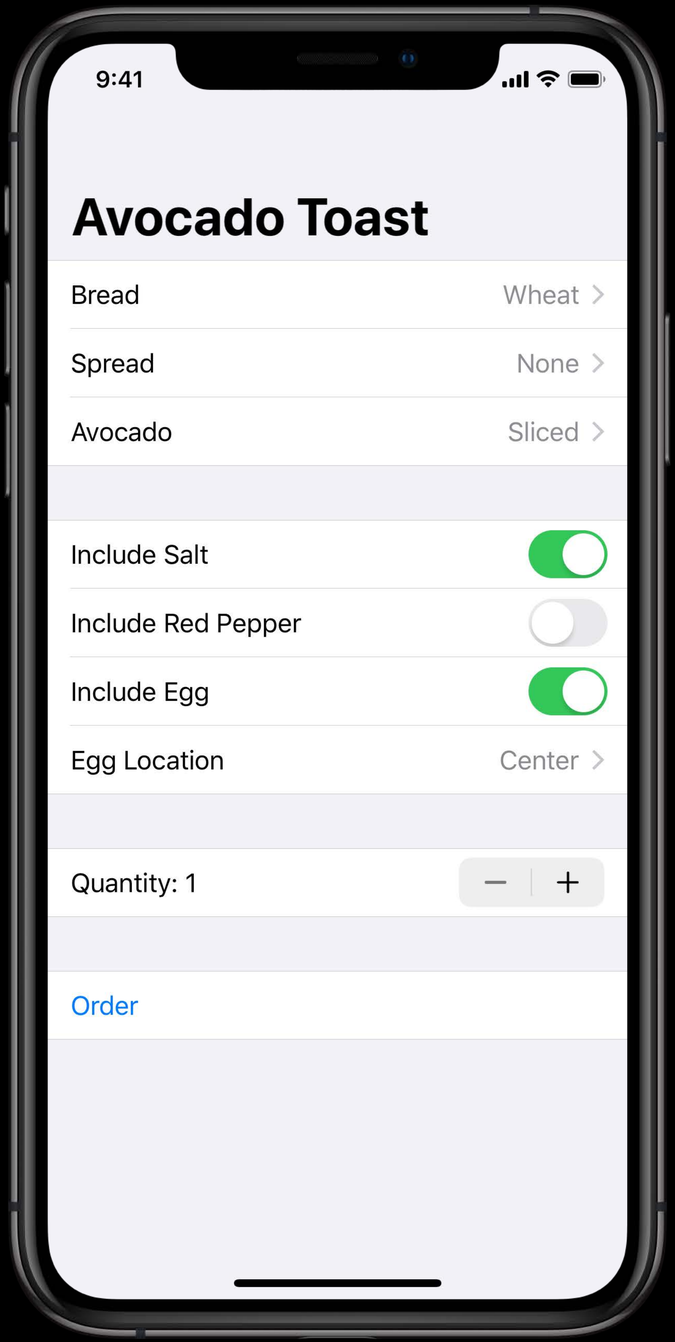
```
struct OrderForm : View {
    var body: some View {
        Form {
            Section { ... }
            Section {
                Toggle(isOn: $order.includeSalt) { ... }
                Toggle(isOn: $order.includeRedPepper) { ... }
                Toggle(isOn: $order.includeEgg.animation()) { ... }
                if order.includeEgg {
                    NavigationButton(destination: EggLocationPicker($order.eggLocation)) {
                        Text("Egg Location")
                    }
                }
            }
            Section { ... }
            Section { ... }
        }.navigationBarTitle(Text("Avocado Toast"))
    }
}
```

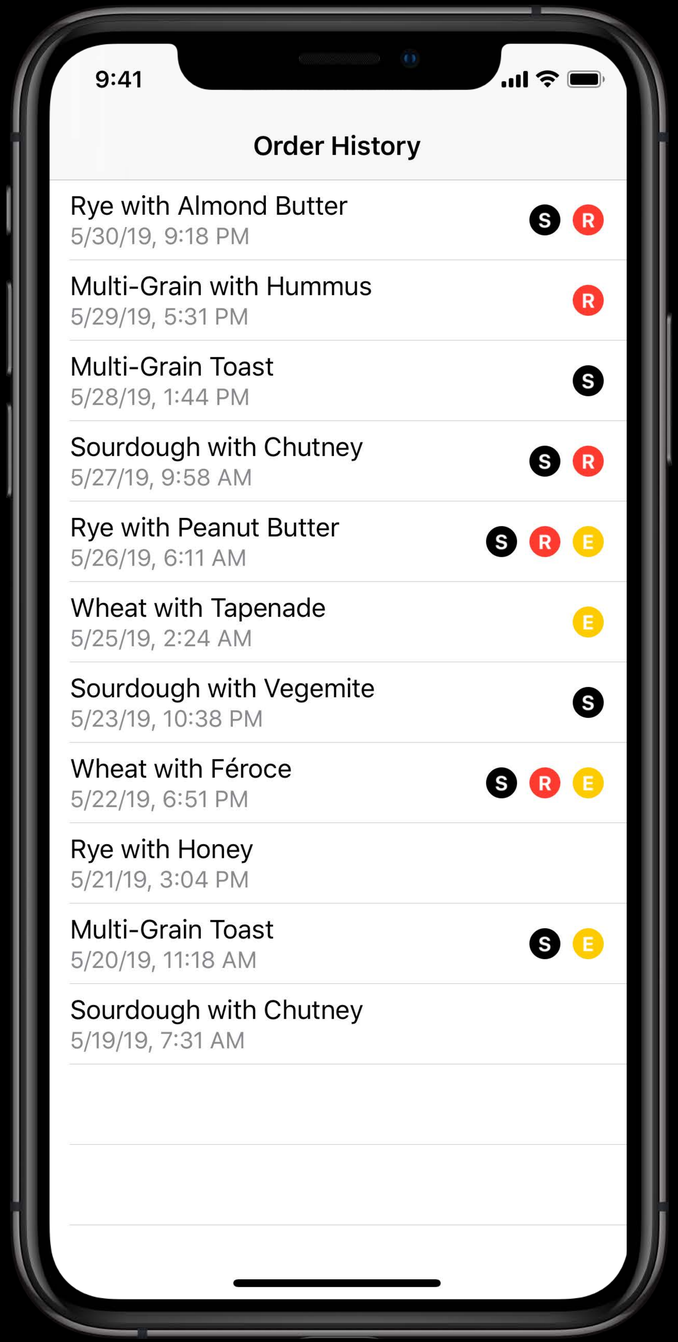
```
struct OrderForm : View {
    var body: some View {
        Form {
            Section { ••• }
            Section {
                Toggle(isOn: $order.includeSalt) { ••• }
                Toggle(isOn: $order.includeRedPepper) { ••• }
                Toggle(isOn: $order.includeEgg.animation()) { ••• }
                if order.includeEgg {
                    NavigationButton(destination: EggLocationPicker($order.eggLocation)) {
                        Text("Egg Location")
                    }
                }
            }
            Section { ••• }
            Section { ••• }
        }.navigationBarTitle(Text("Avocado Toast"))
    }
}
```

```
struct EggLocationPicker : View {
    @Binding var eggLocation: UnitPoint

    var body: some View {
        EggPlacementView($eggLocation)
            .navigationBarTitle(Text("Egg Location"))
            .navigationBarItems(trailing:
                Button(action: resetToCenter) {
                    Text("Reset")
            })
    }

  private func resetToCenter() { ... }
}
```

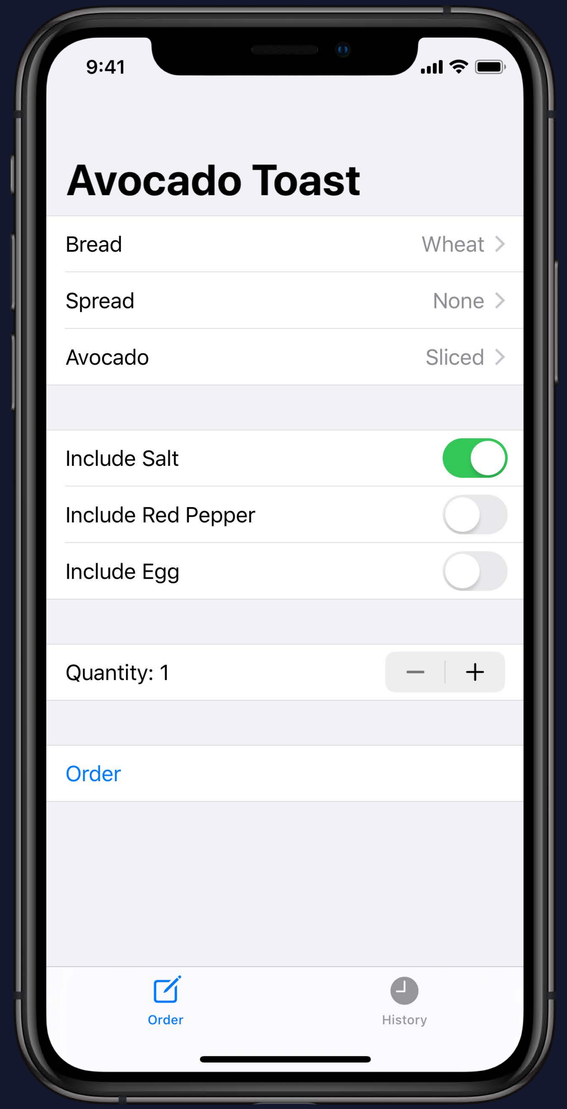
```
struct EggLocationPicker : View {
    @Binding var eggLocation: UnitPoint

    var body: some View {
        EggPlacementView($eggLocation)
            .navigationBarTitle(Text("Egg Location"))
            .navigationBarItems(trailing:
                Button(action: resetToCenter) {
                    Text("Reset")
            })
    }

  private func resetToCenter() { ... }
}
```

```
struct EggLocationPicker : View {
    @Binding var eggLocation: UnitPoint

    var body: some View {
        EggPlacementView($eggLocation)
            .navigationBarTitle(Text("Egg Location"))
            .navigationBarItems(trailing:
                Button(action: resetToCenter) {
                    Text("Reset")
            })
    }

    private func resetToCenter() { ... }
}
```

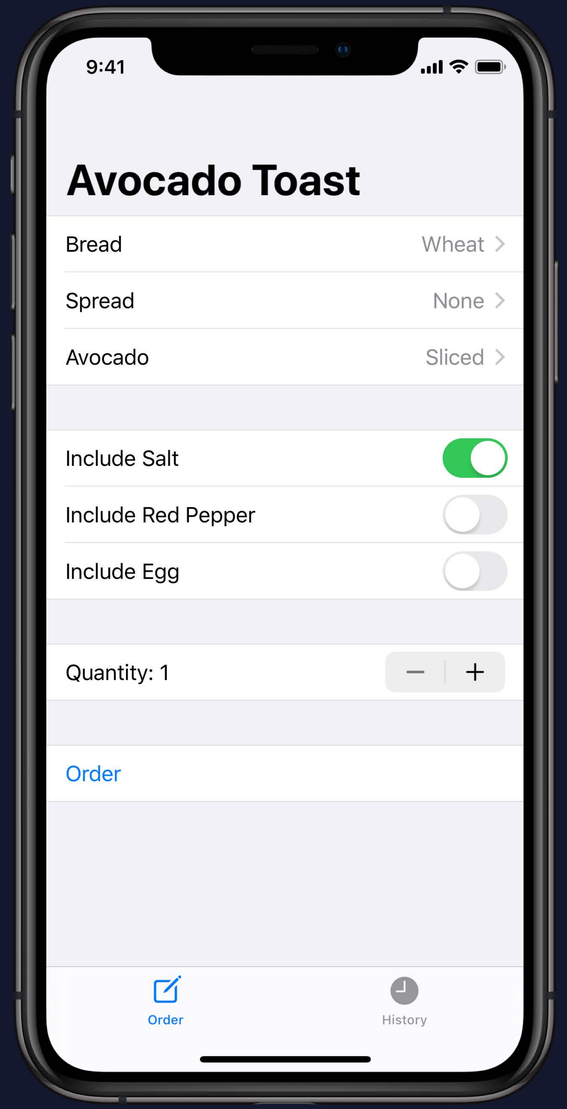

9:41
Avocado Toast
Bread
Wheat
Spread
None
Avocado
Sliced
Include Salt
Include Red Pepper
Include Egg
Egg Location
Center
Quantity: 1
Order


9:41
Order History
Rye with Almond Butter
5/30/19, 9:18 PM
Multi-Grain with Hummus
5/29/19, 5:31 PM
Multi-Grain Toast
5/28/19, 1:44 PM
Sourdough with Chutney
5/27/19, 9:58 AM
Rye with Peanut Butter
5/26/19, 6:11 AM
Wheat with Tapenade
5/25/19, 2:24 AM
Sourdough with Vegemite
5/23/19, 10:38 PM
Wheat with Féroce
5/22/19, 6:51 PM
Rye with Honey
5/21/19, 3:04 PM
Multi-Grain Toast
5/20/19, 11:18 AM
Sourdough with Chutney
5/19/19, 7:31 AM

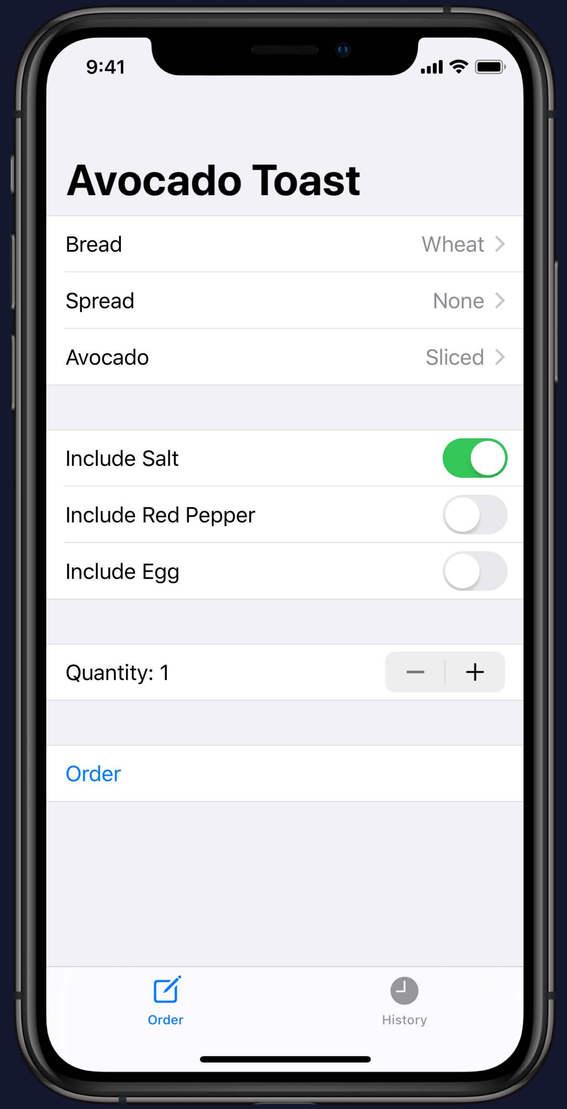
```
struct ContentView : View {
    var body: some View {
        NavigationView {
            TabbedView {
                OrderForm()
                    .tabItemLabel {
                        Image(systemName: "square.and.pencil")
                        Text("New Order")
                    }
                OrderHistory()
                    .tabItemLabel {
                        Image(systemName: "clock.fill")
                        Text("History")
                    }
            }
        }
    }
}
```



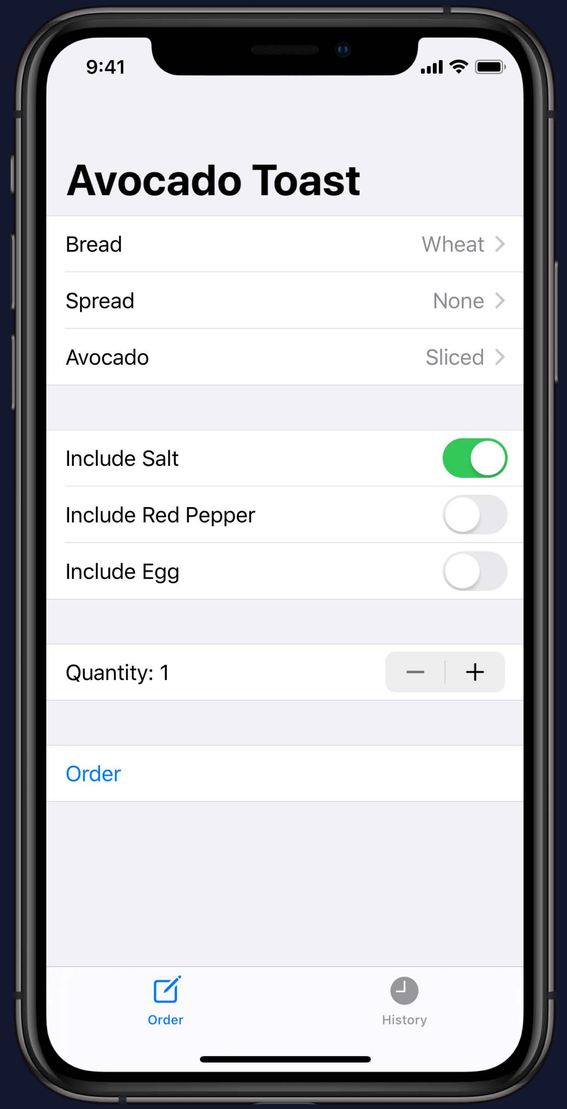
```
struct ContentView : View {
    var body: some View {
        NavigationView {
            TabbedView {
                OrderForm()
                    .tabItemLabel {
                        Image(systemName: "square.and.pencil")
                        Text("New Order")
                    }
                OrderHistory()
                    .tabItemLabel {
                        Image(systemName: "clock.fill")
                        Text("History")
                    }
            }
        }
    }
}
```



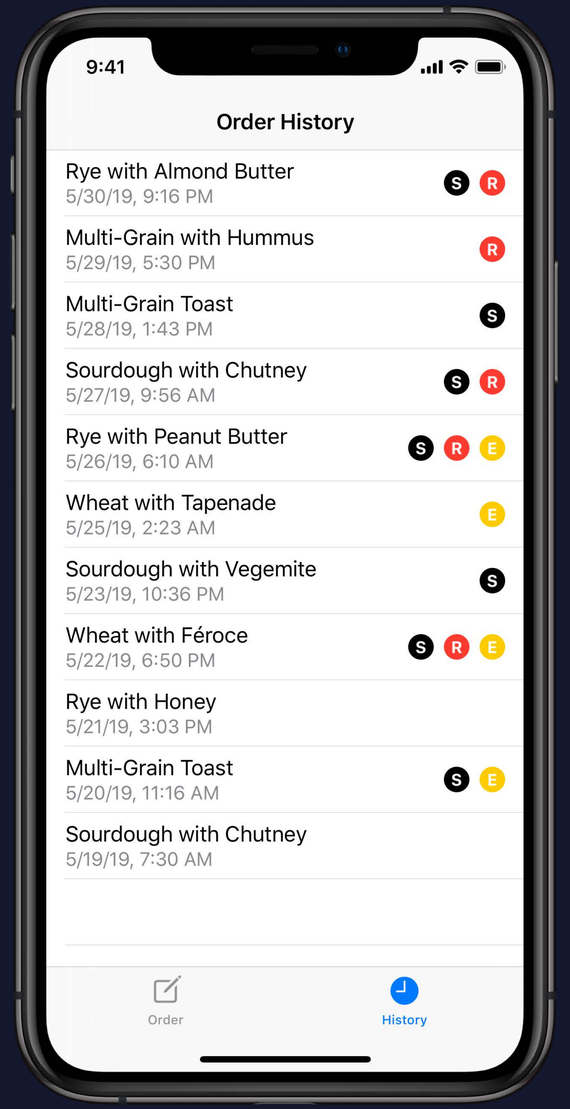
```
struct ContentView : View {
    var body: some View {
        NavigationView {
            TabbedView {
                OrderForm()
                    .tabItemLabel {
                        Image(systemName: "square.and.pencil")
                        Text("New Order")
                    }
                OrderHistory()
                    .tabItemLabel {
                        Image(systemName: "clock.fill")
                        Text("History")
                    }
            }
        }
    }
}
```

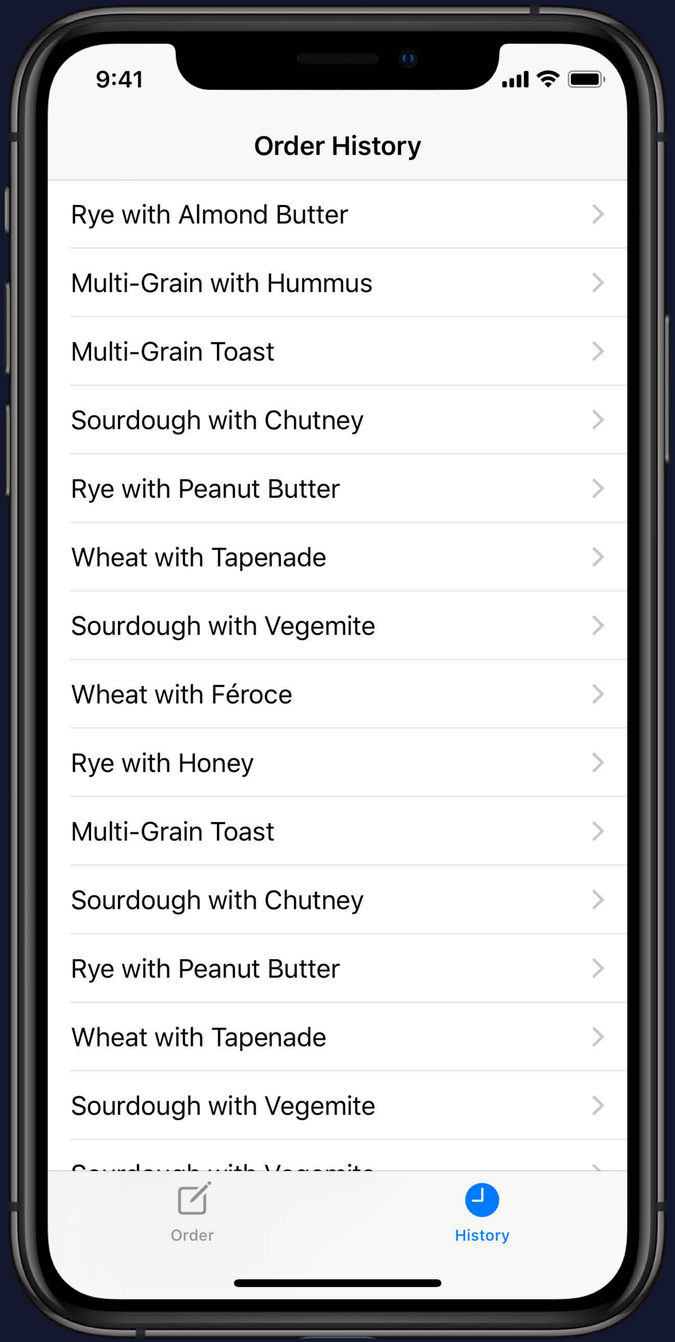


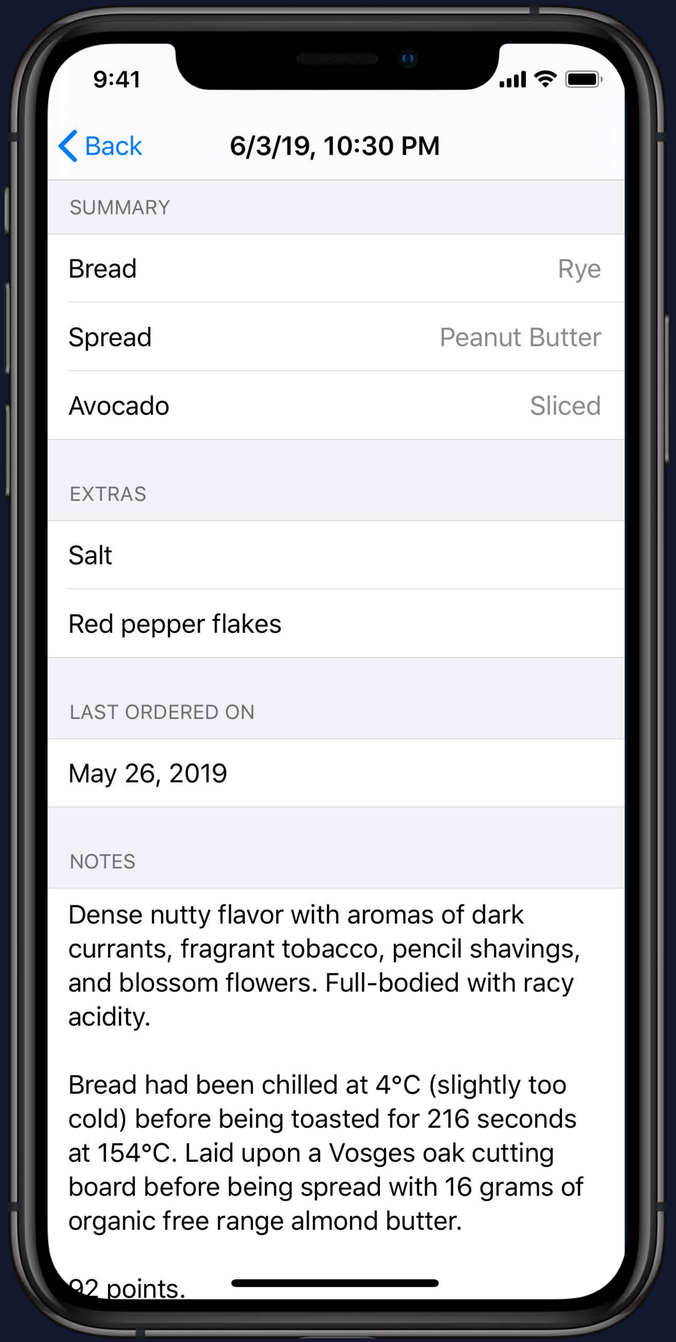
```
struct ContentView : View {
    var body: some View {
        NavigationView {
            TabbedView {
                OrderForm()
                    .tabItemLabel {
                        Image(systemName: "square.and.pencil")
                        Text("New Order")
                    }
                OrderHistory()
                    .tabItemLabel {
                        Image(systemName: "clock.fill")
                        Text("History")
                    }
            }
        }
    }
}
```



```
struct ContentView : View {
    var body: some View {
        NavigationView {
            TabbedView {
                OrderForm()
                    .tabItemLabel {
                        Image(systemName: "square.and.pencil")
                        Text("New Order")
                    }
                OrderHistory()
                    .tabItemLabel {
                        Image(systemName: "clock.fill")
                        Text("History")
                    }
            }
        }
    }
}
```

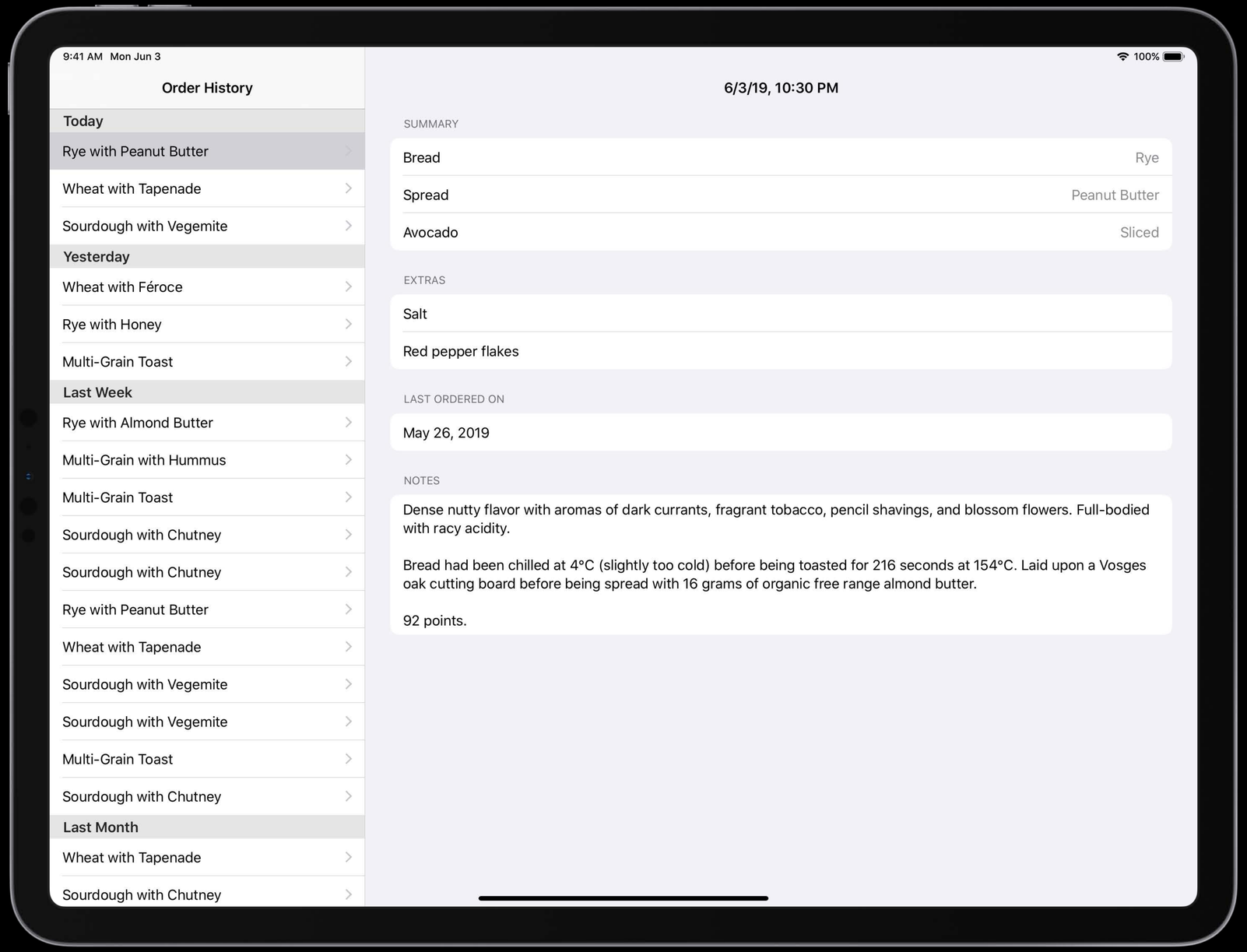



9:41
Order History
Rye with Almond Butter
Multi-Grain with Hummus
Multi-Grain Toast
Sourdough with Chutney
Rye with Peanut Butter
Wheat with Tapenade
Sourdough with Vegemite
Wheat with Féroce
Rye with Honey
Multi-Grain Toast
Sourdough with Chutney
Rye with Peanut Butter
Wheat with Tapenade
Sourdough with Vegemite
Order
History


9:41
Back
6/3/19, 10:30 PM
SUMMARY
Bread
Rye
Spread
Peanut Butter
Avocado
Sliced
EXTRAS
Salt
Red pepper flakes
LAST ORDERED ON
May 26, 2019
NOTES
Dense nutty flavor with aromas of dark currants, fragrant tobacco, pencil shavings, and blossom flowers. Full-bodied with racy acidity.
Bread had been chilled at 4°C (slightly too cold) before being toasted for 216 seconds at 154°C. Laid upon a Vosges oak cutting board before being spread with 16 grams of organic free range almond butter.
92 points.

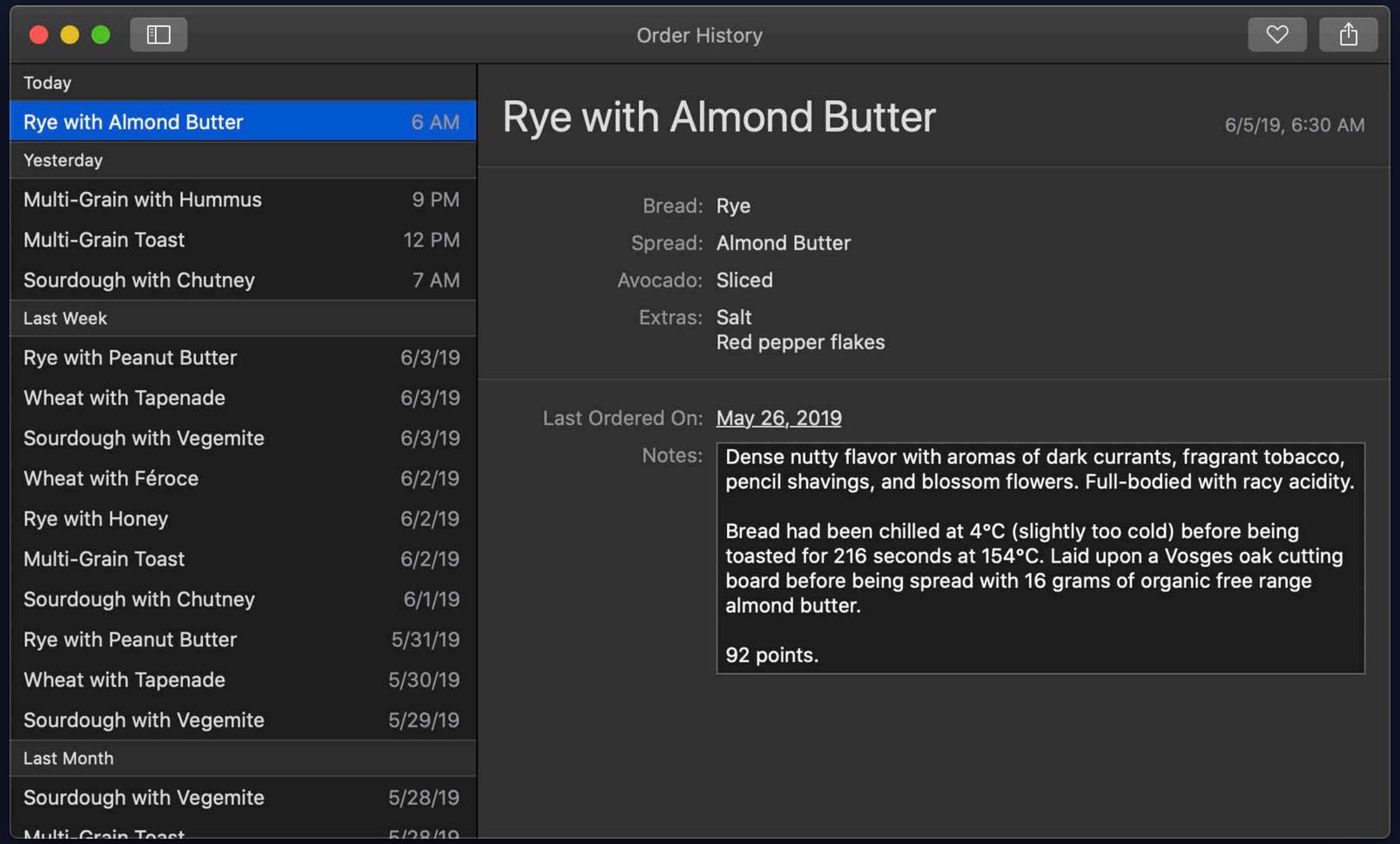
```
struct OrderHistory : View {
   let previousOrders: [CompletedOrder]

   var body: some View {
      List(previousOrders) { order in
         OrderDetail(order: order)
      }
   }
}



struct OrderDetail : View { ••• }
```

```
struct OrderHistory : View {
    let previousOrders: [CompletedOrder]

    var body: some View {
        List(previousOrders) { order in
            NavigationButton(destination: OrderDetail(order: order)) {
                Text(order.summary)
            }
        }
    }
}

struct OrderDetail : View { ••• }
```

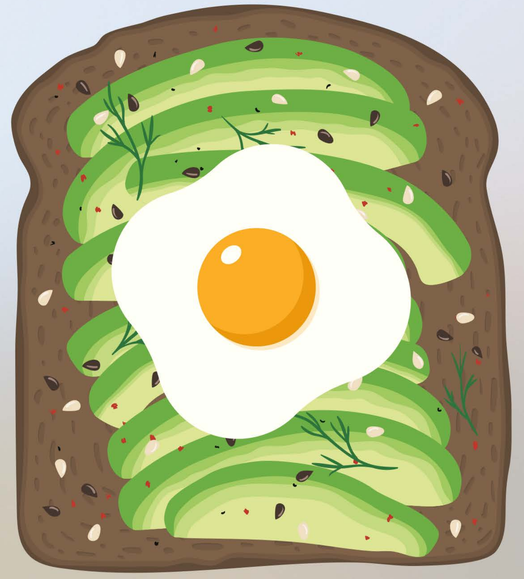

9:41 AM Mon Jun 3
100%
Order History
Today
Rye with Peanut Butter
Wheat with Tapenade
Sourdough with Vegemite
Yesterday
Wheat with Féroce
Rye with Honey
Multi-Grain Toast
Last Week
Rye with Almond Butter
Multi-Grain with Hummus
Multi-Grain Toast
Sourdough with Chutney
Sourdough with Chutney
Rye with Peanut Butter
Wheat with Tapenade
Sourdough with Vegemite
Sourdough with Vegemite
Multi-Grain Toast
Sourdough with Chutney
Last Month
Wheat with Tapenade
Sourdough with Chutney
6/3/19, 10:30 PM
SUMMARY
Bread
Rye
Spread
Peanut Butter
Avocado
Sliced
EXTRAS
Salt
Red pepper flakes
LAST ORDERED ON
May 26, 2019
NOTES
Dense nutty flavor with aromas of dark currants, fragrant tobacco, pencil shavings, and blossom flowers. Full-bodied with racy acidity.
Bread had been chilled at 4°C (slightly too cold) before being toasted for 216 seconds at 154°C. Laid upon a Vosges oak cutting board before being spread with 16 grams of organic free range almond butter.
92 points.

```
struct ContentView : View {
    let previousOrders: [CompletedOrder]
    var body: some View {
        NavigationView {
            OrderHistory(previousOrders: previousOrders)
        }
    }
}


struct OrderHistory : View { ••• }

struct OrderDetail : View { ••• }
```

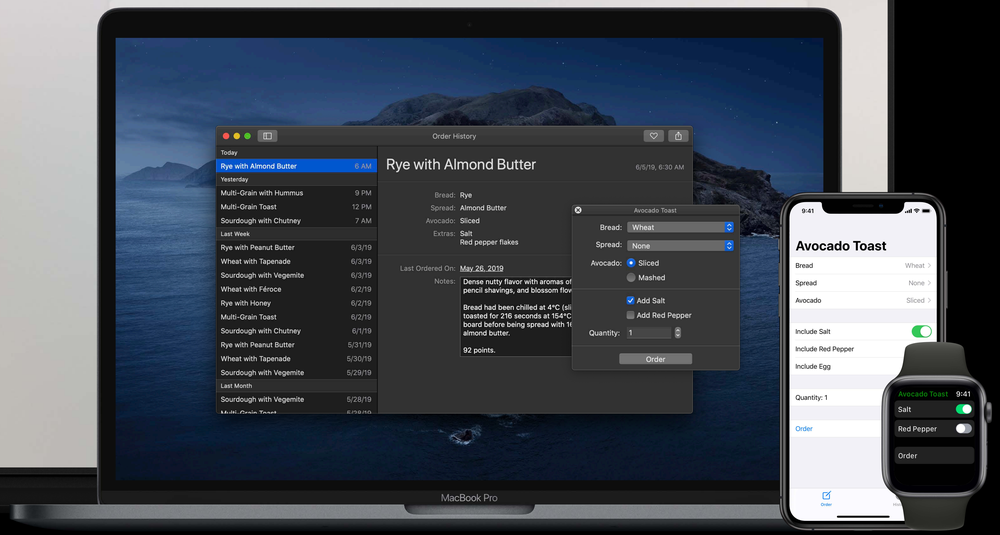
```
struct ContentView : View {
    let previousOrders: [CompletedOrder]
    var body: some View {
        NavigationView {
            OrderHistory(previousOrders: previousOrders)
            OrderDetailPlaceholder()
        }
    }
}

struct OrderHistory : View { ••• }

struct OrderDetail : View { ••• }
```


Order History
Today
Rye with Almond Butter 6 AM
Yesterday
Multi-Grain with Hummus 9 PM
Multi-Grain Toast 12 PM
Sourdough with Chutney 7 AM
Last Week
Rye with Peanut Butter 6/3/19
Wheat with Tapenade 6/3/19
Sourdough with Vegemite 6/3/19
Wheat with Féroce 6/2/19
Rye with Honey 6/2/19
Multi-Grain Toast 6/2/19
Sourdough with Chutney 6/1/19
Rye with Peanut Butter 5/31/19
Wheat with Tapenade 5/30/19
Sourdough with Vegemite 5/29/19
Last Month
Sourdough with Vegemite 5/28/19
Rye with Almond Butter
6/5/19, 6:30 AM
Bread: Rye
Spread: Almond Butter
Avocado: Sliced
Extras: Salt
Red pepper flakes
Last Ordered On: May 26, 2019
Notes: Dense nutty flavor with aromas of dark currants, fragrant tobacco, pencil shavings, and blossom flowers. Full-bodied with racy acidity.
Bread had been chilled at 4°C (slightly too cold) before being toasted for 216 seconds at 154°C. Laid upon a Vosges oak cutting board before being spread with 16 grams of organic free range almond butter.
92 points.

Avocado Toast



Bread Type	Wheat
Spread	None
Avocado	Sliced



Introducing SwiftUI

SwiftUI Essentials

Data Flow
Through SwiftUI

Introducing SwiftUI

SwiftUI Essentials

Data Flow
Through SwiftUI

Building Custom Views
with SwiftUI

Introducing SwiftUI

SwiftUI Essentials

Data Flow
Through SwiftUI

Building Custom Views
with SwiftUI

Introducing SwiftUI

SwiftUI Essentials

Integrating SwiftUI

Introducing SwiftUI

SwiftUI Essentials

Data Flow
Through SwiftUI

Building Custom Views
with SwiftUI

Integrating SwiftUI

Accessibility in SwiftUI

Introducing SwiftUI

SwiftUI Essentials

Data Flow
Through SwiftUI

Building Custom Views
with SwiftUI

Integrating SwiftUI

Accessibility in SwiftUI

SwiftUI on All Devices

More Information

developer.apple.com/wwdc19/216

SwiftUI on watchOS	Wednesday, 2:00
Modern Swift API Design	Thursday, 2:00
What’s New in Swift	WWDC 2019


WWDC19