

App Structure

Lecture 5

CIS 1951

<https://github.com/cis1951/lec5-code>



Previously, on CIS 1951...

SwiftUI State Management

- View hierarchy
- Property wrappers
 - @State, @Binding
 - @ObservedObject, @StateObject, @EnvironmentObject
- .onChange modifier
- Animations and transitions
- **Questions? Comments?**

**So far, we've only made simple,
single-screen apps.**

That changes today.

This week

The tools you need to create larger apps

Navigation & modal presentations

MVVM

Lifecycle events

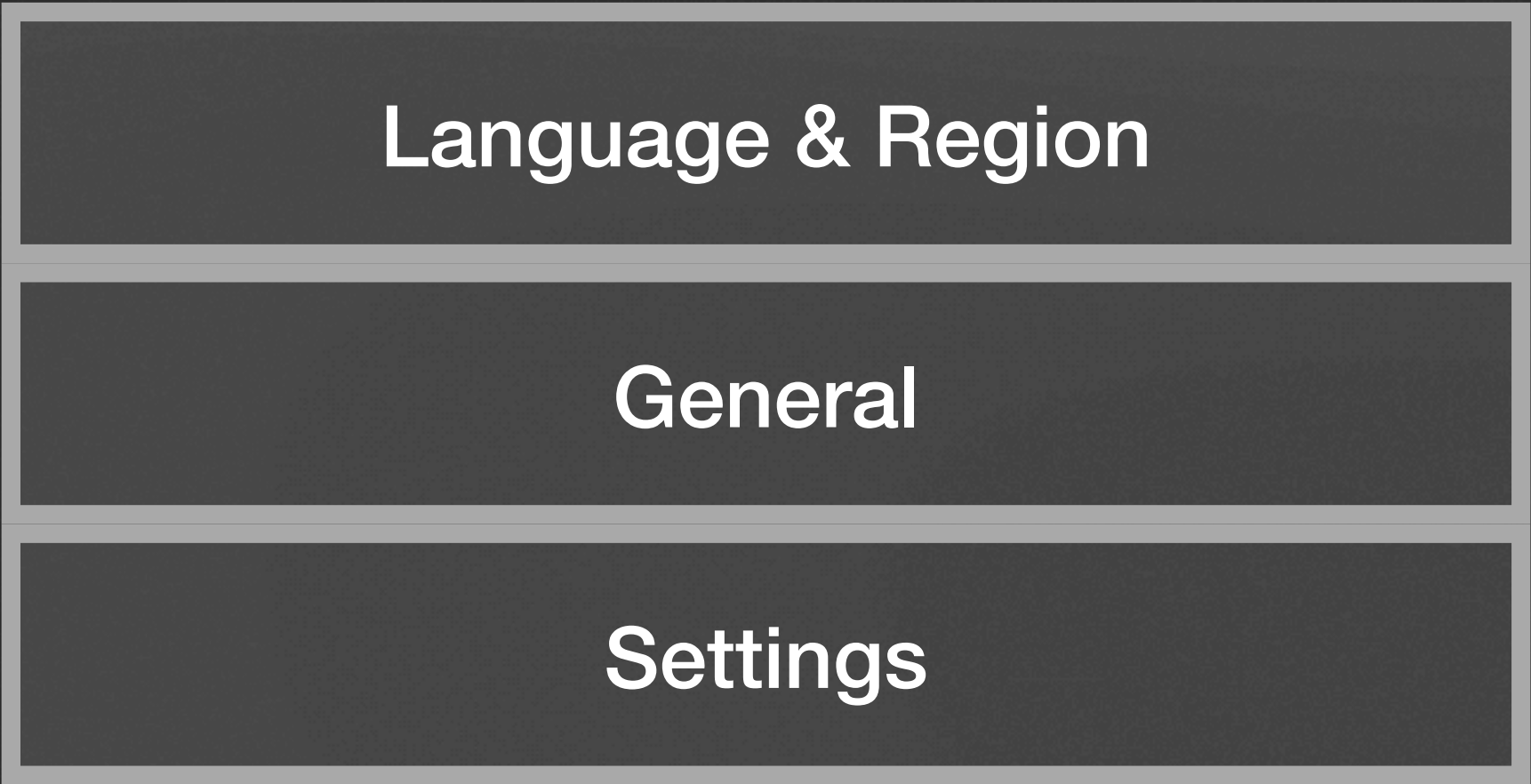
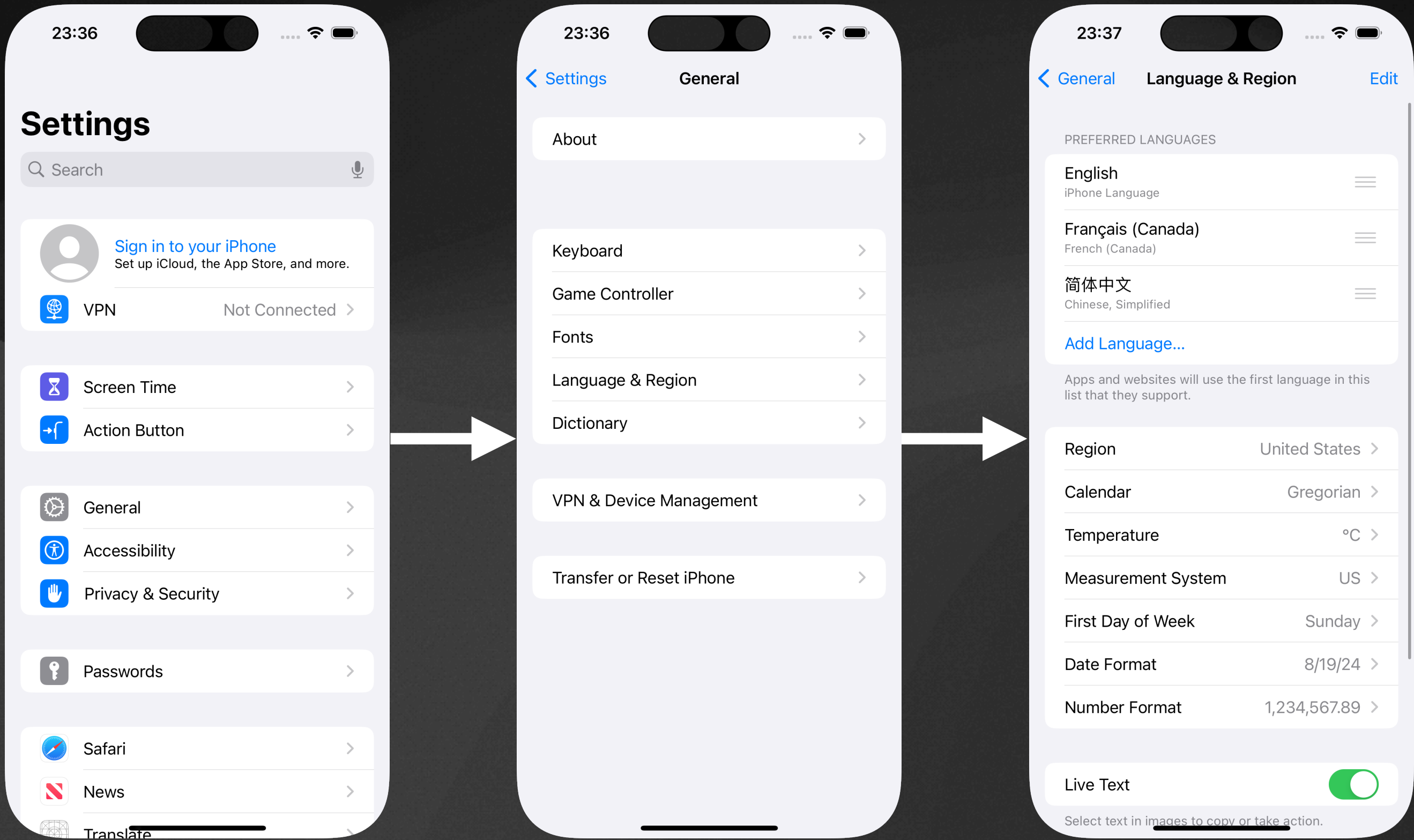
Navigation & Modal Presentations



How do we organize multiple
screens?

Hierarchical Navigation

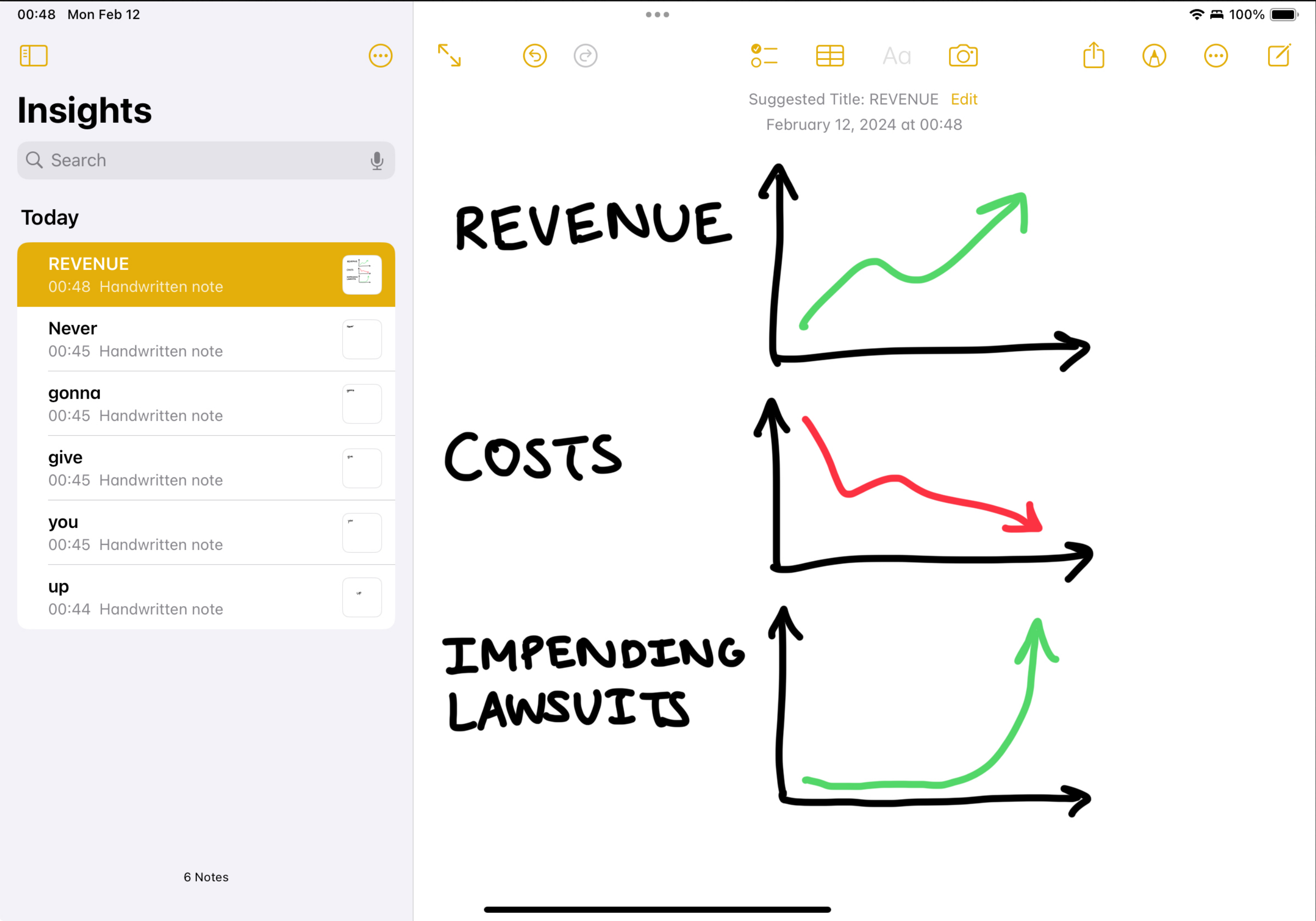
Example: Settings App



Master-Detail Navigation

Example: Notes App

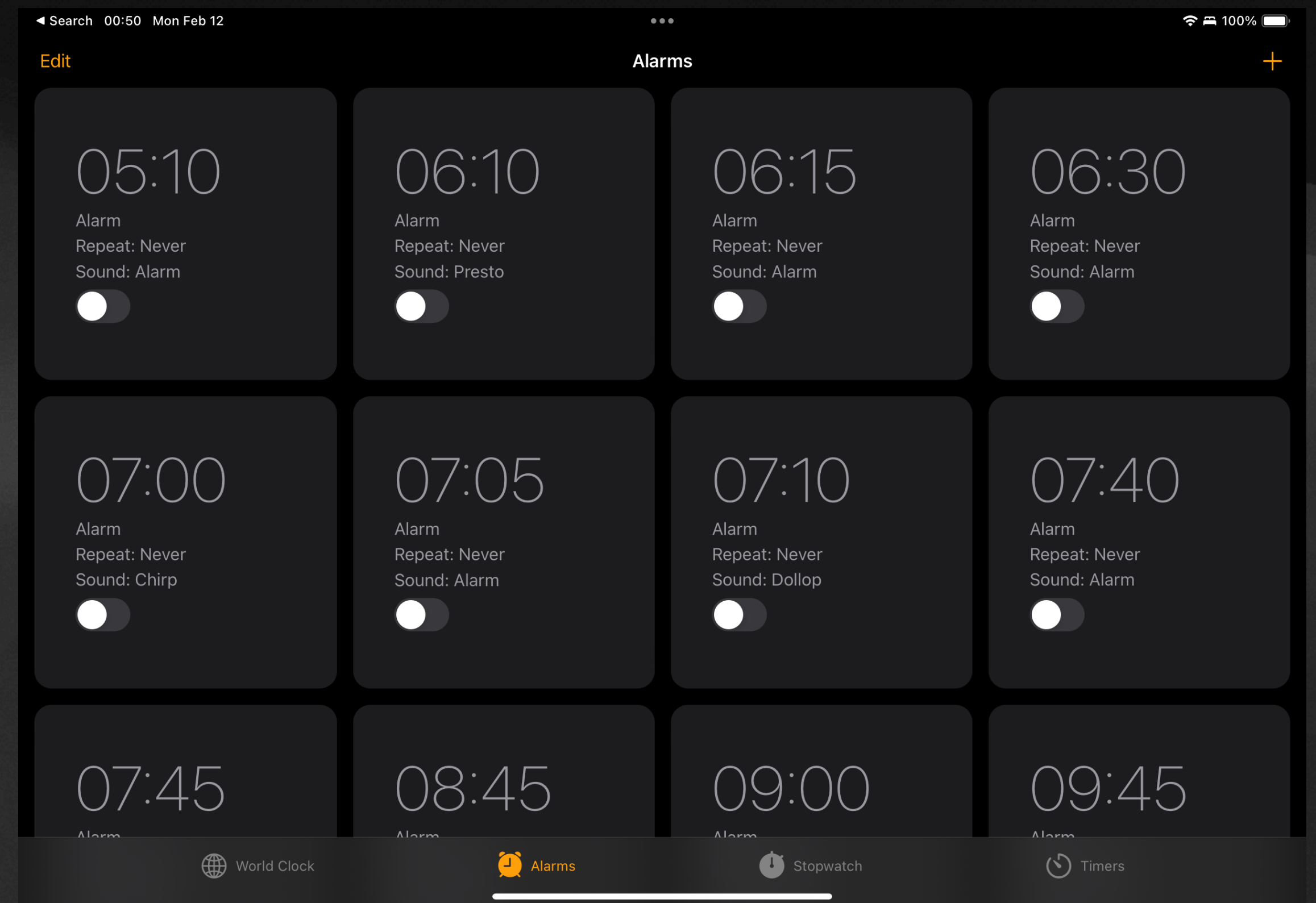
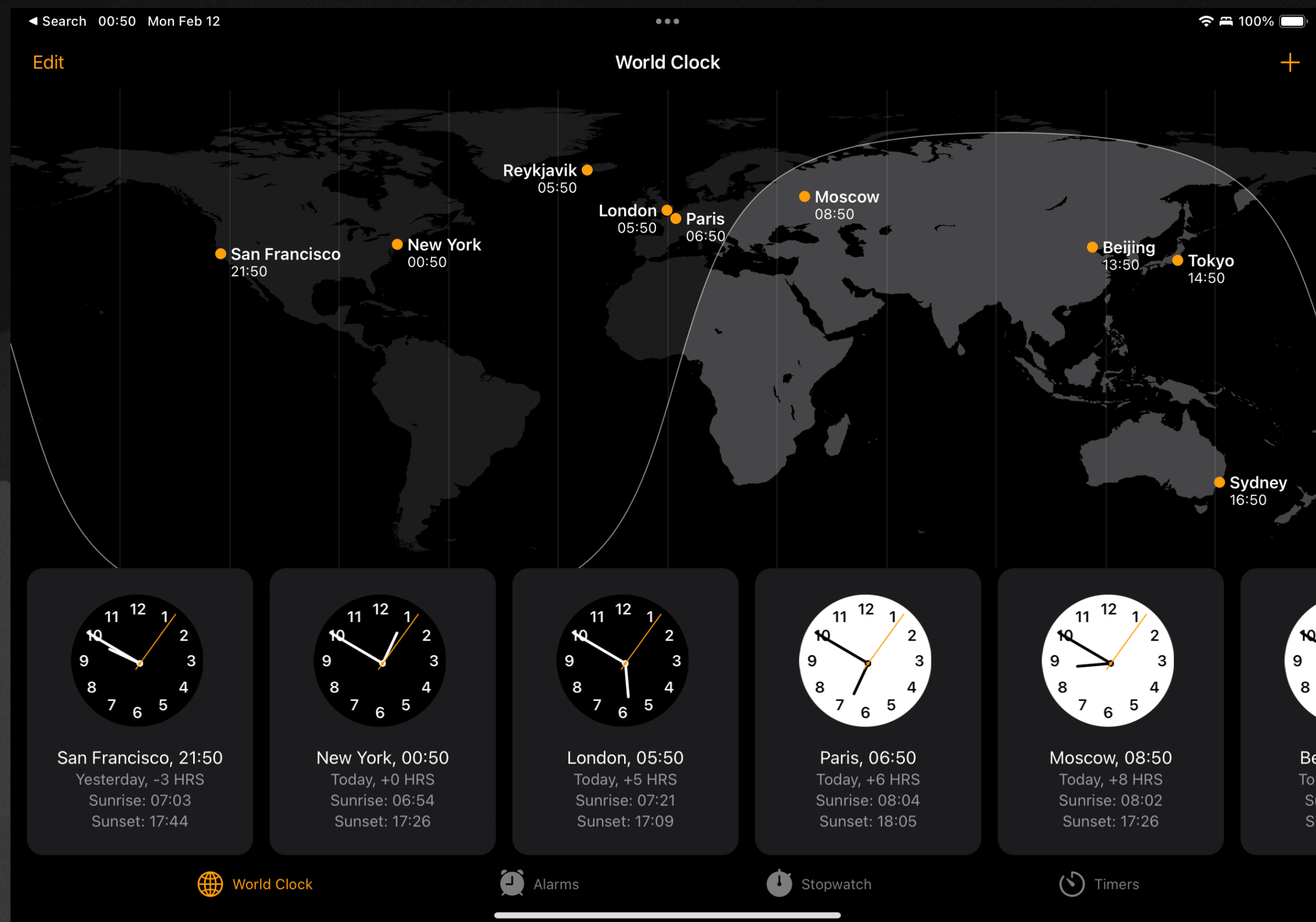
Master
List of notes



Detail
Single note

Tab Bar Navigation

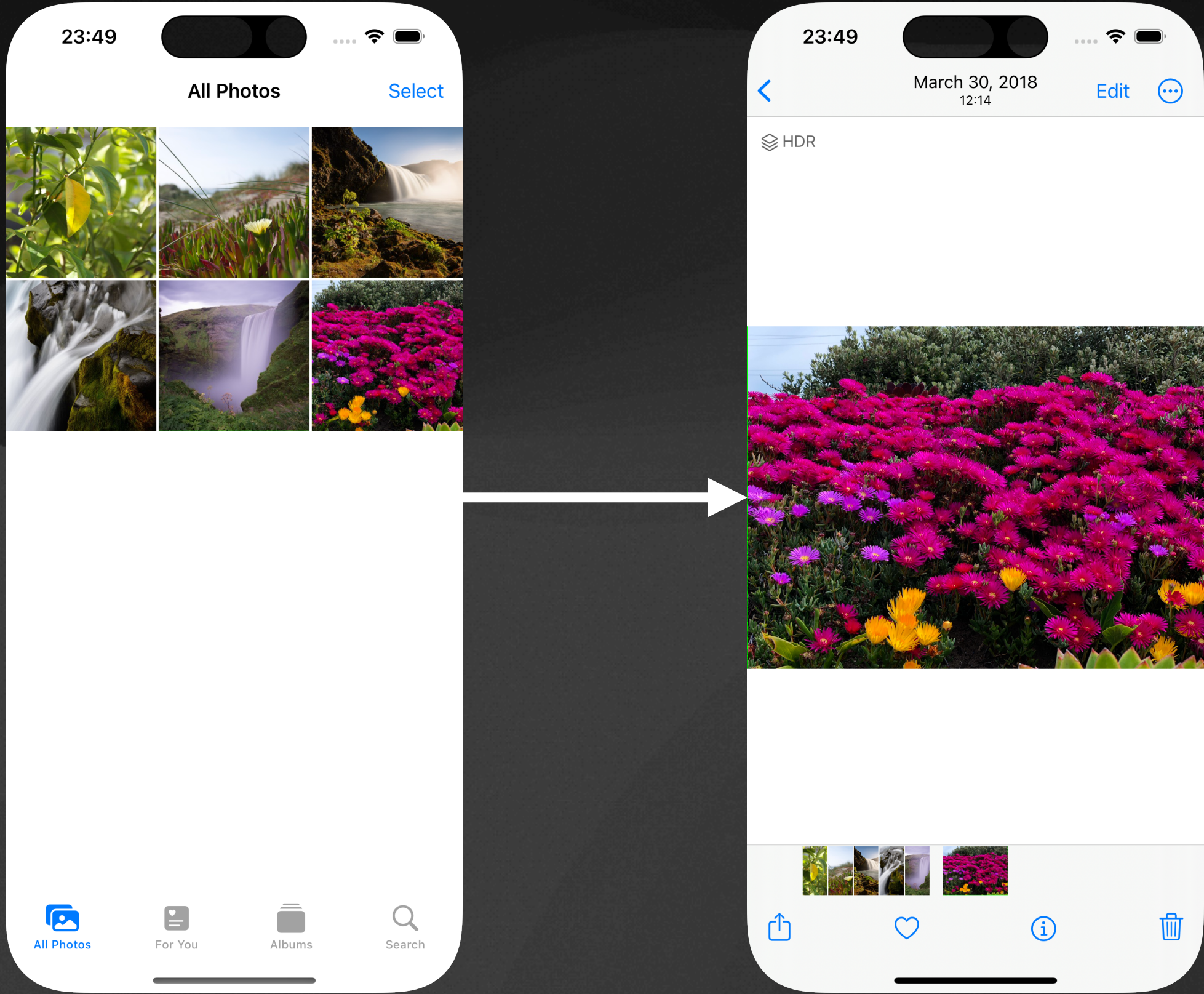
Example: Clock App



Bottom bar for quick navigation

Hybrid Navigation

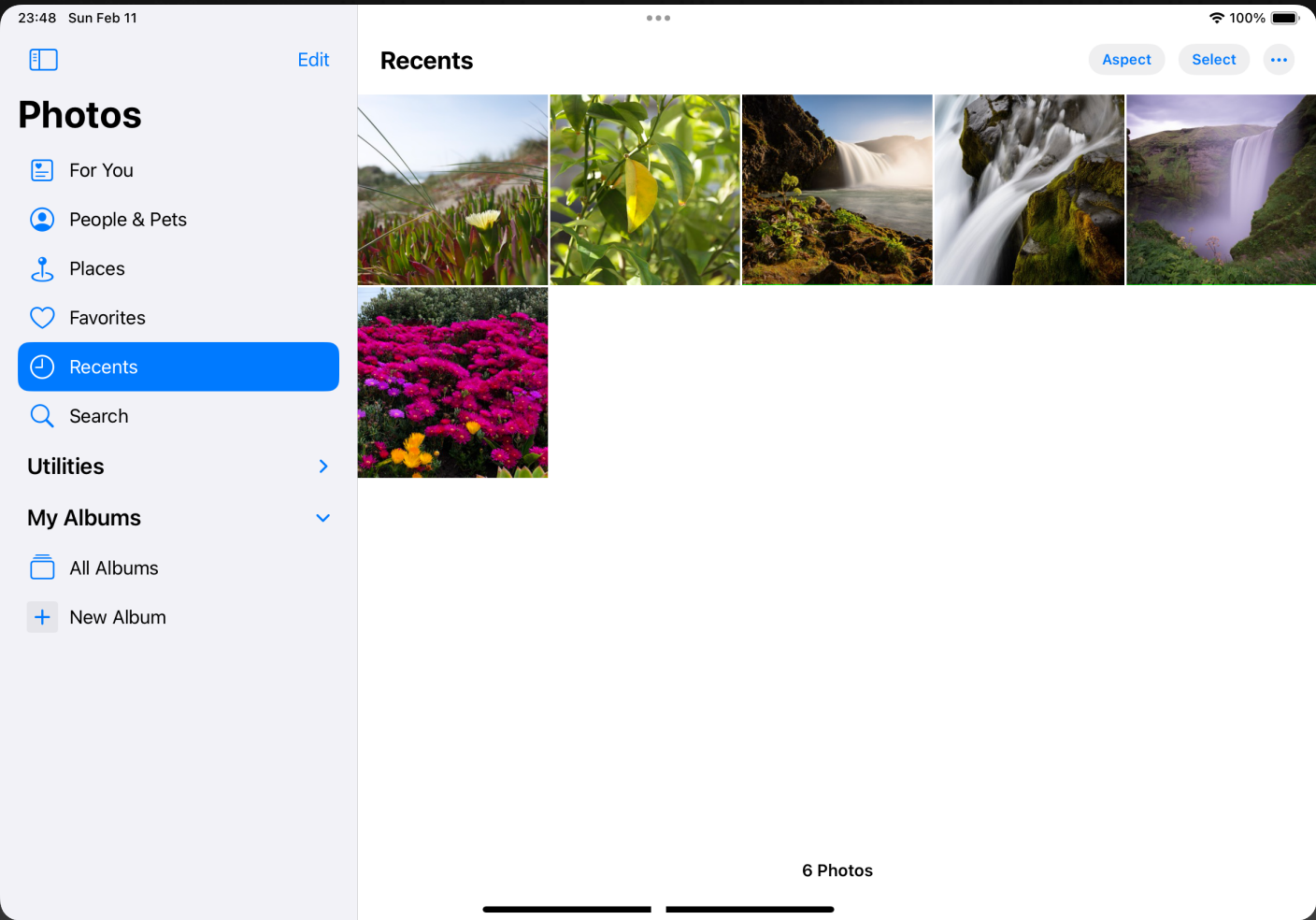
Example: Photos App



Tab bar

Hierarchical

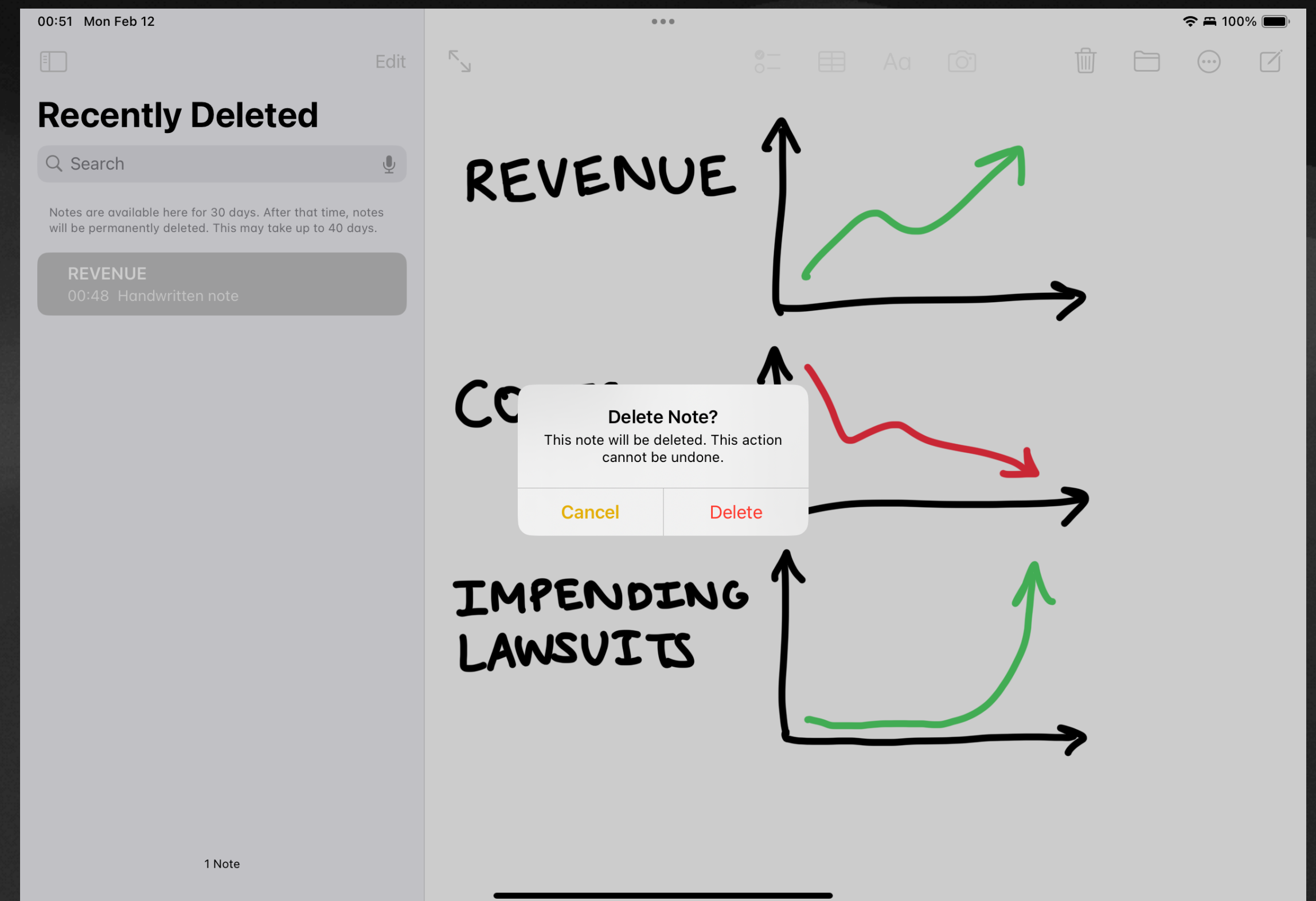
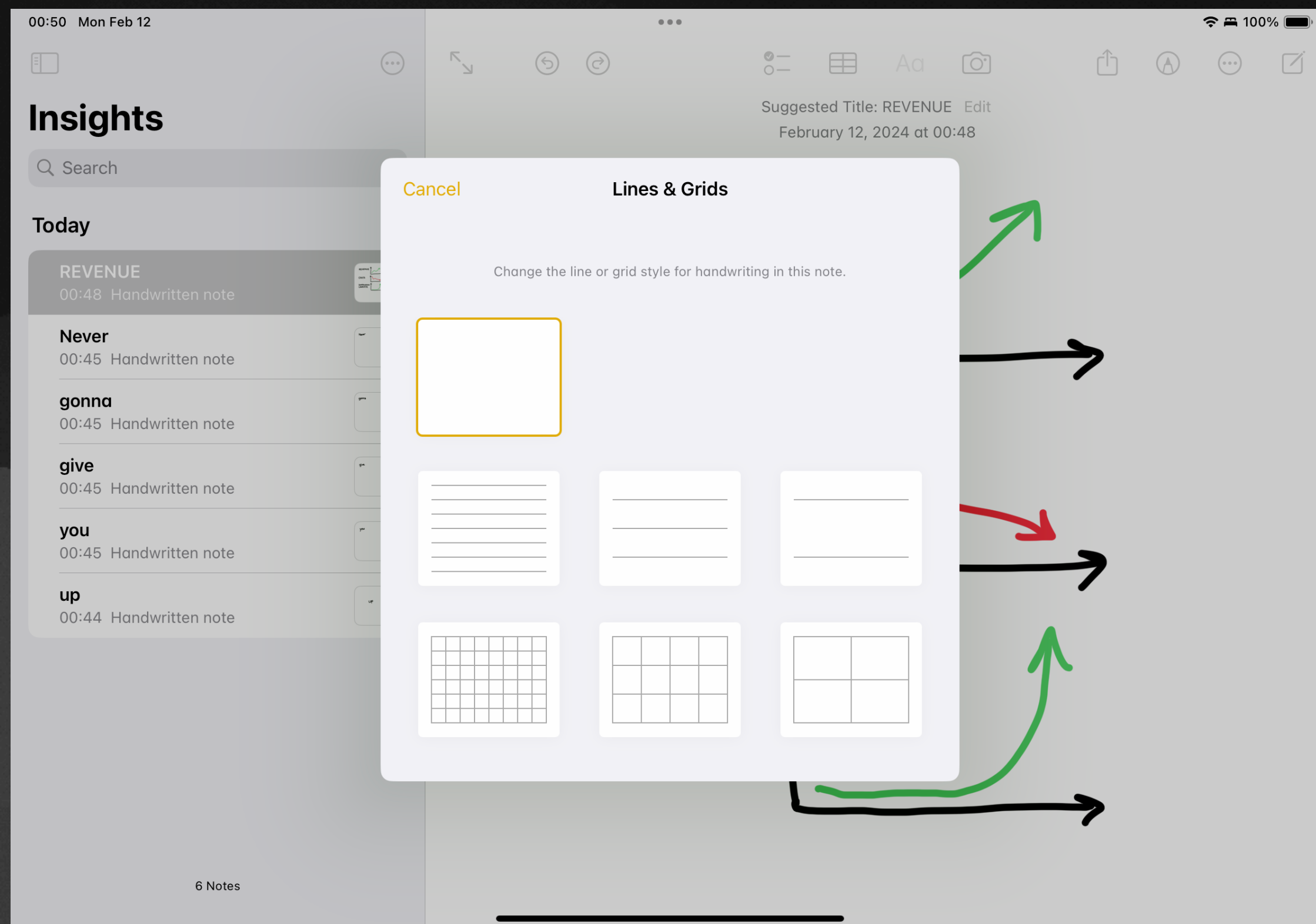
Master-detail



Hierarchical

Modals

Example: Notes App



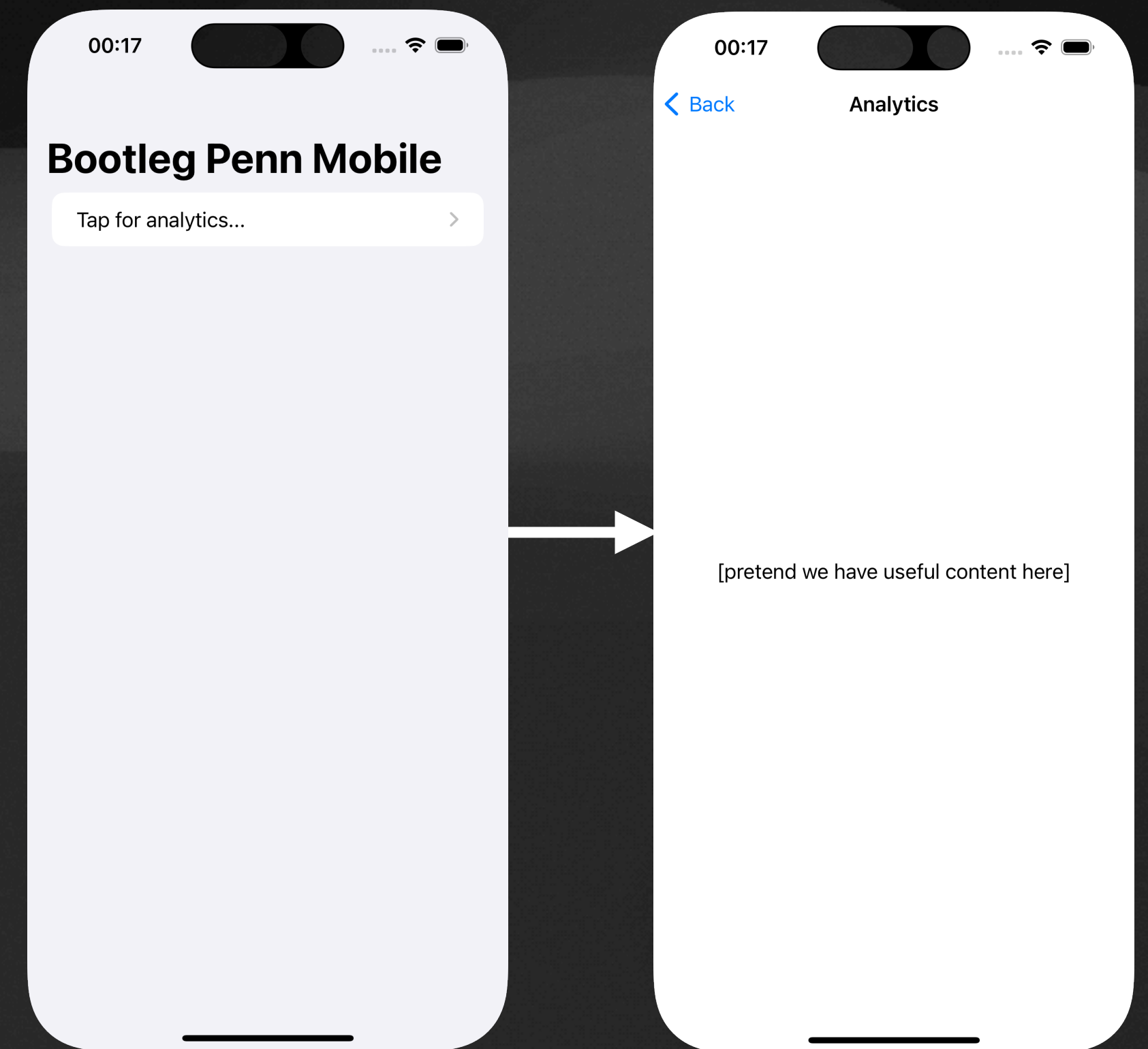
Lightweight and focused interactions

Implementation

NavigationView

Deprecated in recent versions of iOS

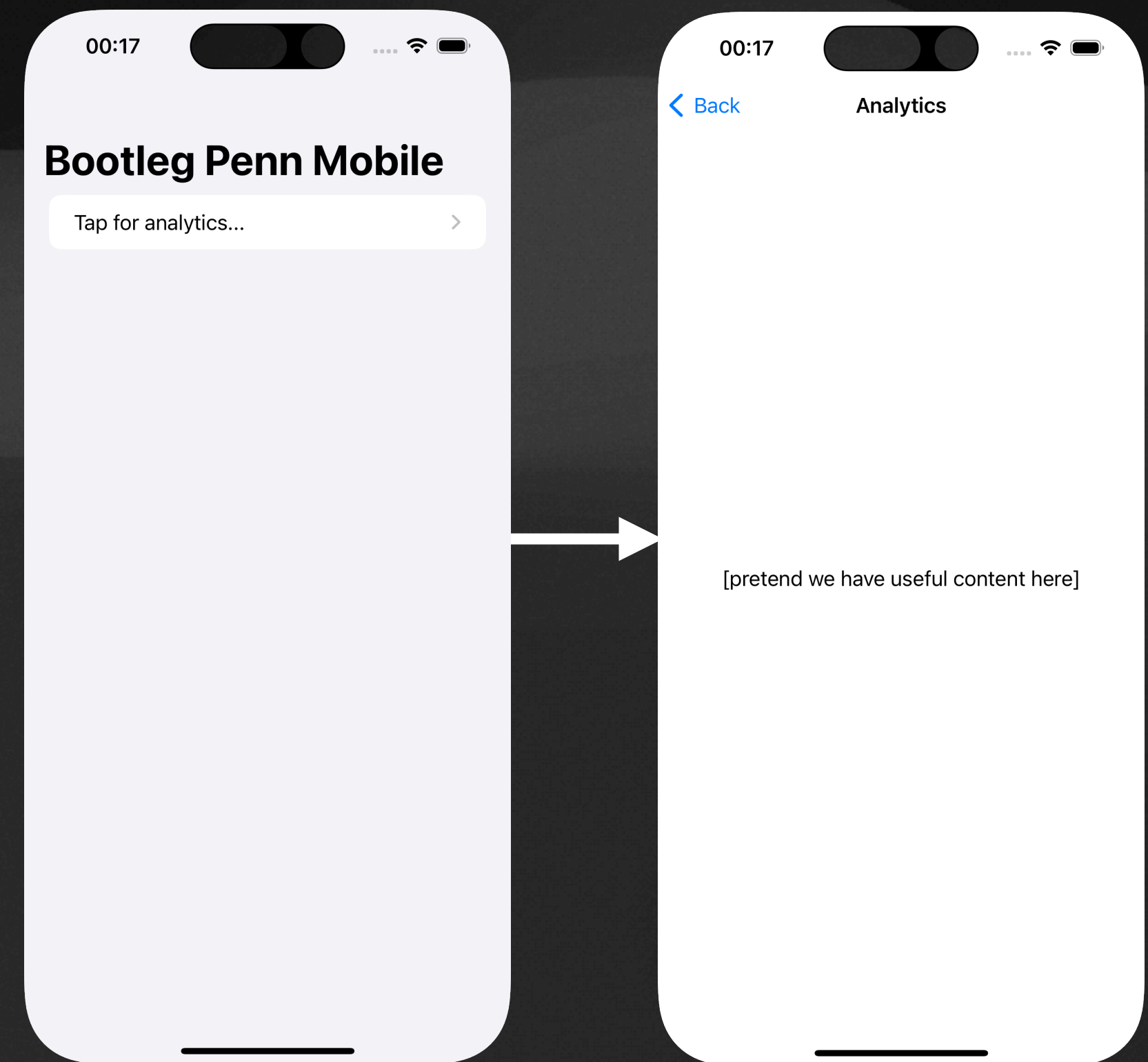
```
NavigationView {  
    List {  
        NavigationLink("Tap for analytics...") {  
            Text("[pretend we have useful content here]")  
                .navigationTitle("Analytics")  
                .navigationBarTitleDisplayMode(.inline)  
        }  
    }  
    .navigationTitle("Bootleg Penn Mobile")  
}
```



NavigationStack

Directly linking to views

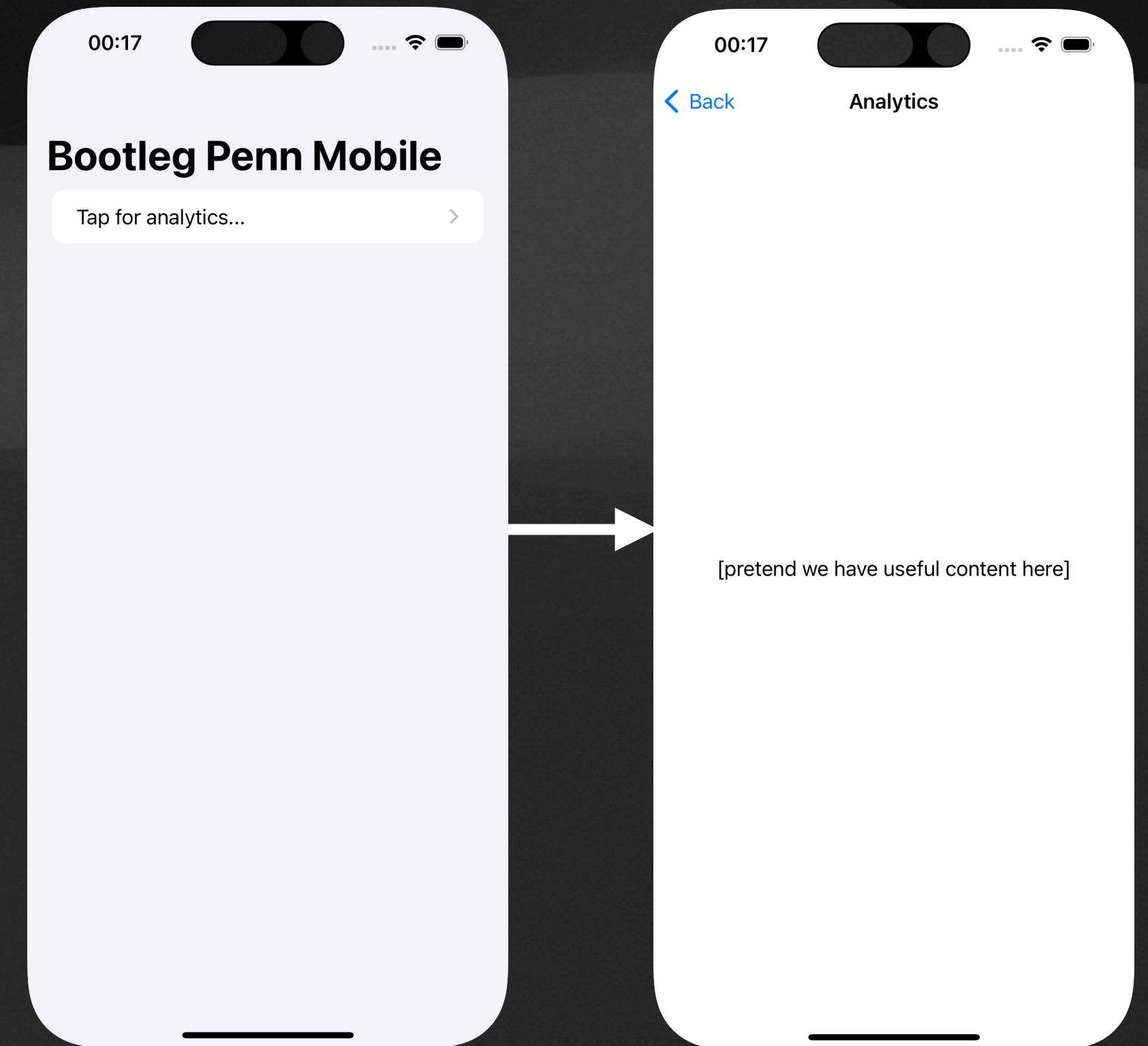
```
NavigationStack {  
  List {  
    NavigationLink("Tap for analytics...") {  
      Text("[pretend we have useful content here]")  
        .navigationTitle("Analytics")  
        .navigationBarTitleDisplayMode(.inline)  
    }  
  }  
  .navigationTitle("Bootleg Penn Mobile")  
}
```



NavigationStack

Presenting based on data


```
NavigationStack(path: $path) {  
  List {  
    NavigationLink("Tap for analytics...",  
      value: "Analytics")  
  }  
  .navigationTitle("Bootleg Penn Mobile")  
  .navigationDestination(for: String.self) { value in  
    Text("[pretend we have useful content here]")  
    .navigationTitle(value)  
    .navigationBarTitleDisplayMode(.inline)  
  }  
}
```



NavigationStack

Presenting based on data

```
NavigationStack(path: $path) {  
  List {  
    NavigationLink("Tap for analytics...", value: "Analytics")  
  }  
  .navigationTitle("Bootleg Penn Mobile")  
  .navigationDestination(for: String.self) { value in  
    Text("[pretend we have useful content here]")  
    .navigationTitle(value)  
    .navigationBarTitleDisplayMode(.inline)  
  }  
}
```



Allows you to modify path programmatically

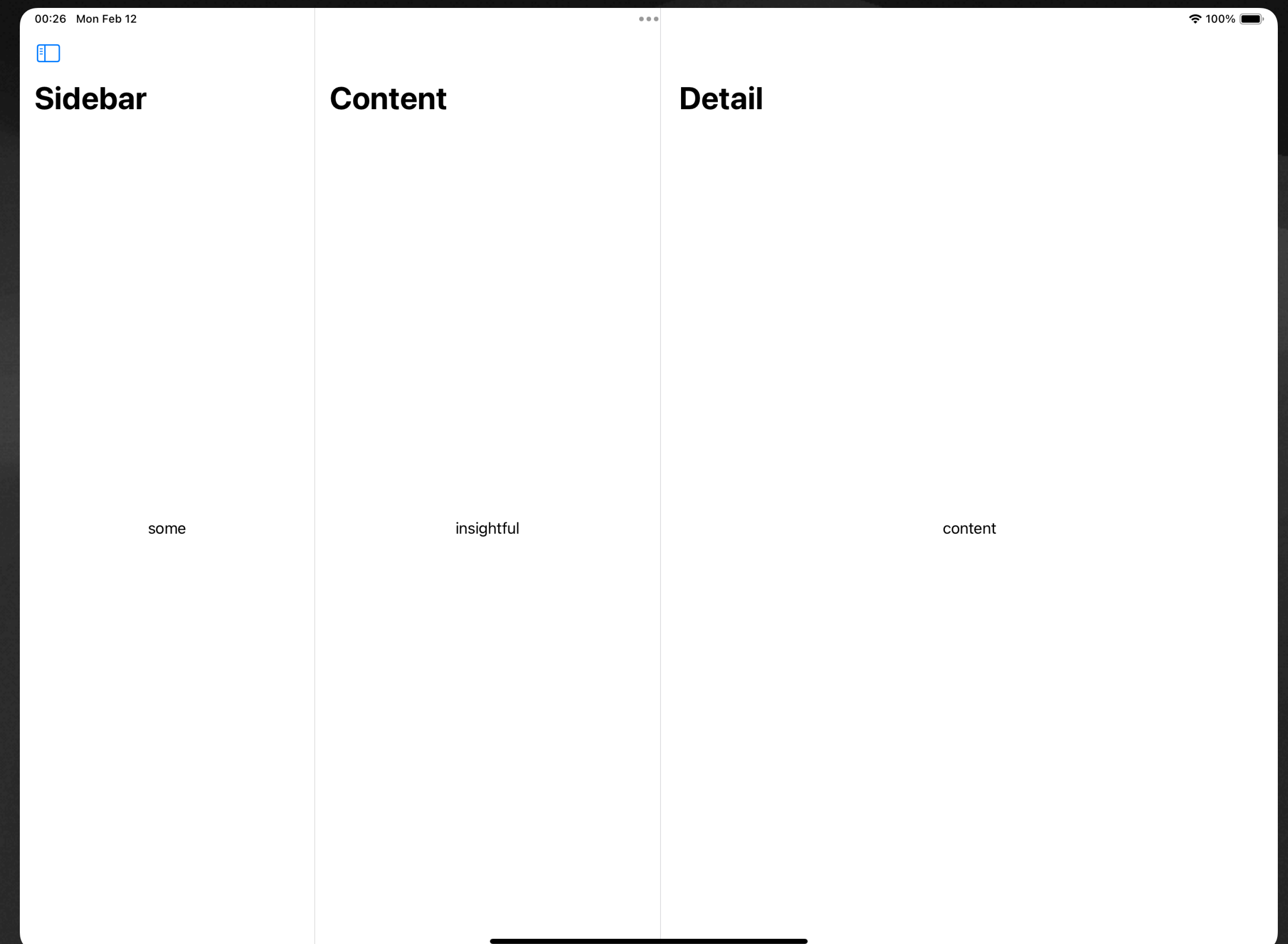
value is passed into **.navigationDestination**

Can help make code cleaner

NavigationSplitView

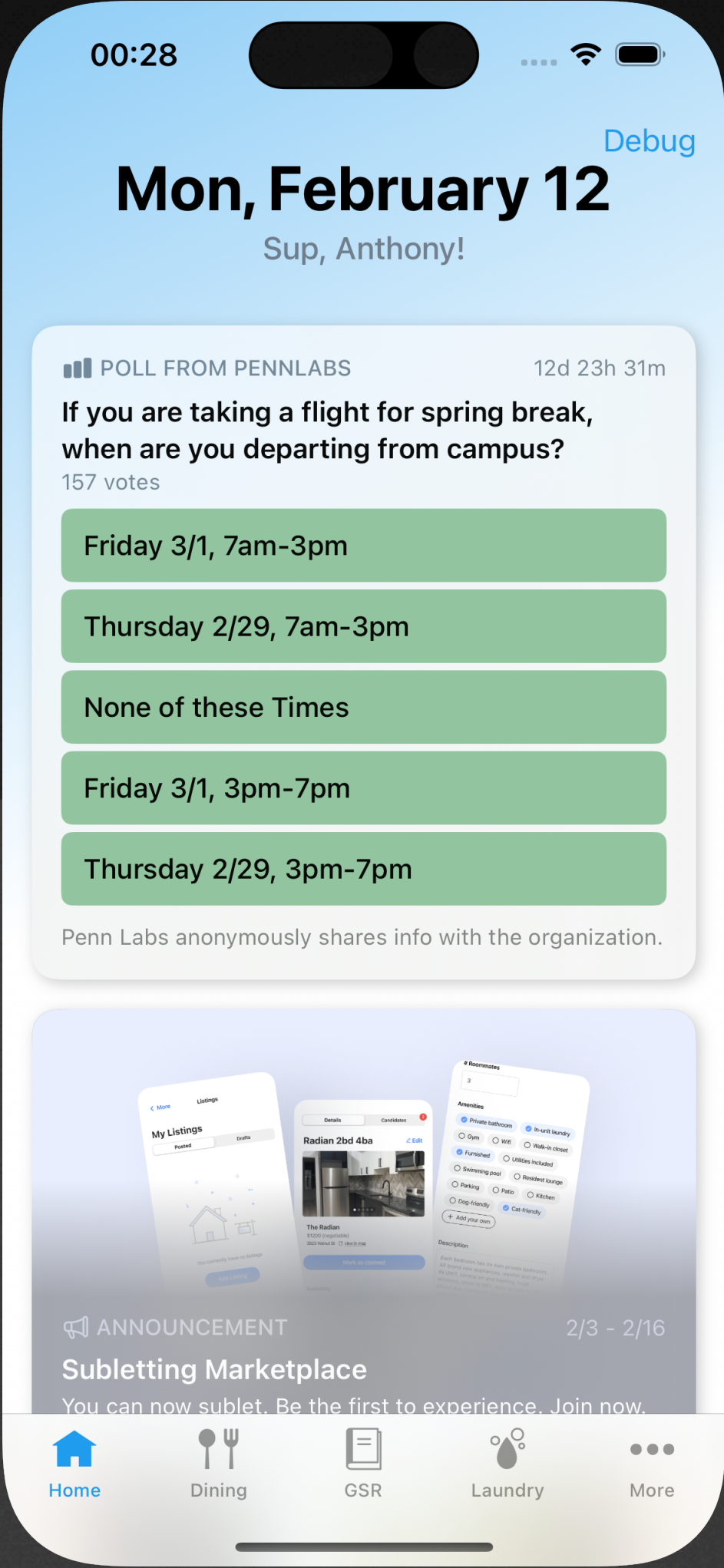
Multi-column layouts

```
NavigationSplitView(sidebar: {  
    Text("Sidebar")  
}, content: {  
    Text("Content")  
}, detail: {  
    Text("Detail")  
})  
.font(.largeTitle)
```



Appears as a NavigationStack on iPhone

TabView



Developer

NewsDiscoverDesignDevelopDistributeSupportAccount

Documentation / Navigation / TabView

Language: Swift API Changes: None

SwiftUI

Presenting views in tabs

TabView

Creating a tab view

init(content: () -> Content)

init(selection: Binding<SelectionValue>?, c...

func tabViewStyle<S>(S) -> some View

func tabItem<V>(() -> V) -> some View

Displaying views in multiple panes

HSplitView

VSplitView

Deprecated Types

NavigationView

Modal presentations

Toolbars

Search

App extensions

Data and storage

Model data

Environment values

Preferences

Structure

TabView

A view that switches between multiple child views using interactive user interface elements.

iOS 13.0+ iPadOS 13.0+ macOS 10.15+ Mac Catalyst 13.0+ tvOS 13.0+ watchOS 7.0+

visionOS 1.0+

struct TabView<SelectionValue, Content> where SelectionValue : Hashable, Conten

Overview

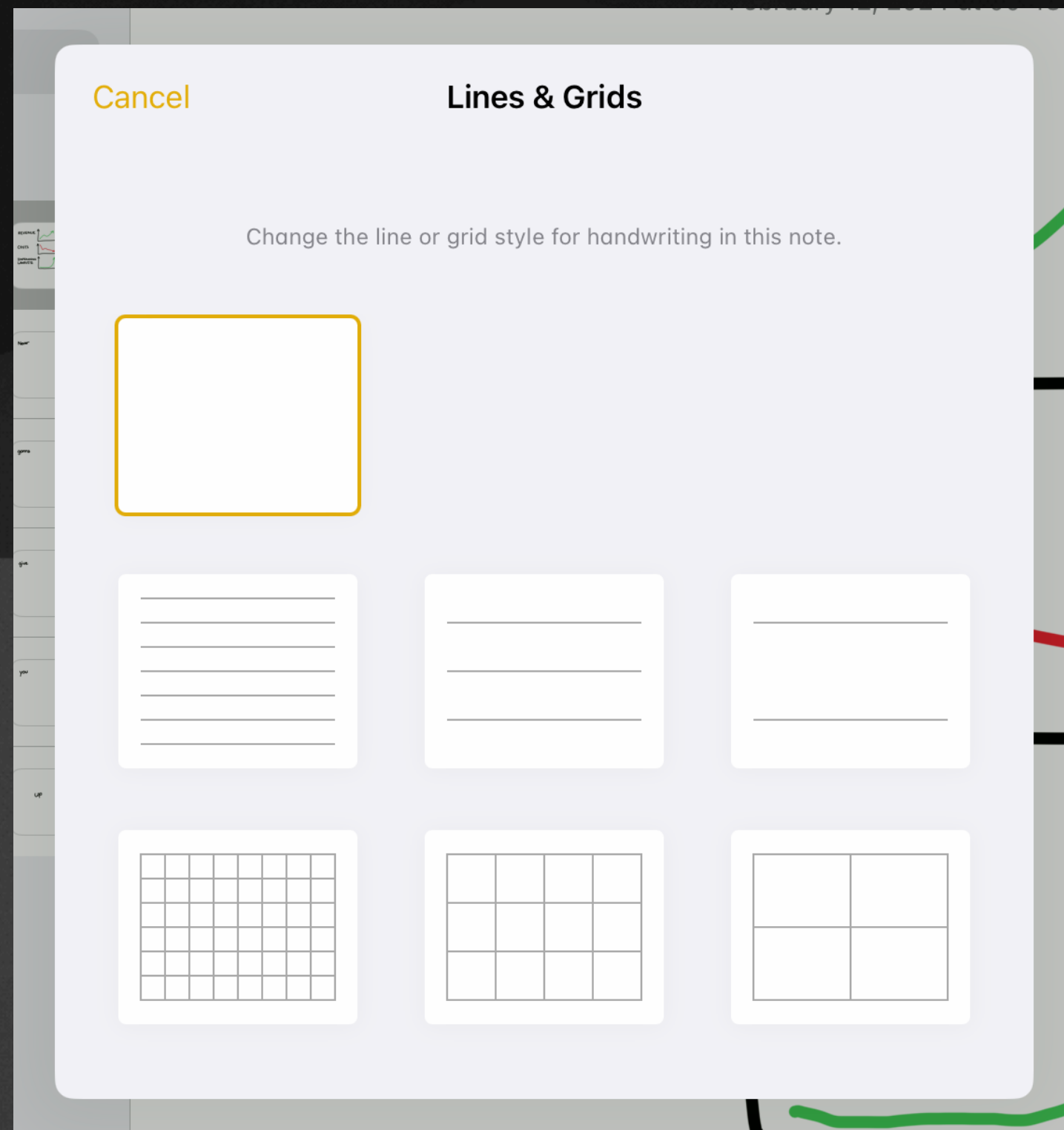
To create a user interface with tabs, place views in a TabView and apply the tabItem(_:) modifier to the contents of each tab. On iOS, you can also use one of the badge modifiers, like badge(_:), to assign a badge to each of the tabs.

The following example creates a tab view with three tabs, each presenting a custom child view. The first tab has a numeric badge and the third has a string badge.

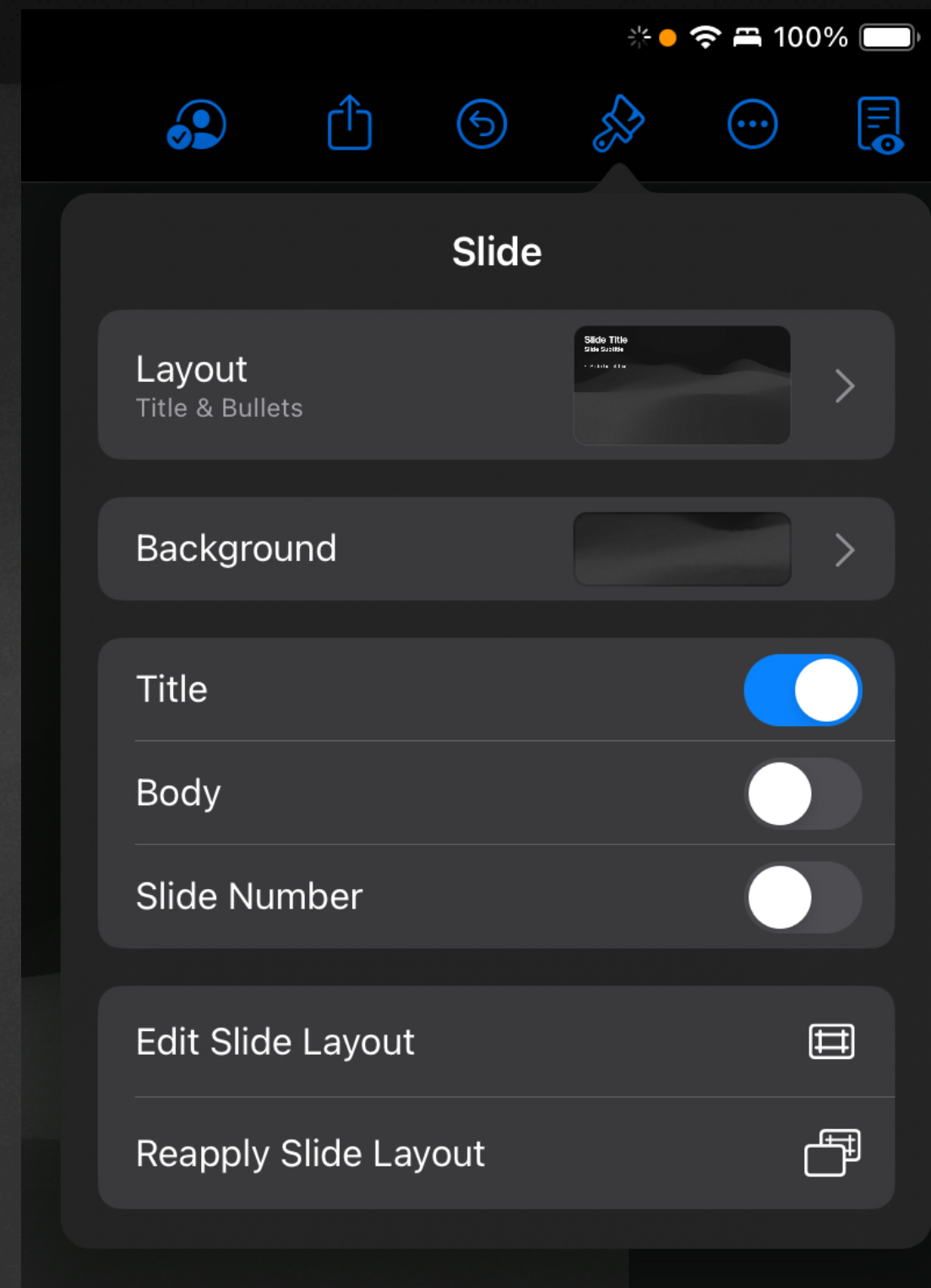
TabView {
 ReceivedView()
 .badge(2)

Modal presentations

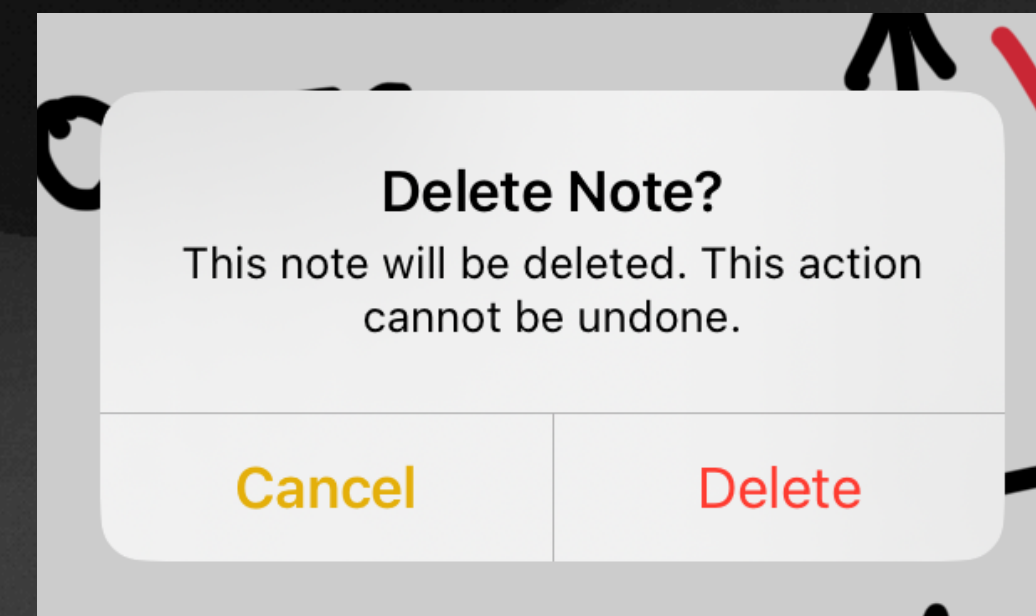
For lightweight, focused interactions



.sheet



.popover



.alert



Menu
— OR —
.contextMenu

developer.apple.com/videos/play/wwdc2022/10054/

Developer

Open in the Developer app

Open

Videos

CollectionsTopicsAll VideosAbout

The SwiftUI cookbook for navigation

The recipe for a great app begins with a clear and robust navigation structure. Join the SwiftUI team in our proverbial coding kitchen and learn how you can cook up a great experience for your app. We'll introduce you to SwiftUI's navigation stack and split view features, show you how you can link to specific areas of your app, and explore how you can quickly and easily restore navigational state.

Resources

⬇

Bringing robust navigation structure to your SwiftUI app

💬

Have a question? Ask with tag [wwdc2022-10054](#)

📄

List

📄

Migrating to new navigation types

📄

NavigationSplitView

📄

NavigationStack

💬

Search the forums for tag [wwdc2022-10054](#)

⬇

[HD Video](#) | [SD Video](#)

Related Videos

Tech Talks

▶

What's new for enterprise developers

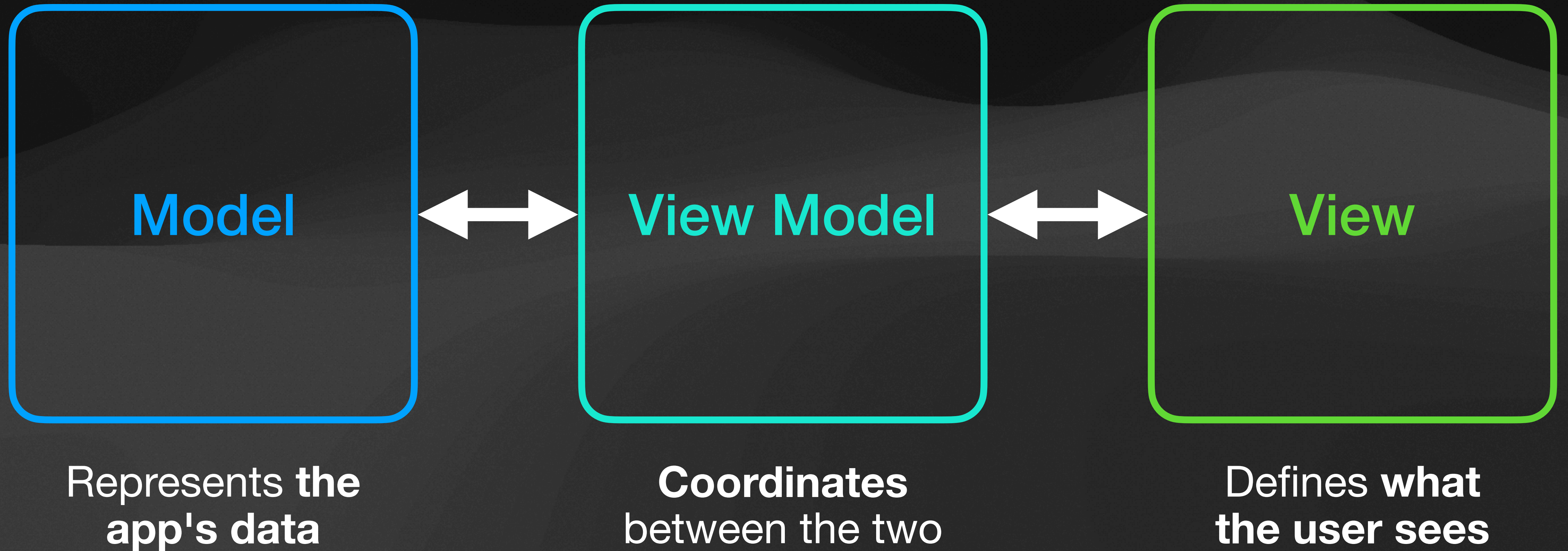
MVVM

Separation of Concerns

- Split code into **modular components**
- Each component only **handles one thing** (a "concern")
- Why? More testable, readable, maintainable code

MVVM

Model-View-View Model



MVVM

The View Model

View Model

Lets the view **bind to data** and **send commands**

Notifies the view of any changes

Converts data to and from what the view wants

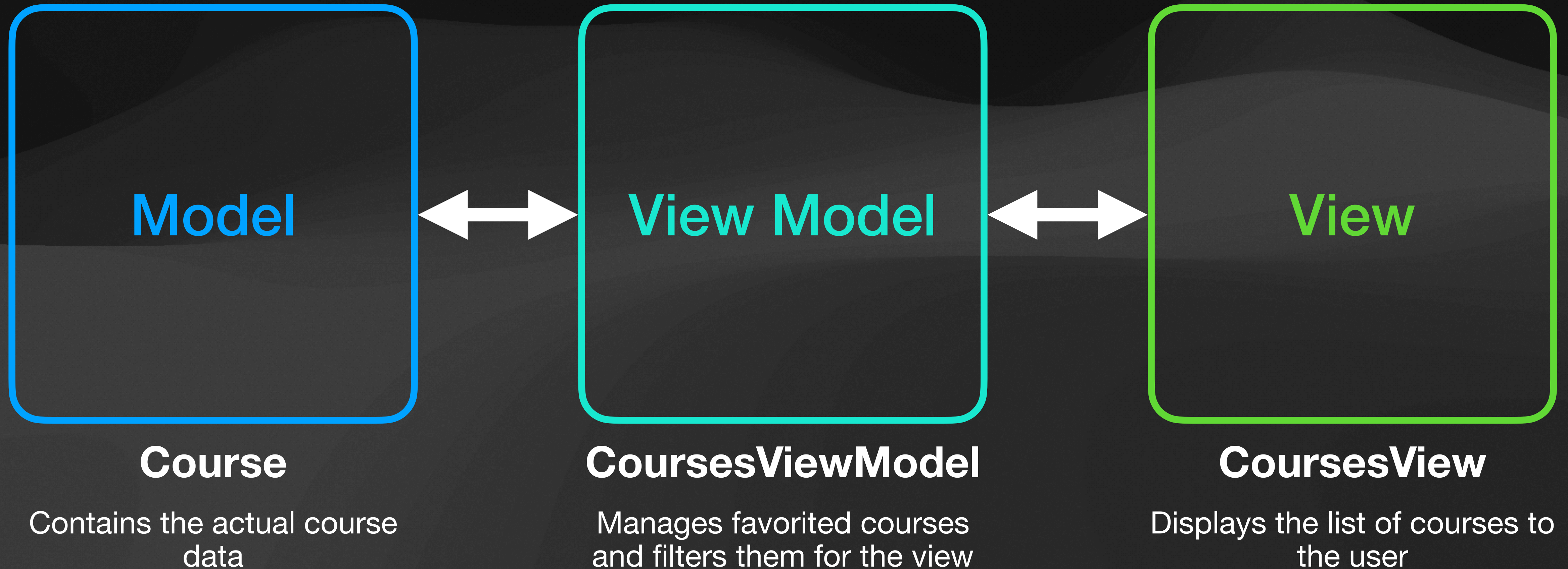
Isolates the view from its underlying data

Usually **a class**

Coordinates
between the two

MVVM

How we'll use it



Lifecycle Events

.onAppear and .onDisappear

```
VStack {  
    Image(systemName: "lightbulb.fill")  
        .imageScale(.large)  
        .foregroundColor(.yellow)  
    Text("Fun fact:")  
        .fontWeight(.bold)  
    Text("Eating is good for you!")  
}  
.onAppear {  
    print("Hello!")  
}  
.onDisappear {  
    print("Goodbye!")  
}
```

.onAppear

Hello!

[View is shown in app]



.onDisappear

Goodbye!

Why should I use lifecycle events?

Side effects

EXAMPLES

Loading or saving data

Making network requests

Triggering animations

Requesting access to sensor data

Cleaning up resources

And more!

Recap

- **Navigation and modal presentation views** let us organize multiple screens
- **Model-view-view model** enables separation of concerns
- **Lifecycle events** let us trigger side effects in response to views appearing and disappearing

Homework 2

Trivia Game

- Will be released **Thursday, 9/26**
- Due on **Thursday, 10/10**
 - Includes break — start early!
- Focuses on **lectures 3-5**
- [details pending]

