Introduction to iPadOS & Adaptive Layouts in SwiftUI

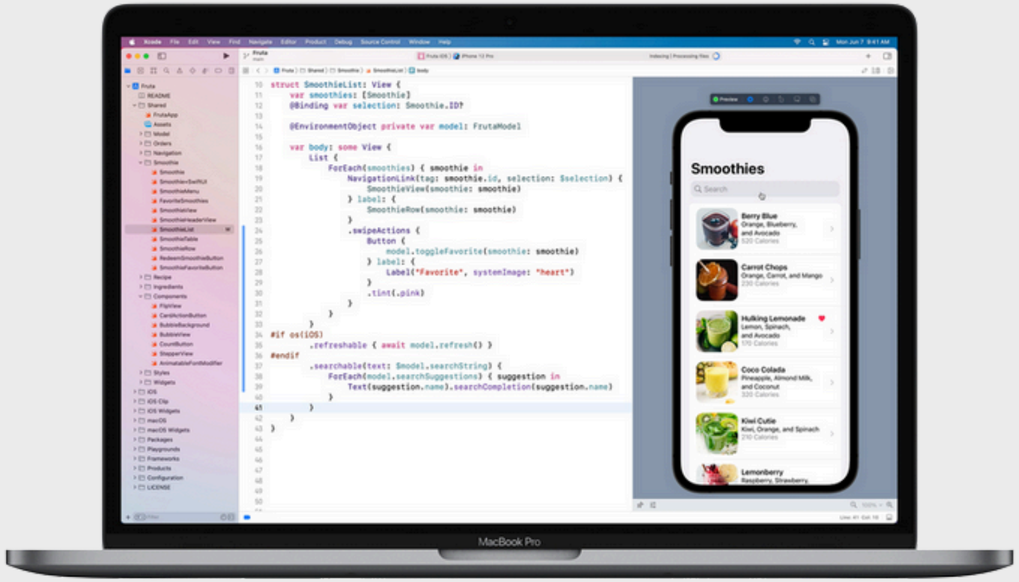
SDGKU
Professor Gabriela

Swift

Swift is a programming language created in 2019 for the creation of Apple Multiplatform content

The first-generation iPad was introduced on January 27, 2010.

The iPhone's iOS operating system (OS) was initially used for the iPad, but in September 2019, its OS was switched to a fork of iOS called iPadOS



What is iPadOS?

- iPadOS is a variant of iOS tailored for the iPad, offering:
  - Larger screen real estate → Requires different layout strategies than iPhone
  - Multitasking & multi-window support
  - Apple Pencil support
  - Keyboard + trackpad + pointer input
  - Split-screen & Slide Over modes

Key Differences: iPadOS vs iOS

iPadOS

Multitasking Made Easy

Use two apps side by side with Split View or open a second app over another using Slide Over—perfect for getting more done on the iPad's larger screen.

iPad Pencil

Only iPads support the Apple Pencil, which allows for handwriting, drawing, and even smart gestures like circling text to highlight it.

Apps Designed Just for iPad

iOS

Works with Apple Watch

Only iPhones can be paired directly with Apple Watches.

Designed for Smaller Screens

Apps Designed Just for iPhone

App Layouts

Organize the information showed inside your View file by using the parameters.

HStack {}

VStack {}

ZStack {}

This will arrange all the properties according to the displayed type used.

App Layouts

Organize the information showed inside your View file by using the parameters.

HStack {}

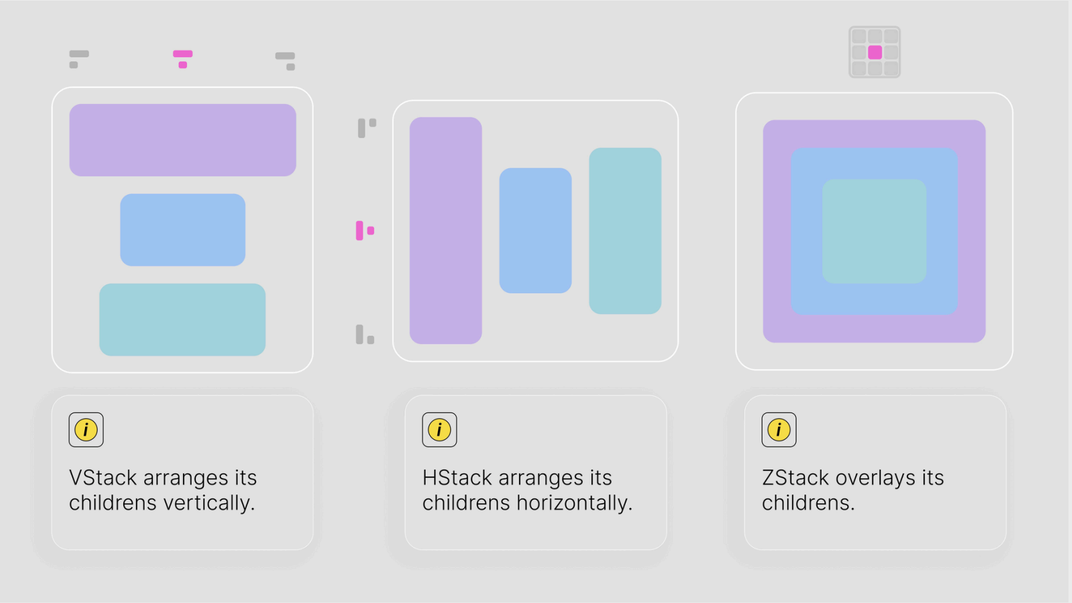
VStack {}

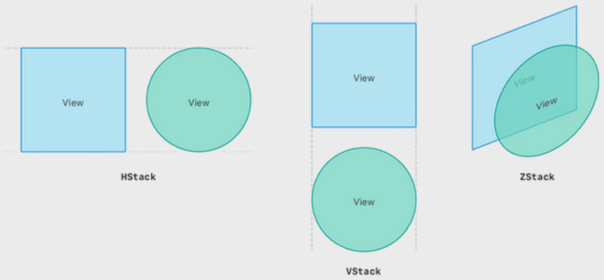
ZStack {}

This will arrange all the properties according to the displayed type used.

```
var body: some View {
    VStack(
        alignment: .leading,
        spacing: 10
    ) {
        ForEach(
            1...10,
            id: \.self
        ) {
            Text("Item \($0)")
        }
    }
}
```

Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9
Item 10





Geometry Reader

GeometryReader is a container view in SwiftUI that measures the size and position of the views it contains relative to its parent. It passes this information down to its child views, allowing you to dynamically adjust your layout based on the available space.

GeometryReader allows you to access the following properties of the view:

- **Size (width and height)**
- **Position (frame origin)**

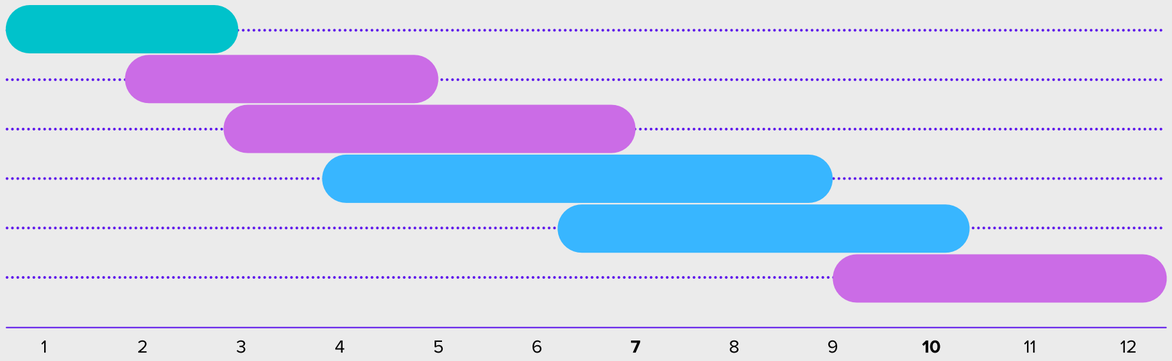
Limitations of GeometryReader

Infinite Size: GeometryReader asks its parent for as much space as possible. If used in a context where the size is undefined (e.g., inside a ScrollView), it might take up all available space.

Performance: The overuse of GeometryReader in complex layouts can lead to performance bottlenecks, as it requires calculating and passing layout information down the hierarchy.

```swift
struct ResponsiveCircleView: View {
    var body: some View {
        GeometryReader { geometry in
            Circle()
                .fill(Color.blue)
                .frame(width: geometry.size.width * 0.5, height: geometry.size.width * 0.5)
                .position(x: geometry.size.width / 2, y: geometry.size.height / 2)
        }
        .background(Color.gray.opacity(0.2))
    }
}
```

Principles of User Design



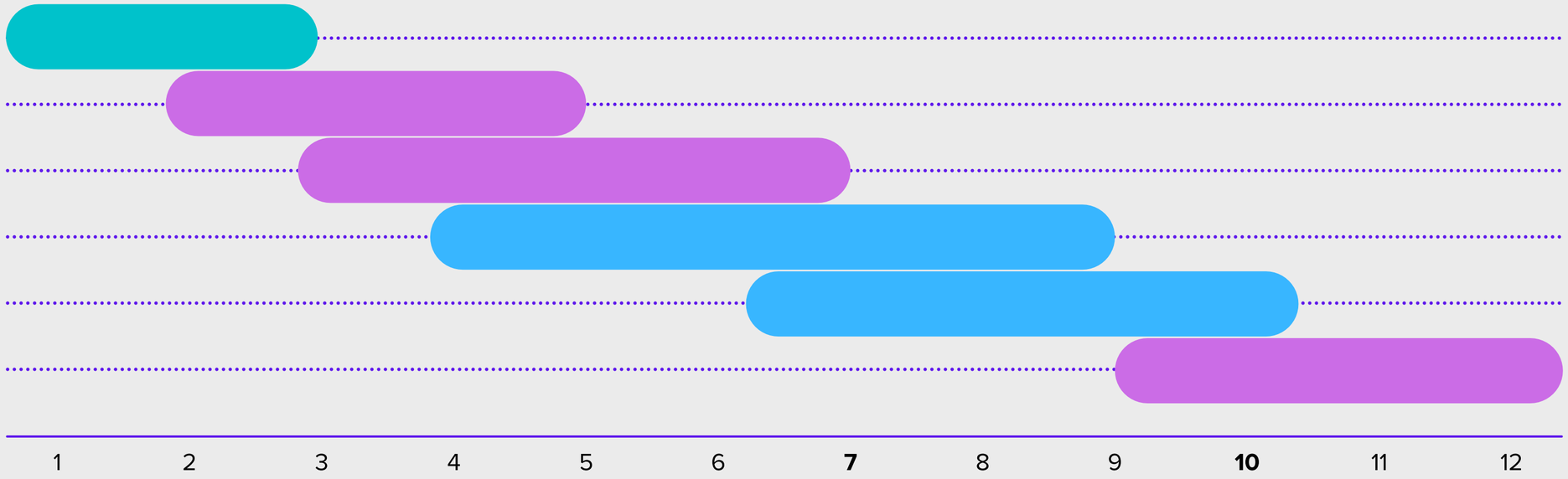
What is Visual Hierarchy?

The principle of arranging elements to show their order of importance. It’s the visual language you use to tell users **what to look at first,** second, and third.

Why need this? Without a clear hierarchy, users are faced with a wall of information, leading to confusion and frustration. A good hierarchy makes an interface feel organized and effortless to use.

The principle of arranging elements to show their order of importance. It’s the visual language you use to tell users **what to look at first,** second, and third.

Why need this? Without a clear hierarchy, users are faced with a wall of information, leading to confusion and frustration. A good hierarchy makes an interface feel organized and effortless to use.



Bad Example

What is Visual Hierarchy?

Key Tools for Building Hierarchy

Size & Scale: Larger elements command more attention.

iPadOS Example: In the Photos app, the title of an album or a "Year" is much larger than the individual photo thumbnails.

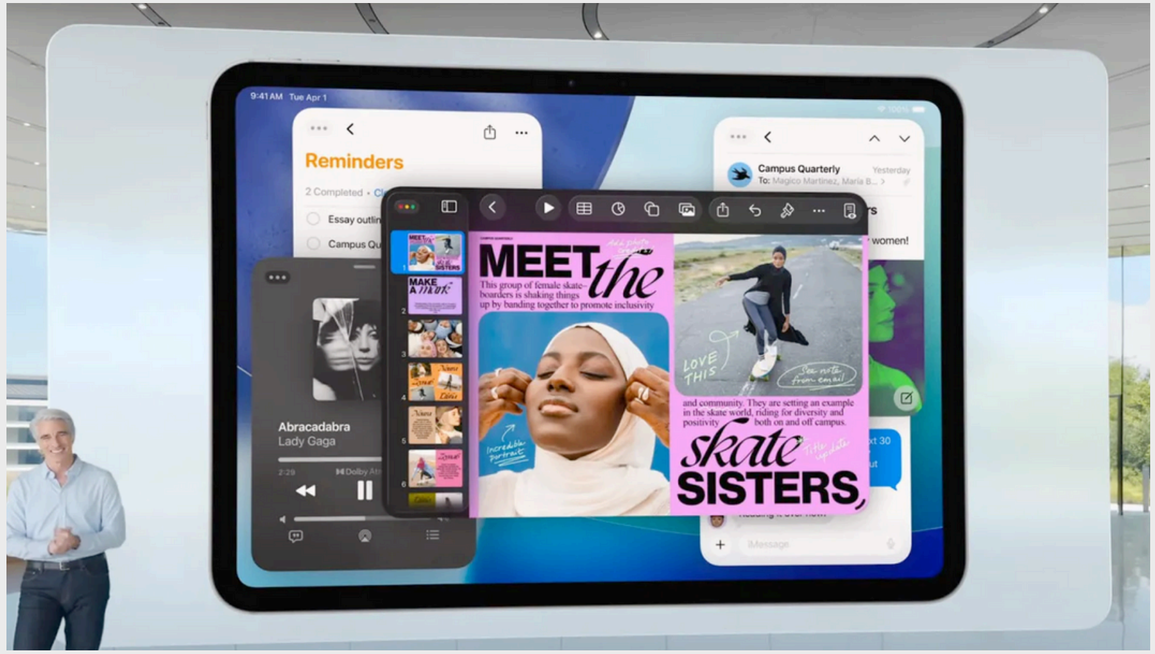
Color & Contrast: Bright colors and high contrast make elements pop.

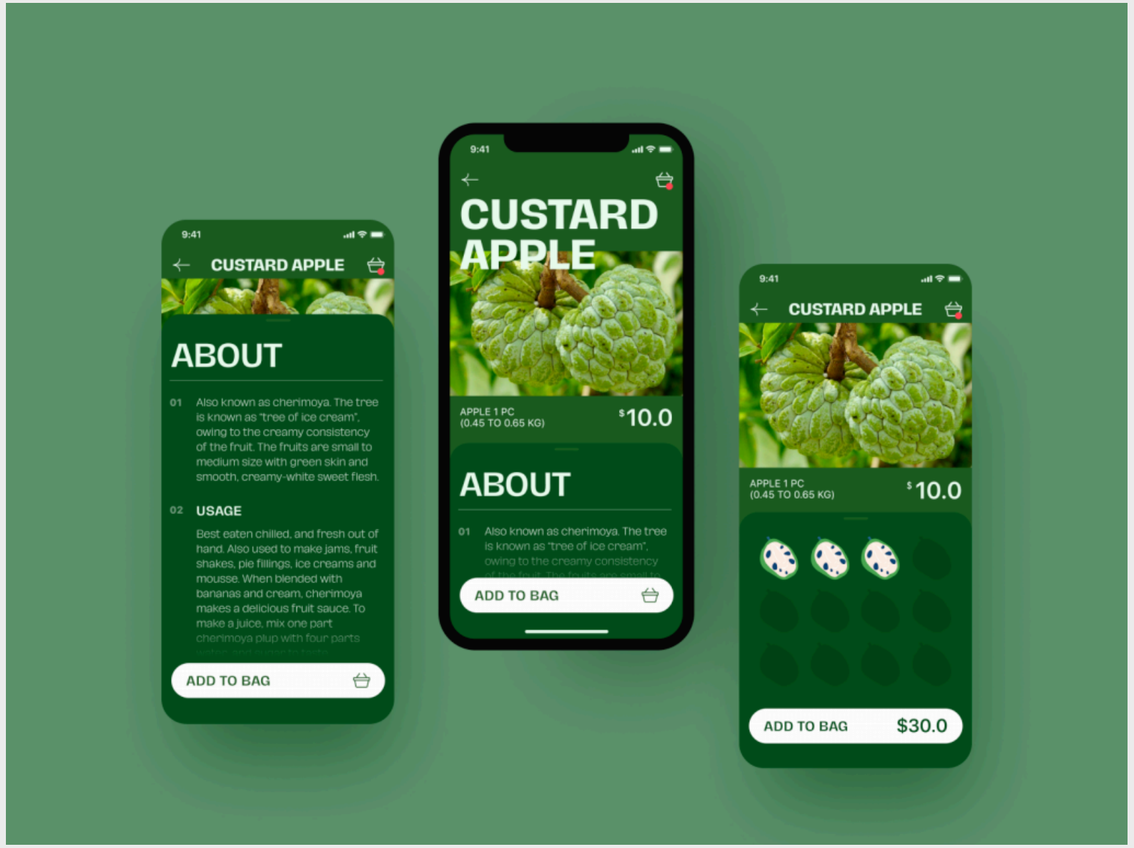
Example: The primary action button, like the "+" to add a new note in the Notes app, often uses a bold, high-contrast color.

Layout & Spacing (Whitespace): How you group elements and the space you leave around them creates a sense of order and separation.

Example: The Settings app uses spacing to group related settings into logical blocks, making the long list scannable.

1





What is Visual Consistency?

ften called the "Principle of Least Surprise," consistency means that similar elements look and behave in similar ways. It’s about creating predictable patterns that users can learn once and apply everywhere.

Why need this? It drastically reduces the user's learning curve. When they don't have to wonder what an icon does or where a menu is, they can focus on their actual tasks.

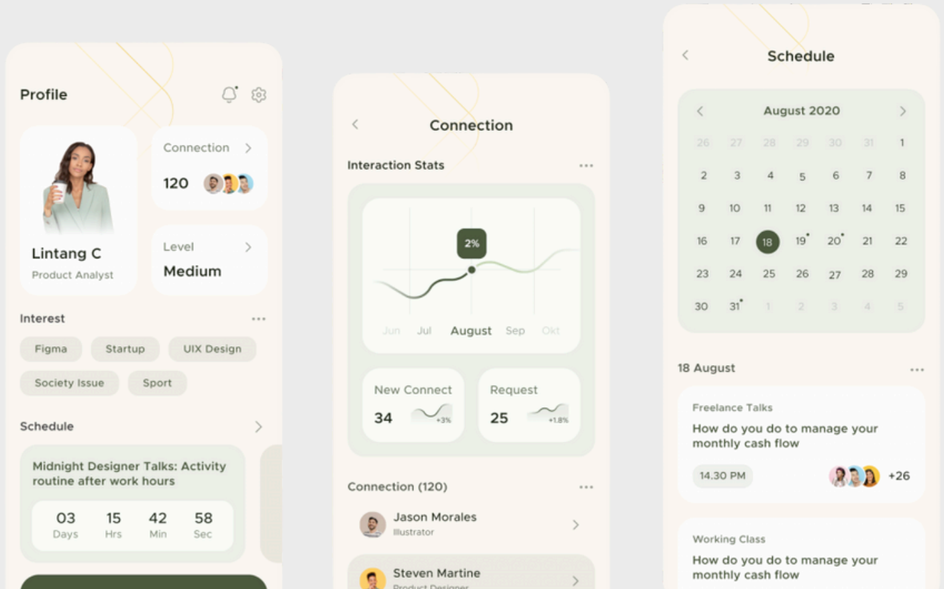
iPadOS Context: Apple's Human Interface Guidelines (HIG) are the bedrock of consistency on the platform. Following them ensures your app feels at home on an iPad, leveraging patterns users already know from other apps.

The Two Levels of Consistency

Internal Consistency (Within Your App):

The elements inside your app are consistent with each other.

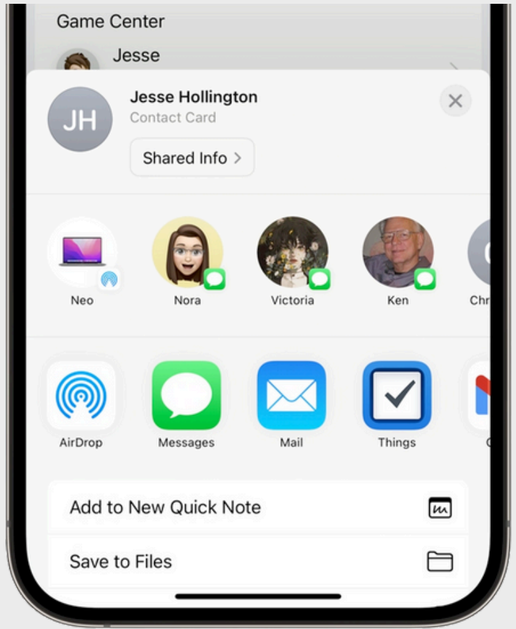
Example: In your app, if the "Delete" action is represented by a red trash can icon 🗑️ on one screen, it should be a red trash can icon everywhere. The primary action buttons should always be the same color and in a similar location.



External Consistency (With iPadOS):

Your app is consistent with the rest of the operating system.

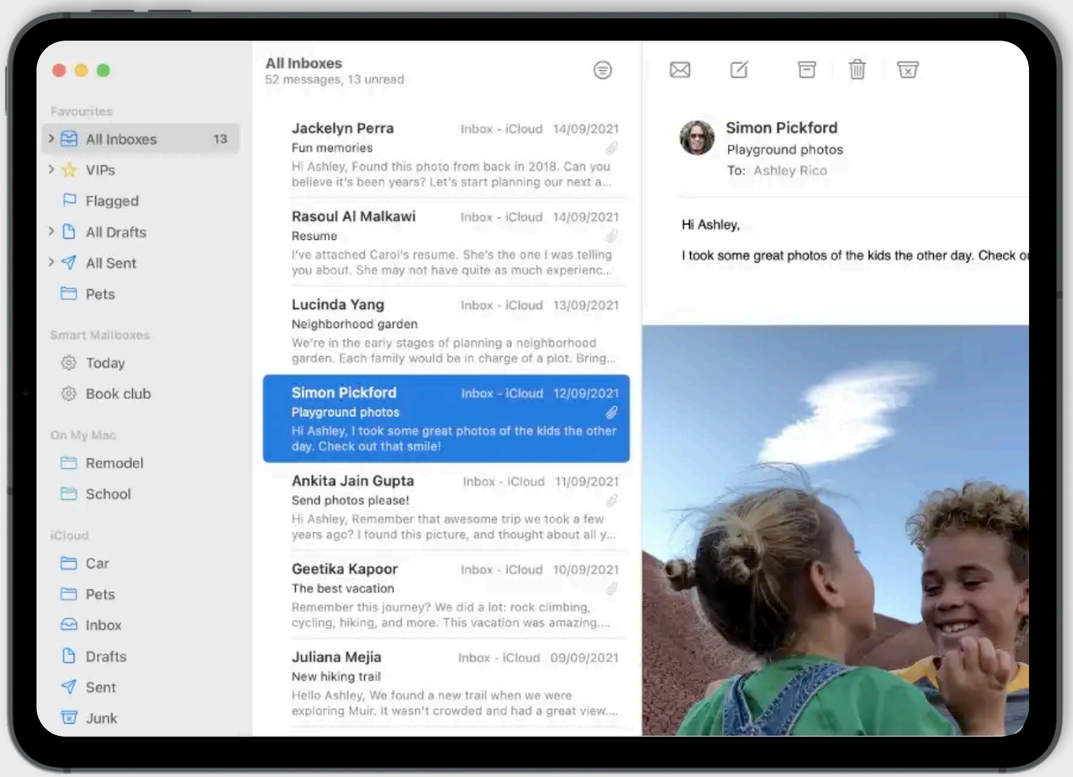
Example: Using the standard iPadOS Share Sheet. When a user wants to share something, they look for the familiar share icon, and they expect the same system menu to appear, regardless of the app. This is a powerful way to make your app feel instantly familiar.



The Core Navigation Pattern

The primary navigation model for iPadOS apps with complex information. It creates up to three columns:

We already understand the why,now lets see the HOW



Sidebar:

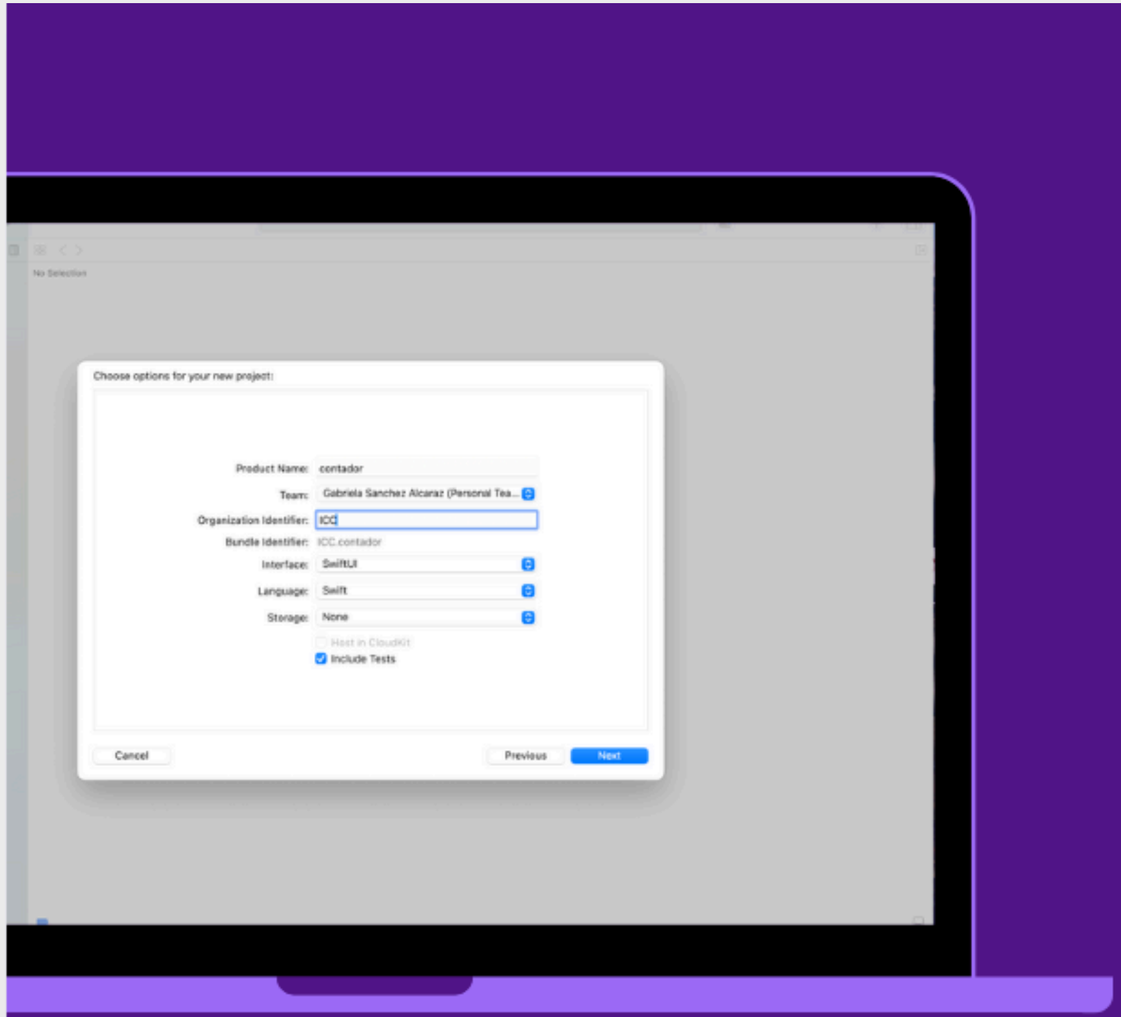
The top-level containers of your app (e.g., Mailboxes in Mail, Folders in Notes).

Content List:

The items within the selected sidebar category (e.g., the list of emails).

Detail View:

The content of a selected item (e.g., the opened email).



EXERCISE

TODO APP

Wardiere Inc.

Thank You

Chiaki Sato
Founder & CEO

www.reallygreatsite.com
hello@reallygreatsite.com