

Pinterest 2.0

Practical Exercise: SwiftUI, Design Systems, and HIG Adherence

This is a native iOS application developed entirely in SwiftUI. The primary goal of this exercise was the **in-depth understanding and practical application** of Apple's Human Interface Guidelines (HIG) and Design Systems principles to build a high-quality, intuitive user interface.

Primary Goal: Design as the Core Foundation

This project served as a **focused exercise to learn the mechanics of app design**. Every code and structural decision was driven by the necessity of achieving a superior, native-feeling user experience.

- **Design is Not an Option:** This exercise demonstrates how attention to detail and HIG adherence are a **fundamental and non-negotiable part** of app development.
- **Architecture for Clarity:** Development of modular Views that respect the principles of visual hierarchy and standard iOS navigation.

Design System & UX Highlights

The project showcases the application of specific design decisions and best practices:

1. Root Navigation Structure

- **HIG Compliance:** Exclusive use of native TabView and NavigationStack ensures the navigation system is immediately familiar and robust for the iOS user.

2. Layout & Visual Composition (The Grid System)

The grid layout was specifically calibrated for visual impact and uniformity:

- **Fixed and Uniform Grid:** Implementation of a LazyVGrid with **2 fixed columns** and images constrained to a **uniform vertical aspect ratio** (240pt height, 140pt minimum width), eliminating visual heterogeneity.
- **Image Handling:** Combined use of .scaledToFit(), .clipped(), and .cornerRadius(12) professionalizes the look of the cells, ensuring aesthetic consistency across all devices.

3. Structure and User Feedback

- **Explicit Titles:** Strategic placement of .navigationTitle and .navigationBarTitleDisplayMode(.large) provides immediate context to the user on every screen.
- **Native Search:** Integration of the native .searchable() modifier ensures a clean, compliant UX, avoiding confusing custom search elements.
- **Empty State:** Use of ContentUnavailableView adheres to HIG for providing informative feedback when no content is present.



Technical Architecture (SwiftUI Best Practices)

The project is built on a clear data architecture typical of modern SwiftUI projects:

File	Architectural Role	Implementation Details
ContentView.swift	Root View & Flow Control	Main container (TabView) responsible for injecting the Data Model into the environment.
Models.swift	Data Models & Logic	Defines the Photo struct and the <i>Source of Truth</i> (SavedPhotosModel) as an ObservableObject.
GalleryView.swift	Primary Content	Implements the filtering logic using the search state binding.
SavedGalleryView.swift	Derived View (Gallery)	Displays saved items and handles the empty state according to HIG.



Requirements and Setup

- **Environment:** Developed using **Xcode 26.0.1**.
- **Target:** **iOS 26+**
- **Language:** **Swift 5+**

Project Startup

1. Open the project in Xcode.
2. Ensure required graphic assets (e.g., `pinterest_logo`, `foto1-5`, etc.) are present and correctly named in `Assets.xcassets`.
3. Run the application on an iOS 26+ simulator (`⌘ + R`).

Next Steps: Implementing Unit Tests (XCTest) for the `SavedPhotosModel` logic and refining the dynamic color system (`.tint()` and Asset Colors).