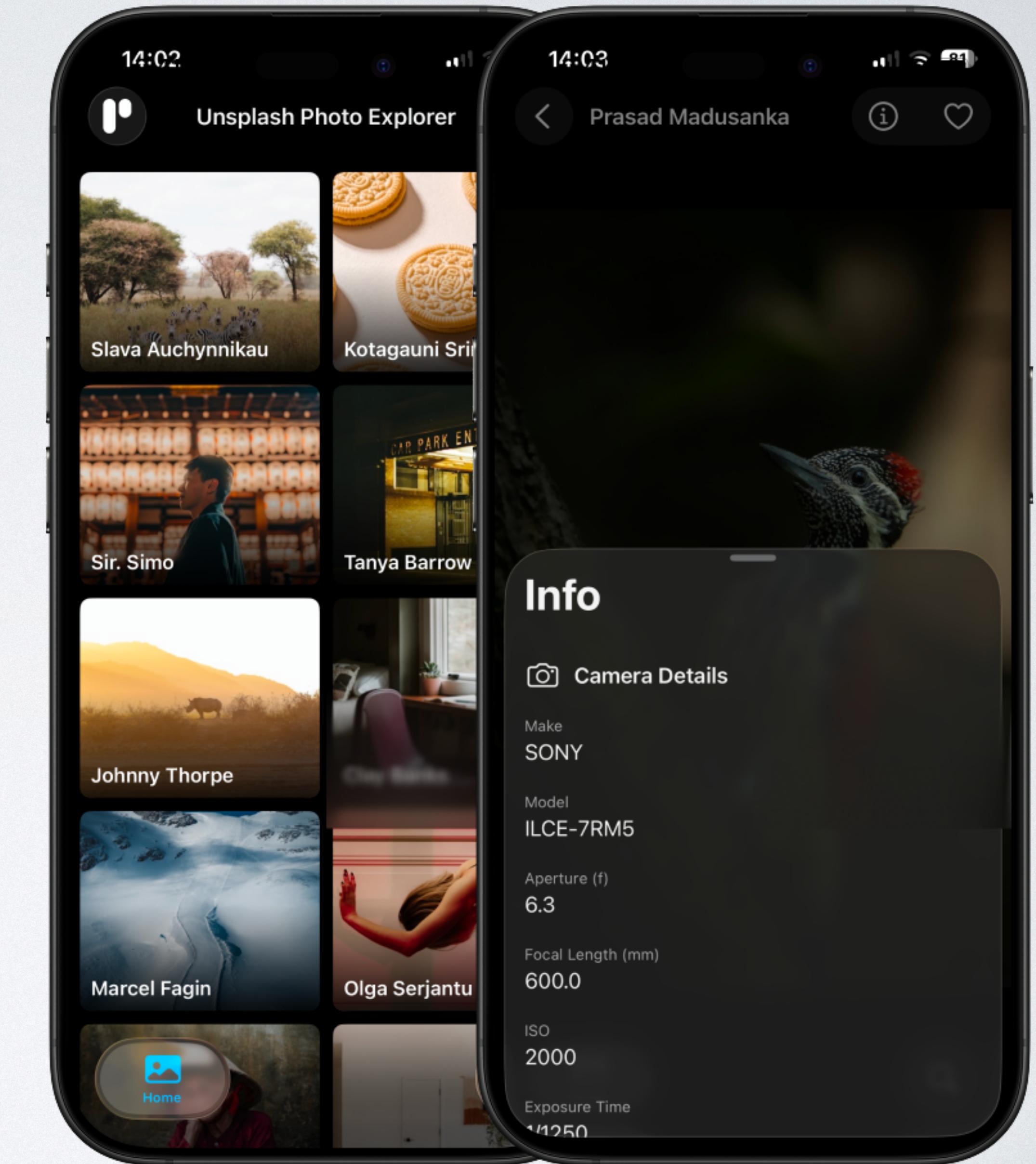
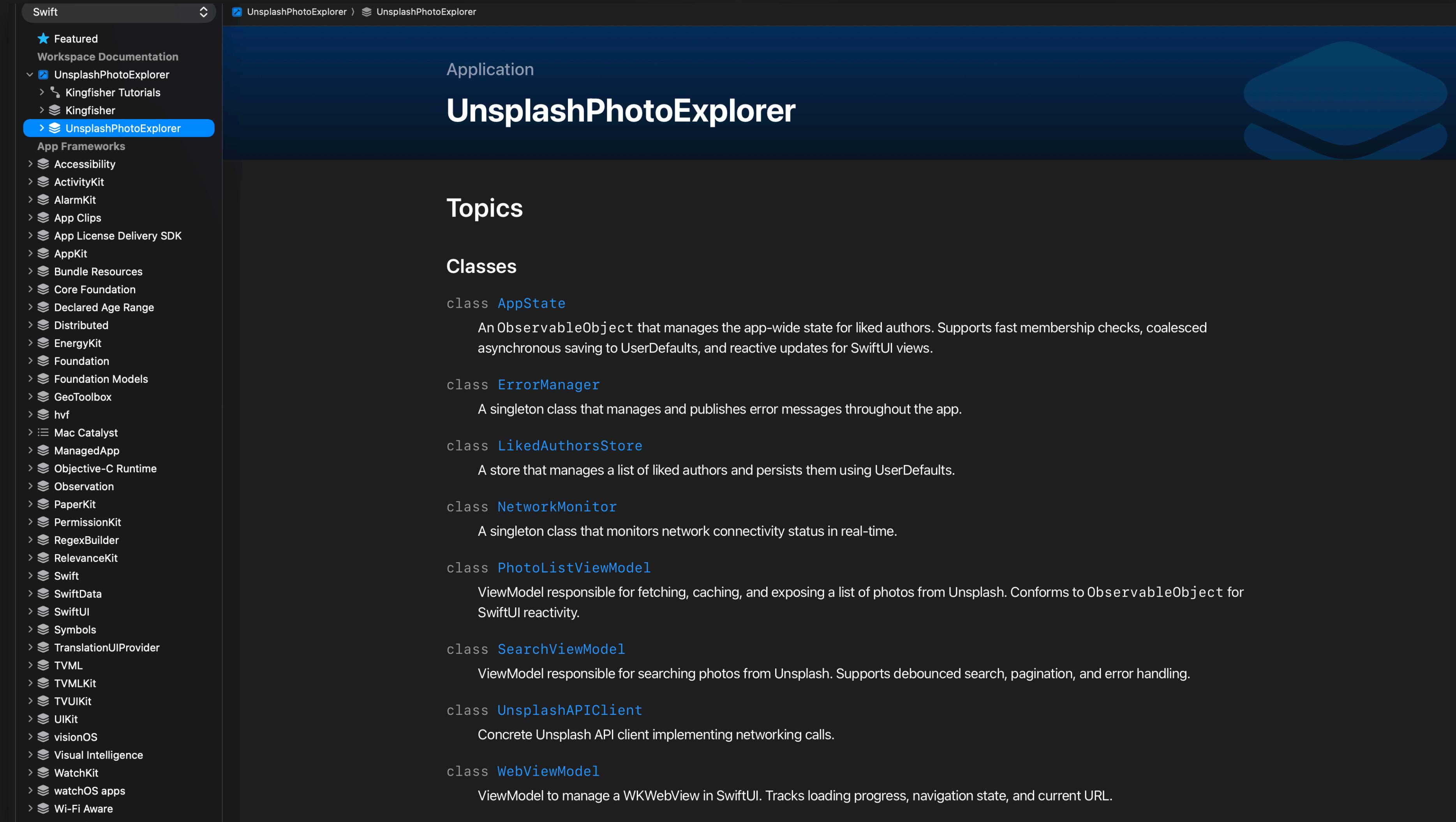


# UNSPLASH PHOTO EXPLORER APP

DEVELOPMENT PROCESS



# DOCUMENTATION



The screenshot shows the Xcode Documentation Browser interface. The left sidebar lists Swift frameworks and libraries, with "UnsplashPhotoExplorer" selected. The main content area displays the "UnsplashPhotoExplorer" application documentation, featuring a large blue header image and sections for Topics, Classes, and other documentation types.

## Topics

## Classes

```
class AppState
    An ObservableObject that manages the app-wide state for liked authors. Supports fast membership checks, coalesced asynchronous saving to UserDefaults, and reactive updates for SwiftUI views.

class ErrorManager
    A singleton class that manages and publishes error messages throughout the app.

class LikedAuthorsStore
    A store that manages a list of liked authors and persists them using UserDefaults.

class NetworkMonitor
    A singleton class that monitors network connectivity status in real-time.

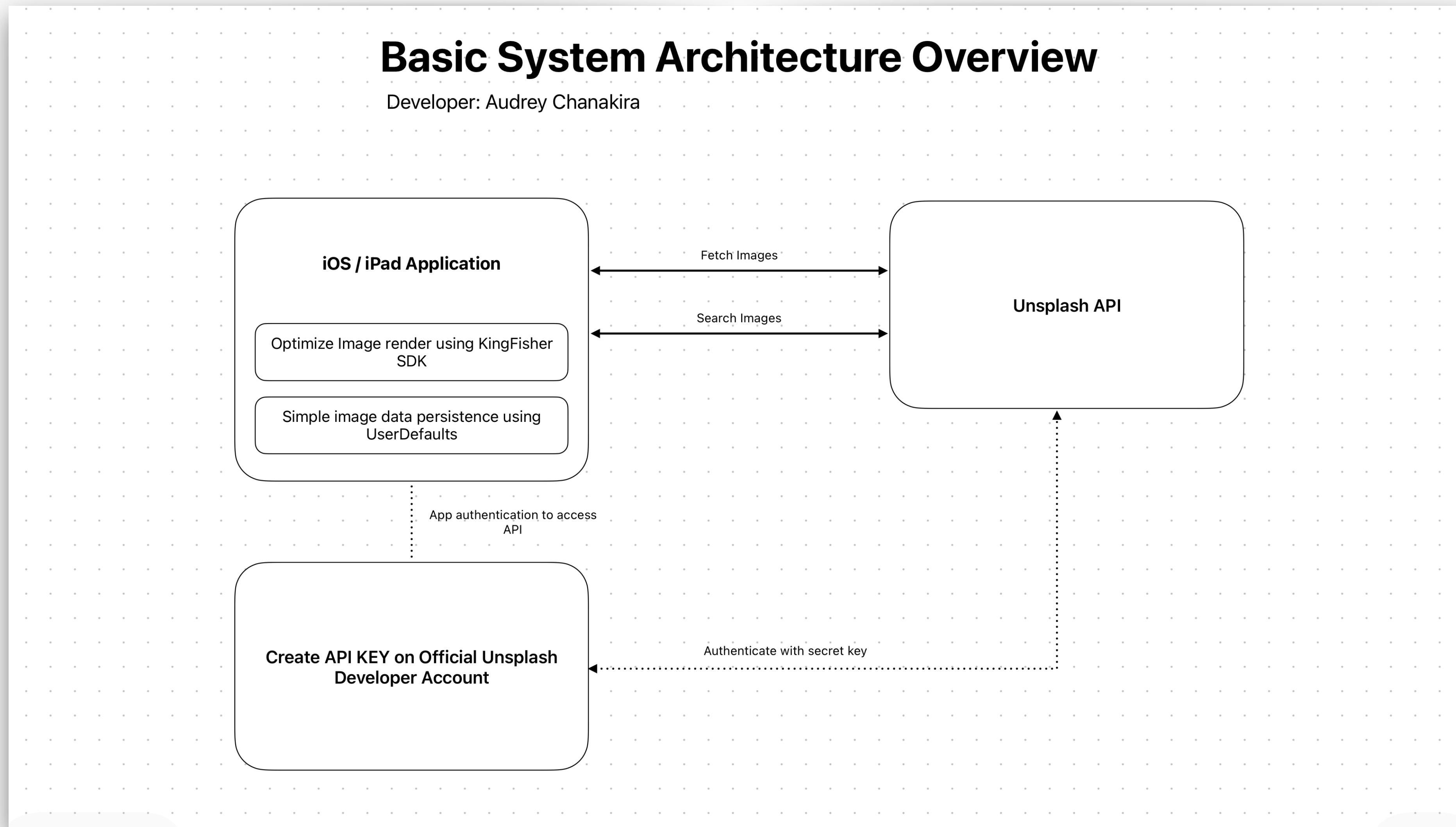
class PhotoListViewModel
    ViewModel responsible for fetching, caching, and exposing a list of photos from Unsplash. Conforms to ObservableObject for SwiftUI reactivity.

class SearchViewModel
    ViewModel responsible for searching photos from Unsplash. Supports debounced search, pagination, and error handling.

class UnsplashAPIClient
    Concrete Unsplash API client implementing networking calls.

class WebViewModel
    ViewModel to manage a WKWebView in SwiftUI. Tracks loading progress, navigation state, and current URL.
```

# BASIC SYSTEM ARCHITECTURE



# FIGMA UI/UX DESIGN

The screenshot displays the Figma application interface in dark mode, showcasing the design of an "Unsplash Photo Explorer" mobile application. The central area features a grid of eight smartphone screens illustrating various app screens:

- Home Screen:** Shows a grid of thumbnail images from Unsplash.
- Photo Detail Screen:** Displays a large, detailed image of a bird.
- Photo Info Sheet:** Provides technical details about the photo, including camera settings like Aperture (f/6.3), Focal Length (mm) 600.0, ISO 2000, and Exposure Time.
- Favourite Author List:** Lists favorite authors with their profile pictures and names.
- Author Web View:** A web view showing a user's profile and photos.
- Search Screen:** A search interface with a search bar and results.
- Developer Info:** Information about the developer, Audrey Chanakira.
- Error Alerts:** Two modal dialogs for "No Internet Connection" and "Error" with "Okay" buttons.

The left sidebar contains the project navigation, file assets, and a layers panel listing components like "Logo White", "Logo Black", and "Error Alert". The right sidebar includes sharing options, color palettes, and export functions.

# POSTMAN API TESTING

The screenshot shows the Postman application interface. On the left, the History sidebar lists several recent requests, including multiple GET requests to the Unsplash API. The main workspace displays a single active request for "https://api.unsplash.com/photos/random?count=10". The request method is "GET", and the URL is "https://api.unsplash.com/photos/random?count=10". The "Headers" tab is selected, showing a table with one row and two columns: "Key" and "Value". The "Body" tab is also visible, showing the JSON response from the API. The response body is a large object containing details about a photo, such as its ID, slug, alternative slugs in various languages, dimensions, and a detailed description. The status bar at the bottom indicates a successful 200 OK response with a time of 331 ms and a size of 10.38 KB.

```
id": "xw6cVv6wW5o",
"slug": "monk-holding-prayer-beads-behind-back-xw6cVv6wW5o",
"alternative_slugs": {
    "en": "monk-holding-prayer-beads-behind-back-xw6cVv6wW5o",
    "es": "monje-sosteniendo-cuentas-de-oracion-detras-de-la-espalda-xw6cVv6wW5o",
    "ja": "念珠を後ろに持つ僧侶-xw6cVv6wW5o",
    "fr": "moine-tenant-un-chapelet-de-priere-derriere-le-dos-xw6cVv6wW5o",
    "it": "monaco-che-tiene-i-rosari-dietro-la-schiena-xw6cVv6wW5o",
    "ko": "등-뒤로-염주를-들고-있는-승려-xw6cVv6wW5o",
    "de": "monch-halt-gebetskette-hinter-dem-rucken-xw6cVv6wW5o",
    "pt": "monge-segurando-contas-de-oracao-atas-das-costas-xw6cVv6wW5o",
    "id": "biksu-memegang-manik-manik-doa-di-belakang-punggung-xw6cVv6wW5o"
},
"created_at": "2025-10-19T01:42:23Z",
"updated_at": "2025-11-21T07:42:27Z",
"promoted_at": "2025-10-29T01:26:00Z",
"width": 3839,
"height": 5374,
"color": "#c00c0c",
"blur_hash": "LKMM*u9G}@D*B,spxCkC0zS5tQn+",
"description": "While you might first think of Catholicism when you hear the word \"rosary\", it is also an important object in Tibetan Buddhism. The Buddhist rosary consists of 108 beads, each representing the various worldly desires and afflictions that humans experience. By rolling the beads in their fingers and moving from one bead to the next, it helps to focus the mind, deepen one's spiritual connection and cultivate mindfulness and peace.",
"alt_description": "Monk holding prayer beads behind back",
"breadcrumbs": [],
"urls": {
    "raw": "https://images.unsplash.com/photo-1760835249990-ff68773c7c4c?ixid=M3w4MzIzOTN8MHwxJHjhbRvbXx8fHx8fHx8fDE3NjM3MjYzMjB8&ixlib=rb-4.1.0",
    "full": "https://images.unsplash.com/photo-1760835249990-ff68773c7c4c?crop=entropy&cs=srgb&fm=jpg&ixid=M3w4MzIzOTN8MHwxJHjhbRvbXx8fHx8fHx8fDE3NjM3MjYzMjB8&ixlib=rb-4.1.0&q=85",
    "regular": "https://images.unsplash.com/photo-1760835249990-ff68773c7c4c?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w4MzIzOTN8MHwxJHjhbRvbXx8fHx8fHx8fDE3NjM3MjYzMjB8&ixlib=rb-4.1.0&q=80&w=1080",
    "small": "https://images.unsplash.com/photo-1760835249990-ff68773c7c4c?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w4MzIzOTN8MHwxJHjhbRvbXx8fHx8fHx8fDE3NjM3MjYzMjB8&ixlib=rb-4.1.0&q=80&w=400",
    "thumb": "https://images.unsplash.com/photo-1760835249990-ff68773c7c4c?crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w4MzIzOTN8MHwxJHjhbRvbXx8fHx8fHx8fDE3NjM3MjYzMjB8&ixlib=rb-4.1.0&q=80&w=200"
}
```

Create collections in Postman  
Use collections to save your requests and share them with others.  
Create a Collection

# JIRA PROJECT MANAGEMENT

The screenshot shows the Jira interface for managing a project titled "Unsplash Photo Explorer - iOS Coding Challenge". The left sidebar includes links for "For you", "Recent", "Starred", "Apps", "Plans", "Spaces", "Recent", "More spaces", "Teams", and "More". The main area displays a Kanban board with four columns: "TO DO", "IN PROGRESS", "IN REVIEW", and "DONE".

- TO DO:** Contains one item: "Bonus: Create Promo video for the App" (KAN-16).
- IN PROGRESS:** Contains one item: "Write Unit Tests" (KAN-15).
- IN REVIEW:** Contains one item: "+ Create".
- DONE:** Contains 11 items:
  - Project Setup
    - KAN-1 (checked)
  - Implement Photo search functionality
    - KAN-8 (checked)
  - Create UI Screens with basic navigation and dummy data
    - KAN-5 (checked)
  - Create Network services to fetch data from Unsplash API
    - KAN-6 (checked)
  - Figma UI/UX Design
    - KAN-4 (checked)
  - Implement Author attribution and WebView
    - KAN-13 (checked)
  - Fetch random photos for initial Home screen setup
    - KAN-9 (checked)

# GIT VERSION CONTROL

The screenshot shows a GitHub repository page for 'UnsplashPhotoExplorer'. The repository is public and owned by 'TateAudrey'. The 'Code' tab is selected. The repository has 1 branch and 0 tags. The commit history shows several commits from 'TateAudrey' fixing formatting in README.md and initializing other files. The README file describes the app as a SwiftUI app for browsing photos from Unsplash. It lists features such as fetching random photos from the API, adaptive grid layout, full-screen photo detail view, camera metadata info sheet, like/unlike functionality, and efficient image loading and caching using Kingfisher. The repository has 0 forks, 0 stars, and 0 watching. It also includes sections for Releases, Packages, Languages (Swift 100%), and Suggested workflows.

TateAudrey / UnsplashPhotoExplorer

Type ⌘ to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

UnsplashPhotoExplorer Public

development 1 Branch 0 Tags Go to file Add file Code

TateAudrey Fix formatting in README.md 3d2a25d · now 24 Commits

UnsplashPhotoExplorer.xcodeproj Save progress... 1 hour ago

UnsplashPhotoExplorer Save progress... 1 hour ago

UnsplashPhotoExplorerTests Initial commit yesterday

UnsplashPhotoExplorerUITests Initial commit yesterday

README.md Fix formatting in README.md now

About

This is a native iOS and iPadOS application that allows users to search and browse high-quality photos from Unsplash API, with advanced features including caching and in-app WebView.

Readme Activity 0 stars 0 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Languages

Swift 100%

Suggested workflows

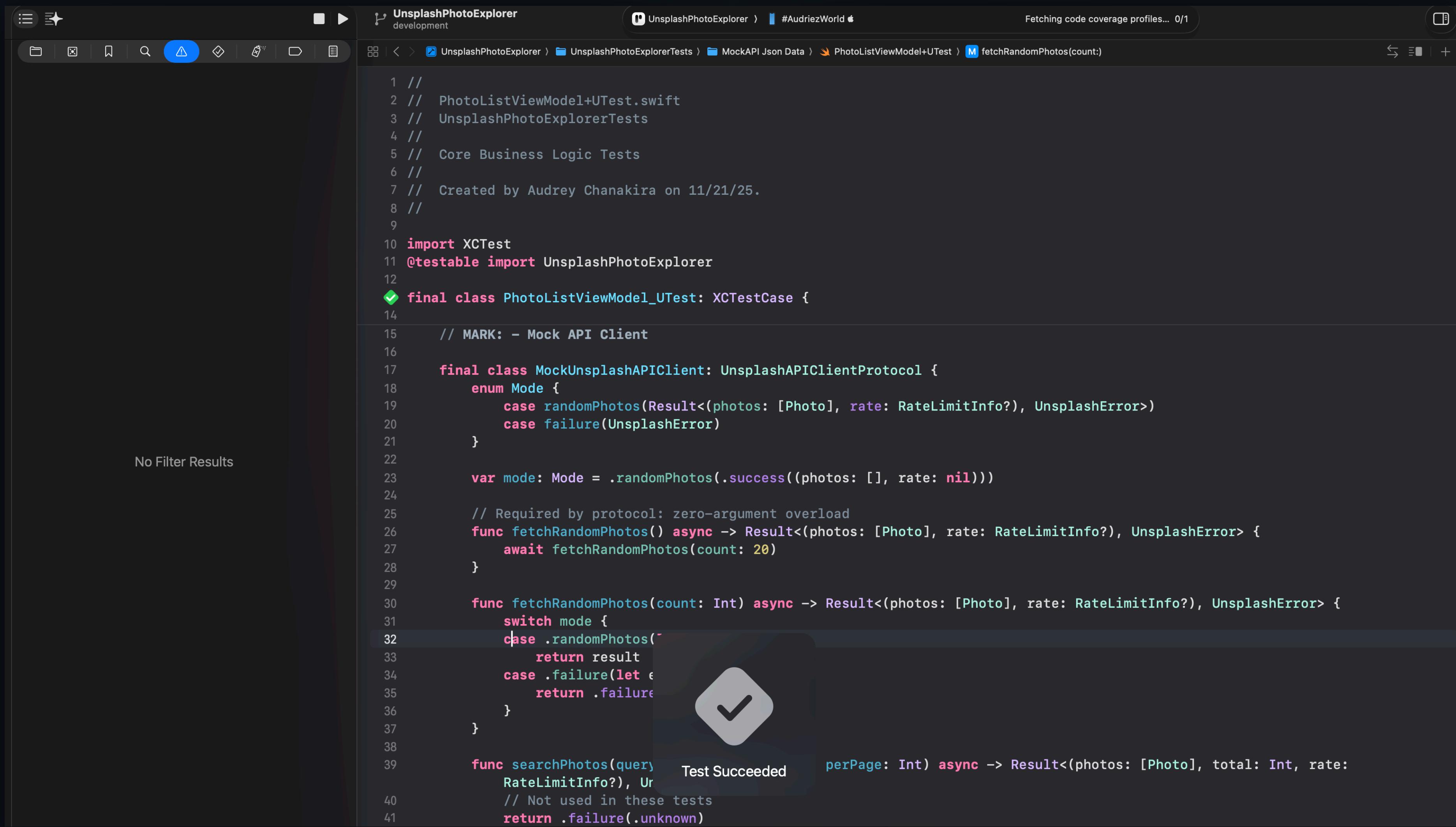
Based on your tech stack

Swift Configure

Build and test a Swift Package.

BY AUDREY CHANAKIRA

# UNIT TESTS



```
1 //  
2 //  PhotoListViewModel+UTest.swift  
3 //  UnsplashPhotoExplorerTests  
4 //  
5 //  Core Business Logic Tests  
6 //  
7 //  Created by Audrey Chanakira on 11/21/25.  
8 //  
9  
10 import XCTest  
11 @testable import UnsplashPhotoExplorer  
12  
13 final class PhotoListViewModel_UTest: XCTestCase {  
14  
15     // MARK: - Mock API Client  
16  
17     final class MockUnsplashAPIClient: UnsplashAPIClientProtocol {  
18         enum Mode {  
19             case randomPhotos(Result<(photos: [Photo], rate: RateLimitInfo?), UnsplashError>)  
20             case failure(UnsplashError)  
21         }  
22  
23         var mode: Mode = .randomPhotos(.success((photos: [], rate: nil)))  
24  
25         // Required by protocol: zero-argument overload  
26         func fetchRandomPhotos() async -> Result<(photos: [Photo], rate: RateLimitInfo?), UnsplashError> {  
27             await fetchRandomPhotos(count: 20)  
28         }  
29  
30         func fetchRandomPhotos(count: Int) async -> Result<(photos: [Photo], rate: RateLimitInfo?), UnsplashError> {  
31             switch mode {  
32                 case .randomPhotos(let result):  
33                     return result  
34                 case .failure(let error):  
35                     return .failure(error)  
36             }  
37         }  
38  
39         func searchPhotos(query: String, perPage: Int) async -> Result<(photos: [Photo], total: Int, rate: RateLimitInfo?), UnsplashError> {  
40             // Not used in these tests  
41             return .failure(.unknown)
```