iOS Developer Take-Home Assignment Guidelines & Best Practices

Objective

This document provides essential guidelines to ensure your submission meets the client's expectations. The goal is to reduce rejections and improve your chances of progressing to the interview stage.

Checklist Before Submission

Before submitting your assignment, ensure that you have met the following **mandatory** requirements:

✅ Technical Requirements

- **SwiftUI** is used (UIKit is NOT allowed).
- **MVVM architecture** is implemented correctly.
- **Live API usage** (Do not use static JSON files).
- **Pagination is implemented** on the List Page.
- **The detail page works correctly**, displaying complete information.
- **Unit tests and UI tests** are included.
- **No hardcoded values** in ViewModels (e.g., API URLs).
- **Proper state management** (e.g., no unnecessary network calls).
- **The code is well-structured**, with proper separation of concerns.

✅ Quality & Performance

- The app **does not crash** or freeze during navigation.
- **Loading states** are correctly implemented.
- **Error handling** is present (e.g., if the API call fails, an error message is shown).
- **Performance is optimized**, avoiding unnecessary computations.

✅ Code & Documentation

- The code is **clean, readable, and maintainable**.
- A **README** file is included, detailing:
    - Setup instructions.
    - Project dependencies.

    - How to run unit/UI tests.
    - Any architectural decisions made.
- **File organization is clean** (No mixing of views, models, and services in the same file).
- **Code is commented on where necessary**.
- The **README and all code** must be in **English**.

✅ User Experience (UI/UX)

- **UI is intuitive and well-structured.**
- **The detail Page displays rich information, not just an image**.
- **Navigation is smooth** (no lag or unexpected behavior).

Common Mistakes to Avoid

Many candidates have been rejected due to the following reasons. Avoid these to improve your chances.

❌ **Skipping pagination implementation.**
❌ **Using a static JSON file instead of fetching live data from the API.**
❌ **Failing to include UI tests.**
❌ **Bugs or crashes when navigating to the Detail Page.**
❌ **Lack of proper error handling (no messages when API fails).**
❌ **Hardcoding API URLs and other constants in the ViewModel.**
❌ **Messy file structure (mixing ViewModels, views, and networking code).**
❌ **Minimal or no documentation in README.**
❌ **Overcomplicating the code unnecessarily (e.g., misusing Factory Patterns).**

Code Review Before Submission

Before you submit, **perform a final self-review**:

- Run the app on a **simulator** and ensure it functions correctly.
- Check for **any unhandled errors or crashes**.
- Confirm that **pagination is functional**.
- Ensure your **Detail Page is complete** (not just an image).
- Review your **README** to ensure it's clear and complete.

Strong Example Features

For reference, an **ideal** submission includes:

- **The pagination** fetches more data dynamically.
- **A well-structured networking layer** (not inside the ViewModel).
- **Unit tests & UI tests** covering significant features.
- **Good UI design** with error and loading states.
- **No unnecessary complexity** while keeping code maintainable.

Final Notes

🚀 By following these guidelines, you **increase your chances** of progressing to the next interview stage.
⚠️ **Submissions that fail to meet the core requirements will not be considered.**
📌 If you have any doubts, review this checklist before submission.

Good luck! 🎯