PayPal Mobile Take Home Exercise

Below, you will find requirements for a mobile app that will allow users to view a collection of photos. Your task will be to:

- Build a native mobile app based on the description provided
- Construct the app in two parts:
    - Reusable SDK that contains the core model and exposes a well defined API.
    - App which utilizes the SDK and delivers the requested user experience

The purpose of this exercise is to give you an opportunity to demonstrate your engineering knowledge as well as your familiarity and experience with either Android or iOS.

Additionally, in an interview setting, it will provide a centerpiece for a discussion about your thought process and architecture applied in this engineering exercise. The intent of this exercise is to replace the majority of the "white-board coding" that is typical in engineering interviews. While we've tried to provide clear instruction, please don't hesitate to reach out with any questions you have about implementing this take-home exercise.

Common Questions

What about Interaction X?

Generally speaking, some of the instruction on this exercise has been left purposely vague. We want to see you make some decisions about how things should work.

Can I use libraries or frameworks?

You may use whatever you feel will help you get the job done. However, a word of caution: as the purpose of this exercise is to demonstrate your knowledge, the more 3rd party code utilized, the less effective the take-home will be for it's stated purpose. Use your judgment.

Which platform should I choose – iOS or Android?

Choose the platform that plays to your strengths.

How much time should I spend on this?

While there is no minimum or maximum, a good guideline is to spend 3 – 5 hours on this exercise. Quality should be prioritized over breadth of features.

App UI Guidelines

1. Allow the user to browse all available photos via a scrollable grid of thumbnails. Grid should scroll smoothly with any size data set.
2. Selecting a photo should trigger a 'detail' view. The intent of the detail view is to show a large version of the photo and, possibly, any associated photo meta data.

SDK Guidelines

1. SDK API should be well designed and easily understood.
2. SDK should contain the data model and handle interfacing to the photo data source. The interface to the data source and the model should be designed to allow for the *possibility* of connecting to a web based photo service.
3. Building the SDK as a separate library/framework is NOT necessary (but could be considered as an option for expansion). However, the SDK should be cleanly separated in the code.

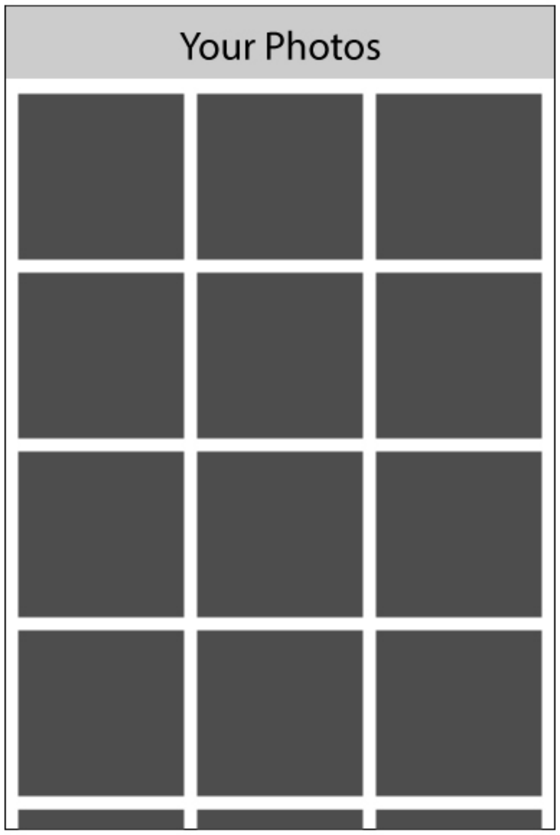
Photo Data Source

The app can access the photo content from any source including the device photo library and/or, optionally, any online photo site that exposes an API (Instagram, flickr, Google Picasa, Smugmug, etc). It is understood that the implementation may only interface to the device's photo library based on time constraints for development.

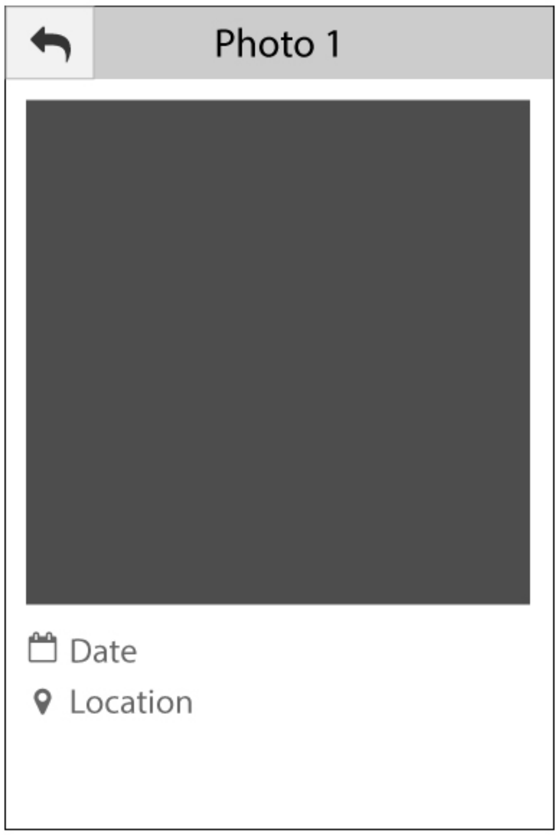
Ideas for Expansion

The above exercise is sufficient for us to begin our conversation. However, if you're interested, we encourage you to think about ways in which you can expand the scope of what we've provided here and implement (or, at least, be prepared to discuss how you would implement) any ideas that you find particularly interesting or you believe would showcase your talents and skills. To that end, we've provided some ideas below of ways this demo app could be expanded; however, feel free to formulate your own ideas as well.

- Add unit testing
- User experience improvements (graphics, sounds, animation, etc)
- Support a photo web service
- Support multiple data sources
- Add the ability to sort of photo grid based on various criteria



Grid View



Detail View