Contribution Overview

This document, and the associated Code Architecture Diagram describes my contribution to the Permutive SDK, as evidenced in the letter of recommendation by the Co-Founder, Joseph Root.

Contribution Details

I was in charge of building the initial version of the SDK from scratch. There was a feature requirement documentation that was shared. Using this documentation, I had to architect the entire SDK and follow it with implementation. This was to be done on a tight timeline. I believe the tight timeline and the scope of the work involved made this a very exciting challenge. This is one of the hardest projects I've completed.

The SDK Architecture consists of an Interface with the specified methods that are required to interact with the application making use of the SDK. This Interface is coupled with an implementation that is responsible for making use of deeper layers of the architecture, a pattern common and well recommended in OO Design in Software Development.

The second layer consisted of a Cache Manager, Reaction Manager and other modules. The core job of the Cache Manager is to hit caches and determine if something already exists in the local cache, using certain protocols and methods as mentioned in the architecture diagram. The Reaction Manager was an internal module specific to SDK functionality. Other modules consisted of generic modules part of every well designed system. The second layer also consisted of Models, an element of the Model View Controller Pattern, useful for creating entities specific to the application.

The third layer consisted primarily of the Networking Manager. This is the last resort if there is a Cache miss in a system. This module is responsible for making network calls using HTTP REST methodologies. We take care of managing threads, updating cache and updating in-memory models using this manager.

The final layer, not included in the diagram, was tests. I wrote Unit tests for all core functionality, including async expectation based tests, which are key to testing responses over the network. The entire SDK was a Universal FAT framework built in Swift 3.0

Finally, there were the Key center modules required for maintaining Key state and authentication, which is again a common pattern useful for any SDK or API.