Mobile Application Store System Architecture

Author: Eric Gieseke

Date: 9/14/2013

Introduction

This document provides the System Architecture for the Mobile Application Store.

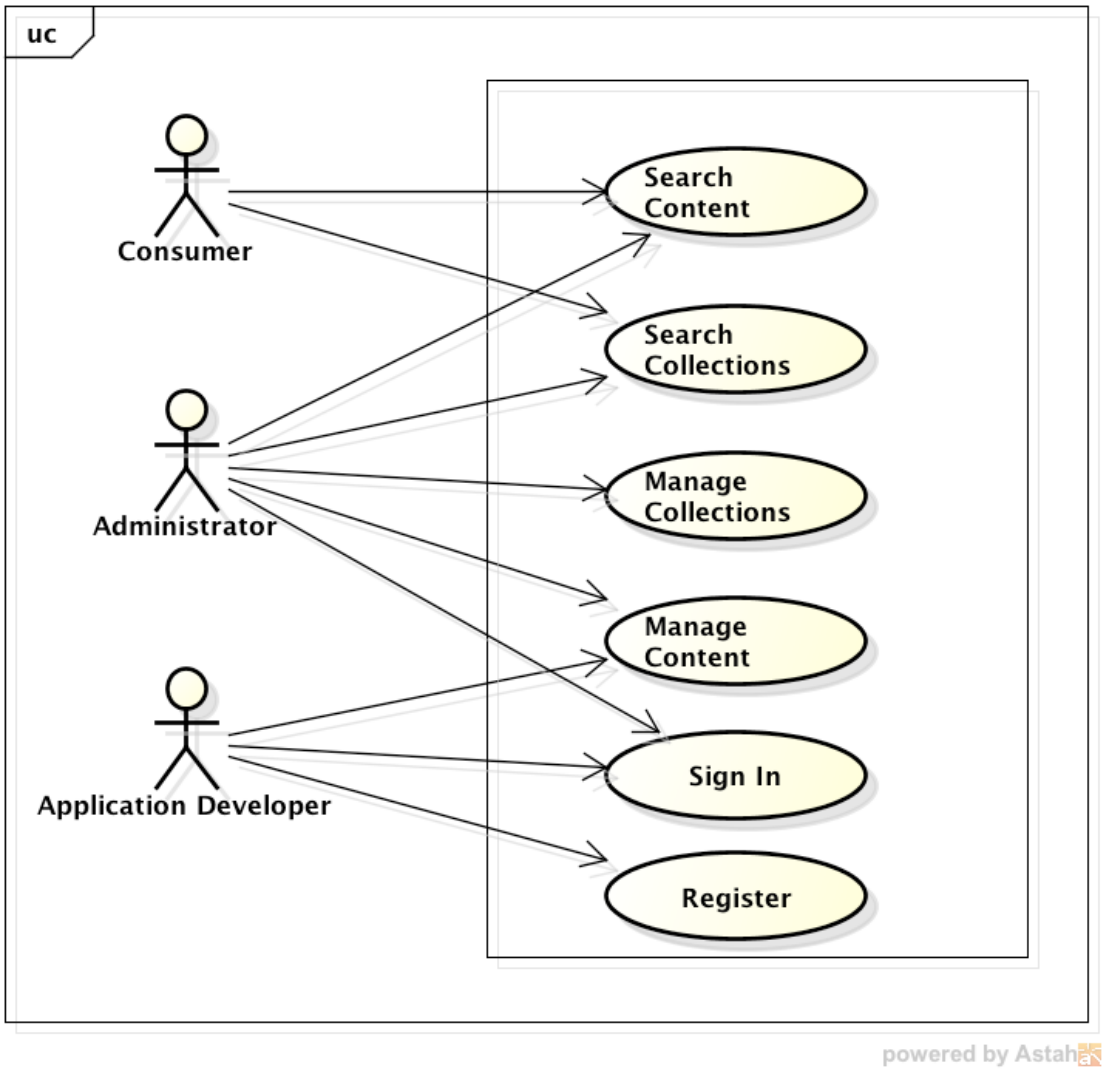
Overview

Mobile Application stores provide a source of applications, ringtones, and wallpapers for mobile consumers to download to and access from their mobile devices. A wide variety of applications are available, including games, office applications, and social applications. Ringtones and wallpapers allow users to customize their mobile experience. Both free and premium (e.g. paid) content is available for download.

Application developers provision meta data about their content to the stores product catalogue, along with the actual binary for download. Meta data includes the name of the application, a description of the application, a thumbnail image, what countries it can be distributed to, what languages are supported, what devices are supported, price (free or some amount). Meta data also includes the author of the application.

Supported Use Cases

The following use case diagram documents the high level use cases supported by the Mobile Application Store.



There are three classifications of Users.

- Consumers
- Administrators
- Application Developers

Consumers are users of the system, who will ultimately use the system to find content that they can download to their mobile devices. Consumers are able to search for and obtain product meta data. Consumers can also search search for and obtain content collections.

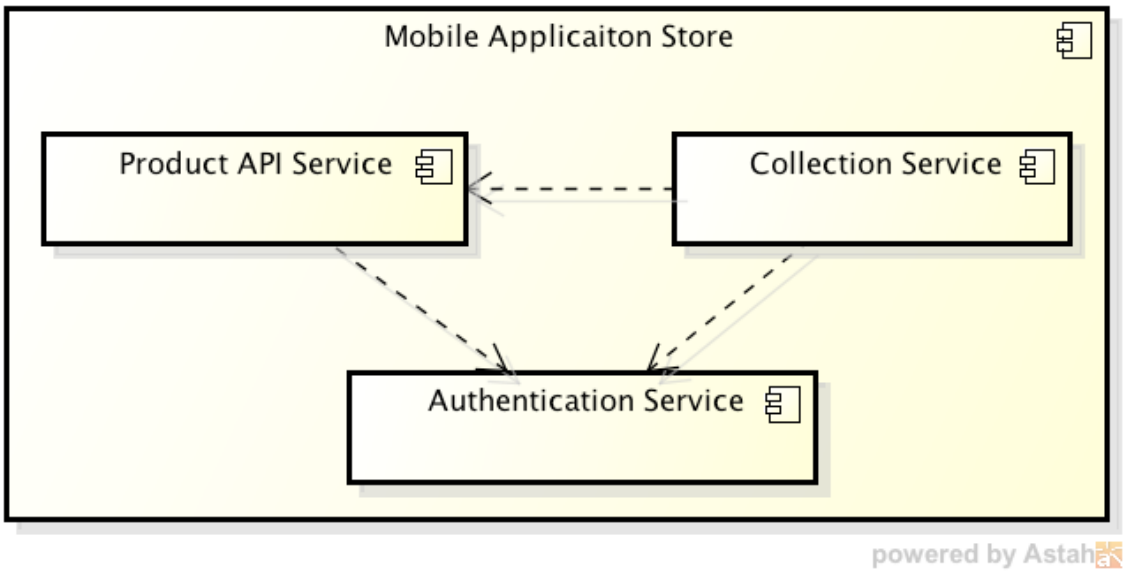
Administrators are responsible for management of the mobile application store. Administrators are able to search for content and collections. Administrators can also manage content and collections. Administrators must login to the system in order to perform administrative functions.

Application Developers are able to register and login to the system. Application Developers can manage content. Updates to content are restricted to content that they have previously created.

System Architecture

This section defines the System Architecture for the Mobile Application Store.

The following component diagram shows the primary system level components of the for the Mobile Application Store.



Product API Service

The Product API Service is responsible for management of the Product inventory. The Product API supports management of inventory, as well as ability to search, sort, and page through the contents of the product catalog.

Reference the Product API Service Design Document for design details (TBD).

Collection Service

The Collection Service supports management of and access to Collections. Both static and dynamic collections are supported where static collections define a static list of content, and dynamic collections determine their content based on a set of criteria or rules.

Collections may contain content or other collections.

Like the Product API Service, the Collection Service supports criteria based searches

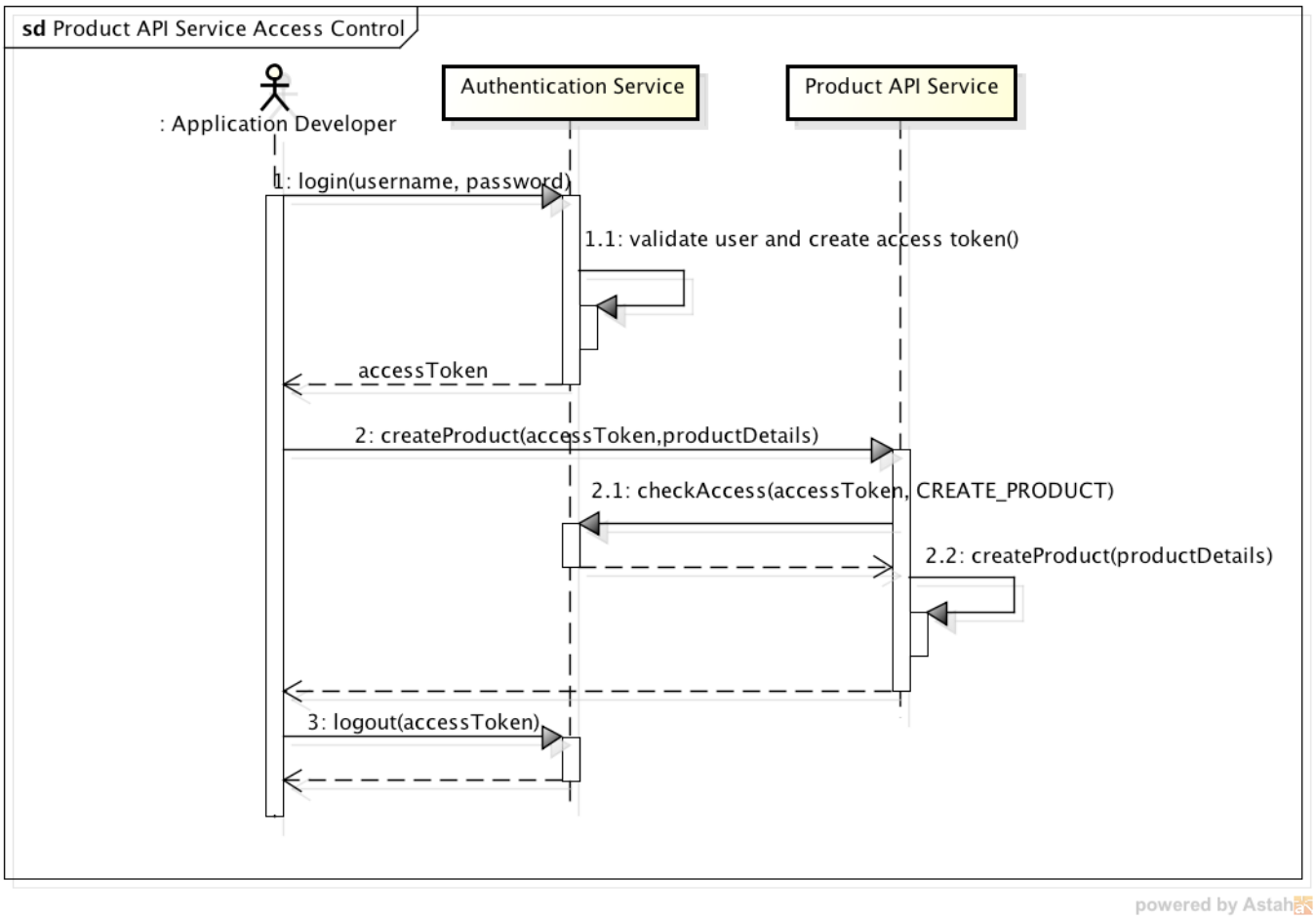
Reference the Collection Service Design Document for design details (TBD).

Authentication Service

The Authentication Service manages authentication of users and controls method level access for services. For restricted access, users must first login to the Authentication Service. Successful login returns an access token. The access token is passed to the restricted service methods. The service then delegates to the Authentication Service to check the access token to make sure that the associated user has the required permission for the method being accessed.

Reference the Authentication Service Design document for design details (TBD).

The following sequence diagram describes a sample workflow for an Application Developer logging into the system to add a new product.



Technology

This section specifies the technology choices for the Mobile Application Store.

The system will be implemented entirely in Java using the JDK 1.7.

The System Architecture follows a Service Oriented Architecture (SOA). Each of the 3 system components will be implemented as a Service.

Each service will:

- Define a Java Interface that provides a list of all public methods supported by the Service
- Provide an implementation in Java of the service interface
- Provide a factory method for accessing a Singleton Instance of the service
- Fully encapsulate the service implementation details; anything external to the service may only access the public service interface

To simplify the implementation, each of the services will be collocated within the same Java Virtual Machine (JVM). This will allow direct java level method access to the Service interfaces by the peer services and client applications.

Also, to simplify the implementation, the product catalog, collection configuration and user configuration will be maintained in memory. No persistence to a database is required.

Assume that there is a single user of the system to avoid concurrency issues.