Mobile Application Store Authentication Service Design Document

Date: 11/10/13

Author: Yoora Yi Tenen

Reviewer(s): Yetish Narayana

Introduction

This document defines the design for the Mobile Application Store Authentication Service, and is organized into the following sections:

- Overview (including a Component Diagram)
- Requirements
- Use Cases (including a Use Case Diagram)
- Implementation (including a Class Diagram and a Class Dictionary containing an Activity Diagram for the AuthenticationServiceAPI class)
- Implementation Details (including a Sequence Diagram)
- Testing
- Risks

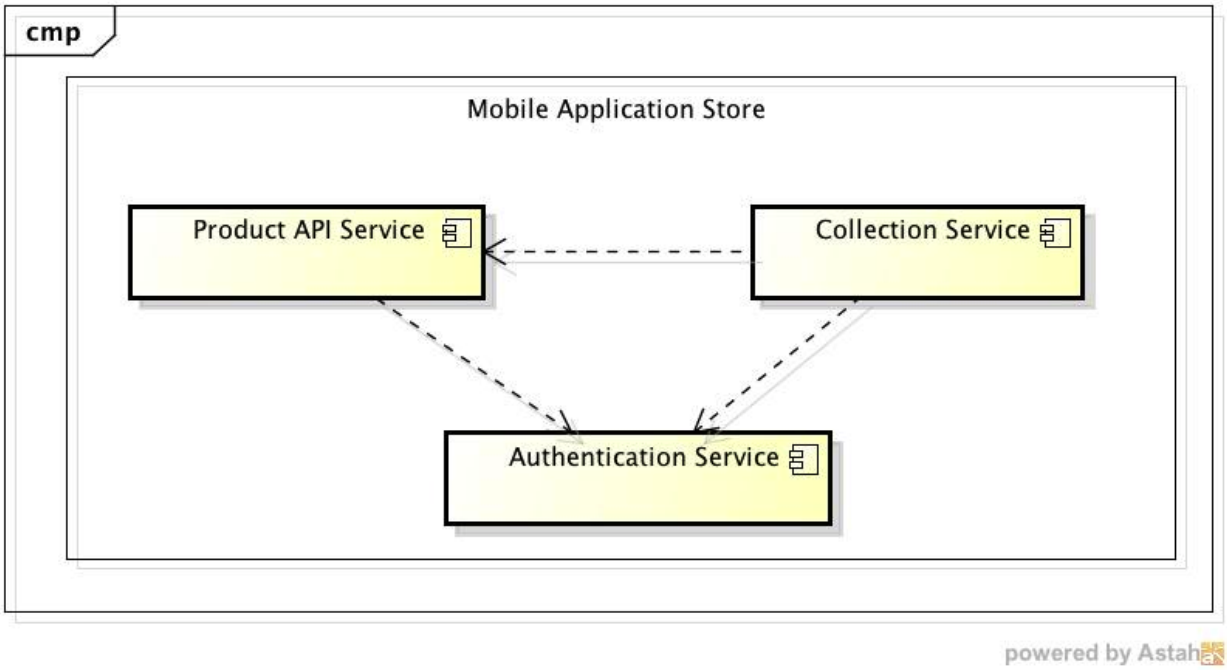
Overview

The purpose of the Mobile Application Store Authentication Service is to manage an inventory of the store's services, users, roles, and permissions, as well as oversee the enforcement of the store's authentication protocols.

A service can be associated with multiple permissions, which correspond to the public restricted methods of the service's API. A user can be associated with multiple permissions and roles (which are composed of sub-roles and permissions) that indicate whether she may execute the associated methods. A user can be assigned username and password credentials, which are required for login purposes and the generation of access tokens that are used when calling restricted methods.

The following component diagram shows the three service modules that comprise the Mobile Application Store. The Product API Service and Collection Service both depend on the Authentication Service to provide and validate the access tokens required for their respective

restricted methods.



Requirements

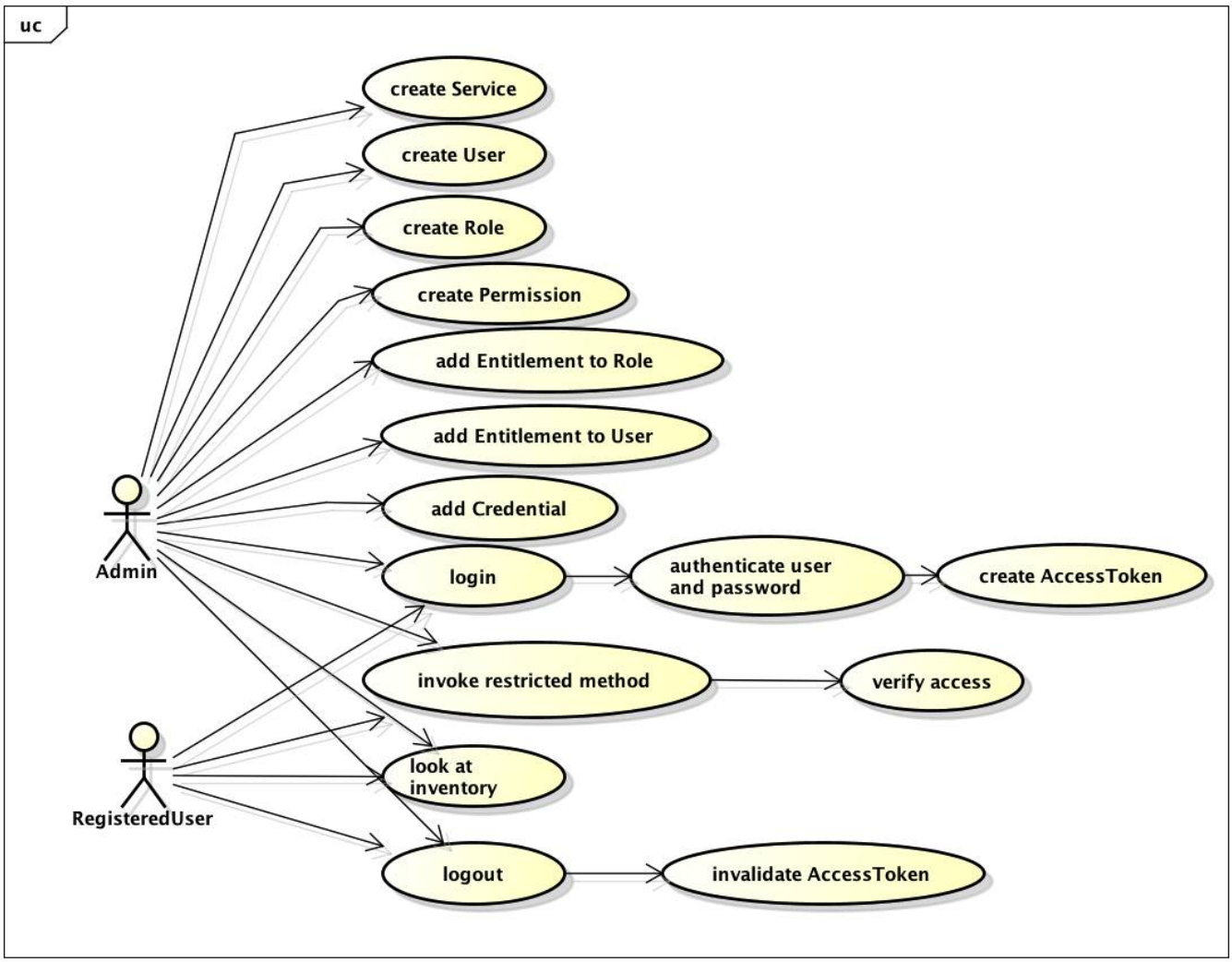
This section defines the requirements for the Mobile Application Store Authentication Service.

1. Functionality for creating Services (restricted access).
2. Functionality for creating Users (restricted access).
3. Functionality for creating Roles (restricted access).
4. Functionality for creating Permissions (restricted access).
5. Functionality for creating Credentials (including password hashing) and adding to Users (restricted access).
6. Functionality for adding entitlements (Roles or Permissions) to Roles (restricted access).
7. Functionality for adding entitlements (Roles or Permissions) to Users (restricted access).
8. Login functionality (including AccessToken creation).
9. Logout functionality (including AccessToken invalidation).
10. Support for invoking restricted methods (including AccessToken validation).
11. Exception handling as appropriate for data import/parsing, user authentication, and access token validation errors.

No memory persistence is required and a single user/thread may be assumed.

Use Cases

1. Admin users create new Services.
2. Admin users create new Users.
3. Admin users create new Roles.
4. Admin users create new Permissions
5. Admin users add entitlements (existing Roles or Permissions) to existing Roles.
6. Admin users add entitlements (existing Roles or Permissions) to existing Users.
7. Admin users create and add new Credentials to existing Users.
8. Registered users (may be admin) login to the Authentication Service. If authenticated, an AccessToken guid is returned.
9. Registered users (may be admin) invoke restricted methods of the Product API, Collection API, or Authentication API services, submitting access tokens for verification.
10. Registered users (may be admin) look at the Authentication Service's inventory of Services, Users, Roles, and Permissions.
11. Registered users (may be admin) logout of the Authentication Service, which invalidates the associated AccessToken.

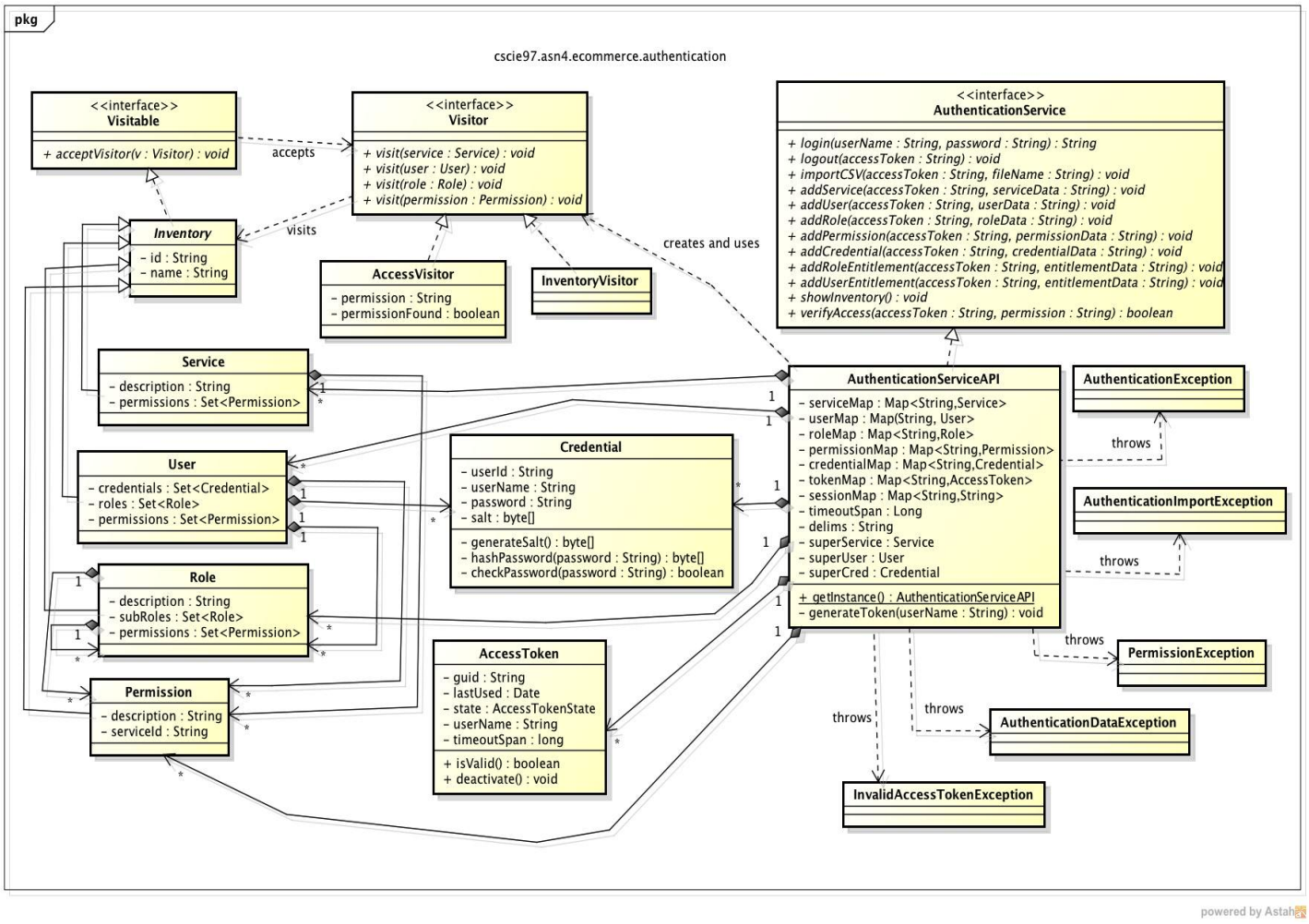


Implementation

Class Diagram

The following class diagram defines the Authentication Service implementation classes contained within the package “cscie97.asn4.ecommerce.authentication”.

In order to reduce the number of lines, the AuthenticationServiceAPI’s dependency points to the Visitor interface (rather than two arrows to the AccessVisitor and InventoryVisitor classes) and the Visitor interface’s dependency points to the Inventory abstract class (rather than four arrows to the Service, User, Role, and Permission classes).



Class Dictionary

This section specifies the class dictionary for the Authentication Service. The classes should be defined within the package “cscie97.asn4.ecommerce.authentication”.

All classes implement standard public getter (and setter, as needed) methods for their properties and associations.

Visitable

The Visitable interface is used to indicate that an object may accept a Visitor, and is implemented by the Inventory abstract class. The Visitor pattern is used by the Authentication Service in order to separate from the Inventory objects the operations related to displaying inventory details and the operations related to access verification.

Methods

Method Name	Signature	Description
acceptVisitor	(v:Visitor):void	Public method that accepts a Visitor and invokes its visit method.

Inventory

The Inventory class implements the Visitable interface and is the abstract class for the objects considered to compose the inventory of the Authentication Service (in this version this includes Services, Users, Roles, and Permissions).

Properties

Property Name	Type	Description
id	String	Private unique identifier for the Inventory item. Is not case-sensitive.
name	String	Private name of the Inventory item.

Service

The Service class extends the Inventory abstract class and represents a service of the Mobile Application Store (in this version this includes the Product, Collection, and Authentication services). A Service may have zero or more permissions, which should correspond to its restricted public methods. In addition to the properties defined in the Inventory class, the Service class has property “descriptions” and association “permissions”.

Methods

Method Name	Signature	Description
acceptVisitor	(v:Visitor):void	Public method that accepts a Visitor and invokes its visit method, passing the Service as a parameter.

Properties

Property Name	Type	Description
description	String	Private description of the Service.

Associations

Association Name	Type	Description
permissions	Set<Permission>	Private set of Permissions that the Service has.

User

The User class extends the Inventory abstract class and represents a registered user of the Mobile Application Store. A User may have zero or more roles and permissions, which indicate which restricted methods she can execute, and is able to login to the Mobile Application Store using assigned credentials (username and password). In addition to the properties defined in the Inventory class, the User class has associations “credentials”, “roles”, and “permissions”.

Methods

Method Name	Signature	Description
acceptVisitor	(v:Visitor):void	Public method that accepts a Visitor and invokes its visit method, passing the User as a parameter.

Associations

Association Name	Type	Description
credentials	Set<Credential>	Private set of Credentials associated with the User.
roles	Set<Role>	Private set of Roles associated with the User.
permissions	Set<Permission>	Private set of Permissions that the User has.

Credential

The Credential class represents username/password credentials that can be associated with a user and are required for login to the Mobile Application Store.

Methods

Method Name	Signature	Description
generateSalt	():byte[]	Private helper method for generating the salt for hashing a password. Throws NoSuchAlgorithmException if the salting algorithm is invalid.
hashPassword	(password:String):byte[]	Private helper method for hashing a password; used for the stored password when creating a new Credential and used for the password that is supplied during login to be validated. Uses PBKDF2 algorithm. Throws NoSuchAlgorithmException and InvalidKeySpecException if the hashing algorithm or key spec is invalid.
checkPassword	(password:String):boolean	Public method for comparing a password to a Credential's stored password hash.

Properties

Property Name	Type	Description
userId	String	Private identifier of the User associated with the Credential.
userName	String	Private unique username for the Credential.
password	byte[]	Private password for the Credential. Password is stored as a hashed value.

salt	byte[]	Private salt that was used to hash the Credential password. Used to hash passwords to compare against the stored password.

Role

The Role class extends the Inventory abstract class and represents a defined role in the Mobile Application Store. A Role is a grouping of sub-roles and permissions and is a means of assigning permissions to users in a standardized fashion. In addition to the properties defined in the Inventory class, the Role class has the property “description” and the associations “subRoles” and “permissions”.

Methods

Method Name	Signature	Description
acceptVisitor	(v:Visitor):void	Public method that accepts a Visitor and invokes its visit method, passing the Role as a parameter.

Properties

Property Name	Type	Description
description	String	Private description of the Role.

Associations

Association Name	Type	Description
subRoles	Set<Role>	Private set of sub-roles that the Role contains.
permissions	Set<Permission>	Private set of Permissions that the Role contains.

Permission

The Permission class extends the Inventory abstract class and represents a defined Permission

in the Mobile Application Store. Permissions correspond to the public restricted methods offered by the service APIs. In addition to the properties defined in the Inventory class, the Permission class has the properties “description” and “serviceId”.

Methods

Method Name	Signature	Description
acceptVisitor	(v:Visitor):void	Public method that accepts a Visitor and invokes its visit method, passing the Permission as a parameter.

Properties

Property Name	Type	Description
description	String	Private description of the Permission.
serviceId	String	Private identifier of the Service associated with the Permission.

Visitor

The Visitor interface is used as a template for the Authentication Service’s Visitors, InventoryVisitor and AccessVisitor. The Visitor pattern is used by the Authentication Service in order to separate from the Inventory objects the operations related to displaying inventory details and the operations related to access verification. Visitor makes uses of an overloaded visit method to which the concrete Inventory object is passed.

Methods

Method Name	Signature	Description
visit	(service:Service):void	Public method that performs the Visitor’s specialized operation on the Service object.
visit	(user:User):void	Public method that performs the Visitor’s specialized operation on the User object.

visit	(role:Role):void	Public method that performs the Visitor's specialized operation on the Role object.
visit	(permission:Permission):void	Public method that performs the Visitor's specialized operation on the Permission object.

InventoryVisitor

The InventoryVisitor class implements the Visitor interface and provides a Visitor for displaying the inventory (Services, Users, Roles, and Permissions) of the Authentication Service.

Methods

Method Name	Signature	Description
visit	(service:Service):void	Public method that prints the id, name, description, and permissions of the Service.
visit	(user:User):void	Public method that prints the id, name, credentials, roles, and permissions of the User.
visit	(role:Role):void	Public method that prints the id, name, description, sub-roles, and permissions of the Role.
visit	(permission:Permission):void	Public method that prints the id, name, description, and associated Service of the Permission.

AccessVisitor

The AccessVisitor class implements the Visitor interface and provides a Visitor that looks for a specific Permission. The AccessVisitor is leveraged by the Authentication Service's method for verifying access for a restricted method.

Methods

Method Name	Signature	Description
visit	(service:Service):void	Public method that looks for a Permission within a Service.
visit	(user:User):void	Public method that looks for a Permission within a User's Roles and Permissions.
visit	(role:Role):void	Public method that looks for a Permission within a Role's sub-Roles and Permissions.
visit	(permission:Permission):void	Public method that checks if a Permission matches the given Permission.

Properties

Property Name	Type	Description
permission	String	Private identifier for the Permission being looked for. Is set upon construction of the AccessVisitor instance.
permissionFound	boolean	Private flag for whether the Permission being looked for has been found by the AccessVisitor instance.

AccessToken

The AccessToken class represents a token which can be used by a User to verify her identity and Permissions to execute a restricted method. An AccessToken is generated for a User upon successful login and authentication and expires upon logout. An AccessToken may also expire if it is not used within a span of time designated by the Authentication Service.

Methods

Method Name	Signature	Description
isValid	():boolean	Public method that returns true if the AccessToken is valid; false if not. Updates the

		state and lastUsed property of the AccessToken accordingly.
deactivate	():void	Public method that deactivates the AccessToken.

Properties

Property Name	Type	Description
guid	String	Private unique identifier of the AccessToken; is also the access token String that is passed as a parameter to restricted methods.
lastUsed	long	Private time in milliseconds when the AccessToken was last used. Is reset whenever AccessToken is successfully used.
state	AccessTokenState (enum)	Private state of the AccessToken. Can be “active” or “expired”.
userName	String	Private user name of the credential the AccessToken was generated for.
timeoutSpan	long	Private period of time after which the AccessToken expires.

AuthenticationService

The AuthenticationService interface provides the public methods for the Authentication Service’s API.

Methods

Method Name	Signature	Description
login	(userName:String,password:String):String	Public method for logging into Mobile Application Store. AccessToken is generated and its guid is returned upon successful authentication of user name and password. Throws AuthenticationException on error authenticating credentials.
logout	(accessToken:String):void	Public method for logging out of the Mobile Application Store. AccessToken is invalidated upon logout. Throws InvalidAccessTokenException on error evaluating access token.
importCSV	(accessToken:String,fileName:String):void	Public, restricted method for importing data from CSV-formatted file to be processed by the Authentication Service. Delegates to other methods for creating objects from the data. Throws AuthenticationImportException on error accessing or processing the file. Throws PermissionException if user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired. Throws AuthenticationDataException if invoked factory method finds error in data.
addService	(accessToken:String,serviceData:String):void	Public, restricted method for creating a new Service. Throws

		AuthenticationDataException on error parsing or validating serviceData. Throws PermissionException if associated user does not have the permission to execute the method.
addUser	(accessToken:String,userData:String):void	Public, restricted method for creating a new User. Throws AuthenticationDataException on error parsing or validating userData. Throws PermissionException if associated user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.
addRole	(accessToken:String,roleData:String):void	Public, restricted method for creating a new Role. Throws AuthenticationDataException on error parsing or validating roleData. Throws PermissionException if associated user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.
addPermission	(accessToken:String,permissionData:String):void	Public, restricted method for creating a new Permission. Throws AuthenticationDataException on error parsing or validating permissionData. Throws PermissionException if associated user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.

addCredential	(accessToken:String,credentialData:String):void	Public, restricted method for creating and adding a Credential to an existing User. Throws AuthenticationDataException on error parsing or validating credentialData. Throws PermissionException if associated user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.
addRoleEntitlement	(accessToken:String,entitlementData:String):void	Public, restricted method for adding an existing entitlement (Role or Permission) to an exisiting Role. Throws AuthenticationDataException on error parsing or validating entitlementData. Throws PermissionException if associated user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.
addUserEntitlement	(accessToken:String,entitlementData:String):void	Public, restricted method for adding an existing entitlement (Role or Permission) to an exisiting User. Throws AuthenticationDataException on error parsing or validating entitlementData. Throws PermissionException if associated user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist

		or is expired.
showInventory	():void	Public method for displaying the inventory of the Authentication Service (the Services, Users, Roles, and Permissions).
verifyAccess	(accessToken:String,permission:String):boolean	Public method for verifying whether a User has the given permission and an active access token for executing a restricted method. Throws InvalidAccessTokenException if access token does not exist or is expired.

AuthenticationException

AuthenticationException is thrown by AuthenticationServiceAPI when an error is encountered authenticating a user's credentials upon login (ex. username is not in the system or user is already logged in).

AuthenticationImportException

AuthenticationImportException is thrown by AuthenticationServiceAPI when an error is encountered importing data to be processed by the Authentication Service (ex. file has incorrect number of fields).

AuthenticationDataException

AuthenticationDataException is thrown by AuthenticationServiceAPI when an error is encountered creating objects from data imported into the Authentication Service (ex. data is incorrectly formatted).

InvalidAccessTokenException

InvalidAccessTokenException is thrown by AuthenticationServiceAPI when an error is encountered when using an access token (ex. expired access token is used for a restricted method or access token doesn't exist).

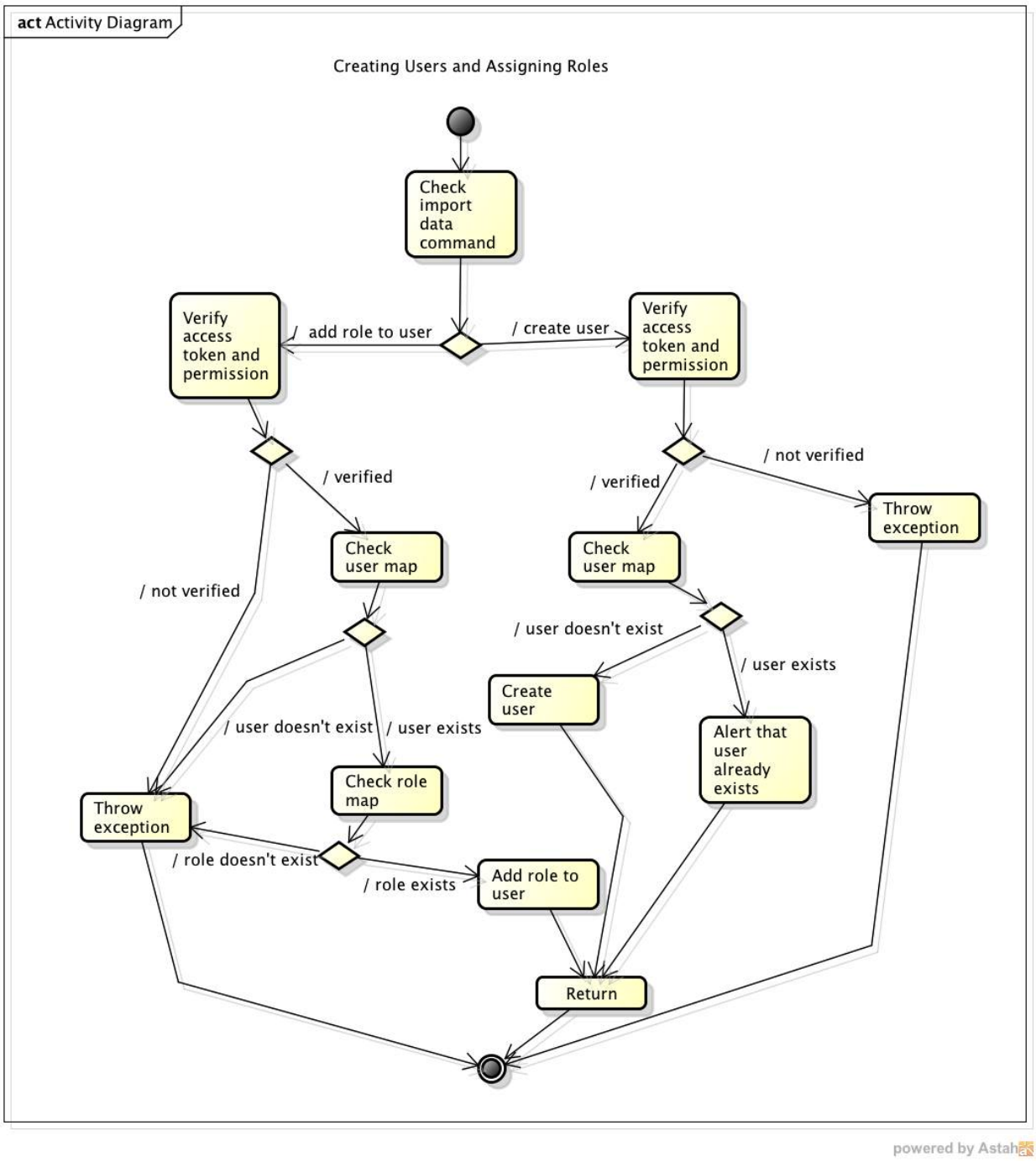
PermissionException

PermissionException is thrown by AuthenticationServiceAPI when an error is encountered checking a user's permission to execute a method (ex. user does not have the permission).

AuthenticationServiceAPI

The AuthenticationServiceAPI is responsible for maintaining the inventory of Services, Users, Roles, and Permission in the Mobile Application Store and oversees the authentication of users and the activities they are permitted to conduct. It implements the AuthenticationService interface and is instantiated as a Singleton, using the static method getInstance().

The AuthenticationServiceAPI makes use of private maps for maintaining its inventory. The following Activity Diagram depicts the flow of creating Users and assigning Roles to Users (for the sake of illustration assuming data that only calls for User creation or Role assigning).



Methods

Method Name	**Signature**	**Description**
login	(userName:String,password:	Public method for logging into

	String):String	Mobile Application Store. AccessToken is generated using private helper method generateToken and its guid is returned upon successful authentication of user name and password. Throws AuthenticationException on error authenticating credentials.
logout	(accessToken:String):void	Public method for logging out of the Mobile Application Store. AccessToken is invalidated upon logout. Throws InvalidAccessTokenException if an error is encountered evaluating the AccessToken.
importCSV	(accessToken:String,fileName:String):void	Public, restricted method for importing data from CSV-formatted file to be processed by the Authentication Service. Delegates to other methods for creating objects from the data. Throws AuthenticationImportException on error accessing or processing the file. Throws PermissionException if user does not have the permission to execute the method.Throws InvalidAccessTokenException if access token does not exist or is expired. Throws AuthenticationDataException if invoked factory method finds error in data.
addService	(accessToken:String,serviceData:String):void	Public, restricted method for creating a new Service. Throws AuthenticationDataException on error parsing or validating

		serviceData. Throws PermissionException if user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.
addUser	(accessToken:String,userData:String):void	Public, restricted method for creating a new User. Throws AuthenticationDataException on error parsing or validating userData. Throws PermissionException if user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.
addRole	(accessToken:String,roleData:String):void	Public, restricted method for creating a new Role. Throws AuthenticationDataException on error parsing or validating roleData. Throws PermissionException if user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.
addPermission	(accessToken:String,permissionData:String):void	Public, restricted method for creating a new Permission. Throws AuthenticationDataException on error parsing or validating permissionData. Throws PermissionException if user does not have the permission to execute the method. Throws InvalidAccessTokenException

		if access token does not exist or is expired.
addCredential	(accessToken:String,credentialData:String):void	Public, restricted method for creating and adding a Credential to an existing User. Throws AuthenticationDataException on error parsing or validating credentialData. Throws PermissionException if user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.
addRoleEntitlement	(accessToken:String,entitlementData:String):void	Public, restricted method for adding an existing entitlement (Role or Permission) to an exisiting Role. Throws AuthenticationDataException on error parsing or validating entitlementData. Throws PermissionException if user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.
addUserEntitlement	(accessToken:String,entitlementData:String):void	Public, restricted method for adding an existing entitlement (Role or Permission) to an exisiting User. Throws AuthenticationDataException on error parsing or validating entitlementData. Throws PermissionException if user does not have the permission to execute the method. Throws InvalidAccessTokenException if access token does not exist or is expired.

showInventory	():void	Public method for displaying the inventory of the Authentication Service (the Services, Users, Roles, and Permissions). Utilizes an InventoryVisitor.
verifyAccess	(accessToken:String,permission:String):boolean	Public method for verifying whether a User has the given permission and an active access token for executing a restricted method. Utilizes an AccessVisitor. Throws InvalidAccessTokenException if access token does not exist or is expired.
getInstance	():AuthenticationServiceAPI	Public method for returning a reference to the single static instance of AuthenticationServiceAPI.
generateToken	(userName:String):AccessToken	Private helper function for creating an AccessToken for a given user name.

Properties

Property Name	**Type**	**Description**
timeoutSpan	Long	Private period of time after which an AccessToken expires.
delims	String	Private default delimiters for imported files. Assumes unescaped comma as delimiter.
sessionMap	Map<String, String>	Private map for maintaining set of user sessions. Used to confirm that user is not already logged in.
superService	Service	Private Service used for super user admin of Authentication Service.

superUser	User	Private User used for super user admin of Authentication Service.
superCred	Credential	Private Credential used for super user admin of Authentication Service.

Associations

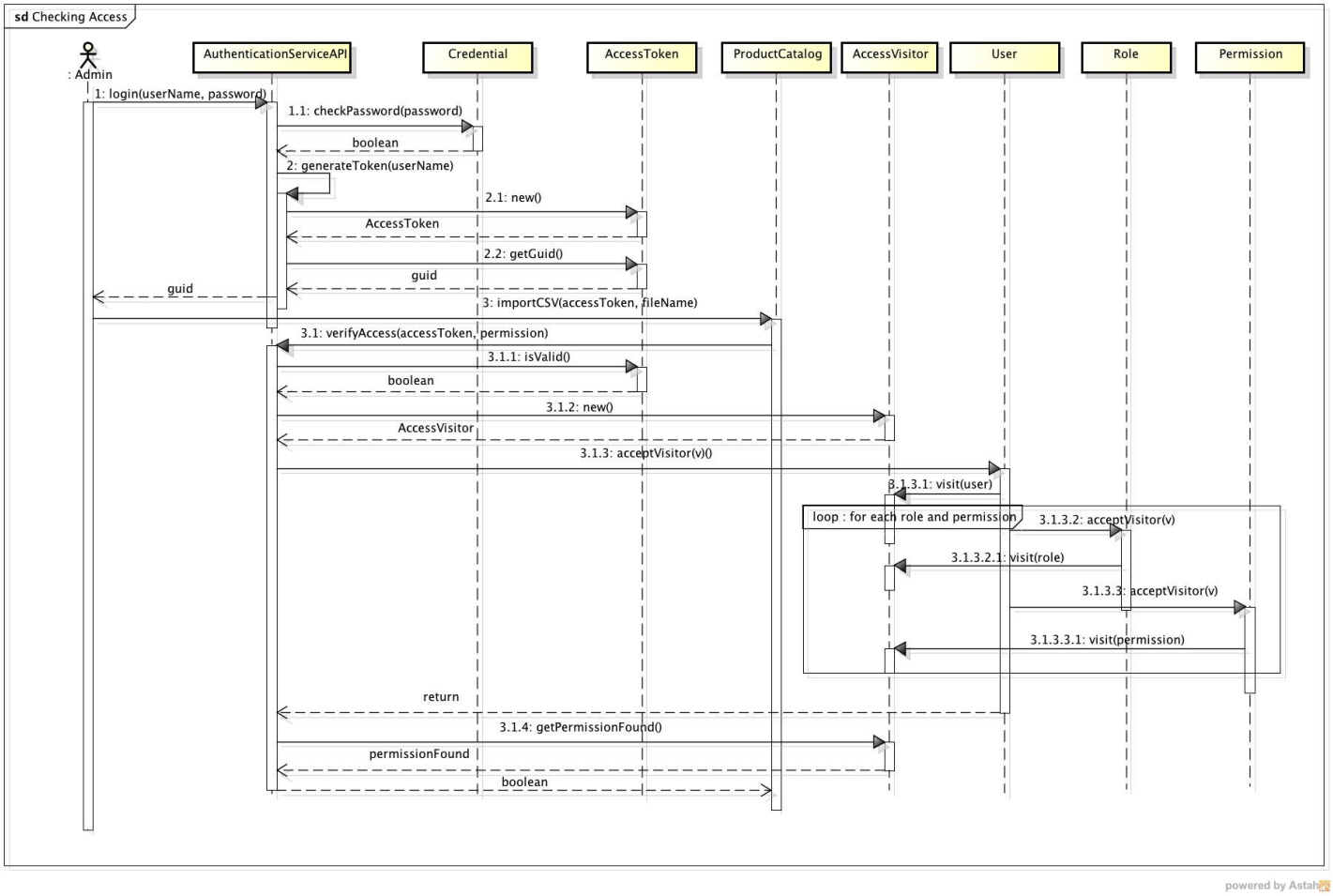
Association Name	**Type**	**Description**
serviceMap	Map<String, Service>	Private association for maintaining active set of valid Services. Key is Service id and value is the associated Service.
userMap	Map<String, User>	Private association for maintaining active set of valid Users. Key is User's id and value is the associated User.
roleMap	Map<String,Role>	Private association for maintaining active of valid Roles. Key is the Role id and value is the associated Role.
permissionMap	Map<String, Permission>	Private association for maintaining active set of valid Permissions. Key is the Permission id and value is the associated Permission.
credentialMap	Map<String, Credential>	Private association for maintaining active set of valid Credentials. Key is the userName and value is the associated Credential.
tokenMap	Map<String, AccessToken>	Private association for maintaining the set of generated AccessTokens. Key is AccessToken guid and value is the associated AccessToken.

Implementation Details

The Authentication Service API is accessed via the AuthenticationServiceAPI Singleton class. AuthenticationServiceAPI has public methods supporting login, logout, Authentication Service inventory object creation, inventory display, and access verification. The Authentication Service API controls access to the Mobile Application Store APIs via the authentication of user credentials. The Product API and the Collection Service API both depend on the Authentication Service API to verify whether a user can execute a restricted method based on her permissions and access token.

The following sequence diagram depicts an admin user logging in and using the AccessToken generated by the AuthenticationServiceAPI upon successful user authentication to call the restricted Product API method importCSV. The Product API in turn invokes the AuthenticationServiceAPI method verifyAccess to confirm that this user can execute the restricted method. AuthenticationServiceAPI checks the access token and utilizes an AccessVisitor as a kind of iterator to check whether the User has the given permission within its associated roles and permissions.

A similar flow follows for restricted method calls on the Collection Service API.



Testing

A test driver class named TestDriver should be implemented in package cscie97.asn4.test with a static main() method that accepts as parameters the file names for five CSV-formatted files containing Country, Device, Content, Collection, and Authentication data. The main() method should first call the getInstance() methods of the three Mobile Application Store APIs to instantiate the services' Singletons. For testing purposes, "super user" data will need to be hard-coded into the Authentication Service in order to bypass the standard authentication process and permit this initial restricted method call to import the Authentication data.

After the super user has logged in, imported data, and logged out, the process will be repeated with a product admin, a product developer, and a collection admin user in order to test the authentication procedures for the ProductCatalog and CollectionService APIs.

Code compiles with the command:
javac cscie97/asn4/ecommerce/product/.java cscie97/asn4/ecommerce/collection/*.java*

cscie97/asn4/ecommerce/authentication/.java cscie97/asn4/test/*.java*

To run the Test Driver that shows the standard operation of the program, run command:
java -cp . cscie97.asn4.test.TestDriver authentication.csv countries.csv devices.csv products.csv collections.csv

To run the Test Driver that shows the examples of exception handling when processing Authentication data, run the above command, replacing the authentication.csv argument with one of the following:
authentication2.csv
authentication3.csv
authentication4.csv
authentication5.csv

To run the Test Driver that shows the program where a user calls a method with an inactive token, run command:
java cp . cscie97.asn4.test.TestDriver2 authentication.csv countries.csv devices.csv products.csv collections.csv

To run the Test Driver that shows the program where a user calls a method for which she doesn't have the permission, run command:
java cp . cscie97.asn4.test.TestDriver3 authentication.csv countries.csv devices.csv products.csv collections.csv

To run the TestDriver that shows the program where an unknown user tries to log in, run command:
java cp . cscie97.asn4.test.TestDriver4 authentication.csv countries.csv devices.csv products.csv collections.csv

Risks

For the scope of this assignment, it is assumed that the number of generated AccessTokens will remain very small. In a more heavily-loaded system, it may make more sense to delete AccessTokens once they expire, rather than continuing to store the expired AccessTokens, since expired AccessTokens and non-existent AccessTokens both generate Exceptions.

An AccessToken may expire in the middle of a series of a User's operations. In this implementation, an Exception is thrown for an expired AccessToken, but in a live system, this would better be handled by alerting the User that the session is about to expire and to take action to extend it.