SquareDesk Authentication Service Design Document

Date: November 20, 2014
Author: Roland L. Galibert
Reviewer: Darren Willis

Contents

Introduction

This document defines the design for the Authentication Service component of the SquareDesk solution, a home office space rental service described in greater detail in the section below.

Overview

SquareDesk is a new service which allows people to rent out portions of their homes as office space. This service will benefit both office space providers, who will have a way of making additional income, and the people who will rent those spaces, who require an inexpensive, quiet place to work as an alternative to a high-price space in an office building or a crowded Starbuck's or public library. The operators of the SquareDesk solution, who will take a small commission in return for use of the service, will also benefit.

A potential office space provider will visit the SquareDesk site to register and provide details about the space he/she has for rent. Potential renters can then go to the SquareDesk web site to search for office

spaces that are near their location, and that meet their requirements in terms of features offered (e.g. rent amount charged, WIFI, allows pets, etc.). The SquareDesk semantic search engine, another selling point of the SquareDesk solution, will return the Renter a list of appropriate office spaces, from which the renter will choose the one that suits him/her best, then reserve and pay for the space.

Finally, SquareDesk will also allow providers and renters to be rated to ensure a quality experience for all parties involved.

Requirements

The specific requirements of the SquareDesk Authentication Service are described in detail in the corresponding requirements document, with clarification and additional specifications provided from the discussion board.

In general, the SquareDesk solution must have provisions for logging users in, logging them out, and ensuring that they have permission to execute restricted methods. This should all be based on a system that maintains credentials (user name and hashed password) for each user and that also supports the use of time-limited access tokens to control the execution of restricted methods. These access tokens must have a unique ID, expiration time and state (active or expired).

Login – The login process should validate the user's name and password and issue an access token to allow the user to access the restricted methods to which he/she is authorized.

Logout – The logout process should mark the given user's access token as invalid, disabling it from further use.

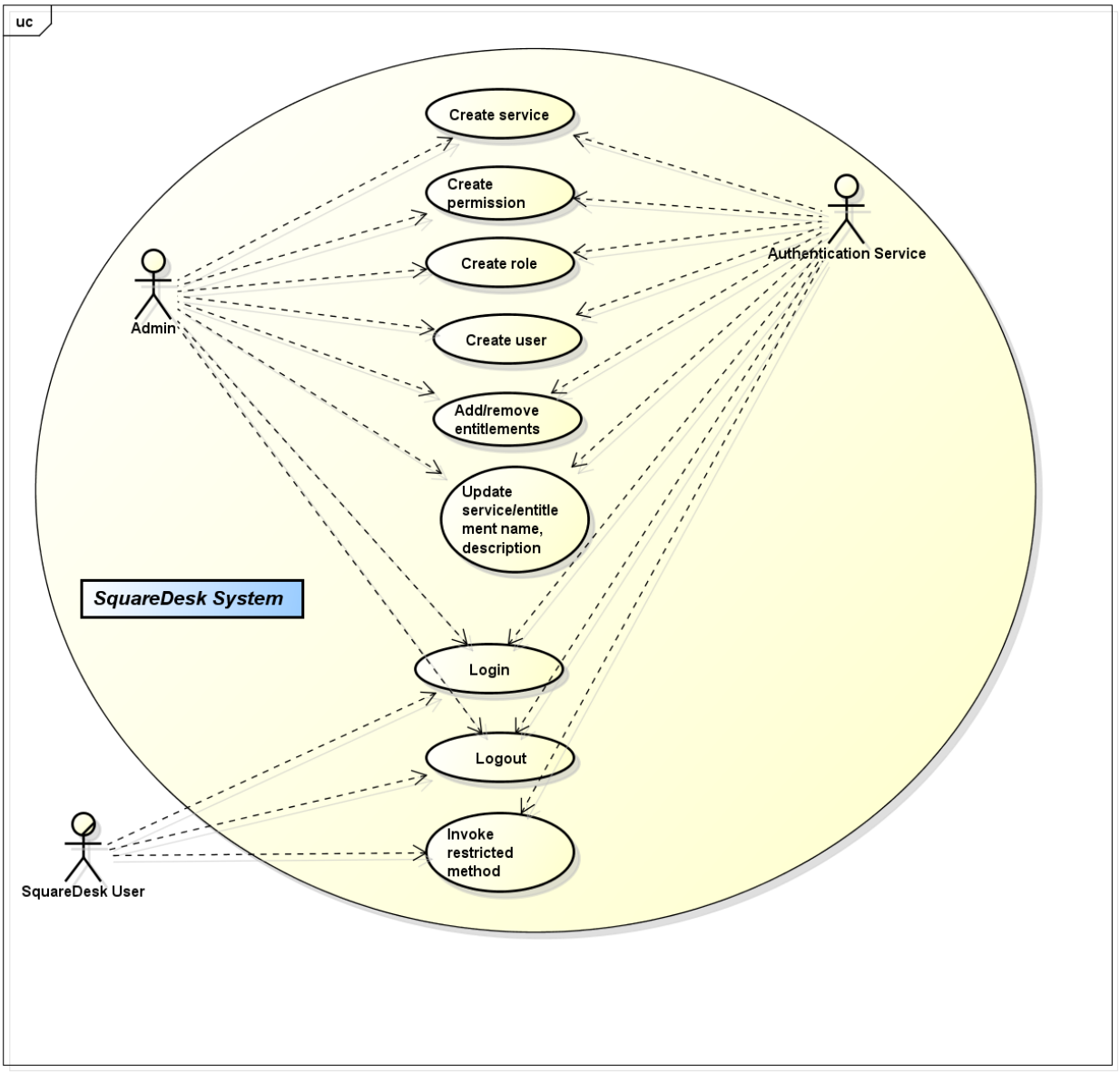
Restricted methods – A user wishing to execute a given operation must pass that operation his/her access token. The authentication service will then allow/disallow the user to execute the operation on the basis of the access token, taking into consideration the user's permissions and other factors such as whether the accesss token has expired.

The authentication service must also support the following objects:

- Service (unique ID, name and description) – examples of these services include the already-implemented OfficeProviderServiceAPI and RenterAPI. Each service object should aggregate permissions to call restricted methods in the corresponding API/service (e.g. creating an office space)
- Permission (unique ID, name and description) – This object represents an entitlement required to access a restricted method (e.g. OfficeProviderServiceAPI createOfficeSpace()). A Service object may be associated with one or more Permission objects, while a User may be associated with zero or more Permission objects.
- Role (including unique ID, name and description) – This object allows Permission objects and other Role objects to be grouped together. A user may be associated with zero or more Role objects, and thus gains the permissions included in that Roles and sub-roles.
- User (user name and password)

Use Cases

The present design document identifies an additional actor (a SquareDesk Admin) and also defines additional use cases for the SquareDesk User actor identified in previous implementation stages. The Authentication Service, the focus of this design document, has also already been identified as an actor in previous implementation stages. The following graphic provides a general overview of these actors and their related use cases:



The design has identified the following use cases for a SquareDesk Admin:
1. Admin creates/modifies a SquareDesk Service.
2. Admin creates/modifies a SquareDesk Permission.
3. Admin creates/modifies a SquareDesk Role.
4. Admin creates/modifies a SquareDesk User.

5. Admin adds/removes entitlements from a Service, User or Role.
6. Admin modifies basic Service/Permission/Role information (name/description).

These use cases follow the same sequence of actions:
1. Admin logs into the SquareDesk system.
2. Provider executes zero or more of the use case operations described in 1 through 6 above.
3. Providers logs out of ShareDesk system.

The design has also identified the following additional use cases and corresponding action sequences for a SquareDesk user (please refer to the Implementation Details in the Implementation section for diagrams of these sequences):

Login

Successful login:
1. User submits user name and password
2. Authentication Service validates user name exists and password is correct, and creates and returns authentication token.

Unsuccessful login (user name does not exist):
2a. Authentication Service is unable to find matching user name and throws InvalidUserNameException.

Unsuccessful login (password is not valid):
2b. Authentication Service determines password input does not match correct password and throws InvalidPasswordException.

Logout

Successful logout:
1. User sends logout request
2. Authentication Service validates access token (then invalidates it against further use) and sends user confirmation of successful logout

Unsuccessful logout:
2a. Authentication Service determines access token is invalid (does not exist or has expired) and throws an InvalidAccessTokenException

Restricted method access (assumes user has logged in successfully)

Successful restricted method call:
1. Client attempts to execute a restricted method, passing in its access token to the corresponding API
2. Corresponding API successfully validates access token (is non-null and not empty) and passes access token to Authentication Service.
3. Authentication Service
- Successfully validates access token (it exists, is active and within expiration period)
- Determines the user associated with the access token has permission to execute the corresponding restricted method

- Indicates to the corresponding API/restricted method that the user may execute the method

4. Corresponding API executes the method and indicates success to the client

Unsuccessful restricted method call (API determines access token is invalid):
2a. Corresponding API determines access token is not valid for some reason (is null or is empty) and throws an InvalidAccessTokenException

Unsuccessful restricted method call (Authentication Service determines access token is invalid):
3a. Authentication Service determines access token is invalid (does not exist or is invalid) and throws an InvalidAccessTokenException

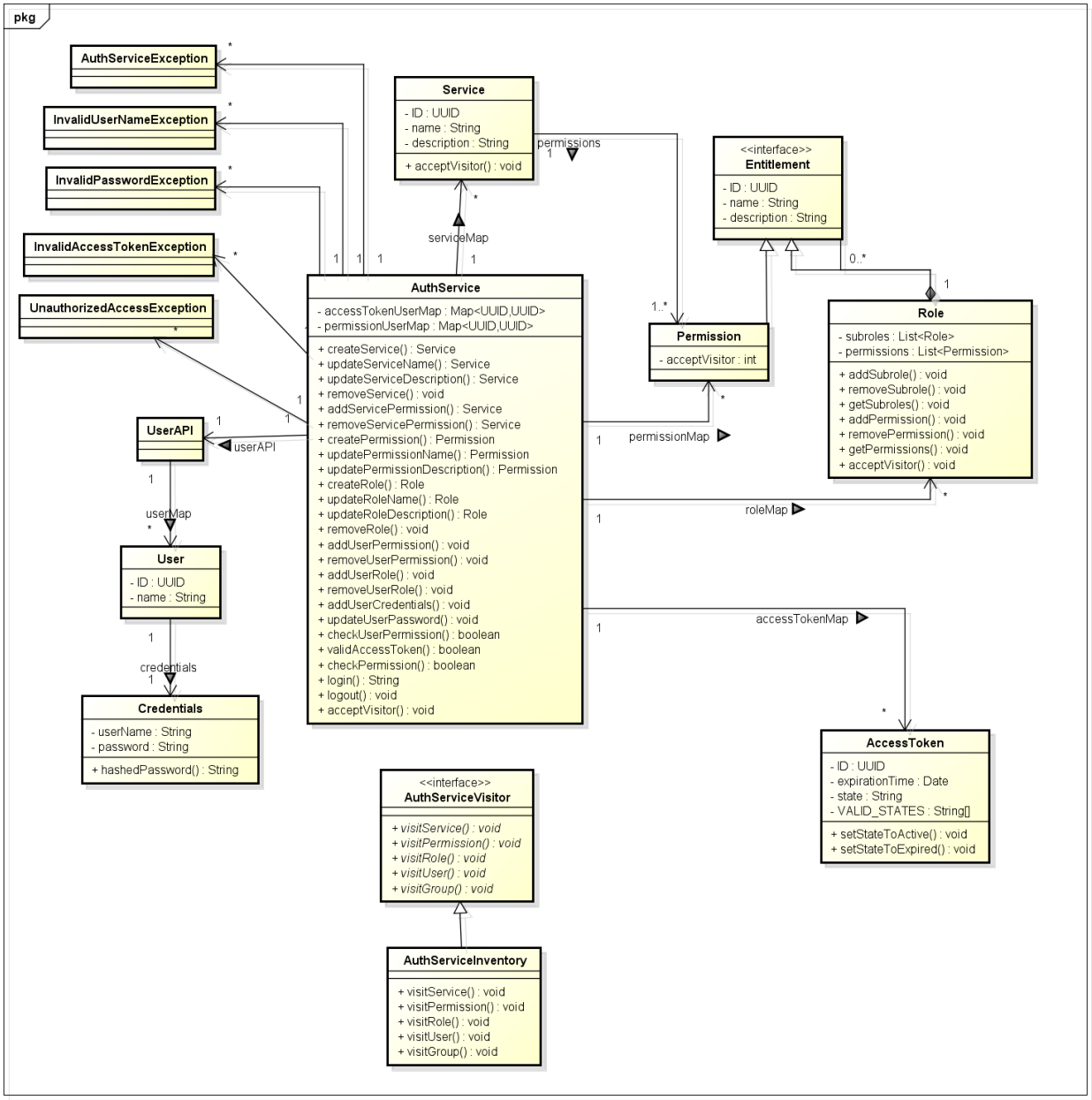
Unsuccessful restricted method call (Authentication Service determines user is not entitled to execute requested operation):
3b. Authentication Service determines user is not entitled to execute the requested operation and throws an UnauthorizedAccessException.

Implementation

(In addition to the class diagram and class dictionary directly below, please refer to the Implementation Details at the end of this section for additional explanations and specifications for implementing the SquareDesk AuthService).

Class Diagram

The following class diagram provides a general overview of the classes used in the SquareDesk AuthService:



Class Dictionary

This section specifies the class dictionary for the AuthService defined within the package cscie97.asn4.squaredesk.authentication.

AuthService

This singleton class maintains maps of the objects required for the SquareDesk authentication system (services, roles and permissions) and defines the functions required for SquareDesk authentication processes (user login, logout and restricted method access). The UserAPI implemented in the previous design stage will continue to store user information (through user maps), though user-related authentication functions will be implemented in AuthService.

Methods

Name	Signature	Description
createService	(accessToken:String, name:String, description:String):Service	Creates the specified SquareDesk service. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException.
updateServiceName	(accessToken:String, serviceID:UUID, name:String):Service	Updates the name of the specified Service object. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if this name is a duplicate).
updateServiceDescription	(accessToken:String, serviceID:UUID, description:String):Service	Updates the description of the specified Service object. Throws InvalidAccessTokenException, UnauthorizedAccessException.
removeService	(accessToken:String, serviceID:UUID):void	Removes the specified Service from the system. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException.
addServicePermission	(accessToken:String, serviceID:UUID, permission:Permission):Service	Adds the given permission to the specified Service object. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException.
removeServicePermission	(accessToken:String, serviceID:UUID, permissionID:UUID):Service	Removes the specified permission from the specified Service object. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException.
createPermission	(accessToken:String, name:String, description:String):Permission	Creates the specified SquareDesk permission. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if permission name already exists).
updatePermissionName	(accessToken:String, permissionID:UUID, name:String):Permission	Updates the name of the specified permission. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if permission name already exists).
updatePermissionDescription	(accessToken:String, permissionID:UUID, description:String):Permission	Updates the description of the specified permission. Throws InvalidAccessTokenException, UnauthorizedAccessException.
createRole	(accessToken:String, name:String, description:String):Role	Creates the specified SquareDesk role. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if role name already exists).
updateRoleName	(accessToken:String, roleID:UUID, name:String):Role	Updates the name of the specified role. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if role name already exists).
updateRoleDescription	(accessToken:String, roleID:UUID, description:String):Role	Updates the description of the specified permission. Throws InvalidAccessTokenException, UnauthorizedAccessException.
removeRole	(accessToken:String, roleID:UUID):void	Removes the specified Role from the system. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if role ID does not exist).

addUserPermission	(accessToken:String, userID:UUID, permission:Permission):void	Adds the given permission to the user with the specified ID. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if user already has this permission)
removeUserPermission	(accessToken:String, userID:UUID, permissionID:UUID):void	Removes the specified permission from the user with the specified ID. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if user does not have this permission)
addUserRole	(accessToken:String, userID:UUID, role:Role):void	Adds the given role to the user with the specified ID. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if user already has this role)
removeUserRole	(accessToken:String, userID:UUID, roleID:UUID):void	Removes the specified role from the user with the specified ID. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if user does not have this role)
addUserCredentials	(accessToken:String, userID:UUID, userName:String, password:String):void	Adds the given credentials to the specified user. Throws InvalidAccessTokenException, UnauthorizedAccessException, AuthServiceException (if the user name already exists).
updateUserPassword	(accessToken:String, userID:UUID, newPassword:String):void	Updates the password for the specified user. Throws InvalidAccessTokenException, UnauthorizedAccessException.
checkUserPermission	(accessToken:String, permission:UUID):boolean	Returns true if the access token is valid AND the user associated with the access token has the specified permission. Calls private validateAccesstoken() and checkPermission() methods below.
validateAccessToken (private method)	(accessToken:String):boolean	Returns true if the given access token exists, is active and has not reached its expiration time.
checkPermission (private method)	(accessToken:String, permission:UUID):boolean	Returns true if the user associated with the access token has the specified permission
login	(userName:String, password:String):String	Performs the necessary checks and operations to log the specified user in, and returns a valid access token if these were successful. Throws InvalidUserNameException, InvalidPasswordException.
logout	(accessToken:String):void	Logs the user associated with the specified accessToken out of the SquareDesk system. Throws InvalidAccessTokenException.
acceptVisitor	(visitor:AuthServiceVisitor):void	Method allowing for execution of the corresponding AuthServiceVisitor operation.
bootstrap	():void	Registers the authentication service itself and its restricted access methods with the authentication service, and also creates a bootstrap super_admin user.

Properties

Property Name	Type	Description
accessTokenUserMap	Map<UUID, UUID>	Access token-to-user map.
permissionUserMap	Map<UUID, List<UUID>>	Map from permissions to List of users who have authority for given permission

Associations

Association Name	Type	Description
serviceMap	Map<UUID, Service>	Set of valid SquareDesk services.
permissionMap	Map<UUID, Permission>	Set of valid SquareDesk service operation permissions.
roleMap	Map<UUID, Role>	Set of valid SquareDesk permission roles.
accessTokenMap	Map<UUID, AccessToken>	Set of active access Tokens.

userAPI	UserAPI	Reference to singleton UserAPI object (implemented in previous stage).

Service

This class is used to describe a given SquareDesk service (e.g. the OfficeProviderServiceAPI or RenterAPI).

Methods

Name	Signature	Description
acceptVisitor	(visitor:AuthServiceVisitor):void	Method allowing for execution of the corresponding AuthServiceVisitor operation.

Properties

Property Name	Type	Description
id	UUID	Unique identifier for given Service object.
name	String	Name of given Service object.
description	String	Description of given Service object.

Associations

Association Name	Type	Description
permissions	Map<UUID, Permission>	Set of permissions associated with the Service object.

Entitlement

Interface defining methods and properties to be implemented for any SquareDesk AuthService entitlement object (Permission or Role). The implementation of Permission and Role objects will be based on the Composite design pattern, with Roles having the ability to contain sub-roles and permissions, and Permissions representing a “leaf node” entitlement.

Properties

Property Name	Type	Description
id	UUID	Unique identifier for given Entitlement object.
name	String	Name of given Entitlement object.
description	String	Description of given Entitlement object.

Role

Implements Entitlement. Describes a given set of permissions and sub-roles for accessing restricted SquareDesk methods. Examples of roles are Provider, Renter and Administrator.

Methods

Name	Signature	Description
addSubrole	(subrole:Role):void	Adds the specified subrole to the given Role’s list of subroles.
removeSubrole	(subroleID:UUID):void	Removes the specified subrole from the given Role’s list of subroles.
getSubroles	():List<Role>	Returns the given Role’s list of subroles.
addPermission	(permission:Permission):void	Adds the specified permission to the given Role’s list of

		permissions.
removePermission	(subroleID:UUID):void	Removes the specified permission from the given Role's list of permissions.
getPermissions	():List<Permission>	Returns the given Role's list of permissions.
acceptVisitor	(visitor:AuthServiceVisitor):void	Method allowing for execution of the corresponding AuthServiceVisitor class operation.

Associations

Association Name	Type	Description
subroles	List<Role>	Set of subroles associated with the given Role object.
permissions	List<Permission>	Set of permissions associated with the given Role object.

Permission

Implements Entitlement. Describes a specific permission for accessing a restricted SquareDesk method. Examples of permissions are createOfficeSpace (OfficeSpaceProviderServiceAPI), createRenter (RenterAPI).

Methods

Name	Signature	Description
acceptVisitor	(visitor:AuthServiceVisitor):void	Method allowing for execution of the corresponding AuthServiceVisitor class operation.

Credentials

Class describing a SquareDesk user's credentials (name and password).

Methods

Name	Signature	Description
hashedPassword	():String	Returns a hashed version of the user's password.

Properties

Property Name	Type	Description
userName	String	User's name.
password	String	User's password.

AccessToken

Class describing a token that allows restricted SquareDesk methods to be accessed.

Methods

Name	Signature	Description
setStateToActive	():void	Sets the state of the given AccessToken object to active.
setStateToExpired	():void	Sets the state of the given AccessToken object to expired.

Properties

Property Name	Type	Description
ID	UUID	Unique identifier for given AccessToken object.
expirationTime	Date	Time at which given AccessToken object will expire.
state	String	State of given AccessToken object.
VALID_STATES	String[]	Permissible state values (ACTIVE or EXPIRED).

AuthServiceVisitor

Interface based on the Visitor design pattern, defining the methods that must be implemented by AuthService visitor classes that must access and process all classes within the service (i.e. Service, Permission, Role, User and Group).

Methods

Name	Signature	Description
visitAuthService	(authService:AuthService):void	Method to access a given AuthService object.
visitService	(service:Service):void	Method to access a given Service object.
visitPermission	(permission:Permission):void	Method to access a given Permission object.
visitRole	(role:Role):void	Method to access a given Role object.
visitUser	(user:User):void	Method to access a given User object.
visitCredential	(credential:Credential):void	Method to access a given Credential object.
visitAccessToken	(accessToken:AccessToken):void	Method to access a given AccessToken object.

AuthServiceInventory

Implements AuthServiceVisitor. Used to create an inventory of all AuthService objects. The class maintains an inventoryDescription StringBuilder() object to which it adds descriptions (in YAML format) as it visits objects, and also an indent StringBuilder() object which maintains proper indentation (this is just a string of tab characters).

Methods

Name	Signature	Description
visitAuthService	(authService:AuthService):void	Stores the AuthService object in the AuthServiceInventory instance, then visits each: • AuthService Service object • AuthService Role object • AuthService User object
visitService	(service:Service):void	Prints a description of the Service object then visits each Permission in the Service
visitPermission	(permission:Permission):void	Prints a description of the Permission object.
visitRole	(role:Role):void	Prints a description of the Role object then visits each: • Subrole object in the Role • Permission object in the Role
visitUser	(user:User):void	Prints a description of the User object then visits each: • User Credential object • User AccessToken object
visitCredential	(credential:Credential):void	Prints a description of the Credential object
visitAccessToken	(accessToken:AccessToken):void	Print a description of the AccessToken object.
increaseIndent	():void	Private method to increase indent.
decreaseIndent	():void	Private method to decrease indent.
resetIndent	():void	Clears class indent StringBuilder object.
getInventoryDescription	():String	Returns inventoryDescription property, in String()

		format.
clearInventoryDescription	():void	Clears inventoryDescription StringBuilder object.

Properties

Property Name	Type	Description
inventoryDescription	StringBuilder	Description of AuthService object inventory, in YAML format.
indent	StringBuilder	String (set of /t tab characters) to maintain proper indentation

AuthServiceVisitorElement (visitable)

Interface defining the method that must be implemented by any AuthService class (i.e. Service, Permission, Role, User, Credential, AccessToken), i.e. a visitable class, that will accept an AuthServiceVisitor object.

Methods

Name	Signature	Description
acceptVisitor	(visitor:AuthServiceVisitor):void	Method to accept an AuthService visitor.

Exceptions
InvalidUserNameException
Indicates a specified user name does not exist.

InvalidPasswordException
Indicates a specified password is not valid.

AuthServiceException
Indicates a general AuthService exception (e.g. if an admin tries to enter a service name which already exists).

InvalidAccessTokenException
Indicates a given AccessToken object is not valid for some reason (doesn't exist or has expired).

UnauthorizedAccessException
Indicates the user associated with a given access token does not have permission to access a given operation.

Implementation Details

AuthService - permissionUserMap
Permissions for accessing restricted SquareDesk methods are generally handled by the permissionUserMap in the AuthService. This map maps permissions (by UUID) to a list of UUIDs of User object who have permission to access the corresponding operation. This map is updated immediately whenever a permission/role is added or removed from a User.

AuthService – accessTokenUserMap

This map allows for easy lookup of the user associated with an access token, and is updated when login() and logout() methods are called.

AuthService – Other Maps

The AuthService also maintains general maps of AuthService objects (Service, Permission, Role, AccessToken) to allow CRUD operations to be carried out for these objects.

Modifications to Existing Code

APIs

- Add checkAccessToken() method to each API to allow for checking that access token is non-null and non-empty.
- Add calls to AuthService check() method to restricted API methods as well as exception handling.
- Add registerWithAuthService() method to register the specific service and its restricted access methods with the authentication service and also create an appropriate role.

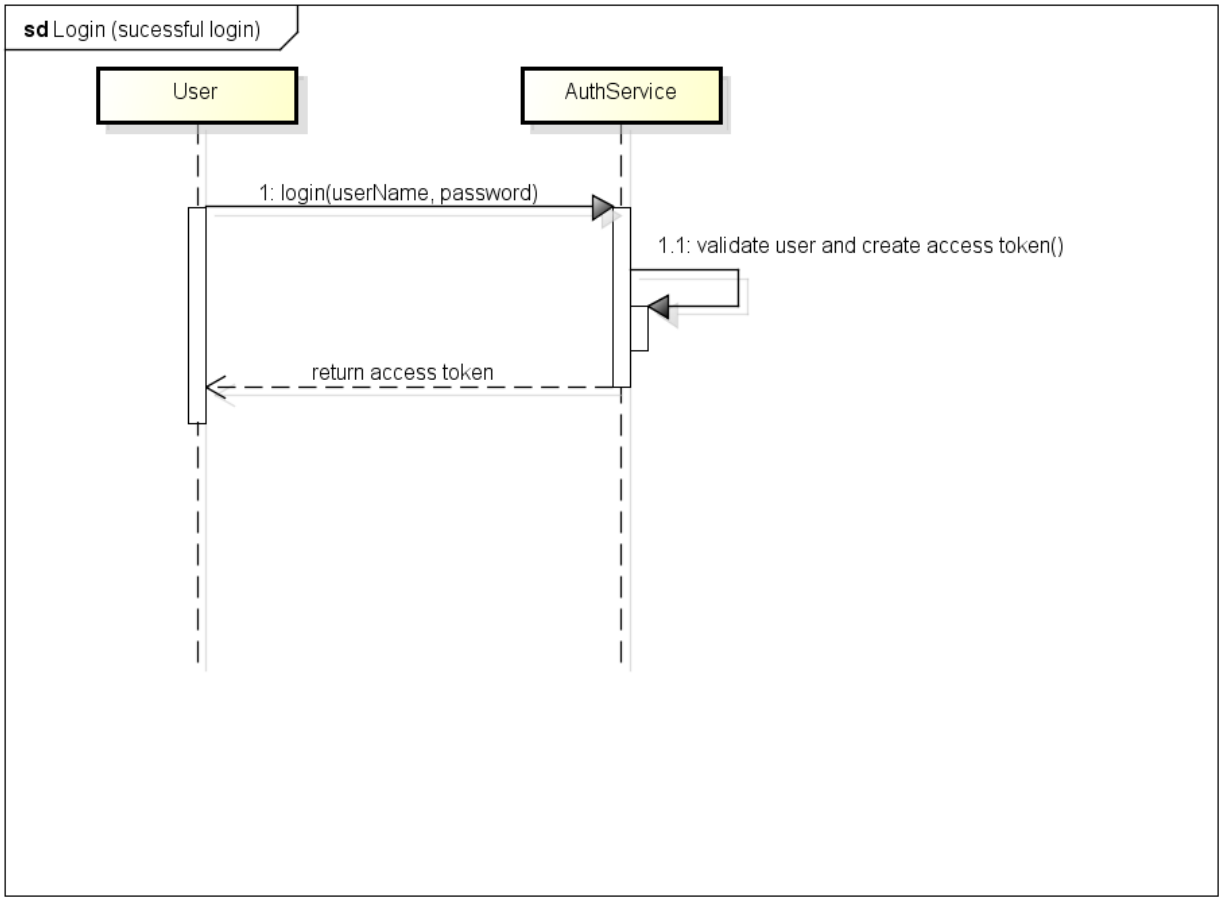
User object (created in assignment 3)

- Add credentials property to User class as well as appropriate getter/setter.
- Add roleMap and permissionMap properties to User class as well as appropriate getters/setters.

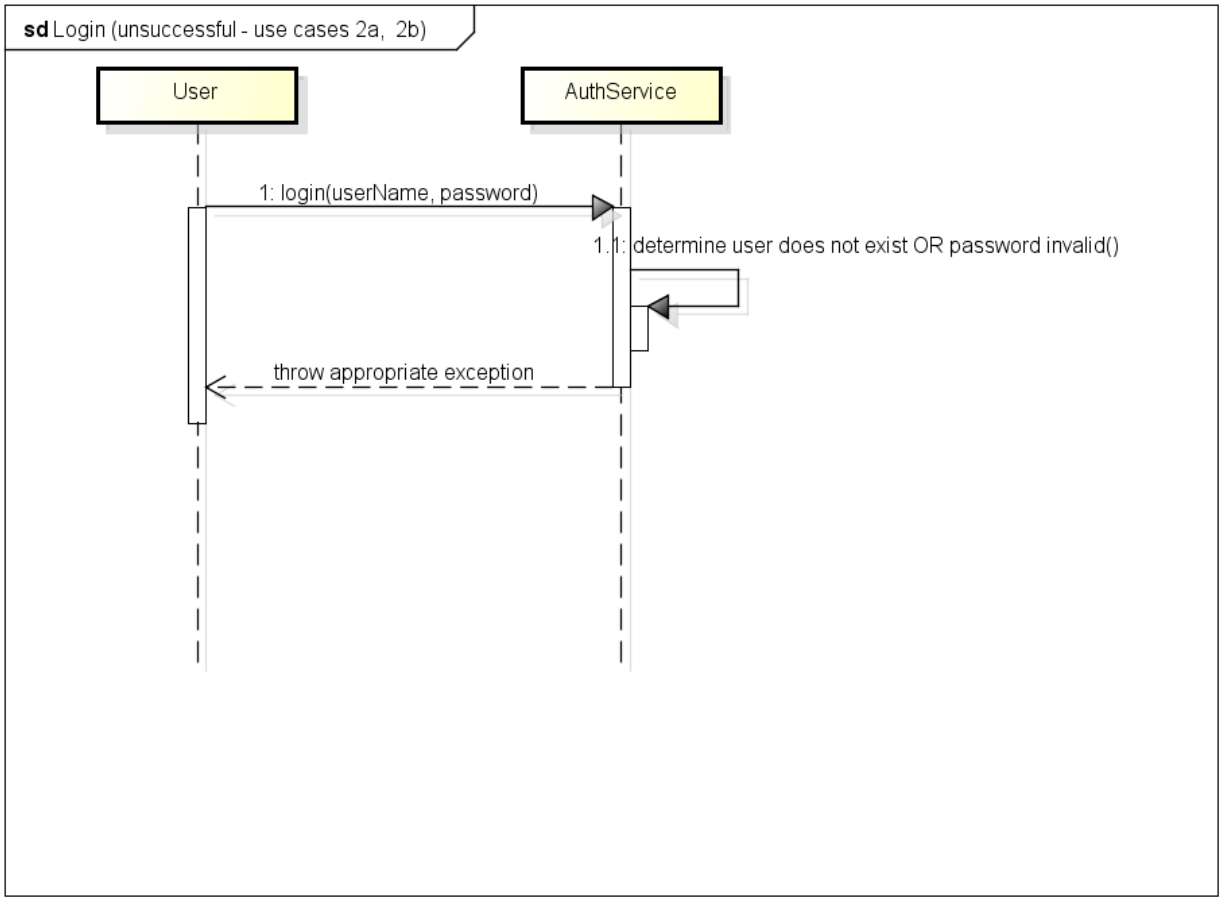
(Note, however, that user methods related to authentication (e.g. modifying passwords) are kept in the AuthService object).

The following sequence diagrams and activity diagrams have also been provided to help illustrate how the separate components of the SquareDesk AuthService should function and interact:

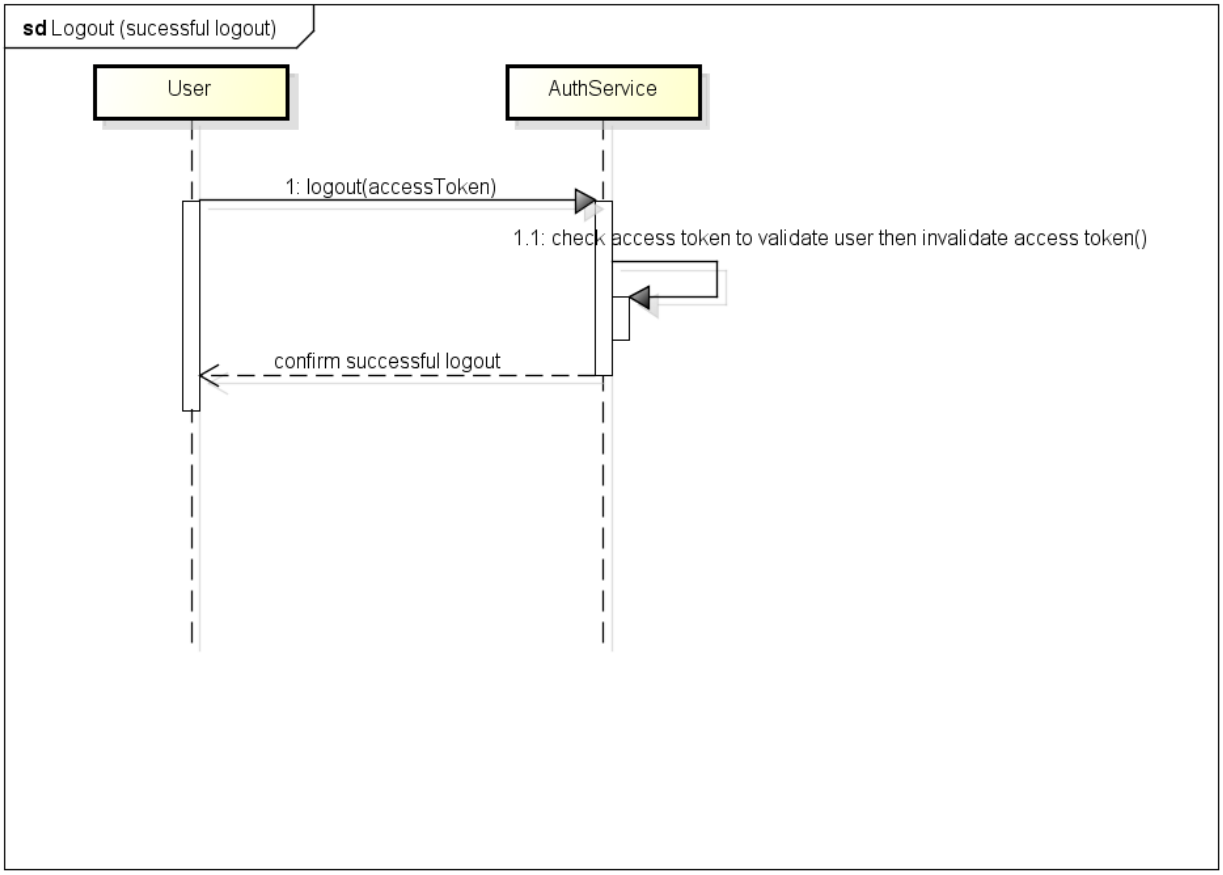
Login sequence – successful login:


sd Login (sucessful login)
User
AuthService
1: login(userName, password)
1.1: validate user and create access token()
return access token

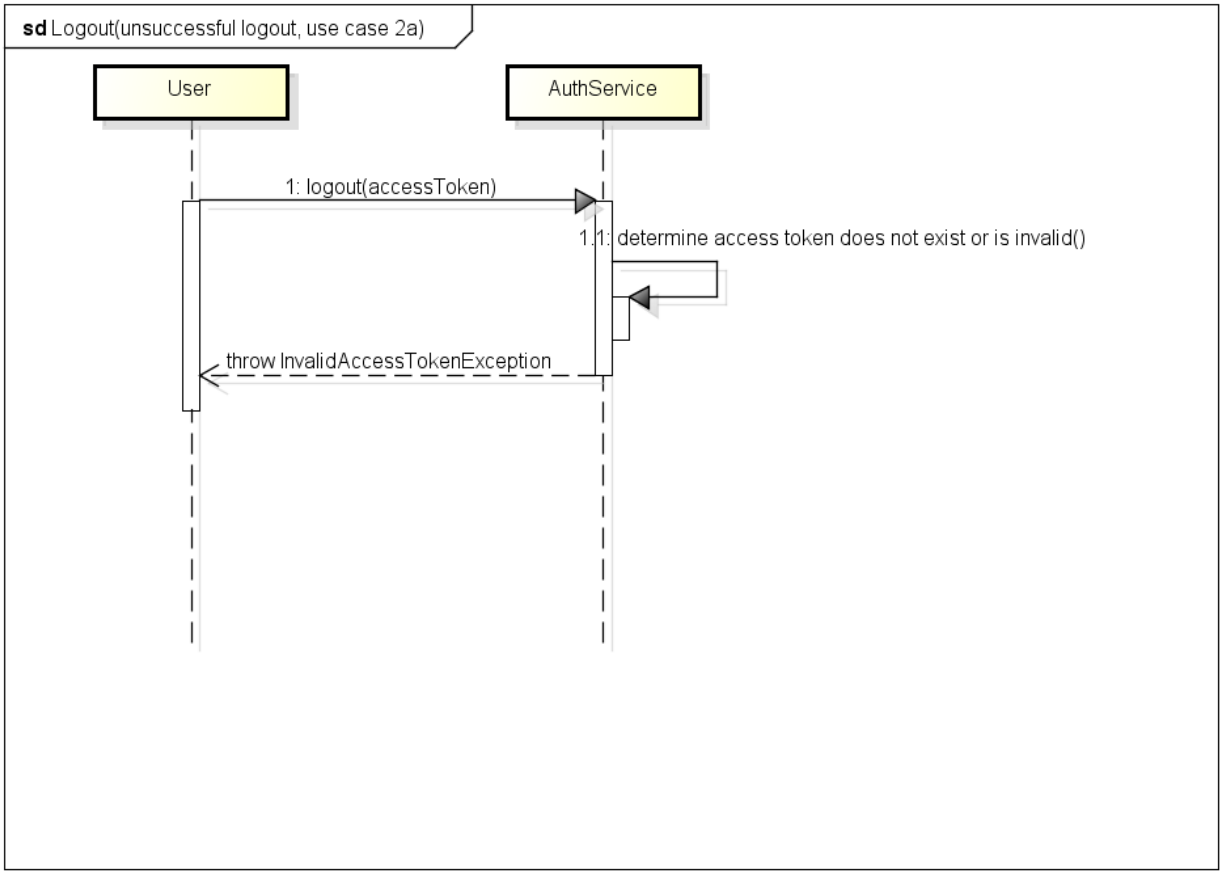
Login sequence – unsuccesful login:



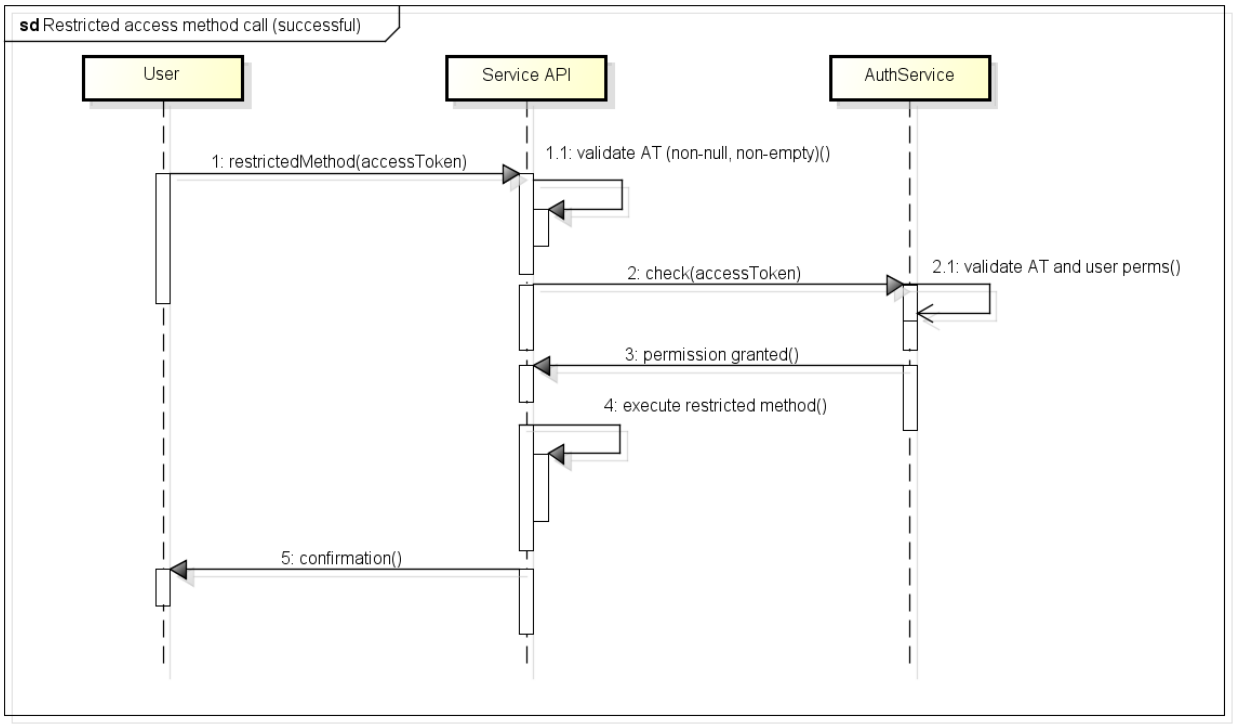
Logout sequence - successful logout:



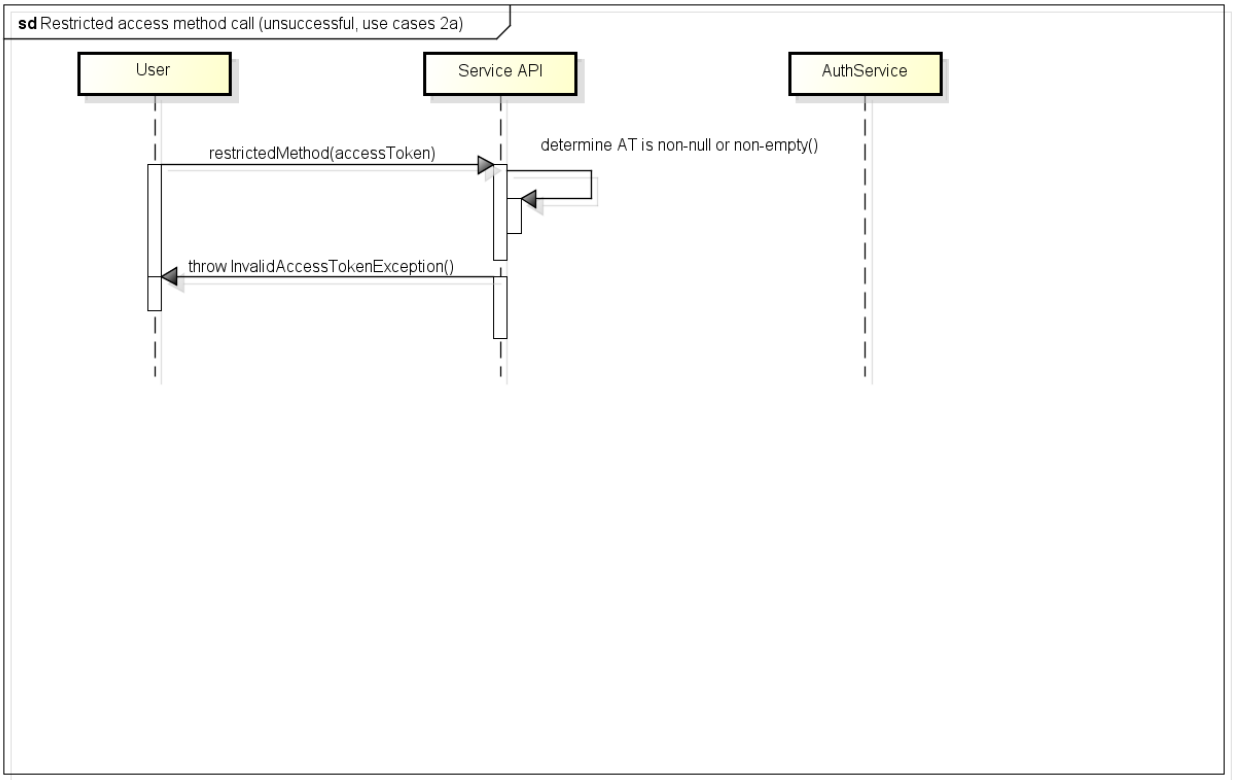
Logout sequence - unsuccessful logout:



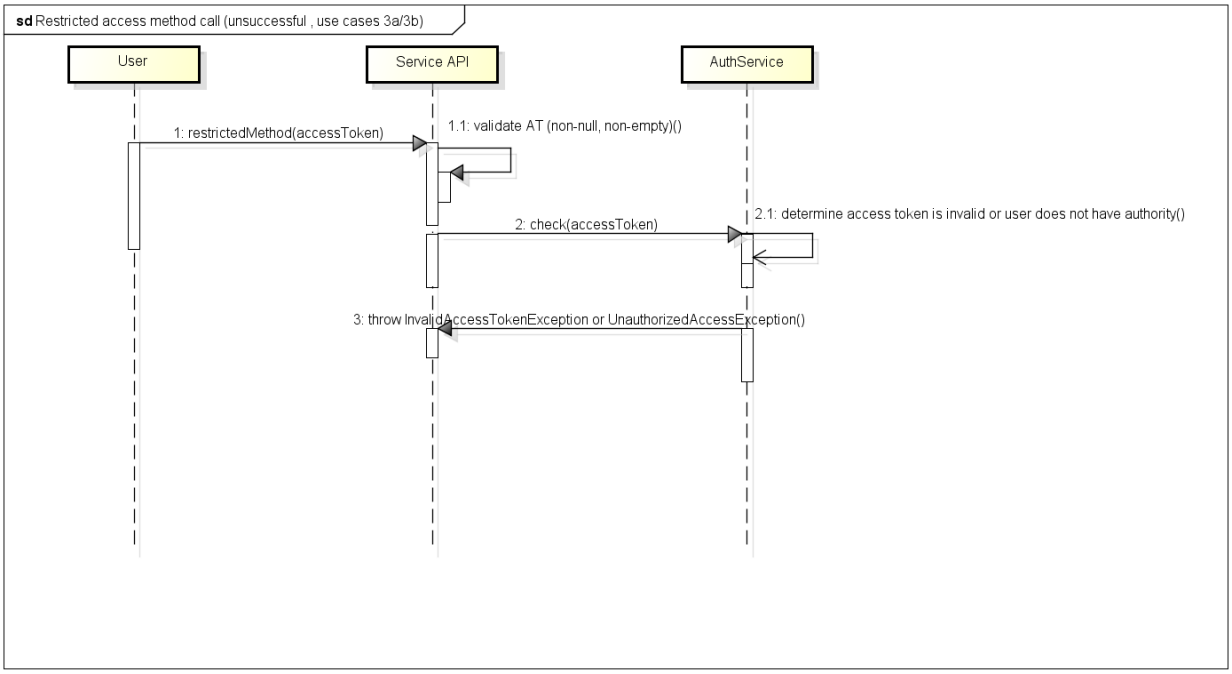
Restricted method access sequence – successful access:



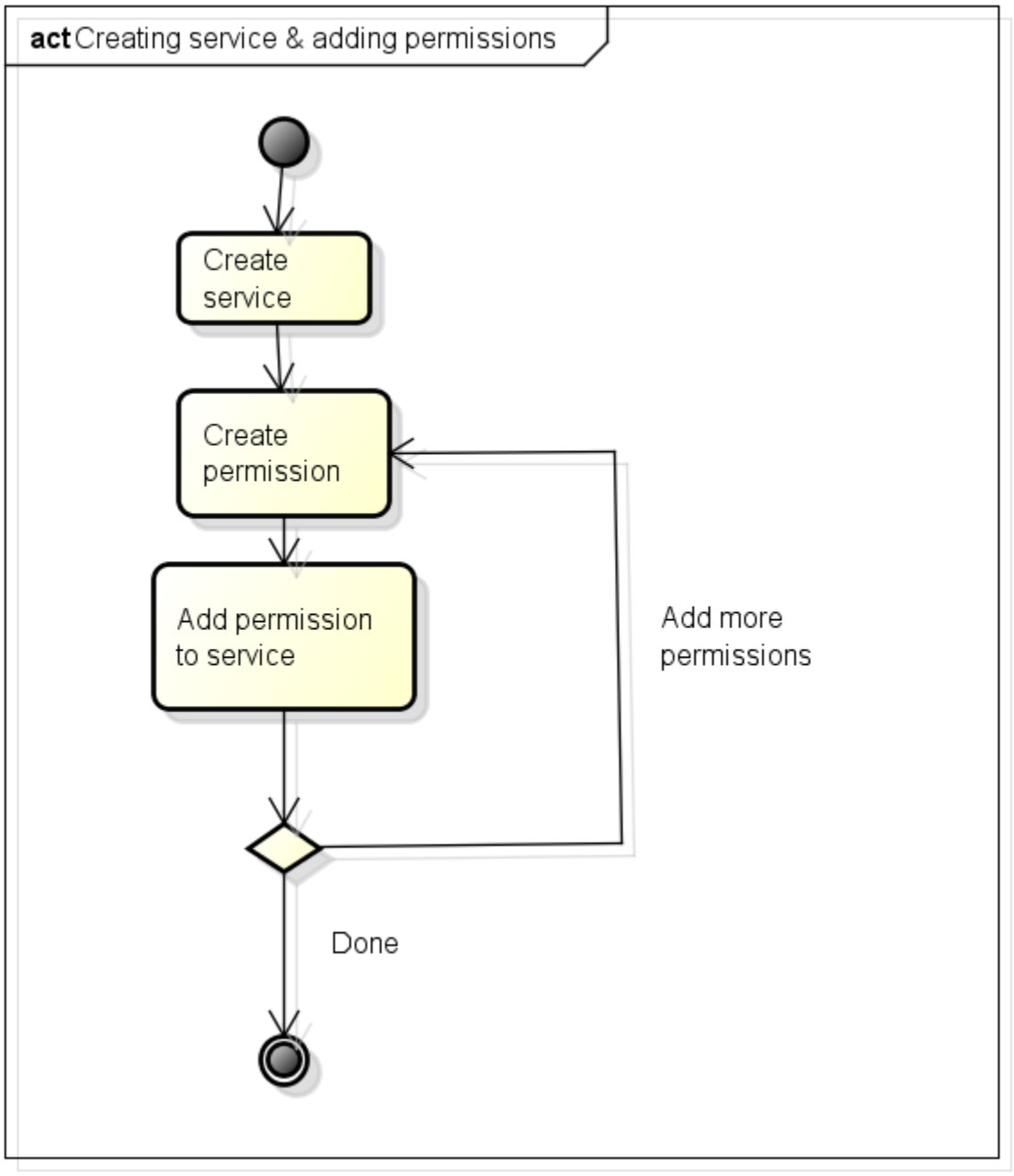
Restricted method access sequence – unsuccessful access (called method determines access token is null or empty):



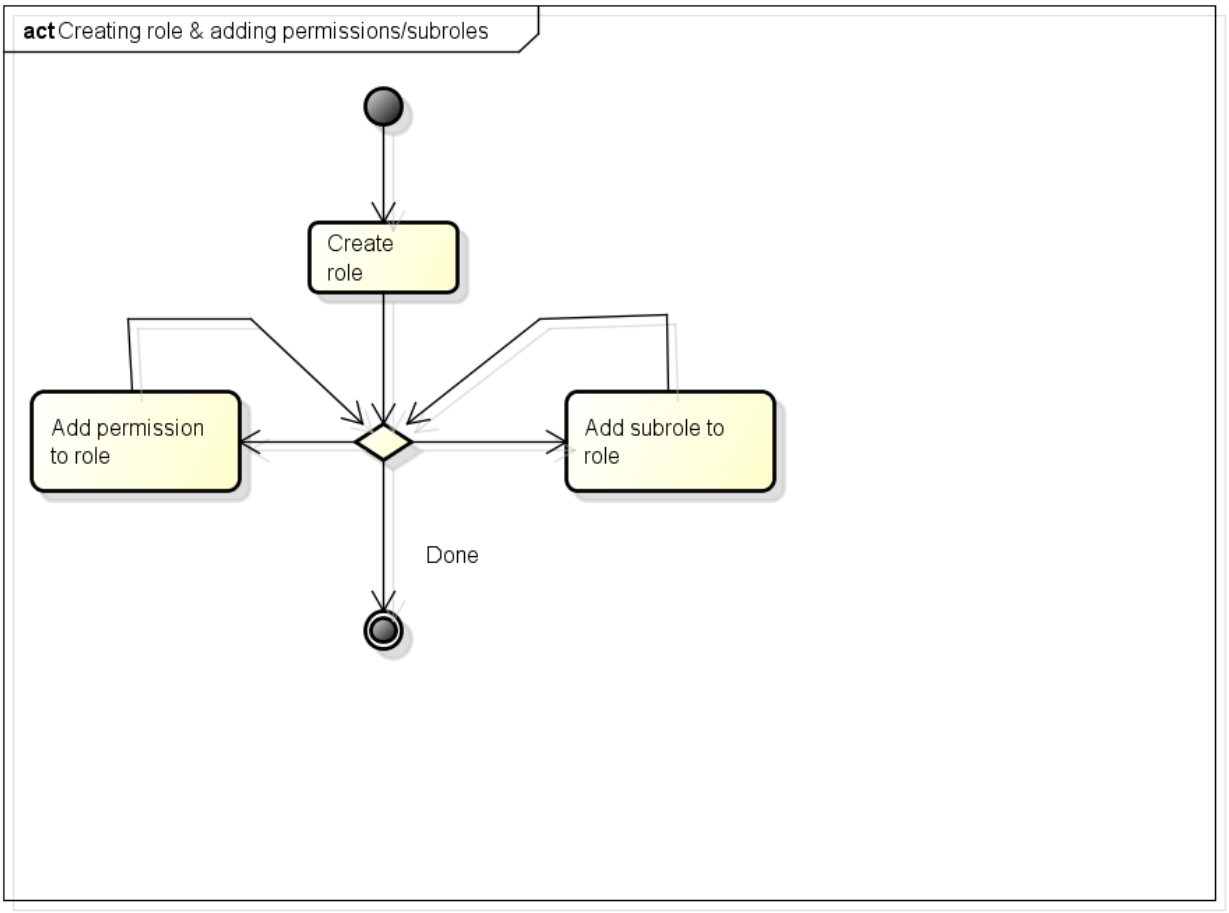
Restricted method access sequence – unsuccessful access (AuthService determines access token does not exist or has expired):



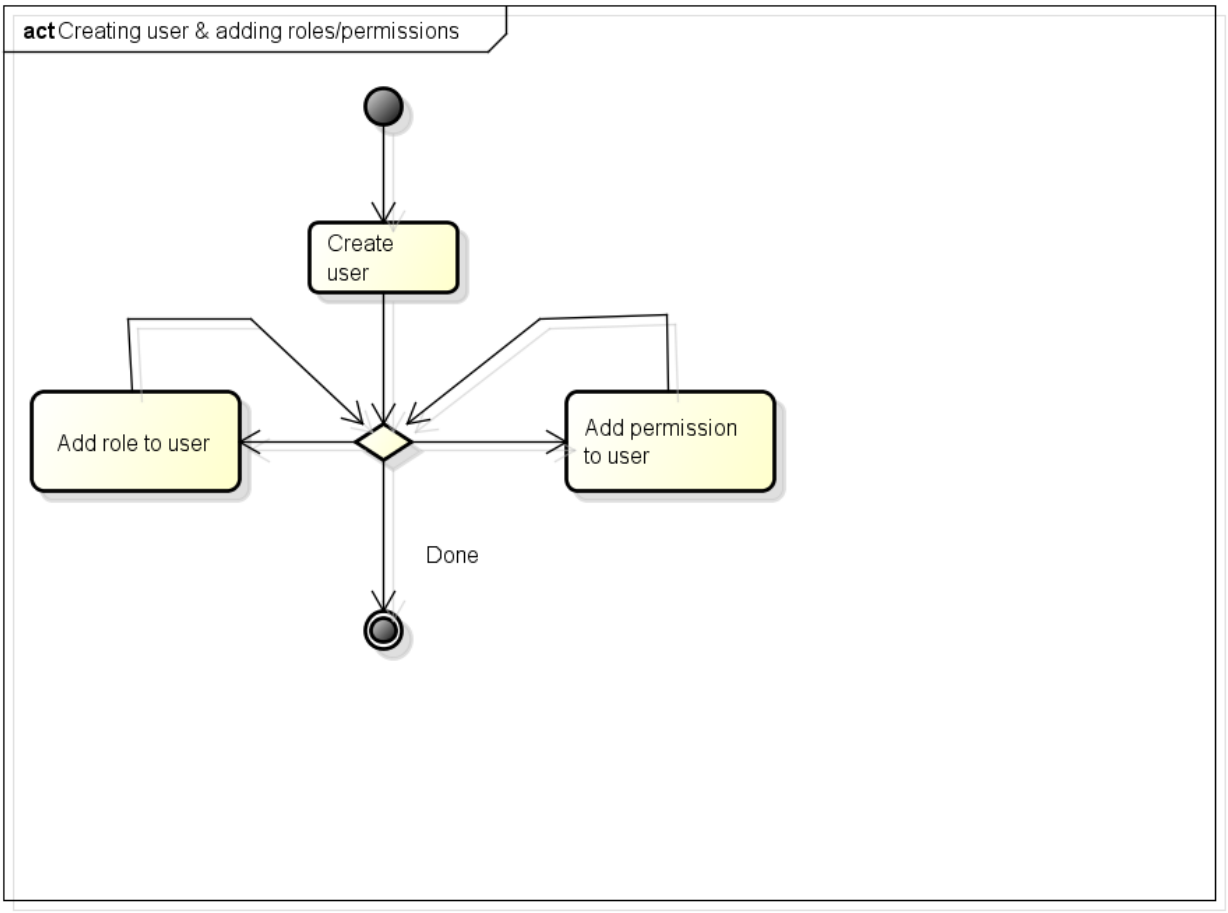
This activity diagram shows the process flow for creating an AuthService Service object:



This activity diagram shows the process flow for creating an AuthService Role object:



Finally, this activity diagram shows the process flow for creating an AuthService User object:



Testing

A test driver that accepts the assignment test data should be created, as well as additional test drivers that fully test all login, logout and restricted method call functionality and exceptions. An Admin role should be created which appropriate permissions and which also includes Provider and Renter roles to test that the role/sub-role/permission entitlement hierarchy functions correctly.

Risks

Per the requirements document, object states will be maintained in memory, so the size of the entire system is limited by the memory allocated to the JVM. Persistence will be a concern in a later project stage, so secondary memory available will also be a factor eventually. Per the requirements document, concurrency concerns for updates are deferred to a later project stage.