# *SquareDesk* Design Document

CSCI E-97 Assignment 4

Authenication Service API

Date: November 20, 2014

Author: Philip Lin

## Introduction

Sites and applications exist that allow people to share their homes, apartments, or other physical spaces that they have with others. This document describes a space sharing application of this sort called SquareDesk. The concept of SquareDesk is to provide a means for individuals to rent out places that they have as office space. The providers can make money from their unused space, and people looking for a place to work have an alternative workplace to traditional office spaces, bookstores, coffee shops or their own homes.

This document describes part of the implementation of the SquareDesk application – the Authentication Service API, which is responsible for controlling access to restricted service methods for renters, providers, and office spaces, as well as the scheduler and search engine.

The Authentication Service API achieves this by acting as central manager of services defined in the application, providing admins and users with defined roles, permissions, and access tokens upon login.

# Overview

When the functions of the Renter Service API (CRUD operations for renters, booking, and search engine functions) or Provider Service API (CRUD operations for providers) are called, they must be authenticated first using an authentication token that is passed into the specific method and validated. In this way, methods from the Renter Service API and Provider Service API must go through the Authentication Service API in order to be used. The authentication token is generated using a username/password login manager that is a part of the Authentication Service API.

Since the Authentication Service API controls access to parts of both the Renter Service API and Provider Service API, it also defines set roles and permissions granted to different levels of users that specify the exact parts of all the Service APIs that can be accessed by the user.

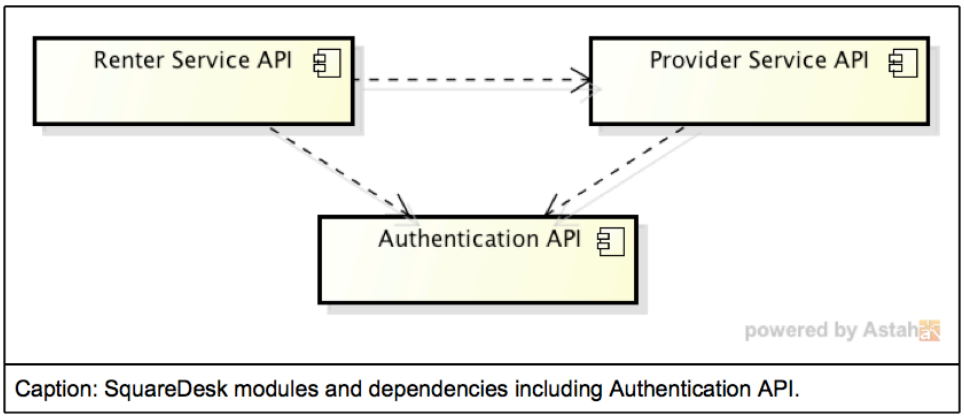


*Figure: SquareDesk structure overview, provided in API Requirements document.*

The Authentication Service benefits the SquareDesk application by providing a layer of security for each function, so that only users or administrators with appropriate access to certain functions will be able to use them, ensuring that the application and it's information are managed properly by the right people.

# Requirements

The Authentication Service API should support the following functions:

1. Create Services
   The Authentication Service API should be able to create the SquareDesk services, which list of the permissions associated with each of the restricted methods in the Provider Service API, Renter Service API, and Authentication Service API. For each restricted method in an API, there will be corresponding permission for that method in the service object for that API. Each service is given a unique ID, name, and description of what it does in the application.

2. Create Roles
   The Authentication Service API should be able to create and define roles for individuals who use the SquareDesk application. These roles are used to group permissions for access to restricted methods together, and roles include SquareDesk administrator, authentication administrator, registered renter, and registered provider. Roles have a unique id, name, and description.

3. Create Permissions
   The Authentication Service API should define permissions to access restricted methods each service of SquareDesk. Permissions dictate whether or not a restricted method is accessible or not to a certain role, and each specific service API method is associated with a specific permission. Each permission object has a unique id, name, and description. A service may have one or more permissions, and a role may be associated with zero or more permissions.

4. Create Users
   The Authentication Service API should be able to create users that represent individual registered users of the SquareDesk application. Users have an id, name, username, and a password

that is hashed for extra security. Users are associated with 0 or more roles or permissions.

5. Support Login Functionality
   The Authentication Service API should support login functionality by accepting credentials and checking them against the database of existing credentials. If a login succeeds, then an authentication token is issued to the login caller that binds the user to the set of permissions to the restricted methods as defined by their role. Authentication tokens have a unique id, an expiration time, and a state indicating whether the token is active or expired. If login fails, then an exception should be thrown.

6. Support Logout Functionality
   The Authentication Service API should support logout functionality, making sure that when someone logs out, the authentication token that they received upon logging in to the system is marked as invalid and isn't used again. Any attempts to use the token again will result in an exception.

7. Invoke Restricted Methods
   The Authentication Service API should allow methods restricted by an authentication token to be invoked when a user or administrator has permission to invoke them. The restricted methods can only be invoked with a proper authentication token that is active, not expired, not null, and not empty. These tokens are provided to certain roles upon login. Exceptions are thrown if any of the security checks fail when trying to invoke a restricted method.

8. Provide Inventory of all Services, Permissions, Roles, Users, and a List of Restricted Methods
   The visitor pattern is used to support traversing the objects of the Authentication Service API in order to provide an inventory of all Services, Permissions, Roles, Users, and Restricted Methods that the API defines.

# Use Cases

SquareDesk Authentication Service supports 2 primary use cases:

1. Administrator

Administrators can login to the SquareDesk system with their username and password. If the login information is correct, the Authentication Service authenticates the admin to perform all the restricted methods in the application by issuing them an authentication token used to run those methods. Every time the admin invokes a restricted method, the authentication token is used to validate the method call in order to proceed. The admin may logout of the SquareDesk system, invalidating the authentication token.

2. Registered User

Registered Users can login to the SquareDesk system with their username and password. If the login information is correct, the Authentication Service authenticates the user to perform operations within the scope of their role and permissions by issuing them an authentication token that is used to run those methods. Every time the user invokes a restricted method, the authentication token is used to validate the method call in order to proceed. The user may logout of the SquareDesk system, invalidating the authentication token.

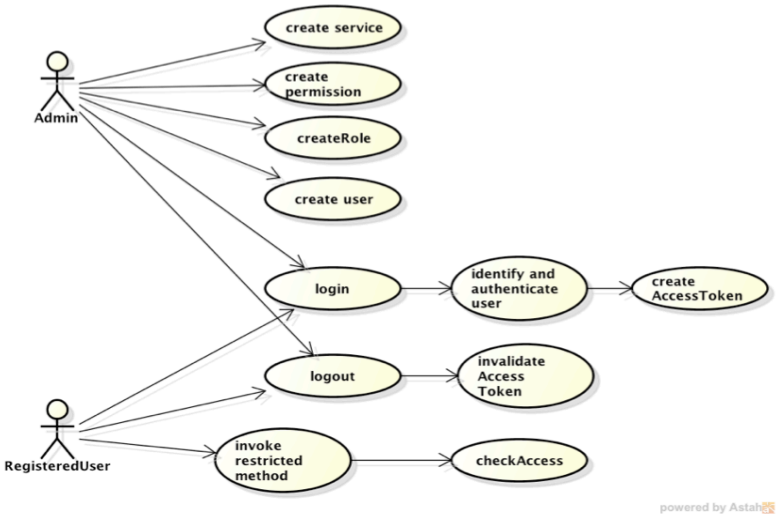


*Figure: Use Case Diagram, provided in API Requirements document.*

# Implementation

## Class Diagram

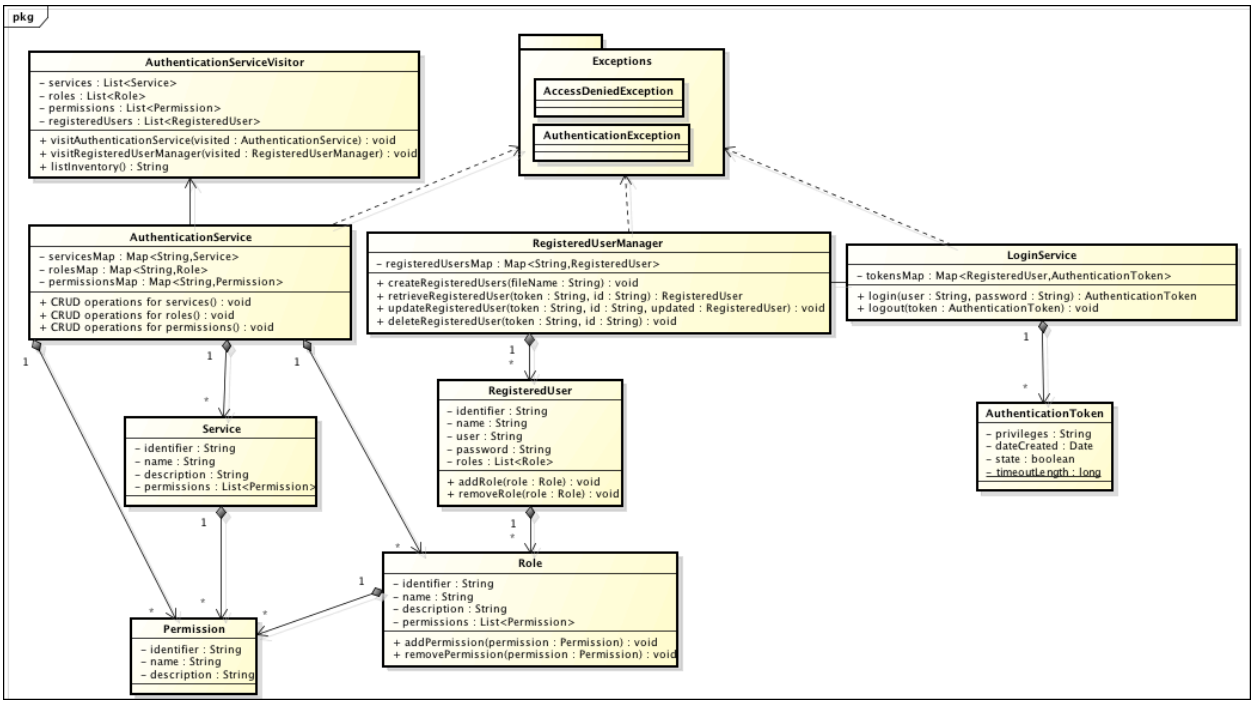


*Figure: Class Diagram.*

## Class Dictionary

The class dictionary of SquareDesk Authentication Service API, defined in package “cscie97.asn4.squaredesk.authenticator.”

### AuthenticationService

The AuthenticationService class is used to define an object that maintains and edits Service, Role, and Permission objects, including create, retrieve and delete operations. The services, roles, and permissions created by the authentication service are used to determine the level of access to restricted methods in the SquareDesk application for registered users.

### *Methods*

| Method Name | Signature | Description |
|---|---|---|
| createServices | (CSVReader reader) : void | Creates service objects from a CSV file using CSVReader class and adds them to the map of services. |
| createRoles | (CSVReader reader) : void | Creates role objects from the CSV file CSV file using CSVReader class and adds them to the map of roles. |
| createPermissions | (CSVReader reader) : void | Creates permission objects from a CSV file CSV file using CSVReader class and adds them to the map of permissions. |
| addPermissionsToRoles | (CSVReader reader) : void | Using the CSV file and the CSVReader class, adds permissions objects to a list maintained by its corresponding role. |
| addPermissionsToServices | (CSVReader reader) : void | Using the CSV file and the CSVReader class, adds permissions objects to a list maintained by its corresponding service. |
| retrieveService | (id : String) : Service | Returns a service object from the map of services. |
| retrieveRole | (id : String) : Role | Returns a role object from the map of roles. |
| retrievePermission | (id : String) : Permission | Returns a permission object from the map of permissions. |
| updateService | (token : AuthenticationToken, id : String, updated : Service) : void | Updates a service object using a new Service DTO. |
| updateRole | (token : AuthenticationToken, id : String, updated : Role) : void | Updates a role object using a new Role DTO. |
| updatePermission | (token : AuthenticationToken, id : String, updated : Permission) : void | Updates a permission object using a new Permission DTO. |
| deleteService | (token : AuthenticationToken, id : String) | Deletes a service object from the map of services. |
| deleteRole | (token : AuthenticationToken, id : String) | Deletes a role object from the map of roles. |

| | | |
|---|---|---|
| deletePermission | (token : AuthenticationToken, identifier : String) | Deletes a permission object from the map of permissions. |

### *Properties*

| Property Name | Type | Description |
|---|---|---|
| servicesMap | Map<String, Service> | Private map of all services maintained by the authentication service. |
| rolesMap | Map<String, Role> | Private map of all roles maintained by the authentication service. |
| permissionsMap | Map<String, Permission> | Private map of all permissions maintained by the authentication service. |

### *Associations*

| Association Name | Type | Description |
|---|---|---|
| reader | CSVReader | CSVReader is a class that can read a CSV file and return information about Services, Roles, Permissions, and RegisteredUsers from the file. |

## AuthenticationServiceVisitor

The AuthenticationServiceVisitor class defines a visitor pattern object that visits objects created by the Authentication Service API and returns an inventory of Services, Roles, RegisteredUsers, and Permissions.

### *Methods*

| Method Name | Signature | Description |
|---|---|---|
| visitAuthenticationService | (visited : AuthenticationService) : | Visits the AuthenticationService object and adds all Services, Roles, and |

| | | |
|---|---|---|
| | void | Permissions found to the respective lists maintained by the visitor. |
| visitRegisteredUserManager | (visited : RegisteredUserManager) : void | Visits the RegisteredUserManager object and adds all RegisteredUsers found to the list maintained by the visitor. |
| listInventory | () : String | Lists the inventory as a String. |

### *Properties*

| Property Name | Type | Description |
|---|---|---|
| services | List<Service> | Private List of all services found by the visitor. |
| roles | List<Role> | Private List of all roles found by the visitor. |
| permissions | List<Permission> | Private List of all permissions found by the visitor. |
| registeredUsers | List<RegisteredUser> | Private List of all registered users found by the visitor. |

## AuthenticationToken

The AuthenticationToken class represents a token given to users after they login. Each token issued to a user has a type that represents the access they have restricted methods according to the user's role. The authentication token has a set time from its creation to its expiration, and this period of time represents a login session in which the registered user can use methods they have permissions for.

### *Properties*

| Property Name | Type | Description |
|---|---|---|
| privileges | String | Private String denoting role privileges of token. |
| dateCreated | Date | Private Date of when the token was created. |
| state | boolean | State marker for the token – true if active, false if inactive. |
| timeoutLength | long | Private number of seconds of the session length. |

## LoginService

The LoginService class represents the service that handles user logins using a username and password. Upon a successful login, the login service will issue an authentication token to a user that allows the user to access certain restricted service methods based on the user's role and permissions.

### *Methods*

| Method Name | Signature | Description |
|---|---|---|
| login | (user : String, password : String) : AuthenticationToken | Checks the user and password against the registered users and issues an authentication token if there is a valid match. Throws an exception if credentials are invalid. |
| logout | (token : AuthenticationToken) : void | Logs a registered user out of their current session, invalidating the state of the user's authentication token. |

### *Properties*

| Property Name | Type | Description |
|---|---|---|
| tokensMap | Map<RegisteredUser, AuthenticationToken> | Private Map maintaining a list of all authentication tokens issued to registered users. |

## Permission

The Permission class represents the permission required to access a specific restricted method in any SquareDesk API. Every restricted method in the Provider Service API, Renter Service API, or Authentication Service API will have a corresponding permission object. If a role has a specific permission object, then the method that the permission object corresponds to will be accessible via

authentication token to users with that role.

### *Properties*

| Property Name | Type | Description |
|---|---|---|
| identifier | String | Private identifier of the role. |
| name | String | Private name of the role. |
| description | String | Private description of the role. |

## RegisteredUser

The RegisteredUser class represents a person who is registered with the SquareDesk application. Each registered user has any number of specific roles with specific permissions to access the methods of the SquareDesk APIs.

### *Properties*

| Property Name | Type | Description |
|---|---|---|
| identifier | String | Private unique non-mutable identifier that is used to identify individual registered users. |
| name | String | Private name of the registered user. |
| user | String | Private user name of the registered user. |
| password | String | Private password of the registered user. The password is hashed for security purposes. |
| roles | List<Role> | Private list of roles associated with the registered user. |

## RegisteredUserManager

The RegisteredUserManager class manages a map of all the registered users, including the CRUD operations of Registered Users. Only one instance of RegisteredUserManager is used in the program

(singleton pattern).

### *Methods*

| Method Name | Signature | Description |
|---|---|---|
| createRegisteredUsers | (reader : CSVReader) : void | Creates a new RegisteredUser object from a CSV file using the CSVReader class. This method does not need to be authenticated since all roles can use this method. |
| addRolesToUsers | (CSVReader reader) : void | Using the CSV file and the CSVReader class, adds role objects to a list maintained by its corresponding user. |
| retrieveRegisteredUser | (id : String) : RegisteredUser | Returns the RegisteredUser object identified by the input identifier. |
| updateRegisteredUser | (token : AuthenticationToken, id : String, updated : RegisteredUser) : void | Updates the RegisteredUser object with a new DTO RegisteredUser. |
| deleteRegisteredUser | (token : AuthenticationToken, id : String) : void | Deletes the RegisteredUser object identified by the input identifier. |
| getInstance | () : RegisteredUserManager | Static method that returns the singleton instance of the RenterManager. |

### *Properties*

| Property Name | Type | Description |
|---|---|---|
| registeredUsersMap | Map<String, Renter> | Private Map maintaining a list of all registered instances of users who are renters. |

### *Associations*

| Association Name | Type | Description |
|---|---|---|
| reader | CSVReader | CSVReader is a class that can read a CSV file and return information about Services, Roles, Permissions, and RegisteredUsers from the file. |

## Role

The Role class is used to group permissions for access to restricted methods together by role. Associated permissions are kept in a list each role has. Roles include SquareDesk administrator, authentication administrator, registered renter, and registered provider.

### *Methods*

| Method Name | Signature | Description |
|---|---|---|
| addPermission | (permission: Permission) : void | Adds a permission object to the role. |
| removePermission | (permission : Permission) : void | Deletes a permission object from the role. |

### *Properties*

| Property Name | Type | Description |
|---|---|---|
| identifier | String | Private identifier of the role. |
| name | String | Private name of the role. |
| description | String | Private description of the role. |
| permissions | List<Permission> | Private list of all the permissions associated with the role. |

## Service

The service class is used to group permissions for access to restricted methods together by the service API those methods are a part of. The three groups are the provider service, renter service, and authentication service. For each restricted method in an API, there will be corresponding permission for that method in the service object for that API.

### *Methods*

| Method | Signature | Description |
|---|---|---|

| Name | | |
|---|---|---|
| addPermission | (permission: Permission) : void | Adds a permission object to the service. |
| removePermission | (permission : Permission) : void | Deletes a permission object from the service. |

### *Properties*

| Property Name | Type | Description |
|---|---|---|
| identifier | String | Private identifier of the service |
| name | String | Private name of the service. |
| description | String | Private description of the service. |
| permissions | List<Permission> | Private list of all the permissions associated with the service. |

## AccessDeniedException

Exception class for handling errors trying accessing restricted methods without appropriate permission, or using invalid or expired access tokens.

### *Methods*

| Method Name | Signature | Description |
|---|---|---|
| getExceptionDetails | () : void | Returns details associated with the exception. |

## AuthenticationException

Exception class for handling errors from using the CRUD methods of the AuthenicationService or RegisteredUserManager classes.

### *Methods*

| Method Name | Signature | Description |
|---|---|---|
| getExceptionDetails | () : void | Returns details associated with the exception. |

# Implementation Details

## CRUD Operations for AuthenticationService and RegisteredUserManager

The create, retrieve, update, and delete operations for instances of Services, Roles, and Permissions are defined in the AuthenticationService class. Instances of Service, Roles, and Permissions are initially created with empty fields, and then updated with relevant information after creation. In order to update the instances of Services, Roles, and Permissions, a Data Transfer Object is used – the original instances are retrieved from the collections that store them, updated, and then stored using the DTO.

The create, retrieve, update, and delete operations for instances of RegisteredUsers are defined in the RegisteredUserMananger class. Instances of RegisteredUsers are initially created with empty fields, and then updated with relevant information after creation. Similar to Services, Roles, and Permissions, the RegisteredUsers are updated using a Data Transfer Object as well.

## Sequence Diagram for Login and Authentication Token Validation

The process of checking whether or not a registered user has access to a certain restricted method is shown in the sequence diagram below. A registered user will login by giving a username and password to the login service, which will check with the registered user manager to see if the username and password is a match for the registered user. If the username and password are valid, then an authentication token is issued to the user based on the user's roles and permissions. This token is used to access restricted methods.

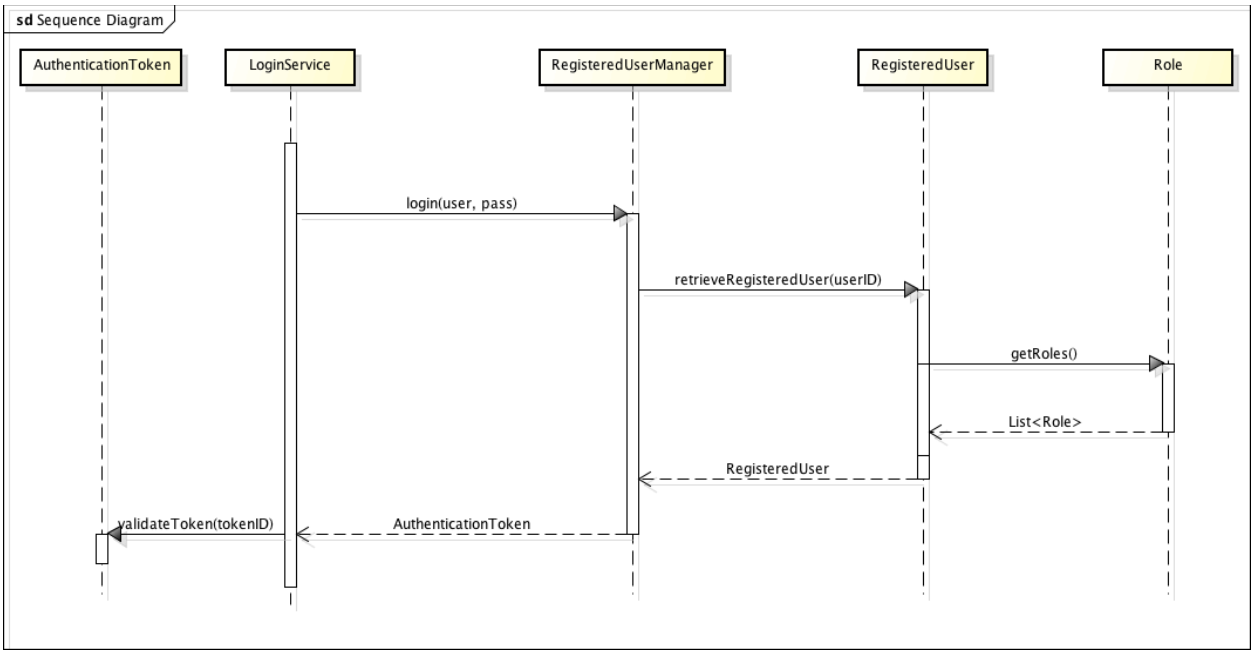


*Figure: Sequence Diagram for Login and Authentication Token Validation.*

## Activity Diagram for how Services, Roles, Permissions, and Registered Users are created

Services, Roles, and Permissions are created using the AuthenticationService class.

Firstly, each individual permission corresponding to a restricted method in the SquareDesk application is created. Next, permissions are grouped into services according to the Service API of the restricted method that the permission is associated with. Then, roles are defined, with each role having access to a specific set of services and permissions.

Registered Users are created using the RegisteredUserManager class. Registered users are then assigned roles based on the access level that they have.

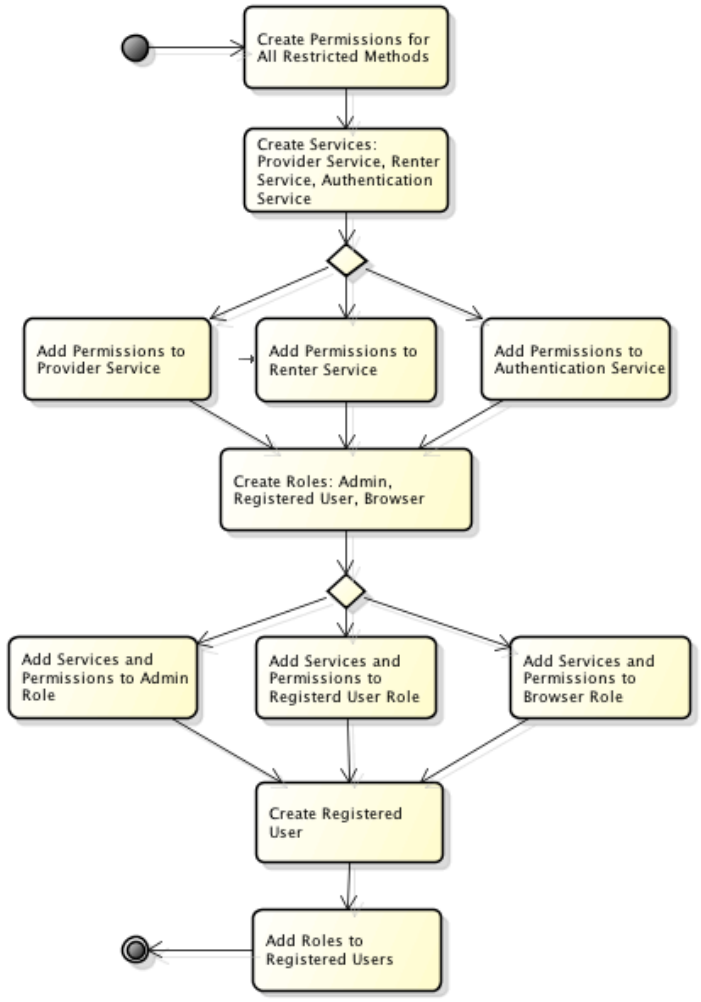


*Figure: Activity Diagram for Creating Services, Roles, Permissions, and Registered Users.*

## Design Patterns Used

Visitor Pattern – This pattern was used to traverse all the objects of the Authentication Service API and return an inventory of all the Services, Roles, Permissions, and Registered Users in the system.

Factory Method Pattern – This pattern was used create all instances of the various classes such as the Service, Role, Permission, and Registered User classes where appropriate.

Singleton Pattern – This pattern was used to return a pointer to an implementation of certain classes in the Authentication Service API.

# Testing

A test driver class called TestDriver, which includes a main method, is used to test the functionality of the API. The test driver should be able to create sample instances of Registered Users, and read a sample file to create sample Services, Roles, and Permissions that can all be retrieved, updated, or deleted.

The test driver should also test access to the restricted methods of the Provider Service API, Renter Service API, and Authentication API, and ensure that services, roles, permissions are all correct for the application. The test driver should test login functionality and that appropriate access tokens are issued to users upon a successful login. Logout functionality should also be tested, with deactivation of authentication tokens being tested.

Exceptions should also be handled for authentication errors, including trying to use invalid or expired authentication tokens and trying to access methods without appropriate permission.

The TestDriver class should be defined within the package “cscie97.asn4.test”.

# Risks

Because of the in-memory implementation, the number objects in the Authentication Service API is limited by the memory allocated to the JVM. Registered Users, Services, Roles, and Permissions must be implemented and updated properly as to avoid making references to objects or fields that don’t exist. Those objects must also be implemented correctly in order to ensure that the correct access is given to appropriate clients of the SquareDesk system. The authentication system should be implemented in a way in which any changes to the system are properly authorized and that the system will still be operable to all users if any exceptions are thrown.