Design Document

Date: Thu Nov 20 22:45:00 EST 2014
Author: Vinod Halaharvi

Introduction

This application is called ShareDesk. This application is akin to AirBnB and other office sharing sites on the Internet. In this app, people can make extra money by renting a part (or whole) of their house as office Space. People who are looking to provide office space can login online and list their facility for rent and people who are looking to rent the office space can also login and search for the office space near by and choose to rent the one they like.

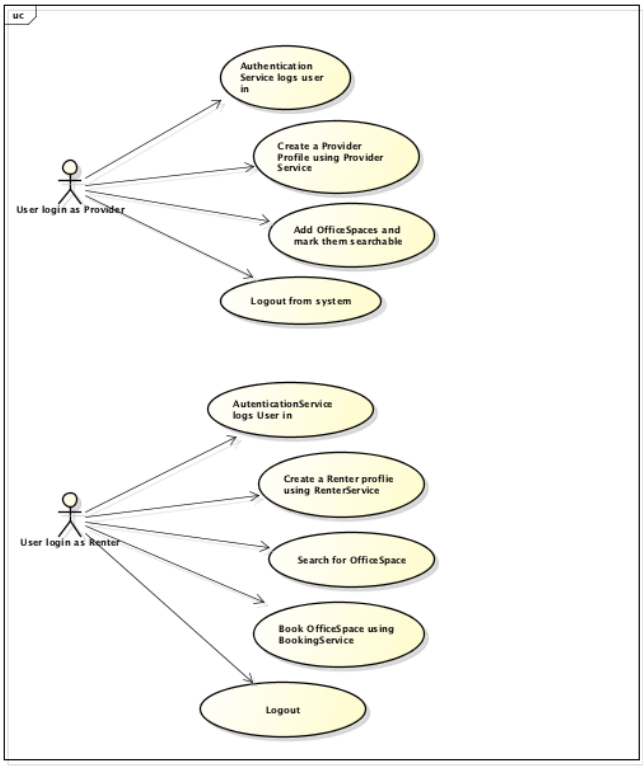
Overview

SquareDesk is a new service that allows people to rent out their home as office space and make additional income by renting out portions of their home as office space. The job of the SquareDesk is to make it easy for people to register and list their homes as office space. Provider simply navigates to the SquareDesk site, registers, provides details about the space they have for rent, and SquareDesk does the rest. Renter goes to SquareDesk web site and search for office space based on various search criteria and selects the officespace he likes the best. He then books this office space. As a commission SquareDesk takes (10%) of what Provider makes. Both Providers and Renters can rate each other

Requirements

Authentication Service API should support creation of users, roles, permissions and services. Authentication Service API should support adding roles to user, and permissions and roles to roles. Should manage user login and logouts by making uses of Authorization tokens. Should support queries from restricted methods by authorizing only the valid tokens

Use Cases



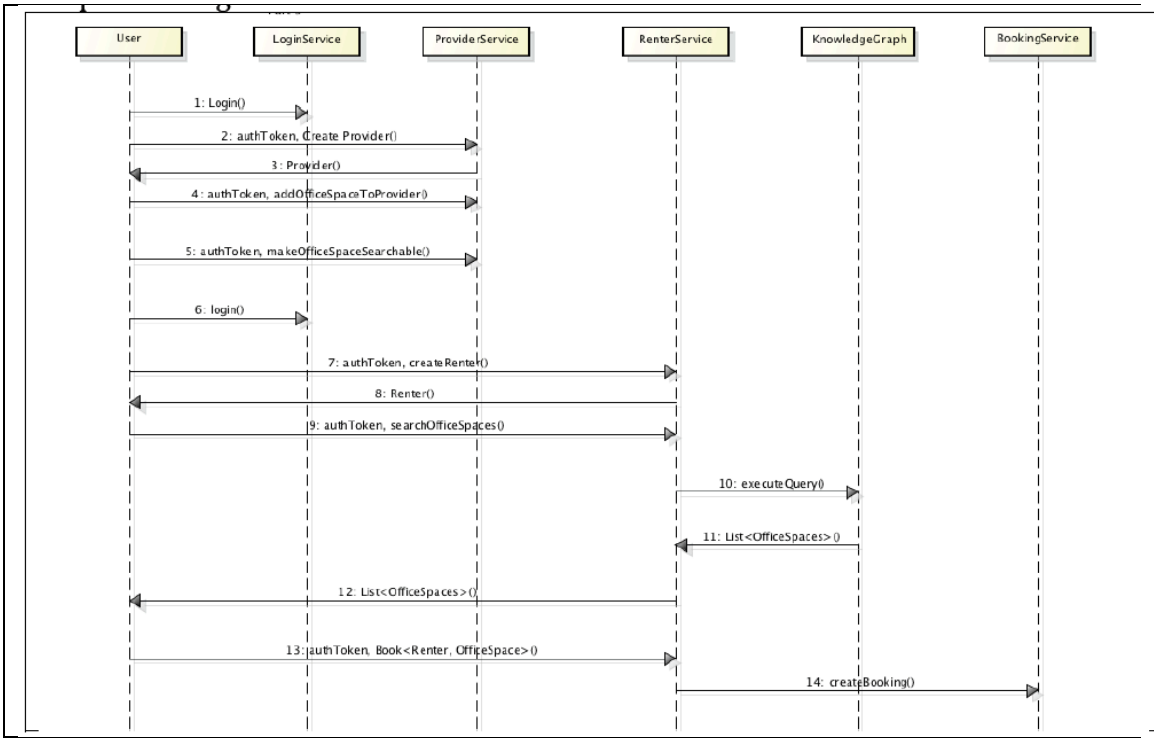
Implementation

Authentication Service is the entry point for a user of the SquareDesk application. User can still view unrestricted methods in the SquareDesk application but cannot call any restricted methods without logging in. User presents login name and password to the Authentication Service. Authentication Service will check if the username is present in the database, if the username is present in the system then it checks if the given password matches the user's password. If both the user and password for the user matches then Authentication Service logs the user into the system. The login() method in the AuthenticationService class performs the user login. In order to login a user, the Authentication Service creates an Authentication token and binds that token to the user object. Each Authentication token has an expiration date and the state of the token. The state of the token represents if the user is still logged in or not, and the expiration date tells when this token is due to expire. Every time the user has to make a call to the restricted methods, user has to present Authentication token. The requested methods then present this token to AuthenticationService checking for the validity of the token. For a token to be valid, the expiration and state of the token has to be valid and the user should have enough access to perform the requested action. hasAccess() method in the AuthenticationService class is the main method that performs the Authentication. This method first makes a check if the token is a valid token (not a null value) and checks the expiration time to make sure the token has not elapsed and then checks if the state of the token is active. On large implementation it's probably more efficient to separate out the authentication and authorization tasks in separate classes. But for SquareDesk application this is performed in single place - within Authentication Service class. In order to authorize, the hasAccess() method first gets user related to the token and iterates over all the roles that user possesses and checks if any of those roles have the permission id that is relevant to the request that

was made by the user. If the permission id is found then the user has the access to perform the requested action and a boolean true is returned to the calling method which then performs that action. If the permissionId is not found in the role list of the user then a boolean false is returned to the calling method which then throws out an AccessException stating the reason and other information of the failure.

So as long as the user is logged in and present a valid token to each restricted call it makes, the user should expect that call to go through successfully. Once the user is done performing the calls, it then chooses to logout of the system by calling AuthenticationService's logout() method. The logout() method mark's the state of user's token as "expired", and the user cannot make anymore calls before logging in again.

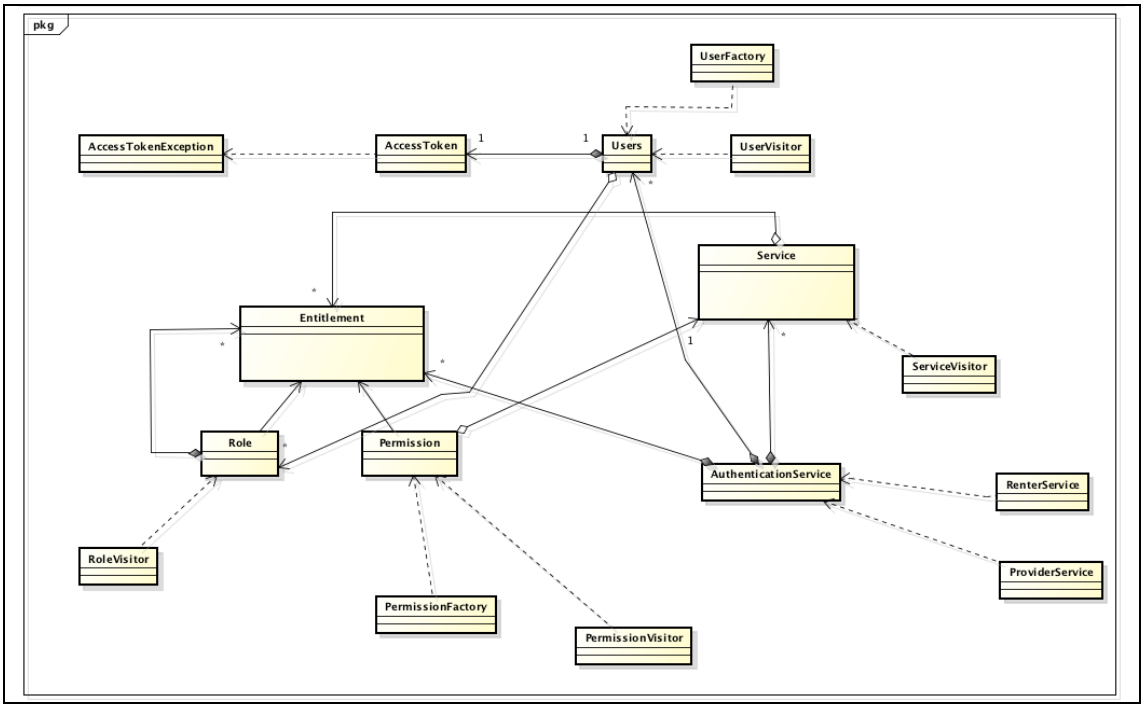
Sequence Diagram



Activity Diagram

Please check the activityDiagram.pdf in the assignment folder.

Class Diagram



Class Dictionary

AccessToken

Property Name	Type	Description
accessTokenId	String	Authorization token that's present to user upon login. User then uses this token to make restricted method calls. All restricted method calls in turn request the AuthenticationService before performing the request action.
expirationTime	Date	Expiration data of the Authorization token
state	String	State of Authorization token. "active" or "expired"

AuthenticationDataImporter

Method Name	Signature	Description
defineServices	(void) : (void)	public method to define services in the system. Services are "ProviderService",

		"RenterService", "AuthenticationService"
definePermissions	(void) : (void)	public method to define Permission in the system. Permission is specific to a method and a service.
defineRoles	(void) : (void)	public method to define roles. Roles are composites for Permission. Both Role and Permission are abstracted using Entitlement class
defineUsers	(void) : (void)	public method to define users. Users must first login to perform restricted method calls

AuthenticationService

Method Name	Signature	Description
createUser	():void	public method to create an user
isValidUser	(String , String):boolean	public method to check if the user is valid
login	(User):AccessToken	public method to login the user. This method will set the authToken of the user to a valid time and marks its state as active
isAccessTokenTimedOut	(AccessToken):boolean	public method to check if the token is timeout?
logout	(User):void	public method to logout the user. This method will mark the User's token invalid
addEntitlementToRole	(Role , Entitlement):void	public method to add Entitlement (A role or permission) to a role. Roles are composites for roles and permissions
addRoleToUser	(User , Role):void	public method to add a role to a specific user
getPermissionById	(String , String):Permission	public method to get the permission object using the permission id
hasAccess	(String , String ,:boolean	public method to check if the user has access. Individual methods call

		use this method to make a Authentication check call to the AuthenticationService class to check if the user is allowed to make this request.
addPermission	(Permission):Permission	public method to add Permission to the system
addUser	(User):User	public method to add an User to the system
addService	(Service):Service	public method to add a Service to the system
createUser	(String , String , String):User	public method to create an user
deleteUser	(String , User):void	public method to delete an User
getRoleListForUser	(User):Collection<Role>	public method to get all the roles that belongs to an User
doesUserHasPermissions	(User , Permission):boolean	private method to check if this user has request permissions
hasPermission	(Role , String):boolean	private method to check if the role has requested permissions
createToken	(User):AccessToken	public method to create a Token
getUserByName	(String , String) throws UserNotFoundException:User	public method to get user object using login name of the user
getUserByAuthToken	(String):User	public method to get user object using authorization token of the user

Property Name	**Type**	**Description**
users	HashSet<User>	private field to store user associations
services	HashSet<Service>	private field to store service associations
entitlements	HashSet<Entitlement>	private field to store entitlements

Entitlement

Property Name	**Type**	**Description**
entitlementId	String	private filed to store

		EntitlementId. An entitlementId could be a roleId or a permissionId

Permission

Property Name	Type	Description
serviceId	String	private field to store serviceId of the service, services are renter service , provider service, authentication service
permissionName	String	private field to store name of the permission
permissionDescription	String	private filed to store the description of the permission

PermissionFactory

Method Name	Signature	Description
createPermission	(String, String, String, String) : Permission	public method to create a permission

PermissionVisitor

Method Name	Signature	Description
visitPermissionList	() : void	public method to visit list of permission
beforeVisitPermission	(Permission) : void	public method to be overridden by the client that performs before visit operations
afterVisitPermission	(Permission) : void	public method to be overridden by the client that performs after visit operations
visitPermissionList	(Permission) : void	public method to be overridden by the client that performs during visit operations

Role

Method Name	Signature	Description

addEntitlementToList	(Entitlement) : Entitlement	public method to add entitlement to entitlement list
removeEntitlementFromList	(Entitlement) : void	public method to remove entitlement from entitlement list
getEntitlements:	() : HashSet<Entitlement>	public method to get the entitlements
getRoles	() : HashSet<Permission>	public method to get the roles
getPermissions	() : HashSet < Permission>	public method to get the permissions

Property Name	**Type**	**Description**
entitlements	Hash<Entitlement>	private field to hold entitlement associations
roleName	String	private field to hold role name
roleDescription	String	private field to hold role description

RoleFactory

Method Name	**Signature**	**Description**
createRole	(String, String, String) : Role	public method to create Role object

RoleVisitor

Method Name	**Signature**	**Description**
visitRoleList	() : void	public method to visit list of permission
beforeVisitRole	(Role) : void	public method to be overridden by the client that performs before visit operations
afterVisitRole	(Role) : void	public method to be overridden by the client that performs after visit operations
visitRoleList	(Role) : void	public method to be overridden by the client that performs during visit operations

ServiceVisitor

Method Name	**Signature**	**Description**

visitServceList	() : void	public method to visit list of permission
beforeVisitServce	(Service) : void	public method to be overridden by the client that performs before visit operations
afterVisitServce	(Service) : void	public method to be overridden by the client that performs after visit operations
visitServceList	(Service) : void	public method to be overridden by the client that performs during visit operations

UserVisitor

Method Name	Signature	Description
visitUserList	() : void	public method to visit list of permission
beforeVisitUser	(User) : void	public method to be overridden by the client that performs before visit operations
afterVisitUser	(User) : void	public method to be overridden by the client that performs after visit operations
visitUserList	(User) : void	public method to be overridden by the client that performs during visit operations

Service

Property Name	Type	Description
serviceId	String	private field to store unique identifier of service
serviceName	String	private field to store name of the service
serviceDescription	String	private field to store description of the service

ServiceFactory

Method Name	Signature	Description
createService	(String, String, String) : Service	public method to create a service

User

Method Name	Signature	Description
addRoleToList	(Role) : void	pubic method to add a role to the user's Role list
removeRoleFromList	(Role) : void	pubilc method to remove a role from the user's Role list

Property Name	Type	Description
userId	String	private field to store user's unique identifier
userDescription	String	private field to store description for the user
loginName	String	private field to store login name of the user.
password	String	private field to store the hash password for the user
authToken	String	private field to store authorization token for the user
roles	HashSet<Role>	private filed to store the list of roles for the user

UserFactory

Method Name	Signature	Description
createUser	(String, String, String) : User	public method to create the user

Risks:
1) No DTO pattern has been applied yet, so RenterService, ProviderService and BookingService will all return the actual objects to the client. The client can modify these objects, which will break the encapsulation. The last sprint should address this concern.

2) KnowlegeGraph holds all combinations of associations in memory and has order polynomial memory requirement. Testing: Testing has been made more modular in this sprint. There is a TestBaseDriver class, which is the Base class for other TestDriver classes. AuthenticationService, RenterService, ProviderService, BookingService, Renter, Provider all have their separate TestDriver classes. There is a single 'TestDriver.java' file where the 'main' function is defined that calls the other test.

REFERENCE and CREDIT

1) SnakeYaml is used for parsing renter.yaml and provider.yaml files.
2) Eclipse software features and plugins like JAutodoc are used for code implementation and documentation

---