Store Authentication Service Design Document

Date: 11/4/21
Author: Haley Huang
Reviewer(s): Burak Ufuktepe, Fatma Ekim

Table of Contents

Introduction

Automation and AI are quickly replacing the need for a human workforce. Homes are managed by virtual home assistants. Restaurants are self-serving. Cars are self-driving. Stores have had self-checkout for years, but technology is improving. Now, stores can be configured to be fully automatic without the need for registers or salesclerks using the Store 24x7 Service.

This design specifies the implementation of Store Authentication Service. The Store Authentication Service creates and validates authentication tokens that allow users of the 24x7 Store system to access resources in the system. Access is controlled by Roles and Resource Roles assigned to Users and Permissions assigned to Roles.

Overview

The Store Authentication Service validates users of the Store Model Service to produce a secure experience for the Store 24x7 customers. Administrators must first login before executing any commands using the Store Model Service API. Store managers, customers, and guests must use a valid entitlement in order to enter the store and engage with devices.

Requirements

This section provides a summary of the requirements for the Store Authentication Service. The Authentication Service uses the Visitor Pattern to support traversing User, Resource, Access, Role, and Permission objects and checking for access. The Service also uses the Singleton Pattern to return a pointer to an interface of the Authentication Service Model API for other 24x7 Modules to use. Finally, the Service uses the Composite Pattern to manage the whole part relation of Roles, Resource Roles, and Permissions.

Authentication Service

1. The Authentication Service extends the Authentication Service Model interface which functions as the Authentication Service API, implementing the Singleton Pattern to ensure only one Authentication Service Object is created.
2. Should be able to create an AuthToken for a User that has provided valid Credentials.
3. Should be able to check the access of a User provided a valid AuthToken and has the necessary Permissions required to access the Resource.
4. Should be able to login and logout valid Users. Login may happen through a Username/Password, Faceprint, or Voiceprint.
5. Should be able to create Entitlements (Permissions, Roles, Resources, and ResourceRoles), Users, and Credentials.

Entitlements

1. Entitlements implement the Composite Pattern to maintain Roles, Resource Roles, Resources, and Permissions.
2. Roles and Resource Roles are composites of Entitlements. Roles can be added to Resource Roles, but Resource Roles should not be added to Roles. Permissions and Resources can be added to either Roles or Resource Roles, although it is unlikely that Resources will need to be associated with Roles.
3. Permissions are leaves and may not have other Entitlements added to or removed from them.

Visitor

1. The Visitor interface implements the Visitor Pattern.

2. Visitors iterate through Users, Roles, Resource Roles, Resources, and Permissions.
3. Visitors are used to determine access of Users.

AuthToken

1. AuthTokens prove the validity of a User's access.
2. AuthTokens are flagged as either active or inactive and are given an expiration date.
3. AuthTokens are created by the AuthenticationServiceModel and assigned to a User.
4. An AuthToken is active upon login and inactivated upon logout. However, they are only valid if the current date is before the expiration date.

Users

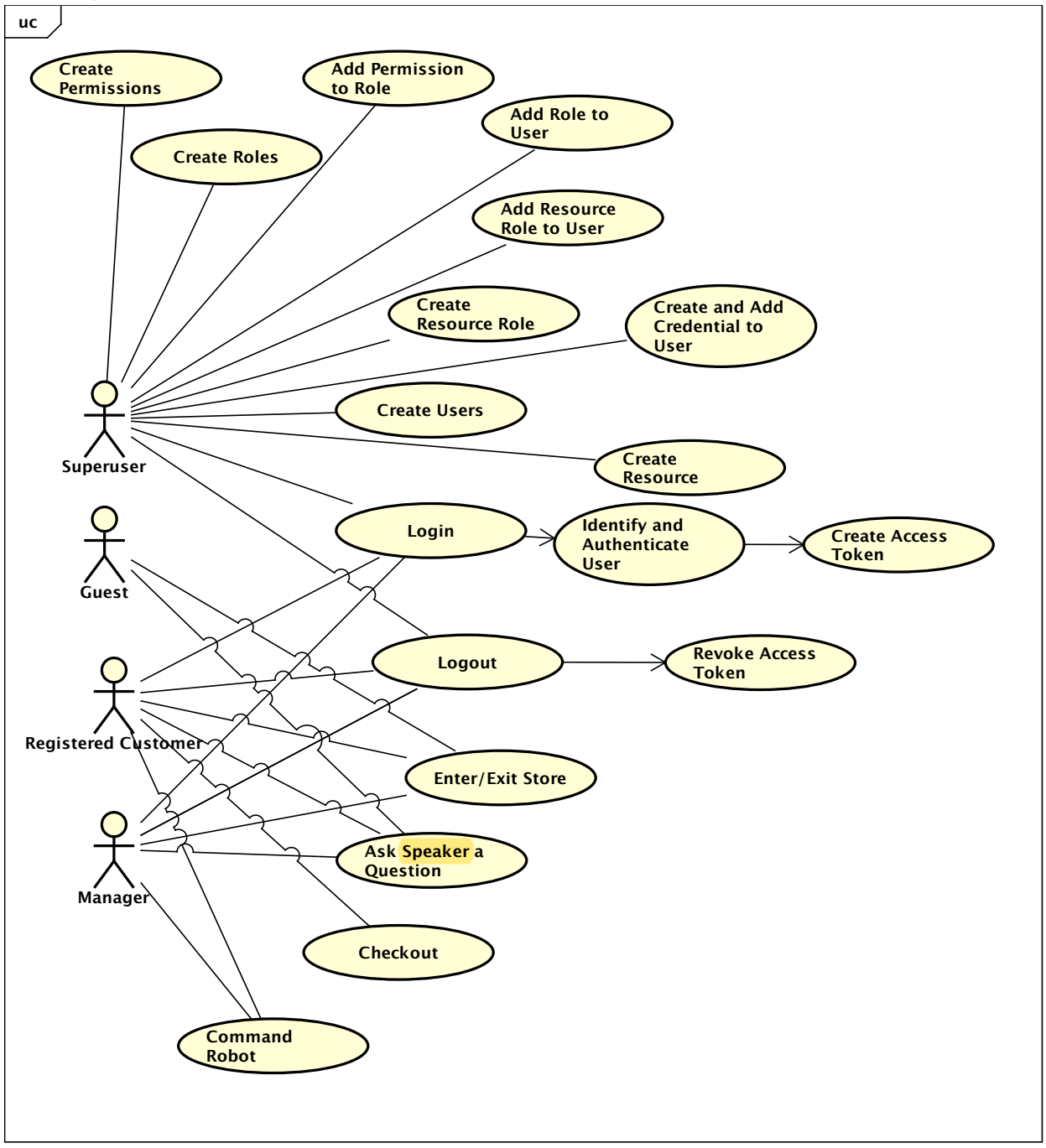
1. Users have Credentials that are used to validate login.
2. Users are assigned Roles and ResourceRoles that give them Permissions required to access certain resources or take certain actions.
3. Users include registered customers, guests, and store managers.

Credentials

1. There are three types of Credentials: Username, VoicePrint, and FacePrint.
2. Credentials are used to valid User login.

Use Cases

The following Use Case diagram describes the use cases supported by the Authentication System.



Actors

The actors of the Ledger System include Superusers, Registered Customers, Guests, and Store Managers.

Superuser

The Superuser is responsible for creating Store Model objects and resources including but not limited to Stores, Aisles, Customers, and Devices. They are also responsible for creating new Users, defining Roles, Permissions, and Resource Roles, and assigning those Entitlements to Users.

Guest

Guests are a type of User that can only enter and exit a store. Their permissions do not allow them to interact with most devices. However, their movements are still tracked by sensors as they move around the store.

Registered Customer

Registered Customers are a type of User that can enter and exit a store as well as interact with devices to perform most actions including asking questions, asking robots to bring products, and checking out.

Manager

A Store's Manager is a type of User that can enter and exit a store as well as interact with devices to command robots to perform actions like cleaning. Their movements are not tracked by devices, nor do the Managers perform Customer actions like filling baskets or checking out.

Use Cases

Create Permissions

Creating a permission makes the permission available to be assigned to roles. This action can only be taken by superusers of the Store 24x7 system.

Create Roles

Creating a role makes the role available to be assigned to users and associated to permissions and resource roles. This action can only be taken by superusers of the Store 24x7 system.

Add Permission to Role

Permissions are associated to roles and then associated with users and resource roles. This action can only be taken by superusers of the Store 24x7 system.

Create User

Users represent any user of the system. This could be superusers, managers, customers, or guests. Users objects that are associated to users must first be created in the system so that they can be assigned roles and permissions. This action can only be taken by superusers of the Store 24x7 system.

Add Role to User

Roles are associated to users and determine what actions they have permission to execute. This action can only be taken by superusers of the Store 24x7 system.

Create Resource

Resources represent physical entities that may be interacted with in a store. Creating resources allows them to be associated to users and roles through resource roles. This action can only be taken by superusers of the Store 24x7 system.

Create Resource Roles

Resource roles are created to associate roles to resources. This action can only be taken by superusers of the Store 24x7 system.

Add Resource Role to User

Resource Roles are associated to Users to assist in handling permissions and access. This action can only be taken by superusers of the Store 24x7 system.

Create and Add Credentials to Users

Credentials are what “recognize” a user and logs them into the system. They may be a username and password, face print, or voiceprint. Users may not have all types of credentials but could have multiple. Creating credentials and associating them to a user is what allows the user to access and interact with the Store 24x7 system. This action can only be taken by superusers of the Store 24x7 system.

Login, Identify and Authenticate User, Create Access Token

Users login with their credentials to access the Store 24x7 system. After logging in, the user receives a valid auth token to access the rest of the system (per what is permitted by their permissions).

Logout, Revoke Access Token

Users logout by performing certain actions such as a forced logout by a superuser or walking out of a store in the case of a customer. When the user logs out, their authentication token is deactivated.

Enter/Exit Store

Guests, Registered Customers, and Store Managers may enter and exit the store. They are logged in upon entry and logged out upon exit.

Ask Speaker a Question

Certain actions with devices are restricted depending on who the user is. Guests, Registered Customers, and Managers may all ask speakers questions.

Command Robot

Certain actions with devices are restricted depending on who the user is. Only Registered Customers and Managers may command robots.

Checkout

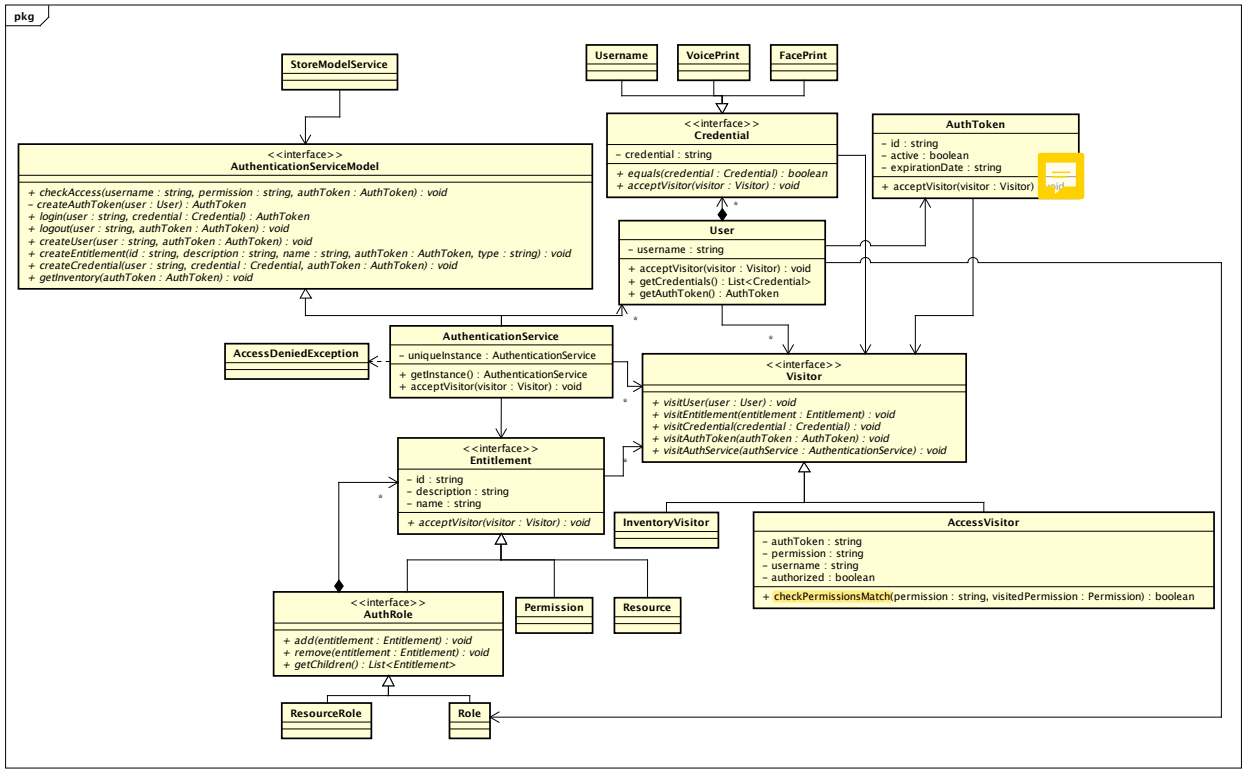
Certain actions with devices are restricted depending on who the user is. Only Registered Customers may checkout.

Design

This section of the document will describe the implementation details for the Store Controller Service.

Class Diagram

The following class diagram defines the classes defined in this design.



Class Dictionary

This section specifies the class dictionary for the Store Authentication Service. The classes should be defined within the package com.cscie97.store.authentication.

The StoreModelService defined in the Store Model Service design document.

AuthenticationService

The AuthenticationService extends the AuthenticationServiceModel interface. The AuthenticationService represents the Authentication Service API. It is associated to Entitlements, Users, and Visitors. It implements the Singleton Pattern to ensure that only one instance of the AuthenticationService is ~~every~~ created for the 24x7 System.

Methods

Method Name	Signature	Description
checkAccess	(username: string, permission: string, authToken: AuthToken): void	Check whether a user has a particular permission by created an AccessVisitor. Throw an AccessDeniedException if the user does not have the permission.

createAuthToken	(user: User): AuthToken	Private method that create a new auth token for a user. Used on login. If an auth token for the user already exists, returns the existing AuthToken, sets its state to active, and reset the expiration date.
login	(user: string, credential: Credential): boolean	Accept a user's name, looks up the associated User, checks if the provided Credential is equal to one of the Credentials associated with the User, and calls createAuthToken to either create or activate an auth token if so. If not, throw an AccessDeniedException.
logout	(user: string, authToken: AuthToken): boolean	Accepts a user's name, looks up the associated User, check if the provided AuthToken is valid for the user, then deactivate the AuthToken. If the AuthToken is invalid, throw an AccessDeniedException.
createUser	(user: string, authToken: AuthToken): void	Creates a user for the given username. If the AuthToken is invalid, throw an AccessDeniedException.
createEntitlement	(id: string, description: string, name: string, type: string, authToken: AuthToken): void	Creates an entitlement. If the AuthToken is invalid, throw an AccessDeniedException.
createCredential	(user: string, credential: Credential, authToken: AuthToken): void	Creates an credential for a user. If the AuthToken is invalid, throw an AccessDeniedException.
getInventory	(authToken: AuthToken): void	Prints all authentication objects. If the AuthToken is invalid, throw an AccessDeniedException.
getInstance	(): AuthenticationService	Gets the AuthenticationService Singleton.
acceptVisitor	(visitor: Visitor): void	Accepts a visitor.

Properties

Property Name	Type	Description
uniqueInstance	AuthenticationService	AuthenticationService Singleton.

Associations

Association Name	Type	Description
users	HashMap<string, User>	A list of Users maintained by the authentication system.
visitor	Visitor	An accepted visitor.
entitlements	HashMap<string, Entitlement>	A list of Entitlements maintained by the authentication system.

Permission

Permission extends the Entitlement interface. It is a leaf in the Composite Pattern. Permissions represent the permission required to access a resource.

Methods

Method Name	Signature	Description
acceptVisitor	(visitor:Visitor): void	Accepts a visitor.

Properties

Property Name	Type	Description
id	string	Unique identifier for the Entitlement.
description	string	Description of the Entitlement.
name	string	Name of the entitlement.

Associations

Association Name	Type	Description
visitor	Visitor	An accepted visitor.

Resource

Resource extends the Entitlement interface. It is a leaf in the Composite Pattern. Resources represent the physical entities that require access in a store.

Methods

Method Name	Signature	Description
acceptVisitor	(visitor:Visitor): void	Accepts a visitor.

Properties

Property Name	Type	Description
id	string	Unique identifier for the Entitlement.
description	string	Description of the Entitlement.
name	string	Name of the entitlement.

Associations

Association Name	Type	Description
visitor	Visitor	An accepted visitor.

Role

Role extends the Entitlement interface. It is a composite in the Composite Pattern. Roles represent the combined authorizations of a User in respect to their role in the Store 24x7 system. Roles are associated to Permissions.

Methods

Method Name	Signature	Description
acceptVisitor	(visitor:Visitor): void	Accepts a visitor.
getChildren	(): List<Entitlement>	Returns a list of associated Entitlements.

Properties

Property Name	Type	Description
id	string	Unique identifier for the Entitlement.
Description	string	Description of the Entitlement.
Name	string	Name of the entitlement.

Associations

Association Name	Type	Description
permissions	HashMap<String, Permission>	A list of permissions associated to a Role.
Visitor	Visitor	An accepted visitor.

ResourceRole

ResourceRole extends the Entitlement interface. It is a composite in the Composite Pattern. ResourceRoles represent the relationship between Roles and Resources. ResourceRoles are associated to both Roles and Resources.

Methods

Method Name	Signature	Description
acceptVisitor	(visitor:Visitor): void	Accepts a visitor.
getChildren	(): List<Entitlement>	Returns a list of associated Entitlements.

Properties

Property Name	Type	Description
id	string	Unique identifier for the Entitlement.
Description	string	Description of the Entitlement.
Name	string	Name of the entitlement.

Associations

Association Name	Type	Description
role	Role	The Role associated to the ResourceRole.
resource	Resource	The Resource associated to the ResourceRole.
visitor	Visitor	An accepted visitor.

InventoryVisitor

InventoryVisitor extends the Visitor interface. It is a concrete visitor in the Visitor Pattern. It visits all the authentication objects in the Authentication System and prints out object details.

Methods

Method Name	Signature	Description
visitUser	(user: User): void	Visits a User object.
visitEntitlement	(entitlement: Entitlement): void	Visits an Entitlement object.
visitCredential	(credential: Credential): void	Visits a Credential object.
visitAuthToken	(authToken: AuthToken): void	Visits an Auth Token object.
visitAuthService	(authService: AuthService): void	Visits the Authentication Service object.

AccessVisitor

AccessVisitor extends the Visitor interface. It is a concrete visitor in the Visitor Pattern. It visits a User Object and its associated Roles and Permissions to determine if a User has the access required to perform an action.

Methods

Method Name	Signature	Description
visitUser	(user: User): void	Visits a User object.
visitEntitlement	(entitlement: Entitlement): void	Visits an Entitlement object.
visitCredential	(credential: Credential): void	Visits a Credential object.
visitAuthToken	(authToken: AuthToken): void	Visits an Auth Token object.
visitAuthService	(authService: AuthService): void	Visits the Authentication Service object.

Properties

Property Name	Type	Description
authToken	AuthToken	The auth token that is provided by the client.
permission	string	The permission type provided by the client.
username	string	The username provided by the client.
authorized	boolean	The final result – whether or not a user has access.

User

User represents the various users of the Store 24x7 system. These can include customers, managers, and superusers. Users may by associated to an Auth Token, various Credentials, and various Roles.

Methods

Method Name	Signature	Description
acceptVisitor	(visitor: Visitor): void	Accepts a visitor.
getCredentials	(): HashMap<String, Credential>	Returns associated credentials.
getAuthToken	AuthToken	Returns the User's auth token.

Properties

Property Name	Type	Description
username	string	The user's username.

Associations

Association Name	Type	Description
credentials	HashMap<String, Credential>	List of associated Credentials.
visitor	Visitor	The accepted visitor.
roles	HashMap<String, Role>	List of associated Roles.
authToken	AuthToken	The user's auth token.

Username

Username extends the Credential interface. Its credential is comprised of a "Username:Password".

Methods

Method Name	Signature	Description
equals	(credential: string)	Determines if a credential that is passed matches the Credential.
acceptVisitor	(visitor: Visitor)	Accepts a visitor.

Properties

Property Name	Type	Description
credential	string	Username:Password

Associations

Association Name	Type	Description
visitor	Visitor	The accepted visitor.

VoicePrint

VoicePrint extends the Credential interface. Its credential is comprised of a “Voice:Username”.

Methods

Method Name	Signature	Description
equals	(credential: string)	Determines if a credential that is passed matches the Credential.
acceptVisitor	(visitor: Visitor)	Accepts a visitor.

Properties

Property Name	Type	Description
credential	string	Voice:Username

Associations

Association Name	Type	Description
visitor	Visitor	The accepted visitor.

FacePrint

FacePrint extends the Credential interface. Its credential is comprised of a “Face:Username”.

Methods

Method Name	Signature	Description
equals	(credential: string)	Determines if a credential that is passed matches the Credential.
acceptVisitor	(visitor: Visitor)	Accepts a visitor.

Properties

Property Name	Type	Description
credential	string	Face:Username

Associations

Association Name	Type	Description
visitor	Visitor	The accepted visitor.

AuthToken

AuthToken represents the authentication token that gives access to a User to perform actions in the Store 24x7 System. Auth Tokens may be active or inactive, an ID, and an expiration date that is refreshed whenever the auth token is activated.

Methods

Method Name	Signature	Description
acceptVisitor	(visitor: Visitor): void	Accepts a visitor.

Properties

Property Name	Type	Description
id	string	Unique ID of an authentication token.
active	boolean	True if the auth token is active. False otherwise.
expirationDate	string	When the auth token expires.

Associations

Association Name	Type	Description
visitor	Visitor	The accepted visitor.

Design Details

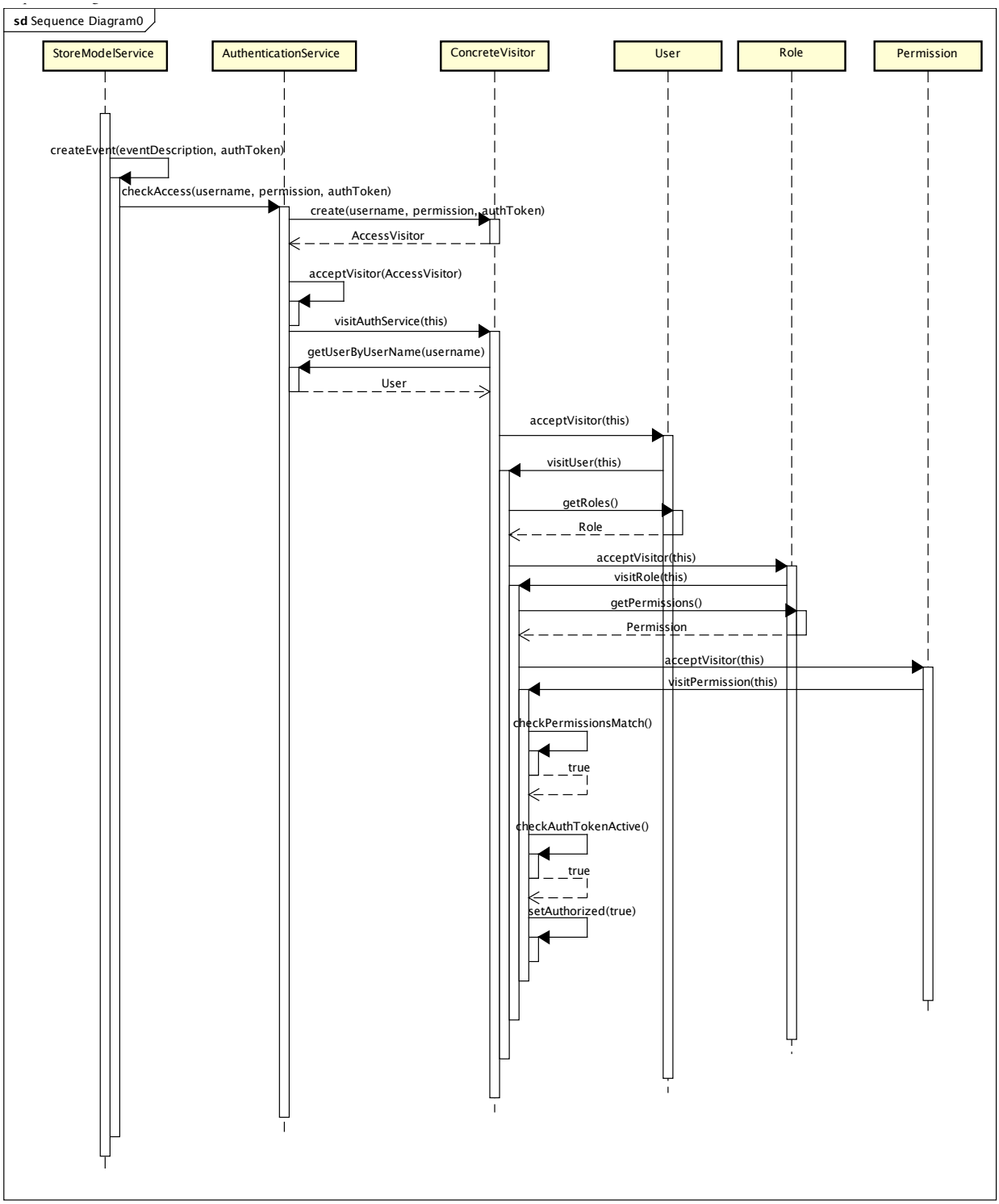
The core component of the Authentication Service is the AuthenticationService Class. The AuthenticationService provides an API for interaction with the Authentication Service and implements the API methods that manage Users, Entitlements (Roles, Resource Roles, Resources, and Permissions), Credentials, and Auth Tokens.

The Auth Token expiration date should be changed depending on the required security of the application. A timeout of 1 day is recommended for this application. Validity of the authentication token is determined both by the active state and the expiration date. If the token is active but past the expiration date, then the access should be denied. Ideally there would be a crone/scheduled job to clean out the authentication tokens, but that is not required for this implementation.
Entitlements have many similar properties – ID, description, name, and the acceptVisitor() method and are tightly associated with each other. Therefore, they should be implemented following the Composite pattern with ResourceRole and Role being composites and Permission and Resource being leaves.

Additionally, consider the following implementation of the AuthenticationService, which follows the Singleton Pattern. Only one instance of the AuthenticationService may be created so long as the AuthenticationService.getInstance() method is called rather than creating a new AuthenticationService.

```
public class AuthenticationService {
      private static AuthenticationService uniqueInstance;

      // variables

      private AuthenticationService() {}

      public static AuthenticationService getInstance() {
            if (uniqueInstance == null) {
                  uniqueInstance = new AuthenticationService();
            }
            return uniqueInstance;
      }

      // other methods
}
```

Below is an instance diagram that shows the checkAccess() method in action. The AuthenticationService uses Visitor Pattern to traverse through Users, Roles, Auth Tokens, and Permissions to determine whether the permission matches the required permission level provided by the client and whether the auth token is valid.



Exception Handling

AccessDeniedException

The AccessDeniedException is returned from the Store Authentication Service in response to an invalid login, authToken, or access request. The AccessDeniedException captures the action that was attempted and the reason for the failure.

Properties

Property Name	Type	Description
action	string	The action that was performed (ex. “translateCommand”).
reason	string	The reason for the exception (ex. “Event not configured to command”).

Testing

Functional

Implement a test driver class called TestDriver that implements a static main() method. The main() method should accept a single parameter, which is a command file. The main method will call the CommandProcessor.processCommandFile(file:string) method, passing in the name of the provided command file. The TestDriver class should be defined within the package “com.cscie97.store.test”.

The test script “store.script” should parse and run as expected (according to the comments in the file). Additional scripts are also provided in the same folder for other functional tests.

Performance

N/A

Regression

When changes are made, re-run the “store.script” test script to ensure results are as expected.

Exception

When exceptions are raised, ensure meaningful error messages are printed to stdout that notify the user where in their script the error occurred.

Risks

Document any risks identified during the design process.
Are there parts of the design that may not work or need to be implemented with special care or additional testing?