*Authentication Service* Design Document

Date: 11/20/2019

Author: Tofik Mussa

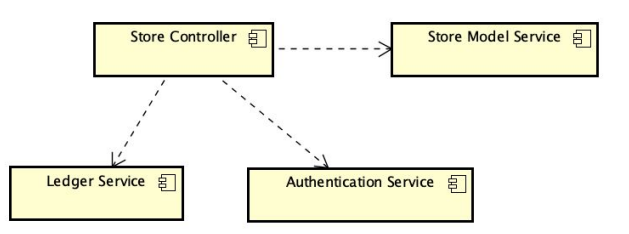
Reviewer(s): Trisha Singh, Peter Chen

# Introduction

This design document defines the Authentication as a generic service that is multipurpose and reusable service. However, it leans towards integrating previously defined services such as the Store Model Service, Store Controller Service and Ledger Service for a secure and satisfactory customer experience. The Authentication Service determines if the user is who he/she claims is and if he/she has permission to perform certain tasks. Several workflows will be examined in the lens of the Store 24X7 system.

Overview

Security is crucial to a self-regulated system like the Store 24X7 system. A poorly managed ecosystem may result in an economic loss, bad reputation and safety issues. A robust system that identifies users with their corresponding permissions is an essential component of the Authentication Service. It prevents intruders from abusing the Store 24X7 system and it reassures valuable customers that the system is reliable and trustworthy. The overall cost of developing a robust authentication service by far outweighs the amount of development hours spent on it. Since it is a maintainable and well-designed system it anticipates future changes pertaining to the Open Closed Principle. The system is modular with clearly defined boundaries and loose dependencies making it integrate easily with other subsystems. For a high-level overview, see the image below



# 

# Requirements

In the context of the Store 24X7 system, the authentication service maintains an inventory of users, role, permissions, resources and tokens which enables to guard the store. The Store Model Service and the Store Controller Service delegate authentication/authorization to this service.

*This section provides a summary of the requirements for the <Component Name>.*

*Provide your understanding of the requirements, both functional and nonfunctional. Reference the provided Requirements and System Architecture documents. Do not cut and paste from the requirements document.*

*Product Manager and others can read this to understand what requirements your design will support. There is already a requirements doc, so keep this brief and to the point, highlighting the important requirements that the design is addressing. Structure in a way to provide a requirements checklist for your design.*

# Use Cases

*Enumerate the use cases supported by the design,*

*This design supports the following use cases:*

*Include a Use Case Diagram.*

*Include descriptions of each of the actors and use cases.*

# Implementation

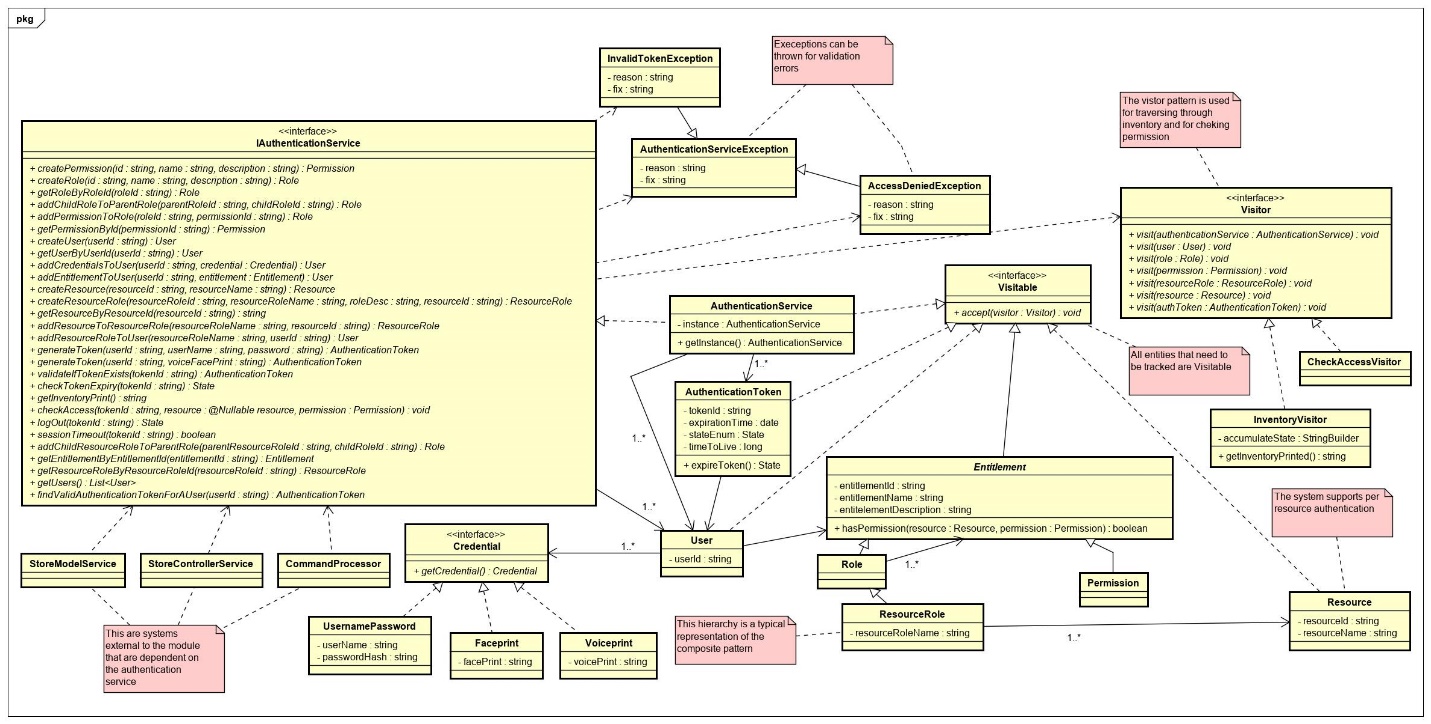
*This section of the document will describe the implementation details for ...*

*The implementation section should cover the following topics:*

* *What are the classes, and their properties, associations and methods?*
* *What are the important interfaces and how they will be implemented?*
* *How are the requirements addressed?*

# Class Diagram

*The following diagram mainly shows classes under the package ‘com.cscie97.store.authentication’. A snapshot of the services dependent on the authentication service are also briefly discussed. There was some modifications to the existing systems to allow security controls.*



# 

# Class Dictionary

*This section describes the classes under the package ‘com.cscie97.store.authentication’. Finer implementation details are also discussed when necessary.*

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| createPermission | (String permissionId, String permissionName, String permissionDescription): Permission |  |
| createRole | (String roleId, String roleName, String roleDescription): Role |  |
| addPermissionToRole | (String roleId, String permissionId): Role |  |
| addChildRoleToParentRole | (String parentRoleId, String childRoleId): Role |  |
| getRoleById | (String roleId): Role |  |
| getPermissionById | (String permissionId): Permission |  |
| createUser | (String userId): User |  |
| getUserByUserId | (String userId): User |  |
| addCredentialsToUser | (String userId, Credential credential): User |  |
| addEntitlementToUser | (String userId, Entitlement entitlement): User |  |
| createResource | (String resourceId, String resourceName): Resource |  |
| getResourceByResourceId | (String resouceId): Resource |  |
| createResourceRole | (String resourceRoleId, String resourceRoleName, String resourceRoleDesc, String resourceId): ResourceRole |  |
| addEntitlementsToResourceRole | (String resourceRoleId, String entitlementId): ResourceRole |  |
| addResourcesToResourceRole | (String resourceId, String resourceRoleId): ResourceRole |  |
| getResourceRoleByResourceRoleId | (String resourceRoleId): ResourceRole |  |
| getEntitlementByEntitlementId | (String entitlementId): Entitlement |  |
| addResourceRoleToUser | (String userId, String resourceRoleId): User |  |
| addChildResourceRoleToParentRole | (String parentRoleId, String childResourceRoleId): Role |  |
| generateToken | (String userId, String userName, String password): AuthenticationToken |  |
| generateToken | (String userId, String voiceFacePrint): AuthenticationToken |  |
| validateIfTokenExistsAndIsValid | (String tokenId): AuthenticationToken |  |
| checkTokenExpiry | (String tokenId): State |  |
| sessionTimedOut | (String tokenId): boolean |  |
| logOut | (String userId):  State |  |
| getInventoryPrint | (): String |  |
| getUsers | (): List<User> |  |
| getTokens | (): List<AuthenticationToken> |  |
| findValidAuthenticationTokenForAUser | (String userId): AuthenticationToken |  |
| checkAccess | (String tokenId, Resource resource, Permission permission):  AccessDeniedException |  |

## *AuthenticationService*

*Provides implementation for the authentication service. It stores users, entitlements and users*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| createPermission | (String permissionId, String permissionName, String permissionDescription): Permission |  |
| createRole | (String roleId, String roleName, String roleDescription): Role |  |
| addPermissionToRole | (String roleId, String permissionId): Role |  |
| addChildRoleToParentRole | (String parentRoleId, String childRoleId): Role |  |
| getRoleById | (String roleId): Role |  |
| getPermissionById | (String permissionId): Permission |  |
| createUser | (String userId): User |  |
| getUserByUserId | (String userId): User |  |
| addCredentialsToUser | (String userId, Credential credential): User |  |
| addEntitlementToUser | (String userId, Entitlement entitlement): User |  |
| createResource | (String resourceId, String resourceName): Resource |  |
| getResourceByResourceId | (String resouceId): Resource |  |
| createResourceRole | (String resourceRoleId, String resourceRoleName, String resourceRoleDesc, String resourceId): ResourceRole |  |
| addEntitlementsToResourceRole | (String resourceRoleId, String entitlementId): ResourceRole |  |
| addResourcesToResourceRole | (String resourceId, String resourceRoleId): ResourceRole |  |
| getResourceRoleByResourceRoleId | (String resourceRoleId): ResourceRole |  |
| getEntitlementByEntitlementId | (String entitlementId): Entitlement |  |
| addResourceRoleToUser | (String userId, String resourceRoleId): User |  |
| addChildResourceRoleToParentRole | (String parentRoleId, String childResourceRoleId): Role |  |
| generateToken | (String userId, String userName, String password): AuthenticationToken |  |
| generateToken | (String userId, String voiceFacePrint): AuthenticationToken |  |
| validateIfTokenExistsAndIsValid | (String tokenId): AuthenticationToken |  |
| checkTokenExpiry | (String tokenId): State |  |
| sessionTimedOut | (String tokenId): boolean |  |
| logOut | (String userId):  State |  |
| getInventoryPrint | (): String |  |
| getUsers | (): List<User> |  |
| getTokens | (): List<AuthenticationToken> |  |
| findValidAuthenticationTokenForAUser | (String userId): AuthenticationToken |  |
| checkAccess | (String tokenId, Resource resource, Permission permission):  AccessDeniedException |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| instance | IAuthenticationService | Used to provide a singleton instance to clients |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| users | List<User> |  |
| tokens | List<AuthenticationToken> |  |
| entitlements | List<Entitlement> |  |
| resources | List<Resource> |  |

## *IVisitor*

*This interface sets the contract for the type of visitors that are defined in this service. Implementations are free to provide logic that leverages the visitor design pattern and abides by this contract.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| visit | (AuthenticationService authenticationService): void |  |
| visit | (User user): void |  |
| visit | (Role role): void |  |
| visit | (Permission permission): void |  |
| visit | (ResourceRole resourceRole): void |  |
| visit | (Resource resource): void |  |
| visit | (AuthenticationToken authenticationToken): void |  |

## *Visitable*

All entities that need to be visited implement this interface

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| accept | (Ivisitor visitor): void |  |

## *CheckAccessvisitor*

*This is a special type of visitor that traverses through the entitlement tree that was structured using the composite pattern and checks for a specific permission along its way.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| visit | (AuthenticationService authenticationService): void |  |
| visit | (User user): void |  |
| visit | (Role role): void |  |
| visit | (Permission permission): void |  |
| visit | (ResourceRole resourceRole): void |  |
| visit | (Resource resource): void |  |
| visit | (AuthenticationToken authenticationToken): void |  |
| hasPermission | (): boolean |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| hasPermission | List<Boolean> |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
| token | AuthenticationToken |  |
| resource | Resource |  |
| permission | Permission |  |

## *InventoryVisitor*

*This visitor traverses through all of the entities and collects their details to be used later for printing.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
| getInventoryPrint | (): String |  |
| visit | (AuthenticationService authenticationService): void |  |
| visit | (User user): void |  |
| visit | (Role role): void |  |
| visit | (Permission permission): void |  |
| visit | (ResourceRole resourceRole): void |  |
| visit | (Resource resource): void |  |
| visit | (AuthenticationToken authenticationToken): void |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

## *IAuthenticationService*

*A top-level interface exposing methods accessible to the outside world. This interface is tasked with defining entities and providing access to them once they are defined. This interface an orchestration engine and an entry point to the service.*

***Methods***

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Signature** | **Description** |
|  |  |  |

***Properties***

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
|  |  |  |

***Associations***

|  |  |  |
| --- | --- | --- |
| **Association Name** | **Type** | **Description** |
|  |  |  |

...

# Implementation Details

*Explain details of the implementation.*

*How do the various parts fit together or interact?*

*How does the design address the requirements? Justify your design decisions and how they address the requirements.*

*Some implementation details may be addressed in the class dictionary, but for things that are not, describe them here.*

*Remember to reference the requirements from the body of the design document to show how your design is addressing the requirements.*

# Exception Handling

*Provide details on your exception handling. What types of exceptions are expected and how are they handled by the design? Describe your exception classes and their properties.*

# Testing

*Provide a testing strategy for testing the component.*

* *Functional*
* *Performance*
* *Regression*
* *Exception Handling*

# 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?*