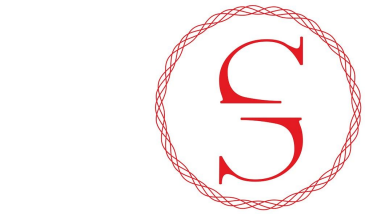


SUPER HEROES IN TRAINING

My Academic Pyramid

Low Level Design Document

CECS 491A Sec 05

December 11, 2018

Team Leader: Krystal Leon, 013986607

Arturo Peña Contreras, 010914811

Luis Julian, 007472593

Hyunwoo Kim, 014392909

Victor Kim, 012016990

Trong Nguyen, 016208983

Revision History

Date	Version	Description
12/11/18	1.0	First draft.

Table of Contents

1. Introduction

This document is made in order to understand low level design per individual feature within the “My Academic Pyramid” web application.

1.1 Purpose of the Document

This document will include the diagrams which shows the flow of data through the feature along with its own error handling diagrams. These diagrams are designed in order for developers to understand the features that we are currently building as well as the errors that can arise in certain areas.

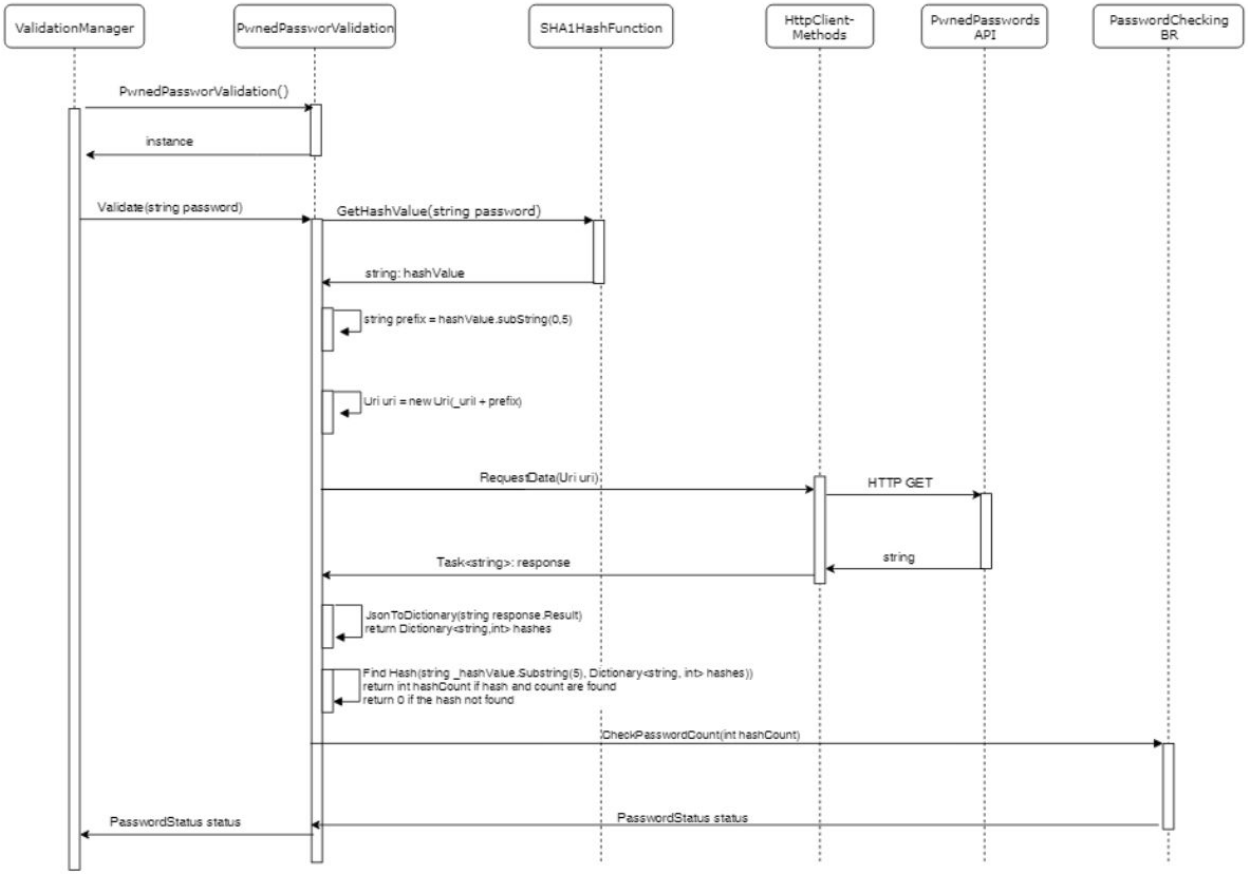
2. Password Validation

A password will be checked to see if it has been breached, returning a status integer representing the level to security of the password.

2.1 Low Level Diagram

To validate a password passed in, the password is first hashed using a SHA1 hash function. A prefix of the hash value is then added as a path to the Pwned Passwords API url. The resulting url is passed into RequestData(Uri uri) the perform an HTTP GET request to the Pwned Passwords API. The list of breached passwords containing the prefix is then returned as a string response. That string is deserialized from JSON to a Dictionary of hash values and counts through JsonToDictionary(string response). Then this dictionary is searched in FindHash(string hashValue, Dictionary<string,int> hashes) to find if the matching hash has been breached. It it is found, it will return the count, and if not it will return zero. Once the count is returned,

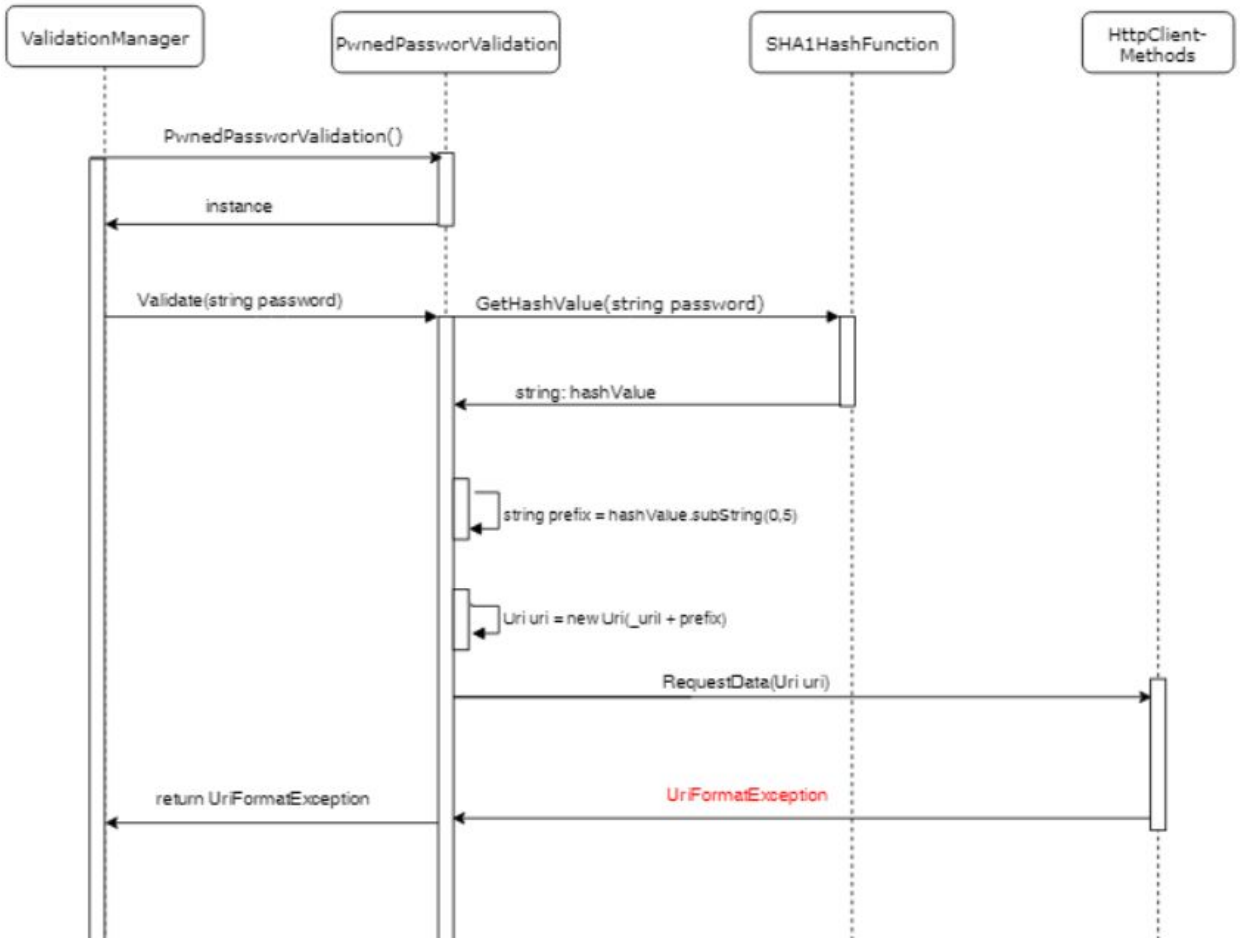
CheckPasswordCount(int hashCount) is then called to validate the business rules for password vulnerability validation, which returns a PasswordStatus (contains int status which determines how safe a password is; 0 - has never been breached, 1 - has been breached once, 2 - has been breached more than once).



Sequence Diagram for PwnedPasswordValidation

2.2 Error Handling UriFormatException

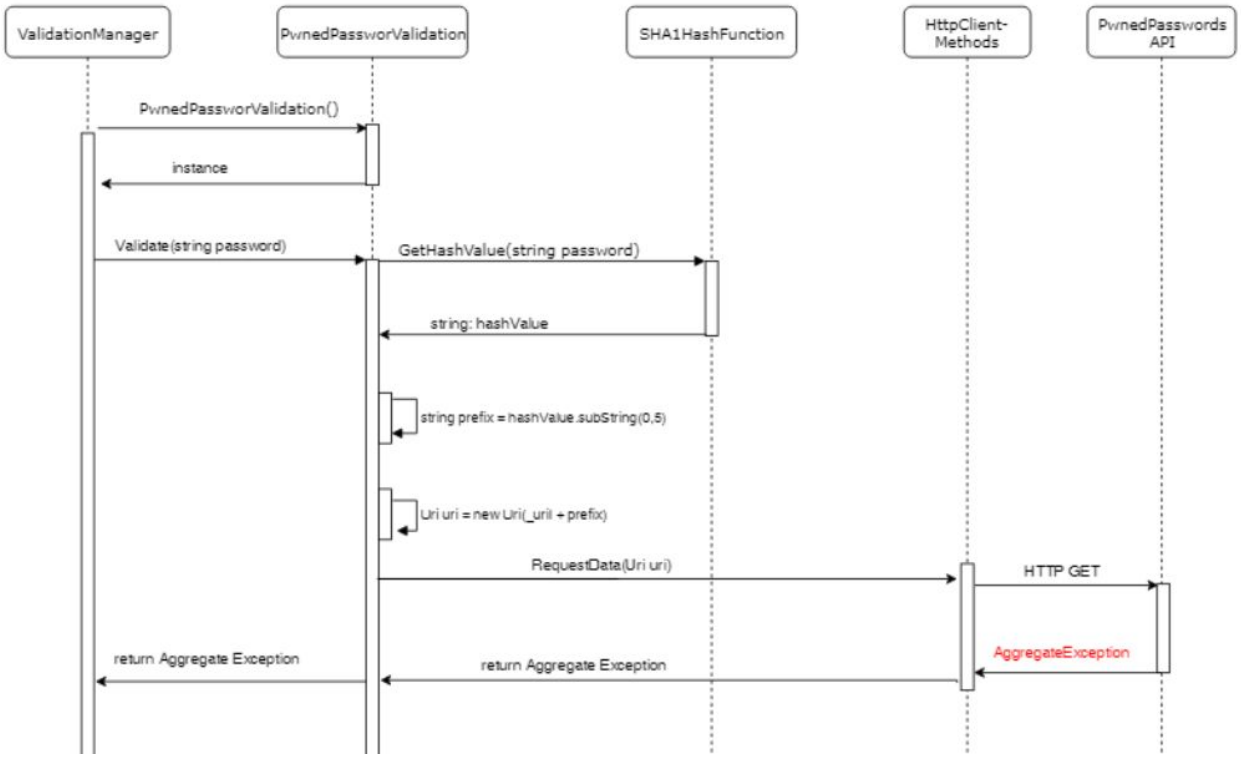
If an invalid url is passed through RequestData(Uri uri), a *UriFormatException* will be thrown because the GET request to the API server would fail.



Sequence Diagram of Password Checking for Error Scenario UriFormatException

2.3 Error Handling AggregateException

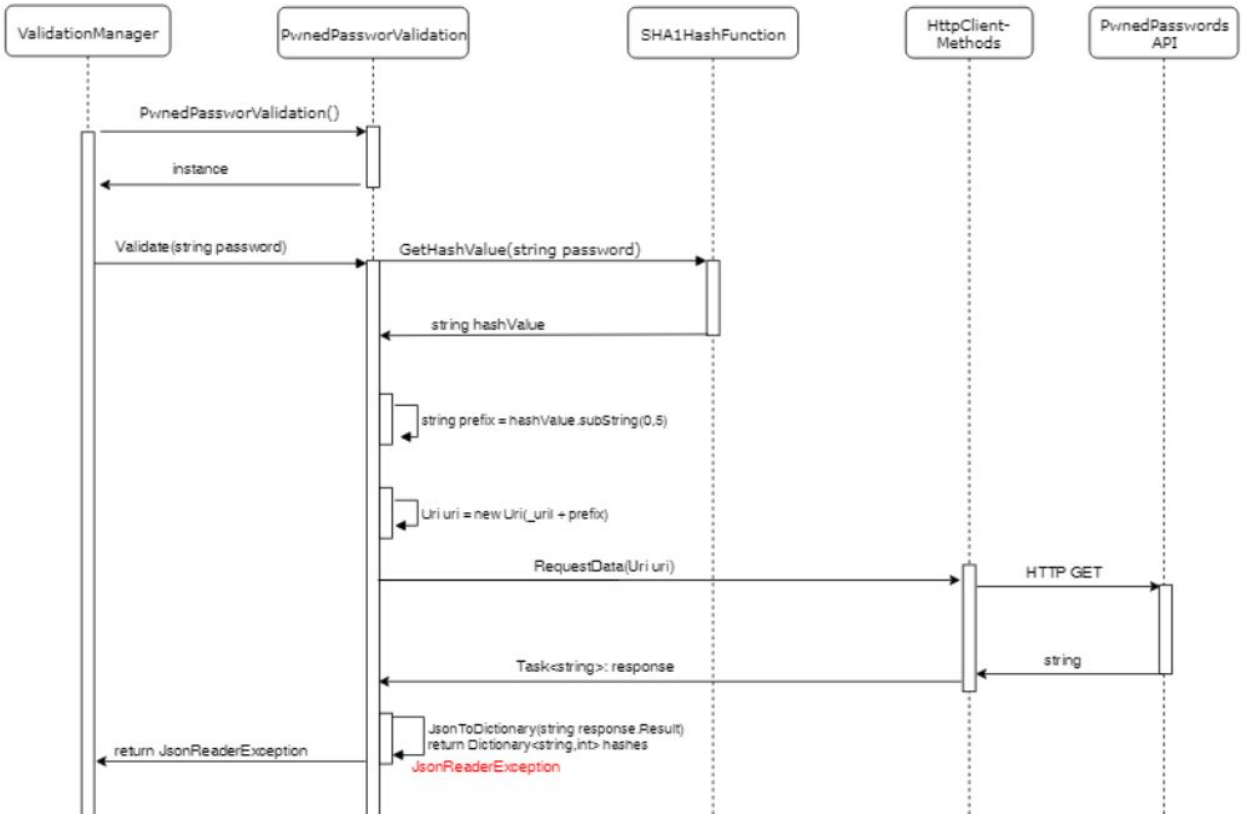
An *AggregateException* will be thrown if the task is cancelled while requesting data from the Pwned Passwords API.



Sequence Diagram of Password Checking for Error Scenario AggregateException

2.4 Error Handling JsonReaderException

A *JsonReaderException* will be thrown if JsonToDictionary(string response) attempts to deserialize a string that is not in valid JSON format.



Sequence Diagram of Password Checking for Error Scenario JsonReaderException

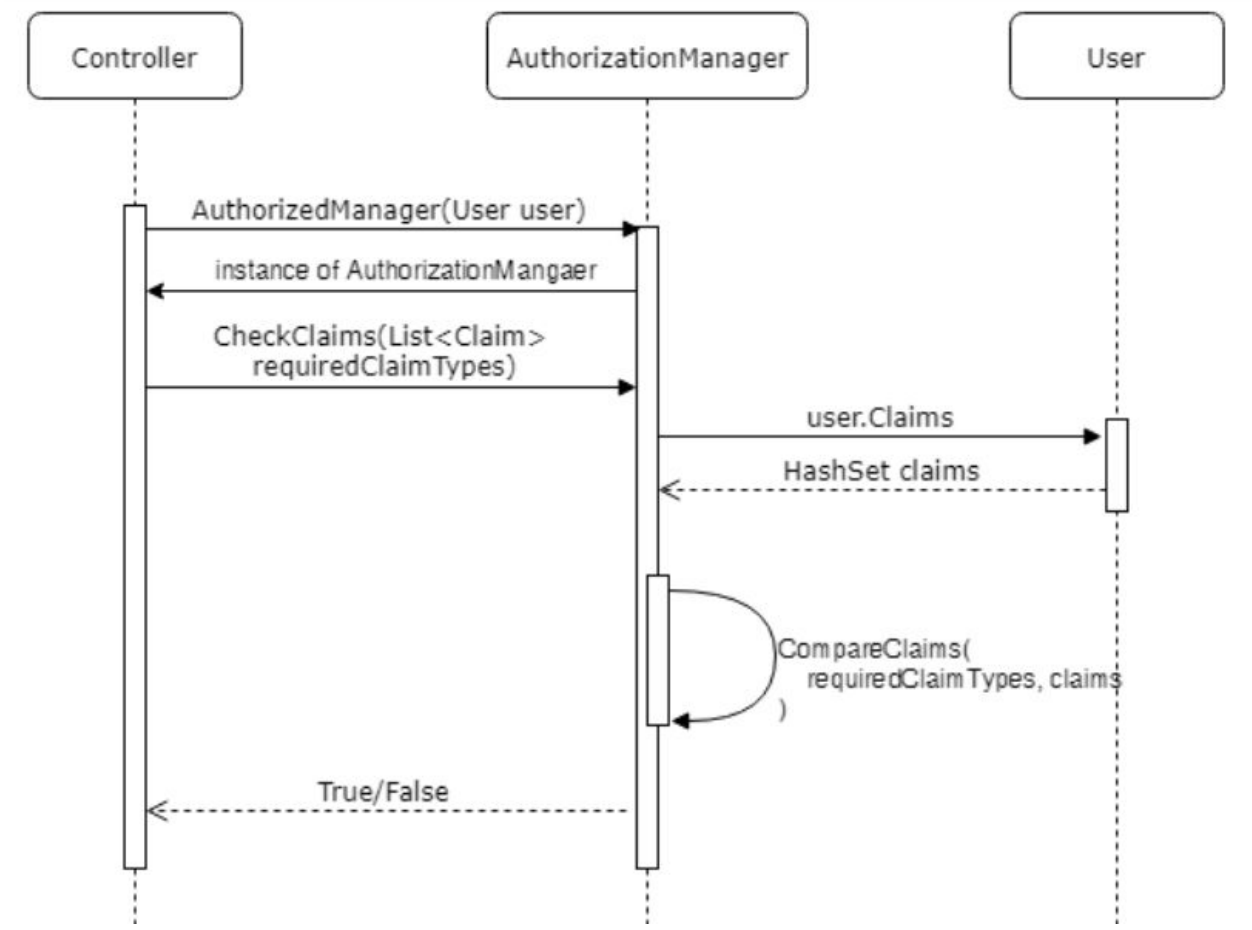
3. Authorization

This section is regarding how authorization will be handled in our web application.

3.1 Low Level Diagram

We have decided to use claim-based authorization, as it adapts to new changes of business rules well. Therefore, it is more extensible compared to role-based authorization, because role-based authorization requires adding a new authority to every role whenever there is an additional feature. The controller creates an instance of AuthorizationManager using a user who requested for the permission. The controller checks whether that user contains the required

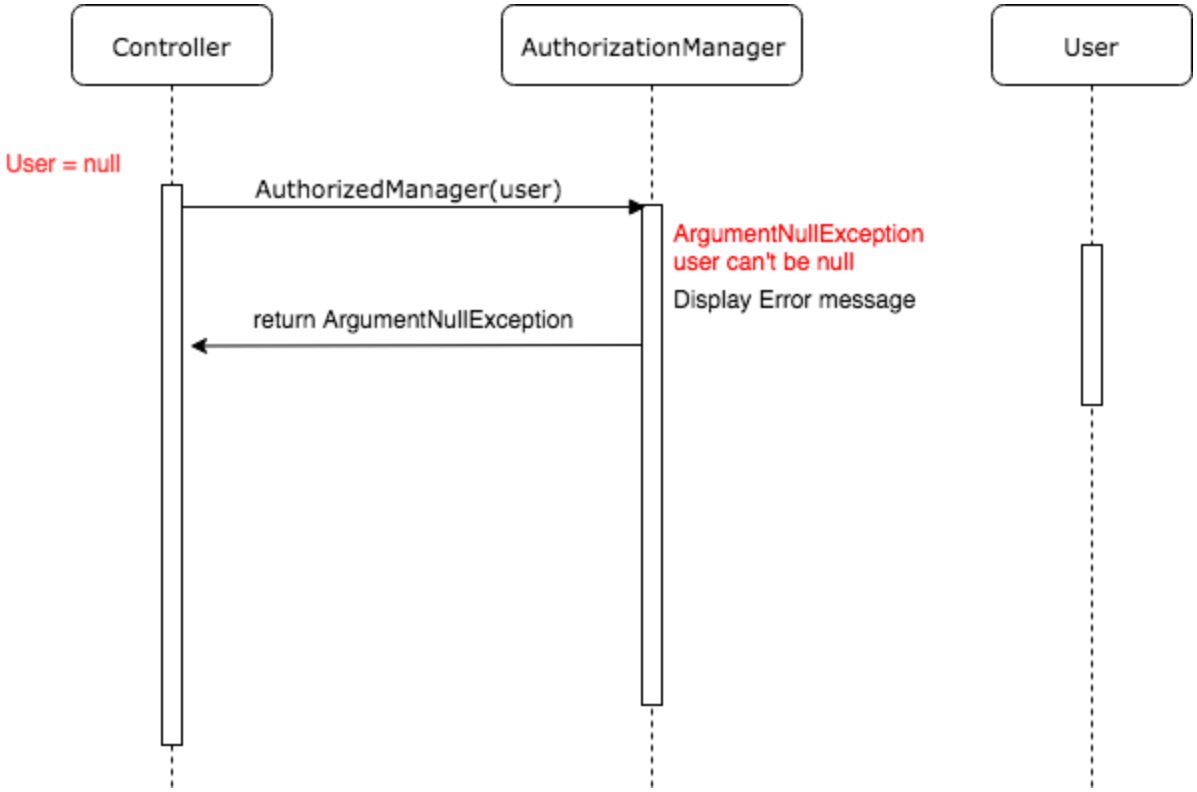
claims in the list using CheckClaims(requiredClaims). If a user does not have required claims in the list, it would return false and the user's request would be denied. If the user has them in the list, it would return true and the user would get an access to the feature.



Sequence Diagram of Authorization

3.2 Error Handling ArgumentNullException

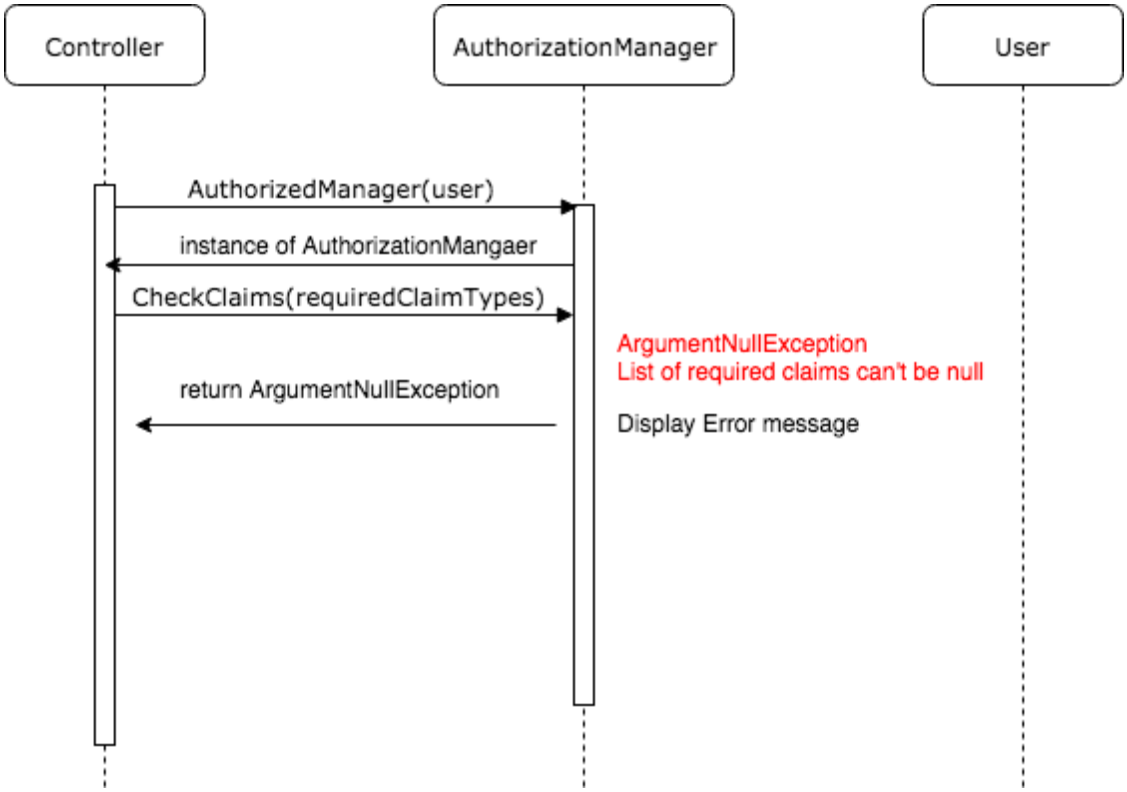
The value of the user cannot be null in order to create an instance of the AuthorizedManager class. If the value of the user happens to be null, it would mean that the user does not exist. If the null value is passed in, a constructor of AuthorizationManager would throw *ArgumentNullException* and display the error message.



Sequence Diagram of Authorization for Error Scenario ArgumentNullException

3.3 Error Handling ArgumentNullException

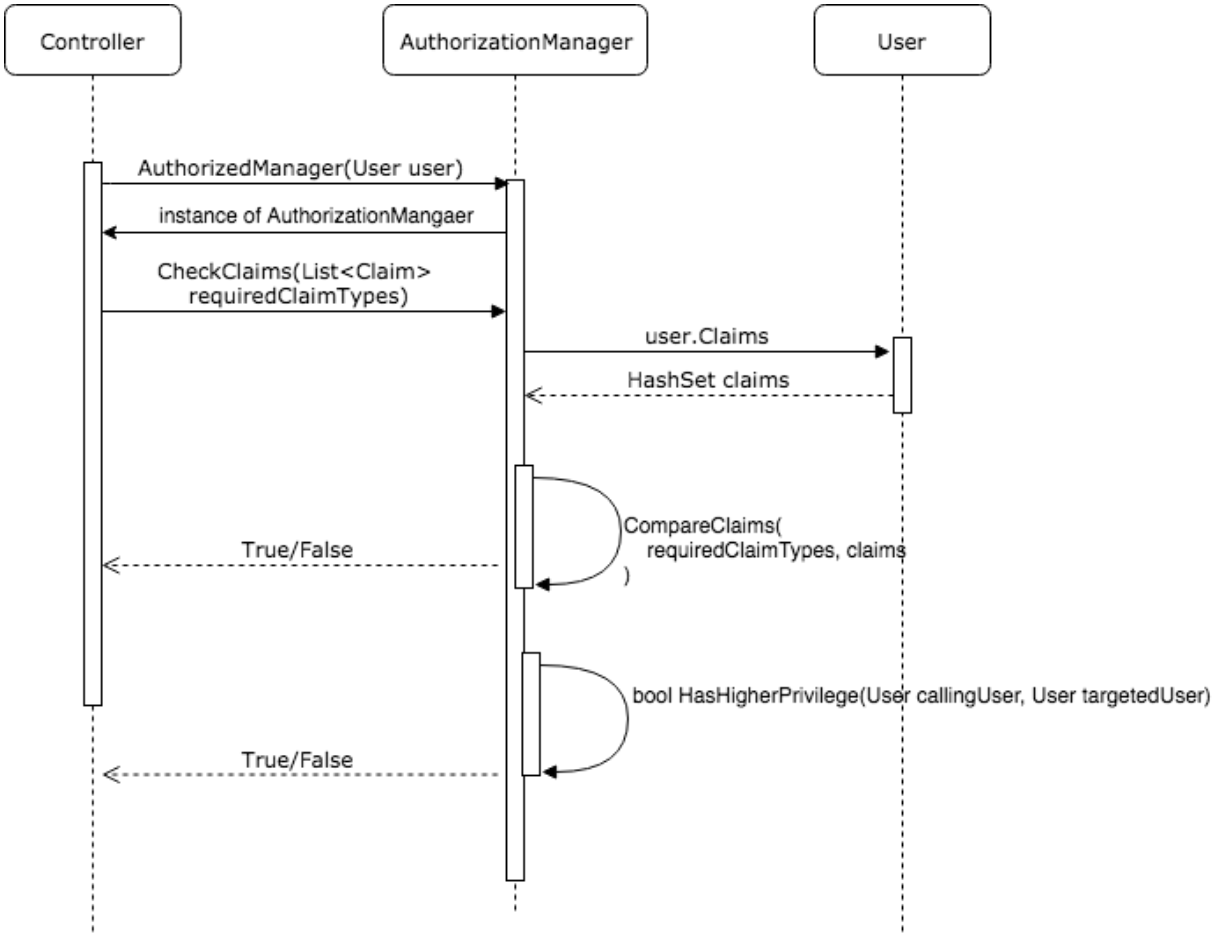
If the value of requiredClaim is null, CheckClaims() will check its value and throw the *ArgumentNullException*. If its value is null, CheckClaims() would not fulfill its role as intended. CheckClaims() is the essential method that determines an authorization level of users. Therefore, if it will not check any claims that user has, it would cause a critical security problem in the future. Thus, CheckClaims() does not allow a null for requiredClaimTypes and it will throw an exception.



Sequence Diagram of Authorization for Error Scenario ArgumentNullException

3.4 Low Level Diagram Deletion

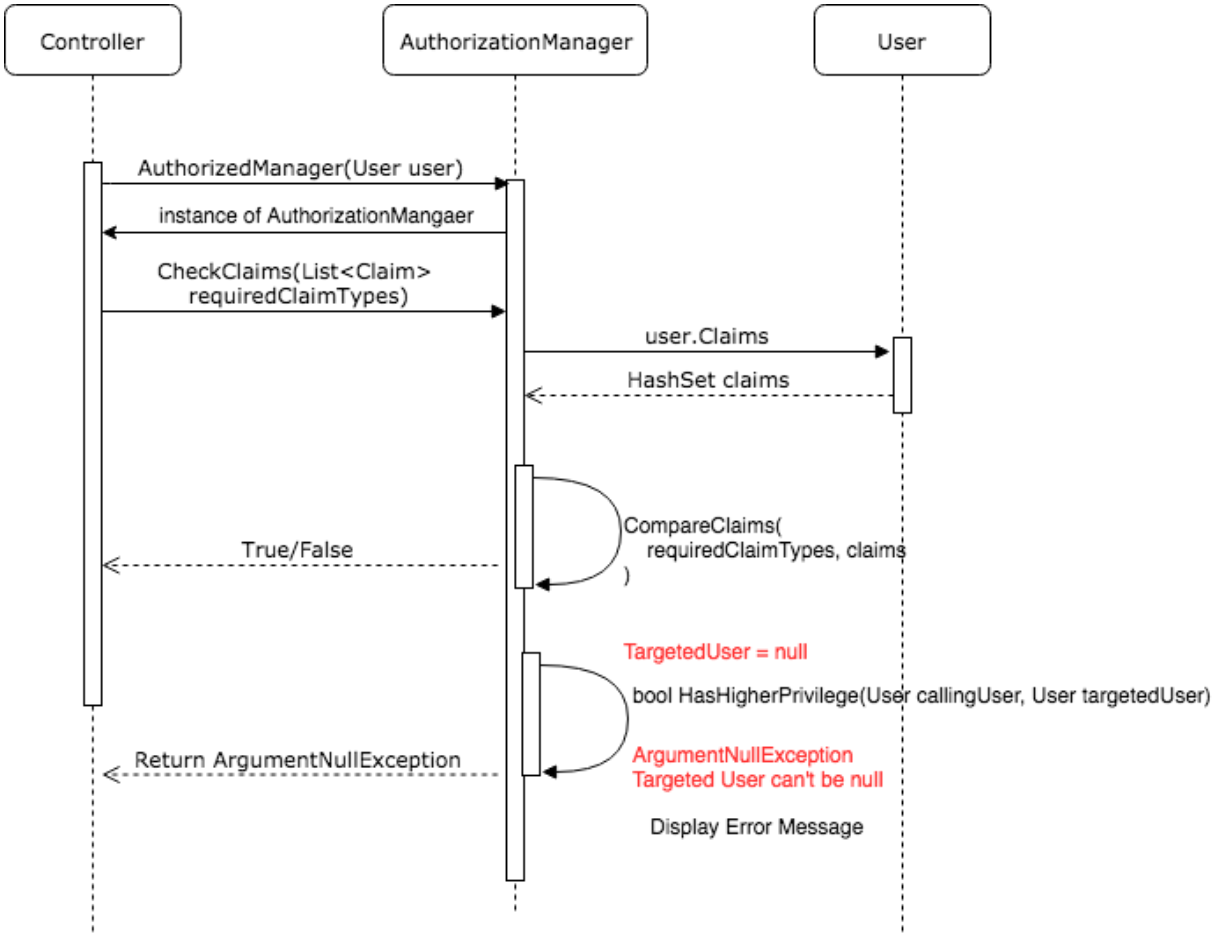
When a user requests to use deletion feature, AuthorizationManager will check the privilege of the user. If the calling user's level is higher than the targeted user or calling user's id is equal to targeted user's id, HasHigherPrivileage() would return true and the user would get a permission to use the deletion feature against the specific user.



Sequence Diagram of Authorization Deletion

3.5 Low Level Diagram Deletion ArgumentNullException

If the targeted user is null, HasHigherPrevileage() would throw an exception, because it would mean targeted user doesn't exist. The error message will inform the user that targeted user doesn’t exist.



Sequence Diagram of Authorization Deletion for Error Scenario ArgumentNullException

4. User Management

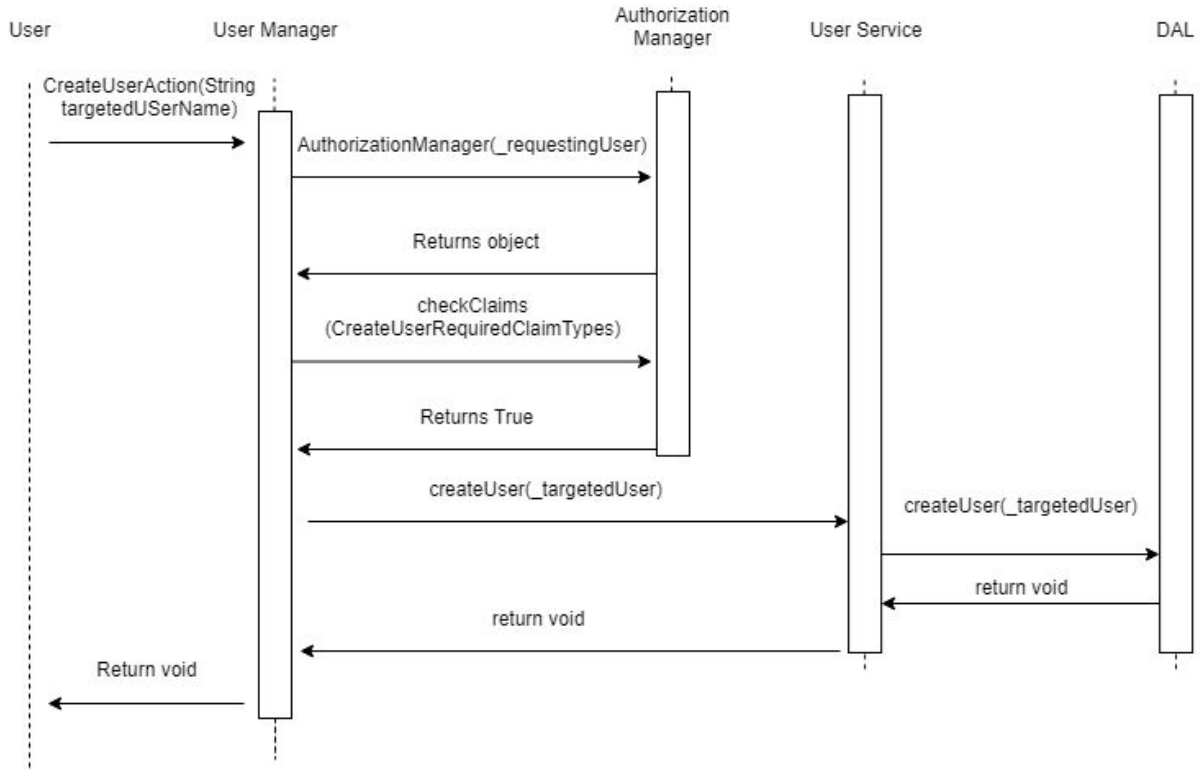
4.1 Registering User

This section is regarding to the registration of the user depending on whether an administrator is adding a user or if it is the user itself.

4.1.1 Low Level Diagram

The Register User diagram illustrates what occurs when a user, preferably an administrator or system administrator, registers another user. The program will run five

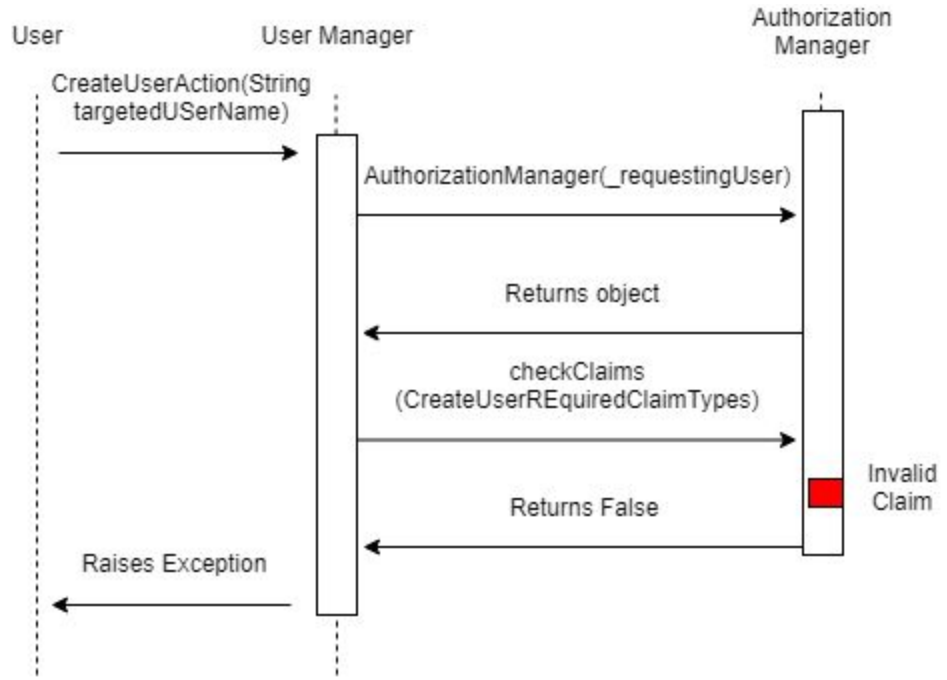
validations in order to validate the registration of another user. If a validation does not work during the process, then the validation will return back to the user manager as false, returning an error message to the user. If the validations are cleared and accepted, the user manager will send the user info into the user service, registering the user inside the data access layer. The validation will use the Password Checker feature in order to validate the password as unique for the user to keep.



Low Level Design of Registering Users within the User Management

4.1.2 Error Handling Invalid Claims

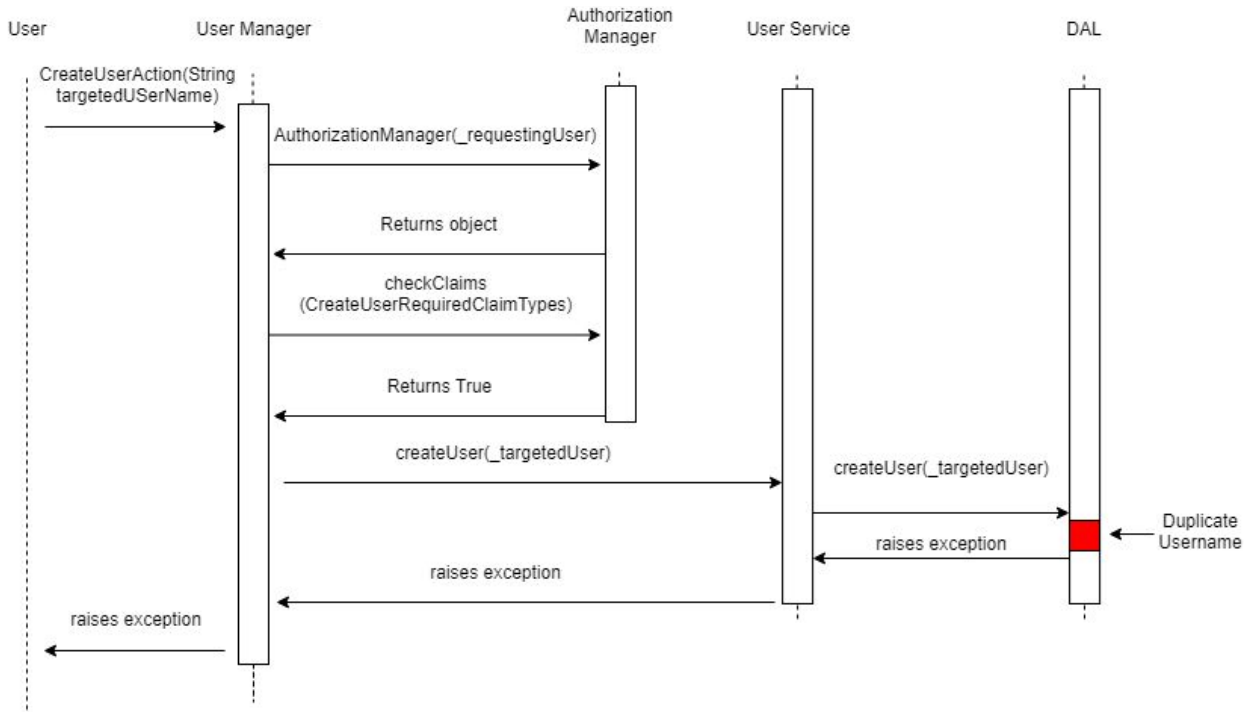
This error handling checks the privilege of the user. If the requesting user has the required claims in the list, then the user management services would create the user. Otherwise, an exception will be thrown.



Sequence Diagram of User Management Registering User for Error Scenario Duplicate Username

4.1.3 Error Handling Duplicate Username

This error handling checks the username, which is the email of the user, in order to see whether it is unique within the Data Access Layer. If the user's username is invalid, the DAL will throw an exception and an error message will be displayed to the user.



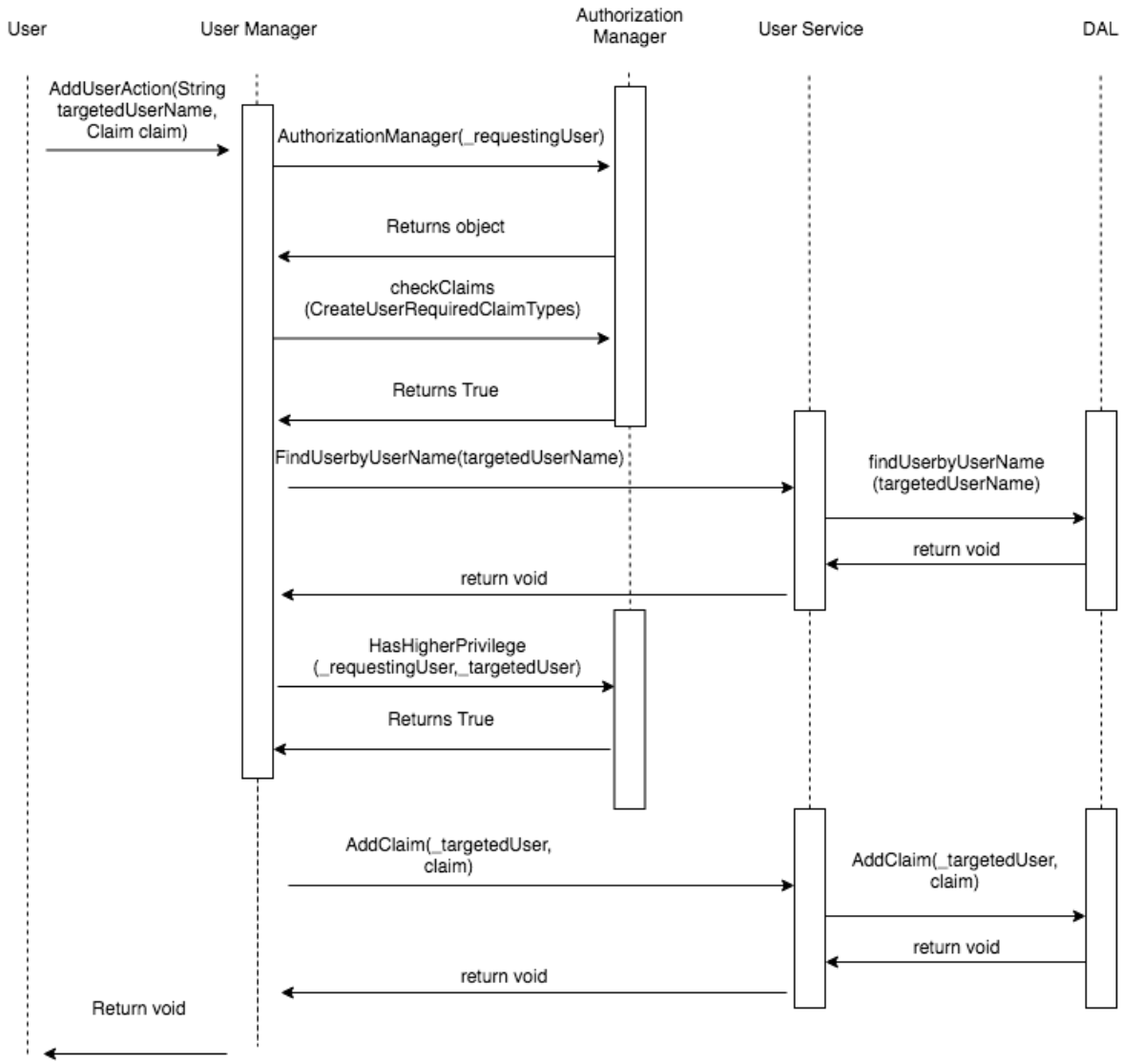
Sequence Diagram of User Management Registering User for Error Scenario Invalid User Name

4.2 Add Claim to User

This section is regarding to adding a claim to the user in order to give them more permissions.

4.2.1 Low Level Diagram

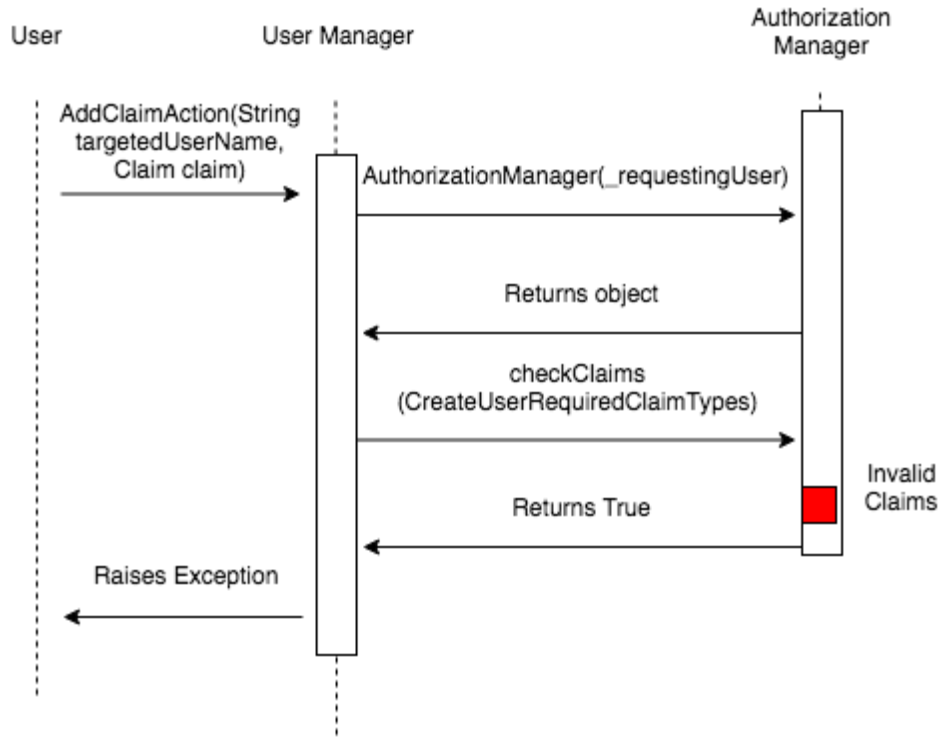
The Add Claim diagram will add a claim to the user and give them permissions to add users.



Sequence Diagram of User Management Adding Claim to User

4.2.2 Error Handling Invalid Claim

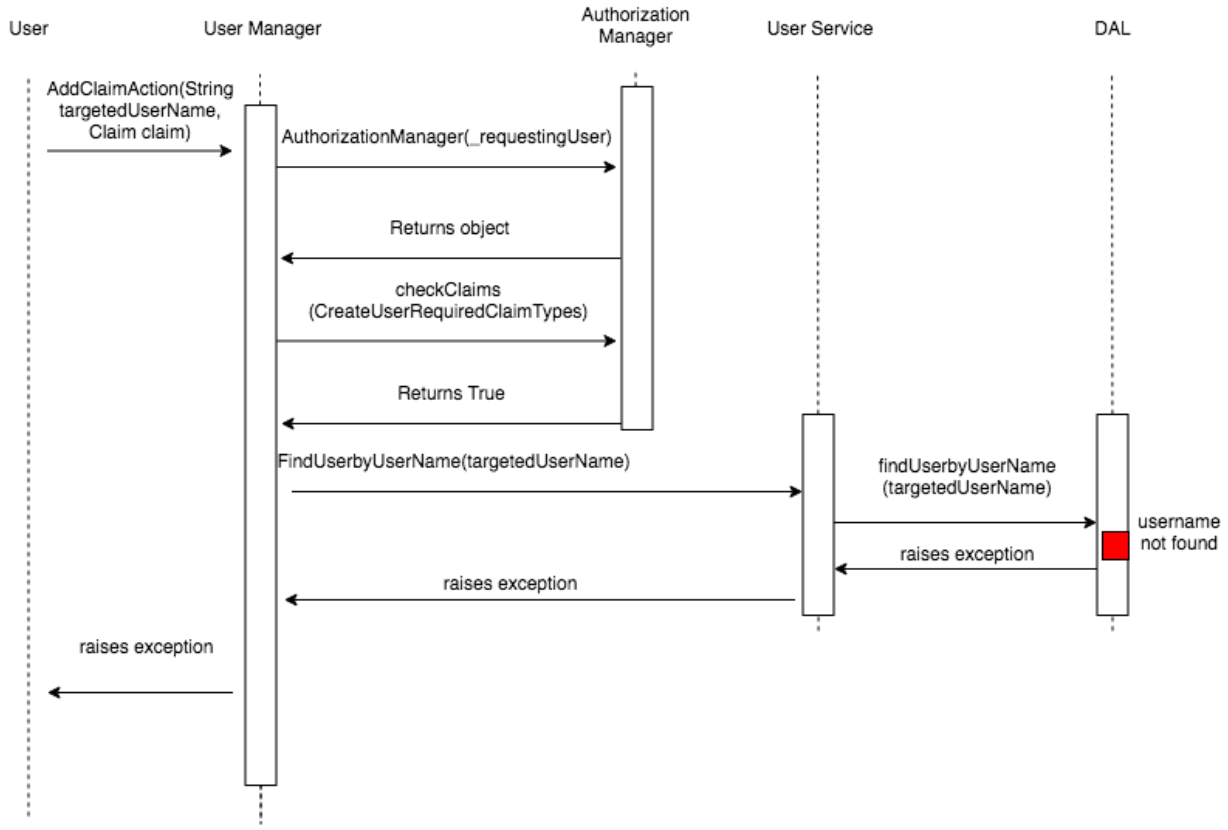
Authorization Manager checks that user contains the required claim in the list. If the user contains the claim, the user would get a permission to add the claim to the user. Otherwise, the user's request would be denied.



Sequence Diagram of User Management Adding Claim to User for Error Scenario Invalid Claim

4.2.3 Error Handling Username Not found

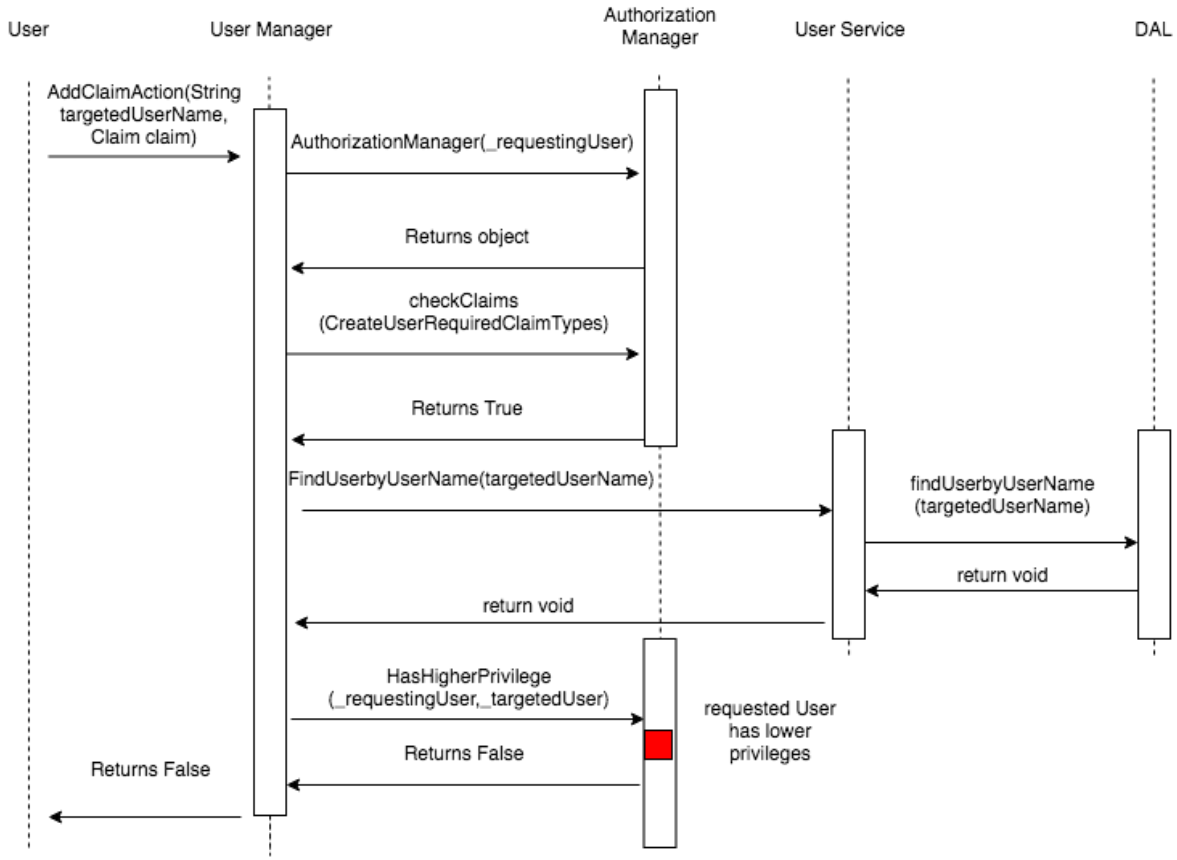
DAL checks the existence of the username of the user. If it's not found, it would throw an exception and an error message will be displayed to the user. If it's found, it would continue to add the claim to the user.



Sequence Diagram of User Management Adding Claim to User for Error Scenario Username Not Found

4.2.4 Error Handling Lower Privileges

If the user's id is lower than calling user's id, it would throw an exception. Otherwise, it will continue to add the claim to the user.



Sequence Diagram of User Management Adding Claim to User for Error Scenario Lower Privilege

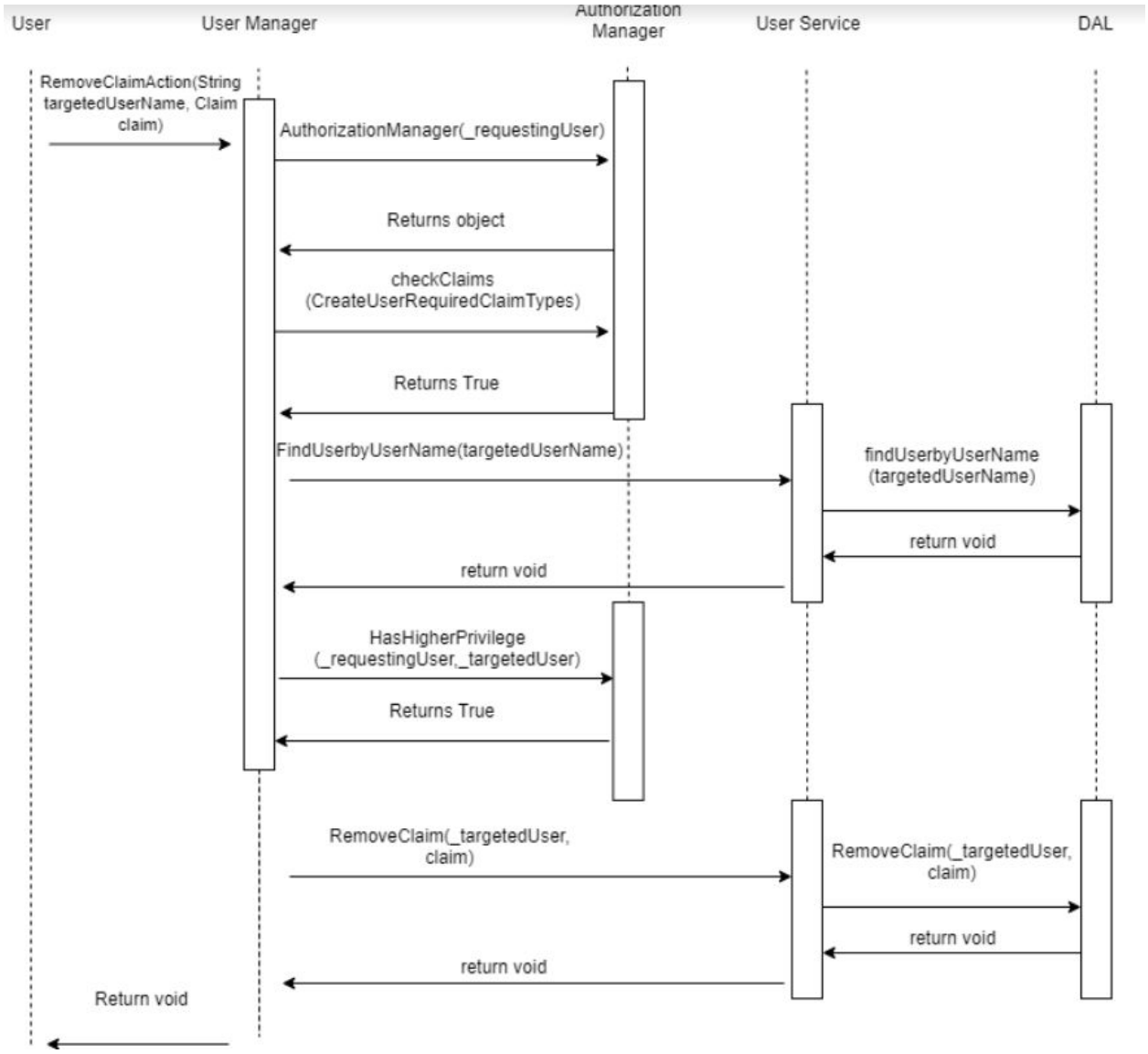
4.3 Remove Claim from User

This section is regarding to removing the user's claim and their permissions the claim offered.

4.3.1 Low Level Diagram

The Remove Claim diagram will check if the user exists within the Database. Once it does, it will check the privilege of the requesting user who is changing the claims of another

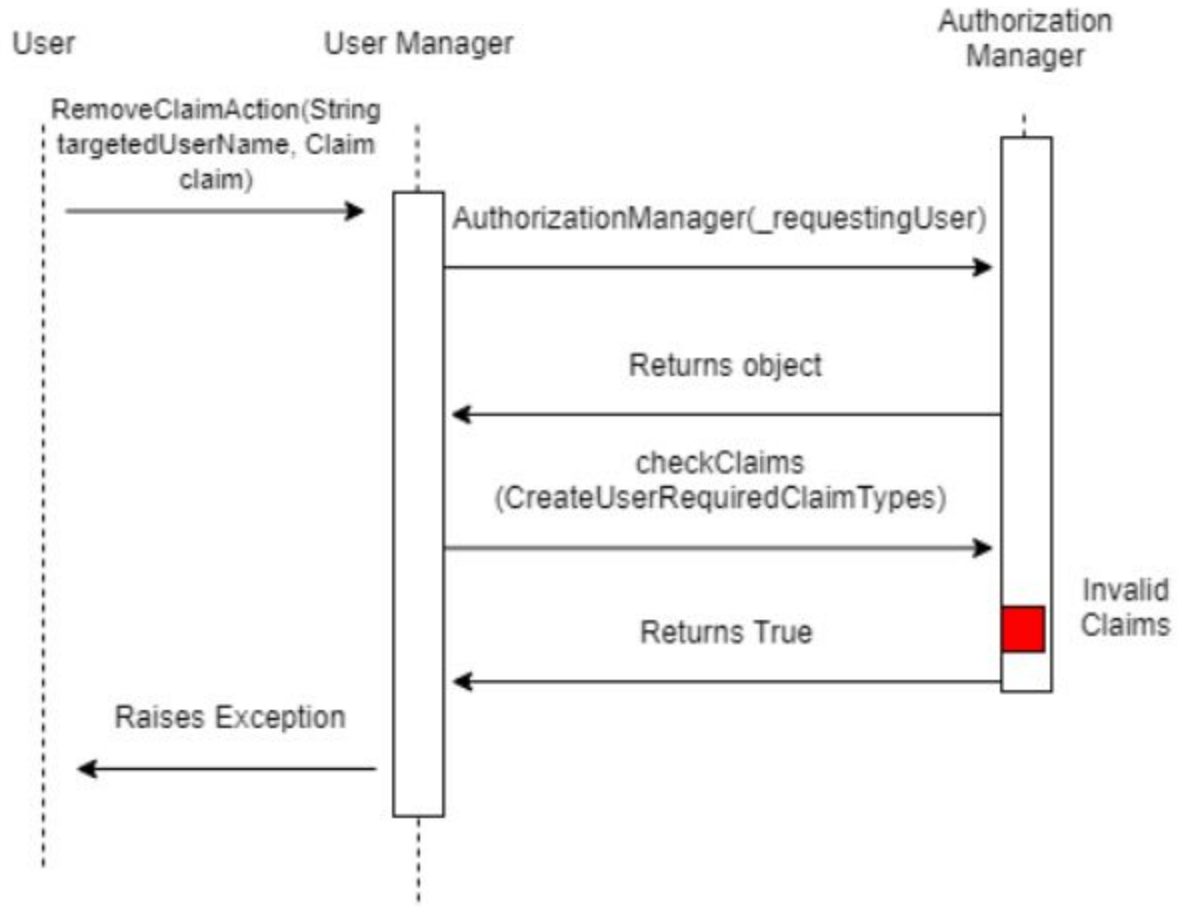
user. If the considered action is true, the targeted user will have their claim removed from the user management services.



Sequence Diagram of User Management Removing Claim to User

4.3.2 Error Handling Invalid Claim

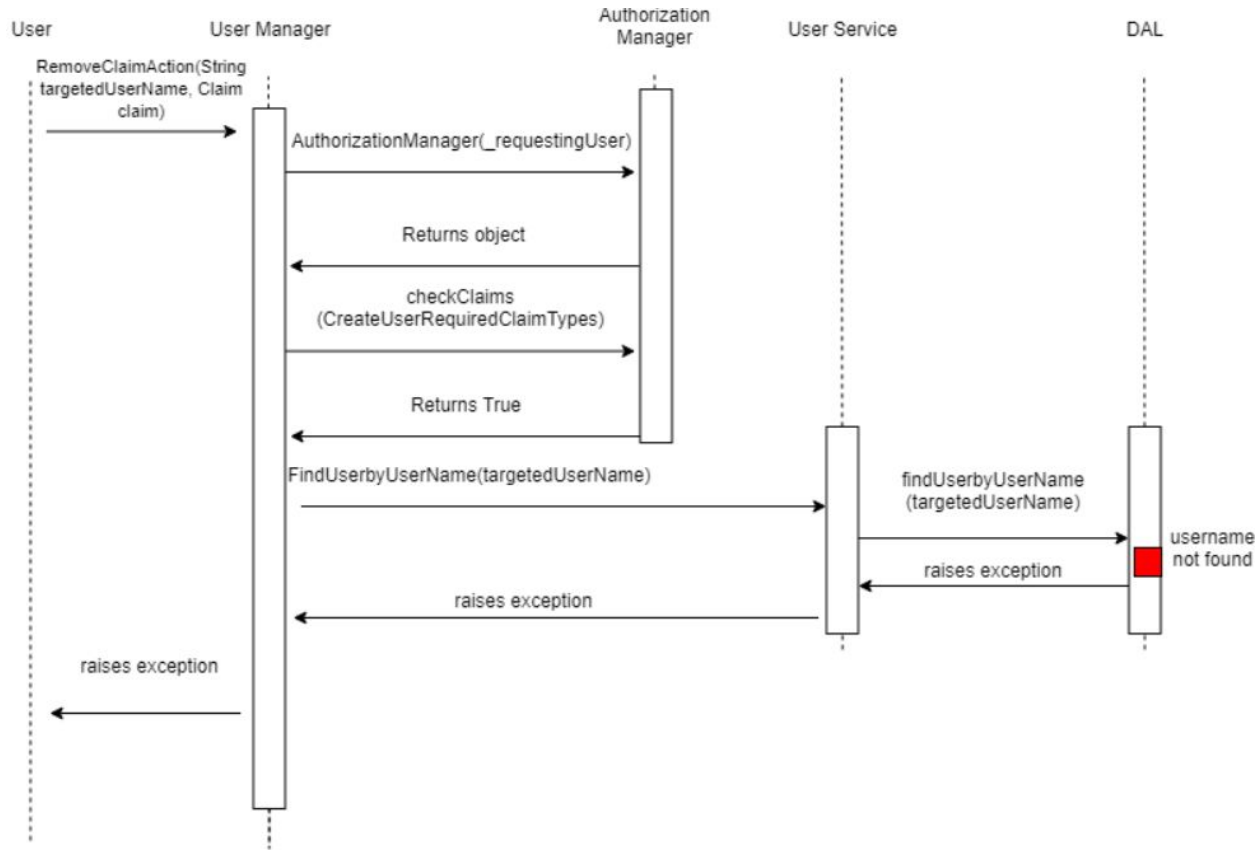
The error handling diagram covers the issue where the user does not have the claims necessary to remove a claim from another user. If the user does not have the necessary claims, the authorization manager will return false.



Sequence Diagram of User Management Removing Claim to User for Error scenario regarding invalid claims

4.3.3 Error Handling Invalid Username

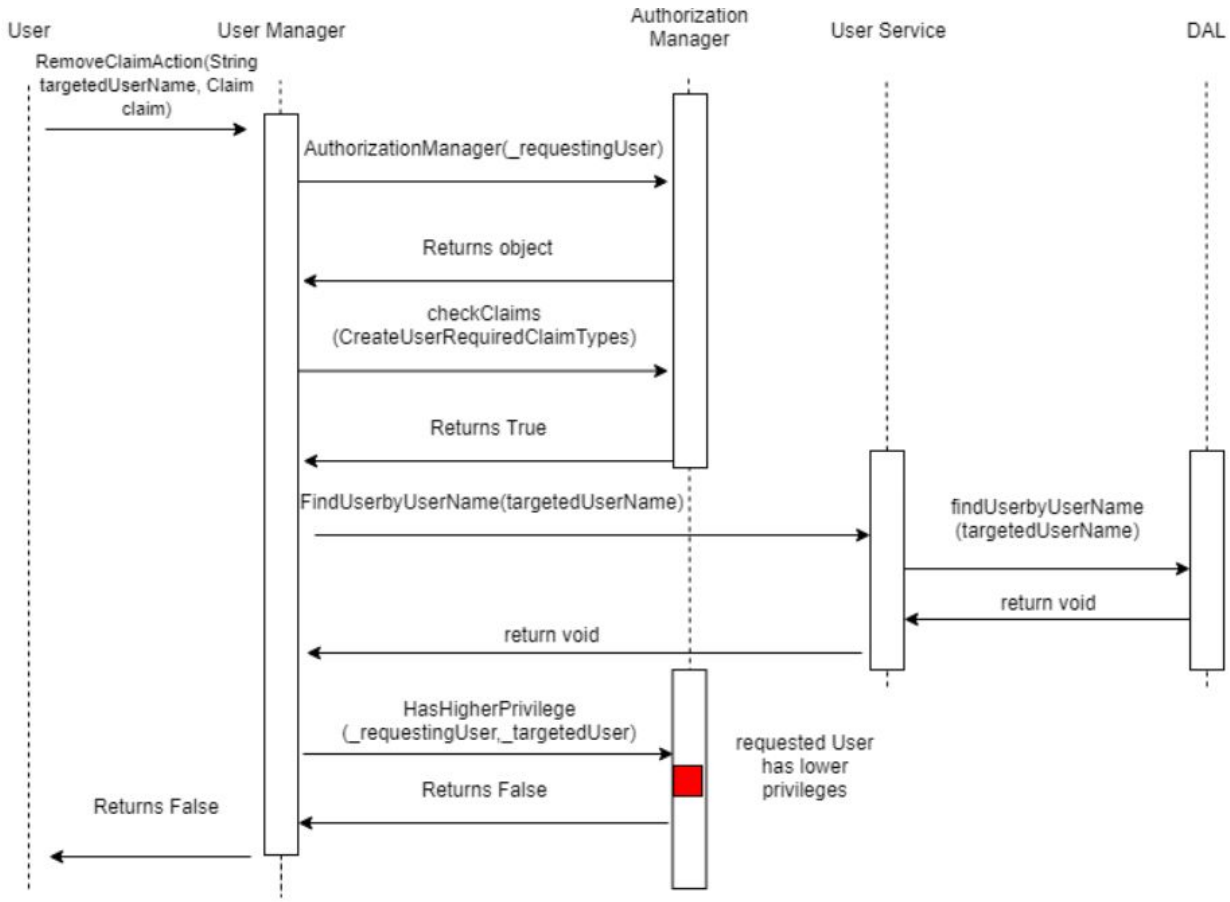
The error handling diagram covers the problems that a user can face if its not found inside the database. This is necessary in case unregistered users try to use our services. By not having a username found in the database, the program will raise an exception.



Sequence Diagram of User Management Removing Claim to User for ERror scenario in invalid usernames

4.3.4 Error Handling Lower Privileges

This error handling diagram covers the issues that a requesting user can face if they have a lower claim than the targeted user. If the statement is true, then the authorization manager will return false and throw an exception.



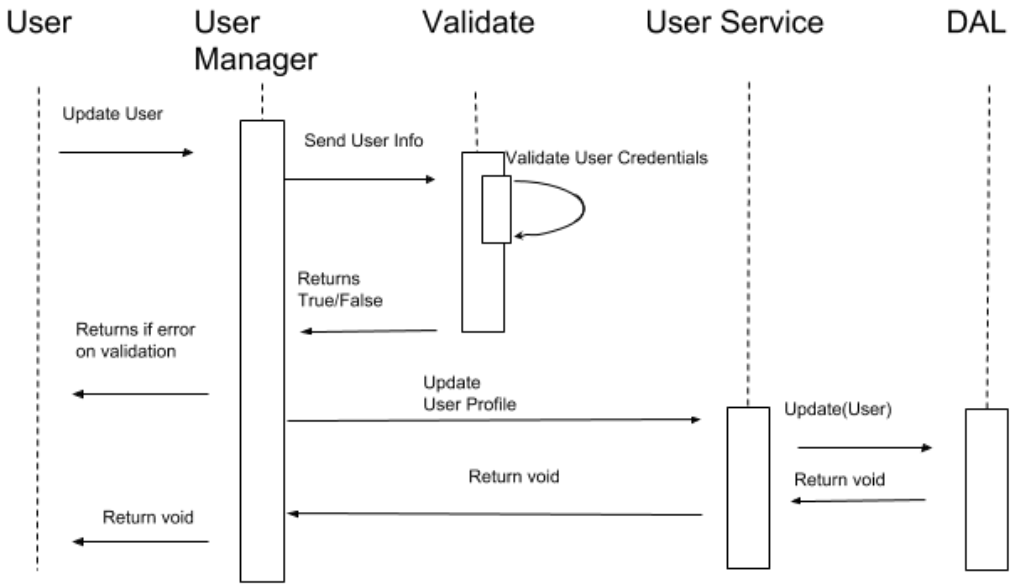
Sequence Diagram of User Management Removing Claim to User for Error Scenario Regarding Lower Privileges

4.4 Update User

This section is regarding to updating the user's information based on either the administrator's decision or the user itself.

4.4.1 Low Level Diagram

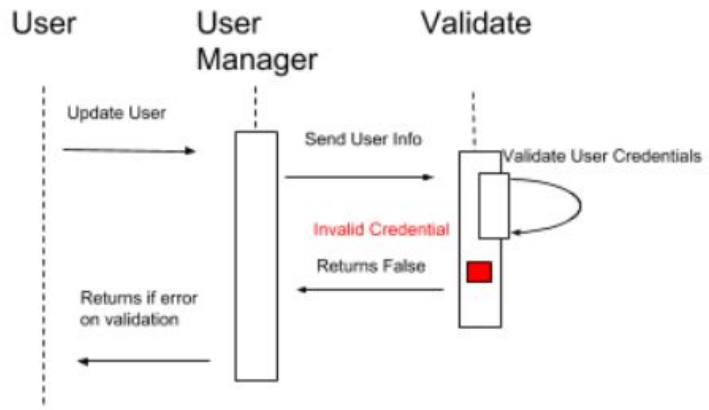
The Update User diagram is one of the functions that edits a user's access to the website. This is mainly used for users who require to update their information or for administrators to update someone else.



Enable User Diagram

4.4.2 Error Handling User Claim

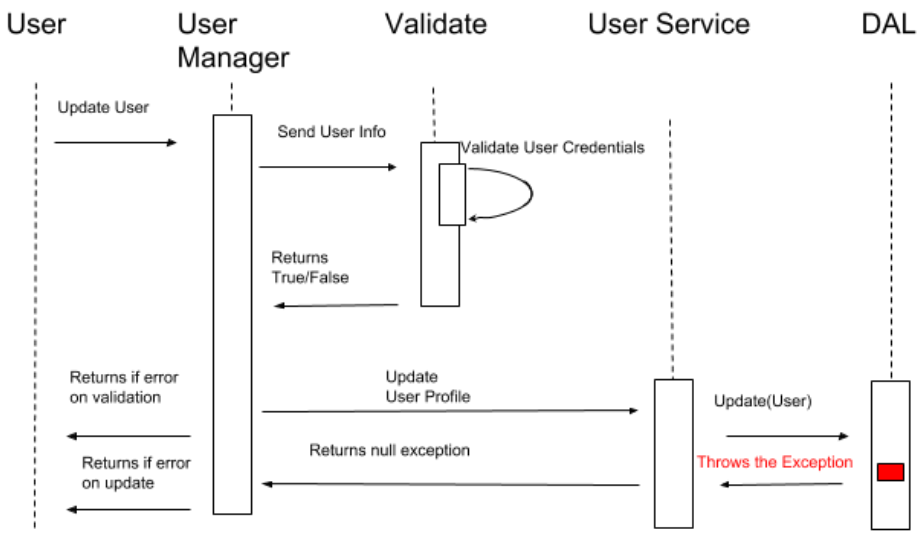
This error handling covers the validation of the user claim in order to make sure the user who is being enabled does not have the same claims as the user itself. If the users have the same claims, the validation will return false and the user would receive an error.



Sequence Diagram of User Management Update User for Error Scenario User Claims

4.4.3 Error Handling Incorrect Data

This error handling will throw an exception if certain information is incorrect during the updating process. Once an exception is thrown, the user will receive a error regarding the failed update.



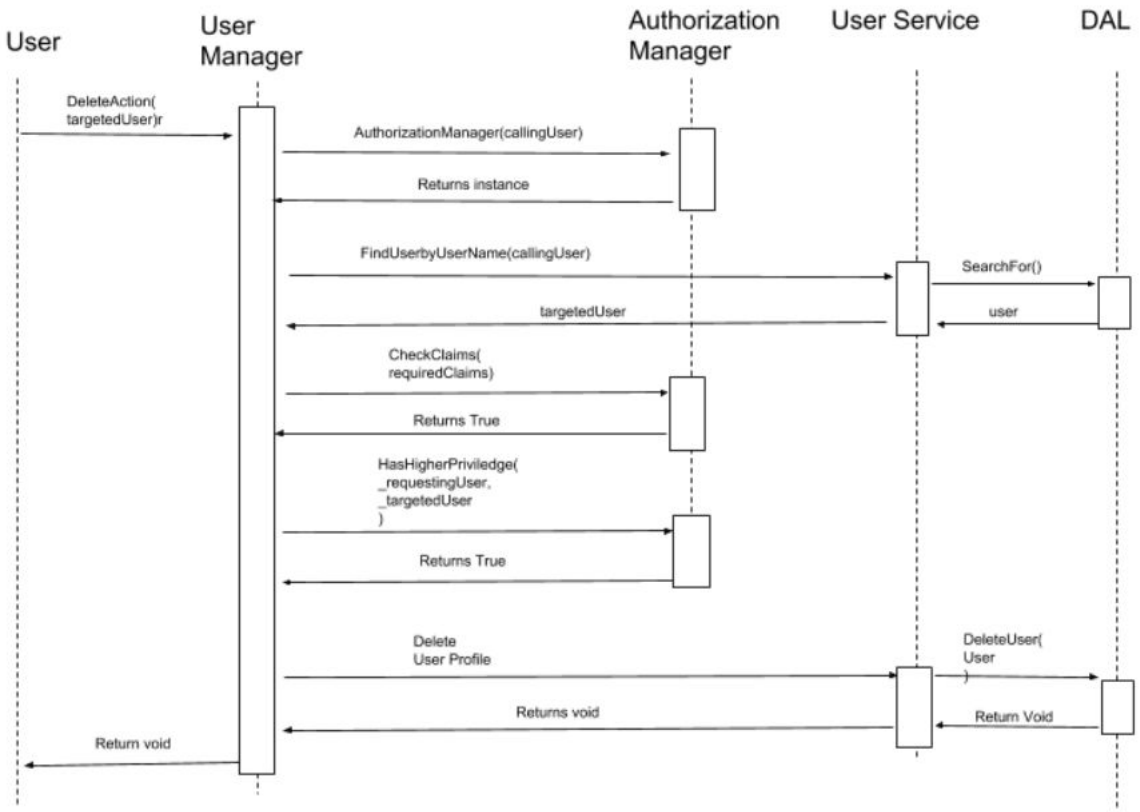
Sequence Diagram of User Management Update User for Error Scenario Incorrect Data

4.5 Delete User

This section is the information for the user's ability to delete themselves. This is also for administrators who need to delete another user.

4.5.1 Low Level Diagram

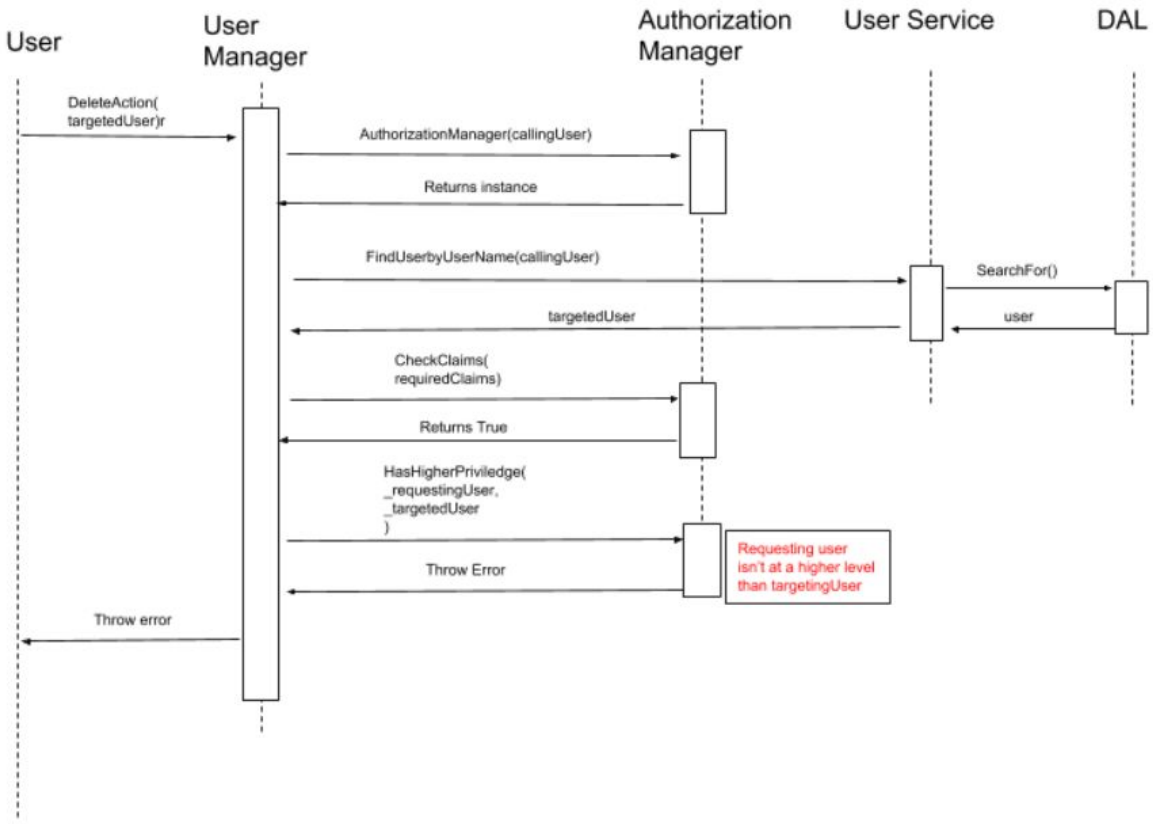
The Delete User diagram reveals the different functions the users can use. Users can delete themselves as they wish and only system administrators and administrators can delete users if they are not on the same level as them. Those who are in the same level as the user that is deleting them will end the delete function and return an error.



Delete User Diagram

4.5.2 Error Handling User Claims

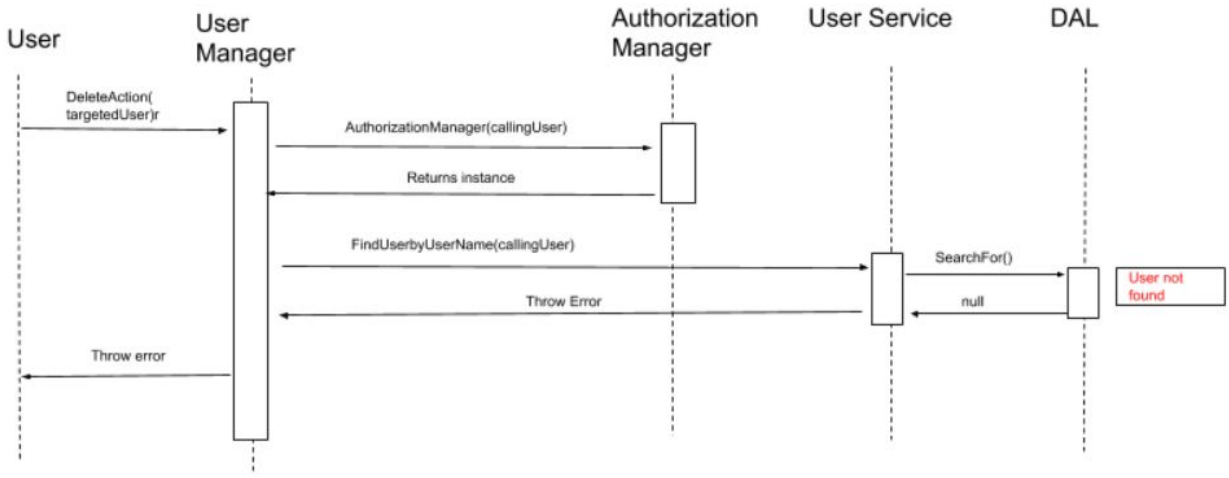
This error handling is similar to the update function as the user that is about to be deleted must have the right claims. This is because there is a possibility that the user that is about to get deleted has claimsthat is equal or higher than the one who is deleting. That is why there is a claim check to make sure that the person who is about to be deleted is either a student or a non-student and not the student itself.



Sequence Diagram of User Management Delete User for Error Scenario User Claims

4.5.3 Error Handling User Identity

This error handling checks the user's information in order to make sure the exact user is being deleted properly. If it does not match with the user list inside the Data Access Layer, then it will throw an exception.



Sequence Diagram of User Management Delete User for Error Scenario User Identity