CECS 491A - Sec 6 - Low Level Design Document

Project Name: ArrowNav

Team Longhorn:

Brayan Fuentes

Christian Lucatero

Curtis Nishihira

Miguel Zavala

Spencer Gravel (Team Leader)

December 15, 2021

Table of Contents

1. Introduction

This low level design document is intended to demonstrate how data is handled when input into the system, specifically with regard to user management. Furthermore, any error handling will be explained with its potential solutions.

2. User Management

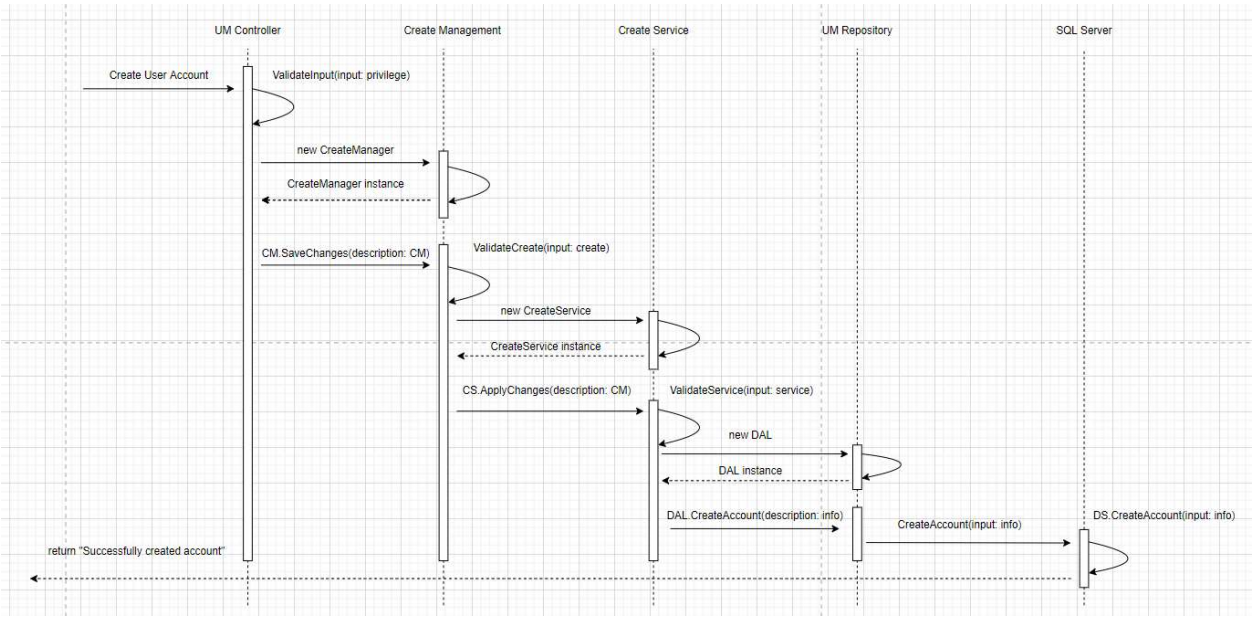
In order to use the application, a user must have an account that has a username and password. The user management aspects of the application allows the user to create, delete, and/or update their account. Furthermore, user management features will allow the developers to have the same abilities that users do, along with enabling and disabling user accounts.

2.1 Creating User Account

This section will outline the registration of a user in the application, which is done by the user itself when registering a unique Username along with a password.

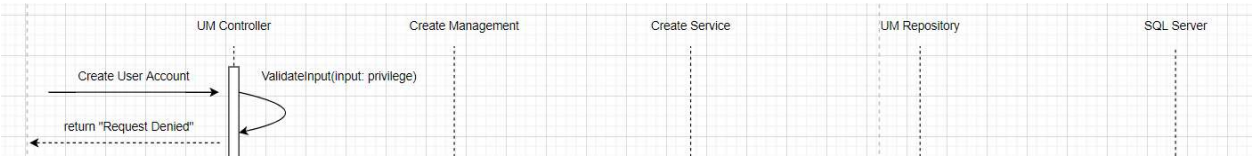
2.1.1 Low Level Diagram

This diagram shows when a user creates a new user account with the program running through a series of validations in order to complete the process of creating a new user account. The user's request for creating an account has to go through the input validation in the Controller which checks the user's privileges to determine if they are allowed to create an account. An instance of CreateManager is then created and has to go through validation in Management. The Manager then creates a Service which is validated in the service layer and the Service applies the changes made to the account, in this case creating one, and then creates an instance of the Data Access Layer which is validated and then applies the changes to the Data Store. If all validations go through and no exceptions are raised then the user will have successfully created an account.

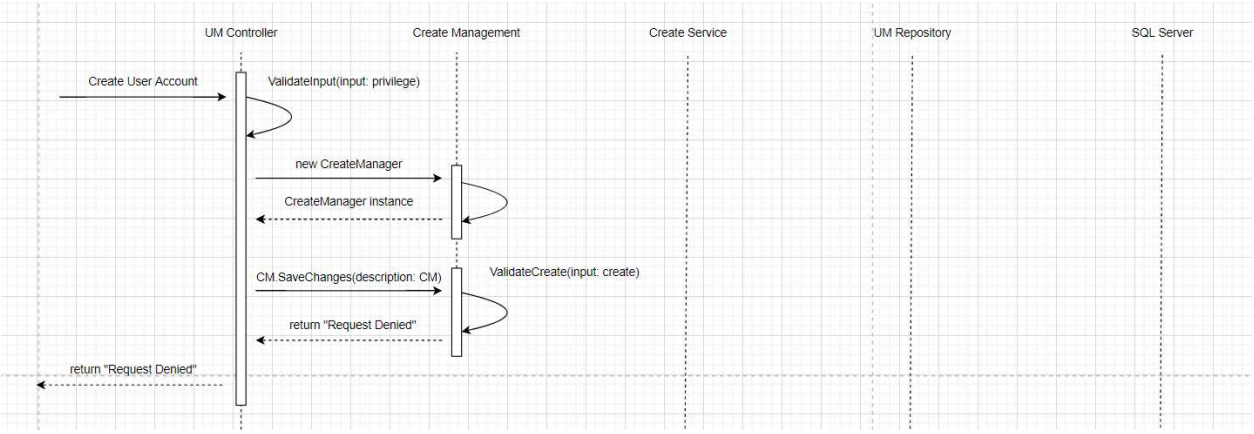


2.1.2 Case Handling

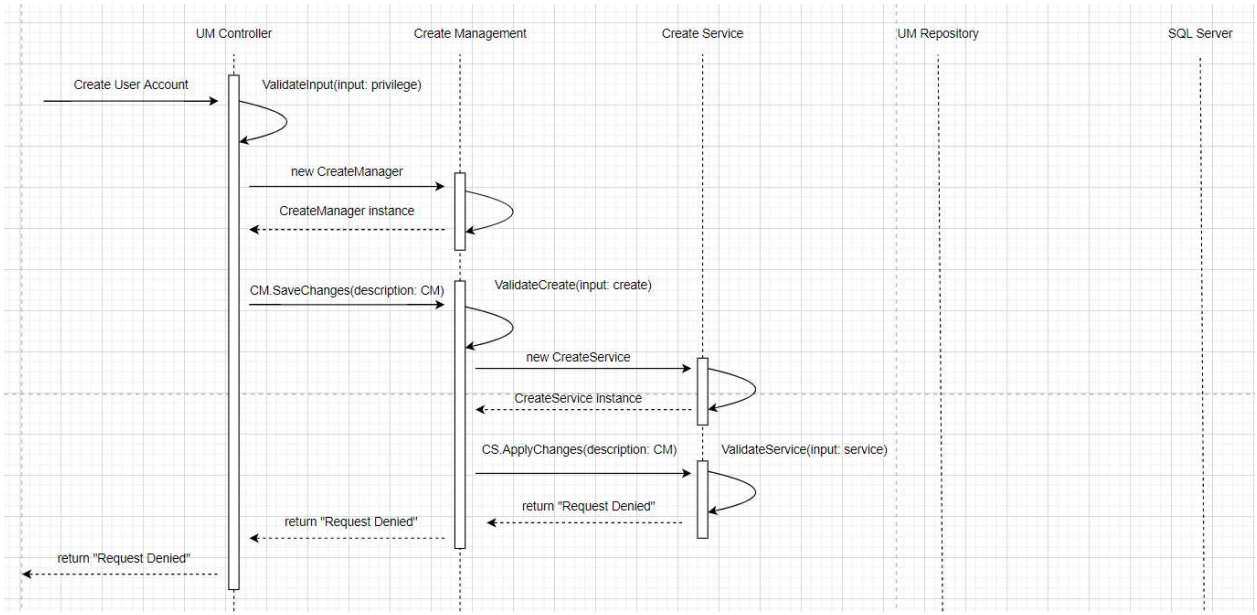
This diagram shows when the user does not have the privilege for creating an account causing his request for creating an account failing the validation in the controller.



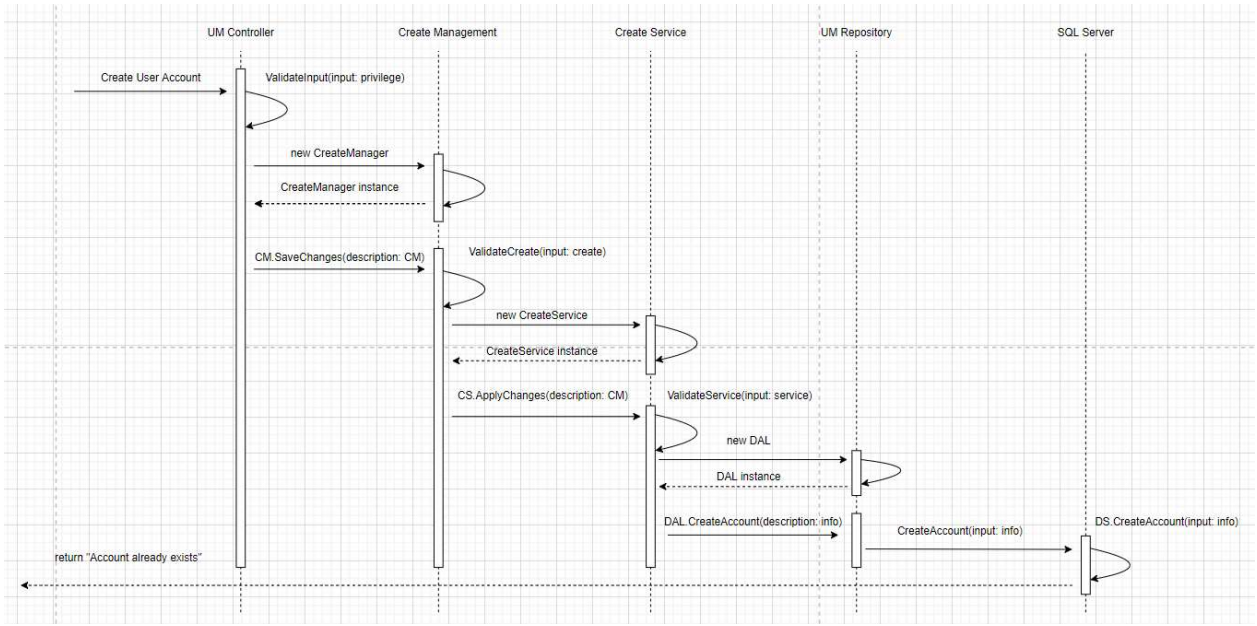
This diagram shows when the instance of CreateManager is not called due to it failing the CreateManagement validation.



This diagram shows when the instance from CreateManagement is not validated when requesting service from CreateService.



This diagram shows the error case when a user tries to create an account and passes through the Controller Validation, Management Validation and the Service Validation but the username of the account that the user is trying to create already exists in the Data Store.

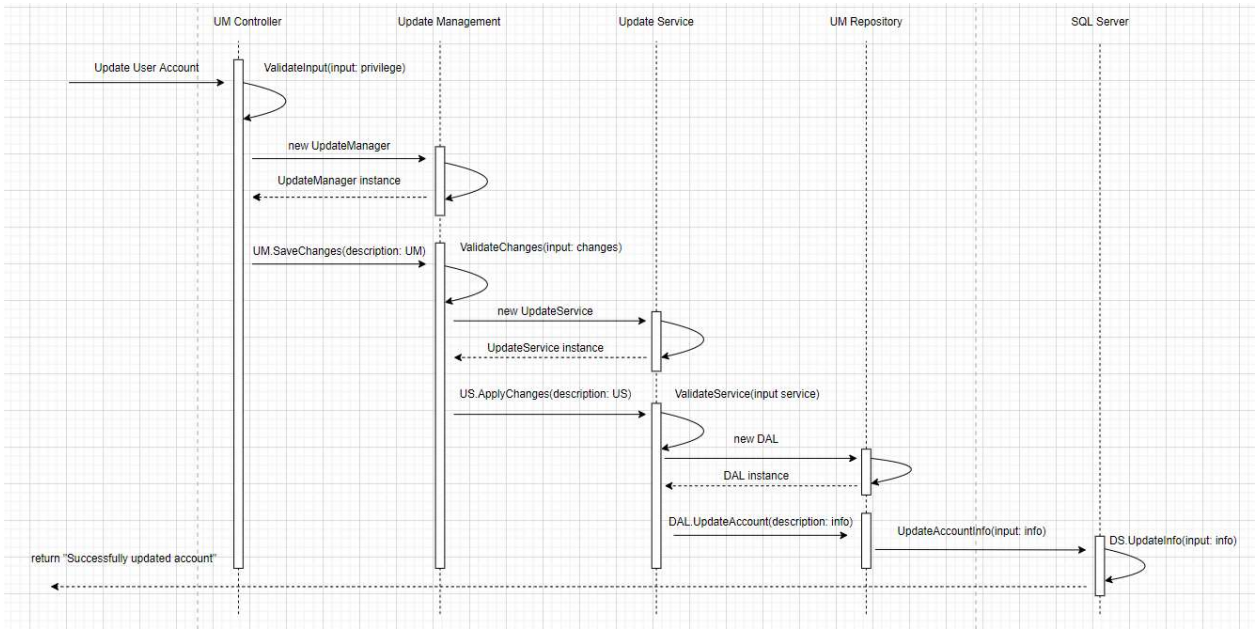


2.2 Updating User Account

This section outlines the process of updating existing account information. These updates include changes to account parameters such as the username, password, or privileges assigned to an account.

2.2.1 Low Level Diagram

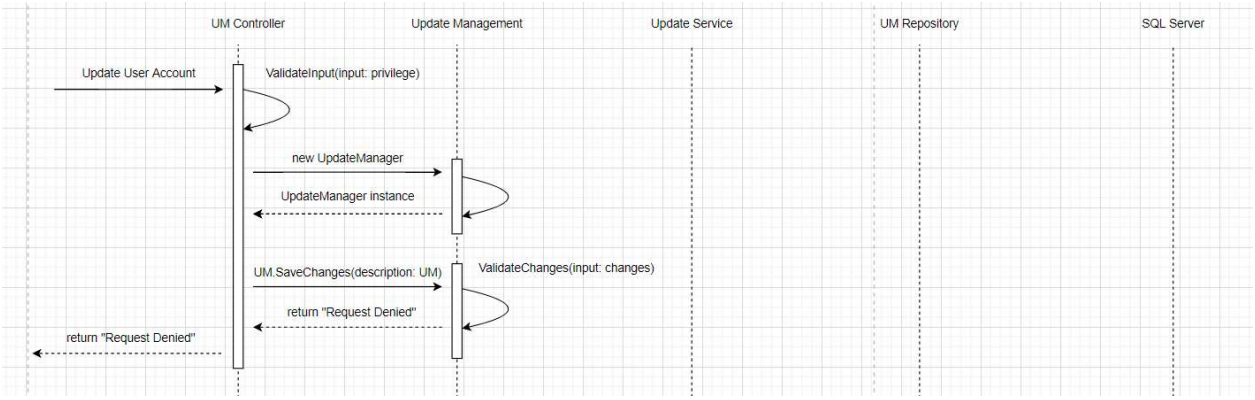
This diagram shows when a user updates a user account with new information with the program running through a series of validations in order to complete the process of updating. The user's request for updating an account has to go through the input validation in the Controller which checks the user's privileges to determine if they are allowed to update an account. An instance of UpdateManager is then created and has to go through validation in Management. The Manager then creates a Service which is validated in the service layer and the Service applies the changes made to the account and creates an instance of the Data Access Layer which is validated and applies the changes to the Data Store. If all validations go through and no exceptions are raised then the user will have successfully updated the account.



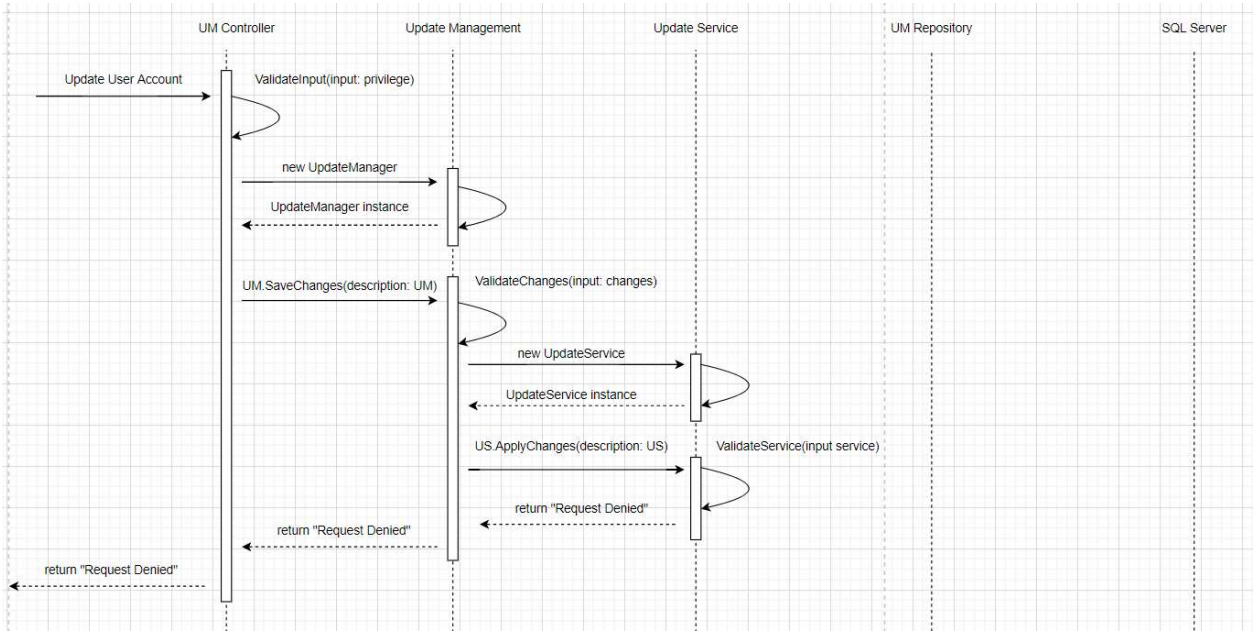
2.2.2 Case Handling

This diagram shows when the user does not have the privilege for updating an account causing his request for updating an account failing the validation in the controller.

This diagram shows when the instance of UpdateManager is not called due to it failing the UpdateManagement validation.



This diagram shows when the instance from UpdateManagement is not validated when requesting service from UpdateService.

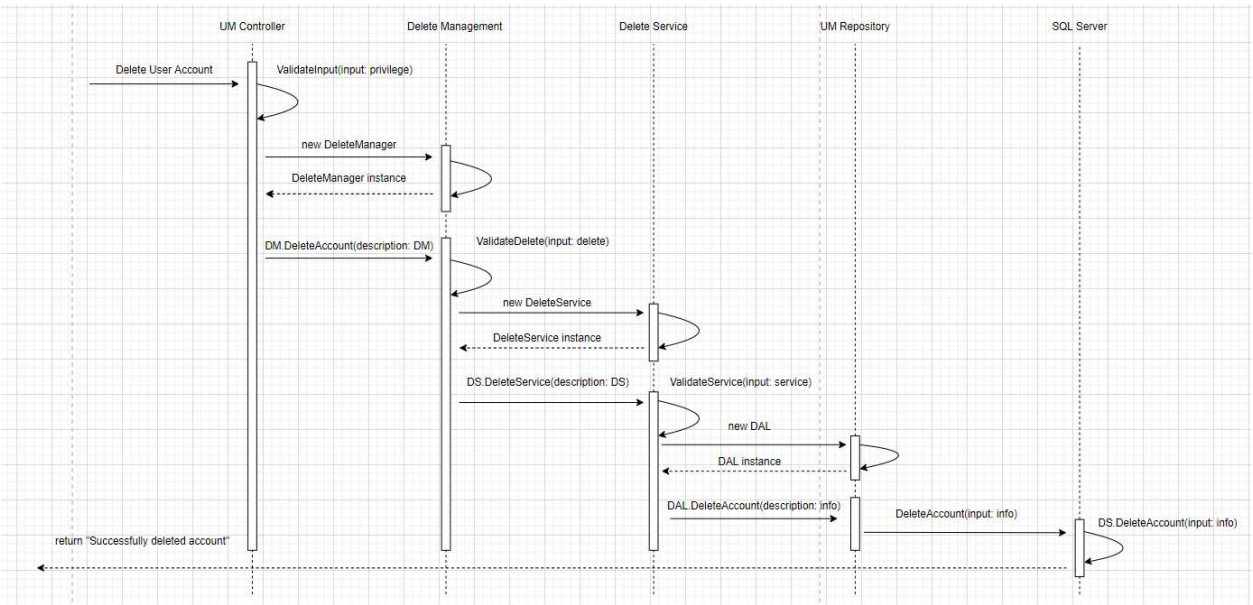


2.3 Deleting User Account

This section outlines the process of deleting a user account from the application's database. When selecting the account, all associated information with the account will be deleted.

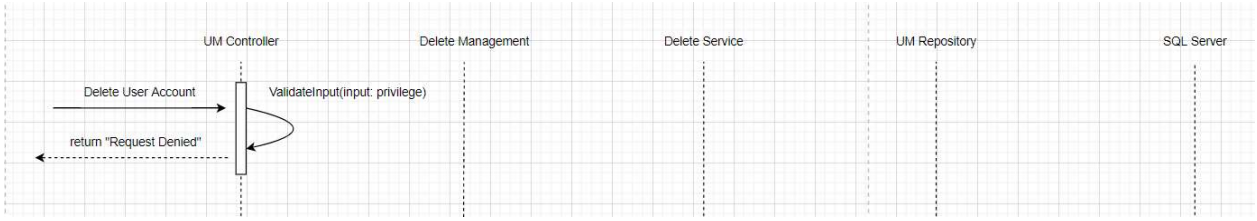
2.3.1 Low Level Diagram

This diagram shows when a user deletes a user account with the program running through a series of validations in order to complete the process of deleting a user account. The user's request for deleting an account has to go through the input validation in the Controller which checks the user's privileges to determine if they are allowed to delete an account. An instance of DeleteManager is then created and has to go through validation in Management. The Manager then creates a Service which is validated in the service layer and the Service applies the changes made to the account, in this case deleting the account altogether, and then creates an instance of the Data Access Layer which is validated and then applies the changes to the Data Store. If all validations go through and no exceptions are raised throughout the process then the user will have successfully deleted the account.

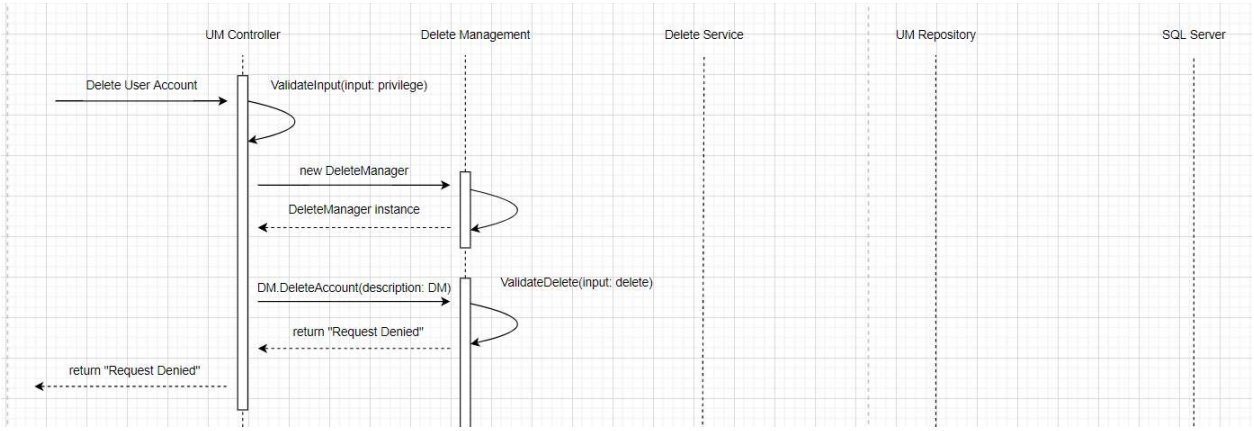


2.3.2 Case Handling

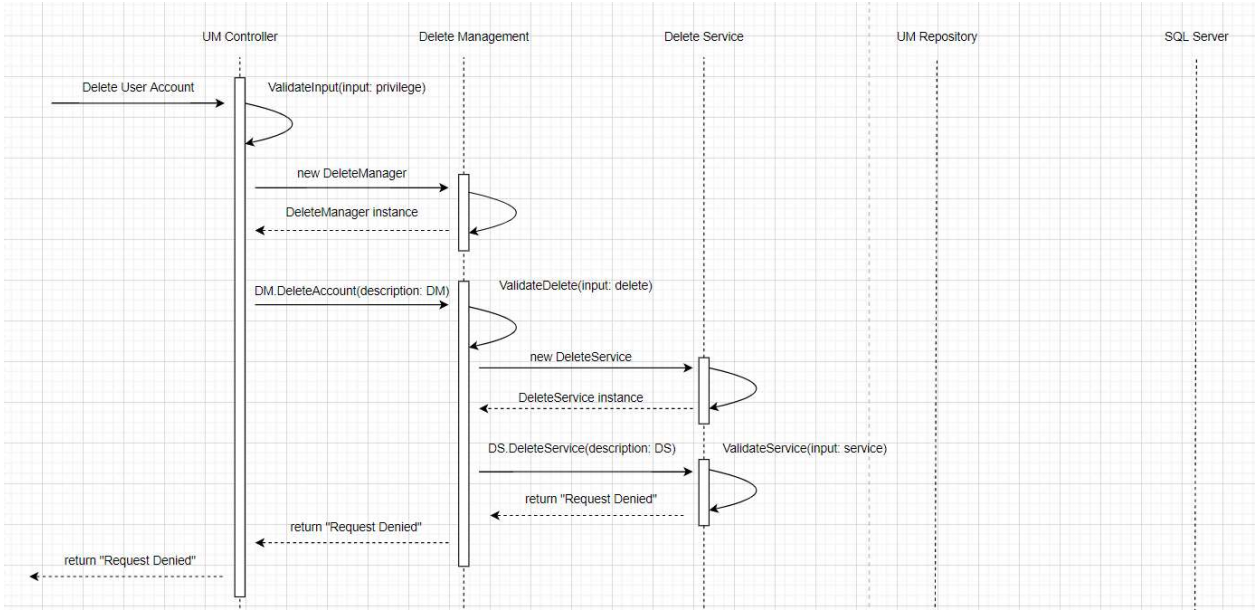
This diagram shows when the user does not have the privilege for deleting an account causing his request for deleting an account failing the validation in the controller.



This diagram shows when the instance of DeleteManager is not called due to it failing the DeleteManagement validation.



This diagram shows when the instance from DeleteManagement is not validated when requesting service from DeleteService.

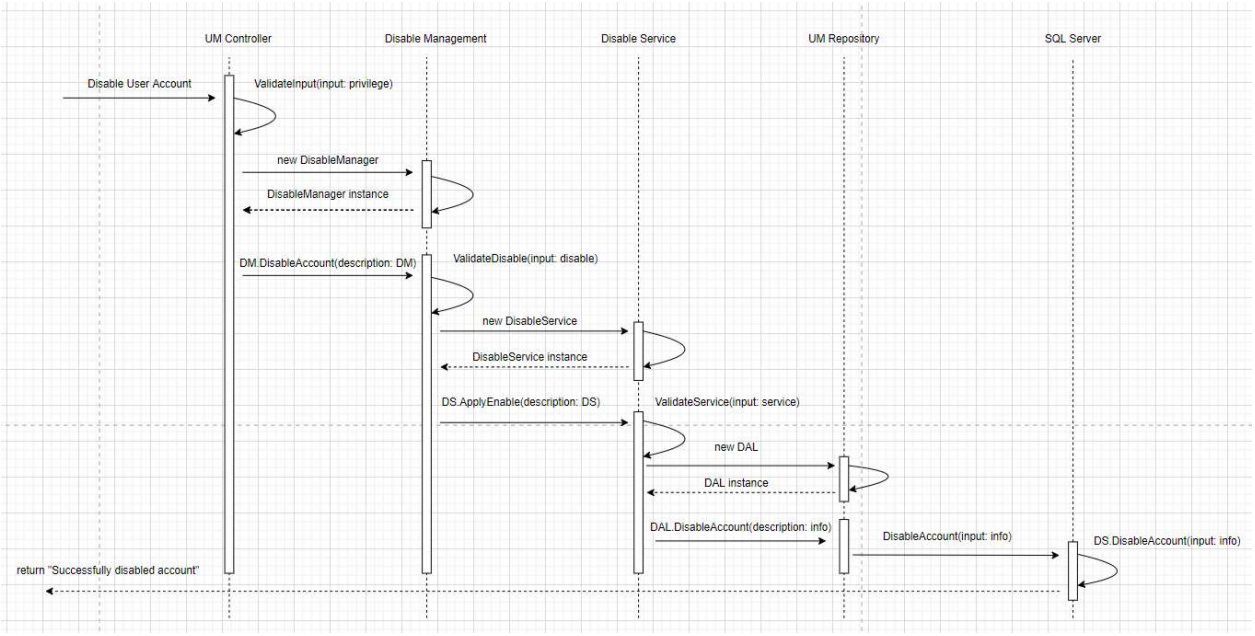


2.4 Disabling User Account

This section outlines how a user account is disabled or given less privileges in a database. This can be carried out by an administrator or someone with privileges that allows for account status to be altered.

2.4.1 Low Level Diagram

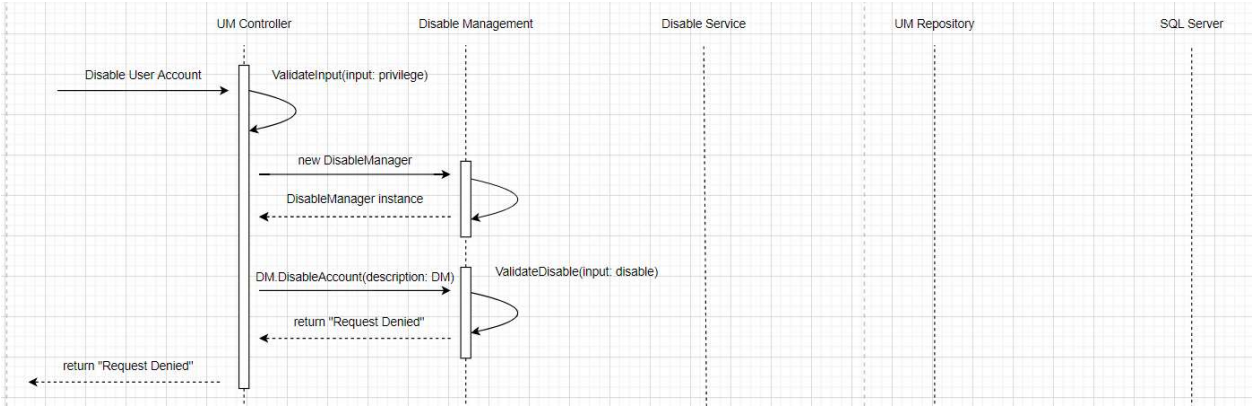
This diagram shows when a user disables a user account with the program running through a series of validations in order to complete the process of disabling an account. The user's request for disabling an account has to go through the input validation in the Controller which checks the user's privileges to determine if they are allowed to disable an account. An instance of DisableManager is then created and has to go through validation in Management. The Manager then creates a Service which is validated in the service layer and the Service applies the changes made to the account, in this case creating one, and then creates an instance of the Data Access Layer which is validated and then applies the changes to the Data Store. If all validations go through and no exceptions are raised then the user will have successfully created an account.



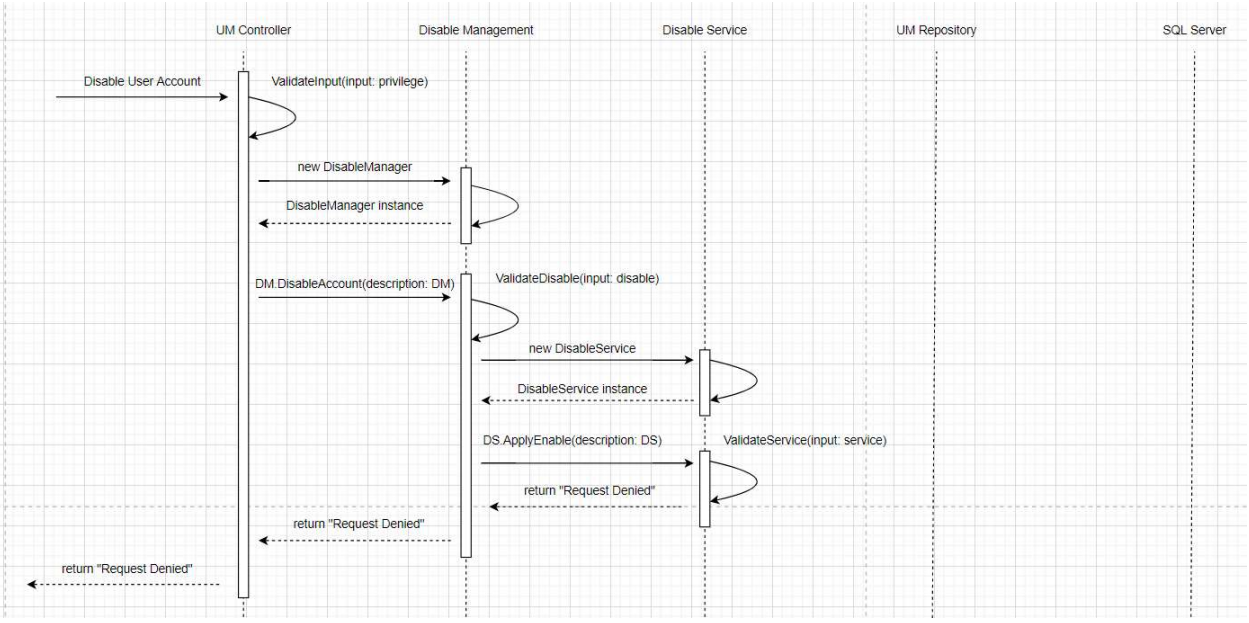
2.4.2 Case Handling

This diagram shows when the user does not have the privilege for disabling an account causing his request for disabling an account failing the validation in the controller.

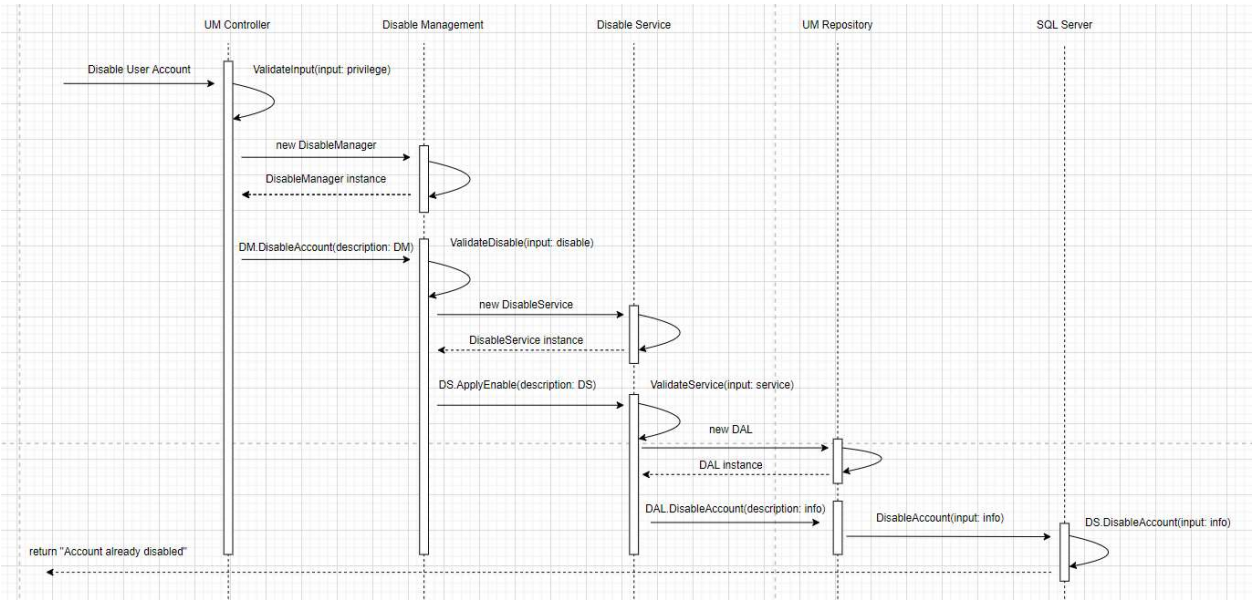
This diagram shows when the instance of DisableManager is not called due to it failing the DisableManagement validation.



This diagram shows when the instance from DisableManagement is not validated when requesting service from DisableService.



This diagram shows the error case when a user tries to disable an account and passes through the Controller Validation, Management Validation and the Service Validation but the account that the user is trying to disable is already disabled in the Data Store.

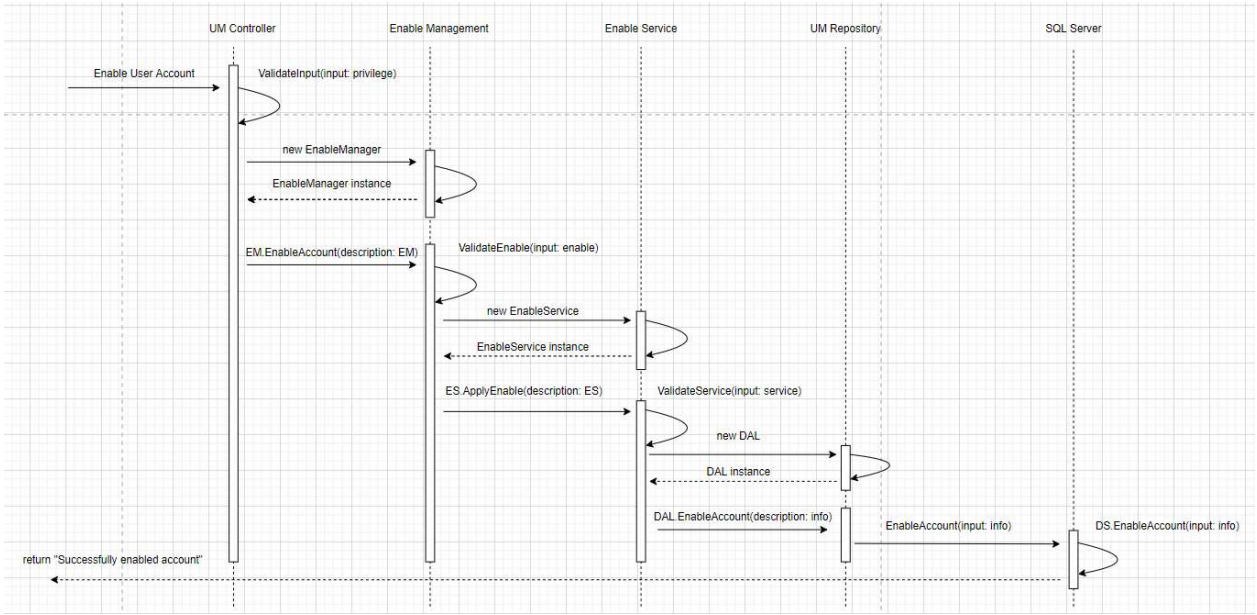


2.5 Enabling User Account

This section outlines how a user account is enabled or given more privileges in a database. This can be carried out by an administrator or someone with privileges that allows for account status to be altered.

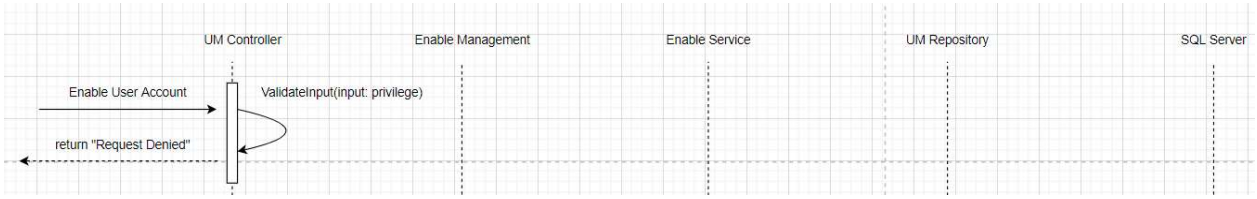
2.5.1 Low Level Diagram

This diagram shows when a user enables a user account with the program running through a series of validations in order to complete the process of enabling the user account. The user's request for enabling an account has to go through the input validation in the Controller which checks the user's privileges to determine if they are allowed to enable an account. An instance of EnableManager is then created and has to go through validation in Management. The Manager then creates a Service which is validated in the service layer and the Service applies the changes made to the account, in this case creating one, and then creates an instance of the Data Access Layer which is validated and then applies the changes to the Data Store. If all validations go through and no exceptions are raised then the user will have successfully enabled the account.

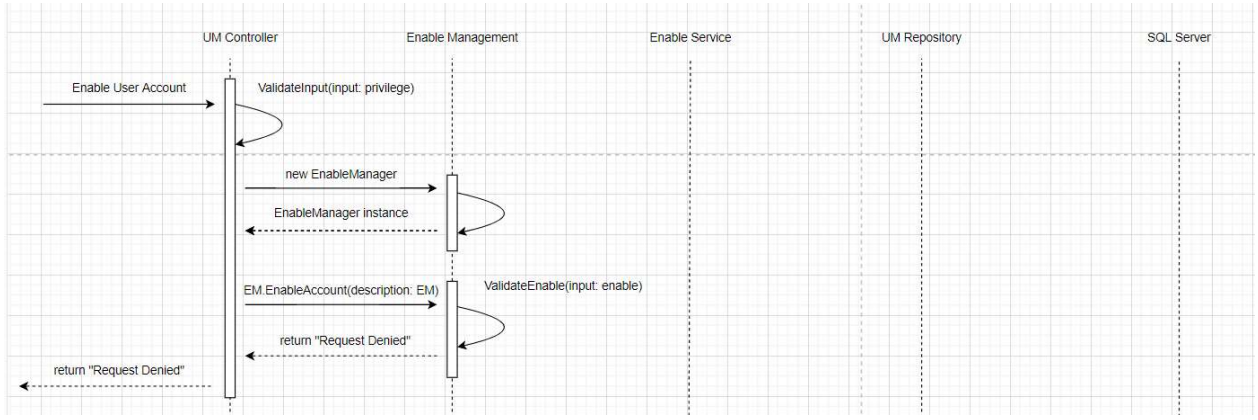


2.5.2 Case Handling

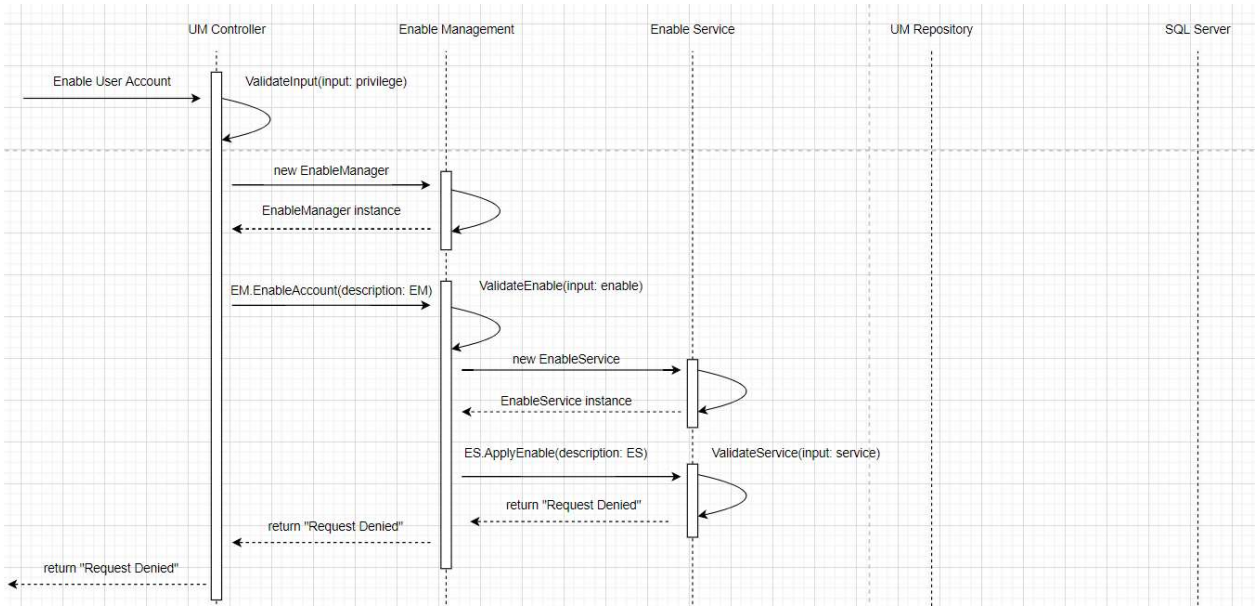
This diagram shows when the user does not have the privilege for enabling an account causing his request for enabling an account failing the validation in the controller.



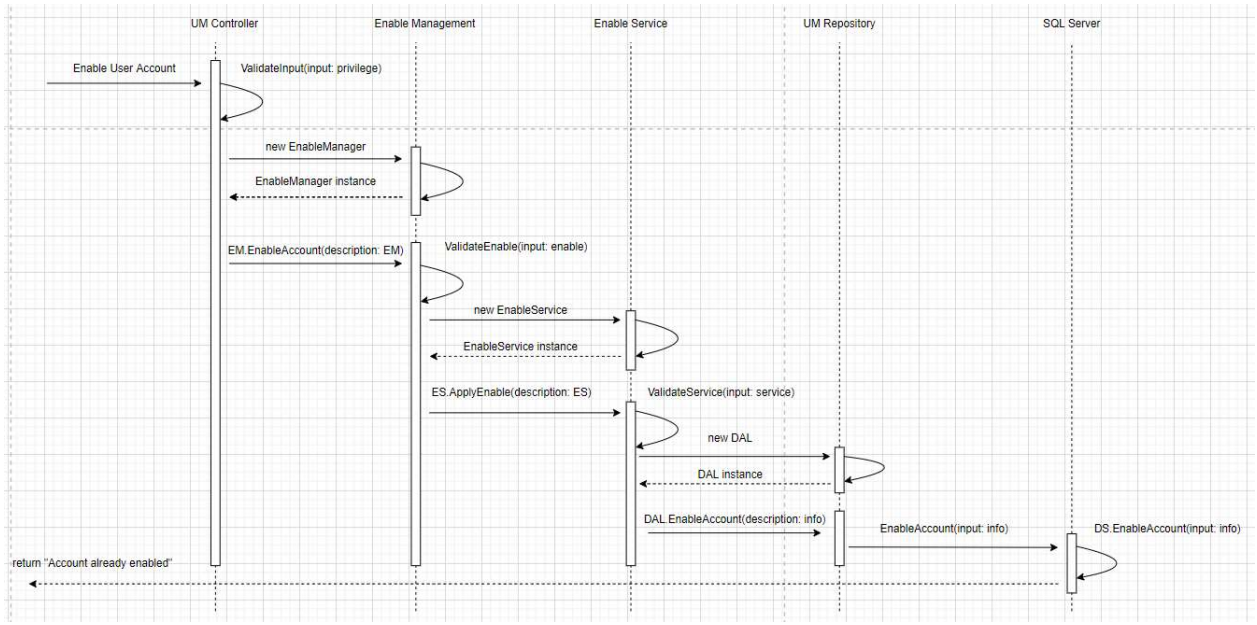
This diagram shows when the instance of EnableManager is not called due to it failing the EnableManagement validation.



This diagram shows when the instance from EnableManagement is not validated when requesting service from EnableService.



This diagram shows the error case when a user tries to enable an account and passes through the Controller Validation, Management Validation and the Service Validation but the account that the user is trying to enable is already enabled in the Data Store.

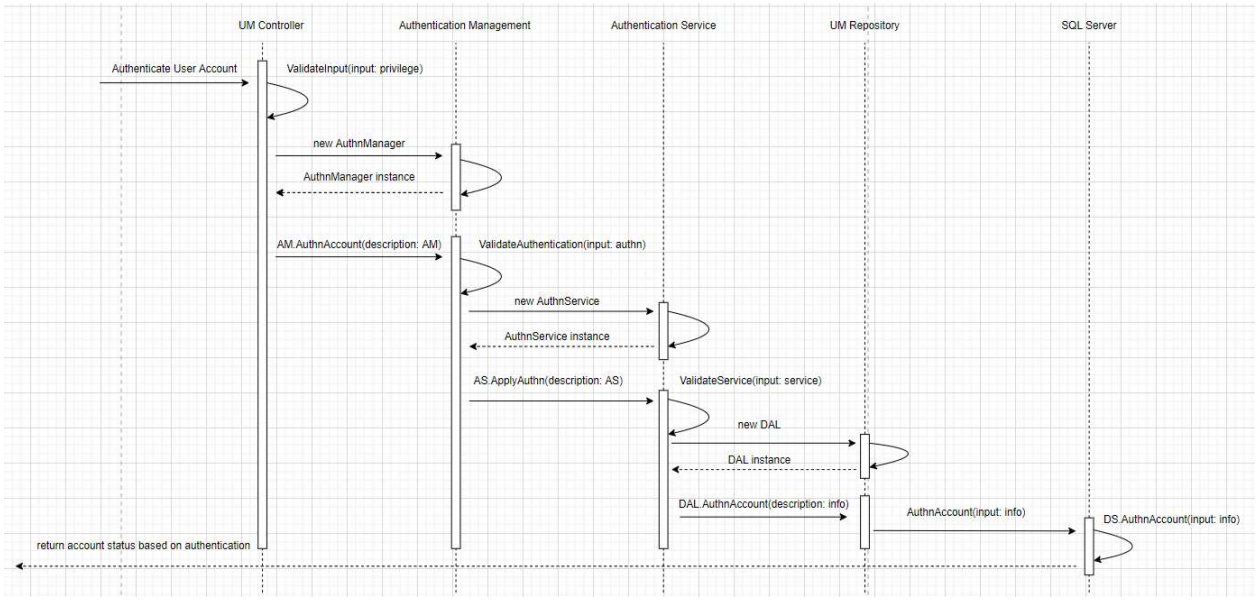


3. Account Authentication

When using the application, the user is required to input their credentials in order to access their account. This process includes authenticating the provided information by the user to see if it matches with the information associated to their account.

3.1 Low Level Diagram

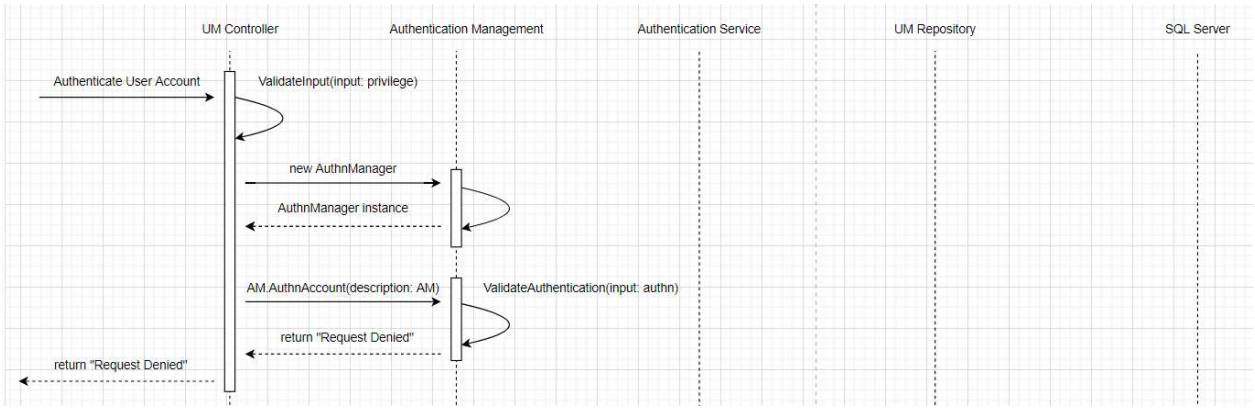
This diagram shows when a user logs in to an account with the program running authentication using a series of validations in order to complete the process of logging into the account. The user's request for logging into the account has to go through the input validation in the Controller which checks the user's privileges to determine if they are allowed to log into an account. An instance of the Manager is then created and has to go through validation in Management. The Manager then creates an instance of the Service which is validated in the service layer and the Service then creates an instance of the Data Access Layer which is validated and then checks the credentials of the user with the Data Store. If all validations go through and no exceptions are raised then the user will have logged in to their account.



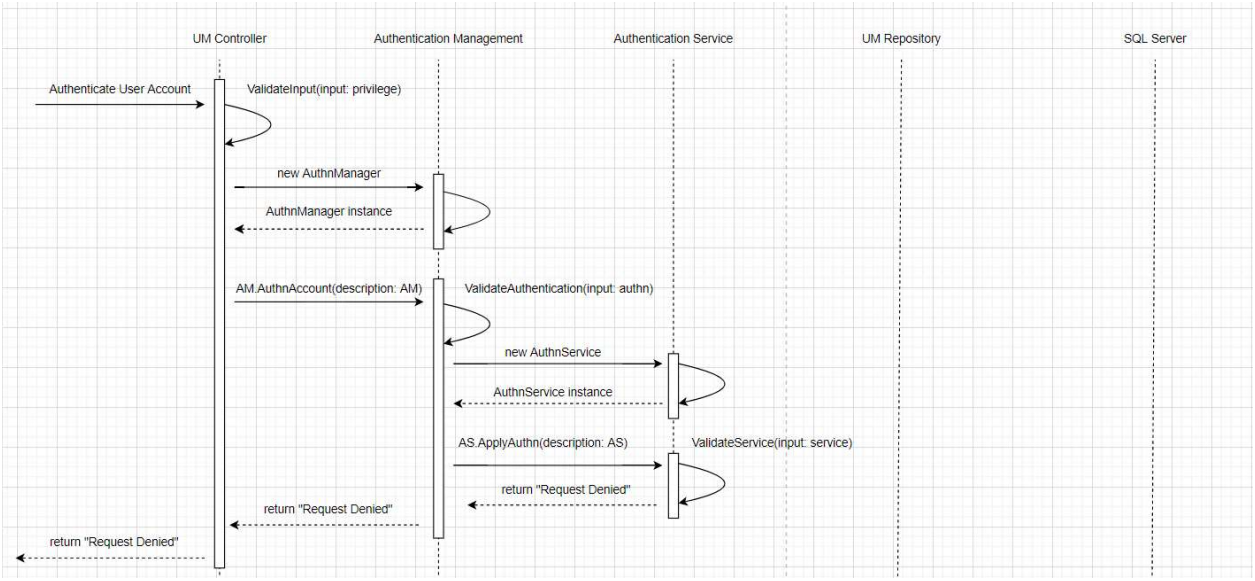
3.2 Case Handling

This diagram shows when the user does not have the privilege for authenticating an account causing the request for authenticating an account failing the validation in the controller.

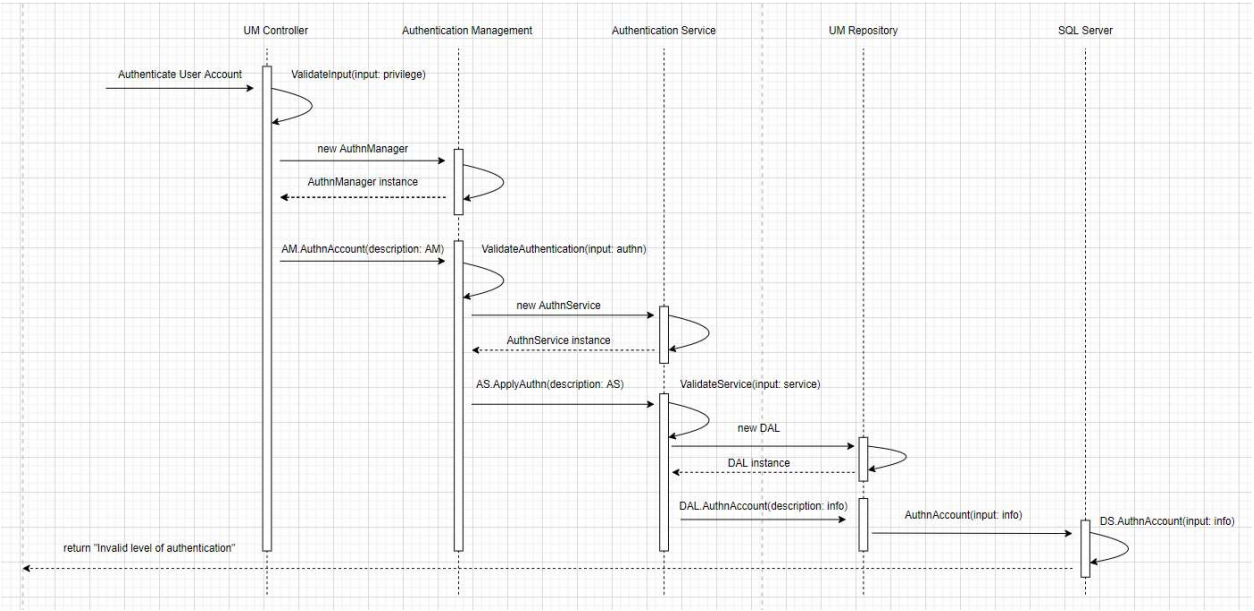
This diagram shows when the instance of AuthnManager is not called due to it failing the AuthenticationManagement validation.



This diagram shows when the instance from AuthnManagement is not validated when requesting service from AuthnService.



This diagram shows the error case when a user tries to enable an account and passes through the Controller Validation, Management Validation and the Service Validation but the account that the user is trying to authenticate does not have the necessary qualifications that are established in the Data Store.

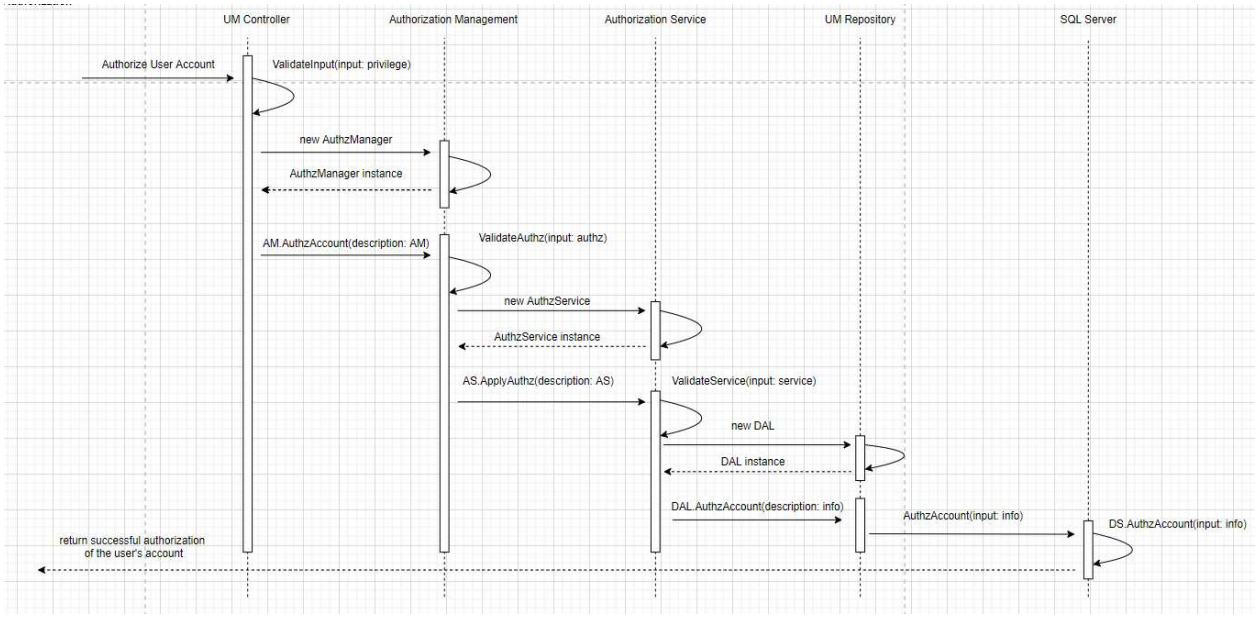


4. Account Authorization

When using the application, the user is required to input their credentials in order to access their account. This process includes authorizing the provided information by the user to see if it matches with the information associated with their account.

4.1 Low Level Diagram

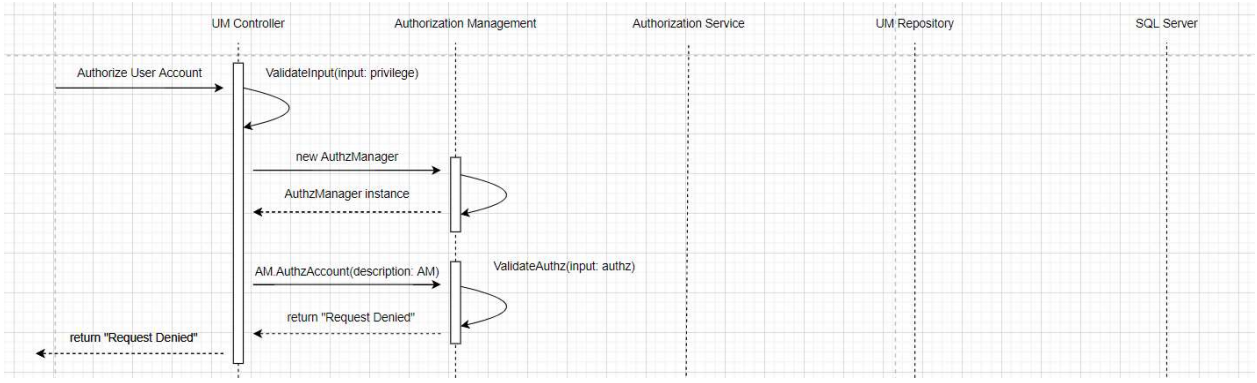
This diagram shows when a user's account goes through authorization to get permissions within the application with the program running the authorization using a series of validations. The user's request to authorize the account has to go through the input validation in the Controller which checks the user's privileges to determine if they are allowed to authorize the account. An instance of the Manager is then created and has to go through validation in Management. The Manager then creates an instance of the Service which is validated in the service layer and the Service then creates an instance of the Data Access Layer which is validated and then checks the credentials of the user with the Data Store. If all validations go through and no exceptions are raised then the user will have successfully authorized their account to gain access.



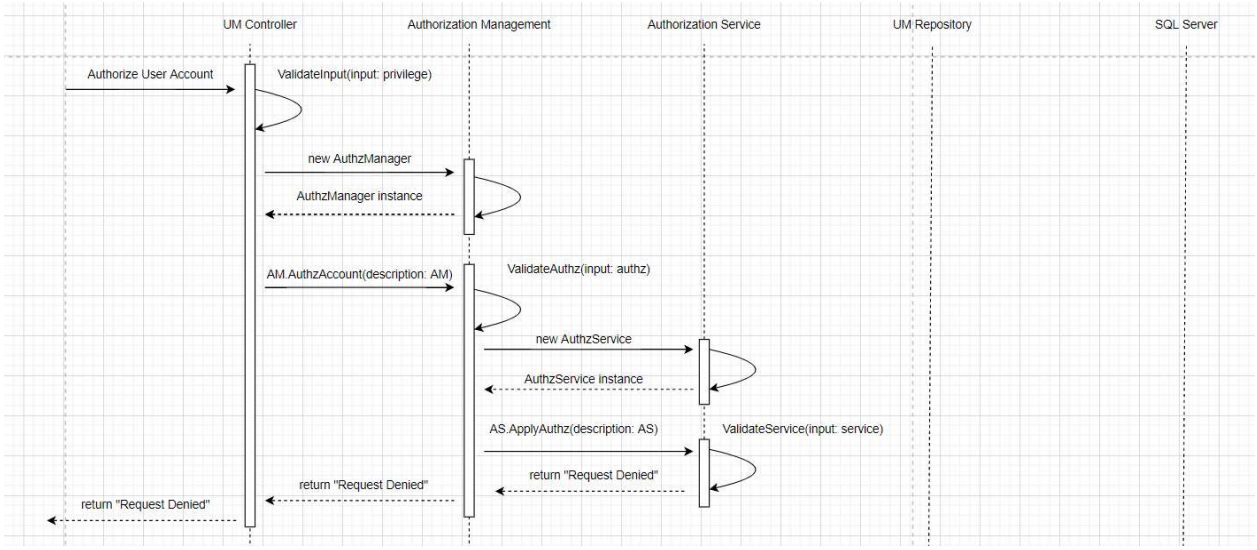
4.2 Case Handling

This diagram shows when the user does not have the privilege for authorizing an account causing his request for authorizing an account to fail the validation in the controller.

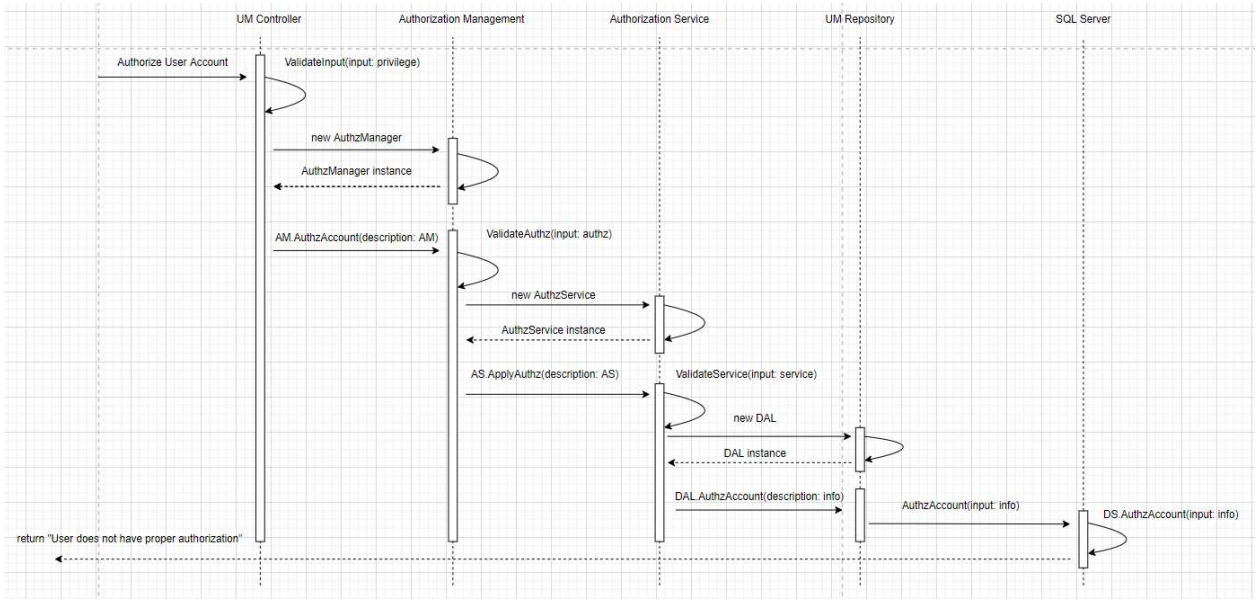
This diagram shows when the instance of AuthzManager is not called due to it failing the AuthorizationManagement validation.



This diagram shows when the instance from AuthzManagement is not validated when requesting service from AuthzService.



This diagram shows the error case when a user tries to enable an account and passes through the Controller Validation, Management Validation and the Service Validation but the account that the user is trying to authorize does not have the necessary qualifications that are established in the Data Store.

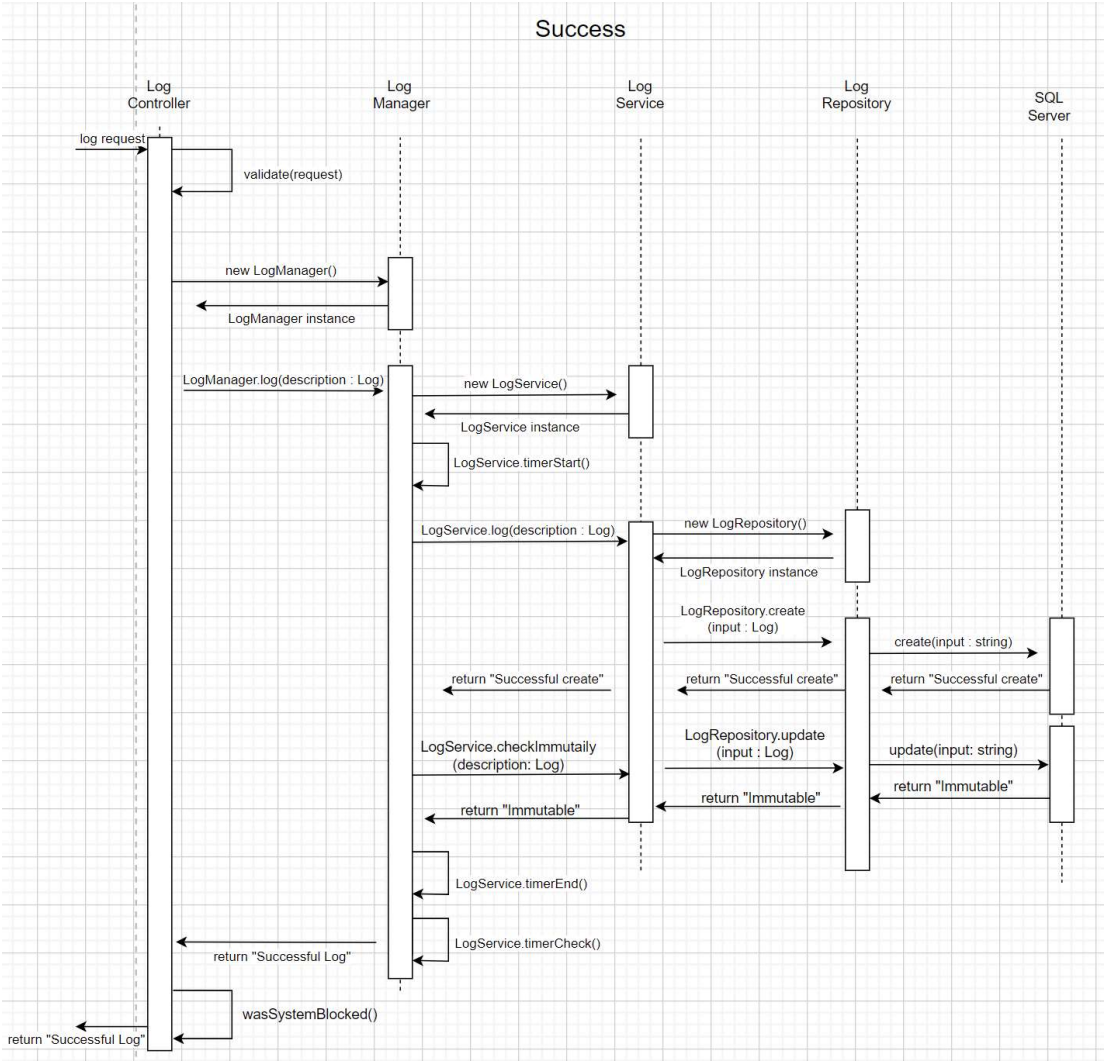


5. Logging

When using the application, the development team is required to have the ability to log for event tracking within the system. This process includes taking some information and writing to a data store.

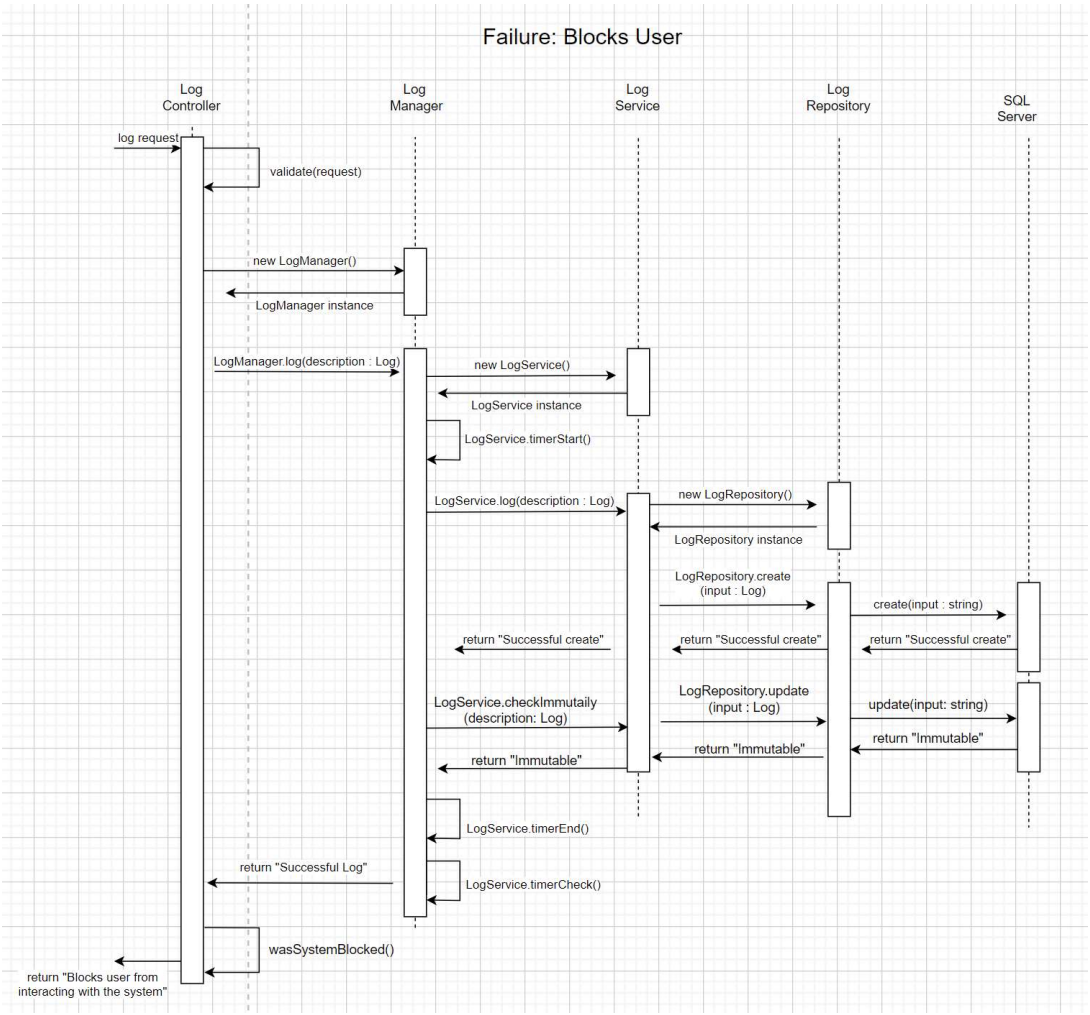
5.1 Low Level Diagram

This diagram shows the flow of control for a request for logging. The request is validated to ensure the request to log is a valid one. A new manager is instantiated to call the log method which in turn instantiates a log service where the information to log is handled. This log service includes methods to check the time it takes to log, check for immutability, and log. It will access an extension of a repository interface, log repository, as part of the data access layer. This repository class will handle the writing of logs to the sql server data store and also the update attempt of logs to check for immutability. A last check is made by the controller to ensure that the system stayed online during the logging process to ensure a successful log scenario.

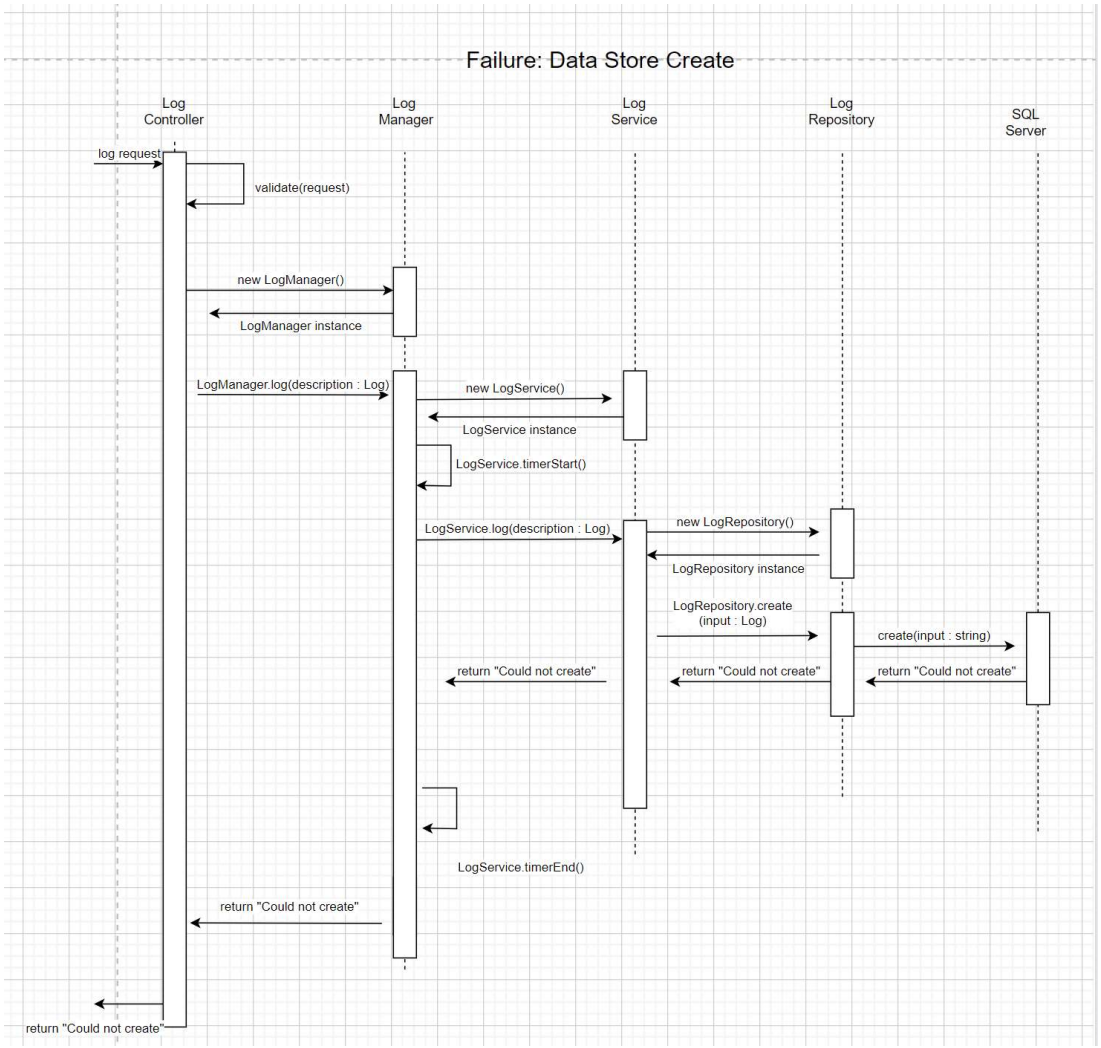


5.2 Case Handling

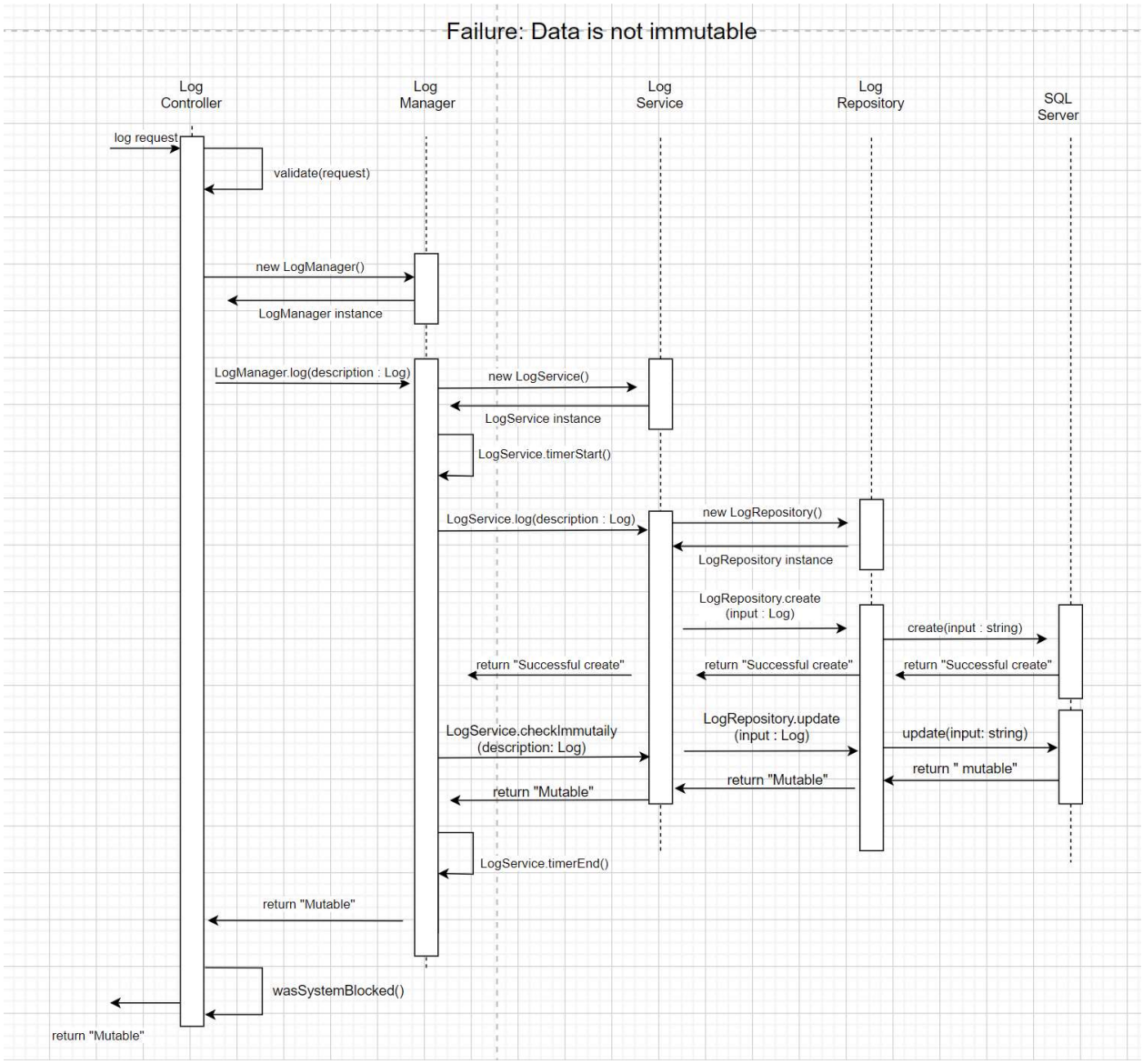
This diagram shows the scenario when the system was found to have not blocked when the logging process occurred.



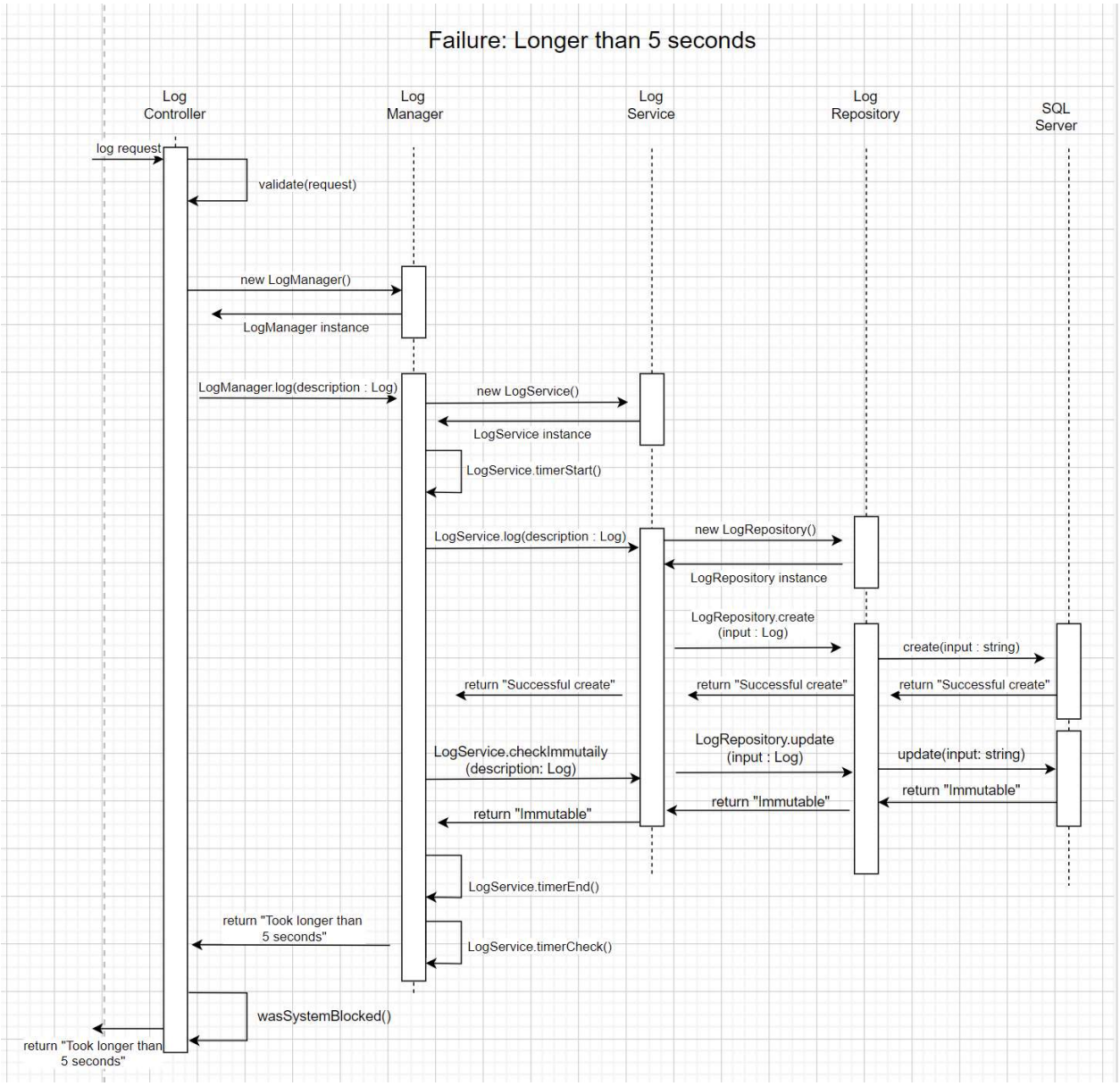
This diagram shows the scenario when the write operation for logging fails. A string is returned in order for developers to tell where in the log method the failure occurred.



This diagram shows the scenario when the logs are found to be mutable instead of immutable. This is considered a failure for logging and will be reported as a string to tell the development team where the failure occurred in the log process. This diagram shows the scenario when the request to log is invalid.



This diagram shows the scenario where the entire log operation takes longer than 5 seconds. This will return a string informing the development team where the failure occurred in the logging process.

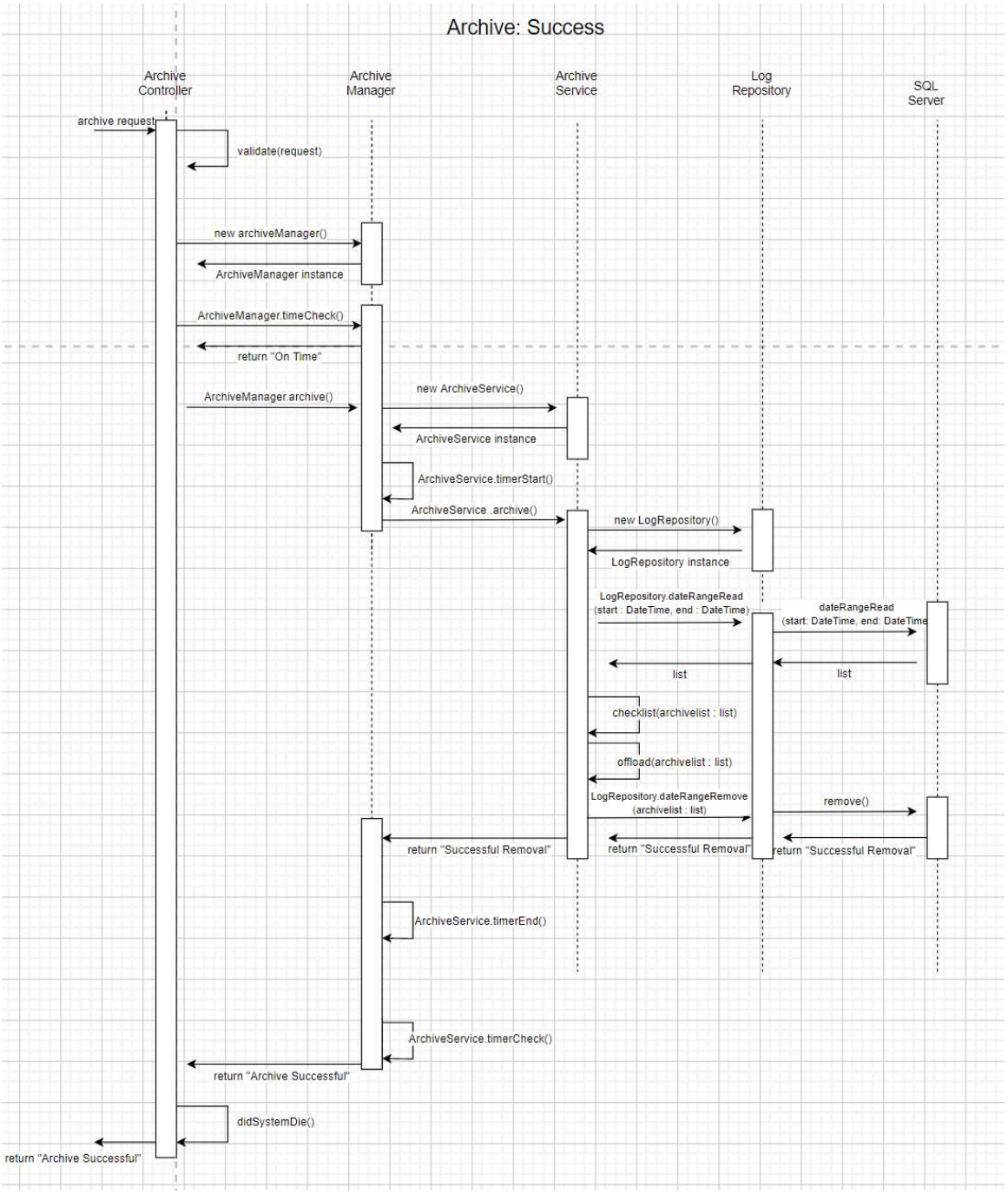


6. Archive

6.1 Low Level Diagram

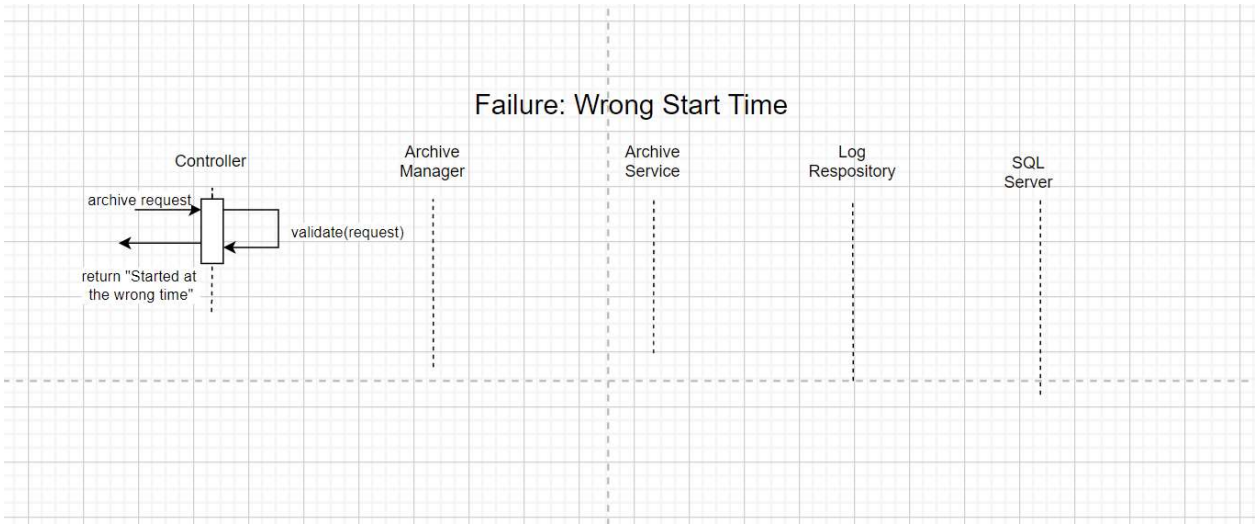
This diagram shows the flow of control for a request for archiving. The request is validated for this reason to check that it is the correct date and time for archiving to occur. A new manager is instantiated to call the archive method which in turn instantiates an archive service where the information to archive is handled. This archive service includes methods to check the time it takes to archive, check for existence, and offload. It will access an extension of a repository interface, log repository, as part of the data access layer. This repository class will handle the reading of logs from the sql server data store and also the removal of logs read. A last

check is made by the controller to ensure that the system stayed online during the archiving process to ensure a successful archive scenario.

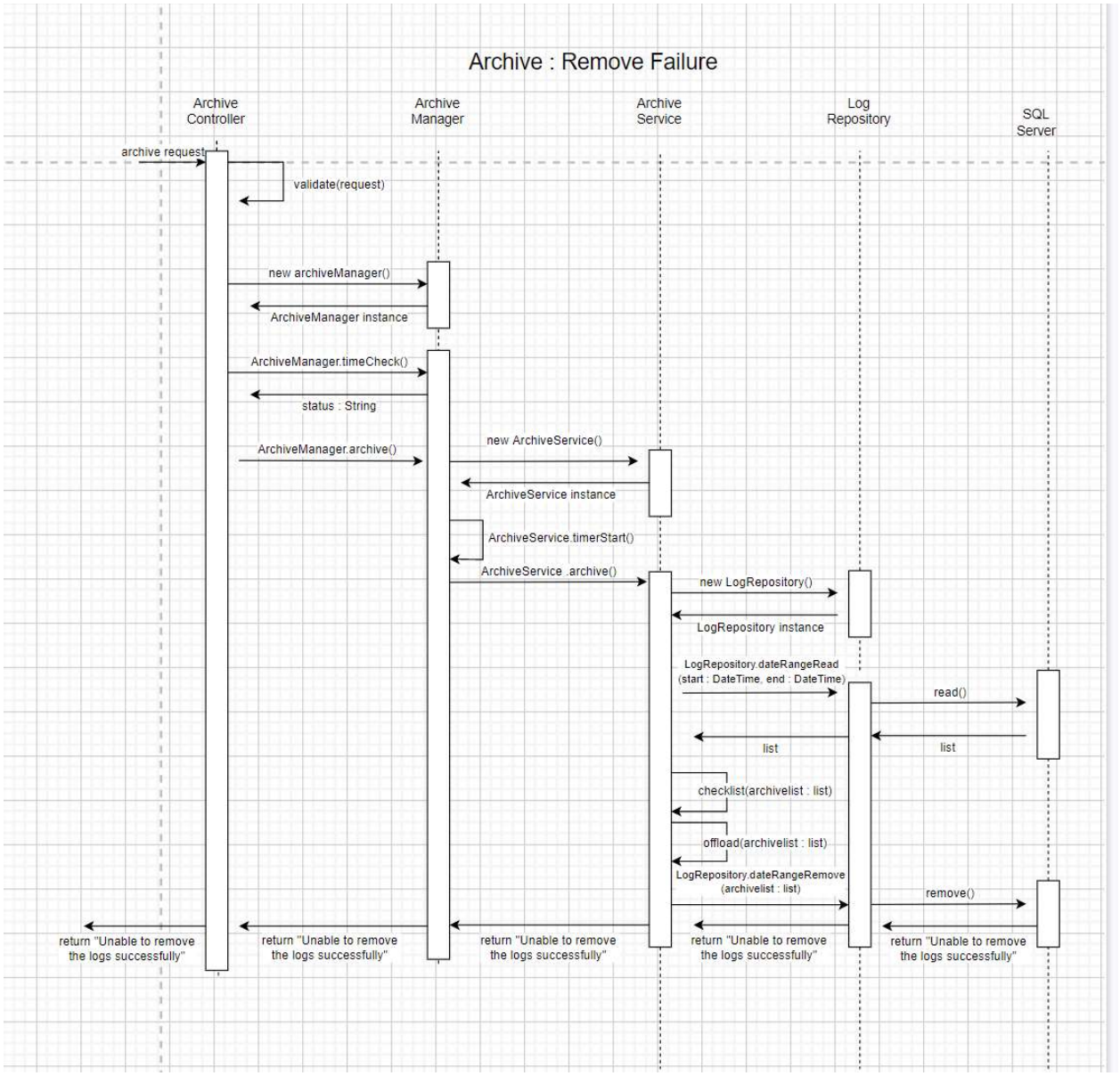


5.2 Case Handling

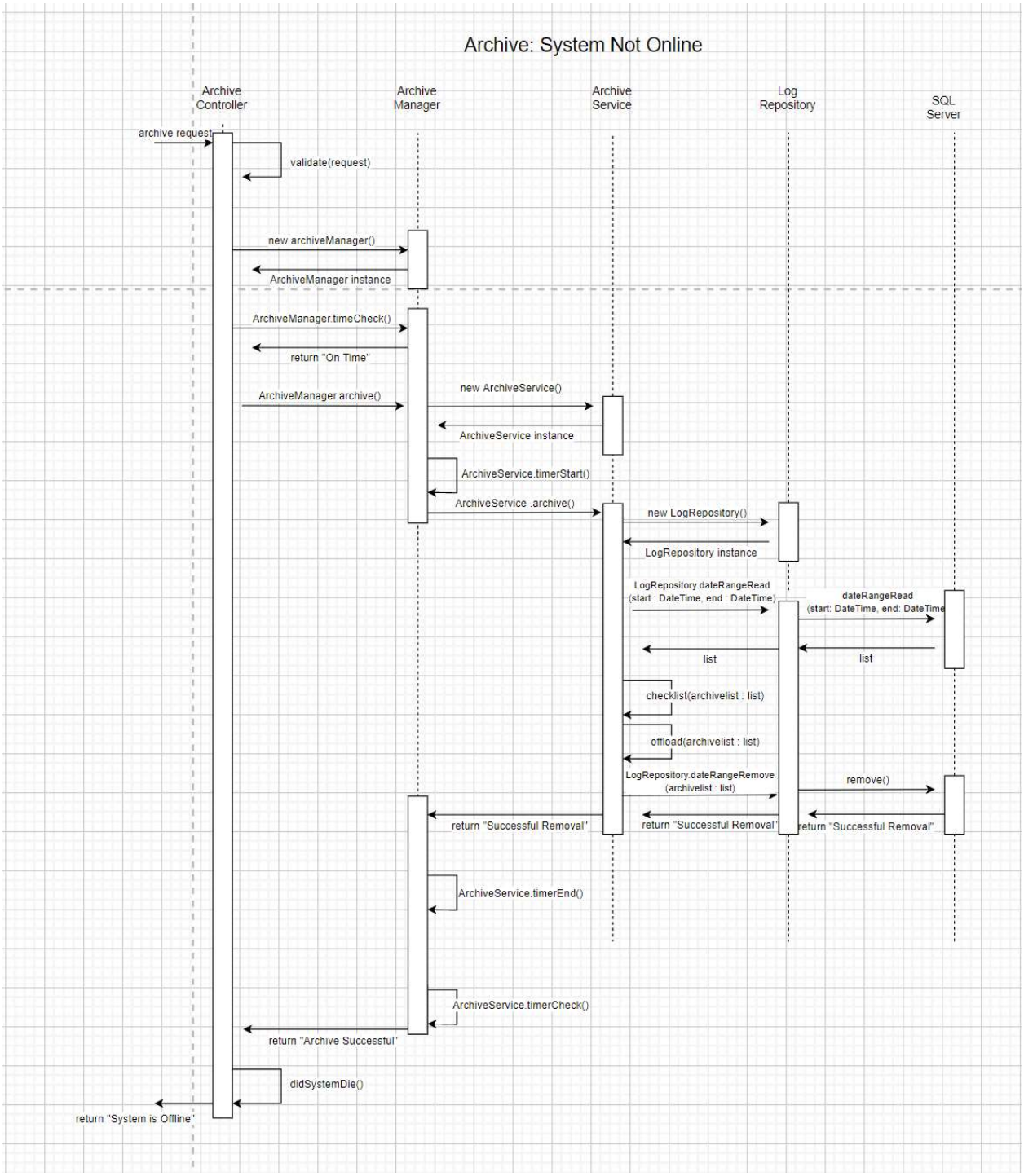
This diagram shows the scenario when the request to archive was invalid. This means it was not the right date time for an archive to occur.



This diagram shows the scenario when the remove operation for archiving fails. A string is returned in order for developers to tell where in the log method the failure occurred.



This diagram shows the scenario when the system was found to have gone offline when the logging process occurred. A string will be returned to inform the development team where the archiving process went wrong.



This diagram shows the scenario where the entire archive operation takes longer than 60 seconds. This will return a string informing the development team where the failure occurred in

the archiving.