Version 2.0.0

Low Level Diagram

November 9, 2022

Issued by:

Algorithmic Alchemist

Team Lead

Sierra Harris

Team Members

Bryant Lam

Abhay Solanki

Faisal Al Muharrami

David Chan

Github: https://github.com/abhay772/AA_Senior_Project/

Version History

Version #	Date	Reason for Change
Version 1.0.0	11/9/2022	Original Document
Version 2.0.0	12/14/2022	Added Registration, Authentication, and Authorization

Table of Contents

Overview

The Low Level Design document will describe the layer interaction of functionalities visually through UML sequence diagrams. This document is designed to help developers understand the flow and interactions of functionality through abstract layers. Layers will mainly cover the system as a whole, from frontend to backend. There will be multiple sequence diagrams for a feature to cover the success case and all failure cases derived from business rules.

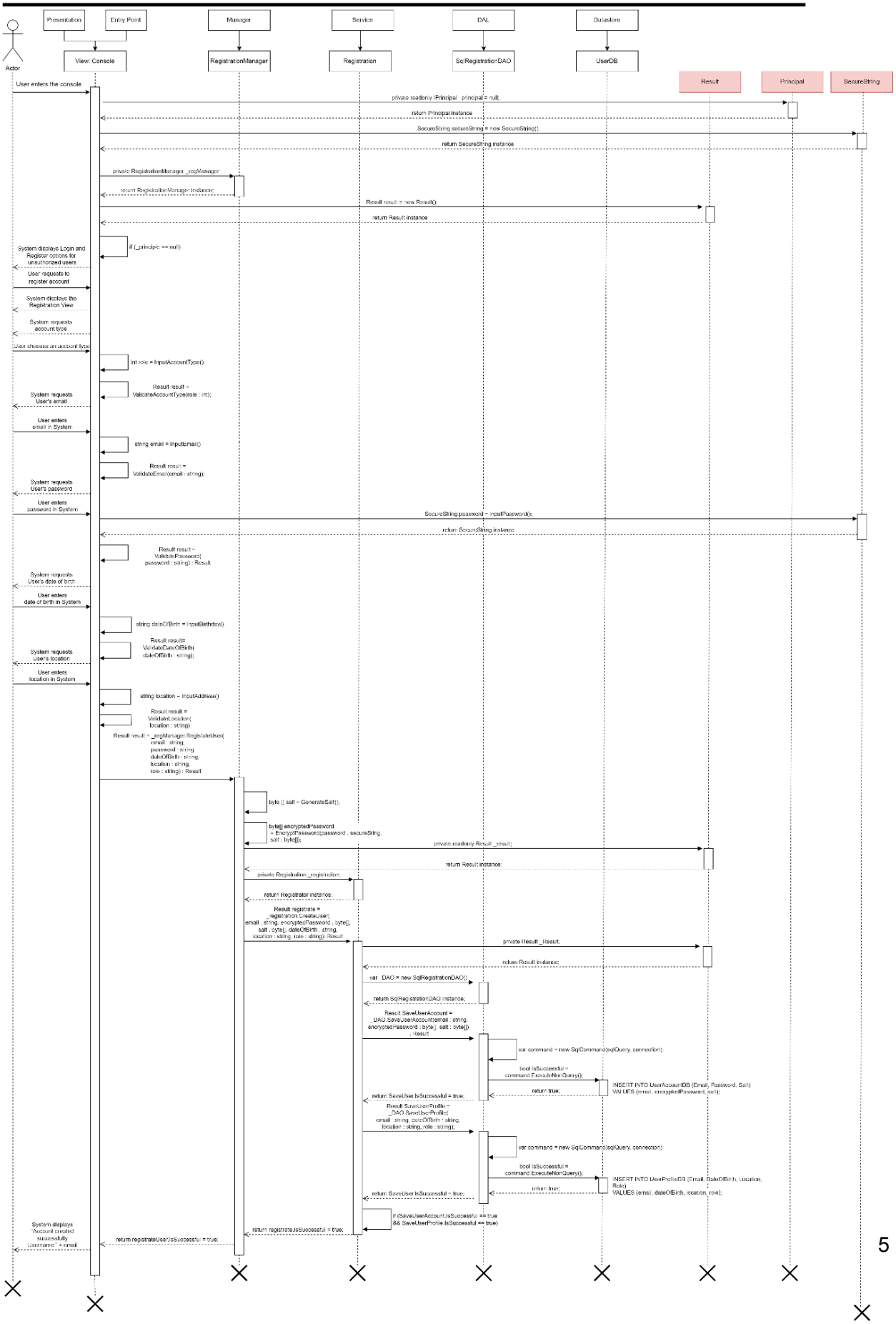
Registration

From this point on, refer to the link below for naming conventions.
Ex. Class member field called "_noun" will be private.
https://github.com/v-vong3/csulb/blob/master/cecs_491/docs/cecs491-coding-standards.pdf

Registration Success Case 1: User registers with a valid email and valid passphrase.

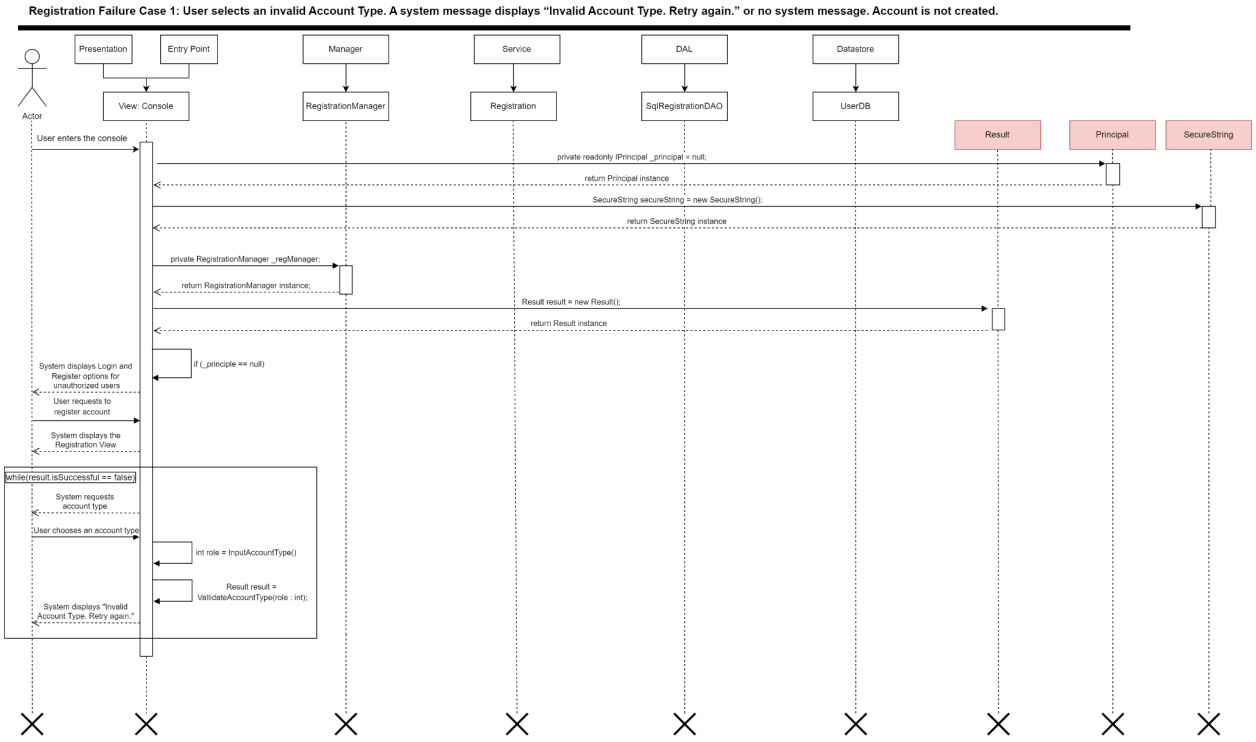
The system is able to assign a system-wide unique username. A system message displays "Account created successfully" within 5 seconds of invoking the registration process. The system provides the username to the user.

Registration Success Case 1: User registers with a valid email and valid passphrase. The system is able to assign a system-wide unique username.
A system message displays "Account created successfully" within 5 seconds of invoking the registration process. The system provides the username to the user.



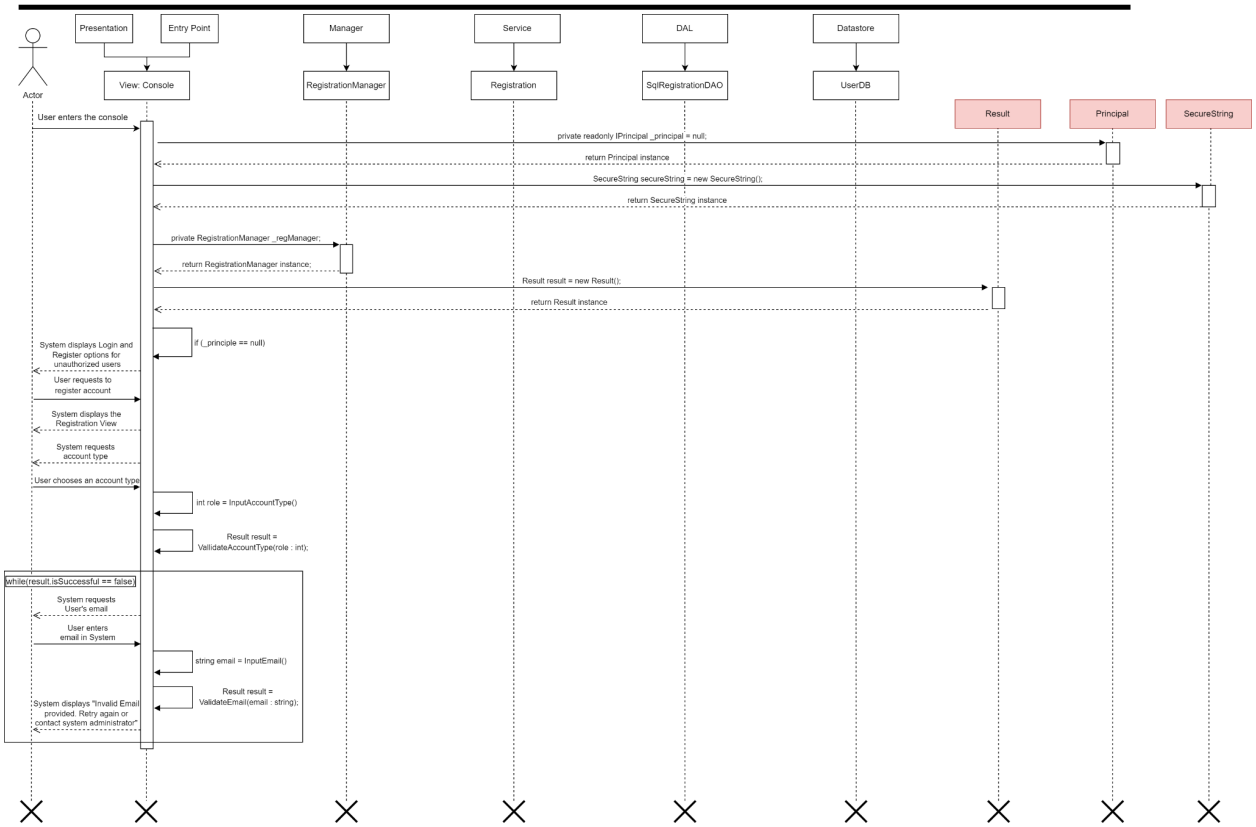
- “RegistrationDB” is the table name within a database relating to user microservices.
- Please refer to Logging Success Case for LogAsync().
- ExecuteSql(string stringConnection) will connect to the database relating to user microservices and only perform insert statements to RegistrationDB for the case of Registration.

Registration Failure Case 1 from Invalid Account Type



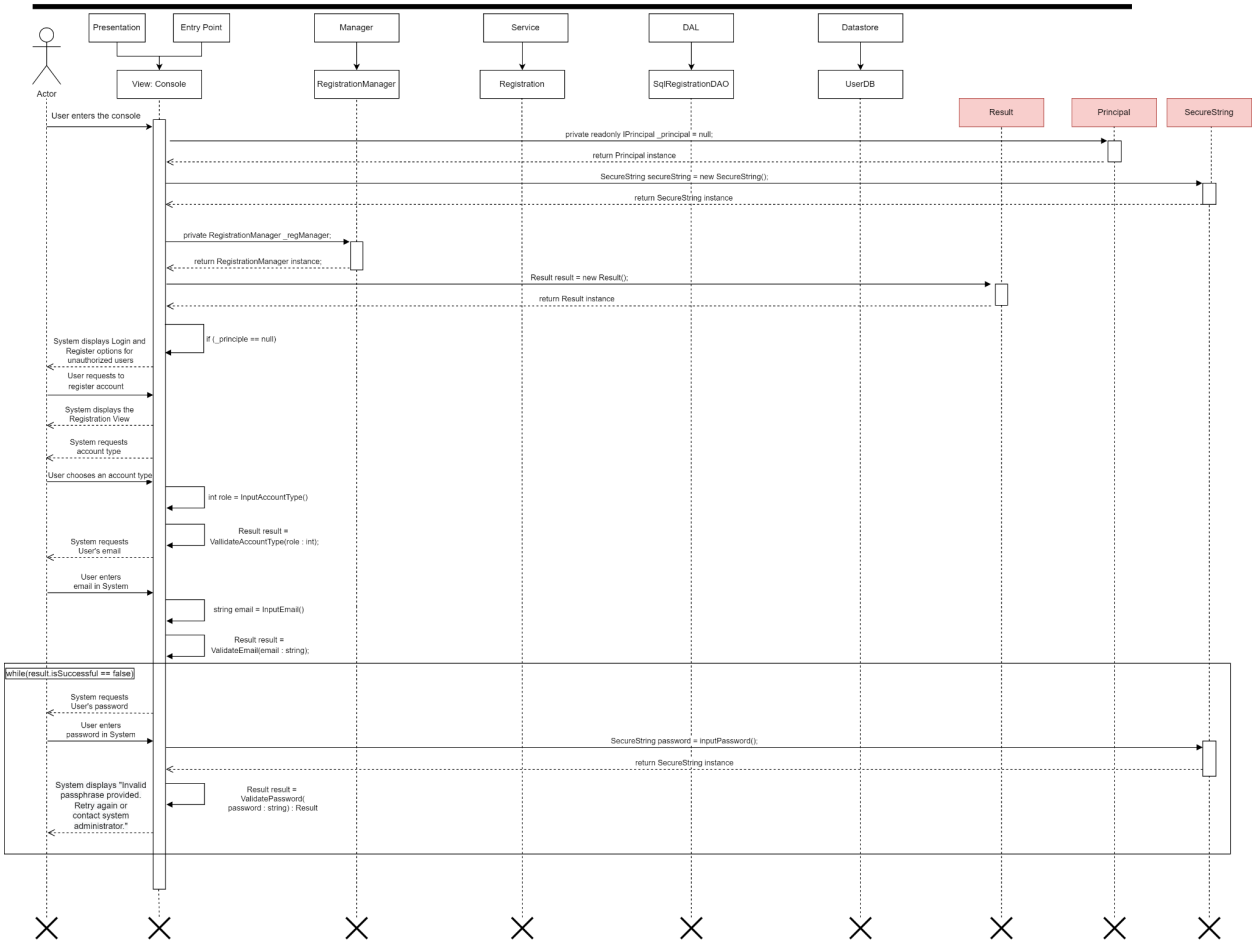
Registration Failure Case 2 from Invalid Email

Registration Failure Case 2: User registers with an invalid email. A system message displays "Invalid email provided. Retry again or contact system administrator" or no system message. Account is not created.

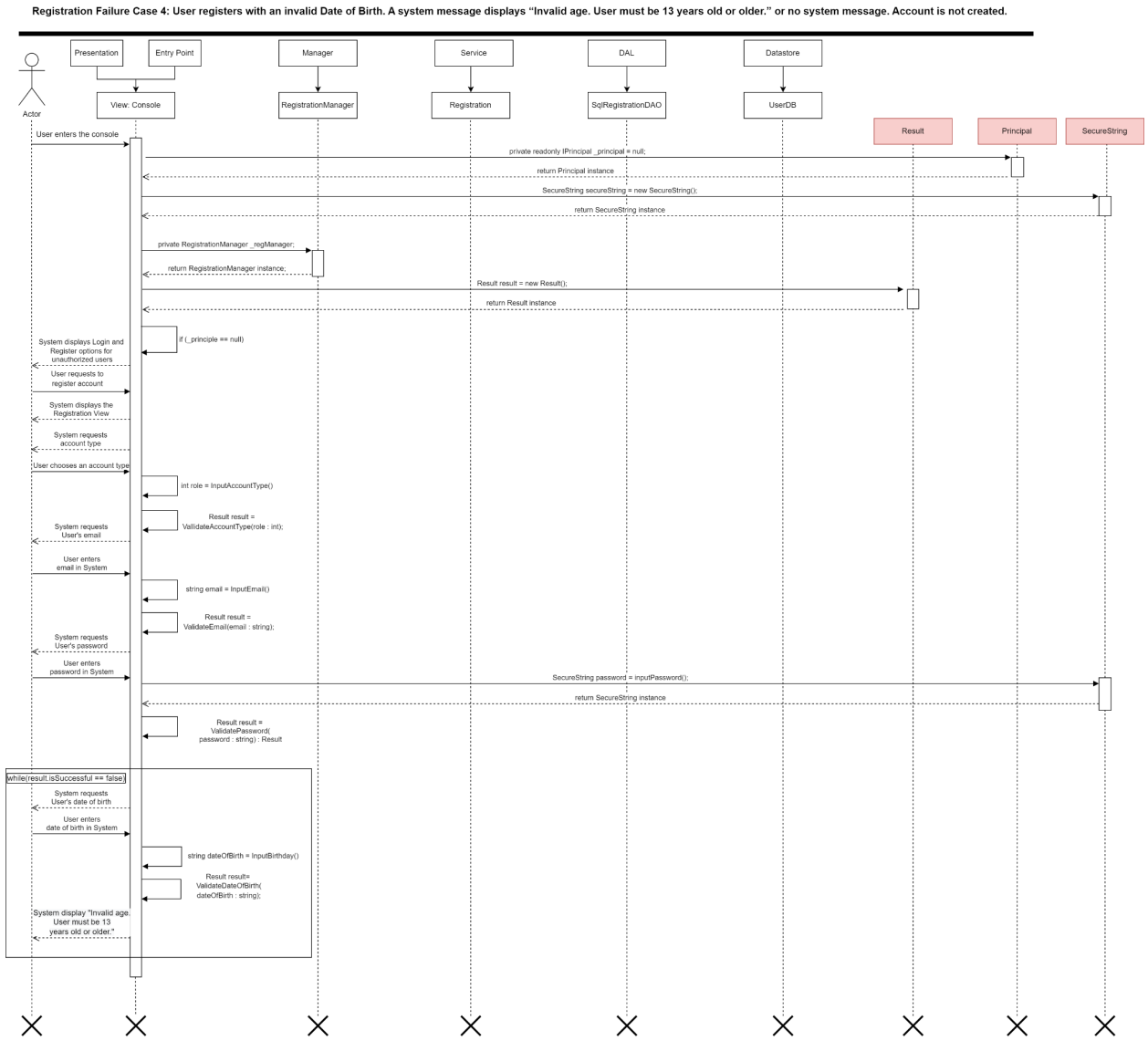


Registration Failure Case 3 from Invalid Passphrase

Registration Failure Case 3: User registers with an invalid passphrase. A system message displays "Invalid passphrase provided. Retry again or contact system administrator" or no system message, Account is not created.

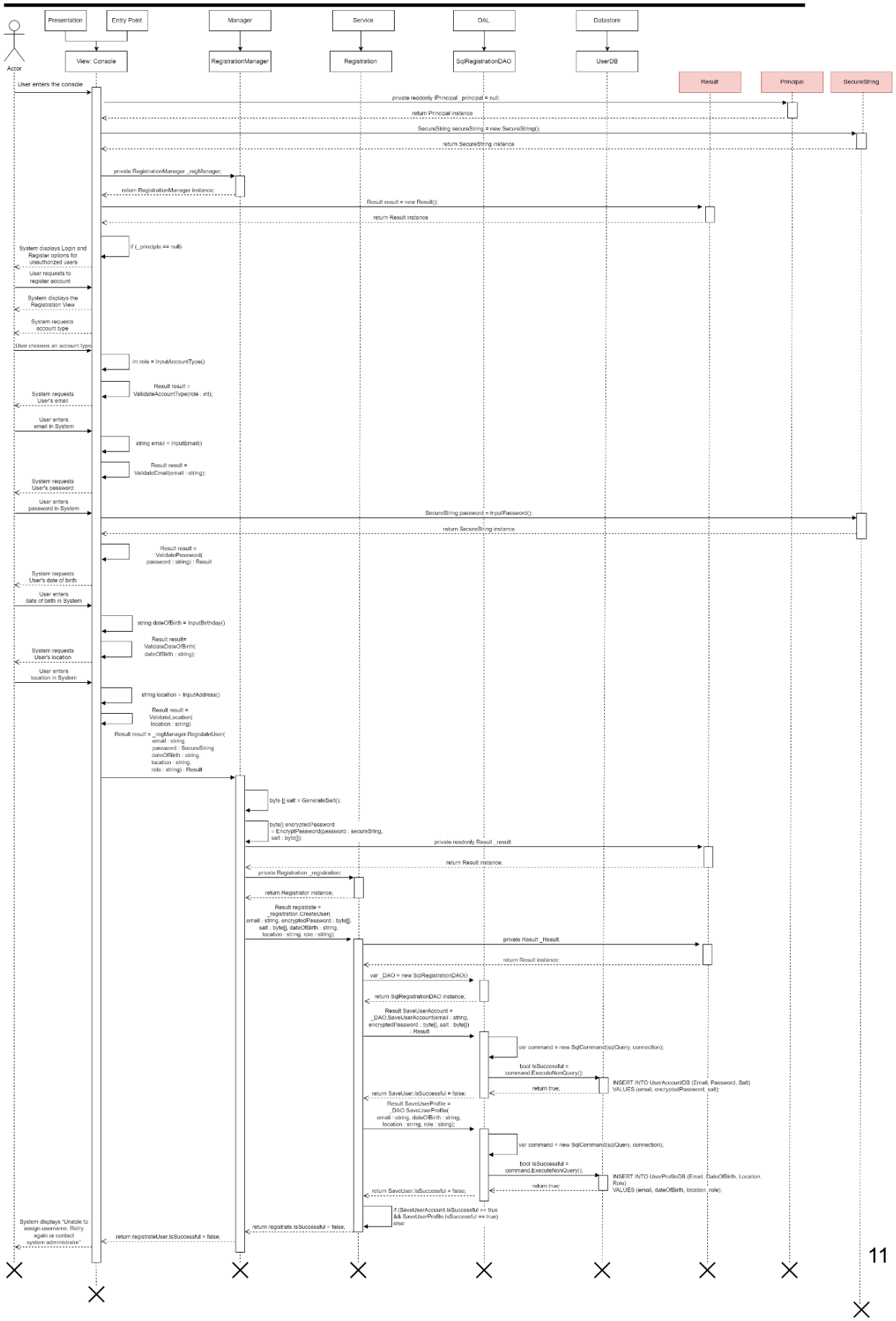


Registration Failure Case 4 from Invalid Date of Birth



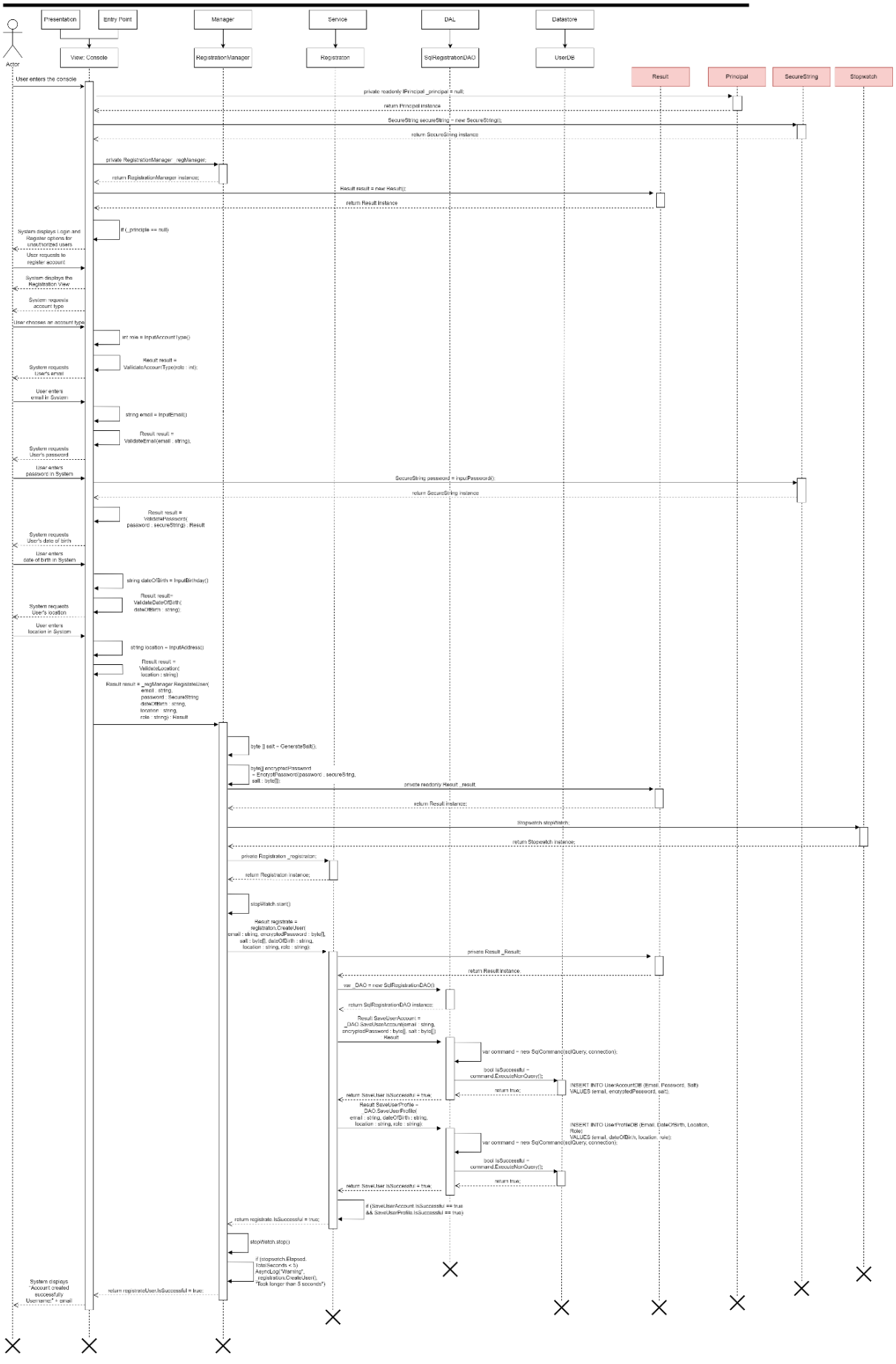
Registration Failure Case 5 from Unable to assign username with valid email and passphrase

Registration Failure Case 5: User registers with a valid email and valid passphrase. The system was unable to assign a system-wide username. A system message displays "Unable to assign username. Retry again or contact system administrator". Account is not created.



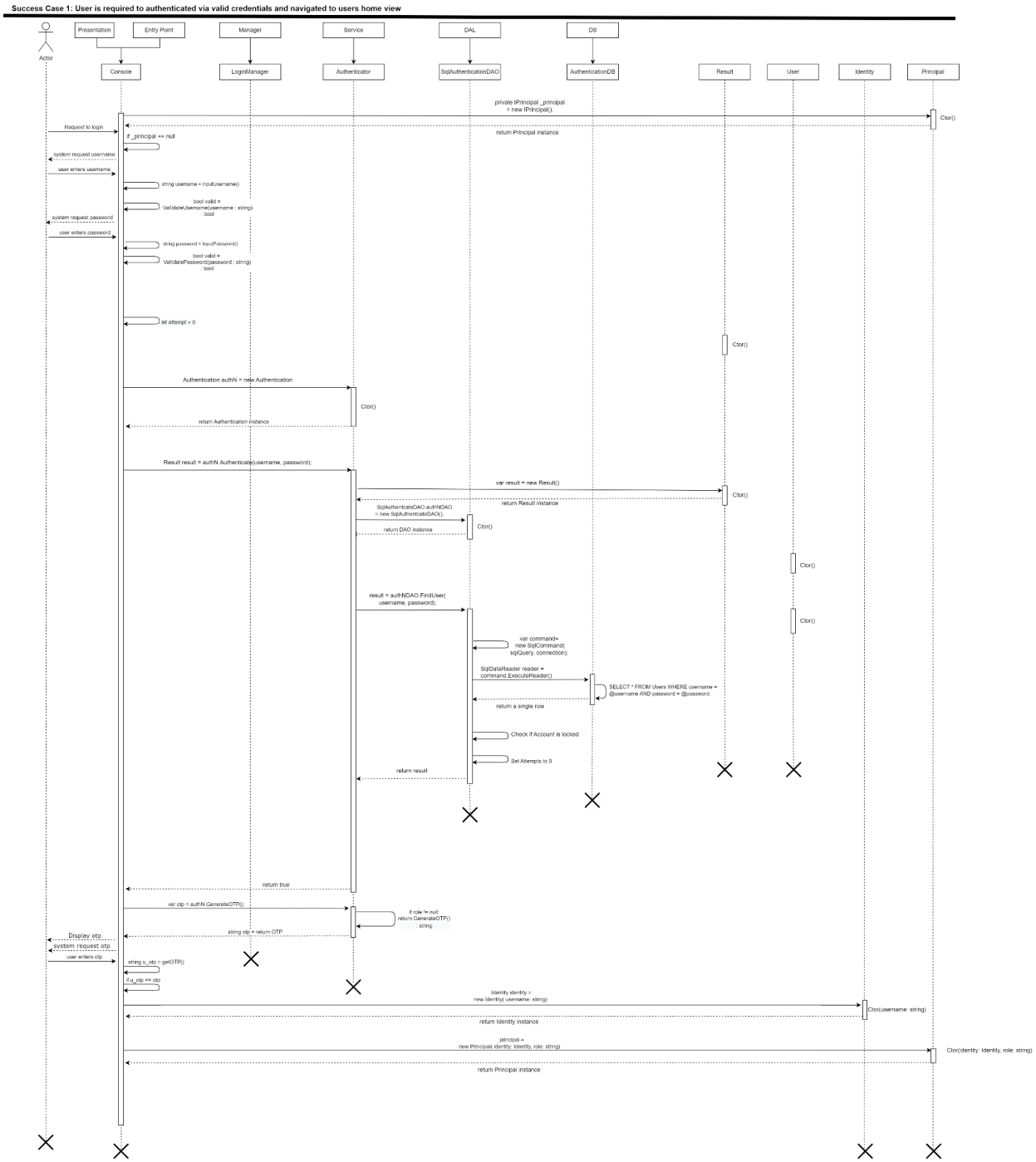
Registration Failure Case 6 from _registrate.CreateUser() took longer than 5 seconds before logging

Registration Failure Case 6: User registers with a valid email and valid passphrase. The system was able to assign a system-wide username. Entire process took longer than 5 seconds. A system log entry is recorded. Account is created.

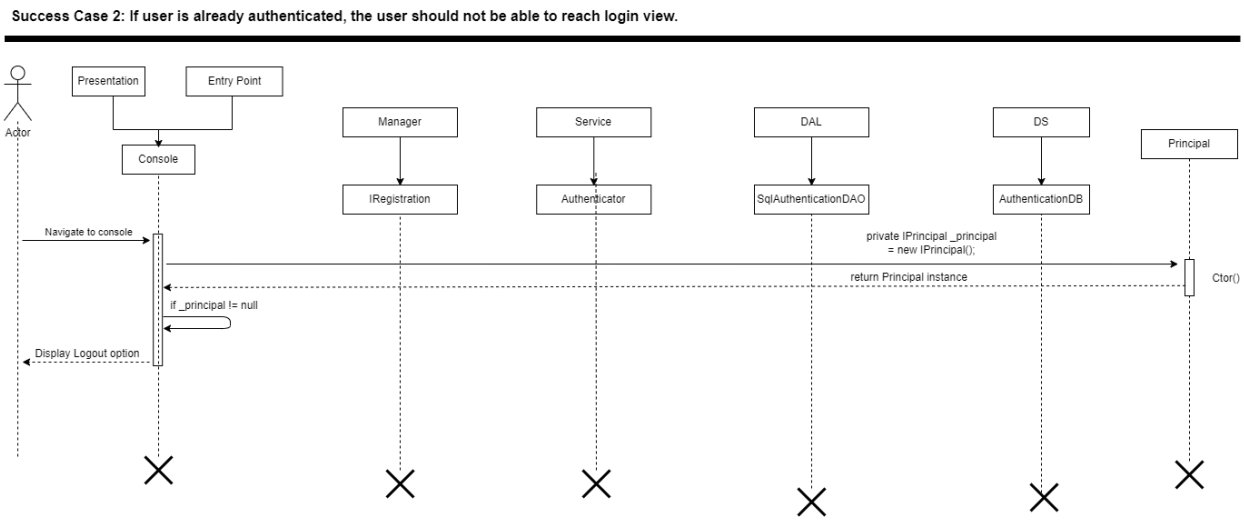


Authentication

Authentication Success Case 1 from user authenticated through valid credentials

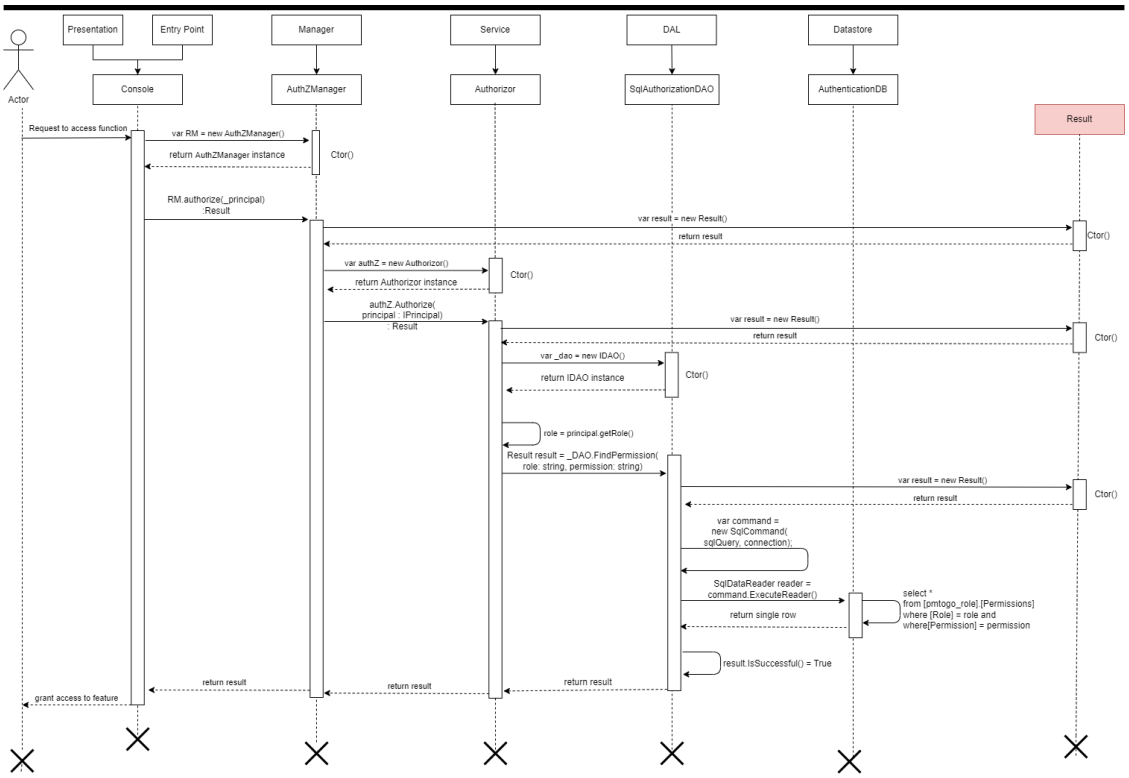


Authentication Success Case 2 from user is already authenticated



Authorization

Authorization Success Case 1 - 4



The Authorization Success Case encompasses all 4 cases:

Success Cases 1: User attempts to access a protected functionality within authorization scope. Access is granted to perform functionality.

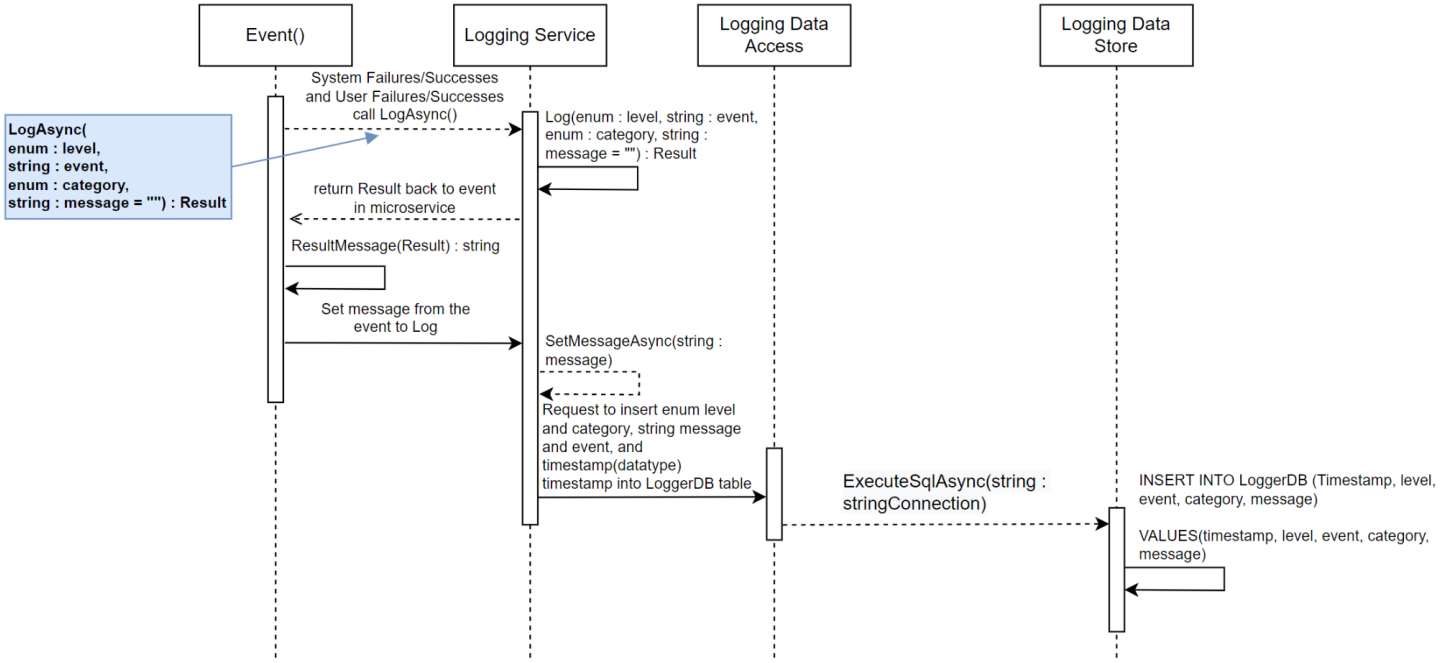
Success Case 2: User attempts to access protected data within authorization scope. Access is granted to perform read operations.

Success Case 3: User attempts to modify protected data within authorization scope. Access is granted to perform write operations.

Success Case 4: User attempts to access protected views within authorization scope. Access is granted to the view. User is automatically navigated to view.

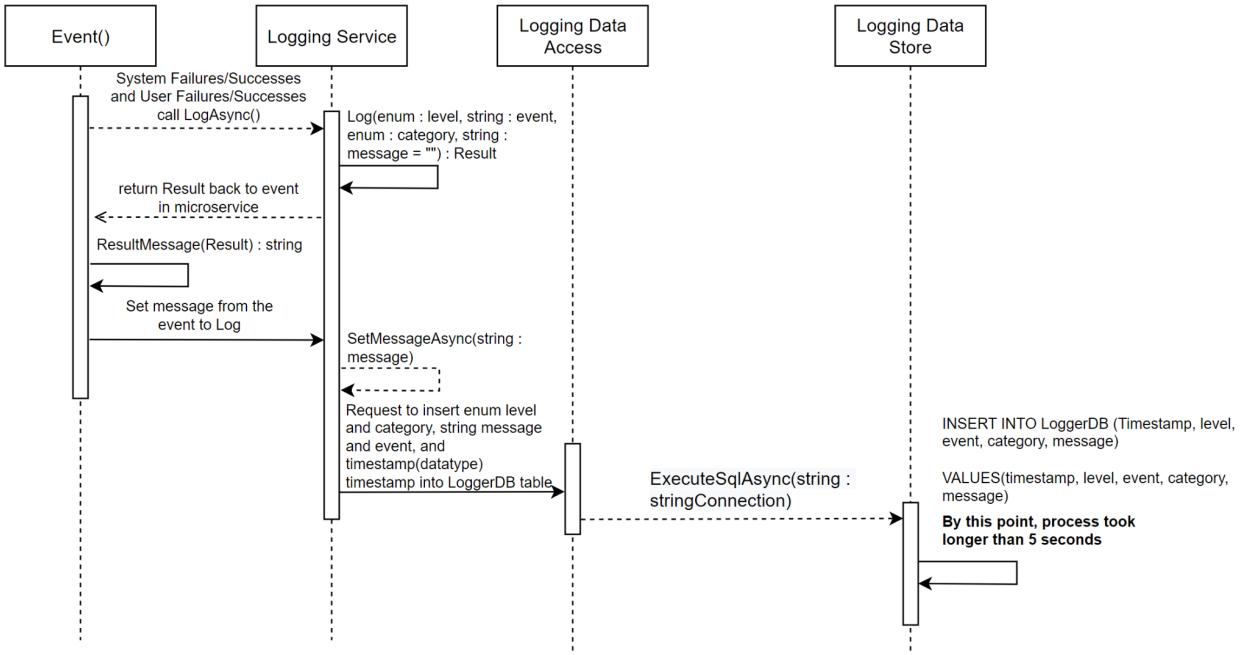
Logging

Logging Success Case

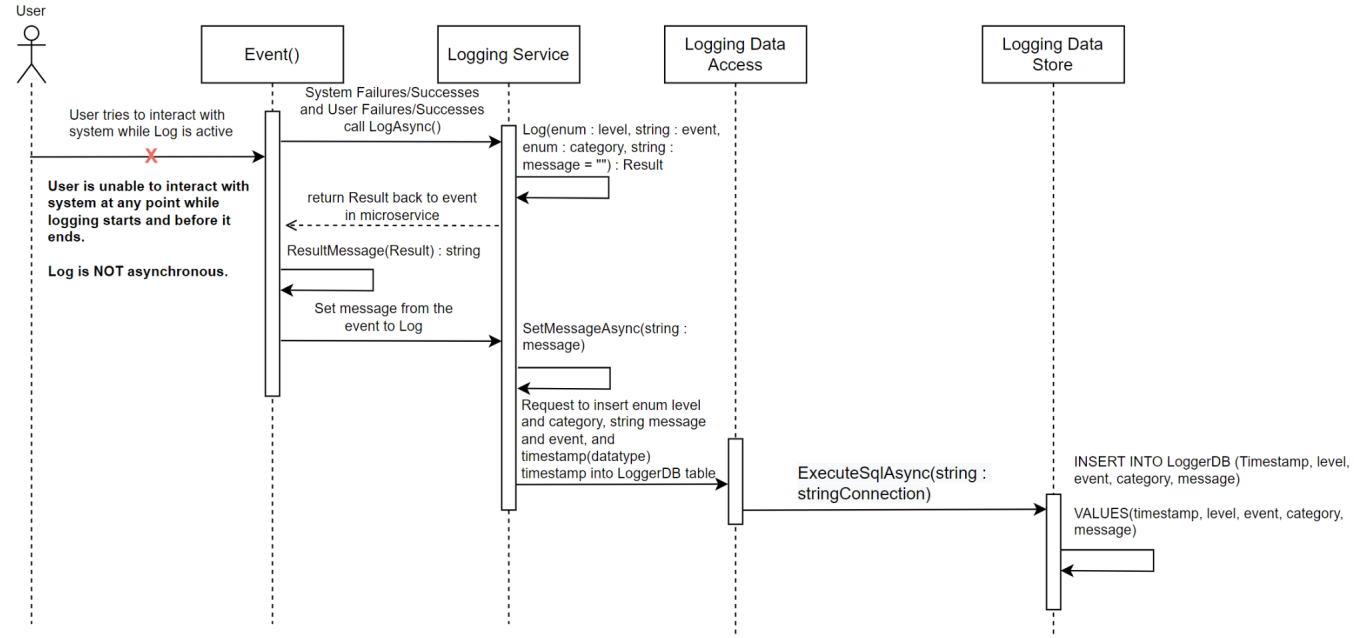


- Event() is the actor and can be considered as any functionality that requires Logging. Functionality can determined by a user request or system functions and features.
- LoggerDB will be the name of the table AND in a separate database dedicated to logging.
- ExecuteSqlAsync() will be asynchronous because it only contains executing insert statements to tablename LoggerDB.

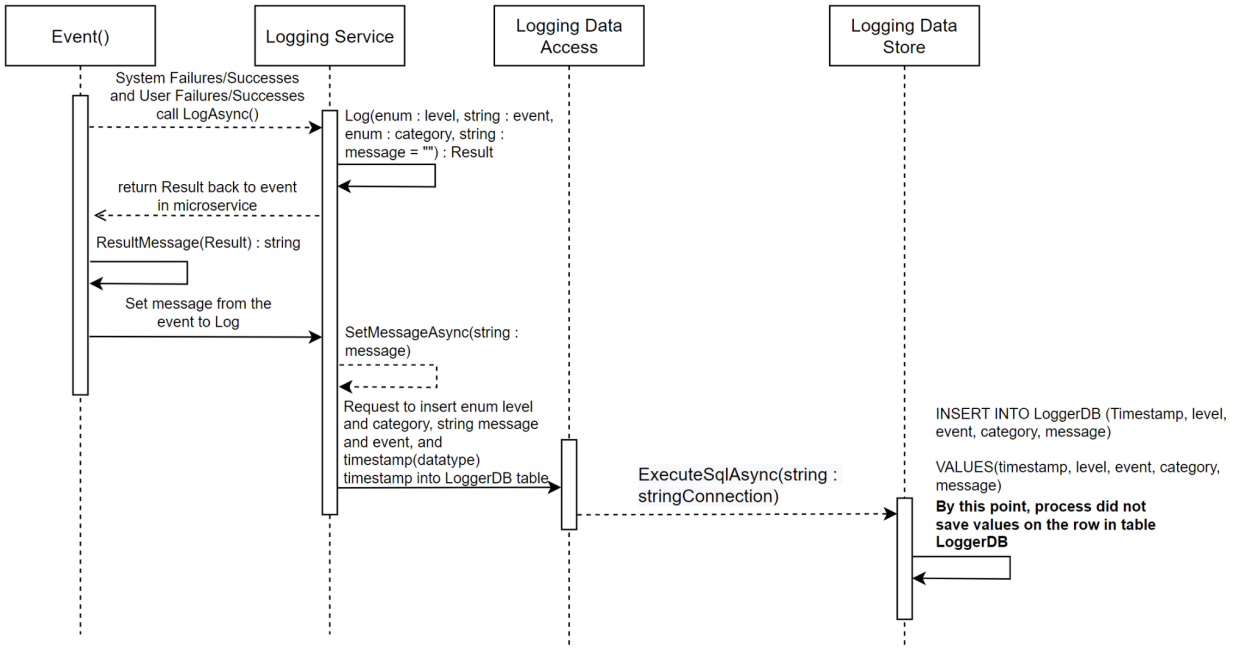
Logging Failure Case by whole process taking longer than 5 seconds



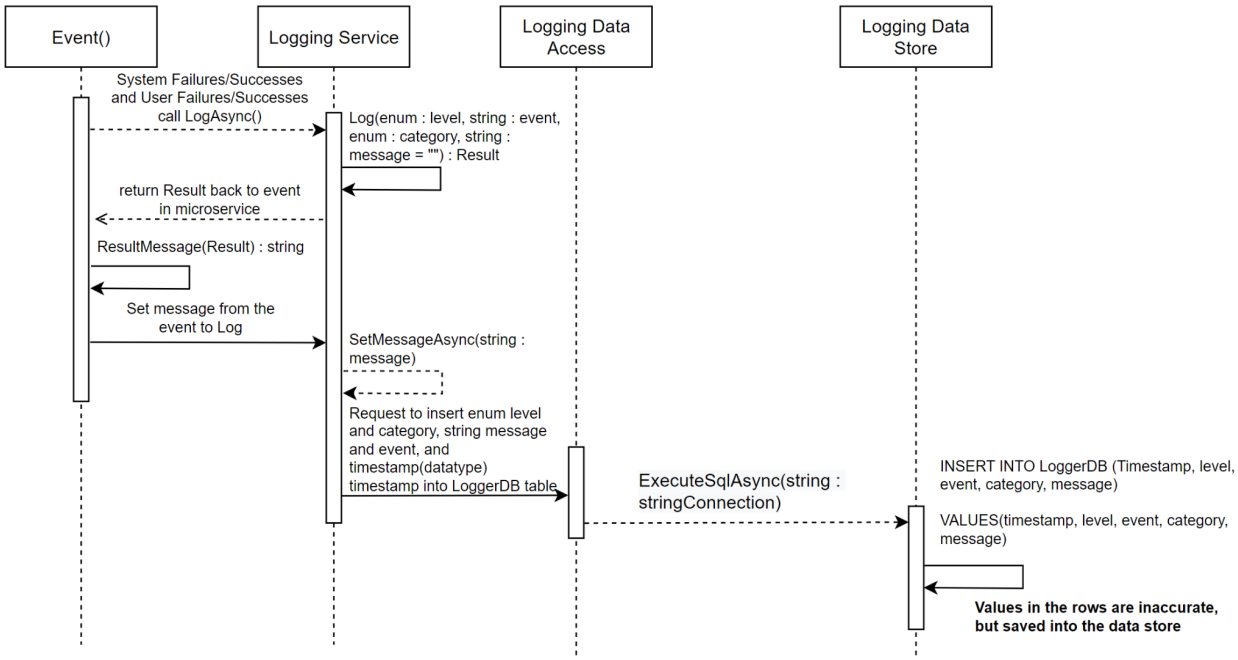
Logging Failure Case by preventing user from interacting with the system



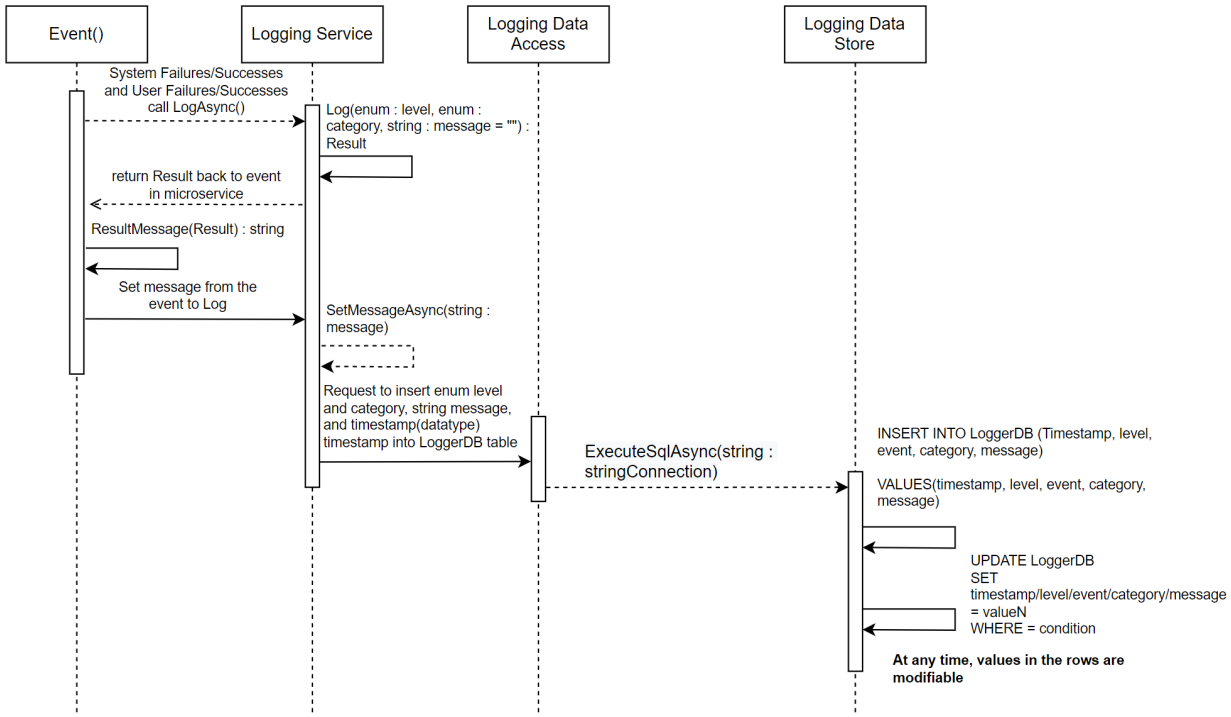
Logging Failure Case by failed to save log in a persistent data store



Logging Failure Case by inaccurately saving the event to a persistent data store

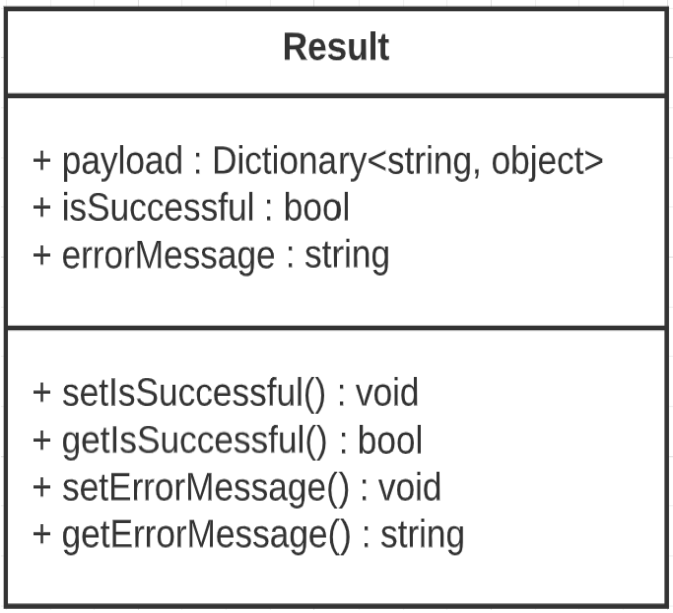


Logging Failure Case by modifiable log entries



- **valueN** is defined as any value that will modify either of the columns shown.
- Ex. SET level = value1, event = value2, category = value3, etc.
- Modifying log entries at any time means it could be outside the sequence diagram as well. It can happen when log is inactive. As long as the persistent storage is on, modifying log entries is **NOT** a success case.

Result



- Result is a class that will be used in every Event() or feature class. The purpose of Result is to confirm with boolean and send an error message if false.
- Ex. Validation to see if something is true, if not then set and display a corresponding error message.
- In C#, use the simplified form : **var { get; set; }**

Glossary

Async (Asynchronous method)	Methods that do not have to wait for an answer

References

Adegeo, et al. “How to Verify That Strings Are in Valid Email Format.” *Microsoft Learn*, 4 Oct. 2022, https://learn.microsoft.com/en-us/dotnet/standard/base-types/how-to-verify-that-strings-are-in-valid-email-format.

“Asynchronous Method Call.” *Techopedia.com*, 18 Aug. 2011, https://www.techopedia.com/definition/25584/asynchronous-method-call#:~:text=An%20asynchronous%20method%20runs%20in,resources%20resulting%20in%20scalable%20application

Corey, Tim. *Logging in .NET Core 3.0 and Beyond - Configuration, Setup, and More*. *YouTube*, 26 Aug. 2019, https://youtu.be/oXNslgIXIbQ. Accessed 1 Nov. 2022

“How to Send and Receive JSON Data to and from the Server.” *Webucator*, https://www.webucator.com/article/how-to-send-and-receive-json-data-to-and-from-the/.

“Low Level Design Template.” *Government of Nepal Department of Information Technology*, https://doit.gov.np/ckfinder/userfiles/files/GEA/Additional%20Aritfacts/Additional%20Aritfacts%201/Low%20Level%20Design%20Template.pdf.

Malek, Piotr. “How to Validate an Email Address in C#.” *Mailtrap*, 28 Feb. 2022, https://mailtrap.io/blog/validate-email-address-c/.

“SQL - Update Query.” *Tutorials Point*, https://www.tutorialspoint.com/sql/sql-update-query.htm.

Vatanik Vong, Lecture on Logging, CECS 491A Sec 04, CSULB, October 26, 2022.

Vatanik Vong, Lecture on Web and UML, CECS 491A Sec 04, CSULB, October 10, 2022.