LOW LEVEL DESIGN DOCUMENT

TEAM NAME: PIXEL PALS

TEAM MEMBERS: BRANDON GALICH
CARINE GORDILLO
SARAH SANTOS
JASON LAM
SARAH PHAN

TEAM LEADER: KAY KAYALE

DATE: NOVEMBER 22, 2023

GITHUB: GITHUB.COM/CARINEGORDILLO/CECS491

Version History

Version	Date	Changes
1	November 22, 2023	Initial submission

Table of Contents

Version History..2

Introduction... 4

Logging Sequence Diagrams...................................... 5

Logging Success:.. 5

Logging Failure - Log LogEntry..................................... 7

Logging Failure - InputValidation.................................... 9

Logging Failure - Time Out:... 11

Data Access Sequence Diagrams.............................. 12

Data Access Successful CREATE/DELETE/UPDATE: 12

Data Access Success - Read:..................................... 14

Data Access Failure - Create/Delete/Update:............. 17

Data Access Failure - Open Connection:.................... 18

Introduction

The purpose of this document is to provide a detailed overview of how each feature in our web application functions. Each feature will have its own diagram showing the flow of data through each layer that feature will go through. Our diagrams will address all success and failure outcomes so that all scenarios regarding that feature will be accounted for. With the help of these diagrams, developers and clients will gain a better understanding of how each feature will work and interact with the different layers that allow our application to function.

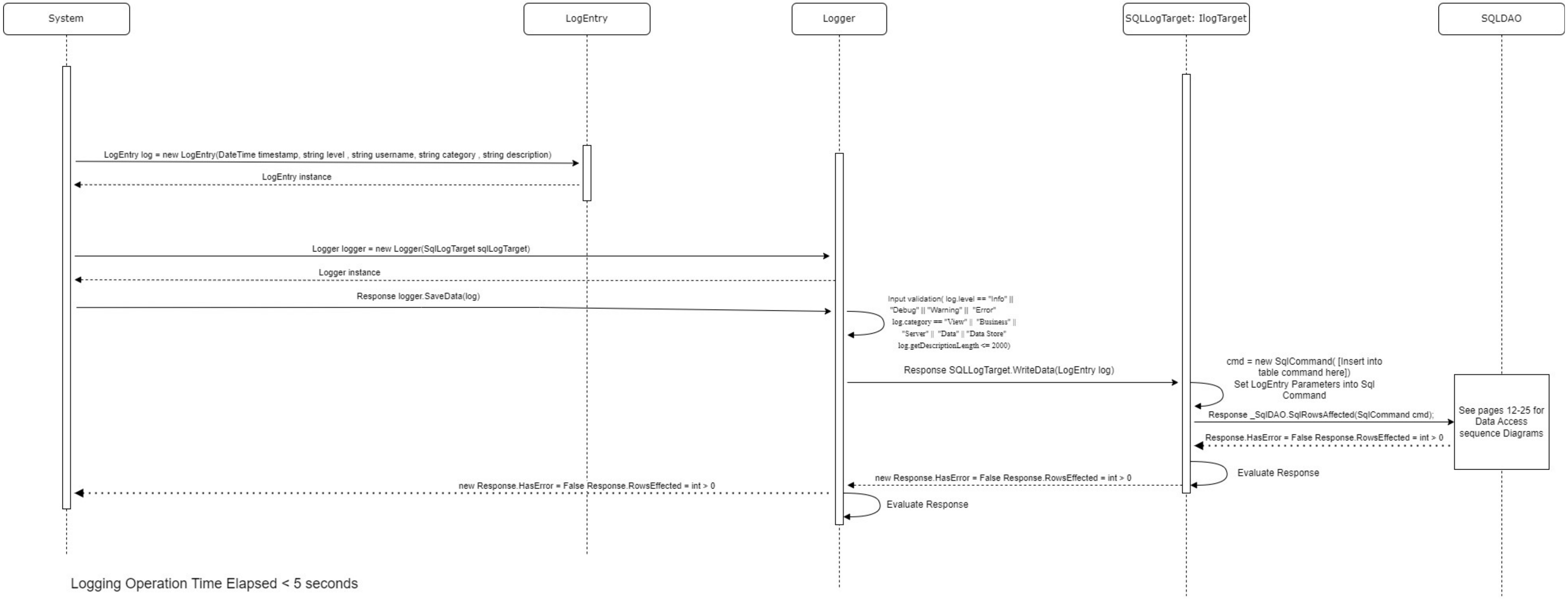
Logging Sequence Diagrams

Logging Success:

Objective: Display a successful scenario of logging an immutable LogEntry into a SQL database.

- **System:** This is the starting point of the logging operation. It's the broader environment or context in which the logging process is initiated.
    - Creates a LogEntry object with the timestamp, level, username, category, and description of the log to the LogEntry layer
        - Return: Not applicable, as this is the start of the process
    - Create new Logger object that takes in the SqlLogTarget
        - Return: Not applicable, as this is the start of the process
- **LogEntry:** A class that represents the specific details of a log entry, such as a timestamp, level, username, category, and description.
    - Encapsulates the details of the LogEntry object
        - Return: Log Entry instance containing log details
- **Logger:** This object is responsible for handling the logger objects.
    - Validate LogEntry inputs
    - Ensures that it uses the WriteData function for that specific log target
    - Evaluates the Response from the SqlLogTarget to determine it's own response.
        - Return: Return: new Response object based on evaluation
- **SQLLogTarget:** This is an abstraction layer that the Logger uses to target a specific logging destination.
    - Creates a new Insert Sql command to send to DAO with the corresponding LogEntry parameters
    - Evaluates response received from the DAO
        - Return: new Response object based on evaluation
- **SQLDAO (Data Access Object):** This layer abstracts and encapsulates all access to the data source. The DAO manages the connection to the data source to obtain and store data.
- **Immutable**
    - The architecture of our logging service ensures that all logs are immutable since the only way a log can have access to the data store is through SqlLogTarget. The only permission that SqlLogTarget allows is for data to be written, so there would be no way for a log to be updated or deleted.

LOGGING to SQLDAO - SUCCESS


System
LogEntry
Logger
SQLLogTarget: IlogTarget
SQLDAO
LogEntry log = new LogEntry(DateTime timestamp, string level , string username, string category , string description)
LogEntry instance
Logger logger = new Logger(SqlLogTarget sqlLogTarget)
Logger instance
Response logger.SaveData(log)
Input validation(log.level == "Info" \|\| "Debug" \|\| "Warning" \|\| "Error" log.category == "View" \|\| "Business" \|\| "Server" \|\| "Data" \|\| "Data Store" log.getDescriptionLength <= 2000)
Response SQLLogTarget.WriteData(LogEntry log)
cmd = new SqlCommand([Insert into table command here]) Set LogEntry Parameters into Sql Command
Response _SqlDAO.SqlRowsAffected(SqlCommand cmd);
See pages 12-25 for Data Access sequence Diagrams
Response.HasError = False Response.RowsEffected = int > 0
Evaluate Response
new Response.HasError = False Response.RowsEffected = int > 0
new Response.HasError = False Response.RowsEffected = int > 0
Evaluate Response

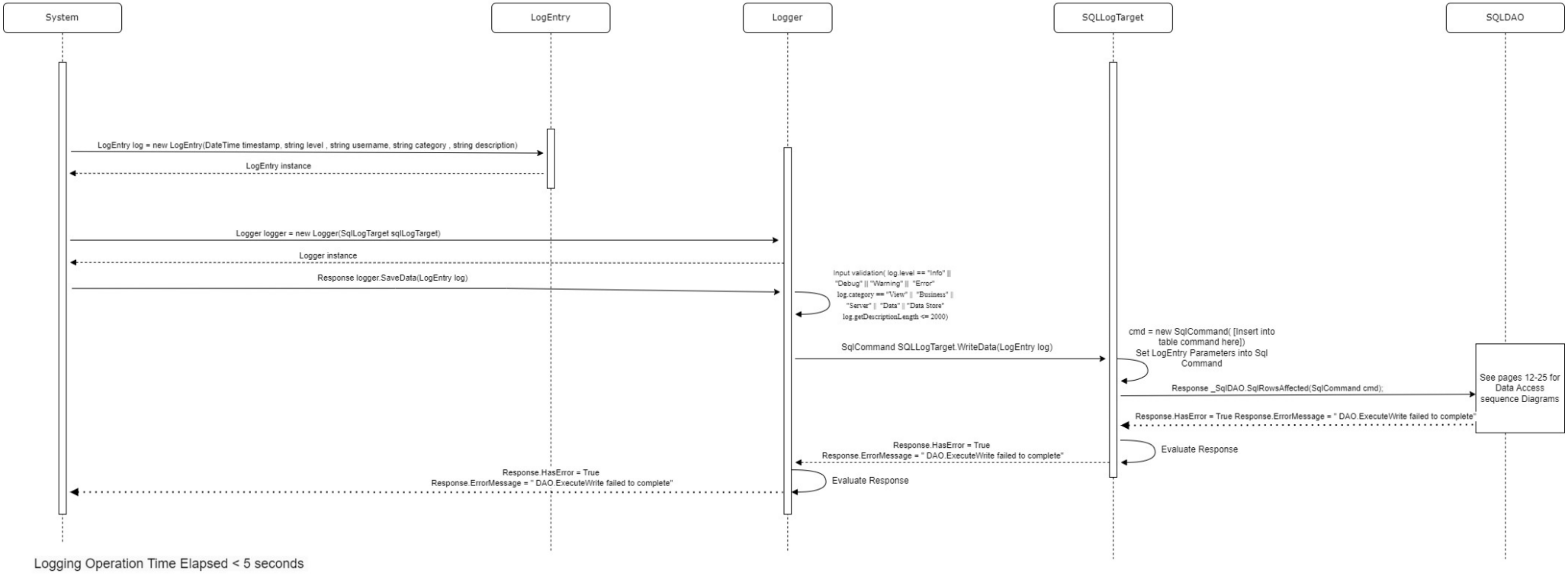
Logging Operation Time Elapsed < 5 seconds

Logging Failure - Log LogEntry

Objective: Display a failure scenario to write to SQL DAO

- **System:** This is the starting point of the logging operation. It's the broader environment or context in which the logging process is initiated.
  - Creates a LogEntry object with the timestamp, level, username, category, and description of the log to the LogEntry layer
    - Return: Not applicable, as this is the start of the process
  - Create new Logger object that takes in the SqlLogTarget
    - Return: Not applicable, as this is the start of the process
- **LogEntry:** A class that represents the specific details of a log entry, such as a timestamp, level, username, category, and description.
  - Encapsulates the details of the LogEntry object
    - Return: Log Entry instance containing log details
- **Logger:** This object is responsible for handling the logger objects.
  - Validate LogEntry inputs
  - Ensures that it uses the WriteData function for that specific log target
  - Evaluates the Response from the SqlLogTarget to determine it's own response.
    - Return: Return: new Response object based on evaluation
- **SQLLogTarget:** This is an abstraction layer that the Logger uses to target a specific logging destination.
  - Creates a new Insert Sql command to send to DAO with the corresponding LogEntry parameters
  - Evaluates response received from the DAO
    - Return: new Response object based on evaluation
- **SQLDAO (Data Access Object):** This layer abstracts and encapsulates all access to the data source. The DAO manages the connection to the data source to obtain and store data.
  - The DAO tries to execute the SQL command to write the LogEntry to the database. If the command execution fails, the DAO would not be able to store the log.
    - Return: An error response object indicating the failure of the log entry to be written to the database due to an error in the data access process.
- **Immutable**
  - The architecture of our logging service ensures that all logs are immutable since the only way a log can have access to the data store is through SqlLogTarget. The only permission that SqlLogTarget allows is for data to be written, so there would be no way for a log to be updated or deleted.

LOGGING to SQLDAO - Failure.ExecuteWrite


System
LogEntry
Logger
SQLLogTarget
SQLDAO
LogEntry log = new LogEntry(DateTime timestamp, string level , string username, string category , string description)
LogEntry instance
Logger logger = new Logger(SqlLogTarget sqlLogTarget)
Logger instance
Response logger.SaveData(LogEntry log)
Input validation(log.level == "Info" || "Debug" || "Warning" || "Error" log.category == "View" || "Business" || "Server" || "Data" || "Data Store" log.getDescriptionLength <= 2000)
SqlCommand SQLLogTarget.WriteData(LogEntry log)
cmd = new SqlCommand([Insert into table command here]) Set LogEntry Parameters into Sql Command
Response _SqlDAO.SqlRowsAffected(SqlCommand cmd);
See pages 12-25 for Data Access sequence Diagrams
Response.HasError = True Response.ErrorMessage = " DAO.ExecuteWrite failed to complete"
Evaluate Response
Response.HasError = True
Response.ErrorMessage = " DAO.ExecuteWrite failed to complete"
Response.HasError = True
Response.ErrorMessage = " DAO.ExecuteWrite failed to complete"
Evaluate Response

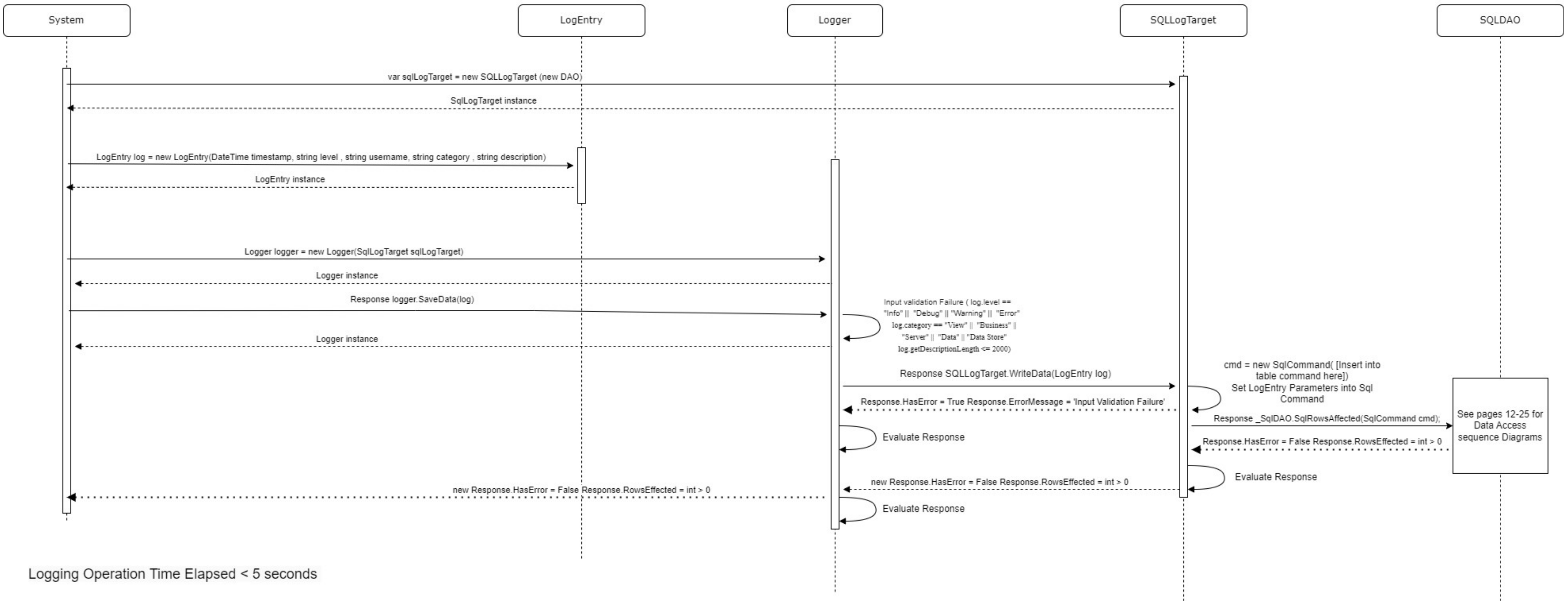
Logging Operation Time Elapsed < 5 seconds

Logging Failure - InputValidation

Objective: Display a failure scenario to receive a valid LogEntry

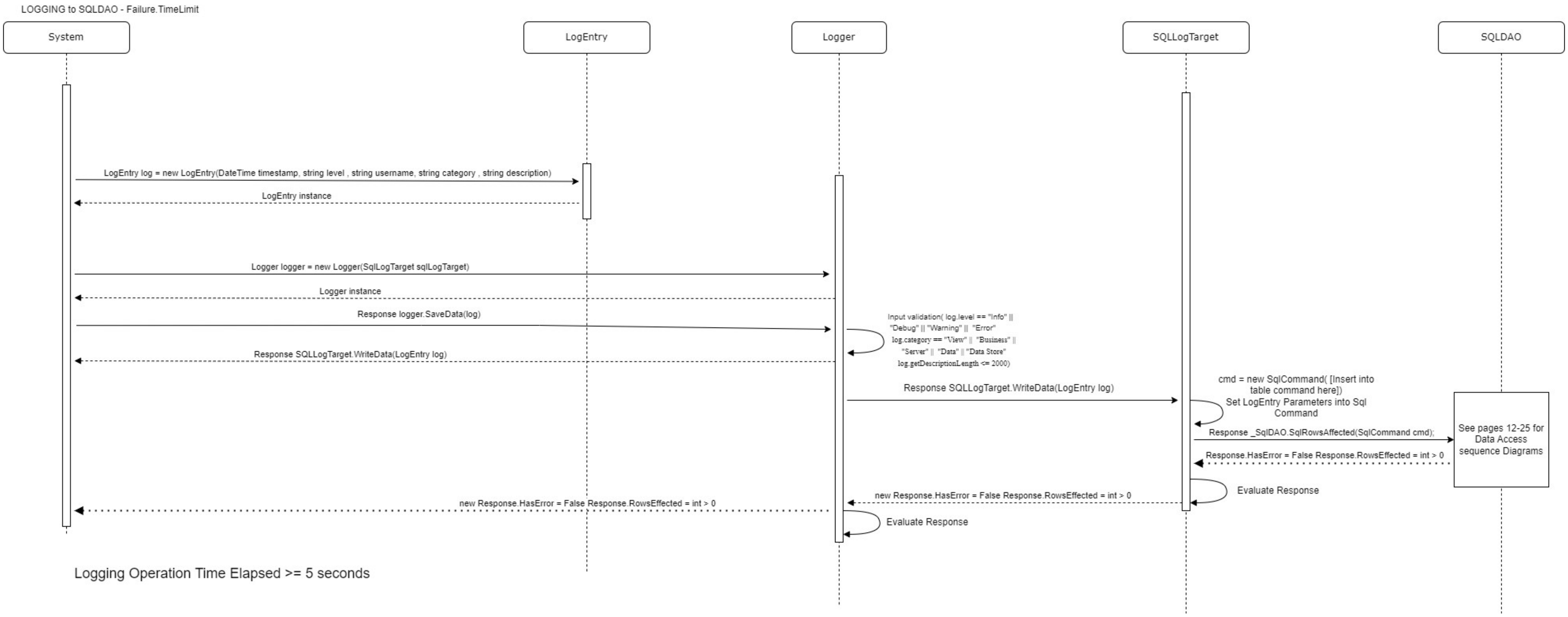
- **System:** This is the starting point of the logging operation. It's the broader environment or context in which the logging process is initiated.
    - Creates a LogEntry object with the timestamp, level, username, category, and description of the log to the LogEntry layer
        - Return: Not applicable, as this is the start of the process
    - Create new Logger object that takes in the SqlLogTarget
        - Return: Not applicable, as this is the start of the process
- **LogEntry:** A class that represents the specific details of a log entry, such as a timestamp, level, username, category, and description.
    - Encapsulates the details of the LogEntry object
        - Return: Log Entry instance containing log details
- **Logger:** This object is responsible for handling the logger objects.
    - Encapsulates the logger object
    - Validate LogEntry inputs
        - If input validation fails, the Logger generates a new Response object encapsulating the error, which prevents the log from being written. The system has up to 3 retries until the entire operation fails out.
    - Ensures that it uses the WriteData function for that specific log target
    - Evaluates the Response from the SqlLogTarget to determine its own response.
        - Return: Return: new Response object based on evaluation
- **SQLLogTarget:** This is an abstraction layer that the Logger uses to target a specific logging destination.
    - Creates a new Insert Sql command to send to DAO with the corresponding LogEntry parameters
    - Evaluates response received from the DAO
        - Return: new Response object based on evaluation
- **SQLDAO (Data Access Object):** This layer abstracts and encapsulates all access to the data source. The DAO manages the connection to the sql database to obtain and store data.
- **Immutable**
    - The architecture of our logging service ensures that all logs are immutable since the only way a log can have access to the data store is through SqlLogTarget. The only permission that SqlLogTarget allows is for data to be written, so there would be no way for a log to be updated or deleted.

LOGGING to SQLDAO - Failure.InputValidation


System
LogEntry
Logger
SQLLogTarget
SQLDAO
var sqlLogTarget = new SQLLogTarget (new DAO)
SqlLogTarget instance
LogEntry log = new LogEntry(DateTime timestamp, string level , string username, string category , string description)
LogEntry instance
Logger logger = new Logger(SqlLogTarget sqlLogTarget)
Logger instance
Response logger.SaveData(log)
Input validation Failure (log.level == "Info" || "Debug" || "Warning" || "Error" log.category == "View" || "Business" || "Server" || "Data" || "Data Store" log.getDescriptionLength <= 2000)
Logger instance
Response SQLLogTarget.WriteData(LogEntry log)
cmd = new SqlCommand([Insert into table command here]) Set LogEntry Parameters into Sql Command
Response.HasError = True Response.ErrorMessage = 'Input Validation Failure'
Response _SqlDAO.SqlRowsAffected(SqlCommand cmd);
See pages 12-25 for Data Access sequence Diagrams
Evaluate Response
Response.HasError = False Response.RowsEffected = int > 0
Evaluate Response
new Response.HasError = False Response.RowsEffected = int > 0
new Response.HasError = False Response.RowsEffected = int > 0
Evaluate Response

Logging Operation Time Elapsed < 5 seconds

Logging Failure - Time Out:

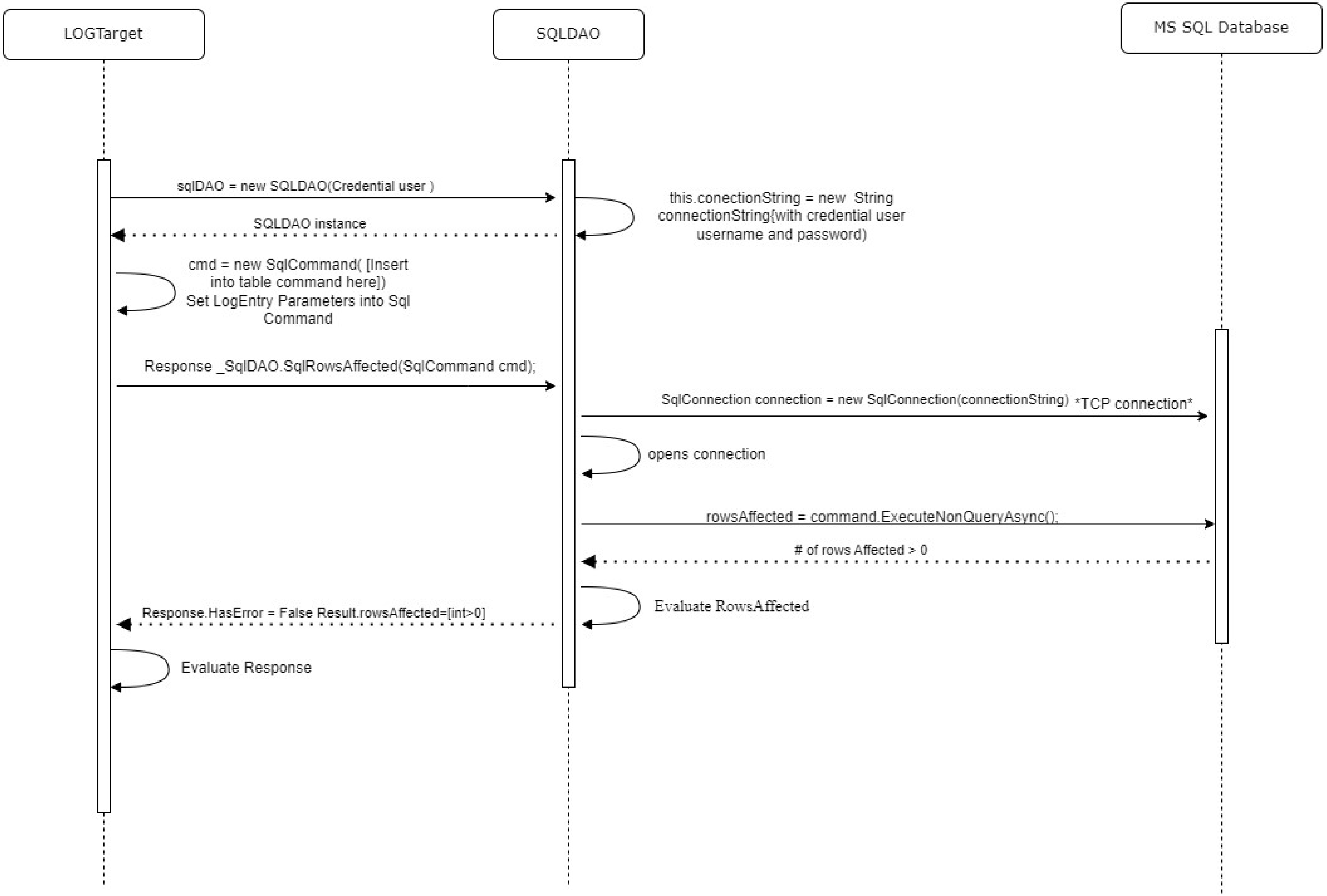

LOGGING to SQLDAO - Failure.TimeLimit
System
LogEntry
Logger
SQLLogTarget
SQLDAO
LogEntry log = new LogEntry(DateTime timestamp, string level , string username, string category , string description)
LogEntry instance
Logger logger = new Logger(SqlLogTarget sqlLogTarget)
Logger instance
Response logger.SaveData(log)
Input validation(log.level == "Info" || "Debug" || "Warning" || "Error" log.category == "View" || "Business" || "Server" || "Data" || "Data Store" log.getDescriptionLength <= 2000)
Response SQLLogTarget.WriteData(LogEntry log)
Response SQLLogTarget.WriteData(LogEntry log)
cmd = new SqlCommand([Insert into table command here]) Set LogEntry Parameters into Sql Command
Response _SqlDAO.SqlRowsAffected(SqlCommand cmd);
See pages 12-25 for Data Access sequence Diagrams
Response.HasError = False Response.RowsEffected = int > 0
Evaluate Response
new Response.HasError = False Response.RowsEffected = int > 0
new Response.HasError = False Response.RowsEffected = int > 0
Evaluate Response
Logging Operation Time Elapsed >= 5 seconds

Data Access Sequence Diagrams

Data Access Successful CREATE/DELETE/UPDATE:

- **LogTarget:** This is the starting point of the Data Access operation. It represents a layer that is able to create a Data Access Object to access our MS Sql Database. The Log Target in the example is sqlLogTarget which Logs a LogEntry.
    - Creates sqlDAO given the credentials of the user and creates the connection string
        - Return: Not applicable, as this is the start of the process
    - Creates a new SqlCommand based on operation. sqlLogTarget only has write so the sqlCommand would look like this "INSERT INTO dbo.Logs VALUES (@Timestamp, @LogLevel, @Username, @Category, @Description); SELECT SCOPE_IDENTITY();"
    - Adds the Data from the LogEntry into the sql statement
    - Uses method to go into the SqlDAO and begin the execution process : SqlRowsAffected(command)
    - Evaluates the Response from the SQLDAO to determine its own response. Response will include a property for RowsAffected which will store the count of rows affected on the table as a result of the sql command being executed.
        - Return: new Response object based on evaluation
- **SQLDAO: (Data Access Object):** This layer abstracts and encapsulates all access to the data source. The DAO manages the connection to the sql database to obtain and store data.
    - Establishes a TCP Sql Connection to the Database
    - Opens Connection to the database specified by the connection string
        - Evaluates the Response from the MS SQL Data Base to determine its own response.
            - Return: Return: new Response object based on evaluation
    - Responsible for executing the sql command
- **MS SQL DataBase:** Stores Tables for out data
    - Contains restrictions such as values unable to be null
    - Data store permissions based on credentials

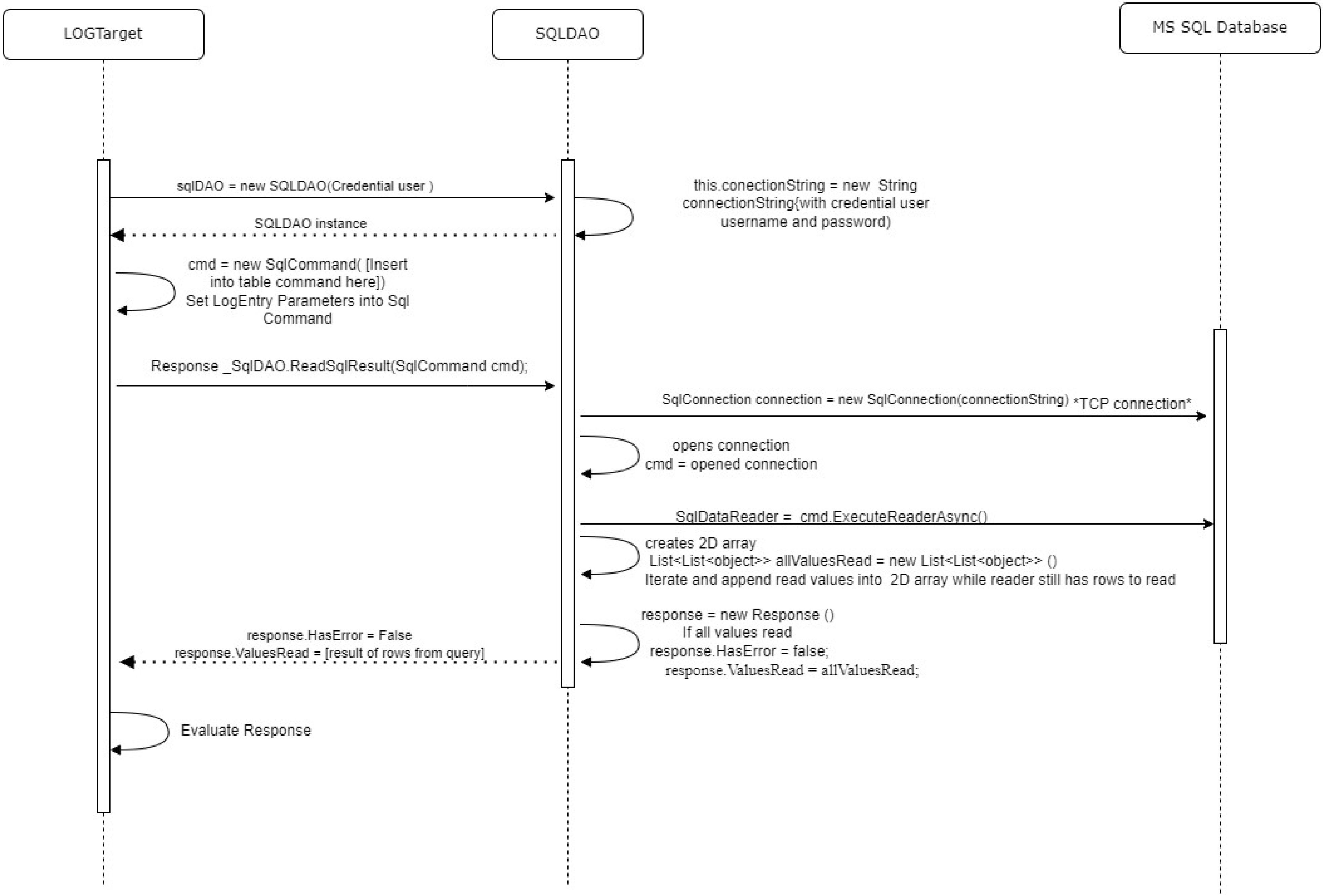
DATA ACCESS - SUCCESS CREATE/DELETE/UPDATE


LOGTarget
SQLDAO
MS SQL Database
sqlDAO = new SQLDAO(Credential user)
this.conectionString = new String connectionString{with credential user username and password)
SQLDAO instance
cmd = new SqlCommand([Insert into table command here]) Set LogEntry Parameters into Sql Command
Response _SqlDAO.SqlRowsAffected(SqlCommand cmd);
SqlConnection connection = new SqlConnection(connectionString) *TCP connection*
opens connection
rowsAffected = command.ExecuteNonQueryAsync();
# of rows Affected > 0
Evaluate RowsAffected
Response.HasError = False Result.rowsAffected=[int>0]
Evaluate Response

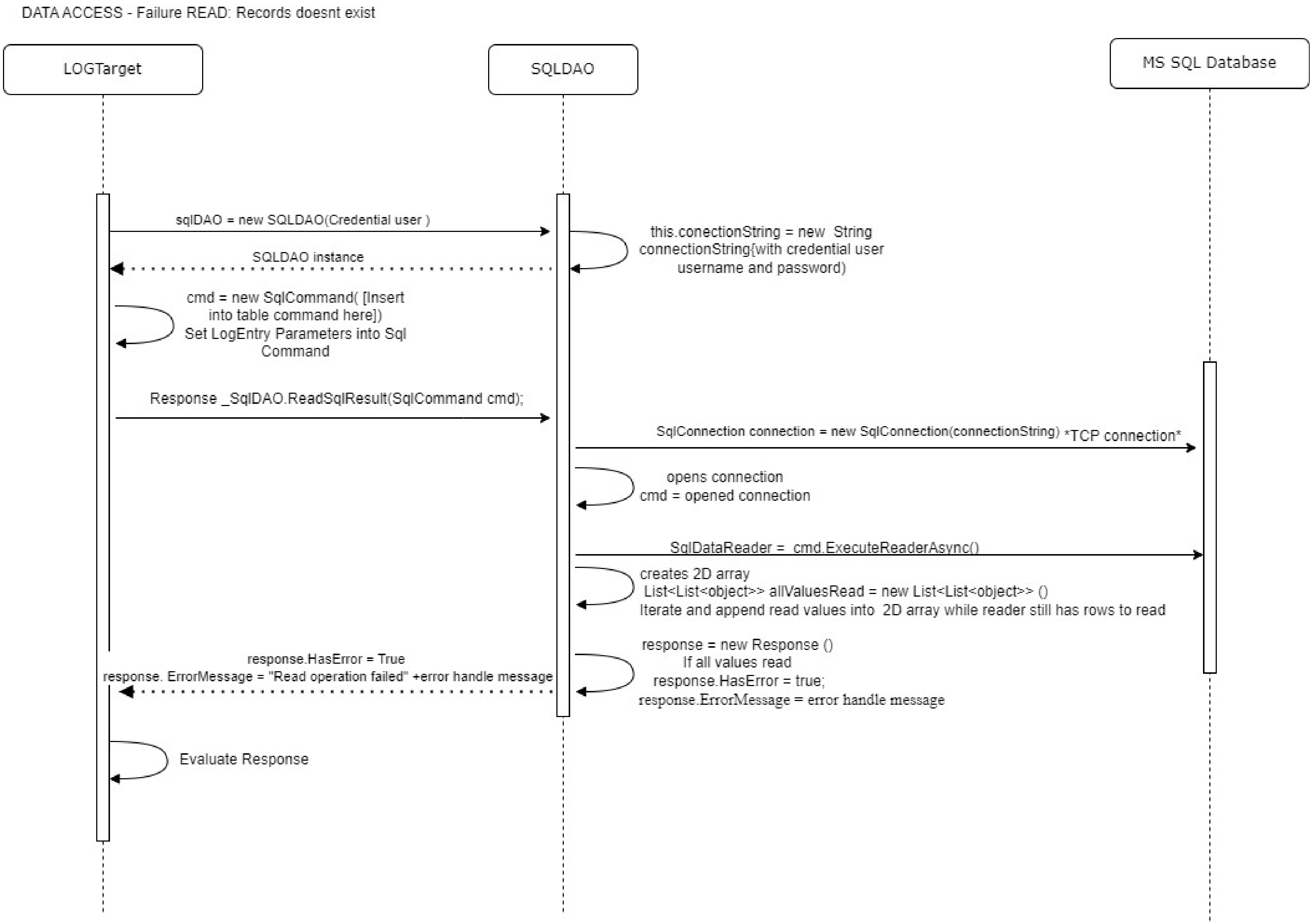
Data Access Success - Read:

- **LogTarget:** This is the starting point of the Data Access operation. It represents a layer that is able to create a Data Access Object to access our MS Sql Database. The Log Target in the example is sqlLogTarget which Logs a LogEntry.
    - Creates sqlDAO given the credentials of the user and creates the connection string
        - Return: Not applicable, as this is the start of the process
    - Creates a new SqlCommand based on operation. sqlLogTarget only has write so the sqlCommand would look like this "INSERT INTO dbo.Logs VALUES (@Timestamp, @LogLevel, @Username, @Category, @Description); SELECT SCOPE_IDENTITY();"
    - Adds the Data from the LogEntry into the sql statement
    - Uses method to go into the SqlDAO and begin the execution process : ReadSqlResult(command)
    - Evaluates the Response from the SQLDAO to determine its own response.
        - Return: Return: new Response object based on evaluation. Response includes a property for ValuesRead of type List<List<object>>. This will store the values read from the sql query executed.
- **SQLDAO: (Data Access Object):** This layer abstracts and encapsulates all access to the data source. The DAO manages the connection to the sql database to obtain and store data.
    - Establishes a TCP Sql Connection to the Database
    - Opens Connection to the database specified by the connection string
        - Evaluates the Response from the MS SQL Data Base to determine its own response.
            - Return: new Response object based on evaluation
- **MS SQL DataBase:** Stores Tables for out data
    - Contains restrictions such as values unable to be null
    - Data store permissions based on credentials

DATA ACCESS - SUCCESS READ


LOGTarget
SQLDAO
MS SQL Database
sqlDAO = new SQLDAO(Credential user)
this.conectionString = new String connectionString{with credential user username and password)
SQLDAO instance
cmd = new SqlCommand([Insert into table command here])
Set LogEntry Parameters into Sql Command
Response _SqlDAO.ReadSqlResult(SqlCommand cmd);
SqlConnection connection = new SqlConnection(connectionString) *TCP connection*
opens connection
cmd = opened connection
SqlDataReader = cmd.ExecuteReaderAsync()
creates 2D array
List<List<object>> allValuesRead = new List<List<object>> ()
Iterate and append read values into 2D array while reader still has rows to read
response = new Response ()
If all values read
response.HasError = false;
response.ValuesRead = allValuesRead;
response.HasError = False
response.ValuesRead = [result of rows from query]
Evaluate Response

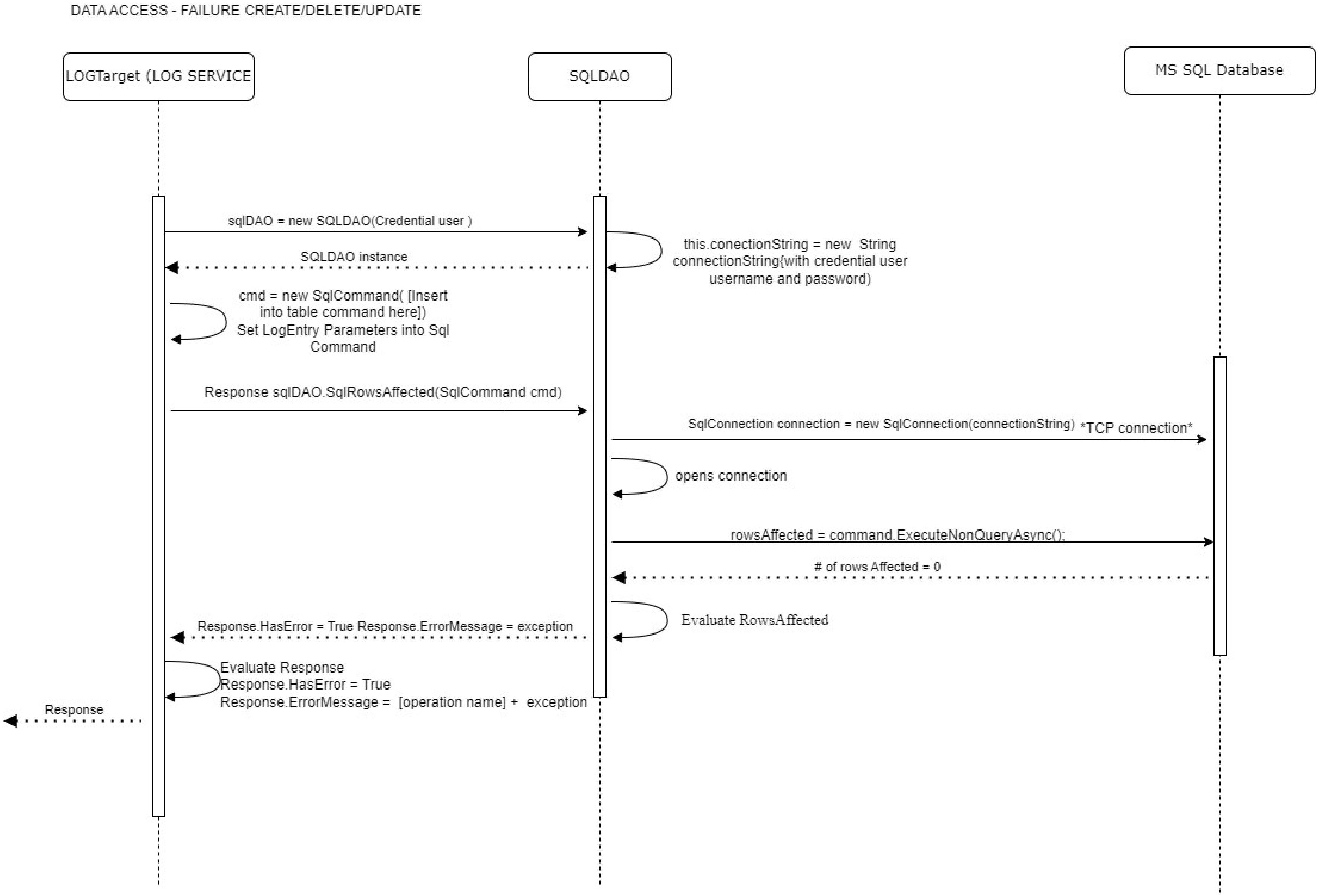
Data Access Failure - Read:



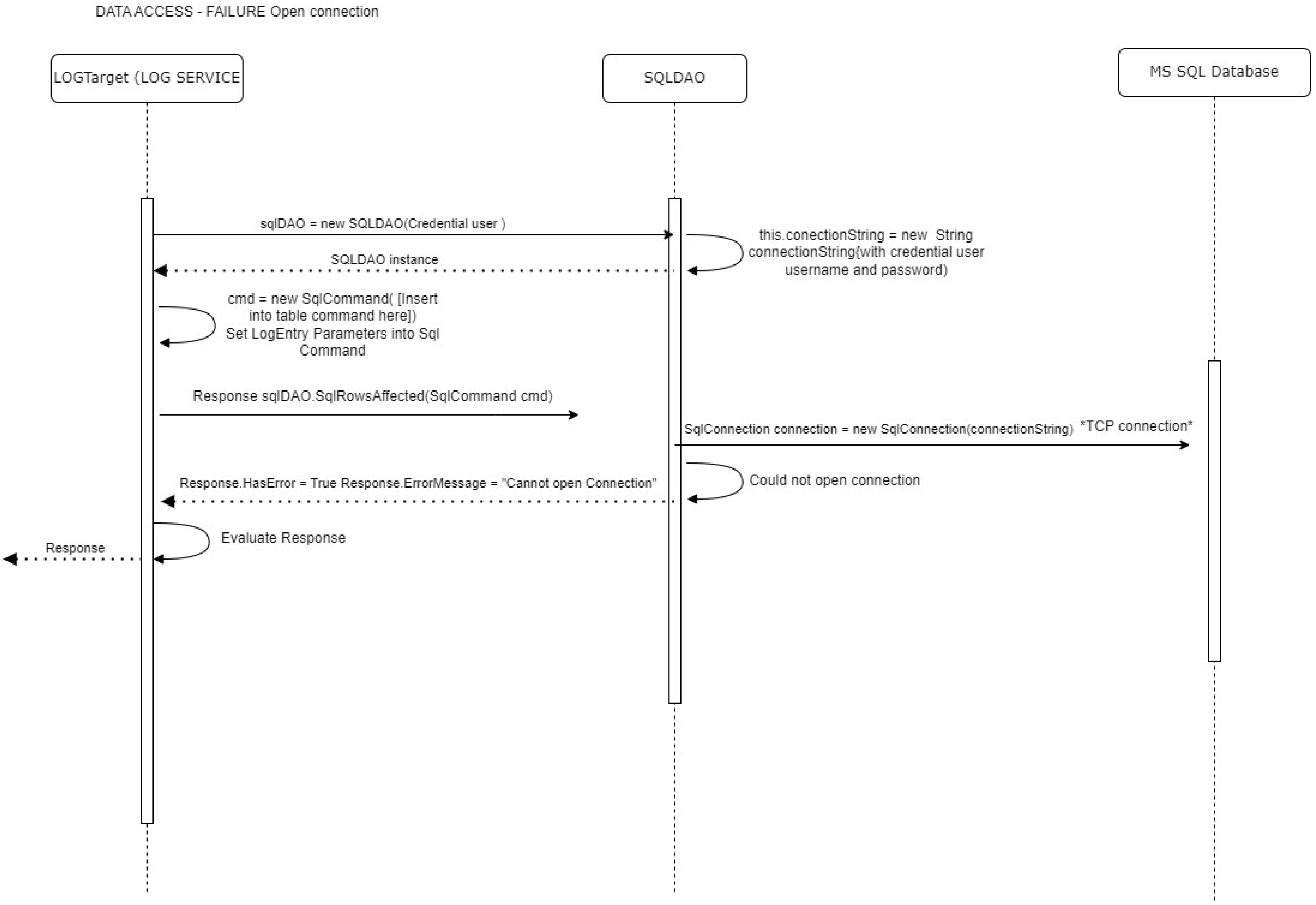
DATA ACCESS - Failure READ: Records doesnt exist

LOGTarget
SQLDAO
MS SQL Database
sqlDAO = new SQLDAO(Credential user)
this.conectionString = new String connectionString{with credential user username and password)
SQLDAO instance
cmd = new SqlCommand([Insert into table command here])
Set LogEntry Parameters into Sql Command
Response _SqlDAO.ReadSqlResult(SqlCommand cmd);
SqlConnection connection = new SqlConnection(connectionString) *TCP connection*
opens connection
cmd = opened connection
SqlDataReader = cmd.ExecuteReaderAsync()
creates 2D array
List<List<object>> allValuesRead = new List<List<object>> ()
Iterate and append read values into 2D array while reader still has rows to read
response = new Response ()
If all values read
response.HasError = true;
response.ErrorMessage = error handle message
response.HasError = True
response. ErrorMessage = "Read operation failed" +error handle message
Evaluate Response

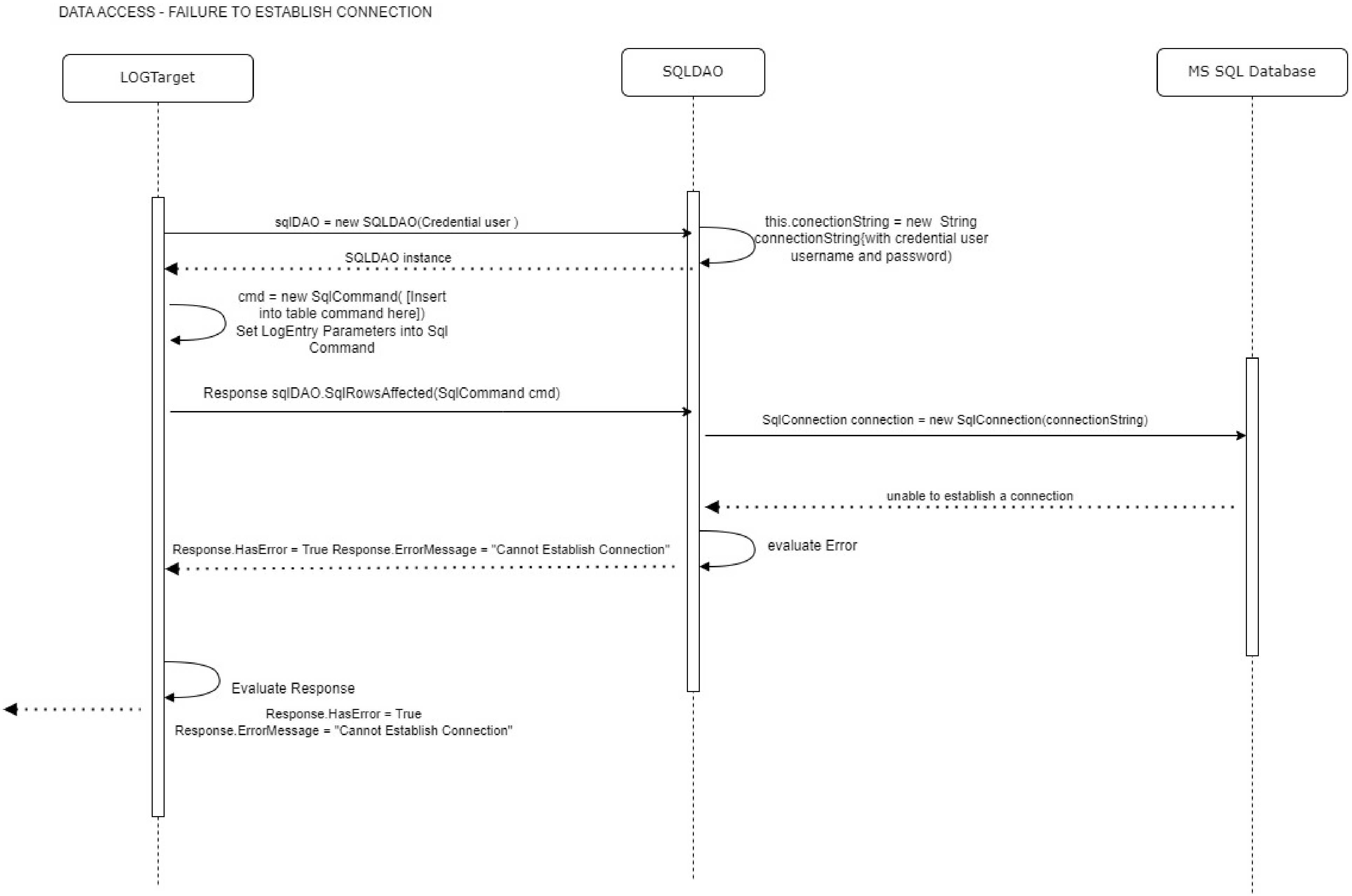
Data Access Failure - Create/Delete/Update:


DATA ACCESS - FAILURE CREATE/DELETE/UPDATE
LOGTarget (LOG SERVICE
SQLDAO
MS SQL Database
sqlDAO = new SQLDAO(Credential user)
this.conectionString = new String connectionString{with credential user username and password)
SQLDAO instance
cmd = new SqlCommand([Insert into table command here]) Set LogEntry Parameters into Sql Command
Response sqlDAO.SqlRowsAffected(SqlCommand cmd)
SqlConnection connection = new SqlConnection(connectionString) *TCP connection*
opens connection
rowsAffected = command.ExecuteNonQueryAsync();
# of rows Affected = 0
Evaluate RowsAffected
Response.HasError = True Response.ErrorMessage = exception
Evaluate Response
Response.HasError = True
Response.ErrorMessage = [operation name] + exception
Response

Data Access Failure - Open Connection:


DATA ACCESS - FAILURE Open connection
LOGTarget (LOG SERVICE
SQLDAO
MS SQL Database
sqlDAO = new SQLDAO(Credential user)
this.conectionString = new String connectionString{with credential user username and password)
SQLDAO instance
cmd = new SqlCommand([Insert into table command here]) Set LogEntry Parameters into Sql Command
Response sqlDAO.SqlRowsAffected(SqlCommand cmd)
SqlConnection connection = new SqlConnection(connectionString)
TCP connection
Could not open connection
Response.HasError = True Response.ErrorMessage = "Cannot open Connection"
Evaluate Response
Response

Data Access Failure - Establish Connection:


DATA ACCESS - FAILURE TO ESTABLISH CONNECTION
LOGTarget
SQLDAO
MS SQL Database
sqlDAO = new SQLDAO(Credential user)
this.conectionString = new String connectionString(with credential user username and password)
SQLDAO instance
cmd = new SqlCommand([Insert into table command here]) Set LogEntry Parameters into Sql Command
Response sqlDAO.SqlRowsAffected(SqlCommand cmd)
SqlConnection connection = new SqlConnection(connectionString)
unable to establish a connection
evaluate Error
Response.HasError = True Response.ErrorMessage = "Cannot Establish Connection"
Evaluate Response
Response.HasError = True
Response.ErrorMessage = "Cannot Establish Connection"