Mining Minds Platform ver. 4.0 (MM4)

Requirements Specification &
Design Document

Principal Investigator

Prof. Sungyoung Lee

Document Editor

Dr. Muhammad Bilal Amin

Project Lead

Dr. Muhammad Bilal Amin

Contributors

DCL Team

Dr. Muhammad Bilal Amin
Bilal Ali, Usman Akhtar, Taqdir Ali,

ICL Team

Tae Ho Hur, Jaehun Bang,

Dr. Thien Huynh The, Muhammad Asif Razzaq

KCL Team

Muhammad Sadiq,
Taqdir Ali, Maqbool Ali,

SCL Team

Syed Imran Ali,
Muhammad Sadiq

SL Team

Shujaat Hussain, Ubaid ur Rehman,
Anees Ul Hassan, Asim Abbas

Facilitators

Professor Byeong Ho Kang,
University of Tasmania,
Australia

Index

Introduction

MMV4 is the 7th major integration for Mining Minds Platform, constituting upon the duration of 6 months, i.e., (Jan 2018 – Jun 2018). MMV4 utilizes the analysis, design and implementation efforts of previous versions, for its benefit and builds on the top of already identified layered abstractions and primary components. The overall architecture of MMV4 is similar to previous iterations; however, components have been added to incorporate newer requirements.

At Data level, Data Curation Layer of MMV4 (DCL), is incorporating not only activity data from smartphone and smart watch, but also acquiring video stream from a camera, making the data input more heterogeneous per user. This data acquisition is purely in real-time with asynchronous, non-blocking communication from data source to the DCL. Furthermore, DCL is also providing the read access to sensory data persisted in big data storage for descriptive analytics and visualization. With the integration of Just-in-time monitoring, DCL is able to provide timely interrupts based on situations.

At Information level, Information Curation Layer of MMV4 (ICL), is incorporating new low-level context awareness mechanisms for the identification of the user location, emotion apart from their physical activities. Moreover, this version also includes high-level context awareness for a reliable and comprehensive determination of the user's context. The communication between DCL and ICL has also been updated in MMV4 as a buffer is introduced between DCL and ICL for incoming sensory data, such that ICL is not stressed by strong influx of data from DCL. Furthermore, ICL only communicates with DCL in response when the context of the user changes, reducing the communication overhead. Knowledge curation, introduced in MMV1.5 is updated in MMV2 as Knowledge Curation Layer (KCL). For data driven knowledge acquisition, this layer now incorporates training of classification models directly from the data persisted as life-log in the intermediate database and big data storage. For expert driven knowledge acquisition, KCL provides production rules for service curation components and derive situations to be hosted at DCL and serviced at high-level service curation.

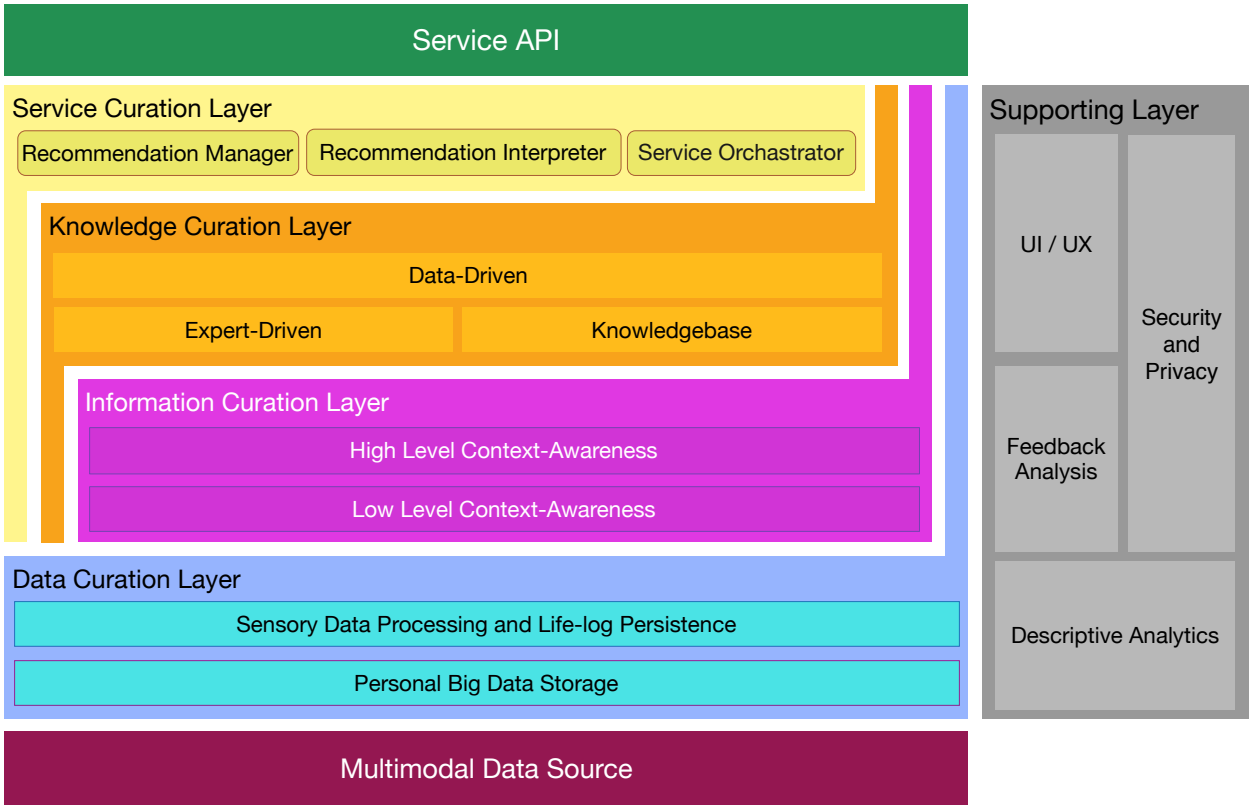
At service level, Service Curation Layer (SCL) is dealing with service request handling, recommendation generation, recommendation interpretations, and service response delivering. It receives service requests from service requester directly or through a trigger based on the events identified in life-log of a user, SCL builds the recommendations based on user profile, lifelog data, and production rules. Based on the context, user characteristics, and environment variables, the recommendation are interpreted, explained, and are delivered to the Supporting Layer i.e. SL in order to serve the service requester.

Supporting layer (SL) deals with providing services to every other layer of MMV4. It is responsible to provide personalized recommendations, trend analytics through adaptive user interface with added services of privacy and security. SL provides security services for securing the communication among different layers in MMV4. It handles the

recommendations from SCL and stores the feedback of the user in DCL. Also, for analysis of the data stored as big data, SL provides services of analytics by communicating with DCL.

This document provides requirement specifications for MMV4 with high-level use cases, sequence, and collaboration diagrams for the implemented platform.

Abstract architecture



1. Data Curation Layer (DCL)

1.1 Functional Requirements

FR ID#	Description
DCL-FR-01	The platform shall read the raw sensory data of the user from his/her data source
DCL-FR-02	The platform shall provide permanent persistence to the user generated raw sensory data
DCL-FR-03	The platform shall provide raw sensory data for context determination of the user
DCL-FR-04	The platform shall maintain user profile data
DCL-FR-05	The platform shall maintain user timeline as a lifelog of daily behaviors
DCL-FR-06	The platform shall provide read, write, delete, and update access to the subscribers of lifelog data
DCL-FR-07	The platform shall provide read access to the subscribers of raw sensory data
DCL-FR-08	The platform shall monitor the lifelog of a user for notify-able situations
DCL-FR-09	The platform shall persist user feedback regarding generated recommendations and identified context
DCL-FR-10	The platform shall manipulate the value of the factors from lifelog and questionnaires related to nutrition, physical activity and diseases.
DCL-FR-11	The platform shall use the questionnaire responses for the factors manipulation when the registration age is less than seven days.
DCL-FR-12	The platform shall use the lifelog for the factors manipulation when the registration age is greater than seven days.
DCL-FR-13	The platform shall identify the risk status associated with the manipulated factors.
DCL-FR-14	The platform shall provide the risk factor's vector for the recommendation generation and adaptive questionnaires.
DCL-FR-15	The platform shall use factor's value to manipulate the index for the healthy behavior identification.
DCL-FR-16	The platform shall provide the details of the healthy behavior index on the basis of the every factors' status on the basis of the week.

1.2 Non-Functional Requirements

FR ID#	Description
DCL-NFR-01	The platform shall read the raw sensory data of the user from his/her personal device in real-time with delay no later than 3 seconds
DCL-NFR-02	The platform shall provide raw sensory data for low level activities determination in real-time with delay no later than 3 seconds
DCL-NFR-03	The platform shall only read the raw sensory data from verified personal device

DCL-NFR-04	The platform shall maintain the consistency, integrity, and reliability of raw sensory data in non-volatile storage

1.3 Terms and Definitions

Term	Definition
DCL	Data Curation Layer
ICL	Information Curation Layer
KCL	Knowledge Curation Layer
SCL	Service Curation Layer
SL	Supporting Layer
Lifelog	Information associated to the user's life-events over time
Lifelog schema	Lifelog schema represents the structure and associated semantics of user profile and lifelog data.
User profile	Information describing the user characteristics (i.e., age, gender, etc.)
Data source	User devices sending the required data, i.e., smartphone, video camera
Raw sensory data	Numerical values describing a physical phenomenon such as human body motion (e.g., acceleration)
Sensory metadata	Information that describes, at least, the source of data (e.g., video), the user to which the raw sensory data belongs (e.g., user ID) and the time in which the raw sensory data was registered (e.g., timestamp)
Sensory data	Raw sensory data plus sensory metadata
SNS data	Data from social networks (i.e., twitter, Facebook)
Context	General concept to refer either to low-level context and/or high-level context
Situation	An abnormal status of a subject caused by unhealthy behaviors
Rule verification and validation	Verification ensures that rule created is consistent with requirements and validation ensures that the rule created is correctly working on real data
Unresolved case	A new case for which the existing knowledge is insufficient to solve
Recommendation	An actionable statement provided to the subject for healthy habit induction
Fact	An informative statement provided to the subject for education

1.4 Use-cases

1.4.1 List

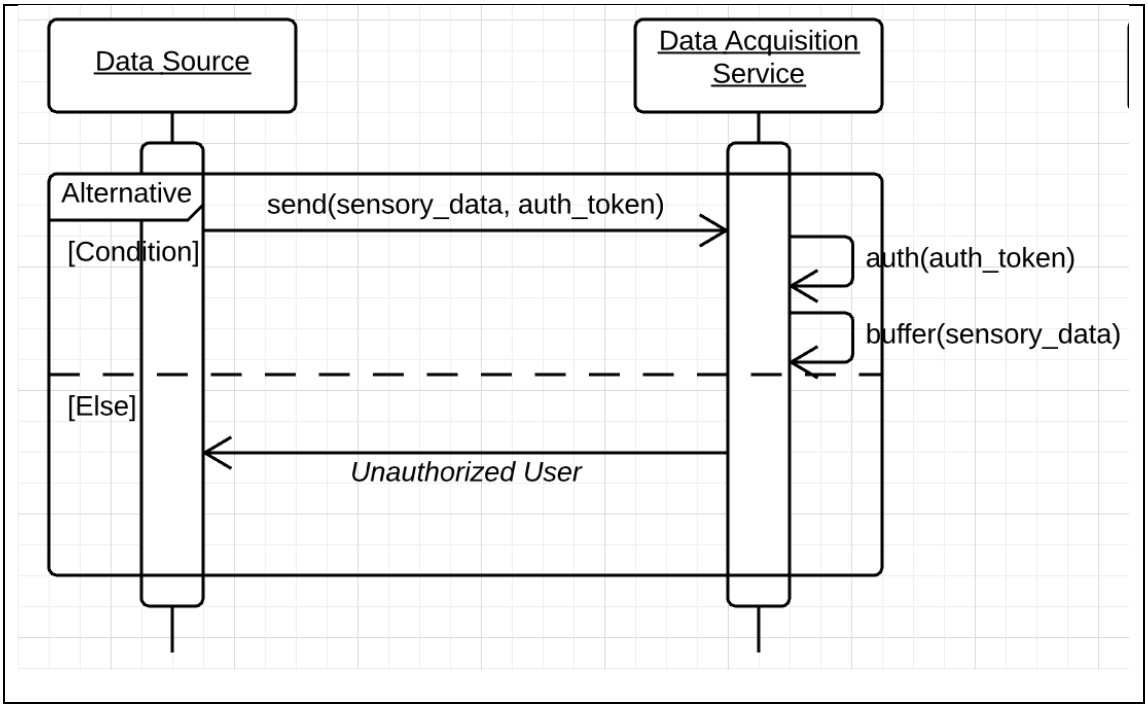
Use case ID#	Name
DCL-UC-01	Receive sensory and environmental data from data source
DCL-UC-02	Receive video data stream from data source
DCL-UC-03	Synchronize heterogeneous user data
DCL-UC-04	Send data for context determination
DCL-UC-05	Receive context data
DCL-UC-06	Retrieve Lifelog Information
DCL-UC-07	Persist Lifelog Information
DCL-UC-08	Map Instances
DCL-UC-09	Validate Instances
DCL-UC-10	Situation configuration
DCL-UC-11	LLM configuration for target variables
DCL-UC-12	LLM for situation detection
DCL-UC-13	Retrieve sensory data from non-volatile storage for intermediate data generation (offline)
DCL-UC-14	Retrieve sensory data from non-volatile storage (online)
DCL-UC-15	Persist sensory data in non-volatile storage
DCL-UC-16	Manipulate Behavior factors
DCL-UC-17	Identify Risky factors
DCL-UC-18	Identify Healthy Behavior Index (HBI)
DCL-UC-19	Generate detail of HBI week wise

1.4.2 Diagram


DCL
Synchronize heterogenous data
Persist sensory data in non-volatile storage
Receive video data stream
«include»
«include»
Data Source
«include»
Receive Sensory, Env. Data
«extend»
Send data for context determination
Validate Instance
«include»
Receive Context data
«extend»
Persist Life-log Information
«include»
Map Instance
ICL
SL
termediate data generation
LLM Configuration
Situation configuration
KCL
Retrieve sensory data from non-volatile storage
Generate weekwise Detailed HBI
LLM for situation detection
«include»
Manipulate Behavior Factors
«include»
Identify Healthy Behavior Index
«include»
Identify Risky Factors
SCL
«extend»
Retrieve Life-log Information

1.4.3 Description

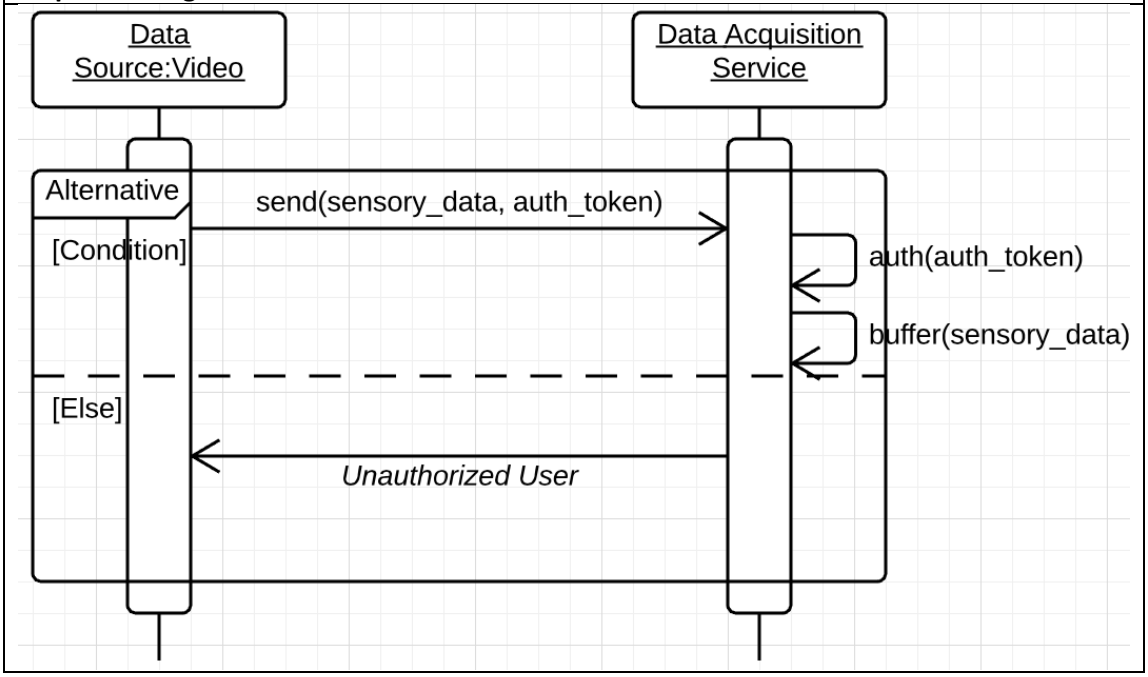
Use Case ID:	**DCL-UC-01**		
Use Case Name:	Receive sensory and environmental data from data source		
Created By:	Bilal Amin	Last Updated By:	Bilal Amin
Date Created:	15 July 2015	Last Revision Date:	20 Dec 2016
Actors:	Data source		
Description:	User sensory data and environmental is received and buffered from data source in real time		
Trigger:	User activity of at least 3 seconds		
Pre-conditions:	User is a registered client of MM platform		
Post-conditions:	Sensory and environmental data is persisted in the buffer		
Normal Flow:	1. Sensory and environmental data is received by a data acquisition component 2. Data source is authenticated, and contents of the data are verified 3. Data is temporary buffered for context determination		
Alternative Flows:	N/A		
Exceptions:	2a. In step 2 of the normal flow, if the user is detected to be un-authorized or contents are un-verifiable Data acquisition component destroys the data		
Includes:	N/A		
Frequency of Use:	Very frequent: every 3 second		
NFR ID:	DCL-NFR-01, DCL-NFR-03		
Assumptions:	Communication contract is defined between data source and data acquisition component		
Notes and Issues:	NA		
Sequence Diagram:			



Use Case ID:	**DCL-UC-02**		
Use Case Name:	Receive video data stream from data source		
Created By:	Bilal Amin	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	Data source		
Description:	User video data stream is received and buffered from data source in real time		
Trigger:	Video camera is streaming user feed		
Pre-conditions:	User is a registered client of MM platform		
Post-conditions:	User video data stream is persisted in the video stream buffer		
Normal Flow:	1. Video data stream is received by a data acquisition component 2. Data source is authenticated, and contents of the data stream are verified 3. Video data stream is temporary buffered for context determination		
Alternative Flows:	N/A		
Exceptions:	2a. In step 2 of the normal flow, if the user is detected to be un-authorized or contents are un-verifiable 1. Data acquisition component destroys the data		
Includes:	N/A		
Frequency of Use:	Less frequent: If video streaming-based data source is available		
NFR ID:	DCL-NFR-03		

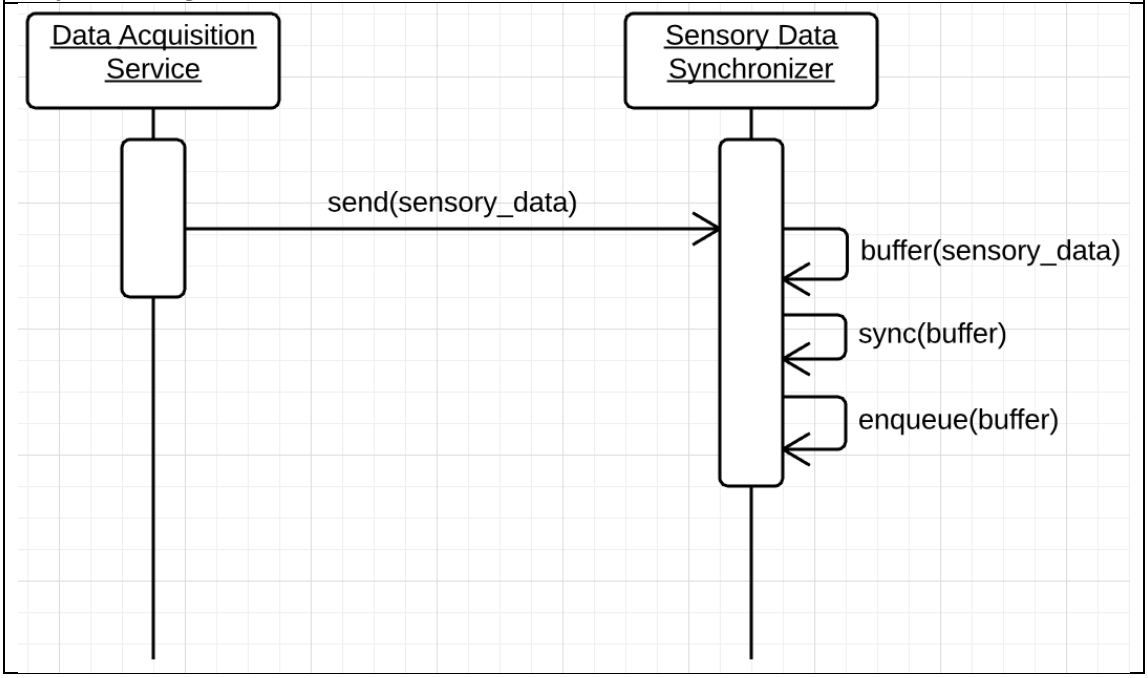
Assumptions:	Video data streaming communication contract is defined between data source and data acquisition component
Notes and Issues:	NA

Sequence Diagram:

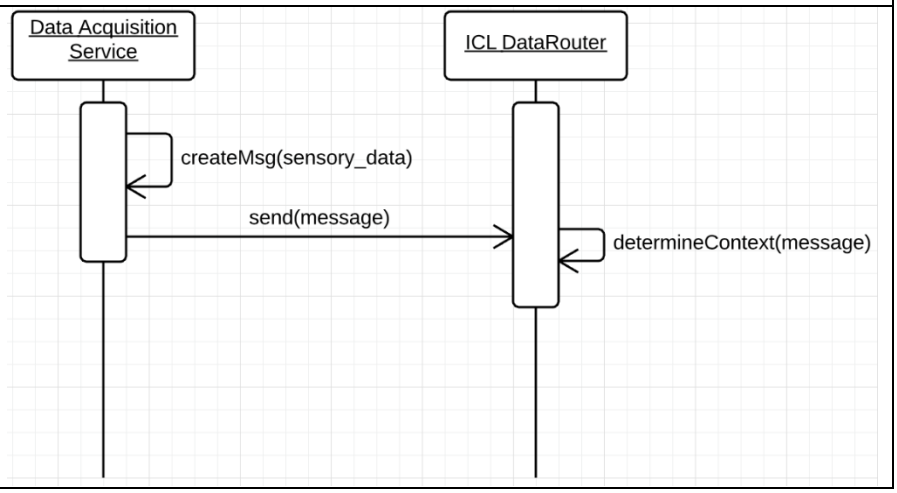


Use Case ID:	**DCL-UC-03**		
Use Case Name:	Synchronize heterogeneous user data		
Created By:	Bilal Amin	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	DCL		
Description:	User video data stream is synchronized with the corresponding sensory data		
Trigger:	Video camera is streaming user feed		
Pre-conditions:	Video data stream is persisted in the video stream buffer		
Post-conditions:	Video data stream is synchronized with its corresponding sensory data		
Normal Flow:	1. Time stamp and user id of video data stream is read from the video stream buffer 2. Sensory data is searched and retrieved from buffer based on the time stamp and user id 3. Retrieved sensory data is concatenated with the video stream data and stored back in the sensory data buffer for context determination		
Alternative Flows:	2a. In step 2 of the normal flow, if the sensory data is not found 1. Data acquisition component deletes the video data stream from the video data stream buffer		
Exceptions:	NA		
Includes:	NA		

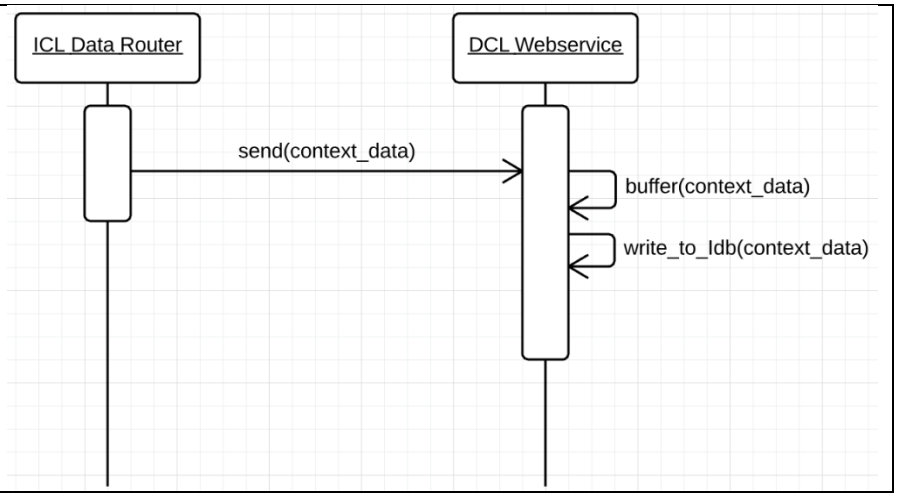
Frequency of Use:	Less frequent: If video streaming-based data source is available
NFR ID:	DCL-NFR-02
Assumptions:	4. Video data streaming communication contract is defined between data source and data acquisition component
Notes and Issues:	NA
Sequence Diagram:	



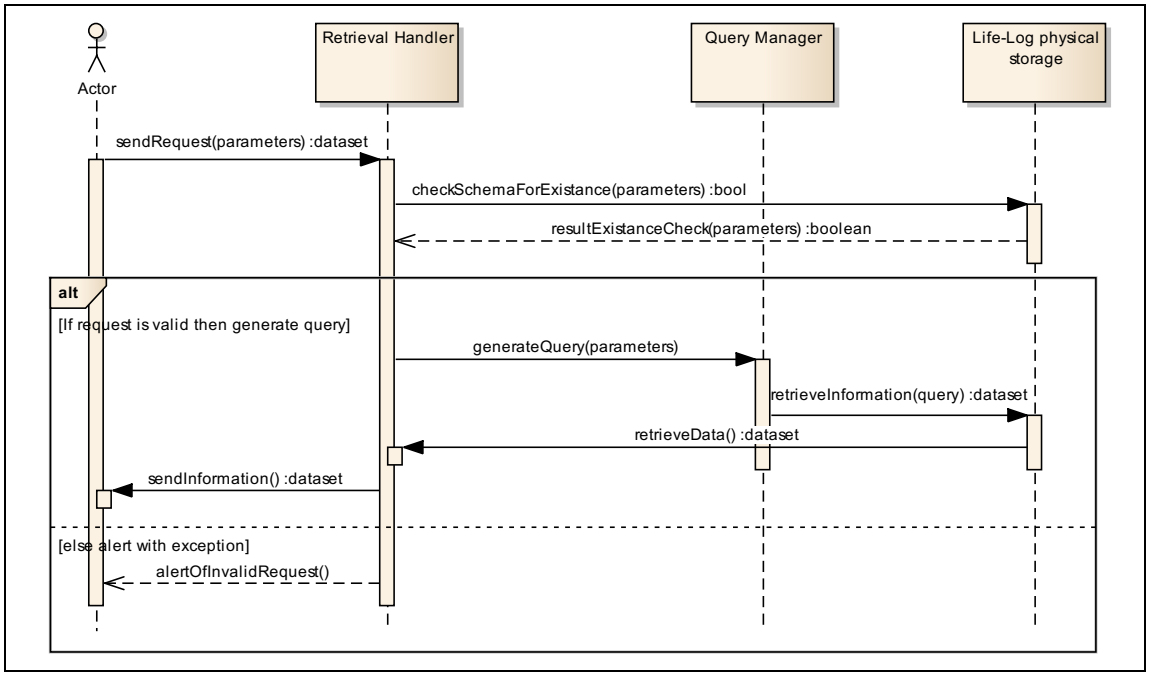
Use Case ID:	**DCL-UC-04**		
Use Case Name:	Send data for context determination		
Created By:	Bilal Amin	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	Context sender		
Description:	Sensory data buffer is sent to ICL for context determination		
Trigger:	Sensory data is available for context determination		
Pre-conditions:	Sensory data is persisted in the buffer		
Post-conditions:	Sensory data is sent for context determination		
Normal Flow:	1. Context sender reads sensory data from the sensory data buffer 2. Context sender creates communication object by serialization 3. Communication object is sent to the ICL server		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		

Frequency of Use:	Frequent: whenever context need to be determined
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	

Use Case ID:	**DCL-UC-05**		
Use Case Name:	Receive context data		
Created By:	Bilal Amin	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	ICL server		
Description:	After the determination by ICL, context is received by context receiver component and forwarded for non-volatile storage		
Trigger:	New context or change in previous context is determined by ICL		
Pre-conditions:	Context data is available		
Post-conditions:	Context data is sent for non-volatile storage		
Normal Flow:	1. Context receiver receives context object 2. Context receiver de-serializes context object 3. Context object is sent for non-volatile persistence (async) 4. Context object is sent for lifelog mapping		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Frequent: whenever context is determined		
NFR ID:	DCL-NFR-02		
Assumptions:	NA		
Notes and Issues:	NA		

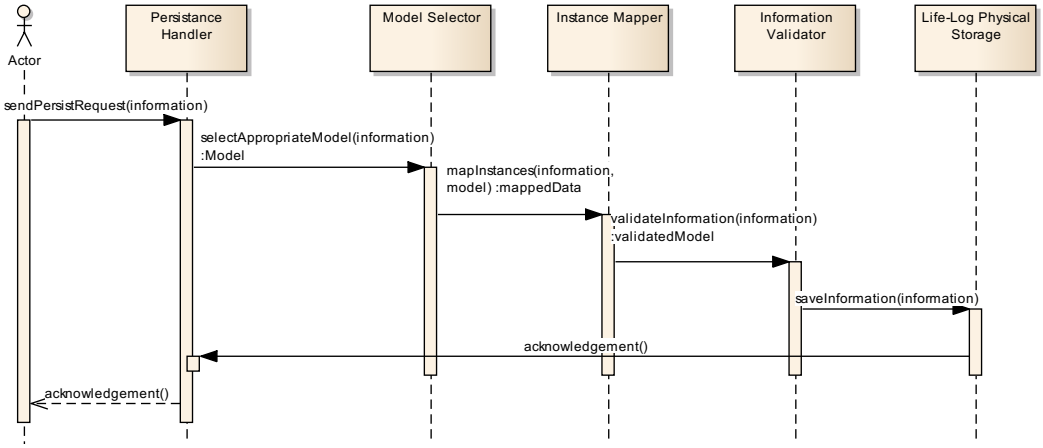
Sequence Diagram:	

Use Case ID:	**DCL-UC-06**		
Use Case Name:	Retrieve Lifelog Information		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	SCL, KCL, SL		
Description:	Each actor needs information from life log for further processing. All actors shall request their related and desired Lifelog information from physical storage.		
Trigger:	On request of an actor to access required information		
Pre-conditions:	The actor shall be authorized with full access on the Lifelog data.		
Post-conditions:	Provide the required data to layer		
Normal Flow:	1. Actor sends request for desired Lifelog information. 2. The desired request shall be checked for information existence. If request is valid a. Prepare the query for desired information based on request b. Load the requested information from physical storage c. Send back the loaded information to the actor		
Alternative Flows:	2b. The desired data is not existing in the schema, invalid request 1. Acknowledge the actor with exception of invalid request.		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Whenever Lifelog information is required.		
NFR ID:	NA		
Assumptions:	NA		
Notes and Issues:	NA		
Sequence Diagram:			



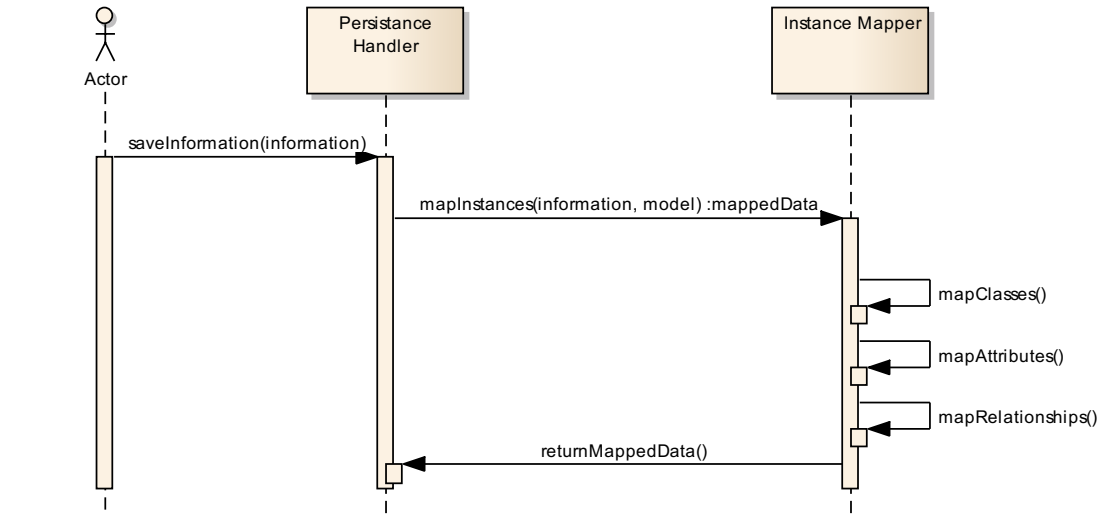
Use Case ID:	**DCL-UC-07**		
Use Case Name:	Persist Lifelog Information		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	ICL		
Description:	Each actor performed some specific operations on incoming data from external resources or on already existing information in Lifelog repository. In both cases the information shall be updated and stored in Lifelog repository.		
Trigger:	On request of a actor to persist required information		
Pre-conditions:	The actor shall be authorized with full access on the Lifelog data.		
Post-conditions:	Successfully stored the created Lifelog information		
Normal Flow:	1. Actor sends request to persist new generated Lifelog information. 2. Passes the new created information to check the appropriate hierarchical structure. 3. The appropriate selected hierarchical structure with input information passes to find the information instances. 4. Check the consistency among the records and their relationship. 5. Store the validated and structured information into physical storage		
Alternative Flows:	NA		

Exceptions:	NA
Includes:	Map Instances, Validate Instances, save information to physical storage.
Frequency of Use:	Whenever new information is generated.
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	Capability to process multiple actor's requests for storage.
Sequence Diagram:	



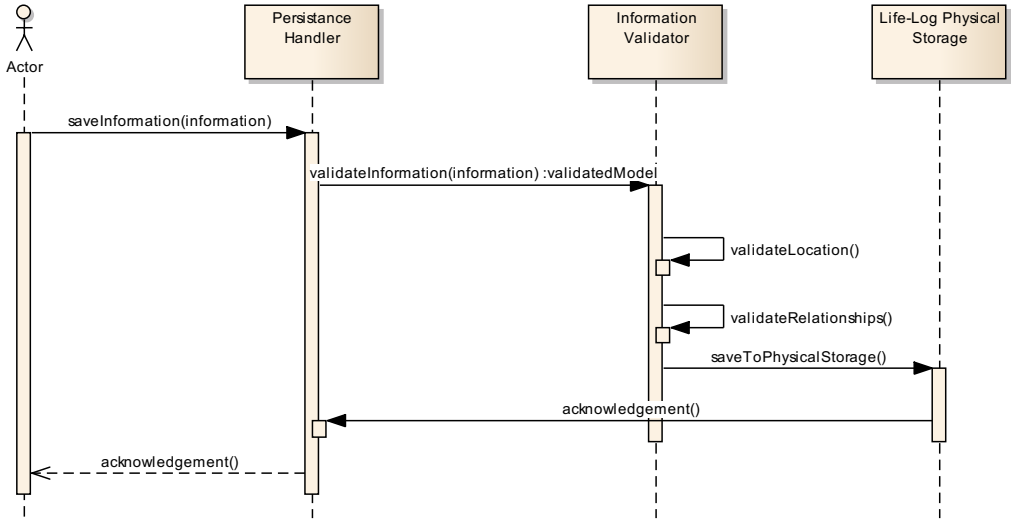
Use Case ID:	**DCL-UC-08**		
Use Case Name:	Map Instances		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	DCL, ICL, SCL, SL, and Lifelog physical storage		
Description:	The information produced by each actor shall be mapped against the hierarchical structure of storage.		
Trigger:	On request of an actor to persist required information		
Pre-conditions:	The actor shall be authorized with full access on the Lifelog data.		
Post-conditions:	Successfully mapped the instances with correct Lifelog information schema		
Normal Flow:	1. Actor sends request to persist new generated information. 2. System searches each information records against hierarchical structure. 3. System finds appropriate classes of the instances 4. System extracts attributes in the instances. 5. Find the relationship among the information records. 6. Pass the annotated information for validation.		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		

Frequency of Use:	Whenever new information is persisted.
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	Capability to process multiple actors' requests for storage.
Sequence Diagram:	



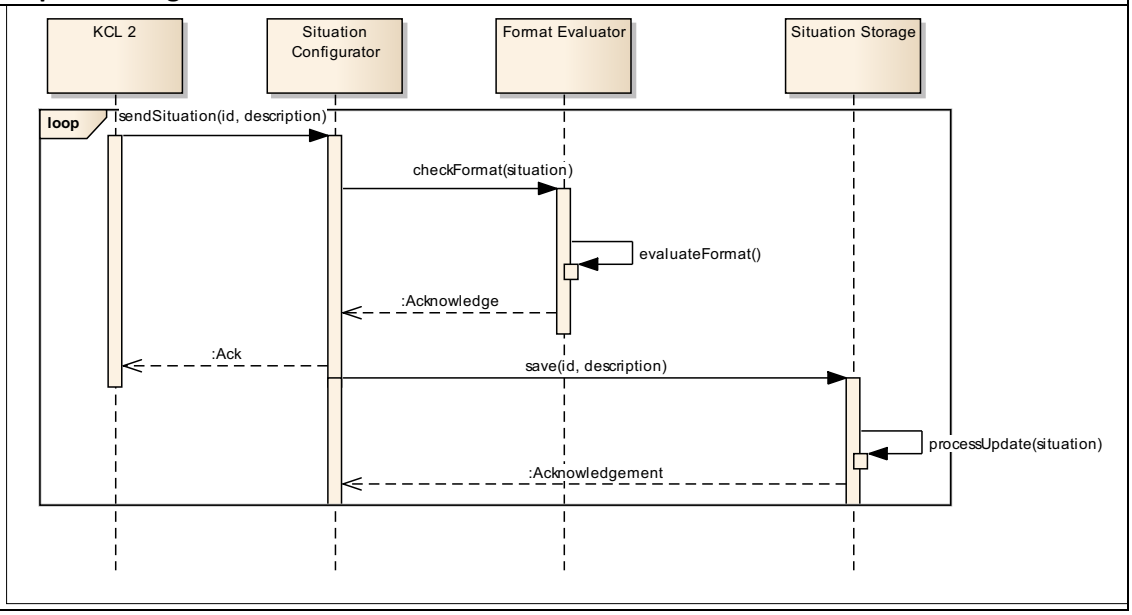
Use Case ID:	**DCL-UC-09**		
Use Case Name:	Validate Instances		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	DCL, ICL, SCL, SL, and Lifelog physical storage		
Description:	The mapped information in previous use case shall be checked for consistency among the existing information.		
Trigger:	On request of an actor to persist required information		
Pre-conditions:	The actor shall be authorized with full access on the Lifelog data.		
Post-conditions:	Successfully validate the instances with correct Lifelog information schema		
Normal Flow:	1. Actor send request to persist new generated information. 2. DCL passes the mapped information for validation of information and their relationships. 3. The system checks the information according to the specific location in the hierarchy. 4. The system checks and builds the relationship among concepts. 5. The verified information shall be passed for persistence.		
Alternative Flows:	NA		
Exceptions:	NA		

Includes:	NA
Frequency of Use:	Whenever new information is persisted.
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	Capability to process multiple actors' requests for storage.
Sequence Diagram:	



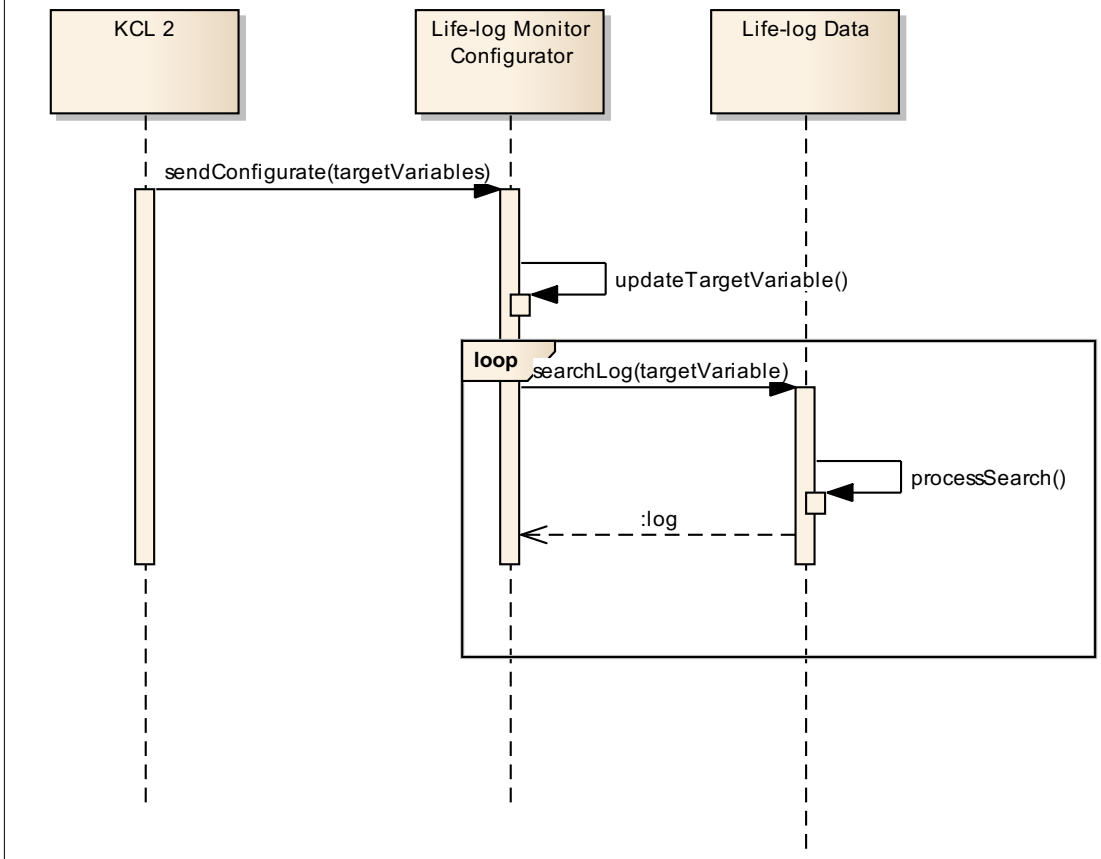
Use Case ID:	**DCL-UC-10**		
Use Case Name:	Situation Configuration		
Created By:	Bilal Ali	**Last Updated By:**	Bilal Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	KCL		
Description:	Situation is determined by experts and is communicated to DCL for monitoring the Lifelog.		
Trigger:	Creation of new rule to capture a situation.		
Pre-conditions:	KCL and DCL should agree on common representation of sharing information of Situation configuration		
Post-conditions:	1. Situation is stored against a specific category. 2. Situation is available for monitoring the Lifelog.		
Normal Flow:	1. KCL connects to DCL and send the newly created situation in common configuration format. 2. DCL evaluates the format of received situation configuration. 3. DCL responds with acknowledgement message. 4. Situation will be parsed into components. 5. Parsed components are updated in persistent storage as per categories.		
Alternative Flows:	NA		
Exceptions:	Format of situation is not according the agreement.		

Includes:	NA
Frequency of Use:	Invoked per situation creation by the expert.
NFR ID:	NA
Assumptions:	Well defined schema is available to store situation persistently
Notes and Issues:	Standardize situation format is a challenging task
Sequence Diagram:	



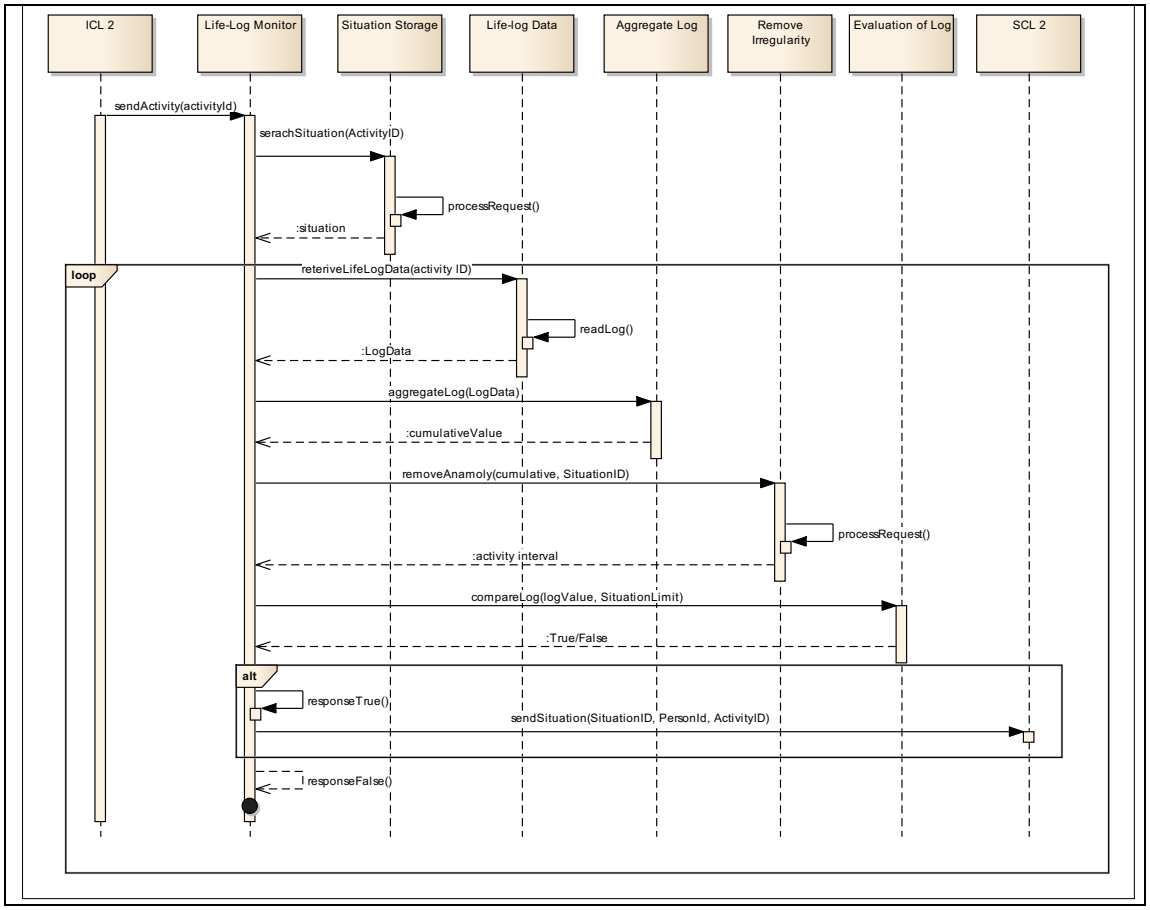
Use Case ID:	**DCL-UC-11**		
Use Case Name:	LLM Configuration for target Variables		
Created By:	Bilal Ali	**Last Updated By:**	Bilal Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	KCL, Experts		
Description:	Configure the Lifelog monitor for the screening of the target variable from Lifelog data.		
Trigger:	On start of user's monitored activity		
Pre-conditions:	1. Expert defines target variable in common configuration format. 2. Access to Lifelog.		
Post-conditions:	Targeted log is retrieved from Lifelog data as per target variable requirements.		
Normal Flow:	1. KCL will share the target variables in common configured format created by expert. 2. Lifelog monitor is configured based on the shared target variable. 3. Lifelog monitor retrieve log data from Lifelog against the target variables.		
Alternative Flows:	NA		

Exceptions:	NA
Includes:	NA
Frequency of Use:	On update of common configuration format.
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	Real time accommodation of update in common configuration format is challenging task.
Sequence Diagram:	



Use Case ID:	**DCL-UC-12**		
Use Case Name:	LLM for Situation Detection		
Created By:	Bilal Ali	**Last Updated By:**	Bilal Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	Lifelog Data, SCL, ICL		
Description:	Identification of the existence of a condition in user activities to highlight the alarming situation as per experts' understanding.		
Trigger:	On start of user's monitored activity		
Pre-conditions:	1. Activity is identified. 2. Situation is configured. 3. Access to Lifelog.		

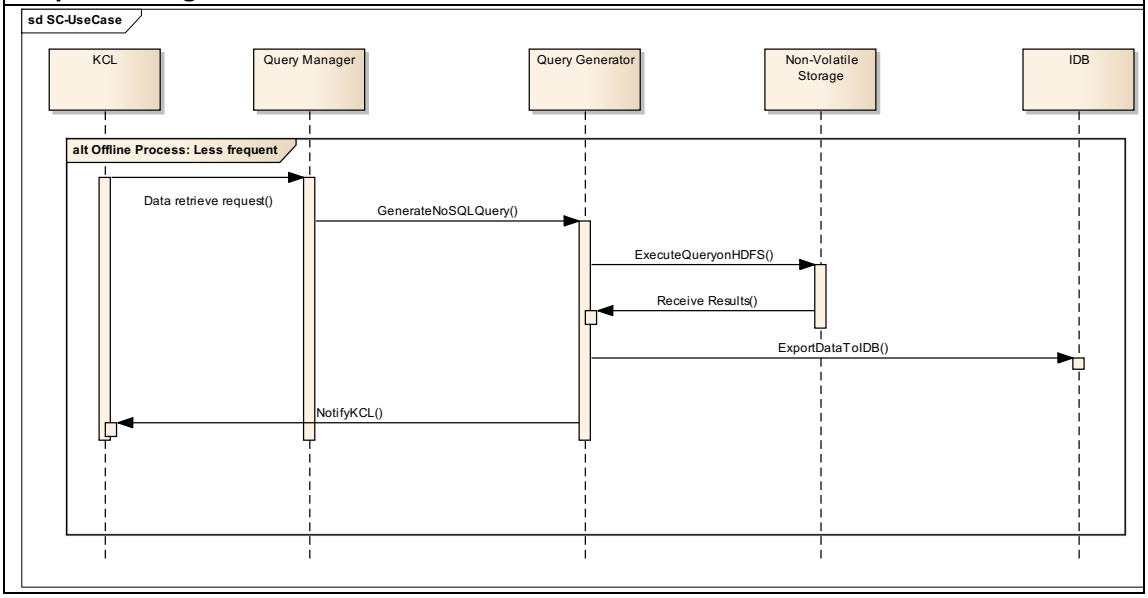
Post-conditions:	Alarming situation is detected and triggered the SCL with situation and user.
Normal Flow:	1. ICL recognizes activity and sends to Lifelog. 2. Lifelog monitor identify the target activity. 3. Retrieve associated situation with the activity. 4. Continuous access that activity log. 5. Aggregate the interval/duration of activity. 6. Remove the irregularity in activity as per situation. 7. Evaluate the duration of activity against the situation. a. If situation condition meets then send message to SCL to inform about the occurrence of a situation along with user information. b. If situation condition does not occur, don't send message to SCL.
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	For every activity with configured situation.
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	Management of irregularity in activity is a challenging task.
Sequence Diagram:	



Use Case ID:	**DCL-UC-13**		
Use Case Name:	Retrieve sensory data from non-volatile storage for intermediate data generation (offline)		
Created By:	Usman	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	KCL		
Description:	Raw data in HDFS is retrieved based on the request from KCL and converted to relational format and stored in IDB.		
Trigger:	KCL requests the data		
Pre-conditions:	1. Relational IDB schema has already been described and shared 2. The data exists in HDFS		
Post-conditions:	1. The data has been transformed and exported to IDB 2. The KCL is informed		
Normal Flow:	1. SCL receives requests from KCL and creates a NoSQL query 2. NoSQL query is executed using Apache Hive on HDFS to retrieve the data 3. Retrieved data is processed and transformed to relational format based on IDB schema 4. Transformed data is exported to IDB		

Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less frequent, offline process, may be executed once or on change in the IDB schema
NFR ID:	NA
Assumptions:	NoSQL query results are easy to transform to relational format
Notes and Issues:	NA

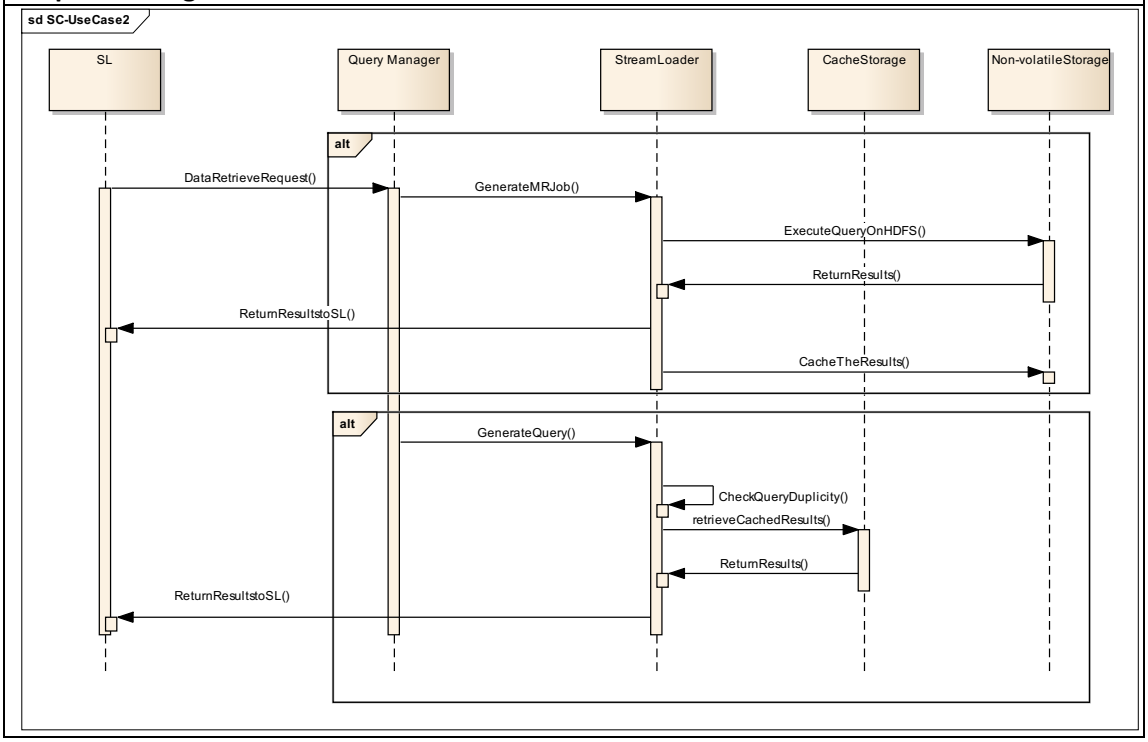
Sequence Diagram:



Use Case ID:	**DCL-UC-14**		
Use Case Name:	Retrieve sensory data from non-volatile storage (online)		
Created By:	Usman	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	SL		
Description:	Raw sensory data in HDFS is retrieved and provided to analytics component of SL in an online process		
Trigger:	Request for data from descriptive analytics is received		
Pre-conditions:	Raw sensory exists and persisted in HDFS (non-volatile storage)		
Post-conditions:	1. Required data is retrieved from HDFS 2. Data is cached locally in SCL 3. Required data is communicated to descriptive analytics in SL directly (online)		
Normal Flow:	1. Descriptive analytics in SL requests data from SCL 2. SCL receives the request and maintains its log. 3. SCL transforms request to a MapReduce Job. 4. MapReduce job is executed on HDFS and results are retrieved.		

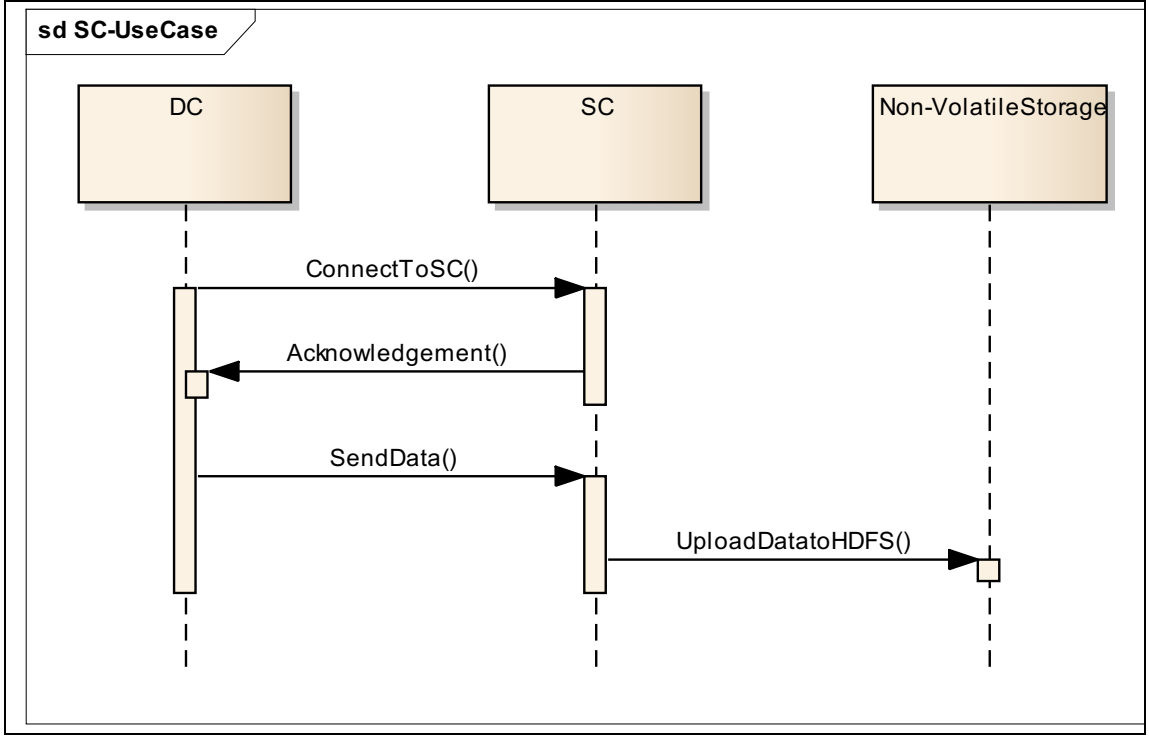
	5. Retrieved results are cached locally in volatile storage 6. Results are forwarded directly to descriptive analytics
Alternative Flows:	1. Descriptive analytics requests data from SCL 2. SCL receives the request and examines it for duplicity(a request coming more than one time) 3. Duplicate request results are directly forwarded from the cached results to descriptive analytics 4. First time requests follow the normal flow.
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent requests from descriptive analytics in SL
NFR ID:	NA
Assumptions:	Request format contract already defined between SL(descriptive analytics) and DCL (SCL)
Notes and Issues:	NA

Sequence Diagram:

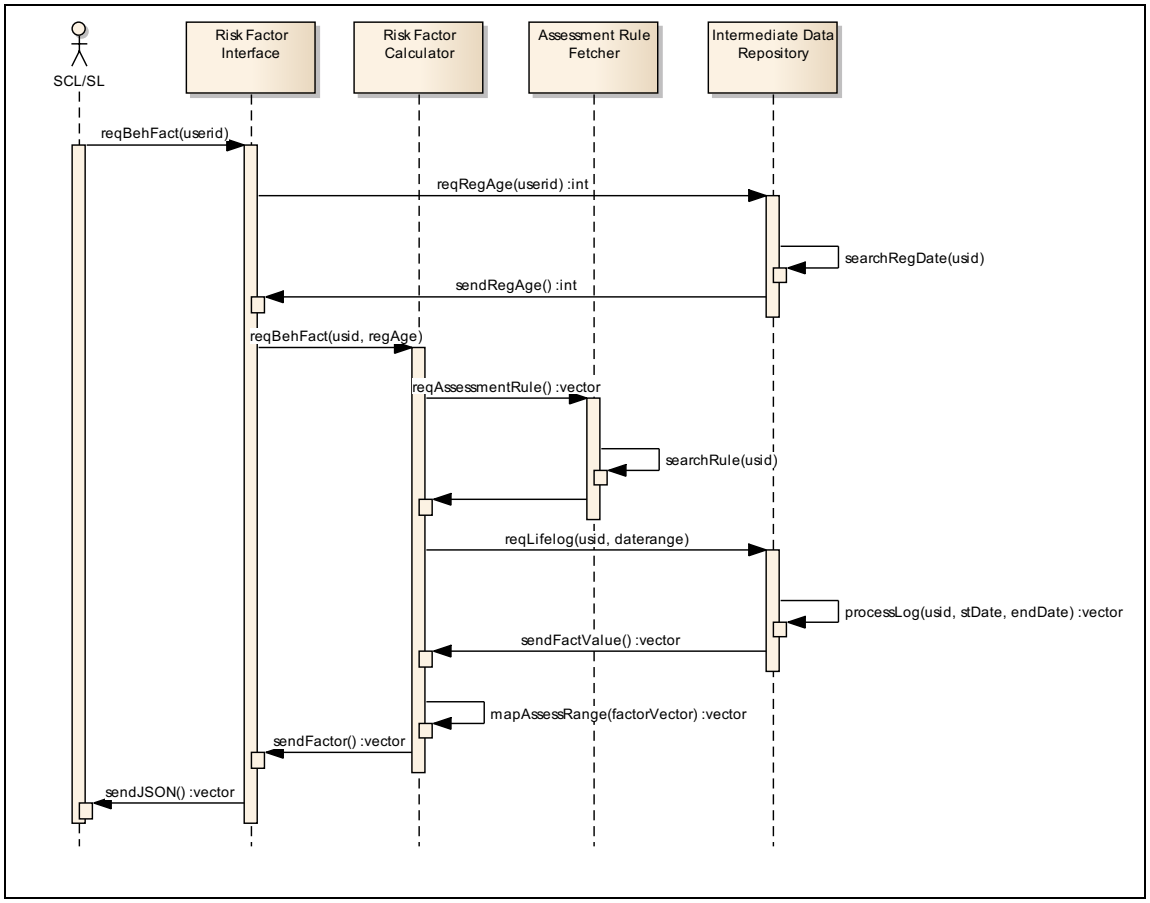


Use Case ID:	**DCL-UC-15**		
Use Case Name:	Persist sensory data in non-volatile storage		
FR ID:			
Created By:	Usman	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	DCL		
Description:	Receive and persist raw sensory data from DCL in to HDFS		

Trigger:	DCL request to upload data, every 3 seconds
Pre-conditions:	1. Data storage structure, directory structure in HDFS defined 2. File formats and data formats in HDFS known 3. Raw sensory data is received from DCL 4. Big Data server is already running
Post-conditions:	1. Raw sensory data is persisted in HDFS non-volatile storage 2. Data is available for processing and access by SL and KCL
Normal Flow:	1. Big Data server listening for data requests from DCL 2. DCL connects and sends data to SCL 3. SCL uploads received data to HDFS
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Very frequent, every 3 seconds
NFR ID:	NA
Assumptions:	Data format and specifications already defined between DCL and SCL
Notes and Issues:	NA
Sequence Diagram:	



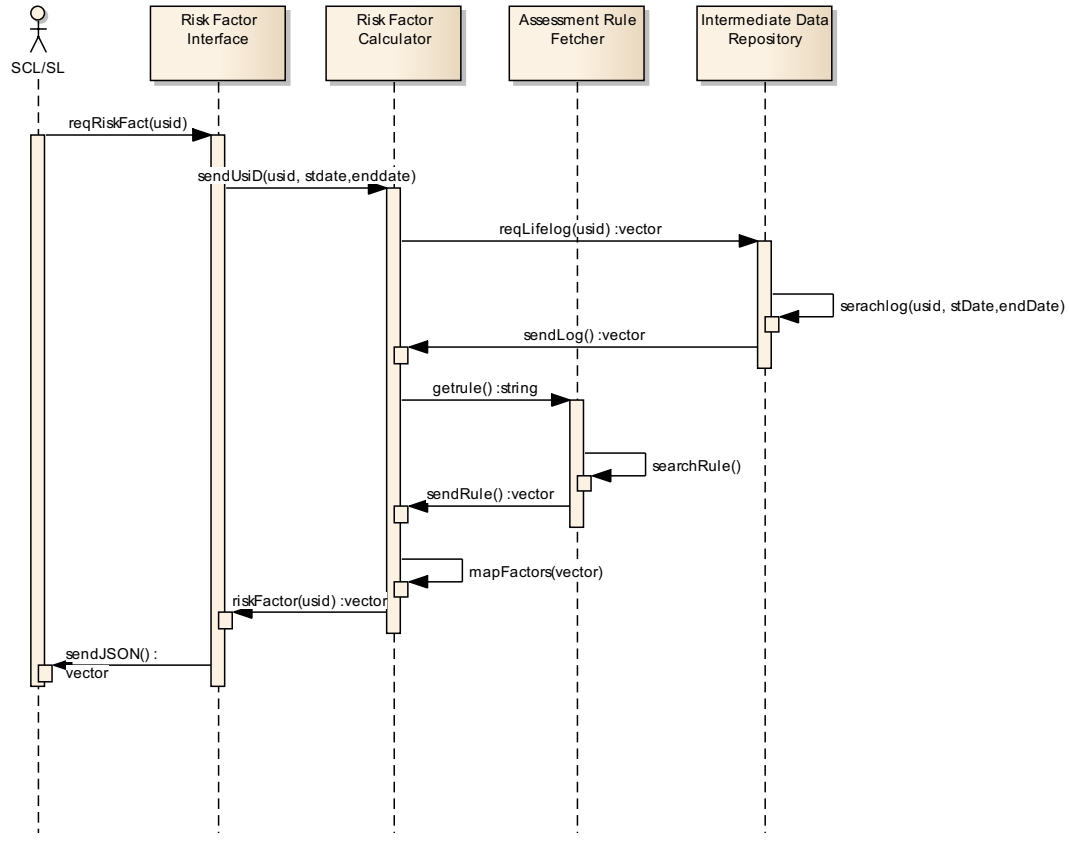
Use Case ID:	**DCL-UC-16**		
Use Case Name:	Manipulate Behavior Factor		
Created By:	Bilal Ali	**Last Updated By:**	Bilal Amin
Date Created:	2 July 2018	**Last Revision Date:**	2 July 2018
Actors:	SCL		
Description:	User lifelog and questionnaires responses are used to manipulate the factors which are related to dietary behavior, physical activities and diseases.		
Trigger:	SCL triggers it when it need to give recommendations and generate adaptive questionnaires.		
Pre-conditions:	User has filled the questionnaire and life-log is maintained		
Post-conditions:	All factors are manipulated and healthy behavior index		
Normal Flow:	1. Receive request from SCL. 2. Calculate the registration age of the user. 3. For qualitative factors assess the latest responses of the questionnaires. 4. For qualitative factors accumulate the lifelog values week wise. 5. Map each factor with assigned range defined by expert. 6. Build factor vector and accumulate the factors with assigned proportion.		
Alternative Flows:	N/A		
Exceptions:	2a. In step 2 of the normal flow, if the user age is zero or data is unavailable the system not recognized exception must occur.		
Includes:	N/A		
Frequency of Use:	Depending on report generation for each user.		
NFR ID:	NA		
Assumptions:	User provides the responses of all the allocated questions and lifelog is maintained over time.		
Notes and Issues:	NA		
Sequence Diagram:			



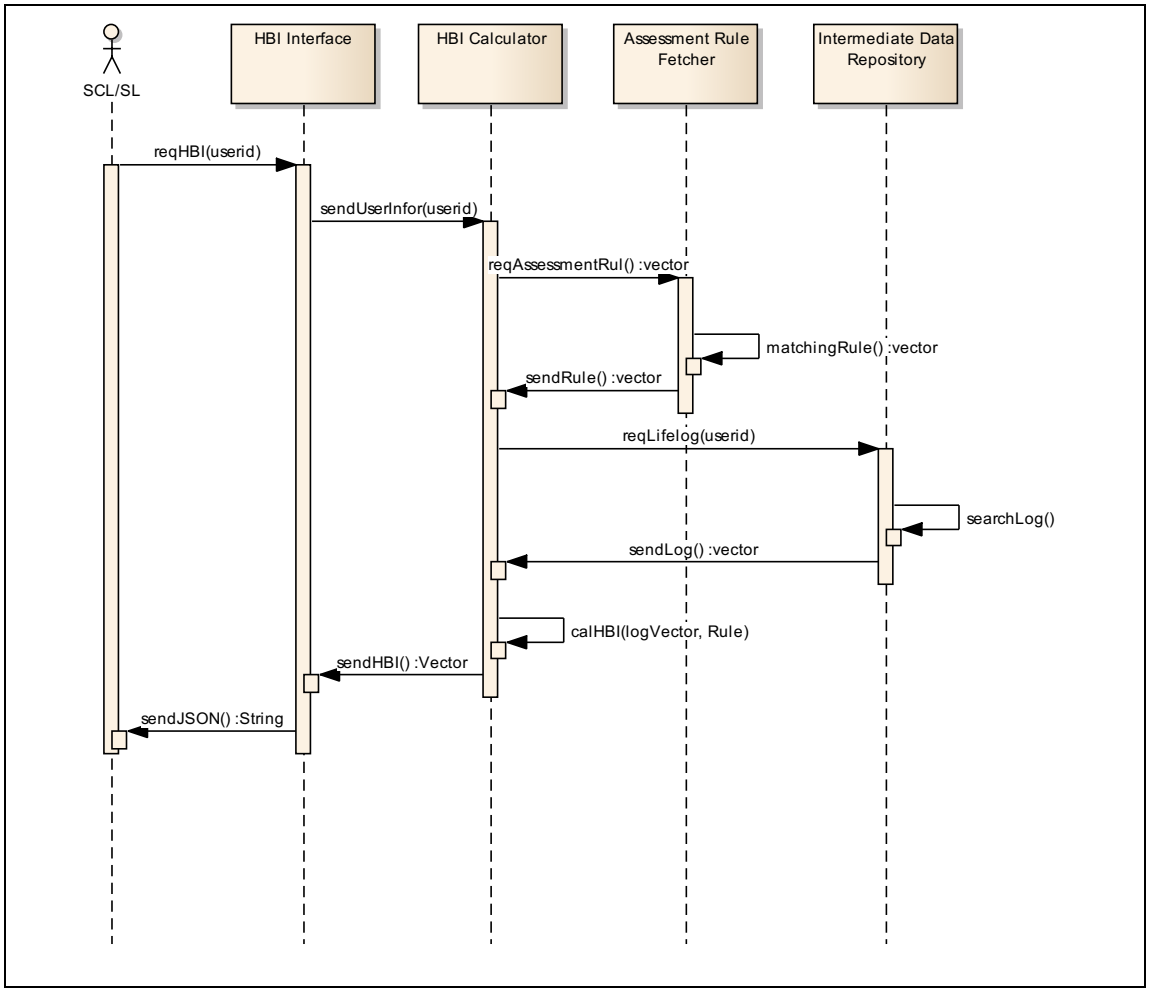
Use Case ID:	**DCL-UC-17**		
Use Case Name:	Identify Risk Factors		
Created By:	Bilal Ali	**Last Updated By:**	Bilal Amin
Date Created:	2 July 2018	**Last Revision Date:**	2 July 2018
Actors:	SCL		
Description:	User lifelog and questionnaires responses are used to manipulate the factors which are linked to find out the risky values on the basis of the rules defined by expert		
Trigger:	SCL triggers it when it need to give recommendations for dinning time and generate monthly questionnaires.		
Pre-conditions:	User has filled the questionnaire and life-log is maintained		
Post-conditions:	Personalized Risky factors vector is available to generate recommendations		
Normal Flow:	1. Receive request from SCL. 2. Calculate the registration age of the user. 3. Identify the registration status of the user. 4. For qualitative factors assess the latest responses of the questionnaires. 5. Map the responses into ranges for identification of the categories for risky factors.		

	6. For qualitative factors accumulate the lifelog values on the basis of the portion of the day passes. 7. Map each factor with assigned range defined by expert to identify the risky factor values. 8. Build risk factors vector and convert it into JSON to communicate with SCL in common communication format.
Alternative Flows:	N/A
Exceptions:	2a. In step 2 of the normal flow, if the user age is zero or data is unavailable the system should generate user not recognized exception.
Includes:	N/A
Frequency of Use:	At-least 3 times a day depending on report generation for each user.
NFR ID:	NA
Assumptions:	User provides the responses of all the allocated questions and lifelog is maintained over time.
Notes and Issues:	NA

Sequence Diagram:

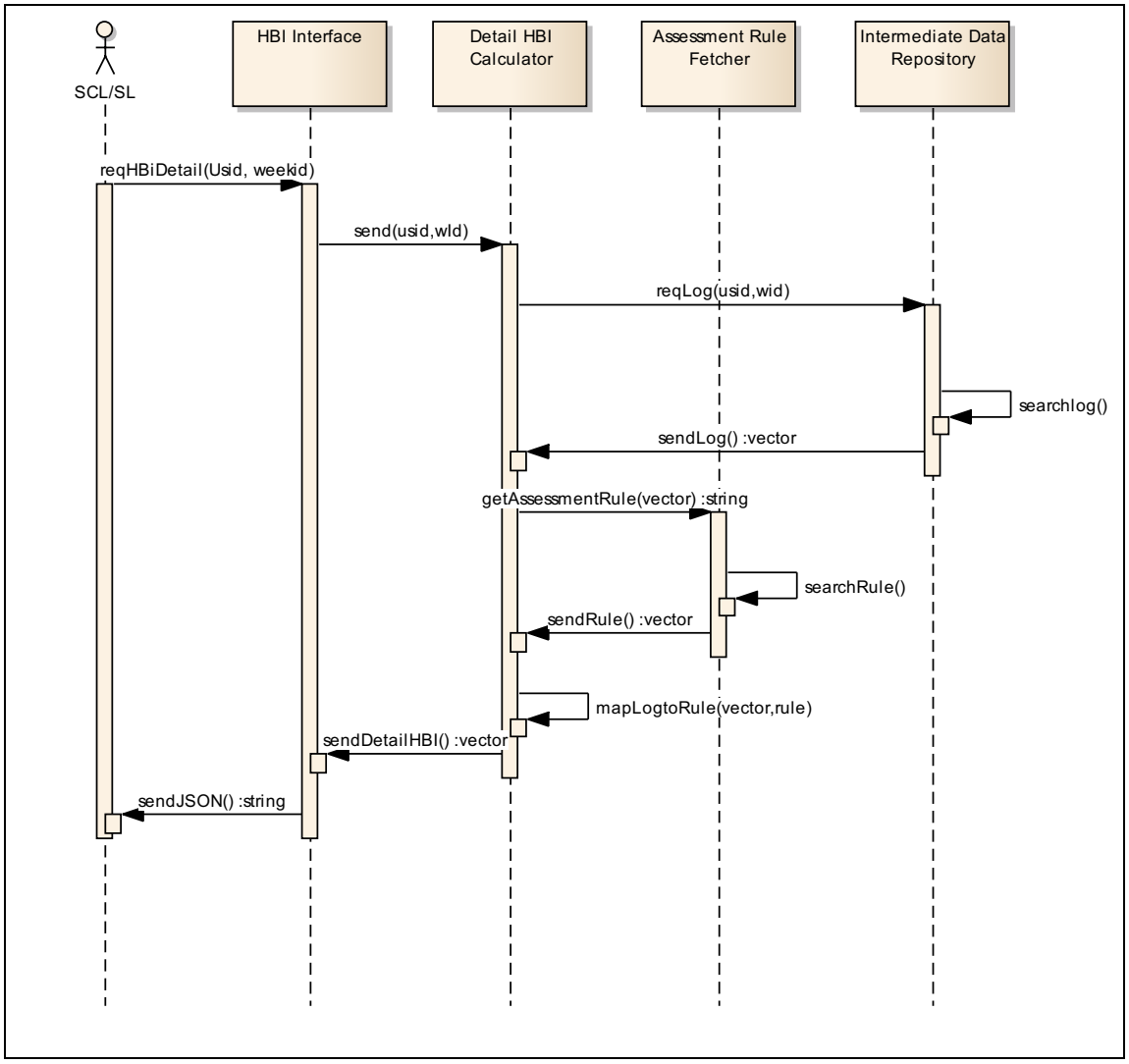


Use Case ID:	DCL-UC-18		
Use Case Name:	Identify Healthy Behavior Index (HBI)		
Created By:	Bilal Ali	Last Updated By:	Bilal Amin
Date Created:	2 July 2018	Last Revision Date:	2 July 2018
Actors:	SCL, SL		
Description:	User lifelog and questionnaires responses are used to recognize the behavior status of the user in-term of the healthy dieting, physical activity, smoking and alcohol consumption.		
Trigger:	SCL, SL can trigger it when they need to generate the progress report for the evaluation of user behavior.		
Pre-conditions:	User has filled the questionnaire and life-log is maintained		
Post-conditions:	Healthy behavior index week wise is available to represent the behavior status.		
Normal Flow:	1. Receive request from SCL. 2. Identify the total number of weeks from lifelog. 3. For qualitative factors assess the latest responses of the questionnaires week-wise. 4. Quantify the qualitative factor and map to a single value from the ranges defined by the expert. 5. Accumulate the qualitative factors from the lifelog values week wise and map it to single value from the ranges defined by the expert. 6. Accumulate the values of the contributing factors on the basis of the weightage and convert it into a single value. 7. Build helathy behavior index vector and convert it into JSON to communicate with SL and SCL in common communication format.		
Alternative Flows:	N/A		
Exceptions:	2a. In step 2 of the normal flow, if the user age is zero or data is unavailable the system should generate user not recognized exception.		
Includes:	N/A		
Frequency of Use:	Any time when user and expert generate request to evaluate the change in behavior.		
NFR ID:	NA		
Assumptions:	User provides the responses of all the allocated questions and lifelog is maintained over time.		
Notes and Issues:	NA		
Sequence Diagram:			



Use Case ID:	**DCL-UC-19**		
Use Case Name:	Generate detail of HBI week-wise		
Created By:	Bilal Ali	**Last Updated By:**	Bilal Amin
Date Created:	2 July 2018	**Last Revision Date:**	2 July 2018
Actors:	SCL, SL		
Description:	User lifelog and questionnaires responses are used to recognize the behavior status of the user in-term of the dietary factors, physical activity, smoking and alcohol consumption in a specified week.		
Trigger:	SCL, SL can trigger it when they need to generate the detail progress report for factors related to behavior which can affect the health of a person.		
Pre-conditions:	User has filled the questionnaire and life-log is maintained		
Post-conditions:	Contributing factors status in a week is available to represent the status.		
Normal Flow:	1. Receive request from SCL and SL. 2. Receive the week number. 3. For qualitative factors assess the latest responses of the questionnaires in the week number.		

	4. Quantify the qualitative factor and map to a single value from the ranges defined by the expert in that week. 5. Accumulate the qualitative factors from the lifelog values in that week and map it to single value from the ranges defined by the expert. 6. Build healthy behavior factor vector and convert it into JSON to communicate with SL and SCL in common communication format.
Alternative Flows:	N/A
Exceptions:	2a. In step 2 of the normal flow, if the required week has no information in lifelog then the system should generate week not registered exception.
Includes:	N/A
Frequency of Use:	Any time when user and expert generate request to visualize the factors value to identify the change in behavior.
NFR ID:	NA
Assumptions:	User provides the responses of all the allocated questions and lifelog is maintained over time.
Notes and Issues:	NA
Sequence Diagram:	


SCL/SL
HBI Interface
Detail HBI Calculator
Assessment Rule Fetcher
Intermediate Data Repository
reqHBiDetail(Usid, weekid)
send(usid,wld)
reqLog(usid,wid)
searchlog()
sendLog() :vector
getAssessmentRule(vector) :string
searchRule()
sendRule() :vector
mapLogtoRule(vector,rule)
sendDetailHBI() :vector
sendJSON() :string

2. Information Curation Layer

2.1 Function Requirements

Requirements #ID	Description
ICL-FR-01	The platform shall provide each low-level context recognizer with the appropriate raw sensory data for recognition
ICL-FR-02	The platform shall identify the user's low-level context
ICL-FR-03	The platform shall identify the user's high-level context
ICL-FR-04	The platform shall provide low-level context information for the generation of the life-log
ICL-FR-05	The platform shall provide high-level context information for the generation of the life-log

2.2 Non-Function Requirements

Requirements #ID	Description
ICL-NFR-01	Overall low-level context recognition accuracy of the platform shall be be greater than or equal to 85%
ICL-NFR-02	Overall high-level context inference accuracy of the platform shall be greater than or equal to 80%

2.3 Terms and Definitions

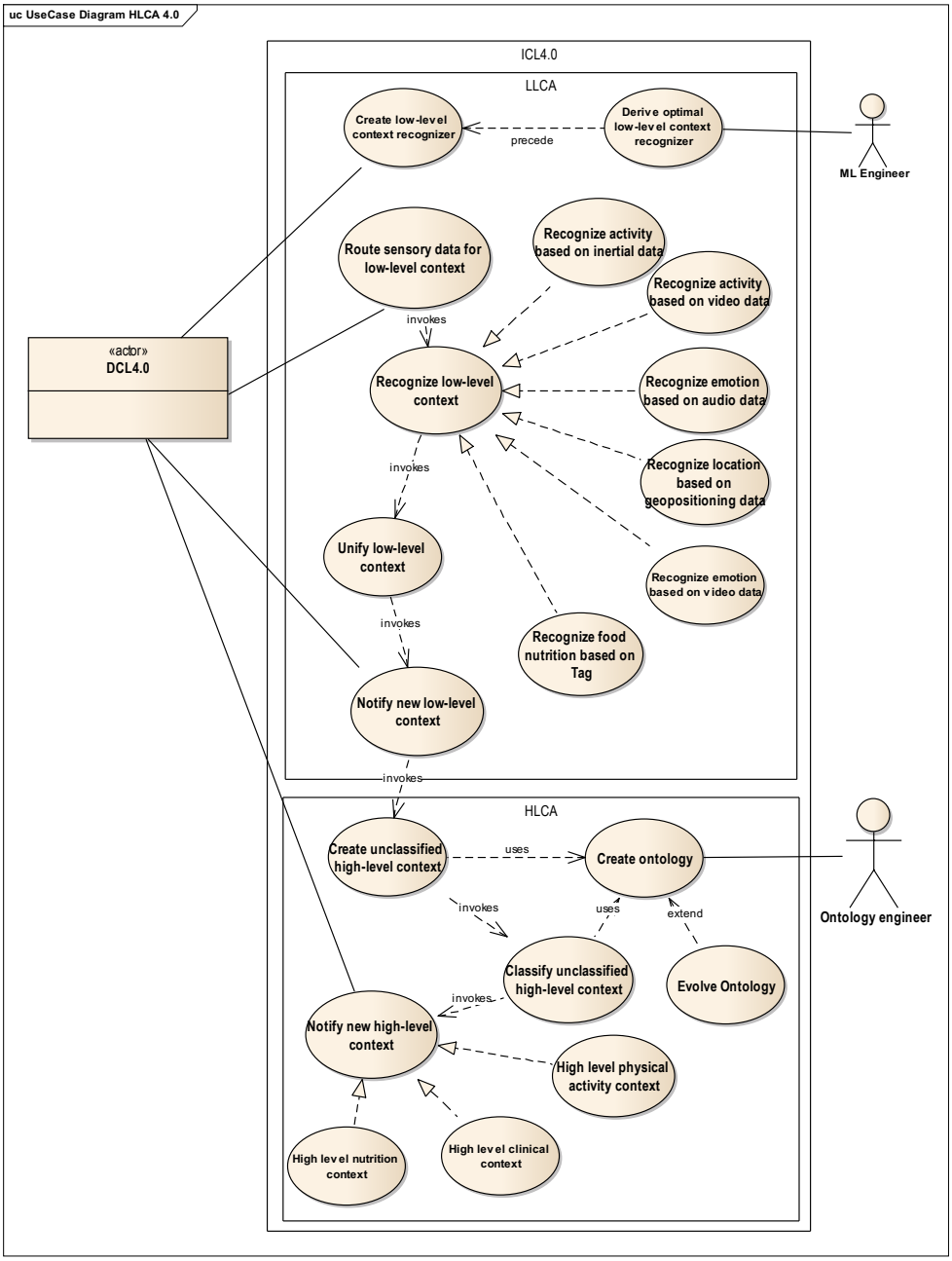
Terms	Description
DCL	Data Curation Layer
KCL	Knowledge Curation Layer
Reasoning	The process of producing recommendations
Low-level context	Information describing the user activities (e.g., sitting), user locations (e.g., restaurant) and user emotions (e.g., happy)
High-level context	Information describing the situation of the user (e.g., lunch)
Context	General concept to refer either to low-level context and/or high-level context

2.4 Use Case

2.4.1 List

Use Case #ID	Description
ICL-UC-01	Derive optimal low-level context recognizer
ICL-UC-02	Create low-level context recognizers
ICL-UC-03	Route sensory data for the low-level context identification
ICL-UC-04	Recognize user low-level context
ICL-UC-05	Recognize user activity based on inertial raw sensory data
ICL-UC-06	Recognize user activity based on video raw sensory data
ICL-UC-07	Recognize user location based on geo-positioning raw sensory data
ICL-UC-08	Recognize user emotion based on audio raw sensory data
ICL-UC-09	Recognize user emotion based on video raw sensory data
ICL-UC-10	Recognize food based on tag data
ICL-UC-11	Unify low-level contexts
ICL-UC-12	Notify new low-level context
ICL-UC-13	Create unclassified high-level context instance
ICL-UC-14	Classify high-level context instance
ICL-UC-15	Notify new high-level context
ICL-UC-16	Load context ontology model
ICL-UC-17	Store context instance
ICL-UC-18	Retrieve context instance
ICL-UC-19	Evolve Ontology
ICL-UC-20	High Level Physical Activity Context
ICL-UC-21	High Level Nutrition Context
ICL-UC-22	High Level Clinical Context

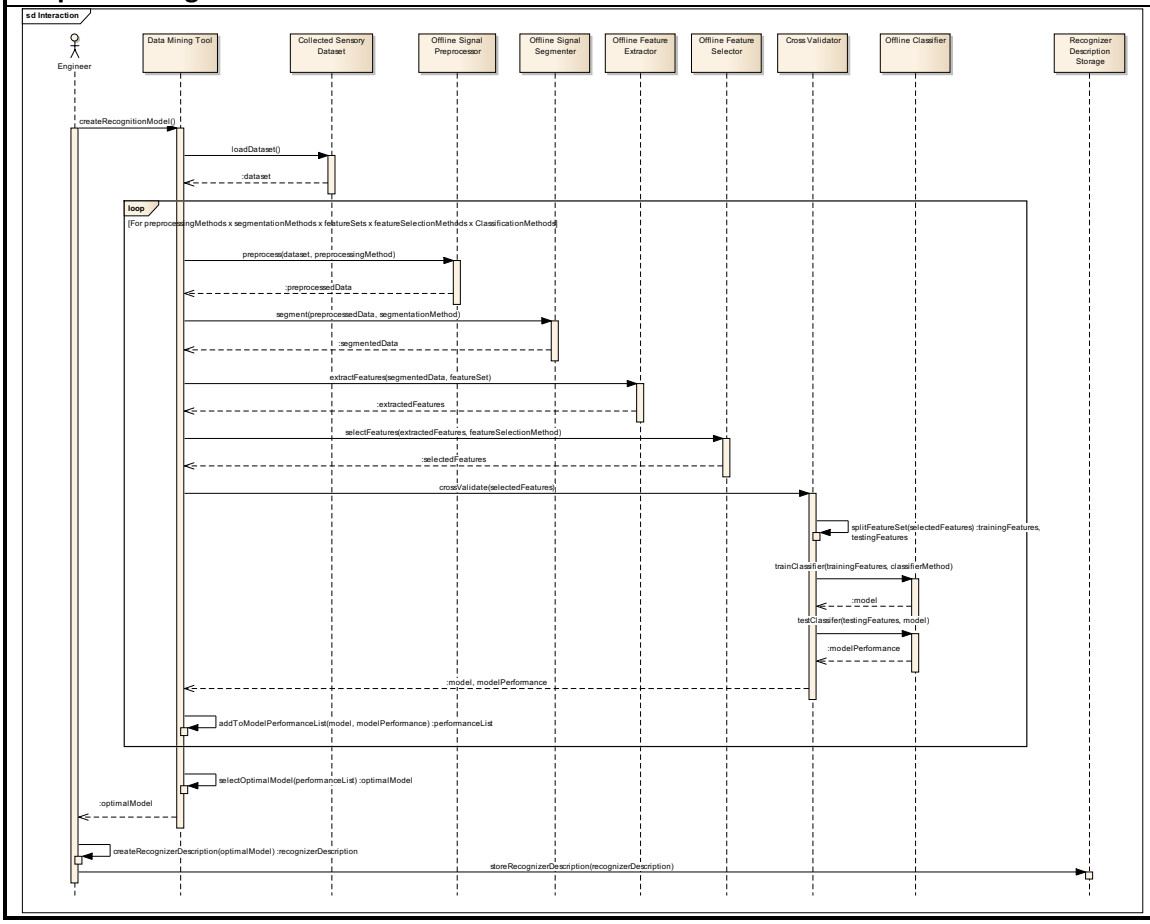
2.4.2 Diagram


uc UseCase Diagram HLCA 4.0
ICL4.0
LLCA
Create low-level context recognizer
precede
Derive optimal low-level context recognizer
ML Engineer
Route sensory data for low-level context
Recognize activity based on inertial data
Recognize activity based on video data
invokes
«actor»
DCL4.0
Recognize low-level context
Recognize emotion based on audio data
Recognize location based on geopositioning data
invokes
Unify low-level context
Recognize emotion based on video data
invokes
Recognize food nutrition based on Tag
Notify new low-level context
invokes
HLCA
Create unclassified high-level context
uses
Create ontology
Ontology engineer
invokes
uses
extend
Classify unclassified high-level context
Evolve Ontology
invokes
Notify new high-level context
High level physical activity context
High level nutrition context
High level clinical context

2.4.3 Description

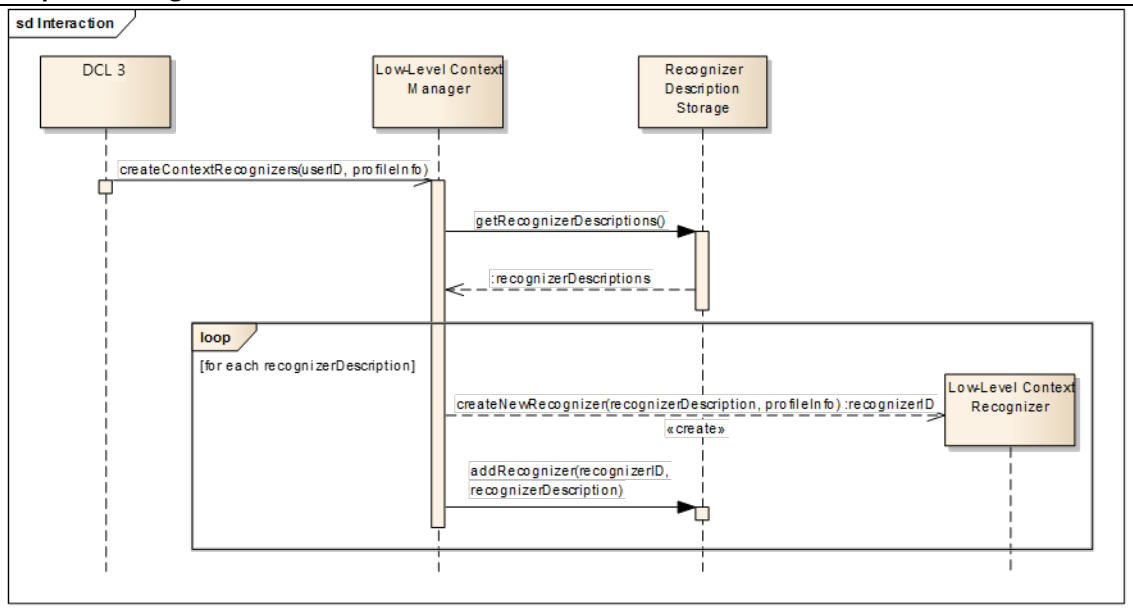
Use Case ID:	**ICL-UC-01**		
Use Case Name:	Derive optimal low-level context recognizer		
Created By:	Oresti Banos	**Last Updated By:**	Jaehun Bang
Date Created:	22 July 2015	**Last Revision Date:**	02 July 2018
Actors:	Engineer		
Description:	Create an optimal recognition model through the evaluation of multiple recognition model candidates (offline process).		
Trigger:	Engineer initiates the process for creating an optimal recognition model		
Pre-conditions:	• A human expert or engineer sets up the experimental setup for the evaluation process		
Post-conditions:	• The optimal recognition model among considered is delivered to the expert • A recognizer descriptor containing the characteristics of the optimal model is stored		
Normal Flow:	1. Load dataset 2. For each combination of preprocessing methods, segmentation methods, feature sets, feature selection methods, and classification methods, the dataset is preprocessed (e.g., filtered) 3. The preprocessed dataset is segmented (e.g., partitioned into windows) 4. Features (e.g., mean, variance) are extracted from each segment of the dataset 5. The best features are selected 6. Cross validation is applied to the selected features a. The feature set is split into training and testing b. The classifier is trained using the training features c. The classifier is tested in order to determine the model performance 7. The model performance is stored 8. Once the model performance has been calculated for all the possible combinations, the optimal model is selected 9. A recognizer descriptor is generated according to the characteristics of the model (e.g., median filtering, 3 sec window size, etc.) 10. The generated recognizer descriptor is stored		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Infrequent		
NFR ID:	MM-NFR-05		
Assumptions:	NA		

Notes and Issues:	Matlab and Weka tools will be used for this task. A multimodal dataset must be collected for the training and evaluation of the candidate's models.
Sequence Diagram:	



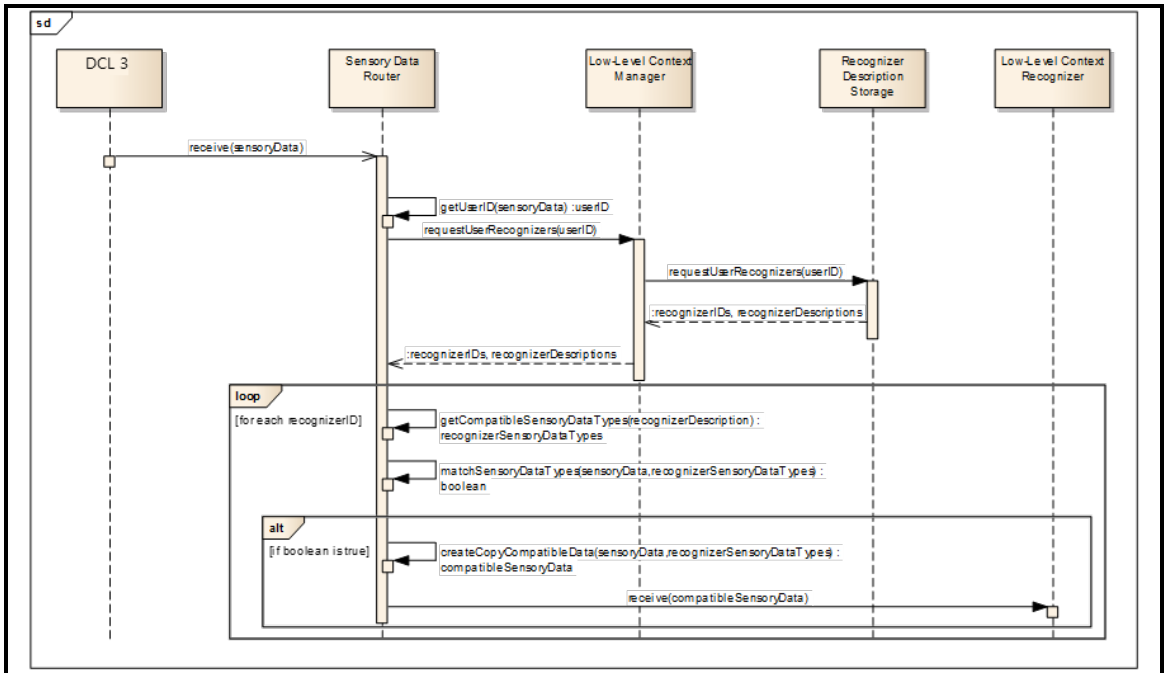
Use Case ID:	**ICL-UC-02**		
Use Case Name:	Create low-level context recognizers		
Created By:	Oresti Banos	**Last Updated By:**	Jaehun Bang
Date Created:	22 July 2015	**Last Revision Date:**	02 July 2018
Actors:	DCL 3		
Description:	Sensory data is received from DCL 3 and it is distributed to the corresponding low-level context recognizer based on the data type(s).		
Trigger:	Receive userID and part of the user profile information send by DCL 3 to ICL4		
Pre-conditions:	• DCL 3 sends the userID and part of the user profile information to ICL 4 whenever a new user is registered in the platform		
Post-conditions:	• Low-level context recognizers are generated for the new user for all context types		

Normal Flow:	1. Receive UserID and (part of the) user profile information 2. Load the recognizer descriptions containing the low-level context model types (e.g., emotion recognizer) and characteristics (e.g., median filtering, 3 sec window size, etc.) 3. Create a new recognizer for each recognizer description 4. Create a recognizer identifier for the generated recognizer 5. Save the recognizer identifier in a persistent storage
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less frequent
NFR ID:	NA
Assumptions:	• DCL 3 will send the required user profile information together with the userID only the first time a user is registered • No user profile updates are considered in this version
Notes and Issues:	NA
Sequence Diagram:	



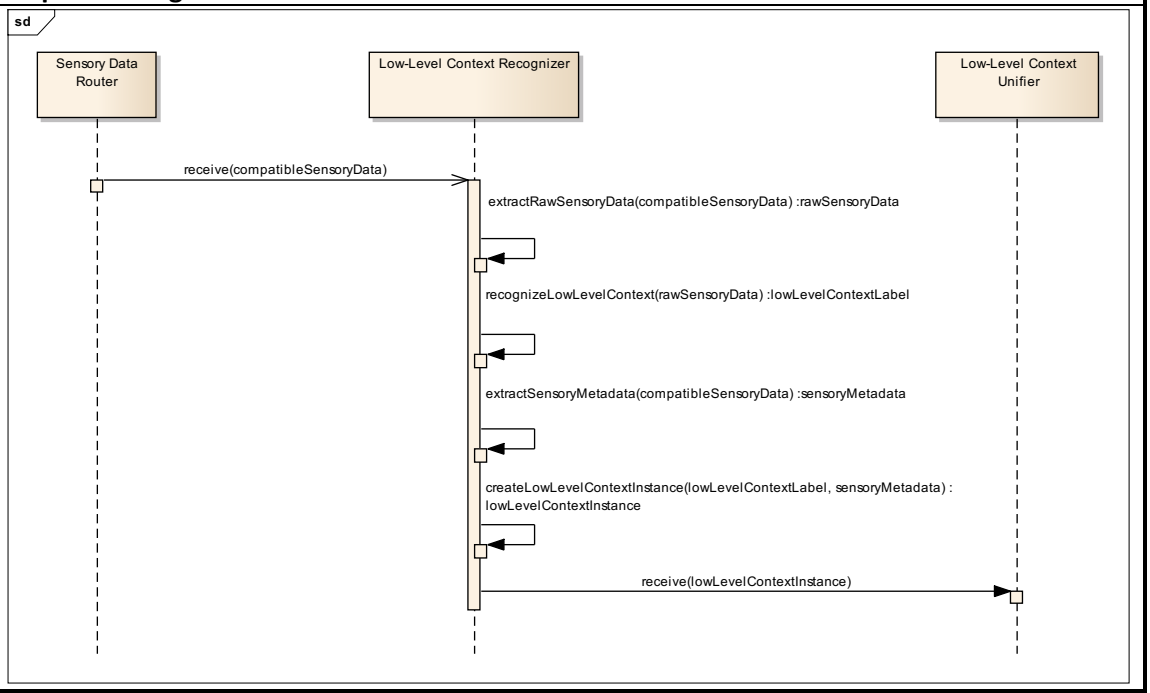
Use Case ID:	**ICL-UC-03**		
Use Case Name:	Route sensory data for the low-level context identification		
Created By:	Tae Ho Hur	**Last Updated By:**	Taeho Hur
Date Created:	06 Dec 2015	**Last Revision Date:**	04 July 2018
Actors:	Sensor Devices		
Description:	Sensory data is received, and it is distributed to the corresponding low-level context recognizer based on the data type(s).		
Trigger:	Receive sensory data send by each sensor device		

Pre-conditions:	• Device sends sensory data, i.e., raw sensory data plus sensory metadata (e.g., data type, time stamp, device ID, device type, and user ID)
Post-conditions:	• The adequate raw sensory data is sent to each low-level context recognizer in order to perform the recognition process
Normal Flow:	1. Receive sensory data 2. Get the user identifier to which the sensory data belongs 3. Load the low-level context recognizers identifiers for the given user 4. For each low-level context recognizer identifier, get the sensory data type(s) it requires 5. Match the received sensory data with the sensory data type(s) required by the low-level context recognizer 6. Create a copy with the compatible data required by the low-level context recognizer 7. Distribute the data to the corresponding low-level context recognizer
Alternative Flows:	NA
Exceptions:	5a. If no compatible data types are identified for the given low-level context recognizer 1. Go to step 3
Includes:	NA
Frequency of Use:	Very frequent
Assumptions:	• There is an established communication between Sensor Devices and the Sensory Data Router module • Incoming sensory data is already preprocessed (i.e., without missing samples and with synchronized streams)
Notes and Issues:	NA
Sequence Diagram:	



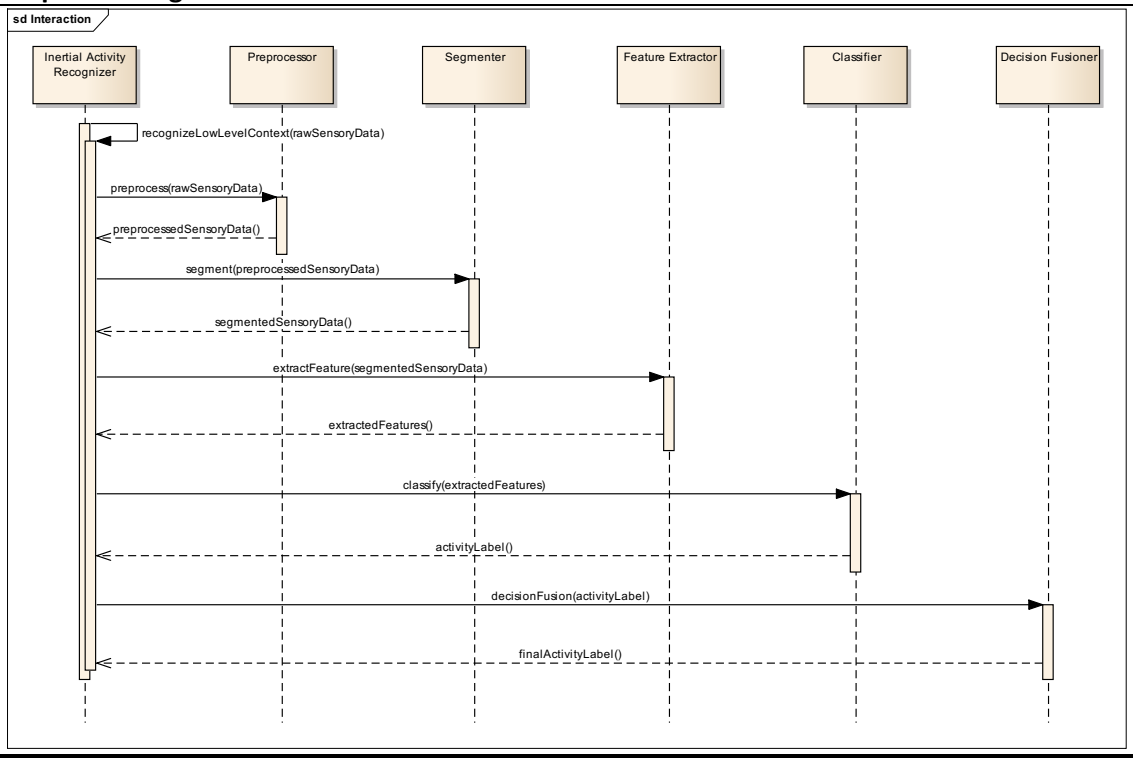
Use Case ID:	ICL-UC-04		
Use Case Name:	Recognize user low-level context		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	14 July 2015	**Last Revision Date:**	02 July 2018
Actors:	Sensory Data Router, Low-Level Context Unifier		
Description:	The low-level context associated to a given user is identified based on the received compatible sensory data. The low-level context recognition may be of diverse nature depending upon the data types, thus this use case defines an abstract representation of the process to be followed.		
Trigger:	Receive compatible sensory data		
Pre-conditions:	• Compatible sensory data is sent to a given low-level context recognizer		
Post-conditions:	• The recognized low-level context instance is provided to the Low-Level Context Unifier		
Normal Flow:	1. Compatible sensory data is received by a given low-level context recognizer 2. The raw sensory data is extracted from the sensory data 3. The low-level context label is recognized 4. The sensory metadata is extracted from the sensory data 5. A low-level context instance is generated by combining the low-level context label and the sensory metadata 6. The generated low-level context instance is provided to the Low-Level Context Unifier		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		

Frequency of Use:	Frequent: at every reception of sensory data
NFR ID:	MM-NFR-05
Assumptions:	• Only compatible sensory data is received by each corresponding low-level context recognizer
Notes and Issues:	NA
Sequence Diagram:	

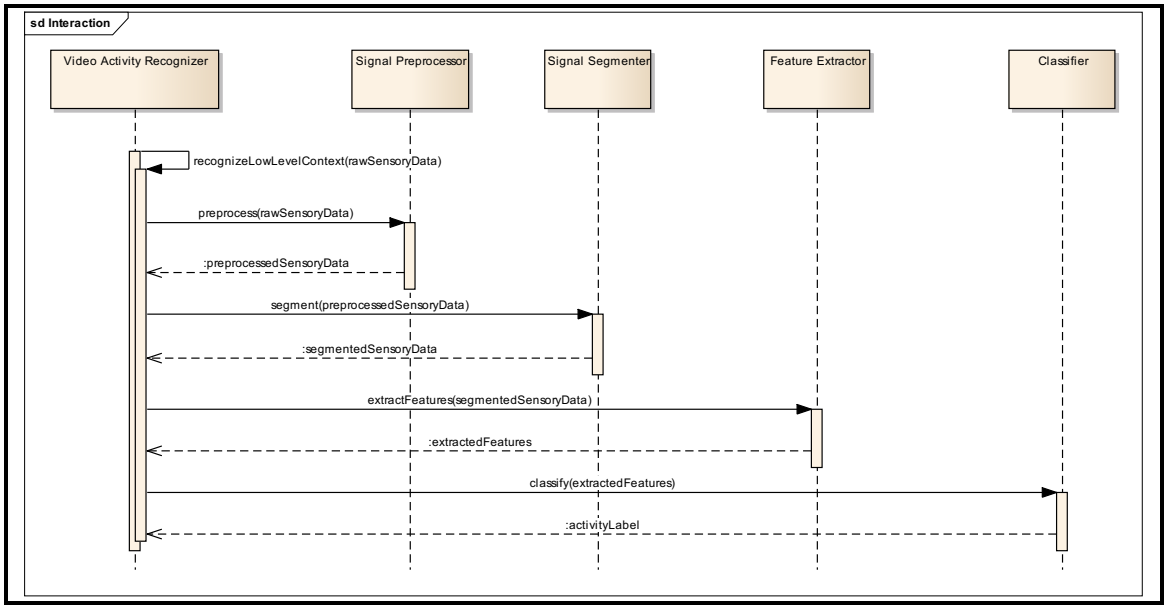


Use Case ID:	**ICL-UC-05**		
Use Case Name:	Recognize user activity based on inertial raw sensory data		
Created By:	Taeho Hur	**Last Updated By:**	Taeho Hur
Date Created:	09 March 2017	**Last Revision Date:**	02 July 2018
Actors:	ICL-UC-04		
Description:	Identification of the user physical activity (e.g., “sitting”) based on the processing of the body-motion raw sensory data collected from an inertial sensor. The body-motion raw sensory data consists of triaxial acceleration, triaxial rate of turn and triaxial magnetic field data.		
Trigger:	Request for the recognition of the user activity based on a given inertial raw sensory data		
Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (inertial sensory data)		
Post-conditions:	• A label corresponding to the recognized activity is generated		

Normal Flow:	1. Inertial raw sensory data is received for analysis 2. The raw sensory data is preprocessed (e.g., filtered) 3. The preprocessed raw sensory data is segmented (e.g., partitioned into windows) 4. Features (e.g., mean, variance) are extracted from each segment of raw sensory data 5. The extracted features are classified 6. A label identifying the corresponding user activity is generated 7. Decision fusion of activities is made based on majority voting 8. Final activity is chosen
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent: at every reception of inertial raw sensory data
NFR ID:	MM-NFR-05
Assumptions:	The raw sensory data is of the nature required by the inertial activity recognizer
Notes and Issues:	NA
Sequence Diagram:	

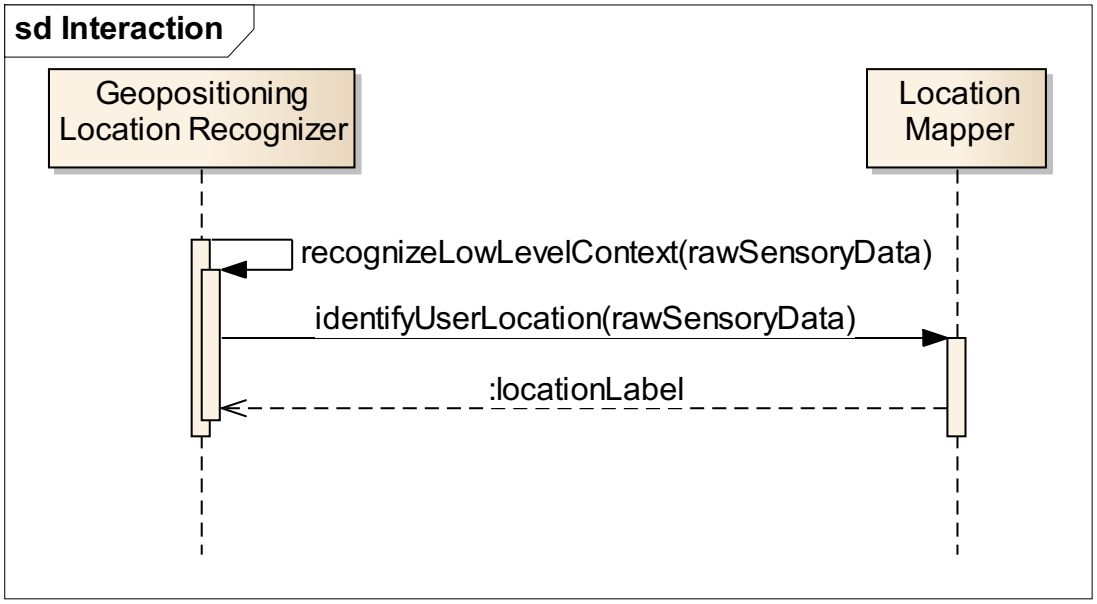


Use Case ID:	**ICL-UC-06**		
Use Case Name:	Recognize user activity based on video raw sensory data		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	17 July 2015	**Last Revision Date:**	02 July 2018
Actors:	ICL-UC-04		
Description:	Identification of the user physical activity (e.g., "standing") based on the processing of the body-motion raw sensory data collected through a video camera. The body-motion raw sensory data consists of RGB and depth video.		
Trigger:	Request for the recognition of the user activity based on a given video raw sensory data		
Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (video sensory data)		
Post-conditions:	• A label corresponding to the recognized activity is generated		
Normal Flow:	1. Video raw sensory data is received for analysis 2. The raw sensory data is preprocessed (e.g., filtered) 3. The preprocessed raw sensory data is segmented (e.g., partitioned into windows) 4. Features (e.g., SIFT, HOG) are extracted from each segment of raw sensory data 5. The extracted features are classified 6. A label identifying the corresponding user activity is generated		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Frequent: at every reception of video raw sensory data		
NFR ID:	MM-NFR-05		
Assumptions:	The raw sensory data is of the nature required by the video activity recognizer		
Notes and Issues:	NA		
Sequence Diagram:			



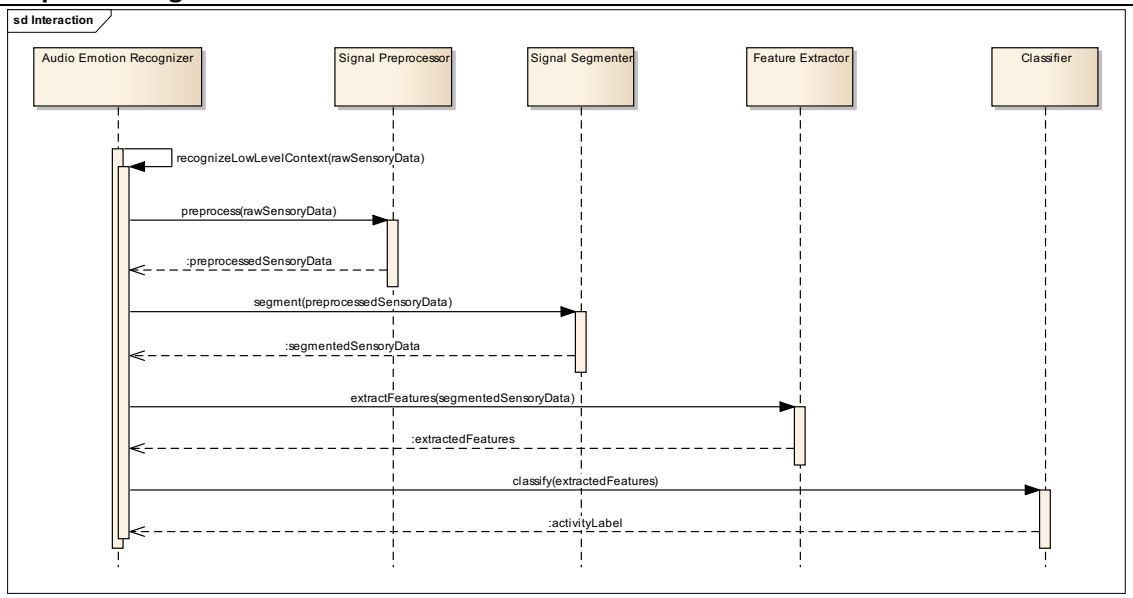
Use Case ID:	**ICL-UC-07**		
Use Case Name:	Recognize user location based on geo-positioning raw sensory data		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	17 July 2015	**Last Revision Date:**	02 July 2018
Actors:	ICL-UC-04		
Description:	Identification of the user location (e.g., “restaurant”) based on the processing of the geo-positioning raw sensory data collected from a portable GPS sensor. The body-motion raw sensory data consists of latitude, longitude and speed data.		
Trigger:	Request for the recognition of the user location based on a given geo-positioning raw sensory data		
Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (geo-positioning sensory data)		
Post-conditions:	• A label corresponding to the recognized location is generated		
Normal Flow:	1. Geo-positioning raw sensory data is received for analysis 2. The geo-positioning raw sensory data is compared with the predefined map coordinates 3. A label identifying the corresponding user location is generated		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Frequent: at every reception of geo-positioning raw sensory data		
NFR ID:	MM-NFR-05		
Assumptions:	The raw sensory data is of the nature required by the geo-positioning location recognizer		

Notes and Issues:	NA
Sequence Diagram:	



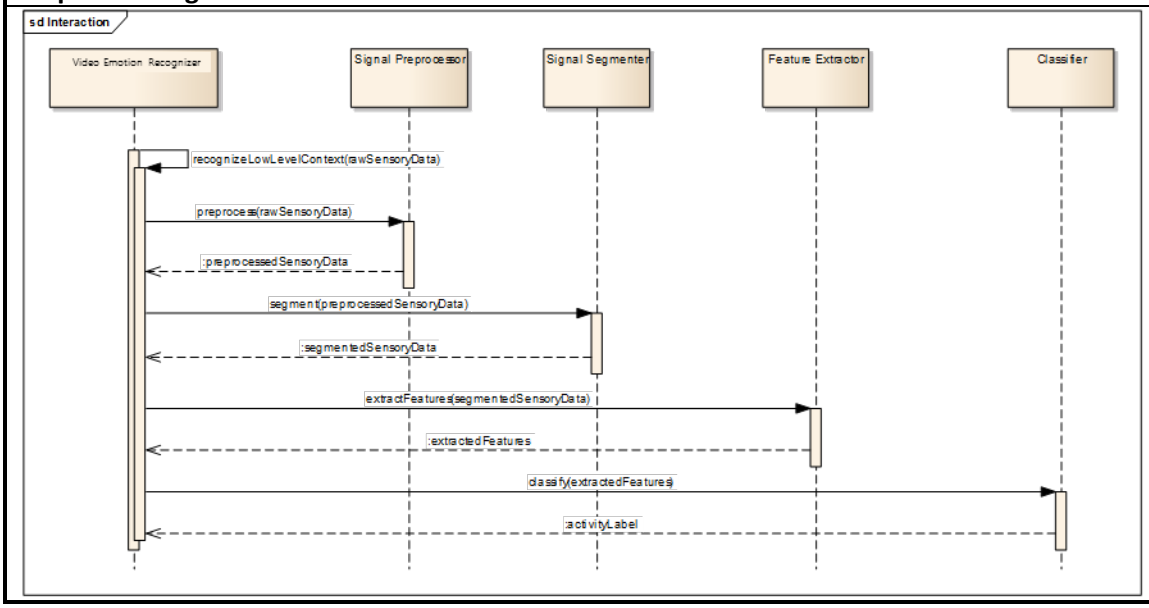
Use Case ID:	**ICL-UC-08**		
Use Case Name:	Recognize user emotion based on audio raw sensory data		
Created By:	Oresti Banos	**Last Updated By:**	Jaehun Bang
Date Created:	17 July 2015	**Last Revision Date:**	02 July 2018
Actors:	ICL-UC-04		
Description:	Identification of the user emotional state (e.g., “happy”) based on the processing of the audio raw sensory data collected from a microphone sensor. The audio raw sensory data consists of the user voice data.		
Trigger:	Request for the recognition of the user emotion based on a given audio raw sensory data		
Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (audio sensory data)		
Post-conditions:	• A label corresponding to the recognized emotion is generated		
Normal Flow:	1. Audio raw sensory data is received for analysis 2. The raw sensory data is preprocessed (e.g., filtered) 3. The preprocessed raw sensory data is segmented (e.g., partitioned into windows) 4. Features (e.g., LPC, MFCC) are extracted from each segment of raw sensory data 5. The extracted features are classified 6. A label identifying the corresponding user emotion is generated		
Alternative Flows:	NA		
Exceptions:	NA		

Includes:	NA
Frequency of Use:	Frequent: at every reception of inertial raw sensory data
Assumptions:	The raw sensory data is of the nature required by the audio emotion recognizer
Notes and Issues:	NA
Sequence Diagram:	



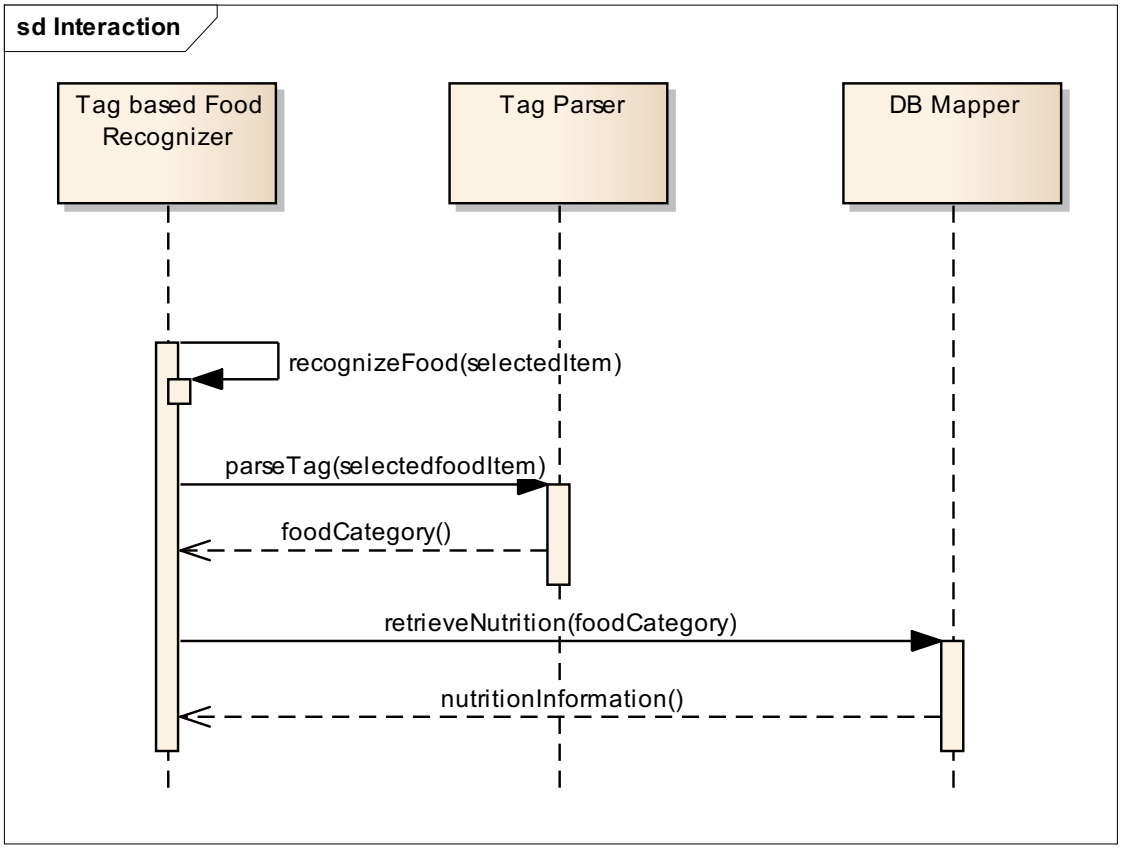
Use Case ID:	**ICL-UC-09**		
Use Case Name:	Recognize user emotion based on video raw sensory data		
Created By:	Jaehun Bang	**Last Updated By:**	Jaehun Bang
Date Created:	9 March 2017	**Last Revision Date:**	02 July 2018
Actors:	ICL-UC-04		
Description:	Identification of the user emotional state (e.g., “happy”) based on the processing of the video raw sensory data collected from a camera sensor. The video raw sensory data consists of the user facial data.		
Trigger:	Request for the recognition of the user emotion based on a given video raw sensory data		
Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (video sensory data)		
Post-conditions:	• A label corresponding to the recognized emotion is generated		
Normal Flow:	1. Video raw sensory data is received for analysis 2. The raw sensory data is preprocessed (e.g., filtered and face detection) 3. The preprocessed raw sensory data is segmented (e.g., partitioned into face area) 4. Features (e.g., face factor statistical features) are extracted from each segment of raw sensory data		

	5. The extracted features are classified 6. A label identifying the corresponding user emotion is generated
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent: at every reception of inertial raw sensory data
NFR ID:	MM-NFR-05
Assumptions:	The raw sensory data is of the nature required by the video emotion recognizer
Notes and Issues:	NA
Sequence Diagram:	



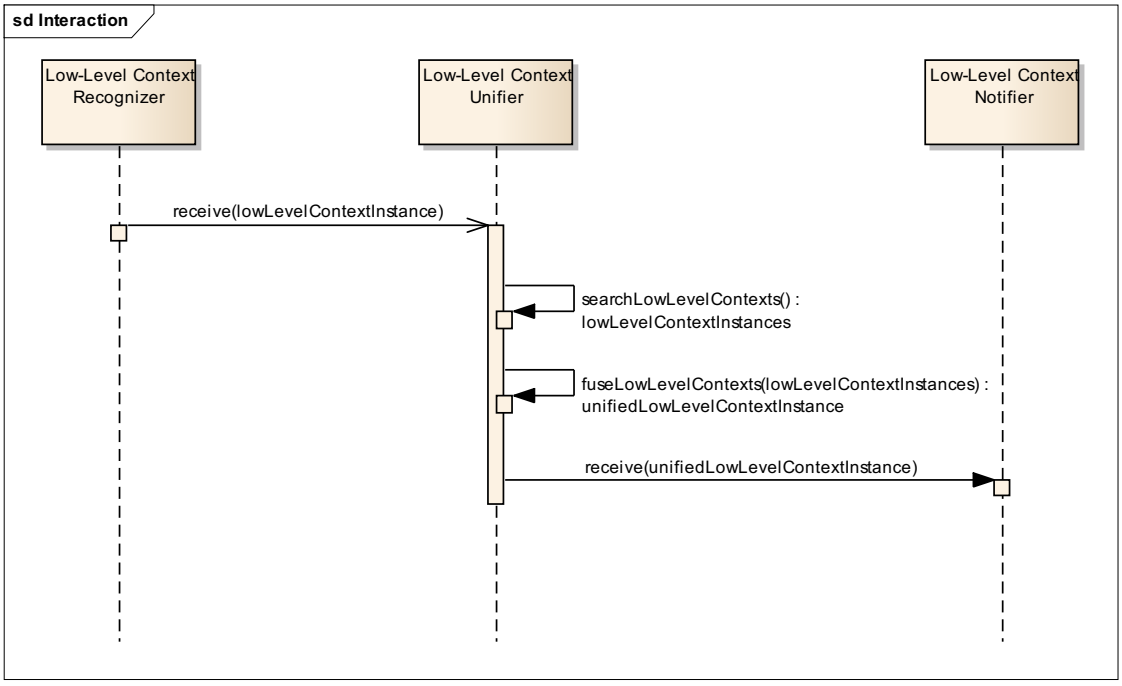
Use Case ID:	**ICL-UC-10**		
Use Case Name:	Recognize food based on tag data		
Created By:	Taeho Hur	**Last Updated By:**	Taeho Hur
Date Created:	9 March 2017	**Last Revision Date:**	02 July 2018
Actors:	ICL-UC-04		
Description:	User selects the category of the intaken food (e.g., “grain”) from the food DB list from the app. The related nutrition information (e.g, “protein”) shall be sent to the HLCA.		
Trigger:	User selects the food category tag		
Pre-conditions:	• User is going to have a meal/snack		
Post-conditions:	• Nutrition information is retrieved based on the selected food		
Normal Flow:	1. User selects the food to intake from the app 2. The selected food item is parsed and conclude the food category 3. The category is mapped to the food DB 4. According nutrition information is retrieved		

Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Episodic: Only when the user has a meal
NFR ID:	MM-NFR-05
Assumptions:	User is having a meal
Notes and Issues:	NA
Sequence Diagram:	



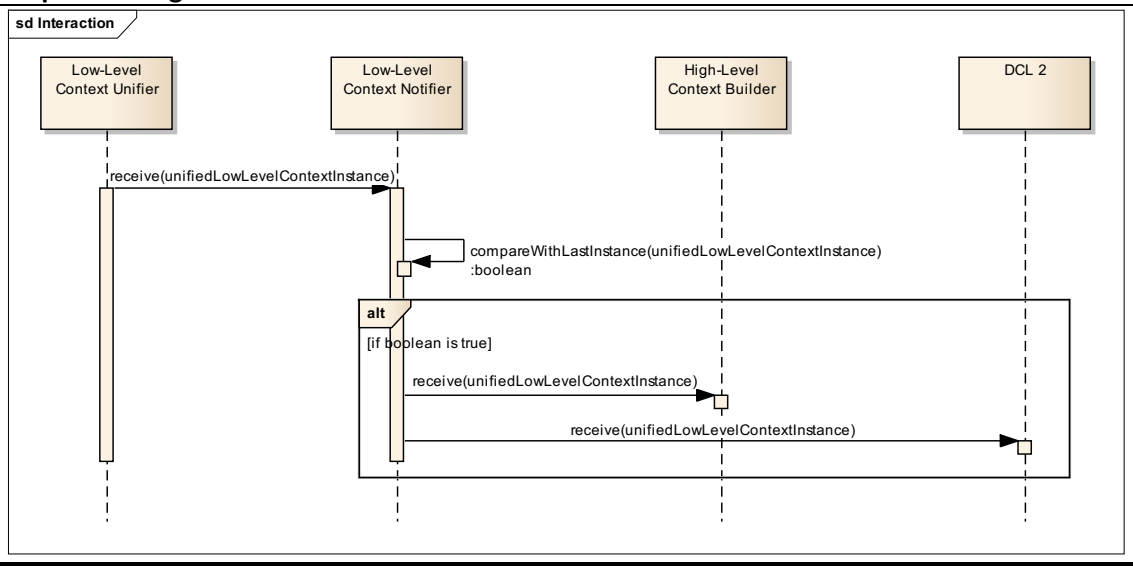
Use Case ID:	**ICL-UC-11**		
Use Case Name:	Unify low-level contexts		
FR ID:	MM-FR-11		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	20 July 2015	**Last Revision Date:**	02 July 2018
Actors:	Low-Level Context Recognizer, Low-Level Context Notifier		
Description:	Aggregation of multiple low-level context instances of the same context type (e.g., activity) corresponding to a similar period of time		
Trigger:	Receive low-level context instance		
Pre-conditions:	• Low-level context instances are received from different recognizers of the same context type		

Post-conditions:	• A single low-level context instance is served for notification
Normal Flow:	1. A low-level context instance is received 2. Search for other low-level context instances of the same type valid at the same time 3. Fuse the identified low-level context instances into a unified low-level context instance 4. Serve the unified low-level context instance for notification
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent: at every reception of a low-level context label
NFR ID:	MM-NFR-05
Assumptions:	Identical labels are used to describe the same low-level context for each recognizer of the same context type (e.g., inertial activity recognizer, video activity recognizer)
Notes and Issues:	NA
Sequence Diagram:	



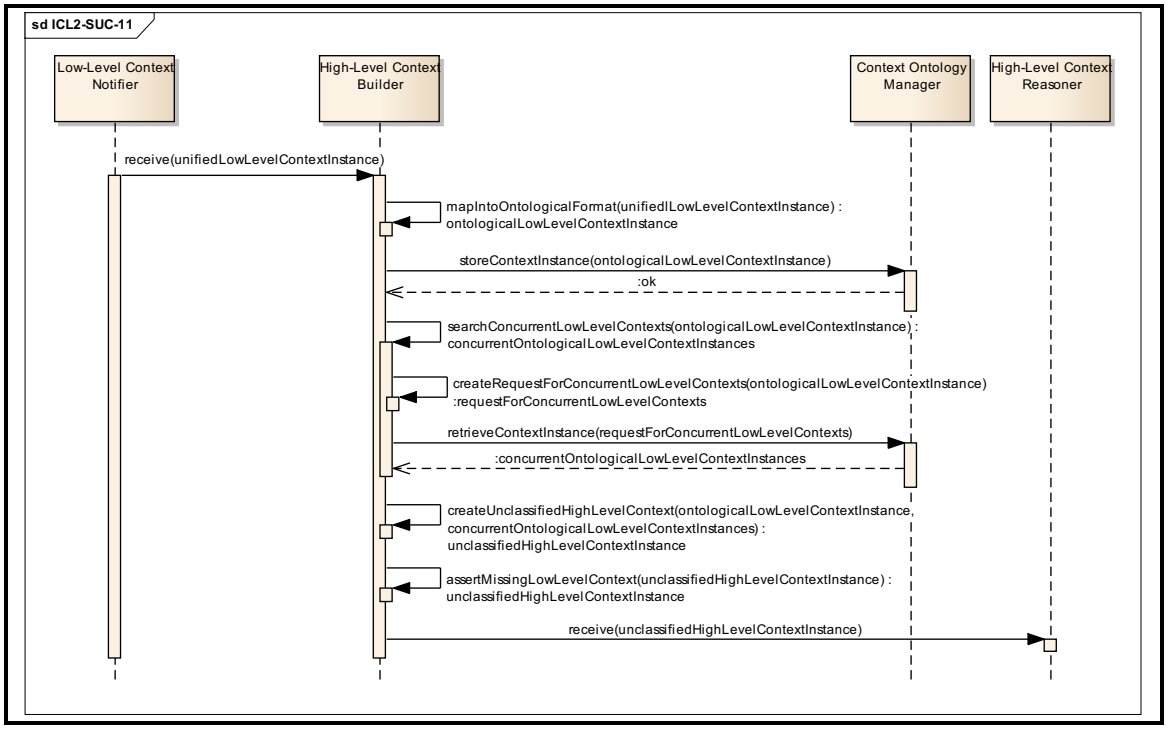
Use Case ID:	**ICL-UC-12**		
Use Case Name:	Notify new low-level context		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	14 July 2015	**Last Revision Date:**	02 July 2018
Actors:	Low-Level Context Unifier, High-Level Context Builder, DCL		

Description:	Serve the newly recognized low-level context for the identification of high-level context and also communicate it to DCL for persistence.
Trigger:	New low-level context is identified
Pre-conditions:	• A unified low-level context instance is received
Post-conditions:	• The unified low-level context instance is served for the identification of the high-level context(s) • The unified low-level context instance is sent to DCL 3
Normal Flow:	1. A low-level context instance is received from the low level context unifier 2. The received instance is compared with the last low-level context instance 3. The new low-level context instance is served for the identification of the high-level context 4. The new low-level context instance is sent to DCL 3
Alternative Flows:	3a. If the received instance contains the same low-level context type as the previous one 1. Finalize
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent: at every reception of a low-level context instance
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	



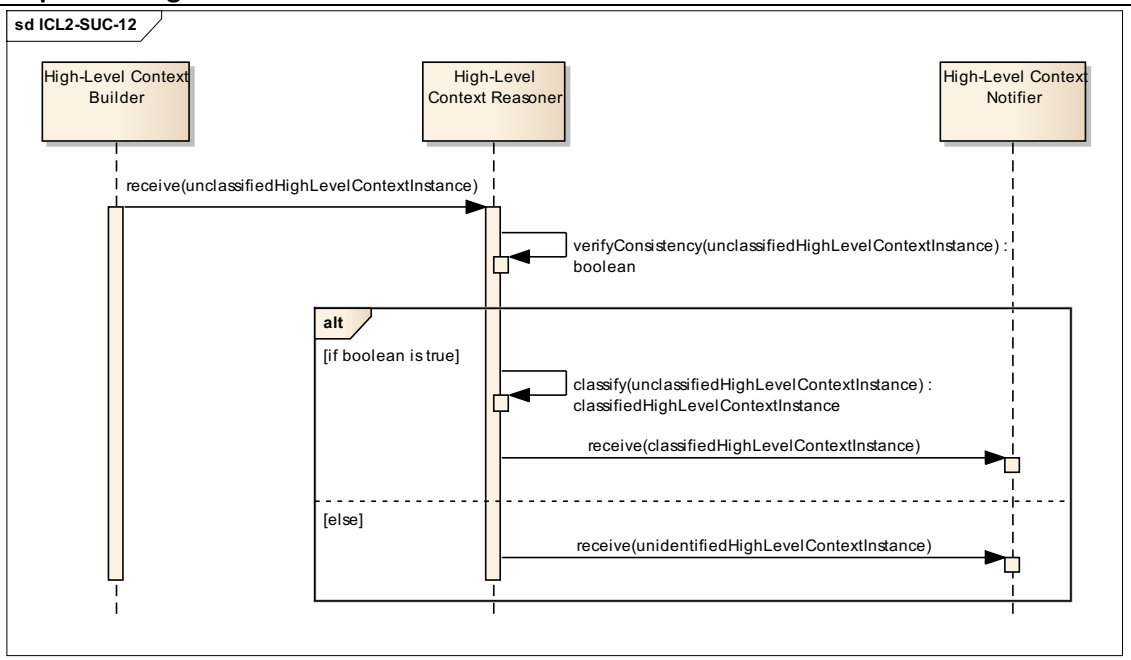
Use Case ID:	**ICL-UC-13**		
Use Case Name:	Create unclassified high-level context instance		
Created By:	Claudia Villalonga	**Last Updated By:**	Claudia Villalonga

Date Created:	14 July 2015	**Last Revision Date:**	02 July 2018
Actors:	Low-Level Context Notifier, Context Ontology Manager, High-Level Context Reasoner		
Description:	Build a high-level context instance based on the identified low-level contexts		
Trigger:	Receive low-level context instance (label plus metadata)		
Pre-conditions:	• A new low-level context instance is served to the high-level context builder		
Post-conditions:	• The unclassified high-level context instance is created		
Normal Flow:	1. Map low-level context instance into ontological format 2. Store low-level context instance (ICL4-SUC-17) 3. Search for other low-level context instances of different type valid at the same time 3.1. Create a request for other low-level context instances of different type that are valid at the same time 3.2. Retrieve matching context instances (ICL4-SUC-18) 4. Create new unclassified high-level context instance which links to the available low-level context instance(s) 5. Assert on the unclassified high-level context instance that the missing low-level context instances do not exist 6. Serve the unclassified high-level context instance for classification		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Less Frequent: whenever a new low-level context is recognized		
NFR ID:	NA		
Assumptions:	Low-level contexts are interpretable		
Notes and Issues:	NA		
Sequence Diagram:			



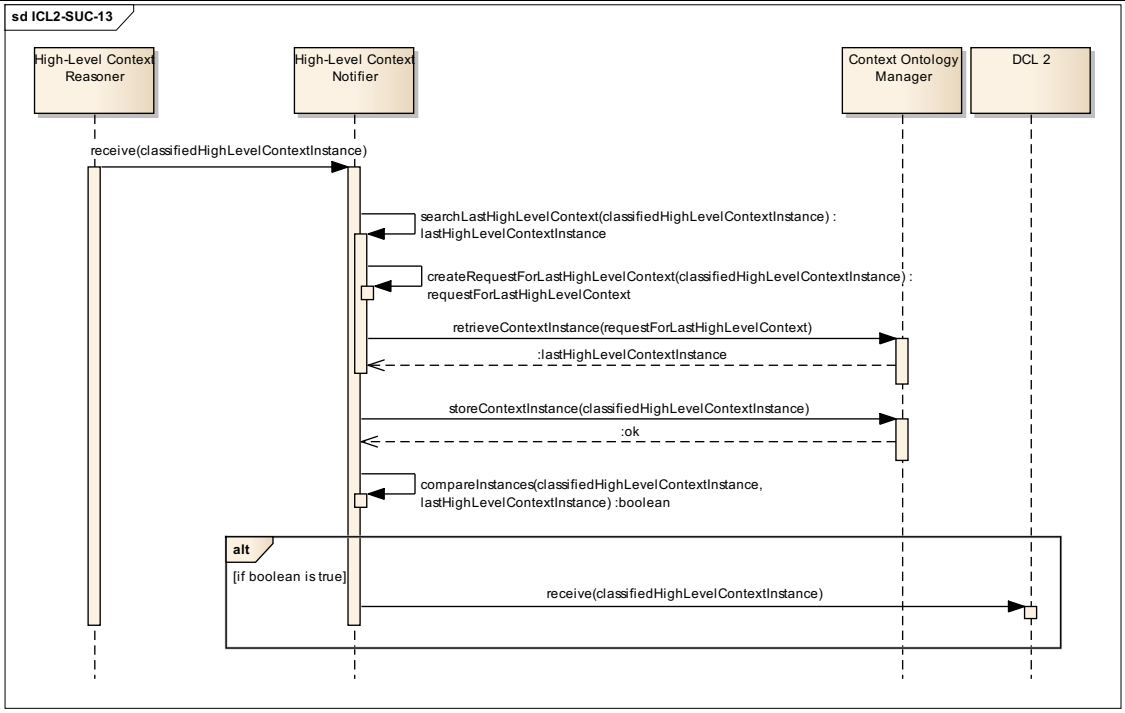
Use Case ID:	**ICL-UC-14**		
Use Case Name:	Classify high-level context instance		
FR ID:	MM-FR-12		
Created By:	Claudia Villalonga	**Last Updated By:**	Claudia Villalonga
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	High-Level Context Builder, High-Level Context Notifier		
Description:	Classify the unclassified high-level context instance into one of the high-level context categories		
Trigger:	Creation of unclassified high-level context instance		
Pre-conditions:	• The unclassified high-level context instance is created		
Post-conditions:	• The classified high-level context instance is served for notification		
Normal Flow:	1. Verify the consistency of unclassified high-level context instance 2. Reason on the unclassified high-level context instance to identify the context type to which it belongs 3. Serve the classified high-level context for notification		
Alternative Flows:	1a. If the unclassified high-level context instance is not valid 1. Communicate unidentified context		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Less Frequent: whenever an unclassified high-level context instance is created		
Assumptions:	Low-level contexts and high-level contexts are interpretable		

Notes and Issues:	NA
Sequence Diagram:	



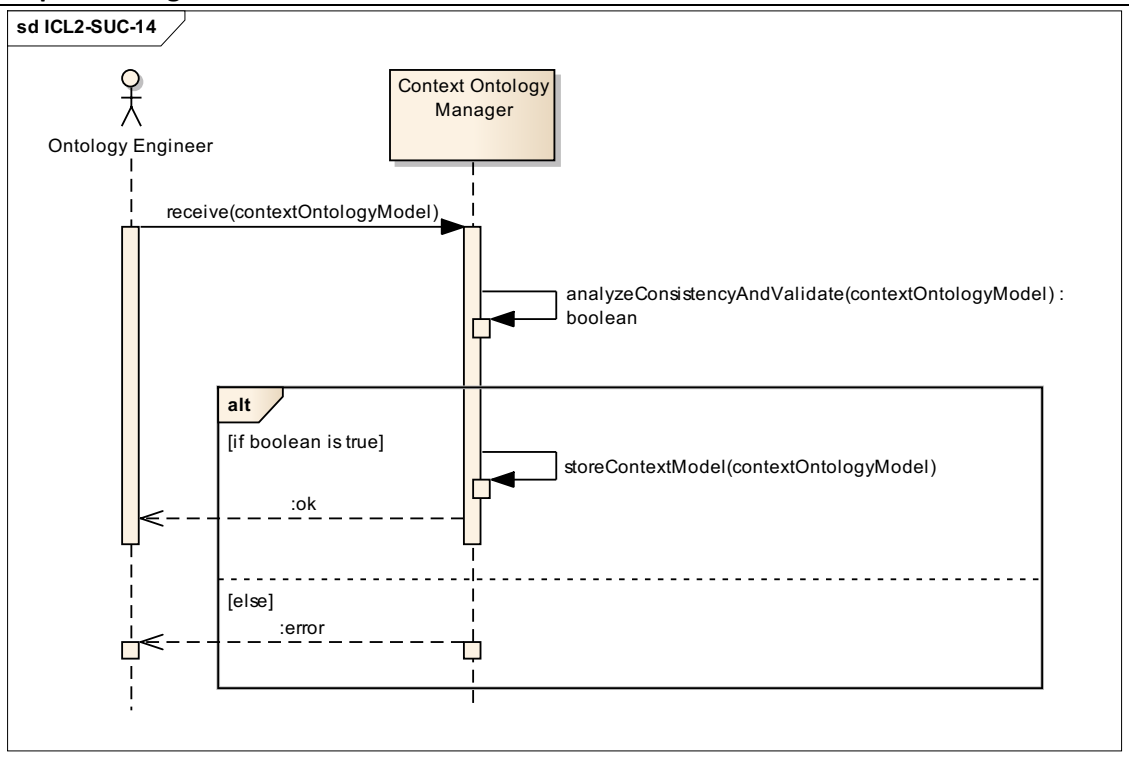
Use Case ID:	**ICL-UC-15**		
Use Case Name:	Notify new high-level context		
FR ID:	MM-FR-14		
Created By:	Oresti Banos	**Last Updated By:**	Claudia Villalonga
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	High-Level Context Reasoner, DCL		
Description:	Communicate the newly recognized high-level context to DCL for persistence.		
Trigger:	High-level context is identified		
Pre-conditions:	• A high-level context instance is received		
Post-conditions:	• The new high-level context instance is sent to DCL		
Normal Flow:	1. A high-level context instance is received from the high-level context classifier 2. Search for the last high-level context instance 2.1. Create a request for the last high-level context instance 2.2. Retrieve the matching context instance (ICL4-SUC-18) 3. Store high-level context instance (ICL4-SUC-17) 4. Compare the high-level context instance with the last high-level context instance 5. The new high-level context instance is sent to DCL 3		
Alternative Flows:	5a. If the received instance contains the same high-level context type as the previous one		

	1. Finalize
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less frequent: at every reception of a high-level context instance
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	



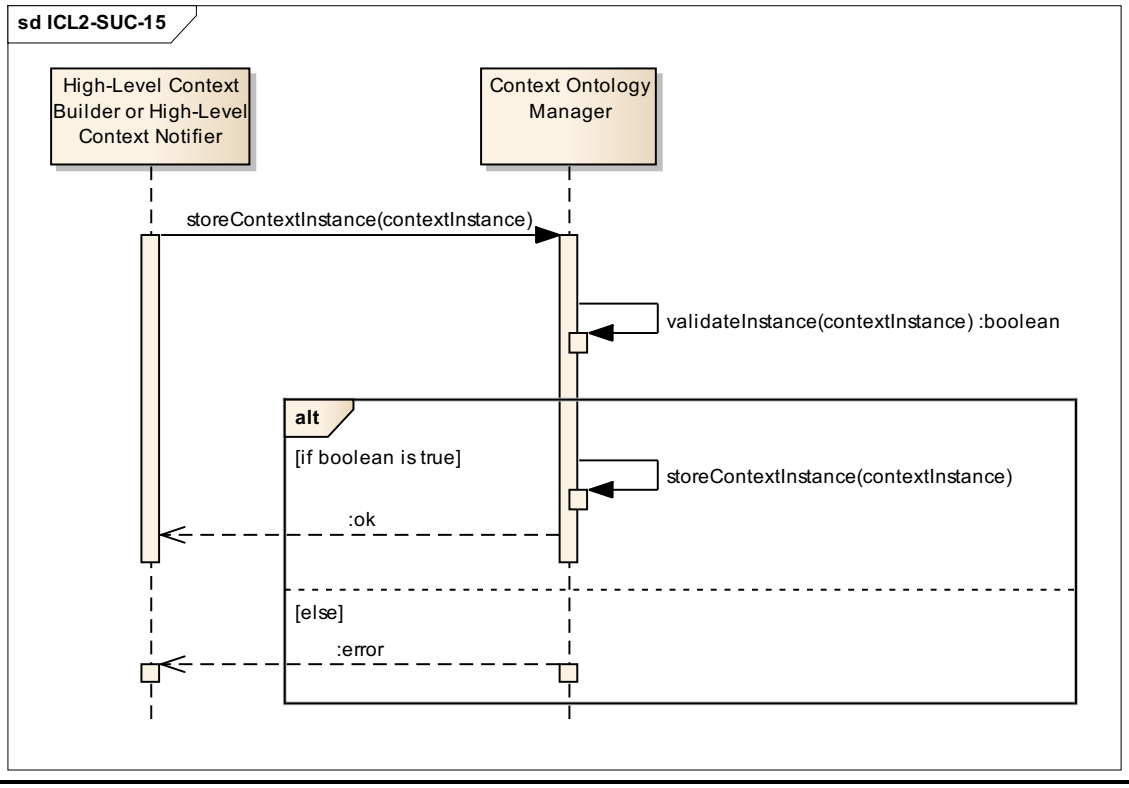
Use Case ID:	**ICL-UC-16**		
Use Case Name:	Load context ontology model		
Created By:	Claudia Villalonga	**Last Updated By:**	Claudia Villalonga
Date Created:	31 August 2015	**Last Revision Date:**	01 July 2018
Actors:	Ontology Engineer		
Description:	Load and store a context ontology model that describes high-level context and its relations to low-level context into the system in order to enable the recognition of high-level context		
Trigger:	The ontology engineer who has created a context ontology model loads it through the interface		
Pre-conditions:	• A context ontology model that describes high-level context and its relations to low-level context has been created		
Post-conditions:	• The context ontology model is stored and available for the recognition of high-level context		

Normal Flow:	1. A context ontology model is received 2. The context ontology model is analyzed for its consistency and validity 3. The context ontology is stored in order to provide persistence 4. Success is notified
Alternative Flows:	2a. If the context ontology model is not valid or inconsistent 1. Error is notified 3a. If there is an error during storage 1. Repeat step 3
Exceptions:	NA
Includes:	NA
Frequency of Use:	Infrequent
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	

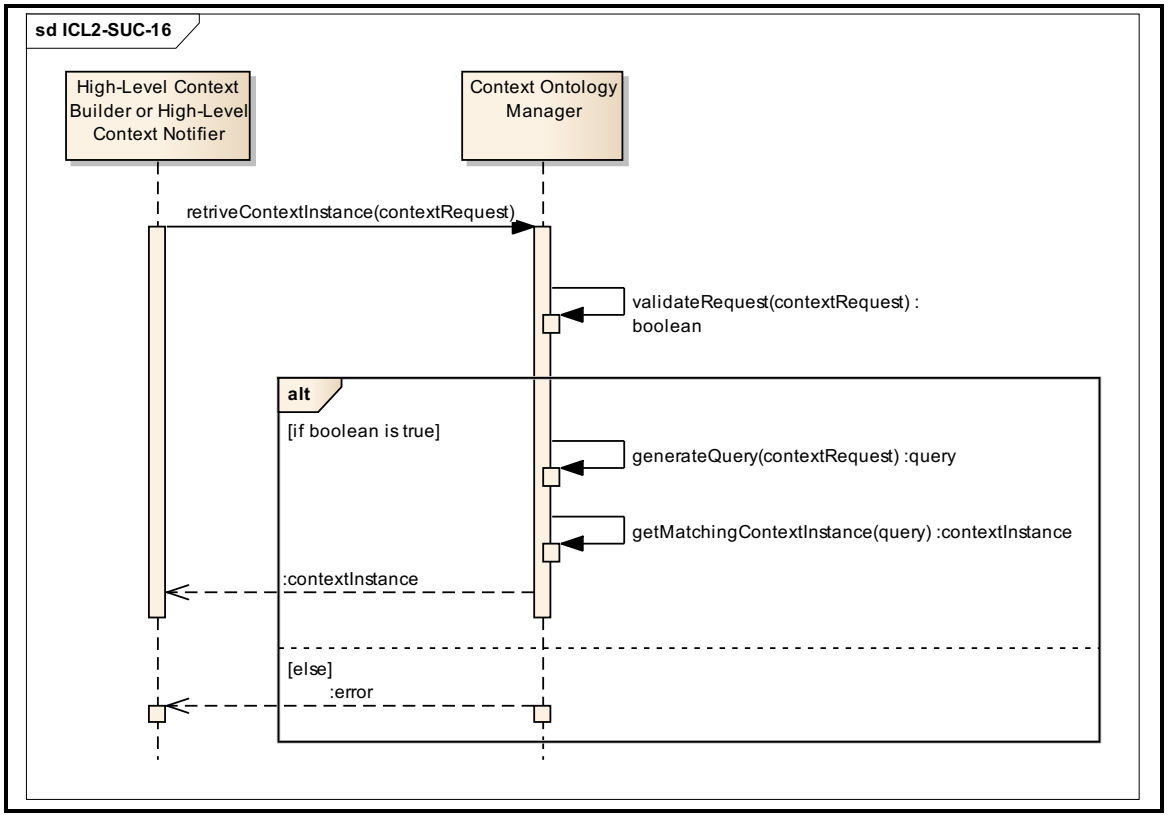


Use Case ID:	**ICL-UC-17**		
Use Case Name:	Store context instance		
Created By:	Claudia Villalonga	**Last Updated By:**	Claudia Villalonga
Date Created:	31 August 2015	**Last Revision Date:**	01 July 2018
Actors:	High-Level Context Builder and High-Level Context Notifier		

Description:	Persist a context instance (high-level context instance or low-level context instance) into the system
Trigger:	A new context instance has been created or identified
Pre-conditions:	• A new context instance is received
Post-conditions:	• The context instance is stored
Normal Flow:	1. Receive context instance 2. Validate the context instance 3. Store the context instance 4. Notify success
Alternative Flows:	2a. If the context instance is not valid 1. Error is notified 3a. If there is an error during storage 1. Repeat step 3
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less Frequent: whenever a new low-level context instance is created, a new high-level context instance is created, or a high-level context instance is classified
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	

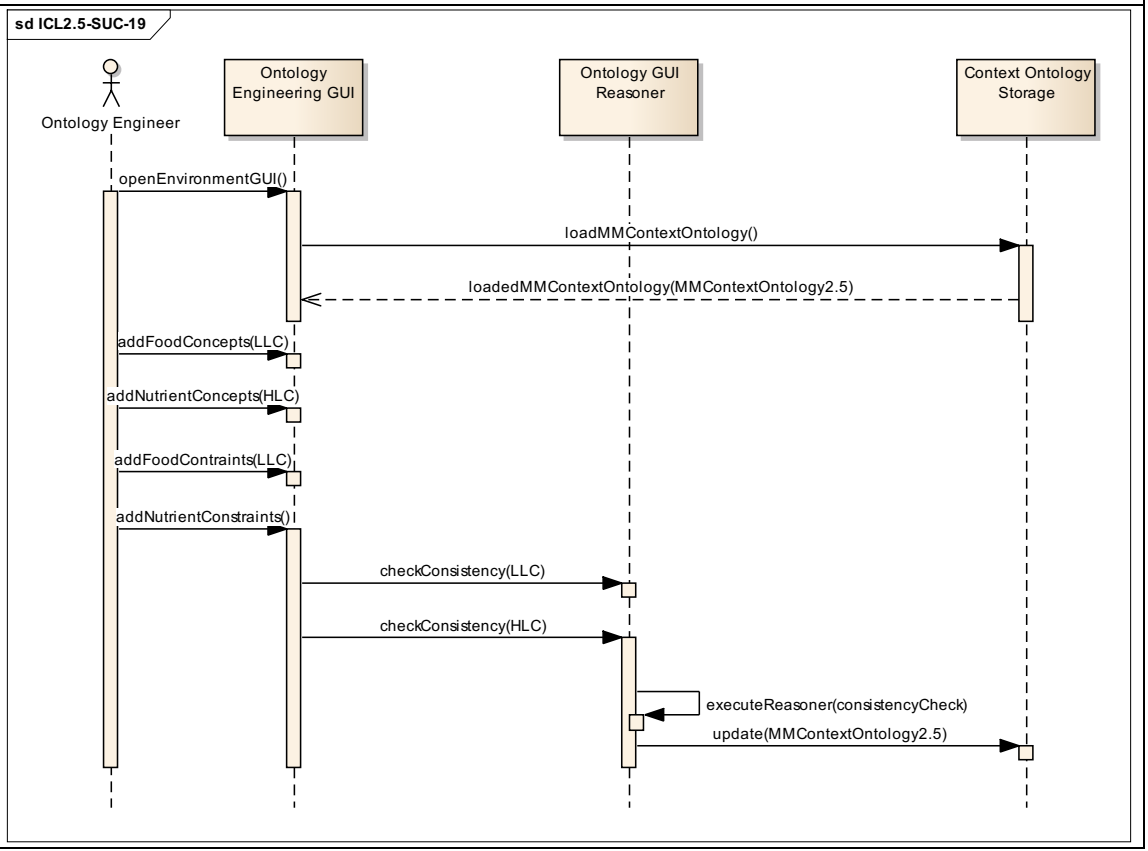


Use Case ID:	**ICL-UC-18**		
Use Case Name:	Retrieve context instance		
Created By:	Claudia Villalonga	**Last Updated By:**	Claudia Villalonga
Date Created:	31 August 2015	**Last Revision Date:**	01 July 2018
Actors:	High-Level Context Builder and High-Level Context Notifier		
Description:	Provide context instances (high-level context instances or low-level context instances) that match a given request		
Trigger:	A requester retrieves context instances		
Pre-conditions:	• A request for context instances is received		
Post-conditions:	• Matching context instances are provided to the requester		
Normal Flow:	1. Receive a request for a specific context 2. Validate the request 3. Generate the query associated to the request 4. Match the query to the stored context instances 5. Return the matching context instances		
Alternative Flows:	2a. If the request for context is not valid 1. Return error message		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Less frequent: whenever context instances are required in order to generate a new high-level context instance or to verify the high-level context		
NFR ID:	NA		
Assumptions:	NA		
Notes and Issues:	NA		
Sequence Diagram:			

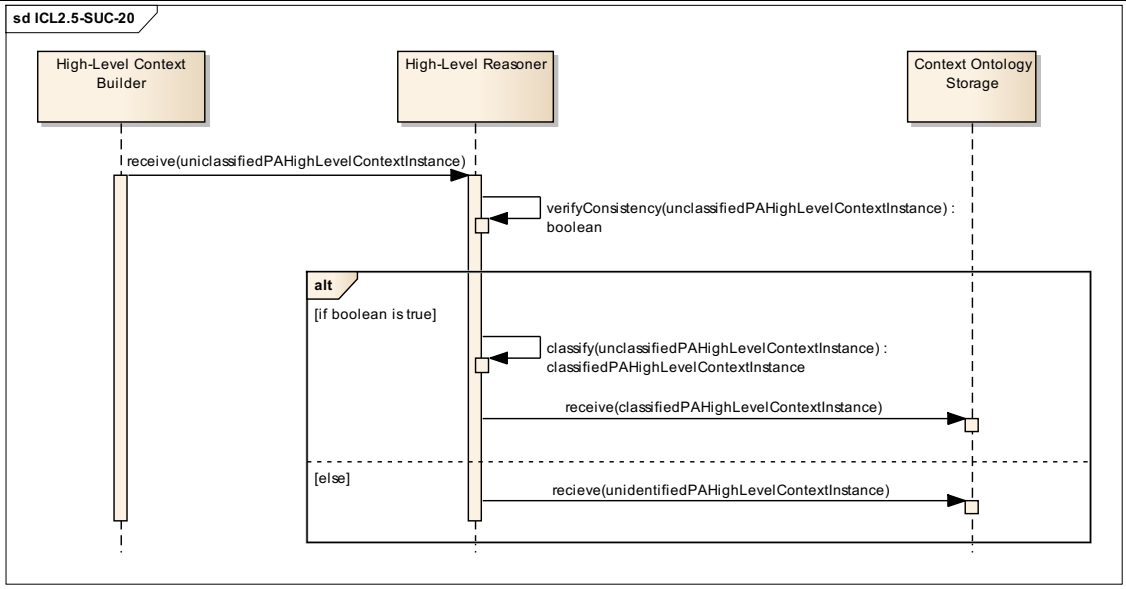


Use Case ID:	ICL-UC-19		
Use Case Name:	Evolve Ontology		
Created By:	Wajahat Ali Khan	**Last Updated By:**	Muhammad Asif Razzaq
Date Created:	18 Mar 2016	**Last Revision Date:**	01 July 2018
Actors:	Ontology Engineer		
Description:	Evolving the already created context ontology based on the clinical service. The low level and high level clinical related contexts are modelled in the context ontology that only included physical activities and nutrition related low level and high-level entities.		
Trigger:	The ontology engineer who has created a context ontology model loads it through the interface		
Pre-conditions:	• A context ontology model that describes high-level context and its relations to low-level context has been created with physical activities, nutrition and clinical information related resources		
Post-conditions:	• The modified or evolved context ontology model is stored and available for the recognition of high-level context		

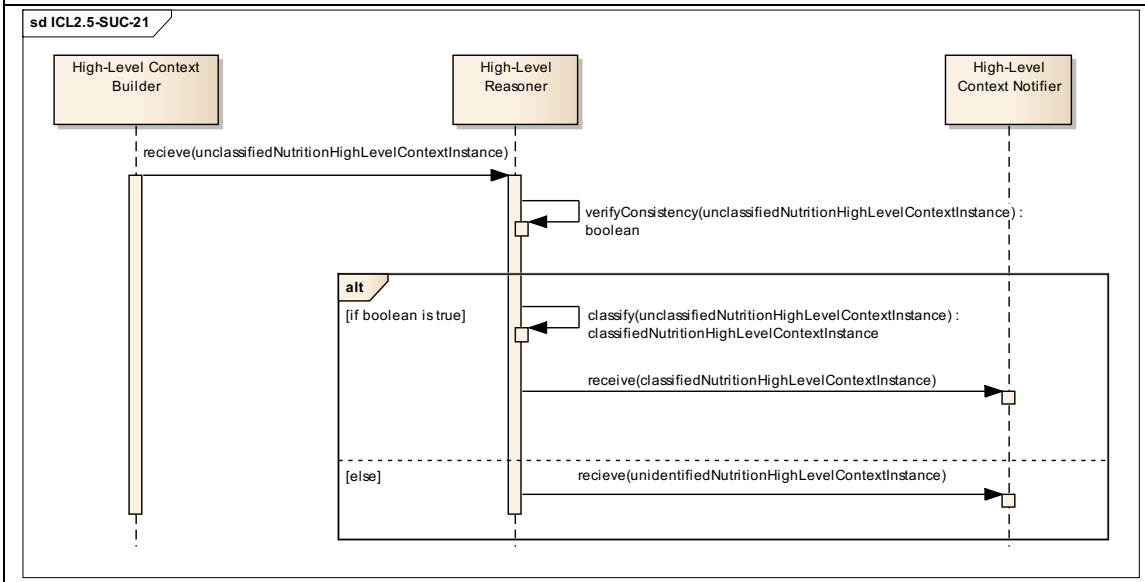
Normal Flow:	1. Ontology engineer defines the low level clinical concepts in addition to the low level physical activities, and nutrition concepts 2. Ontology engineer defines the high level clinical concepts in addition to the high level physical activities and nutrition concepts 3. The constraints are defined on the new concepts added to the ontology 4. Consistency of the ontology is checked with reasoner 5. The modified context ontology is stored for persistence 6. Success is notified
Alternative Flows:	4a. If the context ontology model is not valid or inconsistent a. Error is notified b. Check step 1,2,3
Exceptions:	NA
Includes:	NA
Frequency of Use:	Infrequent
NFR ID:	NA
Assumptions:	Context Ontology based on physical activities and nutrition service already exists
Notes and Issues:	NA
Sequence Diagram:	



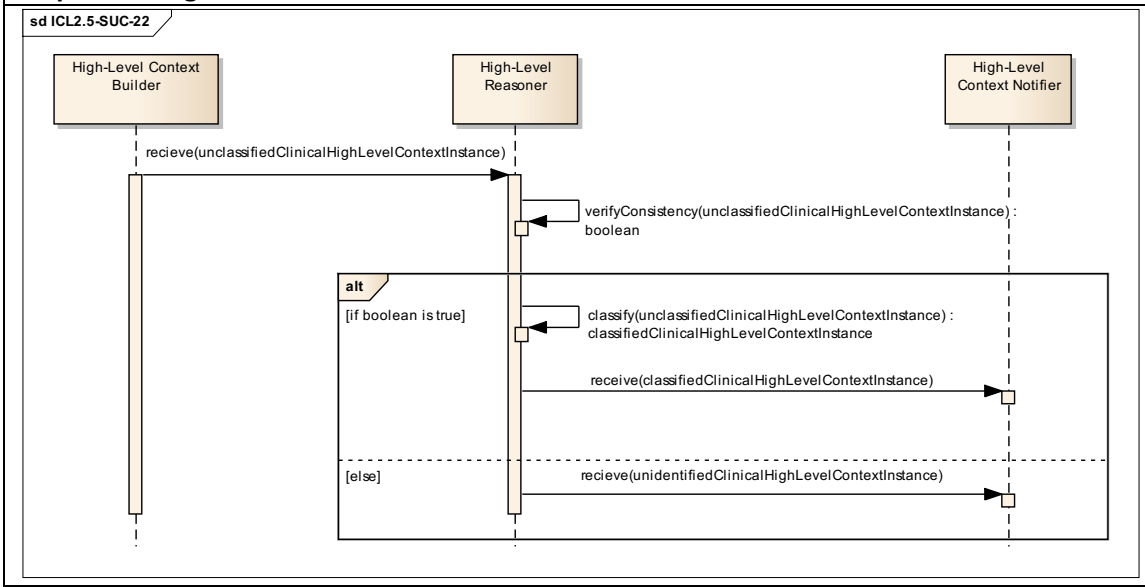
Use Case ID:	ICL-UC-20		
Use Case Name:	High level physical activity context		
Created By:	Wajahat Ali Khan	**Last Updated By:**	Wajahat Ali Khan
Date Created:	18 March 2016	**Last Revision Date:**	18 June 2018
Actors:	High-Level Context Reasoner		
Description:	A classified high level physical activity context is recognized by the reasoner from the unclassified high level context.		
Trigger:	High-level context is identified		
Pre-conditions:	• An unclassified high-level physical activity context instance is received		
Post-conditions:	• The new high-level physical activity context instance is forwarded to Notify DCL		
Normal Flow:	1. A high-level physical activity context instance is processed by high-level context classifier 2. Reasoning is performed by the reasoner to find out classified high level physical activity context 3. The classified high level physical activity context is provided to Notify DCL		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Less frequent: at every reception of a unclassified high-level physical activity context instance		
NFR ID:	NA		
Assumptions:	NA		
Notes and Issues:	NA		
Sequence Diagram:			



Use Case ID:	ICL4-SUC-21		
Use Case Name:	High level nutrition context		
Created By:	Wajahat Ali Khan	**Last Updated By:**	Wajahat Ali Khan
Date Created:	18 March 2016	**Last Revision Date:**	18 June 2018
Actors:	High-Level Context Reasoner		
Description:	A classified high-level nutrition context is recognized by the reasoner from the unclassified high-level context.		
Trigger:	High-level context is identified		
Pre-conditions:	• An unclassified high-level nutrition context instance is received		
Post-conditions:	• The new high-level nutrition context instance is forwarded to Notify DCL		
Normal Flow:	1. A high-level nutrition context instance is processed by high-level context classifier 2. Reasoning is performed by the reasoner to find out classified high level nutrition context 3. The classified high-level nutrition context is provided to Notify DCL		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Less frequent: at every reception of an unclassified high-level nutrition context instance		
NFR ID:	NA		
Assumptions:	NA		
Notes and Issues:	NA		
Sequence Diagram:			



Use Case ID:	ICL-UC-22		
Use Case Name:	High level clinical context		
Created By:	Muhammad Asif Razzaq	**Last Updated By:**	Muhammad Asif Razzaq
Date Created:	10 March 2017	**Last Revision Date:**	10 March 2017
Actors:	High-Level Context Reasoner		
Description:	A classified high level clinical context is recognized by the reasoner from the unclassified high-level context.		
Trigger:	High-level context is identified		
Pre-conditions:	• An unclassified high-level clinical context instance is received		
Post-conditions:	• The new high-level clinical context instance is forwarded to Notify DCL		
Normal Flow:	1. A high-level clinical context instance is processed by high-level context classifier 2. Reasoning is performed by the reasoner to find out classified high level clinical context 3. The classified high level clinical context is provided to Notify DCL		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Less frequent: at every reception of a unclassified high-level clinical context instance		
NFR ID:	NA		
Assumptions:	NA		
Notes and Issues:	NA		
Sequence Diagram:			



3. Knowledge Curation Layer

3.1 Function Requirements

Requirements #ID	Description
KCL-FR-01	The data-driven knowledge acquisition shall know schema detail of lifelog and user profile data in order to load the data and extract feature model.
KCL-FR-02	The data-driven knowledge acquisition shall load all related data specified as feature model from Data Curation Layer (DCL). The loaded lifelog and user profile data will be used for classification model creation.
KCL-FR-03	The expert-driven knowledge acquisition shall share the production rules with Service Curation Layer (SCL) for final reasoning.
KCL-FR-04	The expert-driven environment shall create "Situations" and share its configuration with SCL and DCL. Furthermore, all associated rules with "Situation" shall be provided to SCL.

3.2 Non-Function Requirements

Requirements #ID	Description
KCL-NFR-01	The layer shall persist only verified and validated rules into knowledge base.
KCL-NFR-02	The layer shall ensure consistency of distributed copies of knowledge base.

3.3 Terms and Definitions

Terms	Description
KCL	Knowledge Curation Layer
DCL	Data Curation Layer
SCL	Service Curation Layer
Situation	An abnormal status of a subject caused by unhealthy behaviors.
Production Rule	Production rule is ultimate and shareable rule which is used in reasoning to produce recommendation.
Domain Expert	Domain expert is an actor who will interact with system to create knowledge base.
Verification and Validation	Verification ensures that rule created is consistent with requirements and validation ensures that the rule created is correctly working on real data.
Schema	Schema represents the structure and associated semantics of user profile and life log data.

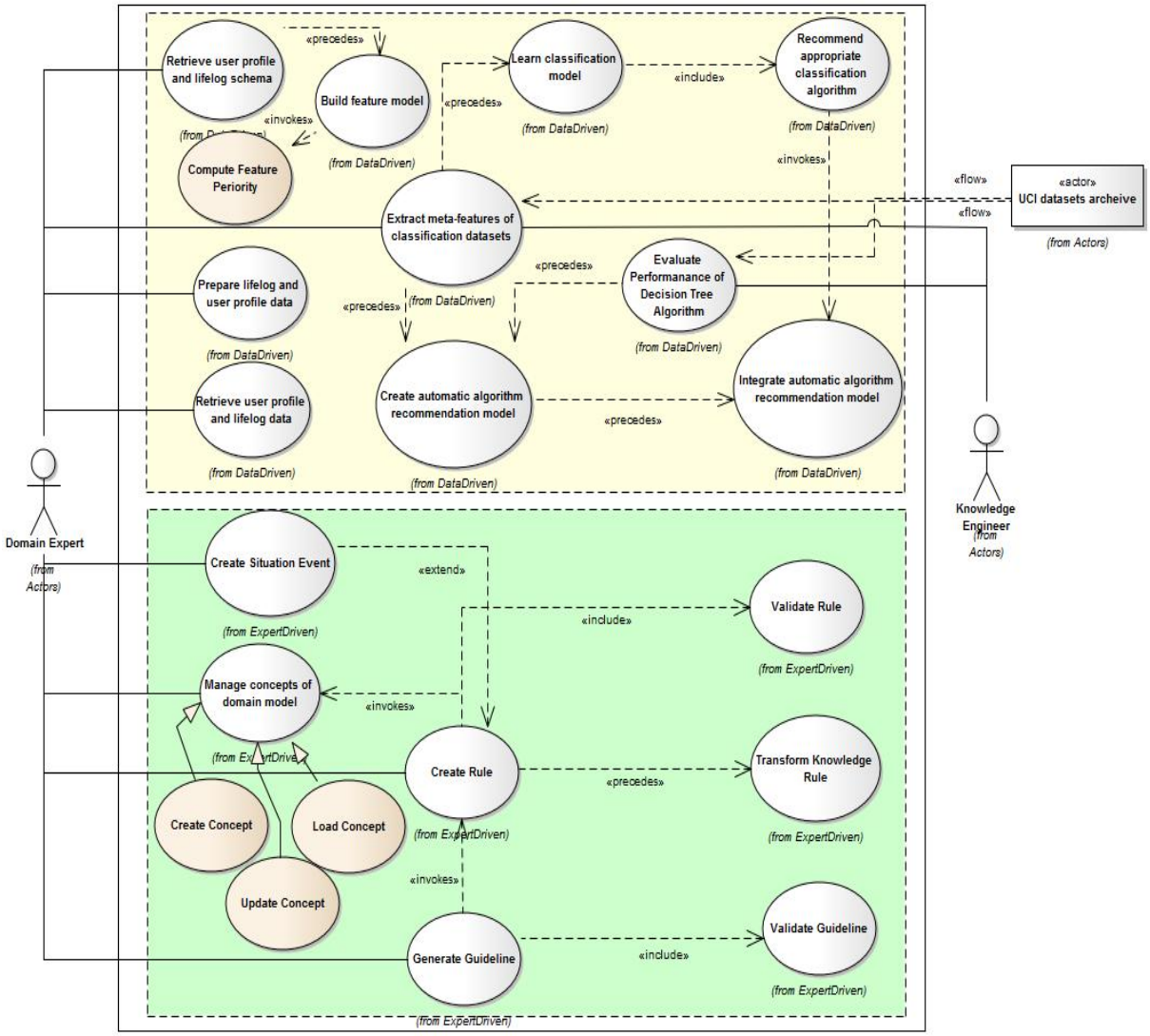
3.4 Use Case

3.4.1 List

Requirements #ID	Description
KCL-UC-01	Select valid combinations of features from lifelog and user profile schema to build feature model for yielding correct classification model.
KCL-UC-02	Apply preprocessing methods on retrieved lifelog and user profile data to prepare the data for classification model learning.

KCL-UC-03	Expert generate guidelines to utilize their practices to create rules in the knowledge bases.
KCL-UC-04	System validate the guidelines in tree structure to maintain the rules.
KCL-UC-05	User profile and lifelog schema is needed to be known before feature modeling and creation of classification model.
KCL-UC-06	Retrieve user profile and lifelog data for creation of classification model.
KCL-UC-07	Extract meta-features of classification datasets.
KCL-UC-08	Evaluate performance of decision tree algorithms (i.e, f-measure)
KCL-UC-09	Create automatic algorithm recommendation model (AARM) from offline datasets. AARM will be used as recommendation model for algorithm selection.
KCL-UC-10	Create rules to enhance the knowledge base of the system to generate recommendations in easy manner.
KCL-UC-11	Rule validation avoid the duplication of rules in the knowledge base and enhance the maintainability of knowledge base.
KCL-UC-12	It integrates AARM dataset in Mining Minds Data Driven knowledge acquisition approach for recommendation of automatic algorithm on given dataset.
KCL-UC-13	It generates classification model from user profile lifelog data that can be explored by model learning mechanism with the help of learning method as well as processed data.
KCL-UC-14	The integrated AARM shall automatically recommend appropriate classification algorithm. Or domain expert can select any algorithm from available set of decision tree algorithms.
KCL-UC-15	Domain model is used in creation of rule. It manages the domain model for creating rule.
KCL-UC-16	It transforms the rules or guidelines into executable knowledge representation.
KCL-UC-17	It creates situation event and index the rule based on situation event.
KCL-UC-18	Compute features priorities to help the domain expert for selecting appropriate features from available schema.

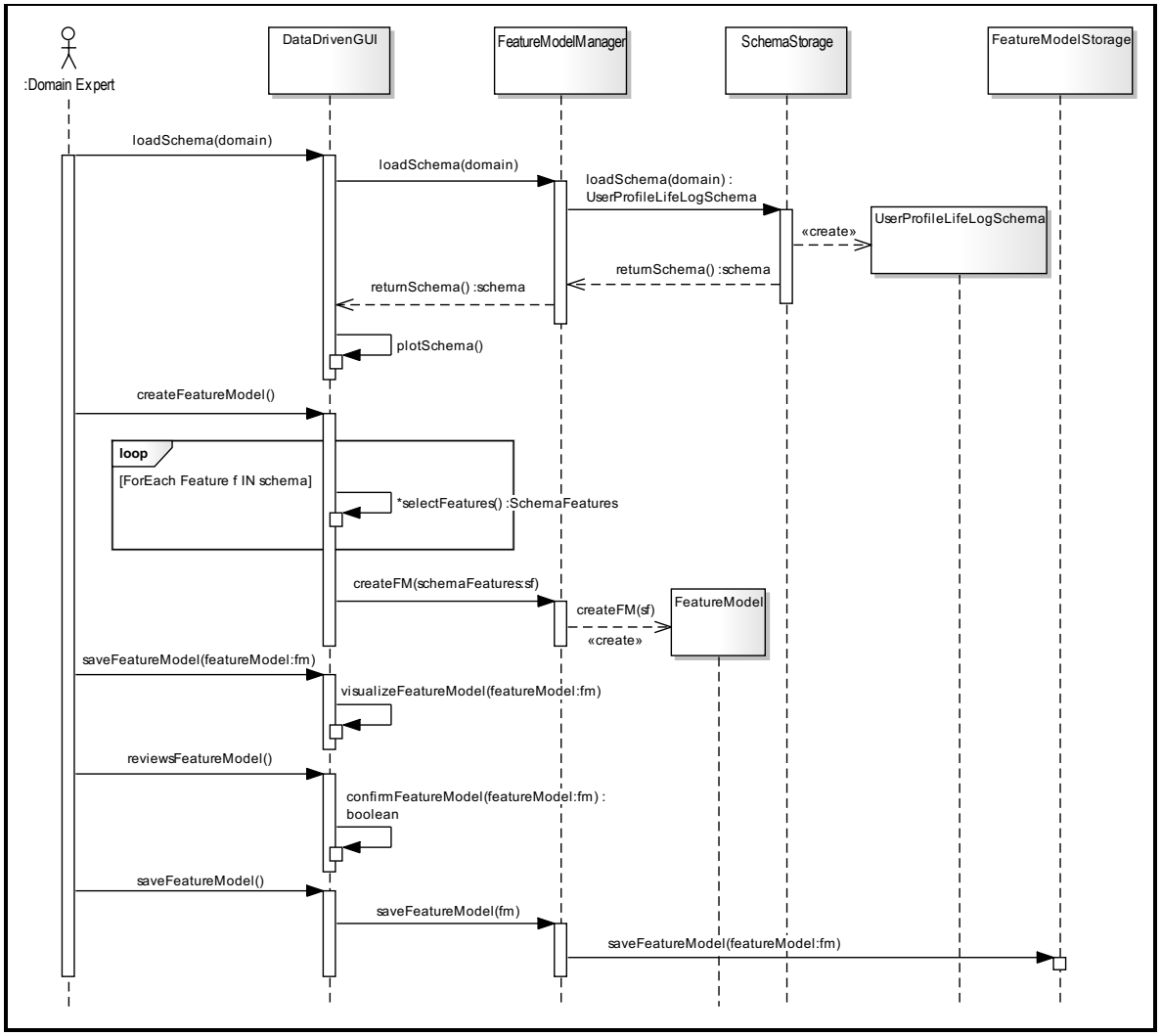
3.4.2 Diagram



3.4.3 Description

Use Case ID:	**KCL-UC-01**		
Use Case Name:	Build feature model		
Created By:	Maqbool Ali	**Last Updated By:**	Maqbool Ali
Date Created:	11-07-2015	**Last Revision Date:**	03-07-2018
Actors:	Domain Expert		
Description:	A feature model defines the valid combinations of features in a domain that enables capturing feature variability and interdependencies. For building feature model and its reusability,		

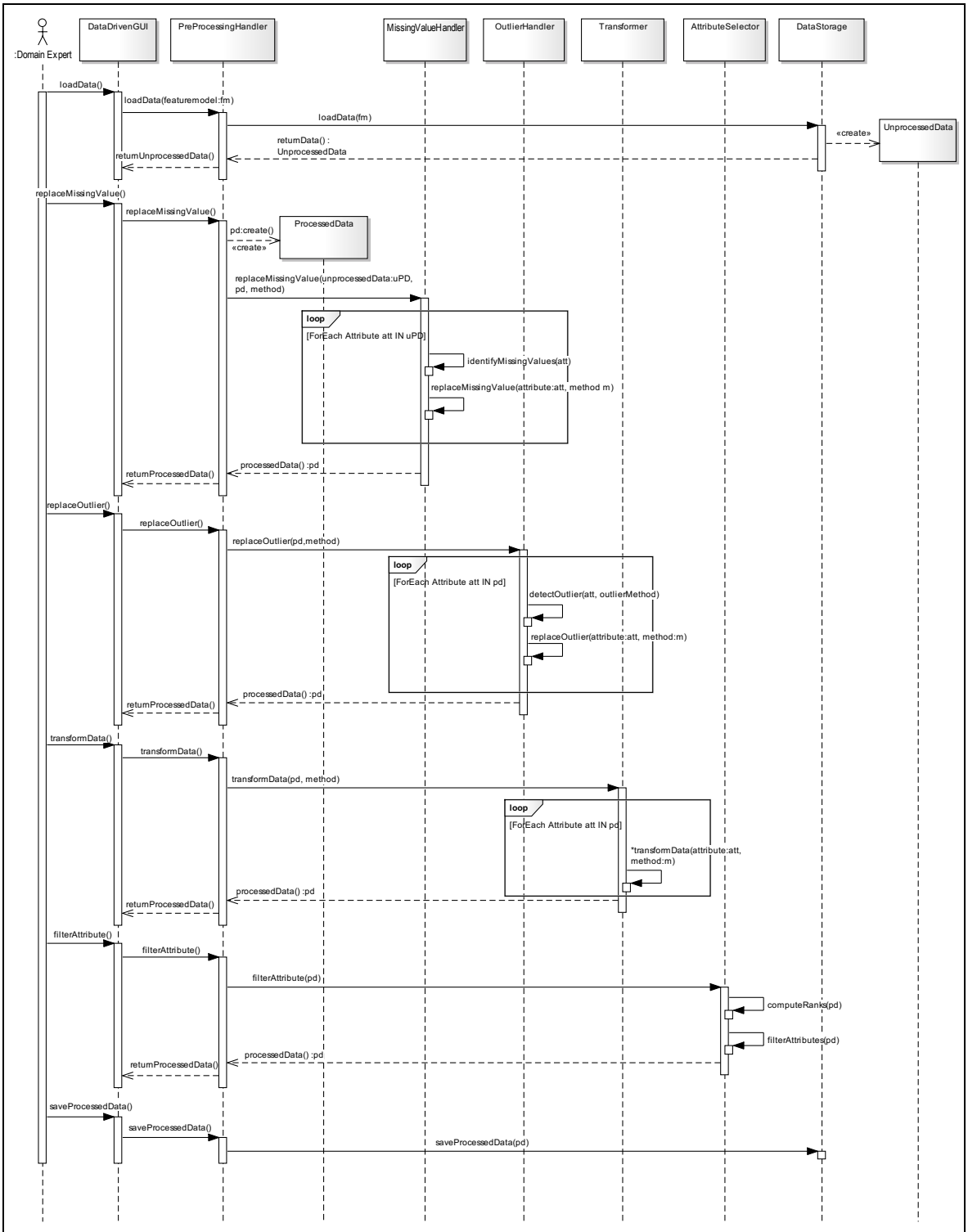
	domain expert uses selected domain schema (i.e. lifelog and user profile schema) and selects the related features for final feature model.
Trigger:	Prior to classification model creation needed for required domain
Preconditions:	• System has retrieved the schema from DCL. • Domain expert has selected domain under consideration (e.g. nutrition).
Postconditions:	System will build the feature model
Normal Flow:	1. Domain expert retrieves the schema from schema storage. 2. System loads and plots the schema 3. Domain expert builds the feature models as follows: a. Select the required features for corresponding domain b. Verify the consistency of the selected features (such as concept hierarchy) c. Save the feature model 4. System creates the feature model based on selected features and visualizes it in hierarchical form 5. Domain expert reviews the feature model and confirms it for saving into repository 6. System persists the feature model into repository.
Alternative Flows:	N/A
Exceptions:	
Includes:	N/A
Frequency of Use:	When new service is required and mining mind have sufficient data for classification model creation
Special Requirements:	N/A
Assumptions:	Initially we assume that feature model is valid
Notes and Issues:	One possible candidate representation for feature model is XML.
Sequence Diagram:	



Use Case ID:	**KCL-UC-02**		
Use Case Name:	Prepare lifelog and user profile data		
Created By:	Maqbool Ali	**Last Updated By:**	Maqbool Ali
Date Created:	10-07-2015	**Last Revision Date:**	03-07-2018
Actors:	Domain Expert		
Description:	It is important to pre-process the data (i.e. lifelog and user profile data) to generate models with high accuracy. '*Prepare lifelog and user profile data*' use case apply various pre-processing techniques such as missing value handling, outlier detection, transformation, and features selections to convert unprocessed data into processed data.		

Trigger:	Prior to classification model creation needed for high accuracy of model learning
Preconditions:	• System has retrieved the data from DCL, which is unprocessed data.
Postconditions:	System will prepare and store the data
Normal Flow:	1. Domain expert loads the unprocessed data 2. System displays the retrieved data 3. For each attribute: a. Domain expert identifies the missing values and select appropriate method from following options for missing value replacement. ▪ Default value ▪ Mean ▪ Mode b. System replaces the missing values using selected method. 4. For each attribute: a. Domain expert apply outlier detection method such as interquartile range and scatterplot. b. System display the outliers c. Domain expert select appropriate method from following options for outlier replacement. ▪ Mean ▪ Mode d. System replace the outlier using selected method. 5. For each attribute: a. Domain expert identifies, normalizes the non-transformed values, and updates the dataset. b. System modifies the values set and update the dataset 6. Domain expert applies the attributes filtration techniques (i.e ranking) 7. System computes the ranks for all attributes and displays to expert 8. Domain expert select the highly ranked attributes (i.e. rank value >= 0.8) 9. System filters the attributes based on selected attributes and displays to domain expert 10. Domain expert saves the processed data into repository 11. System persists the processed data into repository
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A

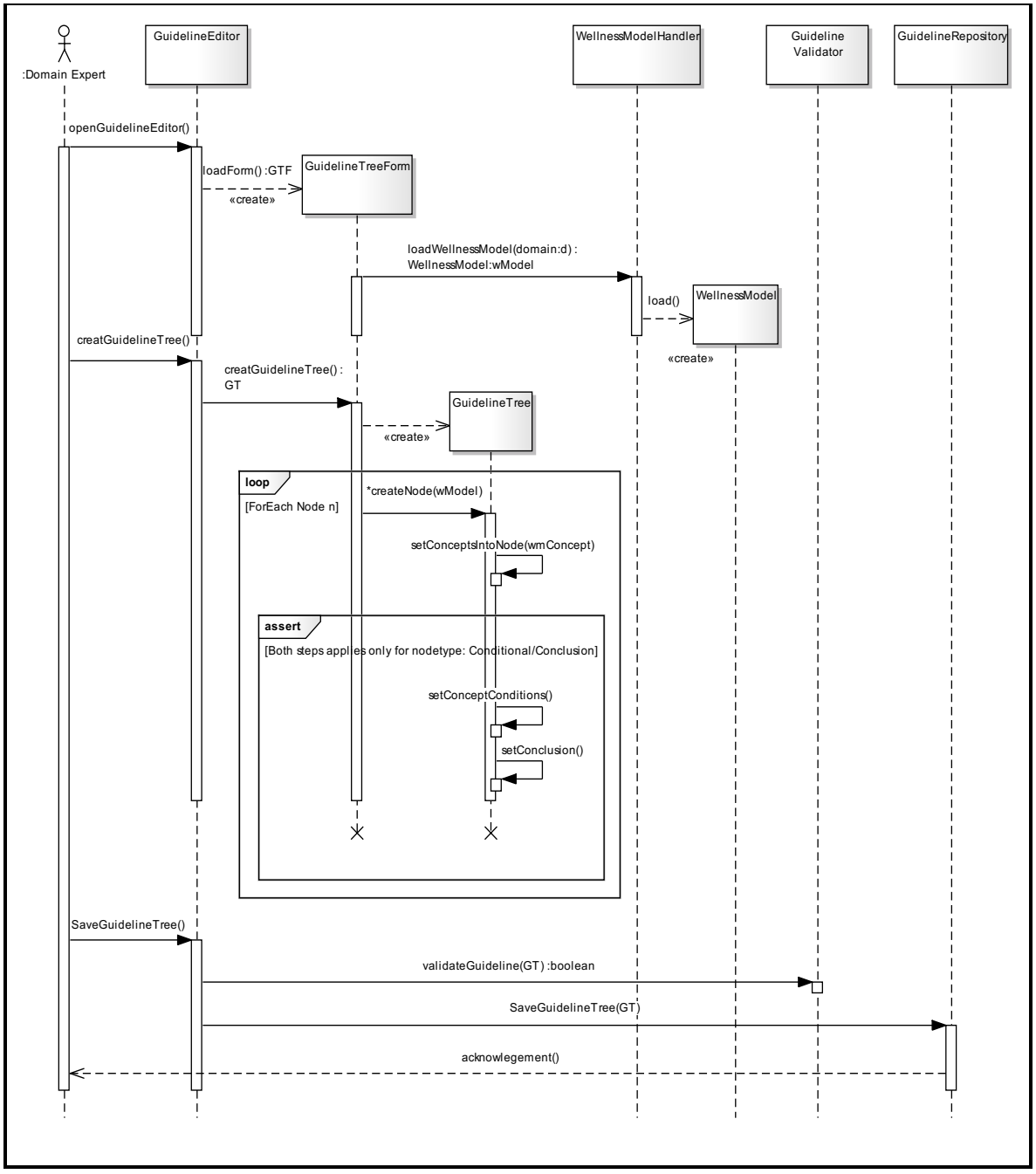
Frequency of Use:	When new service is required, and mining mind have sufficient data for classification model creation
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	• An **outlier** is any value that is numerically distant from most of the other data points in a set of data. It can be detected by histograms, scatterplots, or interquartile range techniques. • **Data transformation** is the process to convert and normalize the data from one format to another. It can be done by Log, square root, or arcsine transformation techniques.
Sequence Diagram:	



Use Case ID:	**KCL-UC-03**
Use Case Name:	Generate Guideline

Created By:	Taqdir Ali and Maqbool Hussain	**Last Updated By:**	Taqdir Ali, Maqbool Hussain
Date Created:	11-07-2015	**Last Revision Date:**	15-07-2015
Actors:	Domain Expert		
Description:	Guidelines are the combination of one or multiple rules in form of decision tree. The tree format guidelines are understandable to the domain experts and it can easily interpret and execute by computer.		
Trigger:	Whenever domain expert wants to generate new guideline or update the existing one.		
Preconditions:	1. The domain expert shall be authenticated with full access on the guideline management. 2. Domain expert shall have existing guidelines as reference for generating guideline tree.		
Postconditions:	• The expert shall generate guidelines to acquire their knowledge into the system.		
Normal Flow:	1. Domain expert opens the guidelines editor. 2. System displays new guideline tree form and load the wellness model. 3. Domain expert selects/drags tree node into editor form. 4. System display the node and open the corresponding properties window, which includes; a. Loads wellness model tree. b. Displays operators, relationships and node type (conditional, conclusion or both) artifacts. 5. Domain expert selects concepts for the node using any of the following methods and confirm to save the node. a. Using wellness model, dragging concepts and facts into node conditional or conclusion part. b. Using auto pop-up Intelli-sense window to select concepts and facts into conditional or conclusion part. 6. System saves the tree node and displays as part of the guideline tree. 7. Domain expert add other nodes to guideline tree by using step repeating step 3 on ward. After completion, (s)he saves the guideline tree. 8. System validates the guideline tree using “KCL-UC-04” and save into guideline repository.		
Alternative Flows:	2a. System loads existing guideline tree for modification (modifying existing or adding new node). a. Domain expert selects existing node in guideline tree or drag new node to appropriate place in guideline tree. b. To modify node, step 3 onward will be invoked in Normal Flow.		
Exceptions:	N/A		

Includes:	Validate Guideline
Frequency of Use:	Whenever domain expert wants to create new guideline tree or update existing guideline tree.
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	• Guidelines tree created will base on existing guidelines of corresponding domain and domain expert shall interpret textual guidelines into tree format. • Appropriate modelling of guideline is challenging task.
Sequence Diagram:	

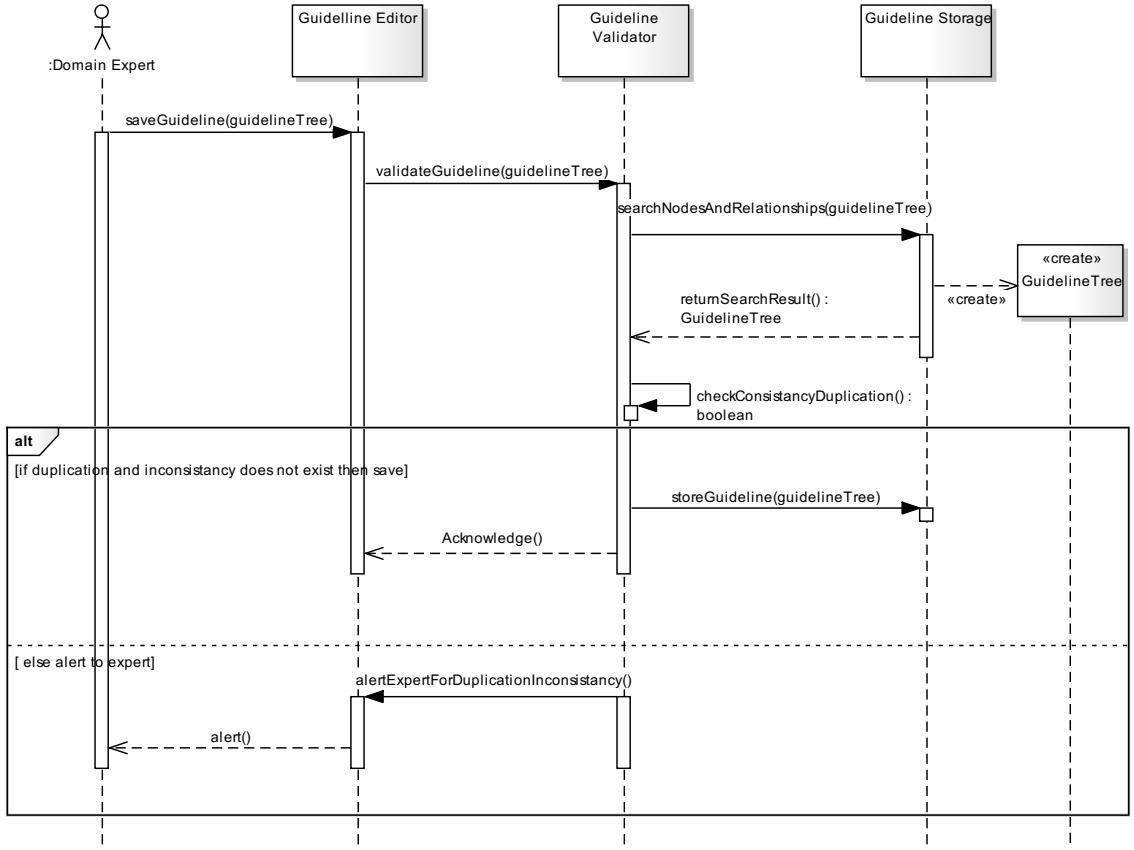


Use Case ID:	KCL-UC-04		
Use Case Name:	Validate Guideline		
Created By:	Taqdir Ali and Maqbool Hussain	Last Updated By:	Taqdir Ali, Maqbool Hussain

Date Created:	11-07-2015	**Last Revision Date:**	15-07-2015
Actors:	Domain Expert		
Description:	Guidelines have different facts and conclusions in form of nodes related with different relationships. The guidelines tree shall be validated for the possible duplication.		
Trigger:	Whenever domain expert wants to generate new guideline or update the existing one.		
Preconditions:	1. System shall be running 2. The domain expert shall be authenticated with full access on the guideline management.		
Postconditions:	• Validated guideline tree		
Normal Flow:	1. Domain Expert save the new guideline or update the existing guideline. 2. The system validate guideline for inconsistency and duplication as follows. a. Fetch the existing guidelines and process each node and relationship b. Guideline Tree is approved for having no inconsistency and duplication of new nodes and relationships of the facts and conclusion with the existing guidelines. c. Guideline Tree is stored into guidelines repository. d. Acknowledge the expert to save guideline successfully.		
Alternative Flows:	2b. Guideline Tree is found having inconsistency or duplication with existing guideline tree a. The system produces alert the inconsistency or duplication in guideline tree b. Domain expert review the alert message and correct the guideline tree. c. Step 1 and Step 2 of normal flow is executed.		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Whenever domain expert wants to create new guideline or update existing guideline.		
Special Requirements:	N/A		
Assumptions:	N/A		

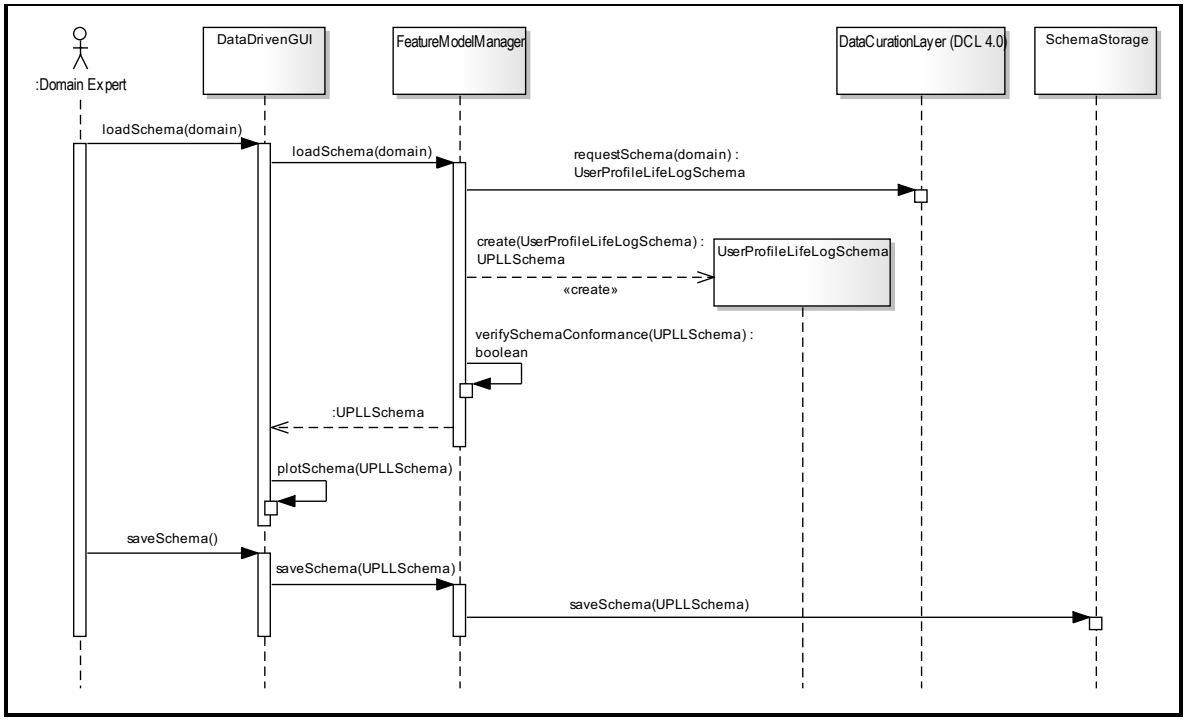
Notes and Issues:	Appropriate modelling of guideline and validation is challenging task.

Sequence Diagram:



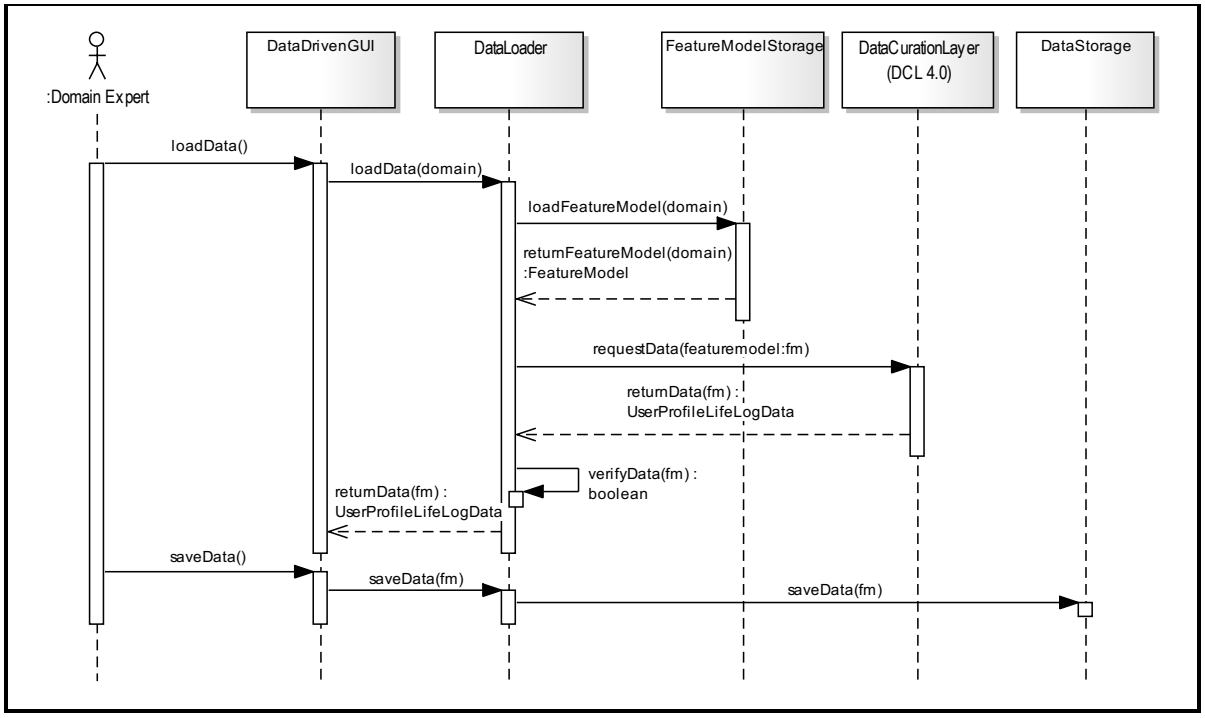
Use Case ID:	**KCL-UC-05**		
Use Case Name:	Retrieve user profile and lifelog schema		
Created By:	Maqbool Ali	**Last Updated By:**	Maqbool Ali
Date Created:	10-07-2015	**Last Revision Date:**	03-07-2018
Actors:	Domain Expert, DCL		
Description:	User profile and lifelog schema retrieval help domain expert to view all available features for building feature model.		
Trigger:	Prior to classification model creation needed for required domain		
Preconditions:	• System has access through service interface to retrieve user profile and lifelog schema from DCL • System and DCL has agreement on common schema representation format		

	• DCL has capability to share user profile and lifelog schema in secure environment.
Postconditions:	System will receive user profile and lifelog schema conform to its representation scheme.
Normal Flow:	1. Domain expert selects the domain and sends requests to DCL for user profile and lifelog schema. 2. DCL shares the user profile and lifelog schema 3. System receives the user profile and lifelog schema 4. Domain expert uses the system and performs the following tasks; a. Verifies the conformance of received schema b. Plots the verified schema c. Saves the verified schema 5. System saves the verified schema
Alternative Flows:	N/A
Exceptions:	1a. System unable to connect to DCL a. System connection is failed during retrieving user profile and lifelog schema b. System hold and will retry after some time to connect to DCL and retrieve the user profile and lifelog schema 4a. System unable to verify lifelog schema conformance a. System fail to conform the schema representation from DCL b. System will send message to DCL about incompatible schema format
Includes:	N/A
Frequency of Use:	When new service is required, and mining mind have sufficient data for classification model creation
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	If DCL is unable to send user profile and lifelog schema in required format, then alternate strategy has to be considered.
Sequence Diagram:	



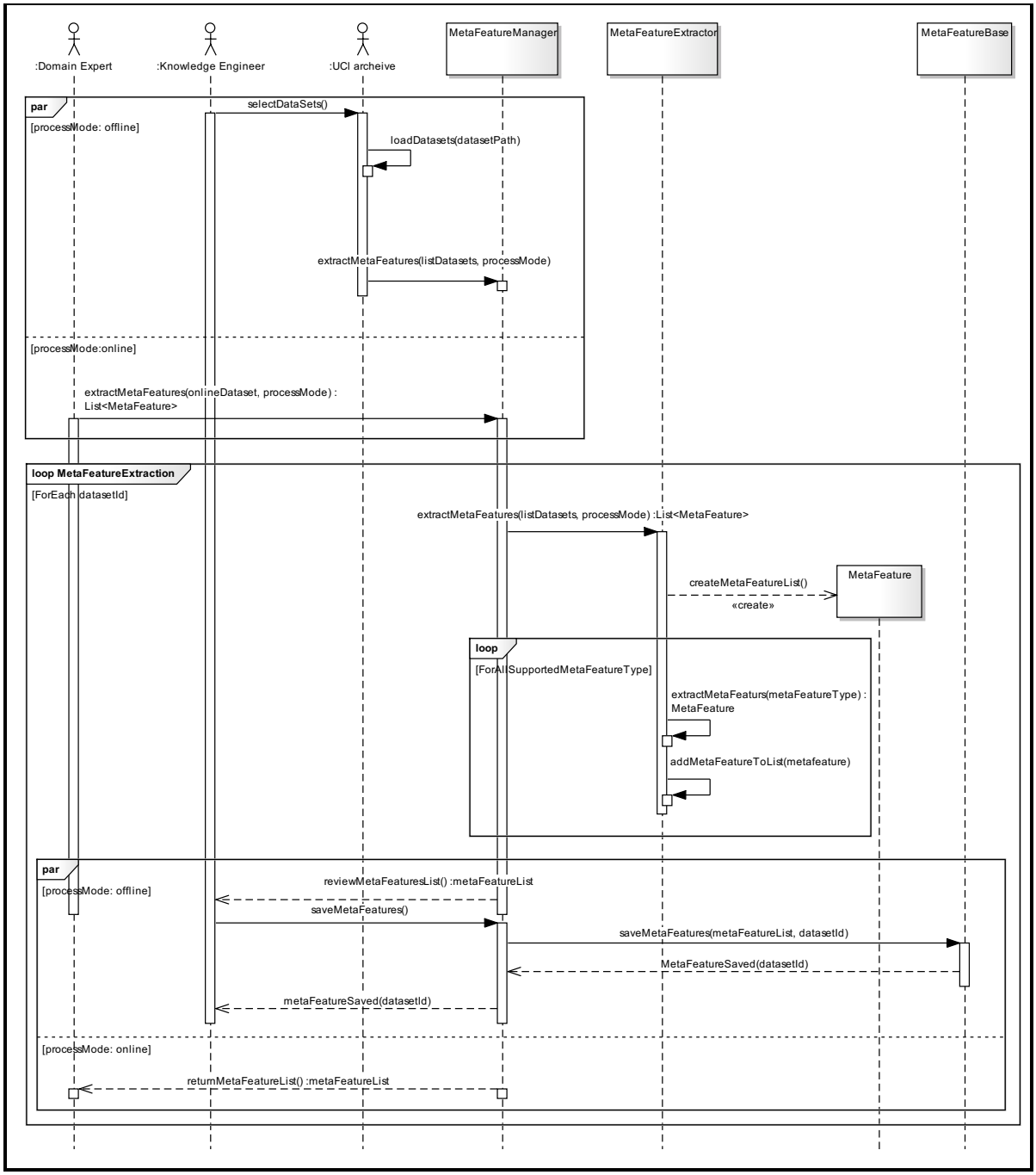
Use Case ID:	**KCL-UC-06**		
Use Case Name:	Retrieve user profile and lifelog data		
Created By:	Maqbool Ali	**Last Updated By:**	Maqbool Ali
Date Created:	10-07-2015	**Last Revision Date:**	03-07-2018
Actors:	Domain Expert, DCL		
Description:	User profile and lifelog data has hidden knowledge that can be explored after loading from data storage		
Trigger:	Prior to classification model creation needed for required data		
Preconditions:	• System has access through service interface to retrieve user profile and lifelog data from DCL • DCL has capability to share user profile and lifelog data in secure environment. • System has already loaded the previously imported user profile and lifelog schema		
Postconditions:	System will receive user profile and lifelog data based on selected schema		
Normal Flow:	1. Domain expert loads the feature model for selected domain 2. System loads the corresponding feature model		

	3. Domain expert sends request to DCL for user profile and lifelog data based on loaded feature model 4. DCL shares the user profile and lifelog data 5. System receives the user profile and lifelog data 6. Domain expert uses the system and performs the following tasks; o Verifies the user profile and lifelog data o Saves the data after verification. 7. System saves the verified data
Alternative Flows:	
Exceptions:	3a. System unable to connect to DCL a. System connection is failed during retrieving user profile and lifelog data b. System hold and will retry after some time to connect to DCL and retrieve the user profile and lifelog data 6a. System receives irrelevant data a. System detects the irrelevant data sent by DCL. b. System request again DCL to make sure that data received is according to feature selected.
Includes:	
Frequency of Use:	When new service is required and mining mind have sufficient data for classification model creation
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	If DCL is unable to send data based on dynamic feature selection from schema, then alternate strategy has to be considered.
Sequence Diagram:	



Use Case ID:	**KCL-UC-07**		
Use Case Name:	Extract meta-features of classification datasets		
Created By:	Rahman Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	16-07-2015	**Last Revision Date:**	20-07-2015
Actors:	Knowledge Engineer/Domain Expert, UCI archive		
Description:	Datasets have simple, statistical, information theory and landmarking meta-features that can best describe nature of a dataset. These features can best used for building an algorithm selection model.		
Trigger:	In the offline process, when the algorithm selection model is build, and in the online process, when an appropriate algorithm is needed to be identified for a new query dataset		
Preconditions:	UCI archive datasets are available and are in refined format		
Postconditions:	The meta-features are ready for being used in building algorithm selection model.		
Normal Flow:	1. Knowledge Engineer selects one dataset from UCI archive. 2. System retrieves selected dataset.		

	3. Knowledge Engineer provides dataset to meta-feature extractor for extracting meta-features. 4. System extracts following meta-features set for selected dataset. a. basic meta-features b. statistical meta-features c. information theory meta-features d. extract landmark features 5. Knowledge Engineer reviews the extracted meta-features and saves it into meta-features base (MFB). 6. System saves meta-features into a MFB. 7. Knowledge Engineer repeats step 1-6 for each intended dataset.
Alternative Flows:	1a. Meta-feature extraction for online dataset a. Domain Expert provides new dataset used for classification model creation. b. Step 3-4 of Normal Flow is executed for Domain expert interactions with system.
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Frequently, whenever a new dataset arrives as a query dataset.
Special Requirements:	Minimum availability of classification datasets > 60 for a reasonable accuracy
Assumptions:	• The archived datasets are available and are in refined .raff format • The meta-feature space is defined in advance
Notes and Issues:	Meta-feature extraction is time consuming task for offline process as we have to take into account more than 60 datasets.
Sequence Diagram:	

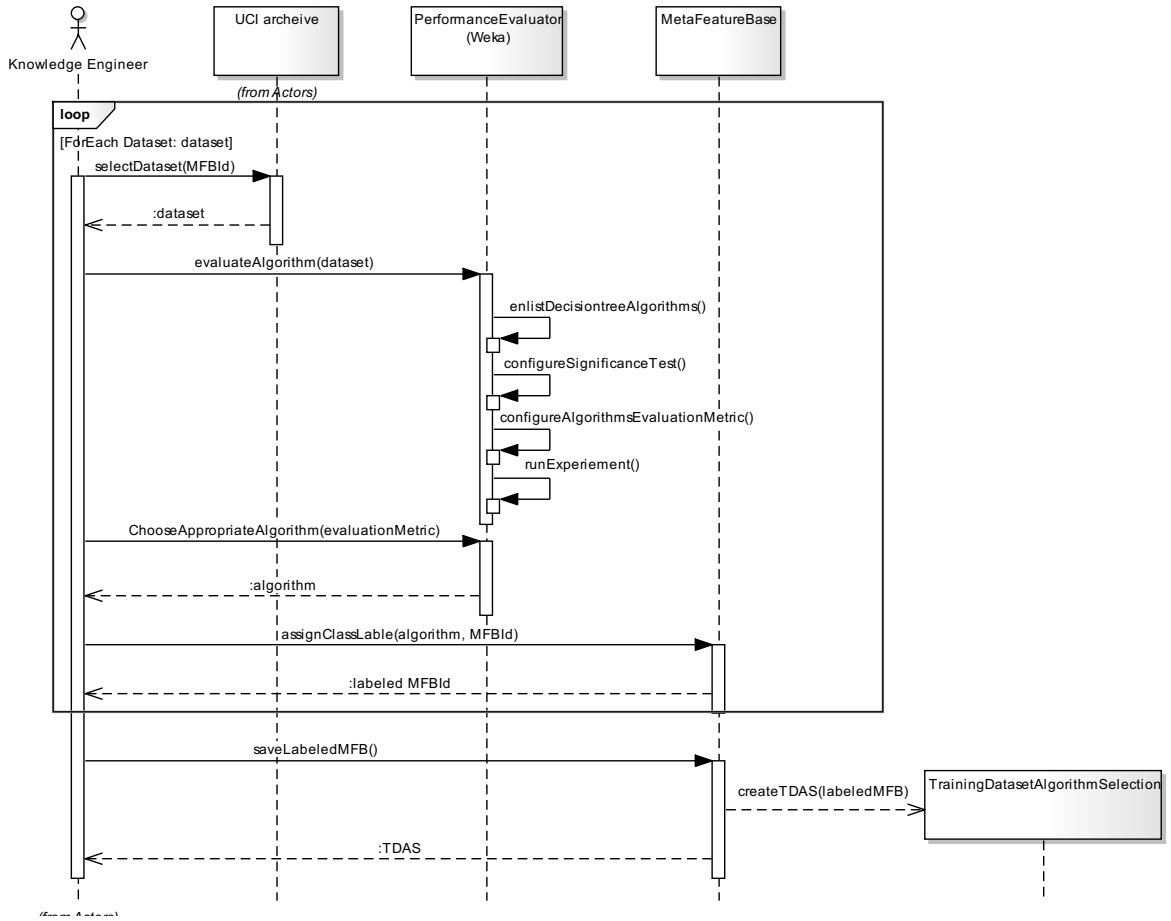


Use Case ID:	KCL-UC-08		
Use Case Name:	Evaluate performance of decision tree algorithm		
Created By:	Rahman Ali	Last Updated By:	Maqbool Hussain
Date Created:	16-07-2015	Last Revision Date:	20-07-2015

Actors:	Knowledge Engineer, UCI archive
Description:	Different algorithms have different performance score for the same dataset. To build an algorithm selection model, performance score of each algorithm needs to evaluate for choosing an appropriate one.
Trigger:	In the offline process, when the algorithm selection model is to build the first time.
Preconditions:	• UCI archive datasets are available and are in .arff format • The algorithm to be considered is specified in advanced (Decision Tree algorithms implemented in Weka) • The evaluation metric is specified (F-measure)
Postconditions:	All datasets records in Meta-Feature Base (MFB) will be assigned with optimal decision tree algorithm class label.
Normal Flow:	1. Knowledge Engineer selects UCI archive dataset, mentioned in MFB, for finding optimal decision tree algorithm. 2. System (Weka) loads selected datasets. 3. Knowledge Engineer setups experiment; a. Enlists all the decision tree algorithms available in system (Weka) b. Configure significance test (alpha=0.5) c. Configure algorithms comparison metric (f-measure) 4. System runs experiment and produces detailed f-score for all selected algorithms. 5. Knowledge Engineer performs following tasks; a. Records evaluation matrix. b. Chooses algorithm with the highest f-score. c. Assigns chosen algorithm as class label in MFB. d. Step 1-5 are repeated for other non-labeled datasets in MFB. e. After finishing labeling all records in MFB, saves the updated MFB as training dataset for algorithm selection (TDAS). 6. System saves the updated records in MFB as final TDAS.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	• Rarely, once enough new datasets are added to the system
Special Requirements:	N/A
Assumptions:	• The decision tree-based algorithms are implemented in Weka.

Notes and Issues:	Evaluating performance of decision tree algorithms over a large number (min 60) classification dataset is a computationally complex task. For minimum 60 datasets and at least 5 decision tree algorithms, a minimum of 300 experiments are required. Moreover, 60 additional significance test experiments are needed. On average, each experiment takes times in minutes ranging from 2 minutes to 30 minutes, depending on the complexity of the dataset.

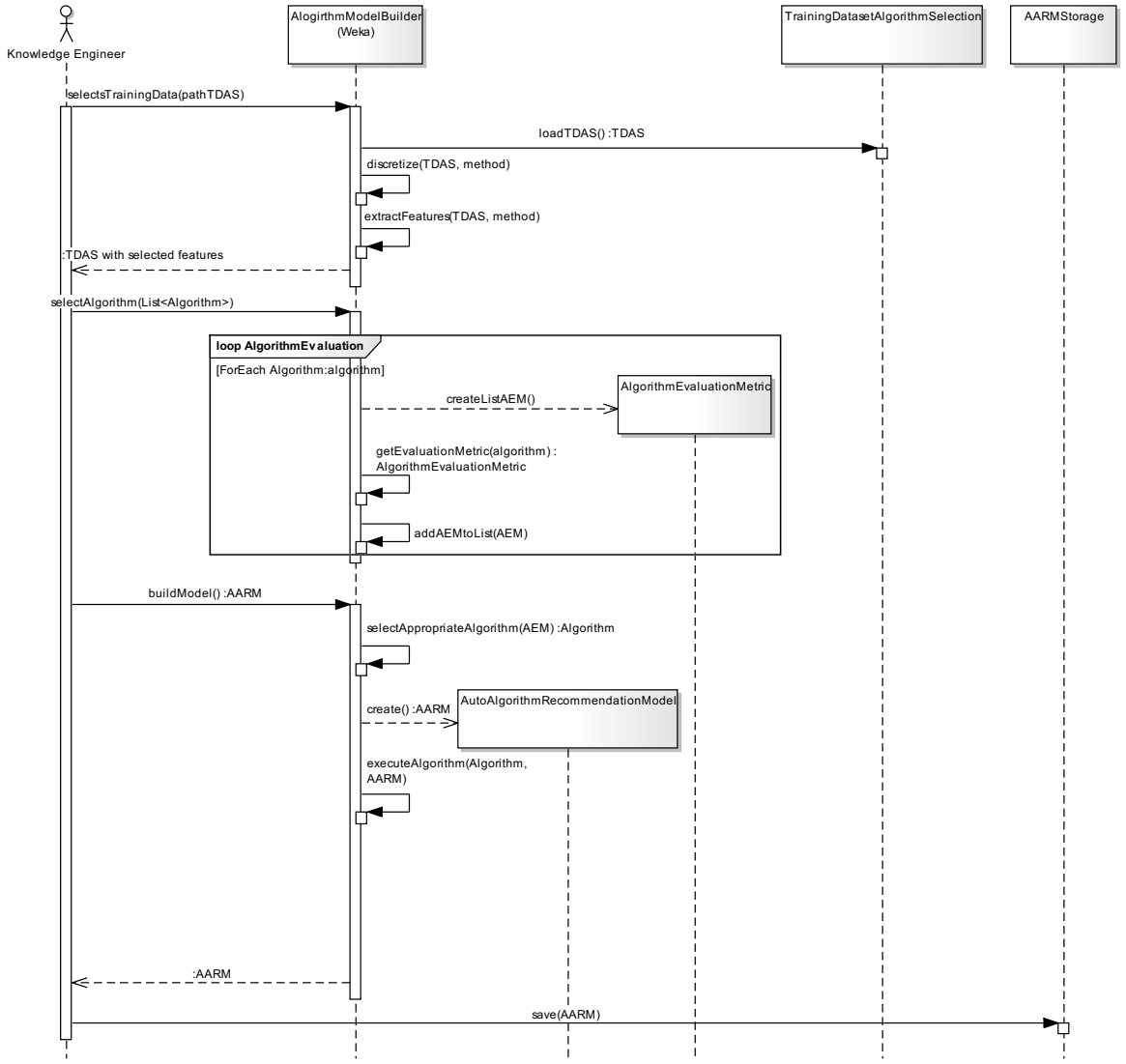
Sequence Diagram:



Use Case ID:	**KCL-UC-09**		
Use Case Name:	Create automatic algorithm recommendation model		
Created By:	Rahman Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	10-07-2015	**Last Revision Date:**	22-07-2015
Actors:	Knowledge Engineer/Domain Expert		
Description:	An automatic algorithm selection model enables knowledge engineer to automatically select appropriate algorithm for building classification model for his new dataset		
Trigger:	When the training dataset comprising datasets meta-features and algorithms performance evaluation are made available.		
Preconditions:	The datasets meta-features and algorithms performance training datasets is made available.		
Postconditions:	The automatic algorithm recommendation model (AARM) is ready to integrate in Mining Minds for real time algorithm selection.		
Normal Flow:	1. Knowledge Engineer selects TDAS. 2. System (Weka) loads TDAS. 3. Knowledge Engineer performs preprocessing of the TDAS (i.e., discretization, and features selection). 4. System refines the datasets with appropriate features. 5. Knowledge Engineer select an algorithm from decision tree (DT) or Rules learner (RL) for building AARM. 6. System executes selected algorithm on TDAS and produces evaluation matrix. 7. Knowledge Engineer records performs following tasks; a. Records evaluation matrix and repeats step 5-6 till all algorithms finished. b. Knowledge Engineer selects the appropriate algorithm. 8. System builds AARM model using the selected algorithm. 9. Knowledge Engineer saves AARM. 10. System saves the AARM into AARM storage.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Rarely, when TDAS is updated with new datasets or algorithms		
Special Requirements:	Availability of records > 60 for minimum acceptable accuracy		
Assumptions:	N/A		

Notes and Issues:	N/A

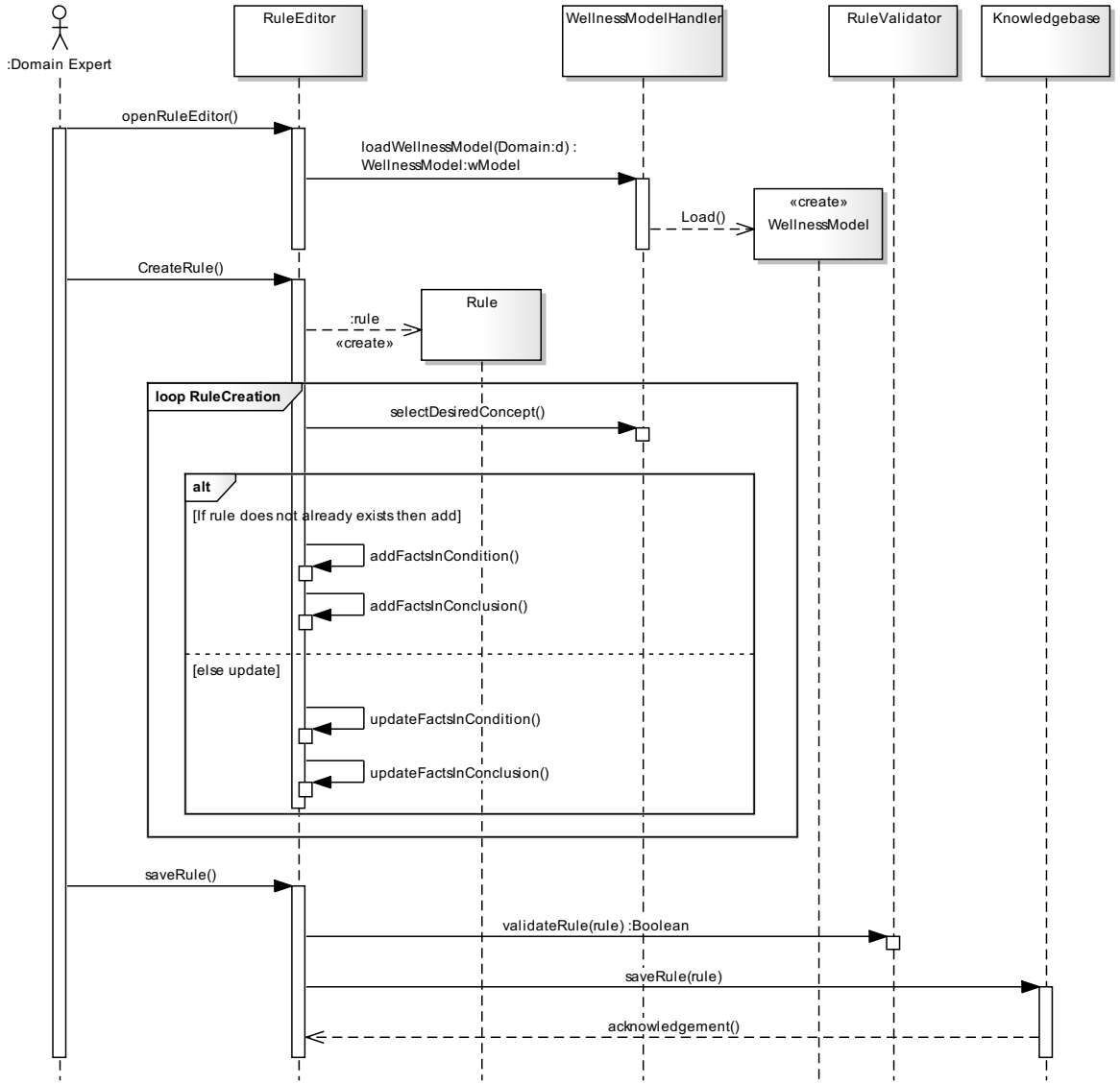
Sequence Diagram:



Use Case ID:	**KCL-UC-10**		
Use Case Name:	Create Rule		
Created By:	Taqdir Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	11-07-2015	**Last Revision Date:**	29-07-2015
Actors:	Domain Expert		
Description:	Knowledge bases need to enhance with up-to-date knowledge for correct recommendation. The expert shall create rules in the knowledge base to transform their practices into executable knowledge in form of rules.		
Trigger:	Domain experts trigger it for rule creation/updating when needed.		
Preconditions:	• The domain expert shall be authenticated with full access of rule management in the knowledge base.		
Postconditions:	The correct rule shall be saved into the knowledge base.		
Normal Flow:	1. Domain expert opens the rule editor. 2. System loads wellness domain model in form of concepts tree. 3. Domain expert selects concepts for the rule conditions and conclusion using any of the following methods; o Using wellness model, dragging concepts and facts into conditional or conclusion part of the rule editor. o Using auto pop-up Intelli-sense window to select concepts and facts into conditional or conclusion part of the rule editor. 4. System checks the existing rules to add/update the rule o Add new facts of the rule in condition. o Add new conclusion according to rule facts. 5. Step 3-4 will be repeated for each new/updated concept added to rule, and domain expert finally saves the rule. 6. System save rule as follows; o The system validates the rule using “KCL-UC-11”. o The system stores the validated rule into the knowledge base		
Alternative Flows:	6a. System founds the rule is already exists in rule repository o Domain expert review the existing facts and conclusion. o Step 5-6 will be followed to change the rule.		
Exceptions:	N/A		
Includes:	Validate Rule		
Frequency of Use:	Whenever domain expert wants to add rule or edit the existing rule.		
Special Requirements:	N/A		

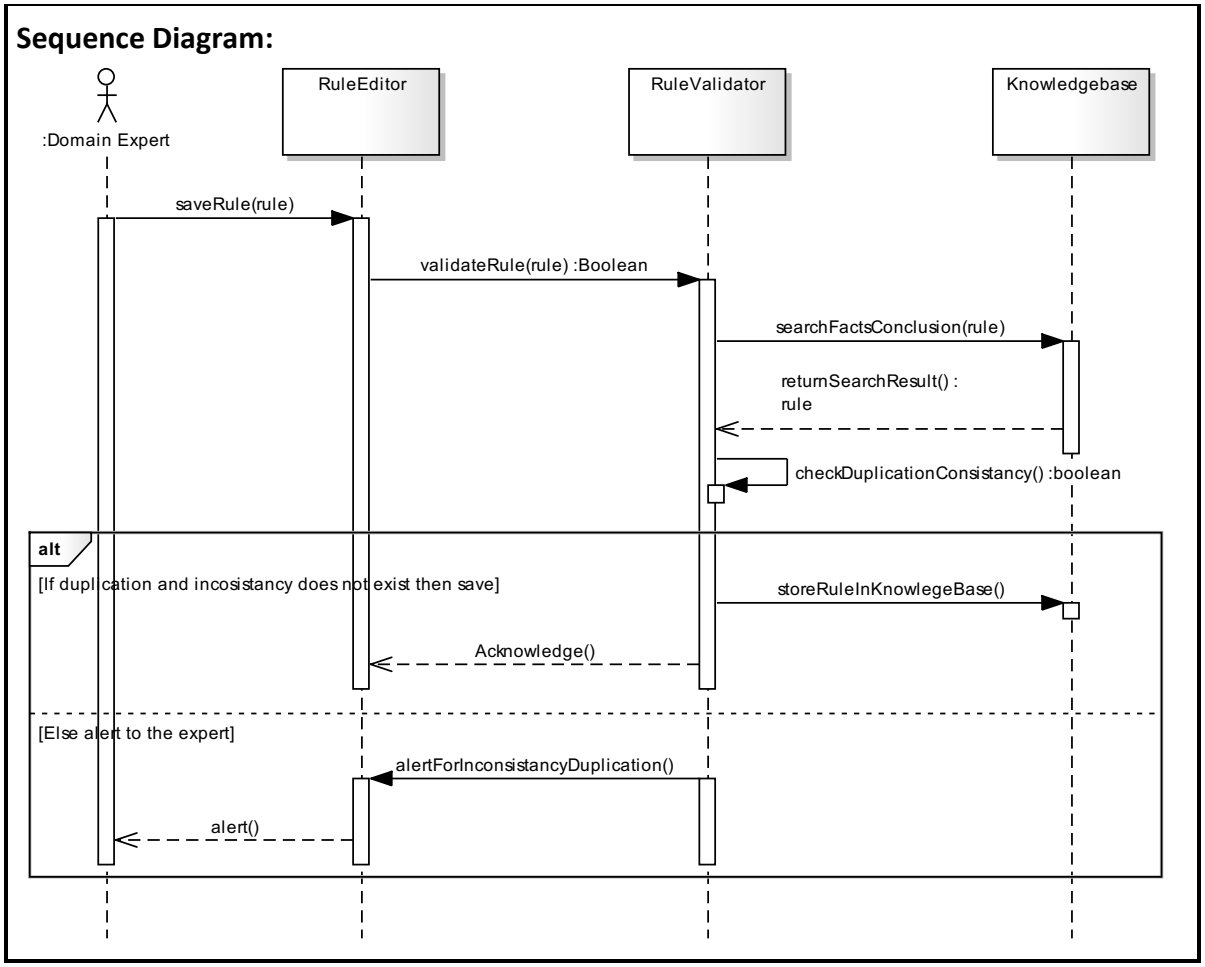
Assumptions:	N/A
Notes and Issues:	• If knowledge base does not exist in system, the administrator shall build the knowledge base first and configure with system. • After investigation, we may use unify representation for rules and guidelines.

Sequence Diagram:



Use Case ID:	**KCL-UC-11**		
Use Case Name:	Validate Rule		
Created By:	Taqdir Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	11-07-2015	**Last Revision Date:**	29-07-2015
Actors:	Domain Expert		
Description:	In new rules creation and editing existing rules, duplication and inconsistency may occur. The validation is needed to validate and find the duplication and inconsistency among the rules.		
Trigger:	• When new rule is going to create. • When existing rule is going to update		
Preconditions:	The rule creation and editing process completed by physician successfully.		
Postconditions:	The validated rule shall be saved into knowledge base.		
Normal Flow:	1. Domain expert saves the created rule. 2. System validate rule for inconsistency and duplication as follows a. Fetch the facts and conclusion of existing rules. b. The new or updated rule approved for having no inconsistency and duplication. c. Created rule stores into the rules repository. d. Acknowledge the expert to save the rule successfully.		
Alternative Flows:	2b. The created rule is found having inconsistency or duplication with existing rules in the rules repository a. The system produces alert the inconsistency or duplication of the rule with existing rules in repository. b. Domain expert review the alert message and correct the created rule. c. Step 1 and Step 2 of normal flow is executed.		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Whenever domain expert want to save the rule		
Special Requirements:	N/A		
Assumptions:	N/A		
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.		

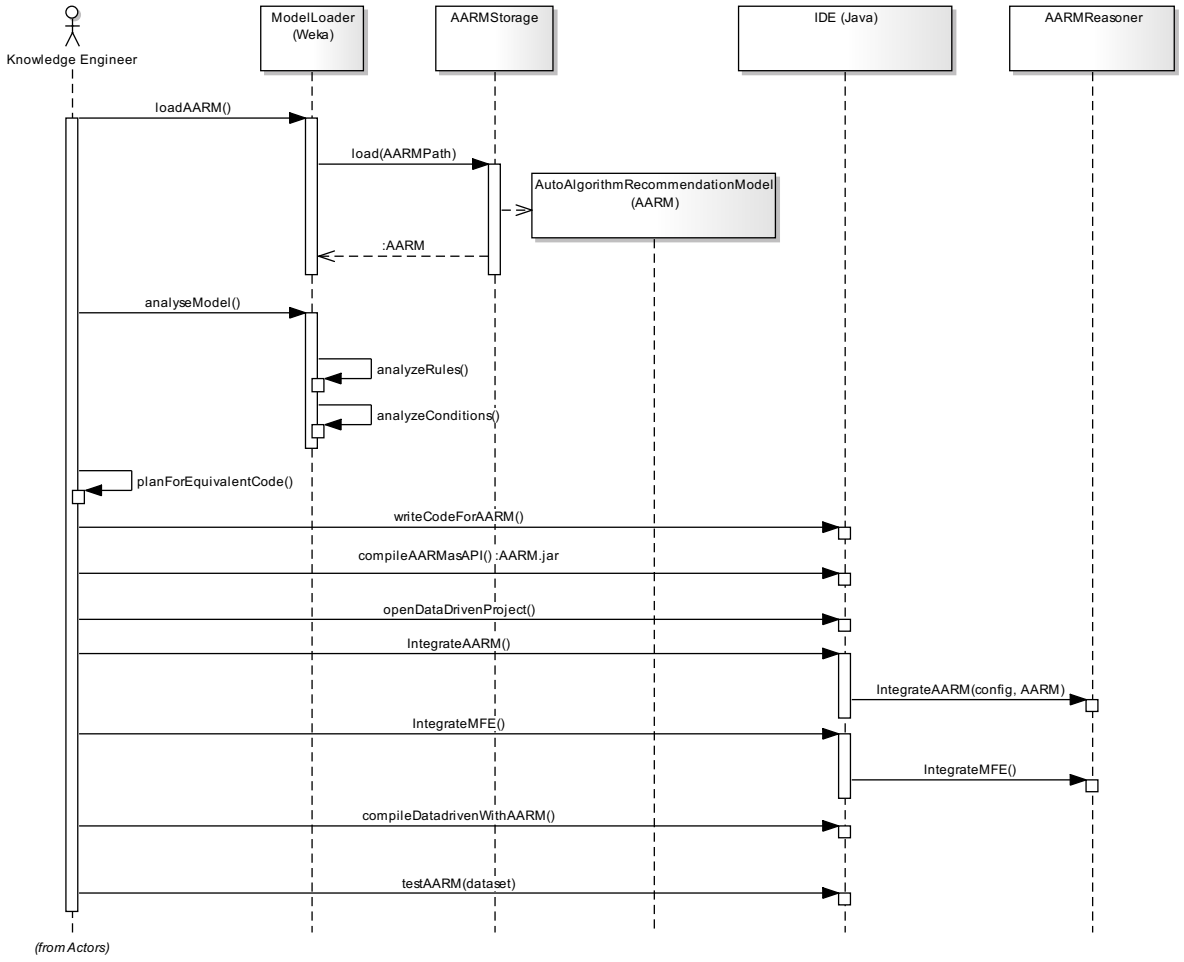
Sequence Diagram:



Use Case ID:	**KCL-UC-12**		
Use Case Name:	Integrate automatic algorithm recommendation model		
Created By:	Rahman Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	10-07-2015	**Last Revision Date:**	28-07-2015
Actors:	Knowledge Engineer		
Description:	For real time recommendation of classification algorithm for a new dataset, AARM need to be integrated in Mining Minds Data Driven knowledge acquisition approach.		
Trigger:	When AARM is built.		
Preconditions:	• AARM is available • Data driven approach has a unified interface to support AARM as plugin • Data driven has unified interface for accessing Meta-Feature Extractor.		
Postconditions:	AARM is plugged into data driven environment and readily available for real time recommendation of appropriate classification algorithm.		
Normal Flow:	1. Knowledge engineer selects AARM and performs the following tasks; • Analyses number of rules in the AARM • Analyses condition attributes used in each rule of AARM • Transforms rules into executable classes (using any IDE of Java). 2. Knowledge engineer integrates the executable AARM into data driven as follows; a. Write integration code (following unified interface) into data driven source code b. Update possible configuration for newly added AARM plugin. c. Update possible configuration for accessing Meta-Feature Extractor. 3. Knowledge Engineer compile the AARM as integral part with data driven code. 4. Knowledge engineer tests AARM with sample dataset.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Rarely, once the AARM is updated		
Special Requirements:	AARM has acceptable accuracy (60%)		
Assumptions:	AARM is created		

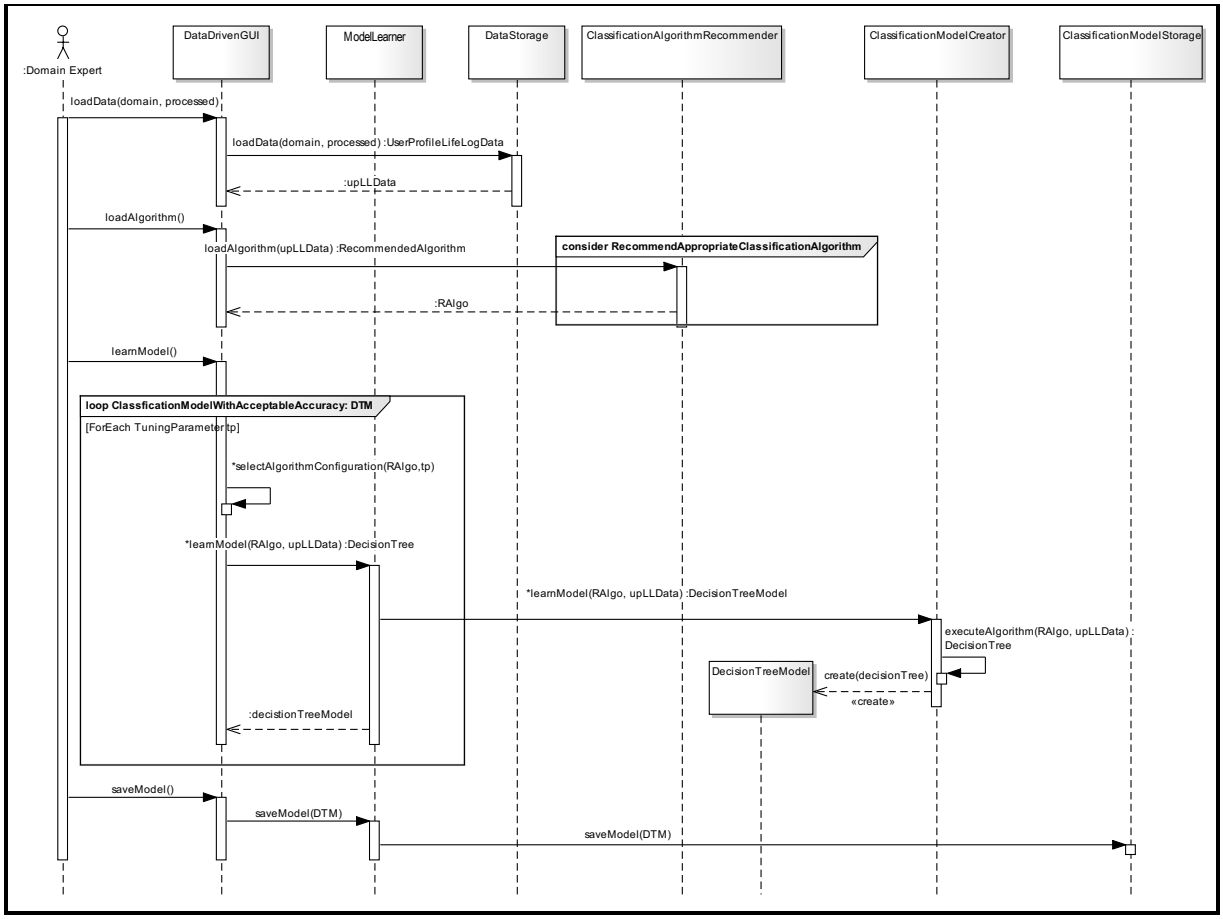
Notes and Issues:	N/A

Sequence Diagram:



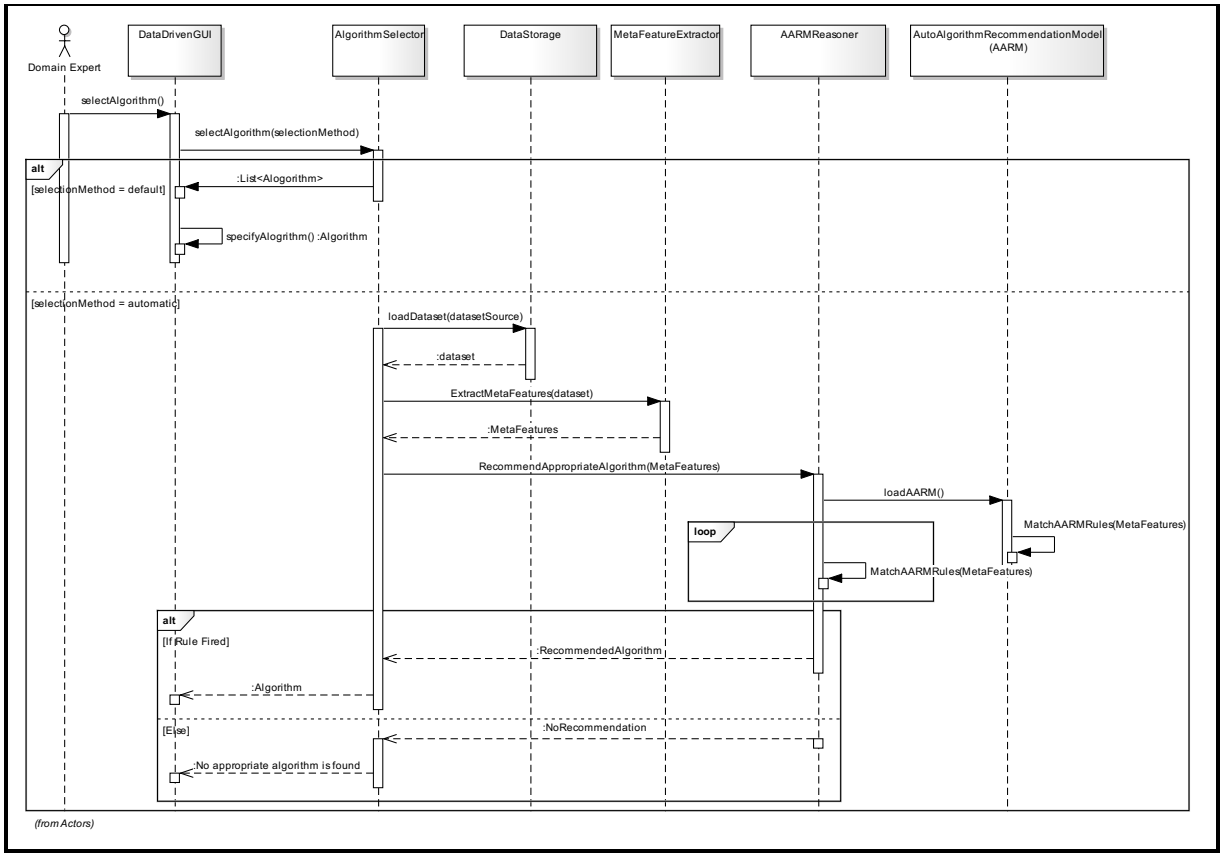
Use Case ID:	KCL-UC-13		
Use Case Name:	Learn classification model		
Created By:	Maqbool Ali	Last Updated By:	Maqbool Ali
Date Created:	10-07-2015	Last Revision Date:	03-07-2018
Actors:	Domain Expert		
Description:	An expert wants to see hidden knowledge from user profile lifelog data that can be explored by model learning mechanism with the help of learning method as well as processed data.		
Trigger:	Learn model required to explore hidden knowledge		

Preconditions:	• System has loaded the prepared user profile lifelog data
Postconditions:	System will build the classification model (decision tree)
Normal Flow:	1. Domain expert loads the user profile lifelog processed data for selected domain 2. System loads the corresponding processed data 3. Domain expert invokes the "Recommend appropriate classification algorithm" use case by providing processed data to load the appropriate learning algorithm 4. System loads the appropriate decision tree learning algorithm 5. Domain expert select the algorithm tuning parameters of selected algorithm for further improving the results 6. System applies the tuning parameters on selected algorithm and computes the learning accuracy after learning the user profile lifelog processed data 7. Repeat the step 5-6 until required learning accuracy is achieved. 8. Domain expert finalizes the classification model with acceptable accuracy and saves the model. 9. System saves the decision tree learning model.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	KCL-UC-14 (Recommend appropriate classification algorithm)
Frequency of Use:	When new service is required, and mining mind have sufficient data for classification model creation
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Acceptable Accuracy: Depends on criticality of the domain. For example, classification model for clinical domain needs high accuracy, while for the case of nutrition domain, high accuracy is not critical.
Sequence Diagram:	



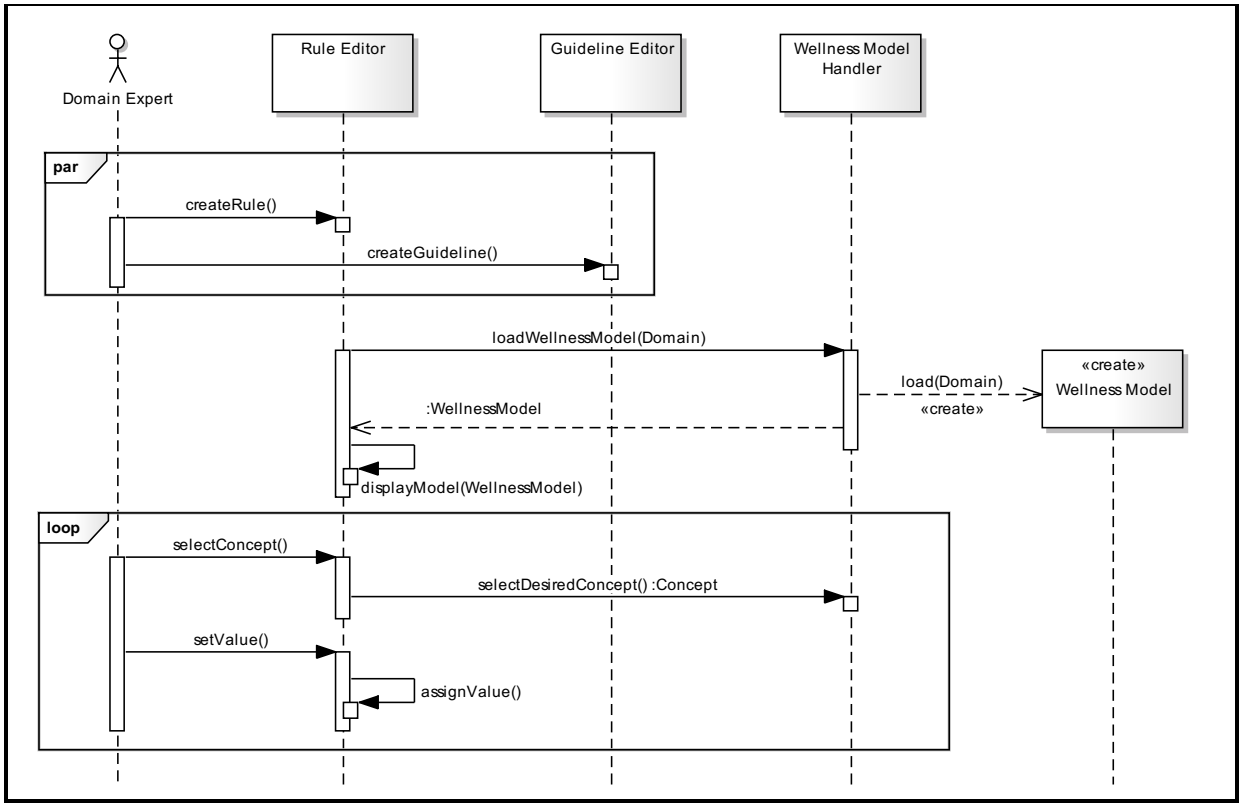
Use Case ID:	**KCL-UC-14**		
Use Case Name:	Recommend appropriate classification algorithm		
Created By:	Rahman Ali	**Last Updated By:**	Rahman Ali
Date Created:	16-07-2015	**Last Revision Date:**	27-07-2015
Actors:	Domain expert		
Description:	For building classification model for the user new dataset, AARM shall automatically recommend appropriate classification algorithm.		
Trigger:	When domain expert wants to build a classification model.		
Preconditions:	• AARM is plugged into the data driven environment • Meta-features extractor is plugged into the data driven environment • New dataset is stored in local machine, structured in .arff file format		
Postconditions:	The recommended appropriate classification algorithm can be used for building classification model		

Normal Flow:	1. Domain expert loads new dataset (.arff file) from the data driven datasets storage using data driven environment. 2. System extracts meta-features of the new dataset by including KCL-UC-07 (alternate flow) 3. Domain experts provides meta-features to system for recommending appropriate classification algorithm 4. System performs meta-reasoning over integrated AARM using the following steps; a. Starts matching each meta-feature value of the new dataset with condition attributes of each rule b. If matched, fires the rule, recommend right hand side of the rule as the appropriate algorithm c. Else, display a message "could not recommend"
Alternative Flows:	1a. 4c(a) If AARM not available or have no acceptable recommendation accuracy, use Weka experimenter.
Exceptions:	N/A
Includes:	KCL-UC-07 (alternate flow)
Frequency of Use:	Frequently, when domain expert needs to select appropriate algorithm for his/her dataset
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram:	



Use Case ID:	**KCL-UC-15**		
Use Case Name:	Manage concepts of domain model		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali, Maqbool Hussain
Date Created:	27-07-2015	**Last Revision Date:**	27-07-2015
Actors:	Domain Expert		
Description:	The concepts of wellness domain shall be used in creation of rules and generation of guidelines in tree format. The domain expert shall easily select the required concepts from wellness domain model.		
Trigger:	Domain Model will be loaded during rule creation or guideline creation.		
Preconditions:	The expert be authenticated with full access of concepts management in domain model		
Postconditions:	The right concept shall be added or edited at the right location in wellness model		

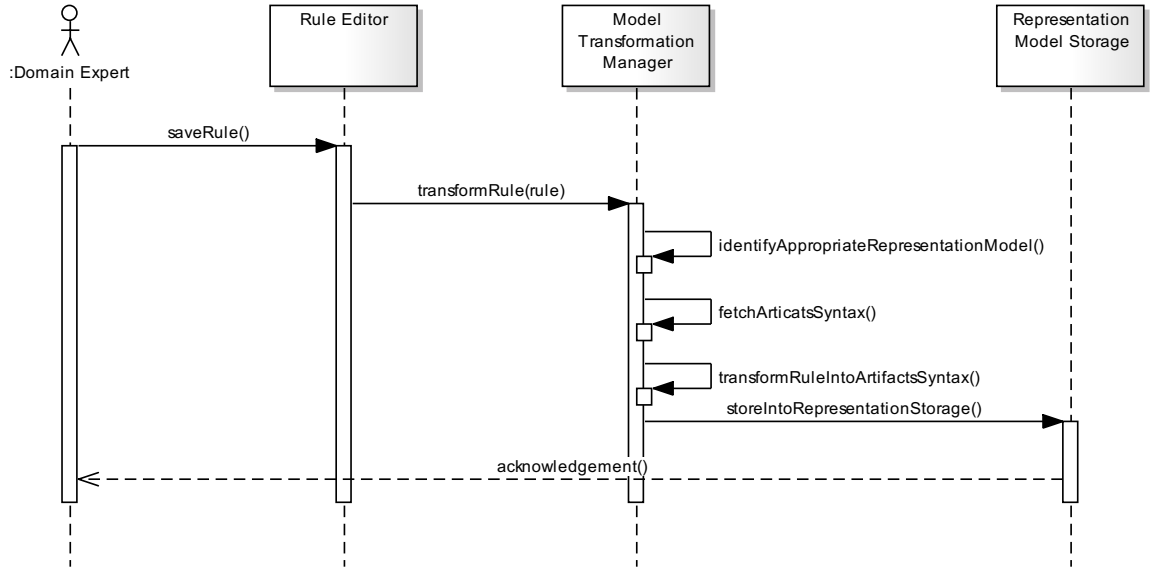
Normal Flow:	1. Domain expert creates rule (using KCL-UC-10) or creates guideline (using KCL-UC-03). 2. System loads domain model for corresponding domain. 3. Domain expert selects concepts from loaded domain model. 4. System associate domain concept to part of rule or guideline tree. 5. Domain expert assigns value to selected concept. 6. System assigns corresponding value to selected concept and show it in rule or guideline tree. 7. Step 4-6 are repeated till rule or guideline is finished.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert wants to add or edit the concepts in wellness model
Special Requirements:	N/A
Assumptions:	Wellness model repository in the system is exist.
Notes and Issues:	If wellness model storage does not exist in system, the administrator shall build the wellness model storage first and configure with system.
Sequence Diagram:	



Use Case ID:	**KCL-UC-16**		
Use Case Name:	Transform Knowledge Rule		
Created By:	Taqdir Ali and Maqbool Hussain	**Last Updated By:**	Maqbool Hussain
Date Created:	27-07-2015	**Last Revision Date:**	27-07-2015
Actors:	Domain Expert		
Description:	The new created rules are needed to transform to some computer interpretable, executable format for execution as well as to shareable, standard format for maintenance and sharing with other organizations.		
Trigger:	Whenever domain expert wants to store the created or updated rule.		
Preconditions:	The expert created rule successfully and the system validated the rule.		
Postconditions:	System shall transform the created and validated rule into appropriate representation.		
Normal Flow:	1. Domain expert save the new created rule or update the existing rule. 2. The system identifies the appropriate representation model 3. Fetch the artifacts of the identified representation model		

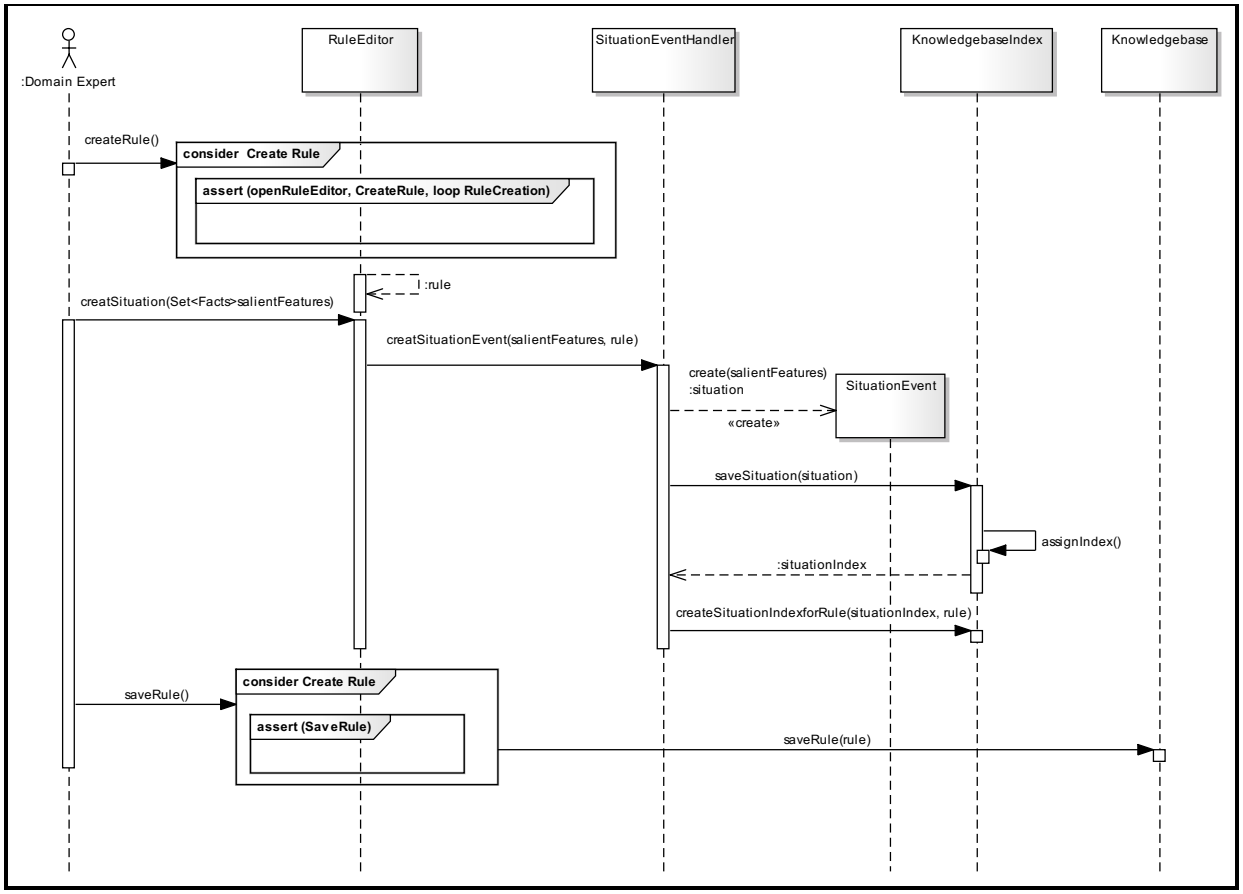
	4. Transforms the rule into the artifacts and syntax of the identified representation model. 5. The rule in the representation model is stored into the repository.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert wants to save rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Appropriate representation configuration is challenging task.

Sequence Diagram:

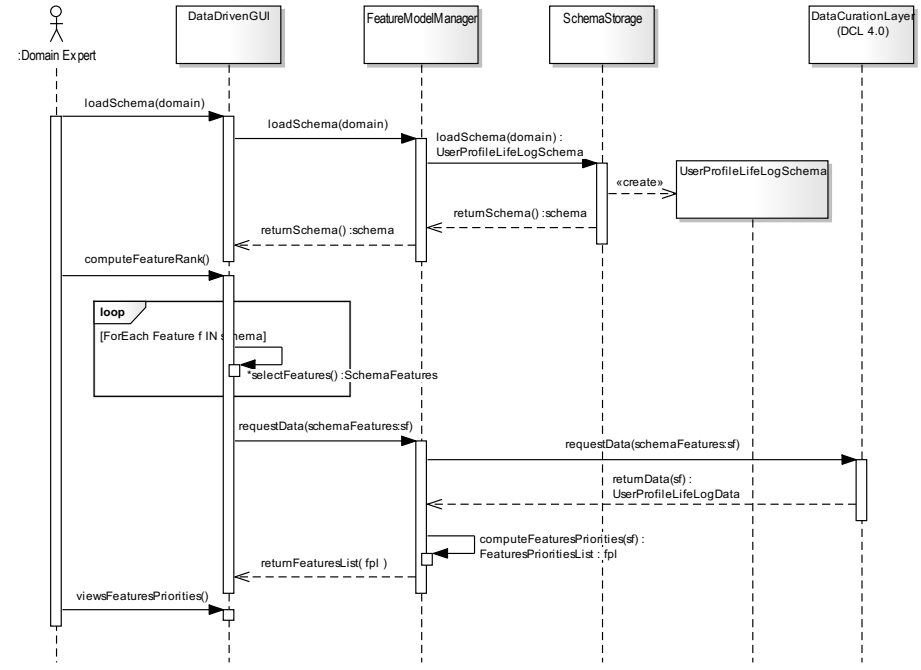


Use Case ID:	**KCL-UC-17**		
Use Case Name:	Create Situation Event		
Created By:	Maqbool Hussain	**Last Updated By:**	Maqbool Hussain
Date Created:	29-07-2015	**Last Revision Date:**	29-07-2015
Actors:	Domain Expert		

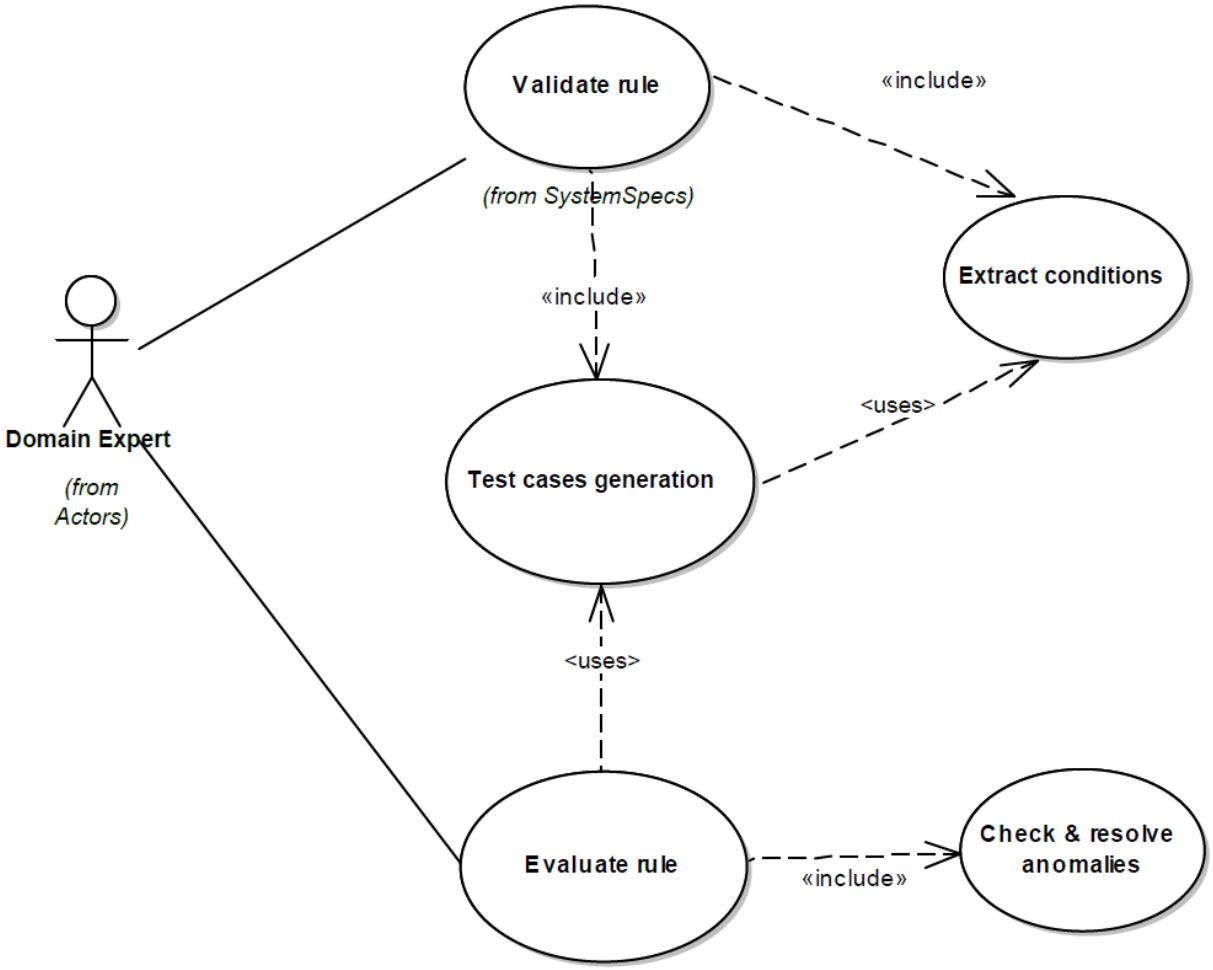
Description:	Situation Event is important features of mining mind which includes set of associated recommendation rules. Situation event is created, and the rule is indexed in knowledgebase based on situation event.
Trigger:	Whenever domain expert wants to store the created or updated rule.
Preconditions:	The rule has salient features based on which the rule can be indexed.
Postconditions:	• Rule is saved into knowledgebase • Rule is indexed based on the created situation event
Normal Flow:	1. Domain expert start creating rule; a. Performs steps 1-5 in KCL-UC-10. b. Selects salient features (indicating as event) from conditions of the rule. 2. The system performs following actions; a. Create situation event with salient features. b. Saves the situation event and assign index (generate if situation event is not existing in knowledgebase index). c. Index the created rule with situation event. 3. Domain expert saves the rule by performing steps 5-6 in KCL-UC-10. 4. System saves the rule and index of the rule.
Alternative Flows:	N/A
Exceptions:	N/A
Extends:	Create Rule (KCL-UC-10)
Frequency of Use:	Whenever domain expert wants to save rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	
Sequence Diagram:	



Use Case ID:	KCL-UC-18		
Use Case Name:	Compute features priorities		
Created By:	Maqbool Ali	Last Updated By:	Maqbool Ali
Date Created:	01-07-2018	Last Revision Date:	03-07-2018
Actors:	Domain Expert, DCL		
Description:	Features priorities computation help domain expert for selecting appropriate features from available schema		
Trigger:	Prior to data preprocessing needed for required data		
Preconditions:	• KCL has access through service interface to retrieve user profile lifelog data from DCL • KCL and DCL has agreement on common data representation format		
Postconditions:	KCL will display the features priorities list		
Normal Flow:	1. KCL connects to DCL via unified service interface and sends request for user profile lifelog data based on selected features of schema and features conditions.		

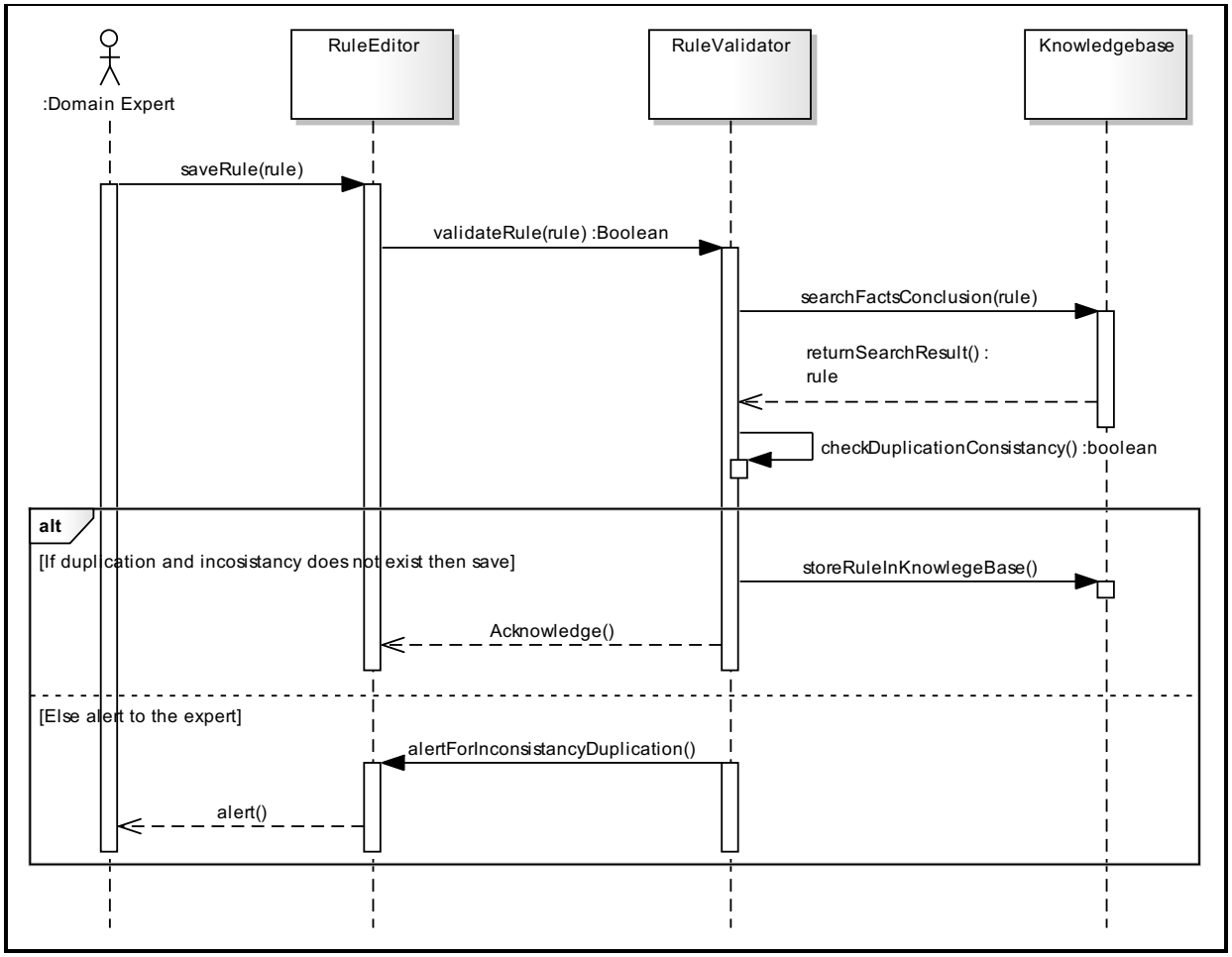
	2. DCL sends the required data to KCL. 3. KCL receives user profile lifelog data 4. KCL computes the features priorities 5. KCL displays the features priorities list
Alternative Flows:	N/A
Exceptions:	2a. KCL unable to connect to DCL a. KCL connection is failed during retrieving user profile lifelog data b. KCL hold and will retry after some time to connect to DCL and retrieve the user profile lifelog data 3a. KCL unable to verify lifelog data conformance a. KCL fail to conform the data representation from DCL b. KCL will send message to DCL about incompatible data format
Includes:	N/A
Frequency of Use:	When new service is required, and mining mind have sufficient data for classification model creation
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	If DCL is unable to send data, then alternate strategy has to be considered.
Sequence Diagram:	

3.4.4 Rule Validator: Use case diagram



Use Case ID:	**KCL-UC-19**		
Use Case Name:	Validate Rule		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017
Actors:	Domain Expert		
Description:	In new rules creation and editing existing rules, duplication and inconsistency may occur. The validation is needed to validate and find the duplication and inconsistency among the rules.		
Trigger:	• When new rule is going to create. • When existing rule is going to update		

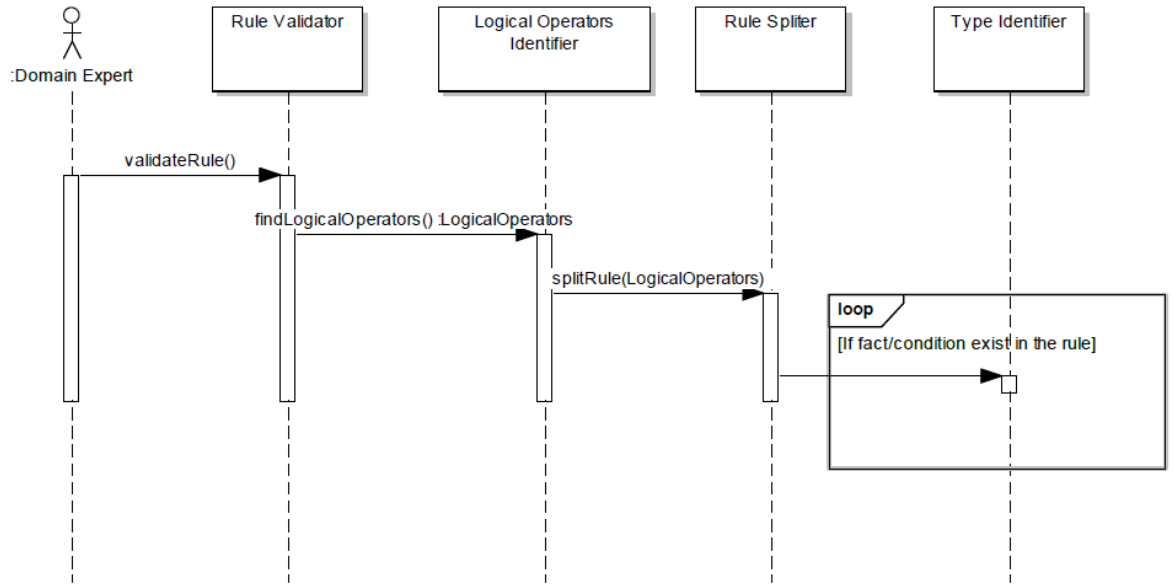
Preconditions:	The rule creation and editing process completed by physician successfully.
Postconditions:	The validated rule shall be saved into knowledge base.
Normal Flow:	3. Domain expert saves the created rule. 4. System validate rule for inconsistency and duplication as follows a. Fetch the facts and conclusion of existing rules. b. The new or updated rule approved for having no inconsistency and duplication. c. Created rule stores into the rules repository. d. Acknowledge the expert to save the rule successfully.
Alternative Flows:	3b. The created rule is found having inconsistency or duplication with existing rules in the rules repository a. The system produces alert the inconsistency or duplication of the rule with existing rules in repository. b. Domain expert review the alert message and correct the created rule. c. Step 1 and Step 2 of normal flow is executed.
Exceptions:	N/A
Includes:	1. Extract conditions 2. Test cases generation
Frequency of Use:	Whenever domain expert wants to save the rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.
Sequence Diagram:	



Use Case ID:	**KCL-UC-20**		
Use Case Name:	Extract conditions		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017
Actors:	Domain Expert		
Description:	Whenever the rule is validated, the test cases are generated, the test cases depends on conditions of the rule. Therefore, all the facts and conditions are needed to extract from rules.		
Trigger:	• When new rule is going to validate. • When existing rule is going to validate.		
Preconditions:	The rule creation and editing process completed by physician successfully.		
Postconditions:	Multiple pairs of facts and conditions are extracted.		

Normal Flow:	1. System fetch the rule, which is being to be added or updated 2. System shall identify the logical operators in the rule. 3. System splits the rule based on the identified logical operators 4. In splat conditions, find the type of keys and their values. a. If type is discrete then execute non-interval test case generation.
Alternative Flows:	4b. If type is interval, then a. Execute interval case generator
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert wants to validate the rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.

Sequence Diagram:

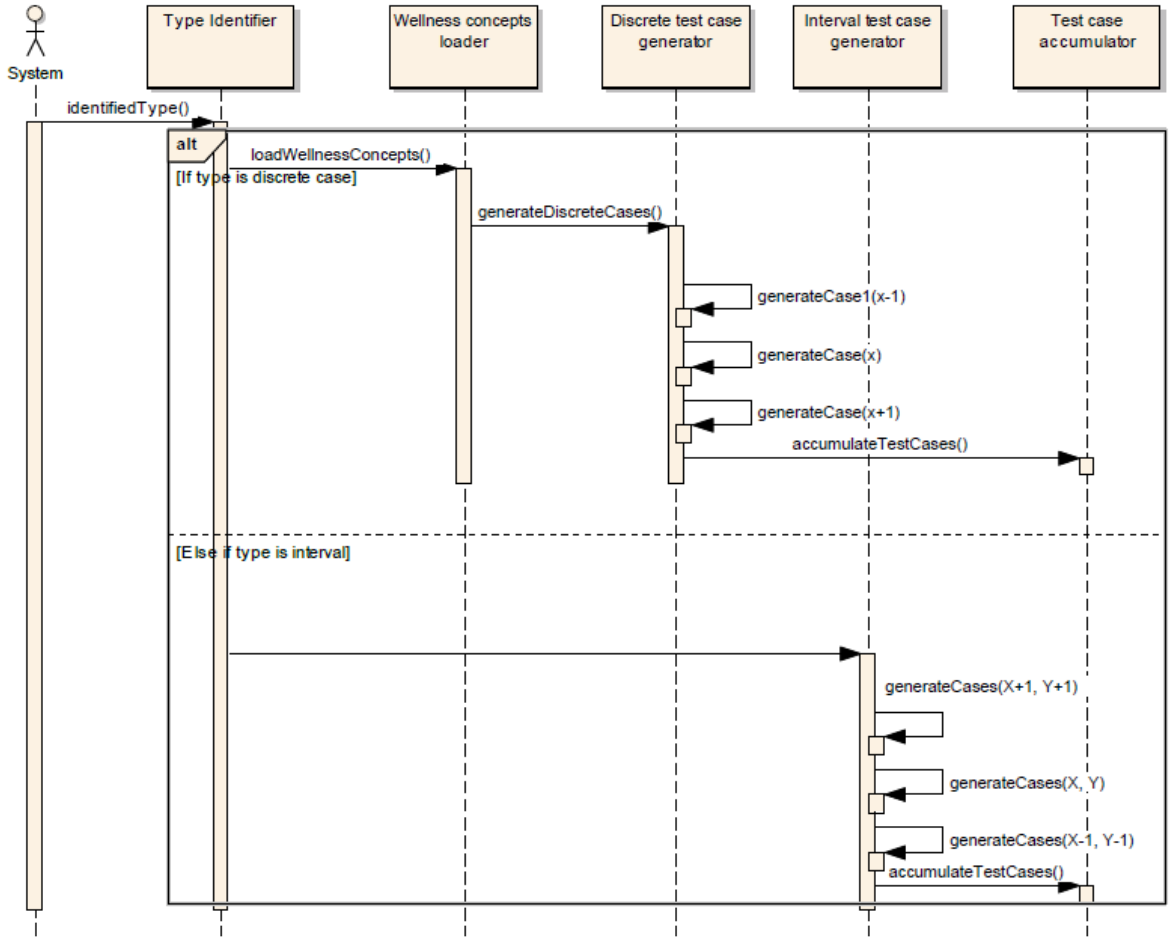


Use Case ID:	**KCL-UC-21**		
Use Case Name:	Test cases generation		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali

Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017
Actors:	Domain Expert		
Description:	Whenever the rule is validated, the test cases are generated, the new created rule or existing modified rule is checking for duplications and conflict using the generated test cases.		
Trigger:	• When new rule is going to validate. • When existing rule is going to validate.		
Preconditions:	The rule conditions and facts are identified and extracted successfully.		
Postconditions:	Multiple test cases shall generate after successful process.		
Normal Flow:	1. If type of extracted condition or fact is discrete a. System shall fetch the possible values from wellness model b. Generate matched test cases with respect to the extracted condition. c. Generate unmatched conditions with respect to the extracted condition. d. Store the generated test cases with corresponding conditions in test case base.		
Alternative Flows:	1. (b) If type of extracted condition or fact is interval a. System finds the type of interval b. If interval is single side infinite interval or exact value interval, then generate following 3 test cases. i. Generate case for X-1 ii. Generate case for X iii. Generate Case for X+1 c. If interval is double sided finite interval, then generate following 6 test cases. i. Generate case for X-1 ii. Generate case for X iii. Generate Case for X+1 iv. Generate case for Y-1 v. Generate case for Y vi. Generate Case for Y+1		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Whenever domain expert wants to validate the rule		
Special Requirements:	N/A		
Assumptions:	N/A		

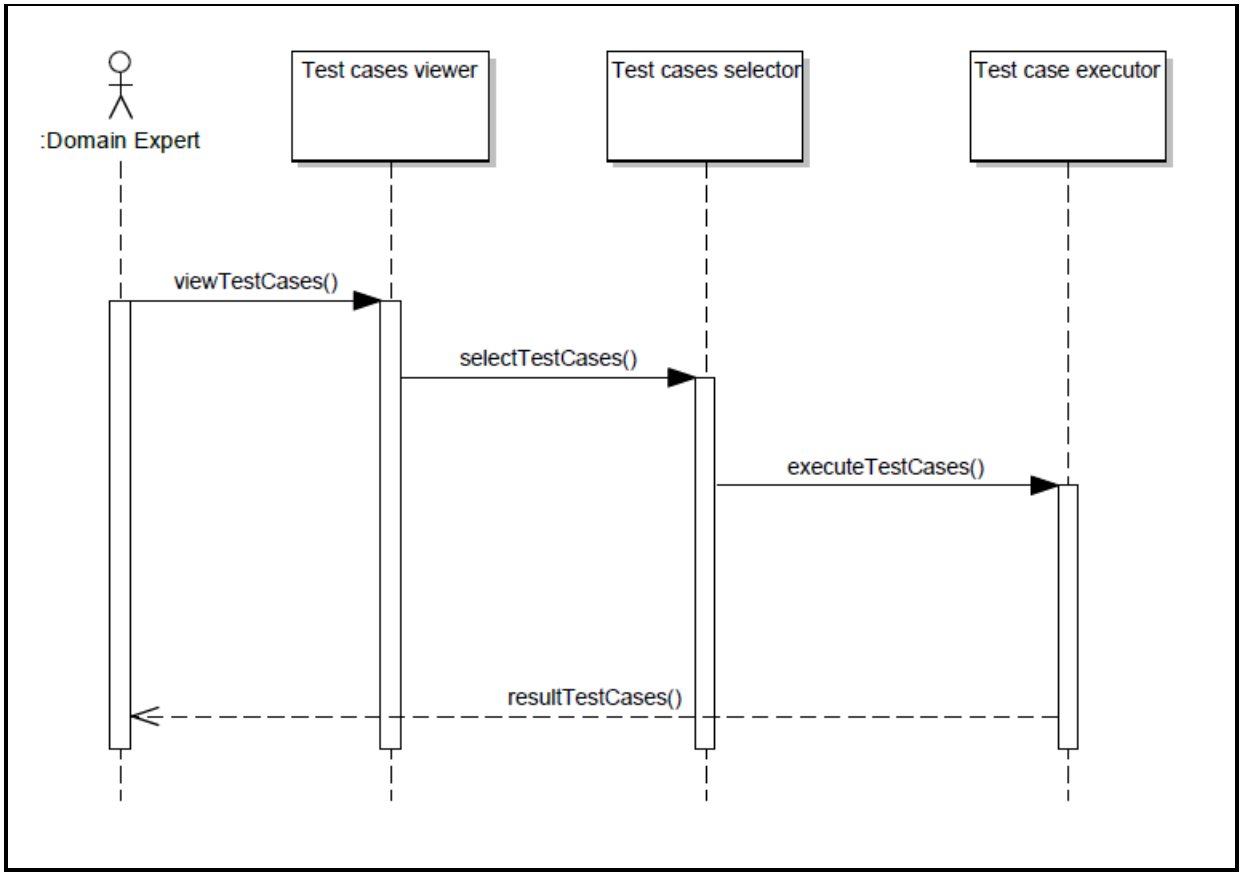
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.

Sequence Diagram:



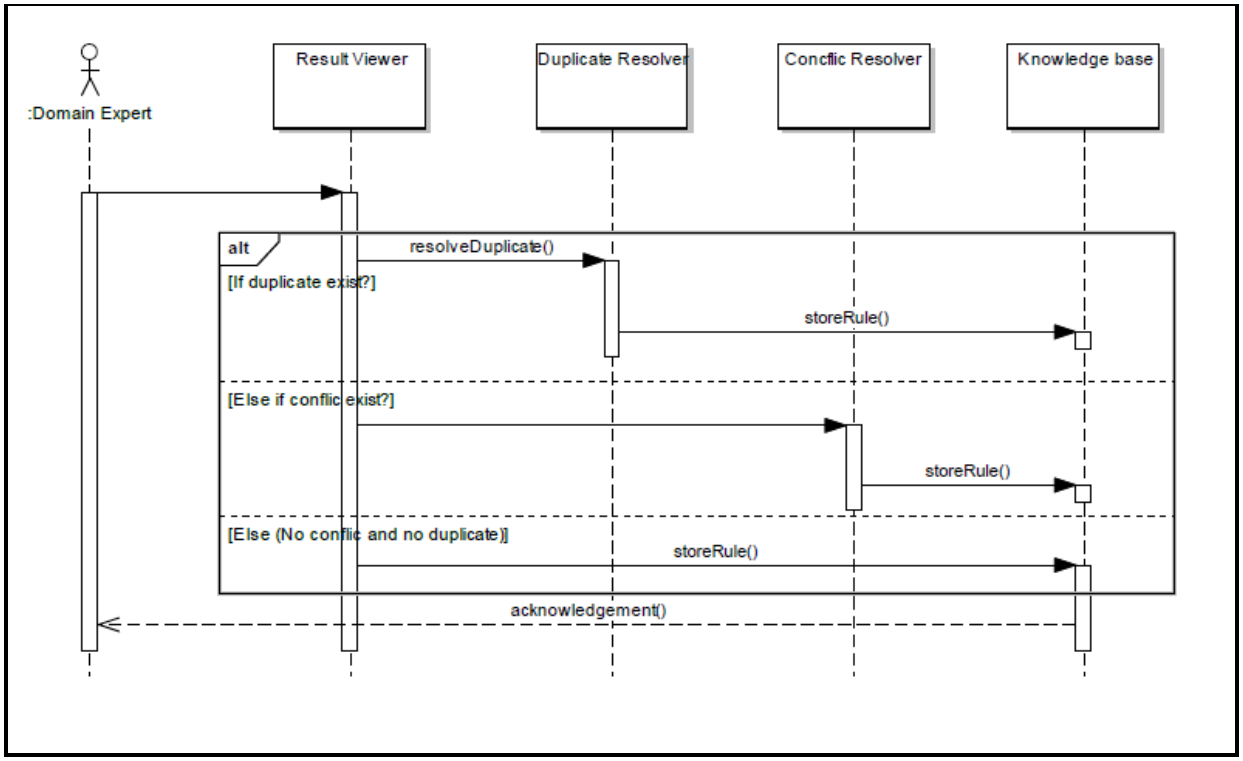
Use Case ID:	**KCL-UC-22**		
Use Case Name:	Evaluate rule		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017
Actors:	Domain Expert		
Description:	Whenever the rule is to be validated, the new created or existing modified rule is needed to test with respect to generated test cases, then evaluate the rule with those test cases		

Trigger:	• When new rule is going to validate. • When existing rule is going to validate.
Preconditions:	The test cases are successfully generated and executed for rule which is going to evaluate.
Postconditions:	Find the duplicate and conflicted rules based on executed test cases.
Normal Flow:	1. Domain experts views all the auto generated test cases on viewer. 2. Domain expert selects the test cases to execute and test on the created rule 3. Domain experts executes the selected test cases 4. The system gives results of executed test cases based on the included use case “check and resolve anomalies”.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert want to validate the rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.
Sequence Diagram:	



Use Case ID:	**KCL-UC-23**		
Use Case Name:	Check and resolve anomalies		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017
Actors:	Domain Expert		
Description:	Whenever the rule is to be validated, the executed test cases checks and identifies the anomalies in form of duplicate and conflict. The domain experts identified and resolve those anomalies.		
Trigger:	• When new rule is going to validate. • When existing rule is going to validate.		
Preconditions:	The test cases are successfully generated and executed for rule which is going to evaluate.		
Postconditions:	Find and resolve the duplicate and conflicted rules based on executed test cases.		

Normal Flow:	1. System shows the results of execution of test cases on the creation rule or modification rule. 2. Domain experts views the result messages on result viewer for finding duplication and conflict 3. If duplicate rules exist a. Discard the new created rule. 4. If conflict exists in the created rule a. Find the conflicting attributes and facts in the rule. b. Resolve the conflict and gives the priority to rules 5. Store the rules into the knowledge base.
Alternative Flows:	3 (a) If there is no duplicate a. Check the rule for conflict. 4 (a) If there is no conflict a. Store the rule into the knowledge base
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert wants to validate the rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.
Sequence Diagram:	


:Domain Expert
Result Viewer
Duplicate Resolver
Conflic Resolver
Knowledge base
alt
[If duplicate exist?]
resolveDuplicate()
storeRule()
[Else if conflic exist?]
storeRule()
[Else (No conflic and no duplicate)]
storeRule()
acknowledgement()

4. Service Curation Layer

4.1 Functional Requirement

Requirements #ID	Description
SCL-FR-01	The layer shall receive the service request from user application, third party application, or mining mind platform generated events
SCL -FR-02	The layer shall retrieve data from intermediate database (user profile, lifelog, and environmental variables)
SCL -FR-03	The layer shall retrieve production knowledge from knowledge base
SCL -FR-04	The layer shall identify the unresolved user requests and notify the corresponding layer for missing knowledge
SCL -FR-05	The layer shall deliver the results to the service requester and to corresponding layer of mining mind for persistence
SCL -FR-06	The layer shall generate goal-based recommendation
SCL -FR-07	The layer shall personalize the recommendation based on location and weather information.
SCL -FR-08	The layer shall provide take care of user preference-based recommendation through continuous feedback from the user.

4.2 Non-Function Requirements

Requirements #ID	Description
SCL -NFR-01	The service request shall include the user identification required to prepare the data request
SCL -NFR-02	The user application and host layer of the services shall connect to the internet

4.3 Terms and Definitions

Terms	Description
SCL	Service Curation Layer
DCL	Data Curation Layer
KCL	Knowledge Curation Layer

SL	Supporting Layer
Reasoning	The process of producing recommendations
RBR	Rule Base Reasoning
Case	A case is collection of data/information that represent a complete state of a subject in a particular time
CBR	Case Base Reasoning
Hybrid Reasoning	A combined reasoning of RBR and CBR
Unresolved Case	A new case for which the existing knowledge is insufficient to solve
Production Knowledge	The knowledge is in production environment and is ready for execution
Recommendation	An actionable statement provided to the subject for healthy habit induction
Fact	An informative statement provided to the subject for education
Goal	A target that a subject is intending to achieve
Achievement	The measurement of the subject status so far towards achieving the goal
Interpreted Recommendation	The recommendation passed through the process of interpretations
Environmental Variables	Environment variables represent the factors related to environment rather than user such as weather, time, season etc.
Service	An outcome of the system in which a subject is interested such as recommendation, facts, alerts, notifications.

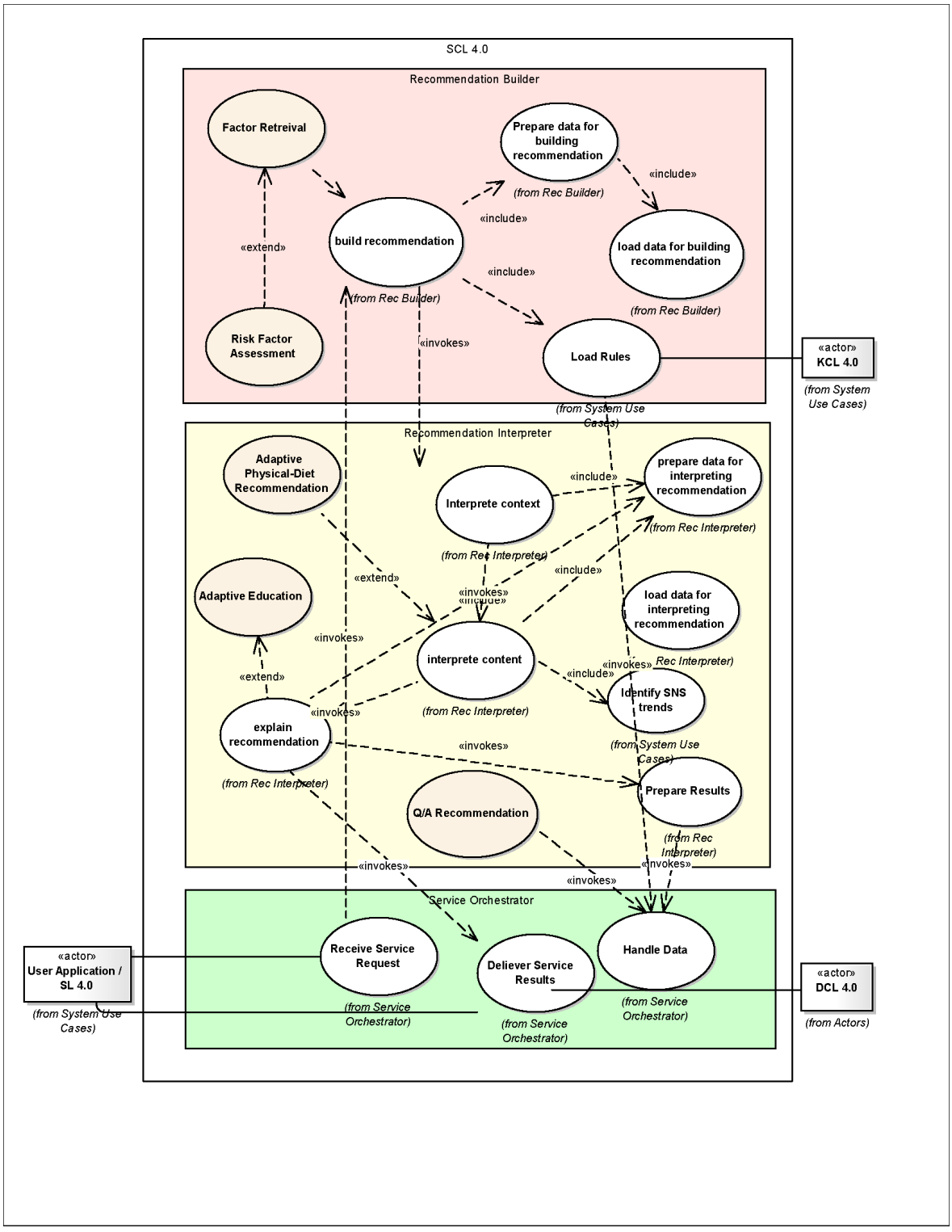
4.4 Use Cases

4.4.1 List

Use Case #ID	Description
SCL-UC-01	Load data for building recommendation
SCL-UC-02	Prepare data for building recommendation
SCL-UC-03	Load Rules
SCL-UC-04	Build Recommendation

SCL-UC-05	Receive Production Knowledge
SCL-UC-06	Report Unresolved Case
SCL-UC-07	Load data for interpreting recommendation
SCL-UC-08	Prepare data for interpreting recommendation
SCL-UC-09	Interpret Context
SCL-UC-10	Interpret Content
SCL-UC-11	Explain recommendations
SCL-UC-12	Prepare Results
SCL-UC-13	Receive service request
SCL-UC-14	Handle Data
SCL-UC-15	Deliver service results
SCL-UC-16	Goal-based recommendation Generation
SLC-UC-17	Provide Educational Support for user awareness
SCL-UC-18	Location- and weather-based Personalized recommendation
SCL-UC-19	User Preference-based refined recommendations
SCL-UC-20	Provide Alternative Recommendation
SCL-UC-21	Retrieve risk factors
SCL-UC-22	Retrieve risk assessment
SCL-UC-23	Adaptive education
SCL-UC-24	Adaptive physical and nutrition recommendation
SCL-UC-25	Question/Answer recommendation

4.4.2 Use Case Diagram

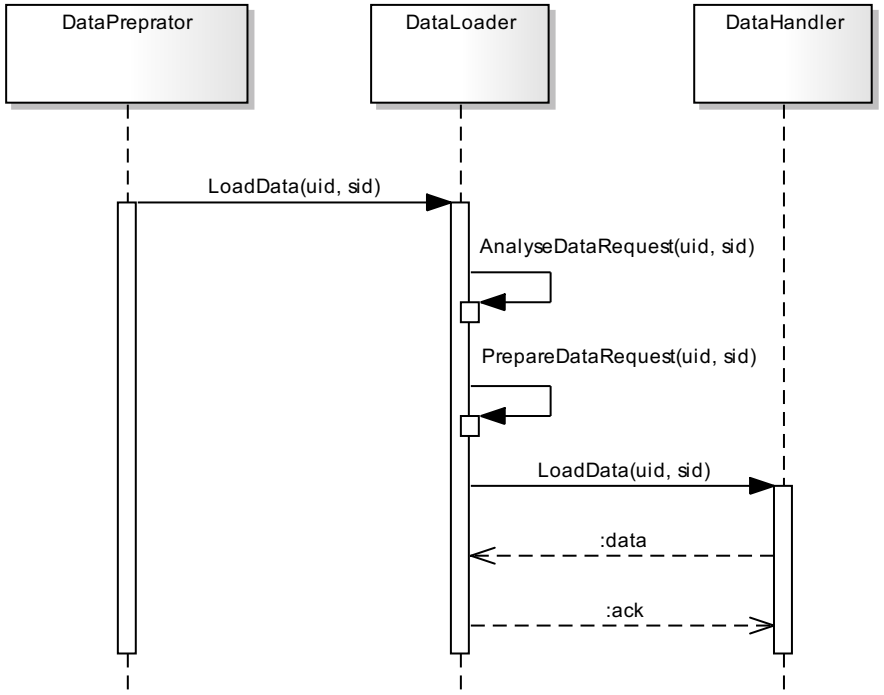


4.4.3 Description

Use Case ID:	SCL-UC-01		
Use Case Name:	Load data for building recommendation		
Created By:	Rahman Ali	**Last Updated By:**	Rahman Ali
Date Created:	July 14, 2015	**Last Revision Date:**	July 28, 2015
Actors:	SCL-UC-02 (Prepare Data)		
Description:	Retrieving user profile and lifelog data is required for reasoning to generate recommendation. This data is retrieved using Data Handler of the Service Orchestrator.		
Trigger:	Triggered when a new service request is received from the user application or DCL.		
Preconditions:	User profile and lifelog data is available in user lifelog.		
Postconditions:	User profile and lifelog data is successfully retrieved and prepared for reasoner to process.		
Normal Flow:	1. Data Preparator sends request for loading data 2. Data Loader receives the request and performs the following tasks; a. Analyses the request and user for the appropriate data loading b. Prepare separate requests for user lifelog data 3. Data Loader sends analyses request to Data Handler 4. Data Handler provides the data to Data Loader		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Very frequent; repeated for every service request		
Special Requirements:	N/A		

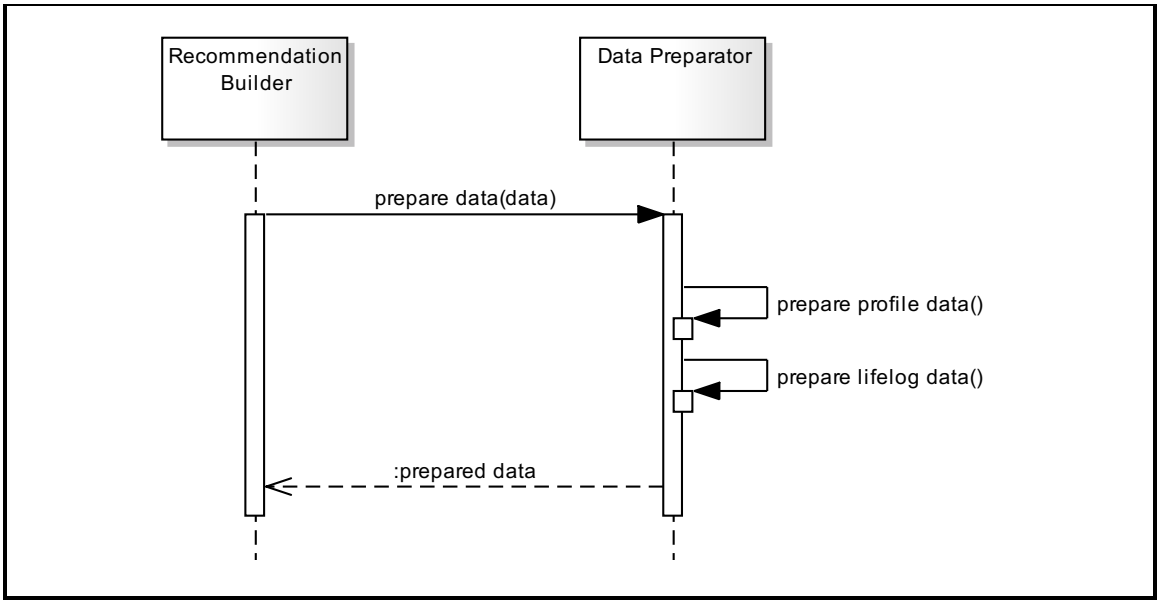
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A

Sequence Diagram



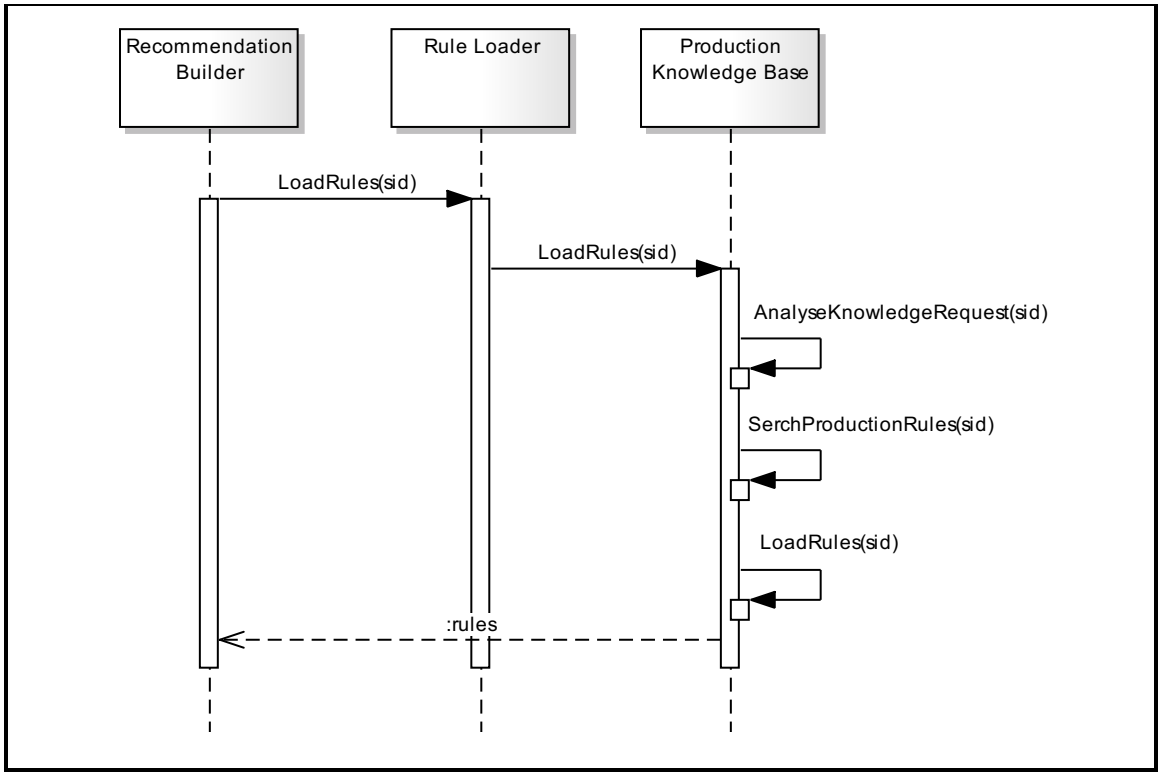
Use Case ID:	SCL-UC-02		
Use Case Name:	Prepare data for building recommendation		
Created By:	Rahman Ali	**Last Updated By:**	Rahman Ali
Date Created:	July 14, 2015	**Last Revision Date:**	July 28, 2015
Actors:	SCL-UC-04 (Build Recommendation)		
Description:	Knowledge based reasoning requires prepared data to execute the rules during the reasoning process.		

Trigger:	Triggered when new service request is made for generating recommendations
Preconditions:	User profile and lifelog data is loaded into RB 2
Postconditions:	User prepared data is readily available for reasoner to process.
Normal Flow:	1. Recommendation Builder sends data preparation request to Data Preparator along with the loaded data 2. Data Preparator prepares profile data 3. Data Preparator prepares lifelog data 4. Data Preparator returns prepared data to Recommendation Builder
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-01
Frequency of Use:	Very frequent; for every service request
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



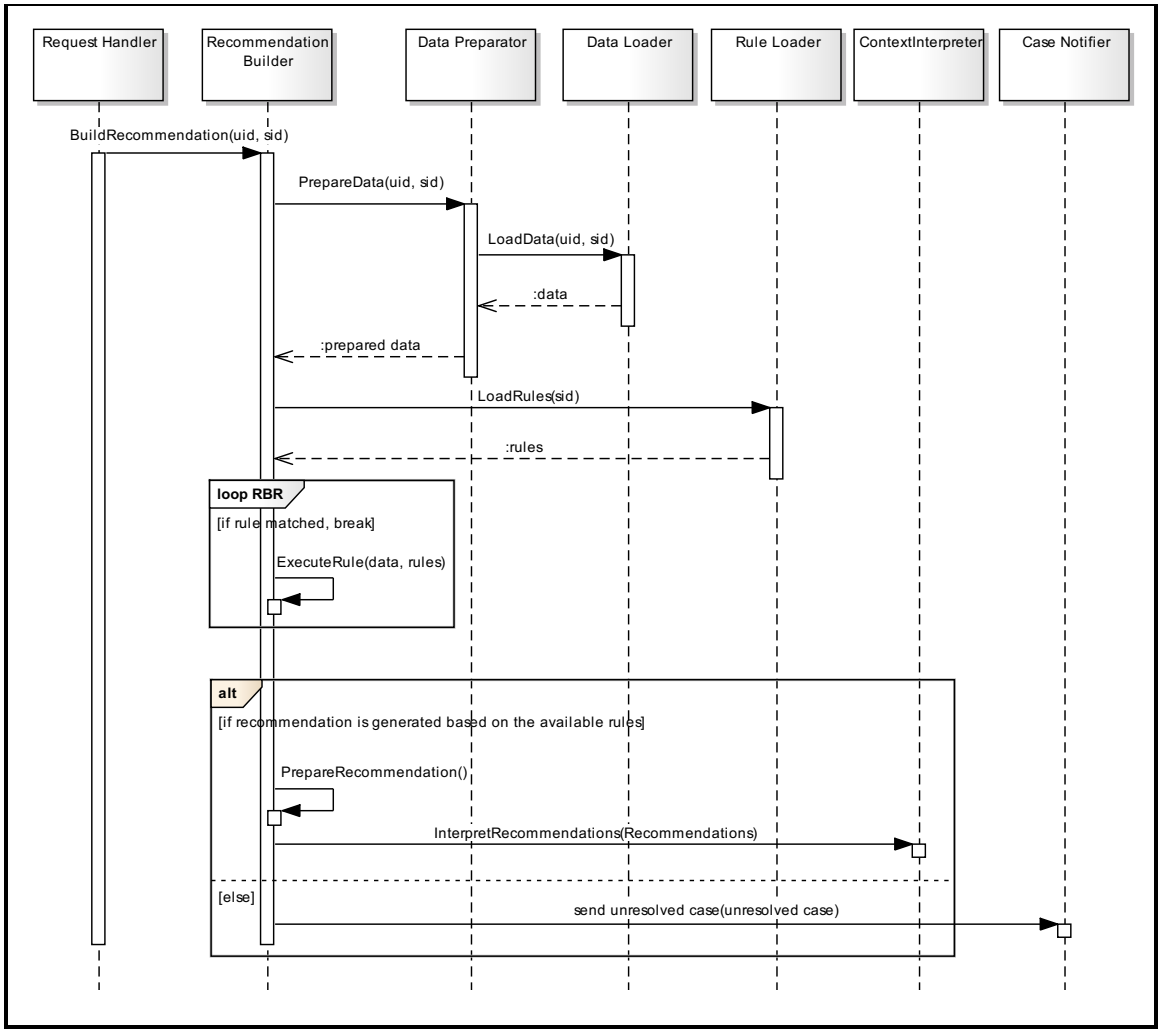
Use Case ID:	SCL-UC-03		
Use Case Name:	Load Rules		
Created By:	Rahman Ali	**Last Updated By:**	Rahman Ali
Date Created:	July 28, 2015	**Last Revision Date:**	July 28, 2015
Actors:	SCL-UC-04 (Build Recommendation)		
Description:	Rule-based reasoned needs knowledge rules to perform reasoning using the prepared data to generate recommendations for the service request.		
Trigger:	At the time when new service request arrives for recommendation.		
Preconditions:	• Updated knowledge is available in Production Knowledge Base. • KCL and RB 2 agree on common format of production rules.		
Postconditions:	The reasoned is ready to execute the rules and generate recommendations.		

Normal Flow:	1. Recommendation Builders send knowledge load request to Rule Loader 2. Rule Loader sends request to Production Knowledge Base 3. System performs the following tasks; a. Analyses the request knowledge b. Search production knowledge base for the requested rules c. Loads the rules d. Provides the rules back to Recommendation Builder
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Frequent: when reasoner is invoked for new service generation.
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



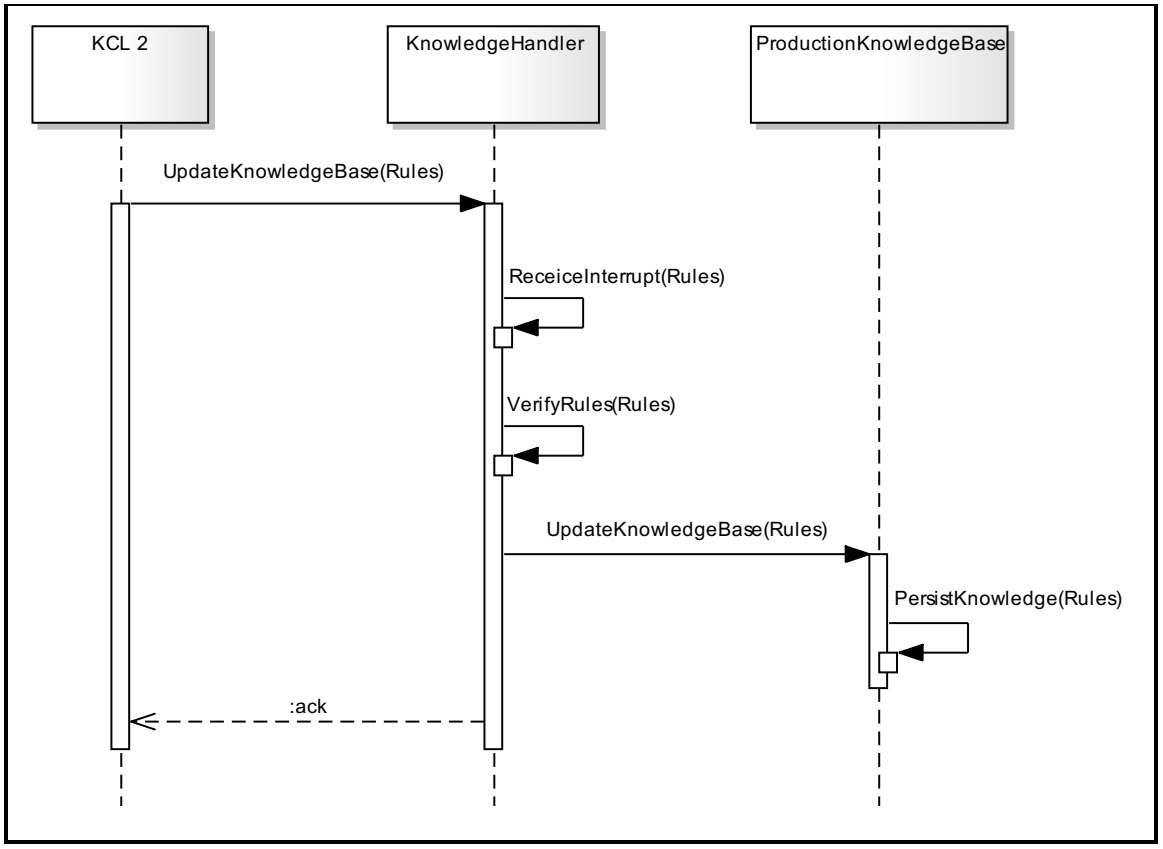
Use Case ID:	SCL-UC-04		
Use Case Name:	Build recommendations		
Created By:	Rahman Ali	**Last Updated By:**	Rahman Ali
Date Created:	July 15, 2015	**Last Revision Date:**	July 15, 2015
Actors:	SCL-UC-13 (Request Handler), SCL-UC-04, SCL-UC-02, SCL-UC-01, SCL-UC-03, SCL-UC-09 (Interpret Context)		
Description:	RBR performs rule-based reasoning to generate recommendations using the production rules and prepared data.		
Trigger:	At the time when new service request arrives for recommendation.		
Preconditions:	Knowledge is available in Production Knowledge Base.		

Postconditions:	The recommendation is reported to RI 2, if reasoning is successful, otherwise the new case is provided to Unified Knowledge Interface along with the missing rule message.
Normal Flow:	1. Request Handler invokes recommendation builder for recommendation 2. Recommendation Builder load prepared data 3. Recommendation Builder retrieves loaded rules 4. Recommendation Builder performs rule-based reasoning on the prepared data and loaded rules 5. Recommendation Builder generates recommendation and perform the following tasks; a. Prepare recommendation b. Provides recommendations to Context Interpreter for interpretation
Alternative Flows:	5a. The system could not find rule to execute a. Recommendation Builder sends message along with Unresolved Case to Case Notifier
Exceptions:	N/A
Includes:	SCL-UC-02, SCL-UC-03
Frequency of Use:	Frequent: when recommendation builder is invoked for generating recommendation.
Special Requirements:	N/A
Assumptions:	KCL and RB 2 agree on common format of production rules.
Notes and Issues:	N/A
Sequence Diagram	



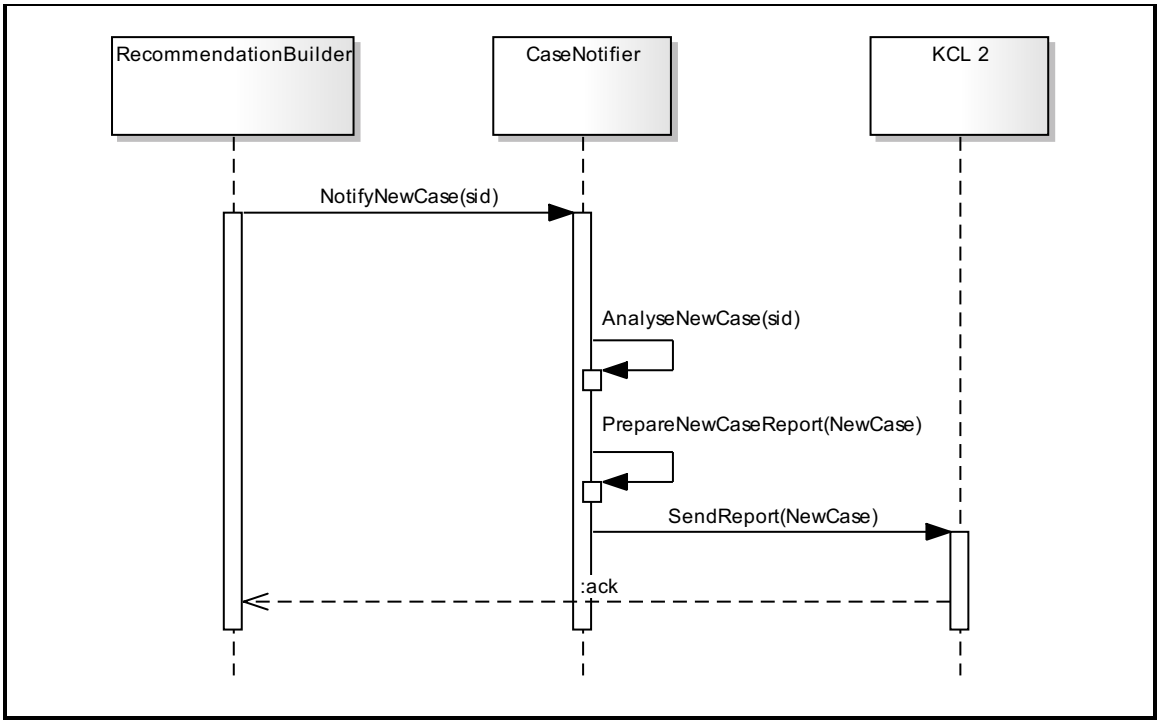
Use Case ID:	SCL-UC-05		
Use Case Name:	Receive Production Knowledge		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 5, 2015	**Last Revision Date:**	July 28, 2015
Actors:	KCL		
Description:	The knowledge is originated by KCL and is transfer to SCL 2 to keep a local copy of the production knowledge.		

Trigger:	At knowledge creation/update time
Preconditions:	SCL 2 and KCL has a common representation agreement
Postconditions:	The SCL 2 copy of knowledge is updated and is synchronized with KCL
Normal Flow:	1. KCL interrupt Knowledge Handler for new knowledge 2. Knowledge Handler verifies the knowledge 3. Knowledge Handler make a local of the received knowledge in the Production Knowledge Base 4. Knowledge is persisted in Production Knowledge Base 5. KCL is acknowledged of the knowledge receipt
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Less frequent: at knowledge creation/update time
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SCL-UC-06		
Use Case Name:	Report unresolved case		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 5, 2015	**Last Revision Date:**	July 28, 2015
Actors:	SCL-UC-04 (Build Recommendation), KCL		
Description:	Notifying KCL that reasoner is incapable to generate recommendation for the service request. KCL may be able to acquire new knowledge for such service request to handle in future.		
Trigger:	At the time when reasoner is not capable to generate recommendation because of insufficient knowledge in the KB.		

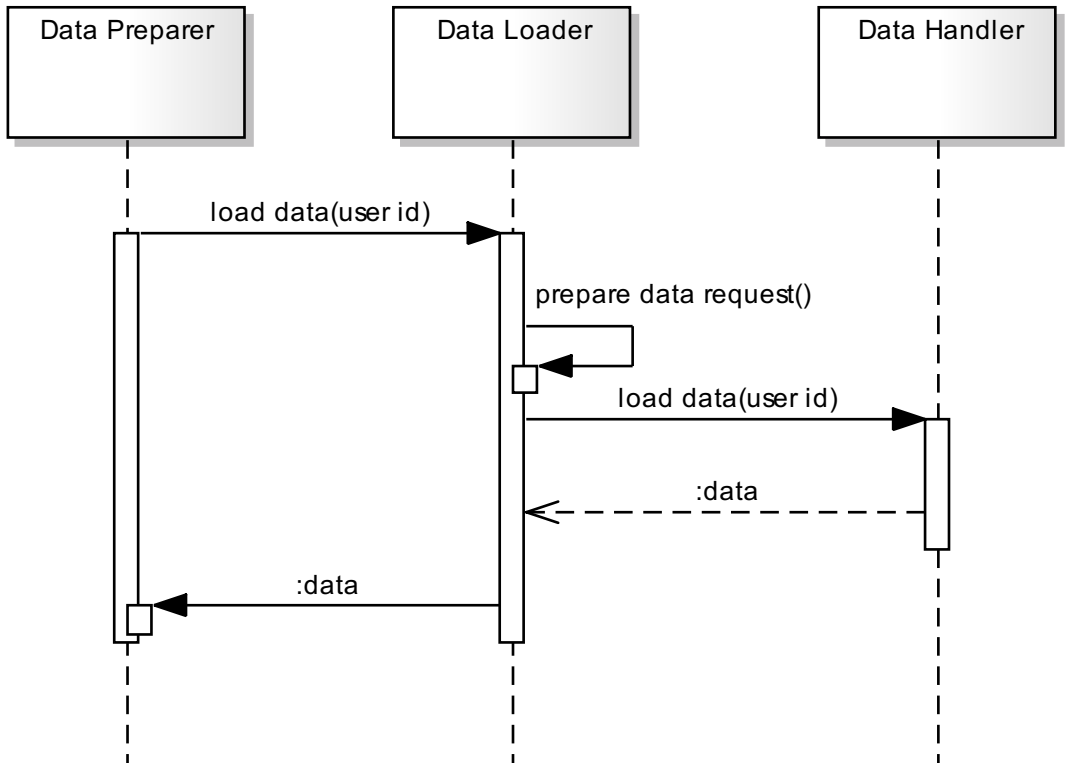
Preconditions:	Reasoner has completed the reasoning process
Postconditions:	The message with reason is successfully reported to KCL
Normal Flow:	1. Recommendation Builder notify unresolved case as new case prepare the report 2. Case Notifier analyses the new case 3. Case Notifier prepare the new case report 4. Case Notifier sends new case report to KCL 5. KCL acknowledges the new case receipt
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Less frequent: when reasoner detects new case not handled with existing knowledge.
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SCL-UC-07		
Use Case Name:	Load data for interpreting recommendations		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 15, 2015	**Last Revision Date:**	July 28, 2015
Actors:	Data Preparer		
Description:	The data is loaded from DCL through Service Orchestrator in order to interpret the recommendations		
Trigger:	After recommendation is built		
Preconditions:	• Recommendation are built • User profile is stored in lifelog • Context is recognized		
Postconditions:	The user profile, lifelog, and environmental variable data is available for preparation.		

Normal Flow:	1. Data Loader receives interrupt from Data Preparer 2. Data loader prepare data request 3. Data loader send request to Data Handler 4. Data loader receives data from Data Handler
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

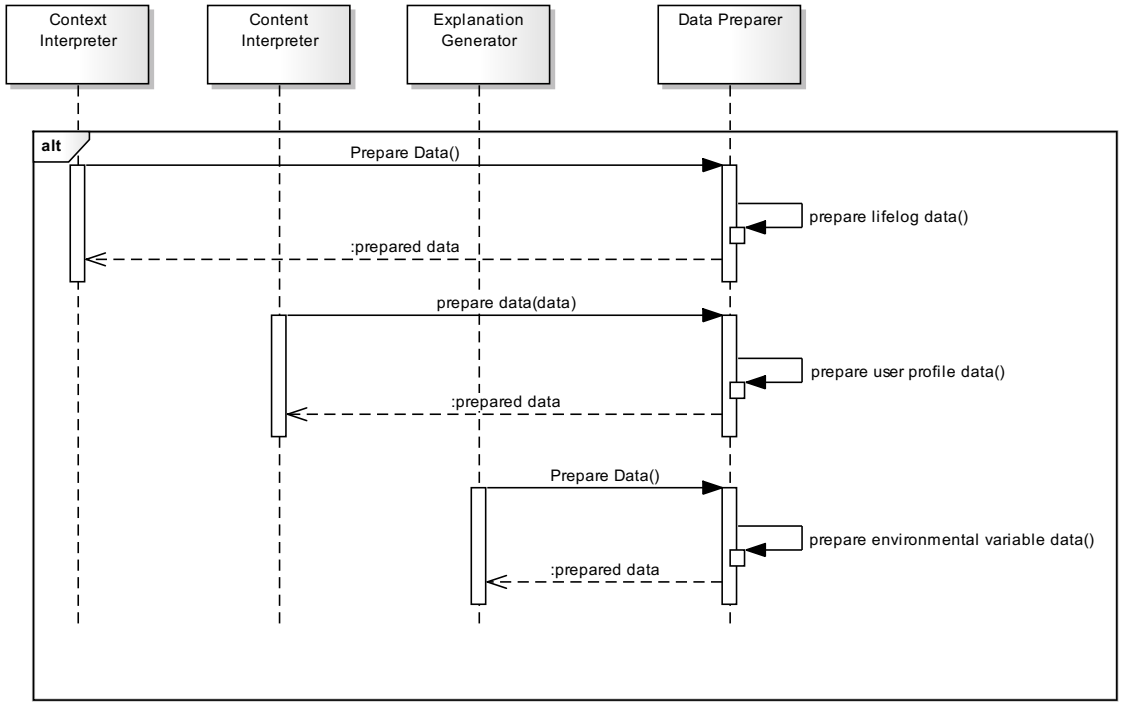
Sequence Diagram



Use Case ID:	SCL-UC-08		
Use Case Name:	Prepare data for interpreting recommendations		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 15, 2015	**Last Revision Date:**	July 28, 2015
Actors:	Context Interpreter		
Description:	The loaded data is prepared for interpretations according to different functions such as lifelog for contextual interpretations, user profile for content interpretations, and environmental variables for explanations.		
Trigger:	After loading data for interpretations		
Preconditions:	• Recommendation are built • Data is loaded		
Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations		
Normal Flow:	1. Context Interpreter sends data to Data Preparer for preparations 2. Data Preparer prepares lifelog data 3. Content Interpreter sends request to Data Preparer for preparing profile data 4. Data Preparer prepares profile data 5. Data Preparer prepares environmental variable data for the Explanation Generator		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	SCL-UC-07		
Frequency of Use:	Very frequent: at every service request		
Special Requirements:	N/A		

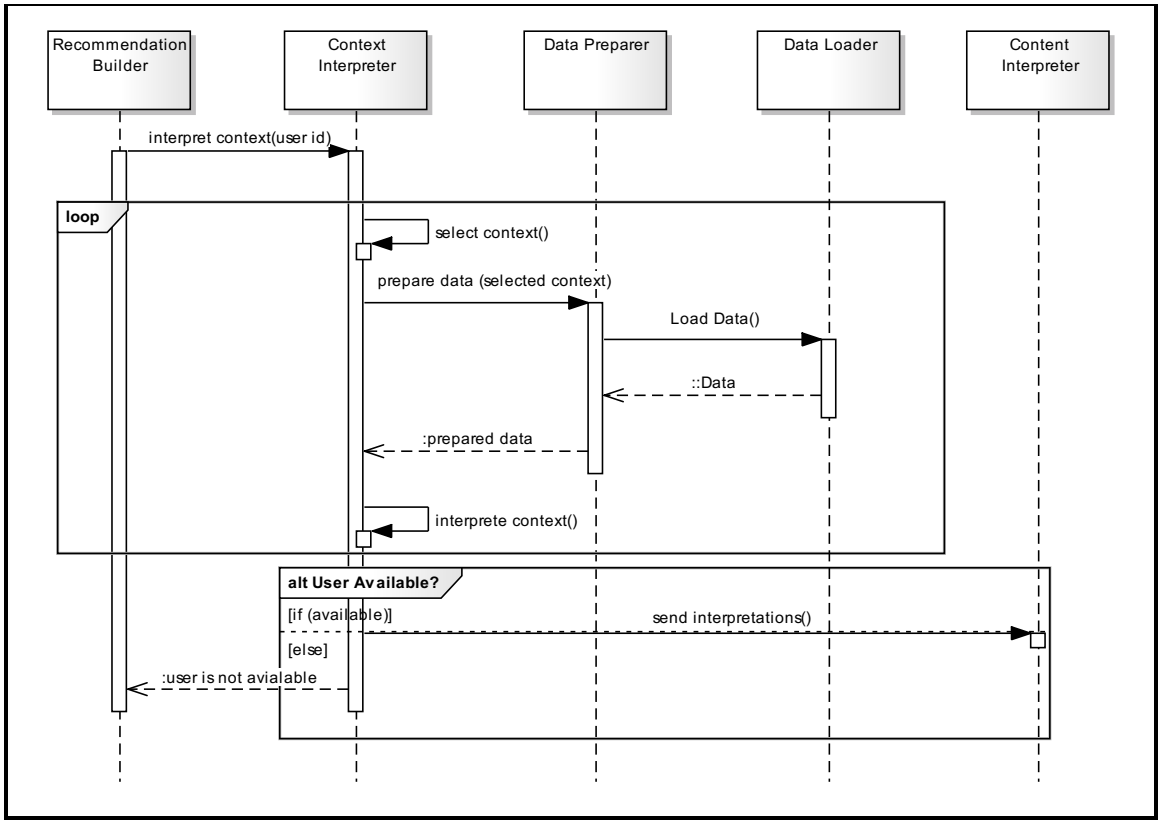
Assumptions:	N/A

Sequence Diagram



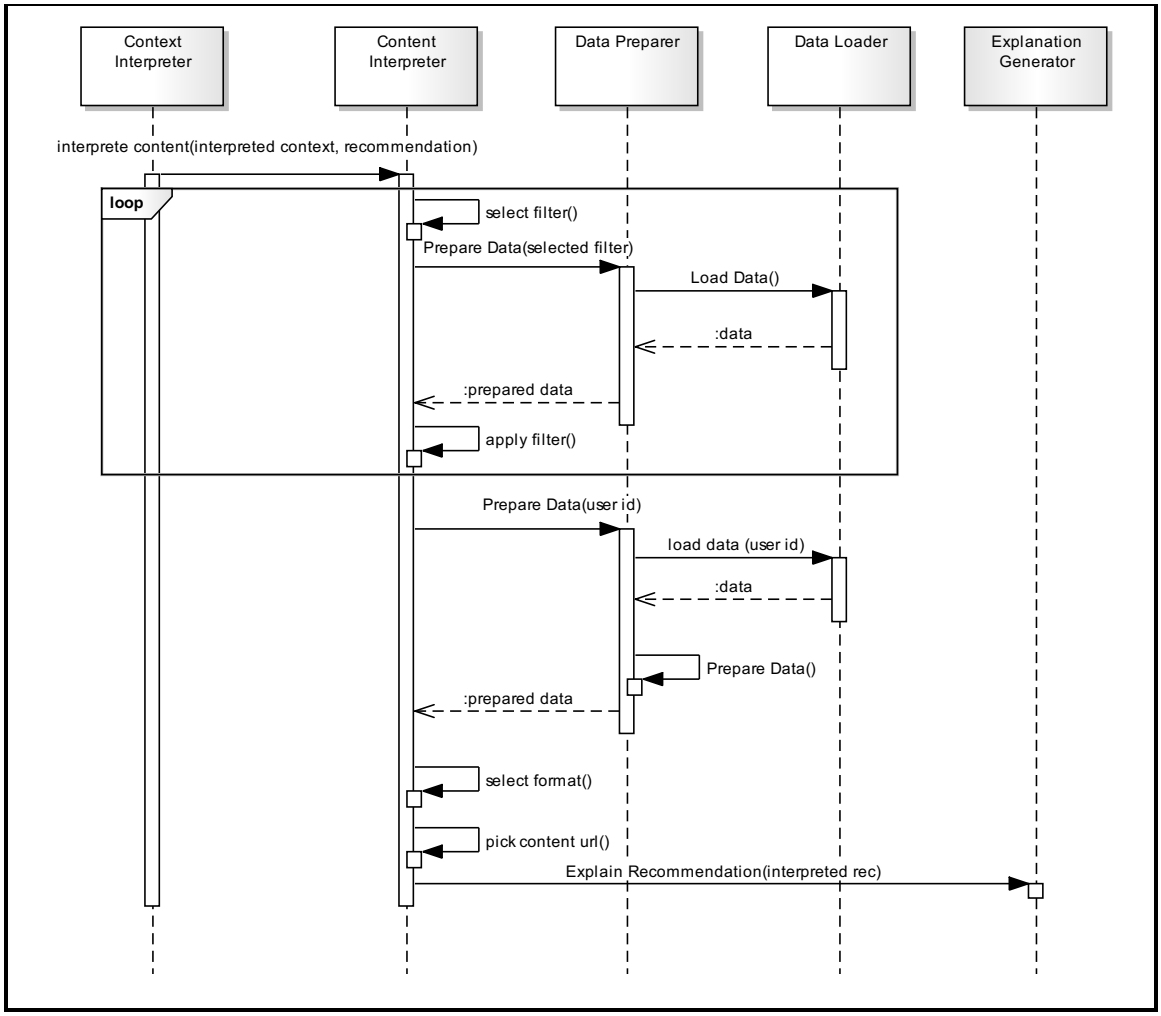
Use Case ID:	SCL-UC-09		
Use Case Name:	Interpret context		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 15, 2015	**Last Revision Date:**	July 28, 2015
Actors:	Build Recommendation		
Description:	The loaded data is prepared for interpretations according to different functions such as lifelog for contextual interpretations, user profile for content interpretations, and environmental variables for explanations.		

Trigger:	After loading data for interpretations
Preconditions:	Recommendation are built, and data is loaded
Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations
Normal Flow:	1. Context Interpreter receives request for from Recommendation Interpreter for context interpretation 2. Context Interpreter load and prepare data lifelog data (contextual data) for interpretations. 3. Context interpreter select a context 4. Context interpreter interprets the context 5. Repeat 2-4 until all applicable contexts interpreted 6. Context Interpreter receives the interpreted context 7. Context Interpreter sends the recommendations to content interpreter for interpreting the contents
Alternative Flows:	7a. if user is not available then the process is halt and message is sent to Recommendation Builder.
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SCL-UC-10		
Use Case Name:	Interpret contents		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali
Date Created:	July 15, 2015	**Last Revision Date:**	July 28, 2015
Actors:	Interpret Context		
Description:	The recommended contents of recommendations are difficult for user to understand. These contents need to be interpreted with support of multimedia contents.		
Trigger:	After interpretation of the context		
Preconditions:	Recommendations are generated, and context is interpreted		
Postconditions:	Recommendations are ready for explanation		

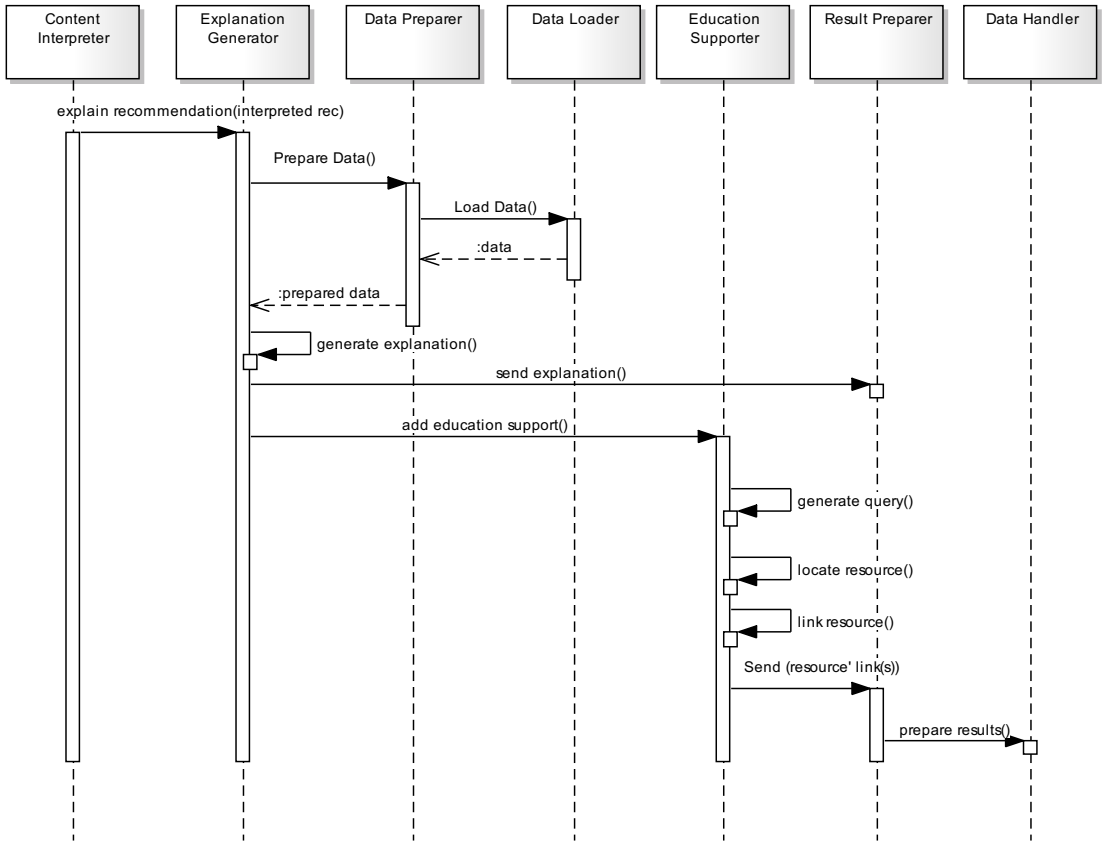
Normal Flow:	1. Context Interpreter sends the contextually interpreted recommendations to the content filterer. 2. Content interpreter perform the following tasks; a. Select appropriate filter b. Applies the filter 3. Step 2 is repeated for all filters 4. Content interpreter selects the appropriate format 5. Content interpreter adds the relevant url 6. Content interpreter forwards the format and filtered contents to explanation generator
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent: when recommendation are generated
Special Requirements:	The format should be defined in advanced based on the user special conditions
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	


Context Interpreter
Content Interpreter
Data Preparer
Data Loader
Explanation Generator
interprete content(interpreted context, recommendation)
loop
select filter()
Prepare Data(selected filter)
Load Data()
:data
:prepared data
apply filter()
Prepare Data(user id)
load data (user id)
:data
Prepare Data()
:prepared data
select format()
pick content url()
Explain Recommendation(interpreted rec)

Use Case ID:	SCL-UC-11		
Use Case Name:	Explain Recommendations		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali
Date Created:	July 15, 2015	**Last Revision Date:**	July 20, 2015
Actors:	Interpret content		
Description:	Usually user don't understand the contents of recommendations. To make them understandable the interpreted recommendations needs to be explained based on the user understandability.		
Trigger:	When contents are interpreted		
Preconditions:	Recommendations are interpreted		
Postconditions:	Recommendations are ready to deliver to the user		
Normal Flow:	1. Explanation generator receives the interpreted recommendations from content interpreter. 2. Explanation generator performs the following tasks; a. Select environment variable b. Generate explanation 3. Explanation Generator sends explained recommendation to educational support handler 4. Educational support handler performs the following tasks a. generate query b. locate resource c. link resource d. send resource link to interpreter 5. System sends explanation and educational resource links to result preparer.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	SCL-UC-08		
Frequency of Use:	Very frequent: when recommendations are interpreted		

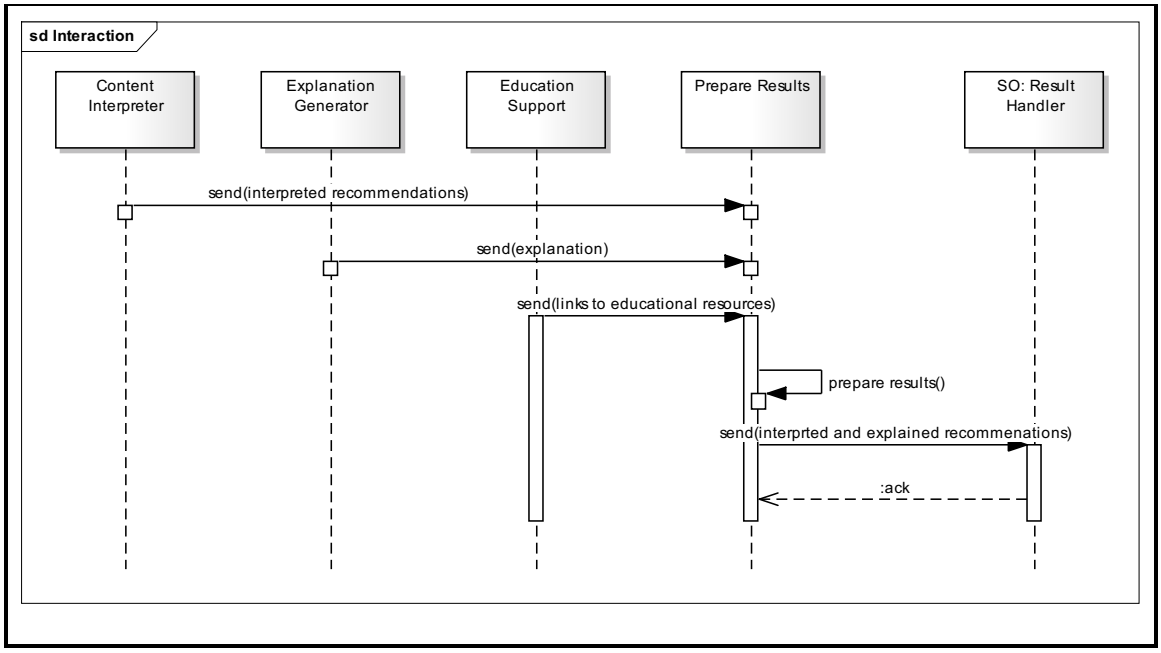
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Sequence Diagram



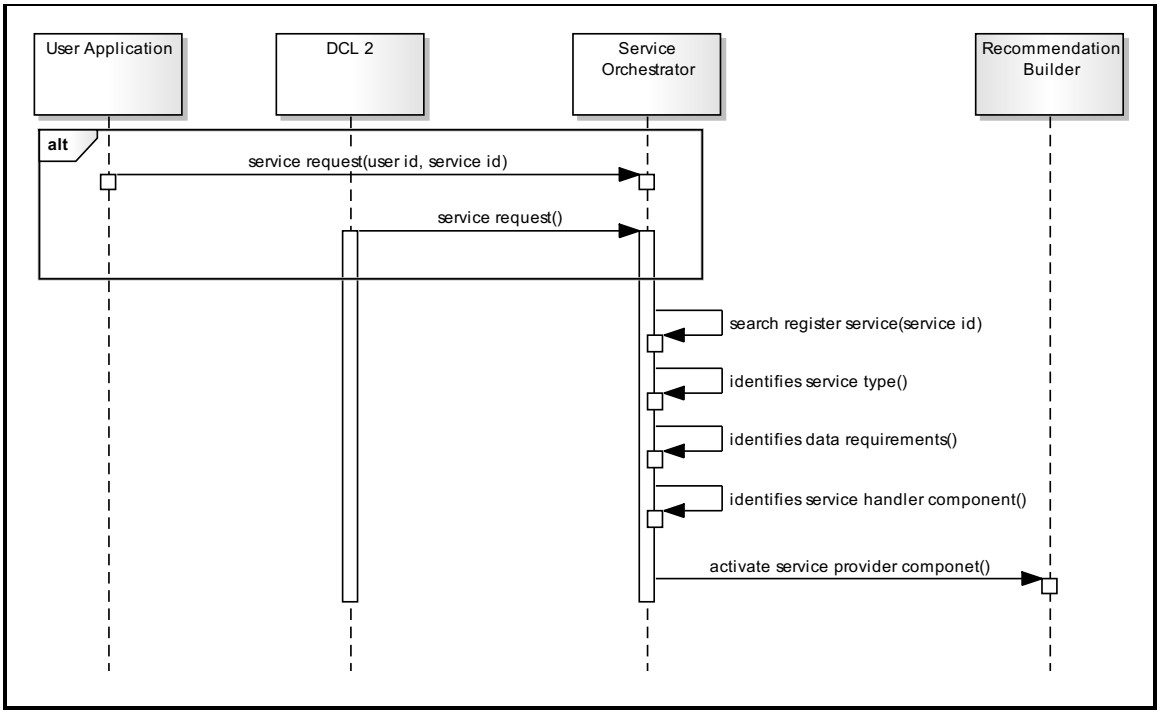
Use Case ID:	SCL-UC-12		
Use Case Name:	Prepare Results		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 5, 2015	**Last Revision Date:**	July 20, 2015

Actors:	Explain Recommendation (SCL-UC-11), Interpret Contents (SCL-UC-10)
Description:	This use case prepares the results accumulated from explanation generator and content interpreter and forwards to results handler of service orchestrator.
Trigger:	When recommendation is interpreted and explained
Preconditions:	The recommendation is interpreted and explained
Postconditions:	The results are forwarded to service orchestrator.
Normal Flow:	1. Result Preparer receives outputs from content interpreter and/or explanation generator as well as education support. 2. Result Preparer combines the received results 3. Result Preparer sends the results to result handler of service orchestrator
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Frequent
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SCL-UC-13		
Use Case Name:	Receive service request		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 5, 2015	**Last Revision Date:**	July 20, 2015
Actors:	User Application / SL		
Description:	Service orchestrator receives request from user application, or DCL for recommendation. Orchestrator parses the request and invokes required service of Mining Mind for responding.		
Trigger:	At the time of a request from the user application, or from mining mind generated events.		
Preconditions:	• User is registered with Mining Minds • Service is registered as mining Minds valid service • Service-data binding is specified in advance		
Postconditions:	The request is received, and recommendations are generated		

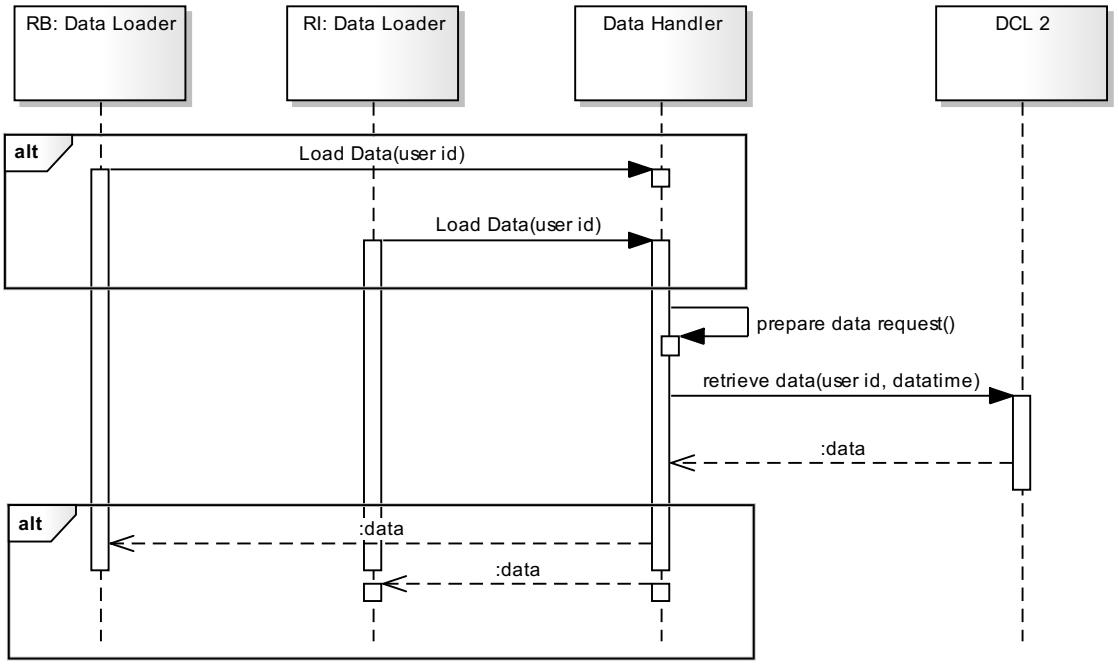
Normal Flow:	1. Service orchestrator receives the service request from user application 2. System parses the request a. Search for the registered service b. Identifies the service type c. Identifies data requirements of the service d. Identifies the appropriate handling module 3. Service Orchestrator passes the request to recommendation builder of SCL 2 to build the recommendation
Alternative Flows:	1.a Event handler of service orchestrator receives the request as an interrupt from DCL, whenever a situation occurs 4.a Step 2-4 of the normal flow are executed.
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Very frequent: at every service request
Special Requirements:	N/A
Assumptions:	Service orchestrator and DCL agreed on service contract.
Notes and Issues:	
Sequence Diagram	



Use Case ID:	SCL-UC-14		
Use Case Name:	Handle Data		
Created By:	Muhammad Afzal	**Last Updated By:**	Muhammad Afzal
Date Created:	July 28, 2015	**Last Revision Date:**	July 28, 2015
Actors:	DCL		
Description:	This use case receives data request from recommendation builder and recommendation interpreter. It makes request from DCL to get the data for requester.		
Trigger:	At data request time		
Preconditions:	Service Request has been received to service orchestrator		
Postconditions:	Data has been provided to requester		
Normal Flow:	1. Data Handler in Service Orchestrator received data loading request from recommendation builder 2. Prepare data request		

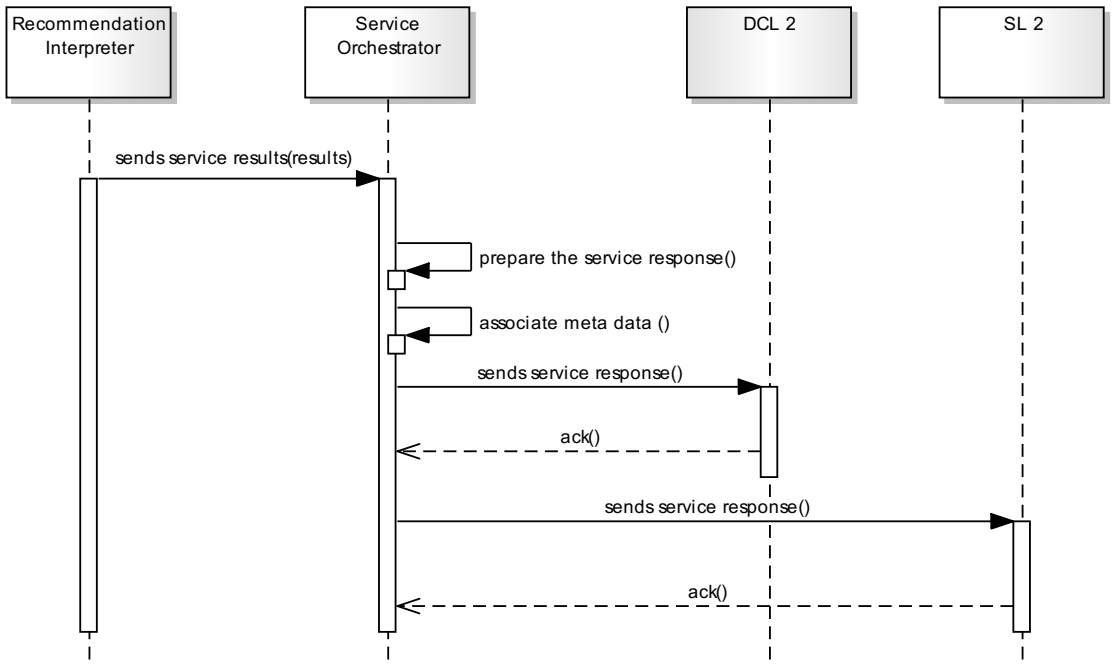
	3. Retrieve data from DCL 4. Send data to RB: Data Loader
Alternative Flows:	1a. Data Handler in Service Orchestrator received data loading request from recommendation interpreter Step 2-3 of normal flow 4a. Send data to RI: Data Loader
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Very Frequent: At every service request
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Sequence Diagram



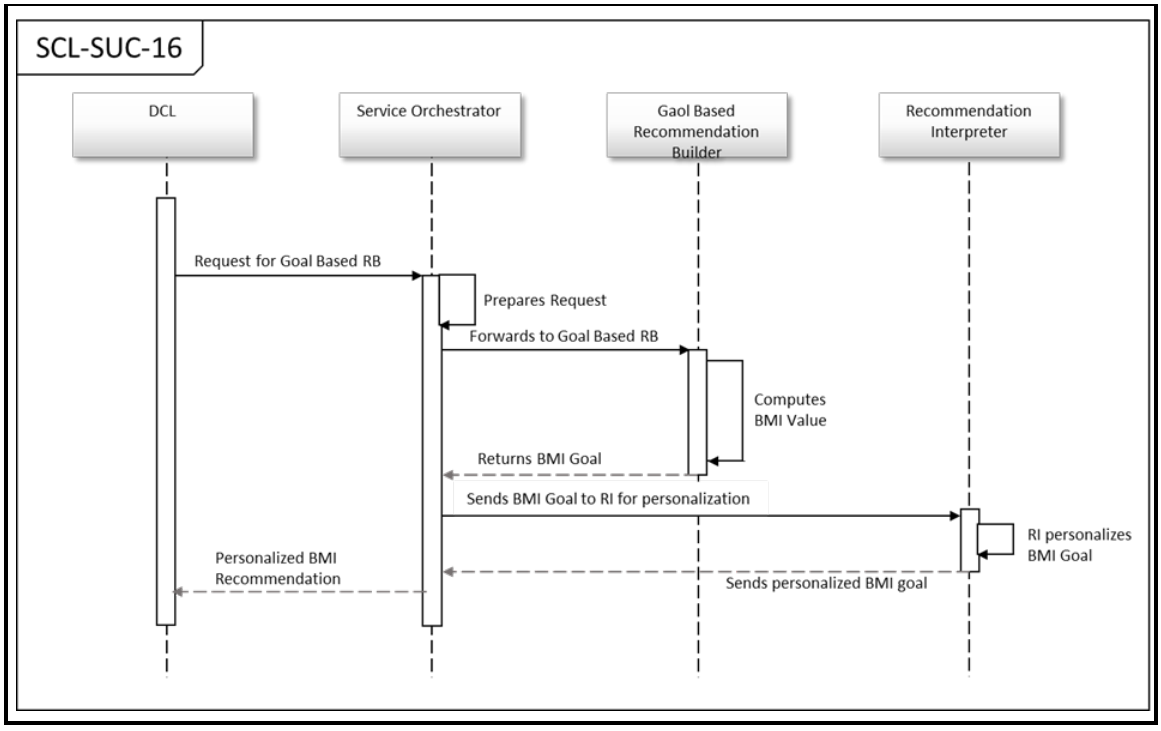
Use Case ID:	SCL-UC-15		
Use Case Name:	Deliver service results		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 5, 2015	**Last Revision Date:**	July 20, 2015
Actors:	Application / SL		
Description:	It is required to send request response to the service requester in the form of recommendation. Service orchestrator delivers the interpreted recommendation to user.		
Trigger:	At the time of completion of interpretations		
Preconditions:	Recommendations are generated and interpreted		
Postconditions:	Service results are successfully delivered to the requester and DCL for persistence		
Normal Flow:	1. Service orchestrator receives results from recommendation interpreter 2. System perform the following tasks; a. Prepares the response message b. Associate recommendations with service meta-data 3. Service Orchestrator sends recommendations to DCL for persistence 4. Service Orchestrator receives acknowledgement of storage 5. Service Orchestrator sends interpreted recommendations to SL 6. Service Orchestrator receives acknowledgement of receipt		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Very frequent: at every service request completion		

Special Requirements:	N/A
Assumptions:	Service orchestrator agreed on service contract with DCL and user application.
Notes and Issues:	N/A
Sequence Diagram	



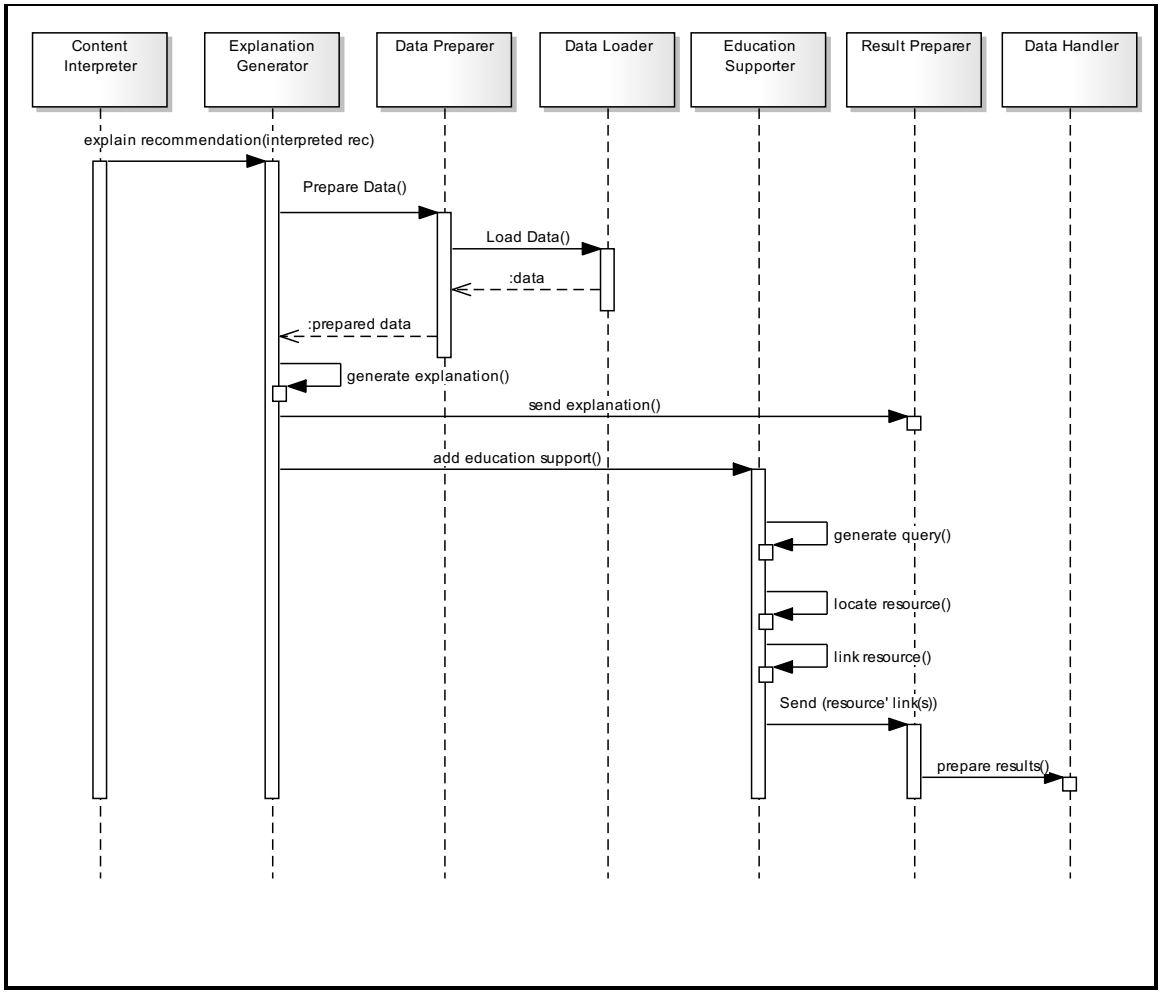
Use Case ID:	SCL-UC-16		
Use Case Name:	Goal-based recommendation Generation		
Created By:	Muhammad Sadiq	**Last Updated By:**	Muhammad Sadiq
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017
Actors:	DCL, Service Orchestrator, Recommendation Interpreter		
Description:	Goal Based recommender generates goal-based recommendation based on the user values		
Trigger:	DCL/User invokes goal-based recommender through service orchestrator for goal based personalized recommendation generation		
Preconditions:	Prepares request		

Postconditions:	Goal based recommendation is ready to be sent
Normal Flow:	1. User characteristic information e.g. height, weight etc. is received 2. Goal Based Reasoner computes user BMI 3. Computed BMI is compared with Standard BMI 4. The difference of computed and standard BMI is generated 5. Categorize user based on the computed BMI difference 6. Generate recommendation based on respective BMI status 7. Return recommended goal and associated information to service orchestrator 8. Service orchestrator sends the recommendation to RI for personalization of the recommendation 9. RI personalizes the generated recommendation and sends it back to service orchestrator 10. Service orchestrator return the generated personalized goal-based recommendation to DCL
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	
Frequency of Use:	Depending on situation.
Special Requirements:	N/A
Assumptions:	KCL 3.0 and RB 2 agree on common format of production rules.
Notes and Issues:	N/A
Sequence Diagram	



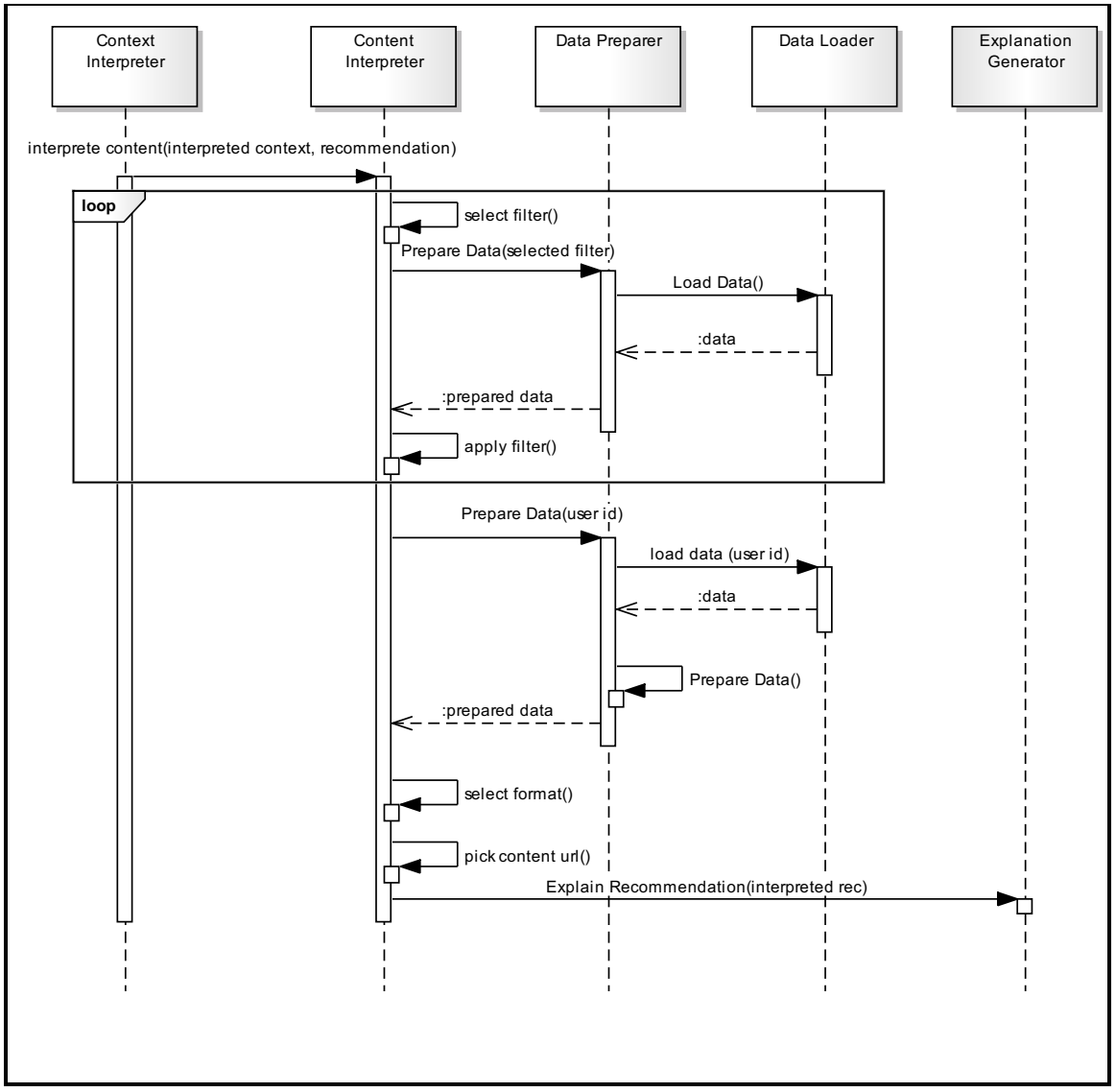
Use Case ID:	SCL-UC-17		
Use Case Name:	Provide Educational Support for user awareness		
Created By:	Imran Ali	**Last Updated By:**	Imran Ali
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017
Actors:	Build Recommendation (SCL-UC-4)		
Description:	Provide educational support to the user for first 4 weeks of the subscription period. Educational support consists of audio/visual aids and informative web resources		
Trigger:	After loading data for interpretations		
Preconditions:	Recommendation are built and data is loaded		
Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations		

Normal Flow:	5. Recommendation Interpreter receives request from Recommendation Builder 6. Recommendation Interpreter extracts users information from lifelog 7. Recommendation Interpreter evaluates the matching educational contents for the target user audience 8. Recommendation Interpreter prepares the required educational contents 9. Recommendation Interpreter sends the required educational content unit to result preparer
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Thrice a day: Time-based
Special Requirements:	N/A
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A
Sequence Diagram	



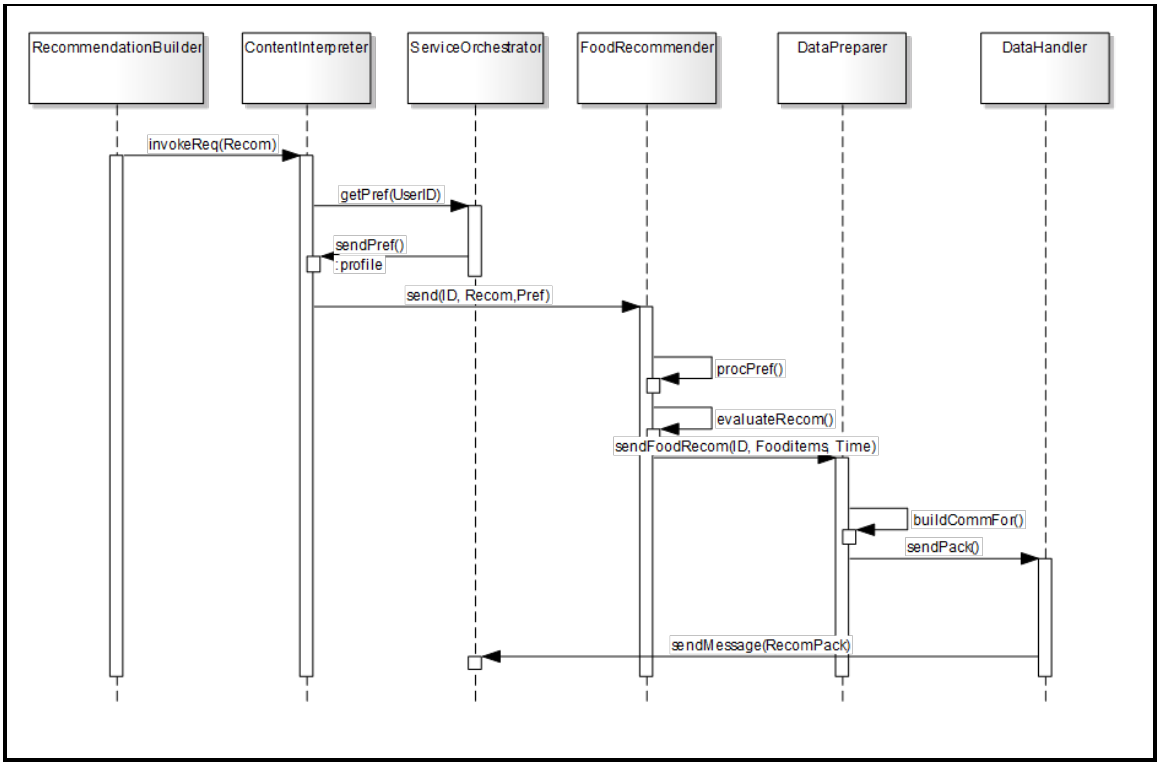
Use Case ID:	SCL-UC-18		
Use Case Name:	Location- and weather-based Personalized recommendation		
Created By:	Imran Ali	**Last Updated By:**	Imran Ali
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017
Actors:	Build Recommendation (SCL-UC-4)		
Description:	Interpret recommendations for environmental contexts and provide the required information in an structured form to SL		
Trigger:	After loading data for interpretations		

Preconditions:	Recommendation are built, and data is loaded
Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations
Normal Flow:	1. Recommendation Interpreter receives physical activity-based recommendation 2. Recommendation Interpreter assess the user interpretability 3. In case of availability Recommendation Interpreter evaluates the contextual viability of the recommendation 4. Recommendation Interpreter sends the recommendation for education support 5. Recommendation Interpreter prepares educational unit for the target user is prepared 6. Final recommendation package is sent to Result Preparer
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A
Sequence Diagram	



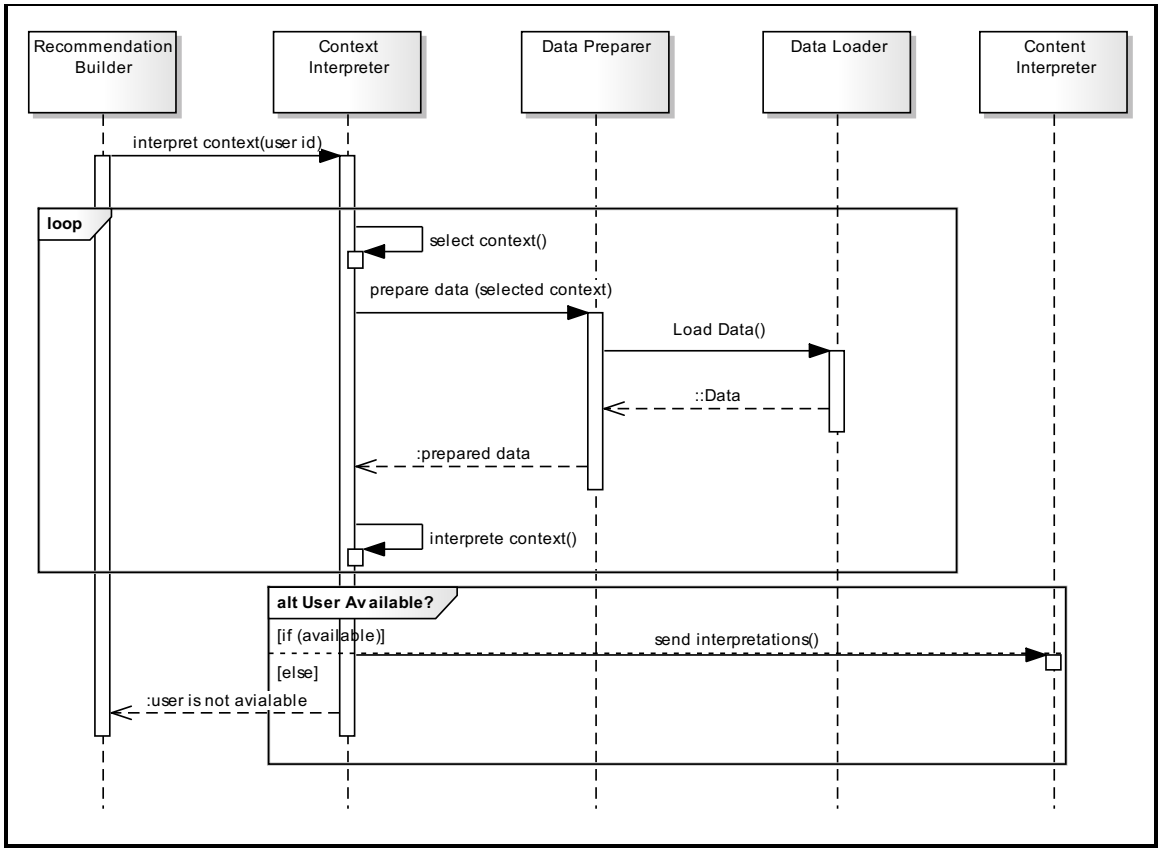
Use Case ID:	SCL-UC-19		
Use Case Name:	User Preference-based refined recommendations		
Created By:	Imran Ali	**Last Updated By:**	Imran Ali
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017
Actors:	Build Recommendation (SCL-UC-4)		

Description:	Interpret Recommendations based on User's Preferences for Food-based Recommendation
Trigger:	After loading data for interpretations
Preconditions:	Recommendation are built, and data is loaded
Postconditions:	The user profile and lifelog data is prepared and is available for interpretations
Normal Flow:	1. Recommendation Interpreter receives food-based recommendation 2. Recommendation Interpreter evaluates the targeted user group 3. Recommendation Interpreter retrieves preferences of the user 4. Recommendation Interpreter matches user preferences with the targeted food-categorizes (menu-sets) 5. Recommendation Interpreter finalizes the menu-set selection 6. Final recommendation package is sent to Result Preparer
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SCL-UC-20		
Use Case Name:	Provide Alternative Recommendation		
Created By:	Imran Ali	**Last Updated By:**	Imran Ali
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017
Actors:	Build Recommendation (SCL-UC-4)		
Description:	Interpret Recommendation for special condition. An alternative recommendation is generated when the provided recommendation is not deemed viable for the user		
Trigger:	After loading data for interpretations		
Preconditions:	Recommendation are built, and data is loaded		
Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations		

Normal Flow:	1. Recommendation Interpreter receives physical activity-based recommendation 2. Recommendation Interpreter evaluates the availability of the user 3. Recommendation Interpreter evaluates the special user conditions 4. Recommendation Interpreter decides of the need for alternative recommendation 5. Educational contents are prepared for the final recommendation 6. Final recommendation package is sent to Result Preparer
Alternative Flows:	1.1 Stall recommendation is user is not available
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A
Sequence Diagram	


Recommendation Builder
Context Interpreter
Data Preparer
Data Loader
Content Interpreter
interpret context(user id)
loop
select context()
prepare data (selected context)
Load Data()
::Data
:prepared data
interprete context()
alt User Available?
[if (available)]
send interpretations()
[else]
:user is not avialable

5. Supporting Layer

5.1 Functional Requirements

Requirements #ID	Description
SL-FR-01	The platform shall retrieve the user profile information, context of use and device information for adaptation of the user interface
SL -FR-02	The platform shall collect the user data such as user feedback and user observational data for the enhancement of user interface
SL -FR-03	The platform shall utilize the user profile data, life-log and raw sensory data for analytics

5.2 Non-functional Requirements

Requirements #ID	Description
SL-NFR-01	The platform shall provide user interface that is easy to use and intuitive

5.3 Terms and Definitions

Terms	Description
DCL	Data Curation Layer
SCL	Service Curation Layer
ICL	Information Curation Layer
KCL	Knowledge Curation Layer
SL	Supporting Layer
Self-report	A method which involves asking a participant about their feelings, attitudes, beliefs and so on
User Capabilities	User cognitive, perceptual and physical characteristics
Context of use	Environmental variables and low-level context (location)
Device Characteristics	Screen size, resolution, memory, and battery
UI Adaptation	The changes in user interface
Observational Data	User interaction data with the user interface

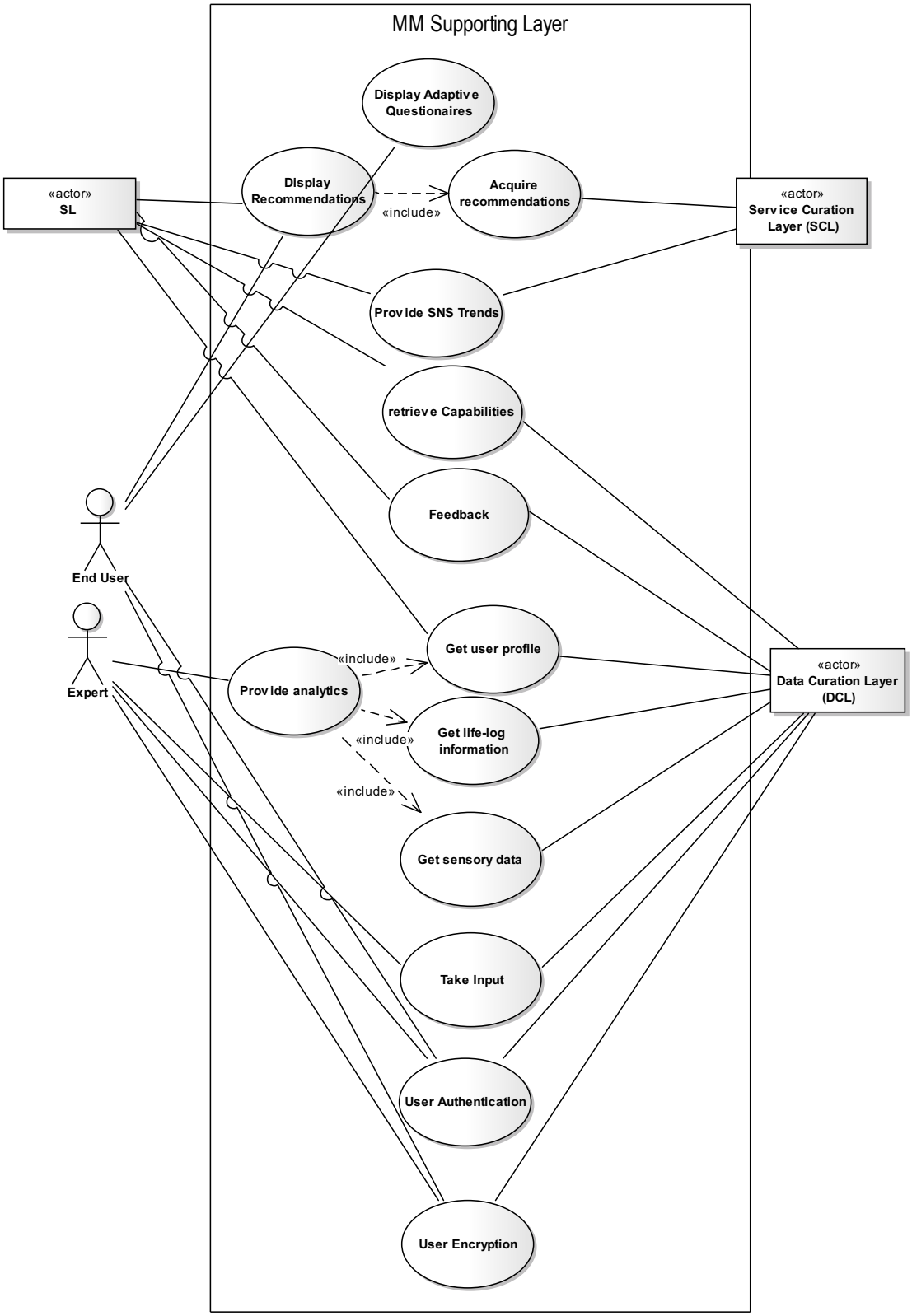
User Experience	User perception, satisfaction about the user interfaces
User Profile	User related data that do not change frequently

5.4 Use Cases

5.4.1 List

Requirements #ID	Description
SL-UC-01	User registration
SL-UC-02	Retrieve capabilities for user interface adaptation
SL-UC-03	Mapping the user capability information into model
SL-UC-04	Adapt user interface based on user profile, context and device
SL-UC-05	User capabilities collection
SL-UC-06	Self-reporting user experience measurement
SL-UC-07	Collect and analyze observational data
SL-UC-08	Acquire Recommendations for displaying to end user
SL-UC-09	Feedback Collection
SL-UC-10	Retrieve capabilities for context-based services
SL-UC-11	Map Request to Query
SL-UC-12	Transform Data
SL-UC-13	Classify Data
SL-UC-14	Analyze Data
SL-UC-15	Display Analytics
SL-UC-16	Take Expert Input
SL-UC-17	Display Adaptive Questions

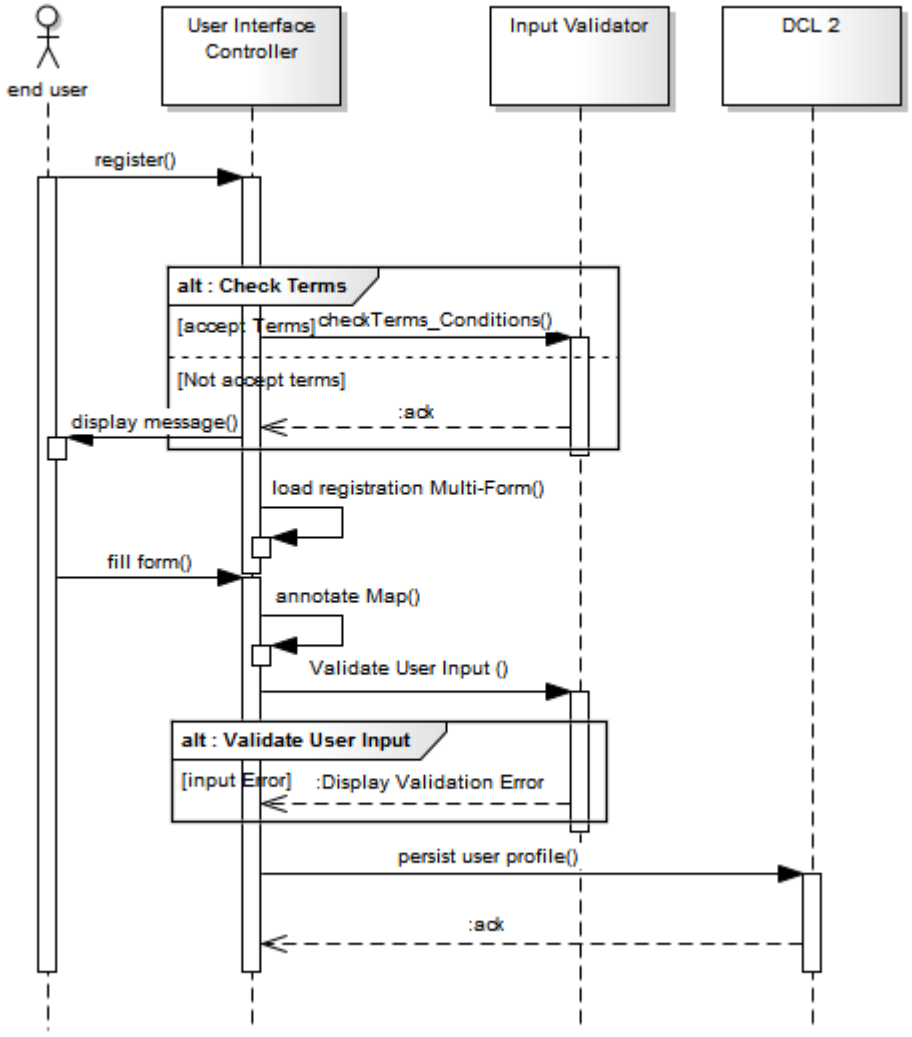
5.4.2 Diagram


MM Supporting Layer
Display Adaptive Questionaires
Display Recommendations
Acquire recommendations
«include»
«actor»
SL
«actor»
Service Curation Layer (SCL)
Provide SNS Trends
retrieve Capabilities
Feedback
End User
Expert
Get user profile
Provide analytics
«include»
Get life-log information
«include»
«include»
Get sensory data
Take Input
User Authentication
User Encryption
«actor»
Data Curation Layer (DCL)

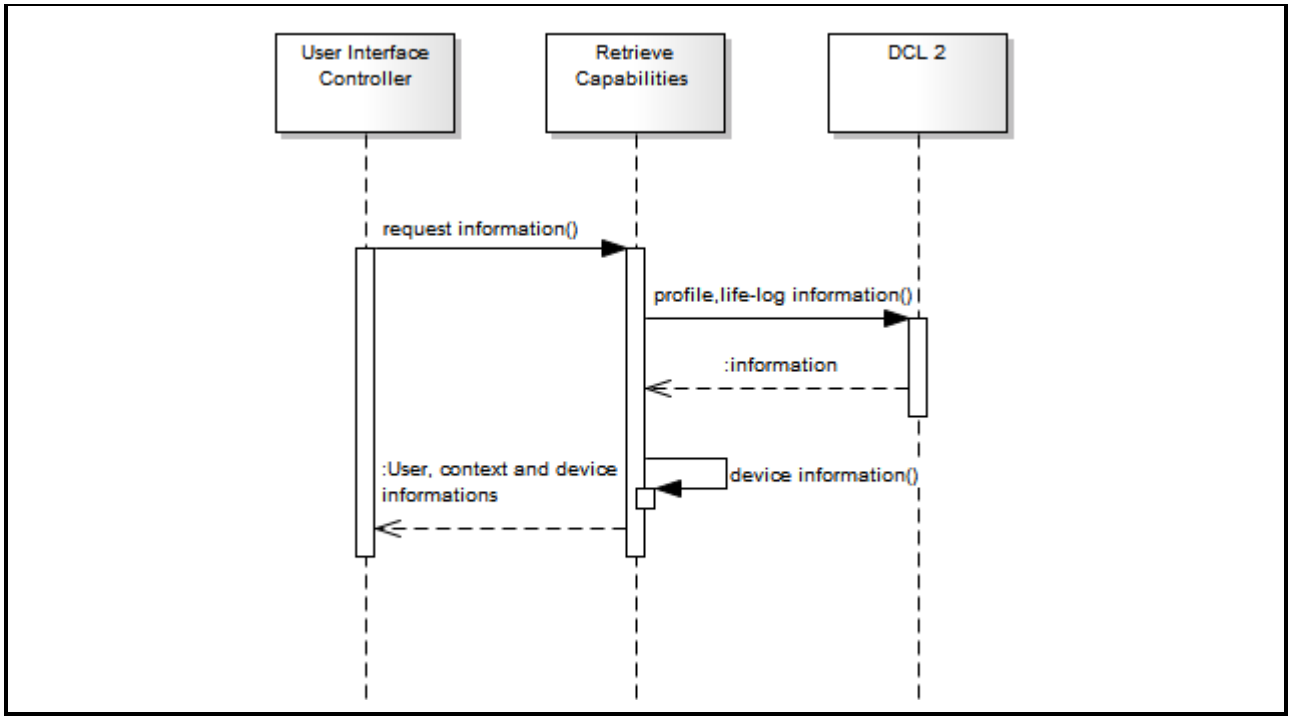
5.4.3 Description

Use Case ID:	SL-UC-01		
Use Case Name:	User registration		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	Primary: End-user Secondary: DCL		
Description:	This use case is for the user registration. A user must register with the MM app before they are able to use it. Registration primarily consists of entering an email address for verification and creating a password. All basic demographics, account, activity level, user interest information, and personalized map information shall be collect from user and persist in DCL.		
Trigger:	End user		
Preconditions:	The non-register user asked the application to register to it.		
Postconditions:	The user successfully registered to the application and can access its functionality		
Normal Flow:	1. The user starts registration of the new account by pressing the “Si g up” button on the application first screen. 2. Then **Terms & Conditions** page displayed 3. The application will display the multi-step registration form with e mpty fields for the account and user profile. 4. The user annotates the map for personalization by selecting differ ent locations of his interest. 5. Validate User Input • The application will automatically validate all the user inp ut for all the required fields • The user cannot proceed until providing the correct data. 6. The user can press “Submit” button and the new account data wil l be persisted to the DCL.		
Alternative Flows:	2a. In step 2. If the user Agree with Terms & Conditions • User is allowed to the next step by click on agreed term & con dition checkbox. 2b. In step 2. If user not agree with terms & conditions, then • User is redirect to the first screen.		

	3a. in step 3. Display Validation Error • If the validation failed, then the validation icon will be displayed nearby the wrong field and there will be validation message.
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	When user first time use the system [low]
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	

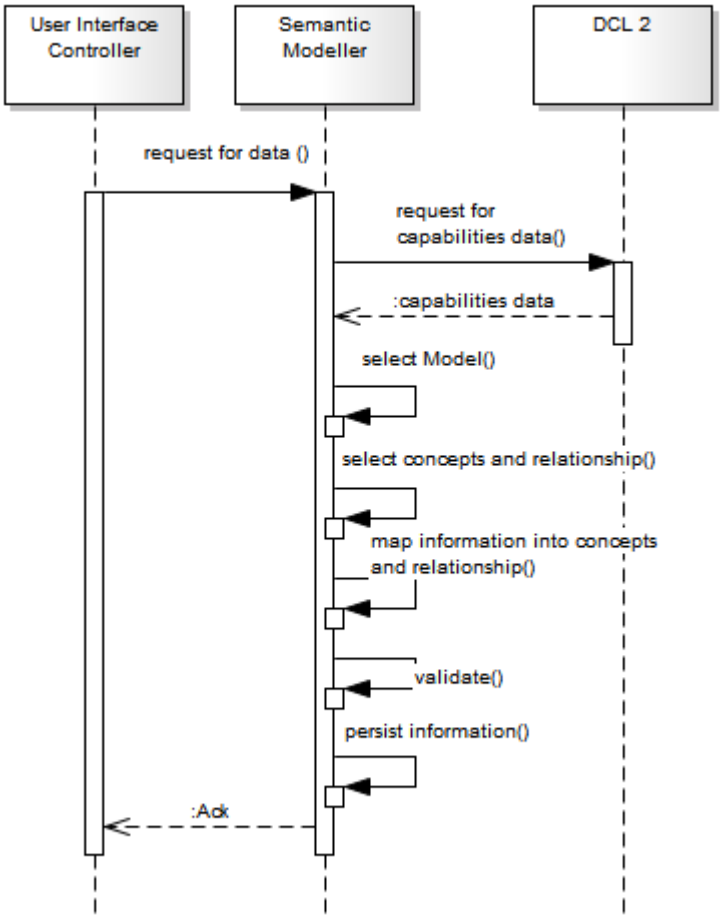


Use Case ID:	SL-UC-02		
Use Case Name:	Retrieve capabilities for user interface adaptation		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	DCL		
Description:	This use case focuses on the retrieval of the capabilities for user interface adaptation. The capabilities include user profile information, context information and device information. It is utilized for adaptation based on changes or observational data.		
Trigger:			
Preconditions:	The DCL provide the access to required information		
Postconditions:	All required capabilities are successfully collected.		
Normal Flow:	1. SL generates request for user, device, and context information collection from DCL 2. This information is utilized for the adaptation of the user interf aces 3. The adaptation is based on changes in user profile, context info rmation or collected observational data		
Alternative Flows:	N/A		
Exceptions:	If there is not capabilities information, then the default user interfaces will be displayed.		
Includes:	N/A		
Frequency of Use:	Always when the application is running [High]		
Special Requirements:	N/A		
Assumptions:	The capabilities information should be available with the DCL.		
Notes and Issues:	N/A		
Sequence Diagram			

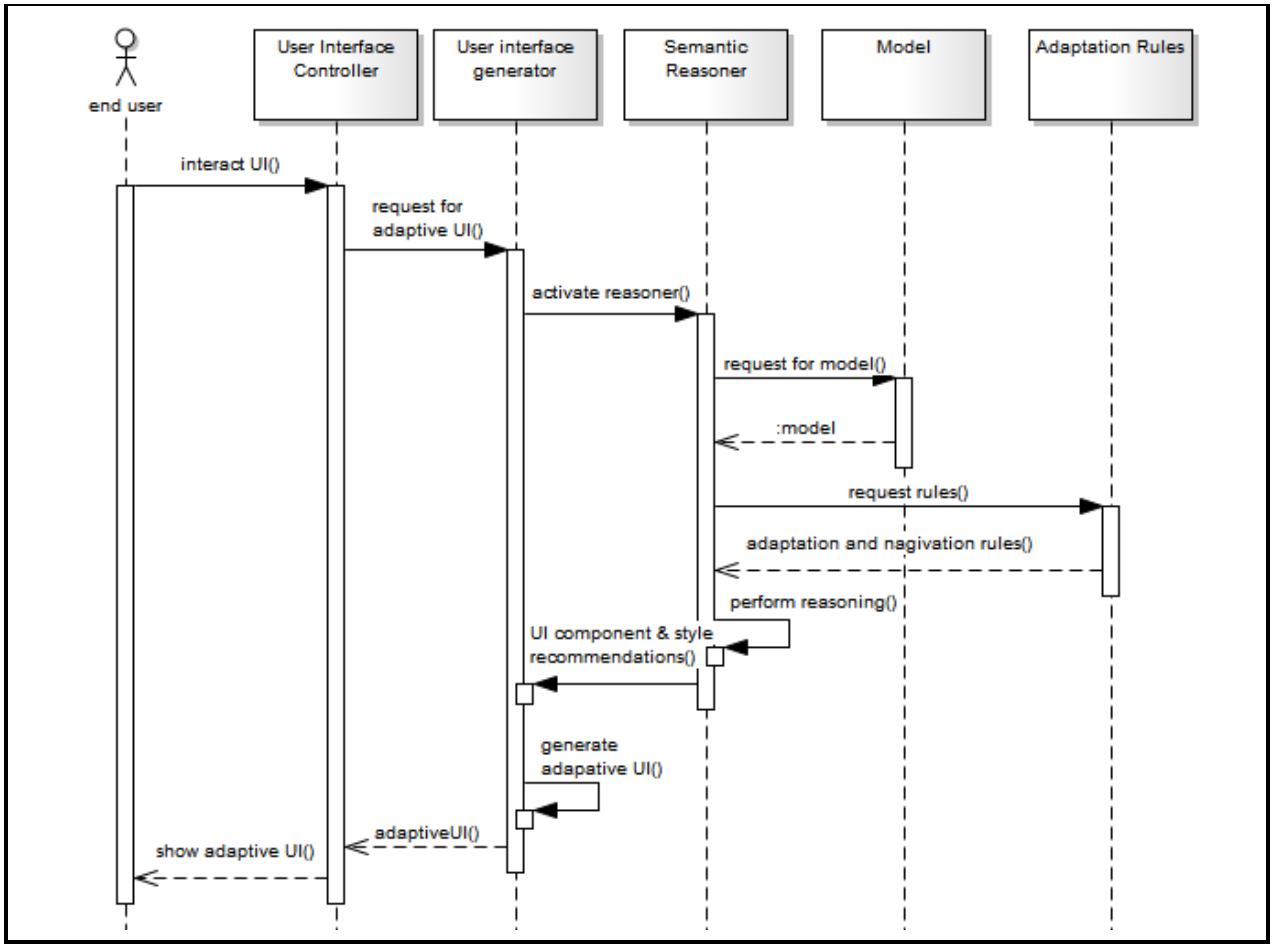


Use Case ID:	SL-UC-03		
Use Case Name:	Mapping the user capability information into model		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	DCL		
Description:	The collected capabilities information from DCL.0 shall be mapped against the hierarchical structure of the model		
Trigger:	SL initiate communication with DCL.		
Pre-conditions:	• User is a registered client of MM platform • Updated user profile must be available		
Post-conditions:	1. User profile and environmental variables are received by UI/UX 2. All collected variables are successfully mapped and validated		

Normal Flow:	1. UI/UX send request to DCL.0 for environmental variables (e.g. , temperature, weather, time, noise, light level etc.) and user profile variables (e.g. uid, name, age, perceptual information) 2. DCL sent back the requested variables. 3. The semantic modeler maps the data to model 4. The mapped information is persisted in model.
Alternative Flows:	N/A
Exceptions:	
Includes:	**SL-UCS-01**
Frequency of Use:	Always when the application is running [High]
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	

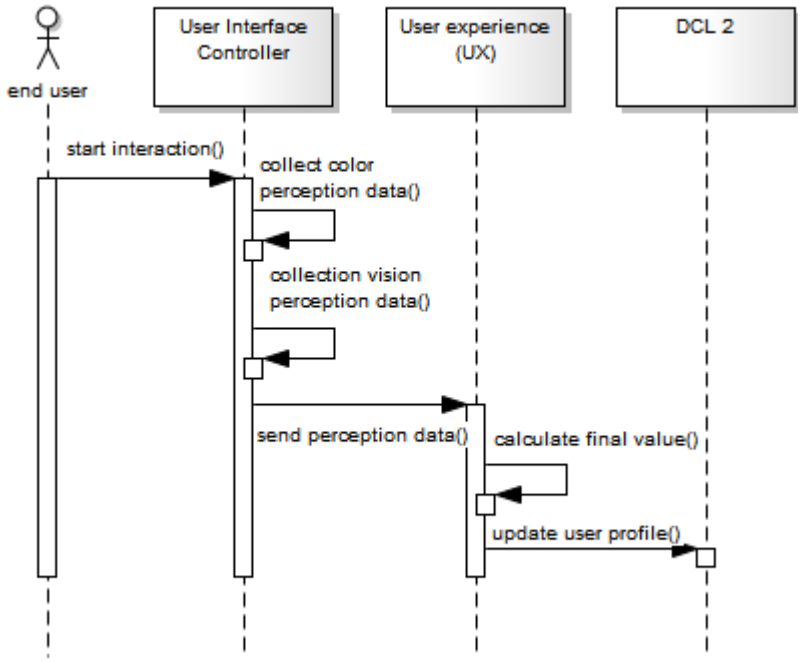


Use Case ID:	SL-UC-04		
Use Case Name:	Adapt user interface based on user profile, context and device		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	DCL, end user		
Description:	The collected information of user profile, context of use and device characteristics from DCL results in adaption of the user interface		
Trigger:	End user start interacting with user interface		
Pre-conditions:	• The user profile and context of use and device data has been collected by UI/UX Authoring tool		
Post-conditions:	• Adaptive UI rendered/generated based on collected informa tion		
Normal Flow:	1. User request for personalize user interface to UI/UX. 2. UI/UX send request to adaptation engine for generating the personalized user interface based on user, context and devic e information. 3. Adaptation engine perform reasoning based on pre-defined adaption and navigations rules. 4. Reasoner recommend the user interface components and its styles to generate the UI. 5. Then personalized generated UI is displayed to end user.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	When user interact with the system		
Assumptions:	N/A		
Notes and Issues:	N/A		
Sequence Diagram			



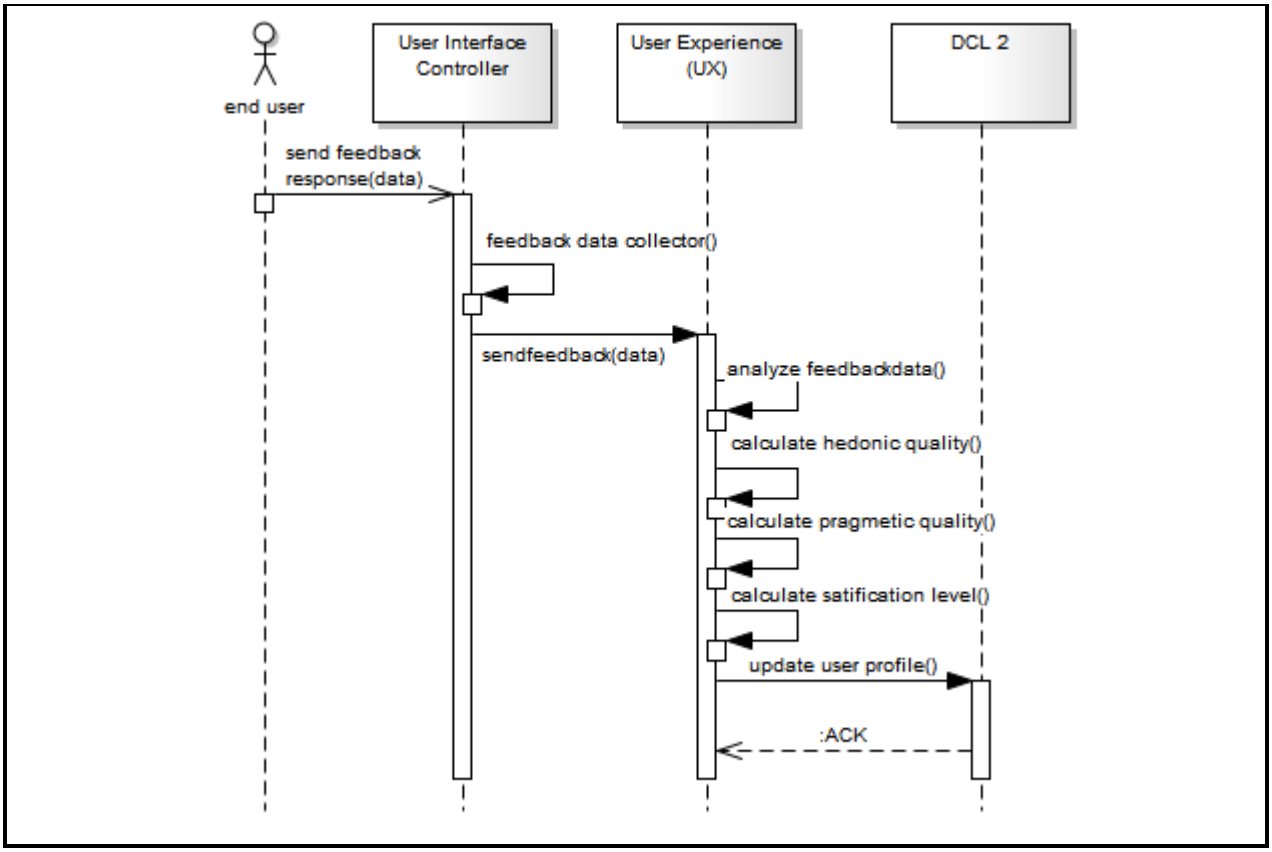
Use Case ID:	SL-UC-05		
Use Case Name:	User capabilities collection		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	End user, DCL		
Description:	The UI/UX collects the user capabilities information's by analyzing the user perception such as user visual and color perception		
Trigger:	User uses the tools for collection		
Pre-conditions:	• Perception collection tools are installed		
Post-conditions:	• User perceptual information successfully collected and upda te information in user profile DCL		

Normal Flow:	1. User select the tools for color and visual perception and interact with it accordingly. 2. Tools acquire its interaction data in order to find the user perceptions 3. User experience calculate its final value. 4. Final values are stored in user profile DCL.
Alternative Flows:	
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	When user interact with the system
Special Requirements:	
Assumptions:	Service contract between SL and DCL is defined
Notes and Issues:	N/A
Sequence Diagram	



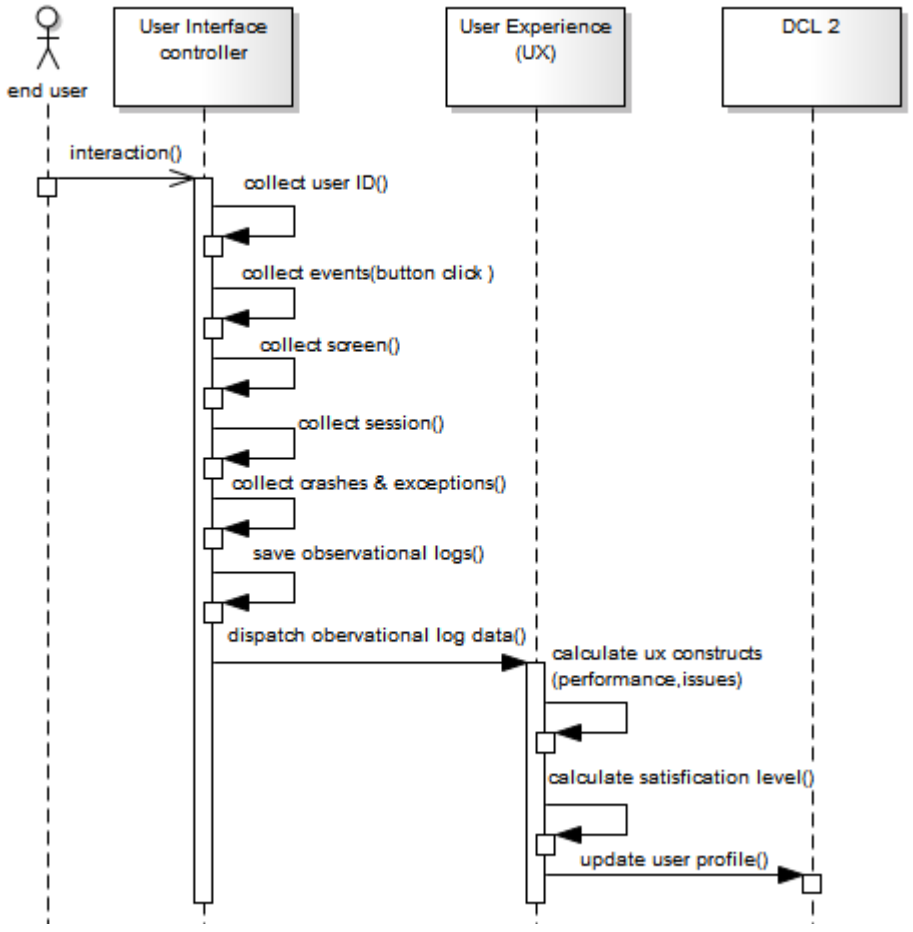
Use Case ID:	SL-UC-06		
Use Case Name:	Self-reporting user experience measurement		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan

Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	End user, DCL		
Description:	The UI/UX shall collect feedback about how users feel about the system during or after use by self-reporting method.		
Trigger:	End user		
Pre-conditions:	• Self-reporting questionnaire already exist		
Post-conditions:	• The feedback is successfully collected		
Normal Flow:	1. The end user provide feedback using the questionnaire. 2. The feedback is sent to user experience in order to evaluate th e user response. 3. user experience variables such as usability, pleasure, beauty a re calculated based on filled questions 4. The UI/UX update the calculated variables values in user profil e by sending request to DCL		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	When user interact with the system		
Special Requirements:	N/A		
Assumptions:	N/A		
Notes and Issues:	N/A		
Sequence Diagram			



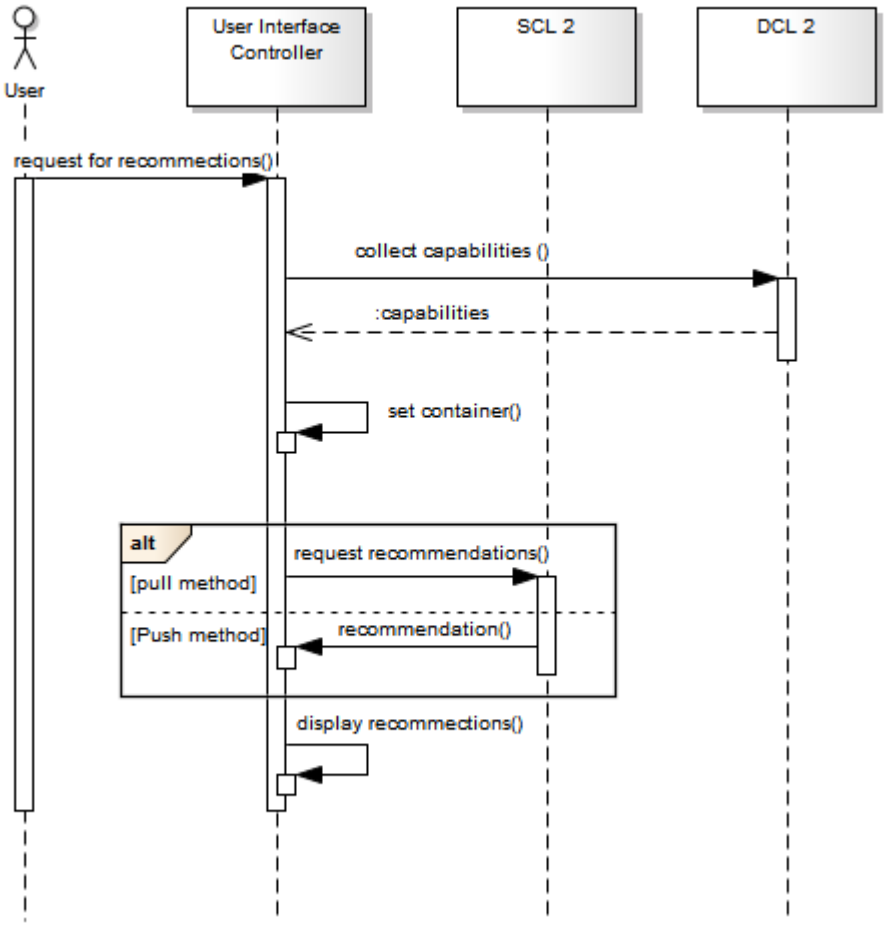
Use Case ID:	SL-UC-07		
Use Case Name:	Collect and analyze observational data		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	End user, DCL		
Description:	The UI/UX shall identify areas of improvement and maximize the user interaction by analyzing the user interaction with app.		
Trigger:	Initiated by end user		
Pre-conditions:	• Analytics tracker is already installed		
Post-conditions:	• Observational data are successful collected and analyzed for u ser experience measurement		
Normal Flow:	1. Analytics collector collect the user interaction data such as us er ID, event, session, screen, crashes & exceptions, and user ti mings 2. The collected data is stored locally before being dispatched		

	3. Data is dispatched for user experience measurement from the app for every 30 minutes 4. the pragmatic quality such as usability-(e.g. performance, issues) are calculated in order to find the user experience (UX) 5. UX quality variables are sent to DCL2 for storage/updating in user profile.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Frequent, request by SL
Special Requirements:	N/A
Assumptions:	Service contract between DCL and SL is defined
Notes and Issues:	N/A
Sequence Diagram	

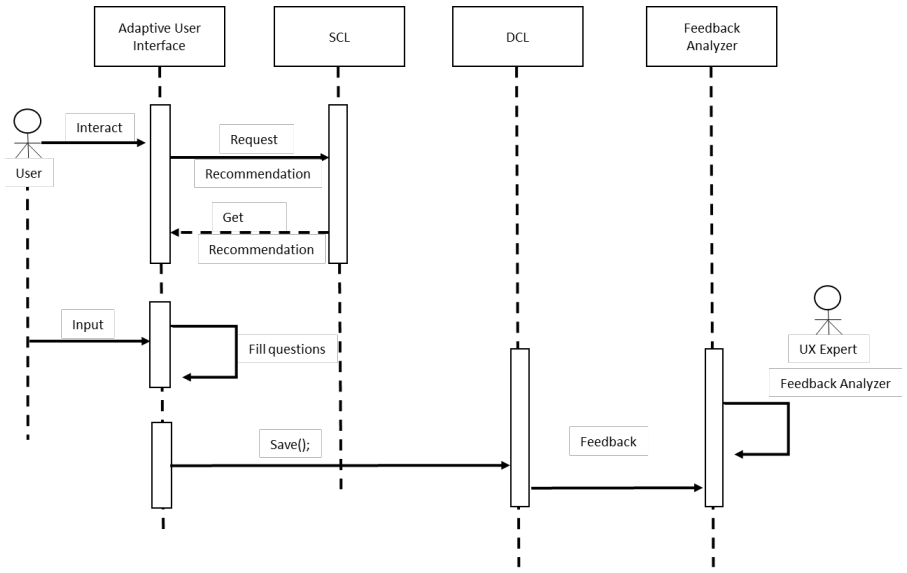


Use Case ID:	SL-UCS-08		
Use Case Name:	Acquire Recommendations for displaying to end user		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	Primary: End-user Secondary: SCL, DCL		
Description:	This use case collects the recommendations generated by SCL and displays it on the user interface for the end users. The provided recommendations are displayed according to user capabilities, context of use, and device characteristics. This information is obtained from the DCL.		
Trigger:	SCL push the recommendations to the App or end-user send request for recommendations		
Preconditions:	End-user subscribes to particular services		
Postconditions:	All recommendations are successfully displayed according to user capabilities, context, and device characteristics.		
Normal Flow:	1. SCL generate the recommendations and provide it to user interface 2. The SCL recommendations are acquired by the SL 3. SL investigates the user capabilities, context of use, and device characteristics by obtaining from DCL 4. The recommendation is displayed in graphical user interface based on collected capabilities of user, context and device information.		
Alternative Flows:	2a. In step 2. The SCL recommendations are acquired by the SL 1. user request for recommendations (pull method) 2b. In step 2. The SCL recommendations are acquired by the SL 1. SL push recommendations to App based on situations		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Whenever the recommendations are generated by SCL [Medium]		
Special Requirements:	N/A		

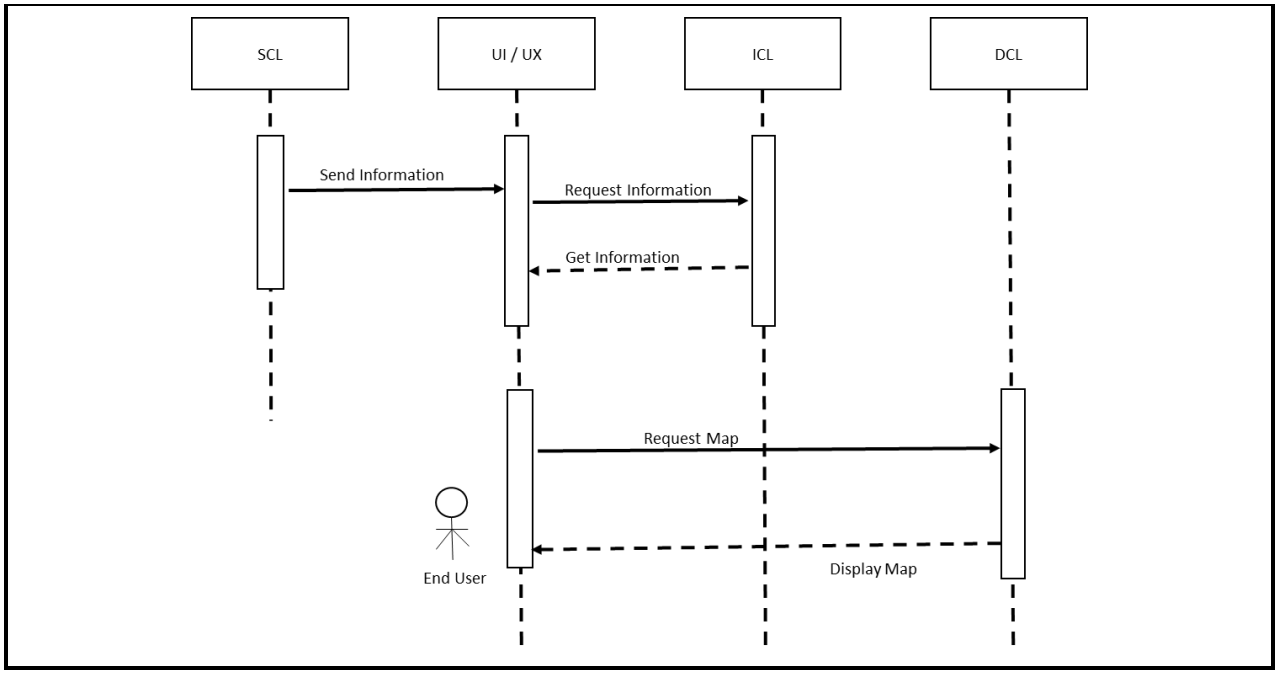
Assumptions:	The user profile data and context information should exist in the DCL
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SL-UCS-09		
Use Case Name:	Feedback Collection		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan
Date Created:	March 09 2017	**Last Revision Date:**	01 July 2018
Actors:	Primary: End-user Secondary: DCL 3		
Description:	The UI/UX shall collect feedback about how users feel about recommendations and additional questionnaires in a predefined time period		

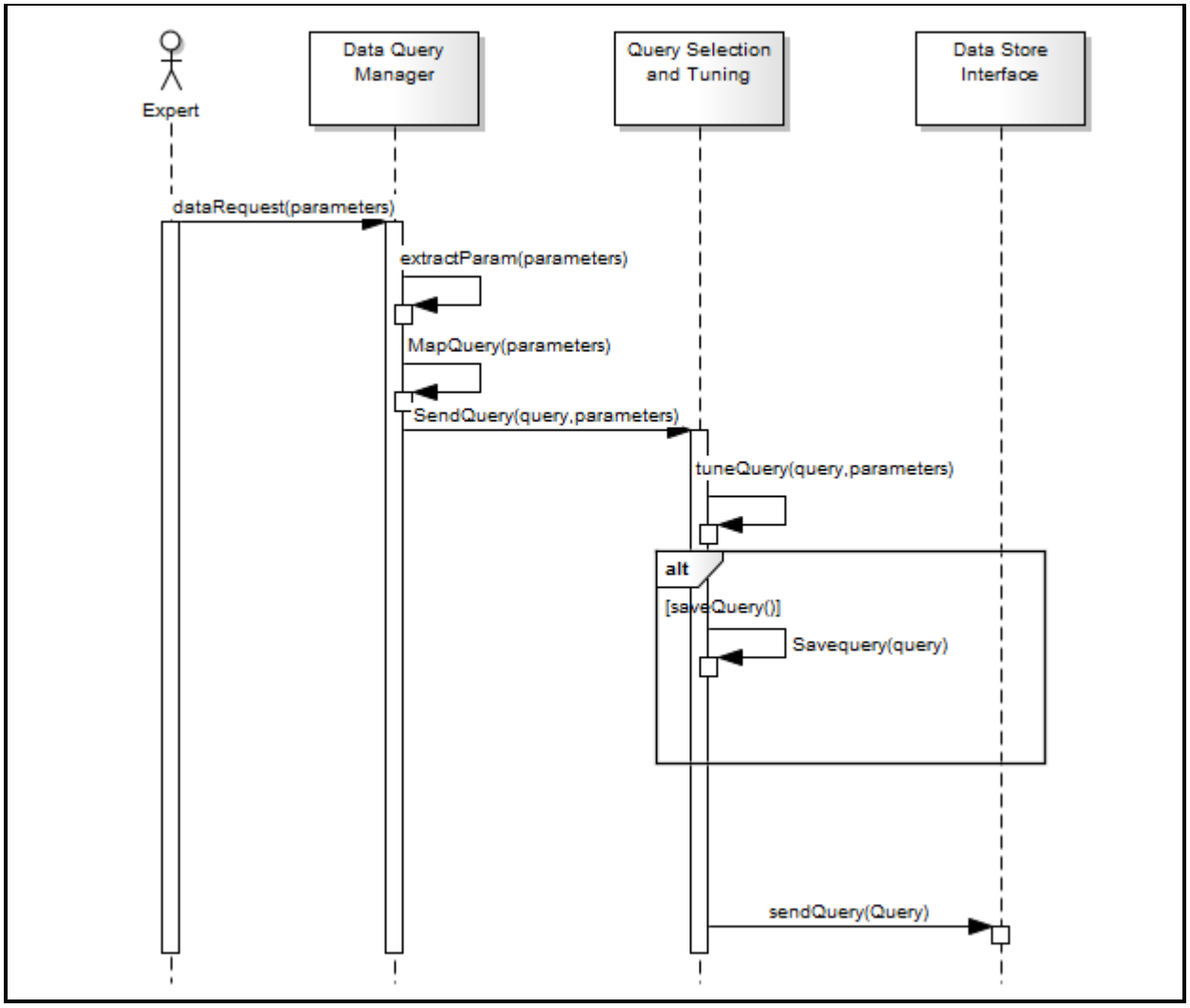
Trigger:	End user
Preconditions:	Self-reporting questionnaire already exist
Postconditions:	The feedback is successfully collected
Normal Flow:	1. The end user provide feedback using the questionnaires and feedback about recommendation. 2. The DCL stores the feedback in the lifelog for user preferences in the case of feedback. 3. The questionnaires are stored in the lifelog for further analysis from the expert
Alternative Flows:	NA
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Every 3 weeks
Special Requirements:	N/A
Assumptions:	The feedback data is saved in DCL
Notes and Issues:	N/A
Sequence Diagram	

Use Case ID:	SL-UC-10		
Use Case Name:	Retrieve capabilities for context-based services		
Created By:	Jamil Hussain	**Last Updated By:**	Anees Ul Hassan
Date Created:	March 09 2017	**Last Revision Date:**	01 July 2018
Actors:	DCL, SCL		
Description:	The UI/UX shall collect feedback about how users feel about recommendations and additional questionnaires in a predefined time period		
Trigger:	End user		
Preconditions:	Self-reporting questionnaire already exist		
Postconditions:	The feedback is successfully collected		
Normal Flow:	1. The end user provide feedback using the questionnaires and feedback about recommendation. 2. The DCL stores the feedback in the lifelog for user preferences in the case of feedback. 3. The questionnaires are stored in the lifelog for further analysis from the expert		
Alternative Flows:	NA		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Every 3 weeks		
Special Requirements:	N/A		
Assumptions:	The feedback data is saved in DCL		
Notes and Issues:	N/A		
Sequence Diagram			



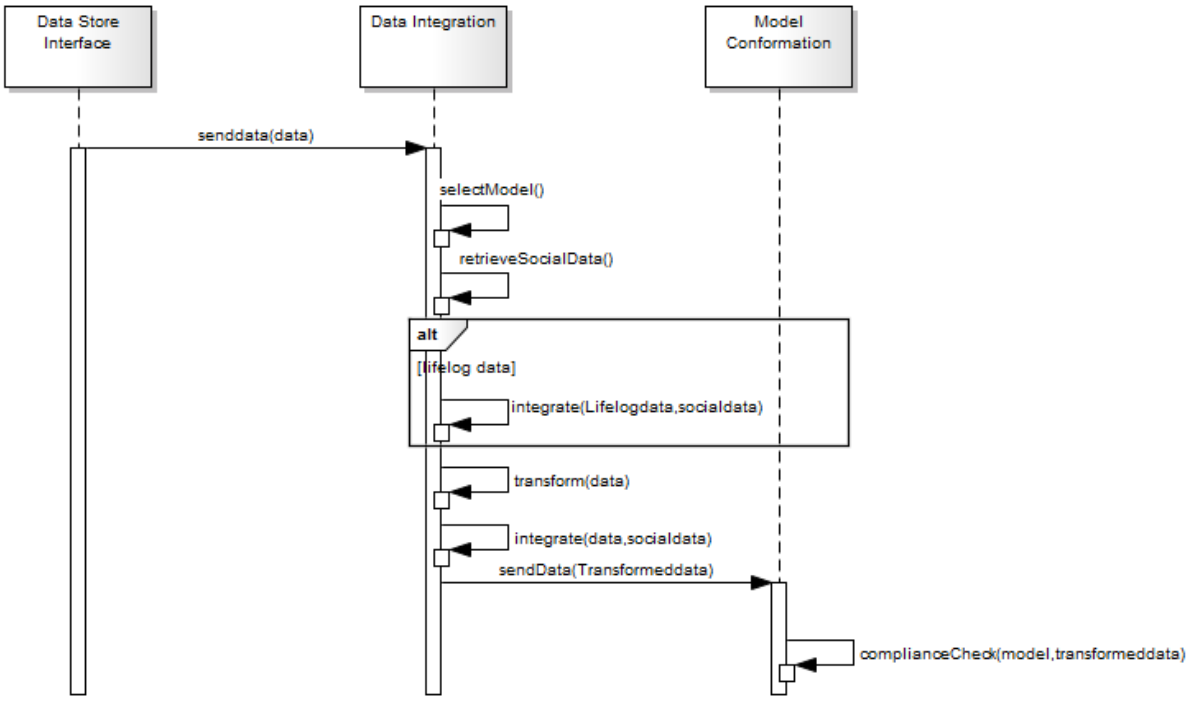
Use Case ID:	SL-UC-11		
Use Case Name:	Map Request to Query		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	Primary: Expert		
Description:	This use case focuses on mapping the expert request to the query library for data store interface.		
Trigger:	The request from the expert panel for analytics		
Preconditions:	A predefined query library for retrieving the big data		
Postconditions:	The query is sent to the data store interface and the data is fetched.		
Normal Flow:	1. The expert requests the analytics for a specific context. 2. The parameters of the request is extracted and sent to query ma nager. 3. The query manager matches the parameters with the predefined queries in the query library. 4. The Query is selected and tuned according to the duration of the data to be extracted.		

Alternative Flows:	4a. In step 4 of the normal flow, if there is a more tuning done than the query 1. The query is saved in the library for future calls.
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	This use case can be used by the domain expert about 5-10 times based on the volume of data. [Low]
Special Requirements:	N/A
Assumptions:	For this use case the assumption is a query library.
Notes and Issues:	1. How many queries are there in the query library?
Sequence Diagram	

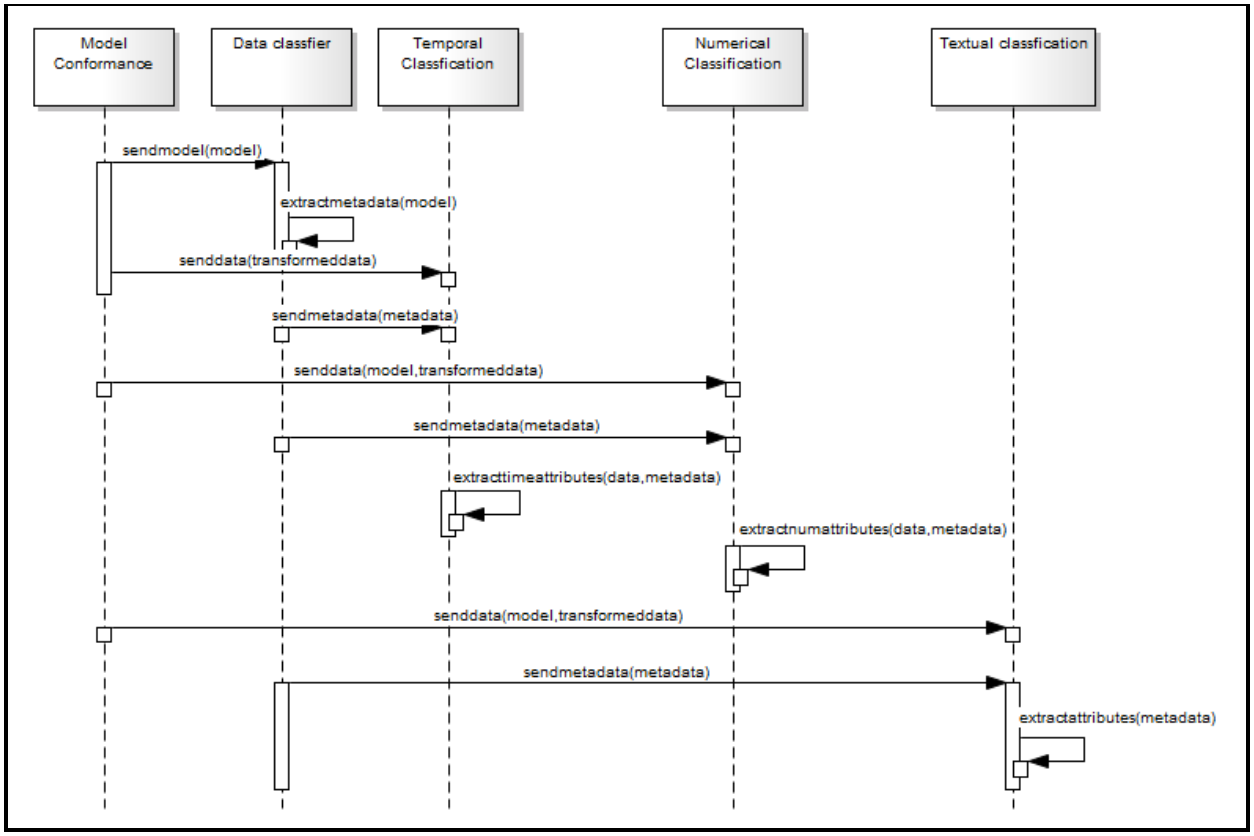


Use Case ID:	SL-UC-12		
Use Case Name:	Transform Data		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	Primary: Expert		
Description:	The mapping query is transformed to specific model structure for trend analysis.		
Trigger:	The data store interface initiates the data transformation process		
Preconditions:	The data is sent from the data store interface.		
Postconditions:	The transformed data is sent to trend analyzer.		

Normal Flow:	1. The unstructured data from the big data repository is sent to the data integration component. 2. The data is transformed in an object model or a table depending on the requirements. 3. The social network data is than additionally integrated which is r etrieved through a web service. 4. The transformed data is then checked for compliance with the m odel template.
Alternative Flows:	2a. In step 2 of the normal flow, if the data is retrieved from the life log then it is sent directly to the integration component.
Exceptions:	If the transformed data does not pass the compliance check, step 2 is started again.
Includes:	N/A
Frequency of Use:	This use case is used when the data comes from the big data and requires social data integration. [Low]
Special Requirements:	N/A
Assumptions:	
Notes and Issues:	1. How many models can the data be transformed in?
Sequence Diagram	

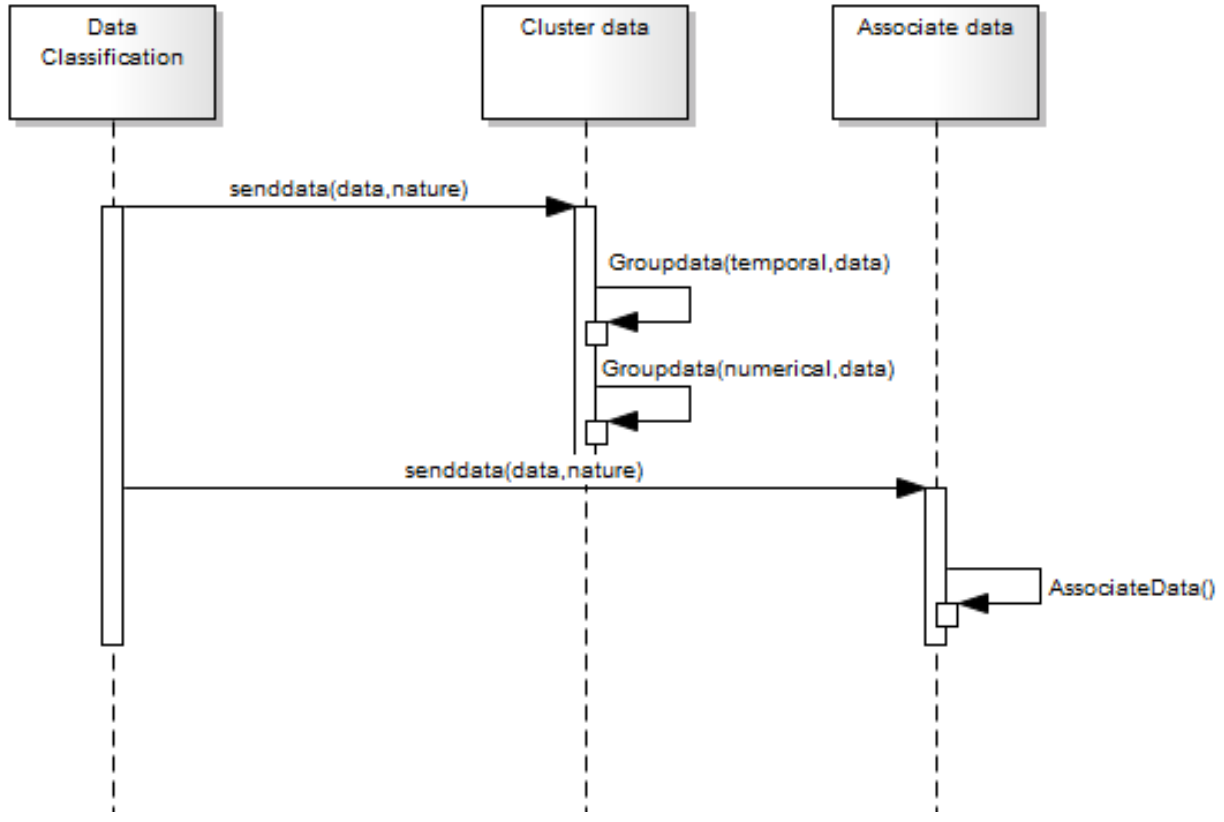


Use Case ID:	SL-UCS-13		
Use Case Name:	Classify Data		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	Primary: Expert		
Description:	The transformed data is further classified and clustered to identify and analyze trends.		
Trigger:	The transformed data is sent for trend analysis.		
Preconditions:	The data is structured into a particular model.		
Postconditions:	The data is classified into temporal, numerical and textual categories		
Normal Flow:	1. The model is passed for the classification. 2. Metadata is extracted from the model. 3. The data is categorized based on the extracted metadata. 4. The temporal, numerical and textual data is extracted from the tran sformed data.		
Alternative Flows:			
Exceptions:	1. There is no temporal data to be classified. 2. There is no numerical data to be classified. 3. There is no textual data to be classified.		
Includes:			
Frequency of Use:	This use case is used when the transformed data comes from model transformation module. [Low]		
Special Requirements:	NA		
Assumptions:	NA		
Notes and Issues:	NA		
Sequence Diagram	NA		

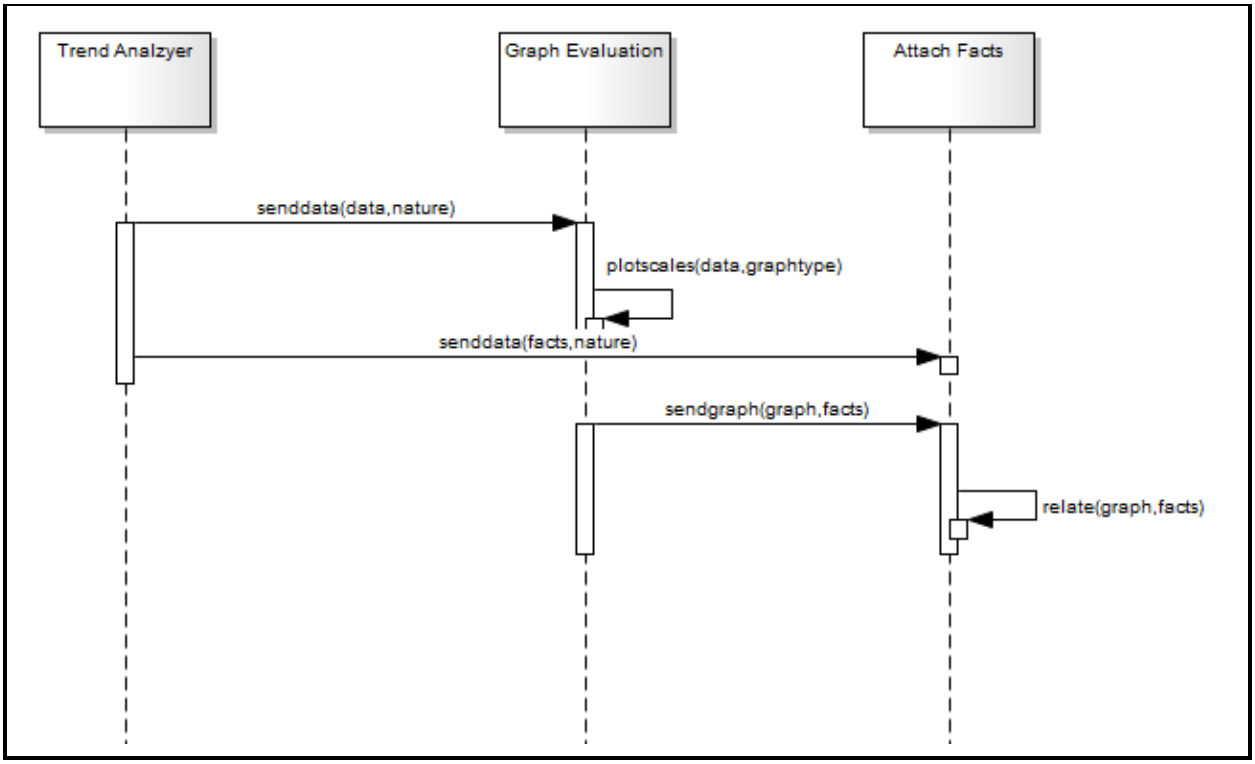


Use Case ID:	SL-UC-14		
Use Case Name:	Analyze Data		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	Primary: Expert		
Description:	The classified data is analyzed through association and clustering techniques for visualization and analytics.		
Trigger:	The classified data is passed to association clustering for finding analytics and trends.		
Preconditions:	The numerical, temporal and textual data is classified separately so that association could be applied.		
Postconditions:	The association is done with the data for analytics and data to be plotted is sent for visualization.		

Normal Flow:	1. The data classifier passes the data for association clustering. 2. The temporal and numerical data is analyzed for clustering. 3. The data is clustered into a group for graph plotting. 4. The textual data is associated with each other to create analytics b ased on the textual attribute and their corresponding facts.
Alternative Flows:	
Exceptions:	1. Clustering could not be done due to multiple outliers.
Includes:	
Frequency of Use:	This use case is used when the classified data is sent for grouping and association. [Low]
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram	NA

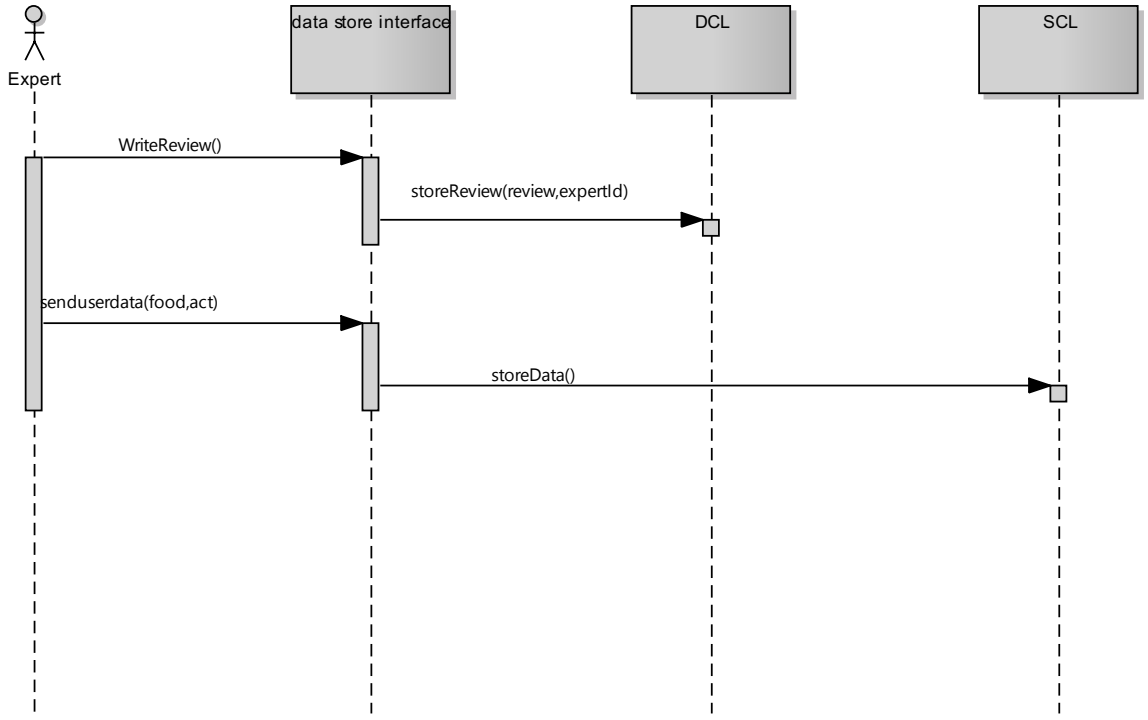


Use Case ID:	SL-UC-15		
Use Case Name:	Display Analytics		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	Primary: Expert		
Description:	The grouped data and relevant analytics is passed to visualization enabler so that the graphs are plotted and displayed.		
Trigger:	The trend analyzer sends the data for graph visualization and plotting.		
Preconditions:	The data is sent to visualization enabler distinguishable by their attributes and association.		
Postconditions:	The analytics and relevant visualization is sent to the user interface.		
Normal Flow:	1. The data is categorized according to the graph templates for visua lization. 2. The scales are defined for the grouped data to be plotted on the c oordinates. 3. The association text and the relevant facts about the data is also a ttached to the graph as analytics.		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	This use case is used when the grouped data is sent for display in graph and analytics form. [Low]		
Special Requirements:	NA		
Assumptions:	NA		
Notes and Issues:	NA		
Sequence Diagram			

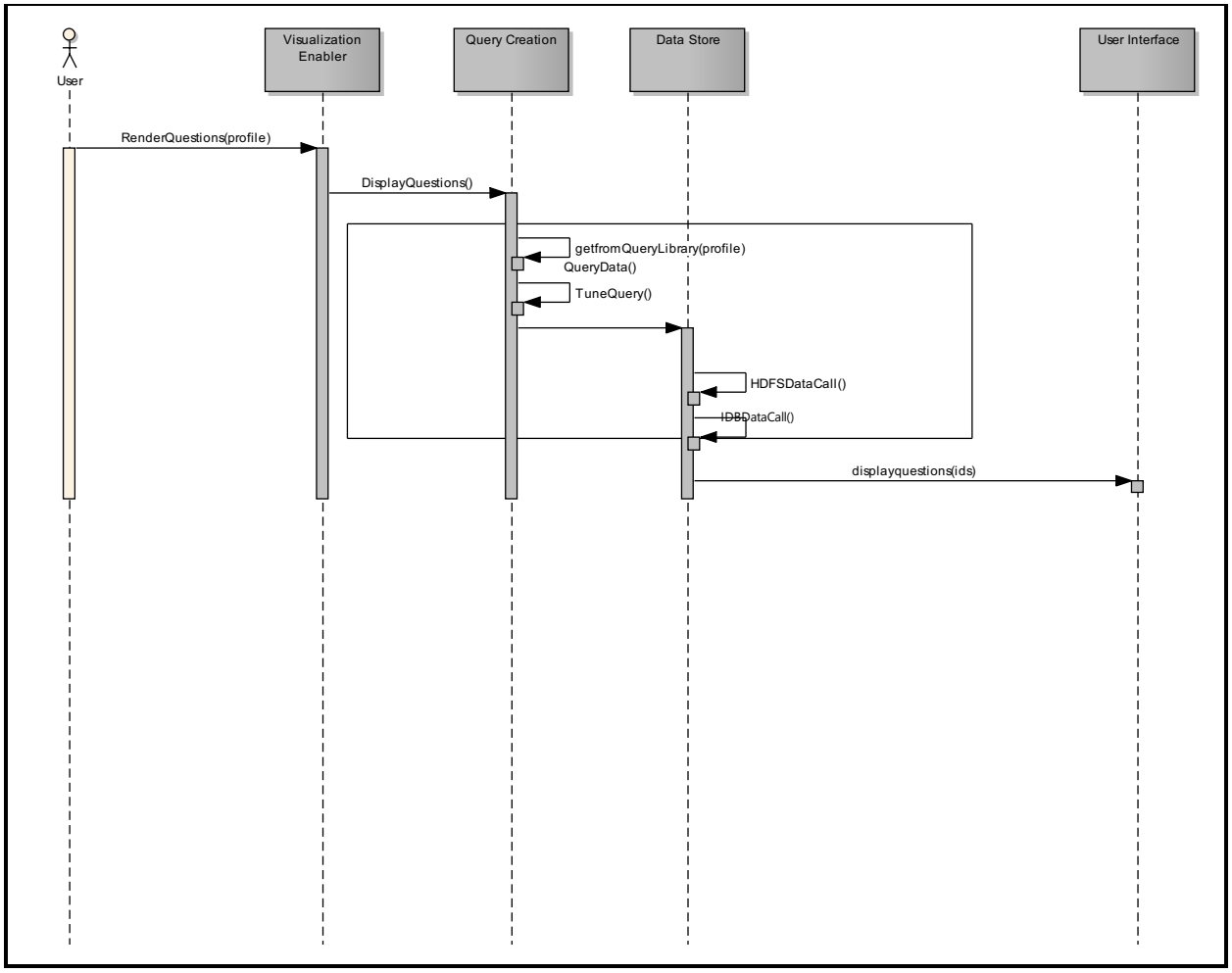


Use Case ID:	SL-UC-16		
Use Case Name:	Take Expert Input		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	01 July 2018
Actors:	Primary: Expert		
Description:	This use case focuses on taking expert input on activity and nutrition for the user's analytics		
Trigger:	The request from the expert panel for input		
Preconditions:	The user meal and activity information is stored		
Postconditions:	The query is sent to the data store interface and the data is stored.		
Normal Flow:	1. The expert requests the input panel for a specific user. 2. The parameters of the request is extracted and sent to query manager. 3. The query manager matches the parameters with the predefined queries in the query library.		

	4. The interface is shown for expert to input activity and meal in formation 5. It is stored in the life log
Alternative Flows:	4a. In step 4 of the normal flow, if there is a more tuning done than the query 1. The query is saved in the library for future calls.
Exceptions:	NA
Includes:	NA
Frequency of Use:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram	



Use Case ID:	SL-UC-17		
Use Case Name:	Display Adaptive Questionnaires		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	26 October 2017	**Last Revision Date:**	01 July 2018
Actors:	Primary: User		
Description:	This use case focuses on displaying questions adaptively depending on the user situation.		
Trigger:	The request from the user after a stage is completed for questions		
Preconditions:	The situation is analyzed to identify the questions.		
Postconditions:	The questions are displayed to the user		
Normal Flow:	1. The user requests questions after completing the stage. 2. The parameters of the request is extracted and sent to quer y manager according to the user situation. 3. The query manager matches the parameters with the prede fined queries in the query library. 4. The questions are shown to the user with respect to the sta ge and situation		
Alternative Flows:	4a. In step 4 of the normal flow, if there is a more tuning done than the query 2. The query is saved in the library for future calls.		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	NA		
Special Requirements:	NA		
Assumptions:	NA		
Notes and Issues:	NA		
Sequence Diagram			


User
Visualization Enabler
Query Creation
Data Store
User Interface
RenderQuestions(profile)
DisplayQuestions()
getfromQueryLibrary(profile)
QueryData()
TuneQuery()
HDFSDataCall()
IDBDataCall()
displayquestions(ids)

6. Collaboration Diagram