SOFTWARE DESIGN DESCRIPTION

Garcon

Uğur Düzel	2171569
Güneş Çepiç	2171494
Beyazıt Yalçınkaya	2172138

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Purpose of the System

Inspired by Microsoft's Garcon[3] project, the Garcon System in METU will make the lives of thousands of people easier and faster. As the population of METU increases every year, the university campus grows and the campus life gets more complicated. Even for students, professors, and university staff, it becomes harder to follow events, announcements, or any other news in the university. Garcon solves these issues and creates a network for infrastructural needs and sharing information among the university members. Garcon is not only a service for university members but also it is a general service for the public. METU campus has a very active scientific and cultural life and it hosts thousands of people every day on many occasions such as conferences, workshops, or art exhibitions. Garcon assists these people on the campus and provides all the necessary information they need. In overall, Garcon is designed to make campus life experience in METU easier, faster, and a lot more entertaining than before both for the university population and the public visitors.

1.2. Scope

The scope of this project is providing users to get information that they need about the METU Campus and to solve problems very quickly with combined systems it consists. Garcon creates an environment to take care of multiple needs of its user:

- Users can report infrastructural issues, request solutions, and the system will notify the corresponding METU unit regarding the issue.
- Users can ask for any necessary information about the campus and the system gives the exact information to them.

Moreover, Garcon System uses Microsoft Azure Language Understanding API to process user input in natural language and extract intention. By this software, users will be able to receive the information that they need as a response and by using the ticket system, problematic issues will be solved efficiently. Moreover, since Microsoft Azure Language Understanding is a machine learning model that recognizes natural language, provided that the user consent is taken, all of the data that flow through services such as queries, location information, appointments, reservations etc. will be used to train Microsoft Azure Language Understanding model in order to enhance the performance of the system. All in all, Garcon is a system designed to meet any need of university members and visitors.

1.3. Stakeholders and their Concerns

- **Member Users:** Member users are people associated with the university, i.e., professors, university staff, and students. They use system to retrieve information regarding the university and campus life as well as sending tickets to the corresponding university units. Their main concern about the system is its performance and responsiveness. They want system to have good performance for their information and ticket requests. They also require system to be responsive to their needs.
- **Non-member Users:** Non-member users are people not associated with the university, i.e., visitors, students' familie etc.. They use system to retrieve general information regarding the university and campus. Their main concern is the responsiveness and simplicity of the response. Since most of the visitors are not familiar with the university, they require system to be responsive to their needs. Moreover, they want system responses to be simple and easy-to-understand so that they can solve their problems fastly.
- **System Admins:** System admins are responsible for the maintenance and development of the system. They update the system to solve bugs, improve system performance etc. and they monitor the system periodically by checking error logs. Their main concern is having well-structured error logs to monitor the system and diagnose problems as well as having all the system data stored to train the Microsoft Azure Language Understanding component for more accurate recognition.

2. References

This document is written with respect to IEEE 29148-2011 standard:
IEEE. (2011, December 1). 29148-2011 - ISO/IEC/IEEE International Standard Systems and software engineering -- Life cycle processes --Requirements engineering. Retrieved from http://ieeexplore.ieee.org/document/6146379/ on March 12, 2018. doi: 10.1109/IEEESTD.2011.6146379

Other Sources:
[1] Language Understanding (LUIS) Documentation,
https://docs.microsoft.com/en-us/azure/cognitive-services/luis/
[2] Microsoft Windows Protocols,
https://docs.microsoft.com/tr-tr/openspecs/windows_protocols/
[3] Microsoft Garcon,
https://www.youtube.com/watch?v=Ad_EHDcomR8&t=20s
[4] Bing Spell Checker,
https://docs.microsoft.com/en-us/azure/cognitive-services/bing-spell-check/
[5] MySQL,
https://dev.mysql.com/doc/

3. Glossary

Term	**Definition**
Non-member User	Person who is not a registered METU Member
Member User	METU Students, academic personnel, METU Staff
System Admin	The experts who are responsible for maintenance of the system and analyzing the problems.
Microsoft Azure	Cloud platform.
ID	Unique number to identify person.
DB	Abbreviation for database.
Smart Box	IoT Device consists of embedded systems
METU Infrastructural Services	A service that receives tickets created by users.
METU Online Services	A multi-procedure service that includes Ring Service, Cafeteria Service, Medico Service and Classroom Scheduling Service.
Microsoft Azure Language Understanding	A machine learning-based service to process natural language and extract intention. It creates LUIS[1] response.
CRUD	Abbreviation for create, read, update and delete operations of database.
SATA	Serial Advanced Technology Attachment which is used for transferring data between a computer and a storage device.
SAS	Serial Attached Small Computer System Interface which is used for transferring data between a computer and a storage device.
MySQL	World's most popular open source database. [5]

Azure Trainer	A machine-learning based service that loads all the stored data from Garcon's database and creates training data for the Microsoft Azure Language Understanding component for more accurate input recognition.
Bing Spell Check	Bing Spell Check is an API performs contextual grammar and spell checking which is a web-based spell-checker that leverages machine learning and statistical machine translation to dynamically train a constantly evolving and highly contextual algorithm. [4]
JSON	JavaScript Object Notation (JSON) is an open-standard file format.

Table 1: Glossary

4. Architectural Views

4.1. Context View

Garcon targets many needs of the user; hence, it interacts with lots of different user types, applications and databases. In particular, Microsoft Azure Language Understanding is the cloud service to process the given text based message and extract the intent of the message and the corresponding confidence score which are the subfields of JSON formatted LUIS Response. Below, we give the context diagram.

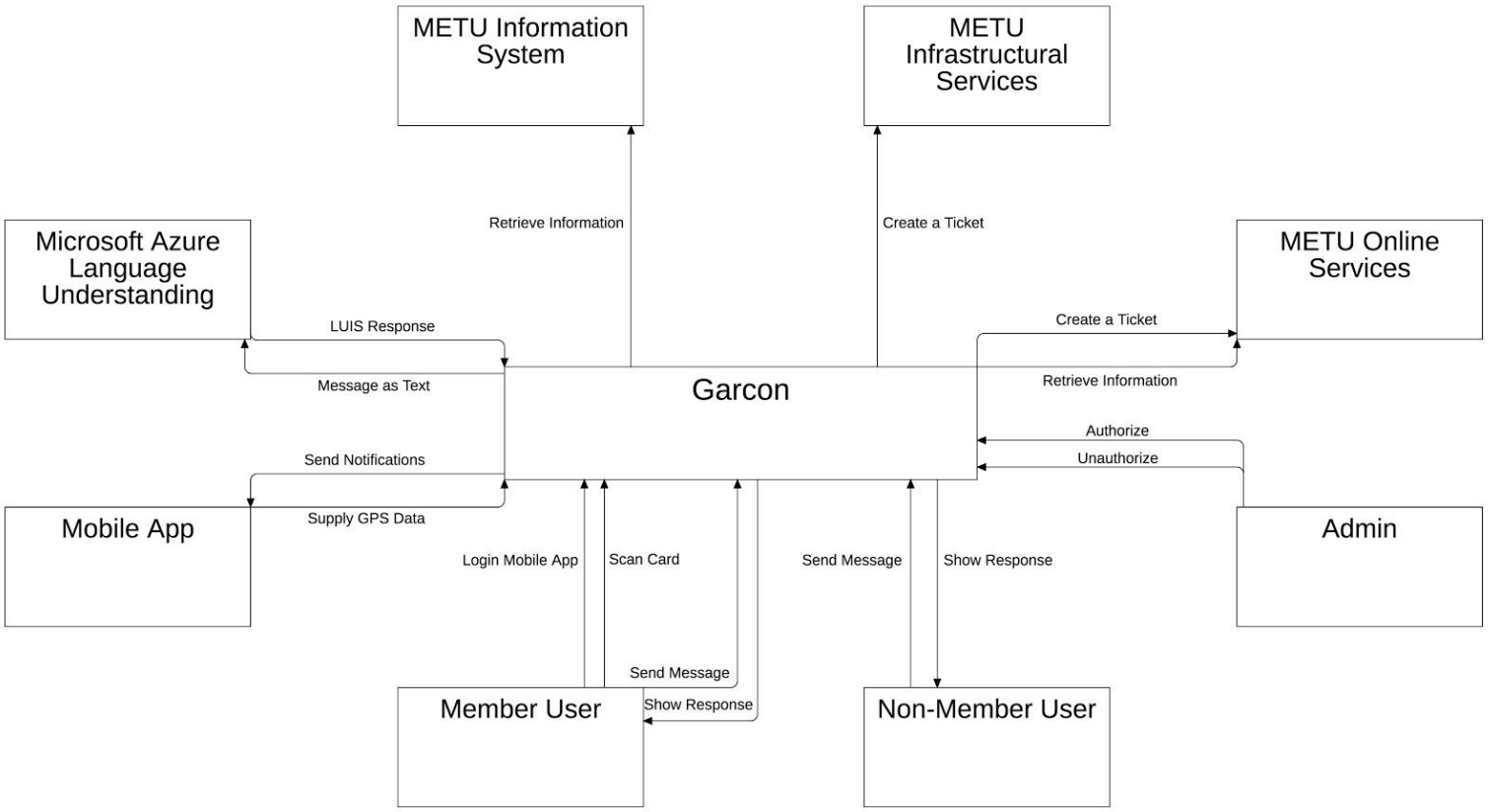

Figure 1: Context Diagram

Below, we give the use case diagram of the system. From this use case diagram, we present four description tables for selected system functions and for two of them we provide detailed sequence diagrams to analyze their operation procedures in depth.

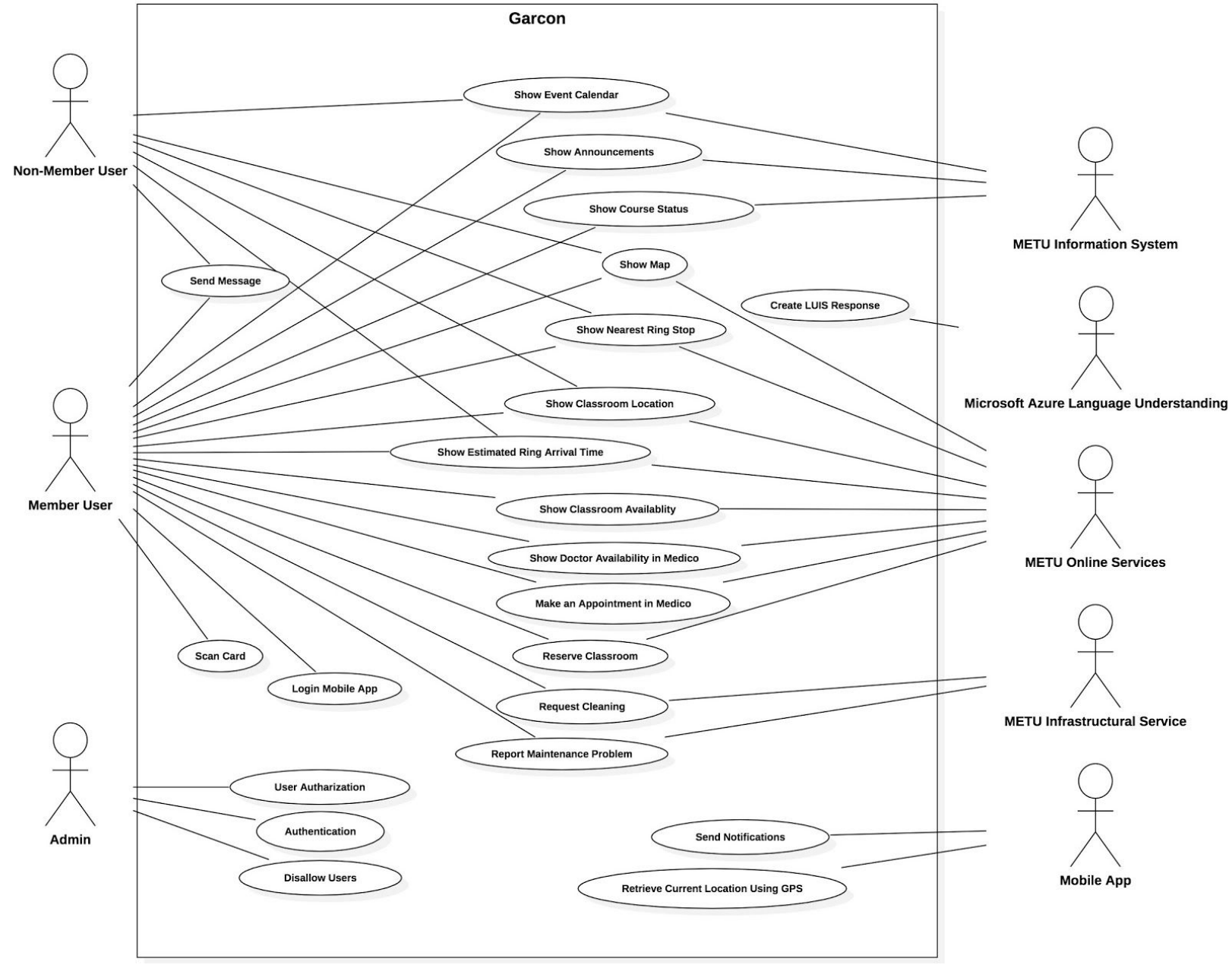

Figure 2: Use Case Diagram

Use Case Name	Report Maintenance Problem
Actors	Member User, METU Infrastructural Service
Description	Member users can report any problem requiring maintenance such as a toilet is clogged or a bathroom sink is broken. A ticket is sent upon the users request to notify METU Infrastructural Service which is the responsible unit for handling these problems.
Data	Identity information of the user who reported the problem, location, time, and additional information about the problem.
Preconditions	The user must be authenticated by scanning its ID card or logging into the mobile app.
Stimulus	Reporting the problem via the smart boxes located all around the campus or via the mobile app.
Basic Flow	**Step 1-** User scans the ID card, to get authenticated. **Step 2-** User tells the problem in natural language. **Step 3-** Smart box turns speech into text. **Step 4-** Server sends it to Microsoft Azure Language Understanding. **Step 5-** Text is translated into LUIS[1] response in Azure. **Step 6-** Server processes the response and sends a ticket to the corresponding METU Infrastructural Services unit.
Alternative Flow #1	**Step 1-** User logs into mobile app, to get authenticated. **Step 2-** User sends the problem as a text message. **Step 3-** Server sends it to Microsoft Azure Language Understanding. **Step 4-** Text is translated into LUIS [1] response in Azure. **Step 5-** Server processes the response and sends a ticket to the corresponding METU Infrastructural Services unit.
Alternative Flow #2	**Step 1-** User tries to get authenticated by scanning an ID card or logging into the mobile app. **Step 2-** Unauthorized access request is detected. **Step 3**- Permission denied.
Exception Flow	If there is an internet problem to connect Azure server, the error is saved to the Error log file in the server.
Postconditions	Garcon displays and tells a message to inform the user.

Table 2: Report Maintenance Problem

Use Case Name	Show Classroom Location
Actors	Member User, Non-Member User, METU Online Services
Description	Member and non-member users can check the exact locations of the classrooms in the campus. Garcon checks the location of the requested classroom from the METU Online Services (specifically, METU Maps) upon the users request and both displays and tells the exact location of the classroom.
Data	Name of the classroom.
Preconditions	A valid classroom name must be provided.
Stimulus	Asking for the location of a classroom via the boxes located all around the campus or via the mobile app.
Basic Flow	**Step 1-** User asks for a classroom location in natural language. **Step 2-** Smart box turns speech into text. **Step 3-** Server sends it to Microsoft Azure Language Understanding. **Step 4-** Text is translated into LUIS[1] response in Azure. **Step 5-** Server processes the response and checks the location of the specified classroom from the METU Online Services. **Step 6-** The location information of the classroom is given to the user as both speech and figure.
Alternative Flow #1	**Step 1-** User enters the classroom name via the mobile app. **Step 2-** Server checks the location of the specified classroom from the METU Online Services. **Step 3-** The location information of the classroom is given to the user as a figure.
Alternative Flow #2	**Step 1-** User enters asks for an invalid classroom name. **Step 2-** The error is reported back to the user as a figure.
Exception Flow	If there is an internet problem to connect Azure server, the error is saved to the Error log file in the server.
Postconditions	Garcon displays and tells an affirmative message to inform the user.

Table 3: Show Classroom Location

Use Case Name	Reserve Classroom
Actors	Member User, METU Online Services, METU Infrastructural Services
Description	Member users can check whether the classroom they want to use is available or not. If the classroom is available, a member user can reserve that classroom for a meeting or lecture by indicating date and time. Garcon creates a ticket upon the user's request and send it to the corresponding METU unit .
Data	Classroom status, date and classroom name.
Preconditions	Classroom must be available for desired time and the user must be authenticated by scanning its ID card or logging into the mobile app.
Stimulus	Asking for the reservation of the classroom via the boxes located all around the campus or via the mobile app.
Basic Flow	**Step 1-** User asks for reservation of the class in natural language. **Step 2-** Smart box turns speech into text. **Step 3-** Server sends it to Microsoft Azure Language Understanding. **Step 4-** Text is translated into LUIS[1] response in Azure. **Step 5-** Server processes the response **Step 6-** Checks if the specified classroom is available from METU Online Services "Show Classroom Availability" **Step 7-** A ticket has created and sent to the corresponding unit of METU Infrastructural Services to save the reservation.
Alternative Flow #1	**Step 1-** User asks for reservation of the class via mobile app. **Step 2-** Server checks if the specified classroom is available. **Step 3-** Server processes the response and send a ticket to the corresponding unit of METU Infrastructural Services to save the reservation.
Alternative Flow #2	**Step 1-** User asks for a unavailable classroom. **Step 2-** The error is reported back to the user as a figure.
Exception Flow	If there is an internet problem to connect mobile app or Azure server, the error is saved to the Error log file in the server.
Postconditions	Garcon displays and tells a message to inform the user.

Table 4: Reserve Classroom

Use Case Name	Show Estimated Arrival Time Of The Ring
Actors	Member User, Non-Member User, METU Online Services
Description	Member and non-member users can check arrival time of the ring in the campus. Garcon checks the location of the requested ring from the METU Online Services (specifically, METU Ring Service) upon the users request and both displays and tells the arrival time of the classroom.
Data	Arrival time of the ring.
Preconditions	A valid ring type must be provided.
Stimulus	Asking for the arrival time of the ring via the boxes located all around the campus or via the mobile app.
Basic Flow	**Step 1-** User asks for arrival time of the ring in natural language. **Step 2-** Smart box turns speech into text. **Step 3-** Server sends it to Microsoft Azure Language Understanding. **Step 4-** Text is translated into LUIS[1] response in Azure. **Step 5-** Server processes the response and checks arrival time of the specified ring from the METU Online Services. **Step 6-** The arrival time information of the ring is given to the user as both speech and figure.
Alternative Flow #1	**Step 1-** User chooses the ring type via the mobile app. **Step 2-** Server checks the arrival time of the specified ring from the METU Online Services. **Step 3-** The arrival time information of the ring is given to the user as a figure.
Alternative Flow #2	**Step 1-** User enters asks for an invalid ring type. **Step 2-** The error is reported back to the user as a figure.
Exception Flow	If there is an internet problem to connect Azure server, the error is saved to the Error log file in the server.
Postconditions	Garcon displays and tells an affirmative message to inform the user.

Table 5: Show Estimated Arrival Time Of The Ring

4.2. Composition View

In this section, we give composition view of the system. First we present the detailed component diagram and deeply discuss its design rationale. Then, the deployment diagram of the system is given with a detailed discussion of its design rationale.

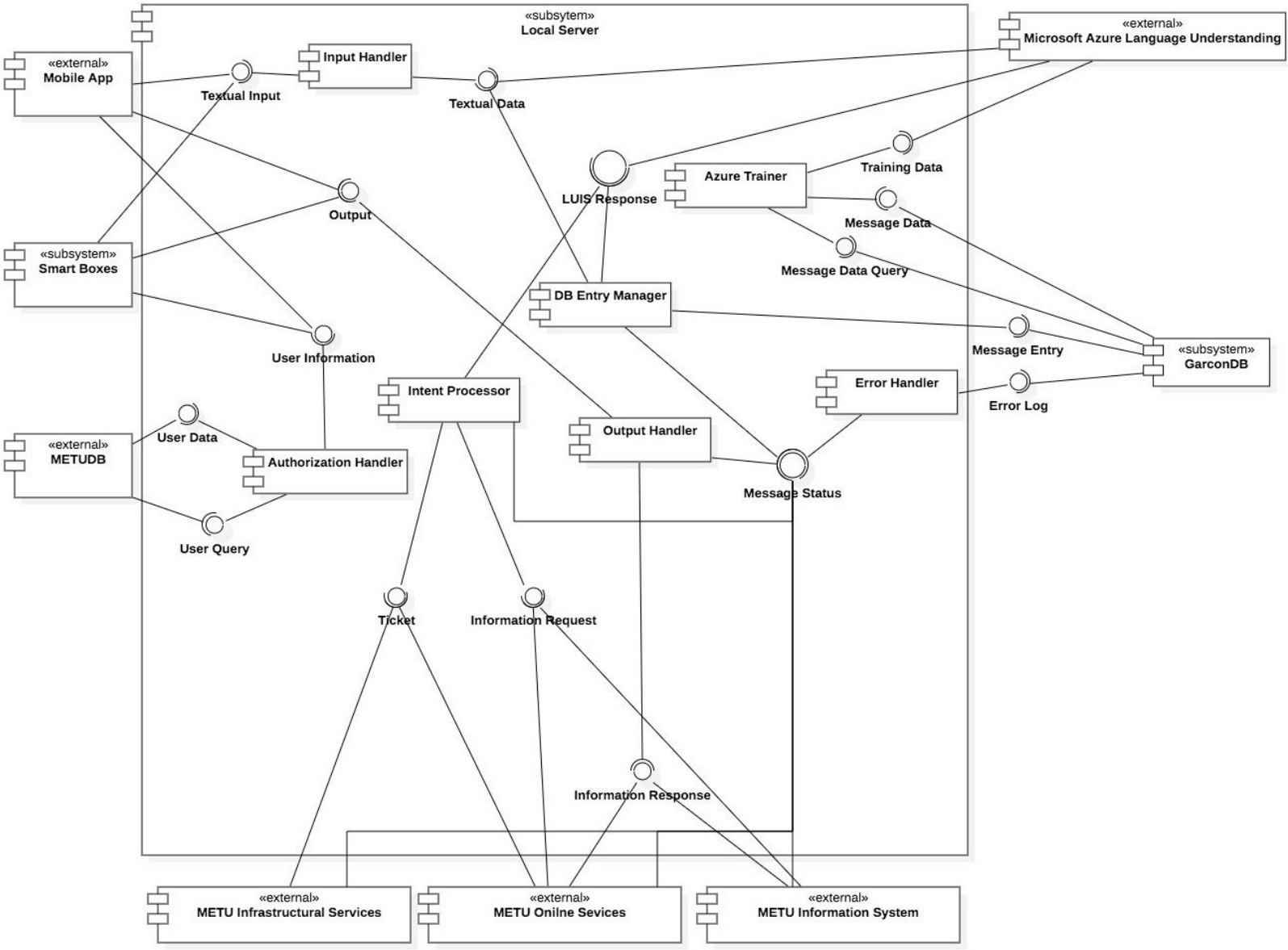

Figure 3: Component Diagram

Design Rationale:

- Smart Boxes component converts given audio input to the textual input and sends this textual input to the Input Handler. It also sends the user data to the Authorization Handler for checking the identity information of the user. It receives the corresponding output of the input from the Output Handler and displays it to user.
- GarconDB component receives data query from the Azure Trainer and it send the corresponding data back to the Azure Trainer. It also receives message entry and error log from DB Entry Manager and Error Handler, respectively.
- Local Server is basically responsible for all functionalities of Garcon. It consists of several subcomponents and each of these subcomponents interacts with several external components, i.e., Mobile App, METUDB, Microsoft Azure Language Understanding, METU Infrastructural Services, METU Online Services, and METU Information System.
- Input Handler component is responsible for directing the textual input received by either Smart Boxes or Mobile App to Microsoft Azure Language Understanding and DB Entry Manager components as textual data.
- Authorization Handler authorizes a user by sending a user query to the METUDB and receiving the corresponding data of the user from the METUDB, It checks whether the data of the user received either from Smart Boxes or from Mobile App matches with the data from the METUDB and if so, it authorizes the user to use the system.
- Intent Processor receives the extracted LUIS Response from the Microsoft Azure Language Understanding component for an input. According to its semantic meaning, it sends a ticket to the METU Infrastructural Services or METU Online Services or sends an information request to METU Online Services or METU Information System.
- Output Handler receives information response from METU Online Services or METU Information System for a requested information and also it receives message status from METU Online Services or METU Information System or METU Infrastructural Services for either a requested information or a ticket. According to the received information response and message status, it sends an output to Smart Boxes or Mobile App.
- DB Entry Manager receives message status from METU Online Services or METU Information System or METU Infrastructural Services, textual data from Input Handler, and LUIS Response from Microsoft Azure Language

Understanding. It combines these and sends message entry to GarconDB for storage.

- Error Handler receives message status from METU Online Services or METU Information System or METU Infrastructural Services and if message status indicates an error in the message, issues an error log and sends it to GarconDB.
- Azure Trainer component sends data query to GarconDB and receives corresponding data from the GarconDB. It sends training data to the Microsoft Azure Language Understanding component for a more accurate recognition in the future.

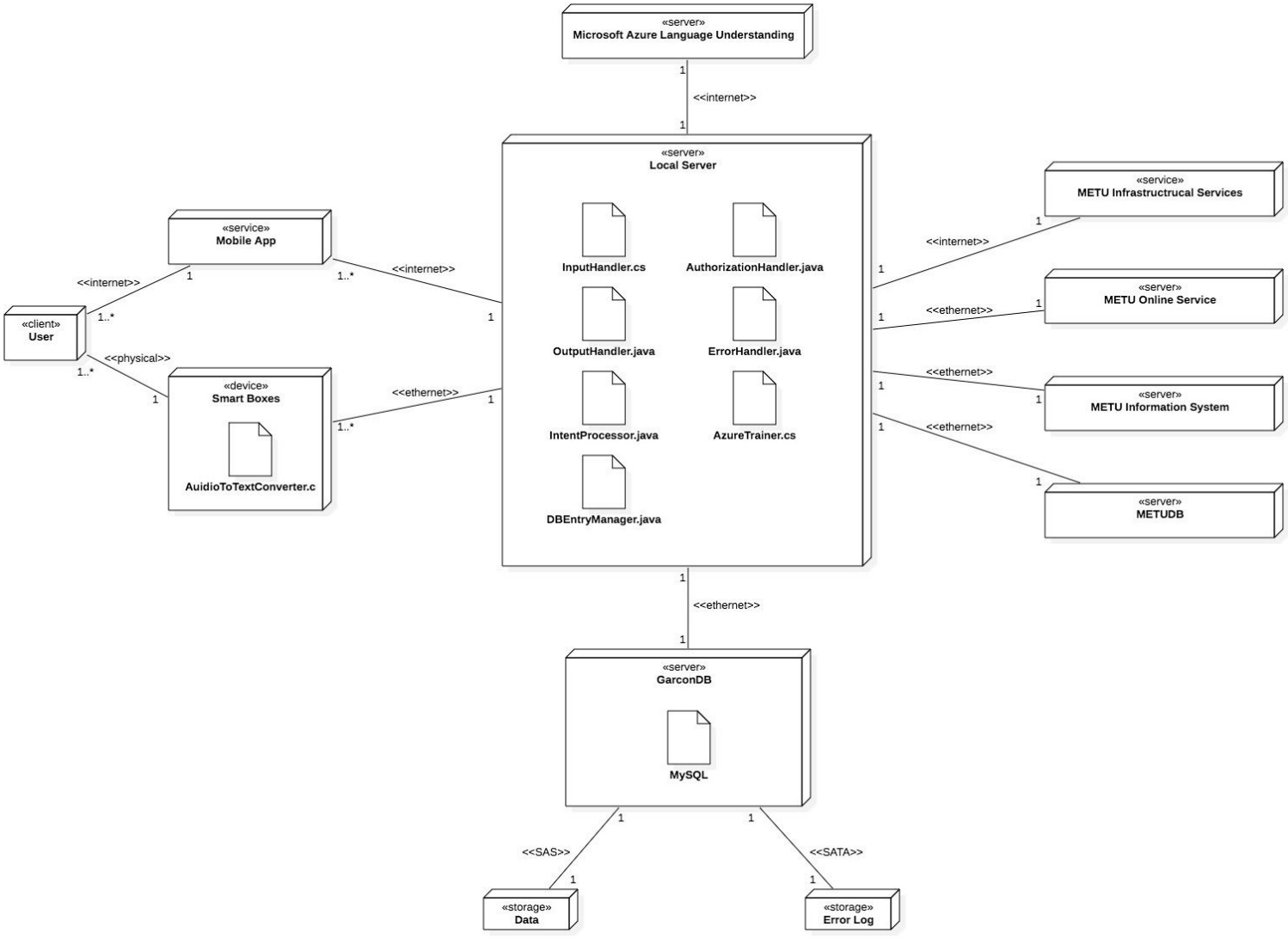

Figure 4: Deployment Diagram

Design Rationale:

- We divided Garcon implementation into two parts: Local Server and GarconDB. This division was necessary since most of the functional computation is made on-fly basis; however, the system needs information regarding previous messages to increase the accuracy of the Microsoft Azure Language Understanding. Both of them are physically located in the METU Campus and hard-wired to each other in order to meet performance requirements of the system as well as to ensure security.
- The backend structure of the system is supported by MySQL and Java. We used MySQL since it was free and open source and Java is a sufficient choice for the backend implementation since it was fast and modular enough. We use C# for the implementation of Input Handler and Azure Trainer since these components interact with Microsoft Azure Language Understanding and it requires these components to be implemented in C#.

- The GarconDB is divided into two parts: Data and Error Log. Data part of the database stores textual input, LUIS Response, and message status information for training Microsoft Azure Language Understanding. Error Log part of the database stores error logs of the system for the maintenance concerns. We used SAS for storing message data for a faster loading training data to the database and we used SATA for error logs since storing error logs does not require fast accesses to the database.
- Each smart box is identical and contains a microprocessor. AudioToTextConverter.c file programs this microprocessor to convert audio input to the textual input.
- Local Server is connected to the external servers by ethernet connection and it is connected to services by internet connection.

4.3. Information View

In this view, the classes of the system and their attributes are shown with their detailed descriptions. Since these informations are stored in databases, how these informations would be stored is specified below by create, read, update, delete operations for every component that will be stored in the database.

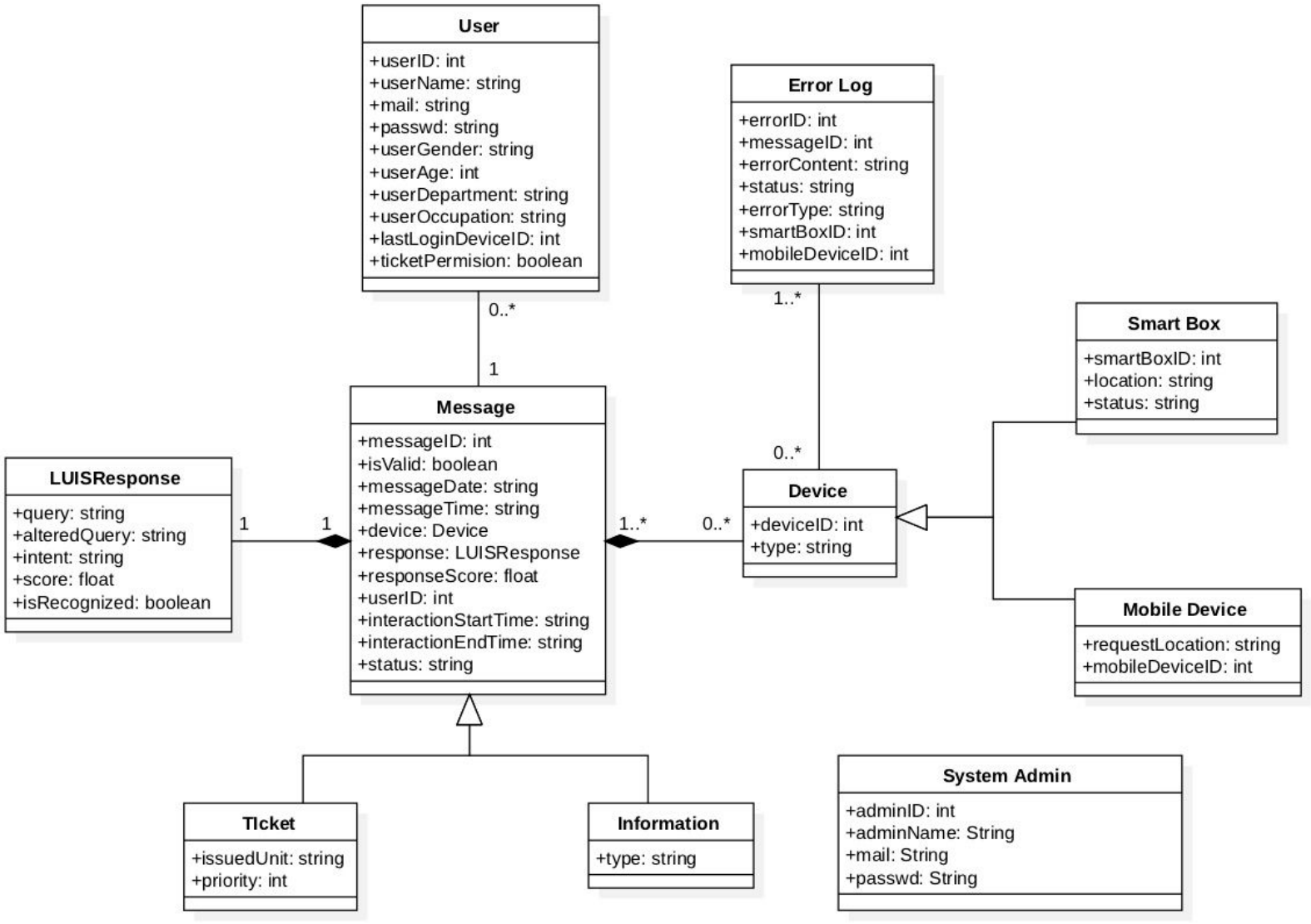

Figure 5: Database Class Diagram

Operation	**CRUD Operations**
topIntent	CREATE: LUISResponse READ: - UPDATE: - DELETE: -
processIntent	CREATE: LUISResponse READ: - UPDATE: - DELETE: -
sendTicket	CREATE: Ticket READ: - UPDATE: - DELETE: -
sendInformation	CREATE: Information READ: Device UPDATE: - DELETE: -
checkRecognition	CREATE: LUISResponse READ: - UPDATE: - DELETE: -
isAltered	CREATE: LUISResponse READ: - UPDATE: - DELETE: -
getIntent	CREATE: - READ: LUISResponse UPDATE: - DELETE: -
getScore	CREATE: - READ: LUISResponse UPDATE: - DELETE: -
getResponseRate	CREATE: - READ: LUISResponse UPDATE: - DELETE: -

setResponseRate	CREATE: - READ: - UPDATE: LUISResponse DELETE: -
getMessage	CREATE: Message READ: - UPDATE: - DELETE: -
sendToAzure	CREATE: - READ: - UPDATE: - DELETE: -
getMessageID	CREATE: - READ: Message UPDATE: - DELETE: -
getMessageLength	CREATE: - READ: Message UPDATE: - DELETE: -
getMessageTime	CREATE: - READ: Message UPDATE: - DELETE: -
checkWhichDevice	CREATE: - READ: Device, Message UPDATE: - DELETE: -
getSmartBoxID	CREATE: - READ: Device UPDATE: - DELETE: -
getLocation	CREATE: - READ: Smart Box, Mobile Device UPDATE: - DELETE: -
receiveMessageStatus	CREATE: -

	READ: Message, Error Log UPDATE: - DELETE: -
waitForInformation	CREATE: - READ: - UPDATE: - DELETE: -
raiseAuthError	CREATE: - READ: - UPDATE: - DELETE: -
raiseRecogError	CREATE: - READ: - UPDATE: - DELETE: -
returnSuccess	CREATE: - READ: - UPDATE: - DELETE: -
getResponse	CREATE: Message READ: - UPDATE: - DELETE: -
authenticate	CREATE: System Admin READ: - UPDATE: - DELETE: -
grantNonMemberAccess	CREATE: System Admin READ: - UPDATE: - DELETE: -
getCertificate	CREATE: - READ: - UPDATE: - DELETE: -
getLUISresponse	CREATE: LUISResponse READ: -

	UPDATE: - DELETE: -
getInformationResponse	CREATE: Information READ: - UPDATE: - DELETE: -
getMessageDetails	CREATE: Message READ: - UPDATE: - DELETE: -
makeConnection	CREATE: - READ: - UPDATE: - DELETE: -
executeQuery	CREATE: any READ: - UPDATE: any DELETE: -
getQuery	CREATE: - READ: - UPDATE: - DELETE: -
getMessageStatus	CREATE: - READ: - UPDATE: - DELETE: -
logRecogError	CREATE: Error Log READ: - UPDATE: - DELETE: -
logAuthError	CREATE: Error Log, System Admin READ: - UPDATE: - DELETE: -
getErrorType	CREATE: - READ: - UPDATE: -

	DELETE: -
getError	CREATE: - READ: - UPDATE: - DELETE: -
trainModel	CREATE: - READ: LUISResponse UPDATE: - DELETE: -
testModel	CREATE: - READ: LUISResponse UPDATE: - DELETE: -
createCrossValidationFolds	CREATE: - READ: LUISResponse UPDATE: - DELETE: -
preprocess	CREATE: - READ: - UPDATE: LUISResponse DELETE: -
getRawData	CREATE: - READ: LUISResponse UPDATE: - DELETE: -
getPreprocessedData	CREATE: - READ: LUISResponse UPDATE: - DELETE: -
getConfigurations	CREATE: - READ: LUISResponse UPDATE: - DELETE: -
getModelSelections	CREATE: - READ: LUISResponse UPDATE: - DELETE: -

getWordErrorRate	CREATE: - READ: - UPDATE: - DELETE: -
getSentenceAccuracy	CREATE: - READ: - UPDATE: - DELETE: -
getCommandAccuracy	CREATE: - READ: - UPDATE: - DELETE: -
getNumberOfFalsePos	CREATE: - READ: - UPDATE: - DELETE: -
getNumberOfFalseNeg	CREATE: - READ: - UPDATE: - DELETE: -
getNumberOfTruePos	CREATE: - READ: - UPDATE: - DELETE: -
getNumberOfTrueNeg	CREATE: - READ: - UPDATE: - DELETE: -

Table 6: CRUD Operations

Design Rationale:

- Message class is generalized class of Ticket and Information classes that is every message is either a Ticket or an Information.
- Device class is generalized class of Smart Box and Mobile Device classes that is every device is either Smart Box or Mobile Device.
- All user informations and the message informations which sent through the devices are hold in database in order to access those informations easily by admins.
- Error Log is a weak entity since without any failure happens in a device an error log cannot exist.
- In Error Log class, errorContent, errorType and status informations are hold in order to detect and handle the error quickly when the system is recovered.
- For a message there is always a LUISResponse to analyze the message according to its intent. Also score attribute is hold in database in order to train Microsoft Azure Language Understanding API according to matching percentage of the real intent of the message and corresponding LUISResponse.

4.4. Interface View

4.4.1. Service Interfaces

Below, we present the interface class diagram and give a detailed table describing its operations.

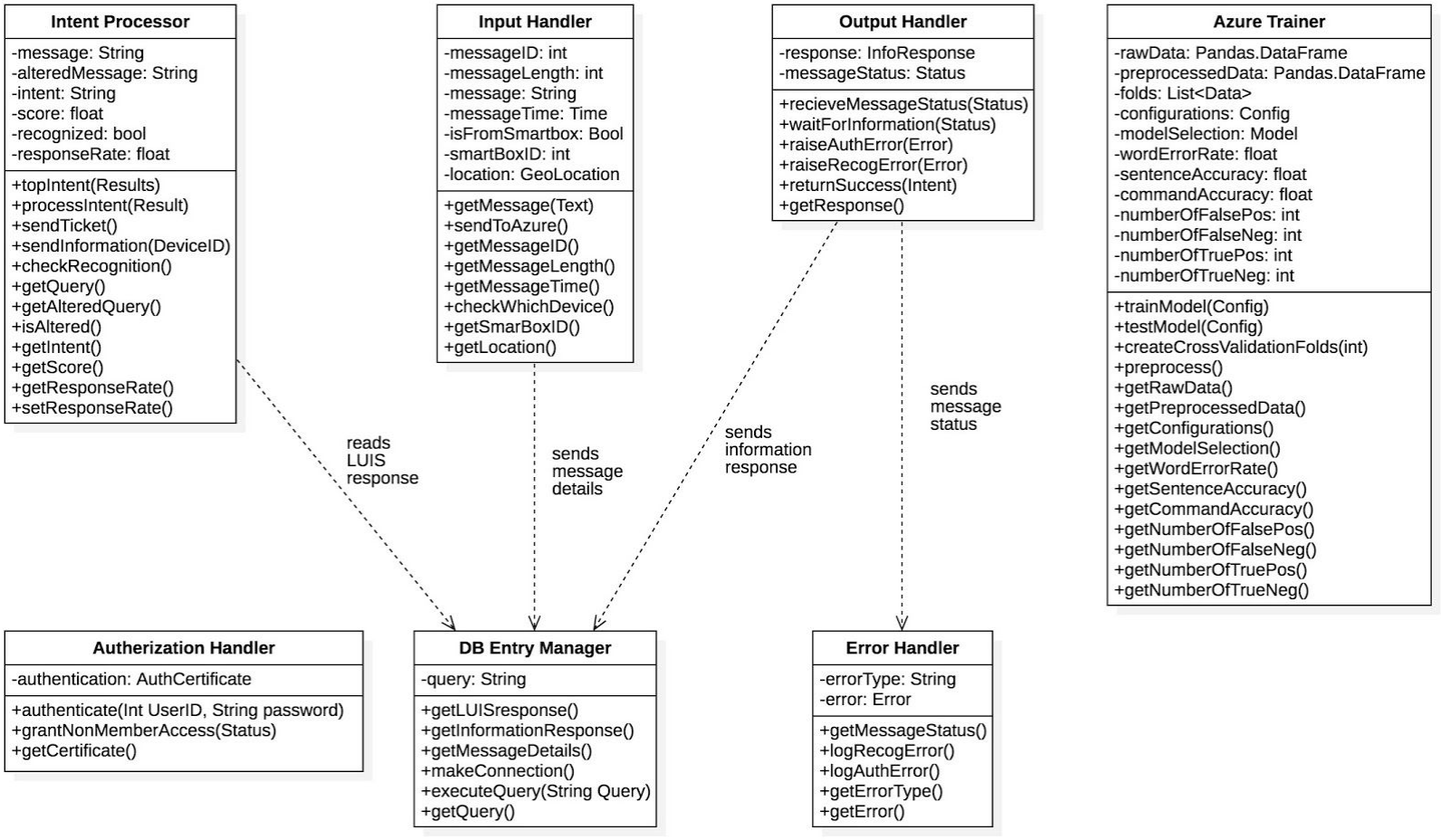

Figure 6: Interface Class Diagram

Operation	Description
topIntent	After receiving the LUIS response, this method determines which intent is has the best score.
processIntent	Processes the intent and determines whether it is a ticket or an information request.
sendTicket	After receiving and processing the intent, if the intent is a ticket then this method sends that ticket to the corresponding METU Staff.
sendInformation	After receiving and processing the intent, if the intent is an information retrieval request and the request is valid for the authorization of the user then the information is send to the user device to be displayed.
checkRecognition	Checks if the message is recognized or not.
isAltered	Checks if the message is altered by the Microsoft Azure Language Understanding built-in Bing Spell Check v7 service.
getIntent	When another class , for example DB Entry Manager, needs to obtain the intent of the message, this method is called.
getScore	This method is called by another class, for example a class admin implements for testing and maintenance, that needs to obtain the scores of the message.
getResponseRate	In case of another class needing the response rate of the message sent, this method is called.
setResponseRate	When another class needs to set the response rate of the message this method is called.

getMessage	This method is used to receive the input from the device.
sendToAzure	Transforms the received input into the Microsoft Azure Language Understanding input format and makes the necessary API call to start the text processing.
getMessageID	When another class , for example DB Entry Manager, needs to obtain the id of the message, this method is called.
getMessageLength	This method is called, when another class , for example a class admin implements for testing and maintenance, needs to obtain the length of the message.
getMessageTime	In case of another class needing the exact time of the message sent, this method is called.
checkWhichDevice	Determines if the message is sent via smartbox or mobile.
getSmartBoxID	Returns the smart box id of the device that recorded the message.
getLocation	Returns the recording location of the message.
receiveMessageStatus	Receives the message status that METU Infrastructural Services, METU Online Services or METU Information System returns, determines if the message is recognized or if the information requested can be supplied by these services.
waitForInformation	This method receives the information that METU Infrastructural Services, METU Online Services or METU Information System sends.
raiseAuthError	Raises authentication error and sends the user device a signal to display the error.
raiseRecogError	Raises recognition error and sends user

	device a signal to display the error.
returnSuccess	Sends the user device a signal to display a message to inform the user that their message has been successfully processed.
getResponse	This method is called, when another class , for example a class admin implements for testing and maintenance, needs to obtain the returned response of the message.
authenticate	Authenticates the provided user information.
grantNonMemberAccess	This method provide the following functionality. If the user is not a member of the premises then the system only allows the user to use certain capabilities.
getCertificate	Returns the certificate of the user authentication.
getLUISresponse	Gets the LUIS response from the Intent Processor class to use it in the queries.
getInformationResponse	Gets the information returned from output handler which gets the response from METU Infrastructural Services, METU Online Services or METU Information System.
getMessageDetails	Gets the message details from Input Handler class to use it in the queries.
makeConnection	Makes connection to DB server.
executeQuery	Executes the given query.
getQuery	This method is called, when another class , for example a class admin implements for testing and maintenance, needs to obtain the query.
getMessageStatus	Error handler gets the response from the Output Handler class to store it in the

	GarconDB.
logRecogError	If a recognition error occurs, this method logs this error into GarconDB.
logAuthError	If an authorization error occurs, this method logs this error into GarconDB.
getErrorType	Return the type of error.
getError	Returns the error itself.
trainModel	Trains Microsoft Azure Language Understanding with the given configurations.
testModel	Tests Microsoft Azure Language Understanding with the given configurations.
createCrossValidationFolds	Divides the training data into the specified number of folds to be used in cross validation state.
preprocess	Preprocesses the raw training data.
getRawData	Returns the raw training data. This method is when admin implements a class for testing and maintenance
getPreprocessedData	Returns the preprocessed training data. This method is when admin implements a class for testing and maintenance
getConfigurations	Returns the configurations which are set by the trainModel method.
getModelSelections	Returns selected model for training and its configurations.
getWordErrorRate	Determines and returns the word error rate which is an essential metric for text recognition.
getSentenceAccuracy	Determines and returns the sentence accuracy which is an important metric for text recognition.

getCommandAccuracy	Determines and returns the command accuracy which is an important metric for text recognition.
getNumberOfFalsePos	Determines and returns the number of false positives.
getNumberOfFalseNeg	Determines and returns the number of false negatives.
getNumberOfTruePos	Determines and returns the number of true positives.
getNumberOfTrueNeg	Determines and returns the number of true negatives.

Table 7: Operation Descriptions

Operation	**Inputs**	**Outputs**	**Exceptions**
topIntent	List of LUIS Responses	Result with the highest confidence score	-
processIntent	LUIS Response	-	Recognition error
sendTicket	-	-	Connection error occurs
sendInformation	DeviceID	-	Connection error occurs
checkRecognition	-	Recognition OK or failed	Recognition error occurs
isAltered	-	True if message is altered by Bing Spell Check v7 else false	-
getIntent	-	Intent of the message	-
getScore	-	Confidence score of the extracted	-

		intent	
getResponseRate	-	User rating of the response quality	-
setResponseRate	User rating of the response quality	-	-
getMessage	Textual input	-	Connection error occurs
sendToAzure	-	-	Connection error occurs
getMessageID	-	Message ID	-
getMessageLength	-	Length of the message	-
getMessageTime	-	Exact time of the message (DD/MM/Y and time)	-
checkWhichDevice	-	Smartbox or mobile	-
getSmartBoxID	-	Smartbox ID	-
getLocation	-	Location of the message record	Connection error occurs
receiveMessageStatus	Status of the message	-	Authentication error occurs
waitForInformation	Status of the message	-	Connection error occurs
raiseAuthError	Error code for authorization error	-	Authentication error occurs
raiseRecogError	Error code for recognition error	-	Recognition error occurs
returnSuccess	Intent of the message	Success code	Connection error occurs
getResponse	-	Structure of InfoResponse	-

authenticate	UserID, Password	Success or not	Authentication error occurs
grantNonMemberAccess	Member Status	Granted or not	Connection error occurs
getCertificate	-	AuthCertificate	-
getLUISresponse	-	LUISResponse	Connection error occurs
getInformationResponse	-	InfoResponse	-
getMessageDetails	-	MessageDetails	-
makeConnection	-	Success or not	Connection error occurs
executeQuery	Query	-	Connection error occurs
getQuery	-	Query String	-
getMessageStatus	-	Message Status	-
logRecogError	-	-	Connection error occurs
logAuthError	-	-	Connection error occurs
getErrorType	-	Error MessageType	-
getError	-	Error Message	-
trainModel	Training configurations	-	Connection error occurs
testModel	Testing configuratinos	-	Connection error occurs
createCrossValidationFolds	Number of folds	-	-
preprocess	-	-	-
getRawData	-	Data	-

getPreprocessedData	-	Processed data	-
getConfigurations	-	Configurations	-
getModelSelections	-	Model name	-
getWordErrorRate	-	Word error rate	-
getSentenceAccuracy	-	Sentence accuracy	-
getCommandAccuracy	-	Command accuracy	-
getNumberOfFalsePos	-	Number of false positives	-
getNumberOfFalseNeg	-	Number of false negatives	-
getNumberOfTruePos	-	Number of true positives	-
getNumberOfTrueNeg	-	Number of true negatives	-

Table 8: Operation Design

Design Rationale:

- Input Handler is responsible for the input. It sends the details (messageID, messageLength, messageTime...) of the input message to the DB Entry Manager. Moreover, if the message is sent from smartBox, it sends the smartBoxID to the DB Entry Manager, or if the message is sent from mobile device, it sends the location information of the mobile device to the DB Entry Manager.
- DB Entry Manager is responsible for managing the connection between database and the service interfaces. It receives message details from Input Handler and information response from Output Handler. It sends LUIS Response to Intent Processor in order to process the intent. Other interfaces are able to query information from database through this interface.

- Intent Processor is responsible for processing the message according to its intent. Intent Processor is the interface decides whether a message is an information or a ticket. After recognition of the message, it sends an information or a ticket to corresponding units.
- Output Handler is responsible for the output. It sends the response message to corresponding components of the system. Also It keeps the messageStatus in order to send it to Error Handler if there is any failure.
- Error Handler is responsible for handling the errors. errorType information is kept in order to know whether the error is a result of recognition failure or authorization failure or system failure. It receives message status from output handler since if the status is failure, an error log must be created and kept to be fixed later.
- Authorization Handler is responsible for authorization of the users. It decides whether a user who gives input to the system is valid or not.
- Azure Trainer is responsible for training Microsoft Language Understanding in machine-learning manner in order to maintain system reliability. It takes rawData, preprocessedData and folds to create testing data and train the selected model accordingly.

4.4.2. Internal Interfaces

In this section, we provide details regarding internal interfaces of the Garcon system and discuss the design rationale for each of them.

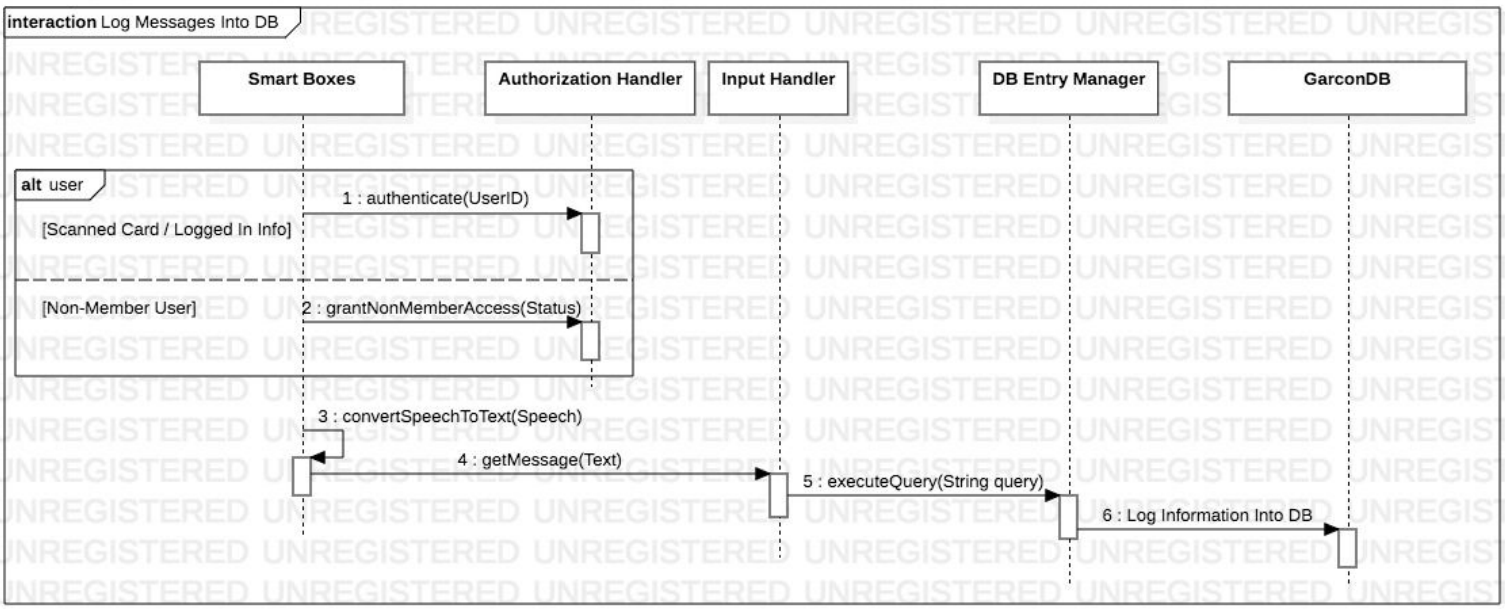

Figure 7: Log Message into DB Sequence Diagram

- **The Interface between the Smart Boxes and Input Handler:**
  - The Smart Boxes component gets the audio input from the user and converts it to text. Then, it directs the textual input to the Input Handler component so that the input can be processed by the system.
  - **Design Rationale:**
    - The Smart Boxes component converts audio input to text and it will be the first point of interaction with the user.
    - The Input Handler is the point where the textual input of the user enters to the system. The Input Handler directs this data to the other components as needed.

- **The Interface between the Smart Boxes and Output Handler:**
  - The Output Handler component gets the output of the requested operation. The output of a requested operation either an error message

indicating the problem or a message indicating that the requested ticket is sent to the corresponding infrastructural unit or a requested information.
    - Then, the Output Handler directs this output to the Smart Boxes component so that the user is notified regarding result of the operation.
    - **Design Rationale:**
        - The Output Handler component produces the resulting output of the requested operation by interacting the necessary components in the system.
        - The Smart Boxes displays the output message to notify the user regarding the result of an operation.

- **The Interface between the Smart Boxes and Authorization Handler:**
    - After a user scanning his/her ID card, the Smart Boxes component sends the necessary information to the Authorization component to check the identity information of the user. The Authorization Handler interacts with necessary external components to check the identity information of the user. If the authorization is successful, the user is informed that he/she can access the system functionality provided to a member user. Otherwise, the user is informed that the provided identity information is not valid; hence, he/she can only access the system functionality provided to a non-member user.
    - **Design Rationale:**
        - The Smart Boxes component sends the identity information of a user to the Authorization Handler component and informs user regarding the result of the authorization.
        - The Authorization Handler interacts with necessary external component to check provided identity information and sends the result of the operation to the Smart Boxes component.

- **The Interface between the Input Handler and DB Entry Manager:**
    - The Input Handler component sends the textual input data to the DB Entry Manager so that it can be stored in the database and later this date is used for training the Microsoft Azure Language Understanding for more accurate input recognition. DB Entry Handler parses this data and merges it with any other data that it collects from other components. Then, these data is combined and directed database.
    - **Design Rationale:**
        - The Input Handler component sends textual input to the DB Entry Handler.

        - The Entry Handler component parses this textual input and creates necessary MySQL queries and runs these queries on the database. Storing textual input is necessary for both training the Microsoft Azure Language Understanding for more accurate input recognition and meeting the legal regulations.

- **The Interface between the DB Entry Manager and GarconDB:**
    - The DB Entry Manager component gets the textual input and parses this text. Then, it creates necessary MySQL queries and runs on the GarconDB to store the textual input in the database.
    - **Design Rationale:**
        - The DB Entry Manager component sends queries to the GarconDB.
        - The GarconDB runs the provided queries and stores the textual input.

- **The Interface between the Azure Trainer and GarconDB:**
    - The Azure Trainer loads all the stored data from the GarconDB and creates a training data for the Microsoft Azure Language Understanding component for more accurate input recognition. GarconDB sends requested data to the Azure Trainer component.
    - **Design Rationale:**
        - The Azure Trainer component requests data from the GarconDB.
        - The GarcoDB component send the requested data to the Azure Trainer.

- **The Interface between the Error Handler and GarconDB:**
    - The Error Handler component processes errors of the system and sends these to the GarconDB for storage. Storing errors is important since system errors are viewed by the system maintainers to improve system performance and functionality.
    - **Design Rationale:**
        - The Error Handler component sends error logs to the GarconDB.
        - The GarcoDB component stores error logs sent by the Error Handler.

- **The Interface between the Intent Processor and Error Handler:**
    - The Intent Processor component notifies the Error Handler if the input is not recognized by the Microsoft Azure Language Understanding

component. This issues an error and the Error Handler handles the error and issues an error log.

- **Design Rationale:**
  - The Intent Processor component informs the Error Handler if the input is not recognized.
  - The Error Handler component issues an error log for the error.

4.4.3. External Interfaces

In this section, we provide details regarding external interfaces of the Garcon system and discuss the design rationale for each of them.

4.4.3.1. User Interfaces

- **Member Interface:**
  - This interface is provided to the member users by either scanning their ID card to the smart boxes in the campus or logging in to the mobile app using their university account. In either way, a user has the same functionality that is provided by the system, i.e., member user privileges. In this interface, users can request detailed information regarding the university and campus life. They can also send tickets to the corresponding university units.
  - **Design Rationale:**
    - This design enables university members to retrieve any information they need and send tickets to the corresponding university maintenance unit.
    - By using mobile app, a user can access the system regardless of his/her current location and by using smart boxes in the campus, a user can interact with the system at the critical campus locations. Notice that, using smart boxes for the interaction may be crucial in emergency cases.

- **Non-Member Interface:**
  - This interface is provided to the non-member users in the university campus. In this interface, non-member users can request general information regarding the university and campus.
  - **Design Rationale:**
    - This design enables non-member users to retrieve general information regarding the campus.

        - The interaction is provided by the smart boxes located at the critical locations of the campus. Non-member users can only use smart boxes for the interaction since they may only need to use the system when they are visiting the campus.

- **System Admin Interface:**
    - In this interface, system admins can view the error logs for maintenance and improvement purposes. They can also view message data stored in the database to train Microsoft Azure Language Understanding for better input recognition.
    - **Design Rationale:**
        - It is crucial that system admins view error logs periodically and improve the system accordingly.
        - Although, the system has a component for training the Microsoft Azure Language Understanding component, it is important that system admins can also view and explicitly intervene the automated training process for a more advanced and informed training of the language recognition component.

4.4.3.2. System Interfaces

In this section, we provide details regarding system interfaces of the Garcon System and discuss the design rationale for each of them.

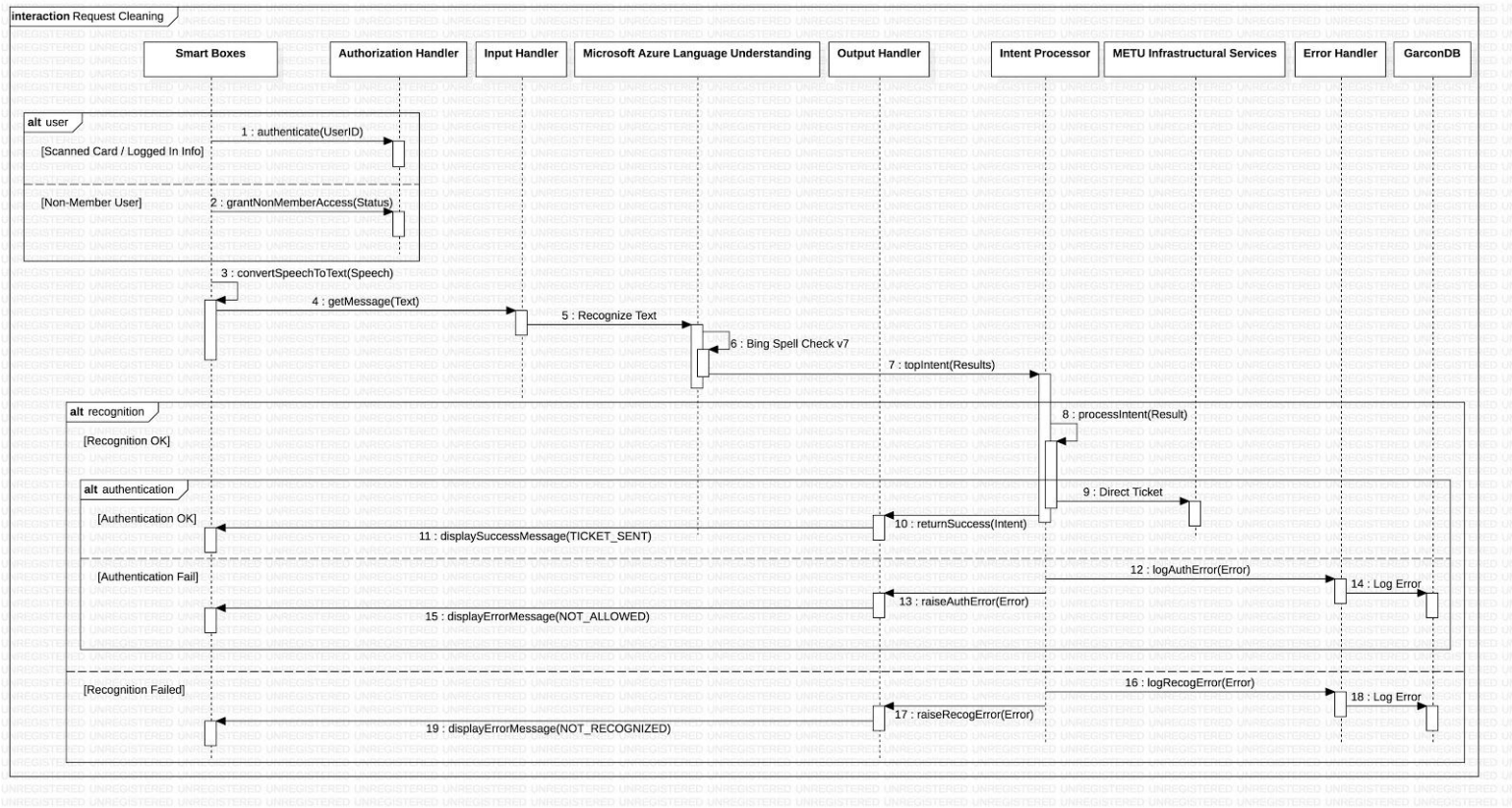

Figure 8: Request Cleaning Sequence Diagram

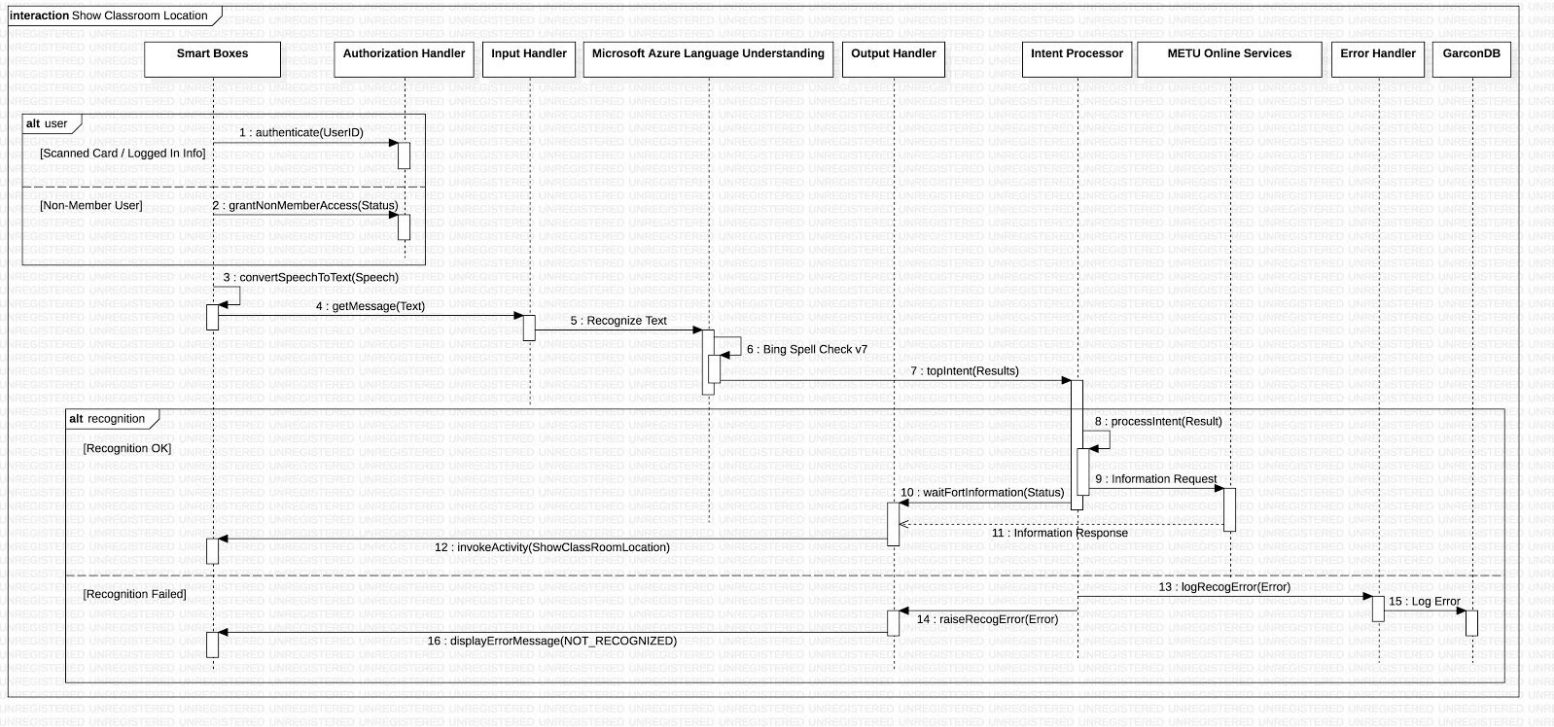

Figure 9: Show Classroom Location Sequence Diagram

- **The Interface Between Mobile App and Authorization Handler**
  - Mobile app gets the login information of the user and directs it to the Authorization Handler in order to get authorized. Authorization Handler is the component which is responsible for authenticating user by interacting with the other external components.
  - **Design Rationale:**
    - The Mobile app component gets the userID and password information of the user.
    - The Authorization Handler component is where user information will be checked whether the user has valid ID and password with the interaction with the METUDB component. If the user has a record in METUDB, Authorization Handler accepts the login activity of the valid user, otherwise it rejects the login activity of invalid user.

- **The Interface Between METUDB and Authorization Handler**
  - Authorization Handler component queries whether the user has a record in METUDB which is a database component filled with user informations.
  - **Design Rationale:**

        - The Authorization Handler component queries the userID and password and waits for the response from METUDB.
        - The METUDB component takes the user query from Authorization Handler, it checks whether there is a record of that user in the database with necessary MySQL queries, then it sends a query result to Authorization Handler.

- **The Interface Between Mobile App and Input Handler**
    - Mobile App component takes the textual input of the user, then directs it to Input Handler to be processed in the system later.
    - **Design Rationale:**
        - User entries a text input to the Mobile App, then the textual input will be directed to the Input Handler.
        - The Input Handler component is the point where the textual input of the user enters to the system. The Input Handler directs this data to the other components as needed.

- **The Interface Between Mobile App and Output Handler**
    - The Output Handler component gets the output of the requested operation. The output of a requested operation either an error message indicating the problem or a message indicating that the requested ticket is sent to the corresponding infrastructural unit or a requested information. Then, the Output Handler directs this output to the Mobile App component so that the user is notified regarding result of the operation.
    - **Design Rationale:**
        - The Output Handler component produces the resulting output of the requested operation by interacting the necessary components in the system.
        - The Mobile App component displays the output message to notify the user regarding the result of an operation.

- **The Interface Between Input Handler and Microsoft Azure Language Understanding**
    - The Input Handler is the point where the textual input of the user enters to the system. The Input Handler directs this data to the Microsoft Azure Language Understanding component to be checked semantically by other components of Microsoft Azure Language Understanding.
    - **Design Rationale:**

        - The Input Handler component directs textual input of the user to the Microsoft Azure Language Understanding component.
        - The Microsoft Azure Language Understanding component first sends the input to the Bing Spell Checker [4] for grammar check. Then, if the text's grammar is valid, it checks the input semantically. If the given input is valid, the processed input will be sent to the other system components.

- **The Interface Between Input Handler and DB Entry Manager**
    - The Input Handler directs textual input of the user to the DB Entry Manager which is a component gets the textual input and parses text input. Then, it creates MySQL queries and runs on the other necessary system components.
    - **Design Rationale:**
        - The Input Handler component directs textual input of the user to the DB Entry Manager.
        - The DB Entry Manager component takes the textual input from the Input Handler then saves the necessary informations into the database.

- **The Interface Between Azure Trainer and Microsoft Azure Language Understanding**
    - Azure Trainer takes message data from GarconDB, creates training data to train and improve Azure Language Understanding.
    - **Design Rationale:**
        - The Azure Trainer component creates training data from gathered data of the other system components, then triggers the Microsoft Azure Language Understanding to be trained.
        - The Microsoft Azure Language Understanding component is an external component of the system which is used for understanding the semantic meaning of the textual input of the user.

- **The Interface Between Intent Processor and Microsoft Azure Language Understanding**
    - Microsoft Azure Language Understanding creates LUIS Response according to the user input, then sends it to the Intent Processor in order to be processed whether as a ticket or information data.
    - **Design Rationale:**

        - The Microsoft Azure Language Understanding component takes textual input then creates LUIS Response and sends it to the Intent Processor.
        - The Intent Processor component is a point which decides whether the given data is classified as ticket or information. Then sends this to the necessary components.

- **The Interface Between DB Entry Manager and Microsoft Azure Language Understanding**
    - Microsoft Azure Language Understanding creates LUIS Response according to the user input. The DB Entry Manager receives the LUIS Response which is created by Microsoft Azure Language Understanding.
    - **Design Rationale:**
        - The Microsoft Azure Language Understanding component takes textual data and creates LUIS Response according to the data.
        - The DB Entry Manager takes the LUIS Response and saves the necessary information to the database as an entry.

- **The Interface Between DB Entry Manager and Garcon DB**
    - The DB Entry Manager component saves the message data entries in itself. In order to improve system quality, the Garcon DB component receive this data from DB Entry Manager to save message data.
    - **Design Rationale:**
        - The DB Entry Manager component saves message data and its status.
        - The Garcon DB component receives message entry and saves it.

- **The Interface Between Azure Trainer and Garcon DB**
    - The Azure Trainer component sends a message query to the Garcon DB in order to improve recognition and train Microsoft Azure Language Understanding.
    - **Design Rationale:**
        - The Azure Trainer component queries the message and waits for the message data.
        - The Garcon DB component receives a message query from the Azure Trainer component, it checks whether there is a record of the given message data, then it sends the message data back to the Azure Trainer.

- **The Interface Between Intent Processor and METU Information System**
    - If the LUIS Response of a message is indicating a information request concerning the METU Information System, the Intent Processor component notifies the METU Information System regarding the request. After acknowledging the request, the METU Information System sends requested data to the necessary components of the system.
    - **Design Rationale:**
        - The Intent Processor component directs information request to the METU Information System.
        - The METU Information System receives the request and sends the data.

- **The Interface Between Error Handler and METU Information System**
    - The METU Information System sends message status of a request to the Error Handler. If there is an error, the Error Handler issues an error log and sends it to the database for storage.
    - **Design Rationale:**
        - The METU Information System component sends message status to the Error Handler.
        - The Error Handler receives the message status and issues an error log if needed.

- **The Interface Between Output Handler and METU Information System**
    - The METU Information System sends result of a request to the Output Handler. Then, the Output Handler directs this output to the necessary components in order to notify the user.
    - **Design Rationale:**
        - The METU Information System component sends result of a request to the Output Handler.
        - The Output Handler receives the results and directs it to other components.

- **The Interface Between DB Entry Manager and METU Information System**
    - The DB Entry Manager receives the message status data from the METU Information System. Message status data indicates whether a message is acknowledged by the METU Information System. The data is attached to its message with other necessary data and sent to the database for storage by the DB Entry Manager.
    - **Design Rationale:**

        - The DB Entry Manager component receives message status data from the METU Information System.
        - The METU Information System notifies DB Entry Manager regarding the status of a message.
- **The Interface Between Output Handler and METU Online Services**
    - The METU Online Services sends result of a request to the Output Handler. Then, the Output Handler directs this output to the necessary components in order to notify the user.
    - **Design Rationale:**
        - The METU Online Services component sends result of a request to the Output Handler.
        - The Output Handler receives the results and directs it to other components.

- **The Interface Between DB Entry Manager and METU Online Services**
    - The DB Entry Manager receives the message status data from the METU Online Services. Message status data indicates whether a message is acknowledged by the METU Online Services. The data is attached to its message with other necessary data and sent to the database for storage by the DB Entry Manager.
    - **Design Rationale:**
        - The DB Entry Manager component receives message status data from the METU Online Services.
        - The METU Online Services notifies DB Entry Manager regarding the status of a message.

- **The Interface Between Error Handler and METU Online Services**
    - The METU Online Services sends message status of a request to the Error Handler. If there is an error, the Error Handler issues an error log and sends it to the database for storage.
    - **Design Rationale:**
        - The METU Online Services component sends message status to the Error Handler.
        - The Error Handler receives the message status and issues an error log if needed.

- **The Interface Between Intent Processor and METU Online Services**
    - If the LUIS Response of a message is indicating a ticket or an information request concerning the METU Online Services, the Intent Processor

component notifies the METU Online Services regarding the request. If it is an information request, METU Online Services sends the requested information to the corresponding component. If it is a ticket request, after acknowledging the ticket, the METU Online Services sends the result to the corresponding component of the system.

- **Design Rationale:**
  - The Intent Processor component directs request to the METU Online Services.
  - The METU Online Services receives the request and acts accordingly.

- **The Interface Between Intent Processor and METU Infrastructural Services**
  - If the LUIS Response of a message is indicating a ticket request concerning the METU Infrastructural Services, the Intent Processor component notifies the METU Infrastructural Services regarding the requested ticket. After acknowledging the ticket, the METU Infrastructural Services sends component the result to the corresponding component of the system.
  - **Design Rationale:**
    - The Intent Processor component directs requested ticket to the METU Infrastructural Services.
    - The METU Infrastructural Services receives the ticket and acts accordingly.

The Interface Between Output Handler and METU Infrastructural Services

- The METU Infrastructural Services sends result of a ticket to the Output Handler. Then, the Output Handler directs this output to the necessary components in order to notify the user.
- **Design Rationale:**
  - The METU Infrastructural Services component sends result of a request to the Output Handler.
  - The Output Handler receives the results and directs it to other components.

The Interface Between Error Handler and METU Infrastructural Services

- The METU Infrastructural Services sends message status of a ticket to the Error Handler. If there is an error, the Error Handler issues an error log and sends it to the database for storage.
- **Design Rationale:**

        - The METU Infrastructural Services component sends message status to the Error Handler.
        - The Error Handler receives the message status and issues an error log if needed.

The Interface Between DB Entry Manager and METU Infrastructural Services

- The DB Entry Manager receives the message status data from the METU Infrastructural Services. Message status data indicates whether a message is acknowledged by the METU Infrastructural Services. The data is attached to its message with other necessary data and sent to the database for storage by the DB Entry Manager.
- **Design Rationale:**
    - The DB Entry Manager component receives message status data from the METU Infrastructural Services.
    - The METU Infrastructural Services notifies DB Entry Manager regarding the status of a message.