INF5120 - Assignment 3

Smart Home – Support services for Elderly being able to stay longer at home. Node-RED Design – ArchiMate Application/Technology layers/views, also with UML/BPMN

Adnan Alisa - adnanali@ifi.uio.no
An Lam - an.n.lam@hiof.no
Lucas Paruch - lucasp@ifi.uio.no
Tanusan Rajmohan - tanusanr@ifi.uio.no

University of Oslo

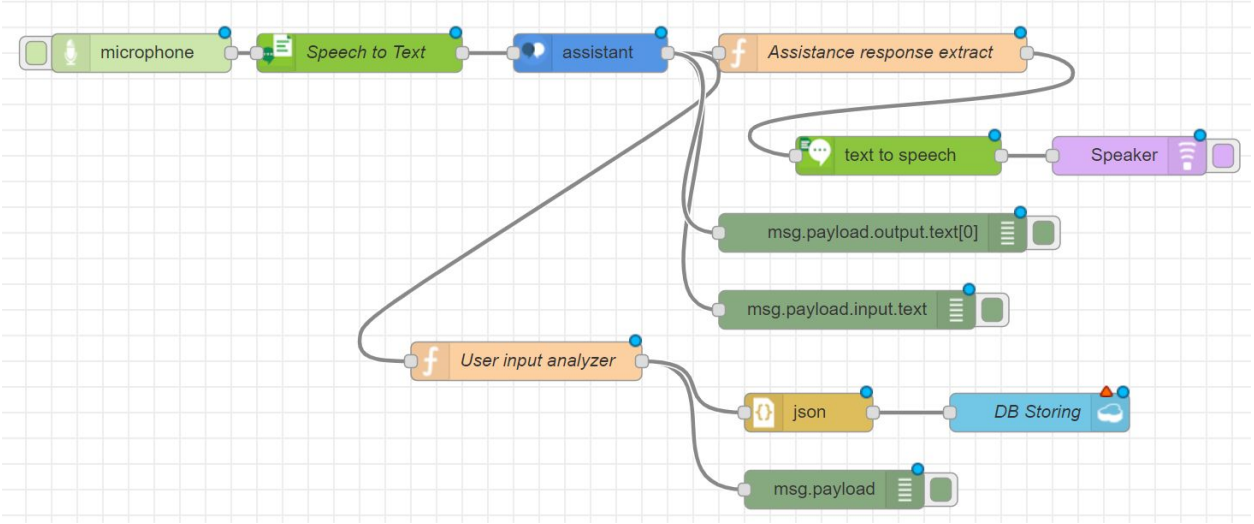
Spring 2019

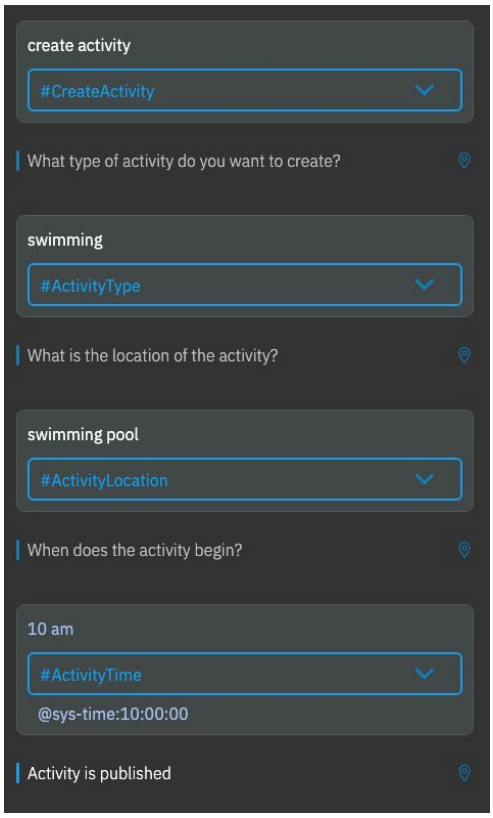
Task 1)

NB! In order for Node-RED to function properly, a new set of valid login credentials and API keys are required for IBM assistant, speech to text, Cloudant database, and text to speech. The speech to text function had a limit of 10 000 characters, which were quickly used up during testing.

Create Activity

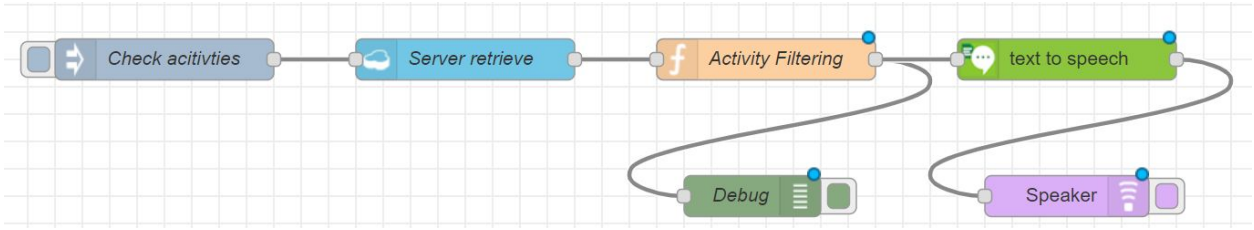
This flow represents the scenario where the end user broadcasts information about the Activity to other users within a specific proximity. The information includes activity type, time and location. These data are gathered with the support of IBM assistance (a.k.a. Chatbot) as shown in the subsequent figure. These data will then be stored in IBM cloud storage and broadcasted to relevant users.





Activity Notification

This flow implements a scenario where the Activity data are periodically retrieved from the IBM cloud storage, filtered for only latest Activity Information, and communicated to the end user under the form of audio format with the support of IBM Text To Speech.



Task 2 & 3)

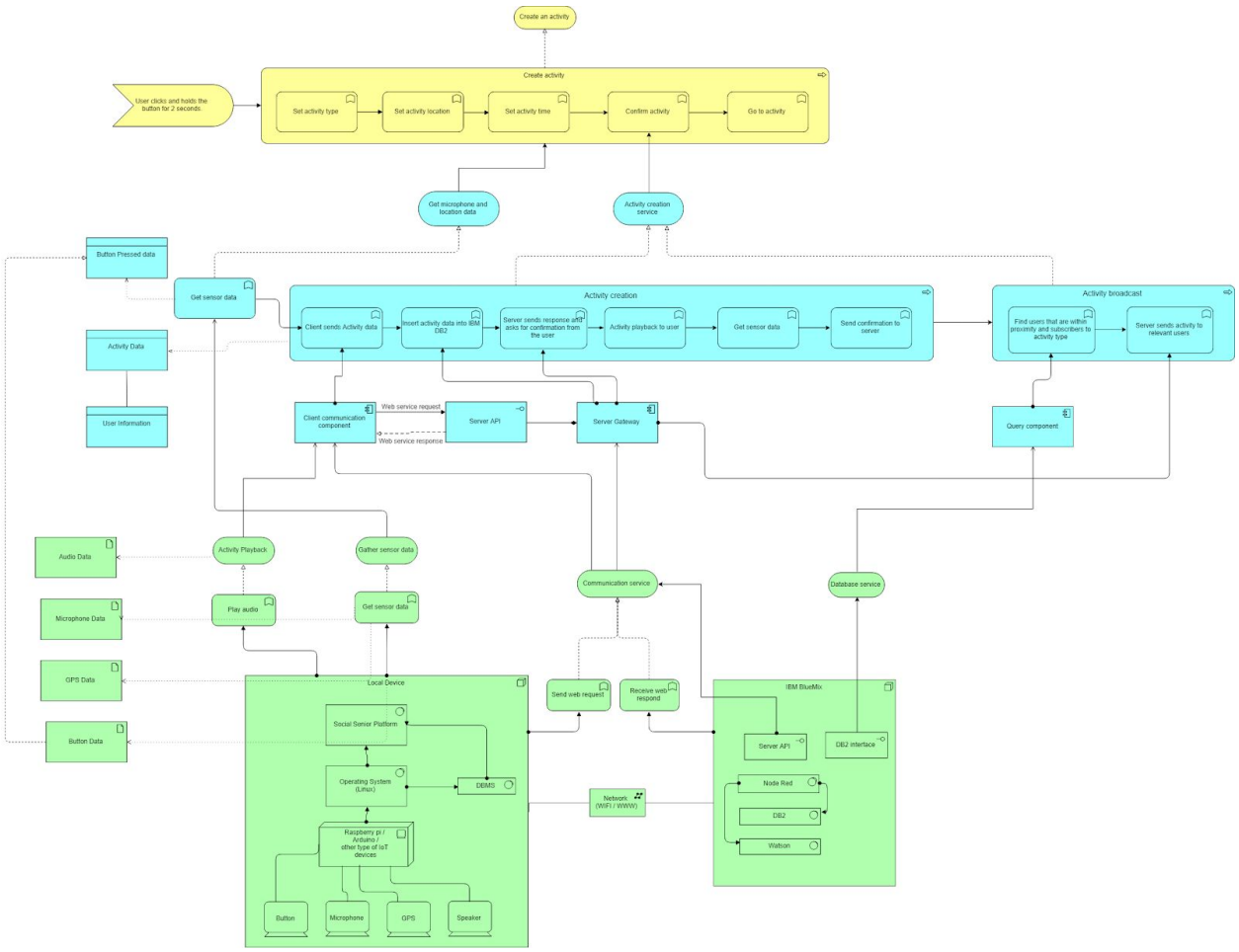
The node functions represent our technology layer, and technology flow is the processes within the Node-RED. The local-device is the IoT device we want to create in this project, so this will be the device with sensors and some type of program/software.

Create an activity - user gets a notification from their own activities, but it is not modeled.

Create Activity

In this use case, our system provides users with the ability to create activities. An activity is an event that other users can participate in. For example, walking in the park could be an activity. Activities have three properties, activity type, activity location and activity time. All three are set by the user using Watson Assistant.

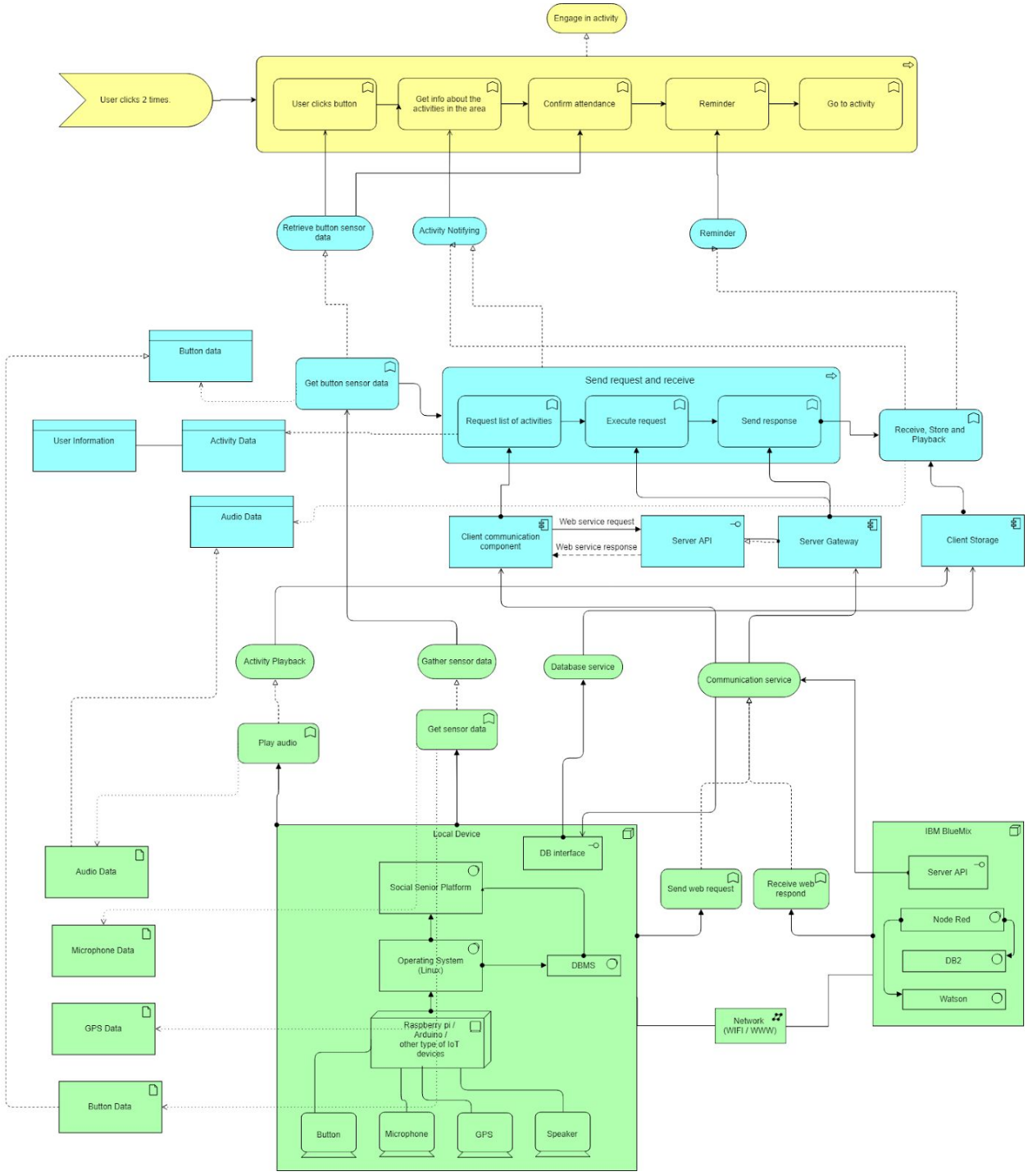
User clicks and holds the button for 2 seconds to indicate that he/she wants to create an activity. This is illustrated as a business event in our diagram. The user sets all three activity parameters, which is the activity data that is sent as a web service request to the server using the server API. The activity data will be inserted into the database. After the activity is created and sent, the local device will play back the activity information of the newly created activity to the user. The user then confirms that the information is correct, and would then attend the event.



Activity Notification

This use case regards the situation where the user is engaged in the activity. By this, we mean the following steps:
1) The user asks for a list of relevant activities within his area.
2) Confirms that he is attending the activity.
3) A reminder is sent to the user shortly before the activity start.
4) The user goes to the activity.

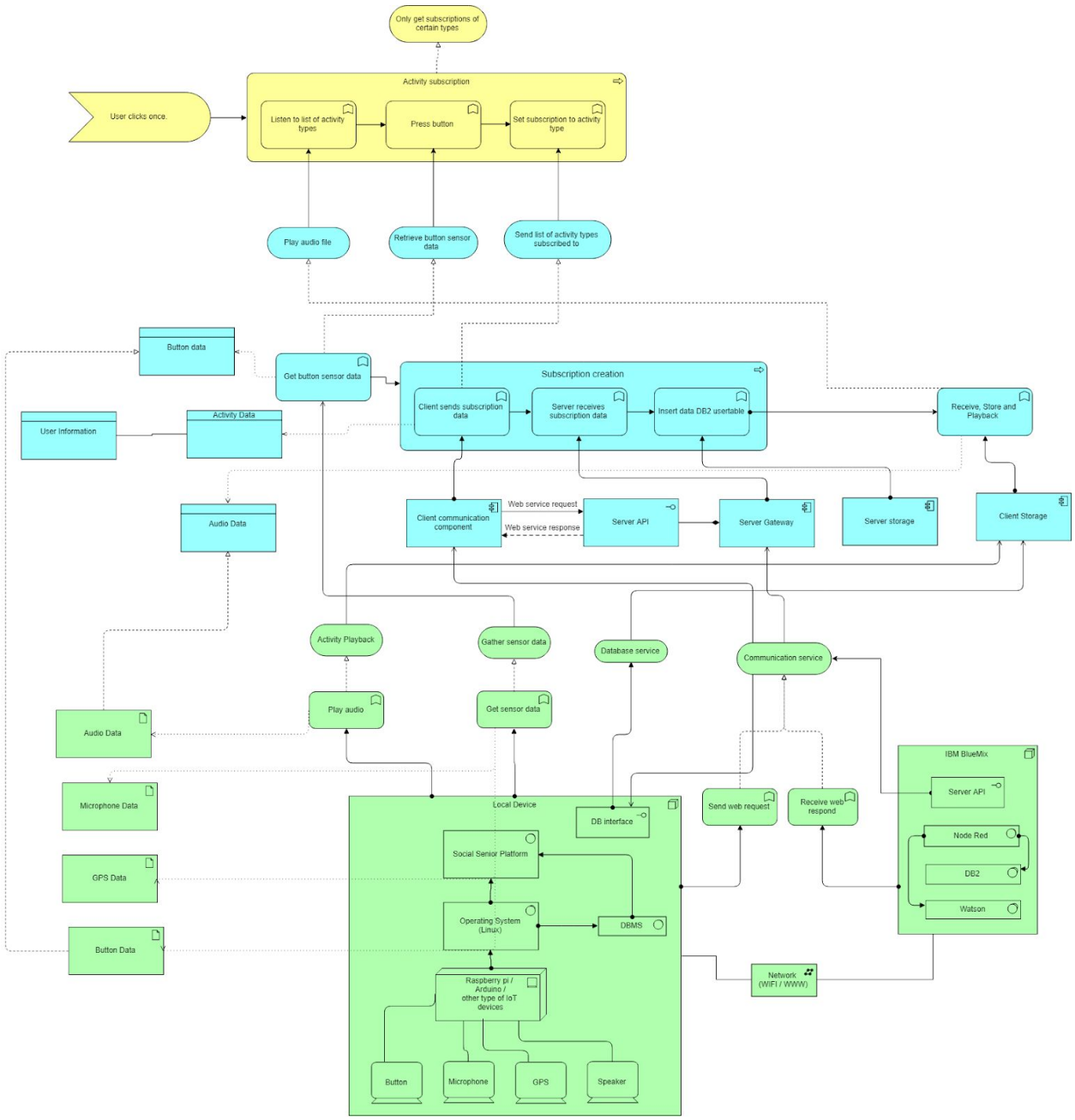
The user clicks 2 times to indicate that he/she wants information about activity nearby. This is illustrated as a business event in the diagram. A request is then sent to the server, which then the server finds the activities with the activity types the user has subscribed to that are within the area of the user. All these activities are sent as a list back to the local device. The local device then goes from text to speech and plays the list to the user so he can hear the activities. The playback will play one activity at a time. The user may confirm attendance by clicking the button once. When the user confirms the event, a reminder will be created by the reminder service. After the user listens to the reminder, he can attend the activity.



Activity Subscription

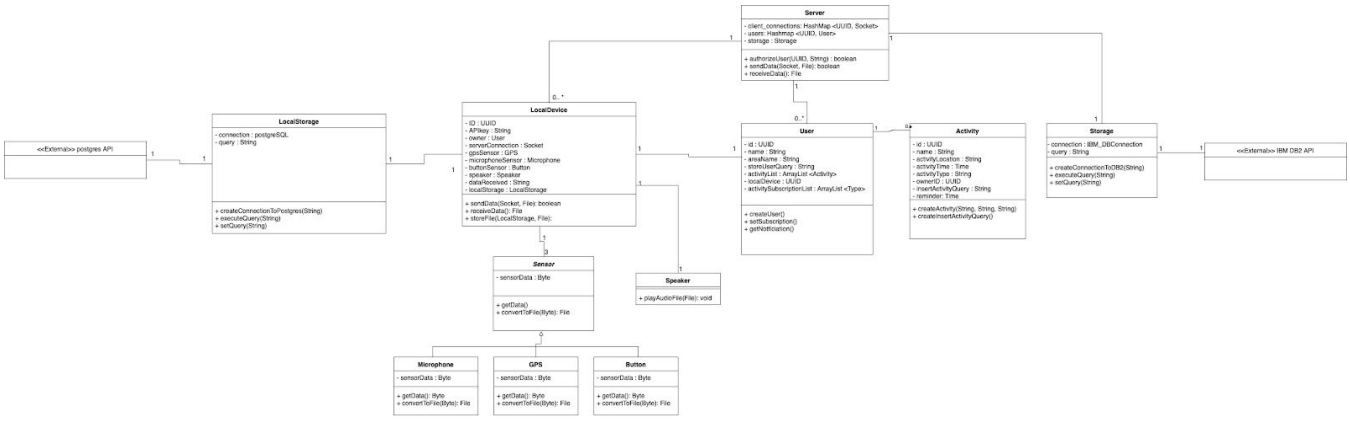
To make it possible for users to subscribe to *different* types of activities, our system has a subscription feature. Different types of activities are a way of categorizing activities. Swimming, running, and eating would be examples of different types of activities. This way the user won't

get activity notification for activities that he is not interested in. The user clicks once to indicate that he wants to listen to the different activity types. Each type is played to the user one by one. If the user presses the button after hearing one activity type, it's an indication that he wishes to subscribe to that specific activity type. The subscription data is sent to the server as a web service request using the server API. The server then inserts/adds the activity type to the user database.



Task 4)

Below you see the UML Class diagram for our system. It consists of both the local device and the server. Within the local device, we have three sensors and a speaker, sensors can retrieve data which will convert data into File-objects. A local device has a unique ID, an API key to access the server, User-object in order to identify the owner. The local device object has a Localstorage object within so it can store sensor data and activities of the user by using the storeFile() function. The device is connected to the server via a Socket-object in order to send and receive messages. The server can be connected to many devices, but the local device can only be connected to one server. When a local device wishes to connect to the server, the server will authenticate the local device by using the function authorizeUser. A server contains many users, and users can have many activities. When an activity is created by the user, the activity variables are set and put into the createInsertActivityQuery() function to create a query. The query is then sent to the database via the server which has an ibm_dbConnection Object to the database. The storage class uses the IBM DB2 API.
The UML Class diagram does not go into details of the system in terms of IBM BlueMix and Node Red since this would clutter the diagram, therefore we chose to leave it out.

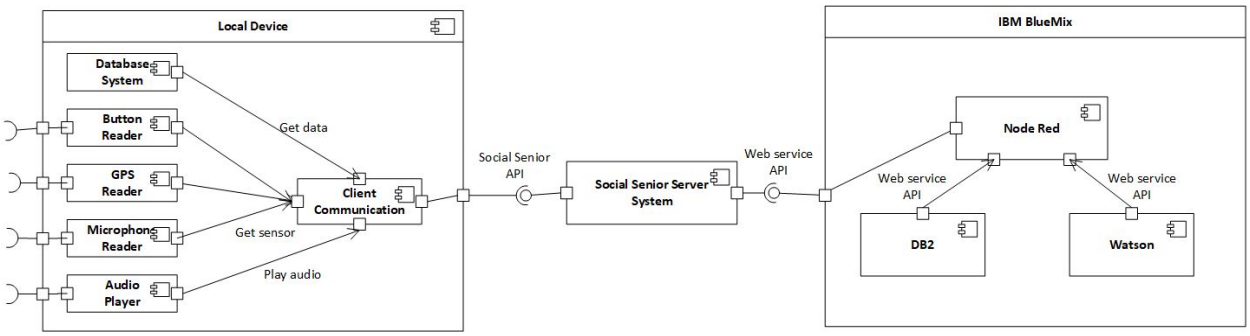


Also, the attributes of the data object in the application layer (e.g., Activity, User Information, etc.) as well as the artifacts in the technology layer (e.g., Microphone Data, Button Data, etc.) are also described in details in this UML model.

Task 5)

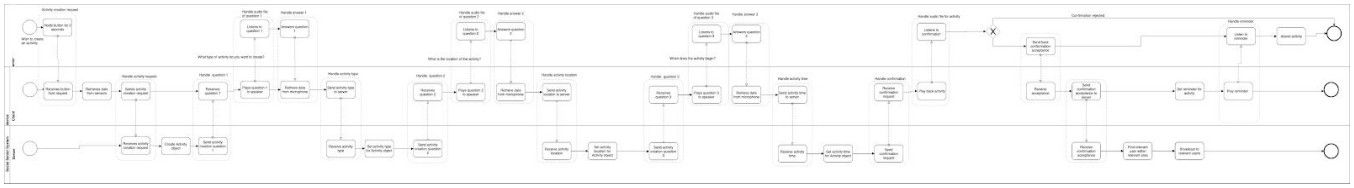
The composite diagram below illustrates the connections among different application systems of our Social Senior Solution. Accordingly, the Local Device represents the software deployed on the Local device (e.g., Raspberry Pi, Arduino or something similar) that uses different sensors (GPS, button, microphones) to interact with the end user. This component corresponds to the

Client Communication Component of the Application Layer in the above archimate diagrams. The Social Senior Server System corresponds to the Server Gateway component from the Application Layer in the above archimate diagrams, implements the functionality from the server side. This system employs different technologies from IBM (e.g., cloud storage, text to speech, speech to text, etc.) and provides a Social Senior API which will then be called by the Local Device to implement different use cases as presented above.



Task 6)

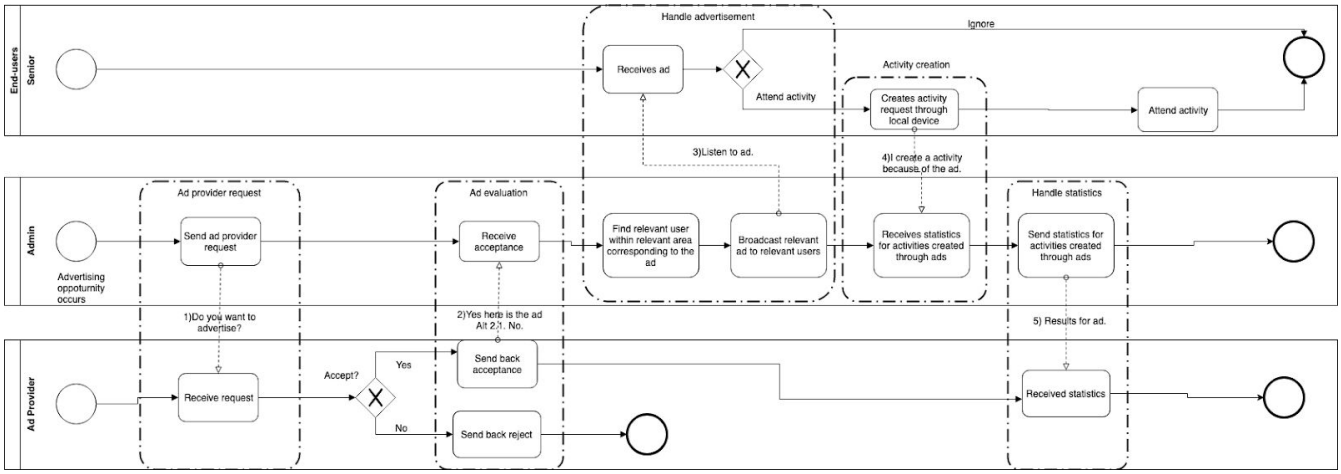
We've made the BPMN for the use case 'Create activity'. This is previously explained in Task 2 & 3.



(See appendix for larger image)

Task 7)

As BPMN mainly focuses on the business aspect, we've decided to create an additional BPMN diagram that shows how ad-providers and admins may interact with each other to include advertisements into the local devices used by the user. We argue that the diagram is worth making because it shows how other different actors interact in the big picture.



Task 8)

The challenges for us related to how to handle the evolution of models/specifications over time can be getting contradictory feedback, not knowing which feedback is good or bad. This is already something that we already have encountered during our visit to the elderly home. Where some elderly people wanted a really simple solution with no screen. While others wanted a more advanced device with a touch screen like a smartphone. Another challenge is that text-to-speech is not good enough and ready for production use. We would have to implement something similar to TensorFlows version of text-to-speech. For our prototype, we are using a NoSQL database called Cloudant. But for our future models, we would like to use relational Db, see appendix file “oblig3.sql”.

See separate document for improvement/changes (Oblig-1-final)