Otto-von-Guericke University Magdeburg

Faculty of Computer Science(FIN)

&

Fakultät für Elektrotechnik und Informationstechnik (FEIT)

Interdisciplinary Team Project

LSF Room Service - An Alexa Skill

Authors:

Athul Raj Vattappilly Sunilkumar	Sini Manu Sali	Freddy Eldhose
athul.vattappilly@st.ovgu.de	sini.manu@st.ovgu.de	freddy.eldhose@st.ovgu.de
Matrikel Nr: 224311	Matrikel Nr: 225887	Matrikel Nr: 224633

January 04, 2021

Advisors:

Supervisor 1 :

Jun.-Prof. Dr.-Ing. Ingo Siegert
Institut für Informations- und
Kommunikationstechnik (IIKT)
Universitätsplatz 2
39106, Magdeburg, G03-323

Supervisor 2 :

Prof. Dr. rer. nat. Frank Ortmeier
Fakultät für Informatik (FIN)
AG Software Engineering
Universitätsplatz 2
39106, Magdeburg, G29-404

Abstract

The aim of this project is to develop an Alexa Skill, a Voice User Interface (VUI) to the LSF Room Reservation web portal of Otto-von-Guericke University, Magdeburg. LSF is a web application that helps to provide information about the university, courses, study events, the structure, and organizational information for teaching and non- teaching staffs, students, research and university members. The VUI works on Speech recognition, which can be defined as the capability of a machine or program to interpret the words or phrases that it takes as input, in order to perform a task or execute an operation to feed an appropriate response back to the user. At present, several applications exist in the market that recognizes the user-speech and executes an intended task. For instance, Google Assistant, Siri, Amazon Alexa.

The developed Alexa skill namely, 'LSF Room Service' target user accessibility by retrieving information in a fast, intelligent and efficient manner. The development and implementation of the project was attained using Amazon Alexa platform – The Alexa Developer Console. The Application Programming Interface (API) of the Amazon Alexa platform enables to implement the whole project with aid from cloud-based Amazon Web Services.

The complete working version and the preliminary version of the skill was subjected to user testing for the analysis of its user friendliness, efficiency, accuracy, robustness, and optimization. This helped in further optimization of the skill with the incorporation of additional features, that makes it a smart web search system, contrast to the typical web search. The VUI inferred to function with a success rate of above 90%. The detailed analysis of the success to fail ratio and performance where also accomplished through testing.

Table of Contents

1. Introduction

1.1 Motivation

Voice Assistants (VUI) are becoming part of our daily life and especially in situations wherein information and data are to be retrieved in the fastest manner. This enables the requests to be formulated quite naturally than how it is re-formulated for specific online forms, as they can show their full potential. The major objective of any VUI is accessibility. And taking to account advantages such as easy understandability, cost effectiveness, suitability, mass communication capability and so on, the speech communication mode stands out from the others. In most of the VUIs developed today, it is this advantage that is being considered.

In the early stages of development, the application of VUIs were kept simple, such as voice dialling in through different embedded systems. However, as soon as various skilled and dignified developers spotted its full potential, they started working on it as a result of which more and more Voice Assisted devices were introduced in the market.

Over the years the world witnessed the development of several Speech Recognition Systems like Siri, Google Assistant, Cortana. However, in 2014 a breakthrough was made by the Amazon Alexa enabled smart device. The virtual assistant technology developed by Amazon was first used in Amazon Echo smart speakers. Moreover, the major capabilities include voice interaction capability, handling real time information such as news, music playback etc. But in the scope of this project we look to explore the most interesting and important feature of Amazon Alexa, that a user is able to extend its capabilities by building custom skills.

The skill can be referred to as the capability of Alexa which it can handle on user requests. Therefore, in this project we try to build an Alexa custom skill that handles the LSF room reservation portal of Otto-von-Guericke-Universität Magdeburg. The LSF portal is a common administrative tool that was introduced mainly for teaching staffs, non - teaching staffs and students. Even though it handles functionalities such as registration and de-registration of Lectures, Assignment, Exams, Timetable supervision etc. With this project we aim to develop a VUI for reserving free rooms by communicating with LSF Web portal. This enables user to get information on free rooms available for reservation.

1.2 The Tasks

The objective of this project revolves around the requirement for a 'Virtual Assistant' which helps the user to interact with the intended system via voice control. Therefore, to achieve this goal, it becomes relevant to put in to use Alexa, a virtual assistant AI technology developed by Amazon.

In normal scenario, LSF Portal is used to reserve the rooms online. So, the aim of this project is to get this done using Alexa. Thus, to develop this skill the following tasks must be accomplished as listed below: -

    i.   To understand and cover all the basic requirements to be achieved by the skill development
   ii.   Carry out a survey to gather all the possible questions that could be asked by the user
  iii.   To be able to collect all required information from the user to compute results and feed them back to the user in the form of audio response
  iv.   To store necessary data (room reservation) in the database
   v.   Auto expire the data in the database at the end of reservation time

vi.   Cancel the room reservation

The skill takes voice inputs such as building number, date, time for reservation etc. Using these values, a web scraping algorithm is run on the LSF Portal to get some search results which consists of the details of rooms that are free for reservation. However, these results are also compared against the records that already exists within the database. The results are fed to the user as voice responses so that the user can select the option, he / she wants. Using the option retrieved from the user, a record is sorted out from the results and is saved in the database as a record of room reservation.

Moreover, some important conditions were to be ensured as well: -
- A user can make only a single reservation
- For instance. imagine a room 300 in building 29 is reserved from 08:00 – 10:00 and the record is moved into the database.
    o   The next user if performs a search through LSF portal, the Room 300 in building shall still be listed.
    o   However, the skill should be capable to hide the data from other users.
- Also, major optimizations were carried out to keep this functioning of the skill as simple as possible.
    o   For instance. if a user has made a reservation, on invoking the skill next time while the reservation is still valid, he / she gets back a Welcome message with the reservation details
    o   This enabled to eliminate the requirement to implement an additional intent, just for a user to view his / her reservation details

The above-mentioned approach enabled to keep and maintain database clean. This is because the data exchange between the database and the application was optimized i.e. minimised.
The Alexa skill was developed in the Alexa Developer Console as an Alexa hosted skill (Python environment). This helped in the following ways: -
- Eliminate the usage and definition of custom lambda functions
- Eliminate usage of additional Amazon Web Services (AWS) services, especially paid ones
- Eliminate usage of Dynamo DB which requires AWS paid subscription
- Make use of the advantage of Python over Node.js
    o   Versatility and flexibility
    o   As the web scraping algorithm was performed using the 'Beautiful Soup (BS4)' library in python, this enabled the whole backend coding in a unified same platform

Thorough user testing was performed to ensure that the skill is robust, performs effective and is user friendly. The main functionalities of the skill were planned as three different phases, as below: -
i.   The first phase was to model the functionalities of the skill, to get the queries related to finding availability of free rooms. This scenario too has two sub-phases
    a.   First phase - For example, "find a free room in building twenty-nine on Monday". As can be seen this scenario searches for a free room in a building on a given date and time
    b.   Second phase - For example, "find a free room now in building twenty-nine". While in this scenario the function searches for a free room in a building at any instant
ii.   The second phase was to model the functionalities of the skill, to get the queries related to reserving a room from a given set of available free rooms
    a.   From the first phase user gets the list of free rooms in the form of audio response, from which he / she can decide to reserve a room or not. For example, "I have found the 2 rooms for you. They are .... Do you want to reserve a room?"
    b.   If a user decides to reserve a room, he can respond with a "yes" otherwise "no".
    c.   If the response is a yes, for example, "Specify your choice". For which the user can specify options in terms of numbers or ordinals. The skill selects a room from the given list

according to user response
d.   User can reserve utmost a single room
iii.   The third phase to model the functionality to cancel the reservation. For example, "I want to cancel my reservation". This will search if a user has a reservation in the database and deletes the existing reservation in the name of the user, if any
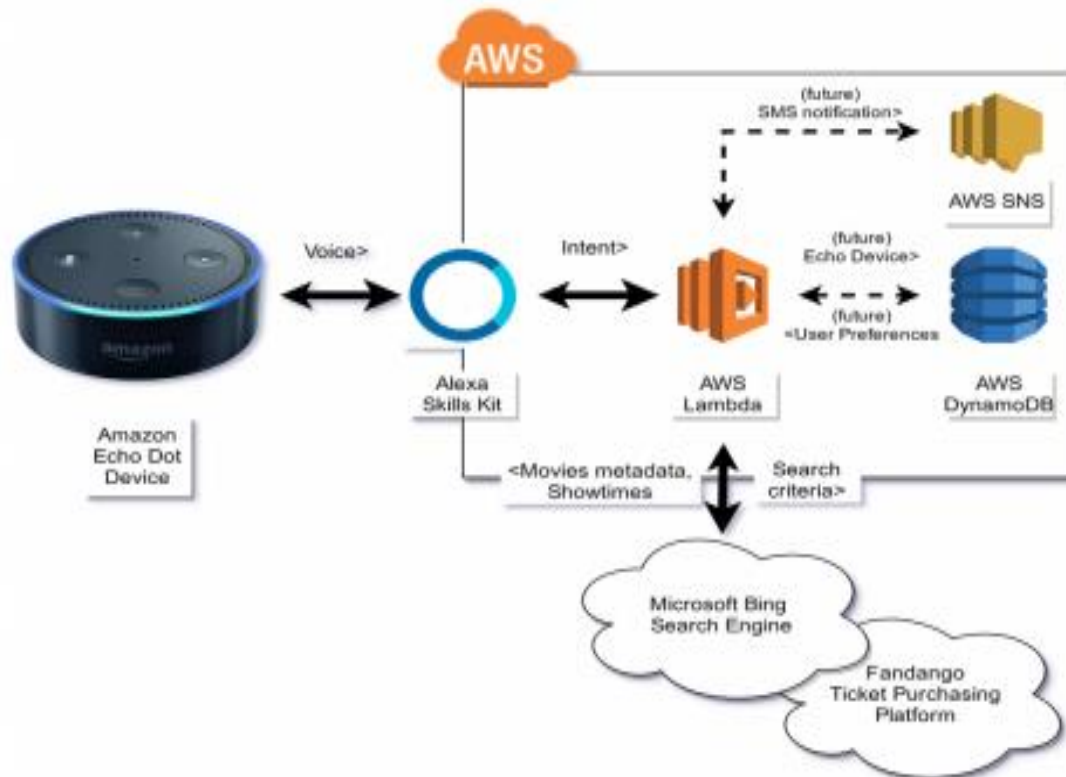
1.3 The Flowchart

Fig. 1 The flowchart of an example use case scenario of Amazon echo dot device [4]

As can be seen in Fig 1, is a sample use case scenario of an Alexa Skill. The following is a brief step by step explanation of the process: -

- The user makes use of the Alexa Echo Dot as a voice assistant device which fetches the voice input for the Alexa Skill Kit.
- This is then mapped to various intent invocation.
- In response to intent invocation, it executes certain AWS Lambda functions, that enables to run some code to compute desired results.
- The Lambda functions could make use of services like Search Engines, HTTP services, API's etc for the same.
- For data handling, it is AWS Dynamo DB that is the most prominently used. This is a No SQL based database, but a wide variety of database could be used apart from the Dynamo DB.
- After computation of the results, the result is fed back to the user through the Alexa device in voice format.

Even though most of the steps are common and similar in case of this project, the major difference comes in how the skill is hosted. In the above scenario its a custom skill hosted using AWS Lambda, whereas in this project its an Alexa hosted skill. The reason for using the latter approach was to simplify the project

development and was seen to suit the most in this case. This could be seen clearly in the brief explanation of both these approaches as below: -

- **Custom skill hosted using AWS Lambda**
    - Enables user to build skill using the cloud-based service for a custom Alexa skill through AWS Lambda, which offers to run code only when necessary, thus this eliminates a need to provision or to run the servers continuously.
    - The code is to be uploaded for the Alexa skill to a Lambda function and the rest is taken by the Lambda, which executes it in response to Alexa voice interactions and manages to compute the resources automatically.
    - This approach eliminates several complexities around setting up and managing endpoints: -
        - Eliminate administration for compute resources for the services.
        - Eliminate the need of SSL certificate.
        - Eliminate the verification of requests coming from the Alexa service. Access to execution of functionalities could be controlled by permissions within AWS instead.
    - More in detail documentation could be found here.
- **Alexa hosted skills**
    - This approach enables to store the code and resources on AWS.
    - With an Alexa-hosted skill, it also enables to quickly make use of the skill templates available in the developer console.
    - The skill is developed in an online ide in the developer console.
    - Also gives the provision use the Alexa Skills Kit Command Line Interface (ASK CLI) for Alexa-hosted skills development and management.
    - It also provisions the following: -
        - AWS Lambda endpoints in all three Alexa service regions [7]
        - An Amazon S3 bucket for media storage [7]
        - An Amazon DynamoDB table for persisting data [7]
        - An AWS Code Commit repository [7]
    - More in detail documentation could be found here.

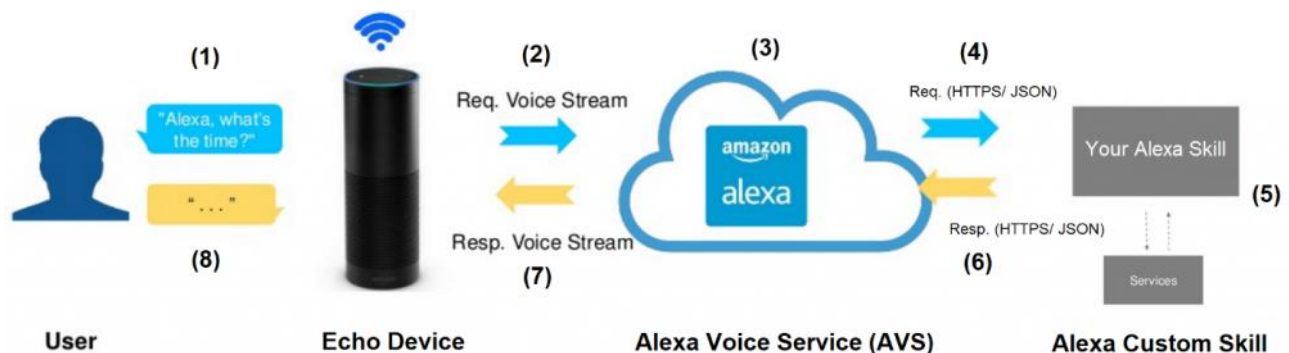This leads to a more general flowchart as can be seen below in Fig 2.

Fig. 2 The flowchart of an example use case scenario of general skills [5]

The differences could be listed as follows: -

- The AWS Lambda block is replaced by the Amazon Skills cloud which provisions for Speech recognition, Machine Learning (ML) algorithms such as Natural Language Understanding (NLP), text to speech etc.
- This also enables to handle HTTPs request - response required for the skill for results

7

computation.
- The response is then fed back to the user in same audio format.

2.Methods

The main objective of the project is to implement the functionalities to make Alexa answer the requests regarding free rooms available for reservation at a given date and time or instant, cancellation of reservation etc. All these methods were implemented through various development phases as mentioned in Section 1.2. The main tasks to be accomplished included user study regarding finding free rooms, reserving free rooms, cancelling the reservations, database preparation and optimization. Moreover, in detail explanation on the Alexa Skill Development and user testing shall be done in this section.
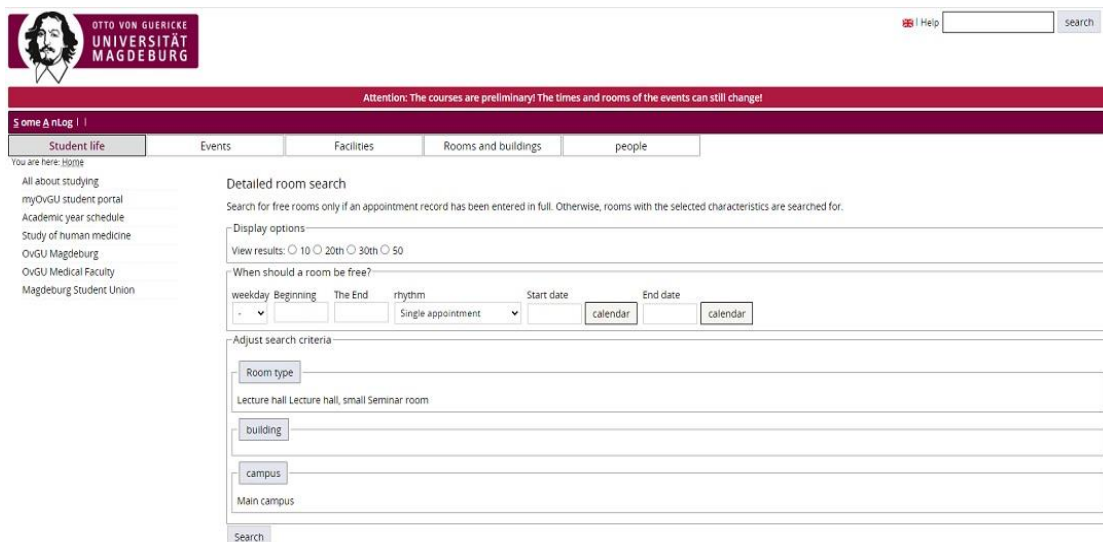
2.1 User Study for Possible Queries

The primary aim of this study is to gather all the possible questions which Alexa should answer. This study was performed with the help of various group of users to draw inference on their interaction with a Voice User Interface (VUI) just as how the interactions would be with a Web Interface. This enabled the users to formulate the web queries to user speech.

To conduct the study, several candidates from different linguistic backgrounds were carefully chosen. For this it had to be made sure that the candidates were frequent regular users of LSF and its various functionalities.

Each candidate was subjected to a set of situations to retrieve all the possible dialog the user might ask Alexa. For instance, 'How would you communicate with Alexa to retrieve information regarding available free rooms in building twenty-nine?', or may be 'How would you try to reserve a room or may be even cancel the reservation' etc. Different responses where retrieved from the users from different part of the world, by conducting small interviews.

At the end of the survey, multiple set of questions from multiple users were formulated and were grouped together. This was later assigned to multiple functional departments of the skill, which is how it was decided, which intent was to be invoked and when. Well all the necessary data for the same was retrieved from the LSF portal.
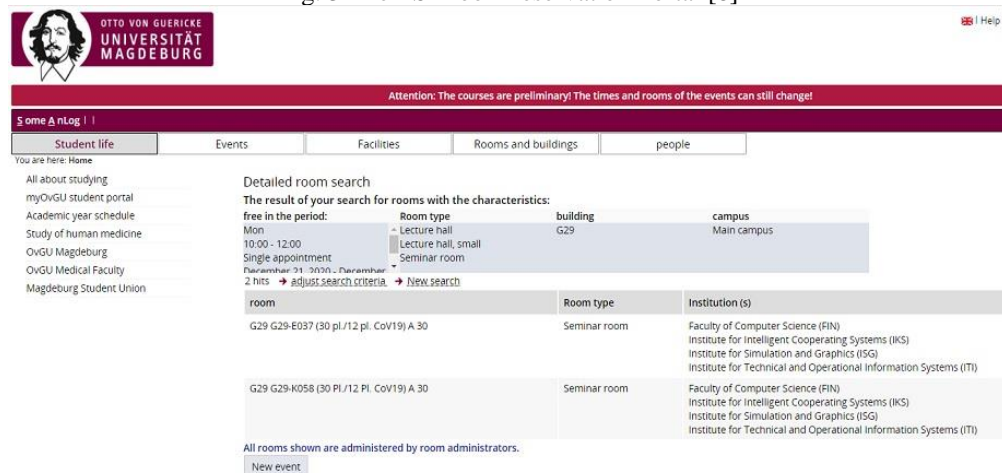
Fig. 3 The LSF room reservation Portal [8]

Fig. 4 The LSF room reservation Portal – sample search result [8]

Fig 4. Shows the details of free rooms available in a specific building with details such as building name, room type, institution etc. It is to be decided which data is to be retrieved from this. The handling of the data is coming in the following sections.

2.2 Web Scraping and Database Preparation

The necessary data required for building the skill, computing results, etc. was accessed directly from LSF webpage. A web scraping algorithm was necessary to scrape this data, to process it, use it for computing the results to feed them back to the user.

2.2.1 Web Scraping and Data Retrieval

As it was decided to go with building Alexa hosted skill, there was option to use Python language – Alexa hosted Skill (Python). On browsing through, it was seen that various web-scraping options are available with Python. After thorough research it was arrived at the decision to use the BeautifulSoup (bs4) library which helped not only to scrape the data, but also to post data on a web page.

On giving necessary input, all the required data are available directly from the LSF room reservation portal. So at first it has to be clear how this portal works to figure out how to implement the various functionalities. Moreover, before retrieving data and using it, it must be processed a bit or cleansed. What this means is to discard unnecessary data and to work on only the good ones.

For instance, in Fig 4, which displays search results, on retrieving data from the same yields a lot of unnecessary white spaces, special characters etc. These are unnecessary and are to be eliminated. Moreover, it is not practical to retrieve and store all of the data into the database or any other storage medium. This approach helps in achieving easier and faster data processing as its accurate and organized.

2.2.2 Getting results

A clear logic was to be developed on how to handle data to obtain results in the most optimal manner. The following steps were taken: -
- The cleansed data was stored in to a dataframe with appropriate columns headings.
- The user id of the user is to be retrieved from the skill to check if there is any reservation existing

9

in the name of the same. This is because only a single reservation can be in the name of a user.
- If there is any reservation for a specific user, then the code should handle the skill in such a way that the reserved room should be hidden from the overall results.
- Moreover, it should also be seen that the results from the LSF should be compared against all the data in the database and only the rest of the data should be shown.
- It is this data that is to be stored in the database.

2.2.3 Database Preparation and Integration

The AWS Dynamo DB is prominently used to store data for the skill. However, this requires an AWS account which is not free. Since the data is handled in an optimal manner and that there is minimal data to be stored beginning AWS account was impractical. Therefore, another alternative was to be found which helps to store data by beginning a free account. Thus, the decision was made to use the free Mongo DB.

2.2.3.1 Preparation

The results dataframe was stored in the MongoDB as document. The following steps.
- The free tier, the M0 tier which gives the capability to store 512 MB was used, as this was more than enough of storage capability.
- A new database instance was defined, in which it gives the capability to various data objects, like tables, documents etc.
- The results dataframe must be converted to JSON (or dictionary) format and can be stored in such a way that the column heading of the dataframe pertains to the title (column) of the data.
- The data stores the user id, room number, buliding number, start time and end time or reservation.
- Lastly, the reservation end time is also set as the time for data auto-expiry. This ensures that the reservation data does not reside in the database after the period of reservation.
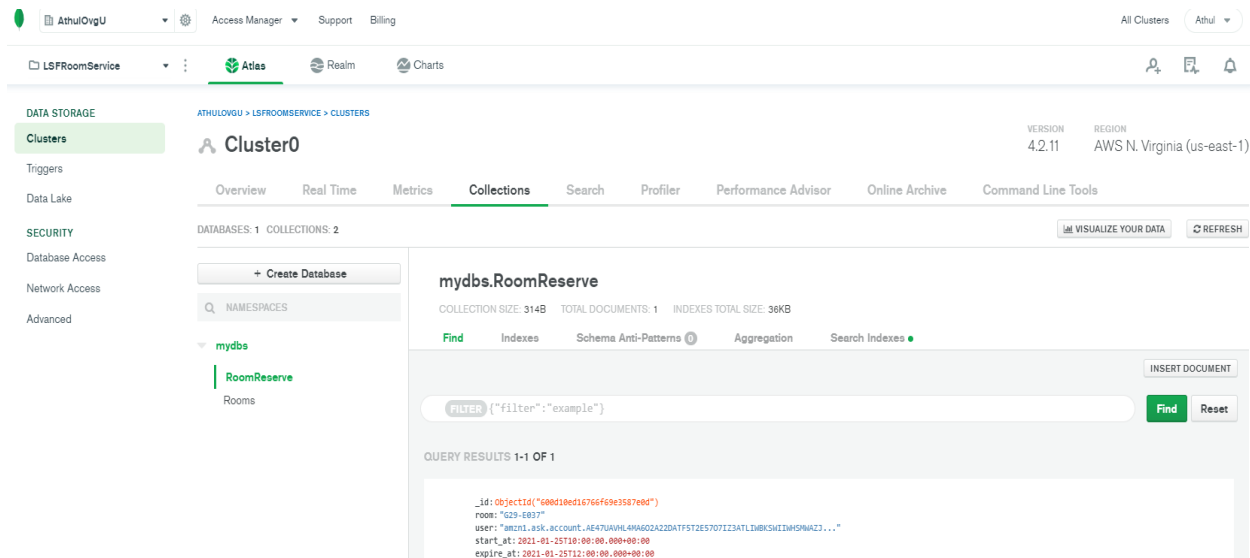- To store the data the authentication credentials are required to be specified in the connection strings.

Fig 5. Data stored in Mongo DB database

2.2.3.2 Integration

The Mongo DB "is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas" [11]. To store

10

data in Mongo DB, a research was done on how to connect to first establish the connection to the same using python. As mentioned in the previous section the M0 tier of the Mongo DB as to be used. The following steps were taken for integration: -

- Initially it is to be signed up for a Mongo DB account.
- After beginning an account these are the steps to be performed manually or through coding
  - o A cluster (collection of databases) definition
  - o A database (collection of tables) definition
  - o A collection (same as tables in RDBMS) definition
- To store data first a connection is to established using connection URL which consists of database URL with authentication credentials.
- IP address is also required to establish connection. Since the skill is accessed from different part of the world its not possible to sort out all the IPs. Thus, the IP were by-passed.
- The PyMongo library was used for connection establishment by defining a MongoClient.
- This library gives all the operation such as insert, update, deleted that could be performed on a typical RDMS database.
- The detailed documentation could be found at [9] and [10]

2.3 Skill Configuration and Modelling with Alexa Developer Console

"The Alexa Skills Kit (ASK) is a collection of self-service APIs, tools, documentation and code samples that you can use to develop skills quickly and easily. Skills are like apps, only for Alexa and allow your customers to do everyday tasks or to interact with your content naturally through their voice." [13]

To set up and configure Alexa skill in Alexa Developer Console the following steps need to be taken: -

- Log in to Alexa Developer Console using Amazon account credentials by traversing to "https://developer.amazon.com/alexa/console".
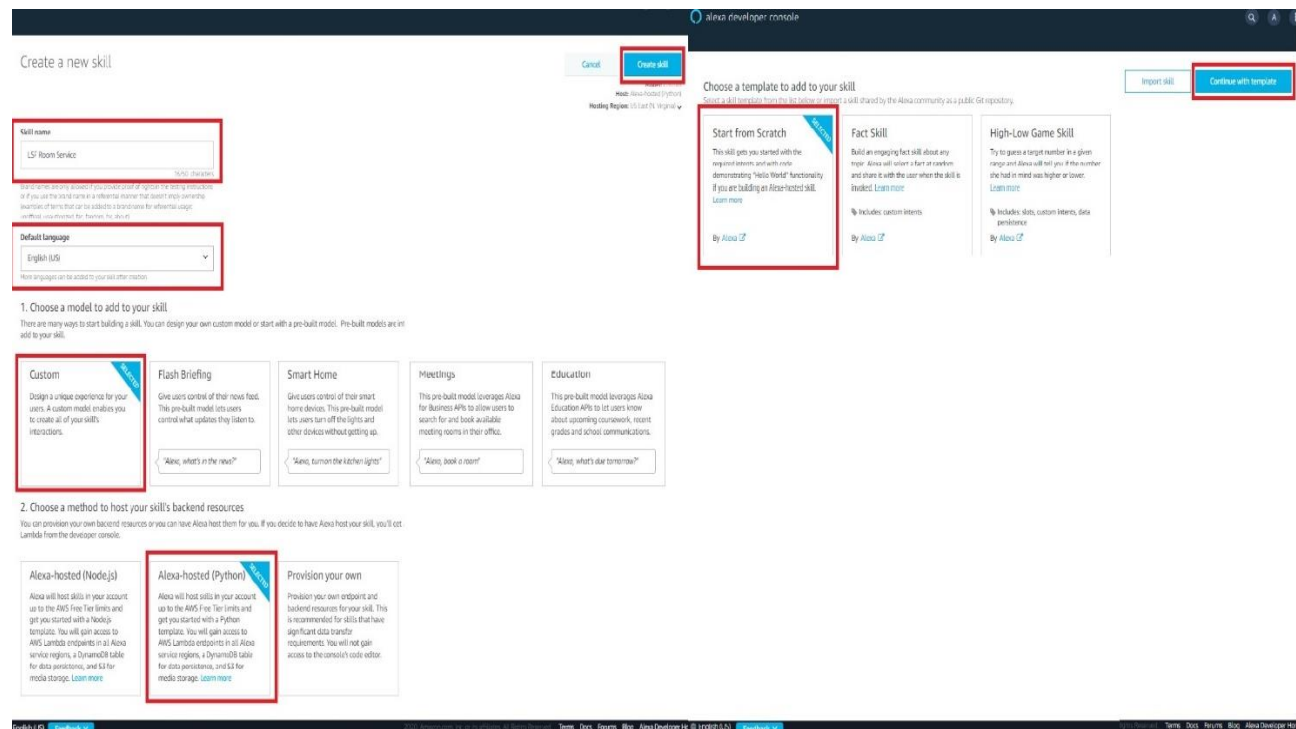
Fig 6. Screenshot of Steps to set up the skill

- Select 'Create Skills' option and follow the steps below to setup the skill (as in Fig 6):-
  - Enter the name as 'LSF Room Service'
  - Set the language as 'English (US)' (which would be selected by default)
  - Select the model as 'Custom' (which would be selected by default)
  - Select the host method as 'Alexa - hosted (Python)'
  - Select 'Create skill'
  - Select the template as 'Start from Scratch' (which would be selected by default)
  - Select 'Continue with template'
- This launches the Alexa Developer Console with the developer options for the 'LSF Room Service' skill displaying all the important tabs.
  - Build
    - The Build page is used to set up the skill, with configurations specific to the interaction model, and specific endpoints for the services.
  - Code
    - The ASK comes with sample custom skills developed with the help of Alexa Skills Kit SDKs.
    - These samples could be tested and deployed as AWS Lambda functions on AWS Lambda.
    - Concepts which might be useful when designing and implementing Alexa skills, such as database connection and other AWS services are also illustrated with these samples.
  - Test
    - Utterance profiles - Gives the option to enter utterances and resolve the intents and slots before the code is written for the services.
    - Alexa device – testing option on a Alexa enabled device.
    - ASK Command Line Interface (CLI) – ASK CLI can be used to test the skill from the command line. It can be used to invoke and simulate skills.
    - Skill Management API - Use the skill testing features of the Skill Management API.
    - Other options such as Skill simulator, Skill Management API, Alexa app etc.
  - Distribution
    - The Distribution page can be used to preview the skill appearance in the skill store and to determine the availability of the skill.
  - Certification
    - The Certification page can be used to verify the readiness of the skill for submission.
  - Analytics
    - The Analytics page can be used to generate reports of usage metrics of a skill.
- Click on the 'Code' tab and follow the following steps to setup the backend code (as in Fig 7)
  - Use the codes from the files in lambda folder
  - Using the options 'New File' and 'New Folder' create the file / folder structure by giving the same name as in the lambda folder
  - To make use of the code it must be deployed by selecting 'Deploy' option
- To setup the interaction model, click on the 'Build' tab (as in Fig 8)
  - Copy the code from the interactionModels/custom folder. This is the most important section and therefore requires a brief explanation.
    - With the interaction model, the logic for the skill and the definition of the the voice interface through which users interact with the skill.

-
  -
    - Intents - It represents the requests to be handled from the user. There are mainly two types Built in Intents, which comes defined with Amazon Alexa and Custom types intents which can be defined by the user.
      - Built in intents like eg: AMAZON.FallbackIntent, AMAZON.YesIntent etc.
      - Custom types like eg: FindRoomImmediately Intent, FindRoomWithDate Intent etc.
    - Intents optionally posses arguments called slots, which are variables that recognizes, handles and organizes data. They are mainly of two types, Built-in Slot types and Custom Slot types.
      - Built in slots like eg: AMAZON.Number ,AMAZON.Time etc.
      - Custom types like eg: YES_NO_SLOT, CUSTOM_NUMBER etc.
    - Sample utterances represents a set of spoken phrases required to invoke intents.
  - Click on the 'JSON Editor' option and select 'Drag and drop a .json file'
  - Paste the code into the editor field
  - Select the 'Build Model' option to train and build the interaction model
- To test the skill, click on the 'Test' tab and follow the following steps (as in Fig 9)
  - Set the 'Skill testing is enabled in' as 'Development'
  - In the 'Alexa Simulator' set up a interaction with sample use case from the documention files to get response
  - The request and response could be seen in the 'JSON Input 1' and 'JSON Output 1' section
- To deploy and preview the skill 'Distribution' and 'Certification' steps has to be followed
- Lastly, to analyze the performance and to preview the metrics check 'Analytics'
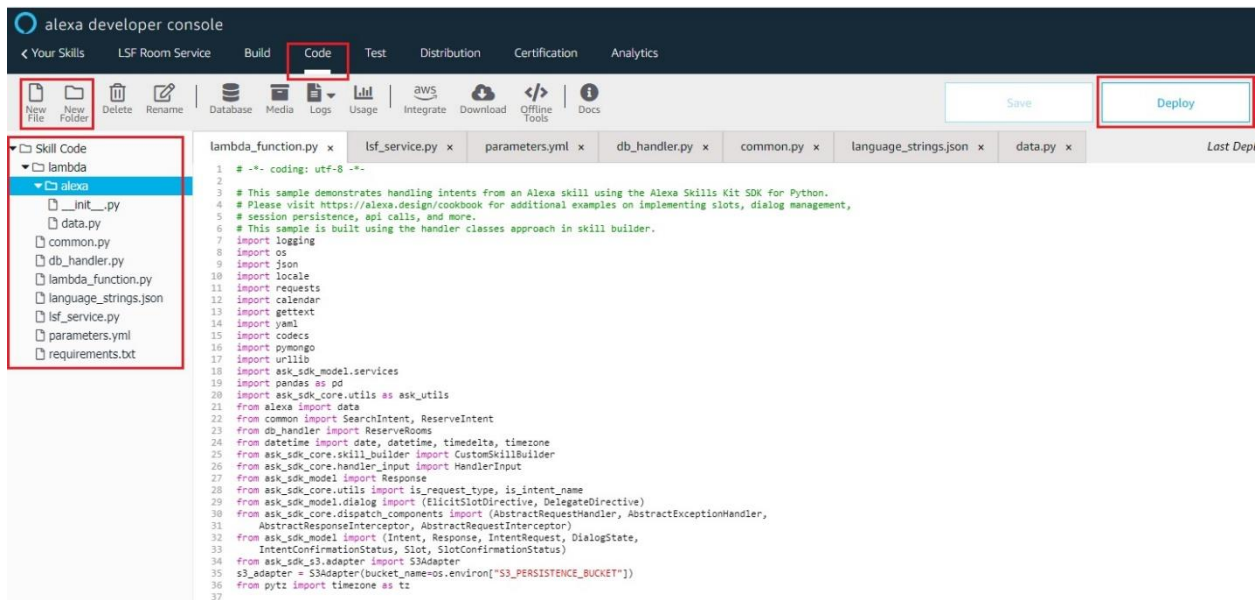

Fig 7. A sample look of code page

An important part of the skill development is the Dialog Delegation Strategy. It enables to identify the "prompts and utterances to collect, validate, and confirm the slot values and intents" [21]. There are mainly two ways to delegate data: -
- Auto Delegation
  - Alexa tries to complete the dialog steps based on defined dialog model. Alexa sends the skill a single IntentRequest when the dialog steps get completed.

13

- Manual Delegation
  - Alexa tries to send the skill an IntentRequest for each turn of the conversation manually with the Dialog.Delegate directive.
  - More details could be found through the [documentation](#).

Selection of the right delegation strategy for a specific interaction depends on how complex the dialog flow is. Auto delegation is simpler because as its not needed to handle any of the dialog through code. However, manual delegation with Dialog.Delegate is more flexible, as the skill can make run-time decision. It can also be used in combination with other Dialog directives to take complete control over the dialog flow if required. Since in the scope of the project was to keep the approach simple, the Auto delegation strategy served the best.
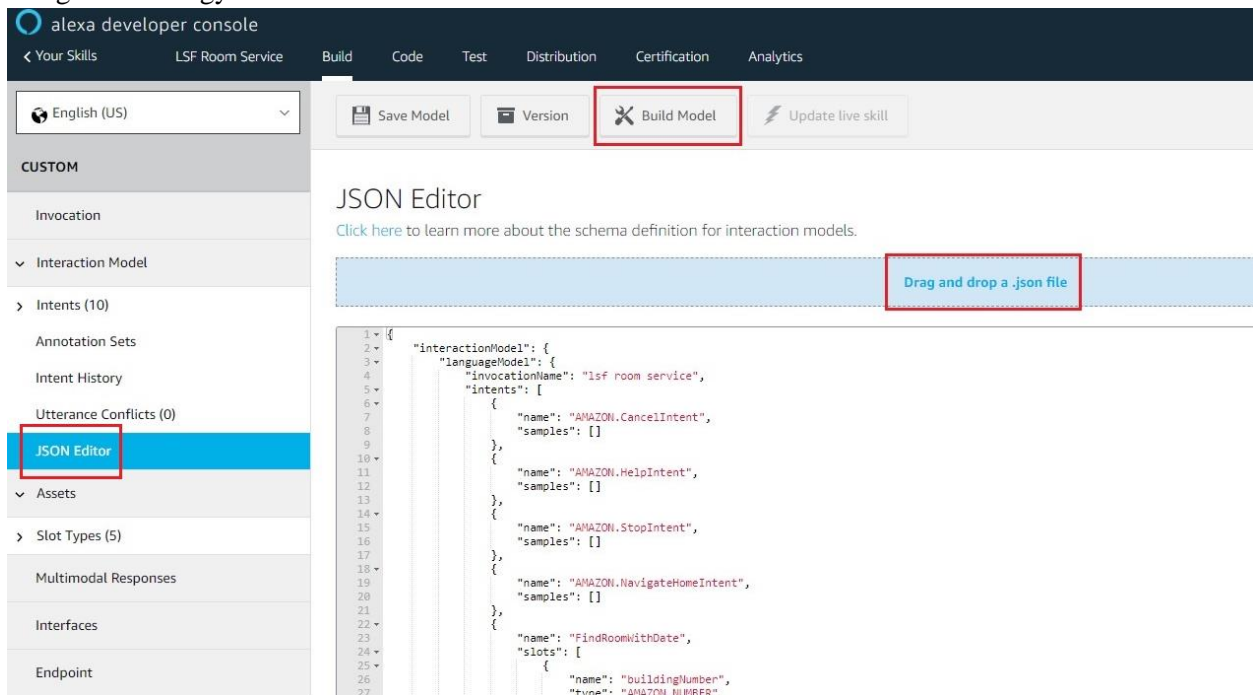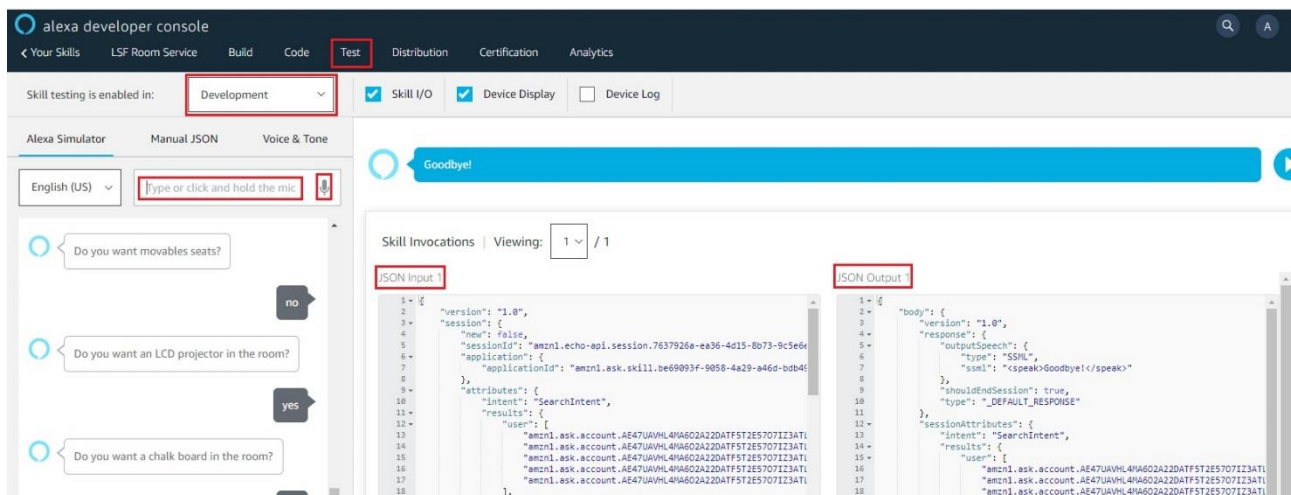

Fig 8. A sample look of build page

Fig 9. A sample look of test page

14

2.4 Alexa API Programming - The Skill Development

2.4.1 Building the handlers for skill intents

As mentioned in the previous sections, the skill was hosted as an Alexa hosted skill. Therefore, all the handlers were coded in the development console. This enabled to eliminate separated definition of backend service in AWS. The triggers that invokes a function is developed in the console itself. However, it woks the same as how an Alexa skill connects to backend, the AWS defined Lambda functions.

Before expanding on the handlers, the most important python file is the 'lsf_service'. This is the class deals with handling the LSF Room Reservation portal. It accepts data from the user when an intent is invoked through slot values. This is used to post data on to the portal and to scrape data from the results page. It has functions such as search and finding free rooms, extracting room details such as id, equipment details etc.

Below are the handlers that are included in lambda file: -
- LaunchRequestHandler - Handles the Launch intent which gets triggered when the skill is launched
- FindRoomWithDate Intent Handler
  - This intent takes necessary inputs such as building number, date, time etc. from the user which is retrieved through filling various slots.
  - After retrieving data from the filled mandatory slot values, it is used to search and find the room details available for reservation, from the LSF Portal.
  - The sample utterances are as below: -
    - Give me a free room {time}
    - find a free room {time}
    - find a free room in building {buildingNumber} for {duration}
    - find a free room for {date}
    - find a free room for {date} at {time} for {duration}
    - Give me a free room in building {buildingNumber}
    - find a free room in building {buildingNumber} for {date} at {time} for {duration} hours
    - Give me a free room in building {buildingNumber} for {date} at {time} for {duration} hours
- FindRoomImmediately Intent Handler
  - This intent takes neccessary inputs such as building number, duration etc. from the user which is retrieved through filling various slots.
  - After retrieving data from the filled mandatory slot values, it is used to search and find the room details available for reservation, from the LSF Portal.
  - The sample utterances are as below: -
    - Find a free room now in building {buildingNumber} for {duration}
    - Find a free room now in building {buildingNumber}
    - Give me a free room now in building {buildingNumber}
    - Give me a free room now in building {buildingNumber} for {duration}
- ReserveRoom Intent Handler
  - This intent is not meant to function on its own.
  - It is used in conjunction with FindRoomWithDate Intent or FindRoomImmediately Intent using a concept called Intent Chaining.
    - Intent chaining is a concept that enables a skill code to start dialog management from any intent, including the LaunchRequest.
    - Any of the custom intents can be chained in to as long as the interaction model posses a dialog model.

15

- ○ The slots 'roomOption', 'roomReserve' for this intent is elicited by prompting the user.
- ○ As this intent is used in conjunction with other intents (Intent Chaining) it does not have any sample utterances. Therefore, it prompts the user directly to elicit the slot values.
- RemoveReservation Intent Handler
  - ○ This intent enables the user to drop the reserved room.
  - ○ It is to be noted that a user is only able to make a single reservation. Therefore, this intent cancels the only existing reservation for a user.
  - ○ If the user response is a 'Yes', when asked for confirmation, then it would drop the single reservation that exists in the name of the user.
  - ○ The sample utterances are as below: -
    - ▪ I want to cancel my reservation
    - ▪ I would like to cancel my reservation
    - ▪ I want to drop my reservation
    - ▪ I would like to delete my reservation
    - ▪ I would like to remove my reservation
    - ▪ I would like to drop my reservation
    - ▪ I want to remove my reservation
    - ▪ I want to delete my reservation
- The detailed documentation, configuration and other usage details are explained clearly in the GitHub Repository.

2.4.2 Sample usage

The below sample gives an idea about the sample requests and responses.

Sample Request
Alexa, open lsf room service
    >> ...Hello. Welcome to LSF room service. What can I do for you?
Find a free room in building twenty nine.
    >> ...On which day do you want the room?...
Monday
    >> At what time do you want the room?
Ten o'clock
    >> ...
...
    >> ...
...
    >> Do you want a chalkboard in your room?
yes
...

Sample Response
We have found .... rooms for you....
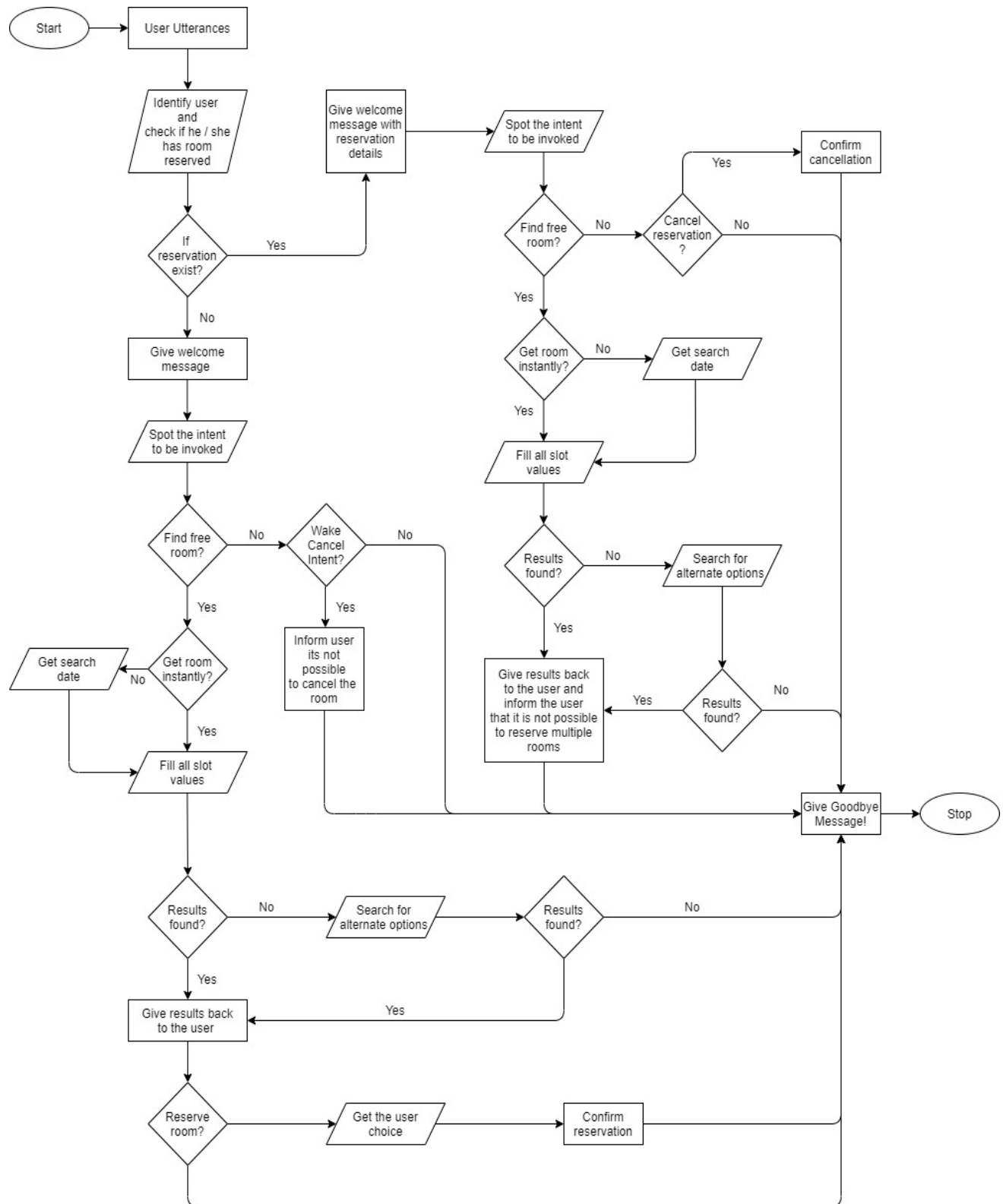Do you want to reserve any room....
...

Intent Chaining
Enter your choice ....

Sample Response
We have reserved ....

2.4.3 The Skill Dialog Flow

2.5 Functional Testing

The testing phase was carried out two stagesof the developed skill was carried out in two phases. The first phase was testing conducted through the Utterance profiler for each intent, and through the Alexa Developer Console. While the second one is the user testing. The in-detailed explanations are as below: -

2.5.1 The First Phase Testing

2.5.1.1 The Utterance Profiler

The utterance profiler enables to test the utterances alongside building the interaction model. The utterances can be entered to see how they resolve to the intents and slots. When an utterance does not invoke the right intent, sample utterances and retest could be updated, before writing any code for the skill. Moreover, the utterance profiler does not call an endpoint, and thus there is hardly any need to develop the service for a skill to test the model. The following steps can be used to test an utterance: -
- Click the Evaluate Model button in the upper-right corner of the developer console.
- Select the Utterance Profiler tab.
- Enter the utterance to test and click Submit.
- Review the utterance profiler results
- On submitting an utterance, the current intent and other considered intents would be displayed.
- On selection of an intent, the utterance profiler displays the relevant prompt from the dialog model and its possible to continue with the dialog.
- Intents without a dialog model, the "session" ends without any prompts displayed.
- More information on the same could be found here.

From Fig. 10,
- Selected intent displays the intent that would be sent to the skill for an utterance. If no intents were selected, this would've displayed "N/A".
- INTENT displays name of the selected intent specified in the interaction model. In this case it would be 'FindRoomWithDate'.
- SLOT(s) lists all of the slots for the intent and shows the various slot values identified from the utterance. From Fig. 10, it would be duration, date, movableSeats etc.
  - NEXT DIALOG ACT displays the dialog action for the next response that a user provides. There could be three possible actions ElicitSlot, ConfirmSlot, ConfirmIntent.
  - From Fig. 10 the ElicitSlot is selected as there are more slots to be filled

2.5.1.2 Testing in Alexa Developer Console

Another testing round was thoroughly performed through the testing feature of developer console itself. The major difference from that of the testing through the Utterance profiler is that through the console it is possible to perform end to end testing. This means as mentioned in previous sections it gives the same feeling of running the skill on an Alexa enabled device.
Therefore, its better to use the utterance profiler for testing working newly implemented features one at a time. Moreover, through the utterances profiler its only possible to test one intent at a time. There are additional features available mainly testing with both voice and text. Therefore, this gives in detailed functioning of the skill, that how effective it is. The user is able to see the JSON input and output (request and response). The Fig. 9^^ gives the exact idea of how the test page looks like while Fig. 5^^ gives an idea about the functioning of the database.

(^^- A click on Fig. 9 and Fig. 5, it traverses to the figure and a click on the image, brings back to this section)

Fig. 10 Utterance Profiler

2.5.2 User Testing

One of the most essential part of, not just the testing but also the project itself is User Testing. A period was set for User Testing based on which the User testing metric was drawn. The period of testing was between 23/11/2020 and 28/12/2020. The explanation can be done based on the metrics.
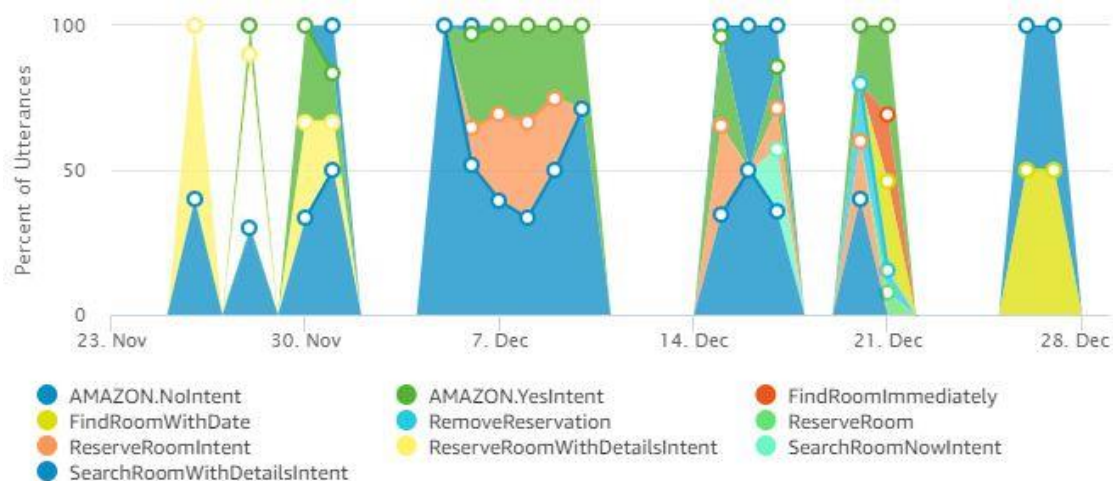

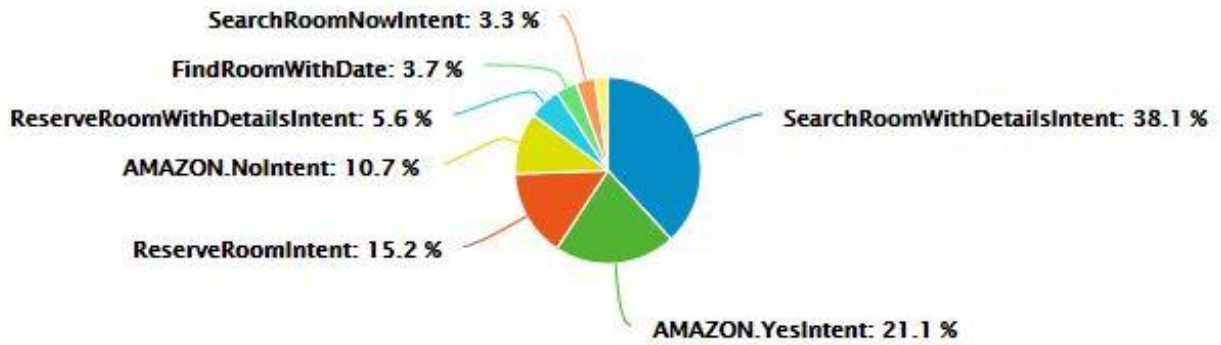Fig. 11a Total Intent invocation (based on day)

Fig. 11b Total Intent invocation (overall)

Fig. 11 shows the metric aggregation. This shows which all intents were used over the period of testing. Fig. 11a shows aggregation per day, whereas Fig. 11b aggregates the overall results and indicates which intent is being invoked most. From this metric its well evident that, most of the people used the skill search for rooms by providing details such as day, time etc. This results in the invocation of 'SearchRoomWithDetailsIntent'. It is same as 'FindRoomWithDetailsIntent' as the name was changed only recently. A considerable amount of user then decides to reserve room thus resulting in the invocation of 'Amazon.YesIntent' and then 'ReserveRoomIntent'. While the invocation of 'Amazon.NoIntent' indicates that users decided against room reservation.
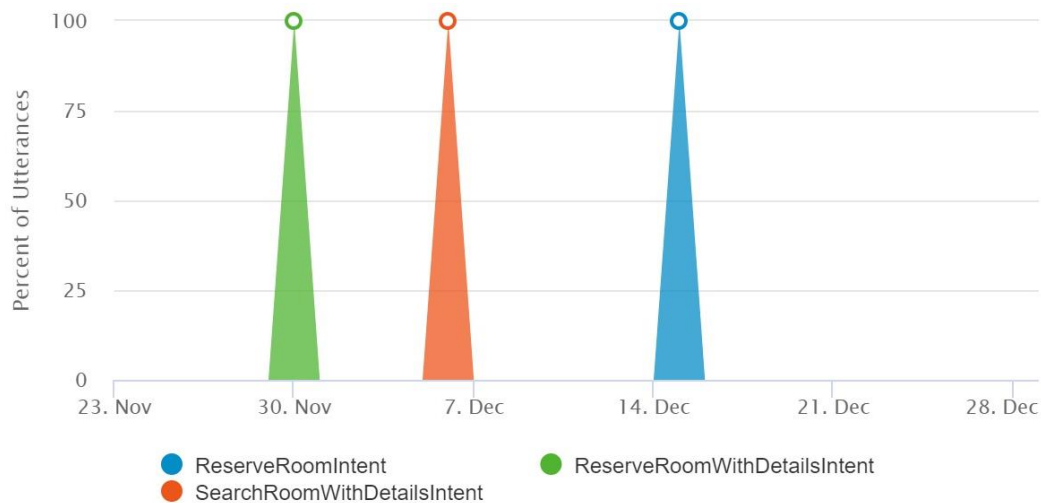

Fig. 11c Failed Intents

Fig. 11d Overall Skill Performance

20

Fig. 11c shows the instances of failed instances (day specific). One of the main reasons for the minimal errors is because of proper skill optimization. To all the users who were used to test the working of the skill, the functionalities, dialog flow, use cases etc. were explained in detail. And in short, the user had minimum room for making errors. While Fig. 11d shows the overall performance of the skill. This indicates that there is almost 95% success rate with the skill performance. The latency indicates that the latency is of the order of 90th latency percentile. 90% of the requests are faster than a given value of latency. Moreover, the very high confidence interval also indicates that there is minimal room for error making.

While, drawing all these metrics, its also important to have a look at the user session metrics.

Fig. 12a Total Sessions Invoked

Fig. 12b Session type Distribution

The session metrics explanation is as below: -
- Fig. 12a shows a distribution of all the sessions that was initiated by the users, daily basis. This is categorized as successful, failed, user did not respond and in-progress/expired. And these are self explanatory by category naming itself.
- Fig. 12b shows basically the percentage distribution of the metrics in that of Fig. 12a.
- Fig. 12c gives the daily basis distribution of sessions initiated by the users.

Fig. 12c Average sessions per customer distribution

- Fig. 12d gives the total session distribution. This gives a detailed distribution of sessions ending in successful, failed, unresponsive users, and session expired status. It can be inferred that most of the sessions end in successful manner, while most failed cases are due to the session getting expired. A good amount of session got terminated in failed status due to existing bugs, which was corrected stage by stage.

2.5.3 Suggestions for improving the skill from the users

The user testing also involved the stage of taking suggestions from the user to build an idea about the room for the improvement of overall skill performance and efficiency. This enabled the skill to perform differently from the typical web search.

The suggestions that were incorporated stage by stage are as below: -

- The skill initially had different Intent handlers to handle searching of free rooms and finding free rooms for reservation. A was suggested to club to a single intent and the reservation was handled through Intend changing. And finally, this worked in a more systematic manner.
- Another suggestion where to limit down the room types to Lecture Halls and to Seminar Rooms, in-order to limit down the total room results, which also helped in reducing response time.
- A suggestion was to limit the equipment facility related input from the user. This is because, more the inputs related to available equipment are taken, lesser would be the room availability.
- It was suggested to provide alternate room reservation options, so that the skill does an intelligent search methodology to provide the user with a satisfactory result at the end. There were three search options performed as below: -
  o Finding free rooms for same time and building with different equipment setup
  o Finding free rooms for same time in different building with same equipment setup
  o Finding free rooms for shorter duration in same building with same equipment setup
- Some of the intents where failing due to user getting confused on, what input is to be fed during a stage. For e.g., 'the search resulted in free rooms G29 109, G29 110, and user was asked to enter a choice to reserve a room. The skill fails if user enter anything other than numeral or ordinal format. This was handled by explicitly specifying the user to give the input in the format in which the skill accepts. For e.g. if the skill expects input in 'Yes' or 'No' format, the user would be prompted, '(Please say Yes or No)'.

3. Results

The results are listed after the conclusion of thorough functional testing. It was achieved with the best possible accuracy, user friendliness as optimization. The approach towards developing the skills were kept simple and well defined as below: -

- On implementation of every minor feature it was thoroughly tested first with the utterance profiler at first, which enabled to fix the bugs in the initial stages of development.
- After fixing them, using the Developer Console, it was monitored how the new feature implementation affects the overall application flow and development.
- After addressing the above the most important aim of the project was optimization
  - This included features like reducing complexity of the overall user interaction.
  - Reducing the amount of data that is to be handled.
  - Preventing multiple room reservations etc.
  - Cancelling the only existing reservation just by a single voice command.
  - Incorporation of a skill

4. Further Discussions and Limitations

The thorough testing of the skill helped in understanding the shortcomings in the skills. Some of these issues are addressable on releasing newer version which happens during the maintenance phase of any application. The current version of the code was developed in the most effective manner with the existing resources and forging through several limitations put forward by the developer console. However, some of these issues are to be addressed and can't be solved from developer end via the developer console: -

- There exists an issue with the AMAZON.NUMBER built in slots (issue with console)
  - For example, it does not recognize single digits such as one, two etc.
  - The skill demanded the requirement for a slot that accepts user choice in the format of single digit number, but this issue prevented from achieving the same.
  - To address this a custom slot called CUSTOM_NUMBER which accepts numbers in both digits as well as in ordinal format. This gave the alternative that instead of one, the choice could be specified as first.
  - However, for a user to whom this issue is unknown will be unable to fully traverse through an entire flow of room reservation.
- There are several shortcomings with the developer console itself that for instance, there are no slots that accepts strings, limited storage capability etc.
  - The main reason for this is that the developer console itself is still under development and thus posses daily nightly releases with minor enhancements.
  - For example, the feature that a slot that can handle multiple types of values in the console is still a beta functionality.
  - Its possible to address some of these issues by handling the same using AWS Lambda functions. But development of application within the console suited more for this project.
- The second issue being that the MongoDB which is currently in use has the storage limitation of 512 MB. Therefore, for more storage requirement either it should be moved to DynamoDB which requires AWS paid account for best scenario or may be paid tier of MongoDB itself.
- By default, there are limited permissions available to access API for accessing personal information through skill. This is Country / Region specific else would need AWS paid account.

References

[1] Wikipedia contributors, "Amazon Alexa," Wikipedia, The Free Encyclopedia,
https://en.wikipedia.org/w/index.php?title=Amazon_Alexa&oldid=996452457,
(accessed December 29, 2020)

[2] Wikipedia contributors, "Virtual assistant," Wikipedia, The Free Encyclopedia,
https://en.wikipedia.org/w/index.php?title=Virtual_assistant&oldid=991058879,
(accessed December 29, 2020)

[3] LSF Wiki, "LSF documentation",
https://wikis.ovgu.de/lsf/doku.php?id=start&rev=1592817099,
(accessed December 29, 2020)

[4] CHINA GALAXY INTERNATIONAL, "COMPANY / INDUSTRY NEWS",
http://www.chinastock.com.hk/ewebeditor/uploadfile/20170714165226781.pdf,
(accessed January 9, 2021)

[5] Flow-chart-echo, "Echo Voice Request Lifecycle",
https://d1842wf5bcpgkm.cloudfront.net/magazine/wp-content/uploads/2017/09/Flow-chart-echo-
1024x393.png, (accessed January 9, 2021)

[6] Custom Voice Model Skills, "Host a Custom Skill as an AWS Lambda Function",
https://developer.amazon.com/en-US/docs/alexa/custom-skills/host-a-custom-skill-as-an-aws-lambda-
function.html, (accessed January 9, 2021)

[7] Tools to Create and Manage Skills, "Build a Skill End-to-end Using an Alexa-hosted Skill",
https://developer.amazon.com/en-US/docs/alexa/hosted-skills/build-a-skill-end-to-end-using-an-alexa-
hosted-skill.html, (accessed January 9, 2021)

[8] lsf.ovgu.de, "Detaillierte Raumsuche",
https://lsf.ovgu.de/qislsf/rds?state=extendedRoomSearch&type=1&next=extendedRoomSearch.vm&next
dir=ressourcenManager&searchCategory=detailedRoomSearch&noDBAction=y&init=y&asi=,
(accessed January 9, 2021)

[9] w3schools.com, "Python MongoDB",
https://www.w3schools.com/python/python_mongodb_getstarted.asp, (accessed January 9, 2021)

[10] w3schools.com, "Python MySQL",
https://www.w3schools.com/python/python_mysql_getstarted.asp, (accessed January 9, 2021)

[11] Wikipedia contributors, "MongoDB," Wikipedia, The Free Encyclopedia,
https://en.wikipedia.org/wiki/MongoDB, (accessed December 29, 2020)

[12] mongodb, "The database for modern applications",
https://www.mongodb.com/, (accessed January 9, 2021)

[13] amazon alexa, "Alexa Skills Kit",
https://developer.amazon.com/de-DE/alexa/alexa-skills-kit, (accessed January 11, 2021)

[14] amazon alexa, "Create the Interaction Model for Your Skill",
https://developer.amazon.com/en-US/docs/alexa/custom-skills/create-the-interaction-model-for-your-skill.html, (accessed January 11, 2021)

[15] amazon alexa, "Build Your Skill",
https://developer.amazon.com/en-US/docs/alexa/devconsole/build-your-skill.html,
(accessed January 11, 2021)

[16] amazon alexa, "Get Custom Skill Sample Code",
https://developer.amazon.com/en-US/docs/alexa/custom-skills/use-the-alexa-skills-kit-samples.html,
(accessed January 11, 2021)

[17] amazon alexa, "Test Your Skill",
https://developer.amazon.com/en-US/docs/alexa/devconsole/test-your-skill.html,
(accessed January 11, 2021)

[18] amazon alexa, "Define Skill Store Details and Availability",
https://developer.amazon.com/en-US/docs/alexa/devconsole/launch-your-skill.html,
(accessed January 11, 2021)

[19] amazon alexa, "Test and Submit Your Skill for Certification",
https://developer.amazon.com/en-US/docs/alexa/devconsole/test-and-submit-your-skill.html,
(accessed January 11, 2021)

[20] amazon alexa, "View Skill Metrics",
https://developer.amazon.com/en-US/docs/alexa/devconsole/measure-skill-usage.html,
(accessed January 11, 2021)

[21] amazon alexa, "Delegate the Dialog to Alexa",
https://developer.amazon.com/en-US/docs/alexa/custom-skills/delegate-dialog-to-alexa.html,
(accessed January 11, 2021)

[22] amazon alexa, "Use Intent Chaining to Enter Dialog Management from a Different Intent",
https://developer.amazon.com/en-US/blogs/alexa/alexa-skills-kit/2019/03/intent-chaining-for-alexa-skill,
(accessed January 11, 2021)

[23] amazon alexa, "Test Your Utterances as You Build Your Model",
https://developer.amazon.com/en-US/docs/alexa/custom-skills/test-utterances-and-improve-your-interaction-model.html, (accessed January 11, 2021)

[24] the business communication, "Advantages and Disadvantages of Speech",
https://thebusinesscommunication.com/advantages-and-disadvantages-of-speech/#:~:text=So%2C%20the%20main%20advantages%20of,the%20speaker%20and%20the%20audience., (accessed January 11, 2021)

Appendix

[1] GitHub Repository, "athulrajvsovgu / LSFRoomSkillAlexa",
https://github.com/athulrajvsovgu/LSFRoomSkillAlexa

[2] LSF Room Reservation Portal, "Detaillierte Raumsuche",
https://lsf.ovgu.de/qislsf/rds?state=extendedRoomSearch&type=1&next=extendedRoomSearch.vm&nextdir=ressourcenManager&searchCategory=detailedRoomSearch&asi=

[3] amazondeveloper, "Voice Design Guide",
https://developer.amazon.com/designing-for-voice/

[4] CodeAcademy : Learn Alexa, "Learn to Program Alexa",
https://www.codecademy.com/learn/learn-alexa

[5] Official Alexa Skills Kit Python SDK, "ask-sdk 1.15.0",
https://pypi.org/project/ask-sdk/

[6] Official Alexa Skills Kit Python SDK Docs, "Alexa Skills Kit SDK for Python",
https://developer.amazon.com/en-US/docs/alexa/alexa-skills-kit-sdk-for-python/overview.html

[7] Official Alexa Skills Kit Documentation, "Build Skills with the Alexa Skills Kit",
https://developer.amazon.com/en-US/docs/alexa/ask-overviews/build-skills-with-the-alexa-skills-kit.html

[8] amazondeveloper, "Amazon Developer Forums",
https://forums.developer.amazon.com/spaces/165/index.html

[9] Hackster.io, "Welcome to Hackster!",
https://www.hackster.io/amazon-alexa