University of Sheffield International Faculty,
CITY College
Department of Computer Science

FINAL YEAR PROJECT

Teaching Alexa to be Alexa

This report is submitted in partial fulfillment of the requirement for the degree of Bachelor of Science with Honours in Computer Science by

Donat Salihu

June, 2021

Approved

Dr. K Dimopoulos

Teaching Alexa to be Alexa

by
Donat Salihu

Dr. K Dimopoulos

Abstract

Speech is the most natural way humans communicate with each other. Vocal communication gave humans the ability to form and share thoughts and ideas. In modern times, this way of communication has led to the rise of many speech processing assistants, which allow humans to communicate with devices more intuitively compared to classic typing or graphical user interface.

This project aimed to build an Alexa Skill that allows users to build other skill only using their voice. The user is going to be able to communicate with this Alexa Skill or template and develop another skill based on their needs via voice interface only.

This report gives a detailed explanation of the challenges and concerns while implementing the project. It includes a literature survey, implementation choices, design and explanation of the technology used.

DECLARATION

All sentences or passages quoted in this thesis from other people's work have been specifically acknowledged by clear cross referencing to author, work and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this thesis and the degree examination as a whole.

Name : Donat Salihu

Singed: Date: 05.06.2021

Acknowledgements

I would like to thank my family

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides background information on human-computer interactions and voice interface. It describes different voice interface technologies while focusing on Amazon's virtual assistant, Alexa. In addition, it describes what an Alexa skill is as well as it sets out the project aims.

1.1 Background

The interaction between humans and computers remains one of the main fields of studies among many computer scientists, engineers and psychologists. The biggest tech companies in the world like Apple, Amazon, Google, Microsoft etc. invest a lot of money, time and human resources into developing user-friendly interfaces. Different devices and operating systems are developed and published every year competing to gain the most of the market sales. The competition and the goal are mainly based on the functionality of the software, robustness and user-friendliness. Computers, phones and other tech devices are designed to fit human needs. Some of these needs include simple tasks as setting up an alarm while some more complex tasks include modelling and simulating an aircraft or a propulsion system. However, to complete these tasks, humans must interact and communicate in a way or other with computers. From the first introduction of computers in the 60s until now, the most popular way of interaction between humans and computers has been the graphical user interface[1]. Graphical User Interface or GUI is a form of user interface that allows the user to interact with computers or other electronic devices through graphical icons. The GUI was developed in the 70s as a response to the problem of inefficiency of text-based command-line interfaces. Since then,

it has been the main user interface for computers. Many modern graphic user interfaces have developed and implemented features such as touch screen and even voice command interaction capabilities. Due to the importance of GUI for applications, according to the results of a survey, in today's applications, 48% of code is used is dedicated to GUI implementation. While the average time portion spent on the user interface is 45% during the designing phase, 50% during the implementation phase, and 37% during the maintenance phase[2].

1.2 Virtual Assistants

A virtual assistant is a software system that can provide services to a human agent. The communication between the user and the virtual assistant is usually done through texting or speech. Virtual Assistants use natural language processing to be able to understand the user input and to respond[3]. Services offered by different virtual assistants include: answering questions, playing music, setting up reminder etc. Virtual assistants are being integrated in different technologies and tools. Phones, computers, cars and a lot of other devices are using virtual assistants to offer better services to the users. There are a lot of different virtual assistant available. Some of them are Alexa Amazon, Google Assistant, Siri by Apple, Cortana by Microsoft etc.

1.3 Project Aim

Alexa skills are voice applications. Those skills are developed to help the user do more with Alexa There are different Alexa skills. Some type of Alexa skills include playing music, setting up a reminder, skills for news, weather forecast or games. However, the user can only use a skill as long as the skill is developed and available to enable on their system. If the user has specific needs for a skill and there are no such skill available to enable, the only option is to develop one. This project aims to design and develop an Alexa Skill or tool that enables the users to program other skills only using their voice. The standard way of developing an Alexa Skill is through Alexa Developer Console, coding the responses and hosting the code on a Lambda function.

1.4 Overview of the report

This report contains eight chaptes, the first chapter contains the introduction, the second chapter is the literature review chapter that discusses on researches about human computer interactions, virtual assistants, dialogue flow of virtual assistants etc. The third chapter is the project management chapter which discusses the technologies and tools used in the project, the software development processes. The fourth chapter is the Requirements and Analysis which discusses the requirements of the project and analysis performed before the designing pahse. The fifth chapter, the Design chapter discusses the different approaches to the aim of the project during development and the next chapter, the implementation chapter, discusses the implementation and the final version of the system as well as the testing performed after the development of the project was finished. The final chapter contains a short summery and the conclusion.

Chapter 2

Literature Review

This chapter will discussed the literature review involved with Virtual assistants and Human Computer Interaction as two main fields of this project.

2.1 Human Computer Interaction

Human-Computer Interaction or HCI, sometimes called Man-Machine Interaction is a field of study that was represented with the public emerging of computers. The reason why HCI emerged is the rise of sophisticated machines and systems. With the advancements in computers and machines, people needed more training and education to be able to operate those machines. The main goal of developing a system is that the system is going to be as functional as possible so that the users can accomplish certain goals most productively. To accomplish those goals, other than developing a good system, the users should be well informed about the system and knowledgeable on how to operate with the system as well. HCI is the field of study that deals with human interaction with computers. It considers many aspects of human behaviours and the complexity of the system when designing an interaction method[1, 4].

2.1.1 Unimodal HCI Systems

An interface mainly relies on the number of inputs and the diversity of those inputs. Inputs and outputs in a system are the communication channels between the user and the system. Through these channels, the interface makes possible the human-computer interaction. Each of these different channels is called a modality. The interfaces that use only one channel or modality are called unimodal systems[4].

There are three different types of unimodal systems.

- Visual-Based
- Audio-Based
- Sensor-Based

2.1.2 Visual-Based HCI

Visual Based HCI is the most widespread area of research in the HCI field. The designing of Visual-Based HCI focuses mainly on the mechanism through which users can complete and perform tasks based on visual elements. Main research areas of Visual-Based HCI include: Facial Expression analysis deals with the study of facial changes in response to the individual's emotional state. Facial analysis systems attempt to analyze facial expression changes from visual information[4]. Other areas of study are Body Movement Tracking, Gesture Recognition, Eyes Movement Tracking. Gaze detection is used to detect the user's attention in specific situations. While Eye Movements tracking is an interface for helping disabilities where eye movements play an important role on command(blinking for clicking)[4].

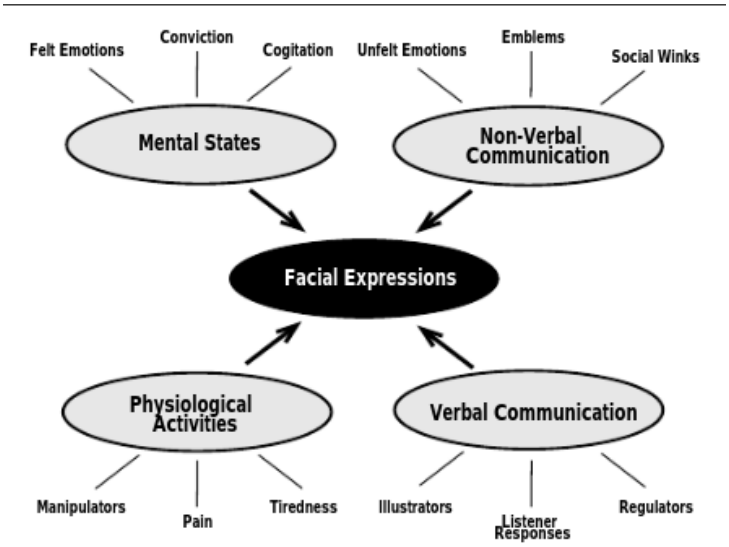

Figure 2.1: Sources of facial expressions

2.1.3 Audio-Based HCI

Another important area of HCI is the audio interaction between the human and a computer. This area deals with different ways audio is acquired and processed by the computer. Usually, these auditory systems are bidirectional connections between a human user and a computer or any other technical product. This interface includes research areas involving in machine listening, speech recognition and dialogue system, auditory emotion analysis, human-made noises etc[4, 5].

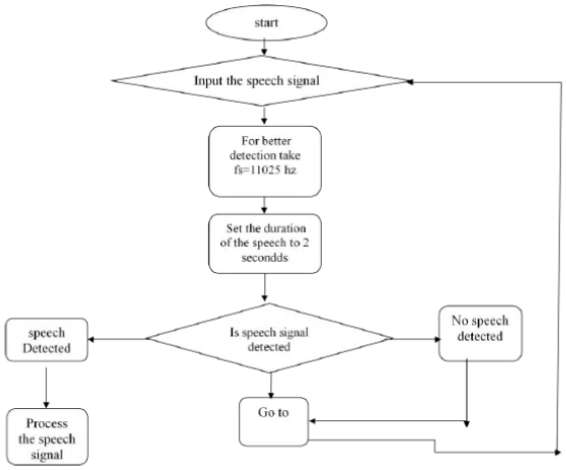

Figure 2.2: Basic Voice-Interface System

The above image, a single flowchart explaining the process of a basic voice-user interface with speech detection.

2.1.4 Dialogue system

A dialogue system, also called a conversational agent, is a programmed system intended to conversate with a human in a natural language. The main purpose of a dialogue or a spoken system is to create an interface between the user and the computer application. Usually, those computer applications are database or expert systems. Dialogue systems can be classified according to the methods used to control the dialogue in three main types[7, 8]:

- Finite-state
- Frame-based state
- Agent-based

2.1.4.1 Finite State-Based Systems

In a finite state dialogue, the system leads the conversation. There are predefined ways and states where the systems lead the conversation. The system recognizes or rejects specific phrases when the user responds. It only produces actions based on the recognized response. A finite-state automaton example is a system for bank balance or for withdrawing money.

System: What action would you like to perform, reply with account balance or withdraw?
User: Account Balance
System: You have (the_balance) euro in your account?
User: Thank you!
System: Bye!

System: What action would you like to perform, reply with account balance or withdraw?
User: Withdraw
System: How much would you like to withdraw, please only specify the amount?
User: 200 euro!
System: Proceeding...!
.

In a finite state system, the user can only reply with a word or a specific short sentence predefined by the system.

2.1.4.2 Agent-Based Systems

Agent-Based or AI systems are designed to have complex conversations with the user and the underlying application in order to solve the problem or to complete the user's request[6].
User: Is there any available flight ticket to Basel on Monday?
System: No, there are not any available flights to Basel on Monday or the next week at all. But there are available tickets on Monday to Zurich, 75.2 km away from Basel. You can book a flight to Zurich on Monday and take the train to Basel. You will be in Basel from Zurich

in less then 1 hour.
User: Okay, thank you!

In the example above, the system not only informs the user that there's not any ticket to Basel, but finds the closest Airport to Basel, checks if there's a free ticket on Monday, and then gives the user information about travelling means from Zurich to the desired location. So, rather than the system replying simply 'NO', it tries to give information that might be helpful to the user. The conversations between the user and the system in an Agent-Based could be considered conversations between two agents. Both agents are capable of reasoning. The conversation usually evolves dynamically as a sequence of related steps.

2.1.5 Sensor-Based HCI

In sensor based HCI, physical sensors are used between the machine and the user in order to provide the interaction[4]. Some of these sensors include:

- Pen Interaction
- Mouse and keyboard
- Joysticks
- Motion Tracking Sensors
- Pressure Sensors
- Ultrasonic movement sensors

2.1.6 Multimodal HCI Systems

A Multimodal HCI System is a combination of different modalities or the usage of more than one communication channel signal between the user and the system. A multimodal interface acts as connector channel between human computer interaction of more than one modes of input or output. Inputs on multimodal interactive models include multiple input modes such as speech, handwriting, keyboard, audio and other input modals. While the output of an interactive multimodal implementation includes output models such as graphics, audio files, speech, text, animations etc. The logical component that coordinates data and manages resources, inputs and outputs, as well as execution flow, is the interaction manager[4].

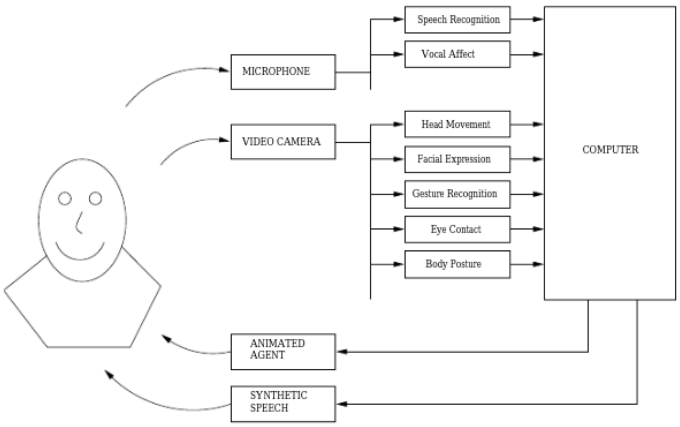

Figure 2.3: Multimodal HCI System

2.2 User Interface

The user interface(UI) is the meeting point where the users communicate with systems or computers. UI is designed in interaction layers that engage with human senses. User interfaces include input and output devices. Some of these devices are mouse, keyboard, monitors, printers, microphones, audio systems etc. There are different types of User Interfaces developed to allow users to interact with computers in different ways. Some of them are:

- **Graphical User Interface** – GUI is an interactive system or interface that allows users to interact with the computer via visuals.
- **Voice User Interface** – VUI allows users to interact with the computer or systems via their voice only.
- **Command Line Interface** – CLI is a text based interface. Allows the users to interact with computer via text. It is used mainly to view images or files.
- **Touch User Interface** – Allows users not only to interact with the computer via visuals but also the user to interact directly to the screen based on tactile interaction input.

The most popular user interface is the Graphical User Interface[3, 7].

2.3 Virtual Assistans

Speech is the primal way of communication between people. The desire of humans to design a machine that has the capability of speaking naturally has led to much research and studies by different scientist. In 1930, Homer Dudley and Bell Laboratories have made significant steps into designing a small machine that could only recognize a set of sounds[12]. Another great step forward in the field was the invention of "ShoeBox" in 1962 from IBM. The "ShoeBox" machine had the capability to recognize 16 words. In 1996, IBM released another system called "VoiceType Simply Speaking" which had 42,000 words of vocabulary. The system supported English and Spanish languages[11]. From a single machine that could only recognize a set of sounds, the field has developed so much that now we have fluently speaking voice interface systems that humans interact with to complete their daily tasks.

2.4 Speech Recognition

Speech recognition allows the computer to understand human natural language. It translates the human's spoken words into a text, thus allowing the system to process and understand the word and finally give a response back to the user. Speech recognition is the ability of a system to respond to a user's commands. Before any machine or system can process the speech, a microphone is used to translate the user's voice and transfer it to wave electrical signal. These systems can analyze the users' speech or commands by reading the electrical waves, analyzing them and responding to the user with electrical signals, that is translate into sounds or words that humans can understand[12].

2.5 Natural Language Dialogue

The main objective of a natural language dialogue system is to identify and process the information given by the user so the dialogue manager can make use of the information. Once the information has been processed by the natural language dialogue system(interface agent), it has to be processed and if necessary, stored. The interaction agent observes three different. First, it has to extract the data or the static information. Secondly, it should get the underlying intentions and slots and lastly, it has to design the structure of interaction. Finally, the system has

the answer needed to output according to the information the user-provided. The template used to generate the natural language response to the user might require arguments from the user to fill in slots[12].

2.6 Speech Generator

The job of a dialogue system is to automatically generate a response to the user[15]. Input detector is the component responsible for detecting and recognizing user input. It recognizes the voice invocation for the software as well as the utterances that the system might need to invoke any intent and respond to the user. The speech generator translates the message constructed by the system(response generator) into a spoken form that the user can understand. There are two approaches to translating the message generated from the response generator to the spoken message. The first one is to use prerecorded voice samples. The second solution is to use text to speech technologies. Some of the text to speech technologies are Contaminative Speech Synthesis, Text to Speech(TTS) and Text to Phoneme conversion[14].

2.7 Conversational AI

Conversational AI is the set of technologies that combine natural language processing with machine learning. Conversational AI can recognize speech and text, decipher different languages and communicate back to the user like a real human. Conversational AI gives the feeling to the user that the user is communicating with another human and not a machine. The reason why Conversational AI can respond in a natural way to the user is that it has different principle components that allow it to generate these responses. The main components integrated into a conversational AI system are Machine Learning(ML) and Natural Language Processing(NLP)[13, 18].

2.7.1 Machine Learning

Machine learning is a field of Artificial Intelligence that provides the system with the ability to self-learn. Its is made up of a set of algorithms, features and data sets which help the system to continually improve itself. As time goes, the AI machine gets better at recognizing patterns[19].

2.7.2 Natural Language Processing(NLP)

Is the field of study that deals with the voice and speech interaction between the human and the computer. It is the part of the system which translates the human speech or voice into electrical signal so that the system can understand, process and respond back to the user[12]. In Conversational AI, the NLP consist of four steps. These four steps are Input Generation, Input Analysis, Dialogue Management and Reinforcement Learning[12, 16].

2.7.3 Reinforcement Learning

Reinforcement Learning is a field of Machine Learning. Various software are included into the system that help it find the best behavior to reach the goal and the reward[19].

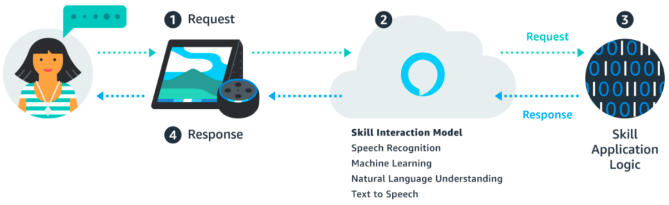

Figure 2.4: AI Conversation Skill by Amazon Alexa

2.8 Virtual Assistants Systems

Virtual assistants are software agents that can perform different tasks. These assistants respond to the user's requests or questions. Users can ask verbal questions and the system is going to process the input and via the software and algorithms integrated, the system will be able to verbally respond to the user. Some of the most popular virtual assistants are Siri by Apple, Google Assistant by Google, Alexa by Amazon, Cortana by Microsoft, Fyle, Extreme, Bixby etc[20].

2.8.1 Siri

Siri, developed by Apple is a voice-controlled virtual assistant. It is only available for Apple users. It was first released in October 2011. Siri can assist the user by responding to the queries, setting alarms, calling contacts, opening applications, read messages, write messages, make notes etc. It is available in 21 languages.

This virtual assistant works on two principles, speech recognition and Natural Language Processing[20].

2.8.2 Google Assistant

Google Assistant is a virtual assistant developed and maintained by Google. It is a two-way conversation assistant. It was first released in May 2016. Google Assistant, like many other popular virtual assistants, is build based on natural language processing. The way Google handles the user's queries is by recording the speech and due to the large computational power to analyze the input, it sends it to the Google servers. At the servers, the input is broken down into words and it tries to find similar words on the database to better understand what the request was. It identifies keywords and calls on intents or functions corresponding with those words. For example, if it notices words like weather or temperature, it opens the weather app, gets the information and sends it back to the user[20, 21].

2.8.3 Amazon Alexa

Alexa is a cloud-based virtual assistant system that interacts with human agents in a natural language. Alexa is available in millions of devices[16](Amazon devices and third-party devices). It allows the users to interact in a natural language to complete daily tasks like setting up a reminder, check the calendar, build facts skills, educational purposes, customer service, playing games, learning a foreign language and many more. There are also Conversational AI Alexa skills/applications which are in Beta mode[22]. However, Alexa has its kit which allows for an intelligent development of utterances, dialogue manager, dialogue flow and integration with the database information.

2.8.4 Software Development Kit

Alexa Amazon and Google Assistant are two virtual assistants which allow everyone to make their skills or voice applications. Both have their SDK which allows every developer to make a skill or voice application and publish it. There are already a lot of Alexa skill developed and available to use. The fact that these two virtual assistants have their SDK where everyone can develop a skill, led to a large number of skills/voice applications to be developed. There are currently more than 100,000 Alexa skills out there[23, 24] (more on Alexa SDK in the next

chapter).

2.8.5 Smart Alexa

Alexa gets smarter every day. User's request to Alexa is used by Alexa not only to invoke a skill or to prompt for an answer from the database but to train speech recognition and natural language understanding as well. Speech recognition and natural language processing are trained by the integrated machine learning algorithms in Alexa[25].

2.8.6 Alexa Skills

Alexa skills are voice applications. The user verbal interacts with Alexa through the skill to complete different daily tasks. The user can enable or disable a skill using the Alexa app the same way other apps are installed or uninstalled from a phone or a tablet. There is a lot of available skill in different categories. The main categories of Alexa skills are business and finance, productivity, news, weather etc. All of the Alexa skill is free to use, however, some skill might charge to fully use them[25].

2.8.7 Developing an Alexa Skill - ASK

To develop an Alexa skill, the developer should use a framework. The most popular framework for building an Alexa skill is the Alexa Skill Kit(ASK) framework developed by Amazon itself. Alexa supports two types of voice interaction models[26]. It supports the **Pre-built voice interaction model** and the **Custom voice Interaction model.**

2.8.8 Steps to build an Alexa skill

To build an Alexa skill, the developer has to use Alexa Developer Console and Lambda function hosted on AWS. The lambda function hosted on the AWS is going to be the endpoint of the skill. The code and the logic of the skill are on the lambda function, while in the developer console, the intents and utterances are stated and developed. The developer has to log in to the Amazon Developer Console, create a skill and then add Intents and Utterances as needed for that particular skill. To add the intent, the developer should go to the intents section of the skill[27].

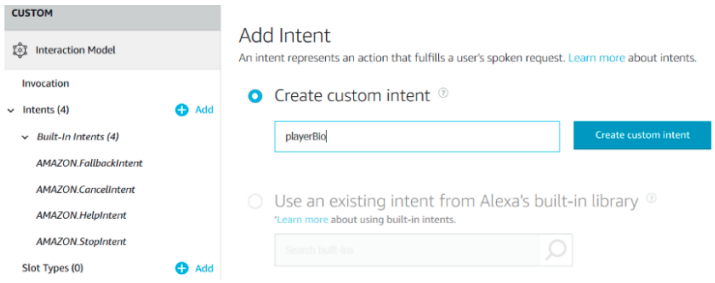

Figure 2.5: Adding Intent to an Alexa Skill using Alexa Developer Console

An intent in an Alexa skill, represents an action that has to fullfill the user's spoken request. The intent is invoked by the user by using utterances.
Utterances are a set of spoken phrases which are mapped with a certain intent. No two intents can have the same or similar utterances because Alexa might not be able to differentiate the intents. The developer has to add the utterances that map to a specific intent[27]. For example the (PlanMyTrip) intent might have utterances like:

```
i am going on a trip on friday
i want to visit portland
i want to travel from seattle to portland next friday
i'm driving from seattle to portland
i'm driving to portland to go hiking
```

Figure 2.6: Utterances and slots sample

2.8.9 Slots

A slot is a variable that is mapped to intent and allows Alexa to better understand information about the spoken request by the user. The user request might take the form I want to go to Basel. Basel in this case is the slot. The utterance with the slot would be inserted in the skill as follows: I want to go to city, where the city is a slot/variable which will store the user input, in our case the Basel. There are different slot types. The slot types indicate how the data is stored and handled[27].

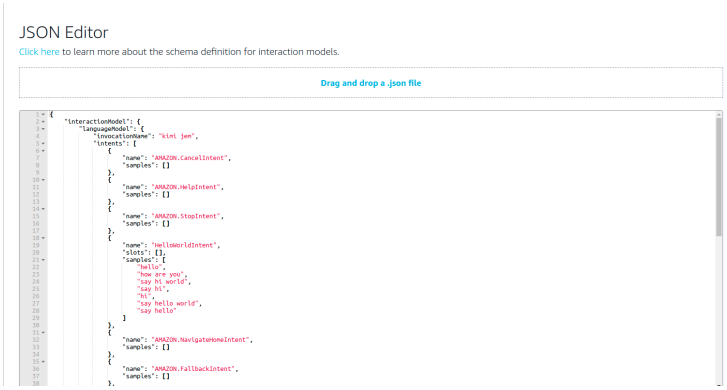

Figure 2.7: Generate JSON file in Alexa Developer Console from the utterances and slots declared

After the intents and utterances get declared, a JSON file is automatically generated from the Alexa developer console. The JSON file stores all the utterances that the users can use to call the Intent. Below is a representation of a JSON file generated by the Alexa Developer Console based on the intents and utterances designed by the developer of the skill.

2.8.10 Creating the AWS Lambda Function

To have a function Alexa skill, the developer should link the interface created on the Alexa Console Developer with a lambda function on AWS. This Lambda function receives events from the interface and returns responses to it. The predefined utterances are used to invoke an intent that is going to return a response by the Lambda function based on the input[29]. The user utterances invoke different REQUESTS. Three main types of requests are: Launch Request, Intent Request and SessionEnded Request.

2.8.11 Steps to Create a Lambda Function For Alexa Skill

To create a lambda function, the developer must use the AWS services. In the AWS, the Lambda Service must be selected. The developer must be careful when choosing the hosting location of the lambda function. Both the skill interface developed in the Alexa Developer Console and the function developed in the Lambda services must have the same geographical locations, for instance, N.Virginia.

The next step is to create a function. A new function “from scratch” must be created[29].

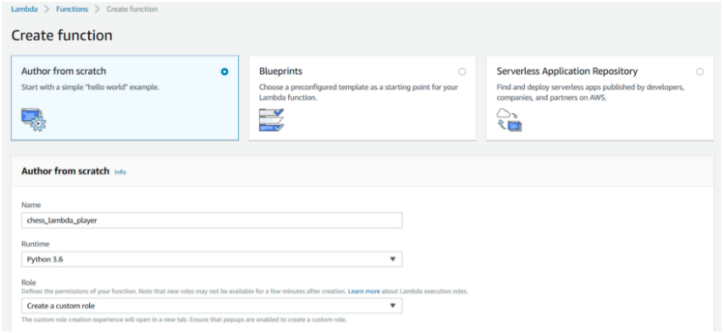

Figure 2.8: Create a Lambda Function from scretch

The next step is to give the function a name. Now that the function has been created, the function triggers and the code need to be implemented in the Lambda function just created. However, in order to have a function that is compatible with Alexa, an Alexa Skill Trigger must be added as a trigger to the function. This process is demonstrated on the image below[29].

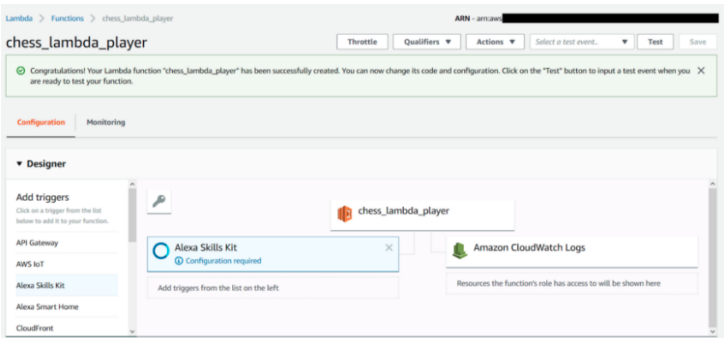

Figure 2.9: Adding Alexa Skill Kit trigger to the Lambda Function

2.8.12 Linking Lambda function with Alexa Skill

The next step is to add the id of the skill created on the Alexa Developer Console. After every step was completed, the final step is to write the code on the Lambda function code editor which will contain the logic of the Skill developed. An example of Lambda code editor containing code for an Alexa skill can be viewed below. After the code is implemented on the lambda function and the Lambda function

is linked with the skill interface in the Alexa Developer Console, the next step is to test it. The test is done in the Developer Console[30].

Chapter 3

Project Management

3.1 Introduction

The project aims to develop an Alexa Skill or template that allows the users to develop Alexa skills only using their voice. The standard way to develop an Alexa skill is through the use of the Alexa Developer Console and the Lambda service as the endpoint of the skill that contains all the logic of how the skill will interact with the user based on the input of the user. In other words, to develop an Alexa Skill, the developer must be familiar and experienced with the Alexa Developer Console and the Lambda function(including the code logic). This chapter explains the definition, planning and the technologies used to develop a template that allows users to develop an Alexa Skill only using their voice, therefore excluding Alexa Developer Console and the AWS Lambda function or any other hosting services.

3.2 Technology overview

This section contains an overview of the technologies used in the project.

3.2.1 NodeJS

NodeJS is a back-end, cross-platform JavaScript environment. It was introduced in 2009. It is an open-source cross-platform built upon the Google Javascript V8 engine. The NodeJS process is based on an asynchronous I/O model. NodeJS lets developers write tools for server-side scripting and allows them to produce dynamic web pages content before the page is sent to the user's web server. The reason why NodeJS was chosen is that it makes it easier for Alexa Skill Developers

to develop skills. Another reason was that most Alexa skills were developed using NodeJS, so, it was easier to study, analyze and learn how to develop an Alexa Skill using NodeJS[33].

3.2.2 Alexa Skill Kit

Alexa Skill Kit is a software development framework developed by Amazon and it is used to develop Alexa Skills. ASK framework allows the developer to create skills. ASK is compromised of different tools, APIs and code samples etc. This framework allows the developer to use more than 10,000 voice recognition capabilities that are available on Alexa[26].

3.2.3 AWS Lambda

AWS Lambda is a service that allows the code to run without provisioning or managing servers. Lambda runs the code on high-capabilities and availability infrastructure as well as administrates all the computer resources including monitor scaling, operating system maintenance, code monitoring etc. Lambda services allow the code to run for any type of application or back-end service. The code is deployed in the lambda function and Lambda runs the function only when needed. Another advantage of using lambda functions is that scaling is done automatically, the function can run and send requests from a few times a day to thousand times per second[29].

3.2.4 Alexa Developer Console

Alexa developer console is a tool and interface that allows the developer to create, manage and publish skills. The console is used to create intents, create utterances, manage and design the conversation flow, hold the JSON file as well as a JSON editor where the developer can edit the JSON file directly. It contains the link to the endpoints if skill code is being hosted into any service for example the endpoint to Lambda function. The beta model of the console allows creating AI Conversation skills as well. Lately, the code can be hosted on the console as well. The code section is a live code editor which can host the code for the skill and it is invoked whenever needed to send requests or to respond[34].

3.2.5 Alexa Skill Toolkit for Visual Studio Code

The Alexa Skill Toolkit for VS Code is an extension that makes it easier to locally create, test and deploy Alexa Skills. It provides a dedicated workspace for developing Alexa Skills in VS Code. The toolkit allows the testing and debugging locally in the VS Code as well. For testing, it provides a simulation of the skill where the developer can input requests and wait for the response from the skill[28].

3.2.6 Jovo Framework

Jovo is an open-source development framework used to develop cross platforms voice apps. The Jovo framework allows the developer to build voice experiences that work across different devices and platforms. Some of these platforms include Amazon Alexa, Google Assistant, Raspberry PI etc[31].

3.2.7 Jovo Debugger

The Jovo debugger allows the developers to test and debug skills locally. If the jovo framework was used to develop an Alexa Skill, the developer can simply run: $ jovo run in the console and the debugger is going to open. Jovo debugger is going to include the current states of the app(route, state, inputs etc)[31].

3.2.8 DynamoDB

Amazon DynamoDB is a fully integrated NoSQL database. It offers reliable performance, managed experience and a small API. DynamoDB is a good fit for applications with large data and strict latency requirements as well as serverless applications using AWS. DynamoDB allows to creation database tables that can store and retrieve data and serve any level of request traffic. This service provides back capabilities. Another very useful attribute of DynamoDB that it automatically spreads data and traffic for the tables over a decent number of servers to handle the throughput and the storage requirements[32].

3.2.9 Google Sheet

Google Sheets is a spreadsheet app provided by Google Docs Editors offered by Google. It looks and functions like any other spreadsheet tool but the difference is that Google Sheet is an online app. The advantages of Google Sheet is that it is web-based meaning it can be used everywhere, it works from any device and

it is online so it is possible to gather data from other sources and automatically upload them in the spreadsheet[35].

3.2.10 gspred API

gspread is a python API used to connect with Google Sheets. It enables to connect from a python application and automatically upload to or download data from the Google Sheet. The authentication to the google services can be done by using a username and password or by using a google OAuth token[36].

3.3 Software Development Process

Software Development is a process used in the software industry to develop, design and test software products. The software development process aims to produce high-quality software that meets the needs of the customers and reaches the completion of the product within the estimated time. A software development process divides the software development into smaller parts to improve the design, project management and the final product overall. The software development process is also known as the software development life cycle(SDLC). There are several models of SDLC with different approaches based on activities that take place during the development of a software product[37]. These models include:**Waterfall Model, V Model, Incremental Model, RAD Model, Iterative Model, Spiral Model,Agile Model.**

3.3.1 Waterfall Model

A waterfall model is a sequential approach to the project development process. In a waterfall model, each process must be completed before the next phase begins. In this approach, the whole software development process is divided into separate phases and the outcome of one phase is usually the input for the next phase sequentially. At the end of each phase, the whole project is reviewed to check if the project is going on the right path. The waterfall process should be used when requirements are known, there are no ambiguous requirements and the project is short[37].

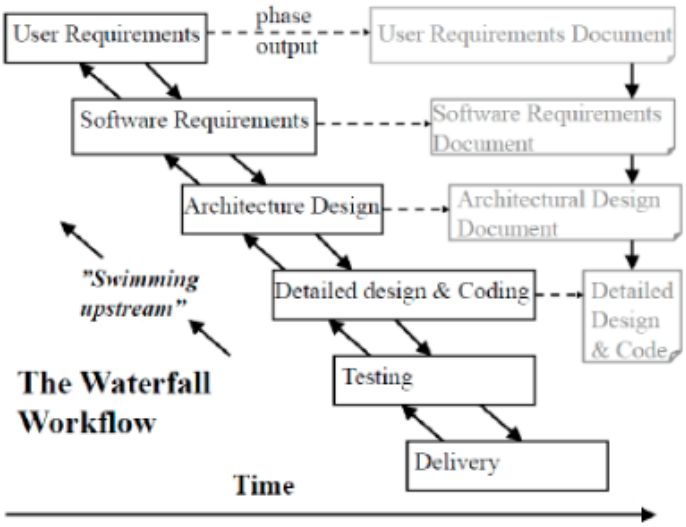

Figure 3.1: Waterfall Model

The advantage of the waterfall process are:

- Simple and easy to use
- Easy to manage since the review process can be completed on at a time.
- Works well for small projects.

The disadvantages of the waterfall process are:

- Very difficult to go back when the application is on the testing phase
- No working software is produced until late during the processes cycle
- Not good for large projects
- High amounts of risk

3.3.2 Incremental Model

An incremental model is a software development process where the requirements are divided into multiple modules of development cycles. Because of the multiple modules take place in the development, the model is sometimes called as "multi-waterfall" or "incremental-waterfall" Each on of the module goes through the requirements, design, implementation and testing phase respectively. In this model, a version of working software is produced during the first module. Each

release adds function to the previous version released. The process continues until the goal is achieved and the system is ready to be used[37].

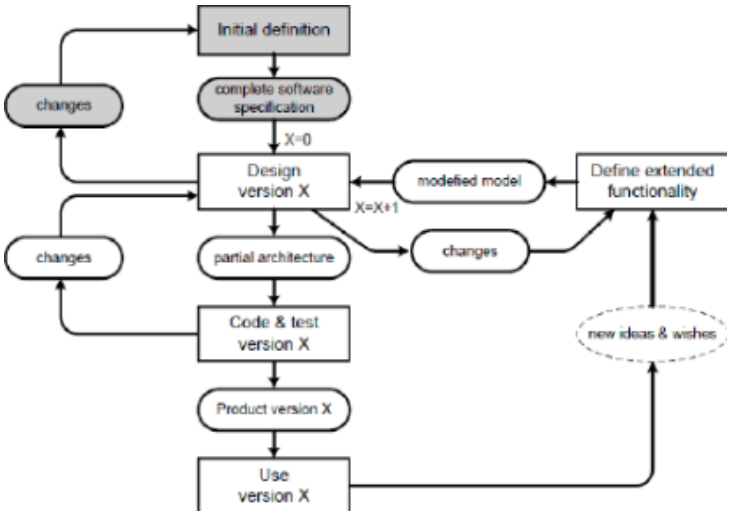

Figure 3.2: Incremental Model

Advantages of Incremental Model:

- Generate working process in the early phase during software life-cycle
- More flexible- easier to change in the later phases of the software life-cycle
- Costumer feedback during each version
- Lower initial cost
- Easy to manage risks because of the iterations

Disadvantages of Incremental Model:

- Needs good planning
- Needs a clear and complete definition
- Total cost is high

3.3.3 Iterative Model

In the Iterative model, only a few requirements are needed to start and develop the first version of the software. If the first iteration did not complete all the requirements, a new version of software will be developed with a new iteration.

Every release of an iterative version of the project over a fixed time is called an iteration. The different parts/phases of an Iterative Model are gathering of the requirements, class diagrams, implementation, testing, deployment, review, maintenance[37].

Advantages of Iterative Model: Testing is easy, Parallel development possible, Easy to apply change during any phase, Risks are resolved during iterations, Less time on documentation and more time in the design.

Disadvantages of Iterative Model: Not suitable for small projects, A lot of resources are required, Imperfect requirements may cause the design to change multiple-times, Project completion date not confirmed

3.3.4 Chosen Software Development Process

Taking into consideration that the requirements of the projects were well predefined, clear and there would not be any change of the requirements during the project life-cycle, the waterfall model was chosen. Another reason why the waterfall model was seen as the best fit for this project is the size of the project. Even though there were a lot of challenges during the development, the project was a relatively small scale project with only one requirement, develop a tool that allows to voice program Alexa. However, before starting the project development, the developer developed a lot of prototypes to better learn the technologies needed for the project. A lot of skills templates were developed before starting to work on the project.

Advantages of waterfall model for this project: *Simple to follow and use, Works well for small projects, Easy to document, There was no risk of changing the requirements, Requirements very well understood*

Disadvantages of waterfall model for this project: *Inflexible, hard to go back, No final product until the very end, High amounts of risks and uncertainty*

Chapter 4

Requirements and Analysis

This chapter will include requirements in terms of functional and non-functional requirements. Moreover, an analysis of the technologies and a justification of tools used in the project is provided. In the end, the collection a risk management plan is discussed.

4.1 Requirements

The final skill needed to deal with user input, save the user input and respond to the user the way the user's voice programmed the skill to respond. A typical Alexa Skill has Intents, utterances, slots, handlers, dialogue manager and responses on the skill code hosted on the Lambda function etc. The process of developing an Alexa Skill involves using the Alexa Developer Console for the intents, utterances, slots and dialogue management as well as code the skill responses and host the code on a Lambda Function. The main aim of the project is to achieve all of these steps via voice input, in other words, to develop a skill that allows the user to voice program an Alexa Skill instead of using all of the above steps. (Steps to develop an Alexa skill are included in the Literature Chapter as well).

Functional requirements table:

The user should be able to program the skill only using their voice	(High)
Skill's responses must be clear	(High)
User input should be stored	(High)
User should be able to create intents	(Medium)
User should be able to create utterances	(High)
User should be able to create slots	(Medium)
User should be able to control skill's dialogue flow	(Low)
User should be able to develop AI skills	(Low)
The skill should respond back to the user the saved input	(Low)

Table 4.1: Functional requirements

Non-functional requirements table:

The template should be easy to use	(High)
The developed skill can be used immediately	(Medium)
The user can easily publish the skill	(Low)
The conversation with Alexa should be intuitive	(High)
The language of Alexa responses should be formal	(High)
Efficient time to respond after the user request	(High)
The template should allow future improvements or integration	(High)

Table 4.2: Non-functional requirements

4.2 Skill Description

The Skill should be able to allow users to program a skill without prior knowledge on programming or knowledge on the technologies used for developing an Alexa Skill. The user should be able to develop the skill even without having to use the Alexa Developer Console interface. The skill should be able to handle all the users input and provide help for the users on how to voice program skills. However, the skill/template itself should be developed using the Alexa Developer Console

and other tools and the code for dealing with user input should be hosted in the Lambda function.

4.3 Implementation Tools

The best fit as an implementation tool for Alexa Skill is the ASK. ASK an SDK for developing Alexa Skills, is the main tools used to create Skills. The tool can be used locally as well through the ASK extension for the VS Code. The Developer Alexa Console is used for implementing and adding intents and utterances as well as generating the JSON file containing all the information needed to invoke the skill or any intent in the skill. To make a template that allows users to voice program an Alexa Skill, the ASK,

4.4 Programming Languages

NodeJS, Python, Go and Java programming languages are supported in AWS Lambda function. For Alexa Skills the supported languages are NodeJS, Python and Java. NodeJS is the most popular programming language to develop Alexa Skills. One of the main reasons why NodeJS was chosen as the development language was that there was more documentation and templates on how to develop an Alexa skill compared to Python. Another reason why NodeJS was seen as the best fit for developing Alexa Skills is that the developer was already familiar with JavaScript and the NodeJS framework. Since the database was planned to use for storing the user input, NodeJS was the best programming language for the integration of libraries and the database integration. However, Python would be very close to NodeJS concerning the performance.

4.5 Current Alexa Skills

Alexa Skills are interactive voice interface applications. Alexa skills are usually used to perform everyday tasks like setting reminders, checking the calendar, listening to music, checking the news or playing video games. Some of the most popular templates for Alexa Skill Development like building facts skills, music template skills were closely analyzed and studied to better understand how an Alexa Skill works. Some of the most popular templates are flash briefing, smart home, music, video(lets users consume video content), knowledge, meeting etc.

4.5.1 Fact Based Skills

Simple templates to create facts skills. The developer can upload any facts to Alexa. When the users request a fact from Alexa, Alexa will respond with the predefined fact. Fact-based skills are used by a lot of institutions to provide facts or information about their company to the customers.

4.5.2 Music Skill Templates

Simple skill using Alexa.Media.Search GetPlayableContent slot type. The user requests a song, and Alexa, using APIs plays that song.
The sample templates were very useful for the developer to learn how to develop Alexa skills and other technologies that can be integrated into an Alexa skill, for example, APIs or database integrations.

4.6 Risk Management

Risk management is important for software projects. Identifying, analyzing and planning preventative measures are critical for successful delivery of the final product within the estimated time. If not carefully planned and managed, unwanted events may take place during the development phase. Some of the risks identified and the preventative methods are listed below:

- **Lack of technology knowledge** – *Ensure enough time is spent on learning and study new technologies by reading or practical examples*
- **Addition of requirements** – textitEnsure that all the requirements are predefined before the implementation of the project starts.
- **Inadequate time to finish the project** - *Ensure necessary steps are taken in every phase of the project so the project finishes on time.*
- **Code corruption**- *Ensure that the code is backed up in the cloud or locally.*
- **Project goal not achievable**- *Ensure necessary steps are taken to change the aim of the project before it is too late.*
- **Poor quality performance** – *Ensure regular testing during the development.*

Chapter 5

Design

This chapter includes the design of the project before the implementation phase. It includes all the details taken into consideration for the designing of the interaction between the human agent and the virtual assistant. It includes all the technologies used as well as how the processes path changed during the development phase of the project. Multiple technologies were taken into consideration and tested before starting the implementation of the project. All of those designing methods and technologies will be explained in this chapter.

5.1 System Architecture

The skill is going to have intents, utterances, a JSON file, and code for responses to the user requests and handlers. The code is going to be deployed and hosted on a Lambda Function. The skill is going to be connected to the Lambda Function through the "arn" unique code of the lambda function. The "arn" code is passed to the skill's interface as an endpoint which can be modified in the Alexa Developer Console. In addition, the skill is going to have a database integration as well. The database would be used to save the user input for the next skill which is going to be developed based on the user input.

5.2 Interaction Model

The skill is going to have some no-name intents declared and the user is going to be able to name the intents and provide predefined utterances for the intents. The reasonable number of predefined no-name intents is going to be three, but

the template can be changed and added more intents if necessary. The Skill is going to ask the user how they would want to name the intent and the user would respond with a name for the intent. The name of the intent would be saved as one of the utterances for a predefined intent. When the user says that name, the intents are going to be invoked and respond to the user.

5.3 JSON file

The JSON file is generated automatically when the developer designs and declares the intents and utterances on the Alexa Developer Console. The template that is going to be developed, will generate a new JSON file for every new skill that the user wants to develop. The user input is going to be saved as utterances, including the name of the intent. However, to the user developing the skill, the name of the intent that the user gives is going to appear as the real name of the intent.

5.4 User Interaction

The intents that were identified during the design phase were predefined with descriptive names and the user would be able to "name" the intents. The user input would be save as utterances to invoke the intents. Some of the intents identified in the designing phase are: *AMAZON.CancelIntent, AMAZON.HelpIntent, AMAZON.StopIntent, HelloWorldIntent, AMAZON.NavigateHomeIntent, AMAZON.FallbackIntent, FirstIntent, SecondIntent, ThirdIntent, TeachMeSomethingInten, WhatDidYouLearnIntent.*

5.4.1 Declared Intents

The FirstIntent, SecondIntent and the ThirdIntents are intents that the user is going to develop and give utterances to invoke them. TeachMeSomethingIntent is an intent that is predefined by the developer and invoked to give information to Alexa for the utterances of other intents. While WhatDidYouLearnIntent is an intent developed by the developer as well and it is used to send a request by the user on what has Alexa saved/learned from the user input.

5.5 Saving user input in dynamoDB

One approach during the designing phase was to save the user input on database service as dynamoDB. DynamoDb a non-SQL database service offered by AWS was seen as the best fit to store user input when developing Alexa Skills. Alexa Skills are stateless, meaning the intents and handlers only run when invoked otherwise, they have no state. The attributes given in the slots are only temporary and valid as long as the skill is running. The moment the user decides to end the conversation with Alexa, the "ALEXA.StopIntent" is invoked and Alexa "forgets" all the user input or attributes that the user might have declared while talking to Alexa.

5.5.1 Persistence Attributes

An Alexa Skill must have a connection with a database. The connection with the database in the skill code is done using an adapter. The easiest way to connect a skill to a database, dynamoDB in particular, is through using the code editor integrated into Alexa Developer Console.

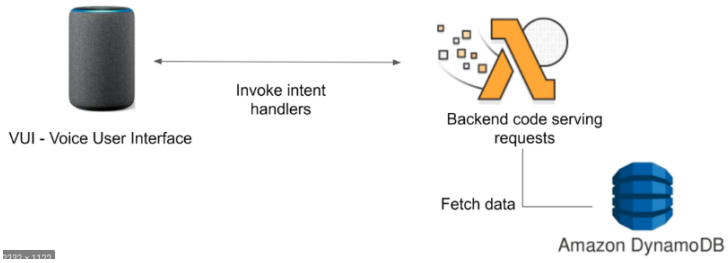

Figure 5.1: Saving user input in dynamoDB

The user input would be saved on slots, for example: "The intent name is going to be name". The name is the variable that is going to save the user input. The value on the slot is saved in a dynamoDB table. The next time the skill would be used. The same dynamoDB table would be integrated into another Alexa Skill, and when the user invokes the Skill, the skill is going to be programmed and use the utterances the user gave in the previous skill. This approach was not successful because the input saved on the dynmoDB tables could not be directly integrated into the JSON file. The user would have to manually get the saved input and add it to the JSON file.

5.6 Generating JSON file form a skill

Another approach during the designing phase was to get the user input and try to generate a JSON file containing all the intents and the user input the user declared. However, the user input would be saved on the skill, but changing the current JSON file while the skill is running was not possible. So another approach was to generate another JSON file but while not modifying the current JSON file that the skill was using to operate. The approach was to generate a new JSON file, copy that JSON file, input on another skill and the user would have a voice programmed Alexa skill. However, generating new files was not possible with the Lambda function. Lambda function is statles and cannot generate, write or upload any file. Lambda function can only create temporary files. To create a new file while the lambda function is executing, the "temp/newFile". However, these temp files are only available while the function is executing, the moment the function stops executing, all the temporary files are discarded.

5.7 Alexa Skill and Google Sheets

Since generating a new JSON file containing all the information of intents and utterances for the next Alexa Skill was not achievable, the new approach was to use other tools and technologies to achieve the goal of the project. The other approach was to get the responses of Alexa Skills from Google Sheet. The data in a specific Google sheet row would be read by the skill, and respond to the user. It was a great approach to have the skill not programmed on how to respond to the user, but to read the response from an outer source. This goal was achieved using Jovo, an open-source framework used to develop Alexa Skills or any other Voice Apps for different platforms. This approach was very successful and led to the final version of the skill.

5.8 Alexa Skill using Jovo Framework

To develop the Alexa Skill with the Jovo framework, the Jovo CLI should be installed in the system. The developer should install the jovo framework using: $ npm install -g jovo-cli in the terminal. Another advantage of using the Jovo Framework was that it can be tested in real-time. Jovo framework has its debugger called Jovo Debugger. Jovo Debugger can be started using: "$ jovo run".

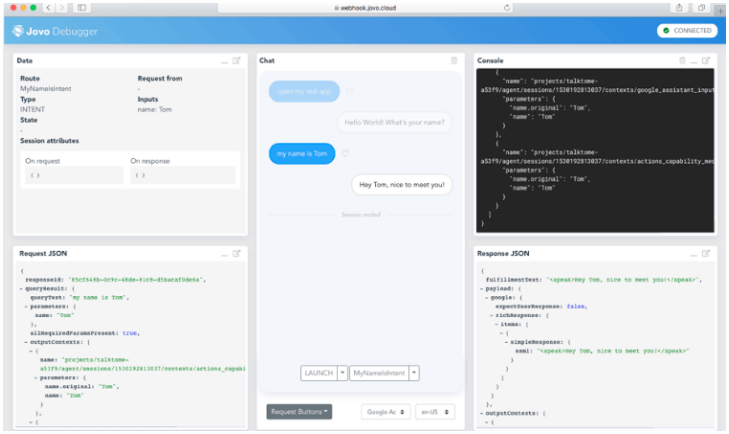

Figure 5.2: Jovo Debugger

Jovo debugger was used to test and debug the skill while developing and designing the skill that would take the respond sentences from an outer source, from Google Sheets. Jovo enables a CMS integration for Google Sheet. Using the functionalities of the Jovo Framework, the developer was successfully able to fetch data from the Google Sheet and respond to the user input with the fetched data. The code used to connect the skill with the Google Sheet:

```
const { GoogleSheetsCMS } = require('jovo-cms-googlesheets');

app.use(new GoogleSheetsCMS());
```

Figure 5.3: Fetching data from Google Sheet with Alexa Skill

5.9 Getting data from dynamoDB

Since the developer found a way to get Alexa skill to respond from an outside source using Google Sheet integration for Jovo Framework, the next step was to write in a google sheet row from Alexa skill. By making an Alexa skill writing and reading to a google sheet row, the user could program Alexa skill responds only using their voice. The approach was to store the user input into a persistence variable on the skill during the interaction and upload it to the google sheet. From google sheet, in the same row, the next skill would get the response and respond to the user depending on the request. That way, the user would have an Alexa skill that using the voice, can program the responses of another skill, in other words,

the user would have an Alexa skill that would respond to predefined sentences the user gave during the last interaction. using this approach, the user would not have to use Alexa Developer Console, or any database or host the code on a lambda function, but instead, program the skill using the voice.

5.10 Using an API Server

The best way to achieve the goal discussed in the previous paragraph was to automatically upload the data from DynamoDB table to Google spreadsheet, where, the other skill would fetch those data and use as responses to the user requests was to use CData API server. CData API serves offers services that when integrating the dynamoDB, create an API which allows to automatically upload data from dynamoDB to Google Sheets. The developer tried this approach and was able to use the CData API server, but to use the dynamoDB integration for the CData API Server, a fee to the service would have to be paid. However, this approach was helpful in the sense that led to the solution of the problem. The developer found a way on how to achieve the goal by using other tools and approaches. The final approach or the implementation will be closely discussed in the next chapter.

Chapter 6

Implementation and Testing

This chapter discusses how the design considerations made in the previous chapter led to the final implementation of the system. It discusses how the implementation was done and the different technologies that were used for the final system. This chapter includes a discussion about the testing of the final system as well.

6.0.1 System description

As discusses in the previous chapter, a skill has already been developed in the designing phase. The skill developed, was able to read and respond to the user's request from an outer source. That was achieved using the Jovo framework and the CMS integration for Google Sheets. However, to have a voice programmable skill, the skill should be able to write on specific rows of Google Sheet as well. The skill would ask the user on what they would like to respond on specific requests, save the user input in an attribute, upload the attribute on a specific row on the spreadsheet so the next time the user would make the request, the skill would read the response from the google spreadsheet. The response that the user predefined only using their voice.

6.0.2 Second skill

The best approach by the developer to add the user input on a spreadsheet after many unsuccessful attempts discussed in the previous chapter was to develop a new skill that automatically uploads the user input in specific cells of a Google Sheet. The new skill would be a template which can be further developed by any future developer that wants to make custom skill which allows voice programmable skills.

It can also be used by different organizations to get the data directly from a google spreadsheet or by any institution that has to frequently change the responses to the user's requests. This can simply be achieved only by using voice as the most natural way of interaction, to change the responses using the developed template.

6.1 Interaction Model

The interaction model of the second skill consists of a JSON file that contains the skill intents, utterances and slots defined. Alexa Developer Console was used to create the JSON file for the skill. However, in the code files, which are hosted by the Alexa Developer Console, other interaction models can be found as well. Those interaction models are only valid for the skill while the skill is running. These interaction models are used If the user is going to make any wrong input, or just to ask the user what action they would like to perform. The JSON files in the code are used to reprompt to the user. Some of the repromts include:

- **HELP_REPROMT** – *what would you like to do*
- **FALLBACK**: - *Sorry I did not understand that*
- **CANCEL_STOP_RESPONSE**: - *Good bye,*
  -Okay I'll be here if you need me.

6.1.1 Intents, Utterances and Slots

Utterances and slots have been predefined for the skill by the developer. The JSON file containing the utterances and the slots was developed using Alexa Console Developer. There are a total of 6 intents. Those 6 Intents include:*AMAZON.NavigateHomeIntent, AMAZON.CancelIntent, AMZAON.HelpIntent, AMAZON.StopIntent, AMAZON.FallbackIntent, GoogleSheetsIntent.*
The GoogleSheetsIntent contains four utterances. Those utterances are:

- *yes I want to add* {***else***}
- *can I add* {***sentence***}
- *can you save* {***sentence***}
- *can you save* {***else***} *for the new skill*

By using these utterances, the user is able to save the responds in google sheet where the other skill is going to get the input and respond based on those inputs. The skill has two different slots. The else and the sentence. These two slots have the same purpose. Save user input into a cell. The reason why the developer used two different slots for the same purpose is that, during the implementation, there would be a conflict with the utterances in the intents on using only one slot. By using two different slots, any conflict or problem using the skill by the user would be prevented.

6.1.1.1 Slot types

Slots are variables declared in an utterance of intent and save the user input temporarily while the skill is running. The slots have different types. Some of the slot types include AMAZON.FirstName, AMAZON.City, AMAZON.Actor etc. To get sentences as user input the slot type had to be declared as AMAZON.SearchQuery. AMAZON.SearchQuery is a slot type to used dynamic questions from the user. However, it was the only slot type that would allow to save the user input as a sentence and only as one word.

6.2 Final Implementation of second Alexa skill

The second skill was developed to save the user input which the first skill is going to get the saved input and respond to the user. The second skill was developed using Alexa Developer Console. The developer decided to host the code of the skill on the Alexa Develop Console and not on the Lambda Function. The reason why the developer chose to host the code on Alexa Developer Console rather than on Lambda function is that during the development a lot of testing had to be performed, so it was easier to test the skill when it was being hosted on the Console. Below is a representation of the code hosted on the Alexa Developer Console.

6.2.1 Connecting the second skill with Google Sheet

To connect the second skill with google sheets the developer used a library, gspread, which when connected with a python application can directly modify cells in google sheet. The gspread, is a Python API for Google Sheet. Since gspread is supported

only by Python applications, the developer decided to develop the second skill using Python.

6.2.2 Google Developer Console

To connect the Google Sheet with Alexa skill, a new project on Google Developer Console should be made. The next step after creating a project on the Google Developer Console is to enable two APIs. The first API is the Google Drive API and the Google Sheet API. Since the Google Sheet is stored in Google Drive, access to both is needed by the skill. The same process is done to enable the Google Sheet API. The next step after enabling both the APIs is to obtain credentials for the project. A new credential is developed using Service Account.

6.2.2.1 Credentials File

The next step after creating the Service Account email is to click on the email generated and create a key. Immediately after creating the key in JSON format, a file will be downloaded on PC containing the JSON format of the key generated. The next thing is to add credentials to the skill. After adding the credentials to the skill, the Google Sheet should be shared using the email generated on Google Developer Console.

6.2.3 Integration of spreadsheet into the skill

To integrate the Google Sheet to the skill the gspread library was used. Firstly the sheet url and the gspread service account credentials obtained from the Google Developer Console. The next step is to access the sheet on the row so the skill rows where to get or upload data into. The picture below contains the code to write use input into a specific row in the spreadsheet.

```
class GoogleSheetsIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        return is_intent_name("GoogleSheetsIntent")(handler_input)

    def handle(self, handler_input):
        language_prompts = handler_input.attributes_manager.request_attributes["_"]
        slots = handler_input.request_envelope.request.intent.slots
        first = slots['sentence'].value

        worksheet = sh.get_worksheet(0)
        worksheet.update('B2',first)

        speech_output =  ("It was successfully added as a skill response to your new skill, would you like to add another response")

        second = slots['else'].value

        worksheet = sh.get_worksheet(0)
        worksheet.update('B3',second)
        #worksheet.update('B1', 'Bingo!')

        #speech_output = cell_value
        #reprompt = cell_value


        return (
            handler_input.response_builder
                .speak(speech_output)
                .set_should_end_session(False)
                .response
            )
```

Figure 6.1: Writing user input to a Google cell from Alexa skill

6.2.4 Voice programmable skill

After the second skill is developed as well, the Google Sheet has linked to the first Alexa skill. The first Alexa Skill was developed using Jovo Framework and CMS integration for Google Sheet. Whatever the user would want to input on the first skill to get as a response on specific requests, the second skill would fetch that input and respond as voice programmed by the user in the first skill. The system developed, allows the user to develop a skill without having to use Alexa Developer Console or the code which is hosted on the Lambda Function. It also allows other developers to make different custom skills for different categories and allow their customers to self program those skills only using their voice.

6.3 Testing

Testing of the system was performed using Alexa Developer Console. After the system was completed, both the skill were tested in the Console. However, during the development process, the first skill was tested using Jovo Debugger. After the skill was developed and deployed on the Alexa Developer Console, and the code on Lambda, the testing was performed on the console. Regarding the second skill, the testing was performed on the console during and after the development. The testing was performed by invoking the second skill, adding a sentence for the response, and then invoking the second skill, making a request and testing if the responses is the same as it was given on the first skill. Below can be found an

image representation of the testing phase after the development and implementation.

The first image is the testing of the second skill the VoiceProgrammingSkill which is develop to program the responses of the first skill.

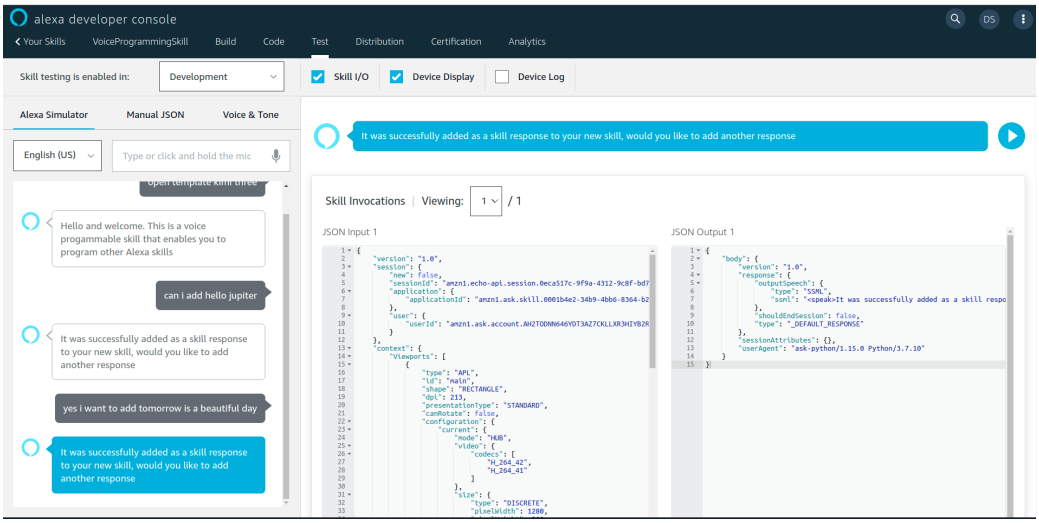

Figure 6.2: Testing the *VoiceProgrammingSkill* using Alexa Developer Console

While, the first skill, the JovoSelfProgrammable skill, responds back to the user, what the user inputed on the second skill showed on the previous image.

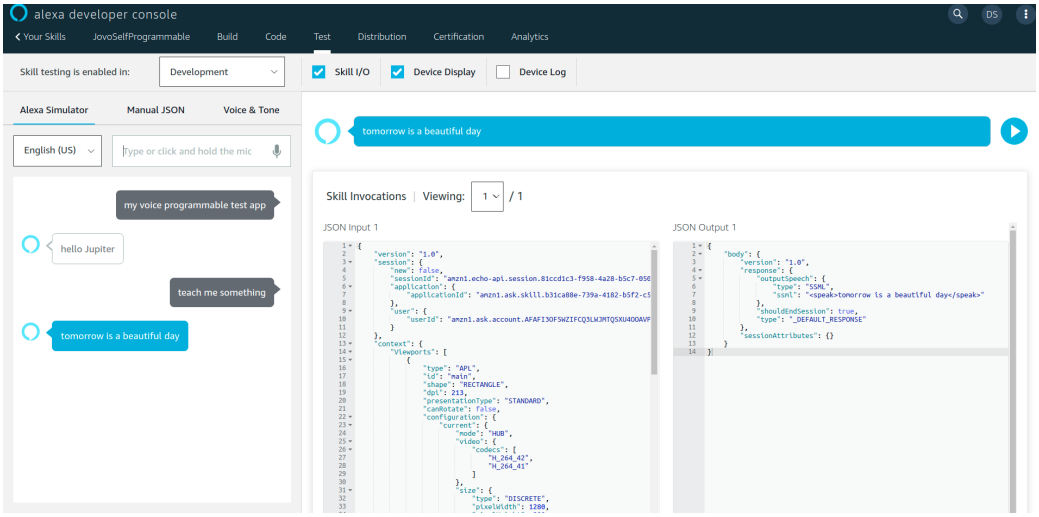

Figure 6.3: Testing the *JovoSelfProgrammable* using Alexa Developer Console

Chapter 7

Evaluation

The evaluation chapter is going to discuss and define if the project was successful. Three types of procedures are used to evaluate this project.

- Evaluating of the system in regards to objectives
- Evaluating the system in regards to software processes
- Self Evaluation

7.1 Evaluating an system in regards to objective

The main objective of the system is to have a template that allows to voice program Alexa Skills. The table below shows the objectives that could be achieved and the objectives that were not achievable.

Project Objectives	Achieved
Voice program an Alexa Skill	YES
Voice Program Intent names	NO
Voice Program the code hosted in Lambda Function	NO
Voice program skill's responses	YES
Voice program utterances	NO

Table 7.1: Objectives evaluation

The first and most important objective, to have a template that allows the user to develop Alexa skills has been achieved. The process of how this objective was achieved is discussed in the implementation chapter.

The second objective, to Voice Program the utterances and Voice Program Intent names, was not possible to achieve due to constraints of the code being hosted on the Lambda Function. Alexa Skills have a JSON file that is used for the intent names and the utterances, however, editing the file while the skill was executing on the Lambda Function was not possible. The other approach, generating a new JSON file for a new skill with the intents and utterances declared from the user was not achievable because of Lambda Function constraints that do not allow to create a new File during execution.

Voice Program the code hosted in Lambda Function that deals with the responses of the skill-based on the intents declared on the Alexa Developer Console, was not achievable because there is not any technology available that allows humans to code programs or applications in nodeJS or any other programming language by only using their voice. However, the goal of having responses of the skill for the different requests being declared by voice was achieved successfully.

7.2 Evaluating the system in regards of software processes

The Software process followed for this project was the waterfall model. This section will discuss the main advantages and disadvantages of the chosen software engineering process.

Advantages analysis

- Simple and easy to use: It can be observed in the report that the software process was successful for this project. It was easy to use and easy to document everything during the process.
- Easy to manage since the review process can be completed one at a time: The project was easy to manage and implement or design different approach

during the development. Moreover, the project was finished within the estimated time.

- Works well for small projects: The project was relatively small, it only had one request. The request was to make an Alexa Skill that allows developing other skills only using a voice interface. The main goal of the project was not easily achievable because it required knowledge of a lot of technologies that were new to the developer and because of the lack of documentation about such projects as well.

Disadvantages analysis

- No working software is produced until late during the processes cycle: The advantage of having no product to demonstrate until late can be observed in the report as well. During the designing phase, the developer tried a different approach to achieve the goal but was not successful until very late. However, the project was finished as estimated.
- Not good for the large project: As discusses earlier, this software development process is good for small projects and this proved to be true in the case of this project. However, since this was a relatively small project, the developer cannot evaluate this disadvantage for large projects.
- High amount of risk: The amount of risk was relatively high for this project. However, necessary steps were taken during the analysis phase to manage the risk during the designing and implementation phase. Risk management is discussed in the Requirements and Analysis Chapter.

7.3 Self Evaluation

In this section, the developer will evaluate the main constraints and issues faced during the whole process.

7.3.1 Constrains

Some of the constraints in the project were the lack of documentation or sources for voice programming. The developer had to make a lot of research and testing during the designing phase which led to the next constraint, the time. Due to lack of documentation, the developer had to do a lot of testing in the designing phase

which consumed most of the time which was not initially estimated. Another constraint was the lack of knowledge on the technologies used for developing an Alexa Skill. However, the biggest constraint was the restrictions of usage limits on Amazon Web Services. Amazon Web Services offers a "Free Tier Service" which expires within a limit amount of time.

7.3.2 Evaluation

The developer had the opportunity to learn about the new technologies that were required to finish the project. NodeJS, Alexa Developer Console, Lambda Function, DynamoDB, gspread, AWS Services were all-new or tools that the developer had only basic knowledge about. During the development, the developer had to learn of the above technologies even if in the final implementation some of them were not required. The overall process was a very positive learning opportunity for the developer. There might have been a difficult situation and the developer had to change the approach to the problem more than once, but in the end, the project can be considered a success.

Chapter 8

Summary and Conclusion

8.1 Summary

The project aimed to develop a tool that allows users to voice program Alexa Skills. This was achieved using two different skills. The skill development uses APIs to connect with Google Sheet. The integration of Google Sheet to Alexa skills was done through the creation of APIs and generating keys. The second skill takes the user input and writes it into a specific cell in Google Sheet, while the first skill, reads the user input and based on the request, responds to the user using the input given on the previous skill.

The whole designing process and different approaches to reach the aim of the project were discussed in the design chapter. The main requirements of the system were to have allowed the user to program and develop a skill only using their voice.

Other requirements included having a user-friendly system and formal language etc. The requirements and the risks of the project were discussed in the requirements chapter. The implementation and the final version of the system are closely discussed in the implementation chapter, where, the developer explains step by step how the two skills were linked together to achieve the aim of the project, developing a skill that allows voice programming of other skills.

The second and the third chapter discuss the literature review and the project management respectively.

8.2 Conclusion

A system that allows the user to voice program the responses of an Alexa Skill has been developed. Different approaches were made by the developer on the designing phase, however, the implementation is done using two different skills. The first skill responds to the user by an outer source. It does so by fetching the data into a specific cell in google sheet as programmed and responds to the user. The second skill, on the other hand, takes the user input for specific requests and stores them automatically in the google sheet. The google sheet is used as a link between the skill used to program the responses and the skill which is programmed to respond based on the input of the second skill. The first skill is hosted on the Lambda function while the second skill is hosted on Alexa Developer Console. The testing and the evaluation of the skill was successfully achieved and discussed in the previous chapters. Moreover, the skill lets space for further improvements by other developers, as just a template that describes the voice programming of skill is developed. It can be concluded that the aim of the project was successfully achieved.

Bibliography

[1] A. Dix, J. Finlay, G. Abowd and R. Beale [et al.], *Human-computer Interaction*, 3rd ed. Harlow: Pearson Education, 2004, pp. 11-164

[2] Al-Dabet, Saja & Jusoh, Shaidah, *Usability Evaluation of iPhone Built-in Applications.*, Romania, 2019.

[3] D. Perez Garcia, S . Saffon Lopez and H, Donis, "Everybody is talking about Virtual Assistants, but how are people really using them?", 2018, Available: 10.14236/ewic/hci2018.96 [Accessed 4 May 2021].

[4] Karray, Fakhri, *Human-computer interaction: Overview on state of the art. International journal on smart sensing and intelligent systems.*, Waterloo, Canada , 2008.

[5] Kortum, P., *HCI Beyond the GUI: Design for Haptic, Speech, Olfactory, and Other Nontraditional Interfaces.*, Amsterdam: Elsevier Science, 2008, pp. 147-159

[6] Ying-Li Tian,Takeo Kanade,Jeffrey F. Cohn, "Facial Expression Analysis" in *Handbook of face recognition.*, Springer-Verlag London Limited, 2011,

[7] Daniel Jurafsky, James H. Martin, "Chatbots & Dialogue Systems" in *Speech and Language Processing*, 2020,

[8] Mctear, Michael, *Spoken Dialogue Technology: Enabling the Conversational User Interface. ACM Comput. Surv.. 34. 90-169.* , 2002.

[9] Ben Shneiderman,Catherine Plaisant *Designing the user interface.*, Pearson-Education,Inc. 4th ed, 2005.

[10] Juang, B. , Rabiner, Lawrence, *Automatic Speech Recognition - A Brief History of the Technology Development*, 2005.

[11] IBM, "Speech Recognition," [Online]. Available : https://www.ibm.com/cloud/learn/speech-recognition. [Accessed 23 April 2021].

[12] M. Halle, *[9]M. Halle, From Memory to Speech and Back: Papers on Phonetics and Phonology, 1954-2002.*, De Gruyter Mouton, pp. 25-28.

[13] Garcia-Serrano, Rodrigo-Aguado, Luis , Francisco. , *Natural Language Dialogue in a Virtual Assistant Interface.* , 2002.

[14] Suket Arora1, Kamaljeet Batra, Sarabjit Singh, *Dialogue System: A Brief Review.*,

[15] A Pargellis, Hong-Kwang Kuo, Chin-Hui Lee, *Automatic Dialogue Generator Creates User Defined Applications.*, 2000.

[16] IBM Conversaitional AI, "Conversational AI" [Online]. Available : https://www.ibm.com/cloud/learn/conversational-ai [Accessed 06 May 2021].

[17] IBM Conversaitional AI, "Conversational AI" [Online]. Available : https://www.ibm.com/cloud/learn/conversational-ai [Accessed 06 May 2021].

[18] Dialogue Management, "Dialog Management with Alexa Conversations" [Online]. Available : https://developer.amazon.com/en-US/alexa/alexa-skills-kit/dialog-management [Accessed 06 May 2021].

[19] S Singh, M Kearns, M Litman, M Walker *Reinforcement Learning for Spoken Dialogue Systems.*, 1999.

[20] M.Hoy, *Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants.*, vol.37, no.1, pp.81-88 2018, Available: 10.1080/02763869.2018.1404391.

[21] Google Store, "Google Assistan Skills," [Online]. Available : https://store.google.com/gb/magazine/compare_nest_speakers_displays. [Accessed 12 May 2021].

[22] Amazon Alexa, "Conversational AI," [Online]. Available : https://developer.amazon.com/en-US/alexa/alexa-skills-kit/conversational-ai. [Accessed 4 May 2021].

[23] Amazon Alexa skills, "Statistics on Alexa Skills available," [Online]. Available : https://voicebot.ai/2020/10/25/amazon-alexa-skill-growth-has-slowed-further-in-2020/. [Accessed 20 April 2021].

[24] Amazon Alexa SDK, "Alexa Skills Kit SDKs," [Online]. Available : https://developer.amazon.com/en-US/alexa/alexa-skills-kit/get-deeper/sdk. [Accessed 20 November 2020].

[25] Amazon Alexa skill, "How Does Alexa Work?," [Online]. Available : https://www.amazon.com/how-does-alexa-work/b?ie=UTF8&node=21166405011. [Accessed 18 March 2021].

[26] Alexa Skill Kit, "What is the Alexa Skills Kit?," [Online]. Available : https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html. [Accessed 10 March 2021].

[27] Steps to Build a Custom Skill, "Steps to Build a Custom Skill," [Online]. Available : https://developer.amazon.com/en-US/docs/alexa/custom-skills/steps-to-build-a-custom-skill.html [Accessed 5 November 2020].

[28] Alexa Skill Toolkit, "Get Started with the Alexa Skills Toolkit for Visual Studio Code," [Online]. Available : https://developer.amazon.com/en-US/docs/alexa/ask-toolkit/get-started-with-the-ask-toolkit-for-visual-studio-code.html. [Accessed 22 December 2020].

[29] Lambda Function, "Create a Lambda function with the console" [Online]. Available : https://docs.aws.amazon.com/lambda/latest/dg/getting-started-create-function.html [Accessed 10 December 2020].

[30] Host a Custom Skill, "Host a Custom Skill as an AWS Lambda Function " [Online]. Available : https://developer.amazon.com/en-US/docs/alexa/custom-skills/host-a-custom-skill-as-an-aws-lambda-function.html [Accessed 26 January 2021].

[31] Jovo Framework, "Jovo Framework Documentation," [Online]. Available : https://www.jovo.tech/. [Accessed 17 April 2021].

[32] Amazon DynamoDB, "What Is Amazon DynamoDB?," [Online]. Available : https://docs.aws.amazon.com/amazondynamodb/latest/developerguide /Introduction.html. [Accessed 28 November 2020].

[33] ASK SDK for Node.js, "ASK SDK for Node.js", [Online]. Available : https://developer.amazon.com/en-US/docs/alexa/alexa-skills-kit-sdk-for-nodejs/overview.html. [Accessed 20 March 2021].

[34] Amazon Alexa Developer Console Documentations, "Alexa Developer Console Documentation", [Online]. Available : https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html [Accessed 5 March 2021].

[35] Google Sheet, "Google Sheet," [Online]. Available : https://www.google.com/sheets/about/ [Accessed 13 May 2021].

[36] gspread, "gsrpead API Documentation," [Online]. Available : https://docs.gspread.org/en/latest/index.html [Accessed 18 May 2021].

[37] I. Saker, F. Faruque, U.Hossen and A. Rahman , *A Survey of Software Development Process Models in Software Engineering.*, vol.9, no. 11, pp. 55-70, 2015, Available: 10.14257/ijseia.2015.9.11.05.