Universität Ulm | 89069 Ulm | Germany

Faculty of Engineering, Computer Science and Psychology
Databases and Information Systems Department

Supporting remote therapeutic interventions with voice assistant technology

Bachelor's thesis at Universität Ulm

Submitted by:
Jens Fischer
jens.fischer@uni-ulm.de

Reviewer:
Prof. Dr. Manfred Reichert

Supervisor:
Dr. Marc Schickler

2019

Version from September 13, 2019

Composition: PDF-$\LaTeX\,2_\varepsilon$

Abstract

Nowadays, digital personal assistants are incorporated in many devices. Smart TVs, smartphones and stand-alone voice assistants like Amazon Alexa allow owners to control their smart home systems, play music on command or lookup information on the internet via voice queries. Using custom skills from various third-party vendors, almost any company can have a skill supporting the needs of their customers or control their devices.

Furthermore, therapeutic interventions represent a vital part of most therapies, but there are some underlying struggles during therapies for which therapists can utilize the support of smart mobile devices. As an extension of an already existing system called Albatros, its features have been converted into an custom Alexa skill called remote interventions.

But voice assistants can do more than improving everyday life, like helping people during medical therapies. A vital part of such therapies are therapeutic interventions, but therapists often face struggles when monitoring a patients progress and results. To overcome this problem, an existing system called Albatros allows a therapist to review the patients status. As an extension to the existing Albatros system, its features have been incorporated into a custom Alexa skill called remote interventions.

Aiming to contribute to the proof of concept, the objective of this thesis is to demonstrate the development process of a custom Alexa skill which implements the features of retrieving exercises, allowing patients to record feedback via a smart speaker which can then be accessed by the therapist. With the addition of a notification feature the system also supports patients in remembering how and when to do their exercises properly. Due to the proof of concept nature of the project, apart from the actual development process, an analysis of whether or not the ideas and features translate well into a voice driven platform is performed.

Acknowledgment

I want to acknowledge the support of my family and friends in the time period of creating this thesis. Special thanks to my brother Thomas Fischer for proof reading and to Dr. Marc Schickler, the supervisor for this project, for all the help and trust in letting me create something in connection to his dissertation.

Contents

1 Introduction

Intelligent personal assistants (IPAs) are quickly rising in popularity for businesses and even more in the home environment. Since 2014, which marks the beginning of sales of Alexa devices, more than 100 million products with built-in Alexa capabilities have been sold. Counting all digital voice assistants (VA), which includes voice assistants in smart TVs, smart phones and wearables, the overall number of digital voice assistants is predicted to triple in the next five years from 2.5 billion to a staggering 8 billion in 2023 [1]. IPAs, or more specifically smart speakers are good at doing particular tasks like playing music, controlling electronics in a smart home environment and quick information searches [2]. Amazon, with the Alexa and Echo technologies, holds the biggest market share, approximately 70 percent in the USA [3], which is one of the reasons it was chosen as the platform for this project.

With smart speakers like the Amazon Echo, these tasks can be performed with only verbal commands and no physical effort [4]. Having no need for haptic touch interactions with small devices like smart phones is a big factor in acceptance especially for the elderly [5]. Additionally, these voice assistants can be installed on almost any device apart from standalone devices like *Amazon Echo* and *Google Home*. Nowadays, smart TVs, personal computers and smart mobile devices are capable of the same features exactly [6].

The foundation and groundwork for this thesis is provided by a research project called **MobileTx**. It was conducted as part of a dissertation by Marc Schickler [7] called "A framework for the mobile support of therapeutic interventions", translated from German.

As mentioned in one of the research papers [8], "therapeutic interventions constitute a fundamental part of most therapies". As part of such an intervention, complex homework is assigned by the therapist to the patient, who must memorize and perform the tasks at home and unsupervised.

The goal of the research project was to support therapeutic interventions and specifically therapeutic homework to improve effectiveness and outcome of the therapy.

Therapists demand smart mobile device support in order to have more insight into the homework process and make adjustments on the fly. To achieve this, a proof of concept prototype platform which is called *Albatros* was designed and created.

Furthermore, to accomplish the objective, with the concept of mobile processes in mind, a configurator has been developed to enable therapists to create, configure and manage therapeutic interventions for their patients. For the purpose of validating the configurator as a component of the **Albatros** platform, this usability study [9] was conducted. The study, in which I have also participated, came to a result which confirmed the configurator as a working tool to create homework based on mobile processes. One of the tasks in the study was to create an instruction text based on a mobile process diagram. This instruction text constitutes a major part of the project in this thesis.

The major aspects of the Albatros project in addition to the configurator's purposes are to support patients in successfully performing their homework, a way for the patients to provide feedback for the homework available for the therapist to see, and a notification feature to remind the patient to do his exercises [10].

Specified mobile processes and the Albatros project as a whole facilitate the development of smart mobile device applications to support remote therapeutic interventions [11].

Finally, this builds the basis for this thesis, the objective of which is described in the next section.

1.1 Objective

The objective of this thesis is to incorporate the Albatros features into an Alexa skill. The paper serves as a proof of concept, this means that a prototype Alexa skill called *remote interventions* has been developed and deployed on the Amazon Alexa platform.

It aims to support patients during their therapy who either already own a smart speaker or who simply prefer verbal commands over a small and sometimes complicated smart mobile device. This is especially important for older patients that are not very familiar with modern technology. With natural language understanding an effort has been made to make usage as simple as possible for everyone.

In addition to showcasing the development process of a custom Alexa skill, there is an evaluation of how the features have been implemented with the available tools and if the usage of a digital voice assistant makes sense in this scenario.

1.2 Structure of the Thesis

After explaining the origin of this project and giving an introduction in section 1, the main principles of therapeutic interventions and the Amazon Alexa environment are elaborated in the fundamentals chapter 2. Following that, a scenario with the functional and non-functional requirements are given to create an outline of what the skill should be capable of doing in the requirements chapter 3. These functionalities are further explained in the concept and design section 4 with insights on the conception of the features design and architecture. How everything was carried out is documented in the implementation section 5, with specifics about the tools used, the dependencies and requirements, as well as a detailed look into the development process. The thesis ends with a summary and outlook with a discussion of the outcome of the project in chapter 6.

2 Fundamentals

This chapter will focus on basics of remote therapeutic interventions as well as Amazon Alexa technology and how it can enable remote therapeutic interventions.

2.1 Remote Therapeutic Interventions

Therapeutic interventions in this context are efforts made by professionals to improve the health or comfort of patients that seek help with either physical or psychological problems. These can be applied either during a meeting or in-between meetings, in the form of a therapeutic homework assignment. In psychotherapy, for example with cognitive behavioral therapy (CBT), which includes exercises for changing behaviour and thought patterns, performing corresponding techniques in the form of homework proves to have an essential effect on the rate of success [7]. Similarly, in physical therapy, in-session work or massages can be beneficial short term, yet long term success almost certainly includes performing homework exercises. There are a lot of different types of homework exercises that can be administered by a licensed therapist. They range from exercises to improve mobility and strength in certain areas to correcting imbalances. Frequent and careful but proper execution of these exercises can help with recovering from an injury and reducing pain. Considering the patients specific situation and therapy goals, the therapist plans and recommends exercises that should be performed at home in order to improve personal sessions and to speed up the process of healing in general.

Due to descriptions and instructions of the homework being too complex and difficult to remember, patients often have troubles in correctly performing exercises at home and consequently adherence gets worse.

To prevent this from happening, memorizing the exact steps on how to perform an exercise is no longer necessary. In [8], an approach has been developed that provides the needed assistance for patients along with creating an opportunity for patients that allows them to give feedback outside of a session. Additionally, therapists can monitor homework outcomes remotely, which gives them time to properly adjust the exercises as well as prepare for the next session more effectively.

Naturally, patients can still simply forget to do their homework partially or completely, hence notifications can be implemented to remind patients at specific times.

The ability for the therapist to gain useful feedback in order to adjust exercises and give tips is another crucial aspect that lacks support, which is something the Albatros system and this thesis seek to address. Therapists are looking for a way to collect more feedback in real time and provide unlimited modifications to the therapy plan of their patients.

This thesis and the corresponding Alexa skill will serve as a proof of concept of a voice application extending the already existing Albatros system and smart phone application. Since the topic of psychotherapy would require more previous knowledge, this thesis is focused mostly on the physical therapy aspect and the implementation details.

2.2 Voice Assistants - Amazon Alexa

Virtual Personal Assistants (VPAs) are not limited to Alexa, there are several more voice assistants able to perform similar tasks but with different technologies. The second most popular set of devices is developed by Google, with *Google Home* as its main smart speaker device. They offer five different smart speakers and displays which all respond to "OK, Google" and "Hey, Google" as wake words. On top of that, all Android owners can use the pre-installed Google assistant on their smart mobile device. The assistant allows users to schedule events and alarms, search the internet for information or adjust settings on the device, all with natural voice commands or keyboard input. The Google assistant uses a Deep Neural Network (DNN) [12] for the natural language processing algorithm which allows it to have two-way conversations with context awareness.

Also worth mentioning is Apple's virtual assistant called *Siri*. It is an included feature in almost all Apple devices on the market and can perform similar tasks, including controlling compatible smart home devices, asking general questions or playing music from their own music provider, Apple music. Apple was hesitant to join the stand-alone smart home device market but eventually did so with the *HomePod* halfway through 2018.

One interesting feature, available for Google's and Amazon's assistant but not for Siri, is multi-user voice recognition. Different voice profiles in a home or work environment can be configured and consequently recognized for personalized and customized responses and access permissions.

For this project, Amazon Alexa as the voice assistant and the Amazon Echo as the smart speaker device were chosen. They are the most sold devices and together with the Google assistant, they provide the biggest variety and the most well-rounded set of features. Amazon has seven smart speakers and displays available which all utilize automatic speech recognition (ASR) and natural language understanding (NLU) to improve intent recognition from spoken text, more details in 4.2.1. With virtual assistants, especially Alexa, every interaction has to begin with the wake word, a drawback that still prevents truly natural conversations but is projected to improve in the near future.

Before diving into the details of the application and how it functions specifically, it is helpful to have some background information on how Amazon's voice assistant devices operate, what skills are and how they work to customize the user experience.

2.2.1 Alexa Skills

Generally speaking, all Alexa devices work with so called skills, many of which are already enabled on the smart speaker. Without doing anything more than setting up an internet connection, built-in capabilities like asking for the weather, playing music from multiple music streaming corporations like Spotify and things like joke skills can be used directly. Other custom skills from third-party vendors can be downloaded from *the Skill Store*. Similar to the Google Play Store or the Apple App Store it is a digital online market place. However, most if not all applications are free to download. To earn money, developers can enable in-skill purchases of premium content if available. A new skill can be developed and published by anyone as long as they have a free Amazon account and the skill passes a short review process.

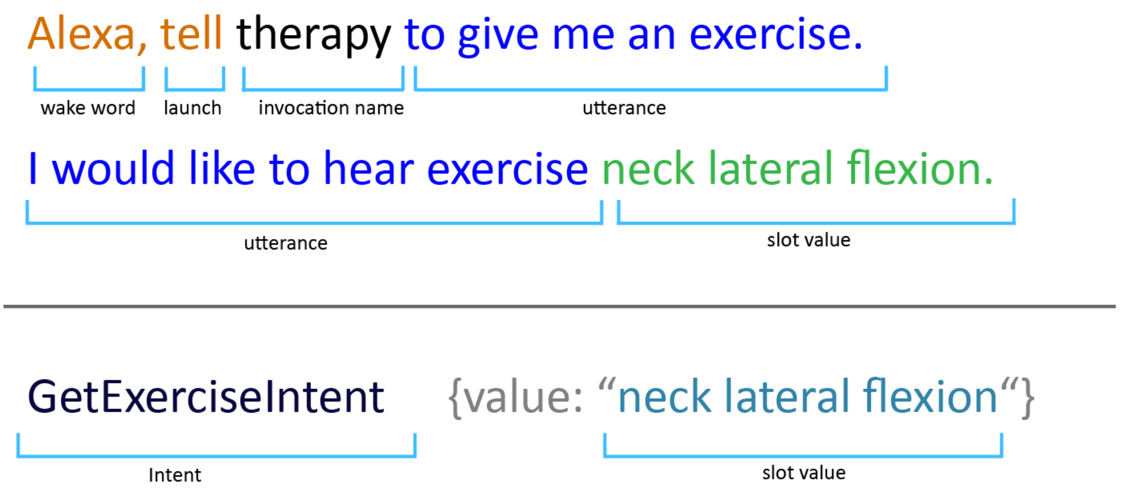

Alexa, tell therapy to give me an exercise.
wake word
launch
invocation name
utterance
I would like to hear exercise neck lateral flexion.
utterance
slot value
GetExerciseIntent
{value: “neck lateral flexion“}
Intent
slot value

Figure 2.1: Model

2.2.2 Interaction Model and Skill Invocation

The Alexa enabled device starts listening after it hears the wake word, by default this is "Alexa", but it can be customized to listen for "Echo", "Computer","Amazon" or newly introduced "Jarvis". The additional audio after the wake word is streamed to the Alexa online service for evaluation.

In 2.1 we can see the general structure of a spoken sentence Alexa will listen to and be able to work with. After the wake word, the user has to invoke a specific skill. There are a few different ways to do that using launch phrases.

Either you already know what you want from a skill:

- "tell ... to,that"
- "ask ... to,for,about,whether"
- "get ... from"
- "search ... for"

Or if you just want to launch a skill:

- "launch, open, start, run ... "

Between the launch word and the following preposition, the user has to say the invocation name of the skill he is trying to use. This invocation name is used to identify the skill and will initiate the conversation with the device [13]. When creating a custom skill, the developer has to define sample utterances, which are a set of phrases the user is likely to say when using the skill. To fulfill the spoken request by a user, multiple utterances that lead to the same action are grouped together and mapped to an intent. All the intents constitute the core functionality of the skill.

To complete the created interaction model, the intents need slots that can be filled by the user input. There are many built-in slots which can be used to recognize specific limited inputs like *AMAZON.time, AMAZON.name or AMAZON.city*. When in need of a new slot type which does not exist yet, the developer can create custom slots and fill them with examples that represent the slots values. A slot can be customized to be required or to need confirmation, which can be used to create a multi-turn conversation to either collect all required information or to confirm user input.

Alexa skills can be split into different categories, one possible division is by the interaction model. While there are custom interaction models, which the remote intervention skill falls into, there are smart home skills, flash briefing skills, video and music skills. A different categorization would be by the purpose of the skill. This skill would be categorized as an assistant skill, in contrast to other skills purpose being info and educational, entertainment like games and music, or companion skills [14].

2.2.3 Amazon Web Services - AWS Lambda

All of the above configurations to the interaction model can be done in the *Alexa Developer Console*. The interaction model is a Java Script Object Notation (JSON) file and with Alexas voice service technology, requests are created and sent to the specified endpoint to compose a useful response. Consequently, a web service is needed to form these responses. It will handle requests and look up information online or in a database when necessary. You can either create a web service using any cloud hosting provider yourself or use various on-demand cloud computing platforms from the **Amazon Web Services** (AWS). AWS offers different services ranging from computing, storage, networking, databases and many more. Specifically interesting in our context is AWS Lambda, with which you can create a function and run your code without having to manage a server. That means a serverless computing service, or *Function as a Service* (FaaS), which means an abstraction of the server away from the developer. The billing and allocating of resources in this case is based on the number of executions of a function and not on server instances like with *Amazon Elastic Compute Cloud* (EC2). The Lambda function is connected to the skill as an endpoint with an *Amazon Resource Name* (ARN). It is easily scalable and only executes code when it receives a request. The computing resources are automatically allocated in order to inspect the request, take all actions necessary to complete it and form an appropriate response. To use AWS Lambda, a region for the server location needs to be selected. There are available regions all over the world to host a serverless platform, the main one for Europe is located in Ireland.

More other useful services are the *Identity and Acess Management* (IAM), as well as storage services. More information about the IAM will be covered in the implementation section 5.4.1.

One of the used storage services is a database called *DynamoDB*. It is a NoSQL database built for high throughput and low latency. Custom key-value pairs can be stored without many restrictions to create a unique database structure suited for the skill's requirements. The three main components of the structure are tables, items and attributes. From the Lambda function, queries can be created and executed to retrieve or modify data formatted in JSON.

The other one is a cloud storage service to save images and other resources called AMAZON S3. Since it is better suited for storing larger files such as images or binary data, S3 is a simple web-based cloud storage service for backups, archiving and retrieving data.

2.2.4 High-level Architecture

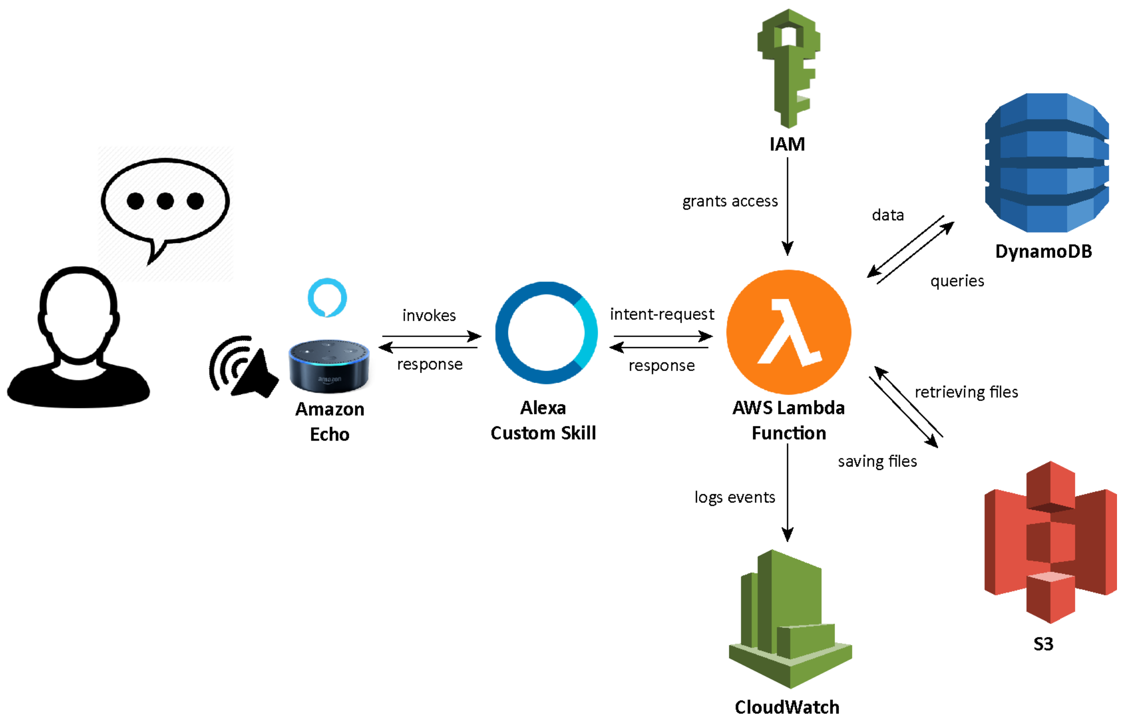


Figure 2.2: High-Level Architecture

The next step after presenting the basics about the different components of the skill is to inspect all communication between the different entities and to provide an architectural overview for the skill. Initially, the device hears the wake word and the recorded audio is sent to the Alexa skill via the Alexa Cloud. During this process the Alexa Cloud uses Natural Language Processing (NLP) to convert the audio containing spoken words to text in order for the skill to retrieve and recognize the Intent. The Intent is formed into an Intent-Request, which is sent to the AWS Lambda function to be processed. Optionally, the Lambda function interacts with the user via Alexa to retrieve additional information which may be required to fulfill the request. If the request needs to, the AWS Lambda interacts with the DynamoDB database accordingly. Items can be added, deleted, modified or retrieved. Typically, one intent just does one of these operations. Another service previously not mentioned, but useful for development is Amazon CloudWatch. Here, every interaction and decision are logged, it also provides general logging via the code for debugging and error messages.

If everything completed successfully, a structured response is generated by the Lambda function and the Alexa device reads it out loud back to the user.

2.2.5 Security

With voice assistant technology being relatively new, and especially with the introduction of Alexa for Business, there is a concern for security related vulnerabilities. Therefore it is crucial for security experts to understand how Alexa behaves concerning audio streaming and how to efficiently implement security measures to make sure important information is handled carefully.

The first matter regarding general personal data privacy, is the question if private conversations are recorded and forwarded to the Alexa Voice Service without explicitly articulating the wake word [15]. The problem here is based on the fact that many words in different languages can be mistaken for the wake word by the Alexa device. Consequently, sensitive conversations could be streamed by accident, which on its own is a delicate subject. Another possibility are external audio sources like a TV, which enunciate the wake word explicitly or a similar homophone [16].

In the same way, the authors of the study in [17] wanted to know whether or not the interactions are properly and entirely logged and accessible for inspection by the user.

Results show inconsistencies between the logging of response cards on the Android device compared to the corresponding Amazon Voice Service (AVS) traffic. Therefore, users can not have complete certainty that their communications are fully logged and they are not exposed to any additional activity conducted with their intelligent voice assistant device.

Finally, in the same experiment, they determined if the on/off button for the microphone of the Echo Dot device actually works and if private conversations or audio of any kind is still streamed to Amazon. Fortunately, the Alexa device in the experiment that had its microphone turned off from start to finish did not create enough network traffic to worry about leaked information. It solely sent out some keep-alive data, but no response cards were logged nor was any audio recorded.

Unfortunately, there is one potential vulnerability that may lead to a new form of attack, specifically on VPAs. When launching a skill, it is unavoidable to use its invocation name. However, these invocation names are not unique and can be chosen without many restrictions. For that reason, an attack called voice-squatting is possible and an attacker can try to impersonate a skill with sensitive information on the line. Homophone names or just similar ones can be used to mistakenly open the wrong skill and collect data. Additionally, people often use words like "my" or "please" [18] in their interaction which can also just be incorporated into the adversarial invocation name for a squatting attack.

3 Requirements

3.1 Scenario

In this chapter, the general scenario is discussed by considering the different actors involved and their various roles. Two usage scenarios are illustrated on the next page. These form the basis for the requirements engineering result. In the first scenario, illustrated in figure 3.1, the procedure starts with a personal meeting between the therapist and the patient. During this meeting, the therapist tries to get an overview of the patients' problems and needs. He then creates a therapy plan including exercises to do as homework. The homework instructions are provided and put into the database by the therapist or one of his employees. At home, the patient can retrieve these instructions with his voice assistant by using the remote interventions skill. Figure 3.1 describes a basic scenario in which the patient uses his Amazon Echo to get his homework instructions in speech form. Additionally, he can look up the instructions in text form on his mobile device. After having carried out his exercise he can mark it as completed by adding feedback to the exercise in the database.

The aforementioned feedback can be read and evaluated by the therapist on his workstation by looking at the database or exporting it into a different format to more efficiently make assessments and improve the development of the therapy. Note that the provided text in the speech bubbles does not represent realistic interactions between the patient and the digital voice assistant, how exactly these interactions pan out will be described in more detail in the following two chapters.

Similarly, in the second figure 3.2, a scenario is illustrated where the patient wishes to set a reminder to do his homework for later in the day at a specific time. He tells the

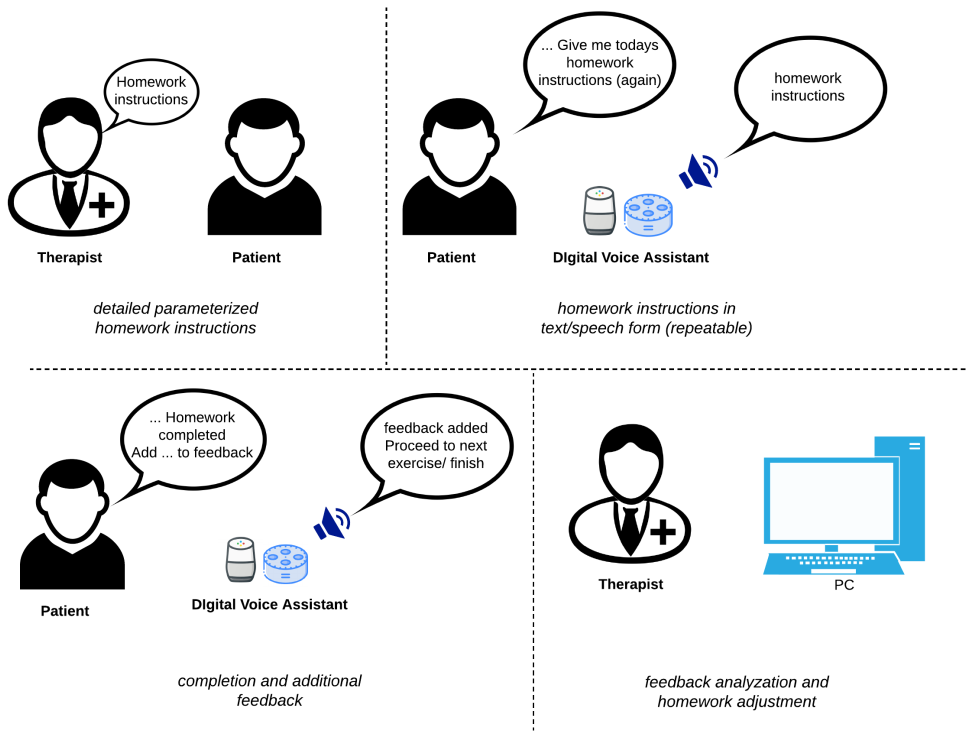

Homework instructions
Therapist
Patient
detailed parameterized homework instructions
... Give me todays homework instructions (again)
homework instructions
Patient
DIgital Voice Assistant
homework instructions in text/speech form (repeatable)
... Homework completed Add ... to feedback
feedback added Proceed to next exercise/ finish
Patient
DIgital Voice Assistant
completion and additional feedback
Therapist
PC
feedback analyzation and homework adjustment

Figure 3.1: General Skill Usage Scenario

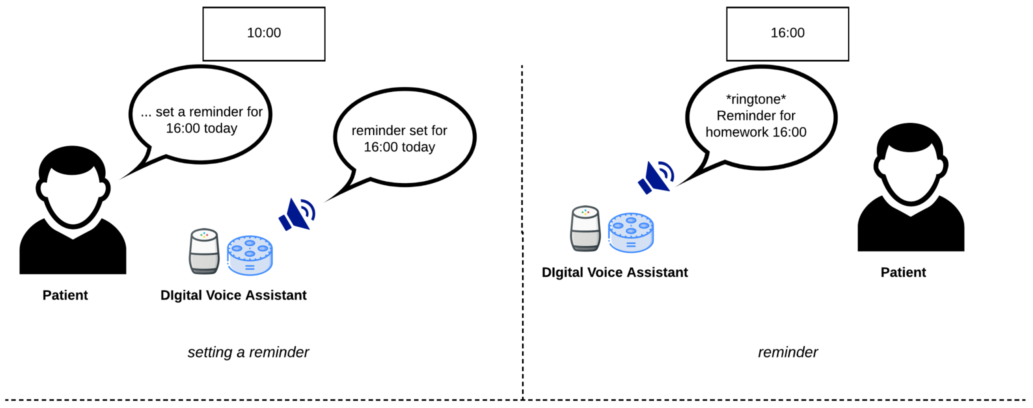

10:00
... set a reminder for 16:00 today
reminder set for 16:00 today
Patient
DIgital Voice Assistant
setting a reminder
16:00
ringtone Reminder for homework 16:00
DIgital Voice Assistant
Patient
reminder

Figure 3.2: Reminder Usage Scenario

skill he wants to set a reminder and provides the time of day at which he needs to be reminded. The skill will understand the intent and will set a reminder within the Alexa's system. The reminder will chime at the requested time and can also fire an alert on your mobile phone if you have the Alexa companion application installed and notifications enabled.

3.2 Functional Requirements

On top of that, the functional requirements for the skill regarding the specific roles are presented as well as a set of non-functional requirements with quality attributes and specifications. There are only two roles or actors for these requirements. The first one is the therapist, he does not interact with an Alexa enabled device, he just provides, collects and analyses data.

3.2.1 The Therapist

Table 3.1: Functional Requirements - Therapist

Functional Requirement	**Description**
Add New Exercise	The therapist is able to add new exercises in the database
Modify Exercise	Existing exercises can be modified
Remove Exercise	Existing exercises can be removed when no longer needed
Export database table	A table in the database can be exported to a .csv file
List of Users and Amazon Id	The therapist has a list of all his patients and corresponding User Ids to correctly assign exercises and feedback.

Requirements realized specifically for therapist usage are mostly just working with the DynamoDB database. The interface is structured intuitively, therefore not a lot of training or instruction is necessary.

3.2.2 The Patient

All actions available to the Patient are realized by interaction with an Alexa-enabled device.

Table 3.2: Functional Requirements - Patient

Functional Requirement	**Description**
Get exercise	If there are exercises available the user can ask for one and get it read out to him
Get exercise by name	Exercises can be retrieved by saying the name
Get exercise by index	Exercises can be retrieved by saying the number of its index
Give feedback custom slot	Add feedback from a list of intended words to a specific exercise
Give custom feedback	Add any feedback to a specific exercise
Add Reminder	The patient can add a reminder for a specific exercise
Remove Reminder	Previously added reminders can be removed again
Launch Skill	The skill can be opened and options for usage are presented
Help	The user can initiate the HelpIntent and get information on how to use the skill.
Instructions, Manual	There is a manual with examples and instructions to ensure comfortable usage.
Home Cards	After asking for something from the skill the user can re-read instructions in the Alexa smart phone app.
Alexa Show component	A visual representation of the response and optional additional information (pictures)

3.3 Non-Functional Requirements

In contrast to the functional requirements which describe specific behaviours, the purpose of non-functional requirements is to set boundaries and standards for the evaluation of the systems quality and performance aspects. The following table defines system attributes which ensure usability and effectiveness of the system.

Table 3.3: Non-Functional Requirements - General

Non-functional Requirements	**Description**
Confidentiality	Sensitive user data should be protected against unwarranted disclosure
Scalability	The skill and all components utilized should be scalable for as many users as necessary
Performance	Responses should be quick, so the user does not have to wait
Simplicity	The skill should be self-explanatory or at least easy to use with basic instructions
Primary Languages	The skill works both with the German and with the English language
Other Languages	Other languages are easily addable
Correctness	The skill should deliver the correct exercises and add feedback to the respective one

4 Concept and Design

Before turning the requirements into an implementation, a concept has to be designed. The core features of the skill will be described in more detail and how they can be accomplished using a voice assistant and its available capabilities. In addition, more insight on how to design for voice applications and how best practices differ from mobile and web development are discussed.

Creating a concept in this context means finding the right tools and features of the Alexa Skills (Kit) Software Development Kit (ASK SDK) to realize the technical parts of the skill, as well as thinking about how to optimally design the skill's interaction model considering the intents. However, ahead of doing that, an overview of what it means to build a voice assistant application is established to understand the decision making of the conceptualization.

4.1 Voice Design

This being the first time developing for a voice assistant, advantages and disadvantages are considered along with a survey on how building for voice differs from building for screen applications [19].

Elemental advantages of digital voice assistants include the simple fact you can interact with your device from across the room and do not have to break away from your current activity. Also known as multitasking, this concept allows users to perform an exercise and getting the instructions without using their hands. The usage of a smart mobile device can be distracting, but it may sometimes be used in conjunction with the voice application in order to provide additional information or just to reassure the user. A major disadvantage, which nowadays unfortunately not only applies to voice assistants but most technology, is the dependence on the internet and its cloud services. On top of that, while voice recognition and natural language understanding is getting impressively accurate, it still is not perfect and having to repeat the commands occasionally is unavoidable. Issues regarding security, as previously stated in 2.2.5, could deter sales in business environments as well. Expert opinions on this matter propose the creation of more specialized personal assistants with a limited number of tasks to improve in certain problem areas [20].

When looking at the differences between voice and mobile or web, it becomes apparent basic principles and the role of the application have to be reconsidered. First of all you have to be adaptable, in other words you have to let users speak in their own words. Common affordances are not applicable, an affordance in voice is more about what the user would normally say taking into account the fundamental intent he wants to fulfill. A big challenge is figuring out out what people might say in their specific context coupled with what people think they might be able to say. It is noteworthy, that with Alexa's speech recognition technology, explained in more detail in 4.2.1, very similar utterances can accomplish the same intent with enough training data for improvement using machine learning. When developing for web or mobile, consistency is a very important principle for a positive user experience. Even though this is also applicable for voice applications, variability should also be considered as a reasonable convention. The skill can be tailored over time, for example the first time the skill is opened, a different greeting can be used with an introduction, or memory can be used to improve and simplify future interactions.

Normally, when designing a user interface on a website for example, developers think a lot about menus and steps to perform to get certain information. In comparison, voice user interfaces should be really wide at the top level, so that specific information is available and accessible without having to perform multiple or too specific interactions.

This leads us to two different design patterns, visualized in figure 4.1. With a graph-based approach, decisions are made which then branch off into another path working their way

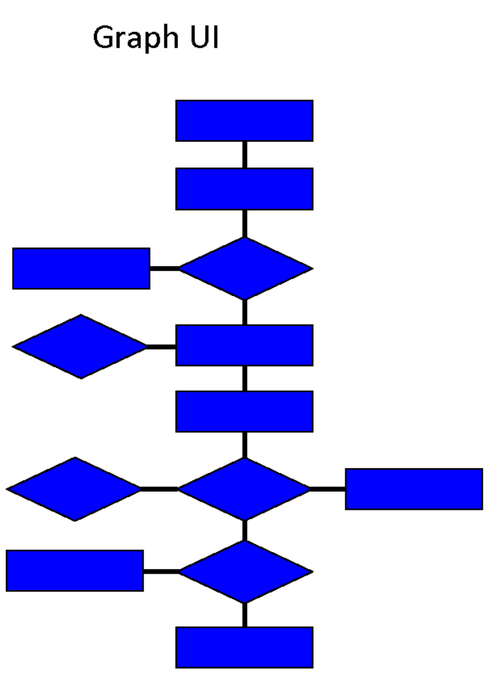


Figure 4.1: Graph vs. Frame design pattern

down towards the goal. Although this works great in mobile or web, it is different for voice as it is too much for the user to remember and too much for the skill to keep track of. For this reason, a frame-based approach is introduced, where the user is guided to the point where information must be collected. The user is prompted for the required information until completed and is given a response afterwards. The frame-based approach can be realized using multi-turn dialogs to elicit information. A multi-turn dialog is the opposite of a one-shot-interaction, where for example a voice assistant is asked to turn on the lights or wants to know the weather for a specific city. Usually, to create an appropriate response for a specific request from the user, information to fill the input slots has to be gathered. Consequently, the conversation is related to a specific intent with specific slots and questions are asked by the device to complete those slots [21].

Most skills or basic features Alexa has preinstalled just get some information and bring it back. Additional interactions would be unnecessary and maybe even annoying because the user just wants information quickly (Urgency). More complex skills can play specific roles though [22], like being a guide, mentor or coach (Guidance). With that in mind, certain embellishments can be added to make the experience more enjoyable. Examples could be a welcome message, or just guidance along the way of collecting information to let the user know what is happening.

4.2 Concept

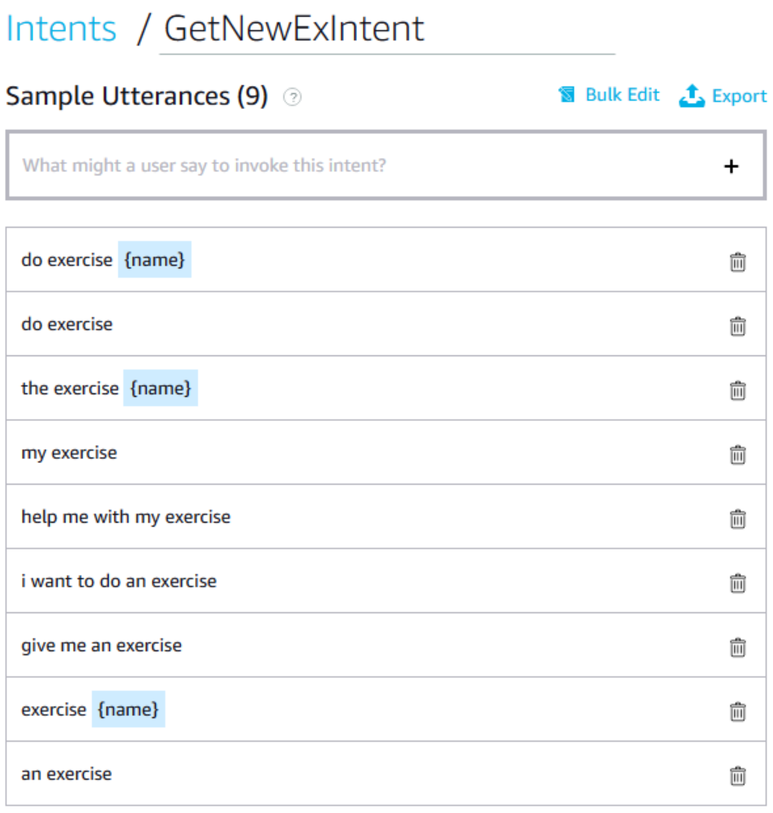
After finding and analysing best practices for designing voice applications, the core features of the skill will be explained more thoroughly. The theoretical requirements are translated into a more detailed specification. In this case, the interaction model and intent schema are outlined and the interactions themselves are planned out.

4.2.1 Speech Recognition

First, let us have a more detailed look at how Alexa's technology converts spoken words into text a computer can understand and work with. All of this happens within seconds yet consists of a series of rather complex operations. Whenever audio is identified, signal processing enhances the target signal to optimally recognize the words by minimizing ambient noises. As soon as the wake word is detected, the audio signal is sent to the cloud for speech recognition software to do its job. A big part of the technology developed by Amazon to deduce intent from human language is called **Natural Language Understanding** (NLU) [23]. Its responsibility is to transform text into a meaningful representation by utilizing artificial intelligence for pattern recognition and decision making with statistical models. The main challenges here are improving the software around recognizing emotion and the problem of classifying the intents.

4.2.2 Retrieving Exercises

An exercise consists of a name and an instruction text. Naturally, a user can communicate various utterances to the device he wants to do an exercise with, including the word *exercise*, examples are shown in figure 4.2. Optionally, the name of the exercise can already be included in the first sentence the user speaks to the device, which would result in a one-shot interaction, where the user directly receives a response. Otherwise, because the slot for the name of the exercise in the *GetNewExIntent* is marked as required to fulfill the intent, Alexa will prompt the user for the name. The user can respond by only saying the name of the exercise or including it in a sentence. How this kind of multi-turn conversation would look and how the components interact with each other can be seen in figure 4.3.


Intents / GetNewExIntent

Sample Utterances (9)

do exercise {name}
do exercise
the exercise {name}
my exercise
help me with my exercise
i want to do an exercise
give me an exercise
exercise {name}
an exercise

Figure 4.2: Sample Utterances for retrieving exercises

4.2.3 Saving Feedback

Feedback is defined by two distinct slots, one of which is a custom slot and the other one is a slot where any phrase can be saved. The custom slot consists of simple words or expressions describing the difficulty level or other information for the therapist to evaluate. If the expression used by the patient is not included in the custom slot list, the other slot gathers the words and saves it as text. The type for unrestricted and less-predictable text is called *Amazon.SearchQuery*.

4.2.4 Setting a Reminder

Reminders are not part of the skill itself, but rather the Alexa reminder API is used to set a reminder in the Alexa system, similarly to setting it without using the remote interventions skill. To set a reminder, the patient can ask the skill to set a reminder and will get prompted for the time of day he wants to set the reminder for. Alternatively, the time can be included in the initial request. Reminders can also be set for an upcoming day and recurrence can be customized as well, not from within the remote interventions skill though.

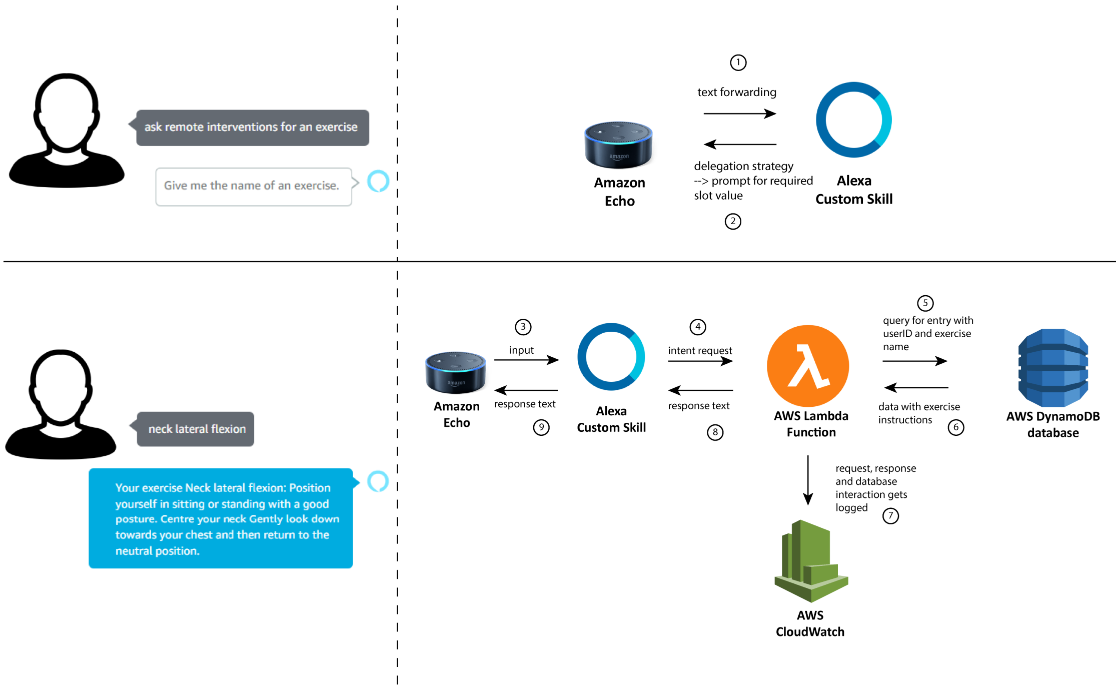

ask remote interventions for an exercise
Give me the name of an exercise.
1
text forwarding
delegation strategy --> prompt for required slot value
2
Amazon Echo
Alexa Custom Skill
neck lateral flexion
Your exercise Neck lateral flexion: Position yourself in sitting or standing with a good posture. Centre your neck Gently look down towards your chest and then return to the neutral position.
3
input
response text
9
Amazon Echo
Alexa Custom Skill
4
intent request
response text
8
AWS Lambda Function
5
query for entry with userID and exercise name
data with exercise instructions
6
AWS DynamoDB database
request, response and database interaction gets logged
7
AWS CloudWatch

Figure 4.3: Retrieving Exercise Component Interaction

5 Implementation

This chapter explains the implementation details and showcases how the previously mentioned services from Amazon were used in order to create an Alexa skill with the functionalities necessary to fulfill the requirements defined in section 3.2. The development was mostly done remotely using the **Alexa Skills Kit Command Line Interface** explained in more detail in 5.3.

5.1 Tools and Accounts

Visual Studio Code Visual Studio Code is the Integrated Development Environment (IDE) used to implement the Lambda function for this skill. VS Code is an easily customizable source code editor by Microsoft, and it is the recommended and probably most used editor for Alexa skill development [24].

JSON Java Script Object Notation is the data format used for defining the Interaction Model for the different languages. In addition, communication between the Lambda function and the skill uses JSON as a format for the request and response bodies.

Accounts There are three accounts needed to develop for Alexa devices. First off all, you need a basic or prime **Amazon Account**, which is the account used on the Alexa enabled device. The second account required for development is an **Alexa Developer Account**. It is directly linked to the first account and grants access to several Amazon development kits including the Alexa developer console which is essential for Alexa skill development. The third and last account is the **AWS Free Tier Account** which is linked to the second account and allows you to use various services from AWS for free within usage and data limitations. However, your credit card information must be provided for possible payments. These include for example one million free Lambda requests per month and 25 GB of DynamoDB storage. With automatic scaling provided by those services, additional charges and tier upgrades are available if the application is published and used by a lot of people. For this skill and its proof of concept purpose the free tier will suffice.

Amazon Echo Dot Device The physical device used for testing and experimenting is an Amazon Echo Dot (first generation), for picture reference, see 2.2. For development purposes a physical device is not necessary except when working with reminders, which only work with Alexa-enabled devices excluding Fire TV Sticks.

Alexa Companion App The Alexa Companion App is a smart phone application for android and iOS. It is used for managing Alexa settings like the initial setup, remote control and installing new skills.

Node.js Node.js is an asynchronous event-driven JavaScript runtime built on Chrome's V8 JavaScript engine [25]. Together with Python, JavaScript is one of the two main programming languages used to build Alexa skills. Numerous templates can be used to create simple skills without too much background in the programming language. Node.js is a runtime environment for JavaScript code, executed outside of a browser. For this implementation, the chosen version is *Node.js 8.10*, which is the latest *Long Term Supported* (LTS) version for Node.js and Lambda.

5.2 Alexa Developer Console - Interaction Model

The Alexa developer console is a modern, i.e. sleek, streamlined web-based tool from Amazon to create, manage and publish self-made Alexa skills. The first part of implementing the skill is choosing the interaction model and creating an intent schema in the Alexa developer console. The type of interaction model for our skill is called a **Custom Interaction Model**. Using this kind of model, new intents can be added and customized in the console. Customization includes sample utterances and the creation of slots. This establishes the basis for the skill and defines an outline of functionalities for further handling by the Lambda function which is accessible using the endpoint.

To have users interact with this particular custom skill it needs an invocation name. Originally, the word *therapy* was chosen before discovering several requirements and restrictions the invocation name has to meet. One of the restrictions being no one word invocation names. Thus, the new invocation name is *remote interventions*. The invocation name can be changed during development until the skill is submitted for certification and published.

Fetching exercise instructions as described in 4.2.2 is implemented in the intent *Get-NewExIntent*. Sample utterances for this intent are, amongst other things, "give me an exercise", "i want to do an exercise", or for the simplest usage with the slot already included in the utterance "exercise {name}". There is only one slot included in the intent called *name*. It is the only required value to fulfill the intent and serves as part of the key for the database to retrieve the exercise instructions. Since there are too many distinct exercises to save them all as values for a custom slot, the type of this slot is *AMAZON.SearchQuery*, where any exercise can be named and retrieved. Due to the slot being required, more expressions have to be defined, which allow Alexa to ask for the required information. One of them is the prompt for collecting the name of the exercise and the others are user utterances which respond to the prompt and contain the value of the slot. The exact phrases can be seen in the screenshot B.1.

There is another intent for retrieving an exercise called *GetNewExNumberIntent*, even if the user does not know the exact name of the exercise. He can get the exercise instructions using the number of the index for the exercise. When using the skill in the English language the user may also use the ordinal, e.g. first, second and so on. Unfortunately, ordinals are not supported for other languages.

Usually after completing the exercise, the user can add feedback for evaluation by the therapist. The user may also add feedback at a later time. The *GiveFeedbackIntent* contains three slots, the first being the name of the exercise, similar to the slot in the previous intent. The name of the exercise has to be collected with an additional prompt after the dialog has been initiated, since only one slot of the type *AMAZON.SearchQuery* can be included in the initial phrasing. Because they have fewer constraints on format and content, it would be impossible to distinguish different phrase slots within the same prompt. The remaining two slots are for saving feedback. Any phrase can be added as feedback using the *feedbackText* slot, which is also of the type *AMAZON.SearchQuery*. The third slot for saving feedback is called *feedback*, for which a **custom slot type** called *possibleFeedback* has been created. To assemble a custom slot like this, a list of reasonable words or short expressions to describe the outcome of an exercise has to be established. This list can be expanded at any time so more possible expressions can be added to make it more practical. A subset of expressions contained in the slot are shown in B.2. In consideration of these slots, sample utterances for the intent have to be specified. To ensure simplicity and correct functionality of this intent, utterances are kept quite simple. Users can either say "*feedback*" and then their feedback using expressions from the custom slot or any other wording. The slot with the custom slot type is also required to fulfill the intent, guaranteeing feedback will always be saved. Thus, a speech prompt and corresponding user utterances were defined requesting feedback from the list of slot values, see screenshot B.3.

A few more intents were implemented, however most of them were mainly used for testing and development purposes. Therefore, implementation details for only one more intent will be examined more closely in the following.

For the purpose of setting a reminder, the *ReminderIntent* consists of various sample utterances. Some of which already contain a slot for the time at which the reminder is supposed to be scheduled. If utterances not already including the time are voiced, the user will be prompted for the time of day and ideally the response would be a valid time, see screenshots B.4 and B.5.

Setting reminders for upcoming days or creating reminders that are recurring is not included in the custom skill implementation. Customizing reminder utterances for upcoming dates would be possible, however there is no slot type for setting the frequency of recurrence. Consequently, if the user wants to add repeating reminders, he can use the core reminder functionalities of Alexa itself. This works more consistently, and the created reminders are saved identically to the reminders from our custom skill. For all of the previously mentioned intents which include required slots, the dialog delegation strategy is set to *fallback to skill setting*. This allows dialog and elicitation to happen as intended because *auto delegation* is enabled in the developer consoles interface section.

Every functionality is implemented in English and German. In the above descriptions of the intent schema and the attached screenshots, sample utterances are showcased in English only. For the german skill, the sample utterances are mostly just the corresponding translation or sometimes small deviations for more natural language understanding. The intent schema is saved in multiple JSON files, one for each language the skill is available in. For the german language this file is called *de-DE.json* and for the english language *en-US.json*.

There are a few more functionalities and specifications worth mentioning included in the Alexa developer console. First of all, to test utterance behaviour and to find the best sample utterances and slots for our intents, there is a useful tool called the **Utterance Profiler**. Trial interactions can be performed and the resulting intent with resolved slots are displayed. For a usage example, see figure B.6.

Two more configurations are necessary for correct functionality. To connect the skill with the interaction model, a service endpoint is specified. As previously stated, we are working with AWS Lambda, so the Amazon Resource Name (ARNs) for the specific Lambda function is provided in the endpoint input box. Furthermore, permissions required by the skill can be enabled in the permissions section of the console. Our skill specifically requires the permission to save and edit self-created reminders in the Alexa system. The Alexa developer console also allows for actual testing of the skill, this will be elaborated in section 5.5.

5.3 Remote Development and Prerequisites

Prior to implementing the AWS Lambda function, a few prerequisites for using the **Alexa Skills Kit Command Line Interface (ASK CLI)** have to be met. After installing *Node.js*, Git and the *Node Package Manager (npm)*, as an administrator in the Windows

PowerShell, npm can be used to first install the windows-build-tools manager and subsequently to install the ASK-CLI. When initializing the command line interface, using the *ask init* command, you can login to your Amazon developer account and connect the AWS account by creating an ASK profile and specifying credentials created by *AWS Identity and Access Management*. More on the IAM configuration in 5.4.1.

As previously mentioned, **Visual Studio Code** was used for the development of this skill. The IDE has the option to open an integrated terminal or shell command prompt. Following the initialization, multiple ASK CLI commands are available to create and manage the skill using the command line. With the *ask new* command new skills can be created from a template. After executing the command, a runtime environment has to be chosen, the two options are node.js and python. Afterwards there are over ten different options for templates to chose from. For this skill the Amazon-provided fact skill template was used.

To deploy the skill to the developer account as well as the code to the AWS Lambda function, the *ask deploy* command is used. If the eTag matches the one on the server, the skills manifest, all models for each language and the source code will be updated or created if it does not exist yet. Should this deployment fail, or whenever you want to inspect the differences between the local and the remote version, *ask diff* can be used and will list all differences between the different targets.

When using a different machine or just to download the newest version of the deployed skill, *ask clone* is the command that will create a local skill project directory and download all the necessary parts of the skill. Afterwards local changes can be made, and the project can be deployed again using the above command. This basically describes the everyday deployment process during development.

5.4 Amazon Web Services

Amazon Web Services (AWS) is a cloud platform offering over 165 featured services from data centers all over the world. These services provide distributed computing, auto scaling and you can use it on-demand for free if you stay within the usage limits.

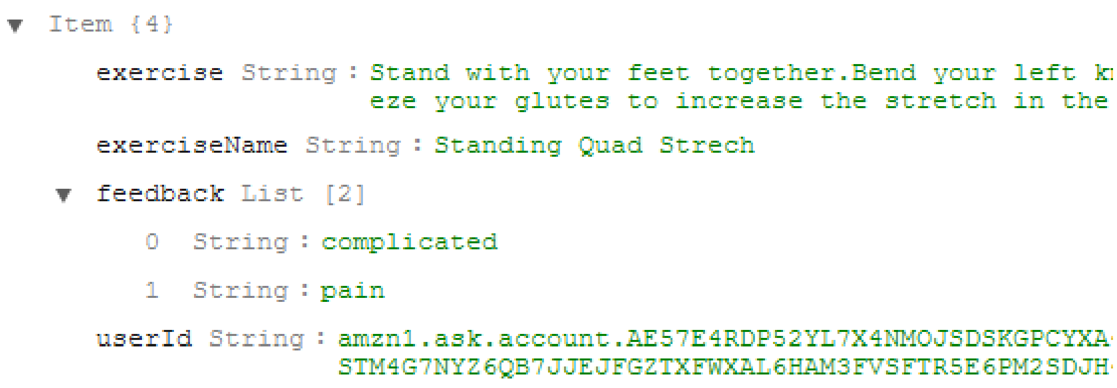
For the purposes of this thesis, the most used service is the AWS Lambda which was used to implement the endpoint for the skill. Additionally, the services IAM for security, Cloudwatch for logging, DynamoDB as a database and S3 for storage have been used over the course of this project.

5.4.1 Identity and Access Management (IAM)

Before being able to code the Lambda function and using the DynamoDB database, five steps have to be completed to earn security status. At first, root access keys must be deleted because they provide unrestricted access to the AWS resources. Instead, individual users should be created and assigned security credentials. Since this is a solo project there are two users, one for the stationary and one for the mobile workstation. Generally, permissions are assigned to groups, therefore an admin group was created and both users were assigned to that group. Two permissions were assigned to the admin group to allow database access and function execution. The names of these permissions are *AmazonDynamoDBFullAccess* and *AWSLambdaBasicExecutionRole*. Additionally, multi-factor authentication must be enabled to add another layer of protection. And finally, a password policy has to be used to require users to create strong passwords and rotate them regularly.

5.4.2 DynamoDB Database

The used database only consists of one table, which is called *dynamo-exercises*. When creating a table in DynamoDB, the primary key has to be specified. The primary key here is actually a *composite primary key*, which means it is composed of two attributes. The first one is the user id linked to the Amazon account, it is the partition key for any item, also known as its hash attribute. It ensures separation of different users and grouping items of the same user. The other one, the name of the exercise, serves as the primary sort key. Therefore, exercises from the same user are grouped together and sorted alphabetically by the name of the exercise. A table consists of multiple items and every item is a collection of attributes. Besides the two primary keys, these include the instruction text for the exercise as well as a list of all feedback given to the specific exercise by a specific user. The types of all attributes as well as the tree representation in the web interface can be observed in figure 5.1.



```
▼ Item {4}
    exercise String : Stand with your feet together.Bend your left k
                      eze your glutes to increase the stretch in the
    exerciseName String : Standing Quad Strech
  ▼ feedback List [2]
      0  String : complicated
      1  String : pain
    userId String : amzn1.ask.account.AE57E4RDP52YL7X4NMOJSDSKGPCYXA
                    STM4G7NYZ6QB7JJEJFGZTXFWXAL6HAM3FVSFTR5E6PM2SDJH
```

Figure 5.1: Database Item Screenshot

5.4.3 Lambda Function

As previously mentioned in the remote development section 5.3, the *ask new* command creates the default API endpoint, which in this case is the Lambda function and translates into a single JavaScript file with the name *index.js*. In Node.js 8.10, the used LTS version of the runtime, asynchronous operations are handled with so-called **Promises**. They *promise* the eventual completion or failure of an asynchronous block of code. To prevent blocking by I/O functions, in our case database calls, **Callbacks** are used. A Callback is a function that is called at the completion of a given task and they form the basis of Node.js. Another important concept are **async** and **await** functions. Attached to a function declaration, async tells the Node interpreter that the function is asynchronous and await can be used inside such a function to wait for a resolve or reject which are the possible results of a promise.

A blueprint is provided by the *skill-sample-nodejs-fact* skill template [26]. It is a template skill providing facts about space when invoked. The code can be divided into four sections, which are constants, intent handlers, data and helper functions. In Node.js, loading modules is done by defining constants using require functions. The modules required for the skill are:

- **ask-sdk-core** This is the Alexa Skills Development Kit. It contains the basic components and default implementations of *ASK SDK v2* and helps with boiler-plate code.
- **request-promise** Powered by Bluebird, this is the http request client with Promise support, it is only needed for the parts containing Promises.
- **i18next and sprintf-postprocessor** I18next is an internationalization framework, it aids with translation and string formatting.
- **dbHelper** DBHelper is a self-implemented module for all database interactions and helper functions.
- **ask-sdk-dynamodb-persistence-adapter** Add-on package for *ASK SDK v2* to implement a persistence-adapter for the custom *Skill Builder*.

There are many fixed phrases in both languages, saved in two data constants. They define what the device responds in every case, depending on whether it is working as intended or if an error of some sort occurred. Captured in the screenshot B.8, all of the English phrases are shown.

For each intent set out in the intent schema, there is a linked IntentHandler implemented in the Lambda. Included in the template, there are several boiler-plate functions to implement the auxiliary intents. This includes a *LaunchHandler*, which fires when the skill is invoked without more information, hence if the user says something like "launch" or "open" with the invocation name. In this case, a welcome and a help message with instructions are played. Furthermore, there are handlers for errors and different scenarios where things are not working out as intended.

For general errors, the *ErrorHandler* logs the origin of the error and any additional information to the AWS CloudWatch. A general error message is uttered, and the user is encouraged to try again.

An interaction can be cancelled and the user can completely exit the skill by saying something like "never mind" or "stop", this is represented as a *AMAZON.CancelIntent* or *AMAZON.StopIntent*. Both intents are intercepted by the *ExitHandler* which makes the device say "Goodbye!". In this case or if Alexa is waiting for a response and the user does not respond, a *SessionEndedRequest* is made to the skill. No response can be made to this request, it simply logs the reason to CloudWatch.

In the event that the user asks for help, a help intent invokes the *HelpIntentHandler* method and information on how to use the skill is provided.

User utterances not matching any of the skills custom intents default to the *AMAZON.FallbackIntent*. Like the previous intent, instructional information on how to use the skill correctly is offered to the user. Unfortunately, this is only supported for English locales.

After looking at how the built-in intents are implemented, next up are four custom intents that actually implement the core functionalities of the skill. All of the following intents have slots required to fulfill the intent, therefore prior to the intent handler function getting called, if the dialog state is not completed, an *InProgress* function is called witch delegates the dialog back to Alexa by returning the *Dialog.Delegate* directive. Alexa then checks the dialog model for the next step and uses the predefined prompt to ask the user for more information.

For the *GetNewExHandler*, the purpose is to collect exercise instructions using the name of the exercise. First, the *userID* and the exercise name are saved to variables, they will serve as primary and secondary keys for the database. Afterwards, utilizing the database helper module and a promise, a query is created and executed. Then, the speech text is formed, consisting of a basic prelude followed by the exercise instructions provided by the therapist and retrieved from the database. Additionally, a home card is created, which is displayed in the *Alexa Companion* smart phone application. Within this home card, the instruction text is displayed which enhances the voice interaction.

Similarly, in the *GetNewExNumberHandler*, the output created is the same. However, in this case, the key for retrieving the exercise instructions is not the name of the exercise but the index or ordinal. This is done by retrieving every exercise from the specific user with the Amazon user id into a list and concatenating the corresponding instructions to the speech text variable.

Giving feedback invokes the *GiveFeedbackHandler*. Here, the user id and exercise name are needed for the database call, the latter of which is collected by the dialog directive. This time though, data is saved into the database for the therapist to review. Feedback has two different slot types, one of them being able to save any phrase called *feedbackText* and the other one being from a predetermined set of expressions called *feedback*. The feedback slot is required and therefore directly saved to an array at the start. Afterwards, if the other slot has been used aswell, its value is pushed onto the same array. In the database call, this array is appended to a list of every feedback added to this exercise using an update expression for DynamoDB.

The last handler further described is the *ReminderHandler*, which uses a time input from the user to create a reminder in the alerts section of the Alexa device. The timezone is taken from the device and the time has to be formatted to ISO-8601 format for proper setup. The request consists of a header containing an authorization bearer token that has to be requested previously from the system api. An example for the body of a request for an absolute scheduled reminder can be found in the code snippet A.1.

For the reason that we have to access permissions to create reminders, the Alexa Service API needs to be called. A method for this connection does not exist within the standard *Skill Builder* and instead the custom skill builder had to be implemented. This is also the reason the persistence adapter is required, the methods for database connection are different between the standard and the custom skill builder. When using the custom skill builder, the persistence adapter is the only way to tell the index file what the database table is called in the DynamoDB.

In the case the user's response is not understood or the user does not respond while the *shouldSessionEnd* property is false, Alexa prompts the user with a reprompt text provided in every intent handler.

For each intent, code for *Alexa Show* devices is implemented. They have a screen on which the spoken text can be displayed by default. Additionally, the layout can be customized to show more information like an instruction picture or different text. For this skill specifically, a layout has been created and the instruction text will show on screen when installed on a *Alexa Show* enabled device.

5.5 Testing and Debugging

Testing the skill can be done in several different ways. Generally, it can be performed by voice or by text input. At first, for testing the utterances and the collection of slot information, the previously mentioned utterance profiler in the Alexa developer console is used.

Naturally, with any Alexa enabled device or with the companion application, testing can be done in a realistic setting by using the skill the same as a patient would.

Furthermore, also in the Alexa developer console, there is a testing tab which includes an **Alexa Simulator**. Realistic interactions can be performed with either voice or text input and text and voice responses are provided. In addition, only with text of course, the same thing can be done remotely in the terminal using the ask cli command "ask simulate". Another very useful thing offered by the developer console while simulating is the ability to access the request and response body of the communication in JSON format. An example of the respective request and response bodies from a basic dialog can be seen in A.2 and A.3.

With the JSON request, which the Lambda function receives to form the response, different test cases can be created in the AWS Lambda portal. For every intent and its variations, up to a total of ten test events can be configured if necessary. This enables convenient testing of functionalities by simply choosing the desired test event and clicking the "Test" button. Whether or not the test event completed successfully, the result returned by the function execution is displayed together with a summary of the event completion including the duration. Additionally, there is a section displaying the log output saved in the CloudWatch as one line in the log group. Specifically for testing and debugging this log output combined with the ease of executing test events is an essential part of developing an Alexa skill. It aided greatly in fixing many bugs and errors during the development process. A screenshot of such a test event execution and its result is shown in B.7. In the beginning of the development process, at a point in time when no valid JSON request can be copied from the developer console testing simulator, sample test events supplied by Amazon can be used.

5.6 Publishing and Usage

Getting a skill submitted for certification and subsequently getting it published demands a list of requirements. The submission checklist includes a list of policy guidelines helping to ensure the skill is appropriate for all Alexa customers. Fortunately, because the skill is hosted as a Lambda function in contrast to a custom web service, the security requirements list is a lot shorter. Security requirements in this case mostly guarantee customer data protection. Additionally, all required functional, voice interface and user experience tests must be performed successfully. For these tests, the Alexa developer console includes a Certification tab with validation, functional test and submission tools.

After passing all the tests and meeting all the requirements, the skill can be submitted. After it has been certified, an email will arrive with an estimate for when it will be available for end users. During this process, the publication status will change from "in Development" to "in Certification" to "live".

The remote intervention skill is not meant to be discovered by any user with an Alexa enabled device, rather it is meant to be recommended for additional usage when currently in therapy. An Alexa skill can also just be published privately, this way the skill does not have to go through the certification process. The skill can still be used normally, it is just not discoverable in the public Alexa skills Store.

Nevertheless, when publishing, you are asked to provide example phrases which are displayed in the "Try saying" section. These are included in the publishing information and should highlight basic functionalities and serve as a guide for people to let them know how to use the skill correctly. It is important that these example phrases follow a specific structure. For one thing, they have to contain the wake word "Alexa", as well as the skill's invocation name "remote interventions". Moreover, all examples must match the sample utterances exactly. For this skill, the example phrases read as follows.

- "Alexa, ask remote interventions to give me an exercise"
- "Alexa, tell remote interventions feedback good, hurt"
- "Alexa, ask remote interventions to set a reminder"

Furthermore, the publishing information consists of the invocation name, a short description which Alexa reads out when asking for help and a longer more detailed description for reading. Category key words can be added as well to make the skill more discoverable.

After all these requirements have been met and the skill is published to the Alexa Skills store it is available for installation on the user's device.

5.7 Requirements Comparison

In the requirements in chapter 3, the functional and non-functional requirements for the skill are defined. Correspondingly, in this section, there is an analysis to what extent these requirements have been met after the implementation. All requirements for the therapist in 3.2.1 are not in direct connection to the Alexa skill, the therapist simply interacts with the collected data stored in the database table in DynamoDB. Therefore all of his requirements are fulfilled by using the functionalities of the AWS service to store the data remotely.

Regarding the patients and the non-functional requirements, the following tables will show to what extend they have been fulfilled. First, for the functional requirements, the features are listed and their fulfilment status is named and explained. Afterwards, the list of non-functional requirements is compared to the properties of the implemented skill.

Table 5.1: Functional Requirements Comparison

Requirement	**Status and Explanation**
Get exercise	Fulfilled, using one of the implemented intents. Any exercise from the database can be retrieved and read out
Get exercise by name	Fulfilled with the GetExNameIntent
Get exercise by index	Fulfilled, exercises can be retrieved by index or ordinal (English only)
Give feedback custom slot	Fulfilled, using the *possibleFeedback* custom slot type
Give custom feedback	Fulfilled, in addition to the above feedback, any phrase can be added as feedback to any exercise
Add Reminder	Fulfilled partially, reminders for later in a day can be added. Following the best practises for developing Alexa voice applications, the official reminder api is used to add the reminders to the device.
Remove Reminder	Not fulfilled, can be done using Alexa voice commands or the Companion application
Launch Skill	Fulfilled, LaunchIntent is included automatically and the necessary functionality was implemented
Help	Fulfilled, the user can ask for help or look up instructions
Home Cards	Fulfilled, every Alexa response can be seen in the Companion application as a home card
Alexa Show component	Fulfilled partially, instruction text is visualized in a customized layout. Pictures are not included, but Amazon S3 is set up, however it was not used

Table 5.2: Non-functional Requirements Comparison

Requirements	**Status and Explanation**
Confidentiality	Fulfilled, user data in the DynamoDB is only accessible by the therapist. Security measures are implemented
Scalability	Fulfilled, scalability is provided by AWS
Performance	Fulfilled, with a good enough internet connection responses are almost instant
Simplicity	Fulfilled, utterances are kept as simple as possible
Primary Languages	Fulfilled, all utterances and responses are implemented in English and German
Other Languages	Not fulfilled, localization is implemented, other languages could be easily added
Correctness	Fulfilled, if exercise names are unique, the correct exercise is given. The therapist is responsible for data integrity

6 Summary and Outlook

The goal of this thesis was to implement an Alexa skill as a complement to the *Albatros* smart phone application and reviewing its feasibility and verifying its practical potential. The skill is designed to further aid in reducing effort and complexity of therapeutic homework execution and memorization.

To achieve this goal, first various essential concepts and technologies had to be researched and analysed. I had to get accustomed to a new programming language and a whole new way of thinking about top level user interfaces. Upon adjusting to the unfamiliar environment and finding the right tools to realize the skill, the development went smoothly with a few challenges regarding permissions and the conceptualization of the intents and the structure of the utterances.

One of the goals was to make the usage of the skill as intuitive as possible in light of the fact that a majority of people in need of physical therapy are elderly and may be a bit more out of touch with modern technology in general. On the other hand, it may be easier for them to talk to a device with a well defined set of phrases than to operate a smart mobile device with endless capabilities [27]. This is supported by various publications [28, 29, 30] studying voice assistant interactions in contrast to small screens and keyboards.

In my experience, summarizing the name and the general outline of a physical therapy exercise is feasible compared to remembering the exact and most of the time critical details for every part of the body and posture. Therefore, an auditory reminder of the step by step instructions with supplementary visual and textual assistance can be really beneficial to remote therapeutic interventions. The purpose of an assistant is to help you achieve a goal, and in this regard the skill can definitely be advantageous. Similar to yoga instructions, where in many cases you do not see the instructor or teacher, audio instructions should mostly be sufficient, especially if you have seen and done the exercise before, which happens in the meetings when introducing homework to the patient. If the user has an Alexa Show device in their possession, the screen can show the instructions in textual form and maybe even have a picture or an animation with a visual representation of the exercise. If comprehensible instructions can be provided by the therapist, the skill can support the patient in order to successfully perform their homework exercises. Additionally, the therapist can make adjustments considering the feedback, change the instruction text and maybe even include messages at the start on how to perform a homework exercise more effectively bearing in mind what problems the patient had while executing it.

However, voice assistant technology is still relatively new and not yet sufficiently sophisticated. Besides that, most users are not very familiar with the technology and how to use it correctly. Alexa devices work differently than a voice assistant would on a smart mobile device like Siri or the Google voice assistant, they all have their unique ways of detecting language, handling natural language conversations and invoking skills. If a person is used to another kind of assistant, it could take them a while to adapt to a new device. At the same time, the importance of valuable and clear instructions is increased to ensure proper usability.

In the future, similar applications for other voice assistants, especially for the *Google Home* voice assistant devices could be developed to expand availability and test out functionalities on other systems. Furthermore, voice assistants will soon be able to handle more than one related request, which means the wake word does not have to be repeated for every command and usability and natural language will be improved by a rather large margin.

Personally, I learned a lot about designing voice interfaces and got to familiarize myself with various different technologies like the programming language JavaScript, the server-sided development platform Node.js and the possibilities of the Amazon web services. Both Node.js and AWS Lambda are very capable and effective tools to create web services. I understood the potential of Node.js and Lambda as event-driven platforms to make creating fast, serverless and scalable platforms as easy as possible.

Voice assistant technology also has a lot to offer, but beyond *Alexa for Business*,I still see the main potential in smart home automation. Turning lights on or off, controlling the temperature of a home or playing music on command works very well and has great benefits compared to doing it manually. If additions like the remote intervention skill can expand the possibilities beyond making the home more convenient into the area of medical therapies, then the concept of guiding or assisting skills for voice applications like **remote interventions** prove to be a valuable addition to the area of smart applications.

Bibliography

[1] JuniperResearch: Digital voice assistants by 2023. https://www.businesswire.com/news/home/20190212005064/en (2019) [Online; Accessed: 01-09-2019].

[2] Lopatovska, I., Williams, H.: Personification of the amazon alexa: Bff or a mindless companion. In: Proceedings of the 2018 Conference on Human Information Interaction & Retrieval. CHIIR '18, New York, NY, USA, ACM (2018) 265–268

[3] Sterling, G.: Alexa devices maintain 70 https://marketingland.com/alexa-devices-maintain-70-market-share-in-u-s-according-to-survey-265180 (2019) [Online; Accessed: 01-09-2019].

[4] Seymour, W.: How loyal is your alexa?: Imagining a respectful smart assistant. In: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems. CHI EA '18, New York, NY, USA, ACM (2018) SRC20:1–SRC20:6

[5] Burbach, L., Halbach, P., Plettenberg, N., Nakayama, J., Ziefle, M., Calero Valdez, A.: "hey, siri", "ok, google", "alexa". acceptance-relevant factors of virtual voice-assistants. In: 2019 IEEE International Professional Communication Conference (ProComm). (2019) 101–111

[6] Winkler, R., Söllner, M., Neuweiler, M.L., Conti Rossini, F., Leimeister, J.M.: Alexa, can you help us solve this problem?: How conversations with smart personal assistant tutors increase task group outcomes. In: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems. CHI EA '19, New York, NY, USA, ACM (2019) LBW2311:1–LBW2311:6

[7] Schickler, M.: Ein rahmenwerk zur mobilen unterstu?tzung therapeutischer interventionen. unpubished (2018)

[8] Schickler, M., Pryss, R., Schobel, J., Schlee, W., Probst, T., Reichert, M.: Towards flexible remote therapeutic interventions. In: 30th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2017), IEEE Computer Society Press (2017)

[9] Schickler, M., Pryss, R., Schlee, W., Probst, T., Langguth, B., Schobel, J., Reichert, M.: Usability study on mobile processes enabling remote therapeutic interventions. In: 31th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2018), IEEE Computer Society Press (2018) 146–151

[10] Schickler, M., Pryss, R., Stach, M., Schobel, J., Schlee, W., Probst, T., Langguth, B., Reichert, M.: An it platform enabling remote therapeutic interventions. In: 30th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2017), IEEE Computer Society Press (2017)

[11] Schickler, M., Pryss, R., Schobel, J., Reichert, M.: Supporting remote therapeutic interventions with mobile processes. In: 6th IEEE International Conference on AI & Mobile Services (IEEE AIMS 2017), IEEE Computer Society Press (2017)

[12] Këpuska, V., Bohouta, G.: Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In: 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC). (2018) 99–103

[13] Amazon: Alexa documentation - alexa skills kit. `https://developer.amazon.com/de/docs/custom-skills/` (2019) [Online; Accessed: 15-08-2019].

[14] Purington, A., Taft, J.G., Sannon, S., Bazarova, N.N., Taylor, S.H.: "alexa is my new bff": Social roles, user satisfaction, and personification of the amazon echo. In: Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems. CHI EA '17, New York, NY, USA, ACM (2017) 2853–2859

[15] Spalding, H., Chow, M.: Investigation of amazon alexa's explicit invocation policy. COMP116 (2018)

[16] Lau, J., Zimmerman, B., Schaub, F.: Alexa, are you listening?: Privacy perceptions, concerns and privacy-seeking behaviors with smart speakers. Proc. ACM Hum.-Comput. Interact. **2** (2018) 102:1–102:31

[17] Ford, M., Palmer, W.: Alexa, are you listening to me? an analysis of alexa voice service network traffic. Personal Ubiquitous Comput. **23** (2019) 67–79

[18] Zhang, N., Mi, X., Feng, X., Wang, X., Tian, Y., Qian, F.: Understanding and mitigating the security risks of voice-controlled third-party skills on amazon alexa and google home. CoRR **abs/1805.01525** (2018)

[19] Cutsinger, P.: How building for voice differs from building for the screen. `https://build.amazonalexadev.com/how-building-for-voice-differs-from-screen-on-demand-webinar-video-ww.html?aliId=35577122` (2018) [Online; Accessed: 15-08-2019].

[20] Polyakov, E.V., Mazhanov, M.S., Rolich, A.Y., Voskov, L.S., Kachalova, M.V., Polyakov, S.V.: Investigation and development of the intelligent voice assistant for the internet of things using machine learning. In: 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT). (2018) 1–5

[21] Williams, C.: 4 principles of conversational voice design. `https://developer.amazon.com/de/blogs/alexa/post/57d0bb9c-19a6-4c51-bfa2-fc6753d14b68/4-principles-of-conversational-voice-design` (2018) Alexa Blog [Online; Accessed: 15-08-2019].

[22] Kaye, J.J., Fischer, J., Hong, J., Bentley, F.R., Munteanu, C., Hiniker, A., Tsai, J.Y., Ammari, T.: Panel: Voice assistants, ux design and research. In: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems. CHI EA '18, New York, NY, USA, ACM (2018) panel01:1–panel01:5

[23] Amazon: Alexa Skills Kit - NLU. `https://developer.amazon.com/de/alexa-skills-kit/nlu` (2019) [Online; Accessed: 13-09-2019].

[24] Microsoft: Vs code. `https://code.visualstudio.com/` (2019) [Online; Accessed: 15-08-2019].

[25] Foundation, L.: Node js. `https://nodejs.org/en/about/` (2019) [Online; Accessed: 15-08-2019].

[26] Franklin Lobb, Akersh Srivastava, K.S., McCauley, R.: Github fact skill template. `https://github.com/alexa/skill-sample-nodejs-fact` (2019) [Online; Accessed: 22-08-2019].

[27] Reis, A., Paulino, D., Paredes, H., Barroso, I., Monteiro, M.J., Rodrigues, V., Barroso, J.: Using intelligent personal assistants to assist the elderlies an evaluation of amazon alexa, google assistant, microsoft cortana, and apple siri. In: 2018 2nd International Conference on Technology and Innovation in Sports, Health and Wellbeing (TISHW). (2018) 1–5

[28] Sayago, S., Neves, B.B., Cowan, B.R.: Voice assistants and older people: Some open issues. In: Proceedings of the 1st International Conference on Conversational User Interfaces. CUI '19, New York, NY, USA, ACM (2019) 7:1–7:3

[29] Kowalski, J., Jaskulska, A., Skorupska, K., Abramczuk, K., Biele, C., Kopec, W., Marasek, K.: Older adults and voice interaction: A pilot study with google home. CoRR **abs/1903.07195** (2019)

[30] Ziman, R., Walsh, G.: Factors affecting seniors' perceptions of voice-enabled user interfaces. In: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems. CHI EA '18, New York, NY, USA, ACM (2018) LBW591:1–LBW591:6

A Sources

A.1 Reminder Request

```
{   "trigger": {
         "type" : "SCHEDULED_ABSOLUTE",
         "scheduledTime" : "2019-09-22T19:00:00.000",
         "timeZoneId" : "Frankfurt/Germany",
         "recurrence" : {
             "freq" : "NONE",
             "byDay": ["MO"]
         }
    },
    "alertInfo": {
         "spokenInfo": {
             "content": [{
                 "locale": "en-US", "text": "exercise"
             }]
         }
     },
     "pushNotification" : {
          "status" : "ENABLED"
     }}
```

Listing A.1: reminder request

A.2 Alexa Request Body

```
{
   "request": {
                 "type": "IntentRequest",
                 "requestId": "amzn1.echo-api.request.
                 1e17ca16-9487-4606-bb35-96a410e73644",
                 "timestamp": "2019-08-26T12:33:11Z",
                 "locale": "en-US",
                 "intent": {
                         "name": "GetNewExIntent",
                         "confirmationStatus": "NONE",
                         "slots": {
                                 "name": {
                                         "name": "name",
                                         "value": "neck lateral
                                            flexion",
                                         "confirmationStatus": "
                                            NONE",
                                         "source": "USER"
                                 }
                         }
                 },
                 "dialogState": "COMPLETED"
         }
}
```

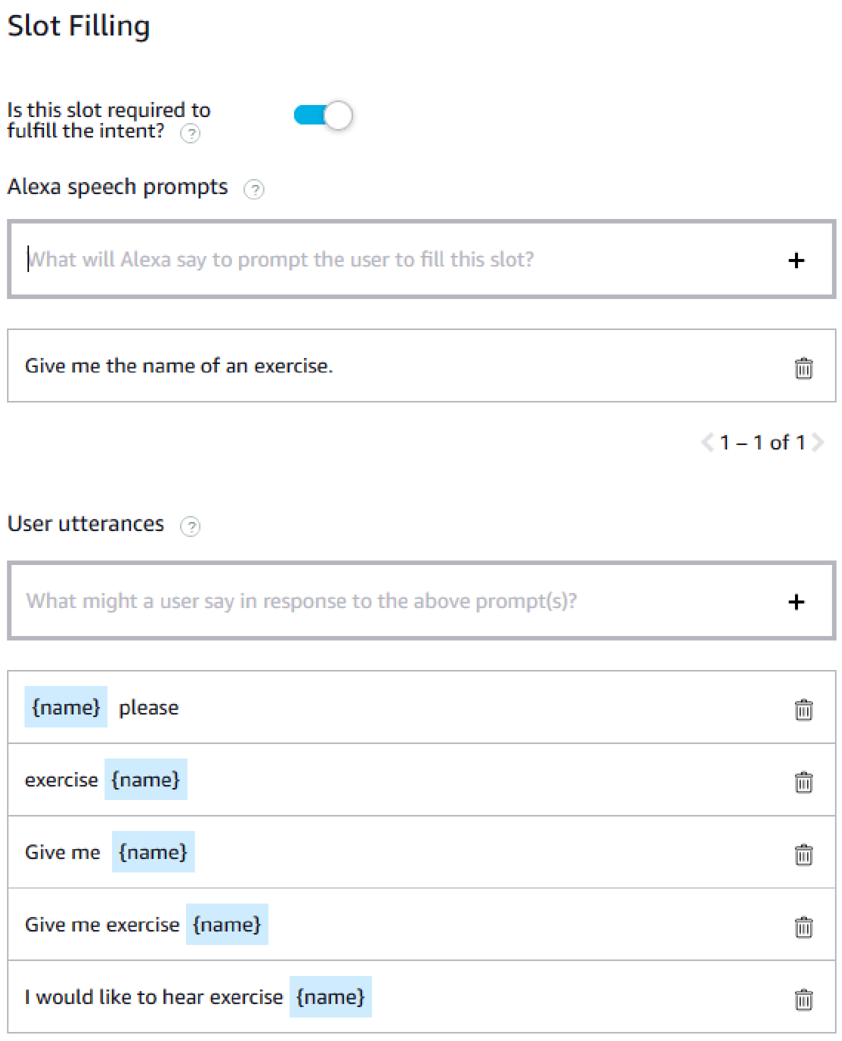
Listing A.2: alexa request body

A.3 Alexa Response Body

```
{
        "body": {
                "version": "1.0",
                "response": {
                        "outputSpeech": {
                                "type": "SSML",
                                "ssml": "<speak>Your exercise
                                   Neck lateral flexion:
                                   Position yourself in sitting
                                    or standing with a good
                                   posture. Centre your neck
                                   .....</speak>"
                        },
                        "reprompt": {
                                "outputSpeech": {
                                        "type": "SSML",
                                        "ssml": "<speak>What
                                           would you like to do
                                           ?</speak>"
                                }
                        },
                        "shouldEndSession": false,
                        "type": "_DEFAULT_RESPONSE"
                },
                "sessionAttributes": {},
                "userAgent": "ask-node/2.5.2 Node/v8.10.0"
        }
}
```

Listing A.3: alexa response body

B
Screenshots



Slot Filling

Is this slot required to fulfill the intent? ?

Alexa speech prompts ?

What will Alexa say to prompt the user to fill this slot? +

Give me the name of an exercise.

1 – 1 of 1

User utterances ?

What might a user say in response to the above prompt(s)? +

{name} please

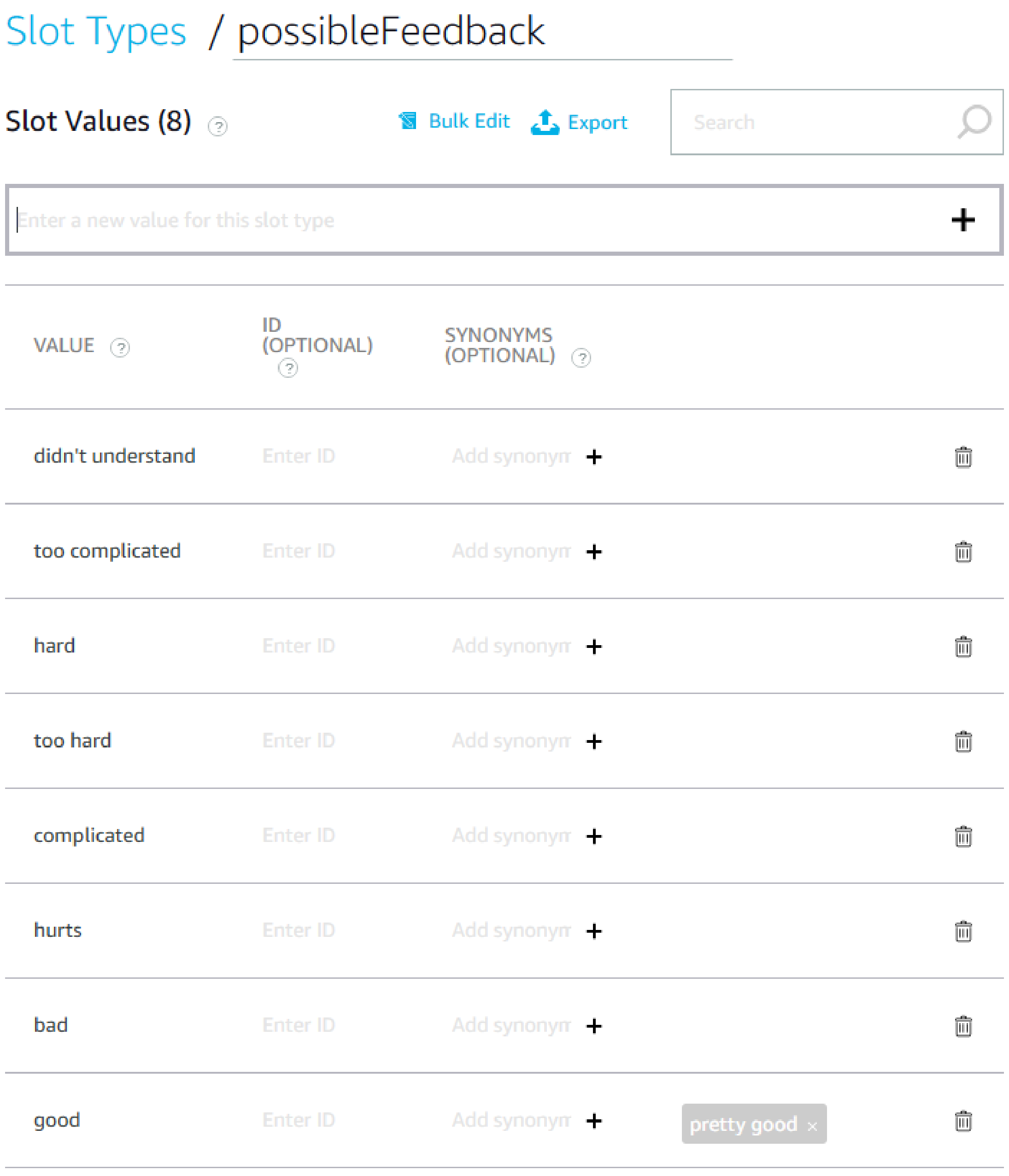
exercise {name}

Give me {name}

Give me exercise {name}

I would like to hear exercise {name}

Figure B.1: Exercise name utterances



Slot Types / possibleFeedback

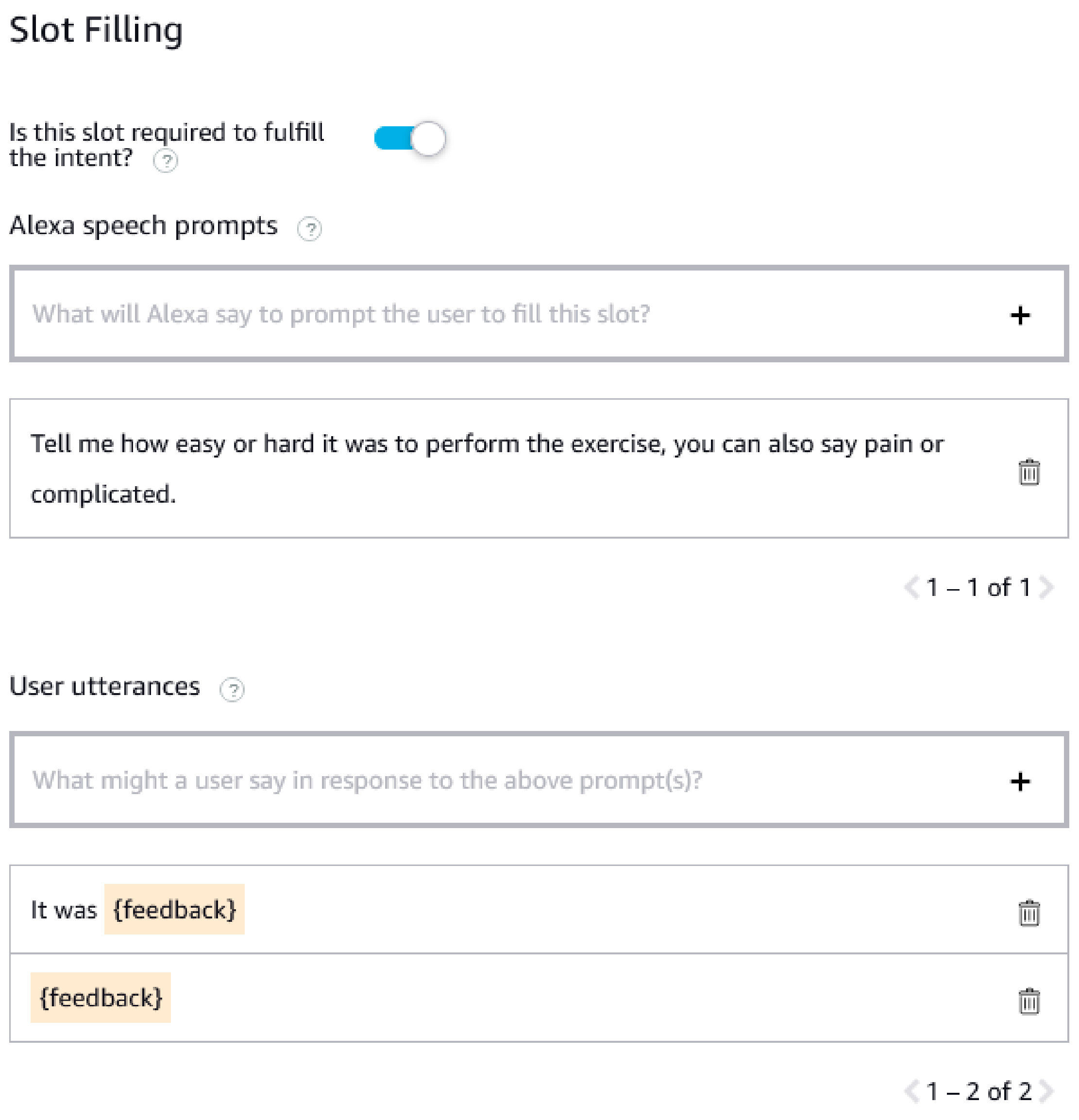
Slot Values (8)

Bulk Edit Export Search

Enter a new value for this slot type

VALUE	ID (OPTIONAL)	SYNONYMS (OPTIONAL)
didn't understand		
too complicated		
hard		
too hard		
complicated		
hurts		
bad		
good		pretty good

Figure B.2: Custom Slot Feedback



Slot Filling

Is this slot required to fulfill the intent?

Alexa speech prompts

What will Alexa say to prompt the user to fill this slot?

Tell me how easy or hard it was to perform the exercise, you can also say pain or complicated.

1 – 1 of 1

User utterances

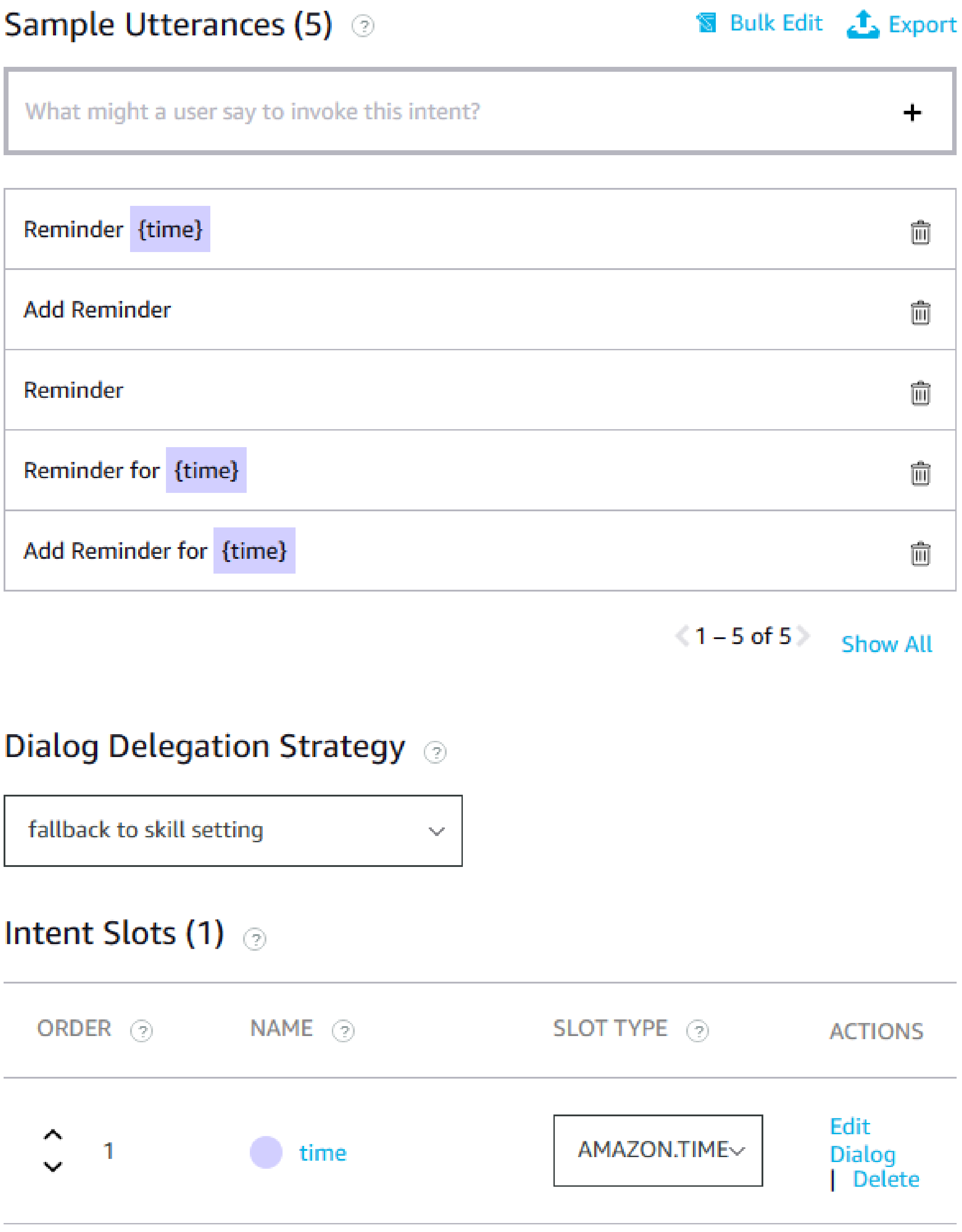
What might a user say in response to the above prompt(s)?

It was {feedback}

{feedback}

1 – 2 of 2

Figure B.3: Feedback Prompt Phrases



Sample Utterances (5)

What might a user say to invoke this intent?

Reminder {time}
Add Reminder
Reminder
Reminder for {time}
Add Reminder for {time}

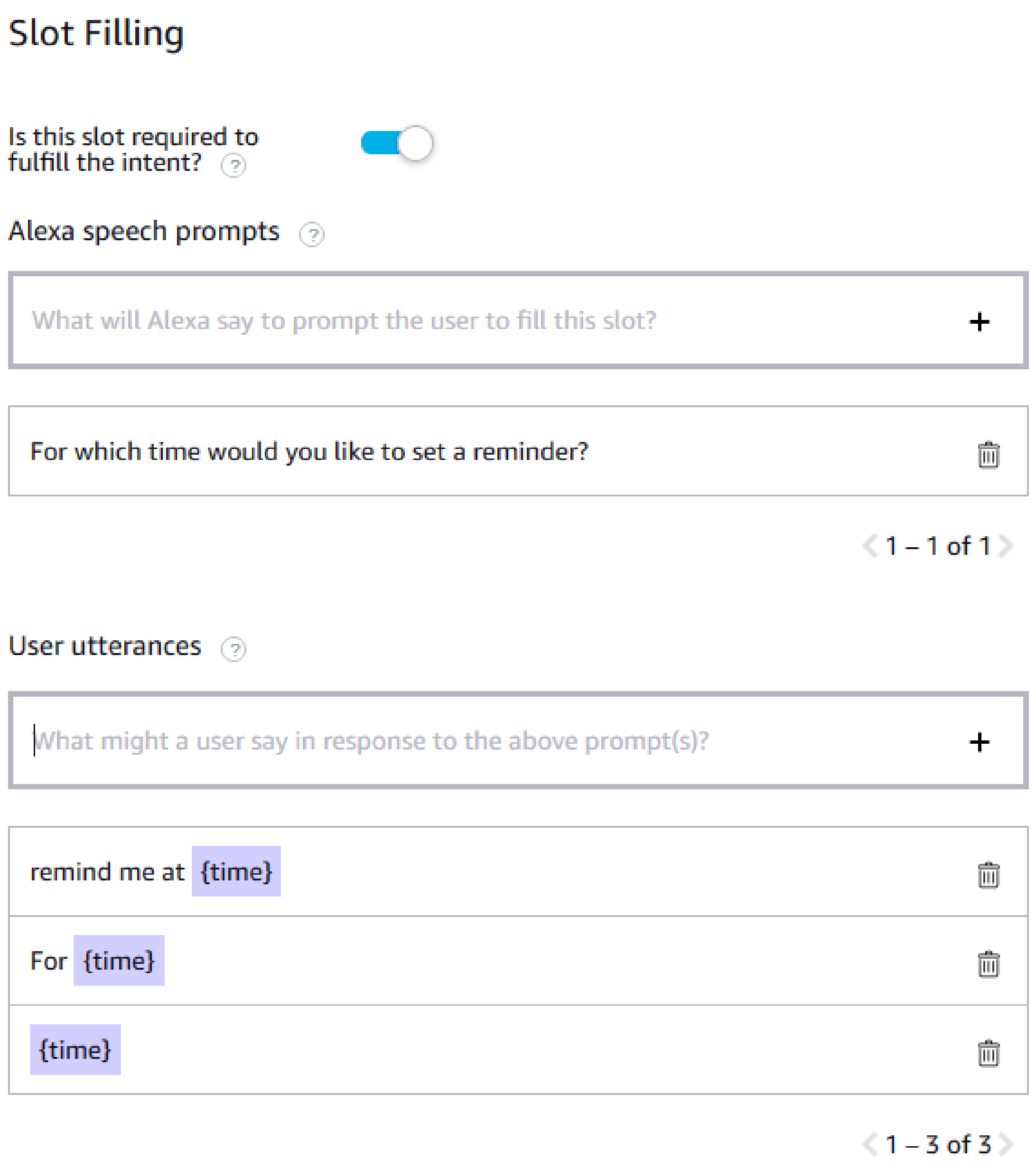
Dialog Delegation Strategy

fallback to skill setting

Intent Slots (1)

ORDER	NAME	SLOT TYPE	ACTIONS
1	time	AMAZON.TIME	Edit Dialog | Delete

Figure B.4: Reminder Sample Utterances



Slot Filling

Is this slot required to fulfill the intent? ?

Alexa speech prompts ?

What will Alexa say to prompt the user to fill this slot? +

For which time would you like to set a reminder?

1 – 1 of 1

User utterances ?

What might a user say in response to the above prompt(s)? +

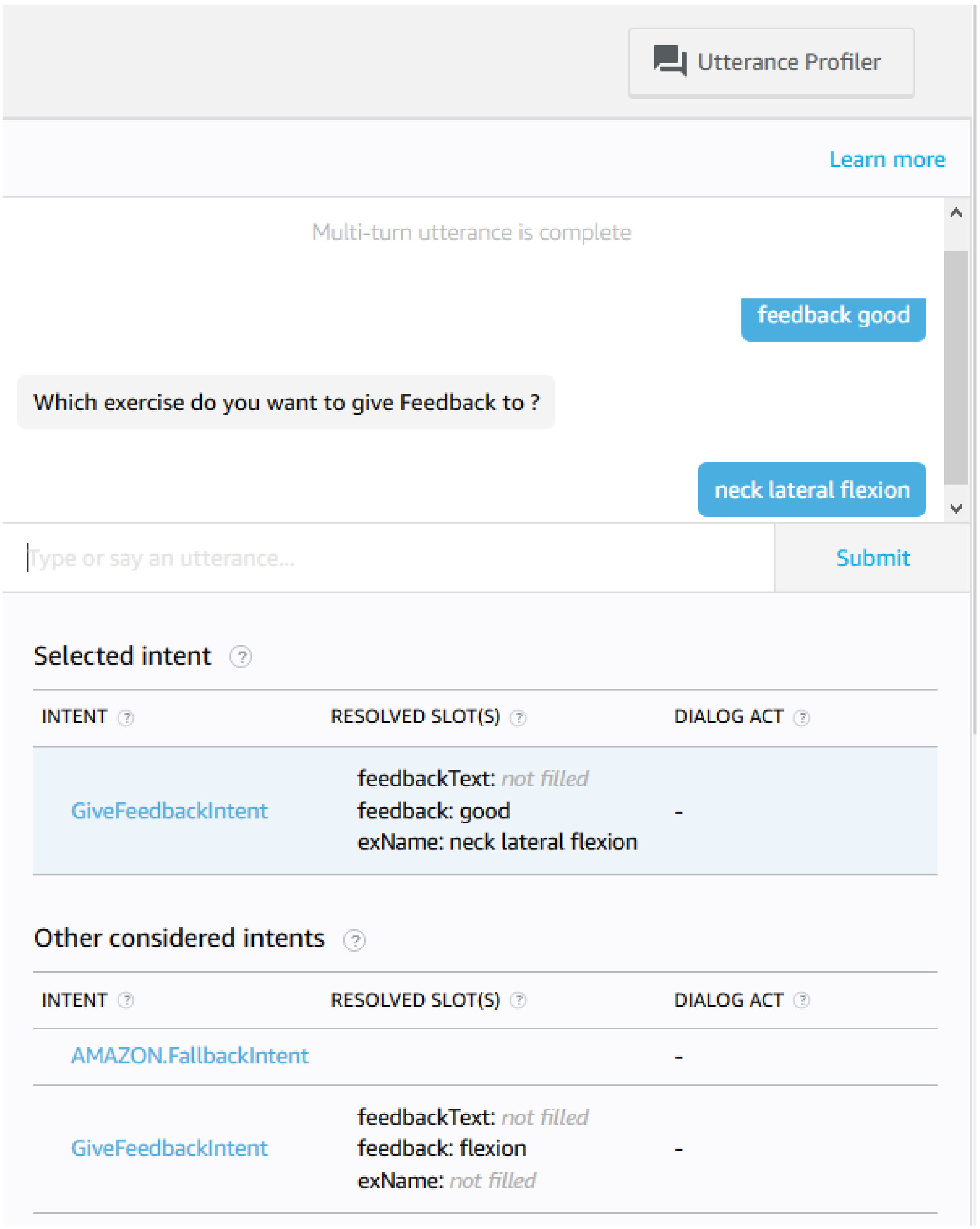
remind me at {time}

For {time}

{time}

1 – 3 of 3

Figure B.5: Reminder Prompt Phrases



Multi-turn utterance is complete

feedback good

Which exercise do you want to give Feedback to ?

neck lateral flexion

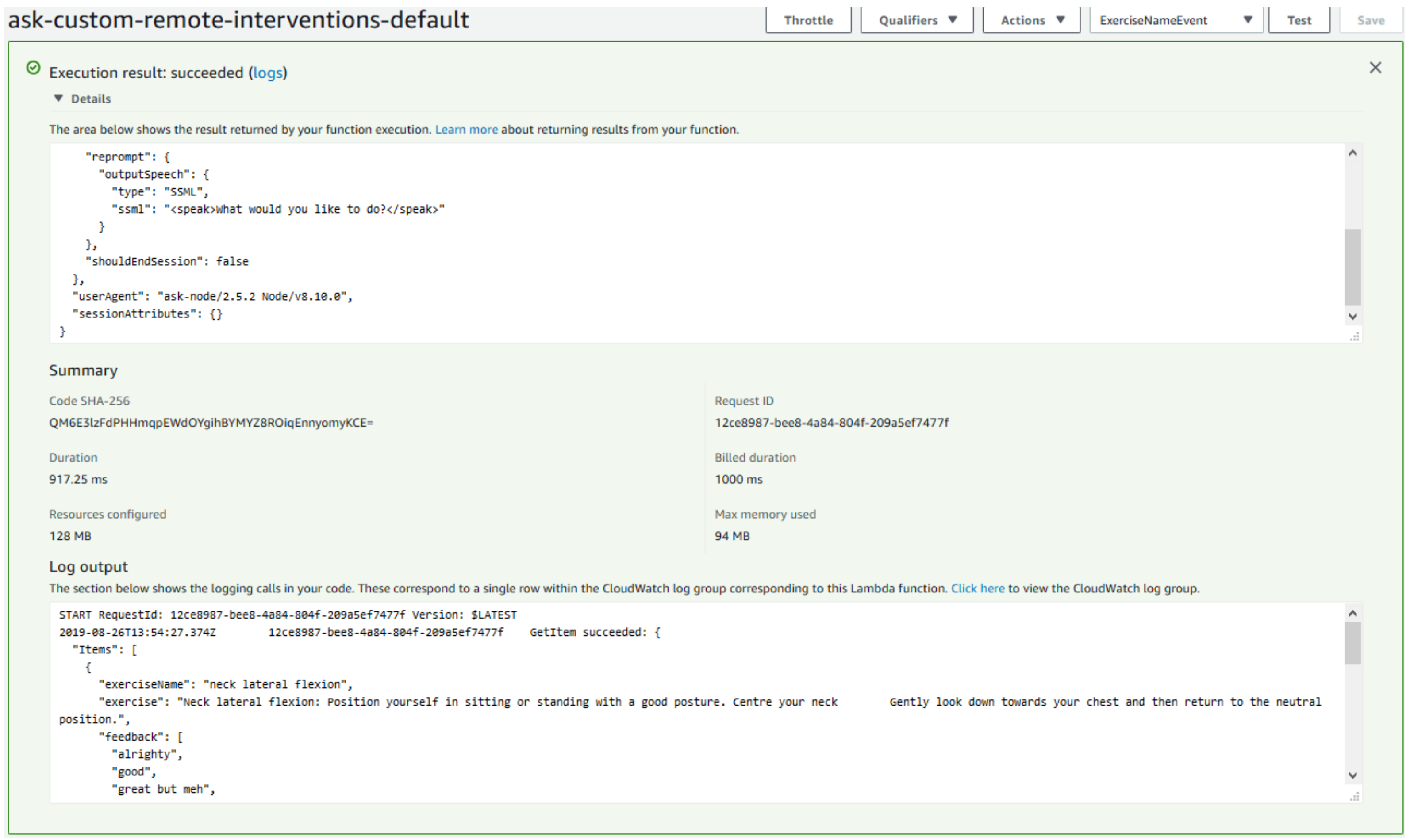
Selected intent

INTENT	RESOLVED SLOT(S)	DIALOG ACT
GiveFeedbackIntent	feedbackText: *not filled* feedback: good exName: neck lateral flexion	-

Other considered intents

INTENT	RESOLVED SLOT(S)	DIALOG ACT
AMAZON.FallbackIntent		-
GiveFeedbackIntent	feedbackText: *not filled* feedback: flexion exName: *not filled*	-

Figure B.6: Utterance Profiler Example



Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
    "reprompt": {
      "outputSpeech": {
        "type": "SSML",
        "ssml": "<speak>What would you like to do?</speak>"
      }
    },
    "shouldEndSession": false
  },
  "userAgent": "ask-node/2.5.2 Node/v8.10.0",
  "sessionAttributes": {}
}
```

Summary

Code SHA-256
QM6E3lzFdPHHmqpEWdOYgihBYMYZ8ROiqEnnyomyKCE=

Request ID
12ce8987-bee8-4a84-804f-209a5ef7477f

Duration
917.25 ms

Billed duration
1000 ms

Resources configured
128 MB

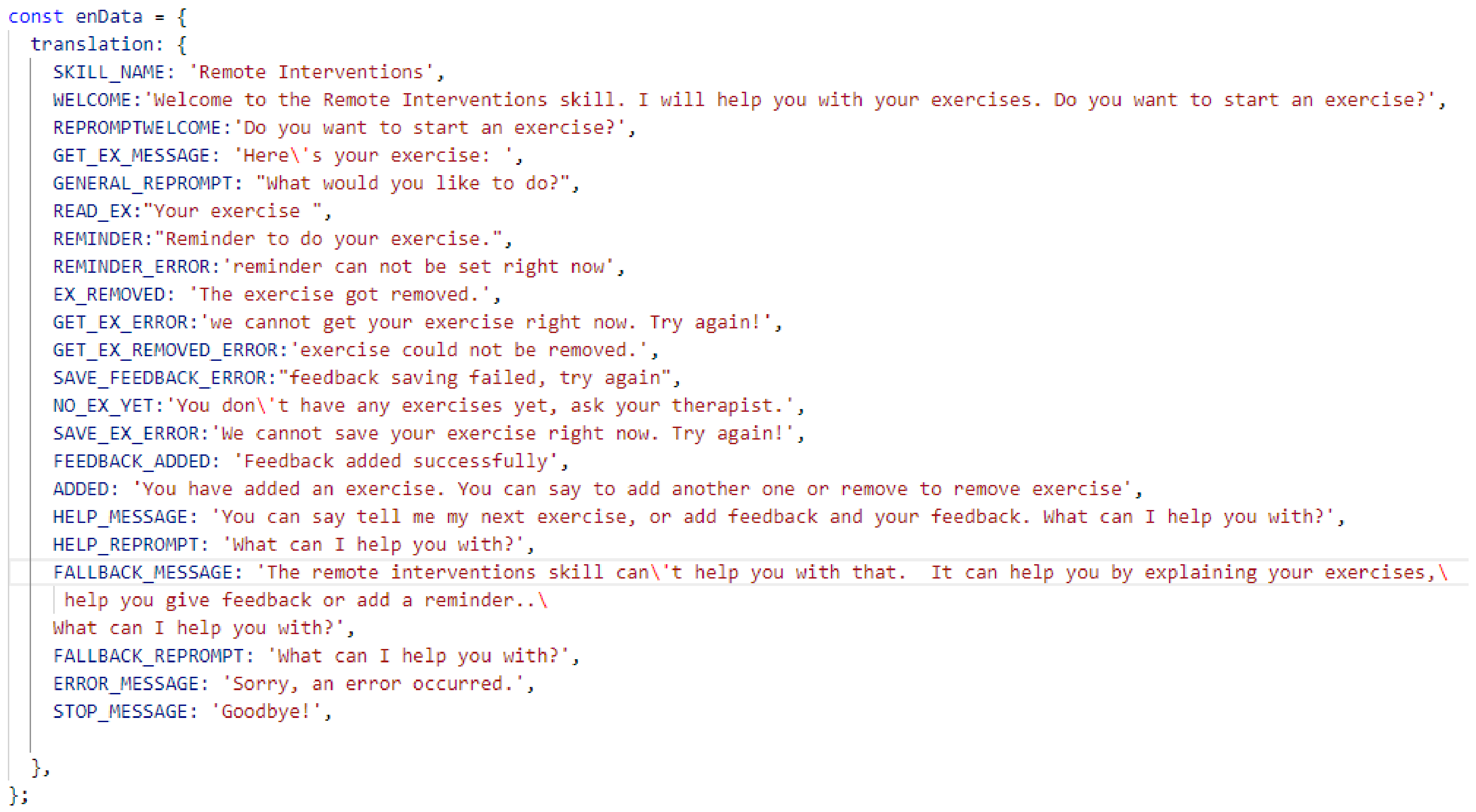
Max memory used
94 MB

Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.

```
START RequestId: 12ce8987-bee8-4a84-804f-209a5ef7477f Version: $LATEST
2019-08-26T13:54:27.374Z	12ce8987-bee8-4a84-804f-209a5ef7477f	GetItem succeeded: {
  "Items": [
    {
      "exerciseName": "neck lateral flexion",
      "exercise": "Neck lateral flexion: Position yourself in sitting or standing with a good posture. Centre your neck	Gently look down towards your chest and then return to the neutral position.",
      "feedback": [
        "alrighty",
        "good",
        "great but meh",
```

Figure B.7: Test Event



```
const enData = {
  translation: {
    SKILL_NAME: 'Remote Interventions',
    WELCOME:'Welcome to the Remote Interventions skill. I will help you with your exercises. Do you want to start an exercise?',
    REPROMPTWELCOME:'Do you want to start an exercise?',
    GET_EX_MESSAGE: 'Here\'s your exercise: ',
    GENERAL_REPROMPT: "What would you like to do?",
    READ_EX:"Your exercise ",
    REMINDER:"Reminder to do your exercise.",
    REMINDER_ERROR:'reminder can not be set right now',
    EX_REMOVED: 'The exercise got removed.',
    GET_EX_ERROR:'we cannot get your exercise right now. Try again!',
    GET_EX_REMOVED_ERROR:'exercise could not be removed.',
    SAVE_FEEDBACK_ERROR:"feedback saving failed, try again",
    NO_EX_YET:'You don\'t have any exercises yet, ask your therapist.',
    SAVE_EX_ERROR:'We cannot save your exercise right now. Try again!',
    FEEDBACK_ADDED: 'Feedback added successfully',
    ADDED: 'You have added an exercise. You can say to add another one or remove to remove exercise',
    HELP_MESSAGE: 'You can say tell me my next exercise, or add feedback and your feedback. What can I help you with?',
    HELP_REPROMPT: 'What can I help you with?',
    FALLBACK_MESSAGE: 'The remote interventions skill can\'t help you with that.  It can help you by explaining your exercises,\
     help you give feedback or add a reminder..\
    What can I help you with?',
    FALLBACK_REPROMPT: 'What can I help you with?',
    ERROR_MESSAGE: 'Sorry, an error occurred.',
    STOP_MESSAGE: 'Goodbye!',

  },
};
```

Figure B.8: English Phrases

List of Figures

List of Tables

Name: Jens Fischer Matriculation number: 906518

Honesty disclaimer

I hereby affirm that I wrote this thesis independently and that I did not use any other sources or tools than the ones specified.

Ulm, ...

Jens Fischer