ALEXA FOR INSTRUCTIONS

Detailed Design

Team 7122: WAM@CC
Bowers, Cole
Branson, Courtney
Giddings, Allie
Subramanian, Kalya
Wulber, Steven

10/ 22/ 17

Table of Contents

List of Tables

List of Figures

Glossary

Alexa - An artificial intelligence agent created by Amazon.

Amazon Echo - A hands-free speaker that you control with your voice. It can complete tasks such as playing music, making phone calls, and getting news updates.

Amazon Web Services (AWS) - A cloud-based computing platform that provides a variety of services.

Custom slot - Alexa's way of allowing any phrase the user says to be recognized as long as there are example phrases that could be said provided.

Guide - A guide is a sequence of instructions (or steps) that can be used to accomplish a certain goal. Specifically, guides in this document come from the iFixIt website. Guides contain other useful information such as estimated time to complete the guide, number of steps, pictures, comments, flags, and materials.

iFixIt - A crowd-sourced online API that hosts thousands of repair guides for everyday items.

Intent - A phrase spoken by the user that our skill recognizes as a valid action step.

Lambda endpoint - An event-driven computing service that AWS provides. It hosts and runs code for Alexa skills.

Sample utterance - The different ways any one command could be stated by a user.

Skill - An application that can be added to Amazon devices that have Alexa enabled including the Echo and Echo Dot.

Introduction

Our project, myFixIt, is an Alexa skill that will read out instructions step by step from a guide selected from the iFixIt website. Our client is Dr. Joy Robinson, a former Georgia Tech professor who is now a professor at the University of Alabama. The main goal for this project is to enable users to do repair work without having to constantly return to a physical instruction guide to read the next instruction. By having Alexa read out instructions, the user will be able to continuously work on their task. In order to effectively achieve this goal, the user should be able to use Alexa to select a guide from myFixIt, ask for the next instruction in the guide, repeat an instruction, and go back to previous instructions. Alexa should also able to send images associated with an instruction guide to the user's phone via the Alexa app.

In addition to reading out the instructions in a guide, the user should also be able to ask Alexa for additional metadata about the guide including but not limited to the time estimate for a guide, its difficulty, and the number of instructions that are left. Some guides also include flags, tools, and comments. Alex should also be able to add the functionality to query the flags, tools, and comments for a particular guide. Additionally, a stretch goal for this project is to add a bookmarking system into our skill so that Alexa will remember guides that have been started and where the user left off so that one can resume a project when the skill restarts.

This detailed design document will serve as a technical introduction to our project. It covers the architecture and design of our system, including the static and dynamic components. This report also details how data is stored in our system. Lastly, the user interface (UI) of our skill is explained in great detail since the majority of the user's interactions with our skill will be through speech. This section includes a flowchart that represents how the conversation between a user and Alexa should go, and an explanation of how our system represents that flowchart in code. The UI section will also cover the sample utterances which are used to prompt Alexa to continue to the next stage of the conversation. After reading through this

detailed design document, technical readers should understand how to debug and maintain myFixIt.

System Architecture

All Amazon Alexa skills follow the same basic system architecture. The user is responsible for issuing voice commands to the Echo device. The Echo device is responsible for processing that command and sending it to the Amazon Alexa Service Platform. The platform serves as a gateway between application code that processes the command and the device. The platform also compresses the data being sent between the application code and Amazon devices. The application code is accessed via an Amazon Web Services (AWS) Lambda endpoint. The application then determines how to respond to the user command and passes back the response to the Echo device. It does this through the Amazon Alexa Service Platform. See Figure 1, which was adapted from [2], below for more details.

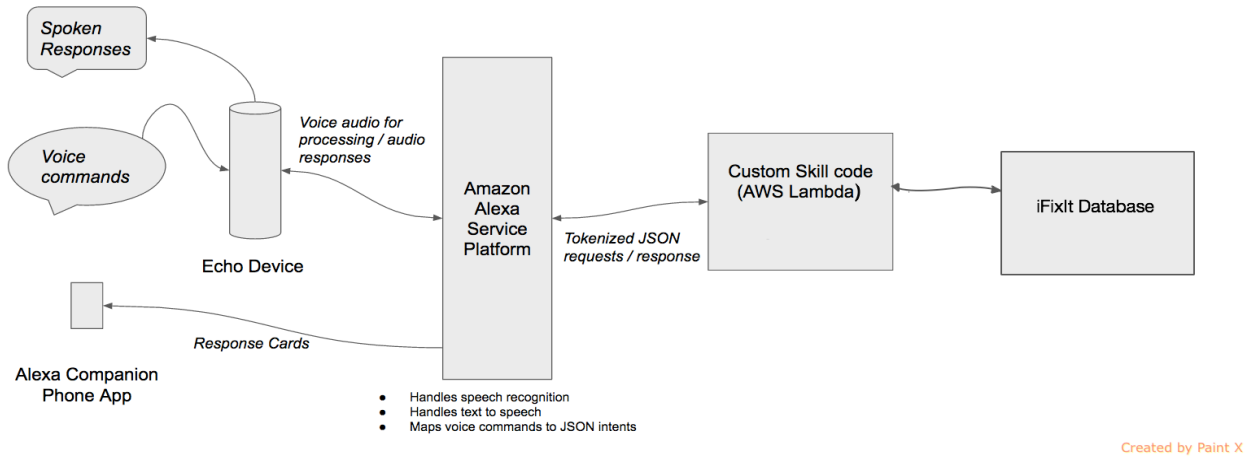

Figure 1: Alexa Skill Static Elements

The dynamic components of our application include the interactions between Alexa and the user. At each stage of the conversation between Alexa and the user, Alexa should prompt the user with a question and then process the user's answer to that question. Based on the user's response, Alexa should move to the next appropriate stage in the conversation. To see an example of a user invoking myFixIt and the conversation that follows, see Figure 2 below.

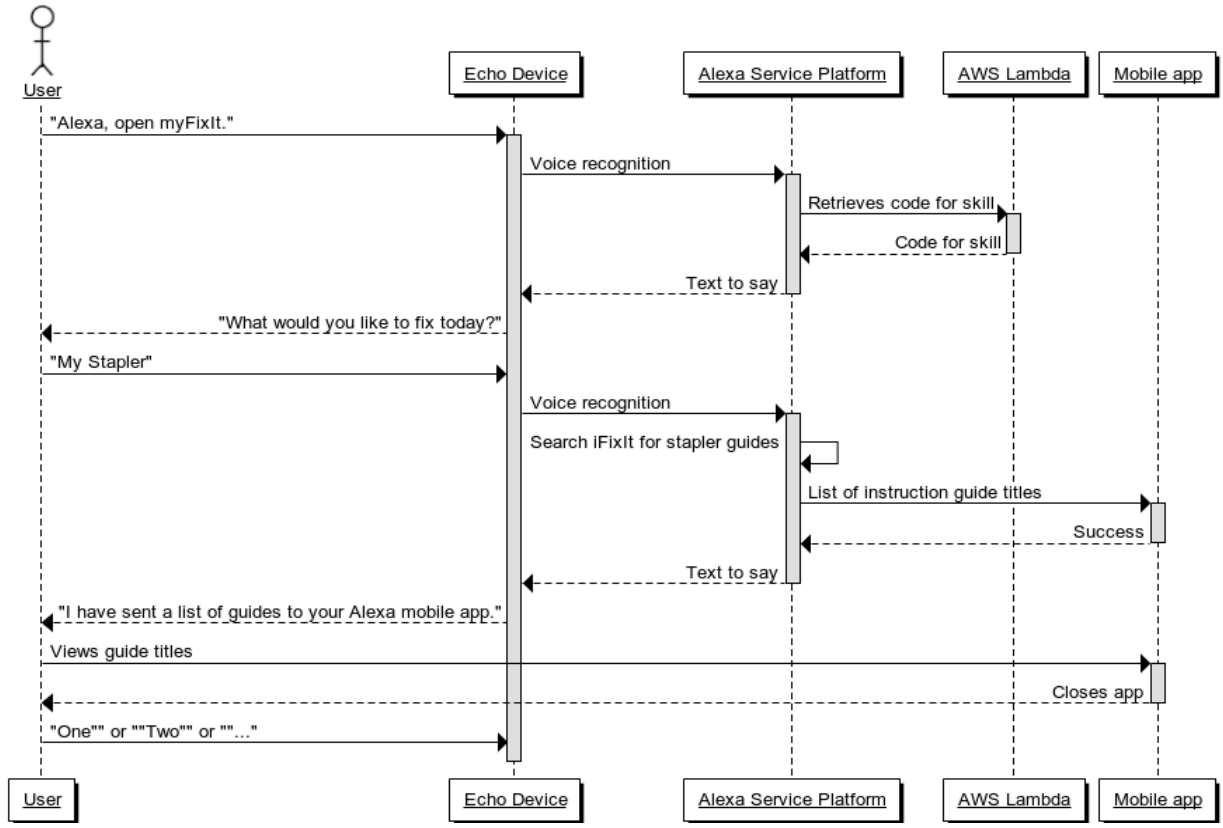

Figure 2: myFixIt Dynamic Elements

Data Storage Design

Our system currently does not rely on data storage but rather acts as a medium between a currently existing database and Alexa enabled devices. Similarly, we do not use files to store or read information. All our work is done through data exchange. Our system interfaces with the IFixIt database that stores thousands of instruction guides and information about those guides. IFixIt created an API to get this information. All information from IFixIt is formatted as strings including images, which are encoded as URLs to images hosted by IFixIt. This API sends a JSON response that the system keeps as a local session attribute during the lifetime of the Alexa skill. A session attributes is a python dictionary tied to the current instance of the lambda function that is running. Session attributes also store all global variables and application state information. When the user requests any type of information the system references this local copy of the data to extract the data needed and then presents that data to the user in two

different ways. Data is transferred to Amazon devices as both auditory instructions and visual output when needed. For both visual and auditory data transfer, we go through the Flask-Ask Python package that acts as an Alexa API wrapper.

As a stretch goal, we plan to set up a DynamoDB database to allow users to bookmark steps and instruction guides. DynamoDB is a NoSQL database. This database stores the user ID (given by Amazon user), guide ID (given by IFixIt), guide title, and step number. This is the minimal amount of information we need to store to restore a user to a particular step. The database has a table of users. Each user will have a list of bookmarks which has three attributes the guide ID, guide title, and the step number. Each entry must have the user ID attribute as it is utilized as a key. The database includes the guide title for efficiency. When displaying all the bookmarks to the user, the system will not need to make HTTP calls to the iFixIt database for each guide but rather only one call after a guide is selected. This is a very small database as it is used only for a minor part of our system. Figure 3 shows a sample database layout and Figure 4 shows the ER diagram.

```
{
    "Users": [
        {
            "UserID": "123ABC",
            "Bookmarks": [
                {
                    "GuideID": 123456,
                    "GuideTitle": "Sample Title",
                    "StepNumber": 4
                },
                {
                    "GuideID": 987654,
                    "GuideTitle": "Title Two",
                    "StepNumber": 12
                }
            ]
        }
    ]
}
```

Figure 3: DynamoDB Sample Layout (Stretch Goal)

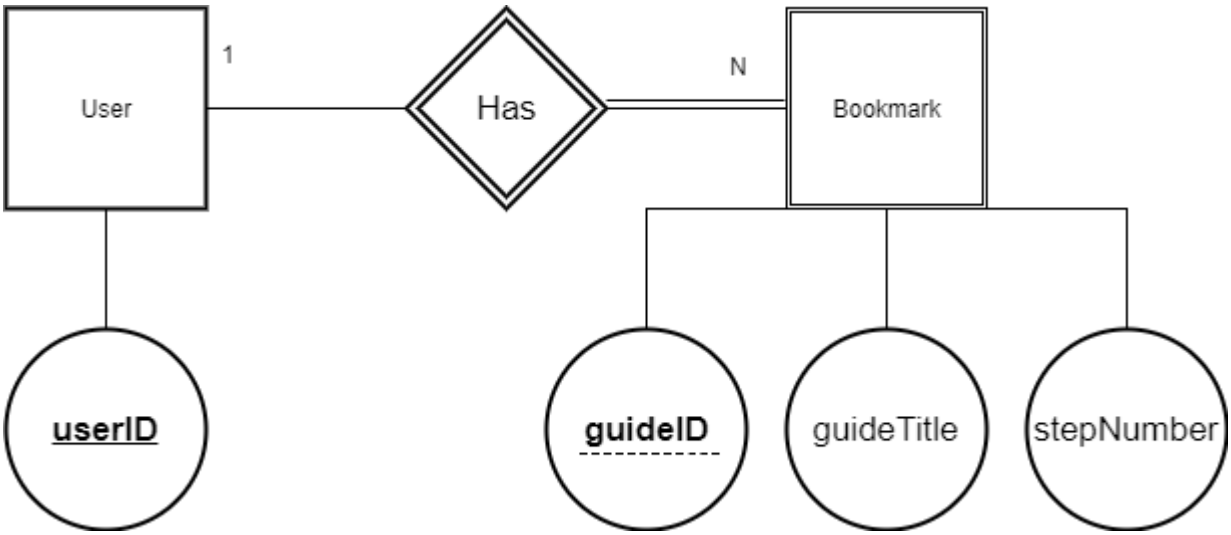

Figure 4: myFixIt ER Diagram (Stretch Goal)

Component Detailed Design

Due to the role of our application, the component design for our application is relatively simple. myFixIt is essentially an interface between IFixIt and Alexa. Organization and storage of instruction guides is handled by IFixIt, and language processing is handled by Alexa, so our application just needs to access and present that information according to the user's voice commands.

The component architecture of myFixIt is based on event-driven design that is required to work with Alexa. Alexa requires the existence of three main files: a main python class that holds methods for each intent, a list of sample utterances for each intent, and the intent schema. On top of this, we use a file that defines our custom slots, PyFixIt - a wrapper around the IFixIt API, and Flask-Ask - a Flask extension for building Alexa Skills. When the user gives a command to Alexa, Alexa builds and sends a request to myFixIt. MyFixIt will match what the user said to an intent via the intent schema, custom slot, and sample utterances. It then looks for the method associated with the intent in main.py. If an intent requires information about a guide, the skill will get it using PyFixIt, a wrapper around the IFixIt API. Finally, once the intent has finished building its response using Flask-Ask, myFixIt sends it to the Alexa device.

We have chosen to implement a finite state machine to represent how the user will interact with the different components of our system. The states in our application include: the launch state, the search guides state, the present search results state, the no results found state, the guide selected state, the guide completed state, the new guide option state, the help state, the instruction state, the start most recent guide state, the no bookmarks state, the present bookmarks state, and the remove bookmarks state. See figure 5 for more details. Each of the intents in our main python file corresponds to a voice command from the user. These intents are python methods that are called upon transitioning from one state to the other. The same intent will respond differently depending on the state. For example, The user could say 'yes' at any point, but in some states, 'yes' won't make any sense, and the skill will have to respond differently if the state is "Bookmark Option," or "New Guide Option."

Table 1: myFixIt States

States	**Question by Alexa**
launch	Would you like to continue a previous project or start a new one?
Search guides	What do you want to fix today?
Present search results	I sent a list of possible guides through the Alexa app. Please choose one or try searching again
No results found	No Instructions matched your search. Please try again.
Guide selected	You have selected [name of Instruction Guide]. [List tools needed]
Guide completed	You are done! Say what step you want to return to or say finish
New guide option	Would you like to work on something else?
Help	Help
Instruction	Step [x]: [Instruction] (Send Picture if Available)
Start most recent guide	[Instruction] at step [x] is your most recent save continue from here?
No bookmarks	No bookmarks: Start new instruction or check bookmarks again?

Present bookmarks	Here are your bookmarks (send bookmarks to phone)

Figure 5 (see below) is a diagram of how myFixIt components will interact dynamically with users. Blue ovals represent the state of the skill, while red boxes represent transitions to another state as a result of a user voice command. In the special cases when the user asks for steps remaining, the current step, the length of the guide or any other command that just loops back to the instruction state, the blue oval, is only represented as an intent in the application instead of a state. The arrows indicate the direction of flow from state to state. The dynamic components diagram includes all planned states and transitions in detail.

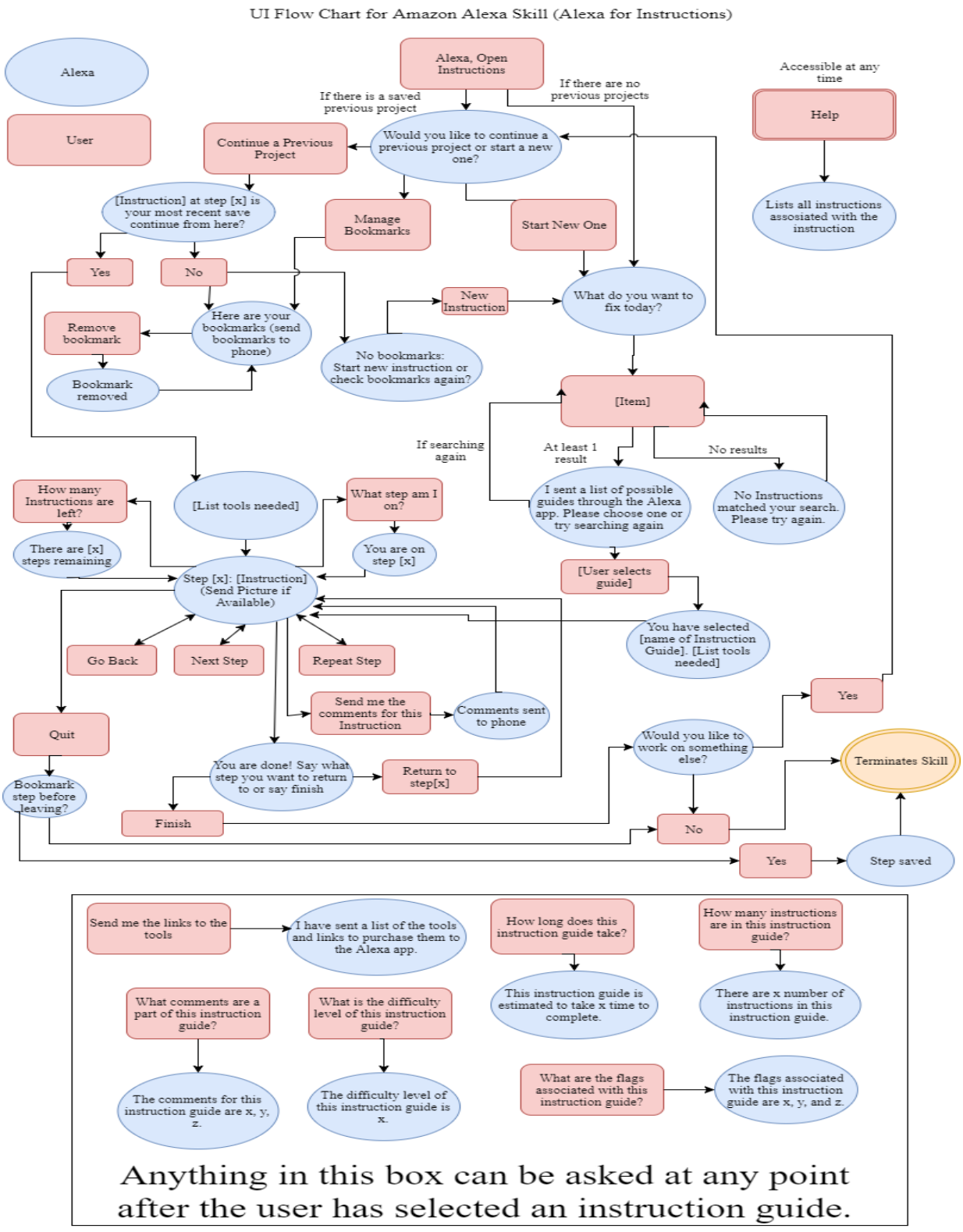

Figure 5: myFixIt State Machine

UI Design

Our UI design focused on the flow of conversation between Alexa and the user, as well as phrases Alexa should be able to recognize. We created a flowchart, which was shown in the component detailed design section of this document (Figure 5, myFixIt State Machine). Since the initial design of the flow chart, we have learned significantly more about how Alexa responds to user input. Essentially, for every function we have, we need to specify in the intents schema the name of the intent and slots which allow the function to receive information from the user. There is also a sample utterances page, where users can learn what phrases you can use to interact with Alexa for that skill. These phrases are also how Alexa determines what intent to go to (and how to respond to the user). For each of the circles on our flow chart that represents an interaction between Alexa and the user, we have a corresponding function and intent. Table 2, myFixIt Intents, details our current intents and their functionality:

Table 2: myFixIt Intents

Function	**Purpose**	**Place in flowchart**
search(item)	Sends a list of guides from searching iFixIt for a given item	Blue- I sent a list of possible guides to the Alexa app...
selectguide(guide number)	Selects a guide from a list using the guide number	Blue- You have selected{guide}
no_intent()	Exits the application	Orange- terminate skill
repeat_intent()	Says out loud the current instruction	Blue- Step[x] instruction
next_intent()	Says out loud the next instruction	Blue- Step[x] instruction
previous_intent()	Says out loud the previous instruction	Blue- Step[x] instruction
help_intent()	Gives help based on the current state	Red - Help

lenofguide_intent()	Returns a time estimate for how long a guide takes to complete	Blue - The instruction guide is estimated to take ...
numinstructions_intent()	Returns the total number of instructions for the selected guide	Blue - There are x number of instructions in this guide
curinstruction_intent()	Returns the step number you are on	Blue -You are on step [x]
instructionsleft_intent()	Says the number of instructions remaining in the guide	Blue - There are [x] steps remaining
difficulty_intent()	Tells the user the difficulty of the selected guide	Blue- The difficulty level of this guide is [x]
next_picture_intent()	Sends any pictures associated with the current instruction to the phone	Blue - Send pictures if available
tools_intent()	Informs the user what tools are required to complete the instruction guide	Blue - List tools needed
comments_intent()	Tells the user what comments are associated with a guide from the iFixit website	Blue- send comments
flags_intent()	Informs the user of any flags that may be associated with a guide. The flags are warnings to the user or any other special information the user needs to know about the guide.	Blue- Guide flags

Table 3 (see below) details our sample utterances, which are phrases someone can say to invoke the different intents. We designed the sample utterances last semester and updated them after having them reviewed by Junior Design team 7154. Below are the sample utterances we are currently using in our system. Some of these utterances are Amazon defaults, and we have marked those in the table.

Table 3: Sample Utterances

Intent	**Phrase**
search(item)	Search {item} Search for {item} {item} I want to fix my {item}
selectguide(guide number)	{guide_number} Guide number {guide_number}
no_intent()	Amazon Default: Stop
repeat_intent()	Amazon Default: Repeat
next_intent()	Amazon Default: Next
previous_intent()	Amazon Default: Previous
help_intent()	Help I don't know what to do
lenofguide_intent()	What is the length of guide {len_guide_number}
numinstructions_intent()	How many instructions are there
curinstruction_intent()	What instruction number am I on
instructionsleft_intent()	How many instructions are left
difficulty_intent()	What is the difficulty
next_picture_intent()	Next Picture
tools_intent()	What are the tools
comments_intent()	What are the comments
flags_intent()	What are the flags

After the initial design of our UI, we did some usability testing with other students using Nielsen's heuristics [1]. One principle is aesthetic and minimalist design, and so we chose to keep Alexa's responses short and concise. One user commented "Application dialogs contain only the few words that are necessary. Lots of free space instead of clutter." Users also had

positive comments for the Flexibility and Ease of Use heuristic, saying “The system is friendly to new users and not hard to learn. Uses a system of static, immutable commands.” After the user testing, we realized there was room for improvement with the Help and Documentation heuristic. We completely redesigned the way the application would respond when a user asked for help, by making the help specific to the state of the application. Instead of asking the user what they needed help with, Alexa would tell the user what they should say next to move forward.

References

[1] Nielson, Jakob. "Enhancing the explanatory power of usability heuristics." In *CHI '94 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 152-58. Boston, MA: Association for Computing Machinery, 1994.

[2] Prickett, Simon. "Alexa Custom Skill System Architecture." Digital image. Medium. August 13, 2016. Accessed October 4, 2017 [Online]. Available: https://medium.com/modus-create-front- end-development/creating-an-interactive-voice-experience-with-amazon-alexa-2fed5d9b0e16.