Staying Sharp

In partnership with ITS, Spring 2019

1. Executive Summary

Declination of cognitive abilities such as memory and focus has been a prominent issue to affect the elderly population. In order to 'disrupt aging', the Tech Nest has developed *Staying Sharp* for the Amazon Alexa platform. *Staying Sharp* uses scientifically proven exercises to strengthen your brain and improve memory and focus.

2. Background

According to the National Institute of Aging, declination of cognitive abilities is a prime concern amongst the elderly. One's cognitive abilities determines their ability to perform everyday tasks and their ability to live independently. As one ages, certain parts of the brain shrink - especially those important to learning and other cognitive activities such as memory. Blood flow in the brain also decreases with age. Vascular factors not only contributing to cognitive problems in ageing but also to the two most common dementias. Moreover, Neurotransmitters such as dopamine and serotonin see a decline in levels with age, affecting cognitive and motor abilities as well as synaptic plasticity and neurogenesis.

Through *Staying Sharp,* AARP tries to empower the ageing population by allowing them to improve their cognitive abilities and thus lead an independent life.

3. Prototype developed

The following prototype was developed in the Fall 2017 and Spring 2017 at the Tech Nest:

3.1 ***Staying Sharp* for Alexa:** AARP is bringing the *Staying Sharp* program to the Amazon Alexa platform to make the program more accessible through an intuitive voice assistant. The Amazon Echos are becoming increasingly popular amongst senior citizens and are being used to improve their quality of life[1].

The *Staying Sharp* program is brought to the Echo devices as an Alexa skill. Within the skill, the user can choose between two different memory exercises called 'numbers' and 'words'. Both these exercises are adaptive and increase or decrease in difficulty in order to match the user's proficiency. At the end of the exercise, Alexa provides the user with a score indicating their mastery. This score allows them to quantitatively measure their performance.

4. The Amazon Alexa Platform and Interaction Table

4.1 Alexa Skill Development: The Alexa Skill for *Staying Sharp* was developed in the Amazon Alexa Developer Console and the Amazon Web Services Lambda platform. The Developer Console supported the front end interactions between the user and Alexa while the Lambda function controlled the back end logic

[1] Amazon Echo for Dementia: Technology for Seniors - DailyCaring. (2019). DailyCaring. Retrieved 23 April 2019, from https://dailycaring.com/amazon-echo-for-dementia-technology-for-seniors/

for the exercises such as evaluating responses, measuring the user's proficiency and performing a variety of non-linear[2] tasks.

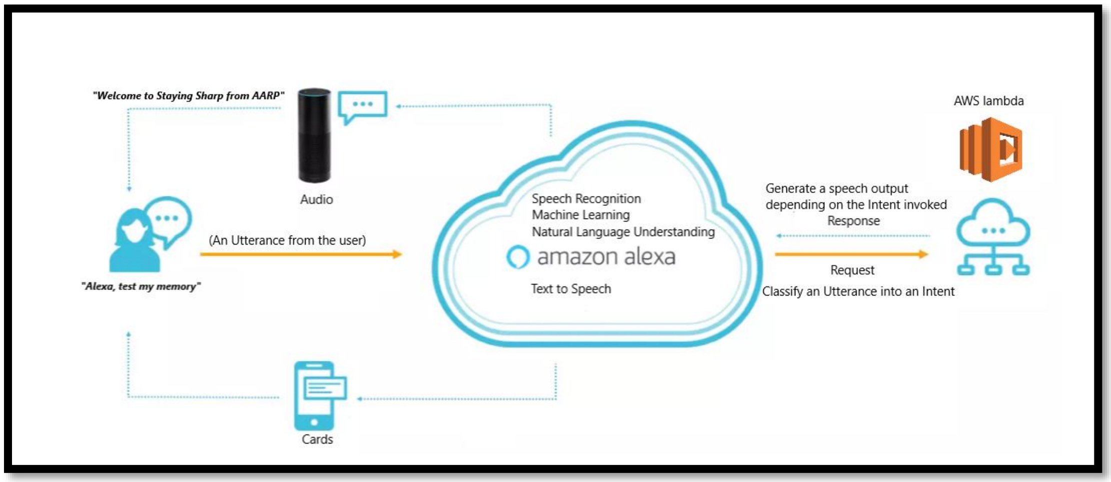

"Welcome to Staying Sharp from AARP"
Audio
(An Utterance from the user)
"Alexa, test my memory"
Speech Recognition
Machine Learning
Natural Language Understanding
amazon alexa
Text to Speech
AWS lambda
Generate a speech output depending on the Intent invoked
Response
Request
Classify an Utterance into an Intent
Cards

Figure 1 A flowchart depicting the interaction between the user and Alexa and the front-end and back-end components involved

The Alexa Developer Console contains various 'intents'. Each intent is classified by a few sample utterances[3]. For example, by saying *"Alexa, restart the game",* an utterance, the user intends to restart the game. This is handled by the AWS lambda function. After the user says an utterance, Alexa will classify that into an intent. The intent will then be invoked in a handler in the AWS lambda function and the appropriate action will be undertaken such as evaluating an answer or prompting the user for a new response. In the case for restarting the game this would involve resetting the variables and prompting the user to choose between numbers and words.

The code containing the JSON with the front-end schema and the Node.js files on AWS lambda can be found on AARP's GitLab. A link to the repository is provided below (4.2).

4.2 Link to GitLab: http://10.137.124.150/samarth.makhija-AARP/Staying-Sharp

4.3 Interaction Table: The following table can be used to interact with Alexa as a user. Alternatively, you can say "*Instructions*" after launching the skill by saying "*Alexa, test my memory*". *Note: these are a small subset of numerous possible instructions.*

Scenario	Possible Instructions	Alexa response
Launch the skill	*"Alexa, test my memory"*	Alexa starts the skill
Select an exercise	"*words*" "*numbers*"	Starts the selected exercise
Provide an answer	*"four three two one" "leopard"*	Evaluates answer and continues
Ask for help	*"Help" "What was the original list"*	Repeats the original list
Ask for instructions	*"I'm confused" "Instructions"*	Says a set of instructions
Restart the exercise	*"restart"*	Restarts the skill
Ask for a tip	*"Can I get a tip?"*	Provides some tips to the user
Stop the exercise	*"Stop" "I am done"*	Ends the current session

[2] Non-linear tasks are defined to be those which are outside the expectations of gameplay. For example, the user asking Alexa for help, tips, instructions or to restart the game would outside the linear expectations of Alexa asking a question and the use providing an answer.

[3] These utterances may be inbuilt for the existing intents made by Amazon.

5. Description of the Exercises

Staying Sharp for Alexa contains two different tracks of exercises – ‘Numbers’ and ‘Words’.

5.1 Numbers: In the Numbers exercise, Alexa will say a series of 4 numbers after which she’ll prompt the user to repeat the series and evaluate their response as correct or incorrect. Upon each correct response, Alexa will increase the length of the series by 1 while on each incorrect response Alexa will decrease the length by 1 until a minimum of 4 numbers.

5.2 Words: In the Words exercise, Alexa will say a series of 7 words from an arbitrary theme and ask the user to remember them. Then, Alexa would say a series of 3 words, only one of which belongs to the original list. The user will be prompted to repeat the word they recognize from the original list. The length of the original list and the length of the words to choose from will increase upon correct answers.

5.3 Non-linear interactions: *Staying Sharp* for Alexa also supports interactions which may not be within the expected flow of the game but exist to provide the user with more control over the game. We term these as *non-linear interactions*. Non-linear interactions include the user asking for help, tips, instructions and restarts. *Figure 2* depicts a detailed flowchart illustrating the gameflow of the *Staying Sharp* exercises.

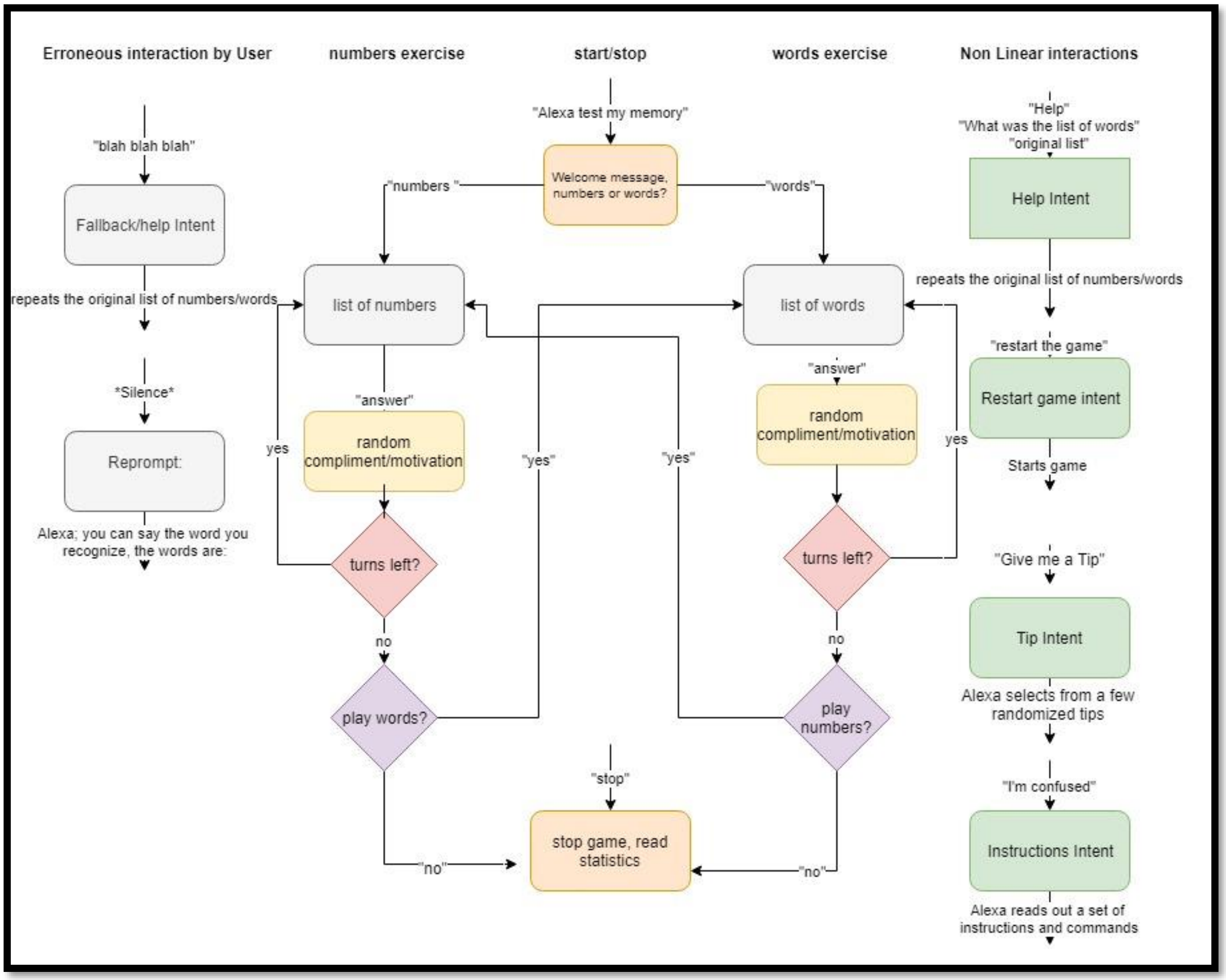


Figure 2 A flowchart indicating the game flow including the various ways in which a user might interact with Alexa

Legend

“Quotation marks” : Indicates an utterance from the user

Absence of Quotation marks: Indicates an action performed by Alexa

◆ : Indicates a conditional

🟧 : Indicates an Intent or a set of actions performed by the lambda function

6. Features Added and the Development Process

The feedback from the Alpha testing of the skill led to the result that the skill isn’t completely intuitive. Some instructions were unclear, the users had to use very specific phrases to interact with Alexa and the skill didn’t encourage the users as a learning experience should. Various features were added to the existing framework to enhance the user experience. Some of the features include:

6.1 Custom slot types: Added functionality to interact with Alexa using a single word. For example, a scenario from the words game would change as follows:

Then:

> **Alexa:** *Repeat the word that you recognize from the original list. tiger cat monkey cow. Just say the word is and ‘the word’*
> **User:** *The word is cow*

Now:

> **Alexa:** *Repeat the word that you recognize from the original list. tiger cat monkey cow.*
> ***User***: *Cow*

This has been enabled by creating a custom slot type which accepts all the possible answers in the words game. This is similar to creating a new data type or class in a computer program.

6.2 Tackling unresponsiveness and erroneous input: Earlier, Alexa would say “undefined” if the user was unresponsive when supposed to answer for the words or numbers game. Alexa now prompts the user for the answer. This also occurs as a fallback for erroneous input. Moreover, Alexa lets the user know that they can ask her for instructions since silence or erroneous input may indicate confusion.

6.3 Skipping the yes response: The process to start the game was too long earlier, requiring you to open staying sharp, say yes to play and then requiring you to choose between numbers and words. Now you can choose between the exercises right when you open staying sharp. This provides a more intuitive UX as requested in the feedback. Example:

Then:

> **Alexa:** *Welcome to Staying Sharp from AARP, would you like to exercise your memory? Say yes to begin the exercises or no to stop.*
> **User*:*** *Yes*
> **Alexa:** *Would you like to work with numbers or words? Say work with and then numbers or words.*
> **User*:*** *Work with words*

Now:

Alexa: *Welcome to Staying Sharp from AARP, we have two tracks numbers and words. Which one would you like to try first?*
User: *Words*

This combines with custom slot types to provide a more natural interaction model.

6.4 Restarts: Added functionality to restart the game when asked by the user to restart the game as opposed to stopping the program and restarting it.

6.5 Intuitive Dialogues: Used the principles from the Alexa design guide to rewrite dialogues to keep them clear, concise and supportive. A few examples of the process included asking users if they wanted more instructions or a full list of commands, letting the users know they can ask for instructions and writing brief dialogues to retain the user's attention span.

6.6 Don't know Response: Added a help response to support a user who doesn't know the answer or forgets the original list of numbers or words. If the user asks Alexa for help or says "I don't know", Alexa will repeat the original list of numbers or words.]

6.7 Randomly generated responses: Feedback from the Alpha testing showed that the skill was not able to set a positive learning environment with monotonous, repetitive dialogues and little to no encouragement or motivation. Several responses were added for both correct and incorrect responses to present a more natural setting.

6.8 Adding expressions to voice: To further stimulate a positive learning environment, we needed to add expression to Alexa's voice to make it sound more natural. Speech Synthesis Markup Language was used to develop this leading to responses such as "*Wow*", *"Excellent" and "Congratulations"* having an excited positive sound while words such as "*Bummer*" or "*Darn!*" had a motivating, 'Oh, so close' tone attached to it.

6.9 Game Switching: Since we had two tracks, we wanted to provide the user with the ability to switch between both games while keeping track of their performance on both the games. This was done by adding a feature which asks the user if they would like to begin the other game upon the completion of the first.

7. Documentation

7.1 Deployment Package: The deployment package is a compressed folder that contains all the files relevant to the project. The .zip can be uploaded directly to AWS Lambda in order to get the project's back-end up and running. The front_end.json file on the other hand, can be copied into the Alexa Developer Console in order to get access to the interaction models for the skill.

7.1.1 Explanation of files:

1. **Index.js:** This is the main file for the program that handles all functionality, features and actions for different intents.
2. **Front_end.json:** This file is the front-end schema that gets put onto the Alexa Developer console and makes for the voice recognition and natural language processing aspect of the program.
3. **Package.json:** This is default file generated by AWS Lambda that ensures that the right Alexa SDK is imported

4. **Dictionary.js:** Contains all the different words and categories that form the database for the "words" game. This also contains code that takes input from the main program and returns the list of words to remember and the list of words that are used to question the user, with just one right answer.
5. **ChunkingDictionary.js:** Generates the random list of numbers based on the difficulty level set by the program

7.1.2 Index.js Code documentation:

7.1.2.1 Variables:

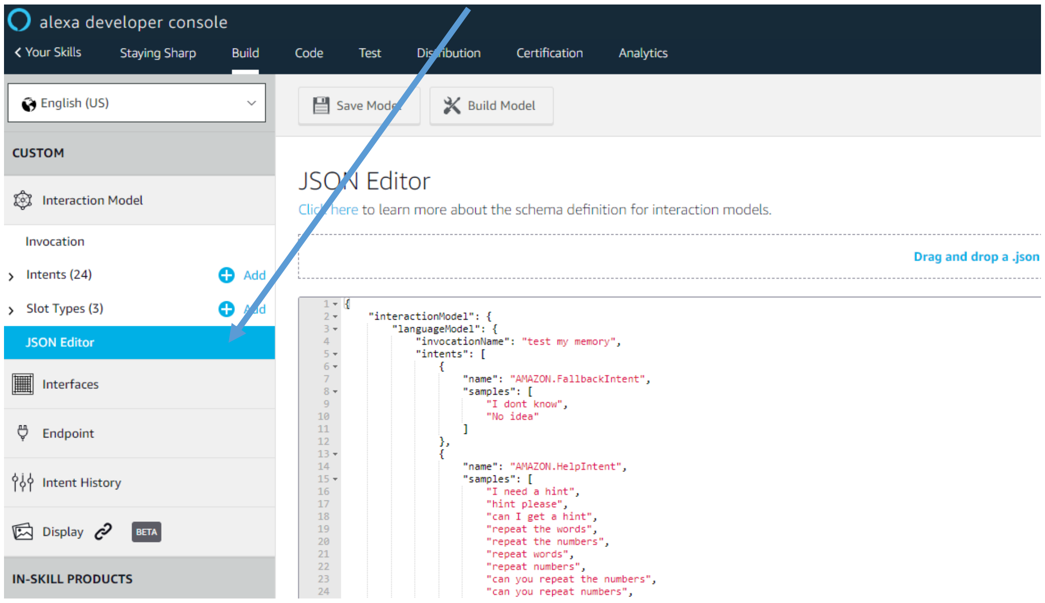
1. **Dialogues:** WELCOME_MESSAGE, HELP_MESSAGE, HELP_REPROMPT, STOP_MESSAGE, GAME_SELECTION_MESSAGE, INSTRUCTIONS_MESSAGE, CONFUSED_REPROMPT, wordTIPS, numTIPS, HELP_MESSAGE_NUMBERS, compliments, motivations
2. **Instructions:** Boolean to check if the program is coming from the instructions intent

3. **previousTipIndex:** The index of the previous tip in order to keep them dynamic
4. **transitionMessage:** Variable to store the fragments of the message that Alexa has yet to speak in order to maintain dialogue consistency
5. **lastQuestionMessage:** Similar to transitionMessage
6. **DEFAULT_WORDS_TO_REMEMBER_LENGTH:** Same as name
7. **DEFAULT_WORDS_TO_PICK_FROM_LENGTH:** Same as name
8. **MAX_NUMBER_TURNS:** Maximum number of rounds that can be played in the words game
9. **MAX_NUMBER_TURNS_CHUNKING:** Maximum number of rounds that can be played in the numbers game
10. **numCorrectAnswers:** Tracks number of correct answers overall
11. **num_CorrectWordAnswers:** Tracks number of correct word answers
12. **num_CorrectNumAnswers:** Tracks number of correct chunking exercise answers
13. **numIncorrectAnswers:** Tracks number of incorrect answers overall
14. **num_IncorrectWordAnswers:** Tracks number of incorrect word answers
15. **num_IncorrectNumAnswers:** Tracks number of incorrect chunking exercise answers
16. **gotchunkingQuestionCorrect:** Flag to check if the user got the chunking exercise question right
17. **isChunkingGame:** Flag to determine which type of game the user is in
18. **isWordGame:** Flag to determine which type of game the user is in
19. **leftoverChunkingRoundSpeech:** Variable to handle leftover fragments to maintain dialogue consistency
20. **percent_NumCorrect:** Chunking game score
21. **percent_WordCorrect:** Word game score
22. **card Object:** Logos for visuals
23. **tell_user_speech:** Used to restart the game by adding sorry I didn't catch that when user says something unexpected at initiation
24. **currentDifficulty:** Determines the current level of difficulty
25. **level:** Used to calculate difficulty
26. **round:** The number of rounds played within current game
27. **numTurns:** The number of total turns within a game
28. **chunkingWordLength:** Default number of words to remember
29. **wordsList:** List of three words that has one answer
30. **wordsToRemember:** List of (7/10/12) words that make up the answers to the questions
31. **exerciseSelectionFlag:** Determines whether an exercise has been selected or not
32. **randomNumber:** Same as Name
33. **isTip:** Whether the previous intent was tip intent or not
34. **finished(Words/Numbers)Game:** Status of games
35. **endgame:** True if both games have been played and game needs to end
36. **fallback:** True if the previous intent was the fallback intent

37. **wordsToPickFrom:** Same as wordsList
38. **setofInstructions:** Depending on context contains the instructions to be said
39. **levelupspeech:** Maintains continuity on increasing difficulty

7.1.2.2 Functions: Function Descriptions are within the code itself. Refer to 4.2 to access the code.

7.2 Front End: The front end can be copied straight from the json file and then pasted onto the Alexa Developer Console.



```
{
    "interactionModel": {
        "languageModel": {
            "invocationName": "test my memory",
            "intents": [
                {
                    "name": "AMAZON.FallbackIntent",
                    "samples": [
                        "I dont know",
                        "No idea"
                    ]
                },
                {
                    "name": "AMAZON.HelpIntent",
                    "samples": [
                        "I need a hint",
                        "hint please",
                        "can I get a hint",
                        "repeat the words",
                        "repeat the numbers",
                        "repeat words",
                        "repeat numbers",
                        "can you repeat the numbers",
                        "can you repeat numbers",
```

7.2.1 Explanation of Intents:

1. **Exercise Selection:** Used for determining the game type
2. **Start:** Used to start the game/invoke the skill
3. **UserNumberAnswer:** Takes in the numbers to determine whether the user answered correctly or not
4. **UserAnswer:** Takes in the words to determine whether the user answered correctly or not
5. **Tip:** Gives users tips around the game to improve their performance
6. **Restart:** Used for restarting the game
7. **Instructions:** Gives users context on the game as well as all the commands they can Alexa to control different aspects of the game
8. **Amazon.Fallback:** Handles the unhandled inputs when user says something the game isn't programmed for
9. **Amazon.Help:** Reminds users the original list in case they forget it
10. **Amazon.Stop:** Stops the program no matter the stage and tells users their score
11. **Amazon.NavigateHome:** Same as restart
12. **Amazon.Yes/No:** Handle yes and no dynamically based on the previous intent the game was in by making use of flags

8. Results & Limitations

By the end of the internship, The Tech Nest was able to create an infographic dashboard detailing every connection taking place on the AARP Network across their 3 major servers. After crawling through terabytes of data, some meaningful information was presented in the form of data visualized pie charts, graphs, maps etc. Some derived data displayed included connection specific details like connection type, log file type, data traffic, network protocols utilized, source and destination locations of the connections etc. Additionally, we made some filter tools to red flag connections by checking any suspicious connection timings, by comparing logs against Alexa's top 1 million websites to disregard commonly visited IPs (or domains), and by integrating a VirusTotal API to compare file signatures against known virus sources.

Some limitations of the project included the inability to operate on 'actual live' data, space limitations on the AWS EC2 instance, and inefficiency of the machine learning algorithms. The script took at least one hour to produce an output of suspicious network connection logs. The ELK Stack had to be refreshed multiple times before getting a stable output. More data requires more responsibility and storage space. After updating our storage space subscription of the AWS EC2 instance multiple times, we had to resort to clear old data in a cyclic fashion to create space on the AWS cloud. This also limited the amount of training data available for training supervised and unsupervised (clustering) machine learning algorithms, producing inaccurate results.

9. Setbacks and Challenges

We faced a few challenges in starting the development of the project as well as various setbacks that arose from the continuous feature updates that we implemented.

9.1 Empathizing with the User: One of the biggest challenges we faced, while trying to make the skill more user centric was trying to empathize and embody the user, in order to predict the kinds of dialogues they would say and how they would interact with the skill. Having mapped out various user inputs, we also had to go about acting on the inputs in context, a yes at different points in the game would mean different things, as well as classifying things into the type of help the user needed, i.e. they forgot the original list, they needed some context and instructions or they needed a tip.

9.2 Understanding Code Written by Others: Having taken over the project from a different team that worked on it previously, the first challenge that we faced was actually understanding the code well enough to make meaningful contributions of our own. Since there are multiple ways of approaching and solving the same problem, we had to align our implementation with the previous implementation in order to ensure that the code functioned properly.

9.3 Programming Language and Hardware Limitations: We had to work around a lot of challenges brought by JavaScript's asynchronous nature, using a lot of flags to Deliver game consistency, as well as limitations presented by Alexa's hardware and API, such as the time it take for the voice recognition to be processed, and the wait time between when Alexa says something and is ready to accept user inputs.

9.4 Bugs from Feature Updates: Continuously pushing various features which interacted with the rest of the game affected functionality in unpredictable ways as the feature updates would slot in between existing flow, managing these bugs and fixing them took considerable time and effort. A complete list of bugs has been provided at the end of the document.

10. Recommendations

First steps towards releasing the skill would be incorporating user feedback from a second Beta test and improving the program where possible. Post release, future potential improvements can include:

10.1 Updates: Continuous word releases to expand the game dictionary and keep users engaged and active.

10.2 Images for Screen devices: Integration with a photo generation API to show relevant images with words on devices with screens.

10.3 Fun Facts as hints: A library of associated facts that can be used when the user asks for a hint. For example, if the correct answer is Black Panther, the hint can be, "The correct answer is the same animal as the one called Bagheera in Rudyard Kipling's Jungle Book"

11. The Team

Product Sponsor: Kathy Washa
Product Lead: Julie Mackinnon
Student Contractors: Samarth Makhija and Rishabh Anand
Director, The Tech Nest: Miranda Kemp

12. Bug Log

12.1 Handling User input gracefully

Description: When prompted by Alexa to provide the answer for the words or the numbers exercise, the user must say "The answer is Softball". If the user said "Softball" instead, the skill will create an error.
Status: Fixed.
Methodology: The introduction of custom slots has allowed the user to just speak the answer without needing any other words in his/her reply.

12.2 Exercise Selection

Description: Sometimes when choosing between the two exercises, Alexa plays a round of the game on its own. We know this is happening because she starts the program by saying *"The correct answer was...."*
Status: Fixed.
Methodology: The bug was stemming from incorrect voice recognition, perhaps due to an accent. Alexa would recognize words as *"Violet"* or another word that she believed to be an answer. To handle this error gracefully, an `exerciseSelectionFlag` was introduced to check whether an exercise has been selected or not. If an exercise hasn't been selected, the words exercise cannot begin and the user would be prompted to choose again.

12.3 No FallbackIntent

Description: Upon receiving erroneous input, the skill would bug out due to the lack of a FallbackIntent.
Status: Fixed.
Methodology: A FallbackIntent was added which told the user, it didn't understand them.

12.4 "*Work with XYZ"* leads to the numbers exercise

Description: Any statement *"Work with {XYZ}"* leads to the numbers exercise.
Status: Fixed.
Methodology: Custom slots only accepting, numbers and words as a value were introduced.

12.5 Alexa says *"Undefined"* if the user is silent

Description: Alexa says *"Undefined"* if the user is silent when expected to provide an answer for the words or numbers exercise.
Status: Fixed.
Methodology: Reprompts where added to make the code `.listen(remprompt)`instead of `.listen()`.

12.6 "*Work with XYZ"* leads to the numbers exercise

Description: Any statement *"Work with {XYZ}"* leads to the numbers exercise.
Status: Fixed.
Methodology: Custom slots only accepting, numbers and words as a value were introduced.

12.7 Incorrect incrementing of difficulty

Description: If you initially answer correctly, then even if you answer incorrectly later on, it will increase the difficulty.
Status: Fixed.
Methodology: The variables were appropriately reset and kept intact.

12.8 Statistics aren't recorded if the game is switched

Description: The statistics from the first game are lost when switching games.
Status: Fixed.
Methodology: New global variables were introduced to preserve the statistics appropriately.

12.9 *"Correct answer was Let's…"*

Description: Alexa would say the *"correct answer was"* and follow it up with *"Let's end the game here.."*
Status: Fixed.
Methodology: The `numTurns` variable was appropriately decremented and reset.

12.10 Pronunciation for some words was incorrect

Description: Alexa would pronounce words such as "Amazing" incorrectly.
Status: Fixed.
Methodology: SSML tags were added and removed as required.

12.11 Only 4 words are repeated

Description: Alexa would only say 4 words upon calling `HelpIntent`
Status: Fixed.
Methodology: The correct variables were used.

12.12 Double Introduction Message

Description: Alexa would say the introductory message for the words or numbers game twice.
Status: Fixed.
Methodology: The `numTurns` variable was being incorrectly reset.

12.13 Game switches, no dialogue

Description: Although the game switches after the first game is complete, the introductory dialogue is not spoken.
Status: Fixed.
Methodology: A new `exerciseCompleted()` function was introduced to make all necessary changes to the exercise to implement bug free game switching.

12.14 Second game, Instructions and Help

Description: When transitioning from the first game to the next, the InstructionsIntent does not provide dynamic instructions specific to the current game. Specifically, when transitioning from the numbers exercise to the words exercise, the HelpIntent doesn't work, providing a list of numbers in the words exercise.
Status: Unfixed.

Methodology: various techniques such as the introduction of new Boolean variables, changes to the `resestExerciseVariables()` function and so on were tried to no avail. It was deduced that the `resestExerciseVariables()` was called somewhere incorrectly but no such logical flow was found.