Tools to Create and Democratize Conversational Artificial Intelligence

by

Jessica Van Brummelen

B.ASc., The University of British Columbia (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author . . . . . .
Department of Electrical Engineering and Computer Science
May 23, 2019

Certified by . . . . . .
Harold Abelson
Class of 1922 Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by . . . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Tools to Create and Democratize Conversational Artificial Intelligence

by

Jessica Van Brummelen

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2019, in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

The world is becoming increasingly saturated with voice-first technology, such as Amazon Alexa and Google Home devices. As this technology becomes more complex, the skill set needed to develop conversational AI applications increases as well. This work bridges the gap, democratizes AI technology, and empowers technology consumers to become technology developers. In this thesis, I develop block-based Alexa Skill programming tools, enabling anyone — even elementary school students — to create complex conversational AI applications. During high school workshops, students created Alexa Skills to help others remember forgotten words, learn math concepts, ease recycling, and display Alexa's speech on screen for those hard of hearing.

Additionally, I developed a conversational AI curriculum and partnered with MIT's High School Studies Program to provide workshops to the Boston community. We taught students about the capabilities, limitations, and implications of conversational AI, and explored research questions, such as "What do students believe, understand, and think about conversational agents?"; "Can students develop their own conversational AI applications?"; and "What do students envision for the future of conversational AI?"

The results from a pre-workshop assessment suggested that despite not understanding how conversational agents worked, students could think of ways for conversational agents to solve problems. The post-workshop assessment suggested that through the workshops, students learned conversational AI and machine learning concepts; could identify capabilities and limitations of conversational agents; felt proud of their project development; were interested in developing their projects further; and were generally hopeful and excited about the future of conversational AI. Through this research, students learned about the power and limitations of AI, were empowered to solve real-world problems using AI, and developed socially useful conversational AI agent applications.

Thesis Supervisor: Harold Abelson
Title: Class of 1922 Professor of Computer Science and Engineering

Acknowledgments

I would like to thank my advisor, Hal Abelson, who sowed seeds of creativity and empowerment, provided fertile App Inventor soil, and equipped me with tools to grow something new; Evan and Jeff, who pruned, debugged, and provided emergency water (even on weekends); Marisol, who watered day in and day out; and the rest of the incredible MIT App Inventor staff (past and present), who provided new idea-seeds, trellises of support, and sunlight on those dark Boston winter days.

Thanks to those at UBCO, esp. Dr. Najjaran & Mina: I wouldn't be here without you. Thanks also to Amazon for valuing teaching kids about the wonders/challenges of AI.

CC: thanks for constructing a multitude of blocks while fighting an seemingly unending battle of bugs, broken computers, and bafflingly complicated code. Tommy: thanks for adventurously traversing the depths of Alexa skills, LSTMs, and the especially-merciless Lambda Management Console. Terryn: thanks for investigating — with perceptiveness that would make Sherlock think twice — the numerous possibilities for conversational agent backends, new AI extensions, and alternative voice assistants. *Team App Inventor Alexa* (Julia, CC, Tommy, and Terryn): thanks for teaching and coding with enthusiasm, patience, and an unwavering positivity. Couldn't have done this without you.

Thanks also to the 5th Floor Heroes, who cheer each other on, round up the masses for debugging days, and fight for each other — even for those without a desk. J-team: thanks for the laughter, the late-night encouragement, and for not stealing my hockey skates.

To my friends far and near, thanks for laughing with me, crying with me, and givingmereasontonotdropoutofschoolevenwhenIfeellikeI'mgoingtobedeported (yes, I mean you, Judy, Jess, & Jen). I didn't know it was possible to feel so loved and supported. (I mean, who needs financial support when you've got friendo support, amiright?) Love you all.

Finally, I would like to thank my family. Despite being over four thousand kilometers away and being an absolute-wonderful-beautiful mess, your constant support, friendship, late-night FaceTimes, goofy humour (yes, I'm talking about those x-mazing zorillas), and love are why I'm here today. Thanks for instilling in me a passion for better education, a desire to help others, and a Canadian resilience to American weather. You're the best.

Contents

1 Introduction: Positively affecting communities through conversational artificial intelligence 15

1.1 Sheila's Storybook App and Jaidon's Conversational Entry into Karabo's World 16

1.2 Motivation and Background: Conversational AI is Everywhere 18

1.2.1 Motivational Examples 18

1.2.2 Background on MIT App Inventor 22

1.2.3 Motivation for Conversational Artificial Intelligence 24

1.3 Related Work: Conversational AI is a New Field 25

1.3.1 Currently Available Technology Democratization Tools 25

1.3.2 Recommendations from Research-based AI Democratization Tools 27

1.3.3 Conversational AI Principles 29

1.3.4 Artificial Intelligence and Computational Thinking in Education . 33

1.4 Contributions of this Research: Conversational AI Interface, Curriculum, and Workshop Results 37

2 Technical Implementation 38

2.1 Creating Sheila's Storybook App 39

2.2 Amazon Alexa Skills and Associated Terminology 42

2.3 MIT App Inventor Blocks for Conversational AI 46

2.4 Using the Interface 49

2.5 Interfacing with Amazon 51

2.5.1 Technical Details of the Send to and Get from App Blocks 57

2.5.2 Technical Details of the Generate Text (LSTM) Block 59
2.6 Technical Implementation Summary 62

3 Curriculum and Workshops **64**
3.1 Relation to Computational Thinking and Artificial Intelligence Dimensions 67

4 Results **71**
4.1 Pre-Questionnaire: Students appreciated that conversational agents could solve problems without fully understanding how the agents worked 72
4.1.1 Pre-questionnaire Likert Scale Question Analysis 73
4.1.2 Pre-questionnaire Short Answer Question Analysis 74
4.2 Workshop Activities: Students learned conversational AI concepts through remixing a storybook app and developing projects 76
4.2.1 Student Projects: Students developed conversational AI applications to address problems in their communities 80
4.3 Post-Questionnaire: Students felt successful after developing ambient assisted living and environmental related conversational agents 83
4.3.1 Post-questionnaire Likert Scale Question Analysis 83
4.3.2 Post-questionnaire Short Answer Question Analysis: 84

5 Discussion and Conclusions **89**
5.1 Students can make positive change through technology development . . . 89

6 Future Work: Interface Improvements, Natural Language Programming, and AI Education **92**

A Code **95**
A.1 JavaScript Endpoint Header 95
A.2 JavaScript Endpoint Footer 97

B Assent and Consent Forms for Human Subjects Research **98**
B.1 Assent Form 98
B.2 Consent Form 101

C Workshop Lesson Plans **109**

D Questionnaires **118**

D.1 Pre-questionnaire 118

D.2 Post-questionnaire 122

E Workshop Handouts **128**

E.1 Conversational AI Rules Worksheet 128

E.2 Storybook Worksheet 131

E.3 Storybook Solutions 136

E.4 Alexa Skill Development Tutorial Handout 141

E.5 Project Brainstorm: Individual Worksheet 150

E.6 Project Brainstorm: Group Worksheet 154

E.7 Running List of Website Links and Lesson Summaries 158

F Other Documentation **169**

F.1 Documentation for Testing Sessions 169

List of Figures

1-1 An example rendering of Sheila's storybook app. Modified from [78]. . . . 17

1-2 Speaking with Alexa contextually with Sheila's storybook. Modified from [78]. . . . 20

1-3 An example rendering of Miss Makinen's interactive textbook app. . . . 21

1-4 An example rendering of Harry's Forest Adventure app. . . . 22

1-5 The App Inventor interface, showing code blocks connected together to create a "Hello World" app. . . . 23

2-1 The app designer page, showing the development of a storybook application. Notice the Label, Image, and Button objects in the Components panel and how they appear visually in the Viewer panel. . . . 40

2-2 The MIT App Inventor Blocks page, showing the development of a storybook application. Notice the event blocks, including the *when screen initialize* block and *when button click* blocks. These blocks describe what happens when particular user events occur, and make function calls that set text and graphics on-screen to media described by the *initialize global variable* block. . . . 41

2-3 Adding a skill to an MIT App Inventor project. Notice the "Add Skill" button at the top of the workspace. . . . 42

2-4 Blocks defining the VUI for an Alexa Skill. Notice the *define intent* block. This block defines the utterances that the user can say to invoke the "tell the story intent". When this intent is spoken, the endpoint blocks linked to this intent will run. . . . 42

2-5 The blocks Sheila used to create the *whereZorillasLive* intent. Notice the *get slot* blocks. These are used as placeholders for words such as, "Africa" or the "United States". By using this block, Sheila does not have to list all possible places users might say. 43
2-6 Sheila's endpoint blocks for the *whereZorillasLive* intent. Notice the *get slot value* block. This block returns the value the user stated in the utterance, such as "Africa" or the "United States". 43
2-7 Blocks defining the endpoint function for the "tell the story" intent. When an utterance from the "tell the story" intent is spoken, these blocks will run, and Alexa will read the current sentence aloud. 43
2-8 The endpoint event for the *talkToKarabo* intent. These blocks cause Alexa to respond with a uniquely generated sentence that may sound similar to a sentence Dr. Seuss came up with. 44
2-9 The MIT App Inventor Alexa Skill designer page. Notice the *Login to Amazon* and *Generate endpoint JavaScript* buttons. 50
2-10 The MIT App Inventor Alexa Skill blocks page. Notice the *Voice* drawer on the left side of the screen in the Blocks panel. This is where the VUI and endpoint blocks reside. 51
2-11 The user workflow starting with creating an Alexa skill in MIT App Inventor and finishing with testing the skill on an Amazon device. Notice how the user never observes the JSON VUI code, as it is sent from MIT App Inventor to Amazon through the SMAPI interface. Note that the current implementation requires users to manually upload the generated JavaScript to an endpoint server. In future iterations, this will be automatically completed by MIT App Inventor, as described in Figure 2-13 and in Section 6. 52
2-12 The current architecture for the conversational AI interface. The green, orange, and blue arrows signify VUI-, endpoint-, and CloudDB-related communication respectively. 56

2-13 The desired architecture for the conversational AI interface. The green, orange, and blue arrows signify VUI-, endpoint-, and CloudDB-related communication respectively. Notice that in this case, the user does not need to manually upload data to AWS Lambda, as depicted by the orange arrows that do not interact with the computer (unlike in Figure 2-12). . . . 56
2-14 The *send to* and *get from app* blocks. These blocks are used in Alexa Skills to communicate with mobile apps. . . . 57
2-15 Communicating between mobile apps and Alexa Skills with Redis. Note that mobile apps can subscribe to changes in the Redis database, whereas Alexa Skills must initiate a request to a variable in order to determine whether it has changed. . . . 58
2-16 The *generate text* or *LSTM* block. This block generates sentences using an LSTM network pretrained for a certain number of epochs (shown in the second drop-down menu) on a specific corpora (shown in the first drop-down menu). The response is generated with respect to a *seed text* input. . 60
2-17 The communication between Alexa Skills and LSTM neural networks on a remote server. Alexa Skills can request uniquely generated sentences from pretrained LSTMs via GET requests. . . . 60
2-18 Example text generated character-by-character by the pretrained models used in the *generate text* block. The amount of training increases from left to right. Notice how the text generation becomes progressively better with more training, and how the text generated in (a) is clearly from a Nancy-Drew-trained model, whereas the text in (b) is clearly from a Dr.-Seuss-trained model. . . . 61
2-19 An alternative *generate text* block with the option to change the output length. In this case, the output length is set to 15 characters. . . . 62

4-1 A slide from the workshops about the unique capabilities of conversational agents. Students were encouraged to leverage these capabilities in their final projects. . . . 77

4-2 A slide from one of the workshops presenting some ideas for environmental applications. Students were encouraged to think of how conversational AI could help solve a problem they were passionate about related to the environment or ambient assisted living. 78
4-3 A slide from one of the workshops presenting some ideas for assisted living applications. Students were encouraged to think of conversational AI projects to help solve a problem related to the environment or ambient assisted living. 78
4-4 A slide from the workshops about the unique capabilities of mobile devices. Students were encouraged to leverage these capabilities in their final projects. 80
4-5 Example dialog from the *Synonym Finder* final project. This skill helped remind people of words they had forgotten. 82
4-6 The results of the pre- and post-questionnaires. The darker colored boxes are the post-questionnaire results. Notice that the minimum values as well as the means of the responses increased or remained the same from the post-questionnaire to the pre-questionnaire. Note that one student entered strongly agree (5) for every answer in the pre-questionnaire, despite his/her short answer responses not reflecting the same sentiment. Since there was a small sample size, this significantly skewed the pre-questionnaire results upwards. 84

List of Tables

1.1 The target user group, technology, and goals of various technology-democratizing tools. . . . 26
1.2 Symbolic-rules- and machine-learning-based examples of classification, prediction and generation concepts in AI. . . . 36

2.1 Conversational AI and Alexa skill terminology. . . . 44
2.2 VUI and endpoint blocks for Alexa Skills interface. Note that forest-green blocks are VUI blocks, and teal blocks are endpoint blocks. . . . 46
2.3 The code snippets that correspond to each block. The VUI (forest-green) blocks correspond to JSON key-value pairs, whereas the endpoint (teal) blocks correspond to JavaScript code snippets. Note that the "<input>" strings in the code snippets correspond to the input "cutouts" in the blocks (where input blocks can be placed). . . . 53

3.1 Workshop series overview. Each workshop consisted of a short lecture about conversational AI as well as time to work on unplugged activities or developing applications. . . . 65
3.2 Applicability of the computational concepts found in [10] and the proposed AI concepts, 'symbolic rules' and 'machine learning', to conversational AI. . . . 67
3.3 Implementation of the computational practices found in [10] and the proposed AI-related practice, 'training, testing, and validating', in the conversational AI curriculum. . . . 68

3.4 Applicability of the computational perspectives found in [10] and the proposed AI perspective, 'evaluating', to conversational AI. 69

4.1 The results of the Likert scale questions on the pre-questionnaire, where one corresponds to *strongly disagree* and five to *strongly agree*. In general, students had interacted with some sort of conversational agent, varied in their responses to whether they understood how they worked, and could think of how these agents could solve problems. Note that one student entered strongly agree (5) for every answer in the pre-questionnaire, despite his/her short answer responses not reflecting the same sentiment. Since there was a small sample size, this skewed the pre-questionnaire results upwards significantly. 73

4.2 The mean and range of the post-questionnaire Likert scale responses, where one corresponded to *strongly disagree* and five to *strongly agree*. Students generally felt they understood how conversational agents decide what to say, comfortable developing conversational AI applications, and their understanding of conversational agents increased through the workshops. . . 83

Listings

2.1 53

2.2 53

2.3 53

2.4 53

2.5 53

2.6 54

2.7 54

2.8 54

2.9 54

2.10 54

2.11 54

2.12 54

2.13 54

2.14 55

2.15 55

2.16 55

2.17 55

2.18 The `setCloud`DB function. It utilizes the Node.js Redis client to set values that may be accessed by MIT App Inventor mobile apps. 58

2.19 The `getCloud`DB function. It utilizes the Node.js Redis client to retrieve values that may be set by MIT App Inventor mobile apps. 58

2.20 The `getText` function for the *generate text* block. It utilizes the Node.js Fetch library to make HTTP GET requests. In Appendix A, this function is shown in the context of the endpoint header. 61
A.1 The header of the Node.js Javascript file that defines the Alexa Skill endpoint. It includes constants, functions, and import statements to enable specific endpoint block functionalities. 95
A.2 The footer of the Node.js Javascript file that defines the Alexa Skill endpoint. It includes the 'unhandled' callback and the handler export function. 97

Chapter 1

Introduction: Positively affecting communities through conversational artificial intelligence

"Technology is not just a tool. It can give learners a voice that they may not have had before."
— George Couros

Voice-based technology is rapidly permeating Americans' daily lives. Vehicles are explaining how to get to the nearest grocery store, thermostats are being told to turn up the heat, and microwaves are being commanded to prepare people's meals [81]. Companies, such as Amazon, Google, and Apple, are developing voice-first devices to ease data access, automation, and natural interaction with technology, and the general American population is rapidly consuming them [67, 64]. According to a 2017 survey, nearly fifty percent of American adults use voice assistants, and according to a 2018 survey, over twenty percent of American adults own at least one smart speaker [16, 67].

As permeation of such technology increases, is important to consider ethical questions. For instance, do the developers have the best interests of all in mind? If the general population were developing such artificial intelligence (AI) technology, would they have different goals? How does this imbalance in power affect how AI develops? Researchers

have concluded democratizing this technology such that the general population — rather than a biased subset of the population — can develop AI applications would provide a more broadly beneficial impact [62]. If we are serious about democratizing the development of AI, development tools should be easily accessible to everyone in terms of availability as well as usability.

In this thesis, I present an intuitive, online visual coding interface to enable anyone to develop conversational AI applications. I also present high school workshop curriculum to teach students how to use the interface and about conversational agents. In the workshops, students developed conversational AI projects to address problems in their communities, including memory difficulties, hearing loss, and ineffective recycling programs. These projects and curriculum are described in Sections 3 and 4. In this section, we present a fictional character named Jaidon to illustrate the interface's capabilities. Jaidon is interacting with a conversational agent developed by his cousin, Sheila.

1.1 Sheila's Storybook App and Jaidon's Conversational Entry into Karabo's World

> Jaidon flipped through colorful pages on his tablet. One of the animals on the screen looked strikingly similar to a skunk. He blurted out, "Alexa, why do they call that animal a zorilla, not a skunk?"
>
> A voice from his smart speaker replied, "Karabo isn't actually a skunk, he's a zorilla!"
>
> Jaidon kept questioning. "Alexa, what's the difference between a skunk and a zorilla?"
>
> "One difference is that skunks eat plants *and* meat, whereas zorillas only eat meat. Surprisingly, zorillas are more closely related to weasels than skunks!"
>
> "Alexa, do zorillas live in the United States?"
>
> "Actually, zorillas don't live in the United States. They mostly live in Africa."
>
> "Oh, that's why I've never heard of them!"
>
> Satisfied, Jaidon flipped to the next page. Karabo was spraying a nasty looking yellow cloud at an owl. "Alexa, are zorillas smellier than skunks?"
>
> "I don't know that one."
>
> "Alexa, can I ask Karabo who's smellier?"
>
> "Sure, here's what Karabo says: 'So smelled and he shook. We sat in the hat.' Karabo doesn't know how to speak *human* very well!"

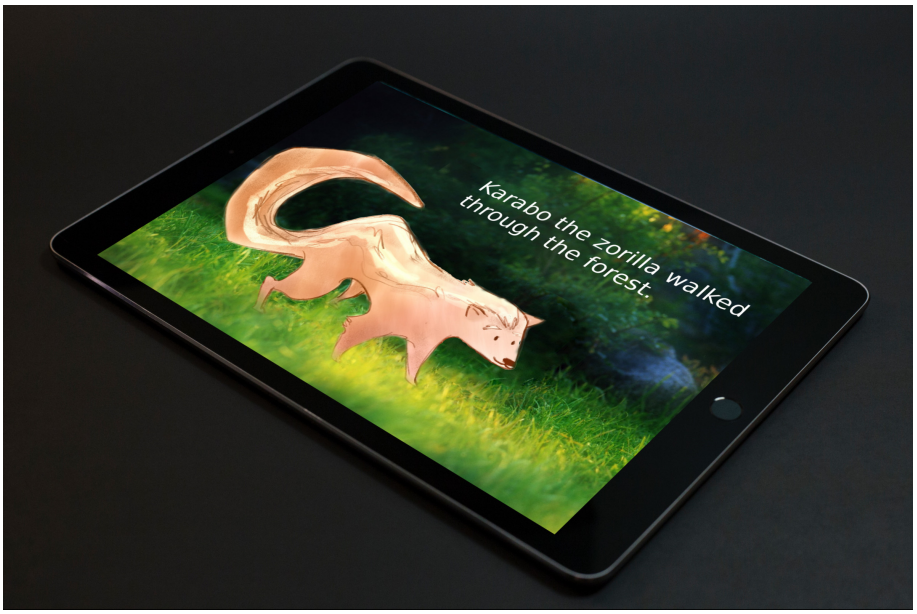


Figure 1-1: An example rendering of Sheila's storybook app. Modified from [78].

> Jaidon laughed and decided to ask his cousin, Sheila about zorilla-smelliness later on. On the next page, Karabo was sniffing the ground.
>
> "Alexa, Karabo looks hungry. Can I feed him?"
>
> "Aye aye! Teleporting food to Karabo, right away!"
>
> A mouse suddenly appeared on the storybook page. Shocked, Jaidon asked Alexa if Karabo ate mice. She politely explained that as a carnivorous zorilla, Karabo ate all sorts of small rodents.
>
> "I guess Karabo's kind of like a cat!"
>
> After pondering zorilla culture for a while, Jaidon called Sheila to ask about zorilla smelliness, to which Sheila replied, "I'm not sure if zorillas or skunks are smellier! But I bet Alexa will know in a few minutes..."
>
> After a few minutes of clicking and typing away at her computer, Sheila told Jaidon to try asking Alexa about zorilla-smelliness again. This time Alexa responded, "Although I don't have a nose myself, according to my research, skunks are much larger than zorillas, and I'd guess that a bigger animal means a bigger stench!"
>
> That settled it for Jaidon. He would choose a zorilla over a skunk, any day!

Like the fictional Sheila, real students also developed Alexa skills during high school workshops at MIT. The remainder of this thesis describes tools that enable students to develop such conversational agents. Furthermore, it describes the curriculum, data collection, and results from the workshops.

1.2 Motivation and Background: Conversational AI is Everywhere

> *"What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from any single, perfect principle" [57]*
>
> — Marvin Minsky

As AI, robotics, and autonomous systems become more common, the need for AI education also increases. Knowing the capabilities and limitations of autonomous vehicle AI, for example, is extremely important in judging the amount of human supervision required for safety [77]. Similarly, knowing whether you are speaking with a human or a humanoid is important for good decision making. For example, should you trust the "person" who is speaking to you over the phone with your social security number? In a world with scammers, hackers, and incredibly realistic (and near-ubiquitous) AI [85, 51], the need for AI education is strikingly apparent.

To address such a significant need, we should use equally effective educational techniques. One such educational technique is *project-based learning*, which has been shown to increase the deepness of students' learning and understanding, as well as motivation to learn [70, 5]. This research implements project-based learning through workshops in which high school students create their own conversational AI projects. Through data collection during the workshops, research questions, such as whether high school students can learn about the capabilities and limitations of conversational AI systems, and gain skills useful for addressing challenges in an increasingly AI-filled world, are explored.

1.2.1 Motivational Examples

To illustrate the capabilities of the conversational AI interface in MIT App Inventor, we delve into the details of Sheila's storybook app (which was introduced in Section 1) and present additional examples of people developing conversational AI applications. These

include Miss Makinen, a fictitious high school teacher, and Harry, a fictitious middle-school student, who both have ideas for their own conversational AI apps. Section 2.1 further explains how Sheila's app may be created in MIT App Inventor.

Sheila's Storybook App

Sheila loves stories. When she was younger, she imagined jumping into the pages of her storybook and interacting with the characters. When she heard about MIT App Inventor and the Alexa Skills interface during a seventh grade computer lesson, she had a brilliant idea: to create a talking storybook. The storybook would be about zorillas, little-known animals that appear to be skunks, but are actually *striped polecats* from South Africa. Sheila would create the app using MIT App Inventor and run it on her tablet. The app would have a couple of main features:

- You could swipe through "pages" of the storybook while reading and viewing illustrations on-screen
- You could ask Alexa about the characters, setting, and narrative
- You could ask Alexa to read you the story, and as Alexa reads, the sentence on the app's page would be highlighted
- You could have "conversations" with the storybook characters
  - For example, when you ask a character a question, a response would be automatically generated

Through MIT App Inventor's Alexa skills interface, Sheila could make her interactive storybook a reality. Her (imagined) development process is described in Section 2.1.

Miss Makinen's Textbook App

Miss Makinen, a math teacher, had a similar idea to Sheila: She wished books — specifically math textbooks — could magically speak to her students. Sometimes it felt like there

Figure 1-2: Speaking with Alexa contextually with Sheila's storybook. Modified from [78].

were two types of students in her classroom: those who were too afraid to read the textbook (and would constantly be talking to her) and those who were too afraid to come talk to her (and would get stuck on questions that she could easily help explain). She envisioned textbooks talking with the vocal students (who did not enjoy reading), giving Miss Makinen extra time to interact with students who could use an extra nudge. It would be a talking-textbook-time-balancer.

The talking textbook's main features would be:

- The app could be downloaded by anyone, taken home, and spoken with
- You could ask Alexa to explain equations presented in the text
- You could ask Alexa to explain more about a certain topic, or explain something in a more down-to-earth way than the textbook read
- You could ask Alexa for hints about how to start a problem
- If Alexa did not have the answer to a question, it would notify Miss Makinen to come help

Figure 1-3: An example rendering of Miss Makinen's interactive textbook app.

Miss Makinen would gradually create and add features to the talking textbook throughout the school year. When she noticed that students kept asking similar questions, she would program Alexa to respond to that question. With time, the textbook would become increasingly interactive. Miss Makinen could even share the app with other math teachers, and ask them to add information to the app as they went through their curriculum too. Miss Makinen imagined the talking textbook growing to become an essential learning tool, spreading to classrooms across the globe.

Harry's Forest Adventure App

Harry also had a dream: to discover the world and its secrets. As a fifth grade student living on the edge of Stanley Park, a large forested area surrounded by the Pacific waters of Vancouver, he loved to explore nature. Harry wanted to create an interactive, intelligent guide for the park, so that when his family did a house swap, the Australian family's daughter, Kit, could discover Harry's favorite spots. Kit heard about his idea and started

creating a similar app for Royal National Park in Australia too.

Harry's application would have the following features:

- As Kit walked through the park, Alexa (on her mobile device) would guide her through the forest and talk about Harry's favorite places
- A map would be shown on screen, and Kit could ask Alexa about pin-pointed areas
- Kit could ask Alexa to add her current location to the *favorite places* list

Harry and Kit swapped apps and discovered each others favorite places from worlds apart.

Figure 1-4: An example rendering of Harry's Forest Adventure app.

1.2.2 Background on MIT App Inventor

In the past, developing technology such as mobile applications, autonomous systems, and AI could only be achieved by highly specialized engineers and computer scientists. Now,

however, there are tools, such as MIT App Inventor, available for anyone to develop complex technological systems. App Inventor aims to empower anyone to learn how to code, develop computational thinking skills, and create his or her own mobile apps. As of 2018, MIT App Inventor had over 7.9 million unique users from 195 different countries who built over 24 million different apps [60]. Using App Inventor, people have developed apps to collect research data, help visually impaired classmates navigate halls, and track mood to encourage users to seek community support when they need it [56, 61].

App inventor uses *block-based coding*, in which users connect puzzle-piece-like blocks to develop fully functional computer programs, as shown in Figure 1-5. Block-based coding simplifies the programming process, empowering anyone, including primary school students, to create their own fully-functional computer programs. Specifically, the MIT App Inventor project enables people to develop complex and cutting-edge technology, such as mobile apps, Internet of Things (IoT) connected devices, and AI agents. MIT App Inventor aims to democratize technology development, giving everyone access to today's powerful technological tools.

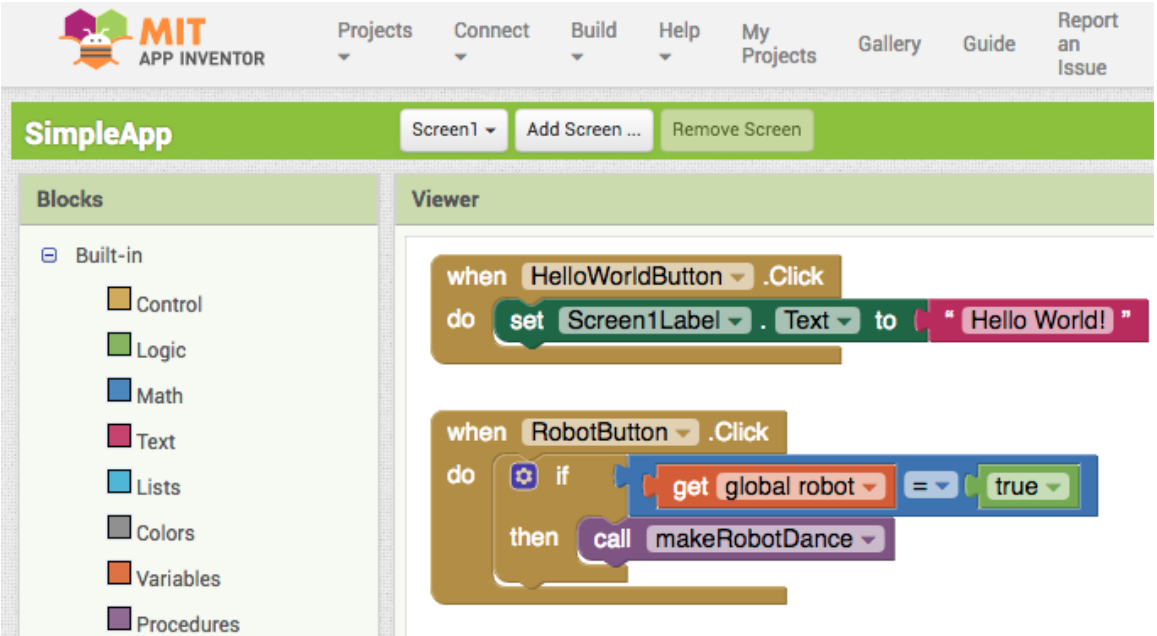


Figure 1-5: The App Inventor interface, showing code blocks connected together to create a "Hello World" app.

1.2.3 Motivation for Conversational Artificial Intelligence

> *"[Intelligence is the] mental quality that consists of the abilities to learn from experience, adapt to new situations, understand and handle abstract concepts, and use knowledge to manipulate one's environment." [71]*
>
> — Robert J. Sternberg, *Encyclopaedia Brittannica*

> *"[Artificial Intelligence is] the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings." [18]*
>
> — B.J. Copeland, *Encyclopaedia Brittannica*

As technology becomes increasingly "smart", the line between human and humanoid becomes increasingly difficult to discern. For example, consider Google's conversational AI appointment scheduler, "Google Duplex" [51]. During the 2018 Google I/O conference, Google Duplex scheduled appointments over the phone with frightening anthropomorphic intonation, pauses, and conversational flow [85]. This caused some to believe that Google Duplex passed the Turing test [28, 73], although others disagreed [27, 79]. Either way, the implications of such human-like conversational agents are considerable. One can imagine frightening situations with deceptive conversational agents chatting over the phone, impersonating family members or authority figures.

In all likelihood, today's young people will have to address these types of problems in the future. Thus, this research creates tools, including a conversational AI development interface and curriculum, to equip students with a greater understanding of AI and help them better address these problems. This thesis discusses tools and research related to this goal.

1.3 Related Work: Conversational AI is a New Field

1.3.1 Currently Available Technology Democratization Tools

Similar tools to MIT App Inventor in terms of technology democratization include MIT's Scratch, Dexter, If This Then That (IFTTT), Google's AIY, the Amazon Alexa Skills Kit (ASK), and Amazon Alexa Skill Blueprints [55, 65, 69, 29, 49, 40]. Each of these tools simplify complex technology development. For example, Scratch simplifies programming through block-based coding [55]. Dexter simplifies conversational AI through a straightforward chatbot development platform. IFTTT simplifies programming and Internet of Things (IoT) development through if-then statements and straightforward connectivity [65]. Google's AIY enables development of hardware with AI capabilities, such as a speaker connected to the Google assistant API [29]. The ASK simplifies Voice User Interface (VUI) development with an SDK for Amazon's voice-based, virtual assistant, Alexa [49]. Alexa Skill Blueprints further simplify VUI development through fill-in-the-blank guides for Alexa Skills [40]. Each of these tools have specific niches in terms of users and technology. Table 1.1 outlines each tool's target user group and technological area.

As shown in Table 1.1, MIT App Inventor's target users and goals overlap with other tools' niches. For example, MIT App Inventor and Scratch both aim to teach Electrical Engineering and Computer Science (EECS) skills to young people. Nonetheless, App Inventor's uniqueness lies in its goal to allow anyone to develop their own fully-functional smartphone and tablet applications free of cost. Furthermore, this research enables App Inventor users to develop conversational agents, providing avenues for learning high-level AI concepts, and furthering App Inventor's goal of democratizing state-of-the-art technology development.

Table 1.1: The target user group, technology, and goals of various technology-democratizing tools.

Tool	**Target Users**	**Target Technology**	**Goals**
MIT's Scratch [55, 72]	Ages 8-16	Online computer programs such as games and animations	To teach programming and CT skills
Dexter [69]	Businesses and inexperienced programmers	Online chatbots, including Twitter and Alexa Skill bots	To simplify the development of automated conversations
IFTTT [65]	Smartphone and other smart-device users	Smart devices and the IoT	To allow easy connection between electronic devices
Google's AIY [29]	Makers	Microcontroller kits from Google	To enable makers to easily incorporate AI APIs into specific hardware
Amazon ASK [49]	Programmers	Alexa-enabled devices and the IoT	To allow developers to create voice-enabled applications (known as "skills") for Alexa-enabled devices
Amazon Alexa Blueprints [40]	Non-programmers	Alexa-enabled devices	To enable non-programmers and others to create specific types of voice-enabled applications for Alexa-enabled devices
MIT App Inventor [60]	Primary school students to adult learners	Smartphone and tablet applications	To teach programming and CT skills, and to enable anyone to develop apps

1.3.2 Recommendations from Research-based AI Democratization Tools

"Research is formalized curiosity. It is poking and prying with a purpose."

— Zora Neale Hurston

Other researchers have also developed tools to teach students high-level AI concepts. For example, *Cognimates*, a platform in which students can program and engage with AI agents, was developed to investigate children's perception of AI and teach them AI concepts. This work also explored how children currently perceive and interact with AI. In particular, it found that many older children believed Amazon Alexa was smarter than they were [22]. Another study showed that children often ascribe human characteristics to conversational AI technology [7]. These notions further reveal the need for conversational AI education.

PopBots, customizable mobile-phone based robots, were also developed to teach AI concepts [82]. These robots could be programmed using block-based coding, and customized physically with *LEGO Duplo* blocks. Through the *PopBots* curriculum, students learned about three main AI concepts: (1) rule-based systems, (2) supervised machine learning, and (3) generative AI. They found that young children (preschool- and kindergarten-aged) could most easily understand rule-based systems, and had the most difficulty understanding generative AI [82]. Thus, in MIT App Inventor's AI curriculum, we include explanations and examples of generative and rule-based AI, and include a survey question to determine the effectiveness of these explanations.

The authors of both *Cognimates* and *PopBots* provide recommendations on developing successful AI literacy curriculum. They both agree that AI tools should be transparent, trainable, and personalizable [21, 82]. Specifically, students should be able to observe relationships between their own thinking and the AI agents' algorithms, teach the AI agents, and design their own algorithms for the AI agents. These points are directly addressed through MIT App Inventor's block-based interface, which makes algorithms highly visible, alterable, and personalizable through visual coding.

The *Cognimates* and *PopBots* authors also recommend encouraging collaboration and discussion between students, emphasizing how machines learn and why that is significant, and engaging the students with fun examples and showing personal excitement for STEM [21, 82]. In my curriculum, I address these recommendations through team-based projects encouraging collaboration, emphasizing the different ways in which machines learn (rule- and machine learning-based AI) with real-world examples, and sharing excitement for STEM through personal projects using the conversational AI interface.

The *Cognimates* research also suggests that AI technology should be transparent and not attempt to seem more intelligent than it actually is [21]. The conversational AI interface addresses this through enabling students to observe and engage with how conversational AI technology is actually developed. Through the programming interface, students are given full control of Alexa's responses, and by developing purposeful and functional conversational AI projects, the students engage with the reality of Alexa's "intelligence". With such project development, there is little room for Alexa to seem overly intelligent.

Another suggestion from *Cognimates* includes enabling students to compare and contrast AI to human intelligence [21]. For example, questioning whether the AI generates unique responses, like humans, or gives canned, unintelligent responses. In our curriculum, this concept was explored through the *generate text* block (as described in Section 2.5.2), which generates unique responses that become more intelligible as better-trained networks are used, and rule-based blocks, such as if statement blocks. In the workshops, instructors provided students with a description of how machines may learn (e.g., through direct programming of rules, or generative machine learning) and provided time for students to explore these concepts in the interface.

Across the globe, AI curriculum is being developed and conducted as well. For example, *Snap!* AI cloud services were developed at the University of Oxford to teach students about current machine learning systems [45]; *cs4fn* developed teaching resources that discuss machine learning in the context of future developments [6, 19]; and *AIinSchools* discusses the future of AI and machine learning, emphasizing deep learning and neural networks [17]. Other examples include middle school machine learning resources in the Wolfram language [84]; *ai4k12*, open-source AI curriculum for K-12 [76]; and *Apps for*

Good, AI curriculum in the context of positive change [25]. Evidently, AI curriculum research is growing at a remarkable pace. This work aims to provide tools and teaching resources to be used in curricula such as these, focusing on conversational applications and project-based learning.

1.3.3 Conversational AI Principles

This work aims to teach students AI principles, and more specifically, conversational AI principles. Conversational AI is a relatively new field of research, and has not received nearly as much attention as other AI fields, such as computer vision [80]. This work aims to distill current research in the field, and share this knowledge with students through interactive, engaging curriculum. The following section describes conversational AI principles, including voice-first design principles and fundamental concepts.[1]

Before designing voice-based systems, it is important to note that effective voice user interfaces (VUIs) are designed differently from graphical user interfaces (GUIs). With GUIs, content is easily accessible and can be viewed at any time; whereas with VUIs, the user must either remember from previous experience what the interface can and cannot do, or ask the interface, which takes time and can be frustrating. Conversational agents should be flexible in what they understand to prevent users from having to memorize things and minimize users' cognitive loads. I call this the *Flexibility* principle.

Amazon and Google both have frameworks pointing out common VUI design problems, such as the memorization problem [42, 30]. Both companies borrow design principles from Grice, a linguist who developed four "conversational maxims" [20, 31, 33]. Grice's maxims (although not necessarily initially intended for VUI design) provide an excellent basis for effective communication, and transfer well to conversational AI design. The four principles focus on quantity, quality, relation, and manner. Essentially, a good remark or response in conversation should (1) share a sufficient and concise amount

[1]Note that although I wanted to include conversational AI design principles in the presented curriculum, due to the vast amount of information already involved, I focused on the fundamental principles. This included (1) how to program, (2) what conversational AI is, (3) how to program conversational agents, and (4) other concepts related to the conversational AI blocks, such as how data is stored and accessed in the cloud. Nonetheless, I plan to include voice-first design principles in future curricula.

of information (quantity), (2) be correct and well-grounded (quality), (3) be relevant to the conversation (relation), and (4) be logical, unambiguous, and natural-sounding (manner). These principles are often referenced in VUI design literature [35, 23]. I refer to them as the (1) *Concise*, (2) *Correct*, (3) *Relevant*, and (4) *Natural* VUI design principles

Another design concept includes the difference between goal-oriented and social agents. Goal-oriented conversational agents are designed to efficiently achieve specific goals, whereas social conversational agents are designed to engage users in interesting conversation. Generally speaking, current commercial AI agents, including Amazon Alexa, Apple's Siri, and Google Home, are goal-oriented [52]. For example, instead of spontaneously conversing with users, these agents wait for users to initiate intents, determine the task the user wants to complete, and perform a relevant routine. In general, goal-oriented agents do not ask questions unrelated to a specific task, such as whether the user had a good sleep last night or what the user learned today. Conversely, social agents ask such questions to glean information about users, which may be used to do a better or more engaging job of reaching users' goals at a later time [2].

Goal-oriented agents also do not generally fill an emotional need or build trust with the user [8, 2], whereas social agents are designed with emotion, trust, and social experience in mind [66, 9]. For example, some social agents have been developed to take on therapeutic roles for children with autism or the elderly [15, 11], whereas goal-oriented agents may be designed to efficiently keep track of tasks needed to be completed in the workplace, for instance. When designing an agent, developers should consider whether the purpose of their agent is to complete a specific task, fulfill a social-emotional need, or do both. I refer to this consideration as the *Social Extent* principle: Does the end user want to engage socially with the agent or complete a job efficiently?

Other conversational AI principles focus less on voice-based design and more on general AI theory and ethics. Our goal with the workshop curriculum is to provide a basis for AI literacy and ethical AI development. Thus we focus on these concepts. Students are nonetheless encouraged to investigate other aspects of conversational AI including the design principles.

One fundamental AI concept discussed in the workshops includes the difference be-

tween symbolic rule-based AI or Good Old-Fashioned AI (GOFAI), and machine learning- or neural network-based AI. In symbolic rule-based AI, collections of if-then statements or other rules determine how AI agents behave [83, 14]. For example, a programmer might define a rule such as, *If the user says, "Turn on the light", then send an electrical signal to the light.* This rule-based method of AI has shortcomings. In particular, explicitly programming all possible rules for an AI agent would be exhausting. Thus, researchers have determined ways to enable machines to "learn" [83]. Although learning methods have other shortcomings, such as being difficult to interpret and requiring large amounts of data to train, they provide programmers with methods to automate rule development [14]. I refer to the concept of rule- versus machine learning-based development as the *Direct Definition* principle, which I discuss further in Section 1.3.4.

Since conversational devices are necessarily designed to interact with conversational organisms (i.e., humans), it is imperative to design such systems ethically. As more AI systems are deployed, more instances of ethically-questionable AI are being discovered. For instance, Northpointe AI software has been used in court to examine the likelihood of a criminal to re-offend. In theory, having computers help sentence criminals could reduce judicial bias; however, it was found that Northpointe's software tended to assign higher risk scores to African Americans than to Caucasians. This was despite not using racial data to develop the software [86, 24]. Another example includes the gender and racial bias in many AI facial recognition systems [13].

Evidently, ethical ambiguity and bias can creep into AI systems. When gone unchecked, they have serious consequences in the real world. In our workshops, we considered questions of ethics, including safety and privacy concerns of conversational agents, and developing ethical, socially useful final projects. Results from a post-workshop questionnaire revealed students' concern and regard for such issues, as did their final projects, as discussed in Section 4. I refer to considering and integrating ethics before, during and after AI system design as the *Ethical* principle.

To reiterate, the conversational AI principles discussed include,

1. **The Flexibility Principle**

- The agent should minimize the user's cognitive load by enabling flexible communication (e.g., understanding synonyms, enabling commands at any time during conversation, etc.).

2. **The Concise Principle**
    - The agent should not overshare or force users to wait on lengthy responses.
3. **The Correct Principle**
    - The agent should not under-share such that the information is no longer correct. The information shared should be well-grounded and complete.
4. **The Relevant Principle**
    - Although information might be factually correct, it may not be relevant to the current conversation. The agent should respond directly to the user's requests.
5. **The Natural Principle**
    - Information sharing and conversation should flow naturally between the agent and user. The agent should use appropriate diction (e.g., in formality) for the situation and should understand users' natural speech and vocabulary.
6. **The Social Extent Principle**
    - The agent should engage with an appropriate level of efficiency. Depending on the context, relational and emotional responses may or may not be appropriate.
7. **The Direct Definition Principle**
    - When deciding whether to use rule- and/or machine learning-based methods to create conversational agents, developers should ensure the rest of the principles are upheld. This may involve constraining machine learning-based responses to uphold conciseness or replacing a rule-based response with a machine learning-based response to uphold flexibility.

8. **The Ethical Principle**

    - Ethics should be integrated into the design process before, during and after deployment.

In our workshops, we engaged students in discussion about the future of conversational AI. Current commercial conversational agents have many limitations. In general, they do not have a deep understanding or representation of the world and its concepts. For example, when you ask Siri to order a latte, it may respond with a list of nearby vendors instead of ordering the latte itself [52]. Furthermore, conversational agents generally cannot learn on the fly or be taught by end users [52]. Agents are also limited in their abilities to have engaging, natural conversation; to have contextual memories for follow-up questions and conversations; and to have emotional understanding to provide relational and therapeutic support [47]. Students picked up on these limitations when responding to the workshop questionnaires, as discussed in Section 4. With increased consideration of principles discussed here, these limitations will likely decrease and conversational agents will become more useful, natural-sounding, and enjoyable to engage with.

1.3.4 Artificial Intelligence and Computational Thinking in Education

One of App Inventor's goals is to teach students computational thinking (CT) skills. These skills are valuable in today's workplaces, especially as the number of computer science and STEM related jobs increases [34, 63, 4]. Furthermore, CT skills, including logic, abstraction, and creativity, are useful in many aspects of life, and thus valuable skills for everyone to learn [34, 4]. I developed the conversational AI interface to teach students these skills, as well as skills necessary for AI development.

To study students' CT skills with respect to the interface, I developed curriculum using Brennan and Resnick [10]'s CT framework. This framework defines computational thinking in terms of three main dimensions: computational concepts, practices, and perspectives. In Section 3, the Tables 3.2, 3.3, and 3.4 relate these dimensions to our curriculum. Furthermore, to structure the AI components of our curriculum, we introduce five

principal AI concepts, practices, and perspectives: classification, prediction, generation, training/validating/testing, and evaluating. Specific examples of these components in the literature are shown in Table 1.2 in the context of the symbolic rule / machine learning paradigm.

Although AI is oftentimes equated (erroneously) to machine learning, there are other forms of AI. These can be understood within the AI symbolic rule / machine learning paradigm. In symbolic rule-based AI, collections of if-then statements or other rules determine how AI agents behave. In machine learning-based AI, machines determine how to behave through extracting patterns [83]. Both methods have shortcomings, such as the difficulty of programming an exhaustive list of rules for rule-based AI, and the limited interpretability of machine learning models [14].

Within the symbolic rule / machine learning paradigm, designers use AI to classify, predict, and generate information. For example, a conversational agent may classify utterances as positive or negative, predict the user's intent, and generate a new response. These concepts provide a basis for understanding what AI agents can accomplish. Thus, we integrate them into our curriculum and propose adding Classification, Prediction, and Generation as AI-related concepts to Brennan and Resnick [10]'s CT framework. We define these concepts as follows.

Classification. Machines often sort information into categories for downstream decision-making through rules (e.g., "the sentence is positive because it contains 'happy'") or learning algorithms (e.g., after observing "positive" sentences, similar sentences are classified as "positive").

Prediction. To act intelligently, machines predict future values and behavior. This includes predicting the category an object may fall into, an object's future behavior, or the best action to take next (e.g., after saying, "I am a", a conversational agent may predict the next best word to be "robot").

Generation. Using information gathered, machines can generate new data. This may include synthesizing previous examples, creating new information, or making decisions (e.g., a machine constructing and speaking a new sentence).

A common practice when developing AI systems includes training, validating and

testing. In the conversational AI curriculum, students can choose the amount of training a neural network has undergone, and will have to validate their agents' accuracy and ability to meet their needs. We propose adding Training/Validating/Testing to Brennan and Resnick [10]'s CT framework, as defined below.

Training, Validating, and Testing. Developing a robust ML model requires waiting for the model to learn to recognize patterns, testing if it generates correct predictions, and determining if it is sufficient for the task. Training involves providing examples (or an environment) for the model to iteratively learn from (or experiment in). Testing and validating involves providing different examples (or environments) to observe how the model behaves, comparing the model's behavior to other models, and determining whether the model is sufficient. This includes assessing the accuracy of the model (e.g., the percentage of correct classifications) using test/validation datasets and using the model loss (which is a value used to update model weights during training).

Finally, we propose the following AI-related perspective:

Evaluating. Some AI's (e.g., neural networks and other learning techniques') behavior can be difficult to predict or unintuitive to humans. Programmers must think about how well the program behaves and whether it achieves the necessary goals (e.g., How can we improve the program? Did we over- or under-train the model? Is the model biased towards certain people?). These considerations are especially pertinent when considering the large number of input-output relationships with ML.

Note that although evaluating and validating/testing may seem similar, evaluating is performed in the context of the final product or application, whereas testing and validating are performed only considering the model itself. For example, when evaluating, one might ask, "Is my app biased towards classifying people as middle-aged?"; whereas when validating, one may ask, "Why is my model achieving 42% accuracy?".

The conversational AI curriculum provides a platform for asking students these questions and introducing the idea of evaluating projects through a practical as well as ethical lens. It also provides opportunities for students to investigate classification, prediction, generation, and training/validating/testing. In Section 3.1, I relate the AI and CT dimensions to the conversational AI high school workshop curriculum. The relations are shown

Table 1.2: Symbolic-rules- and machine-learning-based examples of classification, prediction and generation concepts in AI.

	Symbolic Rules	Machine Learning
Classification	*Practical Reasoning for very expressive description logics* categorizes knowledge as "satisfiable" or "contradictory", and as more or less general than other knowledge. It does this through representing knowledge using formal logic and generating proofs [37].	*ImageNet classification* uses a deep neural network to group images into particular categories. For example, it may label images as "leopard", "mushroom", "lifeboat", etc. depending on the content of the image [48].
Prediction	*Expressive probabilistic description logics* use symbolic rules to describe the Semantic Web (an extension of the World Wide Web) as probabilistic knowledge. This allows computers to predict the meaning of and relationships between different knowledge on the Web [54].	Neural network load forecasting with weather ensemble predictions uses a neural network to predict future scenarios of a weather variable [74].
Generation	*The Rete Match Algorithm* finds matches between patterns and objects efficiently on a large scale. It does so through symbolic rules, a tree-structured sorting index, and storing object state data [26].	*Deep photo style transfer* uses a deep neural network to generate an image in the style of another image, such as generate a nighttime version of an image by transferring the "nighttime style" onto the content of a daylight image [53].

in Tables 3.2, 3.3, and 3.4.

1.4 Contributions of this Research: Conversational AI Interface, Curriculum, and Workshop Results

The four main contributions of this thesis are (1) a block-based interface in MIT App Inventor for creating conversational agents, (2) conversational AI curriculum involving the interface, (3) the implementation of the curriculum through workshops, and (4) pre- and post-workshop questionnaire results and analysis. These components are described in detail in the following sections.

Chapter 2

Technical Implementation

> *"Technology is nothing. What's important is that you have a faith in people, that they're basically good and smart, and if you give them tools, they'll do wonderful things with them."*
>
> — Steve Jobs

To enable students like Sheila, who was introduced in Section 1.2.1, to develop interactive storybooks, talking textbooks, favorite spot guides, and other mobile-device-connected conversational AI agents, I developed a block-based interface to simplify conversational agent programming. I created this interface within the MIT App Inventor framework to include the following features [1]:

- a *designer page* where the user can create and send the Alexa skill to Amazon
- a *blocks page* where the user can program the Alexa skill, and
- fifteen new *voice* blocks used to program the Alexa skill.

To familiarize readers (and future developers like Sheila) with conversational AI terminology, the following sections describe these features with respect to Sheila's storybook app.

2.1 Creating Sheila's Storybook App

This section describes how Sheila, the primary school student introduced in Section 1.2.1, programmed her storybook app in MIT App Inventor using the conversational AI interface.

> For her seventh grade art project, Sheila designed her own snazzy, stapled-together storybook. As much as she loved her physical storybook, she also realized that a digital storybook could reach a wider audience, and could be easily shared with little cousin, Jaidon, who lived far away from her hometown in Boston. In her computer technology class, Sheila heard about MIT App Inventor and had a brilliant idea: Create a storybook app that could be read on mobile devices anywhere in the world! She also heard about MIT App Inventor's conversational AI tools and imagined creating a conversational agent for her storybook character, Karabo the Zorilla. Her cousin could speak to Karabo and learn about zorilla culture!
>
> To start, Sheila uploaded scans of her storybook illustrations to MIT App Inventor. She attached the first illustration to an Image object in the Components panel, and created a Label object with the storybook text, as shown in Figure 2-1. Then she added Button objects labelled "Next" and "Prev", which would be used to flip through the storybook pages. To make the Button objects more than just idle pixels, Sheila went to the Blocks workspace and programmed page-flipping functionality, as shown in Figure 2-2.
>
> To allow Jaidon, who couldn't yet read, to experience her exciting new story, Sheila decided to create an Alexa Skill that could read the book for him. In the MIT App Inventor interface, she added a Skill by clicking the *Add Skill* button near the top of the screen, as shown in Figure 2-3. Next, she created an *intent* for reading the story by using VUI blocks, as shown in 2-4. To program Alexa to read the story, Sheila connected *endpoint* blocks together and linked them to the story-reading intent using the dropdown menu shown in Figure 2-7.
>
> Before making the storybook app, Sheila had shared her story with her friends. They had asked her about what zorillas were and where they came from. Sheila had completed a whole science report on zorillas, so she told her friends all about their natural habitat, their defence mechanisms, and even their phylum, class and order. Sheila decided to add some of these facts to the Alexa skill so that others could ask and learn more about zorillas.
>
> To do so, Sheila added *whatZorillasEat*, *whenZorillasSleep*, and *whereZorillasLive* intents. For the *whereZorillasLive* intent, she used *slot* blocks. With these blocks, she programmed Alexa to respond differently to questions containing locations where zorillas live versus other locations. For instance, if someone asked Alexa, "Do zorillas live in the United States?", Alexa would respond, "Actually, zorillas don't live in the United States. They mostly live in Africa"; whereas Alexa would respond positively if they asked, "Are there zorillas in Africa?". Sheila used the *get slot* block, as shown in Figure 2-5, as a placeholder for countries and continents. To get the name of the country or continent the user said, Sheila used the *get slot value* block, as shown in Figure 2-6, and added it to Alexa's response.

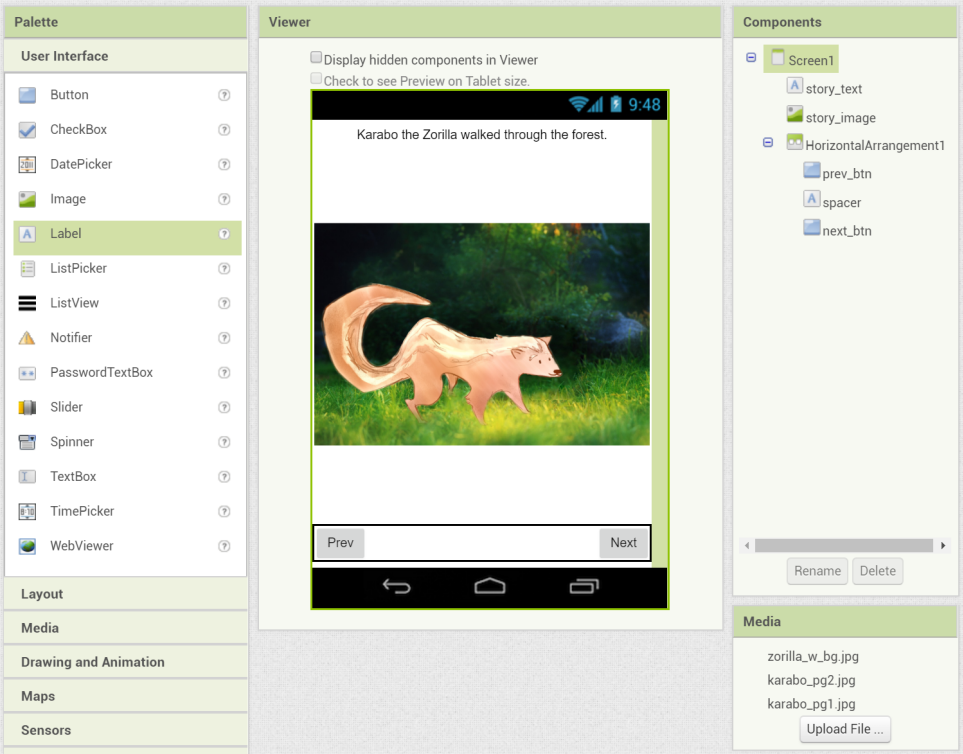


Figure 2-1: The app designer page, showing the development of a storybook application. Notice the Label, Image, and Button objects in the Components panel and how they appear visually in the Viewer panel.

Finally, to top off her story, Sheila wanted Jaidon and other users to be able to talk with Karabo, as if the zorilla spoke English. She didn't want to program each conversation directly, though, because this would take *forever*, so she talked to her computer teacher, Mr. Danyluk. He had a fantastic idea, "You could use a *generate text* block!"

"What's that?" Sheila asked.

"It's a block that can generate English words and sentences by using something called a neural network. This network has read hundreds of sentences in storybooks, like *Dr. Seuss*, and learned to guess what the next best letter in a string of letters would be based on the sentences it read. This means that you could give a string of letters to an *generate text* block — for example, give it a sentence that someone said to Karabo — and have the *generate text* block generate new sentences in response. Essentially, you could have Alexa generate words for Karabo to say, instead of manually programming each of Karabo's responses.

"You could say something like, "Alexa, what's Karabo's favorite food?", and she might respond by generating a Dr. Seuss-like sentence, like, 'Green eggs and ham, Sam I am!'"

The *generate text* block seemed like a great time-saver to Sheila. She immediately went back to her computer, created an intent for talking with Karabo, found the *generate text* block, and added it to the endpoint for the *talkToKarabo* intent, as shown in Figure 2-8.

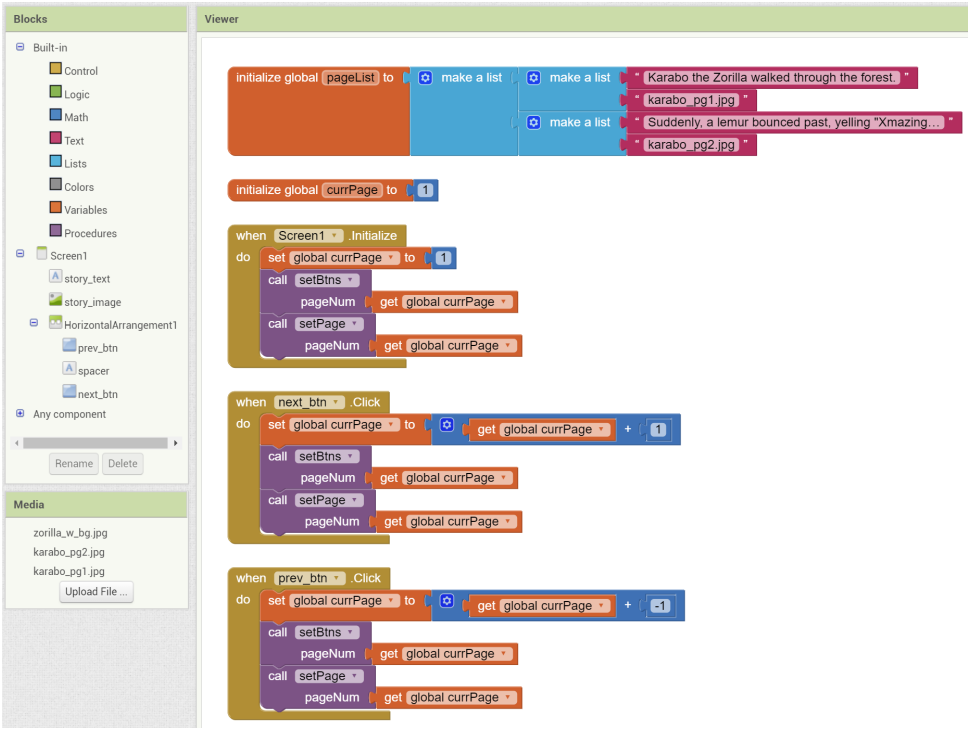


Figure 2-2: The MIT App Inventor Blocks page, showing the development of a storybook application. Notice the event blocks, including the *when screen initialize* block and *when button click* blocks. These blocks describe what happens when particular user events occur, and make function calls that set text and graphics on-screen to media described by the *initialize global variable* block.

After finishing the VUI and endpoint function, she sent her Alexa Skill VUI to Amazon by clicking the "Send updates to Alexa button", and uploaded her endpoint to AWS Lambda with her parents' help. Finally, Sheila built the app, tested it, and sent it to her aunt and uncle. The next day, she called them, and told Jaidon about her app. Jaidon was very excited, and immediately went to find his tablet. After installing the app (with Sheila's help, of course), he started flipping through the pages and talking with Karabo the Zorilla using the Alexa app on his tablet. Although the sentences Karabo responded with weren't very logical, Jaidon had a blast hearing the infinite different ways Karabo could respond, and laughed when Karabo said, "Oily all. Some you will see supermen!". Hearing Jaidon's laughter meant the world to Sheila. In her mind, an app that allowed her to connect with her cousin thousands of miles away was a huge success.

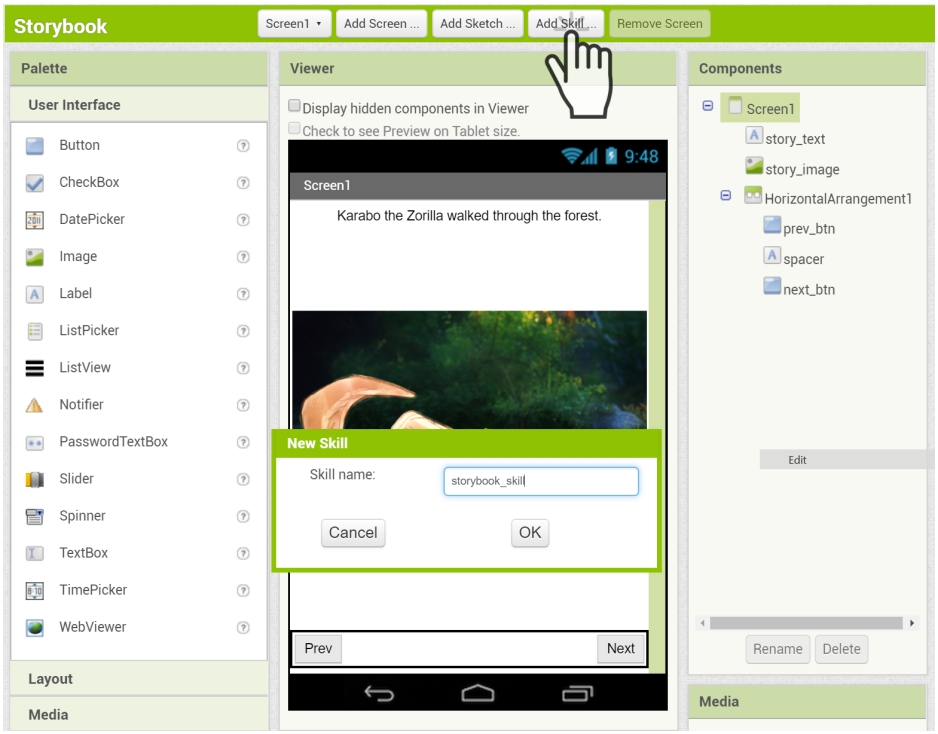


Figure 2-3: Adding a skill to an MIT App Inventor project. Notice the "Add Skill" button at the top of the workspace.

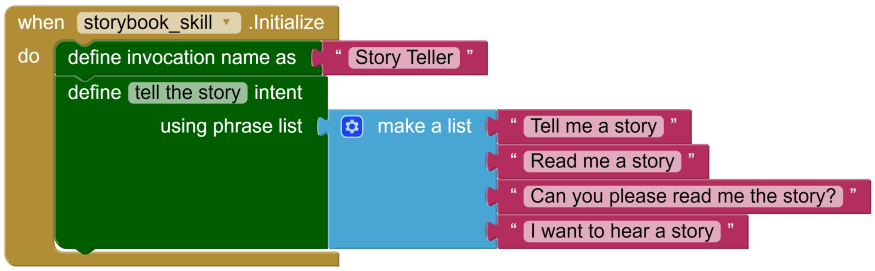


Figure 2-4: Blocks defining the VUI for an Alexa Skill. Notice the *define intent* block. This block defines the utterances that the user can say to invoke the "tell the story intent". When this intent is spoken, the endpoint blocks linked to this intent will run.

2.2 Amazon Alexa Skills and Associated Terminology

In Section 1.1, Jaidon speaks with Alexa about Sheila's storybook app. This was made possible through an *Amazon Alexa Skill* that Sheila created in MIT App Inventor. An Alexa skill is a voice-based app for an Alexa-enabled device. These devices include tablets with the Alexa App installed, home devices created by Amazon, such as an Amazon Echo [43], and computers logged into the Alexa Developer Console. The skill's *voice user interface*

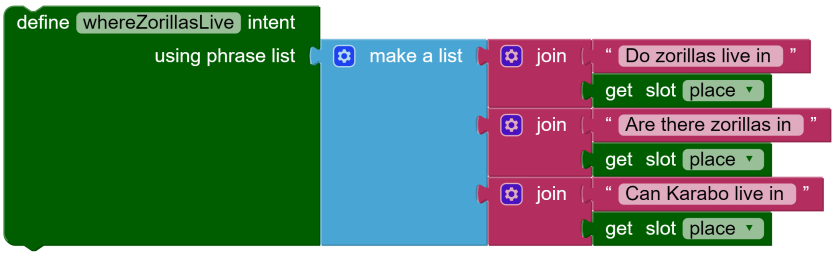


Figure 2-5: The blocks Sheila used to create the *whereZorillasLive* intent. Notice the *get slot* blocks. These are used as placeholders for words such as, "Africa" or the "United States". By using this block, Sheila does not have to list all possible places users might say.

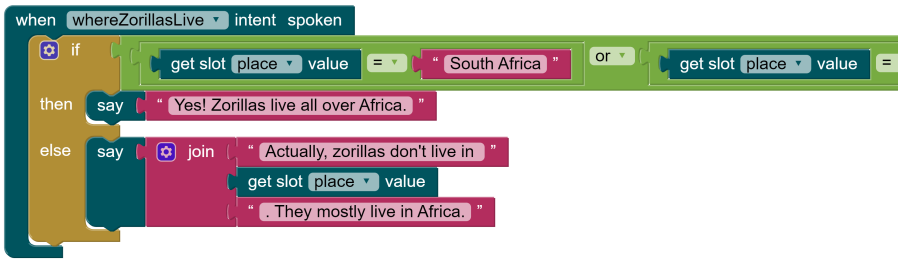


Figure 2-6: Sheila's endpoint blocks for the *whereZorillasLive* intent. Notice the *get slot value* block. This block returns the value the user stated in the utterance, such as "Africa" or the "United States".

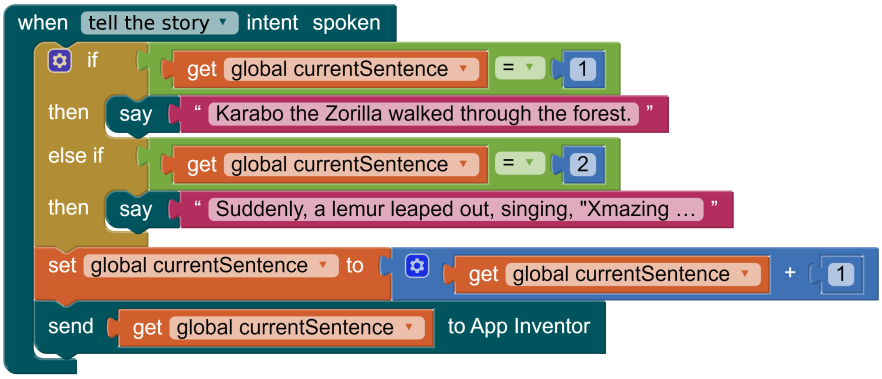


Figure 2-7: Blocks defining the endpoint function for the "tell the story" intent. When an utterance from the "tell the story" intent is spoken, these blocks will run, and Alexa will read the current sentence aloud.

(VUI) manages interactions with Alexa. In Sheila's storybook app, the VUI consists of utterances that Alexa understands, such as "What kind of animal is Karabo?" or "Read me the story", and Alexa's responses, such as "Karabo is a zorilla!" and "Karabo walked

Figure 2-8: The endpoint event for the *talkToKarabo* intent. These blocks cause Alexa to respond with a uniquely generated sentence that may sound similar to a sentence Dr. Seuss came up with.

through the forest". Alexa determines how to respond via the *endpoint function* that Sheila programmed. The endpoint function resides on a server, such as Amazon Web Services' Lambda function servers.

High school students are likely to be unfamiliar with the terminology describing Alexa skills and conversational AI applications. Table 2.1 provides definitions for this terminology, which are included in the high school workshop curriculum, as described in Section 3.

Table 2.1: Conversational AI and Alexa skill terminology.

Term	**Meaning in the context of Sheila's storybook app**
Skill	A voice-based application for Amazon Alexa-enabled devices, such as Amazon Echo Dots or Alexa-enabled smartphones. This is the program that enables Jaidon to speak with Alexa about Sheila's storybook.
Voice User Interface (VUI)	The spoken part of a skill; in other words, the statements, phrases, and questions spoken to and by the skill. For example, the phrases, "What's smellier, a zorilla or skunk?" and "A bigger animal generally means a bigger stench, so I'd guess skunks are smellier!", in the storybook skill. In the Alexa Developer Console, the VUI is defined by a JSON. In MIT App Inventor, this JSON is created and sent to the Console using VUI blocks, as shown in Section 2.3.
Wake Word	The word that the Amazon Alexa-enabled device listens for before beginning to interact with the user. After the wake word is spoken, the device listens for additional words and phrases, such as an invocation name. Generally, the wake word is "Alexa", but this can be changed to "Amazon", "Echo", or "computer" [38].
Invocation Name	A name associated with a skill that causes Alexa to open (start) the skill. For example, given that the invocation name for the storybook app is "Zorilla Storybook", one can say "Alexa, launch *Zorilla Storybook*" to open the skill.

Intent	Part of a skill's VUI associated with a particular action in the skill. An intent includes sample utterances that will invoke the intent. For example, Sheila's *talkToKarabo* intent is associated with the action of Alexa generating and speaking unique sentences. The *talkToKarabo* intent can be invoked by saying utterances such as "Hi Karabo" or "How are you today?". When an intent is invoked, Alexa sends a request to the associated endpoint, which carries out an associated action, such as generating a sentence. Note that before invoking an intent, the skill must be specified by stating the *invocation name*, as described above. For instance, one cannot just say, "How are you today?", to Alexa to invoke the *talkToKarabo* intent, since Alexa may associate this intent with multiple different skills. To specify the intent, one can instead say, "Alexa, launch *Zorilla Storybook* and ask, 'How are you today?'".
Built-in Intent	An intent that Amazon automatically creates for each skill. Some built-in intents include the "help", "stop", "cancel", and "fallback" intents. Each of these intents have predefined sample utterances, such as "help me", and send out predefined intent requests. For more information about built-in intents, see [39]. For the storybook app, Sheila could program the "help" intent to cause Alexa to say, "If you're not sure what to do, try asking me to read you the story or talk to one of the characters".
Utterance	The phrases a skill associates with a particular intent or slot. For example, Jaidon might speak the utterance, "Say hi to Karabo" to invoke the *talkToKarabo* intent.
Slot	A variable within an utterance that can be filled by the user. Each slot has a specific slot type. For example, the slot type in Sheila's *whereZorillasLive* intent is "place", as shown in Figure 2-5. To fill the *place slot*, Jaidon might say, "Do zorillas live in *South America*?" In this case, the slot's value would be South America. For more information about slots and slot types, see [41].
Endpoint	A service that can receive JSON requests and return JSON responses. Generally, this service is an Amazon Lambda Function (but it can also be a custom HTTP web service). To illustrate, after Alexa hears an utterance, such as "feed Karabo pizza", it sends a JSON request including the intent and slot information to the endpoint, which returns a JSON response. For instance, the *feed Karabo* request would cause the endpoint to send a signal to the storybook app to display Karabo with pizza and return a JSON response causing Alexa to say, "Aye, aye! Teleporting pizza to Karabo!".
Lambda Function	In the context of Alexa skills, this is an endpoint that Amazon hosts on Amazon Web Services. This function is written in JavaScript (or another accepted language) and contains event handlers. After receiving a request, the Lambda function will send a JSON response back to Alexa.

Trigger	Something that sends requests to a Lambda function (or another endpoint). In the context of Alexa skills, the trigger is the Alexa VUI.

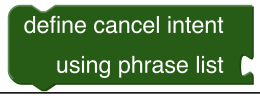
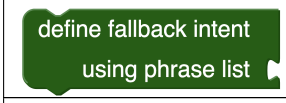
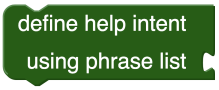
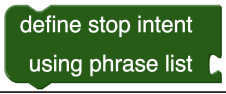
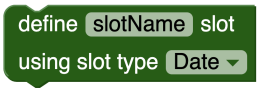
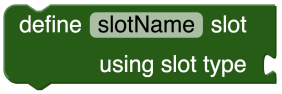
In summary, to communicate with an Alexa skill, first, one says the *wake word*, which is usually "Alexa". Next, one specifies a particular skill with an *invocation name*, such as, "Zorilla Storybook". To cause this skill to perform a particular action, one says one of the *utterances* associated with a particular *intent*, such as the utterance, "Hi Karabo", which is associated with the *talkToKarabo* intent. This causes an *endpoint function* to perform an action, such as having Alexa speak the words, "Hi, I'm Karabo!". Altogether, one can say, "Alexa, launch *Zorilla Storybook* and say, 'Hi Karabo'", to cause Alexa to respond with, "Hi, I'm Karabo!". Blocks in the conversational AI interface enable people to customize interactions such as these and create their own conversational agents.

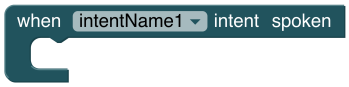
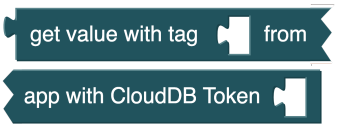
2.3 MIT App Inventor Blocks for Conversational AI

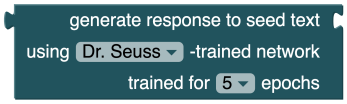
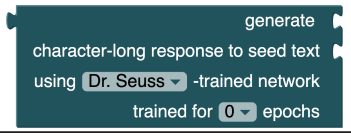
In Section 2.1, Sheila used forest-green *VUI* blocks to enable Jaidon to ask Alexa to read him the story (as shown in Figure 2-4). She also used teal *endpoint* blocks to cause Alexa to read the story when Jaidon asked (as shown in Figure 2-7). Unbeknownst to her, each of the blocks were translated into text-based programming languages to be run and accessed by Alexa-enabled devices. The VUI blocks were translated into a JSON and sent to the Alexa Developer Console, and the endpoint blocks were translated into JavaScript and sent to Amazon Web Services (AWS) Lambda. This section describes each of the conversational AI (or *voice*) blocks. Furthermore, Table 2.2 in Section 2.5 describes each block and its relevant JSON or JavaScript code snippet.

Table 2.2: VUI and endpoint blocks for Alexa Skills interface. Note that forest-green blocks are VUI blocks, and teal blocks are endpoint blocks.

Block	**Purpose and function**

	Defines the invocation name of the VUI, such as "Zorilla Storybook". See Table 2.1 for more information about invocation names.
	Defines intents for the VUI, such as the "feed Karabo intent", using a list of phrases, such as "feed Karabo" and "give Karabo food". See Table 2.1 for the definition of an intent.
	Defines Amazon's "cancel" built-in intent. See Table 2.1 for more information about built-in intents [39].
	Defines Amazon's "fallback" built-in intent. See Table 2.1 for more information about built-in intents [39].
	Defines Amazon's "help" built-in intent. See Table 2.1 for more information about built-in intents [39].
	Defines Amazon's "stop" built-in intent. See Table 2.1 for more information about built-in intents [39].
	Defines a slot. This block has a drop-down menu containing common, predefined Amazon slot types; for example, "place" (which would correspond to the slot type, "AMAZON.Place", in the Amazon corpus). This slot may be filled by a user's utterance, such as "Are there zorillas in *Europe*?" (which would result in a slot value of "Europe"). See Table 2.1 for more information about slots and slot types [41].
	Defines a slot using an input block. The input block should be a string containing a predefined Amazon slot type; for example, "AMAZON.Place". See Table 2.1 for more information about slots and slot types [41].
	Used as a placeholder variable for a slot. For example, if an intent utterance is "Are there zorillas in <*Asia*>?", the *get slot* block may be used in place of <*Asia*> when defining the utterance. This allows variables (such as the *place slot*) to be filled by the end user. See Figure 2-5 for an example set of blocks using the *get slot* block. See Table 2.1 for more information about slots and slot types [41].

	Defines what occurs when a specific intent is spoken to Alexa. This block can contain regular App Inventor blocks (e.g., an if-statement block) as well as other endpoint blocks, such as the "say" block. The inner blocks that this block contains define the JavaScript for a skill's endpoint. The drop-down menu contains a list of intents defined by VUI blocks in the workspace.
	Causes Alexa to say something. For example, if the input block was a text block containing, "Karabo walked through the forest", Alexa would speak this sentence.
	Causes Alexa to send information to an app created in MIT App Inventor. For example, if the block in the *value* input contains the word "food", then "food" is sent to CloudDB. In Sheila's storybook app, this would trigger a picture of Karabo with food to be shown on-screen. Additionally, the *tag* input may be specified, which provides a name (tag) for the value to enable easy access in CloudDB. To ensure the tag and value are sent to the correct application, one must specify the *CloudDB Token* input. This token is a string that can be found in the properties section of designer panel. See Section 2.5.1 for more information about this block's implementation.
	Causes Alexa to retrieve information from an app created in MIT App Inventor. Prior to Alexa retrieving this information, the app must store the information using CloudDB. For example, Sheila may store the value, "Karabo ate food" with the tag *ateFood* in CloudDB after Jaidon taps on Karabo's food in the storybook app. If Jaidon asked Alexa, "Has Karabo eaten the food?", Alexa would access the *ateFood* variable using this block, and respond with, "Yes, Karabo ate his food". See Section 2.5.1 for more information about the implementation of this block.

	Generates unique, 35-character long sentences using a pretrained long-short term memory (LSTM) neural network. For example, Sheila used this block to generate Dr. Seuss-like sentences when Alexa is asked, "Can I talk to Karabo?" Other pre-trained LSTM networks can be chosen with the drop down menu to generate *Nancy Drew*-, *Alice in Wonderland*-, *Tom Sawyer*-, etc. -like sentences. The amount of training the network has undergone can also be adjusted with a second dropdown menu. This can be set to zero, one, five, or twenty epochs. Section 2.5.2 for more information about the implementation.
	Generates unique sentences using a pretrained long-short term memory (LSTM) neural network. This block enables users to set the length of the output sentence. Section 2.5.2 for more information about the implementation.
	Returns the value of the slot that the end-user specified. For example, if Jaidon asked Alexa, "Are there zorillas in Australia?", the *get slot value* block would return *Australia*. See Figure 2-6 for an example of an endpoint function using the *get slot value* block.

2.4 Using the Interface

In Section 2.1, Sheila used the conversational AI interface (or the "Amazon Alexa Skills" interface) in MIT App Inventor, which consists of two main pages. The first is the *designer* page. This page displays an Alexa-enabled device, and has buttons to send the user-created VUI to Amazon and to generate the endpoint code, as shown in Figure 2-9. The second is the *blocks* page. This page contains conversational AI blocks, as described in Section 2.3, enabling users like Sheila to program Alexa Skills. The *blocks* page is shown in Figure 2-10.

Prior to the high school workshops, this interface was tested by a number of MIT App Inventor software developers, students, and staff. Each participant began creating an Alexa Skill in MIT App Inventor using a tutorial, and wrote down any bugs, difficulties, or confusion they came across during the session. A version of the tutorial they used can be

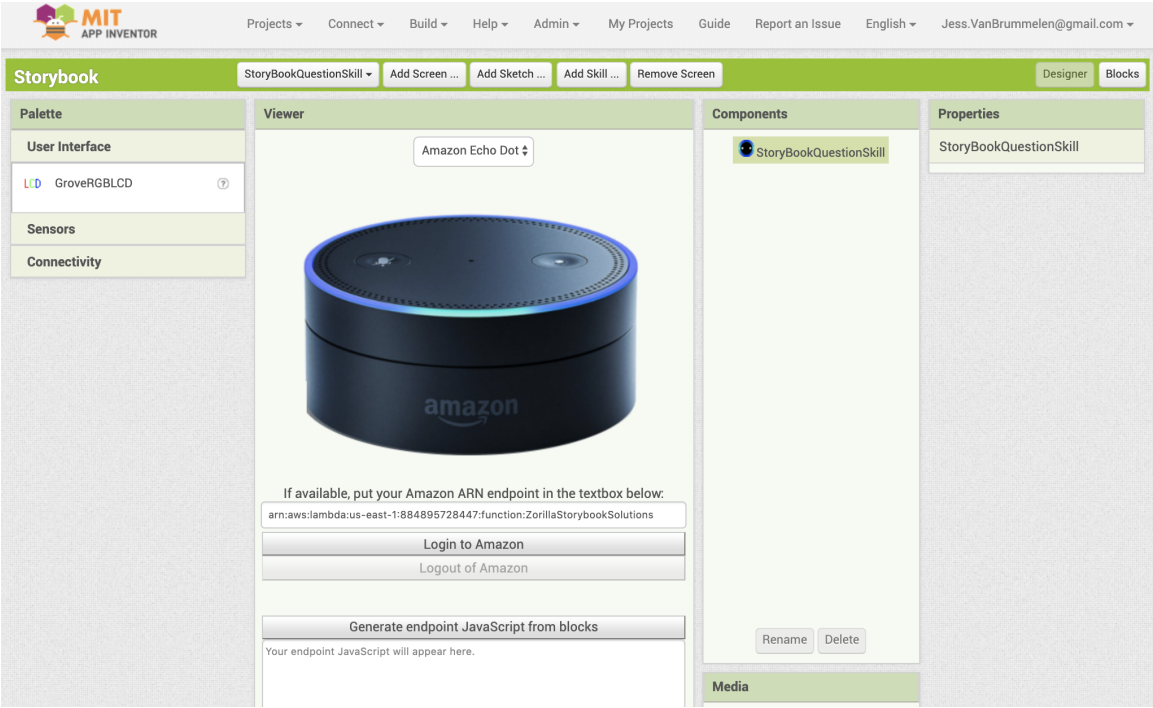


Figure 2-9: The MIT App Inventor Alexa Skill designer page. Notice the *Login to Amazon* and *Generate endpoint JavaScript* buttons.

found in Section F.1 of the Appendices. Bugs ranged in severity and salience; for example, erroneous quotation marks appeared in the generated JavaScript and in certain cases, the website would not load due to Chrome extension interference.

After the sessions, the MIT App Inventor developers and students debugged the interface to prepare for the high school workshops. During the workshops, the students were provided with the tutorial shown in Section E.4 of the appendices. This tutorial is similar to the testing-session tutorial; however, it does not require students to have an AWS account. Instead, workshop instructors upload students' endpoints to a shared AWS account. Using this interface, students developed socially useful Alexa Skills, such as skills to help people sort recyclables, learn math concepts, see Alexa's speech on-screen, and recall forgotten words, as described in Section 4.2.1.

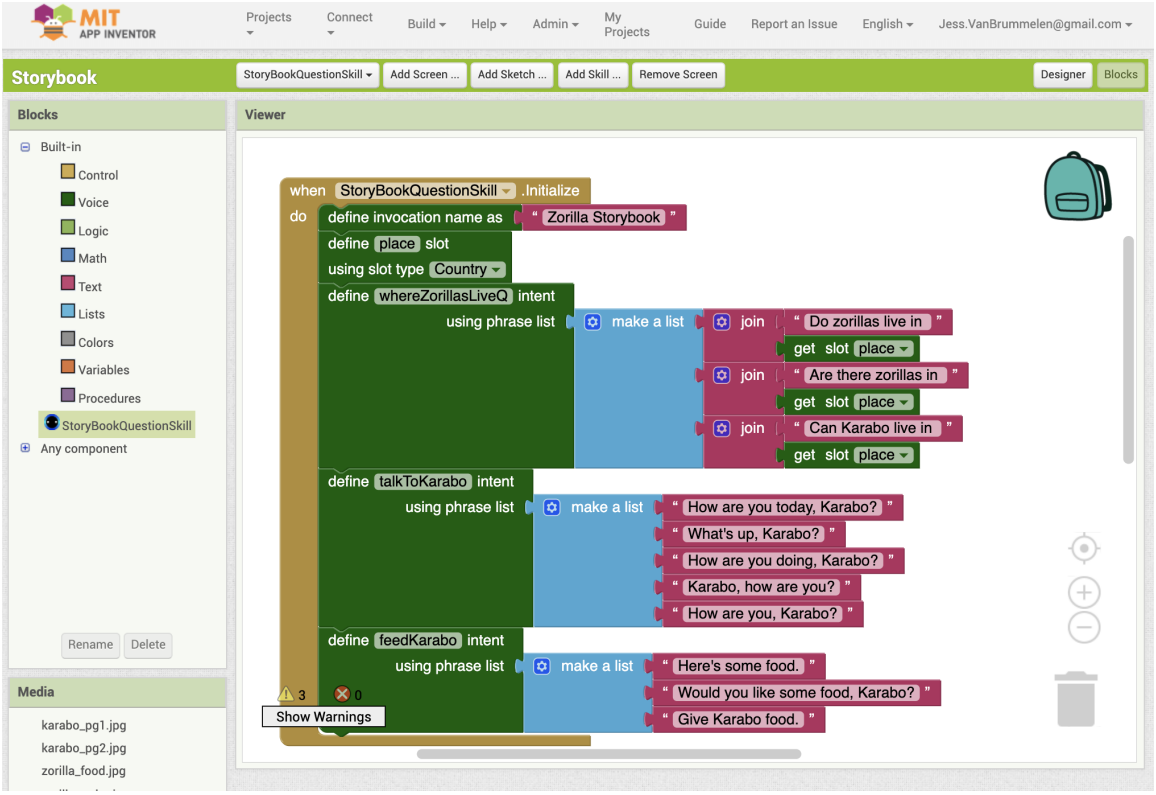


Figure 2-10: The MIT App Inventor Alexa Skill blocks page. Notice the *Voice* drawer on the left side of the screen in the Blocks panel. This is where the VUI and endpoint blocks reside.

2.5 Interfacing with Amazon

In Section 1.1, Jaidon spoke to the storybook skill through the Alexa app on his mobile device. To enable the Alexa app to recognize Jaidon's utterances, Sheila had to transfer the VUI she created from MIT App Inventor to the Amazon Developer Console. To do so, she clicked the *Send to Amazon* button on the *designer* page, as described in Section 2.4. MIT App Inventor then generated a SMAPI-understandable JSON based on the VUI blocks in the conversational AI workspace. To generate this JSON, each VUI block in the Sheila's workspace was converted to a key-value pair. The key-value pairs for each block in the interface are shown in Table 2.3. After conversion, the JSON was sent to Amazon using the Alexa Skill Management API (SMAPI) and OAuth 2.0 authentication. This process, which is shown in Figure 2-11, enables anyone to send MIT App Inventor-defined VUIs to Amazon devices.

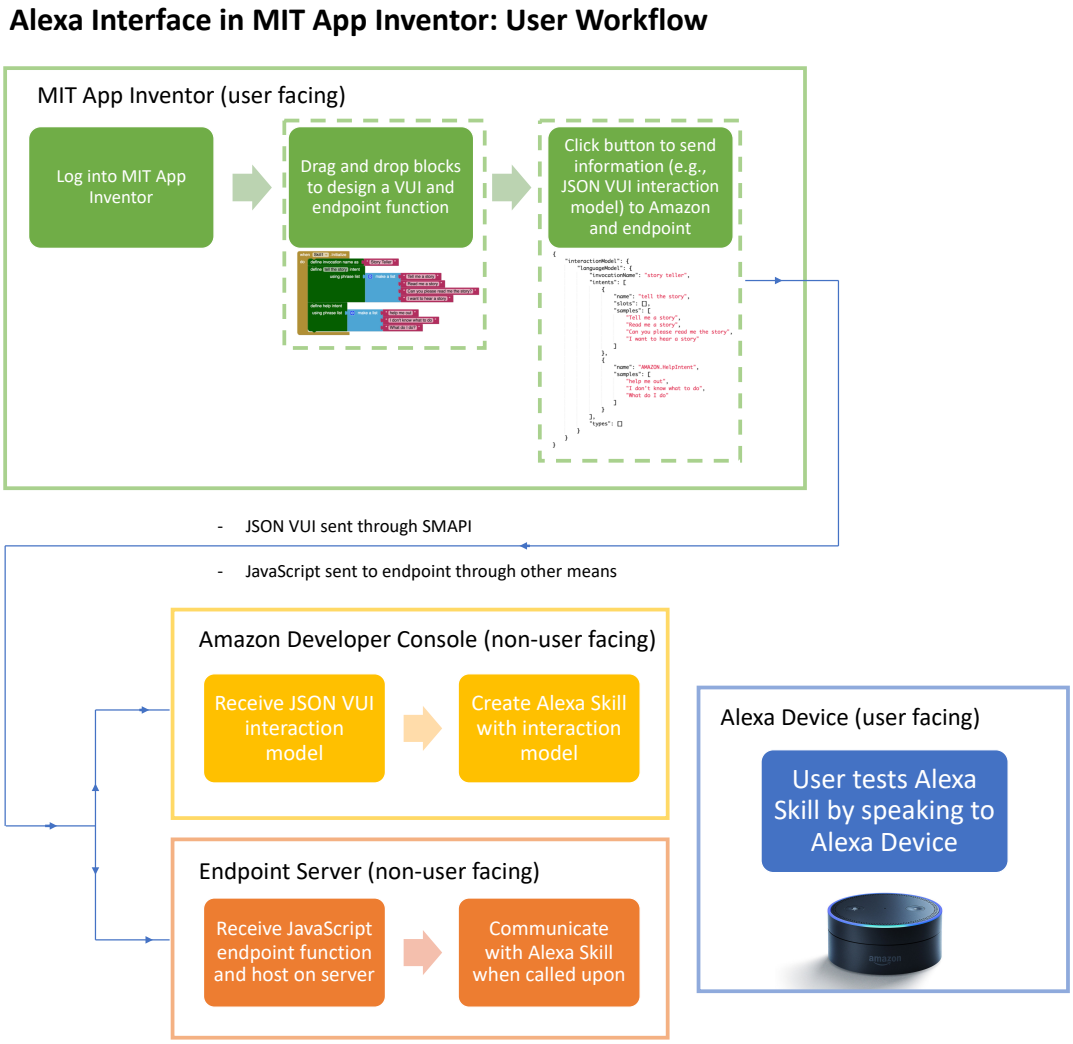


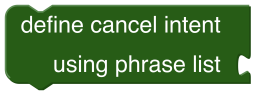
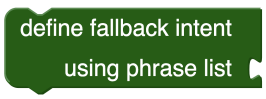
Figure 2-11: The user workflow starting with creating an Alexa skill in MIT App Inventor and finishing with testing the skill on an Amazon device. Notice how the user never observes the JSON VUI code, as it is sent from MIT App Inventor to Amazon through the SMAPI interface. Note that the current implementation requires users to manually upload the generated JavaScript to an endpoint server. In future iterations, this will be automatically completed by MIT App Inventor, as described in Figure 2-13 and in Section 6.

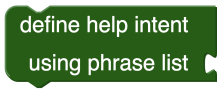
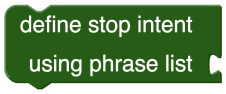
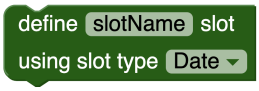
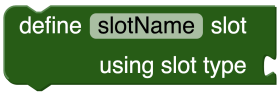
Furthermore, to enable Alexa to respond to the utterance, "Is a skunk smellier than a zorilla?" (as in the excerpt in Section 1.1), Sheila defined an endpoint function using the *when intent spoken* and *say* blocks, which are shown in Table 2.3. When Sheila clicked the *Generate endpoint JavaScript* block, App inventor recursively converted these blocks to Node.js JavaScript snippets and combined them into an endpoint function. This function

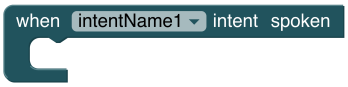
included a standardized header and footer with information about the required JavaScript libraries, such as Redis and the Alexa SDK [68, 3], and pre-defined functions and constants to be used in the endpoint code snippets. After compiling the JavaScript Node.js function, Sheila uploaded it to AWS Lambda (with the help of her parents), enabling the Alexa VUI to interact with the endpoint, and Alexa to respond to Jaidon's utterances.

The standardized header and footer code snippets for the JavaScript file are shown in Listing A.1 and A.2 in Appendix A. These include predefined functions and constants, and information about the required libraries. The code snippets for each block in the Alexa Skills interface are shown in Table 2.3.

Table 2.3: The code snippets that correspond to each block. The VUI (forest-green) blocks correspond to JSON key-value pairs, whereas the endpoint (teal) blocks correspond to JavaScript code snippets. Note that the "<input>" strings in the code snippets correspond to the input "cutouts" in the blocks (where input blocks can be placed).

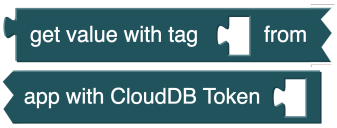
Block	Corresponding Code
	`interactionModel.languageModel.invocationName: "<input>"`
	`interactionModel.languageModel.intents[i].name : "intentName"` `interactionModel.languageModel.intents[i]. samples: [<input>]`
	`interactionModel.languageModel.intents[i].name : "AMAZON.CancelIntent"` `interactionModel.languageModel.intents[i]. samples: [<input>]`
	`interactionModel.languageModel.intents[i].name : "AMAZON.FallbackIntent"` `interactionModel.languageModel.intents[i]. samples: [<input>]`

	`interactionModel.languageModel.intents[i].name : "AMAZON.HelpIntent"` `interactionModel.languageModel.intents[i]. samples: [<input>]`
	`interactionModel.languageModel.intents[i].name : "AMAZON.StopIntent"` `interactionModel.languageModel.intents[i]. samples: [<input>]`
	`interactionModel.languageModel.intents[i]. slots[j].name: "slotName"` `interactionModel.languageModel.intents[i]. slots[j].type: "AMAZON.DATE"`
	`interactionModel.languageModel.intents[i]. slots[j].name: "slotName"` `interactionModel.languageModel.intents[i]. slots[j].type: "<input>"`
	`... "{slotName}" ...`

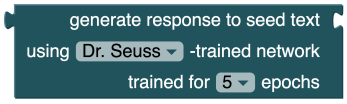
	`...`
	`"intentName1": async function() {`
	`... <input> ...`
	`this.emit(":responseReady");`
	`},`
	`...`

	`...`
	`this.response.speak("<input>");`
	`...`

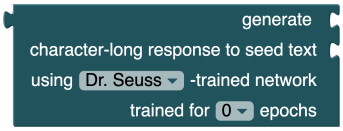
	`...`
	`setCloudDB("<input_key>","<input_value>",< ProjectName>,"<input_token>");`
	`...`
	The function definition for `setCloudDB` is shown in Appendix A.1, and the technical details are outlined in Section 2.5.1.

<table>
<tr>
<td> 
</td>
<td>

```
...
await getCloudDB("<input_key>",<ProjectName>,"
    <input_token>")
...
```

The function definition for <code>getCloudDB</code> is shown in Appendix A.1, and the technical details are outlined in Section 2.5.1.</td>
</tr>
<tr>
<td> 
</td>
<td>

```
...
(await getText(csail_url, "<input>", "
    drSeuss_5", 35)).generated
...
```

The function definition for <code>getText</code> is shown in Appendix A.1, and the technical details are outlined in Section 2.5.2.</td>
</tr>
<tr>
<td> 
</td>
<td>

```
...
(await getText(csail_url, "<input_seed_text>",
     "drSeuss_0", <input_length>)).generated
...
```

The function definition for <code>getText</code> is shown in Appendix A.1, and the technical details are outlined in Section 2.5.2. This block allows users to specify the length of the output text.</td>
</tr>
<tr>
<td> 
</td>
<td>

```
this.event.request.intent.slots.slotName.value
```

</td>
</tr>
</table>

Figure 2-12 illustrates the interactions between the systems involved in developing a conversational agent with MIT App Inventor. The green arrows represent the VUI JSON being sent from MIT App Inventor to the Amazon Developer console, which sends the skill to Amazon devices. The blue arrows represent variables being set and retrieved from the CloudDB database, as described in Section 2.5.1. The orange arrows represent the endpoint JavaScript being manually uploaded to AWS Lambda, and the Alexa skill interacting with this endpoint function. In future iterations, I plan to remove the need for users to upload JavaScipt to AWS Lambda and rather have MIT App Inventor interact directly with AWS Lambda through an API (which is not currently an AWS feature) or through an alternative endpoint service, such as the Jovo framework [44]. A diagram

depicting this alternative architecture is shown in Figure 2-13.

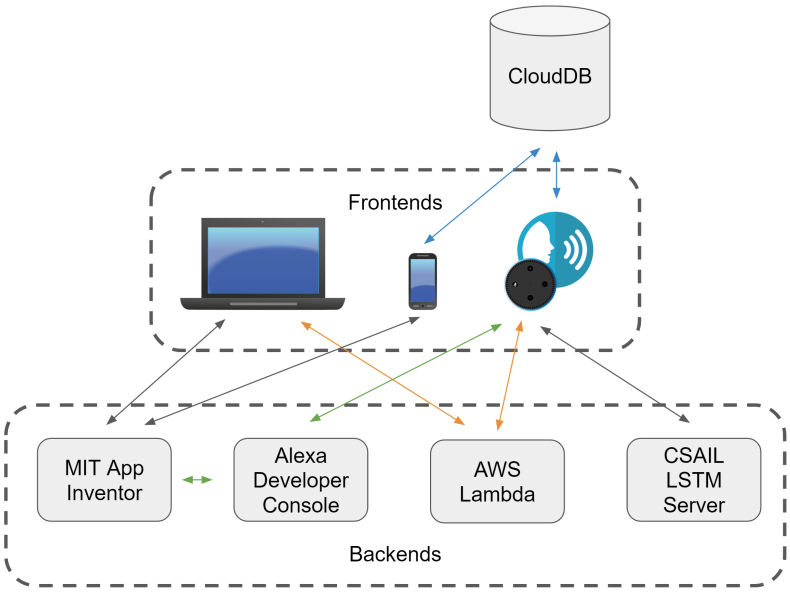


Figure 2-12: The current architecture for the conversational AI interface. The green, orange, and blue arrows signify VUI-, endpoint-, and CloudDB-related communication respectively.

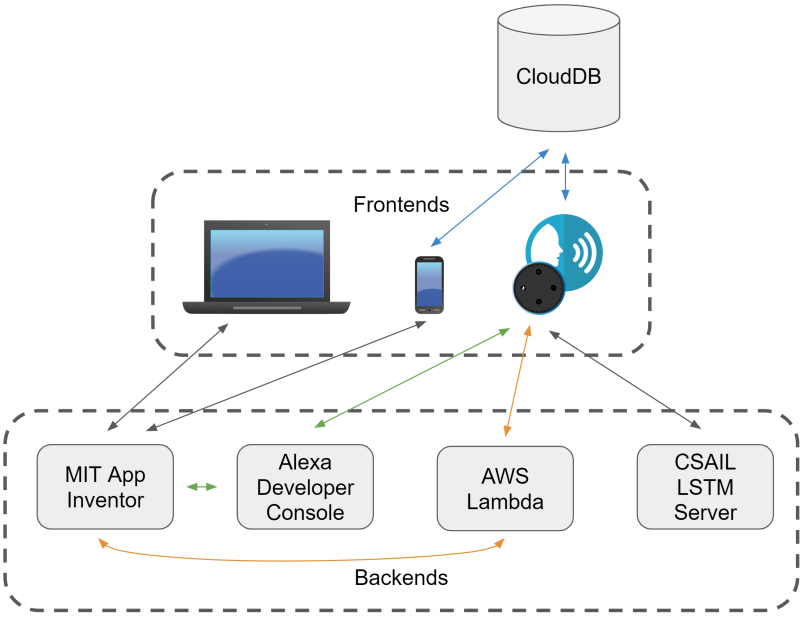


Figure 2-13: The desired architecture for the conversational AI interface. The green, orange, and blue arrows signify VUI-, endpoint-, and CloudDB-related communication respectively. Notice that in this case, the user does not need to manually upload data to AWS Lambda, as depicted by the orange arrows that do not interact with the computer (unlike in Figure 2-12).

2.5.1 Technical Details of the Send to and Get from App Blocks

In the storybook app, Sheila used the *send to app* block, as shown in Figure 2-14, to enable Jaidon to feed Karabo by saying, "Alexa, give Karabo some food", which sent the string, "food", to the app. Another block — the *get from app* block — could have enabled Alexa to determine whether Karabo had eaten the food. For example, if Jaidon fed Karabo by tapping on food in the app, the value "true" for the variable *hasKaraboEaten* could be sent to CloudDB. Alexa could access this variable using the *get from app* block. Thus, if Jaidon asked, "Alexa, has Karabo eaten?", Alexa could be programmed to check the *hasKaraboEaten* variable and reply, "Yes, he's very full" or "No, he's starving! If you want to feed him, you can say, 'Feed Karabo'".

Figure 2-14: The *send to* and *get from app* blocks. These blocks are used in Alexa Skills to communicate with mobile apps.

Evidently, the *send to* and *get from app* blocks enable communication between Alexa skills and mobile apps developed in MIT App Inventor. They make use of Redis, an open-source key-value database [68]. In App Inventor-created mobile apps, a *CloudDB* component (i.e., the Redis client in MIT App Inventor) may get and set values in the database. Furthermore, when a relevant value is changed in the database, an event may be triggered in the mobile app [50]. Alexa Skills may also get and set values in the database via a Node.js Redis client. However, events cannot be triggered by external packages in Alexa Skills, as this could cause serious privacy issues. (For example, if Redis triggered an event in a skill, Alexa-enabled devices could potentially start listening or speaking without users' consent.) The communication pathways between apps, Redis, and Alexa Skills are illustrated in Figure 2-15. Furthermore, an architecture diagram showing communication between MIT App Inventor, Amazon, and other relevant systems is shown in Figure 2-12.

The *send value to app* and *get value from app* endpoint blocks enable Alexa Skills to

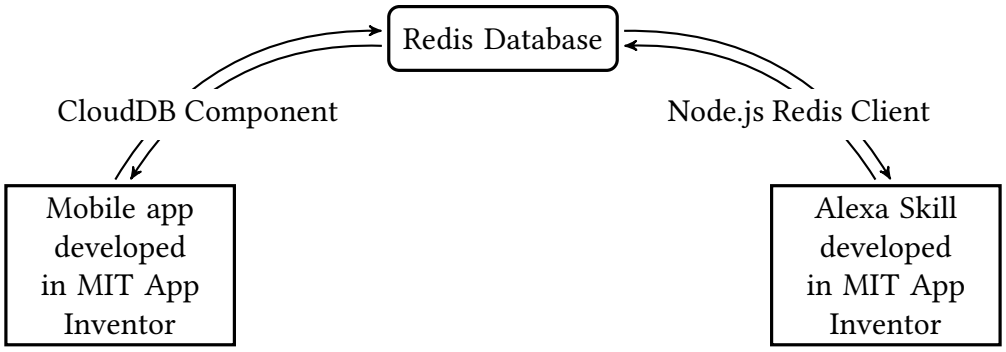


Figure 2-15: Communicating between mobile apps and Alexa Skills with Redis. Note that mobile apps can subscribe to changes in the Redis database, whereas Alexa Skills must initiate a request to a variable in order to determine whether it has changed.

get and set values in the Redis database and thereby communicate with mobile apps. The blocks correspond to the functions `setCloudDB` and `getCloudDB` respectively. These functions are shown in Listing 2.18 and 2.19. These functions are also shown in the context of the endpoint function header in Appendix A.1.

Listing 2.18: The `setCloudDB` function. It utilizes the Node.js Redis client to set values that may be accessed by MIT App Inventor mobile apps.

```
async function setCloudDB(key, value, projectName, cloudDBAuthKey) {
  let client = redis.createClient(urlHostPort, {
    'password': cloudDBAuthKey,
    'tls': {}
  });
  let json_value = JSON.stringify(value);
  return new Promise(((resolve, reject) => {
    client.evalsha(SET_SUB_SCRIPT, 1, key, json_value, JSON.stringify([
        json_value]), projectName, (err, res) => {
      if (err) console.log(err);
      client.end(true, (e, r) => console.log(e));
      resolve("Yes");
    });
  }));
}
```

Listing 2.19: The `getCloudDB` function. It utilizes the Node.js Redis client to retrieve values that may be set by MIT App Inventor mobile apps.

```
async function getCloudDB(key, projectName, cloudDBAuthKey) {
  let client = redis.createClient(urlHostPort, {
    'password': cloudDBAuthKey,
    'tls': {}
  });
  return new Promise(((resolve, reject) => {
    client.get(projectName + ':' + key, (err, value) => {
      if (err) console.log(err);
      client.end(true, (e, r) => console.log(e));
      resolve(value);
    })
  }));
}
```

2.5.2 Technical Details of the Generate Text (LSTM) Block

In the story presented in Section 1.1, Jaidon speaks with the storybook character, Karabo, through the Alexa skill Sheila designed. Every time Jaidon asks, "Alexa, can I speak to Karabo?", Karabo responds with a new sentence. Each sentence is similar to what one would see in a Dr. Seuss book, and was generated by the "*generate text*" or "*LSTM*" block. For instance, this block generated the sentences, "We sat in the hat", "I do not like them anywhere!", and "What a mouse!".

To generate such sentences, the *generate text* block, as shown in Figure 2-16, communicates with a pretrained long-short term memory (LSTM) network residing in a remote server. The block sends a HTTP request to this server, as shown in Listing 2.20, to generate text using one of the LSTMs specified by particular input corpora (e.g., Dr. Seuss, Alice in Wonderland, etc. corpora) and the amount of training the network has undergone (e.g., zero, five, etc. epochs). The communication channels between the remote server and Alexa Skills are illustrated in Figure 2-17 as well as in the architecture diagram in Figure 2-12. The code used to originally train the models as well as the server-side code for running the pretrained models can be found in the GitHub repositories [36] and [12] respectively.

The remote server contains twenty-eight LSTM networks with the same architecture,

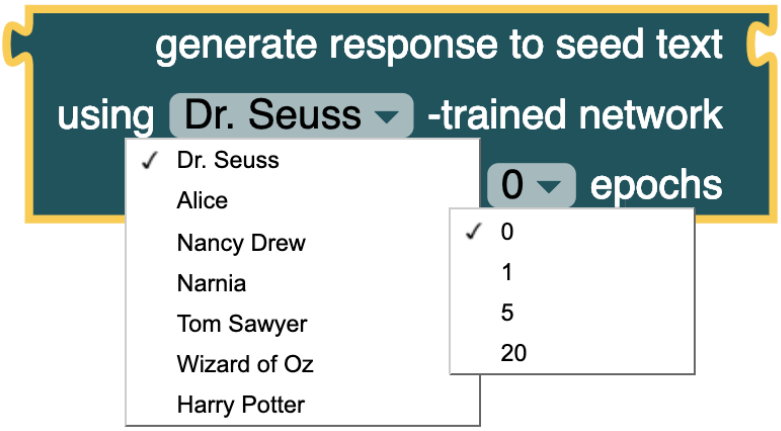


Figure 2-16: The *generate text* or *LSTM* block. This block generates sentences using an LSTM network pretrained for a certain number of epochs (shown in the second drop-down menu) on a specific corpora (shown in the first drop-down menu). The response is generated with respect to a *seed text* input.

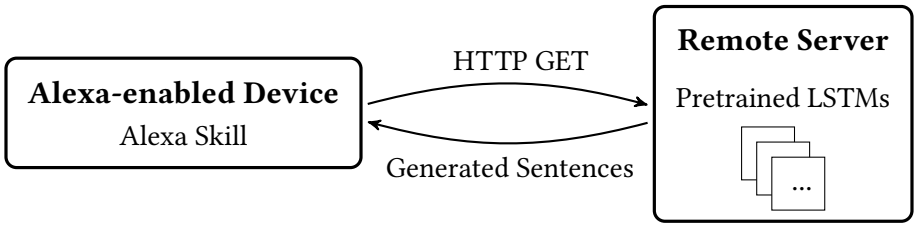


Figure 2-17: The communication between Alexa Skills and LSTM neural networks on a remote server. Alexa Skills can request uniquely generated sentences from pretrained LSTMs via GET requests.

but different weights. Each network was originally trained on one of seven different corpora for zero, one, five or twenty epochs. With the *generate text* block, students can choose one of these pretrained networks and explore the effects of varying the input and amount of training (without having to wait for the networks to train). For example, students might observe that an untrained network returns essentially random characters, whereas a network trained on Dr. Seuss for five epochs returns words and sentences akin to those written in Dr. Seuss books. These trends are shown in example output from the network in Figure 2-18. Potential AI learning outcomes from experimenting with this block include:

1. A neural network can generate unique sentences

2. The amount of training affects a neural network's output

3. The type of training data affects a neural network's output

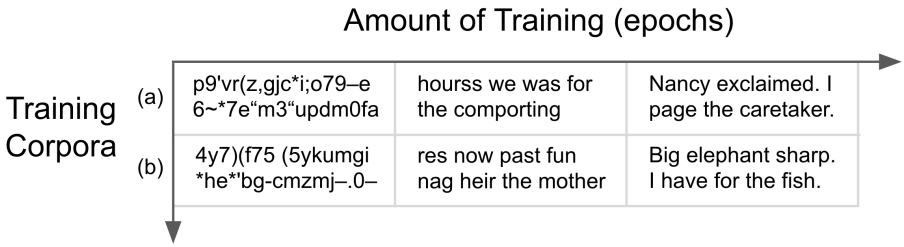


Figure 2-18: Example text generated character-by-character by the pretrained models used in the *generate text* block. The amount of training increases from left to right. Notice how the text generation becomes progressively better with more training, and how the text generated in (a) is clearly from a Nancy-Drew-trained model, whereas the text in (b) is clearly from a Dr.-Seuss-trained model.

The LSTM architecture is based on a generative model in the Keras GitHub repository [46]. It generates text character-by-character; contains an LSTM layer composed of 128 hidden units and a densely connected layer; and has a memory of forty time-steps. We trained models with this architecture using an RMSprop optimizer with a learning rate of 0.01 and a batch size of 256. The models now generate text on a remote server in a Node.js environment, based on an LSTM example in the TensorFlow.js GitHub repository [75]. When seed text is sent to the remote server, the last forty characters are used as initial memory for the LSTM, and a response is generated and returned. The length of the response can be changed as desired through the *outputLength* parameter in the HTTP request, and the desired pretrained model (e.g., the *Dr. Seuss* model) can be selected through the *model* parameter. With the default *generate text* block, the output length is set to 35 characters. Alternatively, a second *generate text* block, as shown in Figure 2-19, enables users to set the output length as desired.

Listing 2.20: The `getText` function for the *generate text* block. It utilizes the Node.js Fetch library to make HTTP GET requests. In Appendix A, this function is shown in the context of the endpoint header.

```
1 const getText = async function(baseUrl, input, model, length) {
2     let json;
3     let url = baseUrl + "?inputText=" + input + "&model=" + model + "&
          outputLength=" + length;
```

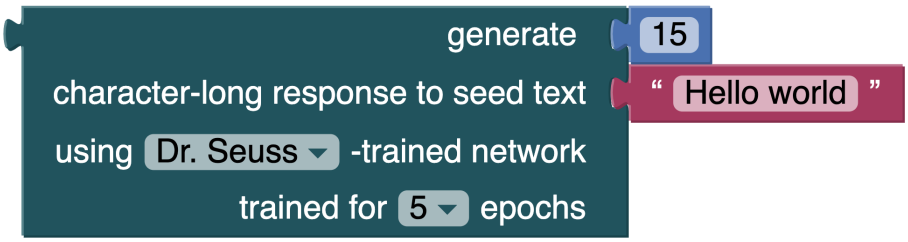


Figure 2-19: An alternative *generate text* block with the option to change the output length. In this case, the output length is set to 15 characters.

```
    try {
        const response = await fetch(url);
        json = await response.json();
    } catch (err) {
        console.log(err);
    }
    return json;
};
```

2.6 Technical Implementation Summary

In this section, we discussed how MIT app Inventor enables anyone to program conversational AI applications and connect them to mobile phone applications. The following capabilities were implemented in MIT App Inventor:

- the ability to specify a name for an Alexa skill (i.e., the invocation name), which enables Alexa to start the specified program,
- the ability to specify a name for *intents*, which enables Alexa to respond to specific utterances,
- the ability to specify *slot* variables, which allows end users to specify dates, numbers, names, etc. when interacting with Alexa skills,
- and the ability to program Alexa's response to specific utterances (i.e., to program the endpoint function). This response may involve Alexa

- speaking,
- computing values,
- communicating with a mobile app,
- or generating unique sentences.

The next section discusses curriculum I developed to teach high school students to create such applications.

Chapter 3

Curriculum and Workshops

> *"Technology will not replace great teachers but technology in the hands of great teachers can be transformational."*
>
> — George Couros

When developing the conversational AI interface, our goal was to empower high school students to create conversational AI applications. To test this, I developed a workshop curriculum for the interface and ran six hour-long workshops on Saturdays starting February 23, 2019 until April 6, 2019 (skipping March 16 for SPARK [59]) with Yulia Gonik, Tommy Heng, Terryn Brunelle, and Atsu Manigar. The workshop curriculum taught conversational AI concepts, how to program in MIT App Inventor, and how to use the conversational AI interface we developed. To conclude the workshops, we provided time for students to develop conversational AI projects. The workshop format was based on the MIT App Inventor CloudDB workshops in terms of project-based learning, number and length of workshops, and overarching organization through MIT's High School Studies Program (HSSP) [50, 58].

HSSP provided classroom scheduling, workshop advertising to the greater Boston area, and other teaching resources. Students were very interested in the workshop topic, with over 75 of ~350 students ranking the workshop in their top three choices out of 34 different workshops. Due to teaching assistant, classroom space, and hardware availability constraints, we limited the number of enrolled students to sixteen. Throughout the

semester, there were consistently 7-15 students attending the workshop, except for on the final day, in which there were fewer students[1]. An overview of each workshop topic is shown in Table 3.1, a detailed lesson plan for each workshop is shown in Appendix C, and a running summary document for the students is shown in Appendix E.7.

Table 3.1: Workshop series overview. Each workshop consisted of a short lecture about conversational AI as well as time to work on unplugged activities or developing applications.

Workshop	**Overview**
Workshop 1	• Introduce programming and mobile app development with MIT App Inventor. • Teach students how to program in MIT App Inventor with tutorials. (See the tutorial list under the heading *Lesson 1 Links* in the Appendix E.7.) • Complete post-questionnaire assessment. (See Appendix D.1 for the questionnaire.)
Workshop 2	• Introduce conversational AI and Amazon Alexa skills. • Explain basic AI concepts, such as the difference between rule-based AI and machine learning, and terminology, such as neural networks, training, testing, and weights. • Explain basic conversational AI terminology, such as endpoint function, VUI, invocation name, utterances, intents, and other terms. • Complete conversational agent unplugged activity where students create rules for a conversational agent and role-play. (See worksheet in Appendix E.1.) • Introduce the Alexa Skills interface in MIT App Inventor through a storybook skill. • Provide students with worksheets related to the "Talking to a Storybook" skill. (See worksheet in Appendix E.2.)

[1]This low attendance may have been due to the brilliant Boston weather on this particular Saturday (which had been rare throughout the spring).

Workshop 3	• Review the "Talking to a Storybook" worksheet solutions. (See solutions handout in Appendix E.3.) • Give time for the students to remix the storybook skill and create their own skills.
Workshop 4	• Introduce the conversational AI design project. • Brainstorm project ideas and create a project plan. (See individual and group worksheets in Appendix E.5 and E.6.) • Provide time for students to start programming their projects.
Workshop 5	• Briefly review conversational AI topics, including machine learning and rule-based AI, and some of the more complicated Alexa Skill blocks. • Provide students with time to complete final projects and prepare presentations.
Workshop 6	• Finish developing projects and provide time for student presentations. • Complete post-questionnaire assessment. (See Appendix D.2 for the questionnaire.)

As described in Section 1.3.2, I developed curriculum in response to recommendations provided by other researchers in the K-12 AI democratization field. Specifically, I addressed the recommendations by providing avenues for students to

- compare and contrast how humans and conversational agents think,
- view, design, and personalize algorithms for the agents,
- learn the difference between generative and rule-based AI through exploration and concrete examples,
- collaborate and discuss with peers how machines learn and why this is significant, and

- observe and explore the limits and capabilities of conversational agents through designing projects.

The curriculum also addresses the dimensions of computational thinking presented in the framework by Brennan and Resnick [10], as well as the artificial intelligence dimensions presented in Section 1.3.4. The next section describes how these dimensions are addressed in detail.

3.1 Relation to Computational Thinking and Artificial Intelligence Dimensions

In Section 1.3.4, I introduced Brennan and Resnick [10]'s framework for computational thinking (CT) to structure the workshop curriculum. I also introduced five new AI dimensions to supplement this framework, including classification, prediction, generation, training/validating/testing, and evaluating. The following tables describe how the CT and AI dimensions relate to the workshop curriculum.

Table 3.2: Applicability of the computational concepts found in [10] and the proposed AI concepts, 'symbolic rules' and 'machine learning', to conversational AI.

Computational Concept	**Relation to Conversational AI**
Sequences	Sequences are the backbone of all conversational AI. Words must be placed in a particular order; sentences must follow a particular flow; and conversational agents must take turns when they speak. Sequences are taught in the context of conversations.
Loops	When someone mishears or misunderstands a question, the question must be repeated. This is an example of a loop in conversational AI.
Events	When a conversational agent hears the "wake-word", it begins listening; when a conversational agent is asked a specific question, it responds. These are examples of events in the context of conversational AI.

Parallelism	While the user is speaking with a conversational agent, other events can occur on-screen in the app. This is an example of parallelism. Blocks in the conversational AI interface, as well as the regular MIT App Inventor interface can be executed at the same time.
Conditionals	To decide what happens in a conversation, conditionals can be used. For example, if Alexa receives the response, "dogs", to the question, "What's your favorite animal?", she could respond with "I agree! Dogs and zorillas are the best."; otherwise, she could respond with "That's cool.".
Operators	One example of an operator being used in conversational AI is the "equal to" operator. This may be used to check whether a user's response is equivalent to another.
Data	Data is important in conversational AI — especially for contextualization of the conversation. For example, the conversational AI agent could store a variable containing information about the conversation topic, such as a string containing "favorite animals".
Classification	During speech recognition, conversational agents classify parts of speech waveforms into phonemes, then words, and ultimately utterances.
Prediction	When using the *generate text* block in the conversational AI interface, a neural network iteratively predicts a best next letter until a complete sentence is formed. This block is discussed in detail in Section 2.5.2.
Generation	The *generate text* block generates sentences using pre-trained long-short term memory networks. This enables Alexa to respond with unique sentences. This block is discussed in detail in Section 2.5.2.

Table 3.3: Implementation of the computational practices found in [10] and the proposed AI-related practice, 'training, testing, and validating', in the conversational AI curriculum.

Computational Practice	**Curriculum implementation**
Being incremental and iterative	Students are encouraged to implement small changes to their conversational AI programs, test, and repeat.
Testing and debugging	Students are encouraged to test often and try to test as many possible user inputs as possible. For example, students are encouraged to draw diagrams outlining the intended conversational flow, and step through each possible branch, debugging as they go.

Reusing and remixing	Students develop a standard conversational AI program using a tutorial. They are then encouraged to modify it to be more interesting.
Abstracting and modularizing	Students modularize their code through function development. For example, they may create a function that decides whether a comment is positive or negative, and use this function in different parts of their program.
Training, testing, and validating	Students learn about machine learning and generative AI through the *generate text* block, as described in Section 2.5.2. This block has a drop-down menu that enables students to vary the time spent training and the training corpora of an LSTM network. Students can then test the models and observe the difference in output with various training times and corpora. Furthermore, they can determine whether a quickly-trained or extensively-trained model is more valid for their application. More information about concepts learned through experimenting with this block are discussed in Section 2.5.2.

Table 3.4: Applicability of the computational perspectives found in [10] and the proposed AI perspective, 'evaluating', to conversational AI.

Computational Perspective	**Implementation in proposed research**
Expressing	Programming conversational agents gives students the opportunity to express themselves through engaging others in creative conversations. One student may develop a goal-oriented conversational agent that enables users to move a robot with their voice, whereas another student may develop a social agent that asks users about their days. Students will be able to express themselves in many different ways through programming their own unique conversational agent.
Connecting	Programming conversational agents lends itself to connection since speech is conspicuous, and students will inevitably hear other students' agents. Students may combine agents with different abilities to create an agent that can respond to more than one student's programmed intent.

Questioning	AI is often opaque. It would be difficult to determine exactly how a conversational AI agent works without observing the code behind it. Even after observing the code, machine learning-based AI can still be difficult interpret. The curriculum includes discussion questions, such as "In what ways is symbolic AI better than machine learning and vice versa?" Students are encouraged to think deeply about conversational AI topics as well as ask questions.
Evaluating	It is important to evaluate technology and ask questions such as, "Does this program accomplish the task I intended it to accomplish?", "What are the limitations to my program? What are the capabilities?", and "Is this technology good for society?". Students will be encouraged to evaluate their conversational AI projects' quality, capabilities, limitations, and potential effects on users.

To further illustrate the connection between the CT and AI dimensions, and the workshop curriculum, detailed overviews of each workshop are shown in Appendix C.

Chapter 4

Results

The conversational AI workshop series for high school students provided insight into students' thoughts about conversational agents' intelligence, safety, and future. Fifteen students attended the workshops, ranging from 9th to 12th grade, with a median of 9th grade. The majority of the students had previous experience coding using blocks, and each student had unique ideas for conversational agent projects.

During the first workshop, students and their parents were provided with a consent form, which asked whether they would like to engage in the research and data collection. These forms are shown in Appendix B. A total of nine students returned the consent forms, although not all students completed the questionnaires. Seven students responded to the first questionnaire and five students responded to the second. The second questionnaire's lower response rate was due to low attendance during the final workshop[1]. Despite the fewer post-questionnaire responses, the general trends and student quotations gathered from the questionnaire are constructive and will be used to inform future research.

The data collected provided insight into the following research questions.

- Before and after engaging with the workshop curriculum, what do students believe, understand, and think about conversational AI agents?

- Before and after engaging with the workshop curriculum, what do students think

[1] As mentioned previously, the low attendance on the last day of the workshops may have been due to uncharacteristically good Boston weather. Many people in Boston were out enjoying the sunshine on April 6, 2019.

about AI agents' consciousness? How do they think AI differs from human intelligence?

- With a reasonable level of abstraction, can students develop their own conversational AI applications?
- Can students learn about the capabilities, limitations, and implications of conversational AI through workshops and developing conversational AI mobile apps?
- What concepts do students learn through this conversational AI curriculum? For example, do students learn in what ways symbolic AI and machine learning differ?
- What do students envision for the future of conversational AI?

These questions are addressed in the sections below.

4.1 Pre-Questionnaire: Students appreciated that conversational agents could solve problems without fully understanding how the agents worked

Before providing students with the initial questionnaire, we described and provided examples of conversational agents as follows:

Conversational agent: A machine that can use language (i.e., voice or text) to interact with humans. For example, chatbots, Google Home devices, Siri, and Amazon Alexa.

We asked students whether they had previously interacted with Amazon Alexa, and the answers varied greatly. One student piped up saying, "I have 6 Alexas at home!", whereas other students had never interacted with Alexa before.

After a brief introduction to conversational AI, we provided the students with a questionnaire which we will refer to as the pre-questionnaire (since it was filled out prior to the workshop curriculum). The pre-questionnaire consisted of four multiple choice questions in which students ranked statements on a five-point Likert scale. The results of these questions are shown in Table 4.1. The questionnaire can be found in Appendix D.

Table 4.1: The results of the Likert scale questions on the pre-questionnaire, where one corresponds to *strongly disagree* and five to *strongly agree*. In general, students had interacted with some sort of conversational agent, varied in their responses to whether they understood how they worked, and could think of how these agents could solve problems. Note that one student entered strongly agree (5) for every answer in the pre-questionnaire, despite his/her short answer responses not reflecting the same sentiment. Since there was a small sample size, this skewed the pre-questionnaire results upwards significantly.

#	Question	Mean	Range
1	I have interacted with conversational agents.	4.4	3-5
2	I understand how conversational agents decide what to say.	3.0	2-5
3	I feel comfortable making apps that interact with conversational agents.	3.3	1-5
4	I can think of ways that conversational agents can solve problems in my everyday life.	4.4	3-5

4.1.1 Pre-questionnaire Likert Scale Question Analysis

The first statement in the pre-questionnaire was, "I have interacted with conversational agents". Although students had not necessarily interacted with Amazon Alexa, they generally felt they had interacted with some sort of conversational agent before. These results are consistent with smart speaker research. In 2018, over twenty percent of US households were estimated to have smart speakers, which is a 78% increase from the year before [67]. Nonetheless, the results from the questionnaire suggested that students generally did not understand how conversational agents decided what to say, as the mean of the second question was approximately neutral, at 3.0. Also note that one student responded with *strongly agree* to each question, despite his/her short answer responses not reflecting this sentiment. This significantly skewed the results due to a small sample size.

Before the workshops, there was no consensus on whether students felt comfortable making apps that interacted with conversational agent. Answers for this question ranged the full breadth of the spectrum and averaged approximately neutral, at 3.3. Perhaps this was because certain students felt comfortable with the idea of developing conversational AI applications, whereas others did not yet know if they would be able to develop the applications.

The final Likert scale question stated, "I can think of ways that conversational agents

can solve problems in my everyday life". The answers to this question were all positive or neutral, suggesting that students had ideas for conversational AI applications even before learning about how conversational agents are developed. Students also responded with interesting ideas for conversational AI applications in the short answer portion of the questionnaire.

4.1.2 Pre-questionnaire Short Answer Question Analysis

The first short answer question asked, "Before hearing about this workshop, did you have any experience programming?" and if so, describe the previous experience. Out of the seven responses to this question, only one student had no prior programming experience. Other students' experience ranged from "a little [experience using] Scratch" to web development and Python programming.

The second short answer question asked, "Describe how you think conversational agents decide what to say." Many students answered this question generically; for example, stating "using algorithms". It was evident that these students did not have a clear idea of how conversational agents worked. Other students described rule-based AI concepts. One student wrote, "they have many responses programmed inside them [and] then based on what the user says, they choose a response from the list they have." None of the answers indicated any understanding of machine learning concepts.

The third question was, "What do you think are conversational agents' capabilities/limitations? What do you think they can/can't they do?" In this question, a number of students pointed out limitations of rule-based AI, and the difficulty of programming a response for every possible question. For example, one student stated, "I think that they have to be programmed for each question, and not every question can be thought of". Again, the students' responses did not indicate any prior understanding of machine learning. Another common theme in the responses included pointing out how conversational agents differ from human beings emotionally and in their thought processes. For instance, one student wrote, "[the agents] cannot think outside the box [or show] emotions. They [...] cannot interact with you as a real human".

> “Yes, [conversational agents are intelligent] to a non-emotional extent, just as intelligent as a search engine.”

The fourth short answer question asked whether students thought conversational agents are intelligent, and to what extent. In general, the students concluded that these agents are intelligent in some ways and unintelligent in others. For example, one student wrote, "Yes and no. They know lots of info and can find answers very quickly, but if you ask them something not part of the program, they can’t do anything". Similarly to their answers in the previous question, students pointed out how agents differ from humans. This included how the agents are not "emotionally" intelligent, as well as how they are only "as ’intelligent’ as they are programmed to be".

> “[Conversational agents] can’t comprehend what it’s like to be alive.”

The fifth question asked, "Do you think conversational agents are safe to use?" and to what extent. The majority of students stated they thought the agents are safe, albeit with reservations. Five out of the seven students had privacy concerns, stating "We don’t know if they are recording us" and "your prompts and responses can be recorded and sold to or hacked by someone who would use it against you. And this is assuming that they only record things directed at it", among other responses.

> “I think [conversational agents] are as "intelligent" as they are programmed to be. They are smart enough to respond to things the user says and the programmer has created responses for, but not smart enough to respond to things they have never heard about.”

The final question asked, "How might you use conversational agents in the future to create something that solves a problem you see in the world?" Responses ranged from

having the agents provide "therapy sessions" to "moderating conversations/arguments between people" to "find[ing] statistics of ongoing conflicts in today's world". Evidently, students could imagine interesting ways to use this technology.

> “ [Conversational agents] could replace traditional caretakers for people who either cannot see or cannot move to the extent needed. ”

All in all, the results from the pre-questionnaire suggested that the majority of the students had some programming experience, experience interacting with conversational agents, and could think of how conversational agents could be useful for solving problems. This was despite not having an understanding of how the agents worked or of machine learning, and stating concerns over privacy.

4.2 Workshop Activities: Students learned conversational AI concepts through remixing a storybook app and developing projects

During the workshops, we introduced conversational AI in the context of an application similar to Sheila's storybook app. This app modeled concepts including the VUI, endpoint function, rule-based AI, machine learning, and multimodal interaction. We provided students with worksheets which prompted them to describe how the block-based program worked. For example, one question asked which blocks changed the picture on the app's screen; another question asked which blocks defined how Alexa responded to utterances; and another question asked students to describe how they would modify the program to cause Alexa to respond with a unique sentence. This worksheet is shown in Appendix E.2.

During the following class, we discussed the solutions to this worksheet and provided time for students to remix the program in MIT App Inventor. The natural next step was for the students to start developing their own conversational agents. In the fourth class, stu-

dents brainstormed projects individually and in pairs, formed project groups and started developing projects related to the environment or ambient assisted living. Slides related to these topics, as shown in Figure 4-2 and 4-3 were presented to begin the brainstorming process.

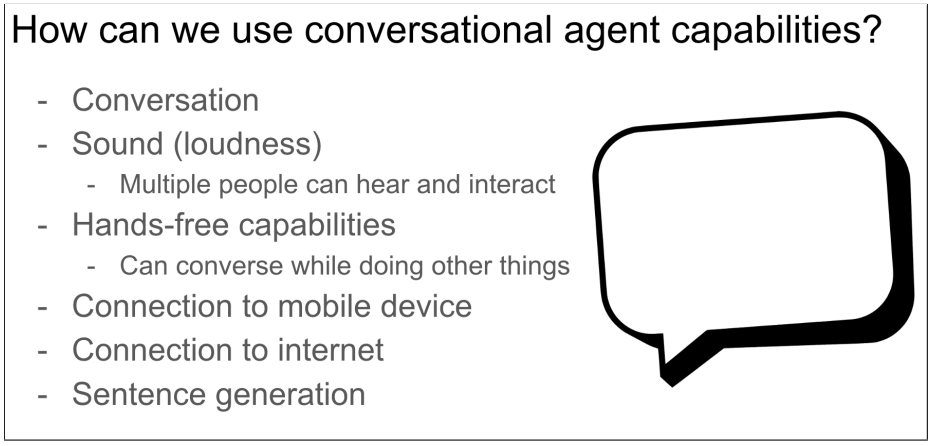


Figure 4-1: A slide from the workshops about the unique capabilities of conversational agents. Students were encouraged to leverage these capabilities in their final projects.

Note that in general, ambient assisted living technology is developed to help the elderly or those living with disabilities; however, we broadened the definition to allow students to help *anyone* live or function better. We chose ambient assisted living and environmental topics to emphasize the importance of using AI technology for good. Students were also encouraged to choose topics they were passionate and interested in.

During the project development classes, we also emphasized leveraging the unique capabilities of conversational AI and as mobile app technology. For instance, voice-based technology can be heard by multiple people at once, enabling group interaction; does not require hands, enabling concurrent multimodal interaction; and provides a natural interface for those who do not know how to type, such as non-digital-natives or young children. Figure 4-1 shows a slide used in the workshops illustrating conversational AI capabilities. Unique capabilities of mobile devices are shown in Figure 4-4.

During the workshops, we found certain teaching strategies, activities, and interface features could be improved. The following list describes challenges we faced and suggestions for future curricula.

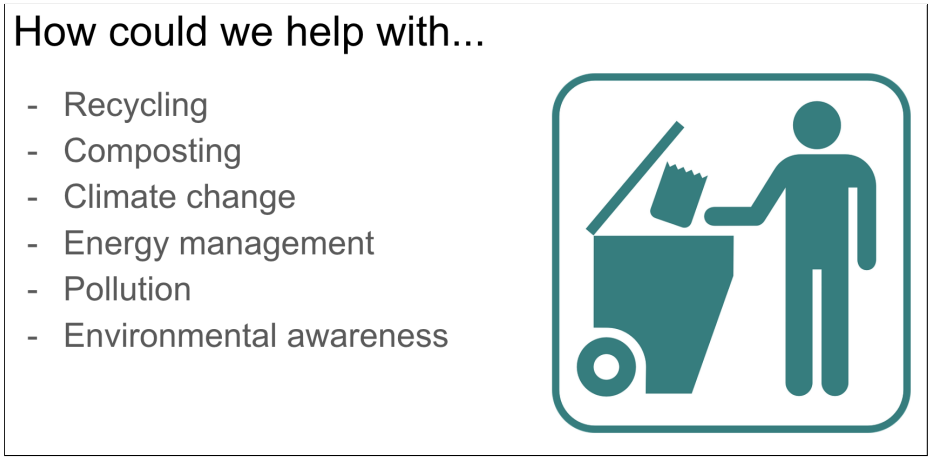


Figure 4-2: A slide from one of the workshops presenting some ideas for environmental applications. Students were encouraged to think of how conversational AI could help solve a problem they were passionate about related to the environment or ambient assisted living.

Figure 4-3: A slide from one of the workshops presenting some ideas for assisted living applications. Students were encouraged to think of conversational AI projects to help solve a problem related to the environment or ambient assisted living.

- The hour-long workshops went by very quickly, and not all students were able to complete the activities in class. I suggest increasing the length of each workshop to an hour and a half. The projects would also benefit from increased time.

- Since there was no way to ensure consistent attendance through the HSSP program and students were not required to complete work outside of class, some students fell behind. Perhaps organizing the workshops through a local high school would afford more consistent attendance and ensure students kept up with the material.

- Although the storybook application activity provided a good entry point for students to learn about the more complicated blocks, some students did not fully grasp the basic conversational AI programming principles before diving into this activity. I suggest providing a simpler activity, such as a "Hello World" Alexa Skill activity, prior to the storybook activity to ensure each student has a solid understanding of the basic principles before moving on.

- Some students forgot to give research permission slips to their parents, so they could not be involved in the study. Perhaps try organizing the workshops by oneself (instead of through HSSP) such that parents must sign permission slips prior to the workshops and the onus does not rest on the students to have the slips signed.

- By having a joint AWS account for students' Lambda functions, students did not need their own AWS account, and could rather send the teachers their endpoint JavaScript for uploading. Although this removed the burden of zipping and uploading a JavaScript file from the students, the non-linearity of emailing the teachers during development confused students. To simplify this process, we are developing a separate server (using the Jovo client [44]) for endpoint functions such that students may click a button and immediately run their endpoint functions.

- Although we removed the requirement for students to have an AWS account, students still found logging into their Alexa Developer account confusing. We plan to integrate an Alexa Skills testing console, where students can type in utterances and view Alexa's response, into the MIT App Inventor interface. With this, students would not have to log onto the Alexa Developer console to test their Alexa Skills.

All in all, through listening to short lectures, engaging in organized activities, and experimentally exploring the interface, students were able to learn conversational AI concepts, remix conversational AI programs, and develop unique conversational AI projects. The results from the pre- and post-questionnaires provide insight into students' learning and are discussed in Section 4.1 and 4.3. In the next iteration of the curriculum, certain aspects of the activities and interface will be updated to streamline the development process and improve the learning and project development timeline.

4.2.1 Student Projects: Students developed conversational AI applications to address problems in their communities

"Technology is best when it brings people together."

— Matt Mullenweg

As described in Section 4.2, students were encouraged to develop Alexa Skills and mobile apps that addressed a problem related to the environment or ambient assisted living. They were also encouraged to leverage the unique capabilities of conversational AI and mobile technology, as described in the slides in Figure 4-1 and 4-4. After brainstorming individually, students formed groups and decided on a final project that they were each interested in.

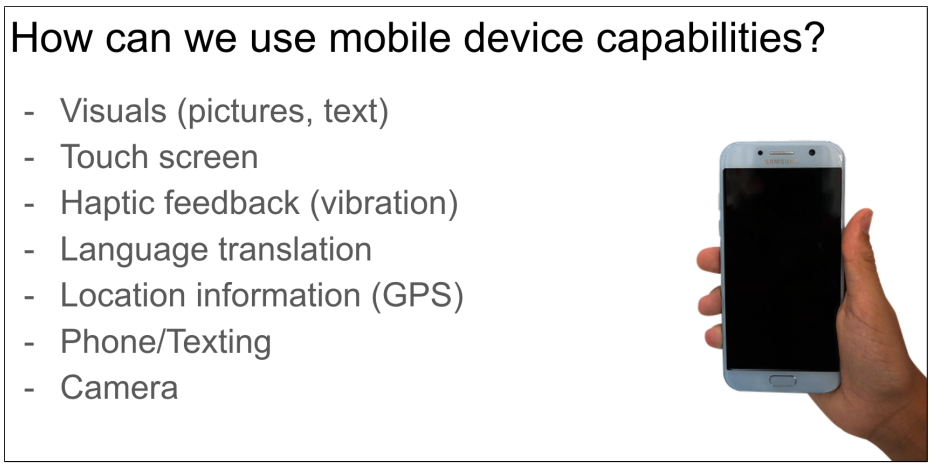


Figure 4-4: A slide from the workshops about the unique capabilities of mobile devices. Students were encouraged to leverage these capabilities in their final projects.

In total, there were five different project teams with one to four students on each team. The five projects were as follows.

1. **Memory Helper: Synonym Finder**

    - A tool to help people remember forgotten words
    - Example utterance: "Tell synonym finder the word is a fruit that grows in an orchard"
    - Example response: "Sounds like you might be thinking of the word, 'apple'"

- See example dialog from this Alexa Skill in Figure 4-5

2. **Can I recycle this?**

    - A tool to help people sort recyclables (or non-recyclables) correctly
    - Example utterance: "Can I recycle this plastic container?"
    - Example response: "Can you find a recycling symbol on the container?" ... (If yes) ... "What number is inside the symbol?"

3. **Alexa Speech to Text**

    - A tool to enable those with hearing loss to read Alexa's speech on an app
    - Example utterance: "Tell me a random fact"
    - Example response: "Although you will likely stop growing taller at around eighteen years old, your ears and nose will never stop growing"
        - This text is displayed on the mobile phone app as well as spoken aloud

4. **Math Teacher / Calculator**

    - A tool to help students with math
    - Example utterance: "What is three times five?"
    - Example response: "Three multiplied by five is the same thing as three groups of five items, which is fifteen in total."

5. **Cipher/Decoding Tool**

    - A tool to help decode hidden messages
    - Example utterance: "Decode 'KHOOR ZRUOG' using the Caesar cipher"
    - Example response: "Using the Caesar cipher, 'KHOOR ZRUOG' is 'HELLO WORLD'

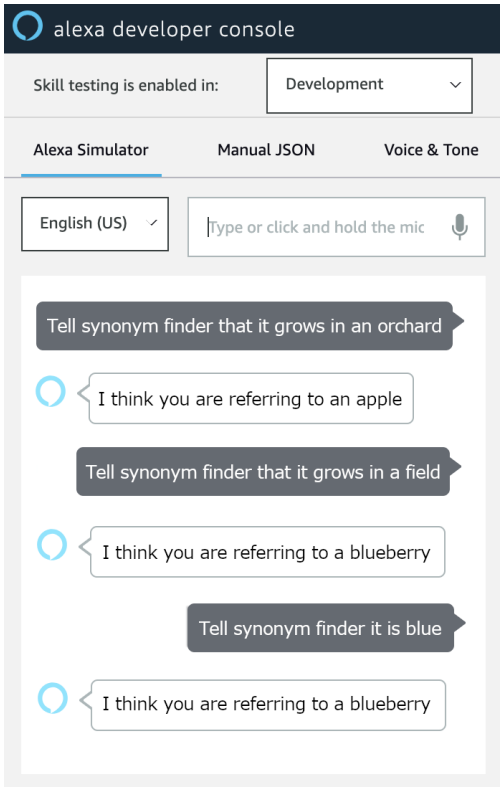


Figure 4-5: Example dialog from the *Synonym Finder* final project. This skill helped remind people of words they had forgotten.

> “I feel pretty good [about our project]. Our project could actually be used in our everyday life.”

Due to time constraints, not all students finished their projects. With an additional workshop, or additional time in the original workshops (e.g., 1.5 hour long workshops), students likely would have been able to develop more robust applications with additional features. Nonetheless, students generally responded positively when asked about how they felt about their projects, as discussed in Section 4.3.2. For example, one student stated about their project, "I feel good about it. I think it could actually help people and be put out into the world, if we put in some more work and make it function for more [cases]." Furthermore, as shown by the results in Section 4.3.2, students learned valuable AI concepts through the workshops and development process.

Table 4.2: The mean and range of the post-questionnaire Likert scale responses, where one corresponded to *strongly disagree* and five to *strongly agree*. Students generally felt they understood how conversational agents decide what to say, comfortable developing conversational AI applications, and their understanding of conversational agents increased through the workshops.

#	Question	Mean	Range
1	I have interacted with conversational agents.	4.6	4-5
2	I understand how conversational agents decide what to say.	4.4	3-5
3	I feel comfortable making apps that interact with conversational agents.	3.8	3-5
4	I can think of ways that conversational agents can solve problems in my everyday life.	4.4	4-5
5	My understanding of conversational agents improved through these workshops.	4.6	4-5

4.3 Post-Questionnaire: Students felt successful after developing ambient assisted living and environmental related conversational agents

Students completed the post-questionnaire (which can be found in Section D) during the final workshop. Although the sample size was too small to make any definitive conclusions, the results suggested students improved their understanding of conversational agents, could think of how conversational agents could be used to solve problems, and felt comfortable developing conversational agent applications. Future research may involve a larger study in a high school setting.

4.3.1 Post-questionnaire Likert Scale Question Analysis

The post-questionnaire contained four questions identical to the pre-questionnaire Likert scale questions, and a final question assessing whether students felt their understanding of conversational agents had improved. The results from these questions are shown in Table 4.2 and compared to the pre-questionnaire in Figure 4-6.

Notice that compared to the pre-questionnaire, the minimum value for each question

increased and the means increased or remained the same. This suggests students agreed more with each statement after the workshops, including whether they had interacted with conversational agents, their understanding of how conversational agents work, their comfort level in developing conversational AI applications, and their abilities to think of ways conversational agents can solve problems. The answer to the final question also supports these conclusions, with all students selecting either "agree" or "strongly agree" to the statement, "My understanding of conversational agents improved through these workshops".

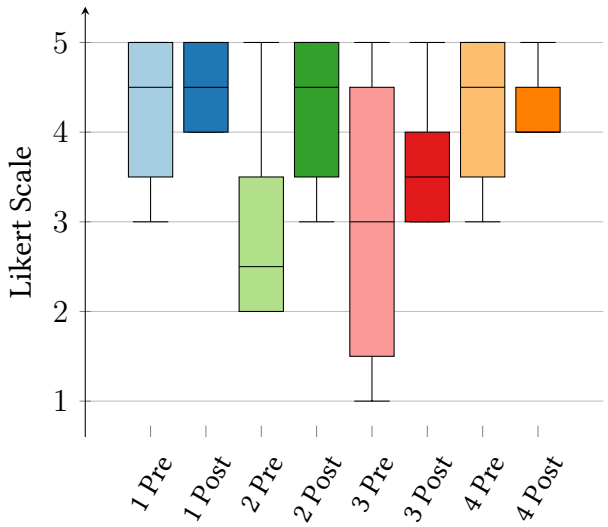


Figure 4-6: The results of the pre- and post-questionnaires. The darker colored boxes are the post-questionnaire results. Notice that the minimum values as well as the means of the responses increased or remained the same from the post-questionnaire to the pre-questionnaire. Note that one student entered strongly agree (5) for every answer in the pre-questionnaire, despite his/her short answer responses not reflecting the same sentiment. Since there was a small sample size, this significantly skewed the pre-questionnaire results upwards.

4.3.2 Post-questionnaire Short Answer Question Analysis:

"The best way to predict the future is to create it."

— Peter Drucker

The responses to the first short answer question, "How do you feel about the project you made?" were generally positive, and indicated students would have liked more time to

further develop their projects. For instance, one student stated, "[...] I am happy with the turnout and what I have learned. Hopefully, I will have the ability to extend my projects further in the future." Another student said, "I feel pretty good. Our project could actually be used in our everyday life." Other responses had similar sentiments.

> “I feel good about [our project]. I think it could actually help people and be put out into the world if we put in some more work and make it function for more [cases].”

The second short answer question, "What was hard about developing the project?", revealed that students initially found learning how to use the interface and certain conversational AI concepts difficult. For instance, one student mentioned, "It was at first difficult to understand the way in which intents are defined". Another student stated, "It was hard to use both the app part and the Alexa part, but I figured it out after some thinking." Despite these difficulties, students showed a solid understanding of conversational AI through their answers to the fourth question (as described below), as well as through their projects. This likely indicates that although high school students found the content of the workshops challenging, this grade level is a good fit, as the students were able to understand and use the concepts.

The responses to, "What were some things you didn’t expect when you were developing the project or learning in class?", suggested a general feeling of excitement and interest in the conversational AI interface. One student stated, "I did not expect that I would be one of the first people to use it, so that was very cool — to be able to use the beta". Based on the feedback, I would encourage other researchers interested in developing educational tools to do user testing early, even if it is still under development. In our experience, students were forgiving of the interface, especially since they saw how we were updating it as the workshops went on.

> “I did not expect that I would be one of the first people to use it, so that was very cool — to be able to use the beta.”

In the responses to the fourth and fifth questions, students demonstrated their understanding of machine learning and rule-based AI. The majority of the students understood that machine learning algorithms can change and "learn" in response to input, whereas rule-based AI generally responds the same way to the same input no matter how much input it is given. For instance, one student described machine learning as "morph[ing] over time in response to input" and rule-based AI as "always respond[ing] the same way to the same stimulus or intent". Another student stated that machine learning is "more prone to make mistakes" whereas rule-based AI "has limitations towards how much it can do with the same amount of time spent [programming each rule]". These answers are consistent with the LSTM and rule-based Alexa Skills we presented in class.

> “Machine learning evolves as it gets more experience in the world. Rule-based A.I. provides the same results every time which are not very variable.”

The sixth question asked, "What do you think are conversational agents' capabilities/limitations? What do you think they can/can't they do?" Students' answered varied for this question. Some students focused on the limitations of the Alexa skills interface, stating that "The agent can't connect with [...] another website to get information from it" or "They sometimes can't understand what you might try to say as they might have [...] different invocation words for an action than the one you might be using". Other students focused on broader conversational agent technology, stating that "The way [agents] interact with new situations or stimuli may never be quite as human as we'd like" or "They cannot [perceive] the human's emotions or feelings". In terms of capabilities, students generally responded positively, making statements like "They are capable of solv[ing] problems and help[ing] people", "They can read things for people, suggest words, inform people, and many other things" and "They [...] will definitely improve over time".

> “ [Conversational agents] are capable to solve problems and help people. ”

In their responses to whether they thought conversational agents were "intelligent", students generally gave a balanced answer, stating things like, "They are intelligent with a human’s help, but on [their] own they are completely useless [since] they go off of words spoken by people" and "I think they are [intelligent] because they can learn as they go along with machine learning to choose the better response to an input rather than having a hard coded response. They aren’t intelligent from the start because they haven’t learned much yet". One student cited the definition of intelligence as "the ability to acquire and apply knowledge and skills" and deemed conversational agents as intelligent since "conversational agents, especially machine learning based ones, certainly learn from the past and apply that knowledge".

Students generally felt the agents were safe to use, but had privacy and security concerns. For instance, one student stated, "They are mostly safe, but the danger can come from data collection or spying". Another student stated "I think they are safe to use but they can become dangerous quickly because others could be listening to you and could steal your personal information". It would be interesting to do further research in this area to determine the general sentiment on conversational agent safety for different age groups.

> “ They are mostly safe, but [...] if you tell an untrustworthy program information about you, someone could try to use it to steal your identity. ”

The responses to the final question, "If you were to use conversational agents in the future to create something that solves a problem you see in the world, what would it be?", yielded ideas ranging from a "tool that CADs what you explain to it by using what you say to draw images" to "language translation". One student’s answer brought up an interesting case study: "[The agent] would be a guidance counselor but in a conversational agent

so it would be able to increase confidentiality and reliability in counselors". It would be interesting to ask students whether conversational agents truly increase confidentiality and whether it is okay that they send data to companies like Amazon and Google. Nevertheless, the ideas presented in the responses showed an overall hopefulness and excitement for the future of conversational AI.

On the whole, although only five students completed the post-questionnaire, the results were informative and useful to guide future research. The main takeaways included:

- Students felt proud of their project development and were interested in continuing to develop them more fully
- Students learned conversational AI concepts, including the difference between machine learning and rule-based AI, despite finding such concepts difficult at first
- Students would have liked to have more time to develop projects and learn concepts
- Despite using an interface that was still under development, students found it exciting to be the first users
- Students could identify capabilities and limitations of conversational agents
- Students identified privacy and security implications of conversational agents
- Students could think of positive applications for conversational agents
- Students were generally hopeful and excited about the future of conversational AI

Evidently, through these workshops, students learned conversational AI principles, took great interest in conversational agent development, and were able to design socially useful Alexa Skills.

Chapter 5

Discussion and Conclusions

With the rapid rise of conversational AI, it is increasingly important for people to understand the capabilities, limitations, and implications of AI. In this work, I developed tools to empower students to develop conversational agents and learn about artificial intelligence. Through high school workshops, students learned artificial intelligence concepts, including topics such as generative AI, machine learning, and rule-based AI; conversational AI concepts, including topics such as utterances, intents, and endpoints; and developed conversational agents to address problems they saw in the world. The deliverables of this work include:

- Conversational AI (Alexa Skills) interface for MIT App Inventor
- *Generate text* (LSTM) extension for MIT App Inventor
- Conversational AI curriculum for six workshops
- Analysis of pre- and post-questionnaires from workshops

5.1 Students can make positive change through technology development

Through the interface development, workshops, and questionnaire analysis, my key findings were:

- **The MIT App Inventor conversational AI interface empowered students to develop Alexa Skills using blocks.** Block-based coding tools, such as MIT's Scratch or MIT App Inventor, have been enabling children as young as primary school students to program for over a decade. By leveraging MIT App Inventor's low barrier to entry for programming, the conversational AI interface enables nearly anyone to develop conversational agents.

- **The *generate text* block provides an entry point for learning about ML.** The *generate text* block provides a highly abstracted way to implement machine learning in mobile apps and conversational agents. By experimenting with LSTM models pre-trained for different number of epochs and on various input corpora, students can learn how training a ML model in different ways affects the model's output.

- **Students learned conversational AI concepts, such as the difference between ML and rule-based AI, conversational AI terminology, through the workshop curriculum.** Other workshop topics included the capabilities, limitations and implications of AI, how to program mobile applications and conversational agents, and understanding and implementing voice user interfaces, endpoint functions, utterances, invocation names, and intents.

- **Despite initially finding conversational AI concepts challenging, students were able develop complex, purposeful conversational AI agent projects.** Developing Alexa skills is complicated; however, through block-based coding tools and workshop curriculum, students were able to develop conversational AI projects.

- **Students were optimistic about the future of conversational AI.** Through the questionnaire responses, it was clear that students could think of interesting, positive applications for conversational AI, and thought the technology would improve with time.

- **Students' conversational AI projects were positive and purposeful.** From a memory aid to a recycling management tool, students' projects addressed problems they saw in the world.

As evidenced by the workshops, students were passionate about technology development for a better world. It is my belief that it is not technology itself that changes the world, it is what *we do with* and how *we develop* technology that changes the world. In these workshops, students developed powerful, purposeful projects to address problems in their communities. In the future, it will be increasingly important to emphasize the implications of AI and how to use this technology wisely. Through positive technology development education, we can move towards a safe, positive, AI-filled future.

Chapter 6

Future Work: Interface Improvements, Natural Language Programming, and AI Education

There are many opportunities to extend this work, including developing a more robust, scalable conversational AI interface, updating the workshop curriculum, and developing further tools to democratize conversational AI. The following list describes some such opportunities.

- Conversational AI interface
  - **Increase the usability of the interface.** The current interface requires copying JavaScript generated by MIT App Inventor to upload it to AWS Lambda. In future iterations, we plan to automatically deploy endpoint functions for Alexa Skills either using the Jovo framework or working with Amazon to deploy AWS Lambda functions through an API or other means [44].
  - **Integrate the interface into the public-facing version of MIT App Inventor.** The current conversational AI interface is running on a low-traffic server specifically for the HSSP workshops. Before adding it to a public-facing version of App Inventor, we need to complete further user testing, debugging, and code review.

- **Create additional blocks and extensions.** Conversational agents are commonly used to query the web. For instance, someone may ask Alexa what the weather is like, and Alexa will respond with information retrieved by querying a weather database. In the workshops, the students developing the Synonym Finder project wanted to access an online thesaurus. This might be achieved through developing a query block (e.g., HTTP request and/or API query blocks), and/or through the GraphQL component, which is currently under development.

- Workshop Curriculum
  - **Increase the time allotted for each workshop.** In the questionnaires, students noted that they would have liked more time to develop projects and complete activities in class. In the next iteration of the workshops, the time allotted for each workshop will be increased to 1.5 hours.
  - **Integrate more AI ethics discussion into the curriculum.** Ethics are vitally important and highly relevant to the AI technology development process. Although our curriculum touched on the subject, I would like to dedicate more time to this topic with an extended workshop series including an ethical discussion each class.
  - **Add voice-first design principles to the curriculum.** As noted in Section 1.3.3, voice-based interfaces are very different from visual interfaces, and this should be taken into account during design. With an extended workshop series, voice-first design principles could be added to the curriculum.
  - **Add simple *Hello World* example to the curriculum.** As mentioned in Section 4.2, the complexity of the storybook Alexa Skill was difficult for some students to grasp. In the next iteration of the curriculum, a simpler *Hello World* Alexa Skill with fewer blocks will be introduced before the storybook skill.

- Other conversational AI democratization tools
  - **Implement Google Actions in the conversational AI interface.** We are

currently modifying the backend of the conversational interface enable Google Action development using the same Alexa Skills blocks. This feature utilizes the Jovo library for JavaScript [44].

- **Develop Natural Language Programming tools.** Voice-base interfaces simplify technology interaction through natural language. For example, those who do not know how to type, can query the web by talking to Alexa or Google Home devices. I am currently developing a voice interface for programming a conversational agent.

Through this research, I have had the opportunity to engage with students who are enthusiastically optimistic about the future of conversational AI. With an improved conversational AI interface, refined workshop curriculum, and additional democratization tools, students will be able to develop even more powerful tools. I hope this thesis will inspire students to develop positive, socially useful technologies for a better future.

Appendix A

Code

This appendix contains source code from the conversational AI interface in MIT App Inventor.

A.1 JavaScript Endpoint Header

The following code snippet is placed at the top of the Node.js JavaScript file that is generated for the Alexa Skill endpoint.

Listing A.1: The header of the Node.js Javascript file that defines the Alexa Skill endpoint. It includes constants, functions, and import statements to enable specific endpoint block functionalities.

```
1  const redis = require('redis');
2  const urlHostPort = 'rediss://clouddb.appinventor.mit.edu:6381';
3  const SET_SUB_SCRIPT = "ITS_A_SECRET_TO_EVERYBODY";
4  async function setCloudDB(key, value, projectName, cloudDBAuthKey) {
5    let client = redis.createClient(urlHostPort, {
6      'password': cloudDBAuthKey,
7      'tls': {}
8    });
9    let json_value = JSON.stringify(value);
10   return new Promise((resolve, reject) => {
11     client.evalsha(SET_SUB_SCRIPT, 1, key, json_value, JSON.stringify([
         json_value]), projectName, (err, res) => {
```

```
12        if (err) console.log(err);
13        client.end(true, (e, r) => console.log(e));
14        resolve("Yes");
15      });
16    }));
17  }
18  async function getCloudDB(key, projectName, cloudDBAuthKey) {
19    let client = redis.createClient(urlHostPort, {
20      'password': cloudDBAuthKey,
21      'tls': {}
22    });
23    return new Promise(((resolve, reject) => {
24      client.get(projectName + ':' + key, (err, value) => {
25        if (err) console.log(err);
26        client.end(true, (e, r) => console.log(e));
27        resolve(value);
28      })
29    }));
30  }
31  const fetch = require('node-fetch');
32  const csail_url = 'http://appinventor-alexa.csail.mit.edu:1234/';
33  const getText = async function (baseUrl, input, model, length) {
34    let json;
35    let url = baseUrl + "?inputText=" + input + "&model=" + model + "&
        outputLength=" + length;
36    try {
37      const response = await fetch(url);
38      json = await response.json();
39    } catch (err) {
40      console.log(err);
41    }
42    return json;
43  };
44  const Alexa = require('alexa-sdk');
45  const handlers = {
46    'LaunchRequest': function () {
```

```
47      if (handlers["AMAZON.HelpIntent"]) {
48        this.emit('AMAZON.HelpIntent');
49      } else {
50        this.response.speak("This Skill was created with MIT App Inventor.
              ");
51        this.emit(':responseReady');
52      }
53    },
54    /* --- User defined functions go here --- */
```

A.2 JavaScript Endpoint Footer

The following code snippet is placed at the bottom of the Node.js JavaScript file that is generated for the Alexa Skill endpoint.

Listing A.2: The footer of the Node.js Javascript file that defines the Alexa Skill endpoint. It includes the 'unhandled' callback and the handler export function.

```
1   /* --- User defined functions end here --- */
2   'Unhandled': function () {
3     this.response.speak("I don't know what that phrase means. Make sure
          each 'define intent' block has a corresponding and 'when intent
          spoken' block.");
4     this.emit(':responseReady');
5   },
6 };
7 exports.handler = function (event, context, callback) {
8   const alexa = Alexa.handler(event, context, callback);
9   alexa.registerHandlers(handlers);
10  alexa.execute();
11 };
```

Appendix B

Assent and Consent Forms for Human Subjects Research

This appendix contains the assent and consent forms provided to and approved by MIT's Committee on the Use of Humans as Experimental Subjects (COUHES).

B.1 Assent Form

The following assent form was provided to students who attended the high school workshops.

ASSENT TO PARTICIPATE IN RESEARCH
(For students under 18)

Learning to Program Conversational Artificial Intelligence with MIT App Inventor and Amazon Alexa

1. My name is Jessica Van Brummelen and I'm a graduate student at MIT.

2. We are asking you to take part in a research study because we are trying to see if students can learn to create their own conversational artificial intelligence apps using MIT App Inventor, Amazon Alexa, and the computer programming skills that we'll be teaching. *Conversational artificial intelligence* is the ability for a computer to have conversations with humans. *Amazon Alexa* is a voice-based technology that has the ability to talk with people. *MIT App Inventor* is a tool that anyone can use to develop his or her own smartphone or tablet apps.

3. If you agree to be in this study, you will complete six workshops that will teach you about computer programming, MIT App Inventor, and conversational artificial intelligence. The workshops will involve short lectures about these topics, tutorials about how to use MIT App Inventor, time for you to program apps and conversational artificial intelligence, and a final project. During the workshops, you will be asked to fill out surveys online and talk about what you have learned. For example, you may be asked, "What's the most interesting thing you've learned so far?". We will use your responses, and information about the apps you created for our study. When we look at the apps and your responses, we won't be judging your abilities or knowledge, but rather judging the MIT App Inventor interface to see if it is a good interface for learning. If you decide at any time that you no longer want to participate, you are completely free to stop participating in the workshops and/or study. There is no penalty for not participating. You may continue completing the workshops without participating in the study.

4. In the workshops, you will be using a computer, which may put you at risk for eyestrain, back strain, and/or other strain related to computer use. The maximum amount of time spent on a computer during the workshops would be for one hour once a week, so it is unlikely you will experience strain. If you do experience strain, let the instructor know, and you may take a break from using the computer.

5. By participating in this study, you will likely learn about advanced technology and learn computer-programming skills. These skills and knowledge will likely be valuable for your future, whether you decide to continue to pursue computer programming, you will be working with people who computer program in the future, or you learn that you would rather not work with computers in the future. If you choose to work with others on a final project, this will also give you valuable teamwork experience.

6. If you are under the age of 18, please talk this over with your parents before you decide whether or not to participate. We will ask your parents to give their permission for you to take part in this study. Note that even if your parents say "yes" you can still decide not to participate.

7. If you don't want to be in this study, you don't have to participate. Remember, being in this study is up to you and no one will be upset if you don't want to participate or even if you change your mind later and want to stop.

8. You can ask any questions that you have about the study now. If you have a question later, you can call me at 1-857-928-2806 or ask me next time you see me. You can also call the Chairman of the Committee on the Use of Humans as Experimental Subjects at M.I.T. at 1-617-253-6787 if you feel you have been treated unfairly.

9. Signing your name below means that you agree to participate in this study. You (and your parents if you are under the age of 18) will be given a copy of this form after you have signed it.

_____
Printed Name of Subject

_____ _____
Signature of Subject Date

B.2 Consent Form

The following consent form was provided to students who attended the high school workshops for their parents to sign.

CONSENT TO PARTICIPATE IN
NON-BIOMEDICAL RESEARCH
(For students 18 or older, or parents/guardians of children under 18)

Learning to Program Conversational Artificial Intelligence with MIT App Inventor and Amazon Alexa

You are asked to participate in a research study conducted by Jessica Van Brummelen, B.A.Sc., Yulia Gonik, Terryn Brunelle, Evan Patton, Ph.D., and Hal Abelson, Ph.D., from the Electrical Engineering and Computer Science department at the Massachusetts Institute of Technology (M.I.T.). The results of this study will contribute to Jessica Van Brummelen's master's thesis.

You were selected as a possible participant in this study because you applied to the HSSP Programming Amazon Alexa with MIT App Inventor course. The purpose of this project is to determine whether students can understand conversational artificial intelligence (the ability for a computer to have conversations with humans) through workshops and developing programming projects. Students will complete six one-hour workshops that will teach them about conversational artificial intelligence (AI) using Amazon Alexa, and programming using a visual drag-and-drop coding interface called MIT App Inventor (http://appinventor.mit.edu/explore/about-us.html). Students will learn to develop mobile phone and tablet applications, as well as "Alexa Skills", which are voice-based applications for Alexa. If participating in this study, students will complete interviews and questionnaires about what they have learned and created during the workshops. The questionnaires/interviews will be held during the one-hour workshops and take approximately 15 minutes or less each week. Students may experience computer related strain (e.g., eyestrain) if participating in this study.

You should read the information below, and ask questions about anything you do not understand, before deciding whether or not to participate.

- **PARTICIPATION AND WITHDRAWAL**

Your participation in this study is completely voluntary and you are free to choose whether to participate or not. If you choose to be in this study, you may subsequently withdraw from it at any time without penalty or consequences of any kind. The investigator may withdraw you from this research if circumstances arise which warrant doing so. For instance, if you do not attend a significant number of the course workshops, you may be withdrawn from the study. Withdrawing from the study does not mean you need to withdraw from the course. You may continue participating in the course if you withdraw from the study.

- **PURPOSE OF THE STUDY**

This study will investigate whether and to what extent students can learn about conversational artificial intelligence (AI) through developing their own smartphone apps and Amazon Alexa skills using MIT App Inventor. Research questions include, "Are students able to understand how conversational AI applications are developed?", "With a reasonable level of abstraction, can students develop their own conversational AI applications?", and "After attending workshops and developing conversational AI mobile apps, do students understand the capabilities, limitations and implications of conversational AI?".

- **PROCEDURES**

If you volunteer to participate in this study, we would ask you to do the following things:

- Participate in the HSSP "Conversational Artificial Intelligence with MIT App Inventor and Amazon Alexa" course workshops
    - The workshops include:
        - Short lectures about computer programming, MIT App Inventor, conversational artificial intelligence (AI), Amazon Alexa, and related topics
            - *MIT App Inventor* is a tool to empower anyone to develop their own smartphone and tablet applications
            - *Conversational AI* is the ability for a computer to carry out intelligent conversations
            - *Amazon Alexa* is a voice-based technology that has conversational AI abilities
        - Tutorials about how to use MIT App Inventor
        - Free time for developing smartphone/tablet and conversational AI apps using MIT App Inventor
- Complete surveys online and short interviews in which you write or talk about what you have learned in the workshops
    - These surveys/interviews will ask you about what you have learned about computer programming, smartphone/tablet applications, conversational AI, and related topics
    - These surveys/interviews will not be judging you on your abilities or understanding, but rather used to study whether the MIT App Inventor interface and workshops are good platforms for learning
    - There will be a total of two online surveys, which will take approximately 15 minutes or less to complete
    - Interviews will occur during the workshops and may include questions such as, "What is the most interesting thing you have learned so far?"
    - Time will be given within the workshops to complete the surveys and interviews

The workshops will be held for one hour every Saturday from Feb. 23rd 2019 to April 6th 2019 (excluding March 16th for a holiday) in a computer lab on the M.I.T. campus.

During the workshops, you will choose partners for a final project (or, if you prefer, choose to work on your final project alone), develop Alexa Skills and mobile apps, learn about programming and conversational AI, and complete surveys and interviews, as described above.

- **POTENTIAL RISKS AND DISCOMFORTS**

You will be using computers and may experience eyestrain, back strain, and/or other strain related to computer use. The maximum amount of time spent on a computer during the workshops would be for one hour once a week, which should not pose a significant risk of strain. If you do experience strain, let the instructor know, and you may take a break from using the computer.

- **POTENTIAL BENEFITS**

By participating in this study, you will likely learn about advanced technology and learn computer-programming skills. These skills and knowledge will likely be valuable for your future, whether you decide to continue to pursue computer programming, you will be working with people who computer program in the future, or you learn that you would rather not work with computers in the future. If you choose to work with others on a final project, this will also give you valuable teamwork experience.

Additionally, through this research, written works (e.g., research papers) will be created explaining the effectiveness of a visual coding interface and workshops in teaching students how to program conversational AI applications. This will likely help future educators and software developers to learn how to best teach and develop software related to conversational AI. As well, the applications developed by the students may solve real-world problems, and if the students decide to release their app, this app could benefit society in general.

- **PAYMENT FOR PARTICIPATION**

You will not receive payment for participating in this study.

- **CONFIDENTIALITY**

Any information that is obtained in connection with this study and that can be identified with you will remain confidential and will be disclosed only with your permission or as required by law. In addition, your information may be reviewed by authorized MIT representatives to ensure compliance with MIT policies and procedures.

All personal information, research data, and related records that are physically collected will be stored in a locked file cabinet in a locked room that only members of the research team can access. All personal information, research data, and related records that are computer-based will be anonymized, encrypted, and securely stored on an MIT computer.

The data will be retained for at least five years following analysis. After the five-year period, if the data is no longer useful for research, it will be deleted. In academic reports (e.g., academic paper or master's thesis), the data will be reported as anonymized statistics, or as anonymous quotes or pictures of drawings (if students drew answers to survey questions).

- **IDENTIFICATION OF INVESTIGATORS**

If you have any questions or concerns about the research, please feel free to contact any of the following people:

- Principal Investigator: **Hal Abelson**
  - Phone number: (617) 253-5856
  - Address: MIT Computer Science and Artificial Intelligence Laboratory
    Room 32-G516, The Stata Center
    32 Vassar Street
    Cambridge, MA 02139
- Co-Investigator: **Jessica Van Brummelen**
  - Phone number: (857) 928-2806
  - Address: MIT Computer Science and Artificial Intelligence Laboratory
    Room 32-G528B, The Stata Center
    32 Vassar Street
    Cambridge, MA 02139
- Co-Investigator: **Evan Patton**
  - Phone number: (401) 484-7865
  - Address: MIT Computer Science and Artificial Intelligence Laboratory
    Room 32-G506, The Stata Center
    32 Vassar Street
    Cambridge, MA 02139
- Co-Investigator: **Yulia Gonik**
  - Phone number: (949) 413-2178
  - Address: MIT Computer Science and Artificial Intelligence Laboratory
    Room 32-G506, The Stata Center
    32 Vassar Street
    Cambridge, MA 02139
- Co-Investigator: **Terryn Brunelle**
  - Phone number: (603) 305-2515
  - Address: MIT Computer Science and Artificial Intelligence Laboratory
    Room 32-G506, The Stata Center
    32 Vassar Street
    Cambridge, MA 02139

- **EMERGENCY CARE AND COMPENSATION FOR INJURY**

If you feel you have suffered an injury, which may include emotional trauma, as a result of participating in this study, please contact the person in charge of the study as soon as possible.

In the event you suffer such an injury, M.I.T. may provide itself, or arrange for the provision of, emergency transport or medical treatment, including emergency treatment and follow-up care, as needed, or reimbursement for such medical services. M.I.T. does not provide any other form of compensation for injury. In any case, neither the offer to provide medical assistance, nor the actual provision of medical services shall be considered an admission of fault or acceptance of liability. Questions regarding this policy may be directed to MIT's Insurance Office, (617) 253-2823. Your insurance carrier may be billed for the cost of emergency transport or medical treatment, if such services are determined not to be directly related to your participation in this study.

- **RIGHTS OF RESEARCH SUBJECTS**

You are not waiving any legal claims, rights or remedies because of your participation in this research study. If you feel you have been treated unfairly, or you have questions regarding your rights as a research subject, you may contact the Chairman of the Committee on the Use of Humans as Experimental Subjects, M.I.T., Room E25-143B, 77 Massachusetts Ave, Cambridge, MA 02139, phone 1-617-253 6787.

SIGNATURE OF RESEARCH SUBJECT OR LEGAL REPRESENTATIVE

I understand the procedures described above. My questions have been answered to my satisfaction, and I agree to participate in this study. I have been given a copy of this form.

_____
Name of Subject

_____
Name of Legal Representative (if applicable)

_____ _____
Signature of Subject or Legal Representative Date

SIGNATURE OF PERSON OBTAINING INFORMED CONSENT

In my judgment the subject is voluntarily and knowingly giving informed consent and possesses the legal capacity to give informed consent to participate in this research study.

_____
Name of Person Obtaining Informed Consent

_____ _____
Signature of Person Obtaining Informed Consent Date

Appendix C

Workshop Lesson Plans

This appendix contains the curriculum and lesson plans for six high school workshops. Each lesson is approximately fifty minutes long with a ten minute buffer.

Lesson 1: Introducing App Inventor (50 minutes with 10 minute buffer)
Pre-workshop preparation:

- Print out consent and assent forms, and staple together
    - Make sure to print double the number of forms as there are students so that the students can keep a copy
- Send an email to students about making an account on MIT App Inventor and the Amazon Developer Console (click "sign-in" on the top right, and then create an Amazon Developer Account)

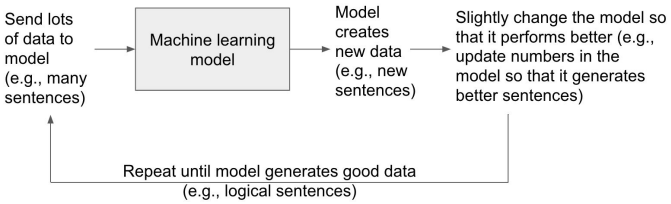
Time	Activity
5 min	Class introductions ● Name, grade, why are you taking this class, any programming experience you have (Scratch, Java…)
5 min	What is App Inventor? ● A website that lets you design and make apps! (Show site: http://appinventor.mit.edu/explore/) ● Have the students go to the App Inventor website ● Have the students open a new tab and navigate to the development page What is Amazon Alexa? ● A conversational agent that can be spoken to and used on tablets and phones via the Alexa App, or on other Alexa-enabled devices Make sure everyone has downloaded the App Inventor Companion app and the Alexa App on their mobile devices and/or the provided tablets. Let students create their own accounts on the Amazon site and the MIT App Inventor site, and figure out how to use App Inventor at their own pace.
40 min	Have students go through a list of tutorials. Encourage them to talk to other students in the class if they have questions. When they are done with the first app, teach them individually how to download the app through the QR code and APK method. Beginner tutorials (Please try to finish all of these): ● Talk to Me: Part 1 and Talk to Me: Part 2 ● My Favorite Things (provide joke pictures and recordings) ● Paint Pot ● QuizMe ● Minigolf Hand out consent/assent forms to students (and parents, if applicable).

Lesson 2: Introducing Conversational AI (50 minutes with 10 minute buffer)
Pre-workshop preparation:

- Print out Conversational AI Rules worksheets

- Print out Alexa Blocks Handouts
- Print out "Talking to a Storybook" Worksheets
- Update links doc with any links below

Time	Activity
5 min	Introduce the concept of conversational artificial intelligence ● Ask students if they have ever interacted with a chatbot / conversational agent ○ What are some examples? (E.g., Siri, Alexa, etc.) ● How do they think conversational agents work? ● How realistic are today's conversational agents? Explain that researchers today are working to develop more realistic chatbots, or "conversational AI agents" ● Show Google Duplex (1:23-2:18) ● How does Google Duplex compare to human interactions? ○ What makes Google Duplex realistic? ○ What would make Google Duplex more realistic?
10 min	Have students fill out the pre-questionnaire if they agreed to participate. Students who are not participating in the study may work on projects in MIT App Inventor.
10 min	Introduce the "Conversational Agent Rules" example: ● Context: You want to create a conversational agent that can control your smart-house (e.g., smart TV, smart light bulbs, etc.) ● Ask: What rules do you set out for your agent? ○ e.g., "If you ask the agent to turn on the lights in the living room, then the agent responds with 'Sure, I'll turn on the living room lights', and sends a signal to the lights to turn on ● Explain: ○ The "Voice User Interface" or "VUI" is the part of the program that listens for questions from the user (listens for things) ○ The "endpoint" is the part of the program that responds to the users' questions (does things) Let's try it out! (Pairs of students) ● In pairs, write out all the rules you think are necessary to control a smart home ○ Think about: What happens when you ask your agent something it has never heard before? ○ Fill out this Conversational AI Rules worksheet ● Try it out! ○ One person is the agent and one person is the human ○ Human: ■ Ask the agent questions ■ Think of questions that the agent may or may not be able to answer

	○ Agent: ■ Respond to questions according to the rules that you created ● What was hard? What did you *want* the agent to be able to do, but it couldn't based on your rules?
5 min	Was it difficult to write out rules for every situation possible? ● Explain how computer scientists have created networks/models that can "learn" or change in order to generate new data **Training a machine learning model**  ● Once a machine learning model has been trained, it can be used to classify things (e.g., recognize images) or create things (e.g., generate sentences) ● One example of a machine learning model is an "LSTM", which can be used to generate sequences, like sequences of letters or words ● Machine learning models don't need predefined rules to work ● Introduce the LSTM or "sentence generator" block in App Inventor ○ Generates sentences based on the input data it was trained on, and an initial "seed text" ○ "Epochs" represent the amount of training
5 min	Motivate the Conversational AI Interface by reviewing the "Talking to a Storybook" app / Alexa Skill and what it does (allows users to interact with the storybook character, Karabo, by talking to Alexa) ● I.e., show the "Talking to a Storybook" app on your phone and with Alexa ● Ask students if they think they could make this with App Inventor
5 min	Introduce the Conversational AI Interface, and show the VUI and endpoint blocks. This will allow you to program Amazon Alexa. Blocks with descriptions are included in the Alexa Blocks Handout. ● The VUI blocks allow you to create different "intents" or questions that Alexa can answer ● The endpoint blocks allow you to program how Alexa responds to those "intents" or questions
10 min	Pair the students up and have them do the "Talking to a Storybook" Worksheet ● Tell them that they will figure out and code their own "Talking to a Storybook" Alexa Skill / app ● Give them the partially complete AIA for the app. It is without the Alexa Skill blocks and currently just has an app where you can flip through storybook pages

	● They should make a plan for how to improve the Alexa skill according to the worksheet requirements

Lesson 3: "Talking to a Storybook" Tutorial Wrap-up and Project Introduction (50 minutes with 10 minute buffer)

Pre-workshop preparation:

- Print out "Talking to a Storybook" solutions
- Update links doc with any links below
    - Also include the "Making a Simple Alexa Skill" link (or print this out)

Time	Activity
15 min	Give the students the "Talking to a Storybook" solutions printout. Pair the kids up with new people than last time and have them share and discuss their solutions to the "Talking to a Storybook" Worksheet from the last lesson. Let them try to pair program a working app. Walk around to see if any pairs figured it out. See if they can use the printout to get their code working.
5 min	Ask the pair with the best solution to share their ideas for programming "Talking to a Storybook". Have them draw on the board. If no students are able to explain it properly, explain the solutions yourself ("Talking to a Storybook" solutions). ● *Allow the user to ask Alexa about whether zorillas live in any country (not just the USA)* ○ Explain how *slot* blocks of type *Country* get filled when a user says a country's name ○ Explain how you can use the "get slot value" block to retrieve which country the user said ○ Show on solutions handout ■ Is the *if statement* here very effective? Can you think of another way to do it? (e.g., use ML to categorize the country into continents and instead check for "Africa") ● *Make Karabo say a unique sentence every time you ask Alexa to talk to Karabo* ○ Explain how the *sentence generator (LSTM)* block can be used to generate unique sentences ○ Show on solutions handout ● *Make a picture of Karabo with food pop up on the app's screen when you feed Karabo* ○ Explain how you can send information to App Inventor by using the *send to App Inventor* block ○ Explain how you can receive the information in App Inventor with the *CloudDB* block, and then change the picture with the *set.Picture* block ○ Show on solutions handout
30 min	Have students try making a simple Alexa Skill themselves using the "Making a

	Simple Alexa Skill" document. Pose challenges for students to think about how to code and say that they are free to work on them for the rest of the class and we will help: ● How they would implement questions for Alexa that would cause different things to happen depending on what page the user was on ○ E.g., if the user asks to feed Karabo on the first page, a picture of Karabo with a certain type of food would appear, but if the user asks to feed Karabo on the second page, a different picture would appear ● How they would implement a word game where you make up a story along with Alexa ○ Can you generate text? ○ Can you add pictures to go along with the text? Would you need pre-defined storylines for this? ● How they would create a conversational agent to talk to their grandma/grandpa ○ Can you add family photos to the app when grandma/grandpa asks specific questions? ○ Can you allow grandma/grandpa to text you by talking to Alexa?

Lesson 4: Conversational AI Independent Design Project Workday 1 (50 minutes with 10 minute buffer)

Pre-workshop preparation:

- Print out Individual Design Worksheets
- Print out Group Project Design Worksheets
- Print out (or link) to Communicating between App Inventor and Alexa Handout
- Update links doc with any links below

Time	Activity
5 min	Introduce the design project. ● Tell the students that this project is their opportunity to create a mobile app and Alexa Skill of their own design and imagination ● They will have two classes to make their projects with a partner and will do a short presentation during the final class ● There are two constraints to their projects: the project's theme will be the ENVIRONMENT and/or ASSISTED LIVING, so the app should have some relation to that theme and everyone's projects must include an app as well as an Alexa Skill ○ How can you utilize the benefits that come along with a conversational agent? ■ E.g., hands-free, can interact from anywhere in the room, etc. ○ How can you utilize the benefits that come along with a mobile phone? ■ E.g., accelerometer sensor, touch tools, microphone, speaker,

	visualizations, etc.
10 min	Have students complete the **Individual Design Worksheet**
15 min	Give students time to find a partner to do their projects with (or group of 3 if class has an odd number). Recommend that students find someone who shares a problem-solving interest. Once students finish pairing up, have each group do the **Group Project Design Worksheet** to decide what they are going to make. ● Check with every team to make sure they have a reasonable project plan ● Make sure the projects can be completed in a reasonable time
10 min	Tell students they should start by trying to implement the Alexa Skill connection to the app, perhaps by using the *CloudDB* and *send to App Inventor* blocks. An example of how to do this is shown in the Communicating between App Inventor and Alexa Handout. Once they have tested to make sure it works, then they can try to add the rest of the functionality.
10 min	Tell students that they are about to start coding. Before they do, either play the Pair Programming Video or write out the rules. Keep the rules up on the board. Rules: ● **Driver**: Sits at computer and uses keyboard ● **Navigator**: Helps driver by answering questions and pointing out potential problems/mistakes ● Be respectful ● Keep talking: tell each other what you're doing/thinking ● Switch roles often ● Don't be a bossy navigator ● Don't grab the driver's mouse/keyboard Optional: Every 8 minutes, ring a bell/make sure students are trading places with their partner. Meet with each team near the end of the class to make sure they're on track to have a high level of completeness by the final class. **Have students email jess.vanbrummelen@gmail.com their AIA files so there's a backup in case they somehow get deleted.**

Lesson 5: Conversational AI Independent Design Project Workday 2 (50 minutes with 10 minute buffer)

- Update links doc with any links below

Time	Activity
5 min	Briefly review conversational AI topics, including machine learning and rule-based AI, and Alexa Skill blocks
40 min	Students finish pair programming their projects. Remind students to switch roles every 8 minutes or so if needed.
5 min	Explain the final project presentations next week. Each team will need to make a **3-minute Google Slides presentation** about their project with a 1-minute question and answer session after. The presentation should follow this general template (email out the link) but students can feel free to change the background or add any graphics.

Lesson 6: Conversational AI Independent Design Project Final Presentations (50 minutes with 10 minute buffer)

Time	Activity
15 min	Allow students to make and practice their presentations, as well as make any final touches to their apps.
20 min	Student groups give their final project presentations to the whole class
15 min	For the students who are a part of the study, fill out the **post-questionnaire**. Other students may work on MIT App Inventor.

Appendix D

Questionnaires

This appendix contains the questionnaires provided in the first and last workshops.

D.1 Pre-questionnaire

The pre-questionnaire was provided to students via a Google Form [32]. A printer-friendly version of the form is included here.

Programming Amazon Alexa with MIT App Inventor - Survey

The purpose of this research study is to determine whether students can understand conversational artificial intelligence (the ability for a computer to have conversations with humans) through workshops and developing programming projects.

Participation in this study is completely voluntary. You may decline to answer any or all of the questions. If you decline to answer any of the questions, you will no longer be participating in the study. To decline participation, you may close this survey webpage, and let the instructor know that you no longer want to participate. You may decline participation at any time, and if you choose to do so, there will be no adverse consequences. You will still be allowed to participate in the workshops.

Through proper research measures and safeguarding of data, your confidentiality and anonymity will be assured. The data collected in this study will be reported in such a way that the identity of individuals is protected.

* Required

1. *Mark only one oval.*

   ◯ I agree to participate, and understand that if I no longer want to participate, I can decline at any time by closing the webpage and letting the instructor know.

   ◯ I do not want to participate. *Stop filling out this form.*

Programming Amazon Alexa with MIT App Inventor - Survey

Please answer the following questions based on how strongly you disagree or agree with the statement.

2. **I have interacted with conversational agents.** *
   Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

3. **I understand how conversational agents decide what to say.** *
   Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

4. **I feel comfortable making apps that interact with conversational agents.** *
   Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

5. **I can think of ways that conversational agents can solve problems in my everyday life.** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

Programming Experience

6. **Before hearing about this workshop, did you have any experience programming? (E.g., Scratch, MIT App Inventor, Python, etc.)** *
Mark only one oval.

- ◯ Yes
- ◯ No *Skip to question 8.*

Skip to question 8.

Programming Experience
In the last section, you mentioned you had previous programming experience.

7. **Describe any previous programming experience you have had (e.g., Scratch, MIT App Inventor, Python, etc.).** *

Short Answer 1/5
Please answer the following question in your own words.

8. **Describe how you think conversational agents decide what to say.** *

Short Answer 2/5
Please answer the following question in your own words.

9. **What do you think are conversational agents' capabilities/limitations? What do you think they can/can't they do?** *

Short Answer 3/5

Please answer the following question in your own words.

10. **Do you think conversational agents are "intelligent"? To what extent are/aren't they?** *

Short Answer 4/5

Please answer the following question in your own words.

11. **Do you think conversational agents are safe to use? To what extent do you think they are/aren't?** *

Short Answer 5/5

Please answer the following question in your own words.

12. **How might you use conversational agents in the future to create something that solves a problem you see in the world?** *

D.2 Post-questionnaire

The post-questionnaire was provided to students via a Google Form [32]. A printer-friendly version of the form is included here.

Programming Amazon Alexa with MIT App Inventor - Survey

The purpose of this research study is to determine whether students can understand conversational artificial intelligence (the ability for a computer to have conversations with humans) through workshops and developing programming projects.

Participation in this study is completely voluntary. You may decline to answer any or all of the questions. If you decline to answer any of the questions, you will no longer be participating in the study. To decline participation, you may close this survey webpage, and let the instructor know that you no longer want to participate. You may decline participation at any time, and if you choose to do so, there will be no adverse consequences. You will still be allowed to participate in the workshops.

Through proper research measures and safeguarding of data, your confidentiality and anonymity will be assured. The data collected in this study will be reported in such a way that the identity of individuals is protected.

* Required

1. *Mark only one oval.*

   ◯ I agree to participate, and understand that if I no longer want to participate, I can decline at any time by closing the webpage and letting the instructor know.

   ◯ I do not want to participate. *Stop filling out this form.*

Programming Amazon Alexa with MIT App Inventor - Survey

Please answer the following questions based on how strongly you disagree or agree with the statement.

2. **I have interacted with conversational agents.** *
   Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

3. **I understand how conversational agents decide what to say.** *
   Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

4. **I feel comfortable making apps that interact with conversational agents.** *
   Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

5. **I can think of ways that conversational agents can solve problems in my everyday life.** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

6. **My understanding of conversational agents improved through these workshops.** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

Short Answer 1/9

Please answer the following question in your own words.

7. **How do you feel about the project you made?** *

Short Answer 2/9

Please answer the following question in your own words.

8. **What was hard about developing the project?** *

Short Answer 3/9

Please answer the following question in your own words.

9. **What were some things you didn't expect when you were developing the project or learning in class?** *

Short Answer 4/9

Please answer the following question in your own words.

10. **How would you describe the difference between machine learning and rule-based AI?** *

Short Answer 5/9

Please answer the following question in your own words.

11. **Describe how you think conversational agents decide what to say.** *

Short Answer 6/9

Please answer the following question in your own words.

12. **What do you think are conversational agents' capabilities/limitations? What do you think they can/can't they do?** *

Short Answer 7/9

Please answer the following question in your own words.

13. **Do you think conversational agents are "intelligent"? To what extent are/aren't they?** *

Short Answer 8/9

Please answer the following question in your own words.

14. **Do you think conversational agents are safe to use? To what extent do you think they are/aren't?** *

Short Answer 9/9

Please answer the following question in your own words.

15. **If you were to use conversational agents in the future to create something that solves a problem you see in the world, what would it be?** *

Appendix E

Workshop Handouts

The following documents are worksheets provided to students during the workshops.

E.1 Conversational AI Rules Worksheet

The following worksheet asked students to write down rules for a smart home agent and role play as the agent with a partner. Students learned that making rules for every situation can be time consuming and difficult. This provided an opportunity to teach about machine learning methods and how they can improve efficiency in certain situations.

Jessica Van Brummelen, jess@csail.mit.edu

Conversational Agent Rules

Voice User Interface (VUI):
The VUI is the part of the program that listens for specific keywords and questions from the user (e.g., the VUI listens for the words “turn on the lights”).

Endpoint:
The endpoint is the part of the program that does something based on what was said to the VUI (e.g., the endpoint turns on the lights).

VUI If Statement	**Endpoint Reaction**
e.g., *If user says, “Turn on the lights in the living room”*	e.g., *Then respond with, “Sure, I’ll turn on the living room lights”, and send a signal to the living room lights to turn on*

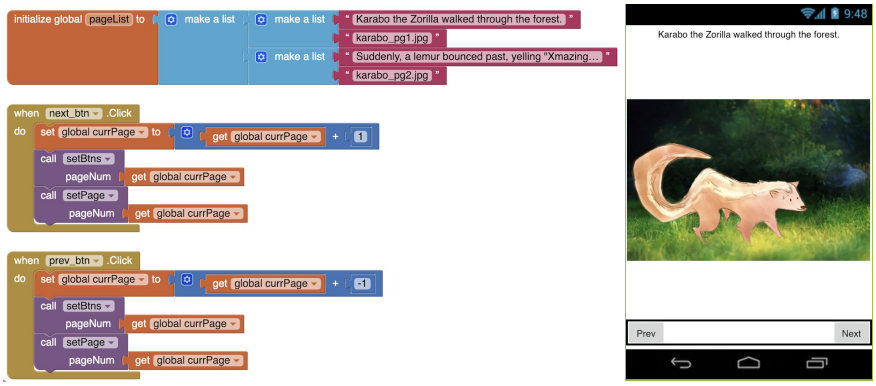
VUI statement	Endpoint reaction

E.2 Storybook Worksheet

The following worksheet asked students about the conversational AI storybook application.

Jessica Van Brummelen, jess@csail.mit.edu

"Talking to a Storybook" App and Alexa Skill Design

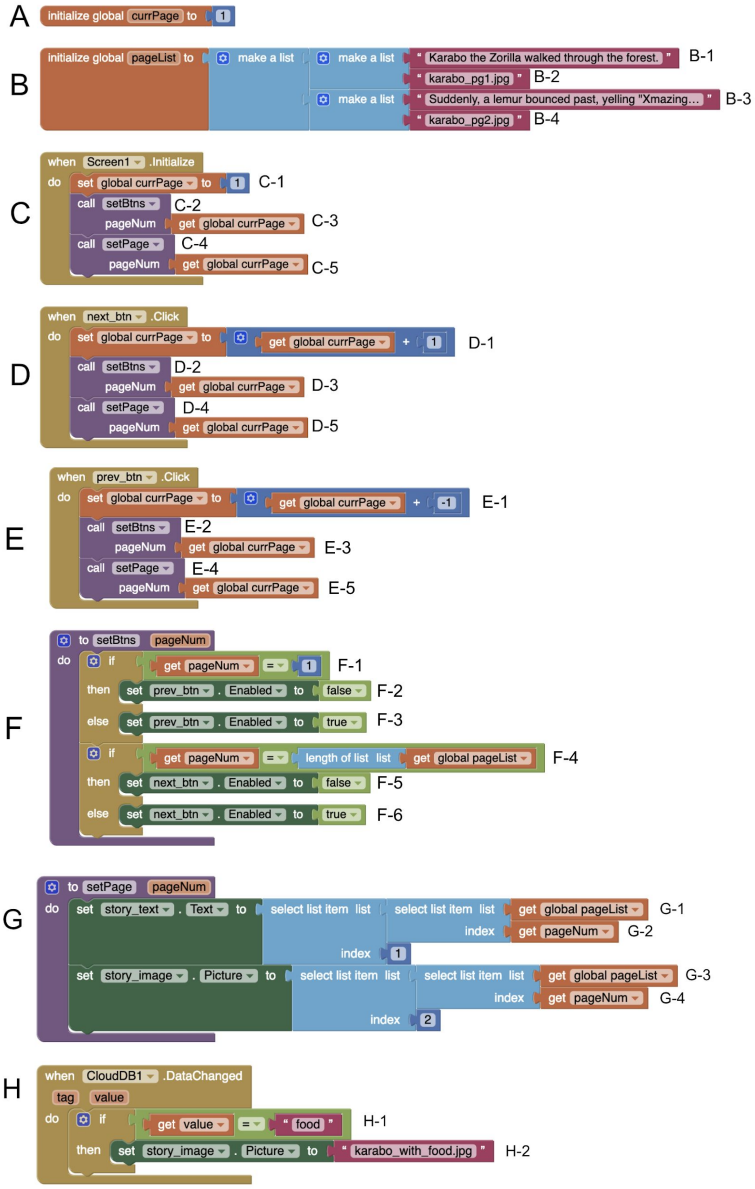


See the next page for the full list of blocks for this app

1. We want to make a storybook app that lets you talk to Alexa about a story shown on the app's screen. Open up the AIA project in MIT App Inventor. You can flip between storybook pages in the app. Play with it and look at the blocks.
   a. Which blocks save the page number? See the blocks on the next page, and write down the corresponding letters. (**Hint:** There's one type of block that initializes the page number, and another block that updates it)

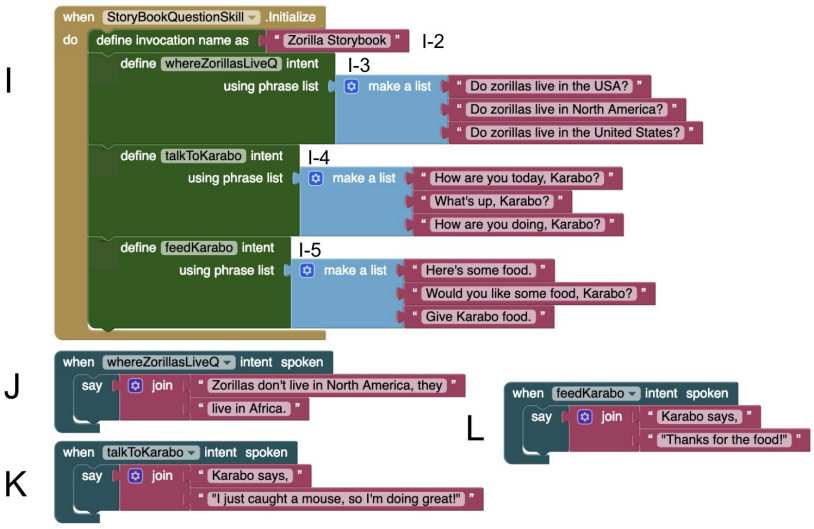
   b. Which blocks make the next/previous buttons work? (**Hint:** There's an *event listener* block that listens for a "click", and purple procedure blocks that are *called* when the button is clicked)

   c. Which blocks change the picture on the screen? (**Hint:** one of the lists contains *.jpg* filenames, which are selected in another block that changes the picture)


A
initialize global currPage to 1
B
initialize global pageList to
make a list
make a list
"Karabo the Zorilla walked through the forest."
B-1
"karabo_pg1.jpg"
B-2
make a list
"Suddenly, a lemur bounced past, yelling "Xmazing…""
B-3
"karabo_pg2.jpg"
B-4
C
when Screen1 .Initialize
do
set global currPage to 1
C-1
call setBtns
C-2
pageNum get global currPage
C-3
call setPage
C-4
pageNum get global currPage
C-5
D
when next_btn .Click
do
set global currPage to get global currPage + 1
D-1
call setBtns
D-2
pageNum get global currPage
D-3
call setPage
D-4
pageNum get global currPage
D-5
E
when prev_btn .Click
do
set global currPage to get global currPage + -1
E-1
call setBtns
E-2
pageNum get global currPage
E-3
call setPage
E-4
pageNum get global currPage
E-5
F
to setBtns pageNum
do
if get pageNum = 1
F-1
then set prev_btn . Enabled to false
F-2
else set prev_btn . Enabled to true
F-3
if get pageNum = length of list list get global pageList
F-4
then set next_btn . Enabled to false
F-5
else set next_btn . Enabled to true
F-6
G
to setPage pageNum
do
set story_text . Text to
select list item list get global pageList
index get pageNum
G-1
G-2
index 1
set story_image . Picture to
select list item list get global pageList
index get pageNum
G-3
G-4
index 2
H
when CloudDB1 .DataChanged
tag value
do
if get value = "food"
H-1
then set story_image . Picture to "karabo_with_food.jpg"
H-2

2. With the Alexa Skill, you can ask Alexa about the main character, a zorilla named Karabo.
   a. Which blocks define what you can ask Alexa about? (**Hint**: If you don't see the green Skill blocks in the workspace, try clicking on the "Screen1" dropdown menu. Do you see a Skill name there?)

   b. Which blocks define how Alexa responds?



3. Make a plan for how you would change the Alexa Skill to do the following things:
   - Make Alexa understand more sentences. For instance, allow someone to say, “Karabo, how are you?” to trigger the “talkToKarabo” intent.
   - Allow the user to ask Alexa about whether zorillas live in any country (not just the USA)
     - **Hint**: look at the *slot* blocks on the handout
   - Make Karabo say a unique sentence every time you ask Alexa to talk to Karabo
     - **Hint**: look at the *generate text* block on the handout
   - Make a picture of Karabo with food pop up on the app’s screen when you feed Karabo
     - **Hint**: look at the *send to App Inventor* block on the handout

Write or draw out your ideas in the space below.

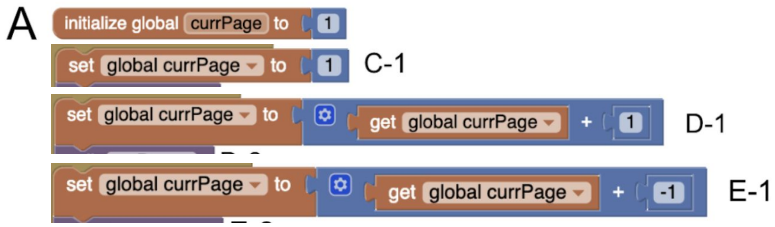
E.3 Storybook Solutions

This document outlines the solutions to the conversational AI storybook application worksheet shown in Appendix E.2.

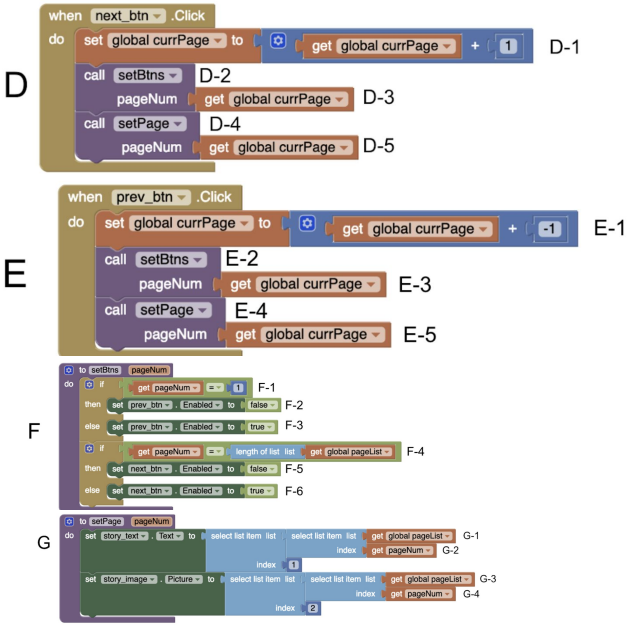
Jessica Van Brummelen, jess@csail.mit.edu

“Talking to a Storybook” Q1 Solution

a. The following blocks initialize and save the page number:



b. The following blocks make the next/previous buttons work by setting the page number and setting the picture and storybook text that go along with the current page number:

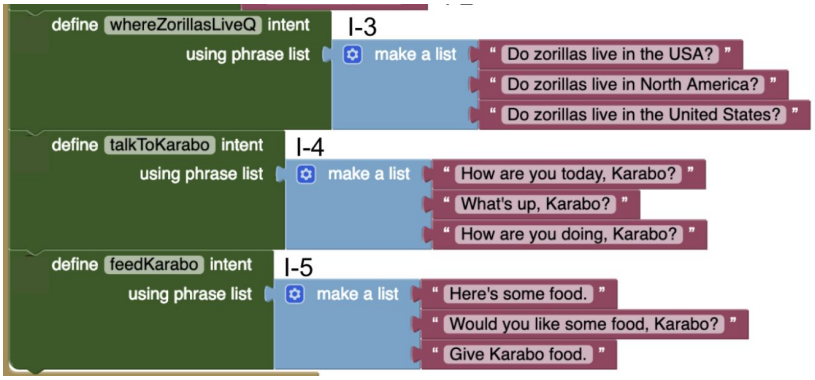


c. The following blocks change the picture on the screen. Note that G-3 and G-4 are used to get the JPG file name that goes along with the current page number (from the pageList list):

Jessica Van Brummelen, jess@csail.mit.edu

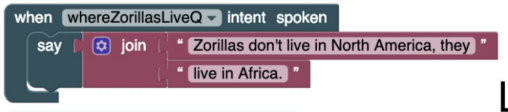
“Talking to a Storybook” Q2 Solution

a. These blocks define what you can ask Alexa about:

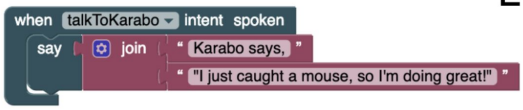


b. These blocks define how Alexa responds:

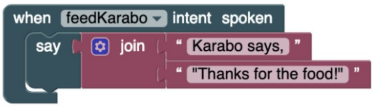
J



K



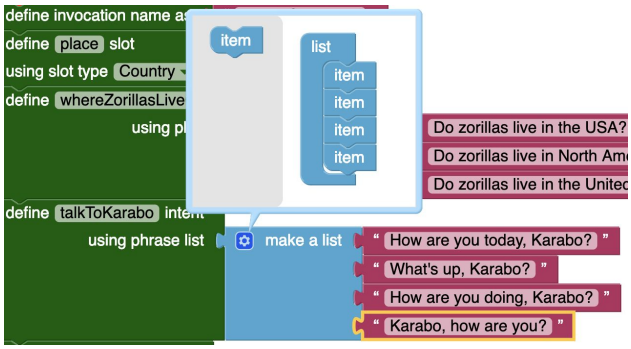
L



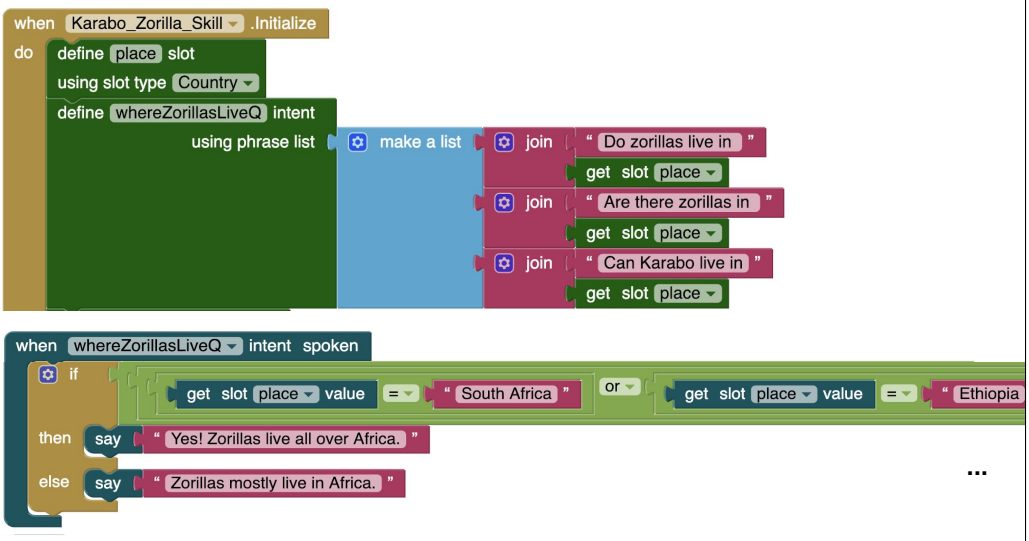
Jessica Van Brummelen, jess@csail.mit.edu

“Talking to a Storybook” Q3 Solution

3. To make Alexa understand the phrase, “Karabo, how are you?”, in the Alexa Interface, use the dark blue icon on the “make a list” block to add an item to the list. Then add the phrase to the list:

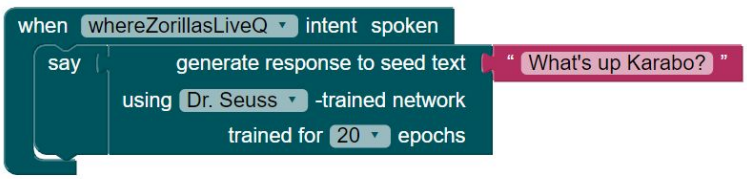


To enable other places to be spoken to Alexa, in the Alexa Interface, use “slot” blocks:



...

To generate unique sentences, in the Alexa Interface, use the “generate response to seed text” block:

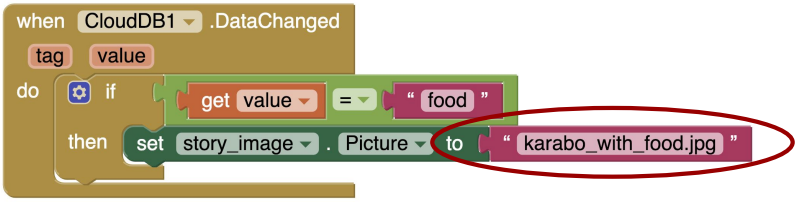


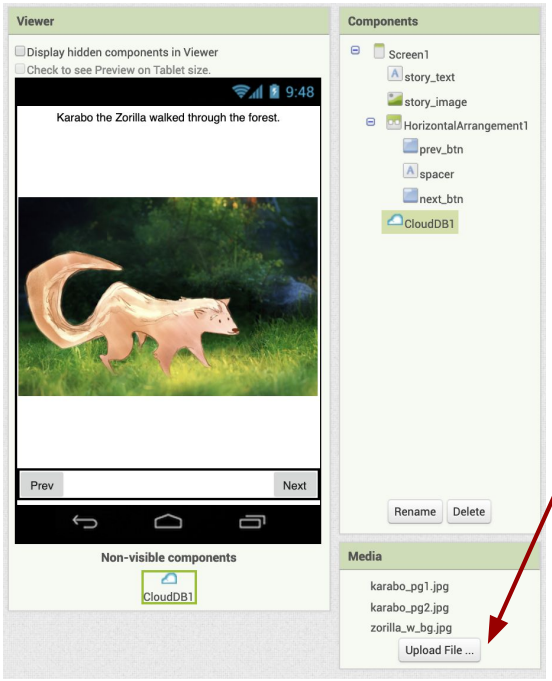
Jessica Van Brummelen, jess@csail.mit.edu

"Talking to a Storybook" Q3 Solution Cont'd

In the Alexa Interface, add a "send" block that sends the word "food" to App Inventor:

In the App Inventor interface, notice the CloudDB component, and the block that says "set story_image" to "karabo_with_food.jpg". To enable the "set story_image" block to work, you have to create a picture with Karabo and food, name it, "karabo_with_food.jpg" (as in the text block), and upload it to App Inventor:





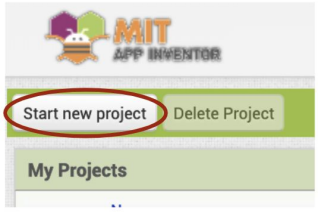
karabo_with_food.jpg

E.4 Alexa Skill Development Tutorial Handout

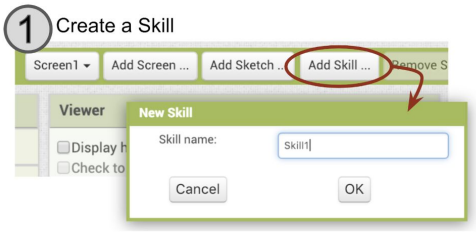
We provided the following handout, which contains detailed information about each conversational AI block, during the third workshop.

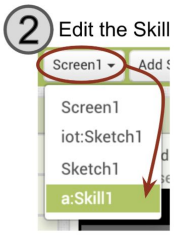
Making a Simple Alexa Skill in MIT App Inventor

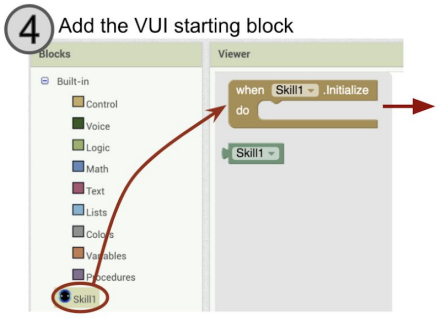
1. Create an Amazon Developer Account (Note that this is different than a regular Amazon account and different than an AWS account):
    a. https://developer.amazon.com/edw/home.html
2. Go to App Inventor test server:
    a. http://alexa.appinventor.mit.edu
3. Log in and create a new App Inventor project:

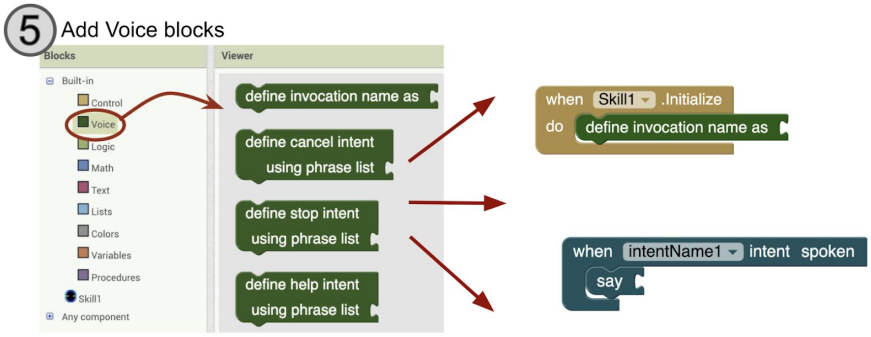


    a. Feel free to name your project anything you'd like!
4. Create a new Alexa Skill:

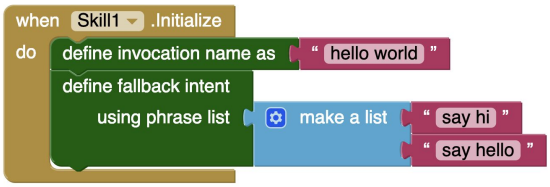






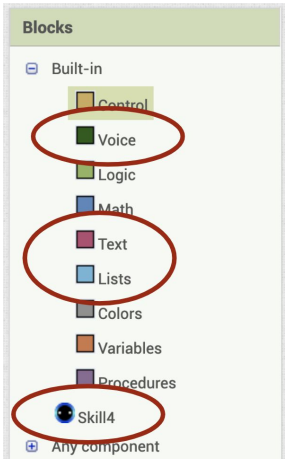


a. Create the Voice User Interface (the phrases that Alexa will understand) by arranging the blocks similarly to what's shown here:



i.

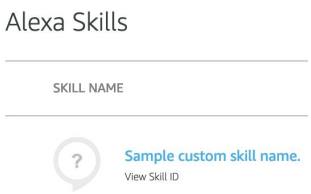
1. The "when <skill_name>.initialize" block can be found in the **Skill drawer** (see image below)
2. The define invocation name and define intent blocks can be found in the **Voice drawer**
3. The "make a list" block can be found in the **Lists drawer**
4. The "__" block can be found in the **Text drawer**

a. 

ii. The **invocation name** is the words you say to Alexa to open your skill
    1. E.g., “Alexa, open hello world” would open this skill
iii. The **list of phrases** contains sentences that you can say to **invoke an “intent”** or a function that you will define later
    1. E.g., “Alexa, open hello world and say hello” would cause the sayHello intent to be invoked (and Alexa to take some sort of action)

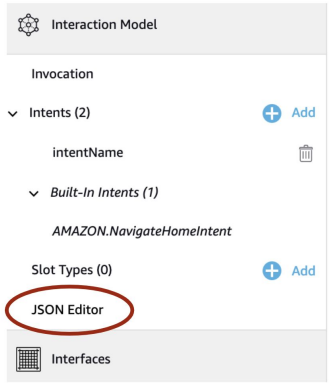
5. Go back to **designer page** (with the Echo Dot picture) and **send updates to Alexa**:

a. This sends information to the Amazon Developer Console about the **Voice User Interface (VUI)** you created
    i. Check to make sure it has updated on the [Alexa Developer](https://developer.amazon.com) (https://developer.amazon.com) website
        1. Has your Skill appeared on this website?



a.

2. Click on your skill and in the left panel, click JSON Editor



a.

3. Do you see your invocation name and sample phrases in the JSON?

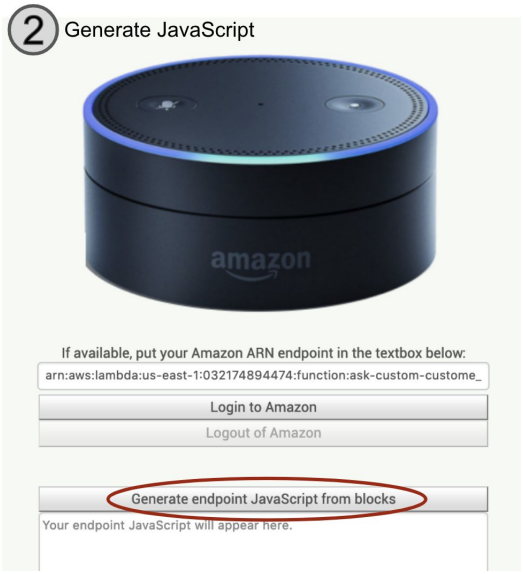
b. Hopefully it updated! You're not quite done yet, though. Now, we want to cause Alexa to *do something* in response to what you say.

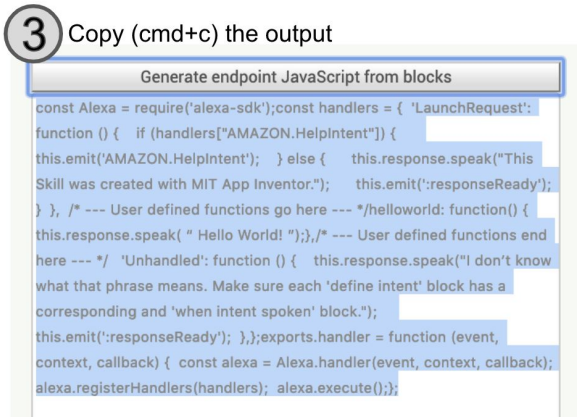
6. Create the **Endpoint Function** (the reaction that Alexa has to the phrases you say) by arranging the blocks similarly to what's shown here:

a.

   i. Find the "when <intent_name> intent spoken" and the "say" blocks in the "Voice" drawer, and the "<text>" block in the "Text" drawer
   ii. Note that you should select the name of your intent (e.g., Fallback) from the drop-down menu, or type a custom name in yourself

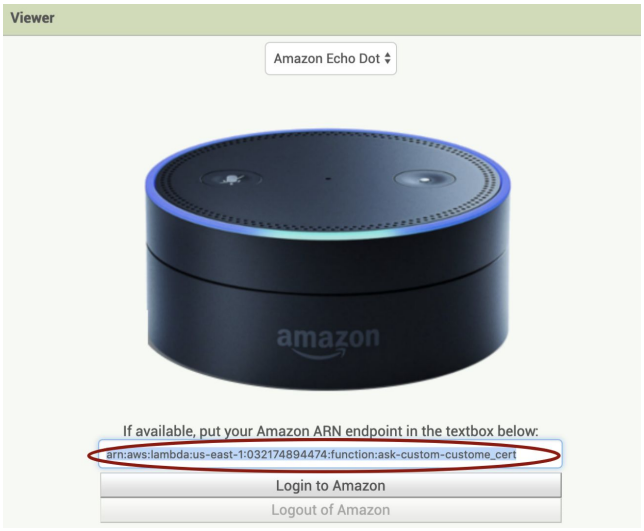
7. Go back to the designer page and click "**Generate endpoint JavaScript from blocks**" and copy the output
   a. This generates JavaScript based on the "when intent spoken" functions that you defined



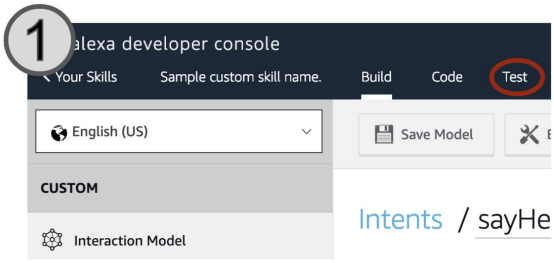


8. Send an email to the teachers at C12799s1-teachers@esp.mit.edu that includes:
    a. Your name (first and last)
    b. The name of your Skill
    c. The JavaScript output that you copied in the step above
9. We will upload this JavaScript to Amazon Web Services (AWS) and send you a reply email with the "endpoint ARN", that will look something like this:
    arn:aws:lambda:us-east-1:032174894474:function:ask-custom-custome_cert

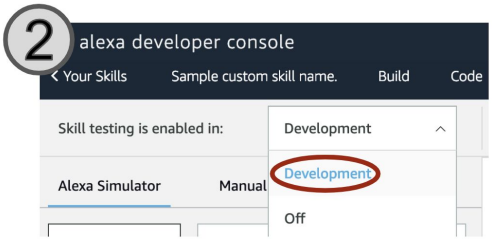
10. Connect the Voice User Interface and Endpoint function:
    a. You can connect your Voice User Interface to the endpoint by pasting the endpoint function's ARN into App Inventor, and updating your skill:



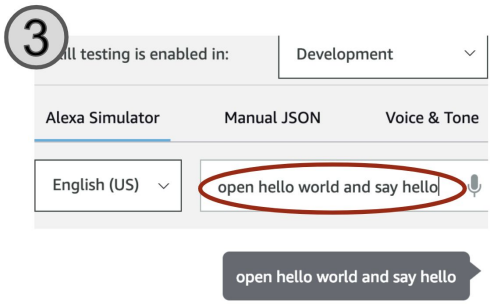
    b. Make sure you send updates to Amazon by clicking the button below the ARN
11. Congrats! You made your first MIT App Inventor Alexa Skill :-)
12. Now, you can try out the skill using the Alexa Developer Console
    a. Go back to the Alexa Developer website and click on test:



    b. Select Development testing:



c. Try out your skill by typing, “open <your_invocation_name> and <your_intent_phrase>”, or just, “tell <your_invocation_name> to <your_intent_phrase>”, and see how Alexa responds.



E.5 Project Brainstorm: Individual Worksheet

The following handout asked students to brainstorm conversational agent project ideas individually. This document was based on the worksheet provided in [50].

Jessica Van Brummelen, jess@csail.mit.edu

Conversational AI Individual Design Project Worksheet

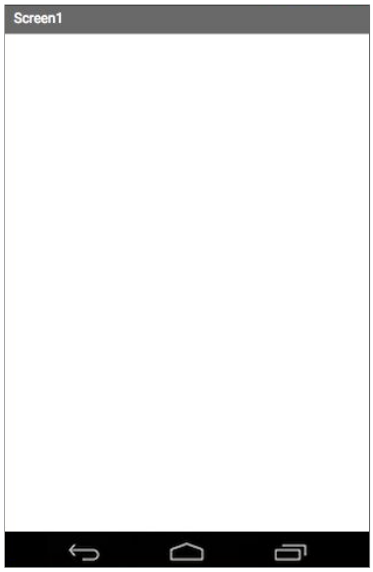
You will design and make an app and Alexa Skill about the ENVIRONMENT and/or ASSISTED LIVING. This worksheet will help you think about what kind of app you would like to make.

1. Come up with at least 3 problems having to do with the ENVIRONMENT and/or ASSISTED LIVING that you are *passionate* about.

Problem statement	**Why I care about this problem**

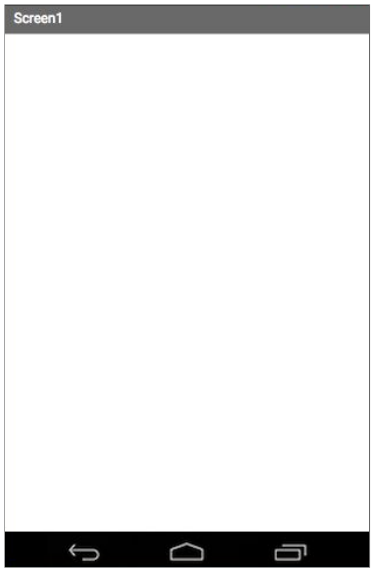
2. Share your problem statements with the person next to you. Discuss possible ideas for simple apps that would use Alexa to help solve these issues. Take notes below:

3. On the following page, draw ideas for possible apps/skills you could make that would use a conversational agent somehow to solve problems. How would someone use your app/skill?



Alexa Skill Ideas:

Jessica Van Brummelen, jess@csail.mit.edu



Alexa Skill Ideas:

E.6 Project Brainstorm: Group Worksheet

The following handout asked students to brainstorm conversational agent project ideas with a group. This document was based on the worksheet provided in [50].

Jessica Van Brummelen, jess@csail.mit.edu

Conversational AI Group Design Project Worksheet

Now that you've formed a group, your group needs to decide what you want to make! All group members can use this worksheet to write down ideas, but only one needs to be completed.

1. Take turns discussing the ideas that you came up with individually. When it's your partner's turn, write down specific things s/he talked about that interested you the most:

2. Decide which problem your group wants to tackle for this project. Talk about ideas for what app you want to make to solve the problem. Keep in mind that you have limited class time to code, so try to make sure that you can finish your app on time (with help from teachers)!

What is the problem you are trying to solve?_____

_____

How will your app/skill solve it/What will your app/skill help people do?_____

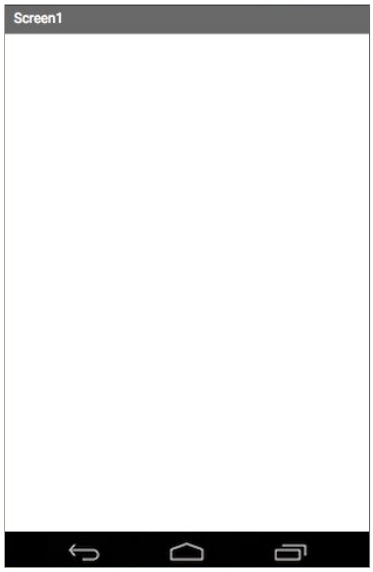
_____

How does your app/skill use conversational AI?_____

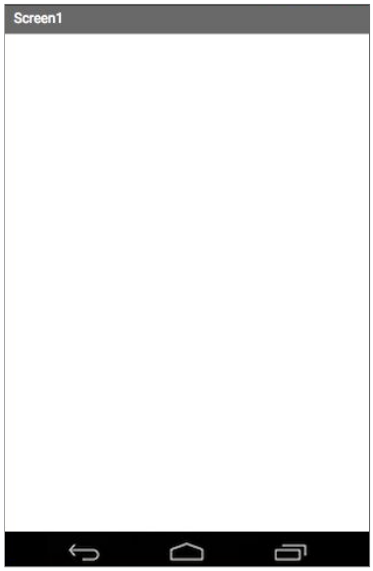
_____

3. On the back of this page, draw some ideas for what your final app/skill will look like. Try to mark what buttons users would press to do something and how the app/skill would respond to these user actions.

Jessica Van Brummelen, jess@csail.mit.edu

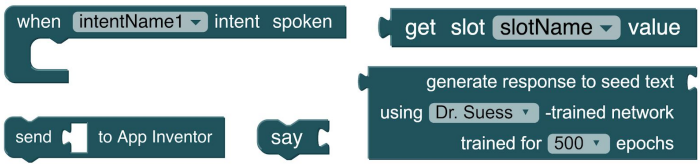


Alexa Skill Ideas:



Alexa Skill Ideas:

4. Now, let's focus on how you will use and code the Alexa Skill portion of your app. Think about what the conversational AI blocks you've worked with so far can do and how you can use them for your project.



List the main intents that Alexa will respond to:

1. An example phrase for the intent:_____

   Any *slots* (with *slot type*) the phrase will need:_____

   The spoken response will be (circle): *Rule-based* *AND/OR* *Generated*

   The pre-defined response and/or seed text:_____

   What happens in the app with this response:_____

   What value is sent to App Inventor:_____

2. An example phrase for the intent:_____

   Any *slots* (with *slot type*) the phrase will need:_____

   The spoken response will be (circle): *Rule-based* *AND/OR* *Generated*

   The pre-defined response and/or seed text:_____

   What happens in the app with this response:_____

   What value is sent to App Inventor:_____

3. An example phrase for the intent:_____

   Any *slots* (with *slot type*) the phrase will need:_____

   The spoken response will be (circle): *Rule-based* *AND/OR* *Generated*

   The pre-defined response and/or seed text:_____

   What happens in the app with this response:_____

   What value is sent to App Inventor:_____

E.7 Running List of Website Links and Lesson Summaries

We provided the students with a Google Document containing a running list of websites and workshop lesson summaries for reference throughout the semester. This document is shown below. Note that the tutorials under the *Lesson 1 Links* section were originally implemented in [50].

Programming Amazon Alexa with MIT App Inventor

Workshop Links and Summaries

Table of Contents

Table of Contents **1**

Links **2**

- Lesson 1 Links 2
- Lesson 2 Links 2
- Lesson 3 Links 2
- Lesson 4 Links 2
- Lesson 5 Links 2
- Lesson 6 Links 3

Lesson Summaries **4**

- Lesson 1: Introduction to MIT App Inventor 4
- Lesson 2: Introduction to MIT App Inventor 4
- Lesson 3: Making Alexa Skills 5
- Lesson 4: Project Brainstorming 6
- Lesson 5: Review 8
  - Conversational AI 8
  - Skill-App Communication 8
  - Slots 8

Links

Lesson 1 Links

- MIT App Inventor

MIT App Inventor tutorials

- Talk to Me: Part 1 and Talk to Me: Part 2
- My Favorite Things (provide joke pictures and recordings)
- Paint Pot
- QuizMe
- Minigolf

Lesson 2 Links

- Alexa App Inventor Version (Still in Beta! **Please don't share this link with anyone**)
- Alexa Developer Console
- Storybook Example - AIA File

Lesson 3 Links

- Making a Simple Alexa Skill
- Once you've gotten to the part of the tutorial where you generate JavaScript, send an email to the teachers at C12799s1-teachers@esp.mit.edu that includes:
    - Your name (first and last)
    - The name of your Skill
    - The JavaScript output

  so that we can upload your Skill to Amazon.

Lesson 4 Links

- Communicating between App Inventor and Alexa Tutorial

Lesson 5 Links

- alexa.appinventor.mit.edu

- Making a Simple Alexa Skill Tutorial
- Communicating between App Inventor and Alexa Tutorial
- Using Slots

- Presentation slide template (feel free to change the background or add graphics!)

Lesson 6 Links

- Upload your AIA files here
- How to create an Alexa Skill in MIT App Inventor (full tutorial)

Lesson Summaries

Lesson 1: Introduction to MIT App Inventor

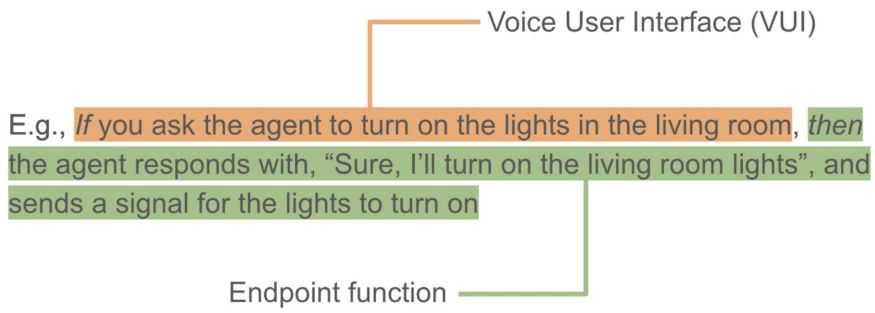
In the first week, we went through MIT App Inventor tutorials and learned how to create Android Apps! See the “Lesson 1 Links” above.

Lesson 2: Introduction to MIT App Inventor

Just to recap everything we went over for those of you who didn't make it out, in Lesson 2, we introduced conversational agents (e.g., Amazon Alexa) and went through an example of an App Inventor Alexa Skill.

The main concepts that we taught were:
- A conversational agent is anything that isn't human, and can interact or have a conversation with a human (e.g., a chatbot, Amazon Alexa, Google Home, Siri, etc.)
- When programming conversational agents, you have to think about two main things:
  - The Voice User Interface (VUI), which defines the things you can say to the agent (i.e., the things/commands that the agent understands)
  - The endpoint function, which is basically the "brains" behind an agent (e.g., what the agent does when it hears a command)

Here's an example that shows what part of a particular sentence corresponds to the VUI and to the endpoint:



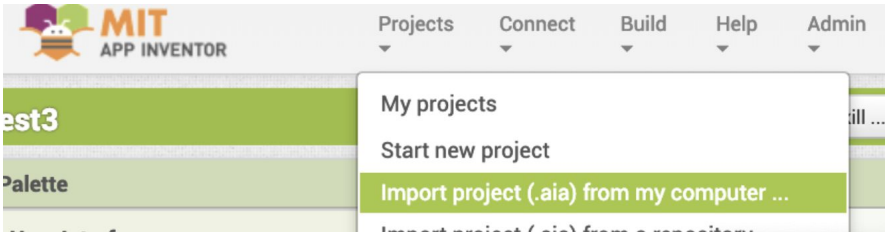
- To program an Alexa Skill, there needs to be three main things:
  - An invocation name, which directs Alexa to the correct "skill" (e.g., if there was an email Alexa Skill, the invocation name might be "Gmail" or "Yahoo!")

- An intent, which tells Alexa what the user's "intent" or purpose is for talking to it (e.g., a "send an email" intent)
- An endpoint function, which causes Alexa to do something when it hears the intent (e.g., causes Alexa to actually send an email)

We went through the App Inventor blocks that allow you to define the invocation name, intents, and endpoints of an Alexa Skill during class. Here's a link to a handout with more information about the Alexa Skills blocks (see pages 1-3).

Don't worry too much about knowing exactly what each block does yet. You'll get some more hands-on experience in the next workshop and learn more as we go. For now, try to complete this worksheet (and if you don't know the answers, no worries! We'll go through the solutions next class, and as always, feel free to email us about it!)

To complete the worksheet, you can either look at the blocks on the 2nd and 3rd page of the worksheet and pick out the correct blocks, or you can play with the actual blocks by downloading the AIA file here and importing it to the Alexa App Inventor website (note that this website is still in beta, and you will probably run into bugs. If you run into bugs, feel free to email us!)



Looking forward to next class! You'll get to start creating your own Alexa Skills!

Lesson 3: Making Alexa Skills

In Lesson 3, we went over the solutions to the worksheet from Lesson 2. The solutions can be found here.

We also started creating our own Alexa Skills using this tutorial:

- Making a Simple Alexa Skill
- Once you've gotten to the part of the tutorial where you generate JavaScript, send an email to the teachers at C12799s1-teachers@esp.mit.edu that includes:
    - Your name (first and last)

- The name of your Skill
- The JavaScript output

so that we can upload your Skill to Amazon.

Lesson 4: Project Brainstorming

In this lesson, you'll have time to brainstorm project ideas and create a group. The project theme is *ambient assisted living* and/or *the environment*. You will create an app and Alexa Skill that will address a problem in this area.

For example,

How could we help with...

- Memory loss
- Blindness
- Movement difficulty
- Mental health
- Muteness
- Learning impairment
- Disability awareness

How could we help with...

- Recycling
- Composting
- Climate change
- Energy management
- Pollution
- Environmental awareness

When thinking about your project, you should consider the capabilities of the conversational AI and mobile app tools that you are using. For example,

How can we use conversational agent capabilities?

- Conversation
- Sound (loudness)
  - Multiple people can hear and interact
- Hands-free capabilities
  - Can converse while doing other things
- Connection to mobile device
- Connection to internet
- Sentence generation

How can we use mobile device capabilities?

- Visuals (pictures, text)
- Touch screen
- Haptic feedback (vibration)
- Language translation
- Location information (GPS)
- Phone
- Camera

Here are the overall project guidelines:

Project Guidelines

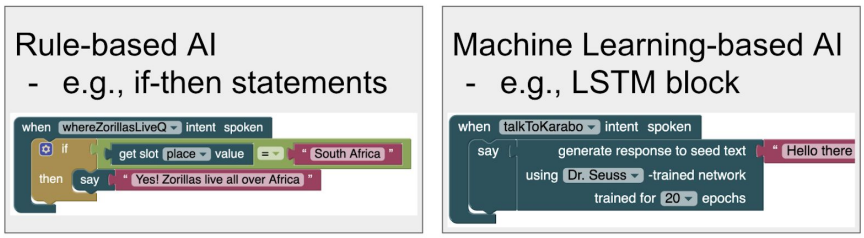
- Includes an **Alexa Skill**
- Includes an **app**
- Has to do with **assisted living** and/or the **environment**
- Work in **groups of two**
- Work on it during **this and next class**
- Show the class your project on **April 6th**

Finally, you can go through this tutorial to learn how to make an app that connects with your Alexa skill.

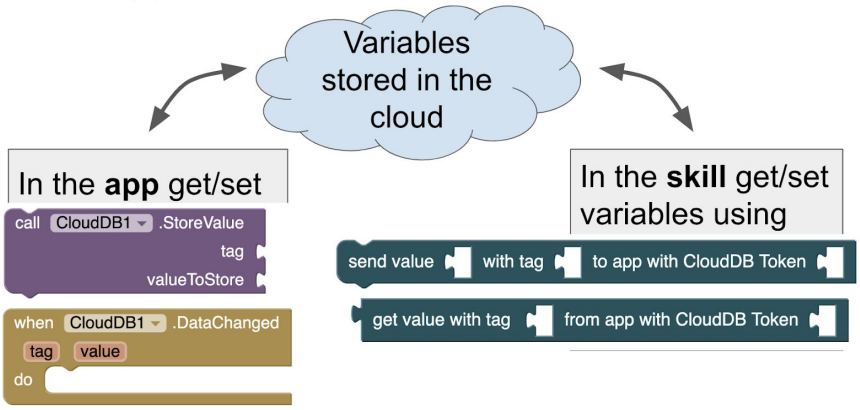
Lesson 5: Review

Conversational AI

- Two main ways to program:



Skill-App Communication



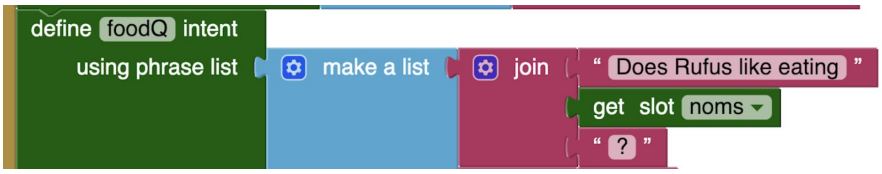
Slots

If you're interested in using *slots* in your app (e.g., to get specific information from a user, such as a date or a type of food), you can take a look at the third page of the Alexa Blocks Handout, or the explanation below:

1. Add the "define slot" block to your VUI initialization:

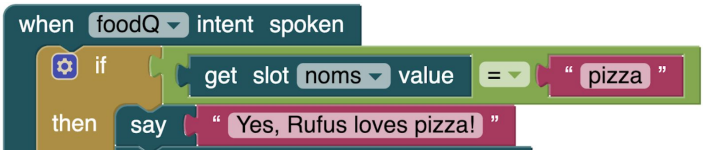
This defines a "food-slot" called "noms" that can be used as shown below:

2. Use the "get slot" block in an intent phrase:



This allows Alexa to understand phrases such as, "Does Rufus like eating *pizza*?", "Does Rufus like eating *salad*?", etc.

3. Use the "get slot value" block in your endpoint function:



This causes Alexa to check the phrase that the user said (e.g., Does Rufus like eating pizza/salad/etc.), grab the value of the slot (e.g., pizza/salad/etc.), compare the value to the word "pizza", and then say something in response (e.g., "Yes, Rufus loves pizza!").

Appendix F

Other Documentation

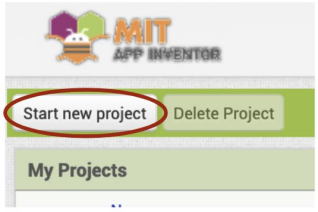
This appendix contains documentation provided to MIT faculty, staff, and students who attended a testing session for the conversational AI interface.

F.1 Documentation for Testing Sessions

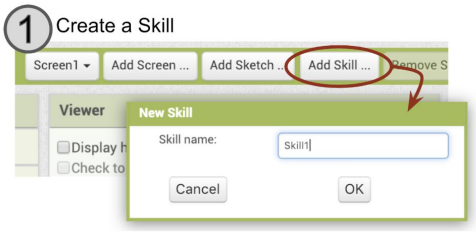
This section contains a version of the document given to those who attended the initial testing sessions of the Alexa Skills interface. The testers ranged from MIT App Inventor developers to MIT staff with little to no coding experience.

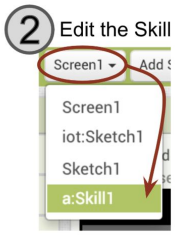
Standard Process to make an Alexa Skill in App Inventor

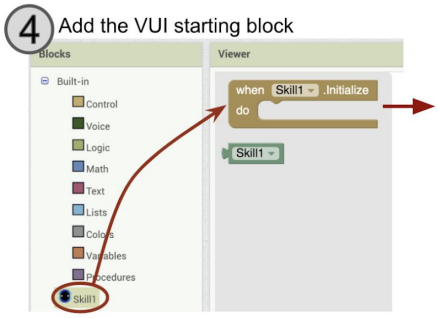
1. Create an Amazon Developer Account (Note that this is different than a regular Amazon account and different than an AWS account):
    a. https://developer.amazon.com/edw/home.html
2. Go to App Inventor test server:
    a. http://alexa.appinventor.mit.edu
3. Log in and create a new App Inventor project:

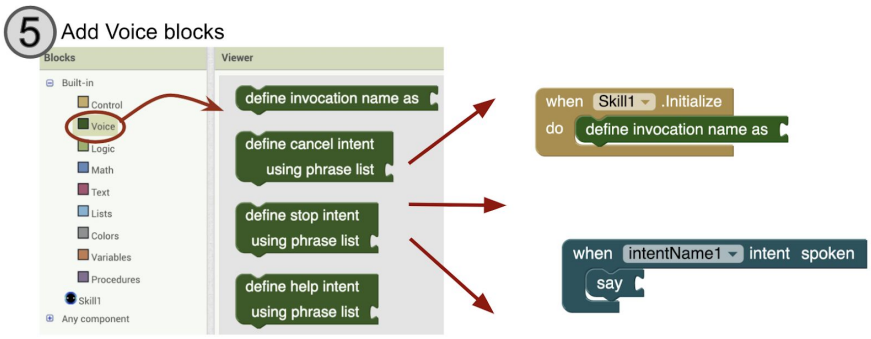


    a. Feel free to name your project anything you'd like!
4. Create a new Alexa Skill:

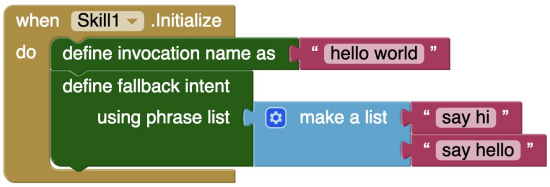






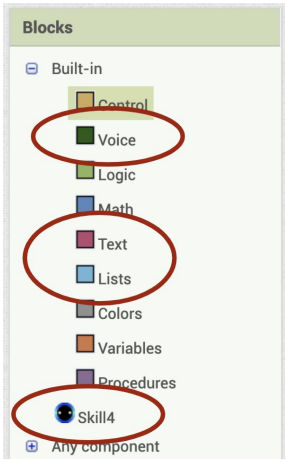


a. Create the Voice User Interface (the phrases that Alexa will understand) by arranging the blocks similarly to what's shown here:



i.

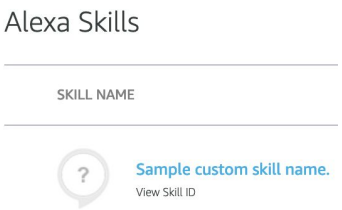
1. The "when <skill_name>.initialize" block can be found in the **Skill drawer** (see image below)
2. The define invocation name and define intent blocks can be found in the **Voice drawer**
3. The "make a list" block can be found in the **Lists drawer**
4. The "__" block can be found in the **Text drawer**

a. 

ii. The **invocation name** is the words you say to Alexa to open your skill
    1. E.g., “Alexa, open hello world” would open this skill
iii. The **list of phrases** contains sentences that you can say to **invoke an “intent”** or a function that you will define later
    1. E.g., “Alexa, open hello world and say hello” would cause the sayHello intent to be invoked (and Alexa to take some sort of action)

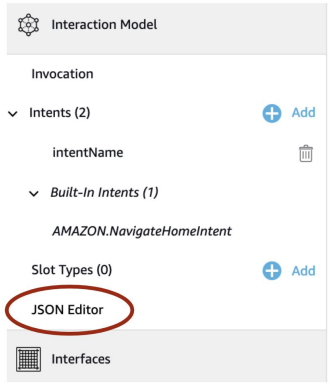
5. Go back to **designer page** (with the Echo Dot picture) and **send updates to Alexa**:

a. This sends information to the Amazon Developer Console about the **Voice User Interface (VUI)** you created
    i. Check to make sure it has updated on the [Alexa Developer](#) website
        1. Has your Skill appeared on this website?



a.

2. Click on the your skill and in the left panel, click JSON Editor



a.

3. Do you see your invocation name and sample phrases in the JSON?

b. Hopefully it updated! You're not quite done yet, though. Now, we want to cause Alexa to *do something* in response to what you say.

6. Create the **Endpoint Function** (the reaction that Alexa has to the phrases you say) by arranging the blocks similarly to what's shown here:

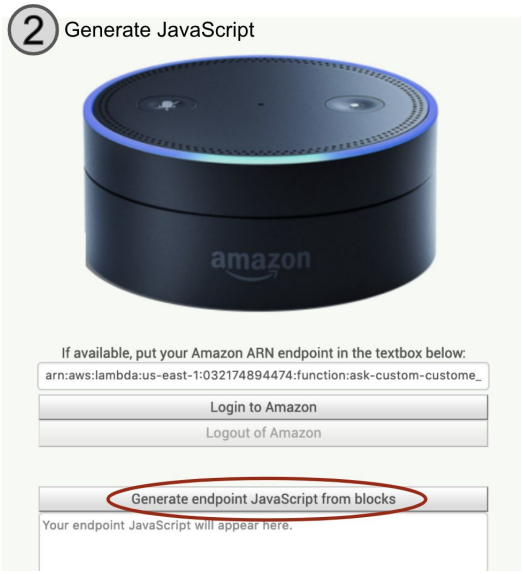
a.

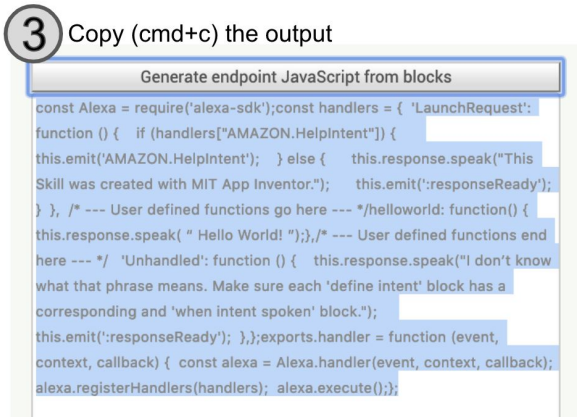
   i. Find the "when <intent_name> intent spoken" and the "say" blocks in the "Voice" drawer, and the "<text>" block in the "Text" drawer

   ii. Note that you should select the name of your intent (e.g., Fallback) from the drop-down menu, or type a custom name in yourself

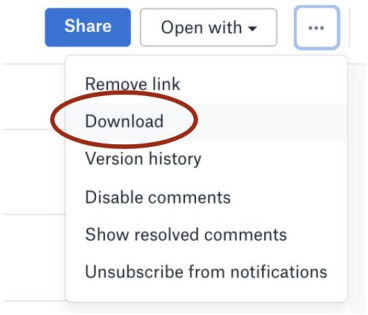
7. Go back to the designer page and click "**Generate endpoint JavaScript from blocks**" and copy the output

   a. This generates JavaScript based on the "when intent spoken" functions that you defined

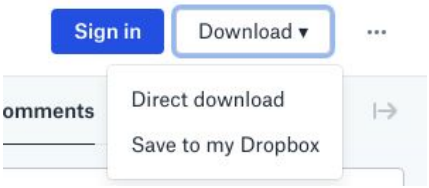




8. Download the following zip file by clicking the three dots in the top right hand corner of the Dropbox page: https://www.dropbox.com/s/4uiozk5j7gpetbx/HelloWorldIntentExample.zip?dl=0

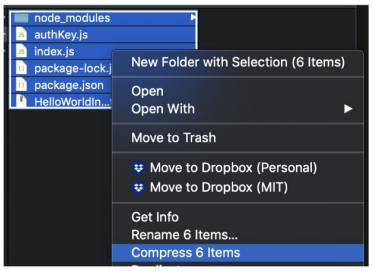


If you are not signed in, then just click the “direct download” download link



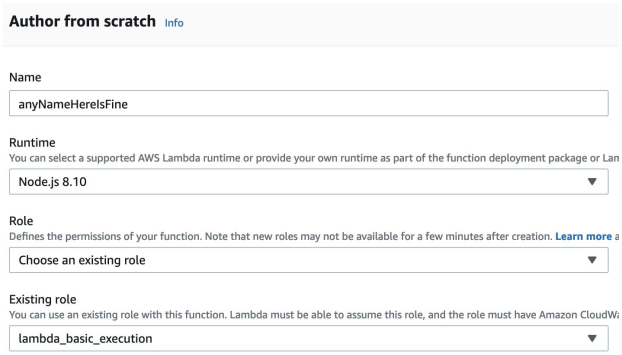
9. Unzip the folder, and open index.js
    a. Select all of the text in the index.js file and delete it
    b. Paste the JavaScript output you generated in a previous step into this index.js and save the file
        i. Please check this file for left/right quotation marks (i.e., “ or ”), replace them with non-directional quotation marks (i.e., **"**)
            1. Make a note in the bug testing notes the of where the left/right quotation marks occurred in your function (e.g., paste the JavaScript output and highlight/bold the left/right quotes
10. Zip the folder and upload it to AWS:

Zip your function by selecting all of the files in your folder (including index.js), and clicking “compress” (on a Mac) or “Send to” → “Compressed (zipped) folder” (on Windows). (NOTE: Don’t just compress the containing folder or you will get errors later on!)



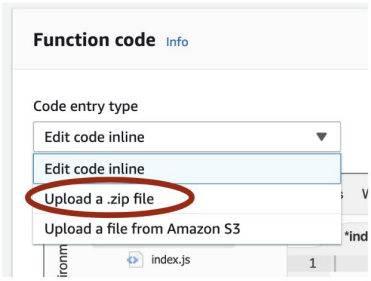
a. Now, we’re going to upload this .zip to AWS Lambda. Go to https://console.aws.amazon.com/lambda/, and log in
b. Make sure that your location on AWS is set to **North Virginia** (otherwise you won’t be able to connect your endpoint to the Alexa VUI):

c. Create and name a function by authoring from scratch, select its existing role (as below), and then click “Create Function”.

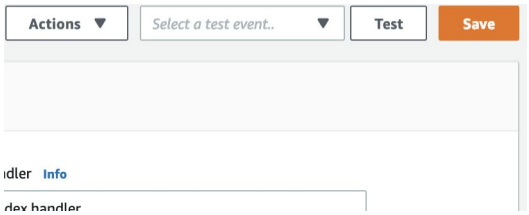


   i. If you don’t see the lambda_basic execution role, then instead of “Choose an existing role”, choose “Create a custom role”
      1. In the new tab, make sure the IAM role is lambda_basic_execution
      2. Click “allow” in the bottom right
      3. Back on the Alexa Skill, “Author from scratch”, webpage, choose the lambda_basic_execution role

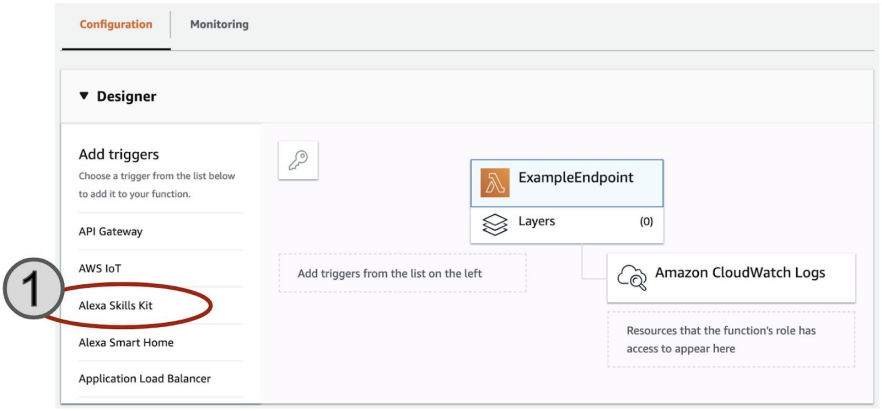
d. Finally, upload your newly zipped JavaScript to AWS by scrolling down to “Function Code” and clicking on upload a .zip file.



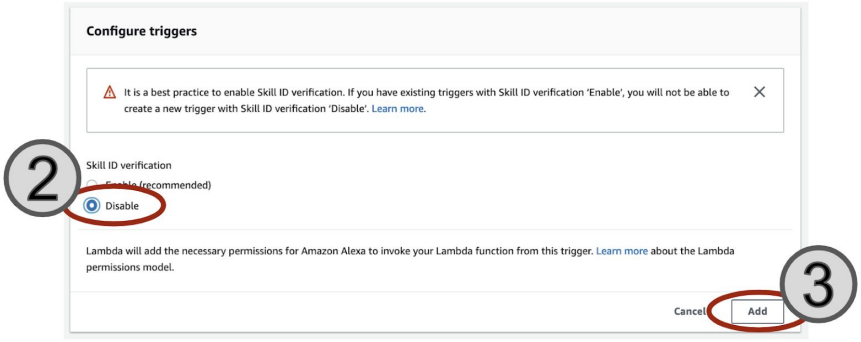
e. Save your Lambda function by scrolling to the top of the page and clicking the orange save button



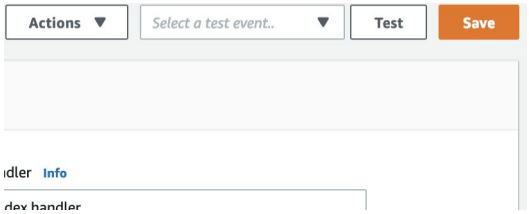
11. Update the endpoint on AWS to include Alexa Skills:
    a. On the same AWS Lambda webpage, under configuration, click Alexa Skills Kit to add it as a trigger



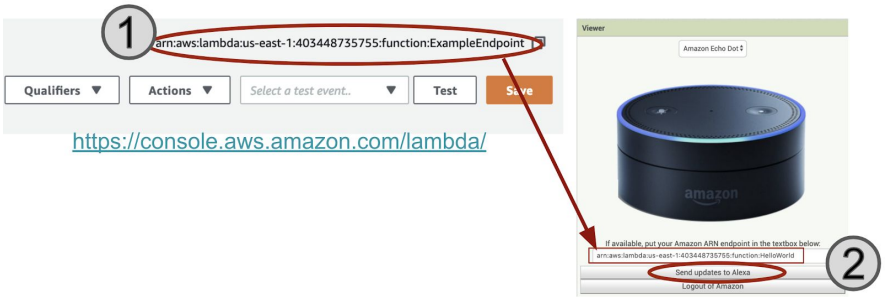
    b. Disable Skill ID verification and add the trigger to your project:



c. Save your Lambda function by scrolling to the top of the page and clicking the orange save button

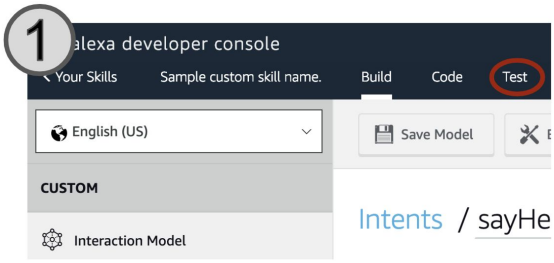


12. Connect the Voice User Interface and Endpoint function:
    a. Once you have created an endpoint function on AWS Lambda, you can connect your Voice User Interface to the endpoint by copying the Lambda function's ARN into App Inventor, and updating your skill:
        i. The ARN can be found in the top right hand corner of the AWS webpage where you uploaded your .zip file:

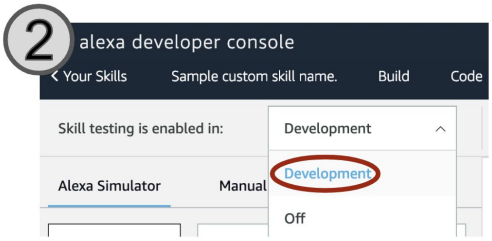


13. Congrats! You made your first MIT App Inventor Alexa Skill :-)
14. Now, you can try out the skill using the Alexa Developer Console

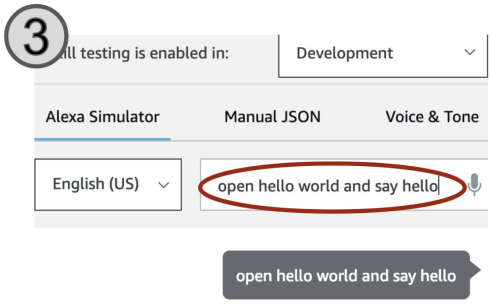
a. Go back to the [Alexa Developer website] and click on test:



b. Select Development testing:



c. Try out your skill by typing, “open <your_invocation_name> and <your_intent_phrase>”, or just, “<your_intent_phrase>”, and see how Alexa responds.



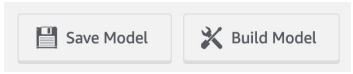
Testing Groups

What to test (Click link for more info)	**Tester 1**	**Tester 2**	**Tester 3**
VUI: Intent blocks			
VUI: Define Invocation Name			
VUI: Slot blocks			
Lambda: Say block			
Lambda: Send to App Inventor	N/A	N/A	N/A
Lambda: Generate response to seed text	N/A	N/A	N/A
Sending Updates to Amazon	N/A	N/A	N/A
Control Blocks			N/A
Variables			N/A
Lists			N/A
Logic			N/A
Math			N/A
Text			N/A

VUI: Intent blocks

	Defines the **phrases you say** when you are using the Alexa Skill (e.g., “Alexa, read me the story” and “Alexa, what does the story say?”).
	Like the block above, these blocks define **phrases you say**. Each block has a specific purpose (e.g., for the *help intent* block, the phrases might be “Alexa, I don’t understand” and “Alexa, how do I use this?”)

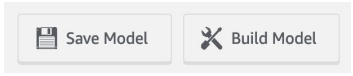
What you might consider trying/testing:

- Try defining various intents with various phrase lists
- Update your skill using the “Send updates to Alexa” button on the Designer page
- Go to the Alexa Developer console, click on your skill and see if your updates took place
    - Click on JSON Editor → Are your updates there?
    - Save and build the model
        - 
        - What kind of errors do you come across?

VUI: Define Invocation Name

	Defines the **name you say** when you open the Alexa Skill (e.g., “Alexa, open My Story Book”).

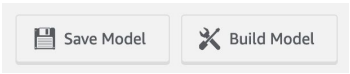
What you might consider trying/testing:

- Try defining various invocation names with various text blocks (e.g., using the “join” block)
- Update your skill using the “Send updates to Alexa” button on the Designer page
- Go to the Alexa Developer console, click on your skill and see if your updates took place
    - Click on JSON Editor → Are your updates there?
    - Save and build the model
        - 
        - What kind of errors do you come across?

VUI: Slot blocks

define slotName slot using slot type DATE	Defines a **slot with a specific type**. Slots are **variables that are filled when you talk to Alexa**. E.g., after defining a slot that you named "noms" that is of type "Food", you can add this slot to your intents, and say to Alexa, "Alexa, order me some pizza" (where "pizza" would fill the slot). See the example on page 3.
define slotName slot using slot type	Defines a **slot with any type**. This block works in the same way as the above block, but instead of selecting a slot type from a drop down menu, you **connect a *text* block** with your own slot type. The slot type must be one of the accepted Amazon slot types.
get slot slotName	Used as a **placeholder for a slot** when defining an intent. See the example on page 3.

What you might consider trying/testing:

- define slots with different types from the dropdown
- define slots with types from the Amazon slot types
- use the get slot block in an intent
- Update your skill using the "Send updates to Alexa" button on the Designer page
- Go to the Alexa Developer console, click on your skill and see if your updates took place
    - Click on JSON Editor → Are your updates there?
    - Save and build the model
    - 

    - What kind of errors do you come across?

Lambda: Say Block

say	Causes **Alexa to say the words** in the *text* blocks connected to this block.

What you might consider trying/testing:

- try making alexa say various things using various text blocks (e.g., the join block)
- get the javascript and copy-paste it into the zip and upload to AWS Lambda, as described in the handout
    - Look at the javascript you copy-pasted. Does it make sense? Is it what you wanted to create with your blocks?
    - Save the Lambda function
- connect the endpoint to the VUI, as described in the handout
- Test your skill (with a corresponding VUI intent). What errors do you run into?

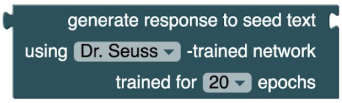
Lambda: Send to App Inventor Block

	**Sends** the information contained in this block **to the App Inventor app**. The App Inventor app must have a **CloudDB component** to receive this information. E.g., if this block contained a *text* block that said “play music”, the CloudDB component in the app would receive the text, “play music”. See the example below.

What you might consider trying/testing:

- Look at the example on the bottom of page 8 of the handout and try to understand it
- Try sending multiple values to the CloudDB, using the values in various contexts
- get the javascript and copy-paste it into the zip and upload to AWS Lambda, as described in the handout
    - Look at the javascript you copy-pasted. Does it make sense? Is it what you wanted to create with your blocks?
    - Save the Lambda function
- connect the endpoint to the VUI, as described in the handout
- Test your skill (with a corresponding VUI intent). What errors do you run into?

Lambda: Generate response to seed text

	**Generates text** using a particular **machine learning model**. The text generation uses the *text* block connected to this block as its starting point.

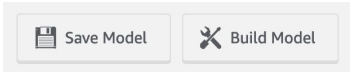
What you might consider trying/testing:

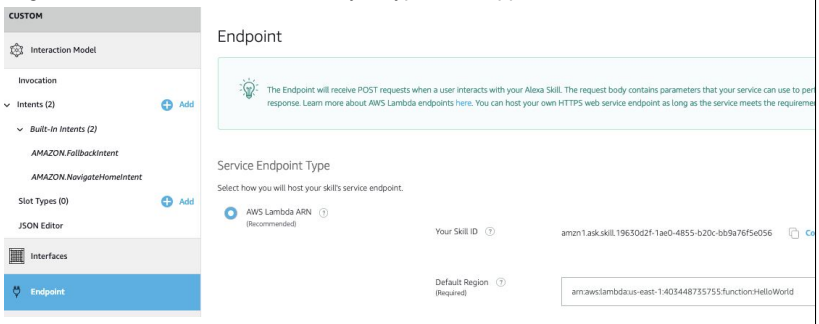
- Using the output from the block in another block, or making the seed text block a non-trivial text block (i.e. join or the result of a substring)

- get the javascript and copy-paste it into the zip and upload to AWS Lambda, as described in the handout
    - Look at the javascript you copy-pasted. Does it make sense? Is it what you wanted to create with your blocks?
    - Save the Lambda function
- connect the endpoint to the VUI, as described in the handout
- Test your skill (with a corresponding VUI intent). What errors do you run into?

Sending Updates to Amazon

Check to make sure updates are correctly sent to the Amazon Developer Console. There are three main things to test:

1. The content of the blocks are uploaded to the console:
    a. Create a VUI with various blocks (e.g., define invocation name, define slot, etc.)
    b. Go to the Alexa Developer console, click on your skill and see if your updates took place
        i. Click on JSON Editor → Are your updates there?
        ii. Save and build the model
            1. 

            2. What kind of errors do you come across?
2. The endpoint is correct in the console:
    a. Make an endpoint on Lambda and add it to your skill, as described in the handout
        i. Go to the Amazon Developer Console and check to see if the endpoint was updated. Click on “endpoint” (in blue below) and check the “Default Region” ARN-- Is it the same as what you typed into App Inventor
        ii. 

3. The endpoint is saved in App Inventor and correctly loads when you log in / log out
    a. Try logging in / logging out of App Inventor at various steps during the process of updating an endpoint

b. After you sent the endpoint ARN to Amazon and logged in / logged out, was the ARN the same in App Inventor or did it change back to an old ARN or change to a default ARN?
    i. i.e., after “sending updates to alexa” with a new endpoint ARN, is does the value in the textbox below stay as your new endpoint?
    ii. 

Control Blocks

If (else), for, while, etc.

What you might consider testing: Nested loops, nontrivial conditions, arbitrary number of else statements → put these in the “when intent spoken” block and check if the function that’s outputted in the console correctly creates these loops etc.

Variables

Setting and getting local and global variables

What you might consider testing: Variables with similar names, testing that variables are accessible within their scopes → use these in the “when intent spoken” block and check if the function that’s outputted in the console correctly accesses the variables etc.

Lists

Operations on lists of items

What you might consider testing: Making lists of arbitrary size, operations on lists, using results of list operations as inputs for other blocks → put these in the “when intent spoken” block and check if the function that’s outputted in the console correctly makes a list etc.

Logic

Booleans, comparisons, etc

What you might consider testing: Nested comparisons, comparing non-trivial elements, using outputs as input for other functions → put these in the “when intent spoken” block and check if the function that’s outputted in the console correctly creates if statements etc.

Math

Numbers and basic operations

What you might consider testing: Nested operations, non-trivial computations, using outputs as inputs for other blocks → put these in the “when intent spoken” block and check if the function that’s outputted in the console correctly has the math information etc.

Text

Lexical operations

What you might consider testing: Join combinations, functions on text, using outputs as inputs for other blocks → put these in the “when intent spoken” block and check if the function that’s outputted in the console correctly has the text information etc.

Bibliography

[1] Harold Abelson. MIT App Inventor sources. `https://github.com/mit-cml/appinventor-sources`, 2019.

[2] Hua Ai, Rohit Kumar, Dong Nguyen, Amrut Nagasunder, and Carolyn P Rosé. Exploring the effectiveness of social capabilities and goal alignment in computer supported collaborative learning. In *International Conference on Intelligent Tutoring Systems*, pages 134–143. Springer, 2010.

[3] Amazon. AWS SDK for JavaScript in Node.js. `https://aws.amazon.com/sdk-for-node-js/`, 2019.

[4] Chris Belanger. Confidence and common challenges: The effects of teaching computational thinking to students ages 10-16. Master's thesis, St. Catherine University, St. Paul, Minnesota, May 2018.

[5] Stephanie Bell. Project-based learning for the 21st century: Skills for the future. *The Clearing House*, 2010.

[6] Tim Bell, Paul Curzon, Quintin Cutts, Valentina Dagiene, and Bruria Haberman. Overcoming obstacles to CS education by using non-programming outreach programmes. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 71–81. Springer, 2011.

[7] Tanya N Beran, Alejandro Ramirez-Serrano, Roman Kuzyk, Meghann Fior, and Sarah Nugent. Understanding how children understand robots: Perceived animism in child–robot interaction. *International Journal of Human-Computer Studies*, 69(7-8): 539–550, 2011.

[8] Dan Bohus and Alexander I Rudnicky. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda. In *Eighth European Conference on Speech Communication and Technology*, 2003.

[9] Cynthia Breazeal. Social interactions in HRI: the robot view. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2):181–186, 2004.

[10] Karen Brennan and Mitchel Resnick. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual American Educational Research Association meeting, Vancouver, Canada*, volume 1, page 25, 2012.

[11] Joost Broekens, Marcel Heerink, Henk Rosendal, et al. Assistive social robots in elderly care: a review. *Gerontechnology*, 8(2):94–103, 2009.

[12] Jessica Van Brummelen. LSTM web server: Index. https://github.com/jessvb/lstm_web_server/blob/master/index.js, 2019.

[13] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, pages 77–91, 2018.

[14] Jenna Burrell. How the machine 'thinks: Understanding opacity in machine learning algorithms. *Big Data & Society*, 3(1):2053951715622512, 2016.

[15] John-John Cabibihan, Hifza Javed, Marcelo Ang, and Sharifah Mariam Aljunied. Why robots? a survey on the roles and benefits of social robots in the therapy of children with autism. *International Journal of Social Robotics*, 5(4):593–618, Nov 2013. ISSN 1875-4805. doi: 10.1007/s12369-013-0202-2. URL https://doi.org/10.1007/s12369-013-0202-2.

[16] Pew Research Center. Topline questionnaire: Voice assistants. Technical report, Pew Research Center, May 2017.

[17] Beverly Clarke and James McClung. AIinSchools, 2018. http://aiinschools.com, Last accessed on 2018-10-16.

[18] B.J. Copeland. Artificial intelligence, 2018. https://www.britannica.com/technology/artificial-intelligence, Last accessed on 2018-08-12.

[19] Paul Curzon. A bit of cs4fn: AIs, 2016. https://abitofcs4fn.org/ais/, Last accessed on 2018-10-16.

[20] Paul Cutsinger and Amazon. 4 must-have design patterns for engaging voice-first user interfaces, 2018. https://build.amazonalexadev.com/vui-vs-gui-guide-ww.html, Last accessed on 2018-10-17.

[21] Stefania Druga. Growing up with AI: Cognimates: from coding to teaching machines. Master's thesis, Massachusetts Institute of Technology, 2018.

[22] Stefania Druga, Randi Williams, Cynthia Breazeal, and Mitchel Resnick. Hey Google is it ok if i eat you?: Initial explorations in child-agent interaction. In *Proceedings of the 2017 Conference on Interaction Design and Children*, pages 595–600. ACM, 2017.

[23] Mateusz Dubiel, Martin Halvey, Leif Azzopardi, and Sylvain Daronnat. Investigating how conversational search agents affect user's behaviour, performance and search experience. In *The Second International Workshop on Conversational Approaches to Information Retrieval*, 2018.

[24] Anthony W Flores, Kristin Bechtel, and Christopher T Lowenkamp. False positives, false negatives, and false analyses: A rejoinder to machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. *Fed. Probation*, 80:38, 2016.

[25] Apps for Good. Apps for Good: Machine learning, 2018. https://www.appsforgood.org/courses/machine-learning, Last accessed on 2018-10-16.

[26] Charles L Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. In *Readings in Artificial Intelligence and Databases*, pages 547–559. Elsevier, 1989.

[27] Bill Franks. Why Google Duplex did not pass the turing test, 2018. https://iianalytics.com/research/why-google-duplex-did-not-pass-the-turing-test, Last accessed on 2018-08-12.

[28] David Gewirtz. Google Duplex beat the Turing test: Are we doomed?, 2018. https://www.zdnet.com/article/google-duplex-beat-the-turing-test-are-we-doomed/, Last accessed on 2018-08-12.

[29] Google. AIY: Do-it-yourself artificial intelligence, 2018. https://aiyprojects.withgoogle.com/, Last accessed on 2018-10-16.

[30] Google. Conversation design, 2018. https://developers.google.com/actions/design/, Last accessed on 2018-10-17.

[31] Google. Conversation design - learn about conversation, 2018. https://designguidelines.withgoogle.com/conversation/conversation-design/learn-about-conversation.html, Last accessed on 2018-10-17.

[32] Google. Google forms, 2019. https://www.google.com/forms/about/, Last accessed on 2019-04-22.

[33] H Paul Grice. Logic and conversation. *1975*, pages 41–58, 1975.

[34] Shuchi Grover and Roy Pea. Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, 42(1):38–43, 2013.

[35] Randy Allen Harris. *Voice interaction design: crafting the new conversational speech systems*. Elsevier, 2004.

[36] Tommy Heng. LSTM text generator: Train model. https://github.com/tomiyee/lstm-text-gen/blob/master/train-model.py, 2019.

[37] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *Logic Journal of IGPL*, 8(3):239–263, 2000.

[38] Amazon Inc. Change the wake word, 2018. https://www.amazon.com/gp/help/customer/display.html?nodeId=201971890, Last accessed on 2018-08-21.

[39] Amazon Inc. Implement the built-in intents, 2018. https://developer.amazon.com/docs/custom-skills/implement-the-built-in-intents.html, Last accessed on 2018-08-28.

[40] Amazon Inc. Skill blueprints, 2018. https://blueprints.amazon.com/, Last accessed on 2018-08-01.

[41] Amazon Inc. Slot type reference, 2018. https://developer.amazon.com/docs/custom-skills/slot-type-reference.html, Last accessed on 2018-08-28.

[42] Amazon Inc. Voice design guide, 2018. https://developer.amazon.com/designing-for-voice/, Last accessed on 2018-10-17.

[43] Amazon Inc. Alexa developer console, 2019. https://developer.amazon.com/alexa/console/ask, Last accessed on 2019-04-13.

[44] Jovo. Jovo: Build Alexa skills and Google actions, 2019. https://www.jovo.tech/, Last accessed on 2019-04-27.

[45] Ken Kahn and Niall Winters. Child-friendly programming interfaces to AI cloud services. In *European Conference on Technology Enhanced Learning*, pages 566–570. Springer, 2017.

[46] Keras. LSTM text generation example. https://github.com/keras-team/keras/blob/master/examples/lstm_text_generation.py, 2019.

[47] Lorenz Cuno Klopfenstein, Saverio Delpriori, Silvia Malatini, and Alessandro Bogliolo. The rise of bots: a survey of conversational interfaces, patterns, and paradigms. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, pages 555–565. ACM, 2017.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[49] Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Bjorn Hoffmeister, and Markus Dreyer. Just ASK: Building an architecture for extensible self-service spoken language understanding. *arXiv preprint arXiv:1711.00549*, 2017.

[50] Natalie Lao. Developing cloud and shared data capabilities to support primary school students in creating mobile applications that affect their communities. Master's thesis, Massachusetts Institute of Technology, 2017.

[51] Yaniv Leviathan and Yossi Matias. Google Duplex: An AI system for accomplishing real-world tasks over the phone, 2018. https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html, Last accessed on 2018-08-12.

[52] Toby Jia-Jun Li, Igor Labutov, Brad A Myers, Amos Azaria, Alexander I Rudnicky, and Tom M Mitchell. An end user development approach for failure handling in goal-oriented conversational agents. *Studies in Conversational UX Design. Springer*, 2018.

[53] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4990–4998, 2017.

[54] Thomas Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, 2008.

[55] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The Scratch programming language and environment. *Trans. Comput. Educ.*, 10(4):16:1–16:15, November 2010. ISSN 1946-6226. doi: 10.1145/1868358.1868363. URL http://doi.acm.org/10.1145/1868358.1868363.

[56] Metro. A group of tween girls use a competition to launch Hello Navi, an app designed to help their blind classmate, 2015. https://www.metro.us/lifestyle/a-group-of-tween-girls-use-a-competition-to-launch-hello-navi-an-app-designed-to-help-their-blind-classmate/zsJojD---oxr1jEO4WkQoo/, Last accessed on 2018-07-19.

[57] Marvin Minsky. *Society of mind*, chapter 30.8. Simon and Schuster, 1988.

[58] MIT. MIT ESP: HSSP, 2018. https://esp.mit.edu/learn/HSSP/index.html, Last accessed on 2019-04-23.

[59] MIT. MIT Spark: HSSP, 2018. https://esp.mit.edu/learn/Spark/index.html, Last accessed on 2019-05-03.

[60] MIT App Inventor. Anyone can build apps that impact the world, 2017. http://appinventor.mit.edu/explore/, Last accessed on 2018-07-31.

[61] MIT App Inventor. Stories, 2017. http://appinventor.mit.edu/explore/stories.html, Last accessed on 2018-07-19.

[62] Gabriel Axel Montes and Ben Goertzel. Distributed, decentralized, and democratized artificial intelligence. *Technological Forecasting and Social Change*, 141: 354 – 358, 2019. ISSN 0040-1625. doi: https://doi.org/10.1016/j.techfore.2018.11.010. URL http://www.sciencedirect.com/science/article/pii/S0040162518302920.

[63] Bureau of Labor Statistics. Occupational outlook handbook, 2018. https://www.bls.gov/ooh/, Last accessed on 2018-08-08.

[64] Laurie M. Orlov. The future of voice first technology and older adults. Technical report, Aging in Place Technology Watch, February 2018.

[65] Steven Ovadia. Automate the internet with "if this then that" (IFTTT). *Behavioral & social sciences librarian*, 33(4):208–211, 2014.

[66] R. W. Picard, S. Papert, W. Bender, B. Blumberg, C. Breazeal, D. Cavallo, T. Machover, M. Resnick, D. Roy, and C. Strohecker. Affective learning — a manifesto. *BT Technology Journal*, 22(4):253–269, October 2004. ISSN 1358-3948. doi: 10.1023/B:BTTJ.0000047603.37042.33. URL http://dx.doi.org/10.1023/B:BTTJ.0000047603.37042.33.

[67] National Public Radio. The smart audio report, winter 2018. Technical report, NPR and Edison Research, December 2018.

[68] redislabs. Redis. https://redis.io/, 2019.

[69] Inc. Run Dexter. Dexter, 2018. https://rundexter.com/platform/, Last accessed on 2018-08-16.

[70] Laio Oriel Seman, Romeu Hausmann, and Eduardo Augusto Bezerra. On the students' perceptions of the knowledge formation when submitted to a project-based learning environment using web applications. *Computers & Education*, 117:16 – 30, 2018. ISSN 0360-1315. doi: https://doi.org/10.1016/j.compedu.2017.10.001. URL http://www.sciencedirect.com/science/article/pii/S0360131517302208.

[71] Robert J. Sternberg. Human intelligence, 2018. https://www.britannica.com/science/human-intelligence-psychology, Last accessed on 2018-08-12.

[72] Al Sweigart. *Scratch Programming Playground: Learn to Program by Making Cool Games*. No Starch Press, 2016.

[73] Andrew Tarantola. Pretty sure Google's new talking AI just beat the Turing test, 2018. https://www.engadget.com/2018/05/08/pretty-sure-googles-new-talking-ai-just-beat-the-turing-test/, Last accessed on 2018-08-12.

[74] James W Taylor and Roberto Buizza. Neural network load forecasting with weather ensemble predictions. *IEEE Transactions on Power systems*, 17(3):626–632, 2002.

[75] TensorFlow.js. LSTM text generation example. https://github.com/tensorflow/tfjs-examples/tree/master/lstm-text-generation, 2019.

[76] David Touretzky, Fred Martin, Christina Gardner-McCune, and Deborah Seehorn. ai4k12, 2018. https://github.com/touretzkyds/ai4k12/wiki, Last accessed on 2018-10-16.

[77] Jessica Van Brummelen, Marie O'Brien, Dominique Gruyer, and Homayoun Najjaran. Autonomous vehicle perception: The technology of today and tomorrow. *Transportation research part C: emerging technologies*, 2018.

[78] Vanessa Van Brummelen. The X-mazing Zorilla, 2018.

[79] Peter Voss. No, Google Duplex didn't pass the turing test, 2018. https://uxdesign.cc/no-google-duplex-didnt-pass-the-turing-test-93f7235e6c40, Last accessed on 2018-08-12.

[80] Web of Science. Web of Science topic: (computer vision): 40,907 results; topic: (conversational artificial intelligence): 89 results, 2018. http://apps.webofknowledge.com/, Last accessed on 2018-09-06.

[81] Karen Weise. Hey, Alexa, why is Amazon making a microwave?, 2018. https://www.nytimes.com/2018/09/20/technology/amazon-alexa-new-features-products.html, Last accessed on 2019-03-30.

[82] Randi Williams. PopBots: Leveraging social robots to aid preschool children's artificial intelligence education. Master's thesis, Massachusetts Institute of Technology, 2018.

[83] Patrick H Winston and Sarah A Shellard. *Artificial intelligence at MIT: expanding frontiers*. MIT Press, 1990.

[84] Stephen Wolfram. *An elementary introduction to the Wolfram Language*, chapter 22. Wolfram Media, Incorporated, second edition, 2017.

[85] Sam Wong. Should you let Google's AI book your haircut?, 2018.

[86] Li Zhou. Is your software racist, February 2018. URL https://www.politico.com/agenda/story/2018/02/07/algorithmic-bias-software-recommendations-000631.