KATbot: Final Design Document

Authors: Ashika Koganti, Abha Agrawal, Jade Traiger: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract* -- KATbot is a MadLib style storytelling robot that interacts with children to aid in language and reading comprehension. It is inspired by a storytelling companion robot created by the MIT Personal Robot Group [1]. This robot had children create stories for the robot to tell by manipulating images on a tablet. In contrast, KATbot takes in words through speech input to personalize short stories, similar to a MadLib, or fill in the blank story, and includes a display that allows the user to read along with the story. The major components of KATbot are a Raspberry Pi 4 that connects to the displays, a microphone, a speaker, and one degree of freedom (DoF) robot arms. The Raspberry Pi communicates with a laptop running the story generation algorithm. KATbot integrates signal processing, machine learning, and robotics to provide an educational and engaging user experience for elementary school age children. This paper will explain in detail the requirements, system design and components, validation plan, and project schedule.**

***Index Terms* -- Machine Learning, Robot, Speech Processing, Story Generation, Storytelling, Synonym Generation, Text to Speech**

I. INTRODUCTION

RESEARCH shows that early language development directly relates to performance in later education [2]. It is crucial to help children improve their language skills at a young age. Since language is an essential component of social and interactive context, assisting children in language development and comprehension would help them become an active part of society [3]. Robot companions would be instrumental in this area because they integrate the benefits of technology, such as accessibility and adaptive, easy-to-update software, with the benefits of social agents, such as communication skills and understanding social cues [4].

For these reasons, we created a robotic learning companion to help children with language and reading development. It is not only important that our robot is a useful language learning toy, but equally important that it has a friendly, clean, and engaging user interface for the children to interact with.

KATbot is a standalone interactive robot with the following qualities. It tells customized stories by asking the user to fill in parts of the story as it goes. It responds to user input within 4 - 6 seconds, and has at least 85% word recognition accuracy, at least 90% part of speech accuracy, at least 85% synonym generation accuracy, cohesive stories, and relatively good user satisfaction. Testing with children was outside of the scope of what we could accomplish this semester, therefore KATbot was tested by adult users.

II. DESIGN REQUIREMENTS

Some of our requirements, especially within hardware, were adapted to address the constraints placed on us due to the COVID-19 pandemic situation. They will be addressed on an individual basis in the following sections.

A. Hardware Requirements

The hardware aspect of KATbot consists of a custom-made robot, embedded processors that communicate with peripherals, and a laptop running a story generation algorithm. The peripherals in KATbot consist of a microphone, speaker, and displays as well as motor drivers and motors.

We chose to create a custom-made robot instead of using an off-the-shelf product because it would not only let us create a friendly looking robot suitable for children, but also let us design a product that could hold the electronics needed for this project.

To build KATbot, we needed a processor with enough processing power to handle the speech processing, text to speech, and user-facing display algorithms. Since our product is meant to be standalone, this processor needed to be on a small form factor to reduce the space it would take up inside the robot. The processor needed to have many interfaces, such as HDMI, USBs, GPIOs, etc) to connect to the many peripherals we have for the product. Finally, the processor needed to be low cost so that the product's bill of materials stays within the $600 constraint.

KATbot has two robot arms, each of which are one degree of freedom. The arms move to help show emotion while the robot tells the story and interacts with the user. Although the arms were originally going to be made of acrylic and PLA filament, due to a lack of access to a laser-cutter and 3D-printer with the COVID-19 pandemic, the arms were constructed from cardboard. Based on the weight of cardboard, the motors for the arms needed to have a minimum torque of 1.18 kg/cm. The calculations are below.

Assume that the arm dimensions are $6in \times 2in \times 2in$. *The acrylic frame covers a surface area of* $56\ in^2$. *Cardboard has a height of* $0.125in$. *Therefore, the volume is* $7\ in^3 = 114\ cm^3$. *The density of cardboard is* $0.69\ g/cm^3$. *Therefore, the mass is* $114cm^3 * 0.69g/cm = 78.66g$. *Torque is F*r, where F =* $0.078kg$ *and r = 6in = 15.24cm.*

Therefore, Torque = $1.18\ kg/cm$.

Fig. 1. Torque Calculations for the robot arms

KATbot also has displays on its body. It has a display on the torso of the robot that shows the current sentence that the robot is speaking. This is to aid the user with reading comprehension. It also has a face display for KATbot's face, so that it can show emotions. Eye movements and gestures are incredibly important when expressing emotion [5]. We chose to use a display for the face so that we could dynamically change the emotions that the eyes and mouth express to match the emotions of the story.

B. Software Requirements

The story generating component of KATbot has four main components. First, it needs to preprocess story templates prior to use for lower latency and to prepare for user input. Next, it should process the user input word by word, check for correct word semantics in the context of the story, and update the story as it goes. On the user interaction side, KATbot needs to be able to communicate effectively what type of word (part of speech or entity) it requires for each blank the user needs to fill in. On the algorithm side, it needs the ability to continue the story with any grammatically correct user input (i.e. the user should not be able to 'break' the algorithm with an unexpected input). To do this, the algorithm uses two features: synonym generation to fill in the story cohesively and part of speech detection to enforce correct syntax. For part of speech detection, we were aiming for a 90% accuracy level, which is around the minimum of popular natural language models [6]. For synonyms, we wanted at least an 85% similarity rate for generated synonyms and antonyms, based on a paper on supervised learning synonym identification models, which evaluates a number of different synonym detection and recall algorithms [7]. Another major requirement is to keep the language and word choice at a child friendly level, to match the educational level of our target audience.

In addition, the storytelling feature of KATbot has two comprehensive requirements: cohesion and enjoyment. Cohesion refers to how well the story flows and each sentence is related to the previous ones. Enjoyment refers to user satisfaction and how likely users are to continue interacting with KATbot. More information on the evaluation of these requirements can be found in the validation plan.

Another major component of KATbot is the user interface. KATbot is meant to help early elementary school aged children who are working on their reading skills. This means that they cannot reliably interact with a completely text based system. We decided that in order to combat this problem, all users will interact with KATbot through speech.

In speech recognition, word error rate (WER) is a common measure of system error. Having a low WER is an indicator that a system is performing well. In 2017, it was found that Sphinx4, a commonly used open-source speech recognition package, had a 37% WER, while Microsoft's and Google's speech recognition APIs had a WER of 18% and 9% respectively [8]. With KATbot, we aimed to have a WER of 15% which is the average of the non open-source speech recognition APIs referenced.

We would like to acknowledge that automatic speech recognition systems have been shown to perform poorly with speech from young users [10]. In one human robot interaction study, the best speech recognition system under the best conditions could only achieve a 38% recognition rate on children's speech [9]. Because of poor recognition accuracy on children's speech, our WER metric was applied only for adult users.

Testing our system with children was outside of the scope of this project because we did not have the proper paperwork or resources to test KATbot with children. Because speech recognition for children is an ongoing challenge for the field, we designed KATbot to perform well with adult users.

C. Overall Requirements

Overall we aim for KATbot to be a standalone children's toy with a friendly and engaging user experience. Because KATbot is meant to be a toy, we intended for it to be played with wherever a child wants to. Due to this, originally KATbot would have needed to have batteries for power. We found that children at the age of 5 can play with something that interests them for around 15 minutes and can interact with other children for around 10 - 25 minutes [11]. Because KATbot is an inherently interactive system, we estimate children will want to play with it for up to 25 minutes. Thus, we originally required that KATbot should be able to run for 30 - 45 minutes off its own power system. However, due to the transition to remote instruction, we removed the power requirement and replaced it with a reliability metric. We would like the product to be able to be played with for 30-45 minutes without any reboot, loss of connection, or loss of power issues.

In order for KATbot to be a standalone toy, it needs to house all of the electronics for the product. These electronics include a processor that controls the peripherals, a microphone, a speaker, motors, and a motor shield for the robot arms, and displays.

Another important part of our system design is latency between user input and dialogue feedback. We have found that the average response time per person for task oriented dialogue between humans is around 4 - 6 seconds [12]. Thus, we aimed for having a total system latency of at most 4 - 6 seconds, which accounts for the time between when the user has finished uttering their response to the system and when the system says its next line of dialogue.

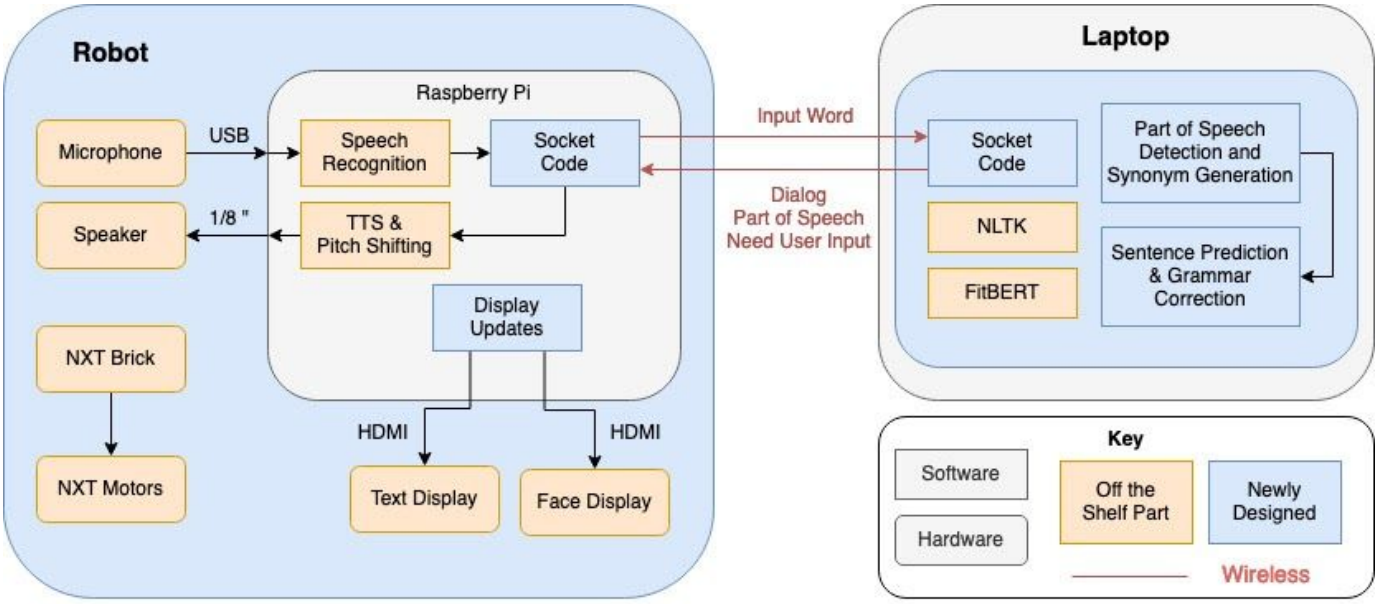

Fig. 2. Overall System Diagram. The storytelling algorithm is run on the laptop while audio input/output, the displays, and the arms are located and run in the robot itself. The laptop and Raspberry Pi communicate wirelessly via sockets. All the tools, packages, and hardware components that we incorporated are highlighted in orange.

III. ARCHITECTURE AND PRINCIPLE OF OPERATION

To meet these requirements, we built a standalone (all electronics are housed within the robot) custom-made robot. The robot houses a main processor that communicates with the peripherals and handles the speech processing, text to speech, and user-facing display algorithms.

The custom-made robot houses the processor and peripherals. The robot's body and head dimensions were carefully chosen to be big enough to fit all the electronics within it. The robot has two one-degree-of-freedom arms. We used motors with a small form factor, so that they could fit within the robot, and that met the torque requirement of 1.18 kg/cm.

We used a microphone with an omnidirectional pickup pattern and a pickup distance which can cover up to 5ft around the robot itself. The pickup pattern and distance were chosen so that the user can move around while interacting with the robot. We used speakers with a small form factor so that they could fit in the base of the robot.

The processor inside the robot communicates to a laptop that contains the story generation algorithm. Even though the end goal of this product was to be a standalone toy, given the time constraints of a semester project, we did not have enough time to not only have a completed story generation algorithm that meets our requirements, but also to have it run on an embedded processor. Therefore, we decided to run this algorithm on a laptop as part of the scope of this project.

The storytelling algorithm starts with templates that are manually configured based on a loose set of constraints that ensure context and clear word relations. KATbot prompts the user for the word of the specified part of speech for the blank, and the user input goes through error detection to ensure it is the right part of speech. To make the system a little more forgiving, particularly for children with limited grammar knowledge, small grammar mistakes are corrected (e.g. singular vs. plural). On the algorithm side, any related words are filled in using generated synonyms and antonyms of the user input in conjunction with a machine learning based best-fit heuristic. This amplifies how much the user can customize the story, which adds both variety to the stories as well as a better user experience.

IV. DESIGN TRADE STUDIES

A. Machine Learning Components

1. Part of speech tagging

We chose to use the NLTK speech processing package for part of speech tagging because it meets our metric of 90% accuracy. The default tagger in this package is the perceptron tagger, which was reported to have a 96.5% accuracy [14]. Another popular natural language model is SpaCy, which is more object-oriented than NLTK. While this is better for semantic meaning, NLTK conducts sentence tokenization much more quickly and has more tools and integrative capabilities than SpaCy [15]. Since the part of speech is the only feature we need for this portion, and NLTK is well suited

for other components, too, the advantages of using NLTK outweigh the advantages of SpaCy.

Table 1. Accuracy and Latency Measurements for Different NLTK Part of Speech Taggers.

Tagger	Accuracy	Latency
Perceptron	90.38%	0.000740 s
Bigram	74.60%	0.000741 s
Combined	93.82%	0.000740 s

The NLTK perceptron tagger yielded a 90.38% accuracy (see the validation plan for more information on how we tested accuracy). This met our requirement, but just barely. To boost this number, KATbot combines it with the NLTK bigram tagger. The perceptron tagger assigns part of speech based on a pre-trained machine learning model of word clusters, while the bigram tagger looks up the correct part of speech in a lookup table of the most frequently used word pairings, defaulting to the perceptron tagger if it cannot find anything [19]. The bigram tagger on its own only has a 74.6% accuracy, but when combined with the perceptron tagger, the whole system has an accuracy of 93.82%. This means that if the user input has the right part of speech when tagged with either tagger, it passes the error detection. The system latency for all three taggers (perceptron, bigram, and both), is about 0.00074 seconds, so there is no apparent disadvantage to using both.

2. Synonym/Antonym generation

For this aspect of word generation, we initially chose to continue with the NLTK package because it already has a large database for its synonym detection and recall capabilities. One alternative is word2vec, but this tool does not come with its own corpus for training, and it is focused only on word similarity.

To measure the accuracy of the synonym generation algorithm, we checked whether the generated words are listed on thesaurus.com. It is difficult to quantify whether the generated synonyms are the optimal synonyms, but this allowed us to at least see if they are commonly linked to the original word. When tested this way by generating synonyms and antonyms for a random set of 10 adjectives, the NLTK tool only had a 64.072% accuracy, but a relatively short latency of 0.07274 seconds. The tool also did not always return any synonyms for simple words, like “huge”, which affected the overall storytelling performance greatly.

We then looked at fetching the synonyms and antonyms directly from thesaurus.com. This guaranteed that all the words returned were accurate and that the system always returns something. The latency is a little bit longer, at 0.38575 seconds, but not enough to be noticeable. Upon observation, however, a lot of the generated words are at a much higher vocabulary level than the rest of the stories; for example, one resulting synonym of “huge” is “leviathan”. To combat this problem, we switched to using kids.wordsmyth.net, a kids dictionary and thesaurus. These words suit the story much better, and when this thesaurus cannot return any words, our algorithm resorts to thesaurus.com. Using both these thesauruses slows down the synonym generating system quite a bit, at 1.82123 seconds, but KATbot still meets its latency requirement and the performance benefits are significant enough to justify using this option.

3. FitBERT

We chose to use FitBERT to help fill in the templates given user input, because it is an open-source, pretrained model. According to documentation, BERT, FitBERT’s parent package, has had success with actual MadLibs, and its easy-to-use python tools work well when incorporated with the other modules of the story generation algorithm. Also, its grammar correction capabilities make it particularly useful, and it has a low latency, which is important for real time performance.

B. Speech Recognition

The evaluation metrics we used to evaluate speech recognition were as follows: ability for the package to be used on an embedded device, whether the package required connection to the internet, ease of installation and usage, and accuracy.

We evaluated several different packages including PocketSphinx, Julius and SpeechRecognition. Below is an evaluation of each package.

PocketSphinx is an API meant for speech recognition from CMU. Some of the benefits that PocketSphinx provides are that it can run on an embedded device and does not require internet connection. It is also stable, has been designed to have a small code footprint and attempts to reduce memory consumption [16]. Some of the downsides to PocketSphinx are that it requires a base library to get it running, it can be hard to install, and it does not have the best recognition accuracy of the packages we considered.

Julius is another open source API that we considered. Its benefits are that it is a real time system that has a low memory requirement (< 64 MB). However, we did not find much documentation for it in regards to running it on an embedded system.

The last package we considered was a python package called SpeechRecognition. SpeechRecognition supports several different speech recognition engines including CMU Sphinx, Google Speech Recognition, Google Cloud Speech API, Microsoft Bing Voice Recognition, IBM Speech to Text and others. It is also easily installable and can be installed on an embedded device.

We decided to go with the python SpeechRecognition package for the following reasons. The first and most important was configurability. SpeechRecognition allows the user to choose a recognition engine which adds flexibility. Specifically, we used the Google Speech Recognition engine. This engine requires a connection to the internet, but performs speech recognition better than other packages, and well enough to meet our metrics. In fact, the Google Speech API has been shown to be the best system for speech recognition in a study investigating speech recognition in human robot interaction for children [8].

C. Text to Speech

We evaluated several different text to speech packages for KATbot. These included Festival, Flite, eSpeak, say, spd-say, google_speech, gTTS (google text to speech), and AWS Polly. The evaluation metrics for the text to speech modules were that they had to be able to process the input words, generate speech quickly, and that the generated voice was clear and understandable.

Of the packages evaluated, Festival, Flite, eSpeak, say, and spd-say were eliminated because they all had voices that were either robotic or unnatural.

Google_speech, gTTS, and AWS Polly all offered good voices for the user to interact with. However, AWS Polly requires AWS credits which we wanted to avoid so that the users of KATbot would not have to pay for anything.

We chose gTTS from the remaining two packages as it was updated more frequently than google_speech. gTTS provides a female voice, with good pronunciation and minimal latency in processing; however, it also requires internet connection.

D. Pitch Shifting Algorithm

We perform pitch shifting on the speech output of gTTS to make KATbot's voice more appealing to the user. Pitch shifting refers to the process by which the pitch of the audio input is increased without changing the duration of the audio input. There are three main approaches that we considered. The first two involved writing our own pitch shifting software based on existing algorithms, and the last one involved using a python package to perform pitch shifting.

The first algorithm for pitch shifting we considered was Pitch Synchronous Overlap Add (PSOLA). It relies on looking at the input speech signal and identifying periodic amplitude peaks within the waveform. These periodic amplitude peaks correspond to pitch marks within the signal. Suppose the pitch period at a certain point in the signal is t_m. Then, samples are taken from around each pitch period by multiplying the samples with a Hanning window ($h_m[n]$) of a length that will allow it to overlap between 50 - 75% with the next windowed pitch period. For the mth windowed portion of the input signal, $x_m[n] = h_m[t_m - n] * x[n]$. To shift the pitch of the input signal, a new set of pitch periods are generated such that they are either spaced closer together or farther apart. Then, the original frames are placed at the new pitch period, t_q, such that the output signal has pitch period frames $y_m[n] = x_m[n + t_m - t_q]$ spaced at the new pitch period [17]. This algorithm requires careful analysis of the time domain signal to find pitch marks and pitch periods.

The second algorithm for pitch shifting we considered was phase vocoding. Phase vocoding relies on taking the short time fourier transform (STFT) of the original signal, $X[n,k]$. The STFT provides a method by which the frequency content at discrete frequencies of a signal can be evaluated at periodic time intervals. It differs from the discrete time fourier transform (DTFT) which provides the frequency content of the entire signal. The STFT is calculated by multiplying the input audio signal by periodic windows to generate frames, then calculating the DTFT of each frame. Once we have the STFT representation of the signal we can calculate both the magnitude and phase of each frame for each frequency. From the magnitude and phase information of each frame, we can calculate the instantaneous phase, or the derivative of the phase for each frame. To perform pitch shifting we increase the instantaneous phase at each frame for each frequency. Reconstruction is performed by summing up sinusoids at each frequency with the magnitude and altered instantaneous phase at each frame. Below is a block diagram of the system as well as relevant equations.

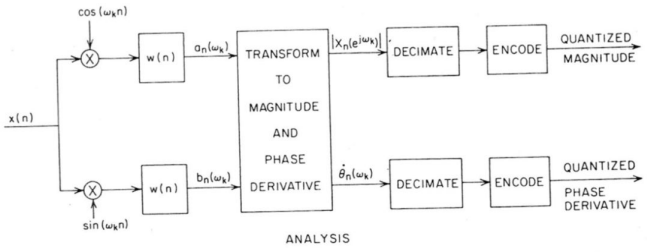

Fig. 6.54 Complete single channel of a phase vocoder analyzer.

Fig. 3. Complete single channel of a phase vocoder analyzer. The input comes in and is multiplied by a sine and cosine of frequency ω_k and convolved with a window to get the "real" and "imaginary response" $a_n(\omega_k)$ and $b_n(\omega_k)$ of the input at time window n and frequency ω_k [20]

$$|X_n(e^{j\omega k})| = (a_n(\omega_k)^2 + b_n(\omega_k)^2)^{1/2}$$

Fig 4. The equation for the magnitude response of the input given the "real" and "imaginary" responses for window n and frequency k.

$$\dot{\theta}_n(\omega_k) = \frac{b_n(\omega_k)\dot{a}_n(\omega_k) - a_n(\omega_k)\dot{b}_n(\omega_k)}{a_n^2(\omega_k) + b_n^2(\omega_k)}$$

Fig. 5. Discrete time instantaneous phase equation. Instantaneous phase is calculated per input window, n, and is calculated from the "real" and 'imaginary" responses of the input signal. Instantaneous phase is calculated using derivatives of the "real" and "imaginary" responses. It is the instantaneous phase that is changed in order to change the pitch of the input signal without changing the timing. [21]

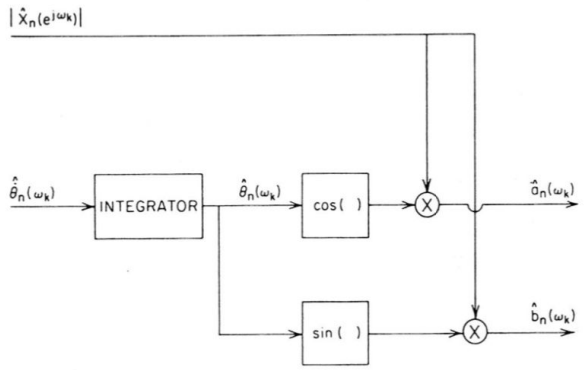

Fig. 6. Reconstruction of the output of a phase vocoder analyzer. The magnitude is preserved across the reconstruction, the instantaneous phase term is changed in order to shift pitch. Reconstruction of the signal at time n and frequency ω_k involves performing a continuous integral of the phase, then re-assembling the signal by applying the proper phase to a cosine and sine at frequency ω_k and multiplying it by the magnitude response.[22]

The third option for pitch shifting involved using AudioSegment from a python package called pydub. AudioSegment has built-in functions to take an audio input and play it back at a new sampling rate without changing the duration of the audio input.

E. Embedded Processor

The three embedded processors in the market today that satisfied our requirements were the Raspberry Pi 3, Raspberry Pi 4, and Nvidia Jetson Nano.

The Raspberry Pi 4 has upgraded specifications, as compared to the Raspberry Pi 3, with respect to the CPU, GPU, and RAM. It has two HDMI ports, unlike the Raspberry Pi 3 which is useful as each HDMI port could drive one of KATbot's two displays. Our processor needed to be able to handle fast computation to meet our latency requirements. More RAM aided in running the programs more quickly, reducing latency, which was a major concern of ours. Upgraded hardware, more ports, more memory, and faster processing and performance speeds caused the Raspberry Pi 4 to be a better fit for our project than the Raspberry Pi 3.

When comparing the Raspberry Pi 4 to the Jetson Nano, most of the specifications are very similar. However, the CPU on the Raspberry Pi 4 has a slightly faster clock and uses 20% less power than the CPU on the Nano. In addition, at $99, the Nano is almost double the price of the Raspberry Pi 4, which is important due to our constrained budget of $600.

We decided to go with the Raspberry Pi 4 as our robot's processor because it met our needs, was less expensive than the NVIDIA Jetson Nano, and had better specs than the Raspberry Pi 3.

F. Microphone

We had several different choices for a microphone system for KATbot. The options included a head mounted microphone, a table microphone, and a studio microphone. Head mounted microphones are good for spoken dialogue systems because the microphone is placed very close to the user's mouth. Similarly, studio microphones are designed to best capture human voice and have been shown to produce the best audio inputs for automatic speech recognition systems [9]. Despite offering high quality audio inputs to our system, we decided against a head mounted microphone and a studio microphone in order to maintain KATbot as a standalone product. Both types of microphones also restrict the user from moving around while interacting with the robot.

The other option that we considered was a conference table microphone with omnidirectional input that can be housed within or on KATbot. Despite the fact that the table microphone provides a worse audio signal input to speech recognition it can be easily integrated with the robot and has both omnidirectional pickup and pickup from a distance. This allows the user to move around while interacting with the robot.

G. Displays

The face display is important to be able to express emotion that matches the tone of the story. To do so, movements of both the eyes and the mouth is important. This is why we chose to have one display for the entire face that would show the eyes and the mouth, as opposed to two small displays for just the eyes.

It is important to have displays that meet the requirements, but draw minimal current to reduce power consumption. In addition, to meet our requirements, we needed displays that could communicate face shape/movement or text effectively to the user, and this does not require high resolution. Finally, it was important that any display that we choose can interface well with our embedded processor. We chose to use LCD screens, as opposed to LED screens, for KATbot since off-the-shelf LCD screens have the capability to remove backlight, and therefore, reduce power consumption. Although OLED screens use less power than LCD screens, we could not find any OLED screens with dimensions big enough for this project.

V. SYSTEM DESCRIPTION

A. Text to Speech

Text to speech is the primary way that KATbot communicates with the user. We used Google's text to speech package called gTTS as KATbot's text to speech synthesizer. gTTS has a simple API which takes in a string and converts it to a .mp3 file.

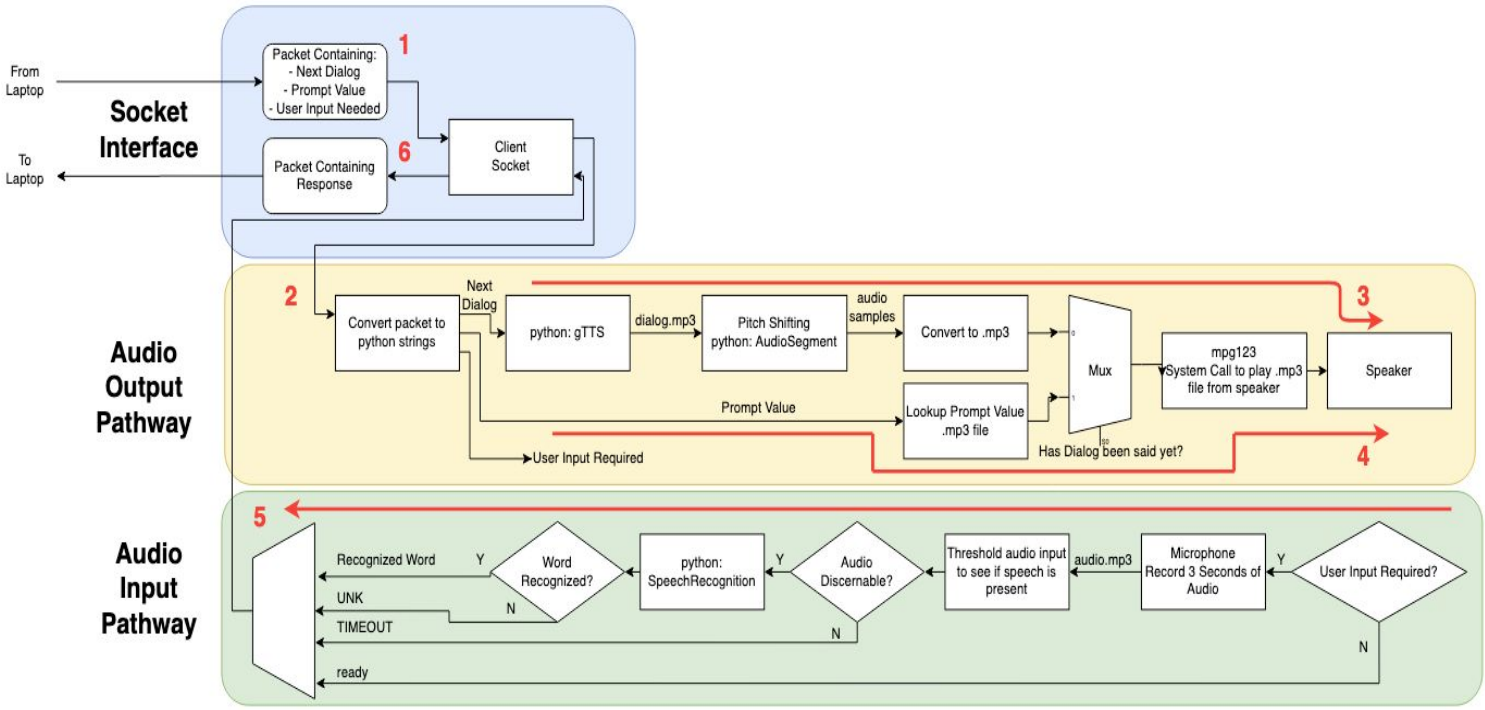

Fig. 7. Diagram of the audio subsystem within KATbot. A single iteration of the user interacting would behave as follows. 1. The Raspberry Pi receives a packet from the laptop containing strings with the next line of dialog, a prompt value, and whether user input is needed from the socket interface. 2. The packet is passed into the audio output pathway. 3. The next line of dialog string is processed using functions from a python package called gTTS which performs text to speech on the input file to generate a .mp3 file of the input text. The .mp3 file is pitch shifted upward using python's AudioSegment package and then played out through the speakers via a system call to mpg123. 4. In order to save time, all prompts' text to speech files are generated in the setup function on the Raspberry Pi. The prompt value is used to look up and find the proper .mp3 file which is then played through the speakers. 5. In the event that user output is not required, it returns a ready signal to the laptop. In the event that user input is required, three seconds of audio is recorded from the microphone. The recording is thresholded to see if speech is present. If speech is not present TIMEOUT is returned. If audio is discernible, it is processed using functions from python's SpeechRecognition package. If a word is recognized, it is returned by the audio input pathway; otherwise, UNK is returned. 6. The return value of the audio input pathway is passed to the socket interface and written as a packet to the laptop.

When the Raspberry Pi in KATbot receives the next line of dialogue from the story generation algorithm, it contains three pieces of information, the next line of dialogue, a user prompt type, and whether user input is required. The next line of dialogue is converted by gTTS to an .mp3 file, the file is then pitch shifted using AudioSegment and played through the speakers using a system call to a program called mpg123. The user prompt value includes various parts of speech and is spoken in the normal gTTS output pitch. During KATbot's setup function we pre-generated the mp3 files for prompts to save time. Sample dialogue includes: "There once was a dog who was <<adjective>>." The items in angle brackets indicate the system speaking in a lower pitch.

B. Pitch Shifting and Voices

One important part of KATbot is its voice, as it is the main user interactive portion of our interface. KATbot has two "voices," a higher pitched voice during narration and a lower pitched voice to cue the user for an input. gTTS synthesizes text to an adult female voice, so we used the original gTTS output for the cueing and pitch shifted the voice upward during narration.

To generate the narration voice we pitch shifted the TTS output upward by 2 semitones. We arrived at this amount through shifting the TTS voice over a range of pitch shifts and choosing the most appealing voice.

Of the three options for pitch shifting, we decided to go with pydub's AudioSegment package. We chose this method to save on time and put the rest of our time towards integrating each of the individual parts of KATbot.

C. Speech Recognition

We used python's SpeechRecognition package with Google's speech recognition API as the speech recognition engine. The API has commands that will listen through the system's microphone, automatically recognize speech, and end its recognition when it hears silence. The API also has functions that perform speech recognition on audio files.

We decided to implement speech recognition by recording the user for three seconds after they are prompted and then passing the recording of the user to SpeechRecognition. This

method was advantageous over having SpeechRecognition access the microphone itself as it prevented the package's functions from hanging or ending abruptly. Ultimately, recording the user for some duration and passing the file to SpeechRecognition yielded a much more fluid user experience.

KATbot uses two different packages to record the user depending on what system the robot code is running on. If the program is running on the Raspberry Pi we used arecord as it was a fast linux system call that was easy to run. However, in the case that the system that the robot code was running on didn't have arecord on it, then we defaulted to using pyAudio, a python package that does recording.

Should recording fail in any way, KATbot will fall back on SpeechRecognition's function that listens through the system's microphone.

The speech recognition output consists of three possible outputs. The first type of output is a user timeout. Timeouts are reported in the case that the user did not say anything within the recording duration. Timeouts are determined by taking the max amplitude of the recording and comparing it with a threshold determined through testing. If the max amplitude is below the threshold then it is likely that the user had not said anything. The other two outputs from speech recognition include the word recognized by the speech recognition package and 'unknown' in the event that the word could not be recognized.

Once speech recognition occurs, its output is passed to socket code to be sent to the storytelling algorithm.

D. Embedded Processor

The robot houses a Raspberry Pi 4, which is the main processor that communicates with the peripherals and handles the speech processing, text to speech, and user-facing display algorithms. With four USB ports, a 40 pin GPIO header, 2 HDMI ports, and a four-pole audio port, it has the necessary ports needed for all the peripherals in this project. In addition, with a quad core Cortex A72 CPU and 4GB of RAM, it has sufficient processing power for the necessary algorithms.

The Raspberry Pi connects with the microphone via USB, and powers the speakers through USB. The speaker signal comes from the four-pole audio port. The Raspberry Pi's two micro HDMI outputs are used to drive a face display and a text display on the robot.

The Raspberry Pi is used to run all of KATbot's I/O software. It has two threads, a thread that runs display code and a thread that runs audio input and output.

The display code was implemented using a package called Kivy, which is a python package used to develop applications. The display code updates the display every time a new input comes in from the ML algorithm to the Raspberry Pi. The display shows the current line of dialogue being said by the robot, and, at the end of the storytelling process, displays the full user generated story with user and algorithm generated inputs highlighted.

The thread that runs the audio input handles receiving packets from the storytelling algorithm, and parses out the next line of dialogue, prompt value, and whether a user input is expected. The thread then passes the next line of dialogue to the display, and runs TTS to say the next line of dialogue and prompt value. If a user input is expected, it runs speech recognition code and passes the user output back to the storytelling algorithm.

E. Raspberry Pi and Laptop Communication

The Raspberry Pi and laptop communicate wirelessly through python's sockets interface. For KATbot, we had the laptop act as a server and the Raspberry Pi act as the client. The process for communication involves setting the laptop running the storytelling algorithm as the host for the Raspberry Pi to connect to, and then setting the port number in both the laptop and Raspberry Pi code to be the same port. The storytelling algorithm is first started up and creates a socket with the laptop's IP and a port number. The storytelling algorithm then prints that it is ready to receive a connection and stalls until the Raspberry Pi connects to it.

The data that is passed from the storytelling program to the Raspberry Pi includes, as mentioned previously, the next line of dialogue, the prompt type, and whether user input is expected. This information is sent from the laptop to the Pi in a single packet using a socket send function. Once the Pi has finished its processing it sends back speech recognition output or a ready signal in the case that user input was not expected. Note that the speech recognition output can be one of a timeout error, an unknown word error, or the word recognized through speech recognition. The storytelling algorithm takes the Raspberry Pi inputs and uses them to decide what the next line of dialogue should be.

F. Robot Body

The custom-made robot initially was planned to have a laser-cut acrylic frame with a 3D-printed and/or cloth shell. Due to limited resources because of the COVID-19 pandemic, we built the robot with cardboard instead. It houses the processor, peripherals, and batteries. To house all of these components, the robot body's dimensions are 8 inches by 8 inches by 10 inches and the robot's head's dimensions are 6 inches by 6 inches by 9 inches.

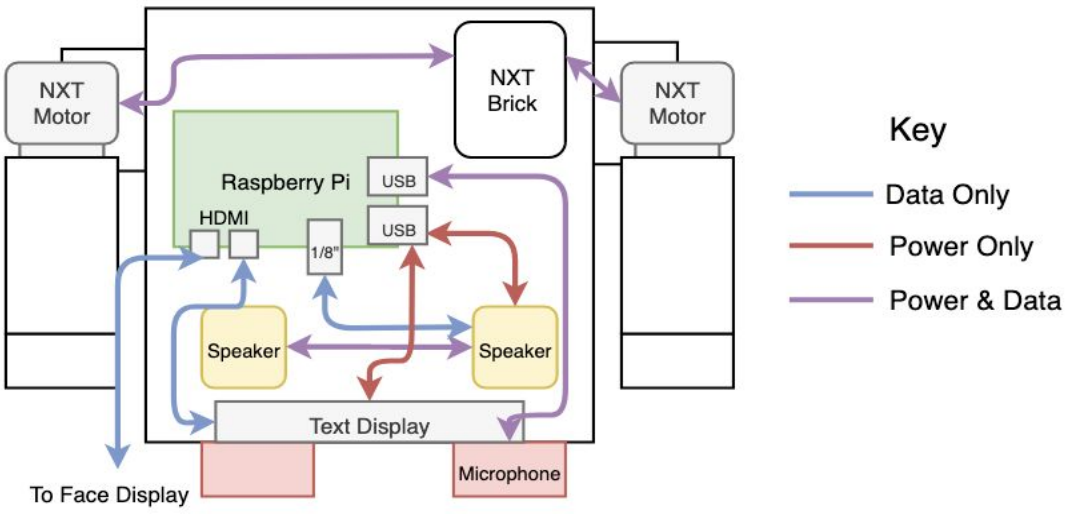

Fig 8. Top down view of the custom robot to show how the hardware is housed within the body. The speakers are powered by a USB port on the Raspberry Pi, and receive signal from the ⅛" port on the Pi. Both displays on the robot are powered by USB ports on the Raspberry Pi and receive data from the HDMI ports. The microphone receives power and passes data to the Raspberry Pi via USB. The motors for KATbot's arms are powered and receive data from a NXT Brick. The NXT Brick, which provides data and power to the NXT motors are not integrated with the Raspberry Pi code.

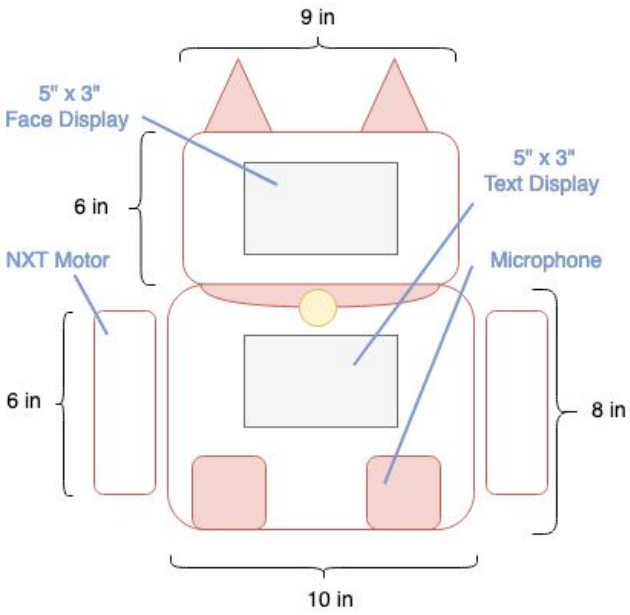

Fig. 9. Dimensions and external view of the custom-made robot. The robot has two displays, one for displaying facial expressions, and the other to display text for the user to read as they interact with KATbot. The arms are driven by NXT motors (see Fig. 10 & 11) and the Microphone is disguised as a "paw" on the robot. The other paw was made with clay and matches the shape of the microphone.

Fig. 10. Completed Custom-made Robot. The robot is constructed out of cardboard, and other household items due to lack of access to 3D-printing and laser cutting during remote instruction.

G. Robotic arms

The robot has two one-degree-of-freedom arms. Each arm is about 6 inches long. To meet the torque requirement of 1.18 kg/cm for the motors for the robot arms, we were originally going to use servo motors, since standard servo motors provide a torque of 4.4 kg/cm at 4.8 volts. Since we could not receive servo motors during the semester due to shipping delays because of COVID-19, we used Lego Mindstorms motors which have a torque of 1.5 kg/cm.

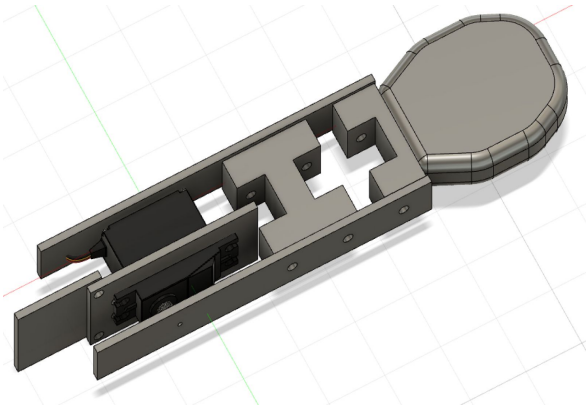

Fig. 11. CAD of the Original Robot Arm Frame

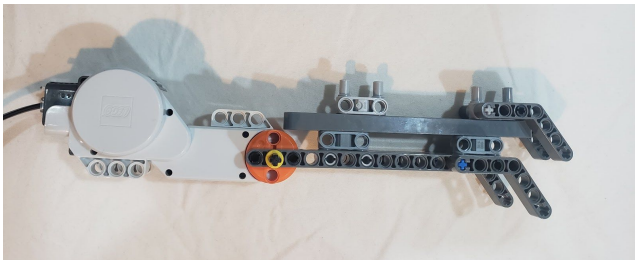

Fig. 12. Current Robot Arm Frame

Fig. 13. Current Robot Arm with Shell

H. Peripherals

1. Microphone

KATbot's audio input comes from a conference table microphone with omnidirectional pickup pattern and a 11.5 foot pickup distance. We have decided on this microphone because its pickup range is roughly what we expect for how far users will be from KATbot when interacting with it. The omnidirectional aspect of the microphone pickup also helps if the user decides to move around while interacting with the robot. The microphone is mounted on the outside of KATbot and masquerades as a foot on the robot. It was placed on the outside of the robot so that it could best pick up audio inputs.

2. Speakers

For our speakers we decided to use a set of speakers from a previous 18-500 group. They have a small form factor and sound good enough to work for our purposes.

3. Text and Eyes Displays

The display for the text is on the body of the robot. It is 3 inches by 5 inches. The robot has another 3 in. by 5 in. display for the face. Both displays are connected to the Raspberry Pi via HDMI.

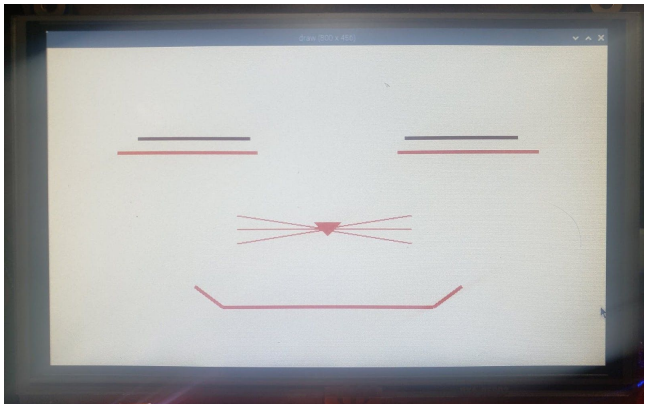

Fig. 14. Example of "Happy" Face on KATbot's face display.

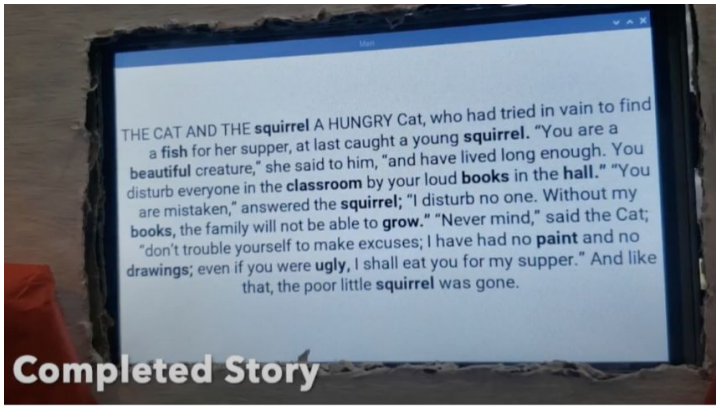

Fig. 15. Example of Completed Story on the Text Display. Bolded words are either user inputs (after grammar correction) or FitBERT inputs

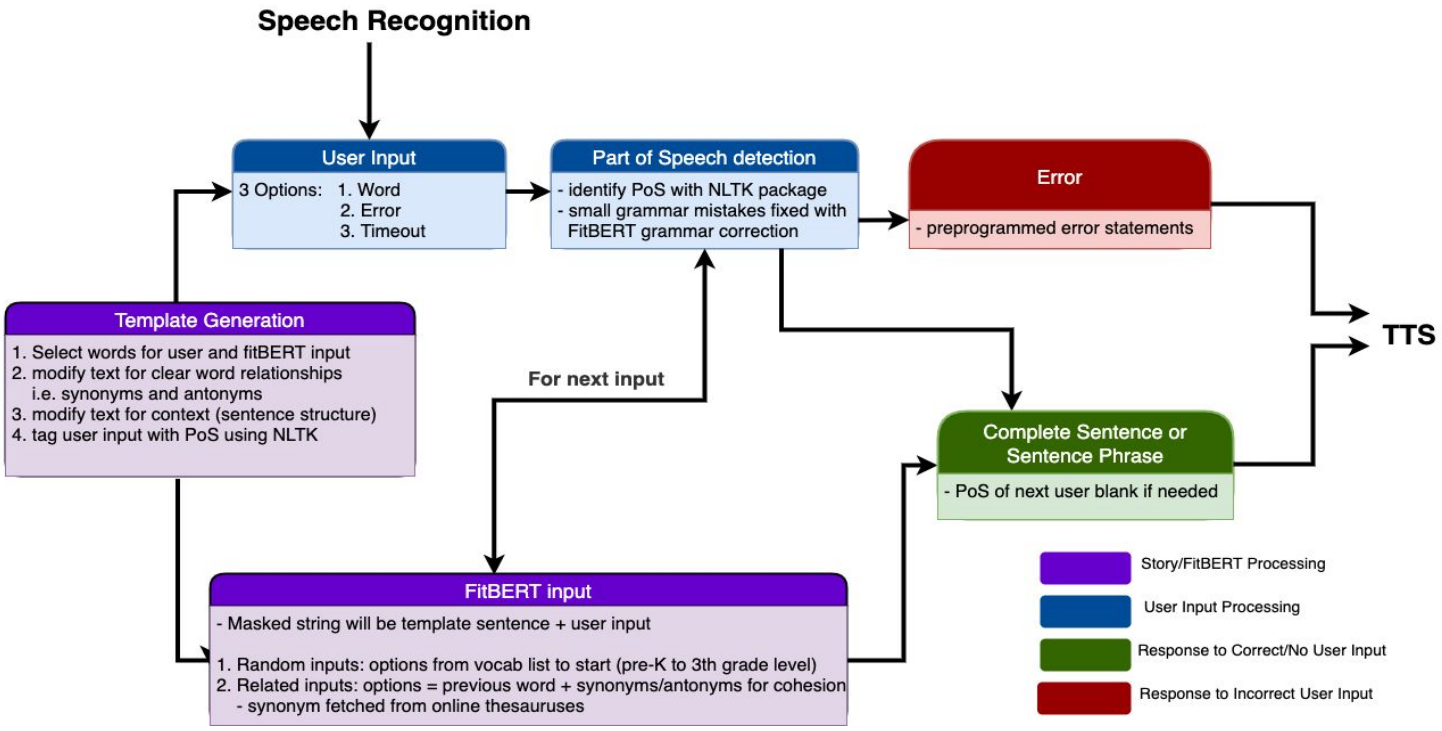

Fig. 16. Storytelling Generation Model. The template is generated prior to the start of the program. As the story continues, new sentences are generated continuously, starting from the template and ending with the sentence or sentence phrase for KATbot to say, and the process is only stalled when waiting for user input. User input is passed in from the speech recognition module, and the sentence output is passed to the TTS module.

H. Story Generation Algorithm

The storytelling algorithm that KATbot uses has three main components. The first is generating the barebone templates for the stories. By using templates, we can have more control over cohesion and a fixed story length since we are building off of actual short stories. To customize the story, we remove all the keywords that drive the narrative and fill them in as we go instead. The next component is the user input, which the algorithm receives word by word from the speech recognition module. The last component is word prediction, which builds off of the user input to complete the sentences and tailor the story.

1. Template Generation

The templates are based on Aesop's Fables. We have a collection of 177 fables that are each about 5-7 sentences long. The vocabulary is approximately at a preK-3rd grade level. For the scope of this project, the robot matches this level by using a similar vocabulary list for filling in the blank and customizing the story. All template generation was done manually, and the following protocol makes it easy to add new templates to the list. Creating the templates has three main steps:

a. Marking words for user input and FitBERT input. There should be at most one of each input type for each sentence or clause. User input blanks should be prompted by some context, so the user will have some basis in choosing words. FitBERT input blanks should have some relationship with the user input, so that the user can feel like they are in charge of the story while maintaining cohesion.

b. Rewording text, if needed, for more clear word relationships. This includes changing parts of speech and matching phrases throughout the story for more cohesion. This step helps FitBERT make more educated decisions when filling in the blanks.

c. Restructuring text for more context. For optimal performance, any algorithm or user input should be near the end of the sentence so there is more context for choosing a word. For cases where the key words are concentrated at the beginning, the sentence structure should be rearranged.

2. FitBERT input

FitBERT is the open-source, fill-in-the-blanks version of BERT (Bidirectional Encoder Representations from Transformers). It works by taking in a list of words and a sentence with a blank and ranking the words in order of best fit into the sentence. Then, it outputs the full sentence with the word inserted. For KATbot's algorithm, the input is the templated sentence or sentence phrase with any user inputs already entered (i.e. one blank left). There are two cases of FitBERT inputs. First, the input is not dependent on user input

and is random to set the tone of the story (for story variety). In this case, the list of words is the entire corpus matching that part of speech, taken from a preK to 3rd grade vocabulary list. Second, the input is dependent on user input and tries to promote cohesion. In this case, the list is either the user input and a select number of synonyms or a list of antonyms, based on what kind of input the word blank needs.

Original	Template
A Nightingale sitting on the top of an oak, singing her evening song, was spied by a hungry Hawk, who swooped down and seized her. The frightened Nightingale prayed the Hawk to let her go. “If you are hungry,” said she, “why not catch some large bird? I am not big enough for even a luncheon.” “Do you happen to see many large birds flying about?” said the Hawk. “You are the only bird I have seen to-day, and I should be foolish indeed to let you go for the sake of larger birds that are not in sight. A morsel is better than nothing.”	A Nightingale sitting on the top of an **USER-R** , singing her **FB-R** song, was spied by a Hawk that was **USER-R**, who swooped down and seized her. “If you are hungry,” said the Nightingale, “why not catch some **USER-1** bird? I am not **FB-1** enough for even a luncheon.” “Do you happen to see many **FB-1** birds **USER-R** about?” said the Hawk. “You are the only bird I have seen today, and if I let you go for the sake of **FB-1** birds that are not in sight, I would be **USER-R**. A **FB-1A** bird is better than nothing.”

Fig. 17. Example template from “The Hawk and the Nightingale”. Each input is either ‘USER’ or ‘FB’ (FitBERT) and the relationships between words are indicated with numbers or ‘R’ for standalone words. A number followed by ‘A’ indicates antonyms, and just a number indicates a synonym or the same word.

3. User Input

The start of the sentence is spoken by KATbot, and the missing input is denoted with a change in voice and its part of speech. If the user needs more clarification of the part of speech, there is a command phrase, “Help me”, indicated in the directions that the user can use for more information. Once the user says a word, the word is compared to the desired part of speech. Small grammatical mistakes are forgiven, such as singular vs. plural or a mistake in verb conjugation. These mistakes are handled with FitBERT’s grammar correction tool. If there is no match in part of speech, KATbot outputs an error statement and gives the user another chance (up to 3 tries). Once all inputs, user and FitBERT, have been provided, the full sentence is delivered to the text to speech module and KATbot finishes saying the sentence. Upon completion of the story KATbot displays the story in full, with all user and FitBert words bolded, and asks the user if they would like it to tell another story or end the session. The user experience looks like Figure 18.

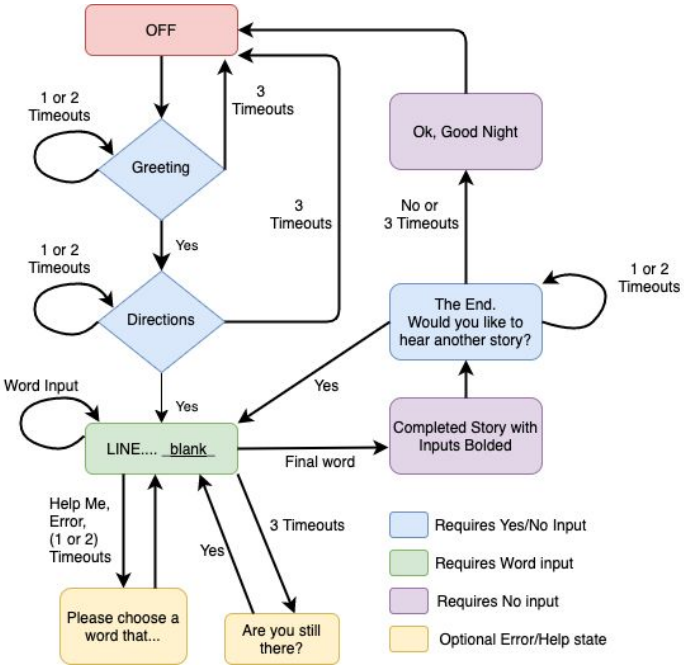

Fig. 18. User Flow Chart. The boxes represent the dialogue that KATbot says, and the transitions represent the user responses that drive the state machine.

The start of the sentence is spoken by KATbot, and the missing input is denoted with a change in voice and its part of speech. If the user needs more clarification of the part of speech, there is a command phrase, “Help me”, indicated in the directions that the user can use for more information. Once the user says a word, the word is compared to the desired part of speech. Small grammatical mistakes are forgiven, such as singular vs. plural or a mistake in verb conjugation. These mistakes are handled with FitBERT’s grammar correction tool. If there is no match in part of speech, KATbot outputs an error statement and gives the user another chance (up to 3 tries). Once all inputs, user and FitBERT, have been provided, the full sentence is delivered to the text to speech module and KATbot finishes saying the sentence. Upon completion of the story KATbot displays the story in full, with all user and FitBert words bolded, and asks the user if they would like it to tell another story or end the session. The user experience looks like Figure 18.

VI. Validation Plan

A. Component Testing

1. Machine Learning

To test the accuracy of the part of speech error detection, we took a vocabulary list of 1038 words from a preK to 3rd grade level. For each word, we got the top two most commonly used parts of speech from dictionary.com, along with one false part of speech, if there is one, and sent it to the detection algorithm to see if it would return correct or

incorrect as expected. We paired each word with a random sentence from our templates that had a blank with the same part of speech because the part of speech tagging system takes sentence context as input as well. The results indicated a 93.82% accuracy, passing our requirement.

For synonym generation, we originally planned to test the accuracy level similarly, with a random test dataset of words from the preK to 3rd grade vocabulary list. We were going to judge accuracy by cross listing the generated synonyms and antonyms with those from an online thesaurus. However, as mentioned in the trade studies section, we changed our design to fetch the synonyms and antonyms directly from online thesauruses. This makes this metric obsolete.

2. Speech Recognition Accuracy

KATbot receives single word inputs from the user, so, in order to test speech recognition we created a test which probes the user to say a specific word as input, runs our speech recognition function, and then compares the inputs. Each test consisted of 100 words from a list of 1000 of the most common words in English. The list of words contains a mix of adjectives, nouns, and verbs which is what most of KATbot's inputs are.

We had two users test the speech recognition system with 10 trials of 100 words each between both of them. The tests were performed using the Raspberry Pi and microphone that KATbot uses. Below are the test results.

Table 2. Results of the speech recognition system test. Each trial was run by two users each saying 100 words. The second column represents the percentage of words correctly recognized, and the third column represents the number of words with homophones that were included in the trial.

Trial Number	Percentage Correct	Homophones
1	81%	0
2	91%	0
3	89%	6
4	80%	0
5	91%	4
6	91%	4
7	87%	8
8	88%	6
9	88%	8
10	91%	6
Average	87.7%	6

The average accuracy was 87.7 % over 10 trials of 100 words each. For some trials output files were made containing the speech decoding errors. For these trials the number of homophones (words that have the same pronunciation but different spellings) were recorded. An example of this from our testing includes the prompt word being 'share' and the speech recognizer recognizing 'Cher.' The average number of homophones per test was 6 homophones. Technically, these words were recognized correctly from their sounds, however we did not count them towards our average accuracy because they weren't what we were expecting.

3. User Interface

Since this product is aimed for children, it is extremely important to have a clean and engaging user interface. The user interface includes the robot's aesthetics, robotic arms, the displays, and the audio output. All of these factors tie in together to create a friendly and interactive robot. We validated this with the user satisfaction survey, which is described below.

B. System Testing

1. System Latency

We tested our system's latency by recording five separate storytelling sessions from start to finish which included user errors, timeouts, and unknown words. We then used Audacity, an audio processing software, to measure exact latencies by looking at the audio waveforms. We measured latency in this way because while it was possible to measure latencies within functions that we wrote, it was hard to measure latencies incurred by functions from python packages that we were using.

There were two latency measurements that were of interest to us. The first latency measurement was to measure user latency, or the time between when the user stopped speaking to when the next line of dialogue was spoken. This latency includes the time needed for speech recognition, communication, and text to speech pre-processing. The second latency measurement we made was non-user latency, which was measured from when a line of dialogue was spoken to when the next line of dialogue was spoken. This latency measures the latency of the communications system between the laptop and pi as well as pre-processing for text to speech. We decided to measure both latencies because we noticed that, depending on whether the system took a user input or not, latency times fluctuated.

We measured latency across five separate trials. Below is an example of the measurements from one trial.

We found that user latency was often much higher than non-user latency, and measured them separately.
Below are some of our findings about user and non-user latency.

Table 3. Measured latencies from one trial. "user" indicates that user input was required, and the latency is the time between the user finishes speaking and KATbot starts speaking."non" indicates that no user input was required, and the latency is the time between KATbot says two consecutive sentences.

Type of Sentence	Latency (sec)
user	5.326
user	4.422
non	2.283
user	4.803
non	2.045
… 12 more trials ...	
Average User Latency:	**4.811625**
Average Non User Latency:	**2.425375**
Average Latency	**3.549705882**

Table 4. Statistics after five trials of measuring latencies between sentences.

Type	Min (sec)	Max (sec)	Average (sec)
User	1.981	7.582	2.5795
Non User	3.658	9.273	4.840
All Latencies	1.981	9.273	3.849

From this we can see that the average user and non-user latencies are both within the 4-6 second range that we wanted. In addition, an average overall latency measurement shows that we have a total average latency of 3.849 seconds which is within our goal range. From this we can also see that user latency was much higher than non-user latency, which indicates that speech recognition took a non-negligible portion of time. Finally, while the average latencies were within our goals, max user and non-user latencies were 7.58 and 9.27 seconds respectively. So, KATbot on average performs within our latency goal, however, individual interactions may take longer than our latency goal.

3. Story Cohesion

Story cohesion was tested through user surveys. We gathered three types of stories: the original stories, user-generated stories, and random stories. The user-generated stories were created by friends and family members (due to limited accessibility), where each user created 2-3 stories. The random stories were created by filling in each of the user and FitBert blanks in the templates with random inputs from a preK to 3rd grade vocabulary level, still matching the appropriate parts of speech. The stories were then given to different family members to score in a blind study. There are five variables related to narrative cohesion that we will look for during this evaluation: logical sense, themes, genre, narrator, and style [18]. Each story was graded on a scale of 0-10 for each of the five variables, for a total score out of 50. The goal was to achieve a story cohesion score for the user generated stories between that of the random input stories and the original stories.

Table 5. Story Cohesion Scores based on User Surveys. Each variable was scored out of 10, and the total score is out of 50.

Variable	Random Input Stories	**User Generated Stories**	Original Stories
Logical Sense	1.6	**4.0**	8.4
Theme	5.0	**6.13**	8.4
Genre	5.4	**6.0**	8.4
Narrator	5.2	**6.47**	8.4
Style	5.0	**6.4**	8.4
Total	22.2	**29.0**	42.0

As shown above, the story cohesion passed our goal, scoring consistently higher than the random input stories for each variable and in total.

4. User Satisfaction

An important part of our project is user satisfaction. Our project was inspired by an MIT robotics group's interactive storytelling robot so we wanted to be able to match their metrics. They evaluated their robot on several different characteristics. This included 87.5% of users liking the stories, 100% wanting to play again, 87.5% believing that the robot was friendly, 87.5% believing the robot's stories were interesting, and 100% of the users rating the robot's stories as understandable. We tested user satisfaction with 3 users who were quarantined with the team member who had built the robot. They rated the five criterias on a scale of 1-7 where 1 represented Strongly Disagree and 7 represented Strongly Agree. Although we did not meet all of the criterias, we were within 15% of each metric. The results can be seen on Table 6.

5. Reliability.

Based on research of children's attention spans, we would like the product to be able to be played with for 30-45 minutes without any reboot, loss of connection, or loss of power issues.

To test this, we played with the robot for 30-45 minutes at a time over 3 trials. In each trial, the robot had no issues with reboot, loss of connection, or loss of power. The results can be seen on Table 7.

Table 6. User Satisfaction Scores based on User Surveys. Each variable was ranked on a scale of 1-7 where 1 represented Strongly Disagree and 7 represented Strongly Agree.

Variable	MIT robot	**KATbot**
Liked the stories	87.5%	**90.5%**
Wanted to play again	100%	**90.5%**
Robot was friendly	87.5%	**76.2%**
Stories were interesting	87.5%	**90.5%**
Stories were understandable	100%	**85.7%**

Table 7. Reliability Testing Results. The robot was tested for 3 aspects of reliability over 3 trials with a user playing with the robot. For all 3 trials, the robot passed all 3 aspects of reliability measured.

Variable	Trial 1	Trial 2	Trial 3
Did the robot reboot?	No	No	No
Did the robot lose connection with the laptop?	No	No	No
Did the robot lose power?	No	No	No

VI. FUTURE WORK

A. Professional Feedback

We talked with Dr. Henny Admoni, a professor and researcher in Human-Robot Interaction at Carnegie Mellon University. We described our project, gave her a live demo, and received feedback on the viability and usability of this product.

She mentioned that it is important to consider the large gap in knowledge and learning between children at age 5 and at age 8 (the range that we are targeting). For example, it might not be beneficial to teach parts of speech to children at age 5, while parts of speech is probably very important to teach children at age 8. She thought that it would be useful to toggle different features, such as teach parts of speech, so that teachers and parents can customize the robot for the age of the children.

Dr. Admoni also said that she has seen that children tend to touch their toys a lot so it is important to ensure that everything is durable and that the inner electronics can not be accessed easily.

The robot also repeats the last sentence after receiving an user input to aid in story cohesion. Dr. Admoni thought that it might be more efficient to repeat the last verb or noun phrase instead of the entire sentence.

Finally, Dr. Admoni said that speech recognition can be hard with children, and it might be more practical to replace or supplement spoken user input with pictures instead.

B. User Feedback

We also asked the users who participated in our User Satisfaction survey for feedback. They mentioned that it would be beneficial to give a visual indicator of when the robot was looking for user input, such as a red “recording” circle on the display, to give the user a more intuitive understanding of when to speak. In addition, they mentioned that the robot’s voice should be slowed down a little so that they didn’t rely as heavily on the text display to understand the story.

C. Future Work

The feedback that we received from users and from a professional was crucial in helping to understand how to improve our product, especially to a state where we could deploy it.

Future iterations of this product would involve constructing a robot frame made with more durable materials such as laser-cut acrylic and 3d-printed filament. It would also involve a customizable story generation format that could be adapted for children at different ages and/or different learning levels. We would also include visual indicators and pictures on the display to give the users a more friendly and intuitive experience.

VI. PROJECT MANAGEMENT

A. Schedule

Figure 19 below is the Gantt chart of our project. It includes a schedule with the division of tasks per person, pair, and over the whole team for the whole semester.

B. Team Member Responsibilities

Ashika was in charge of story creation and constructed all the story generation algorithms. Jade was responsible for the speech processing aspect and handled both collecting audio input and the text-to-speech capabilities. Abha built the physical robot, including integrating the peripherals and working on the robot arms.

Jade and Ashika worked on integrating the speech processing and story creation modules together, Abha and Ashika worked on displaying the sentences with the robot's display, and Jade and Abha worked on text to speech with the speakers in the robot. Everyone worked on getting the socket code to work with the Raspberry Pi in the robot. Everyone worked on testing their individual components as they went, and everyone worked together to test and evaluate the whole system.

C. Budget

Figure 20 below is the bill of materials for KATbot. Overall, our spending consists mostly of buying peripherals for the robot itself as well as raw materials to design the robot. With the transition to remote instructure, we also ordered duplicates of some parts due to team members being in different locations.

VII. RELATED WORK

A. MIT Storytelling Robot

KATbot's initial inspiration came from the MIT storytelling robot, but its key features and system architecture are quite different. The MIT robot has two modes: storytelling and listening, and KATbot is based on the storytelling mode. The MIT robot creates stories based on the images that a child can move around on a tablet. This is a fairly limited user experience and involves image processing rather than speech processing. KATbot's user experience begins with a little more structure because it starts with story templates, but the user is not restricted to any limited word set as input.

The MIT robot has a face display and a toy-like outer appearance. It makes small movements, primarily with its head, to mimic natural movement. KATbot's exterior is similar, with moving arms and an eye display to add gestures and emotions to the storytelling feature. However, KATbot also has a display to display the sentences as it speaks. This difference is important because KATbot does not rely on a tablet or external technology for the user to read from. This leads to the big advantage that KATbot aims to be a single unit product, with no additional technology required to interact with it.

B. AI Dungeon 2

While storytelling robots are becoming increasingly popular in the AI world, there are few programs out there that have achieved interactive storytelling. The most notable is an AI driven video game that generates narratives based on user input. Similar to a choose-your-own-adventure story, *AI Dungeon 2* (https://play.aidungeon.io/) starts with a user selected setting and character, and then tells the story a few sentences at a time. The user is expected to respond to the question "What will you do?" after each generated output. The story is tailored to this input and changes direction based on what the user chooses to do. Here is an example interaction with the game:

Input: *look for water*
Output: *You search through the cupboards until you find a bottle of water. You drink half of it and immediately feel thirsty again.*

According to documentation, this program was created by training a machine learning model on a collection of choose-your-own-adventure stories [13]. It performs well with grammatical accuracy, with only a few observed semantic mistakes with pronouns. However, it takes in entire sentences from the user, not words like KATbot, and while it does respond to the user input, the advantage of being a video game allows it to only factor in the last few inputs, not the entire narrative. Additionally, the game does not always carry the story; this task falls on the user, and a lot of the generated outputs end without much prompt for the user to go on. While this suits the goal of a game, this is quite different from the goals of KATbot, which has a lot more control over the story because it aims to create full stories with a beginning, middle, and end. Overall, *AI Dungeon 2* is a great model for processing and responding to the semantic meanings of user input, but the end goals differ drastically from KATbot.

VIII. SUMMARY

KATbot exceeded our performance expectations, especially given the circumstances. On the storytelling side, we could improve the part of speech detection even further with more research into taggers or other tools. We could also parallelize the system further to avoid long latencies while the story is updating. On the audio side, we could implement a robust speech recognition system that recognizes when the user stops talking and immediately starts on recognition instead of recording them for a duration and then performing speech recognition. We could also add the option for the user to respond with short phrases instead of single words. For the robot itself, if we had access to resources on campus, we could construct the 3D printed, acrylic robot shell we had initially planned to make. We would also be able to play around with the face display and implement sentiment analysis to integrate the emotions and arm movements with the story.

As we built KATbot, we learned a few key lessons along the way. First, prototyping is always a good idea. Creating small prototypes of the parts we were working on allowed us to flesh out design details and confirm how pieces of the design interfaced. Second, when writing software, good documentation helps your teammates set up the right environments and understand how to run your code. When we were integrating remotely, we spent a lot of time talking through installation or system configuration issues that could have been avoided by keeping a running document of what packages to install and what flags to change from the beginning. Third, having modularity within components of the project was really helpful in terms of parallelizing work. Because the three of us were working on independent components we were able to parallelize a lot of work. While this may be unique to this semester, dividing the work once we went to remote instruction was much easier with mostly independent components. Integration over video calling can be quite restricting and slow, so it was advantageous that we could work on our individual tasks independently.

IX. References

[1] Westlund, J. K. (n.d.). Storytelling Companion. Retrieved from http://robotic.media.mit.edu/portfolio/storytelling-companion/

[2] Fish, M. and Pinkerman, B. 2003. Language skills in lowSES rural Appalachian children: Normative development and individual differences, infancy to preschool. J. Appl. Dev. Psychol., 235, 539-565.

[3] Duranti, A. and Goodwin, C. 1992. Rethinking context: Language as an interactive phenomenon. Cambridge University Press.

[4] Kory, J., & Breazeal, C. (2014). Storytelling with Robots: Learning Companions for Preschool Children's Language Development. In P. A. Vargas & R. Aylett (Eds.),

[5] Jiang, C., Wang, Z., & Yu, J. (2015). An Expressive Eye Model: Using Eye Movement to Show Ocular Emotional Expression. *IFAC-PapersOnLine*, *48*(28), 1456–1461. doi: 10.1016/j.ifacol.2015.12.338

[6] Ruder, S. (n.d.). Part-of-speech tagging. Retrieved from http://nlpprogress.com/english/part-of-speech_tagging.html

[7] Hagiwara, M. 2008. A supervised learning approach to automatic synonym identification based on distributional features. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 1–6, Columbus, Ohio, USA.

[8] Këpuska, V., & Bohouta, G. (2017). Comparing Speech Recognition Systems (Microsoft API, Google API And CMU Sphinx). *International Journal of Engineering Research and Applications*, *07*(03), 20–24. doi: 10.9790/9622-0703022024

[9] Kennedy, J., Lemaignan, S., Montassier, C., Lavalade, P., Irfan, B., Papadopoulos, F., Belpaeme, T. (2017). Child Speech Recognition in Human-Robot Interaction. *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction - HRI 17*. doi: 10.1145/2909824.3020229

[10] Yeung, G., & Alwan, A. (2018). On the Difficulties of Automatic Speech Recognition for Kindergarten-Aged Children. *Interspeech 2018*. doi: 10.21437/interspeech.2018-2297

[11] Children and Age-Appropriate Attention Spans. (2016, September 22). Retrieved from https://www.speechtherapycentres.com/children-and-age-appropriate-attention-spans/

[12] Bull, M., & Aylett, M. (1998). An analysis of the timing of turn-taking in a corpus of goal-oriented dialogue. In *Proceedings of the fifth international conference on spoken language processing* (*ICSLP '98*), Sydney, Australia, (Vol. 4, pp. 1175–1178).

[13] Jeffrey, C. (2019, December 6). AI driven text adventure game give players true non-linear gameplay. Retrieved from https://www.techspot.com/news/83072-ai-driven-text-adventure-game-give-players-true.html

[14] McCoy, N. (2016, October 27). Evaluating NLTK Taggers Tutorial. Retrieved from https://natemccoy.github.io/2016/10/27/evaluatingnltktaggerstutorial.html

[15] Comparison of Python NLP libraries: ActiveWizards: data science and engineering lab. (n.d.). Retrieved from https://activewizards.com/blog/comparison-of-python-nlp-libraries/

[16] Shmyrev, N. (n.d.). Building an application with PocketSphinx. Retrieved from https://cmusphinx.github.io/wiki/tutorialpocketsphinx/

[17] Moulines, E., & Charpentier, F. (1990). Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication*, *9*(5-6), 453–467. doi: 10.1016/0167-6393(90)90021-z

[18] Hargood, Charlie, Millard, David and Weal, Mark (2011) Measuring Narrative Cohesion: A Five Variables Approach. Narrative and Hypertext Workshop at Hypertext 11.

[19] Categorizing and Tagging Words. (n.d.). Retrieved from http://www.nltk.org/book/ch05.html

[20] Reprinted from Digital Processing of Speech Signals pp. 336, by Rabiner and Schafer, 1978, retrieved from http://course.ece.cmu.edu/~ece792/ Copyright 1979 by Pearson

[21] Reprinted from Digital Processing of Speech Signals pp. 335, by Rabiner and Schafer, 1978, retrieved from http://course.ece.cmu.edu/~ece792/ Copyright 1979 by Pearson

[22] Reprinted from Digital Processing of Speech Signals pp. 337, by Rabiner and Schafer, 1978, retrieved from http://course.ece.cmu.edu/~ece792/ Copyright 1979 by Pearson

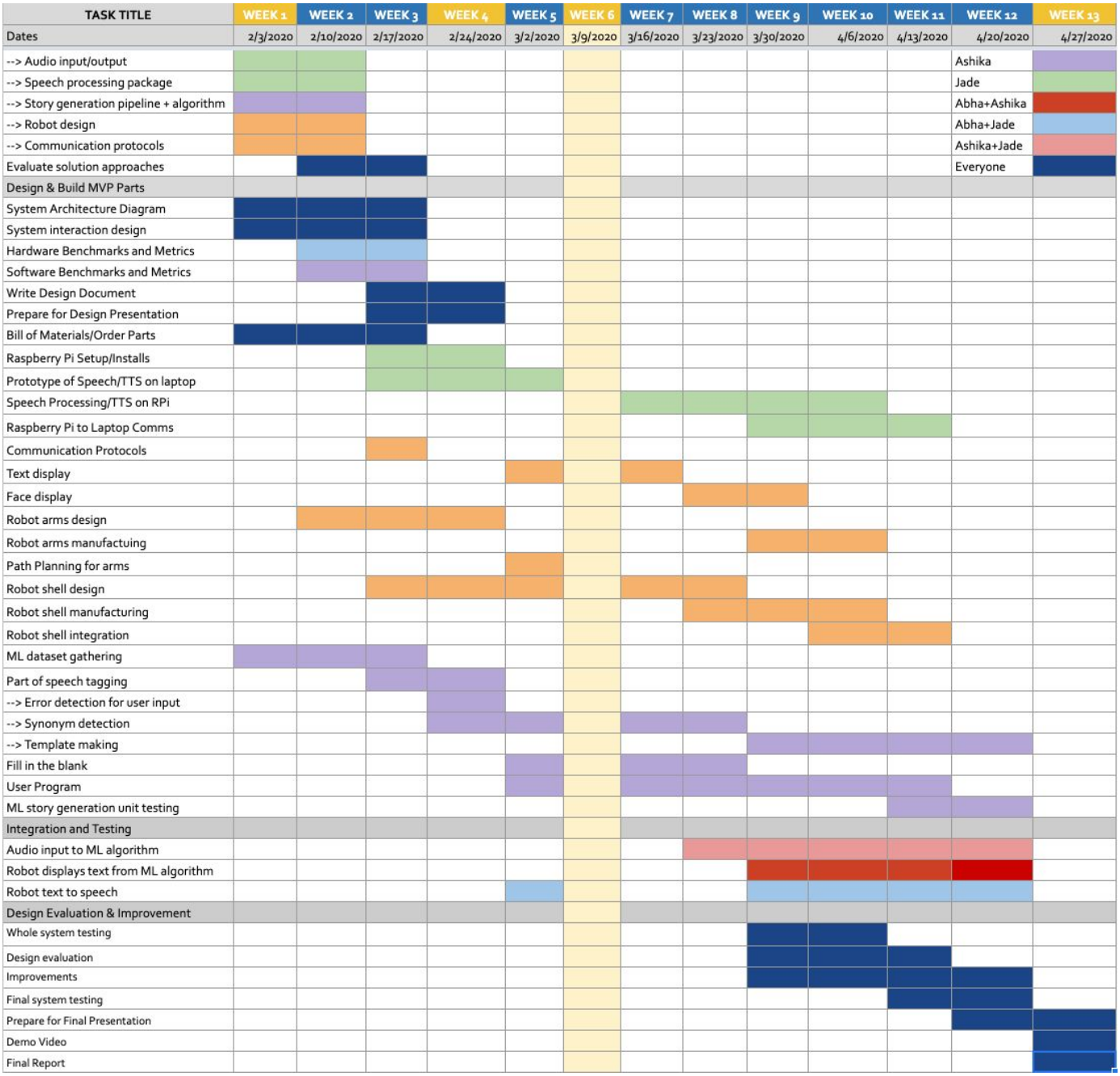

Fig. 19. Gantt Chart

Product	Description/Specs	Quantity	Cost	Total Cost
Raspberry Pi 4	Raspberry Pi 4 Model B - 4 GB RAM	1	55	55
Text Display	Adafruit Industries HDMI 5" 800x480 Display Backpack - with Resistive Touchscreen	1	74.95	74.95
Eye Display	2.0" 320x240 Color IPS TFT Display with microSD Card Breakout	1	49.99	49.99
Microphone	TONOR Conference USB Microphone, Omnidirectional Condenser PC Mic	1	19.98	19.98
Sound Card	External Sound Card USB Audio Adapter, USB Type A to 3.5mm TRS & TRRS Aux Jacks	1	12.99	12.99
Speaker	Earise AL-101 3.5mm Mini Computer Speakers Powered by USB	1	24.56	24.56
Servo Motor	LEGO Mindstorms NXT Motor	2	29.99	59.98
Micro SD Cards	32 GB 2-pack	1	10.99	10.99
Micro HDMI to HDMI Cable		2	5.99	11.98
USB C Cable	2 pack	1	8.29	8.29
Cardboard		1	0	0
Clay		1	19.99	19.99
Paint	White, Red, and Yellow Acrylic Paint	1	8.99	8.99
Hot Glue		1	2.99	2.99
				360.68

Fig. 20. Bill of Materials