# KATbot: 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, microphone, 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. In addition, because 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].

Because of these reasons, we would like to create 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 will be a standalone interactive robot with the following qualities. It will tell customized stories by asking the user to fill in parts of the story as it goes. It will be able to run for 30 - 45 minutes on its own battery power, respond to user input within at most 4 - 6 seconds, have a 15 % word error rate (WER), 90% part of speech accuracy, 85% synonym recall accuracy, cohesive stories, and high user satisfaction. Testing with children will be outside of the scope of what we can accomplish this semester, therefore KATbot will be designed and tested by adult users.

## II. DESIGN REQUIREMENTS

### *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.

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

We will also have two robot arms, each of which are one degree of freedom. The arms will move to help show emotion while the robot tells the story. They will be made with materials easy to prototype with, such as acrylic and PLA filament. Based on the weight of acrylic and PLA (3D-printed material), the motors for the arms will need to have a minimum torque of 2.04 kg/cm. The calculations are below.

*Assume that the arm dimensions are* $6in\ x\ 2in\ x\ 2in$. *The acrylic frame covers a surface area of* $56\ in^2$. *Acrylic has a height of* $0.125in$. *Therefore, the volume is* $7\ in^3 = 114\ cm^3$. *The density of acrylic is* $1.18\ g/cm^3$. *Therefore, the mass is* $114cm^3 * 1.18g/cm = 134.62g$.

*Torque is Fxr, where* $F = 0.134kg$ *and* $r = 6in = 15.24cm$. *Therefore, Torque* $= 2.04\ kg/cm$.

Fig. 1. Torque Calculations for the robot arms

KATbot will also have displays connected to the Raspberry Pis. It will have a display on the body of the robot, that shows the current sentence that the robot is speaking. This is to aid the user with reading comprehension. It will also have two small displays on the face of the robot for the robot's eyes. Eye movements and gestures are incredibly important when expressing emotion [5]. Instead of static eyes that would not show emotion, we chose to use displays for them so that we could dynamically change the emotions that the eyes 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 will need 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 are aiming for a 90% accuracy level, which is around the minimum of popular natural language models [6]. For synonyms, we would like 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 either learning to read or 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 aim 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 will be applied only for adult users.

Testing our system with children will be outside of the scope of this project because we do not have the proper paperwork to test KATbot with children. Because speech recognition for children is an ongoing challenge for the field, we will be designing 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 intend for it to be played with wherever a child wants to. Due to this, KATbot will need 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 require that KATbot should be able to run for 30 - 45 minutes off its own power system.

In order for KATbot to be a standalone toy, it will need 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. All of the batteries will also need to be housed within the body of the robot.

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 dialog between humans is around 4 - 6 seconds [12] Thus, we aim 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.

## III. ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

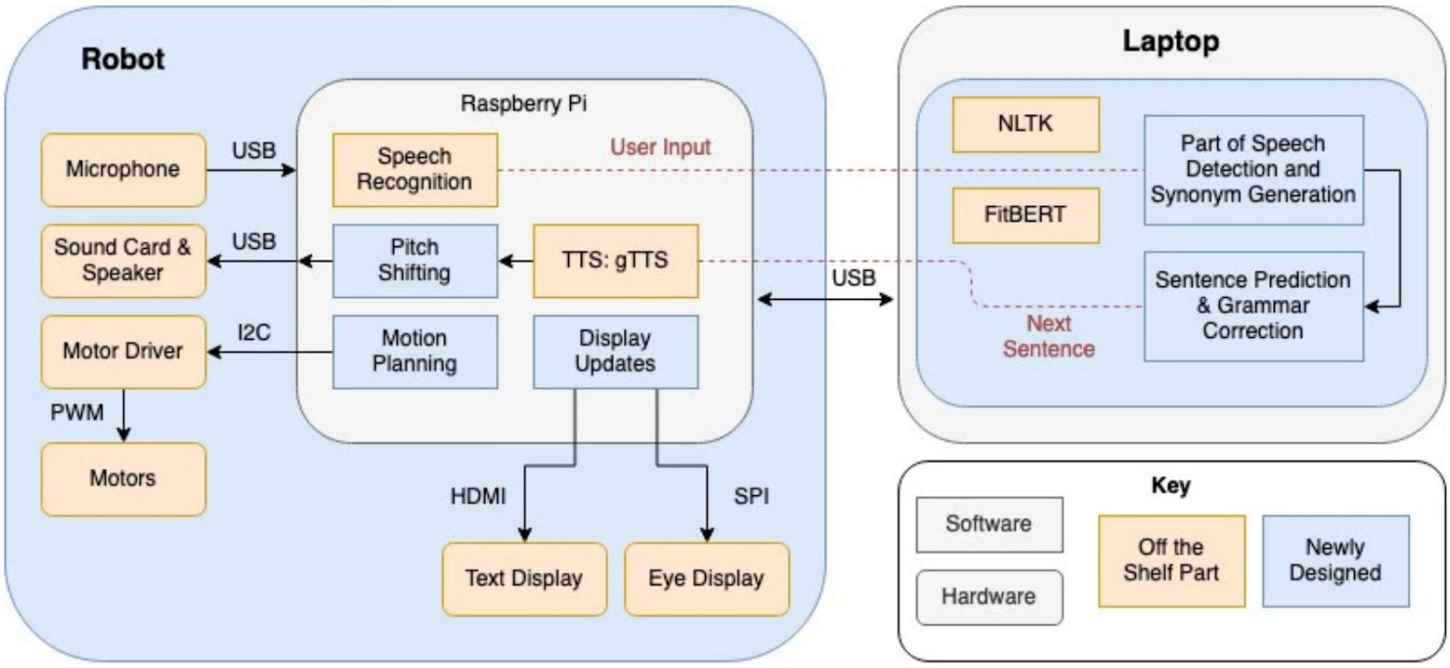


Fig. 2. Overall System Diagram

To meet these requirements, we will be building a standalone (all electronics are housed within the robot) custom-made robot. The robot will house a main processor that will communicate with the peripherals and handles the speech processing, text to speech, robot arm motion planning, and user-facing display algorithms.

The custom-made robot will be made with a sturdy frame that can be handled by children. It will house the processor, peripherals, and batteries. The robot's body and head dimensions will be carefully chosen to be big enough to fit all the electronics within it. The robot will have two one-degree-of-freedom arms. We will use motors with a small form factor, so that they can fit within the robot, and that meet the torque requirement of 2.04 kg/cm.

We will be using 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 are chosen so that the user can move around while interacting with the robot. We will use speakers with a small form factor so that they can fit in the base of the robot.

The processor inside the robot will communicate to a laptop that contains the story generation algorithm. Even though the end goal of this product is to be a standalone toy, we realize that given the time constraints of a semester project, we might not have enough time to not only have a completed story generation algorithm that meets our requirements, but also 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 will start with templates that are manually configured based on a loose set of constraints that ensure context and clear word relations. KATbot will prompt the user for the word of the specified part of speech for the blank, and the user input will go 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 will be ignored (e.g. singular vs. plural). On the algorithm side, any related words will be filled in using synonym and antonym generation 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. 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 will aim 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.

### *C. 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 has 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 (see below), the advantages of using NLTK outweigh the advantages of SpaCy.

2. Synonym/Antonym generation

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

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.

### *D. 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, as well as 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 have 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 will be using the Google Speech Recognition engine. This engine requires a connection to the internet, but performs speech recognition extremely well. The Google Speech API was shown to be the best system for speech recognition in a study investigating speech recognition in human robot interaction for children [8]. We recognize that a risk factor for KATbot is that it might not always have a stable internet connection. To mitigate this risk, we also plan on having a local speech recognition package to fall back on in case connectivity is lost. We plan on also using PocketSphinx, as an engine with SpeechRecognition, in the case of lost internet connectivity, but we chose not to use PocketSphinx as the main speech recognition engine since the Google Speech API is one of the best speech recognition engines.

### *E. 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 offer good voices for the user to interact with. However, AWS Polly requires AWS credits which we are trying to avoid so that the users of KATbot can have a standalone product that will not require them to pay for anything. Of the remaining two packages, google_speech and gTTS are both python packages, however google_speech is less up-to-date than gTTS. We decided to go with gTTS.

gTTS was selected because it offered a good voice with minimal latency in processing. However, it is important to note that gTTS also requires internet connection.

### *F. Pitch Shifting Algorithm*

We will be performing pitch shifting on the voice generated by our text to speech package to make the 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 are considering. The first two involve writing our own pitch shifting software based on existing algorithms, and the last one involves using a python package to perform pitch shifting.

The first algorithm for pitch shifting is 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 is 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.

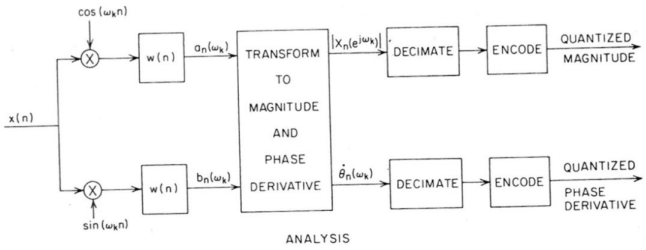


**Fig. 6.54** Complete single channel of a phase vocoder analyzer.

Fig. 3. Complete single channel of a phase vocoder analyzer. 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

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

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

Fig. 4. Discrete tiem instantaneous phase equation. 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

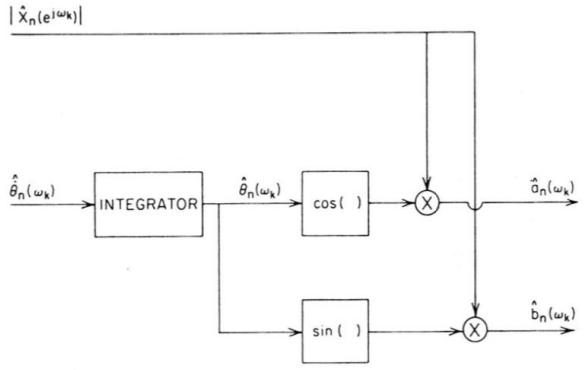


Fig. 5. Reconstruction of the output of a phase vocoder analyzer. Reprinted from Digital Processing of Speech Signals pp. 337, by Rabiner and Schafer, 1978, retrieved from http://course.ece.cmu.edu/~ece792/ Copyright 1979 by Pearson

The third option for pitch shifting involves using a built-in python package called pydub. Pydub 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.

Of the three options, we will be implementing phase vocoding as our pitch shifting algorithm. This is because phase vocoding does not require detailed analysis of the time domain signal. We are also familiar with how phase vocoding works, which will make implementation easier.

### *G. Embedded Processor*

The three embedded processors in the market today that satisfy our requirements are 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. One HDMI port can be used with a display peripheral, and the second for debugging and testing our design. Our processor will need to be able to handle fast computation to meet our latency requirements. More RAM will aid in running the programs more quickly, reducing latency, Upgraded hardware, more ports, more memory, and faster processing and performance speeds cause 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. It is important that we reduce power consumption wherever possible since this product will be battery powered. In addition, at $99, the Nano is almost double the price of the Raspberry Pi 4, which is important to consider under a constrained budget of $600. The reason for this price difference is due to the powerful GPU on the Nano, which is not needed for this project.

### *H. 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 have 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 KATbot. Despite the fact that the table microphone will provide a worse audio signal input to speech recognition it will be able to fit within the robot and can have both omnidirectional pickup and pickup from a distance. This works well for KATbot as it will allow the user to move around while interacting with the robot.

### *I. Displays*

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 eye shape/movement or text or 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 our product 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*

We will be using text to speech as the primary way of communicating with the user. We will be using Google's text to speech package called gTTS. gTTS has a simple API which takes in a string and converts it to an output file. When the Raspberry Pi in KATbot receives the next line of dialogue from the story generation algorithm, it will contain two pieces of information. The first is the dialogue to be said and the second piece is the expected user input. The expected user inputs include various parts of speech and {yes or no}. When KATbot receives the next line of dialogue from the story generation algorithm, KATbot will say it and then cue the user by playing a sound to indicate that they should respond along with an indication for what type of input we require. We will indicate input types by altering the pitch of the KATbot's voice. Sample dialogue might include: "Are you still there **sound cue** <<yes or no>>*?" or "There once was a dog who was **sound cue*** <<noun>>." The items in angle brackets indicate the system speaking in a lower pitch.

### *B. Pitch Shifting and Voices*

One important part of our project is the voice of KATbot. KATbot will have 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 will use the original lower pitched gTTS output for the cueing and pitch shift the voice during narration.

To generate the narration voice we will be pitch shifting 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. We will be writing our own pitch-shifting code using a phase vocoding algorithm which is outlined in the trade studies. Phase vocoding involves operations on discrete signals and is comprised mostly of multiplication of samples by a window, multiplication of samples by cosines and sines, and other mathematical operations that can be implemented via matrix math. We will be using NumPy, a scientific computing package, to perform mathematical operations on the input signal when implementing phase vocoding.

### *C. Speech Recognition*

We will be using python's SpeechRecognition package with Google's speech recognition API as the speech recognition engine. This API has commands that will listen through the system's microphone, automatically recognize speech and end its recognition when it hears silence. KATbot will be "listening" for user speech only after we have probed the user for input as mentioned above. Once the user has given the input, the speech processing algorithm will pass the input to the story generation algorithm which will generate the next line of dialogue.

### *D. Embedded Processor*

The robot will house a Raspberry Pi 4, which will be the main processor that will communicate with the peripherals and handles the speech processing, text to speech, robot arm motion planning, 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.

### *E. Robot Body*

The custom-made robot will have a laser-cut acrylic frame with a 3D-printed and/or cloth shell. It will house the processor, peripherals, and batteries. To house all of these components, the robot body's dimensions will be 8 inches by 8 inches by 10 inches and the robot's head's dimensions will be 6 inches by 6 inches by 9 inches.

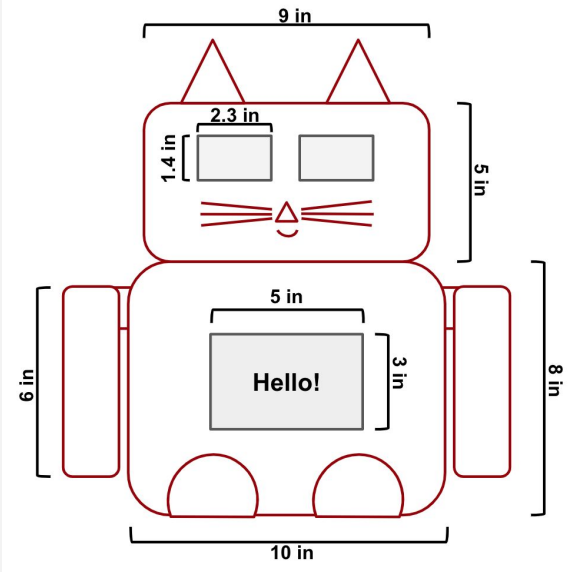


Fig. 6. Dimensions of the Custom-made Robot

### *F. Robotic arms*

The robot will have two one-degree-of-freedom arms. Each arm will be about 6 inches long. To meet the torque requirement of 2.04 kg/cm for the motors for the robot arms, we will use servo motors, since standard servo motors provide a torque of 4.4 kg/cm at 4.8 volts.

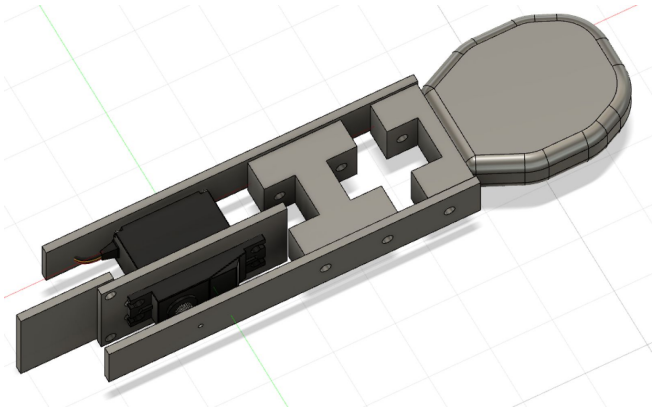

Fig. 7. CAD of the Robot Arm Frame

*G. Peripherals*

1. Microphone

We will use a conference table microphone with omnidirectional pickup pattern and a 11.5 foot pickup distance that will connect to the Raspberry Pi via USB. We have decided on this microphone because we believe that the users of KATbot will be interacting with it at a similar distance to that which a conference microphone can pickup. An omnidirectional microphone also helps if the user decides to move around while interacting with the robot.

2. Speaker

For our speakers we have decided to use a set of speakers from a previous 18-500 group. We have decided to use them because they are small enough to fit in the base of the robot and will work for our purposes. Since they come from a previous 18-500 group, we do not have to purchase speakers for this project which reduces the cost of KATbot.

3. Text and Eyes Displays

The display for the text will be on the body of the robot. It will be 3 inches by 5 inches. The robot will have two 2.3 inches by 1.4 inches displays for the eyes. The text display will be connected to the Raspberry Pi via USB and the eye displays will communicate with the Raspberry Pi using SPI protocol.

*H. Story Generation Algorithm*

The storytelling algorithm that KATbot uses has three main components. The first is generating the barebones 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 will 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 will receive 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.

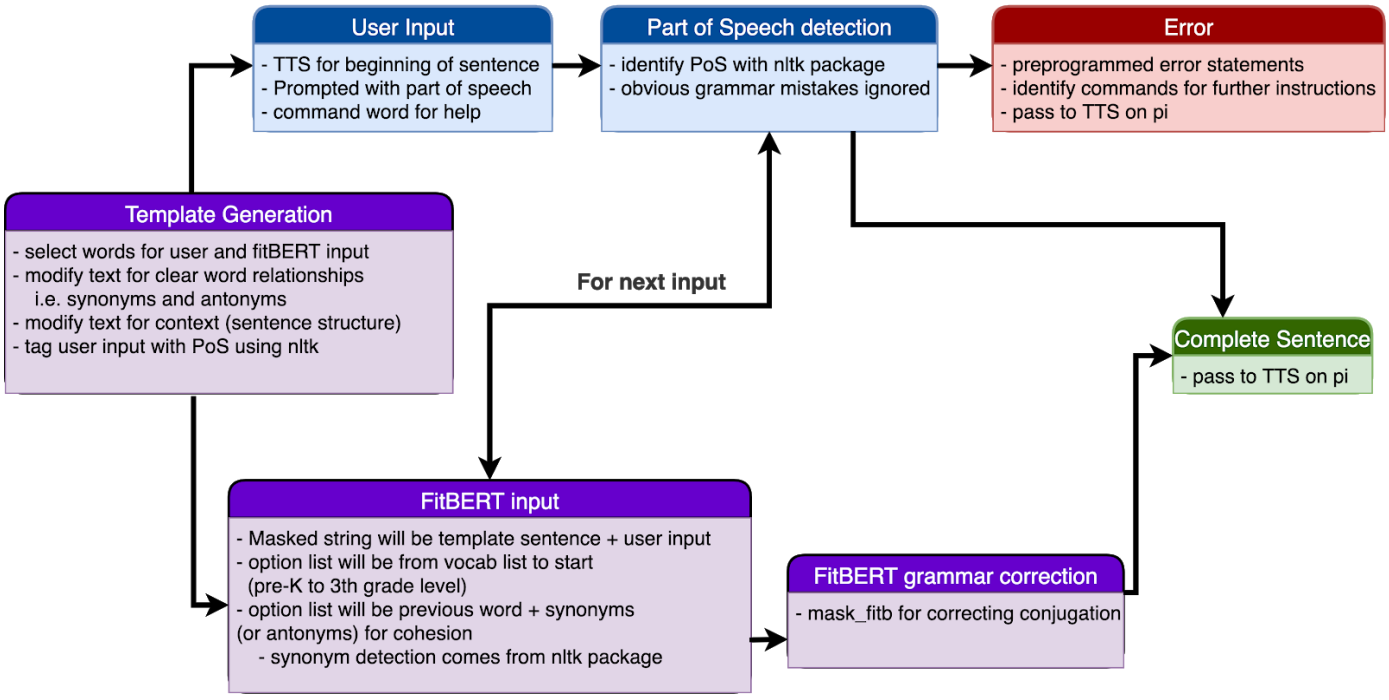


Fig. 8. Storytelling Generation Model

1. Template Generation

The templates will be 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 preK-3rd grade level. For the scope of this project, the robot will match this level by using a similar vocabulary list for filling in the blank and customizing the story. For the minimum viable product (MVP), all template generation will be done manually. 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 for this purpose.

Here is an example of a fable turned into a template:

| 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.<br><br>"If you are hungry," said she, "why not catch some large bird? I am not big enough for even a luncheon."<br><br>"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.<br><br>"If you are hungry," said the Nightingale, "why not catch some **USER-1** bird? I am not<br>**FB-1** enough for even a luncheon."<br><br>"Do you happen to see many **FB-1** birds **USER-R** about?" said the Hawk.<br>"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. 9. 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.

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 will be 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 will be 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 will be either be the

user input and a select number of synonyms or a list of antonyms, based on what kind of input the word blank needs. Once FitBERT has selected a word, it will be fed through FitBERT's grammar correction algorithm to handle any discrepancies in conjugation.

3. User Input

The start of the sentence will be spoken by KATbot, and the missing input will be denoted with a change in voice and its part of speech. If the user needs more clarification of the part of speech, there will be a command word indicated in the directions that the user can use for more information. Once the user says a word, the word will be compared to the desired part of speech. Small grammatical mistakes will be forgiven, such as singular vs. plural or a mistake in verb conjugation. These mistakes will be handled with FitBERT's grammar correction tool. If there is no match in part of speech, KATbot will output an error statement and give the user another chance. Upon completion of the story we will ask the user if they would like to hear the story in full, tell another story or end the session.

Our user experience will look like the following chart:

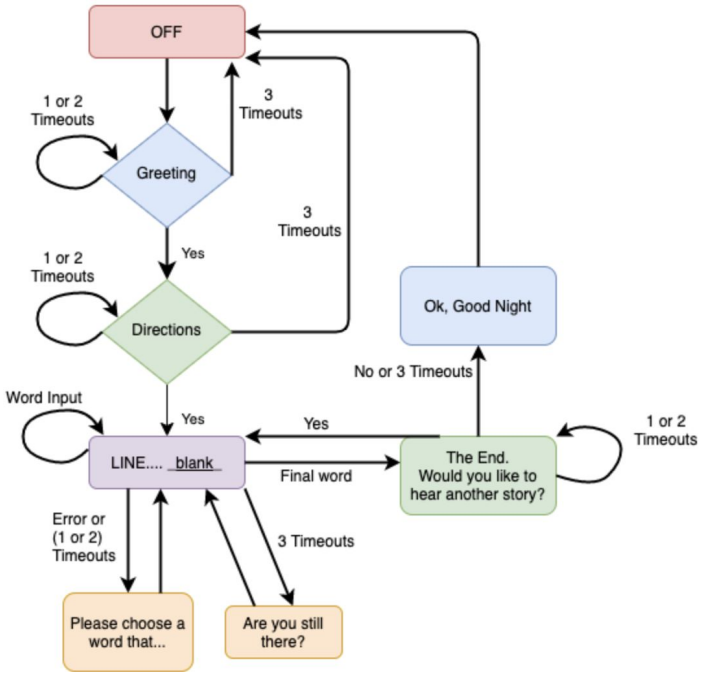


Fig. 10. User Flow Chart

Once all inputs, user and FitBERT, have been provided, the full sentence will be delivered to the text to speech module and KATbot will finish saying the sentence. While doing so, the algorithm can go ahead and start filling in any FitBERT inputs for the following sentences.

## VI. Validation Plan

### *A. Component Testing*

1. Machine Learning

For part of speech detection, the 90% accuracy goal will be tested with a random test dataset of words of a preK to 3rd grade vocabulary level and a random subset of sentences from the story templates to provide context. This component will be tested for both syntactic and semantic accuracy, by checking if the word input matches the general part of speech that is asked for (e.g. 'dog' for noun) as well as if it makes logical sense in the sentence (e.g. action verbs vs. linking verbs).

For synonym generation, the 85% accuracy level will be conducted similarly with a random test dataset of words from a preK to 3rd grade vocabulary list. Accuracy will be judged by cross listing the generated synonyms and antonyms with those from a thesaurus.

2. Speech Recognition Word Error Rate

WER is defined as the number of substitutions, insertions, and deletions over the total number of words to be recognized. To measure how well our speech recognition system is doing we will count substitutions, insertions and deletions while users interact with KATbot and calculate WER per session.

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 will validate this with the user satisfaction survey, which is described below.

### *B. System Testing*

1. System Latency

For system latency, the 4 - 6 second maximum latency will be tested by measuring the total amount of time between when the user has finished responding to a prompt from the system to when the system starts saying the next line of dialogue.

Below is a table of latency estimates from our own testing. We are still missing measurements for synonym generation, FitBERT fill in the blank latency, and data transfer latency from the Raspberry Pi to the laptop and back. To get the pitch shifting latency we used a python package and measured how long it took to run, however we will be writing our own pitch shifting algorithm, so the current value is just an estimate.

Table 1. Latency estimates for KATbot system

| Component | Time | Measurement method |
|---|---|---|
| Part of Speech Detection for user input | 0.07242 sec | Average over 6 inputs with one correct and one wrong input for each of 3 random sentences |
| Text to Speech Processing Latency | 0.012 sec | Average over 6 trials of taking text input and writing to an audio output file for a 17 word sentence. |
| Pitch Shifting | 0.003 sec | Average over 6 trials of pydub pitch shifting program |

2. Power

The goal of running for 30 - 45 minutes on battery power will be tested by interacting with KATbot for at least that long. We will also be doing power calculations for all components within KATbot to ensure that it can run for the required period of time with the batteries that we have chosen. We will measure the power draw of all components, and then buy batteries with enough power to meet the 30 - 45 minutes requirement.

So far, we have calculated the power draw of individual components to estimate power consumption. Before buying batteries, we will measure power consumption for a more accurate result.

Table 2. Current, Voltage, and Power Calculations for Hardware

| Component | Current Draw | Voltage | Power Consumption |
|---|---|---|---|
| Raspberry Pi 4 | 1A | 5V | 5W |
| Text Display | 0.25A | 5V | 1.25W |
| Eyes Displays | 20uA (10uA each) | 3.3V | 66uW |
| Servo Motor | 0.09A* | 4.8V | 0.43W |
| Speaker | 0.25A | 5V | 1.25W |
| Microphone | 0.5mA | 5V | 2.5mW |

*For the servo motor, the current draw is dependent on whether it is in use. For these calculations, we assumed that the robot arm moves no more than 5 seconds for every minute. The current draw in an idle state is 7mA. The current draw when moving can go upto 1A. The current listed above is the amortized current draw.

Our estimated power consumption is 7.9W.

3. Story Cohesion

Story cohesion will be evaluated relative to the original stories that the templates are derived from, Aesop's Fables, and versions of the templated stories with random inputs. There are five variables related to narrative cohesion that we will look for during this evaluation: logical sense, themes, genre, narrator, and style [18]. We will find volunteer graders to keep bias out of the testing process. First, each of the fables will be rated on a scale for each of these variables. Then, we will generate pseudo random stories, and ask the graders to evaluate them, to mimic the lowest possible cohesion we can attain. To simulate randomness, we will take random inputs from vocabulary lists to fill our story templates, still matching part of speech, as opposed to an actual user or an algorithm making educated choices. The last group of stories will be created by a separate group of users that will generate them using KATbot. The story cohesion score that we will aim for with this last group will fall within the range of the random input score and the original story score.

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 want to be able to match their metrics. They evaluated their robot on several different characteristics which we would like to match. 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 plan to implement this system testing by giving the users surveys after each use and having them rank how they felt about each of the above categories on a scale from 1 to 10.

## VI. PROJECT MANAGEMENT

### *A. Schedule*

Figure 11 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 is in charge of story creation and will construct all the machine learning based algorithms. Jade is responsible for the speech processing aspect and will handle both collecting audio input and the text-to-speech capabilities. Abha will build the physical robot, including integrating the peripherals and working on the robot arms.

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

### *C. Budget*

Figure 12 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.

### *D. Risk Management*

1. Story Creation

There are a few different potential points of failure for the story creation. The first is that the sentences have too little cohesion to work as a short story. To handle this risk, we are limiting user input to one word per sentence or sentence phrase, and choosing input blanks that are independent of the previous sentences. This way, the bulk of the personalization process stems from the algorithm, not the user. If the algorithm is unable to fill in the blanks well enough for cohesion, we can use templates that have fewer blanks initially. The other potential problem comes up when processing user input. If the user provides a word that fits the part of speech we are looking for but does not actually make sense with context (semantically), the robot may accept the input but output a nonsensical story. To handle this, we will need to strike a balance between specificity when asking for a word type and keeping the number of potential inputs large for more user control.

2. Internet Connection

The packages we have decided to use for speech recognition and text to speech require internet connection. Because speech is the method of user input and output for our system, we need to ensure that if a stable internet connection is not available, we need to have backups. To reduce this risk, we will be installing speech recognition and text to speech packages in KATbot. Specifically, we will be using PocketSphinx for speech recognition and Flite for text to speech.

## 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.

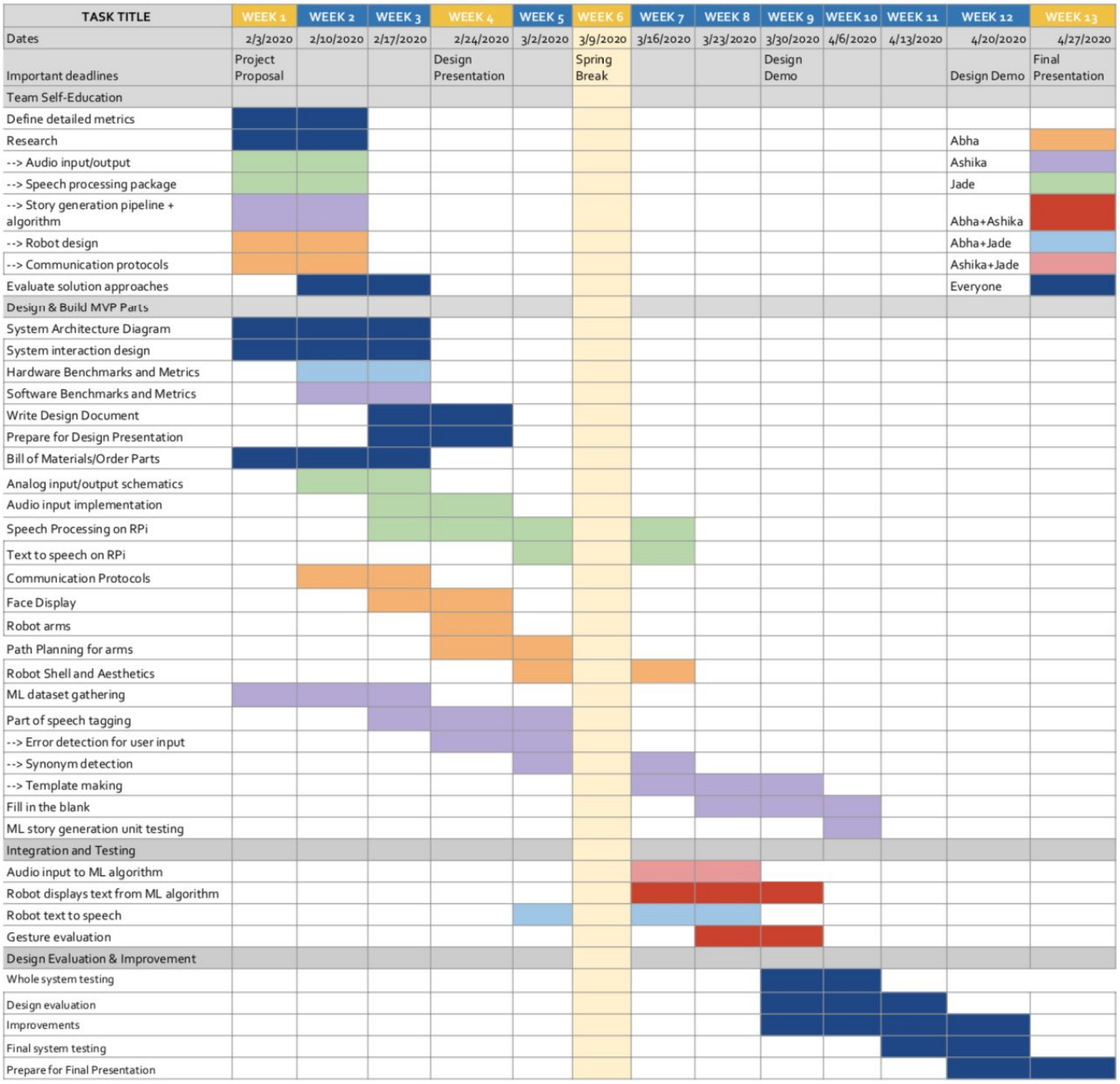


Fig. 11. Gantt Chart

| 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 | 2 | 19.95 | 39.9 |
| 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 | TowerPro SG-5010 - 5010 | 2 | 12 | 24 |
| Motor Shield | Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface - PCA9685 | 1 | 14.95 | 14.95 |
| Micro SD Cards | 32 GB 2-pack | 1 | 10.99 | 10.99 |
| Micro HDMI to HDMI Cable | | 1 | 5.99 | 5.99 |
| USB C Cable | 2 pack | 1 | 8.29 | 8.29 |
| Acrylic | 2ft x 4ft 1/8" Clear Acrylic | 1 | 17.5 | 17.5 |
| PLA Filament | 2.85mm 3D Printer PLA Filament Spool | 1 | 19.99 | 19.99 |
| Battery | 18W 10000mAh USB C Power Bank | 1 | 26.99 | 26.99 |
| | | | | 329.09 |

Fig. 12. Bill of Materials