A.I.Dan

A ChatGPT Integrated Virtual Assistant

By

Andrew Scott

Brahmteg Minhas

Leonardo Garcia

Final Report for ECE 445, Senior Design, Spring 2023

TA: Hanyin Shao

May 2023

Project No. 25

Abstract

A.I.Dan is a ChatGPT integrated virtual assistant built to vastly speed up the process of information lookup. Using speech recognition and OpenAI’s natural language processing models, a user can ask A.I.Dan a question and get a holistic answer within seconds. The original design envisioned a combination of a device for speech input and text, speech and visual output as well as a computer for execution of the software subsystems.

The presented product is fully software, but completes the core requirements of the product: speech recognition, text comprehension and response with an audio and visual output, including properly marked code segments in markup.

Contents

1. Introduction

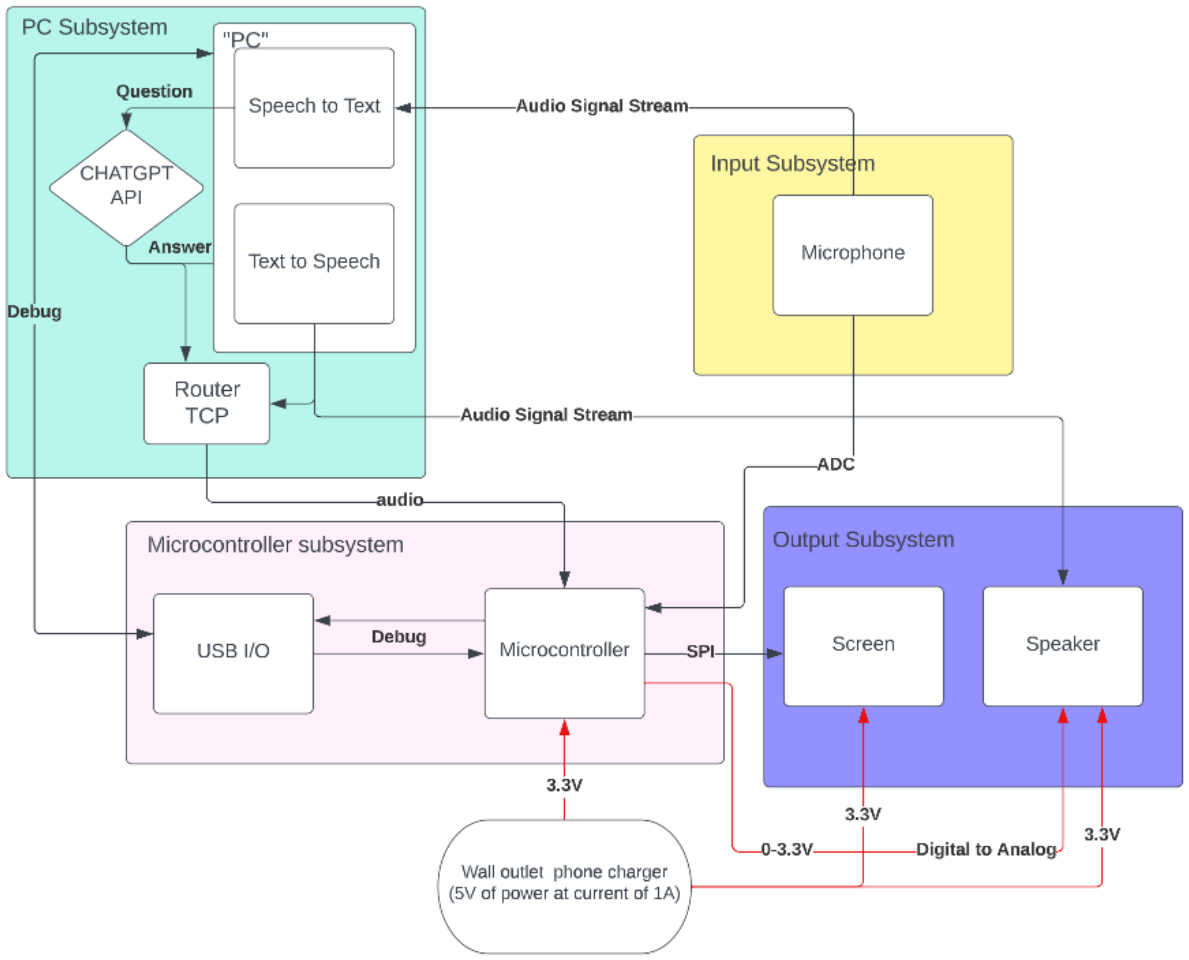
All existing virtual assistants use a search engine (primarily Google Search) as a data retrieval tool for answering questions posed to them by a user. The responses given by a search engine to a virtual assistant is a result or excerpt from the top search result that comes when searching for that question. However, this result is often not useful or relevant. Search engines are built to present multiple information sources from the web when presented with a question, not necessarily output one definitive answer. Furthermore, searches respond poorly to requests, unable to produce an output of a specified form.

OpenAI's ChatGPT is an AI language model that can respond to questions, generate text and have conversations. It is an extremely useful tool to respond to questions or present a singular output in a specified format. However, accessing ChatGPT requires going to the ChatGPT website, logging in and manually typing the question. This process is cumbersome and takes more time than it needs to.

Our solution is a device that integrates ChatGPT with a virtual assistant in order to nullify the weaknesses of both tools. Because ChatGPT is trained on human language, it is much better suited for a virtual assistant, which uses human language as input. As such, integrating ChatGPT into a virtual assistant would yield a much better tool for getting information and tailored responses to user questions. Our solution will allow the user to present the virtual assistant with a prompt or question. The device will then return the response that ChatGPT gives to the user input, both through a speaker and through a screen on the assistant.

2. Design

2.1 Block Diagram



2.2 Physical Design

The four major physical components of our original design are a small (2” diagonal) screen, our PCB, a microphone, and a speaker. These components will all be mounted inside a single 3” 3d-printed cube, with the speaker and microphone on top, the screen on the front, and the PCB on bottom. The USB Ports for power and Debug will be cut out of the back side and directly soldered to the pcb.. Outside of the enclosure will be a standard Wall plug DC 5V power supply, wired to the back of our cube through a power-only type c connection. As previously stated, the whole physical system outside of the PC was not used in demonstration.

2.3 Remote Block

The PC block consists of an external computer connected to the microcontroller through WiFi. The PC performs the majority of the heavy compute lifting of the entire project, performing the Speech recognition, ChatGPT API Calls, and Text to speech processing. Its implementation performed as expected, with each component being fully functional at the end of the project.

2.3.1 PC

The PC accepts audio input from the microcontroller (streamed via WiFi), and constantly translates it to text in chunks, listening for the keyword, "Hey A.I.Dan" (pronounced Aidan). When it hears the phrase, the PC will listen to the next sentence it gets (waiting for a long pause in audio to stop listening), convert the data to text and send it to ChatGPT's API.[1] The API text output is then transmitted as both text (to be displayed on the screen), and as audio (through a text-to-speech conversion) back to the microcontroller to be output to the user. Each software component functioned within the performance thresholds set during the design stage of the project with the exception of the components that interacted with the microcontroller. See Appendix A for more information on the testing done for audio/chatGPT requirements.

Table 1: PC Subsystem Requirement and Verification

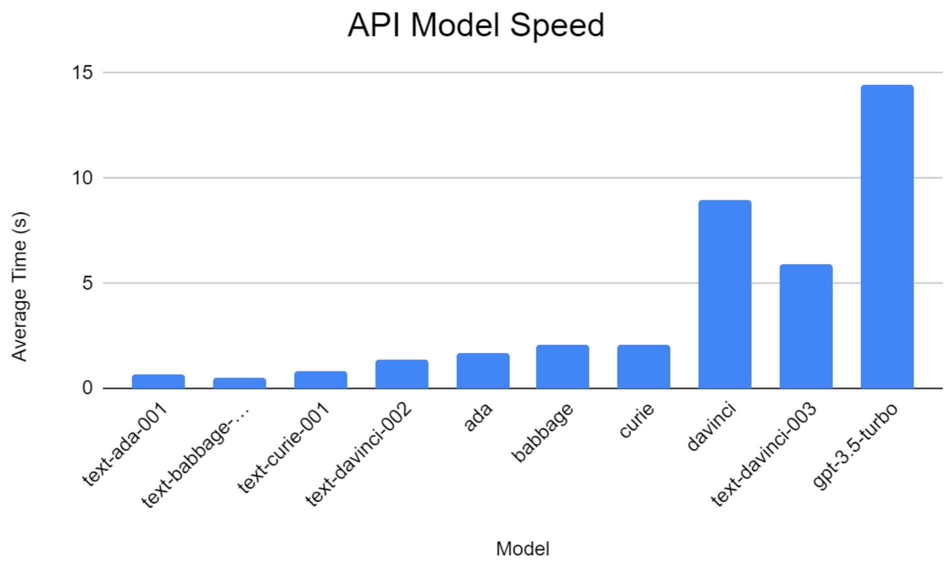
Requirement	Verification Procedure	Verification Result
The PC must maintain an open WiFi port for 1 hour, untouched, to listen for microcontroller input.	*Open a socket on the PC to check for microcontroller connectivity. Regardless of whether the microcontroller connects, ensure that the connection is still open after 1 hour.*	*Passed. Due to the PCB not working, the microcontroller never connected to the open socket on the PC. However, following 1 hour, the PC was still searching on the open socket for potential connectivity.*
The PC must be able to convert clear and simple speech into text using a pre-trained model with an 80% or greater accuracy rate	*Run the speech to text model on a real time audio stream and verify that the text output matches the spoken words with the specified (>80%) accuracy rate.*	*Passed. When speaking clearly into the microphone with simple speech, audio was transcribed as intended. Mistranscription occurred only when speech contained certain proper nouns (which were not classified as simple speech). See Appendix B for details.*
The PC must be able to analyze text to determine whether a	*Run the wake word detection software on real time audio streams. Should the detected*	*Passed. There was a recurring instance where a user saying "Hey Aidan" would transcribe to*

question is valid (contains the wake words "Hey A.I.dan" or similar) with an 80% or greater accuracy rate.	*audio contain the wake word "Aidan" or a similarly pronounced word, the output display shows "Wake Word Detected" and when the audio does not contain the wake word, the output display shows "Wake Word Not Detected." This verification passes with a >80% accuracy rate.*	*"Hayden" with the Speech-to-Text model. Accounting for this led to the verification being passed with a high accuracy.*
The PC must be able to convert standard text into speech using a pre-trained model, understandable at least 80% of the time.	*Input text files containing standard text into the text to speech module.* *Analyze the output audio files to ensure the output speech is understandable and accurate to the source text, within reason, greater than 80% of the time.*	*Passed. Standard text was converted into understandable and accurate speech with an extremely high accuracy. In testing, 100% of plain and simple speech was heard. See Appendix C for details.*
The PC must be able to output both a digital audio signal and text to the ESP32 microcontroller through WiFi, only outputting one or none 20% or less of the time	*Collect a digital audio file and a text file.* *Establish a TCP connection with the microcontroller.* *Send the text file, followed by the audio file, over the TCP connection to the microcontroller.* *Ensure that the microcontroller received data of the same size as the digital audio file and the text file for 80% or more of the trials*	*Failed. Due to the microcontroller being unrecognized by the PC, no data was able to be sent to the microcontroller. However, the files were all present on the PC should the microcontroller have been functional.*

2.3.2 ChatGPT API

The ChatGPT API generates the ChatGPT response. It receives text input from the PC and outputs the response of chatGPT as text back to the PC. During testing, the ChatGPT API worked as expected, with different models performing at varying capabilities. Testing the performance of the different models yielded the results seen in Figure 1.

Figure 1: API Model Speed Comparisons



The final product was delivered using gpt-3.5-turbo due to its performance and code generation capabilities.

One plan that was not able to come to fruition was the ability to pick an appropriate model for different tasks. Prior to deprecation in march of this year, there existed versions of ada and davinci that were designed specifically for code, as their primary models do not provide programming help. Had we been able to use those models in tandem with their text-based counterparts, full functionality would have been achievable in our desired timeframe of 4 seconds. As it stands, however, that would only speed up text-only interactions as gpt3.5 turbo is the current fastest code generation model.

Table 2: ChatGPT API Subsystem Requirement and Verification

Requirement	Verification Procedure	Verification Results
The API must be able to generate an answer to a text	*Collect a text file containing 10 prompts for chatGPT.* *Run the text file through the ChatGPT API model, using*	*Passed. Testing on multiple OpenAI's models gave varying results (see figure 1). GPT-3 text models all passed under the 4*

input with a text output within 4 seconds of the API call	*python's timeit library to track how long it took* *Analyze the output text file of the ChatGPT API model to ensure that it contained a reasonable response to the prompt and received it within the timeframe.*	*second threshold but models with integrated code completion took longer.*

2.4 Onboard Block

The onboard block takes in audio data through a microphone and transmits it through WiFi to the PC. It also receives data through WiFi to be output to a screen and through a speaker. Unfortunately, we were unable to get a working onboard block at the completion of the project. The most likely explanation is due to a short in the ESP32 or the connecting components.

2.4.1 Microcontroller

zConsists of an ESP32 microcontroller with built-in WiFi. It serves as the liaison between input and output data. Input data is received from the microphone and transmitted to the PC through WiFi. From the PC, it receives data back both in the form of text and audio. The microcontroller sends the audio directly to the speaker to be output to the user and converts the text input to visual and sends it to the screen through the SPI protocol. The ESP32 was unable to be integrated into the final project due to the PCB failure.

Table 3: Microcontroller Subsystem Requirement and Verification

Requirement	Verification Procedure	Verification Results
The microcontroller must be able to host a TCP server using Wifi to interface with the PC.	*Establish a TCP connection between the PC and the microcontroller.* *Ensure the TCP connection is valid by checking the remote client (the PC) for a message from the microcontroller which states:* *"ESP32 connected".*	*Failed. The microcontroller was unresponsive so verifying the TCP connection was not possible.*

2.4.2 Input Subsystem

The input subsystem contains a microphone, which is constantly streaming input audio to the microcontroller's built-in analog-to-digital converter. Because of our inability to integrate the PCB, we were unable to utilize this subsystem.

2.4.3 Output Subsystem

The output subsystem displays data received from the microcontroller through a screen through the SPI protocol and through the speaker following a digital-to-analog conversion. Due to the PCB failure, we moved the output subsystem to the PC and displayed output to a python tkinter GUI.

Table 4: Output Subsystem Requirement and Verification

Requirement	Verification Procedure	Verification Results
The screen must be able to properly display code snippets, which requires at least 16-bit color.	*Connect the microcontroller to the PC via USB.* *Download an SPI Code Display Test program which contains a formatted code snippet with text of 16 different colors.* *Ensure that the code displays legibly and 16 colors are distinguishable and present on the display.*	*Failed. The screen received power but was unable to display an image.*

2.4.4 Power

The power subsystem powers all onboard components.

Table 5: Power Estimates for Selected Components

Component	**Power Draw**
Screen	1 W[2]
Microcontroller	2.50 W[3]
Speaker	0.5 W[4]

Digital-to-Analog Converter	<0.1 W[5]
Total	**4.0 - 4.1 W**

Based on the above estimates, a standard wall plug DC 5V power supply operating at 1A should be sufficient.

2.5 Tolerance Analysis

Data Throughput (PC-Microcontroller data transfer)

The largest identified issue associated with this design is the data management on the microcontroller, and its interaction with the PC. Even the most robust microcontrollers only contain around 512 KiB of memory, which corresponds to roughly a second of audio information. Data will be streamed to the PC as fast as possible, and taking and running Text-To-Speech will act similarly quickly, but both streams could prove too much for the microcontroller.

To demonstrate feasibility, assume 100 KiB of memory allocated for each individual stream, up and down. Wifi speed is quite high relative to these numbers, so the actual communication shouldn't be an issue (ESP32-C3 supports 150mbps, and any PC we would be using will support at least 300mbps) Assume standard 44khz audio at 12 bits of digital resolution.

$$44000hz \cdot 12b = 528000\, b/s = 528\, Kib/s \tag{1}$$

A single audio stream comes out to 528 Kib/S. With the initial 100 KiB memory stream assumption, we get:

$$\frac{528\, Kib/s}{100\, Kib\ stream} = 5.28 \text{ refreshes/s} < 160\, mhz \tag{2}$$

That means we need to fully refresh and clear the memory 5 times over each second. This should be acceptable, given the SRAM's posted speed of 160mhz.

2. Costs and Timeline

4.1 Parts

Table 6: Parts Costs

Part	Manufacturer	Unit Cost ($)	Quantity	Total Cost ($)
Chat Completion API	OpenAI			0.28
1mm pitch 11x1 male connector	Molex	1.61	3	4.83
1mm pitch 11x1 female connector	Molex	0.42	3	1.26
28 gauge pre-crimped wire	Molex	1.13	20	22.60
USB B receptacle	Samtec-inc	1.32	2	2.64
USB A-B 2.0 cable	CNC Tech	4.65	1	4.65
Ferrite beads	Laird-Signal Integrity products	0.18	6	1.08
slide switch	E-switch	1.29	4	5.16
10kOhm potentiometer	Vishay Spectrol	4.97	2	9.94
OPA1692 op-amp	Texas Instruments	2.41	2	4.82
LMV321 op-amp	STMicroelectronics	0.62	2	1.24
RF Antenna	Taoglas Limited	3.52	1	3.52
2x1 side-entry terminal block	Wurth Elektronik	0.41	5	2.05
USB-C power only receptacle	Kycon, inc.	0.91	4	3.64
32Ohm Speaker	Soberton Inc	1.75	1	1.75
Omni-directional Microphone	PUI Audio, Inc	3.83	1	3.83
ESP-32 Microcontroller	Espressif inc	3.30	2	6.60
320x240 2” screen	DFRobot	17.90	1	17.90
PCB Order	JLCPCB	2.22	5 (min. order)	11.10
Total				**108.89**

4.2 Labor

UIUC Graduates in Electrical Engineering make ~$80,000 per year on average [1]. Prorating to a 2000 hour work year, 80,000 equates to $40 an hour. Based on this, and our average weekly hours of roughly 8,

$$Labor\ Cost = \$40 * 8\ hrs * 16\ weeks * 3\ people * 2.5 = \$38,400 \tag{3}$$

$$Total\ Costs = Labor\ Cost + Part\ Cost = \$38,400 + \$108.89 = \$38,508.89 \tag{4}$$

4.3 Schedule

Table 7: Actual Timetable for Project Design Cycle

Week	Deadlines	Tasks		
		Brahmteg	Leo	Andrew
2/13		Prepare Design Document and Team Contract		
2/20	Design Document, Team Contract	Text-to-Speech Model	Finish Design Document	PCB Design, Finish Design Document
2/27	Design Review	Speech-To-Text Model	Finalize PCB	Finalized original pcb design
3/06	First Round PCB Orders, Teamwork Evaluation 1	ChatGPT API	Data Stream	Ordered components for original pcb design, worked on requested pcb redesign
3/13		Create pipeline from Text-To-Speech to	Data Stream	finished pcb redesign,

		ChatGPT model		ordered pcbs
3/20		Work on buffer for audio input for speech-to-text transcription	Data Stream	soldering and parts ordering
3/27	Second Round PCB Orders	Test multiple ChatGPT models for best accuracy and performance	Testing USB connection	Soldering, PCB testing
4/03		Complete full pipeline of input audio to output text response	Configuring Ardino code for microcontroller	Soldering, PCB testing
4/10	Team Contract Fulfillment	Create GUI for demonstration of software project	Testing Screen and Configuring Screen code.	Glue Code, setting up GUI, finished full soldering
4/17		Prepare Final Demo, Final Presentation and Final Paper		
4/24		Prepare Final Demo, Final Presentation and Final Paper		

5. Conclusion

5.1 Accomplishments and Uncertainties

While the full original design was not accomplished, a useful tool was developed in the end. The created app allows users to interact via audio with ChatGPT, and use it hands-free. Wake-word recognition was generally successful, and depending on the specific AI model being used, the response times were as fast as a couple of seconds. The best working section of the project was definitely Text To Speech, which had zero complications the whole time. Because this program is coded in python, it would be trivial to migrate it to a small fully-fledged computer such as a Raspberry PI or Jetson Nano, connect up a screen, microphone, and speaker, and have the originally intended experience become a real product.

5.2 Ethical Considerations

When AI solutions are being developed, ethical and safety concerns abound. Based on IEEE's Code of Ethics, there are two major ethical concerns that should be addressed with this design. The first, violating point I.2, is that AIs like ChatGPT are capable of misinformation, despite most users considering them infallible. On newer versions of ChatGPT, it does an excellent job of informing its users when it believes that the question asked might lead to safety concerns or untrue information. This is not perfect, but can be easily patched with a rudimentary End-User Licensing Agreement (EULA) that you have to read and accept before using our product.

One final ethical concern presented by this project is the issue of collecting user data, which also touches on IEEE's Code of Ethics point I.2. In order for this project to work, a microphone must be constantly on and recording the room around it, presenting a safety risk to users were that data to be leaked or tapped. While many companies use this data to reinforce their own learning models, we will be discarding it as soon as humanly possible to protect our users. While we do not keep the audio data, as a result of using Google's API, it is possible that they might be using the audio in ways our users did not intend. According to Google themselves, they only use audio of users who opt into their data collection, which we have not. While it is healthy to be skeptical of security claims by large companies modernly, taking them at their word is as much as we can reasonably do.

5.3 Future Work

A much more involved, but likely invaluable extension of this project would be to move the whole program to a microprocessor, and remove the PC from the equation entirely. In order to do that, and maintain reasonable latency, much of the signal processing (STT and TTS) would need to be done onboard. Accessing chatGPT would still need to be done through the cloud, necessitating an internet connection. Research into the viability of this has been done, and quick Text to Speech has been accomplished within the scale of a microprocessor has been achieved [2], though it did require a built-to-purpose microprocessor. Given the age of that paper (published in 2007), it is likely that a similar approach would no longer require specialized hardware due to power density scaling. While no similar commercially available solutions for Speech recognition/Speech to text exist, some companies claim they can accomplish this with existing hardware, such as MicroAsr [3].

A complication with room for improvement was the microphone experience. In testing, users had to be relatively close to the computer and enunciate clearly in order to have their response taken properly. Accents would also often ruin the speech conversion, making it only really usable by Americans with American accent By changing the microphone to one specifically designed for full-room listening applications, applying more specific post-processing on the audio (like Nvidia's Nvidia Voice), or making our own speech recognition algorithm, this part of the project's performance could be dramatically improved.

References

[1] *“Salary Averages”, web page. Available at: https://ece.illinois.edu/admissions/why-ece/salary-averages*

[2] S. Dey, M. Kedia and A. Basu, "Architectural Optimizations for Text to Speech Synthesis in Embedded Systems," *2007 Asia and South Pacific Design Automation Conference*, Yokohama, Japan, 2007, pp. 298-303, doi: 10.1109/ASPDAC.2007.358002.

[3] “Speech Technologies”, web page. Available at: https://microasr.com/technology.html

Appendix A: ChatGPT API Testing Methodology

For tests where different prompts were needed, we used a list of ten questions that ChatGPT themselves generated. The list of questions was:

1. "Can you tell me a joke that always makes people laugh?",
2. "What's the most interesting fact you know about history?",
3. "How can I improve my writing skills?",
4. "What's the best way to stay motivated when working on a long-term project?",
5. "Can you recommend a good book for me to read?",
6. "What's the best way to approach a difficult conversation with a friend?",
7. "Can you give me some tips for staying organized and productive?",
8. "What's the most important thing to consider when starting a new business?",
9. "Can you explain a complex concept in a simple way?",
10. "What's your opinion on the impact of technology on society?".

This list covers many real question cases, and does not include programming specific language. We left out programming intentionally, because not all of the tested language models have coding capabilities. While it would be possible to add more varied questions, it is not strictly necessary. This battery had relatively consistent results for the same language model over different runs (ignoring the high traffic models that varied a bit depending on the time of day), and had similar length responses over different models, which does affect the overall speed of a response.

Questions like these were also used to test for accurate Speech to Text functionalities, which we graded based upon ChatGPT’s ability to properly interpret. Even if the language was not exactly transcribed perfectly, because ChatGPT is able to understand imperfect speech, as long as the output was appropriate for the original intended question, that was counted as a success.

Appendix B: Speech-To-Text Testing

Table 8: Speech-To-Text Testing Results

Prompt	Transcription	Word Count	Errors
The quick brown fox jumps over the lazy dog.	the quick brown fox jumps over the lazy dog	9	0
Peter Piper picked a peck of pickled peppers.	Peter Piper picked a peck of pickled peppers	8	0
She sells seashells by the seashore.	she sells seashells by the seashore	6	0
How much wood would a woodchuck chuck if a woodchuck could chuck wood?	how much wood would a woodchuck chuck if a woodchuck could chuck wood	13	0
I am speaking to test the accuracy of this speech to text model.	I am speaking to test the accuracy of the speech to text model	13	1
The cat in the hat is a great children's book.	Cat in the Hat is a great children's book	10	1
The rain in Spain falls mainly on the plain.	rain in Spain falls mainly on the plane	9	1
The sun shines bright on my old Kentucky home.	the sun shines bright on My Old Kentucky Home	9	0
I can't believe it's not butter.	Can't Believe It's Not Butter	6	1
The jagged rocks jutted out from the craggy cliff face, casting a shadow over the treacherous path below.	Jagged rocks jutted out from the craggy Cliff face casting a shadow over the treacherous path below	18	1
Totals		**101**	**5**

To test the speech-to-text model, ChatGPT was asked to generate 10 random spoken prompts. These prompts were then spoken and input into the speech-to-text model with the transcription output being compared to the original prompt. Capitalization and punctuation were ignored. The total erroneous word rate across 101 total words is

$$\frac{5}{101} \cdot 100\% = 4.95\% \tag{1}$$

Which is less than the 20% error threshold requirement.

Appendix C: Wake Word Testing

Table 9: Wake Word Testing Results

Prompt	Contains Wake Word	Wake Word Found	Correct
Hey Aidan, can you tell me a joke?	Yes	Yes	Yes
What's your favorite movie of all time?	No	No	Yes
Could you recommend a good book to read?	No	No	Yes
Hey Aidan, what's the weather like today?	Yes	Yes	Yes
Can you suggest a recipe for a quick and easy dinner?	No	No	Yes
Hey Aidan, what's the latest news in technology?	Yes	Yes	Yes
What's your opinion on the current political climate?	No	No	Yes
Hey Aidan, do you have any travel tips for visiting Europe?	Yes	Yes	Yes
What are some of your hobbies or interests?	No	No	Yes
Can you recommend a good podcast to listen to?	No	No	Yes
		Total Correct:	**10**

10 randomly generated prompts with some beginning with “Hey Aidan” were input into the wake word detection algorithm. Of these 10, all 10 were correctly detected (100% detection rate) which fits into the verification threshold of 80% detection.

Appendix D Text-To-Speech Testing

Table 10: Text-To-Speech Testing Results

Prompt	Word Count	Mispronounced Words
The quick brown fox jumps over the lazy dog.	9	0
Peter Piper picked a peck of pickled peppers.	8	0
She sells seashells by the seashore.	6	0
How much wood would a woodchuck chuck if a woodchuck could chuck wood?	13	0
I am speaking to test the accuracy of this text to speech model.	13	0
The cat in the hat is a great children's book.	10	0
The rain in Spain falls mainly on the plain.	9	0
The sun shines bright on my old Kentucky home.	9	0
I can't believe it's not butter.	6	0
The jagged rocks jutted out from the craggy cliff face, casting a shadow over the treacherous path below.	18	0
Totals	**101**	**0**

10 generated ChatGPT prompts were input into the text-to-speech model and the speech output analyzed for incorrect or mispronounced words. The rate of mispronounced words is

$$\frac{0}{101} \cdot 100\% \;=\; 0\% \tag{1}$$

which is less than the 20% error threshold requirement.