

Hidden Guardian

FINAL REPORT

Team 9

Client - Kelli Rout

Adviser - Phillip Jones

Jennifer Frank - Team Lead/Mobile Application Developer

Keng-Yik Ho - Hardware Engineer

Matthew Pedretti - Hardware Engineer

Thomas Kirby - Backend and Database Developer

Jacob Stilwell - Mobile Application Developer

Team Email - sddec18-09@iastate.edu

Team Website - <https://sddec18-09.sd.ece.iastate.edu/>

Table of Contents

List of Figures	3
Introduction	3
Acknowledgement	3
Problem and Project Statement	3
Operational Environment	3
Intended Users and Uses	4
Assumptions and Limitations	4
Deliverables	5
Design	5
Hardware and software	5
Diagrams	6
Implementation Details	7
Schematics	7
Final product functionality	10
Project Requirements	10
Functional Requirements	10
Nonfunctional requirements	10
Constraint Consideration	11
Testing	11
Appendix I: Mobile App Screen Images	14
Appendix II: Operational/User Manual	14
Appendix III: Alternate Designs	17
Appendix IV: Future Suggestions & Additions	17
Appendix V: Code	18

List of Figures

Figure 1 - Conceptual Sketch

Figure 2 - System Block Diagram

Figure 3 - Database Schematic

Figure 4 - Controller Attachment Schematic

Figure 5 - Volume Controls Schematic

Figure 6 - Speaker Schematic

Introduction

ACKNOWLEDGEMENT

Professor Swamy Ponpandi and Professor Phillip Jones provided technical support and shall be acknowledged for assisting Team 9 in the development of Hidden Guardian. Similarly, Professor Zambreno, Professor Daniels and the SE, CPR E and EE departments provided financial support in the creation of this product.

PROBLEM AND PROJECT STATEMENT

Kid's lives today involve online interactions more than ever. Their interactions with strangers invoke the risk of sharing unwanted personal information, password theft, receiving viruses, being cyberbullied and more. However, currently there is no practical parental monitoring system to keep track of these unwanted videogame interactions.

Our solution is Hidden Guardian. Hidden Guardian is a combination of a speaker, controller attachment and mobile application. Through the mobile app the parent user will be able to input keywords, such as "meet", "phone number", "address" and the mobile app will display the conversations that have triggered those keywords. The app will also provide minute long audio clips of the child's conversations. Hidden Guardian will only be able to be enabled or disabled by the parent.

Overall, our goal is to provide a system so that parents can have peace of mind that their kids videogame interactions are safe.

OPERATIONAL ENVIRONMENT

The typical operating environment of Hidden Guardian is a household setting. Given that houses are typically well climate controlled the hardware component won't be subject to any extreme conditions. It will however be handled frequently and should be at least somewhat robust in order to withstand moderate impacts.

INTENDED USERS AND USES

The intended user for this project will be for 2 targets, parents and their children. The children will be able to use the bluetooth speaker for their gaming uses or listening to music. The speaker will provide immersive gaming experience by providing high quality audio. The device will also have a microphone to enable children to communicate with other players online to improve gameplay.

Parents will be using it as a device to monitor their children's activities. By using the microphone on board, the device will record all of the child's conversations while the device is turned on. Parents will be able to access logs of recordings using the app and search for keywords in the recording such as phone numbers, and addresses to help improve the safety of their children. Additionally if the child is using a headset the parent will be able to use the speaker to listen to their child's live conversations.

ASSUMPTIONS AND LIMITATIONS

Assumptions:

- At the end of development, this product will not be commercially ready to consume
- The maximum amount of hours/week of data we will store is 15.
- We don't need to comply with licensing rights, since we are handling the data outside of the console.
- Hidden Guardian is used in a household

Limitations:

- For speech to text recognition we will only support one language (English).
 - We will have support for additional languages in the future, but not for our first iteration of the project
- Our device won't be able to separate each voice and associate it with different users
 - However, the google speech to text API has the capability of splitting up users, though we would still not be able to assign it to a username

- We won't be able to access text chat communications
- The device will require relatively large amounts of power for a battery powered device
- Hidden Guardian's Speaker can only be used in Parent Mode up to 30 feet away

PREVIOUS LITERATURE AND RESEARCH

Activity Monitoring

There are a variety of spyware type applications for computers that will capture every aspect of a users online usage. Veriato Investigator is an enterprise level example of this, with other products offering similar capabilities, some of which are marketed at parents, like WebWatcher. However these products do not allow for recording or monitoring gaming console activity. There are products like Clean Router that work at the router level to monitor and logs all activity from any device accessing the internet, but this is not specific to gaming consoles like XBox and does not allow search capabilities.

Console Activity Monitoring

Microsoft allows parents to set a variety of activity monitoring settings for their children's accounts. Using a feature called activity monitoring parents are able to get weekly reports about games and apps their child is using, search terms the child is using, and websites they are using, among other things. However they do not provide a method to capture and filter through the child's audio conversations.

Similar Devices

Currently, the only devices remotely similar are generic bluetooth speakers and a stereo headset adaptor for the XBox one controller. But none of these devices offer any kind of audio capture capabilities.

Deliverables

Mobile Application

The smartphone app gives the user an interface for looking at the conversations that triggered keywords. The application allows them to create an account that is attached to the "Hidden Guardian" device. Within the app they can also edit and customize the keywords they would like Hidden Guardian to be listening for.

Hardware Speaker

The wireless speaker has a rechargeable battery and a bluetooth receiver. The speaker plays audio data received via its bluetooth receiver, and can be recharged using a micro USB cable. A switch on the speaker controls whether or not the microphone data of the user is played, allowing the speaker to operate as a relay of the whole conversation.

Hardware Controller Attachment

The controller attachment houses the controls for both the microphone and speaker audio. Buttons can control the microphone and speaker volume, as well as muting the microphone. The attachment can attach to the bottom of an Xbox one controller using the 3.5mm jack. The attachment also has a 3.5mm jack itself to serve as a passthrough for any user wanting to use their own headset. The core of the controller attachment is a raspberry pi that handles creation of audio clips and transmitting them to the speaker and the database. The buttons all interface with the pi where volume levels are being controlled.

Database

The database receives audio data and converts it into text using Google Speech-to-text API. Both the audio data and text transcription are stored in the database. The database is an SQL server hosted on a windows 10 machine. The database stores a product code for all devices that have been produced, the user whose account is associated with that device, and the devices customizable name. Another table stores the list of keywords for each user. The last table stores all of the audio messages with information like the time and date of creation, session ID, message ID, and which user and hardware device the message belongs to.

Design

HARDWARE AND SOFTWARE

Hardware

Bluetooth Speaker

- Bluetooth Module
 - Receive audio for the speaker.
- TDA7266 Dual Bridge Amplifier
 - A class AB amplifier to amplify the signal received from the Bluetooth module and send to the speaker.

Rechargeable 2000 mAh Li-Ion battery

Raspberry Pi Zero W

- To send the audio or mic signal to the bluetooth speaker and connect to the database through wi-fi

Raspberry Pi sound card

- Converts analog audio into a digital format

Bluetooth transmitter

- Transmit audio via bluetooth

Software

Microsoft SQL database

- Microsoft SQL Server Management Studio 17

Android app

- Android Studio Version 3.2
- JRE: 1.8.0_152 - release-1136-b06 amd64
- JVM: OpenJDK 64- Bit Server VM by JetBrains s.r.o

DIAGRAMS

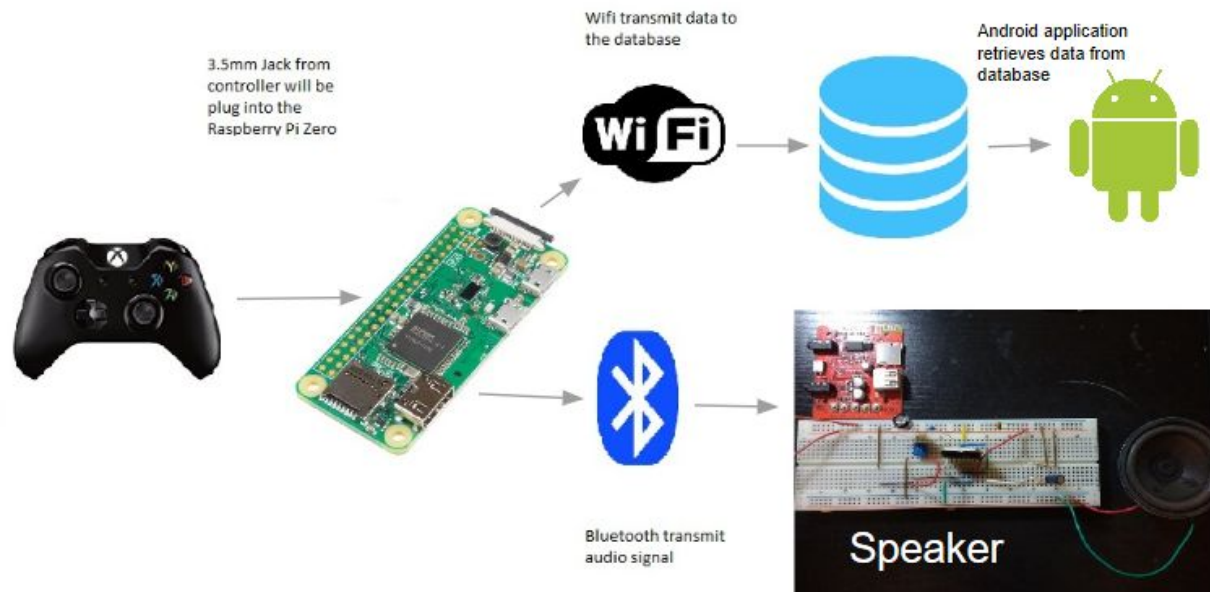


Figure 1 - Conceptual Sketch

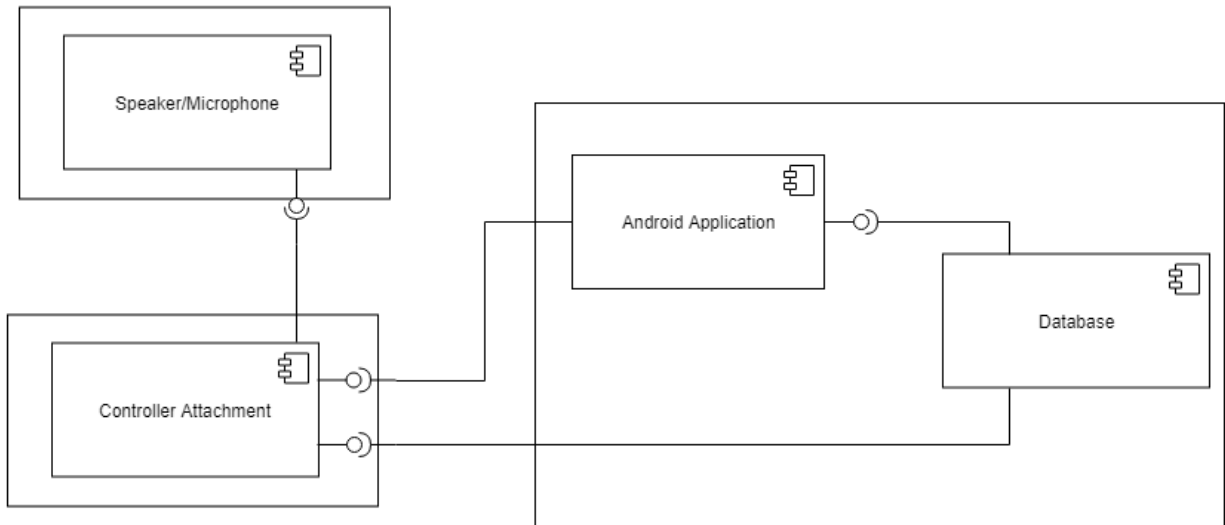


Figure 2 - System Block Diagram

Implementation Details

SCHEMATICS

The following is the schematics of our Microsoft SQL database structure. Each speaker/controller attachment hardware set is assigned a unique “device code” that we define as the machineid in our database. In theory, every time a Hidden Guardian product is produced it is attached to a unique ID that is then stored in the database as the machine id with empty “userid” and “machine name” variables. Then when a user creates an account on the mobile application it prompts them to enter their device code and their machine name, which will appropriately update that table in the database.

It is possible for a user to have multiple Hidden Guardian’s, which means the mobile application uses the machine id to decide which gaming sessions to display for the main page. From there, the unique identifier is the session id combined with the machine id to display the gaming session page and finally the machine id, session id, and message id for the gaming session entry page. The userid is the unique identifier for the keyword page, as the parental user would have the same keywords for multiple devices.

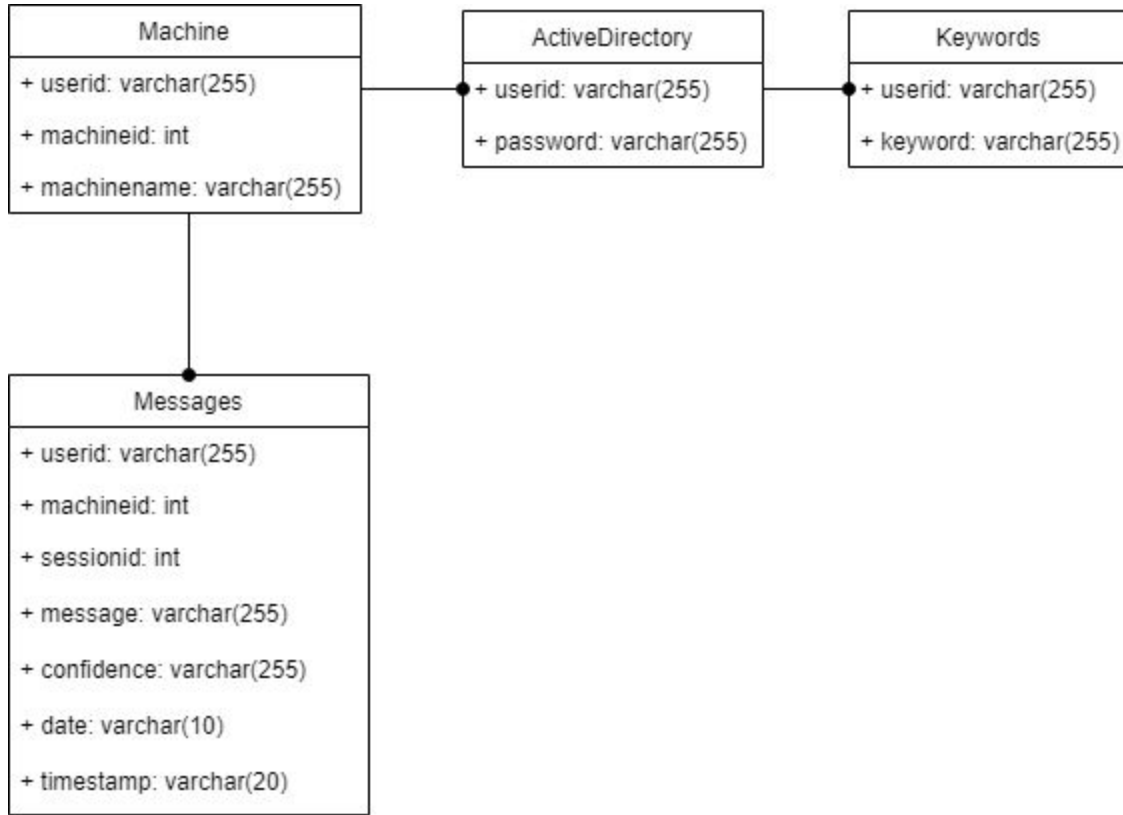
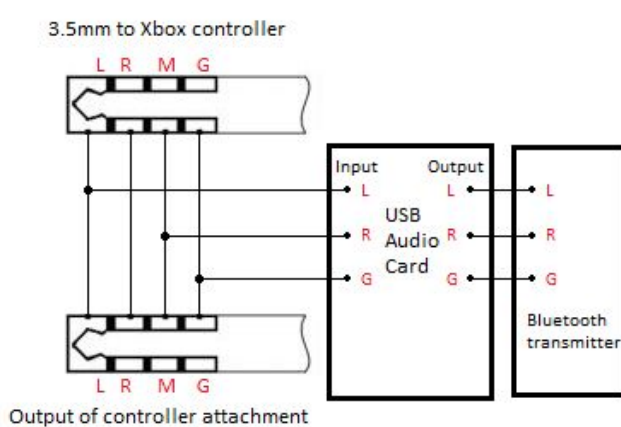


Figure 3 - Database Schematic

The following is the controller attachment schematic. This schematic is one part of the controller that is responsible for the transmission and recording of audio data. The USB audio card and the Bluetooth transmitter are off the shelf components. The audio from the xbox controller will be connected to the 3.5mm output of the controller attachment. The left signal from the xbox will be connected to the left-channel input of the USB audio card. The mic signal from the output of the attachment will be connected to the right-channel input of the USB audio card. The left, right and ground of the USB Audio card output will be connected to the left, right and ground of the Bluetooth transmitter.



L=Left signal
 R=Right signal
 M=Mic signal
 G=Ground

Figure 4 - Controller Attachment Schematic

The following is the schemation of the volume controls of the controller attachment. When the GPIO of the raspberry pi detects 3.3V on its input. The pin input will turn from 0 to 1. When the pin is 1, a script will be triggered to do the specific function. So 3.3V is connected to one end of the switch and the other pin of the switch is connected to the GPIO.

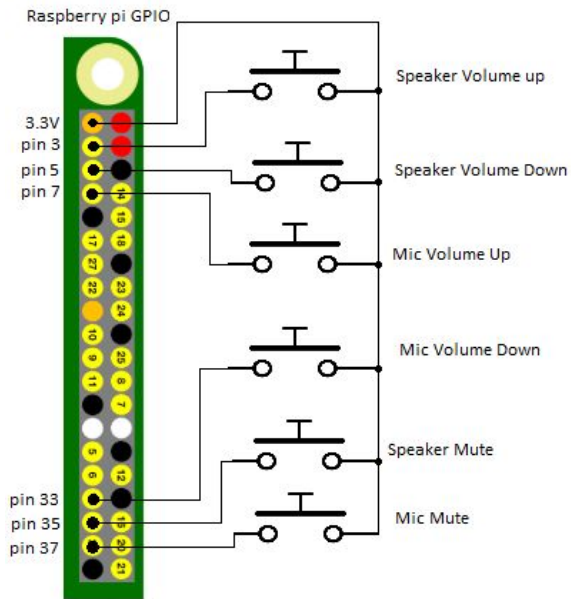


Figure 5 - Volume Controls Schematic

The following is a schematic of the bluetooth speaker. The charger, regulator and bluetooth modules are all off the shelf components. Our implementation of the speaker was created in breadboard only due to time constraints.

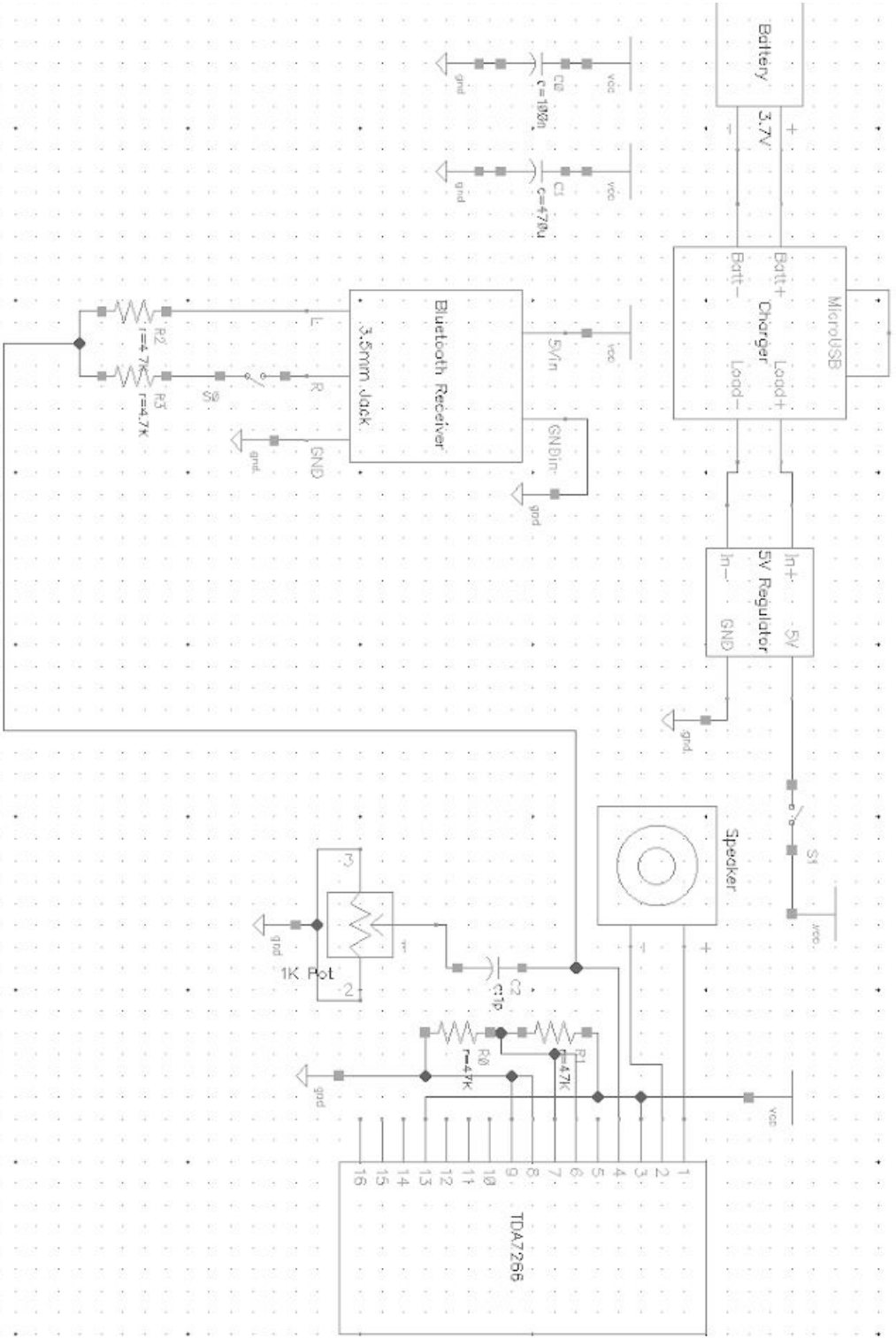


Figure 6 - Speaker Schematic

FINAL PRODUCT FUNCTIONALITY

Our final product consists of a controller attachment, speaker and mobile application. All mobile application screen flows/pages are displayed in Appendix I.

Through our current product we can record sound through our microphone. This is then sent to our server using the MQTT protocol. Following the data transmission the server will read the data, translate the speech using Google speech to text API and insert it into the database. this database can be read from our phone app detailing the message, keyword triggers, confidence, timestamps (so they can figure out when the message was said), as well as ID's denoting which gaming session and message number the data was.

Time constraints hurt our ability to implement a few proposed features and design ideas. This is discussed further in the "Further Suggestions, Additions and Concerns" section.

Project Requirements

FUNCTIONAL REQUIREMENTS

- The device will be able to record the conversations of a user while gaming online
- The conversations will be stored on a database which can only be accessed by an account tied to the device
- The database will be able to be parsed to find relevant conversations according to keywords defined by the user
- The speaker must be wireless and rechargeable
- The controller attachment must have a microphone
 - Due to time constraints this requirement was not able to be implemented.
- The speaker will be optionally able to relay the live conversations of a user by using a headset with the device instead.

NONFUNCTIONAL REQUIREMENTS

Our focus on nonfunctional requirements include:

- Efficiency/Speed
- Interoperability
- Usability
- Security
- Scalability

Our main user is parents who could be of ages 25-60 and will have varying technological understanding. A requirement that is of utmost priority is to make the mobile application easy to understand and use. We also will focus on speed and efficiency with our applications, so that data is being sent at appropriate time intervals, there is not a slow buffer time or cached incorrect data. Our product needs to be interoperable so that the purchase of a speaker contains a code to access the console application which can smoothly connect to a database which connects to a mobile app all synced together. We will also need to have interoperability between the speaker and the Xbox One Controller Attachment via wifi. Security is an important requirement so that we are making sure the text data is only viewable by the parent and their username/password data is protected. We also need to make sure that our product is scalable so for future potential production it can be extended to iOS apps, web applications, PS4's and other gaming systems. Since we cannot test this, we will have to focus on having excellent documentations, commenting and making sure our code is modular and easy to understand and pass on.

CONSTRAINT CONSIDERATION

We will want our product to be user friendly for parents who may not be very tech-savvy. We plan to assist with this by having thorough documentation for setting up the controller attachment and the speaker.

We also have to consider the privacy of this product. Online members who participate in video game chat sign a release with Xbox that they are willing to extend their privacy rights when using this product. This is meeting a standard, though may be considered unethical. The product is helping parents monitor unsafe actions, cyberbullying and avoiding password theft but it is also recording others so it is in a grey area.

Storing data costs money. In order to make the product viable the costs of data storage needs to be maintainable. 500TB of data per month would service about 90000 users at an estimated 15 hours of audio data per week using uncompressed audio files. This assumes that the audio files are deleted at most one week after they are added to the database. Using Amazon's standard storage rates of 3.9 cents/GB for the first 50TB and 3.7 cents/GB for the remaining 450TB gives a total monthly storage cost of \$18600. We tested compressed audio but had to revert to uncompressed due to time constraints and the increased complexity of handling compressed audio before using Google's text-to-speech service. Data usage would decrease by the same ratio as the audio files are compressed. For example if a 5 to 1 compression ratio is achieved (not unreasonable with the assumption that most of the recorded audio will contain little to no

speech) the same number of audio files would occupy 100TB rather than 500TB, costing \$3800 per month. Compressing the audio would also allow for faster load times on the app when retrieving audio data. Even with compressed audio the data storage cost can add up over many months of usage per user. The cost of uncompressed audio with these assumptions comes out to 21 cents to store data for one user per month. This cost is something that should be considered in the price of the product.

There will also be a power constraint with the controller attachment's power consumption. An Xbox controller is powered by either two double A batteries or a rechargeable battery pack. Double A batteries are roughly 2500mAh, giving the controller roughly 25% more storage capacity than even larger smartphone batteries. There are no official sources on the expected battery life of the controller but users typically report between 10-30 hours of usage. Ideally we want to maintain as much of the controller's battery life as possible. The power consumption of a raspberry pi zero when running the program consume about 100mA to 150mA so an average alkaline battery is able to power the raspberry pi for at least 10 hours.

Testing

Environment: Mobile Application Testing

Testing Strategy: Functional robust System testing

The following are the use cases for each page and the current testing results:

Login Page

Use Case 1 - correct username and password

→ returns a login successful message quickly conceded by Main Page

Use Case 2 - username entered with no password, button click "login"

→ Returns message 'Please enter User Id and Password'

Use Case 3 - password entered with no username, button click "login"

→ Returns message 'Please enter User Id and Password'

Use Case 4 - Neither a username or password were entered, button click "login"

→ Returns message 'Please enter User Id and Password'

Use Case 5 - username is entered with incorrect password

→ Returns message 'Invalid credentials'

Use Case 6 - device is not connected to ISU network or VPN'd in

→ Returns message 'Error in connection with SQL Server'

Use Case 7- 'Register' button click

→ Returns the register page

Use Case 8 - Click android device "back" button

→ Returns the previous page the user was on

Register Page

Use Case 1 - new username and a password entered, button click "register"

- Database table ActiveDirectory is updated with the new username and password
- Returns Add Device page

Use Case 2 - username entered with no password, button click "register"

- Returns message 'Please enter a new User Id and Password'

Use Case 3 - password entered with no username, button click "register"

- Returns message 'Please enter a new User Id and Password'

Use Case 4 - neither a username or password were entered

- Returns message 'Please enter a new User Id and Password'

Use Case 5 - an already existing username is entered

- Returns message 'That username is taken'

Use Case 6 - device is not connected to ISU network or VPN'd in

- Returns message 'Error in connection with SQL Server'

Use Case 7 - 'Back to Login' button click

- Returns the 'login' page

Use Case 8 - Click android device "back" button

- Returns the previous page the user was on

Main Page

Use Case 1 - Main page is generated by an account that has sessions

- Main page shows the gaming sessions that are directly related to the session id that was passed in

Use Case 2 - Main page is generated by an account that already exists but doesn't have any sessions

- Main page shows an empty page

Use Case 3 - Main page is generated by a new account

- Main page shows the gaming sessions that are directly related to the session id that was passed in

Use Case 4 - Main page is generated by an existing account that isn't connected to a hardware device

- Main page shows an empty page

Use Case 5 - User clicks on a session that has no keywords triggered

- Returns a blank "gaming session page"

Use Case 6 - User clicks on a session that has 1+ keywords triggered

- Returns the gaming session page

Use Case 7 - User clicks on 'Add/Edit a keyword' button

- Returns the "keyword" page

Use Case 8 - User clicks android device "back" button

- Returns the previous page the user was on

Keyword Page

Use Case 1 - The user types a keyword to be added to the list to be tracked

- The keyword is added to the list on the keyword page
- The keyword is added to the database.

Use Case 2 - The user selects a keyword to remove by pressing the delete button

- The keyword is removed from the list on the keyword page
- The keyword and respective user id entry is removed from the database

Use Case 3 - The user clicks the back button

- Returns the “main page”

Use Case 4 - User clicks android device “back” button

- Returns the previous page the user was on

Add a device Page

Use Case 1 - The user does not enter either the device code or the device name

- Returns message “Please enter Device code and Device name”

Use Case 2 - The user enters the device code but not the device name

- Returns message “Please enter Device code and Device name”

Use Case 3 - The user enters the device name but not the device code

- Returns message “Please enter Device code and Device name”

Use Case 4 - The user enters the device code of a device that already has a userid/name assigned to it

- Returns message “This device is already in use”

Use Case 5 - The user enters a device code that does not exist in the database

- Returns message “This device code is invalid”

Use Case 6 - The user enters a device code that exists in the database but does not have a userid or name assigned to it yet and a valid a device name

- Returns the main page

Use Case 7 - The user clicks on the text “Skip for now”

- Returns an empty main page

Show Device Page

Use Case 1 - A listview object (device name) is selected

- Finds the respective hardware ID to the device selected
- Returns the main page for that respective hardware ID

Use Case 2 - There are two different users with the same device name, i.e. “Sam’s Device”

- Returns the correct main page for the current user

Use Case 3 - A user names their 2+ devices the same

- Returns the first hardware id device in the database, which would be at random
- In the future, this should prompt the user that they have to name their devices differently

Use Case 4 - “Would you like to add a device?” text is selected

- Returns the “Add Device” page

Use Case 5 - The back button is pressed

→ Returns the previous page you were on (possibilities vary from login, and any page that has the action bar)

Gaming Entry Page

Use Case 1 - The message displayed has a keyword

Use Case 2 - The message displayed has a word with a confidence above 90%

Use Case 3 - The message displayed has a word with a confidence between 70-90%

Use Case 4 - The message displayed has a word with a confidence under 70%

Environment: Controller Attachment

Testing Strategy: Functionality Testing

Transmitting audio to speaker

The controller attachment is powered by 2 AA batteries. The batteries are connected to a 5V regulator to power the raspberry pi. The bluetooth transmitter is paired with the bluetooth module of the speaker. A headset with mic is connected to the input of the USB audio card. This test is verified the operation of the circuit.

Volume buttons

Music is played through the raspberry pi and when the buttons are pushed, the volume of the speaker will change or muted.

Power Consumption

The current of the Raspberry Pi Zero W when running without the desktop gui and peripherals is around 100 mA to 180 mA. However our Raspberry Pi Zero stopped functioning so we had to switch to a backup Raspberry Pi 3B+. When the Raspberry Pi 3B+ was running we keep having low voltage warning. This might be caused by the more power consuming processor of the 3B+. We believe that this problem will be solved by having a Raspberry Pi Zero.

Environment: Speaker

Testing Strategy: Functional Testing

Receiving Audio

The speaker circuit powered solely by a 3.7V rechargeable Li-ion battery and was connected via bluetooth to the audio output of a phone playing music. This test verified proper operation of the circuit as well as noise levels. Noise was present but fairly minimal.

A second test was performed using a connection to the bluetooth transmitter of the controller attachment. The passive noise level was much higher in this configuration and through testing of various setups such as connecting headphones to the output of the controller attachment, connecting a phone to the wireless speaker; the source of the noise increase seemed to come largely from the physical interface between the external soundcard of the controller attachment,

and the aux cord connecting it to the bluetooth transmitter. Physically compressing the soundcard's case decreased noise levels significantly, and the noise seemed to pulse in time with the operational indicator LED of the sound card. This leads us to believe that the problem will largely be solved if the sound card and bluetooth receiver were on a single PCB without an indicator LED, as would be likely in a final production design of this product.

Power Consumption

The current output by the battery was measured while connected via bluetooth to a phone playing music. The volume of the phone was at maximum. The circuits peak output was about 0.5A during noisier parts of the song, with an output between 0.1-0.15A during quieter parts of the song.

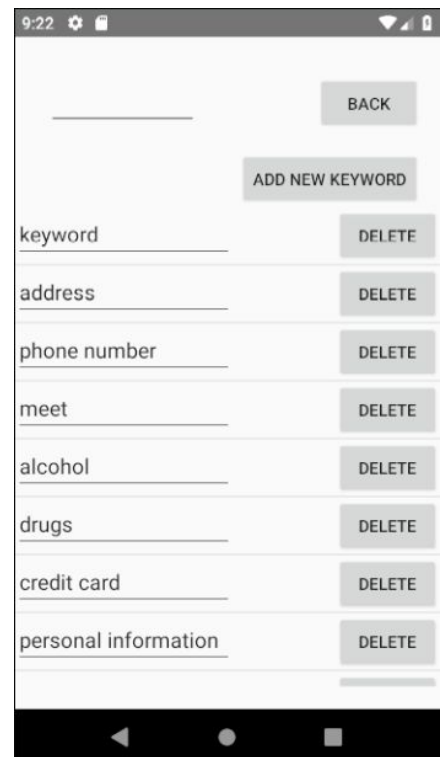
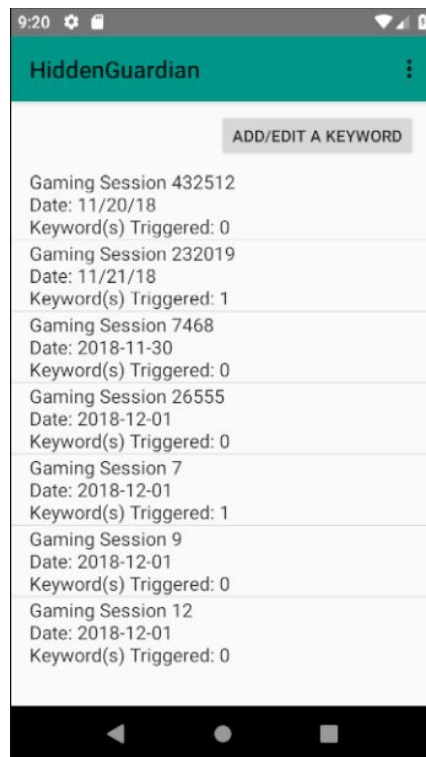
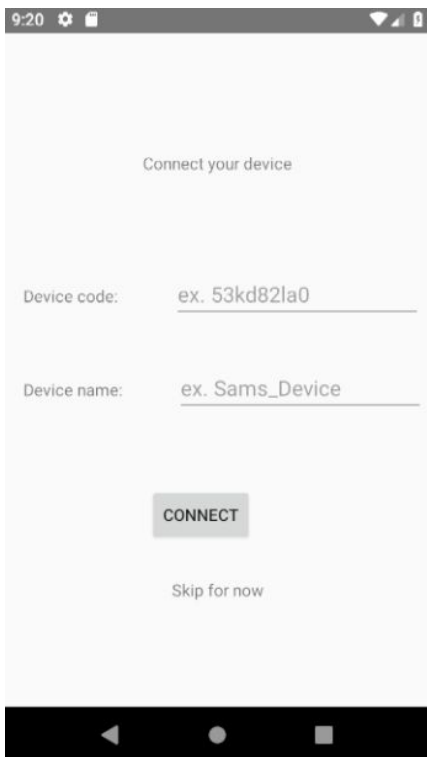
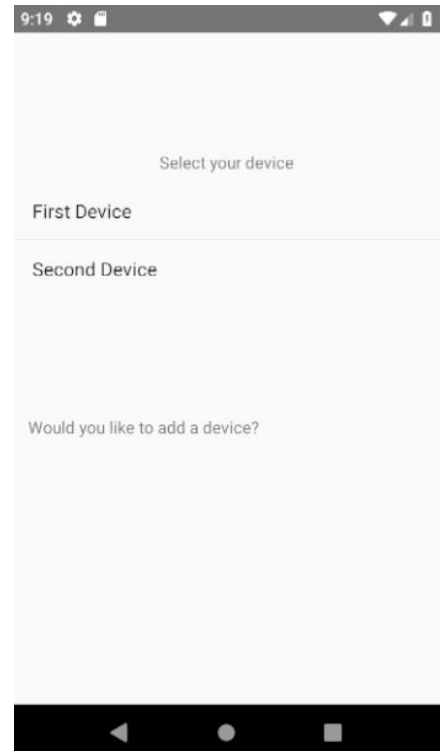
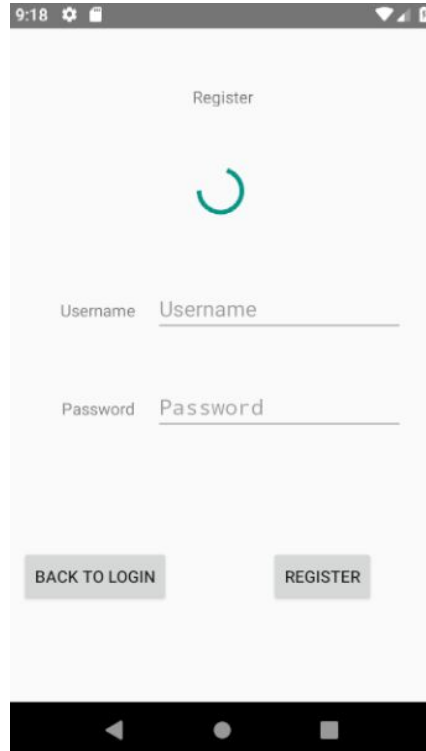
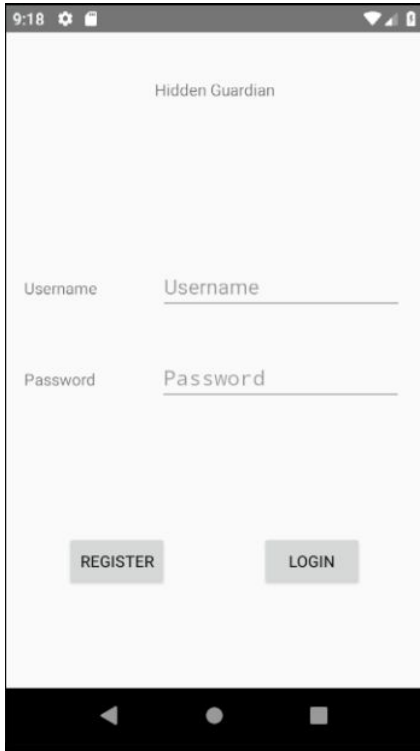
The current output of the battery was also measured with no paired bluetooth device. With no paired device the current reached a steady state between 0.09-0.095A.

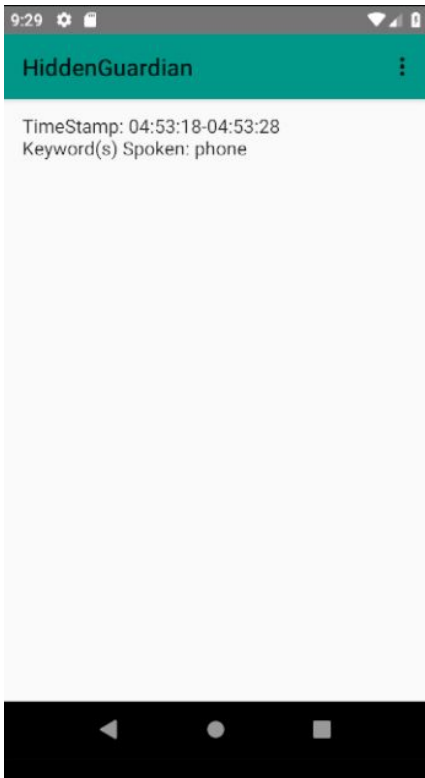
In both cases the current on power up spikes to about 1A, with a gradual decrease down to operating current levels as the bluetooth receiver finishes its power up sequence.

During these tests the battery voltage was at 3.87V. This gives a peak operating power of 1.9W (based on an operating current of 0.5A), and a lower end operating power of 0.39-0.58W (based on the operating current of 0.1-0.15A), and an unpaired power consumption of about 0.35-0.37W.

Given that the capacity of the Li-Ion battery is 2000mAh this gives a battery life of 4 hours up to 20 hours depending on the level of activity in the audio signal. As the intended usage of this speaker is for relaying the chat of an online game, it is expected that audio activity will be sporadic, with occasional conversations filled with spaces of little to no audio. Based on this assumption the battery life will likely be on the longer side of this range.

Appendix I: Mobile App Screen Images





Appendix II: Operational/User Manual

Operation Manual for Development

The following instructions assume you are developing on a Windows platform. For Mac/Linux users, find the respective commands for your OS.

Setting up the software environment:

1. Download Android Studios 3.2 at <https://developer.android.com/studio/>
2. Download JRE / JDK - JRE: 1.8.0_152 - release-1136-bo6 amd64
3. Download JVM - OpenJDK 64- Bit Server VM by JetBrains s.r.o
4. Download git at <http://git-scm.com/download/win>
5. Clone the repository in a git terminal using command:
→ \$ git clone <https://git.ece.iastate.edu/sd/sddec18-09>

Setting up the hardware:

1. Insert 2 AA batteries to the controller attachment
2. Plug in the controller attachment into the controller
3. The raspberry pi is loaded with raspbian, with the files controlling operation residing in the home directory, as well as a folder titled "Done" in the home directory.
4. The file mk1.sh is a shell script that controls the creation of the audio files,
5. The file transmit.sh is a shell script that controls outputting the audio to be transmit over bluetooth.
6. The file buttons.sh is a shell script that allows control over the volume levels and mute of the mic and controller audio via the hardware pushbuttons.
7. In the Done folder the file mqttclient.py is a python script that controls reformatting the audio files and transmitting them to the database.
8. The wireless speaker can be charged using a micro USB cable to the battery charging module
9. The schematic for the speaker and controller attachment are found under the Schematics subsection of Implementation Details above

Demo/Test the system:

1. Launch Android Studio
2. Set up hardware
3. Import and build the git project
4. Test the mobile application based on the use case/requirements in this document.

Proposed User Manual for Production Product


Thank you for purchasing Hidden Guardian. Hidden Guardian includes a controller attachment, speaker, and a free mobile application which you can find on Google Play.

To set up the controller attachment:

The controller attachment is powered by 2 AA batteries.

The controller attachment attaches to the bottom of an Xbox One controller at the base between the handle, plugging into the 3.5mm audio jack of the controller.

The first time you use the controller attachment you will need to connect it to WiFi using WPS. Locate the WPS button on your wireless router and press it. Within the next two minutes press

the WPS button on the controller attachment (It looks like this ).

After the first time you only need to turn the attachment on and it will connect to WiFi automatically.

To set up the speaker:

The speaker will automatically connect to the controller attachment via Bluetooth when both devices are turned on.

The speaker has a rechargeable battery that can be charged using a micro USB B cable.

Mobile Application Set Up:

After downloading the mobile application, it will prompt you to login.

As a new user, you will first need to create a new account by clicking the “Register” button. Once you register, you can connect your new device with your account.

The **Device Code** of your device is:

kq!!-dtF8iod-W3

Enter this when prompted and select a Device Name.

You have now connected your device with your account! You should see a blank page. This is where you will be able to view the user’s gaming sessions. On this page you will also be able to add they **keywords** you would like Hidden Guardian to listen for. This is customizable and can be edited at any time.

Mobile Application FAQ:

Where can I find the text conversations?

On the main page you will be able to select which gaming session you would like to view. From there, if keywords were triggered you will be able to view which keywords were triggered at which timestamp (in minute long intervals). Select which interval of text you would like to view based on the keywords.

What if I would like to see more than the minute interval?

We provide text context for any audio that triggers a keyword. Simply swipe left or right to see the minute of audio before and after.

What do the colors mean?

These are the confidence ratings of the audio. If there is feedback, slurred words, or an unclear connection, the confidence rating will show that. This is explained with:

High Confidence - 90-100%

Medium Confidence - 70-89%

Low Confidence - 0-60%


Can I view gaming sessions where no keywords are triggered?

No. You will be able to monitor all gaming sessions, but Hidden Guardian only shows you the text/audio conversations of gaming sessions that have triggered a keyword. We recommend refining your keywords if you would like to view more gaming sessions.

Can I add a device?

Yes! Guardians can have 2+ Hidden Guardians to differentiate between different users. Click on *insert image of app bar* and select "Add a Device". Enter the respective device code and your preferred device name and you will be connected to another Hidden Guardian.

How do I switch devices?

Click on  and select "Switch Devices". Then select the Device Name of your preferred Hidden Guardian and continue to use the app.

Appendix III: Alternate Designs

Our initial design included an Xbox background application that would parse through their chat logs in addition to the microphone recorder. We had created a background application and had accurately been able to convert speech to text in the application, but through further

testing we found that Microsoft does not allow third party applications to view in game or out of game chat logs due to privacy concerns.

This caused a major change as most of our project was dependent on this application thus far. We had to discuss this with our client and explain the roadblock and the two potential alternatives we could consider moving forward. One of the suggested designs is the design we have presented here in this document. The other option we suggested was to develop our own sample game and to write code that made it possible to send the audio data from the game to the Xbox background application we had already developed. After explaining the options to our client, she decided that our current design would be the best fit for her requirements and end goal. Moving forward, we readjusted our timeline and plan to account for this design change and set forth to work on the final design

On the database end, we had originally envisioned an sqlite database. However, we came across issues because sqlite would be local only to the phone app. Therefore, we would not have easily been able to connect and pull information from multiple account. This caused us to change to Microsoft SQL as we knew it would have all the remote capabilities we needed. Other options considered but not pursued due to unfamiliarity include: MongoDB, MySQL, and Oracle DB.

The software going onto the raspberry pi has changed a small amount throughout the design phase. Discussion on what what would actually be placed on the raspberry pi ended on a python script that is run on startup. We chose this over other languages due to the ease in making it run on startup as well as numerous python libraries regarding other parts of our project design.

We also considered using the VOIP protocol instead of MQTT for sending the messages as VOIP would have continuous connection. This was decided against for a few reasons. One was that we have a fairly slow server and we would not be able to reasonably test the protocol and know if we did something wrong. Further issues include unfamiliarity with the protocol that would likely lead to critical errors that would need strong testing to diagnose. Another issue with the VOIP protocol is registering the server itself as a listener would have trouble on the extensibility as ports could become clogged when several users would try to connect at the same time. MQTT resolves this issue because it queues up messages to ensure that data is not colliding with each other.

Another design change that we had was from using Javoix to Google Speech to Text API. At the end of our first semester, we had to quickly change our design plan due to our inability to have a console application, so from our minimal research we thought the Javoix Speech to Text conversion that could occur within the database would be the fastest and most efficient solution. However, given the time to research over the summer, we realized that the Google Speech to Text API would not be too slow and it is a much more developed and documented way to convert our speech to text. We also saw many beta features like word-enabled confidence ratings, multiple language support, and user identification that would be beneficial to our project.

Appendix IV: Future Suggestions & Additions

There were some features that we were not able to implement. However, with more time we would have liked to support the following:

- Notifications to the parent when a keyword is triggered
 - Currently we only check for keywords to be triggered when the mobile application is running, this would require the pi to ping the mobile application every time a keyword is triggered
- The ability to delete a gaming session
 - This would require the listview in the “main page” to be rewritten as a customizable listview so that there could be a button to delete the gaming session objects (similar to how it is implemented in the keyword page)
- Activities automatically being deleted every week
 - We started to work on this functionality and have this feature partially implemented and commented out in our code. This will be important for Hidden Guardian once it would be in production so to save phone memory and cost
- Addition of a mic on the controller attachment
 - This was originally in our plan, but we did not have time to implement it due to time constraints. The implementation of this is addressed in the controller attachment schematic.
- Another page that describes the keyword confidence color ratings so that the user can get that information from the app and not only the user manual
 - This page can just be linked via the toolbar on the main page, gaming session page and the gaming session entry page
- Audio file image clip at the top of the page that links to the audio file
 - This could be added to the toolbar in the gaming session entry page
 - The database table “Messages” would have an extra column that contains the file path to the audio link
 - When the image was clicked it would trigger the database to get that audio file path and a python script would be passed the audio file path and would return the audio clip that could be sent to the mobile application

Other features we had not planned to implement, but came to the realization would be feasible to implement based on Beta features provided by our Google Speech to Text API include:

Multiple Language Support

- Google speech to text has the functionality to support multiple languages

Identifying when a new user begins to speak

- Google speech to text has the functionality to identify when a new user speaks. This feature functions similarly to how the confidence ratings are output. This could help make the text more understandable, even though it could not define which user was who
- An example of how this would affect the message display would be:
 - User 1: "Hello team"
 - User 2: "Hi how are you?"
 - User 1: "I am good"
 - User 3: "Hello I just joined my name is sam"
 - User 2: "Hi Sam!"
- A potential added feature, upon further development, would be to add native recognition to look for ways users identify themselves, as demonstrated above and assign the name "Sam" to User 3.

Appendix V: Code

<https://git.ece.iastate.edu/sd/sddec18-09>