Technical Report: Speech Buddy

William Anderson, Kevin Dang, Sanjay Jerad

March 28th, 2017

Speech Buddy Project

Speech Buddy Github

Declaration of Joint Authorship

We, William Anderson, Kevin Dang and Sanjay Jerad, confirm that this work submitted for assessment is our own and is expressed in our own words.

William holds scrum master position. In charge of development at the back-end of the SpeechBuddy project.

Sanjay created the graphical user interface for the android application.

Kevin in charge of data storage for the project. Data stored in an external database.

Our team "Arrested Developers" agree that the Speech Buddy project is a group involvement, using every member's own ideas and knowledge throughout the entire duration of the build.

Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. Any outside knowledge we used for the creation of the project, will be cited down in the references at the bottom of the TR. We acknowledge any work used that does not belong to us and respect their proper ownership.

It is mandatory that all sources of information be acknowledged in the TR. Plagiarism is unethical, irresponsible and criminal.

Proposal

Summary

An intelligent voice interface that is able to listen and interpret what the user has spoken. Input will be translated into text and stored in a Database.

Problem

The problem this project solves is that it helps users take simple notes, such as a grocery list or small reminders for when you don't have a pen and paper available. This project will help solved problems where people forget an important detail or appointment, by storing what the user says into a readable text format.

Similar to Apples Siri and Microsoft's Cortana.

Description

As students in the Computer Engineering Technology program, we will be integrating the knowledge and skills we have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a Database as well as to a mobile device application. The internet connected hardware will include a custom PCB with sensors and actuators for speech recognition and voice recording. The Database will store what the user says in a readable format saved on a Database. The mobile device functionality will include storage for the recorded text and any reminders and will be further detailed in the mobile application proposal. we will be collaborating with the following company/department, we will not be collaborating with any companies at this moment. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project.

Background

There are several applications and hardware out there that utilize speech as input, such as Apple's Siri, Microsoft's Cortana or Amazon's Alexa. Some other hardware and software similar to our projects are Digital pens. This piece of hardware records what users write down as input, and saves it in a text format on the computer. In this era there is not much mention of using a person's voice as input, and has not been a part a person's daily lives. You don't see people talking to their phone or microphone everywhere you look.

We have searched for prior art via Humber's IEEE subscription selecting "My Subscribed Content" and have found and read the below articles, which provides insight into similar efforts.

The first article contains information related to text-to-speech output in technology. (Karabetsos, Tsiakoulis, Chalamandaris, & Raptis, 2009)

The second article's information is about discriminating between vocal sounds and environment sounds. (Yuan-Yuan, Xue, & Bin, 2004)

The third article relates to the behaviour of speech with service robots. (Wang, Leung, Kurian, Kim, & Yoon, 2010)

In the Computer Engineering Technology program we have learned about the following topics from the respective relevant courses:

- Java Docs from CENG 212 Programming Techniques In Java,
- Construction of circuits from CENG 215 Digital And Interfacing Systems,
- Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
- Micro computing from CENG 252 Embedded Systems,
- SQL from CENG 254 Database With Java,
- Web access of Databases from CENG 256 Internet Scripting; and,
- Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable us to build the subsystems and integrate them together as my capstone project.

Methodology and Schedule

Phase 1 Hardware build.

Phase 2 System integration.

Phase 3 Demonstration to future employers.

Phase 1 Hardware build

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

Phase 2 System integration

The system integration will be completed in the Winter term.

Phase 3 Demonstration to future employers

This project will showcase the knowledge and skills that we have learned to potential employers.

The tables below provide rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

Labour Estimates	**Hrs**	**Notes**
Phase 1		
Writing proposal.	9	Tech identification quiz.
Creating project schedule. Initial project team meeting.	9	Proposal due.
Creating budget. Status Meeting.	9	Project Schedule due.
Acquiring components and writing progress report.	9	Budget due.

Mechanical assembly and writing progress report. Status Meeting.	9	Progress Report due (components acquired milestone).
PCB fabrication.	9	Progress Report due (Mechanical Assembly milestone).
Interface wiring, Placard design, Status Meeting.	9	PCB Due (power up milestone).
Preparing for demonstration.	9	Placard due.
Writing progress report and demonstrating project.	9	Progress Report due (Demonstrations at Open House Saturday, November 7, 2015 from 10 a.m. - 2 p.m.).
Editing build video.	9	Peer grading of demonstrations due.
Incorporation of feedback from demonstration and writing progress report. Status Meeting.	9	30 second build video due.
Practice presentations	9	Progress Report due.
1st round of Presentations, Collaborators present.	9	Presentation PowerPoint file due.
2nd round of Presentations	9	Build instructions up due.
Project videos, Status Meeting.	9	30 second script due.
Phase 1 Total	**135**	
Phase 2		
Meet with collaborators	9	Status Meeting
Initial integration.	9	Progress Report
Meet with collaborators	9	Status Meeting
Testing.	9	Progress Report
Meet with collaborators	9	Status Meeting
Meet with collaborators	9	Status Meeting
Incorporation of feedback.	9	Progress Report
Meet with collaborators	9	Status Meeting

Testing.	9	Progress Report
Meet with collaborators	9	Status Meeting
Prepare for demonstration.	9	Progress Report
Complete presentation.	9	Demonstration at Open House Saturday, April 9, 2016 10 a.m. to 2 p.m.
Complete final report. 1st round of Presentations.	9	Presentation PowerPoint file due.
Write video script. 2nd round of Presentations, delivery of project.	9	Final written report including final budget and record of expenditures, covering both this semester and the previous semester.
Project videos.	9	Video script due
Phase 2 Total	**135**	
Phase 3		
Interviews	TBD	
Phase 3 Total	**TBD**	
Material Estimates	**Cost**	**Notes**
Phase 1		
A microcomputer composed of a quad-core Windows 10 IoT core compatible Broadcom BCM2836 SoC with a 900MHz Application ARM Cortex-A7 32 bit RISC v7-A processor core stacked under 1GB of 450MHz SDRAM, 10/100 Mbit/s Ethernet, GPIO, UART, I²C bus, SPI bus, 8 GB of Secure Digital storage, a power supply, and a USB Wi-Fi adaptor.	\$120.00	Amazon
Microphone	\$20.00	
Speaker	\$28.99	

CENG Parts Kit	\$110.00
J206 Parts	\$40.00
Project Case	\$13.00
Phase 1 Total	**>\$331.99**
Phase 2	
Materials to improve functionality, fit, and finish of project.	N/A
Phase 2 Total	**TBD**
Phase 3	
Off campus colocation	<\$100.00
Shipping	TBD
Tax	TBD
Duty	TBD
Phase 3 Total	

Concluding Remarks

This proposal presents a plan for providing an IoT solution for better planning and helps people set reminders of important details. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects. I request approval of this project.

Abstract

This report contents information specifying a Voice interface build project. This project hopes to create a fully functioning Voice interface that will complete simple tasks given to it via speech detect. Similar to Apples Siri and Microsoft's Cortana. Information in regards to the build material and cost of this project are listed in this report. Material such as a raspberry Pi, USB microphone and mini speakers were used, in total the SpeechBuddy project's estimated cost is $230. SpeechBuddy will help people with better planning and organization, letting them set info about important details. This project will roughly take up to 8 months to complete. Data is stored in an external database hosted on Firebase. The android application is a simply structure that displays the list the user created via voice input.

Table of Contents

7.2. Mobile Application Specifications

7.2.1 GUI Specifications

7.2.2 Database Integration

7.2.3 Sample User Cases

7.2.4 Application Work Contributions

7.3. Additional Web Specifications

7.3.1 Amazon Voice Services

7.3.2 Firebase Hosting

7.3.3 DynamoDB Database

7.3.4 Lambda Web Service

7.4 Construction

7.4.1 Construction Introduction

7.4.3 Alexa Skill

7.4.2 Creating The Database

7.4.4 Adding

7.4.5 Deleting

7.4.6 Error Checking

7.4.7 Project Case

7.4.8 Android Application

7.5 Build Instructions

7.5.1 Build Introduction

7.5.2 Build Time

7.5.3 Mechanical Assembly

7.5.4 Software Setup

7.5.5 Alexa Skill with Lambda

7.5.6 Database Setup

7.5.7 SpeechBuddy Android Application Setup Instructions

7.5.7.1 Additional Amazon Account Configuration for Android Application

7.5.7.2 Cognito Identity

7.5.7.3 IAM Roles

7.5.7.4 Customizable/Local Installation

7.5.7.5 Play Store Installation

7.5.7.6 The Speech Buddy Application

7.5.8 Power Up and Testing

7.5.8.1 Speech Buddy Application Directions and Controls

7.5.8.2 The Toolbar

7.5.8.2 List Screen

7.5.9 Colour Tutorial Setup

7.6 Progress Reports

7.6.1 Previous Year Work 2016

7.6.2 Status Report

7.6.3 Integration Report

7.6.4 Troubleshoot Report

8. Conclusion

9. Recommendations

10. References

Illustrations and Diagrams

Figure 1:

The casing of the Speech Buddy, laser cut design can be found on our project website.

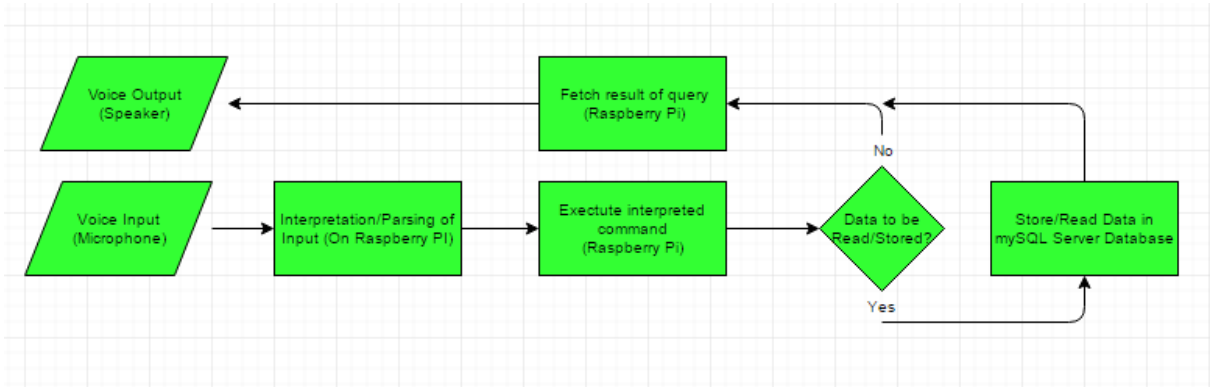

Figure 2:

System diagram of the project, describing general structure of how Speech Buddy works.

Introduction

Objective

The objective of this project is to create a usable voice interface that will help people with organization and planning. Having the ability to help manage tasks and improve the users planning and scheduling efficiently. Similar to Apple's Siri and Microsoft's Cortana. In this current age of time a mobile phone is carried everywhere and used 24/7. We want to utilize this aspect to create a unique interface with a virtual voice to improve the users planning skills. If a piece of technology is constantly with people the majority of the time, why not use that technology in an effort to help them.

Technical Problem

The problem solved by this project is that it helps users take simple notes, such as a grocery list or small reminders for when you don't have a pen and paper available. This project will help solve problems where people forget an important detail or appointment, by storing what the user says into a readable text format. Having the ability to bring up that stored information anywhere the user is current located.

Some problems that were encountered in constructing the Speech Buddy, was trying to convert spoken words into text on the Raspberry Pi. We tackled this problem by first utilizing the "espeak" library on the Raspberry Pi. This did not provide enough efficiency, until we discovered Amazon's Alexa voice service. This voice service allowed us to create specific tasks for the voice to do, making it easier to program speech conversion, and database storage. Another problem was connectivity, the Alexa voice service requires internet access, we have a wired connection using an Ethernet cable. Portability is something to look further in to.

Project Description

Database Specifications

Database Type

Firebase is the Database type that is used for Speech Buddy. Firebase is a JSON style Database that is organized via encrypted keys. It is incorporated directly in android studio for ease of use with mobile applications wanting to store data online.

Database Tables

There are three tables that are used in the Firebase.

1. Users
2. ListNames
3. ItemNames

Users is the table in which all users who sign up for our Speech Buddy Application are stored. They are given a key as a unique identifier.

ListNames is the table in which the names of list the user created are stored as strings. Each Users own Lists that they have created will be unique to them via an identification key within the table.

ItemNames is the table in which the items under each list are stored. Each Item is unique to the list name it was created under and the user that created it via an identification key within the table.

Database Alteration via User Case Examples

User case: The User Adds a list named “groceries” to their speech buddy.

Consequences: A string element “groceries” and key unique to the user are generated in a new row under the ListNames table.

User case: The User Creates an Account with username steve@abc.com and password “123”

Consequences: A string element (in mandatory email format) “steve@abc.com”, a string password “123” and key unique to the newly made user are generated in a new row under the Users Table. All data manipulation under their account will use this identifier, and they must use these credentials to access the application.

User case: The User Adds an Item named “rice” to their “groceries” list.

Consequences: a string element “rice” and key unique to the user are generated in a new row under the ListNames table.

Mobile Application Specifications

Graphical User Interface Specifications

Login Activity: The login screen of the application is the first presented upon launching the application. The user may enter already existing credentials into EditText fields and login, or may choose to go to the sign up activity, both navigations via Buttons.

Sign Up Activity: The sign up screen of the application can be accessed from the login menu. It has two EditText fields for new credentials to be added, which upon submitting via Button will be verified by the username and password guidelines and added to the user database. Incorrect EditText entries will be communicated via Toast message asking for re-entry.

List Activity: The list activity of the application is comprised of a single list, formatted appropriately, displayed across the width of the screen in both portrait and landscape orientation. It loads it’s elements from the database dynamically. If the list length exceeds the screen size, the list becomes scrollable. The top right has the dropdown navigation menu, and add list/delete list buttons.

Items Activity: The item activity of the application is comprised of a single list, formatted appropriately, displayed across the width of the screen in both portrait and landscape orientation. It loads it’s elements from the database dynamically. The elements it loads are specific to the parent list and user selected. If the list length exceeds the screen size, the list becomes scrollable. The top right has the dropdown navigation menu, and add item/item list buttons.

About Application Activity: A simple activity describing the purpose and contributors of the Speech Buddy Application via formatted TextViews.

Dropdown Navigation Menu: In the top right corner of post-login activities. Allows the user to Navigate to list pages, the about page, and logout. Also has an option to clear all data from the current users records to start anew.

Additional Note: For ease of use, users may also press and hold list/item elements to delete them. Any deleted list will also delete it's child items.

Database Integration

The Speech Buddy Application uses the integrated Firebase libraries in Android Studio to connect to the database. Once connected online, it actively retrieves data and adds dynamically updates the lists displayed within the application. All new elements added to the users lists and items are independently added to the database as soon as they are entered for optimal up to date accuracy within the database.

User Cases

Test case: 1

Creating tables

Purpose: Purpose is to observe if our application has created a table that contains the columns. Columns we need are the ID and the name of all list the user created.

Precondition:

Steps:

Expected Result: Table should be created with an ID that auto increments, and content the names for each column

---

- Have created a database
- Make the columns need for table (Strings)
- CREATE table statement

1. Create a database using SQLite
2. Make column names
3. Create a String for the CREATE table
4. Run the statement using database.execSQL()
5. Go to where database saved and check if it is there
6. Open database see if table is there

Test case: 2

Insert data into Table

Purpose: Purpose is to observe if our application can allow users to enter data that will be stored in a database

Precondition:

Steps:

Expected Result: Table should be created with an ID that auto increments and the String the user entered in the EditTextView for the correct column

---

- Have created a database
- Have the data needed to insert into table
- ID (auto incremented)
- List name (user enters this)

1. Get data to enter (list name) from EditTextView
2. Save data in table made variables
3. Insert the data into the table one row at a time
4. Go to where database saved and check if it is there
5. Open database see if table is there

Test case: 3

Delete data into Table

Purpose: Purpose is to let users remove any old list they have or mistakes they have made

Precondition:

Steps:

Expected Result: The data enter by the user should be deleted from the table

---

- Have created a database
- Have the data you want to delete needed in the table

1. Get the name of the list/data user wants to delete from table
2. Create a DELETE statement using that name and the table name
3. Run the statement using database.delete()
4. Open database and table and see result

Test case: 4

Delete all data on database

Purpose: Remove all the data in the database if user wants a fresh restart with no data

Precondition:

Steps:

Expected Result: Database will contain tables with no data

- Have created a database
- Have tables with data in them

1. Go into setting activity and click button
2. Confirm yes or no
3. Run delete Statement for all data in columns

4. Tables will still exist

Test case: 5

Blank inputs

Purpose: See what will be stored in the database if user enters blank items

Precondition:

Steps:

Expected Result: A blank entry will be created in the database

---

- Have created a database
- Text box for user input

1. Go into mylists
2. Add a list will blank information
3. Open database to see result

Test case: 6

Delete using a blank input

Purpose: Observe what when user enters Blank, see how it affects the database

Precondition:

Steps:

Expected Result: Will look for a blank data in table and remove it, if it cannot find data do nothing

---

- Have created a database
- Text box for user input

1. Go into mylists
2. Delete a list will blank information
3. Open database to see result

Test case: 7

User hits the back key on application

Purpose: To test if the application will close if user clicks the back key

Precondition:

Steps:

Expected Result: Application should prompt user if they want to exit the application

---

- Run the application
- VM of android phone

1. Run the application
2. clicks the back key

Test case: 8

Delete using a blank input

Purpose: Observe what when user enters Blank, see how it affects the database

Precondition:

Steps:

Expected Result: Will look for a blank data in table and remove it, if it cannot find data do nothing

---

- Have created a database
- Text box for user input

1. Go into mylists
2. Delete a list will blank information
3. Open database to see result

Test case: 9

Enter a list with the same name as another

Purpose: Test to see if database will create two data entries that are the same

Precondition:

Steps:

Expected Result: If you click any of the listnames with the same name, the item data is the same for all of them

---

- Have created a database
- Enter data for both list and item tables

1. Enter 2 entries with the same listname
2. Click one of them and enter an item

Application Work Contributions

William: The acting scrum master of the project. William is in charge of back-end development including Java and using libraries and API's. He manages the overall architecture and functionality of the project.

Sanjay: In charge of the Graphical User Interface and User experience. Different Graphical interfaces for both mobile phones and tablets. Designs layouts and recommends functionality to William to be implemented.

Kevin: In charge of data storage, manipulation and maintenance. Created the Firebase database, as well as its internal structure and breakdown to be connected to via Firebase Java libraries.

Additional Web Specifications

Amazon Voice Services

The Speech Buddy Hardware utilizes Amazon Voice Libraries for voice input to be added to the database. In addition to the JavaScript classes built for Speech Buddy, Amazon Voice Services offers basic user

searches via an online search engine and can answer simple questions. Use of Amazon Alexa "skill" creation to make use of the libraries to connect to the external Firebase database.

Firebase Database

The Speech Buddy Hardware and application will be utilizing Firebase databases for data storage. The service offers free (to a certain amount of traffic) data hosting and an appropriate size and speed for Speech Buddy's Requirements.

DynamoDB Database

an Amazon Web Service that allows for the create of tables for data storage, and manipulation. Service is free only if data being written to it is not over a certain capacity ans size. Belongs to the same service and company as the Alexa voice service, allowing for easy integration of hardware and software.

Lambda Web Service

Lambda is a service on the Amazon Web Service website for creating Alexa Skills. Using this we can create custom functionalities to install on to the Raspberry Pi. Service allows for multiple code languages, such as JavaScript, Python, NodeJS and many others. The variety in coding languages can allow for coding in a language we are familiar in.

Construction

Construction Introduction

When working on the speech Buddy project, we planned to incorporate an ability to create lists at the users' request. This functionality in the voice interface will allow users to improve their organizational abilities, and help increase their planning, as well as scheduling skills. Throughout the duration of our build, we decided to use the Amazon Voice service, Alexa. This service provided the voice we needed for speech buddy. This voice service included basic functionally such as, being able to get the time, date, google information, by asking questions.

By using Alexa skills we were able to create specific functionalities for the speech Buddy. This provided an Integrated Development Environment (IDE) to begin coding for the List creating skill we would be developing.

The data being inputted by the user was kept in external database online, we incorporated the Amazon Web Service (AWS), DynamoDB database. Using this service with the Alexa skill we were creating, allowed us to take user voice input, and store that information online in tables.

Alexa Skill

On the Amazon Web Services (AWS) we used a service known as, Lambda. This service was the foundation of the List creating skill. Providing an Integrated Development Environment (IDE) to begin coding. Following several tutorials found online for using the Lambda service we managed to understand and learn how to utilize Lambda for our Alexa Skill. The tutorials used are located here and here. Creating the Lambda Skill was done, but the difficult portion was trying to actually install the Lambda skill on to the Voice service Alexa. Following the tutorial we went to our Amazon Developer account to add a skill, this required an intent schema and a sample utterances. The intent schema and sample utterances provide the Alexa skill code with knowledge on what section of code to run, when a specific phrase or sentence is heard. These two requirements proved difficult, needing a complete code to help use understand and learn about them. For this we used a basic Lambda skill that the AWS lambda service provided on the website. Using the sample skill, Color (this is discussed late the the Technical Report), we were able to install skills onto the Raspberry Pi with Alexa.

We started coding the skill using Python. Connection to the database proved simple with just 3 lines of code, and adding permissions for the DynamoDB web service. Main goal was developing code for adding and deleting data in the DynamoDB database. Adding and deleting codes were completed in methods, so they can be easily utilized anywhere in the code.

Creating the Database

Creation of the database was simple on the AWS website, navigating the DynamoDB page to create two separate tables. A table called, ListNames with a column for the Name of a list. A table called, ItemNames

with the columns for, Item name and list name. Using a second table with two columns including item name and list name, allowed for the minimization of tables we needed. This also allowed us to reference different items depending on the list the user placed it in. With this we created an external database to house the data users inputted.

Connection to the database was required for us to manipulate the values users inputted. Both the hardware and software portions of the Speech Buddy project was to be connect to the same external database, DynamoDB, allowing for both pieces to be fully integrated together. The Alexa Skill we were implementing requires the use of the Lambda web Service. This web service utilized a policy will every skill created. The policy allows for other web services, such as DynamoDB to access the skill and also grant permissions to the person using the Alexa skill. To connect to skill to the online database, we included the DynamoDB permissions to the policy. To fully utilize the ability of the DynamoDB database we imported the "boto3" library in the Alexa skill python code. This library and class allowed us to use built-in functions to access and manipulate the tables located in the database.

Adding

Code for adding user input required many things. An entire in the intent schema, utterances, and code were required. Tutorials mentioned above included sample code of adding information into the database. We developed our own Python code for adding data, using the sample code as a base.

Code for Adding a List:

```
def insert_ListNames(listName):
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('ListNames')


    response = table.put_item(Item={
            'NameId' : listName
        })
    return None
```

This code required an entire in intent schema, so the skill was aware when to run the ADDING code. The

intent schema would run a specific part of the code when the user inputs a particular phrase or sentence specified in the sample utterances. For example, the user might say "Speech Buddy add {ListName} list". The "{ListName}" being a variable that the user says, such as "groceries", "workout", or "exams".

Testing this ADDING code proved successful on the Raspberry Pi as well as on the Amazon develop website.

Deleting

The Deleting code was simple after completing the ADDING functionality of the code. Again requirements in both the intent schema and sample utterances was needed.

Code for Deleting a List:

```
def delete_listNames(listName):
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('ListNames')


    response = table.delete_item(
        Key={
            'NameId' : listName
        })
    return None
```

Error Checking

Main goal of developing ADDING and DELETING code was completed. Error checking was next. Developing an error check when adding and deleting in the database was needed. The Adding and Deleting methods of code was only to be run, when the user specifies input that is actual in the DynamoDB database. If values are not found or already exist in the database are found, then error messages will appear to let the user know. Error checking code was develop for both the List table and the Item table. Code created looks through each table for a specified input. And returns a true or false if that input is found or not.

List Error Check:

```
def check_ListName(listName):

    checkName = ""
    nameCount = 0
    isThere = "false"

    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('ListNames')

    response = table.scan()

    nameCount == len(response['Items'])
    for index, item in enumerate(response['Items']):
        checkName = item['NameId']
        if checkName == listName :
            isThere = "true"

    return isThere
```

Item Error Check:

```
def check_ItemName(itemName, listName):

    checkItem = ""
    checkList = ""
    nameCount = 0
    isThere = "false"

    dynamodb = boto3.resource('dynamodb')
```

```
table = dynamodb.Table('ItemName')


response = table.scan()


nameCount == len(response['Items'])
for index, item in enumerate(response['Items']):
    checkItem = item['ItemId']
    checkList = item['ListName']
    if (checkItem == itemName) and (checkList == listName):
        isThere = "true"


return isThere
```

Project Case

The Speech Buddy contains some hardware components such as the Raspberry Pi, mini USB microphone, and a mini speaker. A case was built to house all these components to improve the overall appearance of the Speech Buddy. The case was laser cut using a black acrylic plastic. An acrylic plastic was used as the material for the case because of its cost and weight being low. The casing provides the necessary openings for the mini speaker and microphone, as well as the Ethernet port for computer connections. The layout of each side was done on a case making software online, www.makercase.com. The software Coral Draw can also be used as well, we found that makercase.com was more helpful and easier to use from our perspective.

Android Application

The Speech buddy project connections to an Android Application. The application displays information that the user adds to the DynamoDB database. The data on the Application is designed to display the data dynamical as the user inputs a new value. The application connects to the Amazon Web Service, DynamDB. Having the ability to read and write data to the external online database. The code or the android application for adding and deleting data is similar the python code versions done on the Alexa

skill, expect done using the programming language Java. Creation of the Android Application is done in the Integrated Development (IDE), Android Studio.

Build Instructions

Build Introduction

Speech Buddy is a voice interface intended to make storage and manipulation of data easier, as well as perform simple various tasks via voice commands. It is intended to make simple everyday tasks, such as adding items to your calendar, easier, as well as inquiries of online information.

System Diagram:

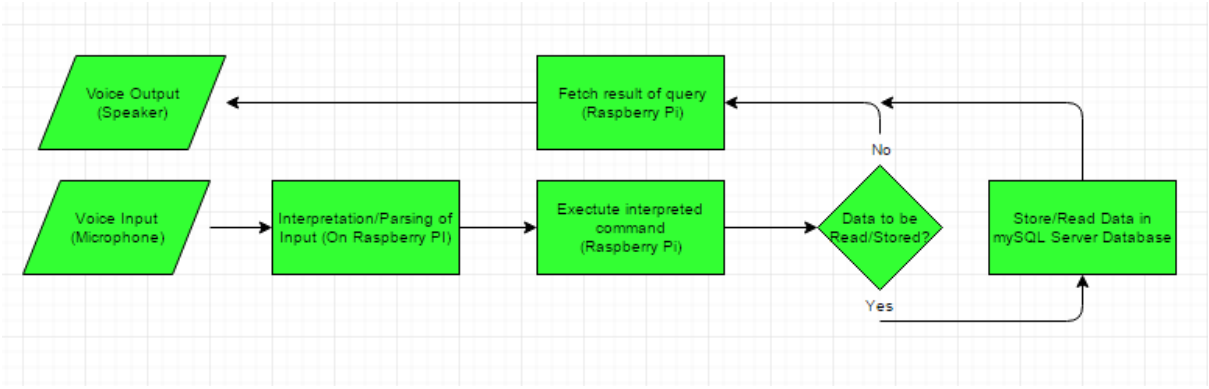

Figure 3:

Build Time

Assuming you have to do no research as you are following our pre-made build method, The approximate time it will take to build a working Speech Buddy is about four to five hours, assuming parts delivery time is not included.

Mechanical Assembly

Setup your raspberry Pi by following the build instruction included in your kit. Connect your external speaker into the pi's 3.5 mm jack and the USB microphone into a USB 2.0 slot. Use acrylic cement to

bond case together. The file for the case design can be found on the Speech Buddy website here, the laser cut design is called “casemodel(10).json”. If you intent to put Speech Buddy permanently in it’s case, it is important to wire the power cord through the designated power slot in the case before sealing the box.

Software Setup

Ensure you have a fresh copy of Rasbian Jesse installed on your Micro SD card for your raspberry pi operating system. Open a terminal on your Pi and type in the command ”sudo alsamixer”. Select Advanced Options, then Audio Options. Set to force Output through 3.5mm jack.

Speech Buddy uses Amazon Voice Services (AVS). Make sure you have an amazon developer account before continuing, then follow the voice services installation guide provided here:

Amazon Voice Setup (Do Not install the wake word activation).

Base on the AVS link provided, the setup is as follows:

First login to your amazon developer account and go to “Alexa”. Create a security profile and register a product, when completed this will create a ProductID, ClientID, and ClientSecret. Remember these values for later. Throughout the Voice Setup it will need information for the “web setting” in the Security profile. Enter “https://localhost:3000” for the Allowed Origins, and “https://localhost:3000/authresponse” for the Allowed Return URLs, under “web settings”.

On the Raspberry Pi, clone the AVS:

```
git clone https://github.com/alexa/alexa-avs-sample-app.git
```

Navigate the directory created, and edit the “automated_install.sh” and add the ProductID, ClientID, and ClientSecret you got when creating the security profile, to the file. Run this file to install AVS on the Pi, this will roughly take 30 minutes to install AVS.

Alexa Skill with Lamdba

Login to Amazon Web Services (AWS) and go the the console, and search for “Lambda”. This service will allow use to create a skill for our Alexa. When in the Lambda page go ahead and create a Lambda

function. For the blueprint, click on a blank function. Click on the broken box on the screen and select "Alexa Skills Kit" after hit next. Enter the name and description you want to use, they are not important. Select "Python 2.7" for the Runtime, this is the coding language that the Alexa skill will be using. The actual code of the Speech Buddy skill will be located on our website here, copy the code into the "Lambda function code" section. Moving onto the next portion is setting up the handler and role. Leave the Handler name as is, and choose "Create a custom role" for the Role. This will open up a new tab, select "Create a new IAM Role" for IAM Role and enter a name for it, there is further setup for this but that will be discussed later. After this click allow on the bottom right, this will take you back to configuring the function. Skip the Advanced settings and hit next, hit Create Function.

Take note of the ARN on the top right when you finished creating the function. this is important for later so remember it. There you go you have created an Alexa skill for the Speech Buddy, but you are not done yet. You have created the skill, but still need to connect to the actual Alexa itself.

To connect the Alexa Skill:

Login to your Amazon developer account and click on ALEXA tab. Click "Alexa Skills Kit", you will see a button on the top right, "Add a New Skill" click that. Leave everything as it is and just enter a Name and Invocation Name for the skill. For the interaction Model the Intent Schema and Sample Utterances can be found on our website here, copy them into the Models. Now the important part, in the Configuration, check the "AWS Lambda ARN", after check "North America". Enter the ARN code that was generated when creating the Lambda function.

It is complete now you have connected the Lambda Skill to the Alexa.

As discussed before there is further setup required for the IAM Role. To do this go to the console page on your AWS account, and search for "IAM". When in the IAM page click the "Roles" tab on the left hand side. Select the Role you are using for the Lambda function after selecting, click on the button "Attach Policy". Find the "AmazonDynamoDBFullAccess" and add this Policy to the Role. This Policy will allow for the Lamdba function to access another AWS service, DynamoDB, we will be using this service to store information on a database. The setup for the database will be discussed later in the Build Insrtuctions.

Database Setup

To set up the database for the Alexa skill use this link, database setup. The link is tailored to the owners own project, so using this as a reference guide you can setup the database for your own specifications. Setup is as follows:

The Speech Buddy uses an online database hosted by Amazon Web Services (AWS), called DynamoDB. To use this service you will be required to have a AWS account. When you login into your AWS account make sure your location on the top right is “North Virginia”, if it is not, change it to this. Some services on the AWS are only available in this area. Go to the console page and search for “DynamoDB”.

When you are in the DynamoDB page click on the “Create table” button. Enter the name of the table you would like to use, in our case “ListNames”, and enter a primary key. The primary key will be the first column of the table, enter “NameId”.

After entering the information for table name and primary key click “create” on the bottom right. Give some time for the table to create, and you are done.

To connect the Alexa skill to the DynamoDB database you will need to enter the following code into the lambda function:

```
import boto3


dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('table_name_here')
```

SpeechBuddy Android Application Setup Instructions

Additional Amazon Account Configuration for Android Application

Cognito Identity

There are some additional setup instructions you must follow in the configuration of your Amazon Web Services Account.

To use AWS services in your Speech Buddy Application, you must obtain AWS Credentials using Amazon Cognito Identity as your credential provider. Using a credentials provider allows you to access AWS services without having to embed your private credentials in your application. This also allows you to set permissions to control which AWS services your users have access to.

The identities of your application's users are stored and managed by an identity pool, which is a store of user identity data specific to your account. Every identity pool has roles that specify which AWS resources your users can access. Typically, a developer will use one identity pool per application. For more information on identity pools, see the Cognito Developer Guide.

To create an identity pool for your application:

1. Log in to the Cognito Console and click Manage Federated Identities, then Create new identity pool.
2. Enter a name for your Identity Pool and check the checkbox to enable access to unauthenticated identities. Click Create Pool to create your identity pool.
3. Click Allow to create the roles with access to your new identity pool.

The next page displays code that creates a credentials provider so you can easily integrate Cognito Identity in your Android application.

For more information about Amazon Cognito Identity, check out http://docs.aws.amazon.com/cognito/latest/developerguide/identity-pools.html

In Addition to this, you must create a policy within your AWS account that will allow the Speech Buddy Application to connect to the Dynamo Database securely.

IAM Roles

To use DynamoDB within the Speech Buddy Application, you must set the correct permissions. The following IAM policy allows the user to perform the creation, deletion and modification of list/item resources identified by ARN.

The policy you must policy you must create is:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem",
                "dynamodb:BatchWriteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:940271869439:table/ListNames"
            ]
        }
    ]
}
```

Apply this policy to the unauthenticated role assigned to your Cognito identity pool, replacing the Resource values with the correct ARN for your DynamoDB table:

1. Log in to the IAM Console.
2. Select Roles and select the ”Unauth” role that Cognito created for you.
3. Click Attach Role Policy.
4. Select Custom Policy and click Select.
5. Copy the policy above into the policy editing box.
6. Click Apply Policy.

For more information on IAM policies, check out http://docs.aws.amazon.com/IAM/latest/UserGuide/access/_policies.html

Customizable/Local Installation

In order to use the Speech Buddy application, you must have Android Studio installed on your machine, which can be found at https://developer.android.com/studio/index.html. During Installation Select all default settings and ensure Google Play Services for SDK 23+ are installed. (This can be done during Installation, where the installer provides and initial list of optional plug-ins to install).

Make sure once Android Studio is installed that is updated to the most recent version. This can be done by accepting the prompts upon start up or following the instructions found at https://developer.android.com/studio/intro/update.html

Android Studio Requires the Java Development Kit to function. This can be installed from http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html/.Follow All default paths and installation instructions.

Once the JDK and Android Studio are Installed and up to date, Create a new Project In Android Studio By Going to File > New Project. Set the Minimum SDK version to 17 (KitKat). Choose any project name and specify the package name as “william.anderson”. Do not create any default activities.

After the Project has been made, go to www.github.com/willcodyanderson/SpeechBuddyProject and download the repository. Unzip the repository in your downloads folder. Now, in Android Studio, In your newly create project, go to File> Add Module > Browse, and select the extracted directory downloaded from github.

After the module has been added to your project, go to the project level build.gradle that will be found under the first expansion of the project directory. Add the following line of code: classpath ’com.google.gms:google-services:3.0.0’. Save the File.

If you have knowledge of any Android Programming, you may now modify and adjust the application in your virtual environment as you wish. If not, continue to the next instruction.

Enable Developer Options on the intended Android Device. This is typically done by going to Set-

tings and repetitively tapping the “Build Number” Option several times. For further details, see www.androidcentral.com/how-enable-developer-settings-android-42.

Once developer settings are enabled on your device, the developer options tab will be displayed under your settings tab. Enter it and select “Enable USB Debugging”. This allows APK’s to be downloaded onto your device from Android Studio.

Connect your device to your computer that is running Android Studio. In Android Studio, within the Speech Buddy Project, select Run > YourDeviceName. This will install the application locally to your device. You may now disconnect the device from your computer and use the application freely as you choose. Keep in mind the Speech Buddy Application requires online access to function.

Play Store Installation

If you have no wish to modify the Speech Buddy Application or adjust internal setting, you may use the basic Speech Buddy App provided in the Google Play Store. Search For “Speech Buddy” within the Play Store and select and install the result that lists “William Anderson” as its developer.

You must have SDK 17+ (KitKat) to install the Speech Buddy Application and meet all hardware requirements. If you do not meet these requirements, the application will not be displayed as an option within the Google Play Store. If this is the case, please follow the Customizable Instructions above and modify the programs minimum API to the appropriate level for you device. This also involves some changes that need to be made to the manifest.xml file. Adjusting this level may cause significant changes within the applications supported features, and will require some programming experience.

The Speech Buddy Application

Now that you have the Speech Buddy Application installed, you can launch it on your device.

The Speech Buddy Application dynamically keeps track of voice entry’s and modifications you have made to your selected data, and is best used in conjunction with you Speech Buddy Hardware. The application also has the options to manipulate and modify data from within your Android device. Any changes made to your data on the application will be represented in the data retrieved by your hardware. For example,

If I deleted “grapes” from “groceries” on my Speech Buddy Application, And then verbally ask the Speech Buddy hardware for my grocery list, “grapes” will no longer be listed.

Power Up and Testing

To Run AVS open two terminals on the Raspberry Pi:

First Terminal enter:

```
cd ~/Desktop/alexa-avs-sample-app/samples
cd companionService && npm start
```

Second Terminal enter:

```
cd ~/Desktop/alexa-avs-sample-app/samples
cd javaclient && mvn exec:exec
```

A Graphical User Interface (GUI) will appear, follow the instructions on the GUI.

Run the Software to make speech buddy listen as directed above. Speech Buddy will output a basic tone if it is working. It currently has basic capabilities such as simple mathematics, google queries, and time and weather updates. Speech Buddy’s location may be incorrect. Test this by asking the current time or weather, and change your location in your amazon profile accordingly.

Test the Alexa Skill by adding a list to the database, ask the Speech Buddy:

“Speech Buddy add grocery list”

An entire in the database can be seen on the Android Application, displaying a “grocery” value in List. Testing of the Android Application is discussed below.

Try adding an item to the recently created list, “grocery”, ask the Speech Buddy:

“Speech Buddy add potatoes to grocery”

The newly added item can be seen on the Android Application by clicking the text “grocery”. this will take you to a new display show the items in the “grocery” list.

Testing of the Deleting functionality can be test the same way by replacing the word “add” with “remove” or “delete”.

Speech Buddy Application Directions and Controls

Upon launching the application, a login screen will be provided. If you have an existing login, enter your credentials and select the “Login” button. If this is your first time using the Speech Buddy Application, select “Sign Up”, enter an email for your username and a password (password must be at least 6 characters in length) and select the signup button. You may now login using these credentials.

After the initial login, Speech Buddy will bring you to the core of the application. This is where all lists and their corresponding sub-items are kept, and all data you have created and manipulated is shown.

The Toolbar

On the toolbar there are several options, including:

Add List/Item: This allows you to create a new data entry within the current directory. Selecting this will prompt a pop up box asking for the information of the new entry. This option may be identified by the + symbol on the button.

Remove List/Item: This allows you to remove a data entry within the current directory. Selecting this will prompt a pop up box asking for the information of the entry to be removed. This option may be identified by the - symbol on the button. For ease of use, the user may also delete data by pressing and holding the chosen entry within the visual list. This will launch a prompt for data deletion.

Drop Down Menu: This provides several options to the user from within the toolbar, including Clear All Data, which removes all lists and their children from your account for essentially a clean start. The About App Dropdown provides a description of the Speech Buddy App and hardware, as well as it’s authors. Finally, the Logout option will sign the current user out and bring them back to the login screen.

Back: This option, a backwards arrow, will be provided on the toolbar when the user has navigated to a subdirectory. It may be used to navigate back to the parent directory.

List Screen

The user may scroll up and down the data entries if lists or items within lists by swiping vertically in either direction. As mentioned prior, a user may delete a data entry by pressing and holding the desired element.

If a data entry is a parent of other data entries (If a List has any Items stored within it),

It may be tapped to display the lists children data entries. Upon entering a sub directory like this, a back arrow will be added to the toolbar, which can be used to navigate back to the parent directory.

Colour Tutorial Setup

To set up the color skill kit that is provided to you on AWS, login to your account, select the Lambda program, and search for "color". A color skill kit should appear. After selecting the skill, follow the steps that were provided to make the skill as the previous skill, "IAM". Once you have completed the steps and filled in the required fields that are needed for the intents and the utterances; these can be found on Sanjays GitHub page https://github.com/Sanjay4966/Sanjay4966.github.io, the PDF are developer-amazon-com and developer-amazon-web. For better customization, you can add more color schemes by adding them into the Intent line so, that all the colors that are set will be able to be picked buy the user and the program will accept the valid inputs. You can also add more utterances that a user can say to speech buddy, that will allow it reply as long as that utterance the user says is in the list of phrases that it is allowed to hear.

Progress Reports

Previous Year Work 2016

Week 2:

During this week, we made our groups and also discussed about what we wanted to do for our project together, after deciding the project we went on to create a team name (The Arrested Developers) and how we are going to create the project with what materials/or code that we are going to use to get this to work. The project we decided was speech buddy that would use the amazon voice services and a server to allow us to say things and it be saved to a server for storage or removal.

Week 3:

In the software projects class, we had to make an android app that would allows us to add contents to the list and allow it to be saved on to a server. So, we could create a list of headings but also allow us to name it with a specific category name. Once this is done we can create thing for it to go into the specific category that you want to put it into.

Week 4:

During this week, we started to work on the code for the app that we are building for our project. The program that we are using is the android studio program that will allow us to create the app creating the java code and the xml files that will have the string file and the xml file that will allow us to create the pages that you will view on the app by building the page to the standards that are required.

Week 5:

Starting from week 5 we have purchased our components that will be need to build our product is buy buying the speakers the USB adapter and the mic that will be arriving in the following weeks. Including to this report for that week we have prof of purchase on our GitHub.

Week 7:

During this week, we had to present the beta part of the Speech Buddy app to the software projects advisor. And see where we need ti do better in for the app to be successful for the long run.

Week 8:

We make a placard that would be used for the open house or any other presentation. The placard would include the description and the specifications to what we needed and having our names illustrated onto the side of the placard saying who made the project.

Week 9:

We made a build video that have use put the components together an then show it running in the first 30 seconds of the video. This video should show what material we used and the where they were put in order for then to work.

Week 10:

Speech buddy will be shown at the open house on Saturday. It is currently capable of speaking through a combination of a text to speech python library, E-speak software, sound configuration file edits and self implemented code. The next step that we are working on will be to implement its speech to text capabilities to receive input through the microphone.

Week 11:

Speech buddy was shown at the open house, however without audio input a there were complications with the microphone. A keyboard was substituted to imitate voice input, but real voice input MUST be implemented by the next week. A new microphone has arrived after placing a new order for a small and better mic, with this we should be able to successfully record voice input for interpretation on the speech buddy. After we have achieved voice input, out project will have a working sensor(voice), working hardware, and a working model, thus making us prepared for the next course (Systems Project).

Week 12:

After running the Alexa voice service guide on how to get it to work with the voice input that it has, speech buddy is now accepting voice input using the same libraries as the amazon Alexa. We have removed the wake word dependencies so we can create our own skills and code for an Independent Speech Buddy platform. It will use already existing libraries to perform basic activities, such as querying google, addition, current time, etc. We are incorporating our skills (Calendars, list) currently. We are finally caught up to the project plan after have issues with the mice, and now preparing for the presentation of the fully working prototype of speech Buddy. Our budget has changed however, as we had to purchase a new USB microphone for 14.99. Having purchased the new mic, we are not going to be needing usb sound card

that slits the audio and mic to separate ports, thus it will not be included in the budget of prospective builders of speech buddy. Aside form the microphone issue, which was successfully resolved, no major obstacles were encountered and the project took a great step forward this week.

Week 13:

After encountering some issues and then solving them from the previous week we are ready to present our presentation to our colleague on what we did for those weeks and show how the prototype works and demonstrate it to other to have then participate by trying it out.

Status Report - sent by Sanjay Jerad [02/14/2017]

Dear Kristian Medri,

Development of Speech Buddy's primary Amazon Voice Service Skill has begun; it is being coded in JavaScript as Amazon Voice Services has the most support for this language. The skill will be the final connection between the Speech Buddy Hardware and the Speech Buddy mobile application, allowing the data shared between the two to be shared synchronously. For testing purposes, simple pre-made skills such as a basic color skill are being deployed to the Speech Buddy hardware to ensure integration capabilities of our hardware.

Speech Buddy has also had a new case designed for the hardware. The previous case model had joints and edges that were too thin by human error and could not support the cases structure. The case has been redesigned to be stronger, as well as have the size of the microphone hole adjusted to the most recent microphone we have been using. The new box is intended to be printed February 14th.

Budget wise, Speech Buddy has gone under no significant changes. The cost of the box could be assumed to be approx. $7.50 – $10.00, as it consumes approximately ¼ - 1/3 of a sheet of acrylic.

In relation to our project plan, we are technically caught up, but development of Speech Buddy itself has slowed prior to this and must be made up for within the coming weeks to ensure the schedule can be maintained with the assumption of unexpected software and hardware obstacles.

No major obstacles were encountered this week, and Speech Buddy has made some progress forward from a slow start this semester. We intend to pick up the pace and have the primary skill, as well as

some additional skills created and downloaded onto our Speech Buddy hardware for demonstration at the open house.

Sincerely,

Sanjay Jerad

Integration Report - sent by William Anderson [03/07/2017]

Dear Kristian Medri,

Speech Buddy's Integration achieved some significant milestones this past week. Kevin has created and deployed a Dynamo database, which we were able to develop a skill for in Python and finally successfully connect to. We are able to add, delete and manage items within our database via voice now.

The Android application has had the Amazon Web Services SDK Installed by William and can successfully connect to the database as well, and now forms a complete integration with the Speech Buddy hardware, accessing the same data.

Sanjay created a common Amazon Developer and AWS Services account for all of us to be able to develop collaboratively and host the final database and skills services.

The next step is creating the appropriate skills to use Speech Buddy's integration. The majority of the work now will be mostly programming Python skills to modify the database based on voice according to certain keywords; the connection is done, but usability must be instantiated.

Some Additional Notes:

The case for Speech Buddy has been printed and is now assembled.

Budget wise, Speech Buddy has gone under no significant changes; If the Dynamo database exceeds certain usage, we will begin being charged a few dollars a month, however this is an unlikely scenario.

In relation to our project plan, we are on track, and will continue to implement usability of our AWS skill and enhance them to take advantage of our integration.

There were some minor android obstacles encountered this week, primarily in relation to compatible android versions and the AWS SDK. These were overcome by making our application compatible and

switching some compile versions. Speech Buddy finally has connections on both sides, and we may now focus strictly on creating the necessary skills. Good progress was made this week.

Sincerely,

William Anderson

Troubleshoot Report - sent by Kevin Dang [03/21/2017]

Dear Kristian Medri,

The integration of the Speech Buddy hardware and software has progressed smoothly the past few weeks, with some hindrances. Kevin has continued working on the python code for the Alexa Skill along with the database. Problems appear with adding specific items to the database, up until a few weeks ago we have been hard coding values in to add data. Kevin has solved this problem by making the Alexa Skill add values depending on what the user says.

He created the second table in DynamoDB to hold the Items of a specific List, and is able to add items. Problems arise around adding items for a list into the database, with needing to add items only after checking if a List exist to add to. Error checking is posing a problem and will be further looked into.

William has continued working on the Application for the Speech Buddy, though the Integrated Development Environment "Android Studio" is posing some obstacles in the form of a context location issue. This is a technical code issue, not a connection issue. A connection is being made but the application is unable to display the data thus far. William hopes to resolve this shortly.

Sanjay has explored the relationship between external libraries and the Alexa Skill. If it is possible to use other resources outside of the AWS services in our Skill. He has run into problems of implementing external libraries into sample skills such as the Color sample, as of now it does not look like Alexa Skill allows for external resources.

Regarding the budget of the project it has been updated with the price of building the case, with pricing of the acrylic used and time on the laser cutter. Other than that budget is the same.

Everything is going well schedule wise, we will continue working on the Skill, adding more features to make it user friendly. Other than that we are moving on track with the project plan.

Sincerely,

Kevin Dang

Conclusion

The Speech Buddy project is a voice interface allowing users to control and manipulate data easily. This project hopes to improve the planning and scheduling abilities of users, in a unique and effective manner. The Speech Buddy project consists of a Raspberry Pi with a microphone and speaker, for input and output. Data is inputted by the user in the form of voice input, which is stored on an external online database. Data can be read on our Android Application, displaying information located on the database in a user-friendly interface. The Android Application also includes the functionality for manual input using the keyboard on smartphones, rather than through the microphone.

Recommendations

The Speech Buddy project is able is work correctly and incorporates the main functionalities we desired. Following the Build instructions mentioned on the Technical Report, if the user would like to construct this project, there are some recommendations we would like to include. The idea of portable is important, by using a small power supply to generated enough Voltage and Current, the Speech Buddy can be transported along with user. A smaller case can be created when looking at the hardware components, if a smaller speaker is available. The Project utilizes Amazon Alexa's voice Service, a platform that allows develops to create different Skills. The Speech Buddy incorporates a skill that allows users to add and delete from a database. Adding additional skills to the project is possible to improve Speech Buddy's overall abilities. Examples such as, ability to calculate temperature, do basic mathematical problems. The Speech Buddy can be tailored to the user, evolving to meet your requirements and standard as an engineer, developer or a regular everyday person.

References

Karabetsos, S., Tsiakoulis, P., Chalamandaris, A., & Raptis, S. (2009). Embedded unit selection text-to-speech synthesis for mobile devices. IEEE Transactions on Consumer Electronics, 55(2), 613–621. https://doi.org/10.1109/TCE.2009.5174430

Wang, D., Leung, H., Kurian, A. P., Kim, H. J., & Yoon, H. (2010). A deconvolutive neural network for speech classification with applications to home service robot. IEEE Transactions on Instrumentation and Measurement, 59(12), 3237–3243. https://doi.org/10.1109/TIM.2010.2047551

Yuan-Yuan, S., Xue, W., & Bin, S. (2004). Several features for discrimination between vocal sounds and other environmental sounds. In 2004 12th european signal processing conference (pp. 2099–2102).