SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: May 30, 2017

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Davis Allen
Robert Bayer

ENTITLED

Conversation Station

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Thesis Advisor

Thesis Advisor

f

Department Chair

Department Chair

Conversation Station

by

Davis Allen
Robert Bayer

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
May 30, 2017

Conversation Station

Davis Allen
Robert Bayer

Department of Computer Engineering
Santa Clara University
May 30, 2017

ABSTRACT

We built a mobile application that improves speed and personalization in conversations for people struggling with verbal communication. Many people diagnosed with autism, Down syndrome, and other disorders face daily challenges involving communication due to speech impediments. Existing solutions allow users to communicate via speech cards or typing on a keyboard. However, these solutions make tradeoffs between personalization and speed, compromising what it takes to have fluid, natural, and rewarding conversations. Our solution speeds up personalized communication by applying Machine Learning principles, Artificial Intelligence, and Natural Language Processing. Our project will predict how a user will respond based on natural language processing results of verbal input and some base-layer artificial intelligence rules, allowing the user to communicate quickly with their own voice.

Table of Contents

List of Figures

Chapter 1

Introduction

Communication is essential to building relationships. A person who has challenges speaking will face a lifetime of roadblocks in building friendships, connecting with family, and meeting daily needs. In the United States, it is estimated that 1 out of every 68 children will be diagnosed with some level of Autism[1], and many of these children will face communication challenges or be rendered completely nonverbal, depriving these children of a voice. As this number continues to rise[2], finding ways for everyone to clearly communicate is essential to enhancing the human experience.

Today, nonverbal children such as those diagnosed with autism or Down syndrome communicate using many methods including gestures, sign language, and picture symbols. One of the most popular methods of communication is a device that generates speech. These devices come in many different forms. Some are similar to keyboards on which the child can type out what they want to say, while others have a list of buttons with pre-programmed messages from which the child can choose. Both of these options have also been incorporated into touchscreen devices such as the Apple iPad, so they are easily portable.

While the ability to type out any response gives a flexible voice to the children, it can be tedious to retype similar responses and frustrating for all involved due to the time it takes to construct responses using a keyboard. The solutions with pre-programmed messages solve this problem by speeding up communication, but they impede the expressiveness of the children by limiting their response options. These limitations do not allow subtleties in diction, syntax, and personal preference to be communicated, erasing the voice from the personality behind it.

[1]Maguire, Corinne. "Autism on the Rise: A Global Perspective." Harvard College Global Health Review. (retrieved October 7, 2016).

[2]Christensen, Deborah L.. "Prevalence and Characteristics of Autism Spectrum Disorder Among Children Aged 8 Years ? Autism and Developmental Disabilities Monitoring Network, 11 Sites, United States, 2012." Centers for Disease Control and Prevention. (retrieved October 7, 2016).

Using Machine Learning principles, Artificial Intelligence and Natural Language Processing, we developed an application that makes conversations quick but also personalized, giving people their own voice. We propose a solution that listens to conversations and gives the child several quick options to speak. These options will be personalized to each child, so that they maintain their voice. Additionally, we will have the option to type out a response if the available quick options are not what the child wants to communicate. The typed answers will be used to help the system learn how the child responds, thereby improving future suggestions. We plan on testing this solution with nonverbal children at Hope Technology School in Palo Alto, California.

Our solution combines the benefits of both current solutions, while eliminating the problems. By having the system learn about the child on an individual level, this communication tool will allow children to share their voice with the world. The inclusion of the quick suggestions will drastically improve response time, easing the communication process both between children and between verbal adults, such as the teacher or parent, and the child. Communication is crucial to forming human connections. Our proposed solution allows for seamless, fluid communication, giving everyone a voice.

Chapter 2

Requirements

Listed below are the Functional Requirements, Non-Functional Requirements, and Design Constraints of the project as gathered during the Requirements gathering phase.

2.1 Functional Requirements

Critical:

- Provide input via the keyboard.
- Capture audio via device microphone.
- Speak through device speaker.
- Suggest responses based on voice input.

Recommended:

- Provide input options via picture cards.

Suggested:

- Say keyword to activate microphone and save battery.

2.2 Non-Functional Requirements

Critical:

- The system will be able to handle multiple users.
- The system's user interface will be intuitively designed.

Recommended:

- The system will be secure when storing private data.

Suggested:

- The system's user interface will be able to use graphics that are intuitive and familiar.
- The system will be unaffected in loud environments.

2.3 Design Constraints

- The system must consist of a mobile application that runs on iPad devices.

Chapter 3

Use Cases

Use case diagrams demonstrate how different users will interact with our system. In our case, a verbal user will be able to speak to the mobile device, and the nonverbal user will be able to formulate a statement and use the device to speak.

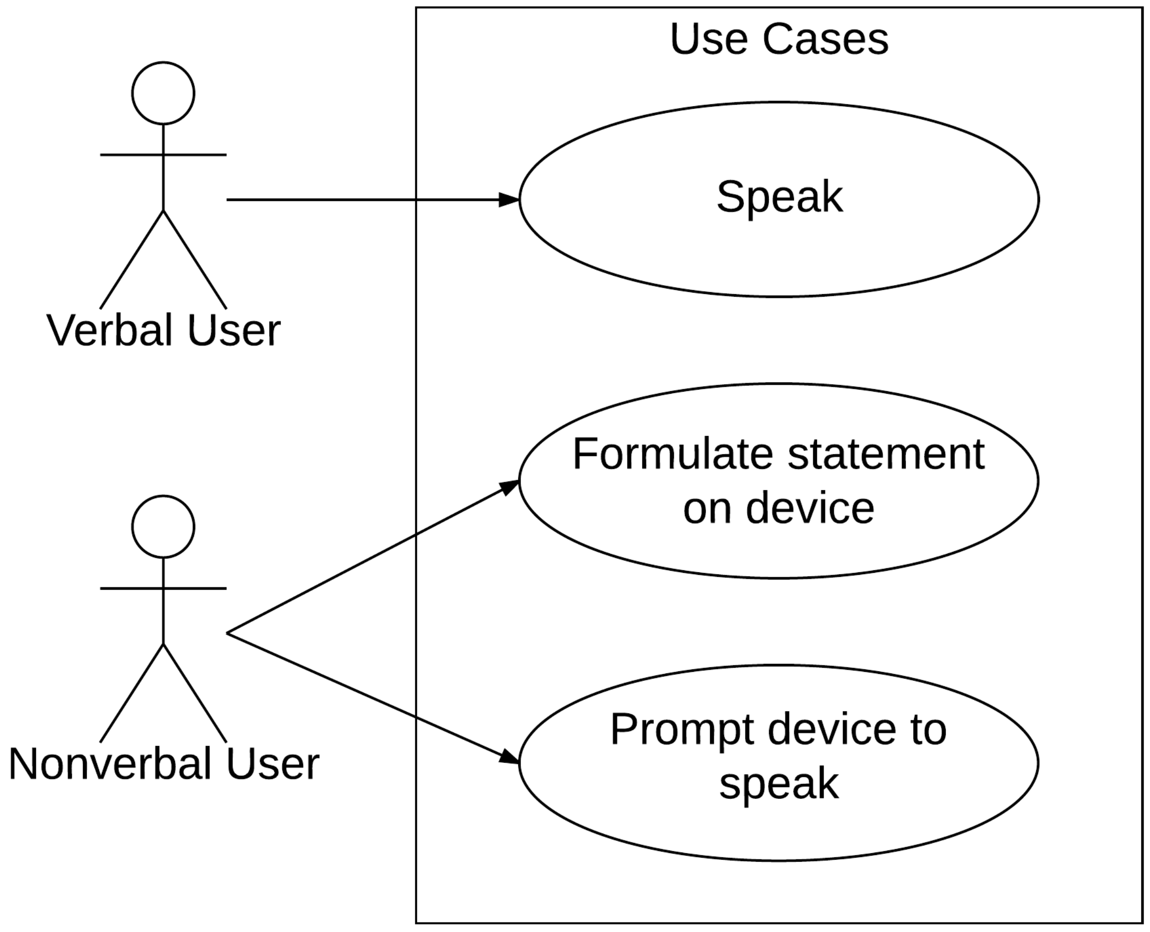

Figure 3.1: Use Cases

1. Speak
    - **Goal:** Ask a question for mobile device to record and process
    - **Actors:** Verbal User
    - **Pre-conditions:**

      (a) Application must be open on device
    - **Steps:**

      (a) Nonverbal user opens application

      (b) Nonverbal user presses button to record voice

      (c) Verbal user speaks and device records

      (d) Device translates audio to text and sends to server
    - **Post-conditions:**

      (a) None
    - **Exceptions:**

      (a) Application must have device permission to use microphone
2. Formulate Statement on Device
    - **Goal:** Create a statement to respond to or initiate with the verbal user
    - **Actors:** Nonverbal User
    - **Pre-conditions:**

      (a) Application must be open on device
    - **Steps:**

      (a) Open application

      (b) Uses keyboard or picture card to add a word

      (c) Add words until statement is complete
    - **Post-conditions:**

      (a) Application displays fully formatted statement

- **Exceptions:**

  (a) None

3. Prompt Device to Speak

   - **Goal:** Speak the statement that has been formulated by user
   - **Actors:** Nonverbal User
   - **Pre-conditions:**

     (a) Statement must be fully or partially made on device

   - **Steps:**

     (a) Open application

     (b) Formulate a statement (if necessary)

     (c) Press the speak button

   - **Post-conditions:**

     (a) Device will speak the statement

   - **Exceptions:**

     (a) Application must have device permission to use speaker

     (b) Device volume must be on or nothing will be heard

Chapter 4

Activity Diagram

The following flowcharts outline the workflow of both verbal and nonverbal users actions and interactions with the system. The diagram does not include the system actions, such as processing audible speech and displaying relevant suggestions.

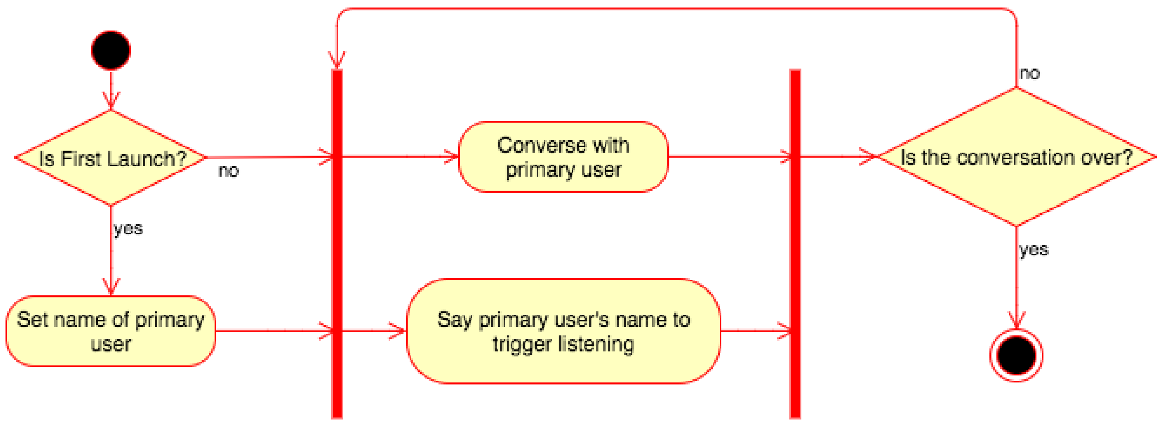
Is First Launch?
no
yes
Set name of primary user
Converse with primary user
Say primary user's name to trigger listening
Is the conversation over?
no
yes

Figure 4.1: Work flow of the verbal, guardian user interacting with system

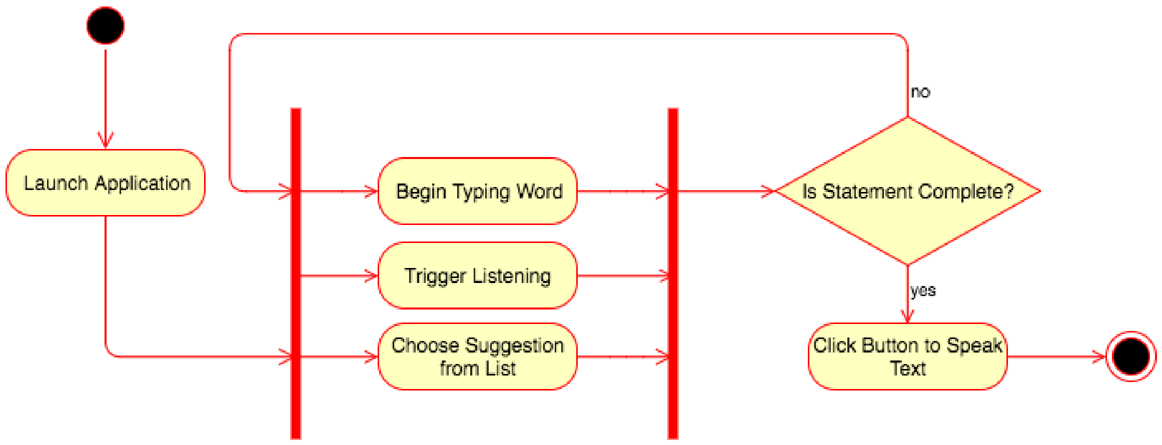

Figure 4.2: Work flow of the non-verbal, child user interacting with system

Chapter 5

Conceptual Model

Users will navigate to our application on an iPad tablet in order to use the system. Upon launching the app, they will be greeted with the keyboard view, as shown in Figure 5.1, which they can immediately start using to type what they want to communicate.

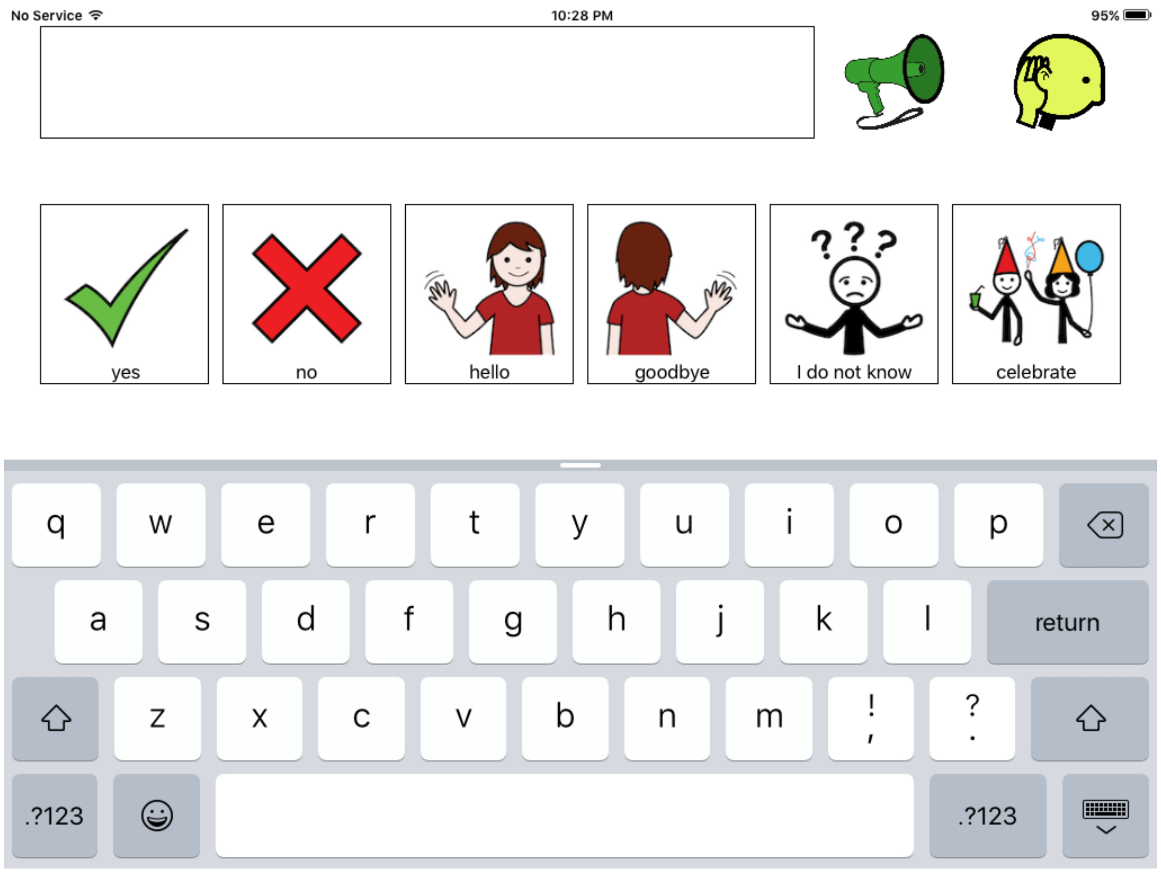

Figure 5.1: Initial Interface on Application Launch

Upon saying the nonverbal user's name, which is programmed into the application on the first launch, or by clicking the listen button, the system will bring up the listening interface, shown in Figure 5.2, and begin processing verbal input.

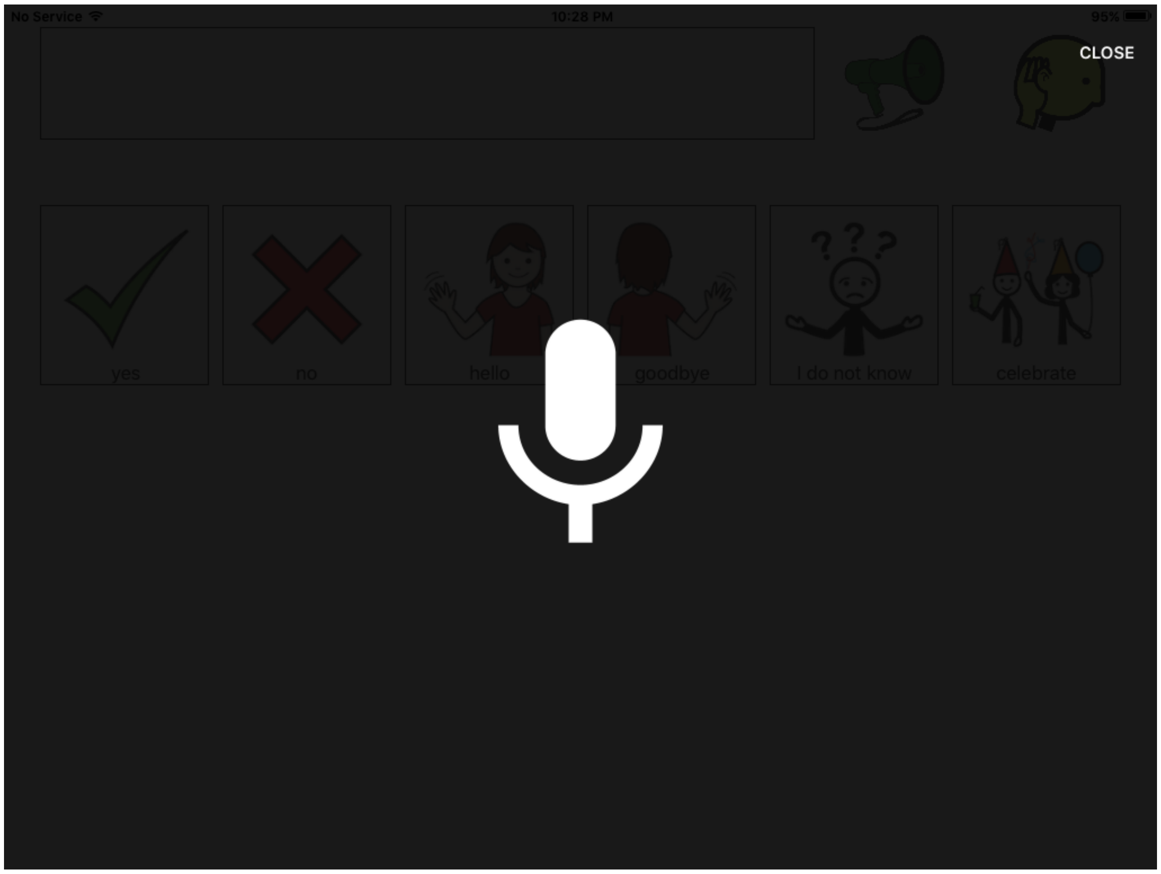

Figure 5.2: The Listening Interface

When the verbal user is done speaking, the app will finish processing the verbal input and decide which pre-populated category of picture cards to display above the keyboard. Figure 5.3 shows how the app would adjust if the verbal user said ”How do you feel?”.

At any point during the communication process, the nonverbal user can either press one of the picture cards to immediately speak the word or phrase beneath the card, or press the button on the right of the screen, which will speak what they have typed into the input box aloud.

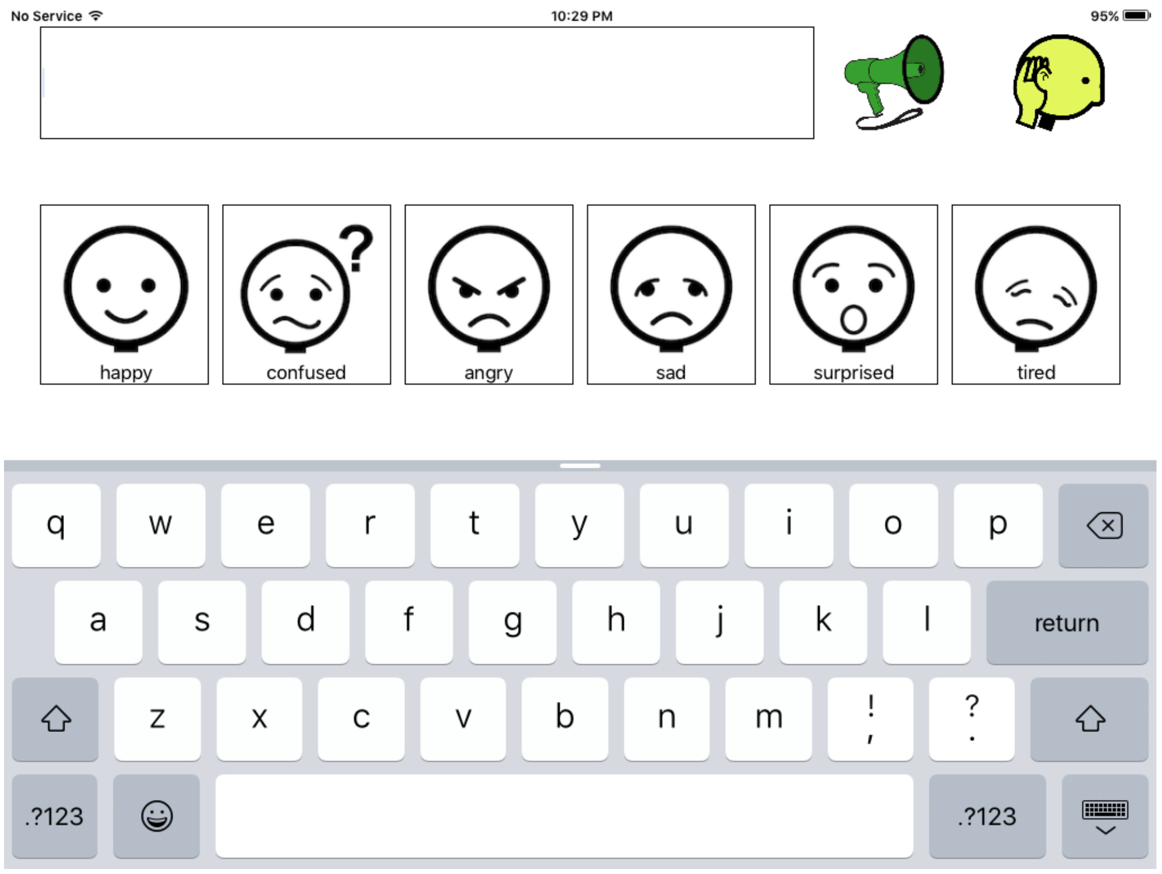

Figure 5.3: Demonstration of the Picture Cards adjusting to Dynamic Verbal Input

Chapter 6

Architectural Diagram

We plan to utilize a basic client-server architecture to complete this project as shown in Figure 6.1.

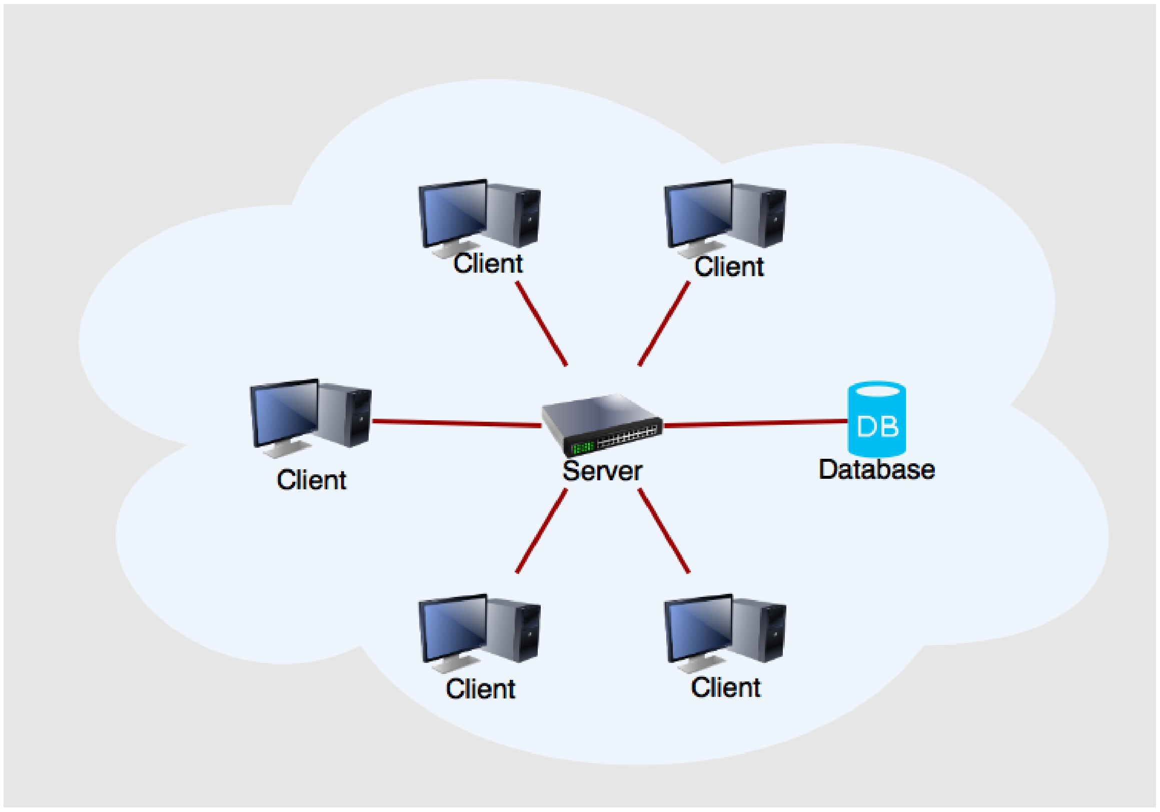

Figure 6.1: Client-Server Architecture

Users will be able to use our application on their mobile tablet, acting as a client, in order to access our site. The server in our system will be responsible for storing data for each child in the database, formulating a categorization response of the verbal input, and delivering these results to the client.

The database will be Google Datastore, and the Google App Engine server will write to and read from it order to access data.

Chapter 7

Technologies Used

- Hardware
  - Development
    - Macbook Pro
    - Macbook Air
  - Application Testing
    - iPads
  - Server Processing
    - Google App Engine
- Programming Languages
  - Swift
    - For iOS Programming
  - Python
    - For web server programming
  - JSON
    - For communicating with REST APIs
- IDEs
  - XCode

- SDKs
  - Nuance Developer's SDK
- APIs
  - Google's Cloud Natural Language API
- Database
  - Google Datastore

Chapter 8

Design Rationale

We are leveraging a client-server model in order to minimize impact on the client mobile devices. In our model, a client will collect audial data and translate it into text using Nuance Developers SDK and send the text to the server. The server will use NLP tools such as Google's Cloud Natural Language API to process the text and generate suggested classification responses to send back to the device as a JSON object. Having the server do the processing will minimize the use of the device's battery life, and will also decrease the disk space our app will require to be stored on the device.

We are using iPads for our testing because that is what our partners, the Hope Technology School, primarily uses in their classroom. This will eliminate the need for us to buy iPads for beta testers to use; saving money, time, and scheduling complexity.

Making use of the existing APIs like Google's Cloud Natural Language allows us to avoid re-inventing the wheel and focus on how those tools can be used to fulfill our purpose.

We will develop the application in XCode, using the Swift language because it is the designated language compatible with iOS to make applications. We will use Python to design the web server because we have experience using Google App Engine to make web servers specifically in Python and the APIs are well documented for the language.

Finally, we will use Google Datastore to store conversation data for the user because it easily integrates with Google App Engine.

Chapter 9

Testing Procedure

Our testing process can be broken into three main sections: unit testing, an alpha release, and a beta release. Unit testing was done throughout the development process from Week 1 - 10 of Winter Quarter. Our alpha testing took place between Weeks 7 and 9 of Winter Quarter. At this point, the system was the first iteration of all the features of our application. The feedback from this stage helped us refine our interface and move into the beta, which took place from Weeks 1-5 of Spring Quarter.

Chapter 10

Risk Analysis

Table 10.1, shown below, depicts the risks and consequences involved in completing the system. The table also includes probability of the risk involved along with a severity ranking between 1 and 10. The probability and severity are then multiplied to receive an impact measurement. For each risk, there are at least two mitigation strategies that are included to avoid the issue.

Risk	**Consequences**	**Probability**	**Severity**	**Impact**	**Mitigation Strategies**
Illness	Team member(s) will not be able to work on project for a period of time.	.5	4	2	1. Get a lot of sleep. 2. Stay hydrated.
Insufficient Development Knowledge	The system will not be adequate and may not be finished on time.	.4	5	2	Get a lot of sleep and stay hydrated.
Insufficient Web Development Knowledge	The system will not be adequate and may not be finished on time.	.4	5	2	1. Use online tutorials. 2. Revisit past notes/projects.
Team Coordination Failure	The team will not finish the project or certain aspects of the projects in time.	.15	8	1.2	1. Keep an organized schedule of due dates. 2. Follow Development Timeline. 3. Hold team members accountable for what they need to get done.
Requirements are Incorrectly Communicated	Extra time gathering requirements. Customer doesn't agree with the end system.	.1	8	.8	1. Meticulously develop requirements with customer. 2. Regularly hold update meetings with customer to make sure on the right path.
Server Failure	The system will not have a server to host the client-server architecture and will therefore fail.	.02	10	.2	1. Research other hosting strategies in case of DC server failure. 2. Use local resources for backup storage and processing.
File Loss	Lost work up to the date of the file loss.	.01	10	.1	1. Use github to manage source code and versioning. 2. Update project tree after every development session.

Table 10.1: Risk Analysis Table

Chapter 11

Ethical Analysis

Even though our project can seem benevolent, there are several ethically ambiguous scenarios that must be considered and weighed prior to making decisions.

First, at an organizational level, our team has a couple of ethical dilemmas to consider. One such dilemma is how to ensure workload is being divided evenly among teammates. Like many problems, there seems to be a simple solution, by making a list of the work to be done and divide it evenly by number of tasks. But this process and judgment gets trickier with added complications, such as certain tasks requiring different amounts of time or team members missing deadlines. There are an infinite amount of potential situations to assess, so a global solution must be developed that can be applied across multiple scenarios. A good solution template would begin with a group discussion to try to work out the issue. If not everyone is satisfied with the results, we should schedule a meeting with our advisor to ask for him or her to mediate the discussion, allowing the experienced advisor to offer advice or make decisions for the team if necessary. Though this mediation strategy is generic it can be applied across multiple scenarios, which makes it a good basis for resolving ethical issues.

Secondly, concerning the social aspects of our project, there are many more ethical scenarios to consider in order to ensure that our application maximizes social benefits while minimizing concerns. One question we must consider is if we, as the developers, should have the power to control someone's voice. Our application will try to predict what a user will typically say and how he or she would respond to a question. This gives us the capability to influence what options the user will choose in speaking. This feature can be potentially misused to influence and control someone's voice in an extreme case. Questions determining the extent of control this application need to be evaluated to determine what kind of control the application should have in determining how a user will say something or respond to a question.

Thirdly, we have to consider the ethical implications of how our product system will be implemented. Our application will need to save records of the user's conversations to remember the interests and voice patterns of the user. One situation to evaluate is the option to save that personal information locally on the user's device or on our servers in the cloud. Both options have their respective pros and cons, namely that the cloud storage will improve speed and battery usage, and local storage will improve security. These trade-offs will force us to evaluate the situation and have us weigh the priority of features to determine how secure we need to store the user's data. Likely, these priorities will be shifted as the application gets tested by users and they give feedback, so it will be important for our team to come back to this dilemma and re-evaluate the best storage option from time to time. The extent and implementation of features need to be evaluated and assessed in the development of our application with user safety in mind.

Chapter 12

Development Timeline

The development timeline shown below depicts the set of tasks that were completed throughout the year. We listed each task set by school quarter which roughly followed a requirements, design, implementation, then testing (RDIT) procedure.

- Fall Quarter
  - Initial requirements gathering
  - Research and design
  - Documentation
- Winter Quarter
  - Implementation
  - Testing
  - Demo with HTS representative
- Spring Quarter
  - Implement suggestions for improvement from demo
  - Final testing
  - Final documentation

Chapter 13

Lessons Learned

Over the course of this project we climbed over many walls to develop a consistent and finished product. Here are the lessons we learned that helped us in this project and that we will carry on towards future work.

1. **Be wary of dependencies.** The more outside tools that your project relies on, the more points of failure are induced. Be careful of using too many tools that are outside of your control, and be prepared for them to suddenly change or not work.

2. **Multiply estimated time by 5.** In most circumstances, whatever time you estimate a task to take, it will take longer. Planning for unexpected development time will provide for a more accurate time assessment.

3. **Seek feedback early and often.** To avoid unexpected changes of requirements and unnecessary implementations of features that the customer may not want, maintain continuous contact with the customer. The earlier along in the development stages you can get feedback, the easier it is to iterate on those changes.

4. **Build it up piece by piece.** Incremental design will allow for easy feature integration and natural divisions for unit testing. Building piece by piece will improve code understanding and reduce the number of unexpected errors.

Chapter 14

User Manual

1. Install and launch the app.
2. Set the "wake" name, which should be the first name of the nonverbal child user.
3. Choose any option from the below list and repeat in a cycle until conversation is complete:
   - Type a statement using the keyboard.
   - Speak the value of the input field over the iPad speakers.
   - Select a preset picture card to speak the associated word or phrase.
   - Select the listening button to trigger the device to begin processing verbal input.
   - Verbal user can say the nonverbal child user's "wake" name to trigger the device to begin processing verbal input.

Chapter 15

Install Guide

15.1 Server Setup

None, the server is running at the URL hard-coded into the iOS mobile application. The server is already set up for you.

15.2 iPad Application Installation

1. Double click the Conversation Station.xcworkspace file to open it in Xcode.
2. Connect your iPad to the computer and select that iPad as the device to run the application.
3. Click Run from the menu to install the application and run it on the iPad.