Boston University
Electrical & Computer Engineering

EC464 Capstone Senior Design Project

User's Manual

Submitted to

Andrey Vyshedsky
ImagiRation
(617) 433-8577
imagiration@gmail.com

by

Team 14
Vobot

Team Members

Laura Salinas laums@bu.edu
Priya Kapadia priyak@bu.edu
Steven Graham smgraham@bu.edu
Shivani Bhatia scbhatia@bu.edu
Arley Trujillo arley78@bu.edu

Submitted: 4/18/18

Table of Contents

Executive Summary

Approximately 1.5% (one in sixty-eight) of children in the US are affected by Autism Spectrum Disorder (ASD). Of these children, 30-40% remain nonverbal for the rest of their lives and are placed into institutions as they cannot take care of their own needs. Children with ASD only have a short window of time for language acquisition – the first five years. That being said, it is imperative to maximize their exposure to language from a young age, thus improving their communication and speech skills. Maximizing the amount and quality of exposure is not a simple undertaking, and is a responsibility that usually falls on the child's parents. Parents often opt to enlist the talents of trained professionals for treatment, but doing so is very costly. While such assistance is helpful and can provide results, the costs mentioned may cut sessions short or deter a family from even considering the option. To further compound these issues, there is a personnel shortage of these professionals.

Our group is determined to deliver a voice controlled language therapy robot that will help assist children with their language acquisition skills. Our proposed technical approach is to use the WowWee CHiP Robot as reward platform which the child will interact with. An Android phone with our open-source application will be used to

simulate model words and to record vocalizations made by the users. The application, through the use of third party API, processes what the child has said and returns a similarity score indicating how close the child's articulation is to the word that is being practiced. Upon a successful attempt, the robot will reward the child by performing one of its pre-programmed skills, like dancing. The application will allow parents or therapists to track the results of the child as they interact and learn with the robot. The robot and application are meant to be a cost efficient supplement to speech therapy, being a tool for doctors and parents to use on the days when speech therapy through a professional is not available (such as at home sessions).

While there are devices in the market that aim to do similar language acquisition training, neither of these products offer an individualized learning curriculum, nor do they communicate effectively with the child. We hope to change this by providing a fun, friendly, adaptable, and customizable language therapy robot that will interact with the child, while also motivating them to learn different words.

1 Introduction

The purpose of this project is to deliver a system capable of assisting children with Autism Spectrum Disorder (ASD) in the development of their articulation and communication (i.e., verbal) skills.

Our approach has solved our user's needs through identifying, piece-by-piece, which topics had causal relations with others. For example, we would not be able to implement a levelling system without an application to bridge the gap between storing information and obtaining information. Similarly, we would not be able to implement a levellying system or its subsequent rewards without a speech processing component to deliver performance results. It was through this methodology of identifying causal relations that we managed to work on parts ot the project in a chronological order.

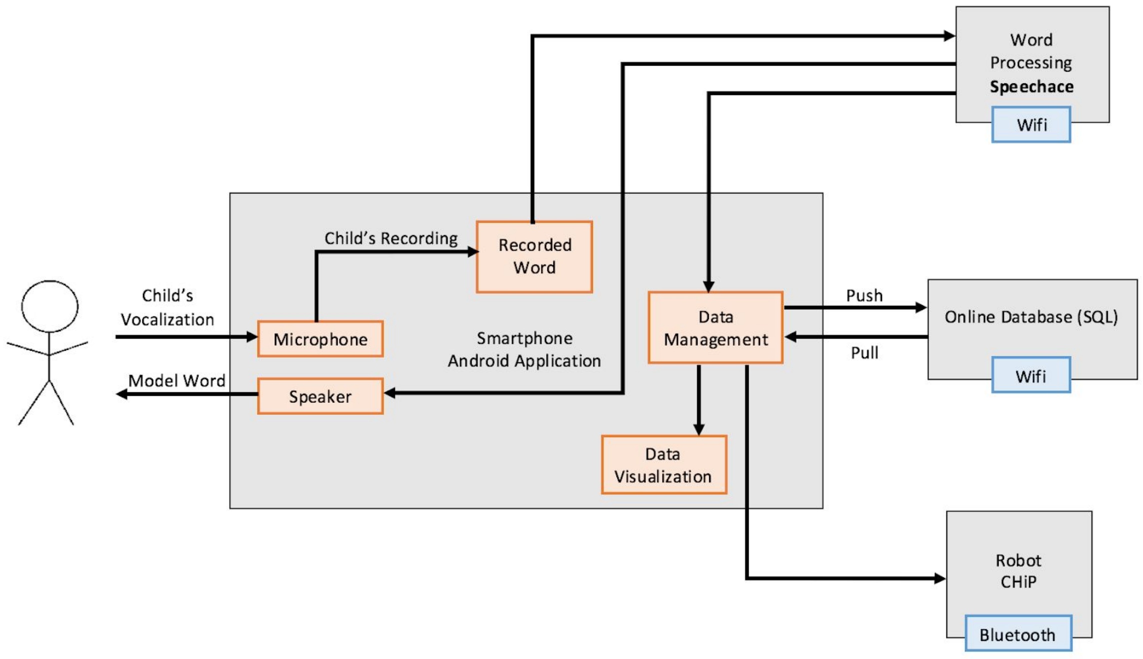
Encouraging language acquisition is of the utmost importance because humans are constrained to a closing window of five years to obtain a firm understanding of a native language. Since 30-40% of children with ASD remain nonverbal, this raises the stakes drastically. At a glance, the project is able to deliver results through the help of an acoustic-based neural network, Android application, Amazon Web Services (AWS) cloud-based database, and a smart robot-dog CHiP.

Certain circumstances that shaped the design were our customer's desires in the application user interface (UI) as well as the levelling and rewards system. The application UI was requested to be easy to use and friendly in case of parents who are not tech savvy as well as having a low number of bugs in order to maximize effectiveness of learning sessions. The particular levelling system implemented was specifically requested by our customer and is a technique called "progressive shaping of successive approximation" - whose usage by researchers in improving vocalizations in children with ASD is promising. As for the rewards system, we leveraged CHiP's API to deploy a random dance or maneuver upon certain milestones in levelling.

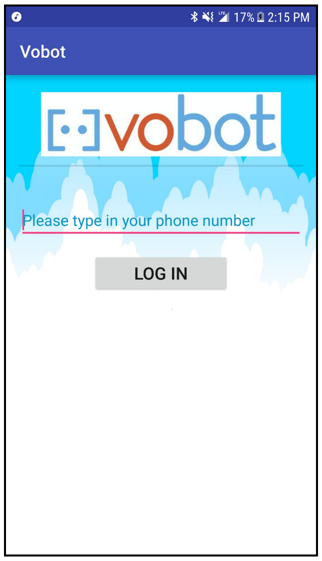
Highlights and special features of our project include a simple application UI, functional and relational database, alongside the scoring of unintelligible speech. The application UI simply requires a one-time input of a phone number to begin, and the rest of the application is straightforward in its menu layout (by requiring navigation as easy as one tap on the screen to proceed or go back) to further access learning sessions, progress reports, and settings. The database interacts with application in storing session number, words practiced, and subsequent scores to provide progress reports in the form of line graphs displaying a certain word's score versus session number. The scoring of unintelligible speech will return a similarity score even for partial vocalizations (e.g., "ah", "ba", "che") relative to a model vocalization that is a complete, intelligible vocalization. With the understanding of what we aim to achieve with this project, we can now explore a more in depth technical approach and step by step guide on how to install and run both the software and hardware package.

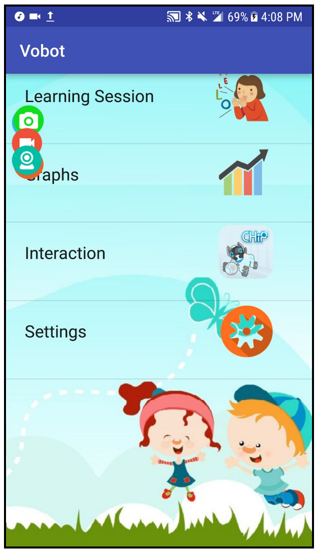
2 System Overview and Installation

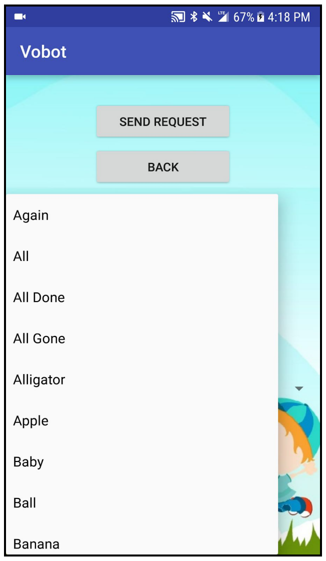
2.1 Vobot System Block Diagram



2.2 User Interface





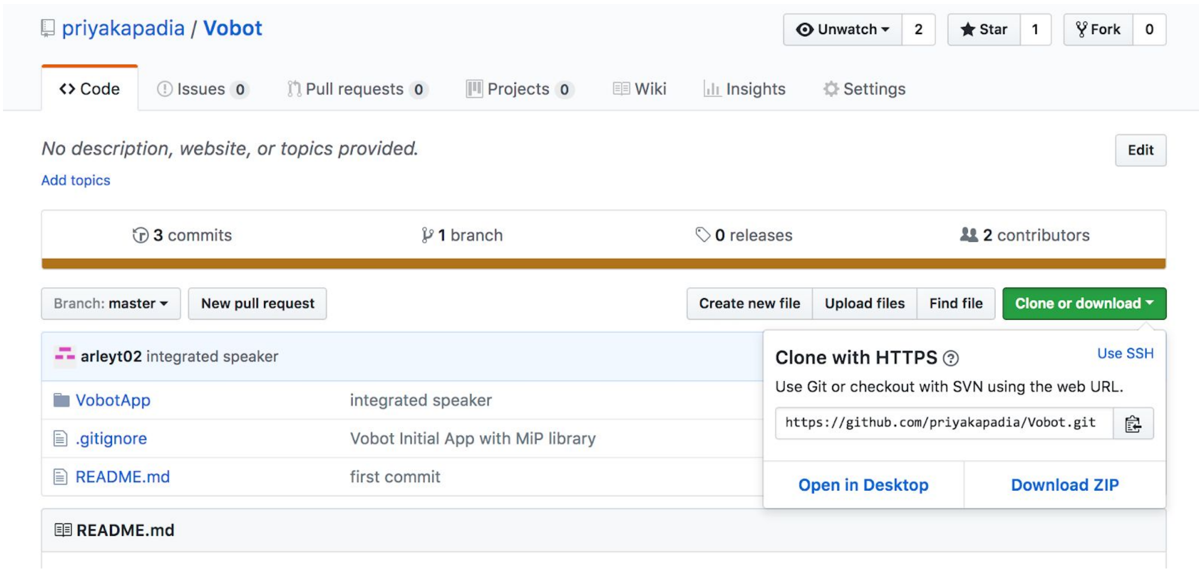


2.3 Physical description

2.4 Installation, setup, and support

2.4.1 Android App

1. To get started with the Android App, clone the repository from github: https://github.com/priyakapadia/Vobot



No description, website, or topics provided.

arleyt02 integrated speaker	
VobotApp	integrated speaker
.gitignore	Vobot Initial App with MiP library
README.md	first commit

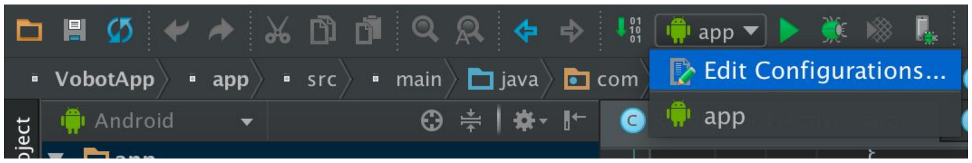
Clone with HTTPS
Use Git or checkout with SVN using the web URL.

```
https://github.com/priyakapadia/Vobot.git
```

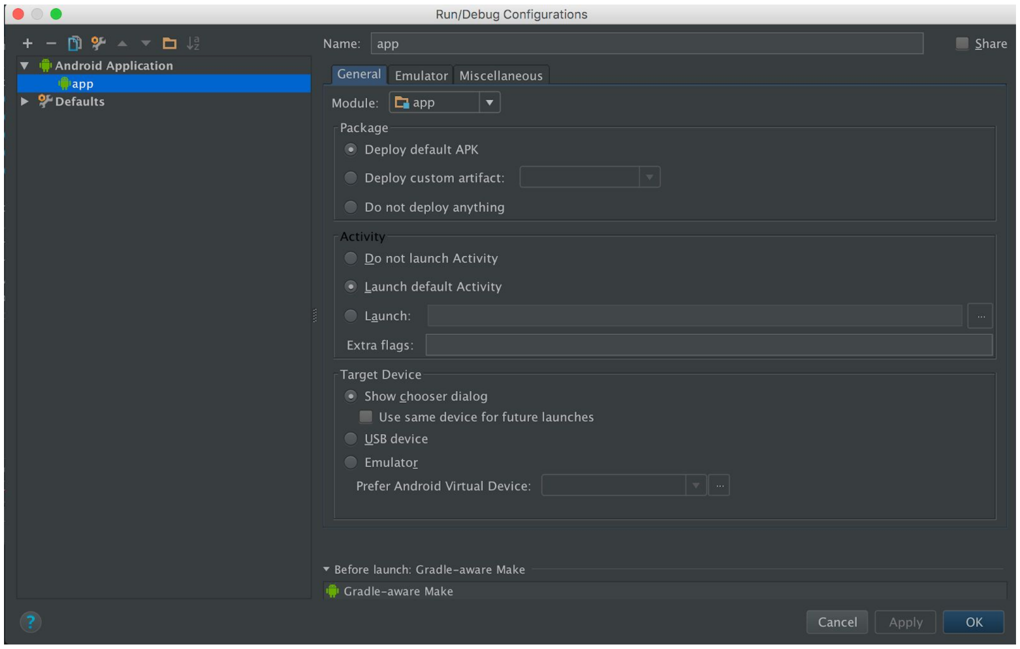
You may do so by clicking "Download ZIP" as shown above, or copying the url and typing "*git clone https:...*" on terminal. To do the latter, github needs to be installed on the machine.

2. Next, download Android Studio and import the project that you have just cloned. (File → Import Project)

3. Once you import the project, Android Studio should give you an option to sync Gradle files on the top right.
4. Then, edit configurations as shown in the image below:



5. Create a new android application build with the following specifications:



Run/Debug Configurations

- Android Application
  - app
- Defaults

Name: app Share

General | Emulator | Miscellaneous

Module: app

Package
- [✓ Deploy default APK]
- Deploy custom artifact:
- Do not deploy anything

Activity
- Do not launch Activity
- [✓ Launch default Activity]
- Launch:

Extra flags:

Target Device
- [✓ Show chooser dialog]
  - Use same device for future launches
- USB device
- Emulator

Prefer Android Virtual Device:

Before launch: Gradle-aware Make
Gradle-aware Make

Cancel Apply OK

6. Run the program using the play button and select the device you'd like to upload your app onto. Open the app on your phone, and start playing.
7. It should be noted that in the Android app setup, the user needs to change the IP address and password of the database in utils/SQLHelper.java. The Speechace API key should be put into the Speechace URL in the file ScoreHelper.java/SpeechAce.java.

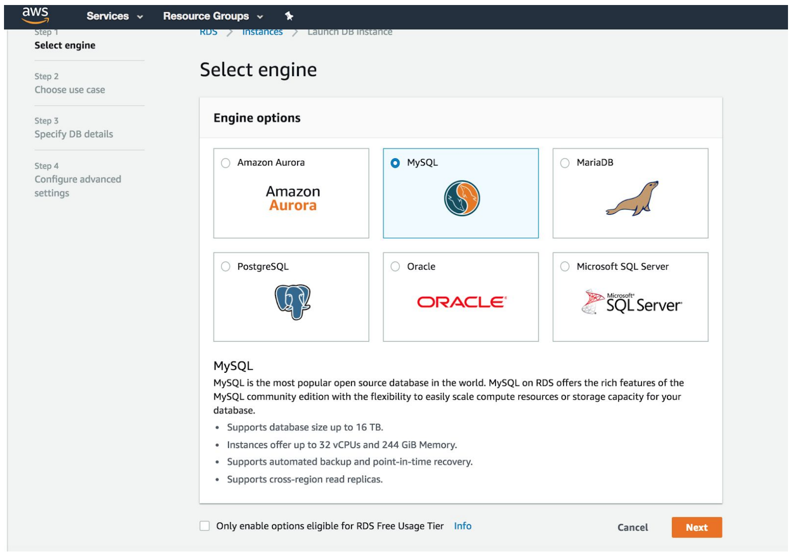
2.4.2 CHiP Robot

As far as setting up the WowWee robot out of box, there are instructions that come from the manufacturer that aid in assembly and setup.

1. First you will want to find all the wheels and wheel guards and set them aside for assembly. One at a time, place a wheel face down so the screw holes are facing up. Align the wheel guard with the holes and secure screws with a screwdriver. Place wheel on CHiP according to picture on the box and turn the wheel cap clockwise until you see the green indicator that it is secure. Repeat for all 4 wheels.

2.4.3 *Amazon Database*

1. The database is hosted using Amazon Web Services. To get started with the creation and set-up of the database, create an AWS account and log into the console.
2. After logging into the AWS console, navigate to RDS. Click on launch DB instance, and select MySQL.



Select engine

Engine options

Amazon Aurora ( ) · [✓ MySQL] · MariaDB ( ) · PostgreSQL ( ) · Oracle ( ) · Microsoft SQL Server ( )

MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 16 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.

☐ Only enable options eligible for RDS Free Usage Tier Info

3. Enter the following instance specifications. Create your own username and password to access the database. Save this information in a secure area.

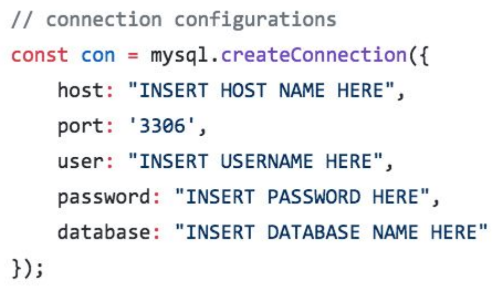
License model	general-public-license
DB Engine Version	mysql 5.6.39
DB Instance Class	db.t2.micro
Storage Type	General Purpose (SSD)
Allocated Storage	100GB
DB Instance Identifier	vobot

4. Enter the following security specifications on the next page:

Virtual Private Cloud	Default VPC (vpc-1e8e9b66)
Subnet Group	Default
Public Accessibility	Yes

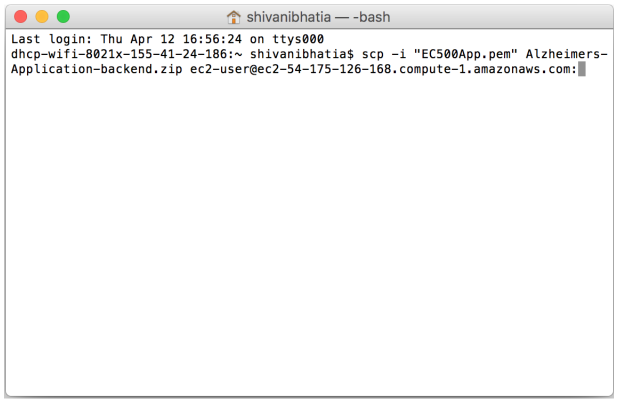
Database name	Vobot
Database Port	3306
DB parameter group	default.mysql.5.6
Option group	default:mysql-5-6
Backup Retention Period	30 days

5. Click on Launch DB Instance. This will create your database instance.
6. Next, navigate to EC2 on your AWS console. Click on Launch Instance and select Amazon Linux AMI 2017.09.1 (HVM), SSD Volume Type.
7. Select General Purpose t2.micro as your instance type.
8. Configure your security group to "All traffic" and set the source to 0.0.0.0/0.
9. Click on launch instance. You will be prompted to download a private key pair file. Download this and save it in a secure location.
10. Navigate to https://github.com/priyakapadia/Vobot/tree/shivani and download the server.js file. Open the code, and replace the connection configurations with your RDS connection configurations.



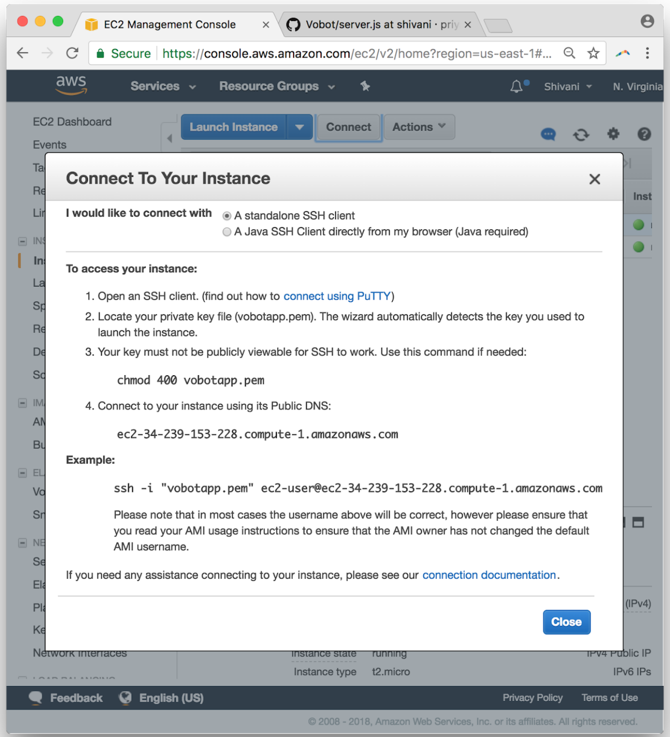
```
// connection configurations
const con = mysql.createConnection({
    host: "INSERT HOST NAME HERE",
    port: '3306',
    user: "INSERT USERNAME HERE",
    password: "INSERT PASSWORD HERE",
    database: "INSERT DATABASE NAME HERE"
});
```

11. Save the file. Navigate to the location where your private key pair from earlier is saved. Open terminal and type the following command:
scp -i "YOUR PRIVATE KEY PAIR NAME" /path/to/server.js/file AWS EC2 instance
For example, your terminal should look like:



```
Last login: Thu Apr 12 16:56:24 on ttys000
dhcp-wifi-8021x-155-41-24-186:~ shivanibhatia$ scp -i "EC500App.pem" Alzheimers-
Application-backend.zip ec2-user@ec2-54-175-126-168.compute-1.amazonaws.com:
```

12. Once finished, log into your EC2 instance through terminal. In order to find your appropriate command, navigate to the AWS EC2 console, select your EC2 instance, and click on connect. You will see something similar to this:


EC2 Management Console
Vobot/server.js at shivani
Secure https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#...
aws
Services
Resource Groups
Shivani
N. Virginia
EC2 Dashboard
Events
Launch Instance
Connect
Actions
Feedback
English (US)
Privacy Policy
Terms of Use
© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Connect To Your Instance

I would like to connect with (● A standalone SSH client / ○ A Java SSH Client directly from my browser (Java required))

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (vobotapp.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 vobotapp.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-34-239-153-228.compute-1.amazonaws.com
```

Example:

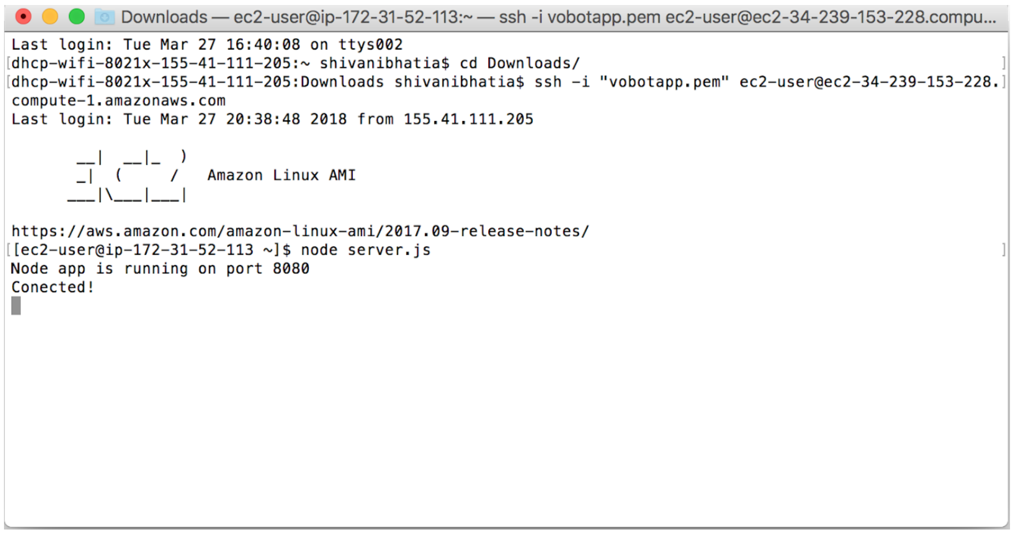
```
ssh -i "vobotapp.pem" ec2-user@ec2-34-239-153-228.compute-1.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation.

Close

Copy the ssh command into terminal where your private key pair file is stored. Once logged in, your terminal will show you the following:


Downloads — ec2-user@ip-172-31-52-113:~ — ssh -i vobotapp.pem ec2-user@ec2-34-239-153-228.compu...

```
Last login: Tue Mar 27 16:40:08 on ttys002
dhcp-wifi-8021x-155-41-111-205:~ shivanibhatia$ cd Downloads/
dhcp-wifi-8021x-155-41-111-205:Downloads shivanibhatia$ ssh -i "vobotapp.pem" ec2-user@ec2-34-239-153-228.
compute-1.amazonaws.com
Last login: Tue Mar 27 20:38:48 2018 from 155.41.111.205

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/
[ec2-user@ip-172-31-52-113 ~]$ node server.js
Node app is running on port 8080
Conected!
```

13. After logging into your EC2 instance, use the following instructions to download node.js in the instance: https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html
14. Next, run the following in terminal:
    npm init
15. Running the above command will create a package.json file. Continue to click enter on your keyboard until you are prompted to exit the file.
16. In order to run the database, run node server.js in terminal. This will launch the server and database, allowing the app to function properly.
17. If you need to close the database at any time, Control-C on your keyboard will close the connection to the database. In order to logout of your EC2 instance, type logout into terminal.

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

After installation and set-up of both hardware and software is complete, learning sessions proceed in the following systematic way *(audio from phone should be turned to high before starting)*:

1. Upon launching the app from the cell phone user will be brought to the first splash screen prompting the entry of a telephone number. This step will relate all following session information with that particular user and will only appear upon first time use. *See Figure 1.*

2. Next we see the main menu for interacting with the robot. From here we can select to start a lesson by clicking “Learning Session”, track child’s progress using “Graphs” or manipulate CHiP’s settings with “Interaction” and “Settings”. *See Figure 2.*

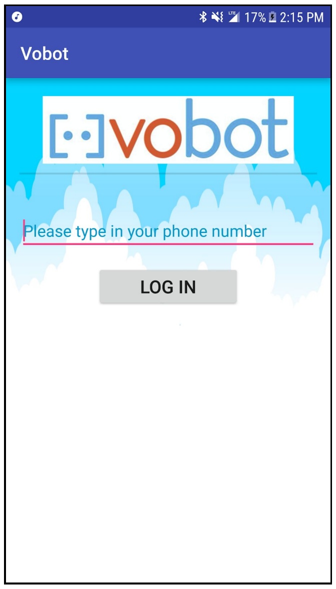
3. To start a session we click “Learning Session” and we now see a drop down menu asking us to choose a word. Once we’ve selected a word to practice we hit the “Start” button and the model word is said twice through the phone speakers. All of the child’s attempts are continuously recorded, each one being saved separately. After 5 attempts the model word is repeated again. With each vocalization scores are computed in the background. *See Figure 3.*

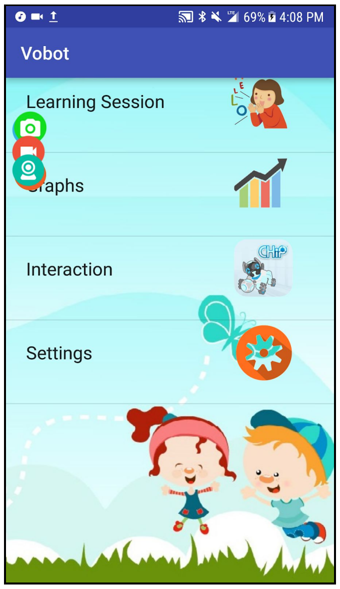
4. On any attempt in Level 1 of a word CHiP will dance for 5 seconds. In the following levels, reward threshold increases such that CHiP will only dance if an improvement in score has been made. Improvements must be made in sets of 5 such that the child is not advancing levels via trial-and-error.

5. To halt recording and end a session press the “Stop” button.

6. To see scores from completed sessions navigate to the main menu and select “Graphs”. Here you can find scores for every attempt in a session. *See Figure 4.*

In addition to the lesson learning page, the application features “Interaction” and “Settings” pages where users can manually control CHiP’s actions such as movements, sounds and driving modes. The “Settings” page in particular holds controls for CHiP’s eye color and volume levels.





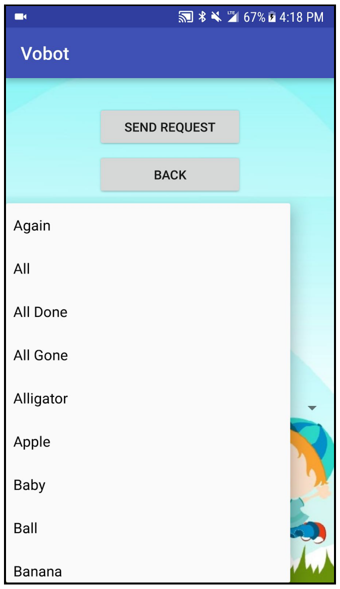


Figure 1 *Figure 2* *Figure 3*

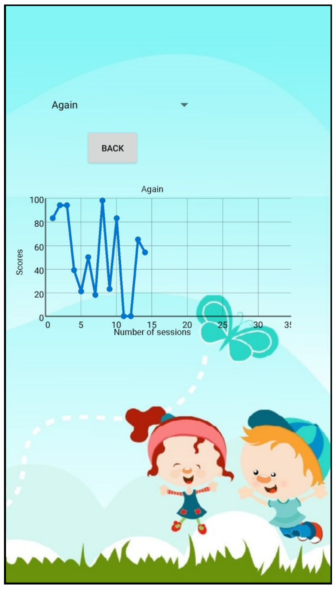


Figure 4

3.2 Operating Mode 2: Abnormal Operations

During normal operation the application should run smoothly and bug free such that the user is able to continuously run through lessons and track progress. While we have tested the application for bugs there may still be issues that present themselves upon unique user usage.

Due to the nature of the way in which we process child articulations, if a user does not have strong Wi-Fi signal at the time in which lessons are occuring there may be no response from CHiP in terms of a reward. This occurs because without Wi-Fi, the third party system that returns a similarity score would not receive the audio packet and our system would have nothing to base a reward off of. On a similar note, speech processing is done externally and thus we have no control over what specifically is detected as “correct” speech versus “incorrect”. The scoring of intelligible speech (complete vocalizations) can sometimes render false positives wherein a score is close to correct although the vocalization was wrong. The software has a relatively low mistake ratio but we have experienced false positive cases. This technology is still being developed and is not perfect.

Additionally, since the entire backend of this application lives on a database, it should remain active at all times once the application goes live on an app store and users are downloading it. If a user is unable to enter their phone number or access the “Graphs” menu item, this is an indication that there is something wrong with the database backend which allows for those items to be functional.

There is no diagnostics/recovery mode on this application. If users run into problems they are encouraged to quit the application and relaunch to clear any other issues during sessions. We will include an FAQ resource page on the code repository and update that as users report bugs and ongoing issues with the application.

3.3 Safety Issues

While there are no inherent dangers with this system, it should be noted that the CHiP robot, as with most electronics, should not be placed within any bodies of water or places with extreme heat. Operation of the robot can be manually controlled by the user and as such caution should be practiced so as to not run into fragile objects in its environment or intentionally harm others.

4 Technical Background

The minimum requirements for our system are to be able to process a child's vocalization, produce a similarity score of the vocalization compared to a "model word", provide an appropriate response through the robot, and create an interface for the child's parents and therapist to monitor progress. Based on these requirements the project was broken down into three main components: an application, a database, and a speech recognition scoring algorithm.

In order to satisfy the scoring component we partnered with Speechace, a third party company dedicated to developing speech recognition software designed specifically for language learners. Their patented technology is able to score a user's speech and pinpoint individual syllable and phoneme level mistakes in their pronunciation in real time. They were chosen for this project among many other options for having the most advanced accessible technology and availability to share API.

Regarding the acoustic-based neural network, it is the mechanism that allows our system to process and score vocalizations that contain unintelligible speech. It is critical to be able to probabilistically determine incomplete vocalizations instead of automatically correcting them to words they most sound like because children with ASD often vocalize unintelligible speech. We have implemented this technique with the help of Speechace's API. Specifically, we make cURL requests to communicate with the service, sending a recording and receiving a JSON object containing vocalization breakdown information (e.g., similarity score, syllable score). This current system we have implemented needs an API key from Speechace - which while they they have provided to us free-of-charge, will most likely not be the case for other users.

The team decided that it was best to create a phone application to have a central place where interactions and progress would live and be controlled. After looking at the compatibilities of the CHiP Robot SDK and the speech processing API, we decided on an Android application.

The application was developed using Java in the Android Studio environment and uses Gradle, which is an open source build automation system. Java is used for the functionality of the application, and XML for the view or display of the application. The app is built using Gradle and can be uploaded any Android Phone. The home page uses the CHiP library to automatically detect any CHiP robots available in the bluetooth connectivity range. To prompt the robot to carry out other functionality, the library sends a command to the sensors inside CHiP.

The choice to develop and implement a database stemmed from wanting to handle simple user authentication as well creating a backend where children's progressions across words would be pushed to and pulled from during learning sessions. This adds a layer of simplicity for the application user, and also serves to slim down the application size so that all the years of data are not stored locally on the device. The ability to pull and push information from the database allows Vobot to create profiles for each individual user and designate learning data as it corresponds to individual users.

As far as legal issues, we do not store any sensitive child information on this application and thus do not need special permissions or data encryption. The speech processing software we use is not open source as of our project release and thus licensing and ownership would need to be configured with Speechace directly.

Our application itself is open source, available on Github, and has the Apache License 2.0. This means it is a permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Ownership of the WowWee robot and its maintenance falls to the client, Andrey Vyshedsky. Maintenance on the database entails making sure it is not overloaded by user calls and adherence for AWS payment procedures and protocols.

5 Cost Breakdown

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	**Quantity**	**Description**	**Unit Cost**	**Extended Cost**
1	1	WowWee Robot	$149.95	$149.95
2	1	Phone application	Free	Free
			Beta Version-Total Cost	$149.95

For this project there are only two main components which are client facing, the robot and the application users download to their phones. The application will be released as freeware given indications from our client. On a release version later on, the robot and application will most likely be bundled together after partnership agreements are made with the manufacturer.

6 Appendices

6.1 Appendix A - Specifications

Requirement	Value, range, tolerance, units
Weight	< 3 lbs
Networking/Connectivity	Wi-Fi on 802.11ac, BLE
Recording Frequency	16KHz
Recording Duration	< 5 sec
Recording Format	.wav, .mp4
Response Time (speech recognition)	< 1 sec
Response Time (child reward)	< 2 sec
Similarity Score	Measured between 0-100%

6.2 Appendix B – Team Information

Name	Phone Number	Email Address	Role	Description	Future Endeavours
Laura Salinas	(305) 904-2582	laums@bu.edu	Similarity Score Algorithm Co-Lead, Application Quality Co-Lead	Computer Engineering, '18	Amazon (AWS) Solutions Architect
Shivani Bhatia	(267) 885-8880	scbhatia@bu.edu	Database Lead	Computer Engineering, '18	Adobe Technical Consultant
Steven Graham	(860) 387-8813	smgraham@bu.edu	Similarity Score Algorithm Co-Lead, Speech Signal Processing Lead	Electrical Engineering, '18	
Priya Kapadia	(551) 999-1648	priyak@bu.edu	Android App Co-Lead, integrating with Similarity Score Algorithm and Databasing	Computer Engineering, '18	
Arley Trujillo	(786) 715-1355	arley78@bu.edu	Android App Development Co-Lead	Computer Engineering, '18	Teach for America