Test Report

To: Professor Pisano

From: Arley Trujillo, Shivani Bhatia, Laura Salinas, Priya Kapadia, Steven Graham

Team: 14

Date: 04/01/18

Subject: Functional Deliverable Test Report

---

1. Project Objective

1.1. The overall objective of this device is to help children on the autistic spectrum develop their articulation and communication skills.

2. Test Objective & Significance

2.1. CHiP

2.1.1. The purpose of CHiP in this functional test is for it to perform the reward action of dancing to a song for approximately five seconds. This action is **only** triggered once the threshold score has been met according to the parameters we set in the database. The update for CHiP during this testing comes in the form of levels that were implemented across the application and database. These levels now make it so that CHiP will initially perform the reward action from any vocalization it hears, and then as more practice with the same word is recorded, the database will aggregate the results and only trigger an action if the level is higher than previous session scores.

2.1.2. As of this test CHiP is fully functional for real world user testing and completes the requirements for a complimentary robot per our client's instructions. This deliverable was essential because it ties together the vocalization evaluation with the reward behavior that motivates the child to continue learning.

2.2. Android App

2.2.1. For the functional test the Android application had to have all previous software deliverables completely integrated in one easy to use package as per our client requirements.

2.2.2. Design: In terms of design, the application runs through four main pages that users cycle through during session interaction. The current layout for

the application utilizes simple click through buttons that clearly point the user to the desired functionality. We wanted the application to have a sophisticated feeling but also indicate that it is a learning tool for children being used by adults. With these ideas in mind we want the application to have a friendly, fun feel and not be completely serious.

2.2.3. Speechache: In its current form the application fully supports speech analysis for word similarity by employing Speechace API. The API is used twice per loop, first to provide a clear model pronunciation of the word to be learned in US English, and second to actually perform similarity analysis for the recorded vocalization on the phoneme level.

2.2.4. Leveling: To satisfy the need to monitor a child's progress over time we followed our client's instructions for creating a leveling schema. The schema proceeds as follow:

- ❏ Level 1: CHiP will dance for any vocalization, and will progress to Level 2 alongside rewarding the child after 3 vocalizations.
- ❏ Level 2: CHiP will dance for more improved vocalizations (quality score > 10). If after 3 attempts, the child's average quality score is greater than 10, they will progress to the next level and are rewarded; however, if the quality score is less than 10, the child returns to Level 1.
- ❏ As the levels increase in number, the threshold quality score to advance increases as well. If the quality score > 90 on an attempt, CHiP will reward with a dance.

2.2.5. Reward: Once the vocalization is scored and pushed to the database it is checked against previous scores for that word and if the threshold is passed after level 1 the score is sent to the application for display and CHiP is triggered to perform the reward action. Current reward action is a dance accompanied by an instrumental song. If the score is not satisfactory, ie not better than previous attempts, CHiP will not perform any action. After three unsatisfactory attempts the level will be decremented.

3. Equipment & Setup

3.1. CHiP

3.1.1. As mentioned in the test plan, the equipment for this test included WowWee's CHiP robot without additional modifications.

Figure 1: CHiP Robot

3.1.2. During testing CHiP performed well but we observed instances when CHiP performed the reward action when a word that was *not* the model word was spoken. This will require further tweaking as we work more closely with Speechace, the company whose API we are using to dictate word similarity.

3.2. Android Cellphone

Figure 2: Cellphone model used

3.2.1. The phone application is currently set up only for Android cellphones and we use a Samsung Galaxy S6 for testing purposes.

3.2.2. During the test we ended briefly using two Android devices, one to highlight the first log in screen for users to enter their phone number (which only occurs one time) and the other to actually run through learning sessions of our system.

3.3. Laptop (Database Purposes)

3.3.1. The laptop is used to start the instance of the back end server on Amazon's EC2. To confirm the contents of the database we can use Postman to retrieve information using the 'get' command, and push information using the 'post' command.

3.3.2. During testing Professor Pisano highlighted the importance of not starting the ECE demo from level 1 on the database since it would take too long for users to level up and notice differences in CHiP's behavior (reward vs no reward).

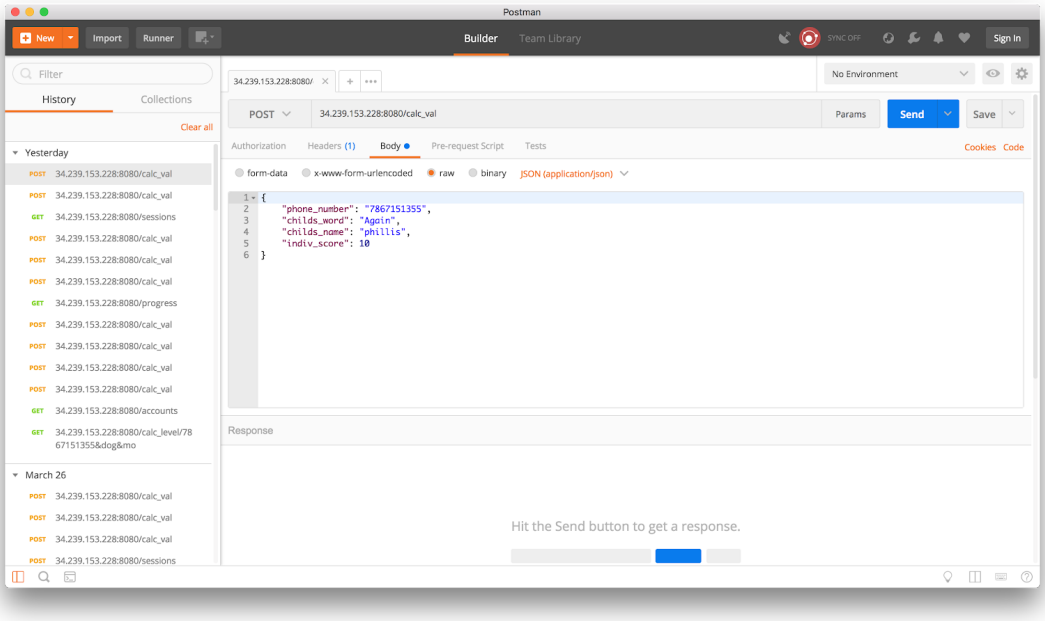

Figure 3: Postman showing user information

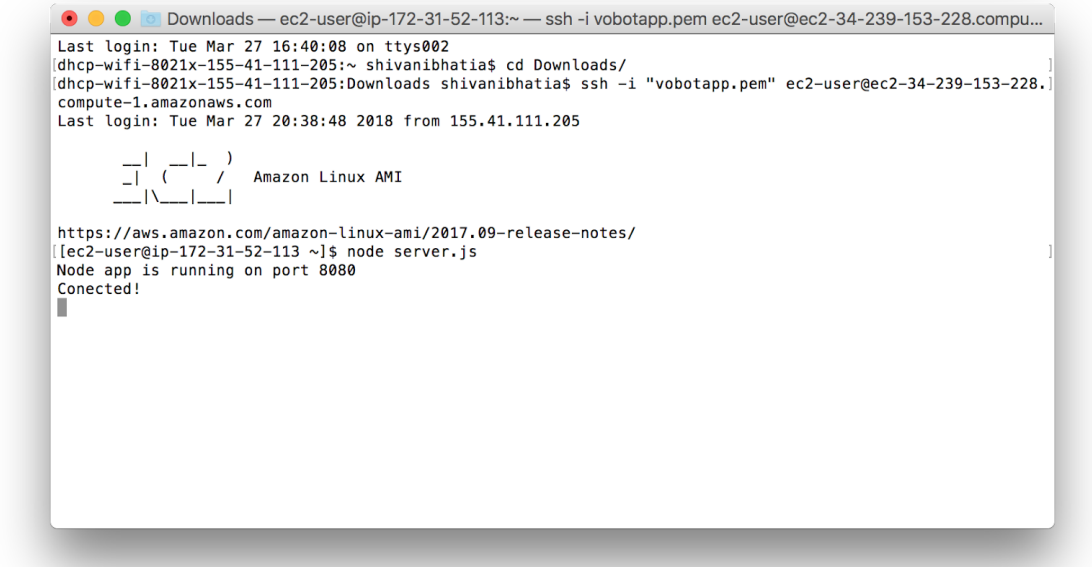

Figure 4: Initiate instance of server backend through terminal

4. Measurements & Data

4.1. Audio Quality & Speechace Scores

4.1.1. Throughout our time testing on Android, we had been receiving poor scores for excellent vocalizations of words, which confounded us - there was, at times, little-to-zero noise and the scores would return absurdly low (in the 20's) regardless. We initially believed this to be a serious noise issue in which we embarked upon a journey of filter design, after taking a much closer look at the code written for recording and saving files that were subsequently sent to SpeechAce we noticed that we were only

recording at a sampling rate of 8kHz. This was the first of a few fundamental errors, we were also not sending the audio encoding formats accepted by SpeechAce (.wav or .mp3), and had not set a bitrate associated with the audio sample.

4.1.2. In the end, it was a quick fix after all and looks something like this in our application:

```
recorder.setAudioSamplingRate(44100);
recorder.setAudioEncodingBitRate(384000);
```

4.2. Level Data Pull/Push

4.2.1. The database contains various fields of information about the child's sessions and progress including, but not limited to, the child's word, overall similarity score, overall session number, as well as the level the child is on for the specified word. Following the leveling system specified above, the Node.js server that is connected to the database calculates the appropriate level for the child, and determines whether the child can increase a level. This information is then pushed to the database using a POST command. In order to check if the database has updated the level, similarity score, and session numbers, a GET command is called through Postman to view the contents of the database, which will return this information in a json format.

4.2.2. In order to test the leveling system and view the contents of the database for our application, multiple sessions for the same word were run in the Android application, which then pushed information to the database. In between each session, a GET request was then made using Postman, so that the contents of the database would be seen. It can be seen whether the leveling system works properly by viewing the progression of the level field in the database through the GET requests.

5. Conclusions

5.1. Leveling Schema

5.1.1. Progress with the database was extensive for this final testing and we have successfully integrated the database with the application and are able to smoothly pull learning sessions information to display on the phone.

5.1.2. Taking feedback from the professors during testing, we will me making changes prior to ECE such that demos done that day will start off from higher levels so that not every vocalization receives a reward. Since the threshold for levels are done through the database, it's as simple as just changing the numbers in the database before we put out the product for display that day.

5.2. Similarity Score

5.2.1. While we have been using Speechace API since the start of this project and determined it to be the best available option for what we needed to accomplish, there were instances during the functional test that indicated speech processing was not occuring perfectly on their end. We have gone through several tests with Speechace by saying portions of as well as complete words and getting back scores that seemed reasonable and accurate for what we were inputting. However, during testing it became clear that for longer words or words with similar syllables to the model word, Speechache was still returning scores in the ranges of 50+ when in reality it should've been returning much lower, if any, scores.

5.2.2. Our next step before handing off the project to our client is to reach out to Speechace and show them demos of our project indicating the issues we are noticing with similarity score and see if they can offer any advice for improvements given the importance it has for our project.

5.3. Real World Reward Action Test Cases

5.3.1. Based on the functional requirements for this objective, we can say that we have successfully been able to deliver positive reinforcement actions from the robot based on word score. Moving forward we need to test how the actions affect the children they would be applied to. This level of testing would most likely happen during customer installation when we can deliver the full system package to a therapist that would perform controlled testing with children on the ASD spectrum. Based on the information gained from this step we can work on changing the default reward action or focus on indicating to adult users that they can change the action at any point from the main application menu if their child indicates they are more stimulated by something other than the default.