Text-it-Loud!: Real-time Captioning and Transcribing App for Inclusive Teaching-Learning of Hearing impaired

Dr. Prasanna J. Shete
Dept of Computer Engineering
K.J. Somaiya College of Engineering, Vidyavihar
University of Mumbai
Mumbai, India
prasannashete@somaiya.edu

Pulin D. Shah
Dept of Computer Engineering
K.J. Somaiya College of Engineering, Vidyavihar
University of Mumbai
Mumbai, India
pulinshah07@gmail.com

Pravar U. Parekh
Dept of Computer Engineering
K.J. Somaiya College of Engineering, Vidyavihar
University of Mumbai
Mumbai, India
pravar98@gmail.com

Jaineel N. Shah
Dept of Computer Engineering
K.J. Somaiya College of Engineering, Vidyavihar
University of Mumbai
Mumbai, India
jaineelchamp@gmail.com

***Abstract*—These days variety of audio-visual presentation tools are used for teaching the millennial learners. Although such tools improve the active participation of students in teaching-learning, these are least useful in scenarios where hearing impaired students are also part of the same class; a lecture delivered using presentation slides turns out to be a bulleted text appearing on the screen for a hearing-impaired candidate! Thus, these students may be left out from rest of the class if teaching-learning tools are not made accessible for them.**

In this work we propose an Android application "Text-it-Loud!" that renders speaker's verbal content as text, in real-time along with the audio-visuals. Although, there are variety of tools available for speech-to-text conversion, our application is novel in the sense that it displays the speaker's voice as caption text in the foreground of any application. e.g. In a presentation the verbal contents of speaker will pop-up on top of slide contents. Further, the proposed tool provides additional features like generating transcript of lecture and participation in group discussion and thereby attempts to improve the learning experience of hearing-impaired students.

Keywords— Inclusive Education, PwD, hearing impairment, Accessibility, Android

I. Introduction

As per Census 2011, Persons with Disability (PwD) constitutes 2.21% (i.e. around 2.68 Cr people) of the Indian population. Among the total disabled in the country, 19% are with disability in hearing [1]. Recent years have also seen increasing enrolment of PwD students in higher education. There are 74435 PwD students enrolled for higher education in our country, wherein approximately 14,000 are hearing impaired [2]. Due to challenges faced by the hearing-impaired students in classroom teaching, learning curve for such students becomes so steep that many are not able to cope up and thus fallback from the race [3-4]. One of the major factors for their tribulations is that, majority of education is through verbal lecture delivery. Although, audio-visual presentation tools are used, these are least useful for such students; e.g. a lecture delivered using presentation slides turns out to be a bulleted text appearing on the screen for a hearing-impaired candidate! With lack of ability to hear, the only option left for these students would be to read the lips, which would become difficult and turn out to an entire new problem for them. Thus, for providing inclusive education environment to these students special measures need to be taken [4-6].

Usage of hearing assistive technology (HAT) that includes specialized hearing aids, hearing loops, tele-coils, voice recognition software, screen captioning, Communication Access Real-time Translation (CART) systems etc., can improve the classroom learning experience of hearing-impaired students [7-8]. However, these solutions are either expensive or require trained man power and hence not available to everyone/everywhere. The aim of this work is to develop a real-time captioning and transcribing system that is inexpensive, easily deployable anywhere without requiring specialized hardware or skilled professionals. This will be achieved by developing a real-time captioning and transcribing application for widely used Android devices.

Rest of the paper is organized as follows. Section II presents discussion on speech recognition and transcription and introduces proposed Text-it-Loud! App highlighting its features. System Architecture of proposed Text-it-Loud! app and implementation details are presented in section III. Results are presented in section IV and section V concludes the paper.

II. Text-it-Loud!: Real-time Captioning and Transcribing App

Our objective is to empower the students with hearing disabilities to learn together with their non-hearing-impaired peers by employing real time speech-to-text captioning and transcription. There has been a plethora of research on speech recognition and various implementations have been adopted throughout the years in applications ranging in variety of fields [9]. The prevalent speech-to-text conversion software tools available from various vendors can generate transcript of spoken content. However, these tools are not meant for real time captioning and thus have limited usage for improving classroom experience of the hearing impaired. Thus, for improving classroom participation of hearing-impaired students, only viable solution is to provide live screen captioning and on-demand transcript generation that supplies the speech info as text without losing its context and maintaining spontaneity of conversation. This is the basis for design of our proposed "Text-it-Loud!" Android application. By utilizing the Google speech to text API, the proposed app provides speaker's verbal content as live screen captions as well as transcript. Text-it-Loud! not only assists the hearing-impaired students in classroom learning but also facilitates active participation of hearing-impaired candidates in meetings or group discussions. Further, it can also act as a

utility for listeners not well versed in English by providing real-time captioning and transcript of the speech.

Following features highlight the novelty of proposed Text-it-Loud! App.

1) The application can run in the background like a service: User (both Listener and speaker) can open other documents, files, presentations to view and our application will be running in background generating spoken messages.
2) Secure as every group created will have its unique group ID, which will be provided to other users for joining the group.
3) Generation and downloading of transcript: After seminar/meeting, one can view and download a transcript of the whole monologue/dialogue.
4) Can be used from any corner of the world: As its backend is based on firebase. One can attend a particular seminar or hear the meeting from any corner of the world.
5) Dynamic generation of Toasts: Even if a sentence is too long, it will automatically show toasts of sentences at regular intervals of time for user friendly reading experience.
6) Dynamic removal of user if app is stopped: Suppose a user closes the application directly by clearing the cache (removing application from the recent applications). His name from the list will be automatically removed.

III. Implementation

Our goal was to develop an Android application that is both lightweight and universal. Potentially, it would be used by many people, mainly teachers, hence the application should necessarily work on a large range of devices, and here we mean, the oldest devices to the newest ones. Another functional requirement of the application was its availability in paucity of the Internet, as the lack of Internet — which is a common occurrence — should not hinder the usage of the application. After some research [10], it was decided that Android's inbuilt "Speech Recognition" service available in API level 8 of Android 2.2 – Froyo, would be the best solution since it is available even without the internet. Hence, even in the oldest devices, the application would still function as required.

Google's inbuilt Speech to Text converter is very fast, smooth and flexible due to the range of options provided by it [11]. One of them being confidence scores, i.e. for an input audio or speech, it could get several possible conversions, and would give the correctness of each of the conversions based on confidence scores from 0.0 to 1.0. Another option was to provide partial results, wherein it would provide conversion word by word instead of the entire sentence after its completion. Other options included language, minimum duration for recognition, maximum duration of silence, etc.

To extend the usage of our application to people who wish to remotely participate, we added an option to use the application in a group. A user would create a group and others would join it via a key. Everyone in the group will get the converted text over the internet.

Description and working of important modules: -

A. Home

The home activity of the application functionally provides 4 options: User manual, create a group, join a group, or offline mode.

B. Create a Group

Any user can create a group. 3 required input fields are; the name of the user, password and the name of the group. There is a non-editable, randomly generated 5 letters "group key". The group key consists of 5 letters and each of the letters are randomly generated using Random.nextInt(26) function in Java. It basically generates a number between 0 and 25, and to convert it to characters, we simply add 97 to it (97 to 122 numbers corresponds to ASCII values of alphabets) and typecast them to characters.

C. Join a Group

The user, who created the group, needs to share the randomly generated group key to others. After receiving the group key, the user has to enter their name, the group key, select the "Listener" option, and then tap on the "JOIN" button. After a moment, the user is logged in. Now, to allow the person — who initially created the group — to join the group again (if not in the group already), he/she must select the "Speaker" option from the drop down and enter the password, which was set when he/she created the group. In this way, it is possible for the speaker to join the group again as the speaker, even after leaving it unintentionally.

There are some rules implemented to restrict unauthorized users. The first and foremost is a valid group id. Second, except for the Speaker, the name entered should not match with the name of any user already in the group. For the speaker, the name entered must match with the name of user who created the group, and the passwords should also match. If any of the conditions are not satisfied, an appropriate error message is displayed.

D. Recognition and Display

After creating a group or joining a group, all users are redirected to this activity. At the top, the name of the group and the group ID is displayed. Below is the list of users, or the users currently present in the group, and at the bottom, are the "Pause" and "Exit" buttons. The pause button is disabled for listeners, but available for speakers. Also, for the speaker, there is an image-button, whose image is like a mic, prompting the speaker to tap on it to speak. It is not available for listeners. On tapping, the app requests permissions from the user. The first permission is for recording audio, and the second is for accessing storage. Without granting both these permissions, the speech recognition will not work. To use the device's inbuilt speech recognition service, we need to make an API call. However, this is like an internal API call, and everything is taken care by Android SDK.

Our aim was to have the application work in the background, i.e. work over other applications. The application uses the "SpeechRecognizer" interface. Before recognition starts, the options discussed above are to be set. This is done by using an Intent. Specifically the RecognizerIntent is used. An object is created and initialized for speech recognition by

ACTION_RECOGNIZE_SPEECH option. Now to attach the extra options, also known as "Extras", putExtra() method is used, with the option as a parameter [11][12]. We used the "EXTRA_LANGUAGE_PREFERENCE" option, and specified "en" (for English) as its value and the "EXTRA_PARTIAL_RESULTS" option with "true" as its value.

Owing to significant developments in the speech recognition field supported by Machine Learning and Artificial Intelligence, Google has deprecated the inbuilt speech recognition service several years back. Therefore, some of the options didn't support the Android OS developments past Android 4. The most important option was the maximum duration of silence, which allowed the speech recognizer to listen or keep working — even when there was no input speech/sound — until the maximum duration timer expires. The speech recognizer stops after performing speech recognition once. Hence, for continuous speech recognition, every time it stops, recognition object is destroyed and reinitialized. Finally, as shown in the Fig. 1, with the RecognizerIntent object available, we start recognition by calling startListening() method through the SpeechRecognizer object and pass RecognizerIntent object as parameter [12, 13].

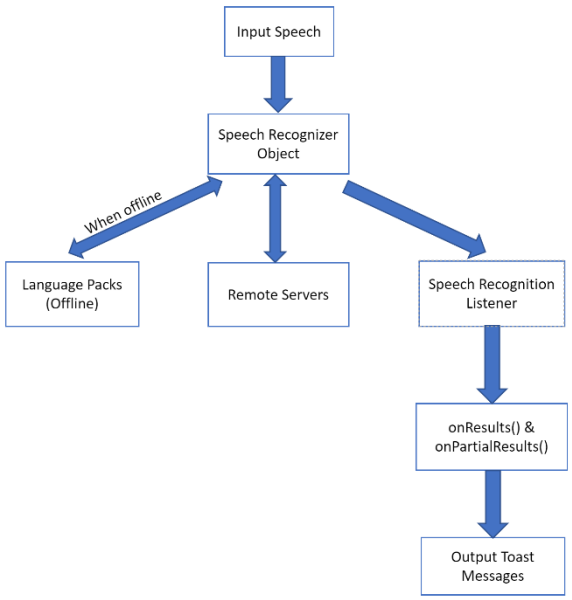

Fig. 1 Internal Processing for Speech translation

If we were to display a new Toast message for every word in a hundred word sentence, it would be very slow, and hence we implemented it in a way that a new Toast message is displayed when 20 or more letters are added to existing result text.

However, this process is only done on the Speaker's device, because speech input is only provided to the Speaker's device. Hence, to display the same Toast messages on other users' devices, we took help of Firebase. Firebase provides a means of storage of data on the Google Cloud. Firebase provides a free plan for budding developers which is called "Spark" plan. Android Studio automatically links our project to Firebase with one click. We use the Realtime Database, and set it up for "testing" mode and implement data transfer as shown in Fig. 2.

First, we get the reference of the root, by getReference(), and store it as object of DatabaseReference. Now we get the remaining references by traversing the tree using the child() method, and passing name of child node of the tree. Example, rootReference.child("Group_Name") [14]. All these references are also stored as objects of DatabaseReference.

We attach an event listener to the DatabaseReference object (DatabaseReference.addValueEventListener()), which points to a particular point in the database. Every time, a change is detected in the children of the reference, the ValueEventListener is fired, and corresponding code is executed [14].

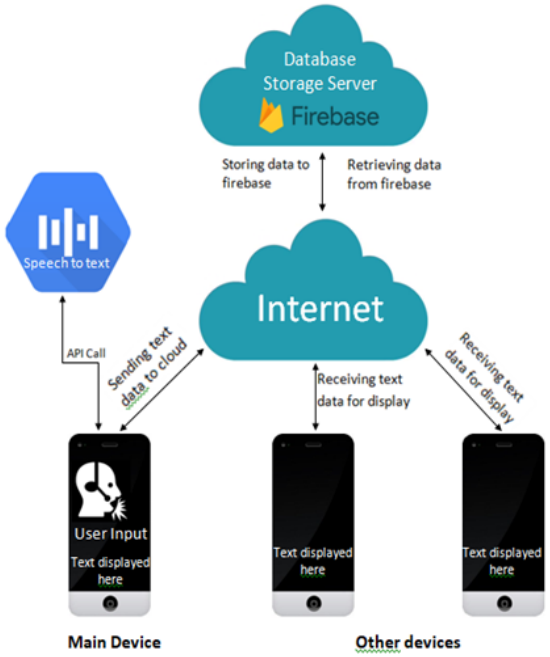

Fig. 2 System Architecture

The code inside the listener is to retrieve the data and display using Toast messages. Also, it is mandatory to override remaining methods provided by the SpeechRecognizer class.

In all, speech is converted to text, and text is uploaded to Firebase, and retrieved by Listeners.

E. Transcript and File handling

After speech recognition is performed, the generated text is stored on Firebase, as well as on the Speaker's device dynamically. Since, there is no trace of the group key once its purpose was over, the file name consisted of the group key, and hence no one has to remember the group key, and will always stay in the device. The file name can also include the group name, so it is easy for any user to identify the file he wants. A FileOutputStream is opened in the "append" mode, and data is passed through it, and written to the file. It is better to keep all the files within a folder so they are organized.

However, a provision to view the transcript was also kept available in a new activity. Once the activity is opened, the application will fetch the data stored on Firebase, and display it in a TextView. For listeners, a file is not created dynamically, and hence to get the text file of the transcript, a download button was also added. A certain logic must be

implemented, to avoid tapping on the download button multiple times, or create the same files repeatedly.

F. Dynamic User List

However, a provision to view the transcript was also kept available in a new activity. Once the activity is opened, the application will fetch the data stored on Firebase, and display it in a TextView. For listeners, a file is not created dynamically, and hence to get the text file of the transcript, a download button was also added. A certain logic must be implemented, to avoid tapping on the download button multiple times, or create the same files repeatedly.

To keep the list of users up-to-date, it is necessary to properly remove users when the exit the group, or close the application. Hence, a service was implemented, which would start running whenever a user either leaves the group, or closes the application by any means except shutting down the device. Using the DatabaseReference.removeValue() function, after assigning a DatabaseReference to the particular user, the entry of user is removed from database.

G. Offline Mode

The application is also developed for offline operation. This was done in order to extend the usability to the users who do not have stable internet connection. This mode allows the user to use the app in a standalone fashion.

IV. RESULTS

This section presents the activity flow chart along with the screenshots of user interfaces and the performance results of the application.

Screenshots from the application's user interface are presented next, which follow the activity flow chart of Fig. 3.

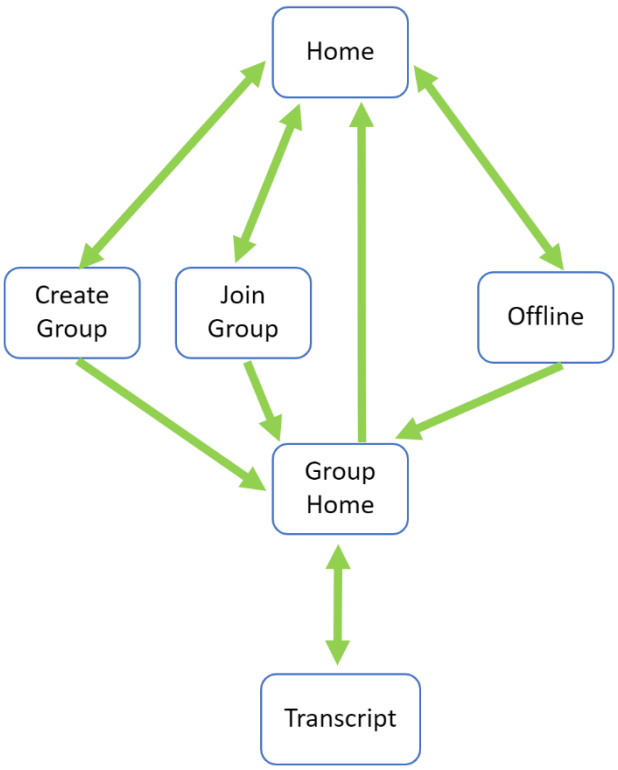

Fig. 3. Activity Flow Chart

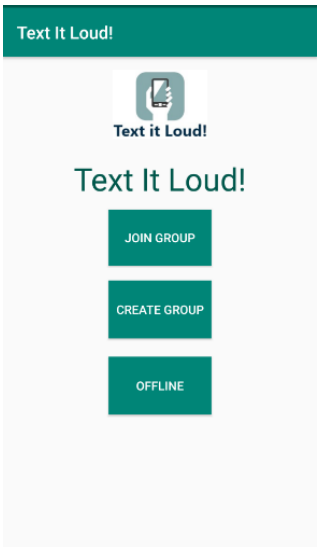

Fig. 4

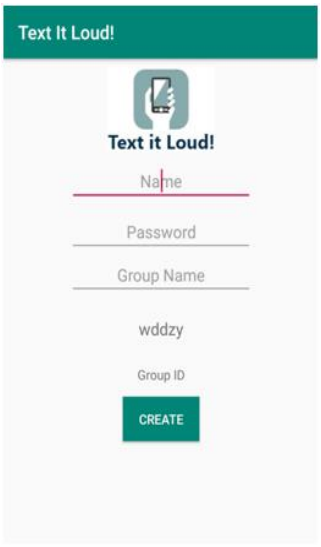

Fig. 5(a)

Fig. 4 represents home activity that is triggered on starting the application. It navigates to join group, create group, and offline activities. Fig. 5(a) represents the create group activity in which name, password, & group name are required input fields, and also the randomly generated group key is displayed.

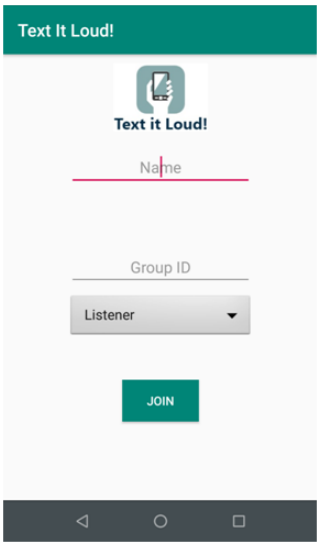

Fig. 5(b)

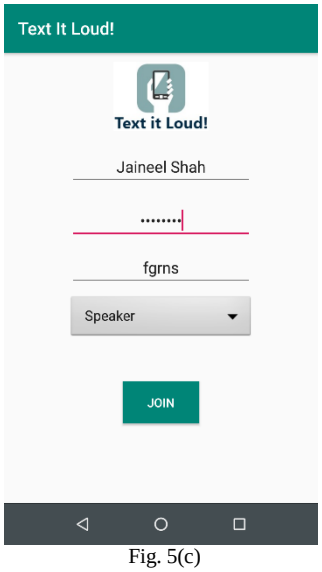

Fig. 5(c)

Fig. 5(b) represents the activity which loads after selecting the join group option. Any user who wants to join needs the corresponding group key, & if speaker is selected in the dropdown list, then password is also required, which was set while creating the group as shown in Fig. 5(c).

After proceeding to the next step from each of join, create, & offline activities, the group home activity starts. Fig. 6(a) shows the group home activity with a mic icon indicating that it is captured by a speaker. The mic icon is unavailable for listeners. Fig. 6(b) shows the transcript activity which opens when transcript button is selected.

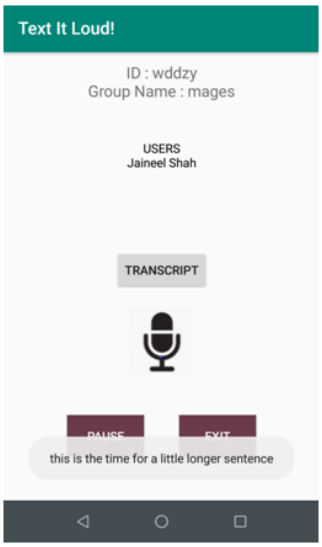

Fig. 6(a)

Fig. 6(b)

Fig. 7

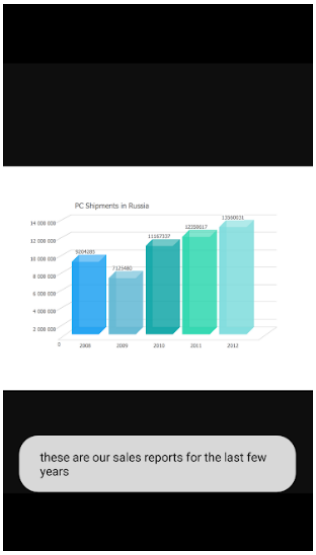

Fig. 8

Screenshot taken while using the application in offline mode is shown in Fig. 7. Fig. 8 represents a screenshot taken on a listener's device that is connected to a group of participants involved in a presentation. In both the screenshots, a presentation slide is open in the foreground, with Text-it-Loud! running in the background.

The accuracy of the application has been verified by examining the application through teachers and experts. It was reported that, Text-it-Loud! provided 87-90% accurate results in sentence transcription. The application has also been tested for hearing-impaired student of third year Computer Engineering in classroom environment. The student has hearing disability from birth and uses hearing aids since early her childhood. Whereas, a normal person can hear sounds with decibel level as low as 30 decibels, this student can only hear sounds at 105 decibel or higher. Also, the student follows only English - as it was recommended by doctors as well as speech therapists that she should learn only one language - she cannot speak her own mother tongue! Although, the hearing aid assists her by amplifying the sound multiple times, however it cannot process speech coming from different directions equally; and is the main hurdle in classroom learning. The sound signal from a teacher can often get completely drowned in background noise or distorted, before reaching her hearing aid. Thus, she resorted extensively to lip-reading as an additional means of understanding instructions. With the help of live captioning and transcript, the student has experienced significant improvement in understanding/learning of the subject matter taught in the class and has given an encouraging feedback about Text-it-Loud!.

V. Conclusion

The paper presented an Android app "Text-it-Loud!," that primarily aimed at assisting the hearing-impaired students in classroom learning. Using the Google speech to text API, the application converted the speaker's verbal content into text and rendered it in an elegant way as real time screen caption with the aid of "Toast" messages. Also, it provided the option of generating transcript of the session when required. The novelty of the application lies in its ability to run in the background and present the captions on top of any other application screen running in the foreground. Thus, Text-it-Loud! proves to be a viable hearing assistive technology for teaching-learning of hearing-impaired students.

The current implementation is restricted only to Android devices. We intend to provide an option to use Cloud Speech to Text, which will be the default speech recognition service in iOS devices. In future we also plan to extend the app to generate Braille transcript, and thus assist the blind or low vision students.

References

[1] Report on "Disabled Persons in India - A statistical profile 2016," Social statistics division, Ministry of Statistics and Programme Implementation, Government of India, 2017.

[2] Report on "All India Survey on Higher Education (2015-16)," Ministry of Human Resource Development, Department of Higher Education, New Delhi, 2016.

[3] Mahwish Safder, Mahr Muhammad Saeed Akhtar, Ghulam Fatima, Misbah Malik, Problems Faced by Students with Hearing Impairment in Inclusive Education at the University Level, Journal of Research and Reflections in Education, Vol.6, No.2, pp129 -136, Dec. 2012.

[4] Beryl Ndongwa Bamu, Elisabeth De Schauwer, Sara Verstraete & Geert Van Hove, Inclusive Education for Students with Hearing Impairment in the Regular Secondary Schools in the North-West Region of Cameroon: Initiatives and Challenges, International Journal of Disability, Development and Education, Vol. 64 - Issue 6, pp. 612-623, Apr 2017.

[5] https://www.ferris.edu/HTMLS/colleges/university/disability/faculty-staff/classroom-issues/hearing/hearing-strategy.htm

[6] https://www.adcet.edu.au/inclusive-teaching/specific-disabilities/deaf-hearing-impaired/

[7] https://www.ferris.edu/HTMLS/colleges/university/disability/faculty-staff/classroom-issues/adaptive-technology.htm

[8] https://www.hearingloss.org/hearing-help/technology/

[9] https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf

[10] https://www.simplifiedcoding.net/android-speech-to-text-tutorial/

[11] https://developer.android.com/reference/android/speech/RecognizerIntent

[12] https://developer.android.com/reference/android/speech/SpeechRecognizer

[13] https://developer.android.com/reference/android/speech/RecognitionService.Callback

[14] https://firebase.google.com/docs/database/android/read-and-write