Speech Radar

Goldsmiths University of London

Shah Ali

Reg No. 33455846

BSc Computer Science

Supervised by Jamie Ward

May 16th, 2019

Abstract

Speech Radar is an Android application that allows users to locate their phone through speech recognition. The application uses Java, XML and Firebase Database (as backend) to achieve this. Python with TensorFlow module is used to train a model for speech recognition, which will be used to make predictions in the app.

Contents page

Chapter 1 Introduction

1.1 Purpose

This report introduces Speech Radar, an Android-based application that allow users to locate their phone through speech recognition. Suppose for instance an individual must rush to work and is unable to find their phone. They know the device is situated somewhere inside their room. Speech Radar can speed up the search time just by an utterance of a specific word. Once the phone detects the utterance, it will begin ringing at maximum sound level, which will help the individual locate their phone. This app can be of use to a great number of people as it is common nowadays for individuals to lose sight of their phone and to search everywhere for it.

1.2 Background

In the area of speech recognition, it is said that Microsoft can now interpret *human speech with a 5.1% error rate* [1]. Google have enhanced its *accuracy by more than 20% in the past five years* [1]. And to date, Amazon's Alexa has been getting better at responding to users' question. *Researchers asked the voice assistant 800 different queries* [2] in various categories. *On average, Alexa answered queries accurately 73% of the time, up 12 percentage points from 61%* [2]. These are little known facts of how top companies are taking speech recognition to a whole new level and how they've made it worth looking into and investing in for personal and business use.

Since 1784, Speech recognition is something that was just a topic of talk. It wasn't until 1952 when a six-foot machine was created by Bell Labs, *capable of recognizing spoken digits with 90% accuracy* [3], but when uttered by its owner. The development would continue in 1962, where IBM created a machine the size of a shoebox that could *understand 16 English words* [3]. In 1971 a student of Carnegie Mellon University created the Harpy that could *comprehend 1011 words and some phrases* [3]. In 1986, IBM would create another ground-breaking machine that used the *Hidden Markov Model* [3] to recognise 20000 different words from various speakers and type them on paper. The list of inventions would go on with Google launching a voice search application in 2008, *bringing speech recognition to mobile devices* [3]. In 2011, Apple would announce Siri, *ushering in the age of the voice-enabled digital assistant* [3].

At present, we are seeing digital assistants decentralise from smartphones and are seeing companies primarily focus on voice-activated home speakers that can query and control smart home devices. From a subjective point of view, these innovations appear to be an approach to accumulate billions of audio data from people that have different accents, so companies in the future can improve the detection rate for fluent and non-fluent English speakers. On the off chance that this improvement happens, we will see speech recognition being used for more advanced tasks, possibly in robotics.

1.3 Aims and objectives

This section states all project aims and objectives separated in research, design, implementation and testing phase.

1.3.1 Research

- To research on a feasible approach of creating a TensorFlow model for speech recognition task
- To research on an appropriate dataset to use for training. Preferably, the dataset should contain audio files for each word in different accents, rather than phrases
- Gain insight into how the TensorFlow model could be used in Java to make predictions in the app
- Find a possible solution for continuous speech recognition to run in the background of users device

1.3.2 Design

- Design the implementation in TensorFlow using clear flowcharts
- Plan the implementation precisely in Java using class diagrams, use-case diagrams, activity diagrams and pseudocode
- Design the structure of the database using an appropriate schema
- Sketch the appearance of application using wireframes. This will be defined in XML (Extensible Markup Language)

1.3.3 Implementation

- Implement the designs using correct software
- To strictly meet deadlines for each task in this important phase of the project

1.3.4 Testing

- To perform software testing, in order to investigate the quality of Speech Radar
- To perform user testing with volunteers, in order to understand how users interact with the speech recognizer
- Gather all results to find ways in making improvements

1.4 Section overview

Chapter 2 – Consists of the research for the project through literature reviews, in the attempt to critically evaluate other people's work in the area of speech recognition.

Chapter 3 – Contains the specification of the project. The requirements are stated precisely and in detail.

Chapter 4 – The design of Speech Radars appearance and functionality is mentioned in this chapter.

Chapter 5 – The implementation of the designs is critically discussed in this chapter.

Chapter 6 – Specifies the results for user testing and analyse key areas that'll help improve the application. Software testing will also be included through black-box and white-box testing techniques. An evaluation will conclude this chapter, assessing whether requirements stated in Specification chapter were met.

Chapter 7 – Provides a conclusion for the report, with mentions of future work that can be done, successes/failures of the project and how it compared to what others have done.

Chapter 8 – This chapter includes a self-evaluation of the completed project

Chapter 9 – Includes all bibliographies used for the project

Appendix A to H – Contains work done for the project in order to support the analysis made in the chapters

Chapter 2 Literature Review

This chapter introduces the research done in preparation for the next stages of the project. It analyses relevant published articles that can have an impact on the project. The research can help to find solutions for tasks that need to be prioritised and managed carefully during implementation phase. This will help gain knowledge and prepare for any problems that could take place. The focus area is in TensorFlow and Android Studio, where implementation of the project will take place.

2.1 Speech Commands dataset

2.1.1 Broad analysis

An article composed by Pete Warden examines his speech recognition implementation as a web app, using TensorFlow to create the model. Speech Commands dataset was used to supply a way in building and testing *small models that detect when a single word is spoken* [5] from a range of target words. This task is known as *keyword spotting* [5]. The final dataset consists of *105,829 utterances of 35 words* [5]. Each of the utterances in the Speech Commands dataset are stored as a *WAVE file- format* [5] lasting for one second. The sample data is *encoded as linear 16-bit single-channel PCM values* [5], at 16,000 Hz sampling rate. Over 2600 speakers are recorded, some with different accents.

The author of the article explains in a separate section how he wanted to have *a limited vocabulary* [5] of 10 words to ensure that the web app was lightweight and fast in making correct predictions. The choice of words were chosen carefully, so they do not have similar pronunciations and are easily detected. For example, the word 'three' is pronounced as 'TH R IY' and ‘tree’ similarly as 'T R IY'. This can cause the model to make wrong predictions, especially for someone with an accent. Thus, why the author decided to avoid these kinds of words.

Speech Commands dataset would be an incredible asset to the Speech Radar project. A limited vocabulary of code words would seem appropriate and feasible than having a large vocabulary. The article mentioned only 10 words were used. The Speech Radar app will attempt to use more words from this dataset and ensure correct predictions are made most of the times.

2.1.2 Additional information

What was incredibly useful for Speech Radar was the brief mention of model being exported as a protobuf file. This led to more research being done, which provided the definition of what a protobuf file is. It basically carries the graphs definition and weights of the model, which can be loaded and used in Android Studio. Thus, a visual flowchart was created to illustrate our plan in using the model in Android Studio. This can be seen in chapter 4 section 4.1.2.

2.2 Convolutional Neural Network (CNN)

2.2.1 Keyword spotting task

An article written by Tara N. Sainath and Carolina Parada mentions how they used a CNN to solve the keyword spotting problem on a mobile device. This is an issue that manages the recognition of keywords in articulations. Virtual assistants such as Google Assistant and Amazons Alexa use keyword spotting such as "Ok Google" or "Alexa" to wake up when their name is spoken. But for this to work properly, especially with mobile devices, it must have a *small memory footprint and low computational power* [4]. Since this article has proved convincingly that CNN is the best choice for speech recognition tasks like keyword spotting, it will be analysed further to help in our implementation.

Just to summarise the article, a model named *cnn-trad-fpool3* [4] was created. It contained far too many parameters that would be *infeasible for power-constrained small footprint KWS tasks* [4]. Thus, smaller models were created. The best model was *cnn-one-fstride4* [4], which reduced featured maps to limit computation. By *striding the filter with overlap* [4], it provided the best result in clean and noisy environments. The model is so good that TensorFlow themselves used it to create an open-source simple audio recognition.

2.2.2 Wav to spectrogram

A WAV audio file can be represented as a spectrogram, which can be used as an input for a CNN. A spectrogram is an image of the signal strength/loudness of a signal over time at different frequencies present in a specific waveform. Not exclusively would one be able to see that there is energy at a certain hertz but can see how energy levels change after some time. Given that CNN's achieve great results in image classification problems, the information represented in the spectrogram is invaluable and can be extracted and learnt by the network. Figure 1 shows a spectrogram for the word ‘Marvin’. We see the frequency for every utterance needed to be made to say the word. Notice from 0.17 to 0.75 seconds, we see a high level of energy for the first part of the word pronounced as ‘M AA R’ and a low level of energy for the final part pronounced as ‘V IH N’ roughly from 0.79 to 0.95 seconds. This makes sense, since the loudness of voice for each part of the word alters i.e. goes from high to low. The low level of energy seen in 6000 hertz would most likely be the pronunciation of ‘AA’, as it is a high frequency vowel.

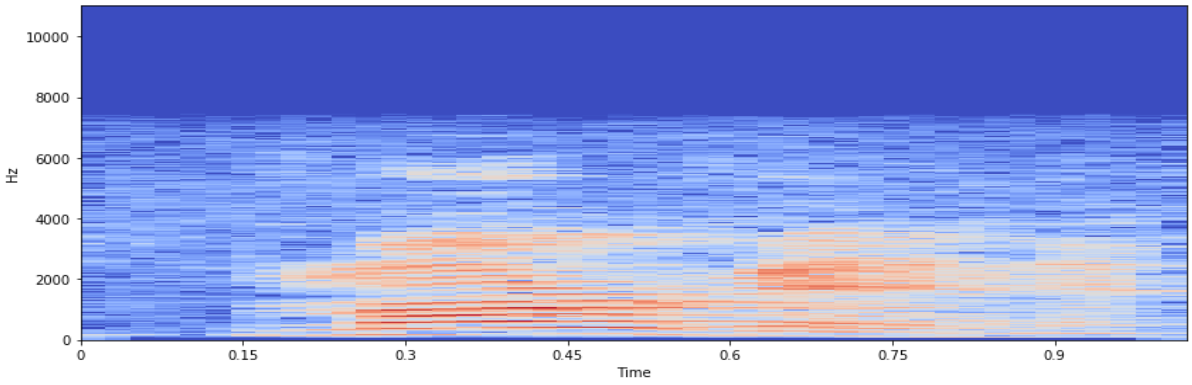


Figure 1: Spectrogram for the word ‘Marvin’

Figure 2 shows a spectrogram for another word 'Happy'. Similarly, like figure 1 we see high levels of energy and frequency for the first part of the word pronounced as 'HH AE' and low levels for the final part pronounced as 'P IY'. Notice how the frequency levels are much higher than figure 1. This tells us that the speech recognition system would be better at detecting the word 'Happy' than 'Marvin'. Thus, it would be unsurprising for the results in the testing section to show different detection rates for each word.

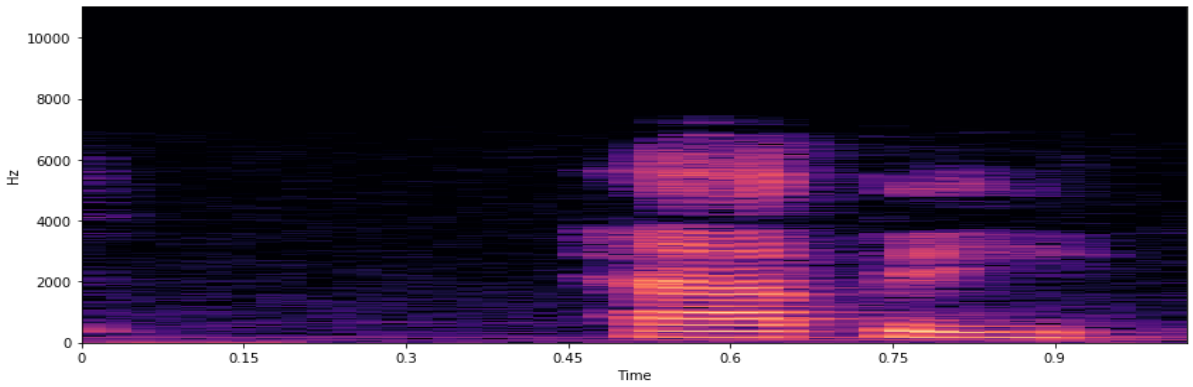


Figure 2: Spectrogram for the word 'happy'

2.2.3 Network architecture

A typical CNN architecture contains convolutional, pooling, normalization and fully connected layers. The table in figure 3 shows a basic model created for the keyword spotting task using the Speech Commands dataset (greater detail mentioned in section 2.1.2). We will call this model A. It is capable in recognizing three words i.e. bed, cat and happy, which is why the last layer contains only three neurons with Softmax as its activation function. Model A is small with only 82,368 parameters and will be interesting to see how it could be improved later.

Model	Layer	Neurons & rate	Activation function	Optimizer	Learning rate	Batch size	Parameters
A	Conv2d	32	Relu	Adadelta	0.001	100	160
	Max_pooling2d	-	-				0
	Dropout	0.2	-				0
	Flatten	-	-				0
	Dense	55	Relu				79,255
	Dropout	0.2					0
	Dense	50	Relu				2,800
	Dropout	0.2	-				0
	Dense	3	Softmax				153
							82,368

Figure 3: Model A

The graph in figure 4 shows the result for model A, where validation loss gradually reduces in each epoch. The loss for training and validation overall is very high and will certainly need improvement. A significant increase can be seen in the validation accuracy after 5 epochs, causing it to be much greater than the training accuracy. This means model A is underfitting. To avoid this, more parameters and layers can be added. Adjusting certain hyperparameters could also help (e.g. good learning rate, batch size etc.) Since a CNN architecture is being used, it might be best to add BatchNormalization layers to the model, as it works well with other regularization techniques such as dropout and L2. It can also speed up the learning process.

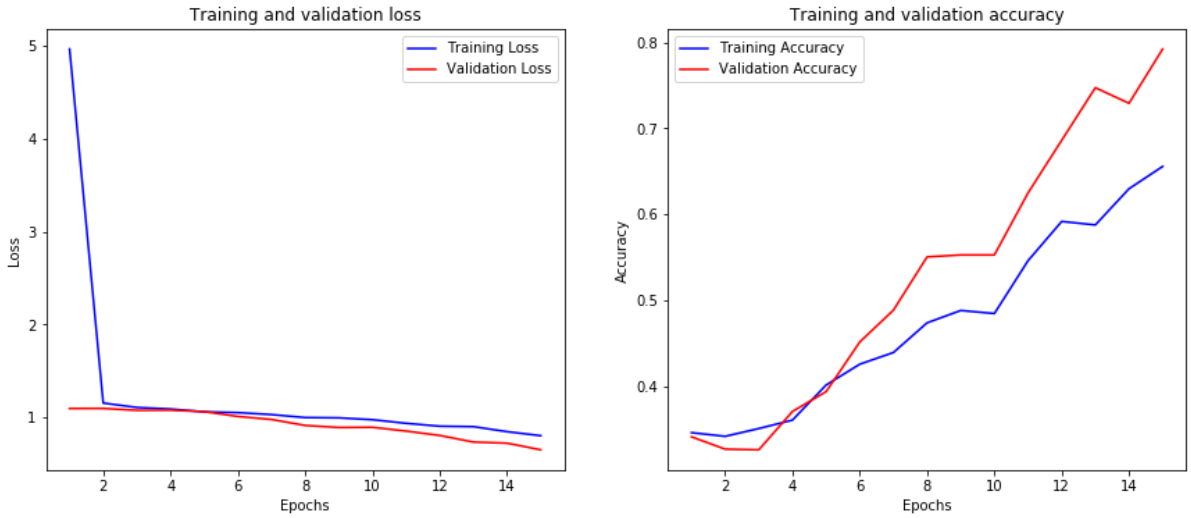


Figure 4: Model A results

The table in figure 5 shows an extension from model A. we will call this model B. Everything highlighted in bold are changes/improvements made to the model. For example, the number of neurons has increased so that model B has more parameters to avoid underfitting. Another example would be the small reduction of learning rate, which will hopefully prevent the performance from diverging and see the error rate reduce rapidly.

Model	Layer	Neurons & rate	Activation function	Optimizer	Learning rate	Batch size	Parameters
B	Conv2d	**80**	Relu	**RMSprop**	**0.0005**	**90**	400
	Batch_normalization	-	Relu				320
	Max_pooling2d	-	-				0
	Dropout	0.2	-				0
	Flatten	-	-				0
	Dense	**85**	Relu				306,085
	Batch_normalization	-	-				340
	Dropout	0.2	-				0
	Dense	**90**	Relu				7,740
	Batch_normalization	-	-				360
	dropout	0.2	-				0
	dense	**3**	Softmax				273
							315,518

Figure 5: Model B

The graph in figure 6 shows the result of model B. Lower training and validation loss can be seen compared to model A. High validation and training accuracy can also be seen. This tells us how well it can make predictions based on unseen data, which is vital for when the model is used in the Android app. Training the model in under 15 epochs was probably a good estimate, as we see the validation loss slowly diverging at the final epoch.

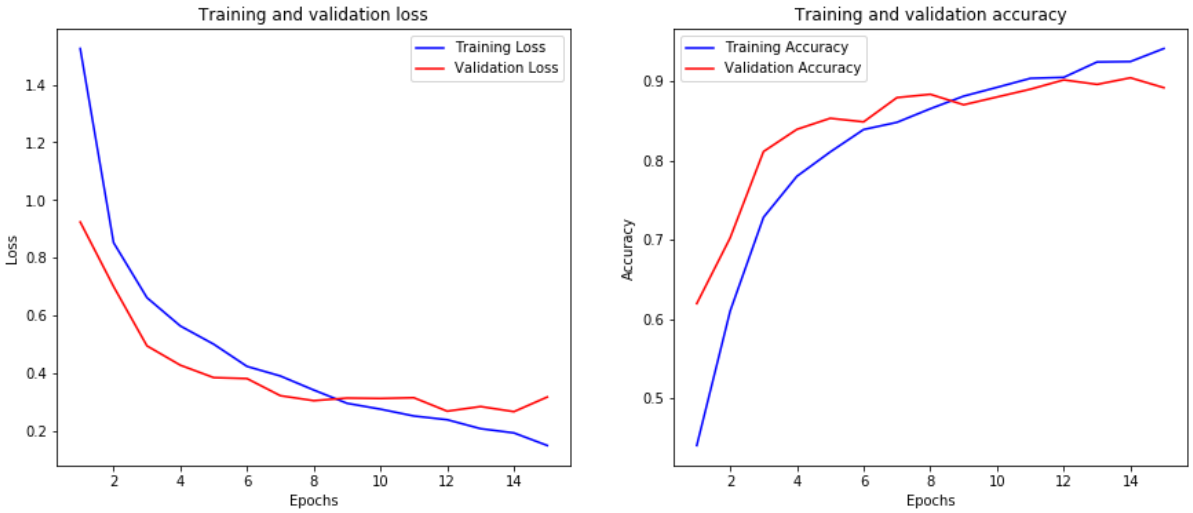


Figure 6: Model B results

2.3 PocketSphinx

2.3.1 Brief

Another article written by Ajav Sharma and Rahul Bhalley explains their implementation of a *real-time speech recognition on portable devices* [6]. It mentions a library called PocketSphinx, a lightweight engine specifically made for mobile devices. It is a collection of *3 components: front end, decoder, and linguist* [6]. Linguist contains an acoustic model (reports the sounds of words in which any grapheme is uttered), dictionary and language model (states probability of each utterance.) Signals are inputted into the front end which are parameterized into an arrangement of features. The linguist interprets the pronunciation data present in the dictionary alongside the language model data and structural data from *acoustic model, into a search graph* [6]. Decoder incorporates *the search manager which inputs the features from front end and search graph from linguist* [6]. This is where the real decoding occurs and where results are produced. These results are sent to the application.

2.3.2 Background service

The authors of this article used PocketSphinx to create an in-app speech recognition system. Given that this lightweight library can also run in the background of a mobile device continuously, the Speech Radar app may require PocketSphinx. On top of this, the article mentions it works without Internet connection, which can be useful if users' phone were to disconnect during the process.

Chapter 3 Specification

Chapter 3 states a detailed explanation of all the requirements needed to create the app. Examples include how it should function, how it could handle failures, what measure should be taken to prepare for potential disaster, how user can interact with the interface and many more. This would be a great section to look back at after implementation and testing has finished, to see whether all requirements have been met.

3.1 Functional requirements

Describes the behaviour of the system. Java with Android development and XML will be used to achieve this.

- The app should load a simple splash screen at the beginning
- The system must send an email verification when user creates account to prove that provided email address exists
- Verified user should be recognised by the system and grant them access to their account
- The app should ask for user permission to record audio
- When mic button is pressed and permission is granted, audio should record for 3 seconds to receive user input of the code word
- If user is happy with the code word, background service of continuous speech recognition should activate when button is pressed
- Background service should deactivate when user utters the code word or terminates the app

3.2 Non-functional requirements

Explains requirements of how the system should perform or is unrelated to the functionality of the app.

- The app should be compatible on all Android devices running on API level 16 (Jelly Bean) or more
- APK file should be 70MB or less
- A backup of all user accounts should be kept in separate database in case of a disaster
- The continuous background service should not deeply affect performance of the phone (e.g. drain too much battery life, slow down the phone etc.)
- All android devices must be touch-screen to use the interface
- Interface should be intuitive (e.g. add few objects and spread them across the screen so the layout is clean)
- System should be able to handle large number of users all online at the same time

3.3 Java requirements

Discusses what tasks should be completed with Java.

- Insert account details to Firebases Real-time Database when user creates an account
- first name field should be pulled from correct record in database and displayed when user logs in to their account
- For security purposes, password should be hashed when added to database
- Utilise Google's free SMTP server to send email verification and password reset emails upon request with Firebase Authentication
- Authenticate email address and password in login screen
- Load the TensorFlow model and get it to make predictions in the speech recognition screen

3.4 Database requirements

Backend database must meet these requirements.

- Offer quick data retrieval
- Data should be correct, consistent and update when needed to

3.5 TensorFlow requirements

Following targets in TensorFlow should be met.

- Should have a similar architecture to the article revised in chapter 2 section 2.2
- Testing accuracy should be over 90%
- Use Speech Commands Dataset for training (explained in chapter 2 section 2.1)
- 18 words should be used for recognition. The words are: yes, zero, stop, learn, happy, house, Sheila, six, three, tree, visual, Marvin, up, down, left, right, backward and forward

Chapter 4 Design

This chapter discusses the design of the Android application through diagrams, flowcharts, pseudocode and wireframes. Wireframes will be used to describe the appearance. Whereas the rest will be used to describe the structure or behaviour of the app. The chapter will only display designs in relation to speech recognition so the main area of the app can be revised and established in this report. Designs for login screen, create account and forget password can be seen in Appendix C.

4.1 Visual flowchart

4.1.1 TensorFlow model

Before designing the application, it would be wise to plan the model architecture in TensorFlow. Figure 7 shows a visual flowchart of how this would look like. The CNN model is very similar to the one used in the article analysed in chapter 2 section 2.2.1, where two convolutional layers are used along with one fully connected layer at the end with SoftMax activation function. Every convolution layer has a ReLU activation function which will follow with max pooling and dropout layer. Batch normalization could also be used, as we saw in chapter 2 section 2.2.3 a sudden increase in validation accuracy and a steady drop in validation loss when used. But, in this case two convolution layers are used, which will give us similar results.

Features such as time and frequency are extracted from the spectrogram to help classify the audio signal. 18 different words will be used for classification. Thus, the reason why 18 outputs are shown in the flowchart, making this a multi-class classification problem.

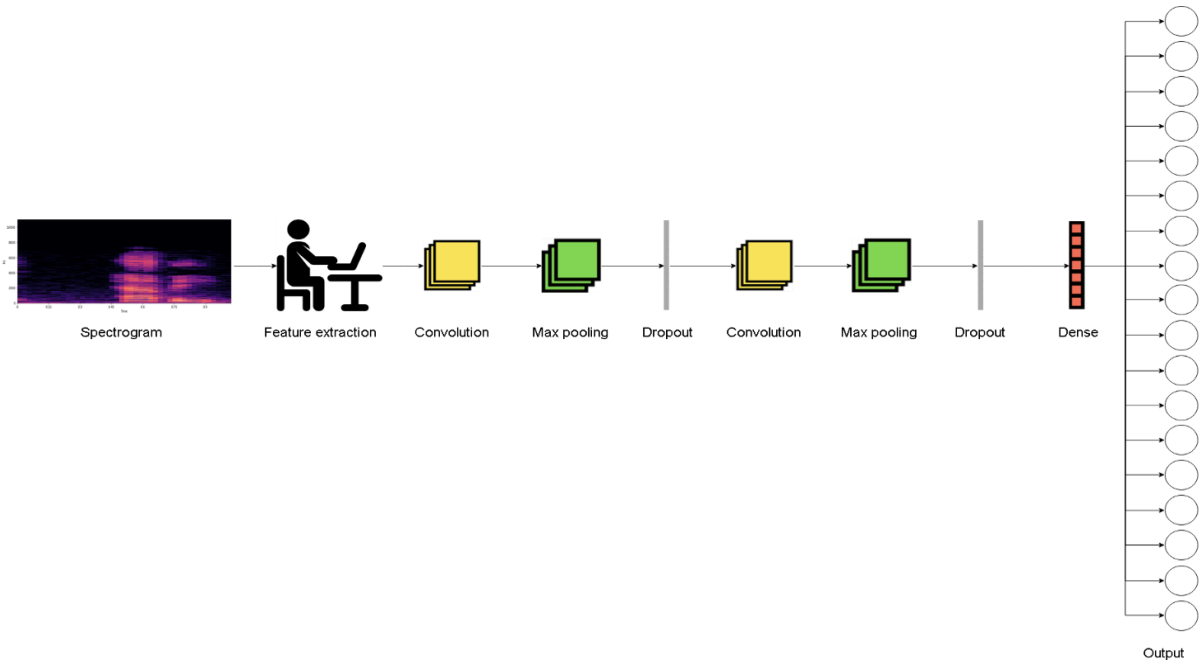


Figure 7: CNN architecture design for speech recognition

4.1.2 Loading model into Android Studio

The flowchart in figure 8 explains with text on how to load the model in Android Studio and get it to make predictions. This visual guide will help generalise the tasks and provide clarity when things start to get tough in the implementation phase.

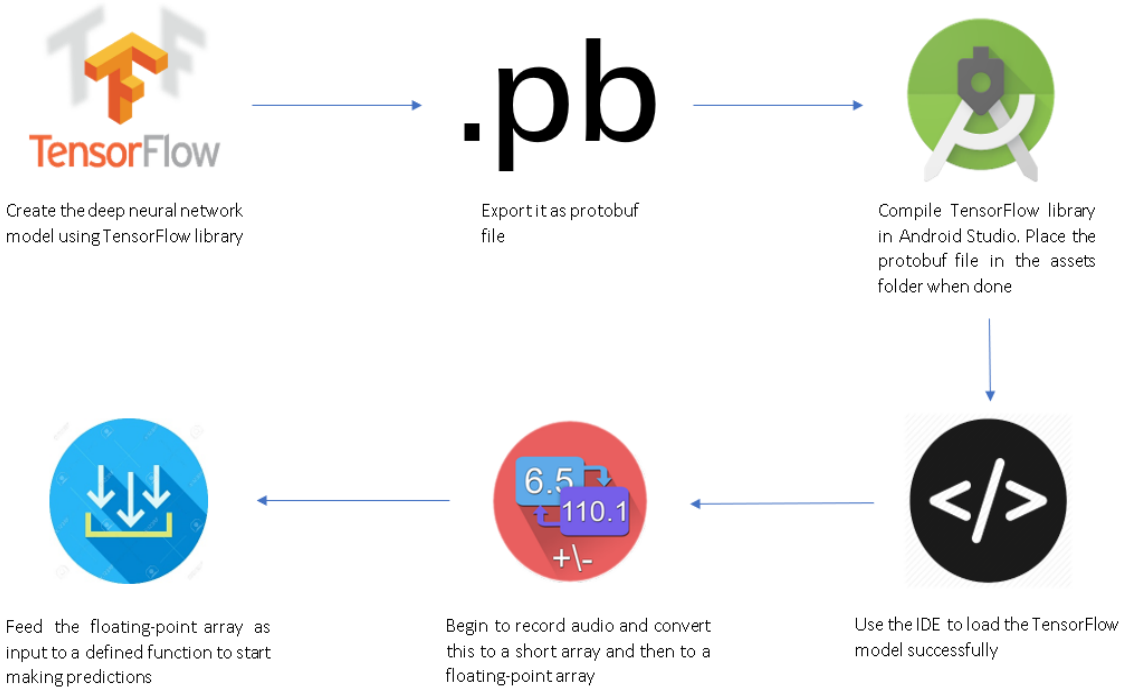


Figure 8: Visual flowchart showing how TensorFlow model can be loaded in Android Studio

4.2 UML diagram

4.2.1 Class diagram

UML helps to provide a way of visualising the design of a system. Class diagrams were used first in this project to envision the classes in Java. It maps out the structure of a system by representing its classes, function and relation existing between objects. This section will go over the speech recognition and background service screen. All class diagrams for the other screens can be seen in Appendix C.

4.2.1.1 Speech recognition

The class diagram in figure 9 shows all classes needed by the speechrecognition class. First, we see a one-to-many relationship between FirebaseDatabase and DatabaseReference, as one database can have many references (e.g. firstname : "JohnZ", lastname: "Watch" etc.) Both classes are needed by speechrecognition mainly to save the code word in the database.

A one-to-one relationship between speechrecognition and loginscreen represents the extraction of email address provided by user in the loginscreen class, so the system can access the correct parent in the database and display the correct account details. The database structure is explained clearly in section 4.4.

Another one-to-one relationship seen between speechrecognition and TensorflowInferenceInterface is needed to start or stop the model from making predictions. Finally, a one-to-one with the Recognise class is required to average the audio signals and return information about a particular label when there is enough proof to consider that a word has been found.

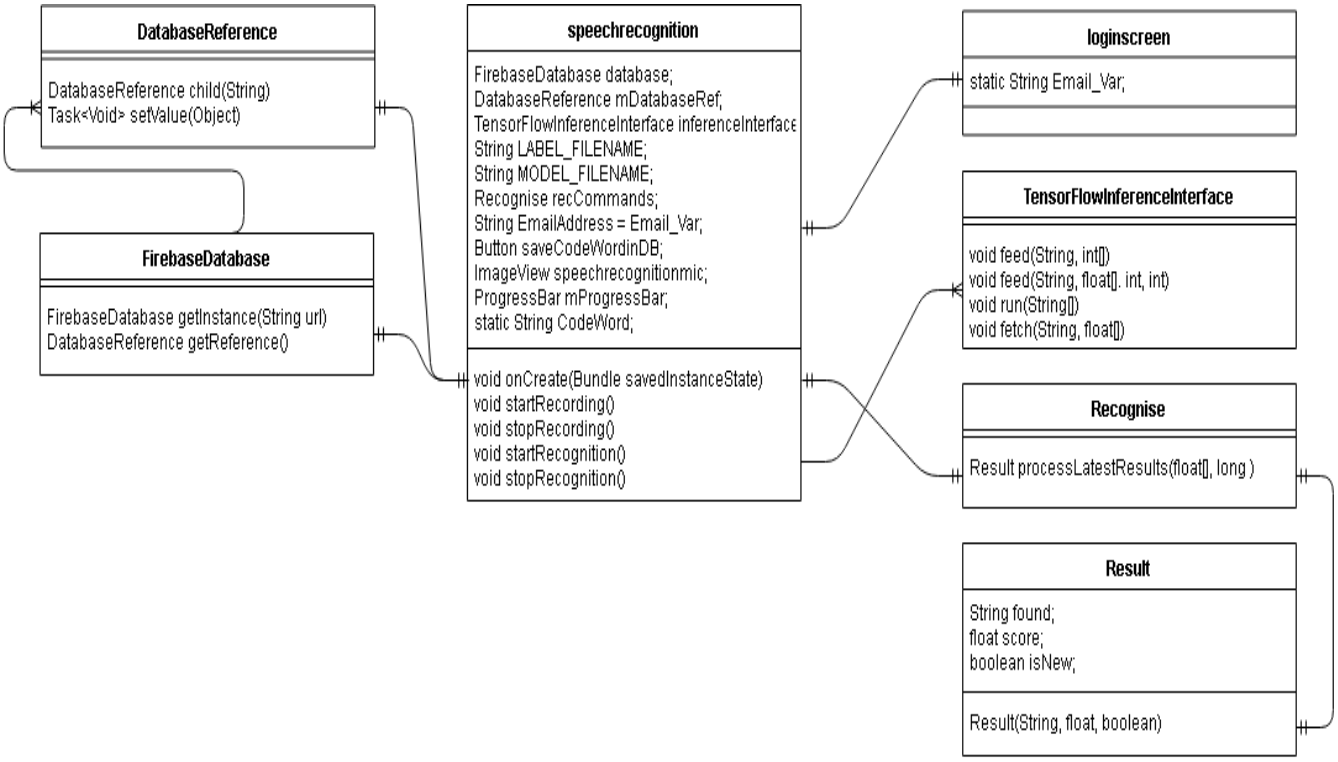


Figure 9: Class diagram for speechrecognition class

4.2.1.2 Background service

The class diagram in figure 10 shows a capable structure of the background service. BackgroundService is an activity class that encompasses the visual aspect of the screen and in running the service. A one-to-many relationship with continuousService is essential, as most of the implementation for the never-ending service is in the continuousService class. All the BackgroundService class needs to do is start it.

The one-to-one relationship between continuousService and speechrecognition is necessary for the audio recognition system running in the background to know which code word to spot. The code word is registered in the speechrecognition class analysed in previous class diagram. Once it detects the code word, it'll play the ringtone to max volume.

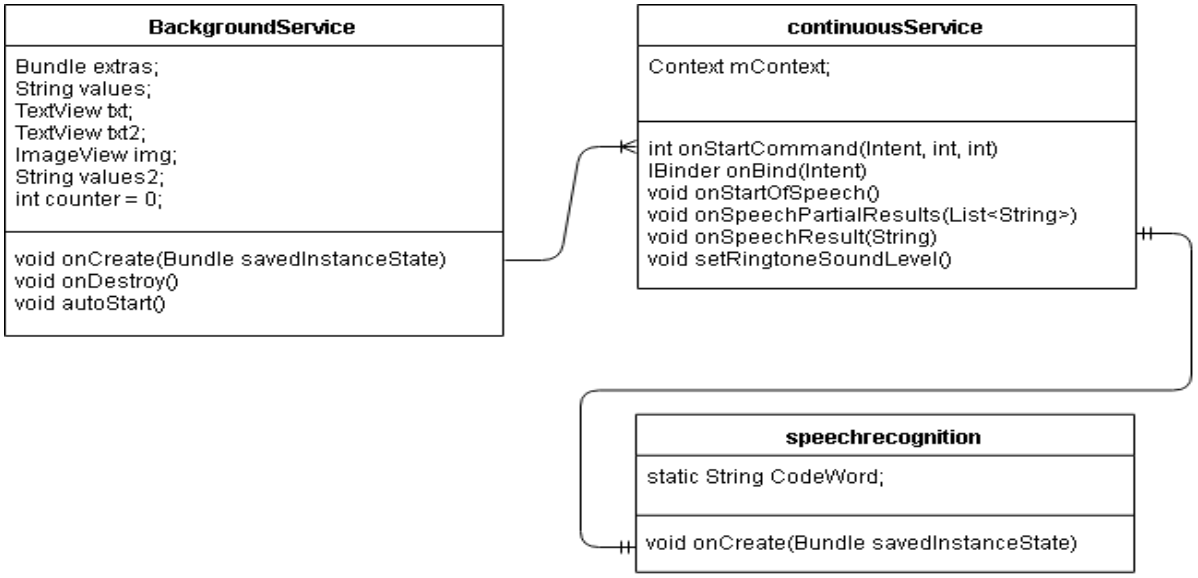


Figure 10: Class diagram for BackgroundService class

4.2.2 Activity diagram

Activity diagram is important in UML to describe the behaviour of a system. It is essentially a flowchart showing the flow from an activity to another. The activities can be thought of as an operation carried out by the system. This can help visualise the sequence of actions that need to be taken in Speech Radar. It can also help generalise the app as a whole. Take for instance the diagram in figure 11, which showcases the entire activity of the app from beginning to end. Indeed, it has helped to think of possible failures the user will face and how the app can respond to the failure. For example, if user has forgotten password, then they are sent an email to reset it and log in again with different password. This prevents from having to create another account.

Mid-way through the diagram, we see the user enter the speech recognition screen after a successful log in. Here, the user clicks the mic button to record audio of them saying a particular code word. The word is saved in the database and user has then the option to terminate the app. If not, then the continuous speech recognition background service will run forever until it detects the code word.

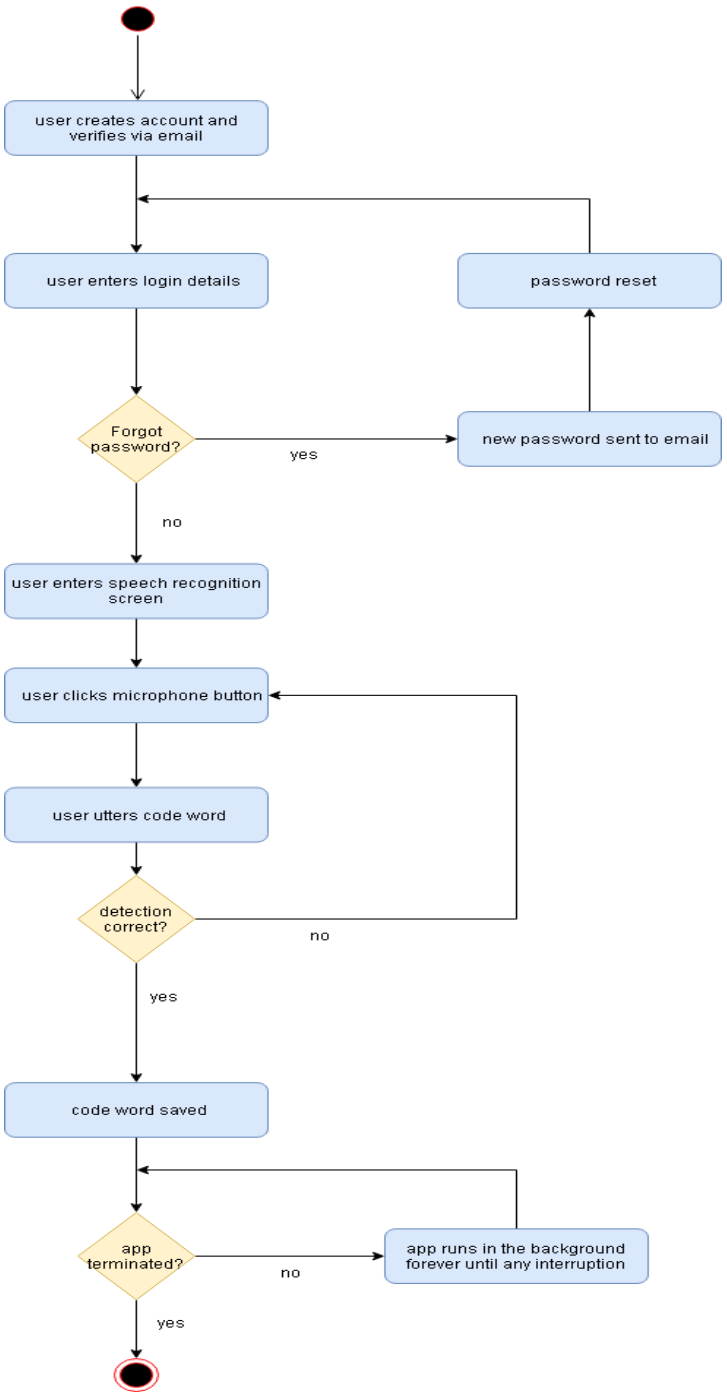


Figure 11: Activity diagram

4.2.3 Use-case diagram

A use case diagram is another important UML that describes the behaviour of a system. It models the process of a system using imaginary actors and use cases that consist of actions that the system must execute. This section will go over the speech recognition and background service screen. All the use case diagrams for the other screens can be seen in Appendix C.

4.2.3.1 Speech recognition

The diagram in figure 12 further illustrates features in the speech recognition screen that users can interact with. On the other side, it shows what the use cases depend on in order to carry out the operation. For example, users can interact with the mic button, which needs the TensorflowInferenceInterface class to make predictions from the recording audio. User can also interact with the activate button, which depends on Firebase Database to add the code word. It depends on Intent class to open the background service activity. Similarly, the back button requires the Intent class to go back to the login screen activity.

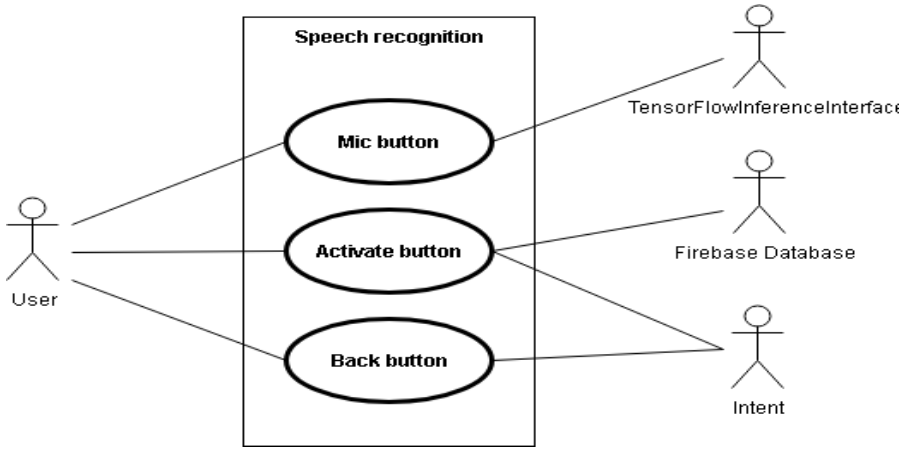


Figure 12: Use case diagram for speechrecognition class

4.2.3.2 Background service

The diagram in figure 13 illustrates how the background service is activated. It shows that the user is not involved in activating it, but rather the system. Once the user enters the background service screen, the service will automatically start with a pop-up suggesting this to the user. This removes the need for an extra button. The system can also stop the service if the user decides to terminate the app or the user utters the code word detected by the background speech recognizer.

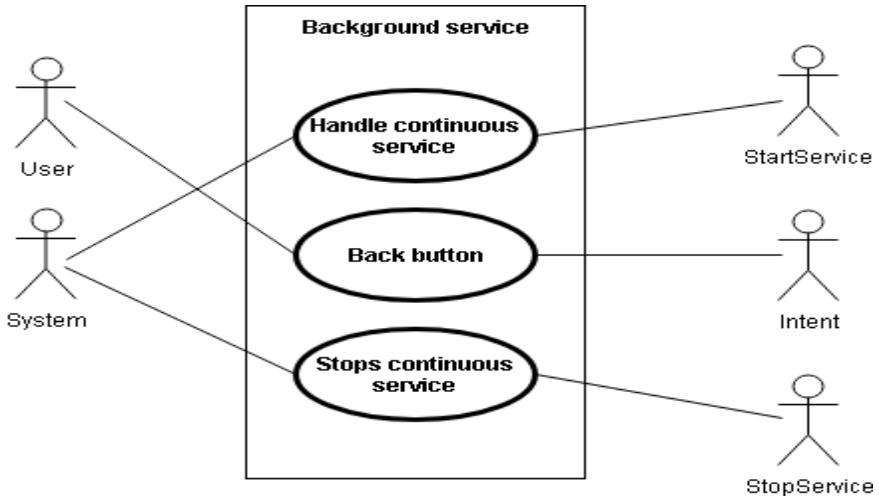


Figure 13: Use case diagram for BackgroundService class

4.3 Tree diagram

Since Firebase is a NoSQL key-value database, a tree-like diagram seen in figure 14 is used to describe the structure. A Database Schema would've been a better choice, but this would be appropriate for designing RDBMS. Underneath the root node, we see two parent nodes. Each parent represents a user account, which will be entitled as the user's email. The child nodes are simply key-value that's added from the app (e.g. firstname: "Alex", lastname: "Smith", password: "3dfk9p31s", codeword: "Marvin" etc.) The password and confirm password nodes contain hash value, to protect user account if database is exposed to attackers.

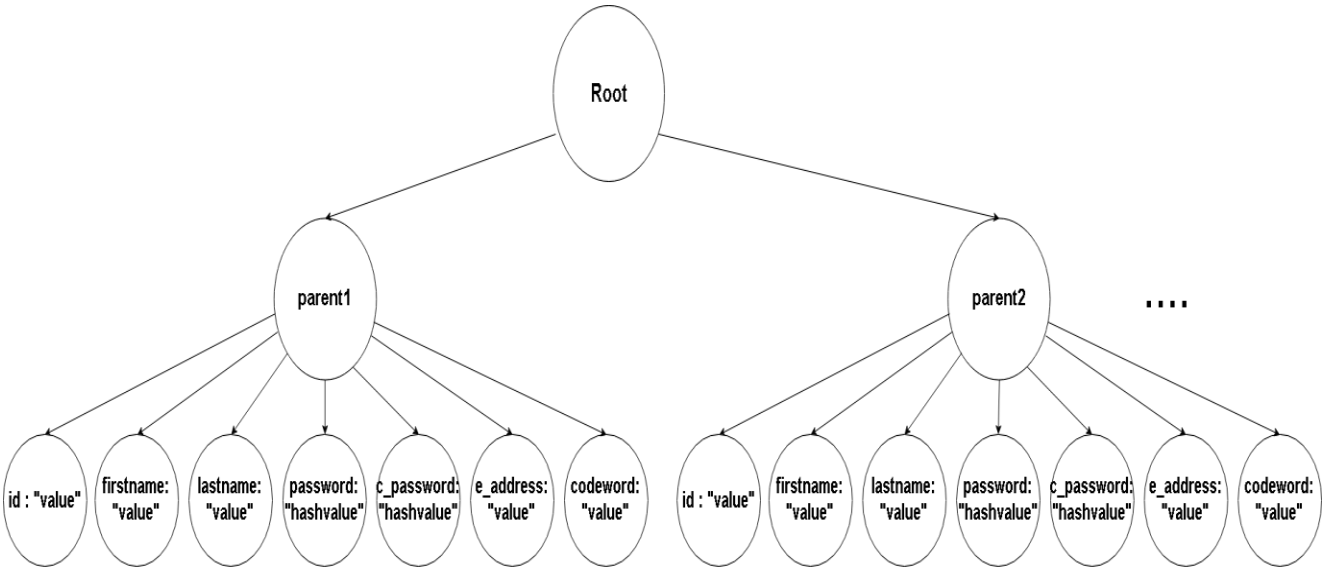


Figure 14: Tree diagram of Firebase Database structure

4.4 Wireframes

Wireframes are a great way of representing the appearance of an app. Their purpose is to order various objects, so one can fulfil a particular objective. This section goes over the wireframes for speech recognition and background service screen. All wireframes for the other screens can be seen in Appendix C.

4.4.1 Speech recognition

The wireframes in figure 15 shows a first glimpse of how the speech recognition screen would look like. We see the screen before the mic button is pressed, where the code word and activate button are visible. We then see the screen after the mic button is pressed, where we see a white rectangle appear showing the results from what the speech recognition system picked up. The layout is like Google Assistant. The two figures show a simple layout to make it intuitive for users.

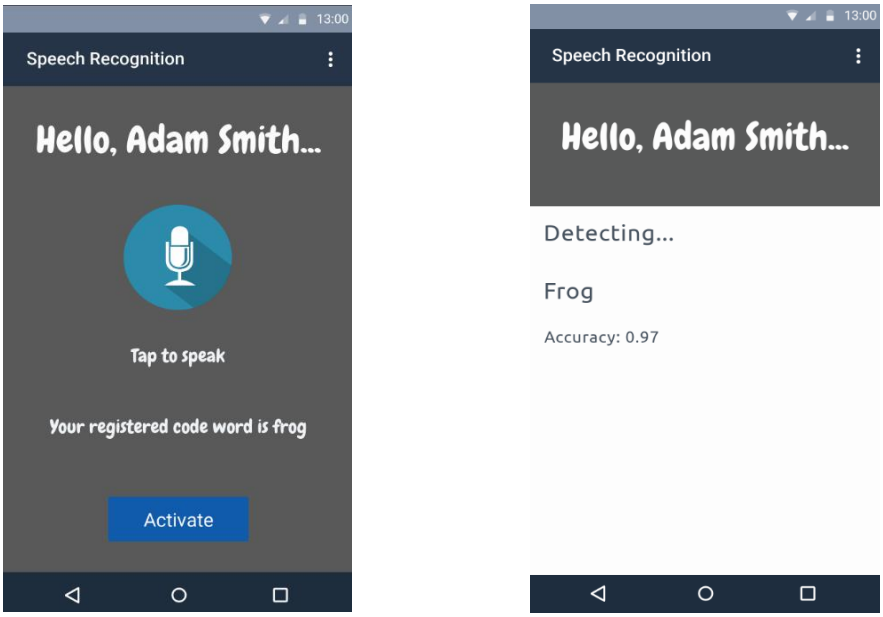


Figure 15: speech recognition wireframes showing before and after of when mic button is pressed

4.4.2 Background service

The wireframe in figure 16 shows the background service screen. Here, there is nothing for the user to interact with. It is just to inform user that the continuous speech recognition service has automatically activated in the background. The screenshot within this screen will show what the user should not do (i.e. to not terminate the app) if they want the service to run.

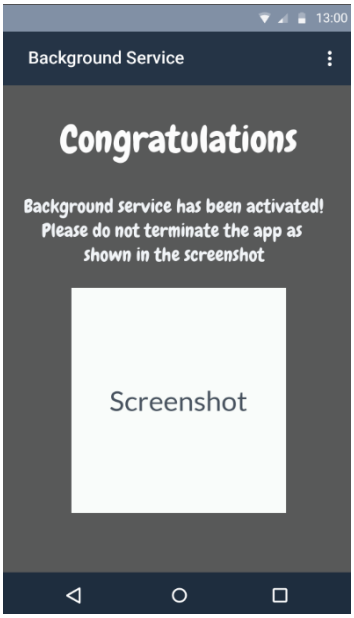


Figure 16: background service wireframe

4.5 Pseudocode

For the final part of design, pseudocode is written for every screen. It uses the structure of a programming language but is written in a way that can be understood by humans rather than a machine. This section will show the pseudocode written for speech recognition and background service screen. All pseudocode for the other screens can be seen in Appendix C.

4.5.1 Speech recognition

Figure 17 shows pseudocode for the speech recognition activity. It is written under the onCreate method, which is executed as soon as the activity starts. It states if user clicks mic button, then the system should read the protobuf file, read the labels file in .txt format and start the audio recording. After, get the model to make predictions of the recording audio. Otherwise, stop the recording and stop the model from making predictions. For the next if statement, it states that if activate button is pressed, then the code word should be added to the database and the background service activity should open. The final if statement states if the back button is pressed, then the login screen activity should open.

```
onCreate()
  IF isClicked(mic_btn) == true THEN
    read → .pb file
    read → labels file
    startRecording()
    startRecognition()
  ELSE
    stopRecording()
    stopRecognition()

  IF isClicked(activate_btn) == true THEN
    DB.child(val).setValue(codeWord)
    startService(myBackgroundService)

  IF isClicked(back_btn) == true THEN
    startActivity(loginscreen)
```

Figure 17: basic pseudocode for speechrecognition class

4.5.2 Background service

Figure 18 shows pseudocode for the background service activity. It states if user clicks back button, then it should go to login screen activity. Under this if statement, a function entitled 'checkServiceRunning()' is used in onCreate. This function will simply check on the continuous speech recognition service to see whether it has stopped unexpectedly or whether the app has been terminated. If it has been stopped, then the function will try to restart it. If the app has been terminated, then the service will stop.

```
onCreate()
  IF isClicked(back_btn) == true THEN
    startActivity(loginscreen)
  checkServiceRunning()


checkServiceRunning()
  IF isStopped(myBackgroundService) == true THEN
    restartService()


  IF isTerminated(App) == true THEN
    stopService(myBackgroundService)
```

Figure 18: basic pseudocode for BackgroundService class

Chapter 5 Implementation

This chapter mentions how Specification and Design section were used to implement the app. It will describe thoroughly the implementation in TensorFlow, Java, XML and Firebase Database. Screenshots will be used to aid in the explanation.

5.1 TensorFlow

TensorFlow's implementation of a simple audio recognition for Android, located in their GitHub repository follows the same article revised in chapter 2 section 2.2 for the keyword spotting task. It uses the Speech Commands dataset containing 35 words. The solution used only 10 words for recognition. This implementation will be extended for the project to recognise 18 words, as models created could not make predictions properly when exported into Android Studio. Also, TensorFlow's implementation is the only solution that has worked so far. Website states that the code is *open source and available as part of the TensorFlow repository on github* [7], which provided further encouragement. The main aim of the project is to implement the app, so this cannot really be seen as a setback. The repository containing the code can be seen in reference [8].

5.1.1 Test accuracy with confusion matrix

At first, TensorFlow's implementation outputted low test accuracies around 86% to 89%. To boost the accuracy, more data was needed in the dataset. Thus, hundreds of additional one second audio clips were added for each word. This helped boost the accuracy to 90.8%, which was enough to meet the specification for this project. The screenshot in figure 19 shows proof of the accuracy after taking 21000 training steps. It also shows a confusion matrix, which helped visualise the performance. Each row and column can be seen as a set of words that are going to be used. Figure 20 visualises this better and shows the words that are going to be used. The basic principle is each word in row will be checked by each word in column by the model to see whether it has predicted correctly, which in our case has. Although 18 words are used, the confusion matrix is 20 x 20. This is because 'Silence' and 'Unknown' labels are included.

```
INFO:tensorflow:Confusion Matrix:
 [[537   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]
 [  3 396   5   5  10  10   5   1   2  10   6   1   0   1  23  22   6  17
    1  13]
 [  0   6 396   0   1   3   0   0   1   1   0   0   0   0   1   0   9   0
    1   0]
 [  1   7   1 392   1   0   0   0   2   2   1   0   9   0   0   0   1   1
    0   0]
 [  2   2   0   0 391   0   0   0   0   1   0   0   0   0  13   1   0   1
    0   0]
 [  2  12   1   6   0 122   0   0   1   1   0   1   0   2   0   6   6   1
    0   0]
 [  1   6   0   0   0   0 183   2   0   4   1   0   0   1   3   2   0   0
    0   0]
 [  2   5   1   1   1   0   1 170   0   2   0   0   0   0   6   1   0   1
    0   0]
 [  1   4   0   4   2   0   0   0 197   2   1   0   0   0   0   1   0   0
    0   0]
 [  1   6   0   4   0   0   0   0   1 380   1   0   0   0   0   0   0   1
    0   0]
 [  0  15   0   1   1   0   2   0   1   3 352  25   0   0   0   0   1   4
    0   0]
 [  3   5   0   1   0   0   1   0   1   1  25 156   0   0   0   0   0   0
    0   0]
 [  1   0   0  15   0   0   0   0   2   1   1   0 145   0   0   0   0   0
    0   0]
 [  1   2   0   2   0   1   1   0   0   0   0   0   1 186   0   0   1   1
    0   0]
 [  3   9   0   0   9   0   3   3   0   0   0   0   0   0 397   5   1   1
    0   0]
 [  1  19   3   0   0   4   0   1   0   0   0   0   0   1   1 381   2   0
    2   0]
 [  0   2   9   2   0   1   0   0   0   4   0   1   0   0   2   0 389   3
    1   0]
 [  1  11   1   1   0   0   0   0   0   2   4   0   0   2   0   1   4 371
    0   0]
 [  0   0   0   1   0   0   0   0   0   1   0   0   2   1   3   0   3   1
  161   0]
 [  1  17   0   0   0   0   1   1   0   0   0   0   0   0   0   1   0   0
    1 142]]
INFO:tensorflow:Final test accuracy = 90.8% (N=6437)
```

Figure 19: confusion matrix

	Silence	Unknown	Yes	Zero	Stop	Learn	Happy	House	Sheila	Six	Three	Tree	Visual	Marvin	Up	Down	Left	Right	Backward	Forward
Silence																				
Unknown																				
Yes																				
Zero																				
Stop																				
Learn																				
Happy																				
House																				
Sheila																				
Six																				
Three																				
Tree																				
Visual																				
Marvin																				
Up																				
Down																				
Left																				
Right																				
Backward																				
Forward																				

Figure 20: table to help understand the confusion matrix in figure 19

5.1.2 Scalar diagram

Figure 21 shows the validation loss in the form of a scalar diagram. It shows two different lines. The line ending at 60000 training steps is the first attempt that was done before adding hundreds of audio clips to the dataset for each word (as mentioned in the previous section). The line below it is the second attempt, which ends up with a similar loss with only 21000 training steps. The loss goes as low as 0.35. A low validation loss is important, as it tells us that the model will do well with unseen data.

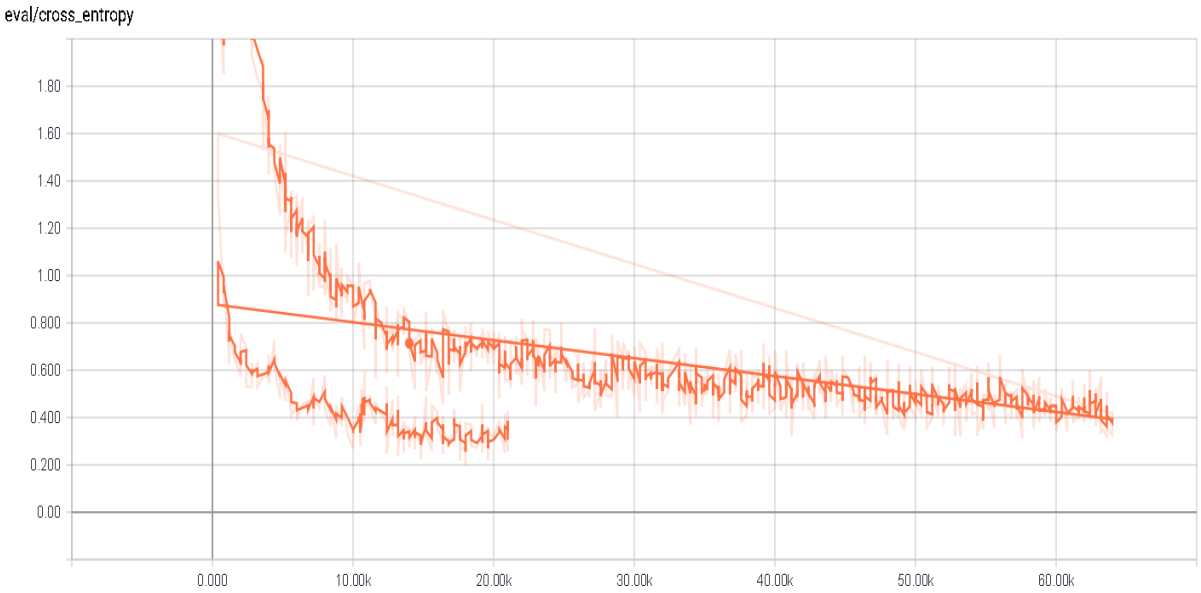


Figure 21: scalar diagram

5.2 Java

This section goes over the main implementation for each class in Java. The aim of this is to explain the code and make sense of it at a human-level. The code was written with Android Studio IDE. Full code for Java can be seen in Appendix G 10.36

5.2.1 Loginscreen

The main purpose of the loginscreen class is to enable users to login via email and password. The check() method in code 1 runs when user clicks the login button. It shows how FirebaseAuth class is used to check whether the account exists, is correctly inputted and has been verified via email verification. If true, then the speechrecognition activity will start, where user can register the code word and activate the background service. Otherwise, a pop-up appears in the interface that displays the error details.

```
public void check() {
loginbutton.setOnClickListener(new View.OnClickListener() {
//onclick listener for the login button
@Override
public void onClick(View view) { //onclick method
try {
firebaseAuth.signInWithEmailAndPassword(userEmail.getText().toString(),userPass.getText().toString()).addOnCo
mpleteListener(new OnCompleteListener<AuthResult>() {
//listener for FireBase Authentication, that authenticates inputted email address and password
@Override
public void onComplete(@NonNull Task<AuthResult> task) {
if (task.isSuccessful()) { //if task has completed
if (firebaseAuth.getCurrentUser().isEmailVerified()) { //and user has verified themselves through email
verification
if (SystemClock.elapsedRealtime() - mLastClickTime < 7000) {   //this if statement and few lines of code
below prevents the speechrecognition activity from opening multiple times if user accidentally clicks the
login button multiple times
return;
}
mLastClickTime = SystemClock.elapsedRealtime();
String email = userEmail.getText().toString(); //store inputted email address
email = email.replace(".", ""); //this removes dot from the email address
email = email.replace(" ", ""); //this remove any spaces in email  addresses
Intent i = new Intent(loginscreen.this, speechrecognition.class);
i.putExtra("email_var", email); //stores the email address without full stop or space so that it can be used
in the next activity (speechrecognition.) this variable is used to access the record stored in the database,
so it can extract the firstname field and display it on the screen

startActivity(i); // the activity switches to speechrecognition
}
else {
Toast.makeText(loginscreen.this, "Please verify your email address", Toast.LENGTH_LONG).show(); //error
message pops-up to suggest email address exists, but is not verified }
}
else {
Toast.makeText(loginscreen.this, task.getException().getMessage(), Toast.LENGTH_LONG).show(); //any other
exceptions caught by task, such as if  email address format is wrong etc.
}
}
});
}
catch (Exception e ) {
Toast.makeText(loginscreen.this, e.toString(), Toast.LENGTH_LONG).show(); //if any exceptions are caught that
are not caught by task, then display it on-screen e.g. nullpointerexception
}}
});
}
```

Code 1: check() method implemented in loginscreen class

5.2.2 Createaccount

The createaccount class is needed for when user creates an account. The following inputs fields require user input: first name, last name, password, confirm password and email address. The AddData() method seen in code 2 runs when user clicks the create account button. The first thing it defines is the validation of each input field. If the input goes against the rules (e.g. if password contains less than 5 digits and is less than 8 characters), then it will stop user from creating an account, display an error message and colour the input fields red to indicate where the errors are present.

Otherwise, the program will store the account details in the database through help from DatabaseReference and FirebaseDatabase class to create a child and insert into it. The password field and confirm password field are hashed using the Hash(String message) method seen in code 3, to protect passwords against potential attacks on the database. Then, an email verification is sent to the inputted email through FirebaseAuth class, to authenticate user.

```
public  void AddData() {
createAccountbtn.setOnClickListener(  //onclick listener for the createaccount button
new View.OnClickListener() {
@Override
public void onClick(View v) { //onclick method
final String passwordstring = password.getText().toString(); //stores inputted password
final String confirmpasswordstring = conf_password.getText().toString(); //stores inputted password
final String firstname = fname.getText().toString(); //stores inputted first name
final String lastname = lname.getText().toString(); //stores inputted last name
final String emailaddress = email_address.getText().toString(); //stores inputted emailaddress
int size1 = firstname.length(); //stores length of characters inputted for first name
int size2 = lastname.length(); //stores length of characters inputted for last name
if(size1 >1) { //if length of characters for inputted first name is 2 or more
fname.getBackground().setColorFilter(Color.WHITE, PorterDuff.Mode.SRC_ATOP); //then color the input field
white to indicate user has correctly completed this field
}
if(size2>1) {
lname.getBackground().setColorFilter(Color.WHITE, PorterDuff.Mode.SRC_ATOP);
}
if(passwordRules(passwordstring) == true) { //checks if password is more than 8 characters and contains at
least 4 digits
password.getBackground().setColorFilter(Color.WHITE, PorterDuff.Mode.SRC_ATOP); //if true,then input field is
colored white to indicate user has correctly completed this field
}
if(confirmpasswordstring.equals(passwordstring)) { //if confirm password is equal to password
conf_password.getBackground().setColorFilter(Color.WHITE, PorterDuff.Mode.SRC_ATOP); //color the confirm
password input field to white to indicate user has correctly completed this field
}
if(isValidEmailAddress(emailaddress) == true) { //checks if email address is written in correct format
email_address.getBackground().setColorFilter(Color.WHITE, PorterDuff.Mode.SRC_ATOP); //then color the input
field white to indicate user has correctly completed this field
}
if(size1 > 1 && size2 > 1 && passwordRules(passwordstring) == true &&
confirmpasswordstring.equals(passwordstring) && !email_address.getText().toString().equals("") &&
isValidEmailAddress(emailaddress) == true) { //if user input meets validation requirement

firebaseAuth.createUserWithEmailAndPassword(email_address .getText().toString().trim(),
password.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthResult>() { //listener for
Firebase Authentication, that inserts user email and password
@Override
public void onComplete(@NonNull Task<AuthResult> task) {
if (task.isSuccessful()) { //if task has completed
firebaseAuth.getCurrentUser().sendEmailVerification().addOnCompleteListener(new OnCompleteListener<Void>() {
//sends email verification
@Override
public void onComplete(@NonNull Task<Void> task) {
boolean isInserted = false;

if(task.isSuccessful()){ //if task has completed
Toast.makeText(createaccount.this, "Registered successfully. Please check your email for verification",
Toast.LENGTH_LONG).show(); // display pop-up to tell user to check email for verification
String id = myRef.push().getKey(); //gets a unique key for when account details are added to the database
String hashpassword = Hash(passwordstring); //Hashes password
String hashconfpassword = Hash(confirmpasswordstring); //hashes confirm password
String emailaddress2 = emailaddress.replace(".", "");
emailaddress2 = emailaddress2.replace(" ","");
Account account = new Account(id,firstname,lastname,hashpassword,hashconfpassword,emailaddress); //Account
class is used to store all details of user
myRef.child(emailaddress2).setValue(account); //inserts user details to the Firebase database
}
else {
Toast.makeText(createaccount.this,task.getException().getMessage(),Toast.LENGTH_LONG).show();
}}});
}
else {
Toast.makeText(createaccount.this, task.getException().getMessage(),Toast.LENGTH_LONG).show();
}}});
}
else { //checks each input field to see if user has incorrectly inputted data. If so, then the input fields
inputted incorrectly will be colored red
if(email_address.getText().toString().equals("")) {
email_address.getBackground().setColorFilter(Color.RED, PorterDuff.Mode.SRC_ATOP);
}
if(size1 <=1) {
fname.getBackground().setColorFilter(Color.RED, PorterDuff.Mode.SRC_ATOP);
}
if(size2<=1) {
lname.getBackground().setColorFilter(Color.RED, PorterDuff.Mode.SRC_ATOP);
}
if(passwordRules(passwordstring) == false) {
password.getBackground().setColorFilter(Color.RED, PorterDuff.Mode.SRC_ATOP);
}
if(!confirmpasswordstring.equals(passwordstring)) {
conf_password.getBackground().setColorFilter(Color.RED, PorterDuff.Mode.SRC_ATOP);
}
if(isValidEmailAddress(emailaddress) == false) {
email_address.getBackground().setColorFilter(Color.RED, PorterDuff.Mode.SRC_ATOP);
}
Toast.makeText(createaccount.this, "Error in above input fields. Please correct them. Make sure password is
min 8 characters long containing 5 digits",Toast.LENGTH_LONG).show();
}}});
}
```

Code 2: AddData() method implemented in createaccount class

```
public boolean isValidEmailAddress(String emailaddress) { //method will check if inputted email address is in
correct format
boolean result = true;
try {
InternetAddress emailAddress = new InternetAddress(emailaddress);
emailAddress.validate();
} catch (AddressException ex) {
result = false;}
catch (Exception e) {
Toast.makeText(createaccount.this, e.toString(),Toast.LENGTH_LONG).show();
}
return result;     }

private final static String salt="DGE$5SGr@3VsHYUMas2323E4d57vfBfFSTRU@!DSH(*%FDSdfg13sgfsg";

public static String Hash(String message) {
String md5 = "";
if(null == message)
return null;
message = message+salt;//adds salt to the string before its hashed.
try {
MessageDigest digest = MessageDigest.getInstance("MD5");//MessageDigest object for MD5
digest.update(message.getBytes(), 0, message.length());
md5 = new BigInteger(1, digest.digest()).toString(16);//Converts message digest value to base 16
} catch (NoSuchAlgorithmException e) {
e.printStackTrace();}
return md5;    }

public static boolean passwordRules(String password) {
// stores a boolean variable to hold the result of the method
boolean c = true;
// stores an int variable to hold the count of each digit
int number = 0;
if (password.length() < 8) {
// if password is less than 8 characters, return false */
return false;  }
// store a char variable to hold each element of the String
char elem;
// Check if the password has 2 or more digits
for(int i = 0; i < password.length(); i++ ){
// Checks each char in the String
elem = password.charAt( i );
// Check if its a digit or not
if( Character.isDigit(elem) ){
// It is a digit, so increment digit
 number++;
} }
// Now check for the count of digits in the password
if( number < 5 ){
// if their are less than 5 digits in password, return false
return false;
}
//regex checks for upper and lower case letters and digits
if( !password.matches("[a-zA-Z0-9]+") ){
return false;
}
//the password is valid, if it reaches this point
return c;
}
```

Code 3: Shows methods used for validation in createaccount class

5.2.3 forgotPassword

The forgotPassword class enables users to reset their password, if they have forgotten it. The send_password_reset_email() method in code 4 runs when reset password button is pressed. It checks whether the inputted email address exists, and if it does then a password reset email is sent. The email will allow the user to create a new password. This is again done via the FirebaseAuth class, which has proven to shorten the lines of code for all classes it has been used for.

```
public void send_password_reset_email() {
resetpass.setOnClickListener(new View.OnClickListener() { //onclick listener for the reset password button
@Override
public void onClick(View view) { //onclick method
try {
FirebaseAuth.getInstance().sendPasswordResetEmail(email.getText().toString().trim())
.addOnCompleteListener(new OnCompleteListener<Void>() { //listener to send password reset email
@Override
public void onComplete(@NonNull Task<Void> task) {
if (task.isSuccessful()) { //if task is complete
Toast.makeText(forgotPassword.this, "Reset password sent to " + email.getText().toString(),
Toast.LENGTH_LONG).show(); // then, display a pop-up to inform user that reset password email has been sent
} else {
Toast.makeText(forgotPassword.this, "Email does not exist! please try another email",
Toast.LENGTH_LONG).show(); //else, display a pop-up to inform user that inputted email address does not exist
}}});
}
catch(Exception e){
Toast.makeText(forgotPassword.this, e.toString(),Toast.LENGTH_LONG).show(); //if any exception is caught,
then show this to user in a pop-up
}}});
}
```

Code 4: send_password_reset_email() method implemented in forgotPassword class

5.2.4 speechrecognition

The speechrecognition class is where the main implementation of Speech Radar lies. It contains a lot of code, mostly obtained and further extended from TensorFlow's GitHub repository. It was mentioned previously that the same repository was used to acquire and use a network architecture in TensorFlow that works in Android. Thus, screenshots of code that were sourced/taken from the repository will not be added in this section but will be explained.

5.2.4.1 Mic button

The onCreate(Bundle SavedInstanceState) method is used to initialise the activity. It contains code that runs when the speech recognition mic button is pressed. This button will begin using TensorFlowInferenceInterface class, which loads the model located in assets folder. Then, the labels file is loaded and added into an ArrayList. Labels include: _silence_, _unknown_, yes, 0, stop, learn, happy, house, Sheila, 6, 3, tree, visual, Marvin, up, down, left, right, backward and forward. After, RecognizeCommands class is instantiated to smooth recognition result and increase accuracy. The full code for the RecognizeCommands class can be seen in GitHub, accessible via reference [9].

Finally, audio starts recording using the startRecording() method and display predictions based on the uttered speech in the startRecognition() method. Once the 4 second timer is up, audio will stop recording through stopRecording() method and predictions will stop being made through stopRecognition() method.

5.2.4.2 startRecognition() & stopRecognition()

The startRecognition() method runs a separate thread to make predictions from the imported TensorFlow model. It first stores the recording audio as a short array. Then, it copies the values into a float array and divides it by 32767, since the encoding of audio is set to 16-bit PCM. The TensorFlowInferenceInterface class is then used to feed the float array to the model. The class is used again to fetch the output i.e. the prediction. Finally, the output is stored as a String and displayed in a

TextView within the interface. The stopRecognition() method halts predictions from being made in a separate thread. The full code can be seen in GitHub, accessible via reference [10].

5.2.4.3 startRecording() & stopRecording()

The startRecording() method runs in a thread that's prioritised to execute first. The recorder is defined using Androids AudioRecord class, which requires input for the sampling rate, encoding and buffer size. The sampling rate used is 16000 and encoding is 16-bit PCM. The AudioRecord class is then ready to use for recording. The stopRecording() method is run in a separate thread, which simply stops the audio from recording. The full code can be seen in GitHub, accessible via reference [10].

5.2.4.4 Activate button

There were important pieces of code in the speechrecognition class that weren't sourced/taken. For example, the code seen in code 5 runs when the activate button is pressed. A 4 second timer is set, where we see progress bar appear in the interface to indicate that something is loading. After the timer has finished, it will save the users code word in the database through the help of FirebaseDatabase and DatabaseReference class. It will then start the backgroundService activity.

```
protected void onCreate(Bundle savedInstanceState)  {
//...........
saveCodeWordinDB.setOnClickListener(new View.OnClickListener() { //onclick listener for the activate button
in speechrecognition screen, that adds the code word into Firebase Database
@Override
public void onClick(View v) { //onclick method
if (!ourtext.getText().toString().contains("Code Word")) { //if statement deals with whether user tries to
press button without registering a code word
Toast.makeText(speechrecognition.this, "You have not registered a code word!", Toast.LENGTH_LONG).show();
} else {
mProgressBar.setProgress(i); //otherwise set the progress for the mini progress bar
mCountDownTimer =new CountDownTimer(4000,1000) {//set a 4-sec timer before opening BackgroundService activity
@Override
public void onTick(long millisUntilFinished) {
i++;
mProgressBar.setVisibility(View.VISIBLE); //during the timer, make progress bar visible
saving.setVisibility(View.VISIBLE);
mProgressBar.setProgress((int) i * 100 / (5000 / 1000));
scrollView.post(new Runnable() { //runs a thread
public void run() {
scrollView.fullScroll(View.FOCUS_DOWN); //scrolls all the way down to the bottom of screen
}});
}
@Override
public void onFinish() { //once timer finished
i++;
mProgressBar.setProgress(100);
Bundle extras = getIntent().getExtras();
String value = extras.getString("email_var");
mDatabaseRef.child(value).child("codeWord").setValue(firstWord); //stores the code word in the database
Intent i = new Intent(speechrecognition.this, BackgroundService.class);
i.putExtra("email_var2", firstWord); //stores variables to use in nect activity
i.putExtra("email_var4", value);
if (SystemClock.elapsedRealtime() - mLastClickTime < 2800) {
return;
}
mLastClickTime = SystemClock.elapsedRealtime();
startActivity(i); //start the BackgroundService activity
}};
mCountDownTimer.start();
}}});
}
```

Code 5: onClick method implemented for activate button

5.2.5 BackgroundService & continuousService

The onCreate(Bundle SavedInstanceState) method in code 6 shows how backgroundService activity starts the running service. It uses Androids startService function that begins the continuous speech recognition process.

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_background_service);
//.............
startService(new Intent(BackgroundService.this, continuousService.class)); //starts the background service


AutoStart(); //some older devices do not allow background service to work continuously, so this function auto
starts the app just in case

}
```

Code 6: code showing how continuousService is started from BackgroundService class

Now, we investigate the code for the continuous background service. This is another important task for Speech Radar. Chapter 2 section 2.3 mentioned that the PocketSphinx library could be applied for this task. The library was used initially, but the process of getting good detection rate was complicated and would require a long period of time to get right. Thus, the Android SpeechRecognizer class is used instead. This gives access to Googles speech recognition service and can run in the background.

The onStartCommand method in code 7 is the first method implemented in continuousService class. Like how onCreate is called at the beginning of execution for activities, onStartCommand is similar in that aspect but works preferably with background services. It simply calls the Speech class, which is a library that instantiates the SpeechRecognizer class and provides easy definition to use for background service. The library can be seen in GitHub, accessible via reference [11]. Using the TensorFlow model to make predictions in the background would've been ideal, but this would eventually stop working. Thus, we rely on this library to listen or stop listening for speech continuously. At the end, it returns Service.START_STICKY, which recreates the service when there is enough memory.

```
@Override
public int onStartCommand(Intent intent, int a, int b) {

Speech tmp = new Speech(this.getApplicationContext()); //instantiating the speech class
myvar = continuousService.this; //interface, which require four methods to implement i.e. onStartOfSpeech(),
onSpeechRmsChanged(float value), onSpeechPartialResults(List<String> results), onSpeechResult(String result)

tmp.setListener(continuousService.this); //sets the listener
if (tmp.isListening()) { // if listening == true
tmp.stopListening(); // stop listening
setRingtoneSoundLevel(); // run function
} else {
RxPermissions.getInstance(continuousService.this).request(Manifest.permission.RECORD_AUDIO).subscribe(granted
-> { //asks for permission to record audio during runtime
if (granted) { //if permission granted
tmp.stopTextToSpeech(); //stops text to speech, as this app needs only speech-to-text
tmp.startListening(null, continuousService.this); //starts the voice recognition
}});
setRingtoneSoundLevel(); //run function
}
return START_STICKY; //tells operating system to re-create service, if their is enough memory
}
```

Code 7: onStartCommand implementation in continuousService

onSpeechResults(String results) seen in code 8 outputs the result of what the speech recognizer has detected in String format. This String is then used to make if statements that state whether it is equal to the code word. If it is, then the devices ringtone will start playing and will terminate the service completely. There are also if statements that express whether the result is equal to words that sound similar to the code word. For example, 'backward' is often misinterpreted by the speech recognizer as 'backwood.' If true, then ringtone will play. setRingtoneSoundLevel() is called in this method, which is explained in the next section.

```
@Override
public void onSpeechResult(String result) { //retrieves speech recognizers result as string
mContext = getApplicationContext(); //stores context of entire app
String arr[] = result.split(" ", 2); //splits the result to two elements
String val = BackgroundService.values2; //stores the code word
ArrayList<String> a = new ArrayList<String>(); //will store code word, if uttered speech from user is equal
to val
Uri uri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_RINGTONE);
Ringtone ringtone = RingtoneManager.getRingtone(mContext,uri);
if (!TextUtils.isEmpty(arr[0])) {
Toast.makeText(this, arr[0], Toast.LENGTH_SHORT).show(); //displays speech result on-screen
}
if(a.size() < 2) {
if (val.equals("Sheila")) { //if code word is equal to sheila
if (arr[0].equals("cheetah") || arr[0].equals("tschida")) { //and if recognizer detects tschida or cheetah
instead, which sounds similar to sheila
a.add(arr[0]); //add it to the arraylist
if (a.get(0).equals("cheetah") || arr[0].equals("tschida")) {
setRingtoneSoundLevel(); //function mutes all sounds apart from ringtone which is set to 90%

ringtone.play(); //play ringtone
stopService(new Intent(getApplicationContext(),MyService.class)); //stop the service straight after ringtone
is playing, as device has been found
}}}
if (val.equals("backward")) {
if (arr[0].equals("backwood")) {
a.add(arr[0]);
if (a.get(0).equals("backwood")) {
setRingtoneSoundLevel();
ringtone.play();
stopService(new Intent(getApplicationContext(),MyService.class));
}}}
if (arr[0].equals(val)) { //if first element equal to code word
a.add(arr[0]); //add it to the arraylist
}
if (a.get(0).equals(val)) { //and if first element of arraylist equal to code word
setRingtoneSoundLevel();
ringtone.play(); //play ringtone
stopService(new Intent(getApplicationContext(),MyService.class)); //stop the service straight after ringtone
is playing, as device has been found
}}}
```

Code 8: onSpeechResults implementation in continuousService

The method in code 9 mutes sounds such as media, notification and system, to block out the beep sound made by the speech recognizer. The ringtone sound is set to 90%, so when user utters the code word, it will play the ringtone loudly.

```
private void setRingtoneSoundLevel() { //mutes all different sounds on phone (e.g. media, system,
notification etc.), apart from ringtone, which is set to 90%
AudioManager amanager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
if (amanager != null) {
amanager.setStreamMute(AudioManager.STREAM_NOTIFICATION, true);
amanager.setStreamMute(AudioManager.STREAM_ALARM, true);
amanager.setStreamMute(AudioManager.STREAM_MUSIC, true);
amanager.setStreamMute(AudioManager.STREAM_SYSTEM, true);
int maxvol = amanager.getStreamMaxVolume(AudioManager.STREAM_RING);
float percentage = 0.9f; //set to 90%
int setvolume = (int) (maxvol * percentage);
amanager.setStreamVolume(AudioManager.STREAM_RING, setvolume, 0);
}}
```

Code 9: setRingtoneSoundLevel method implemented in continuousService

5.3 Firebase database

Firebase uses a NoSQL key-value database, which has a similar structure to a tree. There is a root node, which has many children nodes. Each child must have a unique label, so it was decided to use users email address without full-stop, since every email address is/has to be different. Each child contains more children, that define the account information of user (e.g. firstname, lastname, password, confirmpassword etc.) Screenshot of this can be seen in Appendix D.

5.4 XML

XML, similar to HTML is a mark-up language that uses tags to represent data. It is readable by humans and machine. XML is used in Android to design the layout, as it is lightweight which makes it lighter and quick to load on devices. All layouts made for each activity consist of a ScrollView at the top of the component tree to cater for small devices that cannot fit the whole screen and will need to scroll down. It then consists of a ConstraintLayout, which adds responsiveness to each activity. It defines the amount of spacing needed from each object (left, right, up and down), that translates for all screen sizes. The full XML code can be seen in Appendix G 10.35

XML is also used to define the manifest, which is a key resource file. It holds details required by Android about the app. It acts as link between the developer and the platform. No Android app can run without this file. The full code of the manifest can be found in Appendix G 10.35.1

Chapter 6 Testing

This chapter goes over the testing phase of the project. It will cover software and user testing, which will contain different test scenarios with expected outcomes to discover the number of successes and failures . Corrections will be made for all failures and proof of this and all other conducted tests can be seen in Appendix E.

6.1 Software testing

Software testing provides information about the standard of the program. One way It can do this is by investigating whether defined requirements have been met. This section will cover black box testing techniques such as functional testing, which will be used to produce valuable results of every function carried out in the app. Non-functional testing is another technique that will be used to tell us how Speech Radar performs. Static and dynamic testing will also be used together to find software bugs.

6.1.1 Functional testing

Proof of tests can be seen in Appendix E 10.24

Test	Test scenario	Example test case	Example test data	Precondition	Test steps	Expected outcome	Actual Results	Pass/fail
SPLASH SCREEN								
1	Test the splash screen	Test case 1: Test if splash screen loads when launching app	No data is tested	1. Requires a decent Internet connection 2. Must be on the splash screen	1. Launch application	Splash screen should load when user launches app. It should also open the login screen activity	Splash screen loads and opens the login screen activity	Pass
LOGIN SCREEN								
2	Test if the 'forgot password? ' link and 'Not a user? Register' link opens the correct activity	Test case 1: 'forgot password?' link should navigate user to the reset password activity Test case 2: 'Not a user? Register' link should navigate user to the create account screen	No data is tested	1. Requires a decent Internet connection 2. Must be on the login screen	1. Open application 2. Tap each link separately	Should navigate to the activities specified in the test case column	Navigates to the correct activities	Pass

3	Test if the log in button stops certain users from entering the speech recognition activity	<u>Test case 1:</u> If user inputs incorrect email address and password, then a pop-up should appear stating this <u>Test case 2:</u> If user inputs a correct email address but an incorrect password, then a pop-up should appear stating this <u>Test case 3:</u> If user has inputted correct email address and password, but has not verified their email, then a pop-up should appear stating this	<u>Test case 1:</u> Email address: hello123@gmail.com Password: hello321 <u>Test case 2:</u> Email address: Ishtiyaq93@gmail.com Password: melons169 <u>Test case 3:</u> Email address: ishtiyaq169@hotmail.com Password: hello123456	1. Requires a decent Internet connection 2. Must be on the login screen	1. Open application 2. input the email address and password for each test case stated in the example test data column 3. press log in button	Should see the correct pop-ups for each of the test cases mentioned in the example test case column	Correct pop-ups appear for each of the test cases	Pass
4	Test if the log in button allows authorised users to enter the speech recognition activity	<u>Test case 1:</u> If user inputs correct email address and password and has verified their email address	<u>Test case 1:</u> Email address: ishtiyaq93@gmail.com Password: hello123456	1. Requires a decent Internet connection 2. Must be on the login screen	1. Open application 2. input the email address and password for each test case stated in the example test data column 3. press log in button	Should allow user to access their account and open the speech recognition activity	User can access their account	Pass

RESET PASSWORD SCREEN								
5	Test if user is sent an email to reset their password	<u>Test case 1:</u> an email should be sent to the inputted email address. The user should be able to reset their password from the email and login to their account with the new password	<u>Test case 1:</u> Email address: ishtiyaq93@gmail.com New Password: watermelon101252	1. Requires a decent Internet connection 2. Must be on the reset password screen	1. Open application 2. navigate to the reset password activity by tapping the 'Forgot password? ' link 3. input the email address specified in the example test data column 4. open the email sent from the app. Tap the link 5. reset password	User should receive email and be able to login with the new password	User receives email and can access their account with new password	Pass
CREATE ACCOUNT SCREEN								
6	Test if the create account button stops certain users who've not followed the validation rules from creating an account	<u>Test case 1:</u> A pop-up should appear stating that there are errors. Input fields that were inputted incorrectly should be highlighted as red	<u>Test case 1:</u> First name: Martin Last name: Freeman Password: speed1 Confirm password: spe Email address: water	1. Requires a decent Internet connection 2. Must be on the create account screen	1. Open application 2. navigate to the create account activity by tapping the 'Not a user? Register' link 3. input test data specified in the example test data column 4. tap the create account button	Should stop user from creating an account and colour the incorrect input fields red. These input fields should be: password, confirm password and email address. A pop-up should also appear stating that there are errors	Stops user from creating an account and indicates them to fix their errors via pop-up and highlighting the incorrect input fields as red	Pass

7	Test if an email verification is sent when the create account button is pressed	Test case 1: Once user successfully creates an account, they should receive an email in which they must verify themselves by clicking a link	Test case 1: First name: Shah Last name: Ali Password: hello123456 Confirm password: hello123456 Email address: ishtiyaq93@gmail.com	1. Requires a decent Internet connection 2. Must be on the create account screen	1. Open application 2. navigate to the create account activity by tapping the 'Not a user? Register' link 3. input user details specified in the example test data column 4. tap the create account button	An email verification should be sent if user inputs the example test data	An email verification is sent	Pass
SPEECH RECOGNITION SCREEN								
8	Test if the mic button takes in speech as input and outputs it as text	Test case 1: Once user speaks and says a word, we should see text appear on screen that outputs what the user has said	Test case 1: Yes, zero , stop, learn	1. Requires a decent Internet connection 2. Must be on the speech recognition screen	1. Open application 2. login to your account 3. tap the mic button 4. Say all the words mentioned in the examples test data one-by-one 5. wait for output	The text should show the word user has said	The text shows the word user has uttered	Pass
9	Test if the activate button directs user to the correct activity and activates background service	Test case 1: The activate button should direct user to the background service activity	No data is tested	1. Requires a decent Internet connection 2. Must be on the speech recognition screen	1. Open application 2. login to your account 3. tap the mic button 4. Say the code word you want to register for your account 5. Tap the activate button	Button should direct user to the background service activity, where service should automatically run in the background	Button does direct user to the background service activity and service runs in background	Pass

BACKGROUND SERVICE								
10	Test if the background service stops running upon request	Test case 1: The service should stop when user utters code word or terminates the app	No data is tested	1. Requires a decent Internet connection 2. Must activate the background service	1. Open application 2. login to your account 3. tap the mic button 4. Say the code word you want to register for your account 5. Tap the activate button to switch on service 6. Utter the code word or terminate app	Background service should deactivate when user utters code word or user terminates app	The running service stops when user utters code word or terminates app	Pass

6.1.2 Non-functional testing

Proof of tests can be seen in Appendix E 10.25

Test	Test type	Test scenario	Precondition	Test steps	Expected outcome	Actual Results	Pass/fail
1	Database Security testing	Test the security of the database, by investigating whether the password and confirm password fields for all accounts are securely hashed	1. Requires a decent Internet connection 2. Must have access to the Firebase Console	1. login to your Firebase account 2. Click on database 3. Click on each account	The password and confirm password fields for all accounts should be hashed	The password and confirm password fields are all securely hashed	Pass
2	Storage testing	Test the storage size of the applications .apk file	1. Must have access to devices file directory	1. open My Files under settings 2. Go under downloads to see the size of the .apk file	The .apk file should be 70MB or less	The .apk file is around 59MB, so is under 70MB	Pass
3	Database recovery testing	Test if there is a recovery plan if database gets corrupted or erased	1. Requires a decent Internet connection 2. Must have access to the Firebase Console	1. login to your Firebase account 2. Click on database 3. Go under backups	There should be a recovery of the database containing all user accounts	There is no recovery present. Having a recovery requires payment with Firebase. Perhaps there may be an alternative solution in using SQLite as recovery database to save all user accounts	Fail

4	Compatibility testing	Test if app is device compatible	1. Requires a decent Internet connection 2. Must access the app via virtual device	1. Research on a list of different Android phone companies with different API levels 2. Open Android Studio and test the app using virtual device simulator	For all the phones the app is tested on, it should fully work for all	The app worked on all the devices	Pass
5	Database scalability testing	Test if Firebase allows us to create at least a million user accounts before exceeding 1 GB (would need to pay if it exceeds)	1. Requires a decent Internet connection 2. Must have access to the Firebase Console	1. open the app 2. navigate to the create account screen and insert an account into the database 3. login to your Firebase account, go under database and see how much bytes an account takes from storage 4. Calculate how much it would take from a million	Should be able to fit one million or more user accounts in the database under 1 GB of storage	Based on calculation, one account takes 780 bytes. Thus, one million user accounts would take 780000000 bytes which is the same as 780 megabytes. We can fit even more user accounts under 1 GB of storage	Pass
6	Load testing	Test if app can handle numerous users at once	1. Requires a decent Internet connection 2. Must access the app with multiple devices	1. open the app in 10 different devices using virtual and actual devices 2. Run the app with all devices at the same time	App should be able to handle multiple users at once and run at appropriate speed for each task	App can handle numerous users at once	Pass
7	Performance testing	Test if app takes most battery power of device	1. Requires an Android device	1. Open Settings 2. Navigate to battery	The app should take less than 1% of battery power, when running it in background for at least 3 hours	The app took only 0.4% of battery power	Pass

6.1.3 Static and dynamic testing

Static tests that have passed (contain defects) and all dynamic tests can be seen in Appendix E 10.26

Test	Test type	File/activity name	Defects to search for	Tools used	Test steps	Expected outcome	Actual Results	Pass /fail
LOGIN SCREEN								
1	Static testing	Loginscreen.java	1. Logical Defects 2. Arithmetic Defects 3. Syntax Defects 4. Interface Defects 5. Multithreading Defects 6. Performance Defects	Breakpoints and console	1. open loginscreen java file 2. Review the file to find any of the mentioned defects	To find at least one or more defects	No defects found!	Fail
2	Dynamic testing – unit testing	Loginscreen.java	1. unexpected exceptions 2. any unexpected errors	Logcat and output phone device	1. open loginscreen java file 2. Use the android debugging log utility to log all outputs 3. open the file and input email address and password. Then press the login button 4. Analyse the outputs using Logcat (Android Studio built-in tool)	If email address and password is valid, then a String variable should reformat the email address by removing the full stop. This is so it can search for the account in the database (screenshots of this can be seen in Appendix D), so it can extract first name value and display in speech recognition screen. Logcat should print the output below **Expected output:** Log.D(mytag1) - Ishtiyaq93@gmailcom	Expected outputs matches actual outputs! No defects found!	Fail
RESET PASSWORD SCREEN								
3	Static testing	forgotPassword.java	1. Logical Defects 2. Arithmetic Defects 3. Syntax Defects 4. Interface Defects 5. Multithreading Defects 6. Performance Defects	Breakpoints and console	1. open forgotPassword java file 2. Review the file to find any of the mentioned defects	To find at least one or more defects	No defects found!	Fail

4	Dynamic testing – unit testing	forgotPassword.java	1. unexpected exceptions 2. any unexpected errors	Logcat and output phone device	1. open forgotPassword java file 2. Use the android debugging log utility to log all outputs 3. input email address and then press button 4. Analyse the outputs using Logcat (Android Studio built-in tool)	If inputted email address is valid, then we should see a log that says email sent. If inputted email address is invalid, then we should see a log that says the opposite in Logcat **Expected output:** Valid email: Log.D(valid_input) – "email sent successfully to ishtiyaq93@gmail.com" Invalid email: Log.D(invalid_input) – "h1@gmail.com does not exist! Please try again"	Expected outputs matches actual outputs! No defects found!	Fail
CREATE ACCOUNT SCREEN								
5	Static testing	Createaccount.java	1. Logical Defects 2. Arithmetic Defects 3. Syntax Defects 4. Interface Defects 5. Multithreading Defects 6. Performance Defects	Breakpoints and console	1. open createaccount java file 2. Review the file to find any of the mentioned defects	To find at least one or more defects	No defects found!	Fail
6	Dynamic testing – unit testing	Createaccount.java	1. unexpected exceptions 2. any unexpected errors	Logcat and output phone device	1. open Createaccount java file 2. Use the android debugging log utility to log all outputs 3. input first name, last name, password, confirm password and email address. Then press create account button 4. Analyse the outputs using Logcat (Android Studio built-in tool)	If user has inputted a badly formatted email address, an incorrect password (e.g. less than 8 characters long etc.) or has not inputted anything, then we should see a log for each of those issues in Logcat **Expected output:** Log.D(badly_formated) – email address is badly formatted! Log.D(incorrectpassword) – password is incorrect! Please try again Log.D(no_input) – you have not inputted anything!	**Defect found!** Logs are printed for badly formatted email address and incorrect password, but a NullPointerException is thrown for when first name and last name are not inputted	Pass

SPEECH RECOGNITION SCREEN								
7	Static testing	Speechrecognition.java	1. Logical Defects 2. Arithmetic Defects 3. Syntax Defects 4. Interface Defects 5. Multithreading Defects 6. Performance Defects	Breakpoints and console	1. open Speechrecognition java file 2. Review the file to find any of the mentioned defects	To find at least one or more defects	No defects found!	Fail
8	Dynamic testing – unit testing	Speechrecognition.java	1. unexpected exceptions 2. any unexpected errors	Logcat and output phone device	1. open Speechrecognition java file 2. Use the android debugging log utility to log all outputs 3. press mic button. When colour of button is green, say a word out loud to generate an output as text. Once you have done this, press the mic button again. When colour of button is green, remain quiet 4. Analyse the outputs using Logcat (Android Studio built-in tool)	If user interacts with the mic button and says a word out loud (e.g. Sheila), a log in Logcat should show that word. If user remains quiet, a log stating that user should try again should appear **Expected output:** Log.D(Sheila) – code word is Shelia Log.D(undetected_word) – no word detected! Please try again	Expected outputs matches actual outputs! No defects found!	Fail
BACKGROUND SERVICE								
9	Static testing	BackgroundService.java	1. Logical Defects 2. Arithmetic Defects 3. Syntax Defects 4. Interface Defects 5. Multithreading Defects 6. Performance Defects	Breakpoints and console	1. open BackgroundService java file 2. Review the file to find any of the mentioned defects	To find at least one or more defects	**Defect found!** 1. Performance defect	Pass

6.2 User testing

User testing is a technique used to assess a system with real users. This section will describe all tests made with 216 participants for various scenarios. One test will get participants to test the user interface. All other tests will be dedicated to the speech recognizer. The aim of this is to better our understanding of how actual users interact with Speech Radar and how the app can be improved based on the obtained results.

6.2.1 User interface

Before asking their experience in using the interface, all 216 participants were asked about how they felt of the app idea. Nearly all users felt positively, which is proof that suggests Speech Radar is needed. The result for this can be found in Appendix E 10.27.1. All remaining questions were related to the interface, which asked the difficulty level of using each screen. Most users stated it was easy overall, but there were some that stated the login and create account screen were difficult. Participants were then asked whether they could point out any improvements to the interface. 13 responses were collected, and some were quite interesting. For example, one user suggested that login screen should remember user details to require less user input. Another participant suggested that the interface was too static and needs a bit of animation. Final interesting suggestion stated that the interface should use colours that match the logo, which are blue and black. These improvements made total sense and will be made to the app. Result of the test and changes can be seen in Appendix E 10.27.2

6.2.2 In-app speech recognizer

As a quick reminder, the in-app speech recognizer is located in speech recognition screen and uses the TensorFlow model to make predictions. The tests were conducted to see how many attempts it would take for the system to recognize the code word that user utters. This will tell us its quality level. The maximum number of attempts are 3 and all code words are tested by 12 people each. Thus, the test is only successful if system is able recognize each participants word in under 3 attempts. Thankfully, the bar chart in Appendix E 10.27.3 show that the speech recognizer was able to detect all 216 participants, which confirms that this test was successful. What's even more satisfying, is that the system was able to recognise most participants code word in one attempt.

The other graph in Appendix E 10.27.3 shows the most successful devices in percentage, that required less attempts in recognising the code word. The purpose of this is to generate a bit of statistic that may help for future development (e.g. it may help to identify the weakest device, so the app can be fine-tuned to work properly.) The graph shows that Samsung was the most successful with it reaching 95%. Given that the pie chart in Appendix E 10.27.5 shows that Samsung was the popular device, it suggests how great their devices are in recognising human speech. The least successful was Alcatel.

6.2.3 Background speech recognizer

Just to reiterate from previous sections, the background speech recognizer works continuously as a service and uses Google's speech recognizer. The plan for this test is to experiment in various scenarios such as placing the handheld device on the floor, under a thick furniture and on top of a tall furniture. This is to see how Speech Radar responds when device is in spots that would most likely be the case when user is in need of the app. The same approach taken in the previous section is used for this test. The way we would know that it worked is if the handheld device starts ringing. The bar chart in Appendix E 10.27.4 shows the number of attempts taken for every given scenario. Based on analysis, it took one attempt for the speech recognizer to recognize most participants code word in every scenario. The first scenario (i.e. device on the floor) was most successful, as the distance was not too far and there weren't thick objects blocking audio signals travelling to the device. The other two scenarios have similar results and were the least successful. This was due to users initially thinking that their average ranged voice would travel through thick object or long distances. Thus, the realisation of using a louder voice came to them after the first attempt. This is interesting as it describes how users initially think and can be used to help in future development.

6.3 Self-experimentation

6.3.1 Distance testing

Since this would not be feasible with users, self-experimentation was done to get an idea of how far the background speech recognizer could hear and correctly detect the code word. The result in figure 22 shows different code words being used for each test. It also shows the background speech recognizer was able to reach up to 6 metres, which of course is a complement to Google. This very much verifies that the app is capable of working in the background and carrying out its duty.

Distance (metres)	Code word	Attempt 1	Attempt 2	Attempt 3
0.5	yes	pass		
1	stop	pass		
1.5	down	pass		
2	left	fail	pass	
2.5	right	pass		
3	tree	pass		
3.5	three	pass		
4	marvin	fail	pass	
4.5	sheila	fail	pass	
5	visual	fail	fail	pass
5.5	house	fail	pass	
6	happy	fail	pass	
6.5	six	fail	fail	fail

Figure 22: distance testing results

6.4 Evaluation

All requirements (apart from one) stated in the Specification chapter have been met. This is proven to be true when investigating the Testing chapter, Implementation chapter and the appendices. The only requirement that hasn't been met is the possession of a recovery database of all user accounts. This was identified in the Testing chapter and has been corrected via the use of SQLite. Proof of this can be seen in Appendix E 10.25.3.

Chapter 7 Conclusion

This chapter ends the report by evaluating the success and failures of the project. In addition, the chapter will evaluate whether all original expected requirements were met, how the app compared to what others have done and future work that can be done to improve the app.

7.1 Project success

All tasks were finished on schedule and subsequently, a great deal of success has come out of this project. For example, speech recognizer has been tested to have a great detection rate. The application can run on devices with low API levels beginning from 16 (Jelly Bean). Source code has been organised, so it is easy to work with in the future. In addition, the application has been tested thoroughly to handle various failures (e.g. try-catch blocks have helped to catch NullPointerExceptions when user has not inputted anything in login screen or create account.) But the best success would be that all system requirements were met, suggesting how the project accomplished everything that was originally set out.

7.2 Project failures

The application contains failures which were not fixed due to project time limitation and/or infeasibility. For example, Speech Radars background service is not entirely reliable, as it can crash every once in a while, due to not known reasons. The background service is also reliant on good Internet connection. Else it can take a long time to retrieve results from uttered speech. In rare instances when user utters the code word, ringtone loops forever and does not stop playing, which can be an inconvenience to users.

7.3 Similar apps

7.3.1 Find My Phone Whistle

This Android application requires users to whistle for the device to play a quick sound. It is successful most of the times, however it is sensitive to small sounds such as a cough and seems to think a whistle sound is made. This suggests that the recognizer is not as good as Speech Radars. What's more, the application does not automatically set the media volume, which users may forget to do. Thus, the volume could potentially be on silent, rendering the application to be useless. Speech Radar enforces the ringtone volume to be at 90% when the background service is up and running. The interface on the other hand is much more intuitive and eye-catching.

7.3.2 Voice to Find My Phone

Like Speech Radar, this app plays a ringtone when a code word is uttered. However, Voice to Find My Phone has a poor user interface, which makes it hard to navigate and understand. Speech Radar is far

easier to comprehend and requires less user input. The background service of this app is decent and can detect the word from long distances. However, the creator made it difficult to kill the continuous service. Thus, the ringtone keeps playing non-stop. Speech Radar offers different ways to kill the service. Either user utters the code word or terminates the app.

7.4 Future work

Below, is a list of tasks to meet further requirements that may surface in the future:

- Build an iOS version of the app to expand target audience
- Use the PocketSphinx library to create an offline background speech recognizer, so users can find their phone in situations where Internet is not available
- Add cooler code words to make it less boring
- Enable users to choose what sound they want to play when code word is uttered
- Allow users to find phone by clapping or whistling
- Allow users to deactivate or reactivate the service by uttering a specific word

Chapter 8 Self-Evaluation

In this project, I built a successful speech recognition Android application. What I am most happy about is that all initial designs were properly implemented. It was nice to see this because in past projects, the implementation would be entirely different to the designs. The best thing I have obtained was a better understanding of Java Android platform, which will be invaluable for future projects. I am most pleased to see the background service work the way I pictured it. At first, I was doubting whether getting speech recognition to run continuously in the background would be feasible for the time I had. Fortunately, there were classes in Java that helped ease the implementation and reduce time taken for tasks. The only downside would be that the service requires decent Internet connection to run. If it had worked offline, then the background service would have some credibility to run as expected in any given scenario.

Overall, the experience was extremely useful in shaping me to achieve more in the future. Although there were tasks that felt overwhelming, it did help develop my problem-solving skills. For example, when I found out that certain phones such as Samsung Galaxy S6 required a specific permission for applications to mute all sounds, it became immediately apparent to me that a simple if statement and try-catch block would do the trick.

If Speech Radar was slightly more developed, I can picture it being used by many people and making a positive difference to their lives. I additionally see it contribute to speech recognition in the future and the app idea being used by major companies such as Samsung, Sony, LG etc.

Chapter 9 Bibliography

[1] Erin Myers. "Little Known Facts About Speech Recognition Technology." (2017). Retrieved from https://www.temi.com/blog/little-known-facts-about-speech-recognition-technology/

[2] Peter Samoff. "Alexa is getting better at answering users' questions." (2018). Retrieved from https://www.businessinsider.com/amazon-improving-alexa-voice-accuracy-2018-12?r=US&IR=T

[3] Clark Boyd. "The Past, Present, and Future of Speech Recognition Technology." (2018). Retrieved from https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf

[4] Sainath, Tara, and Carolina Parada. "Convolutional neural networks for small-footprint keyword spotting." (2015). *Keyword Spotting Task*, p. 1, *CNN architectures,* p. 2

[5] Warden, Pete. "Speech commands: A dataset for limited-vocabulary speech recognition." (2018). *arXiv preprint arXiv:1804.03209*(2018). *Introduction*, p.1, *Word-choice* p.4, *Properties* p.6

[6] Sharma, Ajay Shiv, and Rahul Bhalley. "ASR—A real-time speech recognition on portable devices." (2016). *2nd International Conference on Advances in Computing, Communication, & Automation*

[7] TensorFlow. "Simple Audio Recognition." Retrieved from https://www.tensorflow.org/tutorials/sequences/audio_recognition#running_the_model_in_an_android_app

[8] TensorFlow. GitHub. "speech_commands" Retrieved from https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/speech_commands

[9] TensorFlow. GitHub. "RecognizeCommands.java." (2017). Retrieved from https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/android/src/org/tensorflow/demo/RecognizeCommands.java

[10] TensorFlow. GitHub. "SpeechActivity.java." (2018). Retrieved from https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/android/src/org/tensorflow/demo/SpeechActivity.java

[11] gotev. "Android speech recognition and text to speech made easy." (2016). Retrieved from https://github.com/gotev/android-speech

[12] sachinvarma. "Speech-Recognizer." (2018). Retrieved from https://github.com/sachinvarma/Speech-Recognizer/blob/master/app/src/main/java/com/sac/speechdemo/MyService.java

[13] Android Developers. "ScrollView." Retrieved from https://developer.android.com/reference/android/widget/ScrollView

[14] Android Developers. “ConstraintLayout.” Retrieved from https://developer.android.com/reference/android/support/constraint/ConstraintLayout

[15] Android Developers. “SpeechRecognizer.” Retrieved from https://developer.android.com/reference/android/speech/SpeechRecognizer

[16] Android Developers. “Thread.” Retrieved from https://developer.android.com/reference/java/lang/Thread

[17] Android Developers. “Save data using SQLite.” Retrieved from https://developer.android.com/training/data-storage/sqlite

[18] Google Developers. “FirebaseAuth.” Retrieved from https://developers.google.com/android/reference/com/google/firebase/auth/FirebaseAuth

[19] Google Developers. “FirebaseDatabase.” Retrieved from https://developers.google.com/android/reference/com/google/firebase/database/FirebaseDatabase

[20] Google Firebase. “DatabaseReference.” Retrieved from https://firebase.google.com/docs/reference/android/com/google/firebase/database/DatabaseReference

[21] Android Developers. “Create a background service.” Retrieved from https://developer.android.com/training/run-background-service/create-service

[22] faruktoptas. “An easy-to-use customisable show case view with circular reveal animation.” (2017). Retrieved from https://github.com/faruktoptas/FancyShowCaseView

[23] mvnrepository. “JavaMail API For Android.” (2018). Retrieved from https://mvnrepository.com/artifact/com.sun.mail/android-mail/1.6.2

[24] tbruyelle. “Android runtime permissions powered by RxJava2.” (2015). Retrieved from https://github.com/tbruyelle/RxPermissions

[25] case540. “TensorFlowInferenceInterface.java.” (2015). Retrieved from https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/android/java/org/tensorflow/contrib/android/TensorFlowInferenceInterface.java

[26] Google Play. Appache Tools. “Find My Phone Whistle.” Retrieved from https://play.google.com/store/apps/details?id=ru.appache.findphonebywhistle&hl=en_GB

[27] Google Play. Chromatic Zone. “Voice To Find My Phone.” Retrieved from https://play.google.com/store/apps/details?id=com.chromaticzone.voicetofindphone&hl=en_US

Appendix A Preliminary Project Report

STUDENT NAME: SHAH ALI

DEGREE TITLE: COMPUTER SCIENCE

SUPERVISOR NAME: JAMIE WARD

REGISTRATION NUMBER: 33455846

INSTITUTION: GOLDSMITHS,
UNIVERSITY OF LONDON

Shah Ali 33455846

Content page

1. Introduction

In the present society, its normal seeing individuals in homes and workplaces use speech recognition. *With over 10 million Alexa devices sold worldwide and 41 million monthly active Siri users* [1], it's clear to see that a lot of people are taking a massive interest in speaking to virtual assistants. This may be because of how well its detecting words correctly in spite of different accents and background noise. It may also be because of the things it can do that make people's lives easier. For example, an Alexa device can *control a smart home (e.g. turn on the hot water, open or close garage door etc.), get the news, make phone calls* and many more [2]. Since speech recognition is such a topic of interest now, I wanted to base my project in this area so I could contribute to it. The artificial intelligence module I took in first term is very relevant to my project and inspired me to take on this challenge. We learnt how to use TensorFlow, Keras, NumPy, matplotlib and many other libraries that will be needed to attempt this project. On a weekly basis, I have been meeting my supervisor and engaging in discussion over my research and implementation I have done so far.

2. Aims and objectives

a. Aim and objective 1: successful implementation

The main goal of my project is to enable users to search for their phone in a more modern and efficient way. Speech recognition is something that's getting better through inventions like Amazon Alexa and Google Hub as proof. I aim to implement this amazing deep learning task to the best of my abilities and test it thoroughly, so its detection rate is at an appropriate level and is something users can rely on. The way I will achieve this aim is by using TensorFlow and Android Studio interchangeably. TensorFlow is an open-source library that is great in tackling most deep learning problems and is easy to use once you get used to it. Android Studio is going to be our IDE where most of the implementation is going to take place. The great thing about Android Studio is that it enables users to design their app efficiently and as a bonus it works well with TensorFlow.

b. Aim and objective 2: accent detection and appropriate dataset

At the beginning of the project, I aimed to create a speech recognition software that could predict majority of words in the English dictionary and detect different accents at the same time. Based on recent research, I realised that I should aim to only detect a small set of words of around 30 – 60, as I would need to search for millions of audio clips and data to cover the full dictionary. Given the time I have for this project it will simply not be possible for me to

gather it all. Thus, I intend to create a speech recognition that can detect different accents, but with a dataset that is appropriate for the time I have.

3. Methods

a. Aim and objective 1 step-by-step

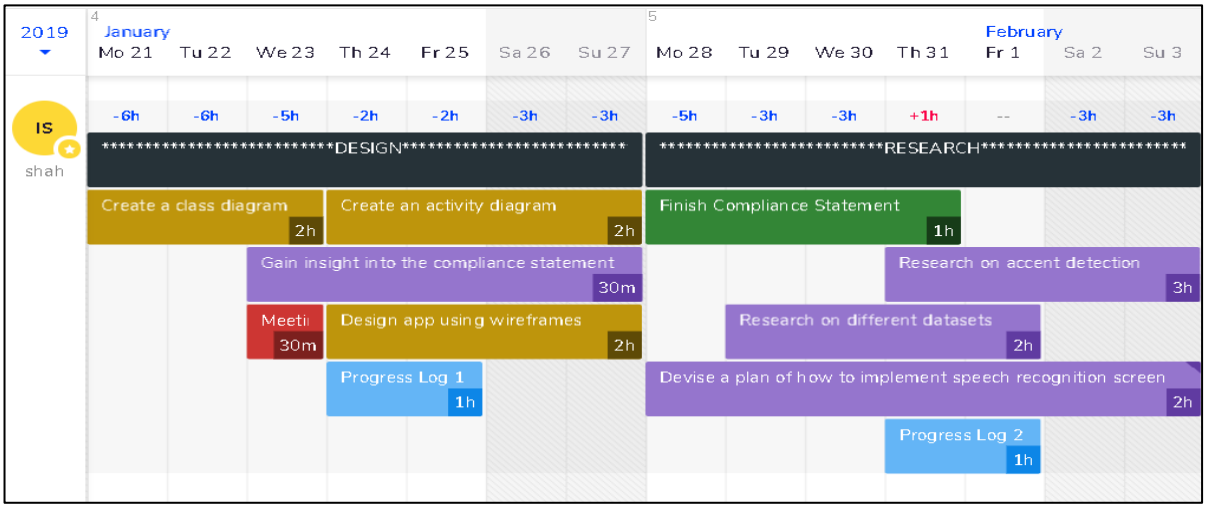
- Create a deep neural network in TensorFlow, which will be trained, validated and tested
- Then, *obtain a frozen version of your model as a protobuf file* [6] and place it in the assets folder within Android Studio. Next, compile TensorFlow into Android Studio via the *build.gradle* [6]. This will allow me to use the TensorFlowInferenceInterface class, which will be needed to feed in inputs and to get the prediction results
- Once the above step is complete, get the model to load successfully into Android Studio
- Once the model has loaded, insert a button that records audio in the app. Store the audio as bytes and *convert it into a floating-point array* [7] and then pass the results into a defined function, which should predict the utterance from the user and choose the correct label and its probability
- Finally, I will get the app to run in the background of the users phone and have it detect voices. Once it recognises the registered code word (e.g. frog), the phone will start vibrating and ringing loudly. This is something I haven't looked into in great detail, but I found out that an *API called Sphinx* [5] is something worth researching and will help in carrying out this task

b. Aim and objective 2 step-by-step

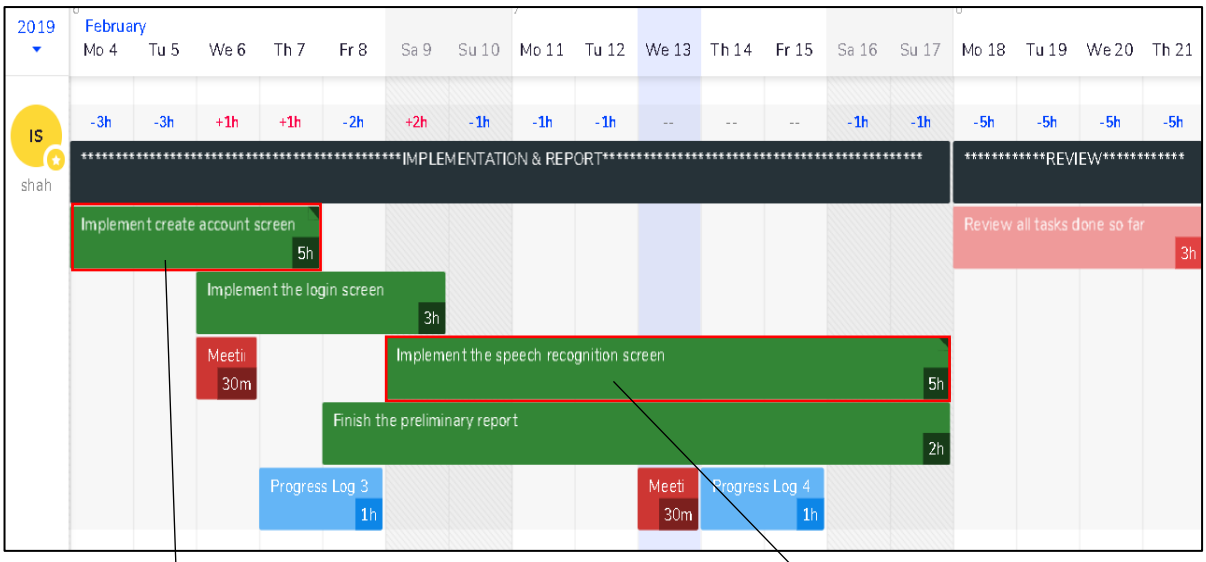
- I looked for many datasets online and found so many potential candidates. But the most appropriate one I found was the Kaggle speech commands dataset as it seemed that it would be easy to work with due to its decent size and neat organization of the audio clips

4. Project plan

a. Design & research



b. Implementation, preliminary report & review



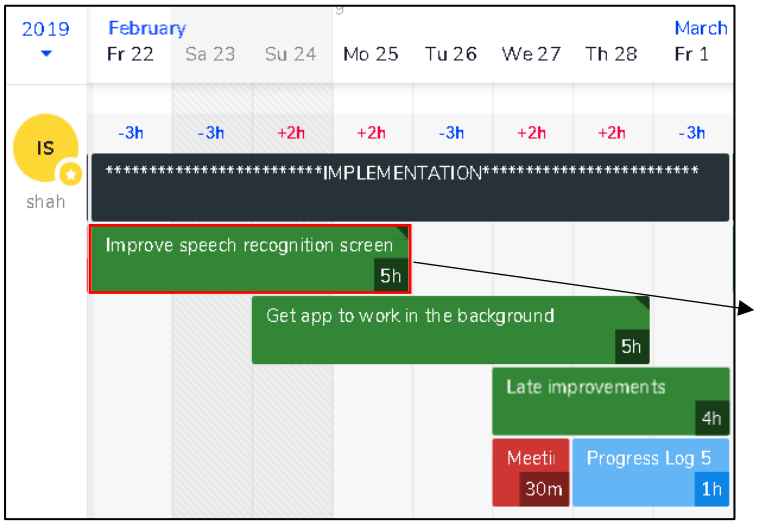
Sub-tasks:

1. Validate each input field
2. Insert the data into the Firebase Database once user clicks on the create account button
3. Send an email verification to the inputted email address

Sub-tasks:

The sub-tasks for this task can be seen in Methods 3a

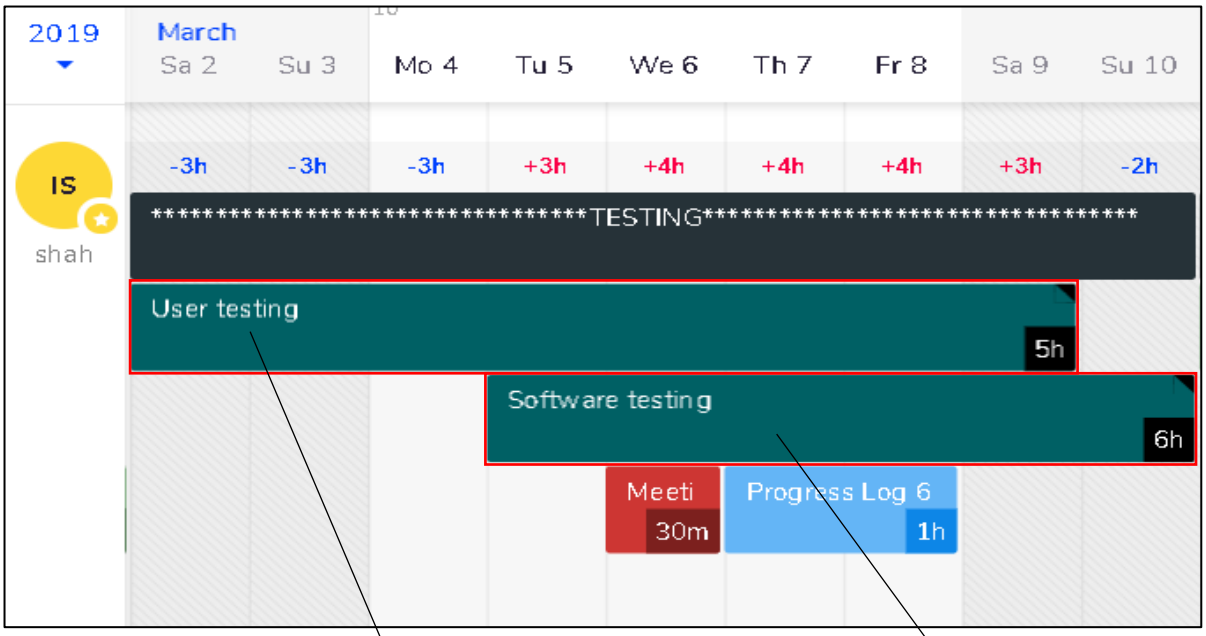
c. Implementation (continued)



Sub-tasks:

1. Improve the TensorFlow model (e.g. fine-tune hyperparameters etc.)
2. Maybe add more words for recognition
3. Enhance its appearance

d. Testing



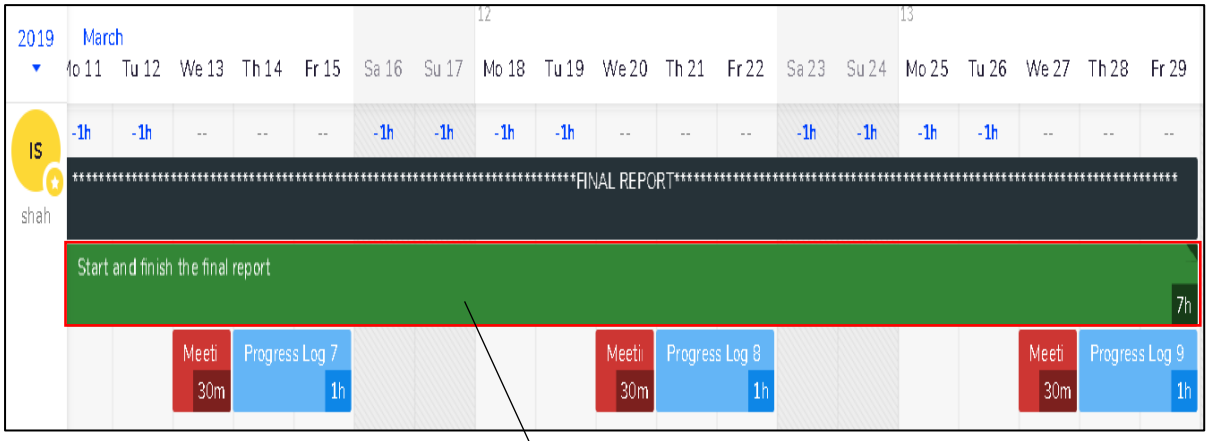
Sub-tasks:

1. Create a survey with questions that require ratings of the app
2. Look for random people in the university, outside university or online and have them test the app and fill the form out
3. Get a large quantity of feedback and make sure to adhere to rules in compliance statement
4. Rectify any issues in the app based on the user feedback

Sub-tasks:

1. Cover functional testing
2. Cover non-functional testing
3. Cover static testing
4. Cover dynamic testing

e. Final report



Sub-tasks:

1. Start off in completing the literature review, methods and results section on the first week
2. Then, complete the discussion section on the second week
3. Finish off with a good introduction in the beginning and a conclusion at the end also on the second week
4. Add all the appendices and references at the end of the report on the third week
5. Add the title page, contents page etc on the third week
6. For the final few days, review everything and add any improvements if necessary

5. Progress to date

a. Design phase

At the beginning of my project, I began the design phase. With insufficient knowledge in what my plan was for this project, it may seem to most people that gaining insight/research should have been what I started doing first. The reason why I chose this order is because I wanted to brainstorm my ideas/thoughts of how I think my app should function and look like. It will be interesting to see at the end what I did in the design phase and compare it to my actual implementation. Just below, are the tasks I did in my first week:

- I created a class diagram to describe the app structure. Can be seen in appendix 7a
- I created an activity diagram to describe the app behaviour. Can be seen in appendix 7b
- I designed the appearance of the app using wireframes. Can be seen in appendix 7c

b. Research phase

A paper I read online helped me to understand how I could convert sound into data. Suppose we have a dataset that contains a set of WAVE files, that are all one-second long. *Each file must be represented in a vector with a sampling rate of 16,000* [3]. We then *extract the features from the raw waveform and convert it into time and frequency domain* [3]. This technique is known as MFCC. I was further encouraged when reading this paper that I could use a library called *Librosa* [3] to do this easily in Python.

Another paper I read online gave me insight into the speech commands dataset. It was publicly provided by Kaggle for users to compete in their audio recognition competition. The dataset is an appropriate size for my project and *consists of 30 words* [4] in total. It contains thousands of one second audio clips for each word uttered in numerous accents. It got me thinking that users of my app could utter one of these words as their code to find their phone.

In the final stage of research, I devised a visual plan of how to implement the speech recognition screen on Android Studio. This can be seen in appendix 7d.

c. Implementation phase

During this phase, I was able to complete my compliance statement and finish my implementation of the create account screen, where users are able to register and have their details sent to the Firebase database safely. The account passwords are securely hashed, so attackers cannot steal passwords if they tried to gain unauthorised access to the backend. Users are also expected to verify via email for authentication. I started my implementation at

an early stage in the attempt to gain more time to spend for implementing the speech recognition screen. Evidence can be seen in appendix 7e.

I finished my implementation of the login screen where the input fields check whether the account exists in Firebase, the information inputted is correct and if the email address has been verified by the user. The login screen can also be seen in appendix 7e.

I began my implementation of the speech recognition screen not too long ago. I have had a few setbacks here and there, but I am confident I can finish on time and ensure it functions the way I want it to. So far, I have designed the screen and got it to recognise three words ('cat', 'bed' and 'happy'.) The speech recognition screen can also be seen in appendix 7e.

d. Summary

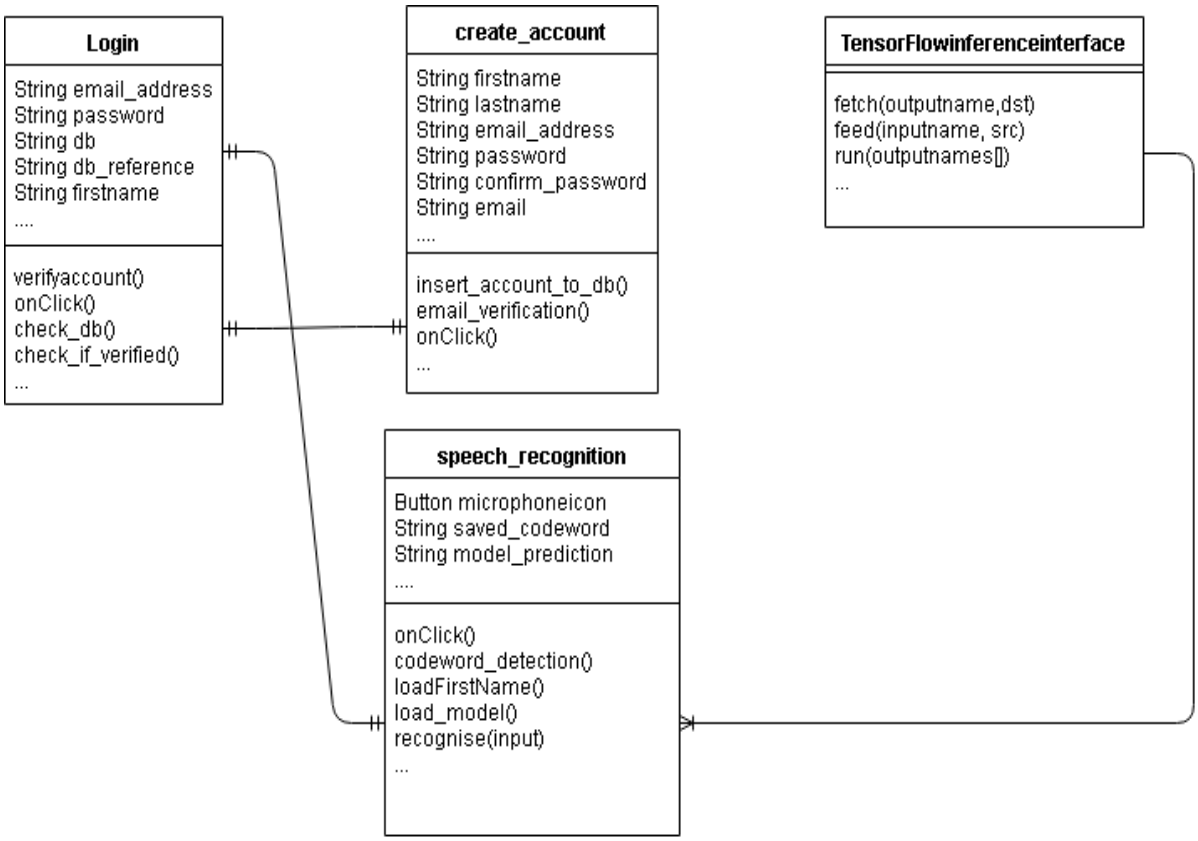
I think I completed all tasks I set out to do up until this point with a few minor setbacks here and there in which I've overcome. One of the key tasks I did previously and am pleased in doing now is researching how to convert audio into recognisable data. It helped me understand the logic behind how the input is generated for my TensorFlow model. This knowledge has assisted me to generate the input in Android Studio using Java, which has now helped in making accurate predictions.

6. Planned work

Based on what I mentioned in the previous section, I completed the design and research phase of my project. Now I am in the implementation phase and have created a quarter of my final application on Android Studio. I have recently increased the amount of time and work I do daily. What I need to do next is to finish off the implementation to quickly get into the testing phase where we will begin user testing and software testing (e.g. functional, non-functional etc), which will go on for quite some time. If this all goes to plan, then I should have enough time at the end for the final report.

7. Appendices

a. Class diagram: entire app



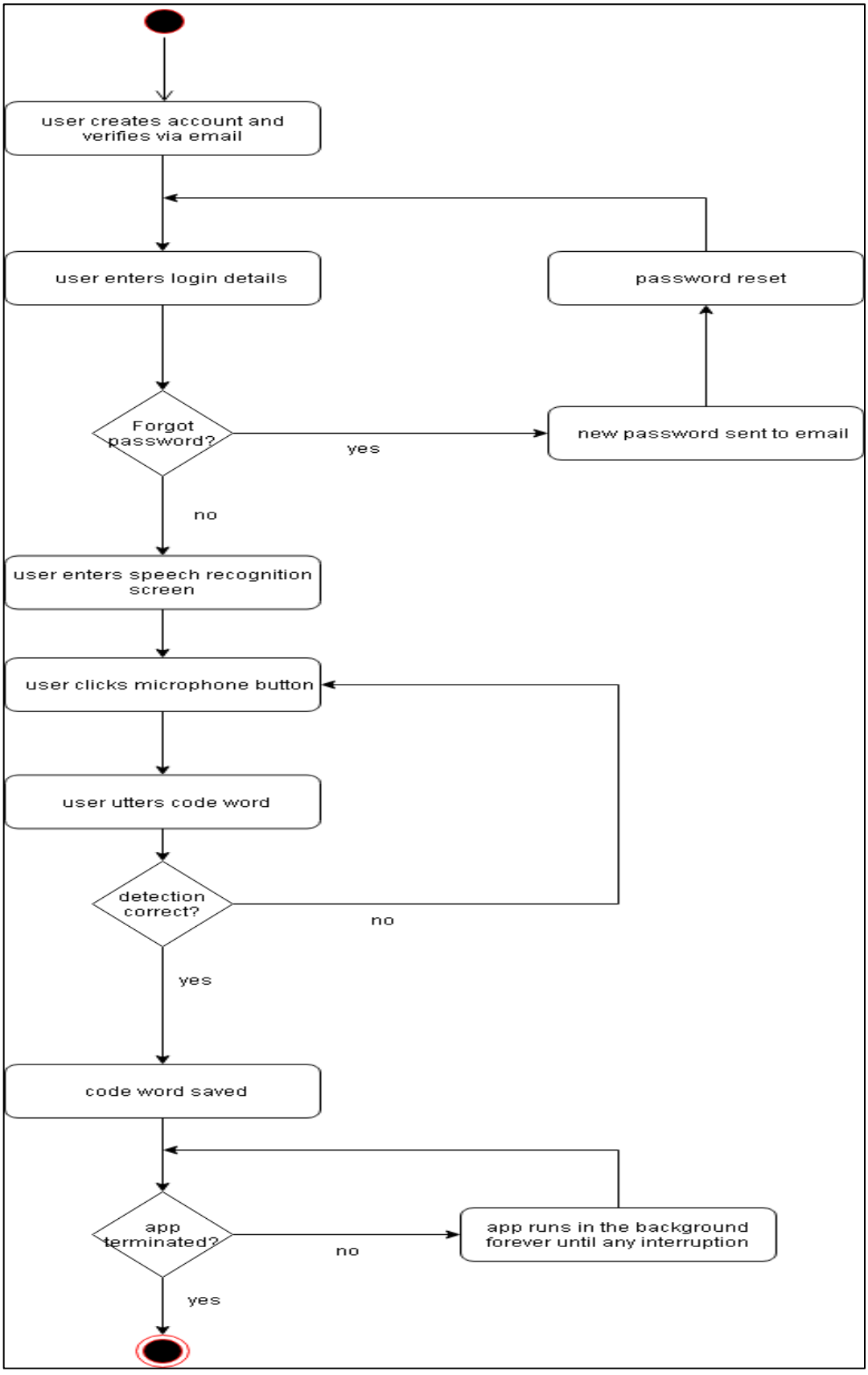
Brief analysis

create_account and login are one-to-one, as one email address can only be used to create an account and one email address can only be used to login to the app

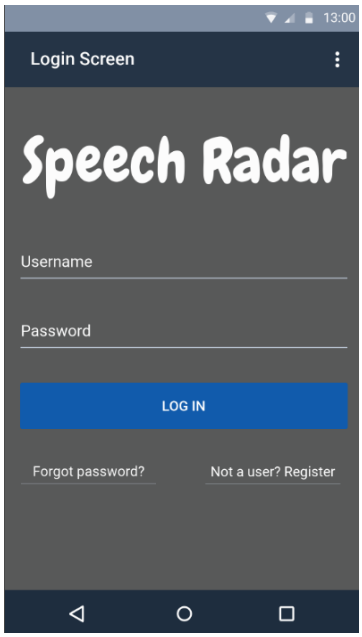
speech_recognition is one-to-one with login, as we extract one firstname from a stored account in our backend database within the login screen and display it in speech_recognition only once

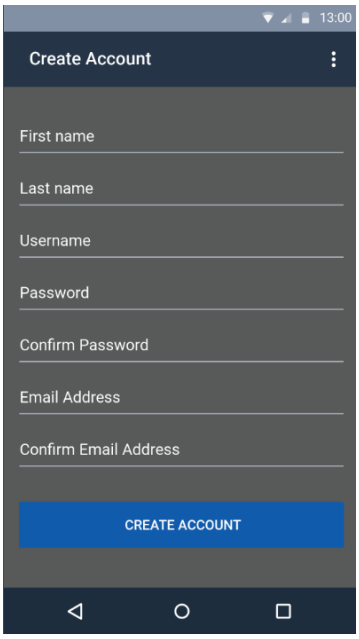
TensorFlowinferenceinterface is one-to-many with speech_recognition as the speech_recognition class will use many functions from the pre-defined TensorFlowinferenceinterface class

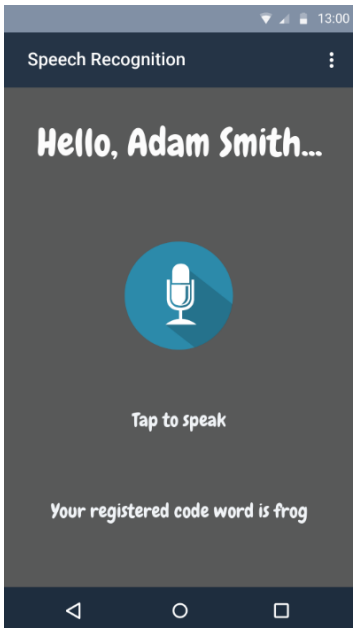
b. Activity diagram

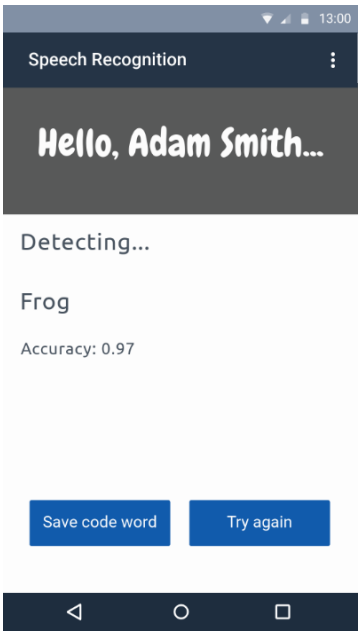


c. Wireframes

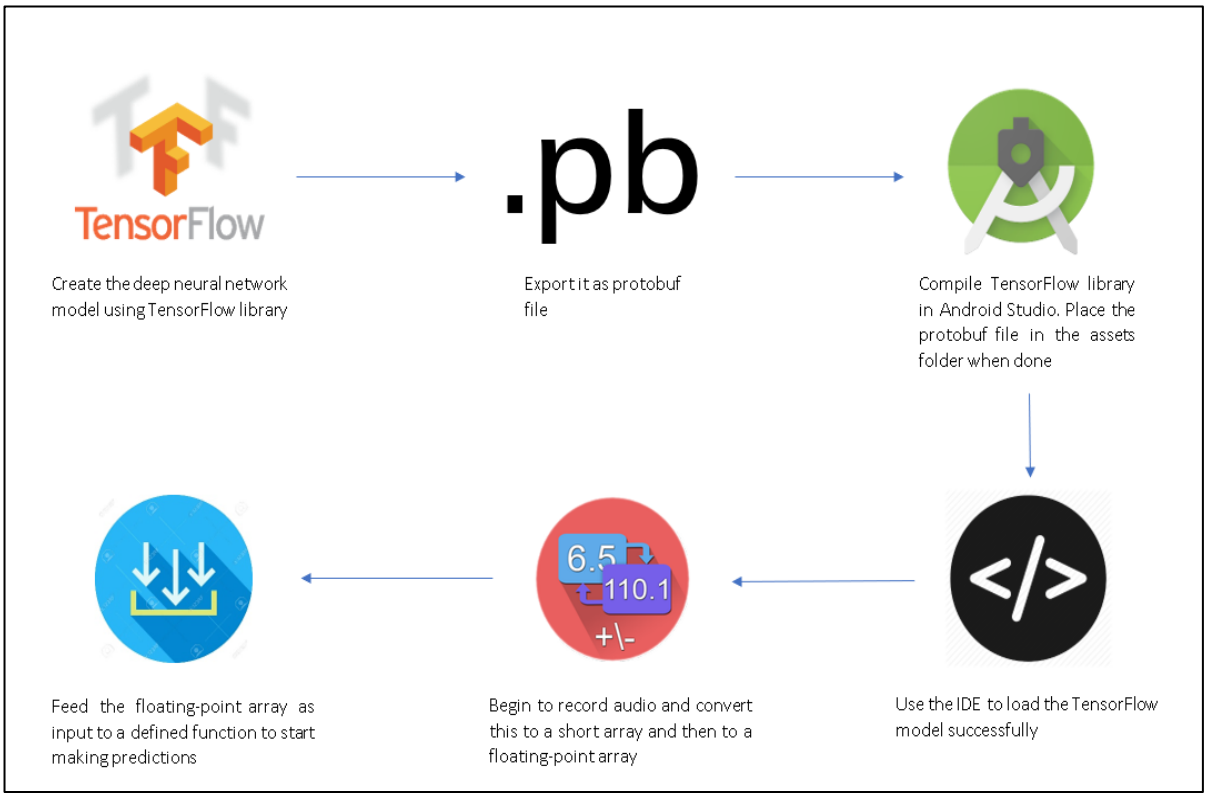








d. Speech recognition screen plan



e. Actual implementation for each screen

8. Reference list

[1] Erin Myers. "Little Known Facts About Speech Recognition Technology." (2017). Retrieved from https://www.temi.com/blog/little-known-facts-about-speech-recognition-technology/

[2] Sharon Profis. "10 of the best things you can do with the Amazon Echo." (2019). Retrieved from https://www.cnet.com/how-to/the-best-things-you-can-do-with-amazon-echo/

[3] Patrick Jansson. "Single-word speech recognition with Convolutional Neural Networks on raw waveforms." (2018). *Sound as data*, p. 9, *Data, tools and pre-experiments*, p. 14

[4] He Liangbo, and Hao Sun. "Attention Incorporate Network: A network can adapt various data size." arXiv preprint arXiv:1806.03961 (2018). *4.2 Audio recognition*, p. 7

[5] LaoFu. "Voice recognition background service possible." (2012). Retrieved from https://www.reddit.com/r/androiddev/comments/r7zea/voice_recognition_background_service_possible/

[6] Yoni Tsafir. "Deploying a TensorFlow model to Android." (2017). Retrieved from https://medium.com/joytunes/deploying-a-tensorflow-model-to-android-69d04d1b0cba

[7] Himanshu Bhutani, “Android:- convert a recorded audio file into a float array .“ (2017). Retrieved from https://stackoverflow.com/questions/42153056/android-convert-a-recorded-audio-file-into-a-float-array

Appendix B Progress Logs

10.1 LOG 1 - 21/01/19 – 27/01/19

Task	Task name	Begin Date	Due date	Time spent	Completed?
1	Create class diagram to describe app structure	21/01/19	23/01/19	2 hours	Yes
2	Create activity diagram to describe app behaviour	24/01/19	25/01/19	1 hour	Yes
3	Design the app using wireframes	25/01/19	26/01/19	2 hours	Yes
4	Gain insight into the compliance statement and make a start on it	26/01/19	27/01/19	3 hours	Yes
				8 hours	0 tasks remaining

Purpose of tasks

For the first week of my project, I began designing almost immediately after the research I did during the winter holiday. I designed the structure and behaviour of the application using UML diagrams. I also designed the appearance of each screen using wireframes, in the hope of visualising how the app would look like. As for the compliance statement, I gained insight into its purpose and was able to start it off. I aim to finish the statement by the end of next week

Discussion with supervisor

Since it was the first week and exams were in progress, we came to the conclusion that a discussion next week would be more useful. However, we were able to come to an agreement of a list of tasks I can complete before we have our discussion again. List can be seen below.

List of tasks to complete for next week:

- Research on how I could solve the issue in detecting accents
- Research more on the Voxforge dataset
- Have some form of plan for implementation
- Begin implementation

10.2 LOG 2 – 28/01/19 – 03/02/19

Task	Task name	Begin Date	Due date	Time spent	Completed?	Comments
1	Finish the compliance statement	28/01/19	29/01/19	5 hours	Yes	
2	Implement the create account screen and setup email verification for authentication on Android Studio	29/01/19	31/01/19	13 hours	Yes	Initially I planned to use MYSQL to store the user accounts, but I have changed to Firebase. I discovered that I can setup email verification for authentication easily with this Google service. I have also discovered that it's cloud based, so I can gain easy access
3	Implement the login screen on Android Studio	31/01/19	01/02/19	8 hours	Yes	
4	Devise a detailed step-by-step plan of how to implement the speech recognition screen	01/02/19	02/02/19	5 hours	Yes	
5	Research on how I could solve the issue in detecting accents.	02/02/19	03/02/19	0 hours	No	Did not have enough time to get around to this task. Should definitely get done by next week
6	Research more on the Voxforge dataset	02/02/19	03/02/19	0 hours	No	Did not have enough time to get around to this task. Should definitely get done by next week
				31 hours	2 tasks remaining	

Purpose of tasks

For the second week of my project I began implementing the login and create account screen, along with setting-up the backend-database where the user accounts will be stored. I've also created an email verification system for authentication when users create an account. For the compliance statement, I made it clear that I will adhere to the set of rules (that apply to my project) mentioned in the pdf file specified in the Computing Project page. Finally, I took the time to devise a detailed plan of how I can implement the speech recognition screen as this will be my main focus area for the coming weeks.

Discussion with supervisor

We couldn't meet this week, as I had a job interview elsewhere. However, I was able to come up with tasks that I could potentially do next week. List can be seen below

List of tasks to complete for next week:

- Prioritise researching on solving the issue in detecting accents and on the Voxforge dataset
- Begin implementation of the speech recognition screen
- Make a start on the preliminary report

10.3 LOG 3 - 04/02/19 – 10/02/19

Task	Task name	Begin Date	Due date	Time spent	Completed?	Comments
1	Prioritise researching on solving the issue in detecting accents and on a potential dataset we can use for our TensorFlow model	04/02/19	05/02/19	20 hours	Yes	
2	Begin implementation of the speech recognition screen	05/02/19	10/02/19	65 hours	Yes	I was able to create my speech recognition model using TensorFlow and load the model on to Android Studio. However, I couldn't get the application to make predictions accurately. This is something I've been trying to fix all week and was unable to find any solutions, so I am looking for alternative methods of how I could implement the speech recognition in the app as the current plan in creating the model in TensorFlow and loading it in Android Studio does not seem to work
				85 hours	0 tasks remaining	

Purpose of tasks

For the third week of my project, I researched on solving the issue of detecting different accents. I have also begun implementation of the speech recognition screen with a dummy dataset of around 6 words in the attempt to test whether the plan I've set out works. Unfortunately, I've been having issues in getting the application to make predictions accurately and am currently assessing my initial plan and making possible changes to it.

Discussion with supervisor

I mentioned to my supervisor that I finished implementation for the login screen and create account screen, so I can use the great length of time in implementing the speech recognition screen. I also mentioned that I am having issues in getting the app to detect audio and having the app to make predictions but is something that I want to solve for myself by the end of reading week. Below are a few tasks we agreed on in which I can complete by next week.

List of tasks to complete for next week:

- Make a start on the preliminary report
- Continue implementation on the speech recognition and have some form of end product for next week so we can discuss on how I can improve on it

10.4 LOG 4 – 11/02/19 – 17/02/19

Task	Task name	Begin Date	Due date	Time spent	Completed?	Comments
1	Make a start on the preliminary report	11/02/19	15/02/19	35 hours	Yes	
2	Continue implementing the speech recognition screen	14/02/19	17/02/19	25 hours	Yes	I was able to fix the issue I had last week with converting audio into floating point. Now I have some form of a functioning app
				60 hours	0 tasks remaining	

Purpose of tasks

For the fourth week of my project I made a start on the preliminary report. I completed all the necessary sections and am currently restructuring certain parts so that it is in the form of a report by next week. I overcame the issue I had last week with the speech recognition screen, where I had an issue in converting the audio to a floating-point array. I was also able to further improve my TensorFlow model (e.g. fine-tuning hyperparameters etc.) in the attempt to improve its accuracy and avoid overfitting.

Discussion with supervisor

I showed my functioning app to my supervisor whom gave me great feedback for beginning implementation at an early stage and managing time exceptionally well. I then mentioned the remaining tasks I have to implement, which is to improve the TensorFlow model, make the app work in the background and take in voice recognition at the same time. Below are a few tasks we agreed on in which I can complete by next week.

List of tasks to complete for next week:

- Finish the preliminary report
- Finish implementation of the speech recognition screen
- Devise a plan for the testing phase, which we will enter in a few weeks' time

10.5 LOG 5 - 25/02/19 – 03/03/19

Task	Task name	Begin Date	Due date	Time spent	Completed?	Comments
1	Get the app to work in the background of the users phone. Ensure that it takes in speech as input	25/02/19	28/02/19	40 hours	Yes	
2	Improve on the TensorFlow model by adding more words for recognition	28/02/19	03/03/19	28 hours	Yes	I have successfully improved my model in recognising more words, but the accuracy is slightly low (e.g. ~80%.) I am currently trying to improve the model by optimizing hyperparameters and getting it to take more training steps
				68 hours	0 tasks remaining	

Purpose of tasks

For the fifth week of my project I attempted to get the app to work in the background of user's phone and at the same time, take speech as input. This task was something I thought was going to be difficult at the beginning of the project, but to my surprise it was fairly simple thanks to the classes provided by Android Studio. They made it much easier/quicker to work through this task. My initial plan was to somehow fit my TensorFlow model in the background to take in the speech. But this objective was something I could not implement. Instead I used the built-in Google speech recognition to take in speech as input in the background. This is probably a better solution in terms of power consumption because if my TensorFlow model were to operate in the background, then it would take a lot of battery power as opposed to the built-in Google speech recognition. Another task I worked on was to improve the TensorFlow model by adding more words for recognition. The details of this task can be seen in the comment's column above.

Discussion with supervisor

I couldn't meet with my supervisor this week as I had a job interview elsewhere. But based on the progress I have made, I think I am going to start on user testing and software testing for the whole of next week. At the same time, I will also make a start on the final report, as the deadline for this is looming.

10.6 LOG 6 – 04/03/19 – 10/03/19

Task	Task name	Begin Date	Due date	Time spent	Completed?	Comments
1	User testing	04/03/19	10/03/19	50 hours	Yes	Not enough data. Must continue with this task next week
2	Software testing	06/03/19	10/03/19	12 hours	Yes	Not enough data. Must continue with this task next week
3	Improve on the TensorFlow model by adding more words for recognition	07/03/19	09/03/19	5 hours	Yes	I have further improved my model from last week in recognising more words, but the accuracy can still improve(e.g. ~89%) if we train it for longer
				67 hours	0 tasks remaining	

Purpose of tasks

For the sixth week of my project, I started with user testing and software testing. For user testing, I aim to test the app with 210 random participants in the university or outside. So far, I have tested with 100 participants so will have to continue with this task next week. For user testing, I am getting people to test the app so I can gain data. For example, the participants I've tested with so far have experimented with the background speech recognizer in different scenarios (e.g. phone under the chair, blocked by a thick object etc.) For software testing, I aim to detect defects within my code and test the functionality/non-functionality of the app. I am currently doing this via functional, non-functional, dynamic and static testing.

Discussion with supervisor

In this week's discussion, I showed my supervisor another demo of the functioning app. I received positive feedback and some tips to enhance it. For example, he stated that I should stop the voice recognition background service from running when it doesn't detect any voices i.e. only activate when it hears a keyword (e.g. 'ok Google' etc.). This is so it can save the users battery power. Below are a few tasks we agreed on in which I can complete by next week.

- Continue with user testing and consider adding more interesting experiments that can output relevant/useful data (e.g. testing the background speech recognizer from specific distances 2 metres, 3 metres, 4 metres….)
- Continue with software testing
- Continue training the TensorFlow model, so its detection rate is at a high level

10.7 LOG 7 - 11/03/19 – 17/03/19

Task	Task name	Begin Date	Due date	Time spent	Completed?	Comments
1	Finish off with user testing	11/03/19	15/03/19	40 hours	Yes	Enough data collected for analysis
2	Finish off with software testing	14/03/19	16/03/19	20 hours	No	Still need to work on static and dynamic tests
3	Make a start on the final report	15/03/19	17/03/19	30 hours	Yes	
4	Continue improving the TensorFlow model	11/03/19	16/03/19	10 hours	Yes	I have improved my model from last week in recognising more words. It is now at ~91% accuracy, but I intend to further improve it again next week
				100 hours	1 task remaining	

Purpose of tasks

For the seventh week of my project, I finished user testing and have gathered the data for analysis, which will help further improve the app. As for software testing, I was not able to finish the static and dynamic tests as I was prioritising task 3. So, task 2 has been delayed for next week. Since week 5, I have been improving the test accuracy of the TensorFlow model, so it can get better at making predictions. Currently, the accuracy is at an appropriate level, but I intend to further enhance it one more time next week, so I can guarantee that the model cannot be improved.

Discussion with supervisor

In this week's discussion, I mentioned continuing with user testing and software testing. My supervisor specified that I should make a start on the draft version of the final report, so I can receive accurate feedback. Hence, why I have started on it this week rather than next week. Below are a few other tasks we agreed on in which I can complete by next week.

- Gather user testing data and visualise it in charts, graphs etc.
- Finish off with software testing
- Build on the work done on the final report
- Train the TensorFlow model one last time, so its detection rate is at a high level

10.8 LOG 8 - 18/03/19 – 24/03/19

Task	Task name	Begin Date	Due date	Time spent	Completed?	Comments
1	Continue with the final report	18/03/19	20/03/19	30 hours	Yes	
2	Finish off with software testing	21/03/19	24/03/19	20 hours	Yes	
4	Continue improving the TensorFlow model	23/03/19	23/03/19	15 hours	Yes	Model has not improved any more than 91% testing accuracy
				65 hours	0 tasks remaining	

Purpose of tasks

For the eighth week of my project, I finished off with software testing and have finally gathered all test data needed to improve the app in the coming weeks. I've also made progress with the final report by filling in most sections to receive an accurate feedback from my supervisor. I've continued to improve the TensorFlow model, but it seems as though the testing accuracy will not go over 91%, so this task will no longer be needed to be prioritised in the future.

Discussion with supervisor

In this week's discussion, I mentioned that I've continued with the final report and have accumulated all testing data, which I may add to the report for better feedback. My supervisor insisted that I should prioritise in finalising the final draft report by next week

10.9 LOG 9 - 25/03/19 – 31/03/19

Task	Task name	Begin Date	Due date	Time spent	Completed?	Comments
1	Finalise the final draft report	25/03/19	31/03/19	90 hours	Yes	
				90 hours	0 tasks remaining	

Purpose of tasks

For the ninth week of my project, I made it a priority to complete the final report draft. I spent a lot of time in doing this, as I think it's important to get a good feedback before submitting in May. This feedback should help me to improve certain areas and most importantly strengthen the introduction and conclusion of the report.

Discussion with supervisor

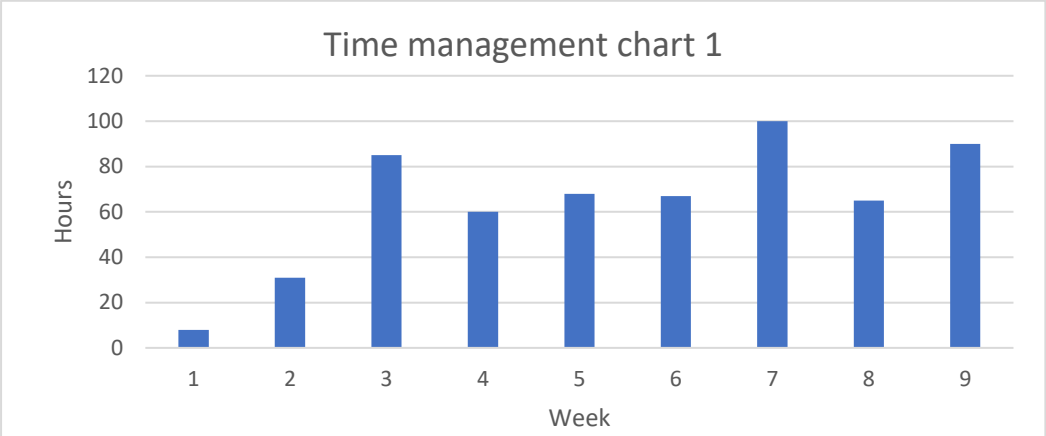
No meeting took place this week, however there are still a lot of tasks that I need to do in the month of April. For example, I need to use the testing data to improve the app, battery consumption of the app can still be reduced if I use an alternative background service, a recovery database (e.g. SQLite) should be added in case of emergency etc.

10.10 LOG 10 - Time management report

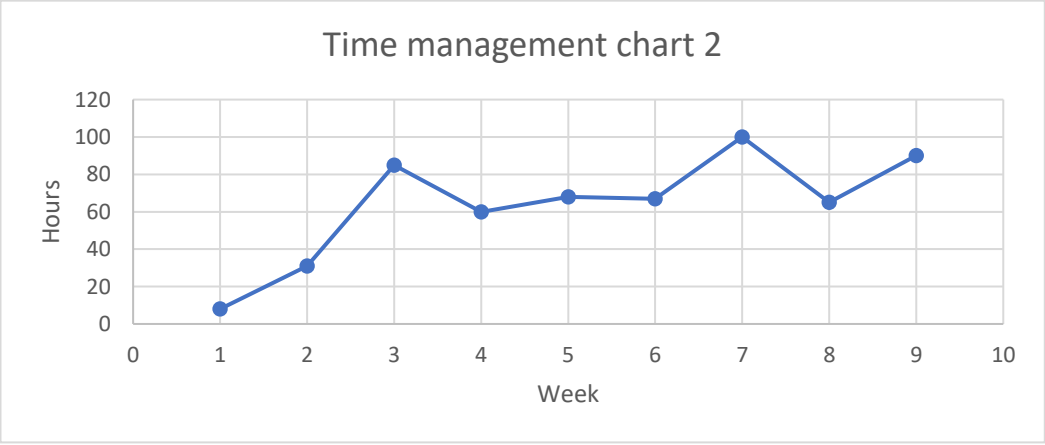
a. Table

Week	Hours
1	8
2	31
3	85
4	60
5	68
6	67
7	100
8	65
9	90
Total	574

b. Chart 1

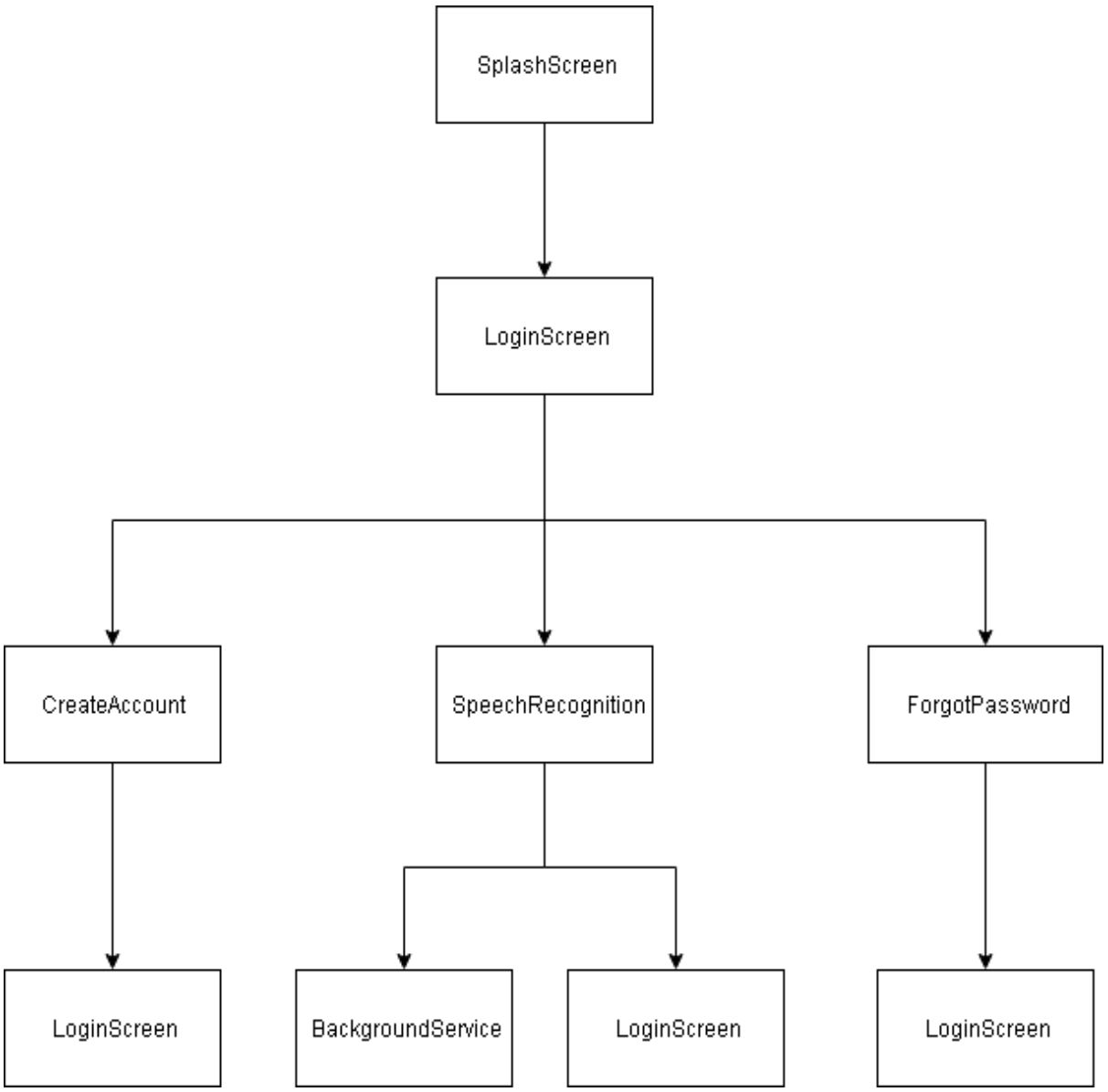


c. Chart 2

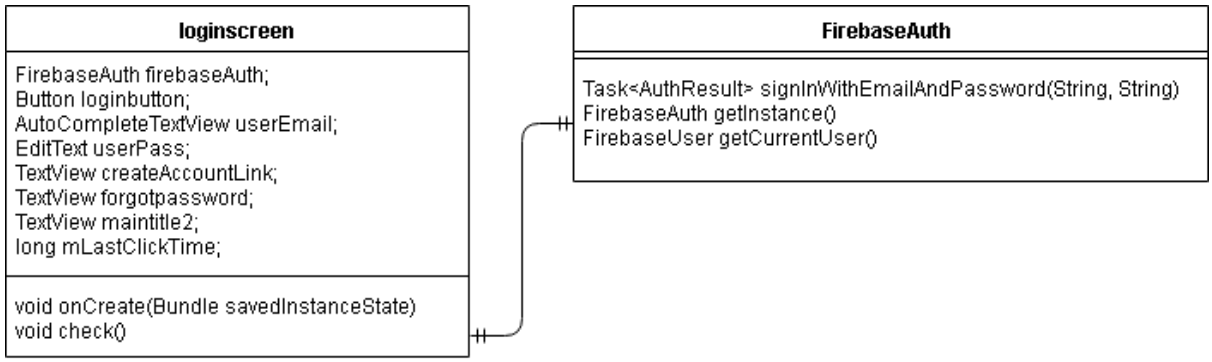


Appendix C Designs

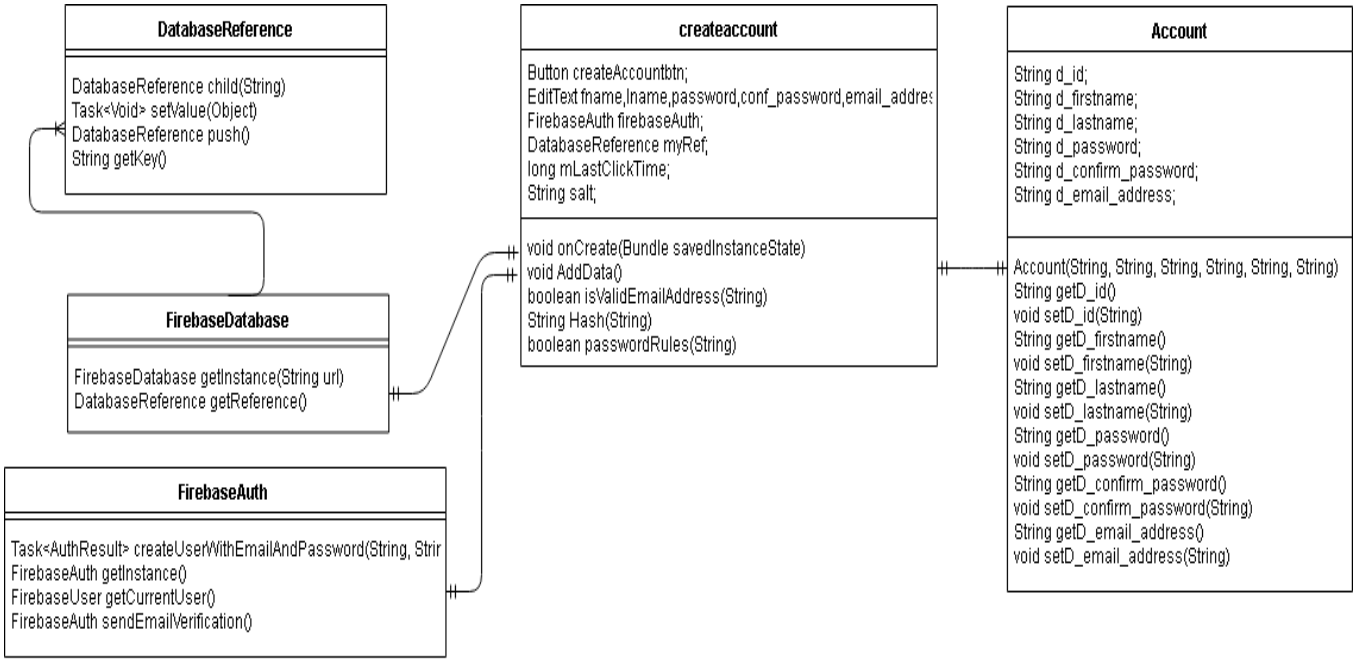
10.11 Hierarchy diagram



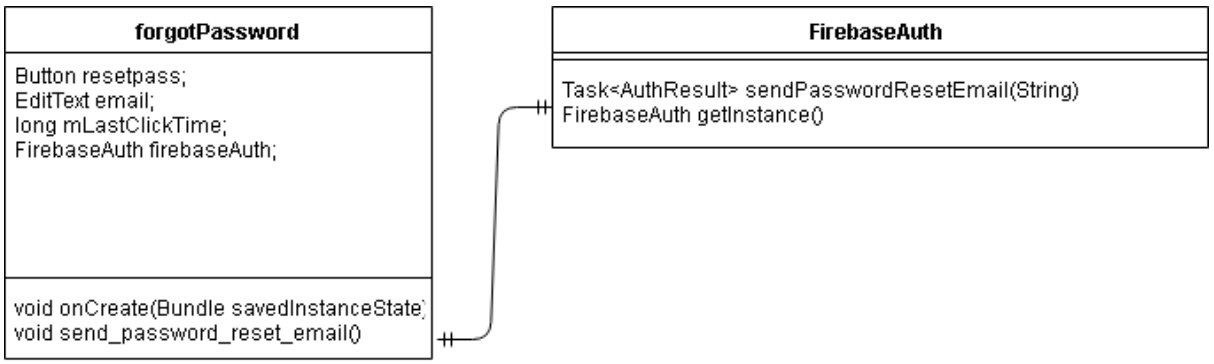
10.12 loginscreen class diagram



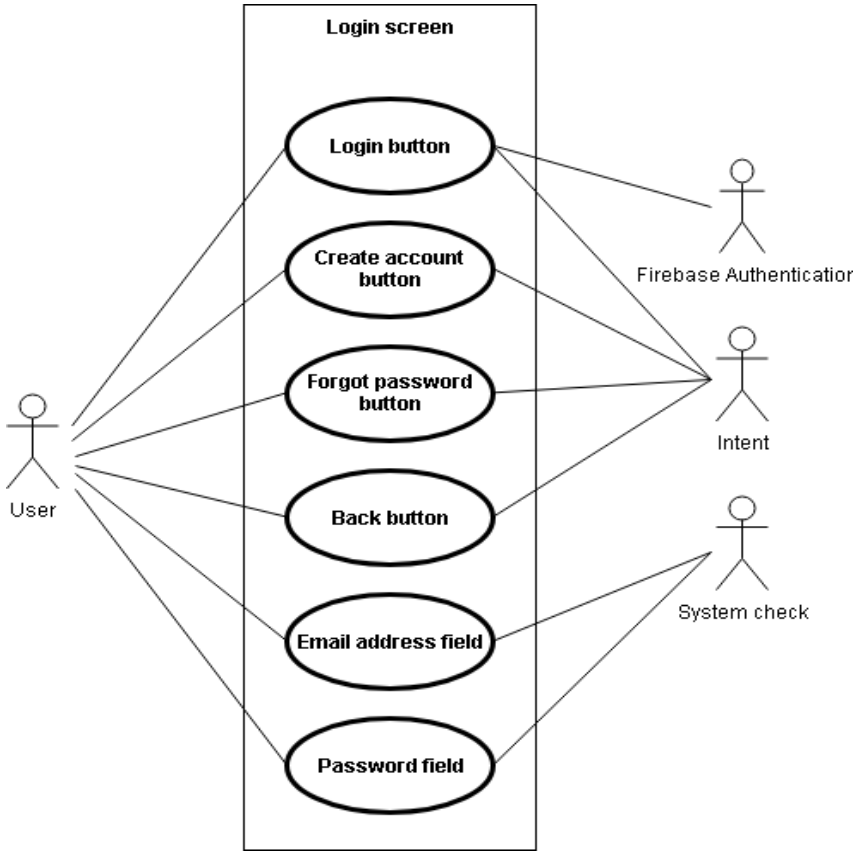
10.13 createaccount class diagram



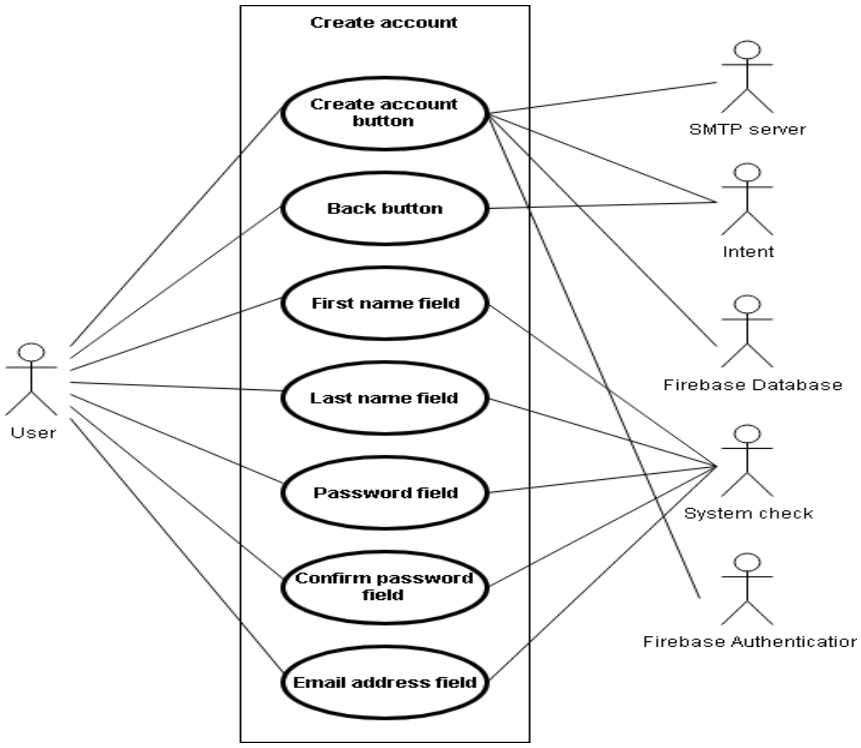
10.14 forgotPassword class diagram



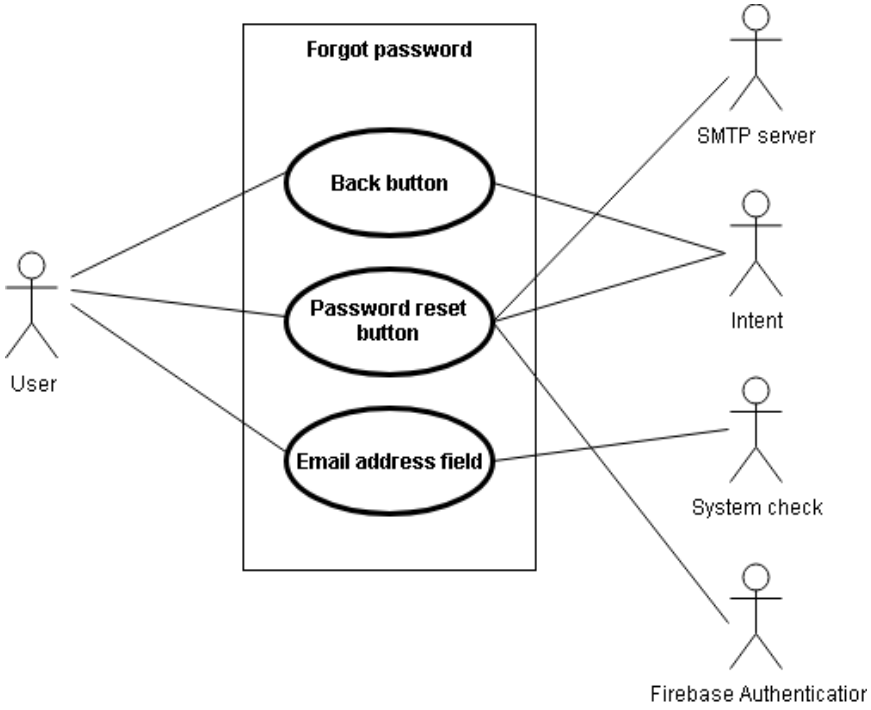
10.15 use-case diagram for loginscreen class



10.16 use-case diagram for createaccount class



10.17 use-case diagram for forgot password class



10.18 basic pseudocode for loginscreen class

```
onCreate()
 IF isClicked(login_btn) THEN
  IF isVerified(email_address.gettext(),password.gettext()) == true AND isCorrectEmailAndPassword(email_address.gettext(),password.gettext()) == true THEN
   startActivity(speechrecognition)
  ELSE
   print("ERROR! please try again")

 IF isClicked(back_btn) == true THEN
  startActivity(loginscreen)

 IF isClicked(forgotpassword_btn) == true THEN
  startActivity(forgotpassword)

 IF isClicked(registeraccount_btn) == true THEN
  startActivity(createaccount)
```

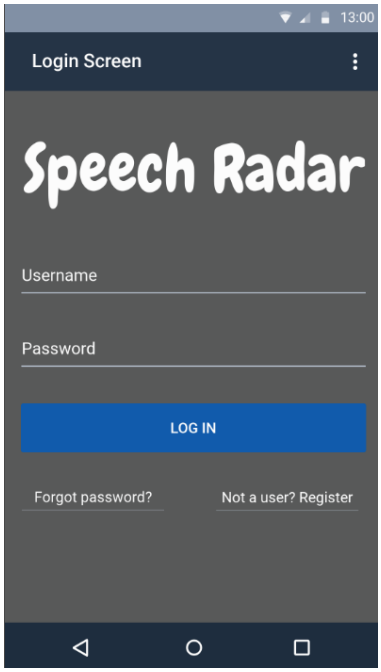
10.19 basic pseudocode for createaccount class

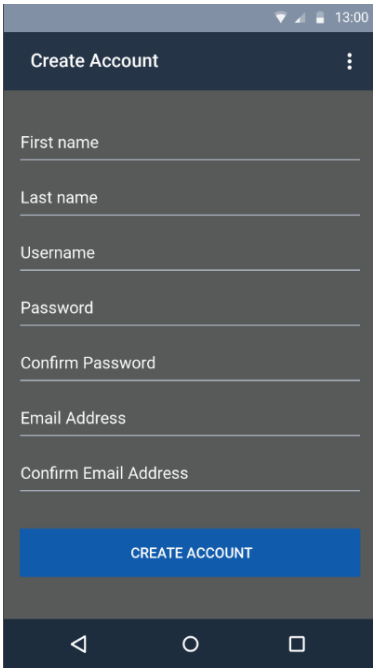
```
onCreate()
 IF isClicked(createaccount_btn) == true THEN
  IF passwordLength(password.gettext(),confirmpassword.gettext()) == true AND isValidEmailAddress(emailaddress.gettext()) == true AND length(firstname,lastname) > 2 THEN
   AddData()
   sendEmailVerification()
  ELSE
   print("ERROR! please try again")

 IF isClicked(back_btn) == true THEN
  startActivity(loginscreen)

AddData()
 DB.child(val).setValue(firstname)
 DB.child(val).setValue(lastname)
 DB.child(val).setValue(password)
 DB.child(val).setValue(confirmpassword)
 DB.child(val).setValue(emailaddress)
```

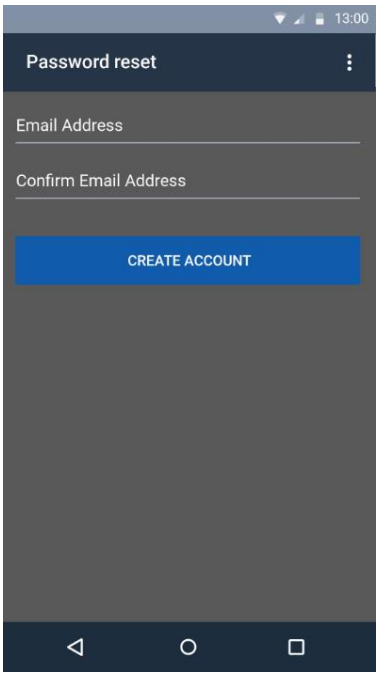
10.20 basic pseudocode for forgotPassword class

```
onCreate()
  IF isClicked(passwordreset_btn) == true THEN
    IF isVerified(email_address.gettext()) == true THEN
      sendPasswordResetEmail()
  ELSE
    print("ERROR! please try again")

  IF isClicked(back_btn)== true THEN
    startActivity(loginscreen)
```

10.21 Remaining wireframes

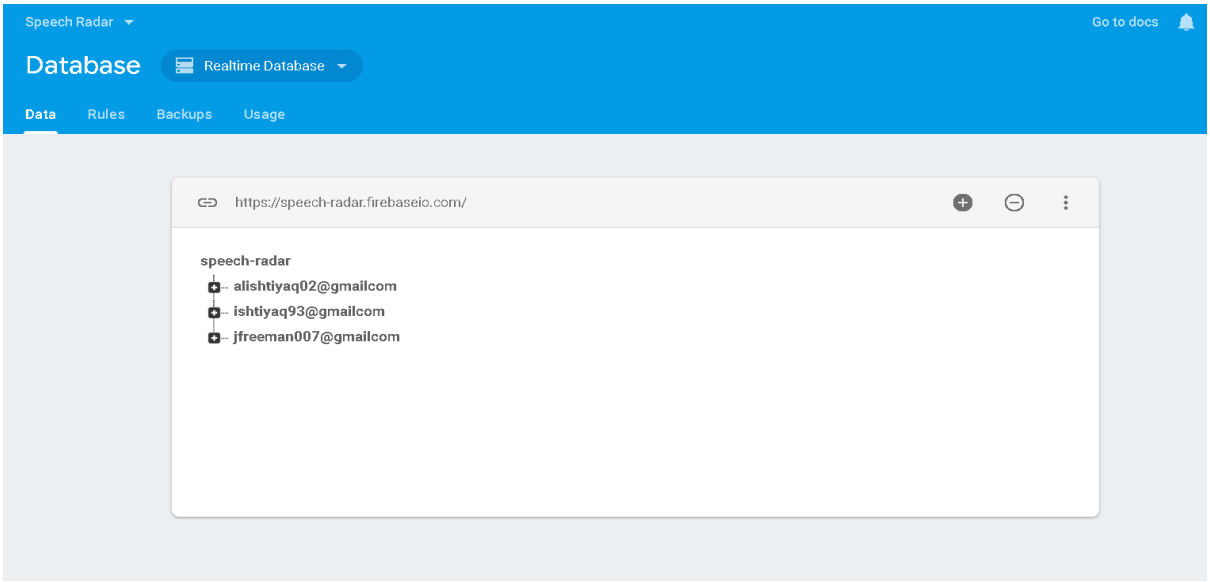




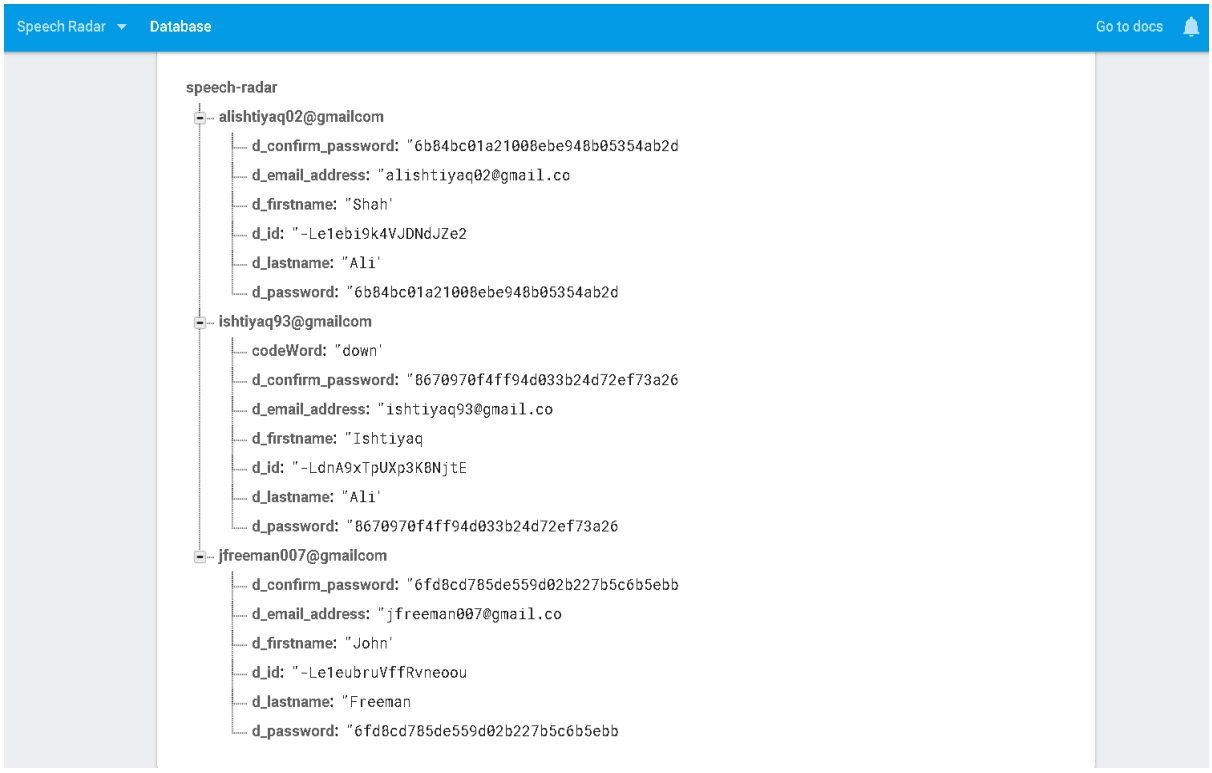


Appendix D Firebase Database

10.22 all parent nodes in Firebase Database



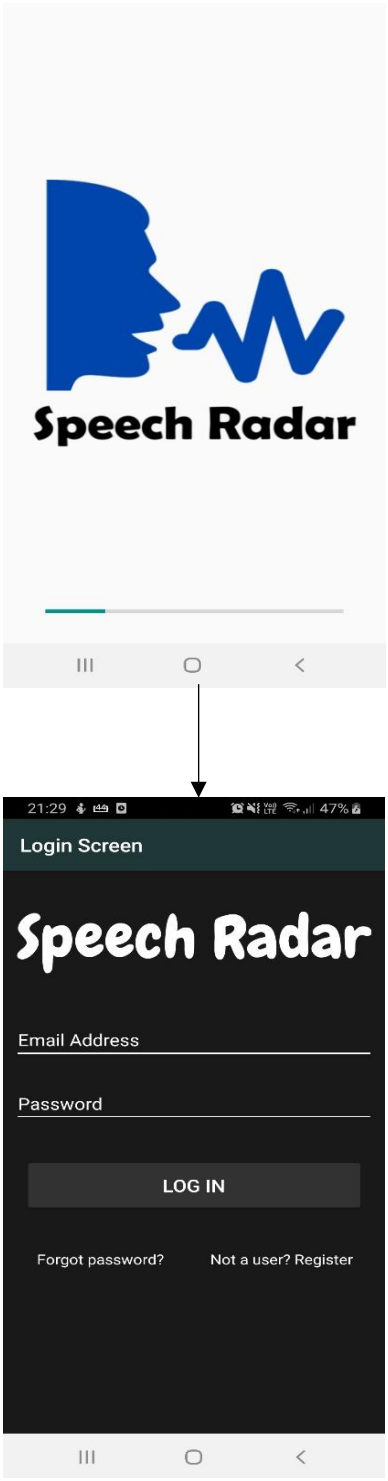
10.23 all children of parent nodes in Firebase Database



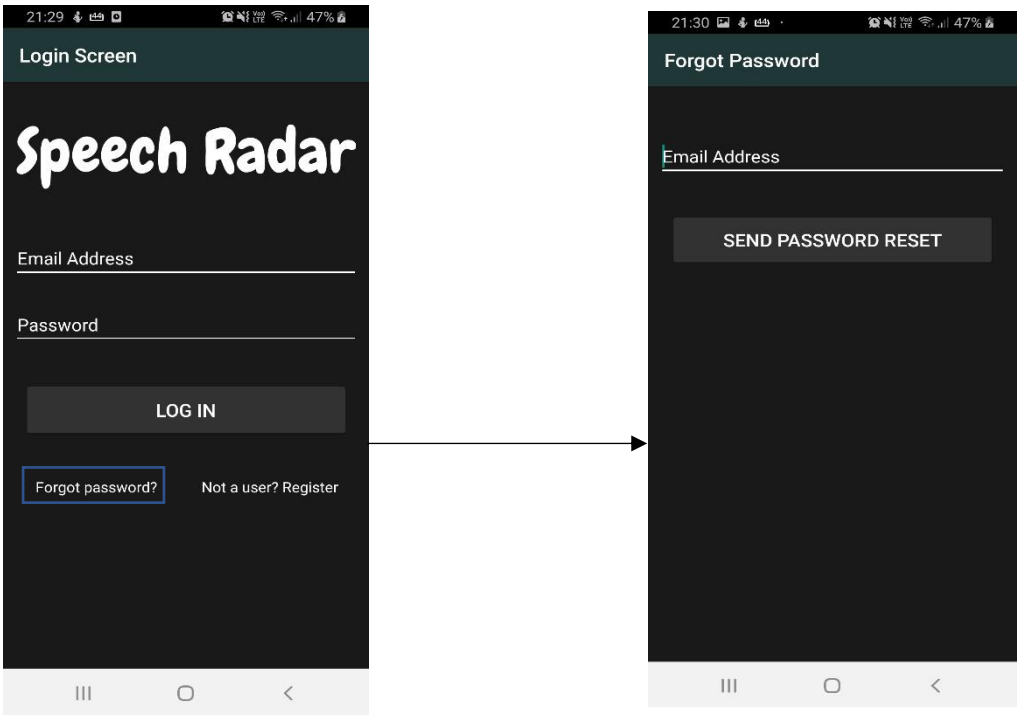
Appendix E Testing

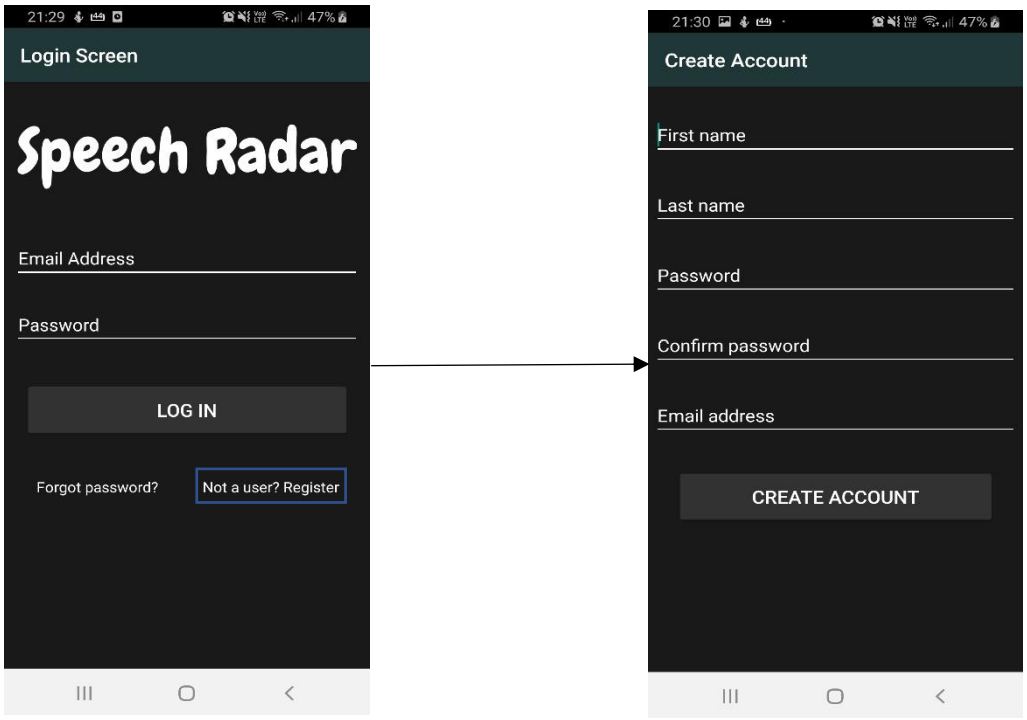
10.24 Functional testing

10.24.1 Test 1

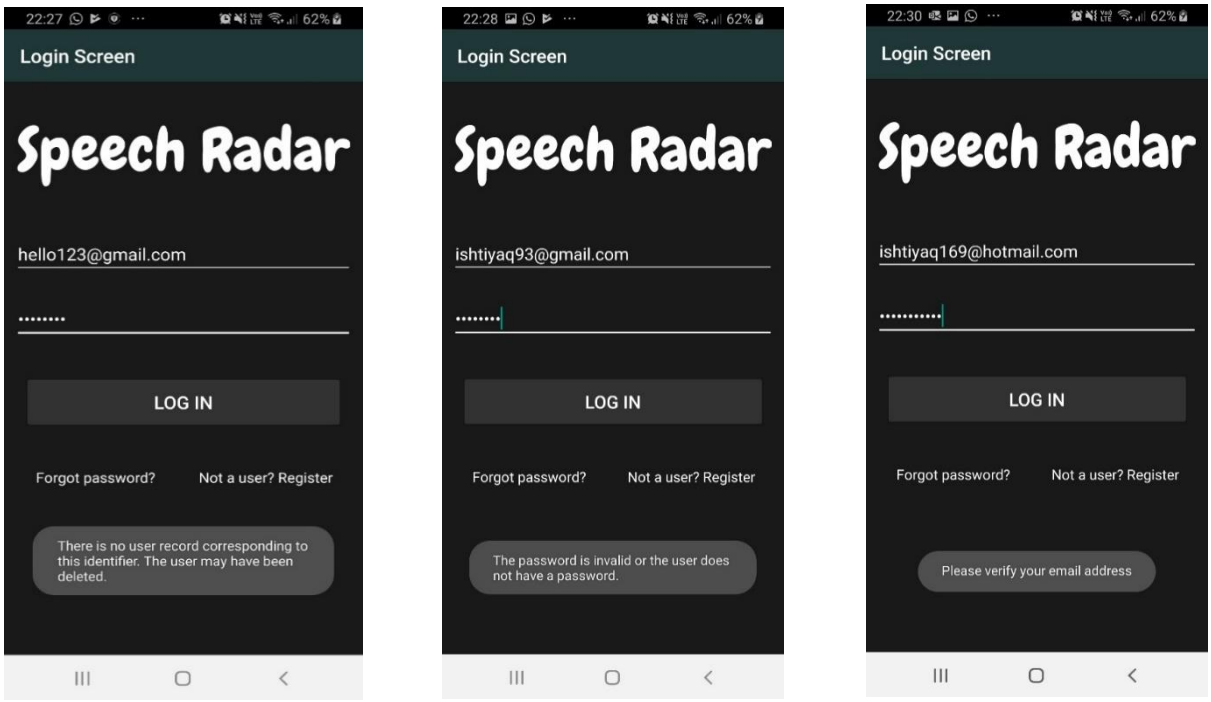


10.24.2 Test 2


21:29
47%
Login Screen
Speech Radar
Email Address
Password
LOG IN
Forgot password?
Not a user? Register
21:30
47%
Forgot Password
Email Address
SEND PASSWORD RESET

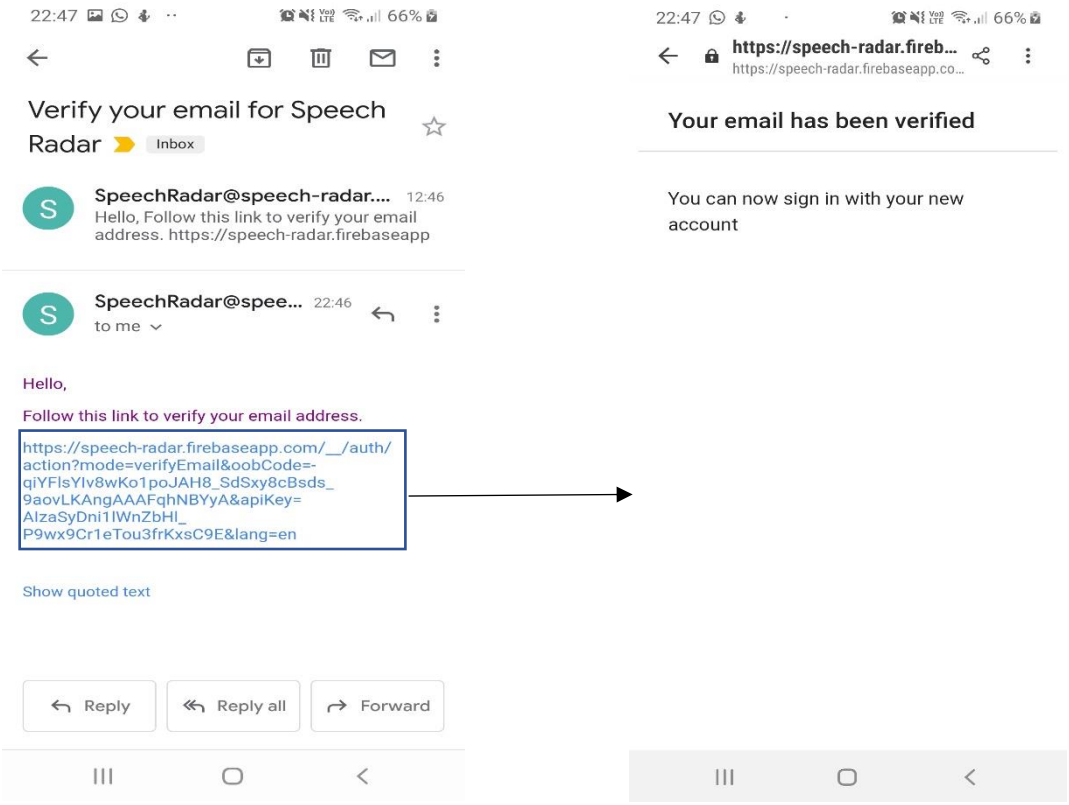

21:29
47%
Login Screen
Speech Radar
Email Address
Password
LOG IN
Forgot password?
Not a user? Register
21:30
47%
Create Account
First name
Last name
Password
Confirm password
Email address
CREATE ACCOUNT

10.24.3 Test 3

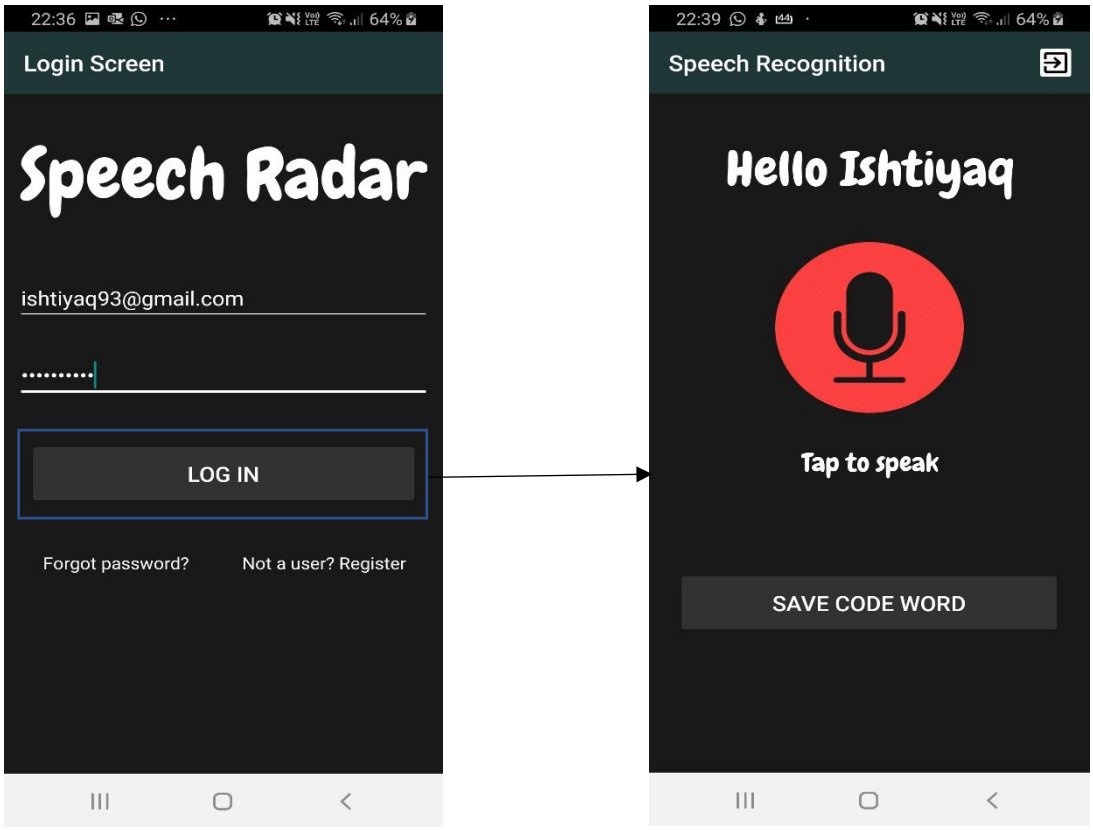


10.24.4 Test 4

10.24.4.1 Process:

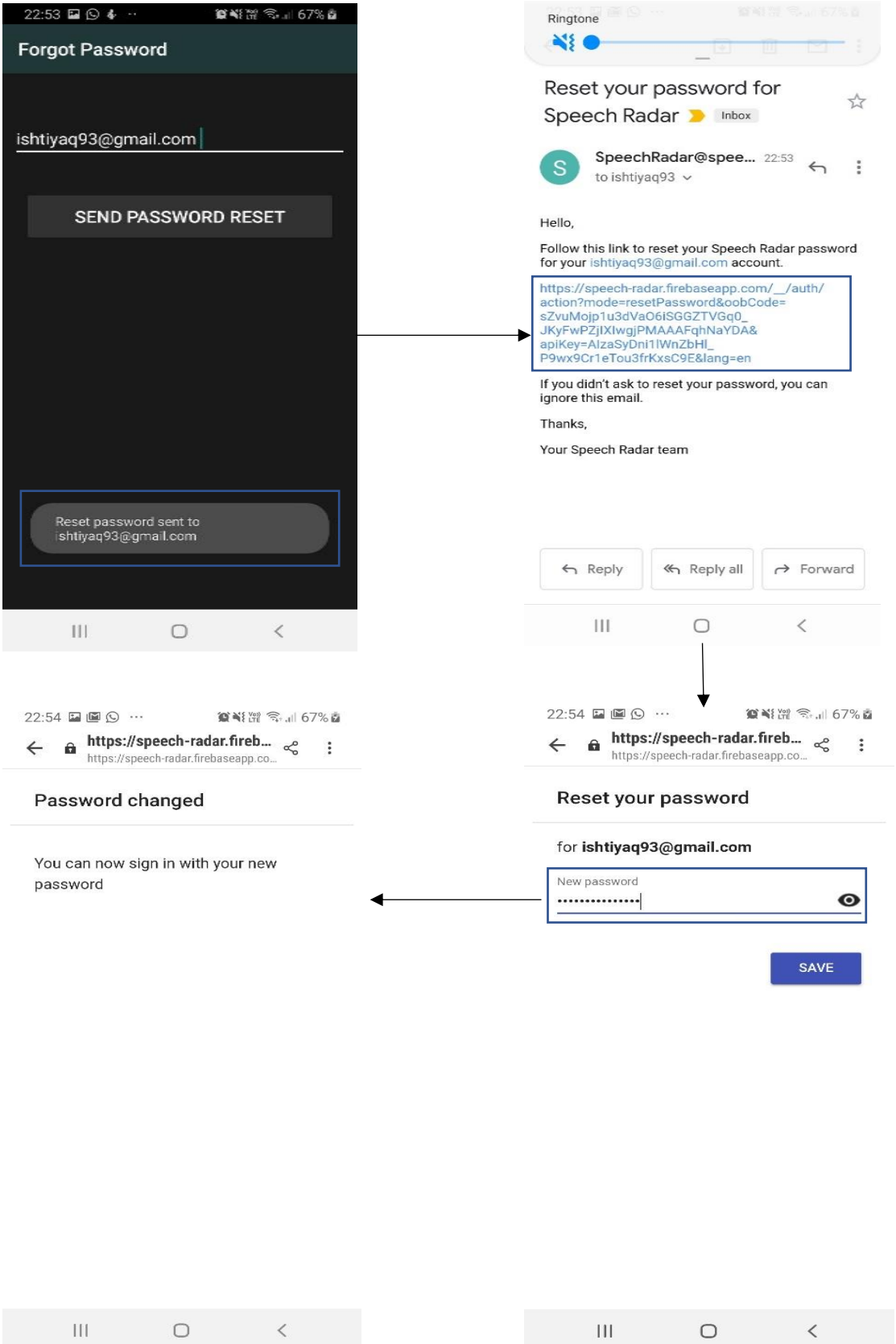


10.24.4.2 Result:

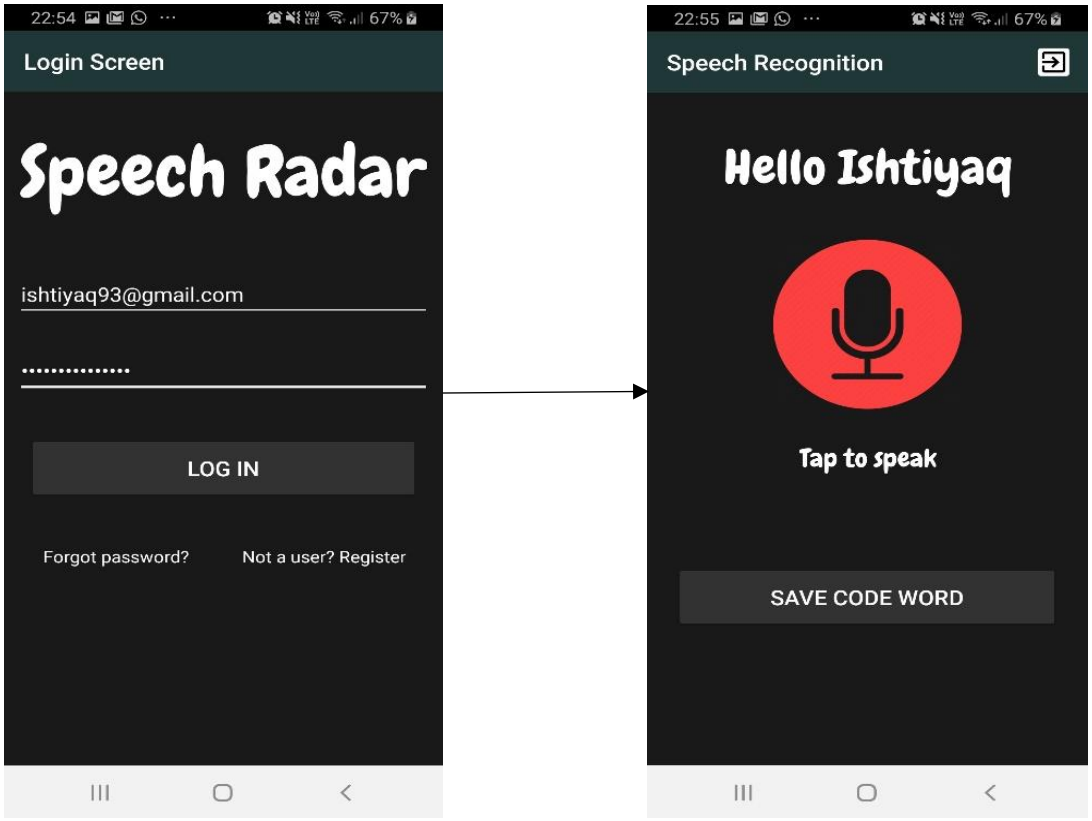


10.24.5 Test 5

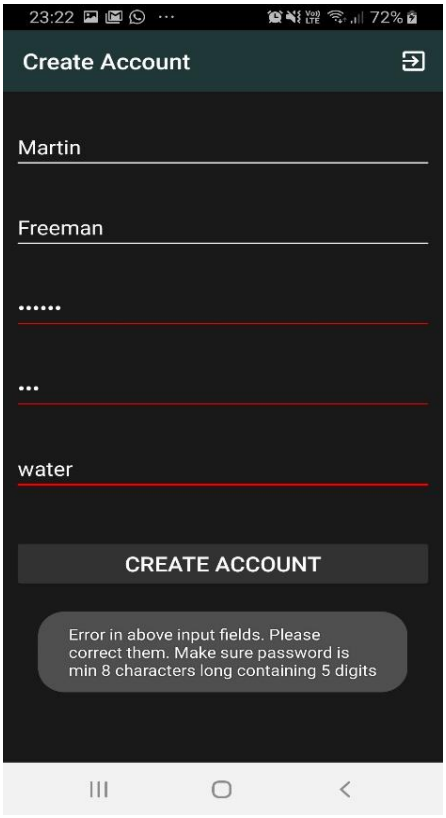
10.24.5.1 Process:



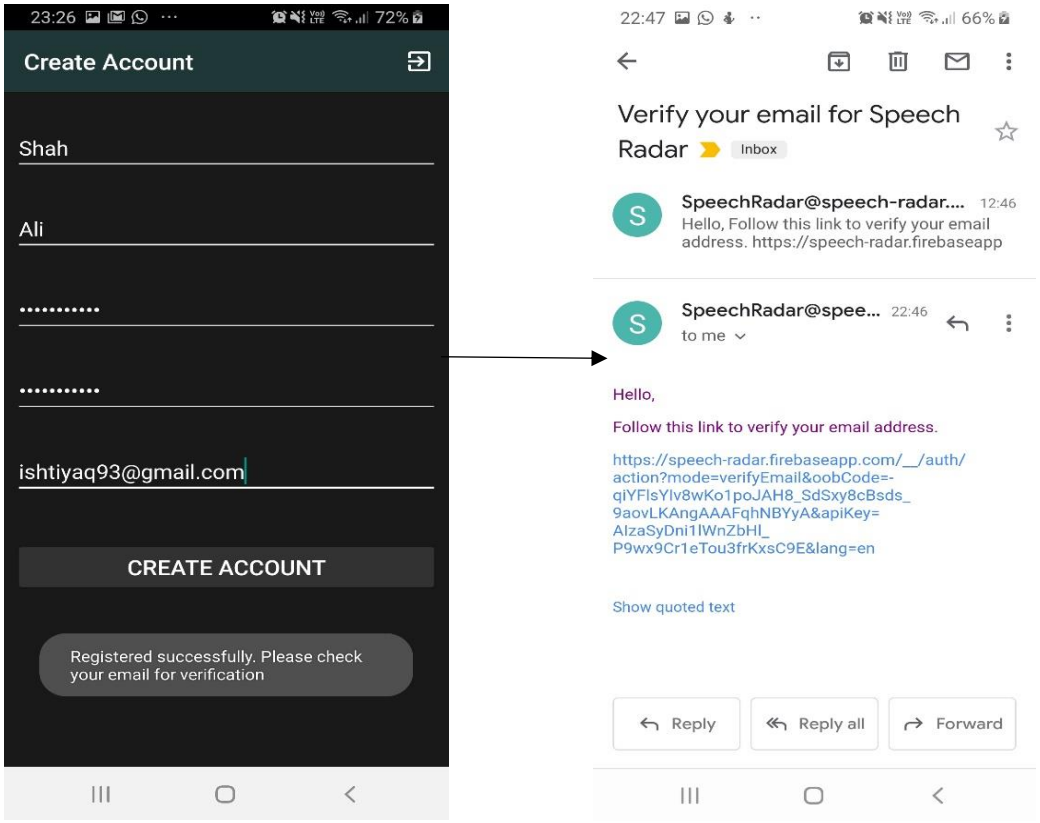
10.24.5.2 Result:



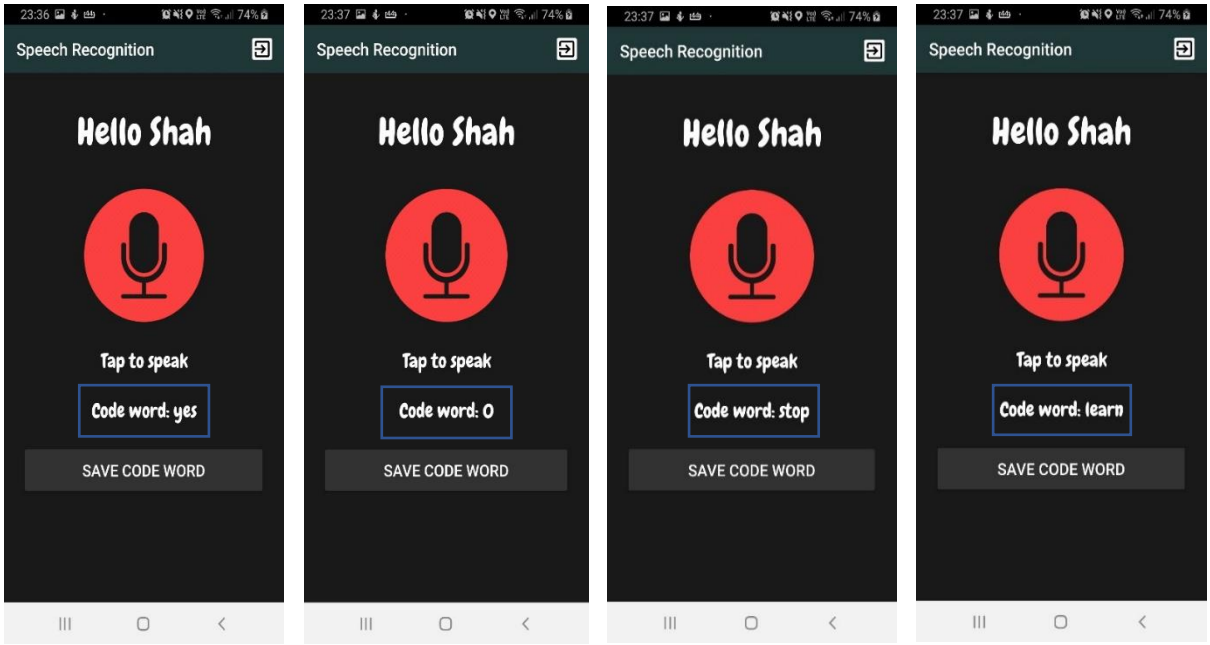
10.24.6 Test 6



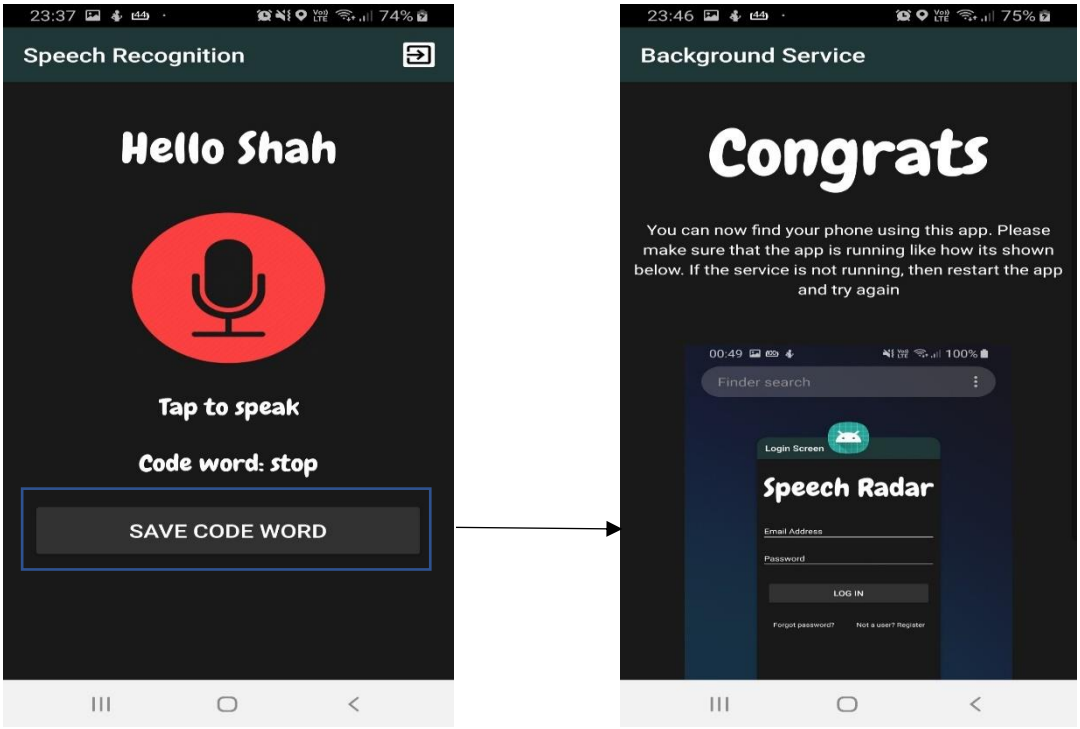
10.24.7 Test 7



10.24.8 Test 8

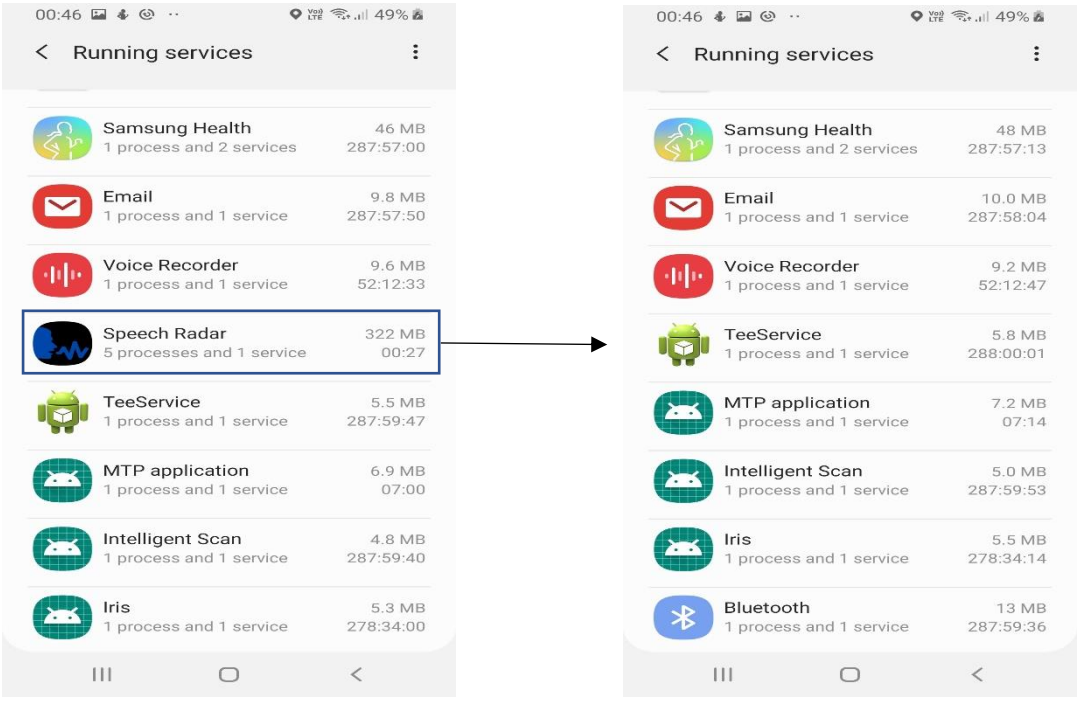


10.24.9 Test 9



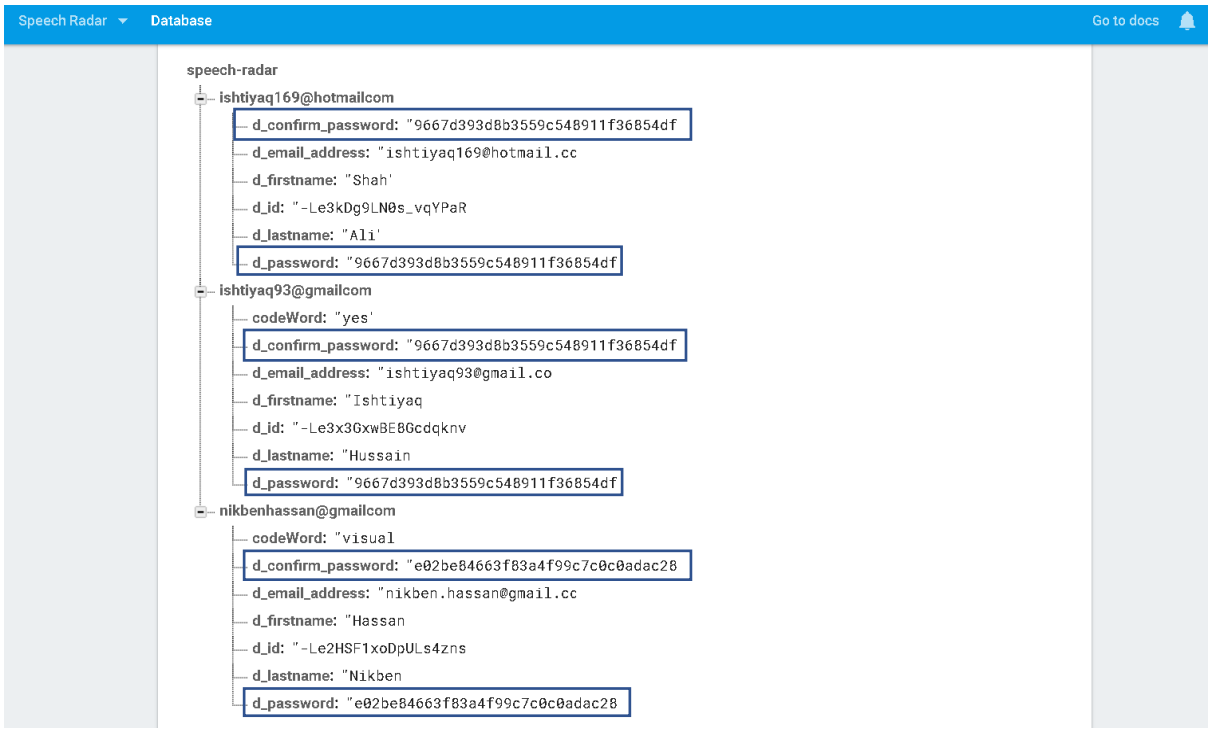
10.24.10 Test 10

10.24.10.1 Before and after

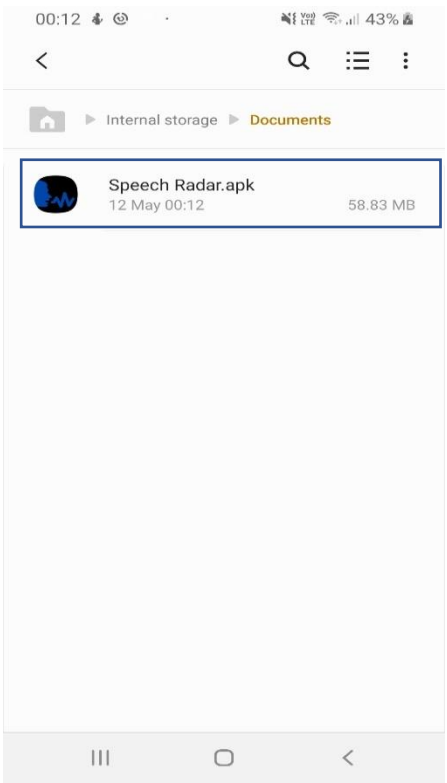


10.25 Non-functional testing

10.25.1 Test 1



10.25.2 Test 2



10.25.3 Test 3

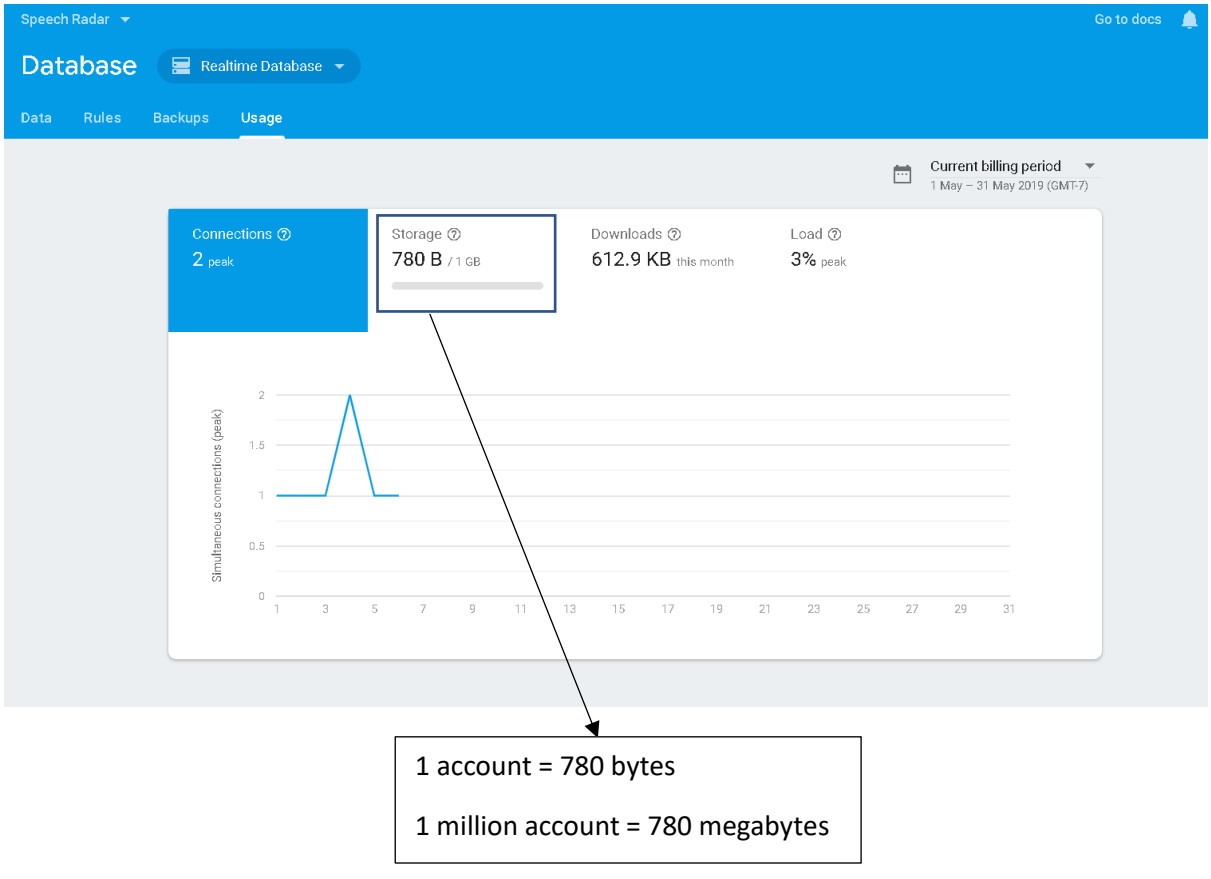
10.25.3.1 Correction – proof of existing SQLite recovery database:

ID	FIRSTNAME	LASTNAME	PASSWORD	CONFIRM_PASSWORD	EMAIL_ADDRESS	CODEWORD
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Shah	Ali	9667d393d8b3559c5489...	9667d393d8b3559c5489...	ishtiyaq169@hotmail.com	*NULL*
2	Ishtiyaq	Hussain	9667d393d8b3559c5489...	9667d393d8b3559c5489...	ishtiyaq93@gmail.com	yes
3	Hassan	Nikben	e02be84663f83a4f99c7c...	e02be84663f83a4f99c7c...	nikben.hassan@gmail.com	visual

10.25.4 Test 4

Device	Company	Model	API level	Code name	Compatible?
1	Samsung	Galaxy S9	28	Pie	yes
2	Huawei	P20 Lite	26	Oreo	yes
3	Samsung	Tab S SM-T705	24	Nougat	yes
4	Motorola	E5 Play	22	Lollipop	yes
5	Samsung	Galaxy S8	28	Pie	yes
6	Google	Pixel 2	28	Pie	yes
7	Google	Nexus 4	24	Nougat	yes
8	Google	Nexus 5	26	Oreo	yes
9	htc	One M8	20	KitKat	yes
10	Samsung	Galaxy J3	23	Marshmallow	yes

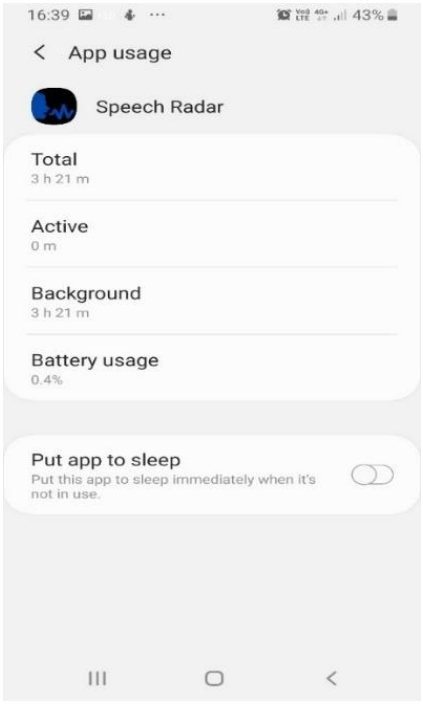
10.25.5 Test 5



10.25.6 Test 6

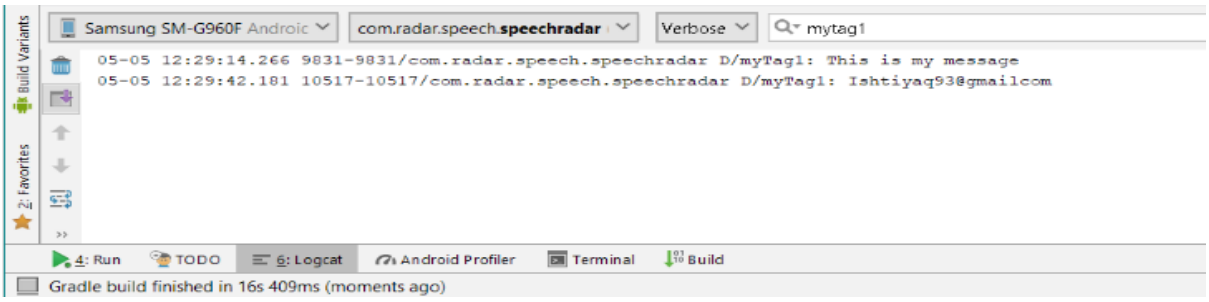
	Device 1	Device 2	Device 3	Device 4	Device 5	Device 6	Device 7	Device 8	Device 9	Device 10	**Device Average**
Load login screen (seconds)	0.5	0.8	0.5	0.8	0.6	0.7	1.1	1	0.8	0.5	**0.73**
Load create account screen (seconds)	0.5	0.6	0.5	0.5	1.3	0.7	1	0.7	0.9	0.7	**0.74**
Load forgot password screen (seconds)	0.8	0.6	0.5	0.5	0.5	0.8	0.8	0.6	0.7	0.9	**0.67**
Load speech recognition screen (seconds)	1.8	1.6	1.5	0.9	1.5	1.7	1.6	1.9	2.3	1.2	**1.6**
Load background service screen (seconds)	1	0.9	1	0.8	1.3	1.5	1.5	1.5	1.2	1.4	**1.21**
Load background service screen (seconds)	2	1	1.2	1.5	3	1.1	2	3.3	1	1.7	**1.78**
Load background service screen (seconds)	1.3	1.7	1.2	1	1.7	1	1.5	1.3	1.8	2.1	**1.46**
Load background service screen (seconds)	**1.13**	**1.03**	**0.91**	**0.86**	**1.41**	**1.07**	**1.36**	**1.47**	**1.24**	**1.21**	**1.17**

10.25.7 Test 7



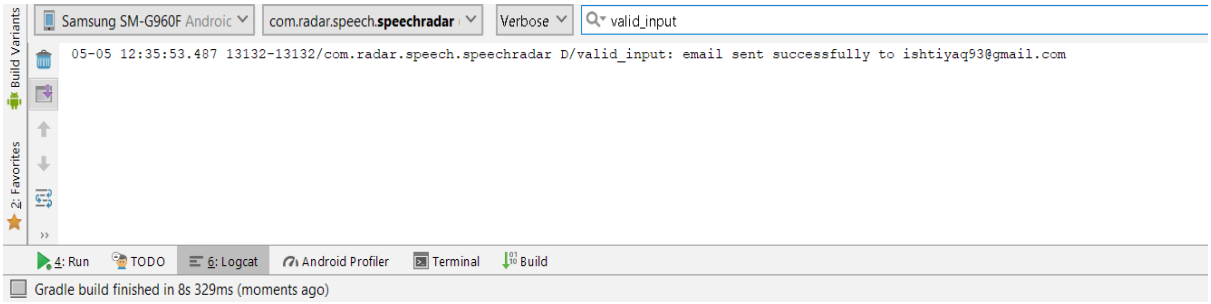
10.26 Static and dynamic testing

10.26.1 Test 2

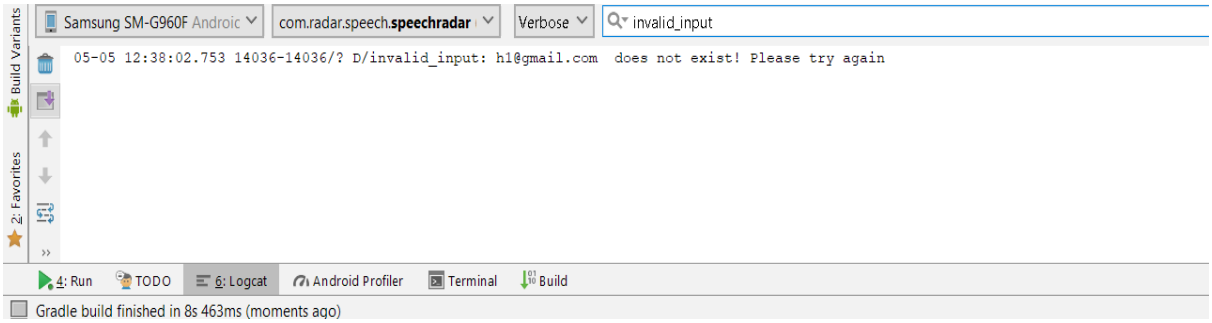


10.26.2 Test 4

10.26.2.1 Valid email:

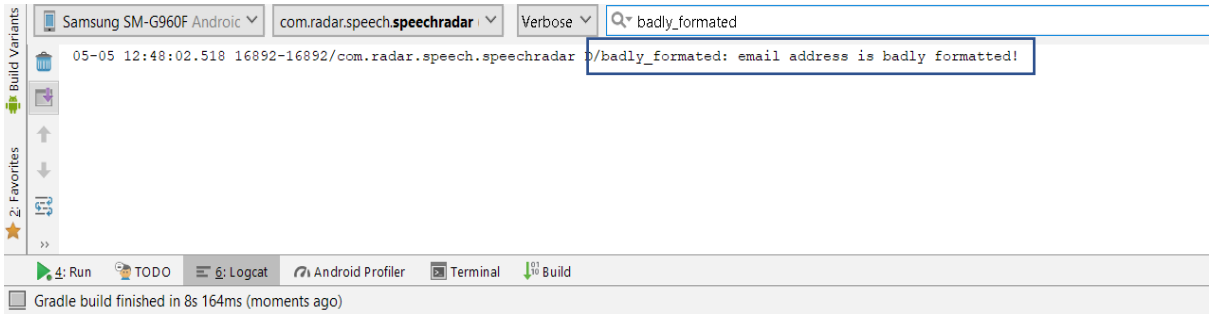


10.26.2.2 Invalid email:

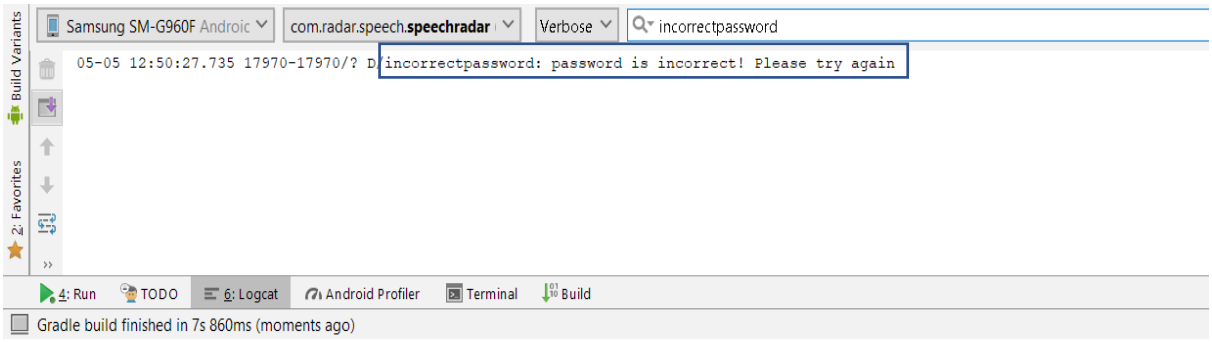


10.26.3 Test 6

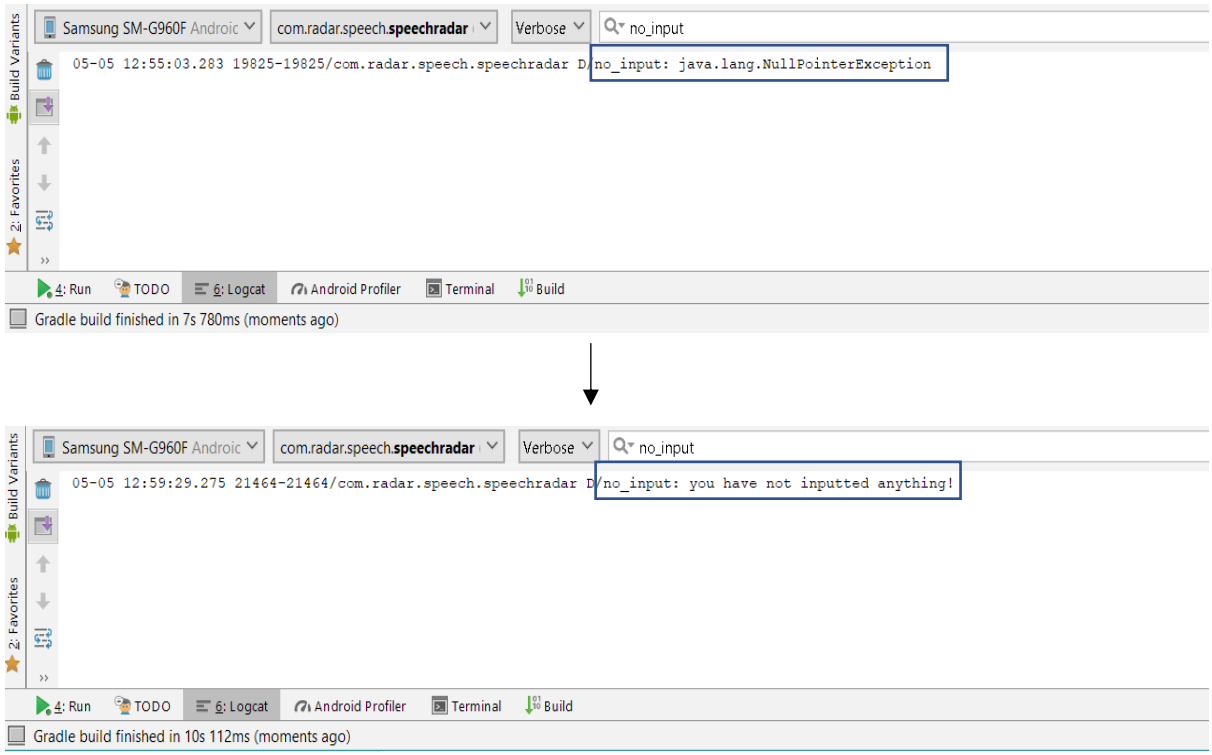
10.26.3.1 Badly formatted email address:



10.26.3.2 Incorrect password:

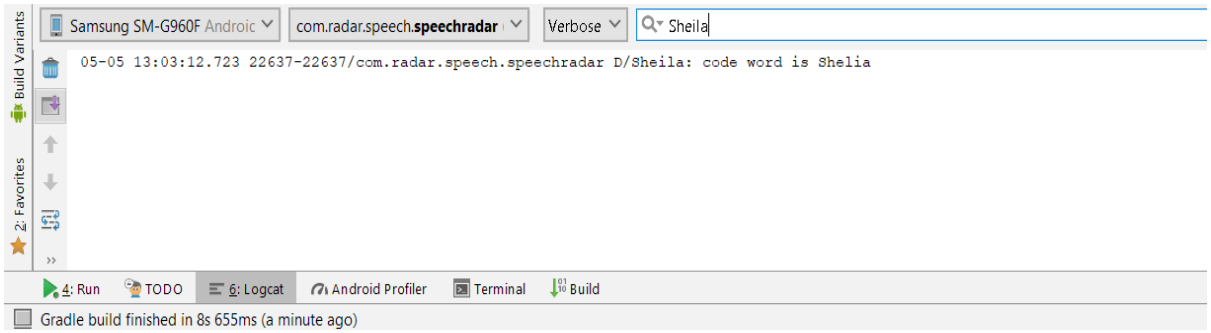


10.26.3.3 Correction made via try-catch:

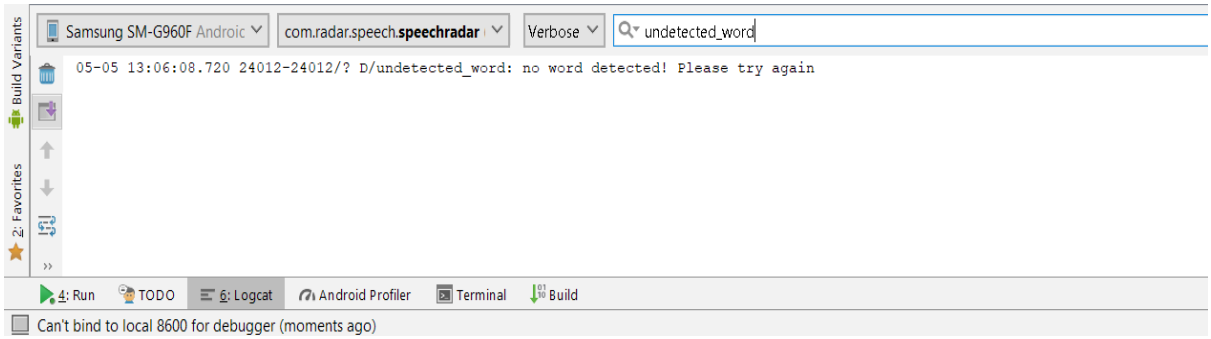


10.26.4 Test 8

10.26.4.1 Detected code word:



10.26.4.2 Undetected code word:



10.26.5 Test 9

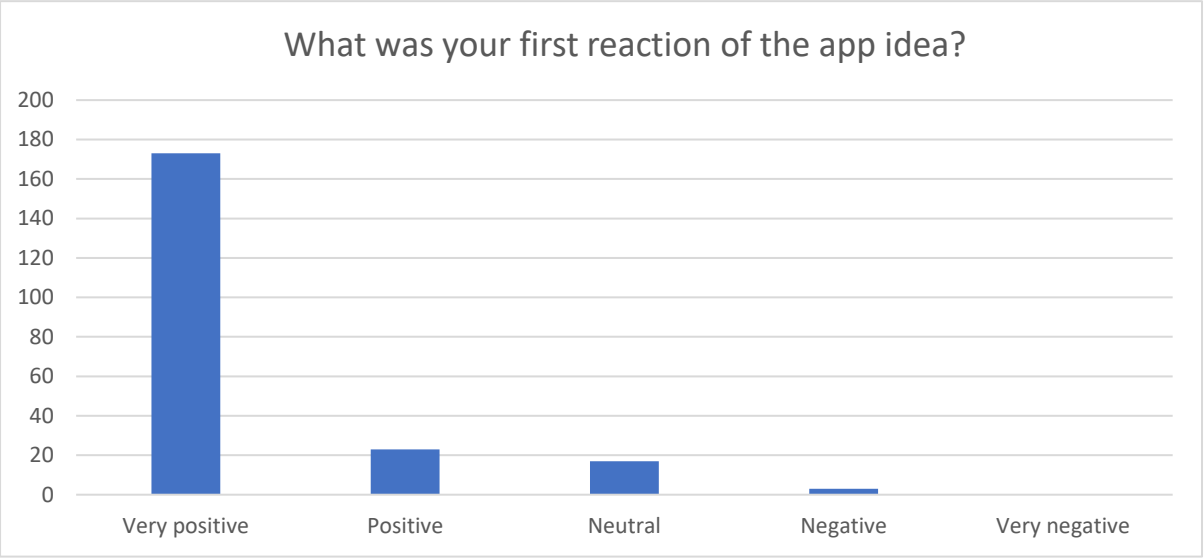
```
import android.app.ActivityManager;
import android.Manifest.permission;
import android.app.AlarmManager;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.media.AudioManager;
import android.media.Ringtone;
import android.provider.MediaStore;
import android.graphics.BitmapFactory;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Build.VERSION;
import android.os.Build.VERSION_CODES;
import android.os.CountDownTimer;
import android.os.IBinder;
import android.text.TextUtils;
import android.util.Log;
import android.widget.Toast;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.TextView;
import com.afollestad.materialdialogs.MaterialDialog;
import com.afollestad.materialdialogs.Theme;
```



```
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.TextView;

import com.afollestad.materialdialogs.MaterialDialog;
import com.afollestad.materialdialogs.Theme;
```

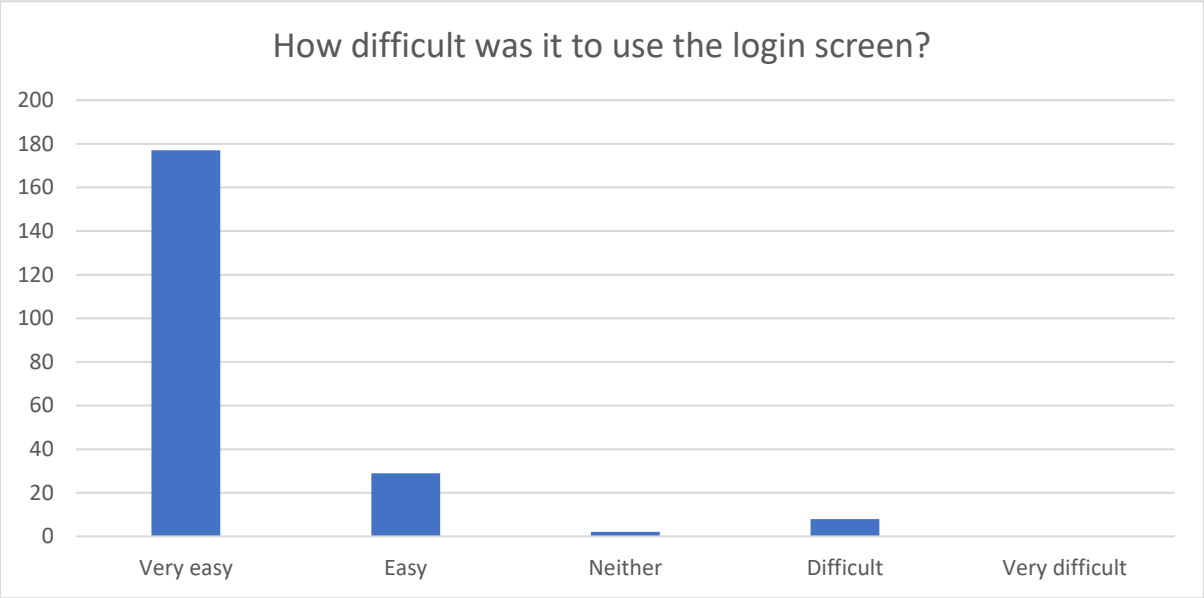
10.27 User testing

10.27.1 User reaction

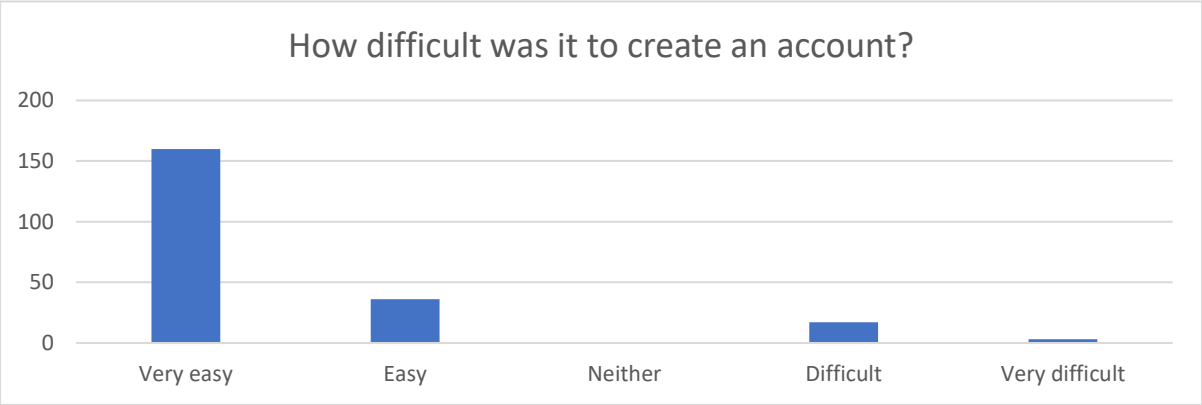


10.27.2 User interface

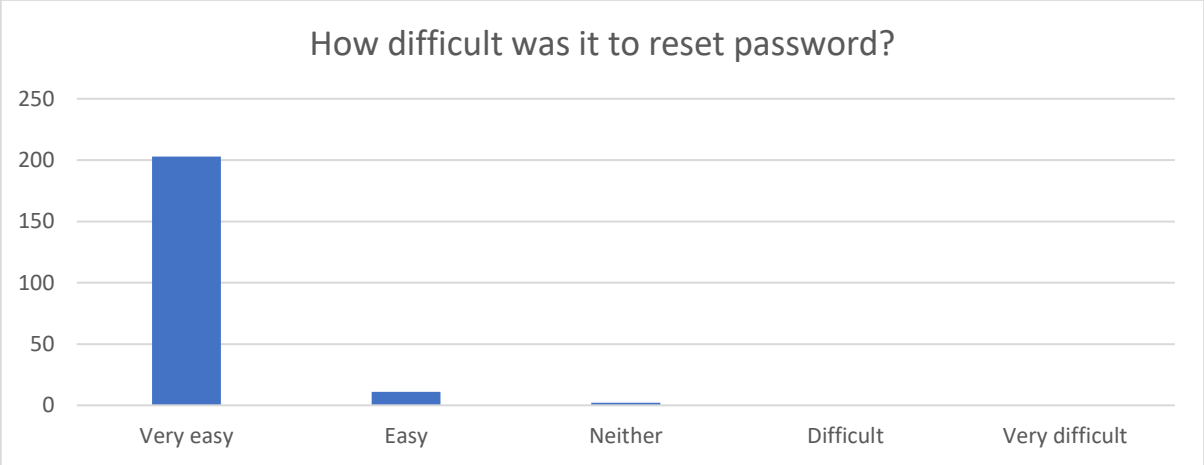
10.27.2.1 Login screen:



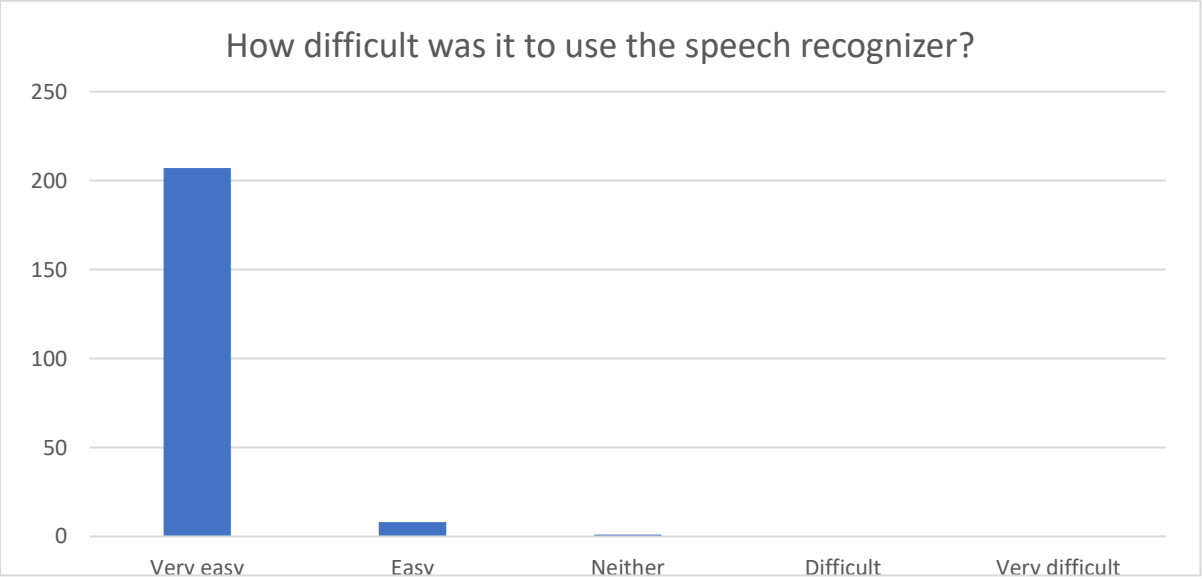
10.27.2.2 Create account:



10.27.2.3 Reset password:



10.27.2.4 Speech recognizer:



i.

10.27.2.5 Possible improvements:

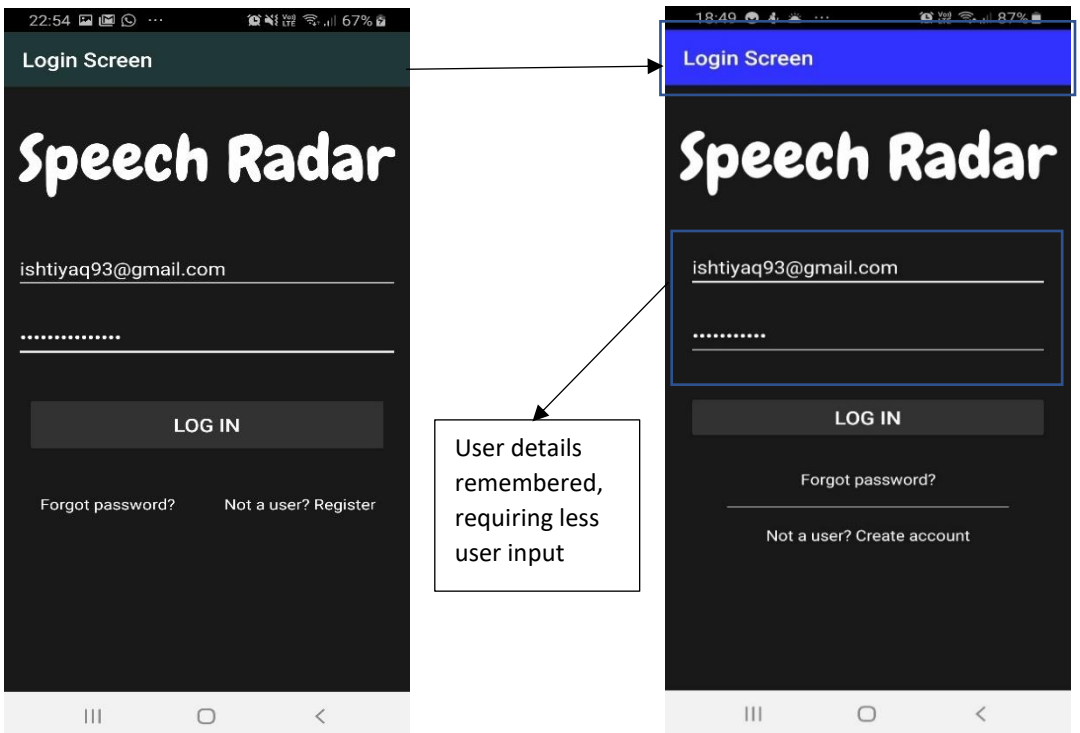
User ID	Possible improvements to interface?
13	**login screen should remember user details, to require less user input**
26	When resetting password, maybe you could input secret answer for additional security?
64	When creating an account, I should've been indicated clearly that password requires four digits minimum
77	Interface could be made to be more formal and appropriate for speech recognition, instead of it looking too cartoony
82	Buttons and other objects could be a bit bigger, for the visually impaired
102	**Colours used in interface are wrong. It should match the logo colours**
120	The bar on top of each screen should be removed because its damaging the appearance of app
162	Create account screen could require less input, so we can get straight into the app
169	Instead of creating an account, let user's login via Google or Facebook
173	Resetting password was too simple and unsecure. Should improve this screen
182	**Interface is too static. Needs more modern touch to it such as animation**
187	Login screen needs to remember account, as it would be painful to constantly login
196	Create account screen could improve in appearance and showing users where they've made errors, as it was difficult to know this when using the app

10.27.2.6 Proof of improvements:

10.27.2.6.1 Entrance animation:

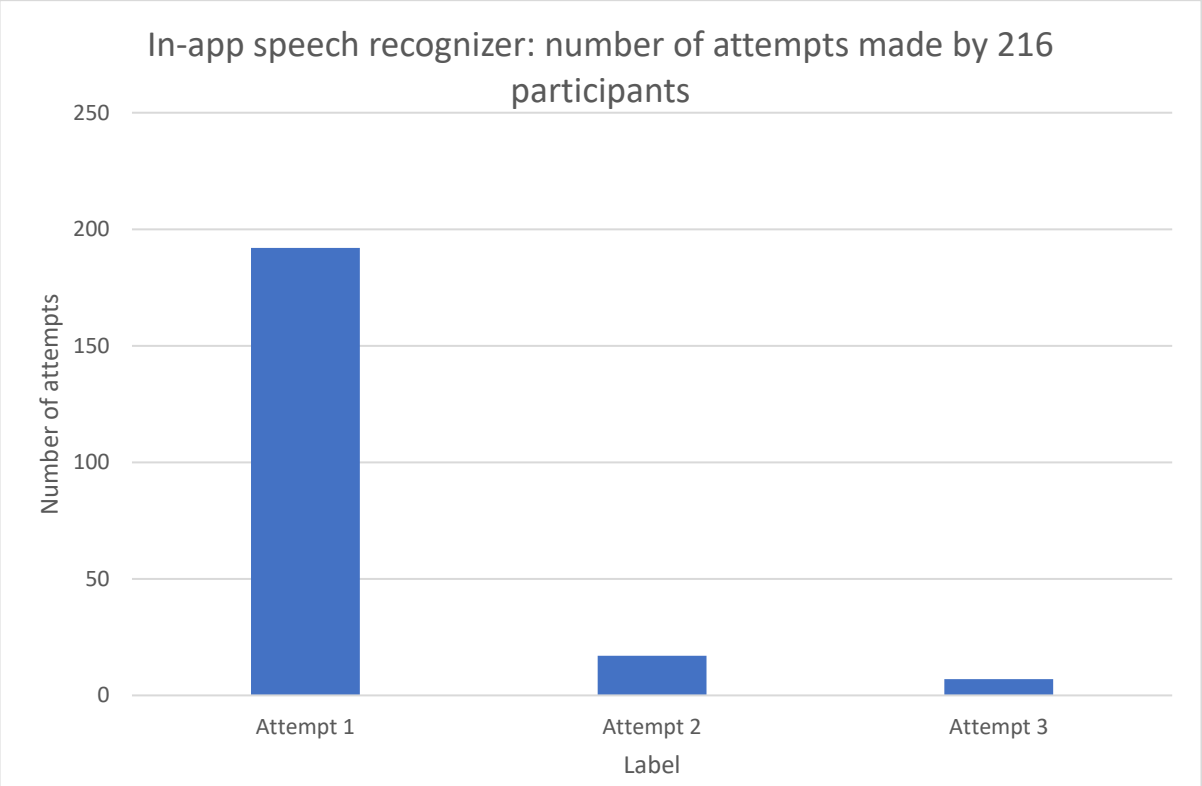
```
loginbutton.startAnimation(AnimationUtils.loadAnimation( context: loginscreen.this,android.R.anim.slide_in_left));
forgotpassword.startAnimation(AnimationUtils.loadAnimation( context: loginscreen.this,android.R.anim.slide_in_left));
createAccountLink.startAnimation(AnimationUtils.loadAnimation( context: loginscreen.this,android.R.anim.slide_in_left));
userEmail.startAnimation(AnimationUtils.loadAnimation( context: loginscreen.this,android.R.anim.slide_in_left));
userPass.startAnimation(AnimationUtils.loadAnimation( context: loginscreen.this,android.R.anim.slide_in_left));
maintitle2.startAnimation(AnimationUtils.loadAnimation( context: loginscreen.this,android.R.anim.slide_in_left));
```

10.27.2.6.2 Remembering user details & colour change:

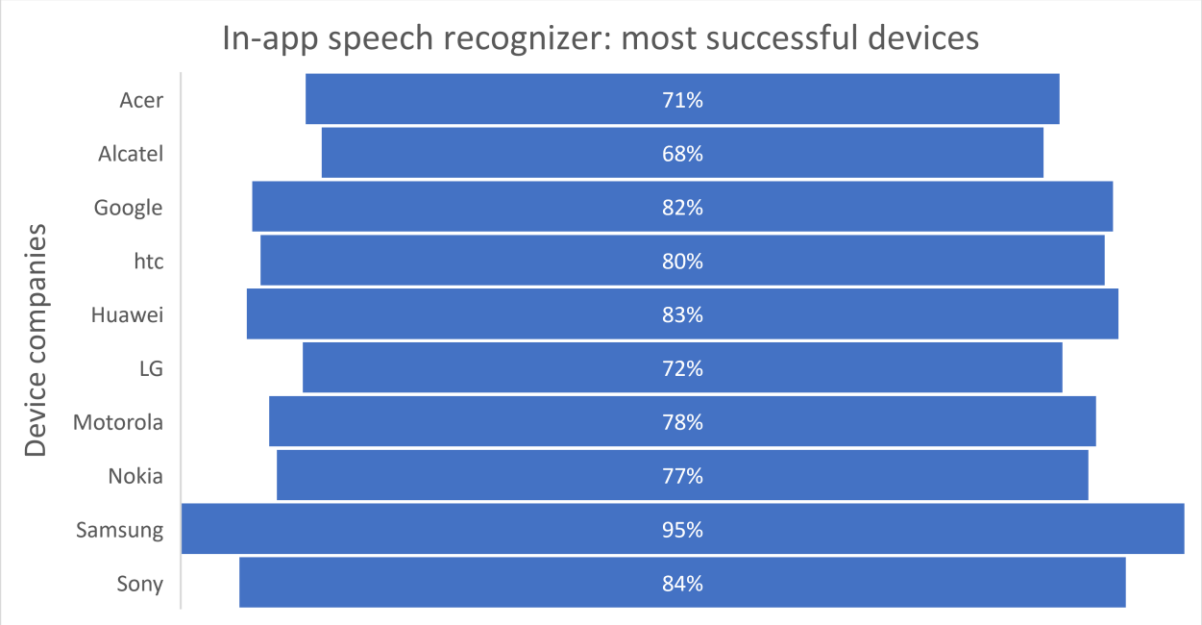


10.27.3 In-app speech recognizer

10.27.3.1 Number of attempts:

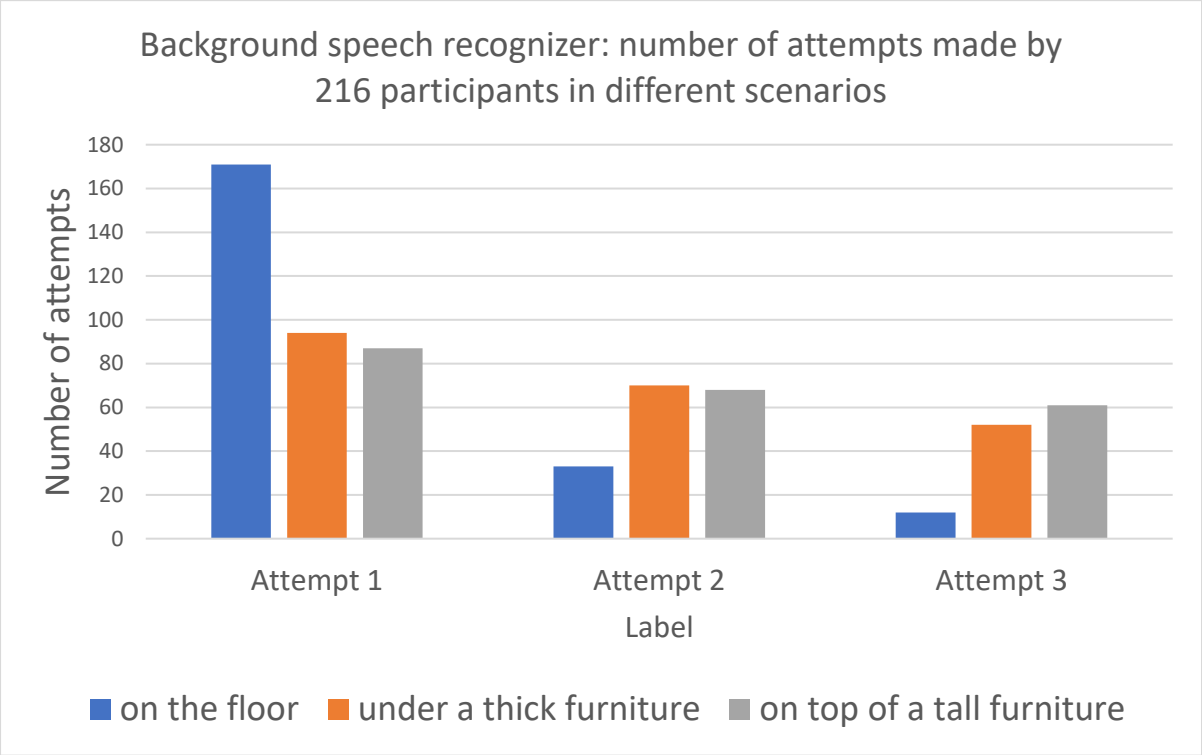


10.27.3.2 Most successful devices:

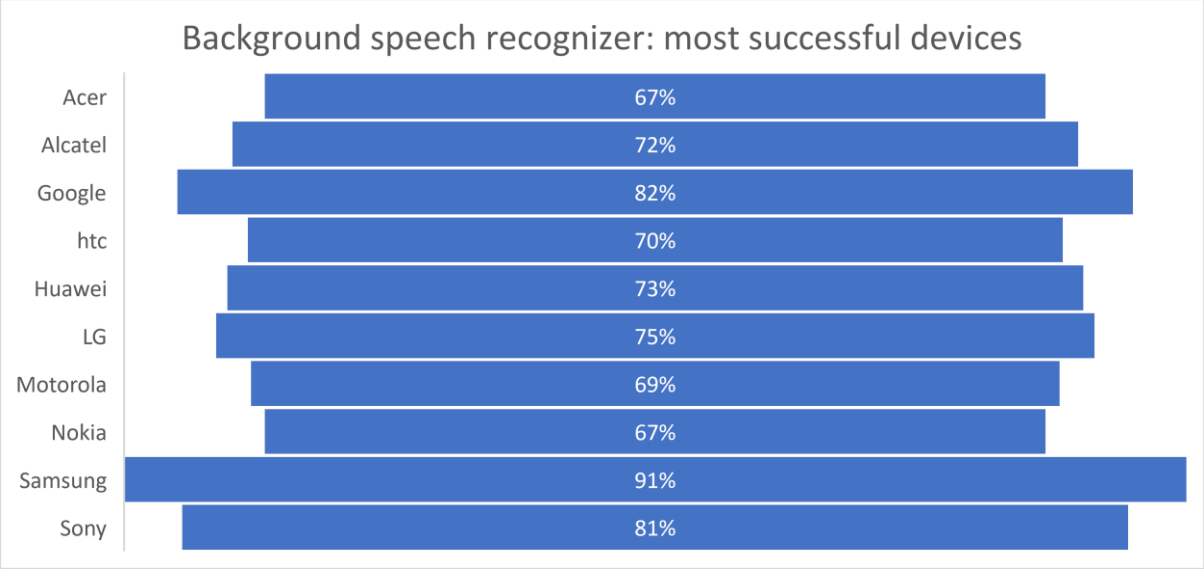


10.27.4 Background speech recognizer

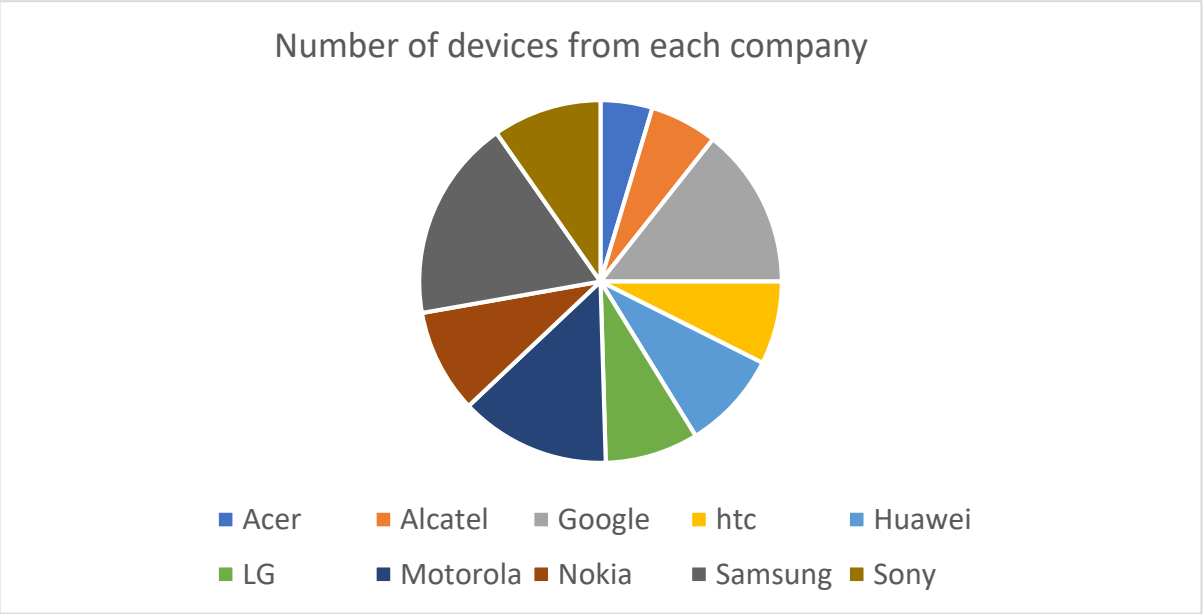
10.27.4.1 Number of attempts for every given scenario:



10.27.4.2 Most successful devices:



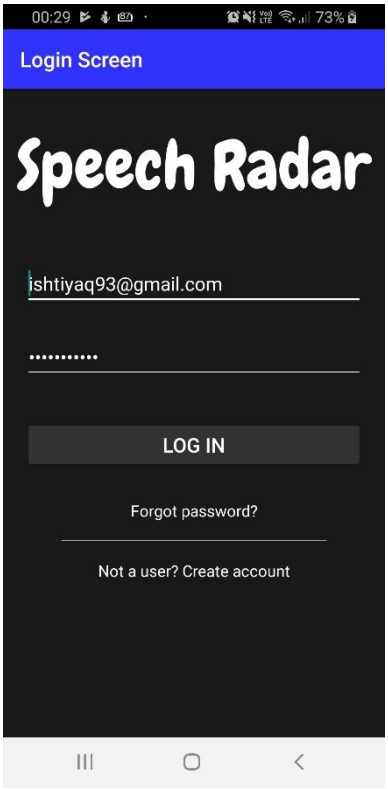
10.27.5 Additional chart



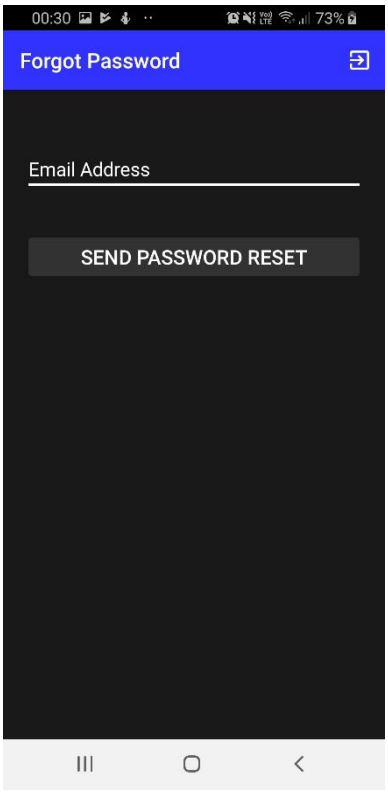
Appendix F Final App Screenshots

10.28 Splash screen

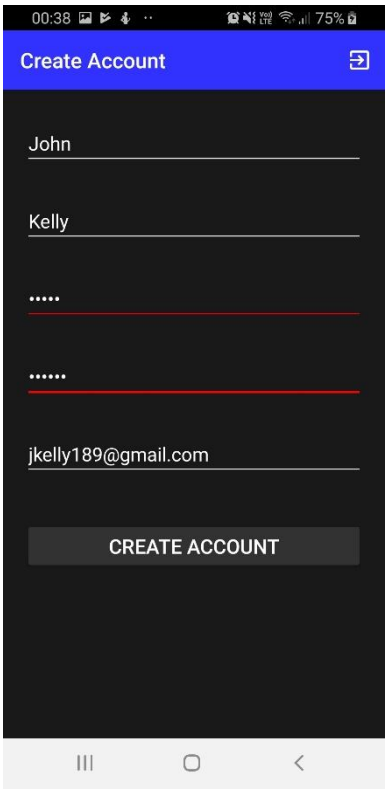
10.29 login screen



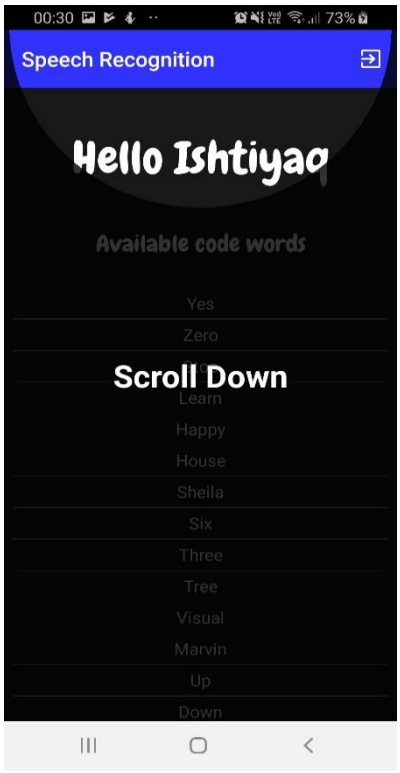
10.30 Forgot password

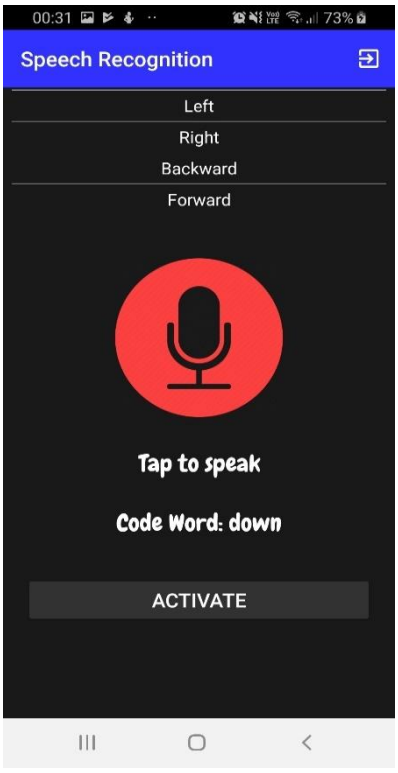


10.31 Create account

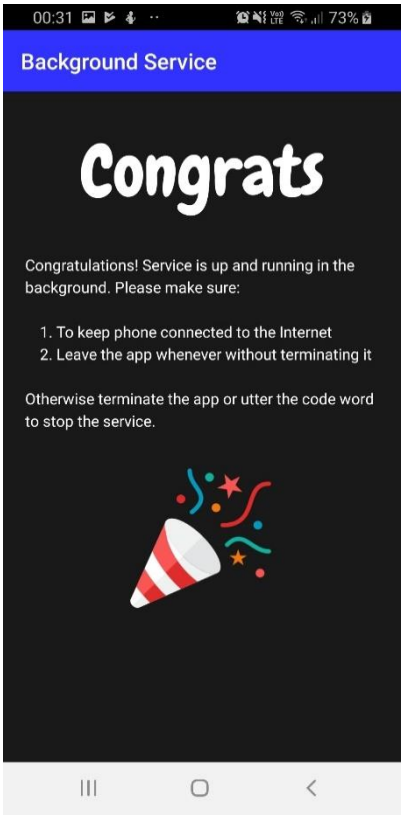


10.32 Speech recognition top and bottom of screen





10.33 Background service



Appendix G Source Code

10.34 Gradle

10.34.1 Build.gradle(Project: speechradar):

```
buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.1.2'
        classpath 'com.google.gms:google-services:4.0.1'
        classpath 'com.github.dcendents:android-maven-plugin:1.2'
        classpath 'com.jfrog.bintray.gradle:gradle-bintray-plugin:1.2'
    }
}

allprojects {
    repositories {
        google()
        jcenter()
        maven { url 'https://jitpack.io'
                }
        maven {
            url "https://maven.java.net/content/groups/public/"
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

10.34.2 build.gradle(Module: app):

```
apply plugin: 'com.android.application'

android {
    packagingOptions {
        pickFirst 'META-INF/LICENSE.txt'
    }
    compileSdkVersion 25
    buildToolsVersion '27.0.3'
    defaultConfig {
        applicationId "com.radar.speech.speechradar"
        minSdkVersion 16      //minimum API level
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
    compileOptions {
        targetCompatibility 1.8
        sourceCompatibility 1.8
    }
}

dependencies {   //all libraries used for this project
    implementation 'com.google.firebase:firebase-auth:16.0.2'
    implementation 'com.google.firebase:firebase-database:16.0.1'
    implementation 'com.google.firebase:firebase-core:16.0.1'
    implementation 'io.reactivex:rxjava:1.2.1'
    implementation 'com.tbruyelle.rxpermissions:rxpermissions:0.7.0@aar'
    implementation 'io.reactivex:rxandroid:1.2.1'
    implementation 'com.afollestad.material-dialogs:core:0.9.4.2'
    implementation 'com.github.faruktoptas:FancyShowCaseView:0.0.3'
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test.espresso:espresso-
core:3.0.2'
    implementation 'org.tensorflow:tensorflow-android:1.10.0'
    implementation 'com.sun.mail:android-mail:1.6.2'
    implementation 'com.sun.mail:android-activation:1.6.2'



    implementation project(path: ':speech')
}
apply plugin: 'com.google.gms.google-services'
```

10.35 XML

10.35.1 Android.Manifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.radar.speech.speechradar">
    <supports-screens android:largeScreens="true" android:normalScreens="true"
android:smallScreens="true" android:xlargeScreens="true" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
    <uses-permission android:name="android.permission.ACCESS_NOTIFICATION_POLICY"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/app_icon"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/app_icon_round"
        android:supportsRtl="true">
        <activity
            android:name=".SplashScreen"
            android:screenOrientation="portrait"
            android:noHistory="true"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter android:label="@string/app_name">
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".loginscreen"
            android:label="@string/loginscreen_actionbar_title"
            android:screenOrientation="portrait"
            android:theme="@style/AppTheme"></activity>
        <activity
            android:name=".createaccount"
            android:label="@string/createaccount_actionbar_title"
            android:screenOrientation="portrait"
            android:theme="@style/AppTheme" />
        <activity
            android:name=".speechrecognition"
            android:label="@string/speechrecognition_actionbar_title"
            android:noHistory="true"
            android:screenOrientation="portrait"
            android:theme="@style/AppTheme" />
        <activity
            android:name=".forgotPassword"
            android:label="@string/forgotPassword_actionbar_title"
            android:screenOrientation="portrait"
            android:theme="@style/AppTheme" />
        <service
            android:name=".continuousService"
            android:enabled="true"
            android:stopWithTask="true" />
        <activity
            android:name=".BackgroundService"
            android:label="@string/BackgroundService_actionbar_title"
            android:screenOrientation="portrait"
            android:theme="@style/AppTheme" />
    </application>
</manifest>
```

10.35.2 activity_splash_screen.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SplashScreen">

    <ImageView
        android:id="@+id/mylogo"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginBottom="48dp"
        android:layout_marginEnd="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginStart="28dp"
        android:layout_marginTop="28dp"
        app:layout_constraintBottom_toTopOf="@+id/loading"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/speechradarfulllogo" />

    <ProgressBar
        android:id="@+id/loading"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="27dp"
        android:layout_marginEnd="40dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="40dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

10.35.3 activity_loginscreen.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrolling"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/lightblack"
    tools:context=".loginscreen">
    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:fontFamily="@font/chewy"
        android:text="Speech Radar"
        android:textColor="@color/white"
        android:textSize="63dp"
        android:textAlignment="center"

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/emaillogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="48dp"

android:digits="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890.-
_:=+#$£^@()*/\;!?,'{}[]|>~"
        android:backgroundTint="@color/white"
        android:ems="10"
        android:hint="Email Address"
        android:inputType="textPersonName"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/title" />

    <EditText
        android:id="@+id/passwordlogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="24dp"
        android:backgroundTint="@color/white"
        android:ems="10"
        android:hint="Password"
        android:inputType="textPassword"
        android:textColor="@color/white"
```

```
        android:textColorHint="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/emaillogin" />
    <Button
        android:id="@+id/loginbtn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="36dp"
        android:backgroundTint="@color/oceanblue"
        android:text="LOG IN"
        android:textColor="@color/white"
        android:textSize="19dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.502"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/passwordlogin" />

        <TextView
            android:id="@+id/forgotPassword"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="28dp"
            android:text="Forgot password?"
            android:textColor="@color/white"
            android:textSize="15dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/loginbtn" />
        <View
            android:id="@+id/view2"
            android:layout_width="match_parent"
            android:layout_height="0.3dp"
            android:layout_marginEnd="55dp"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginStart="55dp"
            android:layout_marginTop="18dp"
            android:background="@color/white"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/forgotPassword" />

        <TextView
            android:id="@+id/createaccountlink"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="18dp"
            android:text="Not a user? Create account"
            android:textColor="@color/white"
            android:textSize="15dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/view2" />

    </android.support.constraint.ConstraintLayout>
</ScrollView>
```

10.35.4 activity_forgot_password.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/lightblack"
    tools:context=".forgotPassword">

    <EditText
        android:id="@+id/enteremail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="52dp"
        android:backgroundTint="@color/white"
        android:ems="10"
        android:hint="Email Address"
        android:inputType="textPersonName"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/passwordreset"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="36dp"
        android:backgroundTint="@color/oceanblue"
        android:text="SEND PASSWORD RESET"
        android:textColor="@color/white"
        android:textSize="19dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/enteremail" />
</android.support.constraint.ConstraintLayout>
```

10.35.5 activity_createaccount.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrolling"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/lightblack"
    tools:context=".createaccount">
    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    <EditText
        android:id="@+id/fname"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="28dp"
        android:backgroundTint="@color/white"
        android:ems="10"
        android:hint="First name"
        android:digits="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
        android:inputType="textCapWords"
        android:maxLength="17"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/lname"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="28dp"
        android:backgroundTint="@color/white"
        android:ems="10"
        android:hint="Last name"
        android:digits="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
        android:inputType="textCapWords"
        android:maxLength="17"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/fname" />

    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="28dp"
        android:backgroundTint="@color/white"
        android:ems="10"
```

```
        android:hint="Password"
android:digits="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890"
        android:inputType="textPassword"
        android:maxLength="20"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/lname" />
    <EditText
        android:id="@+id/conf_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="28dp"
        android:backgroundTint="@color/white"
        android:ems="10"
        android:hint="Confirm password"
android:digits="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890"
        android:inputType="textPassword"
        android:maxLength="20"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/password" />
    <EditText
        android:id="@+id/email_address"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="28dp"
        android:backgroundTint="@color/white"
        android:ems="10"
        android:hint="Email address"
   android:digits="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890.-
_:=+#$£^@()*/\;!?,'{}[]|>~"
        android:inputType="textPersonName"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/conf_password" />
        <Button
            android:id="@+id/create_account_btn"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginEnd="20dp"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginStart="20dp"
            android:layout_marginTop="40dp"
            android:backgroundTint="@color/oceanblue"
            android:text="CREATE ACCOUNT"
            android:textColor="@color/white"
            android:textSize="19dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/email_address" />
    </android.support.constraint.ConstraintLayout>
</ScrollView>
```

10.35.6 activity_speechrecognition.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrolling"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/lightblack"
    tools:context=".speechrecognition">


    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/paragraph"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="40dp"
            android:fontFamily="@font/chewy"
            android:textAlignment="center"
            android:textColor="@color/white"
            android:textSize="43dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.502"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />


        <TextView
            android:id="@+id/codewordtxt"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="40dp"
            android:fontFamily="@font/chewy"
            android:text="Available code words"
            android:textAlignment="center"
            android:textColor="@color/white"
            android:textSize="23dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.502"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/paragraph" />

        <ListView
            android:id="@+id/codewordlist"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="28dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.507"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/codewordtxt" />

        <ImageView
```

```
        android:id="@+id/speechrecognitionmic"
        android:layout_width="154dp"
        android:layout_height="174dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="32dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/codewordlist"
        app:srcCompat="@drawable/redmic" />

    <TextView
        android:id="@+id/tapspeak"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="20dp"
        android:fontFamily="@font/chewy"
        android:text="Tap to speak"
        android:textAlignment="center"
        android:textColor="@color/white"
        android:textSize="23dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/speechrecognitionmic" />

    <TextView
        android:id="@+id/speechtotext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="28dp"
        android:fontFamily="@font/chewy"
        android:text="...."
        android:textAlignment="center"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:textSize="23dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tapspeak" />

    <Button
        android:id="@+id/saveCodeWord"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="20dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="36dp"
        android:backgroundTint="@color/oceanblue"
        android:text="Activate"
        android:textColor="@color/white"
        android:textSize="19dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/speechtotext" />

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:layout_marginEnd="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="16dp"
            android:max="100"
            android:progress="0"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.501"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/saveCodeWord" />

        <TextView
            android:id="@+id/save"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginLeft="8dp"

            android:layout_marginRight="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:text="Saving..."
            android:textColor="@color/white"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/progressBar" />

    </android.support.constraint.ConstraintLayout>

</ScrollView>
```

10.35.7 activity_background_service.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrolling"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/lightblack"
    tools:context=".BackgroundService">


    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    <TextView
        android:id="@+id/title2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:fontFamily="@font/chewy"
        android:text="Congrats"
        android:textColor="@color/white"
        android:textSize="63dp"
        android:textAlignment="center"
        app:layout_constraintLeft_toLeftOf="parent"
       app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
        <TextView
            android:id="@+id/para"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginEnd="20dp"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginStart="20dp"
            android:layout_marginTop="35dp"
            android:lineSpacingExtra="4dp"
            android:text="Congratulations! Service is up and running in the
background. Please make sure:\n\n    1. To keep phone connected to the Internet \n
2. Leave the app whenever without terminating it\n\nOtherwise terminate the app or
utter the code word to stop the service."
            android:textColor="@color/white"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.515"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/title2" />

        <ImageView
            android:id="@+id/myimg"
            android:layout_width="188dp"
            android:layout_height="132dp"
            android:layout_marginEnd="8dp"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="32dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.497"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/para"
            app:srcCompat="@drawable/congratulations" />

    </android.support.constraint.ConstraintLayout>
</ScrollView>
```

10.35.8 colors.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name ="lightblack">#191919</color>
    <color name = "darkblue">#3232ff</color>
    <color name ="white">#ffffff</color>
    <color name="oceanblue">#323232</color>
    <color name="green">#32CD32</color>

</resources>
```

10.35.9 strings.xml:

```
<resources>
    <string name="createaccount_actionbar_title">Create Account</string>
    <string name="loginscreen_actionbar_title">Login Screen</string>
    <string name="speechrecognition_actionbar_title">Speech Recognition</string>
    <string name="forgotPassword_actionbar_title">Forgot Password</string>
    <string name="BackgroundService_actionbar_title">Background Service</string>
    <string name="start_service">Activate</string>
    <string name="stop_service">Deactivate</string>
    <string name="enable_autostart">AutoStart</string>
    <string name="permission_required">grant permission to use the mic</string>
    <string name="ask_permission">Please allow Speech to always run in the
background,else our services can\'t be accessed when you are in distress</string>
    <string name="allow">ALLOW</string>
    <string name="my_service_name">com.radar.speech.speechradar.MyService</string>
    <string name="app_name">Speech Radar</string>
</resources>
```

10.35.10 styles.xml:

```
<resources>

    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/darkblue</item>
    </style>

    <style name="AppTheme.NoActionBar"
parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
        <item name="android:windowFullscreen">true</item>
    </style>
</resources>
```

10.36 Java

10.36.1 SplashScreen.java:

```
package com.radar.speech.speechradar;

import android.content.Intent;
import android.graphics.Bitmap;
import android.os.CountDownTimer;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.ProgressBar;

public class SplashScreen extends AppCompatActivity {
    CountDownTimer mCountDownTimer;
    ProgressBar mProgressBar;
    int i=0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash_screen);
        mProgressBar=(ProgressBar)findViewById(R.id.loading);

        Log.d("undetected_word", "no word detected! Please try again");

        mCountDownTimer=new CountDownTimer(4000,400) {
            @Override
            public void onTick(long millisUntilFinished) {
                i = i + 10;
                mProgressBar.setProgress(i);
            }

            @Override
            public void onFinish() {
                 Intent i = new Intent(SplashScreen.this, loginscreen.class);
                    startActivity(i);
            }};
        mCountDownTimer.start();

    }

    private static Bitmap resize(Bitmap image, int maxWidth, int maxHeight) {
        if (maxHeight > 0 && maxWidth > 0) {
            int width = image.getWidth();
            int height = image.getHeight();
            float ratioBitmap = (float) width / (float) height;
            float ratioMax = (float) maxWidth / (float) maxHeight;

            int finalWidth = maxWidth;
            int finalHeight = maxHeight;
            if (ratioMax > ratioBitmap) {
                finalWidth = (int) ((float) maxHeight * ratioBitmap);
            } else {
                finalHeight = (int) ((float) maxWidth / ratioBitmap);
            }
            image = Bitmap.createScaledBitmap(image, finalWidth, finalHeight,
true);
            return image;
        } else {
            return image;
        }}
}
```

10.36.2 loginscreen.java:

```
package com.radar.speech.speechradar;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Typeface;
import android.os.SystemClock;
import android.support.annotation.NonNull;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class loginscreen extends AppCompatActivity {

    FirebaseAuth firebaseAuth;
    Button loginbutton;
    EditText userEmail;
    EditText userPass;
    TextView createAccountLink;
    TextView forgotpassword;
    TextView maintitle2;
    private long mLastClickTime;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_loginscreen);

        loginbutton = (Button)findViewById(R.id.loginbtn);
        userEmail = (EditText) findViewById(R.id.emaillogin) ;
        userPass = (EditText) findViewById(R.id.passwordlogin);
        createAccountLink = (TextView) findViewById(R.id.createaccountlink);
        forgotpassword = (TextView) findViewById(R.id.forgotPassword);
        firebaseAuth = FirebaseAuth.getInstance();
        maintitle2 = (TextView) findViewById(R.id.title) ;


loginbutton.startAnimation(AnimationUtils.loadAnimation(loginscreen.this,android.R.
anim.slide_in_left));

forgotpassword.startAnimation(AnimationUtils.loadAnimation(loginscreen.this,android
.R.anim.slide_in_left));

createAccountLink.startAnimation(AnimationUtils.loadAnimation(loginscreen.this,andr
oid.R.anim.slide_in_left));

userEmail.startAnimation(AnimationUtils.loadAnimation(loginscreen.this,android.R.an
im.slide_in_left));

userPass.startAnimation(AnimationUtils.loadAnimation(loginscreen.this,android.R.ani
m.slide_in_left));

maintitle2.startAnimation(AnimationUtils.loadAnimation(loginscreen.this,android.R.a
nim.slide_in_left));
```

```
        Typeface custom_font = Typeface.createFromAsset(getAssets(),
"fonts/chewy.ttf");

        maintitle2.setTypeface(custom_font);

        createAccountLink.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (SystemClock.elapsedRealtime() - mLastClickTime < 2500) {
                    return;
                }
                mLastClickTime = SystemClock.elapsedRealtime();
                startActivity(new Intent(loginscreen.this, createaccount.class));
            }
        });
        forgotpassword.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (SystemClock.elapsedRealtime() - mLastClickTime < 2500) {
                    return;
                }
                mLastClickTime = SystemClock.elapsedRealtime();
                startActivity(new Intent(loginscreen.this, forgotPassword.class));
            }
        });

        SharedPreferences prefs = getSharedPreferences("UserData", MODE_PRIVATE);
        String username = prefs.getString("emailaddress","");
        String password = prefs.getString("pword","");

        userEmail.setText(username);
        userPass.setText(password);

        check();

    }

    public void check() {
        loginbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                try {
              SharedPreferences prefs = getSharedPreferences("UserData", MODE_PRIVATE);
              SharedPreferences.Editor editor = prefs.edit();
              editor.putString("emailaddress", userEmail.getText().toString());
              editor.putString("pword", userPass.getText().toString());
              editor.apply();

firebaseAuth.signInWithEmailAndPassword(userEmail.getText().toString().trim(),
                            userPass.getText().toString())
                            .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                                @Override
                                public void onComplete(@NonNull Task<AuthResult>
task) {
                                    if (task.isSuccessful()) {
                                        if
(firebaseAuth.getCurrentUser().isEmailVerified()) {
                                            if (SystemClock.elapsedRealtime() -
mLastClickTime < 7000) {
                                                return;
                                            }
                                        mLastClickTime = SystemClock.elapsedRealtime();
                                        String email = userEmail.getText().toString();
                                        String email2 = userEmail.getText().toString();
                                        email = email.replace(".", "");
                                        email = email.replace(" ", "");
```

```
                    Intent i = new Intent(loginscreen.this, speechrecognition.class);
                                        i.putExtra("email_var", email);
                                        i.putExtra("emailwithstop", email2);
                                        startActivity(i);
                                    } else {
                Toast.makeText(loginscreen.this, "Please verify your email address"
                                                , Toast.LENGTH_LONG).show();
                                    }
                                } else {
                Toast.makeText(loginscreen.this, task.getException().getMessage()
                                                 , Toast.LENGTH_LONG).show();
                                }
                            }
                        });
                  }
                  catch (Exception e ) {

                Toast.makeText(loginscreen.this, e.toString()
                            , Toast.LENGTH_LONG).show();
                  }
              }});}

    @Override
    public void onBackPressed() {

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Are you sure you want to exit the app?")
                .setCancelable(false)
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        Intent intent = new Intent(Intent.ACTION_MAIN);
                        intent.addCategory(Intent.CATEGORY_HOME);
                        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                        startActivity(intent);                    }
                })
                .setNegativeButton("No", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        dialog.cancel();
                    }
                });
        AlertDialog alert = builder.create();
        alert.show();
    }
}
```

10.36.3 forgotPassword.java:

```
package com.radar.speech.speechradar;

import android.content.Intent;
import android.os.SystemClock;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;

public class forgotPassword extends AppCompatActivity {
    Button resetpass;
    EditText email;
    private long mLastClickTime;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_forgot_password);
        resetpass = (Button)findViewById(R.id.passwordreset);
        email = (EditText) findViewById(R.id.enteremail) ;

resetpass.startAnimation(AnimationUtils.loadAnimation(forgotPassword.this,android.R
.anim.slide_in_left));

email.startAnimation(AnimationUtils.loadAnimation(forgotPassword.this,android.R.ani
m.slide_in_left));

        send_password_reset_email();

    }

    public void send_password_reset_email() {
        resetpass.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                try {

FirebaseAuth.getInstance().sendPasswordResetEmail(email.getText().toString().trim()
)
                            .addOnCompleteListener(new OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {

                                    if (task.isSuccessful()) {
                                        Toast.makeText(forgotPassword.this, "Reset
password sent to " + email.getText().toString(),
                                                Toast.LENGTH_LONG).show();
                                    } else {
                                        Toast.makeText(forgotPassword.this,
task.getException().getMessage(),
                                                Toast.LENGTH_LONG).show();
                                    }
```

```
                                    }
                                });
                        }
                        catch(Exception e){
                            Toast.makeText(forgotPassword.this, e.toString(),
                                    Toast.LENGTH_LONG).show();
                        }

                    }


                }
                );
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            MenuInflater inflater = getMenuInflater();
            inflater.inflate(R.menu.speechrecognition_menu, menu);
            return true;
        }
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            // Handle item selection
            switch (item.getItemId()) {
                case R.id.menu_home:
                    if (SystemClock.elapsedRealtime() - mLastClickTime < 2000) {
                        return false;
                    }
                    mLastClickTime = SystemClock.elapsedRealtime();
                    startActivity(new Intent(forgotPassword.this, loginscreen.class));
                    return true;
                default:
                    return super.onOptionsItemSelected(item);
            }
        }
}
```

10.36.4 createaccount.java:

```
package com.radar.speech.speechradar;

import android.content.Intent;
import android.graphics.Color;
import android.graphics.PorterDuff;
import android.os.SystemClock;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;

public class createaccount extends AppCompatActivity {
    Button createAccountbtn;
    EditText fname,lname,password,conf_password,email_address;
    FirebaseAuth firebaseAuth;
    DatabaseReference myRef;
    private long mLastClickTime;
    DBHelper myDb;
    public static final String TABLE_NAME = "recovery_account_table";


    private final static String
salt="DGE$5SGr@3VsHYUMas2323E4d57vfBfFSTRU@!DSH(*%FDSdfg13sgfsg";

    @Override

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_createaccount);
        myRef = FirebaseDatabase.getInstance().getReference();

        createAccountbtn = (Button) findViewById(R.id.create_account_btn);
        fname = (EditText) findViewById(R.id.fname);
        lname = (EditText) findViewById(R.id.lname);
        password = (EditText) findViewById(R.id.password);
        conf_password = (EditText) findViewById(R.id.conf_password);
        email_address = (EditText) findViewById(R.id.email_address);

        myDb = new DBHelper(this);

        firebaseAuth = FirebaseAuth.getInstance();

        fname.startAnimation(AnimationUtils.loadAnimation(createaccount.this,
android.R.anim.slide_in_left));
        lname.startAnimation(AnimationUtils.loadAnimation(createaccount.this,
android.R.anim.slide_in_left));
```

```
        password.startAnimation(AnimationUtils.loadAnimation(createaccount.this,
android.R.anim.slide_in_left));

conf_password.startAnimation(AnimationUtils.loadAnimation(createaccount.this,
android.R.anim.slide_in_left));

email_address.startAnimation(AnimationUtils.loadAnimation(createaccount.this,
android.R.anim.slide_in_left));

createAccountbtn.startAnimation(AnimationUtils.loadAnimation(createaccount.this,
android.R.anim.slide_in_left));

                AddData();

    }
    public  void AddData() {
        createAccountbtn.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {

                        final String passwordstring =
password.getText().toString();
                        final String confirmpasswordstring =
conf_password.getText().toString();
                        final String firstname = fname.getText().toString();
                        final String lastname = lname.getText().toString();
                        final String emailaddress =
email_address.getText().toString();
                        int size1 = firstname.length();
                        int size2 = lastname.length();
                        if(size1 >1) {
                            fname.getBackground().setColorFilter(Color.WHITE,
PorterDuff.Mode.SRC_ATOP);
                        }
                        if(size2>1) {
                            lname.getBackground().setColorFilter(Color.WHITE,
PorterDuff.Mode.SRC_ATOP);
                        }
                        if(passwordRules(passwordstring) == true) {
                            password.getBackground().setColorFilter(Color.WHITE,
PorterDuff.Mode.SRC_ATOP);
                        }
                        if(confirmpasswordstring.equals(passwordstring)) {

conf_password.getBackground().setColorFilter(Color.WHITE,
PorterDuff.Mode.SRC_ATOP);
                        }
                        if(isValidEmailAddress(emailaddress) == true) {

email_address.getBackground().setColorFilter(Color.WHITE,
PorterDuff.Mode.SRC_ATOP);
                        }

                        if(size1 > 1 && size2 > 1 && passwordRules(passwordstring)
== true && confirmpasswordstring.equals(passwordstring) &&
!email_address.getText().toString().equals("") && isValidEmailAddress(emailaddress)
== true) {



firebaseAuth.createUserWithEmailAndPassword(email_address
.getText().toString().trim(),
                                        password.getText().toString())
                                        .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                                            @Override
                                            public void onComplete(@NonNull
```

```
Task<AuthResult> task) {

                                        if (task.isSuccessful()) {


firebaseAuth.getCurrentUser().sendEmailVerification()
                                                .addOnCompleteListener(new
OnCompleteListener<Void>() {
                                                    @Override
                                                    public void
onComplete(@NonNull Task<Void> task) {
                                                        boolean isInserted
= false;


if(task.isSuccessful()){

Toast.makeText(createaccount.this, "Registered successfully. Please check your
email for verification",

Toast.LENGTH_LONG).show();
                                                                String id =
myRef.push().getKey();
                                                                String
hashpassword = Hash(passwordstring);
                                                                String
hashconfpassword = Hash(confirmpasswordstring);
                                                                String
emailaddress2 = emailaddress.replace(".", "");
                                                                emailaddress2 =
emailaddress2.replace(" ","");
                                                                Account account
= new Account(id,firstname,lastname,hashpassword,hashconfpassword,emailaddress);

myRef.child(emailaddress2).setValue(account);

myDb.insert_to_DB(fname.getText().toString(), lname.getText().toString(),
hashpassword,hashconfpassword,email_address.getText().toString(),"yes" );



                                                            }else{

Toast.makeText(createaccount.this,  task.getException().getMessage(),

Toast.LENGTH_LONG).show();
                                                            }

                                                        }
                                                    });
                                        }

                                        else {

                                            Toast.makeText(createaccount.this,
task.getException().getMessage(),
                                                    Toast.LENGTH_LONG).show();
                                        }
                                    }
                                });
                        }
                        else {
                            if(email_address.getText().toString().equals("")) {

email_address.getBackground().setColorFilter(Color.RED, PorterDuff.Mode.SRC_ATOP);
                            }
                            if(size1 <=1) {
                                fname.getBackground().setColorFilter(Color.RED,
```

```
PorterDuff.Mode.SRC_ATOP);
                                }
                                if(size2<=1) {
                                    lname.getBackground().setColorFilter(Color.RED,
PorterDuff.Mode.SRC_ATOP);
                                }
                                if(passwordRules(passwordstring) == false) {
                                    password.getBackground().setColorFilter(Color.RED,
PorterDuff.Mode.SRC_ATOP);

                                }
                                if(!confirmpasswordstring.equals(passwordstring)) {

conf_password.getBackground().setColorFilter(Color.RED, PorterDuff.Mode.SRC_ATOP);

                                }
                                if(isValidEmailAddress(emailaddress) == false) {

email_address.getBackground().setColorFilter(Color.RED, PorterDuff.Mode.SRC_ATOP);

                                }
                                 Toast.makeText(createaccount.this, "Error in above
input fields. Please correct them. Make sure password is min 8 characters long
containing 5 digits",
                                        Toast.LENGTH_LONG).show();
                        }
                    }}
            );
        }

        public boolean isValidEmailAddress(String emailaddress) {
            boolean result = true;
            try {
                InternetAddress emailAddress = new InternetAddress(emailaddress);
                emailAddress.validate();
            } catch (AddressException ex) {
                result = false;
            }
            catch (Exception e) {
                Toast.makeText(createaccount.this, e.toString(),
                        Toast.LENGTH_LONG).show();
            }
            return result;
        }

        public static String Hash(String message) {
            String md5 = "";
            if(null == message)
                return null;

            message = message+salt;
            try {
                MessageDigest digest = MessageDigest.getInstance("MD5");
                digest.update(message.getBytes(), 0, message.length());
                md5 = new BigInteger(1, digest.digest()).toString(16);

            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
            return md5;
        }

        public static boolean passwordRules(String password) {
            boolean c = true;

            int number = 0;

            if (password.length() < 8) {
```

```
            return false;
        }

        char elem;

        for(int i = 0; i < password.length(); i++ ){

            elem = password.charAt( i );

            if( Character.isDigit(elem) ){
                number++;
            }
        }

        if( number < 5 ){
            return false;
        }

        if( !password.matches("[a-zA-Z0-9]+") ){
            return false;
        }

        return c;

    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.speechrecognition_menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_home:
                if (SystemClock.elapsedRealtime() - mLastClickTime < 2000) {
                    return false;
                }
                mLastClickTime = SystemClock.elapsedRealtime();
                startActivity(new Intent(createaccount.this, loginscreen.class));
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

}
```

10.36.5 speechrecognition.java:

most of the code in this Java file was sourced. The parts that were not sourced can be seen in Chapter 5 Implementation

10.36.6 BackgroundService.java:

```
package com.radar.speech.speechradar;

import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.TextView;
import com.afollestad.materialdialogs.MaterialDialog;
import com.afollestad.materialdialogs.Theme;

public class BackgroundService extends loginscreen {
    Bundle extras;
    public  String values;
    TextView txt;
    TextView txt2;
    ImageView img;
    public static String values2;
    public static int counter = 0;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_background_service);
        txt = (TextView) findViewById(R.id.title2);
        txt2 = (TextView) findViewById(R.id.para);
        img = (ImageView) findViewById(R.id.myimg);
        extras = getIntent().getExtras();
        values = extras.getString("email_var2");
        values2 = values;
        startService(new Intent(BackgroundService.this, MyService.class));
        counter++;
        txt.startAnimation(AnimationUtils.loadAnimation(BackgroundService.this,
android.R.anim.slide_in_left));
        txt2.startAnimation(AnimationUtils.loadAnimation(BackgroundService.this,
android.R.anim.slide_in_left));
        img.startAnimation(AnimationUtils.loadAnimation(BackgroundService.this,
android.R.anim.slide_in_left));

        Typeface custom_font = Typeface.createFromAsset(getAssets(),
"fonts/chewy.ttf");
        txt.setTypeface(custom_font);

        AutoStart();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    @Override
    public void onBackPressed() {
    }
}
```

10.36.7 continuousService.java:

```
package com.radar.speech.speechradar;

import android.Manifest.permission;
import android.app.AlarmManager;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.media.AudioManager;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Build.VERSION;
import android.os.Build.VERSION_CODES;
import android.os.CountDownTimer;
import android.os.IBinder;
import android.text.TextUtils;
import android.widget.Toast;
import com.sac.speech.Speech;
import com.sac.speech.SpeechDelegate;
import com.tbruyelle.rxpermissions.RxPermissions;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.Random;

public class continuousService extends Service implements SpeechDelegate,
Speech.stopDueToDelay {

    public static SpeechDelegate del;
    private Context mContext;
    CountDownTimer mCountDownTimer;

    @Override
    public int onStartCommand(Intent intent, int a, int b) {
        Speech tmp = new Speech(this.getApplicationContext());
        del = continuousService.this;
        tmp.setListener(continuousService.this);
        if(tmp.isListening()) {
            tmp.stopListening();
            setRingtoneSoundLevel();
        } else {

RxPermissions.getInstance(continuousService.this).request(Manifest.permission.RECOR
D_AUDIO).subscribe(granted -> {
                if(granted) {
                    tmp.stopTextToSpeech();
                    tmp.startListening(null, continuousService.this);
                }});
            setRingtoneSoundLevel();
        }
        return START_STICKY;
    }


    @Override
    public IBinder onBind(Intent intent) {
        //TODO for communication return IBinder implementation
        return null;
    }

    @Override
    public void onStartOfSpeech() {
    }
```

```
    @Override
    public void onSpeechRmsChanged(float value) {

    }

    @Override
    public void onSpeechPartialResults(List<String> results) {

    }

    @Override
    public void onSpeechResult(String result) {
        mContext = getApplicationContext();
        String arr[] = result.split(" ", 2);
        String val = BackgroundService.values2;
        ArrayList<String> a = new ArrayList<String>();
        Uri uri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_RINGTONE);
        Ringtone ringtone = RingtoneManager.getRingtone(mContext,uri);

        if (!TextUtils.isEmpty(arr[0])) {
            Toast.makeText(this, arr[0], Toast.LENGTH_SHORT).show();
        }
        if(a.size() < 2) {
            if (val.equals("Sheila")) {
                if (arr[0].equals("cheetah") || arr[0].equals("tschida")) {
                    a.add(arr[0]);
                    if (a.get(0).equals("cheetah") || arr[0].equals("tschida")) {
                        setRingtoneSoundLevel();

                        ringtone.play();
                        stopService(new Intent(getApplicationContext(),
continuousService.class));

                        if (ringtone.isPlaying() == true) {
                            mCountDownTimer = new CountDownTimer(7000, 1000) {
                                @Override
                                public void onTick(long millisUntilFinished) {
                                }

                                @Override
                                public void onFinish() {

                                    ringtone.stop();

                                }
                            };
                            mCountDownTimer.start();
                        }}
                }
            }
            if (val.equals("backward")) {
                if (arr[0].equals("backwood")) {
                    a.add(arr[0]);

                    if (a.get(0).equals("backwood")) {
                        setRingtoneSoundLevel();
                        ringtone.play();
                        stopService(new
Intent(getApplicationContext(),continuousService.class));
                        if (ringtone.isPlaying() == true) {
                            mCountDownTimer = new CountDownTimer(7000, 1000) {
                                @Override
                                public void onTick(long millisUntilFinished) {
                                }

                                @Override
                                public void onFinish() {
```

```
                                    ringtone.stop();
                                }
                            };
                            mCountDownTimer.start();
                        }
                    }
                } }
            if (arr[0].equals(val)) {
                a.add(arr[0]);
            }
            if (a.get(0).equals(val)) {
                setRingtoneSoundLevel();
                ringtone.play();
                stopService(new
Intent(getApplicationContext(),continuousService.class));
                if (ringtone.isPlaying() == true) {
                    mCountDownTimer = new CountDownTimer(7000, 1000) {
                        @Override
                        public void onTick(long millisUntilFinished) {
                        }

                        @Override
                        public void onFinish() {

                            ringtone.stop();

                        }
                    };
                    mCountDownTimer.start();
                }
            } }
    }

    private void setRingtoneSoundLevel() {
        AudioManager amanager = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
        if (amanager != null) {
            amanager.setStreamMute(AudioManager.STREAM_NOTIFICATION, true);
            amanager.setStreamMute(AudioManager.STREAM_ALARM, true);
            amanager.setStreamMute(AudioManager.STREAM_MUSIC, true);
            amanager.setStreamMute(AudioManager.STREAM_SYSTEM, true);
            int maxvol = amanager.getStreamMaxVolume(AudioManager.STREAM_RING);
            float percentage = 0.9f;
            int setvolume = (int) (maxvol * percentage);


            if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.M) {
                try {
                    amanager.setStreamVolume(AudioManager.STREAM_RING, setvolume,
0);
                } catch (Exception e) {
                    NotificationManager mNotificationManager =
(NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
                    if (!mNotificationManager.isNotificationPolicyAccessGranted())
{
                        Intent intent = new
Intent(android.provider.Settings.ACTION_NOTIFICATION_POLICY_ACCESS_SETTINGS);
                        getApplicationContext().startActivity(intent);
                    }
                }
            } else {
                amanager.setStreamVolume(AudioManager.STREAM_RING, setvolume, 0);
            }
            }}
}
```

10.36.8 Account.java:

```
package com.radar.speech.speechradar;

public class Account {
    String d_id;
    String d_firstname;
    String d_lastname;
    String d_password;
    String d_confirm_password;
    String d_email_address;

    public Account(String d_id, String d_firstname, String d_lastname, String
d_password, String d_confirm_password, String d_email_address) {
        this.d_id = d_id;
        this.d_firstname = d_firstname;
        this.d_lastname = d_lastname;
        this.d_password = d_password;
        this.d_confirm_password = d_confirm_password;
        this.d_email_address = d_email_address;
    }

    public String getD_id() {
        return d_id;
    }
    public void setD_id(String d_id) {
        this.d_id = d_id;
    }
    public String getD_firstname() {
        return d_firstname;
    }

    public void setD_firstname(String d_firstname) {
        this.d_firstname = d_firstname;
    }

    public String getD_lastname() {
        return d_lastname;
    }
    public void setD_lastname(String d_lastname) {
        this.d_lastname = d_lastname;
    }

    public String getD_password() {
        return d_password;
    }

    public void setD_password(String d_password) {
        this.d_password = d_password;
    }

    public String getD_confirm_password() {
        return d_confirm_password;
    }

    public void setD_confirm_password(String d_confirm_password) {
        this.d_confirm_password = d_confirm_password;
    }
    public String getD_email_address() {
        return d_email_address;
    }

    public void setD_email_address(String d_email_address) {
        this.d_email_address = d_email_address;
    }
}
```

10.36.9 DBHelper.java:

```
package com.radar.speech.speechradar;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHelper extends SQLiteOpenHelper {
    public static final String DATABASE = "Accounts.db";
    public static final String TABLE = "recovery_account_table";
    public static final String _1 = "ID";
    public static final String _2 = "FIRSTNAME";
    public static final String _3 = "LASTNAME";
    public static final String _4 = "PASSWORD";
    public static final String _5 = "CONFIRM_PASSWORD";
    public static final String _6 = "EMAIL_ADDRESS";
    public static final String _7 = "CODEWORD";


    public DBHelper(Context context) {
        super(context, DATABASE, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table " + TABLE +" (ID INTEGER PRIMARY KEY
AUTOINCREMENT,FIRSTNAME TEXT,LASTNAME TEXT,PASSWORD TEXT,CONFIRM_PASSWORD TEXT,
EMAIL_ADDRESS TEXT, CODEWORD TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS "+TABLE);
        onCreate(db);
    }

    public boolean insert_to_DB(String firstname,String lastname,String
password,String confirm_password,String email, String codeword) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(_2,firstname);
        contentValues.put(_3,lastname);
        contentValues.put(_4,password);
        contentValues.put(_5,confirm_password);
        contentValues.put(_6,email);
        contentValues.put(_7,codeword);


        long result = db.insert(TABLE,null ,contentValues);
        if(result == -1)
            return false;
        else
            return true;
    }

    public Cursor AllData() {
        SQLiteDatabase db = this.getWritableDatabase();
        Cursor res = db.rawQuery("select * from "+TABLE,null);
        return res;
    }
```

```
    public boolean updateDB(String id,String firstname,String lastname,String
password,String confirm_password,String email,String codeword) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(_1,id);
        contentValues.put(_2,firstname);
        contentValues.put(_3,lastname);
        contentValues.put(_4,password);
        contentValues.put(_5,confirm_password);
        contentValues.put(_6,email);
        contentValues.put(_7,codeword);

        db.update(TABLE, contentValues, "ID = ?",new String[] { id });
        return true;
    }

}
```

Appendix H GitHub

10.37 Link to GitHub repository

https://github.com/ish2nv/Computing-Project