|  |  |
| --- | --- |
| A picture of a winding road and trees  Speech radar  16/02/19 | **Student name:** shah ali  **degree title:** COMputer science  **supervisor name:** Jamie ward  **registration number:** 33455846 |

## **Contents page**

1. Introduction 2

2. Aims and objectives 2

a. Aim and objective 1: successful implementation 2

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

3. Methods 3

a. Aim and objective 1 step-by-step 3

b. Aim and objective 1 step-by-step 3

4. Project plan 4

a. Design & research 4

b. Implementation, preliminary report & review 4

c. Implementation (continued) 5

d. Testing 5

e. Final report 6

5. Progress to date 7

a. Design phase 7

b. Research phase 7

c. Implementation phase 7

d. Summary 8

6. Planned work 9

7. Appendices 9

a. Class diagram 9

b. Activity diagram 10

c. Wireframes 11

d. Speech recognition screen plan 11

e. Actual implementation for each screen 12

8. Reference list 13

## **1. Introduction**

In today's society, its normal seeing people in homes and offices 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 machines. 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 I will need to use for my 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 improving as days go by through inventions such as Amazon Alexa and Google Hub. 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 use 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, export the TensorFlow model as a *protobuf file* **[6]** and place it in the assets folder within Android Studio. Next, *compile TensorFlow into Android Studio via the 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 the defined ‘recognise’ 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 at the same time. 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

### **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 organization

## **4. Project plan**

### **a. Design & research**

### **b.** **Implementation, preliminary report & review**

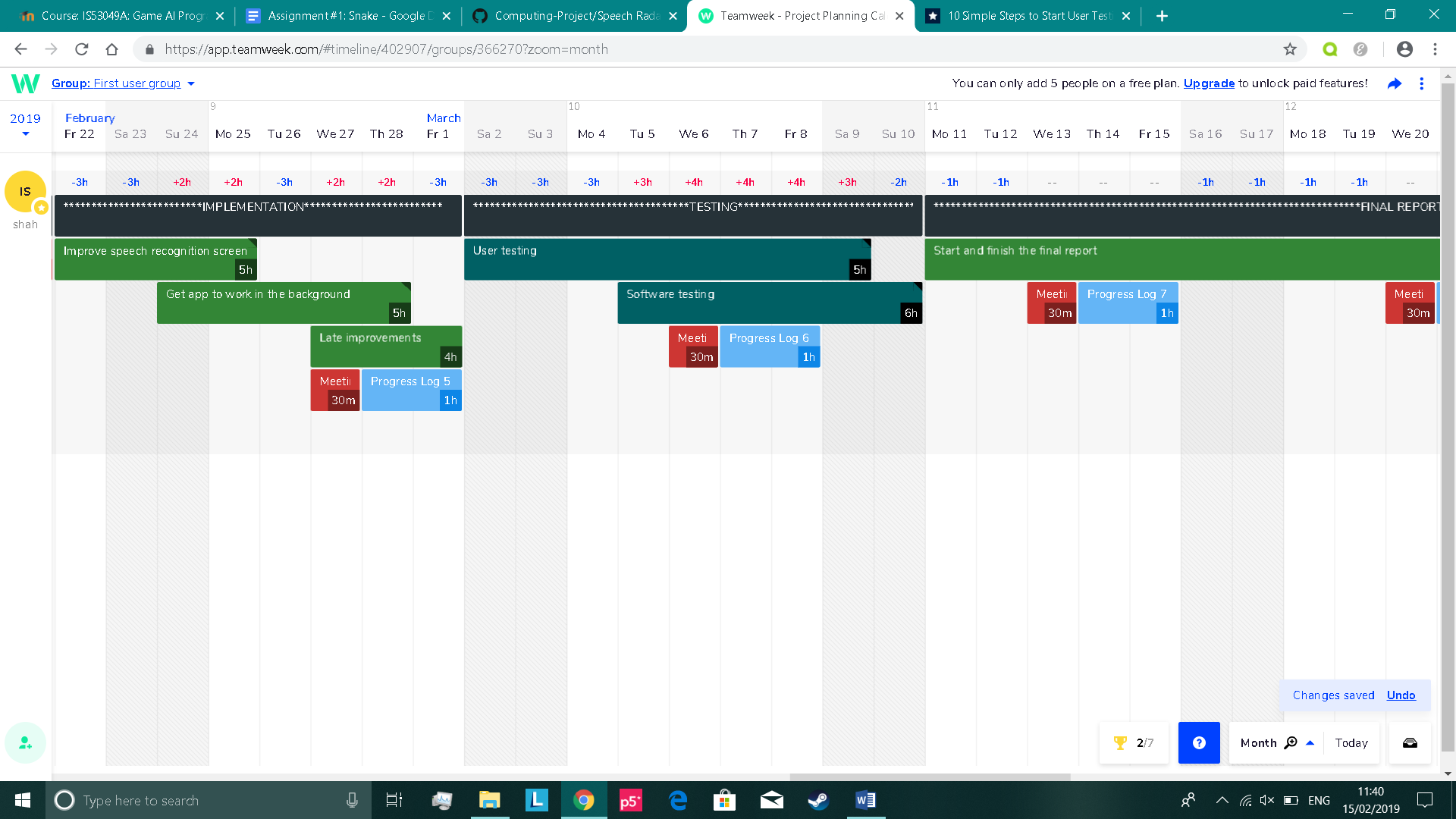
**Sub-tasks:**

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:**

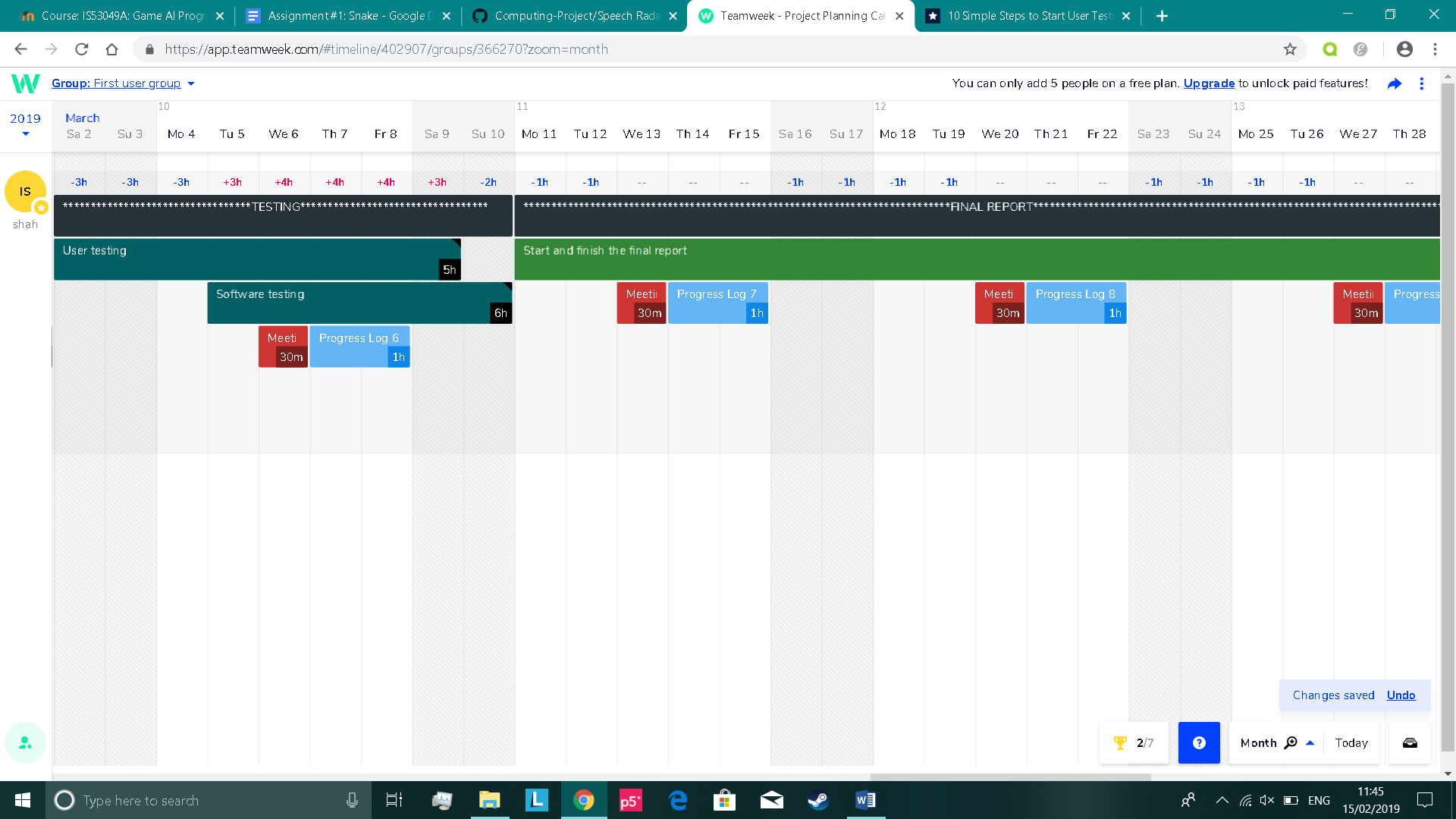
1. Create the tensorflow model first
2. Export the model as a protobuf (.pb) file once finished
3. Move the file to assets folder in Android Studio
4. Successfully load the model
5. Create a button that'll record audio
6. Convert the audio into a floating point array
7. Feed the array into the predefined recognise function
8. Test the application and see if it makes predictions accurately

### **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



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

### **d. Testing**

Sub-tasks:

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

### **e. Final report**

**Textual information:**

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 on the second week

**Visual information:**

1. Add all the appendices and references at the end of the report on the third week
2. Add the title page, contents page etc on the third week

## **5. Progress to date**

### **a. Design phase**

At the beginning of my project, I began the design phase. With little to no 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 would 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. 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 incorrect and if the email address has been verified by the user. Can 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 their, 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’.) Can 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.

## **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 increased the amount of time and work I do drastically on a daily basis. 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) for quite some time. If this all goes to plan, then I should have enough time for the final report at the end.

## **7. Appendices**

### class diagram.png **a. Class diagram**

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

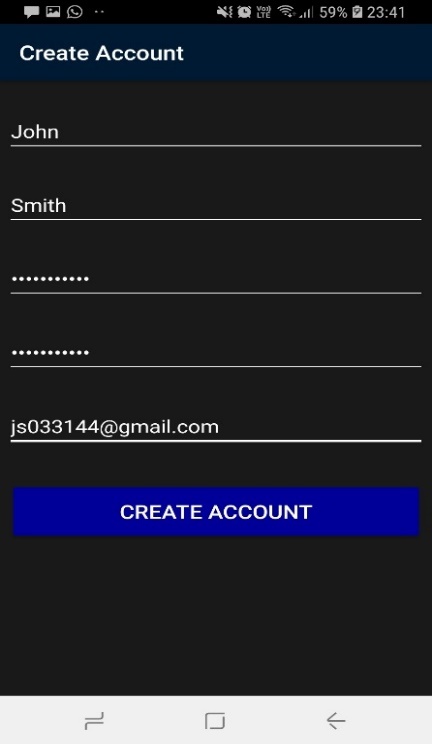
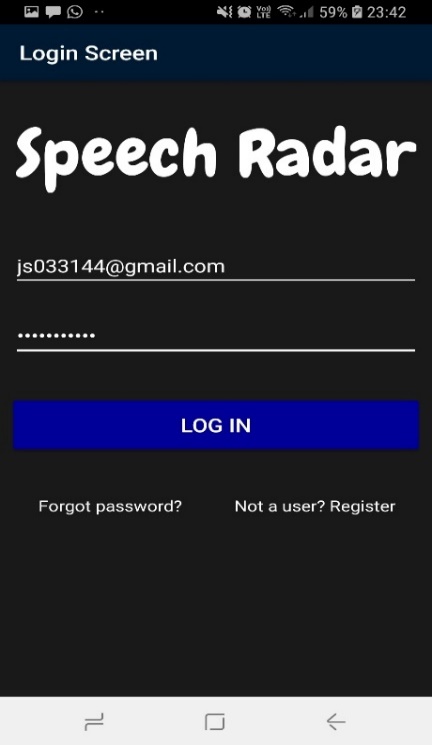
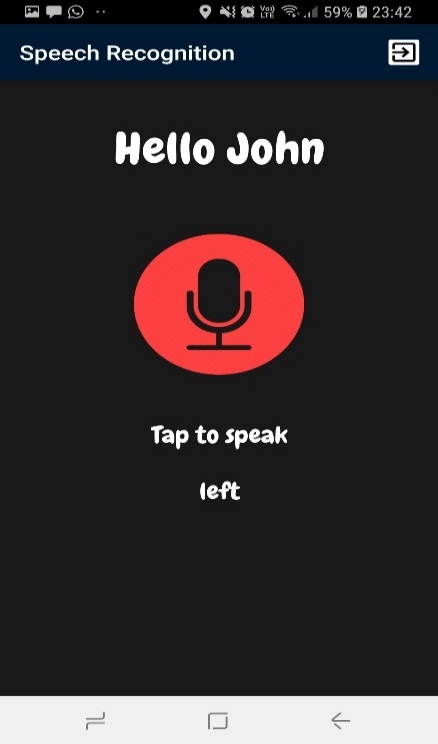
TensorFlowinferenceinterface is **one-to-many** with speech\_recognition as the speech\_recognition class will use many functions from the pre-defined TensorFlowinferenceinterface class

### activity-diagram.png**b. Activity diagram**

### speechrecognition1.pnglogin.pngcreateaccount.pngspeechrecognition2.png **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>