**Aims and objectives**

**First aim**

The situation of losing sight of your phone and not knowing its location is very common amongst all of us. Some people will try to ring their phone and follow the sound to uncover it. Some people will locate it via GPS. And some people will look everywhere for it tirelessly. The aim 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 on a daily basis 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.

**Second aim**

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 at the same time detect different accents. 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.

**Methods**

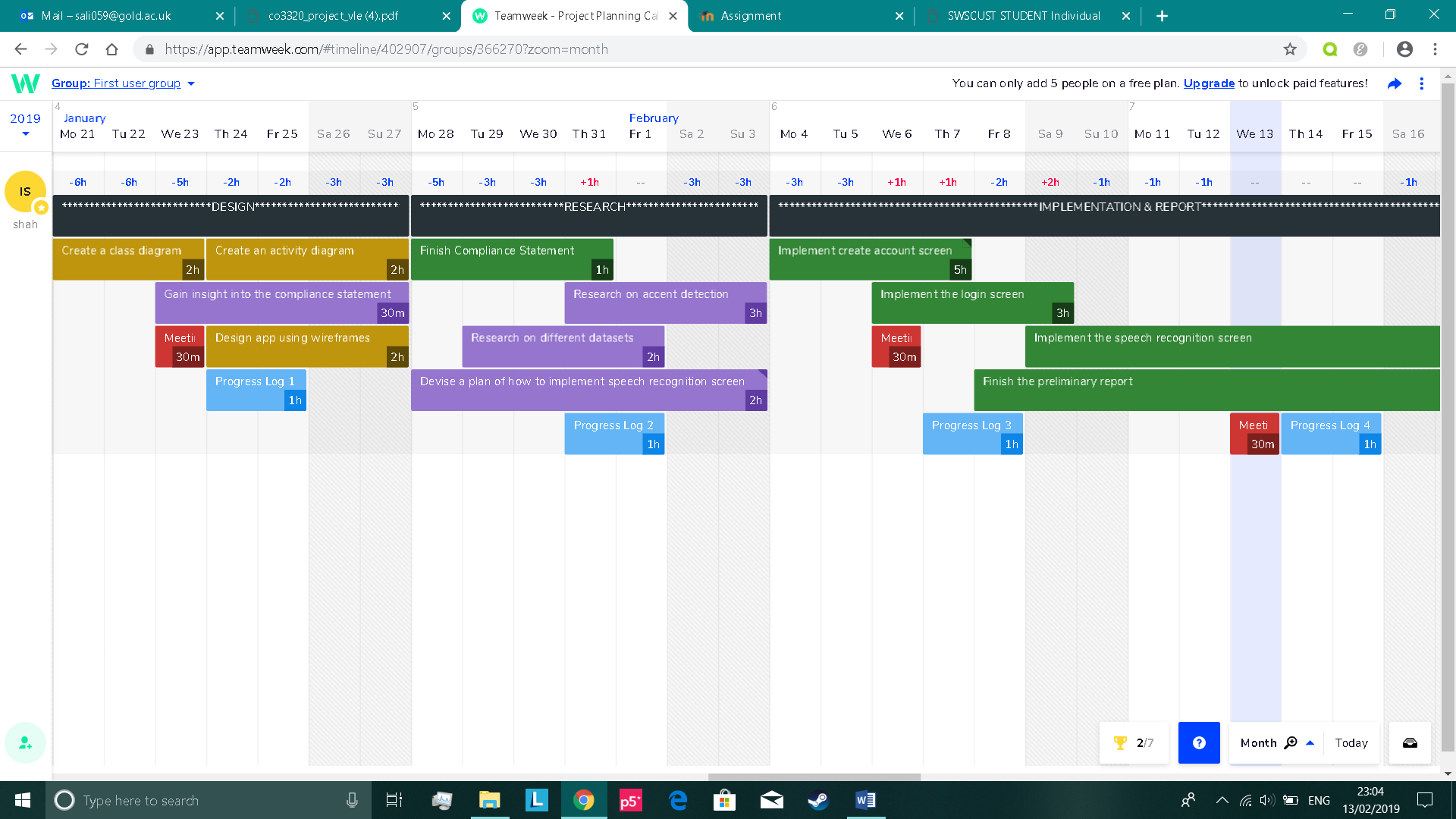
**I will achieve the first aim by following the steps mentioned below**

* Create a deep neural network in TensorFlow, which will be trained, validated and tested. The validation set will assist me in understanding whether the model is good with unseen data. The test set will help in measuring the predictive power of the model
* Then, export the TensorFlow model as a protobuf file and place it in the assets folder within Android Studio. Next, compile TensorFlow into Android Studio via the gradle. 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
* Once the model has been loaded, insert a button that records audio in the app. Store the audio in a short array, convert it into a floating point array 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. This is so it can identify if user utters the code word they registered (e.g. frog.) If it detects it, then the phone will start vibrating and ringing loudly. The links in the reference section show exactly how I could achieve this in Android Studio

**I will achieve the second aim by following the steps mentioned below**

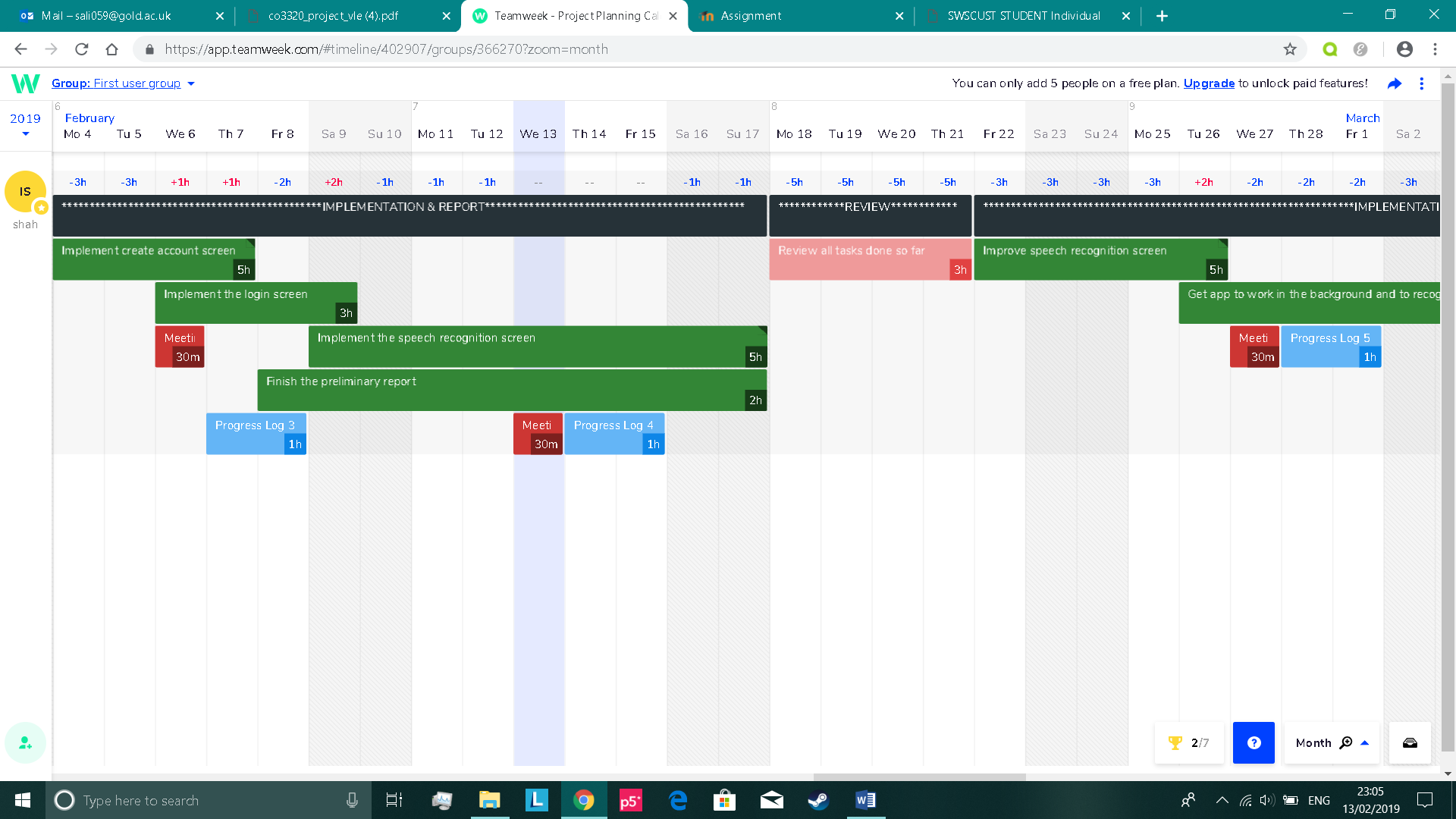
* I will use the Kaggle speech command dataset for my TensorFlow model, in order to classify one second audio clips of around 30 words. There are thousands of audio clips for each word uttered in different accents. This will be useful in enhancing the detection rate

**Project plan**

**Design & Research**

**Sub-tasks:**

1. Use pen and paper to start off the plan
2. Then develop on the plan using diagrams and flow-charts
3. Gain insight through research in whether the plan is feasible

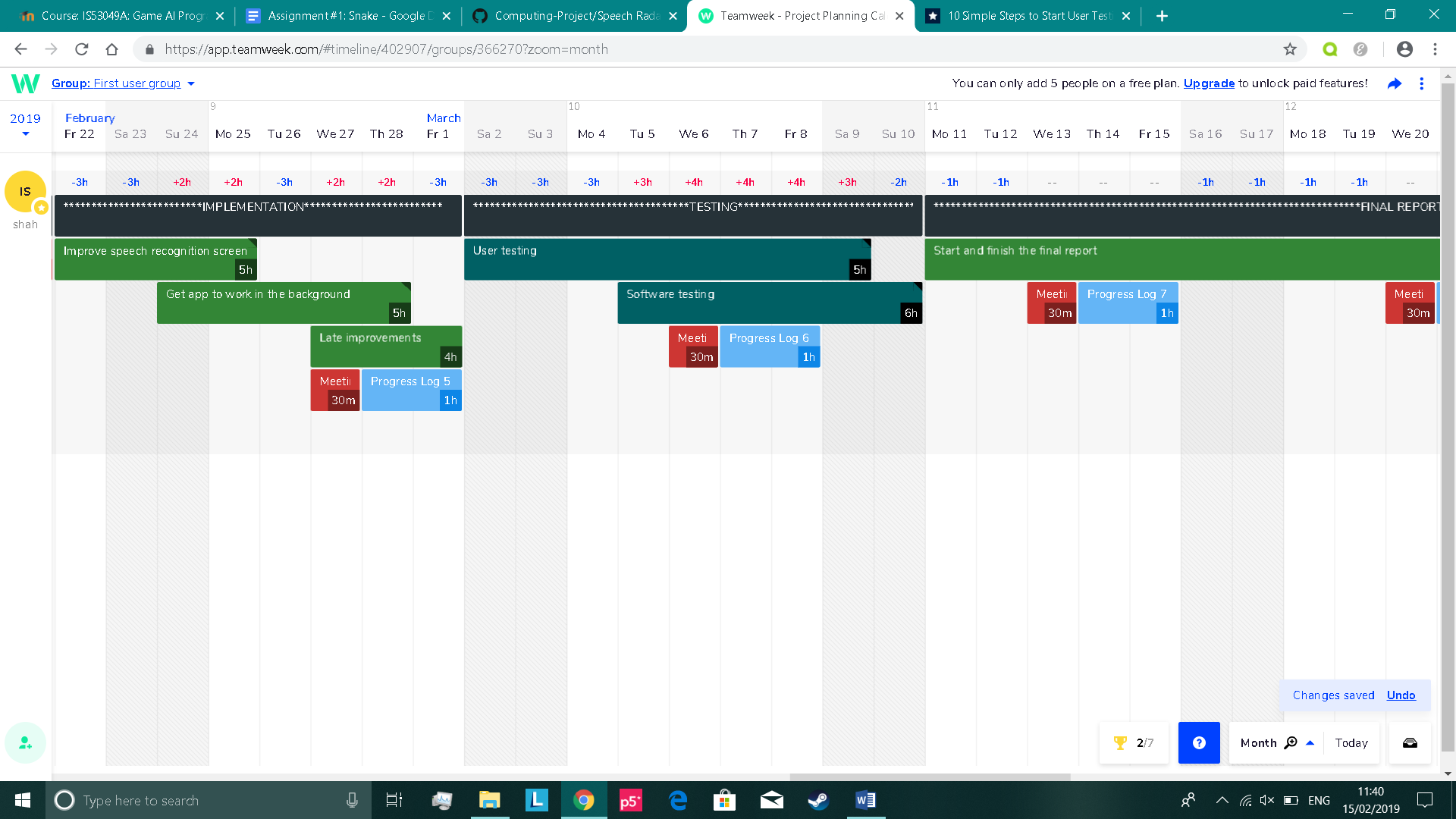
**Implementation, Preliminary Report & Review**

**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

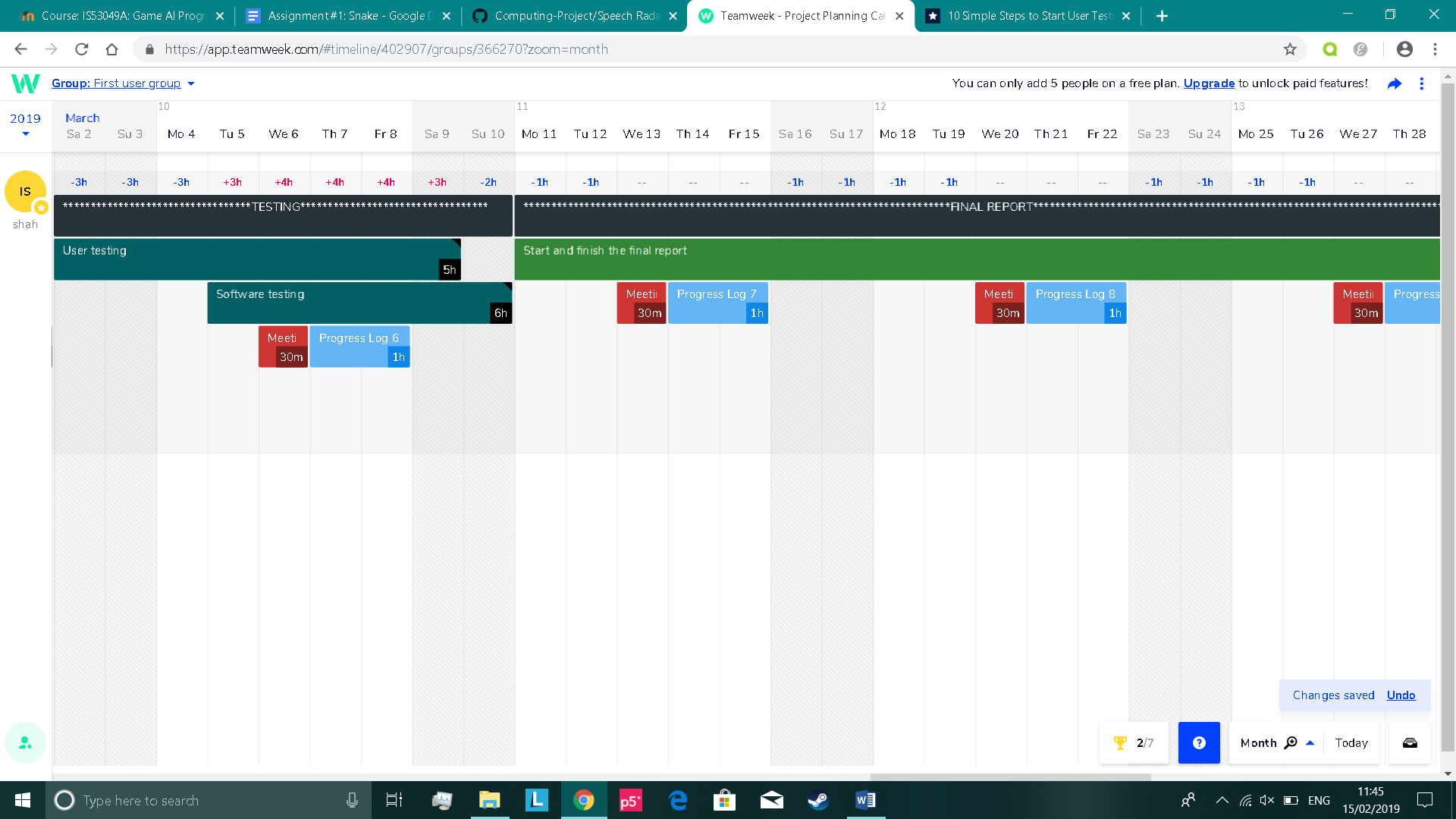
**Implementation (continued…)**

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. Research on how we can get our app to run in the background and to detect voices
2. Once step 1 is complete and is feasible, implement it
3. If step 2 was successful, then look for a way of preventing the app from draining too much battery power

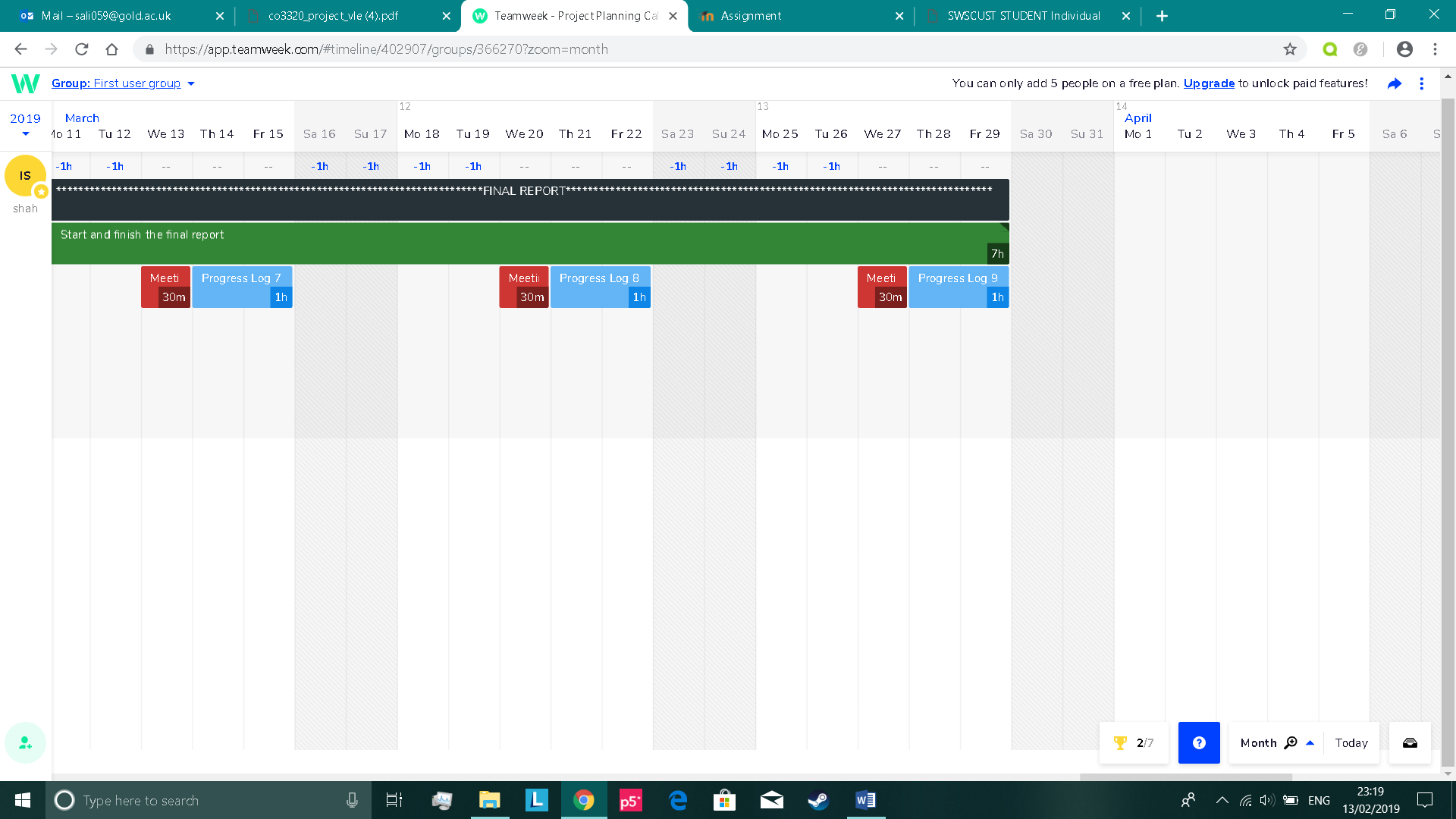
**Testing**

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
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​

**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

**Progress to date**

* I created a class diagram to describe the apps structure
* I created an activity diagram to describe the apps behaviour
* I designed the app using wireframes
* I devised a broad plan of how to implement the speech recognition screen on Android Studio
* I researched on possible datasets I could use to help build my model on Tensorflow
* I researched on the possibility of accent detection and whether it will be feasible for me to handle
* Mention literature reviewed
* I finished the compliance statement
* I finished 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 permitted to verify via email for authentication. I started my implementation at an early stage in the attempt to gain more time in which I can spend for implementing the speech recognition screen
* I finished my implementation of the login screen, where the input fields check whether the account exists, the information inputted is incorrect and if the email address has been verified
* 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

Planned work

* Based on what I mentioned in the previous section, I finished my design and research to help in the implementation phase. Now I have created a quarter of my final application on Android Studio and am increasing the level of work I do 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 some time. If this all goes to plan, then we should have enough time for the final report at the end.

<https://www.reddit.com/r/androiddev/comments/r7zea/voice_recognition_background_service_possible/>

<https://www.developer.com/ws/android/programming/exploring-the-android-speech-api-for-voice-recognition.html>

**EXTRA NOTES**

* Make sure to mention that it makes sense to work with a small set of words, as our aim is to have users register code words, not just any other words like for example “the.”