**4th Year**

**CS322 Music Programming Group Report**

**SoundRite**

**Student no.:** 14339511

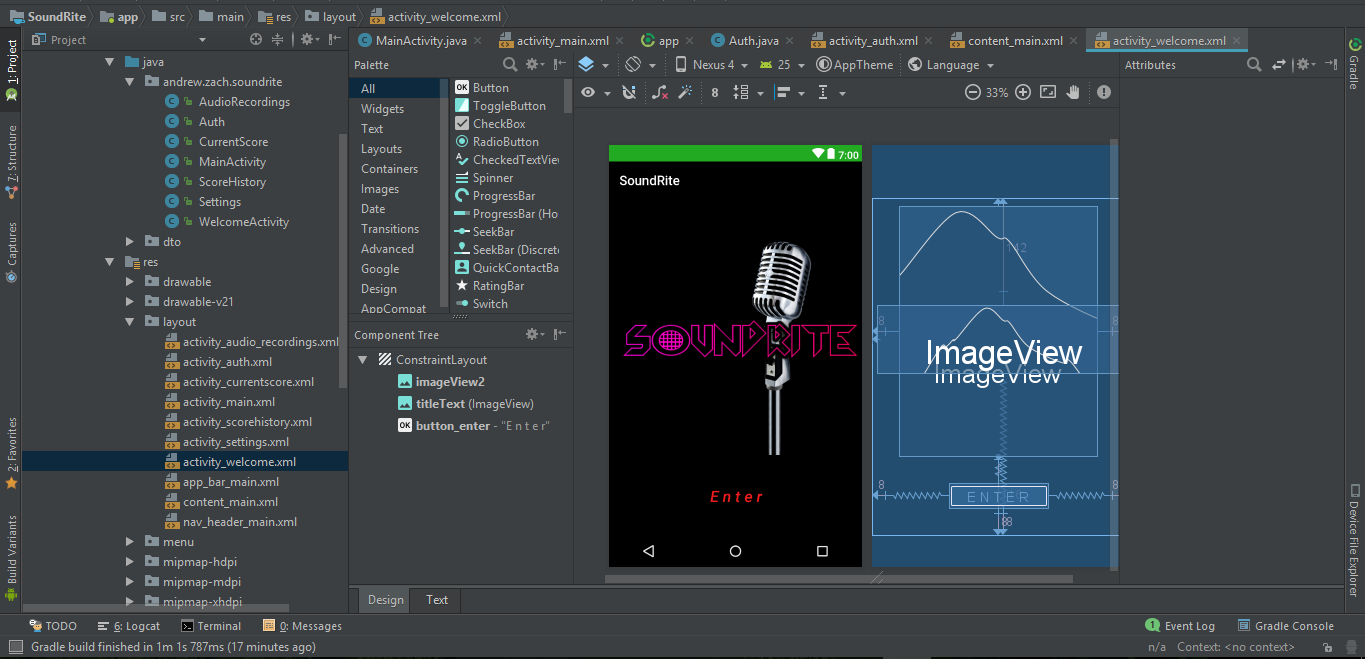
**Student name:** Adanim Zachariah Andrew

**Student no.:** 14545163

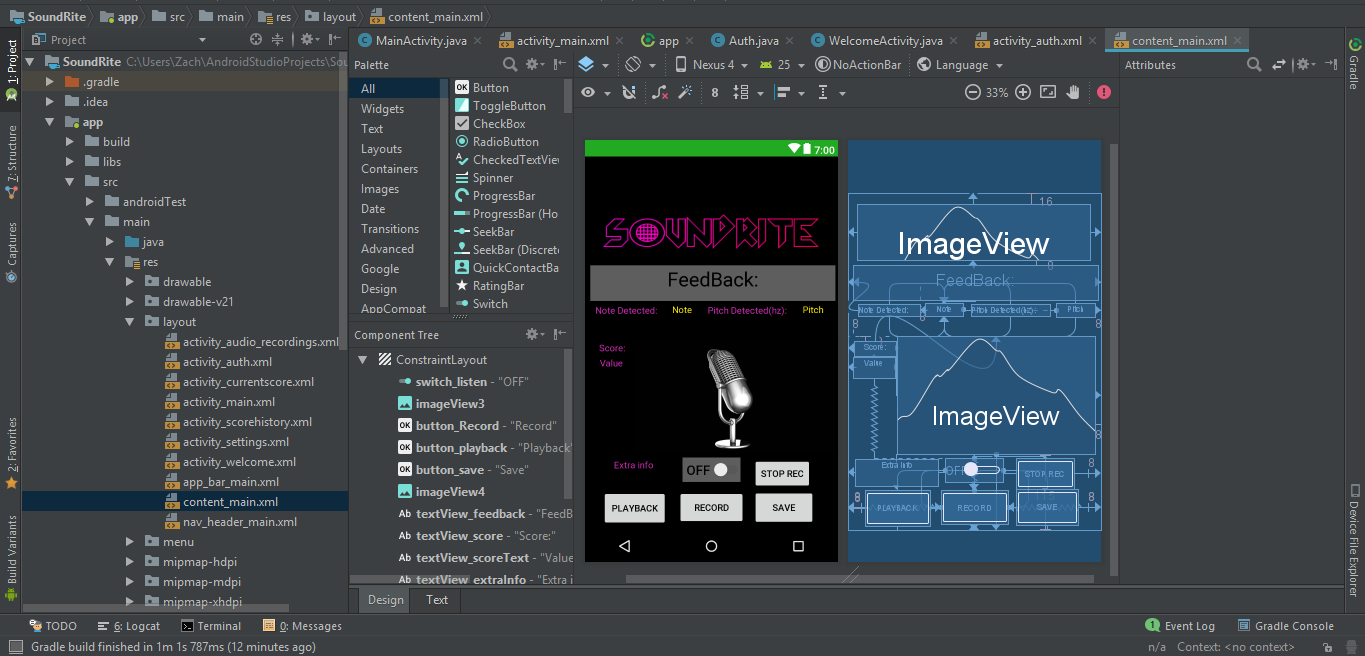
**Student name:** Oghenemaro Okoro

**SoundRite App**

SoundRite is an Android app designed to identify musicality by note, frequency and tone of the frequencies detected within a real-world environment. Users can record their singing voices, playback the recordings and save their recorded audio onto a real-time database.



Users can create their own personal accounts then access the app by logging into it using their e-mail and password. Once created, their user account data will be stored into the connected Firebase Database.



SoundRite is intended to be used not only for singing, but for basic tonality and pitch recognition for anything that is able to produce sound that is detectable by the mic (or connected external mic) into any Android phone. It can be used by singers who wish to have a quick tone check before they start singing onstage to make sure they are about to sing in the correct key. It can be used to identify musical notes played by an instrument such as the piano for example.

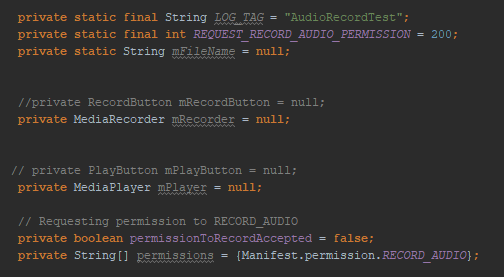
Using SoundRite will ensure that users can correctly detect the right key, notes and have a better understanding of frequencies in relation to music, to help train their ear to notice what notes they are hearing the everyday world of today. This app is very useful for people with an interest in detecting basic musical notes anywhere, at any time in a real-life environment.

**Pitch/Tone Recognition**

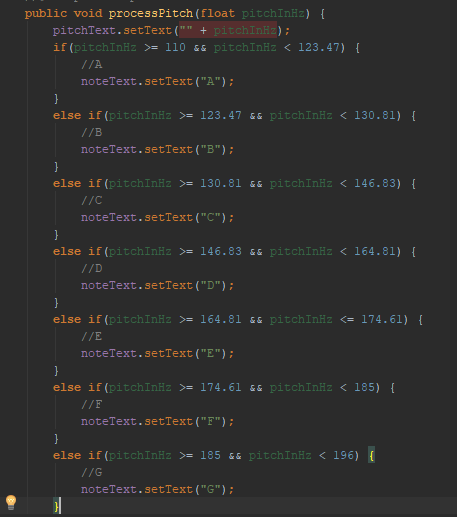
For pitch and tone recognition, we used the TarsusDSP.jar file to handle frequency recognition, allocation and manipulation within our programming application.



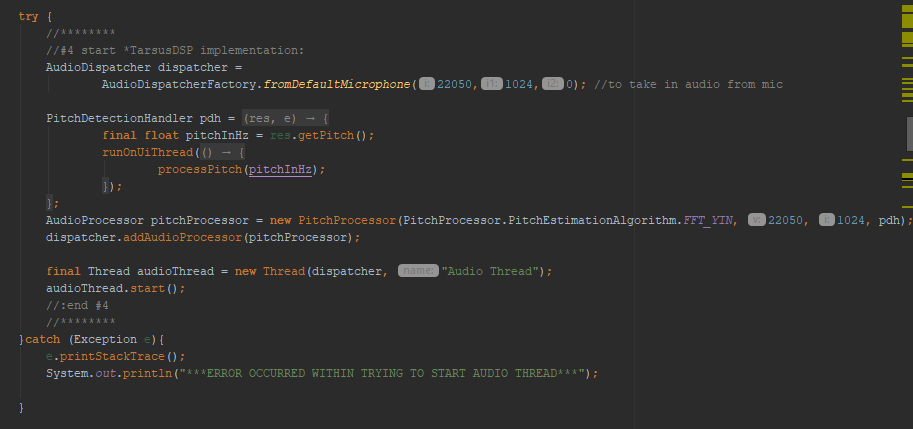
By activating the pitch recognition on playback, the audio received by the mic on the android phone (or connected by an external mic) receives this audio, analyses the frequency in (hz) and displays the data produced by the algorithm we coded to the user.



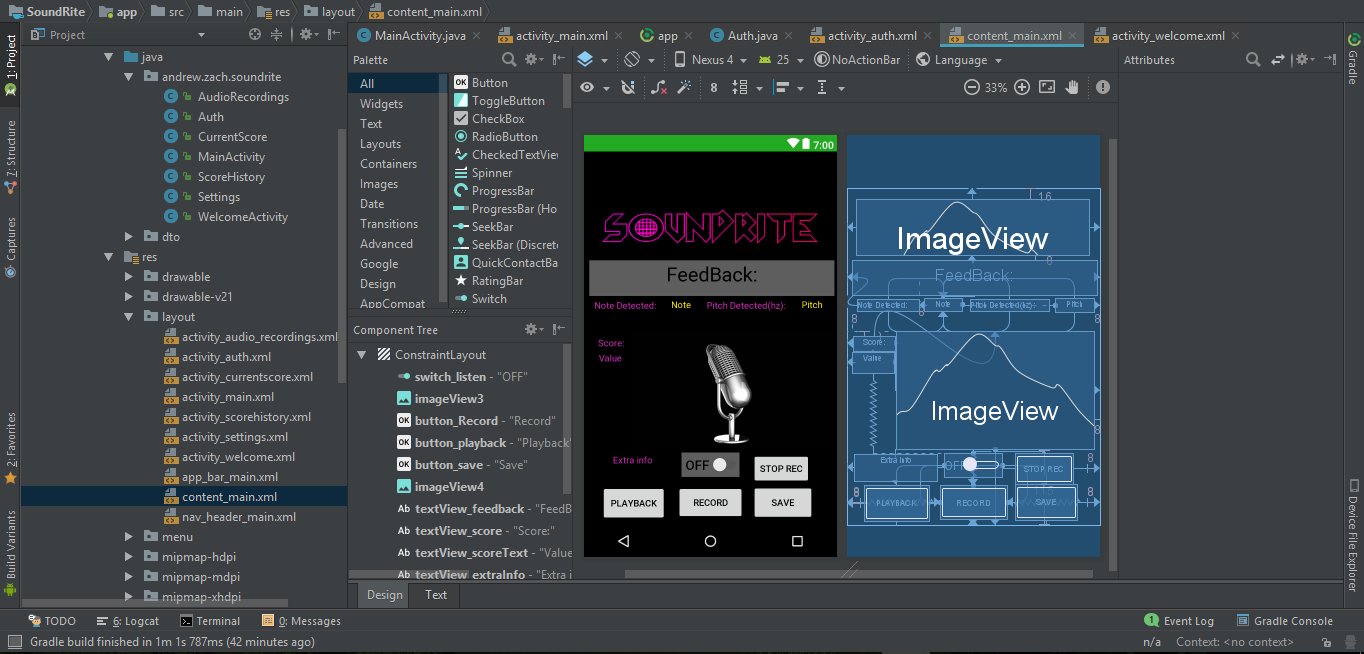
The mic is active in real-time after playback activation for the user to use in detecting any frequency, returning the tone value and returns the frequency detected in the real-time environment. This functionality is made available to the user to allow for easy to use note detection in any environment that is producing some sort of frequency (ideally musical).



Pitch and tone recognition in real-time is the core function of SoundRite. It allows users to acquire tone and musical note data with ease and efficiency. The mic will continue to detect audio frequency within the environment continuously, which is great for singers who wish to sequential notes and validate their accuracy quickly by using this app.

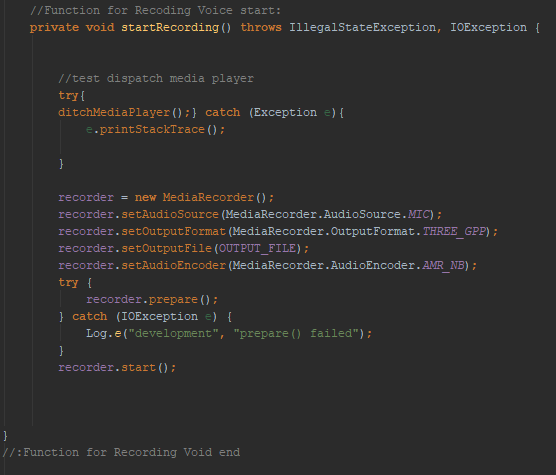


The information feed-back box is also in real-time in accordance with the audio frequencies detected by the mic on playback, this allows instantaneous nots and frequency recognition and musical self-evaluation for the user’s convenience.



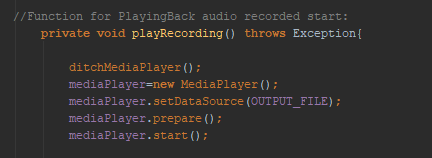
**Record and Playback functions**

The record and playback features of our app is a secondary feature which allows the user to sing/record musical notes/ frequencies in the real world, to play back for the use of the user’s convenience.



On playback, the pitch recognition audio thread will take over the mic, preventing further recording. This is done to allow the user to playback the currently recorded audio while the pitch recognition evaluation code works to analyse it for musical pitch and note values.

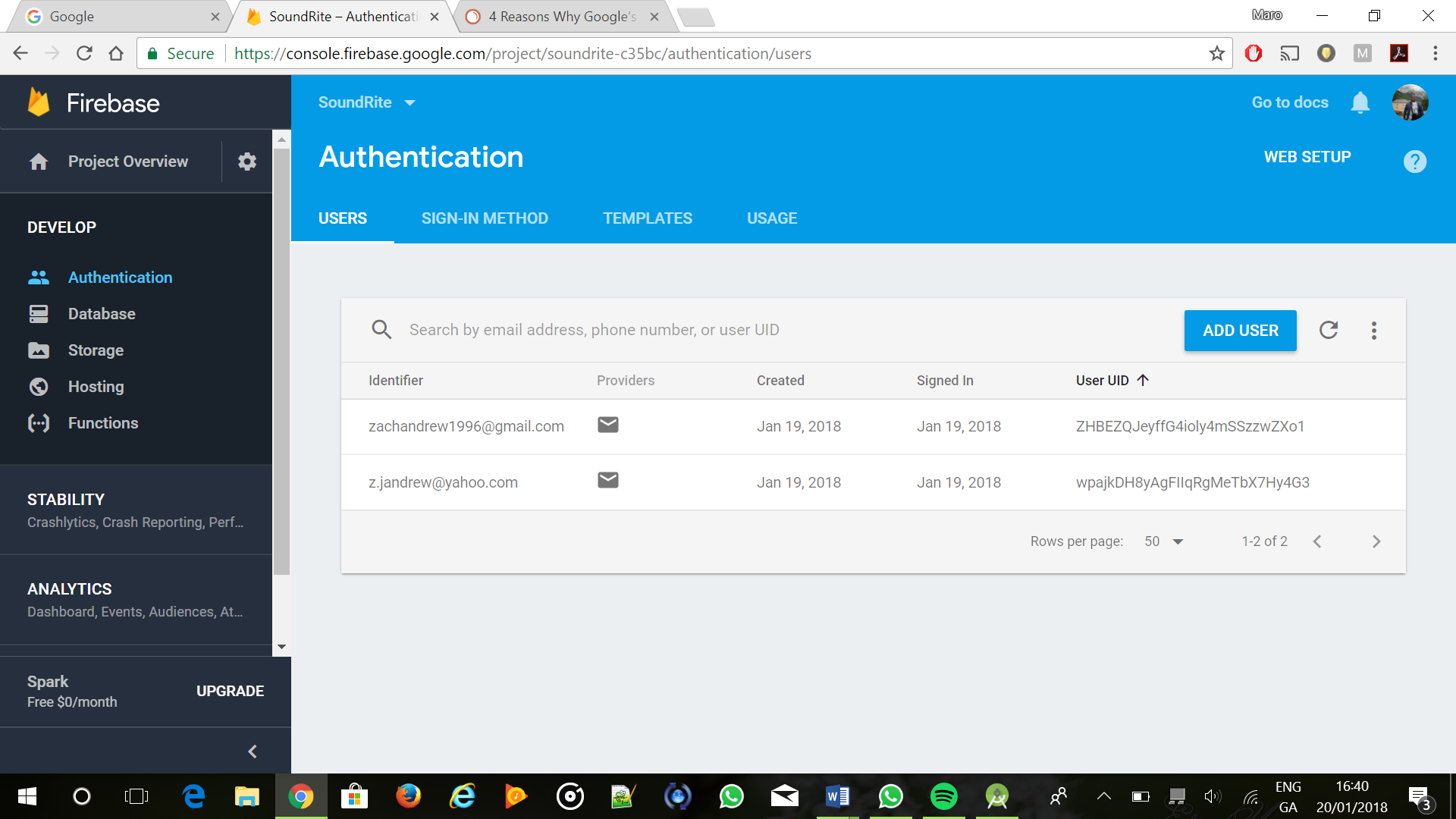
The playback can be repeatedly played until the user is satisfied with the information they have received from the feed-back box in informing them of the musicality of their recorded audio playback.

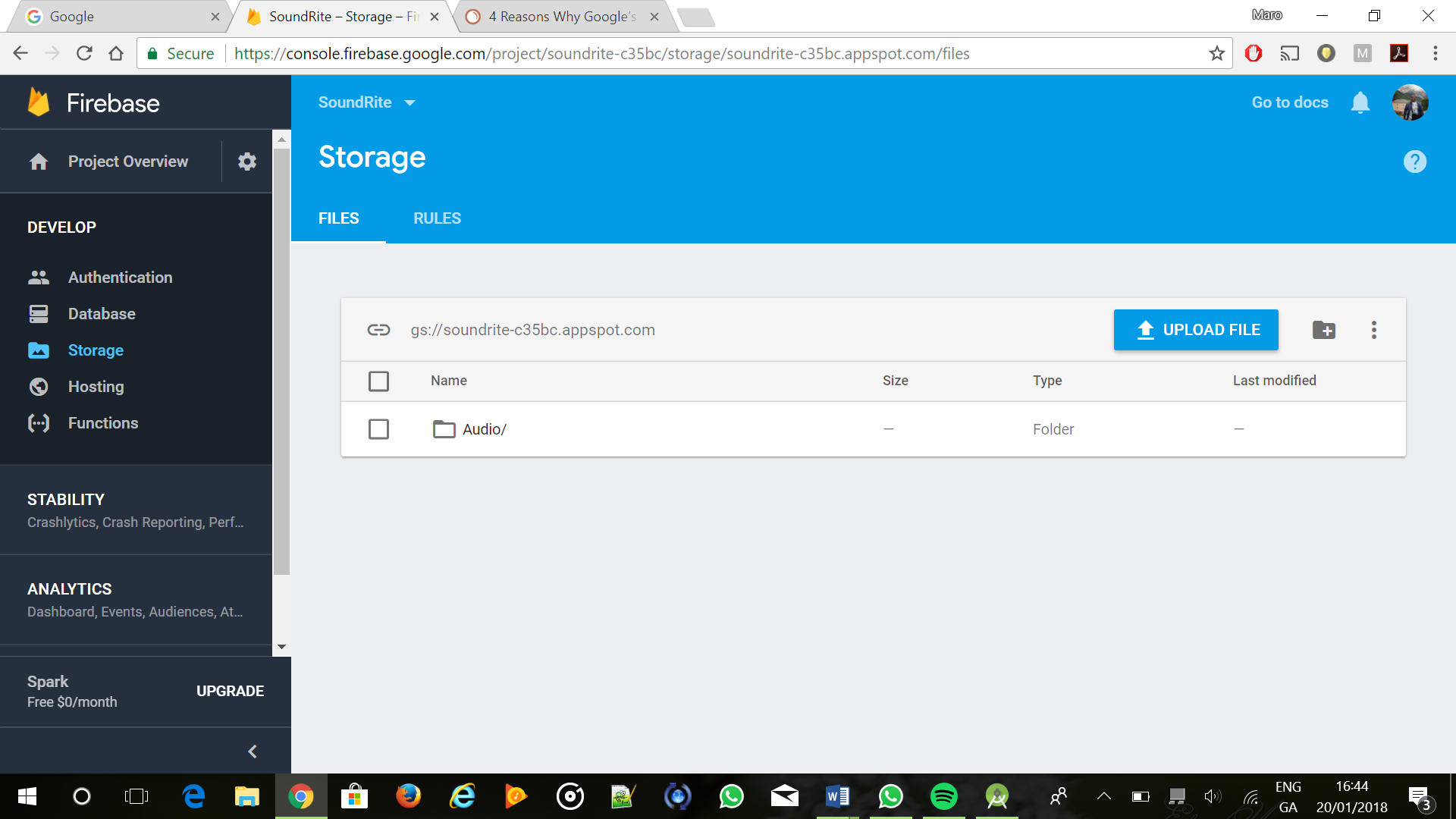


This is useful for singer who wish to playback a melody they have sung to see the notes/frequencies they are hitting for example. For musicians, they could use it as a quick tuner to correct basic notes on a guitar, or identify the note keys being played on a keyboard. This additional feature complements the pitch recognition core feature of the app and creates more options in how a user wishes to analyse their musically obtained environment from the real world.

**Firebase**

We decided to use Firebase to store the users sound flies and to authenticate users into the app. We chose Firebase because it had a lot of documentation in it and was quite easy to use and implement into our app. Using the Firebase Storage functionality, the sound files made were then stored in the Firebase storage bucket. Firebase also has a Firebase authentication functionality. This allowed us to enable what type of verification we wanted for our app and see the users registered for the app

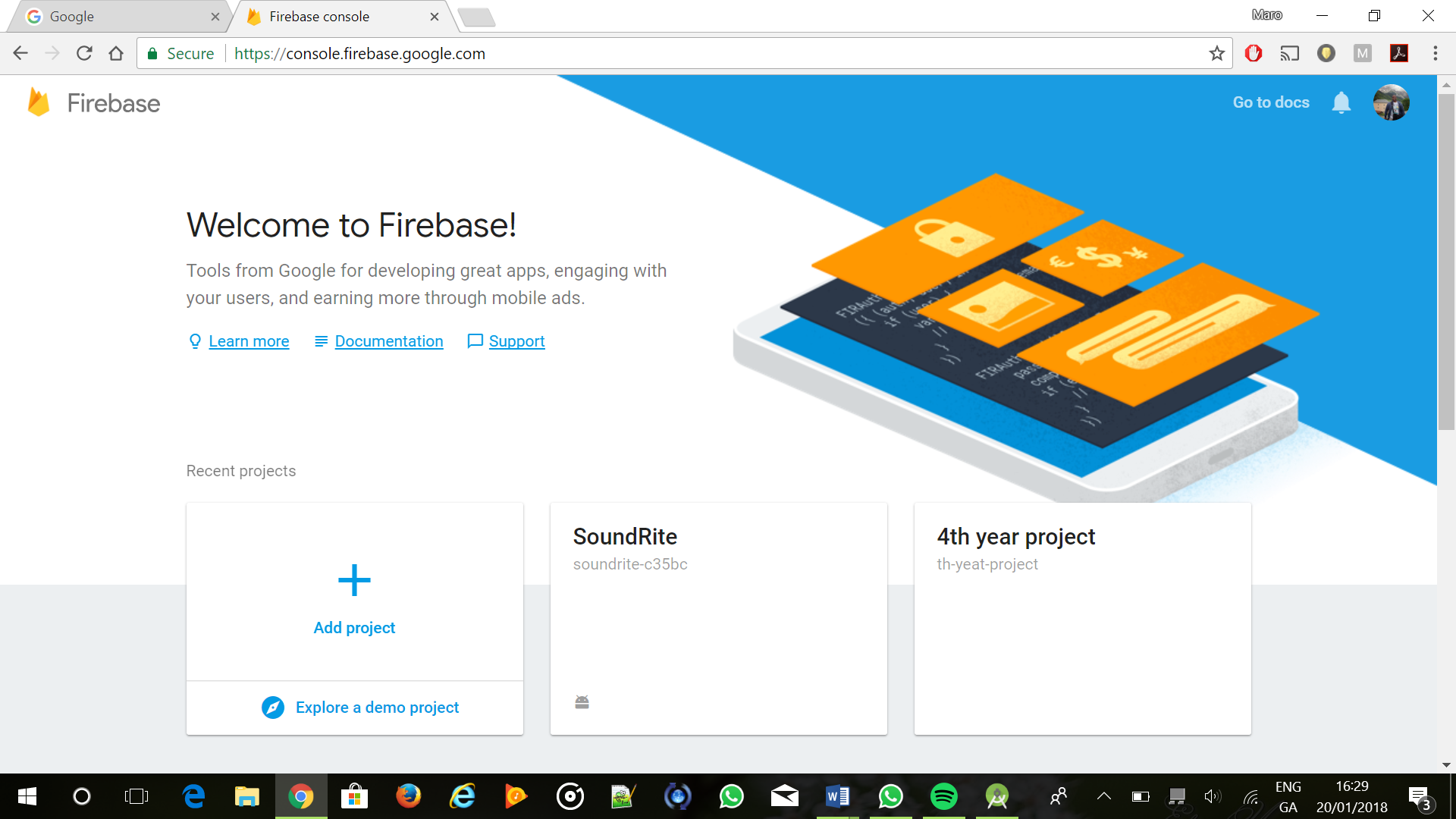




Here are a few more of the features that firebase offers such as a Realtime Database and App Hosting functionality

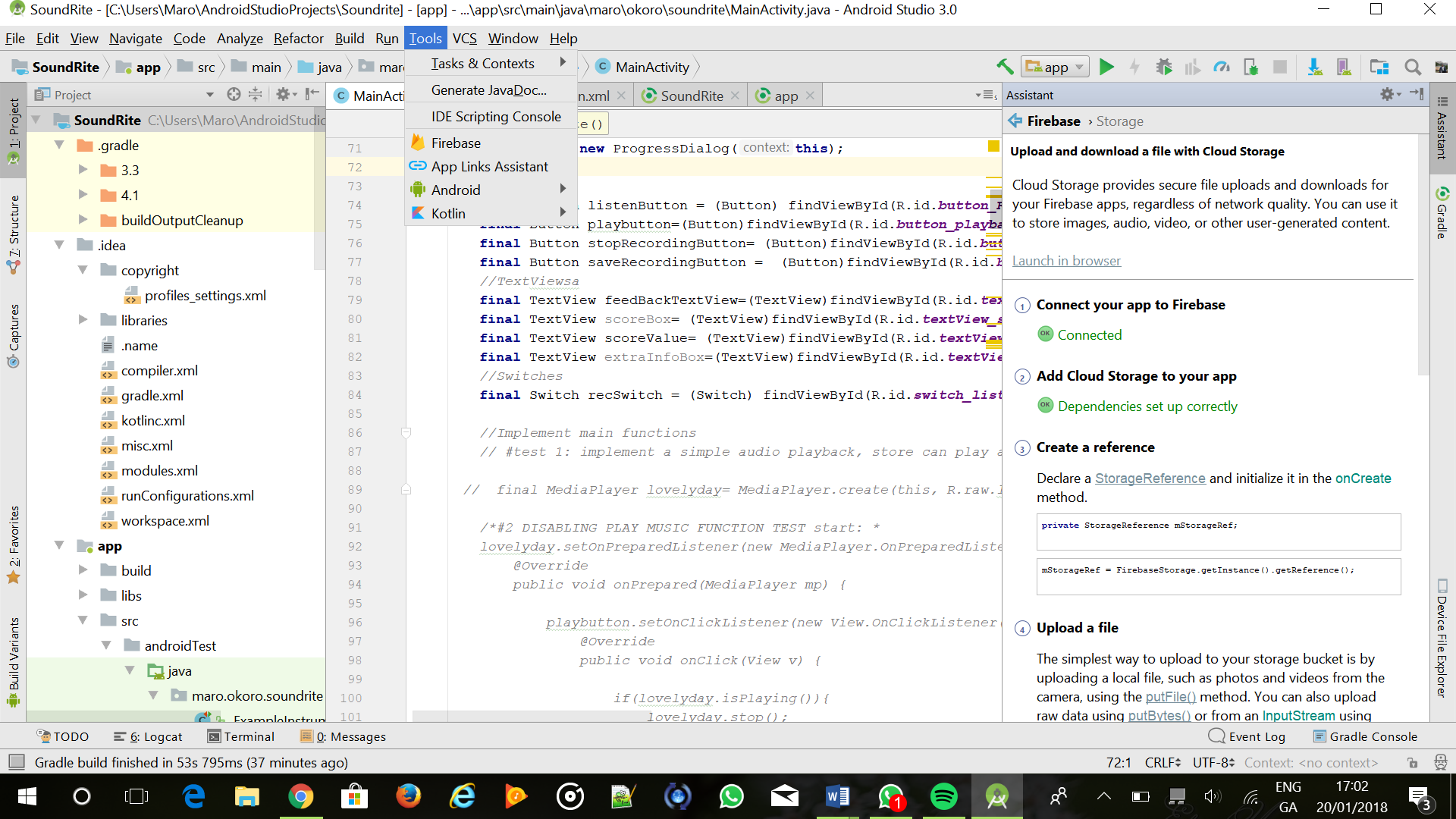


To get started one of us had to have a google account and then the project was created and then the other person was added to the project as a collaborator and then onto storing sound files recorded by the app.

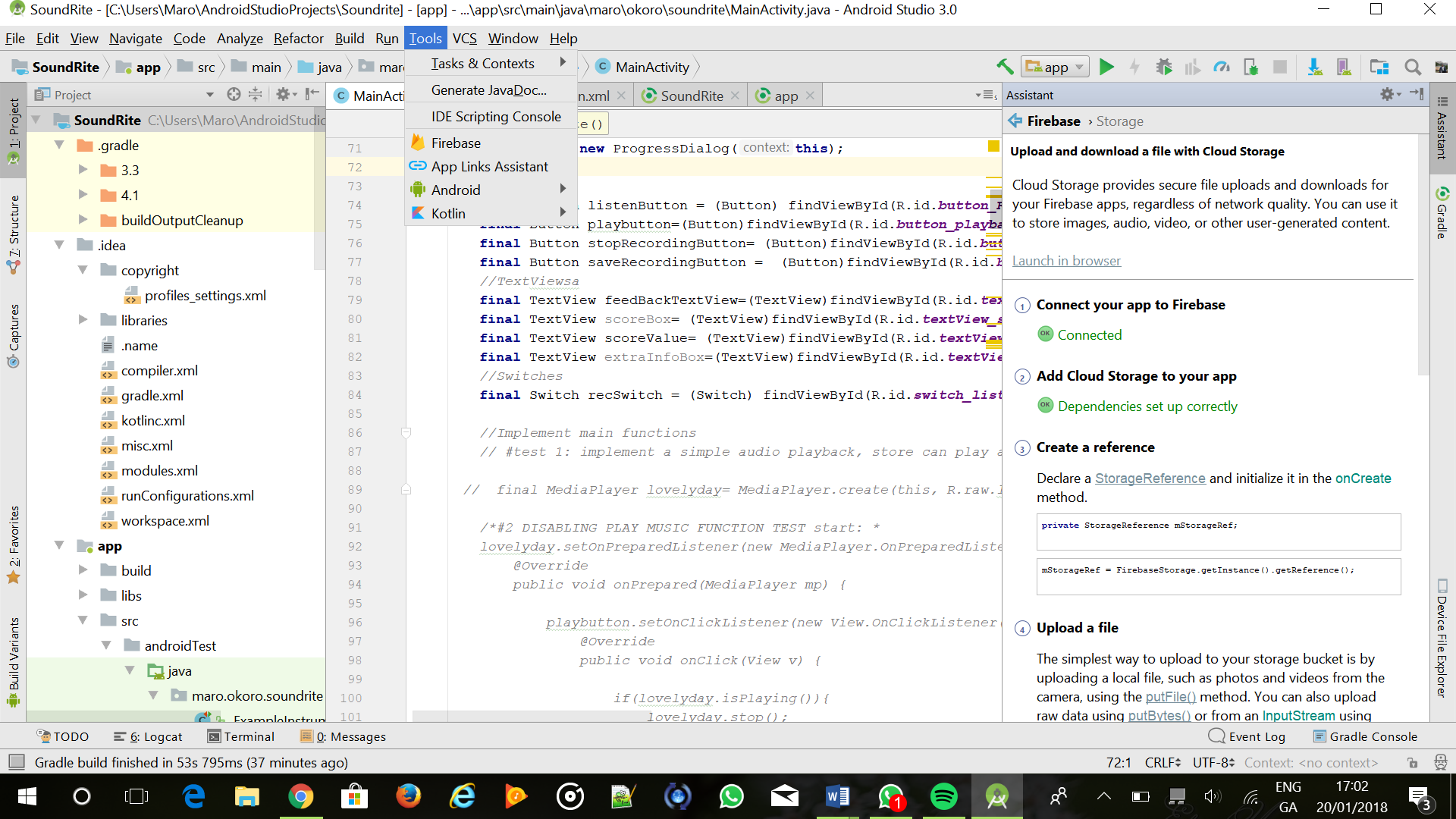


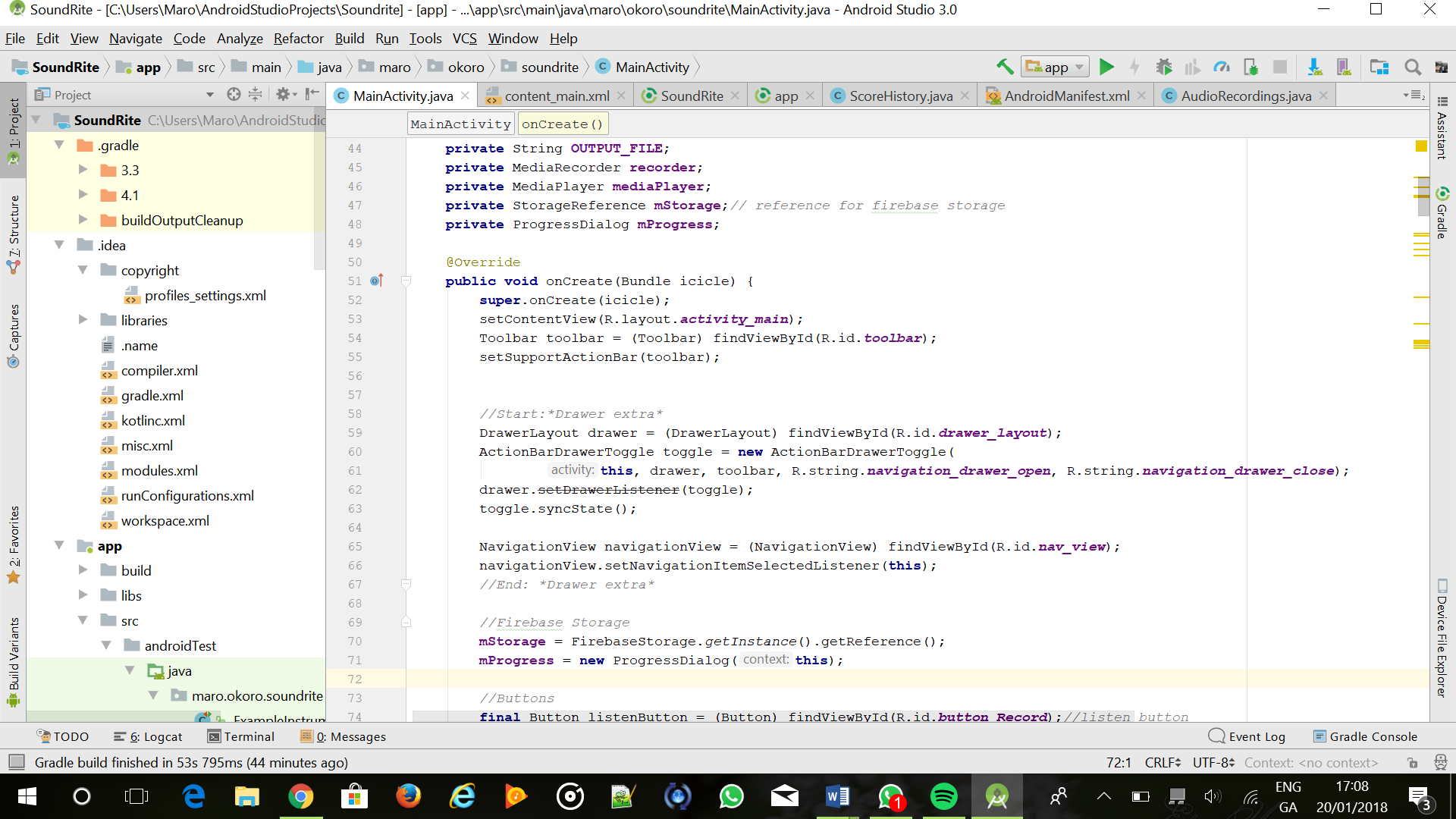
**Firebase Storage**

On Android Studio Firebase was initialised by clicking tools and then Firebase.

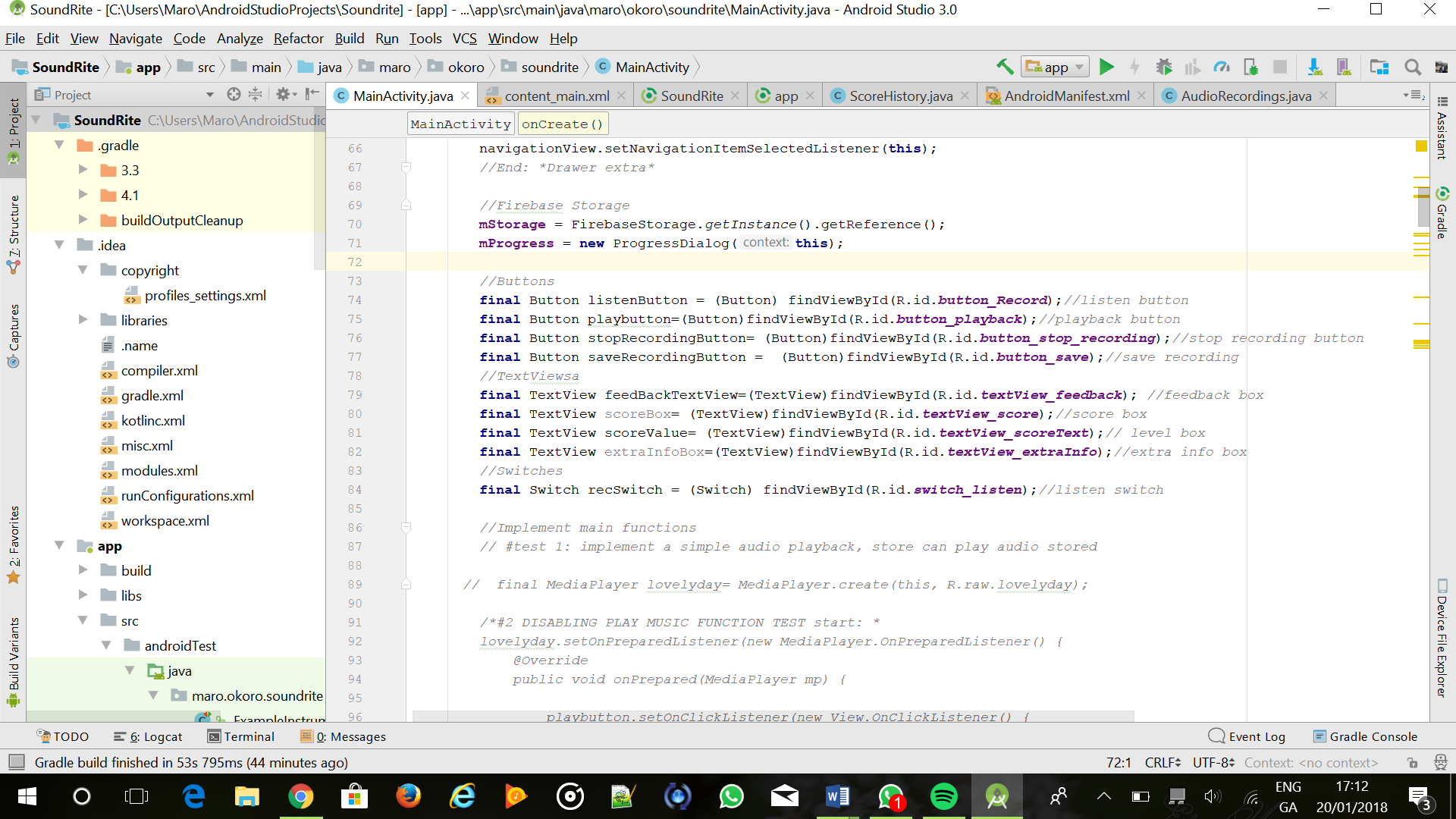


The app was then linked to our firebase project “SoundRite” and the correct dependencies were set up/imported to the app to allow Firebase functionalities.

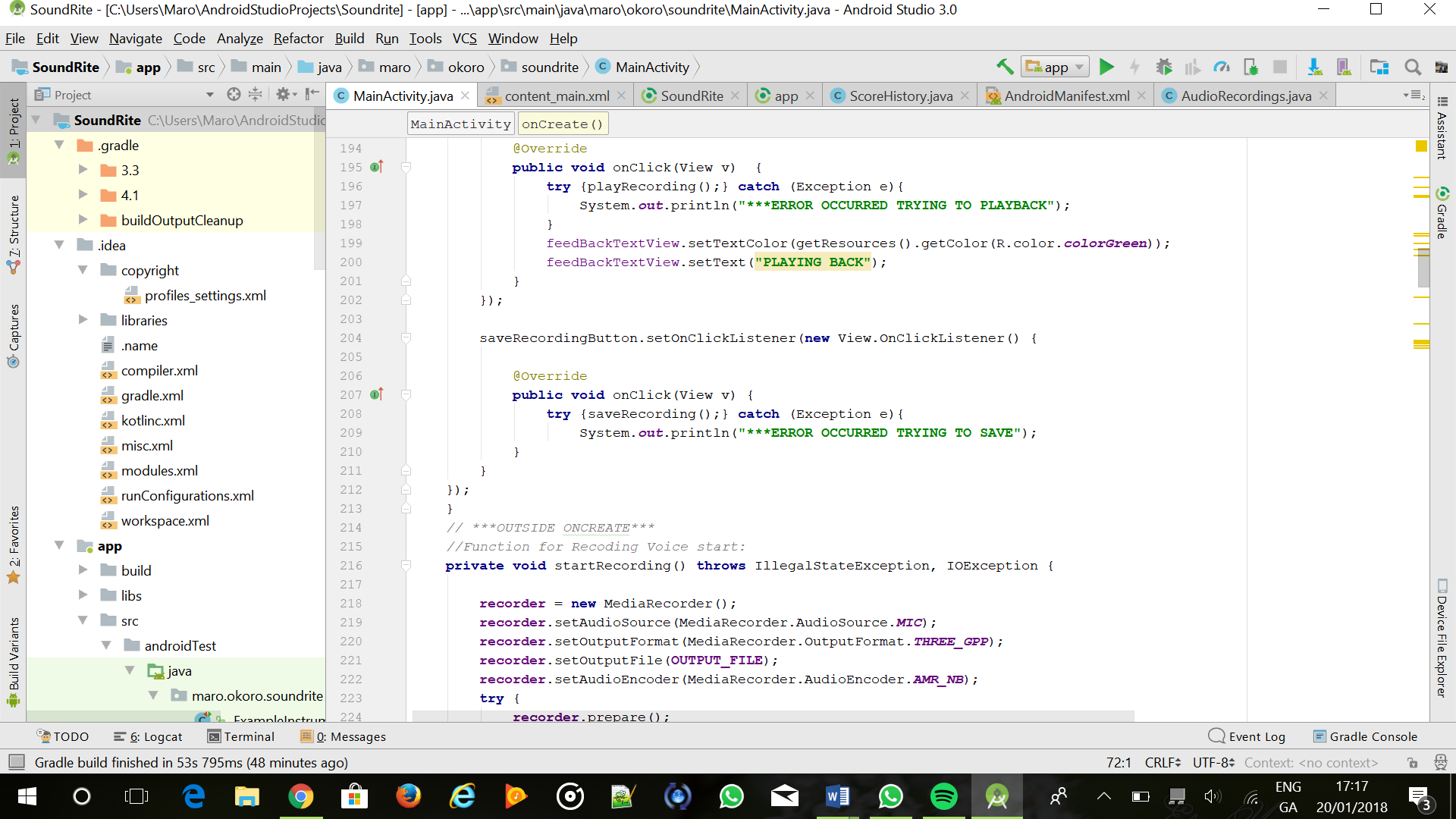




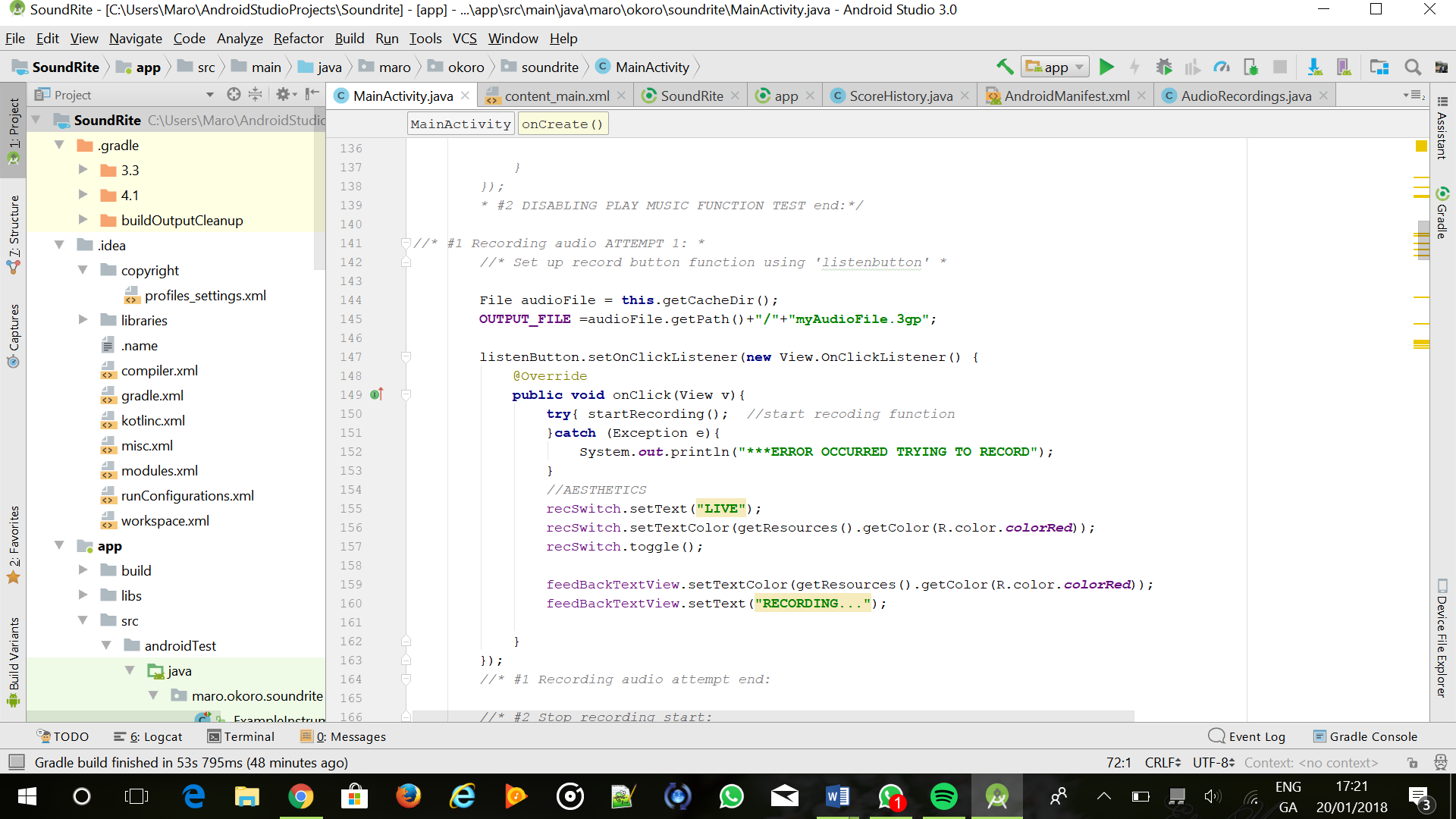
Here we are setting up the reference for our Storage bucket and creating a progress dialog box to show our audio file is uploading.



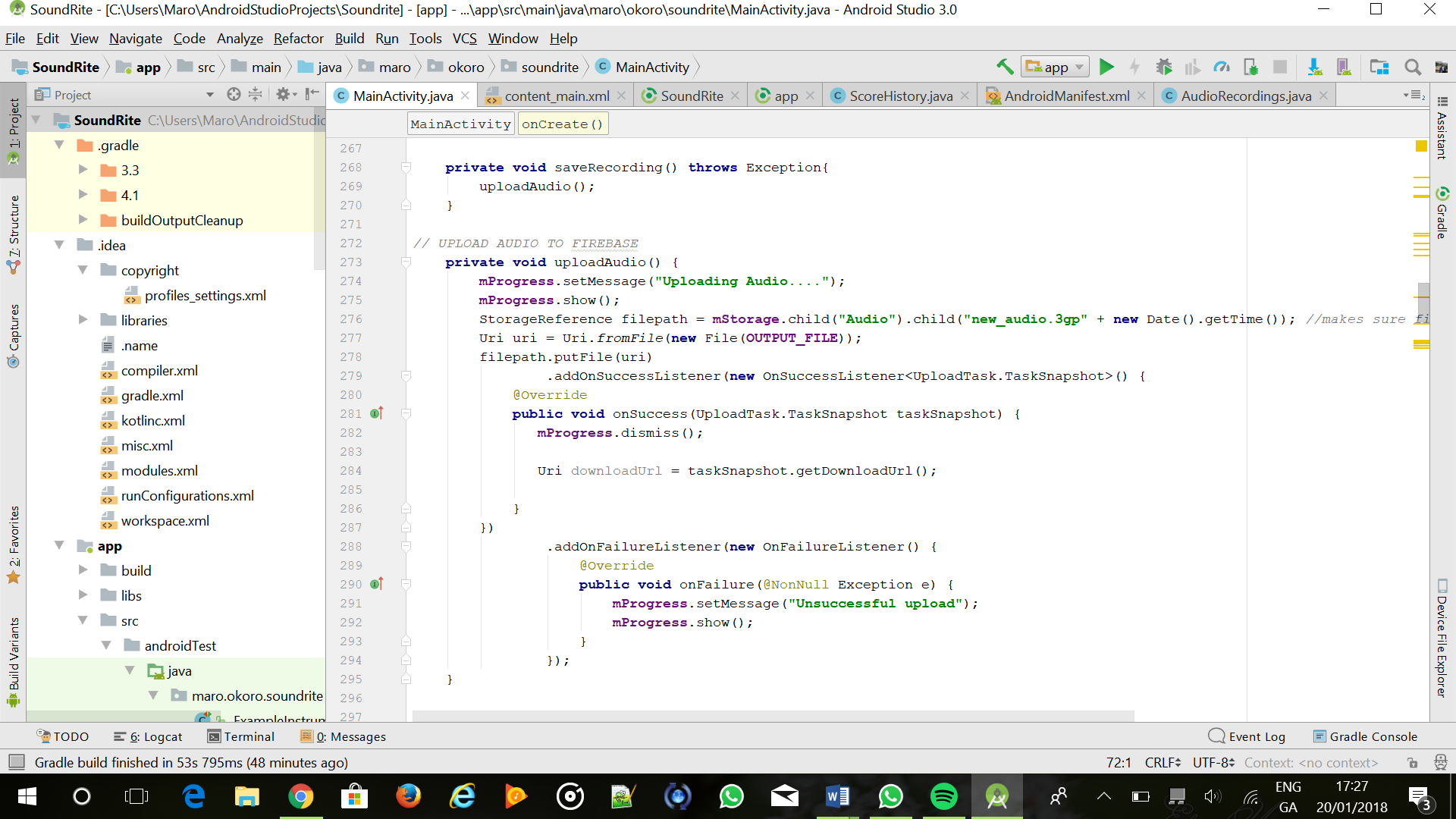
Here we are linking the save button id to the variable “saveRecordingButton” this way we can call the save button easier and specify what it will do



Here an on click listener is applied to the save recording button and within the function we want to try out the function for saving the recording onto Firebase and if it does not work the console will print out saying there was an error when trying to save the audio file.

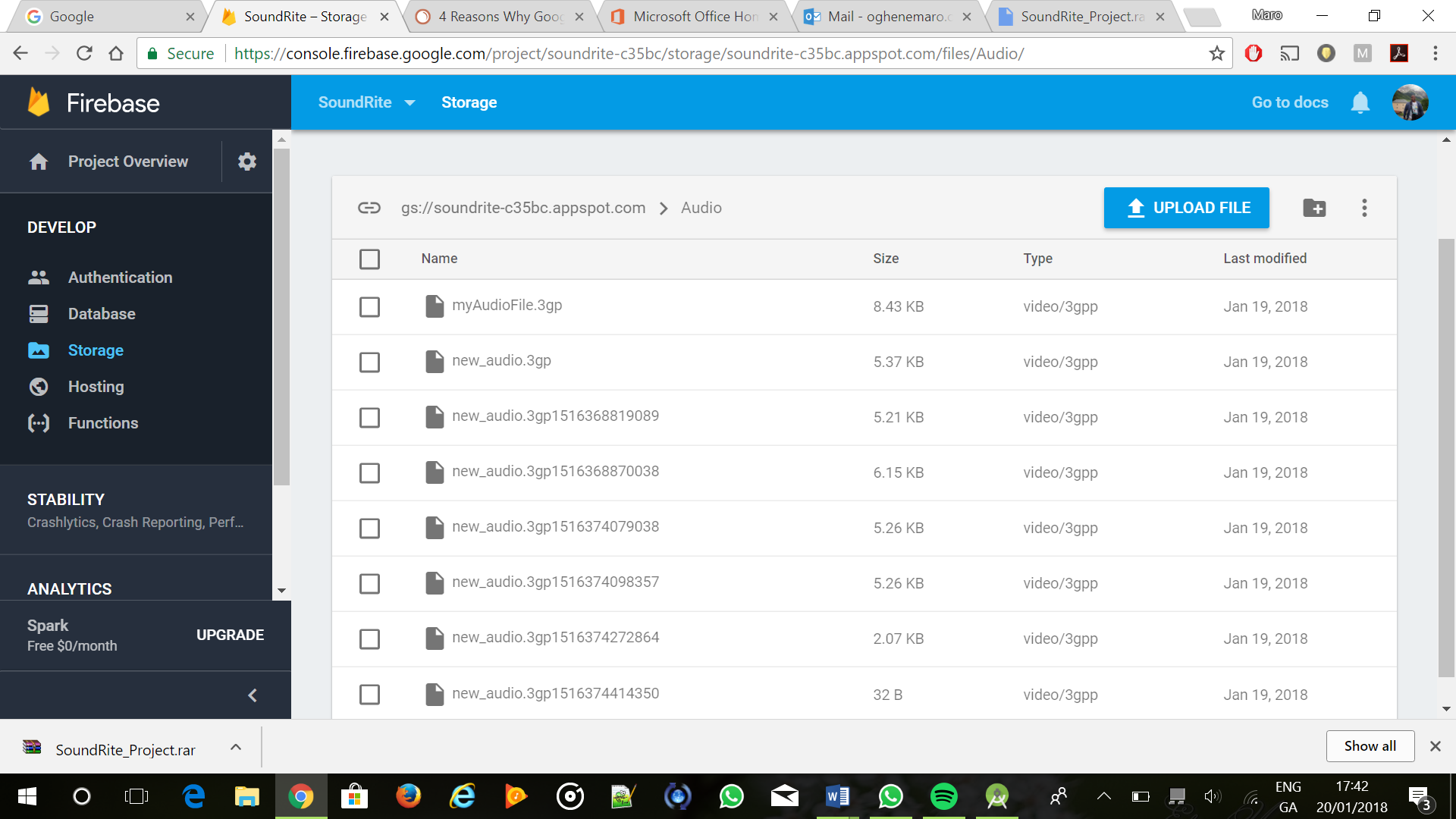


The audio files are being stored in the phones cache directory at first and they’re saved with the variable “OUTPUT\_FILE” and this is what we will upload onto Firebase.



The “saveRecording ()” function then calls the “uploadAudio ()” function. Here we set the progress message to show uploading audio when the sound file is uploaded. We are using our firebase storage reference to store the recorded audio onto a file called Audio (which we’ve created).

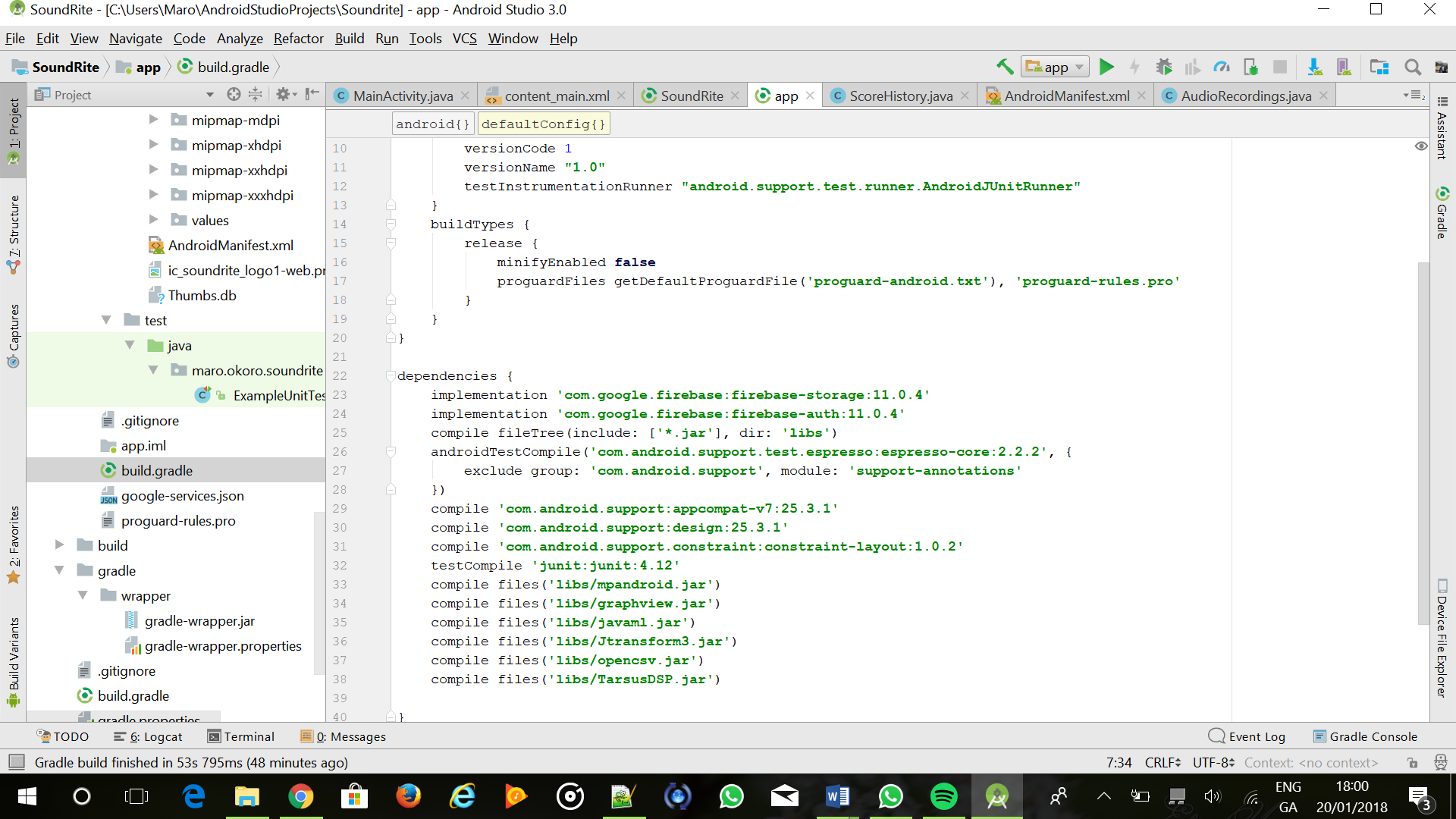
On Firebase, the sound files can be seen in the console.



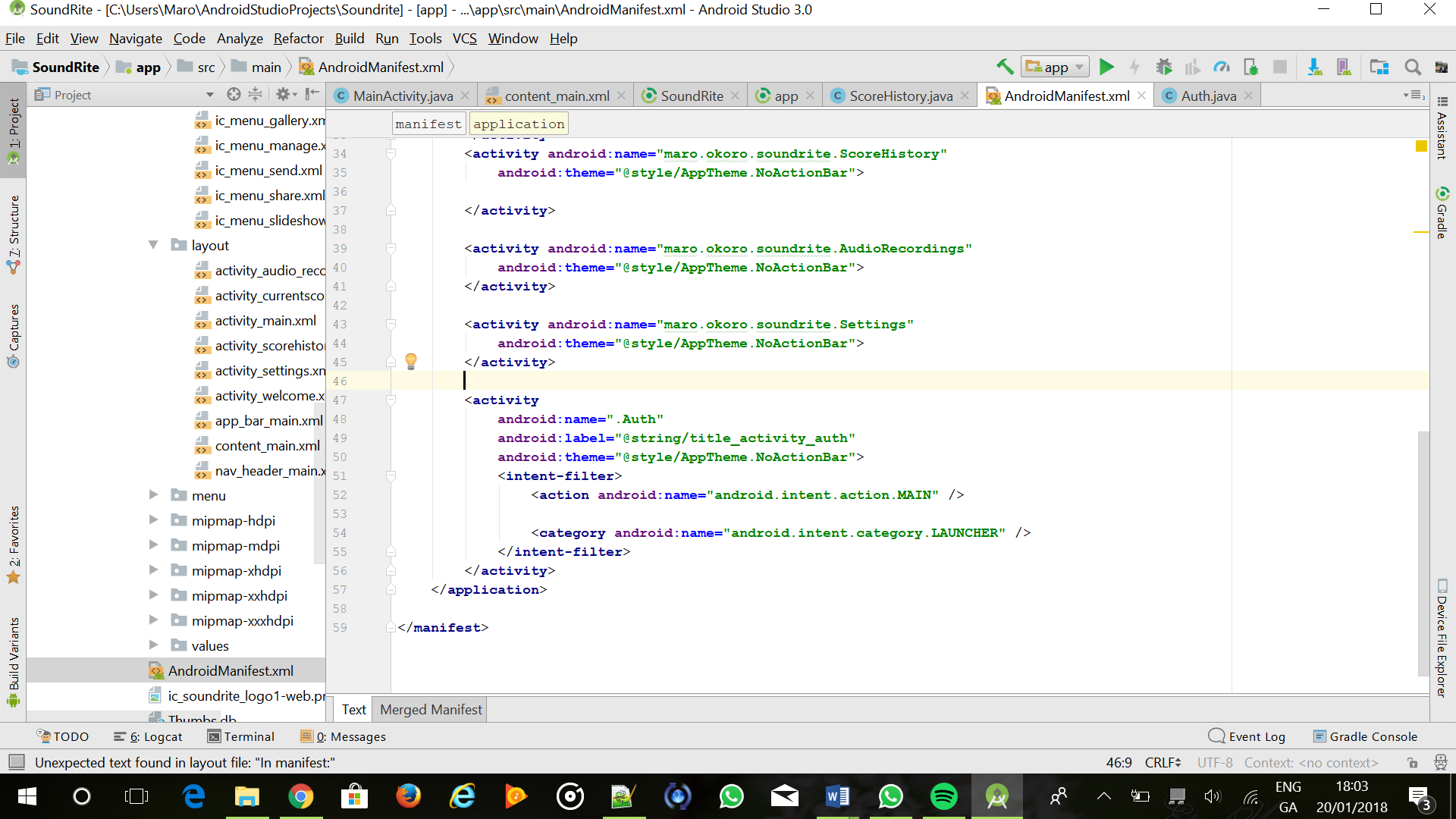
To make sure when you record a new sound file it doesn’t overwrite the previous one, a time stamp is added to differentiate. The on-success listener removes the progress window and the on-failure listener will make the progress window show an “Unsuccessful upload”.

**Firebase Authentication**

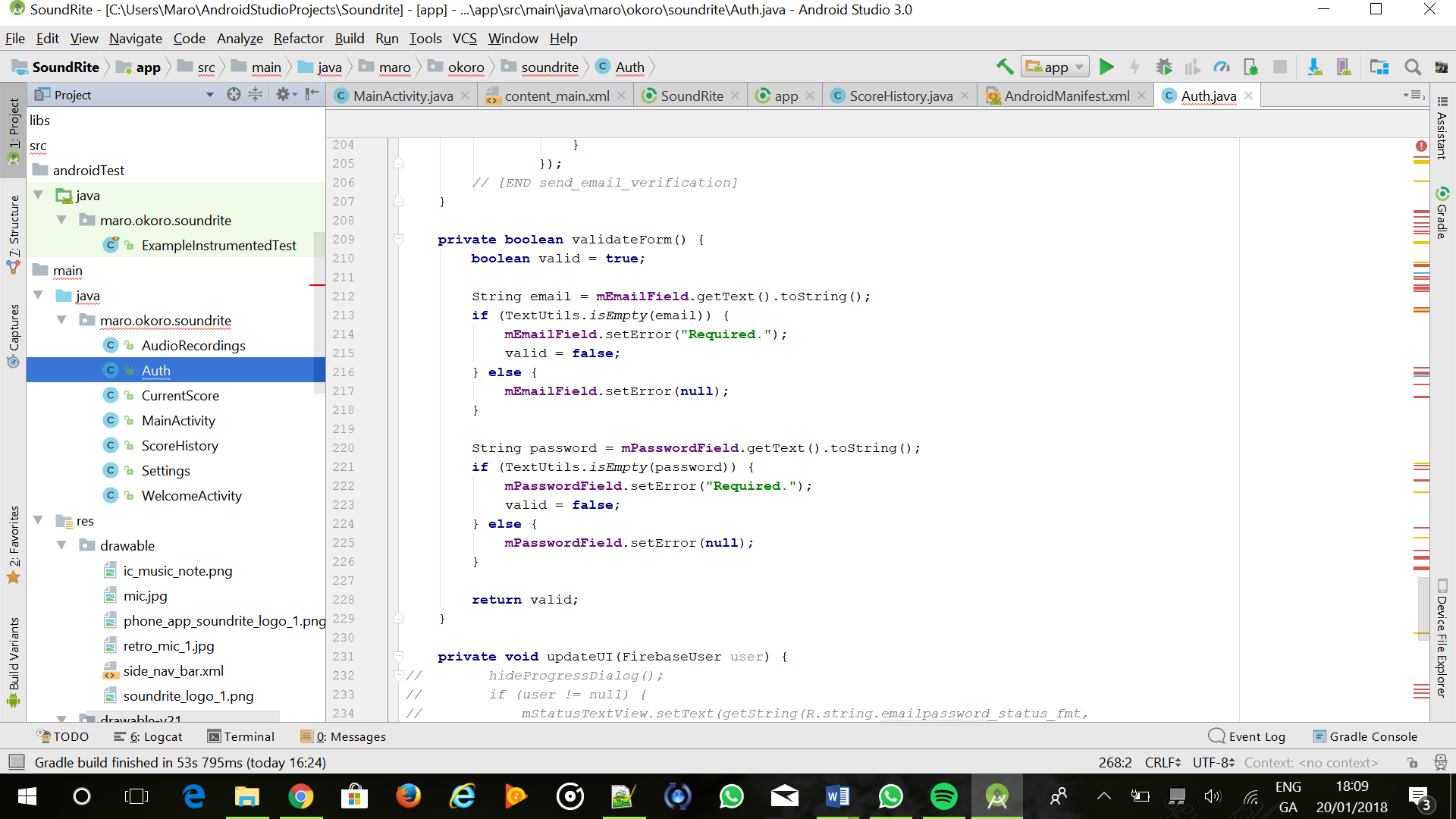
For Authentication, we first started with making sure we had the dependencies were added when we had connected to Firebase when working with the Storage functionality. In the gradle file it was shown like so



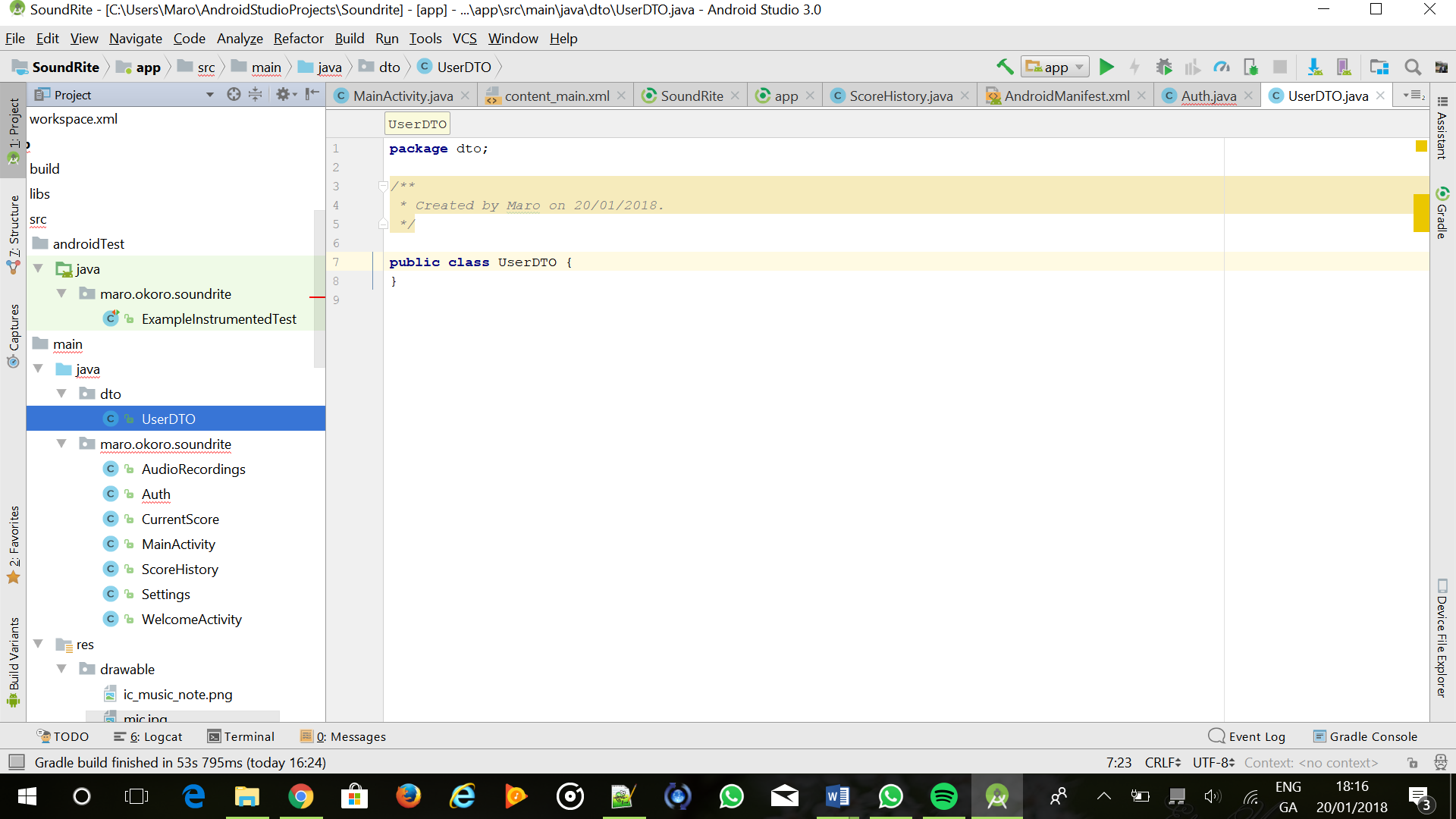
In the android manifest this code was added to bring up the authentication page



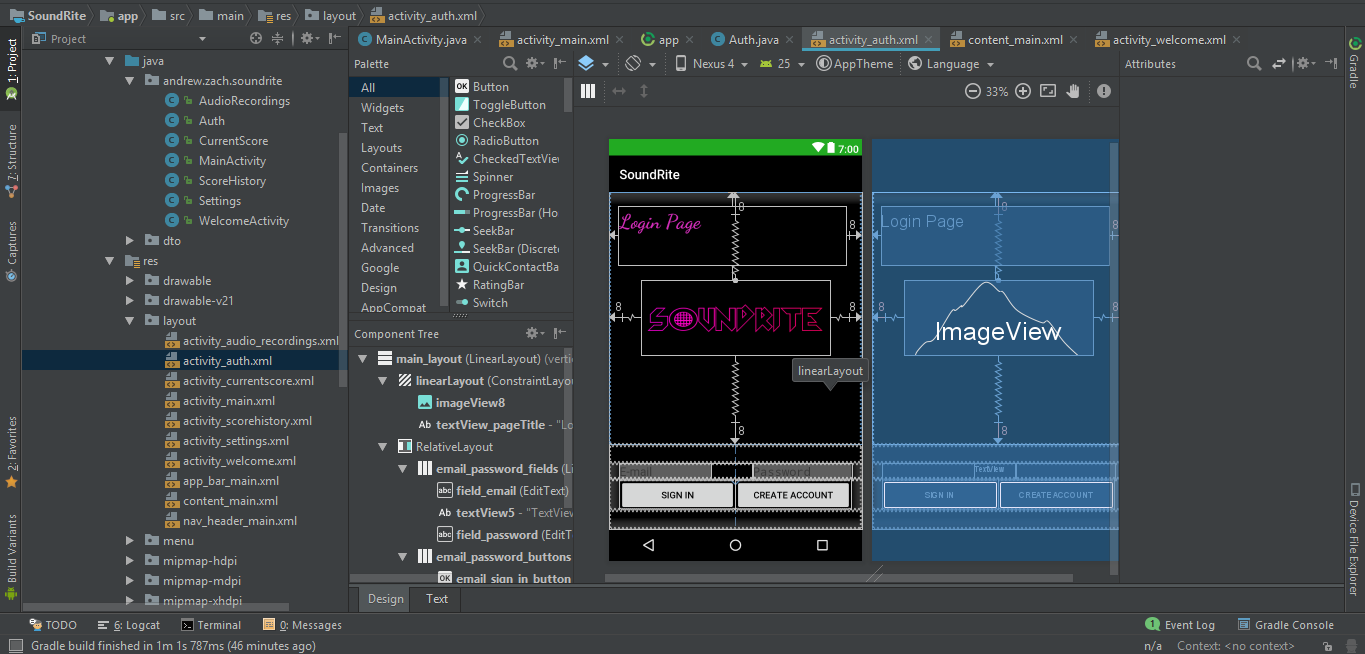
The Auth Java class was then created.



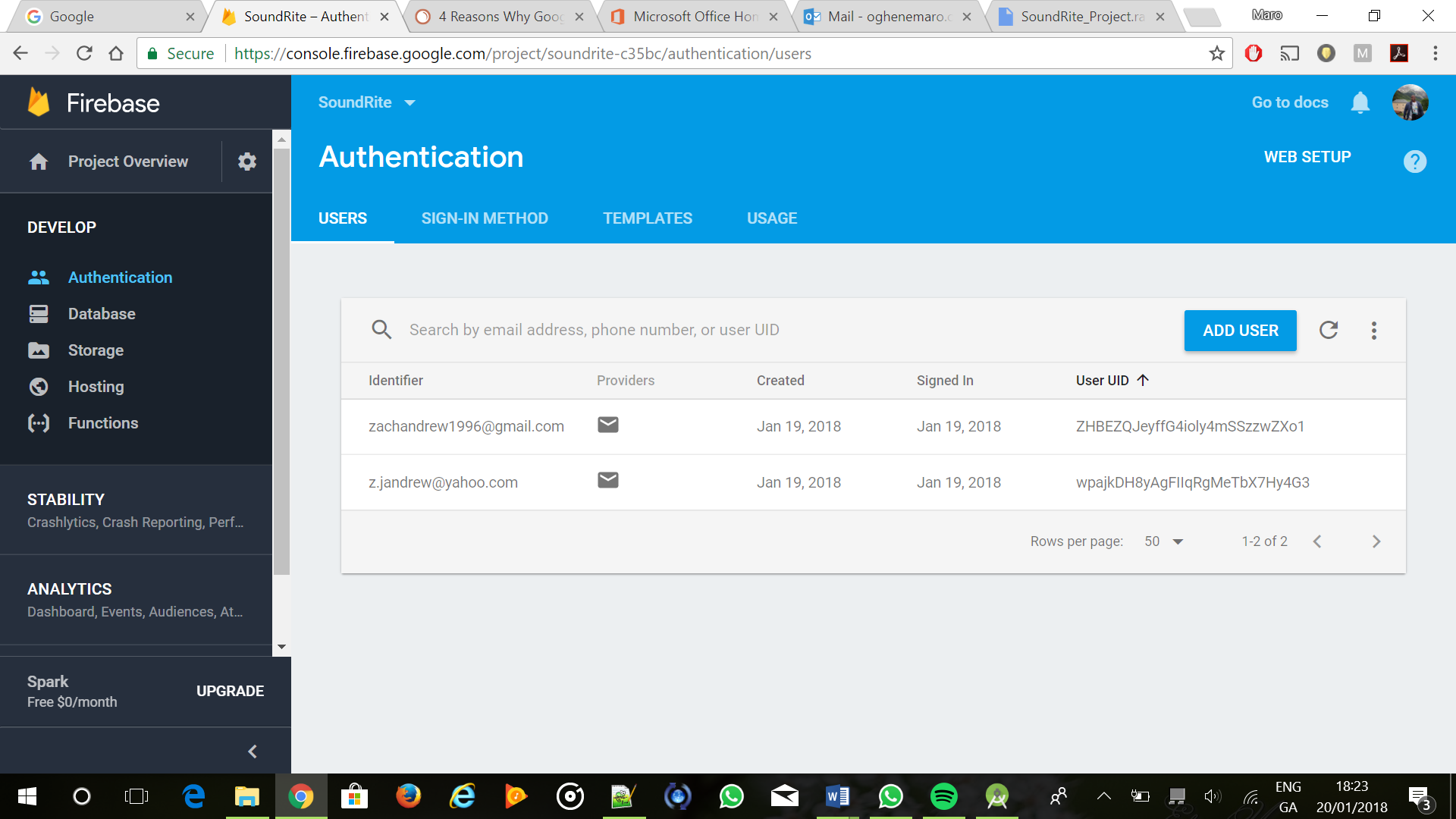
The code is quite explanatory from the comments in which functions are doing what and how it works but the most important part to note is the UserDTO.java file.



It is a user data transfer object called dto. The UserDTO.java file was created in the package dto. UserDTO.java contains getter and setters to allow the new users info to be saved into the database (There are comments in the code that go even deeper into how the code works)



The xml files content\_auth.xml and activity\_auth.xml were also created for the layout of the sign-up page.



So now when a user enters the app they will register with their email and create a new password and form the Firebase console we can see all those users.