A SECURE SCREEN AND APP LOCK SYSTEM FOR ANDROID SMART PHONES USING FACE RECOGNITION

Răzvan Stoleriu[1], Mihai Togan[1]
[1] Computer Science and Cybersecurity Dept., Military Technical Academy "Ferdinand I", Bucharest, Romania
Contact author e-mail: mihai.togan@mta.ro

***Abstract*—Mobile operating systems pervade our modern lives as we interact with these almost every day. Security and privacy are of particular concern on these systems, as they have access to a wide range of sensitive resources and can access private data. While just using the Settings app on iOS smart phones you can both secure the access to the screen and installed apps, in Android smart phones you can only lock the screen. For locking the apps, you must download a special application from a third-party store like Google Play. The objective of this paper is to emphasize the importance of having only one application which secures access to both the screen and the apps on Android operating systems using facial recognition. We created an application that can restrict the access to the installed apps in the system, control them remotely through a web platform and lock the screen. All of these ensure complete security, beginning from the screen, the main gateway that offers access to the phone, to the installed applications that may contain important information. Thus, using a single application, users can secure the entire phone without needing to use the phone settings.**

Keywords—Android; app locker; app security; app's remote control; screen locker

I. INTRODUCTION

Nowadays, many people keep a huge amount of important information in their phones – from bank accounts to contact information. Because of this, smartphone companies are working hard to improve their security and many of the latest phones come with some innovative features to keep our personal data safe. [1] Mobile devices are equipped with characteristics that we would not have imagined 10 years ago and which currently form incredibly powerful computing systems. [2] Mainly, they offer some basic applications like contacts, camera, radio, memos and an alarm, as well as holding different other features. Thus, they make people use them more and more in their leisure time and whenever they have the opportunity. Additionally, they are utilized for many other purposes like networking services (chatting, email, social networking service, etc.), for work processing, real-time communication with other users and multimedia services (multimedia data sharing, video chatting, etc.). Because of the above reasons, the penetration rate of the smart-phones has increased a lot. For the fact that smart phones have many features and can perform a lot of things, people take them almost everywhere and, in this way, they are likely to be stolen or lost. Consequently, most mobile devices come with some built-in locking characteristics. These include PIN (personal identification number) which is a combination of digits, passwords, pattern lock which consists of tracing a specific model of lines on the screen, face recognition and sliding lock which consists of swiping left or right. From all of these, the most widely used are PIN, password, pattern and sliding lock methods, but they are unfortunately vulnerable to shoulder surfing and smudge attacks. As a result, new innovative and performant locking methods are required. [3]

Our approach aims to create an Android solution that restricts the access to the installed apps on mobile devices, remotely control these apps through a web platform and lock the screen. This solution helps users who want to ban unauthorized access to private data stored on their phone. If a particular person that uses a mobile device with Android OS wants to completely secure their phone by configuring its settings, they will eventually realize that it is not possible. These phones only allow screen lock, while those with iOS benefit from both services. So, to restrict the access to the apps, that person needs to install an application from Google Play Store. Our solution was designed to solve these problems, coming to the help of users through an easy to use interface. To lock the screen, a user has the following available mechanisms: face recognition, PIN, pattern, unlock button or slide button, depending on the category, the level of knowledge he has and the restrictions he wants to impose. On the other hand, in order to restrict the access to applications, there are available the following methods: face recognition, PIN or pattern. Security based on face recognition uses the OpenCV library which is among the best and most performant in the field. The library offers three methods of facial recognition: Eigenface, Fisherface and Local Binary Patterns Histogram (LBPH). [4] We opted to use LBPH algorithm in our application for accurate real-time processing of data as its computational complexity is less and more efficient compared to the other face recognition algorithms [5]. The above security methods, except for the unlock or slide buttons, will keep both the screen and applications locked until a valid PIN or pattern is entered or until the user's face is identified. Otherwise, the applications will not be accessible, and the phone screen will always be locked.

The rest of this paper is organized as follows: first, we present in section 2 the related work on this area of research. Then, section 3 introduces the implementations details of the proposed solution, and in section 4 we present an analysis of the experimental results. Conclusions about the methods of securing Android smart phones screen and applications end our paper.

978-1-7281-5611-8/20/$31.00 ©2020 IEEE

II. Related Work

Biometric Identification on smartphones is far away one of the active research areas in secure Information systems. There are several papers regarding face recognition implementations on Android smart phones.

Guillaume Dave et al. [6] had investigated on smart phones the performance of some face recognition algorithms. The mobile devices they analyzed the performance of those algorithms had 600 MHz processor and 256 Mb RAM. The tests were achieved on 134 face images of 10 different persons. Using the Fisherface algorithm, the results indicated a recognition rate of 94% which was achieved in no more than 1.6 seconds.

Another approach on this subject is taken by Vazquez-Fernandez et al. [7]. They presented a smart photo sharing application for Android devices based on facial recognition. The mobile devices used for tests were HTC Desire with 1 GHz processor and 576 MB RAM and Samsung Galaxy Tab with 1 GHz processor and 512 MB RAM. They tested the application for 50 contacts with 4 faces for every contact and obtained 0.35 sec on HTC Desire and 0.47 sec on Samsung Galaxy Tab to recognize the faces.

One effective use case of facial recognition in Android devices is presented by Dospinescu and Popa [8]. They created an application for smartphones that can recognize human faces. The main goal of that app was to grant access to certain areas or rooms only to certain authorized persons. They exposed some examples for the use of the app such as hospitals or educational institutions where there are rooms in which only certain employees can enter. The authors imported the OpenCV library into the Android project and used the FaceIdentity algorithm for the facial recognition process. They tested the application on different types of images that contain from 1 to 100 faces, testing the face detection and face recognition functions. The results depend on the light conditions and on the face position on the camera. They also established that the face detection process works very quickly, making tests on images with groups of 100 persons. In this situation, with group of people pictures, the face detection process works much better than facial recognition.

A great work was done by Chaudhry and Chandra in [9]. The authors presented the design and implementation of a face detection and recognition system for the visually impaired persons using mobile computing. The mobile system is assisted by a server-based support system. For face recognition they used the built-in algorithms from the OpenCV library which was integrated into the Android project. The proposed face recognition system is designed to take advantage of the portability of mobile devices and provide a simple user interface for the visually impaired people. In order to achieve this, it is needed a portable system. So, a mobile device's camera and earpiece are used to form a compact and lightweight one. The recognition program uses the Local Binary Patterns Histograms (LBPH) algorithm which allow for fast feature extraction. The mobile application was tested on a custom video database with 8 videos which were taken at evening and had different face orientations. There was a total of 80 experiment runs for face detection. The results showed an accuracy of up to 93% in well-light conditions. Better detection accuracy was achieved when the person is looking directly at the camera with a neutral facial expression and lower accuracy is obtained when faces are detected at slight angles and different facial expressions are used. For face recognition, there was a total of 50 experiment runs. There was relatively high recognition accuracy for the persons looking almost directly at the camera with a neutral face expression. Lower results were obtained for recognizing faces at angles with different facial expressions. A major factor in low recognition accuracy is the use of only frontal face images for enrolling a person into the face database.

The authors of [10] describe and explain the process of developing an Android mobile application for recognizing person's gender, age and face. The face detection and recognition process in the mobile application is achieved with the help of the OpenCV library. LBP face features classifier was used for face detection and LBPH model was used for face recognition purposes. Although the recognition algorithms successfully performed their job, they are affected by various conditions such as illumination, pose of the person, facial expressions, face coverage, camera features as well as the performance of the mobile device itself.

Marian Harbach et al. [11] present a detailed analysis of the smartphone locking mechanisms. They try to explain that these systems need to prevent unauthorized parties from gaining access to devices (security), while also minimizing the legitimate user's burden (usability), in terms of both cognitive load (e.g., remembering a PIN) and the time it takes to successfully authenticate. Through a month-long field study, they logged events from 134 users with instrumented smartphones and showed how existing lock screen mechanisms provide users with distinct tradeoffs between usability (unlocking speed vs. unlocking frequency) and security. They found that PIN users take longer to enter their codes, but commit fewer errors than pattern users, who unlock more frequently and are very prone to errors. Overall, PIN and pattern users spent the same amount of time unlocking their devices on average. The data presented in this paper is limited in a few ways. First, all participants came were students and staff of the University of Buffalo and is thus not representative of all smartphone users. Furthermore, all of them were using the same type of device, an LG Nexus 5. Also, the authors data collection was limited to a 30-day interval.

III. Solution Design and Implementation

We proposed an application that locks the access to the entire phone. From the screen, the main gateway somebody can get into the phone, to the installed apps in the system which can contain sensitive information (e.g., email apps, message apps, social media apps, chatting apps, contacts app, etc.). Besides that, some apps like Facebook, LinkedIn or Gmail do not require users to introduce their login credentials every time they access them. Therefore, this can be seen as an access control problem because, if someone's phone lies in the hands of other people, then anybody can access those apps. Our solution was designed to avoid this kind of situation by providing application authentication through many security methods. Furthermore, our system offers the possibility to remotely view all installed applications in a phone and to lock or unlock the access to the preferred ones. The best practical approach is the corporate environments where administrators can have a complete

situation as regards the applications access for their employees. Additionally, they may remotely lock or unlock the access to specific apps.

From an architectural point of view, our solution is quite modular and can be easily modified to be integrated into bigger systems. Firstly, the final mobile application can be installed only on mobile phones with a minimum API of 15. According to [12] just 0.3% mobile phones on the Android market use API 15, so it can be installed on any phone. Another condition is that the phone has front camera, for the face authentication process. As more than 85% of smartphones on the market have front camera, this doesn't really represent an issue. In order to secure the screen access, there are available the following methods: unlock button, sliding button, PIN, pattern and face recognition. For securing the access to the installed apps in the phone, there are available the following mechanisms: PIN, pattern and face recognition. Looking at the security methods listed above, we can observe that our application can be used by everyone, no matter their age or their knowledge in the IT field. The overall software solution for face detection and recognition in the mobile application is achieved with the help of the OpenCV library which provides several functionalities for developing computer vision applications. We used the Local Binary Patterns Histogram (LBPH) model for face recognition purposes. The LBP operator labels the pixels of an image by thresholding a 3×3 neighborhood of each pixel with the center value and considering the results as a binary number. Formally, given a pixel at (x_c, y_c), the resulting LBP can be expressed in the decimal form as in the following expression:

$$LBP(x_c, y_c) = \sum_{n=0}^{7} S(i_n - i_c) 2^n \quad (1)$$

where n runs over the 8 neighbors of the central pixel, i_c and i_n are the gray-level values of the central pixel and the surrounding pixel. $S(x)$ is 1 if $x \geq 0$ and 0 otherwise [13]. According to [14], LBPH outperforms other algorithms (Eigenface, Fisherface) and has minimum noise interference. Hence, LBPH is the most accurate and efficient face recognition algorithm available in OpenCV. Also, the changes in illumination didn't cause major problems and, in future, a neural network architecture together with a feature-based approach could be implemented using the most suitable recognition method i.e. LBPH.

Our implementation compared with other implementations could offer the best choice because it completely secures the phone in all points of view, from the screen to the installed apps. Besides, people don't have to install two separated apps – one for screen lock and another for app lock –, our approach offering all functionalities inside just one application. There are situations when persons install an antivirus app in which they use only the app lock functionality just because that's all they need – this brings them more storage capacity usage, low battery life, many services running in background, etc. Our solution offers also the possibility for locking or unlocking apps remotely, by means of a web interface. The frontend of the web application is written in HTML, CSS, along with JavaScript and the backend part is made in Java Spring Boot using IntelliJ IDEA [15]. We also used XAMPP for local hosting. Besides, we utilized MySQL Workbench [16] which represents the graphical user interface of the MySQL server. Here we have created a database in which is stored information about the installed applications in the phone.

For a situation in which the user enters multiple times the wrong PIN or Pattern or their face isn't recognized, it will automatically appear on the screen an activity with the security question for recovery. On the other hand, we use the PIN, Pattern and Face Recognition methods to lock the access to the installed apps in the system. They cover two of three types of information somebody needs in the authentication process, something you know (PIN, Pattern) and something you are (Face Recognition). The last functionality of our solution is the possibility of viewing the locked and unlocked apps in the phone by the means of a web platform. Additionally, an administrator or the user can lock or unlock other apps using the web application and the changes will take effect after a short time in the phone. This approach is usually used in the corporate environments where an admin restricts the access to some apps in the employees' phones (e.g. Facebook, Instagram, Twitter). This process happens in the following way. A user restricts the access to some applications in his/her phone. The list of apps is encapsulated in a JSON format and sent via Android Volley library [17] to the server. The server which is created using Java Spring Boot framework [18], receive the data, processes and stores it to the MySQL database. When the user wants to see the list of applications, gets into the browser, types in the specific URL and will get displayed in a web page the required information. Here, they can supplementary lock or unlock other apps. The mobile application requests every five minutes from the server the list of applications and when it get that, if there are changes, they will be applied in the phone. The entire process is described in Fig. 1:

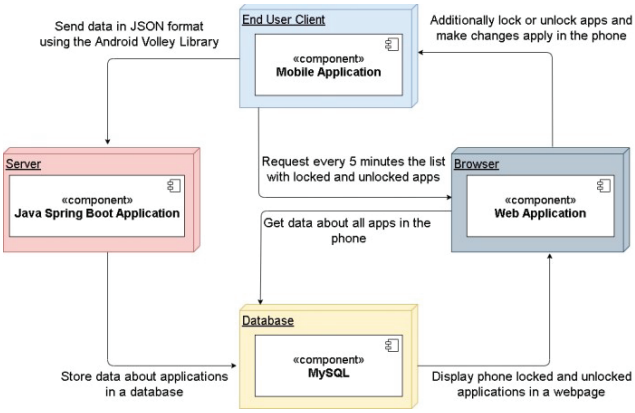

Fig. 1. Software components overview

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In order to evaluate the proposed system, we installed our application on a Lenovo A2016a40 mobile phone running Android OS. Its specifications are listed in Table I:

TABLE I. PLATFORM USED FOR CONDUCTING THE EXPERIMENTS

Specification name	Specification value
Android Version	6.0
CPU	Quad-core 1.0 GHz Cortex-A53
GPU	Mali-T720MP2
RAM	1 GB
ROM	8 GB
Resolution	854 x 480 pixels
Selfie Camera	2 MP

The mobile application offers a menu from which a user can choose which part of the phone to be secured: the screen (Locker Screen Mode in the menu), the access to the installed apps (Locker Mode in the menu) or both. First of all, let's begin our system evaluation with the screen lock part. The application offers some mechanisms, each one with a different level of security: unlock button, sliding button, PIN, Pattern and Face Recognition.

The basic methods are the unlock and sliding button ones. The first gives access to the phone's content by pressing a button and the second one by sliding an icon to the left or to the right. They are usually used when somebody keeps its phone in the pocket and want no action to be taken (e.g. call somebody, send SMS with junk text) if by mistake the phone's screen turns on. The next ones are the PIN and Pattern methods. The PIN method locks the screen and provide access only if the right PIN is entered. In the case of the Pattern method, the user can use the phone only if they draw the right model of lines on the screen. Both, the PIN and the Pattern codes are stored in the application's private files using the SharedPreferences [19] class in Java. Only the application has access to those files, so nobody can change their values. If the user chooses the face recognition method, which is the most performant and very easy to use, then he/she will have to firstly take five pictures of them (behind are taken 10 instant pictures for every captured photo, finally being a number of 50). These photos will be stored on the phone's memory card and used in the recognition process. The next step is setting a PIN and a security question for the situation their face is not recognized and they don't remember the PIN. When the screen is locked using this method, the user will have to scan its face in order to be provided access to the phone.

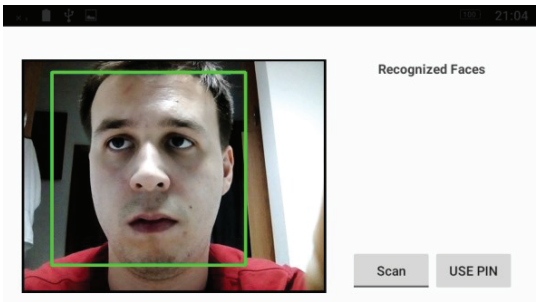

Fig. 2. User's face scanning

The screen lock activity doesn't disappear when any of the buttons Home, Back or Recent Apps are pressed. In the situation in which a user tries to reboot the phone in order to get rid of that security screen, it will appear again, before the PIN insertion (we refer here to the PIN of the phone which consists of 4 digits). Second of all, let's evaluate our system as regards the app lock part. The lock application provides three security mechanisms: PIN, Pattern and Face Recognition.

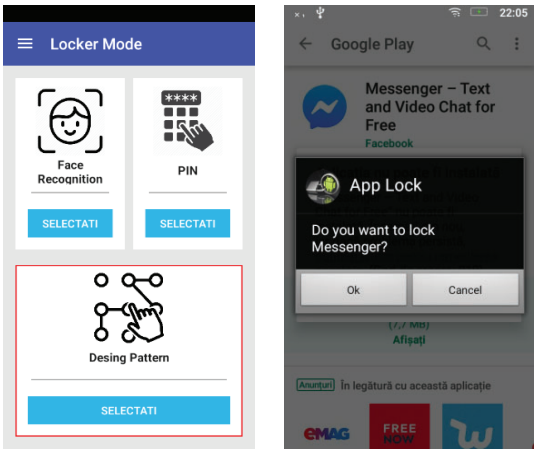

Fig. 3. App lock options and locking the access to a new installed app

Whenever an application is in the block list (e.g. YouTube) and the user tap the icon to get into the application, foremost it will appear our app which will require him/her to enter the PIN or Pattern or to scan their face in order to provide access to the required app. If somebody tries to get rid of this by pressing the Back, Home or Recent Apps buttons, our app goes together with the app it locks, in background. When afterwards that person will bring again the required app in foreground, our application will also come, overlapping the another one. If the PIN or Pattern is forgotten or the face is not recognized, the user has the possibility to answer the security question in order to be provided the access to the wanted app. Another important thing which needs to be mentioned is that our application is aware of every new installed app and asks the user if they want to lock the access to it. The last functionality our solution offers is the web platform by means of which the owner of the phone or an admin can see all installed apps in the system, which of them being locked or not. Besides that, they additionally can unlock or not other apps and the results will be applied after few minutes in the targeted phones. Fig. 4, Fig. 5 and Fig. 6 describe this process, for the Gmail application:

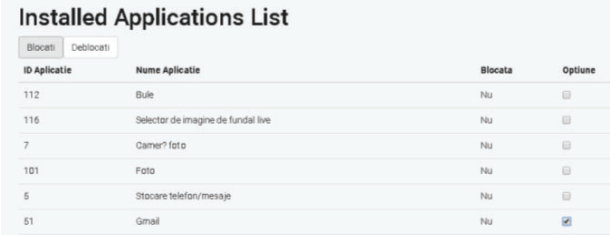

Fig. 4. Lock the Gmail app

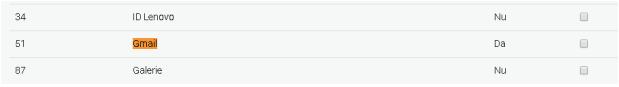

Fig. 5. Gmail is locked

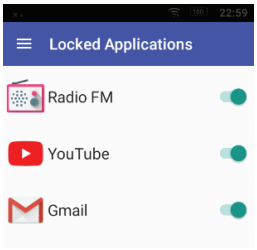

Fig. 6. Gmail is also locked in the phone

The main feature the application's security bases on is face recognition. In order to identify the owner of the phone, it compares the real-time captured photos with the stored ones. If the owner accidentally removes them from the memory card, then the application automatically switches its security mechanism to PIN. That's the reason why PIN is the second method which must be set after face recognition pictures are taken. One of the results we got when tested the face recognition feature against a face from a photo is that the face is not recognized.

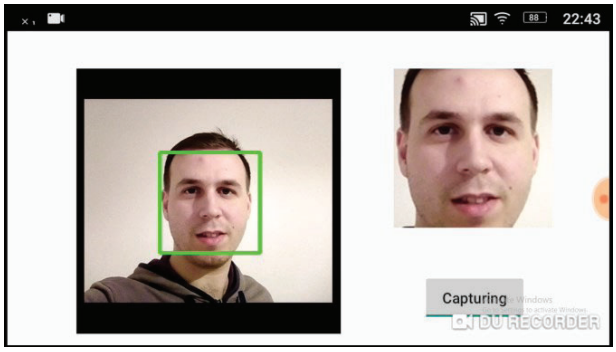

Fig. 7. Face picture stored for future comparisons

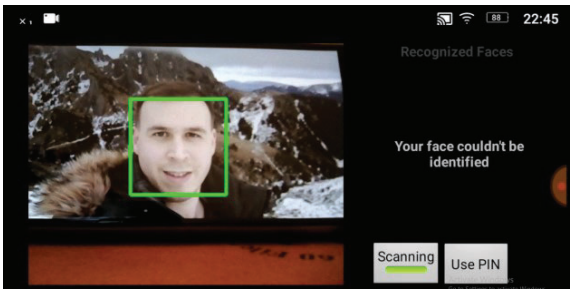

Fig. 8. The same face but from a photo is not recognized

The face recognition process also works in poor lightning conditions, as shown in Fig. 9:

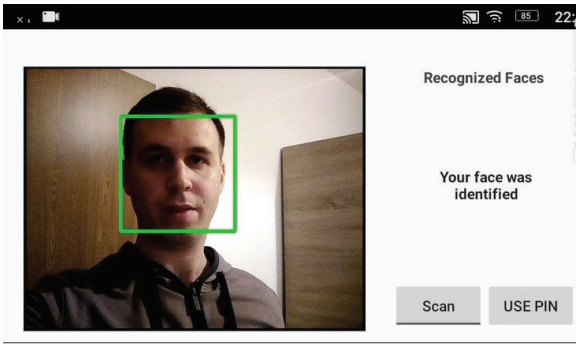

Fig. 9. Face recognition in poor light conditions

V. CONCLUSIONS AND FUTURE WORK

The main contribution of this paper is to propose an Android solution which both secures the screen and applications access. It offers different security authentication mechanisms from the basic to the most advanced ones. The face recognition is fast and performant as it uses LBPH algorithm imported from the OpenCV library. The results showed that the access to the phone or applications cannot be misled, being resistant to some button tricks like Home, Recent Apps, Back, or to the photos of the owner's face. The face recognition process also works well in poor lighting conditions. Our solution offers the advantage of having the entire phone secured by the means of a single app, without the need to install third-party apps or to use the settings app. The solution also provides a web platform from which the user or an admin can view all installed applications, which are locked or not and additionally can lock/unlock others. This approach can be used in corporate environments where there are applied restricts to employees.

As a future work we will try to add other new biometric-based security methods such as fingerprint mechanism. It can be very fast, handy and easy to use. Almost all the phones on the market come with the fingerprint sensor. Secondly, we plan to optimize our solution in order to consume few resources (background services not draining the battery, low processing power, etc.). Last but not least, we take into consideration the

improvement of the response time, in the case of face recognition and fingerprint scan processes. Our expectations are that all of these and other computations or actions will be done almost instant.

REFERENCES

[1] T. Oppong. "The Evolution of Smartphone Security." AllTopStartups.com.

[2] L. Caetano. "Mobile World Congress: The Evolution of Mobile Security Through the Years." McAfee.com.

[3] R. Srilekha and D. Jayakumar, "A Secure Screen Lock System for Android Smart Phones using Accelerometer Sensor," International Journal of Science Technology & Engineering, vol. 1, no. 10, pp. 96, Apr. 2015.

[4] D. L. Baggio, S. Emami, D. M. Escriva, K. Ievgen, N. Mahmood, J. Saragih and R. Shilkrot, "Mastering OpenCV with Practical Computer Vision Projects", Packt Publishing, Dec. 2012.

[5] SudhaNarang, K. Jain, MeghaSaxena and AashnaArora, "Comparison of Face Recognition Algorithms Using Opencv for Attendance System," International Journal of Scientific and Research Publications, vol. 8, no. 2, pp. 268-273, Feb. 2018.

[6] G. Dave, X. Chao and K. Sriadibhatla, "Face Recognition in Mobile Phones," Department of Electrical Engineering Stanford University, 2010. Accessed on: Oct. 9, 2019. [Online]. Available: https://www.researchgate.net/publication/228445783_Face_Recognition_in_Mobile_Phones.

[7] E. Vazquez-Fernandez, H. Garcia-Pardo, D. Gonzalez-Jimenez and L. Perez-Freire, "Built-in face recognition for smart photo sharing in mobile devices," 2011 IEEE International Conference on Multimedia and Expo, Jul. 2011.

[8] O. DOSPINESCU and I. POPA, "Face Detection and Face Recognition in Android Mobile Applications," Informatica Economica, vol. 20, no. 1/2016, pp. 20–28, Mar. 2016.

[9] C. Shonal and C. Rohitash, "Design of a Mobile Face Recognition System for Visually Impaired Persons," Feb. 2015. Accessed on: Sep. 15, 2019. [Online]. Available: https://www.researchgate.net/publication/271855592_Design_of_a_Mobile_Face_Recognition_System_for_Visually_Impaired_Persons..

[10] A. Salihbasic and T. Orehovacki, "Development of Android Application for Gender, Age and Face Recognition Using OpenCV," 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), May 2019.

[11] M. Harbach, A. De Luca, and S. Egelman, "The Anatomy of Smartphone Unlocking," Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, May 2016.

[12] "Distribution dashboard." developer.android.com. https://developer.android.com/about/dashboards (accessed Dec. 2, 2019)..

[13] C. Shan and T. Gritti, "Learning Discriminative LBP-Histogram Bins for Facial Expression Recognition," Procedings of the British Machine Vision Conference 2008, 2008.

[14] SudhaNarang, K. Jain, MeghaSaxena and AashnaArora, "Comparison of Face Recognition Algorithms Using Opencv for Attendance System," International Journal of Scientific and Research Publications, vol. 8, no. 2, pp. 268-273, Feb. 2018. Accessed on: Nov. 25, 2019. [Online]. Available: http://www.ijsrp.org/research-paper-0218.php?rp=P747200.

[15] "IntelliJ IDEA." techopedia.com. https://www.techopedia.com/definition/7755/intellij-idea (accessed May 10, 2019).

[16] IAN. "What is MySQL Workbench?." Database.Guide. https://database.guide/what-is-mysql-workbench/ (accessed June 18, 2019).

[17] "Volley overview." developer.android.com. https://developer.android.com/training/volley (accessed Dec. 2, 2019).

[18] R. Zoltan. "What Is Spring Boot?." DZone.com. https://dzone.com/articles/what-is-spring-boot (accessed May 10, 2019)..

[19] A. PILLAI. "Shared Preferences in Android explained in detail." GadgetSaint.com. http://www.gadgetsaint.com/android/android-shared-preferences-detail/#.XfeykOgzbIU (accessed Mar. 27, 2019).