4.1. Mobile Application

In this section mobile application of Smart Locking Control System will be explained. User controls the door lock from that application and welcome screen of application is as shown as figure 1. It works on Android devices its Android version must be newer than Android 4.4 KitKat (API 19) to use on mobile device.

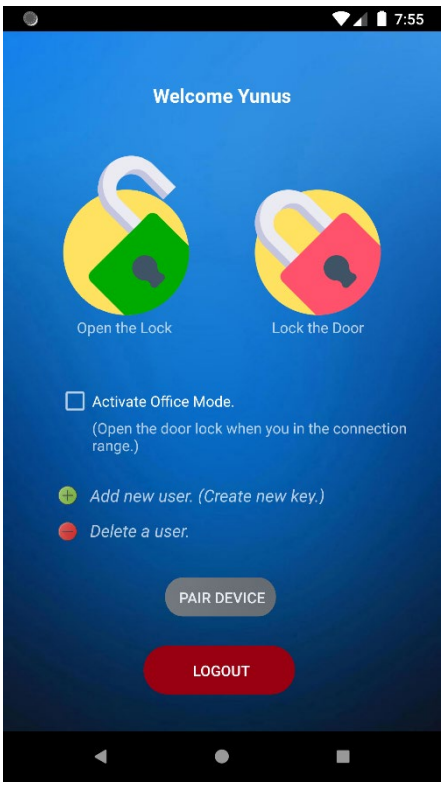

Figure 1: Welcome Screen of Application

Android Studio 3.5.3 was used as IDE and mostly written with Java language. All simulations were done on Android 8.1 Oreo (API 27) and compiled on Android 10.0 Q (API 29). Also, this application uses Retrofit 2.8.1, Gson 2.8.6 and converter Gson 2.8.1 as implemented tools.

It is not possible to put all program codes here because the program has more than 1000 line of codes, so it can be accessed from the link below.

https://github.com/powerstttt/SmartLockingControlSystem/

4.1.1. The Algorithm

Smart Locking Control System Application uses internet and Bluetooth connections to talk with microcontroller and server. It gets the user information from server and sends data to microcontroller in order to control the door lock. When the door is wanted to be unlocked, the following operations are performed in order.

1. User logs into the application.
2. Now, user sees the welcome screen and must pair device with Smart Locking Control System.
3. After device is paired, user can select to lock or unlocked the door.

4. Mobile device sends data to microcontroller via Bluetooth according to selection.
5. Smart Locking Control System checks user from server and lock or unlocked the door.

These instructions are made for every single lock operation. After this stage, the mobile application can be explained with more detail. The application communicates with server and microcontroller of Smart Locking Control System separately, which are represented in figure 2.

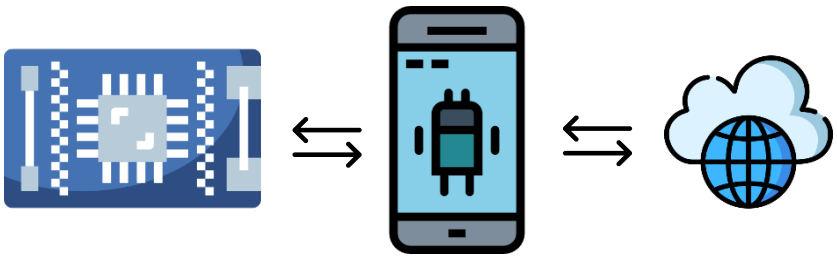

Figure 2: Illustration of mobile device communication

4.1.2. Working Structure

The application was created with one activity named MainActivity. It contains fragments to run other processes. In summary, it can be thought as follows. There is an activity as fundamental of application and it changes fragments inside of it. Figure 3 attempts to show the MainActivity and fragment container to visualize in mind. The main activity can be considered as a mobile device screen, and fragments change in fragment container depending on the user's actions.

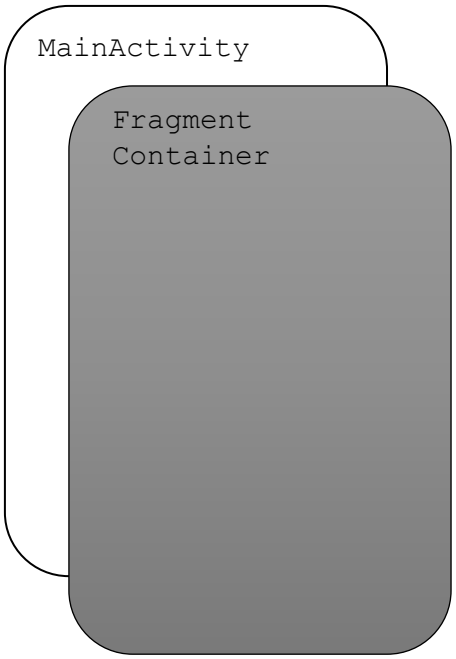

Figure 3: MainActivity and Fragment Container

In this application, there are 5 fragments that names are LoginFragment, WelcomeFragment, RegistrationFragment, DeletionFragment and PairDeviceFragment. All of

them are can be thought of a screen of application singly. WelcomeFragment was already shown in figure 1.

4.1.3. Connections

The application is used two different connections that are Bluetooth and internet, and user must give permission to the application for these. The following codes gives these permissions to the application by asking the user.

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

4.1.3.1. Internet Connection

Internet connection is required for access the database in server. Server stores user data such as username, password, name. The application connects with server through the internet when user is wanted to log in to application or other user operations like create or delete a key.

4.1.3.2. Bluetooth Connection

Smart Locking Control System and its mobile application communicates with the Bluetooth connection. The application sends data to lock or unlocked the door lock and the system makes operation according to that data.

4.1.4. Java Classes

This application consists of 9 java classes and 1 interface. Description of all of them are given below. 7 of them have layouts in *.xml format and they also display.

4.1.4.1. MainActivity.java

This is the main class of activity. It consists of a fragment container and display other fragments. All fragment replacements are managed in here. In addition, every time the program is opened, it checks whether it has been logged in before and it does not direct it to the login screen if it has been done. At the same time, the program communicates with the server every time the program is opened to understand whether the user is still in the database or not. If the user is deleted from the database, then it directs user to the login screen.

Arrangements for how the display will look on the mobile device screen have been made in activity_main.xml. The fragment container that is mentioned before is described in here.

4.1.4.2. LoginFragment.java

This file manages the login processes. This is the first screen of the application. When user enters the username and password as shown in figure 1, the application runs performLogin() function which is defined in that class. It accesses to the server and when takes the response, if the connection is successful and username and password are correct then it passes to locking control screen. If it is not, it gives an error message to the user about the problem. Also, its design parameters are defined in the fragment_login.xml.

4.1.4.3. WelcomeFragment.java

This class consists of control processes. The mobile device must be paired with Smart Locking Control System before using these functions. There are 4 listeners for buttons to access to other fragments. Registration, deletion, login and pair device screens are accessible with these listeners when click on related button.

This class fragment is initialized the Bluetooth adapter in order to send data to HC-05 Bluetooth module, which is inside of Smart Locking Control System. Also, Office Mode can be turned on or off from here. Its design parameters are defined in the fragment_welcome.xml.

4.1.4.4. RegistrationFragment.java

This class provides to create new key for new users. User can assign username, password and name for new user. It sends user information to database of server and new user can access the system instantly. Its design parameters are defined in the fragment_registration.xml.

4.1.4.5. DeletionFragment.java

This class provides to delete a user from the database. It communicates with server and realize the operation. Its design parameters are defined in the fragment_deletion.xml.

4.1.4.6. PairDeviceFragment.java

This class contains two functions. First one is, it toggles the Bluetooth to open or close it. Second one is, it is listed all Bluetooth devices in the mobile device range and provided to make connection between mobile device and Smart Locking Control System. User must be done pairing device before the use to the system. Its design parameters are defined in the fragment_pair_device.xml.

4.1.4.7. PrefConfig.java

This class contains all global functions for the program such as write or read name, login status, address of Bluetooth device etc. It uses shared preferences to store data and all variables are in the strings.xml. This class has no display, so it works in background and user does not see this.

4.1.4.8. User.java

This class contains two parameters (that are response and name) for server communication. When the user tries to log in, the application communicates with the server and the server sends a response. Response value can be “ok” or “failed”. The application continues according to this response.

4.1.4.9. ApiClient.java

This class is initialized the Retrofit2 which creates a connection between the application and HTTP API from server [1]. It is made possible to make a call from the application to the server with Retrofit2. Additionally, this class contains base URL of the server. As we use local host, URL is entered to IP address of NIC card of the local host system. LoginApp shows where the application on the server is installed.

```
public static final String BASE_URL = "http://192.168.1.33/LoginApp/";
```

4.1.4.10. ApiInterface.java

This file is interface of API client which is used call from the application with Retrofit2. It defines which call will call which PHP file in the server. All PHP files are written specially for Smart Locking Control System Application. More detail can be found in the section 4.3. Server Installation.

Figure 4 and figure 5 belong to the images within the application.

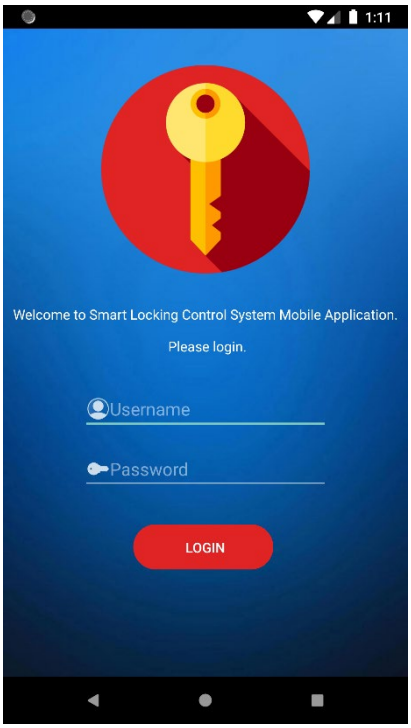

Figure 4: Login Screen

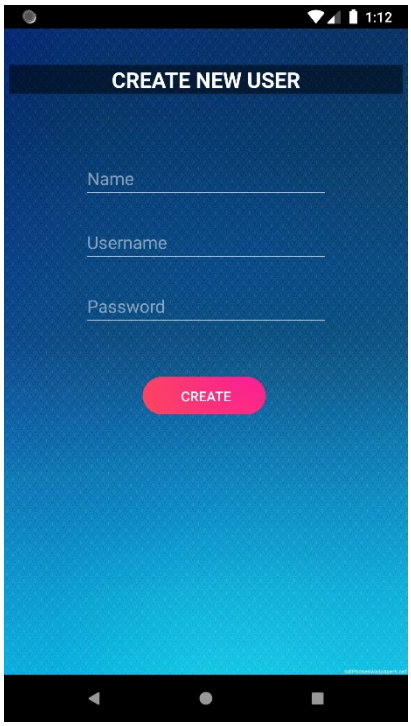

Figure 5: Registration Screen

References (application)

[1] Square, Inc., "retrofit2 (Retrofit 2.7.1 API)", *Retrieved from https://square.github.io/retrofit/2.x/retrofit/, (11 June 2020).*