CS M117 - Computer Networks: The Physical Layer

Final Project Report - Two Factor Access Control (2FAC)

Date: 2nd December, 2016

Professor: R.P. Dzhanidze

Team Members: Anant Mahajan, Chuyue(Camille) Zhang, Christopher Boshae, Rehan Shah, Shounak Roy

I. INTRODUCTION AND PROJECT MOTIVATION

- **Two Factor Access Control – Main Motivation**
  - To scale the two-factor/multi-factor authentication process used to access secure online systems and apply it to a real-world physical security system.
  - To curb the rising threat to privacy at the individual and institutional level

- **Two Factor Access Control – Theory**

Two Factor Authentication, also known as 2FA, two step verification or TFA (as an acronym), is an extra layer of security that is known as "multi factor authentication" that requires not only a password and username but also something that only, and only, that user has on them. The 2FA system usually involves two of the following three authentication factors –

- Physical object in the possession of the user (Key, USB with Secret Token, etc.)
- Secret knowledge of the user (Password, PIN, etc.)
- Physical characteristic of the user (Fingerprint, eye iris, etc.)

Our project, 2FAC draws inspiration from this and utilizes the last two authentication factors in the form of a pin and fingerprint to control access to secure facilities or buildings. It is essentially a dual layer locking mechanism.

Figure 1: Authentication Factors used in 2FAC

II. PROJECT FUNCTIONALITY

❖ **Real World Functionality -**

Electronic access control uses computers to solve the limitations of mechanical locks and keys. A wide range of credentials can be used to replace mechanical keys. The electronic access control system grants access based on the credential presented. When a credential is presented to an access control reader, the reader sends the credential's information to a control panel, a highly reliable processor. The control panel compares the credential's number to an access control list and grants or denies the presented request.

❖ **Proof of Concept Functionality -**

➢ Components of our 2FAC system -

- Access Control Point : Door (Virtual)
- Access Control Medium : Electronic
- Access Control Type : No operator, so non-HITL (no Human-in-the-loop)
- Access Control Reader : Adafruit Fingerprint Sensor
- Access Control List : Fingerprint Database

➢ Equipment used to implement our project -

Equipment Name	Quantity	Basic Functionality
Adafruit Fingerprint Sensor	1	To store fingerprint images of enrolled users and allow for fingerprint matching and validation – 1st level of authentication
OSEPP Mega 2560 (Arduino Compatible Board)	1	Microcontroller Board used to communicate with the fingerprint sensor and the internet
Arduino WiFi 101 Shield	1	Connect the Arduino compatible board to the internet wirelessly (WiFi)
Apple iPhone 5C	1	Uses a 4G cellular connection to act as a personal hotspot
Android Smartphone	1	Houses the Android application that allows users to get through the 2nd level of authentication by using a personalized pin

Table 1: Equipment List for the implementation of 2FAC

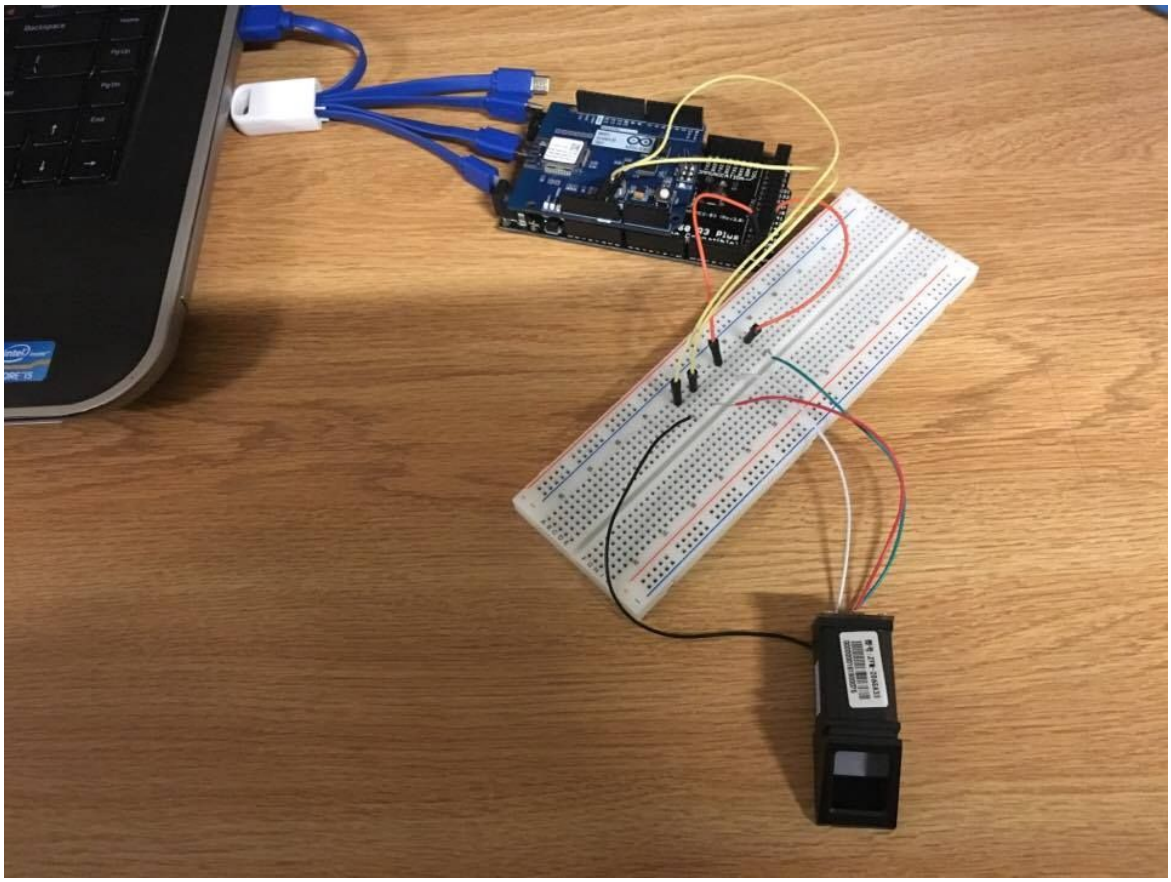

Figure 2: Setup used to demonstrate the functionality of 2FAC

III. WIRELESS NETWORK - WiFi

❖ **<u>WiFi over Bluetooth</u>**

In our project, we chose to use WiFi to connect Arduino to the internet as it has the following advantages in comparison to other wireless communication technologies such as Bluetooth.

❖ **WiFi has a higher bandwidth than Bluetooth.**

WiFi has a bandwidth to up to 11 Mbps compared to Bluetooth, which has a bandwidth of 800 Kbps. As a result, WiFi is a better suited for operating full-scale networks when the the speed of the connection is a factor. While on the other hand, a low-bandwidth wireless network such as Bluetooth is better suited for transferring sound data or byte data (i.e files) when speed is not an issue. Therefore, we chose WiFi for our project because we aim to establish a full-scale web network security system where speed of connection is taken into consideration.

❖ **WiFi has lower latency than Bluetooth.**

WiFi has an average latency of 150ms while Bluetooth has a latency of 200ms. The relatively higher latency found in Bluetooth wireless connection is mainly due to low transmission rate. When a master device assigns each slave a time to transmit, each Bluetooth device transmits at up to 2Mb/s and oftentime less than that. This results in longer total transmission time which slows down the entire network. Thus, we chose WiFi over Bluetooth in an attempt to reduce latency to improve the connection condition.

- **It's easier to implement one wireless technology for multiple purposes.**

The system requires an online database that must be used to store all's individuals who have access to the lock. A local offline database within the Arduino would fulfill the same objective, but it would limit the scalability of the system because memory is limited within the Arduino; therefore that was not a choice. In order to communicate with the online database Wifi is required. If the arduino used Bluetooth to send the code to the user, two wireless protocols would have to be managed within the device. This would create additional complexity as each protocol has its strengths and weaknesses. Therefore, by using just one type of wireless technology, we constrained the factors that might affect the functionality of the system.

- **WiFi has better security**

WiFi connection provides stronger security as a password is required to access a certain network. On the other hand, the security provided by Bluetooth is confined to key matching. Data transmission through Bluetooth can also be vulnerable if the user of the receiver device does not make conscious choice of whether to trust the sender device and thus whether to receive data. In almost all cases, Bluetooth users can establish "trusted devices" that can exchange data without asking permission. Thus, certain responsibility of security is placed on users and thus is susceptible to malicious hacking.

IV. IMPLEMENTATION DETAILS

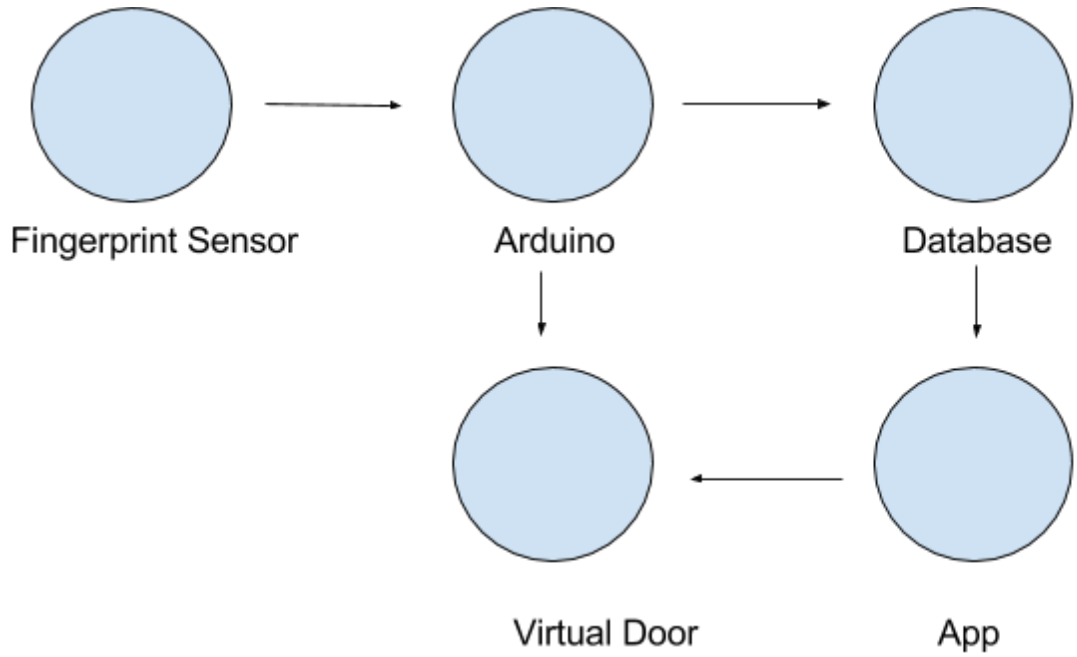

Figure 3: Overview of 2FAC

As shown on the above diagram, our project consists of interaction between 5 devices. The implementation detail can be shown in the following two stages:

- **Stage 1. Fingerprint verification/Code generation**

Summary

The first stage of the 2FAC system is fingerprint detection/ verification. A connection is first establish between the fingerprint sensor and arduino. In our project, we used Adafruit fingerprint sensor which stores up to 160 fingerprint images of individuals who are granted access to a certain facility. When the user fingerprint detection is initiated, fingerprint image is compared with prints in the sensor's database. If a match is found, Arduino connects to WiFi and generates a 4 digit passcode and send it to the database and "door", which is demonstrated virtually in our project by setting up a website that is locked with a passcode.

Technical Details

The entire Arduino and fingerprint sensor part of our project was coded in C as this is the native language of Arduinos. Our Arduino connects to a fingerprint sensor via input pins 51 and 52 on the Arduino board. The fingerprint sensor is accessed programmatically through the company provided Adafruit library. Once we detect a valid finger on the fingerprint sensor, the Arduino connects to WiFi using a WiFi shield and the inbuilt WiFi.h library.

Once connected to the internet, the Arduino generates a random 4 digit code, using the inbuilt uRandom() function, with the RandomSeed() initialized to the input of an unconnected pin on the Arduino board to ensure randomness in the code produced. Once this code is produced, the Arduino attempts to send this code to our online database via an HTTP GET request, formulated using the WifiClient Library for the Arduino. In an ideal world, this connection would be easy to form and straight-forward, but due to our database choice, this connection needed an intermediate step.

Our database (Firebase) required all requests to it to be secured via HTTPS, but the arduino board isn't powerful enough and does not have the memory capabilities to form HTTPS secured web traffic. To solve this problem, we used an intermediate web-page set up to receive a simple HTTP GET request, and translate it into secured HTTPS traffic, which is then sent to our database (Firebase). Another way around this problem could have been to change database, but we preferred to stick with Firebase due to our need for a real-time database and quick updates.

Now, in an actual use case for our project, we would have it connected to a door, but due to our lack of resources, we had to make do with a virtual door. To make the best of a bad situation, the earlier web-page that was used to convert HTTP traffic to HTTPS traffic was made into a virtual door as well. This was done by having it store the codes that it was transmitting to Firebase, and comparing this code to user input, when a user tried to access the site. This website/webpage set up was a collection of simple PHP, HTML and CSS webpages set up using free web-hosting at x10hosting.com.

- **Stage 2: User Receives Code and Enters it**

Summary

Once the code is in the database, it sends it to an Android application. The app consists of three screens as follows:

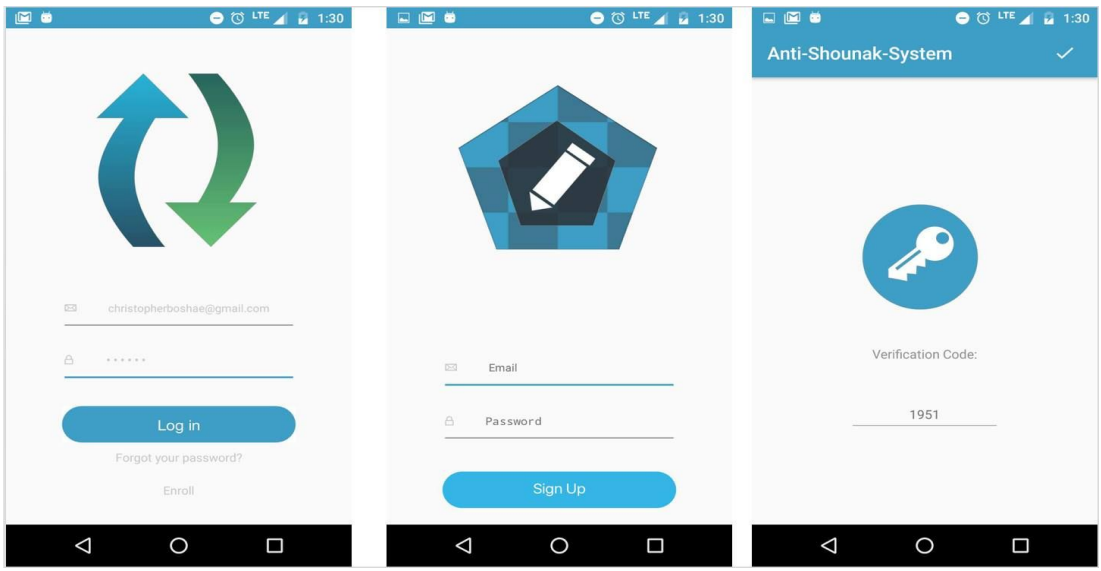

Figure 4: App screens. (L-R) Log in page, Sign up page, Code display page

Once the user logs in to the app, the code from the database gets displayed for 30 seconds while the user gets a chance to enter the code into the door.

Technical Details

The application for our project was made using the Android SDK and coded in Java. The application consists of three activities. The first activity that shows up when the application is opened for the first time is the “LoginActivity”, which allows the user to enter the application and get the code if they have an account. If a user, doesn’t have an account they must create one by pressing on “Enroll”, which will take them to “EnrollActivity”. In this activity, the user just enters their email address and selects a password for their account. If the user presses “Log in” or “Sign Up” in either of the two activities, and their credential information is accepted, they will be able to see the “MainActivity”. In this activity the user will be able to see a verification code, which will be accepted by the authentication system enabling them to enter the premises.

In “LoginActivity” and “EnrollActivity “ the user is given the opportunity to enter their email address and password, so they can access their temporary verification code or create a new account. We use Firebase Authentication to manage the accounts of the users. The layouts of the activities consist of ImageView components, EditText components, and TextView components. The ImageView components are used for images and buttons. The EditText components hold the user’s email address and password. The TextView components are used to show the user clickable options such as “Forgot your password?” and “Enroll” or text. Once the user enters their email address and password, in the EditTexts of “EnrollActivity” or “LoginActivity” and presses “Log in” or “Sign Up”,

the method signInWithEmailAndPassword(email,password) or createUserWithEmailAnd Password(email,password) is called. If the user is in "LoginActivity" and the account credentials are valid or if the user is in "EnrollActivity" and an account is successfully created then the user is presented with the next activity - "MainActivity".

If the user does not remember their password, they can press the text "Forgot your password?" within the "login" activity. This will start a fragment, where the user can enter their email address, in order to send a password reset email to them. A password reset email will only be sent if they already have an account.

The main purpose of "MainActivity" is to allow the user a means of retrieving a verification code that is stored in a Firebase Realtime Database. The layout of the activity consists of a Toolbar, ImageView, and two TextViews. The Toolbar component allows the user to return to the "login" activity and logout by pressing the checkmark. The ImageView component holds an image so the viewer can immediately observe that they have arrived at the final stage of the verification process. The two TextView components show the user the text "Verification Code:" and the code itself. To retrieve the code a Database Reference object is first initialized to point to the root of the Firebase Database JSON tree. We then call the object's method addValueEventListener (ValueEventListener listener), which will read the value stored within the root. The TextView within the activity is then updated to show this value. After thirty seconds, that code is no longer valid for the authentication system, and the value "PIN" is stored within the root.

V. PROJECT RESULTS AND EVALUATION

❖ **Results-**

The project successfully demonstrates the proof of concept of a physical access control system as it correctly grant access to one of the group members while denying access to others. The group member allowed access has his fingerprint previously stored in the fingerprint sensor's database thus getting him through Stage 1 of the authentication process. He then receives a personalized pin on his Android device that allows him to negotiate Stage 2 of the authentication process.

While testing our project, the following critical stages successfully passed testing -

- ➢ Communication between the Arduino microcontroller board and the Fingerprint sensor was established
- ➢ Communication between the Arduino microcontroller board and WiFi was established
- ➢ Multiple fingerprints were successfully enrolled and stored in the sensor database
- ➢ Successful fingerprint matches were generated with a high enough confidence rating
- ➢ 4-digit pin was sent to an online real-time database (Firebase) via a HTTP GET request
- ➢ Intermediate web-page that translates the HTTP GET request to HTTPS secured traffic
- ➢ The pin was visible to registered users on an Android application for 30 seconds
- ➢ Gaining access through a virtual door by comparing user input to the pin previously stored in the database

Each of these passed tests is a strong indicator of the faultless implementation of 2FAC. The fact that this project is very easily scalable to real-world physical security systems makes it a definite success.

- **Evaluation of Risks and Scalability-**
  - Data transfer from arduino to firebase through via a HTTP request is not the most secure option. Instead we could use HTTPS for additional security because all communication is then secured by end-to-end encryption.
  - A major risk of intrusion through this access control system is that someone simply follows a legitimate user through the door. This is often referred to as 'tailgating'. A solution to this is to include an operator in the loop who can assure valid identification.
  - When the project is scaled to actual physical security systems, it would be judicious to maintain an access control log in an independent database. A flashing Red LED coupled with an alarm can be used to notify security personnel of any instance of forced access.

VI. MEMBER CONTRIBUTION

Chuyue(Camille) Zhang, Christopher Boshae -
Android Application Development, UI/UX, Project Report

Anant Mahajan, Rehan Shah, Shounak Roy -
Hardware Integration, Service and Background Functionality, Project Report

VII. APPENDIX -

SOURCE CODE FOR ARDUINO

```
#include <WiFi101.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiSSLClient.h>
#include <WiFiUdp.h>
#include <Wire.h>
#include <SPI.h>
#include <WiFi101.h>
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
```

```
char ssid[] = "Rehan's iPhone";      //  your network SSID (name)
char pass[] = "123bihari123";    // your network password
int status = WL_IDLE_STATUS;     // the Wifi radio's status

int keyIndex = 0;            //your network key Index number
char server[] = "http://ripped-stencil.000webhostapp.com";
WiFiClient client;

int getFingerprintIDez();

// Green is Rx and goes to 52
// White is Tx and goes to 51

SoftwareSerial mySerial(52, 51); // Rx, Tx

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup()
{
  randomSeed(analogRead(0));
  Serial.begin(9600);
  mySerial.begin(9600);

  Serial.println("Adafruit finger detect test");

  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor");
    while (1);
  }
  Serial.println("Waiting for valid finger...");
}

void loop()                     // run over and over again
{
  char outBuff[128];
  int x = getFingerprintIDez();
  if (x!=-1)
  {
    status = WiFi.begin(ssid, pass);
    //delay(10000);
```

```
    Serial.println("check1");

     if (client.connect(server, 80)) {
    Serial.println("connected to server");
    // Make a HTTP request:
    int hash = makehash();
    sprintf(outBuff, "POST /firebase.php?arduino=%d
HTTP/1.1",hash);
    client.println(outBuff);
    client.println("Host: ripped-stencil.000webhostapp.com");
    client.println("Connection: close");
    client.println();
  }
  else {
    Serial.println("couldnt connect");
  }
  while(true)
  {
   if (client.available()) {
    char c = client.read();
    Serial.write(c);
    break;
   }
  }
    if (!client.connected()) {
    Serial.println();
    Serial.println("disconnecting from server.");
    client.stop();
  }
  WiFi.end();
  }
}

int makehash() {
  int result = 0;
  for (int i =0 ;i < 4; i++) {
    long x = random(10);
    result = result*10 + (int)x;
  }
  Serial.println(result);
  return result;
}

uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
```

```
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.println("No finger detected");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK success!

p = finger.image2Tz();
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
p = finger.fingerFastSearch();
if (p == FINGERPRINT_OK) {
  Serial.println("Found a print match!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
```

```
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_NOTFOUND) {
    Serial.println("Did not find a match");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
Serial.println(finger.confidence);
}

// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {

  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
Serial.println(finger.confidence);
  return finger.fingerID;
}
```

SOURCE CODE FOR ANDROID APP

1) LOGIN

```
package com.example.christopherboshae.anti_shounak_system;

import android.app.DialogFragment;
import android.app.ProgressDialog;
```

```
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity implements
ForgotPasswordFragment.ForgotPasswordListener{

    private EditText email;
    private EditText password;
    private ImageView login_button;
    private TextView signup_button;
    private TextView forgot_password_button;
    private FirebaseAuth mAuth;
    private FirebaseAuth.AuthStateListener mAuthListener;
    public static final String TAG = "MESSAGE";

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

        email = (EditText)findViewById(R.id.email);
        password = (EditText)findViewById(R.id.password);
        login_button =
(ImageView)findViewById(R.id.sign_in_button);
        signup_button = (TextView)findViewById(R.id.sign_up_text);
        forgot_password_button =
(TextView)findViewById(R.id.forgot_your_password);

        mAuth = FirebaseAuth.getInstance();

        mAuth.signOut();
```

```
        mAuthListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth
firebaseAuth) {
                FirebaseUser user = firebaseAuth.getCurrentUser();
                if (user != null) {
                    // User is signed in
                    Log.d(TAG, "onAuthStateChanged:signed_in:" +
user.getUid());

                } else {
                    // User is signed out
                    Log.d(TAG, "onAuthStateChanged:signed_out");
                }
            }
        };

        login_button.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                if (validateForm())

signin(email.getText().toString(),password.getText().toString());
            }
        });

        signup_button.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Log.d("TAG", "About to start enroll activity");

startActivity(EnrollActivity.newInstance(LoginActivity.this));
            }
        });

        forgot_password_button.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                DialogFragment newFragment = new
ForgotPasswordFragment();
                newFragment.show(getFragmentManager(), "forgot");
            }
        });
```

```
    }

    @Override
    public void onDialogPositiveClick(ForgotPasswordFragment
dialog) {
        String email_address = dialog.get_email_address();
        final ProgressDialog progress = new ProgressDialog(this,
ProgressDialog.STYLE_SPINNER).show(this,null,null);

mAuth.sendPasswordResetEmail(email_address).addOnCompleteListener(
new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if(task.isSuccessful())
                {
                    Toast.makeText(LoginActivity.this, "Email
Successfully Sent", Toast.LENGTH_SHORT).show();
                    progress.dismiss();
                }
                else{
                    Toast.makeText(LoginActivity.this, "Account
Not Found", Toast.LENGTH_SHORT).show();
                    progress.dismiss();
                }
            }
        });
    }

    @Override
    public void onStart() {
        super.onStart();
        mAuth.addAuthStateListener(mAuthListener);
    }

    @Override
    public void onStop() {
        super.onStop();
        if (mAuthListener != null) {
            mAuth.removeAuthStateListener(mAuthListener);
        }
    }

    private void signin(String email, String password){
        final ProgressDialog progress = new ProgressDialog(this,
ProgressDialog.STYLE_SPINNER).show(this,null,null);
```

```
mAuth.signInWithEmailAndPassword(email,password).addOnCompleteList
ener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task)
{
                if(task.isSuccessful()){
                    progress.dismiss();

startActivity(MainActivity.newInstance(LoginActivity.this));
                }
                // If sign in fails, display a message to the
user. If sign in succeeds
                // the auth state listener will be notified and
logic to handle the
                // signed in user can be handled in the listener.
                if (!task.isSuccessful()) {
                    progress.dismiss();
                    Log.w(TAG, "signInWithEmail:failed",
task.getException());
                    Toast.makeText(LoginActivity.this,
"Authentication Failed",
                            Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    private boolean validateForm() {
        boolean valid = true;

        String emailField = email.getText().toString();
        if (TextUtils.isEmpty(emailField)) {
            email.setError("Required.");
            valid = false;
        } else {
            email.setError(null);
        }

        String passwordField = password.getText().toString();
        if (TextUtils.isEmpty(passwordField)) {
            password.setError("Required.");
            valid = false;
        } else {
            password.setError(null);
        }
```

```
        return valid;
    }

}
```

2) ENROLL/ REGISTER

```
package com.example.christopherboshae.anti_shounak_system;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

/**
 * Created by christopherboshae on 10/29/16.
 */
public class EnrollActivity extends AppCompatActivity {

    private EditText emailField;
    private EditText passwordField;
    private ImageView signUpButton;
    private DatabaseReference databaseReference;
    private FirebaseAuth mAuth;
    private FirebaseAuth.AuthStateListener mAuthListener;
    private ProgressDialog progress;

    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d("TAG", "Enroll Activity Instantiated");
        setContentView(R.layout.enroll_activity);

        emailField = (EditText) findViewById(R.id.email_field);
        passwordField = (EditText)
findViewById(R.id.password_field);
        signUpButton = (ImageView)
findViewById(R.id.sign_up_button);
        mAuth = FirebaseAuth.getInstance();
        databaseReference =
FirebaseDatabase.getInstance().getReference();

        signUpButton.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                if (validateForm()){
                    progress = new
ProgressDialog(EnrollActivity.this,
ProgressDialog.STYLE_SPINNER).show(EnrollActivity.this,null,null);

signUp(emailField.getText().toString(),passwordField.getText().toS
tring());
                }
            }
        });

        mAuthListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth
firebaseAuth) {
                FirebaseUser user = firebaseAuth.getCurrentUser();
                if (user != null) {
                    // User is signed in
                    Log.d("TAG", "onAuthStateChanged:signed_in:" +
user.getUid());

                } else {
                    // User is signed out
                    Log.d("TAG", "onAuthStateChanged:signed_out");
                }
            }
        };
    }
```

```
    private void signUp (String email, String password){
        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull
Task<AuthResult> task) {
                        Log.d("TAG",
"createUserWithEmail:onComplete:" + task.isSuccessful());
                        //Log.d("TAG",
task.getException().toString());
                        if (task.isSuccessful()) {
                            progress.dismiss();

startActivity(MainActivity.newInstance(EnrollActivity.this));
                            finish();
                        }
                        // If sign in fails, display a message to
the user. If sign in succeeds
                        // the auth state listener will be
notified and logic to handle the
                        // signed in user can be handled in the
listener.
                        else {
                            progress.dismiss();
                            Toast.makeText(EnrollActivity.this,
"Authentication Failed",
                                    Toast.LENGTH_SHORT).show();
                        }

                    }

                });
    }

    @Override
    public void onStart() {
        super.onStart();
        mAuth.addAuthStateListener(mAuthListener);
    }

    @Override
    public void onStop() {
        super.onStop();
        if (mAuthListener != null) {
```

```
            mAuth.removeAuthStateListener(mAuthListener);
        }
    }

    public static Intent newInstance (Context context){
        Intent intent = new Intent(context, EnrollActivity.class);
        return intent;
    }

    private boolean validateForm() {
        boolean valid = true;

        String email = emailField.getText().toString();
        if (TextUtils.isEmpty(email)) {
            emailField.setError("Required.");
            valid = false;
        } else {
            emailField.setError(null);
        }

        String password = passwordField.getText().toString();
        Log.d("TAG", Integer.toString(password.length()));
        if (TextUtils.isEmpty(password) || password.length() < 6)
{
            passwordField.setError("Please 6 character password");
            valid = false;
        } else {
            passwordField.setError(null);
        }

        return valid;
    }
}
```

3) MAIN APP

```
package com.example.christopherboshae.anti_shounak_system;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class MainActivity extends AppCompatActivity {
    private TextView pin;
    private DatabaseReference databaseReference;
    private FirebaseAuth mAuth;
    private Toolbar toolbar;

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

        pin = (TextView) findViewById(R.id.pin);
        mAuth = FirebaseAuth.getInstance();
        databaseReference =
FirebaseDatabase.getInstance().getReference();
        toolbar = (Toolbar)findViewById(R.id.my_toolbar);
        toolbar.setTitleTextColor(Color.WHITE);
        setSupportActionBar(toolbar);

        databaseReference.addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                String value =
dataSnapshot.getValue(String.class);
                pin.setText(value);
                pinTimer();
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                Log.w("TAG", "Failed to read value.",
databaseError.toException());
            }
        });
```

```
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()){
            case R.id.log_out:
                FirebaseAuth.getInstance().signOut();
                finish();
                break;
        }

        return super.onOptionsItemSelected(item);
    }

    void pinTimer (){
        CountDownTimer timer = new CountDownTimer(30000,1000) {
            @Override
            public void onTick(long l) {

            }

            @Override
            public void onFinish() {
                pin.setText("PIN");
            }
        };
        timer.start();
    }

    public static Intent newInstance(Context context){
        Intent intent = new Intent(context, MainActivity.class);
        return intent;
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        FirebaseAuth.getInstance().signOut();
    }
}
```

SOURCE CODE FOR WEBPAGE - DOOR Locked/Unlocked

```
<!DOCTYPE html>
<html>
<head>
</head>

<?php

$pass = $_POST["passcode"];

$username = "m117proj_anant";
$password = "gems2010";
$hostname = "localhost";

//connection to the database
$link = mysqli_connect("localhost", $username, $password,
"m117proj_a");

$sql = "SELECT `value` FROM `Values` WHERE `id`=0";

$result = mysqli_query($link, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        if($row["value"] == $pass)
        {
        echo "
<body background='back.jpeg'>
     <h1 style='Padding:10px; text-align:center; border-radius:
10px; width: 75%; color: #D5FF00;position: relative;margin-left:
auto ;margin-right: auto ;'> Welcome. You have proven yourself
worthy. </h1>
       <div style='margin-right: auto; margin-left: auto;position:
relative;'>
        <img src='welcome.jpeg' alt='Welcome'
style='width:350px;height:200px;Padding:30px;'></div>";
        }
        else
        {
        echo "
```

```
<body background='back.jpeg'>
      <h1 style='Padding:10px; text-align:center; border-radius:
10px; width: 75%; color: #D5FF00;position: relative;margin-left:
auto ;margin-right: auto ;'> Sorry, Shounaks and non-coders not
allowed.  </h1>";
        }
    }
}

$link->close();

?>

</body>
</html>
```