Code Review and Inspection

Group 6

Made by – Archit Jugran , 160101087

Shubham Goel , 160101083

Yagyansh Bhatia , 160101079

Index

Page #

1. Introduction

A code review involves one or more developers examining source code they didn't write and providing feedback to the authors, both negative and positive. Ideally the reviewers are completely disengaged from the project they are reviewing as this maximizes objectivity and ensures the code is readable and maintainable even by those not already well-versed in that project. Typically the reviewers will have a standard checklist as a guide for finding common mistakes and to validate the code against the company's coding standards.

Goals

The main code-review objectives are:

1. **Best Practice** ~ A more efficient, less error-prone, or more elegant way to accomplish a given task.

2. **Error Detection** ~ Discovering logical or transitional errors.

3. **Vulnerability Exposure** ~ Identifying and averting common vulnerabilities like Cross-Site Scripting [XSS], Injection, Buffer Overflow, Excessive Disclosure, etc. Although many controls are inapplicable and can be ignored, a STIG [e.g., Application Security STIG 4.3] provides an excellent vulnerability checklist.

4. **Malware Discovery** ~ This often-overlooked and very special code-review objective looks for segments of code that appear extraneous, questionable, or flat-out weird. The intent is to discover back doors, Trojans, and time bombs. In today's world malevolent code is a very real threat and should not be overlooked, especially by Government agencies

The aim of code inspection is to discover some common types of errors caused due to oversight and improper programming. In other words, during code inspection the code is examined for the presence of certain kinds of errors, in contrast to the hand simulation of code execution done in code walk throughs. For instance, consider the classical error of writing a procedure that modifies a formal parameter while the calling routine calls that procedure with a constant actual parameter. It is more likely that such an error will be discovered by looking for these kinds of mistakes in the code, rather than by simply hand simulating execution of the procedure. In addition to the commonly made errors, adherence to coding standards is also checked during code inspection. Good software development companies collect statistics regarding different types of errors commonly committed by their engineers and identify the type of errors most frequently committed. Such a list of commonly committed errors can be used during code inspection to look out for possible errors.

Following is a list of some classical programming errors which can be checked during code inspection:

- Use of uninitialized variables.
- Jumps into loops.
- Nonterminating loops.
- Incompatible assignments.
- Array indices out of bounds.
- Improper storage allocation and deallocation.

- Mismatches between actual and formal parameter in procedure calls.
- Use of incorrect logical operators or incorrect precedence among operators.
- Improper modification of loop variables.
- Comparison of equally of floating point variables, etc.

2. CODE

2.1 Module MainActivity

```
/*
• Name of the module : MainActivity.java
• Date on which the module was created : 24/03/2018
• Author's name : Archit Jugran
• Modification history : By Yagyansh Bhatia 28/04/2018
                         By Shubham Goel 30/04/2018
• Synopsis of the module : The login-register module for logging in an
already registered user or registering a new user
• Functions :   protected void onCreate(Bundle savedInstanceState)
                protected void signout (View view)
                public void another(View v)
                public void onActivityResult(int requestCode, int resultCode,
Intent data)
                protected void onResume()
                protected void onPause()
• Global variables accessed/modified by the module : None (all variables are
private) */

package com.example.shubham.actualproject;
//Importing the required packages and functionalities
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import com.firebase.ui.auth.AuthUI;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.FirebaseDatabase;
import android.widget.ArrayAdapter;
import java.util.Arrays;

public class MainActivity extends AppCompatActivity {
    private String Username;
```

```
    private FirebaseAuth FirebaseAuth;
    private ChildEventListener ChildEventListener;
    private FirebaseAuth.AuthStateListener AuthStateListener;
    private FirebaseDatabase Database ;//initializing FirebaseDatabase object
    private static final int Id=1234; // Notification id
    private static final int RC_SIGN_IN = 123;//a request code you define to
identify the request when the result is returned to your app in
onActivityResult(...)
    /*
        onCreate() just manages the login / register flow using the pre-built
Firebase UI
     */
    @Override
    protected void onCreate(Bundle savedinstancestate) {
        super.onCreate(savedinstancestate);
        setContentView(R.layout.activity_main);//setting the view to
activity_main.xml
        Database = FirebaseDatabase.getInstance();
        FirebaseAuth=FirebaseAuth.getInstance();
        AuthStateListener = new FirebaseAuth.AuthStateListener()
{//AuthStateListener listens for change in auth state
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth
firebaseAuth) {//function called if authState of user changes
                FirebaseUser user = firebaseAuth.getCurrentUser();
                if (user != null) {
                    // User is signed in
                    Username = user.getDisplayName();
                    Toast.makeText(MainActivity.this, "Welcome, You're now
signed in.", Toast.LENGTH_SHORT).show();
                } else {
                    // User is signed out
                    startActivityForResult(
                            AuthUI.getInstance()
                                    .createSignInIntentBuilder()//creating
the sign in intent
                                    .setIsSmartLockEnabled(false)//smart lock
not enabled
                                    .setAvailableProviders(Arrays.asList(
                                            //You can enable sign-in
providers like Google Sign-In by calling the setAvailableProviders method:
                                            new
AuthUI.IdpConfig.EmailBuilder().build(),
                                            new
AuthUI.IdpConfig.GoogleBuilder().build()))
                                    .build(),
                            RC_SIGN_IN);
                }
            }
        };


    }

    /* This function Transfers control to new activity on press of users
button
 */
    public void another(View v){
```

```
        EditText position=(EditText)findViewById(R.id.position);
        EditText room=(EditText)findViewById(R.id.room);
        Intent i =new Intent(this,Display.class);
        i.putExtra("position", position.getText().toString());//sending
variable position to the new intent
        i.putExtra("room",room.getText().toString());//sending variable room
to the new intent
        startActivity(i);
    }

    /* This function signs out the user and starts MainActivity flow i.e.
redirects to the signin /register page
     */
    protected void signout (View view)
    {
        AuthUI.getInstance().signOut(this);
        Intent i =new Intent(this,MainActivity.class);
        startActivity(i);//starting activity from here
        Username="NULL";//update the value of username after signout
    }

    /* This onResume() function means that if we resume code and the user is
logged in ,
    we attach an AuthStateListener which listens for change in authentication
     */
    @Override
    protected void onResume() {
        super.onResume();
        FirebaseAuth.addAuthStateListener(AuthStateListener);
    }

    /* This onPause() function means that if we pause code and irrespective
of the  user being logged in/out ,
    `` we remove the earlier attached AuthStateListener
     */
    @Override
    protected void onPause() {
        super.onPause();
        if (AuthStateListener != null) {
            FirebaseAuth.removeAuthStateListener(AuthStateListener);
        }
    }

    //The authentication flow provides several response codes of which the
most common are as follows: Activity.RESULT_OK if
    // a user is signed in, Activity.RESULT_CANCELED if the user manually
canceled the sign in, ErrorCodes.NO_NETWORK if sign
    // in failed due to a lack of network connectivity, and
ErrorCodes.UNKNOWN_ERROR for all other errors. Typically, the only
    // recourse for most apps if sign in fails is to ask the user to sign in
again later, or proceed with anonymous sign-in if supported.
    @Override
    public void onActivityResult(int requestcode, int resultcode, Intent
data) {
        super.onActivityResult(requestcode, resultcode, data);
        if (requestcode == RC_SIGN_IN) {
            if (resultcode == RESULT_OK) {
                // Sign-in succeeded, set up the UI
```

```
                Toast.makeText(this, "Signed in!",
Toast.LENGTH_SHORT).show();
            } else if (resultcode == RESULT_CANCELED) {
                // Sign in was canceled by the user, finish the activity
                Toast.makeText(this, "Sign in canceled",
Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }
}
```

2.2 Module Display.java

```
/*
• Name of the module : Display.java
• Date on which the module was created : 01/04/2018
• Author's name : Archit Jugran
• Modification history : By Yagyansh Bhatia 04/04/2018
                         By Shubham Goel 05/04/2018
• Synopsis of the module : This module manages generation of notifications on
student's device and just subscribes teacher's device for generating notifications
from another module(when 50% of students have attention level low) , also sends data
to logmessage.java and users.java from the database for printing.
• Functions :   protected void onCreate(Bundle savedInstanceState)
                protected void signout (View view)
                public void Log(View v)
                public void onActivityResult(int requestCode, int resultCode, Intent
data)
                protected void onResume()
                protected void userlist(View view)
                public void generateStudentNotification (View view)
                protected void onPause()
                public void log_store()
                public void periodic_clock_generator()
                public int random_number()
                public void message_generator()
                public void alert_decider()
                public void stateupdate()

• Global variables accessed/modified by the module : None (all variables are private)
*/
package com.example.shubham.actualproject;
//Importing the required packages and functionalities
import android.app.Activity;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Handler;
import android.support.annotation.NonNull;
import android.os.Bundle;
import android.support.v4.app.NotificationCompat;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import com.firebase.ui.auth.AuthUI;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
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;
import com.google.firebase.messaging.FirebaseMessaging;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;

public class Display extends Activity {
    private int prevrange;
    private int counter=0;
    private String Username="NULL";
    private FirebaseAuth FirebaseAuth;// declaring  a FirebaseAuth object
    private FirebaseAuth.AuthStateListener AuthStateListener;//mAuthStateListener
detects the current login/logout status of user
    private FirebaseDatabase Database ;//declaring FirebaseDatabase object
```

```
    private DatabaseReference MyRef,CountRef;//declaring two Database references
    NotificationCompat.Builder Notification;
    private static final int Id=1234; // Notification id
    private String Room;//room stores the room number in which the student is sitting
or teacher is teaching
    private String Position;//position stores either "Teacher" or "Student"
    private ArrayList<String> Log_List = new ArrayList<String>();//to store a list of
log messages for the teacher
    private ArrayList<String> Users_List = new ArrayList<String>();//to store a list
of students sitting in the current room
    private static final int RC_SIGN_IN = 123;//a request code you define to identify
the request when the result is returned to your app in onActivityResult(...)
    private String strtitle="",strtext="";
    private int prevvalue;

    /*
     On starting the activity Display.java , a random initial state is assigned to the
student , the function
     for generating notifications is called periodically using a Runnable . Also
contained are codes which will get
     executed on any changes made to the database
     */
    @Override
    protected void onCreate(Bundle savedinstancestate) {
        super.onCreate(savedinstancestate);
        setContentView(R.layout.display);//setting the view to Display.xml
        Position = getIntent().getExtras().getString("position");//getting the
position variable from Main_Activity.java
        Room = getIntent().getExtras().getString("room");//getting the room variable
from Main_Activity.java
        Database = FirebaseDatabase.getInstance();//declaring an instance of
FirebaseDatabase
        MyRef = Database.getReference(Room).child("notifications");//myRef now points
to /room/notifications/ in database
        CountRef=Database.getReference(Room).child("users");//countRef now points to
/room/users/ in database
        if(Position.equalsIgnoreCase("teacher")){
            Button logbutton = (Button) findViewById(R.id.log3);
            logbutton.setVisibility(View.VISIBLE);//making the log button visible only
if logged in user is the teacher
        }
        prevvalue = random_number();
        Notification = new NotificationCompat.Builder(this);
        Notification.setAutoCancel(true);//ensuring that on clicking notification,it
disappears
        FirebaseAuth=FirebaseAuth.getInstance();//get an instance of firebase auth
        periodic_clock_generator();
        alert_decider();
        log_store();
        AuthStateListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth)
{//function called if authState of user changes
                FirebaseUser user = firebaseAuth.getCurrentUser();
                if (user != null) {
                    // User is signed in
                    Username = user.getDisplayName();
                    Toast.makeText(Display.this, "Welcome to Dashboard",
Toast.LENGTH_SHORT).show();
                } else {
                    // User is signed out
                    startActivityForResult(
                            AuthUI.getInstance()
                                    .createSignInIntentBuilder()//creating the sign in
```

```
intent
                                        .setIsSmartLockEnabled(false)//smart lock not
enabled
                                        .setAvailableProviders(Arrays.asList(
                                                //You can enable sign-in providers like
Google Sign-In by calling the setAvailableProviders method:
                                                new
AuthUI.IdpConfig.EmailBuilder().build(),
                                                new
AuthUI.IdpConfig.GoogleBuilder().build()))
                                        .build(),
                                RC_SIGN_IN);
                    }
                }
            };

        }


    /* This function generates the log to be displayed to the teacher from the database
         */
        public void log_store(){
            ValueEventListener evventListener = new ValueEventListener() {
                @Override
                //This method will be called with a snapshot of the data at this location.
It will also be called each time that data changes.
                public void onDataChange(DataSnapshot datasnapshot) {
                    Log_List.clear();
                    for(DataSnapshot ds : datasnapshot.getChildren()) {//iterating over
database at myref's location
                        String latestnoti = ds.getValue(String.class);//getting all
notifications one by one
                        if(latestnoti.equalsIgnoreCase("junk"))//if it is a valid
notification ,then add to log_list
                            ;
                        else
                            Log_List.add(latestnoti);
                    }
                }

                @Override
                public void onCancelled(DatabaseError databaseerror) {}
            };
            MyRef.addValueEventListener(evventListener);
        }

    /* This function generates clock periodically after an interval of 10seconds
         */
        public void periodic_clock_generator(){
            final Handler handler = new Handler();// two main uses for a Handler: (1) to
schedule messages and runnables to be executed as some point in the future; to enqueue
an action to be performed on a different thread than your own.
            final Button btn = (Button) findViewById(R.id.button);
            Runnable run = new Runnable() {//creating a runnable object in order to run
piece of code after every fixed interval
                @Override
                public void run() {

                    handler.postDelayed(this,10000);//this code will get executed after a
delay of 10000ms = 10s
                    if(Username.equalsIgnoreCase("NULL"))
                        ;
                    else
                        btn.performClick();//if user logged in, then only perform button
```

```
click, equivalent to calling function func
            }
        };
        handler.post(run);
    }



/* This function decides whether or not alert level is 50% or below and accordingly
subscribes teacher for receiving notification or not
     */
    public void alert_decider(){
        ValueEventListener eventListener = new ValueEventListener() {
            @Override
            //This method will be called with a snapshot of the data at this location.
It will also be called each time that data changes.
            public void onDataChange(DataSnapshot datasnapshot) {//if any changes to
data referenced by countRef, this code will execute
                int cntlow=0,totalcnt=0;//cntlow will store total number of students
whose attention is low,
                // totalcnt stores total number of students
                Users_List.clear();//emptying the users list
                for(DataSnapshot ds : datasnapshot.getChildren()) {//iterating over
database at countref's location
                    totalcnt++;
                    String latestnoti =
ds.child("latestnotification").getValue(String.class);//storing the latestnotification
field in ln
                    int len=latestnoti.length();
                    int currentstate=(int)(latestnoti.charAt(len-3)-'0');//extracting
the current state of user from his latest notification
                    if(currentstate==0)
                        cntlow++;//if attention low, increment
                    Users_List.add("Name : "+ds.child("name").getValue(String.class)+
" || Current State : " +currentstate);//update users list
                }
                if(Position.equalsIgnoreCase("teacher")) {
                    if (cntlow * 1.0 / totalcnt >= 0.5) {//if more than 50% of
students have low attention

FirebaseMessaging.getInstance().subscribeToTopic("lowlow"+Room);//subscribing to topic
lowlow so that cloud function can send the teacher the notification according to his
topic

Database.getReference("cloudtriggerlow").push().setValue(Room);//pushing a junk value
here to trigger the firebase cloud function written in JS
                    } else {

FirebaseMessaging.getInstance().subscribeToTopic("highhigh"+Room);//subscribing to
different topic just to remove subscription from lowlow

Database.getReference("cloudtriggerhigh").push().setValue(Room);//pushing a junk value
here to trigger the firebase cloud function written in JS
                    }
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {}
        };
        CountRef.addValueEventListener(eventListener);
    }
```

```
/* This function Transfers control to new activity on press of users button
     */

    protected void userlist(View view)
    {
        Intent intentt =new Intent(this,users.class);
        intentt.putExtra("room",Room.toString());//sending variable room to the new
intent
        intentt.putExtra("mylist",Users_List);//sending list of users to the new
intent
        startActivity(intentt);
    }

    /* This function signs out the user and starts MainActivity flow i.e. redirects to
the signin /register page
     */
    protected void signout (View view)
    {
        AuthUI.getInstance().signOut(this);
        Intent i =new Intent(this,MainActivity.class);
        startActivity(i);//starting activity from here
        Username="NULL";//update the value of username after signout
    }

    /* This function Transfers control to new activity on press of log button
     */
    public void Log(View view)
    {
        Intent intentt=new Intent(this,logmessage.class);
        intentt.putExtra("room",Room.toString());//sending variable room to the new
intent
        intentt.putExtra("mylist",Log_List);//sending list of notifications to the new
intent
        startActivity(intentt);//starting activity from here
    }

    /* This function is responsible for generating notifications periodically( since
this  function is called periodically)
    Also pushes relevant data to the Firebase database as well as transfers control to
another activity on tapping
    notification
     */
    public void generateStudentNotification (View view)
    {
        int newstate = random_number();
        message_generator(newstate,prevvalue);
        prevvalue=newstate;//current range is stored in prevrange for the next
iteration of above code
        if(strtitle.equalsIgnoreCase(""))
            return;
        notificationsender();
    }

    public int random_number() {
        Random rand = new Random();
        int newstate = rand.nextInt(10);
        return newstate;
    }
    public void notificationsender() {
        Notification.setSmallIcon(R.drawable.ic_launcher_background);
        Notification.setTicker("ticker");
        Notification.setWhen(System.currentTimeMillis());//view the notification
instantaneosly
        Notification.setContentTitle(Username+" "+strtitle);
```

```
        Notification.setContentText(Username+" "+strtext);
        Intent intent=new Intent(this,Confirm.class);//to ensure that on clicking
notification,app opens
        String confirmednotification=counter + " "+Username + " " + strtext;
        intent.putExtra("notification",confirmednotification.toString());
        intent.putExtra("room",Room.toString());
        PendingIntent
pendingIntent=PendingIntent.getActivity(this,0,intent,PendingIntent.FLAG_UPDATE_CURREN
T);
        Notification.setContentIntent(pendingIntent);
        NotificationManager nf=(NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        if(Position.equalsIgnoreCase("student")) {
            nf.notify(Id, Notification.build());//sending notificaiton
            stateupdate();
        }
    }


/* This function updates the state of a student in the state table of database
whenever there is a change in state of student
     */
    public void stateupdate(){
        FirebaseMessaging.getInstance().subscribeToTopic("student");
        CountRef.child(Username).child("latestnotification").setValue(strtext);
        CountRef.child(Username).child("name").setValue(Username);
        MyRef.push().setValue(counter+" "+Username + " " + strtext);//pushing
notification to database
        counter++;
    }


    /* This function generates a message to be sent to student later as a notification
.Message is generated according to newstate and previous state of user
     */
    public void message_generator (int newstate,int prevstate) {

        int newrange,prevrange;
                /*
         on basis of range of prevvalue, assigning a range to the state according to
the following table
            Interpretation  | State
                    Low     | 1 to 4
                    Medium  | 5 to 7
                    High    | 8 to 10
          */
        if(newstate<5){
            newrange=0;
        }
        else if(newstate<8) {
            newrange=1;
        }
        else
            newrange=2;
        if(prevstate<5){
            prevrange=0;
        }
        else if(prevstate<8) {
            prevrange=1;
        }
        else
            prevrange=2;
//        String strtitle = "", strtext = "";
        if (newrange > prevrange)//changing notification title based on newstate and
```

```
previous range
        {
            strtitle = "Increase in attention";
            strtext = "Keep it up " + newrange + " " + prevrange;
        } else if (newrange < prevrange) {
            strtitle = "Decrease in attention";//notification title
            strtext = "Warning: Please pay more attention " + newrange + " " +
prevrange;//notification text
        } else {
            strtitle="";
            strtext="";
        }
    }


/* This onResume() function means that if we resume code and the user is logged in ,
    we attach an AuthStateListener which listens for change in authentication
     */
    @Override
    protected void onResume() {
        super.onResume();
        FirebaseAuth.addAuthStateListener(AuthStateListener);
    }

    /* This onPause() function means that if we pause code and irrespective of the
user being logged in/out ,
    `` we remove the earlier attached AuthStateListener
     */
    @Override
    protected void onPause() {
        super.onPause();
        if (AuthStateListener != null) {
            FirebaseAuth.removeAuthStateListener(AuthStateListener);
        }
    }


    /*The authentication flow provides several response codes of which the most common
are as follows: Activity.RESULT_OK if
     a user is signed in, Activity.RESULT_CANCELED if the user manually canceled the
sign in, ErrorCodes.NO_NETWORK if sign
     in failed due to a lack of network connectivity, and ErrorCodes.UNKNOWN_ERROR for
all other errors. Typically, the only
     recourse for most apps if sign in fails is to ask the user to sign in again
later, or proceed with anonymous sign-in if supported.
    */
    @Override
    public void onActivityResult(int requestcode, int resultcode, Intent data) {
        super.onActivityResult(requestcode, resultcode, data);
        if (requestcode == RC_SIGN_IN) {
            if (resultcode == RESULT_OK) {
                // Sign-in succeeded, set up the UI
                Toast.makeText(this, "Signed in!", Toast.LENGTH_SHORT).show();
            } else if (resultcode == RESULT_CANCELED) {
                // Sign in was canceled by the user, finish the activity
                Toast.makeText(this, "Sign in canceled", Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }
}
```

2.3 Module Confirm.java

```
/*
• Name of the module : Confirm.java
• Date on which the module was created : 08/04/2018
• Author's name : Archit Jugran
• Modification history : By Yagyansh Bhatia 10/04/2018
                         By Shubham Goel 13/04/2018
• Synopsis of the module : Confirms the notification on which student tapped by
changing the
                           corresponding entry in the log
• Functions :   protected void onCreate(Bundle savedInstanceState)
                protected void signout (View view)
                public void onActivityResult(int requestCode, int resultCode, Intent
data)
                protected void onResume()
                protected void confirmnotification(final String notification3)
                protected void onPause()
• Global variables accessed/modified by the module : None (all variables are private)
*/

package com.example.shubham.actualproject;
//Importing the required packages and functionalities
import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.view.View;
import android.widget.Toast;
import com.firebase.ui.auth.AuthUI;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
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;
import java.util.Arrays;
import java.util.Random;
public class Confirm extends Activity {
    private String notification3,room;
    private String Username;
    private FirebaseAuth FirebaseAuth;// declaring  a FirebaseAuth object
    private FirebaseAuth.AuthStateListener AuthStateListener;//AuthStateListener
detects the current
    // login/logout status of user
    private FirebaseDatabase Database ;//initializing FirebaseDatabase object
    private DatabaseReference MyRef;//declaring DatabaseReference
    private static final int Id=1234; // Notification id
    private int Flag=0;
    private static final int RC_SIGN_IN = 123;//a request code you define to identify
the request
    // when the result is returned to your app in onActivityResult(...)

    /* This function changes the current notification's (which was tapped)status to
Confirmed*/
    @Override
    protected void onCreate( Bundle savedinstancestate){
        super.onCreate(savedinstancestate);
```

```
        setContentView(R.layout.confirm);
        notification3=getIntent().getExtras().getString("notification");
        room=getIntent().getExtras().getString("room");
        Database = FirebaseDatabase.getInstance();
        MyRef = Database.getReference(room).child("notifications");
        FirebaseAuth=FirebaseAuth.getInstance();
        confirmnotification(notification3);

        AuthStateListener = new FirebaseAuth.AuthStateListener() {//AuthStateListener
listens for change in auth state
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth)
{//function called if authState of user changes
                FirebaseUser user = firebaseAuth.getCurrentUser();
                if (user != null) {
                    // User is signed in
                    Username = user.getDisplayName();
                    Toast.makeText(Confirm.this, "Notification confirmed",
Toast.LENGTH_SHORT).show();
                } else {
                    // User is signed out
                    startActivityForResult(
                            AuthUI.getInstance()
                                    .createSignInIntentBuilder()//creating the sign in
intent
                                    .setIsSmartLockEnabled(false)//smart lock not
enabled
                                    .setAvailableProviders(Arrays.asList(
                                            //You can enable sign-in providers like
Google Sign-In by calling the setAvailableProviders method:
                                            new
AuthUI.IdpConfig.EmailBuilder().build(),
                                            new
AuthUI.IdpConfig.GoogleBuilder().build()))
                                    .build(),
                            RC_SIGN_IN);
                }
            }
        };
        /* Redirecting activity to Display.java after notification gets confirmed*/
        Intent intentt = new Intent(this,Display.class);
        intentt.putExtra("position","student");
        intentt.putExtra("room",room);
        startActivity(intentt);
    }

   // This function is responsible for changing confirmation status of the current
notification
    protected void confirmnotification(final String notification3)
    {
        ValueEventListener evventListener = new ValueEventListener() {
            //This method will be called with a snapshot of the data at this location.
It will also
            //  be called each time that data changes.
            @Override
            public void onDataChange(DataSnapshot datasnapshot) {//if any changes to
data referenced
                // by MyRef, this code will execute
                if(Flag==0){//Flag ensures that this code snippet does not get
executed again and
                    // again after status has been changed to confirmed
                    for (DataSnapshot ds : datasnapshot.getChildren()) {
                        String notificationn = ds.getValue(String.class);//iterating
through all notifications
```

```
                        if (notification3.equalsIgnoreCase(notificationn)) {//if
notification found
                            DatabaseReference ref = ds.getRef();
                            ref.setValue("Confirmed : " +
notificationn);//concatenating Confirmed status to it
                            Flag = 1;
                        }
                    }
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseerror) {}
        };
        MyRef.addValueEventListener(evventListener);//associating valueEventListener
with MyRef
        MyRef.push().setValue("junk");//just pushing a junk value so that the above
onDataChange() gets called and i can modify
        //the current notification to make its status  confirmed
    }

    /* This function signs out the user and starts MainActivity flow i.e. redirects to
the signin /register page
     */
    protected void signout (View view)
    {
        AuthUI.getInstance().signOut(this);
        Intent i =new Intent(this,MainActivity.class);
        startActivity(i);//starting activity from here
        Username="NULL";//update the value of username after signout
    }

    /* This onResume() function means that if we resume code and the user is logged in
,
    we attach an AuthStateListener which listens for change in authentication
     */
    @Override
    protected void onResume() {
        super.onResume();
        FirebaseAuth.addAuthStateListener(AuthStateListener);
    }

    /* This onPause() function means that if we pause code and irrespective of the
user being logged in/out ,
    `` we remove the earlier attached AuthStateListener
     */
    @Override
    protected void onPause() {
        super.onPause();
        if (AuthStateListener != null) {
            FirebaseAuth.removeAuthStateListener(AuthStateListener);
        }
    }

    //The authentication flow provides several response codes of which the most common
are as follows: Activity.RESULT_OK if
    // a user is signed in, Activity.RESULT_CANCELED if the user manually canceled the
sign in, ErrorCodes.NO_NETWORK if sign
    // in failed due to a lack of network connectivity, and ErrorCodes.UNKNOWN_ERROR
for all other errors. Typically, the only
    // recourse for most apps if sign in fails is to ask the user to sign in again
later, or proceed with anonymous sign-in if supported.
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == RC_SIGN_IN) {
            if (resultCode == RESULT_OK) {
                // Sign-in succeeded, set up the UI
                Toast.makeText(this, "Signed in!", Toast.LENGTH_SHORT).show();
            } else if (resultCode == RESULT_CANCELED) {
                // Sign in was canceled by the user, finish the activity
                Toast.makeText(this, "Sign in canceled", Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }
}
```

2.4 Module users.java

```
/*
• Name of the module : users.java
• Date on which the module was created : 08/04/2018
• Author's name : Shubham Goel
• Modification history : By Yagyansh Bhatia 10/04/2018
                         By Archit Jugran 12/04/2018
• Synopsis of the module :
• Functions : protected void onCreate(Bundle savedInstanceState)
• Global variables accessed/modified by the module : None (all variables are private)
*/

package com.example.shubham.actualproject;
//Importing the required packages and functionalities
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import java.util.ArrayList;

public class users extends Activity {
    private ArrayList<String> Users_list = new ArrayList<String>();
    private ListView Lv;
    /*
     This function displays list of students in room to all users
     */
    @Override
    protected void onCreate(Bundle savedinstancestate) {
        super.onCreate(savedinstancestate);
        setContentView(R.layout.users);
        String room=getIntent().getExtras().getString("room");
        Users_list = (ArrayList<String>)
getIntent().getSerializableExtra("mylist");//getting a list from previous activity
        Lv = (ListView) findViewById(R.id.idbitch);//accessing list view of XML file
        // This is the array adapter, it takes the context of the activity as a
        // first parameter, the type of list view as a second parameter and your
        // array as a third parameter.
        ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(
                this,
                android.R.layout.simple_list_item_1,//id of list view in xml file
                Users_list );

        Lv.setAdapter(arrayAdapter);//setting the ListView in xml file to display the
list log_list
    }
}
```

2.5 Module logmessage.java

```
/*
• Name of the module : logmessage.java
• Date on which the module was created : 06/04/2018
• Author's name : Yagyansh Bhatia
• Modification history : By Archit Jugran 07/04/2018
                         By Shubham Goel 08/04/2018
• Synopsis of the module : This module is accessed by the teacher by pressing the log
button and this module is responsible
                           for displaying the log of notifications delivered to
students along with confirmation status
• Functions :   protected void onCreate(Bundle savedInstanceState)
• Global variables accessed/modified by the module : None (all variables are private)
*/

package com.example.shubham.actualproject;
//Importing the required packages and functionalities
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;

public class logmessage extends Activity {
    private ArrayList<String> Log_List = new ArrayList<String>();// an empty dynamic
array of strings declared
    private ListView Lv;//declaring a ListView variable

    /*  This function displays log of messages(notifications) to teacher
     */
    @Override
    protected void onCreate(Bundle savedinstancestate){
        super.onCreate(savedinstancestate);
        setContentView(R.layout.logmessage);
        String room=getIntent().getExtras().getString("room");
        Log_List = (ArrayList<String>)
getIntent().getSerializableExtra("mylist");//getting a list from previous activity
        Lv = (ListView) findViewById(R.id.idbitch);//accesssing list view of XML file

        // This is the array adapter, it takes the context of the activity as a
        // first parameter, the type of list view as a second parameter and your
        // array as a third parameter.
        ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(
                this,
                android.R.layout.simple_list_item_1,//id of list view in xml file
                Log_List );
        Lv.setAdapter(arrayAdapter);//setting the ListView in xml file to display the
list log_list

    }

}
```

2.6 Module index.js

```
/*
• Name of the module : index.js
• Date on which the module was created : 10/04/2018
• Author's name : Archit Jugran
• Modification history : By Shubham Goel 11/04/2018
• Synopsis of the module : This module makes the server send notification
    to the teacher's device when 50 % of the students' attention falls low .
• Different functions supported, along with their input/output parameters :
   functions.database.ref('cloudtriggerlow').onWrite((event))
        --> Input : event
        --> Output : admin.messaging().sendToTopic("lowlow")
• Global variables accessed/modified by the module : NIL */

//Importing required libraries
const functions = require('firebase-functions');
const admin = require('firebase-admin');
admin.initializeApp(functions.config().firebase);
/* This function gets triggered whenever data is written to "cloudtriggerlow" branch
    of database . When 50% class attention becomes low, teacher is subscribed to the topic
    "lowlow" in java code and from here i am sending notification to the devices subscribed
    to topic "lowlow" */
exports.sendNotification = functions.database.ref('cloudtriggerlow').onWrite((event) => {
    const data = event.data;
    console.log('Message received');
    if(!data.changed()){//if no data changed
        console.log('Nothing changed');
        return;
    }else{//otherwise just logging the data for testing purposes
        console.log(data.val());
    }
    const payLoad = {
        notification:{
            title: 'ATTENTION',//title of notification
            body: '50% class down',//text of notification
            sound: "default"
        }
    };
    const options = {
        priority: "high",//setting priority of it to be delivered high
        timeToLive: 60*60*2
    };
    return admin.messaging().sendToTopic("lowlow", payLoad, options);//sending notification

});
```

3. Code inspection and walkthrough

3.1 Code inspection team

The code testing team comprises of the following members, all of whom are 2nd Year Undergraduates currently pursuing Bachelor of Technology at Indian Institute of Technology Guwahati, India in the Department of Computer Science & Engineering. Team members are

a) Shaurya Gomber, Roll no. 160101086
b) Rishabh Jain , Roll no. 160101088
c) Aadil Hoda , Roll no. 160101001

3.2 Report by Shaurya Gomber

- Coding conventions followed are : Class variables' naming convention is of format that every word will have first letter capital, local variables and function parameters are named with all small letters.
- The code is well documented explaining almost every line of code
- Variable are named meaningfully
- On line 98 of module Display.java, a deprecated function NotificationCompat.Builder() is used .
- In all modules, all the variables were private to that class and hence common to all the functions of that class only. So none of the variables are global.
- The code is properly indented.
- No jump (goto) statements used in any of the modules.
- No uninitialized variables found.
- No non terminating loops found.
- Some unnecessary packages were imported by the code.
- In line 20 of module users.java , proper naming convention has not been followed. Variable named as Users_list instead of Users_List.
- For generating the notification inside the runnable after every fixed number of seconds, instead of using a button to call the function , the function could have been called directly
- Naming the local variables and the function parameters in the same way is a bit confusing.
- In module display.java , proper naming convention has not been followed for function : public void Log (View view)
- In function alertdecider() of module display.java , function parameter does not follow naming convention . DatabaseError databaseError should have been DatabaseError databaseerror
- In module display.java , camelCase has not been followed while naming the function userlist().
- Headers for each module have been made properly and enhance readability of the code
- The flow of code is logical and understandable.
- In Confirm.java, class variable "flag" does not follow naming convention

3.3 Report by Rishabh Jain

- The code has been properly explained by comments.
- Coding conventions followed are : Class variables' naming convention is of format that every word will have first letter capital, local variables and function parameters are named with all small letters.
- Some import statements were useless as the classes and functions being imported have not been used anywhere.
- All loops found terminated after a finite number of iterations.
- In Display.java, the function NotificationCompat.Builder() that has been used is outdated and has been replaced by a newer function which should have been used instead.
- Some functions have too many lines of source code. More decomposition of functions should have been done.
- No jump (goto) statements used in any of the modules
- All variables have been initialized.
- In users.java , variable Users_list has not been named properly according to the convention.
- Same naming convention for local variables and function parameters has been used.
- Headers for each module have been properly made.
- On module display.java , function name Log() does not follow proper convention.
- Module display.java has a function parameter in function alertdecider() which does not follow naming convention . DatabaseError databaseError
- In module display.java , function userlist() should have been named as userList().
- The code is well indented.

3.4 Report by Aadil Hoda

- Internal documentation is good and explains the code well.
- Coding conventions followed are : Class variables' naming convention is of format that every word will have first letter capital, local variables and function parameters are named with all small letters.
- Jump or go-to statements have not been used.
- Headers follow all guidelines and are well made.
- Some import statements used in the code were unnecessary.
- All the loops will terminate according to their terminating conditions respectively after some finite iterations
- Naming convention for local variables and function parameters is same.
- The logic of code is sequential or iterative , as jump statements not used.
- Variables initialized properly.
- In module display.java , proper naming convention has not been followed for function : public void Log (View view). It should have been public void log(..).
- In function onCancelled() of function alertdecider() of display.java , function parameter "databaseerror"does not follow naming convention .
- In module display.java ,function userlist() not named according to the convention to be followed.
- A pre-built function used in display.java called NotificationCompat.builder() is deprecated and no longer in use.
- The code is well indented
- In Confirm.java, class variable "flag" should have been named "Flag"

3.4 Conclusion

- Naming the local variables and function parameters in the same manner is a bit confusing.
- Improper naming convention followed for function parameter on line 143 on Display.java
- Improper naming convention followed for function name on line 215 of Display.java
- Improper naming convention followed for class variable “flag” of Confirm.java
- Improper naming convention followed for class variable “Users_list” of users.java
- Improper naming convention followed for function name Log in Display.java
- Unnecessary import statements need to be removed