CODE TESTING DOCUMENT

Prepared for Software Engineering Project - "Attento" version 1.3

Prepared by:- Group 11

Gourika Bang 150101025

Saurabh 150101062

Soumik Roy 150101074

6 May, 2017

Supervised by:-

Prof. Samit Bhattacharya

CONTENTS

REVISION HISTORY

S.No.	Date	Reason for modification	Version
1	25/04/2017	Original Document	1.0
2	03/05/2017	Update in original document	1.1
3	05/05/2017	Remarks from Prof. Bhattacharya	1.2
4	06/05/2017	Final Edits	1.3

UNIT TESTING

1. Models on which App was tested

- Vivo v3
- Xiomi Redmi Note 3

Limitation of the mobile sensor:
The proximity sensor detected only boolean values 0 or 1.

2. BLACK BOX TESTING

2.1 Functional Testing

In this testing, test cases are designed from an examination of the input/output values only.

2.1.1 Login Form

Precondition: The app is opened.

2.1.1.1 Equivalence class partitioning

S.No	Class	Test Cases	Expected Results	Actual Results
1	Valid login credentials for student	1. Enter correct username and password for the student and click on login	The system directs to student's main page	The system directs to student's main page
2	Valid login credentials for professor	1. Enter correct username and password for the professor and click on login	The system directs to professor's main page	The system directs to professor's main page
3	Invalid login credentials for either of them	1. Enter invalid username and click on login 2. Enter invalid password and click on login 3. Username and password entered don't match and click on login	The system displays "wrong login credentials" and redirect to the login page.	The system displays "Incorrect Login" and stays on the same page.

2.1.1.2 Boundary Value Analysis

S.no.	Test case	Expected Result	Actual Result
1	Username field is left blank and click on login	The system displays "Please fill this field"	The system displays "Incorrect Login" and stays on the same page.
2	Password field is left blank and click on login	The system displays "Please fill this field"	The system displays "Incorrect Login" and stays on the same page.

2.1.2 Starting the server

Precondition: Login module is unit tested

2.1.2.1 Equivalence Class Partitioning

S.No.	Class	Test case	Expected Result	Actual Result
1	Session is started	1.Click on "Start Session" and no server issues	The system directs to Professor's home page and the session gets started	The system directs to Professor's home page and the session gets started
2	Session is not started	1.Server is shut down and click on "Start Session" 2. "Start Session" is not clicked.	The system prompts to click again	The app stops working.

2.1.3 Connection to server

Precondition: Login module and Starting the server module is unit tested

2.1.3.1 Equivalence Class Partitioning

S.No.	Class	Test case	Expected Result	Actual Result
1	Connection is established	1. The session is started and click on "Connect to session"	The system directs to the student's home page and connects him to the session	The system directs to the student's home page and connects him to the session
2	Connection is not established	1.The server is shut down and click on "Connect to session" 2.The session is not started and click on "Connect to session" 3. "Connect to session" is not clicked	The system prompts to click again	The app stops working.

2.1.4 For student home page

Precondition: The student and professor are connected to the server.

2.1.4.1 Status Display

2.1.4.1.1 Equivalence Class Partitioning

S.No.	Class	Test case	Expected Result	Actual Result
1	Ideal condition of the device	1.The ideal condition of the device is maintained a) distance between user and device = 25 cm b) movement/shaking frequency of the smartphone <= 2 per minute c) orientation of the device: (x-axis = 45°, y-axis = 0°, z = 0°)	The system displays "Attentive" status	The system displays "Attentive" status
2	Distance between user and device is less than 25 cm	1.The distance between user and device is 18cm.	The system displays "Inattentive" status	The system displays "Attentive" status
3	Distance between user and device is more than 25 cm	1.The distance between user and device is 30 cm	The system displays "Inattentive" status	The system displays "Attentive" status
4	Movement/shaking frequency is more than 2 per minute	1.The shaking frequency of the device is 5 per minute	The system displays "Inattentive" status	The system displays "Inattentive" status
5	The orientation of the device is less than 45° along x-axis	1. The orientation of the device is 30° along x-axis	The system displays "Inattentive" status	The system displays "Inattentive" status
6	The orientation of the device is more than 45° along x-axis	1. The orientation of the device is 60° along x-axis	The system displays "Inattentive" status	The system displays "Inattentive" status
7	The device is oriented at a positive angle across y-axis	1. The orientation of the device is 90° along y-axis	The system displays "Inattentive" status	The system displays "Inattentive" status
8	The device is oriented at a positive angle across z-axis	1. The orientation of the device is 30° along z-axis	The system displays "Inattentive" status	The system displays "Inattentive" status

2.1.4.1.2 Boundary Value Analysis

S.No.	Test Case	Expected Result	Actual Result
1	The distance between the user and the device is 0 cm	The system displays "Inattentive" status	The system displays "Attentive" status
2	The distance between the user and the device is 100 cm	The system displays "Inattentive" status	The system displays "Attentive" status
3	The orientation along x-axis is 0°	The system displays "Inattentive" status	The system displays "Inattentive" status
4	The shaking frequency of the phone is 0 per minute	The system displays "Attentive" status	The system displays "Attentive" status

2.1.5 For professor home page

Precondition: The student and professor are connected to the server.

2.1.5.1 Help Menu

S.No	Test Case	Expected result	Actual Result
1	Click on "Help" menu on the top right corner of the professor home page	The system displays the app details	The system displays the app details

2.1.5.2 Search a particular student

2.1.5.2.1 Equivalence Class Partitioning

S.No	Class	Test Case	Expected Result	Actual Result
1	Valid Search	1. Enter the name of an enrolled student who is connected to the session and click on "Search"	The system displays the details along with the status of the searched student	It displays the name of the student and his real-time status.
2	Invalid Search	1. Enter a name of the student not enrolled to the course and click on "Search" 2. Enter the name of a student who is enrolled but not connected to the session and click on "Search"	The system displays "No results found"	The system displays "No results found"

2.1.5.2.2 Boundary Value Analysis

S.No	Test Case	Expected Result	Actual Result
1	No character is entered and click on "Search"	The system displays "Please fill this field"	The system displays "No results found"

2.1.5.3 Total No of attentive students

2.1.5.3.1 Equivalence class partitioning

S.No	Class	Test Case	Expected Result	Actual Result
1	Increment in total no of attentive students	For one of connected device while keeping other values ideal do either of the following: 1.The distance is made to 25 cm 2.The orientation along x-axis is made 45°. 3.The orientation along y-axis is made 0° 4.The orientation along z-axis is made 0° 5.The shaking frequency is made 1 per minute.	The system increments by one the total no of attentive students.	The system increments by one the total no of attentive students.
2	Decrement in total no of attentive students	One of the connected device in ideal condition do either of the following: 1.Increase the distance to 30 cm 2.Decrease the distance to 20 cm 3.Increase the shaking frequency to 6 per minute 4.Change the orientation along x-axis to 60° 5.Change the orientation along x-axis to 30° 6.Change the orientation along y-axis to 60° 7.Change the orientation along z-axis to 90° 8.Disconnect to session	The system decrements by one the total no of attentive students.	For Test cases 1 and 2 it does not change the count. For other Test cases it decrements the count.

2.1.5.3.2 Boundary Value Analysis

S.No	Test Case	Expected Result	Actual Result
1	The devices connected are kept still without changing any parameter	The system must not change the total no of attentive students.	The system does not change the count

2.1.5.4 Notification alert on change in activity of a student

2.1.5.4.1 Equivalence class partitioning

S.No	Class	Test Case	Expected Result	Actual Result
1	Student activity changes from ideal	1.The distance of an ideal device is increased to 50 cm 2.Decrease the distance to 20 cm 3.Increase the shaking frequency to 6 per minute 4.Change the orientation along x-axis to 60° 5.Change the orientation along x-axis to 30° 6.Change the orientation along y-axis to 60° 7.Change the orientation along z-axis to 90°	The system sends a notification to the professor	For test cases 1 and 2 no notification is send to the professor For all other cases the professor is notified.
2	Student activity remains same	1.The connected devices remain still.	No notification regarding attentiveness is sent to the professor	No notification to the professor.

2.1.5.5 End Session and Report Generation

S.No	Test Case	Expected Result	Actual Result
1	Click on "Log Out" on the top left corner of the professor home page	The session should disconnect for all connected devices and a report regarding student activity should generate	A report about the student status is generated and the device logs out.

2.2 Performance Testing

2.2.1 Stress Testing

S. No.	Test Case	Test Result
1	10 students tried to connect to the server	The system worked fine.
2	15 students tried to connect to the server	The system stopped responding

2.2.2 Volume Testing

S. No.	Test Case	Test Result
1	When the no. of attentive students increased more than 11	The system displays and stores only 9

2.2.3 Regression Testing

- When Search Activity was added in the Professor's Home Page, no new bug was found.
- Also to fix the error of Volume Testing, Custom Array Adapter is used and Regression is run again to identify new bugs.

3. WHITE BOX TESTING - PATH COVERAGE

3.1 Search Activity

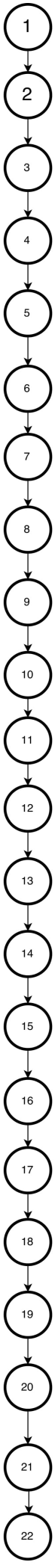
3.1.1 CFG

The CFG is drawn on the next page.

3.1.2 Test cases and Test Results

The number of Linearly Independent Paths in the CFG = 1

Path No.	Test Case	Test Result
1	Enter the name of the student that needs to be searched and click on "Search".	The system displays the result based on the query.



SEARCH ACTIVITY

```
1. public class SearchActivity extends AppCompatActivity {

2.     String urlAddress="http://10.0.2.2/android/searcher.php";
3.     SearchView sv;
4.     ListView lv;
5.     ImageView noDataImg,noNetworkImg;
  @Override
6. protected void onCreate(Bundle savedInstanceState) {
7. super.onCreate(savedInstanceState);
8.         setContentView(R.layout.activity_search);
9.         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
10.        setSupportActionBar(toolbar);
11.lv= (ListView) findViewById(R.id.lv);
12sv= (SearchView) findViewById(R.id.sv);
13sv.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
  @Override
14public boolean onQueryTextSubmit(String query) {
15.                SenderReceiver sr=new SenderReceiver(SearchActivity.this,urlAddress,query,lv,noDataImg,noNetworkImg);
16.                sr.execute();
17return false;
                }
  @Override
18public boolean onQueryTextChange(String query) {
19                 SenderReceiver sr=new SenderReceiver(SearchActivity.this,urlAddress,query,lv,noDataImg,noNetworkImg);
20                 sr.execute();
21return false;
                }
22.        });
      }

  }
```

Linearly Independent Paths:-

1) 1->2->3->4->5->6->7->8->9->10->11->12->13->14->15->16->17->18->19->20->21->22

3.2 Student Activity Display

3.2.1 CFG

The CFG is drawn on the next page.

3.2.2 Test cases and Test Results

The number of Linearly Independent Paths in the CFG = 5

Path No.	Test Case	Test Result
1	When shaking frequency <= 2 per minute and proper orientation along x, y and z axis and distance is 25cm	The system displays status of the student as "attentive"
2	When shaking frequency > 2 per minute and proper orientation along x, y and z axis and distance is 25cm	The system displays status of the student as "inattentive"
3	When shaking frequency <= 2 per minute and improper orientation along x, y and z axis and distance is 25cm	The system displays status of the student as "inattentive"
4	When shaking frequency > 2 per minute and improper orientation along x, y and z axis and distance is 25cm	The system displays status of the student as "inattentive"
5	When shaking frequency <= 2 per minute and proper orientation along x, y and z axis and distance is not 25 cm	The system displays status of the student as "inattentive"

STUDENT'S PARAMETER DISPLAY


```
public class UserActivity extends AppCompatActivity implements SensorEventListener {

    private TextView xText, yText, zText, dist, p, r, mess;
    private Sensor mySensor1, mySensor, mySensor2;
    int i=0;
    private SensorManager SM, SM1, SM2;
    float azimut, pitch, roll, Rx=0, x1, y1, z1;
    float[] mGravity;
    float[] mGeomagnetic;
    float[] mProximity;
    float count = 0;
    float count1 = 0;
    float count2 = 0;
    private boolean init;
    private static final float ERROR = (float) 7.0;
    float count_timer = 0;
    float usable_count_timer;
        Timer T;


    /**
     *
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_user);

    SM = (SensorManager)getSystemService(SENSOR_SERVICE);
    SM1 = (SensorManager)getSystemService(SENSOR_SERVICE);
    SM2 = (SensorManager)getSystemService(SENSOR_SERVICE);

    mySensor = SM.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    mySensor1 = SM1.getDefaultSensor(Sensor.TYPE_PROXIMITY);
    mySensor2 = SM2.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);

    SM.registerListener(this, mySensor, SensorManager.SENSOR_DELAY_NORMAL);
    SM1.registerListener(this, mySensor1, SensorManager.SENSOR_DELAY_NORMAL);
    SM2.registerListener(this, mySensor2, SensorManager.SENSOR_DELAY_NORMAL);


    r = (TextView)findViewById(R.id.msg7);
    mess = (TextView)findViewById(R.id.msg2);
    xText = (TextView)findViewById(R.id.msg3);
    yText = (TextView)findViewById(R.id.msg4);
    zText = (TextView)findViewById(R.id.msg5);
    dist = (TextView)findViewById(R.id.msg6);


    T=new Timer();
    T.schedule(task,0,fixedRate(new TimerTask() {
    @Override
    public void run() {
                runOnUiThread(new Runnable()
                {
    @Override
    public void run()
                {
    count_timer = count_timer+1;
                }
                });
            }
        }, 1000, 1000);
    usable_count_timer = count_timer%60;
    }


    /**
     *
     * @param sensor
     * @param accuracy
     */
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }


    /**
     *
     * @param event
     */
    @Override
    public void onSensorChanged(SensorEvent event) {

    float d = 0;


    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER){
                accelerometer(event);
            }

    if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD)
    mGeomagnetic = event.values;
    if (event.sensor.getType() == Sensor.TYPE_PROXIMITY){
    mProximity = event.values;
    dist.setText("Distance: " + event.values[0]);
                d = event.values[0];
            }


    if (mGravity != null && mGeomagnetic != null) {
    float R[] = new float[9];
    float I[] = new float[9];
    boolean success = SensorManager.getRotationMatrix(R, I, mGravity, mGeomagnetic);
    if (success) {
    float orientation[] = new float[3];
                SensorManager.getOrientation(R, orientation);
    azimut = orientation[0]; // orientation contains: azimut, pitch and roll
    pitch = orientation[1];
    roll = orientation[2];
            }
        }

    xText.setText("X: " + azimut);
    yText.setText("Y: " + pitch);
    zText.setText("Z: " + roll);

    if((pitch < 45.00 || pitch > -45.00) && (azimut < 2.00 || azimut > -2.00) && (roll < 2.00 || roll >-2.00)){
    i=1;
    // if condition for frigueny
    if(i%2 < 2){
    message = "Attentive";
    count2 = count2+1;
    V0 = 1;
                } else {
    message = "Not Attentive";
    V0 = 0;
    count1 = count1+1;
    count2 = count2+1;
                }
            } else {
    message = "Not Attentive";
    V0 = 0;
    count1 = count1+1;
    count2 = count2+1;
            }
        } else {
    message = "Not Attentive";
    V0 = 0;
    count1 = count1+1;
    count2 = count2+1;
        }

    mess.setText("Status: " + message);

    if(usable_count_timer == 0){
                update();
            }


        }

    public void accelerometer(SensorEvent event){
        // assign accelerometer values
    float x = event.values[0];
    float y = event.values[1];
    float z = event.values[2];

    if (!init){
    x1 = x;
    y1 = y;
    z1 = z;
    init = true;
            } else {
    float diffX = Math.abs(x1 - x);
    float diffY = Math.abs(y1 - y);
    float diffZ = Math.abs(z1 - z);

    if (diffX < ERROR) {
                diffX = (float) 0.0;
            }
    if (diffY < ERROR) {
                diffY = (float) 0.0;
            }
    if (diffZ < ERROR) {
                diffZ = (float) 0.0;
            }

    x1=x;
    y1=y;
    z1=z;

    if(diffX > diffY || diffX > diffZ || diffY > diffZ || diffY < diffZ || diffX < diffZ || diffX < diffY){
    count = count+1;
            }

    hz = (count/count_timer);

    r.setText("Frigueny: "+count);

        }
    }

    /**
     *
     */

    private void update(){
    final String name = "demo1";
    final int desg = V0;


    /**
     *
     */
    class UpdateEmployee extends AsyncTask<Void,Void,String>{
        ProgressDialog loading;

    /**
     *
     */
    @Override
    protected void onPreExecute() {
    super.onPreExecute();
        }

    /**
     *
     * @param s
     */
    @Override
    protected void onPostExecute(String s) {
    super.onPostExecute(s);
        }


    /**
     *
     * @param params
     * @return
     */
    @Override
    protected String doInBackground(Void... params) {
                HashMap<String,String> hashMap = new HashMap<>();
                hashMap.put("name", name);
                hashMap.put("value",Integer.toString(desg));

                Request rh = new Request();

                String s = rh.sendPostRequest(Constants.URL_update,hashMap);

    return s;
            }
        }


        UpdateEmployee ue = new UpdateEmployee();
        ue.execute();
    }

    /**
     *
     * @param menu
     * @return
     */

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

            getMenuInflater().inflate(R.menu.nav_menu, menu);
    return super.onCreateOptionsMenu(menu);
    }

    /**
     *
     * @param item
     * @return
     */

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {


    if(item.getItemId() == R.id.settings){
                startActivity(new Intent(getApplicationContext(), RegistorActivity.class));
            }
    if(item.getItemId() == R.id.help){
                startActivity(new Intent(getApplicationContext(), HelpActivity.class));
            }
    if(item.getItemId() == R.id.logout){
                finish();
            }
    return super.onOptionsItemSelected(item);
        }
    }
```

LINEARLY INDEPENDENT PATHS

1) 1->2->3->4.....->46->47->48->90->91->92->93->94->95->96->97
2) 1->2->3->4.....->46->47->48->90->91->92->93->94->98->99->100->101->...->117
3) 1->2->3->4.....->46->48->49->50->51->52->53->...->62
4) 1->2->3->4.....->46->48->49->50->51->52->53->54->55->56->57->...79->80->87->88->.....127
5) 1->2->3->4.....->46->48->49->50->51->52->63->64->65->81->82->83->...86->87->88->.....127

3.3 Notification Activity

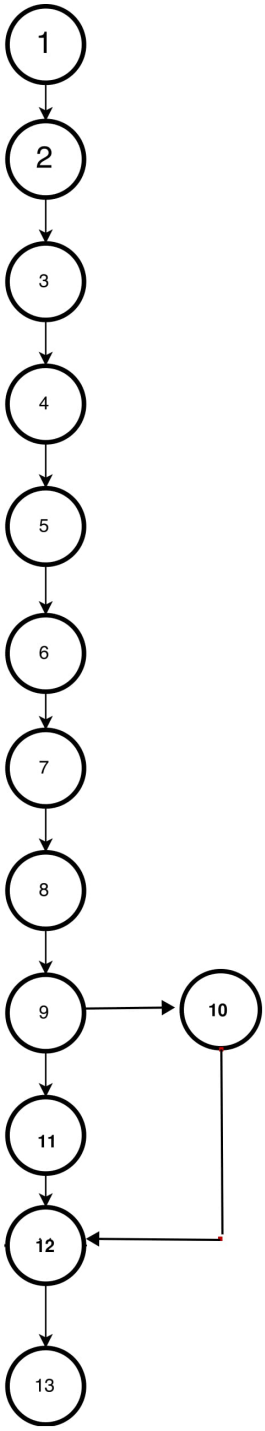
3.3.1 CFG

The CFG is drawn on the next page.

3.3.2 Test cases and Test Results

The number of Linearly Independent Paths in the CFG = 2

Path No.	Test Case	Test Result
1	When shaking frequency <= 2 per minute and proper orientation along x, y and z axis and distance is 25cm	There is no notification sent to the instructor.
2	When shaking frequency > 2 per minute or improper orientation along x, y and z axis or distance is not 25cm	The system notifies the instructor about the student.



NOTIFICATION ACTIVITY

```
1.  public void notification(){
2.  T=new Timer();
3.  T.scheduleAtFixedRate(new TimerTask() {
    @Override
4.  public void run() {
5.              runOnUiThread(new Runnable()
                {
    @Override
6.  public void run()
                    {
7.  count_timer = count_timer+1;
                    }
                });
            }
        }, 1000, 1000);
8.  usable_count_timer = count_timer%60;

9.  if(usable_count_timer == 0){
10.         onBupdate();
        }

11. new ProgressTask(InstructorActivity.this).execute();
    }

12. public void onBupdate(){
13. v.vibrate(50);
    }
```

LINEARLY INDEPENDENT PATHS:

1) 1->2->3->4->5->6->7->8->9->10->12->13

2) 1->2->3->4->5->6->7->8->9->11->12->13