5/6/2017

CODE TESTING

BISHWENDRA CHOUDHARY (150101017)
AYUSH SONI (150101014)
BHOLA SANKAR (150101016)

Table of Contents

Black Box Testing :

1) Equivalence classes in Proximity sensor –

The range of proximity sensor is different for different mobile.

Eg. - In gionee P6 mobile range of proximity sensor is '0' to '1' but In LeEco 2 mobile its value is '0' to '5'.

Proximity sensor reading value at any instance = 'X' .

Equivalent classes proximity for LeEco 2 mobile are as following-

1)Valid classes**-**

- 0<= X <= 5

2)Invalid classes **–**

- X > 5
- X < 0

Equivalent classes proximity for Gionee P6 mobile are as following-

1)Valid classes-

- 0<= X <= 1

2)Invalid classes **–**

- X > 1
- X < 0

So, We have 3 equivalence classes for Proximty sensor In which one is for valid and two classes for invalid. So, we have check for one value in each class.

And we have two boundary value i.e. 0 and 5 .

Test cases inputs (if reference mobile is leEco 2) are

{ -12 , 0 , 2 , 5 , 12 }

Test cases result are –

- For input X = -12 , output is 'Student not detected' Toast shown on screen
- For input X = 0 , output is 'Student not detected' Toast shown on screen
- For input X = 2 , output is 'Student detected' Toast shown on screen
- For input X = 5 , output is 'Student detected' Toast shown on screen
- For input X =12 , output is 'Student not detected' Toast shown on screen

2) Equivalence classes In Camera module –

Count vaiables is number of times student moved his face .
Request_camera variable will set if count > 2.

So, there is only 2 equivalent classes.

Valid equivalent class is –

Count > 2

Invalid equivalent class is –

Count < = 2

Test cases inputs are - **{-1,2,8}**

Test cases result are –

- For input count = -1 ,
  output is “Student is engaged “ Toast shown on screen
- For input count = 2
  Output is “Student is engaged ” Toast shown on screen
- For input count = 8
  Output is “Student is not engaged ” Toast shown on screen

3) Equivalence classes In Registration module –

Input is username and password and Mobile number

X = set of special character ={@,#<$,.....}

Equivalence classes for username ---:

Valid equivalent classes are –

- 4 <= Length of username <= 10
- username does not contain special character

Invalid equivalent classes are –

- Length of username > 10
- Length of username < 4

Test cases inputs are -

{cat , gudu , bishwendra , iraladilip , reddyhareesha , gudu@114 }

Test cases results are - :

- For input username ‘cat’
  Output is ‘string not accept as username’
- For input username ‘gudu’
  Output is ‘string accept as a username’

- For input username ‘bishwendra’
  Output is ‘string accept as a username’
- For input username ‘iraladilip’
  Output is ‘string accept as a username’
- For input username ‘reddyhareesh’
  Output is ‘string not accept as a username’
- For input username ‘gudu@114’
  Output is ‘string not accept as a username’

Equivalence classes for password **-:**

Valid equivalence classes are –

- 7 <= Length of password < = 15
- Password contain atleast one special character

Invalid equivalence classes are –

- Length of password > 15
- Length of password < 7

Test cases inputs are –

{qwer , qwertyu@ , qwertyuio@ , bishwendra786qw@,
bishwendra786qwerty , bishwendra }

Test cases results are -:

- For input password ‘qwer’
  Output is ‘string not accept as password’
- For input password ‘qwertyu@’
  Output is ‘string accept as password’
- For input password ‘qwertyuiop@’
  Output is ‘string accept as password’
- For input password ‘bishwendra786qw@’
  Output is ‘string accept as password’

- For input password ‘bishwendra786qwerty’
  Output is ‘string not accept as password’
- For input password ‘bishwendra’
  Output is ‘string not accept as password’

Equivalence classes for mobile number - :

Valid equivalence classes are –

- Mobile number contains only number from {0,1,2,3,4,5,6,7,8,9}

Invalid equivalence classes are –

- Mobile number contains alphabets
- Mobile number contains special character

Test cases inputs are –

{ 9854998812 , 9854@12345 , 985499881A }

Test cases Results are –

- For input 9854998812
  Output is ‘input accept as phone number’
- For input 985499881A
  Output is ‘input not accept as phone number’
- For input 9854@12345
  Output is ‘input not accept as phone number’

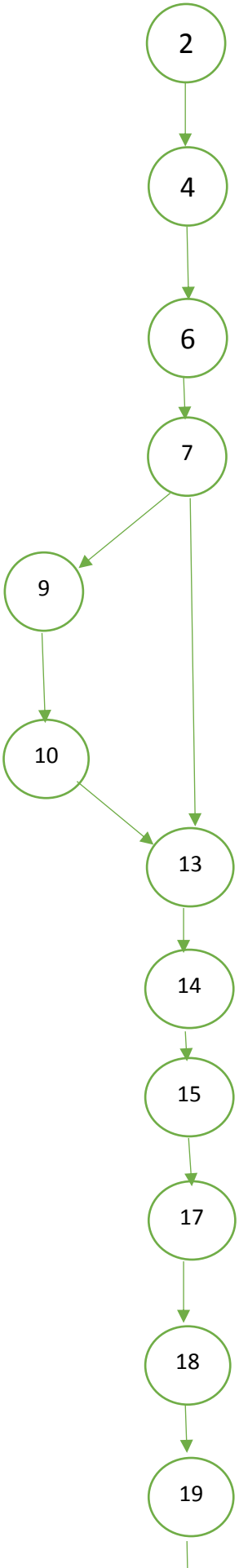
White Box Testing

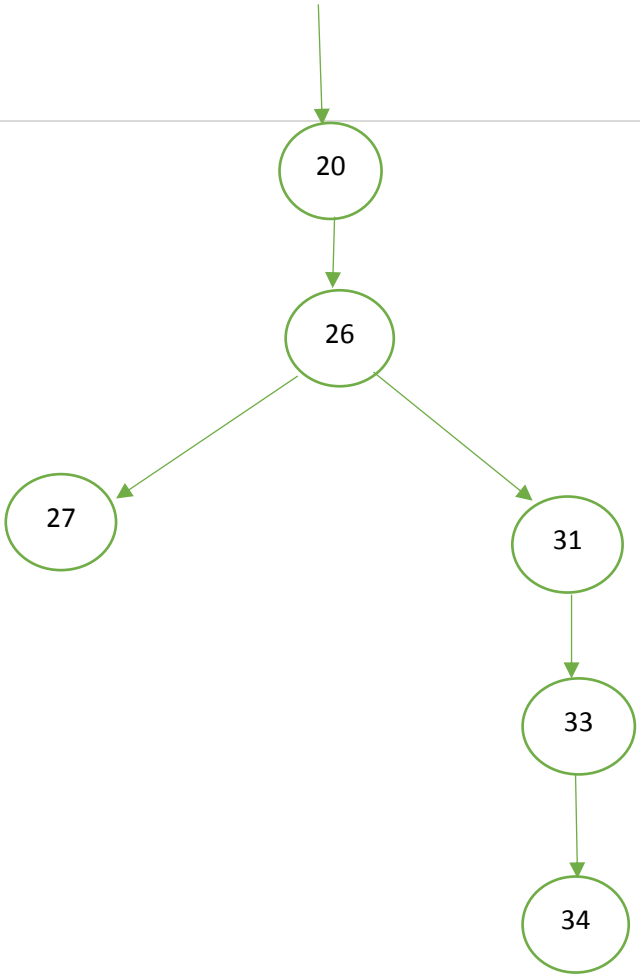
1) Module name – StudentJoin
Critical function - onCreate

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setting layout of activity_student_join
    setContentView(R.layout.activity_student_join);
    //click contains Button with id "button"
    Button click = (Button) findViewById(R.id.button);
    Result_photo = (ImageView) findViewById(R.id.imageView);
    //this if statement disables click if device dont have camera
    if (!hasCamera()) {
        click.setEnabled(false);
    }
    //prints username
    String username = getIntent().getStringExtra("EXTRA_MESSAGE");
    TextView tv = (TextView)findViewById(R.id.Joined);
    tv.setText(username);
    //printing proximity sensor tabs
    ProximitySensor = (TextView) findViewById(R.id.proximitySensor);
    ProximityMax = (TextView) findViewById(R.id.proximityMax);
    ProximityReading = (TextView) findViewById(R.id.proximityReading);
    mySensorManager = (SensorManager) getSystemService(
            Context.SENSOR_SERVICE);
    //getting proximity sensor in myProximitySensor
    myProximitySensor = mySensorManager.getDefaultSensor(
            Sensor.TYPE_PROXIMITY);
    //prints error if there is no Proximity sensor
    if (myProximitySensor == null) {
        ProximitySensor.setText("No Proximity Sensor!");
    }
    else {
        //prints technical name of proximity sensor
        ProximitySensor.setText(myProximitySensor.getName());
        //setting maxvalue of sensor to proximityMax
        ProximityMax.setText("Maximum Range: " + String.valueOf(myProximitySensor.getMaximumRange()));
        mySensorManager.registerListener(proximitySensorEventListener,
                myProximitySensor,
                SensorManager.SENSOR_DELAY_NORMAL);
    }
}
```

Path Covarage for onCreate method ----

Note - Each number inside a node represent a line number of a statement in a program


2
4
6
7
9
10
13
14
15
17
18
19



2) Module Name - StudentPage
Critical Function Name - Show Teacher list

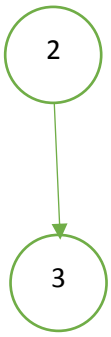
```
public void ShowTeacherList(List<String>Teacher){
        List_view = (ListView)findViewById(R.id.TeacherList) ;
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,R.layout.activity_listview,Teacher);
        List_view.setAdapter(adapter);
        Toast temp = Toast.makeText(StudentPage.this," hello" , Toast.LENGTH_SHORT);
        temp.show();
    }
```

Path Coverage Diagram for Show Teacher List method---

3) Module Name – Teacher Page
Critical fumation -- onClick_logout

```
1  public void onClick_logout(View view) {
2          Intent intend = new Intent(this,MainActivity.class);
3          startActivity(intend);
4      }
```

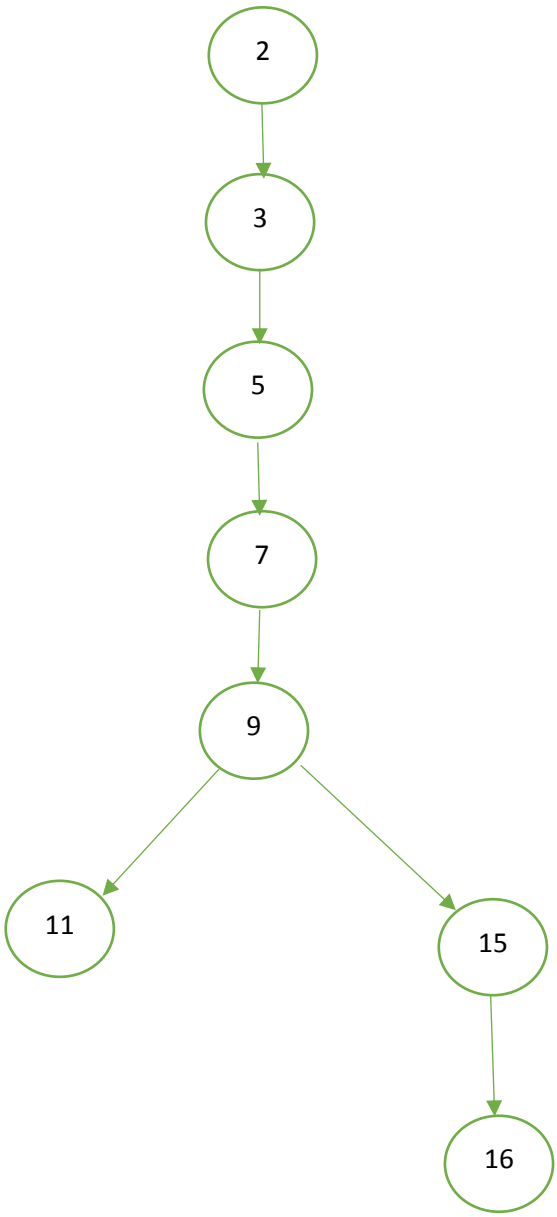
Path Coverage Diagram for onClick_logout method---



4) Module Name - Sign_Up
Critical function Name - AddData()

```
public void AddData() {
        Register.setOnClickListener(
                new View.OnClickListener() {

                    public void onClick(View v) {
                        //boolean isInserted true if all fields have items inputted
                        Boolean isInserted = myDb.insertData(EditUser.getText().toString(), EditPass.getText().toString(), EditMobile.getText().toString(),Spinner

                        if (isInserted==Boolean.TRUE)
                            //Toast the update when account made
                            Toast.makeText(Sign_Up.this, "input is inserted " +
                                    "OnClickListener : " +

                                    "\nSpinner  : "+ String.valueOf(Spinner1.getSelectedItem()), Toast.LENGTH_LONG).show();
                        else
                            Toast.makeText(Sign_Up.this, "input is not inserted", Toast.LENGTH_LONG).show();
                    }
                }
        );
    }
```

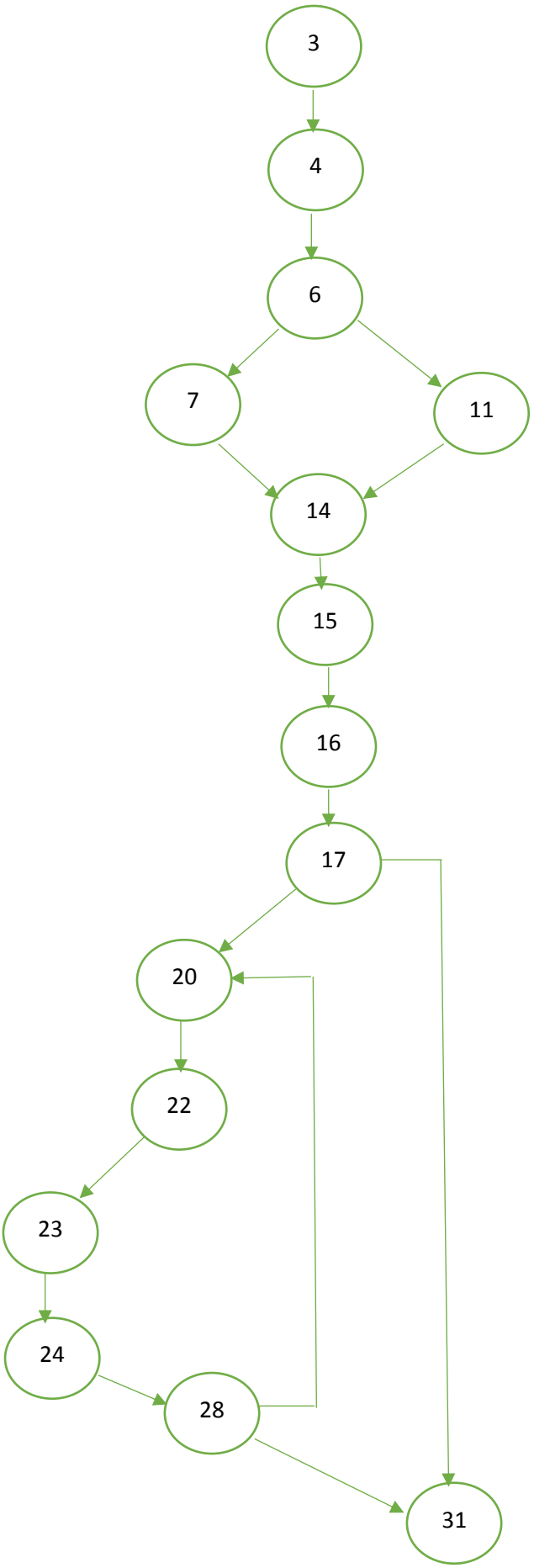
Path Coverage Diagram for onClick_logout method---



5) Module Name - DatabaseHelper
Critical function Name - searchPass()

```
public String searchPass(String uname,String type)
    {
        SQLiteDatabase db = this.getReadableDatabase();
        String query;
        //for searching value in student keys
        if(type=="Student") {
            query = "select user, pass from " + TABLE_STUDENT;
        }
        //for searching value in teacher keys
        else {
            query = "select user, pass from " + TABLE_TEACHER;
        }

        Cursor cursor = db.rawQuery(query , null);
        String a,b;
        b= "not found";
        if (cursor.moveToFirst())
        {   //move b in whole table till a not equal to uname
            do{
                a = cursor.getString(0);

                if(a.equals(uname))
                {
                    b = cursor.getString(1);
                    break;
                }
            }
            while(cursor.moveToNext());
        }

        return b;

    }
```

Path Coverage Diagram for searchPass method---


3
4
6
7
11
14
15
16
17
20
22
23
24
28
31

6) Module Name - CoustomOnSelectListener
Critical function Name - onItemSelected

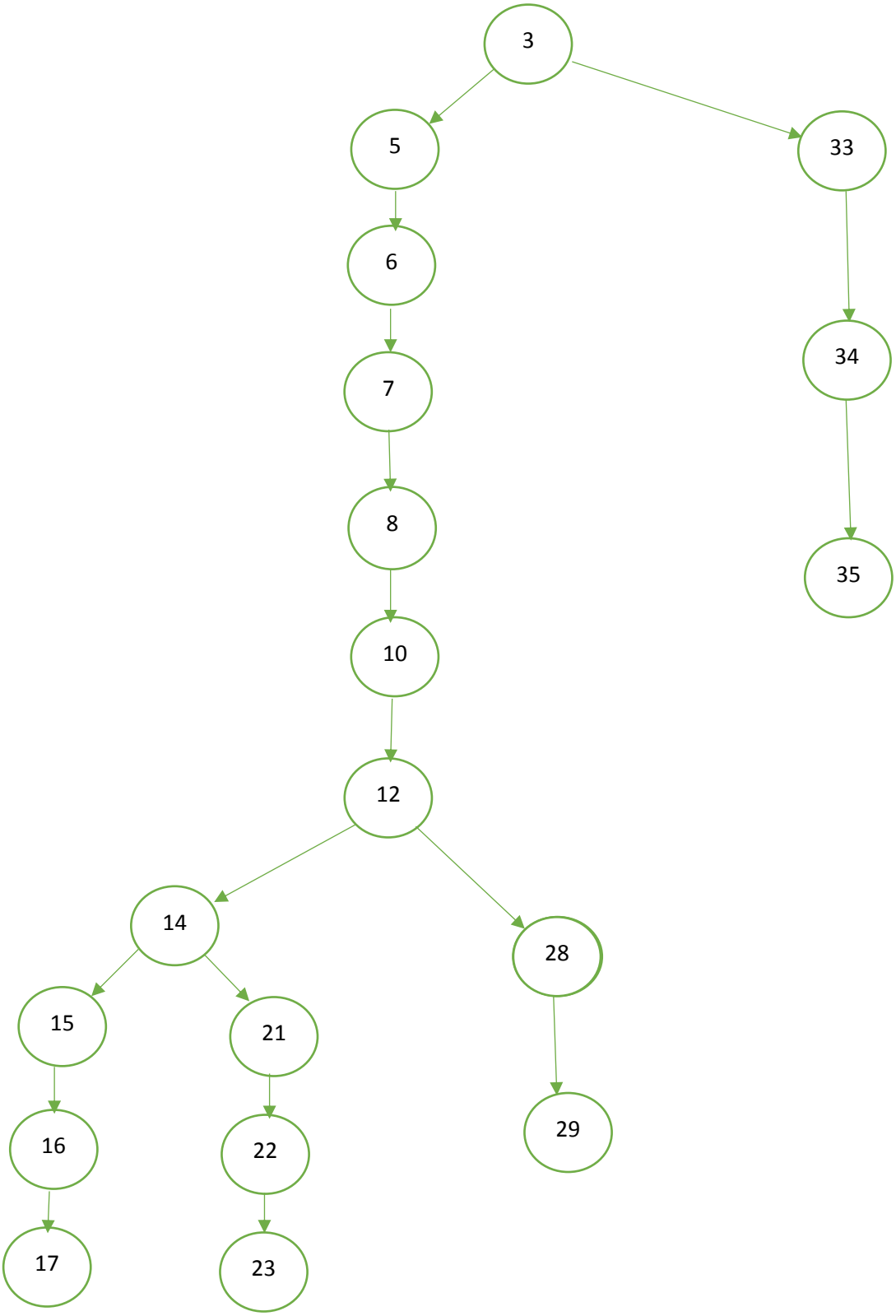
```
1  public void onItemSelected(AdapterView<?> parent,View view,int pos,long id){
2          Toast.makeText(parent.getContext(), "OnItemSelectedListener : "+ parent.getItemAtPosition(pos).toString(), Toast.LENGTH_SHORT).show();
3      }
```

Path Coverage Diagram for onItemSelected method---

7) Module Name - MainActivity
Critical function Name - onClick

```
public void onClick (View view){
    //checking if button clicked is login
    if(view.getId()==R.id.login){
        //taking username , password inputs
        EditText a =(EditText)findViewById(R.id.username);
        String str = a.getText().toString();
        EditText b =(EditText)findViewById(R.id.password);
        String pass = b.getText().toString();
        //getting password saved in database
        String password = db.searchPass(str,Spinner1.getSelectedItem().toString());
        //check if password matched
        if(pass.equals(password))
        {   //move to StudentPage activity when student is selected from spinner button
            if(Spinner1.getSelectedItem().toString()=="Student") {
                Intent i = new Intent(MainActivity.this, StudentPage.class);
                i.putExtra("Username", str);
                startActivity(i);
            }
            //move to TeacherPage activity when teacher is selected from spinner button
            else{
                Intent i = new Intent(MainActivity.this, TeacherPage.class);
                i.putExtra("Username", str);
                startActivity(i);
            }
        }
        //else pop error with Toast
        else{
            Toast temp = Toast.makeText(MainActivity.this,"Username and Password don't match" , Toast.LENGTH_SHORT);
            temp.show();
        }
    }
    //move to sign_up activity if clicked on SignUp
    if(view.getId()==R.id.register) {
        Intent intend = new Intent(this, Sign_Up.class);
        startActivity(intend);
    }
}
```

Path Coverage Diagram for onClick method---


3
5
33
6
34
7
8
35
10
12
14
28
15
21
29
16
22
17
23