Software Plan

1: Scope

1. Function:

The software we are creating is a mobile phone voting software and its trend analyzer. The software should allow the user to send a text message from their smart phone to a provided number for a smart-phone that is running the voting software. And “trend analyzer” should analyze and predict the trend of voting for a specific subject in a year. The voters would have to provide an ID number in the text message of their desired vote. The voting software will save the phone number and ID to prevent duplicate voting. The voting software should be able to display the number of votes in a decreasing number, so winners get displayed in order. And trend analyzer should provide a histogram to let administrator visualize the prediction of the voting trend.

2.Performance:

The performance of the app should be fast and mostly dependent on the user’s connection and ability to properly send a text message. This is because the software should be rather simple and shouldn’t be doing anything overly complex.

3.limitation:

The software also requires users to have a phone and a phone number.

2: Tasks

The tasks for our group will begin with learning the Android software development tools. This and figuring out how to read incoming text messages will likely be the more complicated part of the software. There isn’t a need for a complex user interface since all we need to do is display data since user input is done via text message. The data we need to store are the voted ID’s and the users phone numbers’. We also need to provide a method to access the stored data for the admin running the voting software.

3: Resources

**Hardware**: We will need a computer and an android smartphone for developing this software. The computer to code and create the software and the android smartphone for running the software. **Software**: We will need to use the SIS server, Android Studio, and Java for developing the software. **People**: To run the software, we need an admin running the voting system on the smartphone and other users who are sending text messages to the smartphone running the voting software.

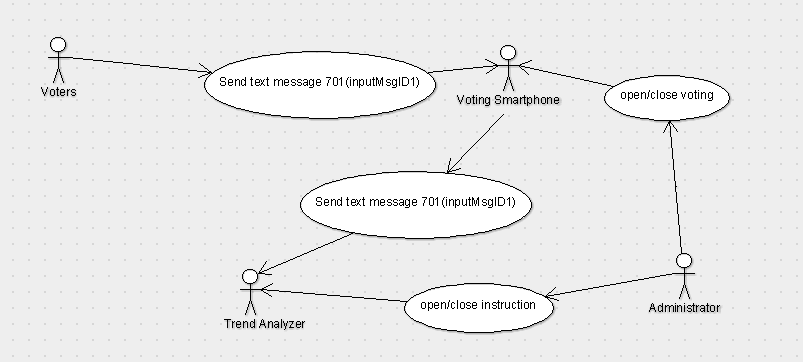
Diagrams

Name: Ehsan Elahi , Yuchao Wang

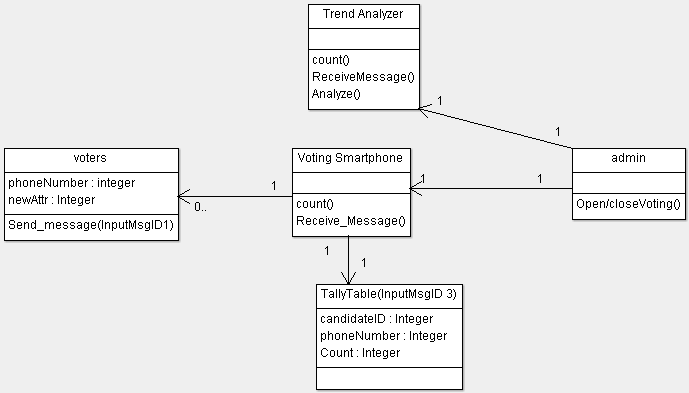
Notice: **msg** 701 (CastVote with attributes VoterPhoneNo, CandidateID)

711 (AcknowledgeVote with attribute Status)

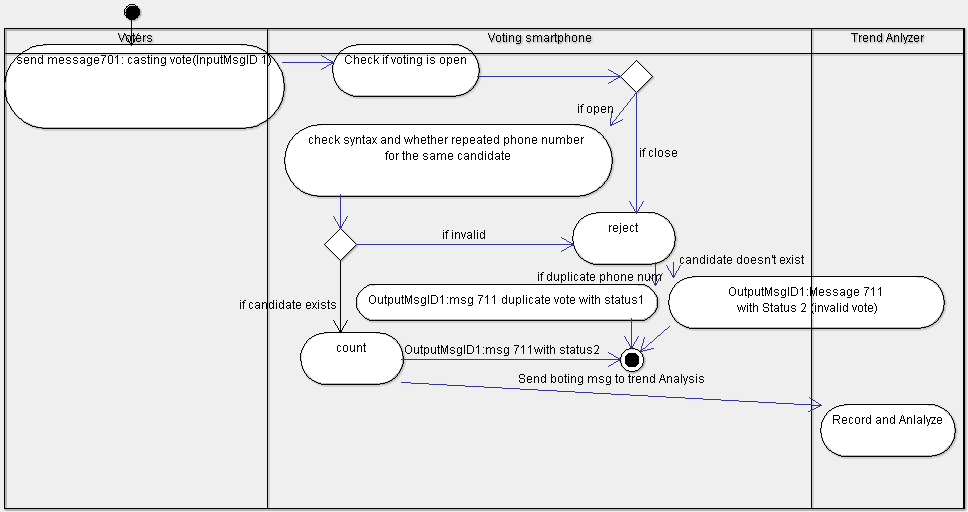
User case:



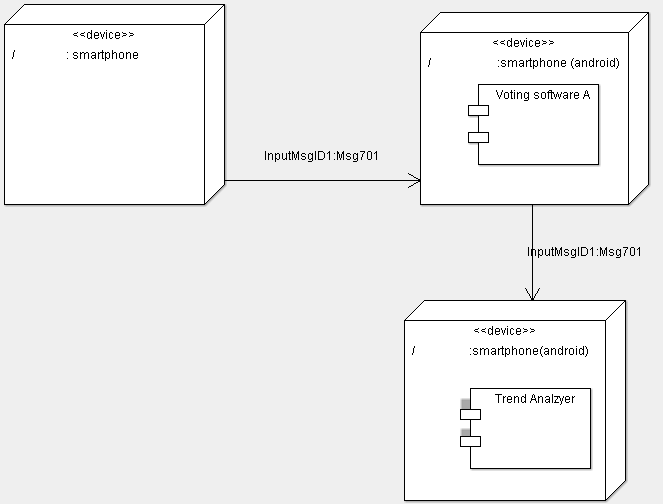
Class diagram:



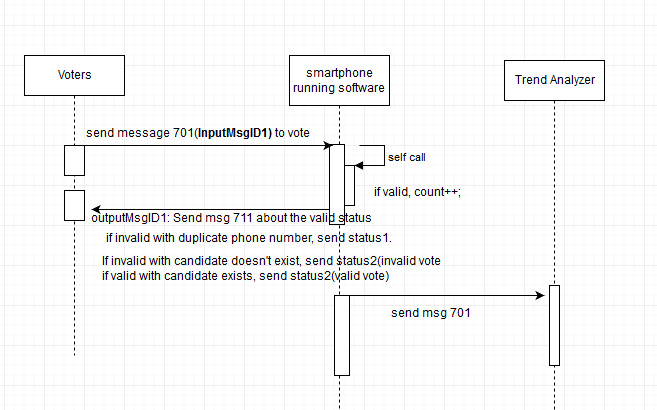
Activity diagram:



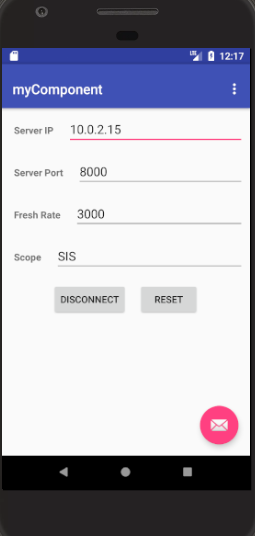
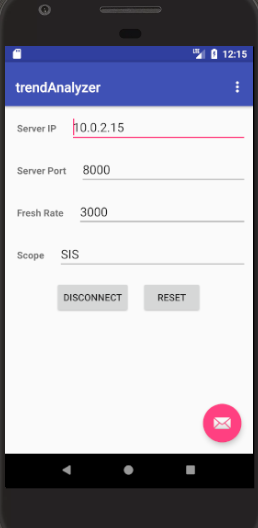
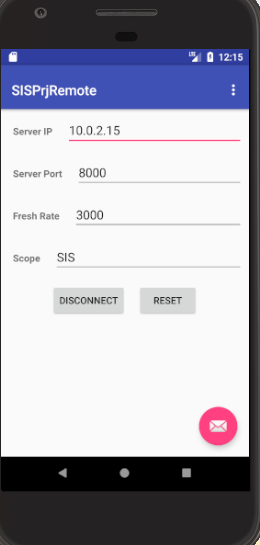
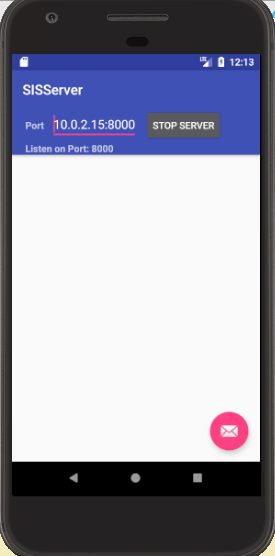
Deployment diagram:



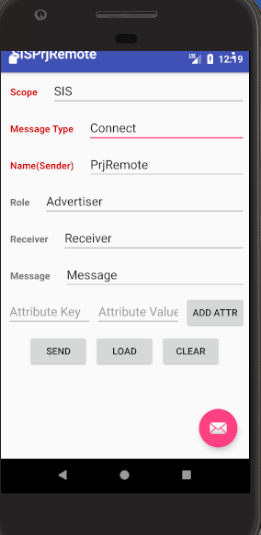
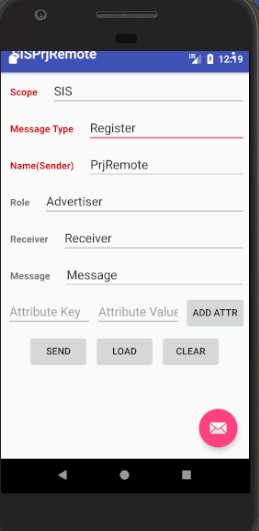
Sequence diagram:



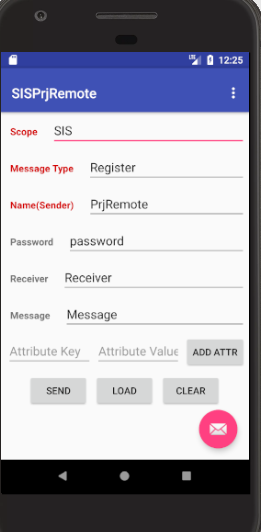
App using Explanation



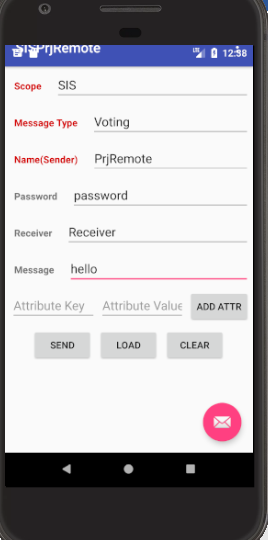
First each component needs to connect to the server. “Prjremote” is the admin that controls the opening of voting pools.



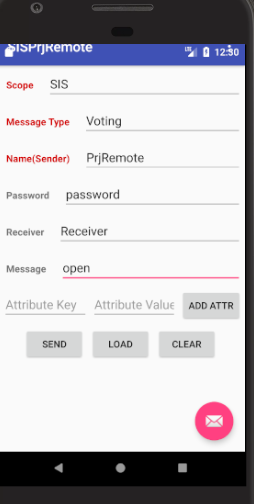
And then each component registers and connects to the Server so that it can communicate with the server.



Enter the password to open/close the voting pool.

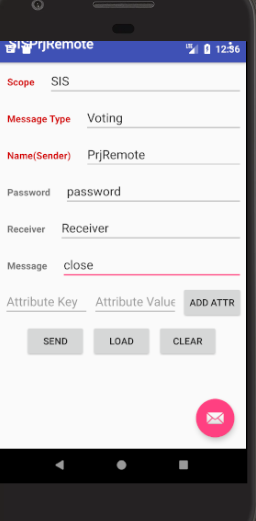


Enter the subject in “ Message “, “hello” for example, before opening the pool.

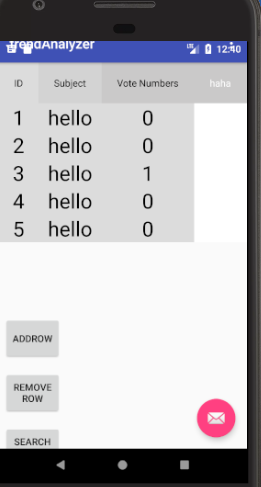
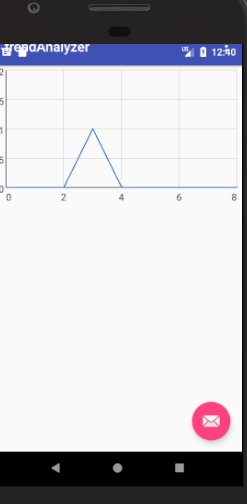


Enter “open” in message to control “open” voting pool.

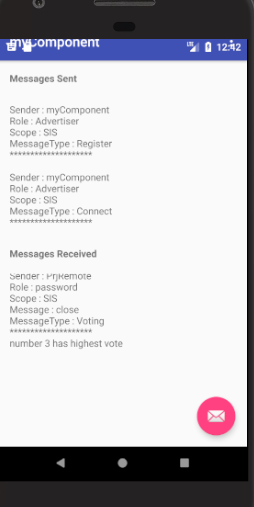
And user can begin voting with a single number, “2” for example, representing the ID of candidates.



admin closes the pool.



Trend analyzer will give you a table that records the history of voting in different subjects, and give a histogram that will predict the voting in a specific subject.



“MyComponent” will show us who wins the vote.

**else if**(num==4 ){  
 **if**(**rootView4**==**null**){  
 **rootView4** = inflater.inflate(R.layout.table, container, **false**);  
 Button addButton = (Button) **rootView4**.findViewById(R.id.button01);  
 **final** Button removeButton = (Button) **rootView4**.findViewById(R.id.button02);  
 addButton.setOnClickListener(**new** View.OnClickListener() {  
 **public void** onClick(View v) {  
 **final** TableLayout tl=(TableLayout)**rootView4**.findViewById(R.id.TableLayout01);  
 **final** TableRow tr1 = **new** TableRow(v.getContext());  
 tr1.setClickable(**true**);  
 TableRow.LayoutParams textLayoutParams = **new** TableRow.LayoutParams(TableRow.LayoutParams.***MATCH\_PARENT***, TableRow.LayoutParams.***WRAP\_CONTENT***);  
 tr1.setLayoutParams(textLayoutParams);  
 TextView textview = **new** TextView(v.getContext());  
 textview.setText(**""**);  
 textview.setTextColor(Color.***BLACK***);  
 textview.setTextSize(30);  
 textview.setClickable(**true**);  
 textview.setGravity(View.***TEXT\_ALIGNMENT\_GRAVITY***);  
 textview.setBackgroundColor(Color.*parseColor*(**"#dcdcdc"**));  
 tr1.addView(textview,textLayoutParams);  
  
 TextView textview2 = **new** TextView(v.getContext());  
 textview2.setText(**""**);  
 textview2.setTextColor(Color.***BLACK***);  
 textview2.setTextSize(30);  
 textview2.setClickable(**true**);  
 textview2.setGravity(View.***TEXT\_ALIGNMENT\_GRAVITY***);  
 textview2.setBackgroundColor(Color.*parseColor*(**"#dcdcdc"**));  
 tr1.addView(textview2,textLayoutParams);  
  
 TextView textview3 = **new** TextView(v.getContext());  
 textview3.setText(**""**);  
 textview3.setTextColor(Color.***BLACK***);  
 textview3.setTextSize(30);  
 textview3.setClickable(**true**);  
 textview3.setGravity(View.***TEXT\_ALIGNMENT\_GRAVITY***);  
 textview3.setBackgroundColor(Color.*parseColor*(**"#dcdcdc"**));  
 tr1.addView(textview3,textLayoutParams);  
  
 tl.addView(tr1, **new** TableLayout.LayoutParams(GridLayout.LayoutParams.***MATCH\_PARENT***, GridLayout.LayoutParams.***WRAP\_CONTENT***));  
 tr1.setOnClickListener(**new** View.OnClickListener() {  
 **public void** onClick(View view) {  
 view.setBackgroundColor(Color.***RED***);  
 removeButton.setOnClickListener(**new** View.OnClickListener() {  
 **public void** onClick(View v) {  
 TableLayout tl=(TableLayout)**rootView4**.findViewById(R.id.TableLayout01);  
  
 tl.removeView(tr1);  
 }  
 });  
  
 }  
 });  
  
  
 }  
 });  
 **final** TableLayout tl=(TableLayout)**rootView4**.findViewById(R.id.TableLayout01);  
  
 **final** Handler handler=**new** Handler();  
 handler.post(**new** Runnable(){  
 @Override  
 **public void** run() {  
 *// upadte textView here* **int** count=tl.getChildCount();  
 **if**(count>2) {  
 **for**(**int** b=2+*skip*;b<count;b++)  
 {  
 **if**(count<*MAX*) {  
 TableRow row = (TableRow) tl.getChildAt(b);  
 *//Log.d("", "===============================================================:2 " );  
 //Toast.makeText(MainActivity.this,"i="+ row.getChildCount(),Toast.LENGTH\_LONG).show();* TextView txtView0 = (TextView) row.getChildAt(0);  
 txtView0.setText(Integer.*toString*(b - 1));  
 TextView txtView1 = (TextView) row.getChildAt(1);  
 txtView1.setText(*Subject*);  
 TextView txtView2 = (TextView) row.getChildAt(2);  
 txtView2.setText(Integer.*toString*(*list*[b - 1]));  
 }  
 }  
 }  
 *//Toast.makeText(MainActivity.this,"i="+i,Toast.LENGTH\_LONG).show();* handler.postDelayed(**this**,500); *// set time here to refresh textView* }  
 });  
  
 Button searchButton= (Button) **rootView4**.findViewById(R.id.button03);  
 searchButton.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **final** EditText searchText=(EditText) **rootView4**.findViewById(R.id.EditText01);  
 **int** count=tl.getChildCount();  
 *//Toast.makeText(MainActivity.this,"row number is "+ count,Toast.LENGTH\_LONG).show();* **for**(**int** i=2;i<count;i++)  
 {  
 **if**(count<*MAX*) {  
 TableRow tb = (TableRow) tl.getChildAt(i);  
  
 **for** (**int** j = 0; j < 3; j++) {  
 String keyword = searchText.getText().toString();  
 TextView txtV = (TextView) tb.getChildAt(j);  
 String txt = txtV.getText().toString();  
 **if** (txt.equals(keyword)) {  
 txtV.setTextColor(Color.*parseColor*(**"#00bfff"**));  
 } **else** {  
 txtV.setTextColor(Color.***BLACK***);  
 }  
 }  
 }  
 }  
 }  
 });  
 }  
 **return rootView4**;  
 }  
 **else if**(num==5 ){  
 **if**(**rootView5**==**null**){  
 **rootView5** = inflater.inflate(R.layout.histogram, container, **false**);  
 **final** GraphView graph = (GraphView) **rootView5**.findViewById(R.id.graph);  
 **final int** datapoint\_num=10;  
 **int**[] pool=**new int**[datapoint\_num];  
 **final** DataPoint[] dp=**new** DataPoint[datapoint\_num] ;  
 **for**(**int** i=0;i<datapoint\_num;i++)  
 {  
 dp[i]=**new** DataPoint(i,*list*[i]);  
  
 }  
  
 *//new DataPoint(1,5);  
 /\*  
 new DataPoint(0, 1),  
 new DataPoint(1, 5),  
 new DataPoint(2, 3),  
 new DataPoint(3, 2),  
 new DataPoint(4, 6)\*/* **final** LineGraphSeries<DataPoint> series = **new** LineGraphSeries<DataPoint>(dp);  
 graph.addSeries(series);  
 **final** Handler handler=**new** Handler();  
 handler.post(**new** Runnable(){  
 @Override  
 **public void** run() {  
 *// upadte textView here* **for**(**int** i=0;i<datapoint\_num;i++)  
 {  
 dp[i]=**new** DataPoint(i,*list*[i]);  
 }  
 LineGraphSeries<DataPoint> series = **new** LineGraphSeries<DataPoint>(dp);  
 graph.addSeries(series);  
 *//Toast.makeText(MainActivity.this,"i="+i,Toast.LENGTH\_LONG).show();* handler.postDelayed(**this**,500); *// set time here to refresh textView* }  
 });  
  
 }  
 **return rootView5**;  
 }  
 **return rootView1**;  
}

**public void** run() {  
 **super**.run();  
 *//Keep listening if there is any incoming messages* **while**(!**killThread**){  
 **try** {  
 *//Build a new socket* **socket** = **new** Socket(*serverAddress*, *serverPort*);  
 **socket**.setKeepAlive(**true**);  
 *msgDecoder* = **new** MsgDecoder(**socket**.getInputStream());  
 *msgEncoder* = **new** MsgEncoder(**socket**.getOutputStream());  
  
 *//System.out.println("lalala");  
 //Tell the activity that a new socket has been built.* Message message = **callback**.obtainMessage(MainActivity.***CONNECTED***);  
 **callback**.sendMessage(message);  
 **killThread** = **false**;  
 **while**(**true**){  
 Log.*d*(**"debug"**,**"hahaha"**);  
 *//Check if there is an incoming message.* KeyValueList kvList = *msgDecoder*.getMsg();  
  
  
 **if** (kvList.size() > 1) {  
  
 String messageType=kvList.getValue(**"MessageType"**);  
 String message2=kvList.getValue(**"Message"**);  
 **if**(messageType.equals(**"Voting"**)) {  
 Log.*e*(MainActivity.***TAG***, **"Received raw: <"** + kvList.encodedString() + **">"**);  
 *//Tell the activity that a new message has been received.* **if**(message2.equals(**"open"**)) {  
 Message msg = **callback**.obtainMessage(MainActivity.***MESSAGE\_RECEIVED***);  
 Message msg3 = **callback**.obtainMessage(MainActivity.***OPEN\_POOL***);  
 msg.**obj** = kvList.toString();  
 **callback**.sendMessage(msg);  
 **callback**.sendMessage(msg3);  
 }  
 **else if**(message2.equals(**"close"**)) {  
 Message msg = **callback**.obtainMessage(MainActivity.***MESSAGE\_RECEIVED***);  
 Message msg2 = **callback**.obtainMessage(MainActivity.***CLOSE\_POOL***);  
 msg.**obj** = kvList.toString();  
 **callback**.sendMessage(msg);  
 **callback**.sendMessage(msg2);  
 }  
 **else** {  
 String subject=message2;  
 Message msg3 = **callback**.obtainMessage(6,subject);  
 **callback**.sendMessage(msg3);  
 }  
 }  
 */\* else{  
 msgProcess(kvList);  
 Message msg = callback.obtainMessage(MainActivity.MESSAGE\_RECEIVED);  
 msg.obj = kvList.toString();  
 callback.sendMessage(msg);  
  
 }\*/* }  
  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 Message message = **callback**.obtainMessage(MainActivity.***DISCONNECTED***);  
 **callback**.sendMessage(message);  
 }  
 **try** {  
 Thread.*sleep*(100);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }

SmsReceiver.*bindListener*(**new** SmsListener() {  
 @Override  
 **public void** messageReceived(String messageText,String sender) {  
 **if**(*open*) {  
 **if**(!*arr*.contains(sender)) {  
 *arr*.add(sender);  
 Log.*i*(**"Text"**, messageText + **"\t "** + *arr*.get(0));  
 Toast.*makeText*(MainActivity.**this**, **"Message: "** + messageText + **"sender"** + sender, Toast.***LENGTH\_LONG***).show();  
 **if** (*isInteger*(messageText)) {  
 Toast.*makeText*(MainActivity.**this**, **"Message: is integer"**, Toast.***LENGTH\_LONG***).show();  
 **int** result = Integer.*parseInt*(messageText);  
 **if** (result > 0 && result < *MAX* + 1) {  
 *list*[result] = *list*[result] + 1;  
 }  
  
 }  
 }  
 }  
 }  
});

**public void** run() {  
 **super**.run();  
 *//Keep listening if there is any incoming messages* **while**(!**killThread**){  
 **try** {  
 *//Build a new socket* **socket** = **new** Socket(*serverAddress*, *serverPort*);  
 **socket**.setKeepAlive(**true**);  
 *msgDecoder* = **new** MsgDecoder(**socket**.getInputStream());  
 *msgEncoder* = **new** MsgEncoder(**socket**.getOutputStream());  
  
 *//System.out.println("lalala");  
 //Tell the activity that a new socket has been built.* Message message = **callback**.obtainMessage(MainActivity.***CONNECTED***);  
 **callback**.sendMessage(message);  
 **killThread** = **false**;  
 **while**(**true**){  
 Log.*d*(**"debug"**,**"hahaha"**);  
 *//Check if there is an incoming message.* KeyValueList kvList = *msgDecoder*.getMsg();  
  
  
 **if** (kvList.size() > 1) {  
  
 String messageType=kvList.getValue(**"MessageType"**);  
 String message2=kvList.getValue(**"Message"**);  
 **if**(messageType.equals(**"Voting"**)) {  
 Log.*e*(MainActivity.***TAG***, **"Received raw: <"** + kvList.encodedString() + **">"**);  
 *//Tell the activity that a new message has been received.* **if**(message2.equals(**"open"**)) {  
 Message msg = **callback**.obtainMessage(MainActivity.***MESSAGE\_RECEIVED***);  
 Message msg3 = **callback**.obtainMessage(MainActivity.***OPEN\_POOL***);  
 msg.**obj** = kvList.toString();  
 **callback**.sendMessage(msg);  
 **callback**.sendMessage(msg3);  
 }  
 **else if**(message2.equals(**"close"**)) {  
 Message msg = **callback**.obtainMessage(MainActivity.***MESSAGE\_RECEIVED***);  
 Message msg2 = **callback**.obtainMessage(MainActivity.***CLOSE\_POOL***);  
 msg.**obj** = kvList.toString();  
 **callback**.sendMessage(msg);  
 **callback**.sendMessage(msg2);  
 }  
 }  
 */\* else{  
 msgProcess(kvList);  
 Message msg = callback.obtainMessage(MainActivity.MESSAGE\_RECEIVED);  
 msg.obj = kvList.toString();  
 callback.sendMessage(msg);  
  
 }\*/* }  
  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 Message message = **callback**.obtainMessage(MainActivity.***DISCONNECTED***);  
 **callback**.sendMessage(message);  
 }  
 **try** {  
 Thread.*sleep*(100);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }