

```
1 package com.example.gui_design;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import android.support.v4.content.ContextCompat;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.MenuItem;
10
11 public class ContactsActivity extends AppCompatActivity {
12     private static Toolbar toolbar;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.contacts_layout);
18         Application.setActivity(this);
19         Application.setContext(this);
20
21         toolbar = this.findViewById(R.id.toolbar);
22
23         setSupportActionBar(toolbar);
24
25         // add back arrow to toolbar
26         if (getSupportActionBar() != null) {
27             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
28             getSupportActionBar().setDisplayShowHomeEnabled(true);
29
30             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
31             upArrow.setColorFilter(getResources().getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
32             getSupportActionBar().setHomeAsUpIndicator(upArrow);
33         }
34
35         TextToSpeechHandler.initializeTextToSpeech();
36         try {
37             Thread.sleep(250);
38         } catch (Throwable th) { Application.toastMessage(th.toString());}
39
40         TextToSpeechHandler.addToSpeakingQueue("holá");
41     }
42
43     @Override
44     public boolean onOptionsItemSelected(MenuItem item) {
45         if (item.getItemId() == android.R.id.home) // Press Back Icon
46         {
47             finish();
48         }
49
50         return super.onOptionsItemSelected(item);
51     }
52 }
53 }
```

```

1 package com.example.gui_design.bluetoothActivities;
2
3 import android.bluetooth.BluetoothAdapter;
4 import android.bluetooth.BluetoothDevice;
5 import android.bluetooth.BluetoothServerSocket;
6 import android.bluetooth.BluetoothSocket;
7 import android.content.BroadcastReceiver;
8 import android.content.Context;
9 import android.content.Intent;
10
11 import com.example.gui_design.Application;
12
13 import java.io.InputStream;
14 import java.io.OutputStream;
15 import java.nio.charset.StandardCharsets;
16 import java.util.ArrayList;
17 import java.util.List;
18 import java.util.Set;
19 import java.util.UUID;
20 import java.util.concurrent.Executor;
21 import java.util.concurrent.Executors;
22 import java.util.concurrent.ScheduledExecutorService;
23 import java.util.concurrent.ScheduledFuture;
24
25
26 public class BluetoothConnection extends BluetoothActivity {
27
28     static ScheduledExecutorService bluetoothThreadPool= Executors.newScheduledThreadPool(1);
29     static ScheduledFuture<?> bluetoothComThread;
30     public static OutputStream outputStream;
31     public static InputStream inputStream;
32     public static BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();
33     public static BluetoothDevice targetDevice;
34     public static BluetoothDevice[] bluetoothDevicesArray;
35     public static List<BluetoothDevice> bluetoothDevicesList;
36     public static BluetoothSocket btSocket;
37     public static BluetoothServerSocket btServerSocket;
38     public static UUID btUUID = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
39     public static String btString = "HC-05";
40     public static Boolean bluetoothConnected = false;
41
42     private static Executor bluetoothConnectorExecutor = Executors.newSingleThreadExecutor();
43     private static Boolean isListAvailableBluetoothDevicesRunning = false;
44
45     public static void listAvailableBluetoothDevices(){
46         if ((!isListAvailableBluetoothDevicesRunning)) {
47             try {
48                 BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.WAITING);
49                 isListAvailableBluetoothDevicesRunning = true;
50                 bluetoothConnectorExecutor.execute(new Runnable() {
51                     @Override
52                     public void run() {
53                         if (btAdapter != null) {
54                             if (btAdapter.isEnabled()) {
55                                 Set<BluetoothDevice> bondedDevices = btAdapter.getBondedDevices();
56
57                                 bluetoothDevicesList = new ArrayList<>();
58                                 bluetoothDevicesList.addAll(bondedDevices);
59
60                                 if (bondedDevices.size() > 0) {
61                                     bluetoothDevicesArray = bondedDevices.toArray(new BluetoothDevice[bondedDevices.size()]);
62                                 }
63                                 updateBluetoothListGUI();
64                             } else {
65                                 BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
66                                     BLUETOOTH_DISABLED);
67                             }
68                         }
69                     }
70                 });
71             } catch (Throwable th)
72             {
73                 Application.toastMessage(th.toString()); th.printStackTrace();
74             }
75         }
76     }

```

```

76     }
77
78     public static Boolean isBluetoothAdapterReady(){
79         return ((btAdapter != null) && (btAdapter.isEnabled()));
80     }
81
82     public static void updateBluetoothListGUI(){
83         if (BluetoothActivity.mAdapter!=null) {
84             BluetoothActivity.deviceList.clear();
85             for (int x = 0; x < bluetoothDevicesArray.length; x++) {
86                 BluetoothActivity.deviceList.add(new AvailableDevice(bluetoothDevicesArray[x].getName
87 (), AvailableDevice.DeviceState.DEFAULT));
88             }
89
89             if (bluetoothConnected) {
90                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.SYSTEM_CONNECTED);
91                 deviceList.get(blueoothDevicesList.indexOf(targetDevice)).setCurrentState(
92                     AvailableDevice.DeviceState.CONNECTED);
93             }
94             else {
95                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.SYSTEM_DISCONNECTED);
96             }
97
97             BluetoothActivity.updateDevicesList();
98         }
99     }
100
101     private static Boolean attemptingBluetoothConnection = false;
102
103
104     public static void connectBluetoothDeviceSync(int deviceIndex){
105         if ((!isListAvailableBluetoothDevicesRunning)&&(!attemptingBluetoothConnection) &&
106             isBluetoothAdapterReady()) {
107             try {
108                 attemptingBluetoothConnection = true;
109                 if (bluetoothConnected) closeAllConnections();
110                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.WAITING);
111                 targetDevice = bluetoothDevicesArray[deviceIndex];
112                 btSocket = targetDevice.createRfcommSocketToServiceRecord(btUUID);
113                 btSocket.connect();
114                 if (btSocket.isConnected()) {
115                     inputStream = btSocket.getInputStream();
116                     outputStream = btSocket.getOutputStream();
117                     bluetoothConnected = true;
118                     BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.SYSTEM_CONNECTED)
119                     ;
120                     BluetoothActivity.updateDevicesList();
121                 } else {
122                     bluetoothConnected = false;
123                     BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
124                         SYSTEM_DISCONNECTED);
125                 }
126                 attemptingBluetoothConnection = false;
127             } catch (Throwable th) {
128                 bluetoothConnected = false;
129                 Application.toastMessage(th.toString());
130                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.SYSTEM_DISCONNECTED);
131                 attemptingBluetoothConnection = false;
132             }
133         }
134     }
135     private static int attemptingConnectionDeviceIndex;
136
137     public static void connectBluetoothDeviceAsync(int deviceIndex){
138         attemptingConnectionDeviceIndex = deviceIndex;
139         if ((!isListAvailableBluetoothDevicesRunning)&&(!attemptingBluetoothConnection) &&
140             isBluetoothAdapterReady()) {
141             bluetoothConnectorExecutor.execute(new Runnable() {
142                 @Override
143                 public void run() {
144                     connectBluetoothDeviceSync(attemptingConnectionDeviceIndex);
145                 }
146             });
147         }
148     }

```

```

147
148
149
150
151
152     private static Executor autoConnectExecutor = Executors.newSingleThreadExecutor();
153     private static Boolean terminateAutoConnect = false;
154     private static Boolean isAutoConnectRunning = false;
155
156     public static void initBTAutoConnectService(){
157         if (isAutoConnectRunning() == false) {
158             autoConnectExecutor.execute(new Runnable() {
159                 @Override
160                 public void run() {
161                     isAutoConnectRunning = true;
162                     try { Thread.sleep(5000); } catch (Throwable th) { th.printStackTrace(); }
163                     while (!terminateAutoConnect && isBluetoothAdapterReady()) {
164                         listAvailableBluetoothDevices();
165                         while (isListAvailableBluetoothDevicesRunning) {}
166                         for (int x = 0; x < bluetoothDevicesArray.length; x++) {
167                             // MAC ADDRESS HC-05 : 98:D3:91:FD:38:05
168                             // MAC ADDRESS XT1068 : 5C:51:88:A2:38:97
169                             if (bluetoothDevicesArray[x].getAddress().compareToIgnoreCase("98:D3:91:
170 FD:38:05") == 0) {
171                                 if (!terminateAutoConnect&&!bluetoothConnected)
172                                     connectBluetoothDeviceSync(x);
173                                 }
174                             }
175                         isAutoConnectRunning = false;
176                         terminateAutoConnect = false;
177                     }
178                 });
179             }else{
180                 terminateAutoConnect = false;
181             }
182         }
183
184     public static void terminateBTAutoConnect(){
185         terminateAutoConnect = true;
186     }
187
188     public static Boolean isAutoConnectRunning(){
189         return isAutoConnectRunning;
190     }
191
192
193
194
195     public static void connectionLost(){
196
197     }
198
199     public static void closeAllConnections(){
200         try {
201             bluetoothConnected = false;
202             if (bluetoothComThread != null) bluetoothComThread.cancel(true);
203             if (btSocket != null) btSocket.close();
204             if (outputStream != null) outputStream.close();
205             if (inputStream != null) inputStream.close();
206
207             btSocket = null;
208             outputStream = null;
209             inputStream = null;
210         } catch (Throwable th){
211             th.printStackTrace();
212         }
213     }
214
215     public enum CommandType {TURN_RELAY_ON, TURN_RELAY_OFF, TURN_EVERYTHING_ON, TURN_EVERYTHING_OFF,
216 CALL_NURSE}
217
218     public static boolean composeAndSend(CommandType cmdType, int parameter){
219         if (bluetoothConnected) {
220             try {
221                 String commandOutput = null;

```

File - C:\ Library256\Feria 2019\Interfaz grafica\app\src\main\java\com\example\gui\design\bluetoothActivities\BluetoothConne

```
221     switch (cmdType) {
222         case TURN_RELAY_ON:
223             commandOutput = String.format("/TURN_RELAY_ON[%s$]", Integer.toString(
224                 parameter));
225             break;
226         case TURN_RELAY_OFF:
227             commandOutput = String.format("/TURN_RELAY_OFF[%s$]", Integer.toString(
228                 parameter));
229             break;
230         case TURN_EVERYTHING_ON:
231             commandOutput = "/TURN_EVERYTHING_ON$";
232             break;
233         case TURN_EVERYTHING_OFF:
234             commandOutput = "/TURN_EVERYTHING_OFF$";
235             break;
236         case CALL_NURSE:
237             commandOutput = "/CALL_NURSE$";
238             break;
239     }
240
241     outputStream.write(commandOutput.getBytes(StandardCharsets.US_ASCII));
242     Application.toastMessage(commandOutput);
243     return true;
244 } catch (Throwable th) {
245     Application.toastMessage(th.toString());
246     return false;
247 }
248
249 public static BroadcastReceiver bluetoothAdapterStatusBR = new BroadcastReceiver() {
250     @Override
251     public void onReceive(Context context, Intent intent) {
252         final String action = intent.getAction();
253         if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)) {
254             final int stateCode = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
255                 BluetoothAdapter.ERROR);
256
257             switch (stateCode) {
258                 case BluetoothAdapter.STATE_OFF:
259                     //Bluetooth adapter turned off handler
260                     BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
261                         BLUETOOTH_DISABLED);
262                     break;
263                 case BluetoothAdapter.STATE_TURNING_OFF:
264                     //Bluetooth adapter turning off handler
265                     terminateBTAutoConnect();
266                     BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
267                         BT_TURNING_OFF);
268                     break;
269                 case BluetoothAdapter.STATE_ON:
270                     BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
271                         SYSTEM_DISCONNECTED);
272                     updateDevicesList();
273                     initBTAutoConnectService();
274
275                     break;
276                 case BluetoothAdapter.STATE_TURNING_ON:
277                     //Bluetooth adapter turning on handler
278                     BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.BT_TURNING_ON
279             );
280             }
281         }
282
283         public static BroadcastReceiver bluetoothScanModeStatusBR = new BroadcastReceiver() {
284             @Override
285             public void onReceive(Context context, Intent intent) {
286                 final String action = intent.getAction();
287
288                 if (action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {
289
290                     int mode = intent.getIntExtra(BluetoothAdapter.EXTRA_SCAN_MODE, BluetoothAdapter.
291                         ERROR);
```

```
290         switch (mode) {
291             case BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE:
292                 break;
293             case BluetoothAdapter.SCAN_MODE_CONNECTABLE:
294                 break;
295             case BluetoothAdapter.SCAN_MODE_NONE:
296                 break;
297         }
298     }
299 }
300 };
301
302 public static BroadcastReceiver bluetoothDeviceStatusBR = new BroadcastReceiver() {
303     @Override
304     public void onReceive(Context context, Intent intent) {
305         String action = intent.getAction();
306
307         switch (action){
308             case BluetoothDevice.ACTION_ACL_CONNECTED:
309                 terminateBTAutoConnect();
310                 updateDevicesList();
311                 break;
312             case BluetoothDevice.ACTION_ACL_DISCONNECTED:
313                 closeAllConnections();
314                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
315 SYSTEM_DISCONNECTED);
316                 initBTAutoConnectService();
317                 updateBluetoothListGUI();
318                 //Application.toastMessage("Dispositivo desconectado!");
319                 break;
320             case BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED:
321                 break;
322             case BluetoothDevice.ACTION_BOND_STATE_CHANGED:
323                 break;
324         }
325     }
326 }
327 }
328 }
329 }
330 }
331 };
332 }
333 }
334 }
```

```

1 package com.example.gui_design.bluetoothActivities;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.graphics.Color;
6 import android.graphics.PorterDuff;
7 import android.graphics.drawable.Drawable;
8 import android.os.Bundle;
9 import android.os.Looper;
10 import android.support.v4.content.ContextCompat;
11 import android.support.v7.app.AppCompatActivity;
12 import android.support.v7.widget.DefaultItemAnimator;
13 import android.support.v7.widget.DividerItemDecoration;
14 import android.support.v7.widget.LinearLayoutManager;
15 import android.support.v7.widget.RecyclerView;
16 import android.support.v7.widget.Toolbar;
17 import android.view.MenuItem;
18 import android.view.View;
19 import android.widget.ImageView;
20 import android.widget.ProgressBar;
21 import android.widget.TextView;
22
23 import com.example.gui_design.Application;
24 import com.example.gui_design.R;
25 import com.example.gui_design.settingsActivities.BluetoothSettingsActivity;
26 import com.example.gui_design.settingsActivities.RecyclerTouchListener;
27 import com.github.ybq.android.spinkit.sprite.Sprite;
28 import com.github.ybq.android.spinkit.style.Circle;
29
30 import java.util.ArrayList;
31 import java.util.List;
32
33 public class BluetoothActivity extends AppCompatActivity {
34     private static Toolbar toolbar;
35
36
37     public static List<AvailableDevice> deviceList = new ArrayList<>();
38     private RecyclerView recyclerView;
39     public static DeviceListAdapter mAdapter;
40
41     public static ImageView imgView_bluetoothStatusIcon;
42     public static TextView textView_bluetoothState;
43     public static ProgressBar loading_progressBar;
44
45     public static Boolean guiInitialized = false;
46
47
48     @Override
49     protected void onCreate(Bundle savedInstanceState) {
50         super.onCreate(savedInstanceState);
51         setContentView(R.layout.bluetooth_layout);
52         Application.setActivity(this);
53         Application.setContext(this);
54
55         toolbar = this.findViewById(R.id.toolbar);
56         textView_bluetoothState = this.findViewById(R.id.textView_btStatus);
57         imgView_bluetoothStatusIcon = this.findViewById(R.id.imgView_btStatusIcon);
58         loading_progressBar = this.findViewById(R.id.loading_progressBar);
59
60         Sprite fadingCircle = new Circle();
61         fadingCircle.setColor(Color.WHITE);
62         loading_progressBar.setIndeterminateDrawable(fadingCircle);
63
64         setSupportActionBar(toolbar);
65
66         // add back arrow to toolbar
67         if (getSupportActionBar() != null){
68             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
69             getSupportActionBar().setDisplayShowHomeEnabled(true);
70
71             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
72                 abc_ic_ab_back_material);
73             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
74             getSupportActionBar().setHomeAsUpIndicator(upArrow);
75         }
76
77         recyclerView = (RecyclerView) findViewById(R.id.deviceListRecyclerView);

```

```

77         mAdapter = new DeviceListAdapter(deviceList);
78
79         recyclerView.setHasFixedSize(true);
80
81         // vertical RecyclerView
82         // keep movie_list_row.xml width to `match_parent`
83         RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());
84
85         // horizontal RecyclerView
86         // keep movie_list_row.xml width to `wrap_content`
87         // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext()
88         // (), LinearLayoutManager.HORIZONTAL, false);
89
90         recyclerView.setLayoutManager(mLayoutManager);
91
92         // adding inbuilt divider line
93         recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.VERTICAL));
94
95         // adding custom divider line with padding 16dp
96         //recyclerView.addItemDecoration(new MyDividerItemDecoration(this, LinearLayoutManager.
97         HORIZONTAL, 16));
98         recyclerView.setItemAnimator(new DefaultItemAnimator());
99
100        recyclerView.setAdapter(mAdapter);
101
102        // row click listener
103        recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext(),
104        recyclerView, new RecyclerTouchListener.ClickListener() {
105            @Override
106            public void onClick(View view, int position) {
107                AvailableDevice device = deviceList.get(position);
108
109                BluetoothConnection.connectBluetoothDeviceAsync(position);
110            }
111
112            @Override
113            public void onLongClick(View view, int position) {
114
115            }
116        }));
117
118        guiInitialized = true;
119
120    }
121
122    @Override
123    public boolean onOptionsItemSelected(MenuItem item) {
124        if (item.getItemId() == android.R.id.home) // Press Back Icon
125        {
126            finish();
127        }
128
129        return super.onOptionsItemSelected(item);
130    }
131
132    public static void updateDevicesList(){
133        if (guiInitialized) {
134            Application.getActivity().runOnUiThread(new Runnable() {
135                @Override
136                public void run() {
137                    BluetoothActivity.mAdapter.notifyDataSetChanged();
138                }
139            });
140        }
141    }
142
143    public void configBtn_onClick(View view){
144        Context currentContext = Application.getContext();
145        Intent i = new Intent(currentContext, BluetoothSettingsActivity.class);
146        currentContext.startActivity(i);
147    }
148
149

```

```

150     public enum bluetoothConnectionState { SYSTEM_DISCONNECTED, SYSTEM_CONNECTED, BLUETOOTH_DISABLED,
151         WAITING, BT_TURNING_OFF, BT_TURNING_ON, UNDEFINED}
152     public static bluetoothConnectionState currentBluetoothState = bluetoothConnectionState.UNDEFINED
153 ;
154
155     public static void setBluetoothStateGui(bluetoothConnectionState state){
156         if (guiInitialized) {
157             currentBluetoothState = state;
158
159             if (Looper.myLooper() == Looper.getMainLooper()) {
160                 switch (currentBluetoothState) {
161                     case SYSTEM_CONNECTED:
162                         textView_bluetoothState.setText(R.string.system_connected_message);
163                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_done_icon);
164                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
165                         loading_progressBar.setVisibility(View.GONE);
166                         break;
167                     case BLUETOOTH_DISABLED:
168                         textView_bluetoothState.setText(R.string.bluetooth_disabled_message);
169                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_bluetooth_disabled
170 );
171                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
172                         loading_progressBar.setVisibility(View.GONE);
173                         break;
174                     case BT_TURNING_ON:
175                         textView_bluetoothState.setText(R.string.bluetooth_initializing_adapater);
176                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
177                         loading_progressBar.setVisibility(View.VISIBLE);
178                         break;
179                     case BT_TURNING_OFF:
180                         textView_bluetoothState.setText(R.string.bluetooth_closing_adapter);
181                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
182                         loading_progressBar.setVisibility(View.VISIBLE);
183                         break;
184                     case SYSTEM_DISCONNECTED:
185                         textView_bluetoothState.setText(R.string.system_disconnected_message);
186                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_close_icon);
187                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
188                         loading_progressBar.setVisibility(View.GONE);
189                         break;
190                     case WAITING:
191                         textView_bluetoothState.setText(R.string.
192                         bluetooth_attempting_connection_message);
193                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
194                         loading_progressBar.setVisibility(View.VISIBLE);
195                         break;
196                 }
197             } else {
198                 Application.getActivity().runOnUiThread(new Runnable() {
199                     @Override
200                     public void run() {
201                         switch (currentBluetoothState) {
202                             case SYSTEM_CONNECTED:
203                                 textView_bluetoothState.setText(R.string.system_connected_message);
204                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_done_icon
205 );
206                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
207                                 loading_progressBar.setVisibility(View.GONE);
208                                 break;
209                             case BLUETOOTH_DISABLED:
210                                 textView_bluetoothState.setText(R.string.bluetooth_disabled_message);
211                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
212                                     ic_bluetooth_disabled);
213                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
214                                 loading_progressBar.setVisibility(View.GONE);
215                                 break;
216                             case BT_TURNING_ON:
217                                 textView_bluetoothState.setText(R.string.
218                                     bluetooth_initializing_adapater);
219                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
220                                 loading_progressBar.setVisibility(View.VISIBLE);
221                                 break;
222                             case BT_TURNING_OFF:
223                                 textView_bluetoothState.setText(R.string.bluetooth_closing_adapter);
224                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
225                                 loading_progressBar.setVisibility(View.GONE);
226                                 break;
227                         }
228                     }
229                 });
230             }
231         }
232     }

```

```
220             case SYSTEM_DISCONNECTED:
221                 textView_bluetoothState.setText(R.string.system_disconnected_message)
222             ;
223             imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_close_icon)
224             );
225             imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
226             loading_progressBar.setVisibility(View.GONE);
227             break;
228         case WAITING:
229             textView_bluetoothState.setText(R.string.
230 bluetooth_attempting_connection_message);
231             imgView_bluetoothStatusIcon.setVisibility(View.GONE);
232             loading_progressBar.setVisibility(View.VISIBLE);
233             break;
234         }
235     }
236 }
237
238
239 }
240
```

```

1 package com.example.gui_design;
2
3 import android.app.Activity;
4 import android.arch.lifecycle.LifecycleObserver;
5 import android.bluetooth.BluetoothAdapter;
6 import android.bluetooth.BluetoothDevice;
7 import android.content.Context;
8 import android.content.IntentFilter;
9
10 import com.example.gui_design.bluetoothActivities.BluetoothConnection;
11
12 import java.io.IOException;
13
14 public class Application extends android.app.Application implements LifecycleObserver {
15
16     @Override
17     public void onCreate() {
18         super.onCreate();
19
20         // ProcessLifecycleOwner.get().getLifecycle().addObserver(this);
21
22         IntentFilter bluetoothAdapterActionChangedFilter = new IntentFilter(BluetoothAdapter.
23             ACTION_STATE_CHANGED);
23         registerReceiver(BluetoothConnection.bluetoothAdapterStatusBR,
24             bluetoothAdapterActionChangedFilter);
24
25         IntentFilter bluetoothScanModeStatusChangedFilter = new IntentFilter();
26         bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
27         bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
28         bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
29         registerReceiver(BluetoothConnection.bluetoothScanModeStatusBR,
30             bluetoothScanModeStatusChangedFilter);
30
31         IntentFilter bluetoothDeviceStatusChangedFilter = new IntentFilter();
32         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
33         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
34         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED);
35         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
36         registerReceiver(BluetoothConnection.bluetoothDeviceStatusBR,
37             bluetoothDeviceStatusChangedFilter);
37
38         Thread internetDetector = new Thread(){
39             @Override
40             public void run() {
41                 super.run();
42                 while(true){
43                     try {
44                         internetConnectionStatus = isConnected();
45                         Thread.sleep(10000);
46                         if (internetConnectionStatus) {Application.toastMessage("Internet connection
47 detected");}
47                         else {Application.toastMessage("Proper internet connection has not been
48 detected");}
48                     } catch (Throwable th) { th.printStackTrace(); Application.toastMessage(th.
49                         toString());}
50                 }
51             };
52
53             internetDetector.start();
54
55         }
56
57         @Override
58         public void onTerminate() {
59             super.onTerminate();
60         }
61
62         private static Activity mActivity;
63         private static Context mContext;
64
65         public static Activity getActivity() { return mActivity; }
66         public static void setActivity(Activity mActivitySet) { mActivity = mActivitySet; }
67
68         public static Context getContext() { return mContext; }
69         public static void setContext(Context mContexttoSet) { mContext = mContexttoSet; }
70

```

```
71     public static void toastMessage(final String message) {
72         mActivity.runOnUiThread(new Runnable() {
73             @Override
74             public void run() {
75                 if (mActivity != null) {
76                     //Toast.makeText(mActivity, message, Toast.LENGTH_SHORT).show();
77                 }
78             }
79         });
80     }
81
82     private static boolean internetConnectionStatus;
83
84     public static boolean isSystemOnline() {
85         return internetConnectionStatus;
86     }
87
88     public boolean isConnected() throws InterruptedException, IOException {
89         final String command = "ping -c 1 google.com";
90         return Runtime.getRuntime().exec(command).waitFor() == 0;
91     }
92
93
94
95
96
97
98 }
99
```

```
1 package com.example.gui_design.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import android.support.v4.content.ContextCompat;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.gui_design.Application;
12 import com.example.gui_design.R;
13
14 public class InternetConnectivityActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.internet_connectivity_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34                 abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
36             getSupportActionBar().setHomeAsUpIndicator(upArrow);
37         }
38     }
39
40     @Override
41     public boolean onOptionsItemSelected(MenuItem item) {
42         if (item.getItemId() == android.R.id.home) // Press Back Icon
43         {
44             finish();
45         }
46
47         return super.onOptionsItemSelected(item);
48     }
49 }
```

```
1 package com.example.gui_design.bluetoothActivities;
2
3 import com.example.gui_design.R;
4
5 public class AvailableDevice {
6
7     public enum DeviceState { DEFAULT, DISCONNECTED, CONNECTED, LOADING}
8     private String deviceName;
9     private DeviceState currentState;
10    private int deviceStateIconID;
11
12    public AvailableDevice() {
13    }
14
15    public AvailableDevice(String deviceName, DeviceState defaultState) {
16        this.deviceName = deviceName;
17        this.currentState = defaultState;
18    }
19
20    public void setCurrentState(DeviceState currentState) {
21        this.currentState = currentState;
22    }
23
24    public DeviceState getCurrentState() {
25        return currentState;
26    }
27
28    public void setDeviceName(String deviceName) {
29        this.deviceName = deviceName;
30    }
31
32    public String getDeviceName() {
33        return deviceName;
34    }
35
36    public int getDeviceStateIconID() {
37        switch (currentState){
38            case DEFAULT:
39                return 0;
40            case CONNECTED:
41                return R.drawable.ic_done_icon;
42            case LOADING:
43                return R.drawable.loading_animation;
44            case DISCONNECTED:
45                return R.drawable.ic_close_icon;
46        }
47        return 0;
48    }
49
50    public int getTextAppearance(){
51        switch (currentState){
52            case DEFAULT:
53                return R.style.btDeviceListAppearance;
54            case CONNECTED:
55                return R.style.btDeviceListSelectedAppearance;
56            case LOADING:
57                return R.style.btDeviceListAppearance;
58            case DISCONNECTED:
59                return R.style.btDeviceListAppearance;
60        }
61        return R.style.btDeviceListAppearance;
62    }
63 }
64
```

```
1 package com.example.gui_design.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import android.support.v4.content.ContextCompat;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.gui_design.Application;
12 import com.example.gui_design.R;
13
14 public class CommandsSettingsActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.commands_settings_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34                 abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
36             getSupportActionBar().setHomeAsUpIndicator(upArrow);
37         }
38     }
39
40     @Override
41     public boolean onOptionsItemSelected(MenuItem item) {
42         if (item.getItemId() == android.R.id.home) // Press Back Icon
43         {
44             finish();
45         }
46
47         return super.onOptionsItemSelected(item);
48     }
49 }
```

```
1 package com.example.gui_design.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.support.v7.widget.RecyclerView;
6 import android.view.LayoutInflater;
7 import android.view.View;
8 import android.view.ViewGroup;
9 import android.widget.ImageView;
10 import android.widget.TextView;
11
12 import com.example.gui_design.Application;
13 import com.example.gui_design.R;
14
15 import java.util.List;
16
17 public class SettingsLayoutAdapter extends RecyclerView.Adapter<SettingsLayoutAdapter.viewHolder> {
18
19     private List<AvailableSetting> settingsList;
20
21     public class viewHolder extends RecyclerView.ViewHolder {
22         public TextView setting_title, setting_subtitle;
23         public ImageView setting_icon;
24
25         public viewHolder(View view) {
26             super(view);
27             setting_title = (TextView) view.findViewById(R.id.setting_title);
28             setting_subtitle = (TextView) view.findViewById(R.id.setting_subtitle);
29             setting_icon = (ImageView) view.findViewById(R.id.setting_icon);
30         }
31     }
32
33
34     public SettingsLayoutAdapter(List<AvailableSetting> settingsList) {
35         this.settingsList = settingsList;
36     }
37
38     @Override
39     public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
40         View itemView = LayoutInflater.from(parent.getContext())
41             .inflate(R.layout.settings_layout_style, parent, false);
42
43         return new viewHolder(itemView);
44     }
45
46     @Override
47     public void onBindViewHolder(viewHolder holder, int position) {
48         AvailableSetting setting = settingsList.get(position);
49         holder.setting_title.setText(setting.getTitle());
50         holder.setting_subtitle.setText(setting.getSubtitle());
51
52         Drawable icon = Application.getContext().getDrawable(setting.getIconResourceID());
53         icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary), PorterDuff.Mode.
54 SRC_ATOP);
55         holder.setting_icon.setImageDrawable(icon);
56     }
57
58     @Override
59     public int getItemCount() {
60         return settingsList.size();
61     }
62 }
```

```

1 package com.example.gui_design.settingsActivities;
2
3 import android.content.DialogInterface;
4 import android.graphics.PorterDuff;
5 import android.graphics.drawable.Drawable;
6 import android.os.Bundle;
7 import android.speech.tts.Voice;
8 import android.support.v4.content.ContextCompat;
9 import android.support.v7.app.AlertDialog;
10 import android.support.v7.app.AppCompatActivity;
11 import android.support.v7.widget.Toolbar;
12 import android.view.MenuItem;
13 import android.view.View;
14 import android.widget.Button;
15 import android.widget.EditText;
16 import android.widget.ProgressBar;
17
18 import com.example.gui_design.Application;
19 import com.example.gui_design.R;
20 import com.example.gui_design.TextToSpeechHandler;
21 import com.github.ybq.android.spinkit.sprite.Sprite;
22 import com.github.ybq.android.spinkit.style.FadingCircle;
23
24 import java.text.SimpleDateFormat;
25 import java.util.ArrayList;
26 import java.util.Calendar;
27 import java.util.Date;
28 import java.util.Locale;
29
30 public class VoiceRecognitionSettingsActivity extends AppCompatActivity {
31     private static Toolbar toolbar;
32
33     private Button speech_test_btn;
34     private Button set_voice_btn;
35     private EditText phrase_editText;
36     private static ProgressBar progressBar;
37
38     @Override
39     protected void onCreate(Bundle savedInstanceState) {
40         super.onCreate(savedInstanceState);
41         setContentView(R.layout.voice_recognition_settings_layout);
42         Application.setActivity(this);
43         Application.setContext(this);
44
45         toolbar = this.findViewById(R.id.toolbar);
46         speech_test_btn = this.findViewById(R.id.speech_test_btn);
47         set_voice_btn = this.findViewById(R.id.set_voice_btn);
48         phrase_editText = this.findViewById(R.id.phrase_editText);
49
50         setSupportActionBar(toolbar);
51
52         // add back arrow to toolbar
53         if (getSupportActionBar() != null){
54             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
55             getSupportActionBar().setDisplayShowHomeEnabled(true);
56
57             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
58             abc_ic_ab_back_material);
59             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
60             getSupportActionBar().setHomeAsUpIndicator(upArrow);
61         }
62
63         TextToSpeechHandler.initializeTextToSpeech();
64
65         progressBar = (ProgressBar) findViewById(R.id.progressBar);
66         Sprite fadingCircle = new FadingCircle();
67         fadingCircle.setColor(R.color.colorPrimary);
68         progressBar.setIndeterminateDrawable(fadingCircle);
69
70     }
71
72     @Override
73     public boolean onOptionsItemSelected(MenuItem item) {
74         if (item.getItemId() == android.R.id.home) // Press Back Icon
75         {
76

```

```

77         finish();
78     }
79
80     return super.onOptionsItemSelected(item);
81 }
82
83 public void speechTestOnClick(View view){
84     if (TextToSpeechHandler.isInitialized){
85         //TextToSpeechHandler.addToSpeakingQueue(phrase_editText.getText().toString());
86
87         Date currentTime = Calendar.getInstance().getTime();
88         SimpleDateFormat df = new SimpleDateFormat("d-MMM-yyy", new Locale("spa"));
89         String todayAsString = df.format(currentTime);
90         SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new Locale("spa"));
91         String onlyTimeString = onlyTimeFormar.format(currentTime);
92
93         TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + " y son las" +
94         onlyTimeString);
95     }else{
96         Application.toastMessage("Not initialized");
97     }
98 }
99
100 public static ArrayList<String> voiceName= new ArrayList<>();
101 public static Voice[] voiceArray;
102
103 public void setVoiceOnClick(View view){
104     AlertDialog.Builder builder = new AlertDialog.Builder(Application.getContext());
105
106     voiceArray = TextToSpeechHandler.getVoicesList().toArray(new Voice[TextToSpeechHandler.
107     getVoicesList().size()]);
108
109     for(Voice voice : voiceArray){
110         voiceName.add(voice.getName());
111     }
112
113     builder.setTitle("Seleccionar voz");
114     builder.setItems(voiceName.toArray(new String[voiceName.size()]), new DialogInterface.
115     OnClickListener() {
116         @Override
117         public void onClick(DialogInterface dialog, int which) {
118             TextToSpeechHandler.setVoice(voiceArray[which]);
119         }
120     });
121     builder.show();
122 }
123
124 @Override
125 protected void onPause() {
126     if (TextToSpeechHandler.isInitialized){
127         //TextToSpeechHandler.shutdownTextSpeech();
128     }
129
130     super.onPause();
131 }

```

```
1 package com.example.gui_design.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import android.support.v4.content.ContextCompat;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.gui_design.Application;
12 import com.example.gui_design.R;
13
14 public class ContactsCommunicationActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.contacts_communication_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34                 abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
36             getSupportActionBar().setHomeAsUpIndicator(upArrow);
37         }
38     }
39
40     @Override
41     public boolean onOptionsItemSelected(MenuItem item) {
42         if (item.getItemId() == android.R.id.home) // Press Back Icon
43         {
44             finish();
45         }
46
47         return super.onOptionsItemSelected(item);
48     }
49 }
```

```
1 package com.example.gui_design.settingsActivities;
2
3 public class AvailableSetting {
4     private String title, subtitle;
5     private int iconResourceID;
6     private Class targetActivity;
7
8     public AvailableSetting() {
9     }
10
11    public AvailableSetting(String title, String subtitle, int iconResourceID, Class targetActivity) {
12        this.title = title;
13        this.subtitle = subtitle;
14        this.iconResourceID = iconResourceID;
15        this.targetActivity = targetActivity;
16    }
17
18    public String getTitle() {
19        return title;
20    }
21
22    public void setTitle(String name) {
23        this.title = name;
24    }
25
26    public String getSubtitle() {
27        return subtitle;
28    }
29
30    public void setSubtitle(String subtitle) {
31        this.subtitle = subtitle;
32    }
33
34    public int getIconResourceID() {
35        return iconResourceID;
36    }
37
38    public void setIconResourceID(int iconResourceID) {
39        this.iconResourceID = iconResourceID;
40    }
41
42    public Class getTargetActivity() { return targetActivity; }
43
44    public void setTargetActivity(Class targetActivity) { this.targetActivity = targetActivity; }
45 }
46
```

```

1 package com.example.gui_design.settingsActivities;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.graphics.PorterDuff;
7 import android.graphics.drawable.Drawable;
8 import android.os.Bundle;
9 import android.support.v4.content.ContextCompat;
10 import android.support.v7.app.AppCompatActivity;
11 import android.support.v7.widget.DefaultItemAnimator;
12 import android.support.v7.widget.DividerItemDecoration;
13 import android.support.v7.widget.LinearLayoutManager;
14 import android.support.v7.widget.RecyclerView;
15 import android.support.v7.widget.Toolbar;
16 import android.view.MenuItem;
17 import android.view.View;
18
19 import com.example.gui_design.Application;
20 import com.example.gui_design.R;
21
22 import java.util.ArrayList;
23 import java.util.List;
24
25 public class settings_layout extends AppCompatActivity {
26
27     private static Toolbar toolbar;
28
29     public static List<AvailableSetting> settingsList = new ArrayList<>();
30     private RecyclerView recyclerView;
31     public static SettingsLayoutAdapter mAdapter;
32
33
34     @Override
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.settings_layout);
38         Application.setActivity(this);
39         Application.setContext(this);
40
41         toolbar = findViewById(R.id.toolbar);
42
43         setSupportActionBar(toolbar);
44
45         // add back arrow to toolbar
46         if (getSupportActionBar() != null){
47             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
48             getSupportActionBar().setDisplayShowHomeEnabled(true);
49
50
51             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
52                 abc_ic_ab_back_material);
53             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
54             getSupportActionBar().setHomeAsUpIndicator(upArrow);
55         }
56
57         recyclerView = (RecyclerView) findViewById(R.id.settingsRecyclerView);
58
59         mAdapter = new SettingsLayoutAdapter(settingsList);
60
61         recyclerView.setHasFixedSize(true);
62
63         // vertical RecyclerView
64         // keep movie_list_row.xml width to 'match_parent'
65         LinearLayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());
66
67         // horizontal RecyclerView
68         // keep movie_list_row.xml width to 'wrap_content'
69         // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext()
70         // , LinearLayoutManager.HORIZONTAL, false);
71
72         // adding inbuilt divider line
73         recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.VERTICAL));
74
75         // adding custom divider line with padding 16dp

```

```

76     //recyclerView.addItemDecoration(new MyDividerItemDecoration(this, LinearLayoutManager.
77     HORIZONTAL, 16));
78     recyclerView.setItemAnimator(new DefaultItemAnimator());
79     recyclerView.setAdapter(mAdapter);
80
81     // row click listener
82     recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext(),
83     recyclerView, new RecyclerTouchListener.ClickListener() {
84         @Override
85         public void onClick(View view, int position) {
86             AvailableSetting setting = settingsList.get(position);
87
88             Context currentContext = Application.getContext();
89             Activity currentActivity = Application.getActivity();
90             Intent i = new Intent(currentContext, setting.getTargetActivity());
91             currentContext.startActivity(i);
92             //currentActivity.overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
93         }
94
95         @Override
96         public void onLongClick(View view, int position) {
97
98         }
99     }));
100
101     mAdapter.notifyDataSetChanged();
102 }
103
104 @Override
105 public boolean onOptionsItemSelected(MenuItem item) {
106     if (item.getItemId() == android.R.id.home) // Press Back Icon
107     {
108         finish();
109     }
110
111     return super.onOptionsItemSelected(item);
112 }
113
114 public static void setupSettingsList() {
115
116     settingsList.add(new AvailableSetting(
117         Application.getContext().getString(R.string.internet_connectivity_title),
118         Application.getContext().getString(R.string.internet_connectivity_subtitle),
119         R.drawable.ic_setting_internet_connectivity,
120         InternetConnectivityActivity.class
121     ));
122
123     settingsList.add(new AvailableSetting(
124         Application.getContext().getString(R.string.bluetooth_settings_title),
125         Application.getContext().getString(R.string.bluetooth_settings_subtitle),
126         R.drawable.ic_setting_bluetooth,
127         BluetoothSettingsActivity.class
128     ));
129
130     settingsList.add(new AvailableSetting(
131         Application.getContext().getString(R.string.voice_recognition_settings_title),
132         Application.getContext().getString(R.string.voice_recognition_settings_subtitle),
133         R.drawable.ic_setting_voice_recognition,
134         VoiceRecognitionSettingsActivity.class
135     ));
136
137     settingsList.add(new AvailableSetting(
138         Application.getContext().getString(R.string.commands_settings_title),
139         Application.getContext().getString(R.string.commands_settings_subtitle),
140         R.drawable.ic_setting_commands_3,
141         CommandsSettingsActivity.class
142     ));
143
144     settingsList.add(new AvailableSetting(
145         Application.getContext().getString(R.string.contacts_communication_title),
146         Application.getContext().getString(R.string.contacts_communication_subtitle),
147         R.drawable.ic_setting_contacts_and_communication,
148         ContactsCommunicationActivity.class
149     ));
150 }
```

```
151     settingsList.add(new AvailableSetting(
152         Application.getContext().getString(R.string.appearance_miscellaneous_title),
153         Application.getContext().getString(R.string.appearance_miscellaneous_subtitle),
154         R.drawable.ic_setting_appearance_and_miscellaneous,
155         AppearanceMiscellaneousActivity.class
156     ));
157
158     settingsList.add(new AvailableSetting(
159         Application.getContext().getString(R.string.developer_console_activity_title),
160         Application.getContext().getString(R.string.developer_console_subtitle),
161         R.drawable.ic_setting_developer_console,
162         DeveloperConsoleActivity.class
163     ));
164
165     settingsList.add(new AvailableSetting(
166         Application.getContext().getString(R.string.information_help_title),
167         Application.getContext().getString(R.string.information_help_subtitle),
168         R.drawable.ic_setting_information_and_help,
169         InformationHelpActivity.class
170     ));
171
172     // notify adapter about data set changes
173     // so that it will render the list with new data
174     // mAdapter.notifyDataSetChanged();
175 }
176
177
178 }
179
```

```
1 package com.example.gui_design.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import android.support.v4.content.ContextCompat;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.gui_design.Application;
12 import com.example.gui_design.R;
13
14 public class AppearanceMiscellaneousActivity extends AppCompatActivity {
15
16     private static Toolbar toolbar;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.appearance_miscellaneous_layout);
22         Application.setActivity(this);
23         Application.setContext(this);
24
25         toolbar = this.findViewById(R.id.toolbar);
26
27
28         setSupportActionBar(toolbar);
29
30         // add back arrow to toolbar
31         if (getSupportActionBar() != null){
32             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
33             getSupportActionBar().setDisplayShowHomeEnabled(true);
34
35             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
36                 abc_ic_ab_back_material);
37             upArrow.setColorFilter(Color.parseColor("#000000"), PorterDuff.Mode.SRC_ATOP);
38             getSupportActionBar().setHomeAsUpIndicator(upArrow);
39         }
40     }
41
42     @Override
43     public boolean onOptionsItemSelected(MenuItem item) {
44         if (item.getItemId() == android.R.id.home) // Press Back Icon
45         {
46             finish();
47         }
48
49         return super.onOptionsItemSelected(item);
50     }
51 }
52
```

```

1 package com.example.gui_design.settingsActivities;
2
3 import android.content.Context;
4 import android.content.res.Resources;
5 import android.content.res.TypedArray;
6 import android.graphics.Canvas;
7 import android.graphics.Rect;
8 import android.graphics.drawable.Drawable;
9 import android.support.v7.widget.LinearLayoutManager;
10 import android.support.v7.widget.RecyclerView;
11 import android.util.TypedValue;
12 import android.view.View;
13
14 /**
15  * Created by Ravi on 30/10/15.
16  * updated by Ravi on 14/11/17
17 */
18 public class MyDividerItemDecoration extends RecyclerView.ItemDecoration {
19
20     private static final int[] ATTRS = new int[]{
21         android.R.attr.listDivider
22     };
23
24     public static final int HORIZONTAL_LIST = LinearLayoutManager.HORIZONTAL;
25
26     public static final int VERTICAL_LIST = LinearLayoutManager.VERTICAL;
27
28     private Drawable mDivider;
29     private int mOrientation;
30     private Context context;
31     private int margin;
32
33     public MyDividerItemDecoration(Context context, int orientation, int margin) {
34         this.context = context;
35         this.margin = margin;
36         final TypedArray a = context.obtainStyledAttributes(ATTRS);
37         mDivider = a.getDrawable(0);
38         a.recycle();
39         setOrientation(orientation);
40     }
41
42     public void setOrientation(int orientation) {
43         if (orientation != HORIZONTAL_LIST && orientation != VERTICAL_LIST) {
44             throw new IllegalArgumentException("invalid orientation");
45         }
46         mOrientation = orientation;
47     }
48
49     @Override
50     public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
51         if (mOrientation == VERTICAL_LIST) {
52             drawVertical(c, parent);
53         } else {
54             drawHorizontal(c, parent);
55         }
56     }
57
58     public void drawVertical(Canvas c, RecyclerView parent) {
59         final int left = parent.getPaddingLeft();
60         final int right = parent.getWidth() - parent.getPaddingRight();
61
62         final int childCount = parent.getChildCount();
63         for (int i = 0; i < childCount; i++) {
64             final View child = parent.getChildAt(i);
65             final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child
66                 .getLayoutParams();
67             final int top = child.getBottom() + params.bottomMargin;
68             final int bottom = top + mDivider.getIntrinsicHeight();
69             mDivider.setBounds(left + dpToPx(margin), top, right - dpToPx(margin), bottom);
70             mDivider.draw(c);
71         }
72     }
73
74     public void drawHorizontal(Canvas c, RecyclerView parent) {
75         final int top = parent.getPaddingTop();
76         final int bottom = parent.getHeight() - parent.getPaddingBottom();
77

```

```
78     final int childCount = parent.getChildCount();
79     for (int i = 0; i < childCount; i++) {
80         final View child = parent.getChildAt(i);
81         final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child
82             .getLayoutParams();
83         final int left = child.getRight() + params.rightMargin;
84         final int right = left + mDivider.getIntrinsicHeight();
85         mDivider.setBounds(left, top + dpToPx(margin), right, bottom - dpToPx(margin));
86         mDivider.draw(c);
87     }
88 }
89
90 @Override
91 public void getItemOffsets(Rect outRect, View view, RecyclerView parent, RecyclerView.State state
92 ) {
93     if (mOrientation == VERTICAL_LIST) {
94         outRect.set(0, 0, 0, mDivider.getIntrinsicHeight());
95     } else {
96         outRect.set(0, 0, mDivider.getIntrinsicWidth(), 0);
97     }
98
99     private int dpToPx(int dp) {
100         Resources r = context.getResources();
101         return Math.round(TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, dp, r.
102             getDisplayMetrics()));
103     }
104 }
```

```

1 package com.example.gui_design;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.speech.RecognitionListener;
6 import android.speech.RecognizerIntent;
7 import android.speech.SpeechRecognizer;
8 import android.view.View;
9
10 import com.example.gui_design.bluetoothActivities.BluetoothConnection;
11
12 import org.apache.commons.lang3.StringUtils;
13
14 import java.text.Normalizer;
15 import java.text.SimpleDateFormat;
16 import java.util.ArrayList;
17 import java.util.Arrays;
18 import java.util.Calendar;
19 import java.util.Date;
20 import java.util.Locale;
21
22 public class GoogleSpeechRecognizer extends MainActivity implements RecognitionListener {
23
24     private static SpeechRecognizer speechRecognizer = null;
25
26     private boolean forceCancel = false;
27
28     public static void startSpeechListening(){
29
30
31         String language = "es-ES";
32         Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
33         intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,language);
34         intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, language);
35         intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE, language);
36         intent.putExtra(RecognizerIntent.EXTRA_ONLY_RETURN_LANGUAGE_PREFERENCE, language);
37         intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE, Application.getActivity().
getPackageName());
38         intent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS,true);
39         if (Application.isSystemOnline()==false) { intent.putExtra(RecognizerIntent.
EXTRA_PREFER_OFFLINE,true); }
40         speechRecognizer.startListening(intent);
41
42
43         if (Application.isSystemOnline()) {
44             Thread autoStop = new Thread() {
45                 @Override
46                 public void run() {
47                     super.run();
48                     try {
49                         Thread.sleep(4000);
50                         forceStopRecognizer();
51                     } catch (Throwable th) {
52                         Application.toastMessage(th.toString());
53                     }
54                 }
55             };
56             autoStop.start();
57         }
58     }
59
60
61     public static void setupRecognizer(){
62         speechRecognizer = SpeechRecognizer.createSpeechRecognizer(Application.getContext());
63         speechRecognizer.setRecognitionListener(new GoogleSpeechRecognizer());
64     }
65
66     public static void shutdownRecognizer(){
67         speechRecognizer.stopListening();
68         speechRecognizer.cancel();
69         speechRecognizer.destroy();
70         speechRecognizer = null;
71         System.gc();
72     }
73
74     public static void forceStopRecognizer(){
75         Application.getActivity().runOnUiThread(

```

```
76     new Runnable() {
77         @Override
78         public void run() {
79             speechRecognizer.stopListening();
80             MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.
81 LISTENING_KEYWORD);
82         }
83     );
84 }
85 }
86
87
88 public static void analizeListenerResults(ArrayList<String> data, float[] confidenceScore) {
89     outerloop:
90     for (int i = 0; i < data.size(); i++) {
91         String hypothesis = Normalizer.normalize((String) data.get(i)).toLowerCase(), Normalizer
92 .Form.NFKD);
93         String[] hypothesisSplit = hypothesis.split(" ");
94
95         if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
96 .contains("velador")) {
97             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 0);
98             break outerloop;
99         }
100
101         if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
102 .contains("estufa")) {
103             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 1);
104             break outerloop;
105         }
106         if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
107 .contains("ventilador")) {
108             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 2);
109             break outerloop;
110         }
111         if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
112 .contains("luz")) {
113             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 3);
114             break outerloop;
115         }
116         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
117 contains("velador")) {
118             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 0);
119             break outerloop;
120         }
121         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
122 contains("estufa")) {
123             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 1);
124             break outerloop;
125         }
126         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
127 contains("ventilador")) {
128             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 2);
129             break outerloop;
130         }
131         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
132 contains("luz")) {
133             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 3);
134             break outerloop;
135         }
136         if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
137 .contains("todo")) {
138             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_EVERYTHING_ON
139 , 0);
140             break outerloop;
141     }
142 }
```

```

137         }
138
139         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
140             contains("todo")) {
140             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.
141             TURN_EVERYTHING_OFF, 0);
141             break outerloop;
142         }
143
144         if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(hypothesisSplit).
145             contains("enfermera")) {
145             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.CALL_NURSE, 0);
146             break outerloop;
147         }
148
149         if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(hypothesisSplit).
150             contains("enfermeria")) {
150             BluetoothConnection.composeAndSend(BluetoothConnection.commandType.CALL_NURSE, 0);
151             break outerloop;
152         }
153
154
155         if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit).
156             .contains("rele")) {
156             for (String word : hypothesisSplit) {
157                 if (StringUtils.isNumeric(word) == true) {
158                     int number = Integer.parseInt(word);
159                     BluetoothConnection.composeAndSend(BluetoothConnection.commandType.
160                     TURN_RELAY_ON, number);
160                     break outerloop;
161                 }
162             }
163         }
164
165         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
166             contains("rele")) {
166             for (String word : hypothesisSplit) {
167                 if (StringUtils.isNumeric(word) == true) {
168                     int number = Integer.parseInt(word);
169                     BluetoothConnection.composeAndSend(BluetoothConnection.commandType.
170                     TURN_RELAY_OFF, number);
170                     break outerloop;
171                 }
172             }
173         }
174
175         if (Arrays.asList(hypothesisSplit).contains("decir") && Arrays.asList(hypothesisSplit).
176             contains("hora")){
176             Date currentTime = Calendar.getInstance().getTime();
177             SimpleDateFormat df = new SimpleDateFormat("d-MMM-yy", new Locale("spa"));
178             String todayAsString = df.format(currentTime);
179             SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new Locale("spa"))
180             ;
180             String onlyTimeString = onlyTimeFormar.format(currentTime);
181
182             //TextToSpeechHandler.initializeTextToSpeech();
183             TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + " y son las" +
184             onlyTimeString);
184             Application.toastMessage("Decir hora");
185         }
186
187     }
188 }
189
190 @Override
191 public void onReadyForSpeech(Bundle params) {
192     MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_COMMAND);
193 }
194
195 @Override
196 public void onBeginningOfSpeech() {
197 }
198
199 @Override
200 public void onRmsChanged(float rmsdB) {
201 }
202

```

```
203     }
204
205     @Override
206     public void onBufferReceived(byte[] buffer) {
207
208     }
209
210     @Override
211     public void onEndOfSpeech() {
212
213         //Application.toastMessage("End of speech");
214     }
215
216     @Override
217     public void onError(int error) {
218         //Application.toastMessage("On error" + error);
219
220         if (forceCancel==false) {
221             PocketsphinxListener.startListening();
222             MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYWORD);
223         }
224     }
225
226     @Override
227     public void onResults(Bundle results) {
228         String str = new String();
229         ArrayList<String> resultsArrayList = results.getStringArrayList(SpeechRecognizer.
RESULTS_RECOGNITION);
230         float[] confidenceScore = results.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES);
231         analizeListenerResults(resultsArrayList, confidenceScore);
232
233         StringBuilder builder = new StringBuilder();
234         for (String line : resultsArrayList) {
235             builder.append(line + "\n");
236         }
237
238         MainActivity.textView_speechRecognitionResults.setText(builder.toString());
239         MainActivity.textView_speechRecognitionResults.setVisibility(View.VISIBLE);
240
241
242         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYWORD);
243
244         PocketsphinxListener.startListening();
245     }
246
247     @Override
248     public void onPartialResults(Bundle partialResults) {
249         String str = new String();
250         ArrayList<String> resultsArrayList = partialResults.getStringArrayList(SpeechRecognizer.
RESULTS_RECOGNITION);
251         float[] confidenceScore = partialResults.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES);
252
253         StringBuilder builder = new StringBuilder();
254         for (String line : resultsArrayList) {
255             builder.append(line + "\n");
256         }
257     }
258
259 }
260
261     @Override
262     public void onEvent(int eventType, Bundle params) {
263
264     }
265 }
266
```

```
1 package com.example.gui_design.settingsActivities;
2
3 import android.content.Context;
4 import android.support.v7.widget.RecyclerView;
5 import android.view.GestureDetector;
6 import android.view.MotionEvent;
7 import android.view.View;
8
9 /**
10 * Created by Ravi Tamada on 03/09/16.
11 * updated by Ravi on 14/11/17
12 * www.androidhive.info
13 */
14 public class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {
15
16     private GestureDetector gestureDetector;
17     private ClickListener clickListener;
18
19     public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final ClickListener clickListener) {
20         this.clickListener = clickListener;
21         gestureDetector = new GestureDetector(context, new GestureDetector.SimpleOnGestureListener() {
22             @Override
23             public boolean onSingleTapUp(MotionEvent e) {
24                 return true;
25             }
26
27             @Override
28             public void onLongPress(MotionEvent e) {
29                 View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
30                 if (child != null && clickListener != null) {
31                     clickListener.onLongClick(child, recyclerView.getChildAdapterPosition(child));
32                 }
33             }
34         });
35     }
36
37     @Override
38     public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {
39
40         View child = rv.findChildViewUnder(e.getX(), e.getY());
41         if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {
42             clickListener.onClick(child, rv.getChildAdapterPosition(child));
43         }
44         return false;
45     }
46
47     @Override
48     public void onTouchEvent(RecyclerView rv, MotionEvent e) {
49     }
50
51     @Override
52     public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {
53
54     }
55
56     public interface ClickListener {
57         void onClick(View view, int position);
58
59         void onLongClick(View view, int position);
60     }
61 }
```

```
1 package com.example.gui_design.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import android.support.v4.content.ContextCompat;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.gui_design.Application;
12 import com.example.gui_design.R;
13
14 public class BluetoothSettingsActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.bluetooth_settings_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34                 abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
36             getSupportActionBar().setHomeAsUpIndicator(upArrow);
37         }
38     }
39
40     @Override
41     public boolean onOptionsItemSelected(MenuItem item) {
42         if (item.getItemId() == android.R.id.home) // Press Back Icon
43         {
44             finish();
45         }
46
47         return super.onOptionsItemSelected(item);
48     }
49 }
```

```
1 package com.example.gui_design;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import android.support.v4.content.ContextCompat;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.MenuItem;
10
11 public class CommandsActivity extends AppCompatActivity {
12     private static Toolbar toolbar;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.commands_layout);
18         Application.setActivity(this);
19         Application.setContext(this);
20
21         toolbar = this.findViewById(R.id.toolbar);
22
23         setSupportActionBar(toolbar);
24
25         // add back arrow to toolbar
26         if (getSupportActionBar() != null) {
27             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
28             getSupportActionBar().setDisplayShowHomeEnabled(true);
29
30             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
31                 abc_ic_ab_back_material);
32             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
33             getSupportActionBar().setHomeAsUpIndicator(upArrow);
34         }
35     }
36
37     @Override
38     public boolean onOptionsItemSelected(MenuItem item) {
39         if (item.getItemId() == android.R.id.home) // Press Back Icon
40         {
41             finish();
42         }
43
44         return super.onOptionsItemSelected(item);
45     }
46 }
```

```
1 package com.example.gui_design.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import android.support.v4.content.ContextCompat;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.gui_design.Application;
12 import com.example.gui_design.R;
13
14 public class InformationHelpActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.information_help_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34                 abc_ic_ab_back_material);
35             upArrow.setColorFilter(Color.parseColor("#000000"), PorterDuff.Mode.SRC_ATOP);
36             getSupportActionBar().setHomeAsUpIndicator(upArrow);
37         }
38     }
39
40     @Override
41     public boolean onOptionsItemSelected(MenuItem item) {
42         if (item.getItemId() == android.R.id.home) // Press Back Icon
43         {
44             finish();
45         }
46
47         return super.onOptionsItemSelected(item);
48     }
49 }
```

```

1 package com.example.gui_design.bluetoothActivities;
2
3 import android.graphics.Color;
4 import android.graphics.PorterDuff;
5 import android.graphics.drawable.ColorDrawable;
6 import android.graphics.drawable.Drawable;
7 import android.support.v7.widget.RecyclerView;
8 import android.view.LayoutInflater;
9 import android.view.View;
10 import android.view.ViewGroup;
11 import android.widget.ImageView;
12 import android.widget.TextView;
13
14 import com.example.gui_design.Application;
15 import com.example.gui_design.R;
16
17 import java.util.List;
18
19 public class DeviceListAdapter extends RecyclerView.Adapter<DeviceListAdapter.ViewHolder> {
20
21     private List<AvailableDevice> deviceList;
22
23     public class ViewHolder extends RecyclerView.ViewHolder {
24         public TextView textView_device_name;
25         public ImageView imgView_device_state_icon;
26
27         public ViewHolder(View view) {
28             super(view);
29             textView_device_name = (TextView) view.findViewById(R.id.textView_bluetooth_device_name);
30             imgView_device_state_icon = (ImageView) view.findViewById(R.id.
31             imageView_bluetooth_device_state_icon);
32         }
33     }
34
35     public DeviceListAdapter(List<AvailableDevice> deviceList) {
36         this.deviceList = deviceList;
37     }
38
39     @Override
40     public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
41         View itemView = LayoutInflater.from(parent.getContext())
42             .inflate(R.layout.bluetooth_list_layout_style, parent, false);
43
44         return new ViewHolder(itemView);
45     }
46
47     @Override
48     public void onBindViewHolder(ViewHolder holder, int position) {
49         AvailableDevice device = deviceList.get(position);
50         holder.textView_device_name.setText(device.getDeviceName());
51         holder.textView_device_name.setTextAppearance(device.getTextAppearance());
52
53         int iconID = device.getDeviceStateIconID();
54
55         if (iconID!=0) {
56             Drawable icon = Application.getContext().getDrawable(iconID);
57             icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary), PorterDuff.
58             Mode.SRC_ATOP);
59             holder.imgView_device_state_icon.setImageDrawable(icon);
60         } else {
61             holder.imgView_device_state_icon.clearColorFilter();
62             Drawable transparentDrawable = new ColorDrawable(Color.TRANSPARENT);
63             holder.imgView_device_state_icon.setImageDrawable(transparentDrawable);
64         }
65     }
66
67     @Override
68     public int getItemCount() {
69         return deviceList.size();
70     }
71

```

```
1 package com.example.gui_design.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import android.support.v4.content.ContextCompat;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.gui_design.Application;
12 import com.example.gui_design.R;
13
14 public class DeveloperConsoleActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.developer_console_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34                 abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
36             getSupportActionBar().setHomeAsUpIndicator(upArrow);
37         }
38     }
39
40     @Override
41     public boolean onOptionsItemSelected(MenuItem item) {
42         if (item.getItemId() == android.R.id.home) // Press Back Icon
43         {
44             finish();
45         }
46
47         return super.onOptionsItemSelected(item);
48     }
49 }
```

```

1 package com.example.gui_design;
2
3 import android.animation.ValueAnimator;
4 import android.app.Activity;
5 import android.content.Context;
6 import android.content.Intent;
7 import android.view.View;
8 import android.view.animation.AccelerateDecelerateInterpolator;
9
10 import com.example.gui_design.bluetoothActivities.BluetoothActivity;
11 import com.example.gui_design.settingsActivities.settings_layout;
12
13 public class MainMenuHandler extends MainActivity{
14
15     public static float dpiConstant;
16     public static float radialMenuRadius = 263.0f;
17     public static boolean menuUncoiled = false;
18     public static boolean isAnimationRunning = false;
19     public static Context mainMenuContext;
20     public static Activity mainMenuActivity;
21
22     public MainMenuHandler(Context context, Activity activity){
23         this.mainMenuActivity = activity;
24         this.mainMenuContext = context;
25     }
26
27     public static void menuButtonOnClick(){
28         long animationDuration = 850;
29
30         if(isAnimationRunning==false) {
31             isAnimationRunning = true;
32
33             if (menuUncoiled == false) {
34                 ValueAnimator animatorCommands = ValueAnimator.ofFloat(360, 72);
35                 animatorCommands.setDuration(animationDuration);
36                 animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
37                 animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
38                     @Override
39                     public void onAnimationUpdate(ValueAnimator animation) {
40                         float value = ((Float) (animation.getAnimatedValue()))
41                             .floatValue();
42
43                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
44                             radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
45                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
46                             radialMenuRadius * translationX));
47                         if (value > 180.0f) {
48                             translationY = -translationY;
49                         }
50
51                         imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
52                         imgBtn_commands.setTranslationY(convertDpToPixel(translationY));
53                     }
54                 });
55                 imgBtn_commands.setVisibility(View.VISIBLE);
56                 animatorCommands.start();
57
58                 ValueAnimator animatorContacts = ValueAnimator.ofFloat(360, 144);
59                 animatorContacts.setDuration(animationDuration);
60                 animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
61                 animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
62                     @Override
63                     public void onAnimationUpdate(ValueAnimator animation) {
64                         float value = ((Float) (animation.getAnimatedValue()))
65                             .floatValue();
66
67                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
68                             radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
69                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
70                             radialMenuRadius * translationX));
71                         if (value > 180.0f) {
72                             translationY = -translationY;
73                         }
74
75                         imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
76                         imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
77
78                     }
79                 });
80             }
81         }
82     }
83
84 }
```

```

74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
    }
    imgBtn_contacts.setVisibility(View.VISIBLE);
    animatorContacts.start();

    ValueAnimator animatorSettings = ValueAnimator.ofFloat(360, 216);
    animatorSettings.setDuration(animationDuration);
    animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
    animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
        @Override
        public void onAnimationUpdate(ValueAnimator animation) {
            float value = ((Float) (animation.getAnimatedValue()))
                .floatValue();

            float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
                radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
            float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
                radialMenuRadius * translationX));
            if (value > 180.0f) {
                translationY = -translationY;
            }

            imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
            imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
        }
    });
    imgBtn_settings.setVisibility(View.VISIBLE);
    animatorSettings.start();

    ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(360, 288);
    animatorBluetooth.setDuration(animationDuration);
    animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
    animatorBluetooth.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
        @Override
        public void onAnimationUpdate(ValueAnimator animation) {
            float value = ((Float) (animation.getAnimatedValue()))
                .floatValue();

            float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
                radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
            float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
                radialMenuRadius * translationX));
            if (value > 180.0f) {
                translationY = -translationY;
            }

            imgBtn_bluetooth.setTranslationX(convertDpToPixel(translationX));
            imgBtn_bluetooth.setTranslationY(convertDpToPixel(translationY));
        }
    });
    imgBtn_bluetooth.setVisibility(View.VISIBLE);
    animatorBluetooth.start();

    imgBtn_commands.animate().alpha(1.0f).setDuration(animationDuration).setInterpolator(
        new AccelerateDecelerateInterpolator());
    imgBtn_contacts.animate().alpha(1.0f).setDuration(animationDuration).setInterpolator(
        new AccelerateDecelerateInterpolator());
    imgBtn_settings.animate().alpha(1.0f).setDuration(animationDuration).setInterpolator(
        new AccelerateDecelerateInterpolator());
    imgBtn_bluetooth.animate().alpha(1.0f).setDuration(animationDuration).setInterpolator(
        new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
        @Override
        public void run() {
            isAnimationRunning = false;
        }
    });

    menuUncoiled = true;

} else {
    ValueAnimator animatorCommands = ValueAnimator.ofFloat(72, 360);
}

```

```

143         animatorCommands.setDuration(animationDuration);
144         animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
145         animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
146             @Override
147             public void onAnimationUpdate(ValueAnimator animation) {
148                 float value = ((Float) (animation.getAnimatedValue()))
149                     .floatValue();
150
151                 float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
152                     radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
153                 float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
154                     radialMenuRadius * translationX));
155                 if (value > 180.0f) {
156                     translationY = -translationY;
157                 }
158
159                 imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
160                 imgBtn_commands.setTranslationY(convertDpToPixel(translationY));
161             }
162         });
163         imgBtn_commands.setVisibility(View.VISIBLE);
164         animatorCommands.start();
165
166         ValueAnimator animatorContacts = ValueAnimator.ofFloat(144, 360);
167         animatorContacts.setDuration(animationDuration);
168         animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
169         animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
170             @Override
171             public void onAnimationUpdate(ValueAnimator animation) {
172                 float value = ((Float) (animation.getAnimatedValue()))
173                     .floatValue();
174
175                 float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
176                     radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
177                 float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
178                     radialMenuRadius * translationX));
179                 if (value > 180.0f) {
180                     translationY = -translationY;
181                 }
182
183                 imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
184                 imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
185             }
186         });
187         imgBtn_contacts.setVisibility(View.VISIBLE);
188         animatorContacts.start();
189
190         ValueAnimator animatorSettings = ValueAnimator.ofFloat(216, 360);
191         animatorSettings.setDuration(animationDuration);
192         animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
193         animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
194             @Override
195             public void onAnimationUpdate(ValueAnimator animation) {
196                 float value = ((Float) (animation.getAnimatedValue()))
197                     .floatValue();
198
199                 float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
200                     radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
201                 float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
202                     radialMenuRadius * translationX));
203                 if (value > 180.0f) {
204                     translationY = -translationY;
205                 }
206
207                 imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
208                 imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
209             }
210         });
211         imgBtn_settings.setVisibility(View.VISIBLE);
212         animatorSettings.start();
213
214         ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(288, 360);
215         animatorBluetooth.setDuration(animationDuration);
216         animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());

```

```

214         animatorBluetooth.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
215             @Override
216             public void onAnimationUpdate(ValueAnimator animation) {
217                 float value = ((Float) (animation.getAnimatedValue()))
218                     .floatValue();
219
220                 float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
221                     radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
222                 float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
223                     radialMenuRadius * translationX));
224                 if (value > 180.0f) {
225                     translationY = -translationY;
226                 }
227
228                 imgBtn_bluetooth.setTranslationX(convertDpToPixel(translationX));
229                 imgBtn_bluetooth.setTranslationY(convertDpToPixel(translationY));
230             }
231             imgBtn_bluetooth.setVisibility(View.VISIBLE);
232             animatorBluetooth.start();
233
234             imgBtn_commands.animate().alpha(0.0f).setDuration(animationDuration).setInterpolator(
235                 new AccelerateDecelerateInterpolator());
236             imgBtn_contacts.animate().alpha(0.0f).setDuration(animationDuration).setInterpolator(
237                 new AccelerateDecelerateInterpolator());
238             imgBtn_settings.animate().alpha(0.0f).setDuration(animationDuration).setInterpolator(
239                 new AccelerateDecelerateInterpolator());
240             imgBtn_bluetooth.animate().alpha(0.0f).setDuration(animationDuration).setInterpolator(
241                 new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
242                 @Override
243                 public void run() {
244                     isAnimationRunning = false;
245                 }
246             });
247         }
248
249         public static void commandsButtonOnClick() {
250             Context currentContext = Application.getContext();
251             Activity currentActivity = Application.getActivity();
252             Intent i = new Intent(currentContext, CommandsActivity.class);
253             currentContext.startActivity(i);
254         }
255
256         public static void contactsButtonOnClick() {
257             Context currentContext = Application.getContext();
258             Activity currentActivity = Application.getActivity();
259             Intent i = new Intent(currentContext, ContactsActivity.class);
260             currentContext.startActivity(i);
261         }
262
263         public static void bluetoothButtonOnClick() {
264             Context currentContext = Application.getContext();
265             Activity currentActivity = Application.getActivity();
266             Intent i = new Intent(currentContext, BluetoothActivity.class);
267             currentContext.startActivity(i);
268         }
269
270         public static void settingsButtonOnClick() {
271             Intent i = new Intent(Application.getContext(), settings_layout.class);
272             mainMenuContext.startActivity(i);
273             //mainMenuActivity.overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
274         }
275
276
277
278
279
280
281
282
283     /**
284      * This method converts dp unit to equivalent pixels, depending on device density.

```

```
285     *
286     * param dp A value in dp (density independent pixels) unit. Which we need to convert into
287     * pixels
288     * return A float value to represent px equivalent to dp depending on device density
289     */
290     public static float convertDpToPixel(float dp){ return dp * dpiConstant; }
291
292     /**
293      * This method converts device specific pixels to density independent pixels.
294      *
295      * param px A value in px (pixels) unit. Which we need to convert into db
296      * return A float value to represent dp equivalent to px value
297      */
298     public static float convertPixelsToDp(float px){
299         return px / dpiConstant;
300     }
301 }
```

```

1 package com.example.gui_design;
2
3 import android.Manifest;
4 import android.content.pm.PackageManager;
5 import android.os.Bundle;
6 import android.support.v4.app.ActivityCompat;
7 import android.support.v4.content.ContextCompat;
8 import android.support.v7.app.AppCompatActivity;
9 import android.util.DisplayMetrics;
10 import android.view.View;
11 import android.widget.ImageButton;
12 import android.widget.TextView;
13
14 import com.example.gui_design.bluetoothActivities.BluetoothConnection;
15 import com.example.gui_design.settingsActivities.settings_layout;
16
17 public class MainActivity extends AppCompatActivity {
18
19     static ImageButton imgBtn_microphone;
20     static ImageButton imgBtn_microphoneBackground;
21     static ImageButton imgBtn_commands;
22     static ImageButton imgBtn_contacts;
23     static ImageButton imgBtn_bluetooth;
24     static ImageButton imgBtn_settings;
25     static ImageButton imgBtn_menu;
26     static MainMenuHandler mainMenuHandler;
27     static TextView textView_speechRecognitionResults;
28
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_main);
34
35         if(savedInstanceState == null) {
36
37             mainMenuHandler = new MainMenuHandler(this, this);
38             Application.setActivity(this);
39             Application.setContext(this);
40             MainMenuHandler.dpiConstant = ((float) this.getResources().getDisplayMetrics().densityDpi
41             / DisplayMetrics.DENSITY_DEFAULT);
42
43             imgBtn_microphone = findViewById(R.id.imgBtn_microphone);
44             imgBtn_microphoneBackground = findViewById(R.id.imgBtn_microphone_background);
45             imgBtn_commands = findViewById(R.id.imgBtn_commands);
46             imgBtn_contacts = findViewById(R.id.imgBtn_contacts);
47             imgBtn_bluetooth = findViewById(R.id.imgBtn_bluetooth);
48             imgBtn_settings = findViewById(R.id.imgBtn_settings);
49             imgBtn_menu = findViewById(R.id.imgBtn_menu);
50             textView_speechRecognitionResults = findViewById(R.id.textView_speechRecognitionResults);
51
52             if (ContextCompat.checkSelfPermission(this,
53                 Manifest.permission.RECORD_AUDIO)
54                 != PackageManager.PERMISSION_GRANTED) {
55                 if (ActivityCompat.shouldShowRequestPermissionRationale(this,
56                     Manifest.permission.RECORD_AUDIO)) {
57                     ActivityCompat.requestPermissions(this,
58                         new String[]{Manifest.permission.RECORD_AUDIO},
59                         527);
60                 }
61             } else {
62                 GoogleSpeechRecognizer.setupRecognizer();
63                 PocketsphinxListener.setupRecognizer();
64                 PocketsphinxListener.startListening();
65             }
66         }
67
68         settings_layout.setupSettingsList();
69
70         BluetoothConnection.initBTAutoConnectService();
71
72     }
73 }
74
75     @Override
76     public void onRequestPermissionsResult(int requestCode,

```

```
77                                     String permissions[], int[] grantResults) {
78     switch (requestCode) {
79         case 527: {
80             // If request is cancelled, the result arrays are empty.
81             if (grantResults.length > 0
82                 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
83
84                 GoogleSpeechRecognizer.setupRecognizer();
85                 PocketsphinxListener.setupRecognizer();
86
87             } else { }
88         }
89     }
90 }
91
92
93     public void onClickListener_menu(View view){ mainMenuHandler.menuButtonOnClick(); }
94
95     public void onClickListener_commands(View view){ mainMenuHandler.commandsButtonOnClick(); }
96
97     public void onClickListener_settings(View view){ mainMenuHandler.settingsButtonOnClick(); }
98
99     public void onClickListener_contacts(View view){ mainMenuHandler.contactsButtonOnClick(); }
100
101    public void onClickListener_bluetooth(View view){ mainMenuHandler.bluetoothButtonOnClick(); }
102
103    public void onClickListener_microphone(View view){ MicrophoneHandler.microphoneOnClick(); }
104
105    @Override
106    public void onDestroy() {
107
108        PocketsphinxListener.shutdownRecognizer();
109        GoogleSpeechRecognizer.shutdownRecognizer();
110        BluetoothConnection.closeAllConnections();
111        TextToSpeechHandler.shutdownTextSpeech();
112
113        super.onDestroy();
114    }
115
116 }
117
```

```

1 package com.example.gui_design;
2
3 import android.graphics.PorterDuff;
4 import android.view.animation.AccelerateDecelerateInterpolator;
5
6 import java.util.concurrent.Callable;
7 import java.util.concurrent.ExecutorService;
8 import java.util.concurrent.Executors;
9 import java.util.concurrent.Future;
10
11 public class MicrophoneHandler extends MainActivity {
12
13     public enum MicrophoneState { INHIBITED, LISTENING_KEYWORD, LISTENING_COMMAND}
14
15     static private MicrophoneState currentState = MicrophoneState.LISTENING_KEYWORD;
16
17     public static void microphoneOnClick(){
18         switch (currentState){
19             case LISTENING_KEYWORD:
20                 PocketsphinxListener.stopRecognizer();
21                 setState(MicrophoneState.INHIBITED);
22                 break;
23             case LISTENING_COMMAND:
24                 GoogleSpeechRecognizer.forceStopRecognizer();
25                 //PocketsphinxListener.startListening();
26                 setState(MicrophoneState.LISTENING_KEYWORD);
27                 break;
28             case INHIBITED:
29                 setState(MicrophoneState.LISTENING_KEYWORD);
30                 PocketsphinxListener.startListening();
31                 break;
32         }
33     }
34
35
36     public static MicrophoneState getState(){
37         return currentState;
38     }
39
40     public static void setState(MicrophoneState desiredState){
41         switch (desiredState){
42             case INHIBITED:
43                 imgBtn_microphoneBackground.clearColorFilter();
44                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.
ColorBackMicInhibited), PorterDuff.Mode.SRC_ATOP);
45                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_off);
46                 endAnimation();
47                 currentState = MicrophoneState.INHIBITED;
48                 break;
49             case LISTENING_KEYWORD:
50                 imgBtn_microphoneBackground.clearColorFilter();
51                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.
ColorBackMicKeyword), PorterDuff.Mode.SRC_ATOP);
52                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
53                 endAnimation();
54                 currentState = MicrophoneState.LISTENING_KEYWORD;
55                 break;
56             case LISTENING_COMMAND:
57                 imgBtn_microphoneBackground.clearColorFilter();
58                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.
ColorBackMicCommand), PorterDuff.Mode.SRC_ATOP);
59                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
60                 startAnimation();
61                 currentState = MicrophoneState.LISTENING_COMMAND;
62                 break;
63         }
64     }
65
66     private static ExecutorService executor = Executors.newSingleThreadExecutor();
67     private static boolean terminateAnimation = false;
68     private static boolean animationHasFinished = false;
69
70     public static void startAnimation(){
71         terminateAnimation = false;
72         Future<Void> future = executor.submit(new Callable<Void>() {
73             public Void call() throws Exception {
74                 try {

```

```
75             while (terminateAnimation == false) {
76                 imgBtn_microphoneBackground.animate().scaleX(0.90f).setDuration(350).
77                     setInterpolator(new AccelerateDecelerateInterpolator());
78                 imgBtn_microphone.animate().scaleX(0.90f).setDuration(350).setInterpolator(
79                     new AccelerateDecelerateInterpolator());
80                 imgBtn_microphoneBackground.animate().scaleY(0.90f).setDuration(350).
81                     setInterpolator(new AccelerateDecelerateInterpolator());
82                 imgBtn_microphone.animate().scaleY(0.90f).setDuration(350).setInterpolator(
83                     new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
84                     @Override
85                     public void run() {
86                         animationHasFinished = true;
87                     }
88                 });
89                 while(animationHasFinished==false);
90                 animationHasFinished= false;
91                 imgBtn_microphoneBackground.animate().scaleX(1.00f).setDuration(350).
92                     setInterpolator(new AccelerateDecelerateInterpolator());
93                 imgBtn_microphone.animate().scaleX(1.00f).setDuration(350).setInterpolator(
94                     new AccelerateDecelerateInterpolator());
95                 imgBtn_microphoneBackground.animate().scaleY(1.00f).setDuration(350).
96                     setInterpolator(new AccelerateDecelerateInterpolator());
97                 imgBtn_microphone.animate().scaleY(1.00f).setDuration(350).setInterpolator(
98                     new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
99                     @Override
100                    public void run() {
101                        animationHasFinished = true;
102                    }
103                });
104            }
105        });
106    }
107
108    public static void endAnimation(){
109        terminateAnimation = true;
110    }
111
112 }
113
```

```

1 package com.example.gui_design;
2
3 import java.io.File;
4
5 import edu.cmu.pocketsphinx.Assets;
6 import edu.cmu.pocketsphinx.Hypothesis;
7 import edu.cmu.pocketsphinx.RecognitionListener;
8 import edu.cmu.pocketsphinx.SpeechRecognizer;
9 import edu.cmu.pocketsphinx.SpeechRecognizerSetup;
10
11 public final class PocketsphinxListener implements RecognitionListener {
12     private static SpeechRecognizer recognizer;
13     private static String KEYPHRASE = "asistente";
14     private static final String KWS_SEARCH = "wakeup";
15
16
17     @Override
18     public void onPartialResult(Hypothesis hypothesis) {
19         if (hypothesis == null)
20             return;
21
22         String text = hypothesis.getHypstr();
23
24         if (text.contains(KEYPHRASE)) {
25
26             Application.toastMessage(KEYPHRASE);
27             recognizer.stop();
28             //recognizer.cancel();
29             //recognizer.shutdown();
30             GoogleSpeechRecognizer.startSpeechListening();
31             //recognizer.startListening(KWS_SEARCH);
32         }
33     }
34
35
36
37     @Override
38     public void onResult(Hypothesis hypothesis) {
39         if (hypothesis != null) {
40             String text = hypothesis.getHypstr();
41
42         }
43     }
44
45     @Override
46     public void onBeginningOfSpeech() {
47     }
48
49     @Override
50     public void onEndOfSpeech() {
51
52     }
53
54     public static void setupRecognizer() {
55         try {
56             Assets assets = new Assets(getApplicationContext());
57             File assetDir = assets.syncAssets();
58             recognizer = SpeechRecognizerSetup.defaultSetup()
59                 .setAcousticModel(new File(assetDir, "es-es-ptm"))
60                 .setDictionary(new File(assetDir, "cmudict-es-es.dict"))
61                 .setKeywordThreshold(Float.MIN_VALUE)
62
63                 .setRawLogDir(assetDir) // To disable logging of raw audio comment out this call (
64                 // takes a lot of space on the device)
65
66                 .getRecognizer();
67             recognizer.addListener(new PocketsphinxListener());
68             recognizer.addKeyphraseSearch(KWS_SEARCH, KEYPHRASE);
69         } catch (Throwable th) {
70             Application.toastMessage(th.toString());
71         }
72     }
73
74     public static void startListening(){
75         recognizer.startListening(KWS_SEARCH);
76     }

```

```
77     public static void shutdownRecognizer() {
78         if (recognizer != null) {
79             recognizer.cancel();
80             recognizer.shutdown();
81         }
82     }
83
84     public static void stopRecognizer() {
85         recognizer.stop();
86     }
87
88     @Override
89     public void onError(Exception error) {
90         Application.toastMessage("Error: " + error.toString());
91     }
92
93     @Override
94     public void onTimeout() {
95         Application.toastMessage("Timeout!");
96     }
97
98
99
100 }
101
102
```

```

1 package com.example.gui_design;
2
3 import android.speech.tts.TextToSpeech;
4 import android.speech.tts.Voice;
5
6 import java.util.Locale;
7 import java.util.Set;
8
9 public class TextToSpeechHandler {
10
11     private static TextToSpeech textToSpeech;
12     public static boolean isInitialized = false;
13
14     public static void initializeTextToSpeech() {
15         if (isInitialized == false) {
16             textToSpeech = new TextToSpeech(Application.getContext(), new TextToSpeech.OnInitListener(
17                 ) {
18                     @Override
19                     public void onInit(int status) {
20                         try {
21                             if (status != TextToSpeech.ERROR) {
22                                 textToSpeech.setLanguage(new Locale("spa"));
23                                 //textToSpeech.setLanguage(Locale.UK);
24                                 //textToSpeech.setOnUtteranceProgressListener(utteranceListener);
25                                 textToSpeech.setPitch(1.0f);
26                                 textToSpeech.setSpeechRate(1.0f);
27                                 textToSpeech.getVoices();
28                                 isInitialized = true;
29                             }
30                         } catch (Throwable th) {
31                             Application.toastMessage(th.toString());
32                         }
33                     });
34                 }
35             }
36
37             public static void addToSpeakingQueue(String text) {
38                 try {
39                     if (isInitialized) {
40                         textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, null, "232");
41                     }
42                 } catch (Throwable th) {
43                     th.printStackTrace();
44                     Application.toastMessage(th.toString());
45                 }
46             }
47
48             public static void shutdownTextSpeech() {
49                 if (textToSpeech != null) {
50
51                     textToSpeech.stop();
52                     textToSpeech.shutdown();
53                 }
54             }
55
56             public static Set<Voice> getVoicesList() {
57                 if (isInitialized) {
58                     return textToSpeech.getVoices();
59                 } else {
60                     return null;
61                 }
62             }
63
64             public static void setVoice(Voice targetVoice) {
65                 if (isInitialized) {
66                     textToSpeech.setVoice(targetVoice);
67                 }
68             }
69
70
71
72     }
73
74
75

```