SENSIYA SDK for Android

Integration Guide

Introduction

SENSIYA for Android is a revolutionary contextual awareness SDK that allows you to personalize your application's content and UI to match each unique user's needs, patterns and real life behaviour.
Using SENSIYA SDK you can create next-genenration experiences, more timely and contextual than before.

This document walks you through the following:

- Integrating the SDK
- Running the SDK
- Getting asynchronous updates
- Calling synchronous API
- GeoFencing
- Proguard rules

Integrating the SDK

Follow these two simple steps to integrate the SDK in your Android application:

Step 1: Add the SENSIYA SDK to your project

- Copy the SensiyaSDK.jar file you have downloaded from the website to your projects libs directory (if libs directory does not exist, create one).
- Add a reference to the above JAR file in your project.

Step 2: Update your Android Manifest file

1. Add the following permissions to the manifest section. In addition to the mandatory permissions, you can choose to add on the permissions required for the module you wish to implement, as specified below.

   Adding more permission will mean higher accuracy and higher confidence level per parameter.

Mandatory basic permissions:

```
<uses-permission android:name="android.permission.INTERNET" />  <!-- used to get demographics data
and updated formula for activity recognition -->
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" /> <!-- used to restart
the SDK activity upon device reboot -->
<uses-permission android:required="false"
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />  <!-- used only for internal debugging
(backup of the sdk) -->
```

Demographics permissions:

```
<!-- Permissions here are not mandatory, but each permission increases the accuracy of the demographics
data -->
<uses-permission android:required="false" android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:required="false" android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:required="false" android:name="android.permission.READ_PROFILE" />
<uses-permission android:required="false" android:name="android.permission.READ_CONTACTS" />
<uses-permission android:required="false" android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:required="false" android:name="android.permission.READ_CALL_LOG" />
<uses-permission android:required="false" android:name="android.permission.READ_SMS" />
<uses-permission android:required="false"
android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS" />
```

Activity Recognition:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" /> <!-- Used to get non
accurate location -->
<uses-permission android:required="false"
android:name="android.permission.ACCESS_FINE_LOCATION"  /> <!-- Used to get accurate location to
improve recognition -->
```

Geo fencing and context awareness

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" /> <!-- Used to get non
accurate location -->
<uses-permission android:required="false"
android:name="android.permission.ACCESS_FINE_LOCATION"  /> <!-- Used to get accurate location to
improve recognition -->

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" /> <!-- Used to get
network state -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" /> <!-- Used to get non
accurate location -->
```

2. Declare the SDK service

```
        <service android:name="com.sensiya.brainssdk.BrainsService"/>
```

3. Declare the receiver used by the SDK

```
<receiver android:name="com.sensiya.brainssdk.BrainsBroadcastReceiver">
        <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
                <action android:name="android.net.wifi.STATE_CHANGE"/>
                <action android:name="android.location.PROVIDERS_CHANGED"/>
        </intent-filter>
</receiver>
```

4. Declare your own receiver with the events you would like to receive from the SDK:

```
<receiver android:name=".YourBroadcastReceiverClassNameHere">
          <intent-filter>
                    <action android:name="brainssdk.intent.action.LEAVING_HOME"/>
                    <action android:name="brainssdk.intent.action.ENTERING_HOME"/>
                    <action android:name="brainssdk.intent.action.LEAVING_WORK"/>
                    <action android:name="brainssdk.intent.action.ENTERING_WORK"/>
                    <action android:name="brainssdk.intent.action.HEADING_TO_WORK"/>
                    <action android:name="brainssdk.intent.action.HEADING_HOME"/>
                    <action android:name="brainssdk.intent.action.WENT_TO_SLEEP"/>
                    <action android:name="brainssdk.intent.action.WOKE_UP"/>
                    <action android:name="brainssdk.intent.action.GEO_FENCE"/>
                    <action android:name="brainssdk.intent.action.SUDDEN_LIGHTS_OFF"/>
                    <action android:name="brainssdk.intent.action.USER_ACTIVITY_CHANGED"/>
                    <action android:name="brainssdk.intent.action.USER_BROWSING"/>
          </intent-filter>
</receiver>
```

5. Add the unique application key that was generated for your app in the web console:

```
<meta-data android:name ="SENSIYA_APP_KEY" android:value="PasteYourApplicationKeyHere"/
```

Running the SDK

The SENSIYA SDK is started by calling its *start* API in the onCreate method of your custom Application, main Activity or Service, depends on the structure of your application. Once you get *onConnected* response to the callback your SENSIYA SDK is ready.

```
            BrainsAPI.start(context, new BrainsAPICallback(){
                 @Override
                public void onConnected() {
                  Log.d(TAG, "SDK is connected. Start using it from here.");
                }

                @Override
                public void onError(Error error) {
                  Log.e(TAG, "Error starting the SDK. code: " + error.getCode() + " message: " +
error.getMessage());
                }

                @Override
                public void onDisconnected() {
                  Log.d(TAG, "SDK is disconnected");
                }

                @Override
                public void onUserDataReady(User user) {
                  Log.d(TAG, "User data ready");
                }

            });
```

Getting asynchronous updates

To get asynchronous notifications from the SDK you need to create a broadcast receiver and subscribe for actions you would like to be notified about.

For example, subscribe for the “brainssdk.intent.action.LEAVING_WORK” intent action to get notified every time the user is leaving his work place.

Calling synchronous API

You can also get a specific user data or SDK information using the direct API. For example:

- Get the SDK version:

```
String version = BrainsAPI.getVersion();
```

- Get user object and its gender:

```
User userGender = BrainsAPI.getUser();

int gender = user.getGender(); //1 – male, 2 – female, 0 – unknown

float genderConfidence = userGender.getConfidence(); //confidence level from 0 to 1
```

- Get user's last known activity:

```
Int activity = BrainsAPI.getLastKnownActivity(); //one of the BrainsIntent activity types
```

GeoFencing

Geofencing is a location-based service that sends messages when users.

Sensiya's SDK makes it easy to define and set custom geo fences for your users. You can define geo fences using the following APIs:

- Define simple geo fence with custom radius:

```
BrainsAPI.GeoFencing.add(geoFenceId, latitude, longitude, radius);
```

- Define simple geo fence with custom radius and expiration:

```
BrainsAPI.GeoFencing.add(geoFenceId, latitude, longitude, radius, expiration);
```

- Define simple geo fence with custom radius, expiration and loitering time to indicate if the user is dwelling :

```
BrainsAPI.GeoFencing.add(geoFenceId, latitude, longitude, radius, loitering);
```

After receiving the geo fence notification on one of the user actions: entered, exited or dwelling by receiving the intent action "brainssdk.intent.action.GEO_FENCE" to your broadcast receiver you will be able to:

- Get geo fences that triggered the event :

```
List<Geofence> triggeringGeoFences = BrainsAPI.GeoFencing.getTriggering(intent);
```

- Check the transition of the geofence:

```
int geoFenceAction = BrainsAPI.GeoFencing.getGeofenceTransition(intent);
```

Proguard

If you are using proguard for your application obfuscation, please add these following lines to your proguard file:

```
-keep public class com.sensiya.brainssdk.BrainsService
-keep public class com.sensiya.brainssdk.BrainsBroadcastReceiver
-keep public enum com.sensiya.brainssdk.api.BrainsAPICallback$** {
            **[] $VALUES;
   public *;
}

-keep public class com.sensiya.brainssdk.api.** {
   <fields>;
   <methods>;
}

-keepnames class * implements com.sensiya.brainssdk.persistence.Synchronizable

-keep public class com.wtools.geofence.Geofence

-keep class com.wtools.uinfra.thirdparty.activeandroid.** {
   <fields>;
   <methods>;
}
```

Questions? Contact us.

We're available 24/7. Email support@sensiya.com for a prompt reply.