On Extending the Sensing of Privacy-Aware Online Social Networks

Asma Wasfi
College of Engineering, UAE University

Qurban Memon
EE Department, UAE University, Al-Ain, 15551, UAE
Email: qurban_memon@uaeu.ac.ae

***Abstract*—Wireless sensor networks (WSNs) are mature enough for widespread adoption. WSNs could be made more attractive to end users by integrating them with online social networks (OSNs). This includes developing novel applications and interaction paradigms between WSNs and OSNs. In this paper, an effort is made by extending the prevalent sensing abilities of online social networks by injecting more sensing features and then processing and clustering some of this information for fruitful use using a locally developed application. Data privacy is enforced using data levels and user roles. Firstly, a conceptual framework is presented followed by local development of its components and services. The application is developed in a typical environment such as Android.**

***Index Terms*—social networks, privacy, role based access**

I. Introduction

Nowadays sensors and social networks can fruitfully interface, from sensors providing contextual information in context-aware and personalized social applications, to using social networks as "storage infrastructures" for sensor information. The integration of sensor networks with social networks leads to applications that can sense the context of a user in better ways and thus provide more personalized and detailed solutions. Social networks have gained popularity recently with the advent of sites such as MySpace, Friendster, Facebook, etc. These networks are a source of data as users populate their sites with personal information. To better understand how online social networks can be integrated with physical world, there is a need to understand services provided by current OSN's, which are (i) identity and authorization services, (ii) Application Programming Interfaces (APIs) to access and manipulate the social network graph and publish and receive updates and (iii) container facilities for hosting third party applications [1].

Some examples of integration of social and sensor networks may be exemplified as, for example the Google Latitude application, which shares the collected mobile position data of the user. As a typical use, the proximity alerts may be triggered when two linked users are within geographical proximity of one another. In another application, the City Sense application collects sensor data extracted from fixed sensors, GPS-enabled cell phones and cabs in order to determine where the people are, and then carries this information to clients who subscribe to this information. A number of real-time automotive tracking applications such as 'Automotive Tracking Application' determine the important points of congestion in the city by pooling GPS data from the vehicles in the city. Animal Tracking uses tracking data collected with the use of radio-frequency identifiers [2]. The CenceMe application injects sensing presence into popular social networking applications such as Facebook, MySpace, and IM (Skype, Pidgin) allowing for new levels of "connection" and implicit communication between friends in social networks [3]. The Green GPS is a participatory sensing navigation service that maps fuel consumption on city streets, to allow drivers to find the most fuel-efficient routes for their vehicles between arbitrary end-points [4]. The Microsoft Sensor Map allows for a general framework where users can choose to publish any kind of sensor data. The sensor data published by a user can be their audio or video feed, location information or text which is typed on a keyboard. The goal of the Microsoft SensorMap is to store and index the data in a way that is efficiently searchable. The SensorMap application enables users to index and cache data. The indexing and caching allows users to issue spatio-temporal queries on the shared data [2].

The innovations in World Wide Web [5] and the recent trends in data protection [6], [7] have increased attraction for use of online social networks. However, the related advancements in technology and tools are also complimented by corresponding privacy concerns. The important thing to note is the lack of awareness for potential risks involved when data is being shared online [8]. Specifically, the external entities can mine this data and use it for different purposes like spamming [9], discovering interaction pattern in the enterprise to offer and develop innovative services, identification of the important person in the network, detection of hidden clusters, identifying user sentiments for proactive strategies etc. [10].

Manuscript received April 15, 2014; revised July 31, 2014.

doi: 10.12720/ijeee.3.4.323-330

The purpose of this research is to extend the integration of WSNs and OSNs, to improve social networking in area of healthcare to help people find motivation to exercise, to doctors to see specific information, to accumulate fitness parameters, to track and locate friends and family members to share certain parameters.

The paper is structured as follows. In the next section, a framework is developed that describes what is needed to build such an application, and which social parameters need to be linked, and how the platform addresses privacy. The section three discusses the framework implementation. The development details are discussed in section four along with results. The comparative analysis is carried out in section five, followed by conclusions in section six.

II. Conceptual Framework

The design and implementation of 'AppName' is described as a sensing application that enables members of a typical social network platform to share their location information with their friends in a private manner. The goals of the development are: (i) to foster and extend the integration of WSNs and OSNs; (ii) to improve social networking; (iii) to create healthcare fun and help people find motivation to exercise; and (iv) to track and locate friends and family members and generate clusters based on a criterion. The application is expected to allow new levels of "connection" and implicit communication between contact groups in social networks. The application uses various sensors to acquire relevant data and display it on user device. The user can check-in to one of the displayed places, and this, in turn, is named as a single visit to a location. Thus, the user can get his/her friends location based on their last check-in. The user should also be able to see if any of his/her friends are checked in nearby. For this, the displayed nearby friends need to be clustered to nearby colleagues, family and so on. In addition the application should enable the user to calculate the distance he/she walked, the duration spent during the walk and the walking related burned calories. Social constraints such as privacy area are also addressed in this application. Based on this, a framework is displayed as shown in Fig. 1.

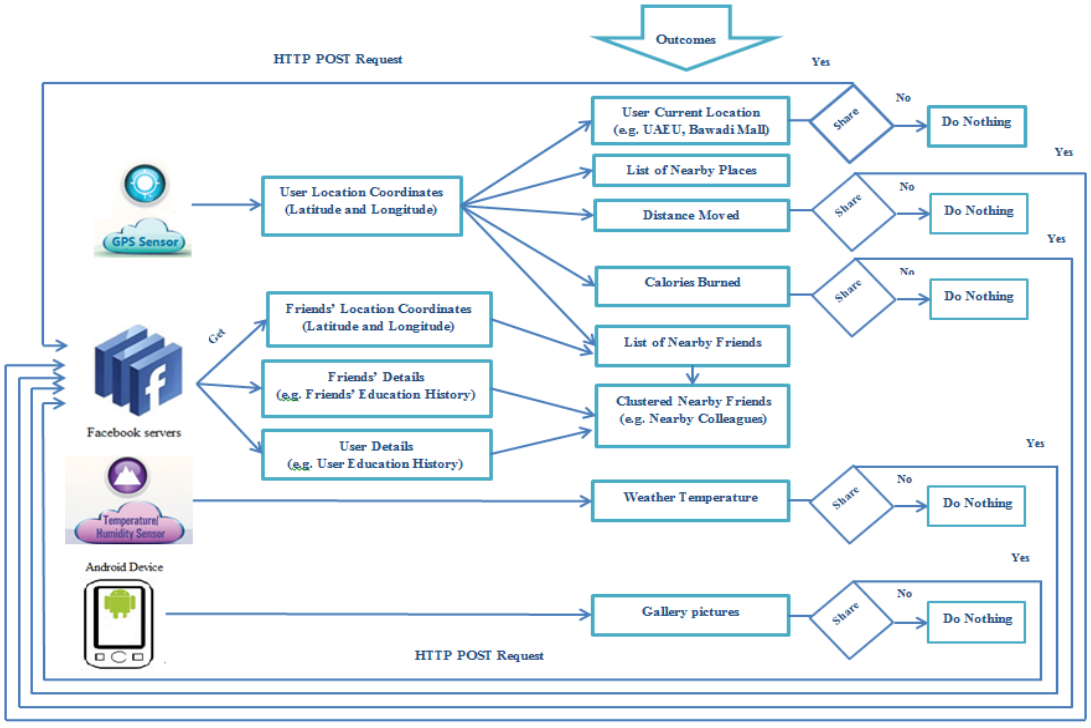

Figure 1. AppName conceptual framework

Based on the framework, the services that meet these goals are detailed in this section. These are stated as follows:

A. Application Platform

There are several popular OSN platforms. The Facebook is the most popular OSN platform today. This application is developed for such a platform and is compatible with Android devices. For development, the Facebook supports different APIs for developers: (i) The Graph API, which is a simple HTTP-based API that gives access to the Facebook social graph, uniformly representing objects in the graph and the connections between them. Most other APIs are based on the Graph API; (ii) The Open Graph API allows applications to tell stories through a structured, strongly typed API; (iii) The Facebook offers a number of dialogs for Facebook Login, posting to a person's timeline or sending requests; (iv) The Facebook Query Language (FQL) enables the developer to use a SQL-style interface to query the data exposed by the Graph API. It provides some advanced features not available in the Graph API such as using the results of one query in another; and (v) The Facebook Public Feed API lets the developer read the stream of public comments as they are posted to Facebook.

B. Application Sensors

The application uses the built-in GPS of the user mobile device to get current location coordinates. It also uses the temperature and humidity sensor to check the weather temperature.

C. Application Services

The services, which can be used using this platform, are:

Show nearby places: This application enables the user to use the built-in GPS of the mobile device to get the current location. It displays a list of nearby places as well.

Public location badge: The user can post location directly on Facebook to increase visibility of information to other users.

Show nearby contacts: The user gets his contact location based on last Check-in. The user can see if any of his/her friends are checked in nearby. The application displays a list of nearby friends, place and the time they checked in.

Cluster nearby contacts: Clustering is important in analysis and exploration of data. This application clusters nearby friends into groups based on colleagues, family and so on.

Tracking distance moved, calories burned and active time: This application tracks distance moved, calories burned and shows active time for the user. The application can also be used for running, cycling, walking and all other distance-based outdoor sports. Once data is shown on network, the user can seek extra encouragement from friends and family to workout. The clinical staff from a healthcare center can also monitor shared clinical data.

Get weather temperature: This application enables the user to use the built-in sensors of a mobile device to check the weather temperature and share with friends.

Share pictures: The application enables image of user's specific injured or monitored body part, once on line, to be seen and examined by a doctor.

D. Application Security and Privacy

At this level, the main focus is how user can control the detail and the accuracy of what other users will be able to access and see. In order to maintain, privacy, the user can turn off this whole or components of this application using the settings option. Another option is to use different levels in data and the user roles based on connectivity. A role represents a group initiated by the user. The idea is that the users in the same cluster with same relationship with the user can get same levels of information. Thus the user assigns relationship to each connection. More relationship provides access to more information. This is depicted in Fig. 2.

It is clear from Fig. 2 that each contact has some level of access to the user data. For example, the contacts 1, 3, and 4 (i.e., friend, colleague, office staff) may only access name of the user, while contact 2 (a healthcare unit staff) may also access address and medical records though all contacts may be part of same social network.

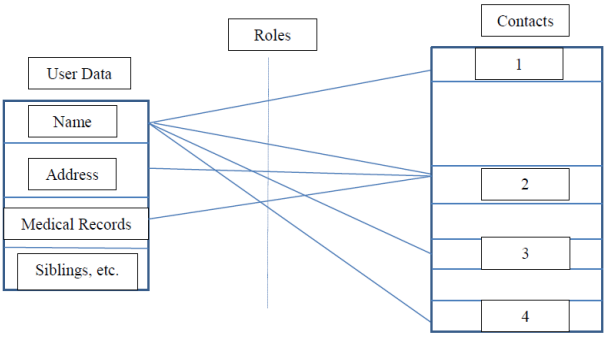

Figure 2. User-Data model

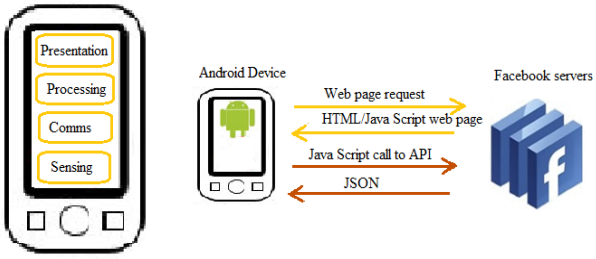

Figure 3. Software architecture of appname

III. Framework Implementation

Based on the framework, the information and process flow inside 'AppName' can be easily visualized, based on which software architecture of 'AppName' is shown in Fig. 3. For the purpose of implementation, the various stages in information and process flow are discussed below.

A. Sensing

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors prove raw data with precision and accuracy. The platform supports three broad categories of sensors: (i) position sensors to measure the physical position of a device. This category includes orientation sensors and magnetometers; (ii) environmental sensors to measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers; and (iii) motion sensors to measure acceleration and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors. In this application, position and environmental sensors are used.

B. Data Acquisition

Data acquisition is the process of gathering information in an automated fashion from analog and digital measurement sources such as sensors. Apart from position and environmental sensors to get position (i.e., latitude and longitude information) and weather information respectively, AppName uses Facebook APIs such as Graph API and Facebook Query Language to extract the user education history, user work history,

friend's last check-in coordinates, friends education history, friends work history and the nearby places.

C. Data Processing

AppName analyses and processes the extracted data to produce meaningful information. After getting the user location coordinates and his/her friends' location coordinates, AppName measures the distance between the user and each one of his friends to produce list of nearby friends. AppName compares the user education-history and other details with friends' education-history to cluster the nearby friends into groups such as colleagues, family and so on. Additionally, AppName uses the acquired location coordinates taken frequently to measure the distance walked, the related duration and the calories burned.

D. Data Sharing

AppName enables the user to share location, the distance walked, burned calories, the weather temperature and pictures. The Graph API updating is done simply with an HTTP POST request to relevant endpoint with the updated parameters. To publish and share new data AppName uses POSTs HTTP requests to appropriate URLs.

E. Presentation

After processing the data, AppName displays a list of nearby places. The user will be able to check in to any of these places by clicking on one of the places. It also has nearby friends icon by pressing on this icon, the user will get a list of his/her nearby friends based on their last check-in. AppName organizes nearby friends into groups of colleagues, work friends, family and others. Moreover, it displays the distance the user walked, total calories burned and the weather temperature.

IV. Development Results

In this section, the main focus is how all the functions, components and services mentioned in section 3 are implemented. Each of these is discussed below.

Login: This has been implemented in an easy way for people to log in to the application such as AppName. AppName uses iOS, Android, JavaScript and Facebook SDKs to speed up the process and build login systems quickly. For secure authorization Facebook uses the OAuth2.0 open protocol for confirming a person's identity and giving them control over right of access to their information.

Permissions: The permissions enable developers to request access to information about someone using their application. AppName asks for the following permissions: offline access, publish stream, publish check-ins, photo upload, user status, user education history, user work history, friends' status, friends' education history and friends' work history. To gain access, AppName requests the permissions transparently through the Login dialog. To maintain information security, almost all API calls at Facebook need to have an access token passed in the parameters of the request.

Show nearby places: The following steps outline how to get user's current location, display a list of nearby places and check in to one of these places with the Facebook SDK for Android.

1) **Set up the Place Picker Item:** This step includes defining a Base List Element class to represent an item in the list. This class contains member variables that define the user interface (UI) as well as methods that are sub-classed to implement the behavior around click events, storing and restoring state info, as well as notifying observers about data changes.
2) **Show the Places Picker:** The Facebook SDK provides a Place Picker Fragment class that displays a list of nearby places. This fragment is hosted in the PickerActivity class. This activity launches when the user clicks on a place in the list. The Place Picker Fragment is used if the incoming intent data matches a pre-defined place picker Uri. Before loading the data, the Place Picker Fragment is configured to specify search criteria like radius, query and maximum results to return.
3) **Display the Selected Place:** In this step, the place will be displayed when the place picker activity is dismissed.

Public location badge: The following steps outline how to publish a story to share the user location with friends. A request will be published by using Request(Session session, String graphPath, Bundle parameters, HttpMethod httpMethod). Graph Object and Open Graph Action interfaces are used to set up a Graph object representation of the POST parameters. Facebook SDK is used to publish the user location by performing the following steps:

1) Construct a new Request for currently active session that is an HTTP POST to the me/checkins Graph API path.
2) Set a Graph Object for the Request instance. The Graph Object represents location parameters, like the selected place ID, message and location coordinates.
3) For best practices, the user is asked for publish_actions write permission in context, when the app is about to publish the user location.

The code shown below publishes the user location to his/her timeline and on the user friends' news feeds.

```
publicvoid onClick(DialogInterface dialog, int which) {
Bundle params = new Bundle();
params.putString("place", placeID);
params.putString("message", message);
params.putString("coordinates", location.toString());
Utility.mAsyncRunner.request("me/checkins", params,
"POST", new placesCheckInListener(), null); }
```

Show nearby contacts: To show nearby friends, Facebook Query Language (FQL) is used. FQL enables to use a SQL-style interface to query the data exposed by the Graph API. Below, the steps are described that show nearby friends.

1) Issue a HTTP GET request to /fql?q=query where query is a JSON-encoded dictionary of

queries. The following code uses FQL to get the friends details and location according to their last check-in:

```
Bundle params = new Bundle();
params.putString("method", "fql.query");
params.putString("query", "SELECT
author_uid,timestamp,coords,checkin_id FROM checkin
WHERE author_uid IN (SELECT uid2 FROM friend
WHERE uid1 = me()) ");
String response = Utility.mFacebook.request(params);
response = "{\"data\":" + response + "}";
```

2) Store the friends' details and locations from JSONObject into the following arrays latitude, longitude, author_uid_array, timestamp and checkin_id. After that, the distance between the user and each one of his/her friends is calculated and stored in distances array.

The following code uses the response for the FQL query from the previous step and extracts the friends' details and locations from the response, and then it stores them in arrays. After that the distance is calculated and stored in distances array.

```
JSONObject json = Util.parseJson(response);
JSONArray data = json.getJSONArray("data");
JSONObject coords;
Long author_uid=(long)0;
for (int i = 0, size = data.length(); i < size; i++){
JSONObject friend = data.getJSONObject(i);
if(author_uid!=friend.getLong("author_uid"))
{
coords =
data.getJSONObject( i ).getJSONObject("coords");
latitude[counter] = coords.getDouble("latitude");
longitude[counter] = coords.getDouble("longitude");
author_uid = friend.getLong("author_uid");
author_uid_array[counter]=friend.getLong("author_uid");
timestamp[counter] = friend.getString("timestamp");
checkin_id[counter]=friend.getLong("checkin_id");
loc. distanceBetween (loc.getLatitude(),
loc.getLongitude(), latitude[counter], longitude[counter],
results);
distances[counter]=results[0];
counter++;}}
```

3) Find out nearby friends by comparing distance between the user and each one of his/her friends with a predefined distance, then store nearby friends' name in an array.

The following code uses the distances array from the previous step to compare the distance between the user and each one of his/her friends with a predefined distance-threshold in order to determine nearby friends. Additionally, the following code uses FQL to get nearby friends academics history details to be used in clustering.

```
for ( int i = 0, size = distances.length; i < size
&&distances[i]!=0 ; i++ ){
if( distances[i]<(float)11500)
{
neededIndex[counter3]=i;
Nearby_friend_id[counter3]= author_uid_array[i];
counter3++;
}}
for (int i = 0, size = counter3; i < size ; i++){
if(i!= size-1)
{
query+= "uid="+Nearby_friend_id[i]+" or ";
}
else
{
query+= "uid="+Nearby_friend_id[i];
}}
Bundle params2 = new Bundle();
params2.putString("method", "fql.query");
params2.putString("query", "SELECT uid, name,
education FROM user WHERE "+query);
String response2 = Utility.mFacebook.request(params2);
response2 = "{\"data\":" + response2 + "}";
JSONObject json2 = Util.parseJson(response2);
data2 = json2.getJSONArray("data");
for (int i = 0, size2 = data2.length(); i <size2; i++){
JSONObject friend2 = data2.getJSONObject(i);
Nearby_friend_Name[i]= friend2.getString("name");}
```

4) Display nearby contact names, their exact location and when they checked in. For example: Dr. Noor Checked in Al-Ain Hospital at 2014-02-10 T09:16

In the following code, when user presses nearby friends' button, a list of nearby friends is displayed with their location and when they checked in.

```
Button mGetNearbyFriends = (Button)
findViewById(R.id.get_nearby_friends);
mGetNearbyFriends.setOnClickListener(new
View.OnClickListener() {
publicvoid onClick(View v) {
try{
TextView friends_Locations = (TextView)
findViewById(R.id.friends_Locations);
friends_Locations.setText("");
String jsonUser=null;
for ( int i = 0, size = counter3; i < size ; i++ ){
jsonUser=
Utility.mFacebook.request(""+checkin_id[neededIndex[i
]]);
obj = Util.parseJson(jsonUser);
      placeName[i]=obj.optJSONObject("place").getStrin
g("name");
created_time[i]=obj.getString("created_time");
friends_Locations.append(Nearby_friend_Name[i] +"
checked in "+placeName[i] +" at "+
created_time[i]+"\n");} }
catch (MalformedURLException e) {
e.printStackTrace();}
catch (IOException e) {
e.printStackTrace();}
catch (FacebookError e) {
e.printStackTrace();}
catch (JSONException e) {
e.printStackTrace();}
}});
```

Cluster nearby contacts: The following steps show, how clustering can be used to group nearby friends.

1) Issue a HTTP GET request, shown below, to /fql?q=query to get user academics history:
2) Issue a HTTP GET request, shown below, to /fql?q=query to get nearby colleagues after placing the user education history in the FQL query and display the nearby colleagues for the user.

The code shown in Appendix (a) shows how to cluster contacts.

Tracking distance moved, calories burned and active time: The following steps show, how tracking distance moved, calories burned and active time is calculated.

1) AppName tracks the user moved distance by capturing user location coordinates periodically, calculating the distance between each two locations and summing the distances from the time the user presses start button till the time the user presses stop button.

The following code stores the user location coordinates periodically till stop button is pressed.

```
publicvoid onLocationChanged(Location loc) {
dialog.dismiss();
if (loc != null) {
try {
location.put("latitude", new Double(loc.getLatitude()));
location.put("longitude", new
Double(loc.getLongitude()));
} catch (JSONException e) {
}
showToast("Location acquired: " +
String.valueOf(loc.getLatitude()) + " "
+ String.valueOf(loc.getLongitude()));
lm.removeUpdates(this);
if(counter==0)
{
fetchPlaces();
latitude[counter]=loc.getLatitude();
longitude[counter]=loc.getLongitude();
}
elseif (counter==1)
{
latitude[counter]=loc.getLatitude();
longitude[counter]=loc.getLongitude();
}
else
{
latitude[counter]=loc.getLatitude();
longitude[counter]=loc.getLongitude();
loc.distanceBetween (latitude[counter-1],
longitude[counter-1],latitude[counter], longitude[counter],
results);
distances[counter]=results[0];
TotalDistance+=distances[counter];}}
```

AppName will request the user to enter his/her weight in kilograms in order to calculate the walking burned calories. AppName uses the below equation to calculate the rate of calories burned per pound of body weight [11].

Rate per Pound (Cal/lb-min) = $A+BV+CV^2+KDV^3$ where:

V=Walking Speed (mph) – Limited to a minimum of 1 mph and a maximum of 5 mph

A= 0.0195

B= - 0.00436

C= 0.00245

D= $[0.000801(W/154)^{0.425}]/W$

W=Weight (lbs)

K= 0 or 1 (0=Treadmill; 1=Outdoors)

The code, shown below, uses the above equation to calculate the walking burned calories. When the user presses stop button, AppName displays the distance walked, the duration spent during the walk and the walking burned calories.

```
privatevoid getDistanceAndCalories() {
TextView DistanceMoved = (TextView)
findViewById(R.id.distance_moved);
TextView Activetime = (TextView)
findViewById(R.id.active_time);
TextView WalkingBurnedCalories = (TextView)
findViewById(R.id.burned_calories);
EditText Weight = (EditText)
findViewById(R.id.weight);
DistanceMoved.setText("Total Distance you walked : "+
TotalDistance );
activeTime= (counter-1)*30/60; //minutes
Activetime.setText("Active time : "+ activeTime +
" minutes \n");
activeTimeInHours=activeTime/60;
totalDistanceInMiles=TotalDistance/(float)1609.344;
A=(float) 0.0195;
B= (float)-0.00436;
C=(float)0.00245;
K=1;
weightInKgs=(float)Double.parseDouble(Weight.getTex
t().toString());
weightInPounds=weightInKgs*(float)2.20462;
D= (float)((Math.pow(weightInPounds/145,
0.425)*0.000801)/weightInPounds);
V=(totalDistanceInMiles/activeTimeInHours); //Walking
Speed (mph) – Limited to a minimum of 1 mph and a
maximum of 5 mph
ratePerPound=
(float)(A+(B*V)+(C*Math.pow(V,2))+(K*D*Math.pow(
V,3)));
walkingBurnedCalories= ratePerPound*weightInPounds;
WalkingBurnedCalories.setText ("Walking burned
calories : "+ walkingBurnedCalories + " calories \n");
}
```

Get weather temperature: The following steps show, how weather information is collected.

1) To acquire data from temperature and humidity sensor, an instance of the SensorManager class is created. This instance is used to get the physical sensor.
2) Register a sensor listener in the onResume() method, and start handling incoming sensor data in the onSensorChanged() callback method.

3) Implement onAccuracyChanged() and onSensorChanged() callback methods. The sensor is unregistered when an activity pauses to prevent the sensor form continually sensing data and draining the battery.

The code shown in Appendix (b) uses the temperature and humidity sensor to get the weather, room temperature and then display it for the user.

Share pictures: In order to share pictures, the following code issues a HTTP POST request to share a photo with friends or healthcare professionals.

```
params.putString("caption", "FbAPIs Sample App photo upload");
Utility.mAsyncRunner.request("me/photos", params, "POST", new PhotoUploadListener(), null);
```

V. Comparative Analysis

A number of recent applications designed in the context of integrating wireless sensor networks with online social networks can be examined for the purpose of comparison. The existing platform applications such as Google Latitude shares the collected mobile position data of the user among different users, and then it generate proximity alerts when two linked users are within geographical proximity of one another. These applications are limited and target specific service only. In this work, 'AppName' not only uses the built-in sensors of the user mobile device to get current location coordinates, view user current location, share location, show nearby places, and show nearby friends, but also provides more services such as clustering nearby friends, tracking distance moved, calories burned and active time, getting weather temperature and sharing pictures. The data privacy is enforced by two options. The first option is available on all available social network platforms, while the second option of using roles versus data levels model.

VI. Conclusion

In this paper, the framework and implementation of 'AppName' was presented. A number of sensors were used to build a sensing application that enables members of social network to share their information with their contacts in a private manner. The user can see if any of his/her contacts are checked-in nearby. It was shown that contacts can be clustered based on colleagues, family or any other criterion. The clustering process takes place in the user device. 'AppName' is a great tool for fitness, weight loss, calorie counting, etc., and can facilitate quicker monitoring of, for example, sugar levels of the user through social network by concerned healthcare units. Social constraints such as privacy were addressed in two ways: simple like turn application ON or OFF; and the other by privacy aware data connectivity based on user roles.

Appendix

a. Clustering Nearby Contacts

```
Bundle params3 = new Bundle();
params3.putString("method", "fql.query");
params3.putString("query", "SELECT name, education_history FROM user WHERE uid =me()");
String response3 = Utility.mFacebook.request(params3);
response3 = "{\"data\":" + response3 + "}";
JSONObject json3 = Util.parseJson(response3);
String MyCollege=json3.getJSONArray( "data" ).getJSONObject(0).getJSONArray("education_history").getJSONObject(0).getString("name");
params3.putString("query", "SELECT name,uid,education_history FROM user WHERE ("+query+ ") AND ""+ MyCollege +"" IN education_history");
response3 = Utility.mFacebook.request(params3);
response3 = "{\"data\":" + response3 + "}";
json3 = Util.parseJson( response3 );
data3=json3.getJSONArray("data");
for ( int i = 0, size2 = data3.length(); i <size2; i++)
{
Nearby_college_id[i]=data3.getJSONObject(i).getLong("uid");
}
int counter4=0;
for ( int i = 0, size = counter3; i < size; i++){
      if(Nearby_friend_id[i].equals(Nearby_college_id[counter4]))
{
friends_Locations.append( Nearby_friend_Name[i]+" checked in "+placeName[i] +" at "+ created_time[i]+"\n" );
counter4++; } }
}
catch (MalformedURLException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
catch (FacebookError e) {
e.printStackTrace();
} catch (JSONException e) {
e.printStackTrace();
} } }); }
```

b. Weather Information

```
private SensorManager mSensorManager;
private Sensor mTemperature;
publicclass Temperature extends Activity implements SensorEventListener {
private SensorManager mSensorManager;
private Sensor mTemperature;
private TextView displayTemperature;
@Override
publicfinalvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
// Get an instance of the sensor service, and use that to get an instance of
```

```
// a particular sensor.
mSensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
mTemperature =
mSensorManager.getDefaultSensor(Sensor.TYPE_AMBI
ENT_TEMPERATURE);
displayTemperature = (TextView)
findViewById(R.id.friends_Locations);
  }
@Override
publicfinalvoid onAccuracyChanged(Sensor sensor, int
accuracy) {
  }
@Override
publicfinalvoid onSensorChanged(SensorEvent event) {
float Temperature = event.values[0];
displayTemperature.setText("Temperature : "+
Temperature + " C");
  }
@Override
protectedvoid onResume() {
// Register a listener for the sensor.
super.onResume();
mSensorManager.registerListener(this, mTemperature,
SensorManager.SENSOR_DELAY_NORMAL);
  }
@Override
protectedvoid onPause() {
// Be sure to unregister the sensor when the activity
pauses.
super.onPause();
mSensorManager.unregisterListener(this);
  }
}
```

REFERENCES

[1] M. Blackstock, R. Lea, and A. Friday, "Uniting online social networks with places and things," in *Proc. the Second International Workshop on Web of Things*, New York, NY, USA, 2011.

[2] C. Aggarwal and T. Abdelzaher, "Integrating sensors and social networks," in *Social Network Data Analytics*, Springer, 2011, ch. 14, pp. 379-412.

[3] E. Miluzzo, N. D. Lane, S. B. Eisenman, and A. T. Campbell, "CenceMe: Injecting sensing presence into social network applications using mobile phones," in *Proc.* 2nd *European Conference on Smart Sensing and Context*, Springer, Oct. 2007, pp. 1-28.

[4] R. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. Abdelzaher, "GreenGPS: A participatory sensing fuel-efficient maps application," in *Proc. Mobisys*, San Francisco, CA, Jun. 2010, pp. 151-164.

[5] Q. Memon and S. A. Khoja, "Semantic web for program administration," *International Journal of Emerging Technologies in Learning*, vol. 5, no. 4, 2010.

[6] A. Moravejosharieh, H. Modares, and R. Salleh, "Overview of mobile IPv6 security," in *Proc. 3rd International Conference on Intelligent Systems, Modelling and Simulation*, 2012, pp. 584-587.

[7] Q. Memon, "A new approach to video security over networks," *International Journal of Computer Applications in Technology*, vol. 25, pp. 72-83, 2006.

[8] Y. Altshuler, Y. Elovici, N. Aharony, and A. Pentland, *Security and Privacy in Social Networks*, Springer, 2012.

[9] M. Huber, M. Mulazzani, E. Weippl, G. Kitzler, and S. Goluch, "Exploiting social networking sites for spam," in *Proc. 17th ACM Conference on Computer and Communications Security*, NY, USA, 2010, pp. 693-695.

[10] B. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-Preserving data publishing: A survey of recent developments," *ACM Computing Surveys*, vol. 42, no. 4, 2010.

[11] K. M. Karkanen, "Walking/Running heart rate monitoring system," U.S. Patent 6013009, Jan. 11, 2000.