TNC Transactions on Networks and Communications
SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
VOLUME 3, ISSUE 2
ISSN: 2054 - 7420

Investigating Privacy Preserving Healthcare Social Network

Qurban A Memon, Asma Fayes and Mustafa
College of Engineering, UAE University, Al-Ain, UAE
qurban.memon@uaeu.ac.ae

ABSTRACT

As health information is valuable, thieves will continue to steal it. Likewise, insufficiently trained employees at organizations or even individuals that pay less attention to creating a privacy-aware culture will suffer data losses when unprotected devices are lost, stolen or sniffed. In this work, sensors are exploited to collect and generate data to be processed, clustered and shared via locally developed application to improve social networking in context of privacy preserving healthcare. The solution is developed for Android operating system. It provides the user with a set of network related social services. In addition to (i) showing specific places (ii) sharing the user location; (iii) showing nearby friends; (iv) getting and sharing weather temperature, it clusters nearby friends; calculates and shares distance moved, calories burned and active time; calculates tracking and shares the user heart rate,. Data privacy model is presented using data session levels and user roles to ensure privacy within healthcare social network.

Index Terms—Social Networks, Privacy, Role based access, Healthcare Network

1 Introduction

Wireless sensor networks (WSNs) have become mature enough for widespread adoption. However, most of the related use cases are highly application-specific, and do not usually affect everyday life. WSNs could be made more appealing to end users by leveraging online social networks (OSNs). This includes developing novel application scenarios and interaction paradigms between WSNs and OSNs. 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, publish and receive updates and (iii) container facilities for hosting third party applications [1].

There are a couple of important drivers for integrating sensor and social networks. One driver for integrating sensors and social networks is to allow the actors in the social network to both publish their data and subscribe to each other's data either directly or indirectly after discovery of useful information from such data. The idea is that such collaborative sharing on a social network can increase real-time awareness of different users about each other. A second driver for integrating sensors and social networks is to better understand or measure the aggregate behavior of self-

DOI: 10.14738/tnc.32.1116
Publication Date: 17th April, 2015
URL: http://dx.doi.org/10.14738/tnc.32.1116

selected communities or the external environment in which these communities function. Examples may include understanding traffic conditions in a city, understanding environmental pollution levels, or measuring obesity trends [2].

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. As another example, 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 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 SensorMap allows for a general framework where users can choose to publish any kind of sensor data. 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].

Sensors provide numerous research challenges from the perspective of analysis. Since the collected data typically contains sensitive personal data (e.g., location data), it is extremely important to use privacy-sensitive techniques to perform the analysis. Another challenge is that the volume of data collected can be very large. For example, in a mobile application, one may track the location information of millions of users simultaneously. The innovations in World Wide Web [5, 6] and the recent trends in data protection [7, 8] 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 [9]. Specifically, the external entities can mine this data and use it for different purposes like spamming [10], 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. [11].

In context of healthcare, currently fitness bands, electronic health records, health information exchanges, connected devices, tools, and sites containing medical data keeps growing. Attackers are also becoming more sophisticated. Cybercriminals are seeking more information than ever about their victims to sell. The concern is that richer personal identity data of individual users, consisting regional and geographic data, personal information and behavior are expected to be traded in the same manner that stolen credit cards are today. Thus, the task of securing health information is becoming more challenging for even the best-prepared organizations.

The purpose of this research is to improve social networking in area of healthcare by sharing specific information (for example accumulated fitness indicators) online with doctor and/or parents in a privacy aware manner.

The paper is structured as follows. In the next section, a model is proposed 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 model 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.

2 Proposed Model

A convenient design gives rise to other challenges that need to be addressed in order to enable development of successful mobile applications which meet user needs. Before describing the model, it seems necessary to highlight challenges in the environment:

- Mobile limitations: Mobile phones have very good computational efficiency, but they offer limited programming and resource usage control.
- Energy limitations: Application developers on mobile phone platforms need to be aware of power consumption when developing an application that depends on using radio interfaces such as GPS.
- Privacy issues: The collected data such as location data contains sensitive and personal data. It is highly important to apply a privacy model to maintain data security.
- Data management: The data collected is huge, so it is extremely important to efficiently process the huge amounts of data.
- Simplicity: The application should be easy to understand and thus target non-expert.
- Determining user location: GPS only works outdoors, and it quickly consumes phone battery power. Android's Network Location Provider helps to determine user location using Wi-Fi signals and cell tower. This strategy helps to provide the location details indoors and outdoors and uses less battery power. For effective use of battery, the application needs to deploy both strategies.

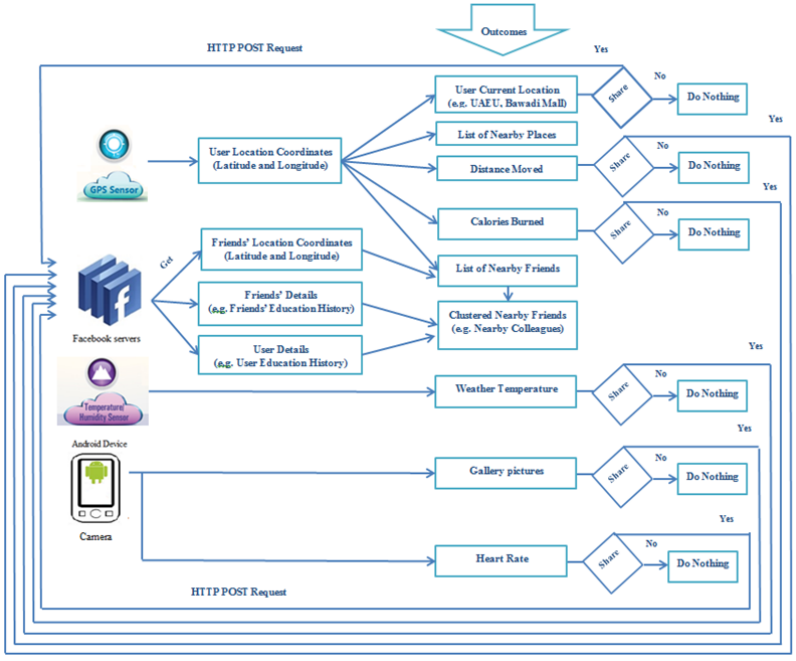

Figure 1: Conceptual Model

In the proposed model, the objective is to extend online social networks by injecting more sensing features like hear rate monitoring, sugar levels etc., and then processing and clustering some of this information for a benefit to a user in an environment like healthcare, using a locally developed application in the mobile device. The proposed model describes an application that enables members of a typical social network platform to share healthcare related features along with their location information with their friends in a private manner. 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 walked, the duration spent during the walk and the walking related burned calories. As a benefit, these services may motivate the user to exercise by competing with friends who exercised more and burned more calories. Moreover the user can check if any of his friends is nearby so he can walk with. The application is expected to encourage users to track their heart rates and share it with their doctor. As a result, this is expected to help the user to track their fitness to a new level. Based on this, a conceptual model is displayed as shown in Figure 1. Based on the model, the platform, components, and services that meet these expectations are detailed as follows:

2.1 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.

2.2 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.

2.3 Application Services:

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

2.3.1 Show nearby Places:

This application enables the user to use the built-in GPS or Androids' network location provider built in the mobile device to get the current location. It displays a list of nearby places as well.

2.3.2 Public Location Badge:

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

2.3.3 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.

2.3.4 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.

2.3.5 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, once connected in privacy mode, can also monitor shared clinical data.

2.3.6 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.

2.3.7 Share pictures:

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

2.4 Application Security and Privacy:

As highlighted in section I, data privacy is a major concern. In order to maintain privacy, the user can turn off this whole or components of this application using the settings option, but this is not the common practice of online users. In order to protect users' shared data, data can be handled in many ways: (i) by bifurcating table into shareable and non-shareable columns i.e. data partitioning, and then play with rights. This will work only in the case when you exactly know which field to be shared and your table entries are fixed (i.e. not dynamic). However for data porting and scaling up the options, one may face problems; (ii) by using MongoDb. MongoDB is a no-SQL database and works on documents rather than entries. It's also hierarchical and supports dynamic data (i.e. table entries cannot be fixed); (iii) (Jugar option) by inserting one column of rights to all your data rows. In the first part of query, check is done to see whether rights are correct, then further processing of query is allowed, otherwise the query is rejected; and (iv) by using role based access control (RBAC) mechanism. In RBAC, permissions are associated with roles, and users are to be made members of appropriate roles [12-13]. This helps to simplify management of permissions. Roles are similar to the group's concept in access control. Role is defined as a set of users on one side and a set of permissions that will be applied to the users on the other side, while groups are defined as a set of users only. The privacy model made as part of the proposed scheme is derived from role based access control (RBAC).

In the proposed model, the roles are generated for various trust levels and contacts are assigned roles based on their relationship with the user. Contacts can be easily reassigned from one role to another. The developed application has predefined role-permission relationships, which makes it simple to assign contacts to the predefined roles. It is difficult, without the new privacy model, to determine what permissions have been assigned to what users. The proposed privacy model helps to perform large-scale authorization management. The basic concept of the proposed privacy model is that the user assigns his contacts to roles, roles have predefined permissions, and contacts acquire permissions by being members of roles. User-role can be many to many, which means that the same user can be assigned to many roles.

The application provides the user with three roles: (i) trusted, (ii) semi-trusted and (iii) un-trusted. The user will set the members of each role according to his/her relationships with his/her contacts. Trusted role has predefined permissions that will enable its members to view and comment on most of the user posts. Semi-trusted will enable its members to view and comment on some of the user posts, while untrusted role members won't be able to view or comment on many of the posts. Moreover, the user can assign relationships to his trusted and semi-trusted contacts. The user can assign his trusted contacts close or not close relationship. Assigning close relationship to contacts will give them the option to view more posts, while not-close relationship won't give them the permission to view more details. Using this privacy model, the users can allow their contacts to share certain level of their information. This model helps in managing different level of information sharing. Contacts will not know what role or relationship the user has assigned them. Such a model is shown in Figure 2.

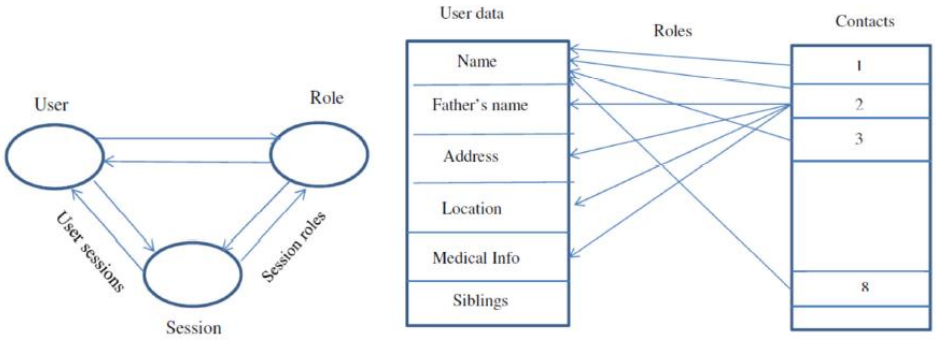

Figure 2: User-Data Model

It is clear from Figure 2 that each of the contacts has some level of access to the user data. For example, the contacts 1, 3, and 8 (i.e., friend, colleague, office staff) may only access name of the user, while contact 2 (a healthcare unit staff or user-parent) may also access address, location and medical records though all contacts may be part of same social network. In order to examine different relationships between entities, consider set of users, roles, objects, operations, collaborative relationships, access levels, and conditions be represented by *U, R, Objs, Oprs, CR, AL, Cond* respectively. Thus, the assignment relations among elements of the privacy-aware model are:

2^R: the power set operations

1. Many to many mapping user-role assignment relation: $URA \subseteq UxR$
2. The set of permissions: $Perms = 2^{(OprsxObjs)}$. This can also be stated as: $Perms = \{(Objs, Oprs) | Objs \in Objs, Oprs \in Oprs\}$
3. The set of sharing and privacy-aware permissions: $PA\_perms = (Perms, CR, Cond, AL)$ This can also be stated as: $PA\_perms = \{(perms, cr, al, cond) | perms \in Perms, cr \in CR, al \in AL, and\ cond \in Cond\}$
4. Many to many mapping permission-role assignment relation: $PRA \subseteq PermsxR$ Many to many mapping sharing and privacy-aware permission-role assignment relation: $Prv - PRA \subseteq PA\_perms\ xR$

3 Model Implementation

Based on the model in section II, the information and process flow inside application can be easily visualized. Based on this, software architecture is shown in Figure 3. For the purpose of implementation, the various stages in information and process flow are discussed below.

3.1 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.

3.2 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, the application 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.

3.3 Data Processing:

The application analyses and processes the extracted data to produce meaningful information. After getting the user location coordinates and his/her friends' location coordinates, the application measures the distance between the user and each one of his friends to produce list of nearby friends. The application 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, the application uses the acquired location coordinates taken frequently to measure the distance walked, the related duration and the calories burned.

3.4 Data Sharing:

The application 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, the application uses POSTs HTTP requests to appropriate URLs.

3.5 Presentation:

After processing the data, the application 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. The application 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.

Figure 3: Software Architecture of application

4 Development Results

Before results are presented, it seems necessary to highlight development environment, briefly described below:

Simulation Environment: The application is developed for Android operating system devices. Android is an open source operating system available to all developers with various expertise levels. Android is a Linux-based operating system created for touchscreen mobile phones and tablets. Android platform allows users to develop, install and use their applications. The application is primarily developed in Java programming language by using the Android software development kit (SDK). The SDK provides the developers with all the tools they need including software libraries, debugger, sample code, emulator, and tutorials. The integrated development environment (IDE) for developed application is Eclipse using the Android Development Tools (ADT) plugin. Facebook SDK for Android was used to integrate this application with Facebook's platform.

In order to start the simulation, prerequisites including Eclipse, Android SDK, Android Developer Tools (ADT) Plugin, and Facebook SDK were installed. Then, Facebook SDK was imported to Eclipse. Every Android app that is developed was signed, as it was required to register each application's key hash with Facebook as a security check for authenticity. After registering on Facebook Developer Site as a developer, application Facebook profile was created and details such as application name, category, and key hash for were entered. After creating application profile, application ID was generated and appeared in the profile. Then application ID was added to project files on Eclipse.

In the following, the main focus is how all the functions, components and services mentioned in section II and III are implemented. The code and a sample view of some of the functions/components are also shown for clarity.

Login: This has been implemented in an easy way for people to log in to the application. The application 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. It 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, the application 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.

a. Set up the Place Picker Item: This step includes defining a BaseListElement 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.

b. Show the Places Picker: The Facebook SDK provides a placePickerFragment 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 PlacePickerFragment is used if the incoming intent data matches a pre-defined place picker Uri. Before loading the data, the

PlacePickerFragment is configured to specify search criteria like radius, query and maximum results to return.

c. 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). GraphObject and OpenGraphAction 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:

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

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.

a. 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 + "}";
```

b. 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++;}}
```

c. 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");}
```

d. 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. The list is shown in Figure 4.

```
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").getString("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();}
}});
```

Figure 4: Nearby friends list

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

Issue a HTTP GET request, shown below, to /fql?q=query to get user academics history:

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.

Tracking Distance Moved, Calories Burned and Active Time: The following steps show, how tracking distance moved, calories burned and active time is calculated.

The application 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 application will request the user to enter his/her weight in kilograms in order to calculate the walking burned calories. It uses the equation as shown below to calculate rate of calories burned per pound of body weight [14].

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, the application displays the distance walked, the duration spent during the walk and the walking burned calories. The Figure 6 shows resulting display on the mobile device.

```
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.getText().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.

- 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.
- Register a sensor listener in the onResume() method, and start handling incoming sensor data in the onSensorChanged() callback method.
- 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 1(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 applications issues a HTTP POST request to share a photo with friends or healthcare professionals.

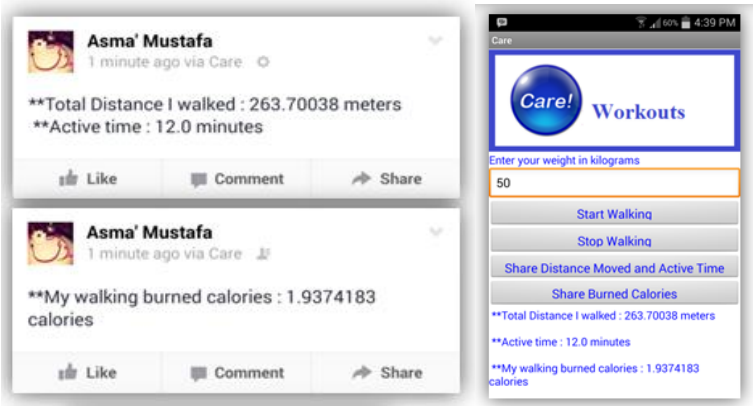

Figure 6: The application workouts

Hear rate monitor: Heart rate monitor uses the camera and its flash to find the user heart rate in beats per minute. The user has to hold the tip of his/her index finger over the camera lens of his/her phone. The application takes between ten to thirty seconds to get accurate heart rate. Heart rate monitoring is based on using the camera with as little focus as possible. When the user puts his/her finger on the camera lens, it won't be focused. The resulted image will be only shades of light and dark RGB. The code looks at single channel (red) and tries to find out when the channel goes from light to dark red.

The application uses the 'PreviewCallback' mechanism to capture the last image from the preview frame. Then the YUV420SP data will be processed to get all the red pixel values. Data smoothing in an integer array is used to figure out the red pixel average value in the image. The heart beat is detected when the average red pixel value in the latest image is greater than the smoothed average.

The application collects data during ten seconds, and then adds the beats per minute to an integer array which will be used to smooth the beats per minute data. As an illustration, the Figure 7 is the resulting display of user heart rate using this application and another medical device at the same time. It turned out that the results are similar.

Implementing Privacy Model: In order to maintain privacy, the application provides the user with three roles: trusted, semi-trusted, and un-trusted as shown in Figure 8. Predefined permissions are assigned to each one of the roles. The members of each role will be assigned by the user from his contact list.

When the user presses Pick Trusted Friends, the following steps will occur:

Issue a HTTP GET request, shown below, to /fql?q=query to get user friends list as shown in Figure 9:

```
String query = "select name, current_location, uid, pic_square from user where uid
in (select uid2 from friend where uid1=me()) order by name";
Bundle params = new Bundle();
params.putString("method", "fql.query");
params.putString("query", query);Utility.mAsyncRunner.request(null, params, new
TrustedFriendsRequestListener());
```

Figure 7: User shared heart rate

Figure 8: Privacy in the application

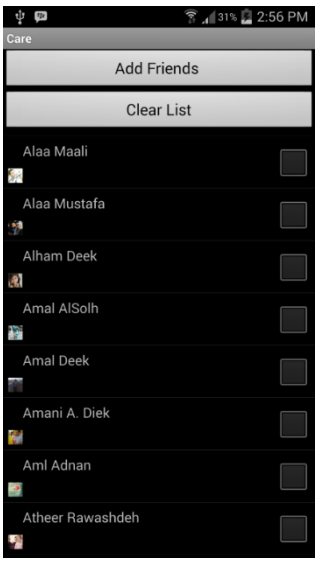

Figure 9: User friend list

When the user press Add Friends, after choosing the ones he wants to select as trusted, the following steps will occur:

- Issue a HTTP POST request to /fql?q=query to create Trusted Friend list.
- Issue a HTTP GET request, shown below, to /fql?q=query to get Trusted Friend list id, then, a POST request to add the selected friends to the list

The same process will be executed when the user presses Pick Semi-Trusted Friends or Pick Un-Trusted Friends.

When the user presses Assign Trused Relationships, the user will get the layout as shown in Figure 10. If he/she pressed 'Close Friends' or 'Un Close Friends', he/she will get a list of the trusted friends only to choose the close or not close friends from them.

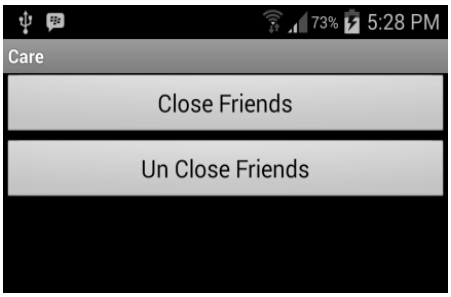

Figure 10: Assigning trusted relationship

When the user presses 'Close Friends', the following steps will occur:

1. Issue a HTTP GET request to /fql?q=query to get user trusted friends only as shown in Figure 11:

```
graph_or_fql = "fql";
String query = "select name, current_location, uid, pic_square from user where uid
in (SELECT uid FROM friendlist_member WHERE flid ="+ trustedFriendListId +") order
by name";
Bundle params = new Bundle();
params.putString("method", "fql.query");
params.putString("query", query);
Utility.mAsyncRunner.request(null, params,
new assignCloseTrustedRequestListener());
```

Adding Friends to close or un-close will be performed the same way as mentioned in the trusted friend list.

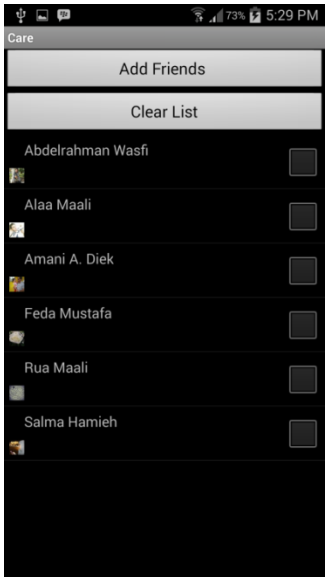

Figure 11: Trusted friend list

Each time a post will be shared such as 'distance moved', 'calories burned', 'active time', 'weather temperature', 'heart rate or pictures', the following steps will be performed to specify who can see and comment on the post and who can't. Below is an example for sharing weather temperature. As we can see trusted and semi trusted friends are allowed to see and comment on the weather post while untrusted friends are denied.

```
private void share() {
String s="";
Bundle params = new Bundle();
params.putString("message", temperature);
JSONObject privacy = new JSONObject();
try {
privacy.put("value", "CUSTOM");
privacy.put("friends", "SOME_FRIENDS");
privacy.put("allow", trustedListId+","+semiTrustedListId);
privacy.put("deny", unTrustedListId);
} catch (JSONException e) {
 // TODO Auto-generated catch block
e.printStackTrace();
}
params.putString("privacy", privacy.toString());
try {
Utility.mFacebook.request("me/feed", params, "POST");
Toast toast;
toast = Toast.makeText(Temperature.this,"Your status has been updated",
Toast.LENGTH_LONG);
toast.show();
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (MalformedURLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();}
```

5 Comparative Analysis

A number of recent applications designed in 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, the application not only uses the built-in sensors of the user mobile device (i.e., 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, hear rate monitoring, getting weather temperature and sharing pictures. This may help to improve healthcare awareness and prompt quicker and safe advice from healthcare professional in social network. In this work, the data privacy is enforced by both options: the first option is available on all available social network platforms – as ON/OFF, while the second option is using user assigned roles versus data levels.

6 Conclusions and Future Work

In this paper, the framework and implementation of an application was presented. A number of sensors were used to build a sensing application that enables members of a 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. The application is expected to be a great tool for fitness, weight loss, calorie counting, etc., and can facilitate quicker monitoring of, for example, heartbeat of the user through social network by concerned healthcare units. Social constraints such as privacy were addressed in two ways: simple feature like turn application ON or OFF; and the other by privacy aware data connectivity based on user roles.

Some future work can be performed in order to extend and improve the built application. Some of its aspects can be improved and more functionality added. Below are some suggestions for further improvement:

- Health support for elders by using sensor information to send alerts if there is abnormal activity. Request for attention can be sent to doctors and nearby friends based on the collected information from sensors such as body position and health measurements.
- Suggesting nearby friends based on common interests.
- Adding more features such as monitoring and tracking user blood pressure. This also includes storing, analyzing and sharing the user blood pressure measurements.

REFERENCES

[1]. M. Blackstock, R. Lea, and A. Friday, "Uniting online social networks with places and things," in Proc. of the Second International Workshop on Web of Things, New York, NY, USA, 2011, pp. 5:1–5:6.

[2]. C. Aggarwal, T.F. Abdelzaher, "Integrating Sensors and Social Networks", Social Network Data Analytics, chapter 14, pp. 379-412, Springer, 2011.

[3]. E. Miluzzo, N. D. Lane, S. B. Eisenman, A. T. Campbell, "CenceMe: Injecting Sensing Presence into Social Network Applications using Mobile Phones", in Proc. of the 2nd European Conference on Smart Sensing and Context, Springer, October, 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. of Mobisys, San Francisco, CA, June 2010, pp. 151-164.

[5]. Q. Memon, S. Khoja, "Academic Program Administration via Semantic Web – A Case Study", Proceedings of International Conference on Electrical, Computer, and Systems Science and Engineering, Dubai, Volume 37, pp. 695-698, 2009.

[6]. Q. Memon, S. A. Khoja, "Semantic Web Approach for Program Assessment", International Journal of Engineering Education, Vol. 25, No. 5, pp. 1020-1028, 2009

[7]. A. Moravejosharieh, H. Modares, R. Salleh, "Overview of Mobile IPv6 Security," in Proc. of 3rd International Conference on Intelligent Systems, Modelling and Simulation, 2012, pp. 584-587, DDI: 10.1109/ISMS.2012.9.

[8]. Q. Memon, "A New Approach to Video Security over Networks", International Journal of Computer Applications in Technology, Vol. 25, 2006, pp. 72-83.

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

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

[11]. B. Fung, K. Wang, R. Chen, and P. S. Yu, “Privacy-Preserving Data Publishing: A Survey of Recent Developments,” ACM Computing Surveys, (CSUR), Vol. 42, No. 4, 2010, doi: 10.1145/1749603.1749605

[12]. Q. Memon, S. A. Khoja, “RFID based Patient Tracking in Regional Collaborative Healthcare”, International Journal of Computer Applications in Technology, Vol. 45, No. 4, 2012, pp. 231–244, doi: 10.1504/IJCAT.2012.051123

[13]. Q. Memon, S Akhtar, AA Aly, “Role management in adhoc networks,” Proceedings of the Spring Simulaiton Multi conference-Vol.1, 2007, pp. 131-137.

[14]. K. M. Karkanen, “Walking/running Heart Rate Monitoring System,” U.S. Patent 6013009, Jan 11, 2000.

