Motivation:

The goal of our project is to provide a mobile LBS (Location-based Services) App for people to organize, manage and participate in social activities conveniently. Our App is motivated to overcome the problems seen in other social mobile Apps by providing several new features and novel perspectives.

Potential Problems of Previous Work:

There already exist some social mobile Apps that have the features to allow users to create activities or register to participate in activities. However, there are several potential problems for those Apps.

The first problem is that most of those Apps do not send notifications (invitations of new activities) to nearby users (possible participants). Thus those Apps require the information of a new created activity to be posted at least a certain period of time (e.g., several days) ahead of the time the activity begins. That period of time is needed to guarantee that there will be enough users seeing the posted information for the activity, registering the activity and planning to attend it in the future. However, this design pattern is not very appropriate for those activities which may start instantly after the organizers decide to held such activities. For example, some pizza and soda are left after a seminar held in the second floor of the CS department finishes. The organizer of the seminar may want to invite college students whoever are willing to come to the building to share the free food. In this case, if the organizer uses those mobile Apps to create a new activity to say that there are some free food in the CS department building, it may be already 4 hours later when some users happen to see the posted information for this activity in the activity list of these App, which maybe too late for any pizza to be left.

The second problem is that those Apps do not keep track of the accurate locations of activities after they start. That may be a big issue if the locations for the activities ongoing are not fixed. Suppose that several days ago you registered to participate in a parade which will begin on 4pm today and last for several hours. The organizer mentions the detail address for the gathering place in the description of the activity. However, what if you are busy with your work today and get late for the parade? What if
when you arrive the gathering place everyone else has already parade to other location and you couldn't find them?

Our Solutions:

To solve the first problem, we introduce a new feature of sending notifications to nearby users whenever a new activity is created. The notifications are location based and they will use the accurate GPS information of the users to decide whether to send notifications to a specific user or not. This feature greatly helps the activity to gather participants quickly and accurately. Thus for the free food example, when the organizer creates the activity about sharing free food, every user within a certain range of the location of the activity (CS building) will receive a notification of this new activity immediately. They no longer need to “wait” for 4 hours to taste the pizza.

For the second problem, we provide a new feature of keeping track of the locations of the activities after they start and let every users who registered to participate those activities know the locations information in time. Thus even you get late for the parade activity, you can still use the App to get to know the immediate location of the activity and go to join the parade at any time you want.

Novel Perspectives:

Location-based Notification: LBS is one of our core concepts for this project. Thus all notification will be entirely based on users’ accurate location. The Server will broadcast the notifications (invitations) of newly created activities to nearby potential users (based on their GPS location) to help activities gather sufficient participant accurately and efficiently. This feature is substantially helpful for those activities that greatly rely on the holding locations.

Adaptive-distance Notification: The server will use Adaptive-distance Notification mechanism when broadcast notifications. Based on the remaining time before the start of the activity and the number of users already register to attend the activity, the server will dynamically adjust (increase or decrease) the distance setting for broadcast to meet the expected number of participants specified by the host in time.

Dynamically Tracking Activity: The server will dynamically track the locations of the activities after they start to let every users who registered to participate those activities get to know the activities’ locations information in time. This feature allows users to find the location of the activity and join the activity conveniently if they get late for it.
Activity-based Message Board: The App provides Activity-based Message Board for users who register for a same activity to communicate and share information with each other. This Message Board is different from the message board in other Apps because it is activity-based instead of User-based. So in our App each individual activity is associate with a separate Message Board to allow its participants to chat and share their same interests conveniently.

Client:
The entire GUI is organized by ActionBar and Fragments. We divide the GUI into three main components. The first tab is NEARBY, which displays a list of nearby activities provided by the server, according to the user’s location. Users can join any of the activities.

As you can see, we use a TextView to show the information about those activities. In fact, I was planning to use a WebView to show all the content. By using WebView we can generate HTML5 codes and javascript code so that we can support images and videos and other fancy features. Besides, we can take advantage of CSS. However, using WebView would introduce too much work here so I give it up. I want to mention it here just because it’s a good idea.

We have also provided a mapView for users to identify the location of the activities.
In the PARTICIPANT tab, the app provides a list of all the activities that the current user has signed up. People are allowed to publish their ideas about the activities and discuss with each other within the app. Users can also cancel their activities here, using the “CANCEL ACTIVITY” button.
In the HOST tab, users are allowed to manage all the activities created by them. Besides, users can create new activities.

All the button icons are provided by Google (downloaded from the Android website). Besides, we have specified the property "ifRoom" for all the buttons (in fact they are all menu item). So when the device screen has no enough space to display the buttons, they will hide in the menu list.

Server:
Our server serves as a role of the bridge between the client and the database. However, some logic is performed on the server instead of the client, as it shall not reveals all the data to the client.

The way that they communicate with server is different, the client use the http protocol to communicate with server, and server uses sql to communicate with the database.

An example of the process is like this:
client->server:
http://130.132.54.179:6789/mm?typeID=2&uID=1

server->database:
SELECT activity.*
FROM user JOIN user_has_activity JOIN activity
ON user.id=user_id and activity_id=activity.id and user.id=1;

(it actually will have more query statement for further advance functionality, but the basic only requires this)

Also by shifting the running burden to the mysql server instead of the weeks-long coded program, the scale and performance is ensured by the long established database software.

The ER-model of our database:

This model is made of main entity like the user and activity, with comments and tags, and also the relations among them.

Most of the query are trivial, but in order to track a moving activity, the server must be able to provide an interface for the query of the current location of an activity. It is not the explicit data on the database, and the server have to configure it out.

Since the only location data we have that are constantly updated is the user's location, we can determine the location by the user.
One easy way to do this is to use the location of the activity’s host. It is just another simple sql query. However, there is actually some possible situation that you can have an activity without the host at the same place for a little while.

Another way is to mining the data of the participants, the location of the activity is usually defined by the location of the participants. However, we only have the data that some people signed up for an activity, but we have no idea that if this person will actually show up at the activity, and only the ones actually goes to the activity shall count as the participants. So the problem shifts to determine who are the ones that in the activity.

To solve this problem, I have to make some assumptions first:
I. At least certain proportion(40% in my code) of people signed up will actually show up.
II. People in the activity will be like a cluster of a certain geographically point.
III. People didn’t goes to the activity will be randomized distributed.

After this assumption, the problem could be solved by a K-means derived method. The sample set will be the people signed up, and instead of solving the cluster problem of K clusters, we will maintain the one cluster with the cluster size of a certain proportion(40%). All the other procedure is the same, decide the sample in this cluster, recalculate the center of the cluster.

This is for 80 clustered data and 120 randomized data simulation

However, the assumption III is not always true. Because it may be common to have a rival activity. The competition will result in multiple clusters, since the same kind of
activity will attract the same group of people. Like the below, two clusters are in the sample set.

To overcome this obstacle, we have make another assumption to make up for the assumption III.
New Assumption:
IV. the host will be closer to his own activity cluster.

In K-means Clustering, the starting point is critical, and in this case, if we set the starting point as the host location, we probably will end up in the neighbourhood of the the host location, and with assumption IV, we can ensure it is the real location.

So, we have to have the assumption I and II and (III or IV) to make this assumption work. However, any of the assumption breaks down, it will not work properly, but this assumption break could serve as a hint, not go.