**Pal Hunter**

**1. Introduction:**

Pal Hunter is an Android-based application that enables 3 major tasks for path finding, location sharing alerts. The first task specifically allows friends to share their path with each other based on actual surface path and also closest transportation network path. The second task allows users to create regions on the map to find or get notified when a friend enters or exists in the regions (regions once created are saved). The app requires registering and finding/adding friends while the danger participatory sensing is shared among everyone.

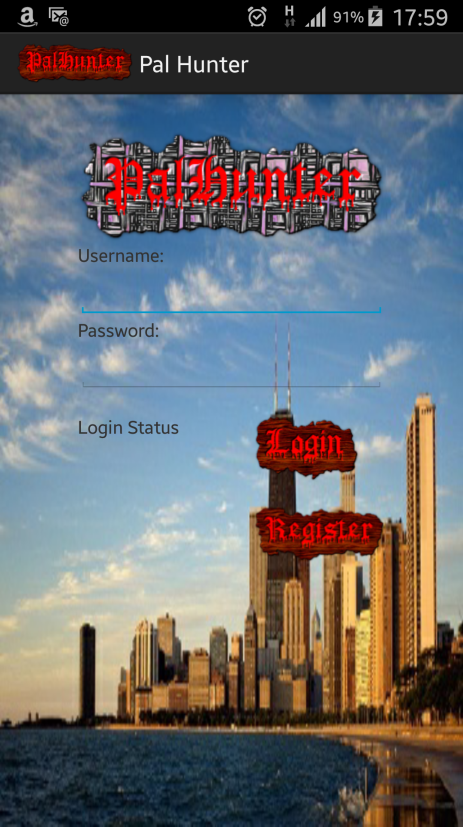


Figure 1. Pal Hunter Homescreen

## **Follow Me Task:**

Suppose you know a favorite place to go for lunch. Because you already know how to go to place you don't need to use a map application. But You decided to go to the place with your friends with a couple of cars. You want other cars to follow your car. By using GPS of the mobile device, other people would be able to see your path and follow that path.

## **Alert Zone Task:**

Basically, the app can be planned to make alert if a friend enters to some specific environment. For example if a friend enters your city, the alert can be activated. Or if a friend enters within 50 meters of you. Or if a friend enters to your school's gym. So we can have settings for different area (school, airport, country, city, bank, hospital) alerts. You can make each alert available based on your desire and your settings. User can set an option that sends message about user's location to desired friends via Facebook or cellular message.

# **2. Basic functions**

## **Client-side**

### **Log in/ Registration:**

· When the application starts, in the first screen, the user is asked to login or register.

### **Find/ Add/ Remove Friends:**

· The user is able to find friends to add them in the friend list.

· The user can add friends to the friend list based on personal interests.

· The user is able to delete friends from the friend list.

### **Sending user location data**

· When application runs, the user location data will be sent over database. User can enable or disable this feature.

· While sending location data roughly every 10 seconds, metadata (location, altitude, time, angle, direction, speed) will be read from the sensors and sent to database.

· The user can press a stop button once they wish to end the recording.

### **Following another User**

· User can select any other user in his friend list to follow

· Once the follow option is selected, the path of followed user that is between 1 minute to 60 minutes time period is queried and shown on the map.

· User can follow one person at a time.

### **Enabling Alert Zone**

· User can specify zones on the map.

· When other users in the friend list enters the specified zones, the user will get notification and other users will get notification or message.

· User can specify many alert zones. Other users wouldn’t see these alert zones.

· Once user enters own alert zone, it will be activated and begin to send notifications. When user leaves the alert zone it will be deactivated and remain in silent mode until user enters the zone again.

## **Server-side**

There will be a web-server running on the server side to process requests of clients and to manipulate the database. The user interface for the server will be a web page with map API to control over the database and client applications.

### **Storing locations:**

· Client applications will send location data to server and server will update the data according to timestamp and client user information.

· While updating the data in the database there will be limited space for each user. The database will be updated in such a way that the last 60 minutes data about the user will be stored.

· Another stored location data is user defined zones. Once user creates a zone, this data will be stored in database. Zones will be activated or deactivated according to user’s location instantly.

· The metadata is processed to generate spatial index information.

### **Query:**

· When user selects to follow other user, client application will query about the path that belongs to followed user. The data that belongs the path will be queried on database and sent to client application.

· Once user enters in specified alert zone, current locations of other users will be queried and sent to client application to check to be processed.

# **3. New ideas/ features**

## **3.1. Following a friend both on surface network and transportation network**

When following a friend’s path, there are two options to see the path. The user can either see the actual path on the surface where the followed friend has passed, or see the network path based on the transportation network, which can used for the time of driving a car. The feature can be used with a key defined in the follow me task.

## **3.2. Danger zone sharing on the following friend route**

While following a friend’s path on the follow me task, the danger zones obtained through crowdsourcing will be visualized in addition to the path. In cases of disaster or attacks where some regions of area is dangerous, the visualization allows for either pursuing a different path or paying more attention and be more cautious.

## **3.3. Automatic notification when friends enter or leave alert zone**

While using the alert zone task, when a friend leave or enter a zone, a notification is sent to the user to notify about it asynchronously. This function allows the user to not pay attention to the screen full-time which might be uninteresting to users. This feature allows an interactive method of communicating with the user when friends enter or leave the zone.

## **3.4. Visualizing and integrating danger zones on the alert zones**

While the user create zones to find and follow friends’ geospatial information in alert zone task, there is an option for integrating the danger zones from danger crowdsourcing task into the alert zone. Therefore, the user get notification when an incident happens in his/her regions of interest. This allows for quick response of the users in cases of incidents if the incident happen in critical location for the users.

## **3.5. Geospatial multi-level danger crowdsourcing**

This feature allows the users to share dangers with some rankings on the geospatial surface which is stored in a geospatial database. Accordingly, this feature allows for geospatial queries on the region annotated with danger with some degrees. In addition, this feature is used in other functions such as follow me task and alert zone task to notify the dangerous conditions.

# **4. Application**

## **4.1. Architecture:**

In this implementation, Model-View-Controller is taken as an architecture. According to created DB schema, model notion is implemented. Server is used to implement controller part. And, mobile application defines view layer.

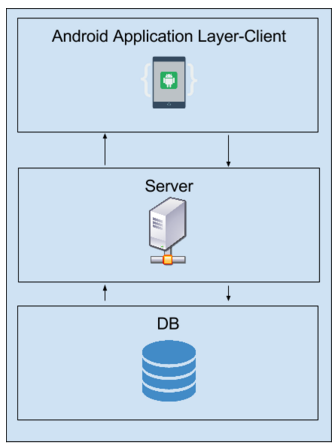


Figure 2. Architecture

## **4.2. Server:**

As server, Amazon Web Services (AWS) EC-2 cloud based service is used. Linux instance is selected to be running server. In addition apache and php is installed to server to accomplish server tasks.

In this implementation server is middle layer between mobile application level and DB level. Server communicates bidirectionally with both mobile application level and DB level. Both returning data to mobile application level and forwarding data to DB are needed to be put in a particular pattern. Logging, registering, querying, update, delete, insert tasks can be handled via php files residing on server.

## **4.3. Database:**

For DB level, Amazon Web Services (AWS) RDS server is used. This server provides different types of DB instances. In this implementation, Postgre is used, because Postgre DB provides spatial data management querying with PostGIS upgrade.

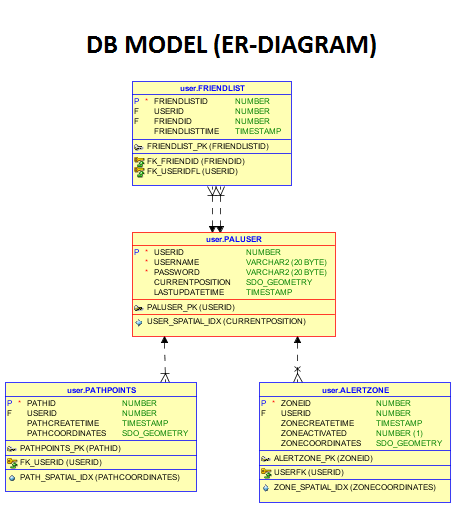


Figure 3. ER Diagram

## **4.4. Android client:**

In mobile application level, android is used to develop client application. In the first step client application is used for registration into system. Then users can log on to their accounts and do further spatial queries and functions.

### **Representative Activities**

### **4.4.1. Login Activity**

In the login page user enters his/her user name and password. If one of them is incorrect then the user is denied from entry. The only way for the user to the app will be a registration.



Figure 4. Login Menu

### **4.4.2 Register Activity**

In the registration activity, user registers him/herself as one of the users of the app. After registration his/her name will be entered to the database. Then he will guided back to the login page again to enter to his/her account.

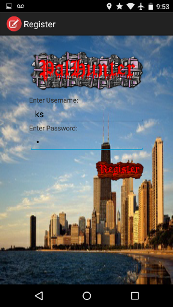


Figure 5. Registration Menu

### **4.4.3 PalHunter Menu Activity**

After entry user see the menu of options. The menu consists of four following activities that only the users that have an account can see. They are: AlertZone, Follow Me, Friends List, and About. The user can also log out if he is finished with the app by the 3-dot options.

Figure 6. Pal Hunter Main Menu

### **4.4.4 Friend List Activity**

Friend List activity is a crucial part of the app since if there is no friend, other activities will be meaningless. So we can search and find people and by clicking on their names we add them to our friends group. It’s also possible to remove one person from our list if we desire so. The friendship relation has been considered two-ways meaning if one adds the other, then the other will have the first one in his friend list as well. So, whatever the first or second person does, the other person can do the same.

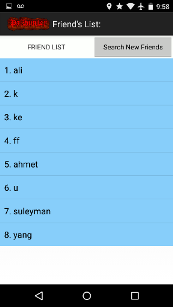
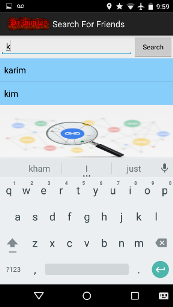
 

Figure 7. Friend List and Searc For Friends Menus

### **4.4.4 Alert Zone Activity**

Alert zone is basically meant to monitor friends inside a predefined zone. One can set many zones of oneself and monitor friends in any combination.

For his convenience location finder has been implemented to the action-bar so the user can simply get to a place by typing the location name or address or coordinates.

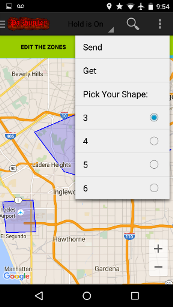
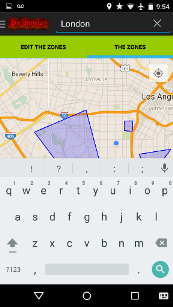
 

Figure 7. Alert Zone Page-1

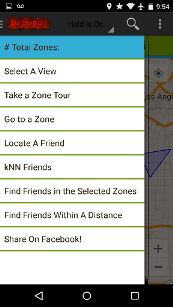
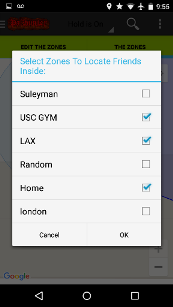
 

Figure 8. Alert Zone Page-2

After search is done, the user will guide to his desired area to set a zone. The search can be LAX, London, Taj Mahal and Everest.

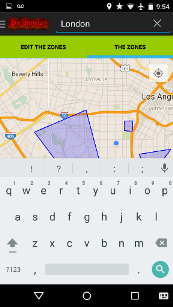
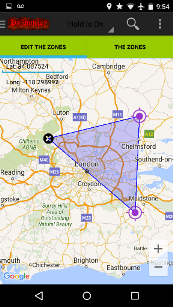
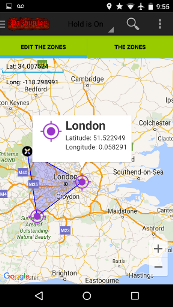
  

Figure 9. Editing zones

In the left drawer menu, user can find many good features such as queries and settings. He can ste his view type to normal, satellite and terrain. Then we implemented spatial queries: K-NN, range query, and search are the ones that we implemented. These queries are separate than the zone monitoring feature. In these queries we help the user find his friends and make boundaries of desire. For example if the user’s friend is coming back from another city the user can make a zone around LAX by defining his/hers zone. But if the user wants 3 nearest friends he can choose so from the menu.

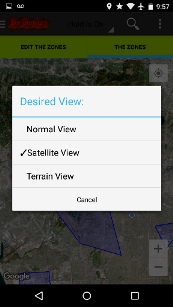
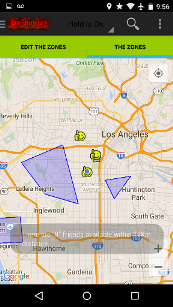
 

Figure 10. Alert Zone Selecting Map Type

For each friend that enters or exits a zone, the user will get relative notifications (alerts). User can set up his/her desired zone and start monitoring for friends’ activities. As you see in below, we have one notification coming for a friend because he left he is about to enter the zone. The zones selected to be monitored are arbitrary and have no limit, i.e. the user can select as many zones s/he wants and get notifications as friends leave or enter.

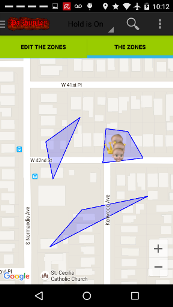
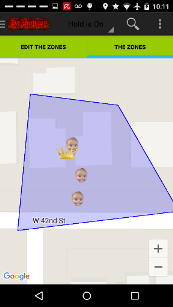
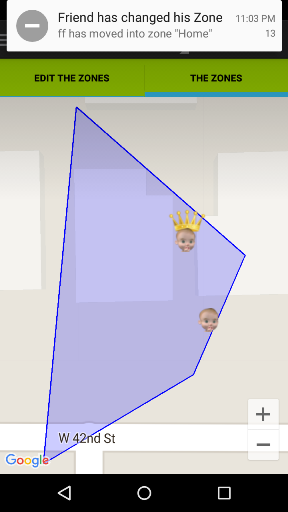


Figure 11. KNN Query with Alert Zone

As user wants he can send distance query. So, for a selected distance the user will see his friends inside. This is different than k-NN search, because we don’t know how many friends are inside the distance circle being returned as the result of the query.

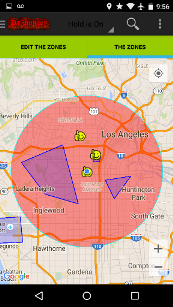


Figure 12. Range Query

The last feature is a simple facebook share dialog that can share our location on facebook. This feed has a link associated with it. After the user desires to share his location on facebook he will be directed to add a caption for the post and share it on facebook. He has the option to go back or continue with his share. So, whoever clicks on the shared post in facebook, can see our location at that time of your share.

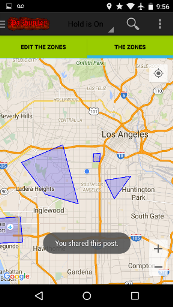
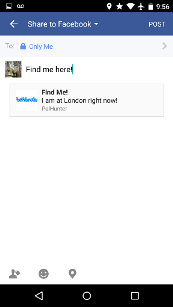


Figure 13. Location Sharing

As I said by clicking the shared post on facebook, we guide to the google maps page with a marker indicating our location:

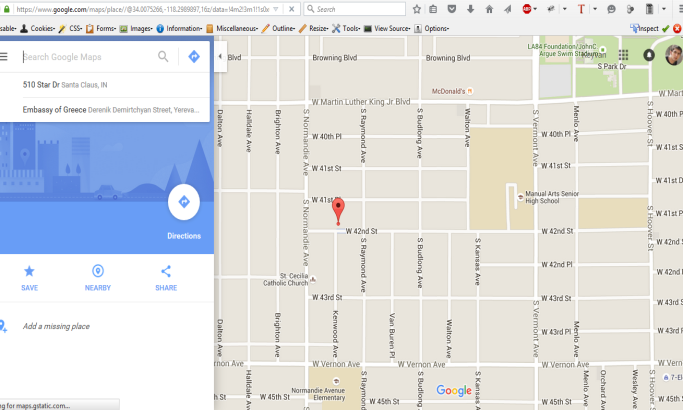
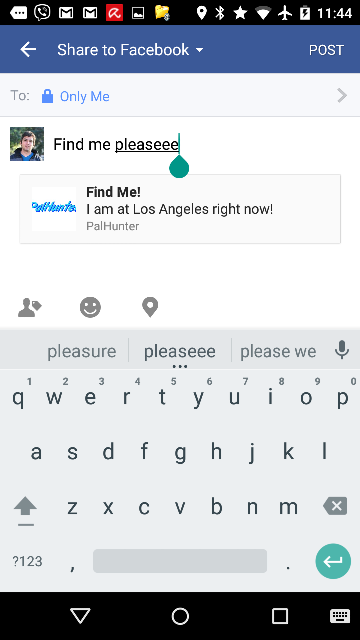
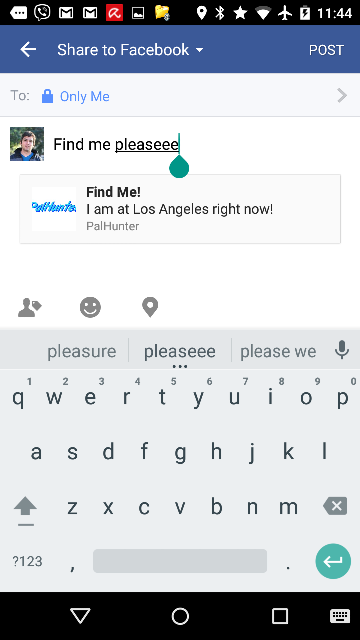


Figure 14. Various Location Sharing Applications

### 

### **4.4.4 Follow Me Activity**

For the follow me section there are two options available. User can either start saving own path, or start following another friend.

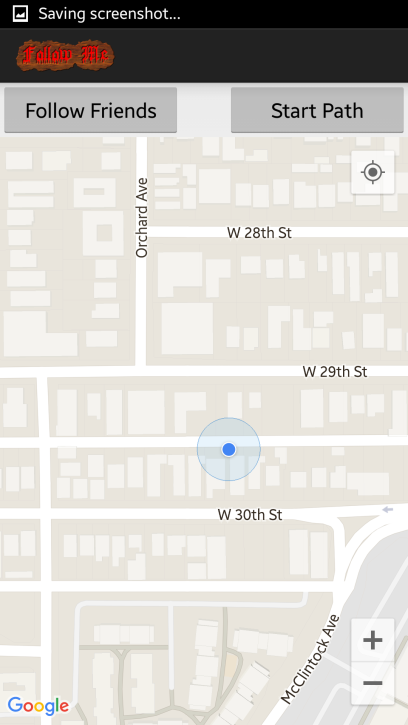


Figure 15. Pal Hunter Follow Me Page

For saving own path into database there are also two options: using road network coordinates or free geometric positions. Here road network positions are aimed to use the application with vehicles. Free geographic positions are more useful for pedestrian applications. During saving own path user can see the path in green line. Note that blue point is the last point on the path.

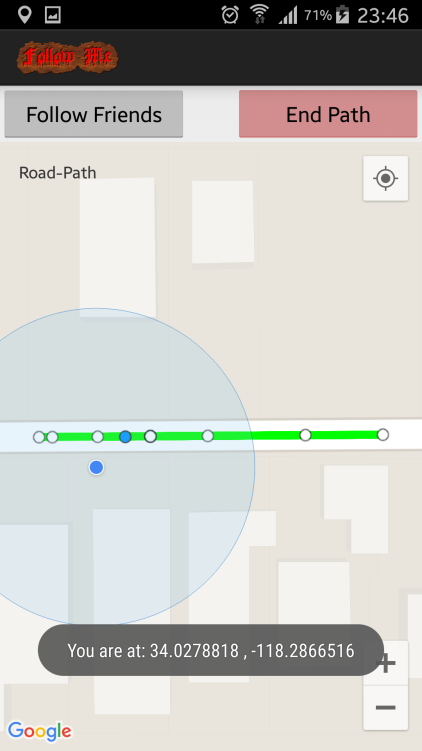
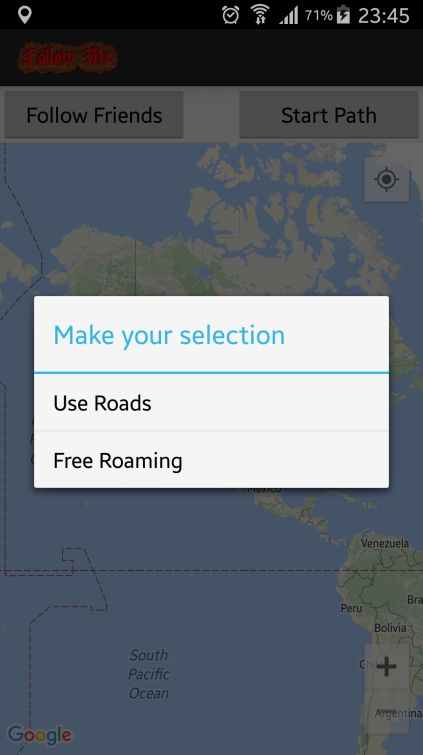


Figure 16. Path Saving

Selection of path method Road network coordinates Free geographic coordinates

Another option is following another user path. Once user clicks the “Follow Friends” button friend names menu appears. User can select any other user in his friendlist. Followed friend’s path is drawn by blue line on the map and blue point is the last updated point that is clickable and shows location data and time. Follow me section grabs the data from DB every 0.5 second and re-adjust the screen zooming as being including both path and user position.

### Screenshot_2015-11-29-23-59-29.png

Figure 17. Following a Friend

# **5. Installation and configuration**

The application consists of 3 main components: client, server, and database. Below are instructions on how to install, configure, and deploy each component to get the application working.

## **Database Installation**

For database installation, AWS RDS service is used. Once the Postgre DB is selected in the control panel, the server completes the installation of DB.

Then for configuring the underlying structure to DB, under “../Schema/” folder:

“paluser\_postgre.php”

“alertzone\_postgre.php”

“friendlist\_postgre.php”

“pathpoints\_postgre.php”

files includes schema DDL codes for schema. These files also includes creating index structure for location data.

“**postgredb.ctnfr2pmdvmf.us-west-2.rds.amazonaws.com:5432**” is the link for the DB instance.

## **Server Installation**

For server installation, AWS EC-2 cloud based service is used. After installation of server machine, Apache server, PHP and Postgre programs are installed into server machine. After setting up completing the server machine the following .php files (under directory of “../phpfiles/”) are included into server:

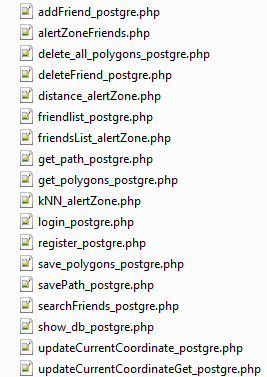


Figure 18. List of PHP files

These files are configured to work with configured Postgre DB server and include queries to manipulate the DB with parameters that are sent by the mobile application level.

This is configured link for the mobile applicaction level: <http://54.187.253.246/selectuser/>”.. ”

In addition “<http://54.187.253.246/selectuser/show_db_postgre.php>” is used for displaying all the dataset inside DB.

## **Client Installation**

For the client installation “../Maps/” directory includes Eclipse based source code.

In order to configure the app to work with different servers, server addresses are need to be changed in the .java files. Another important issue is obtaining a Google API key for the app. Because each API key requires to use particular app and particular Eclipse editor, the API key needs to be changed to get map data from Google Servers.