NYU – Poly CS9033 Mobile Application Programming
Homework 4 – Internet and Location
Professor: Jeffrey Bickford <jbickford@nyu.edu>
Due: 5/4/15 11:55 PM

Overview

In Homework 3 you added a database and shared information across different components and apps using Intents. In Homework 4, we will continue to extend this ETA app with what you have learned in week 8 and 9. We will be adding interaction with a web server as well as the GPS sensors to track a user's location.

Web Server API

For this homework assignment, I have created a web server using Google's AppEngine framework located at the following URL: http://cs9033-homework.appspot.com/. The web server exposes a simple JSON API that accepts HTTP POST requests. The requests are defined as follows:

CREATE_TRIP - sends a message to the web server indicating that a new trip has been created. The web server will return with a 32 bit integer that will identify the trip. The JSON string must include the following objects.

- "command": "CREATE_TRIP"
- "location": [“location name”, ”address”, ”latitude”, ”longitude”]
- "datetime": integer value representing unix/epoch timestamp
- "people": list of people attending the trip
- Request Example: {"command": "CREATE_TRIP", "location": [“location name”, ”address”, ”latitude”, ”longitude”], "datetime": 1382584629, "people": ["John Doe", "Joe Smith"]}
- Successful Response: {"response_code": 0, "trip_id": 3645686546}

UPDATE_LOCATION - sends a location update for the user. The JSON string must include the following objects.

- "command": "UPDATE_LOCATION"
- "latitude": double value obtained via GPS representing the device's latitude
- "longitude": double value obtained via GPS representing the device's latitude
- "datetime": integer value representing unix/epoch timestamp
- Request Example: {"command": "UPDATE_LOCATION", "latitude": 0, "longitude": 0, "datetime": 1382591009}
- Successful Response: {"response_code": 0}

TRIP_STATUS - receives the current status of each person for a trip identified by a trip_id (obtained from the web server via CREATE_TRIP API call). The JSON string must include the following objects.

- "command": "TRIP_STATUS"
- "trip_id": 32 bit integer received by web server after CREATE_TRIP. This should be saved and obtained via a query to your database.
- Request Example: {"command": "TRIP_STATUS", "trip_id": 3645686546}

- Responses Example: {"distance_left": [20.399999999999999, 6.7000000000000002], "time_left": [1920, 900], "people": ["Joe Smith", "John Doe"]}
- The response contains three lists. "people" contains the list of people being tracked for that trip. "distance_left" contains the list of miles left in the order of the "people" list. "time_left" is similar, but reports the number of seconds until arrival at the trip's destination.
- NOTE: for simplicity, the web server will always respond with the above response. Don't worry if the names do not match your actual trip when showing the current trip! Just show these names and time/distance data for your current trip in your database. Please round and convert the data that is coming back to user friendly readable units.

Requirements

- Extend your CreateTrip activity such that it sends a POST request to the web server using the CREATE_TRIP API command. Store the trip_id that the web server returns in your database for future use. Note: you may need to make a modification to your database and update the version number.
- Extend your ViewTrip Activity such that you can “start a trip”. This should set the trip that is viewed as the “current active trip”. Also extend your ViewTrip Activity (or MainActivity) with a button signifying that the user has arrived at their destination.
- Extend your Main Activity such that it displays information related to your current trip. This information will be obtained via the TRIP_STATUS API command. For example, your main activity will now show how many miles and minutes your friends still have to travel.
- Extend your application so that while there is a “current active trip”, it periodically updates the web server with your current location obtained via the GPS.
- You should not be hard coding your JSON requests! You should be querying your database and using JSON methods/APIs to generate the proper request to send to the sever.

Web Server

!!!! NOTE !!!! THE APPENGINE SERVER HAS LIMITED DAILY BANDWIDTH !!!! NOTE !!!!

Due to this being hosted on a free AppEngine instance, bandwidth is limited per day. This means if everyone waits until the final weekend to do this homework, we will run out of bandwidth for that day and you will not be able to connect to the server. If you cannot connect to the server, you will need to host your own appengine server on your computer locally. I have posted the app engine code for you to use if necessary.

Instructions to run AppEngine are located here: https://developers.google.com/appengine/training/intro/gettingstarted. You must run AppEngine using the --host=0.0.0.0 extra parameter so that your emulator/Genymotion can address the local web server. If you are using the emulator, you can address your local web server via http://10.0.2.2:8080. If you are using Genymotion, you can address your local web server via http://192.168.56.1:8080 (or the IP of your VirtualBox host-only network adapter. Use ipconfig/ifconfig to find this.)

What to turn in:

- zip file of your Android project. This should include all resources and source code. We should be able to import it into the Android Development Environment and compile/run your app.
- PDF file with a description of what you have added as well as how you have decided to interact with the Internet. I would also like to know by what method you are using to obtain your GPS location and how often you are doing this. How often are you sending the GPS location to your sever? How often are you interacting with the server in general?