# P1 Instructions

## ChangeLog:

**25 Aug, 2018, 18:20: Initial release**

## Learning objectives

Familiarity with Android development.
Familiarity with Play framework, including server setup and database access.

Streaming mechanism such as Websocket.

## Story

We are implementing an Android application that tracks a user's location during outdoor activities (walking, running, or cycling). The application constantly sends the user's location to a server, which saves the location information in a database and tells the user how far he or she has gone. To save battery on the client, the server also tells the client how long to wait before sending the next update, based on the user's speed.

## Main deliverables

- An Android application that keeps sending its location as a JSON object to a server and showing the total distance returned by the server.
    - Submit **both** of the source code and the APK file. Place the apk file at the root directory so it can be easily found.
    - There should be two text input boxes and one button within the UI. One text input box is for host name of the server, its default value should be **10.0.2.2:9000**, which is the host machine address for android emulator; another text input box is for user name. The button is for starting/stopping. Refer to the screenshot at the last page as an example.
- A server that accepts requests in JSON format, saves the location information in an SQL database, and sends to the client a) total distance and b) how long the client should wait before sending the next update.
    - The wait period is calculated as follows:
        - Keep a moving average of the speed from the last 5 updates
        - If the average <= 1m/s, update interval = 5s
        - If the average >= 20m/s, update interval = 1s (we need more frequent updates to capture fast movement)

        - For speeds between 1m/s and 20m/s, interpolate between 5s and 1s.
        - If the client could not connect to the server in the last attempt, use a default value of 5s.
        - (If your hardware cannot handle fast updates, change the values as you see fit, but use a similar scheme.)
    - Submit the whole project, including the sbt build script(*build.sbt*). Make sure the server can be started by executing command '*sbt run*' in your submitted directory.

## Ingredients

- JDK 8 or higher (required)
- sbt (ver 1.2.1 or higher)
- Play (ver 2.6.18 or higher) [Don't need to download manually]
- Android Studio (ver 3.1.4 or higher)
- (library) Jackson JSON library 2.8.7
- (optional library) Volley (1.1.1 or higher)
- (optional library) Tyrus Standalone Client (1.9 or higher)

# Guidelines

## Server

### 1. Initialization

-- Install the latest version of **sbt**: http://www.scala-sbt.org/download.html

-- Create a new Play project:

*sbt new playframework/play-java-seed.g8*

(Enter project name and other info as prompted)

-- Try running the server from the project root:

*sbt run*

(The first run will take some time as dependencies are downloaded. Test it by visiting localhost:9000.)

Refer to: https://www.playframework.com/documentation/2.6.x/NewApplication

-- (Optional) Setup IDE for your Play application.

Refer to: https://www.playframework.com/documentation/2.6.x/IDE

## 2. Adding a POST Method

-- Add a route in *conf/routes* like this:

```
POST   /locationupdate        controllers.HomeController.handleupdates
```

-- Add a method in your home controller to handle this request. In *app/controllers/HomeController.java*, add *handleupdates()* method.

### Hint

You can use DynamicForm and FormFactory to handle POST requests:

```
...
import com.google.inject.Inject;
import play.data.DynamicForm;
import play.data.FormFactory;
...
@Inject
FormFactory formFactory;
public Result handleupdates() {
        DynamicForm dynamicForm = formFactory.form().bindFromRequest();
        //Logger.info("Username is: " + dynamicForm.get("username"));
        //Logger.info("Time is: " + dynamicForm.get("timestamp"));
        //Logger.info("Latitude is: " + dynamicForm.get("latitude"));
        //Logger.info("Longitude is: " + dynamicForm.get("longitude"));
        return ok("ok, I recieved POST data. That's all...\n");
}
```

-- Handling and Serving JSON

Refer to: https://playframework.com/documentation/2.6.x/JavaJsonActions

Return operation information in JSON format.

### Hint

- Use Jackson JSON library 2.8.7 instead of the newest 2.9.0 to avoid errors.
- Import *jackson.databind.JsonNode* and *jackson.databind.node.ObjectNode*.

## 3. SQL Database Access

-- Add a method to store users' location tracking information in an SQL database, use SQLite as your database engine. Refer to:
https://www.playframework.com/documentation/2.6.x/JavaDatabase
http://www.sqlitetutorial.net/sqlite-java/

### Notes

- Configure database properties in *conf/application.conf* file.
- Inject a *Play.db.Database* to use the default datasource.
- (Optional) Use asynchronous action for database operations.

-- Calculate the movement of the user since start, return it in the response to POST request.

## 4. Running the Server

-- Use:

*sbt run*

to run this application. The default port for a Play application is 9000. You can access your server from:

*localhost:9000*

-- If you wish to access your application via other host names, add them in *conf/application.conf* like this:

```
play.filters.hosts {
    allowed = ["localhost:9001", "example.com:9000", "localhost:9000"]
}
```

You can use:

*sbt run 9001*

to run this application from port 9001.

# Android App

-- Install Android Studio. Use the latest version (Ver 3.1.4).

https://developer.android.com/studio/index.html

-- Create a new Android project.

### Notes

- Minimum SDK: **API 14: Android 4.0 (IceCreamSandwich)**.
- Add internet (*android.permission.INTERNET*) and location (*android.permission.ACCESS_FINE_LOCATION*) permissions in *AndroidManifest.xml*.
- Add 'android:usesCleartextTraffic="true"' to the <application/> tag in your AndroidManifest.xml, if you are targeting API 23 or higher.
- Add an EditText for user name input and at least one Button for basic operations.

### Hint

- You may want to check whether your application actually has location permission upon startup. (Use *ContextCompat.checkSelfPermission*)

-- Add a method for POST requests.

### Notes

- Use Volley. Refer to: https://developer.android.com/training/volley/index.html
- POST at least four parameters: *username*, *timestamp*, *latitude*, and *longitude*.
- When the Button is clicked, start a thread that makes a POST request every x seconds (as defined in the *Main Deliverables* section). Click Button to stop the thread.
- Display the message returned from server.

### Hint

- If you are using an Android Emulator, use your IP address for Server URL instead of "localhost".

-- Use JSON.

Use Volley's *JsonObjectRequest* for JSON post requests.

Example JSON request:

- --header "Content-type: application/json"
- --request POST

- --data '{
  "username": "GUEST",
  "timestamp": "1501745843691",
  "latitude": "47.7474",
  "longitude": "15.7694"
  }'

You may add any other fields to make your application better.

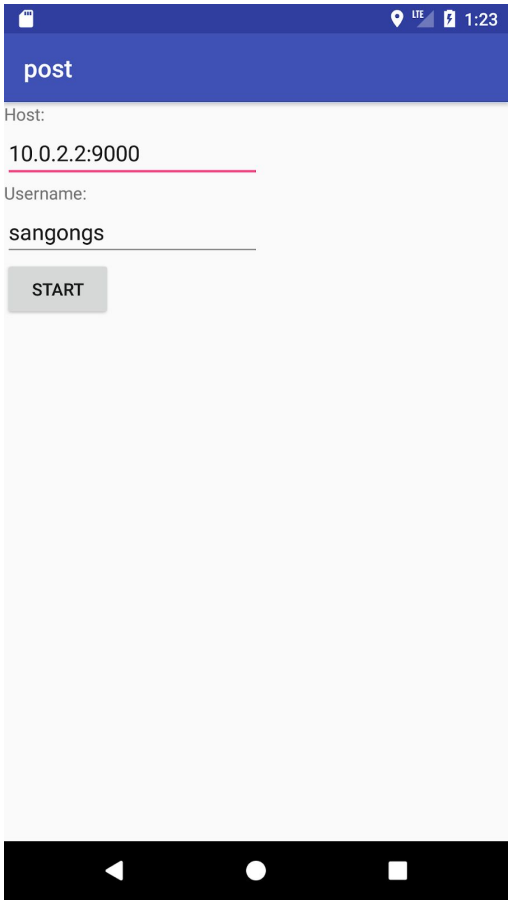
-- Alternative: use Play's WebSocket, and stream location information to server.

Refer to: https://www.playframework.com/documentation/2.6.x/JavaWebSockets

### Hint

- You can use the Tyrus library for WebSocket programming on the Android side. Refer to: https://github.com/tyrus-project/tyrus
- Use *WebSocket.json(JsonNode.class).accept* instead of *WebSocket.Json.accept* on the server side. On the Android side, send JsonNode object instead of Text.
- Include Jackson's core, databind, and annotation libraries to avoid errors.

-- Example screenshot:


1:23
post
Host:
10.0.2.2:9000
Username:
sangongs
START