DOCUMENTATION

Introduction

- 'UCL Quiz' is a location based multiple choice quiz. It takes input from an SQL database, pushing coordinates, location names, quiz questions and answers to the application. This is mediated through a php file, creating the GeoJSON. The required features are fetched, and the main app screen is a map (provided by Google maps) with an array of points marked on it displaying the points from the sql database. These are the locations where when a user reaches them in reality, a proximity alert will trigger. This prompts another activity to load - the question, with a multiple choice answer screen. When the user chooses they then press the 'Post Data' button, which sends their data to the server. When they do this the correct answer is displayed on the screen.

Technical Specifications

- Android version 22 required

Requirements Testing

Element	Pass/Fail	Notes
SQL database set up to record quiz information	Pass	
Website allows users to input values	Pass	
Map with points from SQL database displays on screen	Pass	
On marker click a pop up appears for each point with information about it	Pass	
Upon reaching a point in reality a question pops up	Fail	The question unfortunately pops up only on marker click though the location listener does function with hard coded points and does show a toast message when a user arrives at one of my pre set quiz points
User can select answer and submit it before seeing the correct answer	Pass	

SPECIFIC FEATURES

Hash Map - this in essence fetches the correct question and choices relative to the marker clicked. Ideally this would initialise and select based on the coordinates the user arrived at, however at present the proximity element is not fully collaborative with the quiz pop up and information retrieval.

HashMap()
The constructor constructs a default HashMap. I did this for the question, choice one, choice two, choice three and the answer. An example HashMap initialiser used in my code is below:

This initial constructor establishes a global variable Array List of all values -
ArrayList<HashMap<String, String>> choiceOneLocation = new ArrayList<HashMap<String, String>>();

Within the onMarkerClick function, the various values are parsed, and the value matching the title is pulled to pass through the intent. The title value is the same as the name value in the SQL database and so all values along that row pertaining to particular columns are selected -

```
for (HashMap<String, String> choiceOnes : choiceOneLocation) {
        if (choiceOnes.get(marker.getTitle()) != null) {
            choiceone = choiceOnes.get(marker.getTitle());
            break;
        }
    }
```

Within the addColoursToMarkers class, the individual strings are added -

```
choiceone = feature.getProperty("choiceone");
HashMap<String, String> choiceOne = new HashMap<String, String>();
choiceOne.put(feature.getProperty("name"), choiceone);
choiceOneLocation.add(choiceOne);
```

Intent

The intent passes the correct hash map values through to the SendCheckboxToServer activity.If the proximity sensor was functioning correctly then the intent would be placed in onLocationChanged, however at present it is in the onMarkerClick class. This starts the new activity SendCheckboxToServer when a user clicks a marker point. The information shown on that new screen is relative to the found hash map values for that particular point. The strings e.g. “q” are key values; in the new activity values are selected by their key. On the receiving end the values are integrated into the activity via global string variables.

ShowGeoJSONOnMapWithInfoActivity

```
Intent intent = new Intent();

        intent.setClassName("uk.ac.ucl.cege.cegeg077.zcwcjbu.geojsontest1",
"uk.ac.ucl.cege.cegeg077.zcwcjbu.geojsontest1.SendCheckboxToServer");
        intent.putExtra("q", question);
        intent.putExtra("c1", choiceone);
        intent.putExtra("c2", choicetwo);
        intent.putExtra("c3", choicethree);
        intent.putExtra("a", answer);

        this.startActivity(intent);
        return true;
    }
```

SendCheckboxToServer

```
Bundle extras = getIntent().getExtras();
        question = getIntent().getStringExtra("q");
        c1 = (String) extras.get("c1");
        c2 = (String) extras.get("c2");
        c3 = (String) extras.get("c3");
        answer = (String) extras.get(“a");
```

Website - http://developer.cege.ucl.ac.uk:30522/teaching/user36/SimpleGoogleMapLatLng.php

The website allows the user to add their own quiz points with questions and choices to the SQL database. This as a result allows it to be accessed in the app. The php file contains input boxes for all row values in the SQL table QuestionsApp. This is then processed using processCoordsnew.php and thus the values inputted by the user on the website are added to the SQL table as a new row.

```
<!DOCTYPE html>
<html>
<!-- taken from: http://code.google.com/apis/maps/documentation/javascript/examples/map-simple.html -->
  <head>
    <title>Google Maps JavaScript API v3 Example: Map Simple</title>
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0, user-scalable=no">
    <meta charset="UTF-8">
    <style type="text/css">
      html, body, #map_canvas {
        margin: 0;
        padding: 0;
        height: 100%;
      }
    </style>
    <script type="text/javascript"
        src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>

    <script type="text/javascript">
      var map;

      // this is the function to initialise the map when the page loads - it is called by the 'addDomListener' call below
      function initialize() {
                        // set up startup options for the map - which zoom level, which locaiton should the map be centered on
                        // and which basemap type should appear
                        var myOptions = {
                          zoom: 8,
                          center: new google.maps.LatLng(-34.397, 150.644),
                          mapTypeId: google.maps.MapTypeId.ROADMAP
                        };

                        // create the new map
                        map = new google.maps.Map(document.getElementById('map_canvas'),
                                myOptions);

                        // create a new point to add on the map, using the organisations.png icon
                        var image = 'organisations.png';
                        var myLatLng = new google.maps.LatLng(-33.890542, 151.274856);
                        var beachMarker = new google.maps.Marker({
                                position: myLatLng,
                                map: map,
                                icon: image
                        });

                        // set the text contents of the popup
                var contentString = 'This is my first popup';
```

```
                        // create a default info window pop-up (supplied by Google)
                        var infowindow = new google.maps.InfoWindow({
                                content: contentString
                        });

                        // add a listener onto the point so that it responds to a click event
                        google.maps.event.addListener(beachMarker, 'click', function() {
                          infowindow.open(map,beachMarker);
                        });

                        // add a listener so that when the user clicks on the map the
                        // clicked point is added to the text boxes
                        // document.getElementById allows you to set values for each text box using
its ID
                        google.maps.event.addListener(map, 'click', function(point) {
                                        document.getElementById("latitude").value =
point.latLng.lat();
                                        document.getElementById("longitude").value =
point.latLng.lng();
                                });
    } // end of the initialize function

          // call the initialize method to create the map
          // this listener is called when the window is first loaded
    google.maps.event.addDomListener(window, 'load', initialize);
  </script>
 </head>
 <body>
  <div id="map_canvas" style="width:70%;float:left"></div>
  <div id="lat_lng" style="width:30%;float:right">
        <form action="processCoordsnew.php" method="post">
                Latitude: <input type="text" name="latitude" id="latitude" /><br />
                Longitude: <input type="text" name="longitude" id="longitude" /><br />
                Name: <input type="text" name="name" id="name" /><br />
     Question: <input type="text" name="question" id="question" /><br />
                Choice One: <input type="text" name="one" id="one" /><br />
                Choice Two: <input type="text" name="two" id="two" /><br />
     Choice Three: <input type="text" name="three" id="three" /><br />
     Answer: <input type="text" name="answer" id="answer" /><br />
                <input type="submit" value="Submit" />
        </form>
  </div><!-- end of the lat-lng div -->
 </body>
</html>
```

CLASS BY CLASS FEATURE RUN THROUGH

onMapReady

This initialises the map, using the GoogleMaps function. It establishes the map centre. It also initialises the class retrieveFileFromURL, which in turn initialises the DownloadGeoJSONFile class. The listener is set here to onMarkerClick.

```
public void onMapReady(final GoogleMap newmap) {
      LatLngBounds UK = new LatLngBounds(new LatLng(51, -0.5), new LatLng(51.5, 0.5));
      LatLng mapCentre = new LatLng(51.25, 0);
```

```
        map = newmap;
        map.moveCamera(CameraUpdateFactory.newLatLngBounds(UK, 1, 1, 0));
        retrieveFileFromUrl();
        map.setOnMarkerClickListener((OnMarkerClickListener) this);
```

DownloadGeoJSONFile

```
private class DownloadGeoJsonFile extends AsyncTask<String, Void, JSONObject> {
        protected JSONObject doInBackground(String... params) {
            try {
                // Open a stream from the URL
                InputStream stream = new URL(params[0]).openStream();

                String line;
                StringBuilder result = new StringBuilder();
                BufferedReader reader = new BufferedReader(new InputStreamReader(stream));

                while ((line = reader.readLine()) != null) {
                    // Read and save each line of the stream
                    result.append(line);
                }

                // Close the stream
                reader.close();
                stream.close();

                // Convert result to JSONObject
                return new JSONObject(result.toString());
            } catch (IOException e) {
                Log.e(mLogTag, "GeoJSON file could not be read");
            } catch (JSONException e) {
                Log.e(mLogTag, "GeoJSON file could not be converted to a JSONObject");
            }
            return null;
        }
```

This is a crucial aspect as it allows the GeoJSON file from the server to be converted to a JSON object. The method is accessed in RetrieveFileFromURL. This is where the link to the php coded website is placed.

```
private void retrieveFileFromUrl() {
        String mGeoJsonUrl
                = "http://developer.cege.ucl.ac.uk:30522/teaching/user36/QuestionsApp.php";
        DownloadGeoJsonFile downloadGeoJsonFile = new DownloadGeoJsonFile();
        downloadGeoJsonFile.execute(mGeoJsonUrl);
    }
```

After DownloadGeoJSONFile is triggered, the onPostExecute method checks that there first is a JSON object before creating a new GeoJsonLayer, passing in downloaded GeoJSON file as JSONObject. The addColoursToMarkers method is then triggered.

```
        protected void onPostExecute(JSONObject jsonObject) {
            if (jsonObject != null) {
                mLayer = new GeoJsonLayer(map, jsonObject);
                // Add the layer onto the map
```

```
            addColorsToMarkers();
            mLayer.addLayerToMap();
        }

    }
```

addColoursToMarkers

This class iterates through the features of the layer and scrapes the relevant data. It uses the method getFeatures to get name, question, choice one, choice two, choice three and the answer. It fetches these based on string input pertaining to the column name. The icon, title and snippet are set here, with the latter two directly coming from the feature.getProperty function. I explain the HashMap function in a separate section.

If the proximity sensor is also working effectively, then this segment -

```
            GeoJsonPoint coord = (GeoJsonPoint) feature.getGeometry();
            double lat = coord.getCoordinates().latitude;
            double lng = coord.getCoordinates().longitude;
            GeoPoint gMapPoint = new GeoPoint(lat, lng);
            ArrayList<GeoPoint> pointList = new ArrayList<GeoPoint>();
            pointList.add(gMapPoint);
```

would interact with the GeoPoint activity in order to get that working. The above fetches the coordinates from the layer, and as a double creates a new map point. The map points are then added to an array list pointList (that is initialised within the class, but is also a global variable for access within the onLocationChanged class).

```
  private void addColorsToMarkers() {
        // Iterate over all the features stored in the layer
        for (GeoJsonFeature feature : mLayer.getFeatures()) {

            if (feature.hasProperty("name") && feature.hasProperty("question")) {

                BitmapDescriptor pointIcon =
BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE);
                GeoJsonPointStyle pointStyle = new GeoJsonPointStyle();
                double coords = Double.parseDouble(feature.getProperty("coords"));
                // Create a new point style
                // Set options for the point style
                pointStyle.setIcon(pointIcon);
                pointStyle.setTitle(feature.getProperty("name"));
                HashMap<String, String> qustionL = new HashMap<String, String>();
                qustionL.put(feature.getProperty("name"), feature.getProperty("question"));
                questionLocation.add(qustionL);
                pointStyle.setSnippet(feature.getProperty("question"));
                question = feature.getProperty("question");
                q.add(question);
                Log.d(question, "question");
                GeoJsonPoint coord = (GeoJsonPoint) feature.getGeometry();
                double lat = coord.getCoordinates().latitude;
                double lng = coord.getCoordinates().longitude;
                GeoPoint gMapPoint = new GeoPoint(lat, lng);
                ArrayList<GeoPoint> pointList = new ArrayList<GeoPoint>();
                pointList.add(gMapPoint);
```

```
choiceone = feature.getProperty("choiceone");
c1.add(feature.getProperty("choiceone"));
Log.d(choiceone, "choiceone");
HashMap<String, String> choiceOne = new HashMap<String, String>();
choiceOne.put(feature.getProperty("name"), choiceone);
choiceOneLocation.add(choiceOne);

choicetwo = feature.getProperty("choicetwo");
c2.add(feature.getProperty("choicetwo"));
HashMap<String, String> choiceTwo = new HashMap<String, String>();
choiceTwo.put(feature.getProperty("name"), choicetwo);
choiceTwoLocation.add(choiceTwo);

choicethree = feature.getProperty("choicethree");
Log.d(choicethree, "choicethree");
HashMap<String, String> choiceThree = new HashMap<String, String>();
choiceThree.put(feature.getProperty("name"), choicethree);
choiceThreeLocation.add(choiceThree);

c3.add(feature.getProperty("choicethree"));
answer = feature.getProperty("answer");
HashMap<String, String> answerhash = new HashMap<String, String>();
answerhash.put(feature.getProperty("name"), answer);
answerLocation.add(answerhash);
Log.d(answer, "answer");
feature.setPointStyle(pointStyle);
```

onMarkerClick

This is currently where the intent is, and so the new activity SendCheckboxToServer starts upon the initialisation of this class, which occurs when the user clicks a marker point. The class gets the title and snippet as established in the addColoursToMarkers class. The Hash Map is also extended here from the same class - this again is explain further in a separate segment.

```
public boolean onMarkerClick(final Marker marker) {
    AlertDialog.Builder dialog = new AlertDialog.Builder(this);
    dialog.setTitle(marker.getTitle());
    dialog.setMessage(marker.getSnippet());
    String url;
    WebView wv = new WebView(this);
    wv.setBackgroundColor(Color.CYAN);
    wv.getSettings().setDefaultFontSize(16);
    wv.getSettings().setDefaultTextEncodingName("utf-8");
    dialog.setView(wv);
    dialog.show();


    for (HashMap<String, String> questions : questionLocation) {
        if (questions.get(marker.getTitle()) != null) {
            question = questions.get(marker.getTitle());
            break;
        }
    }
```

```
for (HashMap<String, String> choiceOnes : choiceOneLocation) {
    if (choiceOnes.get(marker.getTitle()) != null) {
        choiceone = choiceOnes.get(marker.getTitle());
        break;
    }
}

for (HashMap<String, String> choiceTwos : choiceTwoLocation) {
    if (choiceTwos.get(marker.getTitle()) != null) {
        choicetwo = choiceTwos.get(marker.getTitle());
        break;
    }
}

for (HashMap<String, String> choiceThrees : choiceThreeLocation) {
    if (choiceThrees.get(marker.getTitle()) != null) {
        choicethree = choiceThrees.get(marker.getTitle());
        break;
    }
}

for (HashMap<String, String> answers : answerLocation) {
    if (answers.get(marker.getTitle()) != null) {
        answer = answers.get(marker.getTitle());
        break;
    }
}
```

onLocationChanged

This class allows the proximity alert to function, and if fully functioning then it would be where the intent goes to start the new activity. Below I have hardcoded the points and this allows the toast message to show, however the intent to start the activity isn't working fully at present. Ideally as previously stated the quiz activity would start here. The pointList array would also ideally contain the location points as fetched from the SQL database and they would be the ones used for the location listener. This is crucial as it allows for the location information to be used as an indicator to fetch particular information i.e. question, choices and answers pertaining to a particular point via Hash Map, in the same way as detailed above but with the coordinates being the determining value rather than marker.setTitle.

```
public void onLocationChanged(Location location) {
        //when location is changed set alert, start an action (in this case the activity)
        //hard coded point list
        ArrayList<GeoPoint> pointList = new ArrayList<GeoPoint>();
        // ucl
        GeoPoint gMapPoint = new GeoPoint(51.524610, -0.133510);
        // petrie
        GeoPoint gMapPoint2 = new GeoPoint(51.523604, -0.132952);
        //bf
        GeoPoint gMapPoint3 = new GeoPoint(51.525000, -0.132621);
        //archaeology
        GeoPoint gMapPoint4 = new GeoPoint(51.524908, -0.131580);
        //string to compare lat received to sql to gather the correct information
        // add these points to the list
        pointList.add(gMapPoint);
        pointList.add(gMapPoint2);
```

```
            pointList.add(gMapPoint3);
            pointList.add(gMapPoint4);
            Toast.makeText(getBaseContext(),
                    "You have moved to a Quiz Question Location: Latitude/longitude:" +
location.getLatitude() + " " + location.getLongitude(), Toast.LENGTH_LONG).show();
            //method to fetch points avoiding hard coding, instead using getFeatures
            for (int i = 0; i < pointList.size(); i++) {
                //GeoPoint gp = pointList.get(i);
                GeoPoint gp = pointList.get(i);
                Log.d("MRG", "It worked");
                Location fixedLoc = new Location("one");
                float distance = location.distanceTo(fixedLoc);
                Double lat = Double.valueOf(String.valueOf(gp.getLatitude()));
                Double lng = Double.valueOf(String.valueOf(gp.getLongitude()));
                fixedLoc.setLatitude(lat);
                fixedLoc.setLongitude(lng);
                Log.i("location", lat + " " + location.getLatitude());
                Log.i("location", lng + " " + location.getLongitude());


            }
            Log.d("MRG", "It worked");
        }
```

GeoPoint

This is a separate activity to the reset of the above, and only contains this one class. It establishes the doubles lat and lng, allowing them to be used to create coordinate pairs in the ShowGeoJSONOnMapWithInfoActivity class.

```
public class GeoPoint {
    private Double latitude;
    private Double longitude;

    public GeoPoint(Double lat, Double lng){
        latitude = lat;
        longitude = lng;
    }

    public Double getLongitude() {
        return longitude;
    }
    public Double getLatitude() {
        return latitude;
    }
}
```

getCheckboxes

This class fetches the extras passed through the intent. These values are then used to set the text for the question and all choices. The class records which of the checkboxes a user checks.

```
private String getCheckBoxes() {
        String result;
        Bundle extras = getIntent().getExtras();
        question = getIntent().getStringExtra("q");
```

```
        c1 = (String) extras.get("c1");
        c2 = (String) extras.get("c2");
        c3 = (String) extras.get("c3");
        answer = (String) extras.get("a");
        // wv = (WebView) findViewById(R.id.textView1);
        //wv.loadUrl(url);
        mTextview = (TextView) findViewById(R.id.textView1);

        mTextview.setText(question);
        CheckBox oneCheckBox = (CheckBox) findViewById(R.id.one);
        oneCheckBox.setText(c1);

        result = "=" + oneCheckBox.isChecked();
        CheckBox twoCheckBox = (CheckBox) findViewById(R.id.two);
        twoCheckBox.setText(c2);

        result += "&" + "=" + twoCheckBox.isChecked();

        CheckBox threeCheckBox = (CheckBox) findViewById(R.id.three);
        threeCheckBox.setText(c3);

        result += "&" + "=" + threeCheckBox.isChecked();
        Log.d("url", c1);
        return result;
    }
```

submitDataPost

This class records the information inputted by the user - including name and checkboxes checked. It encodes the information to be sent to another php file (process_form). This class is triggered when the user presses the submit button.

```
public void onClick(View view)
    {
        switch (view.getId())
        {
            case R.id.button:
                submitDataPost();
                break;
        }
    }

 private void submitDataPost() {
        // extract the data from the form into an array of NameValuePair objects
        //NameValuePair nameValuePairs[] = getFormDetails();

        String data="";

        // create an asynchronous operation that will take these values
        // and send them to the server
        SendHttpRequestTask sfd = new SendHttpRequestTask();
        try {
            EditText et = (EditText)findViewById(R.id.firstName);
            String theFirstName = (String) et.getText().toString();

            // get the surname
```

```
        EditText et2 = (EditText)findViewById(R.id.surname);
        String theSurname = (String) et2.getText().toString();


        String urlParameters =
                "firstname=" + URLEncoder.encode(theFirstName, "UTF-8") +
                        "&surname=" + URLEncoder.encode(theSurname, "UTF-8");
        // also add the checkboxes - are they checked or not
        urlParameters += "&"+getCheckBoxes();
        Log.i("result",urlParameters);

        sfd.execute(urlParameters);
    }
    catch           (UnsupportedEncodingException e){}

}
```

SendHttpRequestTask

This sends a request for the information recorded on the app by the user to be posted as a string to process_form.

```
private class SendHttpRequestTask extends AsyncTask<String, Void, String> {

    @Override
    protected void onPreExecute() {

    }

    @Override
    protected String doInBackground(String... params) {
        URL url;
        String urlParams = params[0];
        String targetURL = "http://developer.cege.ucl.ac.uk:30522/teaching/user36/
process_form.php";

        HttpURLConnection connection = null;
        try {
            //Create connection
            url = new URL(targetURL);
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Content-Type",
                    "application/x-www-form-urlencoded");

            connection.setRequestProperty("Content-Length", "" +
                    Integer.toString(urlParams.getBytes().length));
            connection.setRequestProperty("Content-Language", "en-US");

            connection.setUseCaches(false);
            connection.setDoInput(true);
            connection.setDoOutput(true);

            //Send request
            DataOutputStream wr = new DataOutputStream(
                    connection.getOutputStream());
```

```
            wr.writeBytes(urlParams);
            wr.flush();
            wr.close();

            //Get Response
            InputStream is = connection.getInputStream();
            BufferedReader rd = new BufferedReader(new InputStreamReader(is));
            String line;
            StringBuffer response = new StringBuffer();
            while ((line = rd.readLine()) != null) {
                response.append(line);
                response.append('\r');
            }
            rd.close();
            connection.disconnect();
            return response.toString();

        } catch (Exception e) {

            e.printStackTrace();
            return null;

        } finally {

            if (connection != null) {
                connection.disconnect();
            }
        }
    }
```

onPostExecute

After the above has completed, a separate textView will change on the screen to reveal the correct answer for that particular question. The answer is again pulled from the intent that is passed through to the SendCheckboxToServerActivity.

```
protected void onPostExecute(String response) {
        TextView xTextview = (TextView) findViewById(R.id.answer);
        xTextview.setText(answer);
```

Layout Files

activity_geojson

This layout file displays the map on screen and fills the whole page. It pertains to ShowGeoJSONOnMapWithInfoActivity.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
```

```
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

tools:context="uk.ac.ucl.cege.cegeg077.zcwcjbu.geojsontest1.ShowGeoJSONOnMapWithInfoActi
vity">


    <fragment
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="com.google.android.gms.maps.SupportMapFragment"
        android:name="com.google.android.gms.maps.MapFragment"/>

</RelativeLayout>
```

activity_send_checkbox_to_server

This layout file pertains to SendCheckboxToServer and so displays on the screen when the intent is activated. It contains the following features:
EditText - first name
EditText - surname
TextView - question
Checkbox - choice one
Checkbox - choice two
Checkbox - choice three
TextView - answer
Button - submit

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="uk.ac.ucl.cege.cegeg077.zcwcjbu.geojsontest1.SendCheckboxToServer">


    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:id ="@+id/mainwrapper"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
```

```
        android:layout_height="100px"
        >

        <TextView android:layout_width="150px" android:text="Name"
android:layout_height="wrap_content" android:textAppearance="?android:attr/
textAppearanceLarge" android:id="@+id/textView1"></TextView>
        <EditText android:tag="firstname" android:name="firstname"
android:layout_width="wrap_content" android:id="@+id/firstname2"
android:layout_height="wrap_content" android:layout_weight="1">
            <requestFocus></requestFocus>
        </EditText>
    </LinearLayout>
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="100px"
        >
        <TextView android:layout_width="150px" android:text="Surname"
android:layout_height="wrap_content" android:textAppearance="?android:attr/
textAppearanceLarge" android:id="@+id/textView2"></TextView>
        <EditText android:tag="surname" android:name="surname"
android:layout_width="wrap_content" android:id="@+id/surname"
android:layout_height="wrap_content" android:layout_weight="1">
        </EditText>
    </LinearLayout>

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="100px"
            >


        </LinearLayout>

        <Button
            android:id="@+id/button1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Post Data" />

        <TextView
            android:id="@+id/webResponse"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/hello"
            />

        <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="100px"
            >

        </LinearLayout>
```

```
    <CheckBox
        android:text="TextView"
        android:id="@+id/one"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

      <CheckBox
        android:text="TextView"
        android:id="@+id/two"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

    <CheckBox
        android:text="TextView"
        android:id="@+id/three"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
       />

    <TextView android:tag="firstname" android:name="firstname"
android:layout_width="match_parent" android:id="@+id/firstname" android:layout_height="250dp"
android:layout_weight="1"
        android:textAppearance="@style/Bubble.TextAppearance.Dark"
        android:textColor="@color/abc_input_method_navigation_guard"
        android:textSize="40sp">
        <requestFocus></requestFocus>
    </TextView>



  </LinearLayout>

</RelativeLayout>
```

PHP files

- simpleGoogleMapLatLng
- process_form
- processCoordsnew

Testing

Component Tested	Description of Test	Results	Pass/Fail
User Website	1. Check information displays correctly on the php link 2. Make sure user inputted information is correctly transferred to the SQL database QuestionsApp by first looking at the submit page via php and then checking the database contents on the server by selecting SELECT * FROM QuestionsApp	1. I initially had issues with the formatting and also I did not think to differentiate the coordinate values from the text inputs. This silly error was rectified and then I had no issues viewing the display (detailed in the screenshots section) 2. Once I had sorted the above issue the information showed smoothly.	Pass
Map	Map displays on phone screen, centred at correct point	I had no issues with this as I used a map directly from Google Maps and code that was covered in class to display it.	Pass
Fetching GeoJSON File	1. Php file functioning correctly - flagged up by log statement 'GeoJSON file can't be read' 2. Contents successfully converting to JSON object - flagged up by 'GeoJSON file can't be converted to a JSON object'	Only had minor issues with the first test here - once I had edited my code to incorporate all table values it was no longer an issue. The php file also had checkpoints stating where the issue was e.g. connecting to database, converting etc. so that was really helpful to have display	Pass
Getting Features	Add log statement for each column value that I needed to fetch to check that a. they were not null and b. that they were the right values.	No issues here	Pass
Add Colours to Markers	Checked points showed on the map app display, Checked correct title and snippet were showing for each point.	As long as the GeoJSON file functioned correctly so did this.	Pass
Hash Map	Check correct value displayed when passing through intent Log statement within Hash Map loop.	No issues once I figures out how to correctly set up the Hash Map.	Pass

Component Tested	Description of Test	Results	Pass/Fail
Intent	Check information displayed in checkboxes layout	initially had issues as I placed the get Extras in the wrong part of the code. But after I altered that it worked fine.	Pass
Proximity	1. Toast message pops up at location (hard coded) 2. Points from database added to listener, toast message pops up on arrival 3. Intent starts upon reaching a location point 4. Intent passes with correct information pertaining to a particular location point	1. Fine 2. Had it working at one point however when transferring the code over to a new application to make it cleaner I couldn't pass over this element which is frustrating. If I had the time I would have this working 3. This would have worked by linking values to coords in the database had I got 2 working. 4. Again would have no issues had it not be for the second test.	1. Pass 2. Fail 3. Fail (as a result of failing 2) 4. Fail (as a result of failing 2)
Checkboxes	1. Check correct question information displayed 2. Check answers could be selected on display	1. Linked to intent and Hash Map. As long as these worked it was fine. 2. No issues here.	Pass
Posting Data	1. Check button to submit functioned. 2. Check answer showed on screen 3. Checked user input was recorded on monitor	No issues here.	Pass

Proximity Sensor Issues

The proximity sensor was the one aspect that, whilst I could comfortably set up a location listener and toast message pop up for hard coded points, I had a real struggle to fetch the coordinates from the database. This becomes an issue when a user wants to add their own question points to the App.
I also struggled to fetch the correct information from the database in relation to the question. In the case that a hard coded point triggered an intent (by placing the intent within the onLocationChanged class) a random question popped up as opposed to the specific information. As this testing was before the inclusion of the HashMap function I would like to try and test this out, but at the moment my thinking is that it would have to evaluate the information in the database given the coords values - this depends on the SQL coordinate points being fed to the location listener. Had I the time I feel that I could comfortably manage to marry the whole thing together.

API Used

- Version 2
- AIzaSyD9O0su-RAw4bS7uiV4gcwl4E45Vh0Jr7w

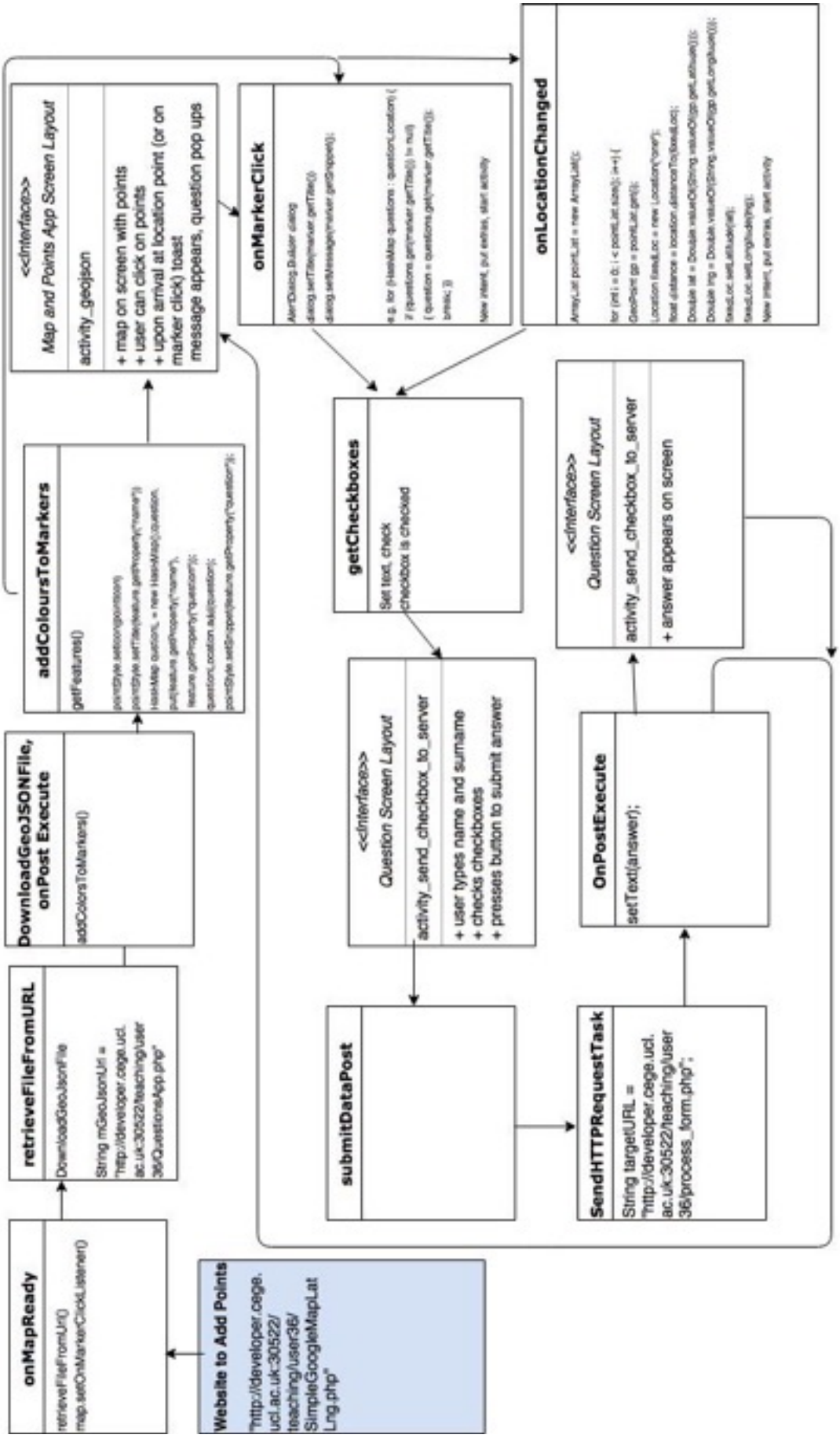
SQL Database Structure

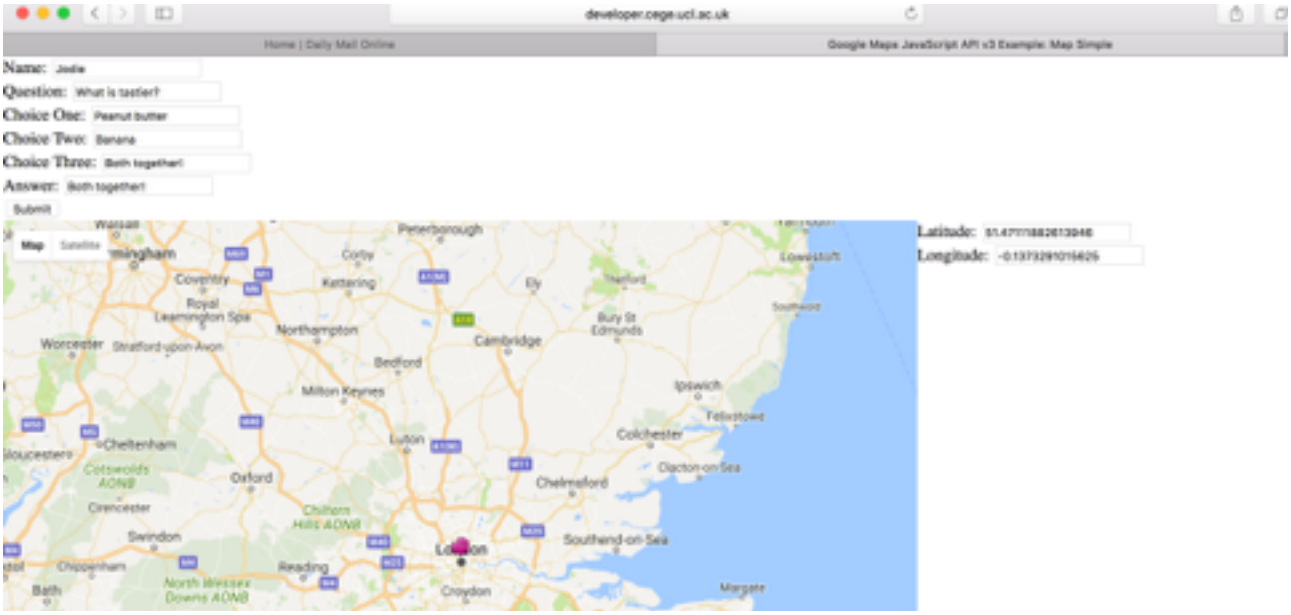
Table Name - QuestionsApp
Columns - name, question, choiceone, choicetwo, choicethree, answer, coords
URL - http://developer.cege.ucl.ac.uk:30522/teaching/user36/QuestionsApp.php

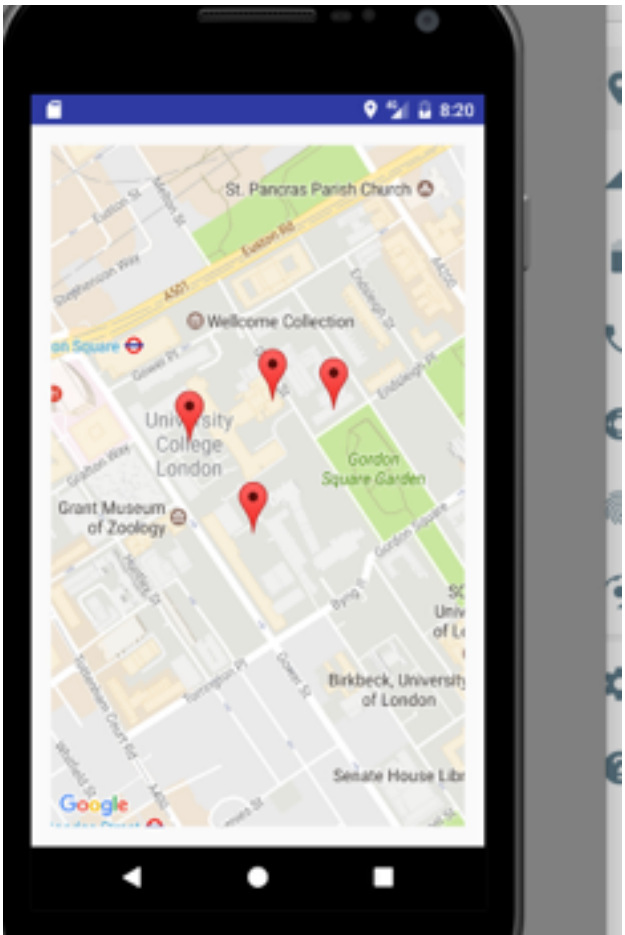
USER GUIDE

- The main screen is a map, displaying several points around UCL, marked in red. You can click on each of these points at any time to see what they are. Upon physically reaching each of these points, a screen will pop up displaying a multiple choice question. You can answer this by clicking a checkbox and pressing 'Post Data'. This send the data to the server and then returns the correct answer. Then press the back button to return to the map screen and continue to the next point.
- Users can also add their own points and quiz markers to the map using an accompanying website (http://developer.cege.ucl.ac.uk:30522/teaching/user36/QuestionsApp.php). You can fill in all questions, choices and name of the point and select the point on the map. It will then be added to the Quiz.

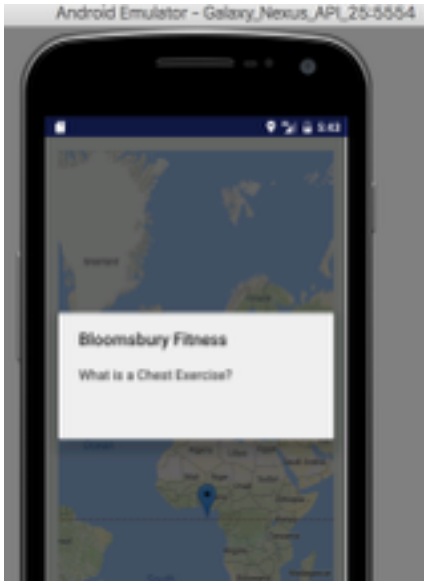
Flow Chart of Activity

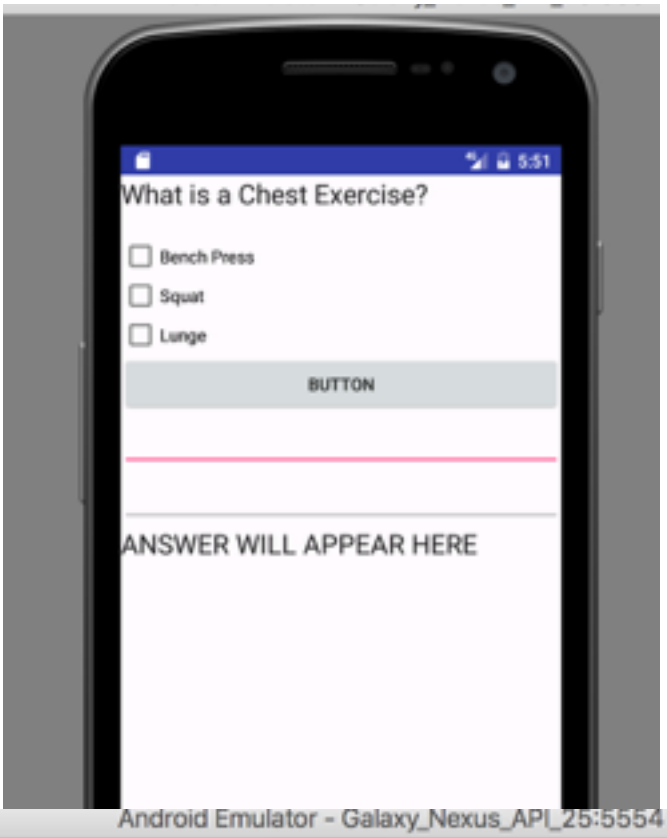


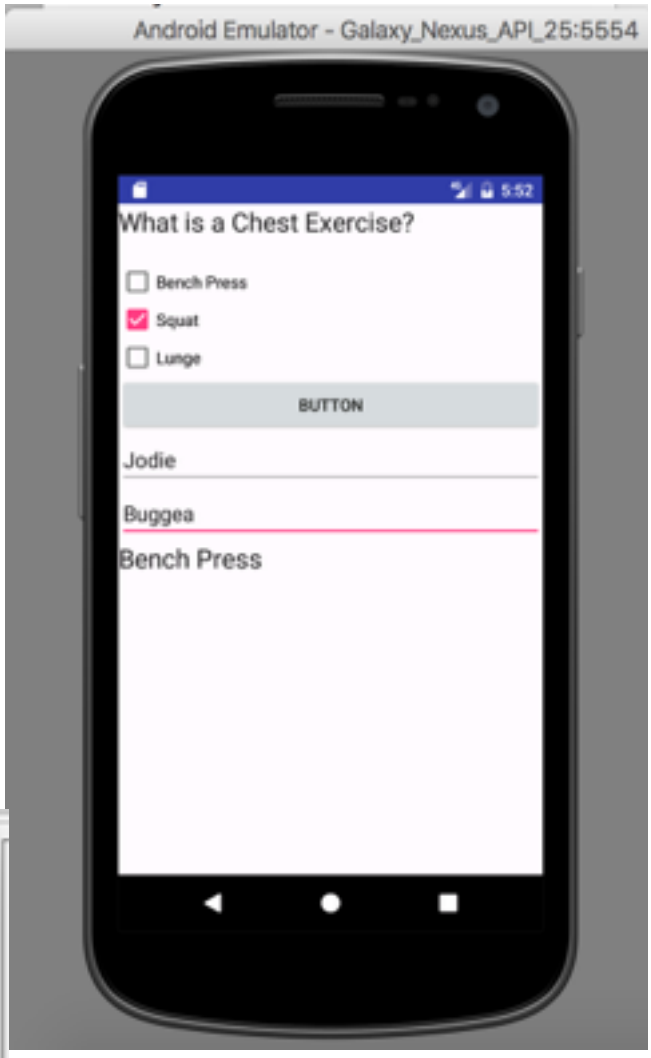


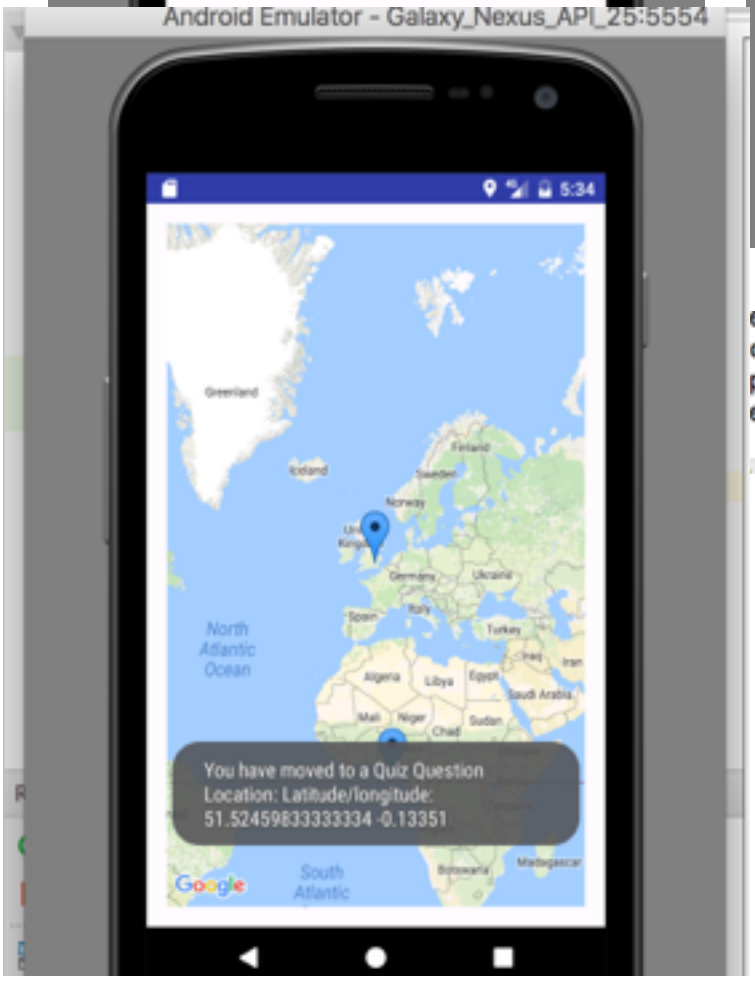


Far above - website for user to add points
Above - the screen shown to the user once their values are submitted
Left - Initial screen upon starting the app - points zoomed in
Below - example of a pop up upon clicking a marker point










```
e&Lunge=false&Bench press=false
cise?
press=true
e&Lunge=false&Bench press=true
```

Above - Process of answering question including initial SendCheckboxToServer activity, page after user submits (showing answer) and log monitor on computer screen showing the processed answers.
Left - Proximity toast message as currently working for hard coded points. States that user has arrived at a quiz location point

Left - Mock up of ideal question screen, including question number and button back to the main map page
Below - QuestionsApp SQL layout

```
Last login: Tue Aug 29 15:50:34 2017 from eduroam-int-pat-8-180.ucl.ac.uk
-bash-4.2$ psql -d user36db -U user36
Password for user user36:
psql: FATAL:  password authentication failed for user "user36"
-bash-4.2$
-bash-4.2$ psql -d user36db -U user36
Password for user user36:
psql (9.4.5)
Type "help" for help.

user36db=> select * from QuestionsApp;
        name         |                question                |     choiceone     |     choicetwo      |   choicethree   |
    answer       |                coords                | lat
-----------------+--------------------------------------+-------------------+--------------------+-----------------+---
 UCL Main Quad           | What year was UCL founded?              | Eighteen Twenty Six | Nineteen Eighteen    | Eighteen Fifty  | E
ighteen Twenty Six | 0101000000047E9D2BF24C349402ABC2D043928C1BF |  52
 Bloomsbury Fitness      | What is a Chest Exercise?               | Squat               | Lunge                | Bench press     | B
ench press         | 0101000000333333333C34940F4A3E194B9F9C0BF |  52
 Petrie Museum           | How many artefacts are at the Petrie Museum? | Fifty Thousand      | One Hundred Thousand | Eighty Thousand | E
ighty Thousand     | 0101000000307D47405C349409109FB359204C1BF |  52
 Institute of Archaeology | How many floors does Archaeology have? | Four                | Six                  | Eight           | S
ix                 | 0101000000053B4722F30C349400F62670A9DD7C0BF |  52
 test                    | test                                    | a                   | b                    | c               | d
                   |                                         |
 c                       |                                         |                     |                      |                 |
                   | 0101000000000000000000E03F000000000001240 |
 d                       | d                                       | d                   | d                    | d               | d
                   | 0101000000A7C714215FB40C000000011CD6240 |
 fd                      | d                                       | f                   | f                    | d               | f
                   | 0101000000D17A55A0F0D340C000000075B46240 |
(8 rows)

user36db=>
```