Kelvin Watson
ONID: watsokel
OSU ID: 932540242

CS 496 Assignment 4: Mobile Development

MALENAH **Native Android** Application

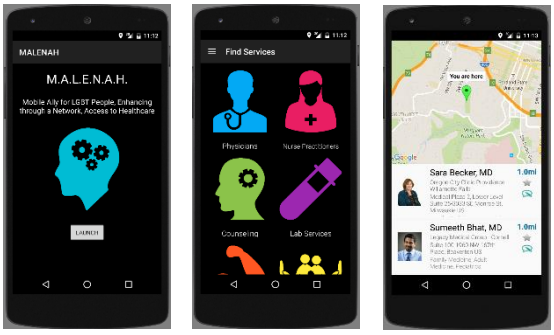
Demo Video URL:
http://web.engr.oregonstate.edu/~watsokel/cs496/watsokel496a4demo.mp4

My API: http://malenah-api-prod.appspot.com
This API is based on the API I wrote for Assignment 3 Part 2, with several additions, described under "Functionality and Details of my API."

Development Platform

I used **Android Studio** to develop my **native** application. The application is written in Java.

Native Android Application



My native Android application is called M.A.L.E.N.A.H., which stands for Mobile Ally for LGBT People, Enhancing, through a Network, Access to Healthcare. Briefly, LGBT people face significant barriers to healthcare access and have a host of unique health-related needs. For many in this population, this leads to avoidance and/or delay of necessary medical care. [1] Challenges to health and well-being include anti-gay physical or psychological violence, increased suicide risk and homelessness of LGBT youth, greater risk of HIV and other sexually-transmitted infections, substance abuse, and mental illness. [2] [3]
Contributing to these issues are a general lack of healthcare professionals equipped to handle LGBT health-related issues, as well as denial of access to insurance coverage for same-sex domestic partners and transgendered people. [4]

As such, my Android application allows users to access information about culturally-competent healthcare providers who understand medical issues faced by LGBT people in the hopes of closing the LGBT health disparities gap.

Code used to make POSTs to the API

The code used to make HTTP POSTs involves using an AsyncTask, which performs a background operation. This is because network calls, such as POSTs cannot be performed in the main UI thread in an Android application.

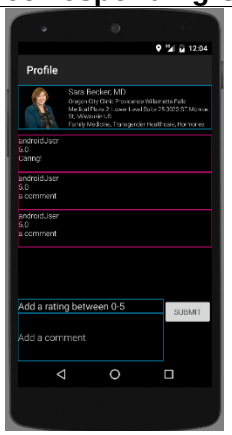
The user can write reviews about a healthcare provider by posting a rating and a comment. The Android application then executes an AsyncTask to perform a POST request to the API. Below is the working code for the HTTP POST and an explanation of the code with corresponding UI screen captures.

Code

AsyncTask

```java
public class PostReviewAsyncTask extends AsyncTask<Void,Void,String> {
    Context context;
    Map<String, String> postParams;

    /**
     * Overloaded constructor
     */
    public PostReviewAsyncTask(Context context, Map<String,String> postParams) {
        this.context = context;
        this.postParams = new LinkedHashMap<String,String>(postParams);
    }

    /**
     * Default constructor
     */
    public PostReviewAsyncTask() {
    }

    public byte[] generatePostData(){
        StringBuilder postData = new StringBuilder();
        for (Map.Entry<String, String> dParam : postParams.entrySet()) {
            if (postData.length() != 0) postData.append('&');
            try {
                postData.append(URLEncoder.encode(dParam.getKey(), "UTF-8"));
                postData.append('=');
                postData.append(URLEncoder.encode(String.valueOf(dParam.getValue()), "UTF-8"));
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
        }
        byte[] postDataBytes = null;
        try{
            postDataBytes = postData.toString().getBytes("UTF-8");
        } catch (UnsupportedEncodingException e){
            e.printStackTrace();
        }
        return postDataBytes;
    }

    @Override
    protected String doInBackground(Void... params) {
        /* Set params */
        byte[] postDataBytes = generatePostData();

        /* Set headers and write */
        URL url = null;
        HttpURLConnection conn = null;
        try {
            url = new URL("http://malenah-api-prod.appspot.com/review");
            conn = (HttpURLConnection)url.openConnection();
            conn.setRequestMethod("POST");
            conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
            conn.setRequestProperty("Content-Length", String.valueOf(postDataBytes.length));
            conn.setDoOutput(true);
            conn.getOutputStream().write(postDataBytes);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch(IOException e){
            e.printStackTrace();
        }

        /* Retrieve response */
        Reader in = null;
        String response = new String();
        try {
            in = new BufferedReader(new InputStreamReader(conn.getInputStream(), "UTF-8"));
            response = "";
            for (int c = in.read(); c != -1; c = in.read()){
                response += (char) c;
            }
        } catch(IOException e){
            e.printStackTrace();
        }
        return response;
    }
```

Explanation of code and corresponding UI screens



When a user makes a comment, the comment is immediately posted to the Scrollable view above the EditText view.

PostReviewAsyncTask is a class that I defined that extends the AsyncTask base class. I overloaded the constructor in order to pass in the application's context and a Map containing the POST parameters to post to the API.

The output stream writes data as bytes, so the post parameters that were passed into the AsyncTask (as a LinkedHashMap) must be converted to an array of bytes.

doInBackground() is a overridden method of the AsyncTask base class. It is the method that performs the main operation of the AsyncTask. In this case, this method opens a URL connection to my API (malenah-api-prod.appspot.com), sets the request headers by specifying the method (POST) and content type (application/x-www-form-urlencoded") and content length, and POSTs to the review URL (http://malenah-api-prod.appspot.com/review). doInBackground() also reads in a response from the API.

onPostExecute() is another method inherited from the AsyncTask base class. It checks to see if the response contains a 'key' property. If so, the Review entity was successfully created and POSTed to the API.

```
    protected void onPostExecute(String resp){
        super.onPostExecute(resp);
        //parse the string
        JSONObject jResp=null;
        try {
            jResp = new JSONObject(resp);
            if(jResp.has("key")) {
                ((ProfileActivity)this.context).postReviewDone(jResp, true);
            } else{
                ((ProfileActivity)this.context).postReviewDone(null, false);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Execute async task

```
new PostReviewAsyncTask(ProfileActivity.this, postParams).execute();
```

In order to execute the PostReviewAsyncTask, the execute() function must be called from the activity that calls the AsyncTask.

Mobile Features of the Application

The MALENAH native Android application uses the following mobile features, which are explained in further detail below:

- Web Connectivity
- Geolocation
- Android AsyncTasks
- Android Services
- Google Maps API
- Drawer Activity

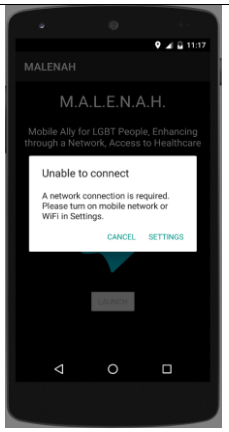
Web Connectivity

The Android app requires data retrieved from my API at malenah-api-prod.appspot.com. As such, it requires an internet connection to retrieve this data. If the user is not connected to the internet, the user is prompted to activate WiFi or cellular data to allow network access, as shown in the screen capture of the Android Virtual Device (Emulator) below:

The below shows code that checks if the user is connected to the Internet. If not, he/she is prompted to enable WiFi or cellular data.

```
public boolean isFullyConnected(Context context){ //Check if internet is enabled
    if (isNetworkAvailable(context)){
        if(isConnectedMobile(context)){
            return true;
        }
        if(isConnectedWifi(context)){
            return true;
        }
    }
    return false;
}

protected void enableInternet(){ //if internet is not enabled, prompt user to enable
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("A network connection is required. Please turn on mobile network or WiFi in Settings.")
            .setTitle("Unable to connect")
            .setCancelable(false)
            .setPositiveButton("Settings",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {
                            startActivity(new Intent(Settings.ACTION_SETTINGS));
                        }
                    }
            )
            .setNegativeButton("Cancel",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {
                            MainActivity.this.finish();
                        }
                    }
```

UI



```
            );
    AlertDialog alert = builder.create();
    alert.show();
}
```

Geolocation

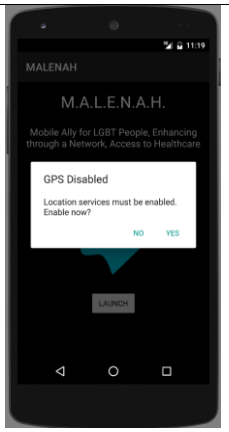
The Android app also requests the user's geolocation. This geolocation is used to plot the user's location on a Google Map (explained below under Google Map API). I have implemented several contingencies for obtaining the user's location, in the event that the last known location from the user's geolocation cannot be obtained. These methods are described below:

1. **Geolocation**: If the user has not activated geolocation services on his/her phone, he/she will be prompted to do so on start of the app. If the user's geolocation (latitude and longitude) cannot be extracted from the geolocation, the user's location will be obtained using one of the contingencies listed below (2 or 3).

The code below shows how the prompt is generated if GPS or a Network Connection is unavailable to determine the user's geolocation. If location has not been enabled, the user is prompted to enable it in his/her Android device settings.

```
protected boolean enableLocation(){
    if(ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 0);
    }
    if (checkLocationPermission()) {
        Log.d("LOCATION (permission)", "user granted permission");
    } else {
        Log.d("LOCATION (permission)", "permission denied");
        return false;
    }
    if  ((Build.VERSION.SDK_INT >= 23) &&
            (ContextCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) &&
            (ContextCompat.checkSelfPermission(context, Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED)) {
        return false;
    }
    locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
    gpsEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER); //check GPS status
    networkEnabled = locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER); // check network status
    if (!gpsEnabled && !networkEnabled){
        Log.d("LOCATION (GPS)", "disabled, ask user to enable");
        //show dialog to allow user to enable location settings
        AlertDialog.Builder dialog = new AlertDialog.Builder(context);
        dialog.setTitle("GPS Disabled");
        dialog.setMessage("Location services must be enabled. Enable now?").setCancelable(false);
        dialog.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                startActivityForResult(new Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS),
SET_LOCATION_REQUEST);
            }
        });

        dialog.setNegativeButton("No", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) { //do nothing if user selects no
            }
        });
        dialog.show();
        return false;
    } else if (gpsEnabled) {
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 10, this);
    }else if (networkEnabled){
        locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,1000,10,this);
    }
    return true;
}
```

UI Prompt



2. **ip-api.com API**: If location services is active but the application was unable to obtain the user's last known location (as is the case if an emulator is used, or if an Android device is newly activated), the application calls the ip-api.com API in an AsyncTask (asynchronous task) as a background operation to obtain a location based on his/her current IP address. [5]

AsyncTask Code

This AsyncTask executes only if the user's last known geolocation is unknown. This is possible if there was no last known location, even if location services are activated on the Android device (e.g. emulator). In that case, the application will attempt to retrieve the user's location using a GET request to ip-api.com in this AsyncTask.

```
public class SetLocationFromIPAsyncTask extends AsyncTask<Void, Void, String> {
    private Context context;
    private double lat = Double.NEGATIVE_INFINITY;
    private double lng = Double.NEGATIVE_INFINITY;
```

API Response

http://ip-api.com/json

{"as":"AS7922 Comcast Cable Communications,

```java
    private URL url;
    private HttpURLConnection urlConnection;
    private Location location;
    private double portlandORLat = 45.52;
    private double portlandORLng = -122.6819;

    public SetLocationFromIPAsyncTask(Context context, URL url, Location location) {
        this.url = url;
        this.location = location;
        this.context = context;
    }

    private Boolean parseJSONString(String jsonStr) {
        try {
            if (jsonStr != null) {
                JSONObject jsonObj = new JSONObject(jsonStr);
                if (jsonObj != null) {
                    Iterator<String> itr = jsonObj.keys();
                    while (itr.hasNext()) {
                        String key = itr.next();
                        if (key.equals("lat")) {
                            setLat(Double.parseDouble(jsonObj.get(key).toString()));
                        } else if (key.equals("lon")) {
                            setLng(Double.parseDouble(jsonObj.get(key).toString()));

                        }
                    }
                    if (getLat() > Double.NEGATIVE_INFINITY && getLat() > Double.NEGATIVE_INFINITY) {
                        return true;
                    }
                } else {
                    return false; //jsonObj is null
                }
            }
        } catch (JSONException e) {
            Log.e("parseJSONString()", "error");
        }
        return false;
    }


    private HttpURLConnection connectToURL(){
        try {
            urlConnection = (HttpURLConnection)url.openConnection();
            return urlConnection;
        } catch (IOException e) {
            Log.e("LOCATION (error)", "error opening connection");
            return null;
        }
    }

    private String retrieveJSON() throws IOException {
        try {
            BufferedReader inputStream = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
            String jsonStr = inputStream.readLine();
            return jsonStr;
        } catch (IOException e) {
            Log.e("LOCATION (JSON)", "error reading stream, internet connection maybe lost");
        }
        return null;
    }

    private void setFailSafeLocation(){
        setLat(portlandORLat);
        setLng(portlandORLng);
    }

    @Override
    protected String doInBackground(Void... params) {
        urlConnection = connectToURL();
        if(urlConnection != null) {
            String jsonStr = null;
            try {
                jsonStr = retrieveJSON();
                if(!parseJSONString(jsonStr)){
                    setFailSafeLocation();
                }
            } catch (IOException e) {
                e.printStackTrace();
                setFailSafeLocation();
            }
        } else {
            setFailSafeLocation();
        }
        urlConnection.disconnect();
        return "done";
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        location.setLatitude(getLat()); //set latitude
        location.setLongitude(getLng()); //set longitude
        ((DrawerActivity)this.context).locationDone(location);
    }

    public double getLat() {
        return lat;
    }

    public void setLat(double lat) {
        this.lat = lat;
    }

    public double getLng() {
        return lng;
    }

    public void setLng(double lng) {
        this.lng = lng;
```

Inc.","city":"Portland","country":"United States","countryCode":"US","isp":"Comcast Cable","lat":45.5073,"lon":-122.6932,"org":"Comcast Cable","query":"73.25.153.201","region":"OR","regionName":"Oregon","status":"success","timezone":"America/Los_Angeles","zip":"97201"}

```
    }
}
```

3. **Default location**: If both of the above fail to obtain the user's geolocation, a default location (in downtown Portland) is set.

Code

See above AsyncTask's setFailSafeLocation() method:

```
private void setFailSafeLocation(){
   setLat(portlandORLat);
   setLng(portlandORLng);
}
```

Android AsyncTasks

Asynchronous Tasks allows background operations to be performed without threads/handlers. Operations such as network calls (e.g. HTTP POST and HTTP GET) cannot be performed on the main UI thread, so they must be either performed as an Android AsyncTask or an Android Service.

My application performs several AsyncTasks, one to fetch data from the API as a GET request, and one to POST data to the API. I have included relevant code for these AsyncTasks:

HTTP **GET** AsyncTask

```
public class FetchReviewsAsyncTask extends AsyncTask<Long, Void, String> {
    …
    @Override
    protected String doInBackground(Long... i) {
        try {
            primitive = providerId.longValue();
            urlStr = "http://malenah-api-
prod.appspot.com/provider/"+primitive+"/review";
            url = new URL(urlStr);
            urlConnection = (HttpURLConnection)url.openConnection();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            BufferedReader inputStream = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
            jsonArrStr = inputStream.readLine();
            jsonArr = new JSONArray(jsonArrStr);
        } catch(JSONException e){
            Log.e("FETCHREV","JSONException");
        } catch (IOException e) {
            Log.e("FETCHREV", "error reading stream, internet connection
maybe lost");

        }
        return "done";
    }
```

HTTP **POST** AsyncTask

```
public class PostReviewAsyncTask extends AsyncTask<Void,Void,String> {
    …
    @Override
    protected String doInBackground(Void... params) {


        /* Set params */
        byte[] postDataBytes = generatePostData();

        /* Set headers and write */
        URL url = null;
        HttpURLConnection conn = null;
        try {
            url = new URL("http://malenah-api-prod.appspot.com/review");
            conn = (HttpURLConnection)url.openConnection();
            conn.setRequestMethod("POST");
            conn.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");
            conn.setRequestProperty("Content-Length",
String.valueOf(postDataBytes.length));
            conn.setDoOutput(true);
            conn.getOutputStream().write(postDataBytes);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch(IOException e){
            e.printStackTrace();
        }

        /* Retrieve response */
        Reader in = null;
        String response = new String();
        try {
            in = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));
            response = "";
            for (int c = in.read(); c != -1; c = in.read()){
                //System.out.print((char)c);
                response += (char) c;
            }
        } catch(IOException e){
            e.printStackTrace();
        }
        return response;
    }
```

Android Services

An Android Service performs operations continuously in the background without a user interface. [6] I use an Android service to obtain all of the provider information using a GET request to malenah-api-prod.appspot.com/provider API prior to allowing the user to proceed to the next screen. Note that the "launch" button is inactive until the user has granted network and location access, and all of the healthcare provider data from the malenah-api-prod.appspot.com has been retrieved. Once the data has been retrieved from the API, it is broadcasted back to MainActivity (the home screen) so that the Launch button can be activated.

Android Service to retrieve provider data from MainActivity (the home screen). Relevant code	**UI**

```java
public class FetchAllDataService extends Service {
    private String filterStr = "com.watsonlogic.malenah.malenah3.providers";
    private String jsonResponse = null;
    private URL url;
    private HttpURLConnection urlConnection;

    public FetchAllDataService() {
    }

    /* Retreive data from database and broadcast it back to MainActivity */
    @Override
    public int onStartCommand(final Intent intent, int flags, int startId){
        try {
            Runnable r = new Runnable() {
                @Override
                public void run() {
                    try {
                        url = new URL("http://malenah-api-prod.appspot.com/provider");
                        urlConnection = (HttpURLConnection)url.openConnection();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    try {
                        BufferedReader inputStream = new BufferedReader(new InputStreamReader(urlConnection.getInputStream()));
                        String jsonResponse = inputStream.readLine();
                        Intent intent = new Intent();
                        intent.setAction(filterStr);
                        intent.putExtra("providers", jsonResponse);
                        sendBroadcast(intent);
                    } catch (IOException e) {
                        Log.e("FETCH (JSON)", "error reading stream, internet connection maybe lost");
                    } finally {
                        urlConnection.disconnect();
                    }
                }
            };
            Thread getDefaultEventAllBeersThread = new Thread(r);
            getDefaultEventAllBeersThread.start();
            return Service.START_STICKY;
        } catch(Exception e){
            e.printStackTrace();
            return Service.START_NOT_STICKY;
        }
    }
}
```

Google Maps API

My application makes use of the Google Maps API to show the user's location, as well as location of healthcare providers. As mentioned before, the application obtains the user's geolocation. This geolocation is then used to plot the user's location on the Google map, using the location object's latitude and longitude.

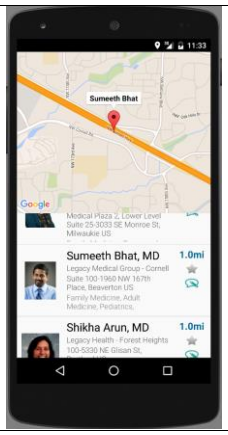
The code below shows how the user marker and the healthcare providers' markers are placed on the map. If the geolocation was obtained via the Android Location Manager (meaning that user's last known location was obtained successfully via GPS or Network connection or via ip-api.com/json), then the user's marker is placed based on that location. However, if the user's location is null (unsuccessfully obtained through GPS, network or ip-api.com/json), then a default location in downtown Portland is set.

UI

```java
public void placeUserMarker() {
    if (location != null && userLatLng != null) {
        map.moveCamera(CameraUpdateFactory.newLatLngZoom(userLatLng, 14));
        userMarker = map.addMarker(new MarkerOptions()
                .position(userLatLng)
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN))
                .title("You are here"));
        updateMapCenter(userLatLng,userMarker);
    } else {
        map.moveCamera(CameraUpdateFactory.newLatLngZoom(defaultLatLng, 14));
        userMarker = map.addMarker(new MarkerOptions()
                .position(defaultLatLng)
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN))
                .title("You are here"));
        updateMapCenter(defaultLatLng,userMarker);
    }
}

public void placeItemMarkers() {
    if (rowItems != null && rowItems.size() > 0)
        for (RowItem ri : rowItems) {
            ri.setMapMarker(map.addMarker(new MarkerOptions()
                    .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED))
                    .position(new LatLng(ri.getLatitude(), ri.getLongitude()))
                    .title(ri.getFirstName()+ " "+ri.getLastName())));
        }
}
```

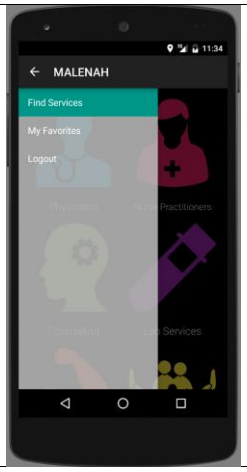
In addition to showing the user's location on the map, clicking on a healthcare provider in the list view animates the map camera and centers the camera on the particular provider's marker on the map.

The code shows how the center of the map is updated when a user clicks on a healthcare provider in the list view.	UI

`protected void updateMapCenter(LatLng l, Marker marker) {`
`  map.animateCamera(CameraUpdateFactory.newLatLngZoom(l, 14));`
`  marker.showInfoWindow();`
`}`
	

Drawer Activity

The drawer activity is a UI element that acts like a navigation sleeve/shelf. It is implemented using Android fragments, which are reusable modules that each have their own lifecycle. Each option in the navigation sleeve is a separate fragment. [8]

UI Drawer in	UI Drawer out
	

Functionality and Details of my API (http://malenah-api-prod.appspot.com/)

I added several additional properties to the Provider entity in the API I submitted for Week 3 Part 2 in order for the Android application to work correctly with the API. I added a category in order to separate the different types of healthcare providers, an icon_url to display the healthcare provider's picture if available, as well as several other address properties. The latitude and longitude were necessary for placing markers on the Google map. I also added an additional Category entity to categorize healthcare providers into different professions to enable users to select a Physician, Nurse or Chiropractor for example.

Provider Entity from Week 3 Part 2's API	**This week's modifications to the Provider Entity (bolded lines indicate newly added properties)**

`class Provider(Model):`
`  first_name = ndb.StringProperty(required=True)`
`  last_name = ndb.StringProperty(required=True)`
`  designation = ndb.StringProperty(required=True)`
`  organization = ndb.StringProperty()`
`  specializations = ndb.KeyProperty(repeated=True)`
`  phone = ndb.StringProperty(required=True)`
`  email = ndb.StringProperty()`
`  website = ndb.StringProperty()`
`  accepting_new_patients = ndb.BooleanProperty()`
	`class Provider(Model):`
	`  category = ndb.StringProperty(required=True)`
	`  icon_url = ndb.StringProperty()`
	
	`  first_name = ndb.StringProperty(required=True)`
	`  last_name = ndb.StringProperty(required=True)`
	`  designation = ndb.StringProperty(required=True)`
	`  specializations = ndb.KeyProperty(repeated=True)`
	
	`  organization = ndb.StringProperty()`
	`  building = ndb.StringProperty()`

```
    street = ndb.StringProperty()
    city = ndb.StringProperty()
    state = ndb.StringProperty()
    country = ndb.StringProperty()
    zipcode = ndb.StringProperty()
    notes = ndb.StringProperty()
    latitude = ndb.FloatProperty()
    longitude = ndb.FloatProperty()

    phone = ndb.StringProperty(required=True)
    email = ndb.StringProperty()
    website = ndb.StringProperty()
    accepting_new_patients = ndb.BooleanProperty()
```

New Category Entity

The category entity enables the user to select the type of healthcare provider he/she requires.

```
class Category(Model):
    name = ndb.StringProperty(required=True)
```

UI

The API (http://malenah-api-prod.appspot.com) returns JSON data, as shown in the samples below:

GET http://malenah-api-prod.appspot.com/provider/


```
[
  {
    "category": "Physician",
    "building": "Medical Plaza 2, Lower Level Suite 25",
    "first_name": "Sara",
    "last_name": "Becker",
    "designation": "MD",
    "city": "Milwaukie",
    "icon_url": "http://www.nwpc.com/wp-content/uploads/2015/02/Becker-S-165_pp.png",
    "notes": "Low-cost care once a month at the Old Town Clinic. Gender Experience Dr. Becker has
probably treated more trans patients than any doctor in Portland. GMW-trained four years ago.",
    "zipcode": "",
    "longitude": -122.631258,
    "email": "",
    "website": "http://www.nwpc.com/index.shtml",
    "phone": "5036594988",
    "state": "OR",
    "street": "3033 SE Monroe St",
    "key": 5105650963054592,
    "country": "US",
    "latitude": 45.445854,
    "organization": "Oregon City Clinic Providence Willamette Falls",
    "specializations": [
      {
        "name": "Family Medicine",
        "key": 5086100271923200
      },
      {
        "name": "Transgender Healthcare",
        "key": 5750085036015616
      },
      {
        "name": "Hormones",
        "key": 5657382461898752
      }
    ],
    "accepting_new_patients": true
  },
  {
    "category": "Physician",
    "building": "Suite 100",
    "first_name": "Sumeeth",
    "last_name": "Bhat",
    "designation": "MD",
    "city": "Beaverton",
    "icon_url":
"http://www.legacyhealth.org/~/media/Images/Providers/b/h/Bhat_Sumeeth.jpg?w=140&h=170&bc=ffffff&as=0"
,
    "notes": "LGBT friendly with open and accepting practice. Welcoming environment with patient
centered practice. Comfortable with handling both adolescents and adults. Multicultural practice with
lots of experience interacting with diverse communities.",
    "zipcode": "",
    "longitude": -122.848045,
    "email": "",
    "website": "http://www.legacyhealth.org/Providers/Sumeeth-Bhat.aspx",
    "phone": "5036726000",
    "state": "OR",
    "street": "1960 NW 167th Place",
    "key": 5144752345317376,
    "country": "US",
```


GET http://malenah-api-prod.appspot.com/specialization/5086100271923200


```
{
  "name": "Family Medicine",
  "key": 5086100271923200
}
```

```
    "latitude": 45.534177,
    "organization": "Legacy Medical Group - Cornell",
    "specializations": [
      {
        "name": "Family Medicine",
        "key": 5086100271923200
      },
      {
        "name": "Adult Medicine",
        "key": 5154321532452864
      },
      {
        "name": "Pediatrics",
        "key": 5161210659995648
      }
    ],
    "accepting_new_patients": true
  }, …]
```


Use of my API

To access resources in my API at http://malenah-api-prod.appspot.com/, the same instructions from Assignment 3 Part 2 apply, and are included below:

API URL Structure for Accessing Resources

The MALENAH API supports HTTP GET, POST, PUT and DELETE, and exposes the following URI's for public consumption (ending forward slashes are optional):

To access Provider entities (and its associated Review entities, and Review entities' associated Reply entities), use the following URL's:

http://malenah-api.appspot.com/provider

http://malenah-api.appspot.com/provider/<providerID>/

http://malenah-api.appspot.com/provider/<providerID>/review/

http://malenah-api.appspot.com/provider/<providerID>/review/<reviewID>/

http://malenah-api.appspot.com/provider/<providerID>/review/<reviewID>/reply

http://malenah-api.appspot.com/provider/<providerID>/review/<reviewID>/reply/<replyID>/

To access Specialization entities:

http://malenah-api.appspot.com/specialization

http://malenah-api.appspot.com/specialization/<specializationID>/

To access Review entities (and its associated Reply entities):

http://malenah-api.appspot.com/review

http://malenah-api.appspot.com/review/<reviewID>/

http://malenah-api.appspot.com/review/<reviewID>/reply/

http://malenah-api.appspot.com/review/<reviewID>/reply/<replyID>/

To access Reply entities:

http://malenah-api.appspot.com/reply

http://malenah-api.appspot.com/reply/<replyID>/

Error Handling

The application implements several mechanisms for error checking:

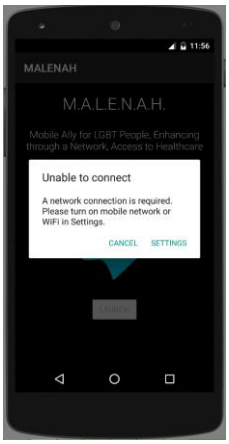
1. checking if an internet connection is enabled on the Android device
2. checking if location services is enabled on the Android device
3. checking for invalid user input (empty or incorrect input)

1. Checking for Internet Connection

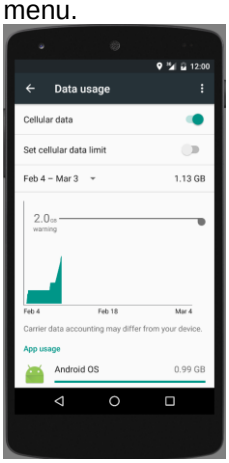
In MainActivity.java (the home screen),I check to see if Internet is enabled on the Android device. If not, the user is prompted to enable their internet connection (either WiFi or cellular data).	**Explanation of code and corresponding UI screens**

`if(!isFullyConnected(getApplicationContext())) { //check connection`
`  Log.d("NETWORK","not connected");`
`  enableInternet();`
`  return;`
`}else{ //internet connected`
`  getData();`
`  Log.d("NETWORK","connected");`
`  if (!enableLocation()) { //check Location`
`  return;`
`  }`
`}`
 	
`public boolean isFullyConnected(Context context){`
`  if (isNetworkAvailable(context)){`
`  Log.d("NETWORK","available");`
`  if(isConnectedMobile(context)){`
`  Log.d("NETWORK","mobile connected");`
`  return true;`
	Check if internet connectivity (WiFi or cell data) is enabled on the Android device.
	
	isFullyConnected() calls several submethods (isNetworkAvailable()), isConnectedMobile(), isConnectedWiFi()) to check for cell data or WiFi connection.

```java
        }
        if(isConnectedWifi(context)){
            Log.d("NETWORK","wifi connected");
            return true;
        }
    }
    return false;
}

//check if network connection is available
public boolean isNetworkAvailable(Context context) {
    ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
    if (activeNetwork != null && activeNetwork.isConnected()) {
        return true;
    } else{
        return false;
    }
}

//check if cell data is enabled
public boolean isConnectedMobile(Context context){
    ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo info = cm.getActiveNetworkInfo();
    if(info != null && info.isConnected() && info.getType() == ConnectivityManager.TYPE_MOBILE) {
        return true;
    }
    return false;
}

//check if wireless connection is enabled
public boolean isConnectedWifi(Context context){
    ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo info = cm.getActiveNetworkInfo();
    if (info != null && info.isConnected() && info.getType() == ConnectivityManager.TYPE_WIFI){
        return true;
    }
    return false;
}


protected void enableInternet(){
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("A network connection is required. Please turn on mobile network or WiFi in
Settings.")
            .setTitle("Unable to connect")
            .setCancelable(false)
            .setPositiveButton("Settings",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {
                            //Intent i = new Intent(Settings.ACTION_SETTINGS);
                            startActivity(new Intent(Settings.ACTION_SETTINGS));
                        }
                    }
            )
            .setNegativeButton("Cancel",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {
                            MainActivity.this.finish();
                        }
                    }
            );
    AlertDialog alert = builder.create();
    alert.show();
}
```



If the user's mobile network or WiFi is not enabled, the user will be prompted to activate it in the device's settings menu.



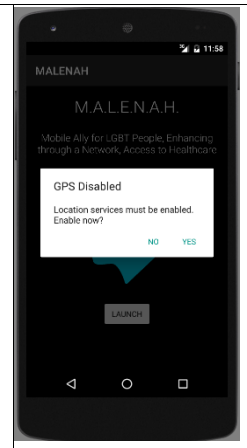
enableInternet() displays an alert to the user to enable their internet connection. Either WiFi or cellular data is accepted by this application.

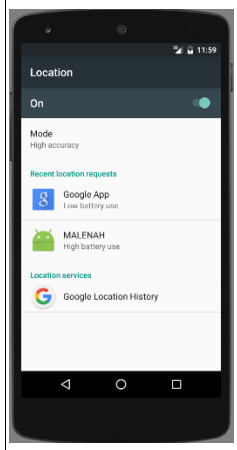
Once internet connectivity is enabled, the user is able to proceed.

2. Checking if Location Services is Enabled

In MainActivity.java, I check to see if location services (either GPS, network) is enabled on the Android device. If not, I display an alert to prompt the user to enable their location services.	**Explanation of code and corresponding UI screens**

```java
protected boolean enableLocation(){
    if(ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 0);
    }
    if (checkLocationPermission()) {
        Log.d("LOCATION (permission)", "user granted permission!");
    } else {
        return false;
    }
    if  ((Build.VERSION.SDK_INT >= 23) &&
            (ContextCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) &&
            (ContextCompat.checkSelfPermission(context, Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED)) {
        return false;
    }
    locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
    gpsEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER); //check GPS status
    networkEnabled = locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER); // check network
status
    if (!gpsEnabled && !networkEnabled){
        Log.d("LOCATION (GPS)", "disabled, ask user to enable!");
        //show dialog to allow user to enable location settings
        AlertDialog.Builder dialog = new AlertDialog.Builder(context);
        dialog.setTitle("GPS Disabled");
        dialog.setMessage("Location services must be enabled. Enable now?").setCancelable(false);

        dialog.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                startActivityForResult(new Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS),
SET_LOCATION_REQUEST);
            }
        });

        dialog.setNegativeButton("No", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) { //do nothing
            }
        });
        dialog.show();
        return false;
    } else if (gpsEnabled) {
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 10, this);
        Log.d("LOCATION (GPS)", LocationManager.GPS_PROVIDER);
    }else if (networkEnabled){
        locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,1000,10,this);
        Log.d("LOCATION (provider)", LocationManager.NETWORK_PROVIDER);
    }
    return true;
}
```





Check if the user has granted permission to use location services on their Android device.
If the user's location services has not been enabled, the user will be prompted to enable it in the location settings.

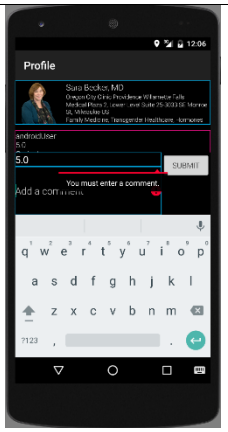
Once the user has enabled location services, he/she can proceed with using the app.

3. Input Validation for POST request

When the user provides empty input, or invalid input (rating < 0.0 or > 5.0), error handling is performed.	Explanation of code and corresponding UI screens

```java
public void onClick(View view) {
    Log.v("EditText", commentEditText.getText().toString());

    /*Check for empty input fields */
    String ratingStr = ratingEditText.getText().toString();
    String commentStr = commentEditText.getText().toString();
    if(TextUtils.isEmpty(ratingStr)){
        ratingEditText.setError("You must enter a rating between 0.0 and 5.0.");
        return;
    }
    if(TextUtils.isEmpty(commentStr)){
        commentEditText.setError("You must enter a comment.");
        return;
    }

    /*Check for invalid input */
    double rating = Double.valueOf(ratingEditText.getText().toString());
    if (rating<0.0 || rating>5.0){
        ratingEditText.setError("Rating must be between 0.0 and 5.0");
        return;
    }

    //POST THIS COMMENT
    Map<String,String> postParams = new LinkedHashMap<>();
    postParams.put("username", "androidUser");
    postParams.put("rating", ratingEditText.getText().toString());
    postParams.put("comment", commentEditText.getText().toString());
    postParams.put("provider", String.valueOf(profile.getId()));
    Log.d("POSTREVIEW","execute async task from ProfileActivity()");
    new PostReviewAsyncTask(ProfileActivity.this, postParams).execute();
    InputMethodManager imm =
(InputMethodManager)getSystemService(Context.INPUT_METHOD_SERVICE);
    imm.hideSoftInputFromWindow((null == getCurrentFocus()) ? null :
getCurrentFocus().getWindowToken(), InputMethodManager.HIDE_NOT_ALWAYS);
    commentEditText.setText("");
}
```

When the user clicks "Submit" with an empty rating field, an error is displayed to the user. An empty rating is considered invalid and cannot be submitted.

		An empty comment field is also considered invalid. The user cannot submit an empty comment field. When the users clicked “Submit” with an empty comment field, an error is displayed.
		A rating of less than 0.0 or greater than 5.0 is considered invalid and cannot be submitted. When the users clicked “Submit” with an invalid rating field, an error is displayed.

Sources

[1] Gay and Lesbian Medical Association. Guidelines for Care of Lesbian, Gay, Bisexual, and Transgender Patients. Gay and Lesbian Medical Association. Web. 23 June 2015. http://www.rainbowwelcome.org/uploads/pdfs/GLMA%20guidelines%202006%20FINAL.pdf

[2] World Health Organization. "Improving the Health and Well-being of Lesbian, Gay, Bisexual and Transgender Persons." EXECUTIVE BOARD 133rd Session Provisional Agenda Item 6.3 6th ser. EB.133 (2013): Improving the Health and Well-being of Lesbian, Gay, Bisexual and Transgender Persons. WHO, 14 May 2013. Web. 23 June 2015. http://www.ghwatch.org/sites/www.ghwatch.org/files/B133-6_LGBT.pdf

[3] GlobalHealth.gov. "Lesbian, Gay, Bisexual, and Transgender Health.” GlobalHealth.gov. Web. 23 June 2015. http://www.globalhealth.gov/global-health-topics/lgbt/

[4] HealthPeople.gov. "Lesbian, Gay, Bisexual, and Transgender Health." Lesbian, Gay, Bisexual, and Transgender Health. HealthyPeople.gov, n.d. Web. 23 June 2015. http://www.healthypeople.gov/2020/topics-objectives/topic/lesbian-gay-bisexual-and-transgender-health

[5] http://developer.android.com/reference/android/os/AsyncTask.html

[6] http://developer.android.com/guide/components/services.html

[7] http://developer.android.com/guide/components/fragments.html

[8] http://developer.android.com/training/implementing-navigation/nav-drawer.html