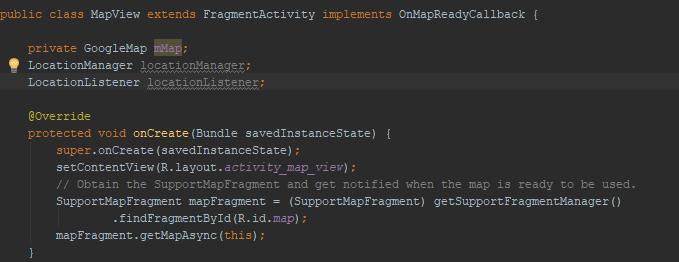
NOTE\* A real phone is required for this project. This tutorial also assumes a knowledge of Sql to build databases and will not cover the database potion.

1. First Create a New Activity with Google Maps.
2. Follow the Link and create your Key
3. Copy your Key to the YOUR\_KEY\_HERE portion of the xml
4. Next go to your map view and create a LocationManager and a LocationListener. Make them outside so that they can be called by all our functions in that class.



1. Don’t touch onCreate, move down to onMapReady
2. First give your LocationManager something

locationManager = (LocationManager) this.getSystemService(Context.*LOCATION\_SERVICE*);

1. Then create a new LocationListener on your listener. Autogenerate the code.

@Override  
public void onMapReady(GoogleMap googleMap) {  
 mMap = googleMap;  
  
 locationManager = (LocationManager) this.getSystemService(Context.*LOCATION\_SERVICE*);  
  
 locationListener = new LocationListener() {  
 @Override  
 public void onLocationChanged(Location location) {  
  
 }  
  
 @Override  
 public void onStatusChanged(String provider, int status, Bundle extras) {  
  
 }  
  
 @Override  
 public void onProviderEnabled(String provider) {  
  
 }  
  
 @Override  
 public void onProviderDisabled(String provider) {  
  
 }  
 };  
}

1. Then we need to check that we have permission to actually use the location data.

//Checks to see android version to ensure the right check is done.  
if(Build.VERSION.*SDK\_INT* < 23){  
 // Doesn't have the better permission check  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 0, locationListener);  
} else {  
 // checks if there is permission and if not asks for it.  
 if (ContextCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*){  
 ActivityCompat.*requestPermissions*(this, new String[]{Manifest.permission.*ACCESS\_FINE\_LOCATION*}, 1);  
 } else {  
 //Otherwise find out where we are and put a marker on the map there.  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 0, locationListener);  
 Location lastKnowLocation = locationManager.getLastKnownLocation(LocationManager.*GPS\_PROVIDER*);  
  
 LatLng currentLocation = new LatLng(lastKnowLocation.getLatitude(), lastKnowLocation.getLongitude());  
 mMap.clear();  
  
 mMap.addMarker(new MarkerOptions().position(currentLocation).title("Current Location"));  
 mMap.moveCamera(CameraUpdateFactory.*newLatLng*(currentLocation)); //moves camera to current location  
 }  
}

1. Now move to just above the onCreate method and start typing onRequestPermissionResults and allow it to generate the code.
2. @Override  
   public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
   }

This is simply an overwrite method for the permission check

1. Now we check if our request code matches the request code for our permission. This allows us varying functionality if there are different permissions given. We then check if we have any permission checks and if they say we have permission. If they do that then we can request a location update

@Override  
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == 1 ){  
 if(grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*){  
 if (ContextCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*) {  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 0, locationListener);  
 }  
 }  
 }  
}  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {

1. Now we add the Locations from the Database List of locations.

final LocationDatabaseHelper dbHelper = new LocationDatabaseHelper(this);  
locationsList = dbHelper.findAllLocations();  
  
for(int i = 0; i < locationsList.size(); i++){  
 ca.dane.dmit.homewifi.LocationModel.Location newLocation = locationsList.get(i);  
 if(newLocation.isActive){  
 LatLng newLocationCoord = new LatLng(newLocation.lat, newLocation.lng);  
  
 mMap.addMarker(new MarkerOptions().position(newLocationCoord).title(newLocation.description));  
 }  
  
  
}

1. This will now list any Locations we have in our database on the map.
2. We also need to add the code to the onLocationChanged method to show us as we move. Currently we draw all the markers each time, this is not ideal but will work for our app.

public void onMapReady(GoogleMap googleMap) {  
 mMap = googleMap;  
  
 locationManager = (LocationManager) this.getSystemService(Context.*LOCATION\_SERVICE*);  
  
 locationListener = new LocationListener() {  
 @Override  
 public void onLocationChanged(Location location) {  
 LatLng currentLocation = new LatLng(location.getLatitude(), location.getLongitude());  
 mMap.clear();  
 mMap.addMarker(new MarkerOptions().position(currentLocation).title("Current Location"));  
 mMap.moveCamera(CameraUpdateFactory.*newLatLng*(currentLocation));  
 final LocationDatabaseHelper dbHelper = new LocationDatabaseHelper(this);  
 locationsList = dbHelper.findAllLocations();  
  
 for(int i = 0; i < locationsList.size(); i++){  
 ca.dane.dmit.homewifi.LocationModel.Location newLocation = locationsList.get(i);  
 if(newLocation.isActive){  
 LatLng newLocationCoord = new LatLng(newLocation.lat, newLocation.lng);  
  
 mMap.addMarker(new MarkerOptions().position(newLocationCoord).title(newLocation.description));  
 }  
  
  
 }  
 }

1. \*\*CAUTION Naming the Location Object Location like I did was foolish as google maps has a Location class as well. This led to naming conflicts which is why you see the long name. I highly recommend using a different name if you decide to implement this.
2. I then build a new activity to display my locations so I can see where they are.
3. Again the populating the listView with the database data will not be covered here but the code that affects the wifi state will be.
4. First we create out WifiManager and our LocationManager and LocationListener again this requires permissions to be asked for, so to get a jump on that we will just overwrite that method now.

LocationManager locationManager;  
LocationListener locationListener;  
WifiManager wifiManager;  
List<ca.dane.dmit.homewifi.LocationModel.Location> locationsList;  
  
double currentLat, currentLong;  
  
@Override  
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
  
 if(grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*){  
 if(ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED* && ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_COARSE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*){  
 return;  
 }  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 0, locationListener);  
 }  
}

1. Then in our on create we give the Managers some data as well as create a WifiInfo class

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_wifi\_toggle*);  
  
 locationManager = (LocationManager) this.getSystemService(Context.*LOCATION\_SERVICE*);  
 wifiManager = (WifiManager) getApplicationContext().getSystemService(Context.*WIFI\_SERVICE*);  
 final WifiInfo wifi = wifiManager.getConnectionInfo();

1. Then we define our location Listener just like last time and let it populate the code

locationListener = new LocationListener() {  
 @Override  
 public void onLocationChanged(android.location.Location location) {  
 }  
  
 @Override  
 public void onStatusChanged(String provider, int status, Bundle extras) {  
  
 }  
  
 @Override  
 public void onProviderEnabled(String provider) {  
  
 }  
  
 @Override  
 public void onProviderDisabled(String provider) {  
  
 }  
 };  
  
 if(Build.VERSION.*SDK\_INT* < 23){  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 0, locationListener);  
 }  
  
 if(ContextCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*){  
 ActivityCompat.*requestPermissions*(this, new String []{Manifest.permission.*ACCESS\_FINE\_LOCATION*}, 1);  
 } else {  
//We’re in here next  
 }  
}

1. Since we know that that does from earlier I will skim it now and fill the else statement.

locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 0, locationListener);  
Location lastKnownLocation = locationManager.getLastKnownLocation(LocationManager.*GPS\_PROVIDER*);  
  
currentLat = lastKnownLocation.getLatitude();  
currentLong = lastKnownLocation.getLongitude();

1. Just like on the map we want the last know location on the onCreate to give us a baseline. We then set our current Lat and Long with that using a double so we can easily pull it from the database.
2. We then scroll back up to our onLocationChanged method. This is where the real work for the wifi is.
3. First we only want to do this check if the wifi is disconnected. Otherwise it will be prone to just turn off the wifi because the location doesn’t match well. So we call our wifi that we created earlier at the beginning of the onCreate method.

if(wifi.getNetworkId() == -1){ //this checks to see if wifi is disconnected

1. Then we iterate through our locations list like on the map. If the location is selected as Active then we check if the current Lat and Long match that locations. If the do the wifi turns on. Otherwise the wifi gets shutdown.

@Override  
public void onLocationChanged(android.location.Location location) {  
 if(wifi.getNetworkId() == -1){  
 for(int i = 0; i < locationsList.size(); i++){  
 ca.dane.dmit.homewifi.LocationModel.Location currentLocation = locationsList.get(i);  
 if(currentLocation.isActive){  
 if(currentLong == currentLocation.lng && currentLat == currentLocation.lat){  
 wifiManager.setWifiEnabled(true);  
 }  
 }  
  
  
 }  
 if(wifiManager.isWifiEnabled() == true){  
 wifiManager.setWifiEnabled(false);  
 }  
  
 }  
}

1. Finally we can’t forget to change our manifest file.

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />

1. Finally if all went well we should have 2 main screens
2. 