Smart Campus Navigation System

Coding

```
<fragment 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:id="@+id/map"
    tools:context=".MapsActivity"
    android:name="com.google.android.gms.maps.SupportMapFragment" />
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:mapbox="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
import android.graphics.BitmapFactory;
import android.os.Bundle;
import java.util.List;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.widget.Toast;

// classes needed to initialize map
import com.mapbox.mapboxsdk.Mapbox;
import com.mapbox.mapboxsdk.maps.MapView;
import com.mapbox.mapboxsdk.maps.MapboxMap;
import com.mapbox.mapboxsdk.maps.Style;
import com.mapbox.mapboxsdk.maps.OnMapReadyCallback;
public class MainActivity extends AppCompatActivity implements
OnMapReadyCallback, MapboxMap.OnMapClickListener, PermissionsListener {
```

```
27. // variables for adding location layer
28. private MapView mapView;
29. private MapboxMap mapboxMap;
30.
31. protected void onCreate(Bundle savedInstanceState) {
32.   super.onCreate(savedInstanceState);
33.   Mapbox.getInstance(this, getString(R.string.access_token));
34.   setContentView(R.layout.activity_main);
35.   mapView = findViewById(R.id.mapView);
36.   mapView.onCreate(savedInstanceState);
37.   mapView.getMapAsync(this);
38. }
39.
40. public void onMapReady(@NonNull final MapboxMap mapboxMap) {
41.   this.mapboxMap = mapboxMap;
42.   mapboxMap.setStyle(getString(R.string.navigation_guidance_day), new
    Style.OnStyleLoaded() {
43.
44.
45.
46.   <com.mapbox.mapboxsdk.maps.MapView
47.       android:id="@+id/mapView"
48.       android:layout_width="match_parent"
49.       android:layout_height="match_parent"
50.       mapbox:mapbox_cameraTargetLat="38.9098"
51.       mapbox:mapbox_cameraTargetLng="-77.0295"
52.       mapbox:mapbox_cameraZoom="12" />
53.
54.   <Button
55.       android:id="@+id/startButton"
56.       android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:background="@color/mapboxGrayLight"
        android:enabled="false"
        android:text="Start navigation"
        android:textColor="@color/mapboxWhite"
        mapbox:layout_constraintStart_toStartOf="parent"
        mapbox:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

  protected void onStart() {
    super.onStart();
    mapView.onStart();
  }

  protected void onResume() {
    super.onResume();
    mapView.onResume();
  }

  protected void onPause() {
    super.onPause();
    mapView.onPause();
  }

  protected void onStop() {
    super.onStop();
```

```
87.     mapView.onStop();
88.   }
89. <color name="colorPrimary">#33C377</color>
90. <color name="colorPrimaryDark">#FF218450</color>
91. <color name="colorAccent">#FF4081</color>
92. <color name="mapboxWhite">#ffffff</color>
93. <color name="mapboxBlue">#4264fb</color>
94. <color name="mapboxGrayLight">#c6d2e1</color>
95. <color name="mapboxPink">#ee4e8b</color>
96. <color name="mapboxYellow">#d9d838</color>
97. <color name="mapboxRed">#b43b71</color>
98. <?xml version="1.0" encoding="utf-8"?>
99. <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
100.         xmlns:mapbox="http://schemas.android.com/apk/res-auto"
101.         android:layout_width="match_parent"
102.         android:layout_height="match_parent">
103.
104.         <com.mapbox.mapboxsdk.maps.MapView
105.             android:id="@+id/mapView"
106.             android:layout_width="match_parent"
107.             android:layout_height="match_parent"
108.             mapbox:mapbox_cameraTargetLat="38.9098"
109.             mapbox:mapbox_cameraTargetLng="-77.0295"
110.             mapbox:mapbox_cameraZoom="12" />
111.
112.         <Button
113.             android:id="@+id/startButton"
114.             android:layout_width="fill_parent"
115.             android:layout_height="wrap_content"
116.             android:layout_marginStart="16dp"
```

```
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:background="@color/mapboxGrayLight"
        android:enabled="false"
        android:text="Start navigation"
        android:textColor="@color/mapboxWhite"
        mapbox:layout_constraintStart_toStartOf="parent"
        mapbox:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
// classes needed to launch navigation UI
import android.view.View;
import android.widget.Button;
import com.mapbox.services.android.navigation.ui.v5.NavigationLauncher;
        button = findViewById(R.id.startButton);
        button.setOnClickListener(new View.OnClickListener() {

          public void onClick(View v) {
            boolean simulateRoute = true;
            NavigationLauncherOptions options = NavigationLauncherOptions.builder()
              .directionsRoute(currentRoute)
              .shouldSimulateRoute(simulateRoute)
              .build();
            // Call this method with Context from within an Activity
            NavigationLauncher.startNavigation(MainActivity.this, options);
          }
        });
```

```
147.    import androidx.appcompat.app.AppCompatActivity;
148.
149.    import android.os.Bundle;
150.
151.    import com.google.android.gms.maps.CameraUpdateFactory;
152.    import com.google.android.gms.maps.3DPoliteknikMap;
153.    import com.google.android.gms.maps.OnMapReadyCallback;
154.    import com.google.android.gms.maps.SupportMapFragment;
155.    import com.google.android.gms.maps.model.LatLng;
156.    import com.google.android.gms.maps.model.MarkerOptions;
157.
158.    public class MapsActivity extends AppCompatActivity implements
    OnMapReadyCallback {
159.
160.        private GoogleMap mMap;
161.
162.        @Override
163.        protected void onCreate(Bundle savedInstanceState) {
164.            super.onCreate(savedInstanceState);
165.            setContentView(R.layout.activity_maps);
166.            // Obtain the SupportMapFragment and get notified when the map is ready to be
    used.
167.            SupportMapFragment mapFragment = (SupportMapFragment)
    getSupportFragmentManager()
168.                    .findFragmentById(R.id.map);
169.            mapFragment.getMapAsync(this);
170.        }
171.
172.
173.            LatLng MERSING= new LatLng(-34, 151);
174.            mMap.addMarker(new MarkerOptions()
```

```
                .position(mersing)
                .title("Marker in mersing"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(mersing));
    }
}

import android.graphics.BitmapFactory;
import android.os.Bundle;
import java.util.List;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.widget.Toast;

// classes needed to initialize map
import com.mapbox.mapboxsdk.Mapbox;
import com.mapbox.mapboxsdk.maps.MapView;
import com.mapbox.mapboxsdk.maps.MapboxMap;
import com.mapbox.mapboxsdk.maps.Style;
import com.mapbox.mapboxsdk.maps.OnMapReadyCallback;

// classes needed to add the location component
import com.mapbox.android.core.permissions.PermissionsListener;
import com.mapbox.android.core.permissions.PermissionsManager;
import com.mapbox.mapboxsdk.location.LocationComponent;
import com.mapbox.mapboxsdk.location.modes.CameraMode;

// classes needed to add a marker
import com.mapbox.geojson.Feature;
import com.mapbox.geojson.Point;
import com.mapbox.mapboxsdk.geometry.LatLng;
```

```
205. import com.mapbox.mapboxsdk.style.layers.SymbolLayer;
206. import com.mapbox.mapboxsdk.style.sources.GeoJsonSource;
207. import static
com.mapbox.mapboxsdk.style.layers.PropertyFactory.iconAllowOverlap;
208. import static
com.mapbox.mapboxsdk.style.layers.PropertyFactory.iconIgnorePlacement;
209. import static com.mapbox.mapboxsdk.style.layers.PropertyFactory.iconImage;
210.
211. // classes to calculate a route
212. import com.mapbox.services.android.navigation.ui.v5.NavigationLauncherOptions;
213. import com.mapbox.services.android.navigation.ui.v5.route.NavigationMapRoute;
214. import com.mapbox.services.android.navigation.v5.navigation.NavigationRoute;
215. import com.mapbox.api.directions.v5.models.DirectionsResponse;
216. import com.mapbox.api.directions.v5.models.DirectionsRoute;
217. import retrofit2.Call;
218. import retrofit2.Callback;
219. import retrofit2.Response;
220. import android.util.Log;
221.
222. // classes needed to launch navigation UI
223. import android.view.View;
224. import android.widget.Button;
225. import com.mapbox.services.android.navigation.ui.v5.NavigationLauncher;
226.
227.
228. public class MainActivity extends AppCompatActivity implements
OnMapReadyCallback, MapboxMap.OnMapClickListener, PermissionsListener {
229.   // variables for adding location layer
230.   private MapView mapView;
231.   private MapboxMap mapboxMap;
232.   // variables for adding location layer
```

```
233.     private PermissionsManager permissionsManager;
234.     private LocationComponent locationComponent;
235.     // variables for calculating and drawing a route
236.     private DirectionsRoute currentRoute;
237.     private static final String TAG = "DirectionsActivity";
238.     private NavigationMapRoute navigationMapRoute;
239.     // variables needed to initialize navigation
240.     private Button button;
241.
242.     @Override
243.     protected void onCreate(Bundle savedInstanceState) {
244.       super.onCreate(savedInstanceState);
245.       Mapbox.getInstance(this, getString(R.string.access_token));
246.       setContentView(R.layout.activity_main);
247.       mapView = findViewById(R.id.mapView);
248.       mapView.onCreate(savedInstanceState);
249.       mapView.getMapAsync(this);
250.     }
251.
252.     @Override
253.     public void onMapReady(@NonNull final MapboxMap mapboxMap) {
254.       this.mapboxMap = mapboxMap;
255.       mapboxMap.setStyle(getString(R.string.navigation_guidance_day), new
    Style.OnStyleLoaded() {
256.         @Override
257.         public void onStyleLoaded(@NonNull Style style) {
258.           enableLocationComponent(style);
259.
260.           addDestinationIconSymbolLayer(style);
261.
262.           mapboxMap.addOnMapClickListener(MainActivity.this);
```

```
263.            button = findViewById(R.id.startButton);
264.            button.setOnClickListener(new View.OnClickListener() {
265.              @Override
266.              public void onClick(View v) {
267.                boolean simulateRoute = true;
268.                NavigationLauncherOptions options = NavigationLauncherOptions.builder()
269.                  .directionsRoute(currentRoute)
270.                  .shouldSimulateRoute(simulateRoute)
271.                  .build();
272.                // Call this method with Context from within an Activity
273.                NavigationLauncher.startNavigation(MainActivity.this, options);
274.              }
275.            });
276.          }
277.        });
278.      }
279.
280.      private void addDestinationIconSymbolLayer(@NonNull Style loadedMapStyle) {
281.        loadedMapStyle.addImage("destination-icon-id",
282.          BitmapFactory.decodeResource(this.getResources(),
    R.drawable.mapbox_marker_icon_default));
283.        GeoJsonSource geoJsonSource = new GeoJsonSource("destination-source-id");
284.        loadedMapStyle.addSource(geoJsonSource);
285.        SymbolLayer destinationSymbolLayer = new SymbolLayer("destination-symbol-
    layer-id", "destination-source-id");
286.        destinationSymbolLayer.withProperties(
287.          iconImage("destination-icon-id"),
288.          iconAllowOverlap(true),
289.          iconIgnorePlacement(true)
290.        );
291.        loadedMapStyle.addLayer(destinationSymbolLayer);
```

```
292.      }
293.
294.      @SuppressWarnings( {"MissingPermission"})
295.      @Override
296.      public boolean onMapClick(@NonNull LatLng point) {
297.
298.        Point destinationPoint = Point.fromLngLat(point.getLongitude(),
    point.getLatitude());
299.        Point originPoint =
    Point.fromLngLat(locationComponent.getLastKnownLocation().getLongitude(),
300.          locationComponent.getLastKnownLocation().getLatitude());
301.
302.        GeoJsonSource source = mapboxMap.getStyle().getSourceAs("destination-source-
    id");
303.        if (source != null) {
304.          source.setGeoJson(Feature.fromGeometry(destinationPoint));
305.        }
306.
307.        getRoute(originPoint, destinationPoint);
308.        button.setEnabled(true);
309.        button.setBackgroundResource(R.color.mapboxBlue);
310.        return true;
311.      }
312.
313.      private void getRoute(Point origin, Point destination) {
314.        NavigationRoute.builder(this)
315.          .accessToken(Mapbox.getAccessToken())
316.          .origin(origin)
317.          .destination(destination)
318.          .build()
319.          .getRoute(new Callback<DirectionsResponse>() {
```

```
320.            @Override
321.            public void onResponse(Call<DirectionsResponse> call,
    Response<DirectionsResponse> response) {
322.              // You can get the generic HTTP info about the response
323.              Log.d(TAG, "Response code: " + response.code());
324.              if (response.body() == null) {
325.                Log.e(TAG, "No routes found, make sure you set the right user and access
    token.");
326.                return;
327.              } else if (response.body().routes().size() < 1) {
328.                Log.e(TAG, "No routes found");
329.                return;
330.              }
331.
332.              currentRoute = response.body().routes().get(0);
333.
334.              // Draw the route on the map
335.              if (navigationMapRoute != null) {
336.                navigationMapRoute.removeRoute();
337.              } else {
338.                navigationMapRoute = new NavigationMapRoute(null, mapView,
    mapboxMap, R.style.NavigationMapRoute);
339.              }
340.              navigationMapRoute.addRoute(currentRoute);
341.            }
342.
343.            @Override
344.            public void onFailure(Call<DirectionsResponse> call, Throwable throwable) {
345.              Log.e(TAG, "Error: " + throwable.getMessage());
346.            }
347.          });
```

```
348.      }
349.
350.      @SuppressWarnings( {"MissingPermission"})
351.      private void enableLocationComponent(@NonNull Style loadedMapStyle) {
352.        // Check if permissions are enabled and if not request
353.        if (PermissionsManager.areLocationPermissionsGranted(this)) {
354.          // Activate the MapboxMap LocationComponent to show user location
355.          // Adding in LocationComponentOptions is also an optional parameter
356.          locationComponent = mapboxMap.getLocationComponent();
357.          locationComponent.activateLocationComponent(this, loadedMapStyle);
358.          locationComponent.setLocationComponentEnabled(true);
359.          // Set the component's camera mode
360.          locationComponent.setCameraMode(CameraMode.TRACKING);
361.        } else {
362.          permissionsManager = new PermissionsManager(this);
363.          permissionsManager.requestLocationPermissions(this);
364.        }
365.      }
366.
367.      @Override
368.      public void onRequestPermissionsResult(int requestCode, @NonNull String[]
    permissions, @NonNull int[] grantResults) {
369.        permissionsManager.onRequestPermissionsResult(requestCode, permissions,
    grantResults);
370.      }
371.
372.      @Override
373.      public void onExplanationNeeded(List<String> permissionsToExplain) {
374.        Toast.makeText(this, R.string.user_location_permission_explanation,
    Toast.LENGTH_LONG).show();
375.      }
```

```
376.
377.     @Override
378.     public void onPermissionResult(boolean granted) {
379.       if (granted) {
380.         enableLocationComponent(mapboxMap.getStyle());
381.       } else {
382.         Toast.makeText(this, R.string.user_location_permission_not_granted,
    Toast.LENGTH_LONG).show();
383.         finish();
384.       }
385.     }
386.
387.     @Override
388.     protected void onStart() {
389.       super.onStart();
390.       mapView.onStart();
391.     }
392.
393.     @Override
394.     protected void onResume() {
395.       super.onResume();
396.       mapView.onResume();
397.     }
398.
399.     @Override
400.     protected void onPause() {
401.       super.onPause();
402.       mapView.onPause();
403.     }
404.
405.     @Override
```

```
406.     protected void onStop() {
407.       super.onStop();
408.       mapView.onStop();
409.     }
410.
411.     @Override
412.     protected void onSaveInstanceState(Bundle outState) {
413.       super.onSaveInstanceState(outState);
414.       mapView.onSaveInstanceState(outState);
415.     }
416.
417.     @Override
418.     protected void onDestroy() {
419.       super.onDestroy();
420.       mapView.onDestroy();
421.     }
422.
423.     @Override
424.     public void onLowMemory() {
425.       super.onLowMemory();
426.       mapView.onLowMemory();
427.     }
428.   }
```

Main activity

```
429.  import android.graphics.BitmapFactory;
430.  import android.os.Bundle;
431.  import java.util.List;
432.  import androidx.annotation.NonNull;
433.  import androidx.appcompat.app.AppCompatActivity;
434.  import android.widget.Toast;
435.
436.  // classes needed to initialize map
437.  import com.mapbox.mapboxsdk.Mapbox;
438.  import com.mapbox.mapboxsdk.maps.MapView;
439.  import com.mapbox.mapboxsdk.maps.MapboxMap;
440.  import com.mapbox.mapboxsdk.maps.Style;
441.  import com.mapbox.mapboxsdk.maps.OnMapReadyCallback;
442.
443.  // classes needed to add the location component
444.  import com.mapbox.android.core.permissions.PermissionsListener;
445.  import com.mapbox.android.core.permissions.PermissionsManager;
446.  import com.mapbox.mapboxsdk.location.LocationComponent;
447.  import com.mapbox.mapboxsdk.location.modes.CameraMode;
448.
449.  // classes needed to add a marker
450.  import com.mapbox.geojson.Feature;
451.  import com.mapbox.geojson.Point;
452.  import com.mapbox.mapboxsdk.geometry.LatLng;
453.  import com.mapbox.mapboxsdk.style.layers.SymbolLayer;
454.  import com.mapbox.mapboxsdk.style.sources.GeoJsonSource;
```

```
455.  import static
      com.mapbox.mapboxsdk.style.layers.PropertyFactory.iconAllowOverlap;
456.  import static
      com.mapbox.mapboxsdk.style.layers.PropertyFactory.iconIgnorePlacement;
457.  import static com.mapbox.mapboxsdk.style.layers.PropertyFactory.iconImage;
458.
459.  // classes to calculate a route
460.  import com.mapbox.services.android.navigation.ui.v5.NavigationLauncherOptions;
461.  import com.mapbox.services.android.navigation.ui.v5.route.NavigationMapRoute;
462.  import com.mapbox.services.android.navigation.v5.navigation.NavigationRoute;
463.  import com.mapbox.api.directions.v5.models.DirectionsResponse;
464.  import com.mapbox.api.directions.v5.models.DirectionsRoute;
465.  import retrofit2.Call;
466.  import retrofit2.Callback;
467.  import retrofit2.Response;
468.  import android.util.Log;
469.
470.  // classes needed to launch navigation UI
471.  import android.view.View;
472.  import android.widget.Button;
473.  import com.mapbox.services.android.navigation.ui.v5.NavigationLauncher;
474.
475.
476.  public class MainActivity extends AppCompatActivity implements
      OnMapReadyCallback, MapboxMap.OnMapClickListener, PermissionsListener {
477.    // variables for adding location layer
478.    private MapView mapView;
479.    private MapboxMap mapboxMap;
480.    // variables for adding location layer
481.    private PermissionsManager permissionsManager;
482.    private LocationComponent locationComponent;
```

```
483.      // variables for calculating and drawing a route
484.      private DirectionsRoute currentRoute;
485.      private static final String TAG = "DirectionsActivity";
486.      private NavigationMapRoute navigationMapRoute;
487.      // variables needed to initialize navigation
488.      private Button button;
489.
490.      @Override
491.      protected void onCreate(Bundle savedInstanceState) {
492.        super.onCreate(savedInstanceState);
493.        Mapbox.getInstance(this, getString(R.string.access_token));
494.        setContentView(R.layout.activity_main);
495.        mapView = findViewById(R.id.mapView);
496.        mapView.onCreate(savedInstanceState);
497.        mapView.getMapAsync(this);
498.      }
499.
500.      @Override
501.      public void onMapReady(@NonNull final MapboxMap mapboxMap) {
502.        this.mapboxMap = mapboxMap;
503.        mapboxMap.setStyle(getString(R.string.navigation_guidance_day), new
    Style.OnStyleLoaded() {
504.          @Override
505.          public void onStyleLoaded(@NonNull Style style) {
506.            enableLocationComponent(style);
507.
508.            addDestinationIconSymbolLayer(style);
509.
510.            mapboxMap.addOnMapClickListener(MainActivity.this);
511.            button = findViewById(R.id.startButton);
512.            button.setOnClickListener(new View.OnClickListener() {
```

```
513.             @Override
514.             public void onClick(View v) {
515.               boolean simulateRoute = true;
516.               NavigationLauncherOptions options = NavigationLauncherOptions.builder()
517.                 .directionsRoute(currentRoute)
518.                 .shouldSimulateRoute(simulateRoute)
519.                 .build();
520.               // Call this method with Context from within an Activity
521.               NavigationLauncher.startNavigation(MainActivity.this, options);
522.             }
523.           });
524.         }
525.       });
526.     }
527.
528.     private void addDestinationIconSymbolLayer(@NonNull Style loadedMapStyle) {
529.       loadedMapStyle.addImage("destination-icon-id",
530.         BitmapFactory.decodeResource(this.getResources(),
    R.drawable.mapbox_marker_icon_default));
531.       GeoJsonSource geoJsonSource = new GeoJsonSource("destination-source-id");
532.       loadedMapStyle.addSource(geoJsonSource);
533.       SymbolLayer destinationSymbolLayer = new SymbolLayer("destination-symbol-
    layer-id", "destination-source-id");
534.       destinationSymbolLayer.withProperties(
535.         iconImage("destination-icon-id"),
536.         iconAllowOverlap(true),
537.         iconIgnorePlacement(true)
538.       );
539.       loadedMapStyle.addLayer(destinationSymbolLayer);
540.     }
541.
```

```
542.      @SuppressWarnings( {"MissingPermission"})
543.      @Override
544.      public boolean onMapClick(@NonNull LatLng point) {
545.
546.        Point destinationPoint = Point.fromLngLat(point.getLongitude(),
    point.getLatitude());
547.        Point originPoint =
    Point.fromLngLat(locationComponent.getLastKnownLocation().getLongitude(),
548.          locationComponent.getLastKnownLocation().getLatitude());
549.
550.        GeoJsonSource source = mapboxMap.getStyle().getSourceAs("destination-source-
    id");
551.        if (source != null) {
552.          source.setGeoJson(Feature.fromGeometry(destinationPoint));
553.        }
554.
555.        getRoute(originPoint, destinationPoint);
556.        button.setEnabled(true);
557.        button.setBackgroundResource(R.color.mapboxBlue);
558.        return true;
559.      }
560.
561.      private void getRoute(Point origin, Point destination) {
562.        NavigationRoute.builder(this)
563.          .accessToken(Mapbox.getAccessToken())
564.          .origin(origin)
565.          .destination(destination)
566.          .build()
567.          .getRoute(new Callback<DirectionsResponse>() {
568.            @Override
569.            public void onResponse(Call<DirectionsResponse> call,
    Response<DirectionsResponse> response) {
```

```
570.             // You can get the generic HTTP info about the response
571.             Log.d(TAG, "Response code: " + response.code());
572.             if (response.body() == null) {
573.               Log.e(TAG, "No routes found, make sure you set the right user and access
    token.");
574.               return;
575.             } else if (response.body().routes().size() < 1) {
576.               Log.e(TAG, "No routes found");
577.               return;
578.             }
579.
580.             currentRoute = response.body().routes().get(0);
581.
582.             // Draw the route on the map
583.             if (navigationMapRoute != null) {
584.               navigationMapRoute.removeRoute();
585.             } else {
586.               navigationMapRoute = new NavigationMapRoute(null, mapView,
    mapboxMap, R.style.NavigationMapRoute);
587.             }
588.             navigationMapRoute.addRoute(currentRoute);
589.           }
590.
591.           @Override
592.           public void onFailure(Call<DirectionsResponse> call, Throwable throwable) {
593.             Log.e(TAG, "Error: " + throwable.getMessage());
594.           }
595.         });
596.     }
597.
598.     @SuppressWarnings( {"MissingPermission"})
```

```
599.      private void enableLocationComponent(@NonNull Style loadedMapStyle) {
600.        // Check if permissions are enabled and if not request
601.        if (PermissionsManager.areLocationPermissionsGranted(this)) {
602.          // Activate the MapboxMap LocationComponent to show user location
603.          // Adding in LocationComponentOptions is also an optional parameter
604.          locationComponent = mapboxMap.getLocationComponent();
605.          locationComponent.activateLocationComponent(this, loadedMapStyle);
606.          locationComponent.setLocationComponentEnabled(true);
607.          // Set the component's camera mode
608.          locationComponent.setCameraMode(CameraMode.TRACKING);
609.        } else {
610.          permissionsManager = new PermissionsManager(this);
611.          permissionsManager.requestLocationPermissions(this);
612.        }
613.      }
614.
615.      @Override
616.      public void onRequestPermissionsResult(int requestCode, @NonNull String[]
    permissions, @NonNull int[] grantResults) {
617.        permissionsManager.onRequestPermissionsResult(requestCode, permissions,
    grantResults);
618.      }
619.
620.      @Override
621.      public void onExplanationNeeded(List<String> permissionsToExplain) {
622.        Toast.makeText(this, R.string.user_location_permission_explanation,
    Toast.LENGTH_LONG).show();
623.      }
624.
625.      @Override
626.      public void onPermissionResult(boolean granted) {
```

```
627.        if (granted) {
628.          enableLocationComponent(mapboxMap.getStyle());
629.        } else {
630.          Toast.makeText(this, R.string.user_location_permission_not_granted,
    Toast.LENGTH_LONG).show();
631.          finish();
632.        }
633.      }
634.
635.      @Override
636.      protected void onStart() {
637.        super.onStart();
638.        mapView.onStart();
639.      }
640.
641.      @Override
642.      protected void onResume() {
643.        super.onResume();
644.        mapView.onResume();
645.      }
646.
647.      @Override
648.      protected void onPause() {
649.        super.onPause();
650.        mapView.onPause();
651.      }
652.
653.      @Override
654.      protected void onStop() {
655.        super.onStop();
656.        mapView.onStop();
```

```
657.     }
658.
659.     @Override
660.     protected void onSaveInstanceState(Bundle outState) {
661.       super.onSaveInstanceState(outState);
662.       mapView.onSaveInstanceState(outState);
663.     }
664.
665.     @Override
666.     protected void onDestroy() {
667.       super.onDestroy();
668.       mapView.onDestroy();
669.     }
670.
671.     @Override
672.     public void onLowMemory() {
673.       super.onLowMemory();
674.       mapView.onLowMemory();
675.     }
676.   }
```