

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
3.   xmlns:tools="http://schemas.android.com/tools">
4.   <uses-permission android:name="android.permission.INTERNET" />
5.   <uses-permission
   android:name="android.permission.ACCESS_FINE_LOCATION" />
6.   <uses-permission
   android:name="android.permission.ACCESS_COARSE_LOCATION" />
7.   <application
8.       android:allowBackup="true"
9.       android:dataExtractionRules="@xml/data_extraction_rules"
10.      android:fullBackupContent="@xml/backup_rules"
11.      android:icon="@mipmap/ic_launcher"
12.      android:label="Travel Buddy"
13.      android:supportsRtl="true"
14.      android:theme="@style/Theme.TravelBuddy"
15.      android:usesCleartextTraffic="true"
16.      tools:targetApi="31">
17.       <meta-data
18.           android:name="com.google.android.geo.API_KEY"
19.           android:value="AIzaSyDvs8kw7EQfAajcmWfMWGRNLd5IgsUEkQP8" />
20.
21.       <activity
22.           android:name=".MainActivity"
23.           android:label="@string/app_name"
24.           android:exported="true"
25.           android:theme="@style/Theme.AppCompat.NoActionBar"
26.           android:windowSoftInputMode="stateHidden|adjustPan"
27.           android:configChanges="orientation|screenSize">
28.           <intent-filter android:priority="100">
29.               <action
30.                   android:name="android.intent.action.MAIN" />
31.               <category
32.                   android:name="android.intent.category.LAUNCHER" />
33.               <category
34.                   android:name="android.intent.category.DEFAULT" />
35.           </intent-filter>
36.       </activity>
37.
38.       <activity
39.           android:name=".ExploreActivity"
40.           android:label="@string/app_name"
41.           android:exported="true"
```

```
39.         android:theme="@style/Theme.AppCompat.NoActionBar"
40.         android:windowSoftInputMode="stateHidden|adjustPan"
41.         android:configChanges="orientation|screenSize">
42.     </activity>
43.
44.     <activity
45.         android:name=".ThingstodoActivity"
46.         android:label="@string/app_name"
47.         android:exported="true"
48.         android:windowSoftInputMode="stateHidden|adjustPan"
49.         android:configChanges="orientation|screenSize">
50.     </activity>
51.
52.     <activity
53.         android:name=".ThingsToDoCategoryAdapter"
54.         android:label="@string/app_name"
55.         android:exported="true"
56.         android:windowSoftInputMode="stateHidden|adjustPan"
57.         android:configChanges="orientation|screenSize">
58.     </activity>
59.
60.     <activity
61.         android:name=".MapRecommendationActivity"
62.         android:label="@string/app_name"
63.         android:exported="true"
64.         android:windowSoftInputMode="stateHidden|adjustPan"
65.         android:configChanges="orientation|screenSize">
66.     </activity>
67.
68.
69. </application>
70.
71. </manifest>
72.
73.
74. <!--     android:theme="@style/Theme.TravelBuddy"-->
75. package com.example.travelbuddy;
76.
77. public class Places {
78.     private String name;
79.     private String address;
80.     private double latitude;
81.
82.     private double longitude;
83.
```

```

84.     public Places(String name, String address, double latitude,
      double longitude) {
85.         this.name = name;
86.         this.address = address;
87.         this.latitude = latitude;
88.         this.longitude = longitude;
89.     }
90.
91.     public String getName() {
92.         return name;
93.     }
94.
95.     public String getAddress() {
96.         return address;
97.     }
98.
99.     public double getLongitude() {
100.         return longitude;
101.     }
102.
103.     public double getLatitude() {
104.         return latitude;
105.     }
106. }
107.
108. package com.example.travelbuddy;
109.
110. import android.content.Context;
111.
112. import io.appwrite.Client;
113. import io.appwrite.services.Account;
114. import io.appwrite.services.Databases;
115.
116.
117. public class AppwriteClientManager {
118.     private static Client client;
119.     private static Account account;
120.     private static Databases database;
121.
122.     public static void initialize(Context context) {
123.         client = new Client(context);
124.         client.setEndpoint("http://10.0.0.172/v1")
125.             .setProject("64136f06c970db619408")
126.             .setSelfSigned(true); // For self-signed
      certificates, only use for development

```

```

127.         account = new Account(client);
128.         database = new Databases(client);
129.     }
130.
131.     public static Client getClient() {
132.         return client;
133.     }
134.     public static Databases getDatabase() {return database;}
135.     public static Account getAccount() {
136.         return account;
137.     }
138.
139.
140. }
141. package com.example.travelbuddy
142.
143. import com.example.travelbuddy.AppwriteClientManager
144. import com.google.gson.Gson
145. import com.google.gson.JsonObject
146. import io.appwrite.Client
147. import io.appwrite.exceptions.AppwriteException
148. import io.appwrite.services.Account
149. import kotlinx.coroutines.CoroutineScope
150. import kotlinx.coroutines.Dispatchers
151. import kotlinx.coroutines.launch
152. import kotlinx.coroutines.withContext
153. import okhttp3.Response
154. import java.io.IOException
155.
156. class AppwriteUserHelper {
157.     private val client: Client =
158.         AppwriteClientManager.getClient()
159.     private val account: Account =
160.         AppwriteClientManager.getAccount()
161.     private var userId: String? = null
162.
163.     init {
164.         CoroutineScope(Dispatchers.IO).launch {
165.             userId = fetchUserId()
166.         }
167.     }
168.
169.     private suspend fun fetchUserId(): String? {
170.         return withContext(Dispatchers.IO) {
171.             try {

```

```

170.             val account =
AppwriteClientManager.getAccount()
171.             val user = account.get()
172.             user.id
173.         } catch (e: AppwriteException) {
174.             e.printStackTrace()
175.             null
176.         }
177.     }
178. }
179.
180. fun getUserId(): String? {
181.     while (userId == null) {
182.         try {
183.             Thread.sleep(100)
184.         } catch (e: InterruptedException) {
185.             e.printStackTrace()
186.         }
187.     }
188.     return userId
189. }
190. }
191. package com.example.travelbuddy;
192.
193. import android.app.Activity;
194. import android.app.AlertDialog;
195. import android.content.Context;
196. import android.view.LayoutInflater;
197. import android.view.View;
198. import android.widget.Button;
199. import android.widget.TextView;
200. import android.widget.Toast;
201.
202. import androidx.annotation.NonNull;
203. import androidx.annotation.Nullable;
204.
205. import com.google.android.gms.maps.GoogleMap;
206. import com.google.android.gms.maps.model.Marker;
207.
208. public class CustomInfoWindowAdapter implements
GoogleMap.InfoWindowAdapter {
209.
210.     private final View mView;
211.     private final LayoutInflater layoutInflater;
212.     private final Context context;

```

```

213.
214.     public CustomInfoWindowAdapter(Context context) {
215.         mView =
            LayoutInflater.from(context).inflate(R.layout.custom_info_window,
            null);
216.         this.context = context;
217.         this.layoutInflater = LayoutInflater.from(context);
218.     }
219.
220.     @Nullable
221.     @Override
222.     public View getInfoWindow(@NonNull Marker marker) {
223.         return null; // Use the default InfoWindow frame
224.     }
225.
226.     @Nullable
227.     @Override
228.     public View getInfoContents(Marker marker) {
229.         View view =
            layoutInflater.inflate(R.layout.custom_info_window, null);
230.
231.         TextView titleView = view.findViewById(R.id.title);
232.         TextView snippetView = view.findViewById(R.id.snippet);
233.
234.         titleView.setText(marker.getTitle());
235.         snippetView.setText(marker.getSnippet());
236.
237.         return view;
238.     }
239. }
240.
241.
242. package com.example.travelbuddy;
243. import androidx.fragment.app.FragmentActivity;
244.
245. import android.os.Bundle;
246. import android.widget.Toast;
247.
248. import com.google.android.gms.maps.CameraUpdateFactory;
249. import com.google.android.gms.maps.GoogleMap;
250. import com.google.android.gms.maps.OnMapReadyCallback;
251. import com.google.android.gms.maps.SupportMapFragment;
252. import com.google.android.gms.maps.model.LatLng;
253. import com.google.android.gms.maps.model.MarkerOptions;
254.

```

```

255. public class ExploreActivity extends FragmentActivity
    implements OnMapReadyCallback {
256.
257.     private GoogleMap mMap;
258.
259.     @Override
260.     protected void onCreate(Bundle savedInstanceState) {
261.         super.onCreate(savedInstanceState);
262.         setContentView(R.layout.activity_explore);
263.         SupportMapFragment mapFragment = (SupportMapFragment)
            getSupportFragmentManager().findFragmentById(R.id.map);
264.         mapFragment.getMapAsync(this);
265.     }
266.
267.     @Override
268.     public void onMapReady(GoogleMap googleMap) {
269.         mMap = googleMap;
270.         // Add a marker in a location and move the camera
271.         LatLng location = new LatLng(-34, 151);
272.         mMap.addMarker(new
            MarkerOptions().position(location).title("Marker in a
            location"));
273.         mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(location, 15));
274.     }
275. }
276. package com.example.travelbuddy;
277.
278. import android.os.Bundle;
279.
280. import androidx.activity.ComponentActivity;
281. import androidx.appcompat.app.AppCompatActivity;
282.
283. import android.util.Log;
284. import android.view.Window;
285. import android.view.WindowManager;
286. import android.widget.EditText;
287. import android.widget.ImageView;
288. import android.widget.TextView;
289. import android.view.View;
290.
291. import java.util.List;
292. import java.util.concurrent.Executor;
293. import java.util.concurrent.Executors;
294.

```

```

295. import io.appwrite.Client;
296. import io.appwrite.exceptions.AppwriteException;
297. import io.appwrite.models.Document;
298. import io.appwrite.services.Account;
299.
300. import io.appwrite.coroutines.CoroutineCallback;
301. import android.widget.Button;
302. import android.widget.Toast;
303. import android.content.Intent;
304. import android.net.Uri;
305. import io.appwrite.Client;
306. import io.appwrite.coroutines.CoroutineCallback;
307. import io.appwrite.services.Account;
308.
309.
310. public class MainActivity extends AppCompatActivity{
311.     public static MainActivity instance;
312.
313.     ImageView imageView;
314.     TextView textView;
315.     private EditText etEmail;
316.     private EditText etPassword;
317.     private Button btnSignUp;
318.     private Button btnSignIn;
319.
320.
321.     @Override
322.     protected void onCreate(Bundle savedInstanceState) {
323.         instance = this;
324.         requestWindowFeature(Window.FEATURE_NO_TITLE);
325.
326.         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
327.             WindowManager.LayoutParams.FLAG_FULLSCREEN);
328.
329.         super.onCreate(savedInstanceState);
330.         setContentView(R.layout.activity_main);
331.         imageView = findViewById(R.id.imageView);
332.         textView = findViewById(R.id.textView);
333.         etEmail = findViewById(R.id.email);
334.         etPassword = findViewById(R.id.password);
335.         btnSignUp = findViewById(R.id.signUpButton);
336.         btnSignIn = findViewById(R.id.signInButton);
337.
338.         // Initialize the Appwrite client manager

```



```

338.         AppwriteClientManager.initialize(this);
339.
340.         btnSignUp.setOnClickListener(v -> signUp());
341.         btnSignIn.setOnClickListener(v -> signIn());
342.
343.
344.
345.     }
346.
347.     private void signUp() {
348.         String email = etEmail.getText().toString().trim();
349.         String password =
            etPassword.getText().toString().trim();
350.
351.         if (email.isEmpty() || password.isEmpty()) {
352.             Toast.makeText(this, "Please fill in all fields",
                Toast.LENGTH_SHORT).show();
353.             return;
354.         }
355.
356.         String[] emailParts = email.split("@");
357.         String userId = emailParts[0];
358.
359.         // Access the Appwrite client and account instances
360.         Client client = AppwriteClientManager.getClient();
361.         Account account = AppwriteClientManager.getAccount();
362.
363.         // Use the list() method to check if the account
            already exists
364.         Executor executor =
            Executors.newSingleThreadExecutor();
365.         executor.execute(() -> {
366.             try {
367.                 // Create a new account
368.                 account.create(
369.                     userId,
370.                     email,
371.                     password,
372.                     new CoroutineCallback<>((result, error)
-> {
373.
374.                         if (error != null) {
375.                             error.printStackTrace();
376.                             return;
377.                         }

```

```

378.                Log.d("Appwrite",
result.toString());
379.                runOnUiThread(() -> {
380.                    Toast.makeText(MainActivity.this, "Successfully signed up!",
Toast.LENGTH_SHORT).show();
381.                });
382.            })
383.
384.        );
385.        } catch (AppwriteException e) {
386.            Log.e("Appwrite", "AppwriteException: " +
e.getMessage());
387.        }
388.    });
389.}
390.
391.
392.    private void signIn() {
393.        String email = etEmail.getText().toString().trim();
394.        String password =
etPassword.getText().toString().trim();
395.
396.        if (email.isEmpty() || password.isEmpty()) {
397.            Toast.makeText(this, "Please fill in all fields",
Toast.LENGTH_SHORT).show();
398.            return;
399.        }
400.
401.        // Access the Appwrite client and account instances
402.        Client client = AppwriteClientManager.getClient();
403.        Account account = AppwriteClientManager.getAccount();
404.
405.        Executor executor =
Executors.newSingleThreadExecutor();
406.        executor.execute(() -> {
407.            account.createEmailSession(
408.                email,
409.                password,
410.                new CoroutineCallback<>((result, error) ->
{
411.                    if (error != null) {
412.                        error.printStackTrace();

```

```

413.                runOnUiThread(() ->
    Toast.makeText(MainActivity.this, "Failed to sign in.",
    Toast.LENGTH_SHORT).show());
414.                return;
415.            }
416.
417.                Log.d("Appwrite", result.toString());
418.                runOnUiThread(() ->
    Toast.makeText(MainActivity.this, "Successfully signed in!",
    Toast.LENGTH_SHORT).show());
419.                // Start ExploreActivity
420.                Intent intent = new
    Intent(MainActivity.this, ThingstodoActivity.class);
421.                startActivity(intent);
422.            })
423.        );
424.    });
425.}
426.
427.}
428. package com.example.travelbuddy;
429. import com.example.travelbuddy.AppwriteUserHelper;
430.
431.
432. import android.Manifest;
433. import android.annotation.SuppressLint;
434. import android.app.AlertDialog;
435. import android.content.Intent;
436. import android.content.pm.PackageManager;
437. import android.location.Location;
438. import android.os.Bundle;
439. import android.util.Log;
440. import android.view.LayoutInflater;
441. import android.view.MenuItem;
442. import android.view.View;
443. import android.widget.Button;
444. import android.widget.TextView;
445. import android.widget.Toast;
446. import android.content.pm.ApplicationInfo;
447.
448. import java.util.UUID;
449.
450. import io.appwrite.Client;
451. import io.appwrite.coroutines.CoroutineCallback;
452. import io.appwrite.exceptions.AppwriteException;

```

```
453.
454. import io.appwrite.services.Account;
455. import io.appwrite.services.Databases;
456.
457.
458.
459.
460. import androidx.annotation.NonNull;
461. import androidx.core.app.ActivityCompat;
462. import androidx.fragment.app.Fragment;
463. import androidx.fragment.app.FragmentActivity;
464. import androidx.fragment.app.FragmentManager;
465.
466. import
    com.google.android.gms.location.FusedLocationProviderClient;
467. import com.google.android.gms.location.LocationServices;
468. import com.google.android.gms.maps.CameraUpdateFactory;
469. import com.google.android.gms.maps.GoogleMap;
470. import com.google.android.gms.maps.OnMapReadyCallback;
471. import com.google.android.gms.maps.SupportMapFragment;
472. import com.google.android.gms.maps.model.LatLng;
473. import com.google.android.gms.maps.model.Marker;
474. import com.google.android.gms.maps.model.MarkerOptions;
475. import com.google.android.gms.tasks.Task;
476.
477. import
    com.google.android.libraries.places.api.model.RectangularBounds;
478. import
    com.google.android.libraries.places.api.model.TypeFilter;
479. import
    com.google.android.libraries.places.api.net.PlacesClient;
480. import
    com.google.android.material.bottomnavigation.BottomNavigationView
    ;
481. import
    com.google.android.material.bottomsheet.BottomSheetDialog;
482. import
    com.google.android.material.navigation.NavigationBarView;
483. import com.google.gson.Gson;
484. import com.google.gson.JsonObject;
485.
486. import org.json.JSONArray;
487. import org.json.JSONException;
488. import org.json.JSONObject;
489.
```

```
490. import java.io.IOException;
491. import java.util.HashMap;
492. import java.util.Map;
493. import java.util.concurrent.atomic.AtomicReference;
494.
495. import okhttp3.Call;
496. import okhttp3.OkHttpClient;
497. import okhttp3.Request;
498. import okhttp3.Response;
499.
500.
501. public class MapRecommendationActivity extends
    FragmentActivity implements OnMapReadyCallback {
502.     private static final String PLACES_API_BASE_URL =
        "https://maps.googleapis.com/maps/api/place/nearbysearch/json";
503.
504.     private Fragment homeFragment;
505.     private Fragment searchFragment;
506.     private String userId;
507.     private AppwriteUserHelper appwriteUserHelper;
508.
509.     private Fragment activeFragment;
510.     private FragmentManager fragmentManager;
511.
512.     private GoogleMap mMap;
513.     private String category;
514.     PlacesClient placesClient;
515.
516.     RectangularBounds bounds;
517.     TypeFilter typeFilter;
518.
519.     @Override
520.     protected void onCreate(Bundle savedInstanceState) {
521.         super.onCreate(savedInstanceState);
522.         setContentView(R.layout.activity_map_recommendation);
523.         AppwriteClientManager.initialize(this);
524.
525.         appwriteUserHelper = new AppwriteUserHelper();
526.         userId = appwriteUserHelper.getUserId();
527.
528.         // Get the intent and the category
529.         Intent intent = getIntent();
530.         category = intent.getStringExtra("category");
531.
```

[illegible]

```

565.             return false;
566.         }
567.     });
568.
569. }
570.
571. private void replaceFragment(Fragment fragment) {
572.     if (activeFragment != fragment) {
573.         fragmentManager.beginTransaction().hide(activeFragment).show(fragment).commit();
574.         activeFragment = fragment;
575.     }
576. }
577.
578. @SuppressWarnings("PotentialBehaviorOverride")
579. @Override
580. public void onMapReady(GoogleMap googleMap) {
581.     mMap = googleMap;
582.
583.     mMap.setInfoWindowAdapter(new
        CustomInfoWindowAdapter(this));
584.
585.     mMap.setOnInfoWindowClickListener(marker -> {
586.         // Show a custom dialog or bottom sheet with
        interactive elements
587.         showBottomSheet(marker);
588.     });
589.
590.
591.     requestLocationPermissions();
592.     // Use the Places API to search for nearby places based
        on the selected category
593.     // and add markers to the map
594. }
595.
596. private void showBottomSheet(Marker marker) {
597.     // Inflate the bottom sheet view
598.     View bottomSheetView =
        LayoutInflater.from(this).inflate(R.layout.bottom_sheet, null);
599.     TextView titleView =
        bottomSheetView.findViewById(R.id.bottom_sheet_title);
600.     TextView snippetView =
        bottomSheetView.findViewById(R.id.bottom_sheet_snippet);

```

```

601.         Button addButton =
            bottomSheetView.findViewById(R.id.add_to_list_button);
602.
603.         titleView.setText(marker.getTitle());
604.         snippetView.setText(marker.getSnippet());
605.
606.         // Set the click listener for the "Add to list" button
607.         addButton.setOnClickListener(v -> {
608.             // Show the AlertDialog when the button is clicked
609.             new
                AlertDialog.Builder(MapRecommendationActivity.this)
610.                .setTitle("Add to list")
611.                .setMessage("Are you sure you want to add
                    '" + marker.getTitle() + "' to your list?")
612.                .setPositiveButton("Yes, add it!", (dialog,
                    which) -> {
613.                    // Call the addPlaceToAppwrite()
                    function to save the place
614.                    try {
615.                        addPlaceToAppwrite(marker.getTitle(), marker.getSnippet(),
                            marker.getPosition().latitude, marker.getPosition().longitude);
616.                    } catch (AppwriteException e) {
617.                        throw new RuntimeException(e);
618.                    }
619.
                    Toast.makeText(MapRecommendationActivity.this, "Added: " +
                        marker.getTitle(), Toast.LENGTH_SHORT).show();
620.                })
621.                .setNegativeButton("No, cancel", null) //
                    No action needed for the negative button
622.                .show();
623.        });
624.
625.        // Show the bottom sheet
626.        BottomSheetDialog bottomSheetDialog = new
            BottomSheetDialog(this);
627.        bottomSheetDialog.setContentView(bottomSheetView);
628.        bottomSheetDialog.show();
629.    }
630.
631.
632.    private void addPlaceToAppwrite(String name, String
        address, double latitude, double longitude) throws
        AppwriteException {

```



```

633.         Client client = AppwriteClientManager.getClient();
634.         Account account = AppwriteClientManager.getAccount();
635.         Databases databases =
            AppwriteClientManager.getDatabase();
636.
637.
638.         // Do something with the userId here
639.         Map<String, Object> placeData = new HashMap<>();
640.         placeData.put("name", name);
641.         placeData.put("address", address);
642.         placeData.put("latitude", latitude);
643.         placeData.put("longitude", longitude);
644.         placeData.put("userId", userId);
645.
646.
647.         String uniqueDocumentId =
            UUID.randomUUID().toString();
648.
649.         databases.createDocument(
650.             "642481c8a9ed2d76e6ef",
651.             "642481d1ba3a9bd5359b",
652.             uniqueDocumentId,
653.             placeData,
654.             new CoroutineCallback<>((result, error) ->
        {
655.                 if (error != null) {
656.                     Log.d("Appwrite", "error" + name +
        address + latitude + longitude);
657.                     error.printStackTrace();
658.                     return;
659.                 }
660.
661.                 Log.d("Appwrite", result.toString());
662.                 Log.d("Appwrite", "Place added
        successfully:");
663.                 })
664.             );
665.
666.     }
667.
668.
669.
670.     private static final int LOCATION_PERMISSION_REQUEST_CODE =
        1;
671.

```

```

672.     private void requestLocationPermissions() {
673.         ActivityCompat.requestPermissions(this,
674.             new
        String[]{Manifest.permission.ACCESS_FINE_LOCATION,
        Manifest.permission.ACCESS_COARSE_LOCATION},
675.             LOCATION_PERMISSION_REQUEST_CODE);
676.     }
677.
678.     @Override
679.     public void onRequestPermissionsResult(int requestCode,
        @NonNull String[] permissions, @NonNull int[] grantResults) {
680.         super.onRequestPermissionsResult(requestCode,
        permissions, grantResults);
681.         if (requestCode == LOCATION_PERMISSION_REQUEST_CODE) {
682.             if (grantResults.length > 0 && grantResults[0] ==
        PackageManager.PERMISSION_GRANTED) {
683.                 getCurrentLocation();
684.             } else {
685.                 Toast.makeText(this, "Location permission
        denied", Toast.LENGTH_SHORT).show();
686.             }
687.         }
688.     }
689.
690.     private FusedLocationProviderClient fusedLocationClient;
691.
692.     private void getCurrentLocation() {
693.         fusedLocationClient =
        LocationServices.getFusedLocationProviderClient(this);
694.
695.         if (ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED
696.             && ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_COARSE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED) {
697.             requestLocationPermissions();
698.             return;
699.         }
700.
701.         Task<Location> locationTask =
        fusedLocationClient.getLastLocation();
702.         locationTask.addOnSuccessListener(location -> {
703.             if (location != null) {
704.                 // Use the user's current location

```

```

705.         double latitude = location.getLatitude();
706.         double longitude = location.getLongitude();
707.         LatLng currentLatLng = new LatLng(latitude,
longitude);
708.
709.         // Add a marker on the user's current location
710.         mMap.addMarker(new
MarkerOptions().position(currentLatLng).title("You are here"));
711.
712.         // Move the camera to the user's current
location
713.         mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(currentLatLng,
15));
714.         // fetchNearbyPlaces (currentLatLng);
715.         try {
716.             getNearbyPlacesBasedOnCategory(latitude,
longitude);
717.         } catch (PackageManager.NameNotFoundException
e) {
718.             throw new RuntimeException(e);
719.         }
720.
721.         } else {
722.             requestLocationPermissions();
723.             Toast.makeText(this, "Unable to get current
location", Toast.LENGTH_SHORT).show();
724.         }
725.     });
726. }
727.
728.     private void getNearbyPlacesBasedOnCategory(double
latitude, double longitude) throws
PackageManager.NameNotFoundException {
729.         LatLng currentLatLng = new LatLng(latitude, longitude);
730.         double radiusInMeters = 5000; // 5 km
731.
732.         // Use the Nearby Search API to search for nearby
restaurants
733.         ApplicationInfo appInfo =
getPackageManager().getApplicationInfo(getPackageName(),
PackageManager.GET_META_DATA);
734.         String apiKey =
appInfo.metaData.getString("com.google.android.geo.API_KEY");

```

```

735.         String url =
            "https://maps.googleapis.com/maps/api/place/nearbysearch/json?loc
            ation=" +
736.             currentLatLng.latitude + "," +
            currentLatLng.longitude +
737.             "&radius=" + radiusInMeters +
738.             "&type=" + category +
739.             "&key=" + apiKey;
740.
741.         OkHttpClient client = new OkHttpClient();
742.         Request request = new
            Request.Builder().url(url).build();
743.
744.         client.newCall(request).enqueue(new okhttp3.Callback()
            {
745.             @Override
746.             public void onFailure(Call call, IOException e) {
747.                 e.printStackTrace();
748.             }
749.
750.             @Override
751.             public void onResponse(Call call, Response
                response) throws IOException {
752.                 if (!response.isSuccessful()) {
753.                     throw new IOException("Unexpected code " +
                        response);
754.                 }
755.
756.                 String responseBody = response.body().string();
757.                 try {
758.                     JSONObject json = new
                        JSONObject(responseBody);
759.                     JSONArray results =
                        json.getJSONArray("results");
760.                     for (int i = 0; i < results.length(); i++)
                        {
761.                         JSONObject result =
                            results.getJSONObject(i);
762.                         JSONObject location =
                            result.getJSONObject("geometry").getJSONObject("location");
763.                         String name = result.getString("name");
764.                         String address =
                            result.getString("vicinity");
765.                         double lat = location.getDouble("lat");
766.                         double lng = location.getDouble("lng");

```

```

767.         LatLng placeLatLng = new LatLng(lat,
        lng);
768.
769.         runOnUiThread(new Runnable() {
770.             @Override
771.             public void run() {
772.                 mMap.addMarker(new
        MarkerOptions()
773.                     .position(placeLatLng)
774.                     .title(name)
775.                     .snippet(address));
776.             }
777.         });
778.     }
779.     } catch (JSONException e) {
780.         e.printStackTrace();
781.     } catch (Exception e) {
782.         throw new RuntimeException(e);
783.     }
784. }
785. });
786. }
787. }
788. package com.example.travelbuddy;
789.
790. public class Places {
791.     private String name;
792.     private String address;
793.     private double latitude;
794.
795.     private double longitude;
796.
797.     public Places(String name, String address, double latitude,
        double longitude) {
798.         this.name = name;
799.         this.address = address;
800.         this.latitude = latitude;
801.         this.longitude = longitude;
802.     }
803.
804.     public String getName() {
805.         return name;
806.     }
807.
808.     public String getAddress() {

```

```

809.         return address;
810.     }
811.
812.     public double getLongitude() {
813.         return longitude;
814.     }
815.
816.     public double getLatitude() {
817.         return latitude;
818.     }
819. }
820. package com.example.travelbuddy;
821.
822. import androidx.annotation.NonNull;
823. import androidx.recyclerview.widget.RecyclerView;
824.
825. import android.view.LayoutInflater;
826. import android.view.ViewGroup;
827. import android.widget.TextView;
828.
829. import
    com.example.travelbuddy.databinding.FragmentItemBinding;
830.
831. import java.util.List;
832.
833. /**
834.  * {@link RecyclerView.Adapter} that can display a {@link
    PlaceholderItem}.
835.  * TODO: Replace the implementation with code for your data
    type.
836.  */
837.
838.
839. public class PlacesItemRecyclerViewAdapter extends
    RecyclerView.Adapter<PlacesItemRecyclerViewAdapter.ViewHolder> {
840.
841.     private final List<Places> mValues;
842.
843.     public PlacesItemRecyclerViewAdapter(List<Places> items) {
844.         mValues = items;
845.     }
846.
847.     @NonNull
848.     @Override

```

```

849.     public ViewHolder onCreateViewHolder(@NonNull ViewGroup
      parent, int viewType) {
850.         FragmentItemBinding binding =
      FragmentItemBinding.inflate(LayoutInflater.from(parent.getContext
      ()), parent, false);
851.         return new ViewHolder(binding);
852.     }
853.
854.     @Override
855.     public void onBindViewHolder(final ViewHolder holder, int
      position) {
856.         Places places = mValues.get(position);
857.         holder.mItem = places;
858.         holder.mIdView.setText(places.getName());
859.         holder.mContentView.setText(places.getAddress());
860.     }
861.
862.     @Override
863.     public int getItemCount() {
864.         return mValues.size();
865.     }
866.
867.     public class ViewHolder extends RecyclerView.ViewHolder {
868.         public final TextView mIdView;
869.         public final TextView mContentView;
870.         public Places mItem;
871.
872.         public ViewHolder(FragmentItemBinding binding) {
873.             super(binding.getRoot());
874.             mIdView = binding.name;
875.             mContentView = binding.address;
876.         }
877.
878.         @Override
879.         public String toString() {
880.             return super.toString() + " '" +
      mContentView.getText() + "'";
881.         }
882.     }
883. }
884. package com.example.travelbuddy;
885.
886. import android.content.Intent;
887. import android.content.pm.ApplicationInfo;
888. import android.content.pm.PackageManager;

```

```
889. import android.os.Bundle;
890. import android.os.Bundle;
891. import android.util.Log;
892. import android.widget.SearchView;
893.
894. import androidx.annotation.NonNull;
895. import androidx.appcompat.app.AppCompatActivity;
896. import com.google.android.gms.common.api.ApiException;
897. import com.google.android.gms.common.api.Status;
898. import com.google.android.gms.maps.model.LatLng;
899. import com.google.android.gms.maps.model.LatLngBounds;
900. import com.google.android.libraries.places.api.Places;
901. import
    com.google.android.libraries.places.api.model.AutoCompletePredict
    ion;
902. import com.google.android.libraries.places.api.model.Place;
903. import
    com.google.android.libraries.places.api.model.RectangularBounds;
904. import
    com.google.android.libraries.places.api.model.TypeFilter;
905. import
    com.google.android.libraries.places.api.net.FindAutocompletePredi
    ctionsRequest;
906. import
    com.google.android.libraries.places.api.net.PlacesClient;
907.
908. import java.util.Arrays;
909. import java.util.List;
910.
911. import androidx.appcompat.app.AppCompatActivity;
912. import androidx.recyclerview.widget.LinearLayoutManager;
913. import androidx.recyclerview.widget.RecyclerView;
914.
915. import com.google.android.gms.maps.SupportMapFragment;
916. import
    com.google.android.libraries.places.widget.AutoCompleteSupportFra
    gment;
917. import
    com.google.android.libraries.places.widget.listener.PlaceSelectio
    nListener;
918.
919. public class ThingstodoActivity extends AppCompatActivity {
920.
921.
922.     PlacesClient placesClient;
```



```

923.     private RecyclerView recyclerView;
924.     private ThingsToDoCategoryAdapter
        thingsToDoCategoryAdapter;
925.     private List<String> categories;
926.
927.     @Override
928.
929.     protected void onCreate(Bundle savedInstanceState) {
930.
931.         super.onCreate(savedInstanceState);
932.         setContentView(R.layout.activity_thingstodo);
933.         try {
934.             ApplicationInfo appInfo =
                getPackageManager().getApplicationInfo(getPackageName(),
                PackageManager.GET_META_DATA);
935.             String apiKey =
                appInfo.metaData.getString("com.google.android.geo.API_KEY");
936.             Places.initialize(getApplicationContext(), apiKey);
937.
938.             placesClient = Places.createClient(this);
939.
940.             final AutocompleteSupportFragment
                autocompleteSupportFragment = (AutocompleteSupportFragment)
                getFragmentManager().findFragmentById(R.id.autocomplete_fr
                    agment);
941.
942.             autocompleteSupportFragment.setPlaceFields(Arrays.asList(Place.Fi
                eld.ID, Place.Field.LAT_LNG, Place.Field.NAME));
943.
                autocompleteSupportFragment.setOnPlaceSelectedListener(new
                PlaceSelectionListener() {
944.                    @Override
945.                    public void onError(@NonNull Status status) {
946.
947.                    }
948.
949.                    @Override
950.                    public void onPlaceSelected(@NonNull Place
                        place) {
951.                        final LatLng latLng = place.getLatLng();
952.                        Log.i("on place selected", "wow" +
                            latLng.latitude);
953.                    }
954.                });

```

```

955.
956.         }
957.         catch (PackageManager.NameNotFoundException e) {
958.             Log.e("TAG", "Failed to load meta-data,
             NameNotFound: " + e.getMessage());
959.         } catch (NullPointerException e) {
960.             Log.e("TAG", "Failed to load meta-data,
             NullPointerException: " + e.getMessage());
961.         }
962.         List<String> categories = Arrays.asList(
963.             "accounting", "airport", "amusement_park",
             "aquarium", "art_gallery", "atm", "bakery", "bank", "bar",
             "beauty_salon", "bicycle_store", "book_store", "bowling_alley",
             "bus_station", "cafe", "campground", "car_dealer", "car_rental",
             "car_repair", "car_wash", "casino", "cemetery", "church",
             "city_hall", "clothing_store", "convenience_store", "courthouse",
             "dentist", "department_store", "doctor", "drugstore",
             "electrician", "electronics_store", "embassy", "fire_station",
             "florist", "funeral_home", "furniture_store", "gas_station",
             "gym", "hair_care", "hardware_store", "hindu_temple",
             "home_goods_store", "hospital", "insurance_agency",
             "jewelry_store", "laundry", "lawyer", "library",
             "light_rail_station", "liquor_store", "local_government_office",
             "locksmith", "lodging", "meal_delivery", "meal_takeaway",
             "mosque", "movie_rental", "movie_theater", "moving_company",
             "museum", "night_club", "painter", "park", "parking",
             "pet_store", "pharmacy", "physiotherapist", "plumber", "police",
             "post_office", "primary_school", "real_estate_agency",
             "restaurant", "roofing_contractor", "rv_park", "school",
             "secondary_school", "shoe_store", "shopping_mall", "spa",
             "stadium", "storage", "store", "subway_station", "supermarket",
             "synagogue", "taxi_stand", "tourist_attraction", "train_station",
             "transit_station", "travel_agency", "university",
             "veterinary_care", "zoo"
964.         );
965.
966.         RecyclerView recyclerView =
             findViewById(R.id.recyclerView);
967.         recyclerView.setLayoutManager(new
             LinearLayoutManager(this));
968.         recyclerView.setHasFixedSize(true); // Set this if your
             RecyclerView items have a fixed size
969.

```

```

970.         ThingsToDoCategoryAdapter adapter = new
           ThingsToDoCategoryAdapter(categories, new
           ThingsToDoCategoryAdapter.OnCategoryClickListener() {
971.             @Override
972.             public void onCategoryClick(int position) {
973.                 String selectedCategory =
           categories.get(position);
974.                 Intent intent = new
           Intent(ThingstodoActivity.this, MapRecommendationActivity.class);
975.                 intent.putExtra("category", selectedCategory);
976.                 ThingstodoActivity.this.startActivity(intent);
977.             }
978.
979.         });
980.         recyclerView.setAdapter(adapter);}
981.
982.
983. }
984.
985. package com.example.travelbuddy;
986.
987. import android.view.LayoutInflater;
988. import android.view.View;
989. import android.view.ViewGroup;
990. import android.widget.TextView;
991.
992. import androidx.annotation.NonNull;
993. import androidx.recyclerview.widget.RecyclerView;
994.
995. import java.util.List;
996.
997. public class ThingsToDoCategoryAdapter extends
           RecyclerView.Adapter<ThingsToDoCategoryAdapter.CategoryViewHolder
           > {
998.
999.     private List<String> categories;
1000.     private OnCategoryClickListener onCategoryClickListener;
1001.
1002.
1003.     public ThingsToDoCategoryAdapter(List<String> categories,
           OnCategoryClickListener onCategoryClickListener) {
1004.         this.categories = categories;
1005.         this.onCategoryClickListener = onCategoryClickListener;
1006.     }
1007.

```

```
1008.     @NonNull
1009.     @Override
1010.     public CategoryViewHolder onCreateViewHolder(@NonNull
        ViewGroup parent, int viewType) {
1011.         View itemView =
            LayoutInflater.from(parent.getContext()).inflate(R.layout.list_it
                em_category, parent, false);
1012.         return new CategoryViewHolder(itemView,
            onCategoryClickListener);
1013.     }
1014.
1015.     @Override
1016.     public void onBindViewHolder(@NonNull CategoryViewHolder
        holder, int position) {
1017.         String categoryName = categories.get(position);
1018.         holder.categoryName.setText(categoryName);
1019.     }
1020.
1021.     @Override
1022.     public int getItemCount() {
1023.         return categories.size();
1024.     }
1025.
1026.     public static class CategoryViewHolder extends
        RecyclerView.ViewHolder implements View.OnClickListener {
1027.
1028.         TextView categoryName;
1029.         OnCategoryClickListener onCategoryClickListener;
1030.
1031.         public CategoryViewHolder(@NonNull View itemView,
            OnCategoryClickListener onCategoryClickListener) {
1032.             super(itemView);
1033.             categoryName =
                itemView.findViewById(R.id.categoryName);
1034.             this.onCategoryClickListener =
                onCategoryClickListener;
1035.             itemView.setOnClickListener(this);
1036.         }
1037.
1038.         @Override
1039.         public void onClick(View v) {
1040.             onCategoryClickListener.onCategoryClick(getAdapterPosition());
1041.         }
1042.     }
```

```

1043.
1044.     public interface OnCategoryClickListener {
1045.         void onCategoryClick(int position);
1046.     }
1047. }
1048. /**
1049.  * Automatically generated file. DO NOT MODIFY
1050.  */
1051. package com.example.travelbuddy;
1052.
1053. public final class BuildConfig {
1054.     public static final boolean DEBUG =
1055.         Boolean.parseBoolean("true");
1056.     public static final String APPLICATION_ID =
1057.         "com.example.travelbuddy";
1058.     public static final String BUILD_TYPE = "debug";
1059.     public static final int VERSION_CODE = 1;
1060.     public static final String VERSION_NAME = "1.0";
1061. }
1062. plugins {
1063.     id 'com.android.application'
1064. }
1065. android {
1066.     namespace 'com.example.travelbuddy'
1067.     compileSdk 33
1068.
1069.     defaultConfig {
1070.         applicationId "com.example.travelbuddy"
1071.         minSdk 24
1072.         targetSdk 33
1073.         versionCode 1
1074.         versionName "1.0"
1075.
1076.         testInstrumentationRunner
1077.             "androidx.test.runner.AndroidJUnitRunner"
1078.     }
1079.
1080.     buildTypes {
1081.         release {
1082.             minifyEnabled false
1083.             proguardFiles
1084.                 getDefaultProguardFile('proguard-android-optimize.txt'),
1085.                 'proguard-rules.pro'
1086.         }
1087.     }
1088. }

```

```
1083.     }
1084.     compileOptions {
1085.         sourceCompatibility JavaVersion.VERSION_1_8
1086.         targetCompatibility JavaVersion.VERSION_1_8
1087.     }
1088.     buildFeatures {
1089.         viewBinding true
1090.     }
1091. }
1092.
1093. dependencies {
1094.
1095.     implementation 'androidx.appcompat:appcompat:1.6.0'
1096.     implementation 'com.google.android.material:material:1.7.0'
1097.     implementation 'androidx.legacy:legacy-support-v4:1.0.0'
1098.     testImplementation 'junit:junit:4.13.2'
1099.     androidTestImplementation 'androidx.test.ext:junit:1.1.5'
1100.     androidTestImplementation
1101.         'androidx.test.espresso:espresso-core:3.5.1'
1102.     implementation("io.appwrite:sdk-for-android:1.2.0")
1103.     implementation
1104.         'androidx.navigation:navigation-fragment:2.4.1'
1105.     implementation 'androidx.navigation:navigation-ui:2.4.1'
1106.     implementation 'com.github.bumptech.glide:glide:3.7.0'
1107.     implementation
1108.         'com.google.android.gms:play-services-maps:17.0.1'
1109.     implementation
1110.         'com.google.maps.android:android-maps-utils:2.2.6'
1111.     implementation
1112.         'com.google.android.libraries.places:places:2.4.0'
1113.     implementation 'com.google.android.material:material:1.4.0'
1114.     implementation 'androidx.recyclerview:recyclerview:1.2.1'
1115.     implementation
1116.         'com.google.android.gms:play-services-location:18.0.0'
1117.     implementation 'com.android.volley:volley:1.2.1'
1118.     implementation
1119.         'androidx.databinding:databinding-runtime:4.0.0'
1120.     implementation
1121.         'androidx.databinding:databinding-adapters:4.0.0'
1122.     implementation 'org.json:json:20210307'
1123.     implementation
1124.         "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
1125.     implementation 'com.squareup.okhttp3:okhttp:4.9.2'
1126.     implementation 'com.google.code.gson:gson:2.8.8'
1127. }
```

```
1119.  
1120.  
1121. }  
1122. apply plugin: 'com.android.application'  
1123. apply plugin: 'com.android.application'  
1124. apply plugin: 'kotlin-android'
```