



## **Module Code & Title**

CS6P05 Final Year Project MAD

Food Share - Android App

Artifact – Google Maps and Firebase push notifications

## **Student Details**

Name: Sita Ram Thing

London Met Id: 22015892

College Id: NP01MA4S220003

Islington College, Kathmandu

24 April 2024

## Contents

1	Introduction .....	4
1.1	Google Map .....	4
1.2	Google mapImplement .....	7
1.3	Firebase cloud messaging set up stapes.....	10
1.3	Implementation of the project.....	15

## List of Figures

Figure 1: View the types of Google services .....	4
Figure 2: Features of Google map .....	5
Figure 3: View the Google Map console.....	6
Figure 4: Google map Api Key access. ....	6
Figure 5: Set the manifest file to have an amp API key .....	7
Figure 6: Set the view model instance.....	7
Figure 7: Repository simple for location .....	8
Figure 8: Give the current location natlang value .....	9
Figure 9: Firebase cloud messaging services. ....	10
Figure 10: Set up the SDK cline app on Android .....	11
Figure 11: The Android project manifest has an edit. ....	11
Figure 12: set up the Firebase messaging design icon and colour .....	12
Figure 13: Set the notification channel .....	12
Figure 14: Notification Permission.....	13
Figure 15: Notification permission set. ....	13
Figure 16: Give the notification permission. ....	14
Figure 17: Firebase messaging event add .....	15
Figure 18: Set up the notification channel icon and colour .....	15
Figure 19: Give the different permissions.....	15
Figure 20: Implement the Firebase messaging service. ....	16
Figure 21: Create the notification channel.....	16
Figure 22: Backend Django have Firebase admin set up.....	17
Figure 23: Firebase could the console have sent the message .....	17

# 1 Introduction

## 1.1 Google Map

Browse help topics

Popular articles	^
Get started with Google Maps	
Download areas & navigate offline	
Find & improve your location's accuracy	
Add, edit, or delete Google Maps reviews & ratings	
Google Maps Timeline	
Discover helpful features in Google Maps	
Maps policies overview	v
Get started with the Google Maps app	v
Search on Maps	v
Get to your destination	v
Explore the map	v
Customize your map	v

Figure 1: View the types of Google services


## Get info about a place

After you find a place on the map you can:

- Get directions to it.
- Get info like business hours and menus.
- Find Street View imagery.

[Learn how to search for places on Google Maps.](#)

## Get directions & start navigation






On a phone or tablet, at the bottom of your map, tap Go . Get travel times and directions to places you might go next, like your home, work, or calendar appointments.

Learn [how to get directions](#) and [start navigation](#).

## Understand Google Maps app features

To help you access features faster, Google Maps app has been updated.

When you open the Google Maps app, you can find 5 tabs at the bottom of the Home Screen:

- **Explore** : Choose where to go.
- **Go** : What to expect along your frequent trips.
- **Saved** : Create lists and recall places.
- **Contribute** : Share experiences, add info and reviews, and fix problems.
- **Updates** : Get notifications for relevant information.

Features like Location sharing, Timeline, and offline maps are available in the top right, in the

Figure 2: Features of Google map

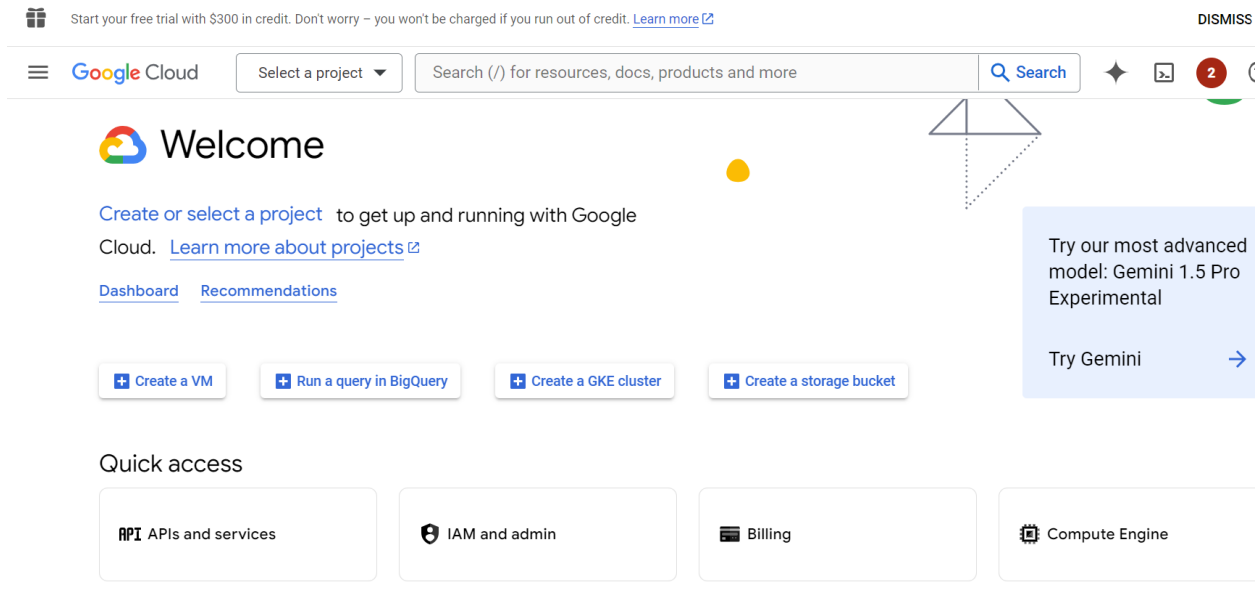


Figure 3: View the Google Map console.

## You need additional access

You need additional access to the project:

• mythic-lead-382501

To request access, contact your project administrator and provide them a copy of the following information:

```
Troubleshooting info:
Principal: NP01MA4S220003@islingtoncollege.edu.np
Resource: mythic-lead-382501
Troubleshooting URL: console.cloud.google.com/iam-admin/troubleshooter;permissions=resourcemanager.

Missing permissions:
resourcemanager.projects.get
```

If your administrator is unable to help, then [contact support](#).

Figure 4: Google map Api Key access.

## 1.2 Google mapImplement

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyDup0LWYQ71XybzV172L5-esHMiDCVCvEI" />
```

Figure 5: Set the manifest file to have an amp API key

```
@HiltViewModel
class GoogleMapViewModel @Inject constructor(private val googleMapUseCase: GoogleMapUseCase) : ViewModel() {

    private val _locationState: MutableStateFlow<LocationState> = MutableStateFlow(LocationState.Loading)
    val viewState = _locationState.asStateFlow()

    Sita Ram Thing
    fun getLocationInfo(
        event: PermissionEvent,
        latitude: Double,
        longitude: Double,
    ) {
        when (event) {
            PermissionEvent.Granted -> {...}

            PermissionEvent.Revoked -> {
                _locationState.value = LocationState.RevokedPermissions
            }
        }
    }
}
```

Figure 6: Set the view model instance

```

/**
 * Repository implementation for managing Google Maps functionality.
 */
class GoogleMapRepositoryImpl @Inject constructor(
    private val context: Context,
    private val locationClient: FusedLocationProviderClient
) : GoogleMapRepository {

    /**
     * Requests location updates based on provided latitude and longitude.
     * @param latitude The latitude coordinate.
     * @param longitude The longitude coordinate.
     * @return A flow emitting the updated location as a LatLng object.
     */
    @SuppressLint("MissingPermission")
    override fun requestLocationUpdates(
        latitude: Double,
        longitude: Double,
    ): Flow<LatLng?> = callbackFlow {

```

Figure 7: Repository simple for location



```

val currentLocation = object : LocationCallback() {
    override fun onLocationResult(locationResult: LocationResult) {
        if (latitude == 0.0 || longitude == 0.0) {
            locationResult.locations.lastOrNull()?.let {
                trySend(LatLng(it.latitude, it.longitude))
            }
        } else {
            trySend(LatLng(latitude, longitude))
        }
    }
}

// Request location updates
locationClient.requestLocationUpdates(
    request,
    currentLocation,
    Looper.getMainLooper()
)

// Remove location updates when the flow is closed
awaitClose {
    locationClient.removeLocationUpdates(currentLocation)
}

```

Figure 8: Give the current location natlang value

## 1.3 Firebase cloud messaging set up stapes

# Firebase Cloud Messaging

[Send feedback](#)

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably send messages at no cost.

Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to drive user re-engagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4096 bytes to a client app.

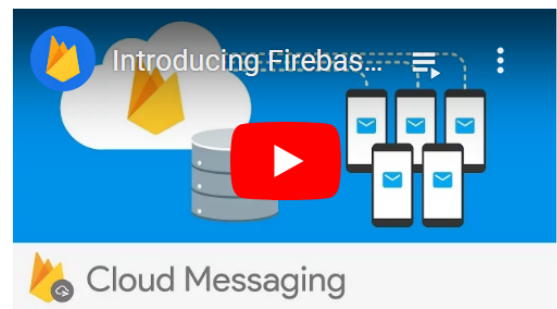
[iOS+ setup](#)[Android setup](#)[Web setup](#)[Flutter setup](#)[C++ setup](#)[Unity setup](#)

Figure 9: Firebase cloud messaging services.

# Set up a Firebase Cloud Messaging client app on Android

[Send feedback](#)

FCM clients require devices running Android 4.4 or higher that also have the Google Play Store app installed, or an emulator running Android 4.4 with Google APIs. Note that you are not limited to deploying your Android apps through Google Play Store.

## Set up the SDK

This section covers tasks you may have completed if you have already enabled other Firebase features for your app. If you haven't already, [add Firebase to your Android project](#)




**Note:** For an optimal experience with FCM, we strongly recommend [enabling Google Analytics](#)  in your project. Google Analytics is a requirement for FCM's [message delivery reporting](#).

Figure 10: Set up the SDK cline app on Android

## Edit your app manifest

Add the following to your app's manifest:

- A service that extends `FirebaseMessagingService`. This is required if you want to do any message handling beyond receiving notifications on apps in the background. To receive notifications in foregrounded apps, to receive data payload, to send upstream messages, and so on, you must extend this service.

```
<service
  android:name=".java.MyFirebaseMessagingService"
  android:exported="false">
  <intent-filter>
    <action android:name="com.google.firebase.MESSAGING_EVENT" />
  </intent-filter>
</service>
```

AndroidManifest.xml 

- (Optional) Within the application component, metadata elements to set a default notification icon and color. Android uses these values whenever incoming messages do not explicitly set icon or color.

Figure 11: Android project manifest has an edit.

- (Optional) Within the application component, metadata elements to set a default notification icon and color. Android uses these values whenever incoming messages do not explicitly set icon or color.



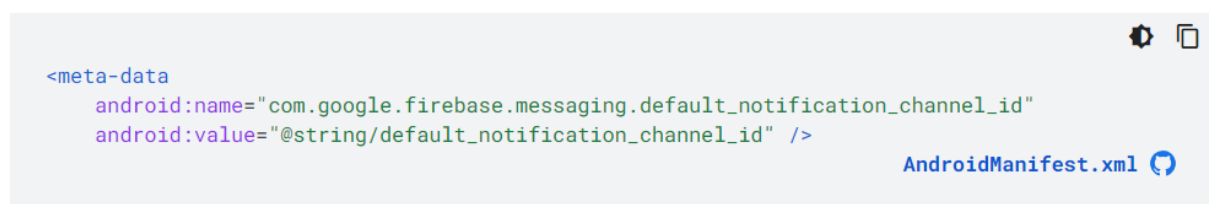
```
<!-- Set custom default icon. This is used when no icon is set for incoming notification messa
See README(https://goo.gl/14GJaQ) for more. -->
<meta-data
    android:name="com.google.firebase.messaging.default_notification_icon"
    android:resource="@drawable/ic_stat_ic_notification" />
<!-- Set color used with incoming notification messages. This is used when no color is set for
notification message. See README(https://goo.gl/6BKBk7) for more. -->
<meta-data
    android:name="com.google.firebase.messaging.default_notification_color"
    android:resource="@color/colorAccent" />
```

AndroidManifest.xml

- (Optional) From Android 8.0 (API level 26) and higher, [notification channels](#) are supported and recommended. FCM provides a default notification channel with basic settings. If you prefer to [create](#) and use your own default channel, set `default_notification_channel_id` to the ID of your notification channel object as shown; FCM will use this value whenever incoming messages do not explicitly set a notification channel. To learn more, see [Manage notification channels](#).

Figure 12: set up the Firebase messaging design icon and colour

- (Optional) From Android 8.0 (API level 26) and higher, [notification channels](#) are supported and recommended. FCM provides a default notification channel with basic settings. If you prefer to [create](#) and use your own default channel, set `default_notification_channel_id` to the ID of your notification channel object as shown; FCM will use this value whenever incoming messages do not explicitly set a notification channel. To learn more, see [Manage notification channels](#).



```
<meta-data
    android:name="com.google.firebase.messaging.default_notification_channel_id"
    android:value="@string/default_notification_channel_id" />
```

AndroidManifest.xml

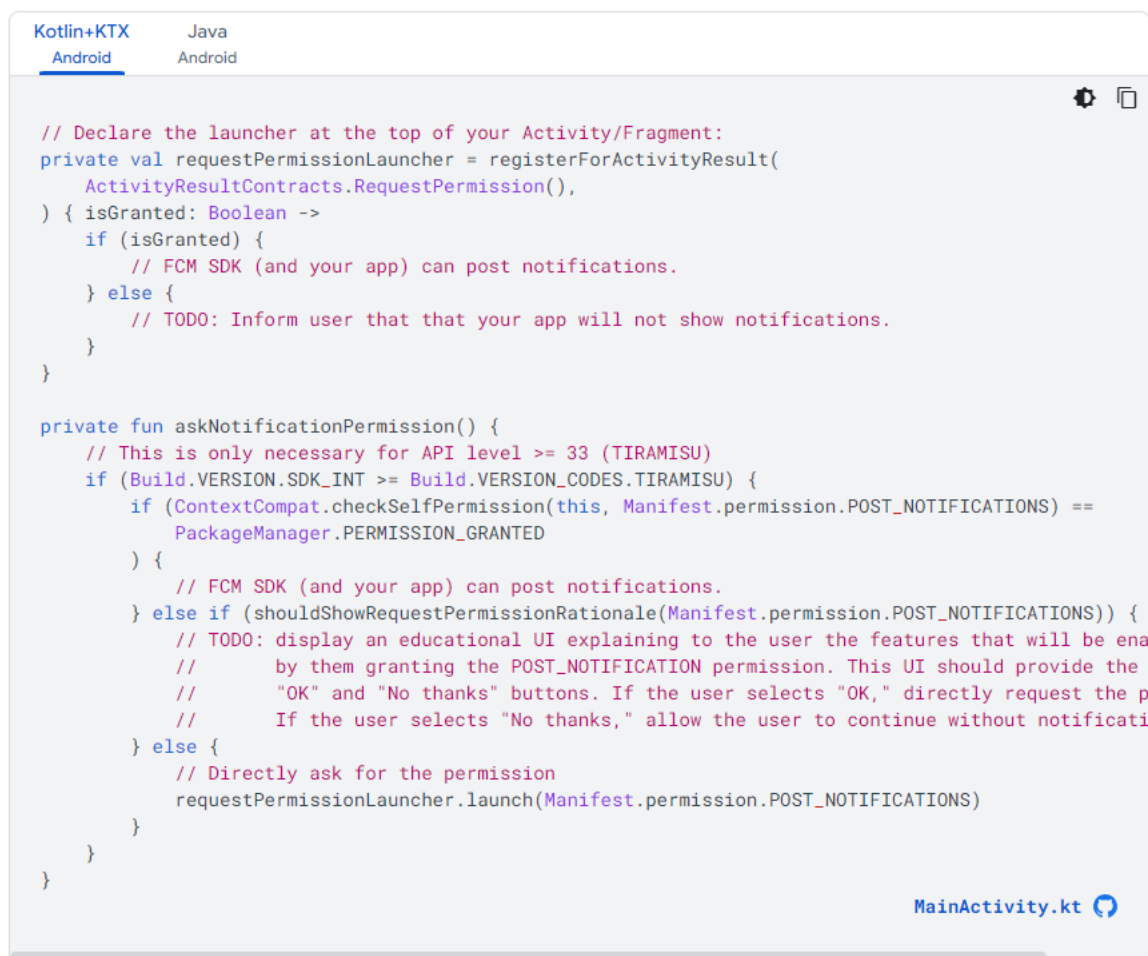
Figure 13: Set the notification channel

## Request runtime notification permission on Android 13+

Android 13 introduces a new runtime permission for showing notifications. This affects all apps running on Android 13 or higher that use FCM notifications.

By default, the FCM SDK (version 23.0.6 or higher) includes the `POST_NOTIFICATIONS` permission defined in the manifest. However, your app will also need to request the runtime version of this permission via the constant, `android.permission.POST_NOTIFICATIONS`. Your app will not be allowed to show notifications until the user has granted this permission.

Figure 14: Notification Permission.



```
Kotlin+KTX      Java
Android        Android

// Declare the launcher at the top of your Activity/Fragment:
private val requestPermissionLauncher = registerForActivityResult(
    ActivityResultContracts.RequestPermission(),
) { isGranted: Boolean ->
    if (isGranted) {
        // FCM SDK (and your app) can post notifications.
    } else {
        // TODO: Inform user that that your app will not show notifications.
    }
}

private fun askNotificationPermission() {
    // This is only necessary for API level >= 33 (TIRAMISU)
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.POST_NOTIFICATIONS) ==
            PackageManager.PERMISSION_GRANTED
        ) {
            // FCM SDK (and your app) can post notifications.
        } else if (shouldShowRequestPermissionRationale(Manifest.permission.POST_NOTIFICATIONS)) {
            // TODO: display an educational UI explaining to the user the features that will be enabled
            // by them granting the POST_NOTIFICATION permission. This UI should provide the
            // "OK" and "No thanks" buttons. If the user selects "OK," directly request the permission
            // If the user selects "No thanks," allow the user to continue without notifications
        } else {
            // Directly ask for the permission
            requestPermissionLauncher.launch(Manifest.permission.POST_NOTIFICATIONS)
        }
    }
}
```

MainActivity.kt

Figure 15: Notification permission set.

See [Notification runtime permission](#) for more best practices on when your app should request the `POST_NOTIFICATIONS` permission from the user.

## Notification permissions for apps targeting Android 12L (API level 32) or lower

Android automatically asks the user for permission the first time your app creates a notification channel, as long as the app is in the foreground. However, there are important caveats regarding the timing of channel creation and permission requests:

- If your app creates its first notification channel when it is running in the background (which the FCM SDK does when receiving an FCM notification), Android will not allow the notification to be displayed and will not prompt the user for the notification permission until the next time your app is opened. This means that *any notifications received before your app is opened and the user accepts the permission will be lost*.
- We strongly recommend that you update your app to target Android 13+ to take advantage of the platform's APIs to request permission. If that is not possible, your app should *create notification channels before you send any notifications to the app in order to trigger the notification permission dialog* and ensure no notifications are lost. See [notification permission best practices](#) for more information.

### Optional: remove `POST_NOTIFICATIONS` permission

By default, the FCM SDK includes the `POST_NOTIFICATIONS` permission. If your app does not use notification messages (whether through FCM notifications, through another SDK, or directly posted by your app) and you don't want your app to include the permission, you can remove it using the [manifest merger's](#) `remove` marker. Keep in mind that removing this permission prevents the display of all notifications, not just FCM notifications. Add the following to your app's manifest file:

```
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" tools:node="remove"/>
```



Figure 16: Give the notification permission.

## 1.3 Implementation of the project

```
<service
    android:name=".features.dashboard.pushNotification.MyFirebaseMessagingService"
    android:exported="false">
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Figure 17: Firebase messaging event add

```
<meta-data
    android:name="com.google.firebase.messaging.default_notification_icon"
    android:resource="@mipmap/ic_launcher_foreground" />
<meta-data
    android:name="com.google.firebase.messaging.default_notification_color"
    android:resource="@color/primary" />
```

Figure 18: Set up the notification channel icon and colour

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_GPS" tools:ignore="SystemPermissionType" />
<uses-permission android:name="android.permission.ACCESS_ASSISTED_GPS" />
<uses-permission android:name="android.permission.ACCESS_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
    android:name="android.permission.CAMERA"
    tools:ignore="PermissionImpliesUnsupportedChromeOsHardware" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
```

Figure 19: Give the different permissions.

```

Sita Ram Thing *
@SuppressLint("MissingFirebaseInstanceTokenRefresh")
class MyFirebaseMessagingService: FirebaseMessagingService() {

    // New fcm access token generated
    Sita Ram Thing
    fun getTokenInstance(): Task<String> {
        return FirebaseMessaging.getInstance().token.addOnCompleteListener { task ->
            if (task.isSuccessful) {
                task.result
            } else {
                task.exception
            }
        }
    }

    // Received notification
    Sita Ram Thing
    override fun onMessageReceived(remoteMessage: RemoteMessage) {
        if (remoteMessage.notification != null) {
            showNotification( title: remoteMessage.notification?.title ?: "", description: remoteMessage.notification?.body ?: "" )
        }
    }
}

```

Figure 20: Implement the Firebase messaging service.

```

private fun showNotification(title: String, description: String) {

    val intent = Intent( packageContext: this, MainActivity::class.java )
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)

    |
    val pendingIntent: PendingIntent? = TaskStackBuilder.create(this).run {
        addNextIntentWithParentStack(intent)
        getPendingIntent( requestCode: 0, flags: PendingIntent.FLAG_UPDATE_CURRENT or PendingIntent.FLAG_IMMUTABLE )
    }

    // Create the notification canal
    val notification = NotificationCompat.Builder( context: this, CHANNEL_ID )
        .setSmallIcon(R.mipmap.ic_launcher_foreground)
        .setContentTitle(title)
        .setContentText(description)
        .setColor(ContextCompat.getColor( context: this, R.color.primary ))
        .setContentIntent(pendingIntent)
        .setAutoCancel(true) // Dismiss the notification when clicked
        .build()

    // Show the notification
    val notificationManager = this.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
    notificationManager.notify( id: 1, notification )
}

```

Figure 21: Create the notification channel.



```

2
3 # Initialize Firebase Admin SDK
4 cred = credentials.Certificate('serviceAccountKey.json')
5 firebase_admin.initialize_app(cred)

```

Figure 22: Backend Django have Firebase admin set up

```

from rest_framework.response import Response
from rest_framework.permissions import AllowAny
from rest_framework.views import APIView
from rest_framework.response import Response
from firebase_admin import messaging

# Test the notification
class SendNotificationView(APIView):
    permission_classes = [AllowAny]

    def post(self, request):
        list_of_tokens = [
            'eeQP7mrTRz2yqkVJJtJ4Jc:APA91bFYbMobbmFtbsuRqUuk7xdh19F-fJDHV7N5aat0TQbkR_YK7Zy0pdT6uMkK91nyMxImX7fwFV5U5c0t6-ILtvdPIf0SAakuCazBIZc-y1T2dwdqcZn9DWJU_7d5'
        ]
        title = 'Title 4 Food Donation'
        body = 'The food is available for donation. Please pick up the food for distribution.'
        return send_notifications(title, body, list_of_tokens)

# Firebase push notification
def send_notifications(title, body, list_of_tokens):
    try:
        # Create the message
        message = messaging.MulticastMessage(
            notification=messaging.Notification(
                title=title,
                body=body,
            ),
            tokens=list_of_tokens,
        )

        # Define headers
        headers = {
            'Content-Type': 'application/json',
            'Authorization': 'AAAAw-NCzFM:APA91bHGq2kJh4m9wBnNEFWIdMfwI_jSCcBxOCSPMDi0Sgvr_A2B8LyQVpPaA5baBVYmGV21d5Q8os0TD0oJVfECNvjke9c0Vqf7T1KHpz6mS0MrNxXfoWU1'
        }
        messaging.send_multicast(message) # send message for multi device
        return Response({'message': 'Notifications sent to all devices', 'is_success': True, 'status': 200})

    except Exception as e:
        return Response({'message': 'Failed to send notifications', 'is_success': False, 'status': 500})

```

Figure 23: Firebase could the console have sent the message