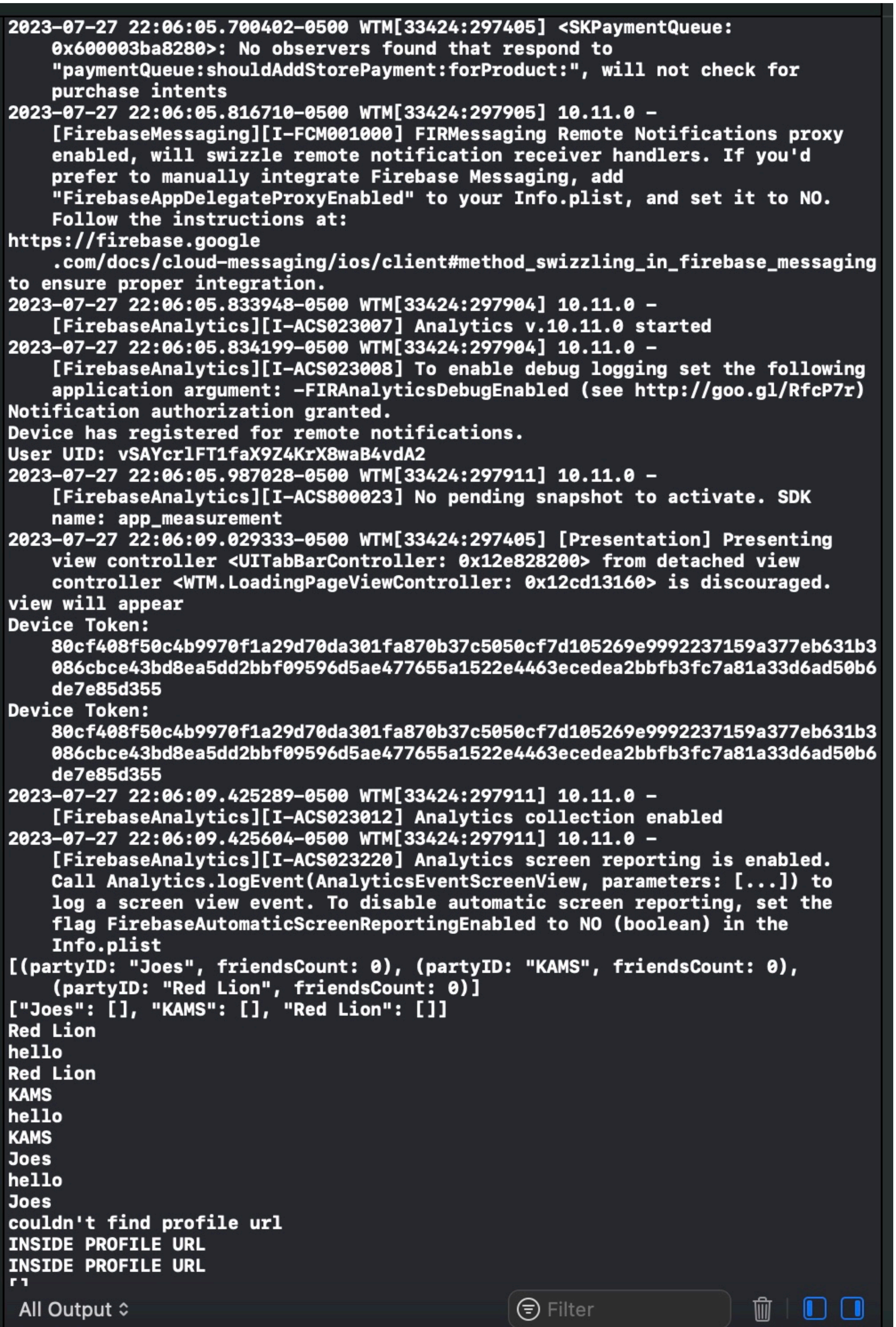
```swift
//
//  AppDelegate.swift
//  WTM
//
//  Created by Nikunj  Tyagi on 1/21/23.
//
//AMAN'S FIRST TEST COMMIT
import UIKit
import Firebase
import FirebaseFirestore
import FirebaseAuth
import GoogleSignIn
import FirebaseCore
import FirebaseMessaging
import UserNotifications

public var partyAccount = false
public var launchedBefore = false
public var authenticated = false
public var user_address = ""
public var currentDate = ""
public var votesLabel = 0
public var barName = ""
public var UID = ""
public var publicOrPriv = true
public var maxPeople = 0

@main
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate,
 UNUserNotificationCenterDelegate {

    func application(_ application: UIApplication,
     didFinishLaunchingWithOptions launchOptions:
     [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        // Override point for customization after application launch.
        FirebaseApp.configure()
        Messaging.messaging().delegate = self
        UNUserNotificationCenter.current().delegate = self
        UNUserNotificationCenter.current().requestAuthorization(options:
         [.alert, .sound, .badge]) { success, _ in
            guard success else {
                return
            }
            print("Notification authorization granted.")
            DispatchQueue.main.async {
                application.registerForRemoteNotifications()
            }
        }
        // Check if the app has a registered device token for remote
         notifications
        if let token = application.currentUserNotificationSettings?.types {
            print("Device has registered for remote notifications.")
        } else {
            print("Device has not registered for remote notifications.")
        }

        application.registerForRemoteNotifications()
        UID = UserDefaults.standard.string(forKey: "UID") ?? ""
        barName = UserDefaults.standard.string(forKey: "barName") ?? ""
        partyAccount = UserDefaults.standard.bool(forKey: "partyAccount")
        launchedBefore = UserDefaults.standard.bool(forKey: "launchedBefore")
        authenticated = UserDefaults.standard.bool(forKey: "authenticated")
        user_address = UserDefaults.standard.string(forKey: "user_address") ??
         "user"
        votesLabel = UserDefaults.standard.integer(forKey: "votesLabel")
        currentDate = UserDefaults.standard.string(forKey: "currentDate") ?? ""

        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "MM/dd/yyyy"
        let newDate = dateFormatter.string(from: Date())

        if newDate != currentDate {
            UserDefaults.standard.setValue(5, forKey: "votesLabel")
            UserDefaults.standard.setValue(newDate, forKey: "currentDate")
        } else {
            UserDefaults.standard.setValue(newDate, forKey: "currentDate")
        }

        if let currentUser = Auth.auth().currentUser {
            let uid = currentUser.uid
            print("User UID:", uid)
            UserDefaults.standard.setValue(uid, forKey: "UID")

        } else {
            print("No user is currently signed in.")
        }



        return true
    }

    func messaging(_ messaging: Messaging, didReceiveRegistrationToken
     fcmToken: String?) {
        // This method is called when the app successfully registers for FCM
         notifications.
        // Store the FCM token in Firestore
        print("Received FCM token:", fcmToken ?? "No token available")

        if let fcmToken = fcmToken, let currentUser = Auth.auth().currentUser {
            let uid = currentUser.uid
        let userRef =
         Firestore.firestore().collection("users").document(uid)
        let data: [String: Any] = [
            "fcmToken": fcmToken,
        ]
        userRef.setData(data, merge: true) { error in
            if let error = error {
                // Handle the error
                print("Error storing FCM token in Firestore:
                 \(error.localizedDescription)")
            } else {
                // Success!
                print("FCM token stored in Firestore.")
            }
        }
    }
}


func userNotificationCenter(_ center: UNUserNotificationCenter,
 willPresent notification: UNNotification) async
    -> UNNotificationPresentationOptions{
        // Customize how the notification should be presented when the app
         is in the foreground.
        // For example, you can show an alert, play a sound, or set the
         badge number.
        return[[.alert, .sound, .badge]]
    }

func userNotificationCenter(_ center: UNUserNotificationCenter, didReceive
 response: UNNotificationResponse, withCompletionHandler
 completionHandler: @escaping () -> Void) {
        // Handle the notification data and perform appropriate actions.
        // For example, you can open a specific screen in your app based
         on the notification data.

        // Get the notification payload data
        let userInfo = response.notification.request.content.userInfo

        // Process the data as needed
        // For example, you can extract information from userInfo
         dictionary and navigate to a specific screen.

        // Call the completion handler after handling the notification
        completionHandler()
    }

func application(_ application: UIApplication,
 didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {
    // This method is called when the app successfully registers for
     remote notifications.
```

```swift
        // You can send the device token to your server if needed.

        // Convert the device token to a string representation (hexadecimal
         format)
        let tokenString = deviceToken.map { String(format: "%02.2hhx", $0)
         }.joined()

        // Print the device token for testing
        print("Device Token: \(tokenString)")

        // Pass the device token to Firebase Cloud Messaging
        Messaging.messaging().apnsToken = deviceToken
    }

    func application(_ application: UIApplication,
     didFailToRegisterForRemoteNotificationsWithError error: Error) {
        // This method is called when there is an error in registering for
         remote notifications.
        // Handle the error as needed.
        print("Failed to register for remote notifications:
         \(error.localizedDescription)")
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting
     connectingSceneSession: UISceneSession, options:
     UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created.
        // Use this method to select a configuration to create the new scene
         with.
        let launchCount = UserDefaults.standard.integer(forKey: "launchCount")
        UserDefaults.standard.set(launchCount + 1, forKey: "launchCount")
        UserDefaults.standard.synchronize()
        return UISceneConfiguration(name: "Default Configuration",
         sessionRole: connectingSceneSession.role)
    }

    func application(_ application: UIApplication, didDiscardSceneSessions
     sceneSessions: Set<UISceneSession>) {
        // Called when the user discards a scene session.
        // If any sessions were discarded while the application was not
         running, this will be called shortly after
         application:didFinishLaunchingWithOptions.
        // Use this method to release any resources that were specific to the
         discarded scenes, as they will not return.
    }
    func application(_ app: UIApplication,
                     open url: URL,
                     options: [UIApplication.OpenURLOptionsKey: Any] = [:]) ->
                      Bool {
```

```
        return GIDSignIn.sharedInstance.handle(url)
    }

}
```

Console next page ->

As the console shows, you can see that the device token is being registered and the device is being authorized for notifications



```
2023-07-27 22:06:05.700402-0500 WTM[33424:297405] <SKPaymentQueue:
    0x600003ba8280>: No observers found that respond to
    "paymentQueue:shouldAddStorePayment:forProduct:", will not check for
    purchase intents
2023-07-27 22:06:05.816710-0500 WTM[33424:297905] 10.11.0 -
    [FirebaseMessaging][I-FCM001000] FIRMessaging Remote Notifications proxy
    enabled, will swizzle remote notification receiver handlers. If you'd
    prefer to manually integrate Firebase Messaging, add
    "FirebaseAppDelegateProxyEnabled" to your Info.plist, and set it to NO.
    Follow the instructions at:
https://firebase.google
    .com/docs/cloud-messaging/ios/client#method_swizzling_in_firebase_messaging
to ensure proper integration.
2023-07-27 22:06:05.833948-0500 WTM[33424:297904] 10.11.0 -
    [FirebaseAnalytics][I-ACS023007] Analytics v.10.11.0 started
2023-07-27 22:06:05.834199-0500 WTM[33424:297904] 10.11.0 -
    [FirebaseAnalytics][I-ACS023008] To enable debug logging set the following
    application argument: -FIRAnalyticsDebugEnabled (see http://goo.gl/RfcP7r)
Notification authorization granted.
Device has registered for remote notifications.
User UID: vSAYcrlFT1faX9Z4KrX8waB4vdA2
2023-07-27 22:06:05.987028-0500 WTM[33424:297911] 10.11.0 -
    [FirebaseAnalytics][I-ACS800023] No pending snapshot to activate. SDK
    name: app_measurement
2023-07-27 22:06:09.029333-0500 WTM[33424:297405] [Presentation] Presenting
    view controller <UITabBarController: 0x12e828200> from detached view
    controller <WTM.LoadingPageViewController: 0x12cd13160> is discouraged.
view will appear
Device Token:
    80cf408f50c4b9970f1a29d70da301fa870b37c5050cf7d105269e9992237159a377eb631b3
    086cbce43bd8ea5dd2bbf09596d5ae477655a1522e4463ecedea2bbfb3fc7a81a33d6ad50b6
    de7e85d355
Device Token:
    80cf408f50c4b9970f1a29d70da301fa870b37c5050cf7d105269e9992237159a377eb631b3
    086cbce43bd8ea5dd2bbf09596d5ae477655a1522e4463ecedea2bbfb3fc7a81a33d6ad50b6
    de7e85d355
2023-07-27 22:06:09.425289-0500 WTM[33424:297911] 10.11.0 -
    [FirebaseAnalytics][I-ACS023012] Analytics collection enabled
2023-07-27 22:06:09.425604-0500 WTM[33424:297911] 10.11.0 -
    [FirebaseAnalytics][I-ACS023220] Analytics screen reporting is enabled.
    Call Analytics.logEvent(AnalyticsEventScreenView, parameters: [...]) to
    log a screen view event. To disable automatic screen reporting, set the
    flag FirebaseAutomaticScreenReportingEnabled to NO (boolean) in the
    Info.plist
[(partyID: "Joes", friendsCount: 0), (partyID: "KAMS", friendsCount: 0),
    (partyID: "Red Lion", friendsCount: 0)]
["Joes": [], "KAMS": [], "Red Lion": []]
Red Lion
hello
Red Lion
KAMS
hello
KAMS
Joes
hello
Joes
couldn't find profile url
INSIDE PROFILE URL
INSIDE PROFILE URL
[]
```