let content = UNMutableNotificationContent()

content.title = "Daily Reminder"

content.body = "Don't forget to check your daily status."

content.sound = UNNotificationSound.default

let trigger = UNTimeIntervalNotificationTrigger(timeInterval: (60 \* 60 \* 24),

repeats: true)

let request = UNNotificationRequest(identifier: "dailyReminder",

content: content, trigger: trigger)

UNUserNotificationCenter.current().add(request) { error in

if let error = error {

print("Error \(error.localizedDescription)")

}

}

func userNotificationCenter(\_ center: UNUserNotificationCenter, willPresent notification: UNNotification, withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void) {

completionHandler([.alert, .sound])

}

func userNotificationCenter(\_ center: UNUserNotificationCenter, didReceive response: UNNotificationResponse, withCompletionHandler completionHandler: @escaping () -> Void) {

if response.actionIdentifier == UNNotificationDefaultActionIdentifier {

// The user swiped to unlock; handle the response

}

completionHandler()

}

content.sound = UNNotificationSound(named: UNNotificationSoundName(rawValue: "customSound.wav"))

let currentBadgeCount = UIApplication.shared.applicationIconBadgeNumber

content.badge = NSNumber(value: currentBadgeCount + 1)

UIApplication.shared.applicationIconBadgeNumber = currentBadgeCount + 1

if let path = Bundle.main.path(forResource: "image", ofType: "png") {

let url = URL(fileURLWithPath: path)

let attachment = try? UNNotificationAttachment(identifier: "image",

url: url, options: nil)

content.attachments = attachment != nil ? [attachment!] : []

}

let action = UNNotificationAction(identifier: "REPLY", title: "Reply", options: [])

let category = UNNotificationCategory(identifier: "MESSAGE", actions: [action],

intentIdentifiers: [], options: [])

UNUserNotificationCenter.current().setNotificationCategories([category])

content.categoryIdentifier = "MESSAGE"

import UserNotifications

UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .badge, .sound]) { granted, error in

// Check for granted permission and handle errors

}

func application(\_ application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {

// Convert token to string and send it to your server

}

UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound, .badge]) { granted, error in

// Check for granted permission and handle errors

if granted {

DispatchQueue.main.async {

UIApplication.shared.registerForRemoteNotifications()

}

}

}

// in AppDelegate

func application(\_ application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {

// Convert device token to string in a more secure way

let tokenParts = deviceToken.map { data in String(format: "%02.2hhx", data) }

let token = tokenParts.joined()

// Send token to server

}

func application(\_ application: UIApplication, didFailToRegisterForRemoteNotificationsWithError error: Error) {

// Handle the error case for remote notification registration

print("Failed to register for remote notifications: \(error)")

}

func application(\_ application: UIApplication, didFailToRegisterForRemoteNotificationsWithError error: Error) {

// Handle registration failure

}

{

"aps": {

"alert": "Your custom message",

"badge": 1,

"sound": "default"

},

"customKey": "customValue"

}

override func didReceive(\_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNotificationContent) -> Void) {

// Example: Modify notification content or add media attachments

contentHandler(modifiedContent)

}

import UserNotifications

func setupMessageCategory() {

let replyAction = UNTextInputNotificationAction(identifier: "reply", title: "Reply", options: [])

let markReadAction = UNNotificationAction(identifier: "markRead", title: "Mark as Read", options: [])

let category = UNNotificationCategory(identifier: "messageCategory", actions: [replyAction, markReadAction], intentIdentifiers: [], hiddenPreviewsBodyPlaceholder: "", options: .customDismissAction)

UNUserNotificationCenter.current().setNotificationCategories([category])

}

// Call this in AppDelegate's didFinishLaunchingWithOptions

setupMessageCategory()

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

// Request Notification Permission

UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound, .badge]) { \_, \_ in }

// Define Actions

let acceptAction = UNNotificationAction(identifier: "ACCEPT", title: "Accept", options: [])

let declineAction = UNNotificationAction(identifier: "DECLINE", title: "Decline", options: [])

// Define Category

let category = UNNotificationCategory(identifier: "INVITE", actions: [acceptAction, declineAction], intentIdentifiers: [], options: [])

UNUserNotificationCenter.current().setNotificationCategories([category])

UNUserNotificationCenter.current().delegate = self

return true

}

func userNotificationCenter(\_ center: UNUserNotificationCenter, didReceive response: UNNotificationResponse, withCompletionHandler completionHandler: @escaping () -> Void) {

switch response.actionIdentifier {

case "ACCEPT":

print("Accepted")

case "DECLINE":

print("Declined")

default:

break

}

completionHandler()

}