TOPIC:

- Local Notifications
- APN and Firebase Notification
- VOIP Notification

Local Notifications:

Local notifications allow an iOS app to display alerts, sounds, and badge updates to notify users even when the app is not in the foreground. Unlike **push notifications**, local notifications are scheduled and triggered by the app itself.

1. Request Notification Permissions

Before scheduling a notification, request permission from the user. Add this to your `AppDelegate.swift` or `ViewController.swift`:

```
import UserNotifications

func requestNotificationPermission() {
    let center = UNUserNotificationCenter.current()
    center.requestAuthorization(options: [.alert, .sound, .badge]) { granted, error in
        if granted {
            print("Permission granted ")
        } else {
            print("Permission denied ")
        }
    }
}
```

 Call `requestNotificationPermission()` in `viewDidLoad()` or an appropriate place.

 Add the **notification entitlement** in `Info.plist`:

```
<key>NSLocalNetworkUsageDescription</key>
<string>We need to send you notifications.</string>
```

2. Schedule a Local Notification

Use `UNUserNotificationCenter` to schedule a notification:

```
func scheduleLocalNotification() {
    let content = UNMutableNotificationContent()
    content.title = "Reminder!"
    content.body = "Don't forget to check the latest updates!"
    content.sound = UNNotificationSound.default
    content.badge = NSNumber(value: 1) // Show badge count on the app icon

    // Trigger after 5 seconds
    let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 5, repeats: false)

    // Create request
    let request = UNNotificationRequest(identifier: "reminder_id", content: content, trigger: trigger)

    // Add to notification center
    UNUserNotificationCenter.current().add(request) { error in
        if let error = error {
            print("Failed to schedule notification: \(error.localizedDescription)")
        } else {
            print("Notification scheduled successfully ")
        }
    }
```

```
22 }
```

- Call `scheduleLocalNotification()` when needed.

3. Handling Notifications When App is in Foreground

By default, notifications won't be shown when the app is in the foreground. To handle them, implement the `UNUserNotificationCenterDelegate`:

```
1  class AppDelegate: UIResponder, UIApplicationDelegate,
   UNUserNotificationCenterDelegate {
2      func application(_ application: UIApplication,
3                       didFinishLaunchingWithOptions
   launchOptions: [UIApplication.LaunchOptionsKey: Any]?) ->
   Bool {
4          UNUserNotificationCenter.current().delegate = self
5          return true
6      }
7
8      // Show notification when app is in foreground
9      func userNotificationCenter(_ center:
   UNUserNotificationCenter,
10                                 willPresent notification:
   UNNotification,
11                                 withCompletionHandler
   completionHandler: @escaping
   (UNNotificationPresentationOptions) -> Void) {
12         completionHandler([.banner, .sound, .badge])
```

```
13     }
14 }
15
```

APNs & Firebase Cloud Messaging (FCM) in iOS

1. Apple Push Notification Service (APNs)

APNs is Apple's native push notification service. It delivers push notifications to iOS, macOS, and watchOS devices.

1.1 Setup APNs in Xcode

1️⃣ Enable Push Notifications in your **App Capabilities**:

- Open **Xcode** → Select your **Target** → Go to **Signing & Capabilities**
- Click **+ Capability** and add **Push Notifications**
- Also, enable **Background Modes** and check **Remote Notifications**

2️⃣ Generate an APNs Key from Apple Developer Portal

- Go to **Apple Developer Account** → Select **Certificates, Identifiers & Profiles**
- Navigate to **Keys** → Click **Create a New Key**
- Enable **APNs** and download the `.p8` file (save the `Key ID`)

3️⃣ Obtain the Device Token
In `AppDelegate.swift`, register for remote notifications:

```
1  import UserNotifications
```

```
2
3  @main
4  class AppDelegate: UIResponder, UIApplicationDelegate {
5      func application(_ application: UIApplication,
6                       didFinishLaunchingWithOptions
   launchOptions: [UIApplication.LaunchOptionsKey: Any]?) ->
   Bool {
7
8
   UNUserNotificationCenter.current().requestAuthorization(optio
   ns: [.alert, .badge, .sound]) { granted, error in
9              if granted {
10                 DispatchQueue.main.async {
11
   application.registerForRemoteNotifications()
12                 }
13             }
14         }
15         return true
16     }
17
18     func application(_ application: UIApplication,
19
   didRegisterForRemoteNotificationsWithDeviceToken deviceToken:
   Data) {
20         let tokenString = deviceToken.map { String(format:
   "%02.2hhx", $0) }.joined()
21         print("APNs Device Token: \(tokenString)")
22     }
23
24     func application(_ application: UIApplication,
25
   didFailToRegisterForRemoteNotificationsWithError error:
   Error) {
26         print("Failed to register for remote notifications:
   \(error.localizedDescription)")
```

```
27     }
28 }
```

 Use the `tokenString` in your backend to send notifications via APNs.

2. Firebase Cloud Messaging (FCM) for iOS

About FCM: https://firebase.google.com/docs/cloud-messaging/concept-options

Integration FCM: https://firebase.google.com/docs/cloud-messaging/ios/client

VOIP Notification:

VoIP (Voice over Internet Protocol) push notifications allow an iOS app to receive **high-priority push notifications** from the server, even when the app is in the background or terminated. They are mainly used for **real-time communication apps**, such as **VoIP calling apps (WhatsApp, FaceTime, Zoom, etc.)**.

Unlike regular APNs push notifications, VoIP notifications: **Wake the app even when terminated**
 Do not show UI banners automatically
 Are used mainly for incoming calls

1️⃣ Steps to Implement VoIP Push Notifications in Swift

Step 1: Enable VoIP Push in Xcode

1. Open **Xcode** → Select your **Target App**
2. Go to **Signing & Capabilities**
3. Click **+ Capability** and **add**:
    - **Push Notifications**
    - **Background Modes** → **Enable "Voice over IP"**

Step 2: Register for VoIP Push Notifications

VoIP push notifications require a **PushKit** framework.

Add PushKit to Your AppDelegate

```swift
import UIKit
import PushKit

@main
class AppDelegate: UIResponder, UIApplicationDelegate, PKPushRegistryDelegate {

    var window: UIWindow?

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

        // Register for VoIP push notifications
        registerForVoIPNotifications()

        return true
    }

    func registerForVoIPNotifications() {
        let voipRegistry = PKPushRegistry(queue: DispatchQueue.main)
        voipRegistry.delegate = self
        voipRegistry.desiredPushTypes = [.voIP]
    }

    // VoIP Token Received
    func pushRegistry(_ registry: PKPushRegistry, didUpdate pushCredentials: PKPushCredentials, for type: PKPushType) {
        let voipToken = pushCredentials.token.map { String(format: "%02.2hhx", $0) }.joined()
```

```
27         print("VoIP Token: \(voipToken)")
28
29         // Send this token to your server to receive VoIP
   pushes
30     }
31
32     // Handle Incoming VoIP Notification
33     func pushRegistry(_ registry: PKPushRegistry,
   didReceiveIncomingPushWith payload: PKPushPayload, for type:
   PKPushType) {
34         print("Received VoIP Push Notification:
   \(payload.dictionaryPayload)")
35
36         // Here, you can trigger a call UI or alert the user
37         // Example: Show CallKit UI for an incoming call
38     }
39 }
```

 This **registers your app** for VoIP notifications and **receives VoIP tokens**.
 The **VoIP token is different from APNs token** and must be sent to your backend.

Step 3: Enable VoIP Push in Apple Developer Portal

1. Go to Apple Developer
2. Navigate to **Certificates, Identifiers & Profiles**
3. Select your **App ID**
4. Enable **Push Notifications** and **VoIP Push Notifications**
5. Create a **VoIP push notification certificate (.p12)**
6. Upload the certificate to your **VoIP server**

Step 4: Send VoIP Push Notification from Server

To send a **VoIP push notification** from your server, use **Apple's APNs VoIP API**.

Step 5: Show Call UI with CallKit

VoIP push notifications **do not show UI alerts** by default. Use **CallKit** to display a call screen.

Show Incoming Call Using CallKit

```
import CallKit

class CallManager {
    let callController = CXCallController()

    func reportIncomingCall(uuid: UUID, handle: String) {
        let provider = CXProvider(configuration:
CXProviderConfiguration(localizedName: "MyApp"))
        provider.setDelegate(self, queue: nil)

        let update = CXCallUpdate()
        update.remoteHandle = CXHandle(type: .generic,
value: handle)
        update.hasVideo = false

        provider.reportNewIncomingCall(with: uuid, update:
update) { error in
            if let error = error {
                print("Error reporting incoming call:
\(error.localizedDescription)")
            }
        }
    }
}

extension CallManager: CXProviderDelegate {
    func providerDidReset(_ provider: CXProvider) {
        print("Call provider reset")
    }
```

```
26 }
```

 This shows the **native iOS call screen** when a VoIP push is received.

6️⃣ Handling App Termination

Even if the app is **killed**, VoIP notifications will wake the app.

1️⃣ When the VoIP push arrives, iOS launches your app in the background.
2️⃣ The **`pushRegistry(_:didReceiveIncomingPushWith:)`** method is called.
3️⃣ You must **trigger CallKit within a few seconds** to avoid the push being ignored.
