Project: Android & iOS SIP application

# Aim

To develop a mobile application for Android and iOS platform using an opensource called linphone

# Problem:

The major problem in any softphone is when the application is in the background mode (screen off state), the OS (iOS mainly) shuts all the abilities of the application in order to receive the call.

There are multiple third party SIP clients available like Zoiper, linphone but the major problem remains the same.

# Solution:

This problem can be solved by setting up a push notification server in order to tell the OS that when an inbound call comes, notification can wake up the SIP application and will show option to accept / reject the call.

# You must know

* What is SIP
* How SIP works
* What is VoIP
* How a SIP call is originated and terminated
* Advanced mobile app development knowledge
* How push notification works (Android and iOS)

# Technical Requirements:

### Android app development environment

* OS: Linux (Debian | Ubuntu)
* Programming language: Kotlin
* Push Notification: Google Firebase

### iPhone App development environment:

* OS: Mac
* Programming language: Swift
* Push Notification: Apple Push Notification Service

Software Requirements Specification

# Introduction

The SIP (Session Initiation Protocol) based softphone application aims to provide robust voice communication on Android and iOS platforms using the Linphone open-source framework. The application will utilize push notifications to enable call reception even when the app is in the background, resolving the issue of OS limitations during inactive states.

# Purpose

The purpose of this document is to outline the detailed requirements and functionalities of the SIP-based softphone application for Android and iOS.

# Scope

The application will allow users to make and receive SIP-based VoIP calls. Key functionalities include:

* SIP account setup and configuration
* Call initiation and termination
* Background call reception through push notifications

# SIP and VoIP Overview

SIP (Session Initiation Protocol): A signaling protocol used for initiating, maintaining, and terminating real-time sessions involving voice and video.

VoIP (Voice over Internet Protocol): Technology enabling voice communication over the internet using packet-switched networks.

# Functional Requirements

### User Authentication and Profile

* Allow users to register and authenticate using SIP credentials.
* Profile settings to manage SIP accounts and configurations.

### Call Management

* Initiate outgoing SIP calls using SIP URI or phone numbers.
* Accept, reject, or terminate incoming calls.
* Ability to switch between audio and video calls.
* Background Call Reception with Push Notifications

### Implement push notification services (Apple Push Notification service (APNs) for iOS, Firebase Cloud Messaging (FCM) for Android).

* Notify the user about incoming calls even when the app is in the background or the device is in sleep mode.
* Wake the application to provide options to accept or reject calls.

### Call Features

* Speakerphone and mute options during calls.
* Call history log for incoming, outgoing, and missed calls.
* Integration with contacts to display caller information.

### Settings and Preferences

* Customizable settings for audio codecs, ringtone, notifications, etc.
* Add account via Credentials, QR-Scan, Login-link
* Support for secure communication using encryption protocols (e.g., TLS/SRTP) for call signaling and media transmission.

# Non-Functional Requirements

### Platform Compatibility

* Support for Android OS (version X and above) and iOS (version X and above).
* Performance
* Efficient resource utilization to minimize battery drain.

### Reliability and Availability

* Maintain high availability and reliability even during network fluctuations.
* Reliable push notification delivery for call wake-up.
* Usability and User Experience

### Intuitive user interface for easy navigation and usage.

* Accessibility features for a diverse user base.
* Security and Compliance

# References

<https://en.wikipedia.org/wiki/Session_Initiation_Protocol>

<https://www.linphone.org/technical-corner/linphone?qt-technical_corner=2#qt-technical_corner>

<https://wiki.linphone.org/xwiki/wiki/public/view/Linphone/>

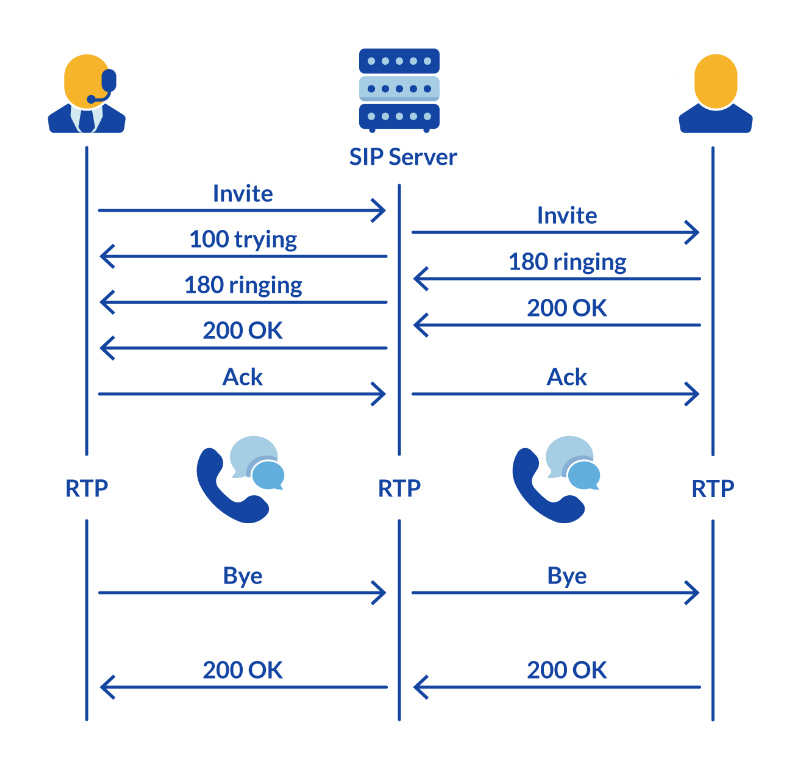
<https://wiki.linphone.org/xwiki/wiki/public/view/Flexisip/>

# Conclusion

The SIP-based softphone application aims to offer seamless voice communication with advanced push notification capabilities, addressing the challenge of background call reception on mobile devices. By leveraging SIP, VoIP, and push notification technologies, this application seeks to provide a reliable and feature-rich communication experience for users on both Android and iOS platforms.

Reference: <https://www.nextiva.com/blog/sip-protocol.html>

## How Does the SIP Protocol Work?



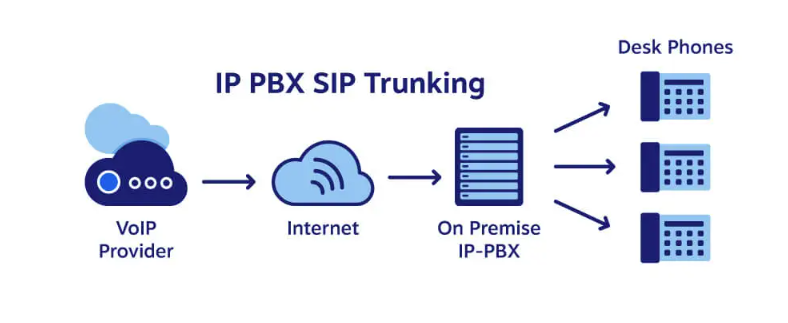
Session Initiation Protocol works with bidirectional communication. For every SIP message, one device sends a request, and the other device receives and later responds.

Responses are coded based on their message. Different preceding numbers in a three-digit sequence have different meanings.

For example, **1xx** response codes mean the device received and is processing the message. Codes starting with **2xx** mean completion, **3xx** is used for redirections, **4xx** is for authentication errors, etc.

The most common code is **200**, meaning the action was completed successfully without further details.

What is the role of a SIP registrar?



While SIP messages can contact another party directly, they usually go through a [**SIP proxy server**](https://www.nextiva.com/blog/sip-proxy-server.html)—kind of like a switchboard.

The SIP server handles SIP requests and directs them to individual users. From there, devices establish trusted communication with each other. Where does a SIP request come from? Most likely, these requests originate from a [**SIP phone**](https://www.nextiva.com/blog/what-is-a-sip-phone.html) or a [**VoIP app**](https://www.nextiva.com/blog/voip-app.html).

Host: pbx2.telxio.com.sg:4144

Credentials:

| S. No | User | Username | Password |
| --- | --- | --- | --- |
| 0 | Sachin | 1017810 | 659d2901ee8e8 |
| 1 | Harsh Aggarwal | 1017811 | 659d2901ee8ec |
| 2 | Yuvraj Singh | 1017812 | 659d2901ee8f0 |
| 3 | Mayank | 1017813 | 659d2901ee8f5 |
| 4 | Kshitij Gupta | 1017814 | 659d2901ee8f9 |
| 5 | Dhananjay Sharma | 1017815 | 659d2901ee8fd |
| 6 | Ritik Chhatwani | 1017816 | 659e5e9630088 |
| 7 | Vishesh Rangwani | 1017817 | 659e5e9630095 |
| 8 | Shreeram Kumar Singh | 1017818 | 659e5e963009c |
| 9 | Yashovardhan Singhal | 1017819 | 659e5e96300a0 |
| 10 | Palak Bhardwaj | 1017820 | 659e5e96300a4 |

User Status

<https://pbx2.telxio.com.sg/data/extension>