**Analysis Doc for SIP Js**

SIP.js is a JavaScript library that provides an implementation of the Session Initiation Protocol (SIP). SIP is a signalling protocol used for initiating, maintaining, and terminating real-time sessions that include voice, video, and messaging applications. SIP.js allows developers to create real-time communication applications directly in the browser using WebRTC (Web Real-Time Communication). **Why Do We Use SIP.js?**1. Real-Time Communication: SIP.js is used to enable real-time voice, video, and messaging

Communication in web applications. It leverages WebRTC to provide a seamless

Communication experience.2. Browser-Based Applications: With SIP.js, developers can create communication

Applications that run entirely in the browser, eliminating the need for additional plugins or

Software installations.3. Ease of Integration: SIP.js simplifies the integration of SIP-based communication into web

Applications, making it easier for developers to add real-time communication features.4. Interoperability: SIP.js works with existing SIP infrastructure and can be used with SIP

Servers, IP-PBX systems, and other SIP-compatible devices. **Advantages of SIP.js**1. Browser Compatibility: SIP.js allows real-time communication directly in the browser,

Leveraging WebRTC, which is supported by most modern browsers.2. Rich Feature Set: Provides features such as call control, media handling, DTMF

(Dual-Tone Multi-Frequency) signalling, and more.3. Standards-Based: Based on the SIP protocol, which is a widely adopted standard for

Real-time communication.4. Open Source: SIP.js is open-source, allowing developers to use, modify, and contribute to

The library freely.5. Community and Support: Being an open-source project, it has a community of developers

Who contribute to its improvement and provide support. **Disadvantages of SIP.js**1. Complexity**:** Implementing SIP-based communication can be complex, especially for

Developers who are not familiar with SIP and WebRTC.2. Browser Limitations: While WebRTC is widely supported, there may be variations in

Implementation across different browsers, leading to compatibility issues.3. Latency and Quality: Real-time communication over the internet can suffer from latency

and quality issues, depending on network conditions.4. Security: Implementing secure communication requires additional considerations, such as

Encryption and authentication, to prevent eavesdropping and other security threats.5. Dependence on SIP Infrastructure: Requires a SIP server or other SIP infrastructure

Components, which may add to the complexity and cost of deployment. **Use Cases of SIP.js**

**-** VoIP (Voice over IP) Applications**\*\*:** Building applications that provide voice calling

Features.**-** Video Conferencing**\*\*:** Enabling video call functionality in web applications.**-** Customer Support**\*\*:** Integrating real-time communication for customer support portals.**-** Messaging Applications**\*\*:** Adding instant messaging features with presence information.

In summary, SIP.js is a powerful tool for adding real-time communication capabilities to web applications, leveraging the SIP protocol and WebRTC. While it offers many benefits, developers must be aware of the complexities and challenges associated with implementing SIP-based communication.**Not send offline notification by SIP.js**

No, SIP.js does not support sending offline notifications. SIP.js is designed for real-time communication over WebRTC, and it relies on active network connections for its operations. For offline notifications, other mechanisms like push notifications through a service worker would be needed.To use SIP (Session Initiation Protocol) for real-time communication, you'll need to set up an account with a SIP provider. This can be for a variety of purposes such as making voice or video calls over IP networks, integrating communication features into your application, or setting up a SIP-based service. Here’s a guide to help you through the process:1. **Choose a SIP Provider** You need to select a SIP provider or service that will manage your SIP accounts. Some

popular SIP providers include:

**-** Twilio **-** Plivo **-** RingCentral **-** OnSIP **-** VoIP.ms These providers offer different features, pricing, and support options, so choose one that

best fits your needs.2. **Sign Up and Create an Account** **-** Visit the Provider’s Website: Go to the website of your chosen SIP provider. **-** Register for an Account: Follow their sign-up process, which typically involves providing

your email address, creating a password, and verifying your email. **-** Select a Plan: Many SIP providers offer different plans based on your usage needs

(e.g., number of users, call volume, etc.). **-** Payment Information: Enter your payment details if applicable.

3. **Configure Your SIP Account** Once you’ve created an account, you’ll need to configure it to use with your SIP client or

Application. This generally involves:

Obtain SIP Credentials: After setting up your account, you’ll receive SIP credentials

including: SIP URI**:** e.g., mailto:sip:username@domain.com Authorization Username**:** The username used to authenticate with the SIP server. Password**:** The password for your SIP account. SIP Server/Proxy**:** The address of the SIP server or proxy you’ll connect to. Configure Your SIP Client**:** Using the SIP credentials, configure your SIP client (e.g., a

SIP phone, a softphone application, or a web application using SIP.js).4. **Test Your Setup -** Make a Test Call: Test your setup by making a call to another SIP address or phone

number. **-** Check Connectivity: Ensure that your SIP client can register with the SIP server and

handle calls properly.

5. **Integration (Optional)** If you’re integrating SIP into an application (e.g., using SIP.js), you’ll need to: **-** Install SIP.js: Include SIP.js in your project via NPM or a CDN. **-** Initialize the SIP User Agent: Set up the SIP user agent with your SIP credentials. **-** Handle SIP Events: Implement event handlers for incoming calls, call states, and other

SIP events.