**# Design Document: letsCollab Web Application**

**## Project Description**

\*\*letsCollab\*\* is a collaborative web application designed to facilitate seamless online collaboration, combining the power of the MERN (MongoDB, Express.js, React, Node.js) stack with Real-Time Communication (WebRTC) and Socket.IO for efficient video conferencing. The platform aims to provide users with a feature-rich environment for collaborative work, enabling real-time discussions, file sharing, and an immersive video conferencing experience.

**## How WebRTC Works**

**### Peer-to-Peer Communication**

WebRTC (Web Real-Time Communication) is a powerful technology for enabling real-time communication directly between web browsers. The primary components of WebRTC include:

**1. \*\*MediaStream:\*\***

- Represents a continuous stream of media content, such as audio or video.

- Obtained from local audio/video sources or remote peer connections.

**2. \*\*RTCPeerConnection:\*\***

- Establishes and manages the peer-to-peer connection.

- Handles negotiation and establishment of the connection.

- Manages the exchange of audio, video, and data streams.

**3. \*\*RTCDataChannel:\*\***

- Enables bidirectional communication of arbitrary data between peers.

**4. \*\*Signaling Server:\*\***

- Manages the exchange of metadata required to set up and manage the connection.

- Not involved in the actual transmission of media.

The WebRTC workflow in letsCollab involves establishing peer connections, exchanging session descriptions, and handling the streaming of audio, video, and data directly between clients.

**## How Socket.IO Works**

**### Real-Time Communication with Socket.IO**

Socket.IO is used for real-time bidirectional event-based communication. In the context of letsCollab, Socket.IO serves as the bridge for real-time messaging and signaling between clients. Key features include:

**1. \*\*Event Handling:\*\***

- Clients can emit events and listen for events.

- Events are used for actions like joining a room, sending messages, and notifying users about participants' activities.

**2. \*\*Rooms:\*\***

- Socket.IO allows users to join specific rooms.

- Rooms are utilized to segregate communication channels, facilitating group-specific interactions.

**3. \*\*Broadcasting:\*\***

- Broadcasting enables sending messages to all clients or specific groups.

- Useful for notifying users about changes in the collaborative space.

**4. \*\*Acknowledgments:\*\***

- Clients can acknowledge receipt of events, ensuring reliable communication.

The combination of WebRTC and Socket.IO in letsCollab provides a comprehensive real-time communication infrastructure, enabling seamless video conferencing and collaboration among users.

**1. \*\*Technology Stack\*\***

- Frontend (React.js)

- Backend (Node.js and Express.js)

- Database (MongoDB , Mongoose)

- Real-Time Communication (Socket.IO and WebRTC)

**2. \*\*Features and Functionalities\*\***

- User Authentication

- User Profile Management

- Create and Join Collaborative Spaces

- Real-Time Video Conferencing

- Chat Messaging

**3. \*\*WebRTC Integration\*\***

- Peer-to-Peer Communication

- Media Streams Handling

- Signaling Server

**4. \*\*Socket.IO Integration\*\***

- Real-Time Communication

- Event Handling

- Room Management

**5. \*\*Security\*\***

- User Authentication (JWT)

**### Connection Diagram**

