Chatroom: WebRTC Based Voice Rooms

Prof. (Dr.) H. D. Kale[1], Hiten Jaiswal[2], Mahima Ilpache[3], Gaurav Kalbonde[4], Varad Kaple[5]
[1]*Assistant Professor, Department of Information Technology, PRMIT&R, Amravati, Maharashtra, India*
[2,3,4,5]*Students, Department of Information Technology, PRMIT&R, Amravati, Maharashtra, India*

***Abstract*— This paper introduces Chatroom, a lightweight, browser-based audio communication platform designed to operate seamlessly without requiring any software installation. By focusing exclusively on voice interaction, the system provides an efficient, low-bandwidth alternative to traditional video conferencing tools, significantly improving accessibility in environments with poor network conditions. At its core, the platform leverages WebRTC (Web Real-Time Communication) to establish direct peer-to-peer audio streams. This decentralized approach minimizes latency by bypassing centralized media servers, while ensuring robust security through built-in encryption protocols like Secure Real-Time Transport Protocol (SRTP) and Datagram Transport Layer Security (DTLS). To balance user convenience with strict security, the system employs a password less authentication model utilizing Twilio's OTP services. User sessions are managed via JSON Web Tokens (JWTs) specifically, one-hour access tokens and one-year refresh tokens which are securely stored within HTTP-only cookies to prevent client-side vulnerabilities. The platform also features a versatile access control framework, offering Open (public), Social (mutual connections only), and Private (password-protected) rooms. Within these spaces, a role-based hierarchy of Moderators, Speakers, and Listeners ensures orderly interactions, giving moderators the administrative power to manage users and scale room capacity from 2 to 100 participants. Beyond core voice functionality, Chat Room enhances user engagement with real-time text chat, emoji reactions, live session timers, and a "raise hand" feature. A built-in social layer further allows users to customize profiles, build follower networks, and receive instant notifications. Developed using a modern full stack architecture, the application integrates a React.js and Redux Toolkit frontend with a Node.js and Express.js backend, all powered by a MongoDB Atlas database. The final system is fully responsive across desktop and mobile devices, with global deployment handled through Vercel for the client side and Render for the server infrastructure.**

***Index Terms*—WebRTC, Real-Time Communication, Peer-to-Peer Networking, JSON Web Tokens (JWT), React.js, Node.js, MongoDB Atlas.**

I. INTRODUCTION

While ubiquitous platforms like Zoom, Google Meet, and Microsoft Teams excel at erasing geographical boundaries, they are inherently optimized for video first communication. Consequently, they demand high bandwidth and significant computational power even when users only require an audio connection. This creates a substantial barrier to entry in regions like India, where widespread reliance on low-cost smartphones and limited internet connectivity remains a daily reality. To address these infrastructural challenges, we propose ChatRoom: a lightweight, browser-based voice communication platform that operates entirely without installation. By functioning smoothly on just 30 to 50 KB/s of bandwidth, ChatRoom is uniquely suited for low network environments.

The system relies on WebRTC to establish secure, low latency, peer-to-peer connections, significantly reducing the reliance on centralized media servers and allowing users to effortlessly create or join real time audio rooms. Security and organized interaction are central to the platform's architecture. ChatRoom ensures strict access control through a password less authentication model powered by Twilio's One-Time Password (OTP) verification, coupled with robust token-based session management. To maintain orderly communication, the platform structures interactions using a defined role hierarchy Moderator, Speaker, and Listener across three distinct room categories: Open (public), Social (restricted to mutual followers), and Private (password-protected). Beyond fundamental voice capabilities, the application enriches the user experience with real time text chat, emoji reactions, a "raise hand" signaling mechanism, session timers, and

configurable participant limits. An integrated social layer further allows users to follow connections, receive notifications, and customize their profiles, complete with support for both light and dark themes. Drawing functional inspiration from audio-centric applications like Clubhouse and Twitter Spaces, ChatRoom distinguishes itself by delivering a fully standalone, browser-based experience specifically tailored for inclusivity on low-end devices. The application is driven by a modern full-stack ecosystem, pairing a React.js frontend with a Node.js and Express.js backend, all seamlessly supported by a MongoDB Atlas database. For global accessibility and reliability, the client side is hosted on Vercel while the backend infrastructure is deployed on Render. Following comprehensive testing across both desktop and mobile environments, the system has proven to be highly scalable, dependable, and accessible for a wide range of users.

II. LITERATURE SURVEY

[1] Zinah Tareq Nayyef et al. suggested a WebRTC based peer-to-peer multimedia communication system. The authors showed how a Node.js signaling server combined with browser-based APIs like HTML5 and JavaScript can facilitate direct user contact without the need for external plugins. The communication architecture of ChatRoom was significantly influenced by this work. The system's major components, getUserMedia() for recording audio streams and RTC Peer Connection for creating low-latency peer-to-peer connections, were validated.

[2] Neelam Kumar et al.'s study concentrated on OTP-based authentication as a safe substitute for conventional password-based systems. The study examined various OTP generating strategies, such as time-based and event-based methods, and highlighted the security benefits of their brief validity and one-time usage. ChatRoom implemented this idea by utilizing Twilio services to enable OTP-based login. OTPs are hashed using SHA-256 and verified using expiration timestamps to improve security and guarantee that private data is never kept in plain text.

[3] A thorough explanation of WebRTC, including signaling processes and the application of STUN, TURN, and ICE protocols, may be found in the work of George Suciu et al. Even when NAT and firewall limitations are in place, these components allow users to communicate across various networks. These ideas were immediately implemented in ChatRoom to guarantee dependable connectivity. Even in intricate network contexts, the system may sustain reliable communication by integrating STUN/TURN servers with RTC Peer Connection.

[4] Basic OTP systems have security flaws, including the possibility of transmission interception, according to Md. Arif Hassan et al. The authors suggested using multi-layer authentication and hashing to improve security. By securely hashing OTPs and putting in place a token-based authentication system with long-lived refresh tokens and short-lived access tokens to add an extra degree of security, ChatRoom integrates these enhancements.

[5] Ben Feher et al.'s study examined WebRTC's security characteristics, emphasizing the usage of SRTP for media stream encryption and DTLS for secure key exchange. Potential weaknesses were also found in the study, especially in signaling channels. The use of ChatRoom's built-in encryption techniques was confirmed by this study. JWT-based authentication is used for all Socket.io connections to address signaling security issues, guaranteeing that only authorized users can access communication rooms.

III. METHODOLOGY

A. Architecture of the System

The Frontend, Backend, and Database layers make up Chatroom’s three-tier architecture, which guarantees scalability and modularity. React.js and Redux Toolkit are used to construct the frontend, which incorporates the Socket.io client and browser-based WebRTC for real-time communication. Vercel is used for its deployment. Node.js and Express.js are used to create the backend, which also includes Twilio services for OTP verification, JWT-based authentication for security, and Socket.io for real time communication. Render is used to host the backend. The database layer manages collections including Users, Rooms, Follows, and Refresh Tokens using MongoDB Atlas with Mongoose ODM. Socket.io is used for real-time updates and REST APIs for common activities

between the frontend and backend. WebRTC reduces latency and boosts performance by establishing direct peer-to-peer connections between users for audio communication, with the server just managing signaling.

TIER 1: FRONTEND	TIER 2: BACKEND	TIER 3: DATABASE
React.js	Node.js	MongoDB
Redux Toolkit	Express.js	Atlas
Socket.io Client	Socket.io Server	Mongoose ODM
WebRTC (Browser)	JWT Auth	Users Collection
CSS Variables	Twilio SMS	Rooms Collection
Hosted on Vercel	Hosted on Render	Follows Collection
		RefreshTokens

Fig. 1. System Architecture of ChatRoom – Three Tier Design

B. Types of Rooms and Access Management

Open, Social, and Private rooms are all supported by ChatRoom. All users have unrestricted access to open rooms. Social rooms are only accessible by mutual followers, therefore in order to watch or join the room, both users must follow one another. Access to private rooms requires a password that has been set by the creator. A multi-layer validation method is used to guarantee safe and regulated access. In order to prevent unauthorized users from viewing restricted rooms, the first layer filters room visibility during API calls. When a user tries to join a room, the second layer verifies their access and returns an error if certain requirements are not satisfied. This three-tiered system guarantees that Unauthorized users are instantly disconnected by the third layer, which implements access control at the Socket.io level. Access limitations cannot be circumvented thanks to this three-layer system.

C. Database Architecture

The system makes use of MongoDB's NoSQL database paradigm, which stores data in document format. User-related data, including phone number, profile information, and account status, is stored in the Users collection. Information regarding room subjects, categories, ownership, and member limits is kept up to date in the Rooms collection. Social room access is made possible via the Follows collection, which depicts user relationships. Furthermore, a collection of Refresh Tokens is utilized to safely manage authentication sessions. The connections among these collections facilitate effective data retrieval and support functions like social networking, room management, and user authentication.

D. Design of System Flow

When a user uses a web browser to access the application, the system flow starts. A secure session is created after the user logs in using OTP-based authentication. Depending on their access permissions, the user can then create or join a room. The backend creates a Socket.io connection for real time communication and runs validity checks when a user joins a room. WebRTC establishes direct peer-to-peer connections for voice communication, with the server facilitating signaling. Real-time events are used to manage additional features including text messaging, reactions, and participant management. Effective communication, minimal latency, and safe interaction across various devices and network circumstances are guaranteed by this architecture.

IV. IMPLEMENTATION

A. Module for Authentication

Using a phone number and a one-time password (OTP), the authentication module guarantees safe user verification. In contrast to conventional systems, identity is verified only using a 4-digit OTP method; no password is needed. The system employs several measures to improve security. To avoid misuse, the OTP is never kept in plain text but rather in a SHA256 hash. Furthermore, the OTP's two-minute validity period lowers the possibility of replay and interception assaults. JSON Web Tokens (JWT), which are saved in HttpOnly cookies to guard against cross-site scripting (XSS) attacks, are used to maintain sessions. Additionally, secure session control and token invalidation during logout are made possible by the MongoDB database's maintenance of refresh tokens.

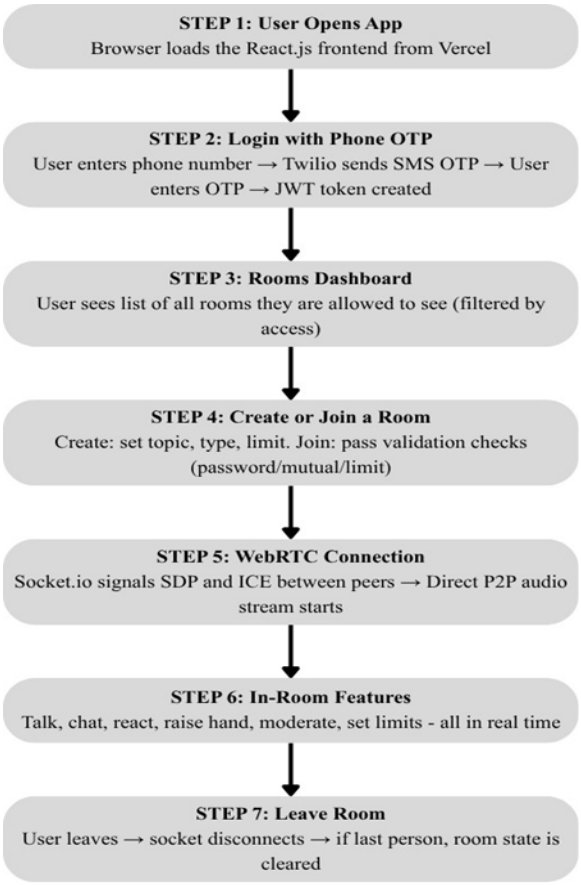

Fig. 2. System Flow Diagram

B. Module for Room Management

Core functions like creating new rooms, listing available rooms, joining existing rooms, and deleting rooms are handled by the room management module. This module makes sure that the system's communication areas are accessible and efficiently organized. Rooms with distinct identifiers can be created by users and saved in the MongoDB database. Depending on availability and access rights, the system enables users to view and join active rooms. Additionally, by eliminating inactive or underutilized rooms, room deletion functionality guarantees appropriate resource management.

C. Audio Module for WebRTC

WebRTC is used to implement real-time audio communication. Every participant establishes a direct connection with every other person in the room using a peer-to-peer mesh network. The Socket.io server is only used for signaling, which includes exchanging ICE candidates and session descriptions (offer/answer). Audio streams are sent straight between clients once the connection is made, which lowers latency and server burden.

D. Module for Role-Based Access

Three roles Moderator, Speaker, and Listener are used by the system to impose access control. Administrative rights, including participant management and room configuration control, are granted to the moderator, who is usually the room founder. While listeners have limited access and must ask to communicate, speakers are able to actively participate in audio conversation. To maintain uniformity and security, role assignment is controlled at the server level.

V. EXPECTED OUTCOMES

A. Latency and Voice Quality

Under various user loads, audio performance was assessed. Voice transmission was clear and had very little latency when two users were connected to the same Wi-Fi network. The audio quality did not change when there were four participants, albeit there was a slight lag. Each participant in WebRTC mesh based communication maintains numerous peer connections, which is consistent with this behavior. For small gatherings, especially those with up to six or seven active speakers, the system works well.

B. Performance on Low-End Equipment An entry-level Android device with 2 GB of RAM and constrained processor power was used to test the system. Over a 4G network, the application started up in about 3-4 seconds. Chat, emoji, and audio communication all ran seamlessly and without any lag. These findings show that the platform is well suited to contexts with limited resources.

C. Screenshots

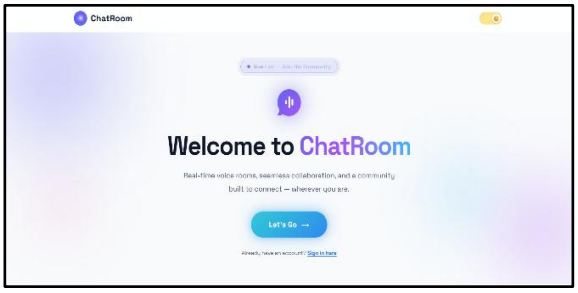

Fig. 3. Landing Page - Entry point for new returning users

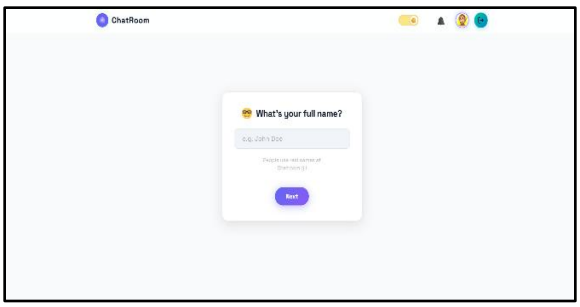

Fig. 4. Activation Page – Enter Username and Avatar

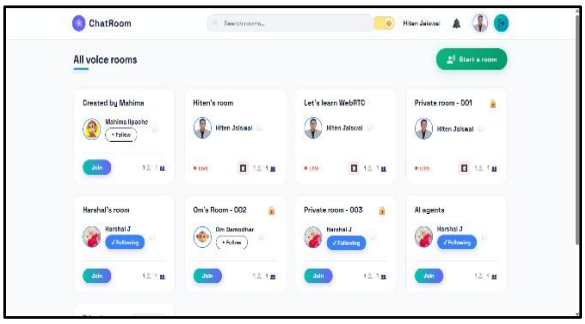

Fig. 5. Rooms Dashboard - Shows all accessible rooms with follow and join buttons

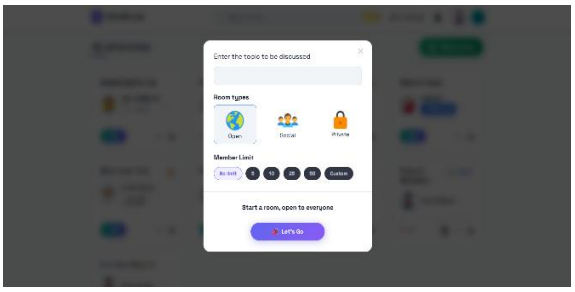

Fig. 6. Rooms Creation Modal

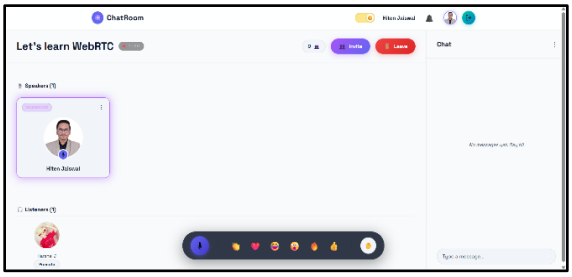

Fig. 7. Voice Room Page

VI. CONCLUSION

Real-time voice conversation via web browsers is made possible by the ChatRoom system, a fully operational and deployed web-based program that does not require the installation of extra software. The platform has been successfully created, put into use, and verified in relation to all of the original goals. With built-in encryption, the system uses WebRTC to create secure peer-to-peer audio connections, guaranteeing that voice data is sent directly between users. Twilio supports a phone number-based OTP approach for user authentication, which eliminates reliance on conventional password-based solutions and improves security. The application offers many room types, such as open, social, and private rooms, to offer customizable communication control. Specifically, the social room mechanism offers a controlled access paradigm in which only users who are related to each other can find and join particular rooms providing a more customized setting for engagement. Users with low-end devices or poor internet connectivity will particularly benefit from the platform. The system functions well even on slower networks like 3G because of its low bandwidth usage (around 30-50 kbps), which makes it a viable substitute for high bandwidth video conferencing options. Technically speaking, the project incorporates a number of contemporary technologies, such as Socket. MongoDB for data storage, Node.js and Express.js for backend services, IO for signaling, and React for frontend development. JSON Web Tokens are used to handle sessions securely. The system's practical application beyond academic requirements is demonstrated by the effective integration and deployment of these components. All of the specified test cases passed after extensive testing. The system's dependable performance of voice communication, messaging, reactions, and social features was confirmed by user input in both desktop and mobile contexts. In conclusion, ChatRoom offers a practical, safe, and effective audio communication solution. It offers a useful substitute for resource-intensive platforms, especially for users who need dependable and straightforward voice interaction.

REFERENCES

[1] Z. T. Nayyef, S. F. Amer, and Z. Hussain, “Peer-to-peer multimedia real-time communication system based on WebRTC technology,” *Int. J. History Eng. Technol.*, Feb. 2019.

[2] N. Kumar, R. Kamje, P. Landge, S. Shitole, and B. Takale, "OTP-based authentication system," *Alochana J.*, vol. 13, no. 12, 2024.

[3] G. Suciu, S. Stefanescu, C. Beceanu, and M. Ceaparu, "WebRTC role in real-time communication and video conferencing," *IEEE Xplore*, Jun. 2019.

[4] M. A. Hassan, Z. Shukur, and M. K. Hasan, "An improved time-based one-time password authentication framework for electronic payments," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 11, Dec. 2020.

[5] B. Feher, L. Sidi, A. Shabtai, and R. Puzis, "The security of WebRTC," *arXiv preprint*, arXiv:1601.00184, Jan. 2016.

[6] K. Tiwari *et al.*, "Voice chat web app using WebRTC," *IJRDO J. Comput. Sci. Eng.*, vol. 8, no. 11, 2022.

[7] P. Kaur *et al.*, "Talk over: A voice chat web application based on WebRTC," *Int. J. Comput. Appl.*, vol. 185, no. 15, 2023.

[8] W3C WebRTC Working Group, "WebRTC 1.0: Real-time communication between browsers," W3C Recommendation, Jan. 2021.