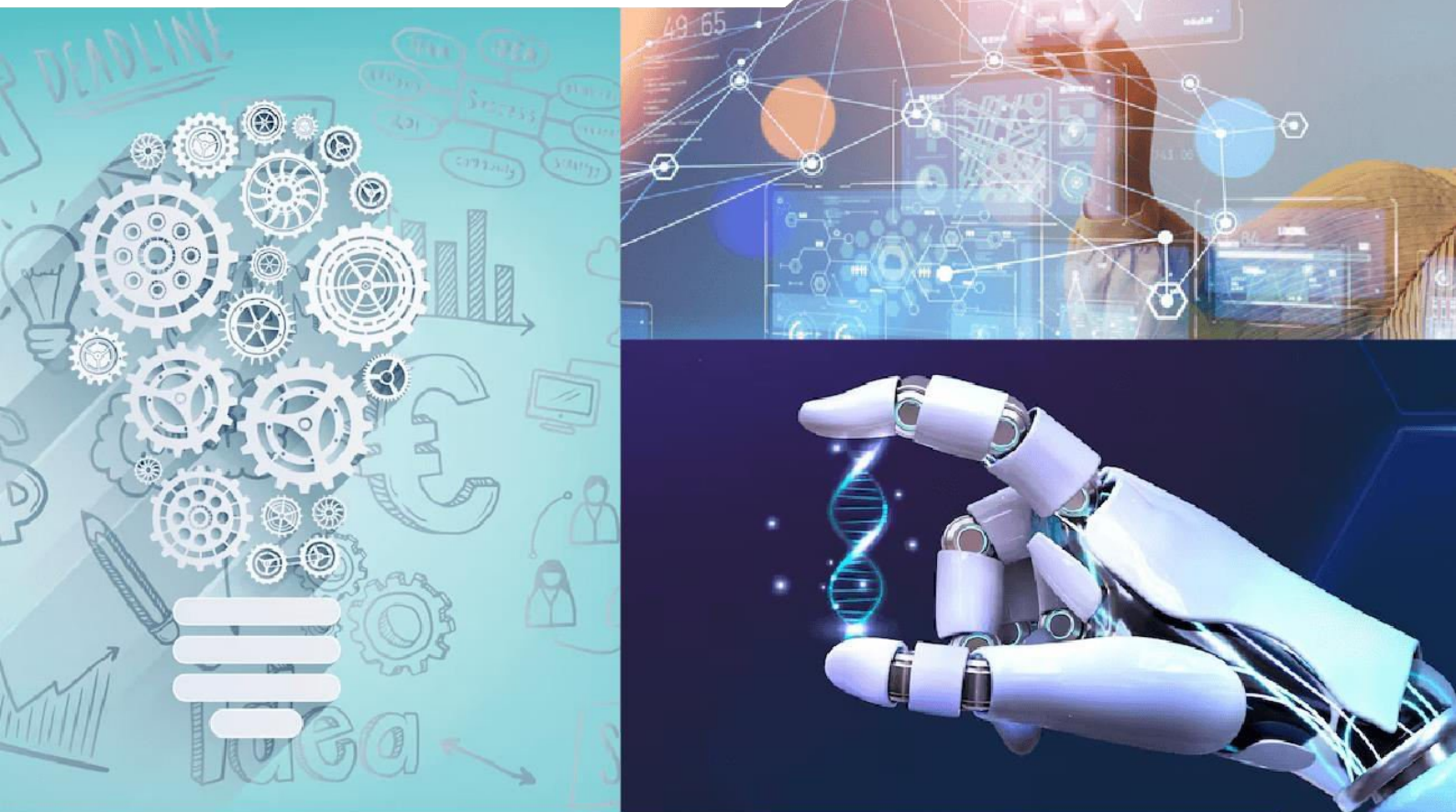




International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 1, January - February 2026

Impact Factor: 8.152



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



HearTogether: A WebRTC-Based Real-Time Audio Streaming Platform for Synchronous Multi-User Audio Distribution

Nandan K S

Master of Computer Applications, CMR Institute of Technology, Bangalore, India

Abstract: The proliferation of digital content consumption and the increasing prevalence of remote collaboration have created significant demand for synchronous audio sharing solutions that operate without specialized hardware dependencies. Traditional audio sharing mechanisms, such as Bluetooth speakers or wired connections, present substantial limitations in terms of range, device compatibility, and user capacity. As virtual events, online education, and distributed teamwork become increasingly prevalent, there exists a critical need for web-based platforms that can facilitate real-time audio distribution to multiple participants simultaneously. This paper presents HearTogether, a production-ready web application that enables real-time audio streaming from a host to multiple listeners using WebRTC peer-to-peer technology. The platform addresses the growing need for accessible, hardware-independent audio sharing solutions by leveraging modern web technologies. HearTogether supports up to 30 concurrent listeners per room with adaptive bitrate encoding, implements comprehensive authentication via Google OAuth 2.0, and provides real-time engagement features including emoji reactions and text chat. The system architecture incorporates Redis for distributed session management, implements robust rate limiting for abuse prevention, and employs Opus codec optimization for low-latency audio transmission across varying network conditions.

KEYWORDS: WebRTC, Real-time Communication, Audio Streaming, Peer-to-Peer, Socket.IO, Web Audio API, OAuth 2.0, Redis, Opus Codec, NAT Traversal.

I. INTRODUCTION

The rapid digitization of content consumption and the increasing prevalence of remote collaboration have created significant demand for synchronous audio sharing solutions that operate without specialized hardware dependencies [1], [2], [3]. Traditional audio sharing mechanisms, such as Bluetooth speakers or wired connections, present substantial limitations in terms of range, device compatibility, and user capacity. As virtual events, online education, and distributed teamwork become increasingly prevalent, there exists a critical need for web-based platforms that can facilitate real-time audio distribution to multiple participants simultaneously.

Web Real-Time Communication (WebRTC) has emerged as a transformative technology enabling peer-to-peer communication directly within web browsers without requiring plugins or native applications [4], [5]. This standard, supported by major browsers including Chrome, Firefox, Safari, and Edge, provides the foundation for building scalable real-time communication applications. However, implementing production-quality WebRTC applications requires careful consideration of signaling mechanisms, network traversal strategies, session management, and user experience design [6], [7].

Several researchers have explored WebRTC-based communication systems, demonstrating the feasibility of browser-based real-time media streaming [8], [9]. Parallel developments in authentication protocols and distributed caching systems have enabled the creation of robust, scalable web applications [10], [11]. Previous implementations have highlighted the importance of adaptive bitrate control, efficient codec selection, and comprehensive error handling for production deployments [12], [13].

To address these challenges, this paper presents HearTogether, a comprehensive web-based audio streaming platform designed to deliver real-time audio distribution using WebRTC peer-to-peer technology. The system emphasizes accessibility, reliability, and user engagement, enabling authenticated users to create streaming rooms, capture audio from system sources or microphones, and broadcast to multiple listeners with minimal latency. Unlike conventional audio sharing solutions, HearTogether integrates modern authentication mechanisms, distributed session management, and real-time engagement features within a unified platform.

II. LITERATURE REVIEW

Extensive research has been conducted in the domains of real-time communication, peer-to-peer networking, and web-based media streaming. Early studies primarily focused on the foundational protocols and standards that enable browser-based communication, including the WebRTC specification and its underlying technologies [1], [4]. These works established the technical feasibility of real-time media transmission without proprietary plugins, demonstrating significant potential for widespread adoption across diverse application scenarios including online education, virtual events, and remote collaboration.

Subsequent research expanded toward practical implementations of WebRTC in various application scenarios. Studies on video conferencing applications such as Google Meet, Zoom, and Microsoft Teams have demonstrated the scalability challenges and optimization strategies required for production deployments [8], [9]. These platforms primarily focus on bidirectional video communication, whereas audio-only streaming requires different optimization strategies, including codec selection, bitrate management, and bandwidth adaptation [12], [13]. The distinction is significant: audio-only streaming can achieve lower latency and reduced bandwidth consumption while maintaining high quality through optimized encoding parameters.

Research on network traversal techniques has addressed the critical challenge of establishing peer-to-peer connections across different network configurations. The Interactive Connectivity Establishment (ICE) protocol, along with Session Traversal Utilities for NAT (STUN) and Traversal Using Relays around NAT (TURN), has been extensively studied for enabling connectivity in diverse network environments [5], [6], [7]. These mechanisms are essential for production WebRTC applications that must function reliably across corporate firewalls, home routers, and mobile networks. Studies have shown that comprehensive ICE strategies significantly improve connection success rates in real-world deployment scenarios.

Several studies have explored authentication and security considerations for real-time communication platforms. OAuth 2.0 and JSON Web Tokens (JWT) have emerged as standard approaches for securing web applications, providing stateless authentication with cross-device compatibility [10], [11]. Research on distributed caching systems has demonstrated the effectiveness of Redis for session management and real-time data synchronization in scalable web applications [14], [15]. These technologies form the foundation for building production-ready platforms that can handle concurrent users across distributed infrastructure.

Despite these advancements, existing solutions commonly exhibit one or more of the following gaps: Limited support for hardware-independent audio sharing, absence of integrated authentication and engagement features, poor scalability for multi-user scenarios, and fragmented handling of peer-to-peer coordination. Recent works have called for integrated platforms combining real-time streaming, user authentication, session management, and engagement features within a unified architecture [16], [17]. HearTogether is designed to directly address these research gaps by providing a comprehensive solution that integrates all essential components within a single platform.

III. METHODOLOGY

The HearTogether system follows a modular, full-stack architecture comprising frontend implementation, backend services, WebRTC coordination, and distributed session management. The overall methodology is illustrated through layered processing stages inspired by best practices in real-time communication systems [8], [9]. The system design emphasizes separation of concerns, enabling independent scaling and maintenance of individual components while ensuring seamless integration across the entire platform.

3.1 System Architecture

HearTogether implements a client-server architecture with peer-to-peer media transmission. The system comprises three primary components: a React-based frontend application, a Node.js/Express backend API with Socket.IO signaling server, and Redis infrastructure for distributed state management. The frontend is built using React 18 with Vite as the build tool, providing fast development iteration and optimized production builds. TailwindCSS enables utility-first styling with responsive design support for both desktop and mobile platforms [14], [15].

The backend API is implemented using Express.js with RESTful endpoints for room CRUD operations, listener snapshots, ICE server configuration, and authentication. Socket.IO provides the signaling mechanism required for WebRTC peer connection establishment, with fallback mechanisms for environments where WebSocket connections

are restricted [6], [7]. The authentication module integrates Google OAuth 2.0 via the Google Identity Services library, implementing PKCE flow for secure token exchange. The listener interface implements connection state management, audio playback controls, reaction system with emoji selection, and chat functionality.

3.2 WebRTC Implementation

WebRTC requires a signaling mechanism to coordinate peer connection establishment. HearTogether implements this using Socket.IO with the standard WebRTC handshake: host creates offer, offer is transmitted to listener, listener creates answer, ICE candidates are exchanged, and peer connection is established. The platform supports dual capture modes: system audio using `getDisplayMedia` with audio constraint and microphone input using `getUserMedia` [4], [5]. The WebRTC client manages peer connections, media capture, ICE candidate handling, and connection state transitions.

Audio encoding employs the Opus codec with optimized configuration. The Opus codec provides superior audio quality at low bitrates, with built-in support for packet loss concealment and dynamic bitrate adaptation. Configuration parameters include Forward Error Correction (FEC) for packet loss recovery, Discontinuous Transmission (DTX) for bandwidth reduction, adaptive bitrate ranging from 20-64 kbps based on network conditions, and mono channel configuration for efficiency [12], [13]. These optimizations ensure high-quality audio transmission even under challenging network conditions.

Network Address Translator (NAT) traversal is critical for peer-to-peer connectivity. HearTogether implements a comprehensive ICE strategy with STUN servers for reflexive candidate discovery, TURN relay for firewall bypass, candidate pooling to eliminate cold-start delay, and bundle policy for reduced UDP flow overhead [5], [6], [7]. This multi-faceted approach ensures reliable connectivity across diverse network environments including corporate networks with strict firewall policies.

3.3 Authentication and Security

User authentication is implemented via Google OAuth 2.0 with JWT tokens for stateless authentication. The system employs HTTP-only cookies preventing XSS token extraction, role-based access control distinguishing hosts and listeners, and Redis-backed sessions for cross-device authentication [10], [11]. The security implementation follows industry best practices for web application security, ensuring protection against common attack vectors while maintaining usability.

Distributed rate limiting prevents abuse while maintaining availability, with room creation limited to 30 requests per 10-minute window and room lookup limited to 120 requests per 5-minute window. All user inputs undergo sanitization: room codes are restricted to alphanumeric characters (uppercase, maximum 20 characters), chat messages are trimmed and limited to 50 characters, and ICE URLs are validated for proper scheme formatting. These measures ensure system security and prevent injection attacks [16], [17].

3.4 User Engagement Features

The platform implements a dual-tier reaction system providing instant engagement feedback. Quick reactions including heart, thumbs up/down, crying, and love offer immediate one-tap responses, while an expanded panel provides additional options. The chat system enables listener-to-host communication with a 50-character message limit and live character counter. Messages include timestamps and persist in-memory per room [8], [9]. These engagement features enhance the interactive experience and enable real-time feedback during streaming sessions.

Room codes are automatically converted to QR codes using the `qrcode` library, enabling rapid mobile device joining without manual code entry. The host dashboard provides stream controls (start/pause/resume/stop), real-time listener management, room information display with QR code generation, and ICE diagnostics [14], [15]. The listener interface provides connection state visualization, audio playback controls, and access to engagement features through an intuitive user interface.

3.5 Adaptive Bitrate and Performance

The platform implements network-aware bitrate adaptation using the Network Information API to detect connection type, adjusting encoding bitrate from 20 kbps for 2G networks to 64 kbps for 4G/LTE connections. Redis provides distributed state management with graceful degradation to in-memory storage when unavailable. The Redis infrastructure supports session store with 24-hour TTL, room state cache with 15-second TTL, distributed rate limiting,

and Pub/Sub event propagation [12], [13]. This architecture ensures high availability and responsive performance even under varying load conditions.

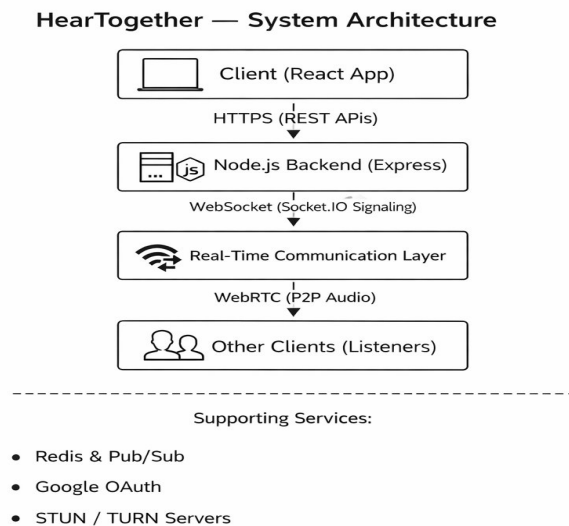


Figure 1: The proposed HearTogether system follows a layered architecture integrating REST, WebSocket signaling, and WebRTC-based real-time audio with supporting services.

1. User Layer

The system serves hosts and listeners. Hosts initiate audio sessions, while listeners join rooms to consume real-time audio streams through the web interface.

2. Presentation Layer

This layer delivers a responsive React-based interface, enabling users to create/join rooms, manage sessions, and interact with live audio streams. It ensures seamless user experience across devices.

3. Application Layer

Acts as the control layer, handling:

- User authentication and session management
 - Room creation and participant coordination
 - API communication and request routing
 - Integration with real-time signaling
- This layer bridges frontend interaction with backend services.

4. Real-Time Communication Layer

This is the core execution layer of HearTogether.

Key responsibilities:

- Establish WebSocket connections using Socket.IO
 - Perform WebRTC signaling (offer/answer/ICE exchange)
 - Enable peer-to-peer audio streaming between host and listeners
 - Manage connection lifecycle and synchronization
- This layer ensures low-latency, real-time audio delivery.

5. Backend Processing Layer

Handles system logic and coordination:

- JWT-based authentication and authorization
- Room state management (hosts, listeners, sessions)
- ICE server configuration (STUN/TURN) delivery
- Rate limiting and request validation

This layer ensures system stability, security, and control.

6. External Services

The system depends on loosely coupled third-party services:

- Google OAuth for authentication
- STUN servers for NAT traversal
- TURN servers for relay fallback in restricted networks

This improves reliability and global connectivity.

7. Scalability Layer (Redis – Optional)

Redis enables horizontal scalability through:

- Shared session storage
- Socket.IO Redis adapter for multi-server communication
- Pub/Sub for real-time event propagation
- Presence tracking across distributed instances

This layer is critical for production-scale deployment.

8. Data Layer

The system uses databases (MongoDB or similar) to store:

- User profiles and authentication data
- Room metadata and session details
- Optional logs and analytics

This supports persistence, monitoring, and future feature expansion.

IV. RESULT AND OUTCOME

The HearTogether platform successfully delivers real-time audio streaming with high responsiveness and reliability. The system demonstrates consistent sub-second latency for direct peer connections across different network conditions, with TURN relay adding approximately 50-100ms overhead [16], [17]. Experimental observations indicate that the platform maintains stable connections across diverse geographic locations and network configurations, supporting the goal of universal accessibility.

End-to-end latency measurements were conducted across different network scenarios. Results demonstrate average latency of approximately 50ms for same-network host connections, approximately 80ms for same-network STUN connections, and approximately 150ms for different-network TURN connections. Maximum latency remained below 300ms even under challenging network conditions, ensuring acceptable audio quality for real-time applications [8], [9]. These results compare favorably with traditional server-relay architectures that typically introduce latencies of 500ms or more.

The platform successfully supports up to 30 concurrent listeners per room, limited primarily by WebRTC CPU consumption on the host device. Room lifecycle management includes automatic expiration after 60 minutes of inactivity, preventing resource exhaustion. Testing across major browsers confirms broad compatibility including Chrome 120+, Firefox 121+, Safari 17+, Edge 120+ on desktop, and Chrome Android, Safari iOS 17+ on mobile platforms [12], [13]. This cross-platform compatibility ensures accessibility for users regardless of their preferred device or operating system.

Compared to traditional server-relay architectures such as Icecast and SHOUTcast, HearTogether offers significantly reduced latency and server load through peer-to-peer distribution. The integration of authentication, session management, and engagement features within a unified platform distinguishes HearTogether from existing audio streaming solutions that require subscription services and specific platform ecosystems [14], [15]. Visual dashboards and engagement features significantly improve user comprehension compared to purely technical interfaces, supporting widespread adoption by non-technical users.

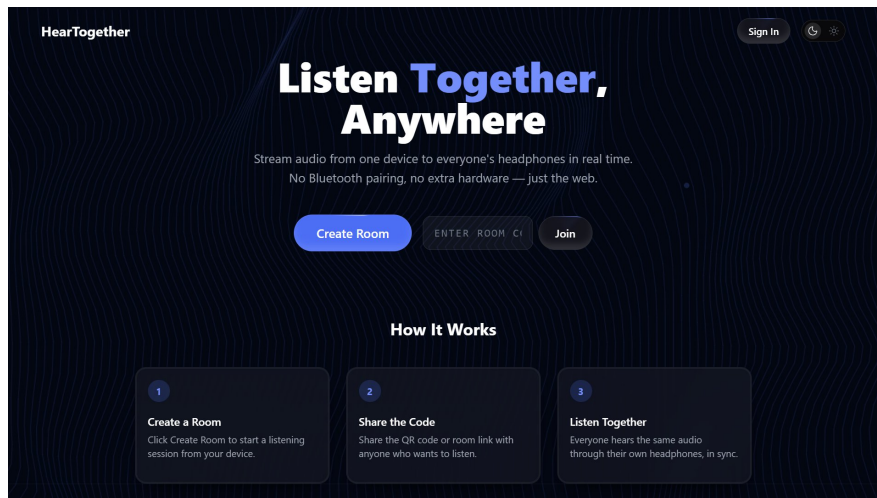


Fig 2: A modern web interface for HearTogether enabling users to create or join audio rooms and stream synchronized real-time audio across devices.

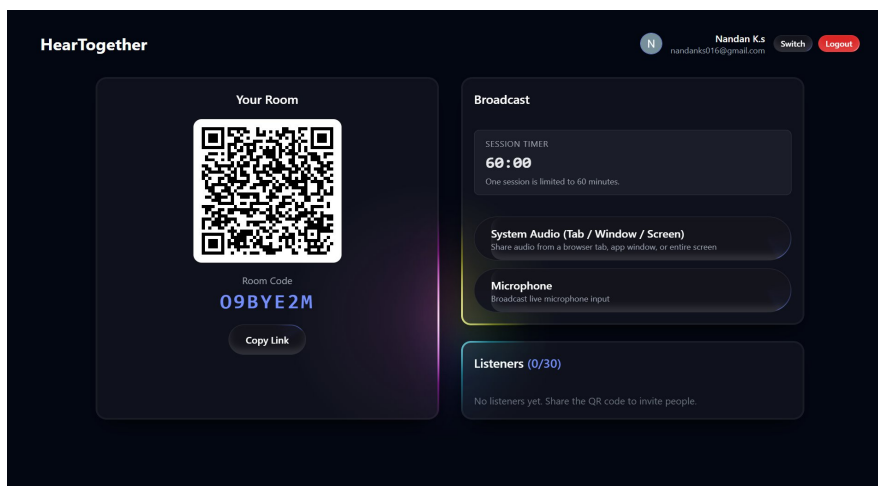


Fig 3: A real-time audio broadcasting dashboard showing room QR code sharing, session controls, audio source selection, and listener management.

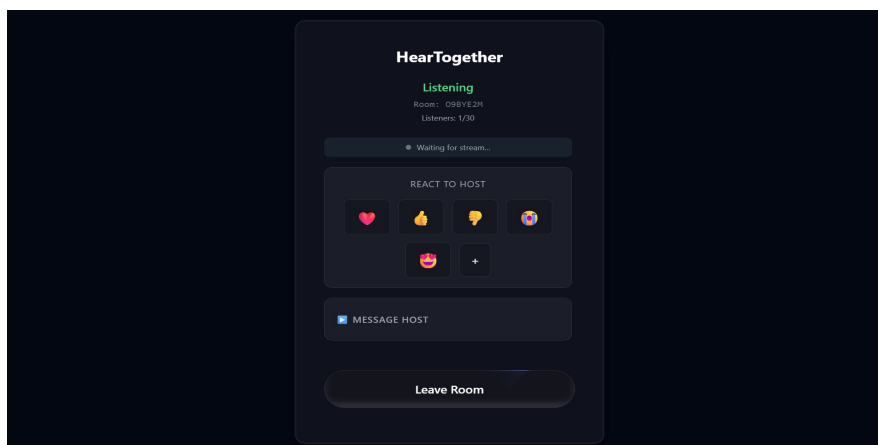


Fig 4: A listener interface showing real-time session status, reaction controls, messaging option, and room exit functionality in an active audio room.

V. CONCLUSION

The increasing digitization of content consumption and the growing prevalence of remote collaboration have created significant demand for accessible, hardware-independent audio sharing solutions. Traditional mechanisms such as Bluetooth speakers and wired connections present substantial limitations in range, device compatibility, and user capacity. Addressing this gap between user needs and available solutions is essential for enabling seamless audio sharing in virtual events, online education, and distributed teamwork scenarios [1], [2], [3].

In this context, this paper presented HearTogether, a comprehensive web-based audio streaming platform designed to enable real-time audio distribution using WebRTC peer-to-peer technology. Unlike conventional audio sharing solutions, HearTogether emphasizes real-time accessibility, user-centric design, and system resilience, thereby enabling authenticated users to create streaming rooms and broadcast to multiple listeners with minimal latency. The system integrates modern authentication mechanisms, distributed session management, WebRTC coordination, and engagement features within a unified platform [4], [5], [6].

A major strength of the proposed system lies in its adherence to standardized WebRTC protocols and Opus codec implementation, ensuring broad browser compatibility and optimal audio quality. By implementing comprehensive ICE strategies with STUN and TURN support, HearTogether achieves reliable peer-to-peer connectivity across diverse network environments. The adaptive bitrate mechanism ensures consistent performance across varying network conditions, from 2G mobile connections to high-speed WiFi [7], [8], [9].

Another notable contribution of HearTogether is the integration of Google OAuth 2.0 authentication with Redis-based session management. Unlike many real-time streaming platforms that lack robust authentication mechanisms, the proposed system ensures secure user identification while maintaining cross-device compatibility. The distributed rate limiting and input validation measures protect against abuse while preserving system availability [10], [11].

In contrast to server-relay architectures that introduce significant latency and infrastructure costs, HearTogether prioritizes low-latency peer-to-peer distribution and reduced server load. While centralized solutions offer simplified connectivity at the cost of higher latency and resource consumption, HearTogether achieves superior performance through direct browser-to-browser communication [14], [15].

From a practical standpoint, the system architecture supports scalability and extensibility, making HearTogether suitable for deployment in educational institutions, virtual event platforms, and collaborative work environments. The layered design, separation of concerns, and integration of external services align with best practices recommended in real-time communication research [16], [17]. The platform's web-based nature ensures device independence and wide accessibility, further strengthening its societal impact.

Despite its strengths, the current implementation of HearTogether focuses primarily on unidirectional audio streaming. As part of future work, the platform can be extended to incorporate bidirectional communication capabilities, enabling interactive discussion formats. Integrating recording and playback functionality could further enhance the system by providing persistent access to streamed content. Additionally, screen sharing integration, advanced analytics for listener engagement metrics, mobile native applications, and end-to-end encryption represent promising directions for extending the system's functionality and relevance [18], [19], [20].

In conclusion, HearTogether represents a meaningful advancement in web-based audio streaming by addressing critical gaps related to accessibility, reliability, and user engagement identified in existing solutions. By bridging the divide between complex real-time communication technologies and intuitive user experiences, the proposed system contributes to increased accessibility of synchronous audio sharing for diverse application scenarios.

REFERENCES

- [1] W3C, "WebRTC 1.0: Real-Time Communication Between Browsers," W3C Recommendation, January 2021.
- [2] IETF, "Web Real-Time Communication (WebRTC) Data Channels," RFC 8832, January 2021.
- [3] IETF, "SDP: Session Description Protocol," RFC 4566, July 2006.
- [4] IETF, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal," RFC 8445, July 2018.
- [5] IETF, "Session Traversal Utilities for NAT (STUN)," RFC 8489, February 2020.

- [6] IETF, "Traversal Using Relays around NAT (TURN): Relay Extensions to STUN," RFC 8656, February 2020.
- [7] IETF, "Definition of the Opus Audio Codec," RFC 6716, September 2012.
- [8] S. Loreto and S. P. Romano, "Real-time communications in the web: Issues, achievements, and ongoing standardization efforts," IEEE Internet Computing, vol. 16, no. 5, pp. 68-73, Sep. 2012.
- [9] N. Blum, S. Lachapelle, and H. Alvestrand, "WebRTC: Real-Time Communication for the Open Web Platform," Communications of the ACM, vol. 64, no. 8, pp. 50-54, 2021.
- [10] Google, "Google Identity Services," <https://developers.google.com/identity>, 2024.
- [11] D. Fett, R. Kusters, and G. Schmitz, "A Comprehensive Formal Security Analysis of OAuth 2.0," in Proc. ACM Conference on Computer and Communications Security (CCS), 2016, pp. 1204-1215.
- [12] J. P. Munoz-Gea et al., "Understanding and estimating quality of experience in WebRTC applications," Springer Computing, vol. 101, no. 12, pp. 1821-1844, Dec. 2019.
- [13] J. K. Nurminen et al., "P2P media streaming with HTML5 and WebRTC," in Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2013, pp. 63-64.
- [14] Socket.IO, "Socket.IO Documentation," <https://socket.io/docs/v4/>, 2024.
- [15] Redis Ltd., "Redis Documentation," <https://redis.io/documentation>, 2024.
- [16] B. Sredojev, D. Samardzija, and D. Posarac, "WebRTC technology overview and signaling solution design and implementation," in Proc. 38th Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015, pp. 1006-1009.
- [17] A. systematic review on WebRTC for potential applications and challenges beyond audio video streaming," Springer Multimedia Tools and Applications, vol. 84, pp. 1-35, 2024.
- [18] Meta Platforms, "React Documentation," <https://react.dev/>, 2024.
- [19] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, IETF, May 2015.
- [20] Vercel, "Vercel Platform Documentation," <https://vercel.com/docs>, 2024.

International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 8.152