"Real Time Communication App"

A Synopsis
Report submitted in partial fulfilment of
The requirements for the degree of

Bachelor of Technology
In
Computer Science and Engineering

BBS College of Engineering & Technology,
Phaphamau, Prayagraj

Department of Computer Science & Engineering
Session: 2025-26

Submitted by:	**Under the Supervision of:**
Name: ADARSH KR. MAURYA Roll No:2301380109001	Mr. PRASHAT SHRIVASTAVA
Name: ADITYA KUMAR Roll No:2301380109002	
Name: VIMALESH KR. YADAV , Roll No:2301380109014	

Table of Contents

1 Introduction

1.1 Background of the Problem / Context

The rise of remote work, online education, and virtual collaboration has made one-on-one communication tools essential. While platforms like Zoom and Google Meet dominate group interactions, they are often over-featured, resource-heavy, and lack privacy for focused 2-person sessions such as interviews, tutoring, counseling, or personal calls. Existing solutions suffer from high latency, complex interfaces, and limited customization, making them unsuitable for lightweight, secure, and innovative personal communication.

1.2 Why the Project is Important; Relevance to Branch/Industry

This project is highly relevant to Computer Science & Engineering as it integrates **full stack web development (MERN)**, real-time systems (WebRTC), peer-to-peer networking, and user-centric design core skills in modern software engineering. In industry, companies like Google, Microsoft, and startups rely on WebRTC for video solutions. The project addresses privacy concerns and introduces unique features not available in Zoom or Meet, making it a standout for technical interviews and startup ideas.

1.3 Definitions of Key Terms

- **WebRTC**: Web Real-Time Communication open framework for browser-to-browser audio/video streaming.
- **P2P (Peer-to-Peer)**: Direct connection between two devices without server re-laying media.
- **Signaling**: Exchange of connection metadata (SDP, ICE candidates) via a server.
- **MERN Stack**: MongoDB, Express.js, React.js, Node.js a full-stack JavaScript framework.

2 Problem Statement / Problem Formulation

Develop a secure, real-time, 2-person video communication web application that allows audio/video calling, screen sharing, and chat, with a strict limit of two users per session. The app must include unique features not found in Zoom or Google Meet, such as **AI driven voice-to-text summaries etc**. The system should be lightweight, private, and deployable on modern browsers.

Scope: Web-based, supports desktop/mobile, uses P2P WebRTC with signaling.
Limitations: No group calls, no cloud recording, requires internet and modern browsers.

3 Objectives

3.1 Main Objective

To design and implement a lightweight, secure 2-person communication platform using MERN stack and WebRTC, with innovative features for enhanced user experience.

3.2 Specific Objectives

1. Implement user authentication and one-click meeting link generation.
2. Enable peer-to-peer audio/video streaming with screen sharing.
3. Develop real-time chat with end-to-end encryption.
4. Integrate unique features: AI driven voice to text summaries etc.
5. Ensure exactly two users per session with automatic third-user blocking.

4 Methodology / Plan of Work

4.1 Approach

Adopt a full-stack development approach with MERN for backend/frontend using MVC, WebRTC for media, and Socket.io for signaling. Use TensorFlow.js for on-device AI features to ensure privacy and low latency.

4.2 Modules / Phases

1. **Research (1 week)**: Study WebRTC, P2P constraints, AI models.
2. **Backend Setup (2 weeks)**: Node.js, Express, MongoDB, JWT, Socket.io.
3. **WebRTC Core (2 weeks)**: Audio/video, 2-user limit enforcement.
4. **Unique Features (1 weeks)**: AI driven voice to text summaries etc
5. **Frontend UI (2 weeks)**: React with Tailwind CSS, responsive design.
6. **Testing & Deployment (2 weeks)**: Security, mobile testing, Vercel/Render.

4.3 Technologies & Tools

- **Frontend**: React.js,Bootstrap, Tailwind CSS, Socket.io-client, TensorFlow.js.
- **Backend**: Node.js, Express.js, Socket.io.
- **Database**: MongoDB Atlas.
- **Real-Time**: WebRTC.
- **Version Control**: Git/GitHub.
- **AI**: AI driven voice to text summaries convertor model.
- **Deployment**: Vercel (frontend), Render (backend), Mongo Atlas.

5 Hardware / Software Requirements

5.1 Software

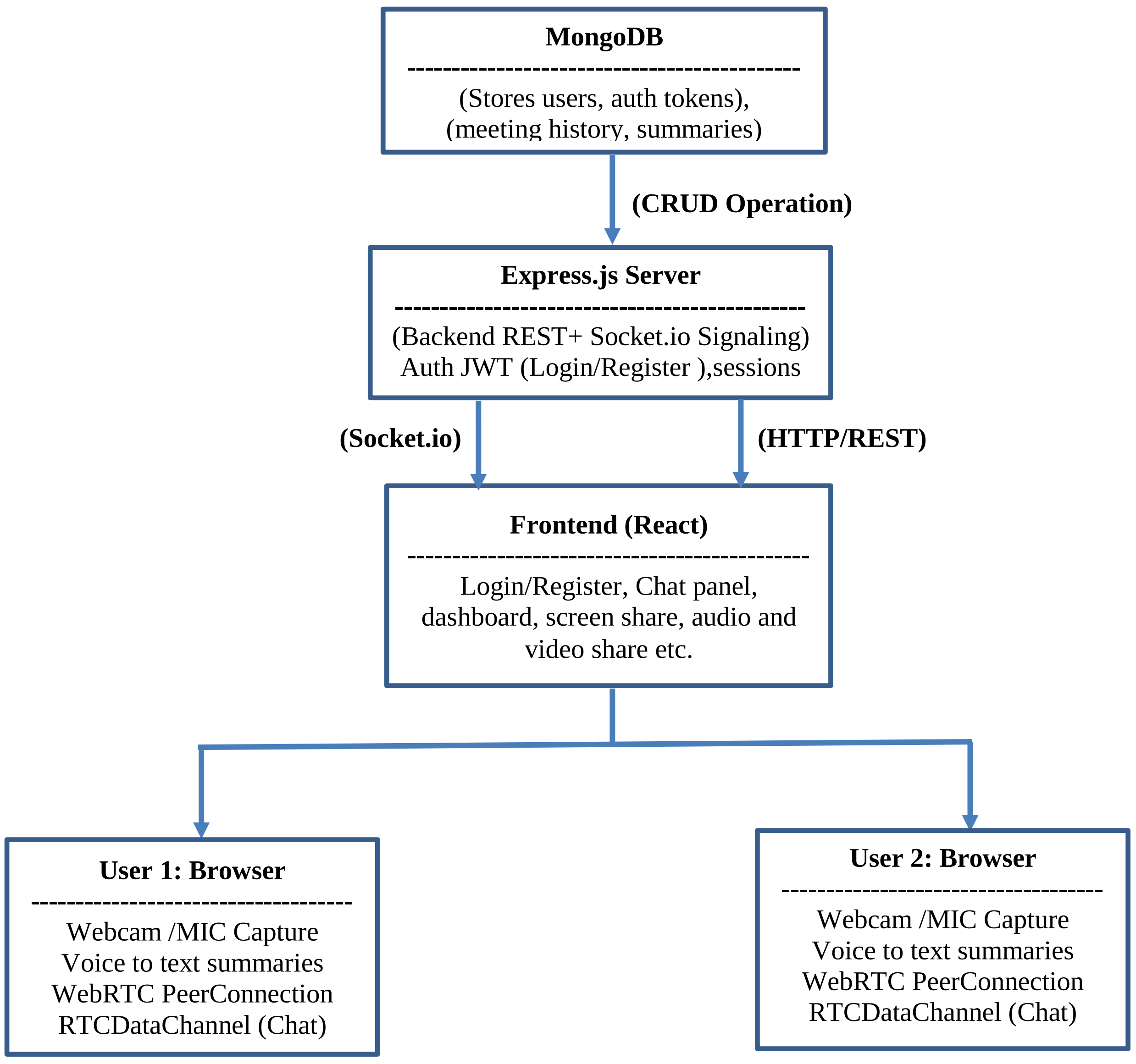
- Node.js (v18+), MongoDB Atlas, VS Code, Git, Postman.
- Browsers: Chrome/Firefox for WebRTC testing.

5.2 **Hardware** (No specific Hardware required)

- Laptop: 4GB RAM, i5/Ryzen 5+, 256GB SSD.
- Webcam & Microphone: For audio/video testing.
- Internet: 5 Mbps+ upload/download.

6 Proposed System / Design

6.1 System Architecture



6.2 Data Flow

1. User 1 creates meeting Gets unique link.
2. User 2 joins.
3. WebRTC establishes P2P connection.

6.3 UI Modules

- Login/Dashboard, Meeting Room (2 video tiles), Chat Panel, Login and Register Module.

7 Expected Outcomes

- Fully functional 2-person communication app with live demo.
- GitHub repository with documentation.
- Project report and presentation
- Unique features: AI driven voice to text summaries etc.
- Live Project deployment.

8 Feasibility Study

8.1 Technical Feasibility

WebRTC and MERN technologies. Scikit learn, Numpy, Pandas TensorFlow.js enables AI based model on server side.

8.2 Financial / Cost Feasibility

Total cost: 0–500 (domain optional). Free tiers: Vercel, Render, MongoDB Atlas.

8.3 Resource Availability

All tools are open-source and accessible. No special hardware needed.

9 References

1. WebRTC Official Documentation. https://webrtc.org
2. MDN WebRTC API. https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API
3. Socket.io Documentation. https://socket.io
4. TensorFlow.js Emotion Detection Tutorial. https://tensorflow.org/js
5. “Real-Time Collaboration with WebRTC” IEEE 2024.
6. Other dependencies https://www.npmjs.com/