Realtime Chat Application

# 1. Title & Team Members

Title: Realtime Chat Application  
Team Members: [Bhavani Prasad Mudili-231FA04139,Manoj N-231FA04346,Naga Bhavayanth-231Fag89 ]

# 2. Abstract

The Realtime Chat Application is designed to provide instant communication between users through one-to-one chats, group conversations, file sharing, and presence indicators. It is built using the MERN stack (MongoDB, Express.js, React.js, Node.js) with WebSocket support via Socket.IO to enable low-latency, bidirectional communication.

# 3. Introduction

This project replicates core features of modern chat applications like WhatsApp. The frontend is developed in React (JavaScript + JSX) styled with Tailwind CSS, while the backend uses Node.js with Express.js and MongoDB for data storage. It supports secure, real-time messaging with group and file-sharing capabilities.

# 4. Problem Statement

Traditional communication methods often suffer from latency, lack of scalability, and poor support for multimedia. The objective is to build a scalable, efficient, and feature-rich realtime chat system.

# 5. Objectives

- Improve communication accuracy and speed  
- Support automation of message delivery  
- Enable multimedia and file sharing  
- Provide real-time presence (online/offline)  
- Ensure cost-effective scalability using MERN stack

# 6. Existing System

Existing chat applications like WhatsApp and Telegram provide robust features, but they are closed-source. Our system aims to replicate similar functionality with open-source technologies.

# 7. Proposed System

The proposed system is a MERN-stack based realtime chat application with the following features:  
- One-to-one chat  
- Group chat  
- File sharing  
- Online/offline status  
Architecture: React frontend, Express.js backend, MongoDB database, and Socket.IO for realtime communication.

# 8. Methodology

Step-by-step process:  
1. Setup backend server with Node.js and Express.js  
2. Integrate MongoDB for storing users, messages, and groups  
3. Implement WebSocket communication using Socket.IO  
4. Build React.js frontend with Tailwind CSS  
5. Implement one-to-one and group chat UIs  
6. Add file sharing with uploads API  
7. Enable online/offline presence updates

# 9. Tools & Technologies

- Frontend: React.js (JavaScript + JSX), Tailwind CSS  
- Backend: Node.js, Express.js  
- Database: MongoDB  
- Communication: Socket.IO (WebSockets)  
- Other: Multer (for file uploads), REST APIs

# 10. Work Done So Far

✅ Backend server setup  
✅ MongoDB database integration  
✅ User authentication and presence tracking  
✅ One-to-one chat  
✅ Group chat  
✅ File sharing module  
✅ React frontend with chat UI

# 11. Work to be Completed

- UI enhancements (emojis, notifications)  
- Improve file sharing (preview, multiple files)  
- Testing and bug fixing  
- Deployment to cloud platform

# 12. Challenges Faced & Solutions

- Challenge: Handling concurrent users → Solution: Used Socket.IO with rooms  
- Challenge: File uploads → Solution: Implemented Multer with secure storage  
- Challenge: Maintaining presence → Solution: MongoDB + WebSocket sync

# 13. Conclusion

The Realtime Chat Application demonstrates the successful implementation of a feature-rich, scalable chat system using the MERN stack. It replicates key WhatsApp-like features such as one-to-one messaging, group chats, file sharing, and presence tracking.