COLLEGE CODE : 8207

COLLEGE NAME : As-salam college of engineering & technology

DEPARTMENT : BE.cse

STUDENT NM-ID : CAD1F561E731681BD594959D7FE60460
87EF5A85BBEB61D8976425EBA872DA33
F390F428FD893C4BE424ACBA32070897
200B4070228446E966344A664E859F31
436826D61FA7F1DBC724561838742D81

ROLL NO : 820723104010
820723104011
820723104012
820723104017
820723104303

GITHUB LINK

HTTPS://GITHUB.COM/MALARVIZHI0602/REAL-TIME-CHAT-APPLICATION.GIT

HTTPS://GITHUB.COM/PRAMIP05-CLOUD/NJ-REAL-TIME-CHAT-APPLICATION-.GIT

DATE : 3.10.2025

Completed the project named as Phase__

TECHNOLOGY PROJECT NAME : NJ REAL TIME CHAT APPLICATION

SUBMITTED BY,

NAME	MOBILE NO
KEERTHIKA.V	8681988168
KALAIRAGINI.A	9087414409
MALARVIZHI.S	7806994345
PRAMI.A	6385794773
KIRUTHIGA.R	6382926028

Node.js Real-Time Chat Application – Enhancements & Deployment

Enhancements & Additional Features

- Private rooms / channels
- Usernames + presence (online status)
- Message history (MongoDB / Redis)
- Typing indicators, read receipts, message edits/deletes
- File / image sharing (S3-compatible uploads)
- Notifications (browser push, desktop)

A/UX Improvements

- Responsive chat UI (mobile-first)
- Message grouping, timestamps, scroll-to-latest
- Offline handling + retry
- Accessibility improvements
- Smooth animations for message add/remove

API Enhancements

- REST endpoints for auth, users, messages, rooms
- WebSocket events: join/leave, message send/ack
- Rate limiting & pagination for history
- Stronger validation and sanitization

Performance & Security Checks

- Rate limiting, input sanitization, message size cap
- CORS & HTTPS enforcement
- Redis for session/pubsub for scaling
- Load testing with k6, logging & monitoring

Testing of Enhancements

- Unit tests (Jest)
- Integration tests (supertest + in-memory DB)
- End-to-end tests (Playwright/Cypress)
- Load tests (k6 concurrent users)

Deployment Options (Pros & Cons)

- Netlify/Vercel (Pros: easy frontend deploy, global CDN. Cons: not ideal for long-lived WebSocket servers).
- Render/Fly/Heroku/DigitalOcean (Pros: WebSocket support, simple scaling. Cons: less integrated frontend).

- Managed realtime providers (Ably, Pusher, Supabase) (Pros: hassle-free scaling, reliability. Cons: external dependency, cost).

Sample Code (Server.js)

```
const express = require('express');
const http = require('http');
const { Server } = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = new Server(server);

io.on('connection', socket => {
  console.log('User connected:', socket.id);
  socket.on('message:send', (msg) => {
    io.emit('message:recv', msg);
  });
});

server.listen(3000, () => console.log('Server running on http://localhost:3000'));
```

Example Output (Terminal)

```
Server running on http://localhost:3000
User connected: abc123
User connected: def456
```