Real-Time Chat Application Comprehensive Technical Documentation

1. Introduction

This document provides a full technical overview of a real-time chat application built using Flutter for the frontend and Node.js, Express, MongoDB, and Socket.io for the backend. It includes system architecture, modules, data models, APIs, WebSocket events, features, and security mechanisms.

2. System Architecture Overview

The system follows a client–server architecture with bi-directional communication using WebSockets for real-time messaging.

Architecture Flow: • Flutter Mobile App → REST API (Express.js) • Flutter Mobile App → WebSocket (Socket.io) • Backend Server → MongoDB Database • Media Uploads handled via Multer

Technology Stack: - Frontend: Flutter, GetX, HTTP, Socket.io client, secure storage, media handling plugins. - Backend: Node.js, Express, Socket.io, MongoDB, Mongoose. - Security & Utilities: JWT Authentication, bcrypt.js hashing, Helmet, CORS, Multer, Rate Limiting.

3. Application Features

The application includes a complete suite of modern messaging features:

• User authentication (email/phone login, password reset, profile update) • Direct one-to-one messaging • Group chat system with admin roles • Real-time messaging with Socket.io • Message reactions and threaded replies • Message editing and soft delete • Typing indicators • Read receipts • Online/offline presence system • Multimedia sharing: images, videos, audio, documents • Built-in user status (Online, Offline, Busy, Away) • Built-in calculator module • Notification system for messages and group updates

4. Database Models (MongoDB with Mongoose)

Main collections:

1. User: - username, email, phone - hashed password - profile image - status (online/offline) - last seen timestamp

2. Message: - senderId, receiverId or groupId - content (text/media) - messageType (text, image, video, audio, file) - reactions (emoji) - replyTo message - readBy[] list for read receipts

3. Group: - name, image - members[] - admins[] - createdAt timestamp

5. REST API Endpoints

Authentication APIs: • POST /auth/register – Register user • POST /auth/login – Login using email/phone • POST /auth/reset-password – Request password reset • PUT /auth/update-profile – Update user profile

User APIs: • GET /users – List all users • GET /users/:id – Get user profile • PUT /users/status – Update online/offline status

Message APIs: • POST /messages/send – Send message • GET /messages/:userId – Fetch chat history • GET /messages/group/:groupId – Fetch group messages

Group APIs: • POST /groups/create • PUT /groups/add-member • PUT /groups/remove-member • PUT /groups/update-admins

6. Real-Time WebSocket Events

Client → Server Events: • connect_user • send_message • typing • join_group • leave_group

Server → Client Events: • new_message • message_updated • message_deleted • user_typing • user_status_changed • group_updated

7. Security Mechanisms

• JWT token authentication for all protected routes • Password hashing using bcryptjs • Helmet for HTTP header protection • CORS restrictions for safe domains • Rate limiting to prevent brute-force attacks • Input validation for all API endpoints • Token expiration and refresh token support • Secure file upload handling using Multer

8. Flutter App Structure

Architecture Pattern: GetX + MVC

Main modules: • Authentication • Home & Chat List • Chat Screen • Group Chat • Profile • Settings • Media Preview • Calculator Module

State Management: • GetX Controllers for Business Logic • Reactive Observables • Dependency Injection

9. Testing Strategy

Unit Tests: • Authentication validation • Message encoding/decoding • API wrappers • Socket handlers

Integration Tests: • User registration + login • Sending text, image, video • Group creation + member management

Load Testing: • Simulated 500 concurrent chat users • Stress test for group chat broadcasting

10. Deployment Pipeline

Backend Deployment: • Deployed on Vercel / Render / AWS EC2 • Environment variables stored securely • Reverse proxy using NGINX if needed

Frontend Deployment: • Flutter build APK/AAB • Optional Flutter Web deployment

11. Conclusion

This documentation captures the full technical details of the real-time chat application. The system is scalable, secure, modular, and extensible. It follows best practices in both backend and frontend development, ensuring smooth real-time communication and performance.