# Team Collaboration Chat Application

## 1. Project Overview

A web-based **Team Collaboration Chat Application** that enables real-time communication among team members. The system supports instant messaging using **WebSockets**, persistent chat history using **MongoDB**, and **admin tools** for managing users, teams, and conversations.

**Tech Stack**

- Frontend: React.js, Tailwind CSS, Socket.IO Client
- Backend: Node.js, Express.js, Socket.IO
- Database: MongoDB (Mongoose ODM)
- Authentication: JWT (JSON Web Tokens)

## 2. Requirements

### 2.1 Functional Requirements

- User registration and login
- Real-time 1-to-1 and group messaging
- Persistent chat history
- Online/offline user status
- Admin role with elevated privileges
- Create and manage teams/groups
- Remove or block users (admin)
- Message timestamps and sender info

### 2.2 Non-Functional Requirements

- Low-latency message delivery
- Scalable to multiple teams
- Secure authentication and authorization
- High availability and fault tolerance
- Responsive UI for desktop and mobile

## 3. System Architecture (High-Level Design)

### 3.1 Architecture Overview

The system follows a **client-server architecture** with real-time communication.

- React Client communicates with:
  - Express REST APIs (HTTP)
  - WebSocket Server (Socket.IO)
- Express Server handles:
  - Authentication

    - Business logic
    - WebSocket events
- MongoDB stores persistent data

```
[ React + Tailwind ]
        |
   REST APIs / WebSocket
        |
[ Express + Socket.IO ]
        |
     MongoDB
```

## 4. Low-Level Design

### 4.1 Database Design (MongoDB)

**User Collection**

- _id
- name
- email
- passwordHash
- role (admin/user)
- status (online/offline)
- createdAt

**ChatRoom Collection**

- _id
- name
- type (private/group)
- members [userId]
- createdBy

**Message Collection**

- _id
- chatRoomId
- senderId
- content
- timestamp

### 4.2 API Design (REST)

| Method | Endpoint | Description |
|---|---|---|
| POST | /api/auth/register | Register user |
| POST | /api/auth/login | Login user |
| GET | /api/chats | Get user chat rooms |
| GET | /api/messages/:chatId | Fetch chat history |

| Method | Endpoint | Description |
|---|---|---|
| POST | /api/admin/block | Block user |

### 4.3 WebSocket Events

| Event | Direction | Description |
|---|---|---|
| connect | Client → Server | User connects |
| join_room | Client → Server | Join chat room |
| send_message | Client → Server | Send message |
| receive_message | Server → Client | Receive message |
| typing | Client → Server | Typing indicator |
| disconnect | Client → Server | User disconnects |

## 5. Security Design

- JWT-based authentication
- Role-based access control (Admin/User)
- WebSocket authentication via token
- Password hashing using bcrypt

## 6. Project Module Breakdown (8 Modules)

### Frontend Modules (3)

#### 1. Authentication & User Interface Module

- Login & Register pages
- JWT storage and auth context
- Tailwind-based responsive UI

#### 2. Chat Interface Module

- Chat window UI
- Real-time message rendering
- Typing indicator and online status

#### 3. Team & Admin Dashboard Module

- Team creation UI
- User management UI (admin)
- Chat room list and navigation

### Backend Modules (5)

#### 4. Authentication Service Module

- User registration and login
- JWT token generation and validation
- Password hashing

**5. User & Team Management Module**

- CRUD operations for users
- Team and chat room creation
- Role-based access control

**6. Messaging Service Module**

- Store messages in MongoDB
- Fetch chat history
- Message validation

**7. Real-Time WebSocket Module**

- Socket.IO server setup
- Room management
- Message broadcasting
- Online/offline tracking

**8. Admin & Moderation Module**

- Block/unblock users
- Delete messages
- Monitor chat activity

## 7. Scalability Considerations

- Horizontal scaling using Node.js clustering
- Redis adapter for Socket.IO
- Database indexing on chatRoomId and timestamp

## 8. Future Enhancements

- File and image sharing
- Message encryption (E2EE)
- Push notifications
- Read receipts
- Audit logs for admin