Requirement Document

Project Overview

- **Objective:** Build a real time chat Application with JWT authentication and Redux Toolkit and Web sockets.

- **Target Audience:** Users who want to connect with friends, family, or colleagues for casual conversations, sharing updates, or planning events.

Functional Requirements

1. **Authentication:**
    - Implement secure authentication using JWT for user login.
    - Include user profile management features, such as updating profile information, changing passwords, and signing out.
2. **Chat Features:**
    - Users can create new chat conversations with other users.
    - Implement basic CRUD functionality for chats (Create, Read, Update, Delete).
    - Enable users to search for existing chats based on various criteria.
3. **Messaging:**
    - Users can send messages within a chat conversation.
    - Implement real-time messaging or notifications for instant communication between users.
    - Include features such as message editing and deleting.
4. **Frontend:**
    - Use React.js for the frontend development of the chat application.
    - Implement a responsive and user-friendly design for the chat interface using CSS.
    - Utilize React components for modular and efficient UI development.
    - Implement features like message input and user profiles in the frontend.
5. **Security Considerations:**
    - Ensure that user authentication tokens are stored securely and have proper expiration.
    - Implement measures to prevent common security vulnerabilities, such as cross-site scripting (XSS) and cross-site request forgery (CSRF).
6. **User Interaction:**
    - Users can initiate new chat conversations with other users.
    - Users can Create a new group for one-to-many communication.

Non-Functional Requirements

- **Performance:** Ensure the application is responsive and can handle a reasonable number of concurrent users.

- **Security:** Implement secure authentication and protect user data.

Technology Stack

- MongoDB for the database
- Express.js as the backend framework
- React.js for the frontend
- Node.js as the runtime environment for the backend
- Mongoose as the ODM (Object Data Modeling) library for MongoDB
- JWT for authentication
- Redux Toolkit for state management
- CSS for styling.
- Material Ui for components.

Deployment

- Deploy the application to a hosting service (e.g., Render).

Design Document

System Architecture

- Client-Server architecture with React.js as the frontend and Express.js/Node.js as the backend.
- Use Redux Toolkit for state management.

Database Schema

- User table: id, username, email, password, etc.
- Group Schema: This table could be used to manage chat groups and their administrators..
- Chat Schema : This table could be used to establish relationships between users or groups involved in a chat.
- Message Schema : This table could be used to store individual messages sent within a chat.

Authentication Flow

- User registration and login with JWT tokens.
- Secure routes and API endpoints requiring valid JWT tokens.

User Interface Design

- Design responsive and intuitive UI using React.js and MaterialUI and CSS.
- Include pages for Chats, user profile, search, etc.

API Endpoints

- Define API endpoints for user authentication, CRUD operations on groups, messaging, etc.

Deployment Strategy

Testing

- Define testing strategies for both frontend and backend components.

Scalability

- Consider potential future scalability requirements and design the system to handle increased user loads.