System Design: Chat Application(JustChat)

Introduction

This chat application is a communication platform designed for one-on-one and group chats. It offers features such as real-time messaging.

System Architecture

- The application employs a client-server architecture.
- Key components include Messaging Service, Session Service, Relay Service, Group Messaging Service.
- Real-time communication is facilitated by the Socket.IO library.

Technologies Used

Backend:

- Node.js for server-side development
- Express.js as the web application framework
- Socket.IO for real-time, bidirectional communication
- MongoDB (NoSQL) for data storage
- JSON Web Tokens (JWT) for authentication

Frontend:

- React for building the user interface
- Chakra UI for a modern, responsive design
- Socket.IO client for real-time messaging

API Endpoints

User Management:

- **Search Users**: Search for users by name or email.
- **Create One-on-One Chat**: Create one-on-one chat conversations.

- **Create Group Chat**: Create group chats.
- **Rename Chat**: Modify chat names.
- **Add/Remove Users**: Manage group chat members.

Messaging:

- **Send Messages (One-on-One)**: Send real-time messages in one-on-one chats.
- **Send Messages (Group Chat)**: Send messages in group chats.
- **Receive Messages**: Implement real-time message delivery with React and Socket.IO.

System Components

Messaging Service

- Establishes real-time WebSocket connections using Socket.IO.
- Routes messages between users and groups.
- Utilizes JWT for authentication and authorization.

Session Service

- Stores user-server mapping for message routing.
- Ensures efficient message delivery to online users..

Group Messaging Service

- Handles group chat messages.
- Utilizes the session service for routing to group members.
- Enables group management APIs.

Database Schema

- MongoDB (NoSQL) is used for data storage.
- Supports flexible data structures for chat messages, user profiles, and group information.

Setting Up the Prototype

Prerequisites:

To set up this Chat Application prototype, you will need the following prerequisites:

- Node.js: Ensure that Node.js is installed on your system.
- MongoDB: Install MongoDB for database storage or use the MongoDb atlas
- Code Editor: Choose a code editor of your preference (e.g., Visual Studio Code).

Installation:

Follow these steps to install and set up the prototype:

1. **Clone the Repository/ Download:** Clone the project repository from GitHub or download it to your local machine.
2. **Install Backend Dependencies:** Navigate to the backend directory and install the required Node.js packages by the following command.
    - *cd backend npm install*
3. **Install Frontend Dependencies:** Go to the frontend directory and install the React dependencies by following command.
    - *cd frontend npm install*

Configuration:

Before running the prototype, you need to configure the following:

- **Backend Configuration:**
    - Create a **.env** file in the **root** directory to store environment variables like the MongoDB connection string, JWT secret, PORT and other sensitive information.

      Example **.env** file:

      PORT=

MONGO_URI=

JWT_SECRET=

NODE_ENV=

- **Frontend Configuration:**
  - The frontend typically doesn't require separate configuration but ensure that the backend API URLs match your development environment in the frontend code.

Running the Prototype:

To run the Chat Application prototype, follow these steps:

1. **Start the Backend Server:** In the **root** directory, start the Node.js server by the following command in terminal.
   - *npm start*

   The backend server will run on port 5000 by default or the port no. you mentioned in the .env file.
2. **Start the Frontend Development Server:** In the **frontend** directory, start the React development server by the following command.
   - *npm start*

   The frontend development server will run on port 3000 by default.

Access the application by opening a web browser and navigating to **http://localhost:3000**.

Dependencies and Libraries

Frontend Dependencies:

The frontend of this Chat Application relies on the following dependencies and libraries:

- **React:** Used as the JavaScript library for building the user interface.
- **Chakra UI:** Provides a modern and responsive design framework for building the UI components.
- **Socket.IO Client:** Used for real-time messaging and WebSocket connections.

Backend Dependencies:

The backend of the application relies on the following Node.js packages:

- **Node.js and Express.js:** The core backend technologies for handling API requests and server-side logic.
- **Socket.IO:** Implements real-time, bidirectional communication between the server and clients.
- **MongoDB:** A NoSQL database used for data storage, including chat messages, user profiles, and group information.
- **JSON Web Tokens (JWT):** Provides authentication and authorization for secure access to the application's features.

Database:

The application utilizes MongoDB as the primary database system. MongoDB is a NoSQL database that allows flexible data storage and retrieval. It is suitable for storing chat messages, user profiles, and group information, providing scalability and performance.