**ANALYSE COMPLÈTE DU PROJET CHATIFY**

**ARCHITECTURE GÉNÉRALE**

* **Frontend** : React.js avec Chakra UI
* **Backend** : Node.js avec Express.js
* **Base de données** : MongoDB avec Mongoose
* **Communication temps réel** : Socket.io
* **Authentification** : JWT (JSON Web Tokens)
* **Stockage d'images** : Cloudinary

**👥 ENTITÉS PRINCIPALES**

**1. USER (Utilisateur)**

**Attributs :**

* \_id : ObjectId (clé primaire)
* name : String (nom de l'utilisateur)
* email : String (email unique)
* password : String (mot de passe hashé)
* pic : String (URL de l'avatar)
* isAdmin : Boolean (statut administrateur)
* createdAt : Date
* updatedAt : Date

**Méthodes :**

* matchPassword(enteredPassword) : Vérification du mot de passe
* pre("save") : Hashage automatique du mot de passe

**2. CHAT (Conversation)**

**Attributs :**

* \_id : ObjectId (clé primaire)
* chatName : String (nom du chat)
* isGroupChat : Boolean (chat de groupe ou individuel)
* users : Array[ObjectId] (référence vers User)
* latestMessage : ObjectId (référence vers Message)
* groupAdmin : ObjectId (référence vers User, admin du groupe)
* createdAt : Date
* updatedAt : Date

**3. MESSAGE (Message)**

**Attributs :**

* \_id : ObjectId (clé primaire)
* sender : ObjectId (référence vers User)
* content : String (contenu du message)
* chat : ObjectId (référence vers Chat)
* readBy : Array[ObjectId] (utilisateurs qui ont lu le message)
* createdAt : Date
* updatedAt : Date

**🔧 FONCTIONNALITÉS PAR MODULE**

**A. GESTION DES UTILISATEURS**

1. **Inscription** (POST /api/user)

* Création d'un nouveau compte
* Validation des données
* Hashage du mot de passe
* Génération du token JWT

1. **Connexion** (POST /api/user/login)

* Authentification par email/mot de passe
* Vérification des credentials
* Retour du token JWT

1. **Recherche d'utilisateurs** (GET /api/user?search=)

* Recherche par nom ou email
* Filtrage des utilisateurs connectés

1. **Mise à jour du profil** (PUT /api/user/profile)

* Modification de l'avatar
* Mise à jour des informations

**B. GESTION DES CONVERSATIONS**

1. **Accéder à une conversation** (POST /api/chat)

* Création ou récupération d'un chat individuel
* Vérification de l'existence

1. **Récupérer les conversations** (GET /api/chat)

* Liste des chats de l'utilisateur
* Tri par date de dernière activité

1. **Créer un groupe** (POST /api/chat/group)

* Création d'un chat de groupe
* Ajout des membres
* Définition de l'administrateur

1. **Renommer un groupe** (PUT /api/chat/rename)

* Modification du nom du groupe

1. **Gérer les membres**

* Ajouter un membre (PUT /api/chat/groupadd)
* Retirer un membre (PUT /api/chat/groupremove)

**C. GESTION DES MESSAGES**

1. **Envoyer un message** (POST /api/message)

* Création d'un nouveau message
* Mise à jour du dernier message du chat

1. **Récupérer les messages** (GET /api/message/:chatId)

* Historique des messages d'un chat
* Population des informations utilisateur

**D. FONCTIONNALITÉS TEMPS RÉEL (Socket.io)**

1. **Connexion WebSocket**

* Setup de l'utilisateur
* Rejoindre les salles de chat

1. **Typing Indicator**

* Indicateur de frappe en temps réel
* Gestion des événements typing/stop typing

1. **Messages en temps réel**

* Réception instantanée des messages
* Notifications pour les chats non sélectionnés

**E. FONCTIONNALITÉS AVANCÉES**

1. **Upload d'images**

* Intégration Cloudinary
* Envoi d'images dans les messages

1. **Enregistrement audio**

* Capture audio via MediaRecorder API
* Upload et envoi de messages audio

1. **Emojis**

* Sélecteur d'emojis intégré
* Ajout d'emojis aux messages

1. **Notifications**

* Système de notifications pour messages non lus
* Filtrage des chats avec notifications

**�� ACTEURS IDENTIFIÉS**

1. **Utilisateur non connecté** : Peut s'inscrire et se connecter
2. **Utilisateur connecté** : Peut utiliser toutes les fonctionnalités
3. **Administrateur de groupe** : Peut gérer les membres du groupe
4. **Système de notification** : Gère les notifications temps réel

**📱 INTERFACES UTILISATEUR**

1. **Page d'accueil** : Login/Signup
2. **Page de chat** : Interface principale
3. **Liste des chats** : Navigation entre conversations
4. **Zone de chat** : Affichage et envoi de messages
5. **Modales** : Création de groupe, profil, etc.

**🔐 SÉCURITÉ**

* Authentification JWT
* Middleware de protection des routes
* Hashage des mots de passe
* Validation des données
* Gestion des erreurs

**📝 SCRIPT POUR CHATGPT - GÉNÉRATION DES DIAGRAMMES UML**

Voici le script que vous pouvez utiliser avec ChatGPT pour générer les diagrammes UML :

Je travaille sur un projet de chat en temps réel appelé "Chatify" avec les spécifications suivantes :

ARCHITECTURE :

- Frontend : React.js avec Chakra UI

- Backend : Node.js avec Express.js

- Base de données : MongoDB avec Mongoose

- Communication temps réel : Socket.io

- Authentification : JWT

ENTITÉS PRINCIPALES :

1. USER (Utilisateur)

- Attributs : \_id, name, email, password, pic, isAdmin, createdAt, updatedAt

- Méthodes : matchPassword(), pre("save")

2. CHAT (Conversation)

- Attributs : \_id, chatName, isGroupChat, users[], latestMessage, groupAdmin, createdAt, updatedAt

3. MESSAGE (Message)

- Attributs : \_id, sender, content, chat, readBy[], createdAt, updatedAt

FONCTIONNALITÉS PRINCIPALES :

- Inscription/Connexion utilisateur

- Recherche d'utilisateurs

- Création de chats individuels et de groupe

- Envoi/réception de messages en temps réel

- Gestion des groupes (ajout/retrait de membres, renommage)

- Upload d'images et enregistrement audio

- Notifications temps réel

- Indicateur de frappe

- Emojis dans les messages

ACTEURS :

- Utilisateur non connecté

- Utilisateur connecté

- Administrateur de groupe

- Système de notification

Peux-tu me générer les diagrammes UML suivants pour ce projet :

1. Diagramme de cas d'utilisation (Use Case)

2. Diagramme de séquence pour les principales fonctionnalités

3. Diagramme de classes

4. Diagramme d'activité pour le processus de chat

Utilise PlantUML pour générer le code de ces diagrammes.