**TALK-A-TIVE Design Document**

**Introduction**

Talk-A-Tive is a Full Stack Real Time Chatting App. The App provides a platform for users to send and receive messages in real time. MERN (MongoDB, Express.js, React.js, Node.js) stack is used for building the website and Socket.io to create web sockets for real time communication.

**Features**

1. Register with a username, email, and password.

2. Send and receive messages in real time.

3. Enable private and group chat functionalities.

4. Manage user profiles, including avatars and personal information.

5. Encrypted data storage in MongoDB.

6. Typing Indicators.

7. Notifications for incoming messages.

8. Search functionality to find user profiles.

**Non-Functional Requirements**

1. Performance: The website should load quickly and handle concurrent user requests efficiently.

2. Security: Authentication and Authorization mechanisms to protect user data and secure access.

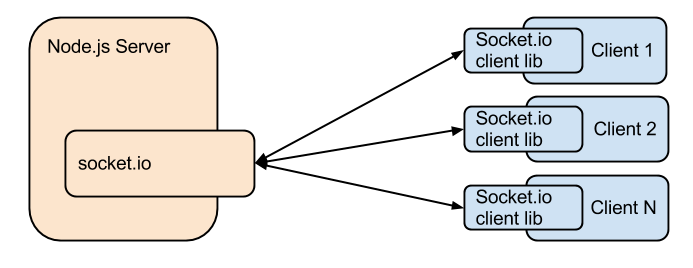
3. User Experience: User friendly and responsive User Interface design for both web and mobile devices.

4. Maintainability: Write clean, modular, and well-documented code for easy updates and maintenance.

**Architecture View**

The Talk-A-Tive website follows a typical MERN stack architecture:

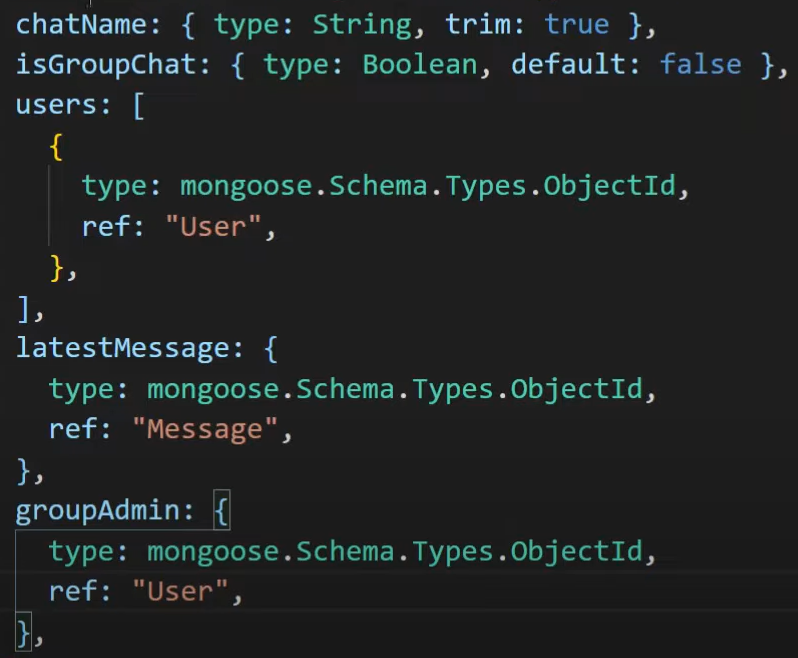
* Frontend: React.js is used for creating the user interface.
* Backend: Node.js and Express.js for handling server-side logic and Socket.io for real time communication.
* Database: MongoDB for user profiles, authentication and message data.
* Deployment: The application is deployed on the Heroku Cloud Platform.



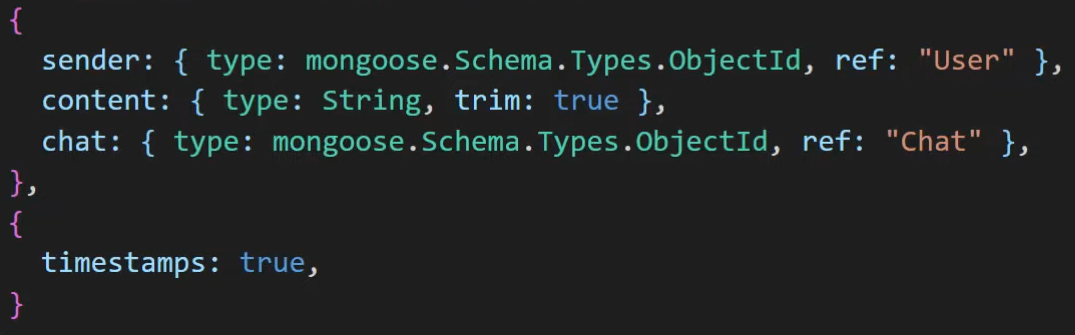
**Database design**

Mongoose is used to define the schema for the following MongoDB collections:

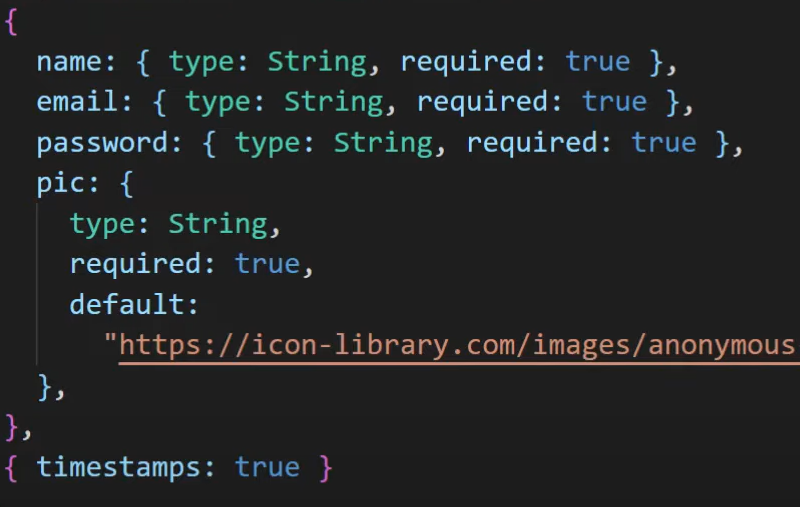
1. Chat: Stores the information about a single one-to-one or group chat.



2. Message: Stores information about a single message like sender, content and time.



3. User: Stores details of single user such as name, email, avatar



**API Design**

The website utilizes the following API endpoints.

1. User:

* POST /api/login
* POST /api/register
* GET /api/users/{userId}

2. Chats

* GET /api/chats
* GET /api/chats/{chatId}/messages
* POST /api/chats/{chatId}/messages
* DELETE /api/chats/{chatId}/messages/{messageId}
* GET /api/chats/groups
* POST /api/chats/groups
* POST /api/chats/groupadd
* POST /api/chats/groupremove

3. Message

* POST /api/message
* GET /api/message/:chatid

**Frontend Design**

The frontend of Talk-A-Tive is designed using React.js and styled with Chakra Ui components to create a modern and user-friendly user interface. Components are organized hierarchically to ensure reusability and maintainability.

- Conversation List: Display user avatars, names, and last messages.

- Chat Window: Show messages with timestamps, support text formatting.

- User Profile: Allow users to view and edit profiles.

- Responsive Design: Optimize for desktop and mobile.

**Security**

- Authentication: Implement JWT-based authentication to authenticate users.

- Authorization: Use role-based access control (RBAC) to restrict options for group admins.

- Input Validation: Sanitize and validate user input to prevent injection attacks and other security vulnerabilities.

- Secure Communication: Use HTTPS protocol to encrypt data transmission between client and server.

**Deployment**

The frontend of the Talk-A-Tive website can be deployed on a cloud platform such as Netlify or Heroku. MongoDB Atlas can be used for hoisting the database.

**Conclusion**

Talk-A-Tive provides an user-friendly platform for real time communication. Utilising the MERN stack along with ChakraUI helps create a robust and scalable architecture. The real-time capabilities provided by technologies like Socket.io can enhance the user experience, ensuring instant message delivery which all together helps with the goal of providing a rich and seamless user experience.