System Design Document

Real-Time Chat Application (OMNAMO Chat)

Deployed Link: https://omnamo-chat-rs.onrender.com/

Since, it is free deploy service it takes some time open. Sorry for the inconvenience.

Demo Video Link:
https://www.loom.com/share/00e3bb1200f342128d13f6d526555240?t=22&sid=ea3dc333-1bb5-431a-995a-0950df5a6e05

Overview:

OMNAMO Chat is a real-time chat application designed to facilitate smooth and secure communication between users. It supports both one-on-one messaging and group chats with real-time updates. The application uses a **NoSQL database (MongoDB)** for data storage and real-time communication with **Socket.IO.**

Key Features:

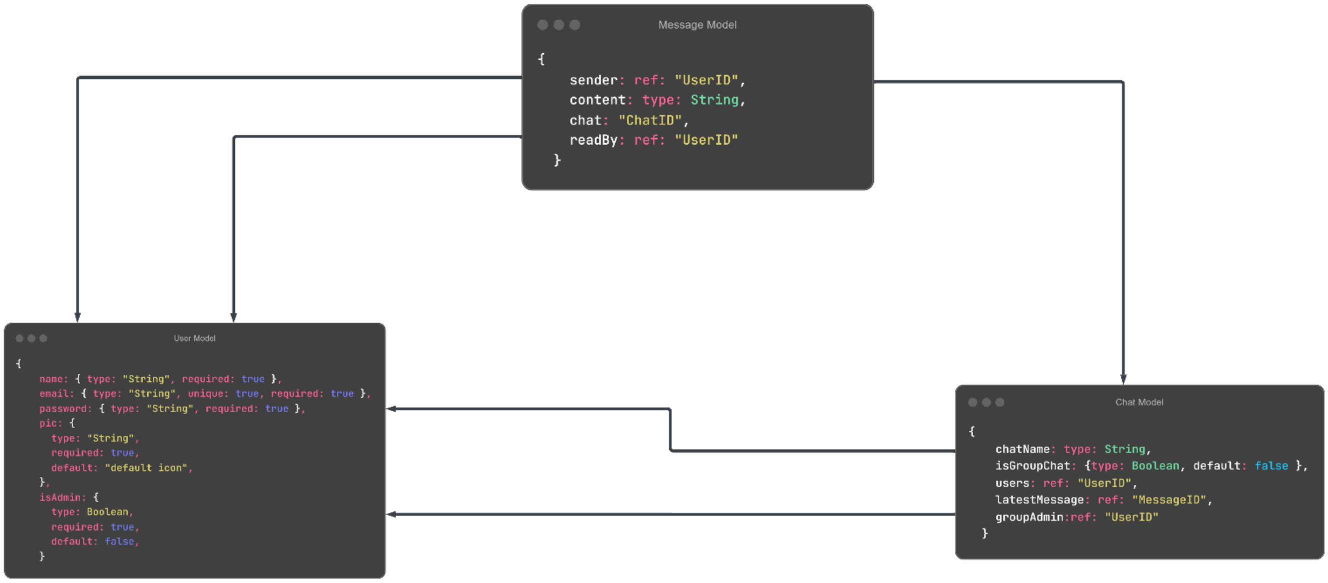
- **Real-Time Messaging**: Real-time, low-latency messaging using Socket.IO.
- **User Authentication**: Secure user authentication using JWT (JSON Web Tokens) and password hashing with bcrypts.
- **Group Chats**: Support for group chats with dynamic participant management.
- **Responsive UI**: Built with **Chakra UI** for responsive and accessible design.
- **Persistent Chat History**: Messages are stored in MongoDB for persistent chat history.

System Architecture:

The system is built on a **Client-Server Architecture**:

1. **Frontend**: Developed using **ReactJS** for an interactive UI with Chakra UI components and real-time chat features.
2. **Backend**: Built on **Node.js** with **Express.js** for handling API routes and real-time features using **Socket.IO**.
3. **Database**: **MongoDB** is used as the database to store user profiles, chat messages, and chats.

Data Model:



```
{
    sender: ref: "UserID",
    content: type: String,
    chat: "ChatID",
    readBy: ref: "UserID"
  }
```

```
{
    name: { type: "String", required: true },
    email: { type: "String", unique: true, required: true },
    password: { type: "String", required: true },
    pic: {
      type: "String",
      required: true,
      default: "default icon",
    },
    isAdmin: {
      type: Boolean,
      required: true,
      default: false,
    }
```

```
{
    chatName: type: String,
    isGroupChat: {type: Boolean, default: false },
    users: ref: "UserID",
    latestMessage: ref: "MessageID",
    groupAdmin:ref: "UserID"
  }
```

Technologies Used:

1. **Chakra UI**
   Provides a modern, accessible, and responsive design framework for React applications. Chakra UI's component library allows rapid UI development while ensuring responsiveness across devices.

2. **Axios**
   A promise-based HTTP client used to handle API requests and responses between the frontend (React) and backend (Express).

3. **Bcryptjs**
   A library used for hashing passwords before storing them in the database. Ensures that sensitive information like passwords are stored securely.

4. **Socket.IO**
   Enables real-time bidirectional communication between the client and server. This is used for implementing live chat functionality, where messages are transmitted in real time.

5. **JWT (jsonwebtoken)**
   Provides secure authentication using JSON Web Tokens. JWTs ensure that only authenticated users can access protected routes and resources.

6. **Mongoose**
   A MongoDB object modeling tool used to define data schemas, manage relationships, and perform queries easily.

7. **React.js**
   A popular JavaScript library for building dynamic and efficient user interfaces. React's component-based architecture allows modular and reusable code.

8. **React Router DOM**
   Enables navigation within the React app by offering routing capabilities for single-page applications.

9. **Nodemon**
   A tool that automatically restarts the Node.js server whenever changes are detected, aiding in efficient development.

10. **Dotenv**
    Manages environment variables like API keys and database URIs, which are stored in a .env file for security.

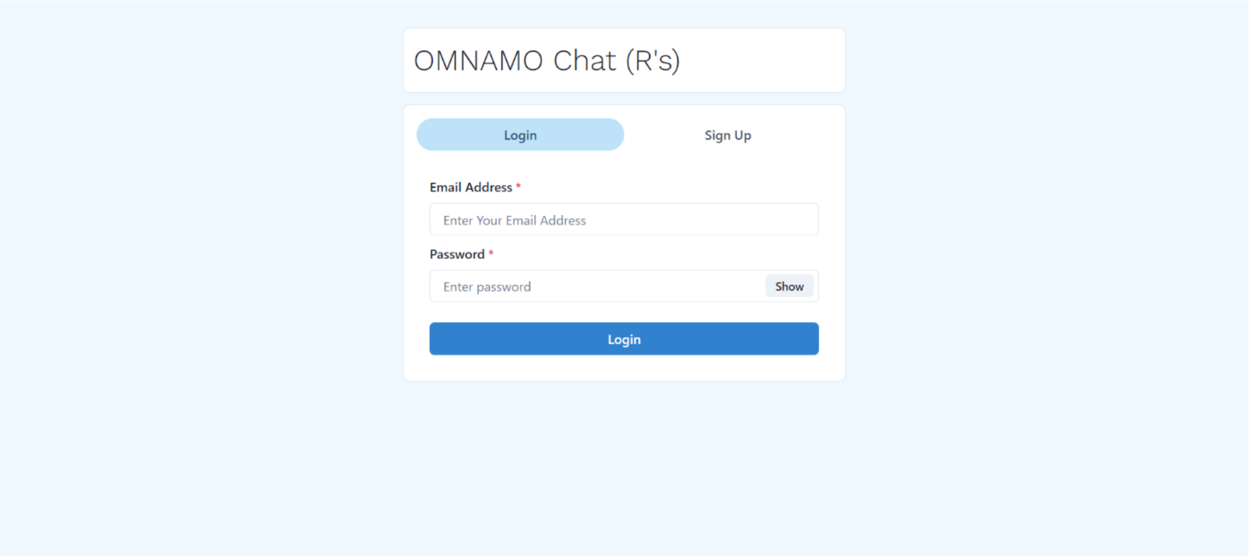
11. **Framer Motion**
    A React animation library used to provide smooth, declarative animations, enhancing the user experience.

12. **MongoDB**
    A NoSQL database suited for storing JSON-like data. MongoDB is ideal for this project due to its scalability and flexibility in handling dynamic chat data structures.

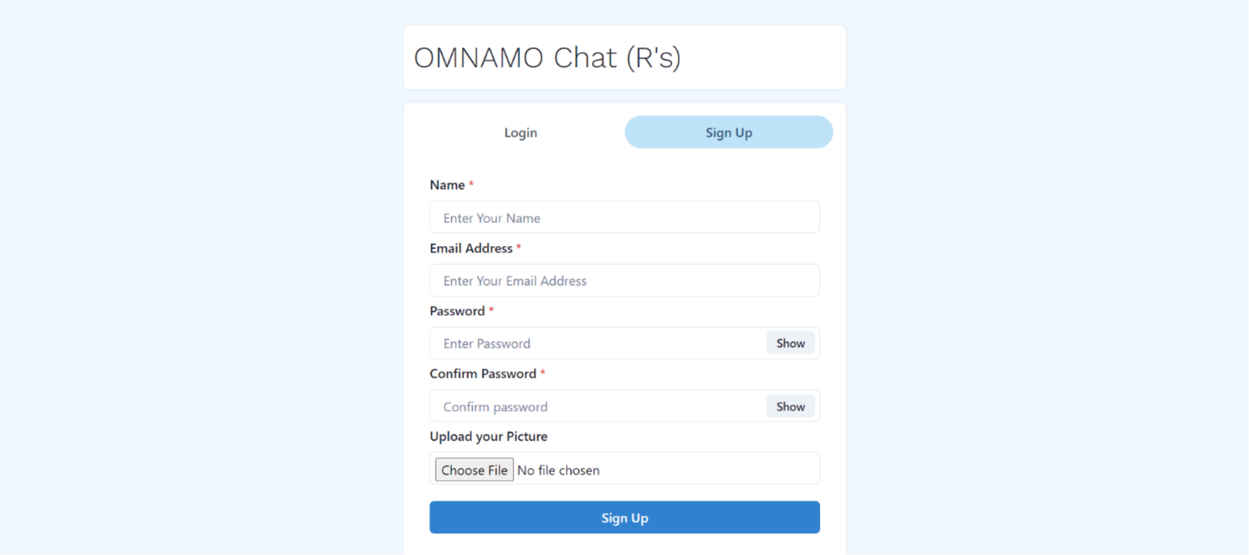
13. **Express**
    A lightweight web application framework for Node.js. It simplifies the creation of API endpoints and server-side logic.

Application Images:

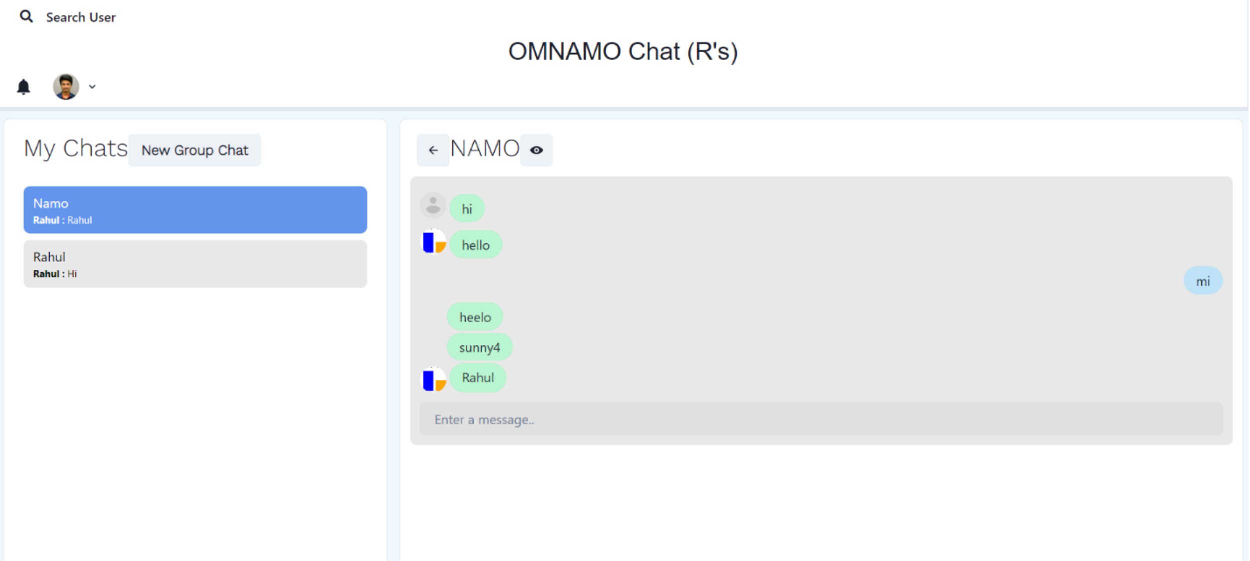
- Login



- Sign Up



- Chat Page



OMNAMO Chat (R's)

Namo
Rahul : Rahul

Rahul
Rahul : Hi

hi

hello

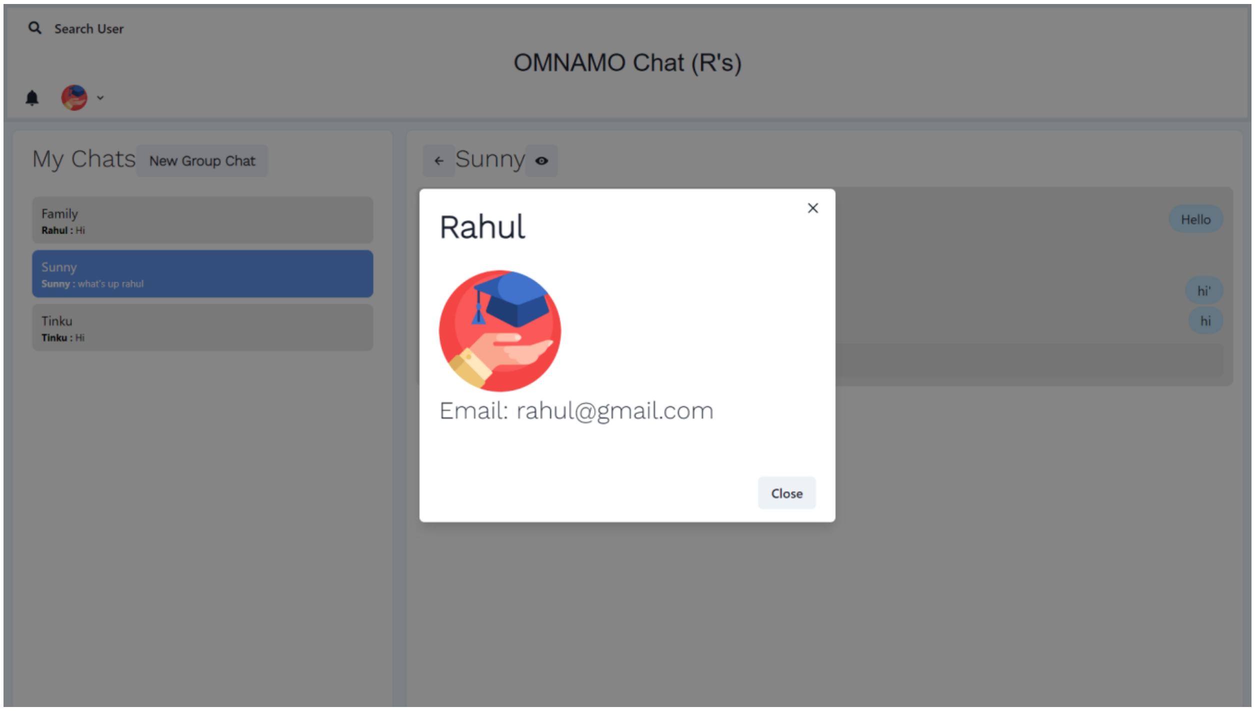
mi

heelo

sunny4

Rahul

- Profile



OMNAMO Chat (R's)

Family
Rahul : Hi

Sunny
Sunny : what's up rahul

Tinku
Tinku : Hi

Hello

hi'

hi

Rahul

Email: rahul@gmail.com

Setup/Installation

Prerequisites:

- **Node.js** (version >= 14)
- **MongoDB** (installed locally or using a cloud service like MongoDB Atlas)
- **NPM**

Steps for Cloning Repository and Running Application

- **Step-1**: Run the following command in bash for cloning Repo

```
git clone https://github.com/Rahulsree1/OMNAMO-chat-.git
```

- **Step-2**: Navigate to the directory

```
cd OMNAMO-chat--main
```

- **Step-3:** Install dependencies by running following commands

```
npm run build
```

- **Step-4:** Setup Environment Variables, by changing the .env file **or** you can run directly no need to change it will run on my database.

```
MONGO_URI=your_mongo_db_uri
JWT_SECRET=your_secret_key
```

- **Step-5**: Run the Application by typing following command.

```
npm start
```

-By

Manitha Rahul Sree

BTech 3rd year EE,

IIT Mandi.

Email: b22268@students.iitmandi.ac.in, m.rahulsree1122@gmail.com

Phone: +91 9581914996

LinkedIn: https://www.linkedin.com/in/manitha-rahul-sree-7435a1204/

GitHub: https://github.com/Rahulsree1