Detailed Design Document: ChatSphere

About ChatSphere:

ChatSphere is a messaging application designed for seamless communication between users. It allows users to create private chats, add contacts, and send messages in real-time. With features like Notification and Video Calling, users can stay connected and engaged effortlessly. ChatSphere ensures a user-friendly interface for an intuitive messaging experience.

1. Introduction

1.1 Purpose

This document aims to present a comprehensive overview of ChatSphere's design and architecture, encompassing features, functionality, and technical specifications.

1.2 Scope

ChatSphere, as a chatting app, endeavors to provide users with a dynamic platform for real-time communication, fostering seamless conversations, multimedia sharing, and interactive features to enhance the overall chatting experience.

1.3 Objectives

- Make it easy for users to chat by creating a simple and friendly interface in ChatSphere.
- Keep user information safe with strong security measures.
- Allow ChatSphere to grow and handle more users in the future.
- Let users work together by making it easy to share and collaborate on chats and activities..

2. System Overview

2.1 System Architecture

ChatSphere employs a client-server architecture with a responsive web-based front end for user interactions and a robust back-end server handling core logic, ensuring an efficient and dynamic chatting experience.

2.2 Key Features

- User registration and authentication
- Real-Time Chatting
- Notification System
- Video Calling
- Emoticons and Stickers

2.4 Technologies Used

- Front-end: React
- Back-end: Express
- Database: MongoDB
- Authentication: JWT
- Additional tools/libraries as needed.

3. Database Design

3.1 Entity-Relationship Diagram

User Entity:

- *UserID (Primary Key):* Unique identifier.
- *Username:* User's chosen name.
- *Email:* User's contact address.
- *Password:* Secure authentication.

Message:

- *Id (Primary Key):* Unique identifier..
- *SenderID:* Sender's UserID.
- *Data:* Message content.
- *SenderID:* Sender's UserID.

Friends:

- *Id(Primary Key):* Unique identifier.
- *UserID (Foreign Key):* Owner of the contact.
- *FriendId(Foreign Key):* UserID of the contact.

4. API Details

- **User Authentication and Authorization:**
  - `/api/auth/register`: Register a new user.
  - `/api/auth/login`: Log in an existing user.
  - `/api/auth/logout`: Log out the current user.
  - `/api/auth/reset-password`: Reset user password.
- **User Profile:**
  - `/api/users/{userId}`: Retrieve user profile information.
  - `/api/users/{userId}/update`: Update user profile information.
- **Messaging:**
  - `/api/messages/send`: Send a new message.
  - `/api/messages/{messageId}`: Retrieve a specific message.
  - `/api/messages/{userId}/inbox`: Retrieve the inbox of a user.
  - `/api/messages/{userId}/outbox`: Retrieve the outbox of a user.
  - `/api/messages/{messageId}/delete`: Delete a specific message.
- **Contacts:**
  - `/api/contacts/add`: Add a new contact.
  - `/api/contacts/{contactId}/remove`: Remove a contact.
  - `/api/contacts/{userId}/list`: List all contacts of a user.

5. Deployment Process

- ChatSphere can be deployed on a cloud platform such as AWS, Google Cloud Platform, or Microsoft Azure. The frontend can be hosted using services like Amazon S3 or Netlify, while the backend can be deployed on platforms supporting Node.js applications like AWS Elastic Beanstalk or Heroku. MongoDB Atlas can be used for database hosting.

4. Database Diagram

User:

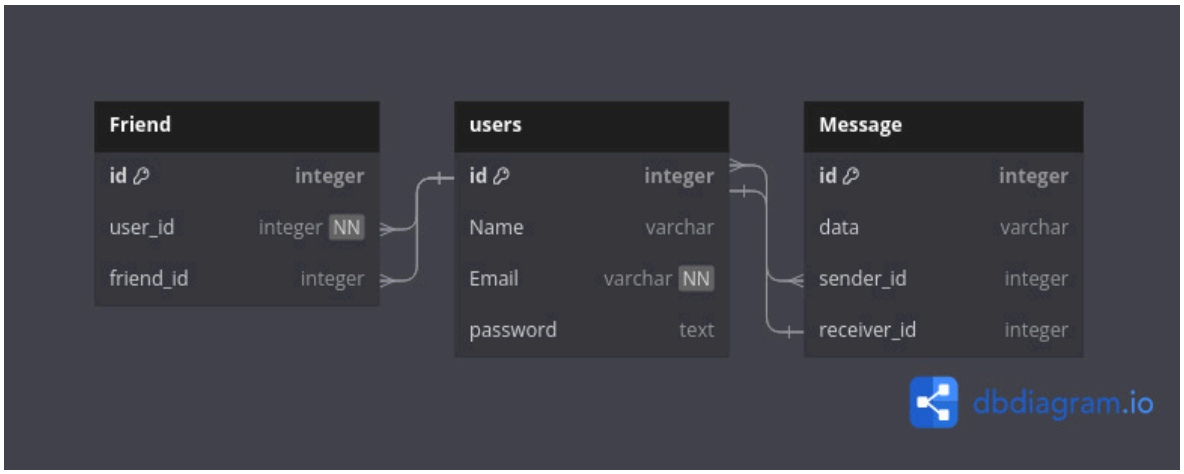
id	(INT)	PRIMARY KEY
name	(VARCHAR(255))	NOT NULL
email	(VARCHAR(255))	NOT NULL, UNIQUE
password	(TEXT)	NOT NULL

Message:

id	(INT)	PRIMARY KEY	
data	(TEXT)	NOT NULL	
senderID	(INT)	FOREIGN KEY	(FROM User.id)
receiverID	(INT)	FOREIGN KEY	(FROM User.id)

Friend:

id	(INT)	PRIMARY KEY	
userID	(INT)	FOREIGN KEY	(From User.id)
friendID	(INT)	FOREIGN KEY	(From User.id)



5. Flow Diagram