ChitChatHub: Messaging App

System Design Document

Table of Contents

1. Introduction

1.1 Purpose of the Document

The purpose of this extensive system design document is to provide a holistic understanding of the Messaging App's architecture, design, and functionality. It serves as a comprehensive reference guide for all stakeholders involved in the project.

1.2 Scope

This document encompasses the entire development lifecycle of the Messaging App, from system architecture and technology choices to security considerations, deployment strategies, and future enhancements. It aims to provide detailed insights into each aspect of the project.

2. System Architecture

2.1 Components

2.1.1 Client

The client side of the Messaging App is developed using React JS. It serves as the primary user interface and is responsible for rendering the application, handling user interactions, and facilitating real-time communication with the server using Socket.io. The client offers a dynamic and responsive user experience, making it accessible across various devices.

2.1.2 Server

The server is developed using Node JS and Express JS. It functions as the application's backend, managing user authentication, message storage, and client communication. The server exposes RESTful APIs to handle various client requests, ensuring a seamless user experience. MongoDB is chosen as the database system to store user data, messages, and application settings, providing efficient data retrieval and management.

2.1.3 Database

MongoDB, a NoSQL database, stores user data and messages. Its flexible schema design allows for efficient data storage and retrieval. Data is organised into collections to maintain data integrity, and indexes are used for optimised query performance.

2.2 Communication Flow

The communication flow within the Messaging App is designed to deliver a smooth and real-time user experience.

2.2.1 User Registration and Authentication

Users register or log in through the client application.
The client sends authentication requests to the server.
The server validates user credentials, generating JSON Web Tokens (JWT) upon successful authentication.
JWT tokens are returned to the client for secure user sessions and included in subsequent API requests.

2.2.2 Real-time Chatting

Users engage in real-time chat conversations using the client.
Socket.io, a WebSocket technology, facilitates instant message updates and typing indicators.
Messages are stored in MongoDB, ensuring message history retrieval and synchronisation across devices.

2.2.3 One-to-One and Group Chats

Users can initiate one-to-one private conversations or create group chats.
Group chat data is stored in MongoDB, enabling dynamic user additions and removals.
Socket.io ensures real-time message distribution within group chats.

3. Technologies and Tools

3.1 Frontend Technologies

The client application is built using React JS, a popular JavaScript library for building user interfaces. Additionally, Chakra UI is used for a responsive and visually appealing design.

3.2 Backend Technologies

The server is powered by Node JS, a runtime environment for executing JavaScript code server-side. Express JS is used as the web application framework to simplify server-side development.

3.3 Database Choice

MongoDB, a NoSQL database, is chosen for its flexibility and scalability. Its document-oriented structure allows for efficient data storage and retrieval.

3.4 Socket.io for Real-time Communication

Socket.io is utilised to provide real-time communication features, ensuring instant message updates and typing indicators.

4. User Authentication

4.1 User Registration

The user registration process is designed to be straightforward and secure. Users provide the necessary information, including a unique username and password. Passwords are securely hashed before being stored in the database to enhance security.

4.2 User Login

Registered users can log in by providing their credentials. The server validates the username and password against stored records. Upon successful login, a JWT token is issued, which the client uses to authenticate subsequent requests.

4.3 Token-Based Authentication (JWT)

JSON Web Tokens (JWTs) are utilised for secure authentication and user sessions. JWTs are signed and issued by the server upon successful login. They contain user-specific information and are used to verify the user's identity on subsequent API requests.

5. Messaging Features

5.1 Real-Time Chat

The core feature of the Messaging App is real-time chat. Users can send and receive messages instantly. Messages are delivered to the recipient's chat window as soon as they are sent. Real-time chat is facilitated by Socket.io, ensuring low-latency communication.

5.2 One-to-One Chat

Users can engage in private one-to-one conversations by selecting a specific user from their contacts list. One-to-one chat conversations are secure and offer real-time message updates and typing indicators.

5.3 Group Chat Functionality

Group chat functionality allows users to create and participate in group conversations. Users can invite others to join a group, and group chat administrators have the ability to manage members, adding or removing users as needed.

5.4 Message History and Storage

All chat messages are stored in MongoDB, ensuring that users can access their message history even after logging out or switching devices. This feature enhances the user experience by providing a complete chat record.

5.5 Typing Indicators

The application includes typing indicators to inform users when their chat partner is actively composing a message. This real-time feature enhances user engagement and communication.

6. Search and User Interaction

6.1 User Search Functionality

Users can search for other users by username or display name. This feature simplifies the process of finding and connecting with friends and acquaintances within the app.

6.2 User Profile View

Users can access and view the profiles of other users, providing a glimpse into their interests, bio, and contact information. Profile view functionality enhances user interactions and connections.

7. Notifications

7.1 Real-time Notifications for Incoming Messages

To keep users informed, the Messaging App employs real-time notifications. Users receive immediate alerts when new messages arrive.

8. Group Chat Management

8.1 Adding/Removing Users from Groups

Administrators of group chats have the authority to add or remove users from the group. This functionality enables the group to maintain its membership according to its intended purpose and focus.

9. User Interface Design

9.1 UI Libraries

Chakra UI is used for a responsive and visually appealing design.

9.2 Responsive Design Principles

A key focus of the user interface design is responsiveness. The application is designed to adapt to various screen sizes and devices, offering a consistent and visually appealing experience.

10. Backend Development

10.1 REST API Design

The backend development involves the creation of RESTful APIs that facilitate communication between the client and server. API endpoints and routes are designed to handle various client requests efficiently.

10.2 API Endpoints and Routes

The API endpoints are carefully structured to support user registration, authentication, messaging, and interactions. Well-defined routes ensure that requests are appropriately handled and processed.

10.3 Server Setup (Node.js, Express.js)

Node.js and Express.js provide the server environment, allowing for efficient request handling and server-side logic execution. These technologies offer scalability and robustness.

10.4 Database Schema and MongoDB

MongoDB, a NoSQL database, is used to store user data and messages. A well-structured database schema ensures data integrity and efficient data retrieval.

11. Deployment and Hosting (Optional)

11.1 Cloud Platform

While the Messaging App can be run locally during development, deploying it to a cloud platform offers several advantages in terms of scalability and availability. I have deployed it on render, where it can be tested. https://chitchathub-7zfc.onrender.com/

11.2 Server Deployment

Deployment to the chosen cloud platform involves configuring server instances, environment variables, and necessary dependencies, which are as follows:

```
PORT= {YOUR CHOSEN PORT NUMBER}
MONGO_URI={YOUR MONGODB DATABASE URI}
JWT_SECRET={YOUR JSON WEB TOKEN SECRET}
NODE_ENV=production
```

11.3 Database Deployment

If MongoDB is used, cloud-based database services like MongoDB Atlas can be leveraged for database deployment. This ensures data availability and scalability, even under high-traffic conditions.

12. Documentation

12.2 Setup and Installation Guide

For developers and users, a clear setup and installation guide is provided. It includes step-by-step instructions for setting up the development environment, installing dependencies, and running the application locally.

```
Clone the project
 git clone https://github.com/Nishthat12/Messaging-App

Go to the project directory
 cd Messaging-App

Install dependencies
 npm install

 cd frontend/
  npm install

Start the server
 npm run start

Start the Client
 //open now terminal
  cd frontend
  npm start

Create a .env file with the following details:
```

```
PORT= {YOUR CHOSEN PORT NUMBER}
MONGO_URI={YOUR MONGODB DATABASE URI}
JWT_SECRET={YOUR JSON WEB TOKEN SECRET}
NODE_ENV=production //for deployment
```

13. Security Considerations

13.1 Authentication and Authorization

User authentication is fortified using JSON Web Tokens (JWTs) and strong password hashing. Authorisation mechanisms ensure that users can only access resources and perform actions they are permitted to.

13.2 User Data Protection

Stringent data protection measures are implemented to prevent data breaches and unauthorised access. Sensitive user information is stored securely in the database, and access controls are enforced.