Software Design Document

Project Title
Messaging App using the MERN Stack

Authors
Gopal Bansal (2021csb1089@iitrpr.ac.in)

GitHub Repository
mathnebula186f/Messaging-App

Table of Contents

Functional Description
The Messaging App is a web-based application developed using the MERN (MongoDB, Express, React, Node.js) stack. It provides real-time messaging features, including personal chat, group chat, and video calls. The app also includes user authentication, ensuring secure access. Key features include:

- *Personal Chat Feature*: Users can have one-on-one conversations with other online users in real-time.
- *Group Chat Feature*: Users can engage in group discussions with multiple online users, create and join groups, and chat within these groups.
- *Video Call Feature*: Users can create video call rooms and invite online users for real-time video conversations.
- *Login and Register*: A fully authenticated system for user registration and login to ensure secure access to the app's features.
- *Interactive UI*: The app boasts an intuitive and interactive user interface to provide a seamless messaging experience.

User Interface
The user interface of the Messaging App is designed to be interactive and user-friendly. It includes the following elements:

- *Login/Register Page*: A clean and straightforward page where users can log in or register for the app.

- *Personal Chat Interface*: A chat window where users can have one-on-one conversations with online users.
- *Group Chat Interface*: Users can create, join, and participate in group chat rooms. The interface provides a list of active groups and chat messages.
- *Video Call Interface*: A user-friendly video call interface where users can create rooms and initiate video calls with online contacts.
- *Profile Page*: Users can view and edit their profiles, including profile pictures, usernames, and other account details.

Goals and Milestones

Goals

- Create a feature-rich messaging app using the MERN stack.
- Enable personal and group messaging, as well as video calling.
- Implement secure user authentication.
- Develop an intuitive and interactive user interface.

Milestones

- Set up the project and establish the basic MERN stack structure.
- Implement user authentication and authorization.
- Create personal chat functionality.
- Develop group chat features, including the ability to create and join groups.
- Add video call capabilities using WebRTC.
- Design and implement an interactive user interface.
- Test the application for functionality and security.
- Deploy the app to a production environment.

Prioritization

Prioritization of features and tasks for the Messaging App:

- Critical Features
    - User Authentication
    - Real-time Messaging
    - Video Calling
    - Group Messaging

Timeline

- **Phase 1**
    - Project Setup
    - Basic MERN Stack
    - User Authentication
- **Phase 2**

    - Personal Chat
    - User Profile
    - Group Management
- **Phase 3**
    - Group Chat
    - Video Calling
    - Interactive UI
- **Phase 4**
    - Testing and Debugging

Libraries Used

Frontend Dependencies:

1. `autoprefixer`: Used to automatically add vendor prefixes to CSS styles for better browser compatibility.
2. `axios`: Used for making HTTP requests from the frontend to interact with the server and external APIs.
3. `lodash`: A utility library used for simplifying JavaScript operations and handling data transformations.
4. `postcss`: A tool for processing CSS, often used in conjunction with `autoprefixer` for styling.
5. `prop-types`: Used for documenting and validating the types of properties that components receive in React.
6. `react` and `react-dom`: The core libraries for building user interfaces in React.
7. `react-lottie`: Used for integrating Lottie animations in React applications.
8. `react-player`: Enables the integration of video and audio players in React applications.
9. `react-router-dom`: A library for routing and navigation in React applications.
10. `socket.io-client`: Enables real-time communication with the server using WebSockets.
11. `tailwindcss`: A utility-first CSS framework for styling the application.

Backend Dependencies:

1. `bcryptjs`: Used for hashing and salting passwords to enhance user data security.
2. `cookie` and `cookie-parser`: Utilized for handling cookies in web applications, which can be useful for user authentication and session management.
3. `cors`: Middleware for handling Cross-Origin Resource Sharing, enabling cross-origin HTTP requests.
4. `dotenv`: Used to load environment variables from a `.env` file into the application's environment.
5. `express`: A web application framework for building server-side applications and APIs.
6. `jsonwebtoken`: Used for creating and verifying JSON Web Tokens (JWTs) for user authentication.
7. `mongoose`: An Object Data Modeling (ODM) library for MongoDB, facilitating interaction with the database.
8. `nodemon`: Monitors for changes in the server code and automatically restarts the server during development.

9. `socket.io`: Facilitates real-time communication between the server and clients using WebSockets.
10. `ws`: A library for implementing WebSocket servers and clients for real-time communication.

How to Run-

To run the app locally, follow these steps:

- Clone this repository to your local machine:

```
git clone https://github.com/your-username/messaging-app.git
```

- Navigate to the project directory:

```
cd messaging-app
```

- Install dependencies for both the server and client:

```
cd api
npm install
cd ../client
npm install
cd ../server2
npm install
```

- Start the Servers

```
cd ../api
node index.js
cd ../server2
node index.js
```

- Start the client

```
cd ../client
npm run dev
```

- Open your web browser and visit http://localhost:3000 to use the Messaging App.