Video Conferencing App

Introduction:

Introducing our revolutionary video conference app! Designed to redefine remote collaboration, our webapp offers an intuitive and innovative platform for seamless communication.

Experience a new level of connectivity with our group video conference feature. Whether you're working with colleagues, connecting with friends, or hosting virtual events, our webapp ensures a smooth and immersive experience that brings everyone together.

Break down barriers and enhance collaboration with real-time screen sharing. Whether you're delivering dynamic presentations, collaborating on projects, or providing hands-on demonstrations, our screen sharing feature allows for seamless interaction and understanding.

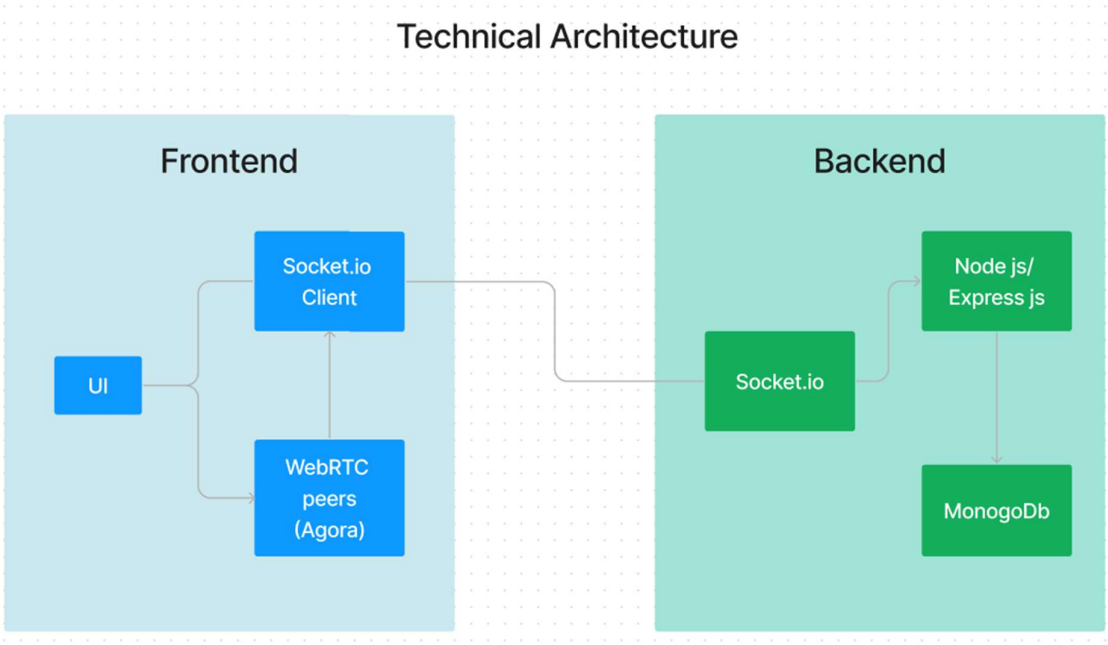
Never miss a moment with our convenient meet recording functionality. Capture important discussions, presentations, or brainstorming sessions for later playback and sharing with absent participants. Your valuable insights are preserved, ensuring nothing gets lost in translation.

Stay engaged and connected with in-meet chat. Instantly communicate, share links, or ask questions alongside the live audio and video streams, making collaboration efficient and effective.

Effortlessly plan and organize your video conferences with our meet scheduling feature. Set up meetings, invite participants, and send automated reminders, ensuring everyone is prepared and punctual for productive sessions.

Your privacy and security are paramount. Our app employs robust encryption to safeguard your data, ensuring all communications remain confidential and protected from unauthorized access.

Step into a new era of remote collaboration and communication. Discover the unmatched convenience of seamless video conferencing, innovative screen sharing, meet recording, in-meet chat, and meet scheduling features – all within our game-changing video conference app. Connect, collaborate, and achieve more together!



The technical architecture of our video conference app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the socket.io-client and WebRTC API.

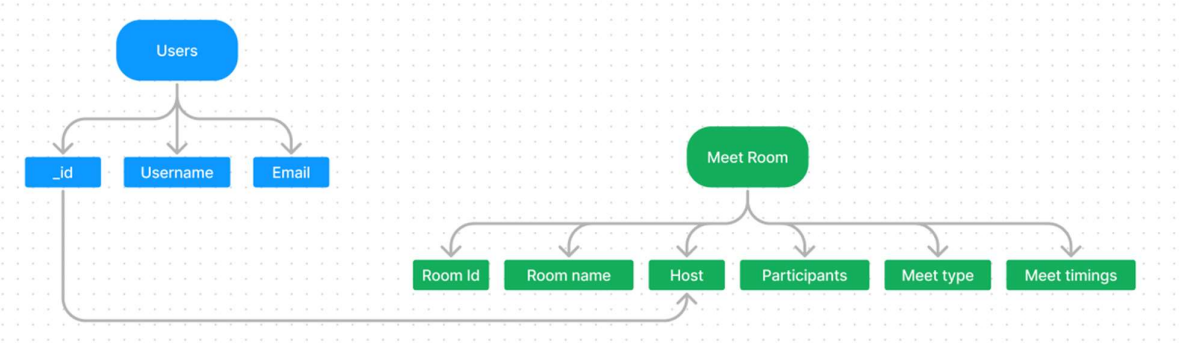
The frontend utilizes the socket.io-client to establish real-time bidirectional communication with the backend server. This enables seamless and instant exchange of audio, video, and chat data between participants during the video conference. Additionally, the WebRTC API plays a crucial role in facilitating peer-to-peer communication, enabling direct audio and video streaming between users without the need for intermediate servers.

On the backend side, we employ socket.io and Express.js frameworks to handle the server-side logic and communication. Socket.io allows for real-time event-based communication between the server and clients, enabling efficient synchronization of video conference data and seamless updates across all participants.

For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data, including user profiles, meeting schedules. It ensures reliable and quick access to the necessary information during video conferences.

Together, the frontend and backend components, along with socket.io, Express.js, WebRTC API, and MongoDB, form a comprehensive technical architecture for our video conference app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive video conferencing experience for all users.

ER Diagram:



In our video conference app, the ER diagram showcases entities such as users, meetings and scheduling. It illustrates how these entities relate to each other, helping us understand the underlying database structure and the flow of information within the app.

Furthermore, the ER diagram represents the relationship between meetings and scheduling, enabling users to plan and organize video conferences efficiently. This aspect highlights how meeting details and scheduling information interact within the system.

Key features:

- ✓ **Group Video Conference**: Connect and collaborate with multiple participants simultaneously in real-time. Experience a seamless video conferencing experience that brings people together regardless of their location.

- ✓ **Screen Sharing:** Share your screen with participants during a video conference, enabling dynamic presentations, collaborative work, and real-time demonstrations. Foster seamless interaction and understanding among attendees.

- ✓ **Meet Recording**: Capture and save important moments of video conferences, including audio and video, for future reference or sharing with absent participants. Never miss out on crucial discussions or valuable presentations.

- ✓ **In-Meet Chat**: Engage in instant messaging within the video conference, allowing participants to ask questions, share links, and provide feedback alongside the audio and video streams. Enhance collaboration and communication during meetings.

- ✓ **Meet Scheduling**: Seamlessly plan and organize your video conferences with our meet scheduling feature. Set up meetings, specify dates and times, and effortlessly manage your schedule. Ensure everyone is aware of the meeting details, promoting effective time management and coordination among participants.

- ✓ **Privacy and Security**: We prioritize the protection of your data and ensure secure communication. Our app employs robust encryption protocols to safeguard your information and ensures that all video conferences and personal details remain confidential and protected.

These key features collectively enhance your video conferencing experience, providing a comprehensive and secure platform for seamless communication, collaboration, and productivity.

PRE-REQUISITES:

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js, Socket.io, Agora RTC, and Agora RTM:

- ✓ **Node.js and npm**:

  Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

  Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

  - Download: https://nodejs.org/en/download/
  - Installation instructions: https://nodejs.org/en/download/package-manager/

- ✓ **Express.js**:

  Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

  Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

  Installation: Open your command prompt or terminal and run the following command:

  ```
  npm install express
  ```

- ✓ **MongoDB**:

  MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

- Download: https://www.mongodb.com/try/download/community
- Installation instructions: https://docs.mongodb.com/manual/installation/

✓ **React.js**:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

Follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

✓ **Socket.io**:

Socket.io is a real-time bidirectional communication library that enables seamless communication between the server and clients. It allows for real-time data exchange, event-based messaging, and facilitates the development of real-time applications such as chat, collaboration, and gaming platforms.

Install Socket.io, a real-time bidirectional communication library for web applications.

Installation:

- Open your command prompt or terminal of server and run the following command: **`npm install socket.io`**
- Open your command prompt or terminal of client and run the following command: **`npm install socket.io-client`**

✓ **Agora RTC**:

Agora RTC (Real-Time Communication): Agora RTC provides a platform for real-time audio and video communication. It offers a range of features like video conferencing, live streaming, and interactive broadcasting, enabling developers to create immersive and interactive communication experiences.

- Sign up for an Agora developer account to access their RTC platform.
- Integration guide and documentation: https://docs.agora.io/en/

✓ **Agora RTM**:

Agora RTM enables real-time messaging and data synchronization between users. It provides reliable and low-latency messaging capabilities, allowing developers to build chat applications, collaborative tools, and real-time notification systems.

- Sign up for an Agora developer account to access their RTM platform.
- Integration guide and documentation: https://docs.agora.io/en/Real-time-Messaging/index.html

✓ **HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

✓ **Database Connectivity**: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database with Node JS go through the below provided link:

• https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

✓ **Front-end Framework**: Utilize Angular to build the user-facing part of the application, including products listings, booking forms, and user interfaces for the admin dashboard.

✓ **Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

• Git: Download and installation instructions can be found at: https://git-scm.com/downloads

✓ **Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from https://code.visualstudio.com/download
• Sublime Text: Download from https://www.sublimetext.com/download
• WebStorm: Download from https://www.jetbrains.com/webstorm/download

To run the existing Video Conference App project downloaded from GitHub:

Follow below steps:

- ✓ **Clone the Repository:**
  - Open your terminal or command prompt.
  - Navigate to the directory where you want to store the e-commerce app.
  - Execute the following command to clone the repository:

    git clone https://github.com/harsha-vardhan-reddy-07/smart-meet.git
- ✓ **Install Dependencies:**
  - Navigate into the cloned repository directory:

    ```
    cd smart-meet
    ```
  - Install the required dependencies by running the following commands:

    ```
    cd client
    npm install
    cd ../server
    npm install
    ```
- ✓ **Start the Development Server**:
  - To start the development server, execute the following command:

    ```
    npm start
    ```
  - The video conference app will be accessible at http://localhost:3000
- ✓ **Access the App:**
  - Open your web browser and navigate to http://localhost:3000.
  - You should see the video conference app's homepage, indicating that the installation and setup were successful.

You have successfully installed and set up the e-commerce app on your local machine. You can now proceed with further customization, development, and testing as needed.

Roles & Responsibilities:

- ✓ **Host**:

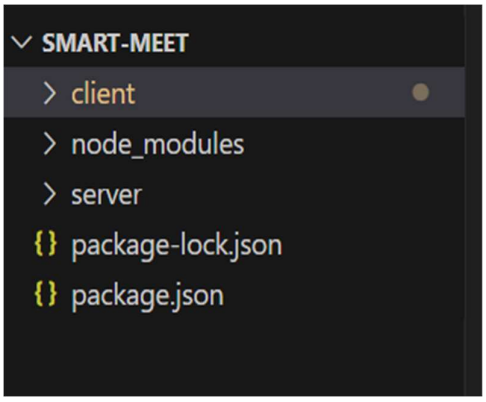
  - Initiate and manage the video conferences.
  - Schedule and send invitations for upcoming meetings.
  - Facilitate smooth communication and collaboration among participants.
  - Ensure the overall organization and flow of the video conference.

- ✓ **Participant**:

  - Join video conferences hosted by the host.
  - Engage in active communication and contribute to discussions.
  - Share screens, if necessary, to present content or collaborate on projects.
  - Utilize in-meet chat to ask questions, share information, or provide feedback.
  - Respect meeting etiquette and follow guidelines set by the host.
  - Actively participate and contribute to a productive and collaborative meeting environment.

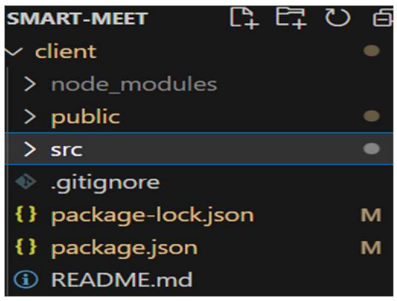
Project structure:

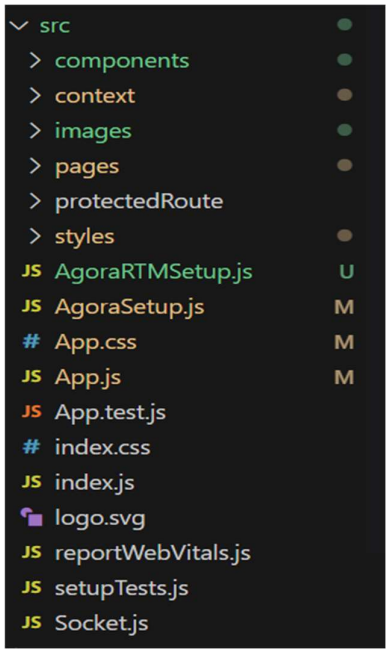
- Inside the smart-meet (video conference app) directory, we have the following folders

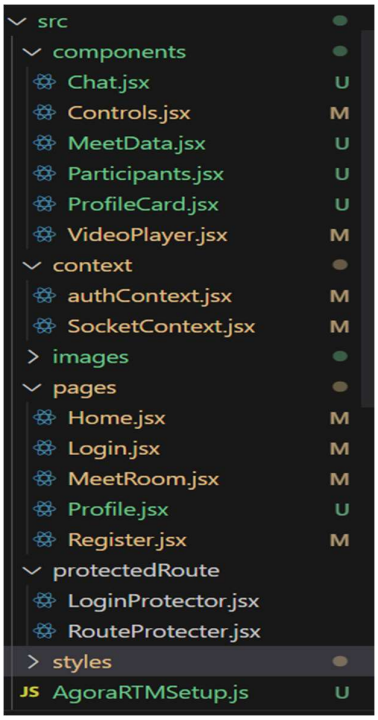


- **Client directory:**

  The below directory structure represents the directories and files in the client folder (front end) where, react js is used along with Api's such as socket.io and agora.

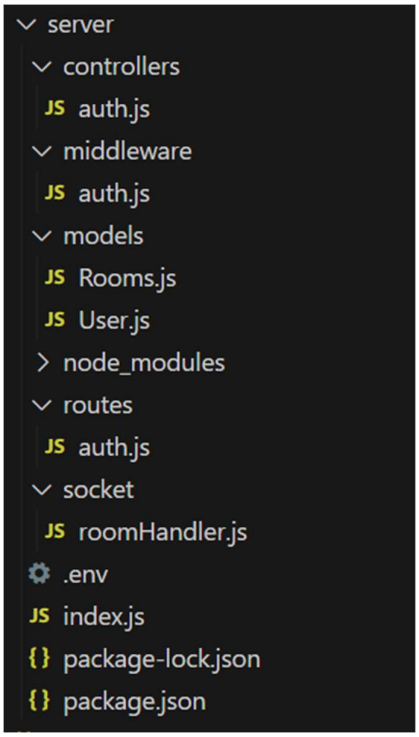

SMART-MEET
client
node_modules
public
src
.gitignore
package-lock.json M
package.json M
README.md


src
components
context
images
pages
protectedRoute
styles
AgoraRTMSetup.js U
AgoraSetup.js M
App.css M
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
Socket.js


src
components
Chat.jsx U
Controls.jsx M
MeetData.jsx U
Participants.jsx U
ProfileCard.jsx U
VideoPlayer.jsx M
context
authContext.jsx M
SocketContext.jsx M
images
pages
Home.jsx M
Login.jsx M
MeetRoom.jsx M
Profile.jsx U
Register.jsx M
protectedRoute
LoginProtector.jsx
RouteProtecter.jsx
styles
AgoraRTMSetup.js U

- **Server directory:**

  The below directory structure represents the directories and files in the server folder (back end) where, node js, express js and mongodb are used along with socket.io Api.



Project Flow:

Project demo:

Before starting to work on this project, let's see the demo.

Demo link: https://drive.google.com/file/d/1cNi-uFaIo0DUnYl8WzB4Q4PGiECvx9X4/view?usp=sharing

Use the code in: https://github.com/harsha-vardhan-reddy-07/smart-meet

or follow the videos below for better understanding.

Milestone 1: Project setup and configuration.

- **Folder setup**:

    1. Create frontend and backend folders
        - client
        - server

- **Installation of required tools**:

    1. Open the frontend folder to install necessary tools
       For frontend, we use:
        - React Js
        - Socket.io-client
        - Bootstrap
        - Material UI
        - Axios
        - recordrtc
        - downloadjs
        - agora-rtc-react
        - agora-rtc-sdk-ng

    2. Open the backend folder to install necessary tools

       For backend, we use:

        - bcrypt
        - body-parser
        - cors
        - dotenv
        - express
        - http
        - jsonwebtoken
        - mongoose
        - socket.io

- For further reference, follow our tutorials
    - Setup & installation: https://drive.google.com/file/d/1Ygx1xT-1CbpkpfwI6pdtTO8q9rb0T6Aa/view?usp=drive_link

Milestone 2: Backend Development

- **Setup express server**

    1. Create index.js file in the server (backend folder).
    2. Create a .env file and define port number to access it globally.
    3. Configure the server by adding cors, body-parser.

- **Create socket.io connection**

    1. Import socket.io in the index.js file.
    2. Create server using the socket.io.
    3. Start the server.

- **Configure MongoDB**

    1. Import mongoose.
    2. Add Database URL to the .env file.
    3. Connect the database to the server.
    4. Create a 'models' folder in the server to store all the DB models.

- **Add authentication**

    1. Create the "User" model for the MongoDB.
    2. Create auth controller file to control the authentication actions.
    3. Import "bcrypt" – used to hash(encode) the password to make it secure.
    4. Import "JWT" to create authentication tokens.
    5. Define registration & login activities.
    6. Using Axios library, make request from the frontend.
    7. Configure frontend & backend for authentication and store authenticated data in Context API in frontend.

- **Create rooms**

    1. Create "Rooms" model to store data of rooms (meet rooms) in Database.
    2. Configure socket.io-client to make client-server communication simpler.
    3. Send message to server if client requested to create a room.
    4. Create a new room and add user to it.
    5. Allow other users to join the room.
    6. Manage user leaving the room.
    7. Use the calendar input, if the user wants to schedule meet for later.

- For further assistance, use our tutorial videos:
    1. Authentication: - Authentication tutorial –

2. Room creation: - create rooms tutorial -

Milestone 3: Web application development

- **Configure API**

    1. Choose an API for media streaming. (You are free to use any WebRTC based API. But in tutorial, we use Agora API).
    2. Create a setup file in client folder (frontend).
    3. Create a new project and copy the token & app id in the API.

- **Access media streams**

    1. Access the camera and microphone.
    2. Capture the media streams and access them.
    3. Also access the display video for screen sharing functionality.

- **Add meet (conference) in room**

    1. Add the users in room to the meet channel.
    2. Publish the media stream of user to others in the meet.
    3. Display the video of all participants in the meet.

- **Add meet (media) controls**

    1. Create a component to add the controlling features.
    2. Add icons to represent the controls.
    3. Create functionalities to mute audio/video and to leave the meet.

- **Add screen sharing functionality**

    1. Access the screen display media when user tap on screen share control option.
    2. Unpublish the camera track of the user and publish the screen track.
    3. Undo the actions when user closed screen share.

- **Add Recording functionality**

    1. Use any Js/react Js libraries to record the meet screen.
    2. Here, we use ”recordrtc” to record the meet and “downloadjs” to download the recorded meet.
    3. Access this functionality using the meet controls.

- **Add in-meet chat facility**

    1. Add the chat feature to seamless communication in the room.
    2. Use any API's like AgoraRTM or simply Socket.io
    3. In case of socket.io, if any user shared the message (text/file), broadcast the message to all other participants in the meet.

- **Add user profile section**

    1. Create a file for profile page.
    2. Display the user details in the page and allow user to edit them.
    3. Use a different page or use the profile page to display the past & upcoming meet details.
    4. Allow user to edit the upcoming meet details.

- For further assistance, use our tutorial videos:
    1. Meet page: - Meeting page tutorial -
    2. Profile section: - Profile page tutorial -

*** Happy coding!! ***