**Full Stack Development with MERN – Project Documentation**

1. **Introduction**

**Project Title**:

Video conferencing App

**Team Members**:

Samarth Gupta (22BCE11394)

Hardik Mishra (22BCE10556)

Yogendra singh rathod (22BCE11383)

Sujal Patel (22BAI10409)

1. **Project Overview**

The Video Conferencing App allows users to communicate in real-time via video and audio.It supports one-on-one calls, group meetings, and virtual classrooms.Users can join meetings via links or scheduled invites.  
The app includes features like screen sharing, text chat, and file sharing.  
Built with a user-friendly interface for easy navigation.Utilizes WebRTC for real-time communication.Ensures secure connections with end-to-end encryption.Supports cross-platform access via web and mobile.  
Authentication and access control are implemented for privacy.  
Ideal for remote work, education, and virtual events.

**Key Aspects of video conferencing App:**

**1.** Real-Time Communication

The app provides seamless real-time audio and video communication between users. It leverages technologies like WebRTC to deliver low-latency, high-quality connections, ensuring a smooth and interactive user experience.

**2**. Multi-User Support

The platform supports multiple participants in a single meeting session. Whether it's a one-on-one conversation or a large group conference, the app maintains stable connections and balanced audio/video streams.

**3.** Screen Sharing

Participants can share their screen with others during meetings. This feature is essential for presentations, demonstrations, remote support, and collaborative work, allowing others to view documents, slides, or applications in real-time.

**4.** Text Chat

In addition to voice and video, the app includes an integrated chat feature that allows users to send messages during meetings. This is useful for sharing quick notes, links, or asking questions without interrupting the speaker.

**5.** File Sharing

Users can share files such as documents, images, and PDFs directly within the meeting interface. This facilitates efficient collaboration, especially in professional and educational environments.

**6.** Meeting Scheduling

The app allows users to schedule meetings in advance, send calendar invites, and generate unique meeting links. Notifications and reminders help ensure participants are aware and ready to join on time.

**7. User Authentication**

To maintain security, the app includes user registration and login functionalities. Only authenticated users can host or join meetings, reducing the risk of unauthorized access and enhancing control.

**8.** End-to-End Encryption

All communication within the app is protected by end-to-end encryption. This ensures that video, audio, messages, and files are secure and can only be accessed by authorized participants.

**9.** Cross-Platform Compatibility

The app is designed to work across multiple platforms including web browsers, Android, iOS, and desktop environments. This ensures accessibility for users regardless of their device or operating system.

**10.** User-Friendly Interface

* The interface is designed with simplicity and ease of use in mind. Clear navigation, intuitive controls, and responsive design make it accessible to users with varying levels of technical experience.

**Purpose:**  
The purpose of the Video Conferencing App is to enable real-time communication between users through video, audio, and chat, regardless of their location. It is designed to support virtual meetings, online classes, remote work, and team collaboration. The app aims to reduce the need for physical meetings, save time and resources, and promote flexibility. It also enhances user interaction through features like screen sharing, file sharing, and secure login. Overall, the app provides a convenient and efficient platform for virtual communication in various professional and personal scenarios.

**Features:**  
The Video Conferencing App offers a rich set of features that make virtual communication simple, smooth, and secure:

* + **High-Quality Video & Audio Calls:** Users can connect in real-time with crystal-clear audio and high-definition video.
  + **Meeting Room Creation & Joining:** Users can easily create or join meetings via unique room links or meeting IDs.
  + **Screen Sharing:** Share your entire screen or specific applications for effective presentations and collaborations.
  + **Text Chat:** Integrated chat allows participants to send messages, links, and notes during meetings.
  + **Meeting Scheduler:** Schedule meetings in advance with automatic reminders and calendar integrations.
  + **Admin Dashboard:** Admins can manage users, monitor active sessions, and configure system settings.
  + **Recording Support:** Option to record meetings for future reference or sharing with absent participants.

The app features a clean and responsive UI/UX, built using Material-UI (MUI), providing a modern, accessible, and consistent experience across all platforms and devices.

**3. Architecture**

**Frontend:**

The frontend of the Video Conferencing App is developed using **React 18**, leveraging its component-based architecture to ensure reusability and scalability. This setup allows for easy integration of new features such as chat, recording, or meeting scheduling without disrupting existing functionality.

**Material-UI (MUI)** is used for UI components, offering a consistent, modern design that is responsive across all screen sizes and devices. This ensures a professional and accessible user experience.

**Redux Toolkit** handles global state management, helping manage user sessions, meeting data, and chat history efficiently. Redux DevTools simplifies debugging by tracking state changes in real time.

**React Router** allows smooth and dynamic navigation between different views like home, meetings, settings, and chat, ensuring a single-page application experience.

For backend communication, **Axios** is used to perform asynchronous HTTP requests (GET, POST, PUT, DELETE), enabling real-time interaction and data synchronization.

**Key Frontend Technologies:**

* **React 18:** Modular and scalable UI development.
* **Material-UI (MUI):** Responsive design and pre-built UI components.
* **Redux Toolkit:** Predictable and centralized state management.
* **React Router:** Client-side routing and navigation.
* **Axios:** Communication with backend APIs.

**Backend:**

The backend is built using **Node.js** and **Express.js**, providing a fast and efficient runtime environment for handling concurrent user connections. The RESTful API manages user authentication, meeting creation, live session management, and chat handling.

Authentication and authorization are managed using **JWT (JSON Web Tokens)**. Upon successful login, users receive a token used in all subsequent API requests for secure access to protected routes like starting a meeting or sending messages.

Middleware ensures that protected endpoints are only accessible to authenticated users. This includes endpoints for creating meeting rooms, managing participants, and accessing chat logs.

**WebSocket (Socket.IO)** is integrated for real-time communication, enabling low-latency video/audio data transfer, instant messaging, and real-time event handling such as user join/leave notifications.

**Key Backend Technologies:**

* **Node.js:** Event-driven server-side environment.
* **Express.js:** API routing and request handling.
* **JWT:** Token-based user authentication and secure access.
* **Middleware:** Route protection and session validation.
* **Socket.IO:** Real-time bi-directional communication.

**Database:**

The app uses **MongoDB** as its primary database, chosen for its flexibility and scalability in managing dynamic data structures. **Mongoose** is used to define schemas and interact with MongoDB in a structured, schema-based manner.

The database manages core entities such as User, Meeting, and Chat.

1. **User Schema**:
   * Stores user details like name, email, password (hashed using **bcrypt**), role (host/participant), and meeting history.
   * Includes token management for authentication and password reset features.
2. Meeting Schema:

* Stores meeting-specific data such as host ID, participant list, meeting link, schedule time, and status (active/inactive).
* Supports scheduled and instant meetings, with metadata for duration and type.

1. Chat Schema:

* Stores messages exchanged during meetings, including sender ID, timestamp, meeting ID, and message content.
* Ensures messages are retained for review and archival if needed.

**Key Database Technologies:**

* **MongoDB:** NoSQL document storage for flexible data modeling.
* **Mongoose:** Schema management and MongoDB integration.
* **bcrypt:** Secure password hashing.
* **WebSocket/Socket.IO Storage Layer (if applicable):** For storing real-time event logs or transient data.

**Overall System Flow:**

1. **User Interactions:** When a user accesses the app (e.g., signs in, starts/join a meeting, sends a chat message), the frontend communicates with the backend via API calls or WebSocket connections to initiate the requested action.
2. **Backend Processing:** The backend validates user credentials using JWT, checks permissions, and handles operations such as creating or joining a meeting, fetching chat history, or broadcasting real-time updates.
3. **Real-Time Communication:** For live meetings and instant messaging, the backend uses **Socket.IO** to establish a WebSocket connection with the frontend, enabling real-time video, audio, and message transmission between participants.
4. **Database Operations:** If the action involves persistent data (e.g., storing user profiles, meeting history, chat logs), the backend uses **Mongoose** to interact with **MongoDB**, ensuring the data is stored or retrieved as needed.
5. **Response Handling:** The backend sends the response or real-time update back to the frontend. This may involve updating the UI, showing alerts, refreshing the chat, or initiating the video call interface.

This architecture ensures that the Video Conferencing App functions in real-time, remains secure, and scales efficiently as user demands and features expand.

1. **Setup Instructions**

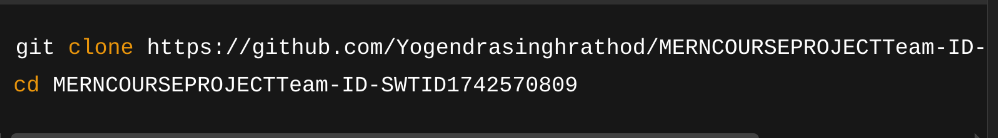
**Prerequisites:**

Before getting started, ensure you have the following tools and technologies installed on your system:

* **Node.js**: A JavaScript runtime environment used for backend development (version 14.x or later).
* **MongoDB**: A NoSQL database for storing application data (local or cloud-based like MongoDB Atlas).
* **npm**: A package manager used to manage project dependencies.
* **Git**: For version control and cloning the project repository.
* **Web Browser**: A modern web browser (e.g., Chrome, Firefox) to access the frontend.
* **Cloud Storage (optional)**: If you plan to use cloud storage for file uploads, services like AWS S3 or Cloudinary may be required for media storage.

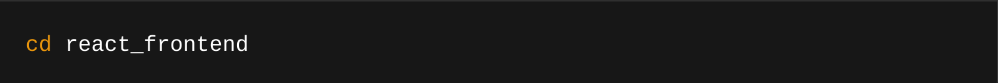
**Installation:**

**1.Clone the Repository**: First, clone the project repository to your local machine:



**2.Frontend Setup:**

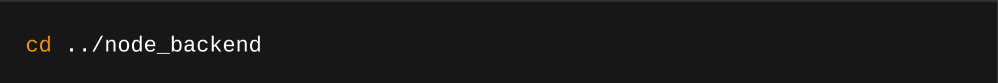
* Navigate to the client directory and install the dependencies:



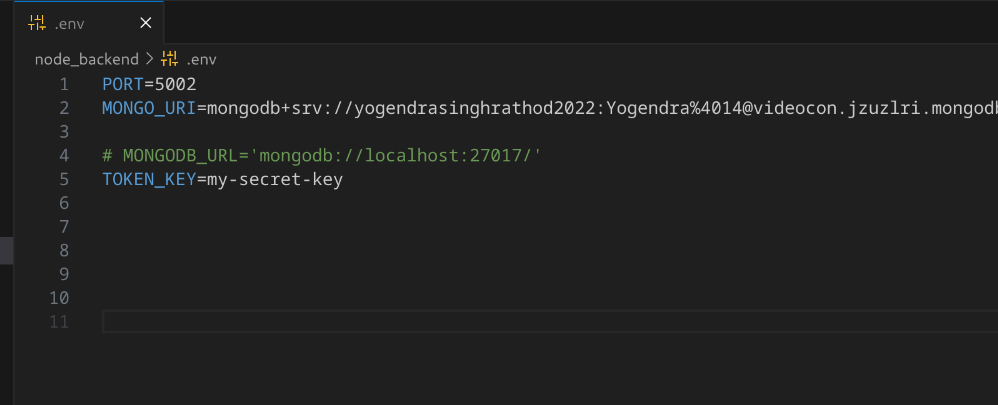
* Create a .env file in the client directory to store environment variables (e.g., API URLs).

**3.Backend Setup:**

* Navigate to the server directory and install the dependencies:



* Create a .env file in the server directory to store sensitive variables like database URLs, JWT secret, etc.

1. **MongoDB Setup:**
2. 

* If you're using a local MongoDB instance, make sure MongoDB is running on your machine.
* For production, use MongoDB Atlas to set up a cloud database.

**5.Running the Application:**

To run the application locally, you’ll need to start both the frontend and backend servers:

* **Frontend**: Navigate to the client directory and run:



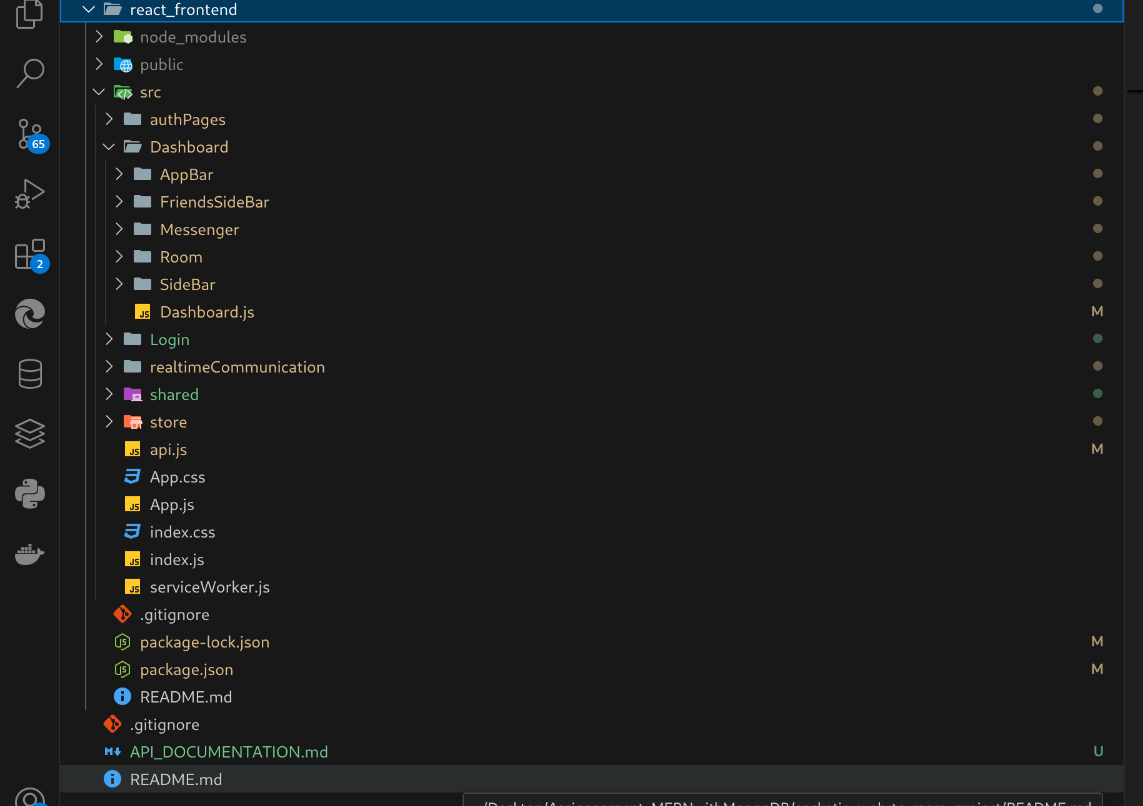
* **Backend**: Navigate to the server directory and run:

The app should now be running at http://localhost:3000 (or your specified port).

1. **Folder Structure**

**Client:**

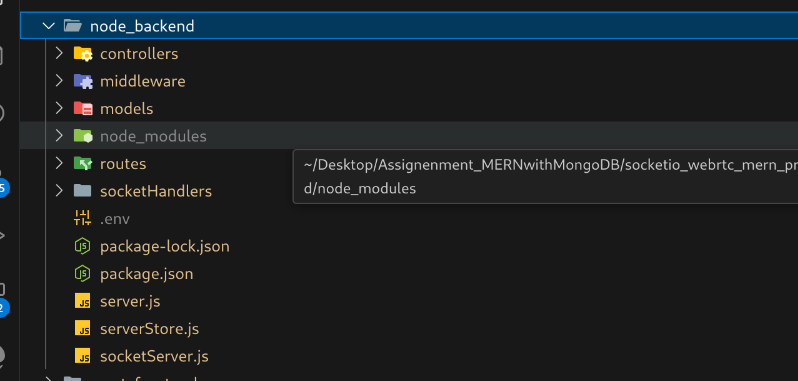
The frontend of the video conferencing app is located in the client directory. Its structure is organized as follows:

* **src/components**: Contains reusable React components used across the app (e.g., Header, VideoPlayer, ChatBox, MeetingControls).
* **src/pages**: Contains different views or pages of the app (e.g., HomePage, JoinMeetingPage, UserProfile, MeetingRoom).
* **src/redux**: Contains Redux slices and store configuration for managing global state, such as user authentication, meeting data, and chat messages.
* **src/api**: Contains files for handling API requests (e.g., axiosInstance for making HTTP requests to the backend).
* **src/assets**: Stores static assets such as images, icons, fonts, and stylesheets.
* **src/utils**: Utility functions or constants used across the app (e.g., WebSocket connection handling, user validation functions).
* 

**Server:**

The backend of the video conferencing app is located in the server directory. It is structured as follows:

* **models**: Contains Mongoose models for core data entities (e.g., User, Meeting, Message, Participant).
* **routes**: Contains API route handlers for the app (e.g., authRoutes, meetingRoutes, messageRoutes, userRoutes).
* **controllers**: Contains the logic for handling API requests, such as user registration, meeting creation, or message sending.
* **middleware**: Contains middleware functions, including JWT authentication for securing routes, user role validation, etc.
* **config**: Configuration files for setting up the database connection, WebSocket configurations, environment variables, and external services.
* **utils**: Utility functions used for tasks like password encryption, token generation, real-time notifications, and meeting validation.

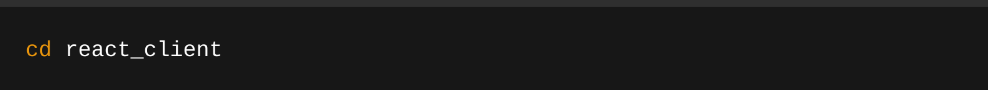


1. **Running the Application**

To run **VideoCon** locally, follow the steps below for both the **frontend** and **backend** servers:

**1. Frontend Setup**

1. **Navigate to the client directory**:



1. **Install Dependencies**:



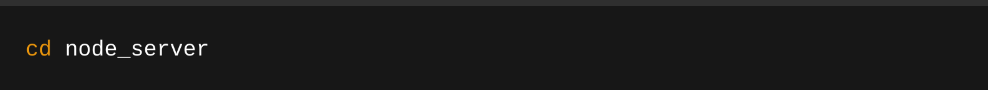
1. **Start the Frontend Development Server**:



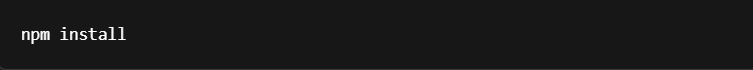
The frontend will be accessible at **[http://localhost:3000](http://localhost:3000" \t "_new)**. The server supports **hot-reloading**, so changes are reflected instantly.

**2. Backend Setup**

1. **Navigate to the server directory**:



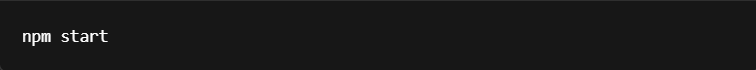
1. **Install Dependencies**:



1. **Set up Environment Variables**:

Create a .env file and add necessary configuration (e.g., MongoDB URI, JWT secret).

1. **Start the Backend Server**:



The backend will run at  **“http://localhost:5002”**

**3. Testing the Application**

* **Frontend**: Visit “**http://localhost:3000”** to see the UI. Changes will reload automatically.
* **Backend**: Use tools like **Postman** to test API endpoints. Errors will appear in the terminal or browser console.

**4. Stopping the Application**

* **Frontend**: Press Ctrl + C in the terminal.
* **Backend**: Press Ctrl + C in the backend terminal.

**5. Optional: Running in Production Mode**

For production, build the frontend and serve it alongside the backend. You can deploy to platforms like **Heroku** or **AWS**.

**7. API Documentation**

VideoCon backend exposes a RESTful API built with **Node.js** and **Express.js**, allowing seamless communication with the frontend. All API endpoints return JSON responses and are secured using JWT where required.

**User Routes**

**➤ POST /api/auth/register– Register a New User**

Registers a new user and returns a JWT for authenticated access.

**Request Body:**



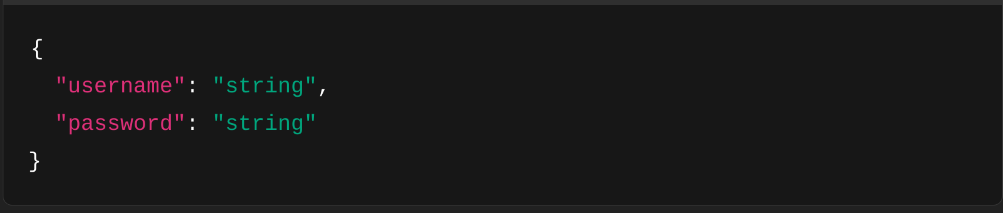
**Response:**



**➤ POST /api/auth/login– Authenticate Existing User**

Validates credentials and returns a JWT if successful.

**Request Body:**



**Response:**



**Room Management**

### **➤**POST /api/rooms/create

Creates a new video room.

**Response:**

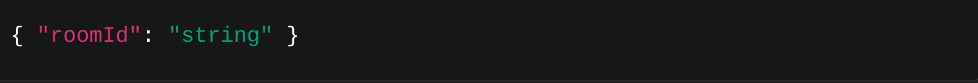


### **➤**POST /api/rooms/join

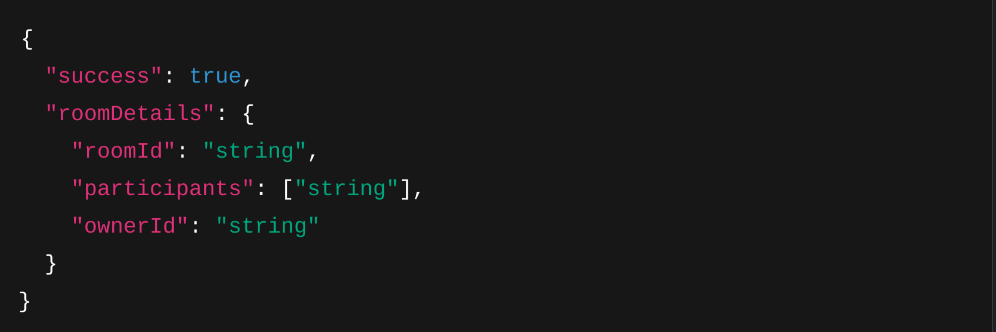
Join an existing room.

Allows a logged-in user (typically a host) to create a new property listing. Requires JWT in the header.

**Request Body:**



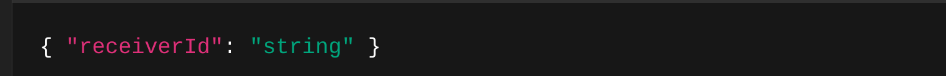
**Response:**



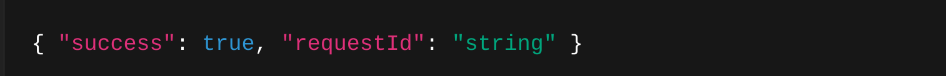
### POST /api/rooms/leave

Leave a room.

Request Body



Response



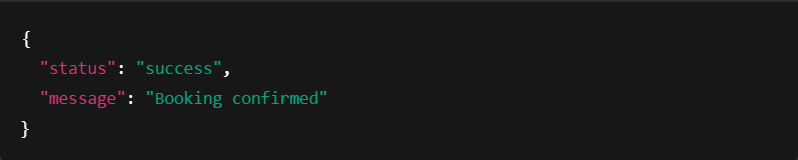
**Friend Management**

### **➤**POST /api/friends/request

Send a friend request.

**Request Body:**

**Response:**



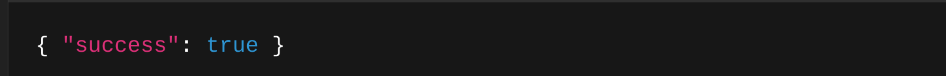
### **➤** POST /api/friends/accept

Accept a friend request.

Request Body



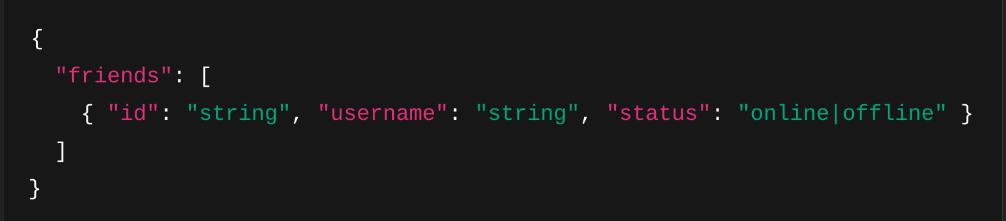
Response



### **➤** GET /api/friends/list

Fetch friend list.

Response



**8. Authentication**

# JWT Authentication System for Video Conferencing App

## Overview

The Video Conferencing App implements JWT (JSON Web Tokens) as its primary authentication and authorization mechanism. This system ensures that only authenticated users can access protected features and routes within the application.

## Authentication Flow

### Token Generation

* During user signup or login, the backend generates a JWT token
* The token contains encrypted user information (ID, role, permissions)
* The token is signed with a secret key to verify authenticity
* The generated token is sent to the client upon successful authentication

### Token Storage

* JWT tokens are stored in the browser's localStorage
* This allows tokens to persist across browser sessions
* Users remain authenticated until the token expires or is manually removed

### Request Authorization

* For protected API requests, the client attaches the JWT token
* Token is included in the HTTP Authorization header as a Bearer token
* Format: Authorization: Bearer [token]
* This pattern secures sensitive operations like:
  + Starting or joining meetings
  + Sending messages
  + Accessing user settings
  + Managing user profiles

### Server-Side Validation

* Backend middleware validates every incoming request to protected routes
* Validation checks:
  + Token presence
  + Token validity (signature verification)
  + Token expiration status
* Invalid requests receive a 401 Unauthorized status response

### Session Management

* JWT tokens have a configurable expiration timeframe
* When a token expires, the application redirects to the login page
* Users must reauthenticate to receive a new valid token
* This prevents unauthorized access if a token is compromised

## Security Benefits

* Fine-grained access control based on user roles and permissions
* Stateless authentication reduces server load and improves scalability
* Protection against common security threats
* No session storage required on the server
* Cross-domain authentication support

**9. User Interface**

The UI/UX of the Video Conferencing App is designed to provide a seamless, intuitive, and engaging experience for users. Built with **React** and **Material-UI (MUI)**, the interface is responsive, fast, and easy to navigate, ensuring users can focus on their meetings without distraction.

Key AI Features :

* **Responsive Design**: The app is fully mobile-compatible, ensuring a consistent experience across desktop, tablet, and mobile devices. MUI's grid system and breakpoints are utilized to adjust the layout dynamically based on screen size.
* **User Login/Signup**: Simple and secure login and signup process, with options for email/password authentication and social logins (e.g., Google, Facebook).
* **Dashboard View**:
  + Users can view upcoming, ongoing, and past meetings.
  + Easy access to account settings, including profile updates and notification preferences.
  + Manage meeting settings, such as scheduling, invites, and participant controls.
* **Meeting Room Interface**:
  + Real-time video streaming with options to toggle audio, video, and screen sharing.
  + Chat functionality for participants to communicate via text during the meeting.
  + Clear buttons for muting, turning off video, and accessing settings.
* **Meeting Scheduling**:
  + Users can schedule a meeting with a simple calendar interface.
  + Options to set reminders and recurring meetings.
  + Email invitations sent to participants with meeting details.
* **Real-Time Notifications**:
  + Notifications for upcoming meetings, new participants joining, and meeting updates.

The UI follows a **minimal and modern design**, ensuring both aesthetics and usability.

**10. Testing**

We prioritize code quality and reliability by implementing thorough testing on both the frontend and backend.

**🔹 Frontend Testing**

* **Tools Used**: Jest, React Testing Library
* **Test Coverage**:
  + Login and registration form behavior
  + Property card rendering
  + Booking flow interaction
* **User Interaction Testing**: Simulate real-world actions such as clicking buttons, entering data, and submitting forms to verify UI responses.

**🔹 Backend Testing**

* **Tools Used**: Mocha, Chai, Supertest
* **API Endpoint Tests**:
  + User authentication (login/signup)
  + CRUD operations for properties
  + Booking requests with both valid and invalid data
* **Validation**: Tests also ensure proper validation messages are returned when required fields are missing or invalid.

**11. Screenshots or Demo**

# Screenshots and Demo Highlights

To better illustrate the functionality and visual design of VideoCon, below are key interface screenshots and demo highlights that showcase the platform in action.

## Home Page

* Clean landing page with featured meeting options, quick join functionality, and prominent call-to-action buttons for sign-up and creating new meetings
* Navigation menu for easy access to all platform features
* User testimonials and key feature highlights

## Meeting Lobby

* Grid view of active and upcoming meetings with thumbnails, titles, host names, and quick join buttons
* Search and filter options for finding specific meetings
* One-click meeting creation button with customizable settings

## Meeting Room Interface

* High-quality video grid displaying all participants
* Control panel with mute/unmute, video toggle, screen sharing, and meeting settings
* Integrated chat panel and participant list sidebar

## User Dashboard

* Profile management with customizable settings
* Meeting history with recordings and analytics
* Personal meeting room management and customization options

## Meeting Setup

* Meeting configuration screen with options for scheduling, permissions, and advanced settings
* Participant invitation system with shareable links and email options
* Meeting template selection for frequent meeting types

## Live Demo

Example: Live Demo

<https://drive.google.com/file/d/1E2-BbpWeGQRYmhEa70nqxpPha5_A-D7Y/view?usp=sharing>

## 12. Known Issues

While the current implementation of the WebRTC-based communication platform is stable for most use cases, several known issues are being actively worked on. These do not break the core functionality but are important for improving user experience and system reliability.

### 1. UI/UX Improvements (Issue 123)

The overall user interface needs refinement to ensure responsiveness and accessibility.

Components such as modals, chat windows, and participant views may render inconsistently across screen sizes.

Enhancing mobile navigation and layout consistency is a priority.

### 2. Webcam Recording Compatibility

Video recordings default to .webm format, which is not supported by all devices or browsers.

Users on certain operating systems face playback issues when trying to view or download recordings.

### 3. Communication Visibility

Real-time communication between two users is sometimes not displayed correctly on both ends.

Message or stream synchronization may lag under network fluctuations or high latency.

### 4. Missing Shareable Room Links

Currently, there is no unique room URL that can be shared to invite participants.

Users must be added as friends first to join a class/room, which limits accessibility and flexibility.

### 5. Insecure Room Access

Friends who are not explicitly permitted can still join rooms if they are in the friend list.

There is a lack of granular access control or whitelist mechanism to prevent unauthorized entry.

**13. Future Enhancements**

# Future Enhancements for VideoCon

To improve the functionality, scalability, and overall user experience of our VideoCon platform, we have identified several key areas for future development. These enhancements are planned for upcoming versions of the application.

## Feature Enhancements

### 1. Advanced Meeting Controls

* Breakout rooms for splitting participants into smaller discussion groups
* Hand raising and participant queuing for organized Q&A sessions
* Advanced host controls for managing participant permissions

### 2. Enhanced Messaging System

* Implementation of persistent chat history using real-time technologies like Socket.io
* Support for rich text formatting, emojis, and file attachments in chat
* Direct messaging between participants during and outside of meetings

### 3. Meeting Analytics & Insights

* Meeting duration, attendance, and participation metrics
* Speaking time distribution across participants
* Engagement analytics to improve meeting effectiveness

### 4. Advanced Recording Features

* Cloud-based recording storage with automatic transcription
* Recording editing tools for post-meeting content creation
* Recording access controls with expiration settings

### 5. Advanced Scheduling Options

* Calendar integration with Google Calendar, Outlook, and iCal
* Recurring meeting setup with customizable parameters
* Time zone management for global team coordination

### 6. Email & Notification System

* Sending meeting invitation emails with joining instructions
* Automated reminders before scheduled meetings
* Follow-up emails with meeting recordings and transcripts

### 7. Admin Dashboard

* A dedicated interface for administrators to manage users, meetings, and system settings
* Dashboard features including analytics, usage monitoring, and content moderation
* User management tools for enterprise deployments

### 8. Virtual Backgrounds & Effects

* Custom background images and blur effects
* Video filters and appearance touch-up features
* Virtual whiteboard and annotation tools for interactive presentations

### 9. Security Enhancements

* End-to-end encryption for all communications
* Advanced authentication options (2FA, SSO)
* Compliance features for regulated industries

### 10. Accessibility Improvements

* Live captioning for better accessibility
* Keyboard navigation enhancements
* Screen reader compatibility improvements

These planned enhancements will significantly improve the VideoCon platform's capabilities, making it more robust, feature-rich, and competitive in the video conferencing market.

**Technical Enhancements**

**1. JWT Token Refresh**

* Implementation of refresh tokens to maintain sessions without constant re-authentication.

**2. Role-Based Access Control (RBAC)**

* Differentiating features and permissions for Admins, Hosts, and Guests.
* Restricting access to certain routes and features based on roles.

**3. Performance Optimization**

* Adding pagination to listing pages to reduce load time.
* Lazy loading images and implementing code splitting for faster rendering.

**4. CI/CD Pipeline**

* Setting up Continuous Integration and Deployment using GitHub Actions or similar tools.
* Streamlining updates through automated testing and deployment.

**Conclusion**

**VideoCon** represents a feature-rich, real-time communication platform built using the MERN stack, WebRTC, and Socket.IO. Designed to enable seamless video calling, room-based collaboration, and friend-based connectivity, the project demonstrates practical implementation of peer-to-peer networking, secure authentication, and dynamic frontend/backend integration.

Through developing VideoCon, we gained in-depth experience with real-time event handling, media stream management, and scalable architectural design. The project highlighted the complexities of synchronizing live communication across devices, ensuring secure access, and delivering a responsive UI/UX.

While the application is already functional and reflects the potential of modern web technologies in building communication tools, there remain opportunities for refinement—such as improved media compatibility, shareable session links, and tighter room access controls. These areas present exciting directions for future updates.

Overall, VideoCon lays a strong foundation for an extensible communication platform, with the potential to evolve into a robust solution for remote collaboration, online classes, and virtual meetings.