Brainstorming

Brainstorm & Idea Prioritization :

The ideation phase formed the backbone of the **Video Conference App Project**. It included identifying real-world issues in remote communication, listing out potential features, and then prioritizing them for development within our project constraints.

Goal of the Project

Create a **Video Conferencing Application using the MERN Stack (MongoDB, Express.js, React.js, Node.js)** that allows users to:

- Host and join real-time video meetings with ease
- Share screens, chat, and record sessions
- Securely schedule and manage meetings
- Access a clean, responsive, and performance-optimized interface

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Objective:

To bring together individuals with diverse skill sets, align on a shared vision, and finalize a real-world problem to address using the MERN stack.

Actions Taken:

Team Formation: A balanced team was formed consisting of frontend developers, backend developers, and database designers—all passionate about solving real-life problems through web development.

Skill Mapping

- Frontend: React.js, Bootstrap, HTML/CSS
- Backend: Node.js, Express.js
- Database: MongoDB

Collaboration Tools Used:

- Communication: WhatsApp, Google Meet
- Project Planning: Trello, Google Docs

Problem Identification:

We identified a recurring problem with complex or unreliable video meeting tools, especially for users seeking lightweight, easy-to-use platforms for quick team catch-ups or virtual classes.

1. Team Formation and Role Distribution

The foundation of any successful project lies in assembling a well-balanced and dedicated team. For our project titled **VIDEOCON– A Video Conference Application Using MERN Stack**, we began by identifying team members with strong enthusiasm for web development, practical problem-solving, and interest in building a real-world application from scratch.

Our team comprises four committed members. Each member was assigned a specific role based on their individual skill set, domain knowledge, and preferences. This structured approach ensured a clear division of responsibilities, accountability, and efficient collaboration throughout the development lifecycle.

Team Member	**Assigned Role**	**Key Responsibilities**
Member 1	**Project Lead & Full Stack Developer**	Coordinating project timelines, managing GitHub repository, integrating frontend and backend, deployment.

Member 2	**Frontend Developer**	Designing responsive user interfaces using React.js, managing UI/UX flow, integrating API data, using Tailwind CSS for styling
Member 3	**Backend Developer**	Developing RESTful APIs using Node.js and Express.js, implementing authentication and authorization (JWT), business logic
Member 4	**Database Manager & QA Tester**	Designing MongoDB database schemas, managing data models, writing queries, performing manual testing and reporting bugs

We adopted a collaborative model that allowed flexibility and learning opportunities. Members often cross-collaborated across modules to support one another, strengthening teamwork and overall output.

2. Collaboration Tools and Workflow

To ensure seamless communication and progress tracking, especially in a hybrid working environment, we utilized a range of collaboration tools. These helped us stay aligned with project goals, deadlines, and updates:

Git & GitHub: For version control, collaborative code reviews, branching strategies, and pull requests.

Google Meet / Whatsapp : For regular virtual meetings, daily/weekly stand-ups, and technical discussions.

Google Drive / Docs: For storing and sharing important project-related documents, reports, and references.

We followed a simplified agile approach with iterative development and continuous integration. Weekly check-ins helped us identify blockers early, track progress, and maintain momentum.

3. Brainstorming and Problem Selection

Our initial goal was to identify a real-world problem that could be solved effectively using the MERN stack. We conducted several brainstorming sessions where each team member proposed different ideas from areas such as:

- Online collaboration and communication platforms
- Virtual classrooms and education tools
- Healthcare appointment and consultation systems
- Event planning and coordination apps
- Remote work productivity and time-tracking tools

After evaluating these ideas based on **feasibility, scope, technical complexity, innovation, and real-world relevance**, we collectively agreed upon building a **Video Conference App** as the most promising and impactful project.

4. Why We Chose This Problem

The decision to work on a **Video Conference Application** was driven by the following key factors:

High Practical Relevance: In the current digital age—especially post-pandemic—video conferencing has become an essential mode of communication for remote teams, educators, students, freelancers, and professionals. Our app aims to support seamless virtual meetings without the complexities or pricing models of existing heavy platforms.

Market Gap: While there are major players like Zoom, Google Meet, and Microsoft Teams, many users (especially small teams, educational groups, and startups) seek simpler, lightweight, and more customizable video conferencing tools. Our solution targets that underserved segment by offering a user-friendly, web-based platform with real-time communication and collaborative features.

Technology Fit: The project was an excellent match for the MERN stack as it involves:

- Real-time communication using **WebRTC and Socket.IO**
- User authentication and session management
- CRUD operations for managing meetings and user profiles
- RESTful API development
- MongoDB-based data storage for users, meetings, and chat logs

Scalability and Innovation Potential: The application holds great potential for future expansion and innovative features, such as:

- Screen sharing and recording capabilities
- Collaborative whiteboard or note-taking tools

- Meeting scheduling with calendar integration
- Breakout rooms and group chat
- AI-based noise suppression or background blur
- Mobile app version using React Native

5. Final Problem Statement

"To design and develop a responsive, secure, and scalable web application using the MERN stack that enables users to host and join real-time video conferences, with features such as user authentication, meeting scheduling, and live chat—thereby providing a reliable and accessible platform for remote communication and collaboration."

This problem not only allowed us to apply our technical knowledge in full-stack development and real-time technologies, but also inspired us to build a practical solution that can be deployed and scaled in real-world educational, corporate, and personal use cases.

Step-2: Brainstorm, Idea Listing and Grouping

Once the team was formed and the problem statement was finalized, the next crucial step in the project development process was to brainstorm different ideas related to the solution, list those ideas collaboratively, and group them into meaningful categories. This step helped us visualize the complete scope of the application, prioritize features, and streamline the development process.

1. Purpose of the Brainstorming Session

The goal of the brainstorming session was to:

- Understand what functionalities users would expect from a VideoCon platform.

- Identify all possible features that can enhance user experience and platform efficiency.

- Break down the problem into smaller manageable modules for better planning and development.

- Align everyone's vision regarding the project scope and deliverables.

2. Brainstorming Methodology

We conducted a series of brainstorming sessions over virtual meetings using tools like:

- **Miro** for real-time collaborative whiteboarding and mind mapping.

- **Google Docs** for live note-taking and idea capturing.

- **Trello** for organizing features into categories and prioritizing tasks.

Each team member contributed ideas based on:

- Personal experiences with tools like Zoom, Google Meet, Microsoft Teams, etc.

- Observations of pain points users face during online meetings.

- Feedback and expectations from students, professionals, and educators who rely on video conferencing.

We used a **Round-Robin format** where every team member was encouraged to present one or more ideas at a time. No idea was considered irrelevant or dismissed immediately — everything was listed for review.

3. Idea Listing

Below is the comprehensive list of ideas/features that emerged during the brainstorming phase:

Core Features:

- [ ] User registration/login (with roles: host, participant)
- [ ] Create and schedule video meetings
- [ ] Join meetings via link or meeting ID
- [ ] Real-time video and audio streaming
- [ ] Screen sharing capability
- [ ] Chat functionality during meetings
- [ ] Mute/unmute and video on/off controls
- [ ] Meeting recording (cloud/local)
- [ ] Responsive user dashboard for managing meetings and joining sessions

Extended Features:

- [ ] Calendar integration (Google Calendar, Outlook) for scheduling

- [ ] Email and in-app notifications for upcoming meetings
- [ ] Host controls (remove participant, mute all, lock meeting)
- [ ] Meeting waiting room and password protection
- [ ] Custom meeting backgrounds or blur feature
- [ ] Participant reactions (raise hand, emojis, etc.)

Future Enhancement Ideas:

- [ ] Breakout rooms for group discussions
- [ ] AI-based noise cancellation
- [ ] Integration with cloud storage platforms (Google Drive, Dropbox) for saving recordings
- [ ] Analytics dashboard for admins (meeting duration, participant count, etc.)
- [ ] Multi-language support and live captioning
- [ ] Mobile app version using React Native for on-the-go access

4. Grouping of Ideas

After listing all possible ideas, we organized them into logical **feature groups** for better clarity and to ease project execution. This step was critical for defining the Minimum Viable Product (MVP) and planning development sprints.

Category	Features Grouped
User Management	Registration, Login, Logout, Profile Management, Role Management (Host/Participant)
Meeting Management	Create Meeting, Join via Link or ID, Schedule Meeting, Dashboard for Meeting Control
Communication	Video/Audio Streaming, Screen Sharing, Chat, Mute/Unmute, Reactions
Security	JWT Authentication, Password Encryption, Meeting Lock, Waiting Room, Role-based Access
Notifications	Email Alerts, In-App Notifications, Calendar Reminders
Admin Operations	User Monitoring, Meeting Logs, Access Control, Reporting Tools
Advanced Features	Meeting Recording, Background Effects, Breakout Rooms, Live Captions, Analytics Dashboard, Mobile App (Future)

5. Key Takeaways from This Phase

We successfully transformed a broad problem statement into well-defined, actionable modules.

- **Prioritized core functionalities** that are essential for the initial version of the video conference application, ensuring a seamless user experience.
- **Documented advanced features** for future versions, such as breakout rooms, live captions, and integration with third-party tools.

• Ensured everyone in the team had a shared understanding of the project's goals and how we would break down tasks across upcoming development sprints.
• Established a strong foundation for **Requirement Analysis**, **System Architecture Design**, and **Sprint Planning** for the next phases of development.

Step-3: Idea Prioritization

After brainstorming and organizing a comprehensive list of ideas and features in Step 2, the next logical step in our development process was **prioritizing these ideas**. This stage was essential to identify which features to develop first (Minimum Viable Product – MVP), which ones to add later (Post-MVP or V2), and which ones to consider as long-term enhancements or stretch goals.

- Effective idea prioritization ensured that:
- The project remained manageable within the given timeline and resources.
- Core user needs were addressed from the start.
- The development process followed a clear, goal-driven roadmap.

1. Prioritization Approach

To determine what features should be prioritized for development, we adopted a combination of two popular techniques:

a. MoSCoW Method

We classified each idea into four categories:

Must Have – Critical features required for the app to function.

Should Have – Important features that enhance user experience but are not critical for MVP.

Could Have – Nice-to-have features that can be added if time/resources allow.

Won't Have (for now) – Features we decided to postpone or not include in this version.

b. Value vs. Effort Matrix

Each feature was analyzed based on:

Value to the end-users (usability, necessity, impact)

Effort required to implement (time, complexity, team skill)

This allowed us to balance our work between quick wins, high-impact features, and manageable complexities.

2. Prioritized Feature List (with Justifications)

Feature	**Priority**	**Justification**
User Registration and Login	Must Have	Basic entry point for all users; essential for access control
JWT Authentication	Must Have	Secures APIs and protects user data
Video Conference Room Creation	Must Have	Core feature; allows users to create and host virtual meetings
oin Conference (with Link)	Must Have	Core feature for participants to join meetings using a unique URL/link

Audio/Video Streaming	Must Have	Essential for the main purpose of the app — enabling real-time communication
Chat Functionality	Must Have	Facilitates real-time text communication alongside video/audio
Notifications (Meeting Invites)	Must Have	Ensures users receive reminders/notifications for scheduled meetings
Admin Dashboard	Should Have	Helps manage and monitor meetings, users, and app settings
Breakout Rooms)	Should Have	Useful for larger meetings or team collaborations within a single conference (future feature)
Meeting Recording	Could Have	Adds value for future versions; useful for later reference
Multi-Factor Authentication	Could Have	Enhances security for real-world applications, but not essential for MVP

Calendar Integration	Could Have	AddSyncs with user calendars to schedule meetings more easily (good for future versions)s extra security; useful in real-world scenarios
Custom Branding	Won't Have	Requires additional design work; not within the scope of the initial version
Payment Gateway Integration	Won't Have	Not needed for the MVP; requires time and effort to implement within the current timeline

3. Final MVP Feature Set

Based on the prioritization, the following features were locked in as MVP (Minimum Viable Product) for the Video Conference App:

- **User Registration & Login (Host/Participant)**
  Allows users to create an account, log in, and manage their profiles.

- **JWT Authentication**
  Secures API endpoints and ensures user authentication for private meetings.

- **Real-time Video/Audio Conferencing**
  Core functionality that enables seamless video and audio communication between users.

- **Meeting Scheduling & Notifications**
  Users can schedule meetings, and automated notifications are sent to participants.

- **Screen Sharing**
  Allows users to share their screen during meetings for collaboration.

- **Chat Functionality**
  Real-time messaging during meetings for communication apart from video/audio.

- **Participant Management**
  Features like muting/unmuting, removing participants, and managing host controls.

4. Post-MVP Planning

We also created a Phase 2 backlog to record "Should Have" and "Could Have" features, which could be implemented after the initial launch if time permits. This list was added to Trello for tracking and sprint planning purposes. These features include:

- **Admin Panel**
  Allows administrators to monitor meetings, manage users, and resolve issues.

- **Recording Functionality**
  Enables meeting recording for later reference and sharing.

- **Breakout Rooms**
  Facilitates smaller group discussions within larger meetings.

- **Meeting Transcriptions**
  Provides automatic transcriptions of meetings for reference and accessibility.

- **Mobile App Version**
  Expands access to the platform on mobile devices using React Native or Flutter.

5. Outcome of the Prioritization Phase

- We defined a clear roadmap that separates core functionalities from auxiliary features for the **Video Conference App**.

- The team could now focus on delivering the most impactful and feasible features, such as user authentication, real-time video/audio conferencing, and screen sharing.

- This process helped manage scope and timeline, ensuring that we avoid overengineering and feature creep, keeping the initial launch on track.

- Set the stage for the next phases, including requirement analysis, UI wireframing, and architecture design, which will define the app's user interface and underlying infrastructure.