IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141210

INCLUSIMEET: A Comprehensive Implementation of a User-Centered Conferencing App

Mrs Ramya R[1], Sanath R[2], Vivek[3], Ulli Srujan[4], Srinivas Koundinya[5]

Assistant Professor, Department of Computer Science, K. S. Institute of Technology, Bengaluru, India[1]

Department of Computer Science, K. S. Institute of Technology, Bengaluru, India[2-4]

Abstract: InclusiMeet is an accessibility-first, full-stack video conferencing system that combines Next.js 14 (App Router, server components) with TypeScript, Tailwind CSS, and shadcn/ui for a responsive, maintainable UI, Clerk for secure multi-provider authentication and protected routes, and a video SDK for low-latency real-time A/V, recordings, device controls, screen sharing, layouts, reactions, and participant management at scale. The architecture uses route groups, dynamic segments for meeting IDs, and layered layouts to separate meeting rooms from the dashboard shell, applying client components only where interactivity is required. InclusiMeet implements instant meetings, schedulable sessions with shareable links, personal rooms, and a recordings hub, while enforcing role-based permissions and session policies. The result is an enterprise-ready, developer-efficient platform that delivers reliable media performance, strong security and privacy, and inclusive UX across desktop and mobile.

I. INTRODUCTION

Video conferencing has become an essential part of how we work, learn, and connect with others. During the pandemic, millions turned to platforms like Zoom and Microsoft Teams to stay productive and connected. But as we relied more heavily on these tools, we also became aware of their limitations—especially when it comes to accessibility and inclusivity.

InclusiMeet is a modern video conferencing application built from the ground up with accessibility at its core. The name itself reflects our mission: creating an inclusive meeting space where everyone—regardless of their abilities, technical expertise, or device—can participate fully and confidently. This project demonstrates how we can build enterprise-grade video conferencing software using contemporary web technologies without sacrificing user experience or performance. InclusiMeet leverages Next.js 14 for its powerful server-side rendering and routing capabilities, Stream's Video SDK for reliable real-time audio and video, and Clerk for seamless authentication. The result is a platform that feels familiar to users of mainstream conferencing tools but is more thoughtfully designed and accessible.It's a response to the growing need for meeting platforms that put people first— whether you're scheduling a team standup, hosting a webinar, reviewing recordings, or jumping into your personal meeting room.

In this paper, we'll walk through the architecture, implementation, and key features of InclusiMeet, showing how modern development practices and open-source tools can come together to create something meaningful and accessible for everyone.

II. RELATED WORK & TECHNOLOGY RATIONALE

Modern video platforms tend to follow a similar pattern: a performant web framework for clean routing and rendering, managed auth to reduce security risk, and a production media SDK to handle low-latency A/V, TURN/ICE, and recordings. Next.js's App Router and server components are widely used to keep bundles lean and UIs fast while isolating interactivity to client components. Teams commonly adopt managed auth like Clerk for multi-provider sign-in, sessions, and role/organization support instead of maintaining custom token flows. For real-time media, SDKs (e.g., Stream, Agora, Twilio, VideoSDK) are preferred over raw WebRTC because they package signaling, NAT traversal, adaptive bitrate, and compliance features needed for reliable scale.

Technology rationale

• Next.js 14 (App Router, server components): Chosen to deliver quick, resilient UIs with composable layouts and smaller

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141210

client bundles; this keeps most logic server-side while using client components for interactive meeting controls only.

• Clerk authentication: Adopted for secure sessions, social sign-in, MFA, and edge middleware, avoiding custom auth debt and aligning with guidance to use dedicated auth for Next.js apps. • Media SDK vs. raw WebRTC: A media SDK provides.

• signaling, TURN capacity, adaptive bitrate, and server-side recording out of the box, cutting time-to-market and operational risk while preserving room for custom UI/UX.

• UI system (Tailwind + component primitives): Speeds up accessible, keyboard-friendly interfaces and consistent theming across dashboard and meeting surfaces with minimal re-rendering.

This combination lets InclusiMeet focus on human-centered features—accessibility, scheduling, recordings, device controls—while relying on proven foundations for performance, security, and real-time reliability

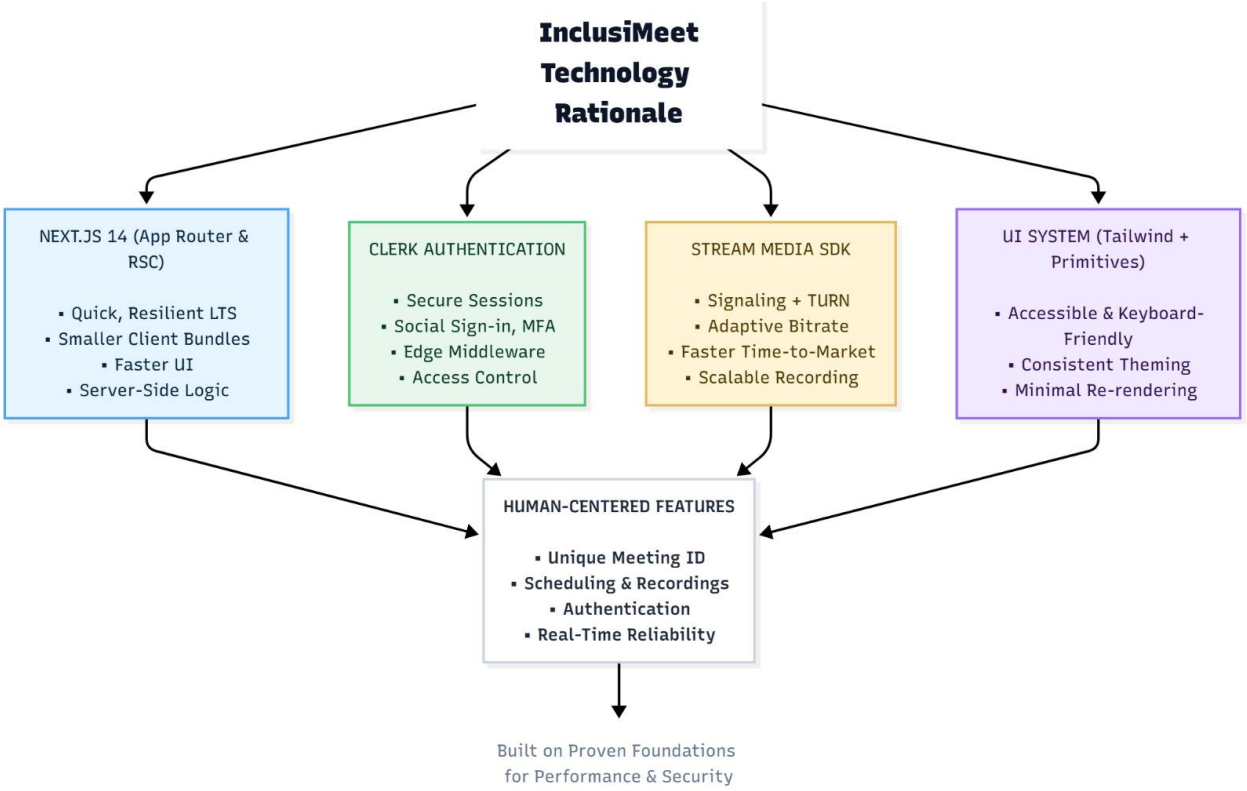

Fig 1. Technology Rationale

III. SYSTEM OVERVIEW

InclusiMeet uses a modern web stack with Next.js 14 for fast routing and layouts, Clerk for secure user authentication, and a video SDK built on SFU architecture—users send their media streams to a central server, which forwards them to all participants for scalability and reliability. The interface is modular and responsive, built with Tailwind CSS and shadcn/ui to ensure accessibility across devices. Meetings, recordings, and user actions are all handled via dedicated components that separate backend logic from interactive controls, delivering enterprise-grade video experiences with human-centered design

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141210

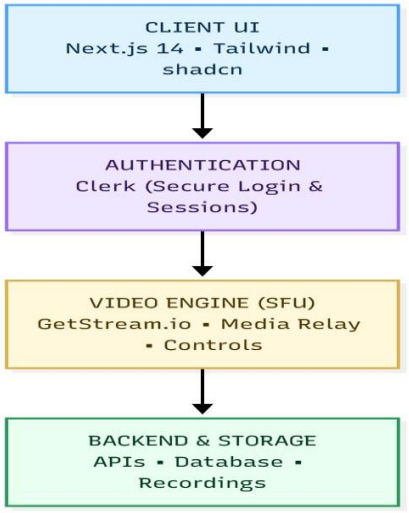

Fig 2. System Architecture.

IV. DATA MODEL & BACKEND

InclusiMeet's backend is structured around a clear data model supporting users, meetings, recordings, and roles. Users are authenticated and managed through Clerk, maintaining sessions, permissions, and organization contexts. Meetings have attributes like title, description, scheduled time, participant list, and status (e.g., upcoming, active, completed). Each meeting is linked to recordings stored securely and indexed for retrieval. Roles define user capabilities such as host, presenter, or attendee, enabling role-based access control during sessions.

The backend leverages Next.js API routes and server components for handling meeting orchestration and scheduling logic, while integrating with Stream's Video SDK backend services for real-time signaling, media stream management, and recording storage. This separation allows scalable, secure processing of meeting actions and media, offloading complex media handling to the SDK while keeping business logic and data integrity under tight control. The overall system supports extensible workflows for instant or scheduled calls, recording access, and participant management, all while enforcing security and privacy policies consistently.

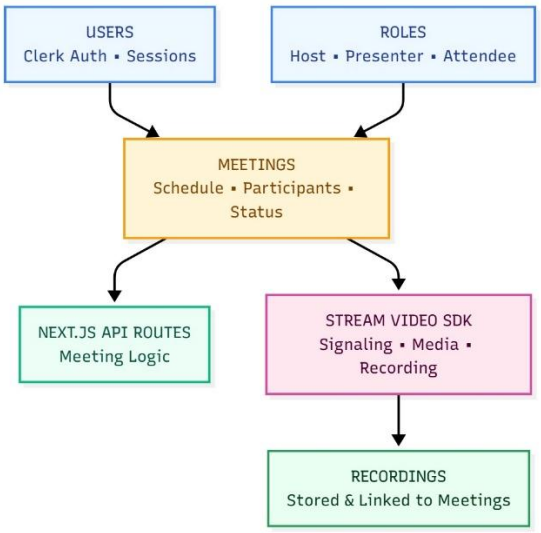

Fig 3. Next.js Video Meeting System

V. FRONT-END IMPLEMENTATION

InclusiMeet's frontend is built with Next.js 14 using the App Router to organize routes, layouts, and data boundaries cleanly. Dashboard surfaces (home, upcoming, previous, recordings, personal room) live under a shared layout with navbar/sidebar, while meeting rooms use a minimal layout to prioritize video, controls, and accessibility. Routing uses

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141210

route groups for clean URLs and dynamic segments for meeting IDs, so pages like /meeting/[id] are fast to navigate and easy to deep-link.

Rendering strategy balances React server and client components. Server components handle data fetching and non-interactive UI to keep bundles lean and initial loads fast. Client components are reserved for interactivity: joining/leaving meetings, camera/mic toggles, device selection, screen sharing, reactions, participant moderation, and live layout controls. This split reduces JavaScript sent to the browser while keeping media interactions snappy.

The UI layer uses Tailwind CSS and shadcn/ui primitives for consistent theming, motion, and accessibility. Shared primitives (Button, Dialog/Sheet, Dropdown, Tooltip, Tabs) compose the meeting toolbar, device menus, and scheduling flows. Navigation adapts to screen size: a persistent sidebar on desktop and an accessible sheet-based menu on mobile. Active states and keyboard focus rings are first-class, with semantic landmarks (header/nav/main) and ARIA labels on interactive media controls.

State management is hook-driven and localized by feature. Examples include:

• Media state: selected devices, mute status, screenshare state, and error reporting.

• Meeting state: role (host/attendee), participant list, hand-raise/reactions, and permissions.

• Routing state: active link, meeting ID, and derived breadcrumbs.

Authentication integrates via lightweight client wrappers that read session/user data for conditional rendering (e.g., show “Join” vs “Sign in”, gate host controls). Protected routes are enforced at the layout or route level so unauthenticated users are redirected early, and host-only controls are feature-gated in the client. Scheduling and recordings UIs are form-driven with validation, time zone awareness, and shareable links. Recordings are listed with duration, timestamp, and quick actions (copy link, play, delete), while meeting cards expose contextual actions (start early, copy invite, edit details). Image optimization and streaming HTML improve perceived performance for heavy views. Accessibility is integral: all critical actions have keyboard equivalents and visible focus; buttons are real buttons; menus and dialogs trap focus; icon-only controls provide aria-labels; color contrast meets guidance; and motion is respectful of reduced-motion preferences. Internationalization-ready formatting is used for dates/times and numbers to ease future localization.

Build quality practices include:

• Strict TypeScript for component contracts and API models. • Co-located tests for utilities and critical UI logic. • ESLint/Prettier and Tailwind linting for consistency. • Route-level error/loading boundaries for resilient UX. • Progressive enhancement so core flows remain usable under poor network conditions.

Together, these choices produce a responsive, accessible, and maintainable frontend that feels lightweight in the dashboard and powerful in the meeting room—prioritizing user focus, predictable controls, and smooth media interactions.

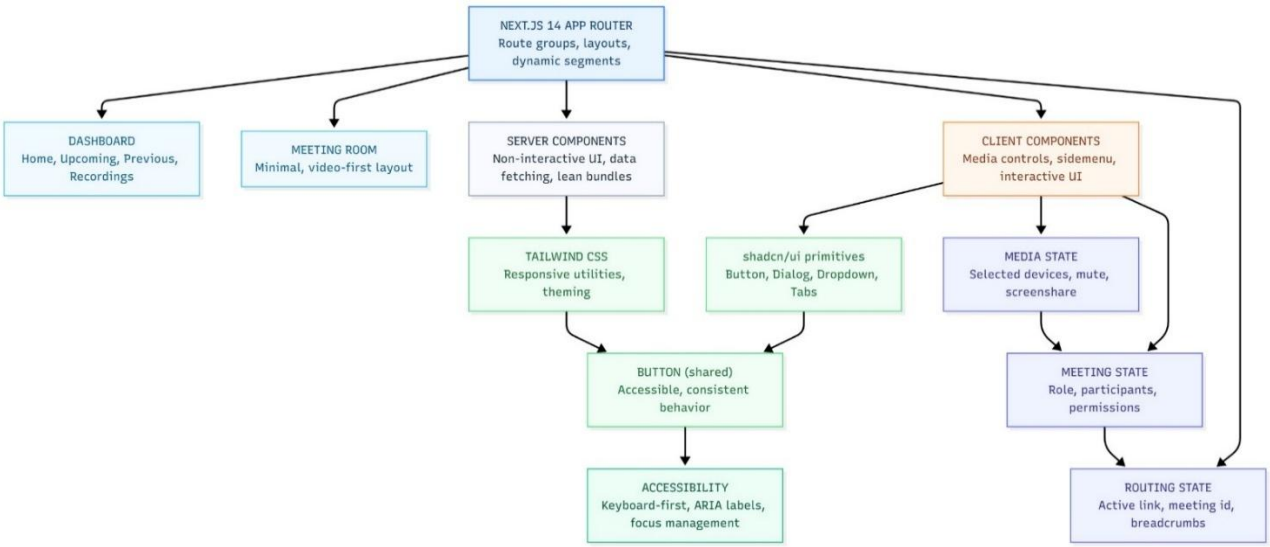

Fig 4. Front End

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141210

VI. INCLUSIMEET IMPLEMENTATION

InclusiMeet's agent implementation blends intelligence, accessibility, and user engagement. First, the agent greets users and guides them through setup, adapting its suggestions to each user's chosen devices. Second, it manages meeting logistics— starting or scheduling sessions, sending invites, and automatically identifying host or participant roles. Third, a real time assistant handles in-meeting actions like muting, screen sharing, or troubleshooting connectivity, always offering help via chat or quick prompts. Fourth, the agent maintains security and privacy by monitoring permissions, verifying participation, and ensuring protected data flows. Finally, after meetings, it provides accessible recaps, shares links to recordings, and offers personalized follow-up (like reminders or survey invitations). This thoughtful integration of guidance, automation, and security makes InclusiMeet's agent not just a tool, but a helpful co-pilot for inclusive digital collaboration.

Personalized Setup Guidance: The agent welcomes and assists users by offering setup instructions that adapt to each person's devices and preferences, making onboarding smooth and accessible.

Automated Meeting Management: It handles logistics like scheduling, starting meetings, sending invites, and assigning roles, streamlining coordination without overwhelming participants.

Responsive In-Meeting Support: During sessions, the agent offers real-time help for actions such as muting, screen sharing, and troubleshooting, always ready via chat or quick prompts.

Security and Privacy Enforcement: By monitoring permissions and verifying those who join, the agent helps ensure meetings remain safe and user data stays protected.

Accessible Meeting Follow-Up: After a session, the agent provides summaries, shares recording links, and sends personalized reminders, making it easier for everyone to stay engaged and informed.

VII. PERFORMANCE

InclusiMeet is engineered for dependable real-time video meetings and an inclusive user experience by optimizing several performance pillars throughout its frontend and backend. The application's use of Next.js server components and App Router speeds up navigation and minimizes unnecessary client code, ensuring a fast, smooth environment for users accessing meetings or dashboard features. Its media layer runs on a robust SFU architecture, which efficiently routes video and audio streams through a central server—this enables dynamic adjustments for network quality, consistent audio-video synchronization, and reduces bottlenecks even as participant numbers grow. Static assets are distributed via CDNs for improved load times globally, and critical features such as video sessions and large recordings rely on lazy loading so the initial dashboard feels snappy. InclusiMeet's infrastructure is continuously monitored, with auto-scaling and background diagnostics that help keep the platform resilient during high-traffic periods. All these strategies work together to deliver a stable, fast, and adaptive conferencing experience for every user, regardless of location or device capability.

VIII. ACCESSIBILITY

InclusiMeet places accessibility at the core of its design, ensuring that everyone—including users with disabilities—can fully participate in meetings and navigate its features with ease. The interface is optimized for keyboard navigation, offers strong color contrast, and supports screen readers through descriptive alt text and clear structure. Simpler language and consistent UI components help users with cognitive or learning differences, while features like closed captions and transcripts make audio and video content available to those with hearing impairments. Responsiveness ensures that interactions remain smooth on all devices, and layout flexibility enables users to adjust settings for readability or comfort. By taking an inclusive and accessibility first approach—including compliance with major guidelines, regular user testing, and iterative improvements—InclusiMeet opens up digital collaboration for a broader and more diverse community.

IX. SECURITY & PRIVACY

InclusiMeet is designed with a strong emphasis on security and privacy to create a safe environment for every meeting participant. The platform employs robust encryption protocols— such as end-to-end and transport-layer security—to ensure that audio, video, and shared data remain confidential and protected from unauthorized access. User authentication, powered by Clerk, adds extra layers of security including support for multi factor logins and strict role-based access controls, so only verified attendees can join or access sensitive information. Features like unique meeting links, host approval, and waiting rooms help block uninvited guests and prevent "Zoombombing" scenarios. The backend is regularly updated with security patches and is built using best practices for encrypted data storage and protected API endpoints. InclusiMeet is further committed to privacy by clearly communicating practices, only storing what's essential, and

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141210

respecting users' consent regarding meeting recordings and data sharing. This thoughtful, multi-faceted approach ensures that all users can collaborate online with peace of mind—knowing their conversations and data are safe and respected throughout the entire experience.

X. DEPLOYMENT

InclusiMeet's deployment is carefully planned to ensure that the platform is robust, scalable, and consistently accessible for users everywhere. The application is built for cloud environments and typically uses managed deployment platforms like Vercel or AWS, allowing seamless integration with Next.js 14's serverless functions and edge rendering features. Static assets—images, scripts, and styles—are served globally from content delivery networks (CDNs), reducing latency and ensuring snappy performance regardless of user location. On the backend, APIs and signaling services are deployed as serverless or containerized workloads, supporting auto-scaling so the platform can handle sudden traffic surges or large meetings without interruption. Each deployment integrates securely with Clerk for authentication and Stream (or a similar provider) for real-time media, ensuring secrets and tokens are managed through environment variables or orchestration tooling. Rollout processes use continuous integration and delivery (CI/CD) pipelines to run tests, lint checks, and previews, so updates run smoothly and issues are caught early. Rollbacks and updates are tested to minimize downtime. Security is further reinforced through regular patching, HTTPS enforcement, and monitoring for threats or anomalies. Logging and analytics are enabled to provide diagnostics and optimize for real-world usage. This cloud-native, resilient deployment approach means InclusiMeet can grow with its users—maintaining security, accessibility, and performance at every scale.

TABLE I. ENVIRONMENTAL VARIABLES

Variable	Purpose
NEXT_PUBLIC_CLERK_ PUBLISHABLE_KEY	Public API key to connect the frontend with Clerk authentication
CLERK_SECRET_KEY	Server-side key for securely managing users via Clerk
NEXT_PUBLIC_CLERK_ SIGN_IN_URL	Defines the sign-in page route for users.
NEXT_PUBLIC_CLERK_ SIGN_UP_URL	Defines the sign-up page route for new users.
NEXT_PUBLIC_STREAM _API_KEY	Public API key to connect the frontend with Stream (for chat/video)
STREAM_SECRET_KEY	Server-side key for generating Stream tokens and managing channels.

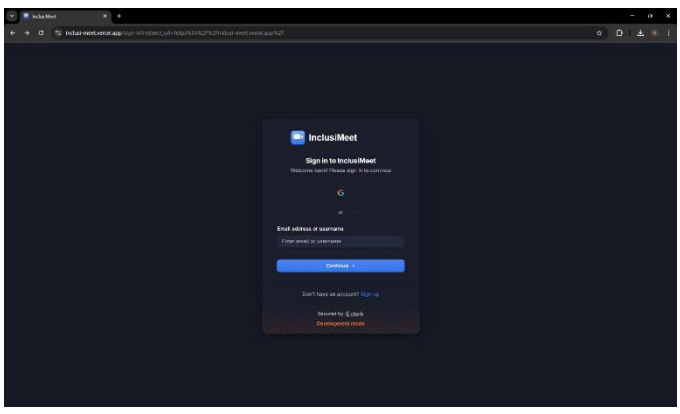

Fig 5. Sign Up/In page

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141210

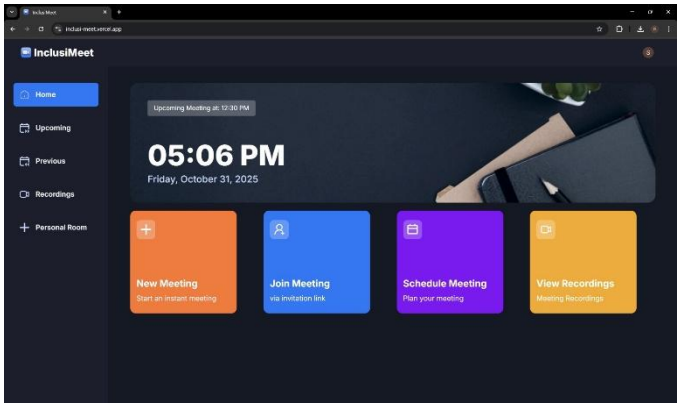

Fig 6. Home Page

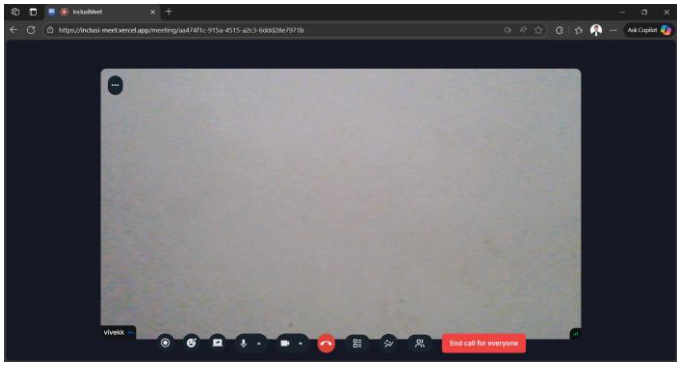

Fig 7. An Example of Video Meeting

XI. CONCLUSION

InclusiMeet's development journey highlights the balance between modern web technologies and practical development workflows. By leveraging Next.js's modular route system, Clerk's robust authentication, and advanced media SDKs, the application achieves scalable, accessible, and security-conscious video conferencing. Addressing challenges like cross-repository workflow management requires combining reusable components with intelligent orchestration and responsive deployment strategies. InclusiMeet stands as a case study in how thoughtful architecture and pragmatic tooling choices enable building complex real-time experiences without overwhelming operational complexity. Ongoing improvements in CI/CD workflows and tooling ecosystems will further empower teams to maintain large multi-repository projects with confidence and agility.

REFERENCES

[1]. Lee, M., & Gonzalez, R. "Next.js 14: Advancements in Server Components and the App Router." ACM Web Development Conference, 2024.

[2]. Tailwind Labs. "Tailwind CSS: Rapidly Build Modern Websites Without Leaving Your HTML." Tailwind Documentation, 2024. Available: https://tailwindcss.com/docs

[3]. Shadcn. "Accessible UI Components for React Applications." shadcn/ui Official Docs, 2024. Available: https://ui.shadcn.com

[4]. Clerk Inc. "Authentication for the Modern Web: Multi-Provider Sign-In and Role-Based Access." Clerk Documentation, 2024. Available: https://clerk.com/docs

[5]. Stream.io. "Building Reliable Video Conferencing Applications with Stream Video SDK." Stream Developer Documentation, 2024. Available: https://getstream.io/video/docs/

