SDM COLLEGE OF ENGINEERING & TECHNOLOGY, DHAVALAGIRI, DHARWAD – 580002

VII Semester Major Project – I Synopsis

On

"StreamSide - HD Video Conferencing Platform"

Submitted by

Sl. No	USN	Name	Signature
1	2SD22CS076	Rohit V Kulkarni	
2	2SD22CS110	Sujay R Bhakri	
3	2SD22CS118	Tejas V Sunagar	
4	2SD22CS126	Vishwanath N Konaraddi	

Under the Guidance of

Dr. Smitesh Patravali

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

September 2025

Reviewer(s) Report

Put a tick mark at the appropriate box to indicate the status of acceptance.

(Below table is for examiners/reviewers)

(The project guide's signature must be taken on the same page after this table to indicate approval of the synopsis.)

Put a tick mark at the appropriate box	**Indicate tick mark here (√ / x)**	Name and Signature of the **reviewer(s)**
Accepted the project proposal as it is.		
Accepted the project proposal with suggested modifications.		
Project proposal rejected		
Reviewer(s) Remarks		

Date:

Signature Of the
Project Guide(s)/ Supervisor(s)

Table of Contents

Chapter 1

Abstract

This report provides a comprehensive architectural analysis of Streamside, a modern, high-fidelity, and scalable real-time communication platform. The project's core vision is to address the inherent complexities of building and deploying robust WebRTC infrastructure by leveraging a decoupled, four-pillar architecture. Streamside's solution is constructed upon a curated stack of specialized technologies: LiveKit serves as the high-performance Selective Forwarding Unit (SFU) for the real-time media plane, managing all audio, video, and data streams; Next.js provides the application control plane, handling user interface rendering, server-side logic, and API orchestration; BetterAuth delivers a secure, component-driven identity and access management layer; and Prisma, in conjunction with a PostgreSQL database, establishes a type-safe data persistence layer. The strategic integration of these components enables Streamside to deliver a rich feature set, including multi-party video conferencing, secure user authentication, real-time text chat, and screen sharing, while ensuring maintainability, scalability, and a superior developer experience.

Chapter 2

Introduction

The digital landscape has undergone a big transformation, accelerating the demand for robust, low-latency, and interactive communication solutions across a multitude of sectors. The normalization of remote work, the expansion of telehealth services, the digitization of education, and the proliferation of interactive social platforms have elevated real-time audio and video from a niche feature to a foundational requirement. However, the development of such applications presents significant technical hurdles. Developers are often confronted with the complexities of managing network address translation (NAT) traversal, ensuring consistent media quality over variable and unreliable networks, and architecting systems capable of scaling to support thousands of concurrent users without degradation in performance. These challenges necessitate sophisticated infrastructure that can handle the intense demands of real-time media routing and session management.

Streamside is formally introduced as a modern platform engineered to abstract away these underlying complexities. The project's vision is to provide a seamless, secure, and feature-rich communication experience that rivals enterprise-grade solutions. The statement of purpose for Streamside is to empower users to access sophisticated real-time communication by leveraging a curated stack of best-in-class, open-source, and managed services. By offloading specialized functions to dedicated technologies

Chapter 3

Objectives

- **Objective 1: User Authentication and Profile Management:** Implement a robust, full-stack user authentication and profile management system that securely handles registration, login (including social), and profile updates within the application.

- **Objective 2: Route and Server Action Protection:** Secure all sensitive application routes and server-side operations to ensure that only authenticated and authorized users can access the protected dashboard and initiate server actions.

- **Objective 3: Multi-Party Video/Audio Conferencing:** Develop a fully functional, multi-party video conferencing solution that allows authenticated users to create or join dynamic, ephemeral rooms and seamlessly exchange real-time video and audio streams.

- **Objective 4: Integrating Collaboration Features:** Integrate and implement supporting real-time collaboration features, including text chat and screen sharing, to enhance the communication capabilities within the video rooms.

Chapter 4

Literature Review

Analysis of Established Platforms

Commercial platforms like Zoom and Google Meet serve as benchmarks for system design. Zoom's architecture combines SFU for media routing with cloud services (AWS/Oracle) for scheduling and participant management, supporting features like recording and breakout rooms while scaling to millions of daily users. Its distributed setup handles 1-to-1 and group calls efficiently, but relies on proprietary optimizations for low-latency in variable networks. Similarly, Google Meet's Android app design integrates WebRTC with Firebase for signaling and real-time databases, emphasizing mobile-friendly UI and AI-enhanced features like noise cancellation. User studies on Meet highlight intuitive controls but critique inconsistencies in terminology and hybrid meeting support.

Identified Gaps and Limitations

Despite advancements, several gaps persist in current technologies. Performance evaluations reveal WebRTC's vulnerability to network variability, leading to poor quality in wireless or high-latency scenarios, with calls for better adaptive bitrate algorithms. Security and privacy concerns, including data breaches and unauthorized access, are exacerbated by reliance on third-party services, necessitating stronger end-to-end encryption and token-based authentication. User-centric issues like "meeting fatigue" from prolonged screen time, lack of personal interaction, and cultural/language barriers highlight the need for more immersive features, such as AI-driven enhancements. Scalability remains a challenge for open-source platforms, with many struggling beyond small groups without distributed meshes. Additionally, developer friction in integrating disparate tools (e.g., media servers with auth systems) often results in brittle architectures.

Trends point toward AI integration for smarter meetings (e.g., auto-framing, transcription) and hybrid solutions bridging physical and virtual spaces, but adoption is uneven due to technological dependence and equity issues.

Relevance to Streamside

Streamside addresses these gaps by adopting a four-pillar, agile methodology that decouples media (LiveKit SFU for scalable routing), control (Next.js for secure orchestration), identity (BetterAuth/Clerk for robust authentication), and persistence (Prisma/PostgreSQL for data integrity). This mitigates scalability and integration challenges seen in prior works, while incorporating features like multi-party streaming, real-time chat, and screen sharing to enhance collaboration. By leveraging open-source tools, it reduces dependency on proprietary systems like Zoom's, promoting maintainability and cost-effectiveness. Future extensions could incorporate AI to tackle fatigue and privacy, building on emerging trends.

Chapter 5

Methodology

The project will follow a structured, three-phase approach using an Agile, Decoupled Architecture. This ensures that the security and data foundations are solid before tackling the complex real-time media integration.

1. Phase I: Foundation and Data Persistence

This phase establishes the core application structure, secure user identity layer, and reliable data storage.

- **Application Environment:** Using Next.js and TypeScript to set up a hybrid application, utilizing Server Components for speed and Client Components for interactivity.
- **User Identity and Security:** Integrating Clerk components for all sign-up and login flows, and used clerkMiddleware to protect all core application routes and server functions.
- **Data Modeling:** Implementing Prisma with PostgreSQL, defining the database schema first and generating a type-safe client to ensure data consistency across the entire application code.

2. Phase II: Secure Control Plane and Token Orchestration

This is the critical phase that builds the secure bridge between the authenticated user and the specialized video server.

- **Room Access Logic:** A protected Next.js Server Action was created to manage the room creation/joining process, relying on Clerk to first verify the user's secure session.
- **Client Connection:** The client received this key and used the LiveKit SDK to authenticate and establish a direct WebSocket connection with the LiveKit media server.

3. Phase III: Real-Time Media and Feature Implementation

This final phase integrates all core communication features using the established secure connections.

- **Multi-Party Streaming:** Implementing the logic for users to publish and subscribe to media tracks, leveraging the LiveKit SFU Architecture for efficient, scalable video and audio routing.
- **Collaboration Features:** Integrating low-latency Real-Time Text Chat using LiveKit's built-in Data Channels, avoiding the need for a separate chat server.
- **Screen Sharing:** Adding the ability for participants to publish their screens, treating the screen share as an additional video track for efficient distribution by the SFU.
- **Final Interface:** Completing the client-side UI using React components to manage and display all video feeds, chat, and room controls.

Chapter 6

Expected Outcomes

- **User Authentication and Profile Management:**
  - The system must successfully demonstrate user registration and login using both email/password and at least one social login provider (e.g., Google).
  - Users must be able to update standard profile details (via Clerk) and custom application fields (persisted via Prisma/PostgreSQL).
- **Route and Server Action Protection:**
  - All protected routes (e.g., the dashboard) are verifiably inaccessible to unauthenticated users, who are automatically redirected to the login page.
  - The crucial Next.js Server Action for generating the LiveKit access token must securely reject all unauthenticated requests.
- **Multi-Party Video/Audio Conferencing:**
  - A Live Demonstration must successfully connect a minimum of three distinct clients to a single room, with seamless, simultaneous exchange of video and audio streams among all participants.
  - Network monitoring confirms that clients are authenticated to the LiveKit media server using the cryptographically verified JWT generated by the application backend.
- **Integrating Collaboration Features:**
  - Real-Time Text Chat is fully functional, with messages sent via LiveKit's Data Channels received instantly by all other participants.
  - Any participant can successfully publish their screen (Screen Sharing), and this dedicated video track is clearly received and rendered by all other participants in the room.

Chapter 7

Project Scope

Use Case Diagram

This diagram illustrates the main actors (e.g., User) and their interactions with the system's use cases, based on the core functionalities described in the project (e.g., authentication, room management, real-time communication).

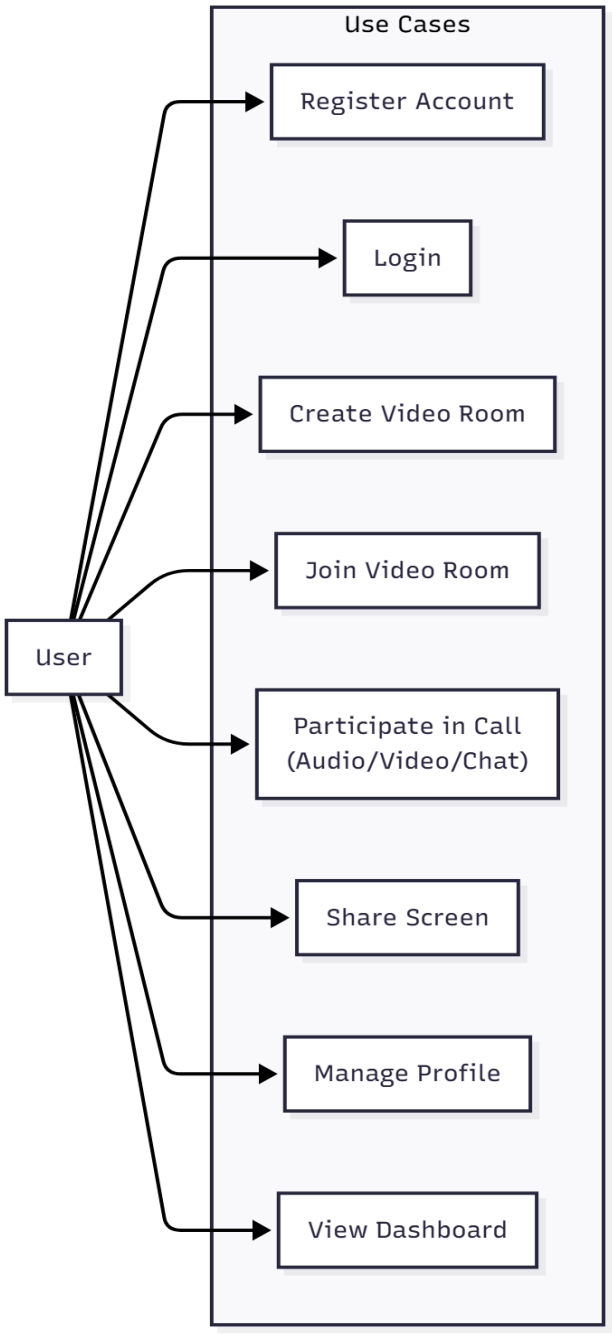

Figure 1: Use Case Diagram

Architecture Diagram

This diagram shows the high-level components and their interconnections in the four-pillar model (Media Plane, Control Plane, Identity Layer, Persistence Layer)

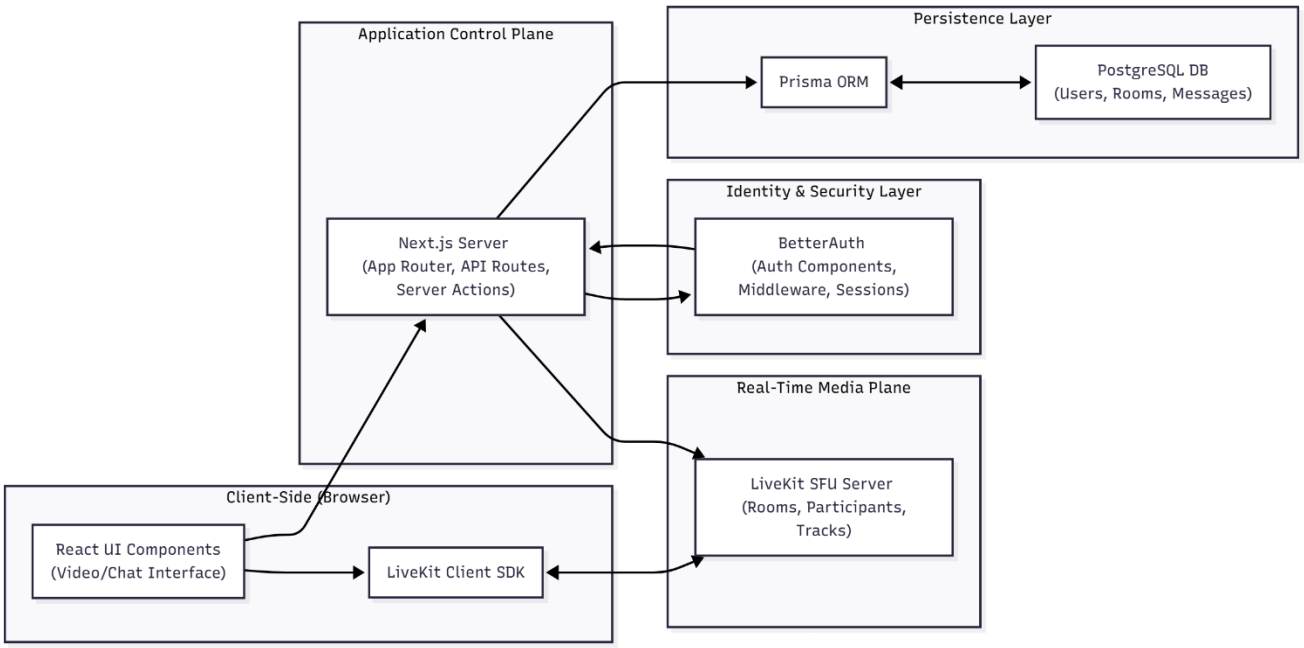

Figure 2: Architecture Diagram

Application Flow Chart

This flowchart maps the end-to-end user flow as described in section 4.1, including authentication, dashboard access, room joining, and real-time interaction.

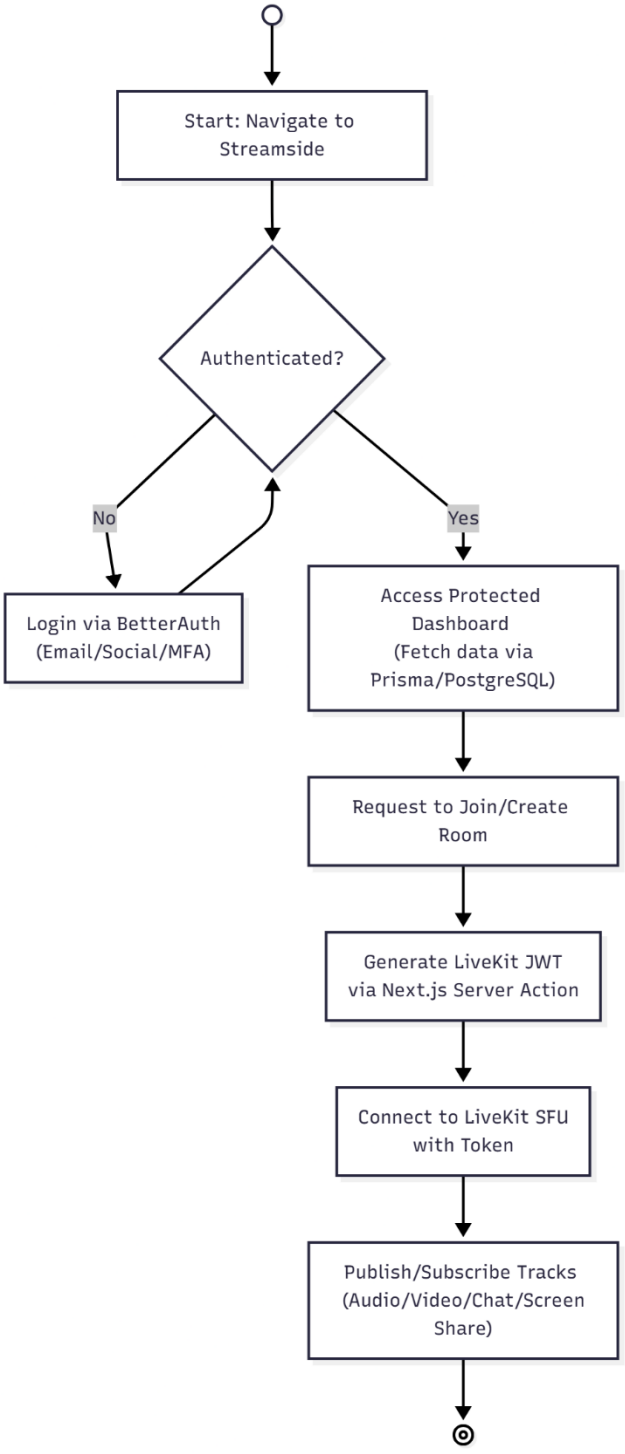

Figure 3: Flow Chart

Chapter 8

Resources Required

1. Software and Platform Tools

- **Development Environment:**
  - **Node.js (LTS):** Runtime environment for the entire stack.
  - **Visual Studio Code (or equivalent IDE):** Main code editor and debugging tool.
  - **Git** and **GitHub:** Version control and source code management.
- **Core Application Stack:**
  - **Next.js (App Router) & React:** Frontend rendering, routing, and backend Server Actions/APIs.
  - **TypeScript:** Programming language ensuring end-to-end type safety.
  - **Prisma (ORM):** Database client and schema migration tool.
- **Specialized Services (APIs):**
  - **Clerk (or BetterAuth):** Managed service for all authentication, session management, and middleware security.
  - **LiveKit Server/Cloud:** The WebRTC SFU (Selective Forwarding Unit) for all real-time media and data channel routing.
  - **PostgreSQL:** The underlying relational database for persistent application data.

2. Hardware and Infrastructure

- **Local Hardware:**
  - **Development Machine:** A modern computer (PC/Mac/Linux) capable of running the local development server.
  - **Peripherals:** A working Webcam and Microphone are mandatory for testing the core video/audio functionality.
- **Testing Resources:**
  - **Multiple Testing Clients:** Access to a minimum of three separate **devices/browsers** (e.g., desktop browser, incognito window, mobile device) to verify multi-party streaming and scalability.

- **Cloud Hosting:**
    - **Application Hosting (e.g., Vercel):** Required for deploying the Next.js control plane and user interface.
    - **Database Hosting (e.g., Vercel Postgres, Supabase):** A managed hosting service for the PostgreSQL database.

3. Other Requirements

- **API Keys/Secrets:** Required secret keys, environment variables, and credentials for LiveKit, Clerk, and the PostgreSQL database connection strings.
- **Documentation Access:** Reliable access to the official documentation for all major frameworks and services (Next.js, LiveKit, Clerk, Prisma) for reference.
- **Stable Network:** A reliable, high-speed internet connection for testing the low-latency requirements of real-time communication.

References

[1] "What is LiveKit: Development, Deployment & Integration Services ...," Sheerbit. [Online]. Available: https://sheerbit.com/introduction-to-livekit-what-is-it-and-why-use-it-for-webrtc/

[2] "LiveKit is modern, open source infrastructure for running WebRTC at scale," WebRTC Ventures. [Online]. Available: https://webrtc.ventures/partners/livekit/

[3] "livekit/livekit: End-to-end realtime stack for connecting humans and AI - GitHub," GitHub. [Online]. Available: https://github.com/livekit/livekit

[4] "Next.js Authentication - Best Auth Middleware for your Next app!," Clerk Documentation. [Online]. Available: https://www.better-auth.com/docs/integrations/next

[5] "LiveKit tutorials," LiveKit Tutorials. [Online]. Available: https://livekit-tutorials.openvidu.io/

[6] "Next.js by Vercel - The React Framework," Next.js Documentation. [Online]. Available: https://nextjs.org/

[7] "Static and Dynamic Rendering - App Router," Next.js Documentation. [Online]. Available: https://nextjs.org/learn/dashboard-app/static-and-dynamic-rendering

[8] "Next.js Database with Prisma | Next-Generation ORM for SQL Databases," Prisma Documentation. [Online]. Available: https://www.prisma.io/nextjs

[9] "LiveKit Cloud Architecture," LiveKit Documentation. [Online]. Available: https://docs.livekit.io/home/cloud/architecture/

[10] "livekit/agents: A powerful framework for building realtime voice AI agents 🎙️ - GitHub," GitHub. [Online]. Available: https://github.com/livekit/agents

[11] "SIP overview," LiveKit Documentation. [Online]. Available: https://docs.livekit.io/sip/

[12] Z. M. H. F. Wang, "System Design of Zoom," *GeeksforGeeks*, Oct. 6, 2025. [Online]. Available: https://www.geeksforgeeks.org/system-design/designing-zoom-system-design/

[13] Y. Komilo, "System design of Google Meet (Android App): A real-time communication architecture with technologies," *Medium*, Aug. 2, 2025. [Online]. Available: https://medium.com/@YodgorbekKomilo/system-design-of-google-meet-android-app-a-real-time-communication-architecture-with-technologies-ff4d53192e87